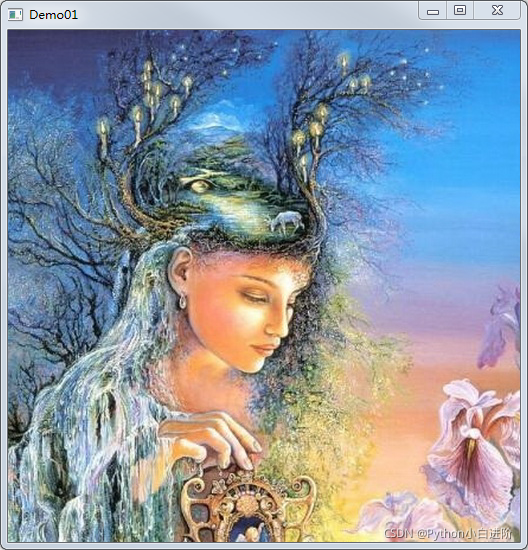
# 1. 图像的基本操作

## 01. 图像的读取（cv2.imread）

retval = cv.imread(filename[, flags])

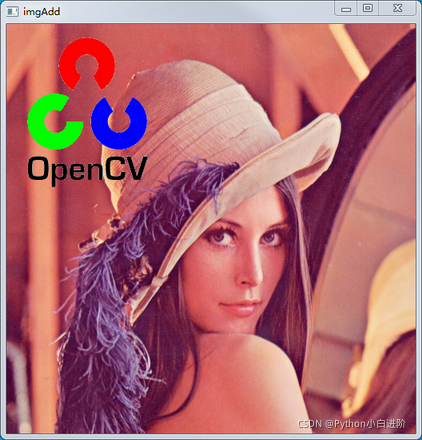
|  |
| --- |
| # 1.1 图像的读取  imgFile = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  img2 = cv2.imread(imgFile, flags=0) # flags=0 读取为灰度图像  # 1.2 从网络读取图像  import urllib.request as request  response = request.urlopen("https://profile.csdnimg.cn/8/E/F/0\_youcans")  imgUrl = cv2.imdecode(np.array(bytearray(response.read()), dtype=np.uint8), -1)  # 1.3 读取中文路径的图像  imgFile = "../images/测试图01.png" # 带有中文的文件路径和文件名  # imread() 不支持中文路径和文件名，读取失败，但不会报错!  # img = cv2.imread(imgFile, flags=1)  # 使用 imdecode 可以读取带有中文的文件路径和文件名  img = cv2.imdecode(np.fromfile(imgFile, dtype=np.uint8), -1) |



## 02. 图像的保存（cv2.imwrite）

retval = cv2.imwrite(filename, img [, paras])

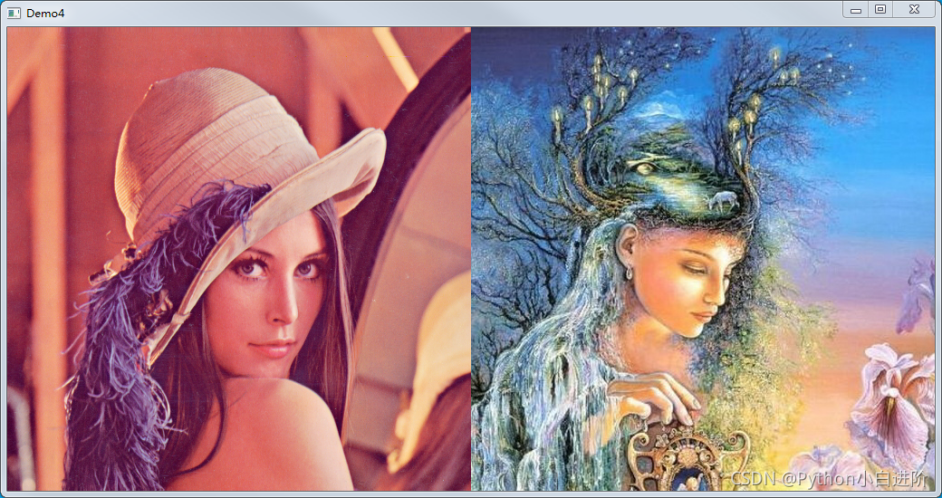
|  |
| --- |
| # 1.4 图像的保存  imgFile = "../images/logoCV.png" # 读取文件的路径  img3 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  saveFile = "../images/imgSave.png" # 保存文件的路径  # cv2.imwrite(saveFile, img3, [int(cv2.IMWRITE\_PNG\_COMPRESSION), 8]) # 保存图像文件, 设置压缩比为 8  cv2.imwrite(saveFile, img3) # 保存图像文件  # 1.5 保存中文路径的图像  imgFile = "../images/logoCV.png" # 读取文件的路径  img3 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  saveFile = "../images/测试图02.jpg" # 带有中文的保存文件路径  # cv2.imwrite(saveFile, img3) # imwrite 不支持中文路径和文件名，读取失败，但不会报错!  img\_write = cv2.imencode(".jpg", img3)[1].tofile(saveFile) |



## 03. 图像的显示（cv2.imshow）

None = imshow(winname, img)

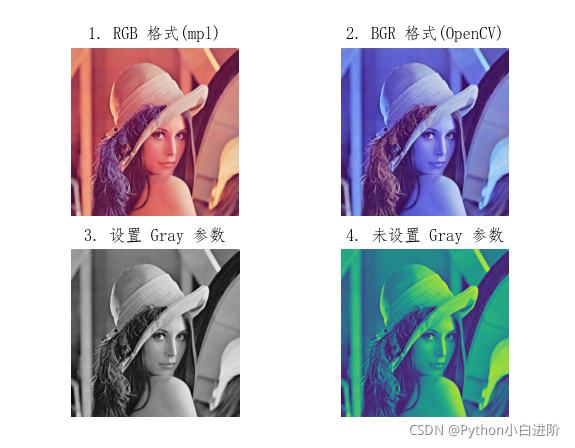
|  |
| --- |
| # 1.6 图像的显示(cv2.imshow)  imgFile = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  img2 = cv2.imread(imgFile, flags=0) # flags=0 读取为灰度图像  cv2.imshow("Demo1", img1) # 在窗口 "Demo1" 显示图像 img1  cv2.imshow("Demo2", img2) # 在窗口 "Demo2" 显示图像 img2  key = cv2.waitKey(0) # 等待按键命令, 1000ms 后自动关闭  # 1.7 图像显示(按指定大小的窗口显示图像)  imgFile = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  cv2.namedWindow("Demo3", cv2.WINDOW\_NORMAL)  cv2.resizeWindow("Demo3", 400, 300)  cv2.imshow("Demo3", img1) # 在窗口 "Demo3" 显示图像 img1  key = cv2.waitKey(0) # 等待按键命令, 1000ms 后自动关闭  # 1.8 图像显示(多个图像组合显示)  imgFile1 = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile1, flags=1) # flags=1 读取彩色图像(BGR)  imgFile2 = "../images/imgGaia.tif" # 读取文件的路径  img2 = cv2.imread(imgFile2, flags=1) # # flags=1 读取彩色图像(BGR)  imgStack = np.hstack((img1, img2)) # 相同大小图像水平拼接  cv2.imshow("Demo4", imgStack) # 在窗口 "Demo4" 显示图像 imgStack  key = cv2.waitKey(0) # 等待按键命令, 1000ms 后自动关闭 |



## 04. 用 matplotlib 显示图像（plt.imshow）

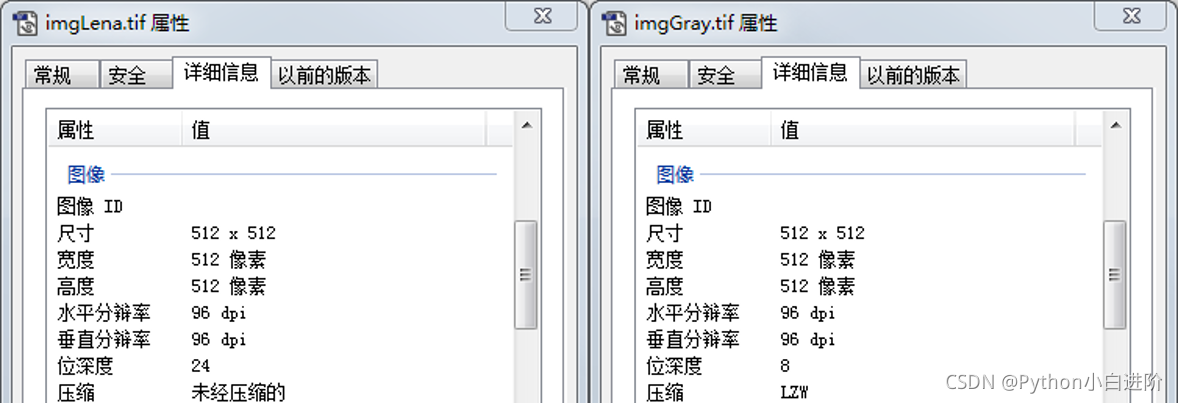
matplotlib.pyplot.imshow(img[, cmap])

|  |
| --- |
| # 1.10 图像显示(plt.imshow)  imgFile = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  imgRGB = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB) # 图片格式转换：BGR(OpenCV) -> RGB(PyQt5)  img2 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY) # 图片格式转换：BGR(OpenCV) -> Gray  plt.rcParams['font.sans-serif'] = ['FangSong'] # 支持中文标签  plt.subplot(221), plt.title("1. RGB 格式(mpl)"), plt.axis('off')  plt.imshow(imgRGB) # matplotlib 显示彩色图像(RGB格式)  plt.subplot(222), plt.title("2. BGR 格式(OpenCV)"), plt.axis('off')  plt.imshow(img1) # matplotlib 显示彩色图像(BGR格式)  plt.subplot(223), plt.title("3. 设置 Gray 参数"), plt.axis('off')  plt.imshow(img2, cmap='gray') # matplotlib 显示灰度图像，设置 Gray 参数  plt.subplot(224), plt.title("4. 未设置 Gray 参数"), plt.axis('off')  plt.imshow(img2) # matplotlib 显示灰度图像，未设置 Gray 参数  plt.show() |



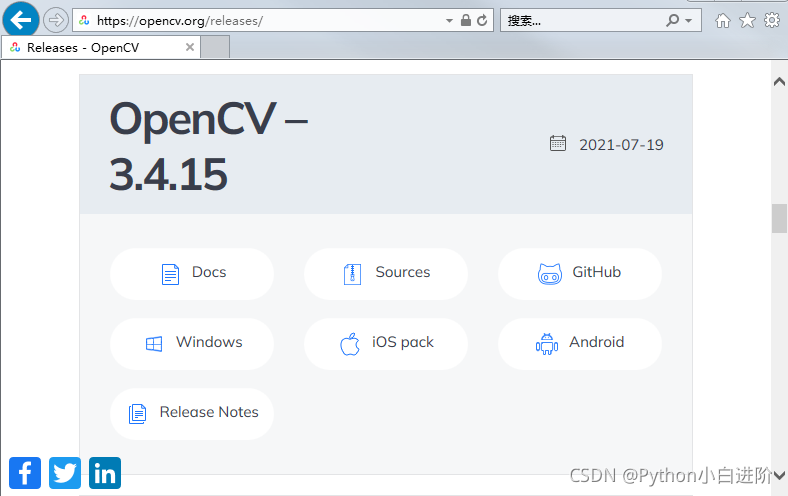
## 05. 图像的属性（np.shape）

|  |
| --- |
| # 1.11 图像数组的属性  imgFile = "../images/imgLena.tif" # 读取文件的路径  img1 = cv2.imread(imgFile, flags=1) # flags=1 读取彩色图像(BGR)  img2 = cv2.imread(imgFile, flags=0) # flags=0 读取为灰度图像  # cv2.imshow("Demo1", img1) # 在窗口显示图像  # key = cv2.waitKey(0) # 等待按键命令  # 维数(ndim), 形状(shape), 元素总数(size), 元素类型(dtype)  print("Ndim of img1(BGR): {}, img2(Gray): {}".format(img1.ndim, img2.ndim)) # number of rows, columns and channels  print("Shape of img1(BGR): {}, img2(Gray): {}".format(img1.shape, img2.shape)) # number of rows, columns and channels  print("Size of img1(BGR): {}, img2(Gray): {}".format(img1.size, img2.size)) # size = rows \* columns \* channels  print("Dtype of img1(BGR): {}, img2(Gray): {}".format(img1.dtype, img2.dtype)) # uint8  //本例程的运行结果如下：  // Ndim of img1(BGR): 3, img2(Gray): 2  // Shape of img1(BGR): (512, 512, 3), img2(Gray): (512, 512)  // Size of img1(BGR): 786432, img2(Gray): 262144  // Dtype of img1(BGR): uint8, img2(Gray): uint8 |



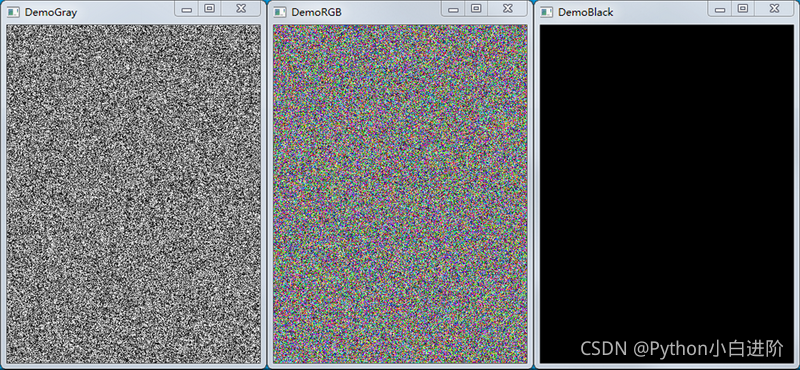
## 06. 像素的编辑（img.itemset）

|  |
| --- |
| # 1.13 Numpy 获取和修改像素值  img1 = cv2.imread("../images/imgLena.tif", flags=1) # flags=1 读取彩色图像(BGR)  x, y = 10, 10 # 指定像素位置 x, y  # (1) 直接访问数组元素，获取像素值(BGR)  pxBGR = img1[x,y] # 访问数组元素[x,y], 获取像素 [x,y] 的值  print("x={}, y={}\nimg[x,y] = {}".format(x,y,img1[x,y]))  # (2) 直接访问数组元素，获取像素通道的值  print("img[{},{},ch]:".format(x,y))  for i in range(3):  print(img1[x, y, i], end=' ') # i=0,1,2 对应 B,G,R 通道  # (3) img.item() 访问数组元素，获取像素通道的值  print("\nimg.item({},{},ch):".format(x,y))  for i in range(3):  print(img1.item(x, y, i), end=' ') # i=0,1,2 对应 B,G,R 通道  # (4) 修改像素值：img.itemset() 访问数组元素，修改像素通道的值  ch, newValue = 0, 255  print("\noriginal img[x,y] = {}".format(img1[x,y]))  img1.itemset((x, y, ch), newValue) # 将 [x,y,channel] 的值修改为 newValue  print("updated img[x,y] = {}".format(img1[x,y]))  // 本例程的运行结果如下：  // x=10, y=10  // img[x,y] = [113 131 226]  // img[10,10,ch]: 113 131 226  // img.item(10,10,ch): 113 131 226  // original img[x,y] = [113 131 226]  // updated img[x,y] = [255 131 226] |



## 07. 图像的创建（np.zeros）

|  |
| --- |
| # 1.14 Numpy 创建图像  # 创建彩色图像(RGB)  # (1) 通过宽度高度值创建多维数组  height, width, channels = 400, 300, 3 # 行/高度, 列/宽度, 通道数  imgEmpty = np.empty((height, width, channels), np.uint8) # 创建空白数组  imgBlack = np.zeros((height, width, channels), np.uint8) # 创建黑色图像 RGB=0  imgWhite = np.ones((height, width, channels), np.uint8) \* 255 # 创建白色图像 RGB=255  # (2) 创建相同形状的多维数组  img1 = cv2.imread("../images/imgLena.tif", flags=1) # flags=1 读取彩色图像(BGR)  imgBlackLike = np.zeros\_like(img1) # 创建与 img1 相同形状的黑色图像  imgWhiteLike = np.ones\_like(img1) \* 255 # 创建与 img1 相同形状的白色图像  # (3) 创建彩色随机图像 RGB=random  import os  randomByteArray = bytearray(os.urandom(height \* width \* channels))  flatNumpyArray = np.array(randomByteArray)  imgRGBRand = flatNumpyArray.reshape(height, width, channels)  # (4) 创建灰度图像  imgGrayWhite = np.ones((height, width), np.uint8) \* 255 # 创建白色图像 Gray=255  imgGrayBlack = np.zeros((height, width), np.uint8) # 创建黑色图像 Gray=0  imgGrayEye = np.eye(width) # 创建对角线元素为1 的单位矩阵  randomByteArray = bytearray(os.urandom(height\*width))  flatNumpyArray = np.array(randomByteArray)  imgGrayRand = flatNumpyArray.reshape(height, width) # 创建灰度随机图像 Gray=random  print("Shape of image: gray {}, RGB {}".format(imgGrayRand.shape, imgRGBRand.shape))  cv2.imshow("DemoGray", imgGrayRand) # 在窗口显示 灰度随机图像  cv2.imshow("DemoRGB", imgRGBRand) # 在窗口显示 彩色随机图像  cv2.imshow("DemoBlack", imgBlack) # 在窗口显示 黑色图像  key = cv2.waitKey(0) # 等待按键命令  // numpy.empty(shape[, dtype, order]) # 返回一个指定形状和类型的空数组  // numpy.zeros(shape[, dtype, order]) # 返回一个指定形状和类型的全零数组  // numpy.ones(shape[, dtype, order]) # 返回一个指定形状和类型的全一数组  // numpy.empty\_like(img) # 返回一个与图像 img 形状和类型相同的空数组  // numpy.zeros\_like(img) # 返回一个与图像 img 形状和类型相同的全零数组  // numpy.ones\_like(img) # 返回一个与图像 img 形状和类型相同的全一数组 |



## 08. 图像的复制（np.copy）

## 09. 图像的裁剪（cv2.selectROI）

## 10. 图像的拼接（np.hstack）

## 11. 图像通道的拆分（cv2.split）

## 12. 图像通道的合并（cv2.merge）

# 2. 图像的数值运算

## 13. 图像的加法运算（cv2.add）

## 14. 图像与标量相加（cv2.add）

## 15. 图像的加权加法（cv2.addWeight）

## 16. 不同尺寸的图像加法

## 17. 两张图像的渐变切换

## 18. 图像的掩模加法

## 19. 图像的圆形遮罩

## 20. 图像的按位运算

## 21. 图像的叠加

## 22. 图像添加非中文文字

## 23. 图像添加中文文字

# 3. 图像的仿射变换

## 24. 图像的仿射变换

## 25. 图像的平移

## 26. 图像的旋转（以原点为中心）

## 27. 图像的旋转（以任意点为中心）

## 28. 图像的旋转（直角旋转）

## 29. 图像的翻转（cv2.flip）

## 30. 图像的缩放（cv2.resize）

## 31. 图像金字塔（cv2.pyrDown）

## 32. 图像的扭变（错切）

## 33. 图像的复合变换

## 34. 图像的投影变换

## 35. 图像的投影变换（边界填充）

## 36. 直角坐标与极坐标的转换

# 4. 灰度变换与直方图

## 37. 图像的灰度化处理和二值化处理

## 38. 图像的反色变换（图像反转）

## 39. 图像灰度的线性变换

## 40. 图像分段线性灰度变换

## 41. 图像的灰度变换（灰度级分层）

## 42. 图像的灰度变换（比特平面分层）

## 43. 图像的灰度变换（对数变换）

## 44. 图像的灰度变换（伽马变换）

## 45. 图像的灰度直方图

## 46. 直方图均衡化

## 47. 直方图处理之直方图匹配

## 48. 直方图处理之彩色直方图匹配

## 49. 直方图处理之局部直方图处理（cv2.createCLAHE）

## 50. 直方图处理之直方图统计量图像增强

## 51. 直方图处理之直方图反向追踪（cv2.calcBackProject）

# 5. 空间域图像滤波

## 52. 图像的相关与卷积运算

## 53. Scipy 实现图像二维卷积

## 54. OpenCV 实现图像二维卷积

## 55. 可分离卷积核

## 56. 低通盒式滤波器

## 57. 低通高斯滤波器

## 58. 非线性滤波—中值滤波

## 59. 非线性滤波—双边滤波

## 60. 非线性滤波—联合双边滤波

## 61. 导向滤波（Guided filter）

## 62. 图像锐化——钝化掩蔽

## 63. 图像锐化——Laplacian 算子

## 64. 图像锐化——Sobel 算子

## 65. 图像锐化——Scharr 算子

## 66. 图像滤波之低通/高通/带阻/带通

## 67. 空间域图像增强的综合应用

# 6. 频率域图像滤波

## 68. 连续周期信号的傅立叶级数

## 69. 连续非周期信号的傅立叶系数

## 70. 一维连续函数的傅里叶变换

## 71. 连续函数的取样

## 72. 一维离散傅里叶变换

## 73. 二维连续傅里叶变换

## 74. 图像的抗混叠

## 75. Numpy 实现图像傅里叶变换

## 76. OpenCV 实现图像傅里叶变换

## 77. OpenCV 实现快速傅里叶变换

## 78. 频率域图像滤波基础

## 79. 频率域图像滤波的基本步骤

## 80. 频率域图像滤波详细步骤

## 81. 频率域高斯低通滤波器

## 82. 频率域巴特沃斯低通滤波器

## 83. 频率域低通滤波：印刷文本字符修复

## 84. 由低通滤波器得到高通滤波器

## 85. 频率域高通滤波器的应用

## 86. 频率域滤波应用：指纹图像处理

## 87. 频率域钝化掩蔽

## 88. 频率域拉普拉斯高通滤波

## 89. 带阻滤波器的传递函数

## 90. 频率域陷波滤波器

# 7. 图像复原与重建

## 91. 高斯噪声、瑞利噪声、爱尔兰噪声

## 92. 指数噪声、均匀噪声、椒盐噪声

## 93. 噪声模型的直方图

## 94. 算术平均滤波器

## 95. 几何均值滤波器

## 96. 谐波平均滤波器

## 97. 反谐波平均滤波器

## 98. 统计排序滤波器

## 99. 修正阿尔法均值滤波器

## 100. 自适应局部降噪滤波器

## 101. 自适应中值滤波器

## 102. 陷波带阻滤波器的传递函数

## 103. 陷波带阻滤波器消除周期噪声干扰

## 104. 运动模糊退化模型

## 105. 湍流模糊退化模型

## 106. 退化图像的逆滤波

## 107. 退化图像的维纳滤波

## 108. 约束最小二乘方滤波

## 109. 几何均值滤波

## 110. 投影和雷登变换

## 111. 雷登变换反投影重建图像

## 112. 滤波反投影重建图像

# 8. 形态学图像处理

## 113. 形态学操作之腐蚀

## 114. 形态学操作之膨胀

## 115. 形态学操作之开运算

## 116. 形态学操作之闭运算

## 117. 形态学操作之顶帽运算

## 118. 形态学操作之底帽运算

## 119. 图像的形态学梯度

## 120. 击中-击不中变换

## 121. 击中-击不中用于特征识别

## 122. 形态算法之边界提取

## 123. 形态算法之孔洞填充

## 124. 孔洞填充的泛洪算法

## 125. 形态算法之提取连通分量

## 126. 形态算法之凸壳

## 127. 形态算法之细化

## 128. 形态算法之骨架 (skimage)

## 129. 形态算法之骨架 (重建开运算)

## 130. 形态学之提取水平和垂直线

## 131. 形态学重建之竖线字符提取

## 132. 形态学重建之孔洞填充算法

## 133. 形态学重建之边界清除

## 134. 形态学重建之细胞计数

## 135. 形态学重建之粒度测定

## 136. 灰度腐蚀和灰度膨胀

## 137. 灰度开运算和灰度闭运算原理

## 138. 灰度开运算和灰度闭运算

## 139. 灰度顶帽变换校正阴影

## 140. 灰度底帽变换校正光照

## 141. 灰度底帽变换的三维地形图

## 142. 基于灰度形态学的图像平滑

## 143. 基于灰度形态学的粒度测定

## 144. 基于灰度形态学的纹理分割

## 145. 形态学之边缘和角点检测

## 146. 基于灰度形态学的复杂背景图像重建

# 9. 阈值处理与边缘检测

## 147. 图像分割之孤立点检测

## 148. 图像分割之线检测

## 149. 图像分割之边缘模型

## 150. 边缘检测梯度算子

## 151. 边缘检测中的平滑处理

## 152. 边缘检测之 LoG 算子

## 153. 边缘检测之 DoG 算子

## 154. 边缘检测之 Canny 算子

## 155. 边缘连接的局部处理方法

## 156. 边缘连接局部处理的简化算法

## 157. 霍夫变换直线检测

## 158. 阈值处理之固定阈值法

## 159. 图像分割之全局阈值处理

## 160. 图像处理之OTSU 方法

## 161. OTSU 阈值处理算法的实现

## 162. 全局阈值处理改进方法

## 163. 基于边缘信息改进全局阈值处理

## 164.使用 Laplace 边缘信息改进全局阈值处理

## 165.多阈值 OTSU 处理方法

## 166.自适应阈值处理

## 167.基于移动平均的可变阈值处理

# 10. 轮廓与图像分割

## 168.图像分割之区域生长

## 169.图像分割之区域分离

## 170.图像分割之K均值聚类

## 171.SLIC 超像素区域分割

## 172.SLIC 超像素区域分割算法比较

## 173.SEEDS 超像素区域分割

## 174.LSC 超像素区域分割

## 175.超像素区域分割方法比较

## 176.图像分割之均值漂移算法

## 177.图像分割之图割法 GraphCuts

## 178.图像分割之 GrabCut 图割法（框选前景）

## 179.图像分割之 GrabCut 图割法（掩模图像）

## 180.基于距离变换的分水岭算法

## 181.基于 Sobel 梯度的分水岭算法

## 182.基于形态学梯度的分水岭算法

## 183.基于轮廓标记的分水岭算法

## 184.鼠标交互标记的分水岭算法

## 185.图像金字塔之高斯金字塔

## 186.图像金字塔之拉普拉斯金字塔

## 187.由拉普拉斯金字塔还原图像

## 188.基于拉普拉斯金字塔的图像融合

## 189.基于掩模的拉普拉斯金字塔图像融合

## 190.基于图像分割的图像融合

## 191.基于图像分割的金字塔图像融合

## 192.Gabor 滤波器组的形状

## 193.基于Gabor 滤波器的特征提取

## 194.寻找图像轮廓（cv.findContours）

## 195.绘制图像轮廓（cv.drawContours）

## 196.图像的矩和不变矩（cv.moments）

## 197.轮廓的基本特征

## 198.基于不变矩的形状相似性检测

## 199.轮廓的外接边界框

## 200.轮廓的基本属性

# 11. 彩色图像

## 201. 图像的颜色空间转换

## 202. 查表快速替换（cv.LUT）

## 203. 伪彩色图像处理

## 204. 图像的色彩风格滤镜

## 205. 调节色彩平衡/饱和度/明度

## 206. Photoshop 色阶调整算法

## 207. Photoshop 色阶自动调整算法

## 208. Photoshop 对比度自动调整算法

## 209. HSV 颜色空间的图像分割

# 12. 绘制图像

## 210. 绘制直线也会有这么多坑？

## 211. 绘制垂直矩形

## 212. 绘制倾斜的矩形

## 213. 绘制圆形

## 214. 绘制椭圆的参数详解

## 215. 基于多段线绘制近似椭圆

## 216. 绘制多段线和多边形

## 217. 鼠标交互获取多边形区域

## 218. 多行倾斜文字水印

## 219. 添加数字盲水印

## 220.对图像进行马赛克处理

## 221.加密马赛克图像处理与解密复原

# 13. 特征提取

## 222. 特征提取之弗里曼链码

## 223. 特征提取之多边形拟合

## 224. 特征提取之提取骨架

## 225. 特征提取之傅里叶描述子

## 226. 区域特征之紧致度/圆度/偏心率

## 227. 特征描述之 LBP 纹理特征算子

## 228. 特征描述之 extendLBP 改进算子

## 229. 特征描述之 LBP 算子比较（skimage）

## 230. 特征描述之 LBP 统计直方图

## 231. 特征描述之 灰度共生矩阵（GLCM）

## 232. 纹理特征之频谱方法

## 233. 区域特征之矩不变量

## 234. 特征提取之主成分分析（PCA）

## 235. 特征提取之主成分分析（sklearn）

## 236. 特征提取之主成分分析（OpenCV.PCA）

## 237. 基于主成分提取的方向校正（OpenCV.PCA）

## 238. OpenCV 中的 Harris 角点检测

## 239. Harris 角点检测之精确定位（cornerSubPix）

## 240. OpenCV 中的 Shi-Tomas 角点检测

## 241. 尺度不变特征变换（SIFT）

## 242. 加速稳健特征检测算法（SURF）

## 243. 特征检测之 FAST 算法

## 244. 特征检测之 BRIEF 特征描述

## 245. 特征检测之 BRISK 算子

原文：https://blog.csdn.net/youcans/article/details/125112487