绘制图形

电子信息工程系

袁羽

目录

CONTENTS

绘制图形

OpenCV提供了许多绘制图形的方法，包括绘制线段的line()方法、绘制矩形的rectangle()方法、绘制圆形的circle()方法、绘制椭圆的ellipse()方法，绘制多边形的polylines()方法和绘制文字的putText()方法。

一 线段的绘制

OpenCV提供了用于绘制线段的line()方法，使用这个方法即可绘制长短不一、粗细各异、五颜六色的线段。line()方法的语法格式如下：

```
img = cv2.line(img, pt1, pt2, color, thickness)
```

参数说明：

img：画布。

pt1：线段的起点坐标。

pt2：线段的终点坐标。

color：绘制线段时的线条颜色。

thickness：绘制线段时的线条宽度。

一 线段的绘制

注意

线条颜色是RGB格式的，例如红色的RGB值是(255, 0, 0)。但是在OpenCV中，RGB图像的通道顺序被转换成B→G→R，因此(0, 0, 255)代表的是红色。

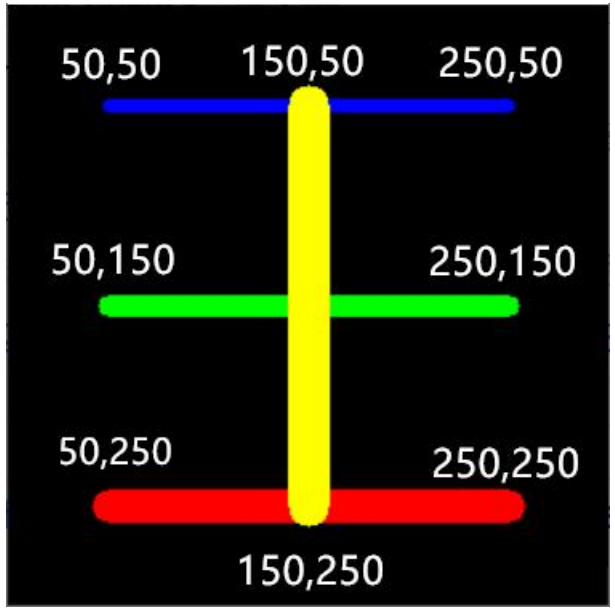
一 线段的绘制

注意

线条颜色是RGB格式的，例如红色的RGB值是(255, 0, 0)。但是在OpenCV中，RGB图像的通道顺序被转换成B→G→R，因此(0, 0, 255)代表的是红色。

例　绘制线段并拼成一个“王”字。

编写一个程序，使用line()方法分别绘制颜色为蓝色、绿色、红色和黄色，线条宽度为5、10、15和20的4条线段，并且这4条线段能够拼成一个“王”字，把其主体部分放在坐标系中，即可确定每条线段的起点坐标和终点坐标。

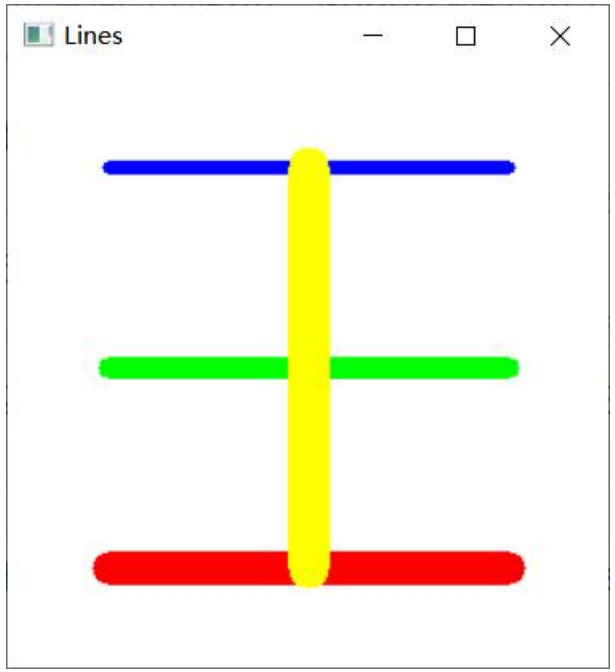


例 绘制线段并拼成一个“王”字。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV 中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 在画布上，绘制一条起点坐标为(50, 50)、终点坐标为(250, 50)、蓝色的、线条宽度为5的线段
canvas = cv2.line(canvas, (50, 50), (250, 50), (255, 0, 0), 5)
# 在画布上，绘制一条起点坐标为(50, 150)、终点坐标为(250, 150)、绿色的、线条宽度为10的线段
canvas = cv2.line(canvas, (50, 150), (250, 150), (0, 255, 0), 10)
# 在画布上，绘制一条起点坐标为(50, 250)、终点坐标为(250, 250)、红色的、线条宽度为15的线段
canvas = cv2.line(canvas, (50, 250), (250, 250), (0, 0, 255), 15)
# 在画布上，绘制一条起点坐标为(150, 50)、终点坐标为(150, 250)、黄色的、线条宽度为20的线段
canvas = cv2.line(canvas, (150, 50), (150, 250), (0, 255, 255), 20)
cv2.imshow("Lines", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

例　绘制线段并拼成一个“王”字。

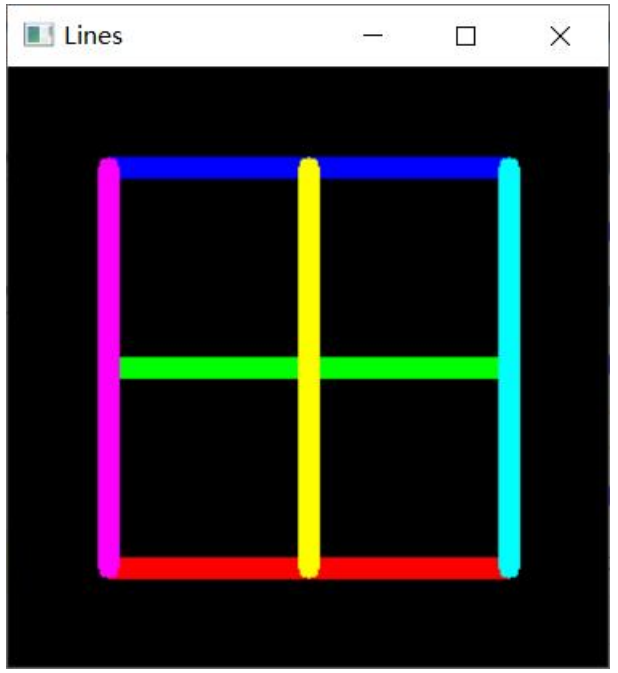
如果想把图中的黑色背景替换为白色背景，应该如何操作呢？



只需将第7行代码替换成如下代码即可：

```
canvas = np.ones((300, 300, 3), np.uint8) * 255
```

练习　绘制线段并拼成一个“田”字。



练习 绘制线段并拼成一个“田”字。

```
import numpy as np  # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV 中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 在画布上，绘制一条起点坐标为(50, 50)、终点坐标为(250, 50)、蓝色的、线条宽度为5的线段
canvas = cv2.line(canvas, (50, 50), (250, 50), (255, 0, 0), 10)
# 在画布上，绘制一条起点坐标为(50, 150)、终点坐标为(250, 150)、绿色的、线条宽度为10的线段
canvas = cv2.line(canvas, (50, 150), (250, 150), (0, 255, 0), 10)
```

练习 绘制线段并拼成一个“田”字。

```
# 在画布上，绘制一条起点坐标为(50, 250)、终点坐标为(250, 250)、红色的、线条宽度为15的线段
canvas = cv2.line(canvas, (50, 250), (250, 250), (0, 0, 255), 10)
# 在画布上，绘制一条起点坐标为(150, 50)、终点坐标为(150, 250)、黄色的、线条宽度为20的线段
canvas = cv2.line(canvas, (150, 50), (150, 250), (0, 255, 255), 10)
canvas = cv2.line(canvas, (50, 50), (50, 250), (255, 0, 255), 10)
canvas = cv2.line(canvas, (250, 50), (250, 250), (255, 255, 0), 10)
cv2.imshow("Lines", canvas)  # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

二 矩形的绘制

OpenCV提供了用于绘制矩形的rectangle()方法，使用这个方法既可以绘制矩形边框，也可以绘制实心矩形。

rectangle()方法的语法格式如下：

```
img = cv2.rectangle(img, pt1, pt2, color, thickness)
```

参数说明：

img：画布。

pt1：矩形的左上角坐标。

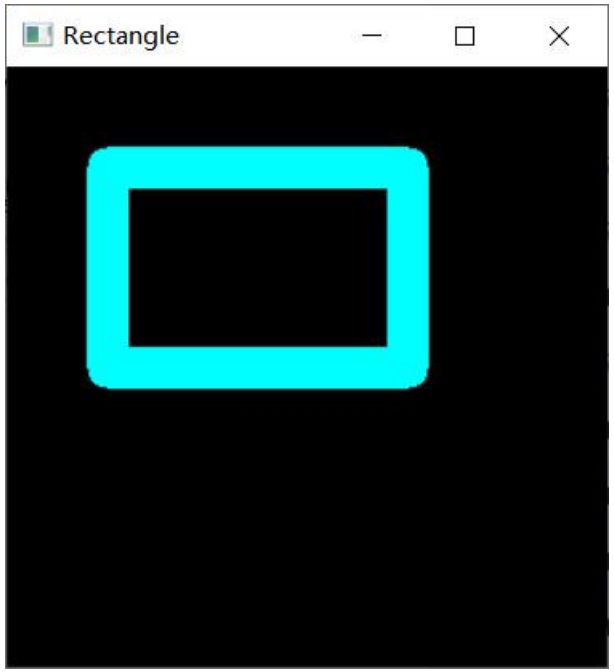
pt2：矩形的右下角坐标。

color：绘制矩形时的线条颜色。

thickness：绘制矩形时的线条宽度，当thickness的值为-1时，即可绘制一个实心矩形。

例 绘制一个矩形边框。

编写一个程序，使用rectangle()方法绘制一个青色的、线条宽度为20的矩形边框。绘制矩形时，矩形的左上角坐标为(50, 50)，矩形的右下角坐标为(200, 150)。

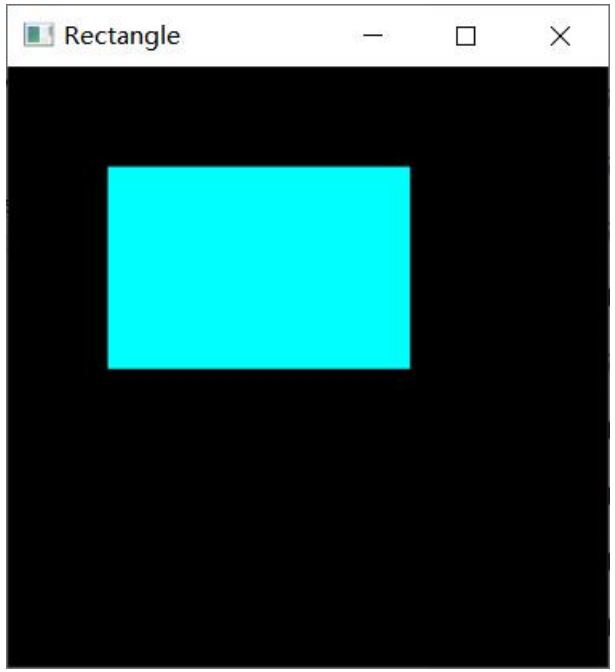


例 绘制一个矩形边框。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 在画布上绘制一个左上角坐标为(50,50)、右下角坐标为(200,150)、青色的、线条宽度为20的矩形边框
canvas = cv2.rectangle(canvas, (50, 50), (200, 150), (255, 255, 0), 20)
cv2.imshow("Rectangle", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

例 绘制一个矩形边框。

如果想要填充图中的矩形边框，使之变成实心矩形，应该如何修改上述代码呢？



当thickness的值为-1时，即可绘制一个实心矩形。修改代码如下：

```
canvas = cv2.rectangle(canvas, (50, 50), (200, 150), (255, 255, 0), -1) # 绘制一个实心矩形
```

例 绘制正方形。

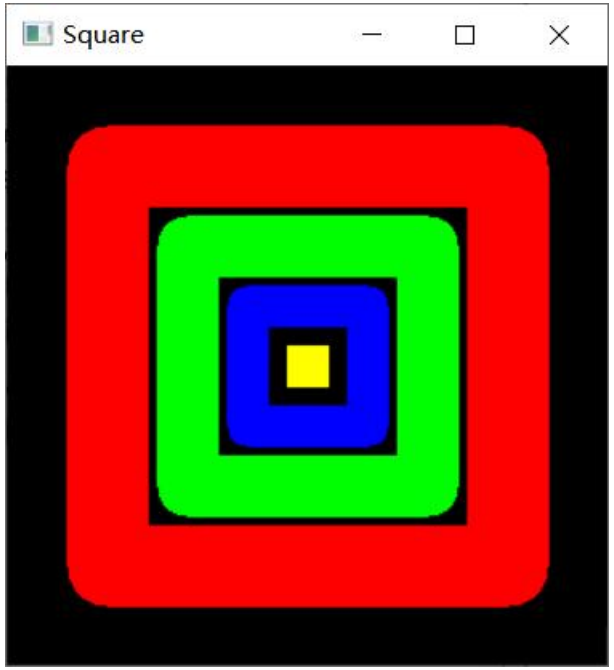
编写一个程序，使用rectangle()方法分别绘制3个正方形边框和1个实心正方形。具体要求如下。

（1）左上角坐标为(50, 50)、右下角坐标为(250, 250)、红色的、线条宽度为40的正方形边框。

（2）左上角坐标为(90, 90)、右下角坐标为(210, 210)、绿色的、线条宽度为30的正方形边框。

（3）左上角坐标为(120, 120)、右下角坐标为(180, 180)、蓝色的、线条宽度为20的正方形边框。

（4）左上角坐标为(140, 140)、右下角坐标为(160, 160)、黄色的实心正方形。



例 绘制正方形。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 绘制一个左上角坐标为(50,50)、右下角坐标为(250,250)、红色的、线条宽度为40的正方形边框
canvas = cv2.rectangle(canvas, (50, 50), (250, 250), (0, 0, 255), 40)
# 绘制一个左上角坐标为(90,90)、右下角坐标为(210,210)、绿色的、线条宽度为30的正方形边框
canvas = cv2.rectangle(canvas, (90, 90), (210, 210), (0, 255, 0), 30)
# 绘制一个左上角坐标为(120,120)、右下角坐标为(180,180)、蓝色的、线条宽度为20的正方形边框
canvas = cv2.rectangle(canvas, (120, 120), (180, 180), (255, 0, 0), 20)
# 绘制一个左上角坐标为(140,140)、右下角坐标为(160,160)、黄色的实心正方形
canvas = cv2.rectangle(canvas, (140, 140), (160, 160), (0, 255, 255), -1)
cv2.imshow("Square", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

三　圆形的绘制

OpenCV提供了用于绘制圆形的circle()方法，这个方法与rectangle()方法的功能相同，既可以绘制圆形边框，也可以绘制实心圆形。circle()方法的语法格式如下：

```
img = cv2.circle(img, center, radius, color, thickness)
```

参数说明：

img：画布。

center：圆形的圆心坐标。

radius：圆形的半径。

color：绘制圆形时的线条颜色。

thickness：绘制圆形时的线条宽度。

例 绘制“交通灯”。

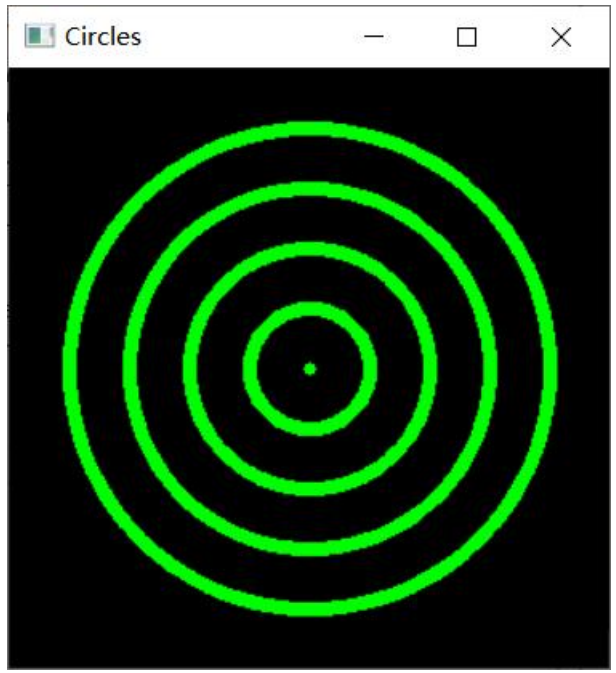
编写一个程序，使用circle()方法分别绘制红色的、黄色的和绿色的3个实心圆形，用于模拟交通灯。这3个实心圆形的半径均为40，并且呈水平方向放置。

例 绘制“交通灯”。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (100, 300, 3)：一个100 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((100, 300, 3), np.uint8)
#在画布上，绘制一个圆心坐标为(50, 50)、半径为40、红色的实心圆形
canvas = cv2.circle(canvas, (50, 50), 40, (0, 0, 255), -1)
# 在画布上，绘制一个圆心坐标为(150, 50)、半径为40、黄色的实心圆形
canvas = cv2.circle(canvas, (150, 50), 40, (0, 255, 255), -1)
# 在画布上，绘制一个圆心坐标为(250, 50)、半径为40、绿色的实心圆形
canvas = cv2.circle(canvas, (250, 50), 40, (0, 255, 0), -1)
cv2.imshow("TrafficLights", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

例 绘制同心圆。

编写一个程序，使用circle()方法和for循环绘制5个同心圆，这些圆形的圆心坐标均为画布的中心，半径的值分别为0，30，60，90和120，线条颜色均为绿色，线条宽度均为5。



练习 绘制同心圆。

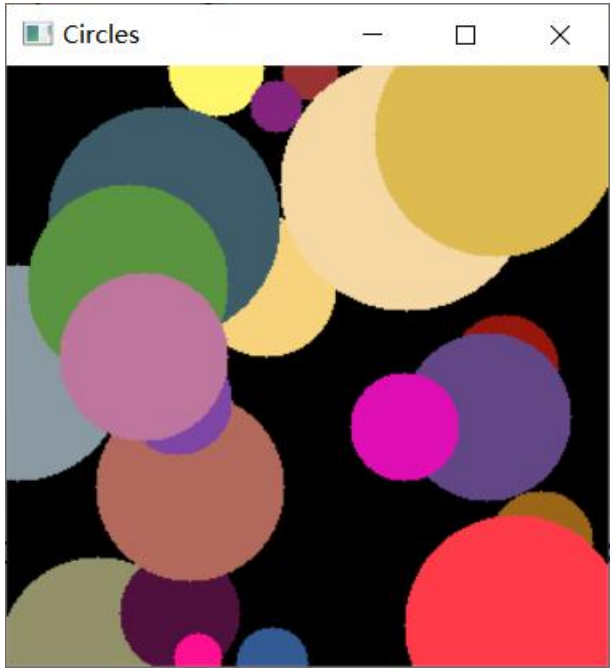
```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
#shape[1]表示画布的宽度，center_X表示圆心的横坐标
#圆心的横坐标等于画布的宽度的一半
center_X = int(canvas.shape[1] / 2)
#shape[0]表示画布的高度，center_X表示圆心的纵坐标
#圆心的纵坐标等于画布的高度的一半
center_Y = int(canvas.shape[0] / 2)
```

练习 绘制同心圆。

```
#r表示半径；其中，r的值分别为0，30，60，90和120
for r in range(0, 150, 30):
#绘制一个圆心坐标为(center_X, center_Y)、半径为r、绿色的、线条宽度为5的圆形
        cv2.circle(canvas, (center_X, center_Y), r, (0, 255, 0), 5)
cv2.imshow("Circles", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

例 绘制27个随机实心圆。

编写一个程序，使用circle()方法和for循环随机绘制27个实心圆。其中，圆心的横、纵坐标在[0, 299]内取值，半径在[11, 70]内取值，线条颜色由3个在[0, 255]内的随机数组成的列表表示。



例 绘制27个随机实心圆。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 通过循环绘制27个实心圆
for numbers in range(0, 28):
    # 获得随机的圆心横坐标，这个横坐标在[0, 299]范围内取值
    center_X = np.random.randint(0,300)
    # 获得随机的圆心纵坐标，这个纵坐标在[0, 299]范围内取值
    center_Y = np.random.randint(0, 300)
```

例 绘制27个随机实心圆。

```
    # 获得随机的半径，这个半径在[11, 70]范围内取值
    radius = np.random.randint(11, 71)
    # 获得随机的线条颜色，这个颜色由3个在[0, 255]范围内的随机数组成的列表表示
    color = np.random.randint(0,256,3).tolist()
    # 绘制一个圆心坐标为(center_X, center_Y)、半径为radius、颜色为color的实心圆形
    cv2.circle(canvas, (center_X, center_Y), radius, color, -1)
cv2.imshow("Circles", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

因为OpenCV中的颜色值是一个列表（例如(0, 0, 255)等），所以color=np.random.randint(0, 256, 3).tolist()中的.tolist()不能忽略，先将数组转换为列表，否则运行程序时会发生错误。

四 椭圆的绘制

在OpenCV中，绘制椭圆比较复杂，要多输入几个参数，如中心点的位置坐标，长轴和短轴的长度，椭圆沿逆时针方向旋转的角度等。cv2.ellipse()函数原型如下所示：

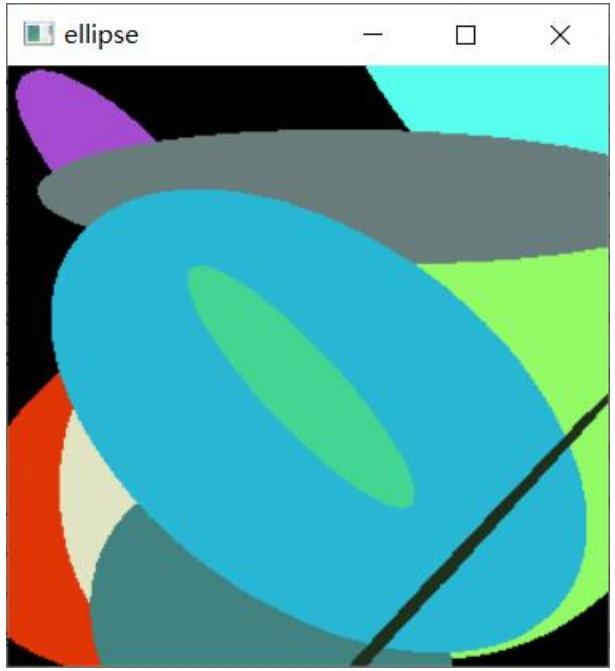
img = cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType[, shift]]])

参数说明：

- img表示需要绘制椭圆的图像
- center表示椭圆圆心坐标
- axes表示轴的长度（短半径和长半径）
- angle表示偏转的角度
- startAngle表示圆弧起始角的角度
- endAngle表示圆弧终结角的角度
- color表示线条的颜色
- thickness如果为正值，表示椭圆轮廓的厚度；负值表示要绘制一个填充椭圆
- lineType表示圆的边界类型
- shift表示中心坐标和轴值中的小数位数

例 绘制10个随机椭圆。

编写一个程序，使用ellipse()方法和for循环随机绘制10个实心椭圆。其中，椭圆圆心的横、纵坐标在[0, 299]内取值，长短半轴随机取值，线条颜色由3个在[0, 255]内的随机数组成的列表表示。



例 绘制10个随机椭圆。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 通过循环绘制10个实心椭圆
for numbers in range(0, 11):
    # 获得随机的椭圆圆心横坐标，这个横坐标在[0, 299]范围内取值
    center_X = np.random.randint(0,300)
    # 获得随机的椭圆圆心纵坐标，这个纵坐标在[0, 299]范围内取值
    center_Y = np.random.randint(0, 300)
```

例 绘制10个随机椭圆。

```
# 获取随机长短轴
    axes_X = np.random.randint(100, 200)
    axes_Y = np.random.randint(0, 100)
    # 获取随机偏转角度
    angle = np.random.randint(0, 360)
    # 获得随机的线条颜色，这个颜色由3个在[0, 255]范围内的随机数组成的列表表示
    color = np.random.randint(0,256,3).tolist()
    # 绘制一个椭圆圆心坐标为(center_X, center_Y)、长短轴为（axes_X，axes_Y）、随机偏转，颜色为color的实心椭圆形
    cv2.ellipse(canvas, (center_X, center_Y),(axes_X,axes_Y) ,angle,0, 360, color, -1)
cv2.imshow("ellipse", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

五 多边形的绘制

OpenCV提供了绘制多边形的polylines()方法，使用这个方法绘制的多边形既可以是闭合的，也可以是不闭合的。polylines()方法的语法格式如下：

```
img = cv2.polylines(img, pts, isClosed, color, thickness)
```

参数说明：

img：画布。

pts：由多边形各个顶点的坐标组成的一个列表，这个列表是一个numpy的数组类型。

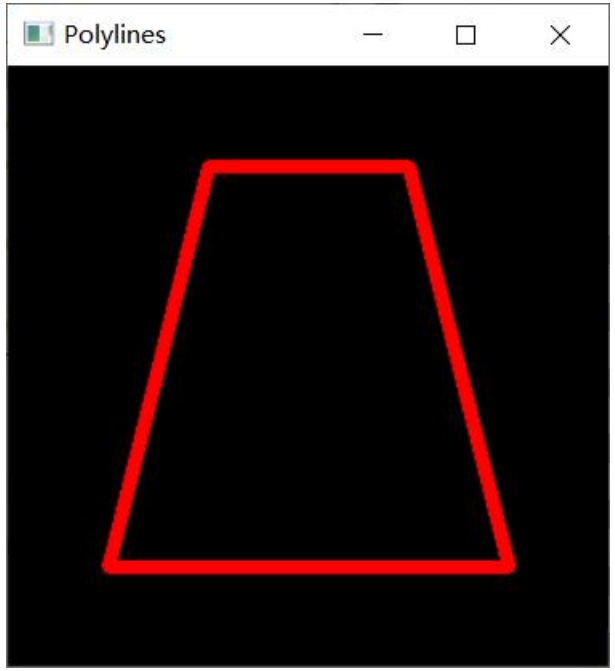
isClosed：如果值为True，表示一个闭合的多边形；如果值为False，表示一个不闭合的多边形。

color：绘制多边形时的线条颜色。

thickness：绘制多边形时的线条宽度。

例 绘制一个等腰梯形边框。

编写一个程序，按顺时针给出等腰梯形4个顶点的坐标，即(100, 50)，(200, 50)，(250, 250)和(50, 250)。在画布上根据4个顶点的坐标，绘制一个闭合的、红色的、线条宽度为5的等腰梯形边框。



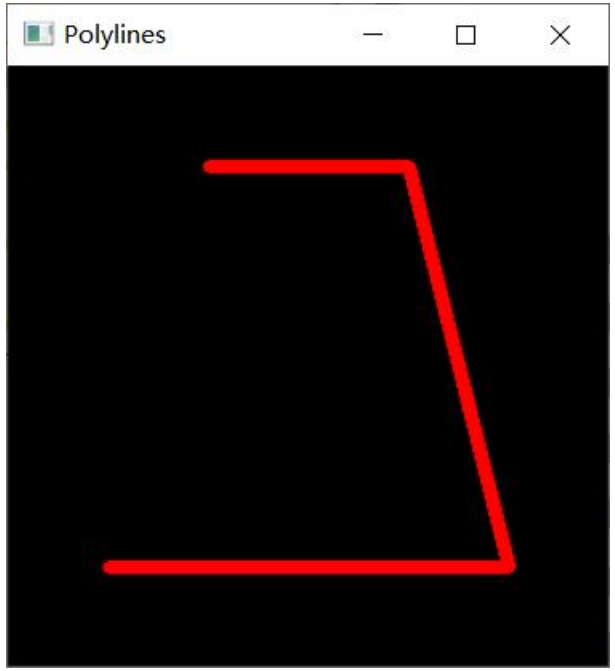
例 绘制一个等腰梯形边框。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (300, 300, 3)：一个300 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uuint8存储的，因此这里的类型也是uint8
canvas = np.zeros((300, 300, 3), np.uint8)
# 按顺时针给出等腰梯形4个顶点的坐标
# 这4个顶点的坐标构成了一个大小等于“顶点个数 * 1 * 2”的数组
# 这个数组的数据类型为np.int32
pts = np.array([[100, 50], [200, 50], [250, 250], [50, 250]], np.int32)
# 在画布上根据4个顶点的坐标，绘制一个闭合的、红色的、线条宽度为5的等腰梯形边框
canvas = cv2.polylines(canvas, [pts], True, (0, 0, 255), 5)
cv2.imshow("Polylines", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

例 绘制一个等腰梯形边框。

把第13行代码中的True修改为False，那么将绘制出一个不闭合的等腰梯形边框，代码如下：

```
canvas = cv2.polylines(canvas, [pts], False, (0, 0, 255), 5) # 绘制一个不闭合的等腰梯形边框
```



六 文字的绘制

OpenCV提供了用于绘制文字的putText()方法，使用这个方法不仅能够设置字体的样式、大小和颜色，而且能够使字体呈现斜体的效果，还能够控制文字的方向，进而使文字呈现垂直镜像的效果。

putText()方法的语法格式如下：

```
img = cv2.putText(img, text, org, fontFace, fontScale, color, thickness, lineType, bottomLeftOrigin)
```

参数说明：

img：画布。

text：要绘制的文字内容。

org：文字在画布中左下的坐标。

fontFace：字体样式

六 文字的绘制

字体样式	含义
CV_FONT_HERSHEY_SIMPLEX	正常大小无衬线字体
CV_FONT_HERSHEY_PLAIN	小号无衬线字体
CV_FONT_HERSHEY_DUPLEX	正常大小无衬线字体比CV_FONT_HERSHEY_SIMPLEX 更复杂)
CV_FONT_HERSHEY_COMPLEX	正常大小有衬线字体
CV_FONT_HERSHEY_TRIPLEX	正常大小有衬线字体 （ 比CV_FONT_HERSHEY_COMPLEX更复杂)
CV_FONT_HERSHEY_COMPLEX_SMALL	CV_FONT_HERSHEY_COMPLEX 的小译本
CV_FONT_HERSHEY_SCRIPT_SIMPLEX	手写风格字体
CV_FONT_HERSHEY_SCRIPT_COMPLEX	比CV_FONT_HERSHEY_SCRIPT_SIMPLEX更复杂
CV_FONT_ITALIC	斜体字

六 文字的绘制

fontScale：字体大小。

color：绘制文字时的线条颜色。

thickness：绘制文字时的线条宽度。

lineType：线型。（线型指的是线的产生算法，有4和8两个值，默认值为8）。

bottomLeftOrigin：标识原点位置，控制绘制文字时的方向。（有True和False两个值，默认值为False），若为true，则表示图像左下角为原点，OpenCV中原点位置一般为左上角。

说明

使用putText()方法时，thickness、lineType和bottomLeftOrigin是可选参数，有无均可。

例 绘制文字“opencv”。

编写一个程序，在画布上绘制文字“opencv”。其中，文字左下角的坐标为(20, 70)，字体样式为FONT_HERSHEY_TRIPLEX，字体大小为2，线条颜色是绿色，线条宽度为5。

例 绘制文字“opencv”。

```
import numpy as np # 导入Python中的numpy模块
import cv2
# np.zeros()：创建了一个画布
# (100, 300, 3)：一个100 x 300，具有3个颜色空间（即Red、Green和Blue）的画布
# np.uint8：OpenCV中的灰度图像和RGB图像都是以uint8存储的，因此这里的类型也是uint8
canvas = np.zeros((100, 300, 3), np.uint8)
# 在画布上绘制文字“opencv”，文字左下的坐标为(20, 70)
#字体样式为FONT_HERSHEY_TRIPLEX
#字体大小为2，线条颜色是绿色，线条宽度为5
cv2.putText(canvas, "opencv", (20, 70), cv2.FONT_HERSHEY_TRIPLEX, 2, (0, 255, 0), 5)
cv2.imshow("Text", canvas) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

说明

不借助其他库或者模块，使用putText()方法绘制中文时，即把实例6.8的第11行代码中的opencv修改为“您好”，代码如下：

```
cv2.putText(canvas, "您好", (20, 70), cv2.FONT_HERSHEY_TRIPLEX, 2, (0, 255, 0), 5)
```

运行上述代码会出现乱码。

1、文字的斜体效果

FONT_ITALIC可以与其他文字类型一起使用，使字体在呈现指定字体样式效果的同时，也呈现斜体效果。

例：

```
fontStyle = cv2.FONT_HERSHEY_TRIPLEX + cv2.FONT_ITALIC
cv2.putText(canvas, "opencv", (20, 70), fontStyle, 2, (0, 255, 0), 5)
```

2、文字的垂直镜像效果

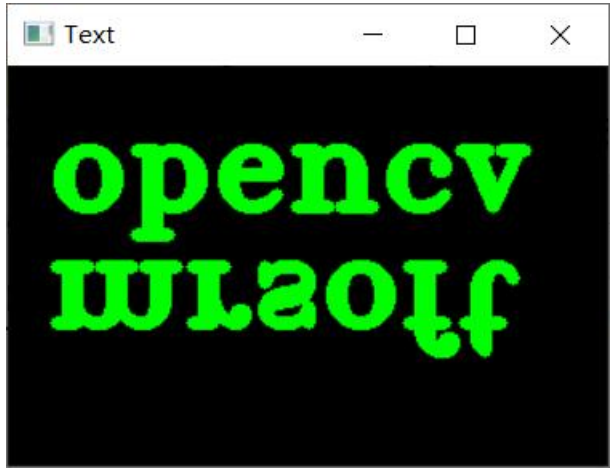
在putText()方法的语法格式中：

cv2.putText(img, text, org, fontFace, fontScale, color, thickness, lineType, bottomLeftOrigin)

有一个控制绘制文字时的方向的参数，即bottomLeftOrigin，其默认值为False。当bottomLeftOrigin为True时，文字将呈现垂直镜像效果。

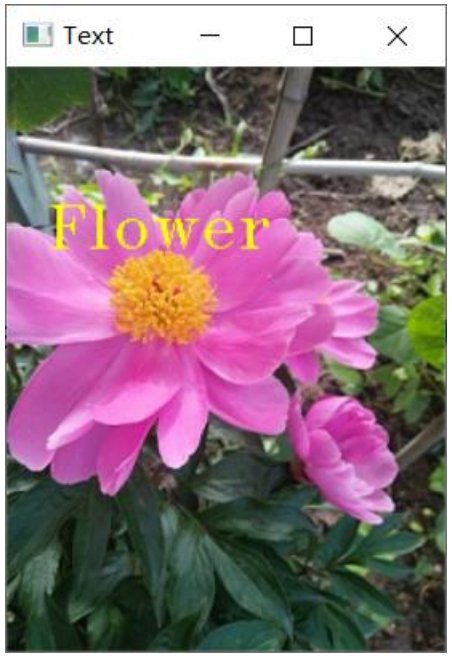
例：

```
img =  cv2.putText(canvas, "mrsoft", (20, 100), fontStyle, 2, (0, 255, 0), 5, 8, True)
```



3、在图像上绘制文字

OpenCV除了可以在np.zeros()创建的画布上绘制文字外，还能够在图像上绘制文字。区别是当在图像上绘制文字时，不再需要导入Python中的numpy模块。



3、在图像上绘制文字

编写一个程序，在flower图片上绘制文字“Flower”。其中，文字左下角的坐标为(20, 90)，字体样式为FONT_HERSHEY_TRIPLEX，字体大小为1，线条颜色是黄色。

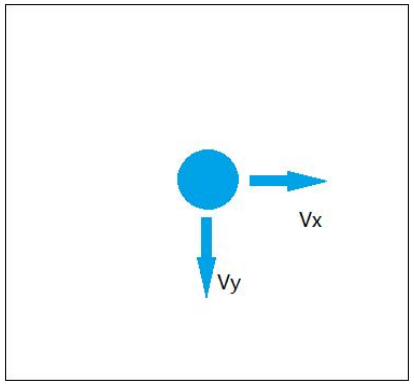
```
import cv2
image = cv2.imread(flower.jpg;) # 读取flower.jpg
#字体样式为FONT_HERSHEY_TRIPLEX
fontStyle = cv2.FONT_HERSHEY_TRIPLEX
# 在image上绘制文字“Flower”，文字左下角的坐标为(20, 90),
# 字体样式为fontStyle，字体大小为1，线条颜色是黄色
cv2.putText(image, “Flower”, (20, 90), fontStyle, 1, (0, 255, 255))
cv2.imshow(“image”, image) # 显示画布
cv2.waitKey()
cv2.destroyAllWindows()
```

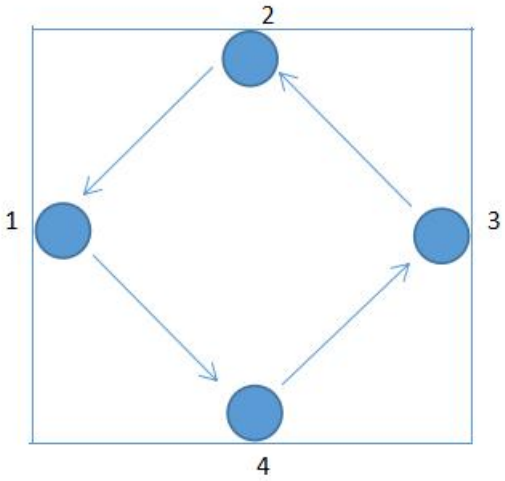
例 弹球动画

在一个宽、高都为200像素的纯白色图像中，绘制一个半径为20像素的纯蓝色小球。让小球做匀速直线运动，一旦圆形碰触到图像边界则开始反弹（反弹不损失动能）。

例 弹球动画

在一个宽、高都为200像素的纯白色图像中，绘制一个半径为20像素的纯蓝色小球。让小球做匀速直线运动，一旦圆形碰触到图像边界则开始反弹（反弹不损失动能）。





例 弹球动画

```
import cv2
import numpy as np
width, height = 200, 200  # 画面的宽和高
r = 20  # 圆半径
x =  20  # 圆心和坐标起始坐标
y = 100  # 圆形纵坐标起始坐标
x_offer = y_offer = 4  # 每一帧的移动速度
while cv2.waitKey(30) == -1:  # 按下任何按键之后
    if x > width - r or x < r:  # 如果圆的横坐标超出边界
        x_offer *= -1  # 横坐标速度取相反值
    if y > height - r or y < r:  # 如果圆的纵坐标超出边界
        y_offer *= -1  # 纵坐标速度取相反值
    x += x_offer  # 圆心按照横坐标速度移动
    y += y_offer  # 圆心按照纵坐标速度移动
    img = np.ones((width, height, 3), np.uint8) * 255  # 绘制白色背景面板
    cv2.circle(img, (x, y), r, (255, 0, 0), -1)  # 绘制圆形
    cv2.imshow("img", img)  # 显示图像
cv2.destroyAllWindows()  # 释放所有窗体
```

小结

1.绘制线段的line()方法

2.绘制矩形的rectangle()方法

3.绘制圆形的circle()方法

4.绘制椭圆的ellipse()方法

5.绘制多边形的polylines()方法

6.绘制文字的putText()方法。

THANKS