TSBB09 Computer Exercise: Camera Calibration

Developed by Maria Magnusson.

Computer Vision Laboratory, Linköping University, Sweden

Last update: November 11, 2020

Contents

1 Preliminaries

Before attending the computer exercise it is necessary to read this guide to the computer exercise. It is also valuable to attend the two lectures on Camera Calibration and to take a look in the course material, [1] and [2]. The guide contains some home exercises. Try to answer them before the session. They are all clearly marked with a pointing finger.

2 Exercises

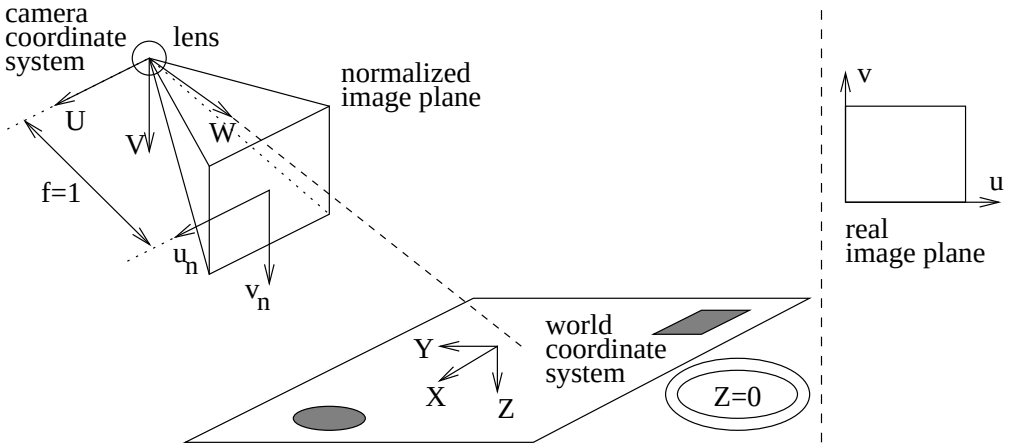
2.1 Images and other files. Camera set-ups.

Open a terminal and write:
`cd /courses/TSBB09`
This way, TSBB09 will be visible for you.
In a window (not a terminal) navigate to `/courses/TSBB09/CameraCalibration1`.
Alternatively you can download all the files you need from Lisam.

The images `im0.png`, `im1.png`, `im2.png`, `im3.png`, `imz0.png`, `imz1.png`, `imz2.png` and `imz3.png`. are taken with the steerable camera cvl-cam-00.
The MATLAB images `objectImage.fig` and `calibrImage.fig` are taken with the non-steerable camera cvl-cam-01.

The camera set-ups are demonstrated in the videos `cvl-cam-00.mp4` and `cvl-cam-01.mp4`, see Lisam.

2.2 Calibration of a flat world, a homography



This lab exercise is about how make a simple calibration of a camera, or more specifically, how to determine a homography. Then we will determine the length of an object through its image. See the figure above. We want to determine the relationship between image coordinates $(u, v)^T$ and flat world coordinates $(X, Y, Z = 0)^T$. The connection between the coordinate systems is

$$(su, sv, s)^T = s(u, v, 1)^T = \mathbf{C} \cdot (X, Y, 1)^T,$$

where s is a scale factor. The matrix $\mathbf{C}$ consequently contains information about

- the transformation from world coordinates to camera coordinates
- the transformation from the normalized image plane to the real image plane.

The matrix $\mathbf{C}$ looks as follows,

$$\mathbf{C} = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & 1 \end{pmatrix}.$$

Let $\mathbf{c} = (C_{11}, C_{12}, C_{13}, C_{21}, C_{22}, C_{23}, C_{31}, C_{32})$.

By using a number of measured corresponding points in the world $((X_1, Y_1), ..., (X_N, Y_N))$ and the image $((u_1, v_1), ..., (u_n, v_N))$, the following equation system is obtained,

$$\mathbf{D} \cdot \mathbf{c} =$$

$$\begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -u_2X_2 & -u_2Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & X_N & Y_N & 1 & -v_NX_N & -v_NY_N \end{pmatrix} \cdot \begin{pmatrix} C_{11} \\ C_{12} \\ C_{13} \\ \vdots \\ C_{32} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ \vdots \\ v_N \end{pmatrix} = \mathbf{f}.$$

Home exercise A: How many points in the world (X_i, Y_i) are, at least, needed to determine $\mathbf{C}$?

Home exercise B: The camera should not be in wide angle mode. Why?

Start MATLAB and give the command:
`figure(1); figure(2)`
so that two figure windows are opened. The previously mentioned images `calibrImage.fig` and `objectImage.fig` contain the calibration pattern and the object you want to measure. They can be loaded into the figure windows.

The calibration points have the following world coordinates:

```
X1 =  0;      Y1 =  0;      X2 =  5;      Y2 =  0;
X3 = 10;      Y3 =  0;      X4 =  0;      Y4 =  5;
X5 =  5;      Y5 =  5;      X6 = 10;      Y6 =  5;
X7 =  0;      Y7 = 10;      X8 =  5;      Y8 = 10;
X9 = 10;      Y9 = 10;
```

QUESTION 1: Which units are used for the world coordinates, inches, mm or cm? Listen to the cvl-cam-01 video!

QUESTION 2: Measure the positions of the calibration points in the image. Try to get sufficiently accurate values by zooming (with the magnifying glass) in the image.

```
u1 = ________; v1 = ________; u2 = ________; v2 = ________;

u3 = ________; v3 = ________; u4 = ________; v4 = ________;

u4 = ________; v5 = ________; u6 = ________; v6 = ________;

u7 = ________; v7 = ________; u8 = ________; v8 = ________;

u9 = ________; v9 = ________;
```

Home exercise C: You should soon calculate $\mathbf{c}$ through your calibration points and the pseudoinverse. The pseudoinverse of a matrix $\mathbf{A}$ is written $\mathbf{A}^+$. Write an equation which shows how $\mathbf{c}$ can be calculated from $\mathbf{D}$ and $\mathbf{f}$.

Now calculate the vector $\mathbf{c}$ in MATLAB. The MATLAB command `pinv` corresponds to the pseudoinverse. For your help there is a non-completed MATLAB code in `calibr.m`.
See also the MATLAB section in the end this laboratory assignment.

QUESTION 3: Give the vector **c** below!

The matrix **C** can then be received from the vector **c** using these MATLAB commands:

```
c = [c; 1];
C = (reshape (c, 3, 3))';
```

You can now perform a test to check if your **C**-matrix seems to be correct. Write the following MATLAB commands:

```
test = C * [5 5 1]'
u5new = test(1) / test(3)
v5new = test(2) / test(3)
```

QUESTION 4: Give the values $(u5_{new}, v5_{new})$ and $(u5, v5)$.

QUESTION 5: Compare the values $(u5_{new}, v5_{new})$ and $(u5, v5)$. They should conform fairly well. What are the reasons if they do not match perfectly?

QUESTION 6: Listen to the cvl-cam-01 video. What is the length of the real potato stick?

QUESTION 7: In the image, measure the two endpoints (u_a, v_a) and (u_b, v_b) of the potato stick:

```
ua = ________; va = ________; ub = ________; vb = ________;
```

Since previously we know that this is valid:

$$s \cdot (u, v, 1)^T = \mathbf{C} \cdot (X, Y, 1)^T.$$

Consequently:

$$(1/s) \cdot (X, Y, 1)^T = \mathbf{C}^{-1} \cdot (u, v, 1)^T$$
$$(X/s, Y/s, 1/s)^T = \mathbf{C}^{-1} \cdot (u, v, 1)^T$$

Determine

```
Xa/sa = ________; Ya/sa = ________; 1/sa = ________;

Xb/sb = ________; Yb/sb = ________; 1/sb = ________;
```

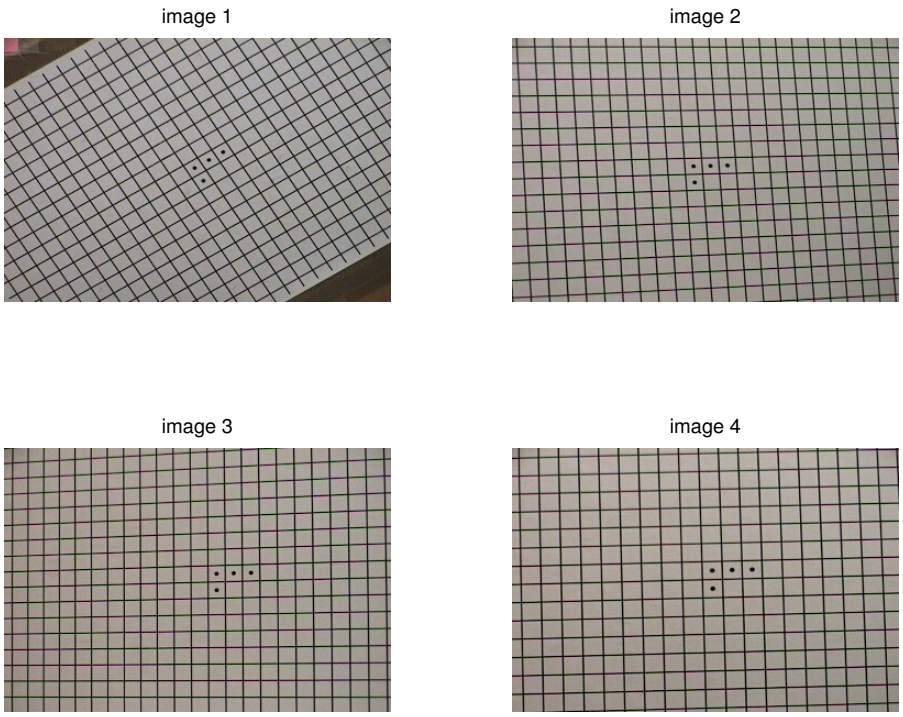
QUESTION 8: Determine the end points in world coordinates.

```
Xa = ________; Ya = ________; Xb = ________; Yb = ________;
```

QUESTION 9: Which length does this corresponds to? Is this length similar to to real length?

QUESTION 10: Mention some reasons why the real object length does not perfectly match the calculated length.

2.3 Complete camera calibration



Here we will perform camera calibration according to Zhang [2]. See the figure above. The images are `im0.png`, `im1.png`, `im2.png` and `im3.png`. The first image, top left, was used as the world coordinate system when **R** and **t** were determined. The X-axis lies along the line above the 3 points. The Y-axis lies along the line to the left of the two points. The Z-axis is consequently orthogonal to the plane. The MATLAB program `run_norm.m` calls `CameraCalibration.m`, which calls `generate_homog.m` and `homography2intrinsic.m`. Those rather small MATLAB files perform the full calibration giving corresponding points in the images and the world. The image points where measused in pixel units and the world points where measured in mm. Refinement of all parameters, including lens distortion parameters in a non-linear minimization algorithm is **not** included in those MATLAB files, however. First, the homographies for the four images are estimated. Thereafter, the **A**-matrix, **R**-matrix and **t**-vector are calculated.

Home exercise D: Look in `generate_homog.m`, is the homographies solved with the homogeneous or inhomogenious solution?

Home exercise E: Look in `homography2intrnsic.m` and locate
`[U,S,V1] = svd(V); b = V1(:,6);`
b is related to **B**. How is **A** related to **B**? Check [2] or the lecture slides and give one equation!

Run the program `run_norm`. It starts by reading the images `im0.png`, `im1.png`, `im2.png` and `im3.png`. It also read a set of corresponding point-pairs that have been detected in the images.
It ends with calling `CameraCalibration.m`. Check that it computes the same **A**[**Rt**] as below!

```
A =
  1.0e+003 *
    1.2872   -0.0020    0.1898
         0    1.1672    0.1340
         0         0    0.0010
R =
    0.8498    0.5255   -0.0147
   -0.5269    0.8505   -0.0034
    0.0107    0.0106    0.9997
t =
  -17.4148
  -12.8092
  853.6756
```

QUESTION 11: What do you think that the points marked with green means? What do you think that the lines marked with red means?

QUESTION 12: What do you think that the points marked with blue means? What do you think that the point marked with magenta means?

Home exercise F: Calculate the distance between the camera center and the world coordinate system origin. Use **A**, **R** and/or **t**.

QUESTION 13: Listen to the cvl-cam-00 video. What is the distance between the camera center (for simplicity, assume the rotation center) and the origin of the world coordinate system. Did you get consistency between calculation and measurement?

QUESTION 14: Listen to the cvl-cam-00 video. Where is the physical camera center located?

Home exercise G: Rotation between camera and world coordinate system has been performed almost exclusively around one axis. Which axis and how much rotation? Use $\mathbf{A}$, $\mathbf{R}$ and/or $\mathbf{t}$.

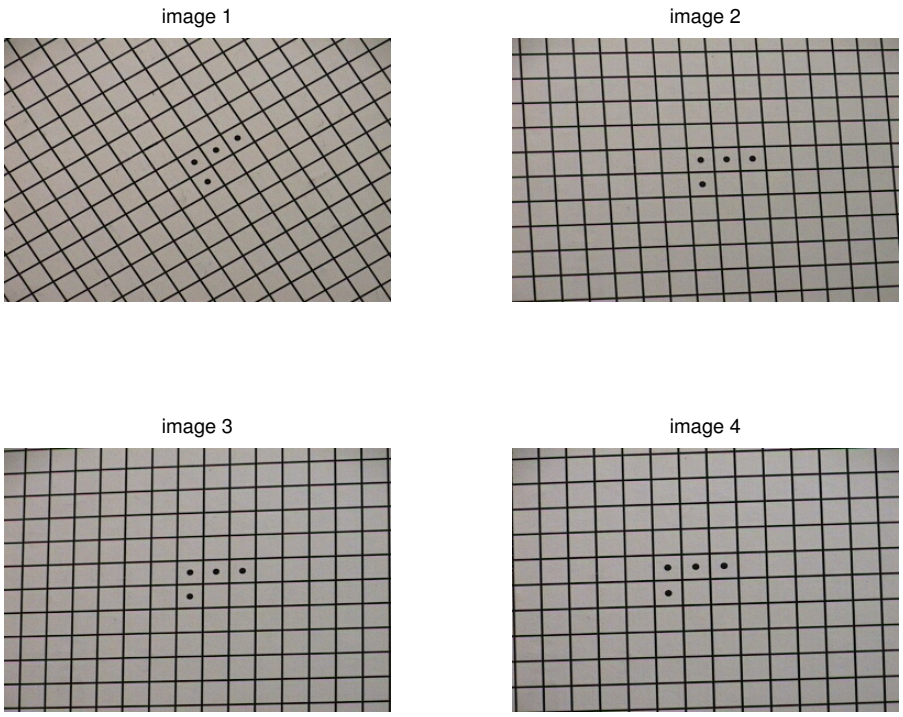
QUESTION 15: Listen to the cvl-cam-00 video. What is the rotation between the camera and the world coordinate system? Did you get consistency between calculation and measurement?

Home exercise H: What is the relationship between the pixel distances in the x- and y-direction. Use $\mathbf{A}$, $\mathbf{R}$ and/or $\mathbf{t}$.

Home exercise I: The image size is 352×240, giving the center at $(176.5, 120.5)$. Where is the correct center located, i.e. where does the optical axis intersect the image plane? Use $\mathbf{A}$, $\mathbf{R}$ and/or $\mathbf{t}$.

Home exercise J: How big is the skew angle ($\xi = \arctan(\gamma/\beta)$) between the image pixels? Give a measurement in degrees. Use $\mathbf{A}$, $\mathbf{R}$ and/or $\mathbf{t}$.

2.4 Complete camera calibration with zoomed lens



Here we will perform the camera calibration according to Zhang [2] again, but now with a zoomed lens. The four images above are the ones named `imz0.png`, `imz1.png`, `imz2.png` and `imz3.png`. Run the program `run_zoom`. It starts by reading the images `imz0.png`, `imz1.png`, `imz2.png` and `imz3.png`, and ends with calling `CameraCalibration.m`. Check that it computes the same `A[Rt]` as below!

```
A =
1.0e+003 *
2.0144 -0.0041 0.2253
0.0000 +1.8249 0.1310
0.0000 +0.0000 0.0010
R =
+0.8493 0.5261 -0.0225
-0.5275 0.8508 +0.0011
+0.0196 0.0109 +1.0001
t =
-29.4055
-10.0248
879.2940
```

QUESTION 16: We will now estimate how much larger the pixels are now compared to before. It is recommended to count the number of squares along the diagonals of im0.png and imz0.png to recive the relation.

QUESTION 17: How much have the pixel size increased according to the matrices and is it in agreement with the above mentioned measurement?

QUESTION 18: Calculate the distance between the camera center and the world coordinate system origin. What can be the reason why the distance differ sligthy compared to before?

QUESTION 19: How big is the rotation according to the matrices now? It is approximately the same as before?

QUESTION 20: What is the relationship between the pixel distances in the x- and y-direction? Is the measure approximately the same as before?

QUESTION 21: The image size is 352×240, giving the center at $(176.5, 120.5)$. Where is the correct image center located, i.e. where does the optical axis intersect the image plane? Compare with previous measurements. If it is not the same as before, try to give an explanation.

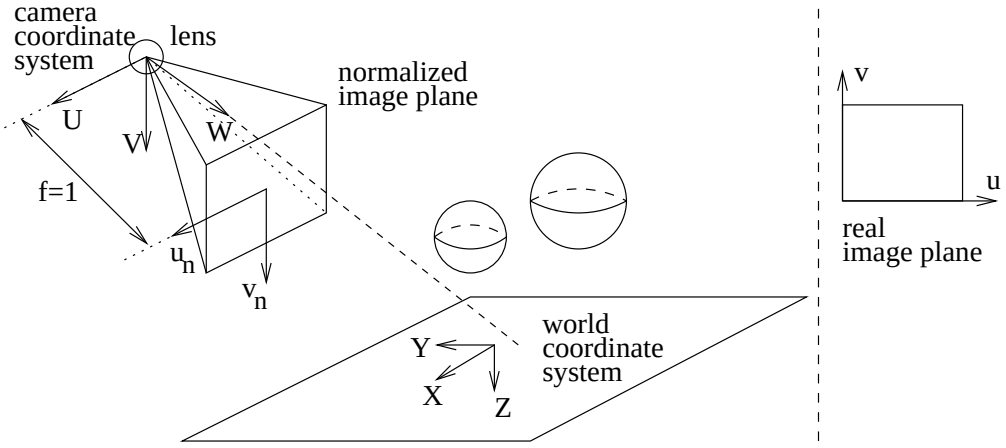
2.5 To follow an object with the camera

Home exercise K: Study the following problem with answer.

The figure shows a 3D world, a camera, a normalized image plane (with focal length $f = 1$) and a real image plane in the camera. These are the connections between the coordinate systems:

$$s(u, v, 1)^T = A[R\ t] \cdot (X, Y, Z, 1)^T, \quad (u, v, 1)^T = A \cdot (u_n, v_n, 1)^T.$$

Consequently, the matrix $A[R\ t]$ denotes the transformation from the world coordinate system to the real image coordinate system.



The camera's task is to follow an object. It can rotate in two angular directions θ_u and θ_v (horizontally and vertically). The image center is located at the coordinate $(u, v) = (250, 200)$. Suppose that we have located the object at the coordinate $(u, v) = (250 + 225, 200 + 175) = (475, 375)$ in the image. How large angles, θ_u and θ_v, should the camera rotate to bring the object to the center of the image? Assume that the A-matrix was

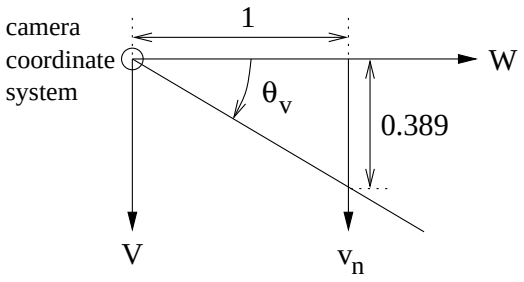
$$A = \begin{pmatrix} 500 & 0 & 250 \\ 0 & 450 & 200 \\ 0 & 0 & 1 \end{pmatrix}.$$

Answer

It seems that the normalized image coordinate $(u_n, v_n) = (0.450, 0.389)$ is transformered to the real image coordinate $(u, v) = (475, 375)$ according to

$$\begin{pmatrix} 475 \\ 375 \\ 1 \end{pmatrix} = \begin{pmatrix} 500 & 0 & 250 \\ 0 & 450 & 200 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0.450 \\ 0.389 \\ 1 \end{pmatrix}.$$

Therefore $\theta_u = \arctan(0.450/1) = 0.423 = 24°$ and $\theta_v = \arctan(0.389/1) = 0.371 = 21°$, see figure.



Home exercise L: See the four calibration images with their **A**-matrix in section 2.3 "Complete camera calibration". Suppose that we have located an object at the coordinate $(u, v) = (0, 0)$ in the image. How large angles, θ_u and θ_v, should the camera rotate to bring the object to the optical center of the image?

Load the color image `basic.jpg`. This is an image with the camera in a basic position. Note that a small magenta patch is put on the calibration pattern near the coordinate $(0, 0)$. (Actually, rather the coordinate $(1, 1)$ is located in the upper left corner.) If rotation according to the previous home exercise is performed with the camera, i.e. it is panned with θ_u and tilted with θ_v, the resulting image is `rot2corner.jpg`.

QUESTION 22: In the image, measure the position of the optical center and compare it with the calibrated optical center. Do they agree?

2.6 Camera resectioning

From Zhang's method we got the **A**- and **R**-matrices as well as the **t**-vector separately. Suppose that we have used a more common calibration procedure, approximately as in section 2.2, but in 3D instead of 2D, and with a 3D calibration object instead of a 2D calibration object. Then we would have received one single **C**-matrix. The method to extract $\mathbf{A}, \mathbf{R}$ and $\mathbf{t}$ from **C** is called *camera resectioning*. Code for this is in `P2KRt.m`.

QUESTION 23: Take one set of $\mathbf{A}, \mathbf{R}, \mathbf{t}$-matrices from section 2.3 "Complete camera calibration". Multiply them together as $\mathbf{C} = \mathbf{A}[\mathbf{R}\mathbf{t}]$. Now send **C** to `P2KRt.m`. What is the result?

QUESTION 24: This seems to be simpler than Zhang's method. What is the advantage with Zhang's method?

References

[1] M. Magnusson. Short on camera geometry and camera calibration. Technical report, ISY, 2010.

[2] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 2000.

3 MATLAB files

3.1 calibr.m

```
% Calibration points in the world
%--------------------------------
X1 =  0; Y1 =  0;
X2 =  5; Y2 =  0;
X3 = 10; Y3 =  0;
X4 =  0; Y4 =  5;
X5 =  5; Y5 =  5;
X6 = 10; Y6 =  5;
X7 =  0; Y7 = 10;
X8 =  5; Y8 = 10;
X9 = 10; Y9 = 10;

% Calibration points in the image
```

```
%--------------------------------
u1 = ; v1 = ;
u2 = ; v2 = ;
u3 = ; v3 = ;
u4 = ; v4 = ;
u5 = ; v5 = ;
u6 = ; v6 = ;
u7 = ; v7 = ;
u8 = ; v8 = ;
u9 = ; v9 = ;

f = [u1 v1 u2 v2 u3 v3 u4 v4 u5 v5 u6 v6 u7 v7 u8 v8 u9 v9]';

% Calibration matrix
%-------------------
D = [
    X1 Y1  1   0  0  0 -u1*X1 -u1*Y1;
     0  0  0  X1 Y1  1 -v1*X1 -v1*Y1;
    X2 Y2  1   0  0  0 -u2*X2 -u2*Y2;
     0  0  0  X2 Y2  1 -v2*X2 -v2*Y2;
    X3 Y3  1   0  0  0 -u3*X3 -u3*Y3;
     0  0  0  X3 Y3  1 -v3*X3 -v3*Y3;
    X4 Y4  1   0  0  0 -u4*X4 -u4*Y4;
     0  0  0  X4 Y4  1 -v4*X4 -v4*Y4;
    X5 Y5  1   0  0  0 -u5*X5 -u5*Y5;
     0  0  0  X5 Y5  1 -v5*X5 -v5*Y5;
    X6 Y6  1   0  0  0 -u6*X6 -u6*Y6;
     0  0  0  X6 Y6  1 -v6*X6 -v6*Y6;
    X7 Y7  1   0  0  0 -u7*X7 -u7*Y7;
     0  0  0  X7 Y7  1 -v7*X7 -v7*Y7;
    X8 Y8  1   0  0  0 -u8*X8 -u8*Y8;
     0  0  0  X8 Y8  1 -v8*X8 -v8*Y8;
    X9 Y9  1   0  0  0 -u9*X9 -u9*Y9;
     0  0  0  X9 Y9  1 -v9*X9 -v9*Y9];
```

3.2 MATLAB code for Zhang's camera calibration

3.2.1 CameraCalibration.m

```
function  [A, R, t] = CameraCalibration(Xplane, XImg, planeNo)
% Camera calibration. Returns intrinsic matrix A for the camera and
% extrinsic parameters [R,t]. Four sets of corresponding points
% from four different calibration images are used.
% XImg: Coordinates of the points in the image.
% XPlane: Coordinates of the points in the world.
% planeNo: Number of the plane that defines the coordinate system
%
% The extrinsic parameters are given in relation to the first image.

% read number of planes
%----------------------
siz = size(Xplane);
noPlanes = siz(2)

% load corresponding points in all 4 images and estimate homographies
%---------------------------------------------------------------------
for i = 1:noPlanes
  uv = XImg{i};  ui = uv(:,1);  vi = uv(:,2);
```

```
  XY = Xplane{i};  Xi = XY(:,1);  Yi = XY(:,2);
  c = generate_homog(ui,vi,Xi,Yi);
  Cbig(:,:,i) = [c(1:3)'; c(4:6)'; c(7:8)',1];
end

% Compute intrinsic parameters.
%------------------------------
Hbig = Cbig;
A = homography2intrinsic(Hbig);

% Compute extrinsic parameters. The extrinsic parameters
% are given in relation to planeNo.
%-------------------------------------------------------
H = Hbig(:,:,planeNo);
h1 = H(:,1);
h2 = H(:,2);
h3 = H(:,3);
invA = inv(A);
lambda=1/norm(invA*h1);

r1=lambda*invA*h1;
r2=lambda*invA*h2;
r3=cross(r1,r2);
t=lambda*invA*h3;
R=[r1,r2,r3];
```

3.2.2 generate_homog.m

```
function c = generate_homog(ui,vi,Xi,Yi)
% function c = generate_homog(ui,vi,Xi,Yi)
% Generates a homography, i.e. determines the matrix
% relating a plane and its image.
% (ui,vi): Coordinates of the points in the image.
% (Xi,Yi): Coordinates of the points in the world.

n = length(ui);

% Set up the calibration matrix
%------------------------------
D=[Xi(1),Yi(1),1,     0,    0,0,-ui(1)*Xi(1),-ui(1)*Yi(1);
       0,    0,0, Xi(1),Yi(1),1,-vi(1)*Xi(1),-vi(1)*Yi(1)];

for i=2:n
  D=[D;
     Xi(i),Yi(i),1,0,    0,    0,-ui(i)*Xi(i),-ui(i)*Yi(i);
         0,    0,0,Xi(i),Yi(i),1,-vi(i)*Xi(i),-vi(i)*Yi(i)];
end

f=[ui(1);
   vi(1)];
for i=2:n
  f=[f;
     ui(i);
     vi(i)];
end

c = pinv(D)*f;
c = [c;
     1];
```

3.2.3 homography2intrinsic.m

```
function A=homography2intrinsic(Hbig)
% The intrinsic matrix A is calculated from n homographies.
% Hbig is a 3x3xn matrix of homographies. This file is used by
% calib_zhang_simple, and is based on Zhang's calibration
% technique (see calib_zhang_simple.m)
%
% Assumes the homogeneous coordinate is at the end (i.e. [x y 1])

% Compute constraints for each homography
%-----------------------------------------
  for n=1:size(Hbig,3)
    H=Hbig(:,:,n)';
    v11=[H(1,1)*H(1,1), H(1,1)*H(1,2)+H(1,2)*H(1,1), H(1,2)*H(1,2), ...
         H(1,3)*H(1,1)+H(1,1)*H(1,3), H(1,3)*H(1,2)+H(1,2)*H(1,3), ...
         H(1,3)*H(1,3)]';

    v12=[H(1,1)*H(2,1), H(1,1)*H(2,2)+H(1,2)*H(2,1), H(1,2)*H(2,2), ...
         H(1,3)*H(2,1)+H(1,1)*H(2,3), H(1,3)*H(2,2)+H(1,2)*H(2,3), ...
         H(1,3)*H(2,3)]';

    v22=[H(2,1)*H(2,1), H(2,1)*H(2,2)+H(2,2)*H(2,1), H(2,2)*H(2,2), ...
         H(2,3)*H(2,1)+H(2,1)*H(2,3), H(2,3)*H(2,2)+H(2,2)*H(2,3), ...
         H(2,3)*H(2,3)]';

    V(n*2-1,:) = v12';
    V(n*2,:)   = (v11-v22)';
  end

  % Solve Vb=0
  %-----------
  [U,S,V1] = svd(V);
  b = V1(:,6);

  % Arrange b to form B
  %--------------------
  B=[b(1),b(2),b(4);b(2),b(3),b(5);b(4),b(5),b(6)];

  % Extract the intrinsic parameters from B
  %----------------------------------------
  v0=(B(1,2)*B(1,3)-B(1,1)*B(2,3))/(B(1,1)*B(2,2)-B(1,2)*B(1,2));
  lambda=B(3,3)-(B(1,3)*B(1,3)+v0*(B(1,2)*B(1,3)-B(1,1)*B(2,3)))/B(1,1);
  alpha=sqrt(lambda/B(1,1));
  beta=sqrt(lambda*B(1,1)/(B(1,1)*B(2,2)-B(1,2)*B(1,2)));
  gamma=-B(1,2)*alpha*alpha*beta/lambda;
  u0=(gamma*v0/alpha)-(B(1,3)*alpha*alpha/lambda);

  % arrange the extracted data to form A
  %-------------------------------------
  A=[alpha,gamma,u0;
     0,    beta, v0;
     0,    0,    1];
```