实例1：图像的手绘效果

DV03

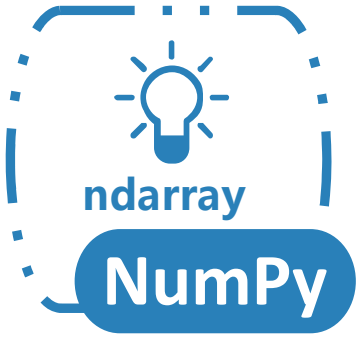
嵩天

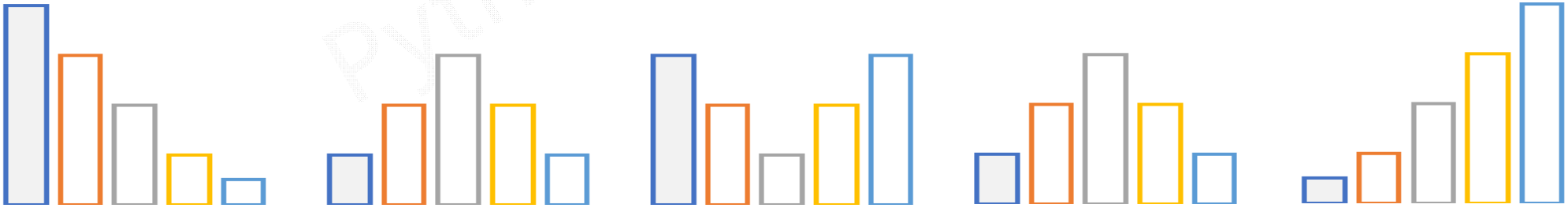
www.python123.org

python
弹指之间 · 享受创新

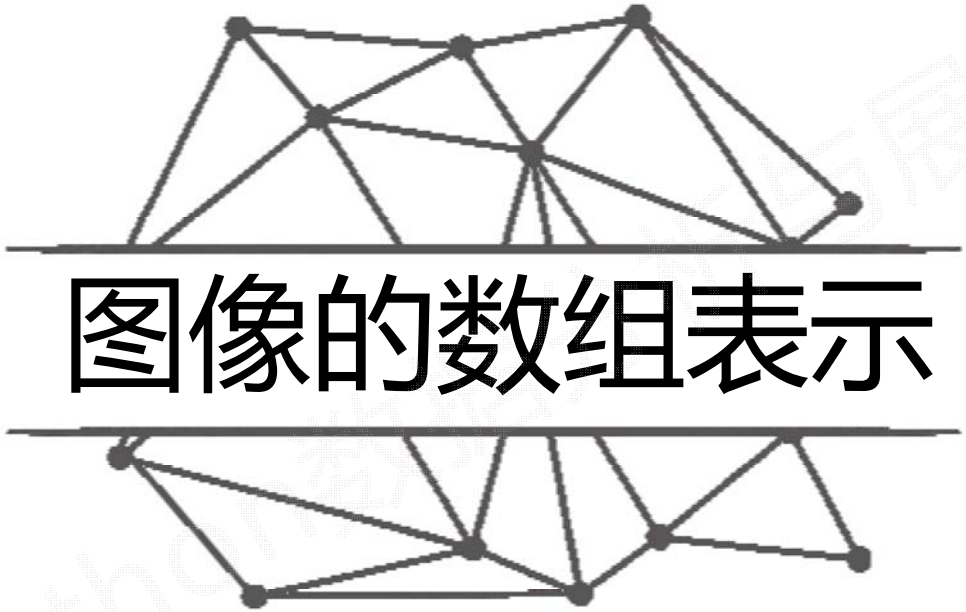
Python数据分析与展示

掌握表示、清洗、统计和展示数据的能力





O4X -Tian



图像的数组表示

图像的RGB色彩模式

图像一般使用RGB色彩模式，即每个像素点的颜色由红(R)、绿(G)、蓝(B)组成

图像的RGB色彩模式

RGB三个颜色通道的变化和叠加得到各种颜色，其中

- R 红色，取值范围，0-255
- G 绿色，取值范围，0-255
- B 蓝色，取值范围，0-255

RGB形成的颜色包括了人类视力所能感知的所有颜色。

PIL库

PIL， Python Image Library

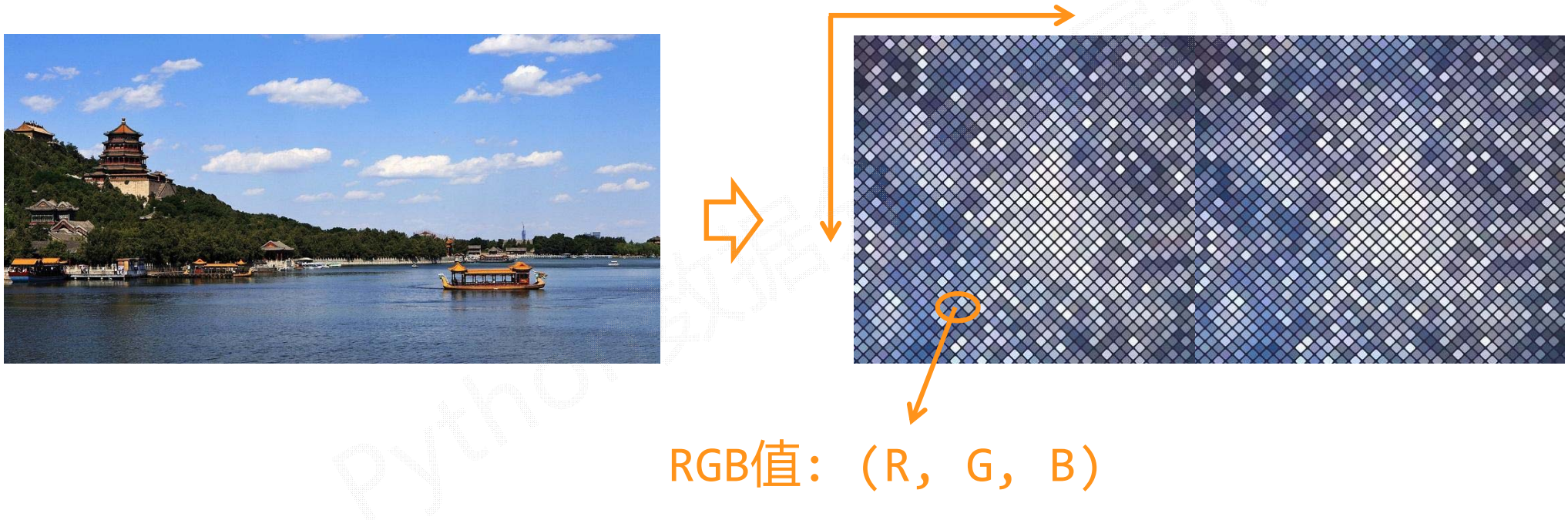
PIL库是一个具有强大图像处理能力的第三方库

在命令行下的安装方法： pip install pillow

```
from PIL import Image
```

Image是PIL库中代表一个图像的类（对象）

图像的数组表示

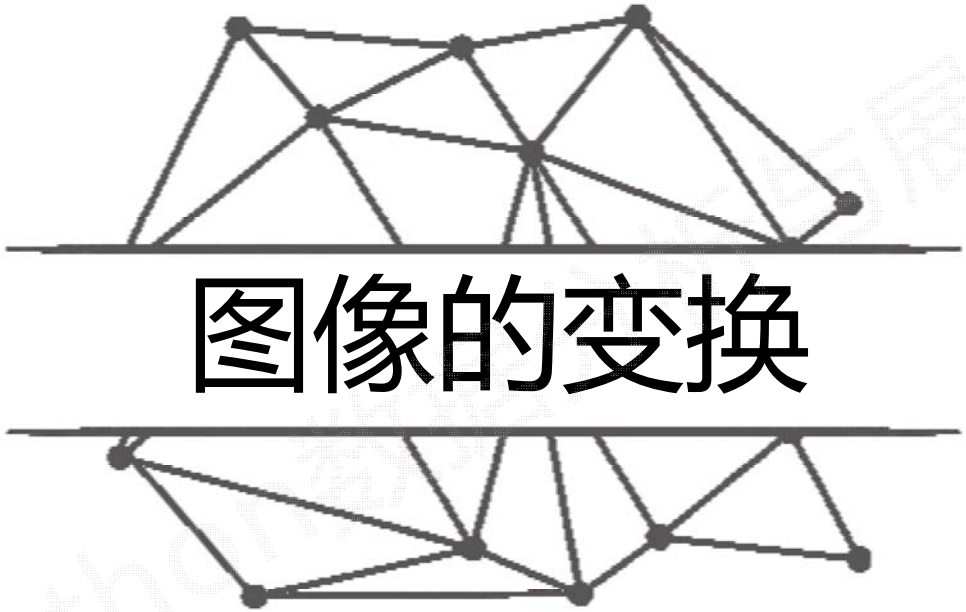


图像是一个由像素组成的二维矩阵，每个元素是一个RGB值

图像的数组表示

```
In [64]: from PIL import Image

In [65]: import numpy as np

In [66]: im = np.array(Image.open("D:/pycodes/beijing.jpg"))

In [67]: print(im.shape, im.dtype)
(669, 1012, 3) uint8
```

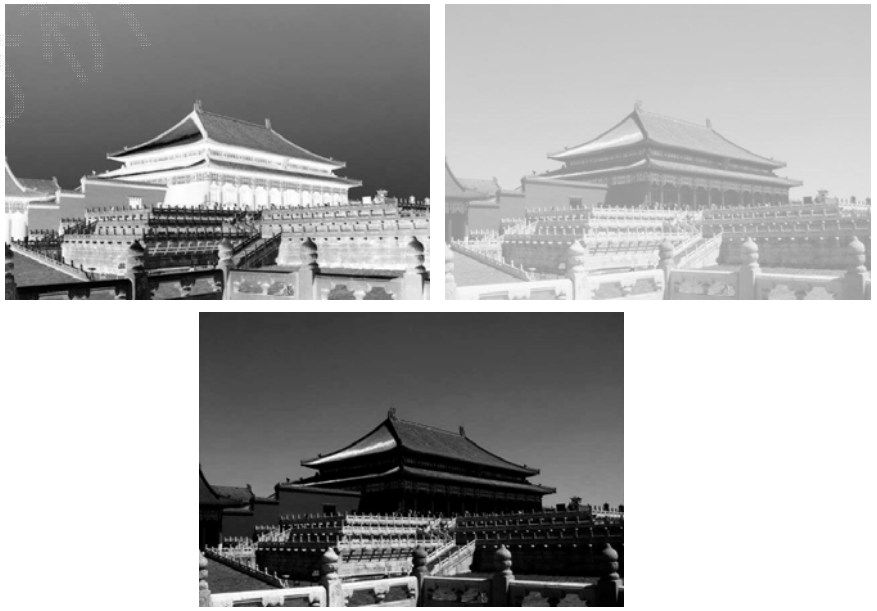
图像是一个三维数组，维度分别是高度、宽度和像素RGB值



图像的变换

图像的变换

读入图像后，获得像素RGB值，修改后保存为新的文件



图像的变换

```
In [81]: from PIL import Image

In [82]: import numpy as np

In [83]: a = np.array(Image.open("D:/pycodes/fcity.jpg"))

In [84]: print(a.shape, a.dtype)
(441, 634, 3) uint8

In [85]: b = [255, 255, 255] -a

In [86]: im = Image.fromarray(b.astype('uint8'))

In [87]: im.save("D:/pycodes/fcity2.jpg")
```

图像的变换

```
In [93]: from PIL import Image

In [94]: import numpy as np

In [95]: a = np.array(Image.open("D:/pycodes/fcity.jpg").convert('L'))

In [96]: b = 255 - a

In [97]: im = Image.fromarray(b.astype('uint8'))

In [98]: im.save("D:/pycodes/fcity3.jpg")
```

图像的变换

```
In [101]: from PIL import Image

In [102]: import numpy as np

In [103]: a = np.array(Image.open("D:/pycodes/fcity.jpg").convert('L'))

In [104]: c = (100/255)*a + 150 #区间变换

In [105]: im = Image.fromarray(c.astype('uint8'))

In [106]: im.save("D:/pycodes/fcity4.jpg")
```

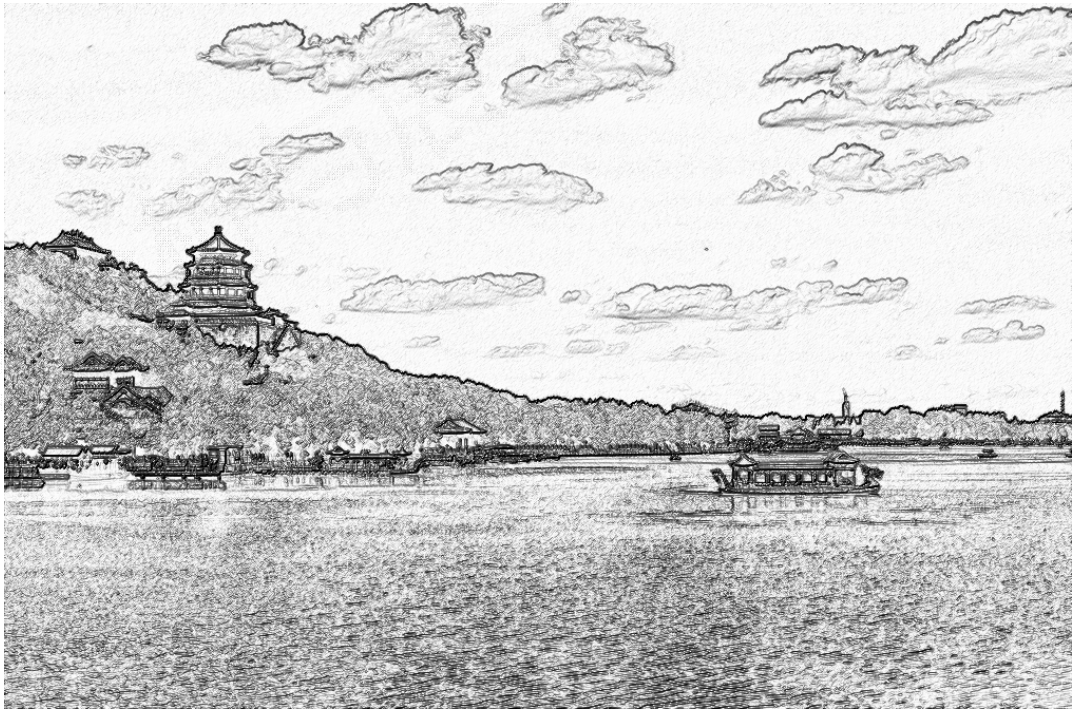
图像的变换

```
In [117]: from PIL import Image

In [118]: import numpy as np

In [119]: a = np.array(Image.open("D:/pycodes/fcity.jpg").convert('L'))

In [120]: d = 255 * (a/255)**2 #像素平方

In [121]: im = Image.fromarray(d.astype('uint8'))

In [122]: im.save("D:/pycodes/fcity5.jpg")
```

“图像的手绘效果”实例分析

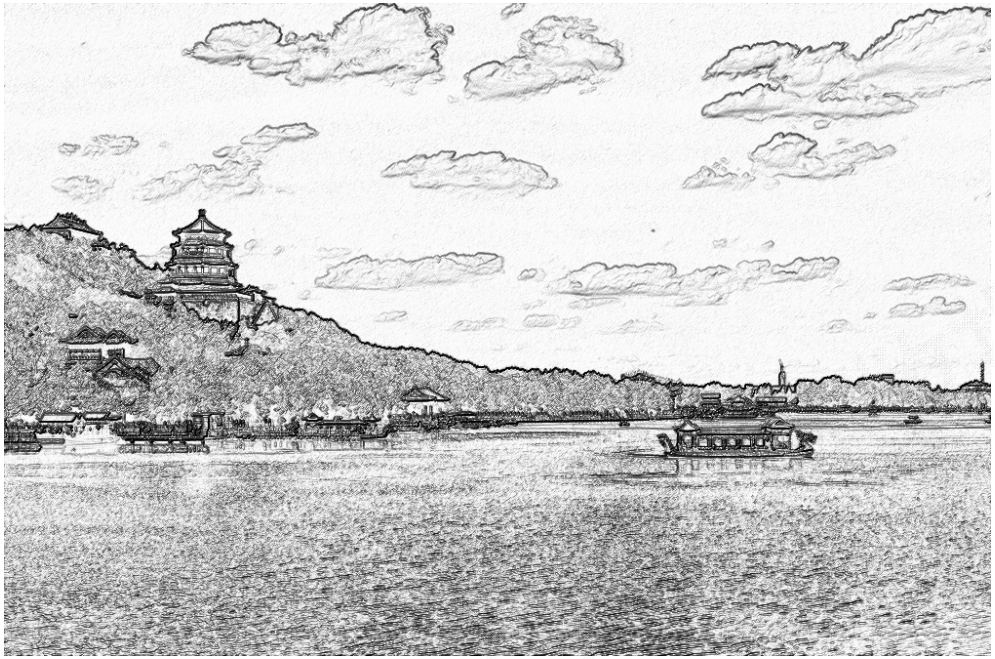
“图像的手绘效果”实例介绍

原图



手绘效果图

“图像的手绘效果”实例分析



手绘效果的几个特征：

- 黑白灰色
- 边界线条较重
- 相同或相近色彩趋于白色
- 略有光源效果

“图像的手绘效果”代码

```
from PIL import Image
import numpy as np

a = np.asarray(Image.open('D:/pycodes/beijing.jpg').convert('L')).astype('float')

depth = 10.                             # (0-100)
grad = np.gradient(a)                   #取图像灰度的梯度值
grad_x, grad_y = grad                   #分别取横纵图像梯度值
grad_x = grad_x*depth/100.
grad_y = grad_y*depth/100.
A = np.sqrt(grad_x**2 + grad_y**2 + 1.)
uni_x = grad_x/A
uni_y = grad_y/A
uni_z = 1./A

vec_el = np.pi/2.2                      # 光源的俯视角度，弧度值
vec_az = np.pi/4.                       # 光源的方位角度，弧度值
dx = np.cos(vec_el)*np.cos(vec_az)      #光源对x 轴的影响
dy = np.cos(vec_el)*np.sin(vec_az)      #光源对y 轴的影响
dz = np.sin(vec_el)                     #光源对z 轴的影响

b = 255*(dx*uni_x + dy*uni_y + dz*uni_z)     #光源归一化
b = b.clip(0,255)

im = Image.fromarray(b.astype('uint8'))      #重构图像
im.save('D:/pycodes/beijingHD.jpg')
```

“图像的手绘效果”实例编写

梯度的重构

利用像素之间的梯度值和虚拟深度值对图像进行重构

根据灰度变化来模拟人类视觉的远近程度

```
depth = 10.
grad = np.gradient(a)
grad_x, grad_y = grad
grad_x = grad_x*depth/100.
grad_y = grad_y*depth/100.
```

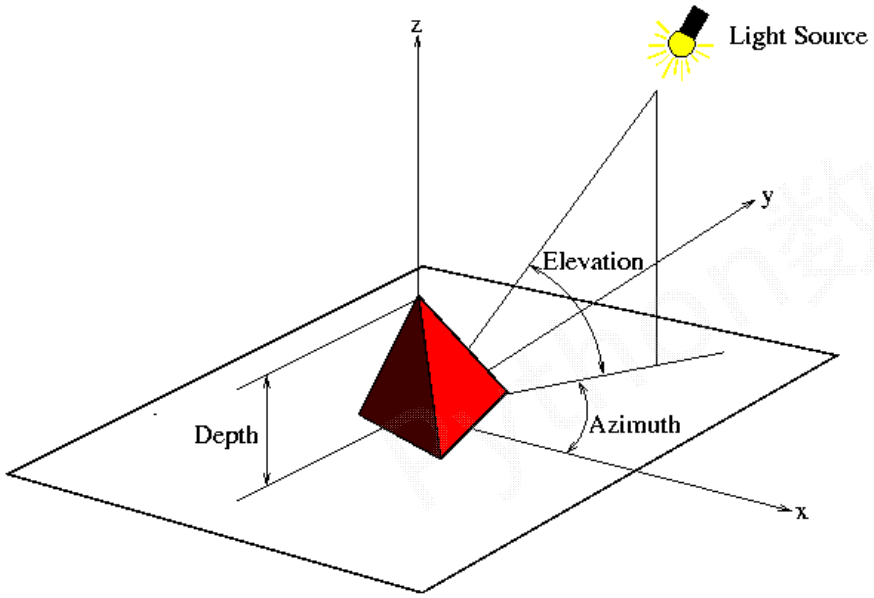
预设深度值为10　取值范围0-100

提取x和y方向的梯度值

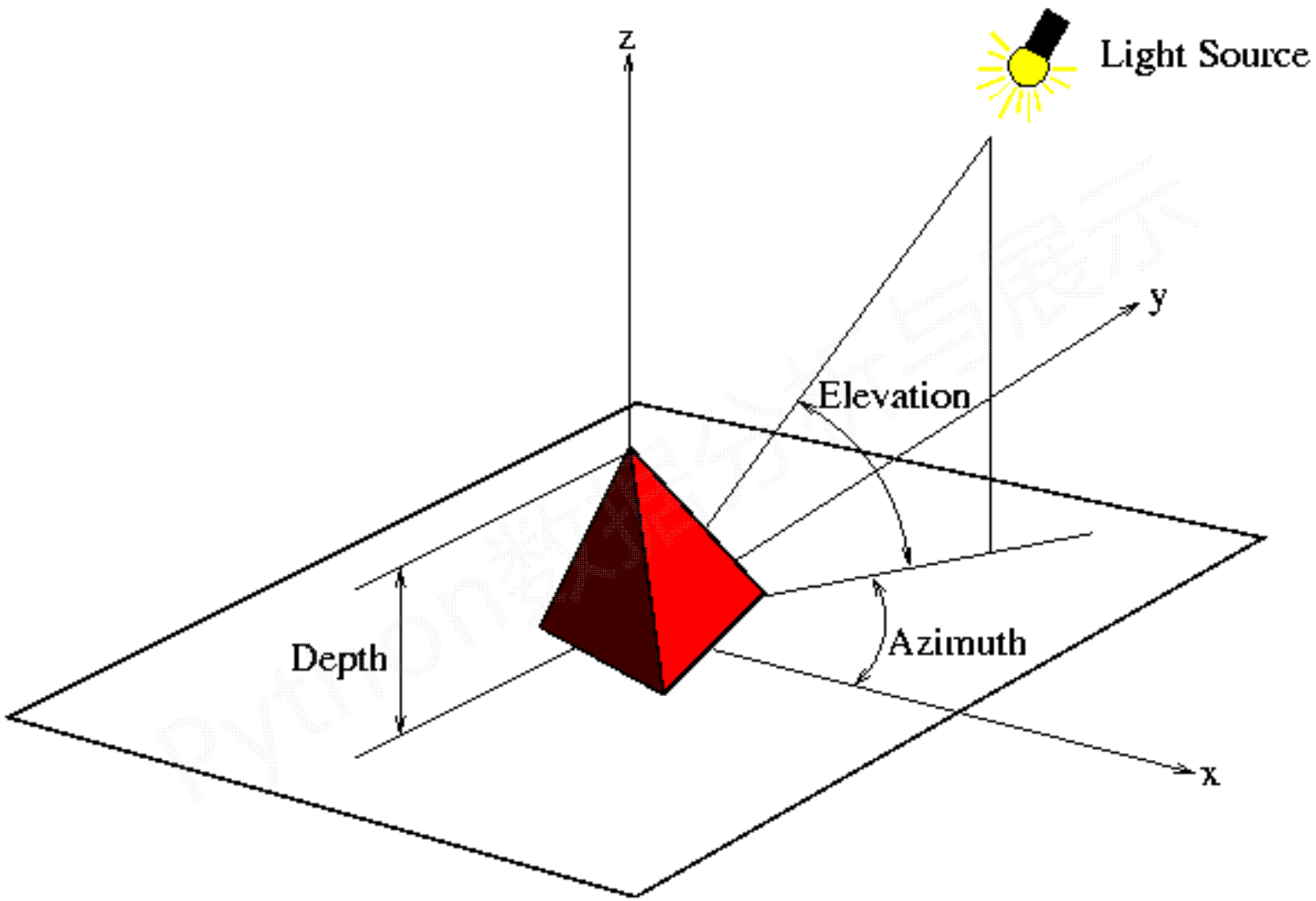
根据深度调整x和y方向的梯度值

光源效果

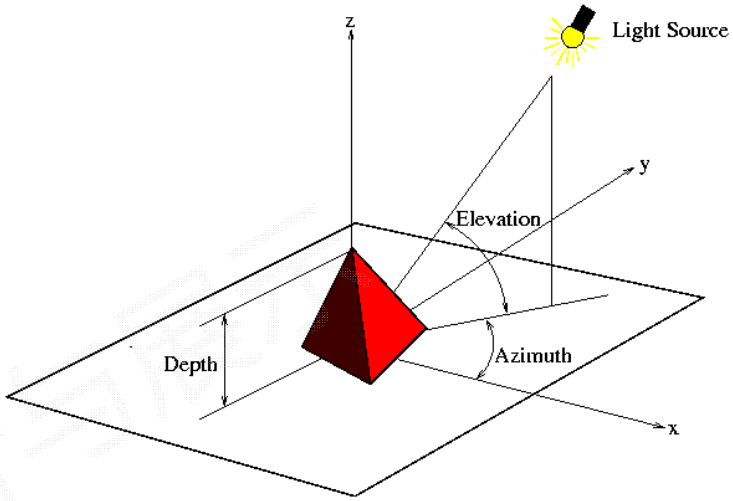
根据灰度变化来模拟人类视觉的远近程度



- 设计一个位于图像斜上方的虚拟光源
- 光源相对于图像的俯视角为Elevation，方位角为Azimuth
- 建立光源对个点梯度值的影响函数
- 运算出各点的新像素值


z
Light Source
y
Elevation
Depth
Azimuth
x

光源效果



np.cos(vec_el)为单位光线在地平面上的投影长度

```
vec_el = np.pi/2.2
vec_az = np.pi/4.
dx = np.cos(vec_el)*np.cos(vec_az)
dy = np.cos(vec_el)*np.sin(vec_az)
dz = np.sin(vec_el)
```

dx，dy，dz是光源对x/y/z三方向的影响程度

梯度归一化

构造x和y轴梯度的三维归一化单位坐标系

```
A = np.sqrt(grad_x**2 + grad_y**2 + 1.)
uni_x = grad_x/A
uni_y = grad_y/A
uni_z = 1./A
b = 255*(dx*uni_x + dy*uni_y + dz*uni_z)
```

梯度与光源相互作用，将梯度转化为灰度

图像生成

为避免数据越界，将生成的灰度值裁剪至0-255区间

```
b = b.clip(0,255)
im = Image.fromarray(b.astype('uint8'))
im.save('./beijingHD.jpg')
```

生成图像