Introduction to Computer Graphics and Computer Vision

Assignment 3: Due 7/27/2015

Nicholas Dwork

For this assignment, don't submit any printouts of images. If you want to turn in some answers handwritten, please submit all images and links in a pdf document electronically. Please don't submit any handwritten code; all code should be submitted electronically.

1 Main Puzzles

P 1 If you're still enjoying these things, visit `http://lightbot.com/hocflash2014.html` and complete the puzzles. (Note, this tutorial will ask you whether you want to play the game online or download the software. Choose to play the game online.) Please only do this outside of the study session.

P 2 Prove the Pythagorean Theorem.

P 3 (*Credit: Dr. Boyd*) In this problem, you will find a rational function (i.e., a ratio of polynomials)

$$f(t) = \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2}$$

that statisfies the following conditions:

$$f(1) = 2, \quad f(2) = 5, \quad f(3) = 9, \quad f(4) = -2, \quad f(5) = -4.$$

Your job is to find numbers c_0, c_1, c_2, d_1, and d_2 for which these conditions hold.

Part a Let $x = (c_0, c_1, c_2, d_1, d_2)$. Explain how to formulate this problem as finding a vector x that satisfies a system of linear equations $Ax = b$. Be sure to specify what A and b are.

Part b Solve the system of linear equations in Matlab to find the coefficients. What do you get for vector x? Plot the function f for `x=-2:0.2:8`. On top of the curve, plot the given points using the following Matlab command.

```
plot( x, f, 'ro', 'MarkerFaceColor', 'r' );
```

Your points should lie on the curve.

P 4 *Alpha Blending:* If we are given two points (or vectors) p_1, p_2, then we can parameterize the line segment between them as follows:

$$f(\alpha) = \alpha\, p_1 + (1 - \alpha)\, p_2,$$

where $\alpha \in [0, 1]$. What value does the function take on when $\alpha = 0$, $\alpha = 0.5$, $\alpha = 1$.

You now know that images are equivalent to vectors in some way (see problem 10 of Assignment 2). We can use the above parameterization to make a line segment between images (gnarly, I know). In this problem we are going to visualize this line. Download two images from the internet that you like. Use Matlab's `imresize` to change these images so that they are the same size. Make an array of α values between 0 and 1 in steps of 0.01. Compute $f(\alpha)$ for each value of α where p_1 is one image and p_2 is the other. Save each image, and make an animated gif of your images (if you want, you can use the `makeAGif` function located at `www.stanford.edu/~ndwork/si2015/code/makeAGif.m`). Store this animation in a Google Drive or Dropbox account and provide a link in your solution.

Note that when you save the images, you're going to want to save them with a naming convention that stores the images in order. The matlab function `num2str` converts images to character arrays (or strings). Suppose you chose to do this problem with a `for` loop (which would be a good thing to do). Let `i` be the index of the for loop. Then suppose you wanted the image to be named `img_i.jpg`, where instead of `i` you replaced it with the index corresponding to that image. And you wanted these images stored in their own directory. You could do that with the following set of commands.

```
mkdir( 'imgLine' );
for i=1:100
  ...
  outFilename = [ './imgLine/img_', num2str( i, '%4.4i' ), '.jpg' ];
  imwrite( outImg, outFilename );
end
```

This will append four digits to the end of each filename.

P 5 In this problem, you will write code to determine if a line intersects a line segment (in 2D). A line is defined by a point and a vector; a line segment is defined by two points. The prototype of the function is as follows:

```
out = lineSegmentIntersect( point, vector, pt1, pt2 )
```

The output should be 1 (or true) if they do intersect and 0 (or false) if they don't. (Hint: for this puzzle, you may want to use the parameterization of the previous puzzle.)

P 6 For this problem, use any image that you like. Use Matlab's `interp2` function to shift this image horizontally and vertically by 0.5 pixels in each direction. Make a gif animation that alternates between the two images. Upload this animation to your Google Drive or Dropbox account and provide a link in your solution.

P 7 Find an example of art that we haven't discussed. Tell me why you think it's artistic. (Feel free to use your own definition of art; you don't have to use mine.)

P 8 *Relief Distortion Maps:* Download the file at the following website address. You can load this data using Matlab's `load` function (`load hawaii.mat`).

`www.stanford.edu/~ndwork/si2015/imgs/hawaii.mat`

This is called a Digital Elevation Map; each location in the image (corresponding to a latitude and longitude) is given an altitude. The value of the image is the altitude of that location in meters. Display this image (you don't need to turn this in; just do this to make sure you've got the data loaded properly). Create a horiztonal relief distortion map of this image (refer to Image Processing Lecture 1 for details). Create vertical and diagonal relief distortion maps. Create relief distortion maps from some image from the internet that you like. Submit all relief distortion maps in your solution.

P 9 Download the images `img1`, `img2`, and `img3`, from the following address. Convert these images to grayscale using `rgb2gray`. Stitch these images together.

`www.stanford.edu/~ndwork/si2015/imgs/pierce`

P 10 Savor some part of your life today like we did with the chocolate.

P 11 Microsoft Paint is an image manipulation program that comes with all versions of Windows. If you open an image and move the cursor around, an info bar in the software shows you the pixel coordinates of your cursor. Write a function that accepts "Microsoft Paint" pixel coordinates and converts them to "Matlab" pixel coordinates. (Make sure that you write a test function for this routine. How can you test it?)

P 12 This puzzle will use the data of Hawaii located at the following address. Place a camera in your scene. Project the island into your camera (put the camera in a location so that the image looks nice). For the color of each point on the island, use a relief distortion map.

`www.stanford.edu/~ndwork/si2015/imgs/hawaii.mat`

P 13 Read the following document, which provides an introduction to programming and objects. Can you predict what the output of the final program will be? Make this program in Matlab. Did you get the answer right?

`http://stanford.edu/~ndwork/intro2Programming.pdf`

You can work off of some sample code I made to help you; download the files `cat.m`, `treatment.m`, and `run_testCat.m` from the following website address.

`www.stanford.edu/~ndwork/si2015/code`

Note: the code in the document is not written in Matlab; it's actually much closer to C++. In this problem, you're being asked to create this function in Matlab. All of the programming that we will do in this class will be done in Matlab unless otherwise specified.

2 Challenge Puzzles

CP 1 Download the image of tiles at the following address. This will be your "source" image. Convert the image to grayscale using `rgb2gray`.

```
www.stanford.edu/~ndwork/si2015/imgs/tiles.jpg
```

Create an image that's 500×500 in size. This will be your "target" image. Choose four points in your target image that make up a square near the center. Use `ginput` to determine the coordinates of the four corners of the center tile in the source image. Determine a homography that maps the four corners of the center tile to the four corners of the of your square in the target image. Then project your source image into your target image. Submit your final image in your solution.

CP 2 Do the puzzles for image stitching and tile projection again, but this time do them in color.

CP 3 *X-Ray Simulation:* In this puzzle, we are going to simulate an X-Ray system with a parallel projection (all the X-rays traveling through the object will be parallel).

Part a Make a 3D array that's $1000 \times 1000 \times 1000$. Each voxel in the array (a *voxel* is to 3D what a *pixel* is to 2D) represents a small volume in space (let's say $100\mu m \times 100\mu m \times 100\mu m$). By default, each location in space will have an attenuation coefficient of 0. Place two overlapping spheres with different centers and of different radii inside the volume, each with a different attenuation coefficient (choose values that are reasonable). For voxels where the spheres overlap, make the attenuation coefficient of each location in space the sum of the attenuation coefficients of the spheres. Use Matlab's `max` function to find the maximum attenuation coefficient along one dimension (it doesn't matter which) and show the result. Make sure both of your spheres are completely contained within the volume and that they're both visible in the maximum image. (Note: each voxel will have a location in space; you're going to have to choose a point in space to call $(0, 0, 0)$.) Submit this image as part of your answer.

Part b Now that you've got your volume with your subject inside it, we're going to simulate an X-Ray projection. The Beer-Lambert law specifies that the attenuation through a uniform volume is modeled by the following formula,

$$I(x) = I_0 \exp(-\mu\, x),$$

where I_0 is the initial intensity of the X-Ray beam, and μ is the attenuation coefficient. Similarly, for a voxel of size Δx,

$$I(x + \Delta x) = I(x) \exp(-\mu \Delta x).$$

Now you will use these equations to simulate the intensity of X-Ray received by an X-Ray detector. Here's how. Choose an initial X-Ray intensity of 1. Choose an axis (it doesn't matter which one), and project X-Rays through the volume along this axis. Make a 1000×1000 image, where the value in each pixel is the final intensity after the X-Rays travel through your volume.

Part c Write a for loop that rotates the volume about its center (here is where your choice of the $(0, 0, 0)$ location is relevant) in steps of $\pi/25$. Rotate the volume about an axis perpendicular to the X-rays. For each angle, simulate the X-Ray projection through the volume. Make an animated gif of your images (if you want, you can use the `makeAGif` function located at `www.stanford.edu/~ndwork/si2015/code/makeAGif.m`). Store this file in your Google drive or Dropbox account and provide a link in your.

CP 4 Prove that there is no smallest positive number. (Note: this will be a proof by contradiction. That means that you start by assuming the statement is true, and then arrive at a statement that doesn't make sense. If all of your steps are valid, then the assumption must be wrong. So your proof should start with, "Suppose there exists a smallest number; we will denote this number as ϵ." See where this takes you; you should arrive at a contradiction.) For this problem, only use the axioms of the real numbers and the following theorems.

- If $\epsilon > 0$, then $\epsilon^{-1} > 0$.
- If $a > b > 0$ then $b^{-1} > a^{-1} > 0$.