CSE152a – Computer Vision – Assignment 1 WI14
Instructor: Prof. David Kriegman.
Revision 0

Instructions:

- This assignment should be solved, *and written up* in groups of 2. Work alone *only* if you can not find a partner. No groups of 3 are allowed.
- Submit your assignment electronically by email to obeijbom@cs.ucsd.edu with the subject line *CSE152 Assignment 1.* The email should have two files attached.

  1. A pdf file with your writeup. This should have all code attached in the appendix. Name this file: CSE_152_hw1_writeup_lastname1_lastname2.pdf.
  2. A compressed archive with all your matlab code files. Name this file: CSE_152_hw1_code_lastname1_lastname2.zip.

  The code is thus attached *both* as text in the writeup appendix and as m-files in the compressed archive.
- Please make this a proper report, with methods, thoughts, comments and discussions. All code should be tucked away in an appendix.
- No physical hand-in for this assignment.
- You may do problems on pen an paper, just scan and include in the writeup pdf file.
- In general, MATLAB code does not have to be efficient. Focus on clarity, correctness and function here, and we can worry about speed in another course.

Geometry [15 points]

Consider a line in the 2D plane, whose equation is given by $ax + by + c = 0$. This can equivalently be written as $l^T\bar{x} = 0$, where $l = (a, b, c)^T$ and $\bar{x} = (x, y, 1)^T$. Noticing that $\bar{x}$ is a homogeneous representation of $(x, y)^T$, we can view l as a homogeneous representation of the line $ax + by + c = 0$. We see that the line is also defined up to a scale since $(a, b, c)^T$ and $k * (a, b, c)^T$ with $k \neq 0$ represents the same line.

All points (x, y) that lie on the line $ax + by + c = 0$ satisfy the equation $l^T\bar{x} = 0$. We note this down as a fact:

Fact 1: A point x in homogeneous coordinates lies on the homogeneous line l if and only if the dot products $\bar{x}^T l = l^T\bar{x} = 0$

1. [2 points] Using Euclidean coordinates, what is the equation of the line passing through the points $(2, 4)$ and $(4, 5)$.
2. [4 points] Prove the following two statements that follow from Fact1:
   (a) The cross product between two points gives us the line connecting the two points
   (b) The cross product between two lines gives us their point of intersection
3. [3 points] What is the line, in homogenous coordinates, connecting the points $(2, 4)$ and $(4, 5)$.
4. [6 points] When a rectangle is observed under pinhole perspective, the image will be arbitrary quadrilateral, and figure 1 shows a projected rectangle. Answer the following questions using your newly learned skilled in working with homogeneous representations.

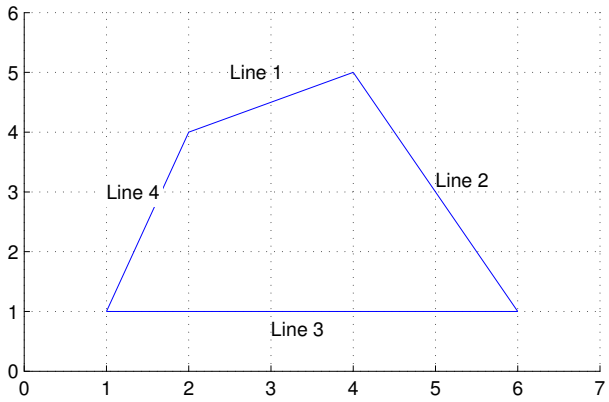

Figure 1: Illustration for problem 1

- Find the line equations (i.e. ax + by + c = 0) of the four edges
- Calculate the Euclidean coordinates of vanishing point for the image of each pair of parallel edge (lines 2&4 and lines 1&3).

Image formation and rigid body transformations [13 points]

In this problem we will practice rigid body transformations and image formations through the projective and affine camera model. The goal will be to 'photograph' the following four points given by ${}^{A}P_1 = (-1, -0.5, 2)^T$, ${}^{A}P_2 = (1, -0.5, 2)^T$, ${}^{A}P_3 = (1, 0.5, 2)^T$, ${}^{A}P_4 = (-1, 0.5, 2)^T$ in world coordinates. To do this we will need two matrices. Recall, first, the following formula for rigid body transformation

$$ {}^{B}P = {}^{B}_{A}R \; {}^{A}P + {}^{B}O_A \tag{1} $$

where ${}^{B}P$ is the point coordinate in the target (B) coordinate system, ${}^{A}P$ is the point coordinates in the source (A) coordinate system, ${}^{B}_{A}R$ is the rotation matrix from A to B, and ${}^{B}O_A$ is the origin of coordinate system A expressed in the B coordinates. The rotation and translation can be combined into a single 4×4 *extrinsic parameter* matrix, Pe, so that ${}^{B}P = Pe * {}^{A}P$. Once transformed, the points can be photographed using the *intrinsic camera* matrix, Pi which is a 3×4. Once these are found, the image of a point, ${}^{A}P$, can be calculated as $Pi * Pe * {}^{A}P$. We will consider four different settings of focal length, viewing angles and camera positions below. For each of these calculate:

- the extrinsic transformation matrix,
- Intrinsic camera matrix under the perspective camera assumption.
- Intrinsic camera matrix under the affine camera assumption.
- Calculate the image of the four vertices and plot using the supplied plotsquare.m function (see e.g. output in figure 2).

Note, that we highly recommend you solve this problem using MATLAB.

1. [**No rigid body transformation**]. Focal length = 1. The optical axis of the camera is aligned with the z-axis, and pointing in the positive direction.

2. [**Translation**] ${}^{B}O_{A} = (0, 0, 1)^{T}$. The optical axis of the camera is aligned with the z-axis.

3. [**Translation and rotation**]. Focal length = 1. ${}^{B}_{A}R$ encodes first a 45 degree rotation around the z-axis and then 60 degrees around the x-axis. ${}^{B}O_{A} = (0, 0, 1)^{T}$.

4. [**Translation and rotation, long distance**]. Focal length = 5. ${}^{B}_{A}R$ encodes a 45 degrees around the z-axis and then 60 degrees around the x-axis. ${}^{B}O_{A} = (0, 0, 10)^{T}$.

Note: we will not use a full intrinsic camera matrix (e.g. that maps centimeters to pixels, and defines the coordinates of the center of the image), but only parameterize this with f, the focal length. In other words: the only parameter in the intrinsic camera matrix under the perspective assumption is f, and the only ones under the affine assumption are: f, x_0, y_0, z_0, where x_0, y_0, z_0 is the center of the taylor series expansion.

In your report, include a image like Figure 2. Note that the axis *are the same for each row*, to facilitate comparison between the two camera models. Also include

1. The actual points around which you did the taylor series expansion for the affine camera models and how you arrived at these points.

2. A brief discussion on how do the projective and affine camera models differ? Why is this difference smaller for the last image compared to the second last?

Each correct image is worth 1 points (8 total). Each affine camera taylor series expansion point is worth .5 points (2 total). Presentation and discussion is worth 3 points.

Colors [8 points]

Write a matlab function imgOut = TransformHSV(imgIn,θ,s,v) to manipulate the color of an image in HSV color space. The function should do the following:

- Convert the image from RGB to HSV color space
- Apply a clockwise rotation of the hue channel by θ degrees
- Multiply the saturation channel by s
- Multiply the value channel by v
- Convert the transformed HSV image back into RGB color space

DO NOT use the built-in rgb2hsv or hsv2rgb functions in Matlab, except to compare your results. Test your function on the image color.bmp, and generate results for TransformHSV(img,80,1,1), TransformHSV(img,0,.5,1), TransformHSV(img,0,1,.5). Include the generated images in your writeup as a 2 by 2 grid using subplot.m.

1 Rendering [20 points]

In this exercise, we will render the image of a face with two different point light sources using a Lambertian reflectance model. We will use two albedo maps, one uniform and one that is more realistic. The face heightmap, the light sources, and the two albedo are given in facedata.mat (each row of the 'lightsource' variable encode a light location).

Note: Please make good use out of subplot.m to display related image next to eachother.

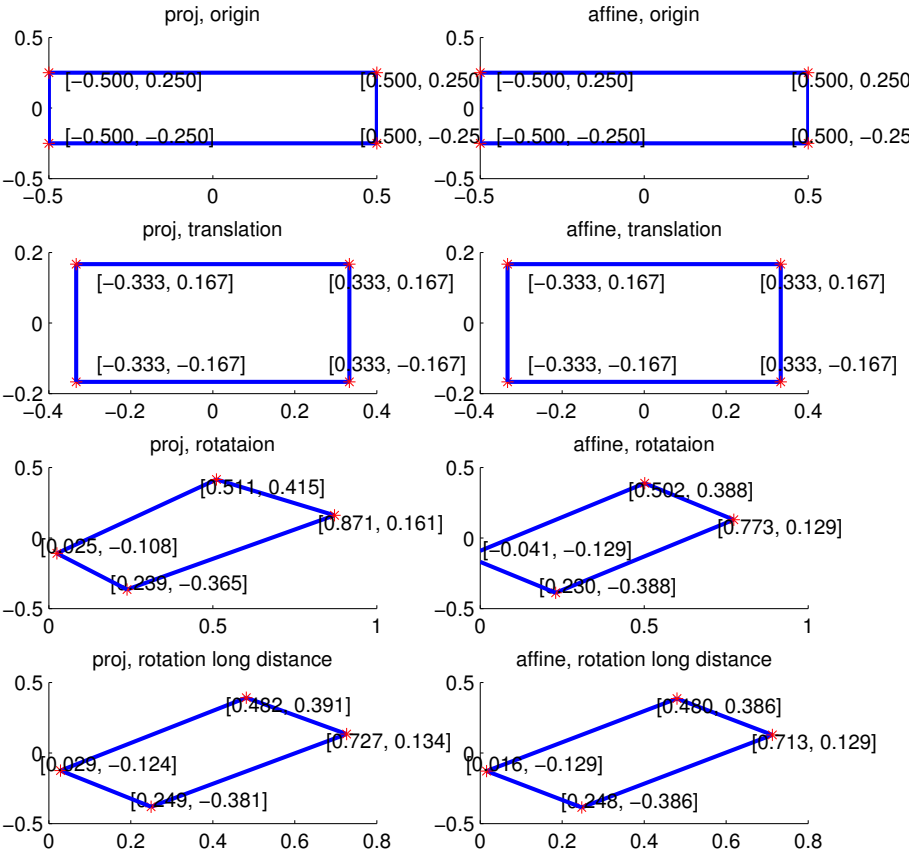

Figure 2: Example output for image formation problem. Note: the angles and offsets used to generate these plots are different from those in the problem statement, it's just to illustrate how to report your results.

1.1 Plot the face in 2-D [2 pts]

Plot both albedo maps using imagesc.m. Explain what you see.

1.2 Plot the face in 3-D [2 pts]

Using both the heightmap and the albedo, plot the face using surf.m. Do this for both albedos. Explain what you see.

1.3 Surface normals [8 pts]

Calculate the surface normals and display them as a quiver plot using quiver3.m. You may have to fiddle a bit with quiver3.m to make it look nice. Recall that the surface normals are given by

$$[-\frac{\delta f}{\delta x}, -\frac{\delta f}{\delta y}, 1]. \tag{2}$$

Also, recall, that each normal vector should be normalized to unit length.

1.4 Render images [8 pts]

For each of the two albedos, render three images. One for each of the two light sources, and one for both light-sources combined. Display these in a 2×3 subplot figure with titles. Recall that the general image formation equation is given by

$$I = a(x, y)\langle \hat{n}(x, y), \hat{s}(x, y)\rangle \frac{s_0}{d^2(x, y)} \tag{3}$$

where $a(x, y)$ is the albedo for pixel x, y, $\hat{n}(x, y)$ the surface normal, $\hat{s}(x, y)$ the light source direction, s_0 the light source intensity, $d(x, y)$ the distance to the light source, and $\langle \cdot, \cdot \rangle$ denotes the scalar product. Use imagesc.m to display these images. Let the light source intensity be 1 and do *not* make the 'distant light source assumption'.