Computer Graphics
CSE 457 - Winter 2015
Instructor: Ira Kemelmacher-Shlizerman

Homework 1

Topics: Displays, Image processing, Affine Transformations, Surfaces of Revolution, Hierarchical Modeling.

Assigned:
Thursday Jan 22, 2015
Due date:
Tuesday Feb 3, 2015, in the beginning of the class.

Feel free to discuss but answer on your own.
Use your own paper to write the solutions.
Show derivations.
Write your name.

Good luck!

Problem 1: Short answers (22 points)

Provide short answers to each of the following questions:

a) (4 points) What is double-buffering and what is its purpose?

b) (4 points) How do you normalize a convolution filter, and what is the purpose of doing so?

c) (4 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. Could this be written as a convolution filter? Explain.

d) (5 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) (5 Points) Consider a pair of three dimensional vectors, *u* and *v*, which are of non-zero length and not parallel to each other. Which 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: Color LCD displays (16 points)

LCD displays operate on the principle of polarizing light at the entrance to each crystal, twisting the polarization by a voltage controlled amount, then passing the resulting light through a final polarizer. The end result is that the light coming out of the LCD has a particular intensity and (fixed) polarization. The color of the light reaching the viewer is controlled with a color filter (red, green, or blue) over each crystal.

Light coming out of the LCD has "linear polarization," characterized by an angle *q*, with a corresponding polarization vector:

$$\mathbf{p} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

A polarizing filter also has an orientation *b* with polarization vector:

$$\mathbf{f} = \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix}$$

The filter only passes the component of the input polarization aligned with filter's polarization. Specifically, it passes a fraction *a* of the light:

$$\alpha = \left|\mathbf{f} \bullet \mathbf{p}\right|^2 = \left|\mathbf{f}^T \mathbf{p}\right|^2 = \left|\cos\beta\cos\theta + \sin\beta\sin\theta\right|^2$$

Now, suppose, through a quirk of manufacturing, all the blue cells for an LCD display have polarization rotated 90 degrees with respect to all the green and red cells. Let's say that the blue polarization is horizontal ($q = 0^o$), and the green and red polarizations are vertical ($q = 90^o$). *Justify your answers* to each of the following questions (i.e., show your work). When giving a color as an answer, it is sufficient to just give the (R,G,B) coordinates.

a) (4 points) If we display solid white – (R,G,B) = (1.0, 1.0, 1.0) – and hold against the screen a polarizer in the vertical ($b = 90^o$) orientation, what color would we see? (We assume in this problem that each color channel ranges from 0.0 to 1.0, so (1.0, 1.0, 1.0) is bright white.)

b) (4 points) What color would we see if we then rotated the polarizer to a diagonal ($b = 45^o$) orientation?

c) (4 points) What color would we see if we then rotated the polarizer to a horizontal ($b = 0^o$) orientation?

d) (4 points) If we keep the filter oriented as in **c**), what color would we see if we now displayed solid green, i.e., (R,G,B) = (0, 1, 0)?

Problem 3: Alpha compositing (23 points)

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

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

where a 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) (4 points) To assist the process of extracting the desired **F** and a 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 a 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 a and L in terms of the color components of **C**. You should comment on how to handle the case when $a = 0$. 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 $\mathbf{C} = a\,\mathbf{F} + (1-a)\,\mathbf{B}$ and $\mathbf{D} = a\,\mathbf{F} + (1-a)\,\mathbf{G}$. Solve for a 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 4: Image Processing (22 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) (4 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 D*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) (4 points) Normalize *B*. What effect will this normalized filter have when applied to an image?

c) (5 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) (3 points) Compute *A*B*, now using *A* and *B* after scaling them according to (a) and (b).

e) (6 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 positive coefficients (e.g., changing the 0's to 1's in the *B* filter given above), rather than just a single column of coefficients?

Problem 5: 3D Coordinate Systems (17 points)

Recall that the canonical coordinate system can be defined in terms of the set of canonical *x*-, *y*-, and *z*-axes and corresponding canonical vectors, $\hat{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, $\hat{\mathbf{y}} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, and $\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. (Here we ignore the homogenous component and assume we are working with linear, not affine, coordinates and transformations.)

Suppose we perform a transformation to a new coordinate system, which maps the canonical axis vectors to new axis vectors ($\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$). I.e., the transformation maps $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ to $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$, respectively. We define $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$ to have similar properties to the canonical vectors: they are each of unit length and orthogonal (perpendicular) to each other, and they follow the right-hand rule, i.e., if you curl the fingers of your right hand from $\hat{\mathbf{u}}$ to $\hat{\mathbf{v}}$ your thumb will point in the direction of $\hat{\mathbf{w}}$.

All of the canonical vectors are of unit length and orthogonal (perpendicular) to each other.

a) (4 points) Suppose we form a 3x3 matrix, each column filled with a canonical vector:

$$M = \begin{bmatrix} \hat{\mathbf{u}} & \hat{\mathbf{v}} & \hat{\mathbf{w}} \end{bmatrix}$$

What kind of geometric transformation does this describe? Explain.

b) (6 points) Solve for $M^T M$, simplifying as much as possible. You can solve in terms of $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$ without expanding them into their components. Or you can explicitly form matrices after expanding the vectors into $\hat{\mathbf{u}} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$, etc., to fill the matrices. Show your work. What does your answer imply about M^T?

c) (7 points) Now suppose you are given two vectors, **a** and **b**, each of arbitrary, non-zero length. We will also assume that these vectors are linearly independent, $\mathbf{a} \neq a\,\mathbf{b}$ for any scalar value of *a*; i.e., the vectors don't point in exactly the same or exactly opposite directions. We can use these vectors to form a right-handed coordinate system, $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, $\hat{\mathbf{w}}$, where:

- $\hat{\mathbf{w}}$ points in the same direction as **b**
- $\hat{\mathbf{u}}$ is orthogonal to both **a** and **b**.

Solve for $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$, each in terms of **a** and **b**. You do not need to expand any vectors into their components; e.g., if you need to take a dot product between **a** and **b**, you can just refer to $\mathbf{a} \bullet \mathbf{b}$ without expanding it into the sum of the products of components. Note that there is more than one possible answer to this problem; you should provide just one solution.