CSE152 – Introduction to Computer Vision – Homework 2

Instructor: Ben Ochoa

Due : Wednesday, April 26, 2017, 11:59 PM

Instructions:

- Review the academic integrity and collaboration policies on the course website.
- This assignment must be completed individually.
- Submit your PDF report on gradescope. You must prepare a report describing the problems, your solutions, and results. The report must contain enough information for a reader to understand the problems and replicate your work without also having the assignment. Make sure you include the source code with comments under Appendix listing at the end of your report.
- Programming aspects of the assignments should be completed using MATLAB or Python.
- Submit your zipped code folder electronically by email to lmelvix@eng.ucsd.edu, jsc078@eng.ucsd.edu, and dperoni@eng.ucsd.edu with the subject line **CSE152 Homework 2**. Make sure to include the **exact** subject line. The email should have one file attached. Name this file: CSE152_hw2_lastname_studentid.zip.
- All your source code should be in code folder with README.txt file explaining your code. For example, if you have **rotate_img.m**, your README.txt should have a detailed description: rotate_img.m: A function that takes an image matrix and degree as input and outputs the image rotated by the degree.
- The code is thus attached *both* as text in the appendix of the report and as m-files/py-files in the compressed archive.
- No physical hand-in for this assignment.

1 Geometry [20 points]

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

All points (x, y) that lie on the line $a\tilde{x} + b\tilde{y} + c = 0$ satisfy the equation $\mathbf{l}^\top \mathbf{x} = 0$.

$$\text{A point } \mathbf{x} \text{ lies on the line } \mathbf{l} \text{ if and only if } \mathbf{l}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{l} = 0 \ \textbf{Statement 1} \tag{1}$$

1. [4 points] Using Euclidean coordinates, find the equation of the line perpendicular to the family of lines $y = x + \lambda$ whereas $\lambda \in (-\infty, \infty)$ and at a distance d from the origin. Your answer should be represented only in terms of the given parameters.
2. [6 points] Prove the following two statements 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. [4 points] What is the line, in homogenous coordinates, joining the inhomogeneous points $(1, 4)$ and $(4, 5)$.

4. [6 points] When a rectangle $ABCD$ is observed under pinhole perspective, the image will be arbitrary quadrilateral $A'B'C'D'$. Answer the following questions using your newly learned skilled in working with homogeneous representations.

   - [3 points] For any arbitrarily imaged rectangle $ABCD$ with non zero area, can $A'B'C'D'$ ever be a non-convex quadrilateral? Explain the intuition behind your answer. (Note : A convex polygon is a simple polygon (not self-intersecting) in which no line segment between two points on the boundary ever goes outside the polygon.)
   - [3 points] Let $A' = (t, t)$, $B' = (t, 6t)$, $C' = (4t, 6t)$ and $D' = (2t, 4t)$ be the vertices of the image. Find all the vanishing points of the quadrilateral (i.e. the points of intersections of pairs of opposite lines through $\{A'B', C'D'\}$ and $\{B'C', A'D'\}$ given $t = 1$.

2 Image Formation[10 points]

In this problem we will practice rigid body transformations and image formations through the projective camera model. The goal will be to 'photograph' the four points $\tilde{\mathbf{X}}_1 = (-1, -0.5, 2)^\top$, $\tilde{\mathbf{X}}_2 = (1, -0.5, 2)^\top$, $\tilde{\mathbf{X}}_3 = (1, 0.5, 2)^\top$, $\tilde{\mathbf{X}}_4 = (-1, 0.5, 2)^\top$ in the world coordinate frame. First, recall the following formula for rigid body transformation

$$\tilde{\mathbf{X}}_{\text{Cam}} = \mathtt{R}\tilde{\mathbf{X}} + \mathbf{t} \tag{2}$$

where $\tilde{\mathbf{X}}_{\text{Cam}}$ is a point in the camera coordinate frame, $\tilde{\mathbf{X}}$ is a point in the world coordinate frame, and $\mathtt{R}$ and $\mathbf{t}$ are the rotation and translation that transform points from the world coordinate frame to the camera coordinate frame. Together, $\mathtt{R}$ and $\mathbf{t}$ are the *extrinsic* camera parameters. Once transformed to the camera coordinate frame, the points can be photographed using the 3×3 camera calibration matrix $\mathtt{K}$, which embodies the *intrinsic* camera parameters, and the canonical projection matrix $[\mathtt{I}|\mathbf{0}]$. Given $\mathtt{K}$, $\mathtt{R}$, and $\mathbf{t}$, the image of a point $\tilde{\mathbf{X}}$ is $\mathbf{x} = \mathtt{K}[\mathtt{I}|\mathbf{0}]\mathbf{X}_{\text{Cam}} = \mathtt{K}[\mathtt{R}|\mathbf{t}]\mathbf{X}$, where the homogeneous points $\mathbf{X}_{\text{Cam}} = (\tilde{\mathbf{X}}_{\text{Cam}}^\top, 1)^\top$ and $\mathbf{X} = (\tilde{\mathbf{X}}^\top, 1)^\top$. We will consider four different settings of focal length, viewing angles and camera positions below.

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. There are no orientations of the camera.

2. [**Translation**] Focal length = 1. $\mathbf{t} = (0, 0, 2)^\top$. The optical axis of the camera is aligned with the z-axis, and pointing in the positive direction. There are no orientations of the camera.

3. [**Translation and rotation**]. Focal length = 1. $\mathtt{R}$ encodes first a 60 degree rotation around the z-axis and then 45 degrees around the x-axis. $\mathbf{t} = (0, 0, 2)^\top$.

4. [**Translation and rotation, long distance**]. Focal length = 5. $\mathtt{R}$ encodes a 60 degrees around the z-axis and then 45 degrees around the x-axis. $\mathbf{t} = (0, 0, 15)^\top$.

For each of these settings, calculate and report:

(i) The matrix $[\mathtt{R}|\mathbf{t}]$ containing the extrinsic camera parameters,

(ii) The camera matrix $\mathtt{K}$ containing the internal camera paramters. Note: we will not use a camera calibration matrix that maps scene points to image points in pixels coordinates, but only parameterize this with the focal length f. In other words: the only parameter in the camera calibration matrix under the perspective assumption is f.

(iii) Calculate the image of the four vertices and plot using the supplied plotsquare.m function for MATLAB and plot_square.py for Python (see output in figure 1 or 2)

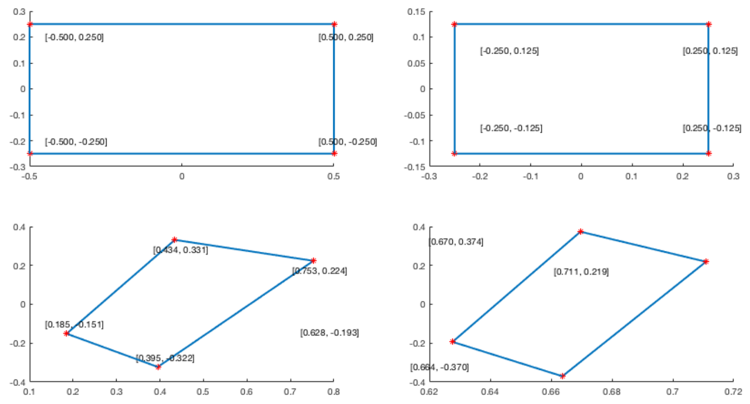

Figure 1: MATLAB example output for image formation problem. Note: the angles and offsets used to generate these plots may be different from those in the problem statement, it's just to illustrate how to report your results.

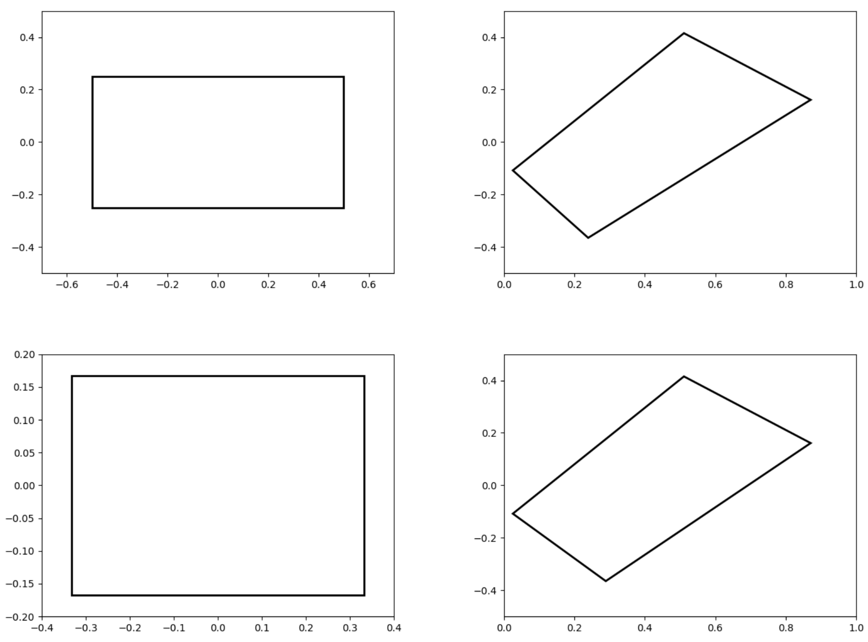

Figure 2: Python example output for image formation problem. Note: the angles and offsets used to generate these plots may be different from those in the problem statement, it's just to illustrate how to report your results.

In your report, include an image like 1 or 2. 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)

3 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 for MATLAB and facedata.npy for Python (each row of the 'lightsource' variable encode a light location). Sample code to access elements in facedata.npy can be found in access_facedata.py

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

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

Plot both albedo maps using imagesc.m for MATLAB and imshow for Python. Explain what you see.

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

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

3.3 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 for MATLAB and quiver in matplotlib.pyplot in Python to make it look nice. Recall that the surface normals are given by

$$[-\frac{\delta f}{\delta x}, -\frac{\delta f}{\delta y}, 1]. \tag{3}$$

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

3.4 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

$$I = a(x,y)\hat{\mathbf{n}}(x,y)^\top \hat{\mathbf{s}}(x,y)\frac{s_0}{(d(x,y))^2} \tag{4}$$

where $a(x,y)$ is the albedo for pixel (x,y), $\hat{n}(x,y)$ is the corresponding surface normal, $\hat{s}(x,y)$ the light source direction, s_0 the light source intensity, $d(x,y)$ the distance to the light. Let the light source intensity be 1 and do *not* make the 'distant light source assumption'.
For MATLAB, use imagesc.m or imshow.m to display these images(set the colormap to gray using **colormap('gray')**)
For Python imshow with appropriate keyword arguments .