TSBB09 Computer Exercise C
Camera Calibration

Developed by Maria Magnusson.

Computer Vision Laboratory, Linköping University, Sweden

Last update: November 5, 2015

Contents

1 Preliminaries

Before attending the computer exercise it is nescessary to read this guide to the computer exercise. It is also valuable to read through the course material in [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 Tasks

2.1 The network cameras

Three network cameras are connected to the network in the computer exercise room. They are manufactured by AXIS and they contain a small Linux computer that continuously sends images over the network. The models are AXIS2120 and AXIS2130. We have named them cvl-cam-01 and cvl-cam-00. The price of the first one was 11195 SEK in 2003. The steerable camera cvl-cam-00 is newer and more expensive. The cameras are easy to set on-line (for those who have passwords). We have set up the cameras to the lowest image size and the least JPEG compression, i.e. the best possible image quality. For those who are interested, more information about the cameras can be found at the web address `http://www.axis.com/`.

2.2 Image reading from the network cameras

Images from the cameras can be loaded via a web browser. Try to go to one of the addresses:

```
http://cvl-cam-00.edu.isy.liu.se
http://cvl-cam-01.edu.isy.liu.se
```

QUESTION: Let your lab partner move something in front of the camera. Can you see it on your computer?

Keep in mind that access to network cameras via the network slows down your computer! Images from the cameras can also be loaded into MATLAB. The following code loads a color image from cvl-cam-01 and shows it.

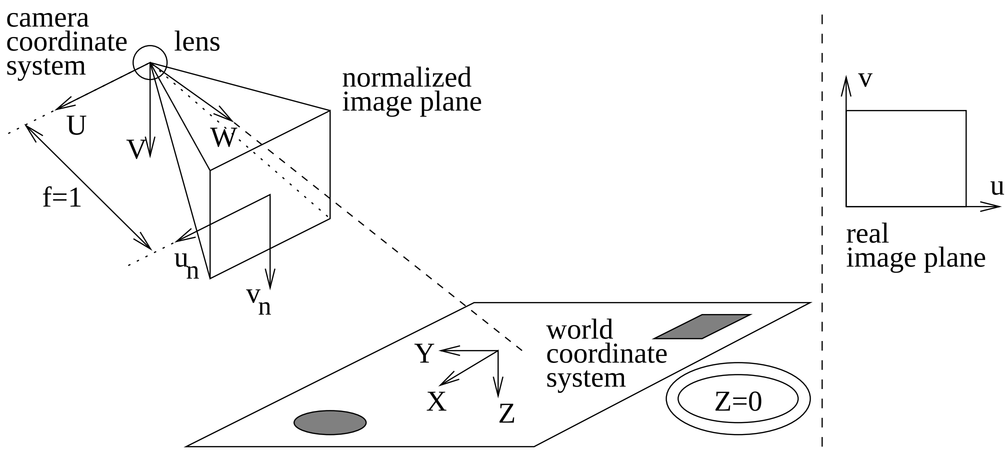
```
figure(1)
bild = imread('http://cvl-cam-01.edu.isy.liu.se/jpg/image.jpg');
imagesc(bild); axis image;
```

QUESTION: Try to load an image! Does the image look good?

2.3 Calibration of a flat world, a homography

This lab task 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 below. 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

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

where s is a scale factor. The matrix 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 C looks as follows,

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

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

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,

$D \cdot c =$

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

Home exercise How many points in the world (X_i, Y_i) are, at least, needed to determine C?

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

The following code loads a color image from cvl-cam-01 and shows it.

```
figure(1);
bild = imread('http://cvl-cam-01.edu.isy.liu.se/jpg/image.jpg');
imagesc(bild); axis image;
```

Load two images, one of the calibration pattern and one of the object you want to measure. The calibration points have the following global coordinates measured in cm:

```
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: Check if this is correct by using a ruler!

QUESTION: 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 You should soon calculate c through your calibration points and the pseudoinverse. The pseudoinverse of a matrix A is written A^+. Write an equation which shows how c can be calculated from D and f.

Now calculate the vector c in Matlab. The Matlab command `pinv` corresponds to the pseudoinverse. For your help there is a non-completed Matlab code in `/site/edu/bb/Bildsensorer/C-CameraCalibration/calibr.m` See also the Matlab section in the end this laboratory assignment.

QUESTION: 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))';
```

QUESTION: Give the matrix C below!

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

QUESTION: You will soon, finally, measure the length of your object. Measure two endpoints (u_a, v_a) and (u_b, v_b):

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

Since previously we know that this is valid:

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

Consequently:

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

QUESTION: Determine

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

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

QUESTION: Determine the end points in world coordinates.

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

QUESTION: Which length does this corresponds to?

QUESTION: Measure the length of the object. What was it and was it rather similar to your calculated value?

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

2.4 Complete calibration of the network camera

image 1

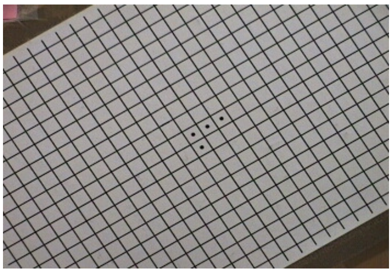


image 2

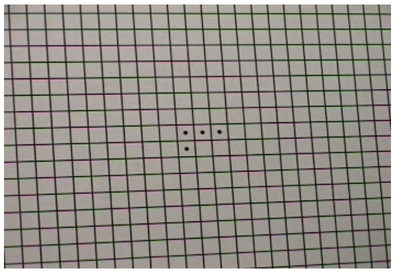


image 3

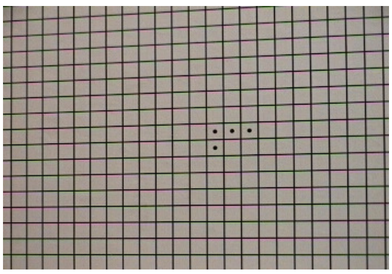
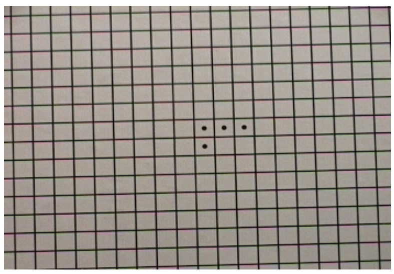


image 4



Here we will perform camera calibration according to Zhang [2]. See the figure above. These four images were obtained during a previous calibration session. The first image, top left, was used as the global 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 `CameraCalibration.m` which calls `generate_homog.m` and `homography2intrinsic.m` is located at `/site/edu/bb/Bildsensorer/C-CameraCalibration/`, see also the MATLAB section in the end this laboratory assignment. 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. In this case the result was:

```
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
```

Using the A-matrix, R-matrix and t-vector, as well as the camera set-up in the lab, answer the following questions.

Home exercise Calculate the distance between the camera center and the world coordinate system origin.

Home exercise Where is the physical camera center located?

QUESTION: Measure the distance between the camera center (assume the rotation center) and the origin of the world coordinate system. Did you get consistency between calculation and measurement?

Home exercise Rotation between camera and world coordinate system has been performed almost exclusively around one axis. Which axis and how much rotation?

QUESTION: Measure the rotation between the camera and the world coordinate system using a protractor (Swedish: gradskiva). Did you get consistency between calculation and measurement?

Home exercise What is the relationship between the pixel distances in the x- and y-direction.

Home exercise 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?

Home exercise How big is the skew angle ($\xi = \arctan(\gamma/\beta)$) between the image pixels? Give a measurement in degrees.

The teacher will now show the previous calibration session. QUESTION: The matrices will be similar to the ones in this document, but more exact for the current case. Write down the matrices below and check so that they are similar to the ones given in this document.

```
A = _______ * _______    _______    _______
              _______    _______    _______
              _______    _______    _______

R =           _______    _______    _______
              _______    _______    _______
              _______    _______    _______

t =           _______
              _______
              _______
```

The teacher will now perform another calibration session in which the camera zooms in.

QUESTION: Estimate, together with the teacher, how much larger the pixels are now compared to before. Do it by calculating squares in the images.

QUESTION: Write down the new matrices below.

```
A = _______ *  _______   _______   _______
               _______   _______   _______
               _______   _______   _______

R =            _______   _______   _______
               _______   _______   _______
               _______   _______   _______

t =            _______
               _______
               _______
```

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

QUESTION: Calculate the distance between the camera center and the world coordinate system origin. Is the distance $\approx$ the same as before?

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

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

QUESTION: The image size is 352 × 240, giving the center at $(176, 120)$. 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.

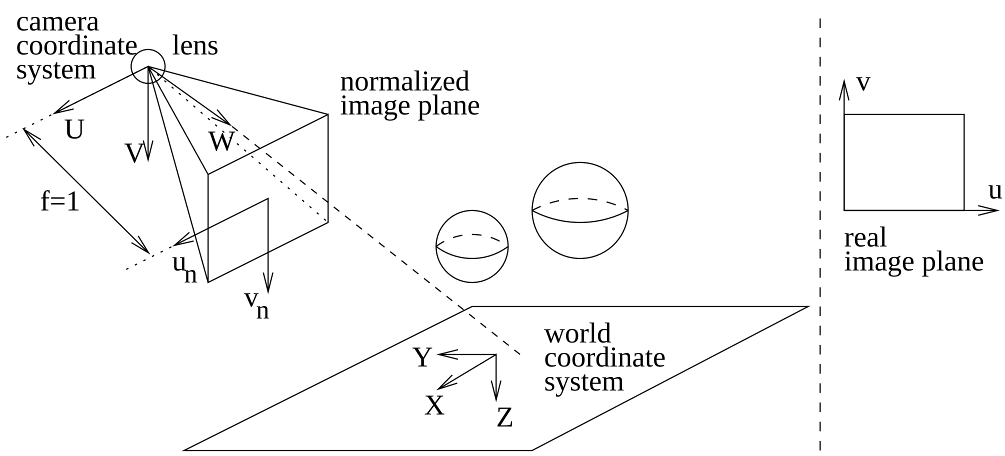
2.5 To follow an object with the network camera

Home exercise 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.



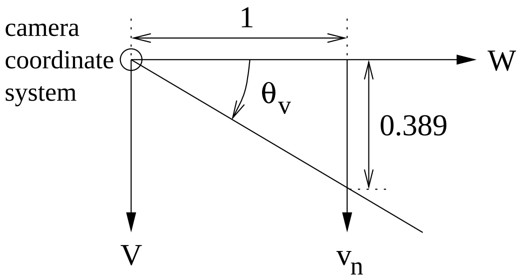
The camera's task is to follow an object. It can rotate in two angular directions θ_u and θ_v (horizontally and vertically). The image center is located at the coordinate $(u, v) = (250, 200)$. Suppose that we have located the object at the coordinate $(u, v) = (250 + 225, 200 + 175) = (475, 375)$ in 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 See the four calibration images with their A-matrix in the previous section. 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?

QUESTION: Make the previous calculation again, but use the more exact A-matrix, measured today in the computer exercise room.

Now you are going to control your calculations using cvl-cam-00. Perform as follows.

- Book cvl-cam-00 on the white board.
- Bring in an image from cvl-cam-00 in the web browser.
- Note that a small red 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.)
- Add the following path in the MATLAB window:
  `addpath /site/edu/bb/Bildsensorer/C-CameraCalibration`
  Then view the code `move2basic.m`. It steers the camera to its original location and shows an image on your screen. Execute it and check that you can see the red patch on your screen.
- Copy the file `rotate2corner.m` to your directory and study it. It steers the camera to the desired angles and displays an image on your computer. Replace the angles `ptz.pan` and `ptz.tilt` with your calculated angles. Execute!
- Cancel the booking of cvl-cam-00 on the white board.

QUESTION: Did you manage to move the camera so that the red patch was in the optical center of the image?

2.6 From C to A, R, t

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 2.3, 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. There are methods to extract A, R and t from C, see the code in `P2KRt.m`.

QUESTION: Take one set of A, R, t-matrices from section 2.4: "Complete calibration of a network camera." Multiply them together as $C = A[Rt]$. Now send C to `P2KRt.m`. What is the result?

QUESTION: This seems to be simpler than Zhang's method. What is the advantages 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.
%
% Programmed 2004 by a project group supervised by
% Per-Erik Forsse'n.
% Modified 2004-10-06 and 2007-10-10 by Maria Magnusson.
% Modified 2015-11-05 by Maria Magnusson (just comments).

% 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.
%
% Programmed 2004 by a project group supervised by
% Per-Erik Forsse'n.
% Modified 2004-10-06 by Maria Magnusson Seger.

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])
%
% Programmed 2004 by a project group supervised by
% Per-Erik Forsse'n.
% Modified 2004-10-06 by Maria Magnusson Seger.

% 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];
```