Computer Graphics	Prof. Brian Curless
CSE 557	Winter 2006

Homework #1

Sampling theory, image processing, affine transformations

Assigned: Monday, January 23, 2006

Due: Monday, January 30, 2006 (at the beginning of class)

Directions: Please provide short written answers to the following questions. Feel free to talk over the problems in general terms with classmates, but please *answer the questions on your own*.

Name:_____

Problem 1. Fourier transforms and signal reconstruction

In this problem, you will take a closer look at Fourier transforms and signal reconstruction. The focus is to evaluate the cost of using sinc vs. linear and bilinear interpolation, as well as the mathematics behind them. For the purposes of this problem, you may assume that all signals are bandlimited sufficiently, so that aliasiang is not a concern. Technically, linear and bilinear filtering will introduce a small amount of aliasing, but it is usually not very objectionable.

[As noted in class, the Mitchell filter offers a nice trade-off between the sinc filters and linear/bilinear filters. Still, bilinear filtering is probably the most ubiquitous kind of resampling filter and is built into all graphics hardware for image resampling operations (such as texturing, a topic for later in the quarter).]

Before moving on to the problem, we'll review some relevant facts.

Recall that the "box" function II(x) is defined as:

$$\mathrm{II}(x) = \begin{cases} 1 & |x| < 1/2 \\ 1/2 & |x| = 1/2 \\ 0 & |x| > 1/2 \end{cases}$$

and its Fourier transform is sinc(s) = sin(πs)/πs. Likewise, the Fourier transform of sinc(x) is II(s). In either case, we can show that sinc(0) = 1.

The "hat" (or "tent") function, $\Lambda(x)$, is defined as:

$$\Lambda(x) = \begin{cases} 1-|x| & |x| < 1 \\ 0 & |x| > 1 \end{cases}$$

We sample a function by multiplying with the comb or shah function:

$$\hat{f}(x) = f(x)\mathrm{III}(x) = \sum_{i=-\infty}^{i=\infty} f(i)\delta(x-i)$$

We reconstruct by convolving with a reconstruction filter $r(x)$:

$$\tilde{f}(x) = r(x) * \hat{f}(x) = \sum_{i=-\infty}^{i=\infty} f(i)r(x-i)$$

The last two equations generalize naturally into 2D. One more convenient fact is that separable filters of the form $c(x,y) = a(x)b(y)$ lead to simplified convolution:

$$\begin{aligned} g(x,y) *_{2\mathrm{D}} c(x,y) &= \iint g(x',y')c(x-x',y-y')dx'dy' \\ &= \iint g(x',y')a(x-x')b(y-y')dx'dy' \\ &= \int \left\{ \int g(x',y')a(x-x')dx' \right\} b(y-y')dy' \\ &= g(x,y) *_{1\mathrm{D}} a(x) *_{1\mathrm{D}} b(y) \end{aligned}$$

So, you can convolve your signal first in the x-direction with $a(x)$, and then in the y-direction with $b(y)$. You can also convolve first in y then x; the answer is the same in either case.

Problem 1 (cont'd.)

Now we get to the problems you need to solve:

a) The "ideal" reconstruction filter is r(x) = sinc(x). Is this filter an interpolating filter, i.e., will the reconstructed function $\tilde{f}(x)$ pass through the original sample points $f(i)$? In other words, if we re-evaluated $\tilde{f}(x)$ at integer locations indexed by, say j, do we expect to find that $\tilde{f}(j) = f(j)$, for all integers j? Justify your answer.

b) Let's say we reconstruct with the sinc(x) function, and we want to evaluate $\tilde{f}(1/2)$. Assume that the samples $f(i)$ are positive 8-bit numbers (0-255), that we perform the reconstruction using floating point math (including negative numbers), and that we want an accurate reconstruction such that the smallest contribution from *any* original sample to the reconstruction summation can be at most ½ bit, i.e, +/-0.5 on a scale of 0-255. Given no prior knowledge about $f(x)$, how many samples $f(i)$ must, in the worst case, be included to compute $\tilde{f}(1/2)$?

c) If we use $\Lambda(x)$ as a reconstruction filter, what is the equation for computing $\tilde{f}(x)$ where we decompose the domain variable into $x = i + \Delta$, and i = floor(x), and Δ is a fractional value between 0 and 1?

d) Is $\Lambda(x)$ an interpolating filter? How many samples are needed to compute $\tilde{f}(1/2)$ to the same precision as in (b)?

e) It is easy to show that $\Lambda(x) = \text{II}(x)*\text{II}(x)$. (Try integrating it yourself, or use the convolution applet.) What is the Fourier transform of $\Lambda(x)$? Show your work and sketch the function.

f) One of the most common 2D resampling methods used in graphics is *bilinear interpolation*, defined mathematically as using a reconstruction filter $r(x, y) = \Lambda(x)\Lambda(y)$. Suppose we now have a 2D input function $f(x,y)$ that has been sampled. What is the equation for $\tilde{f}(x, y)$ where we decompose the domain variables into $x = i + \Delta x$, $y = j + \Delta y$, i = floor(x), j = floor(y), and Δx and Δy each range between 0 and 1?

g) Sketch the shape of $\Lambda(x)\Lambda(y)$. Note: there's a very rough sketch in the lecture notes showing two cross-sections of the function; your sketch should be more complete and suggestive of the entire shape.

h) The "ideal" alternative to using bilinear interpolation is to use the function sinc(x)sinc(y). Suppose you were wanted to compute $\tilde{f}(1/2, 1/2)$. Based on your answers to (b) and (f), estimate the number of input samples needed to compute this function value for both bilinear interpolation and sinc(x)sinc(y) reconstruction.

i) What is the Fourier transform of the filter $r(x, y) = \Lambda(x)\Lambda(y)$? You will need to manipulate the equation for the 2D Fourier transform to justify your answer, but you will not need to compute any integrals. You do not need to sketch the function.

Problem 2: Image Processing

Suppose we have two filters:

0	0	0
-1	0	1
0	0	0

Filter *A*

0	1	0
0	2	0
0	1	0

Filter *B*

a) In class, we described a simple and intuitive version of an *x*-gradient filter: [-1 1]. When applied, this filter computes the *finite difference* gradient in the *x*-direction, essentially solving for $\partial f / \partial x \approx \Delta f / \Delta x$, where $\Delta x = 1$ and pixels are one unit distance from their neighbors. Filter *A*, by contrast, is used to compute what is known as the *central difference* *x*-gradient. Although it cannot be normalized in the usual way, since its values sum to zero, it is usually multiplied by a coefficient of ½. Why?

b) Normalize *B*. What will effect will this normalized filter have when applied to an image?

c) Compute *A*B*, using *A* and *B* from the ***original*** problem statement, i.e., ***without*** using the scale factors described in (a) and (b). You can treat *B* as the filter kernel and assume that *A* is zero outside of its support. You do *not* need to show your work. [Aside: convolution is commutative (*A*B=B*A*), so you would get the same answer by using *A* as the filter kernel. But, you would have to remember to "flip" the kernel to get $\widetilde{A}[i, j] = A[-i, -j]$. We've asked you instead to use *B* as the filter kernel, but since *B* is symmetric, i.e., $\widetilde{B}[i, j] = B[-i, -j] = B[i, j]$, you don't need to worry about flipping.]

d) Compute *A*B*, now using *A* and *B* after scaling them according to (a) and (b).

e) If we apply the result of (c) or (d) to an image *f*, we are computing (*A*B*)**f*. Convolution is associative, so we would get the same results as computing *A**(*B*f*). In other words, we can think of this operation in terms of filtering the image with *B*, and then filtering the result with *A*. Answer the following three questions:

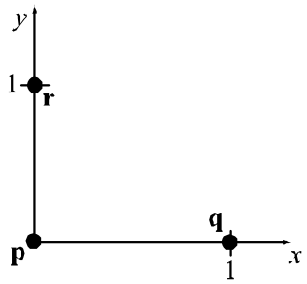
1) Why would it be desirable to apply *B* before computing the central difference gradient (as opposed to not applying *B* at all)?

2) Neglecting efficiency considerations, why might applying *B* be better than applying instead a 3x3 Gaussian filter, call it *B*'?

3) Now, keeping efficiency in mind, would it be faster to compute (*B*f*) then *A**(*B*f*) or instead to compute (*A*B*) then (*A*B*)**f*? Justify your answer.

Problem 3: 2D Affine Transformations

Consider the following points on the 2D plane, and an affine matrix M that will transform them:



$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

a) Suppose we choose M to leave **p** and **q** fixed; i.e., $M\mathbf{p} = \mathbf{p}$ and $M\mathbf{q} = \mathbf{q}$. What constraints does this put on the elements of M? Justify your answer.

b) Provide two (non-identity) affine transformation matrices that satisfy the constraint in (a), and name them; i.e., choose two different transformations from the possibilities we discussed in class. Will the products of these transformations also satisfy the constraint? Show your work.

c) If an affine transformation fixes **p** and **q,** does it also fix any point on the x-axis? Justify your answer.

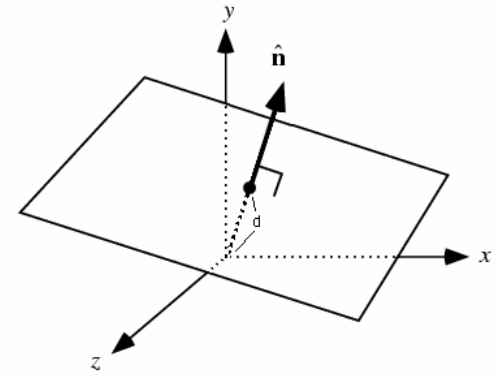
d) If we only wanted to fix **p**, i.e., $M\mathbf{p} = \mathbf{p}$, what constraints would be placed on the elements of M? What general class of transformations satisfies these constraints?

e) Which affine transformation(s) will fix **p, q,** and **r**?

f) In general, if M sends **p**, **q**, and **r**, to **p**', **q**', and **r**', respectively, does knowledge of **p**', **q**', and **r**' determine M uniquely? Justify your answer.

Problem 4: 3D Affine Transformations

The equation $\hat{\mathbf{n}} \bullet \vec{\mathbf{x}} = d$ describes the plane pictured below which has unit length normal $\hat{\mathbf{n}}$ pointing away from the origin and is a distance d from the origin (in the direction of the normal vector).



$$M_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now consider a plane with normal lying in the y-z plane. The normal will have the form $(0, \sin\theta, \cos\theta)$ for some θ. The equation for the plane is then $y \sin\theta + z \cos\theta = d$. Write out the product of 4x4 matrices that would perform a reflection across this plane. One of these matrices will be a reflection matrix; you must use the matrix M_{xy} above, which performs a reflection across the x-y plane. You must write out the elements of the matrices, but you do not need to multiply them out. Justify your answer.