

March 30, 2018

HOMEWORK 1

Problem 1. Line-plane intersection

The line in 3D defined by the join of the points $\mathbf{X}_1 = (X_1, Y_1, Z_1, T_1)^T$ and $\mathbf{X}_2 = (X_2, Y_2, Z_2, T_2)^T$ can be represented as a Plücker matrix $L = \mathbf{X}_1 \mathbf{X}_2^T - \mathbf{X}_2 \mathbf{X}_1^T$ or pencil of points $\mathbf{X}(\lambda) = \lambda \mathbf{X}_1 + (1 - \lambda) \mathbf{X}_2$. The line intersects the plane $\pi = (a, b, c, d)^T$ at the point $\mathbf{X}_L = L\pi$ or $\mathbf{X}(\lambda_\pi)$, where λ_π is determined such that $\mathbf{X}(\lambda_\pi)^T \pi = 0$. Show that \mathbf{X}_L is equal to $\mathbf{X}(\lambda_\pi)$ up to scale.

Solution

Since $\mathbf{X}_L = L\pi$, then

$$\mathbf{X}_L = (\mathbf{X}_1 \mathbf{X}_2^T - \mathbf{X}_2 \mathbf{X}_1^T) \pi = \begin{bmatrix} b(X_1 Y_2 - X_2 Y_1) & + & c(X_1 Z_2 - X_2 Z_1) & + & d(T_2 X_1 - T_1 X_2) \\ a(X_2 Y_1 - X_1 Y_2) & + & c(T_1 Z_2 - Y_2 Z_1) & + & d(T_2 Y_1 - T_1 Y_2) \\ a(X_2 Z_1 - X_1 Z_2) & + & b(Y_2 Z_1 - Y_1 Z_2) & + & d(T_2 Z_1 - T_1 Z_2) \\ a(T_1 X_2 - T_2 X_1) & + & b(T_1 T_2 - T_2 Y_1) & + & c(T_1 Z_2 - T_2 Z_1) \end{bmatrix} \pi.$$

For $\mathbf{X}(\lambda_\pi)$, from $\mathbf{X}(\lambda_\pi)^T \pi = 0$, we have

$$\lambda_\pi = -\frac{aX_2 + bY_2 + cZ_2 + dT_2}{a(X_1 - X_2) + b(Y_1 - Y_2) + c(Z_1 - Z_2) + d(T_1 - T_2)}.$$

Let $m = a(X_1 - X_2) + b(Y_1 - Y_2) + c(Z_1 - Z_2) + d(T_1 - T_2)$, then

$$\mathbf{X}(\lambda_\pi) = -\frac{1}{m} \begin{bmatrix} b(X_1 Y_2 - X_2 Y_1) & + & c(X_1 Z_2 - X_2 Z_1) & + & d(T_2 X_1 - T_1 X_2) \\ a(X_2 Y_1 - X_1 Y_2) & + & c(T_1 Z_2 - Y_2 Z_1) & + & d(T_2 Y_1 - T_1 Y_2) \\ a(X_2 Z_1 - X_1 Z_2) & + & b(Y_2 Z_1 - Y_1 Z_2) & + & d(T_2 Z_1 - T_1 Z_2) \\ a(T_1 X_2 - T_2 X_1) & + & b(T_1 T_2 - T_2 Y_1) & + & c(T_1 Z_2 - T_2 Z_1) \end{bmatrix} \pi.$$

So, $\mathbf{X}_L = \mathbf{X}(\lambda_\pi)$.

Problem 2. Line-quadric intersection

If the pencil of points $\mathbf{X}(\lambda) = \lambda \mathbf{X}_1 + (1 - \lambda) \mathbf{X}_2$ represents a line in 3D, the (up to two) real roots of the quadratic polynomial $c_2 \lambda_Q^2 + c_1 \lambda_Q + c_0 = 0$ are used to solve for the intersection point(s) $\mathbf{X}(\lambda_Q)$. Show that $c_2 = \mathbf{X}_1^T Q \mathbf{X}_1 - 2 \mathbf{X}_1^T Q \mathbf{X}_2 + \mathbf{X}_2^T Q \mathbf{X}_2$, $c_1 = 2(\mathbf{X}_1^T Q \mathbf{X}_2 - \mathbf{X}_2^T Q \mathbf{X}_2)$, and $c_0 = \mathbf{X}_2^T Q \mathbf{X}_2$.

Solution

A 3D point \mathbf{X} is on the quadric Q if and only if $\mathbf{X}^T Q \mathbf{X} = 0$, then

$$(\lambda \mathbf{X}_1 + (1 - \lambda) \mathbf{X}_2)^T Q (\lambda \mathbf{X}_1 + (1 - \lambda) \mathbf{X}_2) = 0 \quad (1)$$

$$\lambda^2 \mathbf{X}_1^T Q \mathbf{X}_1 + \lambda(1 - \lambda) \mathbf{X}_1^T Q \mathbf{X}_2 + \lambda(1 - \lambda) \mathbf{X}_2^T Q \mathbf{X}_1 + (1 - \lambda)^2 \mathbf{X}_2^T Q \mathbf{X}_2 = 0 \quad (2)$$

Since the transpose of a scalar is itself, from equation(0-1), we have

$$(\mathbf{X}_1^T Q \mathbf{X}_1 - 2\mathbf{X}_1^T Q \mathbf{X}_2 + \mathbf{X}_2^T Q \mathbf{X}_2)\lambda^2 + 2(\mathbf{X}_1^T Q \mathbf{X}_2 - \mathbf{X}_2^T Q \mathbf{X}_2)\lambda + \mathbf{X}_2^T Q \mathbf{X}_2 = 0 \quad (3)$$

So, $c_2 = \mathbf{X}_1^T Q \mathbf{X}_1 - 2\mathbf{X}_1^T Q \mathbf{X}_2 + \mathbf{X}_2^T Q \mathbf{X}_2$, $c_1 = 2(\mathbf{X}_1^T Q \mathbf{X}_2 - \mathbf{X}_2^T Q \mathbf{X}_2)$, and $c_0 = \mathbf{X}_2^T Q \mathbf{X}_2$.

Problem 3. Programming: Automatic feature detection and matching



Figure 1: (a) shows the image *price_center20.JPG*, (b) shows the image *price_center21.JPG*.

(a) Feature detection

The size of the feature detection window is 11×11 , the minor eigenvalue threshold value is 5, the size of the local non-maximum suppression window is 9×9 .

The number of features detected in *price_center20.JPG* is 616, and the number of features detected in *price_center21.JPG* is 632.

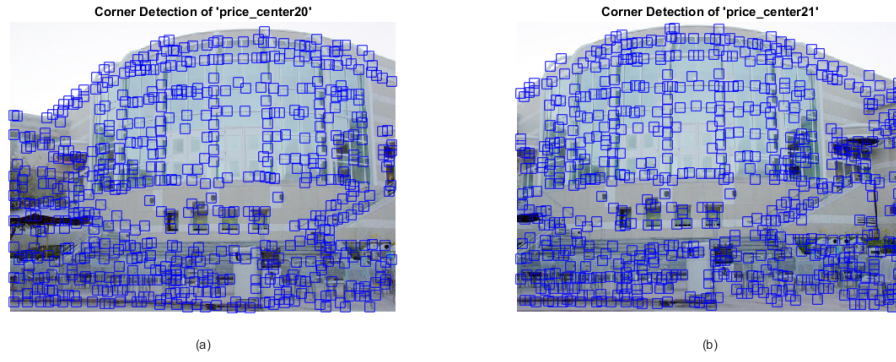


Figure 2: (a) shows detected corners(after local nonmaximum suppression) of the image *price_center20.JPG*, (b) shows detected corners(after local nonmaximum suppression) of the image *price_center21.JPG*.

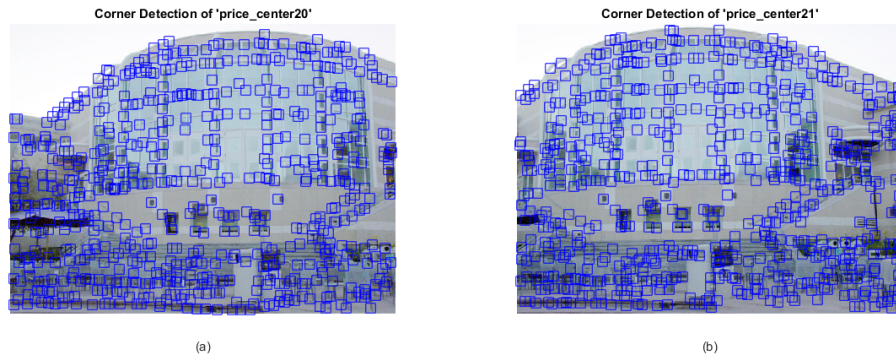


Figure 3: (a) shows detected corners(about the subpixel) of the image *price_center20.JPG*, (b) shows detected corners(about the subpixel) of the image *price_center21.JPG*.

(b) Feature matching

The size of the proximity window is 20×500 , the correlation coefficient threshold value is 0.6, the

distance ratio threshold value is 0.9, and the resulting number of putative feature correspondences is 196.

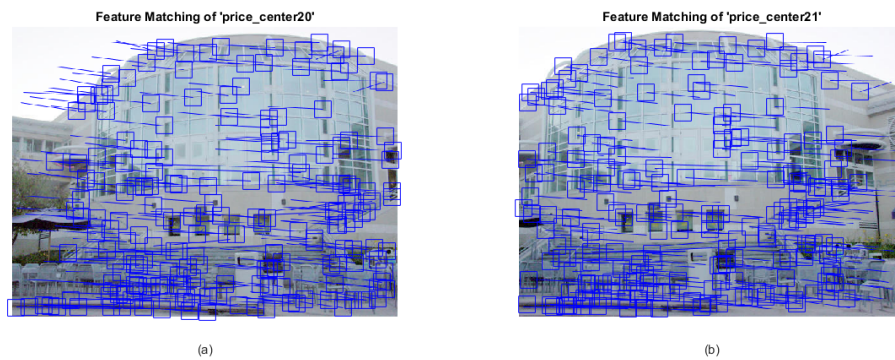


Figure 4: (a) shows matched features in *price_center20.JPG*, (b) shows matched features in *price_center21.JPG*.

Appendix

```

1  close all;
2  clear;
3  clc;
4  %% Corner Detection
5  win_detect = 11;
6  win_spr = 9;
7  eigen_th = 5;
8  I0 = imread('price_center20.JPG');
9  I1 = imread('price_center21.JPG');
10 [r0,c0,x_f0,y_f0] = CornerCoordinate(I0,win_detect,win_spr,eigen_th);
11 [r1,c1,x_f1,y_f1] = CornerCoordinate(I1,win_detect,win_spr,eigen_th);
12
13 figure(1)
14 subplot(1,2,1)
15 imshow(I0);
16 xlabel('(a)');
17 title('price\_center20');

```

```

18 subplot(1,2,2)
19 imshow(I1);
20 xlabel(' (b) ');
21 title('price\_center21');
22
23 figure(2)
24 subplot(1,2,1)
25 imshow(I0);hold on
26 plot(c0,r0,'bs','MarkerSize',win_detect);hold off
27 xlabel(' (a) ');
28 title('Corner\_Detection\_of\_price\_center20');
29 subplot(1,2,2)
30 imshow(I1);hold on
31 xlabel(' (b) ');
32 plot(c1,r1,'bs','MarkerSize',win_detect);hold off
33 title('Corner\_Detection\_of\_price\_center21');
34
35 figure(3)
36 subplot(1,2,1)
37 imshow(I0);hold on
38 plot(y_f0,x_f0,'bs','MarkerSize',win_detect);hold off
39 xlabel(' (a) ');
40 title('Corner\_Detection\_of\_price\_center20');
41 subplot(1,2,2)
42 imshow(I1);hold on
43 xlabel(' (b) ');
44 plot(y_f1,x_f1,'bs','MarkerSize',win_detect);hold off
45 title('Corner\_Detection\_of\_price\_center21');
46
47 %% Feature Matching
48 win_match = 19;
49 simi_th = 0.6;
50 dist_th = 0.9;
51 [X0,Y0,X1,Y1] = FeatureMatch(I0,I1,x_f0,y_f0,x_f1,y_f1,win_match,simi_th,dist_th);
52 figure(4)
53 subplot(1,2,1)
54 imshow(insertShape(I0,'Line',[Y0 X0 Y1 X1],'Color','blue'));
55 hold on
56 plot(Y0,X0,'bs','MarkerSize',win_match);
57 hold off
58 xlabel(' (a) ');
59 title('Feature\_Matching\_of\_price\_center20');
60 subplot(1,2,2)
61 imshow(insertShape(I1,'Line',[Y1 X1 Y0 X0],'Color','blue'));
62 hold on
63 plot(Y1,X1,'bs','MarkerSize',win_match);

```

```

64 hold off
65 xlabel(' (b) ');
66 title('Feature Matching of ' price \_center21 ');

1 function [r,c,x_f1,x_f2] = CornerCoordinate(im,win1,win2,threshold)
2 im = double(rgb2gray(im));
3 A = zeros(size(im));
4 k = [-1 8 0 -8 1]' / 12;
5 imx = imfilter(im,k,'symmetric');
6 imy = imfilter(im,k,'symmetric');
7 %% Gradient matrix
8 for i = (win1+1)/2 : size(im,1) - (win1-1)/2
9     for j = (win1+1)/2 : size(im,2) - (win1-1)/2
10         im_win_x = imx(i - (win1-1)/2:i + (win1-1)/2,...
11             j - (win1-1)/2:j + (win1-1)/2);
12         im_win_y = imy(i - (win1-1)/2:i + (win1-1)/2,...
13             j + (win1-1)/2:j + (win1-1)/2);
14
15         N = zeros(2,2);
16         N(1,1) = sum(sum(im_win_x.^2));
17         N(1,2) = sum(sum(im_win_x .* im_win_y));
18         N(2,1) = N(1,2);
19         N(2,2) = sum(sum(im_win_y.^2));
20         N = N/win1^2;
21         lambda = (trace(N) - sqrt(trace(N)^2 - 4*det(N)))/2;
22         if lambda > threshold
23             A(i,j) = lambda;
24         end
25     end
26 end
27 %% Non-maximum suppression
28 B = ordfilt2(A,win1^2,ones(win1,win1));
29 C = (B == A & B ~= 0);
30 [r,c] = find(C);
31 x_f1 = zeros(size(r));
32 x_f2 = zeros(size(r));
33 %% find corner
34 for k = 1:numel(r)
35     T = zeros(2,2);
36     y = zeros(2,1);
37     im_win_x = imx(r(k) - (win2-1)/2:r(k) + (win2-1)/2,...
38         c(k) - (win2-1)/2:c(k) + (win2-1)/2);
39     im_win_y = imy(r(k) - (win2-1)/2:r(k) + (win2-1)/2,...
40         c(k) - (win2-1)/2:c(k) + (win2-1)/2);
41     xx = [r(k) - (win2-1)/2 : r(k) + (win2-1)/2]' .* ones(win2);
42     yy = [c(k) - (win2-1)/2 : c(k) + (win2-1)/2] .* ones(win2);

```

```

43
44     y(1) = sum(sum( xx.*im_win_x.^2 + yy.*im_win_x.*im_win_y ));
45     y(2) = sum(sum( xx.*im_win_x.*im_win_y + yy.*im_win_y.^2 ));
46     T(1,1) = sum(sum(im_win_x.^2));
47     T(1,2) = sum(sum(im_win_x .* im_win_y));
48     T(2,1) = T(1,2);
49     T(2,2) = sum(sum(im_win_y.^2));
50     x = T\y;
51
52     x_f1(k) = x(1);
53     x_f2(k) = x(2);
54 end
55 end

1 function [X0,Y0,X1,Y1] = FeatureMatch(im0,im1,x0,y0,x1,y1,win,simi_th,dist_th)
2 im0 = double(rgb2gray(im0));
3 im1 = double(rgb2gray(im1));
4 half_win = (win-1)/2;
5 %%
6 % xx0 = round(x0)+half_win;
7 % xx1 = round(x1)+half_win;
8 % yy0 = round(y0)+half_win;
9 % yy1 = round(y1)+half_win;
10 xx0 = x0+half_win;
11 xx1 = x1+half_win;
12 yy0 = y0+half_win;
13 yy1 = y1+half_win;
14 im0 = padarray(im0,[half_win,half_win],'symmetric');
15 im1 = padarray(im1,[half_win,half_win],'symmetric');
16 %% Xcorrelation matrix
17 xc = zeros(numel(xx0),numel(xx1));
18 crd = zeros(numel(xx0),numel(xx1));
19 for i = 1:numel(xx0)
20     i
21     X = fix(xx0(i) - half_win:ceil(xx0(i)) + half_win;
22     Y = fix(yy0(i) - half_win:ceil(yy0(i)) + half_win;
23     im0_win = im0(X,Y);
24     [X,Y] = meshgrid(X,Y);
25     [Xq,Yq] = meshgrid(xx0(i)-half_win:xx0(i)+half_win,...
26                         yy0(i) - half_win:yy0(i) + half_win);
27     im0_win_intep = interp2(X,Y,im0_win,Xq,Yq,'linear');
28 %     im0_win = im0(xx0(i)-half_win : xx0(i)+half_win,yy0(i)-half_win : yy0(i)+half_win);
29     for j = 1:numel(xx1)
30         X = fix(xx1(j) - half_win:ceil(xx1(j)) + half_win;
31         Y = fix(yy1(j) - half_win:ceil(yy1(j)) + half_win;
32         im1_win = im1(X,Y);

```

```

33     [X,Y] = meshgrid(X,Y);
34     [Xq,Yq] = meshgrid(xx1(j)-half_win:xx1(j)+half_win,...
35                         yy1(j) - half_win:yy1(j) + half_win);
36     im1_win_intep = interp2(X,Y,im1_win,Xq,Yq,'linear');
37 %     im1_win = im1(xx1(j)-half_win : xx1(j)+half_win,yy1(j)-half_win : yy1(j)+half_win);
38     xc(i,j) = corr2(im0_win_intep,im1_win_intep);
39     end
40 end
41 %% One-to-One Matching
42 while max(xc(:)) > simi_th
43     [r,c] = find(xc == max(xc(:)));
44     i = r(1);
45     j = c(1);
46     mx = xc(i,j);
47     xc(i,j) = -1;
48     next_mx = max(max(xc(i,:),[],2),max(xc(:,j))));
49
50     if (1-mx) < (1-next_mx)*dist_th
51         crd(i,j) = 1;
52     end
53     xc(i,:) = -1;
54     xc(:,j) = -1;
55 end
56 [i,j] = find(crd);
57 X0 = x0(i);
58 Y0 = y0(i);
59 X1 = x1(j);
60 Y1 = y1(j);
61
62 w = abs(X0-X1);
63 d = abs(Y0-Y1);
64 n = find(w>20 | d >500);
65 X0(n) = [];
66 Y0(n) = [];
67 X1(n) = [];
68 Y1(n) = [];
69 end

```