EECS 442 Computer Vision, Fall 2014 Homework 3

Due on Thursday, October 23, 2014 at 11:55pm.
Please submit your assignment on ctools

Instructions for your write-up: In your solutions to the problems, discuss your approach in words (and equations where necessary). Attach all affiliated code as an appendix to your solutions. You will be graded on the correctness, clarity and brevity of your solution.

1 [20 points] Some Projective Geometry Problems

Suppose there are two parallel lines that extend to infinity in our world coordinates. The images of these two lines intersect at a point, v, on the image plane known as the vanishing point. In this problem we choose to represent the lines as

$$l = \left\{ r \begin{bmatrix} \cos\alpha \\ \cos\beta \\ \cos\gamma \end{bmatrix} \;\middle|\; r \in \mathsf{R} \right\}$$

where α, β, γ are the angles from the x, y and z world axes respectively.

(a) [**8 points**] Using homogeneous coordinates to show that the image coordinates of the vanishing point v can be written as $v = KRd$, where

$$d = \begin{bmatrix} \cos\alpha \\ \cos\beta \\ \cos\gamma \end{bmatrix}$$

K is the camera calibration matrix and R is the rotation matrix between the camera and the world coordinates.

(b) [**6 points**] Express the line direction d in terms of K, R and v.
Remark: Any matrix inverses which do not hold in general, must be shown to hold in this case.

(c) [**6 points**] Now consider the situation where three lines intersect such that each line is orthogonal to the other two. That is, each line's directional vector d satisfies the following.

$$d_1^T d_2 = 0 \quad d_1^T d_3 = 0 \quad d_2^T d_3 = 0$$

Using this property, show that

$$(K^{-1}v_i)^T(K^{-1}v_j) = 0$$

for $i \neq j$.

(a) Image 1 (1.jpg)

(b) Image 2 (2.jpg)

Figure 1

2 [40 points] Single View Geometry

(a) [*Camera Calibration*] Identify a set of N vanishing points from the image 1. Assume the camera has zero skew and square pixels, with no distortion.

(1) [**8 points**] Compute the focal length and the camera center using these N vanishing points. Show derivation and report numerical results.
Note: Use the function 'getVanishingPoint.m' to compute the vanishing point. Read the ReadMe file for usage information.

(2) [**2 points**] Is it possible to compute the focal length for any choice of vanishing points?

(3) [**2 points**] What's the minimum N for solving this problem? Justify answer.

(4) [**2 points**] The method used to obtain vanishing points (as explained in the ReadMe) is approximate and prone to noise. Discuss approaches to refine this process.

This process gives the camera internal matrix under the specified constraints. For the remainder of the computations, use the following internal camera matrix:

$$K = \begin{bmatrix} 2448 & 0 & 1253 \\ 0 & 2438 & 986 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) [**8 points**] [*Orthogonality*] Identify a sufficient set of vanishing lines on the ground plane and the plane on which the letter A exists (plane-B). Use these vanishing lines to verify numerically that the ground plane is orthogonal to the plane-B.

(c) [**9 points**] [*Rotation Estimation*] Assume the camera rotates but no translation takes place. Assume the internal camera parameters remain unchanged. An image 2 of the same scene is taken. Use vanishing points to compute camera rotation matrix. Report derivation and numerical results.

(d) [**9 points**] [*Homography*] Compute the 3D homographic transformation H mapping the letter 'A' in image 1 to the same letter "A" in image 2 (without using point correspondences). Show H can be expressed as function of R. Report derivation and numerical results.

3 [40 points] Affine Camera Calibration

It was shown in the class that a perspective camera can be modeled using a 3×4 matrix:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

which means that the image at point (X, Y, Z) in the scene has pixel coordinates $(x/w, y/w)$. The 3×4 matrix can be factorized into intrinsic and extrinsic parameters.

An *affine* camera is a special case of this model in which rays joining a point in the scene to its projection on the image plane are parallel. Examples of affine cameras include orthographic projection and weakly perspective projection. An affine camera can be modeled as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

which gives the relation between a scene point (X, Y, Z) and its image (x, y). The difference is that the bottom row of the matrix is $[\ 0 \quad 0 \quad 0 \quad 1\ ]$, so there are fewer parameters we need to calibrate. More importantly, there is no division required (the homogeneous coordinate is 1) which means this is a *linear mode*. This makes the affine model much simpler to work with mathematically at the cost of losing some accuracy. The affine model is used as an approximation of the perspective model when the loss of accuracy can be tolerated, or to reduce the number of parameters being modeled.

Calibration of an affine camera involves estimating the 8 unknown entries of the matrix in Eq. 2. (This matrix can also be factorized into intrinsics and extrinsics, but that is outside the scope of this homework). Factorization is accomplished by having the camera observe a calibration pattern with easy-to-detect corners.

Scene Coordinate System

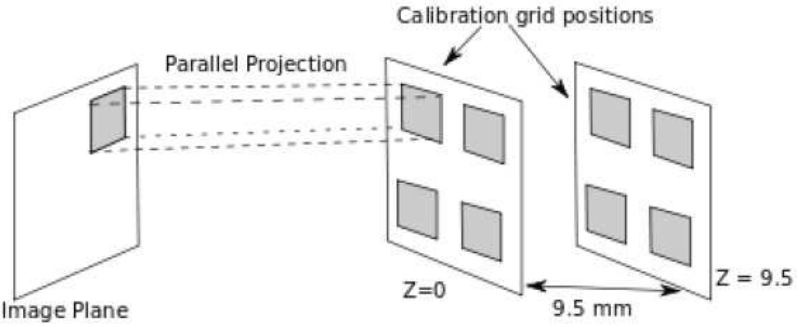
The calibration pattern used is shown in Fig. 2, which is a 6×6 grid of squares. Each square is $17.5mm \times 17.5mm$. The separation between adjacent squares too is $17.5mm$, so the entire grid is $192.5mm \times 192.5mm$. For calibration, images of the pattern at 2 different positions were captured. These images are shown in Fig. 2 and are also present in the assignment folder. For the second image, the calibration pattern has been moved back (along its normal) from the rest position by $9.5mm$.

We will choose the origin of our 3D coordinate system to be the top left corner of the calibration pattern in the rest position. The X-axis runs left to right parallel to the rows of squares. The Y-axis runs top to bottom parallel to the column of squares. We will work in units of millimeters. All the square corners from the first position corresponds to $Z = 0$. The second position of the calibration grid corresponds to $Z = 9.5$. The top left corner in the first image has 3D scene coordinates $(0, 0, 0)$ and the bottom right corner in the second image has 3D scene coordinates $(192.5, 192.5, 9.5)$. This scene coordinate system is labeled in Fig. 2.

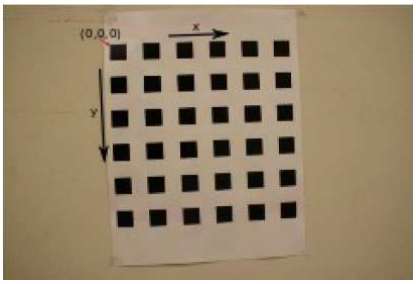
Corner Extraction

The task that you must complete for this question begins here. Extract the feature correspondences required for calibration. In other words, first find the scene coordinates of the square corners of the calibration grid. Next, find the corresponding pixel coordinates in the images.

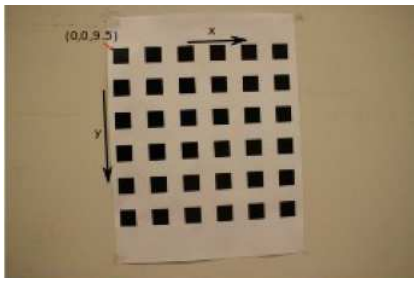
If you wish to measure the pixel coordinates interactively in MATLAB, you may find the function `ginput` useful. It is strongly recommended that you save this matrix of measurements in a files for later use. You do not need to find *all* the corners, but the more corners you use the more accurate the calibration is likely to be. Be sure to include corners for *both* images. We recommend using at least 12 corners from each image.



(a) Image formation in an affine camera. Points are projected via parallel rays onto the image plane



(b) Image of calibration grid at $Z = 0$



(c) Image of calibration grid at $Z = 9.5$

Figure 2: Affine camera: image formation and calibration images

Solving for the camera matrix

Having measured enough features, we can now solve for the camera parameters using Eq. 2. Note that each measurements $(x_i, y_i) \leftrightarrow (X_i, Y_i, Z_i)$ yields two linear equations for the 8 unknown camera parameters. Given N corner measurements, we have $2N$ equations and 8 unknowns. Using your measured feature correspondences as input, write code which constructs the linear system of equations and solves for the camera parameters which minimizes the least-squares error.

(a) [**6 points**] Could you calibrate the matrix with only one checkerboard image? Explain Why.

(b) [**17 points**] The RMS error (in pixels) between the corner positions and the positions predicted by the camera parameters.

(c) [**17 points**] Give your 3×4 calibrated camera matrix.