CHAPTER 5

基于OpenCV的机器视觉

主讲人：吴春彪

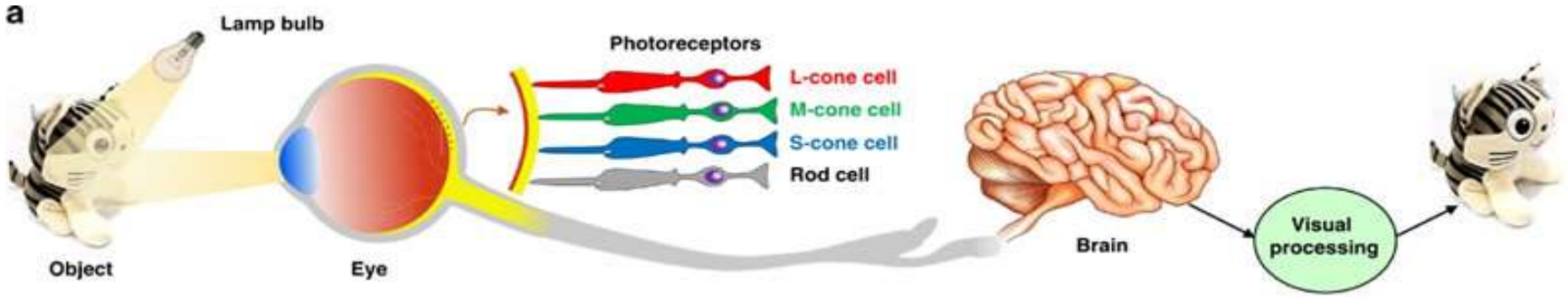
2025年6月8日

01

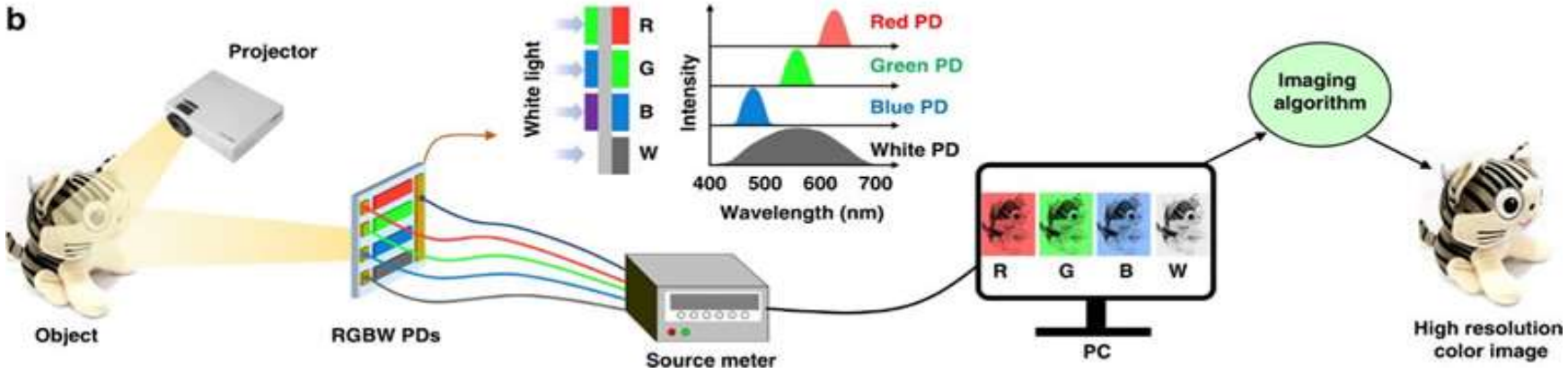
计算机视觉

计算机视觉

- **人类视觉系统** ： 是一个复杂的生物系统，光照射在物体上发生反射，当反射光映射在视网膜上，信息通过视觉神经传输到大脑进行处理。



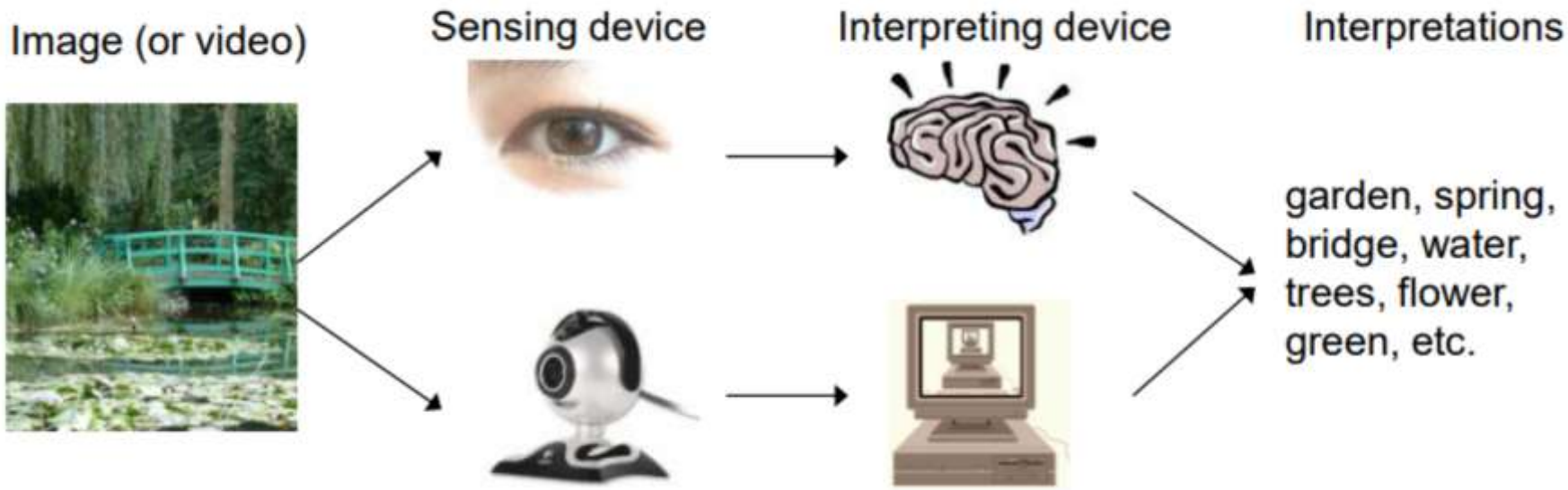
- **计算机视觉系统**：旨在让计算机理解和处理视觉信息，计算机视觉通过**摄像头**获取图像数据，转变为数字信号传输到CPU进行处理。



计算机视觉

■ 计算机视觉的基本概念

- **计算机视觉(Computer Vision, CV)**：是一门研究如何让**计算机实现人类视觉系统功能**的学科。主要研究各种图形、图像(如文学、照片、医学图像、视频图像等)的处理和识别技术。
- **计算机视觉**：给计算机安装上“**眼睛”(摄像机)**和“**大脑”(算法)**，代替人眼对目标进行**识别、跟踪**、 测量等。



计算机视觉

■ 计算机视觉的基本概念

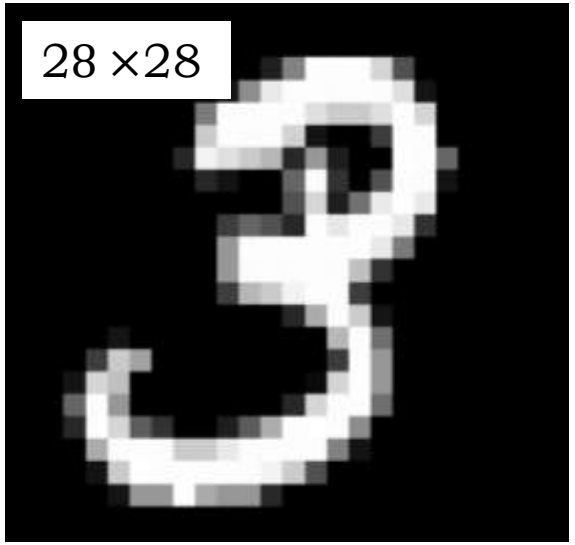
- 像素 (Pixel)：组成图像的**最基本单位**。一张图片由多个像素点以其**明确的位置**和**分配的色彩数值**构成。单位面积上像素点越多，图片清晰细腻。
- 分辨率(Resolution)：一张图片在**长和宽的两个方向上各拥有的像素个数**。

✓ 例如：一张640×480的图片，表示图片在每一个长度的方向上都有640个像素点，每一个宽度方向上都480个像素点，总数就是640X480=307200（个像素），简称30万像素。

① 100万像素：1140×900= 1026000

② 1000万像素：3648×2736= 9980928

✓ **单位面积上像素点越多即像素点越小，这图片就越清晰细腻。**



计算机视觉

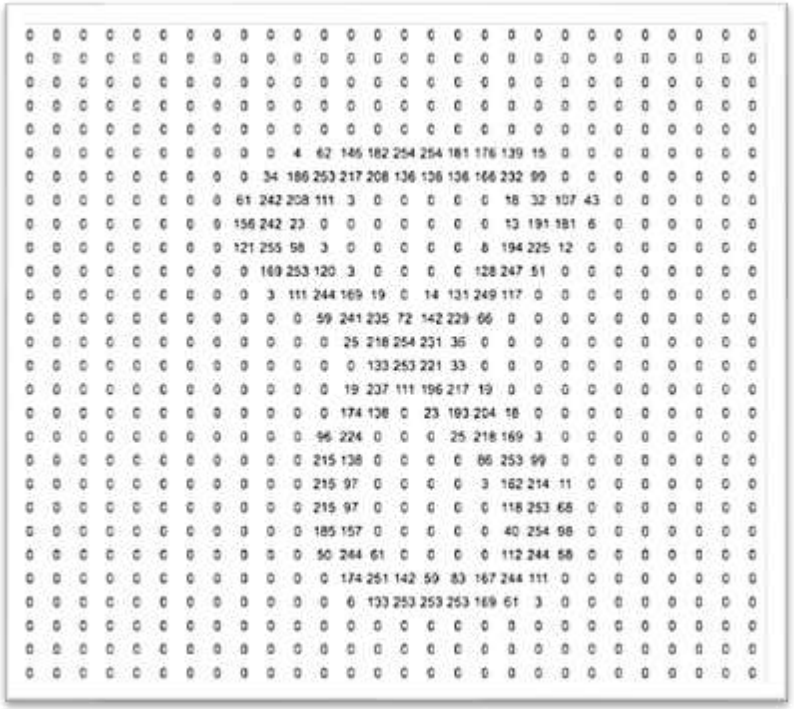
■ 计算机视觉的基本概念

- 像素矩阵

28 ×28
784 pixels

图片读取

取值范围[0，255]



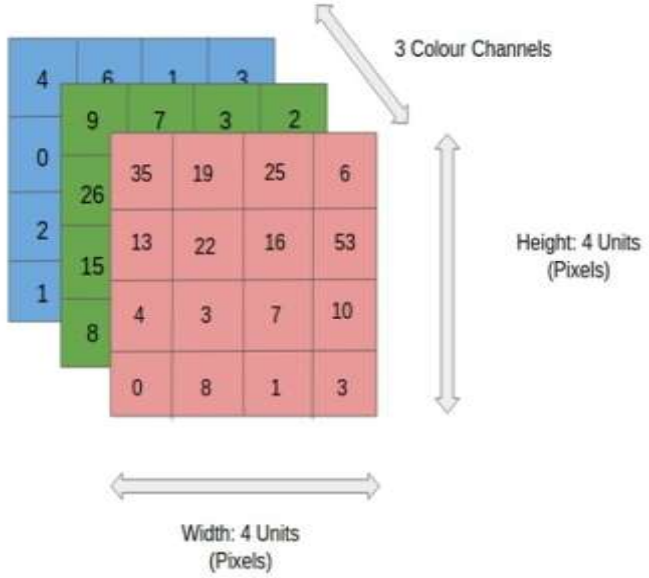
28 ×28 矩阵

计算机视觉

■ 计算机视觉的基本概念

- 彩色图像：采用RGB（红:Red、绿:Green 、蓝:Blue）**三个像素矩阵共同描述**，矩阵中的每个数字的取值范围为[0, 255]。比如 **RGB(255,0,0)表示红色， RGB (255, 255, 255)表示白色， RGB(0,0,0)表示黑色。**

- 每个像素矩阵称之为通道，因此，灰度图像为单通道；彩色图像为3通道。

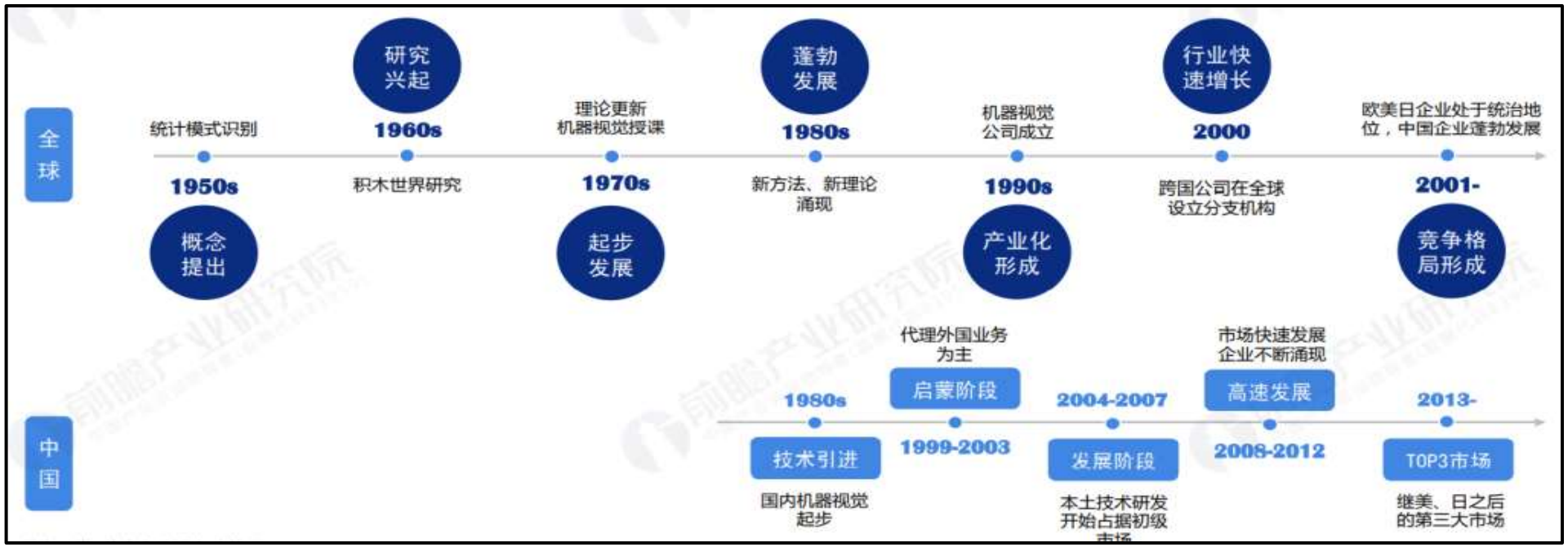


颜色	HTML / CSS名称	十六进制代码 #RRGGBB	十进制代码 (R, G, B)
	黑色	#000000	(0,0,0)
	白色	#FFFFFF	(255,255,255)
	红色	#FF0000	(255,0,0)
	酸橙	#00FF00	(0,255,0)
	蓝色	#0000FF	(0,0,255)
	黄色	#FFFF00	(255,255,0)
	青色/水色	#00FFFF	(0,255,255)
	洋红色/紫红色	#FF00FF	(255,0,255)
	银	#C0C0C0	(192,192,192)
	灰色	#808080	(128,128,128)
	栗色	#800000	(128,0,0)
	橄榄	#808000	(128,128,0)
	绿色	#008000	(0,128,0)
	紫色	#800080	(128,0,128)
	蓝绿色	#008080	(0,128,128)
	海军	#000080	(0,0,128)

计算机视觉

■ 计算机视觉的发展历程

计算机视觉始于1960年代，早期研究聚焦**简单模式识别** → 1980年代引入**机器学习** → 1990年代出现**特征提取方法**（如SIFT） → 2000年后，**深度学习**（尤其卷积神经网络CNN）革命性地提升了图像分类、目标检测等任务的性能 → 近年来，**Transformer**和**三维视觉技术**进一步推动领域发展，广泛应用于医疗、自动驾驶等领域。



计算机视觉

■ 计算机视觉的工作原理

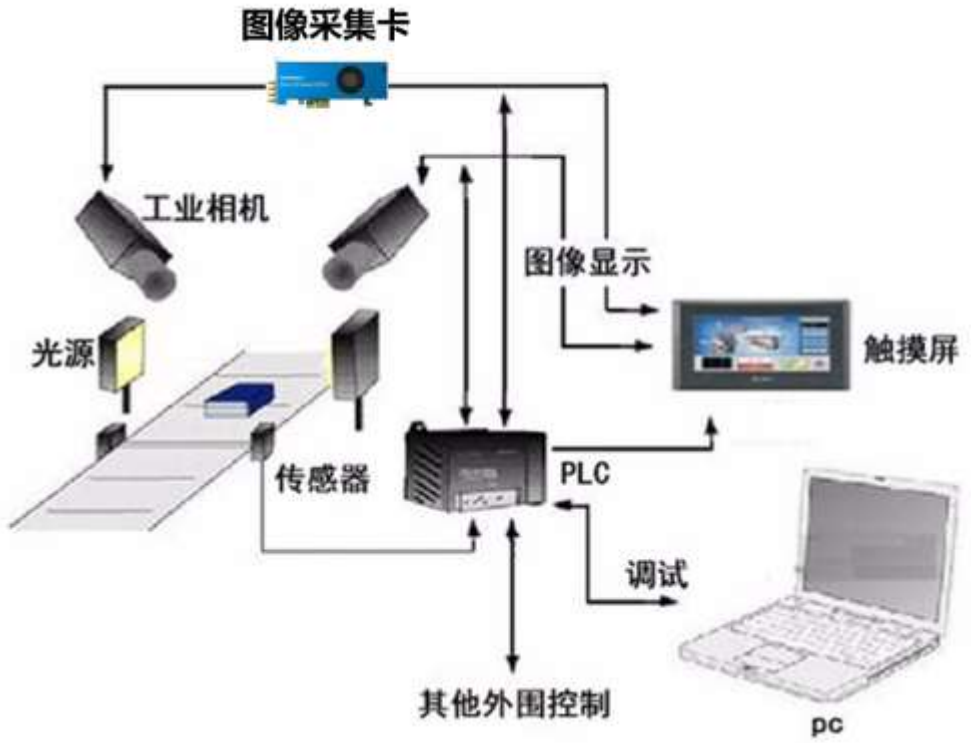
① **图像采集**：通过摄像头、传感器等设备采集数字图像或视频（如RGB图像、深度图、红外图像等）。

② **预处理**：对图像进行去噪、增强、归一化等操作，提高数据质量。

③ **特征提取**：利用传统算法（如颜色、边缘检测 、角点检测 、纹理分析）或深度学习（如卷积神经网络CNN）提取关键特征。

④ **模型训练**：采用机器学习或深度学习算法（如分类、检测、分割模型）进行数据分析。

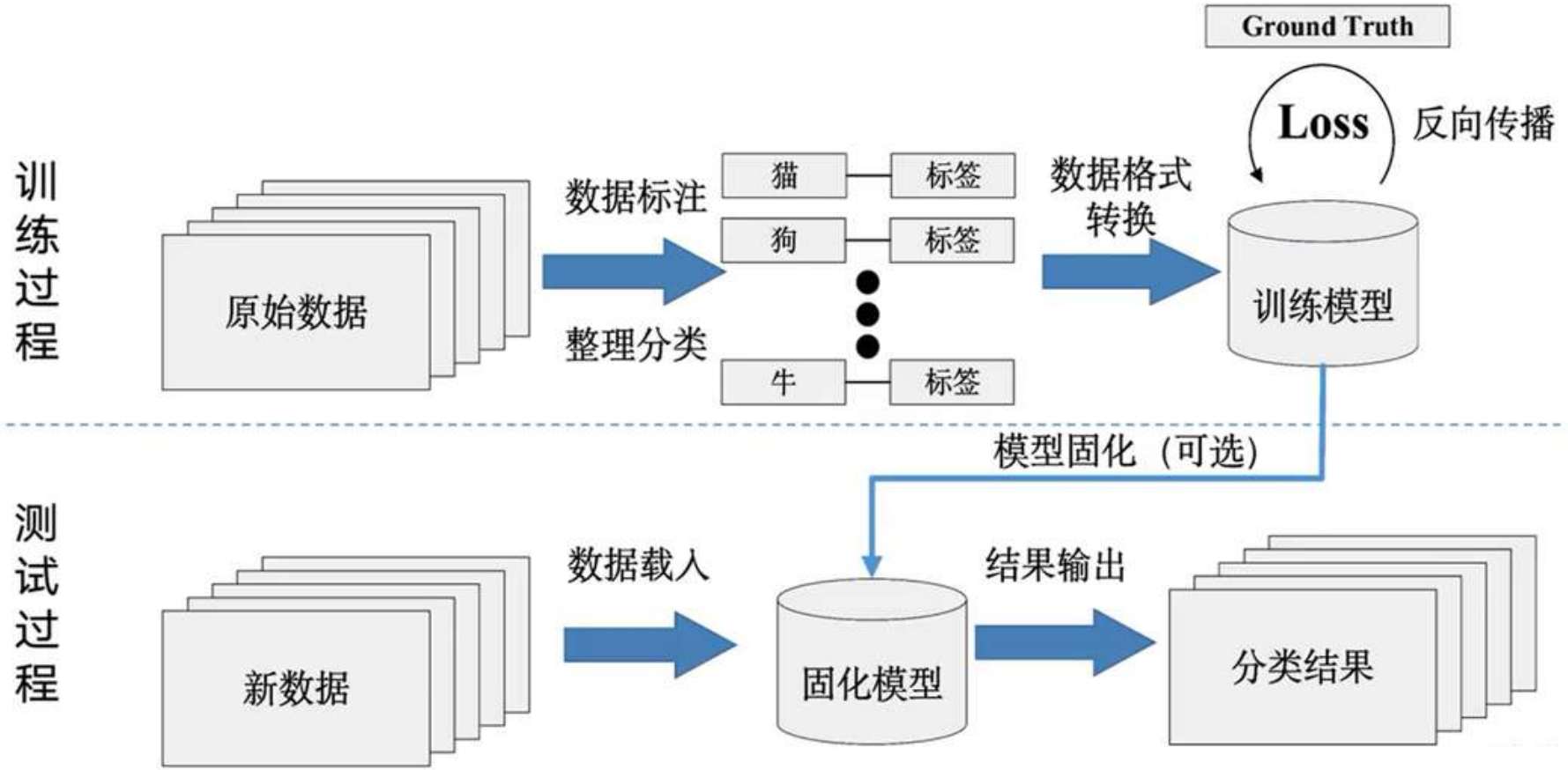
⑤ **输出结果**：根据分析结果执行任务（如工业机器人抓取目标、自动驾驶车辆转向）。



计算机视觉

■ 计算机视觉的工作原理

- 收集图像及相应标签，形成数据集→使用机器学习训练分类器→新的图像，测试性能



计算机视觉

■ 计算机视觉的核心任务

（1）图像分类（Image Classification）：根据图像的原始特征（如**边缘、颜色分布、纹理、形状**等）或 语义信息（**组合低层特征，形成高层语义**）对不同类别图像进行区分。是计算机视觉中重要的基础问题，是物体检测、图像分割、物体跟踪、行为分析、人脸识别等其他高层视觉任务的基础。

计算机视觉

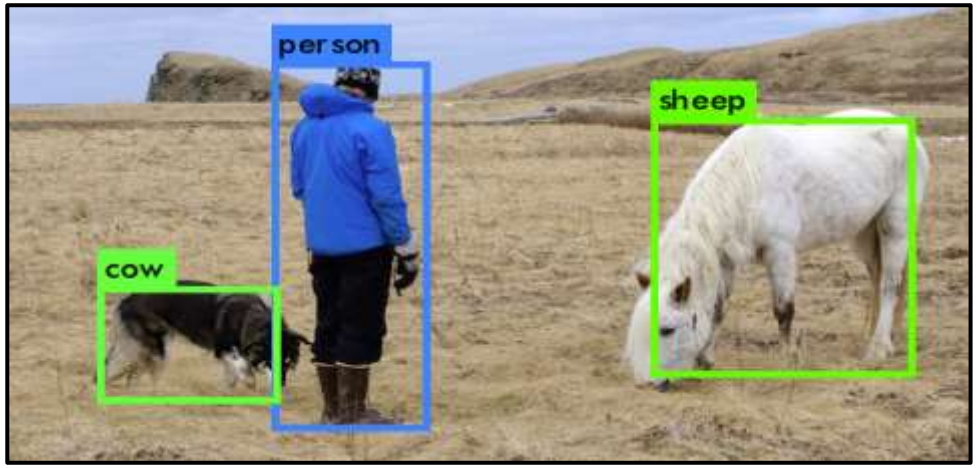
■ 计算机视觉的核心任务

（2）目标检测（Object Detection）：旨在**识别图像或视频中的特定物体，并确定它们的位置和类别。**

- **目标检测** 需要同时完成：

① **目标定位（Localization）**：用边界框Bounding Box的坐标（ **x, y, width, height** ）标出物体的位置和大小。

② **目标分类（Classification）**：判断边界框内的物体属于哪个类别（如人、车、狗等），模型对检测结果的可靠程度评分。



计算机视觉

■ 计算机视觉的核心任务

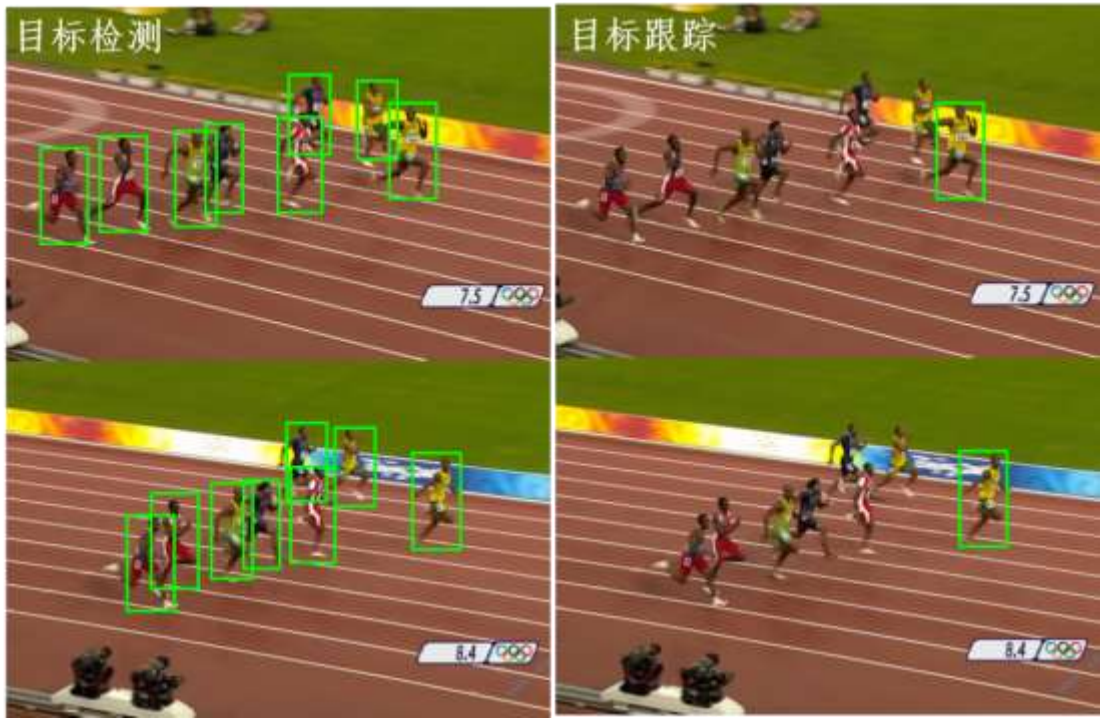
（3）目标跟踪（Object Tracking）：是计算机视觉的核心任务之一，旨在**在视频序列中持续定位特定目标的位置和状态。**其核心是关联帧与帧之间的目标，解决**“目标在哪”**和**“目标是谁”**的问题。

- **视频中的每一帧，跟踪算法输出：**

① **目标位置**：通常用边界框（Bounding Box）、椭圆或掩码（Mask）表示。

② **目标ID**：区分不同目标（尤其在多目标跟踪中）。

③ **目标状态**：如运动速度、方向、外观变化等。

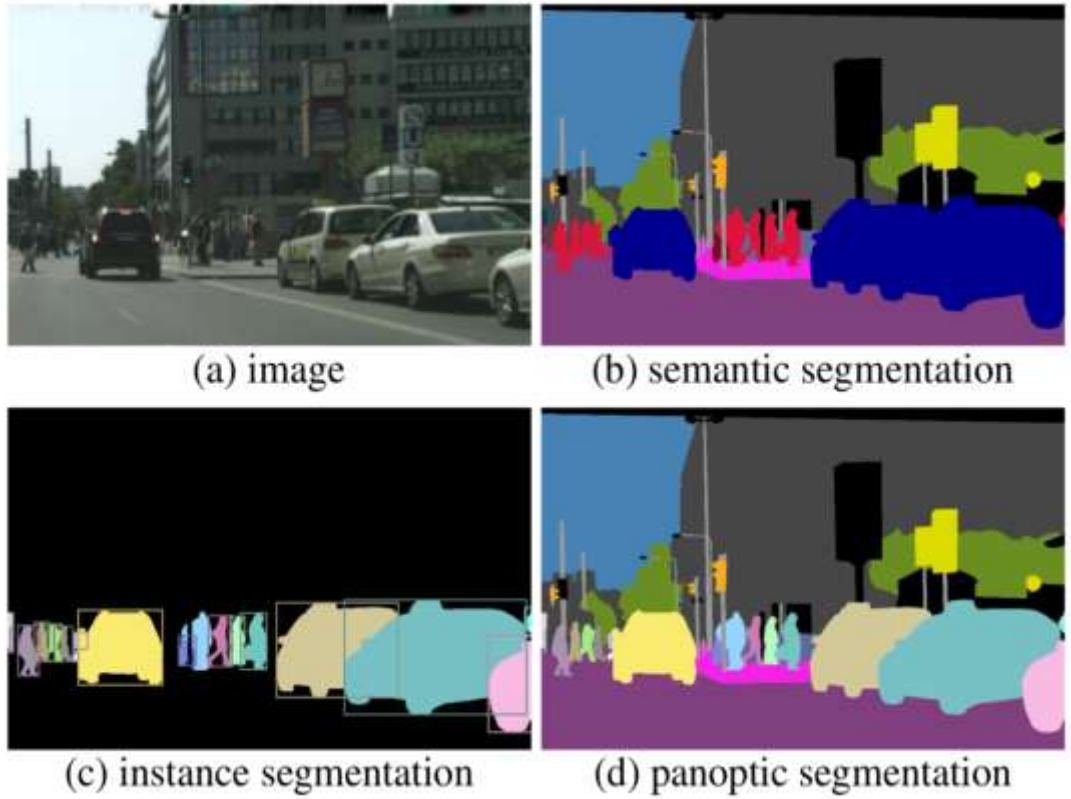


计算机视觉

■ 计算机视觉的核心任务

（4）图像分割（Image Segmentation）：目标是将图像划分为多个具有特定语义的区域，使得**每个区域内的像素具有相似的特征**（如类别、纹理、颜色等），图像分割关注的是像素级别的精细理解

- **语义分割（Semantic Segmentation）**像素级别上的分类，为每个像素分配类别标签，属于同一类的像素归为一类，**不区分同一类别的不同实例**；
- **实例分割（Instance Segmentation）**：语义分割的基础上，分割出每个实例物体，**区分同一类别的不同个体。**
- **全景分割（Panoptic Segmentation）**：结合语义分割和实例分割，**同时标记可数物体（如人、车）和不可数区域（如天空、草地）**



计算机视觉

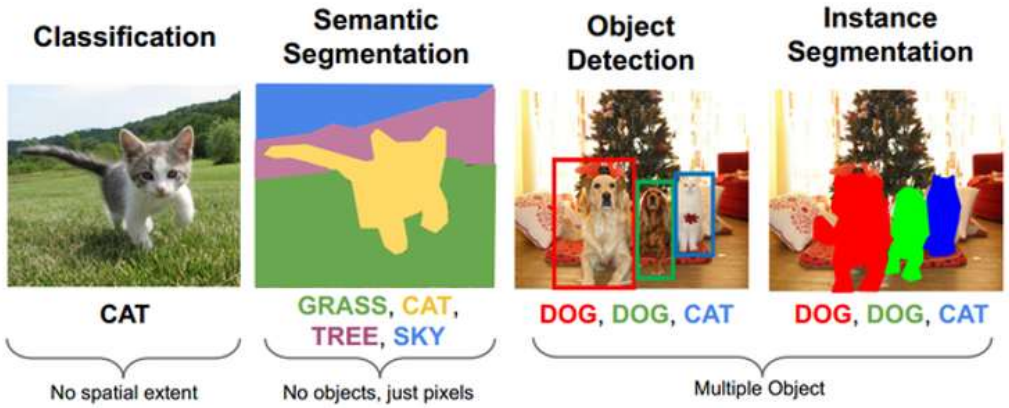
■ 计算机视觉的核心任务

a) 图像分类 （Image Classification）

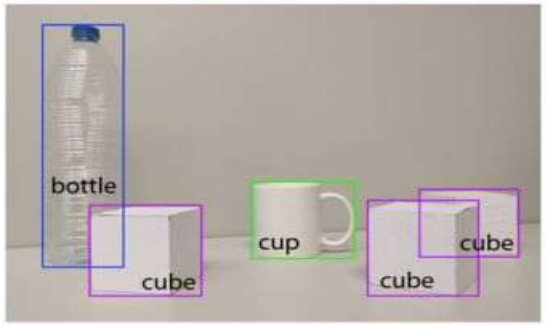
b) 目标检测（Object Detection）

c) 语义分割（Semantic Segmentation）

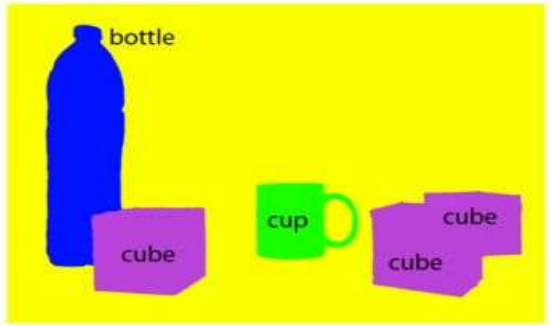
d) 实例分割（Instance Segmentation）



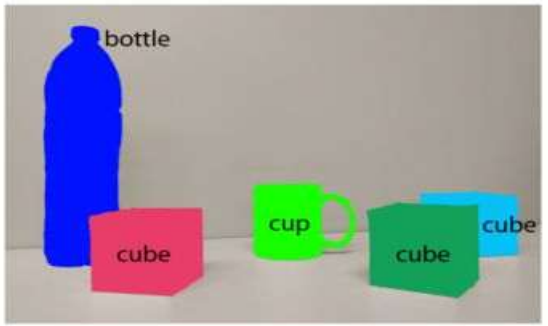
(a) Image classification



(b) Object localization



(c) Semantic segmentation



(d) Instance segmentation

计算机视觉

■ 计算机视觉的应用领域

计算机视觉广泛应用于**工业质检、自动驾驶、医疗影像分析、安防监控、零售（如无人商店）、农业（作物监测）、娱乐（AR/VR）、军事侦察、金融身份认证及智慧城市**等领域。


京东快递机器人


停车场自动收费系统


门禁系统


‘刷脸’支付

计算机视觉

■ 计算机视觉的难点与挑战

- **复杂场景理解**：现实世界场景多变，**光照、遮挡、天气**等因素影响识别精度。
- **小样本学习**：许多领域**缺乏足够标注数据**训练鲁棒模型。
- **实时性要求**：自动驾驶等应用需要**毫秒级响应**。
- **计算资源限制**：高精度模型需要强大算力，**难以部署在边缘设备**。
- **对抗攻击**：**图像微小扰动**可能导致模型误判。
- **跨领域泛化**：在特定领域训练的模型**难以直接迁移**到新场景。
- **3D场景理解**：从**2D图像准确重建3D世界**仍具挑战性。
- **隐私保护**：人脸识别等应用**涉及敏感数据安全问题**。

02

计算机视觉库: OpenCV

计算机视觉库: OpenCV

一、OpenCV 概述

■ OpenCV（Open Source Computer Vision Library）:是英特尔于1999年起开发的一个基于BSD许可（开源）发行的跨平台计算机视觉库，OpenCV 提供了大量的**计算机视觉算法**和**图像处理工具**，广泛应用于图像和视频的处理、分析以及机器学习领域。

- <官网>：https://opencv.org/
- <中文社区>：https://docs.opencv.ac.cn/4.11.0/
- <安装> anaconda模式 → conda install opencv
- <导入> import cv2 as cv
- 强大的图像处理功能：**图像过滤、形态学操作、图像变换、几何校正**等。
- 计算机视觉算法支持：**特征检测和描述、图像分割、目标检测和跟踪、相机标定**等。

计算机视觉库: OpenCV

二、OpenCV的发展历程

1. **OpenCV 1.x 版本（2000年）**：OpenCV最初由Intel于1999年开发，早期版本为1.x系列。这些早期版本主要包含一些基本的图像处理和计算机视觉功能，如图像加载、保存、转换、滤波等。
2. **OpenCV 2.x 版本（2009年）**：OpenCV 2.x版本是较大的重大版本更新，其中引入了许多重要的变化和增强功能。该版本支持多个操作系统平台，包括Windows、Linux、Mac等。此外，OpenCV2.x 还引入了新的模块、工具和接口，并对性能进行了优化。
3. **OpenCV 3.x 版本（2015年）**：OpenCV 3.x版本是另一个重大版本更新，引入了更多的功能和改进。其中最显著的变化是引入了新的C++接口，以及增加了更多的机器学习算法和深度学习模块。
4. **OpenCV 4.x 版本（2018年）**：OpenCV 4.x版本是当前最新的稳定版本。该版本引入了一些新的功能和改进，包括DNN模块的改进、CUDA加速优化、支持ONNX、边缘计算等。
5. **OpenCV 5.x 版本（2022年）**：OpenCV 的未来版本OpenCV 5.x强化AI部署能力，继续提供更多的功能增强和改进。这包括更好的深度学习集成、更快的图像处理和计算、更多的硬件支持等。

计算机视觉库: OpenCV

三、图像处理

(1) 图像读取 cv2.imread (filename, flags)

- filepath：读入image的完整路径
  - ✓ 相对路径: 设置在当前文件夹下　　cv2.imread (“ image.jpg ")
  - ✓ 绝对路径: 图片不保存在当前文件夹下

    cv2.imread (“image / image.jpg”)　或　cv2.imread (“image \\ image.jpg”)
- flags：标志位
  - ✓ cv2.IMREAD_COLOR：默认参数，读入一副彩色图片，可用1作为实参替代
  - ✓ cv2.IMREAD_GRAYSCALE：读入灰度图片，可用0作为实参替代
  - ✓ cv2.IMREAD_UNCHANGED：读入完整图片，包括alpha通道，可用-1作为实参替代

计算机视觉库: OpenCV

三、图像处理

（2）图像显示 cv2.imshow (window_name, image)

- window_name: 一个字符串，代表要在其中显示图像的窗口的名称。
- image: 要显示的图像，可以是彩色图像或灰度图像。

（3）图像保存 cv2.imwrite (filename, image)

- filename：要保存的文件的路径和名称，包括文件扩展名。
- image：要保存的 OpenCV 图像，通常是一个多维数组。

计算机视觉库: OpenCV

三、图像处理

(4) 图像属性 img.shape

- 图像的属性主要包括图像的**宽度、高度、通道。**

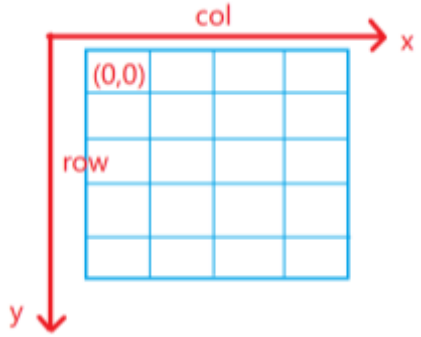
image. shape = (512, 512, 3)

计算机视觉库: OpenCV

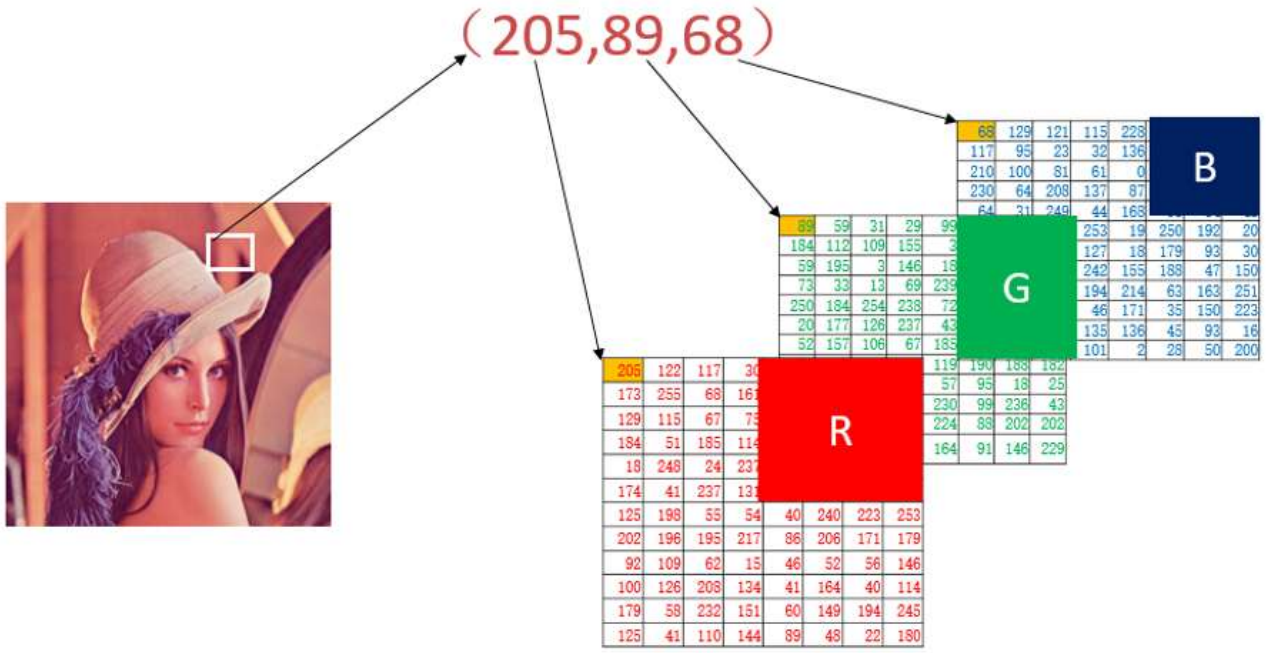
三、图像处理

(5) 图像的数据结构

- 图像的像素值通常是一个多维的Numpy数组。




```
[[[248 250 251]
  [247 249 250]
  [247 249 250]
  ...
  [218 222 227]
  [220 224 229]
  [220 224 229]]]
```



计算机视觉库: OpenCV

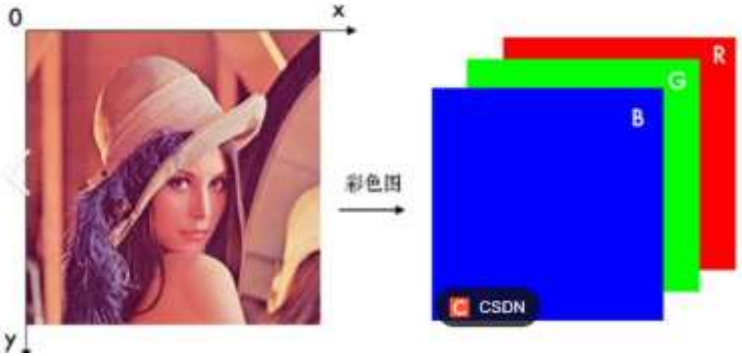
三、图像基础

（6）访问和操作图像的像素值

- 使用**索引访问单个像素**，并使用分配访问修改单个像素值。

  ✓ 例，将[200,150]位置处的像素值修改为红色

  Img [200,150] = (255 ,0, 0)



- 使用**切片访问一系列像素**，并使用分配访问修改这些像素值。

  ✓ 例，将[100:300,150:350]系列处的像素值修改为红色

  Img[100:300, 150:350] = (255 ,0, 0)

计算机视觉库: OpenCV

三、图像基础

（7）图像缩放　cv2. resize()

- 当图像过大时，为了更快地处理图像，往往需要对图像进行缩放。

方式1：固定大小缩放　cv2.resize（src，dsize）　`cv.resize(img, (400, 300))`

- src：原图像
- dsize：缩放后的图像大小

方式2：比例缩放　cv2.resize（src，None, fx，fy）　`cv.resize(img, None, fx=2,fy=2)`

- src：原图像
- dsize：None
- fx，fy ：缩放比例

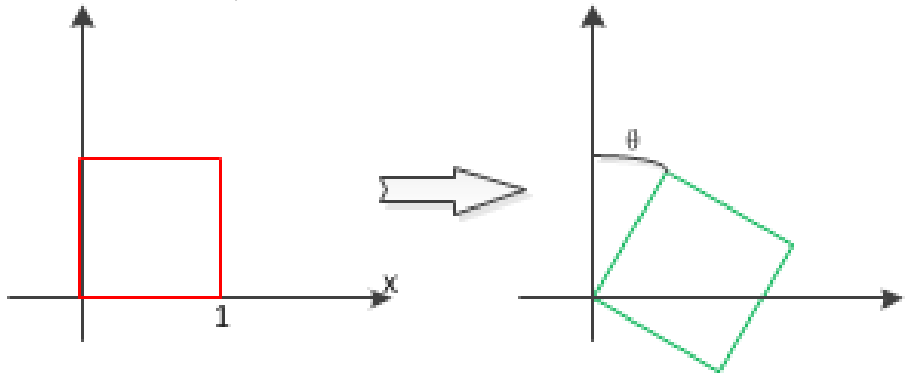
计算机视觉库: OpenCV

三、图像基础

(8) 图像旋转

- **getRotationMatrix2D(center, angle, scale)** 创建图像绕某一点的旋转矩阵
  - center:图像的旋转中心:
  - angle: 旋转角度， angle正的，图像逆时针旋转；反之，顺时针旋转
  - scale : 缩放比例，根据提供的值将图像向上或向下缩放

- **warpAffine(**src, M, dsize**)函数：** 将旋转矩阵应用到图像上,仿射变换
  - src : 原图像
  - M : 变换矩阵
  - dsize： 输出图像的大小



计算机视觉库: OpenCV

三、图像基础

(9) 图像裁剪 → 图像 ROI（Region of Interest）感兴趣区域

- 通过数组剪片操作，进行图像裁剪。

```
img_1 = img[0:150,0:150]
```

- 原图

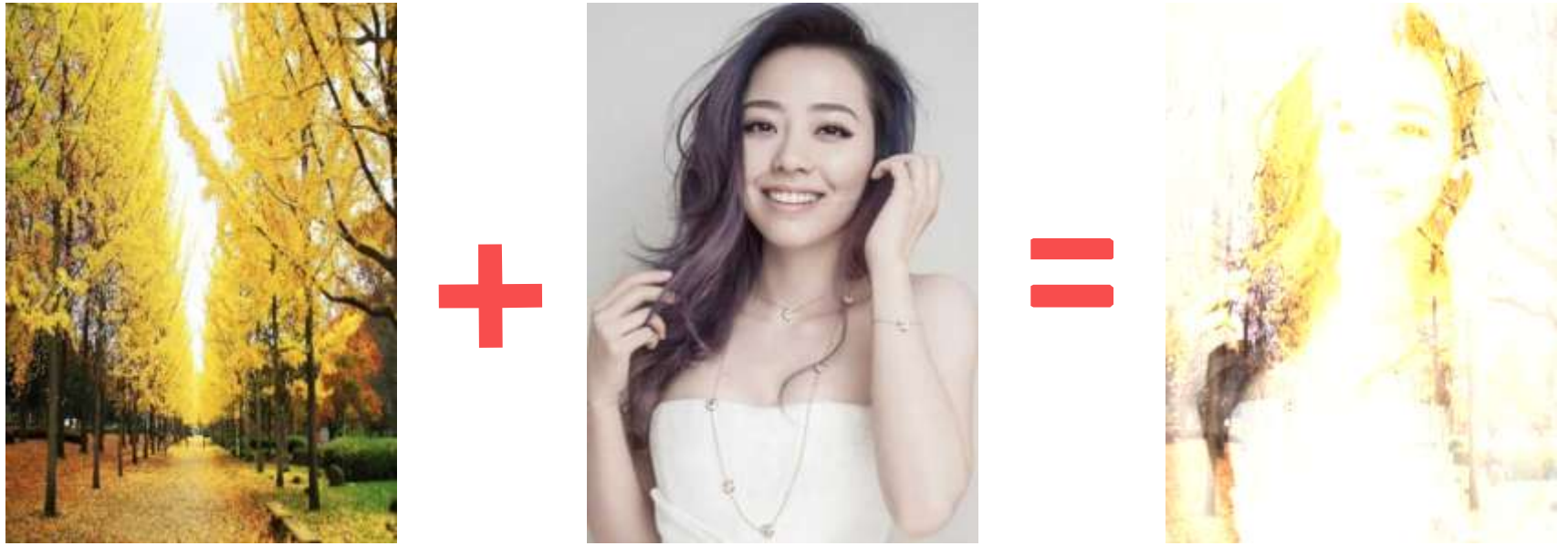
- 裁剪图

计算机视觉库: OpenCV

三、图像基础

（10）图像叠加和融合

- 图像叠加，cv2.add（src1, src2）或 使用numpy中的矩阵加法，但无限制。
  - cv2.add将两个图片进行加和，大于255的使用255计数；**合成的两个图片 大小必须一致**



计算机视觉库: OpenCV

三、图像基础

（10）图像叠加和融合

- 图像融合，cv2.addWeighted（src1, alpha, src2, beta, gamma）
  - src1, src2：需要融合相加的两副大小和通道数相等的图像
  - alpha：src1的权重
  - beta： src2的权重
  - gamma：gamma修正系数，不需要修正设置为0

计算机视觉库: OpenCV

三、图像基础

（10）图像叠加和融合

- 图像无缝融合，cv2. seamlessClone（src, dst, mask, center, flags）
  - src 目标影像，在本次给出的示例中是飞机。
  - dst 背景图像，在本次示例中是天空。
  - mask 目标影像上的mask，表示目标影像上那些区域是感兴趣区域。如果只对飞机感兴趣，那么mask上就只有飞机所在的区域。
  - center 目标影像的中心在背景图像上的坐标！注意是目标影像的中心！
  - flags 选择融合的方式，目前有cv.NORMAL_CLONE、 cv. MIXED_CLONE和cv. MONOCHROME_TRANSFER三种方法。

    -NORMAL_CLONE: 不保留dst 图像的texture细节。目标区域的梯度只由源图像决定。

    -MIXED_CLONE: 保留dest图像的texture 细节。目标区域的梯度是由原图像和目的图像的组合计算出来(计算dominat gradient)。

    -MONOCHROME_TRANSFER: 不保留src图像的颜色细节，只有src图像的质地，颜色和目标图像一样，可以用来进行皮肤质地填充。

计算机视觉库: OpenCV

三、图像基础

(10) 图像叠加和融合

- 图像无缝融合，cv2. seamlessClone（src, dst, mask, center, flags）
  - cv.NORMAL_CLONE

计算机视觉库: OpenCV

四、图像处理

（1）灰度图和二值图

- **彩色图（Color Image）**：包含丰富的颜色信息，通常由由红（Red）、绿（Green）、蓝（Blue）三个通道叠加而成，每个通道的取值范围通常是0~255。
- **灰度图（Grayscale Image）**：单通道图像，每个像素的值表示亮度（灰度级，0-255）仅包含亮度信息，没有颜色信息，表现出丰富的灰度层次，保留亮度信息。

- **二值图（Binary Image）**：，黑白图，是灰度图的特例，只有 0（黑）和 255（白）两种像素值用于表示“前景”和“背景”，分割前景/背景。

计算机视觉库: OpenCV

四、图像处理

（1）灰度图和二值图

■ **颜色空间转换**，cv2. cvtColor（src, code）

- src：原图像。
- code ：色彩空间转换代码，常见的类型有

① COLOR_BGR2GRAY： BGR 转换成灰度图
② COLOR_GRAY2BGR： 灰度图转换成BGR
③ COLOR_BGR2HSV： BGR 转换成HSV
④ COLOR_BGR2RGB： BGR 转换成RGB

色彩空间转换代码code 实际上可以调用的参数多达247种

灰度图

BGR

RGB

HSV

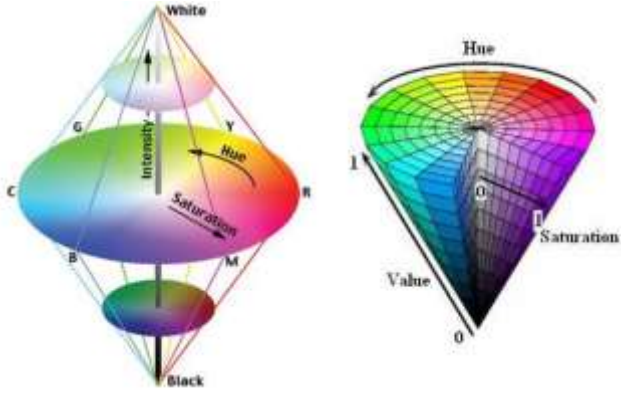
计算机视觉库: OpenCV

四、图像处理

（1）灰度图和二值图

- ✓ BGR vs RGB：**BGR是指颜色通道的顺序为蓝色、绿色、红色，而RGB则是红色、绿色、蓝色。**这两种格式在颜色混合时能产生相同的颜色范围，但在具体的图像处理和显示中，顺序会导致不同的结果。**OpenCV库默认使用BGR格式来存储图像数据**，这是由于早期硬件设备的兼容性和人类视觉系统的感知顺序所决定的。

- ✓ HSV：根据颜色的直观特性创建的一种颜色空间，**H：色调（Hue，也称为色相）； S：饱和度（Saturation）；V：亮度（Value）。**



	黑色	灰色	白色	红色	红色	橙色	黄色	绿色	青色	蓝色	紫色
Hmin	0	0	0	0	156	11	26	35	78	100	125
Hmax	180	180	180	10	180	25	34	77	99	124	155
Smin	0	0	0	43	43	43	43	43	43	43	43
Smax	255	43	30	255	255	255	255	255	255	255	255
Vmin	0	46	221	46	46	46	46	46	46	46	46
Vmax	46	220	255	255	255	255	255	255	255	255	255

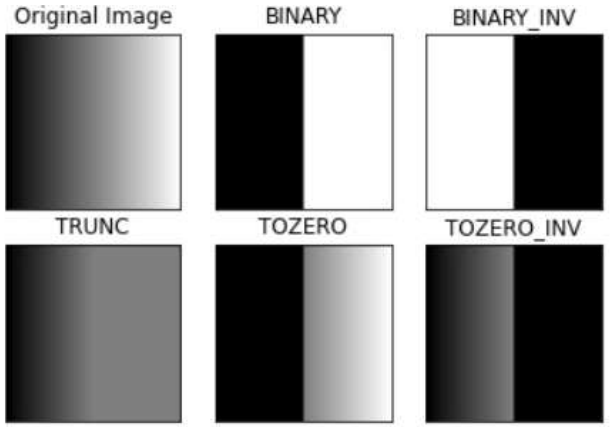
计算机视觉库: OpenCV

四、图像处理

- retval:　实际使用的阈值。
- dst:　处理后的图像。

(1) 灰度图和二值图

■ **图像二值化处理**，retval, dst = cv2.threshold (src, thresh, maxval, type)



- src: 输入图像，通常为灰度图像。
- thresh: 设定的阈值。
- maxval: 当像素值超过（或小于，根据类型）阈值时，赋予的新值。
- type: 阈值处理的类型，常见的类型有：

① cv2.THRESH_BINARY: 如果像素值大于阈值，则赋予 maxval，否则赋予 0。

② cv2.THRESH_BINARY_INV: 与 cv2.THRESH_BINARY 相反，如果像素值大于阈值，则赋予 0，否则赋予 maxval。

③ cv2.THRESH_TRUNC: 如果像素值大于阈值，则赋予阈值，否则保持不变。

④ cv2.THRESH_TOZERO: 如果像素值大于阈值，则保持不变，否则赋予 0。

⑤ cv2.THRESH_TOZERO_INV: 与 cv2.THRESH_TOZERO 相反，如果像素值大于阈值，则赋予 0，否则保持不变。

计算机视觉库: OpenCV

四、图像处理

（1）灰度图和二值图

■ **图像Otsu's 二值化**，retval, dst = cv2.threshold（src，thresh，maxval，type)

Otsu's 二值化是一种自动确定阈值的方法。它通过最大化类间方差来找到最佳的全局阈值，适用于双峰图像（即图像直方图有两个明显的峰值）

- src: 输入图像，通常为灰度图像。
- thresh: 由于 Otsu's 方法会自动确定阈值，因此该参数通常设置为 0。
- maxval: 当像素值超过（或小于，根据类型）阈值时，赋予的新值。
- type: 阈值处理的类型，通常为 cv2.THRESH_BINARY + cv2.THRESH_OTSU 或 cv2.THRESH_BINARY_INV+ cv2.THRESH_OTSU

灰度图

二值图

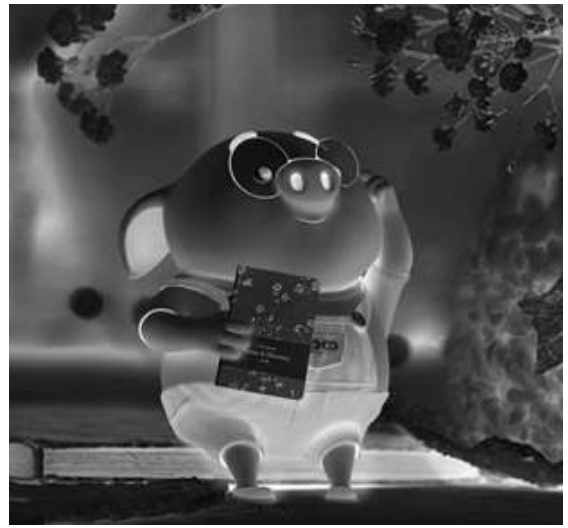
计算机视觉库: OpenCV

四、图像处理

(2) 图像增强

- **图像反转**：最常见的**灰度线性变换**，它通过**反转图像的灰度值**，使得原本亮的部分变暗，暗的部分变亮，类似于照片的“负片”效果。**增强暗区细节和 医学影像分析**

$$S = L - 1 - \mathrm{r}$$



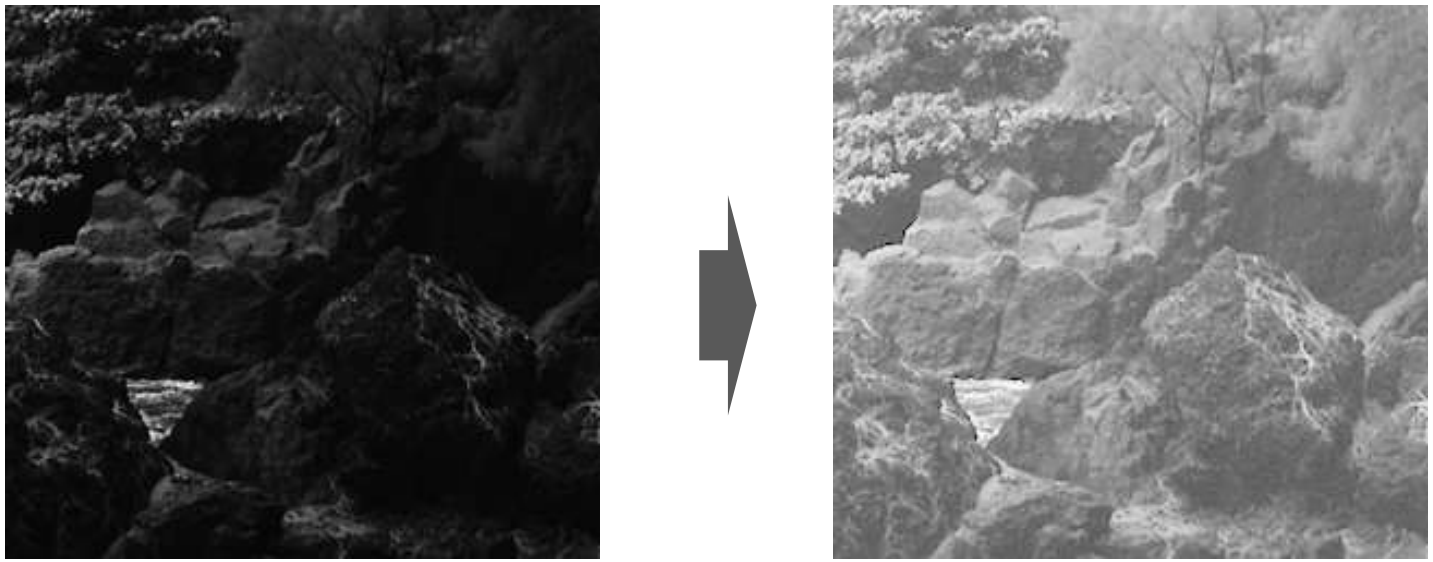
计算机视觉库: OpenCV

四、图像处理

(2) 图像增强

- **Log（对数）变换：**输入图像灰度值进行的非线性操作，**对数变换提升低亮区域，压缩高亮区域，使低亮区域的特征更加突出明显。**

$$S = c * log(1 + x)$$

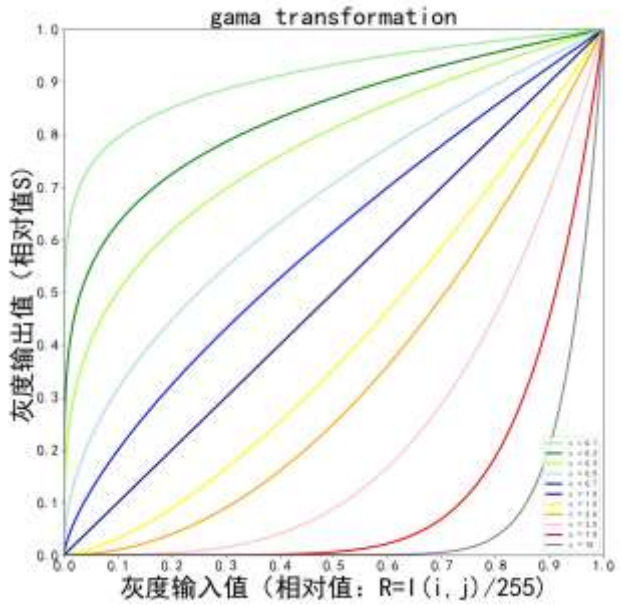


计算机视觉库: OpenCV

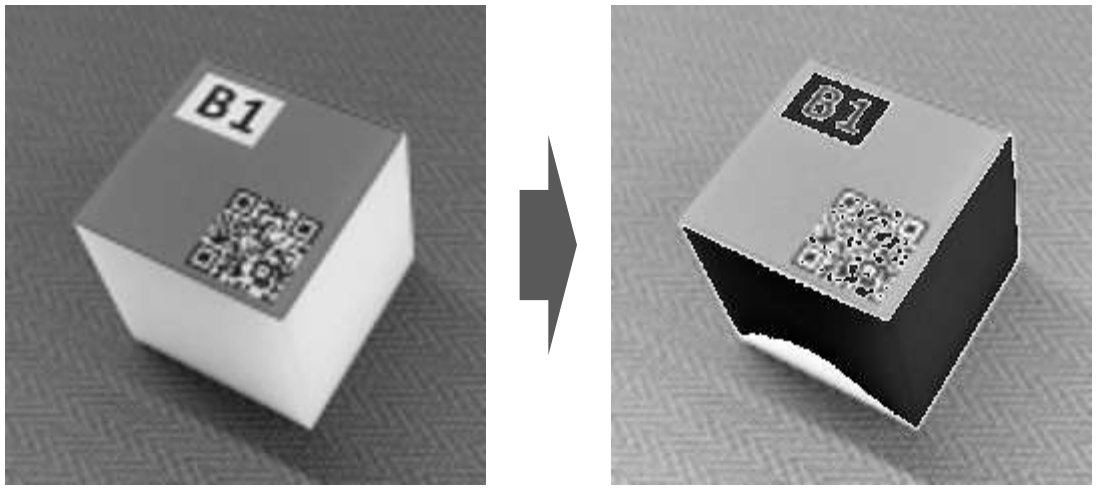
四、图像处理

(2) 图像增强

- **gamma（伽玛）变换：指数变换或幂次变换，提升图像的暗部细节**，让图像从曝光强度的线性响应变得更接近人眼感受的响应，即将漂白（相机曝光）或过暗（曝光不足）的图片进行矫正



$$S = c * x^{\gamma}$$



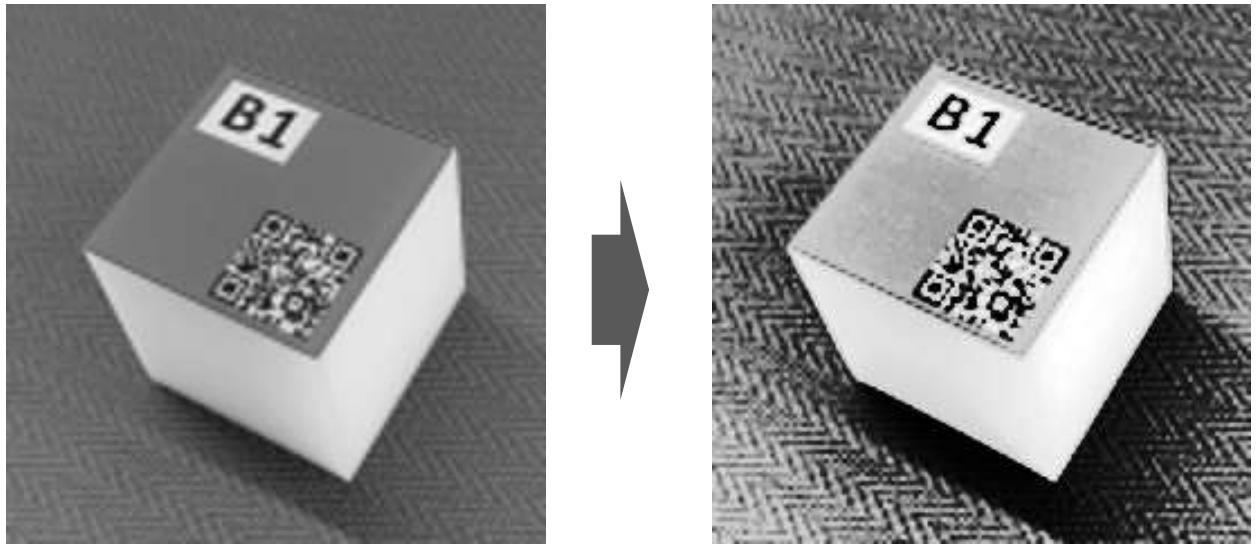
计算机视觉库: OpenCV

四、图像处理

(2) 图像增强

- **直方图变换（Histogram）：通过调整像素的灰度分布来改善图像的对比度、亮度或整体视觉效果。**让图像从曝光强度的线性响应变得更接近人眼感受的响应，即将漂白（相机曝光）或过暗（曝光不足）的图片进行矫正

- **直方图均衡化**： cv2.equalizeHist()　增强图像的对比度，使灰度分布更均匀



计算机视觉库: OpenCV

四、图像处理

（2）图像增强

- **直方图计算函数**： cv2.calcHist （images, channels, mask, histSize, ranges）

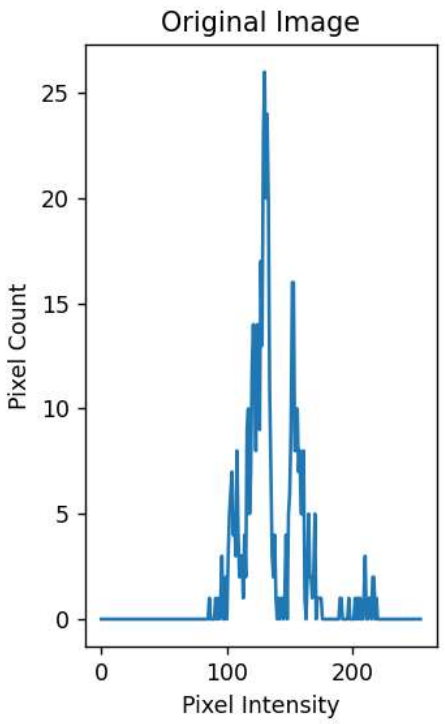
  - images: 输入的图像列表，通常是一个包含单通道或多通道图像的列表。例如 [img]。
  - channels: 需要计算直方图的通道索引。对于灰度图像，使用 [0]；对于彩色图像，可以使用 [0]、[1]、[2] 分别计算蓝色、绿色和红色通道的直方图。
  - mask: 掩码图像。如果指定了掩码，则只计算掩码区域内的像素。如果不需要掩码，可以传入 None。
  - histSize: 直方图的 bin 数量。对于灰度图像，通常设置为 [256]，表示将灰度级分为 256 个 bin。
  - ranges: 像素值的范围。对于灰度图像，通常设置为 [0, 256]，表示像素值的范围是 0 到 255。

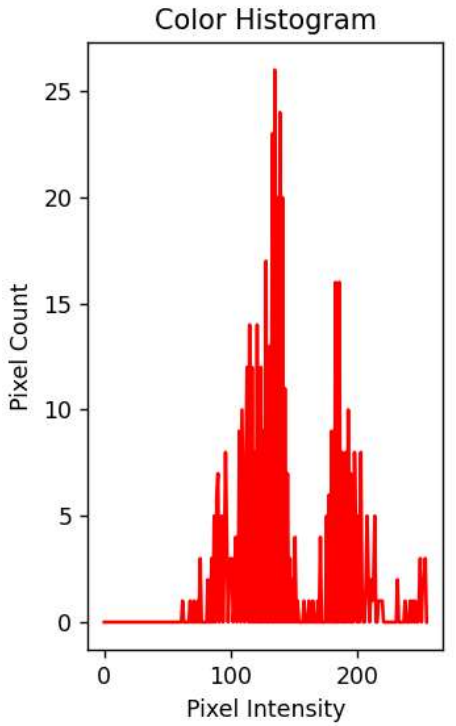
计算机视觉库: OpenCV

四、图像处理

(2) 图像增强

- **直方图均衡化 + 直方图计算**





计算机视觉库: OpenCV

四、图像处理

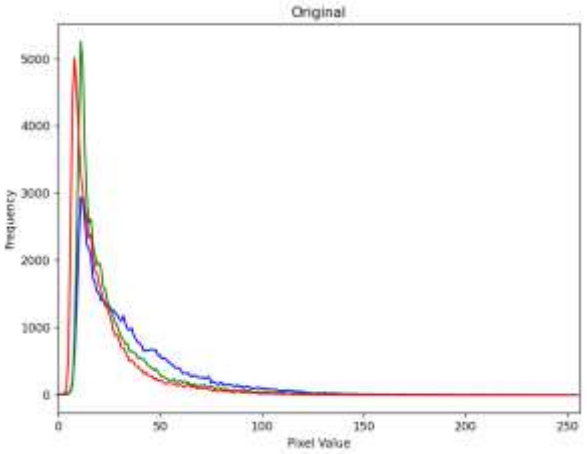
■ 彩色图片：直方图均衡化

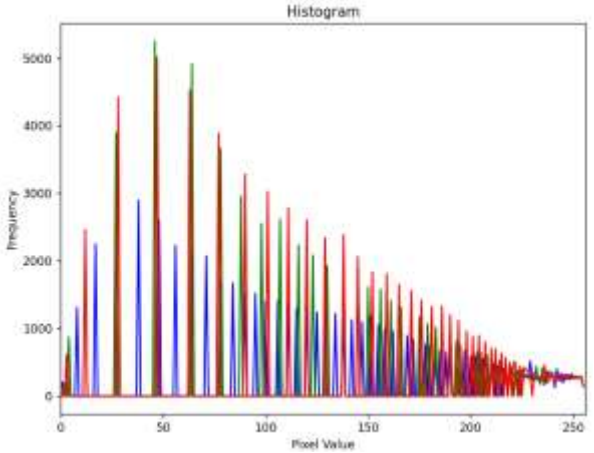
- **图片拆分**

  b, g, r = cv2.split (image)

- **图片合并**

  img= cv2.merge ([b, g, r])





计算机视觉库: OpenCV

四、图像处理

(3) 图像平滑处理

■ **平滑处理：**（也称为模糊处理）是一种常见的操作，用于**减少图像中的噪声或细节。常用的平滑处理技术：均值滤波、高斯滤波、中值滤波**

(a) 均值滤波：原理是将图像中每个像素的值替换为其**周围像素的均值，均值滤波可以有效地去除噪声，但可能会导致图像变得模糊**

dst = cv2. blur（src，ksize）

- src：是需要处理的图像，即源图像。
- ksize：是滤波核的大小，滤波核大小是指在均值处理过程中，其邻域图像的高度和宽度(n，m)，如（3，3），（3，5）

58	22	55	22	60	168	162	232
17	66	33	77	68	14	74	67
22	97	95	94	25	14	5	76
66	93	78	90	171	82	78	65
99	66	91	101	200	192	59	74
88	88	45	36	119	47	28	5
158	3	88	69	211	234	192	120
148	25	45	77	173	226	146	213

计算机视觉库: OpenCV

四、图像处理

（3）图像平滑处理

（b）高斯滤波：是一种基于高斯函数的平滑处理方法，高斯滤波在计算像素平均值时，会给中心像素赋予更高的权重，而给边缘像素赋予较低的权重。**高斯滤波适用于去除图像中的高斯噪声，并且在保留图像边缘信息方面表现较好。**

dst = cv2.GaussianBlur（ src，ksize，sigmaX）

- src ： 是需要处理的图像，即源图像。
- ksize：高斯核大小，ksize.width 并且 ksize.height 可以有所不同，但它们都必须是正数和奇数，如（3，3）
- sigmaX ： 高斯核的标准差，如果为0，则根据核大小自动计算

计算机视觉库: OpenCV

四、图像处理

（3）图像平滑处理

（c）中值滤波：中值滤波是一种非线性平滑处理方法。它的原理是将图像中每个像素的值替换为其周围像素的**中值**。中值滤波在**去除椒盐噪声（即图像中随机出现的黑白点）时非常有效**。

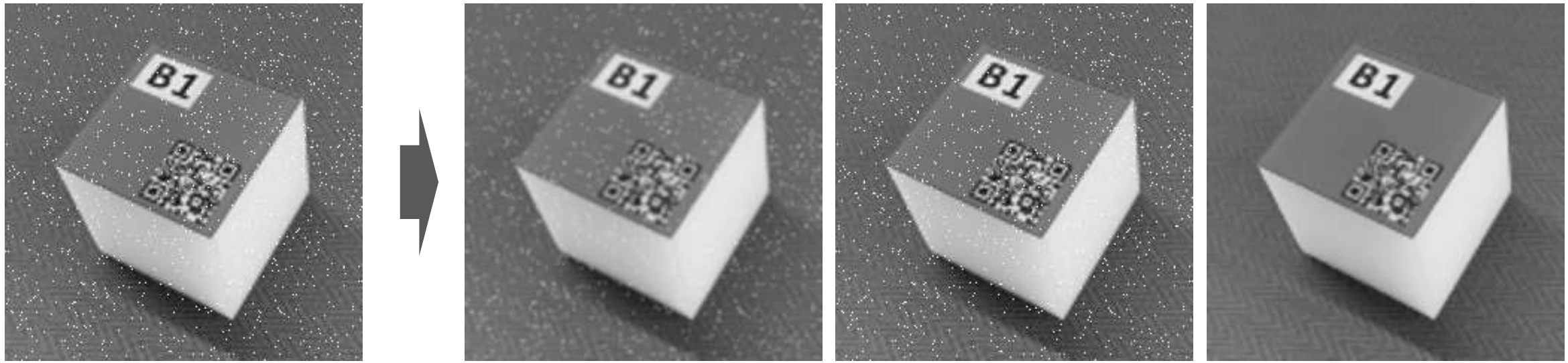
dst = cv2.medianBlur(src，ksize)

- src ： 是需要处理的图像，即源图像。
- ksize： 是滤波核的大小，滤波核大小是指在均值处理过程中，其邻域图像的高度和宽度，奇数，如3，5，7

计算机视觉库: OpenCV

四、图像处理

（3）图像平滑处理



原图+噪声　　均值滤波　vs　高斯滤波　vs　中值滤波

计算机视觉库: OpenCV

四、图像处理

- 练习：图像去噪

原图　　均值滤波　vs　高斯滤波　vs　中值滤波

计算机视觉库: OpenCV

四、图像处理

(4) 图像形态学操作

■ **图像形态学操作**是一种处理图像的形状特征的技术，主要用于**处理二值图像（即黑白图像）**，通过**腐蚀、膨胀、开运算、闭运算和形态学梯度**等操作，可以实现对图像的**噪声去除、对象分离、边缘检测**等效果。

操作	函数	说明	应用场景
腐蚀	cv2.erode()	用结构元素扫描图像，如果结构元素覆盖的区域**全是前景**，则保留中心像素。	去除噪声、分离物体。
膨胀	cv2.dilate()	用结构元素扫描图像，如果结构元素覆盖的区域**存在前景**，则保留中心像素。	连接断裂的物体、填充空洞。
开运算	cv2.morphologyEx()	先腐蚀后膨胀。	去除小物体、平滑物体边界。
闭运算	cv2.morphologyEx()	先膨胀后腐蚀。	填充小孔洞、连接邻近物体。
形态学梯度	cv2.morphologyEx()	膨胀图减去腐蚀图。	提取物体边缘。

计算机视觉库: OpenCV

四、图像处理

(4) 图像形态学操作

(a) 腐蚀：作用是**消除小且无意义的物体**；原理是内核在图像中滑动，只有当内核下的所有像素都为1时，原始图像中的像素(要么为1，要么为0)才会被认为是1，否则会被侵蚀(变成0)。

img = cv2.erode(src, kernel, iterations=1)

- src: 输入图像，通常是**二值图像。**
- kernel: 结构元素，可以自定义（np.ones((5,5), np.uint8）或使用 cv2.getStructuringElement() 生成。
- iterations: 腐蚀操作的次数，默认为1。

■ 卷积核定义： cv2.getStructuringElement(a,b,c)

- a设定卷积核的形状，。 MORPH_RECT(矩形卷积核)； MORPH_CROSS(十字形卷积核)；MORPH_ELLIPSE(椭圆形卷积核)
- b设定卷积核的大小（x，y）
- c表示描点的位置，一般 c = 1，表示描点位于中心。

计算机视觉库: OpenCV

四、图像处理

（4）图像形态学操作

（b ）膨胀：膨胀操作正好与腐蚀相反；原理是内核在图像中滑动，只有当内核下的存在像素都为1时，原始图像中的像素(要么为1，要么为0)才会被认为是1，否则会被侵蚀(变成0)。

img = cv2.dilate (src, kernel, iterations=1)

- src: 输入图像，通常是**二值图像。**
- kernel: 结构元素，可以自定义（np.ones((5,5), np.uint8）或使用 cv2.getStructuringElement() 生成。
- iterations:膨胀操作的次数，默认为1。

原图 **腐蚀操作** **膨胀操作**

计算机视觉库: OpenCV

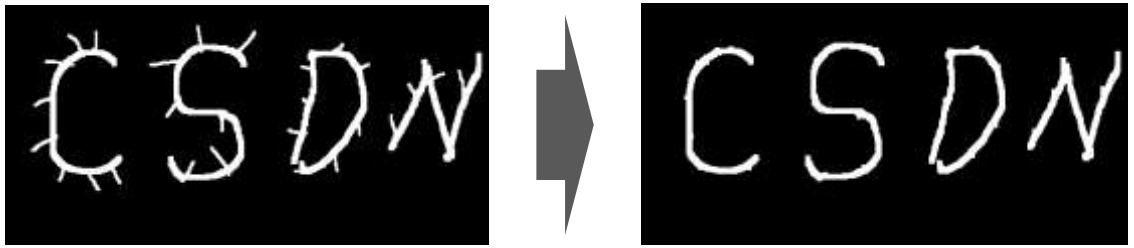
四、图像处理

（4）图像形态学操作

（c）开运算：是先腐蚀后膨胀的组合操作。开运算主要用于去除图像中的小噪声或分离连接的对象。

```
img = cv2.morphologyEx(src, op, kernel)
```

- src: 输入图像，通常是**二值图像。**
- op: 形态学操作类型，开运算使用 cv2.MORPH_OPEN。
- kernel: 结构元素，可以自定义（np.ones((5,5), np.uint8）或使用 cv2.getStructuringElement() 生成。



原图 **开运算**

计算机视觉库: OpenCV

四、图像处理

（4）图像形态学操作

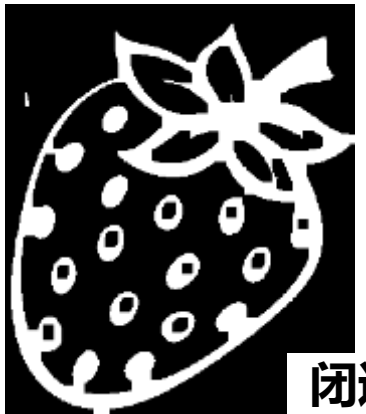
(d) 闭运算：是先膨胀后腐蚀的组合操作。闭运算主要用于**填补前景对象中的小孔或连接断裂的对象。**

```
img = cv2.morphologyEx(src, op, kernel)
```

- src: 输入图像，通常是**二值图像。**
- op: 形态学操作类型，闭运算使用 cv2.MORPH_CLOSE。
- kernel: 结构元素，可以自定义（np.ones((5,5), np.uint8）或使用 cv2.getStructuringElement() 生成。

原图

开运算



闭运算

计算机视觉库: OpenCV

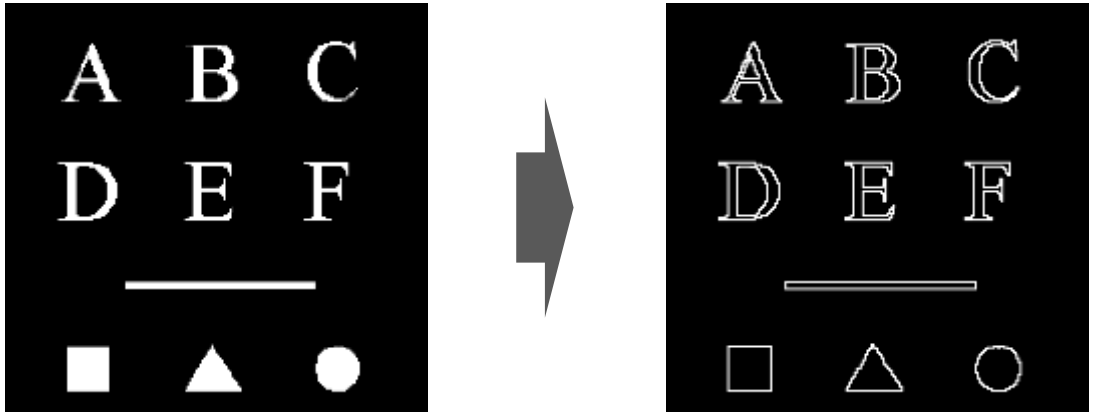
四、图像处理

（4）图像形态学操作

（e）形态学梯度：是**膨胀图像与腐蚀图像的差值**。形态学梯度主要用于**提取图像中前景对象的边缘**。

```
img = cv2.morphologyEx(src, op, kernel)
```

- src: 输入图像，通常是**二值图像。**
- op: 形态学操作类型，闭运算使用 cv2.MORPH_GRADIENT。
- kernel: 结构元素，可以自定义（np.ones((5,5), np.uint8）或使用 cv2.getStructuringElement() 生成。



计算机视觉库: OpenCV

四、图像处理

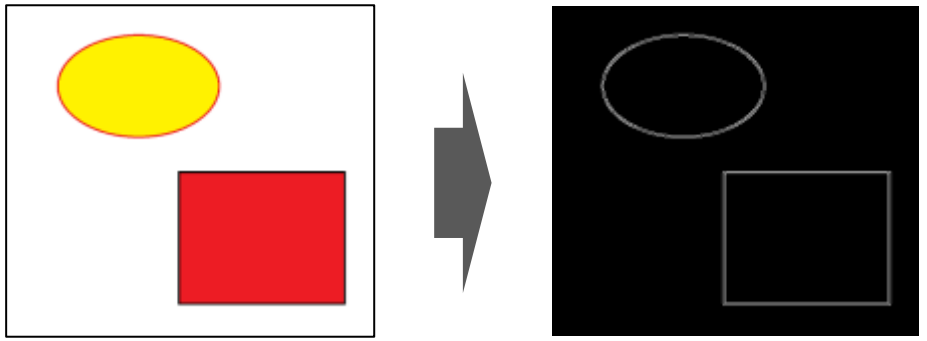
(5) 图像边检检测

■ **图像边缘检测：**用于识别图像中亮度变化明显的区域，这些区域通常对应于物体的边界。

Canny 边缘检测： edges = cv2.Canny(image，threshold1，threshold2， apertureSize=3，L2gradient=False)

- image：输入图像，必须是单通道的灰度图像。
- threshold1：低阈值。
- threshold2：高阈值。调整阈值可改变边缘检测的灵敏度，高阈值 ≈ 2-3倍低阈值
- apertureSize：Sobel 算子的孔径大小，默认为 3。
- L2gradient：是否使用 L2 范数计算梯度幅值，默认为 False（使用 L1 范数）。

输出：图像是二值图（边缘为白色，背景为黑色）



计算机视觉库: OpenCV

四、图像处理

(6) 图像轮廓检测

■ **轮廓检测是**图像处理中的重要任务，**用于提取图像中物体的边界。**

- **轮廓:** 图像中物体的边界，由一系列点组成。
- **轮廓层次结构:** 轮廓之间的嵌套关系，例如一个轮廓是否包含另一个轮廓。
- **轮廓特征:** 轮廓的面积、周长、边界矩形、最小外接矩形、最小外接圆等。

函数名称	功能描述
cv2.findContours()	查找图像中的轮廓
cv2.drawContours()	在图像上绘制轮廓
cv2.contourArea()	计算轮廓的面积
cv2.arcLength()	计算轮廓的周长或弧长
cv2.boundingRect()	计算轮廓的边界矩形

计算机视觉库: OpenCV

四、图像处理

(6) 图像轮廓检测

- contours: 输出的轮廓列表，每个轮廓是一个点集。
- hierarchy: 输出的层次结构信息。

(a) 轮廓查找：contours，hierarchy = cv2. findContours (image，mode，method)

- image: 输入的二值图像（通常为经过**阈值处理**或**边缘检测后的图像，要求背景为黑色，对象为白色**）。
- mode: **轮廓检索模式**，常用的有：

  cv2.RETR_EXTERNAL: 只检测最外层轮廓。

  cv2.RETR_LIST: 检测所有轮廓，但不建立层次关系。

  cv2.RETR_TREE: 检测所有轮廓，并建立完整的层次结构。
- method: **轮廓近似方法**，常用的有：

  cv2.CHAIN_APPROX_NONE: 存储所有的轮廓点。

  cv2.CHAIN_APPROX_SIMPLE: 压缩水平、垂直和对角线段，只保留端点。

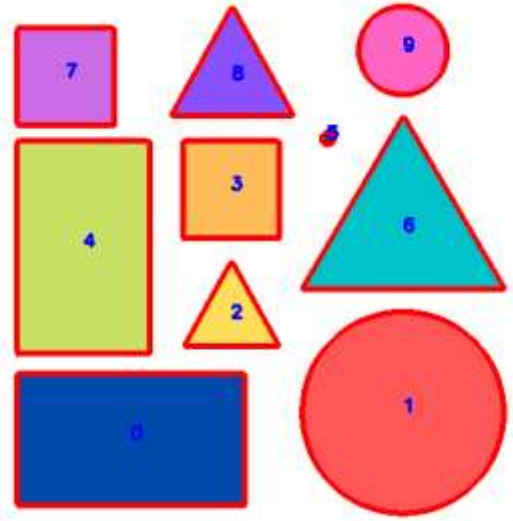
计算机视觉库: OpenCV

四、图像处理

(6) 图像轮廓检测

(b) 轮廓绘制： cv2.drawContours(image, contours, contourIdx, color, thickness)

- image: 要绘制轮廓的图像。
- contours: 轮廓列表。
- contourIdx: 要绘制的轮廓索引，如果为负数，则绘制所有轮廓。
- color: 轮廓的颜色。
- thickness: 轮廓线的厚度，如果为负数，则填充轮廓内部。



- **cv2.putText (img, text, org, fontFace, fontScale, color, thickness) : 图像上绘制文本**
  - **Img**：图片； **text**：要添加的文字；
  - **org**：文字添加到图片上的位置 (x,y);
  - **fontFace:** 字体的类 , **FONT_HERSHEY_SIMPLEX**：**正常大小无衬线字体**； **FONT_HERSHEY_PLAIN**：**小号无衬线字体**
  - **fontScale** :型字体大小; **color**：字体颜色; **thickness:**字体粗细

计算机视觉库: OpenCV

四、图像处理

(6) 图像轮廓检测

(c) 计算轮廓面积： area = cv2.contourArea(contour)

- contour: 输入的轮廓点集

(d) 计算轮廓周长或弧长： length = cv2.arcLength(curve, closed)

- curve: 输入的轮廓点集。
- closed: 布尔值，表示轮廓是否闭合。

(d) 计算轮廓的边界矩形： x, y, w, h = cv2.boundingRect(points)

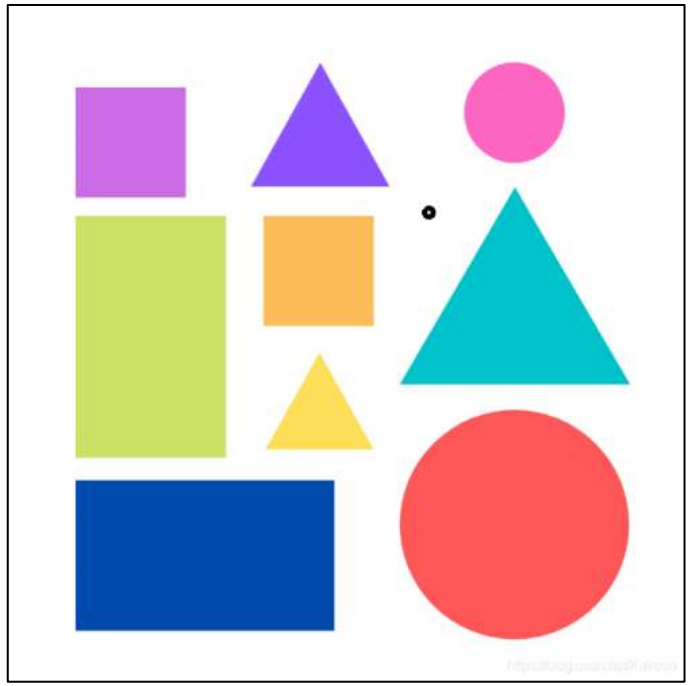
- points: 输入的轮廓点集。

```
第0个轮廓的面积是21308.500
第1个轮廓的面积是22658.500
第2个轮廓的面积是2879.000
第3个轮廓的面积是6639.000
第4个轮廓的面积是20041.000
第5个轮廓的面积是84.500
第6个轮廓的面积是12485.000
第7个轮廓的面积是6718.500
第8个轮廓的面积是4766.000
第9个轮廓的面积是4372.000
```

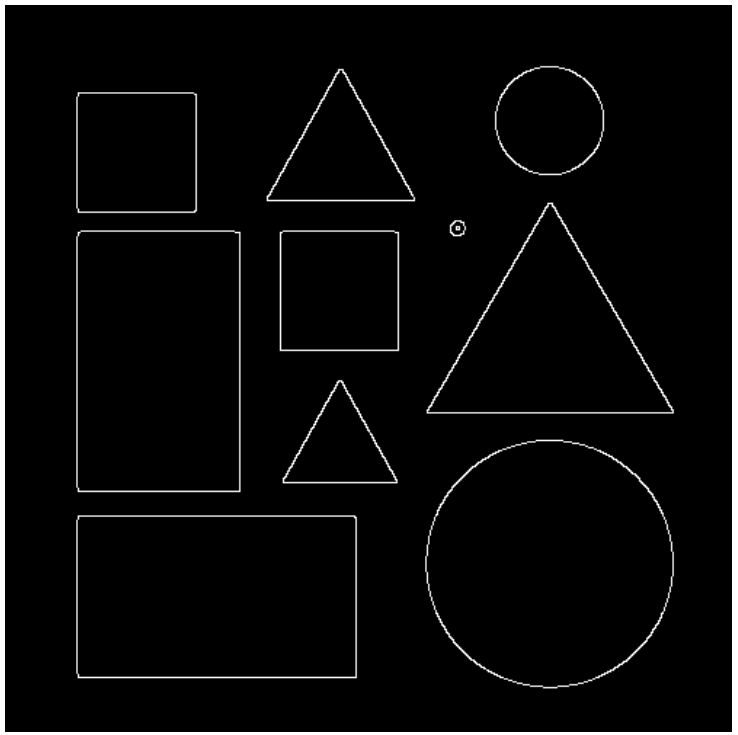
计算机视觉库: OpenCV

四、图像处理

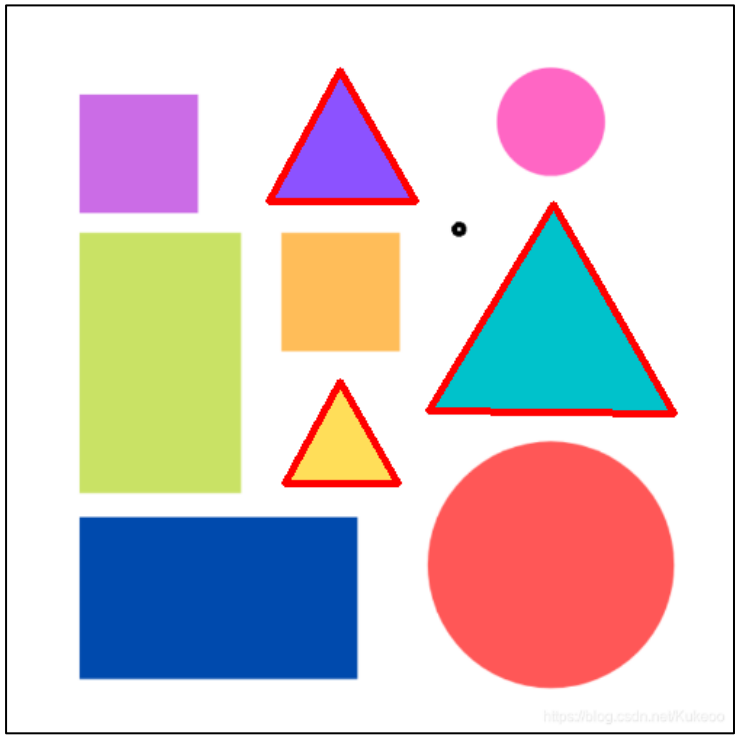
(6) 案例：检测图像形状



原图



边界检测



结果

计算机视觉库: OpenCV

四、图像处理

(7) 图像匹配

- **图像匹配：**指的是在目标图像中寻找与给定的模板图像（或参考图像）**最相似**的部分的过程。它的核心目标是确定**模板图像在目标图像中的位置、方向**，或者**判断目标图像中是否存在与模板相似的物体/场景**。

- **模板匹配**：模板图像在目标图像上进行滑动（逐像素或按步长滑动），并在每个位置计算模板与目标图像对应区域的某种相似性度量值。最终找到相似度值最大（或最小，取决于度量方法）的位置作为最佳匹配位置。**对尺度缩放、旋转变化极其敏感。** 如果模板和目标中的物体大小或方向不同，匹配会失败。

- **特征匹配**：核心思想是提取图像中具有显著性和鲁棒性的局部特征（关键点及其描述符），然后在两幅图像的特征之间建立对应关系。匹配结果通常能适应尺度缩放、旋转、光照变化、部分遮挡和视角变化。

计算机视觉库: OpenCV

四、图像处理

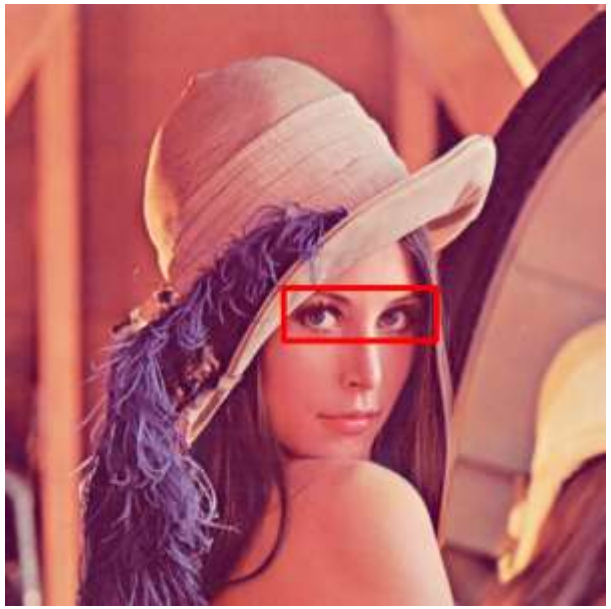
(7) 图像匹配

- 结果映射图（Result Map） 包括相似度、位置 、尺寸等信息

(a) 模板匹配： result = cv2.matchTemplate（image, template, method）

- image: 目标图像（在其中搜索），应为灰度图。
- template: 模板图像（要查找的图像），尺寸应小于 image，同样应为灰度图。
- method: 匹配方法（相似性度量准则）。

方法	说明	理想匹配值
cv2.TM_SQDIFF	平方差匹配，值越小匹配度越高。	最小值 (0)
cv2.TM_SQDIFF_NORMED	归一化平方差匹配，值越小匹配度越高。	最小值 (0)
cv2.TM_CCORR	相关匹配，值越大匹配度越高。	最大值
cv2.TM_CCORR_NORMED	归一化相关匹配，值越大匹配度越高。	最大值 (1)
cv2.TM_CCOEFF	相关系数匹配，值越大匹配度越高。	最大值
cv2.TM_CCOEFF_NORMED	归一化相关系数匹配，值越大匹配度越高。	最大值 (1)



计算机视觉库: OpenCV

四、图像处理

(7) 图像匹配

(b) SIFT 算法：是一种基于尺度空间的**特征点检测算法**，它对图像的**旋转、缩放、亮度变化具有不变性。**

■ 使用 **SIFT** 特征检查器和 **暴力** 匹配算法实现两张图像的特征匹配

- 使用 **sift=cv2.SIFT_create()** 方法初始化SIFT对象，设置默认值。
- 使用 **sift.detectAndCompute()** 方法分别在两张输入图像中检测和计算关键点以及描述符。
- 创建暴力匹配BFmatcher对象 **bf=cv2.BFMatcher()** 并使用其 **bf.match(des1,des2)** 方法匹配描述符。
- 使用 **cv2.drawMatches()** 方法在原始输入图像上绘制匹配结果。

计算机视觉库: OpenCV

四、图像处理

(7) 图像匹配

- **创建 BFMatcher 对象：bf=cv2.BFMatcher(normType，crossCheck = Flase)**
  - normType：距离类型，默认为欧式距离（cv2.NORM_L2）。其他选项包括曼哈顿距离（cv2.NORM_L1）和汉明距离（cv2.NORM_HAMMING）

- **匹配方法：match 和 knnMatch**
  - `matches = bf.match(des1, des2)`

    des1=模板图像，des2=目标图像，

  - `matches = bf.knnMatch(des1, des2, k=2)`

    k：每个查询关键点保留的最佳匹配数量，通常设置为 2

计算机视觉库: OpenCV

四、图像处理

(7) 图像匹配

■ **可视化两幅图像之间的特征匹配结果：** cv2.drawMatches

```
result = cv2.drawMatches(
    img1, keypoints1,
    img2, keypoints2,
    matches1to2,
    outImg,
    matchColor=None,
    singlePointColor=None,
    matchesMask=None,
    flags=None
)
```

- img1：模板图像。keypoints1：模板图像的关键点。
- img2：目标图像。keypoints2：目标图像的关键点。
- matches1to2：从第一幅图像到第二幅图像的匹配列表。
- outImg :输出图像，绘制匹配结果,如果为None，函数会创建一个新图像
- matchColor (可选):匹配线条的颜色（默认随机颜色）
- singlePointColor (可选):未匹配关键点的颜色（默认随机颜色）
- matchesMask (可选):掩码，决定绘制哪些匹配
- flags (可选)：绘图标志，控制绘制方式：

cv2.DRAW_MATCHES_FLAGS_DEFAULT (0)：默认方式
cv2.DRAW_MATCHES_FLAGS_DRAW_OVER_OUTIMG (1)：在现有图像上绘制
cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS (2)：不绘制未匹配点
cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS (4)：绘制带方向和大小的关键点

计算机视觉库: OpenCV

四、图像处理

(8) 图像分割

返回值:

- **compactness**：紧密度，返回每个点到相应重心的距离的平方和
- **labels**：结果标记，每个成员被标记为分组的序号，如 0,1,2,3,4.. K-1
- **centers**：由聚类的中心组成的数组

- **图像分割**：目的是将图像分割成若干个特定的、具有独特性质的区域。k-means算法将图像进行聚类分割，将相近的像素点按照指定的需要划分的类别数量进行分类。

- **k-means聚类**：retval, bestLabels, centers = cv2.kmeans(data, K, bestLabels, criteria, attempts, flags)

  - data: 数据，np.float32数据，每个特征放一列。
  - K:聚类中心数:
  - bestLabels: 通常设为 None
  - criteria：迭代停止的模式选择。格式为（**type, max_iter, epsilon**）=(**迭代终止条件,最大迭代次数,精度阈值**)， type模式包括
    ① cv2.TERM_CRITERIA_EPS ：精确度（误差）满足epsilon，则停止。
    ② cv2.TERM_CRITERIA_MAX_ITER：迭代次数超过max_iter，则停
    ③ cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER：两者结合，满足任意一个结束。
  - attempts：使用不同初始中心运行算法的次数，返回最佳结果
  - flags：初始中心选择，
    cv2.KMEANS_PP_CENTERS：使用kmeans++算法的中心初始化算法 ；cv2.KMEANS_RANDOM_CENTERS：每次随机选择初始中心

计算机视觉库: OpenCV

四、图像处理

(8) 案例： 图片背景颜色替换

原图

背景

前景

处理后