Homework #1

Alpha Compositing, Image Processing,

Affine Transformations, Hierarchical Modeling, Projections

Assigned: Friday, October 15th

Due: Thursday, October 28nd
at the beginning of class

Directions: Please provide short written answers to the following questions on your own paper. Feel free to discuss the problems with classmates, but please ***answer the questions on your own and show your work.***

Please write your name on your assignment!

Problem 1: Short answer (10 points)

a) (4 Points) Suppose you have a standard color LCD display and a pair of LCD shutter glasses. The display first shows a left-eye image, then a right-eye image, and so on, while the LCD shutter glasses synchronously let light reach the left eye, then the right, etc. An LCD shutter is essentially one giant LCD pixel with no color filter, driven with a voltage to be either opaque or transmissive. The display and shutters are designed to give a reasonably bright picture when sitting naturally in front of the display. Assume that the LCD crystal at each display pixel is oriented the same way as every other pixel in the display (regardless of color filter).

- If you tilt your head sideways (i.e., tilting your head over to one of your shoulders, so that the imaginary line segment connecting your eyes is now aligned with the vertical direction), will the displayed images appear dimmer in one eye, both eyes, or neither eye? Justify your answer.

- Suppose you removed the LCD filter panel in front of the unpolarized backlight and looked at the even backlighting with your naked eye(s); you would see even, white light. Roughly how much dimmer would you expect that light to become after putting the panel back on and putting on the shutter glasses (which are turned on and shuttering), assuming the framebuffer is set to white at each pixel for both eyes? [Recall that unpolarized light intensity is cut in half by linear polarization. Assume that the R,G,B sub-pixels each transmit 1/3 of the visible spectrum of the light. Perceived brightness is averaged over time.] Justify your answer.

b) (4 Points) Consider two vectors u and v which are of non-zero length and not parallel to each other. Which of the following is true and which is false:

$$(u\times v)\times u = u\times(v\times u)$$

$$\left[(u\times v)\times u\right]\cdot u = 0$$

$$\left[(u\times v)\times u\right]\cdot v = 0$$

$$\left\{\frac{u}{\|u\|}\times\left[\frac{(u\times v)\times u}{\|(u\times v)\times u\|}\right]\right\}\cdot\left\{\frac{u}{\|u\|}\times\left[\frac{(u\times v)\times u}{\|(u\times v)\times u\|}\right]\right\}=1$$

You do **not** need to justify your answer.

c) (2 Points) In order to draw 3D graphics without noticeable screen refresh artifacts, we always use double-buffering. Do we need to duplicate both the color buffer and the Z-buffer to avoid these artifacts? Explain.

Problem 2: Alpha compositing (19 points)

The alpha channel is used to control blending between colors. The most common use of alpha is in “the compositing equation”

$$\mathbf{C} = \alpha\,\mathbf{F} + (1-\alpha)\,\mathbf{B} \quad \text{or} \quad \begin{bmatrix} C_R \\ C_G \\ C_B \end{bmatrix} = \alpha \begin{bmatrix} F_R \\ F_G \\ F_B \end{bmatrix} + (1-\alpha) \begin{bmatrix} B_R \\ B_G \\ B_B \end{bmatrix}$$

where α is the blending coefficient, **F** is the foreground color, **B** is the background color, and **C** is the composite color. In film production, compositing is a common operation for putting a foreground character into a new scene (background). The challenge faced with real imagery is to extract per pixel alpha and foreground color from a live action sequence, to enable compositing over a new background.

(a) (4 points) When filming an actor, a color **C** is observed at each pixel. If the three observed color channel values C_R, C_G, and C_B are the only knowns at a given pixel, how many unknowns remain in the compositing equation at that pixel? Treating each color channel separately, how many equations are there at the pixel? Is it generally possible to solve for all the unknowns under these circumstances? [Note: we are treating each pixel in isolation, so in each of these problems, you should just be thinking in terms of a single pixel.]

(b) (3 points) To assist the process of extracting the desired **F** and α values, the actor may be filmed against a known background, typically solid blue or green. If the components of **B** are known, how many unknowns remain at a given pixel? Is it possible, in general, to solve for **F** and α under these circumstances?

(c) (6 points) When filming the original Star Wars trilogy, the starships were assumed to contain only shades of gray and were filmed against a solid blue background. Thus, at a given pixel, the visual effects people could assume $\mathbf{F} = [L\ L\ L]^T$, where L is a shade of gray, and $\mathbf{B} = [0\ 0\ 1]^T$, where color channel values are in the range [0...1]. Given an observed color $\mathbf{C} = [C_R\ C_G\ C_B\ ]^T$ at a pixel, compute α and L in terms of the color components of **C**. You should comment on how to handle the case when $\alpha = 0$. Show your work. [Note: if the answer is not unique, just provide one possible solution.]

(d) (6 points) Suppose you had the luxury of two consecutive images of a stationary foreground subject against a blue and a green background in succession, $\mathbf{B} = [0\ 0\ 1]^T$ and $\mathbf{G} = [0\ 1\ 0]^T$, thus recording two colors, **C** and **D**, respectively, at each pixel. You would then have to consider two color compositing equations $\mathbf{C} = \alpha\,\mathbf{F} + (1-\alpha)\,\mathbf{B}$ and $\mathbf{D} = \alpha\,\mathbf{F} + (1-\alpha)\,\mathbf{G}$. Solve for α and the components of the foreground color, F_R, F_G, and F_B at a given pixel in terms of the components of **C** and **D**. Show your work. [Note: if the answer is not unique, just provide one possible solution.]

Problem 3: Image Processing (16 points)

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) (3 points) 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) (3 points) Normalize *B*. What effect will this normalized filter have when applied to an image?

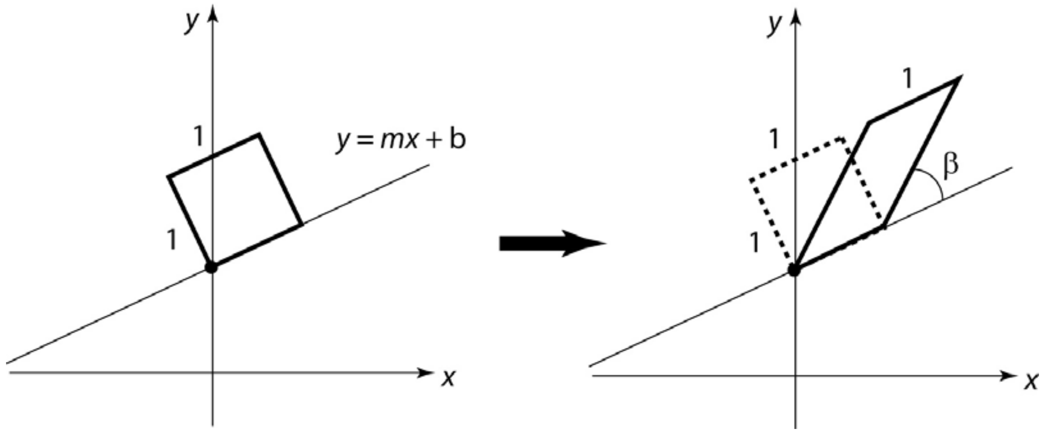
c) (4 points) Compute *A***B*, using *A* and *B* from the ***original*** problem statement, i.e., ***without*** using the scale factors described in (a) and (b). You can treat *B* as the filter kernel and assume that *A* is zero outside of its support. You do *not* need to show your work. [Aside: convolution 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 $\tilde{A}[i, j] = A[-i,-j]$. We've asked you instead to use *B* as the filter kernel, but since *B* is symmetric, i.e., $\tilde{B}[i, j] = B[-i,-j] = B[i, j]$, you don't need to worry about flipping.]

d) (2 points) Compute *A***B*, now using *A* and *B* after scaling them according to (a) and (b).

e) (4 points) 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're filtering the image with *B*, and then filtering the result with *A*. Why would it be desirable to apply *B* before computing the gradient (as opposed to not applying *B* at all)? Why might applying *B* be better than applying a filter *B*' that is filled with a full 3x3 set of coefficients, rather than just a single column of coefficients? [Answer both of these questions.]

Problem 4: Affine Transformations (18 points)

In this problem, you will determine the affine transformation needed to perform a shear with respect to the line $y = mx + b$, while holding fixed the point where the line intersects the y-axis, as illustrated here:



On the left is a unit square sitting on the line before shearing. After transformation, the unit square becomes the solid parallelogram shown on the right; the dotted outline is the original square, for reference. The 1's next to line segments (solid or dotted) indicate their lengths.

Assume you have the following transformations available to you:

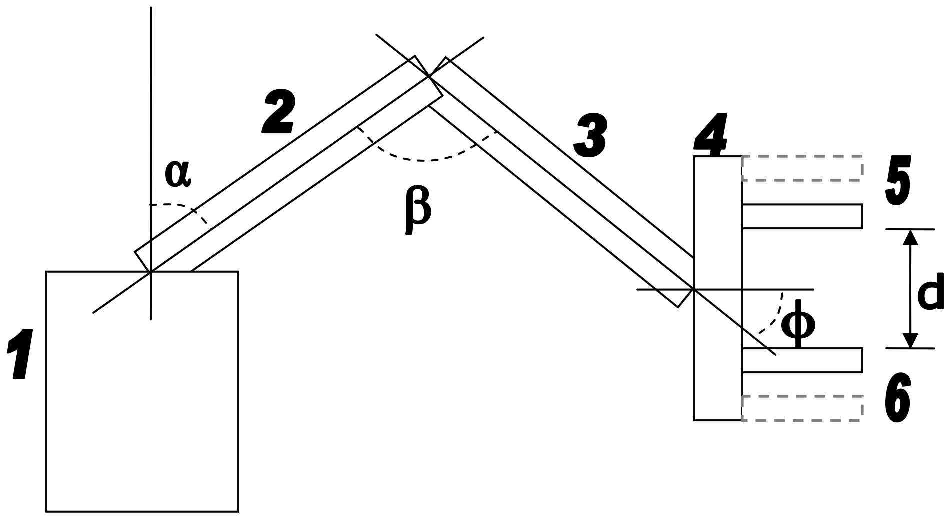
$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad T(c,d) = \begin{pmatrix} 1 & 0 & c \\ 0 & 1 & d \\ 0 & 0 & 1 \end{pmatrix} \quad Sh_x(b) = \begin{pmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Sh_y(a) = \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

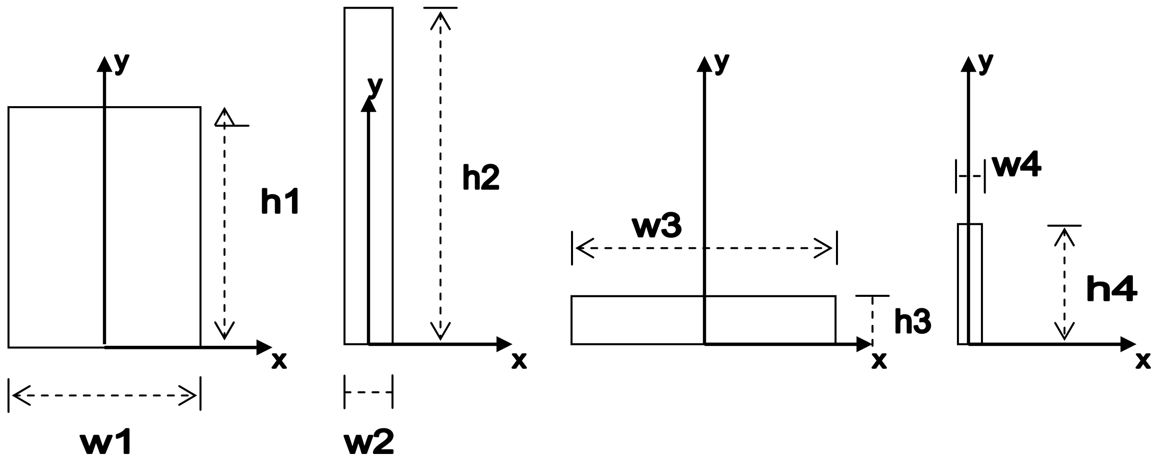
i.e., rotation, translation, and shears in the x-direction and y-direction.

Determine a matrix product that will perform the shear illustrated in the figure using some combination of these transformations. You do ***not*** need to write out the 3 x 3 matrices, just their symbolic references and the arguments they take. Your solution should be parameterized by the slope *m*, y-intercept *b*, and angle *β*. You may assume that *m* is finite (the line is not vertical). Justify your answer by drawing the result of each transformation step, before concatenating the matrices into a single matrix product at the end.

Problem 5: Hierarchical modeling (18 points)

Suppose you want to model the robot arm with calipers, shown below. The arm is made out of six parts (**1**-**6**), and each part is drawn as one of the four primitives (**A**-**D**).





A
(Base)

B
(Arm Segment)

C
(Bar)

D
(Caliper)

The following transformations are also available to you:

- R(θ) – rotate by θ degrees (counter-clockwise)
- T(a, b) – translate by $[a \;\; b]^T$

Note that the angle parameters in the illustration above are each positive in the current configuration in the illustration ($\alpha \approx +60^o$, $\beta \approx +110^o$, $\phi \approx +40^o$), though of course the model can be re-posed by changing these parameters.

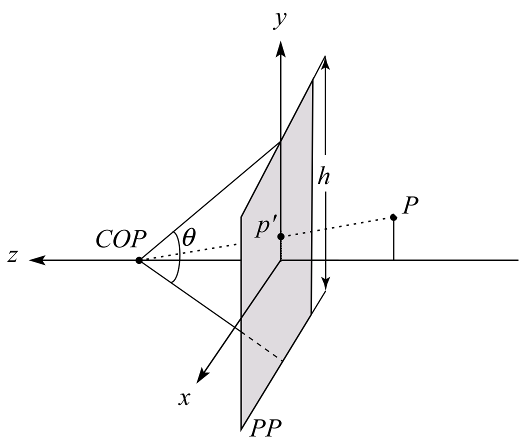
Problem 5: Hierarchical Modeling (cont'd)

a) (15 points) Construct a tree to specify the robot arm that is rooted at **1**. Along each of the edges of the tree, write expressions for the transformations that are applied along that edge, using the notation given above (you do not need to write out the matrices). Remember that order is important! Your tree should contain a bunch of boxes (or circles) each containing one part number (1…6); these boxes should be connected by line segments, each labeled with a corresponding transformation that connects child to parent. The calipers are distance *d* away from each other, and *equidistant from the center of the bar they are attached to*. (Note: Each caliper (objects **5** & **6**) is attached to the bar (object **4**) so you will need to reflect this fact in your tree.)

b) (3 points) Write out the full transformation expression for the part labeled **5**.

Problem 6: Projections (19 Points)

In class, we derived the matrix for perspective projection for a viewer sitting at the origin looking down the –*z* axis. Suppose now that we assume that the projection plane *PP* passes through the origin, and the viewer is at the *COP* somewhere on the +z axis, still looking in the –*z* direction, as illustrated in the figure below.



In addition, assume that we parameterize the projection in terms of viewing angle θ that measures the angle from top to bottom subtended by the image, which is of height *h*, and lies in the projection plane. The size of the image (and thus *h*) is constant throughout this problem.

a) (8 Points) What is the projection matrix that we would use to map a point $P = [x\ y\ z\ 1]^T$ to $p' = [x'\ y'\ 1]^T$. (The *w* coordinate for *p'* will be 1 after doing the perspective divide.) Show your work.

b) (4 Points) Solve for and write out the projection matrix when $\theta = 0$. Also solve for *p'* when $\theta = 0$. What kind of projection does this case correspond to?

c) (2 Points) What happens to *p'* as θ goes to 180^o? Why?

d) (5 Points) Suppose the point *P* starts at position $[x_0\ y_0\ z_0\ 1]^T$ and is initially imaged with viewing angle θ_0, so that its projection is $[x_0'\ y_0'\ 1]^T$. Now suppose we want the projection of *P* to remain constant while now varying the viewing angle. We can do this by moving the camera closer to or further from *P* as we vary θ. Moving the camera closer or further amounts to changing *P*'s *z*-coordinate (and no other coordinates of *P*). How must *P*'s *z*-coordinate vary as a function of θ to achieve the desired effect?