REPORT

PROBLEM 1A:

The coordinates of a set of points in the world frame are given in the file 'pts_world.mat'. Two pictures of these points were taken from a pair of stereo cameras. The corresponding image coordinates for the left and right cameras are given in the files 'pts_viewL.mat' and 'pts_viewR.mat' respectively.

Write a MATLAB function that computes the intrinsic parameters for each camera and the transformations from the world frame to the camera frame. The function is of the form [Kl,Tl,Kr,Tr] = compute_stereo_calib(P, pl, pr) where Kl and Kr are the 3 × 3 matrices for the intrinsic parameters of the left and right cameras, Tl and Tr are the homogeneous transformations from the world frame to the left and right camera frames, and P is the given set of points in the world frame.

Use the calibration technique described in the lecture notes which uses linear least squares to estimate the projection matrix of each camera, and then extracts the intrinsic and extrinsic parameters using the structure of the camera matrix. You can assume that the skew is zero for both cameras. Report all parameters in the PDF report.

Implementation:

1) To compute the projection matrix from given image points and world points. This has been implemented in the `[P, err] = computeCameraProjectionMatrix(X, x)` which takes in world coordinates 'X' and respective image coordinates 'x'. This returns the camera projection matrix 'P'
2) To compute the intrinsic and extrinsic parameters of the camera using the projection matrix. `[K, R, t] = findCameraParameters(P)` has been implemented which takes the projection matrix and computes respective Intrinsic Matrix 'K', Extrinsic Matrix Parameters – Rotational Matrix 'R' and Translation Matrix 't', such that P = K * [R | t].
3) To find the reprojection of the camera using $x = PX$.

Run:

- To run the program, run the file problem_1a, which loads the required parameters 'P', 'pl' and 'pr' and generates the following output.

Note:

- RQ Decomposition is also implemented to find the camera parameters (intrinsic and extrinsic), the output generated has large error, thus discarded.

Output:

Matrix	Left Camera	Right Camera
Projection Matrix	Pl = 0.1111 -0.0394 -0.0022 -0.9806 0.0112 0.0064 -0.0881 -0.1290 0.0002 0.0001 -0.0000 -0.0026	Pr = -0.1009 0.0099 -0.0021 0.9846 -0.0081 -0.0071 0.0773 0.1192 -0.0002 -0.0002 -0.0000 0.0024
Intrinsic Matric	Kl = 0.0885 0 0.0778 -0.0001 0.0878 0.0148 0.0000 -0.0000 0.0003	Kr = 0.0771 0 0.0658 0.0005 0.0776 0.0085 -0.0000 -0.0000 0.0002
Extrinsic Matrix	Tl = -0.4871 0.8733 0.0000 -2.2285 -0.0185 -0.0103 -0.9998 0.2220 0.8731 0.4870 -0.0211 -10.0678	Tr = -0.6933 0.7206 0.0002 3.3933 -0.0204 -0.0199 0.9996 0.3058 -0.7203 -0.6930 -0.0285 10.9815
Actual World Points	pl = 377 47 1 363 39 1 297 54 1 314 60 1 386 119 1 373 118 1 302 126 1 320 126 1 383 139 1 313 144 1	pr = 408 47 1 389 40 1 334 53 1 354 59 1 414 116 1 395 113 1 336 118 1 356 120 1 406 134 1 347 136 1
Computed World Points (K * T * X')	n_pl = 395.1152 46.8548 1.0000 423.7997 38.9607 1.0000 472.9991 53.8069 1.0000 443.0273 59.8065 1.0000 394.3965 119.0913 1.0000 424.6855 118.0356 1.0000 476.3702 125.4348 1.0000 444.6787 125.7755 1.0000 408.8149 138.7958 1.0000 460.4876 143.8724 1.0000	n_pr = 407.8844 47.1162 1.0000 389.2572 39.8360 1.0000 334.2321 53.1382 1.0000 353.8236 58.8496 1.0000 414.2561 116.0171 1.0000 394.8877 113.2030 1.0000 336.0598 117.9483 1.0000 356.5787 120.1473 1.0000 406.5337 133.8248 1.0000 347.2505 135.9190 1.0000

PROBLEM 1B:

Using only the corresponding points in the left and right images, perform 3D triangulation to compute an estimate for the location of the corresponding points in the left camera frame. Write a MATLAB function P_hat = get_world_points(pl, pr) where pl and pr are the sets of corresponding points in the left and right camera frames, and P_hat is the estimated locations of the world points in the left camera frame. Using the extrinsic parameters [Kl,Tl,Kr,Tr] computed in part (a), estimate the average, min, max and standard deviation of the error (defined as distance between the computed and ground truth points). Use this function within a script 'problem_1.m' that generates the required error statistics.

Implementation:

1) To compute the fundamental matrix from given image points of different cameras, function `F = findFundamentalMatrix(x1, x2)` has been added. This takes in image coordinates 'x1' and 'x2' such that it generates fundamental matrix which satisfies $x2 * F * x1' = 0$. This is done using Least Square Methods.
2) Projection Camera Matrix are assumed such that Pl = [I | 0] and Pr = [S(e2)*F, e2], where 'e2' is the epipole of right camera and S(e2) generates an skew-symmetric matrix based on epipole vector of right camera.
3) We take the families of P(X) = pinv(P)*x + lambda*C for both the cameras and create lines such that the 2 points are P(X) = pinv(P)*x [image points, when lambda = 0] and P(X) = C (image center, when lambda = 0).
4) We solve the intersection of the lines to find the world point.

Run:

- To run the program, run the file problem_1, which loads the required parameters 'P', 'pl' and 'pr' and generates the following output.

Output:

P =			P_hat =			Comparing with respect to ground truth: P ErrorMatrix =		
0.4904	-0.0190	-0.0957	1.0812	-0.8910	-0.0141	0.5908	0.8719	0.0816
1.4712	-0.0571	-0.2870	1.0790	-0.8894	-0.0135	0.3921	0.8323	0.2736
1.4712	-2.0187	0.1032	1.0690	-0.8795	-0.0163	0.4021	1.1392	0.1194
0.4904	-1.9806	0.2945	1.0732	-0.8830	-0.0166	0.5828	1.0976	0.3111
0.8806	0.3637	1.8282	1.0747	-0.8856	-0.0222	0.1941	1.2492	1.8504
1.8614	0.3256	1.6369	1.0711	-0.8828	-0.0229	0.7903	1.2084	1.6597
1.8614	-1.6360	2.0270	1.0559	-0.8684	-0.0296	0.8054	0.7676	2.0566
0.8806	-1.5979	2.2184	1.0630	-0.8746	-0.0274	0.1825	0.7234	2.2458
1.4685	0.4403	2.2135	1.0706	-0.8824	-0.0252	0.3979	1.3227	2.2387
1.4685	-1.5213	2.6037	1.0563	-0.8688	-0.0321	0.4123	0.6525	2.6358

E [Total Error] 28.0876	mean_by_axis = 0.4750 0.9865 1.3473	mean = 0.9363
MSE [Mean Squared Error] 4.0544	std_deviation_by_axis = 0.2161 0.2433 1.0248	std_deviation = 1.4841

PROBLEM 2:

Figure 1 shows two images of a scene taken from the left camera (viewL.png) and right camera (viewR.png) of a stereo system. The images have been rectified and are free from radial distortion. Compare the results of following algorithms for computing correspondences and generating a disparity map:

- *Sum of squared differences (SSD)*
- *Cross-correlation (CC)*
- *Normalized cross-correlation (NCC)*

(a) Left camera image. (b) Right camera image.

Figure 1 Images of a scene obtained from a stereo rig

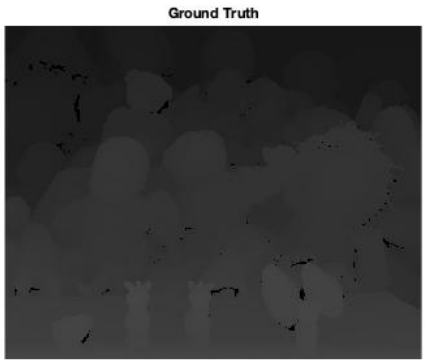
Note that some of these values must be minimized whereas others must be maximized. You will also need to experiment with window sizes and decide on a method to disambiguate in case of ties. Report these decisions as part of your answer. You can use MATLAB functions to convert color images to gray scale. The ground truth for the disparity (right - left) is provided in the file 'disparity.mat'. Write a MATLAB function compute corrs that reads in the images and computes the disparity map using each of the three methods. For each method determine the mean, min, max and standard deviation of the error values using the ground truth. Also report the running times of each method implemented. Using the above criteria, report which method is the best.

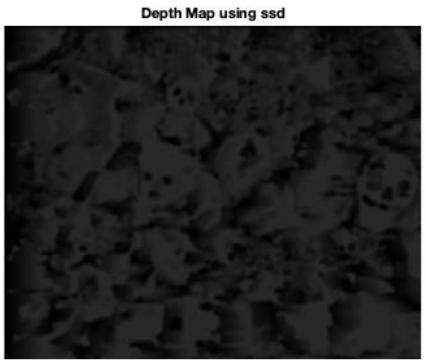
Implementation:

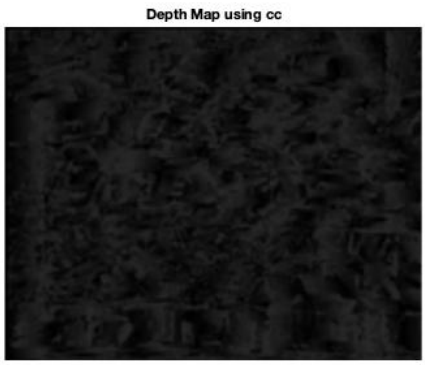
- Initially both the images are padded based on the selected window size. Here we have selected the window size as 5
- Threshold is the max disparity that we can have. Here it is 40.
- For every pixel, a patch is created, where pixel (I,j) is center of the left image and follows the dimensions of the window size.
- Along with the patch, a strip from right image also is created which is the horizontal search zone for the patch created from the left image.
- Based on the threshold, the search is done and based on the method, best disparity is taken and stored for each pixel

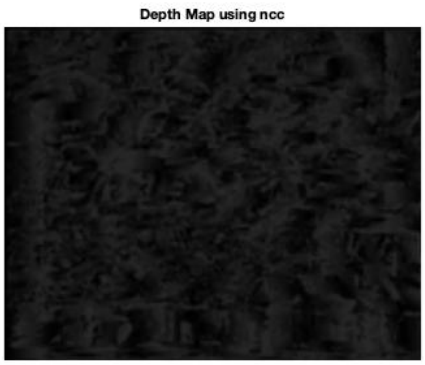
Output:

	SSD	CC	NCC
Error	3564534.44	4363921.56	4363921.56
MSE	650.3887	879.8808	879.8808
Time [HH:MM:SS:FFF]	00:01:52.962	00:05:23.576	00:05:30.761









Note:

- It is observed that we get the best results with SSD method.

PROBLEM 3:

The images 'checkerboard[1-12][l—r].png' in 'checkerboard.zip' were taken using a stereo system. Each square on the checkerboard is of dimensions (25.4mm × 25.4mm). Use the MATLAB camera calibration toolbox (http://www.vision.caltech.edu/bouguetj/calib doc/) to extract the intrinsic parameters for each camera and the extrinsic parameters giving the transformation for points in the right camera frame to those in the left camera frame. Submit the obtained camera parameters in the PDF report. (Hint: axis equal)

Implementation:

1) Load 12 images of a single camera into camera calibration application with 25.4mm.
2) Calibrate using the Calibrate button.
3) Export Camera Parameters to Workspace.
4) For the left camera save as leftCameraParams and for right save as rightCameraParams.
5) Save the sessions into leftCalibrationSession.mat and rightCalibrationSession.mat in Sessions folder.
6) Run the problem_3.m to find the intrinsic and extrinsic parameters of the cameras.

Note:

- Have used Camera Calibrator Application
- Load the left and right images separately, to find properties of the camera.

Output:

	Left Camera	**Right Camera**
Intrinsic Matrix	672.6670 0 344.5111 0 669.4048 249.7597 0 0 1.0000	671.5537 0 344.5883 0 670.1728 235.3556 0 0 1.0000
Extrinsic Matrix	0.8656 -0.1089 0.1914 -95.2544 0.0338 0.8530 0.1687 -117.9041 -0.1796 -0.1190 0.7422 928.9012	0.8647 -0.1116 0.2206 -203.9273 0.0424 0.8522 0.1401 -97.1918 -0.2024 -0.0917 0.7411 933.0623
Projection Matrix	1.0e+05 * 0.0052 -0.0011 0.0038 2.5594 -0.0002 0.0054 0.0030 1.5308 -0.0000 -0.0000 0.0000 0.0093	1.0e+05 * 0.0051 -0.0011 0.0040 1.8457 -0.0002 0.0055 0.0027 1.5447 -0.0000 -0.0000 0.0000 0.0093

Rotational and translation vector for all images

Image	Rotation for l			Translation for l	Rotation for r			Translation for r
checkerboard1	0.9951	-0.0981	0.0117	-77.646	0.9798	-0.0960	0.1753	-176.369
	0.0988	0.9900	-0.1007	132.227	0.1359	0.9630	-0.2326	-110.904
	-0.0017	0.1013	0.9948	919.723	-0.1465	0.2517	0.9567	913.310
checkerboard2	0.9998	0.0193	-0.0018	-182.723	0.9999	0.0052	0.0158	-290.403
	-0.0180	0.9583	0.2851	-127.286	-0.0092	0.9652	0.2612	-105.196
	0.0072	-0.2850	0.9585	936.302	-0.0139	-0.2613	0.9651	938.581
checkerboard3	0.4781	-0.2207	0.8501	-156.084	0.4987	-0.1961	0.8443	-265.508
	0.5212	0.8504	-0.0723	-88.672	0.5335	0.8372	-0.1206	-70.539
	-0.7069	0.4777	0.5216	864.294	-0.6832	0.5106	0.5221	872.763
checkerboard4	0.7663	-0.2042	-0.6092	-75.864	0.7933	-0.2210	-0.5673	-186.749
	-0.0528	0.9249	-0.3765	-67.647	-0.0201	0.9218	-0.3873	-45.616
	0.6403	0.3207	0.6980	1017.124	0.6085	0.3186	0.7268	1014.634
checkerboard5	0.9748	-0.1243	0.1852	169.506	0.9735	-0.1420	0.1795	42.313
	0.1021	0.9869	0.1250	-176.295	0.1259	0.9872	0.0980	-152.767
	-0.1983	-0.1030	0.9747	1024.852	-0.1911	-0.0728	0.9789	1039.038
checkerboard6	0.9846	-0.0238	0.1731	-148.255	0.9813	-0.0251	0.1908	-267.585
	-0.0983	0.7438	0.6612	-426.792	-0.1059	0.7574	0.6443	-397.617
	-0.1445	-0.6680	0.7300	1413.177	-0.1607	-0.6525	0.7406	1423.173
checkerboard7	0.9873	-0.0447	0.1528	-138.005	0.9816	-0.0496	0.1844	-251.861
	-0.0777	0.7022	0.7077	-282.538	-0.0951	0.7105	0.6972	-266.721
	-0.1389	-0.7106	0.6898	1037.582	-0.1655	-0.7019	0.6927	1061.460
checkerboard8	0.9966	-0.0434	0.0698	-140.323	0.9948	-0.0495	0.0887	-246.123
	-0.0262	0.6374	0.7701	-174.479	-0.0346	0.6558	0.7541	-158.420
	-0.0779	-0.7693	0.6341	821.025	-0.0955	-0.7533	0.6507	823.126
checkerboard9	0.9983	0.0294	-0.0507	-109.002	0.9992	0.0207	-0.0339	-208.372
	-0.0270	0.9985	0.0479	-45.060	-0.0196	0.9992	0.0342	-30.937
	0.0520	-0.0464	0.9976	585.671	0.0346	-0.0335	0.9988	587.105
checkerboard10	0.5909	-0.1403	0.7944	-116.478	0.5783	-0.1290	0.8056	-206.352
	-0.0688	0.9724	0.2229	-102.674	-0.0706	0.9758	0.2069	105.130
	-0.8038	-0.1864	0.5650	718.744	-0.8128	-0.1765	0.5552	719.054
checkerboard11	0.6175	-0.4106	0.6709	-104.786	0.6006	-0.4051	0.6893	-217.904
	-0.0152	0.8465	0.5321	87.525	-0.0221	0.8534	0.5207	-76.785
	-0.7864	-0.3388	0.5165	720.565	-0.7993	-0.3280	0.5036	718.310
checkerboard12	0.9977	-0.0455	0.0504	-63.390	0.9959	-0.0516	0.0748	-172.212
	0.0678	0.6247	-0.7779	121.300	0.0908	0.6001	-0.7948	144.075
	0.0039	0.7795	0.6264	1087.748	-0.0039	0.7983	0.6023	1086.188

PROBLEM 4:

Images 'officeL.png' and 'officeR.png', shown in Figure 2, were taken from the stereo pair used for the previous problem. In this problem, you are required to first rectify the two images, then

compute the correspondence between them, and finally produce a depth map and a 3D point cloud for the scene.

PROBLEM 4.1:

Using the camera parameters obtained in problem 3, write a MATLAB function [rectL, rectR] = rectify images(imgL, imgR, Pl, Pr) that takes in the left and right images (imgL and imgR) along with the 3 × 4 camera projection matrices Pl and Pr, and produces the rectified images rectL and rectR.

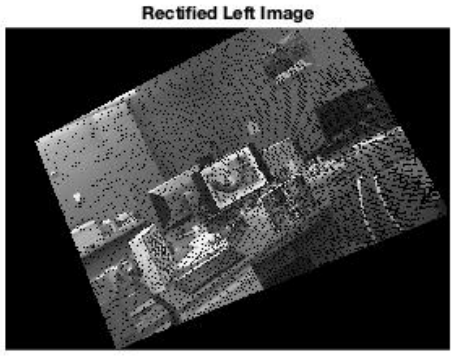
Implementation:

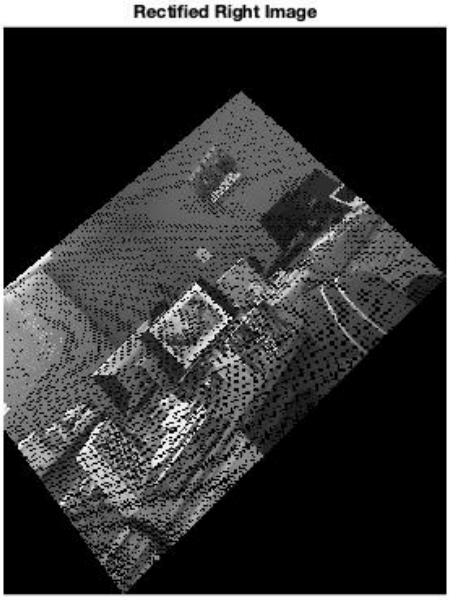
The implementation has been explained in the program `[rectL, rectR] = rectify_images(imgL, imgR, Pl, Pr)`

Note:

- The dimensions of the outputs have been ranging different.

Output:





PROBLEM 4.2:

Use your function compute_corrs in Problem 2 to determine the correspondences between the pair of rectified images using the better of the three methods (SSD, CC, NCC). Using these correspondences along with the camera parameters, determine the 3D locations for all points in the image. (You may choose to ignore those points which do not occur in the other image.) Display the results using a 2D depth map and a 3D point cloud in MATLAB. For this part, submit a script

'problem 4.m' that reads in the image and camera parameters, rectifies them, and generates the point clouds.

[Couldn't complete]

PROBLEM 5:

The file 'linefit.mat' contains three arrays:

- *xs: The x-coordinates of 100 points on a line*
- *n y1: The measurements of the y-coordinates, these measurements have zero-mean Gaussian noise*
- *n y2: A second set of measurements that is similar to the n y1, but contains outliers produced due to device anomalies*

a) *Fit a line through the first set of measurements using a least squares approach. Do not use the pinv function of MATLAB. Instead, compute the pseudoinverse yourself (you can use inv). Report the line parameters.*
b) *Repeat part (a) with the second set of measurements.*
c) *Develop a RANSAC-based algorithm to get rid of the outliers. Explain how you chose the number of points and decided on points that are "close" to the estimated line. Report the line parameters.*

Implementation:

Least Square Method:

- A line can be given as $ax + by + c = 0$, where [a, b] are the coefficients of 'x' and 'y' respectively and 'c' is the intercept (constant). This also can be written as

$$[x \quad y \quad 1]\begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

- This can be written as $\|Ax\| = 0$.
- Here 'A' can be treated as all the homogeneous points in the system and SVD of it, will help us to find the Line equation.
- Since Line has 2 unknowns, the slope and the intercept, we can say that the Line is of rank 2. Thus we select the eigen vector for the second smallest eigen value.

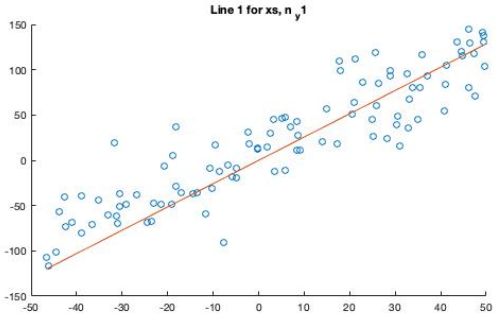
Custom Algorithm to remove outliers:

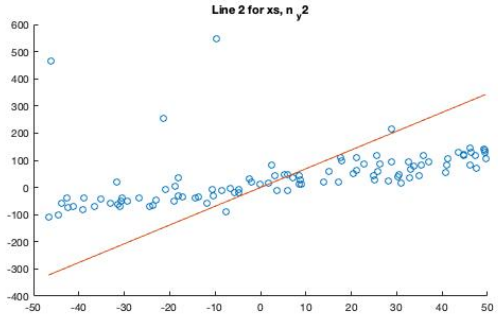
- It has been implemented as a self learning algorithm, following the concepts of neural networks.
- A best fit is calculated using least square method, and distance vector 'd' is calculated.
- Based on mean of 'd' and standard deviation of 'd', the inliers are taken, and a best fit is computed with respect to the same.

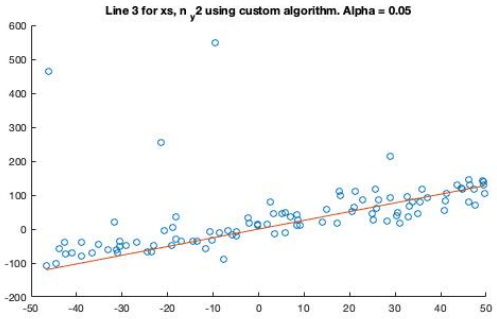
- This is repeated until there is no change in the inliers for couple of continuous loops, thus obtaining the best fit.

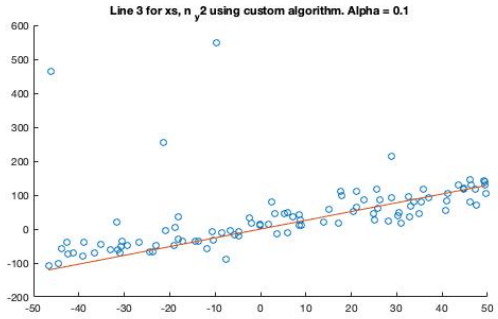
Output:

Line1 = -0.9318 0.3623 0.0238	Line2 = -0.9897 0.1433 0.0008
Alpha = 0.05 Line3 = -0.9317 0.3627 0.0214 Outliers: 28.9029 214.8139 -18.2167 37.3728 -21.4892 256.0695 -46.0816 463.2938 -31.6157 19.7157 -9.6499 548.2151	Alpha = 0.1 Line4 = -0.9323 0.3611 0.0219 Outliers: 28.9029 214.8139 -21.4892 256.0695 -46.0816 463.2938 -31.6157 19.7157 -9.6499 548.2151









References:

- http://mccormickml.com/assets/StereoVision/Stereo%20Vision%20-%20Mathworks%20Example%20Article.pdf
- http://www.cim.mcgill.ca/~langer/558/19-cameracalibration.pdf
- http://www.maths.lth.se/media11/FMAN85/2018/forelas5.pdf
- https://users.cecs.anu.edu.au/~hartley/Papers/triangulation/triangulation.pdf
- https://www.cs.auckland.ac.nz/~rklette/CCV-CIMAT/pdfs/B14-CameraCalibration.pdf
- http://www.sci.utah.edu/~gerig/CS6320-S2012/Materials/CS6320-CV-F2012-Rectification.pdf
- http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO2/node5.html