

绘制三维数据

一、实验目的

本次实验将学习如何使用 Matplotlib 中的 mplot3d 工具包绘制出三维点、线、等轮廓线、表面以及其他基本图形组件,通过图像窗口控件实现三维旋转和缩放。

二、实验内容

本次实验通过使用一个简单的多视图数据集(包括图像点、三维点、照相机参数矩阵)来可视化三维数据。

三、实验环境

实验平台类型	实验所用软件	软件所在位置
Ubuntu16.04	Jupyter notebook+计算机视觉库 OpenCV+OpenGL 库	/data

四、实验原理

为了可视化三维重建结果,我们需要绘制出三维图像。Matplotlib 中的 mplot3d 工具包可以方便地绘制出三维点、线、等轮廓线、表面以及其他基本图形组件,还可以通过图像窗口控件实现三维旋转和缩放。

五、实验步骤

1、点击“打开数据集”,在“计算机视觉/绘制三维数据”目录下下载文件“3D.tar.gz”,打开火狐浏览器的下载按钮。

Containers

计算机视觉 : / 绘制三维数据

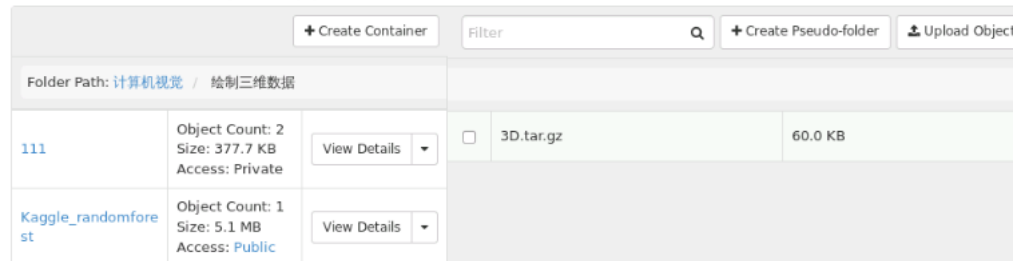


图 1 数据集下载

2、新建/data 目录,打开 jupyter notebook 软件,在“New”下点击 Python3 文件点击左侧的 Rename,新建 python3 文件“3Dplot.ipynb”。

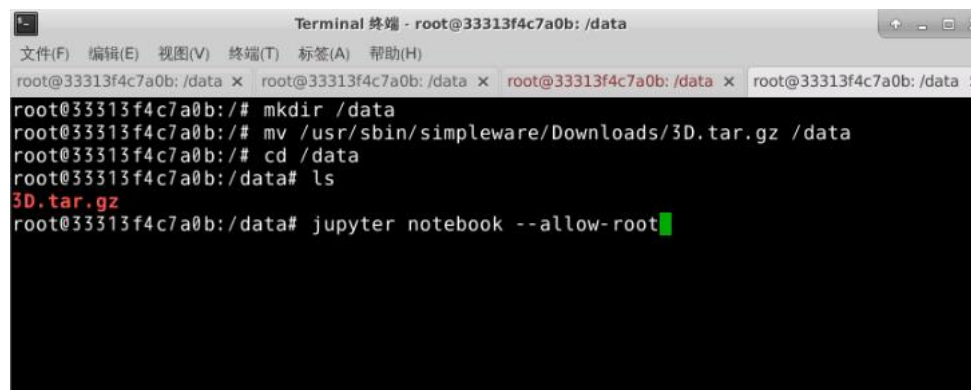


图 2 新建文件

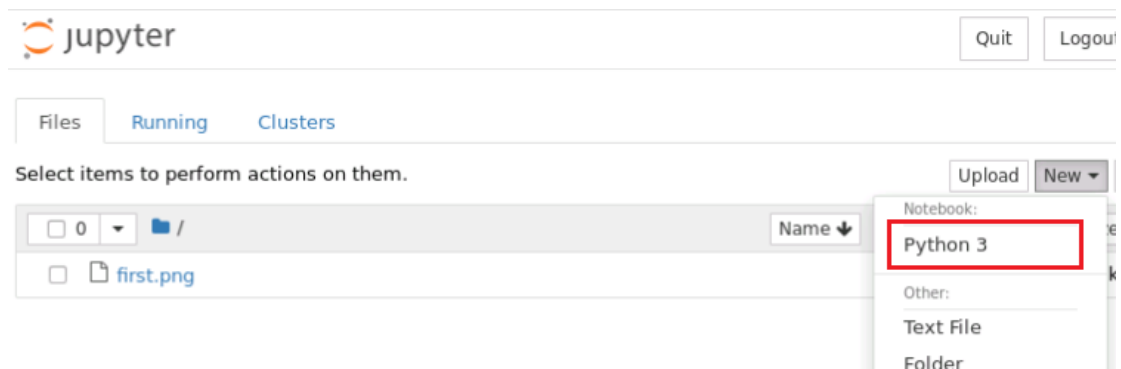


图 3 新建 jupyter 文件

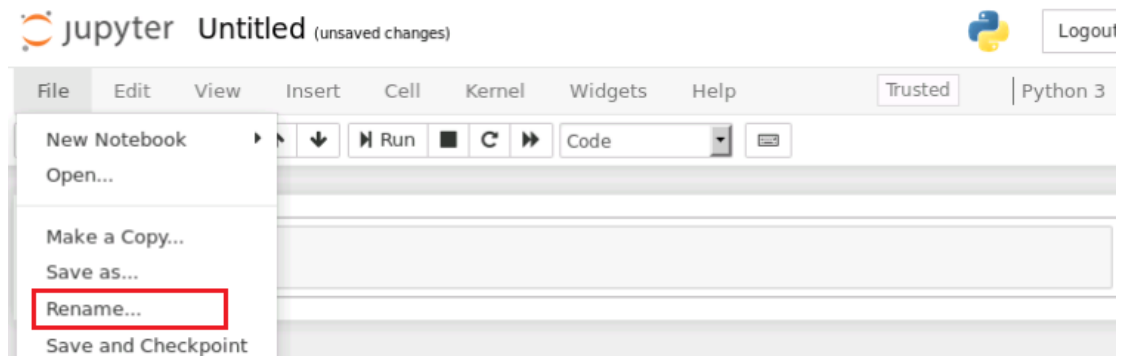


图 4 文件重命名

3、添加如下所示的三维库文件，并读取三维数据点绘制图像。

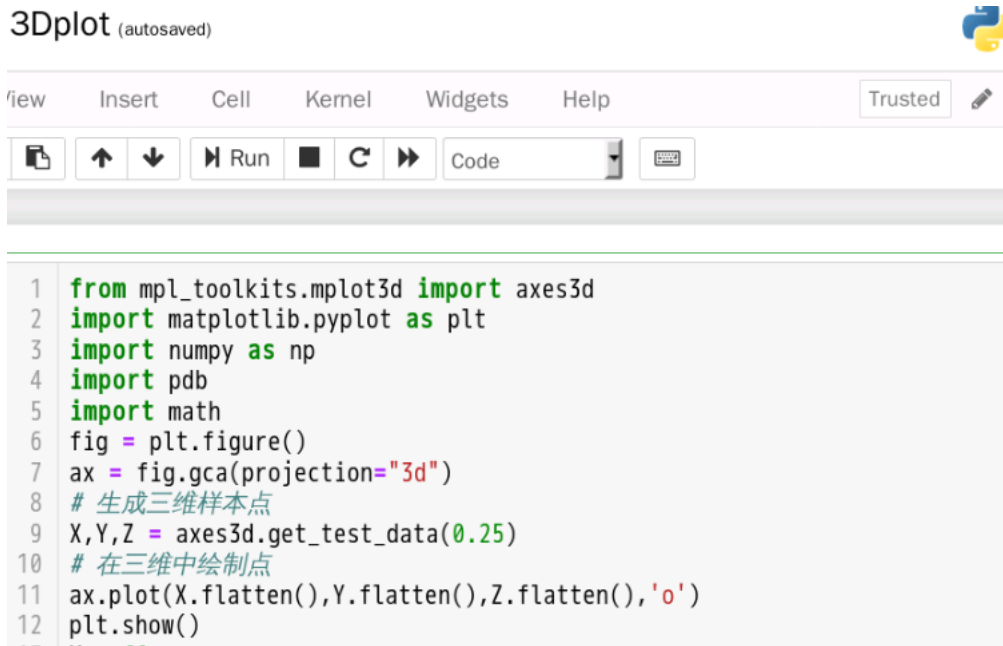
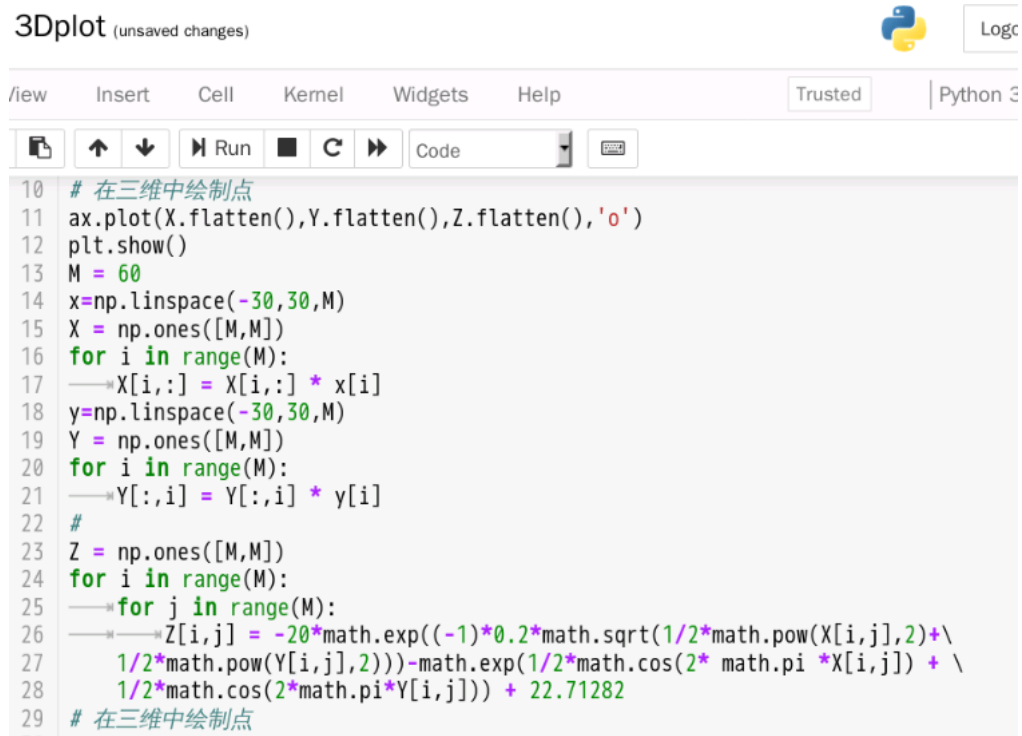


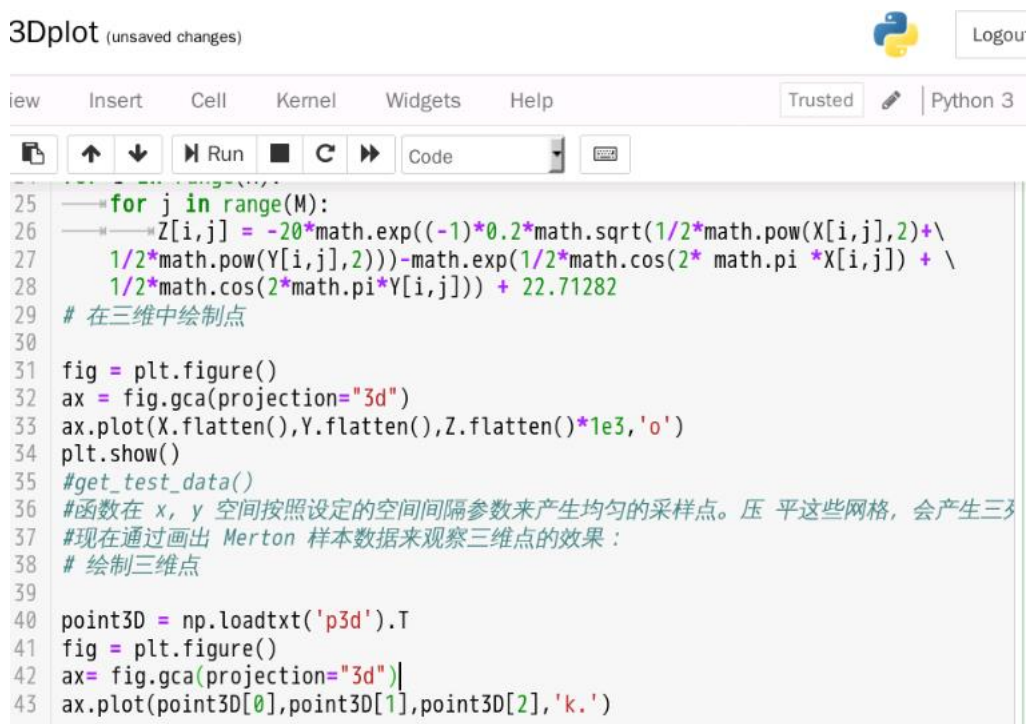
图 5 三维数据

4、以上步骤三维数据点由库提供，现在自己编写一个三维的 Ackley 函数，定义二元自变量的区间，并计算 Z 值。



```
10 # 在三维中绘制点
11 ax.plot(X.flatten(),Y.flatten(),Z.flatten(),'o')
12 plt.show()
13 M = 60
14 x=np.linspace(-30,30,M)
15 X = np.ones([M,M])
16 for i in range(M):
17     X[i,:] = X[i,:] * x[i]
18 y=np.linspace(-30,30,M)
19 Y = np.ones([M,M])
20 for i in range(M):
21     Y[:,i] = Y[:,i] * y[i]
22 #
23 Z = np.ones([M,M])
24 for i in range(M):
25     for j in range(M):
26         Z[i,j] = -20*math.exp((-1)*0.2*math.sqrt(1/2*math.pow(X[i,j],2)+\
27             1/2*math.pow(Y[i,j],2)))-math.exp(1/2*math.cos(2* math.pi *X[i,j]) + \
28             1/2*math.cos(2*math.pi*Y[i,j])) + 22.71282
29 # 在三维中绘制点
```


图 6 三维数据












```
25 for j in range(M):
26     Z[i,j] = -20*math.exp((-1)*0.2*math.sqrt(1/2*math.pow(X[i,j],2)+\
27         1/2*math.pow(Y[i,j],2)))-math.exp(1/2*math.cos(2* math.pi *X[i,j]) + \
28         1/2*math.cos(2*math.pi*Y[i,j])) + 22.71282
29 # 在三维中绘制点
30
31 fig = plt.figure()
32 ax = fig.gca(projection="3d")
33 ax.plot(X.flatten(),Y.flatten(),Z.flatten()*1e3,'o')
34 plt.show()
35 #get_test_data()
36 #函数在 x, y 空间按照设定的空间间隔参数来产生均匀的采样点。压平这些网格，会产生3D
37 #现在通过画出 Merton 样本数据来观察三维点的效果：
38 # 绘制三维点
39
40 point3D = np.loadtxt('p3d').T
41 fig = plt.figure()
42 ax= fig.gca(projection="3d")
43 ax.plot(point3D[0],point3D[1],point3D[2], 'k.')
```

图 7 三维数据

5、最后我们可以利用 matplotlib 中的 mplot3d 工具把三维重建的结果可视化出来，这个工具还能通过图像窗口控件实现三维旋转和缩放等功能。实现时只需要在 axes 对象中加上关键字 “projection= ‘3d’ ” 即可。

3Dplot (unsaved changes)  Logour

iew Insert Cell Kernel Widgets Help Trusted  Python 3

    Run    Code 

```
25 for j in range(M):
26     Z[i,j] = -20*math.exp((-1)*0.2*math.sqrt(1/2*math.pow(X[i,j],2)+\
27         1/2*math.pow(Y[i,j],2)))-math.exp(1/2*math.cos(2* math.pi *X[i,j]) + \
28         1/2*math.cos(2*math.pi*Y[i,j])) + 22.71282
29 # 在三维中绘制点
30
31 fig = plt.figure()
32 ax = fig.gca(projection="3d")
33 ax.plot(X.flatten(),Y.flatten(),Z.flatten()*1e3,'o')
34 plt.show()
35 #get_test_data()
36 #函数在 x, y 空间按照设定的空间间隔参数来产生均匀的采样点。压平这些网格，会产生三
37 #现在通过画出 Merton 样本数据来观察三维点的效果：
38 # 绘制三维点
39
40 point3D = np.loadtxt('p3d').T
41 fig = plt.figure()
42 ax= fig.gca(projection="3d")
43 ax.plot(point3D[0],point3D[1],point3D[2],'k.')
```

图 8 三维数据

6、在工具栏点击“Cell-->Run Cells”，三维效果如下，但此时还能不能旋转。将该代码文件新建为“3Dplot.py”，在终端执行命令“python3 3D.py”，则可以通过拖动图像进行旋转。

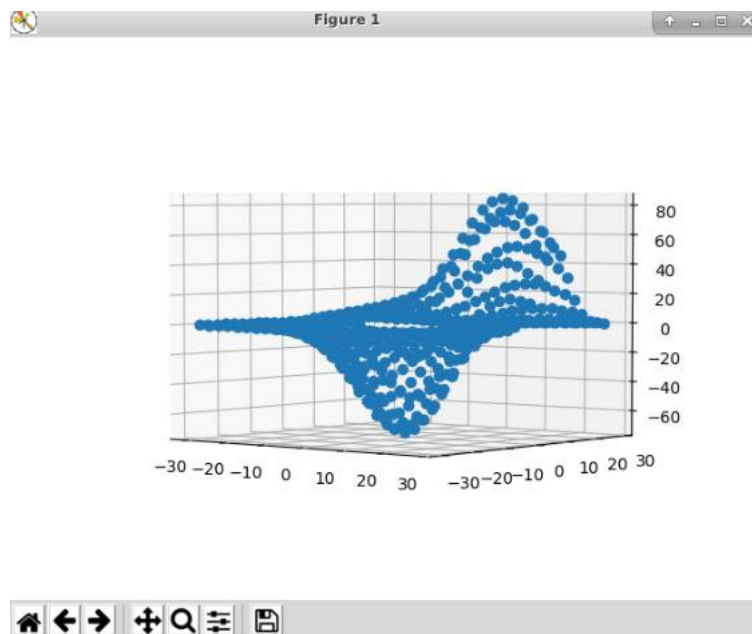


图 9 三维数据

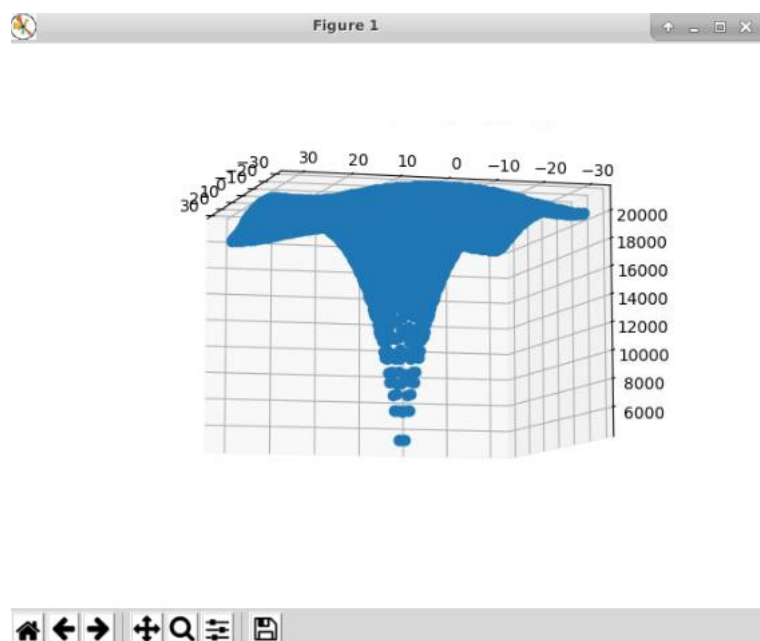


图 9 三维数据

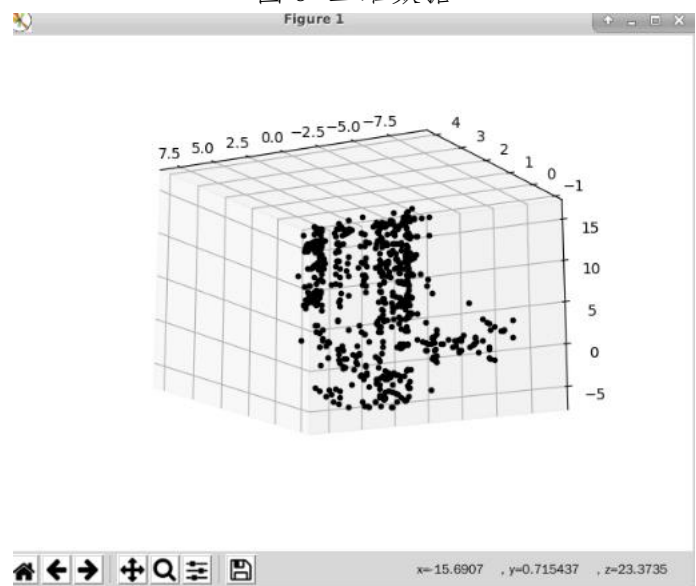


图 10 三维数据

代码:

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
import pdb
import math
fig = plt.figure()
ax = fig.gca(projection="3d")
# 生成三维样本点
X,Y,Z = axes3d.get_test_data(0.25)
# 在三维中绘制点
ax.plot(X.flatten(),Y.flatten(),Z.flatten(),'o')
plt.show()
M = 60
x=np.linspace(-30,30,M)
X = np.ones([M,M])
for i in range(M):
    X[i,:] = X[i,:] * x[i]
y=np.linspace(-30,30,M)
Y = np.ones([M,M])
for i in range(M):
    Y[:,i] = Y[:,i] * y[i]
#
Z = np.ones([M,M])
for i in range(M):
    for j in range(M):
        Z[i,j] = -20*math.exp((-
1)*0.2*math.sqrt(1/2*math.pow(X[i,j],2)+\
1/2*math.pow(Y[i,j],2)))-math.exp(1/2*math.cos(2* math.pi
*X[i,j]) + \
1/2*math.cos(2*math.pi*Y[i,j])) + 22.71282
# 在三维中绘制点

fig = plt.figure()
ax = fig.gca(projection="3d")
ax.plot(X.flatten(),Y.flatten(),Z.flatten()*1e3,'o')
plt.show()
#get_test_data()
#函数在 x, y 空间按照设定的空间间隔参数来产生均匀的采样点。压平这些
#网格,会产生三列数据点,然后我们可以将其输入 plot() 函数。这样,我们
#就可以在立体表面上画出三维点。
#现在通过画出 Merton 样本数据来观察三维点的效果:
# 绘制三维点
```

```
point3D = np.loadtxt('p3d').T  
fig = plt.figure()  
ax= fig.gca(projection="3d")  
ax.plot(point3D[0],point3D[1],point3D[2], 'k. ')
```