Computer Vision Lab 4: 3D Reconstruction

Robin Khatri, M1 MLDM

02 April 2019

Contents

1 Introduction

During the fourth lab session, the primary objective was to reconstruct a 3D scene from a pair of images taken from different camera positions. The computations involved calculating intrinsic and extrinsic parameters thorugh Tsai method. Those parameters (for individual cameras) were then used to construct a global transformation matrix to get world co-ordinates from selected points in image. The process involved calibration within the stereo setup. The process is detailed in the coming sections.

The pair of images from stereo setup that were used in this lab session are presented in figure 1.

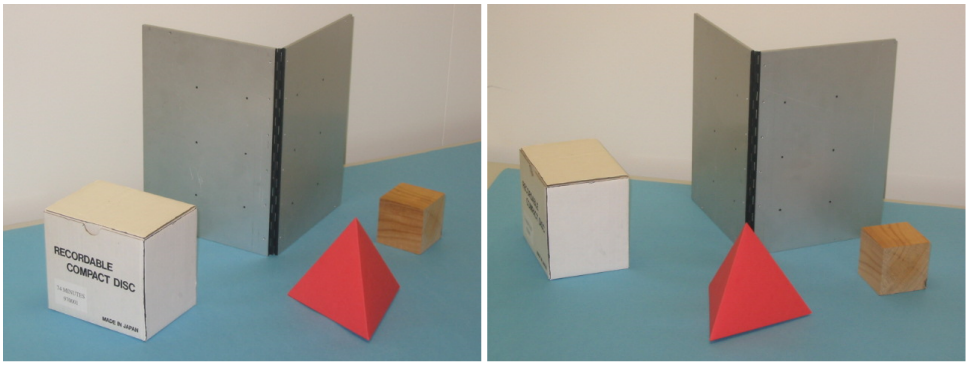

Figure 1: Images used (from stereo setup)

We reconstructed every element (2 cubes, 1 pyramid and a book-cover shaped object) from these images.

2 Verifying parameters of camera calibration matrix

The first step was to re-project 3D points into 2D space. A part of this code (generating calibration matrix) was already included in the main file of this exercise, which is `Let_me_reconstruction_3D.m`. The points used for calibration were also given in three files: `XYZ.mat`, `xyleft.mat` and `xyRight.mat`. The calibration was done using function calibTSAI().

The projection matrix was calculated using $M = K \times RT$, where both K and RT were obtained from calibration. The resulting matrix was stored in P1Tsai and P2Tsai, corresponding to left and right camera poses respectively. These matrices were then multiplied with the given 3D points (in $XYZ.mat$) to produce 2D projected coordinates. Since there was an extra dimension and so to exclude the scaling, the first and second elements of the generated 2D coordinates were divided by the last element (Z-co-ordinate) of every point. The resulting positions of these points in the images are presented in figure 2. Figure 2 confirms the accuracy of calibration parameters.

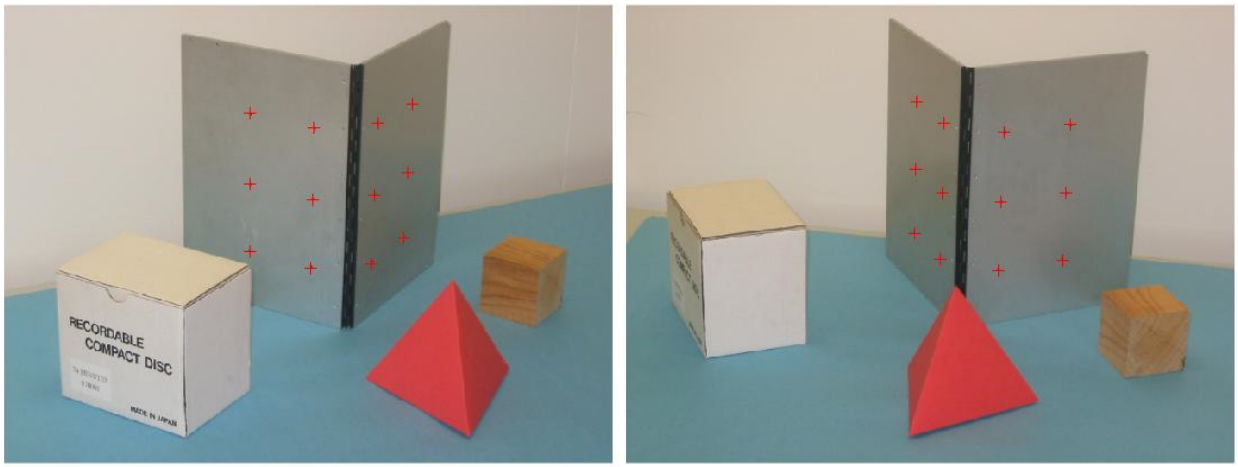

Figure 2: Reprojected points from 3D to 2D (in red)

3 Selecting points for projection to 3D

We selected 4 point from corners of pyramid, 8 points for both the cube and the rectangular box, and 6 points to represent book-cover-shaped large object. Please note that another way to represent the book-cover-shaped object could be to select 4 points and not project all the corners. However, we imagined that it would be a better visual representation if we do it for the complete shape.

In the code provided in `Let_me_reconstruction_3D.m`, there was already code written for selecting the points. However that code did not show the selected point over the image, therefore liberty was taken in slightly modifying it to show the points that were selected. Selected points are presented in figure 3.

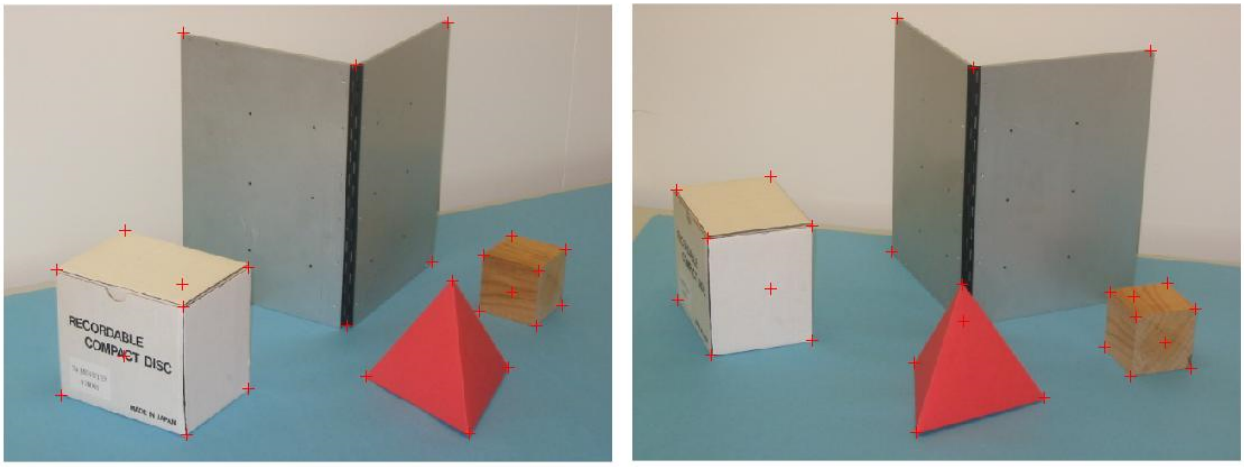

Figure 3: Selected points for each of the object (In red)

Selected point for each object were saved in a seperate file for later use. The points for pyramid are stored in `pyrl.mat and pyrr.mat`, the points for cube is stored in `cubel.mat` and `cuber.mat`, for rectangular box in in `rectl.mat` and `rectr.mat` and for book-cover-shaped object, the points are stored in `bookl.mat` and bookr.mat. (l for the left image and r for the right image)

The co-ordinates of selected points (In homogeneous) are given below:

Pyramid:

```
>> Ppyrlpxhom % In left image
Ppyrlpxhom =
  473.0557  381.8368  490.0413  531.5616
  290.1986  390.8539  451.2471  380.7883
    1.0000    1.0000    1.0000    1.0000
>> Ppyrrpxhom % In right image
Ppyrrpxhom =
  347.2366  296.9089  298.7962  434.0518
  295.8604  377.6429  451.2471  414.1304
    1.0000    1.0000    1.0000    1.0000
```

Cube:

```
>> Pcubelpxhom % In left image
Pcubelpxhom =
  505.1396  563.6455  560.5000  501.3650  534.7071  591.9548  588.1802  535.3362
  263.7765  279.5039  338.0098  322.9115  243.0164  258.1147  316.6206  302.1514
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
>> Pcuberpxhom % In right image
Pcuberpxhom =
  503.2523  530.3034  524.6415  496.9613  563.0164  593.2130  588.1802  561.1291
  303.4096  329.2025  391.4830  361.2864  294.6022  321.6533  383.3047  356.2536
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Rectangular box:

```
>> Prectlpxhom % In left image
Prectlpxhom =
   56.5944  189.9626  193.1081   60.9980  128.9404  258.5341  258.5341  127.0531
  278.2457  317.8788  452.5052  410.9849  237.9836  276.3585  403.4358  369.4646
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
>> Prectrpxhom % In right image
Prectrpxhom =
   46.5288   80.5000   83.6455   48.4161  146.5550  189.3336  189.9626  145.9260
  196.4633  247.4201  369.4646  311.5878  181.9941  233.5799  354.3663  299.6350
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Bookcover shaped object:

```
>> Pbooklpxhom % In left image
Pbooklpxhom =
  189.3336  371.1422  467.3938  451.0374  361.7058  188.7045
```

```
   30.3820   63.7241   19.0583  270.6966  338.0098  293.9731
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
>> Pbookrpxhom % In right image
Pbookrpxhom =
  279.2942  359.8185  546.0308  527.7870  349.1239  273.0033
   16.5419   67.4987   50.5131  307.8132  334.2353  260.6311
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Note:

If you wish to do manual selection of the points instead of using the saved points, it is important to remember that the objects need to be selected in following order: First the pyramid, then cubes (cubes in any order) and lastly, the large bookcover shaped object. It is also important to remember that while selecting the points for bookcover shaped object, the points need to be selected clockwise. This is to avoid connecting points which do not share an edge. For assistance see figure 4.

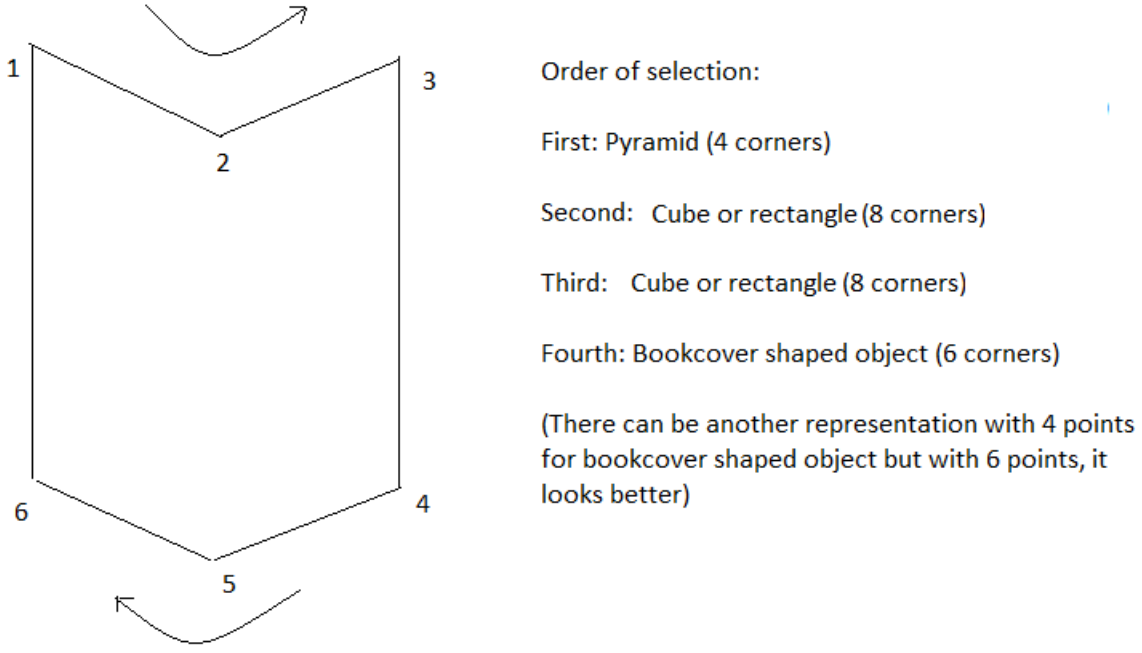

Figure 4: Recommended order of selecting points for 3D visualization later

4 Calculating Camera co-ordinates

Two matrices `K1_sf` and `K2_sf` were computed from individual camera calibration matrices, these are 3×3 matrices and are obtained by removing last columns of K1 and K2 respectively.

We obtained camera coordinates for each of above selected points by using $K1\_sf^{-1}$ multiplied by ar-

ray of selected points (presented in previous section). The points are stored in separate files named for ease as `cam_pyr_l,mat, cam_pyr_r.mat` etc.

5 3D Reconstruction

5.1 Global rotation and translation matrices

To calculate the global rotation and translation matrices, following equations were used:

$$R = R_r R_l' \tag{1}$$

and

$$T = T_l - RT_r \tag{2}$$

Where R and T are global rotation and translation matrices respectively.

5.2 Calculating 3D camera points

To compute 3D camera coordinates, we used algebric and matrix operations for the calibration matrix. Essentially, the following equations were used:

$$P = P_1 + 0.5 * (P_2 - P_1) \tag{3}$$

Where

$$P_1 = a_0 P_l \tag{4}$$

and

$$P_2 = T + b_0 R' P_r \tag{5}$$

Here, a_0 and b_0 are components of a matrix abc that is computed by $A^{-1}T$, and A is given by, $[p_l - R'P_r$ $P_l \times R'P_r]$. P_l and P_r are the points in (2D) camera coordinates respectively for the left and right images.

This was followed by saving the final point P as a homogeneous co-ordinate by adding an extra dimension with 1's.

5.3 Calculating 3D world points

We used following equation for calculating

$$X_c = RTX_c \implies X_w = RT^{-1}X_c \tag{6}$$

The 3D points were then plotted in a 3D graph and the results are shown in figure 5.

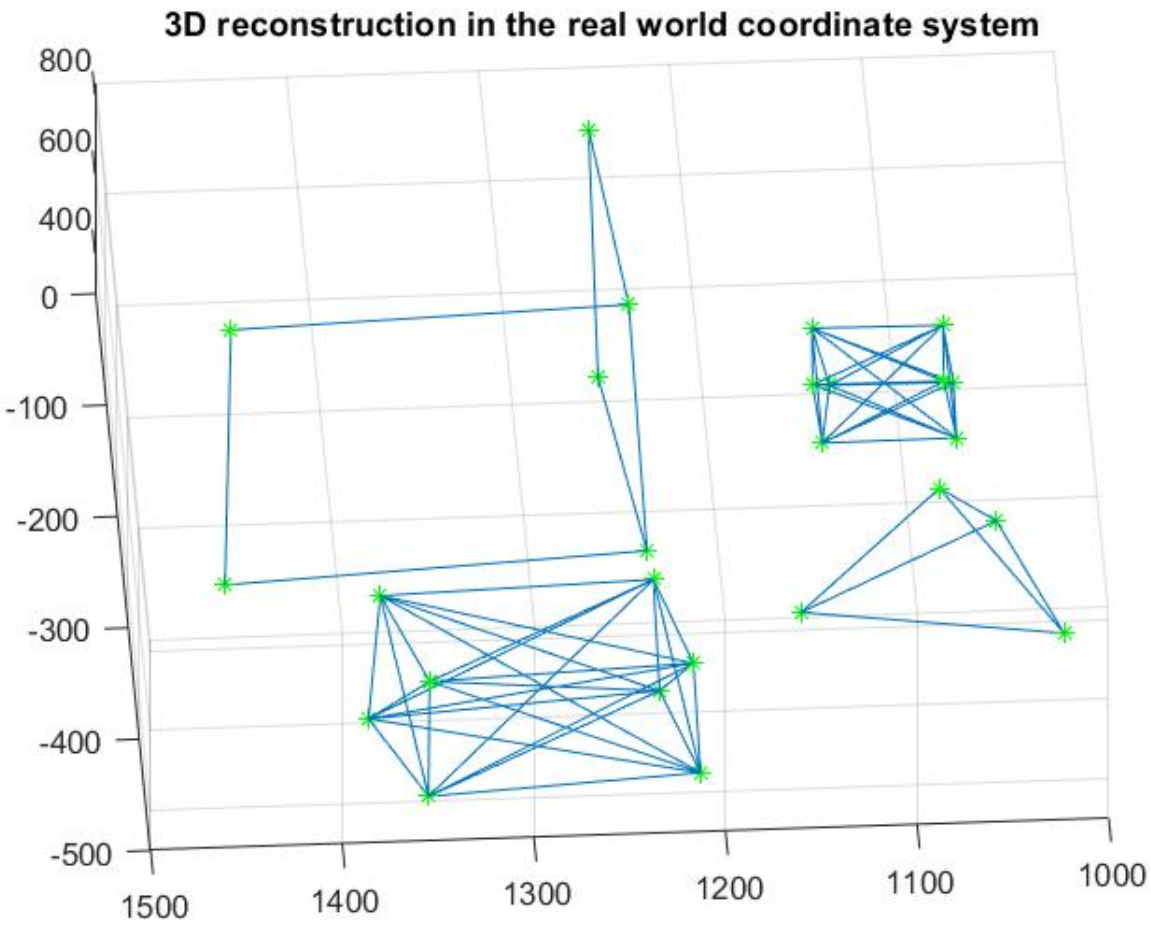

Figure 5: Caption

Note: The cube and rectangular box are presented with lines for diagonals as well. This was more intuitive. However, if needed the lines can easily be removed.

6 Conclusion

We were successful in our task of representing a 3D scene with the input of two images from different camera poses. Further, this lab exercise also completed the necessary tools for basics of Computer Vision - corner detection, calibrating camera, learning about epipolar geometry and 3D reconstruction.