CSE 5524 – Homework #9

10/28/2013

Manjari Akella

1). There's an elephant in the room. Can you find it? Search for the template template.png in the search image search.png using color-based SSD, SAD, and NCC (make sure the standard deviation is "unbiased" with N-1). Assume the origin is in the center of the template image for each approach (i.e., there should be a border around the search image where the metrics cannot be computed). Sort the resulting scores from best to worst for each of the approaches. Plot the sorted scores and show the patches corresponding to the 1st, 2nd, 5th, 10th, 100th, and 500th closest matches. Compare your results.

- All the methods are able to find the 'elephant in the room'
- Figures 1,2,3 show the score plot of the best 3000 patches for SAD,SSD,NCC respectively
- The plot for NCC goes from high to low because a high score signifies better match. For SAD and SSD, the plots go from low to high because a low score means a better match
- The most noticeable difference in the patches (Figure 4,5,6), is the patches plotted at k=500
- While SAD and SSD still show parts of the elephant, NCC doesn't feature any part of the elephant. The NCC plot might have been misled by the purple clothes and gray hair in the patch
- Further, SAD captures more of the 'gray' area of the elephant whereas SSD captures top parts of the elephant including the purple bow
- For k=1,2,5,10, all methods were able to isolate the elephant as in the template
- For k=100, SAD and NCC methods don't capture the purple bow. On the other hand SSD captures the bow but not enough structure of John Swatzwelder, standing beside the elephant

Output

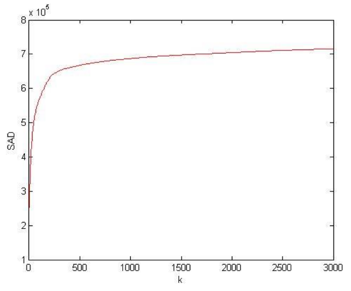


Figure 1: SAD scores

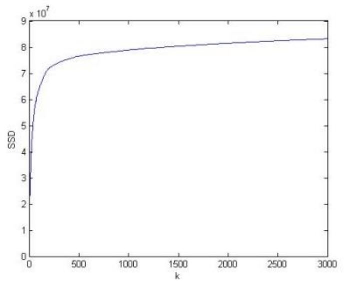


Figure 2: SSD scores

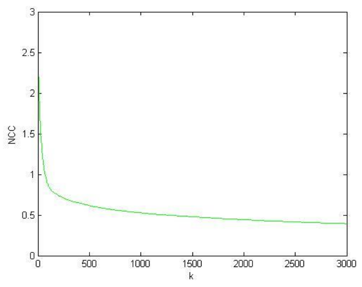
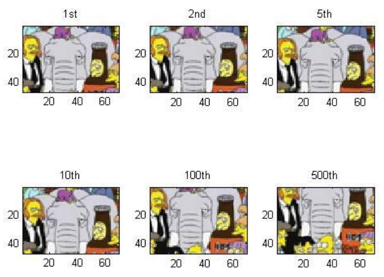
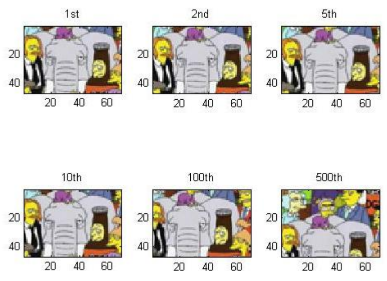


Figure 3: NCC scores





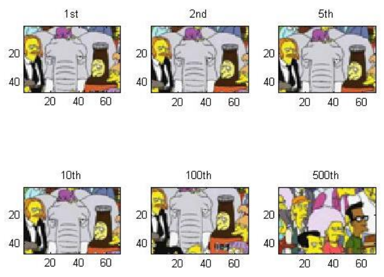
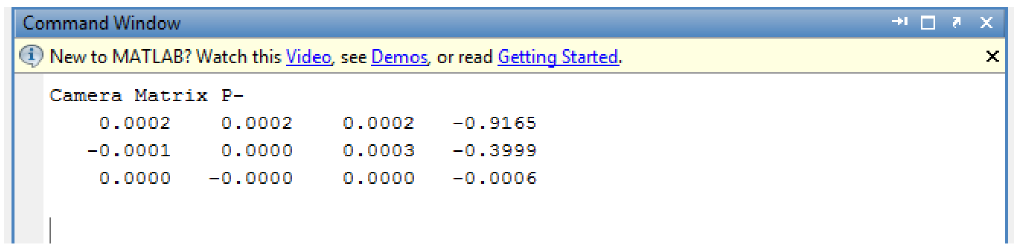


Figure 4: SAD best patches **Figure 5: SSD best patches** **Figure 6: NCC best patches**

2). Load the 100 pairs of corresponding 2-D and 3-D points in the files 2Dpoints.txt and 3Dpoints.txt (the ith row of both files corresponds to the ith point). Use the point correspondences to solve for the camera matrix P (whose rasterized vector p has a unit L2 norm).

- Figure 7 shows the camera matrix computed using point correspondences

<u>Output</u>


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

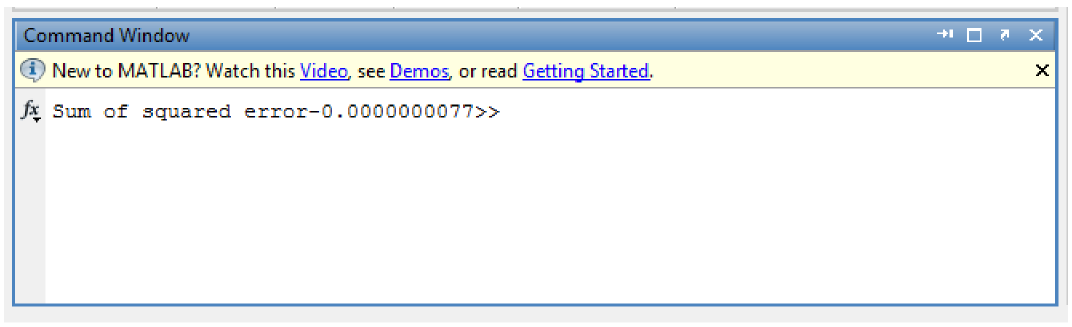
```
Camera Matrix P-
     0.0002     0.0002     0.0002    -0.9165
    -0.0001     0.0000     0.0003    -0.3999
     0.0000    -0.0000     0.0000    -0.0006
```

Figure 7: Camera Matrix

3). Given the matrix P (from Problem 2), project the 3-D homogeneous points (Xi,Yi,Zi,1) onto the 2-D image plane. Compute the sum-of-squared error (sum-of squared distances) between the resulting 2-D points and the given 2-D points (ensure all 2-D points are inhomogeneous).

- As can be seen in figure 8, the sum of squared error is very small, meaning the camera matrix computed provides an *almost perfect* conversion

<u>Output</u>


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

```
Sum of squared error-0.0000000077>>
```

Figure 8: Sum of squared error

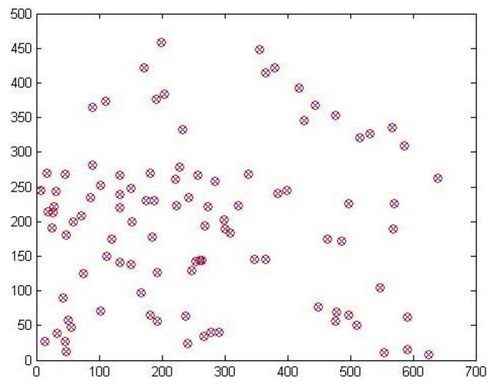


Figure 9: Given 2D points(blue x), Computed 2D points(red o)

CODE

1). HW9.m

```
% Manjari Akella
% CSE5524 - HW9
% 10/28/2013

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 1
clc;
clear all;
close all;
% Read template and search image
template = double(imread('given_pics/template.png'));
search = double(imread('given_pics/search.png'));
% Define size of search and template images
sr = size(search,1);
sc = size(search,2);
tr = size(template,1);
tc = size(template,2);
% Compute the center of the template image (to be used as origin)
center = [ceil(tr/2),ceil(tc/2)];
% Define SAD score matrix
SAD = zeros(sr,sc);
% SAD
for r=1:sr-(tr-1)
    for c=1:sc-(tc-1)
        patch = search(r:r+(tr-1),c:c+(tc-1),:);
        % Compute SAD score for each patch
        SAD(r+(center(1,1)-1),c+(center(1,2)-1)) = sum(sum(sum(abs(patch-
template))));
    end
end
% Sort in ascsnding order
temp = sort(SAD(:),'ascend');
% Find index of last 0 (due to initialization of matrix)
ind = find(temp==0,1,'last');
% Get rid of 0s to find actual top 3000 score values
temp = temp(ind+1:ind+3000,:);
% Plot SAD scores
```

```
figure('Name','Q1: SAD','NumberTitle','off');
plot(temp,'r');
xlabel('k');
ylabel('SAD');
% Find centers of 1,2,5,10,100,500 best patches in the search image
SAD_new =(SAD<=temp(1000)).*SAD;
[r1,c1] = find(SAD_new==temp(1,1));
[r2,c2] = find(SAD_new==temp(2,1));
[r3,c3] = find(SAD_new==temp(5,1));
[r4,c4] = find(SAD_new==temp(10,1));
[r5,c5] = find(SAD_new==temp(100,1));
[r6,c6] = find(SAD_new==temp(500,1));
% Plot 1,2,5,10,100,500 best patches best patches
p1 = search(r1-(center(1,1)-1):r1+(center(1,1)-1),c1-(center(1,2)-
1):c1+(center(1,2)-1),:);
p2 = search(r2-(center(1,1)-1):r2+(center(1,1)-1),c2-(center(1,2)-
1):c2+(center(1,2)-1),:);
p3 = search(r3-(center(1,1)-1):r3+(center(1,1)-1),c3-(center(1,2)-
1):c3+(center(1,2)-1),:);
p4 = search(r4-(center(1,1)-1):r4+(center(1,1)-1),c4-(center(1,2)-
1):c4+(center(1,2)-1),:);
p5 = search(r5-(center(1,1)-1):r5+(center(1,1)-1),c5-(center(1,2)-
1):c5+(center(1,2)-1),:);
p6 = search(r6-(center(1,1)-1):r6+(center(1,1)-1),c6-(center(1,2)-
1):c6+(center(1,2)-1),:);
figure('Name','Q1: SAD best
patches','NumberTitle','off'),subplot(2,3,1),imagesc(uint8(p1));
axis('image');
title('1st');
subplot(2,3,2),imagesc(uint8(p2));
axis('image');
title('2nd');
subplot(2,3,3),imagesc(uint8(p3));
axis('image');
title('5th');
subplot(2,3,4),imagesc(uint8(p4));
axis('image');
title('10th');
subplot(2,3,5),imagesc(uint8(p5));
axis('image');
title('100th');
subplot(2,3,6),imagesc(uint8(p6));
axis('image');
title('500th');

% SSD
% Define SSD score matrix
SSD = zeros(sr,sc);
for r=1:sr-(tr-1)
    for c=1:sc-(tc-1)
        patch = search(r:r+(tr-1),c:c+(tc-1),:);
        % Compute SSD score for each patch
        SSD(r+(center(1,1)-1),c+(center(1,2)-1)) = sum(sum(sum((patch-
template).^2)));
    end
end
% Sort in ascsnding order
```

```
temp = sort(SSD(:),'ascend');
% Find index of last 0 (due to initialization of matrix)
ind = find(temp==0,1,'last');
% Get rid of 0s to find actual top 3000 score values
temp = temp(ind+1:ind+3000,:);
% Plot SSD scores
figure('Name','Q1: SSD','NumberTitle','off');
plot(temp,'b');
xlabel('k');
ylabel('SSD');
% Find centers of 1,2,5,10,100,500 best patches in the search image
SSD_new =(SSD<=temp(1000)).*SSD;
[r1,c1] = find(SSD_new==temp(1,1));
[r2,c2] = find(SSD_new==temp(2,1));
[r3,c3] = find(SSD_new==temp(5,1));
[r4,c4] = find(SSD_new==temp(10,1));
[r5,c5] = find(SSD_new==temp(100,1));
[r6,c6] = find(SSD_new==temp(500,1));
% Plot 1,2,5,10,100,500 best patches best patches
p1 = search(r1-(center(1,1)-1):r1+(center(1,1)-1),c1-(center(1,2)-
1):c1+(center(1,2)-1),:);
p2 = search(r2-(center(1,1)-1):r2+(center(1,1)-1),c2-(center(1,2)-
1):c2+(center(1,2)-1),:);
p3 = search(r3-(center(1,1)-1):r3+(center(1,1)-1),c3-(center(1,2)-
1):c3+(center(1,2)-1),:);
p4 = search(r4-(center(1,1)-1):r4+(center(1,1)-1),c4-(center(1,2)-
1):c4+(center(1,2)-1),:);
p5 = search(r5-(center(1,1)-1):r5+(center(1,1)-1),c5-(center(1,2)-
1):c5+(center(1,2)-1),:);
p6 = search(r6-(center(1,1)-1):r6+(center(1,1)-1),c6-(center(1,2)-
1):c6+(center(1,2)-1),:);
figure('Name','Q1: SSD best
patches','NumberTitle','off'),subplot(2,3,1),imagesc(uint8(p1));
axis('image');
title('1st');
subplot(2,3,2),imagesc(uint8(p2));
axis('image');
title('2nd');
subplot(2,3,3),imagesc(uint8(p3));
axis('image');
title('5th');
subplot(2,3,4),imagesc(uint8(p4));
axis('image');
title('10th');
subplot(2,3,5),imagesc(uint8(p5));
axis('image');
title('100th');
subplot(2,3,6),imagesc(uint8(p6));
axis('image');
title('500th');

% NCC
% Define NCC score matrix
NCC = zeros(sr,sc);
% Compute mean of template for R,G,B
TR = template(:,:,1);
TG = template(:,:,2);
```

```
TB = template(:,:,3);
mt(1,1,1) = mean(TR(:));
mt(1,1,2) = mean(TG(:));
mt(1,1,3) = mean(TB(:));
% Define matrix tr x tc x 3 of ones
one3D=ones(tr,tc,3);
% T(x,y)-T
for i=1:3
    t_dash(:,:,i) = one3D(:,:,i).*mt(:,:,i);
end
T = (template-t_dash);
% Sigma t
st(1,1,1) = std(TR(:));
st(1,1,2) = std(TG(:));
st(1,1,3) = std(TB(:));
for r=1:sr-(tr-1)
    for c=1:sc-(tc-1)
        patch = search(r:r+(tr-1),c:c+(tc-1),:);
        % Compute mean of patch
        PR = patch(:,:,1);
        PG = patch(:,:,2);
        PB = patch(:,:,3);
        mp(1,1,1) = mean(PR(:));
        mp(1,1,2) = mean(PG(:));
        mp(1,1,3) = mean(PB(:));
        % P(x,y)-P
        for i=1:3
            p_dash(:,:,i) = one3D(:,:,i).*mp(:,:,i);
        end
        % (P(x,y)-P).(T(x,y)-T)
        t = (patch-p_dash).*T;
        % Sigma p
        sp(1,1,1) = std(PR(:));
        sp(1,1,2) = std(PG(:));
        sp(1,1,3) = std(PB(:));
        % ((P(x,y)-P).(T(x,y)-T))/(sigma p*sigma t)
        temp1 = t(:,:,1)./(sp(1,1,1)*st(1,1,1));
        temp2 = t(:,:,2)./(sp(1,1,2)*st(1,1,2));
        temp3 = t(:,:,3)./(sp(1,1,3)*st(1,1,3));
        % Score value of NCC is sum(((P(x,y)-P).(T(x,y)-T))/(sigma p*sigma
t))/n-1
        NCC(r+(center(1,1)-1),c+(center(1,2)-1))=
(sum(sum(temp1))+sum(sum(temp2))+sum(sum(temp3)))/(tr*tc-1);
    end
end
% Sort in descending order
temp =sort(NCC(:),'descend');
% Find top 3000 score values
temp = temp(1:3000,:);
% Plot NCC scores
figure('Name','Q1: NCC','NumberTitle','off');
plot(temp,'g');
xlabel('k');
ylabel('NCC');
% Find centers of 1,2,5,10,100,500 best patches in the search image
NCC_new =(NCC>=temp(2000)).*NCC;
[r1,c1] = find(NCC_new==temp(1,1));
```

```
[r2,c2] = find(NCC_new==temp(2,1));
[r3,c3] = find(NCC_new==temp(5,1));
[r4,c4] = find(NCC_new==temp(10,1));
[r5,c5] = find(NCC_new==temp(100,1));
[r6,c6] = find(NCC_new==temp(500,1));
% Plot 1,2,5,10,100,500 best patches best patches
p1 = search(r1-(center(1,1)-1):r1+(center(1,1)-1),c1-(center(1,2)-
1):c1+(center(1,2)-1),:);
p2 = search(r2-(center(1,1)-1):r2+(center(1,1)-1),c2-(center(1,2)-
1):c2+(center(1,2)-1),:);
p3 = search(r3-(center(1,1)-1):r3+(center(1,1)-1),c3-(center(1,2)-
1):c3+(center(1,2)-1),:);
p4 = search(r4-(center(1,1)-1):r4+(center(1,1)-1),c4-(center(1,2)-
1):c4+(center(1,2)-1),:);
p5 = search(r5-(center(1,1)-1):r5+(center(1,1)-1),c5-(center(1,2)-
1):c5+(center(1,2)-1),:);
p6 = search(r6-(center(1,1)-1):r6+(center(1,1)-1),c6-(center(1,2)-
1):c6+(center(1,2)-1),:);
figure('Name','Q1: NCC best
patches','NumberTitle','off'),subplot(2,3,1),imagesc(uint8(p1));
axis('image');
title('1st');
subplot(2,3,2),imagesc(uint8(p2));
axis('image');
title('2nd');
subplot(2,3,3),imagesc(uint8(p3));
axis('image');
title('5th');
subplot(2,3,4),imagesc(uint8(p4));
axis('image');
title('10th');
subplot(2,3,5),imagesc(uint8(p5));
axis('image');
title('100th');
subplot(2,3,6),imagesc(uint8(p6));
axis('image');
title('500th');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 2
close all;
clear all;
clc;
% Load 2D and 3D data points
data2D = load('given_data/2Dpoints.txt');
data3D = load('given_data/3Dpoints.txt');
% Define N
N = size(data2D,1);
% Create A matrix
A=[];
for i=1:N
        % Compute rows for each 3D point
        temp = [data3D(i,1),data3D(i,2),data3D(i,3),1,0,0,0,0,(-
data3D(i,1)*data2D(i,1)),(-data3D(i,2)*data2D(i,1)),(-
data3D(i,3)*data2D(i,1)),-data2D(i,1);
```

```
                0,0,0,0,data3D(i,1),data3D(i,2),data3D(i,3),1,(-
data3D(i,1)*data2D(i,2)),(-data3D(i,2)*data2D(i,2)),(-
data3D(i,3)*data2D(i,2)),-data2D(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);
% Find minimum Eigen Value location
[r,c] = find(EVal==min(EVal(:)));
% Unrasterize corresponding Eigen Vector to form P
PVec = [EVec(1:4,c)';EVec(5:8,c)';EVec(9:12,c)'];
fprintf('Camera Matrix P-\n');
disp(PVec);
% Plot given 2D data points
figure('Name','Q2/3: 2D Data points','NumberTitle','off');
plot(data2D(:,1),data2D(:,2),'bx');
hold on;
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Question 3
computed2D = [];
for i=1:N
    % Multiply in homogenous land
    new2D = PVec*[data3D(i,:)';1];
    % Convert to inhomogenous land
    new2D = new2D./new2D(size(new2D,1),1);
    % Append new 2D point into matrix
    computed2D = [computed2D;new2D'];
end
% Crop off trailing 1 to represent in inhomogenous land
computed2D = computed2D(:,1:2);
% (x1-xo).^2,(y1-yo).^2
e = (computed2D-data2D).^2;
% Sum of squared error
error =sum(sum(e,2));
fprintf('\nSum of squared error-%.10f',error);
% Plot computed points
plot(computed2D(:,1),computed2D(:,2),'ro');
hold off;

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