In this assignment you will investigate the application of the lighting, reflectance, image formation and image transform models discussed in class. To get started, read the questions below and download the file A2handout.zip from the assignment posting on the course webpage.

1. Laplacian Pyramids [20pts]:

Image pyramids have many different uses, some of which are nicely explained by Burt and Adelson (1981,1983). For this problem your task is to use the Laplacian pyramid to implement the method for enhancing image depth of field as described in the lecture notes. That is,

- build Laplacian pyramids for the two images. Code for building the Laplacian pyramids is available and explained well in pyramidTutorial.m. This tutorial explains how to build the pyramid, and how to access the individual layers of the pyramid.

- Given the two pyramids, you want to create a new blended pyramid. To do this you will simply choose coefficients from whichever of the original pyramids have larger magnitude. You may only need to do this in the higher frequency bands.

- Finally, reconstruct the image from the blended pyramid using the functions explained in the tutorial.

We have taken some images to help test your code. You will find the two pairs of test images available in A2handout.zip (in directory focusImages/ called pipes1,2 and tripod1,2). You may also want to experiment with your own images that you take with a digital camera.

Implement the steps explained above in Matlab and apply them to the images provided. Write your solution in the form of a Matlab demo, with lots of comments to explain what you are doing, and by displaying intermediate results that are of interest to demonstrate the workings of your implementation.

In addition, note that there are some serious limits to the approach we described above. With one to two paragraphs in your write up or at the end of your tutorial, mention some of the limitations, perhaps with reference to your experimental results with the images provided.

2. Sampling and Aliasing [20pts]:

Run the script file checkerAliasHandout.m (since it uses some M-files from iseToolbox and utvisToolbox, you will first need to get the file matlabVisTools.zip from the course homepage if you haven’t already)). This generates a sequence of images formed from a synthetic moving camera above an infinite checkerboard. The camera model is artificial in that the value at any pixel \((p_1, p_2)\) is given by the value on the checkerboard precisely at the point of intersection between the checkerboard and the ray passing through both the pixel \((p_1, p_2)\) and the camera’s nodal point (pinhole). The first frame of this sequence is displayed in Figure 1 (left), where the random nature of the image in the neighbourhood of the horizon indicates severe aliasing. In contrast, a properly blurred and sampled image is shown in Figure 1 (right).

The details of the mapping between points on the image plane and the corresponding points on the plane of the checkerboard are described in the notes from the second tutorial on the course web page. The end result is quite simple, and we summarize it next. Let \((p_1, p_2)\) denote pixel coordinates in the image,
with $p_1$ the column and $p_2$ the row in the sampled image, starting with $(p_1, p_2) = (1, 1)$ for the pixel in the top-left corner of the image, and ending with $(p_1, p_2) = (n, n)$ for the pixel in the bottom-right corner. Similarly, let $\vec{X} = (X_1, X_2, X_3)^T$ denote rectangular coordinates in the world coordinate frame, with the checkerboard in the plane $X_3 = 0$. Then the ray through the pixel $(p_1, p_2)$ and the nodal point of the camera intersects the ground plane at a position $\vec{x} = (X_1, X_2, 0)^T$ given by the following 2D homography

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

(1)

Here $F$ is a nonsingular $3 \times 3$ matrix. Also $\alpha$ equals the third component on the left hand side of (1), and it 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).

**Part A.** Currently, checkerAliasHandout.m displays a sequence of images which exhibit strong aliasing. The moving sequence helps highlight the nature of the aliasing, which generates a flickering appearance with smooth camera motion. However, the runtime of the algorithm suggested below is somewhat long, so you do not need to run your solution on this whole sequence. In particular, you can change line 57 of checkerAliasHandout.m from:

```matlab
xRange = (-1:0.1:1)*f;
```

to

```matlab
xRange = -1*f;
```

This causes only the first frame of the sequence to be generated. After you have everything working, you may want to replace xRange with the original array above, and run a longer job to compute the properly sampled sequence. However, this last step of computing a properly sampled sequence is optional.

Copy checkerAliasHandout.m to checkerAlias.m (with line 57 changed). Modify checkerAlias.m so that instead of sampling just one point on the checker-board per image pixel, the average value of many different randomly selected points is used. This averaging is used to model a real camera’s point spread function, as in Fig. 1 (right).
In particular, for the pixel at location \( (p_1^0, p_2^0) \), randomly select \( K \) image points, \( \{ q^k \}_{k=1}^K \), with \( q^k = (p_1^0, p_2^0, 1)^T + (r_1^k, r_2^k, 0)^T \sigma \). Here \( r_1^k \) and \( r_2^k \) are independent samples from a zero mean, unit variance, 2D Normal distribution (i.e. see randn() in Matlab). The constant \( \sigma \) determines the standard deviation of the point spread function in terms of the pixel spacing. With \( \sigma = 1 \) we obtain a point spread function with a standard deviation equal to the distance between two neighboring pixels.

For each of these randomly selected image points \( q^k \), compute the corresponding point on the checkerboard (or the sky) using Equation (1). Determine the grey-level of this sample in the same manner as is done in checkerAliasHandout.m. That is, first determine whether the intersection point is on the plane or in the sky. If the intersection is in the sky, then set the grey-level to the constant \( \text{skyGray} \), as in checkerAliasHandout.m. Otherwise, it is on the checkerboard. You then need to determine whether it is on a white or a black check (see 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 diffraction and defocus. (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 (although the running time may be slow).

Compare the image obtained in this way with the aliased image obtained by the original program checkerAliasHandout.m. Discuss the effect of using sigma equal to \( 0.2 \), \( 2/\pi \), and 2.

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

\[
G(\tilde{r}; \sigma) = \frac{1}{2\pi\sigma^2} e^{-(r_1^2 + r_2^2)/(2\sigma^2)} .
\]

(2)

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

**What to hand in:** Write well commented Matlab demos for the programming tasks described above, displaying intermediate results of interest. Also provide a short report addressing each of the questions asked above (hand-written reports are fine). You can assume that the marker knows the context of the questions, so do not spend time repeating material in the hand-out, or in class notes. Hand in the written work directly to the TA, and email the completed Matlab files for each question to mstefan@cs.utoronto.ca.

**References**
