实验：Hu 矩轮廓匹配

实验概要

Hu 矩

在本实验中，我们将学习如何在数值上找到两个不同轮廓的形状之间的差异。这种差异是基于 Hu 矩 （也称为： 胡矩 / Hu 矩不变量 ）发现的，轮廓的 Hu 矩 是描述轮廓形状的七个数字。详细描述可参考官方文档(https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html)。

$$\mathrm{hu}[0] = \eta_{20} + \eta_{02}$$
$$\mathrm{hu}[1] = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$\mathrm{lu}[2] = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$\mathrm{lu}[3] = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\ln u[4] = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2\right.$$
$$hu[5] = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$hu[6] = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2\right.$$

OpenCV 通过以下方式计算 Hu 矩 ——

1. 用命令计算图像矩（图像像素强度的加权平均值）：

```
img_moments= cv2.moments(image)
```

2. 将图像矩 cv2.HuMoments OpenCV 函数：

```
hu_moments= cv2.HuMoments(img_moments)
```

3. 然后，将此数组展平以获取 Hu 矩 的特征向量：

```
hu_moments= hu_moments.flatten()
```

该特征向量的一行包含七列（七个数值），下表显示了一些样本轮廓形状的 Hu 矩 向量：

CONTOUR	Hu moment vector H						
	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
	$1.401e^{-03}$	$1.306e^{-08}$	$1.232e^{-10}$	$4.882e^{-11}$	$-3.720e^{-21}$	$5.302e^{-15}$	$-7.084e^{-22}$
	$1.258e^{-03}$	$5.039e^{-09}$	$1.130e^{-16}$	$4.176e^{-13}$	$2.869e^{-27}$	$-2.964e^{-17}$	$0.000e^{+00}$
	$1.453e^{-03}$	$2.918e^{-07}$	$4.502e^{-11}$	$1.201e^{-12}$	$8.828e^{-24}$	$-1.651e^{-17}$	$-4.462e^{-25}$
	$6.744e^{-03}$	$2.940e^{-06}$	$2.968e^{-08}$	$6.638e^{-09}$	$-6.418e^{-17}$	$1.138e^{-11}$	$-6.754e^{-17}$
	$1.573e^{-03}$	$5.884e^{-07}$	$4.206e^{-10}$	$1.400e^{-10}$	$2.985e^{-20}$	$1.019e^{-13}$	$-1.627e^{-20}$
	$1.573e^{-03}$	$5.884e^{-07}$	$4.206e^{-10}$	$1.400e^{-10}$	$2.985e^{-20}$	$1.019e^{-13}$	$1.627e^{-20}$

对于胡矩向量，请考虑以下因素：

- 即使通过反射，平移，缩放或旋转来变换图像，前六个值仍保持不变。这意味着，两个基本形状相同的轮廓将具有几乎相等的力矩值，即使其中一个被调整大小，旋转，沿任何方向翻转或改变其在图像中的位置或位置。
- 如果轮廓的图像被翻转，则第七个值会更改符号（正或负）。

使用 Hu 矩进行轮廓匹配

以下代码给出了使用 `Hu 矩` 执行轮廓匹配（两个形状的比较，因此有的人也叫：形状匹配）的命令。它将为您提供一个数值，描述两个形状之间的差异：

```
contour_difference = cv2.matchShapes (contour1, contour2, compar_method, parameter)
```

此输出变量的值（ `contour_difference` ）越小，两个轮廓之间的相似性就越大。关于上述公式，请注意以下几点：

- `contour1` , `contour2` ： 是要比较的两个单独的轮廓对象。
- `compar_method` ： 是比较方法。可以是 `1` 到 `3` 的整数，也可以是对应的字符串命令，如下表所示：

Comparison method	Numeric command	String command
CONTOURS_MATCH_I1	1	cv.CONTOURS_MATCH_I1
CONTOURS_MATCH_I2	2	cv.CONTOURS_MATCH_I2
CONTOURS_MATCH_I3	3	cv.CONTOURS_MATCH_I3

在大多数情况下，您会发现 `cv.CONTOURS_MATCH_I1` 效果最佳。但是，在处理项目时，请对所有三个项目进行测试，以查看哪个最适合您的数据。

- `parameter`： 是与所选比较方法相关的值，在最新版本的 OpenCV 中已没有什么作用。但是请注意，如果不将这个参数传递给 `cv2.matchShapes` 函数，则可能会出错。因此，我们通常传递 `0` 。

举例：比较以下两个轮廓：

要查找它们之间的数值差，则可以编写以下命令：

```
contour_difference = cv2.matchShapes (binary_im1, binary_im2, 1, 0)
```

上述的代码使用 `CONTOURS_MATCH_I1` 距离，告诉我们这两个形状的 `Hu矩 矢量` 之间的差异。在此示例中，结果为 `0.1137` 。

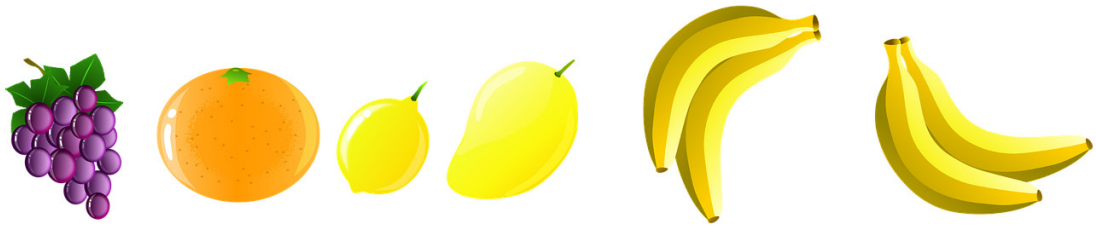
模板匹配

现在，您已经了解了如何匹配可以由单个斑点描述的形状。 如果要匹配具有两个斑点而不是一个斑点的符号怎么办？ 例如，一个问号：

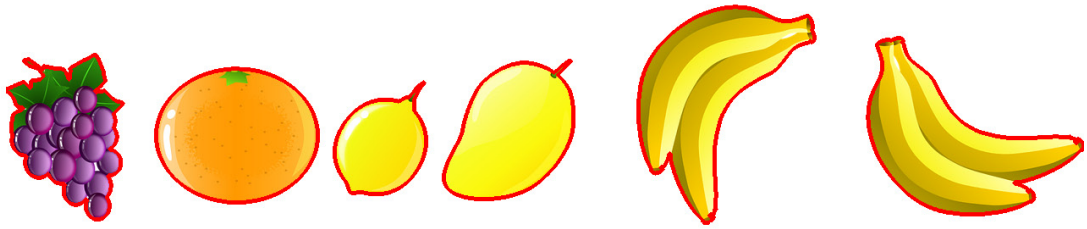
它有两个轮廓。 如果要在图像中找到该符号，则简单的轮廓匹配将不起作用。为此，您将使用另一种类似的技术，称为模板匹配。我们暂不展开讨论，详细描述可参考官方文档(https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html)。

实验目标

在本实验中，我们为后续的轮廓匹配应用实验做前期准备 —— 您将获得以下水果图片：



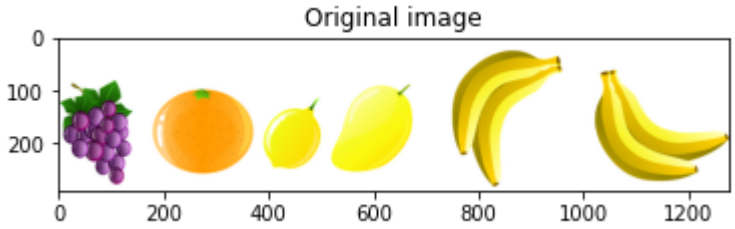
您的任务是检测此图像中存在的所有水果。通过轮廓检测执行此操作，并在图像上绘制轮廓形状，如下所示，以红色绘制水果轮廓：



1. 导入依赖库并加载图像

In [1]:

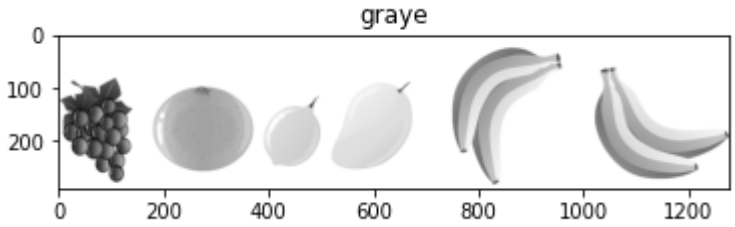
```
import cv2                          # 导入OpenCV
import matplotlib.pyplot as plt    # 导入matplotlib

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

# 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 + "lab4/")
result_path = "result/"
os.makedirs(result_path, exist_ok=True)

# 读取本地图像
image = cv2.imread('./data/many fruits.png')

# 在云实验环境下忽略以下代码，避免程序尝试打开系统窗口显示图片；
# 使用matplotlib替换，使图像直接在 Jupyter Notebook 中输出。

# cv2.imshow( 'Original image' , image )
# cv2.waitKey(0)
# cv2.destroyAllWindows()

plt.imshow(image[:,:,::-1])  # 将图像从 BGR 转换为 RGB
plt.title('Original image')  # 指定输出图像的标题
plt.show()                   # 显示图像
```



2. 转换灰度图像

In [2]:

```
# 将图像转换为灰度图像
gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

# 在云实验环境下忽略以下代码，避免程序尝试打开系统窗口显示图片；
# 使用matplotlib替换，使图像直接在 Jupyter Notebook 中输出。

# cv2.imshow( 'gray' , gray_image )
# cv2.waitKey(0)
# cv2.destroyAllWindows()

# 使用灰色“喷涂”图像输出显示
plt.imshow(gray_image, cmap='gray')
# 指定输出图像的标题
plt.title('graye')
# 显示图像
plt.show()
```

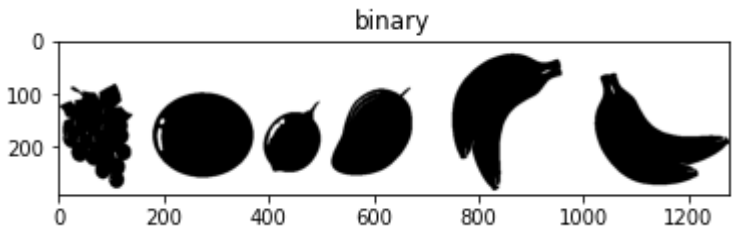


3. 图像二值化

使用适当的阈值将其转换为二值图像并显示。您选择的阈值必须将水果的外部边界作为单个对象。当然，每个水果的中间都可以保留一些空间：

In [3]:

```
# 直接输入阈值与最大值，执行图像二值化
ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)

# 在云实验环境下忽略以下代码，避免程序尝试打开系统窗口显示图片；
# 使用matplotlib替换，使图像直接在 Jupyter Notebook 中输出。

# cv2.imshow( 'binary' , binary_im )
# cv2.waitKey(0)
# cv2.destroyAllWindows()

plt.imshow(binary_im, cmap='gray') # 使用灰色“喷涂”图像输出显示
plt.title('binary')                # 指定输出图像的标题
plt.show()                         # 显示图像
```

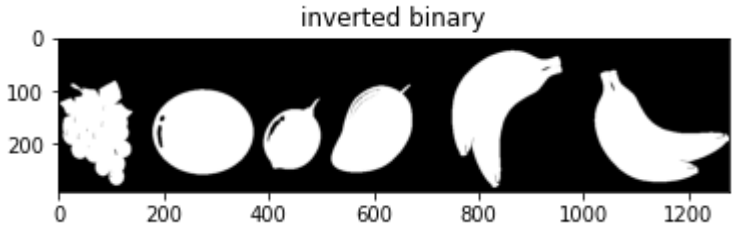


4. 反转图像

由于在 OpenCV 中需要黑色背景上的白色前景来进行轮廓检测，因此我们将按以下方式反转此图像：

In [4]:

```
binary_im= ~binary_im # 反转图像（黑白像素互换）

# 在云实验环境下忽略以下代码，避免程序尝试打开系统窗口显示图片；
# 使用matplotlib替换，使图像直接在 Jupyter Notebook 中输出。

# cv2.imshow( 'inverted binary' , binary_im )
# cv2.waitKey(0)
# cv2.destroyAllWindows()

plt.imshow(binary_im, cmap='gray') # 使用灰色“喷涂”图像输出显示
plt.title('inverted binary')       # 指定输出图像的标题
plt.show()                         # 显示图像
```

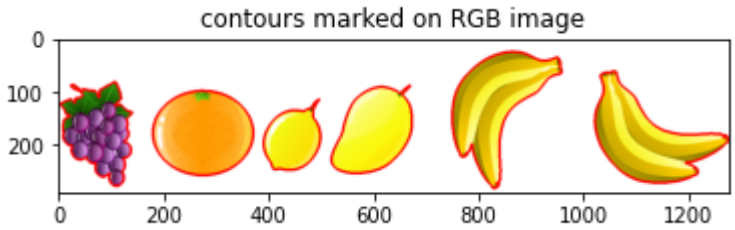


5. 轮廓检测

请注意，水果内部有一些空像素，因此我们接下来将使用 cv2.RETR_EXTERNAL 选项，进行外部轮廓检测：

In [5]:

```
# 从二值图像中找到外部轮廓
contours,hierarchy = cv2.findContours(binary_im,
                            cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

# 绘制轮廓
with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)

# 在云实验环境下忽略以下代码，避免程序尝试打开系统窗口显示图片；
# 使用matplotlib替换，使图像直接在 Jupyter Notebook 中输出。

# cv2.imshow( 'contours marked on RGB image' , with_contours )
# cv2.waitKey(0)
# cv2.destroyAllWindows()

plt.imshow(with_contours[:,:,::-1])          # 将图像从 BGR 转换为 RGB
plt.title('contours marked on RGB image')    # 指定输出图像的标题
plt.show()                                   # 显示图像
```



实验小结

在本实验中，您将通过执行轮廓检测，实现了从水果图片中的提取水果轮廓并绘制。

我们将借助本实验的步骤，在后面的实验中继续执行轮廓匹配的操作。