CSSE 461 – Computer Vision
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Problem Set 5

This problem set is due 29 April 2016.

This document contains hyperlinks and is best viewed as html.

1 Calibrated Reconstruction

Using images of Myers and the results from Problem Set 4 construct a textured 3D model of the scene.

Note: A set of correspondences, polygons, extracted textures and some vrml models for the shed demo are posted under the handouts link (Day18). Testing on the shed data can help identify errors in your code.

1. Repeat the steps from Problem Set 4 as necesary until good data is obtained.

   If you are having trouble verifying the fundamental matrix, **see me for assistance**.

Note: The next 2 steps are interdependent. You should select one of the possible combinations of $\mathbf{P}$ and $\mathbf{P'}$, recover the 3D coordinates using that combination via Step 3, and then check if the combination is correct via Step 2.

2. Construct the four possible solutions for the camera matrices $\mathbf{P}$ and $\mathbf{P'}$ (see result 9.19 on pg. 259) and determine which of them is correct.

Note: $\mathbf{P'}(\mathbf{3},\mathbf{3})$ must be greater than or equal to 0. If not, multiply $\mathbf{P'}$ by -1.

Remember that the correct solution has all of the 3D points (see next step) in front of both cameras. After you've recovered the 3D points, you can simply check the z-coordinate for one camera. You must rotate and translate the points into the coordinate frame of the other camera to complete the check.

Turn in: Save your $\mathbf{P}$, and $\mathbf{P'}$ matrices and your set of 3D points as $\mathbf{X}$ in a matlab file named "`structure.mat`".

3. Reconstruct the 3D coordinates of the correspondences using

$$\begin{bmatrix} x\mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} - \mathbf{p}^{2\top} \\ x'\mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \\ y'\mathbf{p}'^{3\top} - \mathbf{p}'^{2\top} \end{bmatrix} \mathbf{X} = 0$$

   where $\mathbf{p}^{i\top}$ is the i^{th} row of $\mathbf{P}$.

   You can use `plotPoints.m` (which depends on `view3d.m`) to view the reconstructed points.

4. In order to build a good model, you will probably want to augment your set of 3D points.

   - If the vertex you'd like to add is visible in both images, simply reconstruct it using the location in the two images as already done.
   - If the point is visible in only one of the images, you may be able to recover the 3D point by intersecting the line containing the image point with a 3D plane formed from points that are visible in both images.
   - If the point is not visible in either image, you still may be able to recover the 3D point by intersecting two 3D lines that are formed from points that are visible in both images.

5. Identify the set of polygons that will form your model. `showPoints.m` might be useful to identify the index of the vertices for the polygons. Create a cell array containing one array for each polygon. The polygon arrays contain one index[1] for each vertex. **Be sure your polygons are facing toward the camera!** The right-hand rule determines the front (visible) face of the polygon.

6. Use `makeWireframe.m` and `makeUntexturedModel.m`[2] to create a vrml model of your reconstruction. The matlab virtual reality toolkit supports display of vrml files. Use the matlab commands `vrworld`, `open`, `view`, and `close` to view your vrml model. Alternately, you can use any vrml viewer to view your model.

7. Extract rectified (viewed from perpendicular or fronto-parallel) textures for each polygon. Save the texture to an image using `imwrite` and save the texture coordinates (the coordinates of the vertices in the rectified image). You may find this `texture extraction skeleton` helpful.

   If you are having trouble extracting rectified textures, **see me for assistance**.

8. Build a cell array of the texture files names and the texture coordinates corresponding to each polygon in the cell array of polygons.

[1] These indexes should be 1 based, i.e. the first point is index 1. `vrml` uses 0-based arrays. The scripts provided make the conversion for you.

[2] If you're having trouble viewing the untextured model, but the wireframe displays fine, your polygons may be facing the wrong way. You might try adding "solid FALSE" immediately before each "coordIndex" statement.

9. Finally, build a texture vrml model of your reconstruction using `makeTexturedModel.m`.

Turn in: Save your vrml model. You will need to include the texture files with your model.

2 Turning it in

Turn in your vrml model, the items requested above, and a brief discussion of your experiences in electronic form using using svn. Your materials should be placed in the `ProblemSet5` directory of your class repository (`http://svn.csse.rose-hulman.edu/repos/1516c-csse461-<your username>`).