Python 物理建模初学者指南第八章：第三次上机实验

2020 年 4 月 8 日

1 第八章：第三次上机实验

本次上机实验的目标是：

探索在各种图像上做局部平均的效果。

查看如何使用这种平均对图像降噪。
使用特定的滤波器增强指定的图像特性。

1.1 8.1 卷积

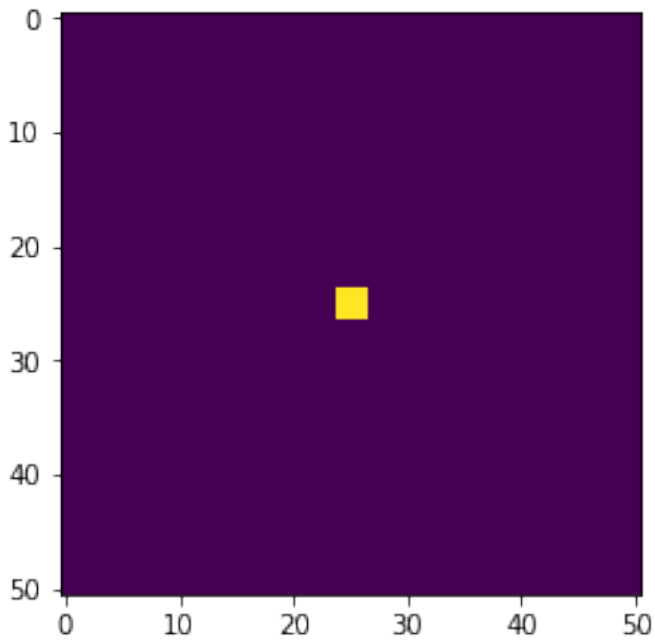
1.1.1 8.1.1 Python 的图像处理工具

[1]:
```
import scipy.ndimage as sim, numpy as np, matplotlib.pyplot as plt
%matplotlib qt5
```

sim.convolve 是 scipy.ndimage 中的一个函数，用 help 可以看到更多的信息

[2]:
```
impulse = np.zeros((51, 51))
impulse[25,25] = 1.0
my_filter_small = np.ones((3,3))/9
response = sim.convolve(impulse, my_filter_small)
plt.figure()
plt.imshow(response)
```

[2]:
```
<matplotlib.image.AxesImage at 0x1de6e292048>
```



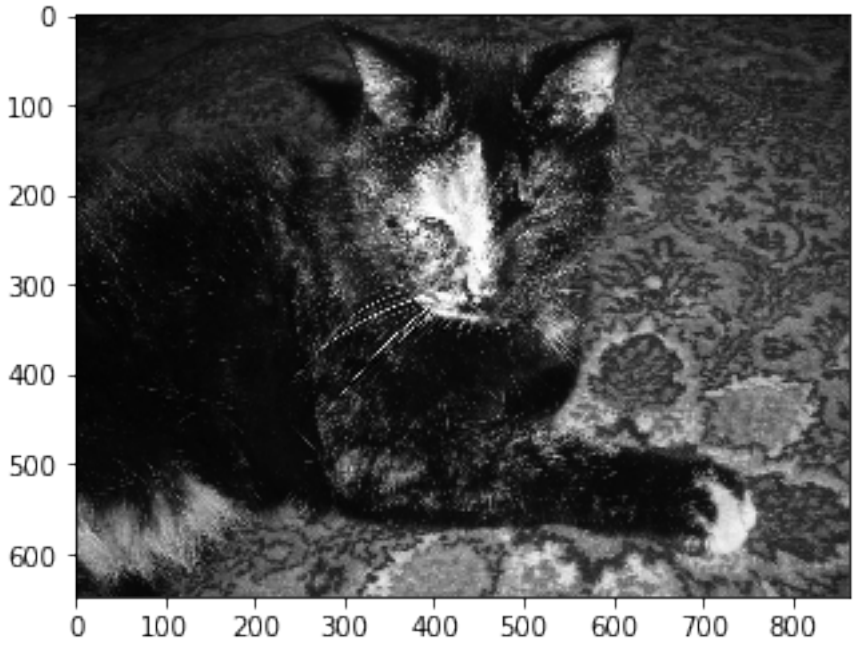
1.1.2 8.1.2 图像平均

一个简单的滤波器会对每个指定区域中的点分配同样的权重。在卷积后的图像中，每个点是原始图像中近邻点的平均值

获取数据集 16catphoto 的图像

[3]:
```
filepath = r'D:\我的课程\计算物理\Datasets\16catphoto\\'
photo = plt.imread(filepath + 'bwCat.tif')
plt.figure('origin cat')
plt.imshow(photo,cmap='gray')
```

[3]:
```
<matplotlib.image.AxesImage at 0x1de6e9c2908>
```

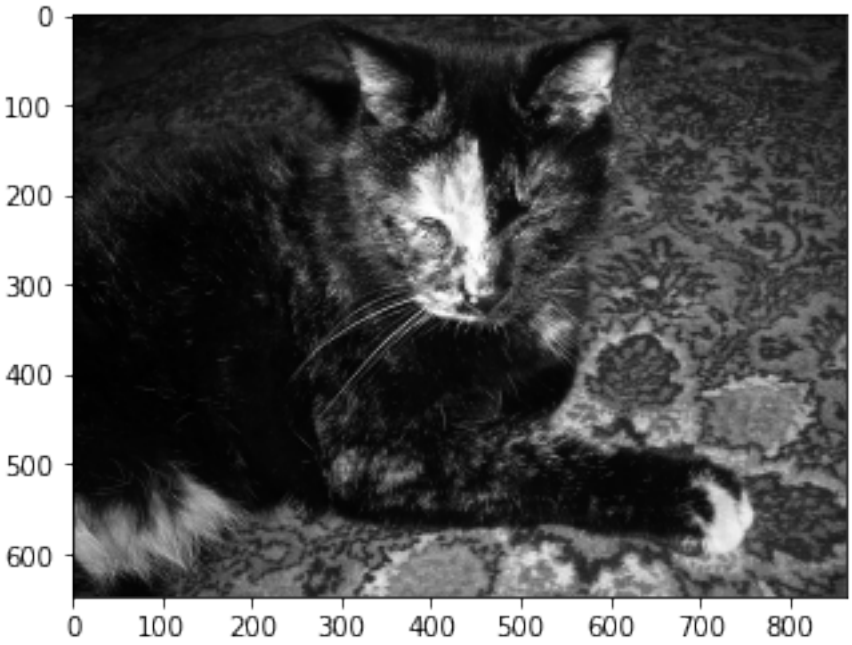


任务

(a) 用前面例子定义的滤波器，调用 sim.convolve 做卷积, 并与 sim.uniform_filter 做比较

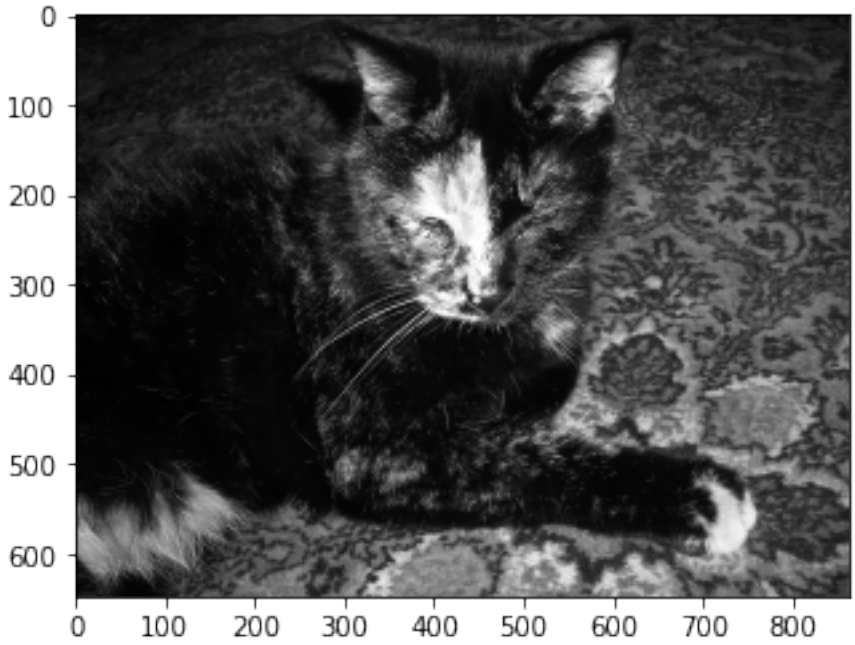
[4]:
```
%%time
plt.figure('sim.convolve')
response= sim.convolve(photo, my_filter_small)
plt.imshow(response,cmap='gray')
```

```
Wall time: 25.9 ms
```



[5]:
```
%%time
response2 = sim.uniform_filter(photo)
plt.figure('uniform_filter')
plt.imshow(response2,cmap='gray')
```

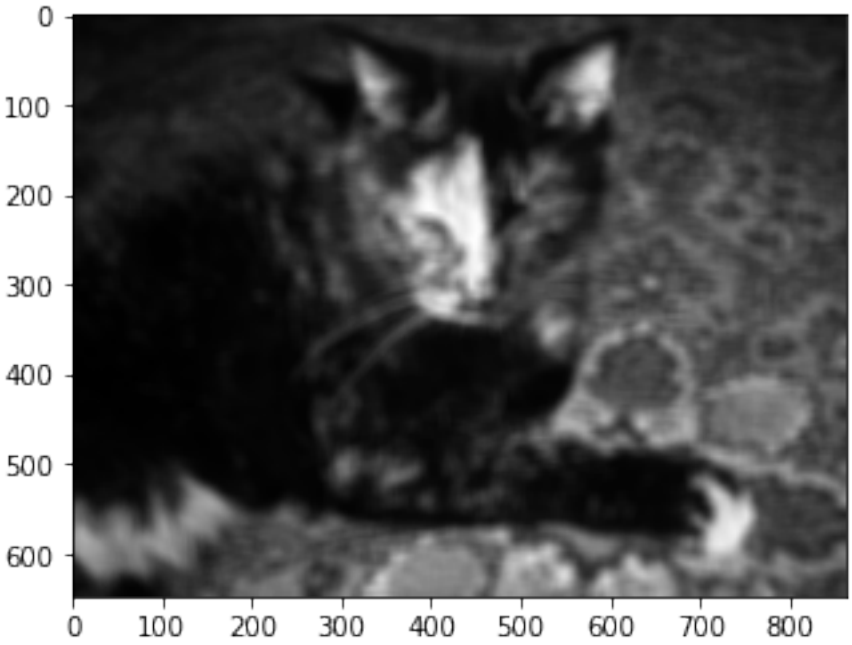
```
Wall time: 27.9 ms
```



(b) 使用一个具有适当值得 15×15 的数组，重复 (a) 中的操作

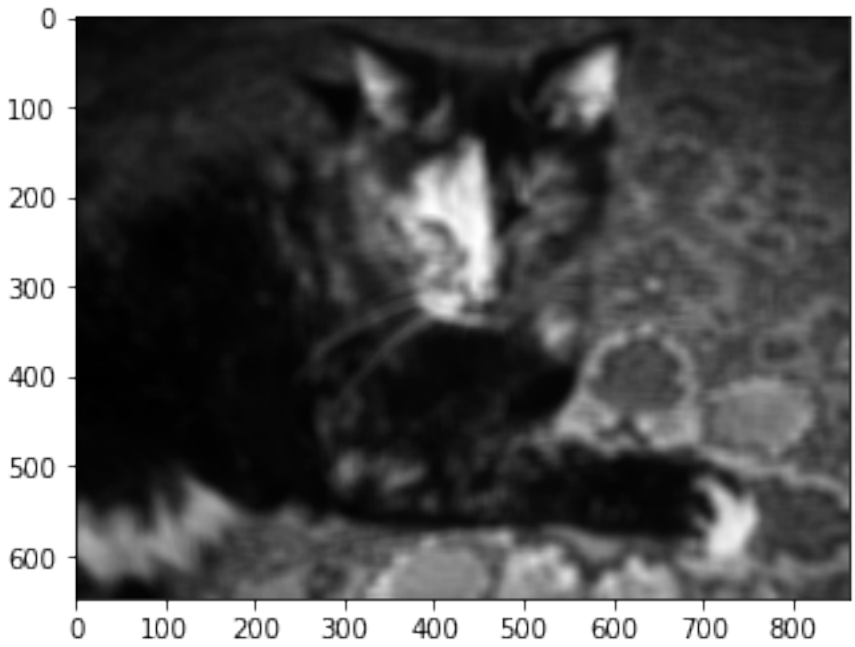
[6]:
```
%%time
my_filter_big = np.ones((15,15))*1/225
response = sim.convolve(photo, my_filter_big)
plt.figure('big filter')
plt.imshow(response, cmap='gray')
```

```
Wall time: 276 ms
```



[7]:
```
%%time
response2 = sim.uniform_filter(photo,size=15)
plt.figure('uniform_filter')
plt.imshow(response2,cmap='gray')
```

```
Wall time: 25.1 ms
```



(c) 使用公式 (8.1) 的定义，显示卷积生成的图像，图像中的每个点是原始图像中邻近像素的平均值??

[8]:
```
def myConvolve(func,img):
    """
    公式 (8.1) C_{i,j} = sum_{k,l}{F_{k,l}*I_{i-k,j-l}}
    """
    ni, nj = img.shape
    nk, nl = func.shape
    print(ni,nj,nk,nl)
    c = np.zeros((ni+nk-1, nj+nl-1))
    for i in range(ni+nk-2):
        for j in range(nj+nl-2):
            sum1 = 0.
            for k in range(min(i, nk)):
                for l in range(min(j, nl)):
                    if (i-k)<ni and (j-l)<nj:
                        sum1 += func[k,l]*img[i-k, j-l]
            c[i,j] = sum1
```

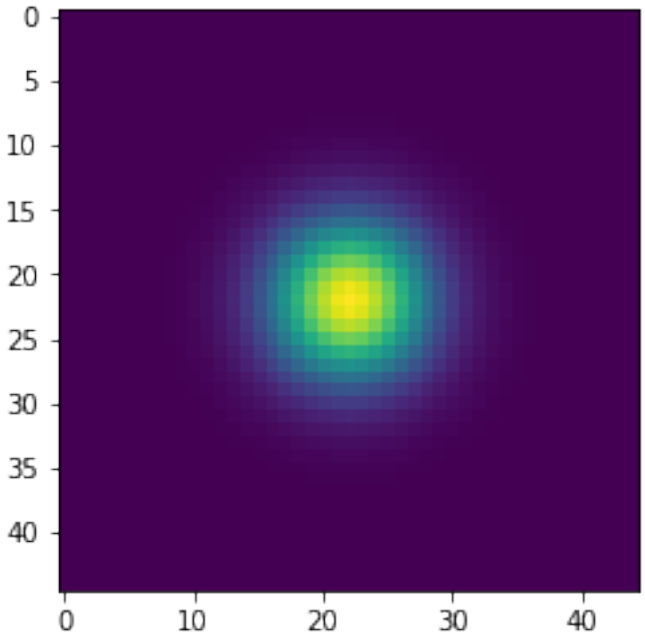
```
    return c
```

1.1.3 8.1.3 使用高斯滤波器做平滑

导入数据：gauss_filter.npy

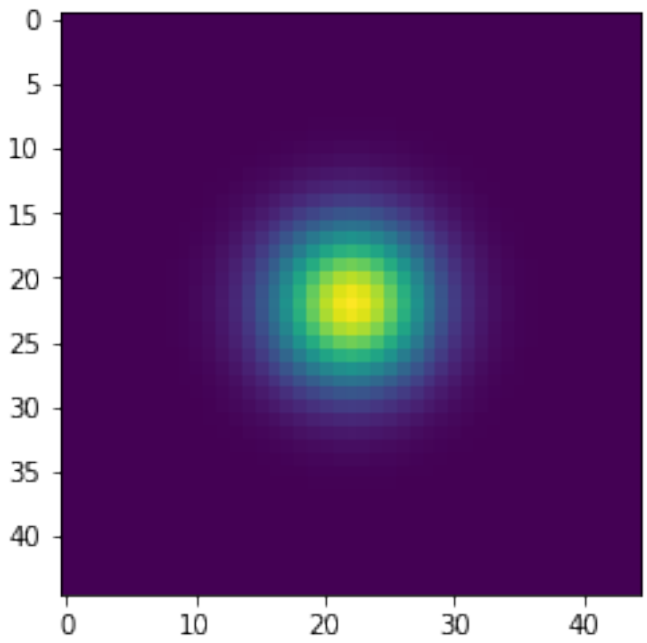
[9]:
```
#gauss_filter.csv
filepath = r'D:\我的课程\计算物理\Datasets\16catphoto\\'
gauss = np.load(filepath + 'gauss_filter.npy')
plt.imshow(gauss)
```

[9]: <matplotlib.image.AxesImage at 0x1de705d6748>



[10]:
```
#gauss_filter.csv
filepath = r'D:\我的课程\计算物理\Datasets\16catphoto\\'
gauss = np.load(filepath + 'gauss_filter.npy')
plt.imshow(gauss)
```

[10]: <matplotlib.image.AxesImage at 0x1de6e9f94a8>

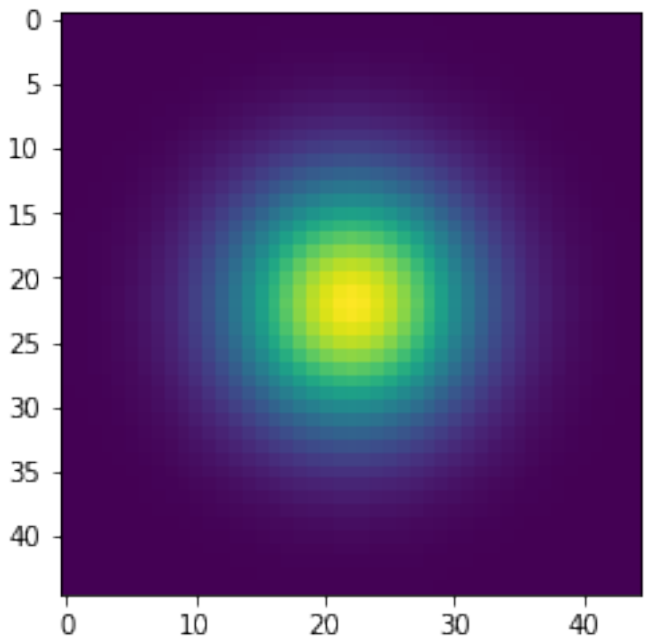


任务：

(a) 显示原始图像的高斯卷积（尝试使用 sim.gauss_filter 函数，使用关键字参数 “sigma = 5”）

[11]:
```
smooth = sim.gaussian_filter(gauss, sigma=5)
plt.figure('smooth')
plt.imshow(smooth)
```

[11]: <matplotlib.image.AxesImage at 0x1de6e909f98>



(b) 调用 plt.imshow 函数，比较使用高斯滤波器的单一点卷积与实验 8.1.2(b) 所使用的正方滤波器

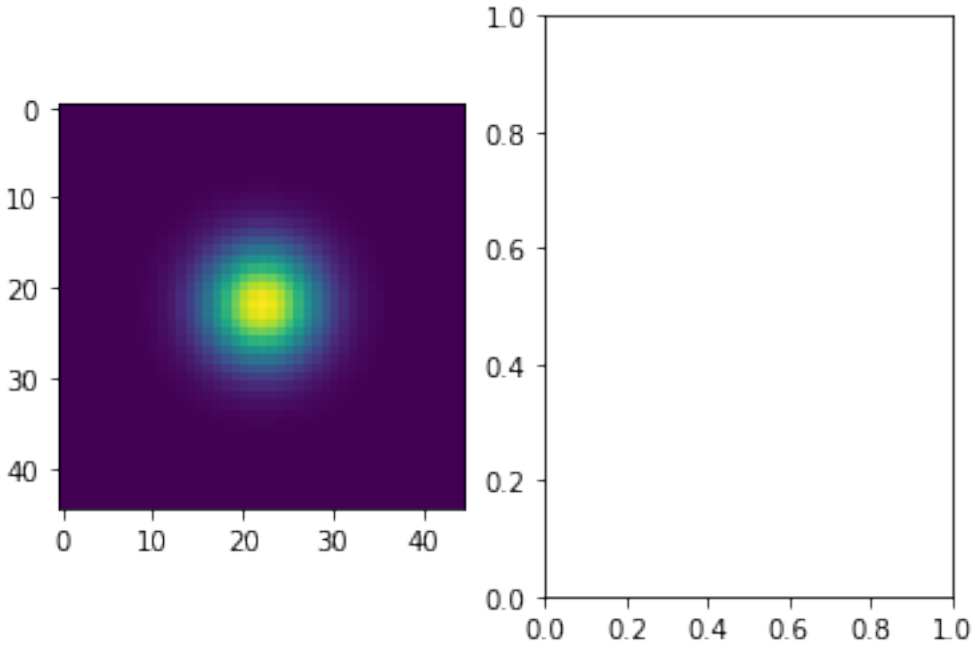
[12]:
```
fig = plt.figure('square')
#square = sim.uniform_filter(gauss, size=3)
square_small = sim.convolve(gauss, my_filter_small)
fig.add_subplot(121)
plt.imshow(square_small)
fig.add_subplot(122)
#square = sim.uniform_filter(gauss, size=3)
square_big = sim.convolve(gauss, my_filter_big)
plt.imshow(square_big)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)
```

```
<ipython-input-12-60bf394d0261> in <module>
  6 fig.add_subplot(122)
  7 #square = sim.uniform_filter(gauss, size=3)
----> 8 square_big = sim.convolve(gauss, my_filter_big)
  9 plt.imshow(square_big)


NameError: name 'my_filter_big' is not defined
```



[ ]:
```
square_small.shape
```

(c) 调用 plot_surface, 从 3 个维度上查看实验 (b) 中的卷积图像。使用卷积的定义介绍原因，然后解释使用高斯滤波器和正方滤波器有哪些不同，在哪些情况下人们更倾向高斯滤波器。

[ ]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure('gauss')
ax1 = fig.add_subplot(121, projection='3d')
x, y = np.mgrid[0:45:45j, 0:45:45j]
```

```
ax1.plot_surface(x, y, smooth)
ax2 = fig.add_subplot(122, projection='3d')
ax2.plot_surface(x, y, square_big)
```

[ ]:
```
plt.imshow(square-smooth)
```

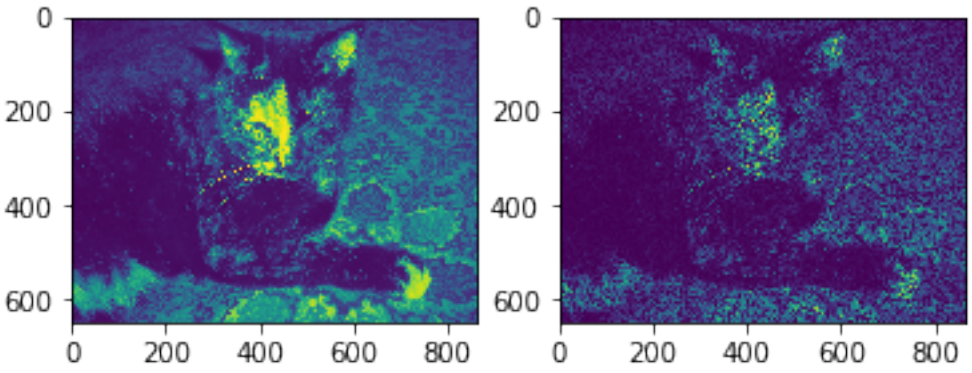
1.2 8.2 图像去噪

任务

(a) 对原始图像的每个点乘以一个介于 0 和 1 之间的随机数，然后将所得的的矩阵按上述步骤进行转换，比较生成的噪声图像和原始图像

[14]:
```
x, y = photo.shape
noise = np.random.random((x,y))
Photo_noise = photo*noise
mi = Photo_noise.min()
ma = Photo_noise.max()
Photo_noise = (Photo_noise-mi)/(ma-mi)*255
Photo_noise = Photo_noise.astype('uint8')
fig, axes = plt.subplots(1,2)
axes[0].imshow(photo)
axes[1].imshow(Photo_noise)
```

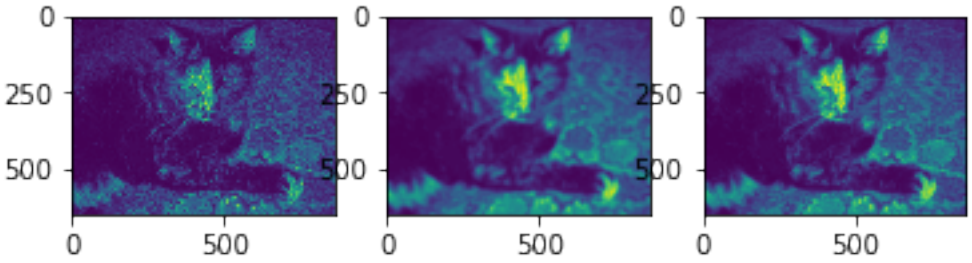
[14]:
```
<matplotlib.image.AxesImage at 0x1de71da0128>
```



(b) 将 8.1.2 和 8.1.3 中给出的每个滤波器 (小正方形滤波器，大正方形滤波器和高斯滤波器) 应用到实验 (a) 生成的噪声图像上。哪个滤波器的效果最好？

[15]:
```
fig, axes = plt.subplots(1,3)
small_square = sim.uniform_filter(Photo_noise, size = 3)
axes[0].imshow(small_square)
big_square = sim.uniform_filter(Photo_noise, size = 15)
axes[1].imshow(big_square)
gauss = sim.gaussian_filter(Photo_noise, sigma=3)
axes[2].imshow(gauss)
```

[15]:
```
<matplotlib.image.AxesImage at 0x1de71e68588>
```



1.3 8.3 特征强调

[ ]:
```
import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```

获取数据集 17stressFibers

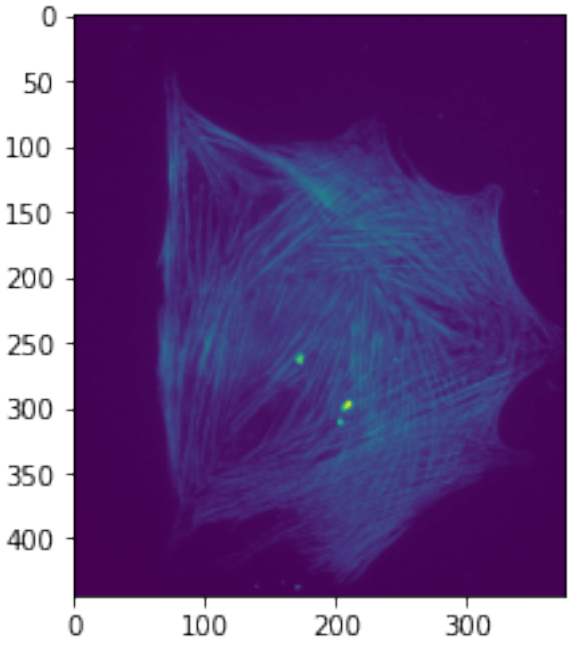
[16]:
```
filepath = r'D:\我的课程\计算物理\Datasets\17stressFibers\\'
stressFibers = np.load(filepath + 'stressFibers.npy')
```

调整数组比例

[17]:
```
#smin, smax = stressFibers.min(), stressFibers.max()
#stressFibers = (stressFibers - smin)/(smax-smin)*255
#stressFibers = stressFibers.astype('uint8')
plt.figure('original')
```

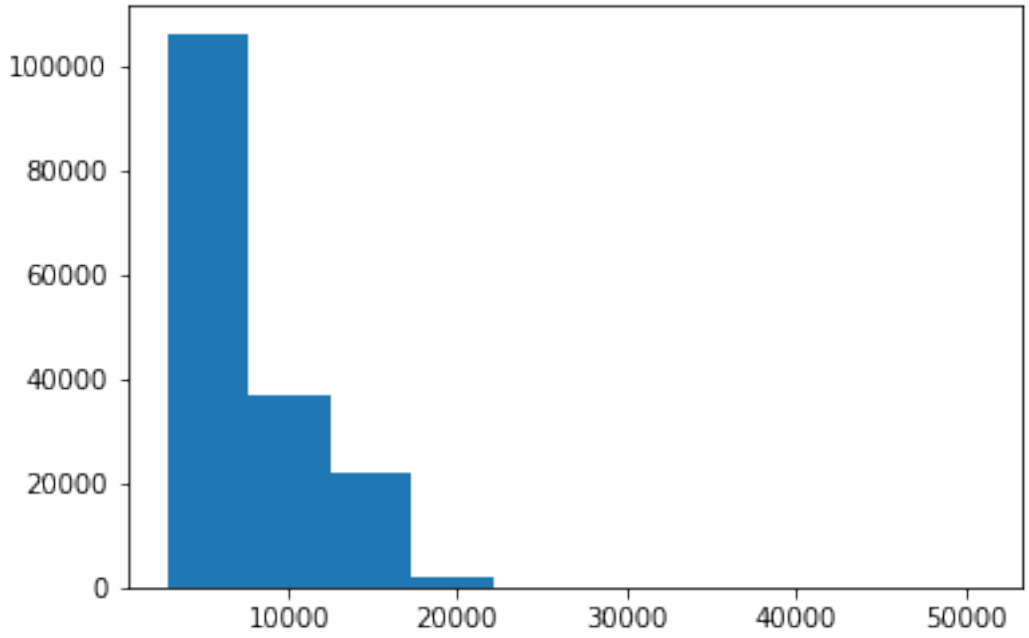
```
plt.imshow(stressFibers)
```

[17]: <matplotlib.image.AxesImage at 0x1de71ed3b00>



[18]:
```
plt.figure('hist')
plt.hist(stressFibers.ravel())
```

[18]:
```
(array([1.0634e+05, 3.6711e+04, 2.1779e+04, 1.9430e+03, 8.7000e+01,
        2.4000e+01, 1.8000e+01, 2.6000e+01, 6.0000e+00, 1.0000e+01]),
 array([ 2894. ,  7697.8, 12501.6, 17305.4, 22109.2, 26913. , 31716.8,
        36520.6, 41324.4, 46128.2, 50932. ]),
 <a list of 10 Patch objects>)
```

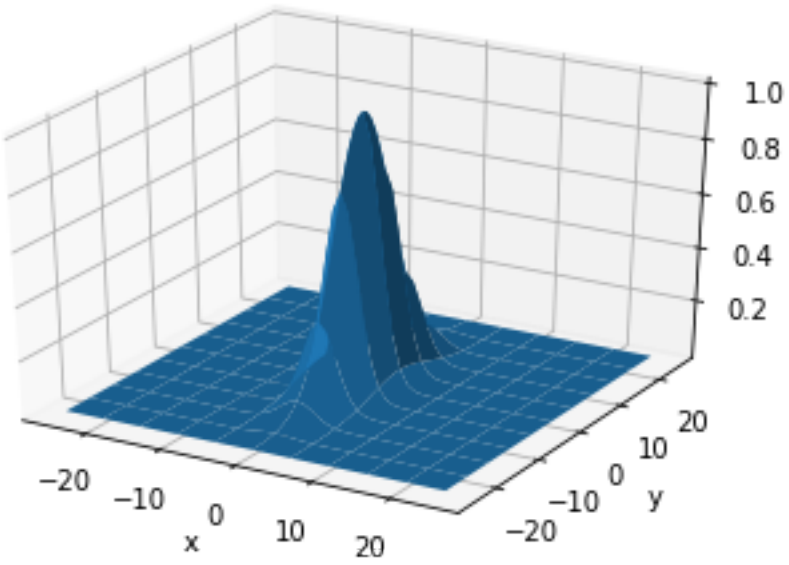


任务：

(a) 执行下面的代码，并绘制滤波器的曲面图，描述其中的显著特性

[20]:
```
from mpl_toolkits.mplot3d import Axes3D
v = np.arange(-25, 26)
X, Y = np.meshgrid(v, v)
gauss_filter = np.exp(-0.5*(X**2/5 + Y**2/45))
fig = plt.figure('Gauss filter')
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, gauss_filter, rstride=5, cstride=5)
ax.set_xlabel('x')
ax.set_ylabel('y')
```

[20]:
```
Text(0.5, 0, 'y')
```

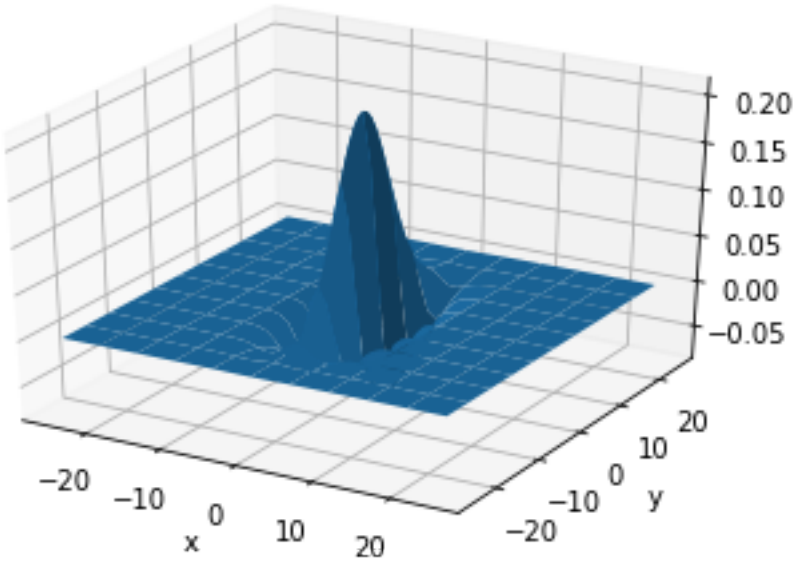


(b) 使用下面的 “黑箱” 代码修改实验 (a) 中的滤波器，然后使用生成的滤波器绘制曲面图，比较 combined_filter 和 gauss_filter 之间的共性和差异

[21]:
```
#laplace_filter = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
laplace_filter=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
combined_filter = sim.convolve(gauss_filter, laplace_filter)
```

[22]:
```
fig = plt.figure('combined filter')
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, combined_filter, rstride=5, cstride=5)
ax.set_xlabel('x')
ax.set_ylabel('y')
```

[22]:
```
Text(0.5, 0, 'y')
```

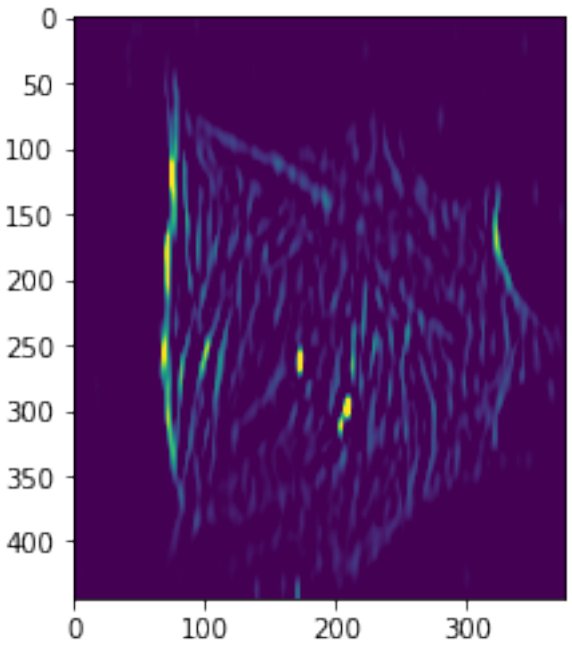


gauss_filter 强调垂直特性，combined_filter 强调这类对象的边缘。

(c) 调用 sim.convolve，将滤波器应用到纤维图像，显示并分析结果。

[23]:
```
fig = plt.figure('Combined Filter result')
ax = fig.add_subplot(111)
combined1 = sim.convolve(stressFibers, combined_filter)
ax.imshow(combined1, vmin=0,vmax=0.5*combined1.max())
combined1.min()
```

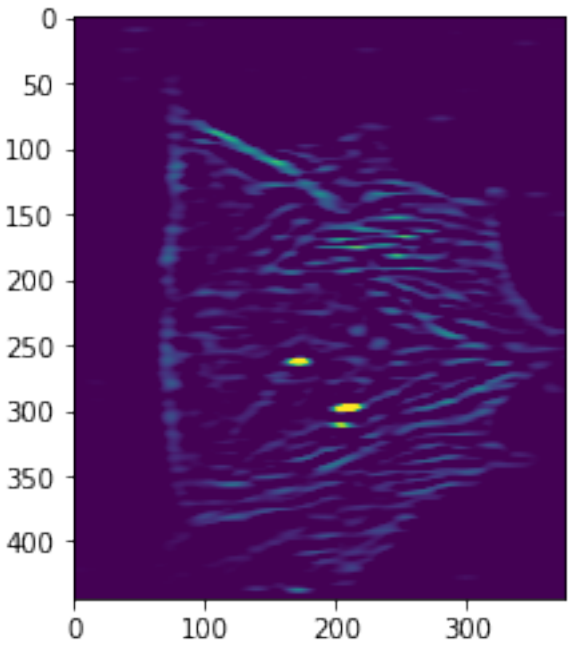
[23]:
```
-45812.58894866888
```



(d) 为强调水平对象，选取不同的 gauss_filter, 重复上面的步骤

[24]:
```
gauss_filter2 = np.exp(-0.5*(X**2/45 + Y**2/5))
#laplace_filter = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
laplace_filter=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
combined_filter2 = sim.convolve(gauss_filter2, laplace_filter)
fig = plt.figure('横滤波')
ax = fig.add_subplot(111)
combined2 = sim.convolve(stressFibers, combined_filter2)
ax.imshow(combined2, vmin=0,vmax=0.4*combined2.max())
```

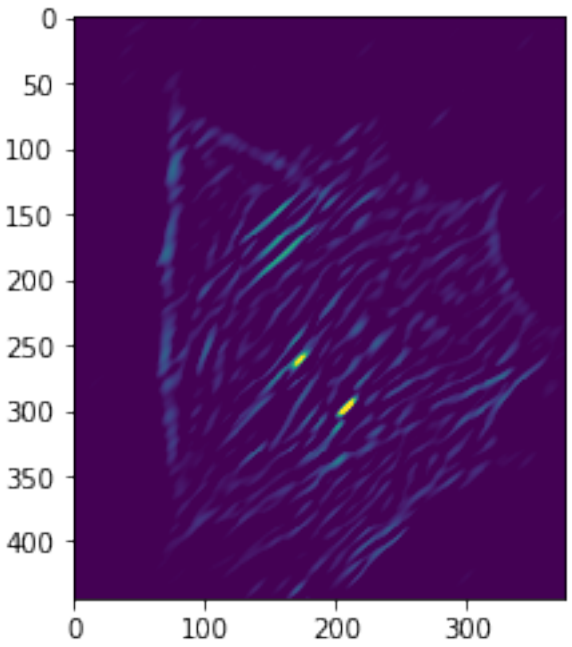
[24]: <matplotlib.image.AxesImage at 0x1de705b4f60>



选做实验：创建另外两个滤波器，实现强调相对于垂直 +/-45 度方向的对象

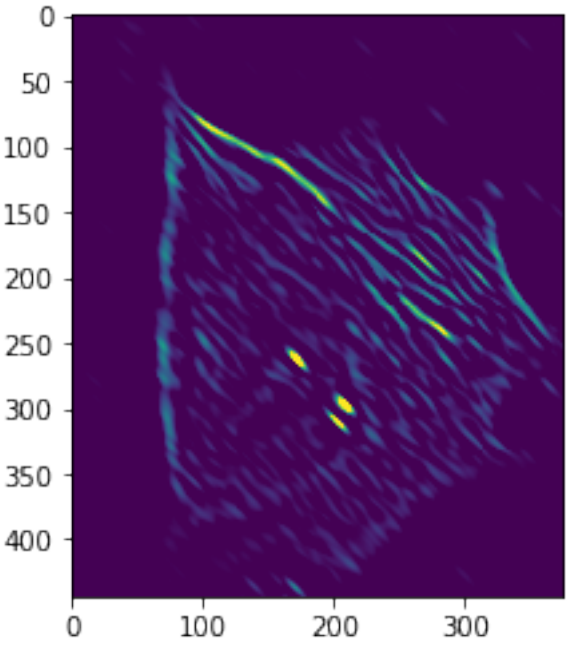
[25]:
```
gauss_filter3 = np.exp(-0.5*(X**2/45 + Y**2/5))
laplace_filter=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
combined_filter3 = sim.convolve(gauss_filter2, laplace_filter)
rotate_combined_filter3 = sim.rotate(combined_filter3, 45)
fig = plt.figure('45 度滤波')
ax = fig.add_subplot(111)
combined3 = sim.convolve(stressFibers, rotate_combined_filter3)
ax.imshow(combined3, vmin=0, vmax = 0.5*combined3.max())
```

[25]:
```
<matplotlib.image.AxesImage at 0x1de71884c50>
```



[26]:
```
gauss_filter4 = np.exp(-0.5*(X**2/45 + Y**2/5))
#laplace_filter = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
laplace_filter=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
combined_filter4 = sim.convolve(gauss_filter4, laplace_filter)
rotate_combined_filter4 = sim.rotate(combined_filter4, -45)
fig = plt.figure('-45 度滤波')
ax = fig.add_subplot(111)
combined4 = sim.convolve(stressFibers, rotate_combined_filter4)
ax.imshow(combined4, vmin=0, vmax = 0.5*combined4.max())
```

[26]: <matplotlib.image.AxesImage at 0x1de722bb470>


0
50
100
150
200
250
300
350
400
0
100
200
300

[ ]: