Lund University

Computer Vision

FMAN95

Assignment 2

Nils Broman

12/03/2021

Contents

2 Calibrated vs. Uncalibrated Recunstruction.

2.1 E1

$$\text{Let } \tilde{\mathbb{X}} = T\mathbb{X}\,, \quad \text{then}$$
$$\lambda x = P\mathbb{X} = PT^{-1}T\mathbb{X} = PT^{-1}\tilde{\mathbb{X}}$$

2.2 CE1

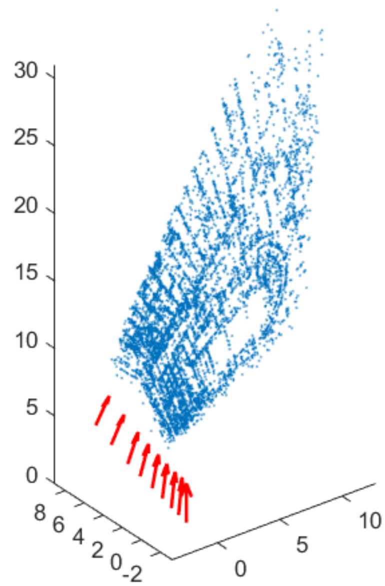

Figure 1: Original 3D reconstruction. Distorted

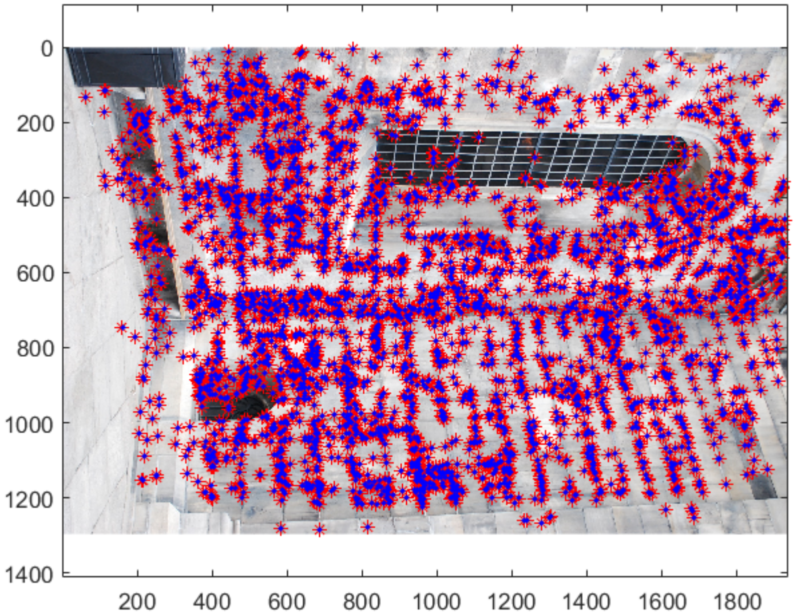

Figure 2: Original points (red) and projected from 3D construction (blue). Appear to be very similar.

.

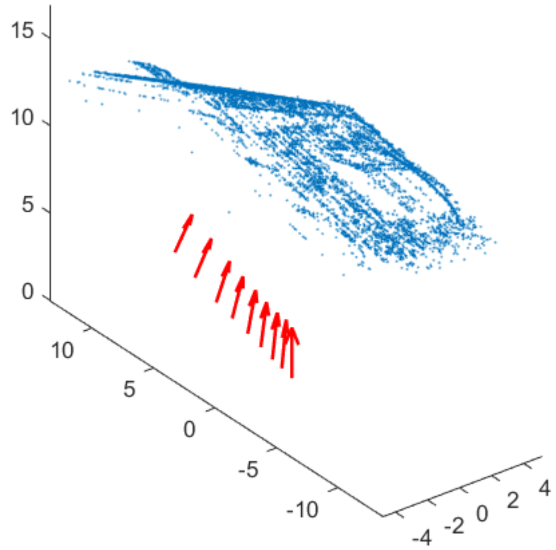

Figure 3: 3D Construction with transformation T_1. Appears to be wider than the true construction, due to $T_1(2,2) = 4T_1(1,1)$, as well as the cameras being from a different angle (higher upp) due to $T_1(4,1), T_1(4,2)$

.

.

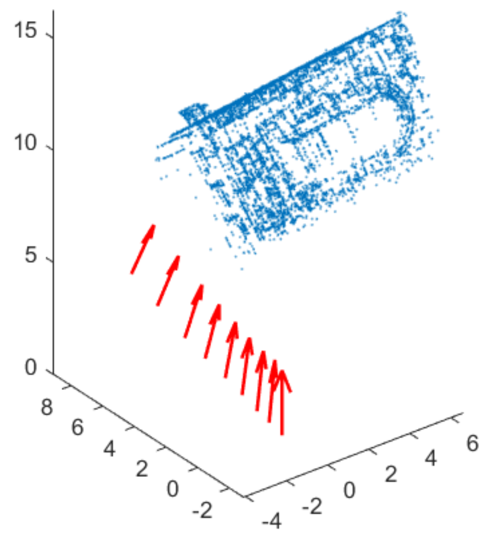

Figure 4: 3D construction with transformation T_2. This looks like a good construction.

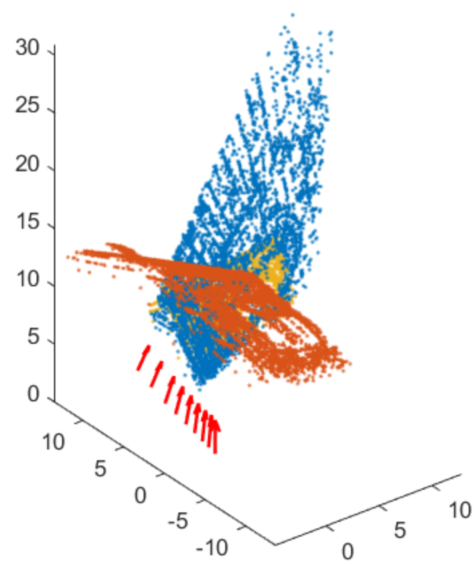

Figure 5: All three constructions in same plot. Original (blue), T_1 (orange) and T_2 yellow.

.

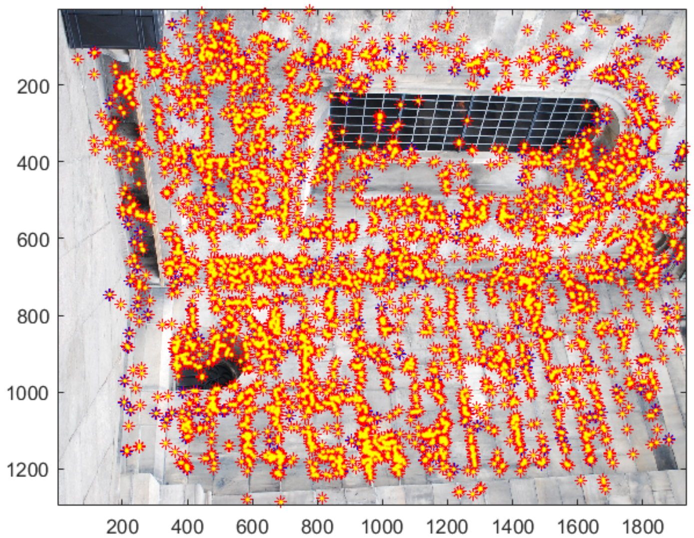

Figure 6: Projected points from T_1 (red) and T_2 (yellow). No change, as expected from E1.

2.3 E2

Calibrated cameras requires that R rotation and t translation only, and only euclidean (similarity) transformations fulfill this criteria.

3 Camera Calibration

3.1 E3

Verified by computing KK^{-1} $(= I)$ and AB $(= K^{-1})$. A scales, B moves (x_0, y_0) to origin. Points of distance f ends up on distance 1.

Using formulas from assignment we find

$$K = \begin{bmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$K^{-1} = \begin{bmatrix} 1/320 & 0 & -1 \\ 0 & 1/320 & -3/4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/320 & 0 & 0 \\ 0 & 1/320 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -320 \\ 0 & 1 & -240 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$\vec{a} = \begin{bmatrix} 0 \\ 240 \\ 1 \end{bmatrix} \qquad \vec{b} = \begin{bmatrix} 640 \\ 240 \\ 1 \end{bmatrix} \tag{3}$$

$$\tilde{a} = K^{-1}\vec{a} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad \tilde{b} = K^{-1}\vec{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \tag{4}$$

$$\angle \tilde{a}\tilde{b} = \arccos(\tilde{a}\tilde{b}) = \pi/2 \tag{5}$$

K is upper triangular with $K_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, and has zero null space and wont change R_3, or for any upper triangular K will only scale R3 in the multiplication KR. This means K is invertible, and therefor:

$$K \begin{bmatrix} R & t \end{bmatrix} = 0 \Rightarrow K^{-1}K \begin{bmatrix} R & t \end{bmatrix} = 0 \Rightarrow [Rt] = 0 \tag{6}$$

And since the principal axis is determined by R_3 it is not changed at all by this K, and for any upper triangular K it would only be scaled, not changing its direction.

3.2 E4

K and K^{-1} from formula in E3. Multiply P with K^{-1}, results in a calibrated camera

$$P = K \begin{bmatrix} R & t \end{bmatrix} = \begin{bmatrix} 1000 & 0 & 500 \\ 0 & 1000 & 500 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & 0.5 & 0 \\ 0 & -0.5 & 0.866 & 1 \end{bmatrix} \tag{7}$$

Corners and center maps to

$$\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}-1/2\\-1/2\\1\end{bmatrix} \qquad \begin{bmatrix}1000\\0\\1\end{bmatrix} = \begin{bmatrix}1/2\\-1/2\\1\end{bmatrix} \tag{8}$$

$$\begin{bmatrix}0\\1000\\1\end{bmatrix} = \begin{bmatrix}-1/2\\1/2\\1\end{bmatrix} \qquad \begin{bmatrix}1000\\1000\\1\end{bmatrix} = \begin{bmatrix}1/2\\1/2\\1\end{bmatrix} \tag{9}$$

$$\begin{bmatrix}500\\500\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{10}$$

4 RQ Factorization and Computation of K

4.1 E5

$$A_3 = fR_3 = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \Rightarrow \qquad R_3^T = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \qquad f = 1$$

$$e = A_2^T R_3 = 700$$

$$dR_2 = A_2 - eR_3 = \begin{bmatrix} 0 \\ 1400 \\ 0 \end{bmatrix} \Rightarrow \qquad R_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad d = 1400$$

$$c = A_1^T R_3 = 800 \qquad b = A_1^T R_2 = 0$$

$$aR_1 = A_1 - c_{R3} = 1600 \begin{bmatrix} -1/\sqrt{(2)} \\ 91/\sqrt{2} \end{bmatrix} \Rightarrow \qquad R_1 = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \qquad a = 1600$$

this gives us

$$K = \begin{bmatrix} 1600 & 0 & 800 \\ 0 & 1400 & 700 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

So
focal length = 1400,
skew = 0,
aspect ratio = 8/7,
principal point = (800, 700).

4.2 CE2

$$K_1 = 10^3 \begin{bmatrix} 2.3940 & 0 & 0.9324 \\ 0 & 9.5925 & 0.6283 \\ 0 & 0 & 0.0010 \end{bmatrix} \quad K_2 = 10^3 \begin{bmatrix} 2.3744 & -0.2717 & 0.9027 \\ 0 & 2.1713 & 0.6153 \\ 0 & 0 & 0.0010 \end{bmatrix} \tag{12}$$

These do not represent the same transformation, e.g. K_1 will stretch it out more in the y-direction.

5 Direct Linear Transformation DLT

5.1 E7

$$P = N^{-1}\tilde{P} \tag{13}$$

5.2 CE3

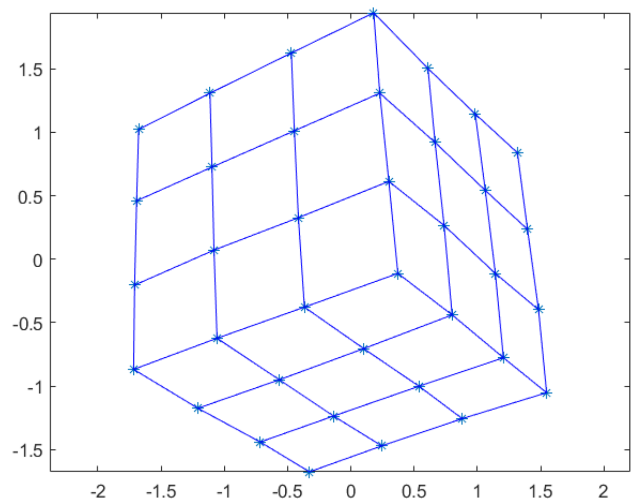

Figure 7: Normalized points image 1

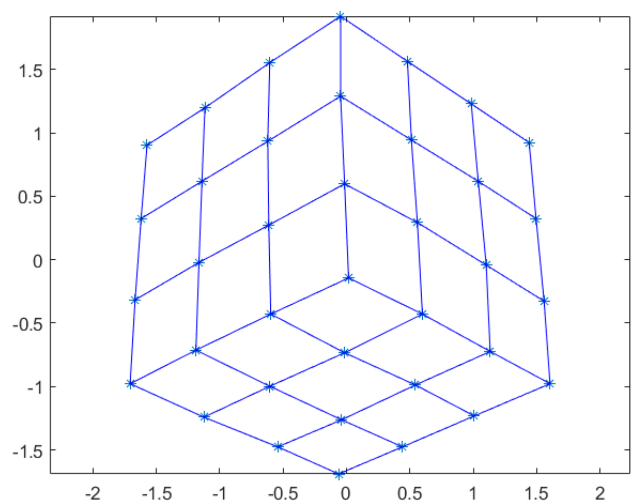

Figure 8: Normalized points image 2

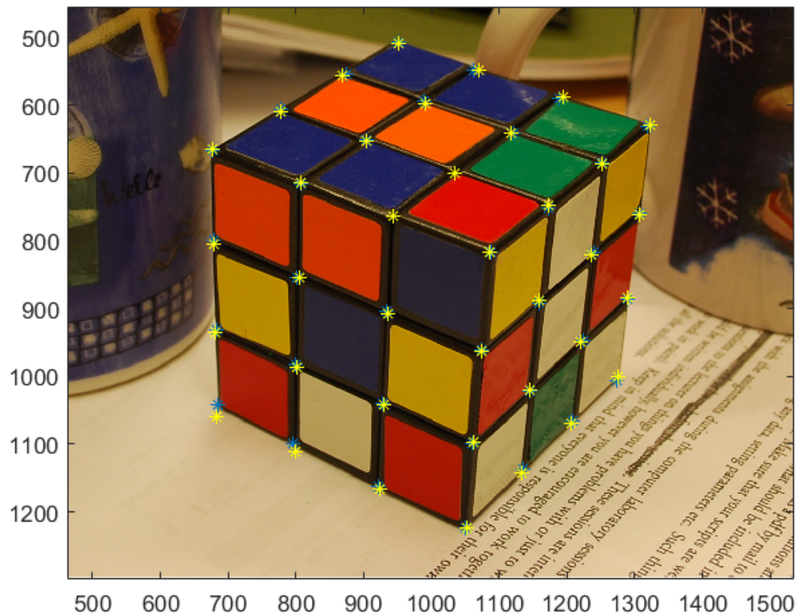

Figure 9: Original points (blue) together with projected model points (yellow) on image 1.

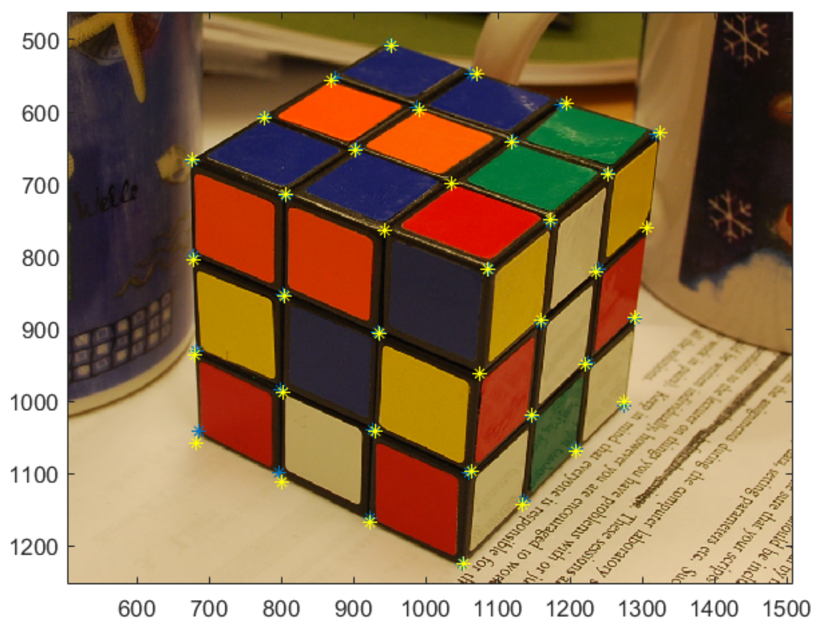

Figure 10: Original points (blue) together with projected model points (yellow) on image 2.

.

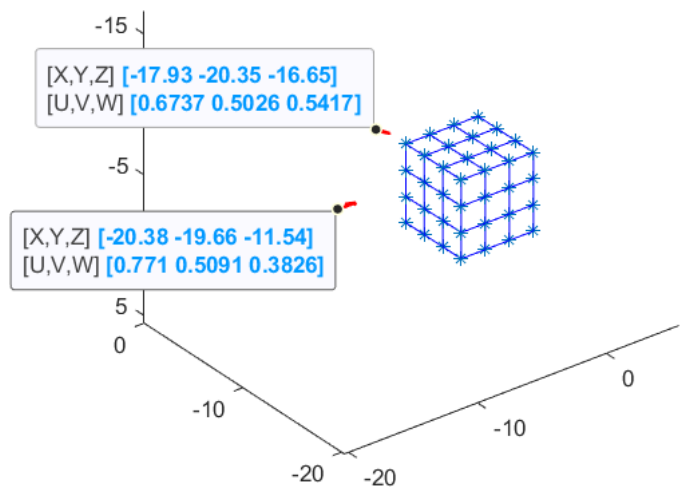

Figure 11: Camera points with viewing directions together with model points.

$$K_1 = 10^3 \begin{bmatrix} 2.5348 & -0.0399 & 1.2060 \\ 0 & 2.5615 & 0.8098 \\ 0 & 0 & 0.0010 \end{bmatrix} K_2 = 10^3 \begin{bmatrix} 2.6674 & -0.0436 & 1.1071 \\ 0 & 2.6862 & 0.7363 \\ 0 & 0 & 0.0010 \end{bmatrix} \tag{14}$$

Note the 10^3 outside matrix.

7 Triangulation using DLT

7.1 CE5

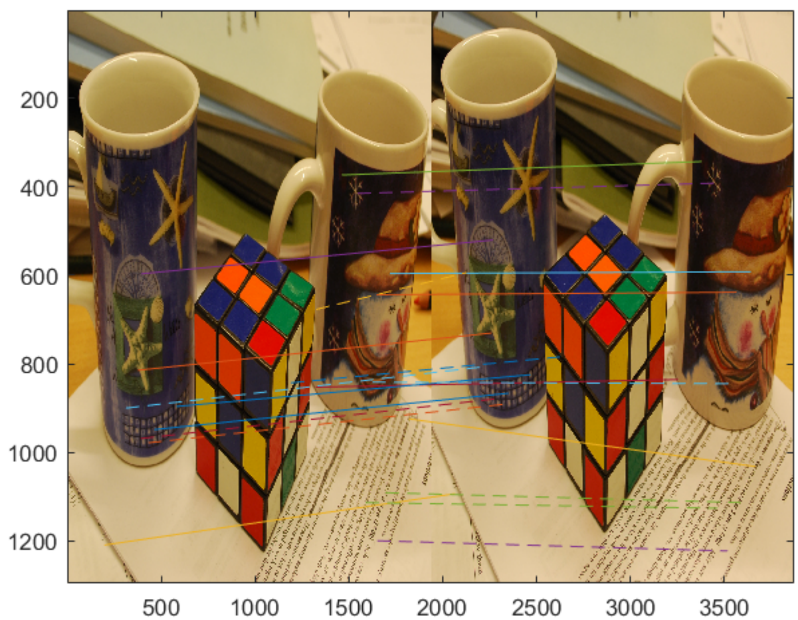

Figure 12: Original SIFT-pints (lines) and computed points (dashed lines)

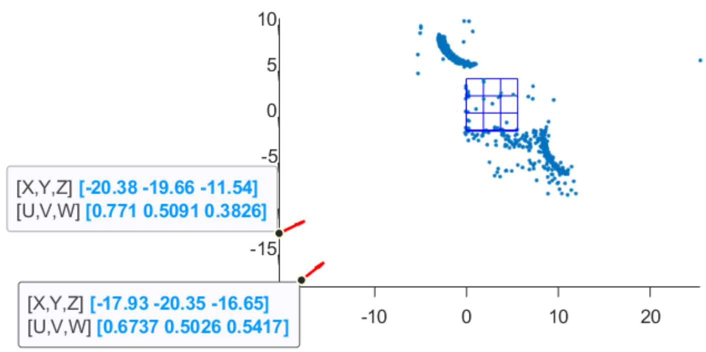

Figure 13: Reconstruction as seen from above

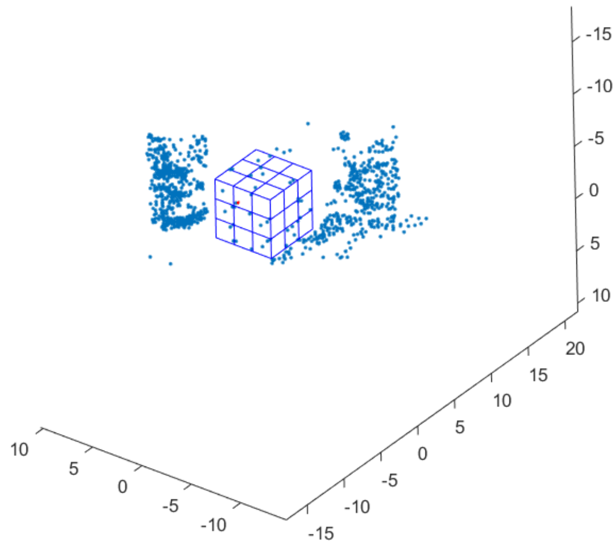

Figure 14: Reconstruction as seen from viewing direction

.

8 Appendix

Listing 1: main.m

```
%% CE 1
load('compEx1data.mat')
%%
figure(1)
plot3(X(1,:), X(2,:), X(3,:), '.', 'Markersize', 1)
hold on
plotcams(P)
axis equal
hold off

%% compute and plot projection and original points from camera 1.
im = imread(imfiles{1});
visible = isfinite(x{1}(1,:));

x1 = pflat(P{1}*X);

figure(2)
imagesc(im)
hold on
plot(x{1}(1,visible), x{1}(2,visible), 'r*')
plot(x1(1,visible), x1(2,visible), 'b*', 'Markersize', 2)
axis equal
hold off

%% Compute and reconstructs using transformation T1
T1 = [1      0    0    0 ;
      0      4    0    0 ;
      0      0    1    0 ;
      1/10 1/10 0    1];

T1X = pflat(T1*X);

figure(3)
plot3(T1X(1,:), T1X(2,:), T1X(3,:), '.', 'Markersize', 1)
hold on
plotcams(P)
axis equal
hold off

```

```
%% Compute and reconstructs using transformation T2

T2 = [1      0    0    0 ;
      0      1    0    0 ;
      0      0    1    0 ;
      1/16  1/16  0    1];

T2X = pflat(T2*X);

figure(4)
plot3(T2X(1,:), T2X(2,:), T2X(3,:), '.', 'Markersize', 1)
hold on
plotcams(P)
axis equal
hold off

%% All reconstructions
figure(5)
plot3(X(1,:), X(2,:), X(3,:), '.', 'Markersize', 3)
hold on
plot3(T1X(1,:), T1X(2,:), T1X(3,:), '.', 'Markersize', 3)
plot3(T2X(1,:), T2X(2,:), T2X(3,:), '.', 'Markersize', 3)
plotcams(P)
axis equal
hold off

%% All projected

T1x1 = pflat(P{1}*T1X);
T2x1 = pflat(P{1}*T2X);


figure(6)
imagesc(im)
hold on
plot(x{1}(1,visible), x{1}(2,visible), '*')
plot(x1(1,visible), x1(2,visible), '*')
plot(T1x1(1,visible), T1x1(2,visible), '*')
plot(T2x1(1,visible), T2x1(2,visible), '*')


hold off
```

```
%% CE 2

K1 = rq(P{1}*T1)
K2 = rq(P{2}*T2)


%% CE 3 - Load and plot images
clear all
load('compEx3data.mat');
q1 = imread('cube1.JPG');
q2 = imread('cube2.JPG');

figure(7)
imagesc(q1)
hold on
plot(x{1}(1,:), x{1}(2,:), '*')
axis equal
hold off

figure(8)
imagesc(q2)
hold on
plot(x{2}(1,:), x{2}(2,:), '*')
axis equal
hold off

%% Compute xTilde and N for figure 1
mean1 = mean(x{1}(1:2,:),2);
std1 = std(x{1}(1:2,:),0,2);

N1 = [1/std1(1) 0           -mean1(1)/std1(1);
      0          1/std1(2)   -mean1(2)/std1(2);
      0          0           1                ];
 x1Tilde = N1*x{1};

%% Plot fig 1 osv
figure(9)
plot(x1Tilde(1,:), x1Tilde(2,:), '*')
hold on
plot([ x1Tilde(1,startind ); x1Tilde(1,endind )],...
    [x1Tilde(2,startind ); x1Tilde(2,endind )],...
    'b-');
hold off
```

```
axis equal

figure(10)
plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
hold on
plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
    [Xmodel(2,startind ); Xmodel(2,endind )],...
    [Xmodel(3,startind ); Xmodel(3,endind)],'b-');
hold off
axis equal

%% Assemble M-matrix and compute v, norm(M*v) etc.

X=[Xmodel;ones(1, 37)];
M1 = [];

for i = 1:18
    M1(i*3 -2, 1:4) = X(1:4,i)';
    M1(i*3 -1, 5:8) = X(1:4,i)';
    M1(i*3 , 9:12) = X(1:4,i)';
    M1(i*3 -2:i*3, i+12) = -x1Tilde(1:3, i);
end

[U,S,V] = svd(M1);
v1 = V(:,end)
min(diag(S))
norm(M1*v1) %same regardless of which column?

P1Tilde = [v1(1:4)';v1(5:8)';v1(9:12)']
P1 = N1\P1Tilde

x1 = P1*X;
x1flat = pflat(x1);

%%
figure(7)
imagesc(q1)
hold on
plot(x{1}(1,:), x{1}(2,:), '*')
plot(x1flat(1,:),x1flat(2,:), 'y*')
axis equal
hold off
```

```
%% Compute xTilde and N for figure
mean2 = mean(x{2}(1:2,:),2);
std2 = std(x{2}(1:2,:),0,2);

N2 = [1/std2(1) 0           -mean2(1)/std1(1);
      0         1/std2(2)   -mean2(2)/std1(2);
      0         0           1               ];

x2Tilde = N2*x{2};

%% Plot
figure(10)
plot(x2Tilde(1,:), x2Tilde(2,:), '*')
hold on
plot([ x2Tilde(1,startind ); x2Tilde(1,endind )],...
    [x2Tilde(2,startind ); x2Tilde(2,endind )],...
    'b-');
hold off
axis equal

figure(11)
plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
hold on
plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
    [Xmodel(2,startind ); Xmodel(2,endind )],...
    [Xmodel(3,startind ); Xmodel(3,endind)],'b-');
hold off
axis equal

%% Assemble M-matrix and compute v, norm(M*v) etc.

X=[Xmodel;ones(1, 37)];
M2 = [];

for i = 1:18
    M2(i*3 -2, 1:4) = X(1:4,i)';
    M2(i*3 -1, 5:8) = X(1:4,i)';
    M2(i*3 , 9:12) = X(1:4,i)';
    M2(i*3 -2:i*3, i+12) = -x2Tilde(1:3, i);
end

[U,S,V] = svd(M2);
v2 = V(:,end);
```

```
min(diag(S))
norm(M2*v2) %same regardless of which column?

P2Tilde = [v2(1:4)';v2(5:8)';v2(9:12)']
P2 = N2\P2Tilde

x2 = P2*X;
x2flat = pflat(x2);

%%
figure(12)
imagesc(q1)
hold on
plot(x{1}(1,:), x{1}(2,:), '*')
plot(x1flat(1,:),x1flat(2,:), 'y*')
axis equal
hold off

%%
cameras = {P1,P2};

figure(13)
plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
hold on
plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
    [Xmodel(2,startind ); Xmodel(2,endind )],...
    [Xmodel(3,startind ); Xmodel(3,endind)],'b-');
plotcams(cameras);
hold off
set(gca, 'Zdir', 'reverse')
axis equal

%%

[R1,Q1] = rq(P1);
[R2,Q2] = rq(P2);
K1ce3 = R1./R1(3,3)
K2ce3 = R2./R2(3,3)

%% CE 4

q1 = imread('cube1.jpg');
q2 = imread('cube2.jpg');
```

```
run vl_setup.m

[f1, d1] = vl_sift( single(rgb2gray(q1)), 'PeakThresh', 1);
[f2, d2] = vl_sift( single(rgb2gray(q2)), 'PeakThresh', 1);

figure(14)
imagesc(q1);
hold on
vl_plotframe(f1);
hold off
axis equal

figure(15)
imagesc(q2);
vl_plotframe(f2);
hold off
axis equal


[matches ,scores] = vl_ubcmatch(d1,d2);

x1 = [f1(1,matches (1 ,:));f1(2,matches (1 ,:))];
x2 = [f2(1,matches (2 ,:));f2(2,matches (2 ,:))];

perm = randperm(size(matches ,2));
figure(16);
imagesc ([q1 q2]);
hold on;
plot([x1(1,perm (1:10)); x2(1,perm (1:10))+ size(q1 ,2)], ...
    [x1(2,perm (1:10));  x2(2,perm (1:10))] ,'-') ;

hold  off;

%% Ce 5 - Set up and solves DLT for trianhylation.

Xtriag = [];

for i = 1:length(x1)
    Mce5 = [cameras{1} -[x1(:,i);1]     zeros(3,1) ;
            cameras{2} zeros(3,1)        -[x2(:, i);1]];
    [U,S,V] = svd(Mce5);
    v = V(:,end);
    Xtriag = [Xtriag v(1:4)];
```

```
end

%% Project triangulated points onto images.
Xflat = pflat(Xtriag);
xproj1 = pflat(cameras{1}*Xflat);
xproj2 = pflat(cameras{2}*Xflat);



perm = randperm(size(matches ,2));
figure(16)
% imagesc ([q1 q2]);
hold on;
plot([xproj1(1,perm (1:10)); xproj2(1,perm (1:10))+ size(q1 ,2)], ...
    [xproj1(2,perm (1:10));  xproj2(2,perm (1:10))] ,'--') ;
hold  off;

good_points = (sqrt(sum((x1-xproj1 (1:2 ,:)).^2))  < 3 &   ...
    sqrt(sum((x2-xproj2 (1:2 ,:)).^2))  < 3);


Xgood = Xflat(:,good_points);


%%
figure(17)
plot3(Xgood(1,:), Xgood(2,:), Xgood(3,:), '.')
hold on
plot3([ Xmodel(1,startind );  Xmodel(1,endind )],...
    [Xmodel(2,startind );  Xmodel(2,endind )],...
    [Xmodel(3,startind );  Xmodel(3,endind)],'b-');
plotcams(cameras);
hold off
axis equal
%%

p1n = rq(P1)\P1;
p2n = rq(P2)\P2;
```