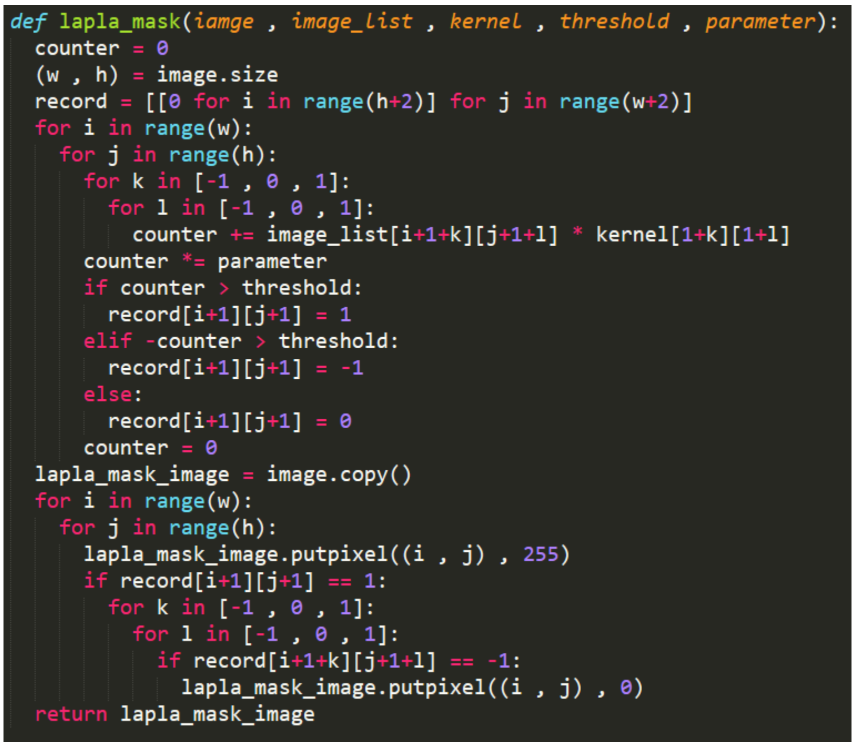
Computer vision homework 10

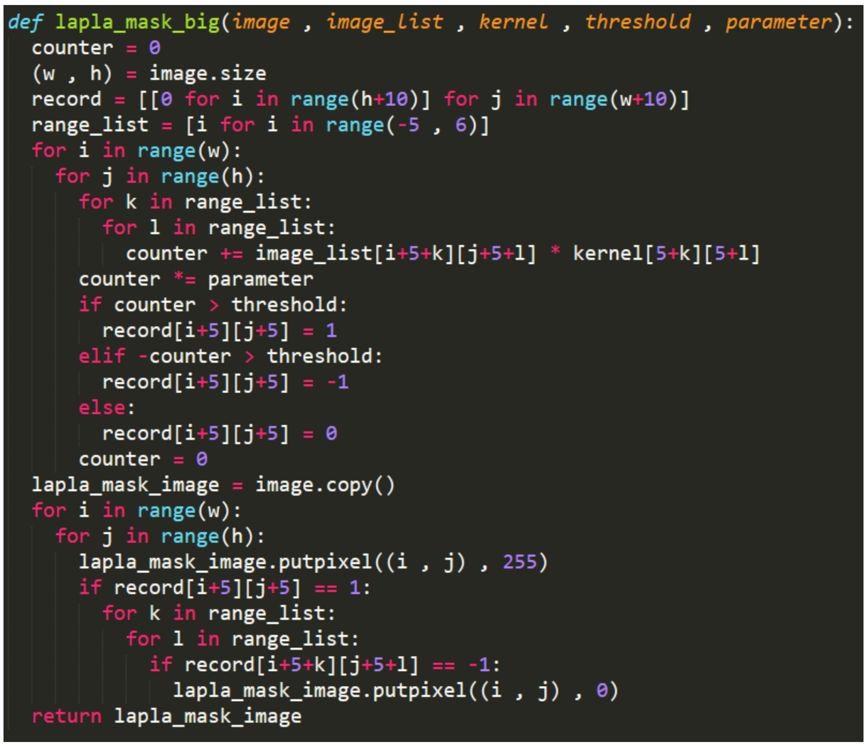
B04902028 資工三 洪浩翔

Original Lena Image



```python
def lapla_mask(iamge , image_list , kernel , threshold , parameter):
  counter = 0
  (w , h) = image.size
  record = [[0 for i in range(h+2)] for j in range(w+2)]
  for i in range(w):
    for j in range(h):
      for k in [-1 , 0 , 1]:
        for l in [-1 , 0 , 1]:
          counter += image_list[i+1+k][j+1+l] * kernel[1+k][1+l]
      counter *= parameter
      if counter > threshold:
        record[i+1][j+1] = 1
      elif -counter > threshold:
        record[i+1][j+1] = -1
      else:
        record[i+1][j+1] = 0
      counter = 0
  lapla_mask_image = image.copy()
  for i in range(w):
    for j in range(h):
      lapla_mask_image.putpixel((i , j) , 255)
      if record[i+1][j+1] == 1:
        for k in [-1 , 0 , 1]:
          for l in [-1 , 0 , 1]:
            if record[i+1+k][j+1+l] == -1:
              lapla_mask_image.putpixel((i , j) , 0)
  return lapla_mask_image
```

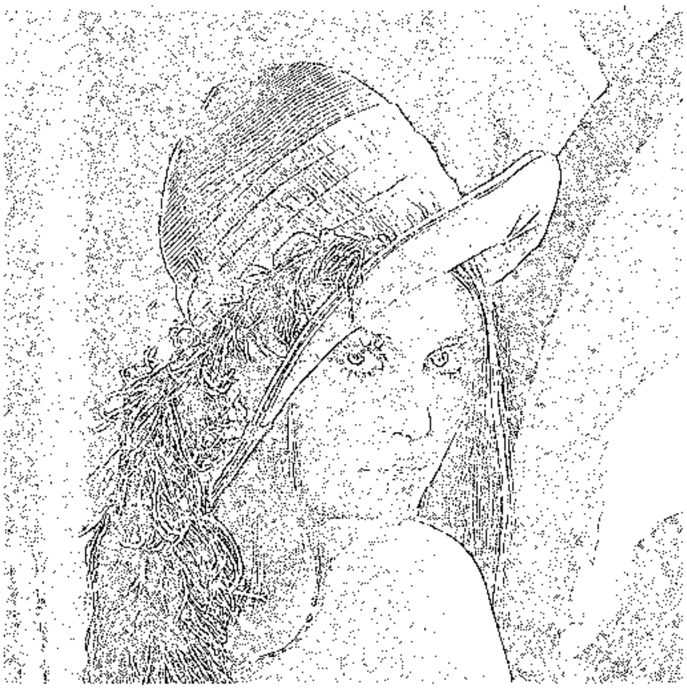
Main function 1 for (1)~(3) to dealing 3*3 kernel



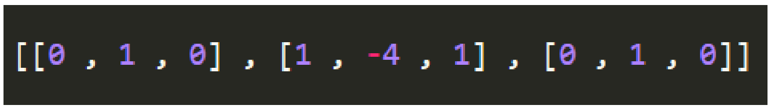
```python
def lapla_mask_big(image , image_list , kernel , threshold , parameter):
  counter = 0
  (w , h) = image.size
  record = [[0 for i in range(h+10)] for j in range(w+10)]
  range_list = [i for i in range(-5 , 6)]
  for i in range(w):
    for j in range(h):
      for k in range_list:
        for l in range_list:
          counter += image_list[i+5+k][j+5+l] * kernel[5+k][5+l]
      counter *= parameter
      if counter > threshold:
        record[i+5][j+5] = 1
      elif -counter > threshold:
        record[i+5][j+5] = -1
      else:
        record[i+5][j+5] = 0
      counter = 0
  lapla_mask_image = image.copy()
  for i in range(w):
    for j in range(h):
      lapla_mask_image.putpixel((i , j) , 255)
      if record[i+5][j+5] == 1:
        for k in range_list:
          for l in range_list:
            if record[i+5+k][j+5+l] == -1:
              lapla_mask_image.putpixel((i , j) , 0)
  return lapla_mask_image
```

Main function 2 for (4)~(5) to dealing 11*11 kernel

1. Laplacian Mask with kernel 1:

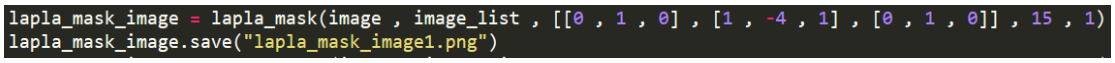


Result of using kernel 1 and threshold = 15



```
[[0 , 1 , 0] , [1 , -4 , 1] , [0 , 1 , 0]]
```

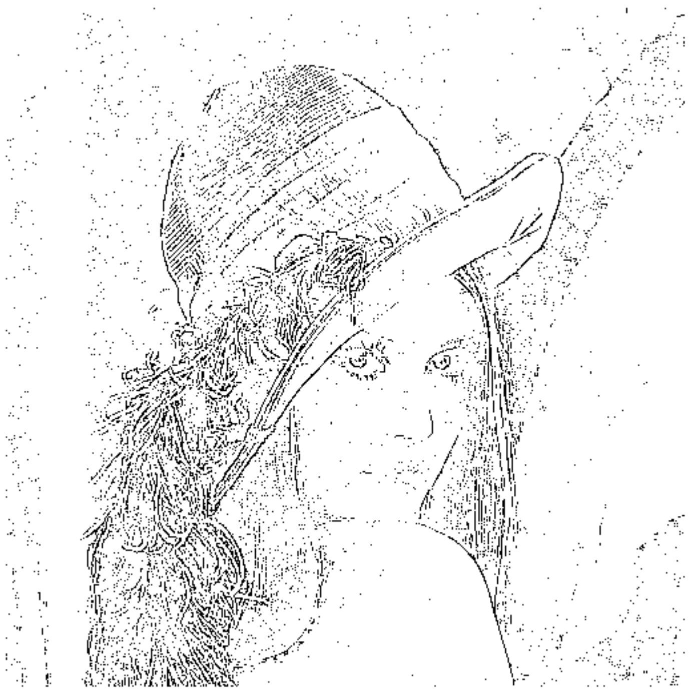
Kernel 1



```
lapla_mask_image = lapla_mask(image , image_list , [[0 , 1 , 0] , [1 , -4 , 1] , [0 , 1 , 0]] , 15 , 1)
lapla_mask_image.save("lapla_mask_image1.png")
```

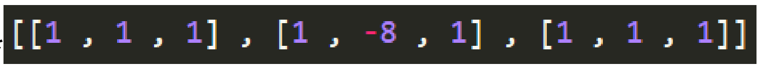
Code of mask

2. Laplacian Mask with kernel 2:



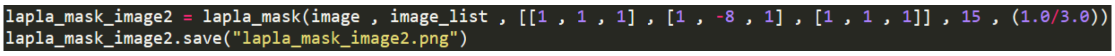
Result of using kernel 2 and threshold = 15

(1/3)*



```
[[1 , 1 , 1] , [1 , -8 , 1] , [1 , 1 , 1]]
```

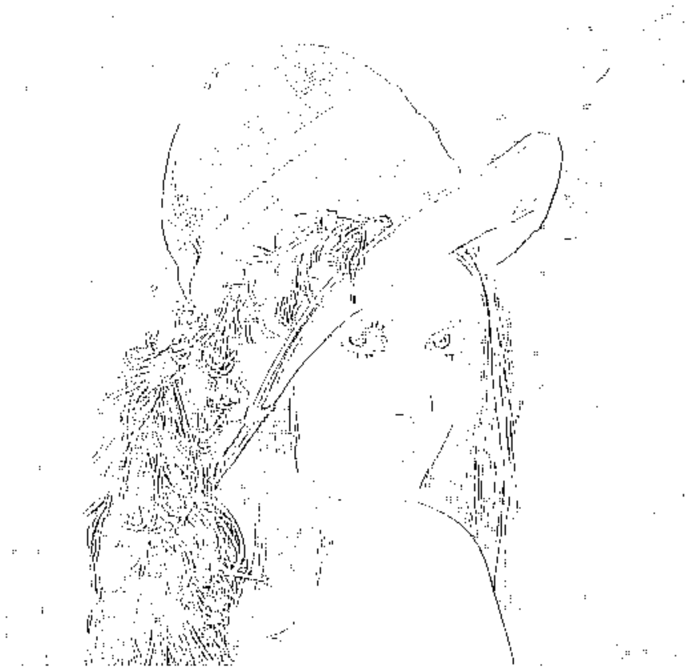
Kernel 2



```
lapla_mask_image2 = lapla_mask(image , image_list , [[1 , 1 , 1] , [1 , -8 , 1] , [1 , 1 , 1]] , 15 , (1.0/3.0))
lapla_mask_image2.save("lapla_mask_image2.png")
```

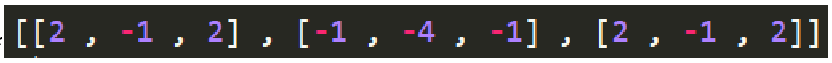
Code of mask

3. Minimum Variance Laplacian:



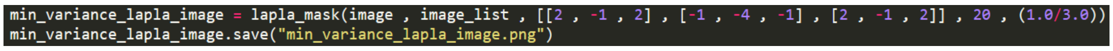
Result of Minimum Variance Laplacian with threshold = 20

(1/3)*



```
[[2 , -1 , 2] , [-1 , -4 , -1] , [2 , -1 , 2]]
```

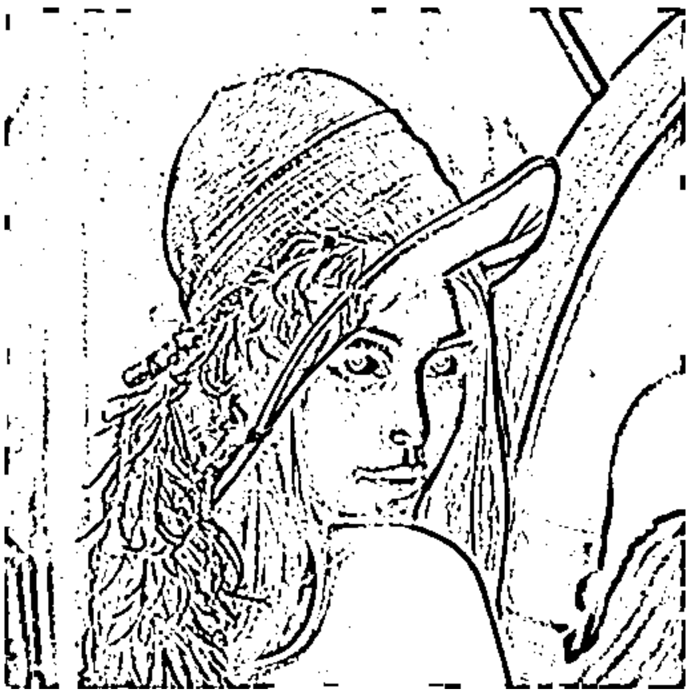
Kernel



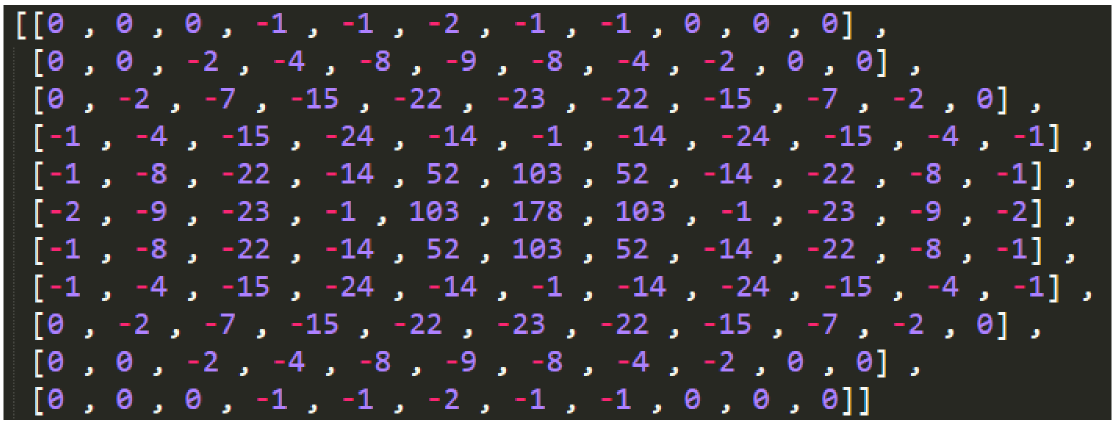
```
min_variance_lapla_image = lapla_mask(image , image_list , [[2 , -1 , 2] , [-1 , -4 , -1] , [2 , -1 , 2]] , 20 , (1.0/3.0))
min_variance_lapla_image.save("min_variance_lapla_image.png")
```

Code of Minimum Variance Laplacian

4. Laplacian of Gaussian:

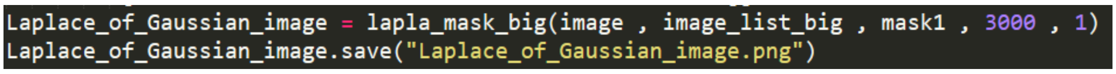


Result of Laplacian of Gaussian with threshold = 3000



```
[[0 , 0 , 0 , -1 , -1 , -2 , -1 , -1 , 0 , 0 , 0] ,
 [0 , 0 , -2 , -4 , -8 , -9 , -8 , -4 , -2 , 0 , 0] ,
 [0 , -2 , -7 , -15 , -22 , -23 , -22 , -15 , -7 , -2 , 0] ,
 [-1 , -4 , -15 , -24 , -14 , -1 , -14 , -24 , -15 , -4 , -1] ,
 [-1 , -8 , -22 , -14 , 52 , 103 , 52 , -14 , -22 , -8 , -1] ,
 [-2 , -9 , -23 , -1 , 103 , 178 , 103 , -1 , -23 , -9 , -2] ,
 [-1 , -8 , -22 , -14 , 52 , 103 , 52 , -14 , -22 , -8 , -1] ,
 [-1 , -4 , -15 , -24 , -14 , -1 , -14 , -24 , -15 , -4 , -1] ,
 [0 , -2 , -7 , -15 , -22 , -23 , -22 , -15 , -7 , -2 , 0] ,
 [0 , 0 , -2 , -4 , -8 , -9 , -8 , -4 , -2 , 0 , 0] ,
 [0 , 0 , 0 , -1 , -1 , -2 , -1 , -1 , 0 , 0 , 0]]
```

Kernel of Laplacian of Gaussian(mask1 in code)

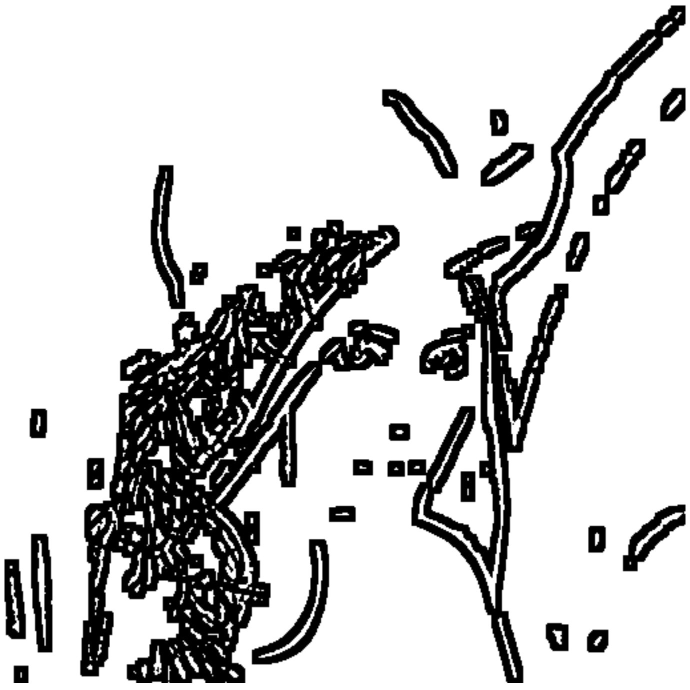


```
Laplace_of_Gaussian_image = lapla_mask_big(image , image_list_big , mask1 , 3000 , 1)
Laplace_of_Gaussian_image.save("Laplace_of_Gaussian_image.png")
```

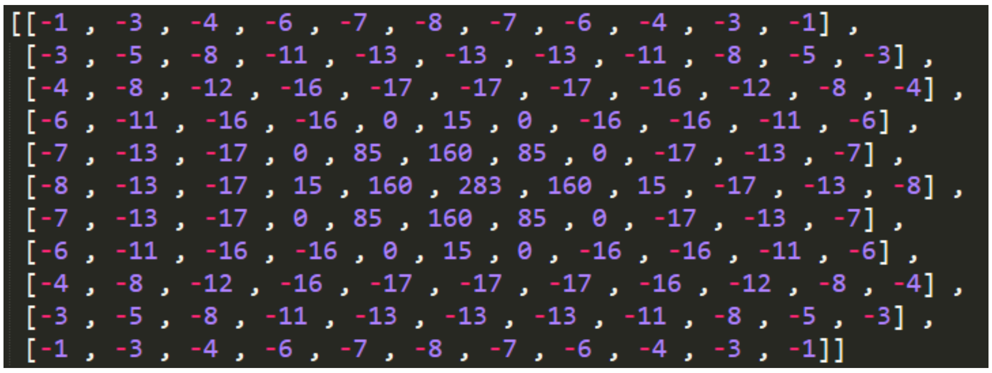
Code of Laplacian of Gaussian

5. Difference of Gaussian:

(nhibitory sigma=1, excitatory sigma=3, kernel size 11x11)

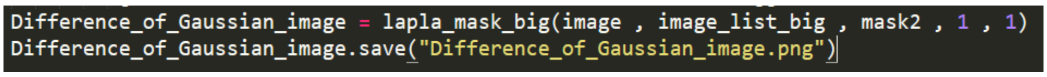


Result of Difference of Gaussian with threshold = 1



```
[[-1 , -3 , -4 , -6 , -7 , -8 , -7 , -6 , -4 , -3 , -1] ,
 [-3 , -5 , -8 , -11 , -13 , -13 , -13 , -11 , -8 , -5 , -3] ,
 [-4 , -8 , -12 , -16 , -17 , -17 , -17 , -16 , -12 , -8 , -4] ,
 [-6 , -11 , -16 , -16 , 0 , 15 , 0 , -16 , -16 , -11 , -6] ,
 [-7 , -13 , -17 , 0 , 85 , 160 , 85 , 0 , -17 , -13 , -7] ,
 [-8 , -13 , -17 , 15 , 160 , 283 , 160 , 15 , -17 , -13 , -8] ,
 [-7 , -13 , -17 , 0 , 85 , 160 , 85 , 0 , -17 , -13 , -7] ,
 [-6 , -11 , -16 , -16 , 0 , 15 , 0 , -16 , -16 , -11 , -6] ,
 [-4 , -8 , -12 , -16 , -17 , -17 , -17 , -16 , -12 , -8 , -4] ,
 [-3 , -5 , -8 , -11 , -13 , -13 , -13 , -11 , -8 , -5 , -3] ,
 [-1 , -3 , -4 , -6 , -7 , -8 , -7 , -6 , -4 , -3 , -1]]
```

Kernel of Difference of Gaussian(mask2 in code)



```
Difference_of_Gaussian_image = lapla_mask_big(image , image_list_big , mask2 , 1 , 1)
Difference_of_Gaussian_image.save("Difference_of_Gaussian_image.png")
```

Code of Difference of Gaussian