Computer Vision
CSCI-UA.0480-002
Assignment 2.

March 5, 2013

Introduction

This assignment contains a number of problems in geometric computer vision. The course slides contains all the information needed to solve the four problems:

1. Epipolar geometry – using epipolar constraints in a two camera system. [*35 points*].

2. Estimating Camera Parameters – using a set of 3D world points and their 2D image locations, estimate the projection matrix P of a camera. [*35 points*].

3. Structure from Motion – infer the 3D structure of an object, given a set of images of the object. [*35 points*].

4. Camera Calibration from a Set of Images – a practical exercise that asks you to use a standard calibration toolbox to estimate the camera parameters. [*15 points*].

Requirements

You should perform this assignment in Matlab, apart from Question 1 which requires just pen and paper.

This assignment is due on **Tuesday April 2nd** by the start class (11am). The late policy is posted on the course webpage. You are strongly encouraged to start the assignment early and don't be afraid to ask for help.

The TA for the class is Pravish Sood (`pravish.sood@nyu.edu`). Please email him for help and assistance, or come to office hours (Thursday 12.30-1.30pm). If you think the assignment is not clear, or there is an error/bug in it, please contact the TA or myself.

You are allowed to collaborate with other students in terms discussing ideas and possible solutions. However you code up the solution yourself, i.e. you must write your own code. Copying your friends code and just changing all the names of the variables is not allowed! You are not allowed to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web.

Your solutions should be emailed to me (`fergus@cs.nyu.edu`) and the TA (`pravish.sood@nyu.edu`) as a single zip file, with the filename: `lastname_firstname_a2.zip`. This zip file should contain: (i) a PDF file `lastname_firstname_a2.pdf` with your report, showing output images for each part of the assignment and explanatory text, where appropriate; (ii) the source code used to generate the images (with code comments), along with a master script (named `master_a2.m`) that runs the code for each part of the assignment in turn.

1 Epipolar geometry

Figure 1 shows a pair of cameras each with a focal length of unity, whose principal axes meet at a point. The y-axes of both cameras are parallel and point out of the page. Assume that the left camera (center C_L) lies at the origin of the world coordinate system. Please answer the following questions about the figure in your report.

1. Write down the camera matrices for this configuration and verify that the fundamental matrix F is:

$$\begin{pmatrix} 0 & -d/2 & 0 \\ -d/2 & 0 & -\sqrt{3}d/2 \\ 0 & \sqrt{3}d/2 & 0 \end{pmatrix}$$

   Hint: Take care when constructing P', the projection matrix of the right camera. Do it by deriving X' in terms of X, via a sequence of stages: (i)

a translation of the coordinate frame from C to the intersection point; (ii) a rotation of the coordinate frame about the intersection point and (iii) another translation to C'.

2. Compute the epipolar line in the right image corresponding to the homogeneous point $\mathbf{x} = (1, 1, 1)^T$ in the left from $\mathbf{l} = \mathbf{F}\mathbf{x}$.

3. Using figure 1 determine where potential correspondences to the left image point $(x, y) = (0, 0)$ can lie in the right image.

4. Describe the rotation and translatation that should be applied to the left camera that would make the epipolar lines in the two images horizontal.

2 Estimating the camera parameters

Here the goal is the compute the 3x4 camera matrix P describing a pinhole camera given the coordinates of 10 world points and their corresponding image projections. Then you will decompose P into the intrinsic and extrinsic parameters. You should write a simple Matlab script that works through the stages below, printing out the important terms.

Download from the course webpage the two ASCII files, `world.txt` and `image.txt`. The first file contains the (X,Y,Z) values of 10 world points. The second file contains the (x,y) projections of those 10 points.

(a) Find the 3x4 matrix P that projects the world points $\mathbf{X}$ to the 10 image points $\mathbf{x}$. This should be done in the following steps:

- Since P is a homogeneous matrix, the world and image points (which are 3 and 2-D respectively), need to be converted into homogeneous points by concatenating a 1 to each of them (thus becoming 4 and 3-D respectively).

- We now note that $\mathbf{x} \times P\mathbf{X} = 0$, irrespective of the scale ambiguity. This allows us to setup a series of linear equations of the form:

$$\begin{bmatrix} 0^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & 0^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & 0^T \end{bmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \qquad (1)$$

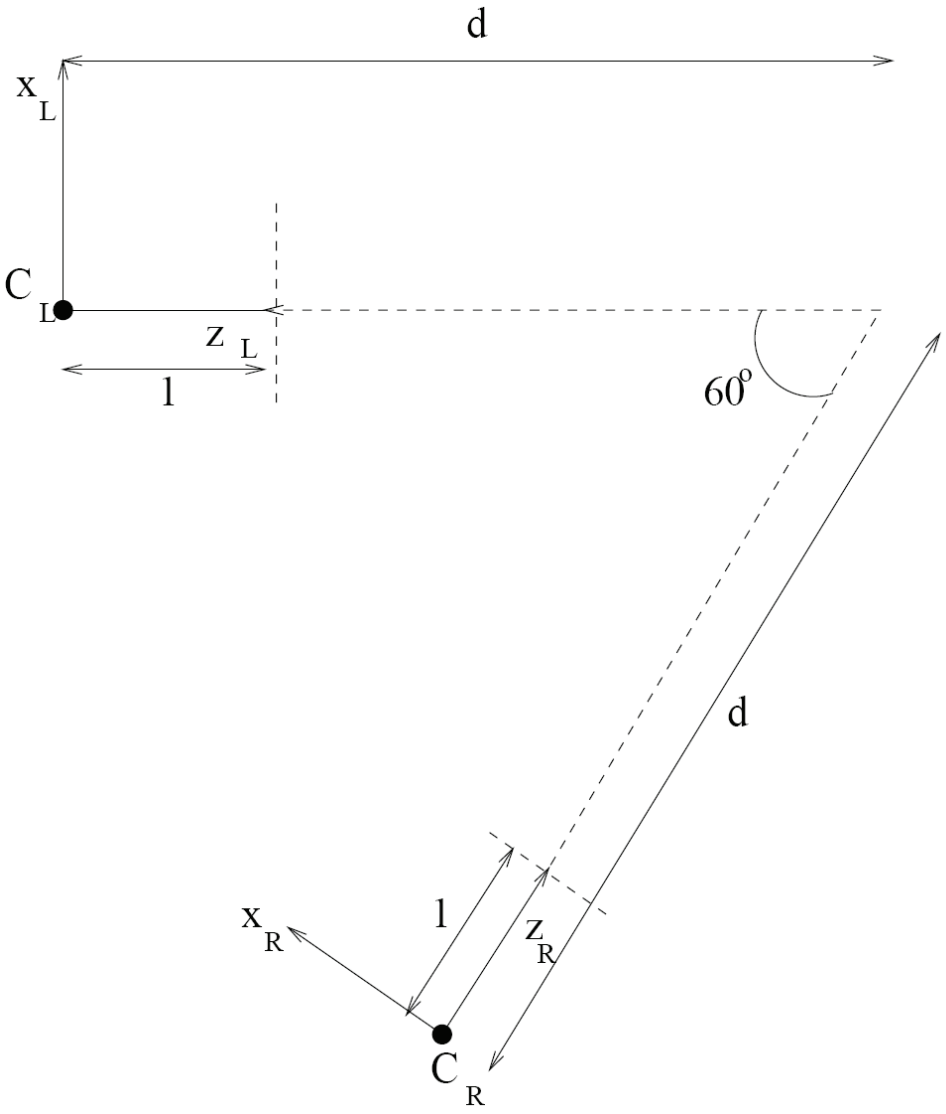

Figure 1: Figure for question 1

for each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$, where $\mathbf{x}_i = (x_i, y_i, w_i)^T$, w_i being the homogeneous coordinate, and P^j is the j^{th} row of P. But since the 3rd row is a linear combination of the first two, we need only consider the first two rows for each correspondence i. Thus, you should form a 20 by 12 matrix A, each of the 10 correspondences contributing two rows. This yields $Ap = 0$, p being the vector containing the entries of matrix P.

- To solve for p, we need to impose an extra constraint to avoid the trivial solution $p = 0$. One simple one is to use $||p||_2 = 1$. This constraint is implicitly imposed when we compute the SVD of A. The value of p that minimizes Ap subject to $||p||_2 = 1$ is given by the eigenvector corresponding to the smallest singular value of A. To find this, compute the SVD of A, picking this eigenvector and reshaping it into a 3 by 4 matrix P.

- Verify your answer by re-projecting the world points $\mathbf{X}$ and checking that they are close to $\mathbf{x}$.

(b) Now we have P, we can compute the world coordinates of the projection center of the camera C. Note that $PC = 0$, thus C lies in the null space of P, which can again be found with an SVD (the matlab command is `svd`). Compute the SVD of P and pick the vector corresponding to this null-space. Finally, convert it back to homogeneous coordinates and to yield the (X,Y,Z) coordinates. Your report should contain the matrix P and the value of C.

3 Structure from Motion

In this section you will code up an affine structure from motion algorithm, as described in the slides of lecture 6. For more details, you can consult page 437 of the Hartley & Zisserman book.

First download the file `sfm_points.mat` from the course webpage. This contains a 2 by 600 by 10 matrix, holding the x, y coordinates of 600 world points projected onto the image plane of the camera in 10 different locations. The points correspond, that is `image_points(:,1,:)` is the projection of the same 3D world point in the 10 frames. The points have been drawn randomly to lie on the surface of a transparent 3D cube, which does not move between

frames (i.e. the object is static, only the camera moves). Try plotting out several frames and the cube shaped structure should be apparent (the `plot3` command may be useful).

To simplify matters, we will only attempt an affine reconstruction, thus the projection matrix of each camera i will have following form:

$$P^i = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} M^i & t^i \\ 0 & 1 \end{pmatrix} \tag{2}$$

where M^i is a 2 by 3 matrix and t^i is a 2 by 1 translation vector.

So given $m = 10$ views and $n = 600$ points, having image locations $\mathbf{x}^i_j$, where $j = 1, \ldots, n$, $i = 1, \ldots, m$, we want to determine the affine camera matrices M^i, t^i and 3D points $\mathbf{X}_j$ so that we minimize the reconstruction error:

$$\sum_{ij} ||\mathbf{x}^i_j - (M^i \mathbf{X}_j + t^i)||^2 \tag{3}$$

We do this in the following stages:

- Compute the translations t^i directly by computing the centroid of point in each image i.
- Center the points in each image by subtracting off the centroid, so that the points have zero mean
- Construct the $2m$ by n measurement matrix W from the centered data.
- Perform an SVD decomposition of W into UDV^T.
- The camera locations M^i can be obtained from the first three columns of U multiplied by $D(1:3, 1:3)$, the first three singular values.
- The 3D world point locations are the first three columns of V.
- You can verify your answer by plotting the 3D world points out using the `plot3` command. The `rotate3d` command will let you rotate the plot.

You should write a script to implement the steps above. The script should print out the M^i and t^i for the first camera and also the 3D coordinates of the first 10 world points. Cut and paste these into your report.

4 Camera Calibration from a Set of Images

Practical methods for camera calibration use a combination of techniques that draw of the methods you have just used in the previous two questions. In Question 2, the 3D locations of the world points was specified, but this hard to do in practice. Therefore, a common solution is to capture multiple images of a calibration target and use Structure from Motion methods to work out the parameters of the camera, as well as the position of the calibration target in each frame.

This question asks you to download and run a standard toolbox that integrates these tasks, and is widely used in practice by many people who need to calibrate cameras. It is a straightforward exercise, but shows you how you would calibrate a camera in practice.

The methods inside the toolbox are very similar to those of Question 3, except that they handle projective cameras[1] (as opposed to the affine cameras from Qu. 3). We will be using the Camera Calibration Toolbox written by Jean-Yves Bouguet, which can be found at `http://www.vision.caltech.edu/bouguetj/calib_doc`.

- Download the toolbox from `http://www.vision.caltech.edu/bouguetj/calib_doc/download/toolbox_calib.zip`. Unzip it into some directory, for example, `/home/username/matlab/toolbox_calib`.
- In Matlab, add this directory to the set that Matlab can see with the command: `addpath('/home/username/matlab/toolbox_calib')`.
- Download the calibration images `part4.zip` from the course webpage and unzip them into your assignment directory.
- Start the calibration tool with the command: `calibgui`. A window should pop up. Select the `Standard` option. This should bring up an array of buttons that we will be working through. Documentation for using this tool can be found at: `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`.
- Click on the **Image Names** button and follow the instructions to load all 20 images into memory.

[1] In a projective camera, typically the 3rd row of P^i will have non-zero elements for the first three elements.

- Click on the **Extract grid corners** button and follow the instructions to load the images into memory. Use default settings. The grid squares at 27.5mm on each side. Follow the directions given in the webpage above, *carefully* clicking on the four corners of the checkerboard in the order top-left; top-right; bottom-right and bottom-left (i.e. short-side,long-side,short-side). In some of the images, the chart is rotated to landscape format. In these case, you should still go short-side,long-side,short-side, thus the click ordering may change.

- Click on the **Calibration** button to run the calibration code. If you have clicked carefully, the pixel error should be less than 1 pixel. Cut and paste the calibration parameters (after optimization) into your report. To show that you understand the role of each of the parameters, write down the 3x3 instrinsic parameter matrix using the values produced. Ignore the skew and distortion values.

- Click on the **Show Extrinsic** button to get a visualization of the position of the chart relative to the camera for each frame.