Assignment 3

CS283, Computer Vision
Harvard University

Due Friday, Sept. 29, at 5:00pm

This problem set explores camera matrices and binocular geometry. As usual, the assignment will be submitted electronically and formatted according to the guidelines. Notice the Hints and Information on the end of the second page.

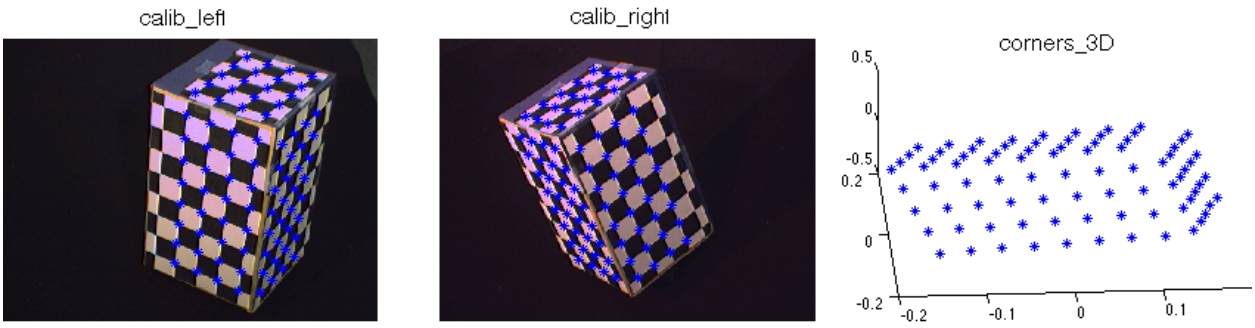
1. *(10 points)* In class we saw that a camera matrix satisfies the equation $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$, and HZ Section 7.1 shows that six 3D-2D matches $\mathbf{x} \leftrightarrow \mathbf{X}$ are sufficient to recover $\mathbf{P}$ using a linear (non-iterative) algorithm.

   (a) Create and describe linear algorithms for computing the camera matrix $\mathbf{P}$ in the special cases when: i) the camera location (but not orientation) is known, and ii) the camera location and complete orientation are known.

   (b) Ignoring degenerate configurations, how many 2D-3D matches are required for there to be a unique solution in each case? Justify your answers.

2. *(10 points)* Let I_0 be a projective image and I_1 be an image of I_0 (an image of an image). Let the composite image be denoted I'. Show that the apparent camera center of I' is the same as that of I_0. Speculate on how this explains why a portrait's eyes "follow you around the room." Verify on the other hand that all other parameters of I' and I_0 may be different. *Hint*: The null space of an $n \times n$ invertible matrix $\mathbf{A}$ is empty, i.e., $\mathbf{A}\mathbf{x} = 0$ if and only if $\mathbf{x} = 0$.

3. *(30 points)* This assignment's data folder contains two stereo pairs of images as well as a .MAT file containing some corresponding image-point coordinates and world-point coordinates. (See Hints and Information below on how to use .MAT files.) In this problem you will calibrate the stereo rig, and in the next problem, you will use this calibration information to reconstruct a 3D object. The calibration data is shown in the figure below.



   (a) Write a function `P=getCamera(X3,X2)` that computes a 3×4 camera matrix `P` from an $n \times 3$ matrix of world coordinates `X3` and an $n \times 2$ matrix of pixel coordinates `X2` representing the projection of the world points into a single projective camera. Your function should use the "Linear Solution" with normalization/denormalization, as described in Algorithm 7.1 of HZ (the iterative refinement based on geometric error is not necessary). See the skeleton script.

   (b) The file `corners.mat` contains two $n \times 2$ matrices `leftpts` and `rightpts` storing the pixel coordinates of the checkered box corners observed in the images `calib_left.bmp` and `calib_right.bmp`. It also contains an $n \times 3$ matrix `corners_3D` storing the world coordinates of the very same box

corners. Using your function from part (a), estimate camera matrices $\mathbf{P}_l$ and $\mathbf{P}_r$ for the left and right cameras in this stereo pair. Submit your Matlab script, which should begin with the command `load ./data/corners.mat`. See the skeleton script.

(c) To display the results, create a 3D plot showing the 3D points in `corners.mat` along with a depiction of the position, orientation, and field of view of your two cameras. (See Hints and Information below.) Make sure your Matlab code is well-commented, and have both your code and plots in your .MLX file. Store your camera matrices $\mathbf{P}_l$ and $\mathbf{P}_r$ in variables named `Pl` and `Pr` (for use in Question 4).

4. *(25 points)* The images `kleenex_left.bmp` and `kleenex_right.bmp` comprise another stereo pair captured by the same rig that was calibrated in Question 3. Here, you will use the calibration information obtained in that problem to reconstruct a 3D model of the Kleenex box.

(a) When the two cameras in a stereo pair are calibrated and a left-right point match $\mathbf{x} \leftrightarrow \mathbf{x}'$ is given, the corresponding world point $\mathbf{X}$ is recovered by back-projecting the rays through pixels $\mathbf{x}$ and $\mathbf{x}'$ and finding their point of intersection in 3-space. This process is referred to as *triangulation*, and in the presence of noise, it can be solved using a linear algorithm. The desired point $\mathbf{X}$ must satisfy the equations $\mathbf{x} = \mathbf{P}\mathbf{X}$ and $\mathbf{x}' = \mathbf{P}'\mathbf{X}$, and these can equivalently be written as

$$\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0 \quad \text{and} \quad \mathbf{x}' \times (\mathbf{P}'\mathbf{X}) = 0,$$

which are linear in the components of $\mathbf{X}$. Show that this system of equations can be written in the form $\mathbf{A}\mathbf{X} = \mathbf{0}$, with the elements of $\mathbf{A}$ given explicitly in terms of $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{P}$ and $\mathbf{P}'$.

(b) Write a function `X=triangulate(x1,x2,P1,P2)` that takes camera matrices `P1`, `P2` and image points `x1`, `x2` and estimates the 3D world point `X` by solving the homogeneous matrix equation developed in part (a). See the skeleton script.

(c) Manually identify the pixel coordinates of the seven visible corners of the Kleenex box in the images `kleenex_left.bmp` and `kleenex_right.bmp`. For each of these seven correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, use your function from part (b) along with the camera matrices `Pl` and `Pr` to find the world coordinates of the corresponding 3D points through triangulation. Estimate the position of the eighth (unseen) corner by assuming symmetry in the box. Create a 3D plot of a wireframe of the box along with the same graphical camera representations used in Question 3(c).

(d) Given that the dimensions of each square in the checker board pattern used for calibration are 2cm×2cm, estimate the volume of the kleenex box.

Bonus Question. *(10 points)* Exploit the fact that the box is polygonal and symmetric to obtain a complete 3D model using the eight points recovered in Prob. 4. The box consists of six planar facets $\Pi_k,\ k = 1 \ldots 6$, and for each facet, there are necessarily homographies between it (as a world plane) and each of the two image planes of the stereo rig. By estimating these homographies, the texture for each (visible) facet can be extracted from the images and "mapped on" to a 3D model.

Create a 3D model with texture taken from the images and mapped onto each facet. (Unobserved facets can be hallucinated by assuming symmetry.) The result can be displayed using repeated calls to the `surf` command as described below. Submit your code along with three views of the 3D model in the .MLX file.

Hints and Information

- .MAT files are binary data container format used by MATLAB. A .MAT file may include matrices, arrays, scalars or even functions. To save some variables in current workspace into a .MAT file, use the following command

```
>> save('filename', 'var1', 'var2', ...);
```

This will create a `filename.mat` file in your current directory, which can be loaded later (say when you exit and reopen MATLAB next time). The loading command is

```
>> load('filename');
```

This will dump the variables you saved last time (`var1`, `var2`, ...) into the current workspace.

- A convenient graphical representation for a camera is a pentahedron. One vertex of the five-sided polygon is placed at the camera center, and the other four vertices are found by back-projecting rays through the four corners of the image and placing a point along each ray at a fixed distance d from the camera center. When these vertices are connected, the camera looks like a pyramid, and it's position, orientation, and field of view can all be readily observed.

  Recovering the camera center and other useful information from a camera matrix is discussed in Sect. 6.2 of HZ. This is an important section to read. In Matlab, a polygon can be drawn as a wireframe using the `plot` command or as a solid using the `patch` command.

- In Matlab, the `surf` command can be used to draw a rectangular facet with an image texture-mapped onto it. This is done by setting the `FaceColor` property of the surface to `'texturemap'` and the `CData` property to a color or grayscale image array. (Attention must be paid to the data type used for the image array. Consult the online documentation for details regarding the handle properties for a Matlab surface.) As an example, the following code loads an image and texture-maps it onto a rectangular facet with top-left corner $(0, 1, 0)$ and bottom-right corner $(1, 0, 0)$.

  ```
  im=imread('../data/calib_left.bmp');
  s=surf([0 1; 0 1],[0 0; 1 1],[0 0; 0 0]);
  set(s,'facecolor','texturemap','cdata',im);
  ```

- Matlab figures can be saved in .FIG files and later re-opened. This can be done on the command line. For example, `saveas(gca,'myfig.fig','fig')` saves the current figure to a file, and `open('myfig.fig')` opens that figure into a new figure window.