

LUND UNIVERSITY

COMPUTER VISION

FMAN95

---

## Assignment 2

---

Nils Broman

12/03/2021

# Contents

<b>2</b>	<b>Calibrated vs. Uncalibrated Recunstruction.</b>	<b>1</b>
2.1	E1 . . . . .	1
2.2	CE1 . . . . .	1
2.3	E2 . . . . .	5
<b>3</b>	<b>Camera Calibration</b>	<b>6</b>
3.1	E3 . . . . .	6
3.2	E4 . . . . .	6
<b>4</b>	<b>RQ Factorization and Computation of <math>K</math></b>	<b>9</b>
4.1	E5 . . . . .	9
4.2	CE2 . . . . .	9
<b>5</b>	<b>Direct Linear Transformation DLT</b>	<b>10</b>
5.1	E7 . . . . .	10
5.2	CE3 . . . . .	10
<b>7</b>	<b>Triangulation using DLT</b>	<b>13</b>
7.1	CE5 . . . . .	13
<b>8</b>	<b>Appendix</b>	<b>15</b>

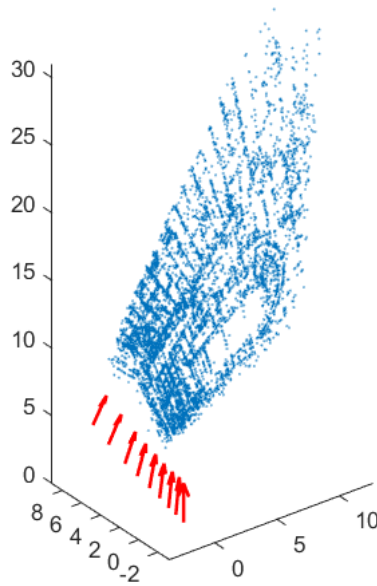
## 2 Calibrated vs. Uncalibrated Reconstruction.

### 2.1 E1

Let  $\tilde{\mathbb{X}} = T\mathbb{X}$ , 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*

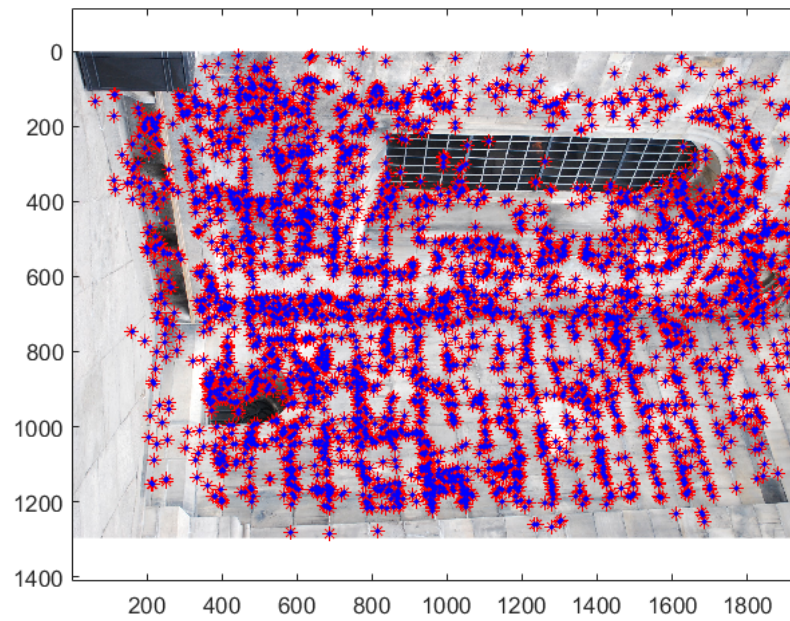


Figure 2: Original points (red) and projected from 3D construction (blue). Appear to be very similar.

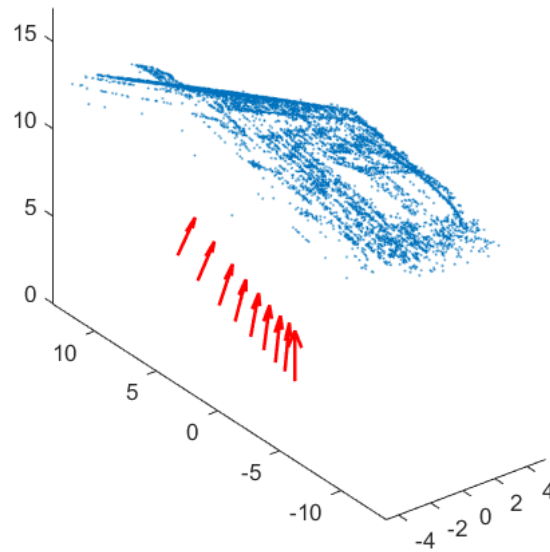


Figure 3: 3D Construction with transformation  $T_1$ . Appears to be wider than the true construction, due to  $T_1(2, 2) = 4T_1(1, 1)$ , as well as the cameras being from a different angle (higher up) due to  $T_1(4, 1), T_1(4, 2)$

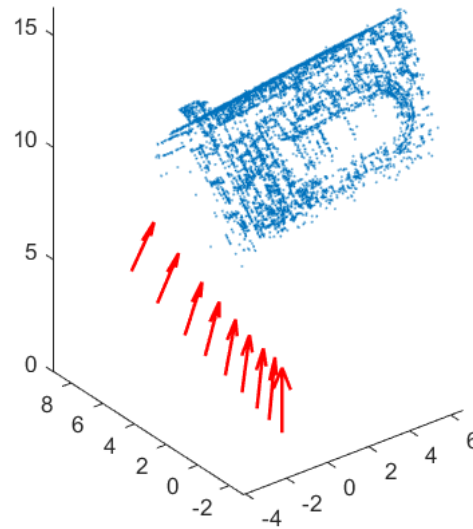


Figure 4: 3D construction with transformation  $T_2$ . This looks like a good construction.

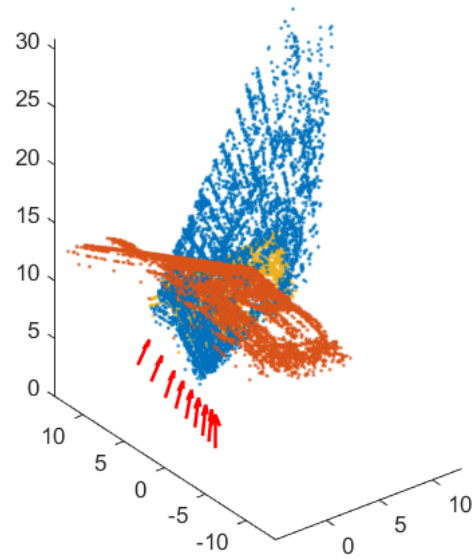


Figure 5: All three constructions in same plot. Original (blue),  $T_1$  (orange) and  $T_2$  yellow.

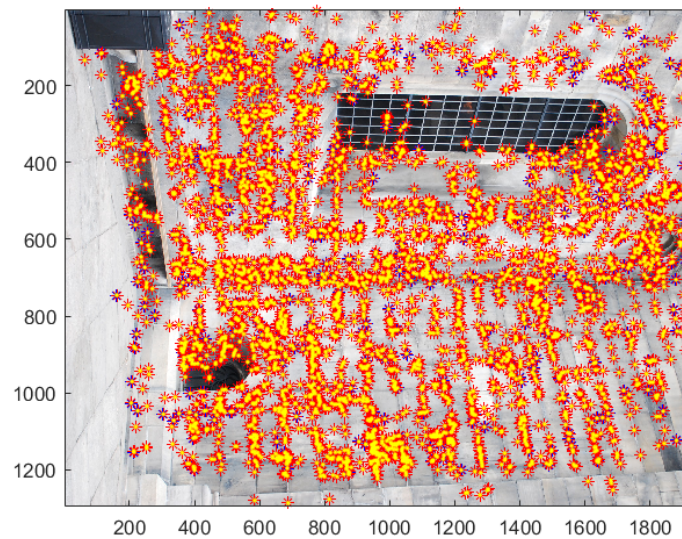


Figure 6: Projected points from  $T_1$  (red) and  $T_2$  (yellow). No change, as expected from E1.

## 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

$$K = \begin{bmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

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

$$\vec{a} = \begin{bmatrix} 0 \\ 240 \\ 1 \end{bmatrix} \quad \vec{b} = \begin{bmatrix} 640 \\ 240 \\ 1 \end{bmatrix} \quad (3)$$

$$\tilde{a} = K^{-1}\vec{a} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \tilde{b} = K^{-1}\vec{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad (4)$$

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

$K$  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  $R_3$  in the multiplication  $KR$ . This means  $K$  is invertible, and therefor:

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

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



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

Corners and center maps to

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

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

$$\begin{bmatrix} 500 \\ 500 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad (10)$$

## 4 RQ Factorization and Computation of $K$

### 4.1 E5

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

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

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

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

$$aR_1 = A_1 - cR_3 = 1600 \begin{bmatrix} -1/\sqrt{2} \\ 91/\sqrt{2} \end{bmatrix} \Rightarrow R_1 = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \quad a = 1600$$

this gives us

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

So

**focal length** = 1400,

**skew** = 0,

**aspect ratio** = 8/7,

**principal point** = (800, 700).

### 4.2 CE2

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

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

$$P = N^{-1}\tilde{P} \quad (13)$$

### 5.2 CE3

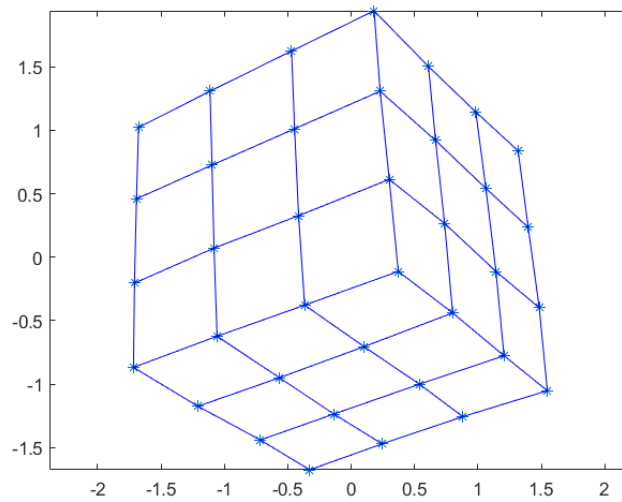


Figure 7: Normalized points image 1

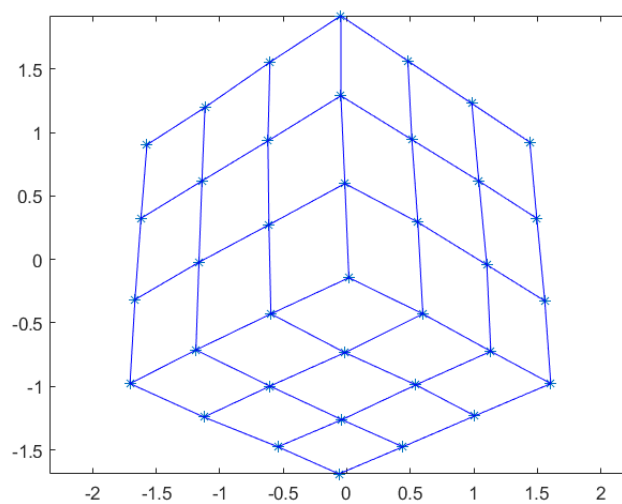


Figure 8: Normalized points image 2

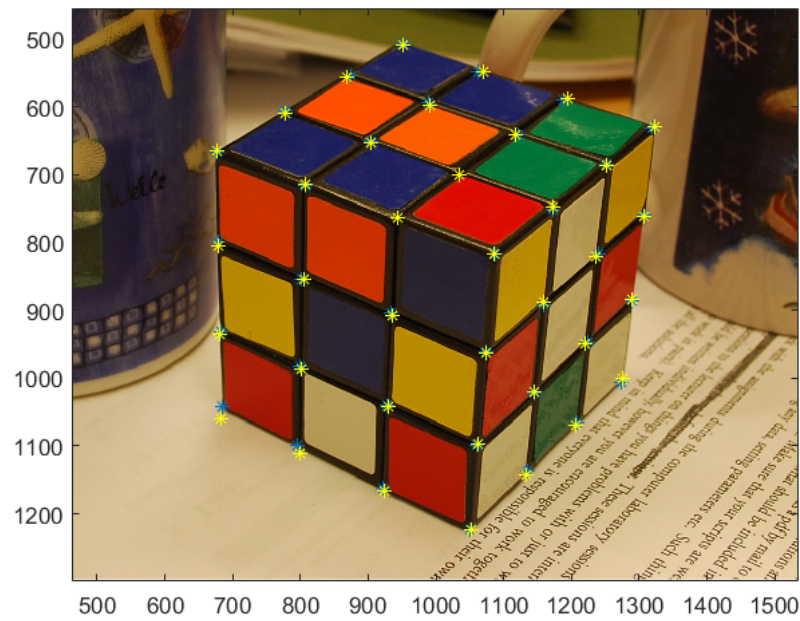


Figure 9: Original points (blue) together with projected model points (yellow) on image 1.

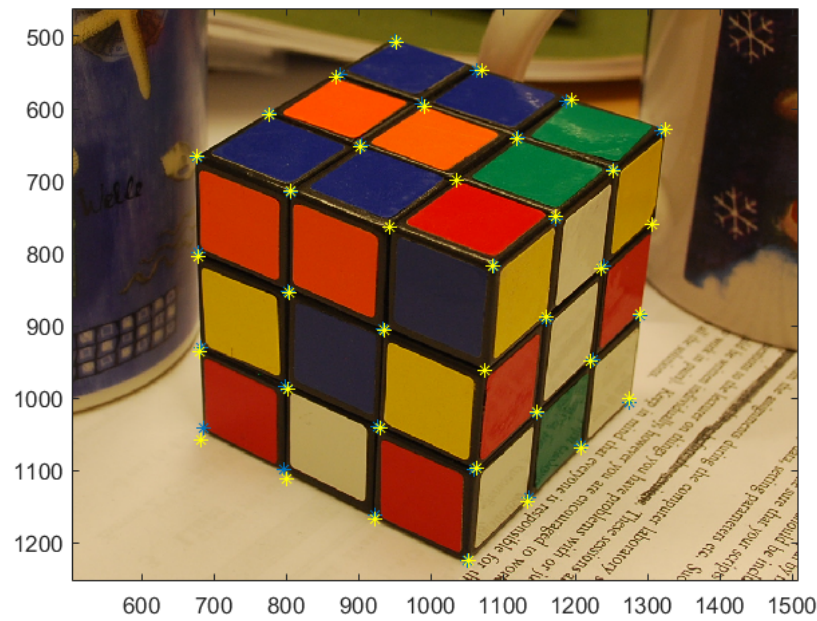


Figure 10: Original points (blue) together with projected model points (yellow) on image 2.

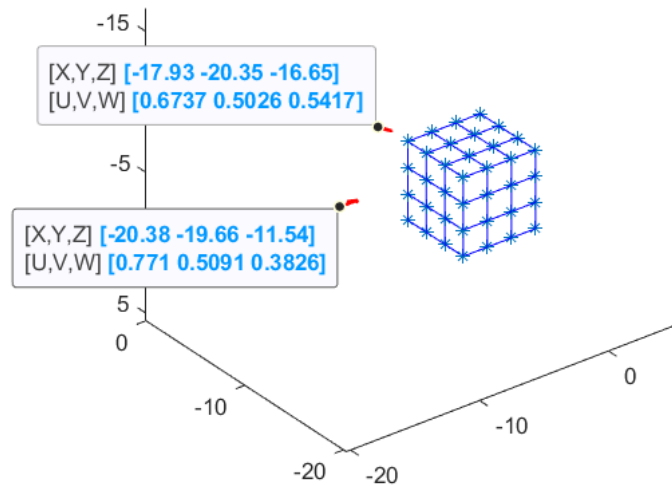


Figure 11: Camera points with viewing directions together with model points.

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

Note the  $10^3$  outside matrix.

## 7 Triangulation using DLT

### 7.1 CE5

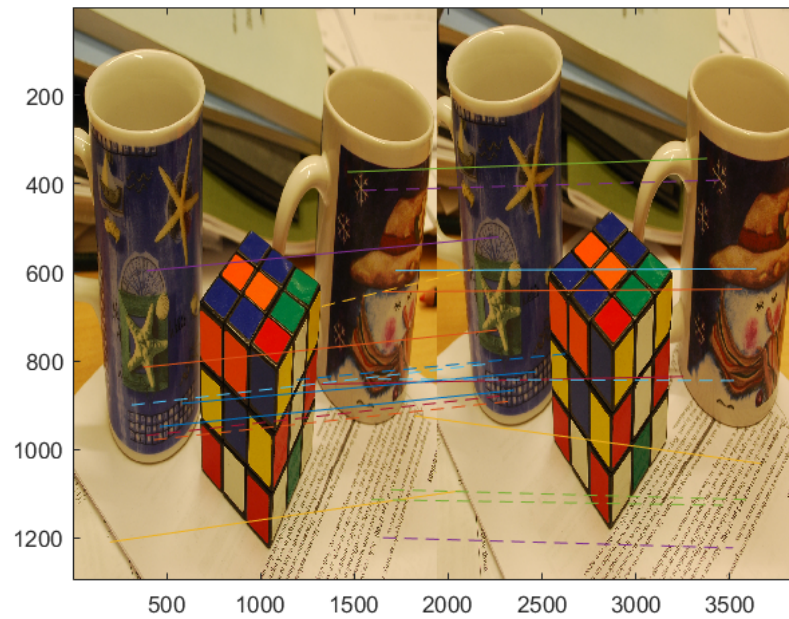


Figure 12: Original SIFT-pints (lines) and computed points (dashed lines)

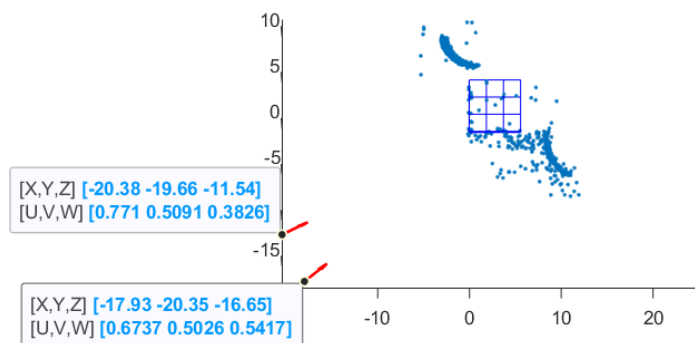


Figure 13: Reconstruction as seen from above

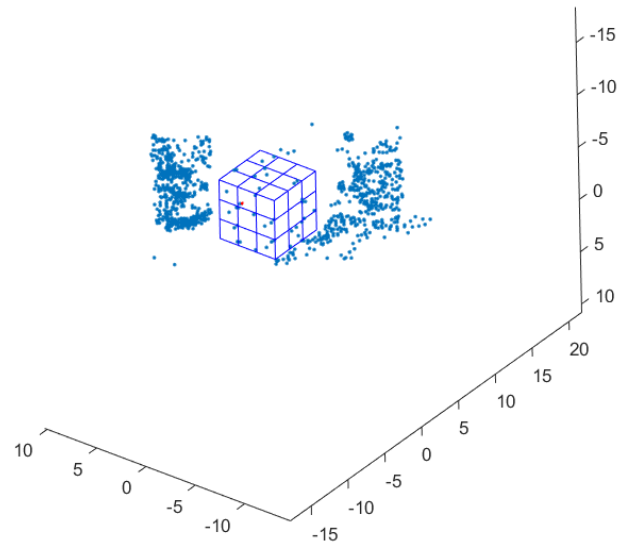


Figure 14: Reconstruction as seen from viewing direction



## 8 Appendix

Listing 1: main.m

```

1  %% CE 1
2  load('compEx1data.mat')
3  %%
4  figure(1)
5  plot3(X(1,:), X(2,:), X(3,:), '.', 'Markersize', 1)
6  hold on
7  plotcams(P)
8  axis equal
9  hold off
10
11 %% compute and plot projection and original points from camera 1.
12 im = imread(imfiles{1});
13 visible = isfinite(x{1}(1,:));
14
15 x1 = pflat(P{1}*X);
16
17 figure(2)
18 imagesc(im)
19 hold on
20 plot(x{1}(1,visible), x{1}(2,visible), 'r*')
21 plot(x1(1,visible), x1(2,visible), 'b*', 'Markersize', 2)
22 axis equal
23 hold off
24
25 %% Compute and reconstructs using transformation T1
26 T1 = [1      0      0      0 ;
27       0      4      0      0 ;
28       0      0      1      0 ;
29       1/10  1/10  0      1];
30
31 T1X = pflat(T1*X);
32
33 figure(3)
34 plot3(T1X(1,:), T1X(2,:), T1X(3,:), '.', 'Markersize', 1)
35 hold on
36 plotcams(P)
37 axis equal
38 hold off
39

```

```

40 %% Compute and reconstructs using transformation T2
41
42 T2 = [1      0      0      0 ;
43       0      1      0      0 ;
44       0      0      1      0 ;
45       1/16  1/16  0      1];
46
47 T2X = pflat(T2*X);
48
49 figure(4)
50 plot3(T2X(1,:), T2X(2,:), T2X(3,:), '.', 'Markersize', 1)
51 hold on
52 plotcams(P)
53 axis equal
54 hold off
55
56 %% All reconstructions
57 figure(5)
58 plot3(X(1,:), X(2,:), X(3,:), '.', 'Markersize', 3)
59 hold on
60 plot3(T1X(1,:), T1X(2,:), T1X(3,:), '.', 'Markersize', 3)
61 plot3(T2X(1,:), T2X(2,:), T2X(3,:), '.', 'Markersize', 3)
62 plotcams(P)
63 axis equal
64 hold off
65
66 %% All projected
67
68 T1x1 = pflat(P{1}*T1X);
69 T2x1 = pflat(P{1}*T2X);
70
71
72 figure(6)
73 imagesc(im)
74 hold on
75 plot(x{1}(1,visible), x{1}(2,visible), '*')
76 plot(x1(1,visible), x1(2,visible), '*')
77 plot(T1x1(1,visible), T1x1(2,visible), '*')
78 plot(T2x1(1,visible), T2x1(2,visible), '*')
79
80
81 hold off
82

```

```

83 %% CE 2
84
85 K1 = rq(P{1}*T1)
86 K2 = rq(P{2}*T2)
87
88
89 %% CE 3 - Load and plot images
90 clear all
91 load('compEx3data.mat');
92 q1 = imread('cube1.JPG');
93 q2 = imread('cube2.JPG');
94
95 figure(7)
96 imagesc(q1)
97 hold on
98 plot(x{1}(1,:), x{1}(2,:), '*')
99 axis equal
100 hold off
101
102 figure(8)
103 imagesc(q2)
104 hold on
105 plot(x{2}(1,:), x{2}(2,:), '*')
106 axis equal
107 hold off
108
109 %% Compute xTilde and N for figure 1
110 mean1 = mean(x{1}(1:2,:),2);
111 std1 = std(x{1}(1:2,:),0,2);
112
113 N1 = [1/std1(1) 0 -mean1(1)/std1(1);
114       0 1/std1(2) -mean1(2)/std1(2);
115       0 0 1];
116 x1Tilde = N1*x{1};
117
118 %% Plot fig 1 osv
119 figure(9)
120 plot(x1Tilde(1,:), x1Tilde(2,:), '*')
121 hold on
122 plot([ x1Tilde(1,startind ); x1Tilde(1,endind )],...
123      [x1Tilde(2,startind ); x1Tilde(2,endind )],...
124      'b-');
125 hold off

```

```

126 axis equal
127
128 figure(10)
129 plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
130 hold on
131 plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
132       [Xmodel(2,startind ); Xmodel(2,endind )],...
133       [Xmodel(3,startind ); Xmodel(3,endind)], 'b-');
134 hold off
135 axis equal
136
137 %% Assemble M-matrix and compute v, norm(M*v) etc.
138
139 X=[Xmodel;ones(1, 37)];
140 M1 = [];
141
142 for i = 1:18
143     M1(i*3 -2, 1:4) = X(1:4,i)';
144     M1(i*3 -1, 5:8) = X(1:4,i)';
145     M1(i*3 , 9:12) = X(1:4,i)';
146     M1(i*3 -2:i*3, i+12) = -x1Tilde(1:3, i);
147 end
148
149 [U,S,V] = svd(M1);
150 v1 = V(:,end)
151 min(diag(S))
152 norm(M1*v1) %same regardless of which column?
153
154 P1Tilde = [v1(1:4)';v1(5:8)';v1(9:12)']
155 P1 = N1\P1Tilde
156
157 x1 = P1*X;
158 x1flat = pflat(x1);
159
160 %%
161 figure(7)
162 imagesc(q1)
163 hold on
164 plot(x{1}(1,:), x{1}(2,:), '*')
165 plot(x1flat(1,:),x1flat(2,:), 'y*')
166 axis equal
167 hold off
168

```

```

169 %% Compute xTilde and N for figure
170 mean2 = mean(x{2}(1:2,:),2);
171 std2 = std(x{2}(1:2,:),0,2);
172
173 N2 = [1/std2(1) 0 -mean2(1)/std1(1);
174       0 1/std2(2) -mean2(2)/std1(2);
175       0 0 1 ];
176
177 x2Tilde = N2*x{2};
178
179 %% Plot
180 figure(10)
181 plot(x2Tilde(1,:), x2Tilde(2,:), '*')
182 hold on
183 plot([ x2Tilde(1,startind ); x2Tilde(1,endind )],...
184      [x2Tilde(2,startind ); x2Tilde(2,endind )],...
185      'b-');
186 hold off
187 axis equal
188
189 figure(11)
190 plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
191 hold on
192 plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
193      [Xmodel(2,startind ); Xmodel(2,endind )],...
194      [Xmodel(3,startind ); Xmodel(3,endind)], 'b-');
195 hold off
196 axis equal
197
198 %% Assemble M-matrix and compute v, norm(M*v) etc.
199
200 X=[Xmodel;ones(1, 37)];
201 M2 = [];
202
203 for i = 1:18
204     M2(i*3 -2, 1:4) = X(1:4,i)';
205     M2(i*3 -1, 5:8) = X(1:4,i)';
206     M2(i*3 , 9:12) = X(1:4,i)';
207     M2(i*3 -2:i*3, i+12) = -x2Tilde(1:3, i);
208 end
209
210 [U,S,V] = svd(M2);
211 v2 = V(:,end);

```

```

212 min(diag(S))
213 norm(M2*v2) %same regardless of which column?
214
215 P2Tilde = [v2(1:4)';v2(5:8)';v2(9:12)']
216 P2 = N2\P2Tilde
217
218 x2 = P2*X;
219 x2flat = pflat(x2);
220
221 %%
222 figure(12)
223 imagesc(q1)
224 hold on
225 plot(x{1}(1,:), x{1}(2,:), '*')
226 plot(x1flat(1,:),x1flat(2,:), 'y*')
227 axis equal
228 hold off
229
230 %%
231 cameras = {P1,P2};
232
233 figure(13)
234 plot3(Xmodel(1,:), Xmodel(2,:), Xmodel(3,:), '*')
235 hold on
236 plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
237       [Xmodel(2,startind ); Xmodel(2,endind )],...
238       [Xmodel(3,startind ); Xmodel(3,endind)], 'b-');
239 plotcams(cameras);
240 hold off
241 set(gca, 'Zdir', 'reverse')
242 axis equal
243
244 %%
245
246 [R1,Q1] = rq(P1);
247 [R2,Q2] = rq(P2);
248 K1ce3 = R1./R1(3,3)
249 K2ce3 = R2./R2(3,3)
250
251 %% CE 4
252
253 q1 = imread('cube1.jpg');
254 q2 = imread('cube2.jpg');

```

```

255 run vl_setup.m
256
257 [f1, d1] = vl_sift( single(rgb2gray(q1)), 'PeakThresh', 1);
258 [f2, d2] = vl_sift( single(rgb2gray(q2)), 'PeakThresh', 1);
259
260 figure(14)
261 imagesc(q1);
262 hold on
263 vl_plotframe(f1);
264 hold off
265 axis equal
266
267 figure(15)
268 imagesc(q2);
269 vl_plotframe(f2);
270 hold off
271 axis equal
272
273
274 [matches ,scores] = vl_ubcmatch(d1,d2);
275
276 x1 = [f1(1,matches (1 ,:));f1(2,matches (1 ,:))];
277 x2 = [f2(1,matches (2 ,:));f2(2,matches (2 ,:))];
278
279 perm = randperm(size(matches ,2));
280 figure(16);
281 imagesc ([q1 q2]);
282 hold on;
283 plot([x1(1,perm (1:10)); x2(1,perm (1:10))+ size(q1 ,2)], ...
284      [x1(2,perm (1:10)); x2(2,perm (1:10))] , '-');
285
286 hold off;
287
288 %% Ce 5 - Set up and solves DLT for trianhylation.
289
290 Xtriag = [];
291
292 for i = 1:length(x1)
293     Mce5 = [cameras{1} -[x1(:,i);1]      zeros(3,1) ;
294            cameras{2} zeros(3,1)      -[x2(:, i);1]];
295     [U,S,V] = svd(Mce5);
296     v = V(:,end);
297     Xtriag = [Xtriag v(1:4)];

```

```

298 end
299
300 %% Project triangulated points onto images.
301 Xflat = pflat(Xtriag);
302 xproj1 = pflat(cameras{1}*Xflat);
303 xproj2 = pflat(cameras{2}*Xflat);
304
305
306
307 perm = randperm(size(matches ,2));
308 figure(16)
309 % imagesc ([q1 q2]);
310 hold on;
311 plot([xproj1(1,perm (1:10)); xproj2(1,perm (1:10))+ size(q1 ,2)], ...
312      [xproj1(2,perm (1:10)); xproj2(2,perm (1:10))] , '--' ) ;
313 hold off;
314
315 good_points = (sqrt(sum((x1-xproj1 (1:2 ,:)).^2)) < 3 & ...
316               sqrt(sum((x2-xproj2 (1:2 ,:)).^2)) < 3);
317
318
319 Xgood = Xflat(:,good_points);
320
321
322 %%
323 figure(17)
324 plot3(Xgood(1,:), Xgood(2,:), Xgood(3,:), 'r')
325 hold on
326 plot3([ Xmodel(1,startind ); Xmodel(1,endind )],...
327      [Xmodel(2,startind ); Xmodel(2,endind )],...
328      [Xmodel(3,startind ); Xmodel(3,endind)], 'b-');
329 plotcams(cameras);
330 hold off
331 axis equal
332 %%
333
334 p1n = rq(P1)\P1;
335 p2n = rq(P2)\P2;

```