Experiment 2

Aim: Use diffrence of gaussian to find edges for the given image. Change parameters of gaussian filter and observe the effect. Apply the same technique on other image.

Name: Rishikeshh Vadodaria

Roll no: C114

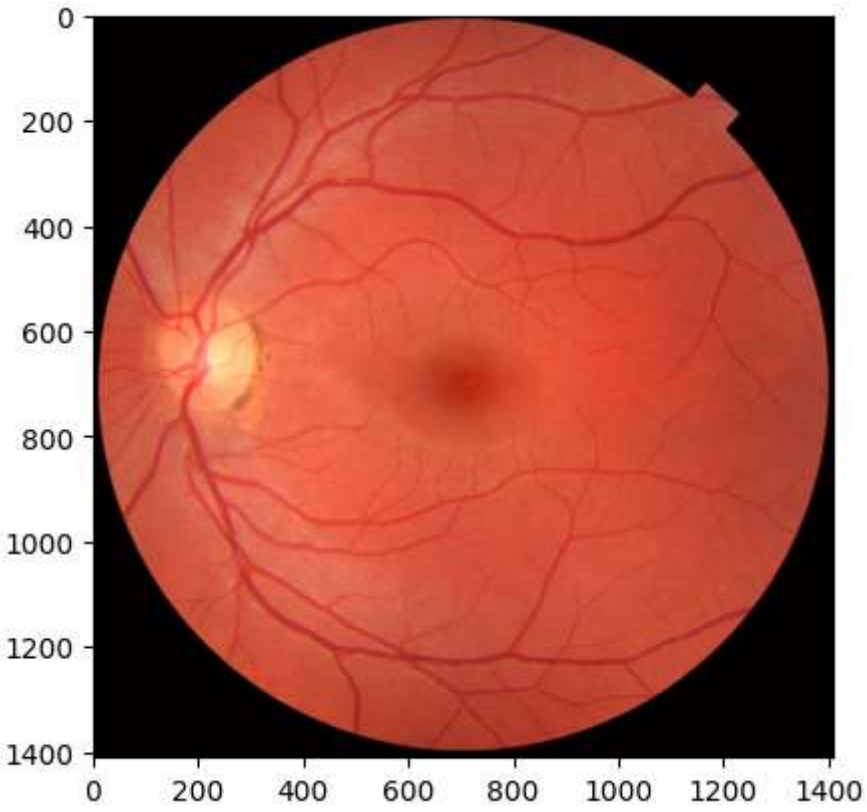
Task 1: Convert image to gray

In [1]:
```
import cv2
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage.color import rgb2gray
```

In [2]:
```
image=data.retina()
```

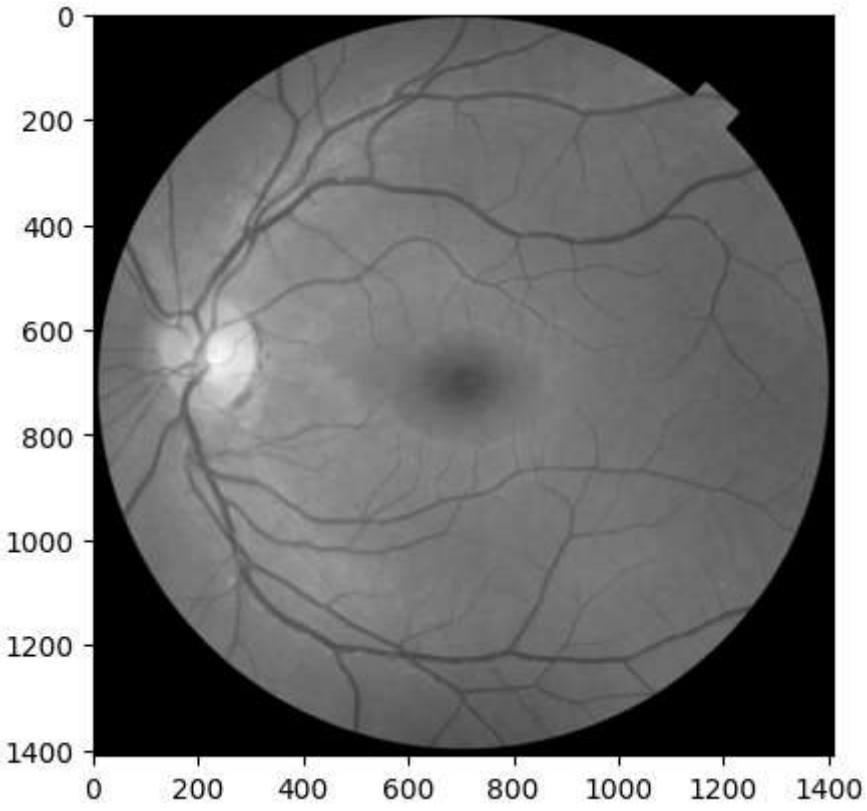
In [3]:
```
plt.imshow(image)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x1ff7fad3290>
```



In [4]:
```
image_gray=rgb2gray(image)
plt.imshow(image_gray, cmap='gray')
```

Out[4]: <matplotlib.image.AxesImage at 0x1ff1a730b90>

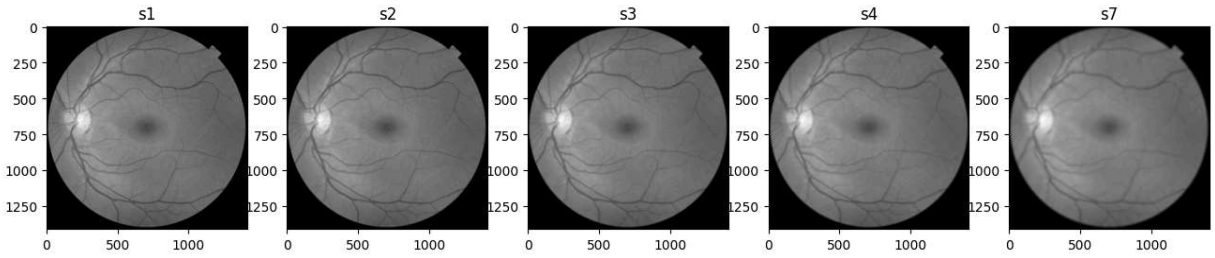


Task 2: Blur it using Gaussian filters of different values of standard deviations

In [5]:
```
image_blur_s1=cv2.GaussianBlur(image_gray,(31,31),1)
image_blur_s2=cv2.GaussianBlur(image_gray,(31,31),2)
image_blur_s3=cv2.GaussianBlur(image_gray,(31,31),3)
image_blur_s4=cv2.GaussianBlur(image_gray,(31,31),4)
image_blur_s7=cv2.GaussianBlur(image_gray,(31,31),7)
```

In [6]:
```
plt.figure(figsize=(16,10))
plt.subplot(1, 5, 1)
plt.imshow(image_blur_s1, cmap='gray')
plt.title('s1')

plt.subplot(1, 5, 2)
plt.imshow(image_blur_s2, cmap='gray')
plt.title('s2')

plt.subplot(1, 5, 3)
plt.imshow(image_blur_s3, cmap='gray')
plt.title('s3')

plt.subplot(1, 5, 4)
plt.imshow(image_blur_s4, cmap='gray')
plt.title('s4')

plt.subplot(1, 5, 5)
plt.imshow(image_blur_s7, cmap='gray')
plt.title('s7')
```

Out[6]:
```
Text(0.5, 1.0, 's7')
```



Task 3: Determine Difference of Gaussians for each combination

In [7]:
```
DoG1_7=image_blur_s1-image_blur_s7
DoG2_7=image_blur_s2-image_blur_s7
DoG3_7=image_blur_s3-image_blur_s7
DoG4_7=image_blur_s4-image_blur_s7
```

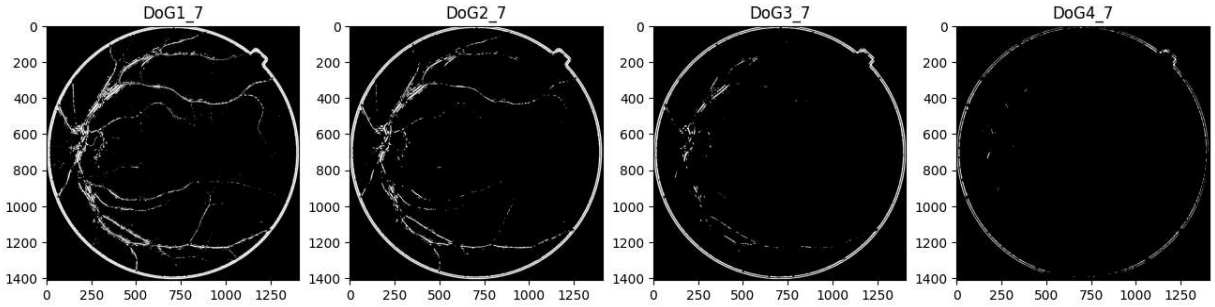
In [8]:
```
threshold=np.max(DoG1_7)*0.2
```

In [9]:
```
DoG1_7[np.abs(DoG1_7)>threshold]=255
DoG2_7[np.abs(DoG2_7)>threshold]=255
DoG3_7[np.abs(DoG3_7)>threshold]=255
DoG4_7[np.abs(DoG4_7)>threshold]=255
```

Task 4: Observe the effect for different combinations of standard deviations

In [10]:
```
plt.figure(figsize=(16,10))
plt.subplot(1, 4, 1)
plt.imshow(DoG1_7, cmap='gray')
plt.title('DoG1_7')

plt.subplot(1, 4, 2)
plt.imshow(DoG2_7, cmap='gray')
plt.title('DoG2_7')

plt.subplot(1, 4, 3)
plt.imshow(DoG3_7, cmap='gray')
plt.title('DoG3_7')

plt.subplot(1, 4, 4)
plt.imshow(DoG4_7, cmap='gray')
plt.title('DoG4_7')
```

Out[10]:
```
Text(0.5, 1.0, 'DoG4_7')
```

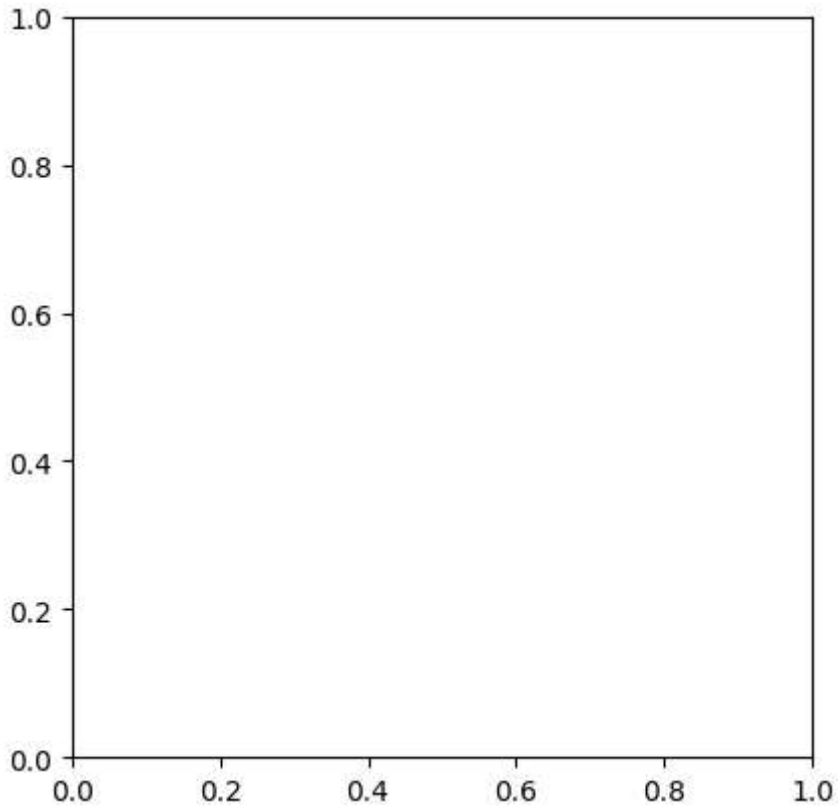


Task 5: Repeat the same for any other image

In [11]:
```
image=cv2.imread('1.jpg')
plt.imshow(image)
```

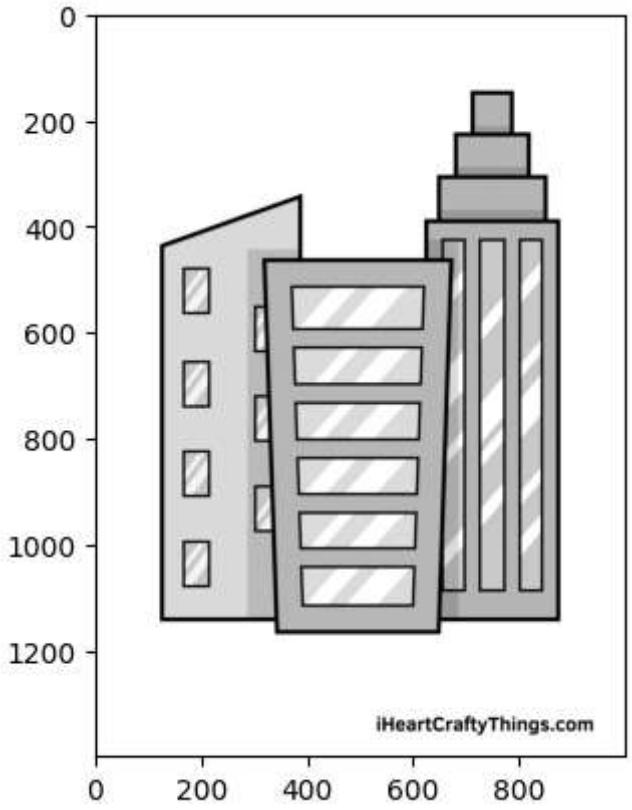
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 image=cv2.imread('1.jpg')
----> 2 plt.imshow(image)

File c:\Users\Rishikesh\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\pyplot.py:3592, in imshow(X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, colorizer, origin, extent, interpolation_stage, filternorm, filterrad, resample, url, data, **kwargs)
   3570 @_copy_docstring_and_deprecators(Axes.imshow)
   3571 def imshow(
   3572     X: ArrayLike | PIL.Image.Image,
   (...)
   3590     **kwargs,
   3591 ) -> AxesImage:
-> 3592     __ret = gca().imshow(
   3593         X,
   3594         cmap=cmap,
   3595         norm=norm,
   3596         aspect=aspect,
   3597         interpolation=interpolation,
   3598         alpha=alpha,
   3599         vmin=vmin,
   3600         vmax=vmax,
   3601         colorizer=colorizer,
   3602         origin=origin,
   3603         extent=extent,
   3604         interpolation_stage=interpolation_stage,
   3605         filternorm=filternorm,
   3606         filterrad=filterrad,
   3607         resample=resample,
   3608         url=url,
   3609         **({"data": data} if data is not None else {}),
   3610         **kwargs,
   3611     )
   3612     sci(__ret)
   3613     return __ret

File c:\Users\Rishikesh\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\__init__.py:1521, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1518 @functools.wraps(func)
   1519 def inner(ax, *args, data=None, **kwargs):
   1520     if data is None:
-> 1521         return func(
   1522             ax,
   1523             *map(cbook.sanitize_sequence, args),
   1524             **{k: cbook.sanitize_sequence(v) for k, v in kwargs.items()})
   1526     bound = new_sig.bind(ax, *args, **kwargs)
   1527     auto_label = (bound.arguments.get(label_namer)
   1528                   or bound.kwargs.get(label_namer))

File c:\Users\Rishikesh\AppData\Local\Programs\Python\Python311\Lib\site-packages\matplotlib\axes\_axes.py:5945, in Axes.imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, colorizer, origin, extent, interpolation_stage, filternorm, f
```

```
ilterrad, resample, url, **kwargs)
   5942 if aspect is not None:
   5943     self.set_aspect(aspect)
-> 5945 im.set_data(X)
   5946 im.set_alpha(alpha)
   5947 if im.get_clip_path() is None:
   5948     # image does not already have clipping set, clip to Axes patch

File c:\Users\Rishikesh\AppData\Local\Programs\Python\Python311\Lib\site-packages\ma
tplotlib\image.py:675, in _ImageBase.set_data(self, A)
    673 if isinstance(A, PIL.Image.Image):
    674     A = pil_to_array(A)  # Needed e.g. to apply png palette.
--> 675 self._A = self._normalize_image_array(A)
    676 self._imcache = None
    677 self.stale = True

File c:\Users\Rishikesh\AppData\Local\Programs\Python\Python311\Lib\site-packages\ma
tplotlib\image.py:638, in _ImageBase._normalize_image_array(A)
    636 A = cbook.safe_masked_invalid(A, copy=True)
    637 if A.dtype != np.uint8 and not np.can_cast(A.dtype, float, "same_kind"):
--> 638     raise TypeError(f"Image data of dtype {A.dtype} cannot be "
    639                     f"converted to float")
    640 if A.ndim == 3 and A.shape[-1] == 1:
    641     A = A.squeeze(-1)  # If just (M, N, 1), assume scalar and apply colorma
p.

TypeError: Image data of dtype object cannot be converted to float
```



In [ ]:
```
image_gray=rgb2gray(image)
plt.imshow(image_gray, cmap='gray')
```

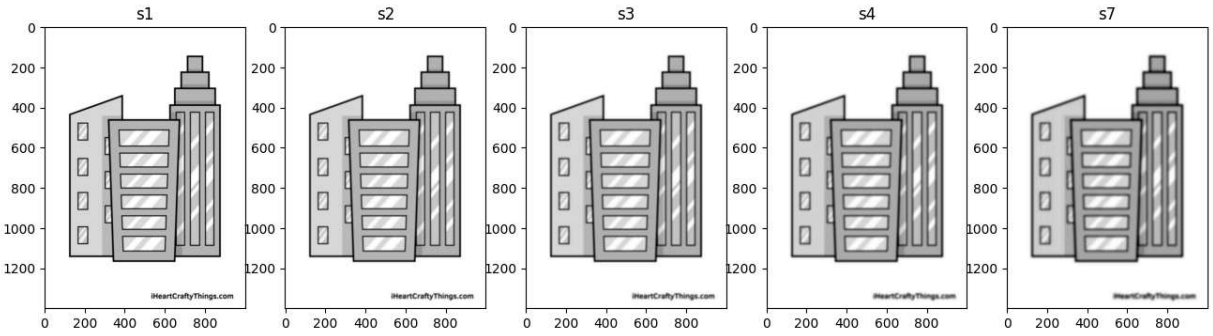
Out[ ]: <matplotlib.image.AxesImage at 0x7b690f7af280>



In [ ]:
```
image_blur_s1=cv2.GaussianBlur(image_gray,(31,31),1)
image_blur_s2=cv2.GaussianBlur(image_gray,(31,31),2)
image_blur_s3=cv2.GaussianBlur(image_gray,(31,31),3)
image_blur_s4=cv2.GaussianBlur(image_gray,(31,31),4)
image_blur_s7=cv2.GaussianBlur(image_gray,(31,31),5)
```

In [ ]:
```
plt.figure(figsize=(16,10))
plt.subplot(1, 5, 1)
plt.imshow(image_blur_s1, cmap='gray')
plt.title('s1')

plt.subplot(1, 5, 2)
plt.imshow(image_blur_s2, cmap='gray')
plt.title('s2')

plt.subplot(1, 5, 3)
plt.imshow(image_blur_s3, cmap='gray')
plt.title('s3')

plt.subplot(1, 5, 4)
plt.imshow(image_blur_s4, cmap='gray')
plt.title('s4')

plt.subplot(1, 5, 5)
plt.imshow(image_blur_s7, cmap='gray')
plt.title('s7')
```

Out[ ]: Text(0.5, 1.0, 's7')



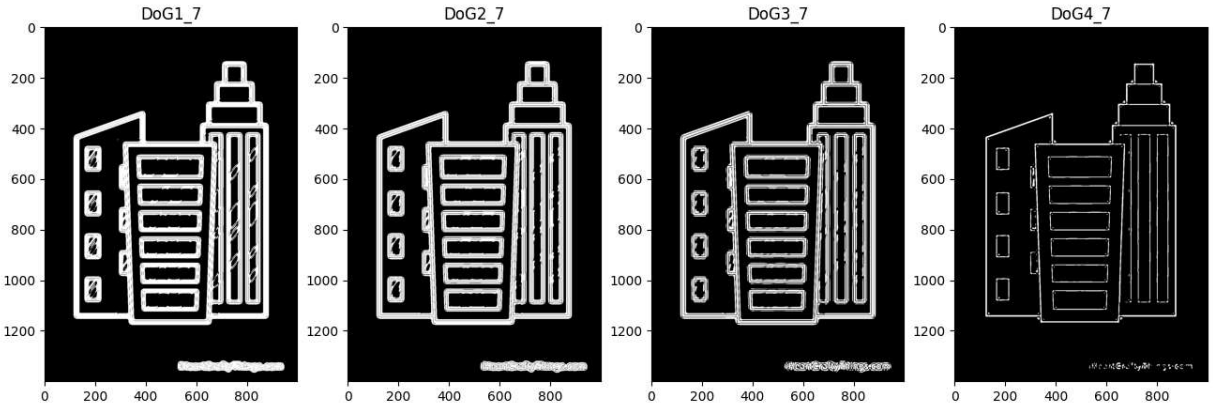
In [ ]:
```
DoG1_7=image_blur_s1-image_blur_s7
DoG2_7=image_blur_s2-image_blur_s7
DoG3_7=image_blur_s3-image_blur_s7
DoG4_7=image_blur_s4-image_blur_s7
```

In [ ]:
```
threshold=np.max(DoG1_7)*0.1
```

In [ ]:
```
DoG1_7[np.abs(DoG1_7)>threshold]=255
DoG2_7[np.abs(DoG2_7)>threshold]=255
DoG3_7[np.abs(DoG3_7)>threshold]=255
DoG4_7[np.abs(DoG4_7)>threshold]=255
```

In [ ]:
```
plt.figure(figsize=(16,10))
plt.subplot(1, 4, 1)
plt.imshow(DoG1_7, cmap='gray')
plt.title('DoG1_7')
plt.subplot(1, 4, 2)
plt.imshow(DoG2_7, cmap='gray')
plt.title('DoG2_7')

plt.subplot(1, 4, 3)
plt.imshow(DoG3_7, cmap='gray')
plt.title('DoG3_7')

plt.subplot(1, 4, 4)
plt.imshow(DoG4_7, cmap='gray')
plt.title('DoG4_7')
```

Out[ ]: Text(0.5, 1.0, 'DoG4_7')



Conclusion:

that the given image is blurred with sigma value 1,2,3,4 and 7.

Difference of Gausian is computed for Gausian Blurred images for the following pair:

1. (1_7)
2. (2_7)
3. (3_7)
4. (4_7)

for the threshold of 20% of the maximum value of DOG1 for the pair (1_7),it is obvserved that sigma pair 1_7 shows maximum number of edges, and sigma pairs of (4_7) shows minimum number of edges.

This is because sigma1 does not blur out fine details also sigma4 blurs out maximum fine details.

If highest sigma value of 7 is increased to 10 then image is blurred more leading to large values of DoG.Therefore more number of edge point cross threshold and more number of edges are identified.

New Image:

For some Images edges are very thick even if the image does not have thick edges in such cases reduce the filter size from 31x31 to 11x11.

All DoG was showing fine details like grass,skin,hair.To avoid fine details Gausain Filter of low sigma value can be given sigma value of 3,4,5,6