CS7495 Computer Vision
Assignment 3: Making a Quick-time VR Movie
(Part1 – Computing Homographies)
Gth749a : Kihwan Kim

1. Objective

A homography is a point to point mapping between two spaces, In 2D, a homography is defined by a 3 by 3 matrix H, which relates the points p in one image to corresponding points p’ in the second image.

\[ Wp’ = Hp \]

Where w is a scale parameter.
With given image set (library1.jpg ~library3.jpg), make their homographies and make the mosaic images by warping algorithm.

2. How did I made the homographies ?

: By manually collected n corresponding points, we can calculate more than 2n th equations, in other word, make matrix A where \( A^*h = 0 \).

A matrix is given by

\[
\begin{align*}
    h11x + h12y + h13 & - h31xx’ - h32yx’ - x’ = 0 \\
    h21x + h22y + h23 & - h31xy’ - h32yy’ - y’ = 0
\end{align*}
\]

and it is 2n x 9 matrix. Then we can do ‘svd’ to calculate it. But before we do the SVD, we should do normalization to make this process insensitive to translate. To do this, we should (1) Center the measurement(T) (2) Scale the modified measurements(S). After we get a solution through SVD we need to de-normalize the solution. Then we can find out the homography.

3. Result and Analysis.

to At first, I take the H1, anchor image(library1)’s homography, like below, to make the center position and scales.

\[
H1 = \begin{bmatrix} 0.8 & 0 & -100 \\ 0 & 0.8 & -100 \\ 0 & 0 & 1.0 \end{bmatrix}
\]

Then, H2, the homography between library1 to library2 is shown below,

\[
H2 = \begin{bmatrix} 0.0392 & -0.5859 & 236.8917 \\ 0.5871 & 0.0099 & -61.9187 \\ 0.0001 & -0.0000 & 0.5601 \end{bmatrix}
\]

Then, H3, the homography between library1 to library3 is shown below,

\[
H3 = \begin{bmatrix} -0.5538 & -0.0195 & 307.8518 \\ 0.0112 & -0.5766 & 204.9142 \\ 0.0000 & -0.0000 & 0.5783 \end{bmatrix}
\]

Fig1. Input pictures (library1.jpg, library2.jpg, library3.jpg)
The result image is shown in Fig2. it made the mosaic well. Also it shows that algorithm calculated the homographies from 3 images well. However, when I tried to pick some corresponding points very closely, it sometimes shows mismatching boundaries. Even though we normalized, there seems to be more possibility to be distorted little bit when we choose some corresponding points very closely. (Ideally it shouldn’t be). So I’ll show the additional constraints when we comes to make automatic algorithm to find the corresponding points by sift detector and RANSAC algorithm in next step.

4. Sources.

File 1: assign3_1_findH.m
This is executable source.

```matlab
clear all;
% file name : assign3_1_findH.m
% for Assignment 3: Making a Quick-time VR Movie
% (Part1 ? Computing Homographies)
H2 = getHfrom2Image(1,2);
H3 = getHfrom2Image(1,3);

H1=[
0.8 0 -100
0 0.8 -100
0 0 1
];

% read images
I1=imread('library1.jpg');
I2=imread('library2.jpg');
I3=imread('library3.jpg');
```
% do mosaicking
colormosaic=mosaic(2*425,2*640,I1,H1,I2,H2*H1,I3,H3*H1);

figure(2);
imshow(colormosaic);

File 1 : assign3_1_findH.m
This is executable source.

function  H  = getHfrom2Image(no1, no2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% kihwan kim
% function name : getHfrom2Image
% for Assignment 3: Making a Quick-time VR Movie
% (Part1 ? Computing Homographies)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nCor = 4;  % Num of correspondences
l1 = imread(sprintf('library%d.jpg',no1));
imshow(l1);
p{1} = ginput(nCor);
l2 = imread(sprintf('library%d.jpg',no2));
imshow(l2);
p{2} = ginput(nCor);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Normalize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% find mx, my
nCntx1 = 0; nCnty1 = 0; nCntx2 = 0; nCnty2 = 0;
for i=1:nCor
   nCntx1 = nCntx1 + p{1}(i,1);
   nCnty1 = nCnty1 + p{1}(i,2);
   nCntx2 = nCntx2 + p{2}(i,1);
   nCnty2 = nCnty2 + p{2}(i,2);
end
Mx1 = nCntx1/nCor; My1 = nCnty1/nCor;
Mx2 = nCntx2/nCor; My2 = nCnty2/nCor;

T1 = [ 1 0 -Mx1 ; 0 1 -My1 ; 0 0 1 ];
T2 = [ 1 0 -Mx2 ; 0 1 -My2 ; 0 0 1 ];
pN{1} = p{1};
pN{2} = p{2};
P1 = p{1};
P2 = p{2};

% Center the MeasurementspN
for i=1:nCor
   ptemp1 = [ p{1}(i,1) ; p{1}(i,2) ; 1 ];
   ptemp2 = [ p{2}(i,1) ; p{2}(i,2) ; 1 ];
pNtemp1 = T1*ptemp1;
pNtemp2 = T2*ptemp2;
pN{1}(i,1) = pNtemp1(1,1);
\[ pN_1(i,2) = pN_{\text{temp1}}(2,1); \]
\[ pN_2(i,1) = pN_{\text{temp2}}(1,1); \]
\[ pN_2(i,2) = pN_{\text{temp2}}(2,1); \]

end

% Calc avg distance.
Cntdist1 = 0;
Cntdist2 = 0;
for i=1:nCor
    Cntdist1 = Cntdist1 + sqrt(pN_1(i,1)*pN_1(i,1) + pN_1(i,2)*pN_1(i,2));
    Cntdist2 = Cntdist2 + sqrt(pN_2(i,1)*pN_2(i,1) + pN_2(i,2)*pN_2(i,2));
end
AvgDist1 = Cntdist1/nCor;
AvgDist2 = Cntdist2/nCor;

S1 = [\sqrt{2}/(AvgDist1) 0 0 ; 0 \sqrt{2}/(AvgDist1) 0 ; 0 0 1 ];
S2 = [\sqrt{2}/(AvgDist2) 0 0 ; 0 \sqrt{2}/(AvgDist2) 0 ; 0 0 1 ];

for i=1:nCor
    ptemp11 = [pN_1(i,1) ; pN_1(i,2) ; 1];
    ptemp22 = [pN_2(i,1) ; pN_2(i,2) ; 1];
    pNtemp11 = S1*ptemp11;
    pNtemp22 = S2*ptemp22;
    pN_1(i,1) = pNtemp11(1,1);
    pN_1(i,2) = pNtemp11(2,1);
    pN_2(i,1) = pNtemp22(1,1);
    pN_2(i,2) = pNtemp22(2,1);
end

% Making A
%----------------------------------------------------------------------------------------------------------------------------------
for i=1:nCor
    if i==1
        A = [pN_1(1,1) pN_1(1,2) 1 0 0 0 -pN_1(1,1)*pN_2(1,1) -pN_1(1,2)*pN_2(1,1) -pN_2(1,1)];
    else
        tempA = [pN_1(i,1) pN_1(i,2) 1 0 0 0 -pN_1(i,1)*pN_2(i,1) -pN_1(i,2)*pN_2(i,1) -pN_2(i,1)];
        A = [A ; tempA];
    end
end

% SVD
[u , s , v] = svd(A);

% De Normalization
Htemp = v(:,9);
ht = [Htemp(1:3)' ; Htemp(4:6)' ; Htemp(7:9)'];
H = inv(T2)*inv(S2)*ht*S1*T1;