创建图像

电子信息工程系

袁羽

目录

CONTENTS

一 创建图像

在OpenCV中，黑白图像实际上就是一个二维数组，彩色图像是一个三维数组。数组中每个元素就是图像对应位置的像素值。因此修改图像像素的操作实际上就是修改数组的操作。

一 创建图像

1．创建黑白图像

在黑白图像中，像素值为0表示纯黑，像素值为255表示纯白。

创建纯白图像有两种方式：第一种是先纯黑图像，然后将图像中所有的像素值改为255；第二种使用NumPy提供的ones()方法创建一个像素值均为1的数组，然后让数组乘以255。

例 创建纯黑图像：创建一个100行、200列（即宽200、高100）的数组，数组元素格式为无符号8位整数，用0填充整个数组，将该数组当作图像显示出来。

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
# 创建指定宽高、单通道、像素值都为0的图像
img = np.zeros((height, width), np.uint8)
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

运行结果：

例 创建纯白图像。

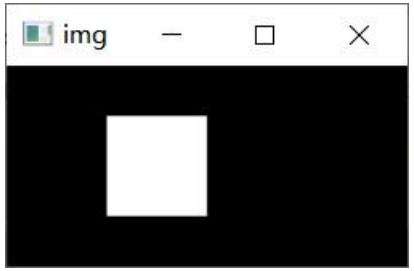
```
# 创建指定宽高、单通道、像素值都为0的图像
  img = np.zeros((height, width), np.uint8)
# 方法一：利用切片索引修改像素
  img[:,:] = 255
# 方法二：利用循环修改像素
 for i in range(height):
     for j in range(width):
         img[i,j] = 255
# 方法三：创建指定宽高、单通道、像素值都为1的图像
img = np.ones((height, width), np.uint8)*255
```

运行结果：

练 在黑图像内部绘制白色矩形：先绘制纯黑图像作为背景，然后使用切片式索引操作将图像中横坐标为50~100、纵坐标为25~75的矩形区域颜色改为纯白色。

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
img = np.zeros((height, width), np.uint8)
# 图像纵坐标25~75、横坐标50~100之间的区域变为白色
img[25:75, 50:100] = 255
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
```

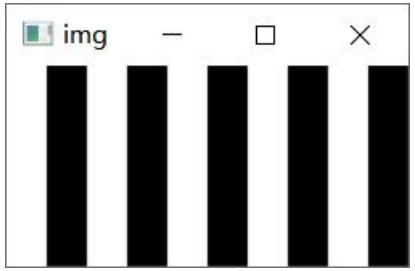
运行结果：



练 创建黑白相间的图像：先绘制纯黑图像作为背景，然后在循环中使用切片式索引操作绘制黑白间隔图像。

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
img = np.zeros((height, width), np.uint8)
for i in range(0, width, 40):
    img[:, i:i + 20] = 255  # 白色区域的宽度为20像素
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

运行结果：



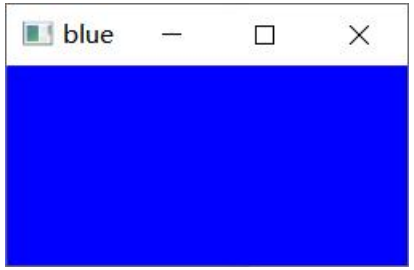
一 创建图像

2．创建彩色图像

OpenCV中彩色图像默认为BGR格式，彩色图像的第三个索引表示的就是蓝、绿、红3种颜色的分量。

例 创建彩色图像：分别创建纯蓝、纯绿和纯红图像。

运行结果：



例 创建彩色图像：分别创建宽为200，高为100的纯蓝、纯绿和纯红图像。

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
# 创建指定宽高、3通道、像素值都为0的图像
img = np.zeros((height, width, 3), np.uint8)
blue = img.copy()  # 复制图像
blue[:, :, 0] = 255  # 1通道所有像素都为255
green = img.copy()
green[:, :, 1] = 255  # 2通道所有像素都为255
red = img.copy()
red[:, :, 2] = 255  # 3通道所有像素都为255
cv2.imshow("blue", blue)  # 展示图像
cv2.imshow("green", green)
cv2.imshow("red", red)
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

练习 创建彩色条纹图像。

运行结果：

练习 创建彩色条纹图像。

```
import cv2
import numpy as np
width = 300  # 图像的宽
height = 200  # 图像的高
# 创建指定宽高、3通道、像素值都为0的图像
img = np.zeros((height, width, 3), np.uint8)
img[:,0:100,0] = 255
img[:,100:200,1] = 255
img[:,200:300,2] = 255
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

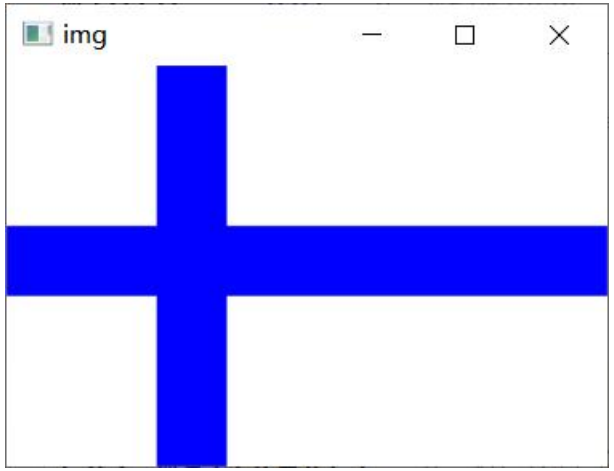
练习 创建彩色条纹图像。

练习 创建彩色条纹图像。

```
import cv2
import numpy as np
width = 300  # 图像的宽
height = 200  # 图像的高
# 创建指定宽高、3通道、像素值都为0的图像
img = np.zeros((height, width, 3), np.uint8)
img[:,0:100,0:2] = 255
img[:,100:200,1:] = 255
img[:,200:300,0::2] = 255
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

练习 创建彩色条纹图像。

```
import cv2
import numpy as np
width = 300  # 图像的宽
height = 200  # 图像的高
# 创建指定宽高、3通道、像素值都为0的图像
img = np.ones((height, width, 3), np.uint8)*255
img[:,75:125,1:] = 0
img[50:100,:,1:] = 0
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

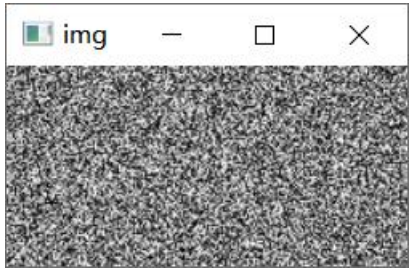


一 创建图像

3．创建随机图像

随机图像是指图像中每一个像素值都是随机生成的，因为像素之间不会组成有效的视觉信息，所以这样的图像看上去就像杂乱无章的沙子。虽然随机图像没有任何视觉信息，但对于图像处理技术仍然很重要，毫无规律的像素数组被称为干扰图像的噪声，可以当作图像加密的密钥。

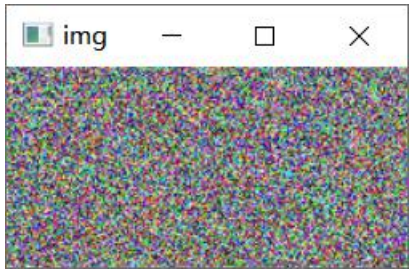
例 创建随机像素的雪花点图像



例 创建随机像素的雪花点图像

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
# 创建指定宽高、单通道、随机像素值的图像，随机值在0~256之间，数字为无符号8位格式
img = np.random.randint(256, size=(height, width), dtype=np.uint8)
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例 创建随机像素彩色图像



例 创建随机像素的雪花点图像

```
import cv2
import numpy as np
width = 200  # 图像的宽
height = 100  # 图像的高
# 创建指定宽高、单通道、随机像素值的图像，随机值在0~256之间，数字为无符号8位格式
 img = np.random.randint(256, size=(height, width, 3), dtype=np.uint8)
cv2.imshow("img", img)  # 展示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

二 拼接图像

NumPy提供了两种拼接数组的方法，分别是hstack()方法和vstack()方法。这两种拼接方法同样可用于拼接图像

二 拼接图像

1．水平拼接数组

hstack()方法可以对数组进行水平拼接（或叫横向拼接），其语法如下：

```
array = numpy.hstack(tup)
```

参数说明：tup：要拼接的数组元组。

返回值说明：array：将参数元组中的数组水平拼接后生成的新数组。

hstack()方法可以拼接多个数组，被拼接的数组必须在每一个维度都具有相同的长度，也就是数组“形状相同”。

例 创建3个一维数组，将这3个数组进行水平拼接

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = np.array([7, 8, 9])
result = np.hstack((a, b, c))
print(result)
```

运行结果如下：

```
[1 2 3 4 5 6 7 8 9]
```

二 拼接图像

2．垂直拼接数组

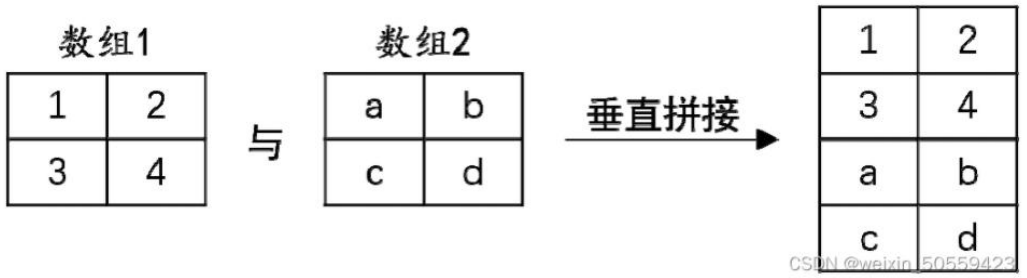
vstack()方法可以对数组进行垂直拼接（或叫纵向拼接），其语法如下：

array = numpy.vstack(tup)

参数说明：tup：要拼接的数组元组。

返回值说明： array：将参数元组中的数组垂直拼接后生成的新数组。

vstack()方法可以拼接多个数组，被拼接的数组必须在每一个维度都具有相同的长度，也就是数组“形状相同”。



例 创建3个一维数组，将这3个数组进行垂直拼接

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = np.array([7, 8, 9])
result = np.vstack((a, b, c))
print(result)
```

运行结果如下：

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

二 拼接图像

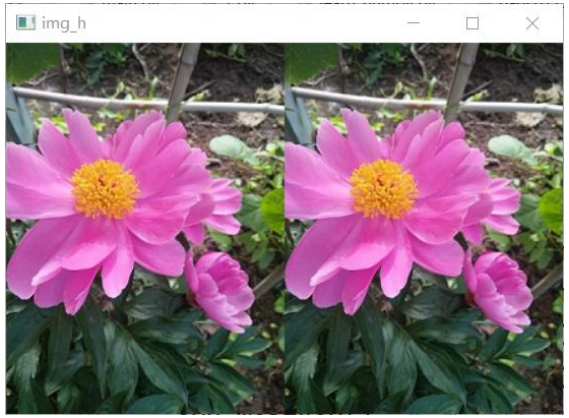
3．在图像处理中的应用

在OpenCV中，图像就是一个二维或三维的像素数组，这些数组同样可以被NumPy拼接。

例 读取一幅图像，让该图像拼接自身图像，分别用水平和垂直2种方式拼接。

```
import cv2
import numpy as np
img = cv2.imread("flower.jpg")  # 读取原始图像
img_h = np.hstack((img, img))  # 水平拼接两个图像
img_v = np.vstack((img, img))  # 垂直拼接两个图像
cv2.imshow("img_h", img_h)  # 展示拼接之后的效果
cv2.imshow("img_v", img_v)
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例 读取一幅图像，让该图像拼接自身图像，分别用水平和垂直2种方式拼接。



二 拼接图像

3．在图像处理中的应用

resize()方法可以随意更改图像的大小比例，其语法如下： dst = cv2.resize(src, dsize, fx, fy, interpolation)

二 拆分图像

1．分割数组

NumPy提供了hsplit、vsplit、dsplit和split函数，可以将数组分割成相同大小的子数组，也可以指定原数组中需要分割的位置。

使用hsplit函数可以对数组进行横向分割。

使用vsplit函数可以对数组进行纵向分割。

split函数同样可以实现数组分割。在参数axis=1时，可以进行横向分割；在参数axis=0时，可以进行纵向分割。

例 创建一个二维数组，将这个数组进行拆分

```
import numpy as np
n = np.array([[1, 2],
              [3, 4]])
n1,n2=np.hsplit(n, 2)  #hsplit函数横向分割
n3,n4=np.vsplit(n, 2)  #vsplit函数纵向分割
print(n1)
print(n2)
print(n3)
print(n4)
```

运行结果：

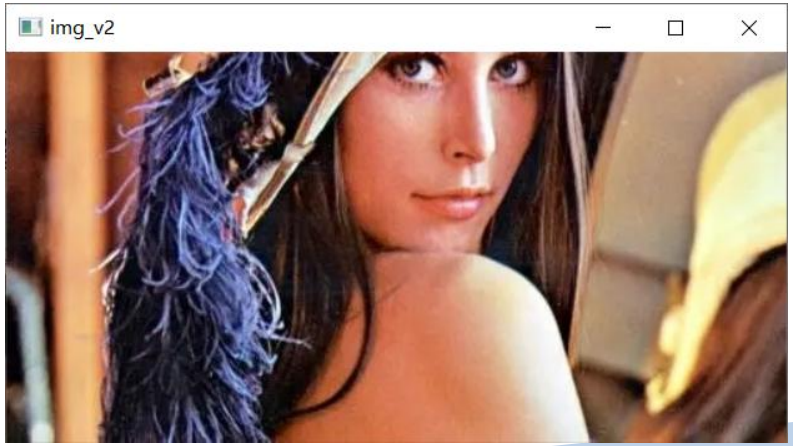
```
[[1]
 [3]]
[[2]
 [4]]
[[1 2]]
[[3 4]]
```

三 拆分图像

2．在图像处理中的应用

在OpenCV中，图像就是一个二维或三维的像素数组，这些数组同样可以被NumPy拆分。

例 读取一幅图像，让该图像分别用水平和垂直2种方式拆分。



例 读取一幅图像，让该图像分别用水平和垂直2种方式拆分。

```
import cv2
import numpy as np
img = cv2.imread("lenna.jpg")  # 读取原始图像
img_h1,img_h2 = np.hsplit(img, 2)  # 水平两个拆分图像
img_v1,img_v2 = np.vsplit(img, 2)  # 垂直拆分两个图像
cv2.imshow("img_h1", img_h1)  # 展示拆分之后的效果
cv2.imshow("img_h2", img_h2)  # 展示拆分之后的效果
cv2.imshow("img_v1", img_v1)
cv2.imshow("img_v2", img_v2)
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

三 拆分图像

2．在图像处理中的应用

resize()方法可以随意更改图像的大小比例，其语法如下： dst = cv2.resize(src, dsize, fx, fy, interpolation)

小结

1.创建图像

2.拼接图像：hstack()水平拼接

vstack()垂直拼接

THANKS