Homework #1

Displays, Image Processing,

Affine Transformations, Hierarchical modeling, Projections

Assigned: Tuesday, January 22nd

Due: Tuesday, January 29th
at the beginning of class

Directions: Please provide short written answers to the following questions, using this page as a cover sheet. Feel free to discuss the problems with classmates, but please ***answer the questions on your own*.**

Name:_____

Problem 1: Short answers (14 points)

Provide short answers to each of the following questions:

a) (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.

b) (2 points) How do you normalize a convolution filter, and what is the purpose of doing so?

c) (2 points) Suppose, for each pixel of a grayscale image, you only average pixels in a 5x5 neighborhood that have values within +/-5% of the pixel being filtered. This is a kind of bilateral filter. Could this be written as a convolution filter? Explain.

d) (4 points) How would you compute the unit-length normal to a triangle in 3D with vertices A, B, and C, specified according to the right-hand rule (where curling the fingers of your right hand from A to B to C will leave your thumb pointing along the normal direction)? What happens if A, B, and C are co-linear?

e) (4 Points) Consider a pair of three dimensional vectors, *u* and *v*, which are of non-zero length and not parallel to each other. Label each of the following is **True**, **False**, or **Nonsense** (i.e., involving an operation that cannot be performed).

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

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

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

$$\frac{u}{\|u\|} \bullet v = u \bullet \frac{v}{\|v\|}$$

You do **not** need to justify your answer for part (e).

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?

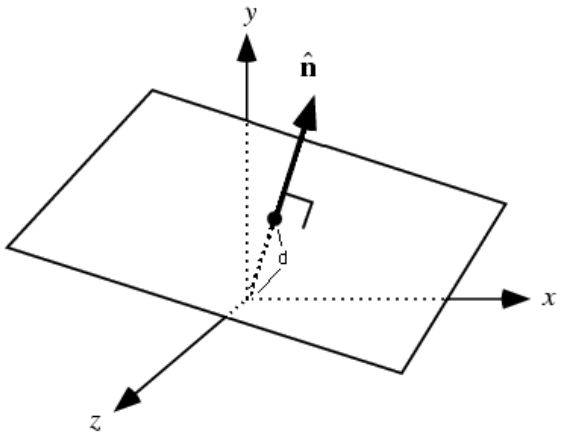
c) (4 points) Compute *A*B*, using *A* and *B* from the ***original*** problem statement, i.e., ***without*** using the scale factors described in (a) and (b). You can treat *B* as the filter kernel and assume that *A* is zero outside of its support. You do *not* need to show your work. [Aside: convolution 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 $\widetilde{A}[i, j] = A[-i,-j]$. We've asked you instead to use *B* as the filter kernel, but since *B* is symmetric, i.e., $\widetilde{B}[i, j] = B[-i,-j] = B[i, j]$, 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: 3D Affine Transformations (19 points)

The equation $\hat{\mathbf{n}} \bullet \bar{\mathbf{x}} = d$ describes the plane pictured below which has unit length normal $\hat{\mathbf{n}}$ pointing away from the origin and is a distance *d* from the origin (in the direction of the normal vector). Any point $\bar{\mathbf{x}} = \begin{bmatrix} x & y & z \end{bmatrix}$ on the plane must satisfy the plane equation $\hat{\mathbf{n}} \bullet \bar{\mathbf{x}} = d$.

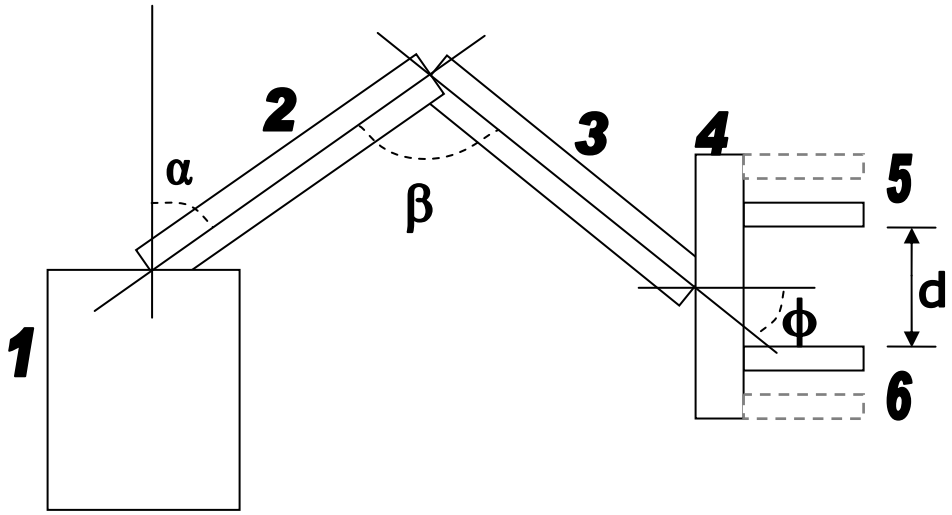


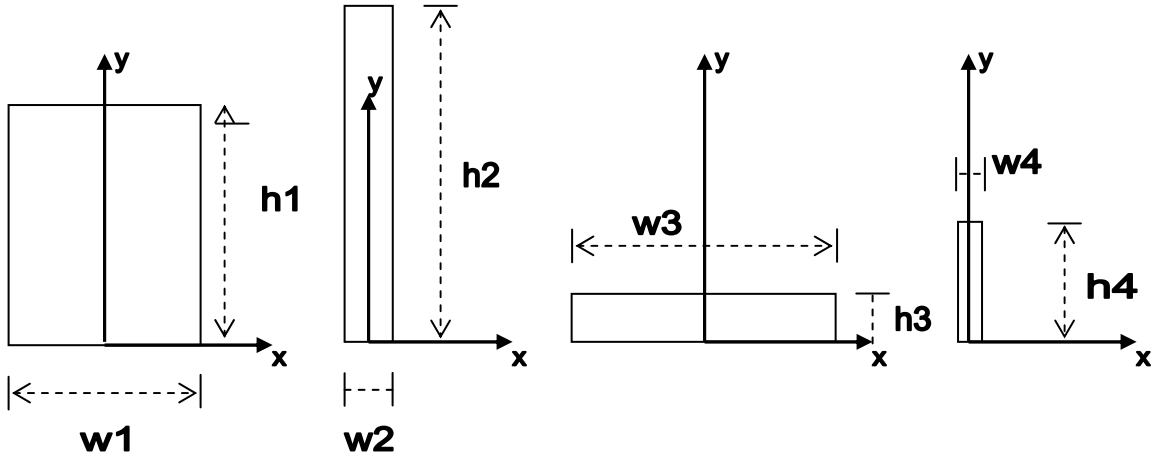
$$M_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now consider a plane with normal lying in the *y*-*z* plane (not shown). The normal will have the form (0, sinθ, cosθ) for some θ. The equation for the plane is then $y \sin\theta + z \cos\theta = d$. Write out the product of 4x4 matrices that would perform a reflection across this plane. One of these matrices will be a reflection matrix; you must use the matrix M_{xy} above, which performs a reflection across the *x*-*y* plane. You must write out the elements of the matrices and the product order in which they would be applied, but you do not need to multiply them out. Justify your answer with words and/or drawings.

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.

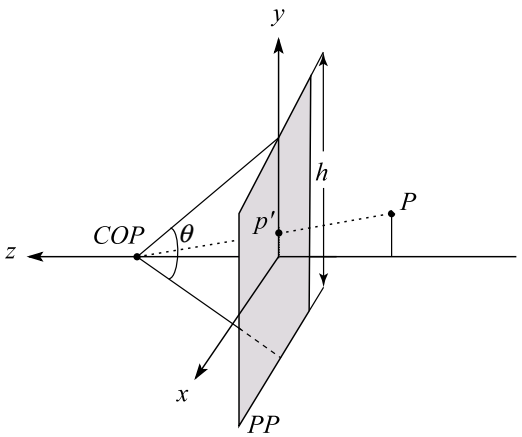
Problem 5: Hierarchical Modeling (cont'd)

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 (14 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.) Your answer should be in terms of θ and *h* only. 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?