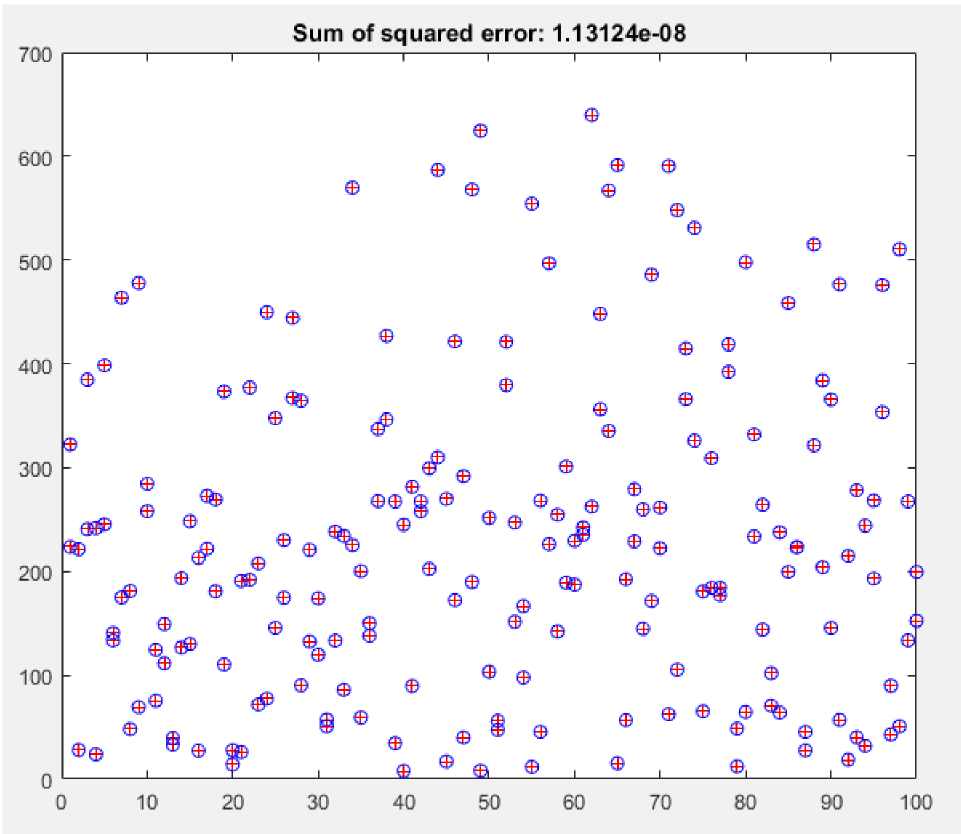
CSE 5524 - Homework #8 **10/12/2016**

Chandrasekar Swaminathan (swaminathan.42)

1) 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).

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



The 2D points from the file are plotted as '+' in the above image along withthe computed points that are plotted as 'o'. The camera matrix that was used to transform 3Dpoints to 2Dpoints is:

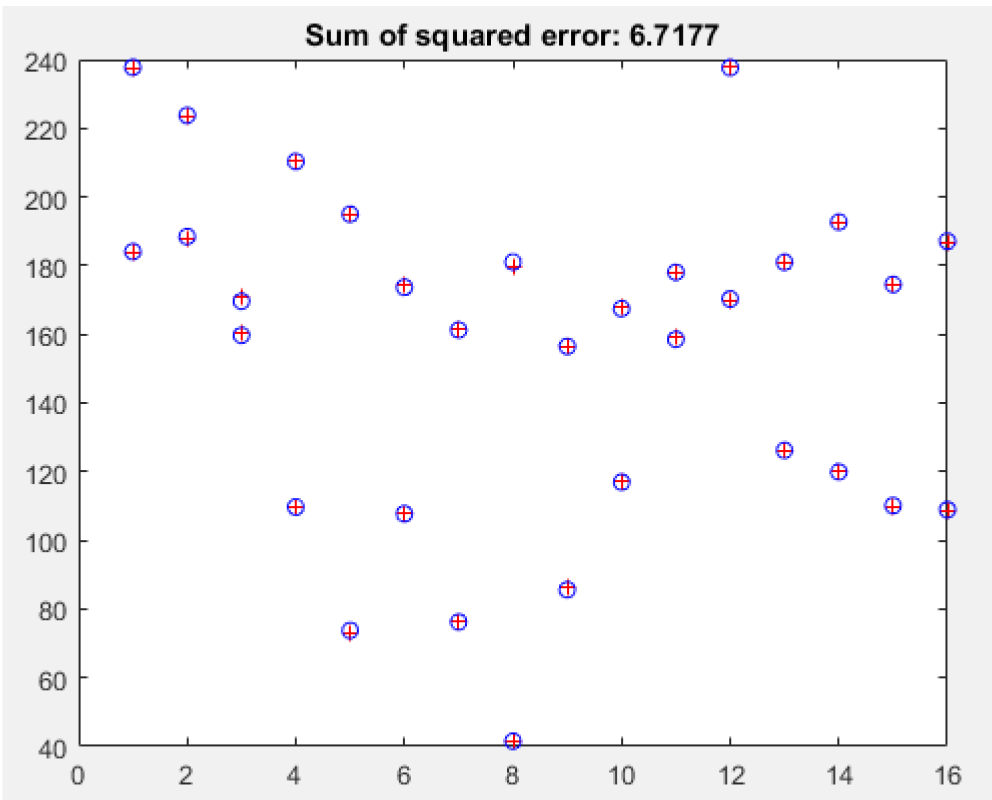
0.0002	0.0002	0.0002	-0.9165
-0.0001	0.0000	0.0003	-0.4000
0.0000	-0.0000	0.0000	-0.0006

The sum of squared errors between the actual 2D points and the computed points is **1.13124e-08**

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., P2 = HP1).

4) 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.

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



The homography matrix was computed as:

$$
\begin{array}{rrr}
0.0108 & -0.0008 & 0.2851 \\
0.0002 & 0.0106 & 0.6826 \\
-0.0000 & -0.0000 & 0.0114
\end{array}
$$

The points from the second image are plotted as '+' in the above image, the points computed by multiplying the homography matrix and the points from the first image are plotted as 'o' in the above image. The sum of squared error between the computed points and the actual points is **6.7177**.

Code

compute_camera_matrix.m
```
function [camera_matrix] = compute_camera_matrix(world_points, camera_points)
    num_points = size(world_points, 1);
    length_of_homo_coords = uint8(size(world_points, 2)+1);
    matrix_a = zeros(2*num_points, (length_of_homo_coords)*3);
    for n=1:2:2*num_points
        world_point = world_points(ceil(n/2), :);
        camera_point = camera_points(ceil(n/2), :);
        matrix_a(n, :) = [world_point 1 zeros(1, length_of_homo_coords) world_point.*-
camera_point(1,1) -camera_point(1,1)];
        matrix_a(n+1, :) = [zeros(1, length_of_homo_coords) world_point 1 world_point.*-
camera_point(1,2) -camera_point(1,2)];
    end
    [eig_vectors, eig_values] = eig(matrix_a' * matrix_a);
    camera_matrix = reshape(eig_vectors(:, diag(eig_values) ==  min(diag(eig_values))),
length_of_homo_coords, 3);
    camera_matrix = camera_matrix';
end
```

compute_homography.m
```
function [norm_homography] = compute_homography (first_image, transformed_image)
    [first_homo_points, first_trans_matrix] = transform_points(first_image);
    [second_homo_points, second_trans_matrix] = transform_points(transformed_image);
    homography = compute_camera_matrix(first_homo_points(1:2, :)', second_homo_points(1:2,
:)');
    norm_homography = inv(second_trans_matrix) * homography * first_trans_matrix;
    norm_homography = norm_homography ./ sum(sum(norm_homography));
end
```

compute_sum_square_error.m
```
function [sum_sq_err] = compute_sum_square_error(points, other_points)
    sum_sq_err = sum(sum((points - other_points).^2, 2));
end
```

convert.m
```
function [inhomogenous_coordinates] = convert(homogenous_coordinates)
    inhomogenous_coordinates(1, :) = homogenous_coordinates(1, :) ./
homogenous_coordinates(3, :);
    inhomogenous_coordinates(2, :) = homogenous_coordinates(2, :) ./
homogenous_coordinates(3, :);
    inhomogenous_coordinates = inhomogenous_coordinates';
end
```

HW8.m

```
% Problem 1 & 2
clear; close all; clc;
world_points = load(fullfile('input', '3Dpoints.txt'));
camera_points = load(fullfile('input', '2Dpoints.txt'));
camera_matrix = compute_camera_matrix(world_points, camera_points);
homo_3D_points = [world_points'; ones(1, size(world_points, 1))];
transformed_homo_2D_points = camera_matrix * homo_3D_points;
transformed_2D_points = convert(transformed_homo_2D_points);
plot(camera_points, 'r+');
hold on;
plot(transformed_2D_points, 'bo');
sum_of_squared_error = compute_sum_square_error(camera_points, transformed_2D_points);
title(sprintf('Sum of squared error: %g', sum_of_squared_error));
disp(sum_of_squared_error);
disp(camera_matrix);
hold off;
pause;
% Problem 3, 4 & 5
clear; close all;
all_points = load('input\homography.txt');
number_of_points = size(all_points, 1);
first_image = [all_points(:, 1) all_points(:, 2)];
second_image = [all_points(:, 3) all_points(:, 4)];
norm_homography = compute_homography(first_image, second_image);
transformed_points = norm_homography * [first_image'; ones(1, number_of_points)];
new_points = convert(transformed_points);
square_error = compute_sum_square_error(second_image, new_points);
plot(second_image, 'r+');
hold on;
plot(new_points, 'bo');
title(sprintf('Sum of squared error: %g', square_error));
hold off;
disp(square_error);
disp(norm_homography);
pause;
close all;
```

transform_points.m

```
function [transformed_points, transform_matrix] = transform_points(points)
   mean_x = mean(points(1, :));
   mean_y = mean(points(2, :));
   s = sqrt(2) / mean(sqrt(((points(1, :) - mean_x).^2 + (points(2, :) - mean_y).^2)));
   transform_matrix = [s 0 -s*mean_x; 0 s -s*mean_y; 0 0 1];
   transformed_points = transform_matrix * [points'; ones(1, size(points, 1))];
end
```