COMPUTER VISION

7th work

2nd Semester Year 2016/17

Aurélien Bernier & Yoann Fleytoux

Motion estimation and 3D reconstruction using the

Tomasi-Kanade

1-Consider the Tomasi-Kanade algorithm, described in the theoretical classes (based on the article attachment).

2-Implement the Tomasi-Kanade algorithm using as normalization method the Method described in the lectures (article "A Sequential Factorization Method for Recovering Shape and Motion from Image Streams "- also attached);

4-Begin by testing the method on synthetic images composed only of dots. For this generates a sequence of images corresponding to a hexagonal prism. The movement of the prism corresponds to 3D rotations between the camera's coordinate system and the Prism (this coordinate system originates from the center of mass of the points). Consider As the model of the orthographic projection model. Use at least 4 images of the sequence. 3D reconstruction of the hexagonal prism and estimate its motion using the Tomasi-Kanade algorithm.

Note: You do not need to generate an image properly. Use only the coordinates (in the image) of the points corresponding to the vertices of the hexagonal prism. The points should be joined by segments of straight lines.

```
clc
clear all;
close all;

% Modelisation of the hexagonal prism, let's turn it and compute it
ogPrism =[0 0 0;1 0 0;1 1 0;0 1 0;...
    1.5 0 0.5; 1.5 1 0.5; 0 0 1;1 0 1;...
    1 1 1;0 1 1; -0.5 0 0.5; -0.5 1 0.5];


Rotate = [cos(pi/4) -sin(pi/4) 0;
          sin(pi/4) cos(pi/4) 0;
          0         0         1];

prism2 = ogPrism*Rotate;
prism3 = prism2*Rotate;
prism4 = prism3*Rotate;

prismPoints = [ogPrism(:,1) ogPrism(:,2), ogPrism(:,3);...
    prism2(:,1) prism2(:,2), prism2(:,3);...
    prism3(:,1) prism3(:,2), prism3(:,3);...
    prism4(:,1) prism4(:,2), prism4(:,3)];

allX = prismPoints(:,1);
allY = prismPoints(:,2);
allZ = prismPoints(:,3);
```

```
W = [allZ(1:12)';allZ(13:24)';allZ(25:36)';allZ(37:48)';...
     allY(1:12)';allY(13:24)';allY(25:36)';allY(37:48)'];

for i=1:4
   meanZ{i} = mean(W(i,:));
   meanY{i} = mean(W(i+4,:));

   W(i,:) = W(i,:)-meanZ{i};
   W(i+4,:) = W(i+4,:)-meanY{i};
end

%Single value decomposition & Fill L
[U D V] = svd(W);
Ucut = [U(:,1) U(:,2) U(:,3)];
Dcut = [D(1,1:3);D(2,1:3);D(3,1:3)];
Vcut = [V(:,1) V(:,2) V(:,3)];

RT = Ucut*Dcut^(0.5);
STemp = Dcut^(0.5)*Vcut';

A = [RT(1,1)*RT(1,1) 2*RT(1,1)*RT(1,2) 2*RT(1,1)*RT(1,3)...
     RT(1,2)*RT(1,2) 2*RT(1,2)*RT(1,3) RT(1,3)*RT(1,3);
     RT(2,1)*RT(2,1) 2*RT(2,1)*RT(2,2) 2*RT(2,1)*RT(2,3)...
     RT(2,2)*RT(2,2) 2*RT(2,2)*RT(2,3) RT(2,3)*RT(2,3);
     RT(3,1)*RT(3,1) 2*RT(3,1)*RT(3,2) 2*RT(3,1)*RT(3,3)...
     RT(3,2)*RT(3,2) 2*RT(3,2)*RT(3,3) RT(3,3)*RT(3,3);
     RT(4,1)*RT(4,1) 2*RT(4,1)*RT(4,2) 2*RT(4,1)*RT(4,3)...
     RT(4,2)*RT(4,2) 2*RT(4,2)*RT(4,3) RT(4,3)*RT(4,3);
     RT(5,1)*RT(5,1) 2*RT(5,1)*RT(5,2) 2*RT(5,1)*RT(5,3)...
     RT(5,2)*RT(5,2) 2*RT(5,2)*RT(5,3) RT(5,3)*RT(5,3);
     RT(6,1)*RT(6,1) 2*RT(6,1)*RT(6,2) 2*RT(6,1)*RT(6,3)...
     RT(6,2)*RT(6,2) 2*RT(6,2)*RT(6,3) RT(6,3)*RT(6,3);
     RT(1,1)*RT(4,1) 2*RT(1,1)*RT(4,2) 2*RT(1,1)*RT(4,3)...
     RT(1,2)*RT(4,2) 2*RT(1,2)*RT(4,3) RT(1,3)*RT(4,3);
     RT(2,1)*RT(5,1) 2*RT(2,1)*RT(5,2) 2*RT(2,1)*RT(5,3)...
     RT(2,2)*RT(5,2) 2*RT(2,2)*RT(5,3) RT(2,3)*RT(5,3);
     RT(3,1)*RT(6,1) 2*RT(3,1)*RT(6,2) 2*RT(3,1)*RT(6,3)...
     RT(3,2)*RT(6,2) 2*RT(3,2)*RT(6,3) RT(3,3)*RT(6,3)];
C = [1;1;1;1;1;1;0;0;0;];
LTemp = pinv(A)*C;
L = [LTemp(1) LTemp(2) LTemp(3);
     LTemp(2) LTemp(4) LTemp(5);
     LTemp(3) LTemp(5) LTemp(6)];

%Eigenvalues
[diagL matL] = eig(L);
R = RT*diagL;
S = inv(diagL)*STemp

%Display
%With the reconstruction, some of the points are on top
%of each other
figure();

%Pretty colors to differenciate to draw the lines but actually it's pretty
plot3(S(1,1),S(2,1),S(3,1),'ro');
hold on
plot3(S(1,2),S(2,2),S(3,2),'r+');
plot3(S(1,3),S(2,3),S(3,3),'r*');
plot3(S(1,4),S(2,4),S(3,4),'bo');
plot3(S(1,5),S(2,5),S(3,5),'b+');
```

```
plot3(S(1,6),S(2,6),S(3,6),'b*');
plot3(S(1,7),S(2,7),S(3,7),'go');
plot3(S(1,8),S(2,8),S(3,8),'g+');
plot3(S(1,9),S(2,9),S(3,9),'g*');
plot3(S(1,10),S(2,10),S(3,10),'ko');
plot3(S(1,11),S(2,11),S(3,11),'k+');
plot3(S(1,12),S(2,12),S(3,12),'k*');
axis square,  grid on
line([S(1,1:4) S(1,1)],[S(2,1:4) S(2,1)],[S(3,1:4) S(3,1)]);
line([S(1,7:10) S(1,7)],[S(2,7:10) S(2,7)],[S(3,7:10) S(3,7)]);
line([S(1,2) S(1,5)],[S(2,2) S(2,5)],[S(3,2) S(3,5)]);
line([S(1,3) S(1,6)],[S(2,3) S(2,6)],[S(3,3) S(3,6)]);
line([S(1,5) S(1,6)],[S(2,5) S(2,6)],[S(3,5) S(3,6)]);
line([S(1,8) S(1,5)],[S(2,8) S(2,5)],[S(3,8) S(3,5)]);
line([S(1,9) S(1,6)],[S(2,9) S(2,6)],[S(3,9) S(3,6)]);
line([S(1,10) S(1,12)],[S(2,10) S(2,12)],[S(3,10) S(3,12)]);
line([S(1,7) S(1,11)],[S(2,7) S(2,11)],[S(3,7) S(3,11)]);
line([S(1,12) S(1,11)],[S(2,12) S(2,11)],[S(3,12) S(3,11)]);
line([S(1,1) S(1,11)],[S(2,1) S(2,11)],[S(3,1) S(3,11)]);
line([S(1,4) S(1,12)],[S(2,4) S(2,12)],[S(3,4) S(3,12)]);

xlabel('X')
ylabel('Y')
zlabel('Z')
```

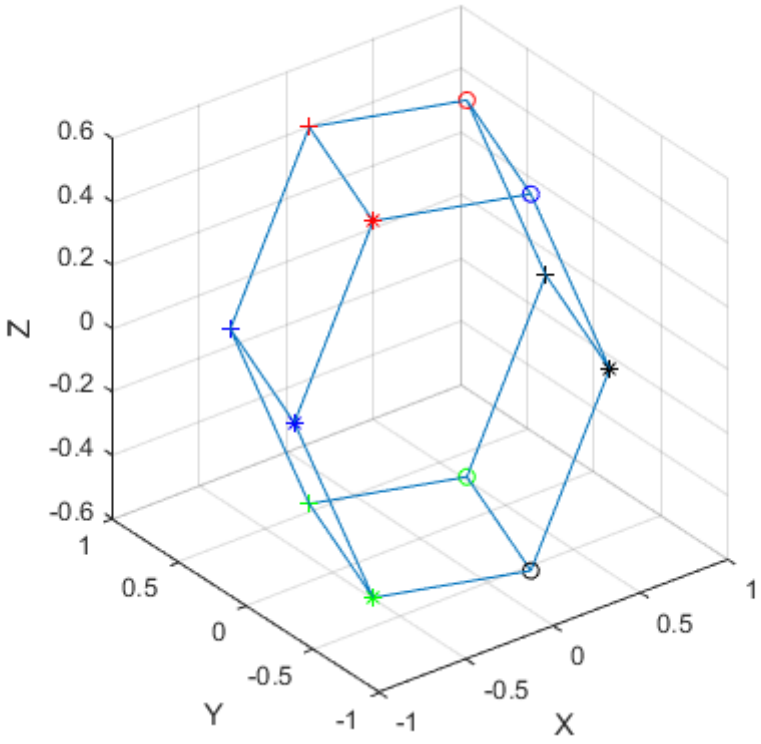

Figure 1 : Reconstruction of the hexagonal prism

5-3D reconstruction of the actual images provided (hotel sequence). You can make the detection of corners using the "detectCheckerboardpoints" command. Use a minimum of 10 images. Points used must be visible on all images in the sequence.

```
% Hotel

clear all;
%close all;
% Read sequence of photos
H{1}=imread('hotel.seq0.png');
```

```
H{2}=imread('hotel.seq4.png');
H{3}=imread('hotel.seq8.png');
H{4}=imread('hotel.seq12.png');
H{5}=imread('hotel.seq16.png');
H{6}=imread('hotel.seq20.png');
H{7}=imread('hotel.seq24.png');
H{8}=imread('hotel.seq28.png');
H{9}=imread('hotel.seq32.png');
H{10}=imread('hotel.seq36.png');

% Put points by hand in an easily readable file
% Bad Idea finally, by hand is dirty
% imshow(H1);
% [x1,y1] = getpts;
% imshow(H2);
% [x2,y2] = getpts;
% imshow(H3);
% [x3,y3] = getpts;
% imshow(H4);
% [x4,y4] = getpts;
% imshow(H5);
% [x5,y5] = getpts;
% imshow(H6);
% [x6,y6] = getpts;
% allX = [x1;x2;x3;x4;x5;x6]
% allY = [y1;y2;y3;y4;y5;y6]
% allPoints = [allX allY]
% dlmwrite('points.pts',allPoints,'delimiter',' ');

%Get all the points
for i=1:10
    %figure
    HF{i} = detectHarrisFeatures(H{i});
    pH{i} = HF{i}.selectStrongest(6).Location;
    %imshow(H{i}); hold on;
    %plot(HF{i}.selectStrongest(6));
end
allPoints=[pH{1,1}];
% Get the points
for i=2:10
    allPoints=[allPoints; pH{1,i}];
end
%allPoints = load('Points.pts');
allX = allPoints(:,1);
allY = allPoints(:,2);

% Measurement Matrix
W = [allX(1:6)';allX(7:12)';allX(13:18)';...
    allX(19:24)';allX(25:30)';allX(31:36)';...
    allX(37:42)';allX(43:48)';allX(49:54)';allX(55:60)';...
    allY(1:6)';allY(7:12)';allY(13:18)';...
    allY(19:24)';allY(25:30)';allY(31:36)';...
    allY(37:42)';allY(43:48)';allY(49:54)';allY(55:60)'];

% Means to normalize around 0
for i=1:10
   meanX{i} = mean(W(i,:));
   meanY{i} = mean(W(i+10,:));

   W(i,:) = W(i,:)-meanX{i};
   W(i+10,:) = W(i+10,:)-meanY{i};
```

```
end

%Single value decomposition & Fill L
[U D V] = svd(W);
Ucut = [U(:,1) U(:,2) U(:,3)];
Dcut = [D(1,1:3);D(2,1:3);D(3,1:3)];
Vcut = [V(:,1) V(:,2) V(:,3)];

RT = Ucut*Dcut^(0.5);
STemp = Dcut^(0.5)*Vcut';

A = [RT(1,1)*RT(1,1) 2*RT(1,1)*RT(1,2) 2*RT(1,1)*RT(1,3)...
    RT(1,2)*RT(1,2) 2*RT(1,2)*RT(1,3) RT(1,3)*RT(1,3);
    RT(2,1)*RT(2,1) 2*RT(2,1)*RT(2,2) 2*RT(2,1)*RT(2,3)...
    RT(2,2)*RT(2,2) 2*RT(2,2)*RT(2,3) RT(2,3)*RT(2,3);
    RT(3,1)*RT(3,1) 2*RT(3,1)*RT(3,2) 2*RT(3,1)*RT(3,3)...
    RT(3,2)*RT(3,2) 2*RT(3,2)*RT(3,3) RT(3,3)*RT(3,3);
    RT(4,1)*RT(4,1) 2*RT(4,1)*RT(4,2) 2*RT(4,1)*RT(4,3)...
    RT(4,2)*RT(4,2) 2*RT(4,2)*RT(4,3) RT(4,3)*RT(4,3);
    RT(5,1)*RT(5,1) 2*RT(5,1)*RT(5,2) 2*RT(5,1)*RT(5,3)...
    RT(5,2)*RT(5,2) 2*RT(5,2)*RT(5,3) RT(5,3)*RT(5,3);
    RT(6,1)*RT(6,1) 2*RT(6,1)*RT(6,2) 2*RT(6,1)*RT(6,3)...
    RT(6,2)*RT(6,2) 2*RT(6,2)*RT(6,3) RT(6,3)*RT(6,3);
    RT(1,1)*RT(4,1) 2*RT(1,1)*RT(4,2) 2*RT(1,1)*RT(4,3)...
    RT(1,2)*RT(4,2) 2*RT(1,2)*RT(4,3) RT(1,3)*RT(4,3);
    RT(2,1)*RT(5,1) 2*RT(2,1)*RT(5,2) 2*RT(2,1)*RT(5,3)...
    RT(2,2)*RT(5,2) 2*RT(2,2)*RT(5,3) RT(2,3)*RT(5,3);
    RT(3,1)*RT(6,1) 2*RT(3,1)*RT(6,2) 2*RT(3,1)*RT(6,3)...
    RT(3,2)*RT(6,2) 2*RT(3,2)*RT(6,3) RT(3,3)*RT(6,3)];
C = [1;1;1;1;1;1;0;0;0;];
LTemp = pinv(A)*C;
L = [LTemp(1) LTemp(2) LTemp(3);
    LTemp(2) LTemp(4) LTemp(5);
    LTemp(3) LTemp(5) LTemp(6)];

%Eigenvalues
[diagL matL] = eig(L);
R = RT*diagL;
S = inv(diagL)*STemp;

%Trying to reconstruct the points on the hostel

figure
imshow(H{1}); hold on;
plot(allPoints(1:6,1),allPoints(1:6,2),'+r');
figure
plot3(S(1,:),S(2,:),S(3,:),'+b');
axis square, grid on
xlabel('X')
ylabel('Y')
zlabel('Z')
```

Figure 2 : Chosen points for the house

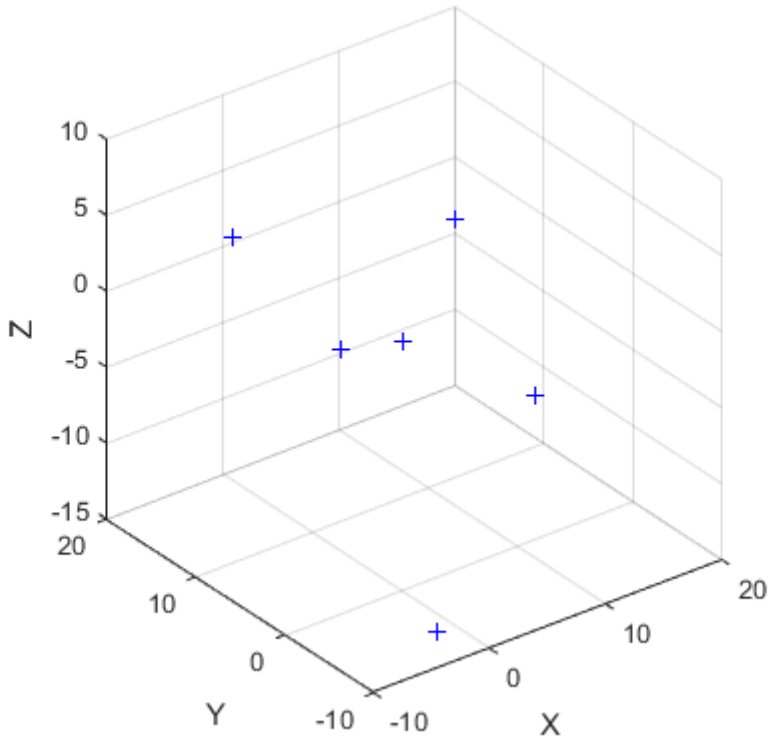

Figure 3 : Reconstruction of these points

6-Suppose that some of the points used are no longer visible (or disappear) in some of the sequence images. How could you solve the problem? Do not consider the trivial solution to eliminate them from the whole sequence.

Depending on the 3d model, we could assume that they passed behind the object if the other points concur. We could also make a mean of the two points of the sequence before and after to try and put it back in its place, while considering that it disappeared behind something if it never reappears.