"""

1. Show what happens when we apply a binary mask on a grayscale image.
2. Slice an 8-bit grayscale image into 8 planes.
3. Show the effect of convolution of a grayscale image with a Laplacian filters and sobel filters.

"""

In [1]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.util import img_as_float
```

In [2]:

```
def plt_img(img_set,img_title):
    ch = len(img_set)
    plt.figure(figsize=(20,20))
    for i in range(ch):
        plt.subplot(2,3,i+1)
        ln = len(img_set[i].shape)
        if ln == 3:
            plt.imshow(img_set[i])
        else:
            plt.imshow(img_set[i],cmap='gray')
        plt.title(img_title[i])
    plt.show()
```

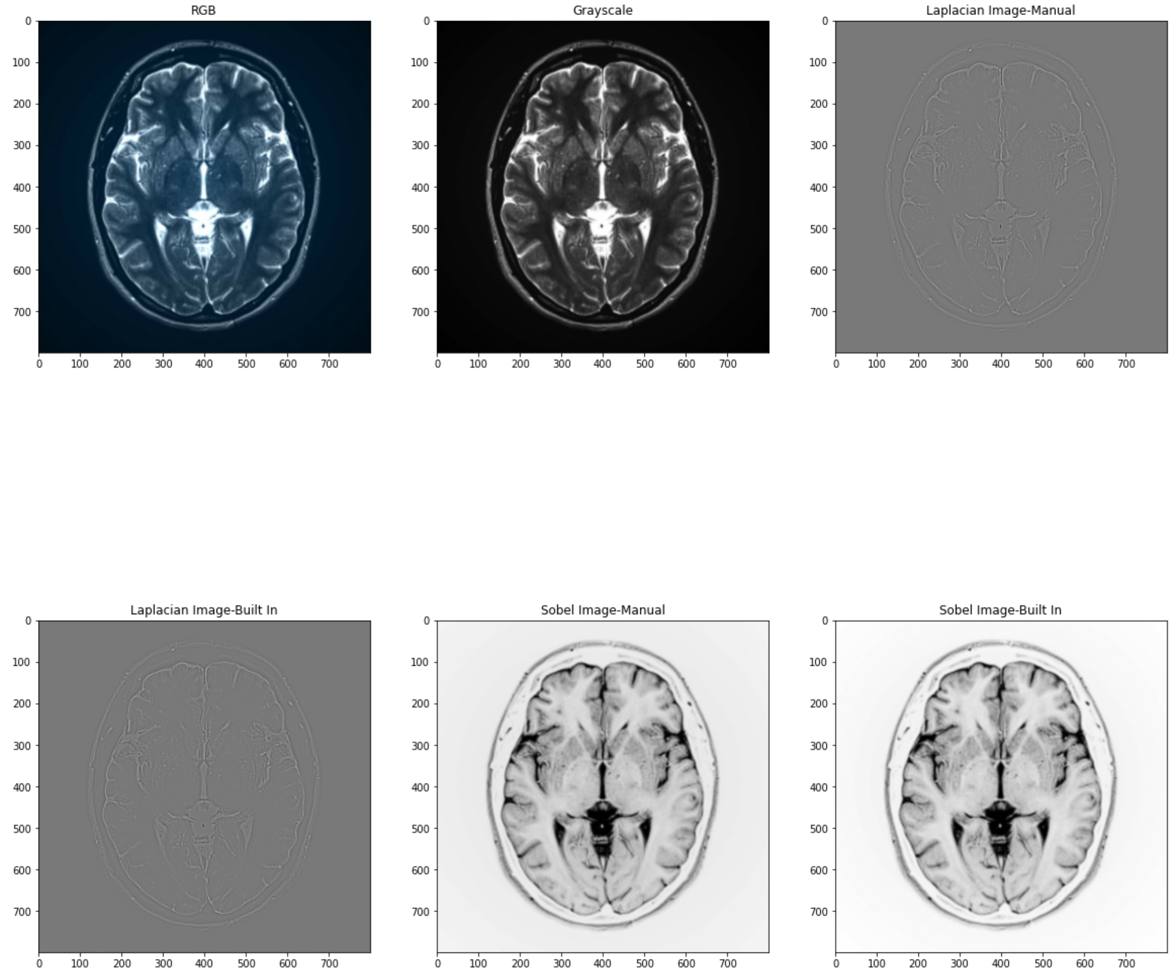
In [3]:

```
def convolution(kernel,grayscale):
    x, y = grayscale.shape
    print(x,y)
    kx,ky = kernel.shape
    print(kx,ky)
    r = x + kx - 1
    c = y + ky - 1
    padded_img = np.zeros((r,c),dtype=np.float32)

    """Zero padding the origianl image"""

    for i in range(x):
        for j in range(y):
            padded_img[i+(kx-1)//2,j+(ky-1)//2] = grayscale[i,j]

    processed_img = np.zeros((x,y),dtype=np.float32)
    for i in range(r):
        for j in range(c):
            for k in range(kx):
                for l in range(ky):
                    if i < x and i >= 0 and j < y and j >= 0:
                        processed_img[i,j] += kernel[k,l] * padded_img[i
                        if(processed_img[i,j]) >= 256:
                            processed_img[i,j] = 255
    return processed_img
```

In [5]:

```
def main():

    rgbImg = plt.imread('mri1.jpg')
    print(rgbImg.shape)

    grayscale = cv2.cvtColor(rgbImg,cv2.COLOR_RGB2GRAY)
    grayscale = img_as_float(grayscale)
    print(grayscale.shape)
    x,y = grayscale.shape

    LaplacianKernel = np.array([
                                [-1,-1,-1],
                                [-1,8,-1],
                                [-1,-1,-1],
                                ])
    print("Laplacian Kernel : {}".format(LaplacianKernel))
    SobelKernel = np.array([
                                [-1,0,-1],
                                [-2,0,-2],
                                [-1,0,-1],
                                ])

    print("Sobel Kernel : {}".format(SobelKernel))

    image1 = convolution(LaplacianKernel,grayscale)
    image2 = cv2.filter2D(grayscale,-1,LaplacianKernel)
    image3 = convolution(SobelKernel,grayscale)
    image4 = cv2.filter2D(grayscale,-1,SobelKernel)

    img_set = [rgbImg,grayscale,image1,image2,image3,image4]
    img_title = ['RGB','Grayscale','Laplacian Image-Manual','Laplacian I

    plt_img(img_set,img_title)


if __name__ == '__main__':
    main()
```

```
(800, 800, 3)
(800, 800)
Laplacian Kernel : [[-1 -1 -1]
 [-1  8 -1]
 [-1 -1 -1]]
Sobel Kernel : [[-1  0 -1]
 [-2  0 -2]
 [-1  0 -1]]
800 800
3 3
800 800
3 3
```


RGB
Grayscale
Laplacian Image-Manual
Laplacian Image-Built In
Sobel Image-Manual
Sobel Image-Built In
0
100
200
300
400
500
600
700

In [ ]: