Lund University

Computer Vision

FMAN95

Assignment 4

Nils Broman

12/03/2021

Contents

2 Robust Homography Estimation and Stitching

2.1 E1

By looking at some arbitrary point which has a projection in both cameras we have

$$x_1 = P_1\mathbb{X} = \begin{bmatrix} A_1 & t_1 \end{bmatrix}\mathbb{X} = A_1X + t \qquad \Rightarrow \qquad X = A_1^{-1}(x_1 - t_1) \tag{1}$$

$$x_2 = \cdots = A_2X + t \qquad \Rightarrow \qquad X = A_2^{-2}(x_2 - t_2) \tag{2}$$

so

$$A_1^{-1}(x_1 - t_1) = A_2^{-2}(x_2 - t_2) \qquad \Rightarrow \qquad x_2 = A_2A_1^{-1}(x_1 - t_1) + t_2 \tag{3}$$

and with the same center, i.e. $t_1 = t_2 = 0$, we finally get

$$x_2 = A_2A_1^{-1}x_1 \qquad \Rightarrow \qquad H = A_2A_1^{-1} \tag{4}$$

2.2 E2

3x3 matrix, scale arbitrary $\Rightarrow$ 8 degrees of freedom.

3 equations from each point, one extra unknown λ per equation, so $3N$ equations and $8 + N$ unknowns $\Rightarrow$ $N \geq 4$ points

10% incorrect correspondence and 98% probability of outlier free sample set, so

$$P[Inliers] = 0.9 \Rightarrow P[n\ set\ outliers] = (1 - 0.9^4)^n \leq (1 - 0.98) \Rightarrow$$

$$n\log(1 - 0.9^4) \leq log(0.02) \Rightarrow n \geq \frac{\log(0.02)}{\log(1 - 0.9^4)} \approx 3.66 \Rightarrow \text{ 4 iterations}$$

2.3 CE1

947 SIFT features for image A and 865 for B, 204 matches and 151 inliers.

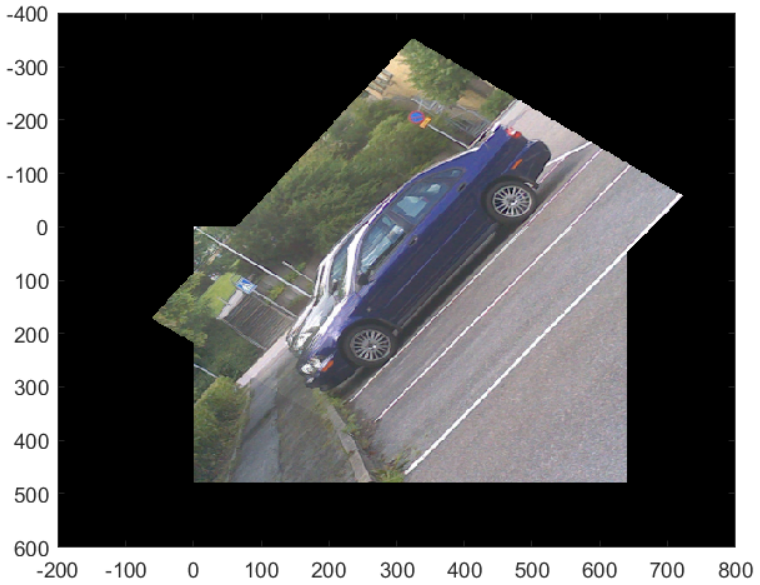

Figure 1: Panorama

3 Robust Essential Matrix Estimation

3.1 E3

R has 3 dofs, t has 3 dofs, scale arbitrary $\Rightarrow$ E has 5 dofs. 8 points in algorithm and with the same restrictions as in E2 we need

$$n \geq \frac{\log(0.02)}{\log(1 - 0.9^8)} \approx 6.95 \Rightarrow 7 \text{ iterations}$$

3.2 CE2

Number of inliers : 1466 RMS(Im1) = 0.5666 RMS(Im2) = 0.6083

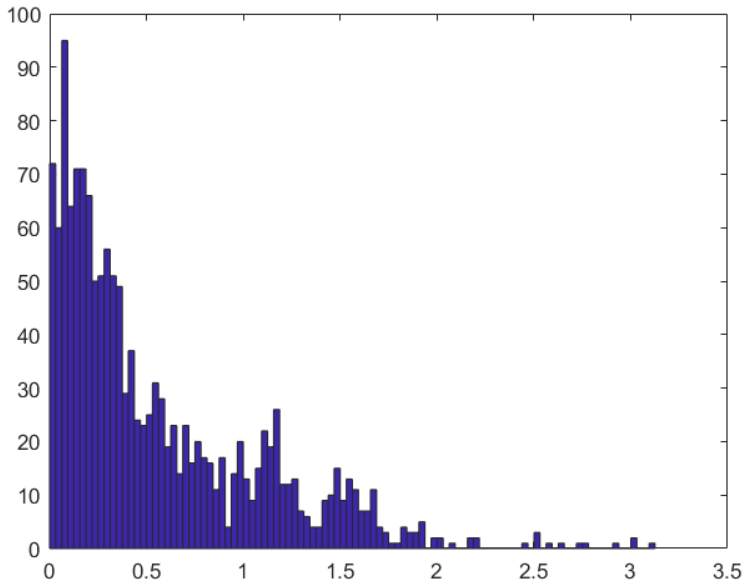

Figure 2: Reprojection errors Image 1

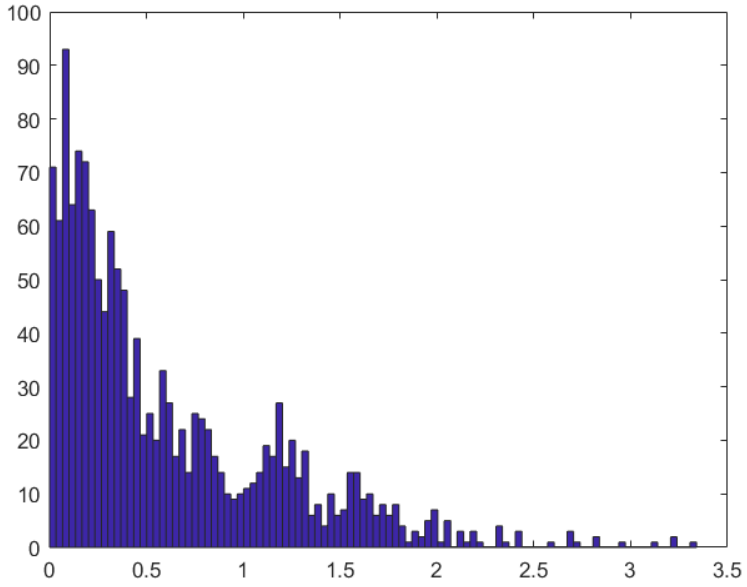

Figure 3: Reprojection errors Image 2

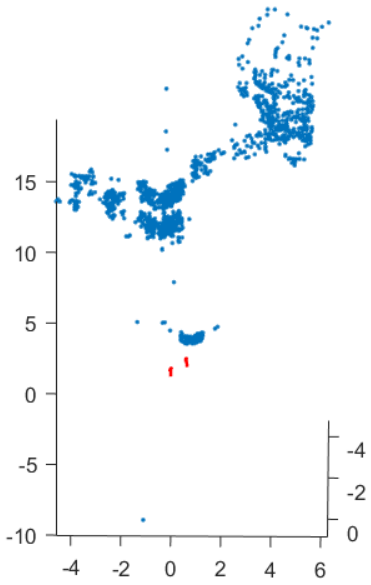

Figure 4: Reconstruction

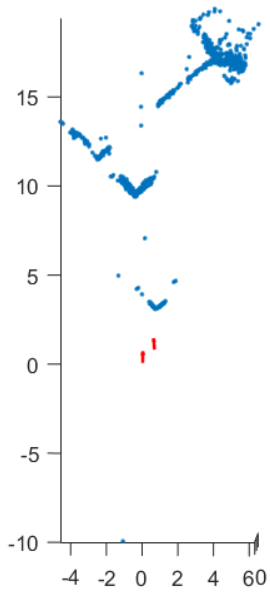

Figure 5: Reconstruction as seen from above

4 Calibrated Structure form Motion an Local Optimization

4.1 CE3

$$\text{RMS(Im1)} = 0.5666 \qquad \text{RMS(Im2)} = 0.5318 \tag{5}$$

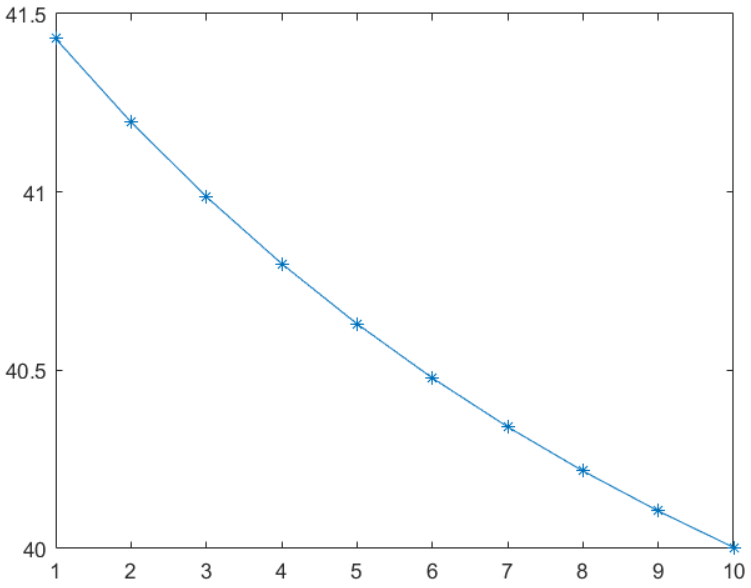

Figure 6: Objective value for 10 iterations

4.2 CE4

$$\text{RMS(Im1)} = 0.1725 \qquad \text{RMS(Im2)} = 0.1582 \tag{6}$$

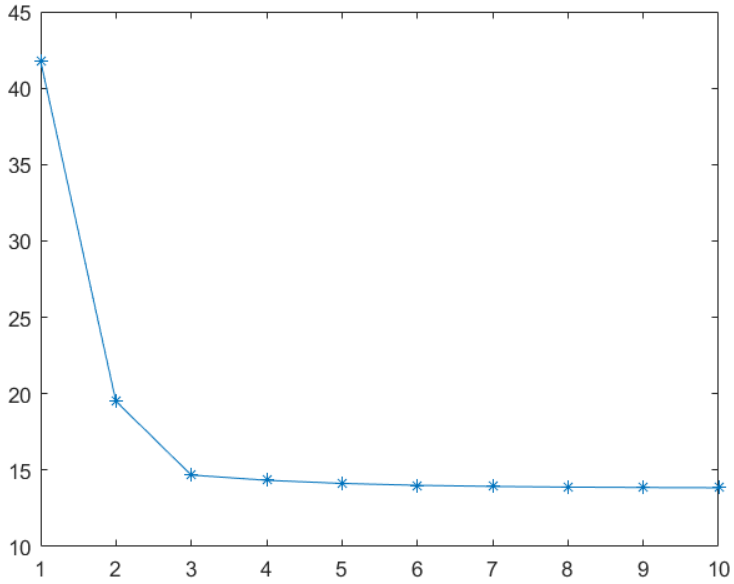

Figure 7: Objective value for 10 iterations, clearly a lot better than the method in CE3

5 Appendix

Listing 1: main.m

```
%% CE1
% load images and setup vl
im_A = imread('a.jpg');
im_B = imread('b.jpg');
% im_A = imread('nils1.jpg');
% im_B = imread('nils2.jpg');
% im_A = imread('tavla1.jpg');
% im_B = imread('tavla2.jpg');

run vl_setup.m;

%% compute sift shit

[fA dA] = vl_sift( single(rgb2gray(im_A)) );
[fB dB] = vl_sift( single(rgb2gray(im_B)) );

%% plots
figure(1)
imagesc(im_A);
hold on
vl_plotframe(fA);
hold off
axis equal

figure(2)
imagesc(im_B);
vl_plotframe(fB);
hold off
axis equal

%% compute matches
matches = vl_ubcmatch(dA,dB);

% xA = fA(1:2, matches(1,:));
% xB = fB(1:2, matches(1,:));

xA = [fA(1,matches (1 ,:)); fA(2,matches (1 ,:))];
xB = [fB(1,matches (2 ,:)); fB(2,matches (2 ,:))];

```

```
perm = randperm(size(matches ,2));

%% plot lines

figure(3);
imagesc ([im_A im_B]);
hold on;
plot([xA(1,perm (1:10)); xB(1,perm (1:10))+ size(im_A ,2)], ...
    [xA(2,perm (1:10));  xB(2,perm (1:10))] ,'-') ;
axis equal
hold  off;


%% compute H
cp = [];
for i=1:5000
    % Pick 4 random points
    rand = randperm(length(xA), 4);
    x = [xA(:,rand); ones(1,4)];
    y = [xB(:,rand); ones(1,4)];

    % Compute estimated H
    Me = zeros(12, 13);
    for j = 1:length(x)
        Me(3*j-2:3*j,:) = ...
         [x(:,j)' zeros(1,6) zeros(1,j-1) y(1,j) zeros(1,4-j);
         zeros(1,3) x(:,j)' zeros(1,3) zeros(1,j-1) y(2,j) zeros(1,4-j);
         zeros(1,6) x(:,j)' zeros(1,j-1) y(3,j) zeros(1,4-j)];
    end

    [U,S,V] = svd(Me);
    v = V(1:9, end);
    He = reshape(v, [3 3])';

    % Compute estimated points and find those with error < 5 pixels
    xBe = He*[xA ; ones(1, (length(xA)))];
    xBeFlat = pflat(xBe);
    err = sqrt((xB(1,:)-xBeFlat(1,:)).^2 + (xB(2,:)-xBeFlat(2,:)).^2);
    cps = find(err<5);

    % Save the best estimate of the loop
    if (length(cps) > length(cp))
        cp = cps;
```

```
        Hest = He;
    end

end

Hest = Hest./Hest(end, end);

length(cp)

%%

% Transform image to comon coordinate system
Htform = projective2d(Hest');
Rout = imref2d ( size(im_A) ,[ -200 800] ,[ -400 600]);
% Rout = imref2d ( size(im_A) ,[ -1500 3800] ,[ -1400 4800]);
[Atransf] = imwarp(im_A,Htform ,'OutputView',Rout );
Idtform = projective2d(eye(3));
[Btransf] = imwarp(im_B, Idtform ,'OutputView',Rout );
AB = Btransf;
AB( Btransf < Atransf ) = Atransf( Btransf < Atransf );
figure(4)
imagesc( Rout . XWorldLimits , Rout . YWorldLimits ,AB );









%% Ce2

load("compEx2data.mat");
im1 =  imread("im1.jpg");
im2 =  imread("im2.jpg");
x1 = [x{1}; ones(1, length(x{1}))];
x2 = [x{2}; ones(1, length(x{1}))];

cp = [];
l1 = [];
l2 = [];
E = [];
F = [];

```

```
126 iter = 1000;
127 for i = 1:iter
128     % get five random points
129     randsel = randperm(length(x1), 5);
130     x1r = x1(:, randsel);
131     x2r = x2(:, randsel);
132 
133     % Calibrate with K
134     x1rn = K\x1r;
135     x2rn = K\x2r;
136 
137     % Calculate essential matrix estimation
138     Ei = fivepoint_solver(x1rn, x2rn);
139 
140     for j = 1:length(Ei)
141 
142         %compute F
143         Fe = K'\Ei{j}/K;
144         Fj = Fe./Fe(3,3);
145 
146         % compute lines
147         l1j = Fj'*x2;
148         l1j = l1j./sqrt(repmat(l1j(1,:).^2 + l1j(2,:).^2, [3 1])); %
               normalize
149 
150         l2j = Fj*x1;
151         l2j = l2j./sqrt(repmat(l2j(1,:).^2 + l2j(2,:).^2, [3 1])); %
               normalize
152 
153         % compute distances to lines
154         dist1 = abs(sum(l1j.*x1));
155         dist2 = abs(sum(l2j.*x2));
156 
157         % compute inliers
158         cp1 = find(dist1 < 5);
159         cp2 = find(dist2 < 5);
160 
161         % compute matching inliers for both images
162         [cpj usch] = intersect(cp1, cp2);
163 
164         % save the best results
165         if (length(cp) < length(cpj))
166             cp = cpj;
```

```
167             l1 = l1j;
168             l2 = l2j;
169             E = Ei{j};
170             F = Fj;
171         end
172     end
173 end
174 
175 disp('number of matching inliers:')
176 length(cp)
177 
178 %% Plot
179 
180 figure(5)
181 imagesc(im1)
182 hold on
183 plot(x1(1,cp), x1(2,cp), 'y*', 'Markersize', 10)
184 rital(l1(:,cp))
185 hold off
186 axis equal
187 
188 figure(6)
189 imagesc(im2)
190 hold on
191 plot(x2(1,cp), x2(2,cp), 'y*', 'Markersize', 10)
192 rital(l2(:,cp))
193 hold off
194 axis equal
195 
196 %% compute cameras and triangulate
197 
198 x1n = K\x1;
199 x2n = K\x2;
200 
201 [U S V] = svd(E);
202 
203 u3 = U(:,end);
204 W = [0 -1  0 ; 1 0 0 ; 0 0 1];
205 
206 P1 = [eye(3) [0;0;0]];
207 % CAmera solutions
208 P2 = {[U*W*V' u3], [U*W*V' -u3], [U*W'*V' u3], [U*W'*V' -u3]};
209 
```

```
in_front = 0;
bestP2 = 0;
bestX = 0;

for i=1:4

    % Triangulate
    X = [];
    for j = 1:length(x1n)
        Me = [P1, x1n(:,j) zeros(3,1) ; ...
              P2{i} zeros(3,1) x2n(:,j)];
        [Ut, St, Vt] = svd(Me);
        X(:,j) = pflat(Vt(1:4,end));
    end

    % Check number of points in front of camera
    temp = sum(P1(3,:)*X > 0 & P2{i}(3,:)*X > 0);
    if  temp >= in_front
        bestP2 = P2{i};
        bestX = X;
        in_front = temp;
        i
    end
end

P2f = K*bestP2;
X = bestX;
x1proj = pflat(K*P1*X);
x2proj = pflat(P2f*X);

%% plots projections of 3d and 3d model with cameras'

figure(7)
imagesc(im1)
hold on
plot(x1(1,cp), x1(2,cp), 'r*', 'Markersize', 5)
plot(x1proj(1,cp), x1proj(2,cp), 'b+', 'Markersize', 5)
hold off
axis equal

figure(8)
imagesc(im2)
hold on
```

```
plot(x2(1,:), x2(2,:), 'r*', 'Markersize', 5)
plot(x2proj(1,:), x2proj(2,:), 'b+', 'Markersize', 5)
hold off
axis equal
%%
figure(22)
plot3(X(1,cp), X(2,cp), X(3,cp), '.')
hold on
plotcams({P1, P2f})
hold off
axis equal


%% Plot histograms of errors

figure(9)
hist(sum(abs(x1(:,cp)-x1proj(:,cp))), 100);

figure(10)
hist(sum(abs(x2(:,cp)-x2proj(:,cp))), 100);

%% Compute rms errors
x1rms = mean(sqrt(sum((x1(:,cp)-x1proj(:,cp)).^2)))
x2rms = mean(sqrt(sum((x2(:,cp)-x2proj(:,cp)).^2)))



%% CE3

P = {K*P1 P2f};
U = X(:,cp);
u = {x1(:, cp) x2(:, cp)};

iter = 10;
gammak = 1;
rnorms = zeros(1, 10);
[r,J] = LinearizeReprojErr(P,U,u);
for i = 1:iter
    lastr = r;
    while 1
        deltav = -gammak*J'*r;
        [Pnew , Unew ] = update_solution(deltav ,P,U);
        [r,J] = LinearizeReprojErr(Pnew,Unew,u);
```

```
296         if norm(r) >= norm(lastr)
297             gammak = .5* gammak;
298         else
299             P = Pnew;
300             U = Unew;
301             break;
302         end
303     end
304     rnorms(i) = norm(r);
305 end
306 %%
307 figure(11)
308 plot(linspace(1, iter, iter), rnorms, '*-')
309 
310 Pnew{2} = Pnew{2}./Pnew{2}(end, end);
311 x1e = pflat(Pnew{1}*Unew);
312 x2e = pflat(Pnew{2}*Unew);
313 
314 
315 figure(12)
316 imagesc(im1)
317 hold on
318 plot(x1(1,:), x1(2,:), 'b*', 'Markersize', 5)
319 plot(x1e(1,:), x1e(2,:), 'r.', 'Markersize', 5)
320 hold off
321 axis equal
322 
323 figure(13)
324 imagesc(im2)
325 hold on
326 plot(x2(1,:), x2(2,:), 'b*', 'Markersize', 5)
327 plot(x2e(1,:), x2e(2,:), 'r.', 'Markersize', 5)
328 hold off
329 axis equal
330 
331 % RMS error
332 disp('RMS x1')
333 mean(sqrt(sum((x1(:, cp)-x1e).^2)))
334 disp('RMS x2')
335 mean(sqrt(sum((x2(:, cp)-x2e).^2)))
336 
337 %% CE4
338 P = {K*P1 P2f};
```

```
U = X(:,cp);
u = {x1(:, cp) x2(:, cp)};

iter = 10;
lambda = 1;
rnorms = zeros(1, 10);
[r,J] = LinearizeReprojErr(P,U,u);
C = J'*J+lambda*speye(size(J ,2));
c = J'*r;
deltav = -C\c;
[Pnew , Unew ] = update_solution(deltav ,P,U);
for i = 1:iter
    lastr = r;
    C = J'*J+lambda*speye(size(J ,2));
    c = J'*r;
    deltav = -C\c;
    [Pnew , Unew ] = update_solution(deltav ,Pnew,Unew);
    [r,J] = LinearizeReprojErr(Pnew,Unew,u);
    rnorms(i) = norm(r);
end

%% plots

figure(14)
plot(linspace(1, iter, iter), rnorms,  '*-')
%%
x1e = pflat(Pnew{1}*Unew);
x2e = pflat(Pnew{2}*Unew);

figure(15)
imagesc(im1)
hold on
plot(x1(1,:), x1(2,:), 'b*', 'Markersize', 5)
plot(x1e(1,:), x1e(2,:), 'r.', 'Markersize', 5)
hold off
axis equal

figure(16)
imagesc(im2)
hold on
plot(x2(1,:), x2(2,:), 'b*', 'Markersize', 5)
plot(x2e(1,:), x2e(2,:), 'r.', 'Markersize', 5)
hold off
```

```
382 axis equal
383 %%
384 % RMS error
385 disp('RMS x1')
386 mean(sqrt(sum((x1(:, cp)-x1e).^2)))
387 disp('RMS x2')
388 mean(sqrt(sum((x2(:, cp)-x2e).^2)))
```