CSE152A – Computer Vision – Assignment 1 (SP15)
Instructor: Ben Ochoa
Maximum Points : 55
Deadline : 11:59 p.m., Friday, 24-April-2015

Instructions:

- This assignment should be solved, and written up in groups of 3.
- Individual work is not allowed.
- There is no physical hand-in for this assignment.
- Coding for this assignment should be done in MATLAB
- All code developed for this assignment should be included in the appendix of the report.
- You may do problems on pen and paper; just scan and include it in the report.
- 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.
- Submit your assignment electronically by email to Akshat Dave `[akdave@ucsd.edu]` with the subject line *CSE152A-Assignment-1*. The email should contain one attached file named [`CSE_152A_HW1_<student1-id>_<student2-id>_<student3-id>.zip`]. This zip file must contain the following two artifacts:
  1. A pdf file named [`CSE_152A_HW1_<student1-id>_<student2-id>_<student3-id>.pdf`] containing your writeup. Please mention all the authors' full names and student identities in the report.
  2. A folder named [`CSE_152A_HW1_<student1-id>_<student2-id>_<student3-id>_code`] containing all your matlab code files

1 Warmup [10 points]

The purpose of this problem is to gain some familiarity with MATLAB programming. MATLAB is intuitive and easy to use! Even if you do not understand a command or a feature of the language, you can simply consult the reference manual that comes with the program. The following two tutorials are available for your reference:

- `http://cseweb.ucsd.edu/classes/wi13/cse152-a/hw0/matlab_intro.m`
- `http://www.math.utah.edu/lab/ms/matlab/matlab.html`.

You are required to write a program that does the following:

1. Reads in an image I.
2. Resizes the image I to I_s of dimensions 256×256 pixels using bilinear interpolation.
3. Tiles the image to form 4 quadrants where:
   - The top left quadrant is the resized image I_s
   - The top right is the red channel of the resized image (other channels set to zero)
   - The bottom left is the blue channel (other channels set to zero)
   - The bottom right is the green channel (other channels set to zero)

(a) (b)

Figure 1: (a) sample input image `applby.jpg` (b) sample output of the program on `appleby.jpg`

Test your program and present your results for the image `flag.jpg` provided on the course website [5 points]. A sample is shown for the image `appleby.jpg` in Fig. 1 (you need to include only the results of `flag.jpg`). Your program should be short (5 to 10 lines). Additionally, write a short paragraph explaining your results. Does your program produce the correct output? Does the red, green and blue channel separation make sense? [5 points]

2 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 $\tilde{\mathbf{l}} \cdot \bar{\mathbf{x}} = 0$, where $\tilde{\mathbf{l}} = (a, b, c)^T$ and $\bar{\mathbf{x}} = (x, y, 1)^T$. Noticing that $\bar{\mathbf{x}}$ is a homogeneous representation of $\mathbf{x} = (x, y)^T$, we can view $\tilde{\mathbf{l}}$ 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 $\tilde{\mathbf{l}} \cdot \bar{\mathbf{x}} = 0$.

$$\text{A point } \bar{\mathbf{x}} \text{ lies on the line } \tilde{\mathbf{l}} \Leftrightarrow \tilde{\mathbf{l}} \cdot \bar{\mathbf{x}} = \bar{\mathbf{x}} \cdot \tilde{\mathbf{l}} = 0 \qquad \texttt{(Statement 1)}$$

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 (using homogeneous coordinates) that follow from `(Statement 1)`:

   (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 2 shows a projected rectangle. Answer the following questions working with homogeneous representations.

   - 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 the opposite edge pairs (lines 2 & 4 and lines 1 & 3).

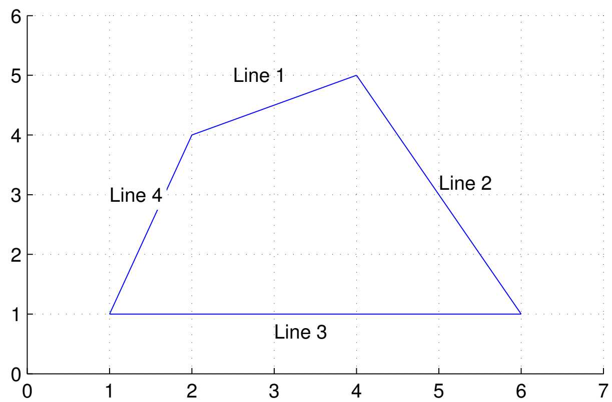

Figure 2: Illustration for problem 1

3 Image formation and rigid body transformations [10 points]

In this problem we will practice rigid body transformations and image formations through the pinhole projective camera model. The goal will be to 'photograph' the following four points given by $\boldsymbol{p}_{w_1} = (-1, -0.5, 2)^T$, $\boldsymbol{p}_{w_2} = (1, -0.5, 2)^T$, $\boldsymbol{p}_{w_3} = (1, 0.5, 2)^T$, $\boldsymbol{p}_{w_4} = (-1, 0.5, 2)^T$ in world coordinates. To do this we will need two matrices. Recall, first, the following formula for rigid body transformation:

$$\boldsymbol{p}_c = \boldsymbol{R}_{w,c}\ \boldsymbol{p}_w + \boldsymbol{O}_c \tag{1}$$

where $\boldsymbol{p}_c$ is the point position in the target (camera) coordinate system, $\boldsymbol{p}_w$ is the point position in the source (world) coordinate system, $\boldsymbol{R}_{w,c}$ is the rotation matrix from the w (world) frame to the c (camera) frame, and $\boldsymbol{O}_c$ is the origin of coordinate system w (world) expressed in the c (camera) coordinates. The rotation and translation can be combined into a single 4×4 *extrinsic parameter* matrix, $\boldsymbol{E}$, so that $\boldsymbol{p}_c = \boldsymbol{E}\ \boldsymbol{p}_w$. Once transformed, the points can be photographed using the *intrinsic camera* matrix, $\tilde{\boldsymbol{K}}$ which is a 3×4 matrix. Once these are found, the image of a point, $\boldsymbol{p}_w$, i.e. $\tilde{\boldsymbol{x}}_{\boldsymbol{s}}$, can be calculated as $\tilde{\boldsymbol{x}}_{\boldsymbol{s}} = \tilde{\boldsymbol{K}}\ \boldsymbol{E}\ \boldsymbol{p}_w$. 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 (pinhole) camera assumption.
- Calculate the image of the four vertices and plot using the supplied plotsquare.m function (see e.g. output in figure 3).

Camera Settings :

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**] $\boldsymbol{O}_c = (0, 0, 1)^T$. The optical axis of the camera is aligned with the z-axis.
3. [**Translation and rotation**]. Focal length = 1. $\boldsymbol{R}_{w,c}$ encodes first a 45 degree rotation around the z-axis and then 60 degrees around the x-axis. $\boldsymbol{O}_c = (0, 0, 1)^T$.
4. [**Translation and rotation, long distance**]. Focal length = 5. $\boldsymbol{R}_{w,c}$ encodes a 45 degrees around the z-axis and then 60 degrees around the x-axis. $\boldsymbol{O}_c = (0, 0, 10)^T$.

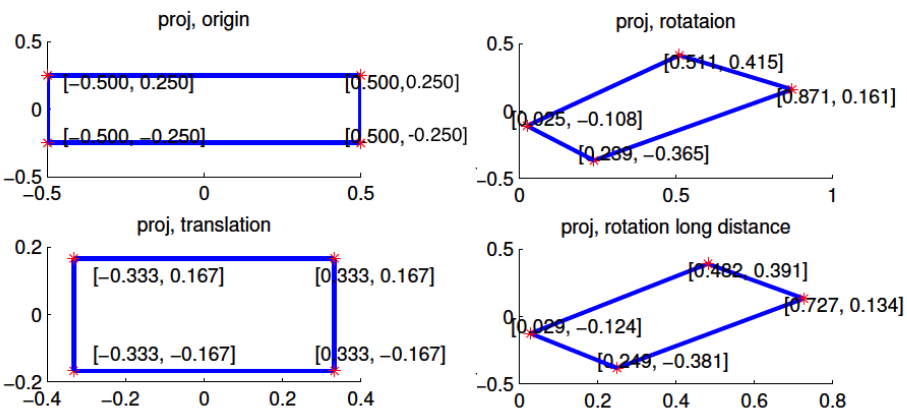

Figure 3: Example output for image formation problem. Note: the angles and offsets used to generate these plots are different from those in the problem statement, it's just to illustrate how to report your results.

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. In your report, include an image like Figure 3.

Each correct image is worth 2 points (4 images i.e. 8 points). Presentation and discussion is worth 2 points (Explaining why you observe any distortions in the projection, if any, under this model).

4 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 each other]

1. Plot the face in 2-D [2 points] : Plot both albedo maps using `imagesc.m`, explain the results
2. Plot the face in 3-D [2 points] : Using both the heightmap and the albedo, plot the face using `surf.m`. Do this for both albedos. Explain what you see.
3. Surface normals [8 points]: Calculate the surface normals and display them as a quiver plot using `quiver3.m`. Consider downsampling for better presentation. Recall that the surface normals are given by: $[-\frac{\delta f}{\delta x}, -\frac{\delta f}{\delta y}, 1]$. Also, recall, that each normal vector should be unitized.
4. Render images [8 points]: 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:

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

where $a(x,y)$ is the albedo for pixel (x,y), $\hat{n}(x,y)$ is the corresponding surface normal, $\hat{s}(x,y)$ is the light source direction, s_0 the light source intensity, $d(x,y)$ is 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'.