第 1 周：计算机视觉导论与图像基础

让机器“看懂”试卷的第一步

计算机视觉课程组

2024-2025 学年

视觉：人类获取信息的最主要渠道

人类视觉：

- 人脑约 50% 的神经元参与视觉处理。
- **语义理解**：我们看到的不是像素，是“人”、“车”、“试卷”。

计算机视觉 (CV)：

- 目标：给机器安装“眼睛”和“大脑”。
- 挑战：图像在计算机眼中只是一组**数字**。

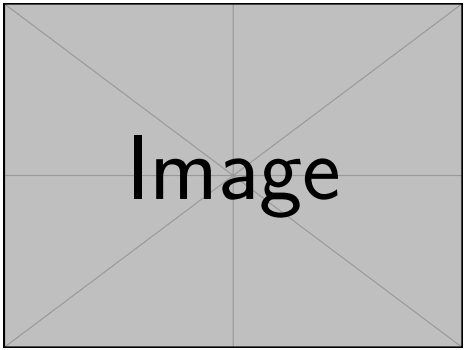


图: 语义 vs 矩阵

CV 的历史与现状

- **1960s**：Larry Roberts (CV 之父) 尝试让机器识别积木世界。
- **1970s-1980s**：提出边缘检测、Marr 视觉计算理论。
- **2012-至今**：**深度学习爆发**，AlexNet 在 ImageNet 竞赛中夺冠。

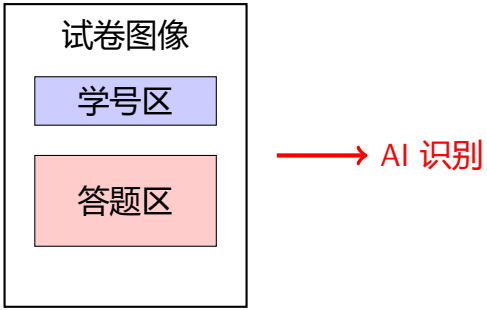
核心任务演变

分类 (是什么?) $\rightarrow$ 检测 (在哪儿?) $\rightarrow$ 分割 (形状如何?) $\rightarrow$ 生成 (画一个出来)

贯穿本学期的项目：AI 阅卷助手

任务分解：

1. **图像采集**：拍照、扫描。
2. **预处理**：纠偏、增强（本周内容）。
3. **定位**：找到答题卡、填空区。
4. **识别**：OCR (光学字符识别)。
5. **评分**：逻辑比对。



图像的底层本质：矩阵 (Matrix)

- 一张灰度图 = 一个 **二维矩阵**。
- 矩阵中的每个元素称为 **像素 (Pixel)**。
- 常用数据类型：`uint8` (0-255)。

$$\begin{bmatrix} 255 & 255 & 254 \\ 120 & 0 & 118 \\ 255 & 253 & 255 \end{bmatrix}$$

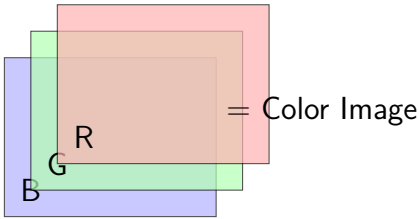
(矩阵数值 $\rightarrow$ 图像亮度)

注意坐标系！

计算机图像坐标系：**左上角为原点 (0,0)**，X 轴向右，Y 轴向 **下**。

彩色图像：RGB 三通道

- 彩色图像 = 三个二维矩阵堆叠 (**三维张量**)。
- 每个通道代表一种颜色光的强度。



OpenCV 的特殊性：默认读取顺序是 **BGR**，而非 RGB。

互动练习：调色盘

如果一个像素的 RGB 值为以下数值，它是什么颜色？

R	G	B	预测颜色
255	255	0	
0	255	255	
128	128	128	
0	0	0	

互动练习：调色盘

如果一个像素的 RGB 值为以下数值，它是什么颜色？

R	G	B	预测颜色
255	255	0	黄色
0	255	255	
128	128	128	
0	0	0	

互动练习：调色盘

如果一个像素的 RGB 值为以下数值，它是什么颜色？

R	G	B	预测颜色
255	255	0	黄色
0	255	255	青色/浅蓝
128	128	128	
0	0	0	

互动练习：调色盘

如果一个像素的 RGB 值为以下数值，它是什么颜色？

R	G	B	预测颜色
255	255	0	黄色
0	255	255	青色/浅蓝
128	128	128	灰色
0	0	0	

互动练习：调色盘

如果一个像素的 RGB 值为以下数值，它是什么颜色？

R	G	B	预测颜色
255	255	0	黄色
0	255	255	青色/浅蓝
128	128	128	灰色
0	0	0	**黑色**

OpenCV 核心函数详解

图像读取的隐患

```
import cv2

# 路径千万不能有中文（新手常见错误）
img = cv2.imread('paper.jpg')

# 检查是否读取成功
if img is None:
    print("错误：文件不存在或路径含有中文！")

# 打印维度 (H, W, C)
print(img.shape)
```

常用 Flag

Matplotlib 显示与 BGR 转换

为什么用 `plt.imshow` 显示出来的人脸是青蓝色的?

```
import matplotlib.pyplot as plt

# OpenCV 是 BGR, Matplotlib 是 RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)
plt.show()
```

思考: 灰度图显示时需要设置 `cmap='gray'`，否则会变成“原油色”。

滤镜 1：灰度化 (Grayscale)

为什么要灰度化?

- 减少计算量（数据量降至 $1/3$）。
- 识别试卷上的文字，颜色信息通常是不必要的。

原理： $Gray = R \times 0.299 + G \times 0.587 + B \times 0.114$
(为什么绿色权重最高？因为人眼对绿色最敏感。)

滤镜 2：反色 (Inversion)

原理： $NewValue = 255 - OldValue$

- 黑色 (0) $\rightarrow$ 白色 (255)
- 白色 (255) $\rightarrow$ 黑色 (0)

应用：增强暗背景下的试卷特征，或者扫描负片。

滤镜 3：亮度调整与“溢出”陷阱

错误做法： `img + 50`
如果像素值是 220，加 50 变成 270。而在 uint8 类型下，270 会变成 **14** (截断/绕回)，导致图像出现难看的噪点。

安全写法

```
# 使用 numpy 的 clip 函数限制范围
bright_img = np.clip(img.astype(np.int32) + 50, 0, 255).astype(np
    .uint8)

# 或者使用 OpenCV 内置函数（推荐，速度更快）
bright_img = cv2.add(img, np.array([50.0]))
```

课堂动手环节：初探试卷图像

1. 使用 `cv2.imread` 读取提供的 `exam.jpg`。
2. 打印该图像的 `shape`，计算总像素个数。
3. **进阶挑战**：尝试将图像中间 100×100 的区域涂成纯黑色（提示：使用切片 `img[y1:y2, x1:x2] = 0`）。

课后作业：我的第一个图像处理器

作业要求

编写一个 Python 脚本，读取一张照片并生成一张包含 4 张子图的对比图：

1. 原图
2. 灰度图
3. 亮度增强后的图
4. 反色后的图

提交方式：截图 + 代码。下周我们将学习如何让 AI 帮我们优化这段代码！

Q & A 准备好进入计算机视觉的世界了吗?