ECE4580 Homework #3

Due: Feb. 02, 2017

On t-square you should be able to find the Matlab files needed for this assignment. Much of the problems can be solved for in Matlab. If you write the proper kinds of functions, then you can perform the calculations real easily. For example, my solution involved writing a function called `isVisible` which computes the visibility calculation for me.

Problem 1. (15 pts) You are given that, with respect to the world, a camera is translated and rotated by

$$T_C^W = \begin{Bmatrix} -10.0000 \\ 15.0000 \\ -5.0000 \end{Bmatrix} \quad \text{and} \quad R_C^W = \begin{bmatrix} 0.8365 & 0.2241 & 0.5000 \\ 0.0173 & 0.9012 & -0.4330 \\ -0.5477 & 0.3709 & 0.7500 \end{bmatrix}$$

Furthermore, the z-axis is the optical axis of the camera, which points in the direction of positive z. Also, the horizontal field of view is 90 degrees while the vertical field of view is 60 degrees. This means that relative to the xz-plane, the angle the point makes in that plane should be within ± 45 degrees. Likewise, for the yz-plane, the limits are ± 30 degree.

Determine the following:

(a) Where are the following points, given in the world frame, when considered with respect to the camera frame,

$$q_1^W = \begin{Bmatrix} 6.6000 \\ -1.7000 \\ 7.6000 \\ 1.0000 \end{Bmatrix}, \quad q_2^W = \begin{Bmatrix} -7.2000 \\ 10.5000 \\ 14.5000 \\ 1.0000 \end{Bmatrix}, \quad q_3^W = \begin{Bmatrix} -41.1000 \\ 10.7000 \\ -13.4000 \\ 1.0000 \end{Bmatrix}, \quad q_4^W = \begin{Bmatrix} -13.5000 \\ 18.3000 \\ -7.4000 \\ 1.0000 \end{Bmatrix}.$$

(b) Determine which points are in the field of view and which aren't.

(c) Consider the opposing case. Given two points located in the camera frame at

$$p_5^C = \begin{Bmatrix} 2.8000 \\ 1.4000 \\ 16.0000 \end{Bmatrix}, \quad p_6^C = \begin{Bmatrix} 13.1000 \\ -11.3000 \\ 28.0000 \end{Bmatrix},$$

where are they in the world frame.

Note: To avoid having to enter the matrices and vectors into Matlab, there is a Matlab file `visible02.mat` *in the Assignment files.*

Problem 2. (15 pts) Suppose that you have a stereo rig setup such that the left and right cameras, with respect to the world frame are:

$$g_L^W = \left[\begin{array}{ccc|c} 0.914 & -0.064 & 0.402 & -8.659 \\ 0.288 & 0.799 & -0.527 & 2.170 \\ -0.288 & 0.597 & 0.749 & 4.830 \\ \hline \multicolumn{3}{c|}{0} & 1 \end{array}\right] \quad g_R^W = \left[\begin{array}{ccc|c} 0.995 & -0.016 & 0.103 & 10.659 \\ 0.074 & 0.808 & -0.584 & 5.830 \\ -0.074 & 0.588 & 0.805 & 1.170 \\ \hline \multicolumn{3}{c|}{0} & 1 \end{array}\right]$$

and the following three (homogeneous) points in the world frame are to be imaged

$$q_1^W = \left\{\begin{array}{c} 16.000 \\ -25.400 \\ \hline 19.100 \\ \hline 1 \end{array}\right\}, \quad q_2^W = \left\{\begin{array}{c} 78.300 \\ -20.900 \\ \hline 7.300 \\ \hline 1 \end{array}\right\}, \quad q_3^W = \left\{\begin{array}{c} 33.100 \\ -39.100 \\ \hline 38.500 \\ \hline 1 \end{array}\right\}.$$

Answer the following:

(a) What is the transformation giving the camera's right frame relative to the camera's left frame?

(b) What are the coordinates of the points given in both of the camera frames?

(c) Supposing that both cameras have horizontal and vertical fields of view of 60° and 40°, respectively. For each point, specify if it is visible to left only, right only, both, or none.

Note: To avoid having to enter the matrices and vectors into Matlab, there is a Matlab file `stereo01.mat` *in the Assignment section.*

Problem 3. (15 pts) Let's get a grips on these rays versus points and their conversion. Suppose that you know both the intrinsic camera projection matrix and the extrinsic camera information (here given as the camera frame relative to world coordinates):

$$\Psi = \begin{bmatrix} 250.000 & 0.000 & 240.000 \\ 0.000 & 250.000 & 320.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \quad \text{and} \quad g_C^W = \left[\begin{array}{ccc|c} 0.000 & 0.707 & 0.707 & 4.000 \\ -0.866 & 0.354 & -0.354 & 3.000 \\ -0.500 & -0.612 & 0.612 & 2.000 \\ \hline & 0 & & 1 \end{array}\right]$$

then, work out the following:

(a) Give the ray (in camera units/pixels) that the point $p_1^C = (24.0000, -9.5000, 25.0000)^T$ lies on. Also give it as a ray with the last element normalized (this would be the actual pixel location in the image that the point projects to, modulo the quantization).

(b) Give the ray (in camera units/pixels) that the point $p_2^W = (12.8927, -78.3964, 10.5141)^T$ lies on. Please take care that this point is in world coordinates.

(c) Give the ray (in world units) that the image point $\vec{r}_3 = (251.0000, 57.0000)^T$ maps to when back-projecting the pixel back out into the world. Back-projection is the partial inversion process discussed in class that maps the 2D image point back out to a 3D ray. Let the ray be given in camera frame coordinates. Don't worry about transforming it to the world frame, as that will come later (of course, if you give a little thought as to how it would turn out, it would be quite helpful towards a future homework).

Note: You should find a file called `rays01.mat` *in the Assignment section with the matrices and vectors.*

Problem 4. (10 pts) For the kinect depth image, there were some issues in grabbing only the person. One way to do this is to use a *distance transform*. By combining the original troublesome binary image of the person with a properly thresholded version of a distance transform image, the person can be extracted. There are some pretty clever ways to do this, but Matlab doesn't quite allow for them to happen, so this problem involves using a slower but equivalent method.

(a) What is the *distance transform*? In aswering this tell me about two possible distance types that can be used.

(b) In the first instantiation of this problem, you used `bwselect` to select a point and then region grow from it. However, there is a cooler function that not only selects, but also computes the distance transform on the pixels that get selected. The Matlab function is called `bwdistgeodesic`. Read about how to use it using Matlab's online documentation or invoke "`help bwdistgeodesic`" from within Matlab. Using "`doc bwdistgeodesic`" also works if I am not mistaken.

Properly use Matlab's `bwdistgeodesic` function to create the distance transformed version of your original binary image. Turn in the distance image.

(c) Figure out a good threshold for the distance transformed image (I would imagine half the height of the person would work) and provide that as an answer to this part. Applying the threshold to the distance image will provide a binary image with points that should be considered potential candidates. Also turn in this image.

(d) Comment on how effective this technique is at getting the whole body. You like or don't like?

Problem 5. (15 pts)**.** This problem is a baby problem that shows two different techniques for solving an unknown matrix given a set of input/output observations of the matrix. While they should give the same answers, one seems to be preferred over the other due to it being more general. Recall from class, that the main idea revolves around having the input, $\vec{x}$, and the output $\vec{y}$ values available, but not the matrix A such that

$$\vec{y} = A\vec{x}.$$

Suppose that A is a 2×2 matrix, then the full form looks like

$$\begin{Bmatrix} y^1 \\ y^2 \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{Bmatrix} x^1 \\ x^2 \end{Bmatrix}.$$

If we want to solve for the unknown matrix coefficients given observed $\vec{x}$ and $\vec{y}$ values, it is not possible because there are two equations (one for each coordinate of $\vec{y}$) and four unknowns! It is more obvious if the two equations are re-written to look like

$$\begin{Bmatrix} y^1 \\ y^2 \end{Bmatrix} = \begin{bmatrix} x^1 & x^2 & 0 & 0 \\ 0 & 0 & x^1 & x^2 \end{bmatrix} \begin{Bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{Bmatrix}.$$

One way to get the problem to work out is to add more observations. Every set of observations adds two more equations. Here we would need two observation pairs to get a complete set of equations,

$$\begin{Bmatrix} y_1^1 \\ y_1^2 \\ y_2^1 \\ y_2^2 \end{Bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & 0 & 0 \\ 0 & 0 & x_1^1 & x_1^2 \\ x_2^1 & x_2^2 & 0 & 0 \\ 0 & 0 & x_2^1 & x_2^2 \end{bmatrix} \begin{Bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{Bmatrix}. \tag{1}$$

where $\vec{x}_1$, $\vec{x}_2$, $\vec{y}_1$, and $\vec{y}_2$ are the observations. Note that we can simply solve this system of equations directly by employing the inverse of the matrix.

In computer vision it is more common to see manipulated versions of the above. Let the above equation be written in compact form as

$$\overrightarrow{Y} = X(\vec{x}_1, \vec{x}_2)\vec{a}.$$

where $\overrightarrow{Y}$ is the super vector of concatenated observations $\vec{y}_i$. Moving all of the elements to one side leads to

$$0 = X(\vec{x}_1, \vec{x}_2) \cdot \vec{a} - \overrightarrow{Y} \cdot 1$$

where the multiplication by 1 was intentionally placed there. The symbol $\cdot$ in these cases means multiplication. The reason for writing it as above, with the $\cdot$ too, is that the above equation can be converted to homogeneous form:

$$0 = \left[\begin{array}{c|c} X(\vec{x}_1, \vec{x}_2) & -\overrightarrow{Y} \\ \hline 0 & 0 \end{array}\right] \begin{Bmatrix} \vec{a} \\ 1 \end{Bmatrix} \tag{2}$$

The matrix is a $5x5$ matrix and the vector is a $5x1$. The solution to the problem is obtained from the singular value decomposition that was explored in an earlier homework. The solution is actually the right vector of the smallest singular value (i.e., the last column of the V matrix from the singular value decomposition). Actually, that is not 100% true. The last element in the last column most likely won't be a 1 even though we know it should be from equation (2). Therefore the solution needs to be rescaled so that the last element in the vector is equal to 1. Once that's done, the solution can be extracted from the first four coordinates of the vector.

We will explore these equations in the context of stereo triangulation, camera calibration, and possible camera motion estimation. Meanwhile, let's try to solve this baby version:

(a) Given the observations:

$$\vec{x}_1 = \begin{Bmatrix} 4.9000 \\ 0.2000 \end{Bmatrix}, \quad \vec{y}_1 = \begin{Bmatrix} 66.0000 \\ 54.6600 \end{Bmatrix}, \quad \vec{x}_2 = \begin{Bmatrix} -2.6000 \\ 1.8000 \end{Bmatrix}, \quad \text{and} \quad \vec{y}_2 = \begin{Bmatrix} -41.1200 \\ -21.7600 \end{Bmatrix},$$

use the direct solution from equation (1) to solve for the unknown matrix A.

(b) Find the solution to A but using the singular value decomposition approach for the matrix in equation (2).

Recall that you can use the `reshape` function to get a matrix out of the vector. You will need to transpose the result. I asked for both techniques since you can use them as a sanity check; they should give the same answer (or numerically very close). The second approach is related to a technique known as the Direct Linear Transformation (DLT) method.