CSE 5524 – Homework #10

11/04/2013

Manjari Akella

1) The file homography.txt contains 16 corresponding 2-D points from two different images, where the first and second columns correspond to the x and y coordinates of the points in the first image and the third and fourth columns correspond to the x and y coordinates of the points in the second image. Load the 2-D points and use the Normalized Direct Linear Transformation algorithm to compute the homography that maps the points from image 1 to image 2 (i.e., P2 = HP1).

Output

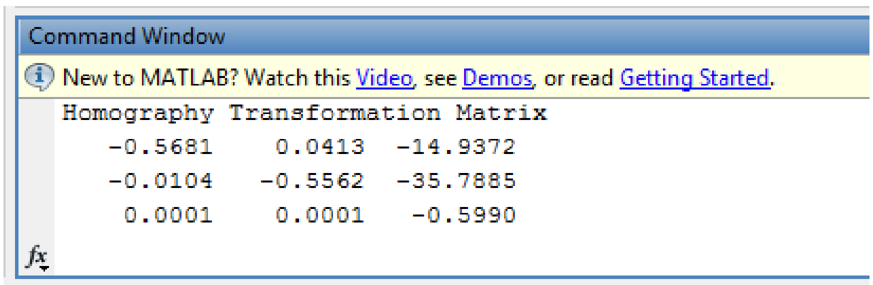


Figure 1: Homography Transformation Matrix

2) Plot the points from image 2 and the projected points from image 1 on the same plot. Make sure the projected points are scaled properly when converting into inhomogeneous form.

Output

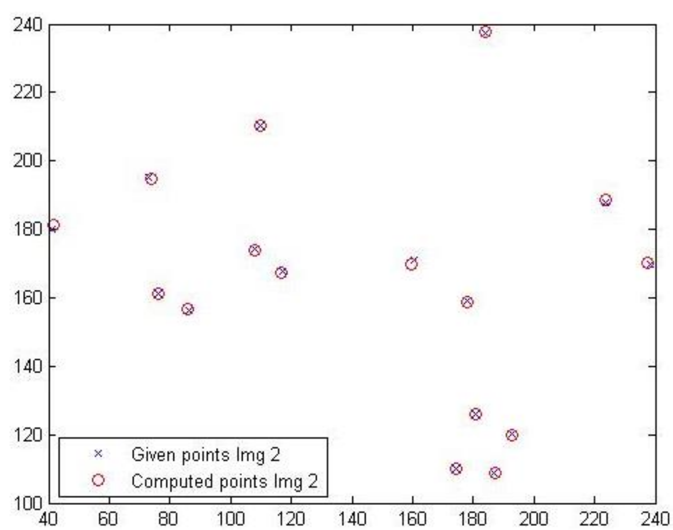
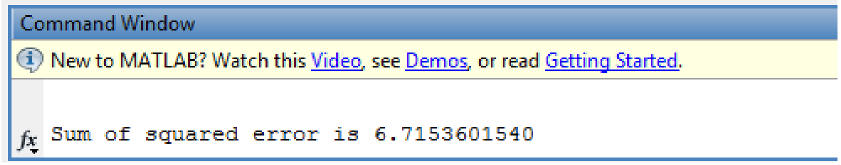


Figure 2: Overlay plot (Given+Projected points)

3) Compute the sum-of-squared error (squared Euclidean distance) between the points from image 2 and the projected points from image 1.

<u>Output</u>

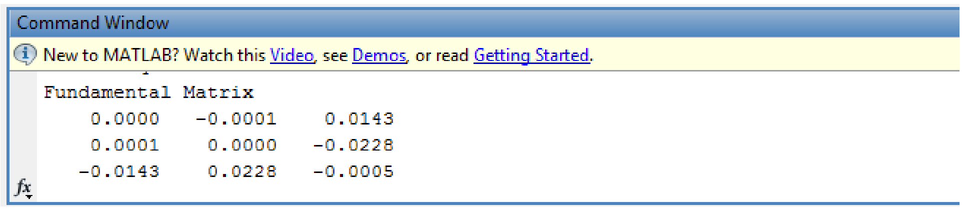

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
Sum of squared error is 6.7153601540
```

Figure 3: Sum of squared error(Homography)

4) The file funMatrix.txt contains 85 points from two images and has the same format as the file homography.txt. Use the normalized 8-point algorithm to compute the fundamental matrix between the image pair assuming the points from the first image correspond to P and the points from the second image correspond to P'. (Remember to make the points homogeneous and enforce the singularity constraint on the fundamental matrix.)

<u>Output</u>

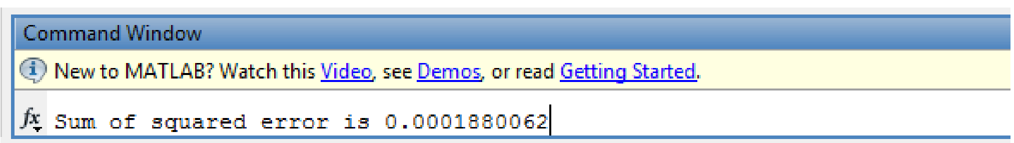

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
Fundamental Matrix
    0.0000   -0.0001    0.0143
    0.0001    0.0000   -0.0228
   -0.0143    0.0228   -0.0005
```

Figure 4: Fundamental Matrix

5) Let the error for the ith pair of corresponding points be $\varepsilon_i = p_i'^{T}Fp_i$. Compute the sum-of-squared error using all 85 points.

<u>Output</u>


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
Sum of squared error is 0.0001880062
```

Figure 5: Sum of squared error

6) Compute a disparity map for the canonically-aligned images left.png and right.png using the basic stereo matching algorithm. Use normalized cross-correlation to perform the template matching for each patch in the left image searching in the right image (search only leftward from the starting point along each row!), and use a window size of 11x11 pixels. Use the following code to display the disparity map D with a gray colormap and clip the disparity values at 50 pixels.

- In Figure 6, the lighter the intensity, the closer the object. Darker (or black) shades correspond to far away objects.
- Figure 7 shows a 'hot' colormap. Hotter shades represent far away objects. For e.g. – the cone is behind the cube
- The border of black is due to padding D with 0s for invalid pixels(where window can't fit) around the border

<u>Output</u>

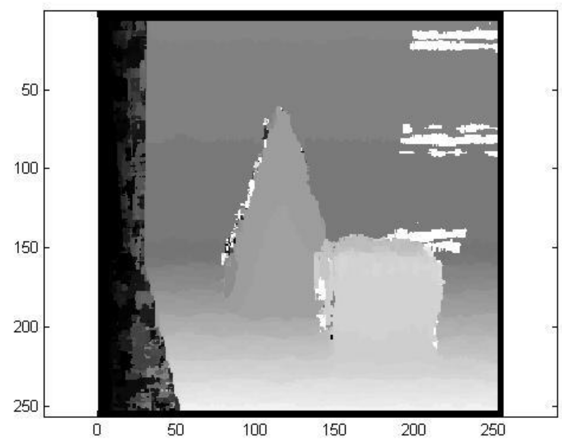


Figure 6: Disparity Map (Gray)

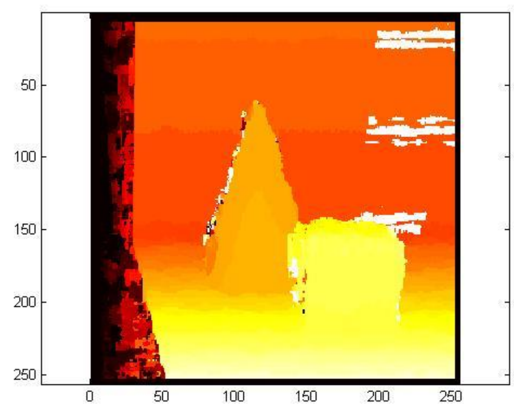


Figure 7: Disparity Map (Hot)

CODE

1).HW10.m

```
% Manjari Akella
% CSE5524 - HW10
% 11/04/2013

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 1
close all;
clear all;
clc;
% Load data
data = load('given_data/homography.txt');
% Set N=number of data points
N = size(data,1);
% Seperate points of the 2 images
P1 = data(:,1:2);
P2 = data(:,3:4);
% Compute mean for each image
m1 = mean(P1);
m2 = mean(P2);
% Compute scale factor for each image
s1 = (sqrt(2))/(sum(sqrt(sum(((P1-[repmat(m1(1,1),[N 1]),repmat(m1(1,2),[N
1])]).^2),2)))/N);
s2 = (sqrt(2))/(sum(sqrt(sum(((P2-[repmat(m2(1,1),[N 1]),repmat(m2(1,2),[N
1])]).^2),2)))/N);
% Define T matrices
T1 = [s1,0,-s1*m1(1,1);0,s1,-s1*m1(1,2);0,0,1];
T2 = [s2,0,-s2*m2(1,1);0,s2,-s2*m2(1,2);0,0,1];
% Transform given points
TP1 = [];
TP2 = [];
for i=1:N
    % Multiply in homogenous land
    t = T1*[P1(i,:)';1];
    % Convert back to inhomogenous land
    t = t./t(size(t,1),1);
    % Append into transformed points vector
    TP1 = [TP1;t'];
    t = T2*[P2(i,:)';1];
    t = t./t(size(t,1),1);
    TP2 = [TP2;t'];
end
% Crop off trailing ones
TP1 = TP1(:,1:2);
TP2 = TP2(:,1:2);
% Create A matrix
A=[];
for i=1:N
        % Compute rows for each point
        temp = [TP1(i,1),TP1(i,2),1,0,0,0,(-TP1(i,1)*TP2(i,1)),(-
TP1(i,2)*TP2(i,1)),-TP2(i,1);
                0,0,0,TP1(i,1),TP1(i,2),1,(-TP1(i,1)*TP2(i,2)),(-
TP1(i,2)*TP2(i,2)),-TP2(i,2)];
```

```
        % Append into A matrix
        A = [A;temp];
end
% Transpose
AT = A';
% Multiply
B = AT*A;
% Find  Eigen values and vectors
[EVec,EVal] = eig(B);
% Minimum Eigen value index
[~,ind]=min(diag(EVal)');
% Unrasterize corresponding Eigen Vector to form H of transformed points
TH = [EVec(1:3,ind)';EVec(4:6,ind)';EVec(7:9,ind)'];
% Un-normalize TH
H = T2\TH*T1;
fprintf('Homography Transformation Matrix\n');
disp(H);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 2
computedP = [];
for i=1:N
    % Multiply in homogenous land
    newP = H*[P1(i,:)';1];
    % Convert to inhomogenous land
    newP = newP./newP(size(newP,1),1);
    % Append new 2D point into matrix
    computedP = [computedP;newP'];
end
% Crop off trailing 1 to represent in inhomogenous land
computedP = computedP(:,1:2);
% Plot image points (projected+given)
figure('Name','Q2: Given vs Projected Image Points ','NumberTitle','off');
plot(P2(:,1),P2(:,2),'bx');
hold on;
plot(computedP(:,1),computedP(:,2),'ro');
hleg = legend('Given points Img 2','Computed points Img
2','Location','SouthWest');
hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 3
% (x1-xo).^2,(y1-yo).^2
e = (computedP-P2).^2;
% Sum of squared error
error =sum(sum(e,2));
fprintf('Sum of squared error is %.10f',error);
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 4
clear all;
close all;
data = load('given_data/funMatrix.txt');
% Set N=number of data points
N = size(data,1);
```

```
% Seperate points of the 2 images
P1 = data(:,1:2);
P2 = data(:,3:4);
% Compute mean for each image
m1 = mean(P1);
m2 = mean(P2);
% Compute scale factor for each image
s1 = (sqrt(2))/(sum(sqrt(sum(((P1-[repmat(m1(1,1),[N 1]),repmat(m1(1,2),[N
1])]).^2),2)))/N);
s2 = (sqrt(2))/(sum(sqrt(sum(((P2-[repmat(m2(1,1),[N 1]),repmat(m2(1,2),[N
1])]).^2),2)))/N);
% Define T matrices
T1 = [s1,0,-s1*m1(1,1);0,s1,-s1*m1(1,2);0,0,1];
T2 = [s2,0,-s2*m2(1,1);0,s2,-s2*m2(1,2);0,0,1];
% Transform given points
TP1 = [];
TP2 = [];
for i=1:N
    % Multiply in homogenous land
    t = T1*[P1(i,:)';1];
    % Convert back to inhomogenous land
    t = t./t(size(t,1),1);
    % Append into transformed points vector
    TP1 = [TP1;t'];
    t = T2*[P2(i,:)';1];
    t = t./t(size(t,1),1);
    TP2 = [TP2;t'];
end
% Crop off trailing ones
TP1 = TP1(:,1:2);
TP2 = TP2(:,1:2);
% Create A matrix
A=[];
for i=1:N
        % Append row corresponding to each point into A matrix
        temp =
[(TP2(i,1)*TP1(i,1)),(TP2(i,1)*TP1(i,2)),TP2(i,1),(TP2(i,2)*TP1(i,1)),(TP2(i,
2)*TP1(i,2)),TP2(i,2),TP1(i,1),TP1(i,2),1];
        A = [A;temp];
end
% Transpose
AT = A';
% Multiply
B = AT*A;
% Find  Eigen values and vectors
[EVec,EVal] = eig(B);
% Minimum Eigen value index
[~,ind]=min(diag(EVal)');
% Unrasterize correspoding Eigen Vector to form F of transformed points
TF = [EVec(1:3,ind)';EVec(4:6,ind)';EVec(7:9,ind)'];
% Perfom SVD
[U,S,V] = svd(TF);
S(end)=0;
NewTF = U*S*V';
% Un-normalize the NewF matrix
F = T2'*NewTF*T1;
```

```
fprintf('\nFundamental Matrix\n');
disp(F);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 5
for i=1:N
    e(i,1) = [P2(i,:),1]*F*[P1(i,:)';1];
end
error = sum(e.^2);
fprintf('Sum of squared error is %.10f',error);
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 6
clear all;
close all;
clc;
% Load images
IL = double(imread('given_pics/left.png'));
IR = double(imread('given_pics/right.png'));
% Define size of search and template images, offset, Disparity matrix
sr = size(IR,1);
sc = size(IR,2);
tr = 11;
tc = 11;
offset = ceil(tr/2);
D = zeros(sr,sc);
for i=1:sr-(tr-1)
    i
    for j=1:sc-(tc-1)
        template = IL(i:i+(tr-1),j:j+(tc-1));
        % Left position
        xl = j+offset;
        % Compute mean of template
        mt = mean(template(:));
        % T(x,y)-T
        t_dash = repmat(mt,[tr tc]);
        T = (template-t_dash);
        % Sigma t
        st = std(template(:));
        search = IR(i:i+(tr-1),1:j+(tc-1));
        % NCC
        NCC=[];
        for col=1:(size(search,2)-(tc-1))
            patch = search(1:tr,col:col+(tc-1));
            % Compute mean of patch
            mp = mean(patch(:));
            % P(x,y)-P
            p_dash = repmat(mp,[tr tc]);
            % (P(x,y)-P).(T(x,y)-T)
            t = (patch-p_dash).*T;
            % Sigma p
            sp = std(patch(:));
            % ((P(x,y)-P).(T(x,y)-T))/(sigma p*sigma t)
            temp1 = t./(sp*st);
```

```
            % Score value of NCC is sum(((P(x,y)-P).(T(x,y)-T))/(sigma
p*sigma t))/n-1
            NCC(1,col) = (sum(sum(temp1)))/((tr*tc)-1);
        end
        % To deal with std(template)=0 case
        flag = isnan(NCC);
        if(all(flag)==1)
            D(i+offset,j+offset)= 0;
        else
            [~,maxpos] = max(NCC);
            % Right position(Best match from NCC)
            xr = maxpos+offset;
            D(i+offset,j+offset)= (xl-xr);
        end
    end
end
figure('Name','Q3: DisparityMap(Gray) ','NumberTitle','off');
imagesc(D, [0 50]);
axis ('equal');
colormap ('gray');
figure('Name','Q3: DisparityMap(Hot) ','NumberTitle','off');
imagesc(D, [0 50]);
axis ('equal');
colormap ('hot');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%
```