Homework #1

Image Processing, Affine Transformations,

Hierarchical Modeling, Projections,

Hidden-surface elimination, Shading

Assigned: Monday, May 04, 2009

Due: 2:30pm, Friday, May 15, 2009
(hand in to Jen Maione CSE 456)

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 and show your work.***

Name:_____

Problem 1: Alpha compositing (22 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) (5 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) (7 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.]

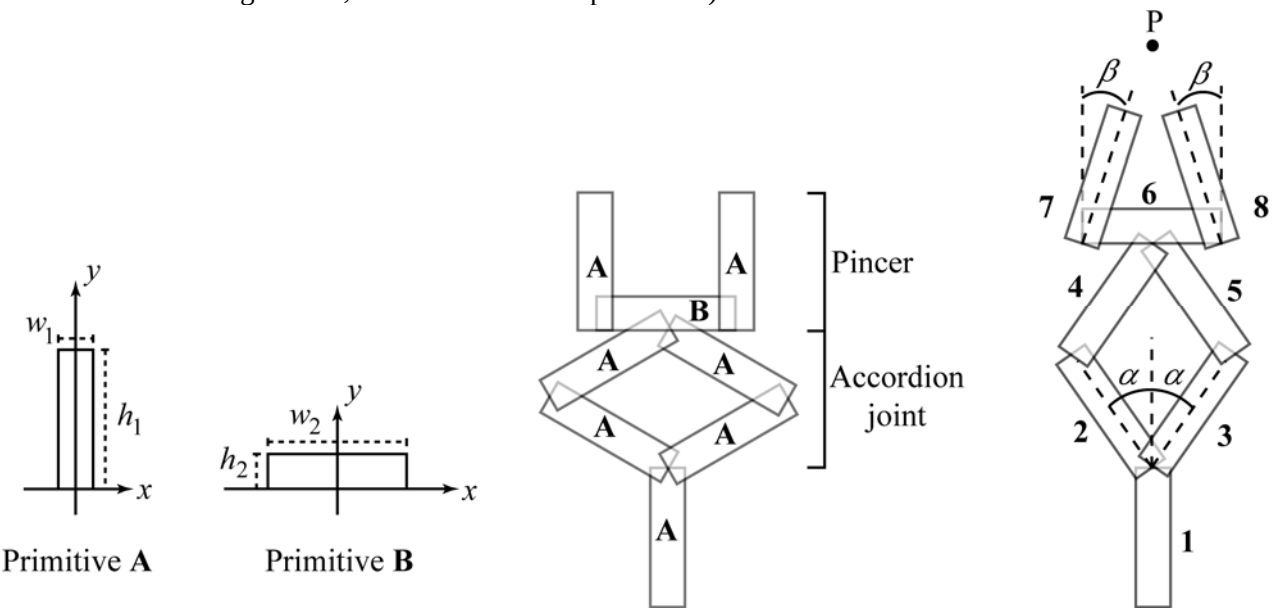
(d) (7 points) Suppose you had the luxury of two consecutive images of a stationary foreground subject against a blue and a green background in succession, $\mathbf{B} = [0\ 0\ 1]^T$ and $\mathbf{G} = [0\ 1\ 0]^T$, thus recording two colors, **C** and **D**, respectively, at each pixel. You would then 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 2: Rotations as Shear Transformations (20 points)

In 2D, a rotation transformation by angle θ can be specified as a series of shear transformation matrices. Give these matrices, or if it cannot be done, prove it.

Problem 3: Hierarchical modeling (26 points)

Suppose you want to model the pincer with accordion joint illustrated below. The model is comprised of 8 parts, using primitives **A** and **B**. The model is shown in two poses below, with the controlling parameters of the model illustrated on the far right. The illustration on the right also shows a point P that the model is reaching toward, as described in sub-problem **c**).



Assume that α and β can take values in the range [0, 90^o]. Also assume that all parts use primitive **A**, except for part **6**, which uses primitive **B**. The model on the left shows the primitives used, the model on the right shows the enumeration (naming) of the parts.

The following transformations are available to you:

- R(θ) – rotate by θ degrees (counter clockwise)
- T(a, b) – translate by $\begin{bmatrix} a \\ b \end{bmatrix}$

a) (16 points) Construct a tree to describe this hierarchical model using part **1** as the root. 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 the order of transformations is important! Show your work wherever the transformations are not "obvious." Your tree should conatin a bunch of boxes (or circles) each containing one part number (1…8); these boxes should be connected by line segments, each labeled with a corresponding transformation that connects child to parent.

b) (3 points) Write out the full transformation expression for part **7**.

c) (7 points) Suppose the primitives are infinitesimally thin, $w_1 = h_2 = 0$, and have lengths $h_1 = 10$ and $w_2 = 12$. Assume that part **1** sits right on the origin in world coordinates. What would the α and β parameters have to be so that the model extends out and closes the pincer just enough to precisely grasp the point $P = \begin{bmatrix} 0 & 28 \end{bmatrix}^T$, in world coordinates. Show your work.

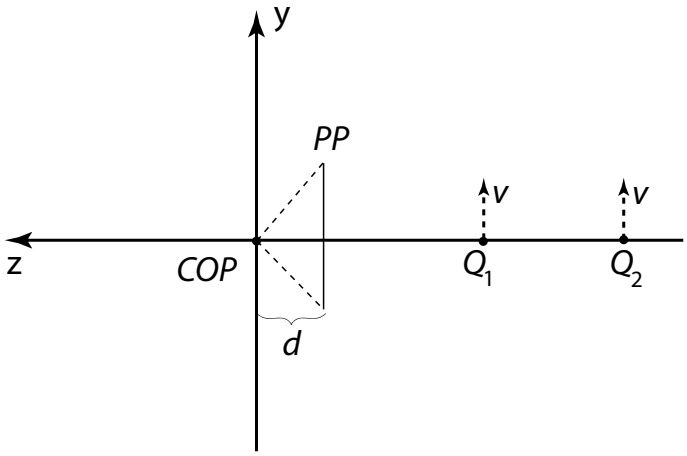
Problem 4: Projections (16 points)

The apparent motion of objects in a scene can be a strong cue for determining how far away they are. In this problem, we will consider the projected motion of points and line segments and their apparent velocities as a function of initial depths.

a) (6 points) Consider the projections of two points, Q_1 and Q_2, on the projection plane *PP*, shown below. Q_1 and Q_2 are described in the equations below. They are moving parallel to the projection plane, in the positive *y*-direction with speed *v*.

$$Q_1(t) = \begin{bmatrix} 0 \\ vt \\ z_1 \\ 1 \end{bmatrix} \qquad Q_2(t) = \begin{bmatrix} 0 \\ vt \\ z_2 \\ 1 \end{bmatrix}$$

$$0 > z_1 > z_2$$



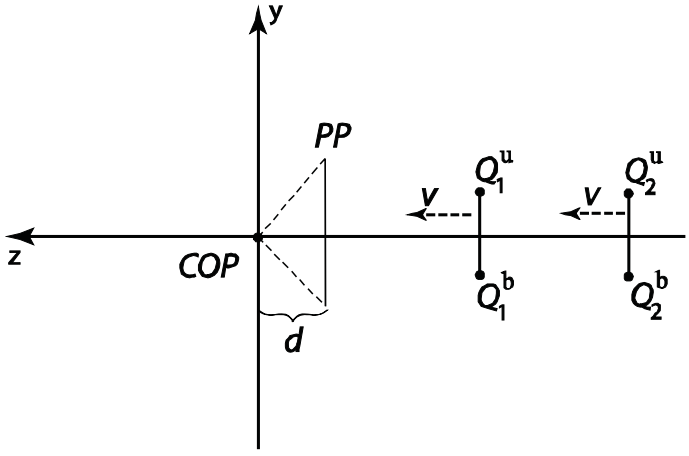
Compute the projections q_1 and q_2 of points Q_1 and Q_2, respectively. Then, compute the velocities, dq_1 / dt and dq_2 / dt, of each projected point in the image plane. Which appears to move faster? Show your work.

b) (10 points) Consider the projections of two vertical line segments, S_1 and S_2, on the projection plane *PP*, shown below. S_1 has endpoints, Q_1^u and Q_1^b. S_2 has endpoints, Q_2^u and Q_2^b. The line segments are moving perpendicular to the projection plane in the positive *z*-direction with speed *v*.

$$Q_1^u(t) = \begin{bmatrix} 0 \\ 1 \\ z_1 + vt \\ 1 \end{bmatrix} \qquad Q_2^u(t) = \begin{bmatrix} 0 \\ 1 \\ z_2 + vt \\ 1 \end{bmatrix}$$

$$Q_1^b(t) = \begin{bmatrix} 0 \\ -1 \\ z_1 + vt \\ 1 \end{bmatrix} \qquad Q_2^b(t) = \begin{bmatrix} 0 \\ -1 \\ z_2 + vt \\ 1 \end{bmatrix}$$

$$0 > z_1 > z_2$$



Compute the projected lengths, l_1 and l_2, of the line segments. Then, compute the rates of change, dl_1 / dt and dl_2 / dt, of these projected lengths. Are they growing or shrinking? Which projected line segment is changing length faster? Show your work.

Problem 5 (12 points)

The Phong shading model can be summarized by the following equation:

$$I_{phong} = k_e + k_a I_a + \sum_i \left[I_{l_i} \left[k_d \left(\mathbf{N} \cdot \mathbf{L}_i \right)_+ + k_s \left(\mathbf{V} \cdot \mathbf{R}_i \right)_+^{n_s} \right] \min\left\{ 1, \frac{1}{a_0 + a_1 d_i + a_2 d_i^2} \right\} \right]$$

where the summation *i* is taken over all light sources. The variables used in the Phong shading equation are summarized below:

I a_0 a_1 a_2 d_i k_e k_a k_d k_s n_s I_a I_{li} $\mathbf{L}_i$ $\mathbf{R}_i$ $\mathbf{N}$ $\mathbf{V}$

a. (3 Points) Which of the quantities above are affected if...

- the viewing direction changes?

- the position of the i^{th} light changes?

- the orientation of the surface changes? Assume that the change in orientation is about the intersection point.

Blinn and Newell have suggested that, when **V** and **L** are assumed to be constants, the computation of $\mathbf{V} \cdot \mathbf{R}$ can be simplified by associating with each light source a fictitious light source that will generate specular reflections. This second light source is located in a direction H halfway between **L** and **V**. The specular component is then computed from $\left(\mathbf{N} \cdot \mathbf{H} \right)_+^{n_s}$ instead of from $\left(\mathbf{V} \cdot \mathbf{R} \right)_+^{n_s}$.

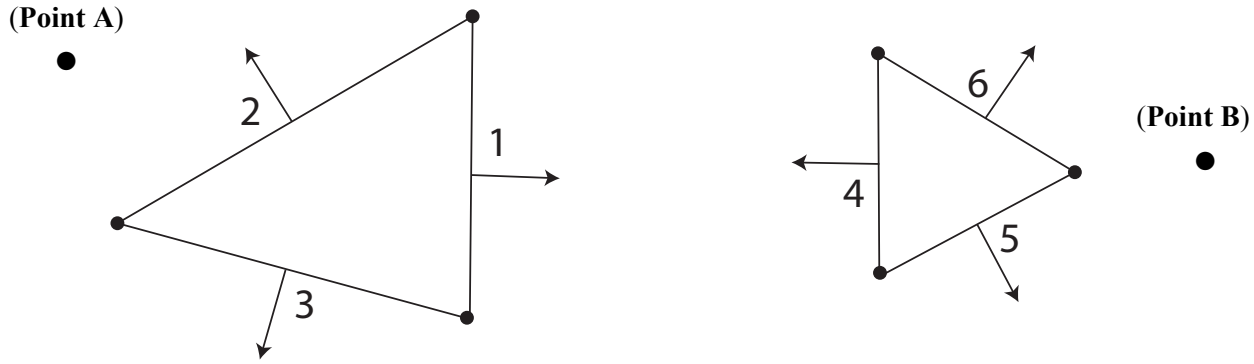
b. (1 Points) Under what circumstances might L and V be assumed to be constant?

c. (2 Points) How does the new equation using H simplify shading equations?

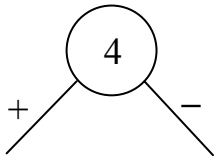
Problem 6. BSP trees (19 points)

Recall that Binary Space Partitioning (BSP) trees break the world up into a tree of positive and negative half-spaces. As spatial partitioning data structures they are very versatile and can be applied to aid in hidden surface removal, collision detection, and even robot motion planning.

Below is a world in two dimensions described by a set of numbered line segments. Note that each segment normal (shown as arrows) points into the positive (+) half-space of its segment.



(a) (4 points) Draw a diagram of a BSP tree that could be generated from this scene using segment #4 as the root (as illustrated below). Note: there are multiple valid answers.



(b) (3 points) Given the viewpoint marked **Point A** in the scene, traverse your BSP tree to list the polygons in the order they would be rendered for hidden surface removal so that no Z-buffer is required (i.e., using the "Painter's algorithm"). For you answer, you only need to provide the list of primitives in the order in which they will be drawn.

(c) (5 points) An alternative approach to hidden surface removal is Z-buffering. Suppose we are performing time-consuming shading at each pixel while Z-buffering. As noted in class, performing shading calculations *after* doing the Z comparison will save some work, but you can still end up overwriting pixel values many times. How can Z-buffering and BSP trees be combined to minimize the number of shading calculations needed to render a scene? Assuming that all objects (primitives) in an arbitrary scene are opaque and non-overlapping, would we ever shade any pixel more than once? Justify your answer.

(d) (3 points) Using the Z-buffer + BSP tree combination from (c) and given your BSP tree from (a), in what order would you draw the line segments for the viewpoint marked **Point B** above?

(e) (4 points) Suppose we render with "backface culling," so that we don't draw any polygons that are facing away from the viewer. How would you modify the BSP-based traversal and drawing algorithm? Explain and then write out the ordered list of primitives that would be drawn for part (b) of this problem and then for part (d), if we used backface culling.