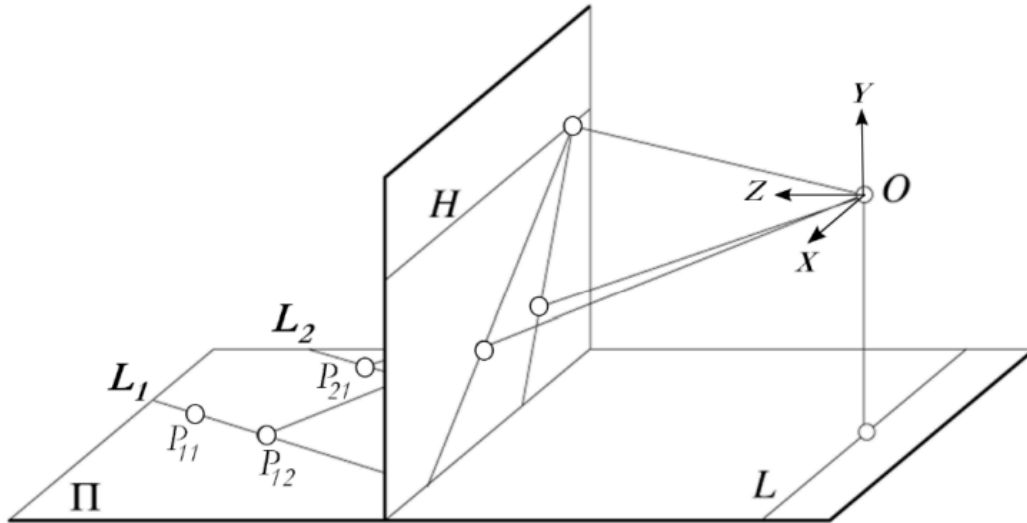# REPORT

PROBLEM 1

Figure 1: The projection of parallel lines.

PROBLEM 1A:

In Figure 1, the origin is co-located with the camera center at point O. The coordinate axes are as shown (the XZ plane is parallel to Π). The equation of plane Π is given as y = −1 while the equation of the image plane is z = 1. Π contains three parallel lines L1, L2 and L3. The points P11 = [−1, −1, 2]T and P12 = [−1, −1, 3]T lie on line L1, P21 = [0, −1, 2]T and P22 = [0, −1, 3]T lie on line L2 and P31 = [1, −1, 2]T and P32 = [1, −1, 3]T lie on line L3. Write a MATLAB function, Q = project point(P), that computes the projection Q on the image plane of a given point P. Apply this function to each of the given points in a MATLAB script. The center of projection is at the origin O. The projection can be computed by determining the point of intersection of the line joining the origin and the given point with the image plane.

Procedure:

1. **Finding the line joining (0,0,0) and the respective point.**
   We know that any point on the line can be expressed as a linear combination of 2 points *(p,q)* on the line. $R = (1 - \lambda) * p + (\lambda * q)$

2. **Find value of λ**
   We can find 'λ' by equating it with 'z' index of the matrix. That is

$$R_z = (1 - \lambda) * p_z + (\lambda * q_z)$$
$$\lambda = \frac{(R_z - p_z)}{(q_z - p_z)}$$

3. Substitute value of λ in first equation to get the point of projection.

SOLUTION:

Point	Actual Point	Point on Image Plane (z = 1)
P11	[-1; -1; 2]	[-0.50; -0.50; 1.00]
P12	[-1; -1; 3]	[-0.33; -0.33; 1.00]
P21	[0; -1; 2]	[0.00; -0.50; 1.00]
P22	[0; -1; 3]	[0.00; -0.33; 1.00]
P31	[1; -1; 2]	[0.50; -0.50; 1.00]
P32	[1; -1; 3]	[0.33; -0.33; 1.00]

PROBLEM 1B:

Using the function project point, write a MATLAB function Q = find intersection(P11, P12, P21, P22) that takes as input two points from each line, L1 and L2, and computes the point of intersection Q of the lines projected on the image plane. Plot the given lines, their projections, and the point of intersection using MATLAB visualization functions (e.g. plot3).

Procedure:

1. To find the line equation
   We know that any point on the line can be expressed as a linear combination of 2 points *(p,q)* on the line. $R = (1 - \lambda) * p + (\lambda * q)$
   Therefore $R_1 = (1 - \lambda) * P11 + (\lambda * P12)$ and $R_2 = (1 - \psi) * P21 + (\psi * P22)$

2. To find λ and ψ
   Say that the lines intersect, then we can equal R_1 and R_2. That is
 $$(1 - \lambda) * P11 + (\lambda * P12) = (1 - \psi) * P21 + (\psi * P22)$$
   $$(P12 - P11)\lambda + (P21 - P22)\psi = (P21 - P11)$$

3. To find R_1 and R_2
   Solving the above linear equation, we get values of λ and ψ. Using the same we can find R_1 and R_2 from the step 1. If R_1 and R_2 are not same, then we can say that lines intersect at $[\infty, \infty, \infty]$ and the point of intersection on image plane would be undefined. That is [NaN, NaN, NaN]. If R_1 and R_2 are same, then it becomes the point of intersection.

SOLUTION:

Let the points for line 1 be [3; 2; 2] and [5; 2; 2]. The points for line 2 be [1, 1, 2] and [-1, -1, 2]. We get the point of intersection as [2, 2, 2] and its projection on image plane as [1, 1, 1].
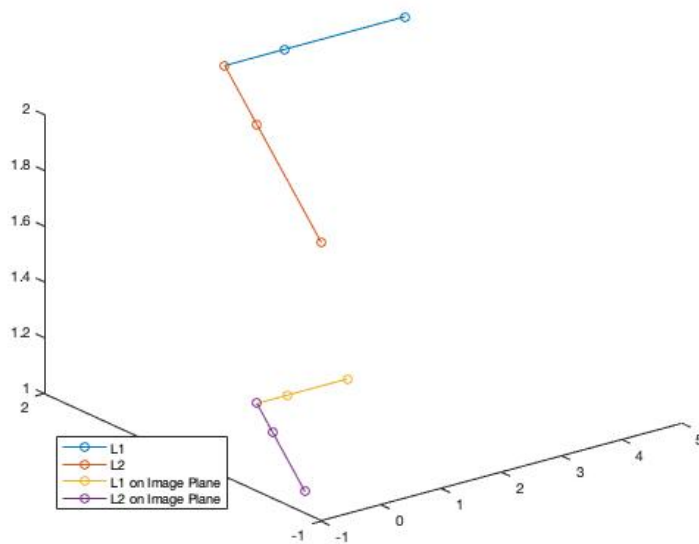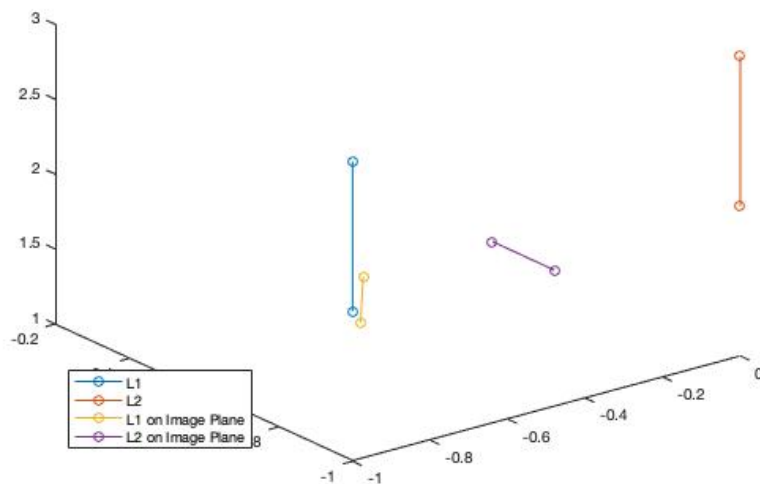
PROBLEM 1C:

Verify that the point of intersection for each pair of parallel lines L1, L2, and L3 is the same by applying the find intersection function to each pair.
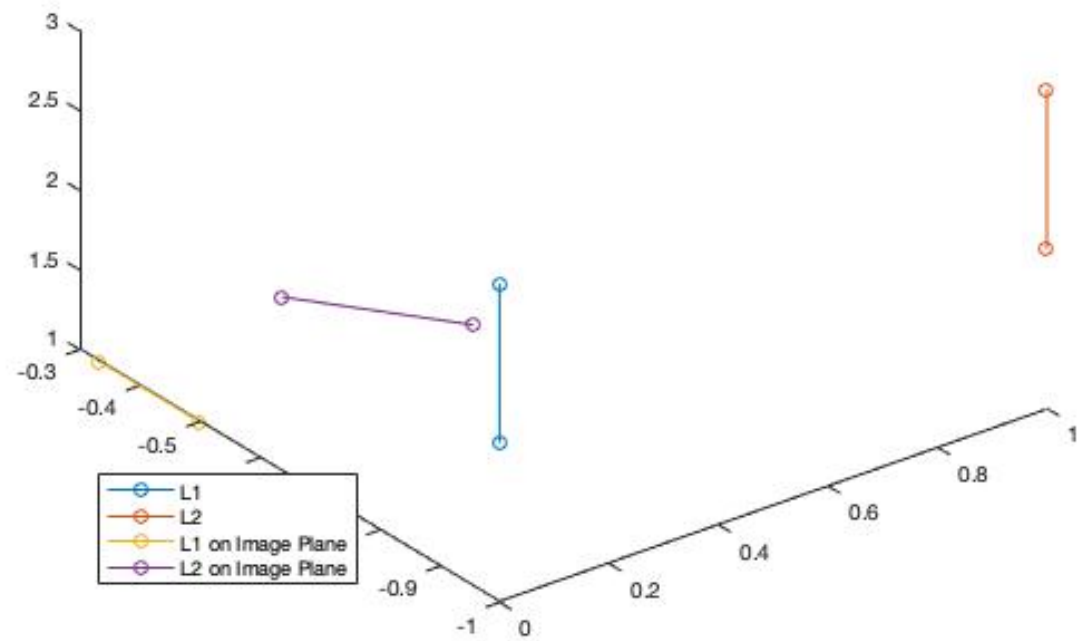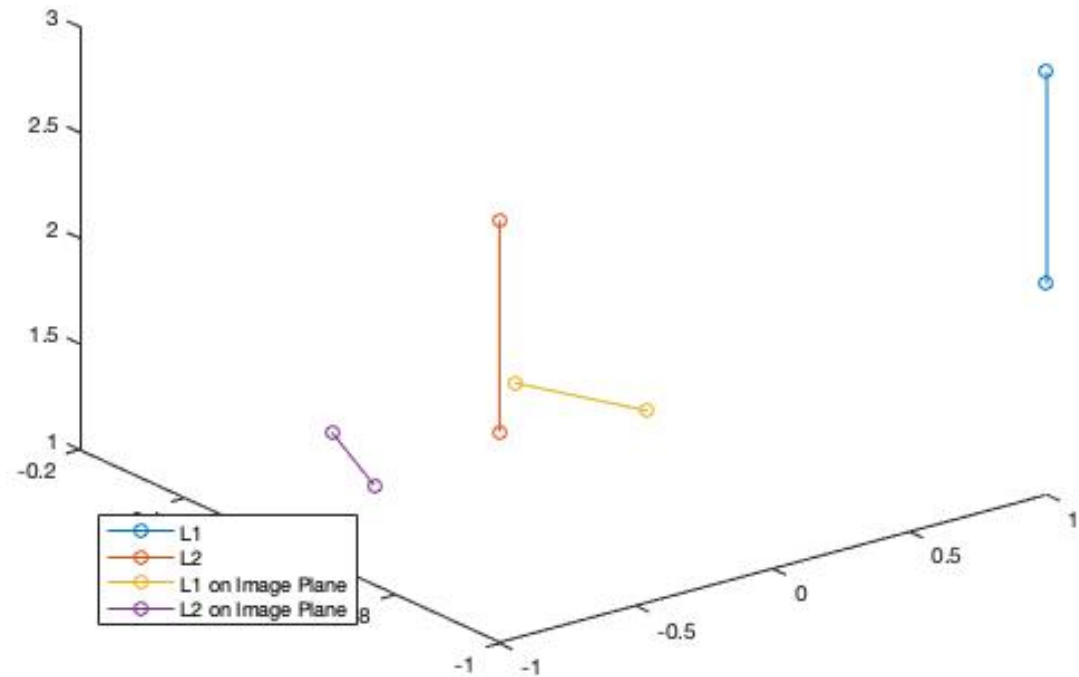
SOLUTION:

1. Comparing line 1 and line 2

This shows that both are parallel lines and intersects at $[\infty, \infty, \infty]$.

2. Comparing line 2 and line 3

This shows that both are parallel lines and intersects at [∞, ∞, ∞].

3. Comparing line 3 and line 1

This shows that both are parallel lines and intersects at $[\infty, \infty, \infty]$.

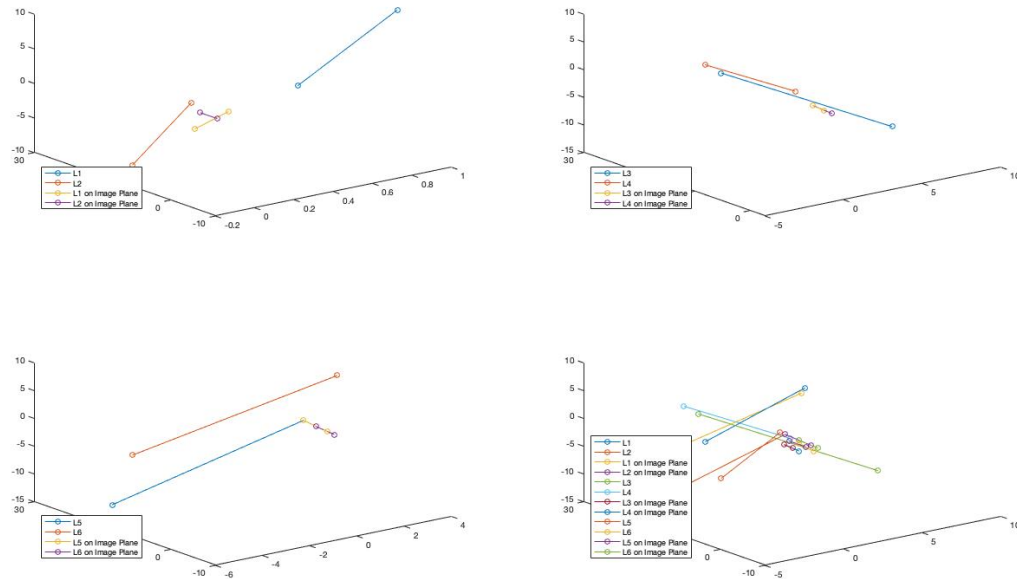## PROBLEM 1D:

Consider three pairs of parallel lines on the plane Π given by the following: (i) x − 1 = 0 and x = 0, (ii) 3x + 2z − 1 = 0 and 3x + 2z − 2 = 0, (iii) 5x − 2z − 1 = 0 and 5x − 2z − 2 = 0. Write a MATLAB script pairwise intersection that uses find intersection to determine the point of intersection for each pair. Verify that the three points found are collinear. Plot the given lines, their projections, and the points of intersection.

SOLUTION:

1. Find 2 points on every line. (function getMyPoints – This generates 2 random points given the line)
2. Store their point of intersection. (Using Problem 1B)
3. Check if the points generated are collinear. (This is done in a way similar to Problem 1A)

We get the 3 points collinear as all the 3 pairs of lines meet at infinity.

PROBLEM 2

The objective of this problem is to introduce basic image processing operations in MATLAB. Load the accompanying "board.tif" image in MATLAB.
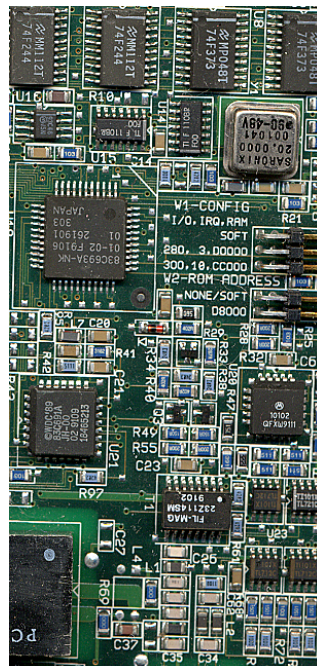
PROBLEM 2A:

Extract the rectangular block of the image between (200, 90) and (300, 180) corresponding to the crystal in the image. Display this block in a separate figure.
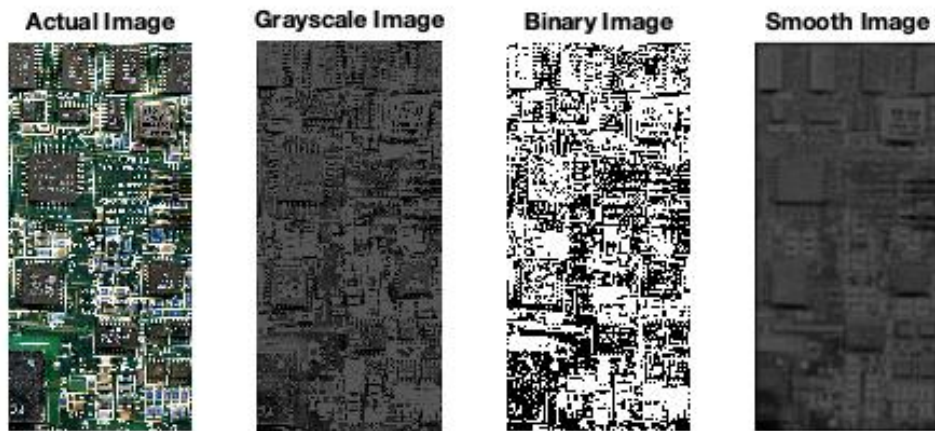

Crystal

OTHER PROBLEMS:

1. Convert the image from RGB to grayscale. For each pixel, take the average of the R, G, and B values as the grayscale value.
2. Convert the grayscale image to a binary image using the mean grayscale value as the threshold. Display both the grayscale and binary images in the same window.
3. Smooth the grayscale image created above using a 7×7 averaging filter. This means that for each pixel at location (i, j), place a 7 × 7 window centered at (i, j) and replace the value of the pixel with the average of the values of the pixels in the window. Decide how you will handle pixels close to the image boundary. Solve this problem in the following two different ways: (i) using for loops, (ii) using the Matlab function conv2. Display the smoothed images.

SOLUTION:

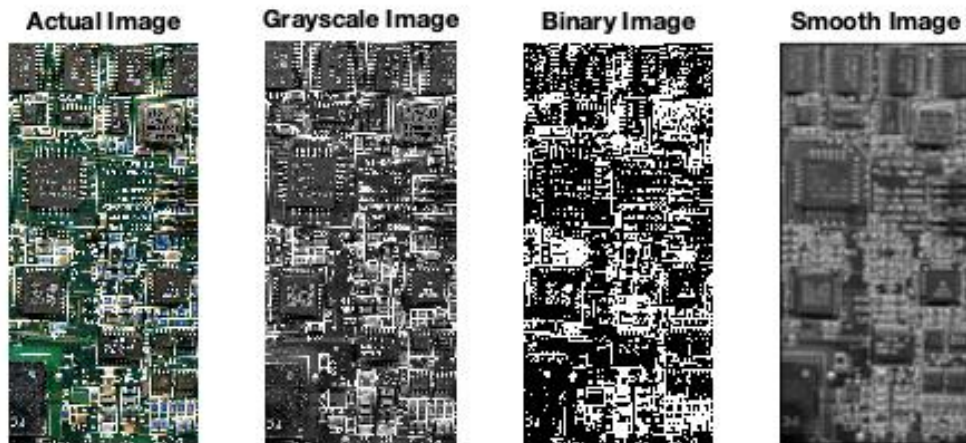1. Using nested for loops and accessing each pixel individually
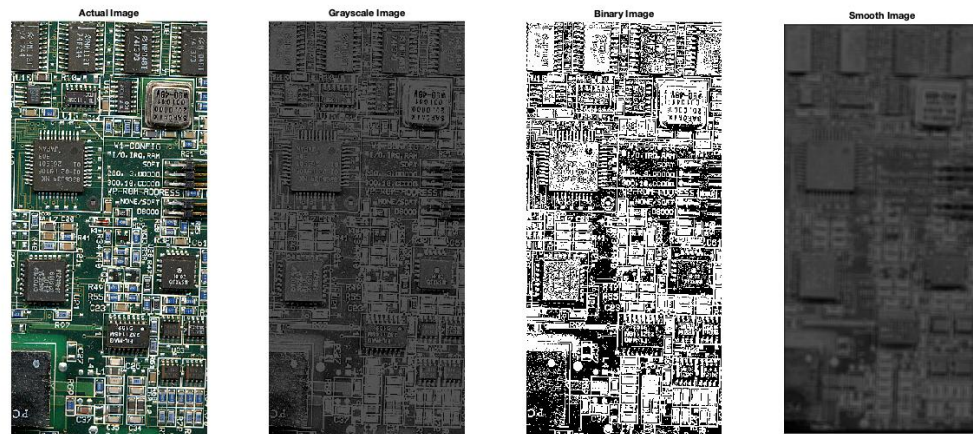
2. using MATLAB matrix operations

3. using built-in image processing functions rgb2gray and im2bw.

References:

- https://www.youtube.com/watch?v=MtI8cB6Z0ds
- https://www.mathworks.com/help/matlab/ref/plot3.html
- https://stackoverflow.com/questions/17156495/matlab-i-have-two-points-in-a-3d-plot-and-i-want-to-connect-them-with-a-line
- https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/
- http://www.songho.ca/dsp/convolution/convolution2d_example.html