1. lab Translation by (x, y) ; 2 DOF

$$T = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Euclidean (Rotation + Translation) ; 3 DOF

$$T = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}$$

Similarity (Scaling by (s, s) + rotation + translation) ; 4 DOF

$$\begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & \\ s\cos\theta & \end{bmatrix}$$

~~(rotation + translation)~~

$$\begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s(\cos\theta) & s(-\sin\theta) & x \\ s(\sin\theta) & s(\cos\theta) & y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine transformation (arbitrary ~~affine~~ 4-DOF linear transformations and Translation) ; 6 DOF

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

Projective Transformation (affine + projective maps) 8 DOF

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

c) Because the scale can be arbitrary so we have to remove 1 degree of freedom. Say a point (x, y), in homogeneous coordinate it is (x', y', w) where $x = \frac{x'}{w}$, $y = \frac{y'}{w}$, there is only 2 degrees of freedom as w is a scaling factor. So, $9 - 1 = 8$ DOF.

d a. $l_1 = [3, 1, 1]$, $l_2 = [-1, 0, 1]$

$\tilde{x} = l_1 \times l_2$

$$= \begin{vmatrix} i & j & k \\ 3 & 1 & 1 \\ -1 & 0 & 1 \end{vmatrix}$$

$$= (1-0)\,i - (3-(-1))\,j + (0-(-1))\,k$$

$$= i - 4j + k$$

$\tilde{x} = [1, -4, 1]$

b. $l_1 = [1, 0, 1]$ and $l_2 = [3, 0, 1]$

$\tilde{x} = l_1 \times l_2$

$$= \begin{vmatrix} i & j & k \\ 1 & 0 & 1 \\ 3 & 0 & 1 \end{vmatrix}$$

$$= 0\,i - (1-3)\,j + 0\,k$$

$$= 0\,i + 2j + 0k$$

$$= [0, 2, 0]$$

1.2a

```
>> whos
  Name          Size                 Bytes  Class    Attributes

  Image       720x1280x3            2764800  uint8
```

By using whos, I can see that the image is 720x1280x3 in size, the first dimension indicates the height (rows) of the image while the second matrix indicates the width (columns) of the image, and the third dimension is the intensity of each red, green, blue colour channel of the image. By using disp(Image(1:5,1:5,:)), I have observed the intensity of the respective colour channel are different and they each contribute different amount of their colour channel to form the final image we see.

Red

(:,:,1) =

67	68	72	75	74
61	62	64	68	70
60	59	60	62	65
61	60	60	61	62
61	60	61	62	62

Green

(:,:,2) =

70	71	74	77	76
64	65	67	70	72
63	62	63	64	65
64	63	63	64	63
64	63	62	63	63

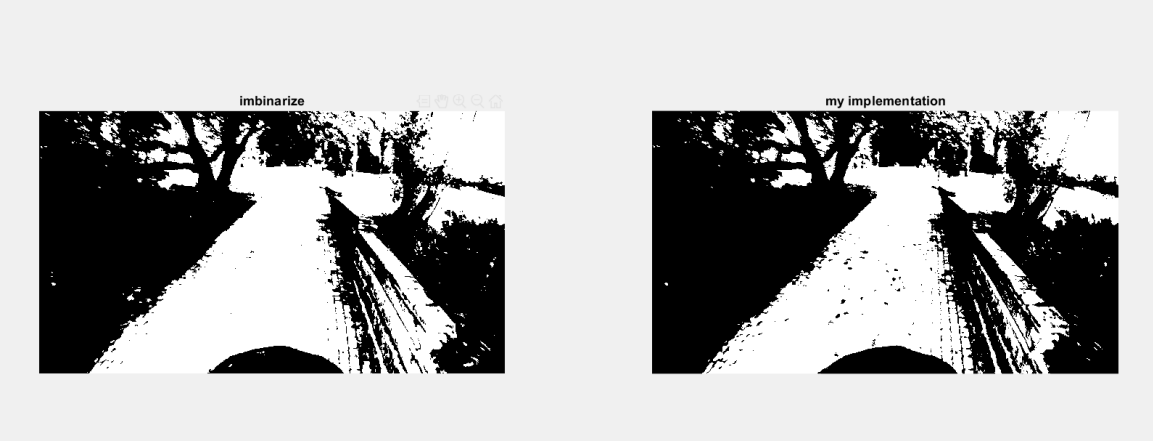
Blue

(:,:,3) =

61	62	69	72	75
55	56	60	65	69
54	53	56	59	63
55	54	56	57	58

1.2b

Using rgb2gray function, it converts the image to grayscale by eliminating the hue and saturation information. The matrix of the image then becomes (720x1280),losing the third dimension which indicates that the **matrix** only **retains the effective luminance of each colour channel**. Same thing can also be achieved if we just split up each of the RGB channel, in actuality it will just be an image of the same size (720x1280) with the intensity of each colour channel, to achieve the grayscale image we have to convert the colour image to grayscale by multiplying the respective colour channels with the following formula grayscaleImage = (0.3 * R) + (0.59 * G) + (0.11 * B).



Binary is achieved using im2binary(grayImage), the full matrix is shown by disp(binaryImage(:,:)), the matrix becomes a combination of two values (0 and 1) with 0 indicating it is a black colour and 1 is white colour. The change is done by selecting value over a certain threshold to be white colour and black colour otherwise. I have also implemented it on myself by changing the value of the grayImage to 255 if its actual value is over 100 and 0 otherwise.

```
binaryImage2 = grayImage;

binaryImage2(grayImage>=100) = 256;

binaryImage2(grayImage<100) = 0;
```

```
>> disp(binaryImage(1:5,1:5))
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

1.2c Indexed colour assign an index to each location of the image, each index addresses a row in the colourmap to display a colour at the specified location in the graphic. For an example, in matlab C is an m-by-n array that is the same size as the z-coordinate array. The value at C(i,j) specifies the colormap index for Z(i,j) to get the colour of the image. While truecolour is an m-by-n-by-3 array in which each row specifies an RGB triplet at every location of the image. For an example, in matlab C is an m-by-n-by-3 array, where C(:,:,i) the same size as the z-coordinate array. C(i,j,1) specifies the red component for Z(i,j). Hence, if the image CData is two-dimensional, then the CData values are treated as lookup indices into the colour map.

Matlab code

%Copy and paste each section separately in the command window to see the result

%Section 1 Reading an image

```
Image = imread('GOPR1515 06102.jpg');

imshow(Image);

disp(Image(1:5,1:5,:));

%disp(Image(:,:,:)); %<- use this to see all of the matrix information


%whos  %<- To see the information about the image
```

%Section 2 Grayscale

```
grayImage = rgb2gray(Image);
```

%%My implementation

% Splitting

```
R = 0.2989 *Image(:,:,1);

G = 0.5870 *Image(:,:,2);
```

```
B = 0.1140*Image(:,:,3);

figure;
subplot(1,3,1);
imshow(R);
title('Red Channel');

subplot(1,3,2);
imshow(G);
title('Green Channel');

subplot(1,3,3);
imshow(B);
title('Blue Channel');

grayImage2 = R+G+B;
figure;
subplot(1,2,1);
imshow(grayImage)
title('rgb2gray');
subplot(1,2,2);
imshow(grayImage2)
title('my implementation');

% Section 3 Binary
binaryImage = imbinarize(grayImage);
imshow(binaryImage)
disp(binaryImage(1:5,1:5))
%My implementation
binaryImage2 = grayImage;
binaryImage2(grayImage>=100) = 256;
```

```
binaryImage2(grayImage<100) = 0;


figure;
subplot(1,2,1);


imshow(binaryImage)
title('imbinarize');
subplot(1,2,2);
imshow(binaryImage2)
title('my implementation');
```

$$C^TC = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 10 & 14 \\ 14 & 20 \end{pmatrix}$$

$$\det(C^TC - \lambda I) = \begin{vmatrix} 10-\lambda & 14 \\ 14 & 20-\lambda \end{vmatrix}$$

$$= (10-\lambda)(20-\lambda) - 196$$
$$= \lambda^2 - 30\lambda + 40 = 0$$
$$\lambda = 15 \pm \sqrt{221}$$

When $\lambda = 15+\sqrt{221}$

We get $-9.886x + 14y = 0$

$14x - 9.886y = 0$

Let $x = -1$

$y = -1.4204$

$x = \frac{-1}{\sqrt{1^2+1.4204^2}}$

$= -0.576$

$y = \frac{-1.4204}{\sqrt{1^2+1.4204^2}}$

$= -0.8174$

$\lambda = 15-\sqrt{221}$

$9.866x + 14y = 0$

$14x + 19.866y = 0$

let $x = 1$, ~~$y = -0.707$~~

$y = -0.7047$

$x = \frac{1}{\sqrt{1+0.7047^2}}$

$= 0.8174$

$y = \frac{-0.7047}{\sqrt{1+0.7047^2}}$

$= -0.576$

$$V = \begin{pmatrix} -0.576 & 0.8174 \\ -0.8174 & -0.576 \end{pmatrix}$$

$$S = \begin{pmatrix} \sqrt{15+\sqrt{221}} & 0 \\ 0 & \sqrt{15-\sqrt{221}} \end{pmatrix} = \begin{pmatrix} 5.465 & 0 \\ 0 & 0.366 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\begin{pmatrix} -0.576 & 0.8174 \\ -0.8174 & -0.576 \end{pmatrix}$$

$$= \begin{pmatrix} -2.2108 & -0.3346 \\ -4.9476 & 0.1482 \end{pmatrix}$$

Normalised

$$U = \begin{pmatrix} -0.4046 & -0.9145 \\ -0.9145 & 0.4046 \end{pmatrix}$$

Answers shown