CSC320— Visual Computing, Winter 2005

Assignment 1: Fourier Transforms, Signal Interpolation, and Sampling

Due: 9:10am, Wed., Feb. 9 (at the start of the lecture)
This assignment is worth 10 percent for your grade in this course.

1. **Discrete Fourier Transforms [15pts]:** Calculate by hand the discrete Fourier transform of each of the following signals $f(n)$, which are defined for integers $0 \le n < N$. In these questions you can assume that N, the length of the signal, is even and $N \ge 16$.

   (a) The signal $f(n)$ where

$$f(n) = \begin{cases} 1 & \text{for } n = N/2, \\ -1 & \text{for } n = N/2+1, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

   (b) The signal $f(n)$ where

$$f(n) = \begin{cases} 2 & \text{for } n = N/2, \\ -1 & \text{for } n = N/2-1 \text{ and } n = N/2+1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

   (c) The signal $f(n)$ where

$$f(n) = \begin{cases} 1 & \text{for } N/2-2 \le n \le N/2+2, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

   In each case verify your results are correct (for a specific choice of N) by using Matlab to plot your solution along with the Discrete Fourier Transform calculated using Matlab's `fft` routine. Hand in both your hand calculations and these plots verifying your results. The axes of your plots must be clearly labelled, including the range of frequencies. Discuss any differences between these Matlab results and the results of your hand calculations.

2. **Fourier Series [15pts]:** Calculate by hand the Fourier series coefficients for each of the following signals $f(x)$. In these questions assume that $f(x)$ is defined for real values of x with $x \in [0, N]$ and $N \ge 16$.

   (a) The signal $f(x)$ where

$$f(x) = \begin{cases} 1 + \cos(\frac{\pi}{4}(x - N/2)) & \text{for } x \in [N/2-4, N/2+4], \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

   (b) The signal $f(x)$ where

$$f(n) = \begin{cases} (x - N/2)/4 + 1 & \text{for } x \in [N/2-4, N/2], \\ 1 - (x - N/2)/4 & \text{for } x \in [N/2, N/2+4], \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

   (c) The signal $f(n)$ where

$$f(x) = \begin{cases} 1 & \text{for } x \in [N/2-4, N/2+4], \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

   In each case use Matlab to sum to first K terms of the resulting Fourier series expansion for $f(x)$ and plot the resulting function of x. Use $K = 5, 10$, and 100. Include the original function $f(x)$ in these plots. For which of the three functions considered above does the Fourier series appear to converge the fastest? Which one appears to converge the slowest?

3. **Up Sampling [10pts]:** Let $S(n)$ be a discrete signal defined for integers n with $0 \leq n < N$. Let the up-sampling rate $r > 1$ be an integer. As in class, define the raw up-sampled signal $U(n)$ for $0 \leq n < rN$ to be

$$U(n) = \begin{cases} S(n/r) & \text{for } n \bmod r = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Derive an expression for the discrete Fourier transform of $U(n)$, namely $\hat{U}(k)$, in terms of the the discrete Fourier transform $\hat{S}$ of S. For a general signal S what is the period of the discrete Fourier transform $\hat{U}(k)$? (Hint: The Matlab code `upSample.m` discussed in the lectures performs this up-sampling for $r = 3$.)

4. **Image Translation [10pts]:** Write a Matlab M-file `translateIm.m` which uses separable convolution to translate an image by any real valued vector. In particular, implement the function described by the following header:

```
function imT = translateIm(im, xVec, interpType)
%  imT = translateIm(im, xVec, interpType)
%
% Translate a 2D image im by the 2-vector xVec = (x,y) (possibly real-valued).
% Input:
%   im - a grey-level image represented by  n x m double array,
%   xVec - a 2-vector (type double) specifying the translation,
%   interpType - either 'bilinear' or 'bicubic'
% Return imT having the same size as im, with:
%   imT(i,j) apprx= im(i-xVec(2), j-xVec(1))
% when 1<=i-xVec(2)<=size(im,1) and 1<=j-xVec(1)<=size(im,2).  Otherwise
% the pixel imT(i,j) falls outside of the boundary of im and its value
% is set to nan. Triangular (1D) filter kernels are used when
% interpType is 'bilinear', and the 1D  Catmull-Rom filter kernels are
% used when interpType is 'bicubic'.
```

Your implementation **must** use the M-file `rconv2sep`, available from the `iseToolbox`, to perform the two separable convolutions necessary to interpolate the image at the desired values. Also, arrange your computation so that bilinear interpolation reqires only filter kernels of size (at most) 2×1 and 1×2, and bicubic interpolation requires filter kernels of size (at most) 4×1 and 1×4. Hand in a written description of your algorithm, a printed listing of it, and describe how you tested it. Finally, electronically submit your completed M-file `translateIm.m` for marking by following the directions posted on the course webpage. (If you wish to use a helper method, say to compute the Catmull-Rom filter kernel, then you can include it at the bottom of the file `translateIm.m` after the `return` statement for the original function.)

5. **Sampling and Aliasing [15pts]:** Run the script file checkerAliasHandout.m (since it uses some M-files from iseToolbox and utvisToolbox you will first need access to those, see the course homepage). This script file generates a sequence of images formed from a synthetic moving camera above an infinite checkerboard.

The script file sets up a synthetic camera and forms an image of an infinite ground plane painted with a checkerboard. In terms of world coordinates $\vec{X} = (X_1, X_2, X_3)^T$, the ground plane corresponds to $X_3 = 0$. Given a point (p_1, p_2) in the image (eg. $(p_1, p_2) = (j, i)$ for the pixel in the i^{th} row and the j^{th} column of the image), the corresponding point on the ground plane is at $\vec{X} = (X_1, X_2, 0)^T$ where

$$\alpha \begin{pmatrix} X_1 \\ X_2 \\ 1 \end{pmatrix} = F^{-1} \begin{pmatrix} p_1 \\ p_2 \\ 1 \end{pmatrix}. \tag{8}$$

Here F is a nonsingular 3×3 matrix which depends on the camera postion and orientation. It is computed in `checkerAliasHandout.m`. The details of this computation involve perspective projection of the ground plane, and are beyond the scope of the current course. For our purposes, we are simply given this 3×3 matrix F^{-1} and we can then evaluate equation (8) for any pixel coordinates (p_1, p_2) (not necessarily integer valued).

Notice α is simply equal to the third component of the vector on the left hand side of of equation (8). It turns out that α must be positive for the ray to intersect the ground plane in front of the camera. When $\alpha \leq 0$ a point on the sky is imaged instead (and the corresponding pixel is set to the grey-level `skyGray`). When $\alpha > 0$, the left hand side of of equation (8) is divided by α, and this provides the coordinates (X_1, X_2) of the point on the ground-plane that is imaged to (p_1, p_2). A simple computation then determines if this point is within a white or a black check.

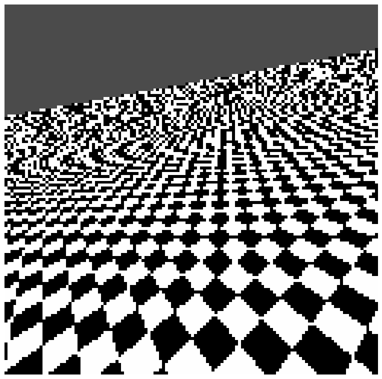
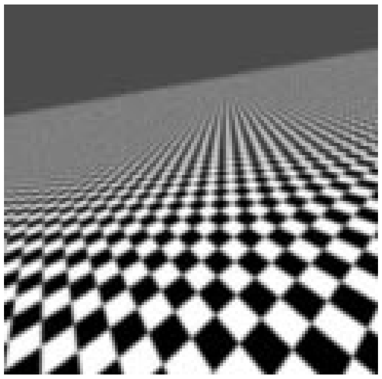

Figure 1: Aliased (left) and properly sampled (right) checkerboard images.

When `checkerAliasHandout.m` is run, notice that the region of the image near the horizon of the ground plane looks bizarre (see Figure 1, left). This effect is due to the true image consisting of very small checks, generating high frequencies in the continuous signal. When this continuous signal is sampled, these high-frequencies are aliased. The problem is that we are sampling one particular point on the ground plane, corresponding to the center of the pixel (p_1, p_2), even when the projected size of the checks become much smaller than a single pixel. The consequence is that it looks almost random as to whether the sampled location is white or black (see the horizon region in Figure 1, left).

Properly designed cameras do not exhibit this problem. Instead of sampling at a particular pixel, the optics of the camera **first** blur the image by a small amount and then this blurred value is sampled and reported as the pixel brightness. This produces a much better image (see Figure 1, right). In this question we will model the optical blur process and (hopefully) learn how this controls the aliasing.

(a) Copy `checkerAliasHandout.m` to `checkerAlias.m`. Modify checkerAlias.m so that instead of sampling just one point on the checkerboard per image pixel, the average value of many different randomly selected points is used. This averaging is used to model a real camera's optical blur.

In particular, for the pixel at location (p_1^0, p_2^0), randomly select K image points, $\{\vec{q}^{\,k}\}_{k=1}^{K}$, with $\vec{q}^{\,k} = (p_1^0, p_2^0, 1)^T + (r_1^k, r_2^k, 0)^T \sigma$. Here r_i^k are independent samples from a zero mean, unit variance, 2D Normal distribution (i.e. see randn() in Matlab). The constant σ determines the standard deviation of the point spread function in terms of the pixel spacing, with $\sigma = 1$ providing a point spread function with a standard deviation equal to the the distance between two neighbouring pixels.

For each of these randomly selected image points $\vec{q}^{\,k}$, compute the corresponding point on the checkerboard (or the sky) using equation (8). Determine the grey-level of this sample in the same manner as is done in checkerAliasHandout.m. That is, determine if the intersection point is on the plane or in the sky. If the point is on the checkerboard plane, then determine whether it is on a white or black check (see checkerAliasHandout.m). Otherwise, the grey-level should be set to a constant, skyGray, as in checkerAliasHandout.m. The average of these grey-levels over all K random samples is then the brightness assigned to the pixel at (p_1^0, p_2^0). This random sampling and averaging process models the optical blur due to camera lens. (There are more computationally efficient ways to model optical blurring, but this method requires the least programming.)

This sampling process needs to be repeated for each pixel in the image, using a different set of random samples $\{(r_1^k, r_2^k)\}$ for each pixel. The number K of samples required to get a good approximation depends on the variability of the grey-level data projecting to each pixel. It turns out that when the white checks have the grey-level of 255 and the black checks have the grey-level 0, the standard deviation of the noise near the horizon after this blurring process is roughly $128/\sqrt{K}$. So using $K = 100$ gives reasonable results.

For small test images (the synthetic blurring is slow), compare the images obtained in this way with the aliased images obtained by the original program checkerAlias.m. Discuss the effect of using sigma equal to 0.2, $2/\pi$, and 2.

(b) Discuss the choice of $\sigma = 2/\pi$ in terms of Fourier transform of the Normal distribution associated with the continuous point spread function, that is, the Fourier transform of

$$G(\vec{r}; \sigma) \ = \ \frac{1}{2\pi\sigma^2} e^{-(r_1^2 + r_2^2)/(2\sigma^2)}. \qquad (9)$$

Explain whether or not this choice of σ is suitable for eliminating most of the aliasing? What about for $\sigma = 0.2$ and $\sigma = 2$?

For your solution to this question include hand written comments, as requested in each of the parts above, along with a printed listing of your completed `checkerAlias.m`. Include printed copies of images generated using the synthetic blur (similar to Figure 1 right, but perhaps smaller) for the three different values of σ mentioned above. Finally, electronically submit your completed M-file `checkerAlias.m` following the directions on the course home-page.