Last Time

- Dithering
- Signal Processing
Today

☐ Filtering
☐ Resampling
☐ Project 1 due October 30
A filter is something that attenuates or enhances particular frequencies

Easiest to visualize in the frequency domain, where filtering is defined as multiplication:

\[ H(\omega) = F(\omega) \times G(\omega) \]

Here, \( F \) is the spectrum of the function, \( G \) is the spectrum of the filter, and \( H \) is the filtered function. Multiplication is point-wise
Qualitative Filters

\[ F \times G = H \]

Input spectrum × Filter spectrum = Filtering result

Low-pass

High-pass

Band-pass
Low-Pass Filtered Image
High-Pass Filtered Image
Filtering in the Spatial Domain

- Filtering the spatial domain is achieved by convolution

\[ h(x) = f \otimes g = \int_{-\infty}^{\infty} f(u) g(x - u) \, du \]

- Qualitatively: Slide the filter to each position, \( x \), then sum up the function multiplied by the filter at that position
Convolution Example
Convolution Theorem

- Convolution in the spatial domain is the same as multiplication in the frequency domain
  - Take a function, $f$, and compute its Fourier transform, $F$
  - Take a filter, $g$, and compute its Fourier transform, $G$
  - Compute $H = F \times G$
  - Take the inverse Fourier transform of $H$, to get $h$
  - Then $h = f \otimes g$

- Multiplication in the spatial domain is the same as convolution in the frequency domain
Filtering Images

\[ I_{output}[x][y] = \sum_{i=-k/2}^{k/2} \sum_{j=-k/2}^{k/2} I_{input}[x+i][y+j]M[i+k/2][j+k/2] \]

- Work in the discrete spatial domain
- Convert the filter into a matrix, the \textit{filter mask}
- Move the matrix over each point in the image, multiply the entries by the pixels below, then sum
Handling Boundaries

\[ I_{output}[x][y] = \sum_{i=-k/2}^{k/2} \sum_{j=-k/2}^{k/2} I_{input}[x+i][y+j]M[i+k/2][j+k/2] \]

- At (0,0) for instance, you might need pixel data for (-1,-1), which doesn’t exist
- Option 1: Make the output image smaller - don’t evaluate pixels you don’t have all the input for
- Option 2: Replicate the edge pixels
  - Equivalent to: posn = x + i; if (posn < 0) posn = 0; and so on for other indices
- Option 3: Reflect image about edge
  - Equivalent to: posn = x + i; if (posn < 0) posn = -posn; and similar for others
Box Filter

- Box filters smooth by averaging neighbors
- In frequency domain, keeps low frequencies and attenuates (reduces) high frequencies, so clearly a low-pass filter
Bartlett Filter

- Triangle shaped filter in spatial domain
- In frequency domain, product of two box filters, so attenuates high frequencies more than a box

Spatial: Triangle (Box⊗Box)  
Frequency: $\text{sinc}^2$
Constructing Masks: 1D

- Sample the filter, then normalize
- eg 1D Bartlett

- Can go to edge of pixel or middle of next: results are slightly different
Constructing Masks: 2D

- Multiply 2 1D masks together using *outer product*

\[ M = mm^T, \text{ or } M[i][j] = m[i]m[j] \]

- \( M \) is 2D mask, \( m \) is 1D mask

\[
\begin{array}{c|c|c|c}
0.2 & 0.6 & 0.2 \\
0.2 & 0.04 & 0.12 & 0.04 \\
0.6 & 0.12 & 0.36 & 0.12 \\
0.2 & 0.04 & 0.12 & 0.04 \\
\end{array}
\]
Gaussian Filter

- Attenuates high frequencies even further
- In 2d, rotationally symmetric, so fewer artifacts

\[
\frac{1}{256} \begin{bmatrix}
1 & 4 & 6 & 4 & 1 \\
4 & 16 & 24 & 16 & 4 \\
6 & 24 & 36 & 24 & 6 \\
4 & 16 & 24 & 16 & 4 \\
1 & 4 & 6 & 4 & 1
\end{bmatrix}
\]
Constructing Gaussian Mask

- Use the binomial coefficients
- Central Limit Theorem (probability) says that with more samples, binomial converges to Gaussian

\[
\begin{array}{c|cc}
\frac{1}{4} & 1 & 2 \\
\frac{1}{16} & 1 & 4 & 6 & 4 & 1 \\
\frac{1}{64} & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
\end{array}
\]
Constructing Gaussian Mask

- Use the binomial coefficients
- Central Limit Theorem (probability) says that with more samples, binomial converges to Gaussian
High-Pass Filters

- A high-pass filter can be obtained from a low-pass filter
  - If we subtract the smoothed image from the original, we must be subtracting out the low frequencies
  - What remains must contain only the high frequencies
- High-pass masks come from matrix subtraction:
- eg: 3x3 Bartlett

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
- \frac{1}{16}
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{bmatrix}
= \frac{1}{16}
\begin{bmatrix}
-1 & -2 & -1 \\
-2 & 12 & -2 \\
-1 & -2 & -1 \\
\end{bmatrix}
\]
Edge Enhancement

- High-pass filters give high values at edges, low values in smooth regions
- Adding high frequencies back into the image enhances edges
- One approach:
  - \( \text{Image} = \text{Image} + [\text{Image} - \text{smooth(Image)}] \)

  \[
  \begin{array}{c}
  \text{Low-pass} \\
  \hline
  \text{High-pass}
  \end{array}
  \]
Edge Enhancement

Figure 6.32. (a) Original image. (b) Blurred image. (c) Difference between first two. (d) Enhanced image.
Fixing Negative Values

- The negative values in high-pass filters can lead to negative image values
  - Most image formats don’t support this

- Solutions:
  - Truncate: Chop off values below min or above max
  - Offset: Add a constant to move the min value to 0
  - Re-scale: Rescale the image values to fill the range (0,max)
Filtering and Color

- To filter a color image, simply filter each of R, G and B separately.

- Re-scaling and truncating are more difficult to implement:
  - Adjusting each channel separately may change color significantly.
Today

☐ Filtering
☐ Resampling
☐ Project 1 due October 30
Resampling

- Making an image larger is like sampling the original function at higher density
  - You need more pixels to represent the same thing, so a higher pixel density

- Reducing an image in size is like sampling at lower density

- Generating new samples of the “same” function is called resampling
  - In theory, 2 steps: Reconstruction and sampling - but not in practice

- Many other image manipulation tasks require resampling
General Scenario

- You are trying to create a new image of some form, and you need data from a particular place in the existing image
  - Always: Figure out where the new sample comes from in the original image
Resampling at a Point

- We want to reconstruct the original "function" at the required point
- We will use information from around the point to do this
- We do it using a filter
- Which filter?
  - We’ll look at Bartlett (triangular)
  - Other filters also work

Use these to reconstruct
Resampling at a Point

- Place a Bartlett filter at the required point
- Multiply the value of the neighbor by the filter at that point, and add them
  - Convolution with discrete samples
- The filter size is a parameter

- Say the filter is size 3, and you need the value at $x=5.75$
  - You need the image samples, $I(x)$, from $x=5$, $x=6$ and $x=7$
  - You need the filter value, $H(s)$, at $s=-0.75$, $s=0.25$ and $s=1.25$
  - Compute: $I(5)H(-0.75)+I(6)H(0.25)+I(7)H(1.25)$
Consider the Bartlett in 1D:

$$H_w(s) = \frac{2}{w} \left( 1 - \frac{2|s|}{w} \right)$$

To apply it at a point $x_c$ and find the contribution from point $x$ where the image has value $I(x)$

$$f(x) = \frac{2}{w} \left( 1 - \frac{2|x - x_c|}{w} \right) I(x)$$

Extends naturally to 2D:

$$f(x, y) = \frac{4}{w^2} \left( 1 - \frac{2|x - x_c|}{w} \right) \left( 1 - \frac{2|y - y_c|}{w} \right) I(x, y)$$
Common Operations

- **Image scaling by a factor** $k$ (e.g. 0.5 = half size):
  - To get $x_{\text{orig}}$ given $x_{\text{new}}$, divide by $k$:
    \[
    \begin{pmatrix}
    x_{\text{orig}} \\
    y_{\text{orig}}
    \end{pmatrix} = \left( \frac{x_{\text{new}}}{k}, \frac{y_{\text{new}}}{k} \right)
    \]

- **Image rotation by an angle** $\theta$:
  \[
  x_{\text{orig}} = \cos(-\theta)x_{\text{new}} - \sin(-\theta)y_{\text{new}}
  \\
  y_{\text{orig}} = \sin(-\theta)x_{\text{new}} + \cos(-\theta)y_{\text{new}}
  \]
  - This rotates around the bottom left corner. It’s up to you to figure out how to rotate about the center.
  - Be careful of radians vs. degrees: all C++ standard math functions take radians, but OpenGL functions take degrees.
Ideal Image Resize

- To do ideal image resampling, we would reconstruct the original function based on the samples
- A requirement for \textit{perfect} enlargement or size reduction
  - Almost never possible in practice, and we’ll see why
An Reconstruction Example

- Say you have a sine function of a particular frequency
- And you sample it too sparsely
- You could draw a different sine curve through the samples
Some Intuition

- To reconstruct a function, you need to reconstruct every frequency component that’s in it
  - This is in the frequency domain, but that’s because it’s easy to talk about “components” of the function
- But we’ve just seen that to accurately reconstruct high frequencies, you need more samples
- The effect on the previous slide is called aliasing
  - The correct frequency is aliased by the longer wavelength curve
Nyquist Frequency

- Aliasing cannot happen if you sample at a frequency that is twice the original frequency - the Nyquist sampling limit
  - You cannot accurately reconstruct a signal that was sampled below its Nyquist frequency - you do not have the information
  - There is no point sampling at higher frequency - you do not gain extra information

- Signals that are not bandlimited cannot be accurately sampled and reconstructed
  - They would require an infinite sampling frequency
Sampling in the spatial domain is like multiplying by a spike function.

You take some ideal function and get data for a regular grid of points.
Sampling in Frequency Domain

Sampling in the frequency domain is like convolving with a spike function:
- Follows from the convolution theory: multiplication in spatial equals convolution in frequency
- Spatial spike function in the frequency domain is also the spike function

![Diagram showing original signal and sampled signal]
Reconstruction (Frequency Domain)

- To reconstruct, we must restore the original spectrum
- That can be done by multiplying by a square pulse

![Diagram showing sample and original signals]
Reconstruction (Spatial Domain)

- Multiplying by a square pulse in the frequency domain is the same as convolving with a sinc function in the spatial domain.
Aliasing Due to Under-sampling

- If the sampling rate is too low, high frequencies get reconstructed as lower frequencies.

- High frequencies from one copy get added to low frequencies from another.
More Aliasing

- Poor reconstruction also results in aliasing
- Consider a signal reconstructed with a box filter in the spatial domain (square box pixels, which means using a sinc in the frequency domain):
Aliasing in Practice

- We have two types of aliasing:
  - Aliasing due to insufficient sampling frequency
  - Aliasing due to poor reconstruction

- You have some control over reconstruction
  - If resizing, for instance, use an approximation to the sinc function to reconstruct (instead of Bartlett, as we used last time)
  - Gaussian is closer to sinc than Bartlett
  - But note that sinc function goes on forever (infinite support), which is inefficient to evaluate

- You have some control over sampling if creating images using a computer
  - Remove all sharp edges (high frequencies) from the scene before drawing it
  - That is, blur character and line edges before drawing
Next Time

☐ Composition
☐ NPR
☐ 3D Graphics Toolkits