

1. 输入 / 输出 I/O

```
# 将图像加载为 BGR (如果为灰度图, 则为 B=G=R)
i = imread("name.png")
# 按原样加载图像 (包括透明度)
i = imread("name.png", IMREAD_UNCHANGED)
# 将图像加载为灰度图
i = imread("name.png", IMREAD_GRAYSCALE)

imshow("Title", i) # 显示图像 I
imwrite("name.png", i) # 保存图像 I
waitKey(500) # 等待 0.5 秒按键 (0 为永远等待)
destroyAllWindows() # 释放并关闭所有窗口
```

3. 通道操作 channel manipulation

```
# 将图像 I 分割为多个通道
b, g, r = split(i)
# 和上面一样, 但 I 有 a 个通道
b, g, r, a = split(i)
# 将通道合并到图像中
i = merge((b, g, r))
```

4. 算术运算 arithmetic operations

```
# min(I1+I2, 255), 例如饱和添加量
i = add(i1, i2)
# min(alpha*I1+beta*I2+gamma, 255), 例如图像混合
i = addWeighted(i1, alpha, i2, beta, gamma)
# max(I1-I2, 0), 例如饱和减法
i = subtract(i1, i2)
# |I1-I2|, 例如绝对差
i = absdiff(i1, i2)
注意: 其中一个图像可以替换为标量。
```

2. 颜色 / 亮度 color/intensity

亮度 intensity

```
i = equalizeHist(i) # 直方图均衡化
# 将 I 在 0 到 255 之间标准化
i = normalize(i, None, 0, 255, NORM_MINMAX, CV_8U)
# 将 I 在 0 到 1 之间标准化
i = normalize(i, None, 0, 1, NORM_MINMAX, CV_32F)
```

颜色 color

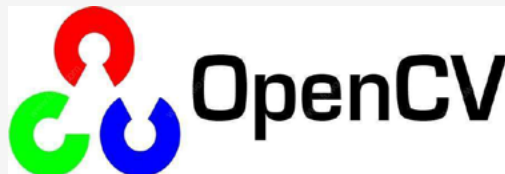
```
i_gray = cvtColor(i, COLOR_BGR2GRAY) # BGR 到 GRAY
i_rgb = cvtColor(i, COLOR_BGR2RGB) # BGR 到 RGB
i = cvtColor(i, COLOR_GRAY2RGB) # 将 GRAY 转换为 RGB
```

其他颜色空间 Other useful color spaces

```
COLOR_BGR2HSV # BGR to HSV (色调、饱和度、值)
COLOR_BGR2LAB # BGR to Lab (亮度、绿色 / 洋红色、蓝色 / 黄色)
COLOR_BGR2LUV # BGR to Luv (~ Lab, 但标准化方式不同)
COLOR_BGR2YCrCb # BGR to YCrCb (亮度、蓝色亮度、红色亮度)
```

5. 逻辑运算 logical operations

```
i = bitwise_not(i) # 反转 I 中的每一位
i = bitwise_and(i1, i2) # I1 和 I2 的逻辑 and
i = bitwise_or(i1, i2) # I1 和 I2 的逻辑 or
i = bitwise_xor(i1, i2) # I1 和 I2 的逻辑异或
```



6. 统计数字 statistics

```
mB, mG, mR, mA = mean(i) # 每个通道的平均值 (即 BGRA)
ms, sds = meanStdDev(i) # 平均值和 SDev p/ 通道 (各 3 或 4 行)
# 通道 c 直方图, 无掩码, 256 个分桶 (0-255)
h = calcHist([i], [c], None, [256], [0, 256])
# 使用通道 0 和通道 1 的 2D 直方图, 每个维度的“分辨率”为 256
h = calcHist([i], [0, 1], None, [256, 256], [0, 256, 0, 256])
```

7. 过滤 filtering

```
i = blur(i, (5, 5)) # 带 5*5 箱式过滤器的过滤器 I (即平均过滤器)

i = GaussianBlur(i, (5, 5), sigmaX=0, sigmaY=0) # 5x5 高斯模糊 I
i = GaussianBlur(i, None, sigmaX=2, sigmaY=2) # 高斯模糊
i = filter2D(i, -1, k) # 基于互相关的二维核滤波器
kx = getGaussianKernel(5, -1) # 长度为 5 的一维高斯核 (自动 StDev)

i = sepFilter2D(i, -1, kx, ky) # 使用可分离内核的过滤器
i = medianBlur(i, 3) # 尺寸为 3 的中值滤波器 (尺寸 ≥ 3)
i = bilateralFilter(i, -1, 10, 50) # sigma=10, sigmaS=50 的双边滤波器, 自动大小
```

8. 边界 borders

所有过滤操作都有参数 borderType, 可设置为:

```
BORDER_CONSTANT # 具有恒定边界的 Pads (需要附加参数值)
BORDER_REPLICATE # 将第一行 / 最后一行和列复制到 padding 上
BORDER_REFLECT # 将图像边框反射到 padding 上
BORDER_REFLECT_101 # 与前面相同, 但不包括边界处的像素 (默认值)
BORDER_WRAP # 环绕图像边框以构建填充
```

```
i = copyMakeBorder(i, 2, 2, 3, 1, borderType=BORDER_WRAP) # 自定义宽度添加边框
```

9. 微分算子 differential operators

```
i_x = Sobel(i, CV_32F, 1, 0) # x 方向的 Sobel 算子:  $I_x = (\partial/\partial x)I$ 
i_y = Sobel(i, CV_32F, 0, 1) # y 方向的 Sobel 算子:  $I_y = (\partial/\partial y)I$ 
i_x, i_y = spatialGradient(i, 3) #  $\nabla I$  (使用  $3 \times 3$  SobelSobel): 需要是 uint8 图片

m = magnitude(i_x, i_y) #  $I_x, I_y$  需要是浮点数类型
m, d = cartToPolar(i_x, i_y) #  $// \nabla I //$ ;  $\vartheta \in [0, 2\pi]$ ;
l = Laplacian(i, CV_32F, ksize=5) #  $\Delta I$ , 核大小为 5 的拉普拉斯算子
```

10. 几何变换 geometric transforms

```
i = resize(i, (width, height)) # 将图像大小调整为宽度 * 高度
i = resize(i, None, fx=0.2, fy=0.1) # 将图像缩放为 20% 宽度和 10% 高度

M = getRotationMatrix2D((xc, yc), deg, scale) # 返回  $2 \times 3$  旋转矩阵 M, 任意 (xc, yc)
M = getAffineTransform(pts1, pts2) # 由 3 个对应关系得到的一个 Affine 变换矩阵 M
i = warpAffine(i, M, (cols, rows)) # 将 Affine 变换 M 应用于 I, 输出大小 = (列, 行)

M = getPerspectiveTransform(pts1, pts2) # 由 4 个对应关系得到的透视变换矩阵 M
M, s = findHomography(pts1, pts2) # 透视变换矩阵  $M \geq 4$  对应 (最小二乘法)
M, s = findHomography(pts1, pts2, RANSAC) # 透视变换矩阵 M
i = warpPerspective(i, M, (cols, rows)) # 将透视变换 M 应用于图像 I
```

插值方法 interpolation methods

默认情况下, 调整大小、扭曲仿射和扭曲透视使用双线性插值。通过调整大小的参数插值和其他参数的标志进行更改:

```
flags = INTER_NEAREST # 最简单、最快
flags = INTER_LINEAR # 双线性插值: 默认值
flags = INTER_CUBIC # 双立方插值
```

11. 分割 segmentation

```
_ , i_t = threshold(i, t, 255, THRESH_BINARY) # 给定阈值级别 t 的手动阈值图像 I
t, i_t = threshold(i, 0, 255, THRESH_OTSU) # 使用 Otsu 返回阈值级别和阈值图像

# 具有块大小 b 和常数 c 的自适应 mean-c
i_t = adaptiveThreshold(i, 255, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, b, c)

# 仅使用色调和饱和度将直方图 h 反向投影到图像 i_hsv 上; 无缩放 (即 1)
bp = calcBackProject([i_hsv], [0, 1], h, [0, 180, 0, 256], 1)

# 返回 K 簇的标签 la 和中心 ct, 10 个簇中的最佳紧性 cp; 1 专长 / 纵队
cp, la, ct = kmeans(feats, K, None, crit, 10, KMEANS_RANDOM_CENTERS)
```

12. 特征 features

```
e = Canny(i, t1, th) # 返回 Canny 边 (e 是二进制的)
l = HoughLines(e, 1, pi/180, 150) # 返回全部  $\vartheta \rho, \vartheta \in [0, 180]$  票, 票面分辨率:  $\rho=1$  像素,  $\vartheta=1$  度
l = HoughLinesP(e, 1, pi/180, 150, None, 100, 20) # 概率霍夫, 最小长度=100, 最大间隙=20

# 返回所有 (xc, yc, r), 至少有 18 个投票, bin 分辨率=1, param1 是 Canny 的第 th 个, 中心必须至少相距 50 像素
c = HoughCircles(i, HOUGH_GRADIENT, 1, minDist=50, param1=200, param2=18, minRadius=20, maxRadius=60)

r = cornerHarris(i, 3, 5, 0.04) # Harris 角点的每像素 Rs, 窗口=3, Sobel=5,  $\alpha=0.04$ 
f = FastFeatureDetector_create() # 实例化星形特征检测器
k = f.detect(i, None) # 检测灰度图像 I 上的关键点
i_k = drawKeypoints(i, k, None) # 在彩色图像 I 上绘制关键点 k

d = xfeatures2d.BriefDescriptorExtractor_create() # 实例化一个简短的描述符
k, ds = d.compute(i, k) # 计算 I 上关键点 k 的描述符
dd = AKAZE_create() # 实例化 AKAZE 检测器 / 描述符

m = BFMatcher.create(NORM_HAMMING, crossCheck=True) # 使用 x 检查和汉明距离实例化 brute-force matcher
ms = m.match(ds_l, ds_r) # 匹配左右描述符
i_m = drawMatches(i_l, k_l, i_r, k_r, ms, None) # 使用 matches ms 从左图像 I_l 上的左关键点 k_l 到右 I_r 绘制匹配
```

13. 检测 detection

```
# 将模板  $T$  与图像  $I$  匹配 (标准化 X-correl)  
ccs = matchTemplate(i, t, TM_CORR_NORMED)  
# ccs 中的最小值、最大值和相应坐标  
m, M, m_l, M_l = minMaxLoc(ccs)  
# 创建“空”级联分类器的实例  
c = CascadeClassifier()  
# 从文件加载预先训练的模型;  $r$  是真 / 假  
r = c.load("file.xml")  
# 每个检测到的对象返回 1 个元组 (x, y, w, h)  
objs = c.detectMultiScale(i)
```

14. 运动与追踪 motion and tracking

```
# 返回 100 个 Shi-Tomasi 角点, 质量至少为 0.5, 彼此相距 10 像素  
pts = goodFeaturesToTrack(i, 100, 0.5, 10)  
  
# 根据  $I_0$  和  $I_1$  之间的估计光流确定 pts 的新位置  
# 如果找到点  $i$  的流量, 则  $st[i]$  为 1, 否则为 0  
pts1, st, e = calcOpticalFlowPyrLK(i0, i1, pts0, None)  
  
t = TrackerCSRT_create() # 实例化 CSRT 跟踪器  
r = t.init(f, bbox) # 使用框架和边界框初始化跟踪器  
r, bbox = t.update(f) # 返回给定下一帧的新边界框
```

15. 图像绘制 drawing on the image

```
line(i, (x0, y0), (x1, y1), (b, g, r), t) # Line 线  
rectangle(i, (x0, y0), (x1, y1), (b, g, r), t) # Rectangle 长方形  
circle(i, (x0, y0), radius, (b, g, r), t) # Circle 圆  
polylines(i, [pts], True, (b, g, r), t) # 闭合 (真) 多边形 (pts 是点数组)  
# 在  $0 \times 0 \times 0$  处写入“Hi”, 字体大小 = 1, 粗细 = 2  
putText(i, "Hi", (x, y), FONT_HERSHEY_SIMPLEX, 1, (r, g, b), 2, LINE_AA)
```

参数 parameters

(x0, y0) # 原点 / 起点 / 左上角
(x1, y1) # 右下角
(b, g, r) # 线条颜色 (uint8)
t # 线粗细 (填充, 如果为负值)

16. 立体相机标定 calibration and stereo

```
r, crns = findChessboardCorners(i, (n_x, n_y)) # 检测角点的二维坐标;  $i$  是灰度;  $r$  是状态; ( $n_x, n_y$ ) 是校准目标的大小  
crnrs = cornerSubPix(i, crns, (5, 5), (-1, -1), crit) # 应用亚像素精度提高坐标  
  
# 计算内部 (包括畸变系数) 和外部 (即每个目标视图 /  $R+T$ ),  $crns\_3D$  包含 1 个 3D 角坐标阵列  $p$  / 目标视图,  $crns\_2D$  包含相应的 2D 角坐标阵列 (即 1 个  $crns$   $p$  / 目标视图)  
r, K, D, ExRs, ExTs = calibrateCamera(crns_3D, crns_2D, i.shape[:2], None, None)
```

```
drawChessboardCorners(i, (n_x, n_y), crns, r) # 在  $I$  上绘制 corners 角点;  $r$  是角点检测的状态  
u = undistort(i, K, D) # 使用 intrinsics 取消  $I$  的变形  
s = StereoSGBM_create(minDisparity=0, numDisparities=32, blockSize=11) # 实例化半全局块匹配方法  
s = StereoBM_create(32, 11) # 实例化一个更简单的块匹配方法  
d = s.compute(i_L, i_R) # 计算视差图 ( $\infty - 1$  个深度图)
```



17. 终止标准 termination criteria

```
# 20 次迭代后停止  
crit = (TERM_CRITERIA_MAX_ITER, 20, 0)  
# 如果“movement”小于 1.0, 则停止  
crit = (TERM_CRITERIA_EPS, 0, 1.0)  
# 上两项任何一项发生就停止  
crit = (TERM_CRITERIA_MAX_ITER | TERM_CRITERIA_EPS, 20, 1.0)
```

OpenCV 是一个基于 Apache2.0 许可 (开源) 发行的跨平台计算机视觉和机器学习软件库, 可以运行在 Linux、Windows、Android 和 Mac OS 操作系统上。
OpenCV 轻量且高效——由一系列 C 函数和少量 C++ 类构成, 同时提供了 Python、Ruby、MATLAB 等语言的接口, 实现了图像处理和计算机视觉方面的很多通用算法。

本篇为 Python 版本的 OpenCV 使用速查手册。

18. 常用知识点总结 useful staff

Numpy (np.)

```
m = mean(i) # 阵列 I 的平均值
m = average(i, weights) # 数组 I 的加权平均值
v = var(i) # 阵列 / 图像 I 的方差
s = std(i) # 阵列 / 图像 I 的标准偏差

# numpy 柱状图也会返回分桶 /bins b
h, b = histogram(i.ravel(), 256, [0,256])
i = clip(i, 0, 255) # numpy 的饱和 / 固定功能
i = i.astype(np.float32) # 将图像类型转换为 float32
x, _, _, _ = linalg.lstsq(A, b) # 解决最小二乘问题
i = hstack((i1, i2)) # 并排合并 I1 和 I2
i = vstack((i1, i2)) # 将 I1 堆叠到 I2 之上
i = fliplr(i) # 左右翻转图像
i = flipud(i) # 上下翻转图像

# copyMakeBorder 的另一种写法 (也包括顶部、底部、左侧、右侧)
i = pad(i, ((1, 1), (3, 3)), 'reflect')
idx = argmax(i) # I 中最大值的线性指数 (即平坦 I 的指数)
r, c = unravel_index(idx, i.shape) # 索引相对于 i 形状的 2D 坐标
b = any(M>5) # 如果数组 M 中的任何元素大于 5, 则返回 True
b = all(M>5) # 如果数组 M 中的所有元素都大于 5, 则返回 True
rows, cols = where(M>5) # 返回 M 中的元素大于 5 的行和列的索引
coords = list(zip(rows, cols)) # 创建一个包含成对行和列元素的列表

M_inv = linalg.inv(M) # M 的逆
rad = deg2rad(deg) # 将角度转换为弧度
```

Matplotlib.pyplot (plt.)

```
imshow(i, cmap="gray", vmin=0, vmax=255) # matplotlib 的 imshow 阻止自动归一化
quiver(xx, yy, i_x, i_y, color="green") # 在 xx、yy 位置绘制渐变方向
savefig("name.png") # 将绘图另存为图像
```

Show Me AI



扫码回复“工具库”
下载最新全套速查表

OpenCV 4.X 速查表 (Python 版本)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

参考 | Antonio Anjos

数据科学工具库速查表



NumPy 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



SciPy 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



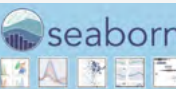
Pandas 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



Matplotlib 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



Seaborn 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



Bokeh 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```

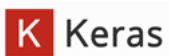


PySpark 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

AI 垂直领域工具库速查表



Scikit-learn 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



Keras 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



PyTorch 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



Hugging Face 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



OpenCV 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

编程语言速查表



SQL 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



Python 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

AI 知识技能速查表



Jupyter Notebook 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



正则表达式 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表