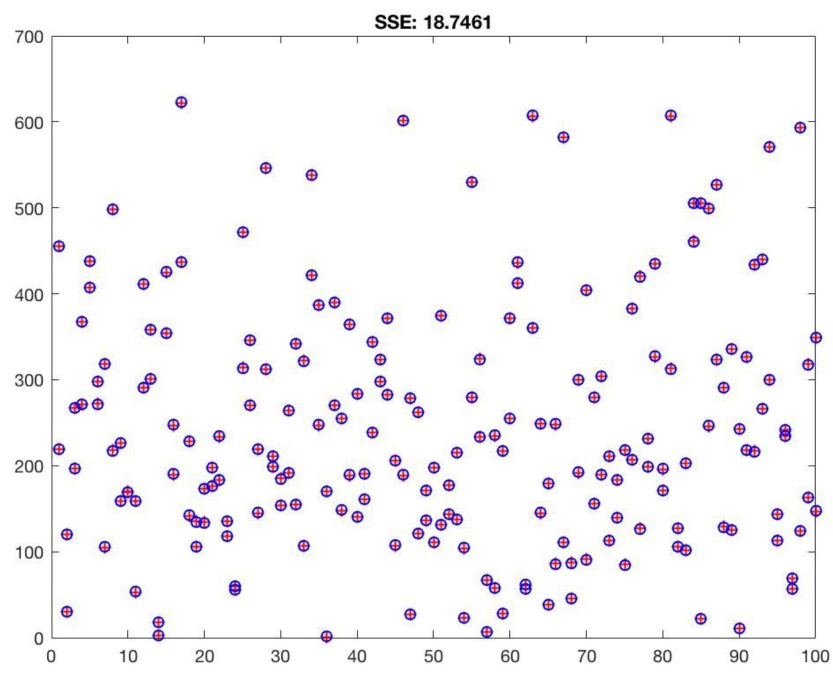
Computer Vision for HCI Abhinav Mahalingam (.15)
CSE 5524
Homework #7

1. Load the 100 pairs of corresponding 2-D and 3-D points in the files 2Dpoints.txt and 3Dpoints.txt (the i_{th} row of both files corresponds to the i_{th} point). Use these point correspondences to solve for the camera matrix P (whose rasterized vector p has a unit L_2 norm). [5 pts]

$$\text{Camera Matrix} = \begin{array}{rrrr} -0.0021 & -0.0020 & -0.0017 & 0.9123 \\ 0.0006 & -0.0001 & -0.0028 & 0.4094 \\ -0.0000 & 0.0000 & -0.0000 & 0.0007 \end{array}$$

2. Given the computed matrix P (from Problem 1), project the 3-D homogeneous points $(X_i,Y_i,Z_i,1)$ to 2-D. Compute the sum-of-squared error (sum-of-squared distances) between the resulting 3-D-to-2-D projected points and the given 2-D points (ensure all 2-D points are inhomogeneous). [3 pts]

The given 2D points are plotted as red + and the projected 2D points are plotted as blue circles in the figure shown below. The SSE was found to be 18.7461.



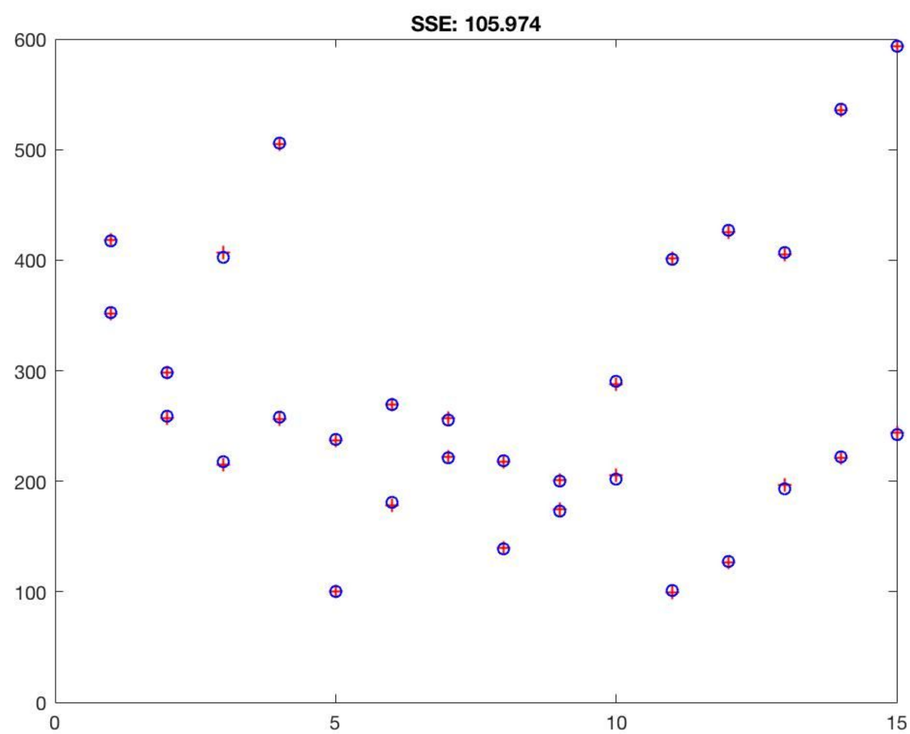
3. The file homography.txt contains 16 corresponding 2-D points from two different images, where the first and second columns correspond to the x and y coordinates of the points in the first image and the third and fourth columns correspond to the x and y coordinates of the points in the second image. Load the 2-D point sets and use the Normalized Direct Linear Transformation algorithm to compute the final homography H that maps the points *from* image 1 to image 2 (i.e., $P_2 = HP_1$). [5 pts]

Homography Matrix =

$$\begin{matrix} 0.3875 & 0.4842 & -21.3951 \\ -0.0609 & 0.2709 & 90.4191 \\ 0.0003 & 0.0003 & 0.4072 \end{matrix}$$

4. Plot the points from image 2 and the projected points from image 1 on the same plot. Make sure the projected points are converted into inhomogeneous form. [1 pt]
5. Compute the sum-of-squared error (squared Euclidean distance) between the actual points from image 2 and the projected points from image 1. [2 pts]

The points from the second image are plotted as red + and the points projected from the first image are plotted as blue circles in the figure shown below. The SSE was found to be 105.9739.



HW8.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1 & 2
world_coords = load('input/3Dpoints.txt');
camera_coords = load('input/2Dpoints.txt');
camera_matrix = compute_camera_matrix(world_coords, camera_coords);
homo_world_space_coords = [world_coords';ones(1,size(world_coords,1))];
homo_projected_camera_coords = camera_matrix * homo_world_space_coords;
projected_camera_coords = convert_to_inhomogeneous(homo_projected_camera_coords);
SSE = compute_SSE(camera_coords, projected_camera_coords);
plot(camera_coords,'r+');
hold on;
plot(projected_camera_coords,'bo');
title(sprintf('SSE: %g', SSE));
hold off;
disp(SSE);
disp(camera_matrix);
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3, 4 & 5
input_data = load('input/homography.txt');
num_of_points = size(input_data, 1);
Im_1 = [input_data(:, 1) input_data(:, 2)];
Im_2 = [input_data(:, 3) input_data(:, 4)];
H = compute_homography(Im_1, Im_2);
homo_projected_points = H * [Im_1'; ones(1, num_of_points)];
projected_points = convert_to_inhomogeneous(homo_projected_points);
SSE = compute_SSE(Im_2, projected_points);
plot(Im_2, 'r+');
hold on;
plot(projected_points, 'bo');
title(sprintf('SSE: %g', SSE));
hold off;
disp(SSE);
disp(H);
```

compute_SSE.m

```
function [SSE] = compute_SSE(coords_1, coords_2)
SSE = sum(sum((coords_1 - coords_2).^2, 2));
end
```

compute_camera_matrix.m

```
function [camera_matrix] = compute_camera_matrix(world_coords, camera_coords)
num_points = size(world_coords, 1);
homo_coord_dim = size(world_coords, 2)+1;
A = zeros( 2*num_points, homo_coord_dim*3);
for n=1:2:2*num_points
   world_coord = world_coords(ceil(n/2),:);
   camera_coord = camera_coords(ceil(n/2),:);
       A(n,:) = [world_coord 1 zeros(1,homo_coord_dim) world_coord.*-camera_coord(1,1)
-camera_coord(1,1)];
      A(n+1,:) = [zeros(1,homo_coord_dim) world_coord 1 world_coord.*-camera_coord(1,2)
-camera_coord(1,2)];
end
[eig_vectors, eig_values] = eig(A' * A);
rasterized_camera_matrix = eig_vectors(:,diag(eig_values) == min(diag(eig_values)));
% camera matrix is already normalized by the eig call
camera_matrix = reshape(rasterized_camera_matrix,homo_coord_dim, 3);
camera_matrix = camera_matrix';
end
```

compute_homography.m

```
function [homography] = compute_homography (Im_1, Im_2)
   [transformed_coords_1, T_1] = similarity_transform(Im_1);
   [transformed_coords_2, T_2] = similarity_transform(Im_2);
   H_1=compute_camera_matrix(transformed_coords_1(1:2,:)', transformed_coords_2(1:2,:)');
   homography = (T_2\H_1) * T_1;
end
```

convert_to_inhomogeneous.m

```
function[inhomogeneous_coords] = convert_to_inhomogeneous(homogeneous_coords)
inhomogeneous_coords(1,:) = homogeneous_coords(1, :) ./ homogeneous_coords(3,:);
inhomogeneous_coords(2,:) = homogeneous_coords(2, :) ./ homogeneous_coords(3,:);
inhomogeneous_coords = inhomogeneous_coords';
end
```

similarity_transform.m

```
function [transformed_coords, T] = similarity_transform(coords)
   mean_x = mean(coords(:, 1));
   mean_y = mean(coords(:, 2));
   s = sqrt(2) / mean(sqrt(((coords(:, 1) - mean_x).^2 + (coords(:, 2) - mean_y).^2)));
   T = [s 0 -s*mean_x; 0 s -s*mean_y; 0 0 1];
   transformed_coords = T * [coords'; ones(1, size(coords, 1))];end
```