实验：绘制 10 Bins 直方图

实验概要

Bins 就是数据分箱，也称为离散组合或数据分桶，是一种数据预处理技术，将原始数据分成小区间，即一个 Bin（小箱子），它是一种量子化的形式。数据分箱技术被广泛应用与信用评级、客户消费分级、时隙分段等多种业务中。

另外，之前曾提及过我们的实验中将这几段代码注释掉：

```
cv2.imshow('grayscale image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

这里再强调一次：当我们使用 Matplotlib 显示图像时，它将显示在笔记本中，而使用 OpenCV 的 cv2.imshow 函数显示的图像将显示在单独的窗口中 —— 而由于我们当前的实验环境是在容器平台上，直接弹出窗口会造成 Jupyter Notebook 的 Kernel 崩溃 —— 因此我们将上面的代码注释掉，全部替换为 Matplotlib，进行图像显示。然而，当你在自己的笔记本上执行实验操作的时候，你完全可以使用上面的代码，通过 cv2.imshow 函数来进行操作。

此外，默认情况下，使用 Matplotlib 显示的图像将具有网格线或 X 轴和 Y 轴。这些将不会出现在使用 cv2.imshow 函数显示的图像中。这是因为 Matplotlib 实际上是一个（图形）绘图库，这就是为什么它显示轴的原因，而 OpenCV 是一个计算机视觉库，并且轴在计算机视觉中并不重要。请注意，无论是否使用轴，无论使用 Matplotlib 还是 OpenCV 显示图像，图像都将保持不变。这就是为什么任何和所有图像处理步骤在 Matplotlib 和 OpenCV 之间也将保持不变的原因。在后面的实验中，我们将交替使用 Matplotlib 和 OpenCV 来显示图像。这意味着有时您会发现带有轴的图像，有时会发现没有轴的图像。在这两种情况下，轴都不重要，可以忽略。

实验目标

在本实验中，我们将读取灰度图像，将其绘制出来，然后绘制 10 个 bin 的 2D 直方图。

同时，我们还会将直方图的应用扩展到二值图像上。

另外，您还能了解到经常在单反相机或者其他图像处理软件上看到的 BGR 彩色图像直方图的制作原理。

1. 导入依赖库

In [1]:

```
import cv2                          # 导入OpenCV
import matplotlib.pyplot as plt     # 导入matplotlib

# 魔法指令，使图像直接在Notebook中显示
%matplotlib inline
```

2. 读取灰度图像

在这里我们使用了 cv2.imread 函数的 cv2.IMREAD_GRAYSCALE 选项，要求 OpenCV 使用灰度模式读取原图像。其中 cv2.IMREAD_GRAYSCALE 它指定以灰度模式加载图像。

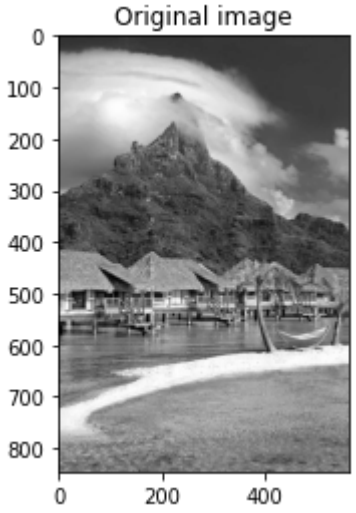
In [2]:

```
# 设置输入输出路径
import os
base_path = os.environ.get("BASE_PATH",'../data/')
data_path = os.path.join(base_path + "lab3/")
result_path = "result/"
os.makedirs(result_path, exist_ok=True)

# 指定以灰度模式加载图像
img= cv2.imread('./data/river_scene.jpg' , cv2.IMREAD_GRAYSCALE)
```

3. 显示图像

In [3]:

```
#cv2.imshow('grayscale image', img)
#cv2.waitKey(0)
#cv2.destroyAllWindows()

plt.imshow(img,cmap="gray")  # 使用matplotlib将图像喷绘成灰色
plt.title('Original image')  # 指定输出图像的标题
plt.show()                   # 显示图像
```

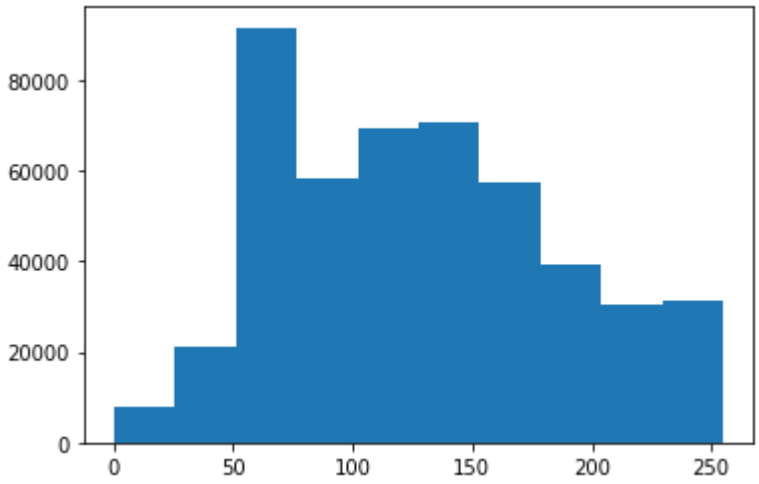


4. 绘制直方图

这里我们不需要指定分箱数量，默认情况下，不指定数量就是 10 个。

In [4]:

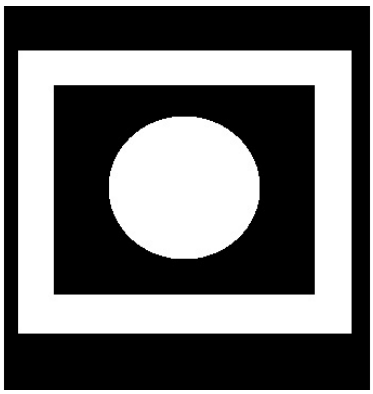
```
ax = plt.hist(img.ravel())
plt.show()
```



如果计算上图中的条数，您将看到它是 10 ，这是箱的默认数目。 这意味着我们可以知道每个强度范围内有多少像素（此范围由 25.6 像素组成，因为 256/10 = 25.6 ）位于每个条形段内。例如：第一栏告诉我们该图像中共有多少个强度在 0 到 25.6 之间的像素，但是你只能看到大致的数量值，以及与其他强度值的像素数量的对比，它不能确切告诉我们该图像中有多少个具有特定强度值的像素。

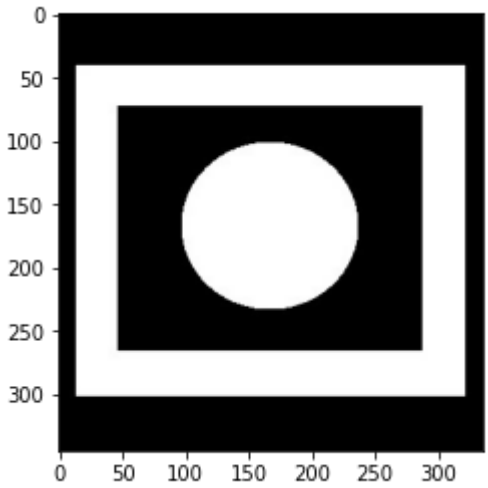
5. 二值图像直方图

与之前的实验相似，我们可以使用以下代码观察具有 256 个 bin 的二值图像的直方图。让我们尝试看看实际效果。我们将使用下图中所示的二进制图像。



In [5]:

```
# 指定以灰度模式加载图像
binaryImg= cv2.imread('./data/binaryImage.jfif',
                      cv2.COLOR_BGR2GRAY)

plt.imshow(binaryImg, cmap='gray') # 使用灰色“喷涂”图像输出显示
plt.show()                         # 显示图像
```

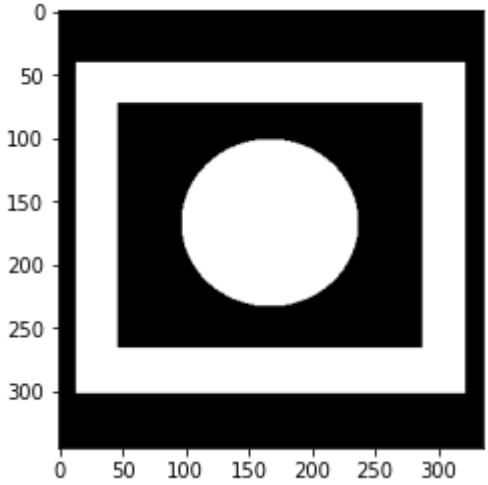


由于上面的图片来源于互联网，经过多次转换后可能已经并非标准的二值图像，但通过肉眼很难区别。

因此，我们先对图片执行严格的二值化：

In [6]:

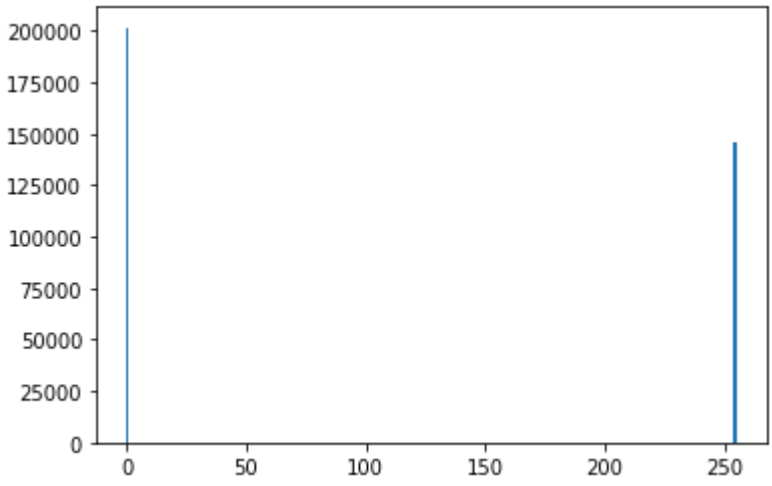
```
# 设置阈（yu）值和最大值
# 设置阈值，小于等于阈值的像素将被替换为0
thresh = 150
# 设置最大值，大于阈值的像素将被替换为这里设置的最大值
maxValue = 255

# 执行图像二值化
th, dst = cv2.threshold(binaryImg, thresh, maxValue,
                        cv2.THRESH_BINARY)

plt.imshow(dst, cmap='gray') # 使用灰色“喷涂”图像输出显示
plt.show()                   # 显示图像
```



Type *Markdown* and LaTeX: α^2

In [7]:

```
# 排列dst图像内像素的向量，并指定将分箱数划分为256个
binaryAx = plt.hist(dst.ravel(), bins=256)
# 显示图像
plt.show()
```



请注意，二值图像上只有两种可能的强度（ 0 和 255 ），因此直方图上只有两条线。

上面的直方图告诉我们，二进制图像上的 0 （黑色区域）比 1 （白色区域）要多。

6. BGR 多通道直方图

另一方面，对于 BGR 图像，它具有三个通道，因此它将具有三个 2D 直方图 —— 每个通道一个。

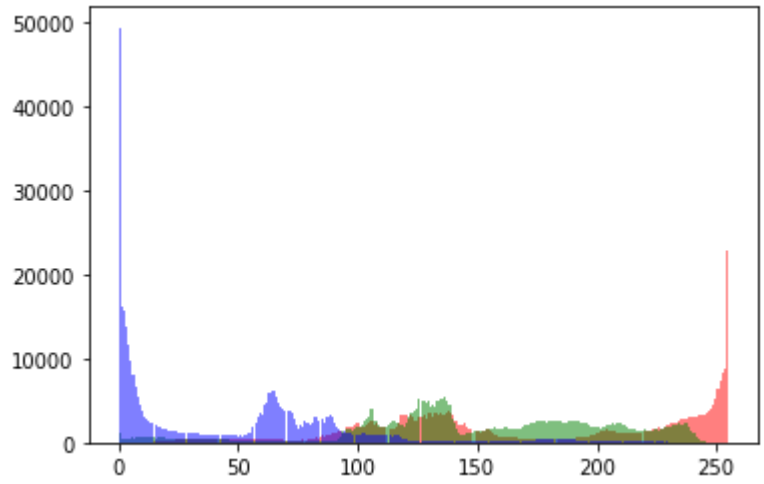
要在单个图上以不同颜色绘制三个通道（ B， G， R ）的三个二维直方图，可以在每个通道上使用 `plt.hist()` 命令后调用 `plt.show()` 命令 。

由于OpenCV 的图像按照 BGR 顺序排列，因此通道的序号分别是 `B=0`； `G=1`； `R=2`；

为了便于理解，下面的代码按照 RGB 的顺序提取每个通道的像素值，画出直方图。

In [8]:

```
# 加载图像
img= cv2.imread('./data/sunflower.jpg')

# 指定读取图像的红色通道
im= img[:, :, 2]
# 绘制图像的红色通道的直方图，透明度（alpha为0.5）
plt.hist(im.ravel(), bins = 256, color = 'Red', alpha = 0.5)

# 指定读取图像的绿色通道
im= img[:, :, 1]
# 绘制图像的绿色通道的直方图，透明度（alpha为0.5）
plt.hist(im.ravel(), bins = 256, color = 'Green', alpha = 0.5)

# 指定读取图像的蓝色通道
im= img[:, :, 0]
# 绘制图像的蓝色通道的直方图，透明度（alpha为0.5）
plt.hist(im.ravel(), bins = 256, color = 'Blue', alpha = 0.5)

# 显示图像
plt.show()
```



如果您喜欢摄影，上面的直方图您应该会非常熟悉，我们在单反相机、修图工具软件上经常会看到类似的直方图。现在您应该了解到：这些图形实际上代表的是在你所拍摄的照片中三个色彩通道的像素值（亮度）分布。下图的单反相机，便是将实时图像的三通道直方图分别显示了出来，您也可以稍微修改上面的代码，通过独立显示三个通道的直方图，以制作同样的效果。

实验小结

在本实验中，您了解了如何制作灰度图像的 10 Bins 直方图，二值图像直方图，以及 BGR 彩色图像的直方图。