Bilateral Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
plt.imshow(cv2.cvtColor(img,0))
plt.show()
im_H = img.shape[0]
im_W = img.shape[1]
def gaussian(sigma,img,ksize,padding):
    gfilter = np.zeros((ksize,ksize),np.float32)
    div = (sigma*sigma)*2
    for i in range(-padding,padding+1):
        for j in range(-padding,padding+1):
            gfilter[i+padding,j+padding] = math.exp(-((i**2+j**2)/div))
    return gfilter
ksize = 5
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
sigma = 5
div = (sigma*sigma)*2
gaussian_filter = gaussian(sigma,img,ksize,padding)
for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, fil = 0,normalization = 0
        ip = img[x,y]
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                iq = img[x-i,y-j]
                fil = gaussian_filter[i+padding,j+padding]*(math.exp(-(((ip-iq)**2)/div)))
                normalization += fil
                a += fil*img[x-i,y-j]
        result[x,y] = a/normalization
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

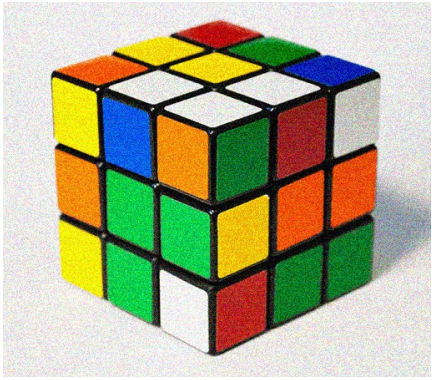

Fig1.1: Input image

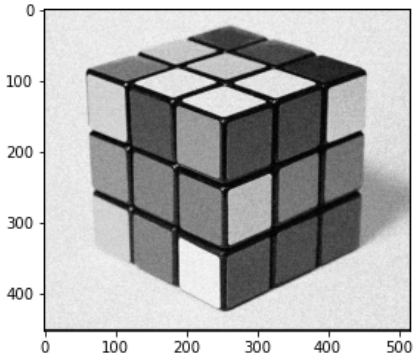

Fig1.2: Output Image

Gaussian Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("lena.png",cv2.IMREAD_GRAYSCALE)
plt.imshow(cv2.cvtColor(img,0))
plt.show()
im_H = img.shape[0]
im_W = img.shape[1]
def gaussian(sigma,img,ksize,padding):
    gfilter = np.zeros((ksize,ksize),np.float32)
    div = (sigma*sigma)*2
    for i in range(-padding,padding+1):
        for j in range(-padding,padding+1):
            gfilter[i+padding,j+padding] = math.exp(-((i**2+j**2)/div))
    return gfilter
ksize = 7
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
```

```
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
sigma = 5
div = (sigma*sigma)*2
gaussian_filter = gaussian(sigma,img,ksize,padding)
 for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, normalize = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += gaussian_filter[i+padding,j+padding]*img[x-i,y-j]
                normalize += gaussian_filter[i+padding,j+padding]
        result[x,y] = a/normalize
        result[x,y] /= 255
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

Fig 2.1: Input Image

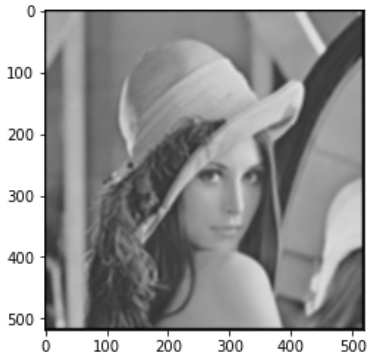

Fig 2.2: Output Image

Mean Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 5
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
div = ksize*ksize
for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += img[x-i,y-j]
        result[x,y] = a/div
        result[x,y] = result[x,y]/255
print(result)
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

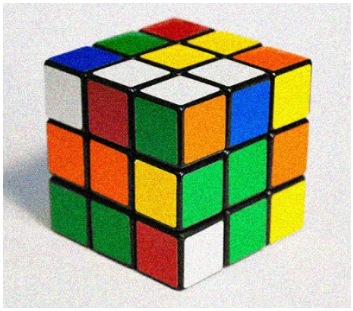

Fig 3.1: Input Image

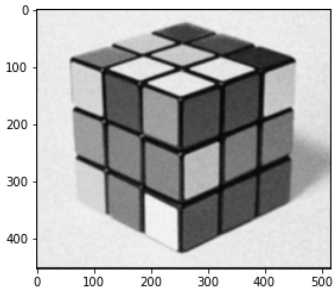

Fig 3.2: Output Image

Median Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
#take input image
img = cv2.imread("cameraman.jpg",cv2.IMREAD_GRAYSCALE)
plt.imshow(cv2.cvtColor(img,0))
plt.show()
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 5
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
div = ksize*ksize
median = div//2
for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = []
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a.append(img[x-i,y-j])
        a.sort()
        result[x,y] = a[median]/255
print(result)
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

Fig4.1: Input Image

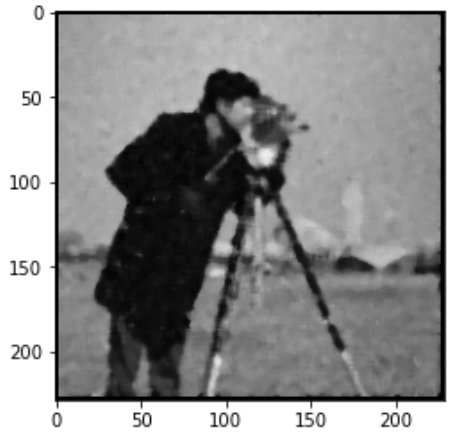

Fig 4.2: Output Image

Sobel Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 3
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
gx = np.zeros((output_H,output_W),np.float32)
gy = np.zeros((output_H,output_W),np.float32)
g = np.zeros((output_H,output_W),np.float32)
hx = np.array(([1,0,-1],[2,0,-2],[1,0,-1]),np.float32)
hy = np.array(([-1,-2,-1],[0,0,0],[1,2,1]),np.float32)
 for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, b = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += hx[i+padding,j+padding]*img[x-i,y-j]
                b += hy[i+padding,j+padding]*img[x-i,y-j]
        gx[x,y] = a
        gy[x,y] = b
        g[x,y] = np.sqrt(a**2+b**2)
cv2.normalize(g, g, 0, 255, cv2.NORM_MINMAX)
g = np.round(g).astype(np.uint8)
plt.imshow(cv2.cvtColor(g,0))
plt.show()
```

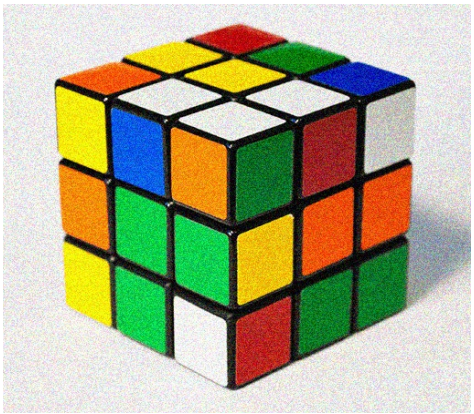

Fig 5.1: Input Image

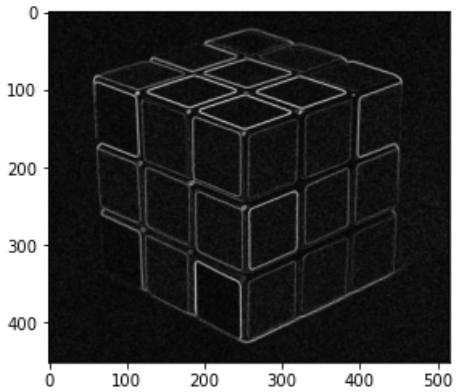

Fig 5.2: Output Image

Prewitt Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 3
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
gx = np.zeros((output_H,output_W),np.float32)
gy = np.zeros((output_H,output_W),np.float32)
g = np.zeros((output_H,output_W),np.float32)
hx = np.array(([-1,0,1],[-1,0,1],[-1,0,1]),np.float32)
hy = np.array(([-1,-1,-1],[0,0,0],[1,1,1]),np.float32)
 for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, b = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += hx[i+padding,j+padding]*img[x-i,y-j]
                b += hy[i+padding,j+padding]*img[x-i,y-j]
        gx[x,y] = a
        gy[x,y] = b
        g[x,y] = np.sqrt(a**2+b**2)
cv2.normalize(g, g, 0, 255, cv2.NORM_MINMAX)
g = np.round(g).astype(np.uint8)
plt.imshow(cv2.cvtColor(g,0))
plt.show()
```

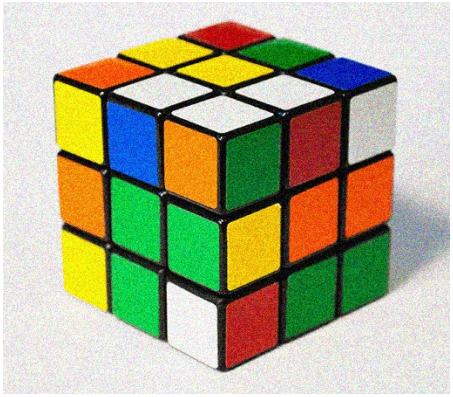

Fig 6.1: Input Image

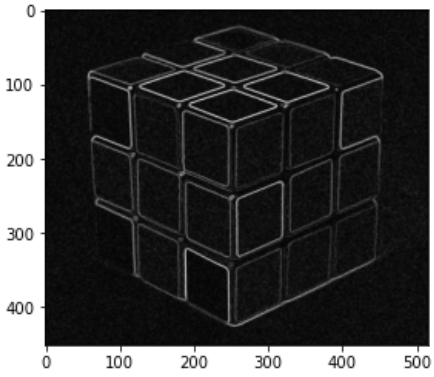

Fig 6.2: Output Image

Scharr Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 3
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
gx = np.zeros((output_H,output_W),np.float32)
gy = np.zeros((output_H,output_W),np.float32)
g = np.zeros((output_H,output_W),np.float32)
hx = np.array(([-3,0,3],[-10,0,10],[-3,0,3]),np.float32)
hy = np.array(([3,10,3],[0,0,0],[-3,-10,-3]),np.float32)
 for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, b = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += hx[i+padding,j+padding]*img[x-i,y-j]
                b += hy[i+padding,j+padding]*img[x-i,y-j]
        gx[x,y] = a
        gy[x,y] = b
        g[x,y] = np.sqrt(a**2+b**2)
cv2.normalize(g, g, 0, 255, cv2.NORM_MINMAX)
g = np.round(g).astype(np.uint8)
plt.imshow(cv2.cvtColor(g,0))
plt.show()
```

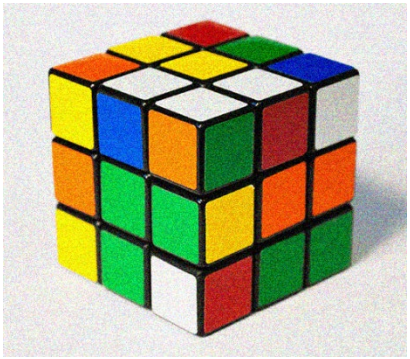

Fig 7.1: Input Image

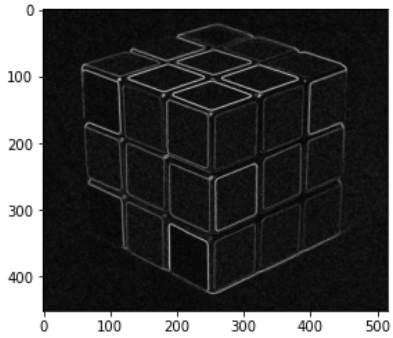

Fig 7.2: Output Image

Robert Filter

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("rubiks_cube.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 2
padding = (ksize-1)
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
gx = np.zeros((output_H,output_W),np.float32)
gy = np.zeros((output_H,output_W),np.float32)
g = np.zeros((output_H,output_W),np.float32)
hx = np.array(([1,0],[0,-1]),np.float32)
hy = np.array(([0,1],[-1,0]),np.float32)
 for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0, b = 0
        for i in range(ksize):
            for j in range(ksize):
                a += hx[i,j]*img[x-i,y-j]
                b += hy[i,j]*img[x-i,y-j]
        gx[x,y] = a
        gy[x,y] = b
        g[x,y] = np.sqrt(a**2+b**2)
cv2.normalize(g, g, 0, 255, cv2.NORM_MINMAX)
g = np.round(g).astype(np.uint8)
plt.imshow(cv2.cvtColor(g,0))
plt.show()
```

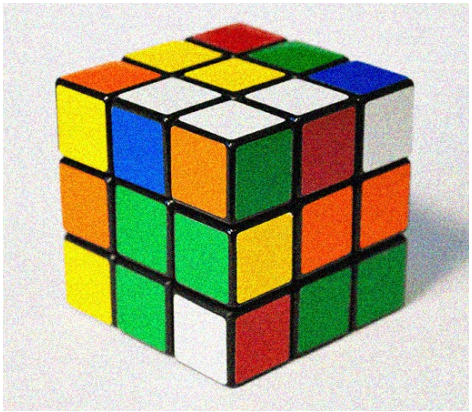

Fig 8.1: Input Image

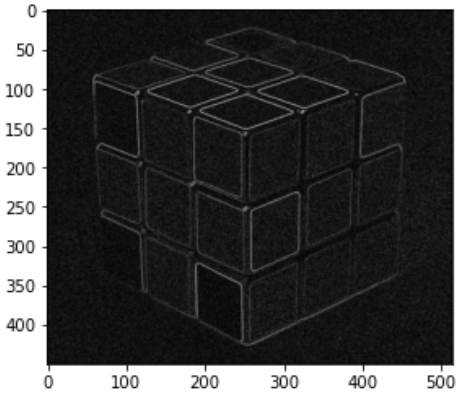

Fig 8.2: Output Image

Laplacian Version 1

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("lena.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 3
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
laplacian = np.array(([0,-1,0],[-1,4,-1],[0,-1,0]),np.float32)
for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += laplacian[i+padding,j+padding]*img[x-i,y-j]
        result[x,y] = a
cv2.normalize(result, result, 0, 255, cv2.NORM_MINMAX)
result = np.round(result).astype(np.uint8)
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

Fig 9.1: Input Image

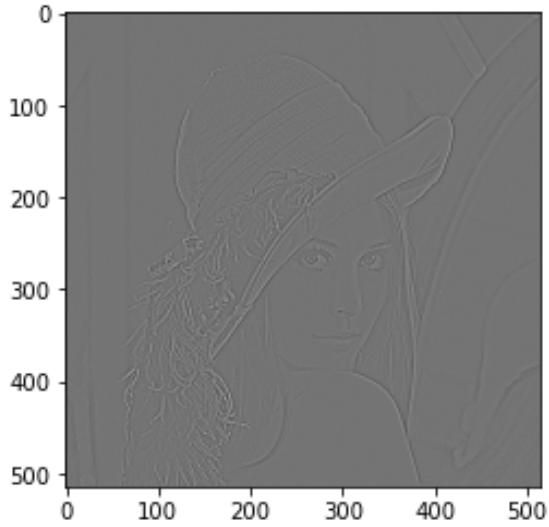

Fig 9.2: Ouput Image

Laplacian Version 2

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("lena.png",cv2.IMREAD_GRAYSCALE)
im_H = img.shape[0]
im_W = img.shape[1]
ksize = 3
padding = (ksize-1)//2
img = cv2.copyMakeBorder(img, padding, padding, padding, padding, cv2.BORDER_REPLICATE)
output_H = (im_H + ksize-1)
output_W = (im_W + ksize-1)
result = np.zeros((output_H,output_W),np.float32)
laplacian = np.array(([-1,-1,-1],[-1,8,-1],[-1,-1,-1]),np.float32)
for x in range(padding,output_H-padding):
    for y in range(padding,output_W-padding):
        a = 0
        for i in range(-padding,padding+1):
            for j in range(-padding,padding+1):
                a += laplacian[i+padding,j+padding]*img[x-i,y-j]
        result[x,y] = a
cv2.normalize(result, result, 0, 255, cv2.NORM_MINMAX)
result = np.round(result).astype(np.uint8)
plt.imshow(cv2.cvtColor(result,0))
plt.show()
```

Fig 10.1: Input Image

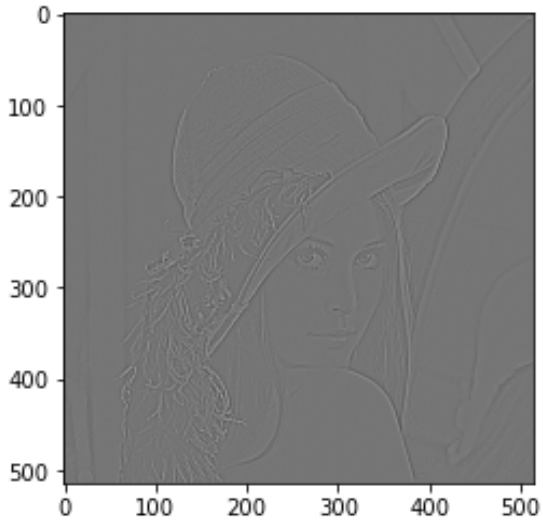

Fig 10.2: Output Image