CS 418 Homework 2

(revised February 11, 2003)

out: Tuesday, February 4, 2003
due: Friday, February 14, 2003

Problem 1: Compositing

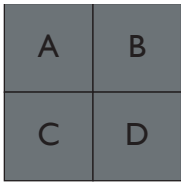
Consider the two squares shown here:

They abut exactly with no gap or overlap and are filled with the pixel value 0.0. Assume we rasterize the squares separately to produce two (r, g, b, α) images, using box filtering for antialiasing and to compute α, then composite the two layers in turn over a white background (pixel value 1.0). In general, the resulting image will not be solid black over the area covered by the two squares.

1. What is the most incorrect pixel value possible and under what conditions does it occur? That is, what is the largest pixel value that could be generated for a pixel that ought to be completely black? You can assume the squares are large compared to the pixel grid.

2. What assumption of the alpha compositing algorithm is being violated to cause this error?

3. Answer part 1 in the case where there are four squares rasterized separately in the same way:



Problem 2: Sampling Theory

In this problem we use the theory of sampling and reconstruction in the frequency domain to analyze the process of taking a Gaussian (which I've chosen as the signal simply because it's a convenient function to work with), sampling it on a grid in 1D, and then linearly interpolating between the samples to reconstruct the signal.

Here are two facts that are relevant to this problem. First, let $b(x)$ be the box function

$$b(x) = \begin{cases} 1 & \text{if } -0.5 < x < 0.5 \\ 0 & \text{otherwise.} \end{cases}$$

Then its Fourier transform $B(u)$ is the sinc function:[1]

$$B(u) = \operatorname{sinc} \pi u = \frac{\sin \pi u}{\pi u}.$$

Second, let $g(x)$ be the Gaussian function with variance 0.5:

$$g(x) = e^{-x^2}/\sqrt{\pi}$$

Then its Fourier transform $G(u)$ is the Gaussian function with variance $1/\pi^2$:

$$G(u) = \sqrt{\pi} e^{-\pi^2 u^2}.$$

1. Compute $b * b$, the convolution of the box with itself. That is, give an expression for $(b * b)(x)$.
2. Assume we point sample $g(x)$ with one sample per unit distance and reconstruct it with a box function.
   (a) What is the frequency content in the original signal at the Nyquist frequency?[2]
   (b) What is the frequency content of the reconstructed signal at the Nyquist frequency?
3. Answer the above questions for a sample frequency of two samples per unit distance.
4. Now assume we reconstruct the samples using linear interpolation – that is, we draw straight lines connecting the samples on the graph. What reconstruction filter does this correspond with?
5. Answer parts 2 and 3 for the linearly interpolated reconstruction.

[1] It doesn't affect this problem, but if the division by zero at $u = 0$ worries you, define $\operatorname{sinc} 0$ to have its natural limiting value from both sides, whith is 1.

[2] Recall that the Nyquist frequency is defined as half the sample frequency.

Problem 3: Image Resampling

Here is some pseudocode for downsampling a grayscale image using a separable filter (for simplicity it uses real numbers in the image). Note that because this is pseudocode all variables can be thought of as real numbers.

```
Image resampleImage(oldImg,        —input pixel array
                    nxOld, nyOld,  —dimensions of input image
                    nxNew, nyNew)  —dimensions of desired image
{
   allocate newImg[nxNew][nyNew];
   sx = nxOld / nxNew; —scale factor in x
   sy = nyOld / nyNew; —scale factor in y
   rx = sx * FILTER_RADIUS; —radius of filter, measured in old pixels
   ry = sy * FILTER_RADIUS;
   for (int jNew = 0; jNew < nyNew; jNew++)
      for (int iNew = 0; iNew < nxNew; iNew++) {
         x = iNew * sx; —sample point in old-image coordinates
         y = jNew * sy;
         il = max(ceil(x - rx), 0); —first and last column in old image
         ih = min(floor(x + rx), nxOld - 1);
         jl = max(ceil(y - ry), 0); —first and last row in old image
         jh = min(floor(y + ry), nyOld - 1);
         sum = 0;
         for (int jOld = jl; jOld <= jh; jOld++)
            for (int iOld = il; iOld <= ih; iOld++) {
               dx = iOld - x; —offset from sample point to old pixel
               dy = jOld - y;
               fval = filter(dx/sx) * filter(dy/sy);
               sum += fval * oldImg[iOld][jOld];
            }
         newImg[iNew][jNew] = sum;
      }
   return newImg;
}
```

This code is inefficient because of the double inner loop over `iOld` and `jOld`, which can be avoided because the filter is separable. Provide pseudocode that computes the same result but takes advantage of separability.

Problem 4: 2D Transformations

Derive the affine transformation T that takes the points $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ to the points $\mathbf{p}'$, $\mathbf{q}'$, and $\mathbf{r}'$, and give the 3x3 homogeneous matrix representation of T. Neither of the sets of points is collinear. Denote the coordinates as $\mathbf{p} = [\,x_p\ y_p\,]^T$, $\mathbf{p}' = [\,x'_p\ y'_p\,]^T$, and so on. Show your work. It is OK for your solution to be in terms of operations like matrix multiplication and inverse; you do not need to write out components unnecessarily.

Hint: It is easier to think of this as a linear algebra problem than as a geometry problem.

Hint: The form $T = D(\mathbf{p}')LD(-\mathbf{p})$, where $D(\mathbf{u})$ is a translation by $\mathbf{u}$ and L is a linear transformation, may be useful.