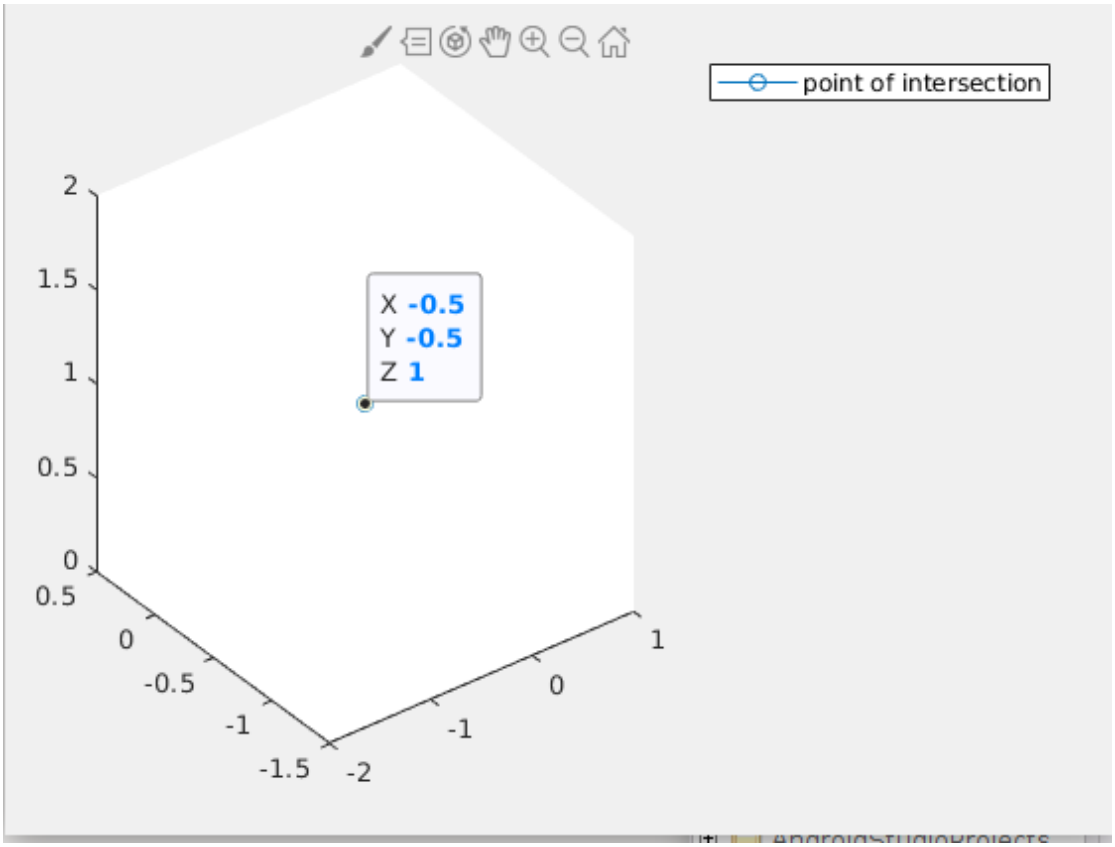
Aditya Das **axd5763**
1001675762

Problem 1

1(a) Point of Intersection

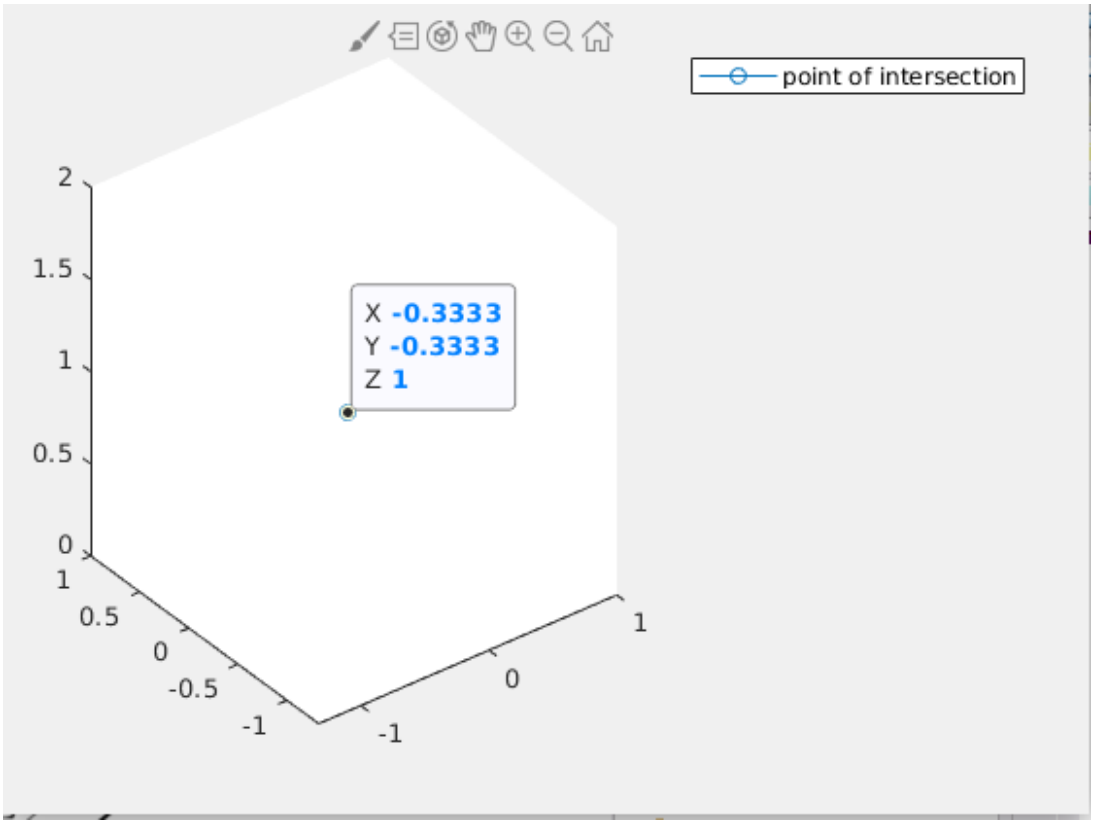



```
Q =

   -0.5000   -0.5000    1.0000

is the point of projection
```

Point of Intersection for the point(-1,-1,2)

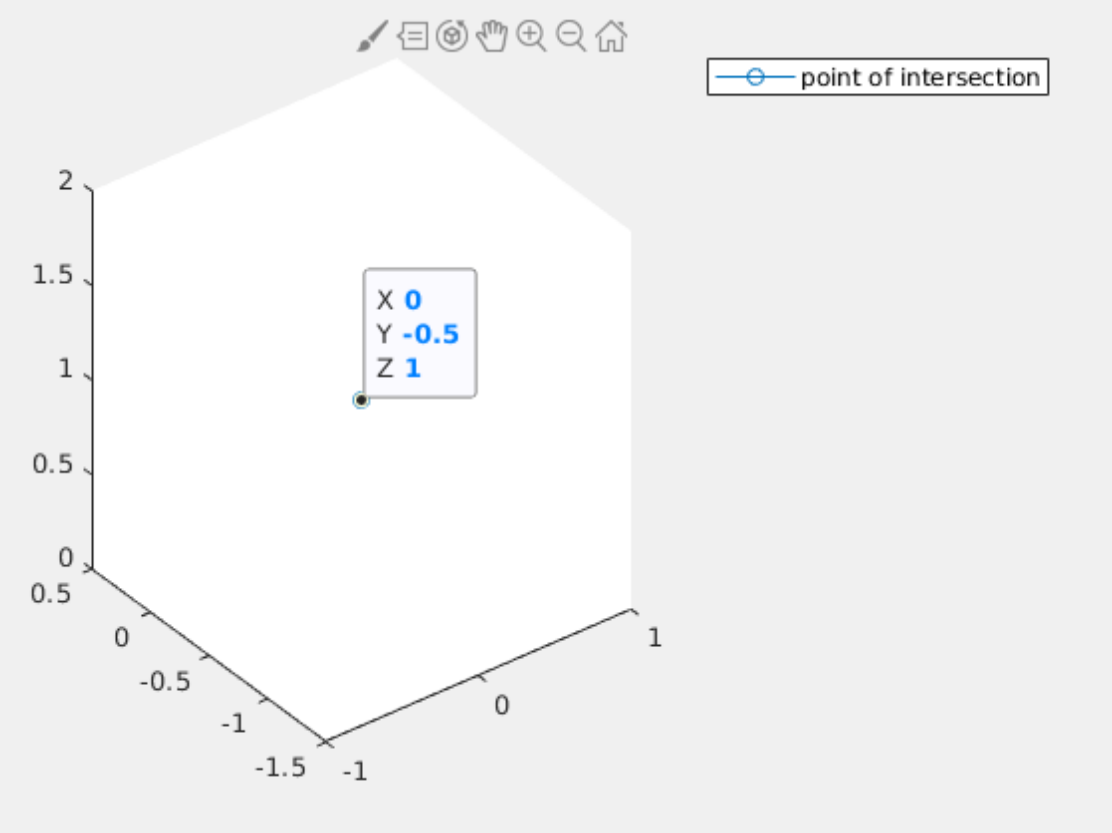



```
Q =

   -0.3333   -0.3333    1.0000

is the point of projection
```

Point of Intersection for Point (-1,-1,3)

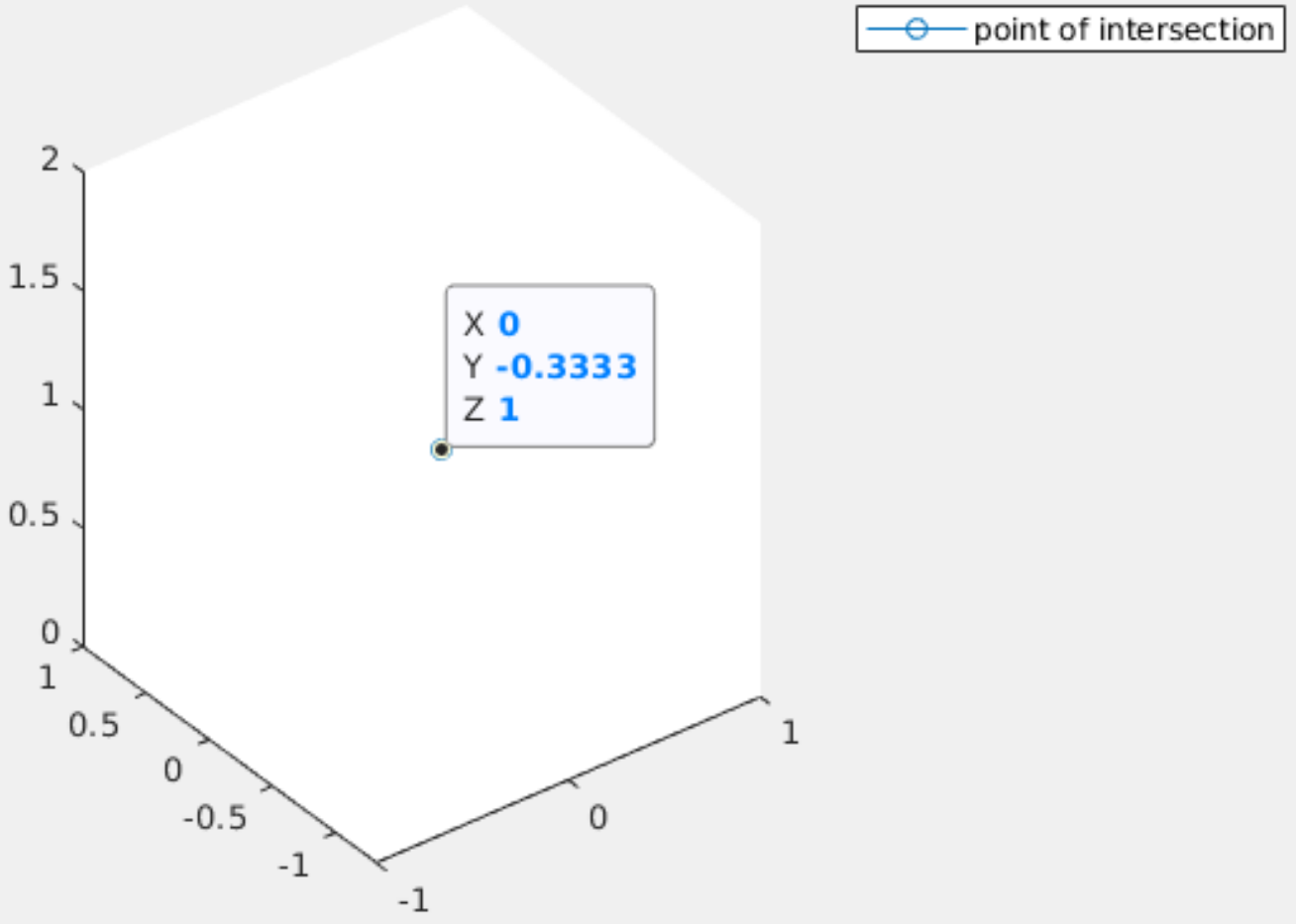



```
Q =

         0   -0.5000    1.0000

is the point of projection
```

Point of Intersection for Point (0,-1,2)

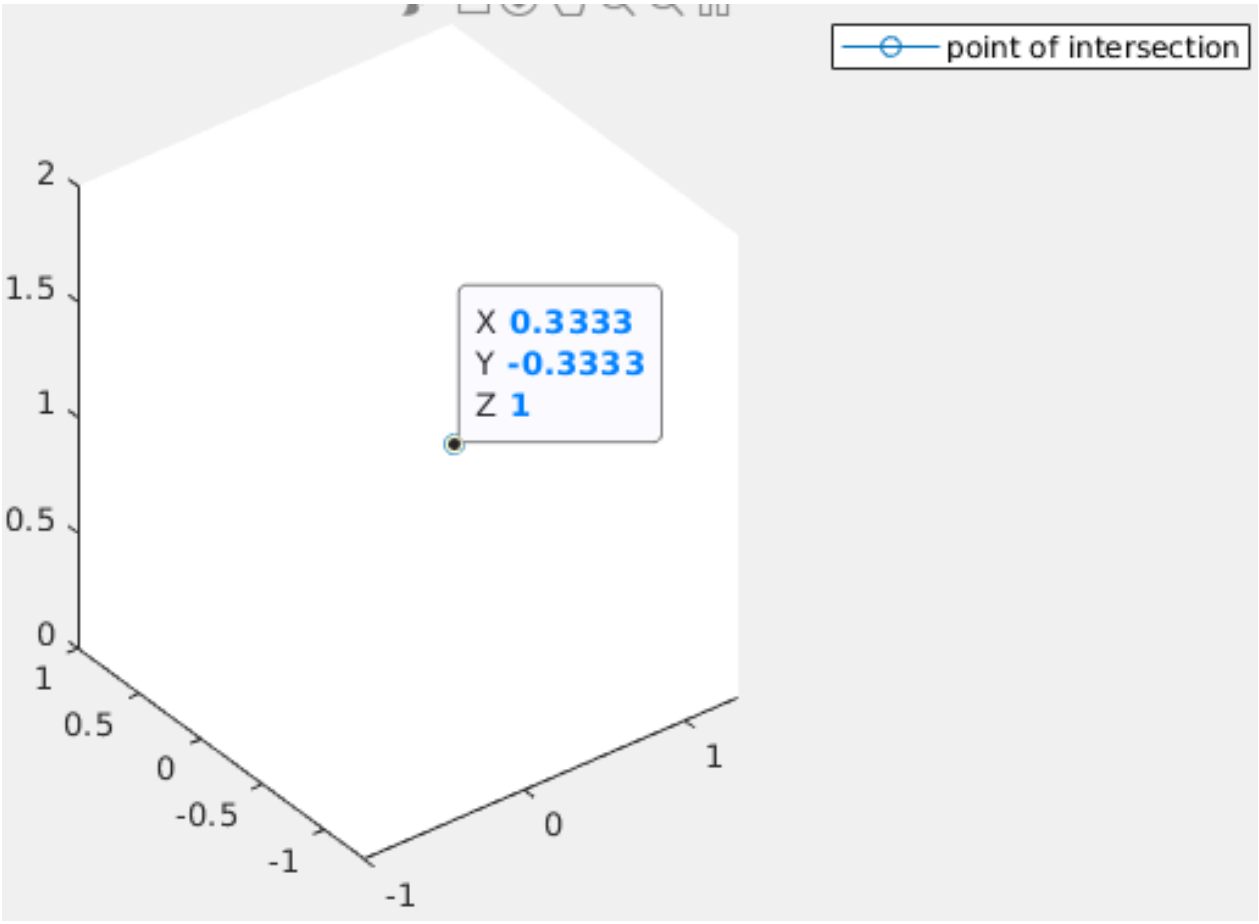



```
Q =

         0   -0.3333    1.0000

is the point of projection
```

Point of Intersection for the Point (0,-1,3)

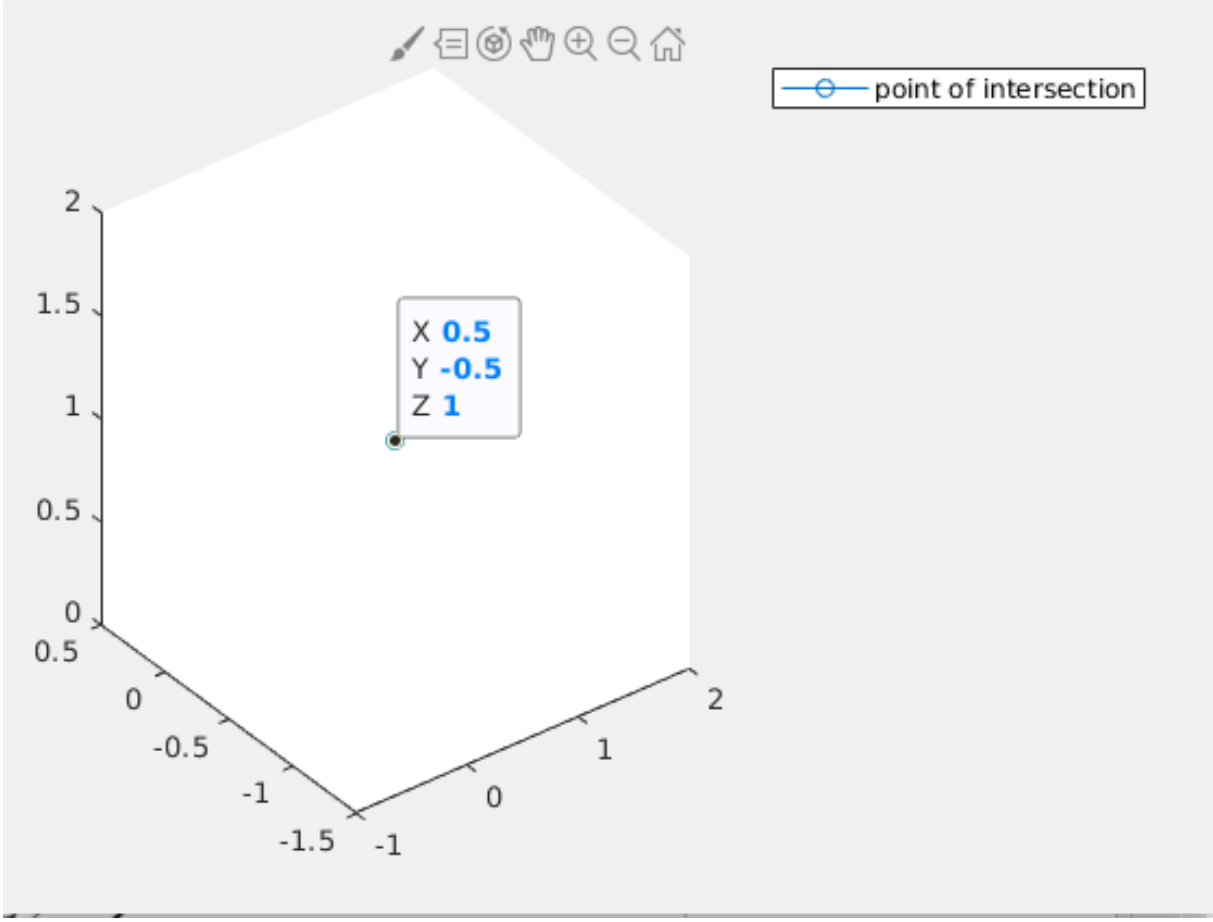



```
Q =

    0.3333   -0.3333    1.0000

is the point of projection
```

Point of Intersection for the Point (1,-1,3)



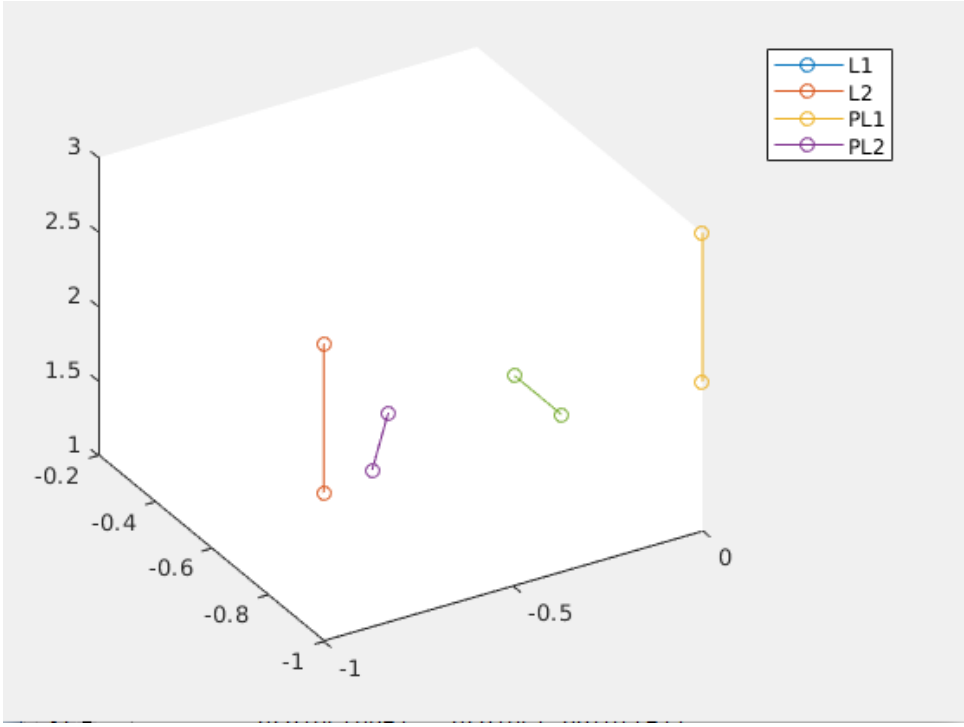

```
Q =

    0.5000   -0.5000    1.0000

is the point of projection
```

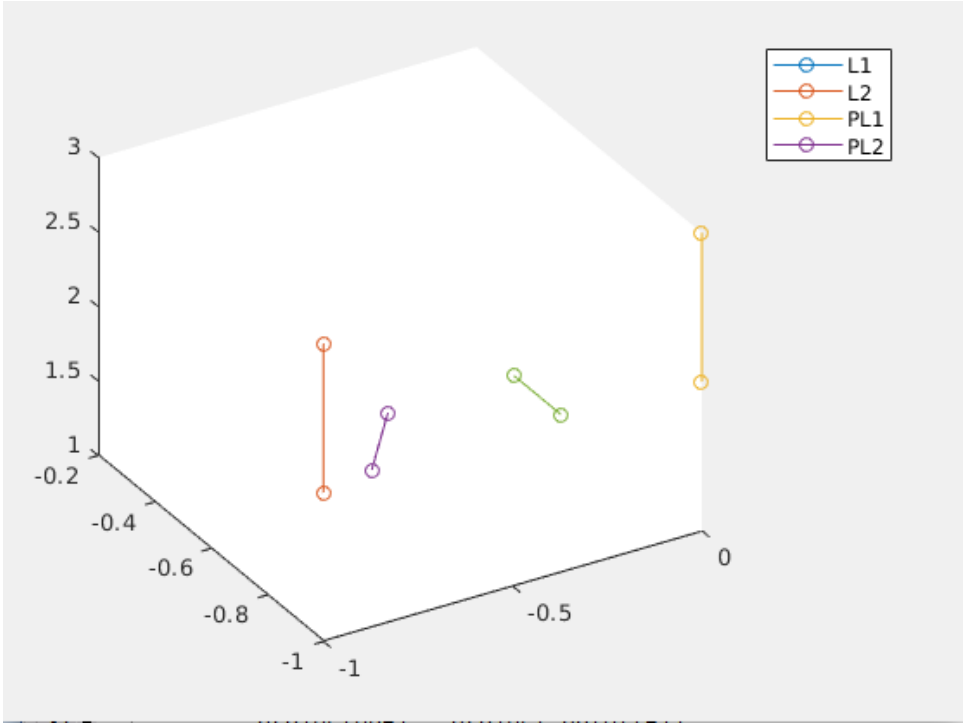
Point of Intersection for the Point (1,-1,2)

1(b)



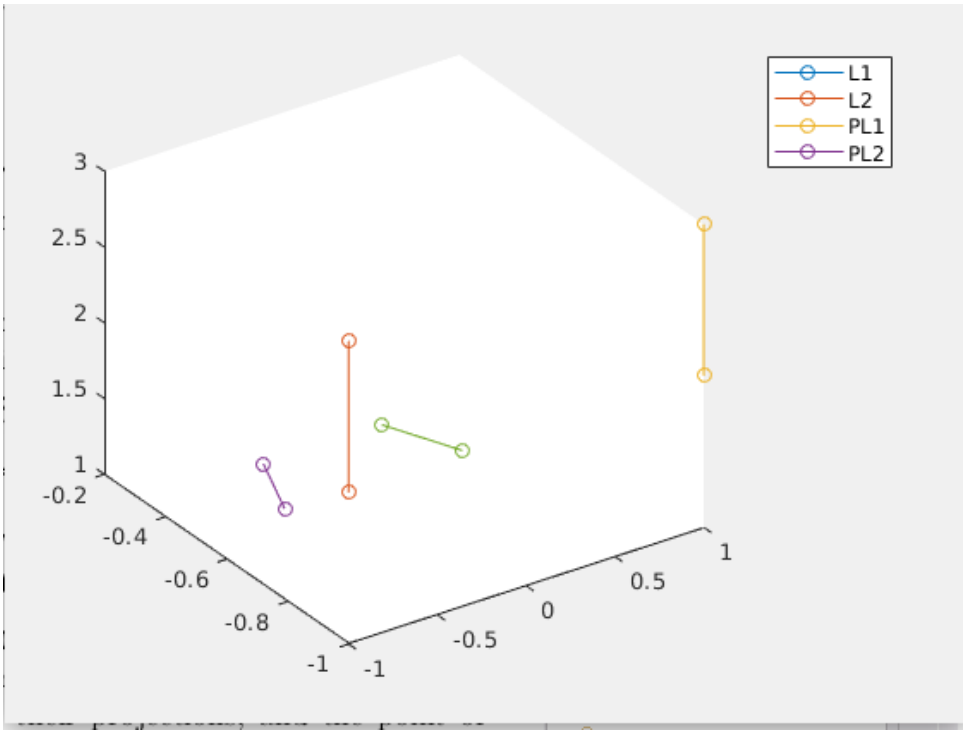

```
>> find_intersection([-1;-1;2;], [-1;-1;3;], [0;-1;2;], [0;-1;3])
lines meet at infinity
```

1(c) For lines L1, L2



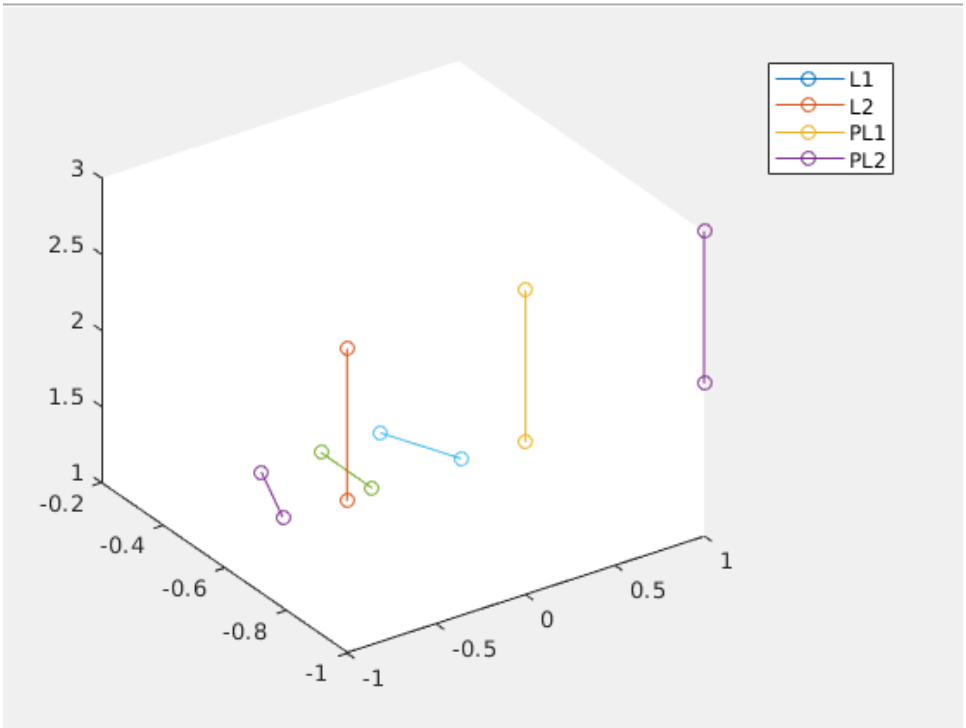

```
>> find_intersection([-1;-1;2;], [-1;-1;3;], [0;-1;2;], [0;-1;3])
lines meet at infinity
```

For Lines L1, L3




```
>> find_intersection([-1;-1;2;], [-1;-1;3;], [1;-1;2;], [1;-1;3])
lines meet at infinity
```

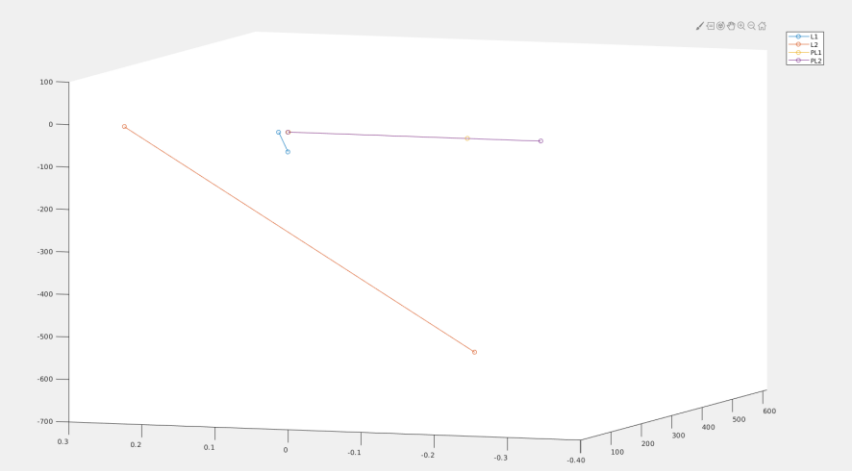
For Lines L2, L3



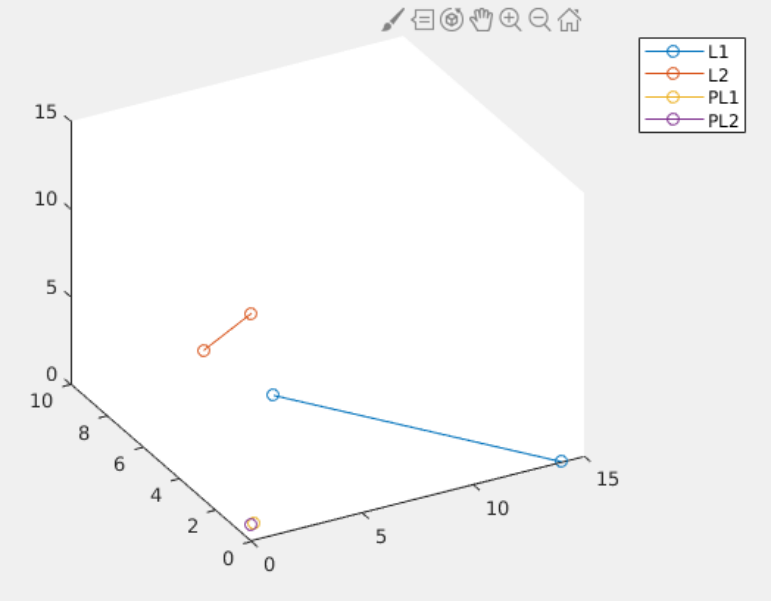

```
>> find_intersection([0;-1;2;], [0;-1;3;], [1;-1;2;], [1;-1;3])
lines meet at infinity
```

1(d)

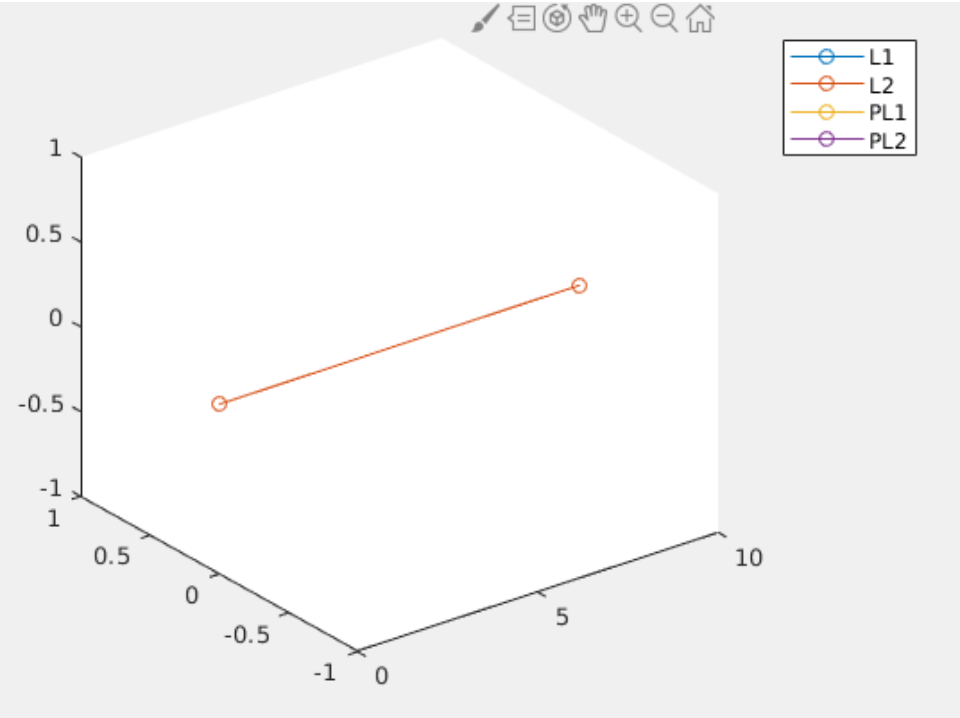
L1. Lines meet at Infnity



L2. Lines meet at Infinity



L3. Lines meet at Infinity



There is an additional file; collinear.m

The points of intersection of 3 different pairs of lines (or any 3 3d coordinates) can be given as input and the file can calculate if the points are collinear or not.

Problem 2

2(a) Extracting a rectangular block of the image given. The coordinates of the block are(200,90) and(300,180)

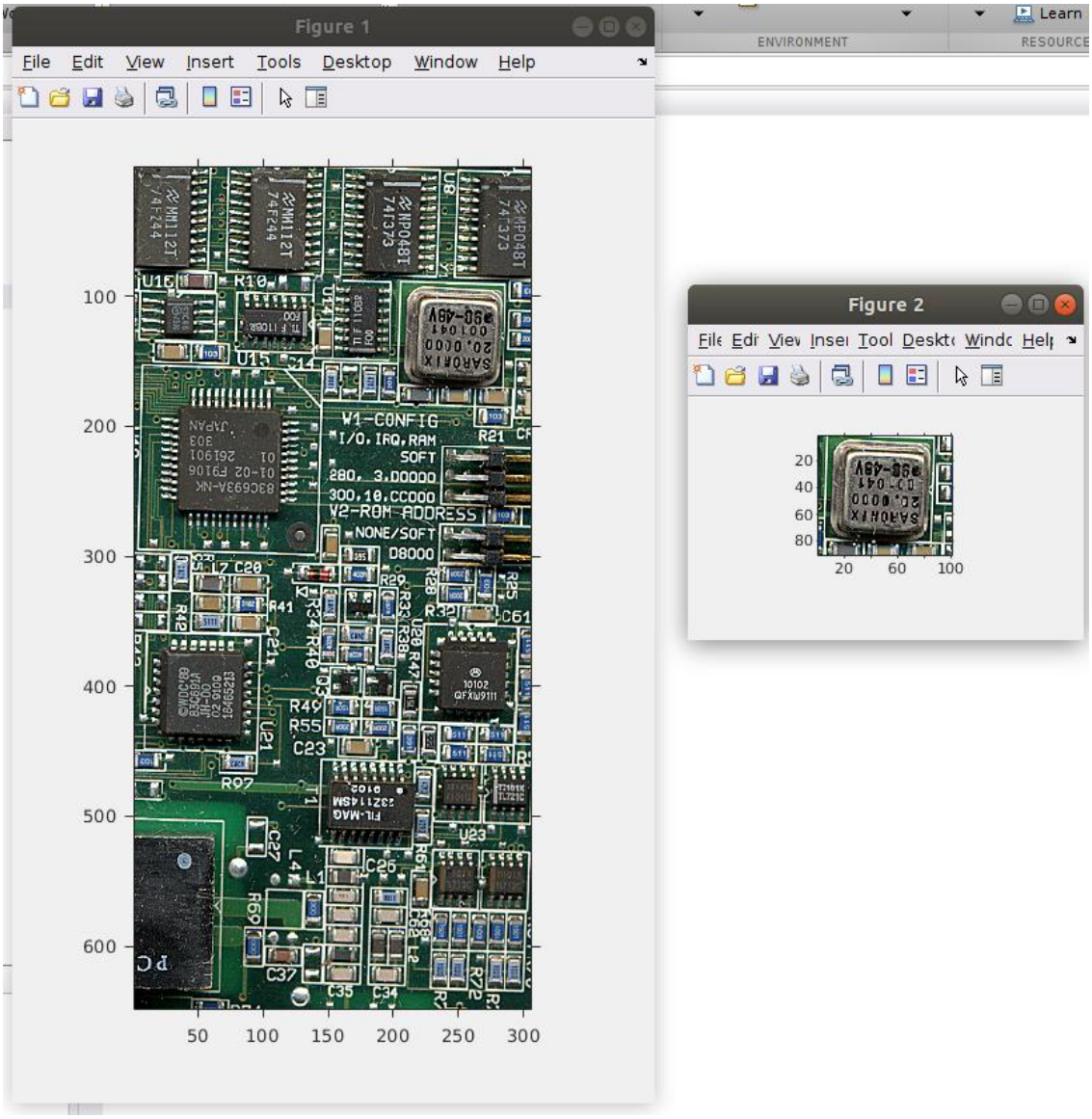


Figure 1 in the image corresponds to the loaded picture and Figure 2 is the cropped part of the picture.

2(b) Converting a RGB picture to gray scale using Nested for loops.

This image was solved in two ways.

2 (b) (i) First using the Standard NTSC conversion formulae for converting RGB to gray scale %https://en.wikipedia.org/wiki/Luma_(video).
Elapsed time is 0.415895 seconds.

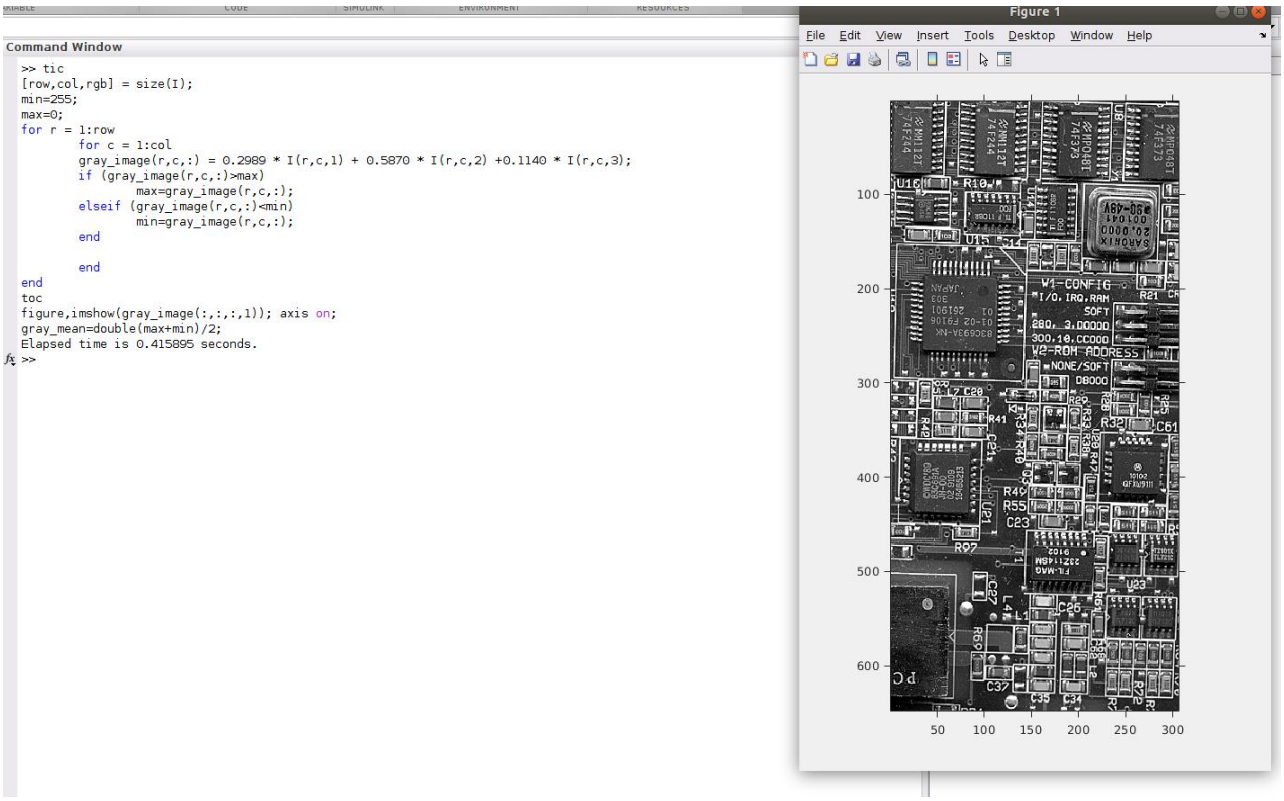


Figure 1 in the above image corresponds to the image derived from the nested for loop algorithm using Standard NTSC conversion formulae.

2 (b) (ii) Taking the average of the RGB of each pixel and multiplying it in a nested for loop. Elapsed time is 0.489756 seconds.

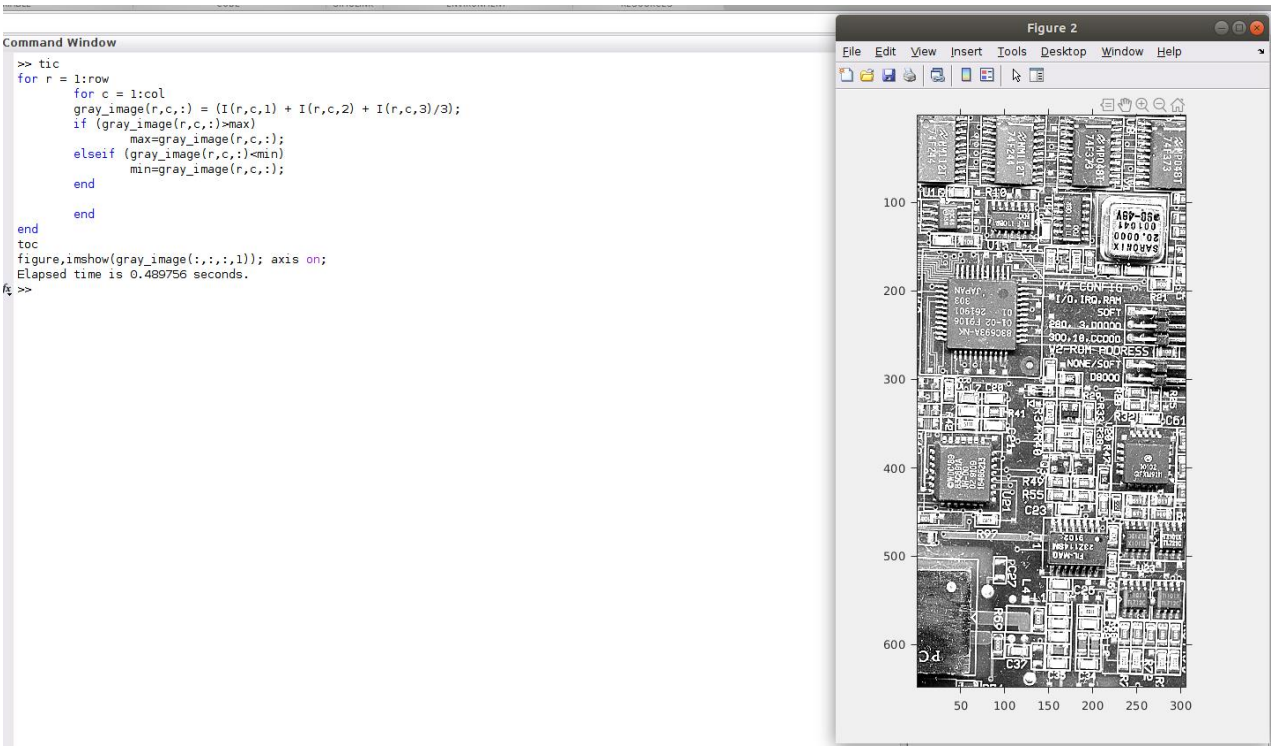


Figure 2 in the above image corresponds to the image derived from the nested for loop algorithm by taking the average of the RGB of each pixel. Elapsed time is 0.489756 seconds.

2 (b) Converting a RGB picture to gray scale using Matlab Matrix Operations

This was again done in two different ways

2 (b) (i) First using the Standard NTSC conversion formulae for converting RGB to gray scale
%https://en.wikipedia.org/wiki/Luma_(video).
Elapsed time is 0.025129 seconds

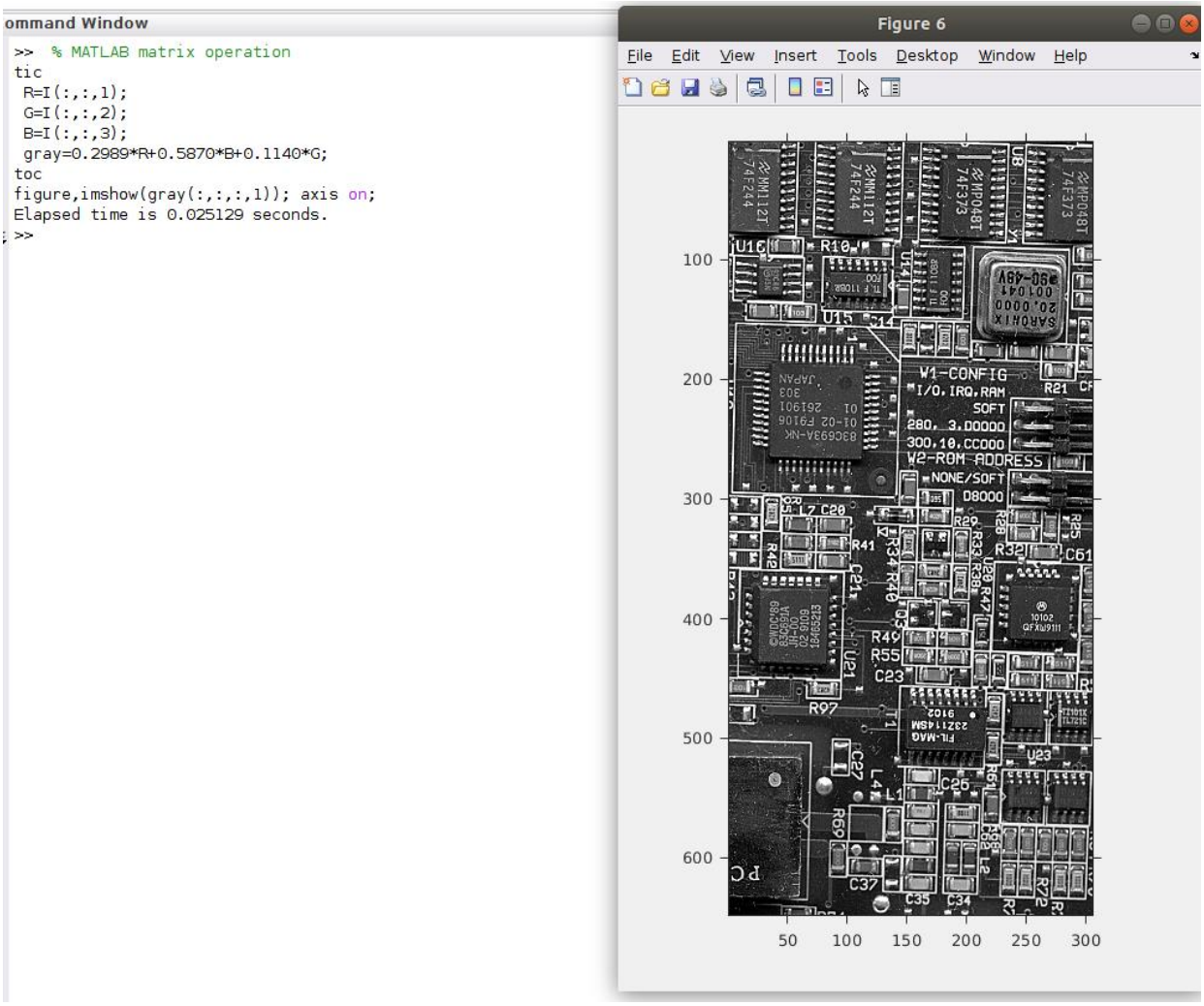


Figure 6 in the above image corresponds to the image derived using Standard NTSC conversion formulae and Matlab Matrix Operations.Elapsed time is 0.025129 seconds

2 (b) (ii) Taking the average of the RGB of each pixel . Elapsed time is 0.489756 seconds.

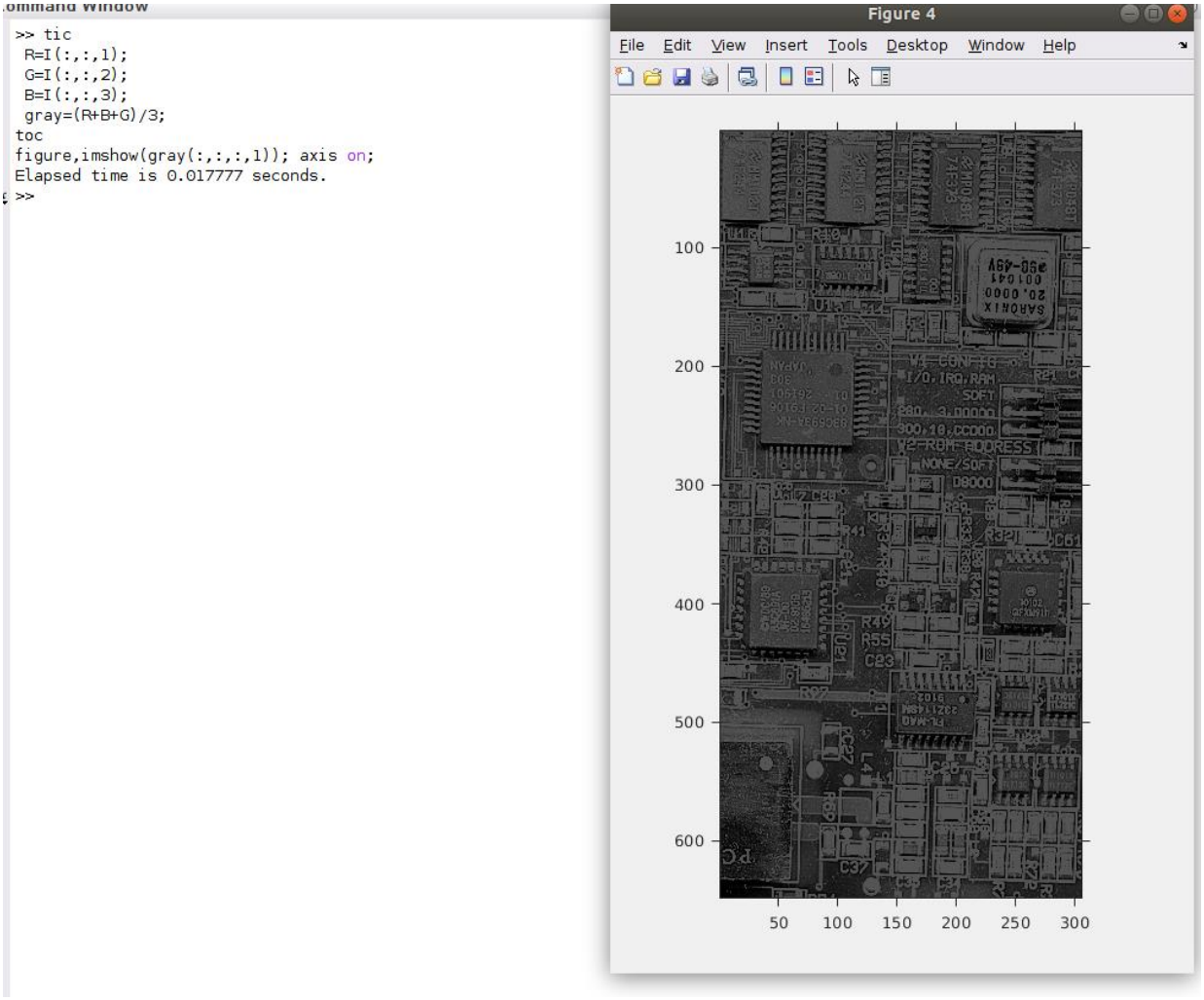


Figure 4 in the above image corresponds to the image derived using Matlab Matrix Operations and average of the RGB pixel. Elapsed time is 0.017777 seconds.

2 (b) Converting a RGB picture to gray scale using Inbuilt Image processing library

```
>> %Built-in-image processing
tic
built_grey=rgb2gray(I);
   imshow(built_grey);
toc
Elapsed time is 0.127864 seconds.
>>
```

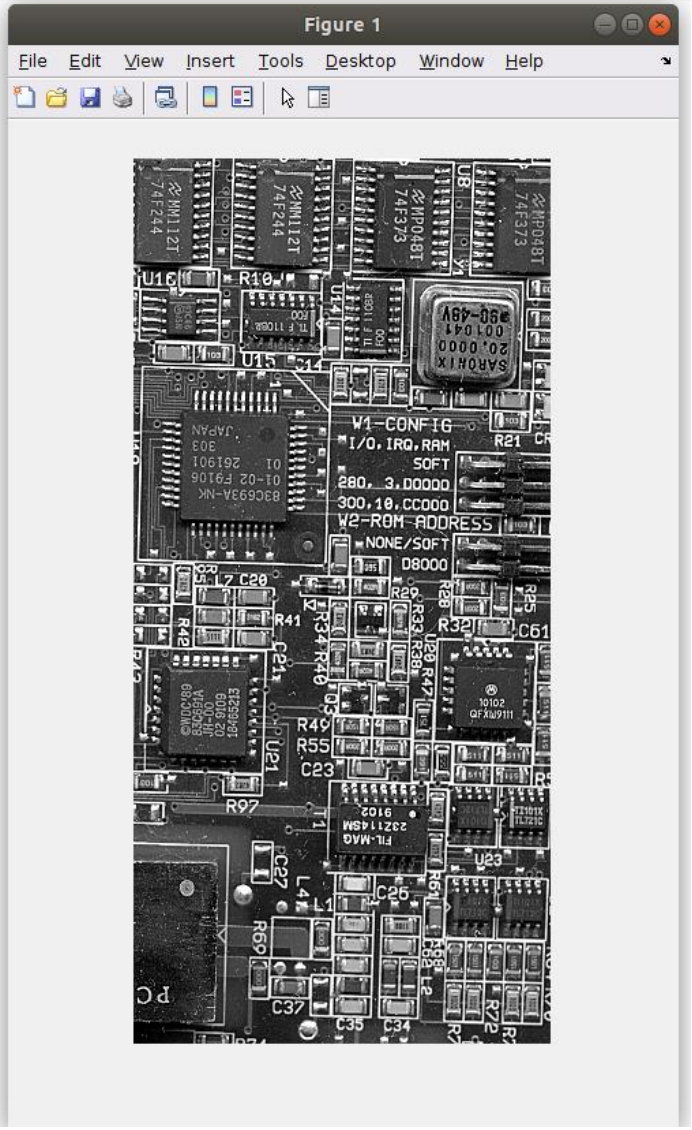


Figure 1 in the above image corresponds to the image derived using Matlab's inbuilt image processing library. Elapsed time is 0.127864 seconds.

2(c) Converting a gray scale picture to binary image using nested for loops

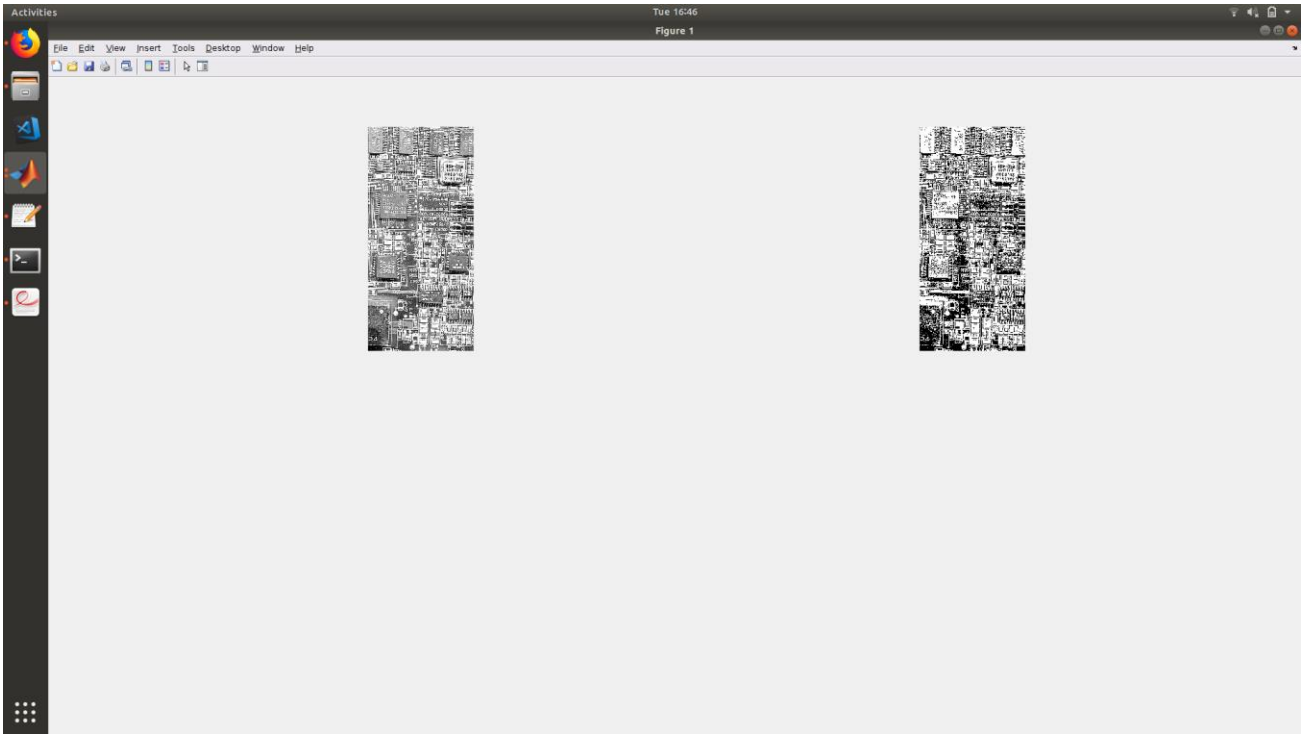


Figure 1 in the image above corresponds to the side by side comparison of a gray scale picture to a binary picture using for loop Elapsed time is 0.282838 seconds.

2 (c) Converting a gray scale picture to binary image using Matlab Matrix Operations.

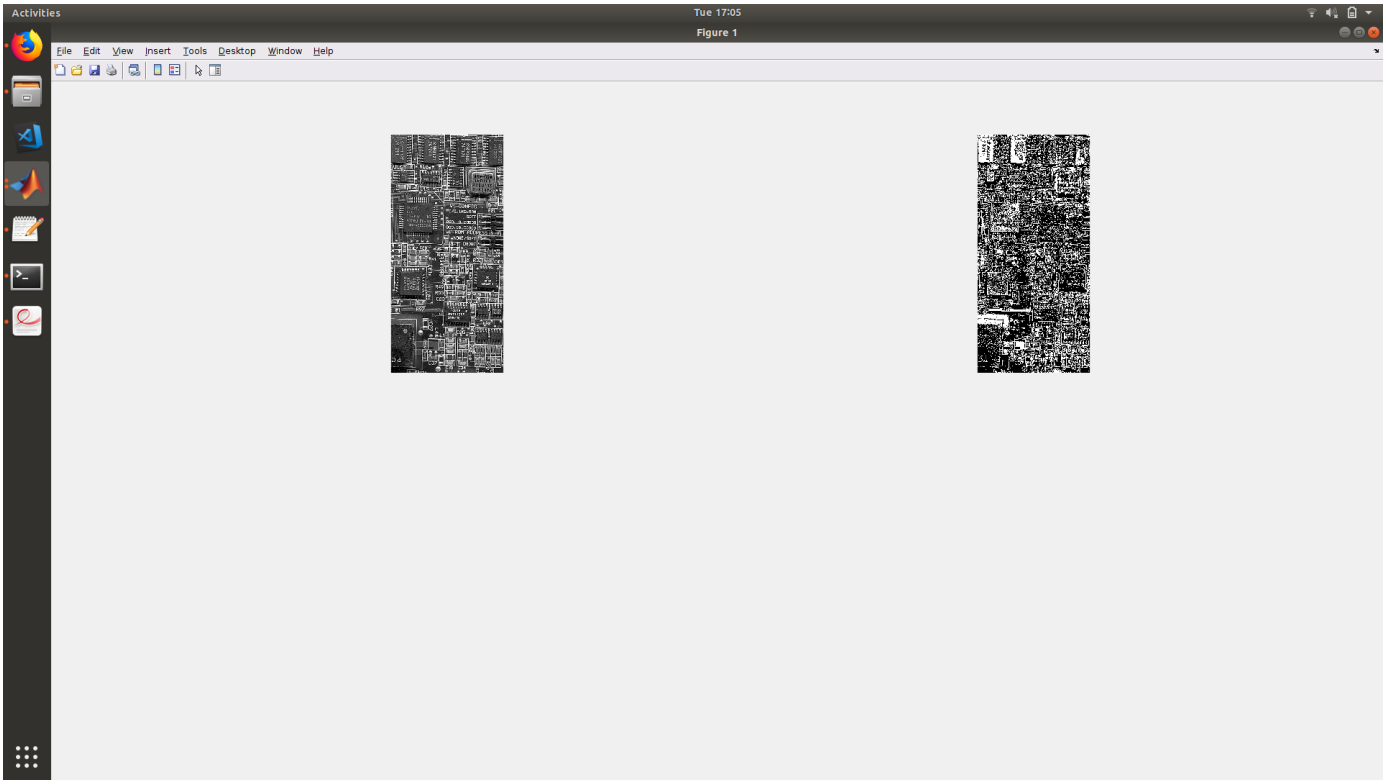
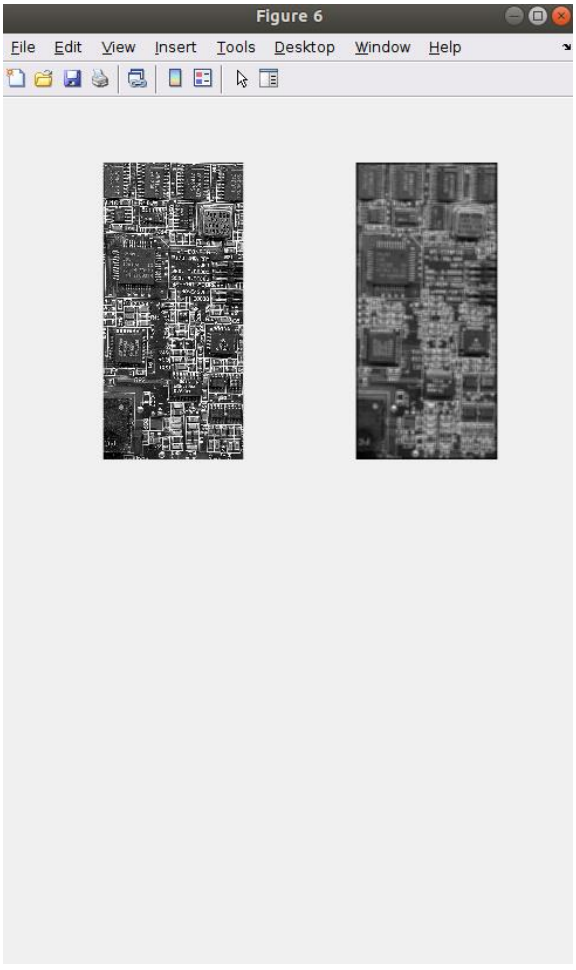


Figure 1 in the image above corresponds to to the side by side comparison of a gray scale picture to a binary picture using Matrix operations. Elapsed time is 0.131266 seconds.

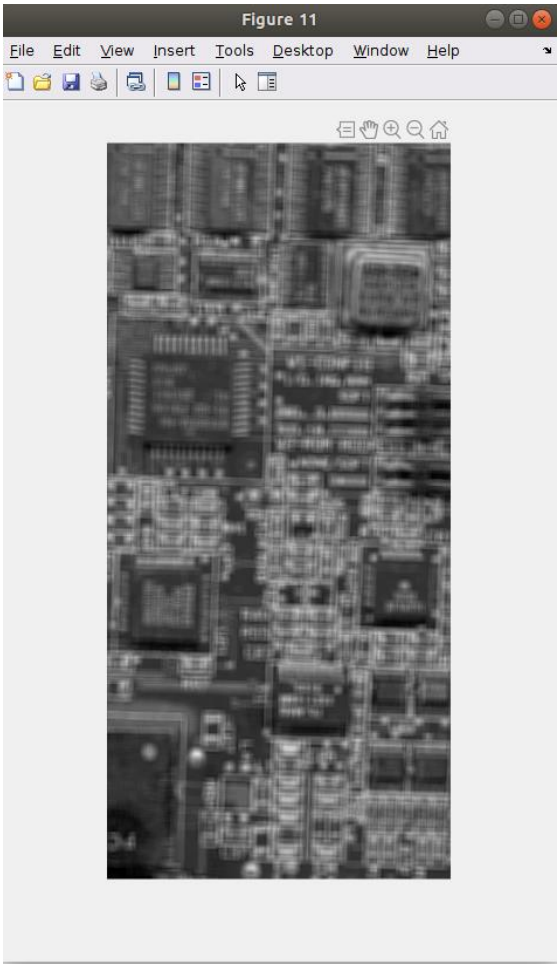
2 (d) Smoothening an image using conv2

Elapsed time is 0.135946 seconds.



2(d) Smoothening using nested for loops

Elapsed time is 0.374465 seconds.



I am un-padding the entire smoothed image and then displaying the image.

References:

https://en.wikipedia.org/wiki/NTSC

https://www.mathworks.com/help/matlab/ref/subplot.html

https://www.mathworks.com/matlabcentral/answers/122388-how-would-i-open-multiple-figures-from-one-script

https://www.mathworks.com/help/matlab/ref/randi.html

http://www.ambrsoft.com/TrigoCalc/Line3D/LineColinear.html

https://www.mathworks.com/help/images/ref/padarray.html

https://www.mathworks.com/matlabcentral/answers/427629-conv2-valid-implementation