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

北京石油化工学院\人工智能研究院\王文通

北京石油化工学院
BEIJING INSTITUTE OF PETROCHEMICAL TECHNOLOGY

北京石油化工学院
人工智能研究院

2025-2026 学年

课程概览

本周内容：

- 计算机视觉导论
- 图像的数字表示
- OpenCV 基础
- 代码实战演练
- 工业硬件知识

学期项目：AI 阅卷助手

1. 图像采集与预处理
2. 答题卡定位（Timing Marks）
3. 填涂检测与识别
4. 手写文字 OCR
5. 成绩统计与输出

本周时间分配（160 分钟 = 3 学时）

第 1 学时（45 分钟）：

00:15 理论讲解：CV 导论（15min）

00:35 Live Coding：OpenCV 基础（20min）

00:45 讨论与答疑（10min）

第 2 学时（45 分钟）：

01:10 理论讲解：图像理论（25min）

01:30 实践：图像滤镜（20min）

第 3 学时（70 分钟）：

01:30-01:50 硬件知识（20min）

01:50-02:20 Live Coding：阅卷系统（30min）

02:20-02:35 互动测验（15min）

02:35-02:45 总结与作业（10min）

时间控制提示

如果进度落后，建议跳过”挑战任务”

本周分组策略

分组原则：

- 每 4 人为一组
- 确保不同专业背景混合
- 建议包含：理工科、文科、无编程基础、有编程基础

角色分工：

角色	职责	适合
组长	统筹协调、进度管理	组织能力强的
算法实现者	实现 OpenCV 代码、图像处理	有编程基础的
参数调优者	调整滤波参数、优化效果	细心负责的
测试者	收集测试用例、报告问题	细心负责的

本周协作任务

多屏协同设计

本课程采用多屏协同教学方式：

主屏（左侧）：理论讲解

- PPT 幻灯片
- 概念和原理讲解
- 图像和代码展示
- 互动测验

侧屏（右侧）：实时演示

- OpenCV 代码实时演示
- 图像处理效果展示
- 参数调整实时反馈
- 调试过程展示

移动设备互动

使用手机参与互动测验（问卷星）

课前预备知识

需要的预备知识：

- **NumPy 基础**
  - 数组索引与切片
  - 矩阵运算
- **Python 基础**
  - 函数定义
  - 循环与条件判断

预备视频链接：

- NumPy 基础（5 分钟）
  - `https://...`
- Python 基础（10 分钟）
  - `https://...`

重要提示

如果你从未接触过 Python，建议先看预备视频！

选择适合你的学习路径

根据你的基础和兴趣，选择适合的学习路径：

观察者路径

适合：无编程基础的学生

- 理解原理和概念
- 观看代码演示
- 完成基础任务
- 参与课堂讨论

”理解原理最重要”

使用者路径

适合：有一定编程基础

- 运行示例代码
- 调整参数观察效果
- 完成核心任务
- 用 AI 辅助学习

”动手实践出真知”

创造者路径

适合：编程基础较好

- 修改和优化算法
- 创新功能实现
- 完成挑战任务
- 帮助他人解决问题

”探索与创新”

你可以随时在不同路径间切换！

本课程鼓励互助学习，创造者路径的同学可以协助观察者路径的同学

AI 辅助学习：向 AI 提问的技巧

场景 1：理解 OpenCV 概念

Prompt 示例

请解释 OpenCV 中的 BGR 和 RGB 有什么区别，为什么 OpenCV 默认使用 BGR?

场景 2：调试 OpenCV 代码

Prompt 示例

我的 OpenCV 代码显示的图像颜色不对。
运行环境：Windows 11, Python 3.9.7, OpenCV 4.8.0
相关代码：`img = cv2.imread('test.jpg') plt.imshow(img)`
请帮我分析原因并提供解决方案。

场景 3：学习 OpenCV 函数

AI 编程工具推荐

本周虽然重点是学习 OpenCV，但强烈推荐了解 AI 编程工具：

工具	特点	适用
Cursor	AI 原生 IDE，Ctrl+K 原地编辑	**最推荐**
ChatGPT	对话能力强，代码生成准确	学习、调试
Claude	代码分析深入	代码审查
通义千问	中文友好，国内可用	中文问题
DeepSeek	编程能力强，开源	代码生成

重要

下周（Week 2）我们将系统学习这些 AI 工具的使用方法！

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**。

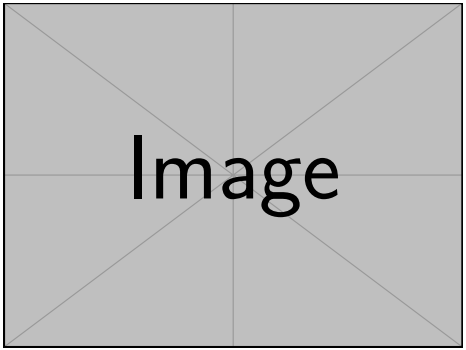


图: 语义 vs 矩阵（请替换为实际图片）

计算机视觉应用全景

自动驾驶

Tesla Autopilot、Waymo

- 车道检测
- 交通标志识别
- 行人检测

医疗影像

CT/MRI 诊断

- 肿瘤检测
- 器官分割
- 病理分析

工业检测

产品质检

- 缺陷识别
- 尺寸测量
- 质量控制

人脸识别

支付宝、安防系统

- 身份验证
- 门禁系统
- 犯罪侦查

OCR 文字识别

文档数字化

- 发票识别
- 车牌识别
- 手写转录

AR/VR

增强现实

- 虚拟试衣
- 游戏交互
- 远程协作

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。

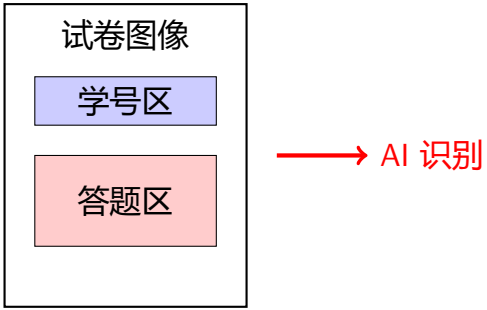
核心任务演变

分类 (是什么?) → 检测 (在哪儿?) → 分割 (形状如何?) → 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。



AI 阅卷系统的技术挑战

挑战 1：手写字迹识别难度

- 每个人的字迹不同
- 连笔、潦草、抖动
- 相似字符混淆（如：0 vs O, 1 vs l）

挑战 2：答题卡污渍处理

- 涂改痕迹
- 折痕污损
- 水渍污染

挑战 3：多种笔迹类型识别

- 钢笔、圆珠笔、铅笔
- 蓝色、黑色、红色
- 粗细不同、压力不同

挑战 4：防作弊机制

- 检测异常填涂
- 识别多选作弊
- 图像篡改检测

工程价值

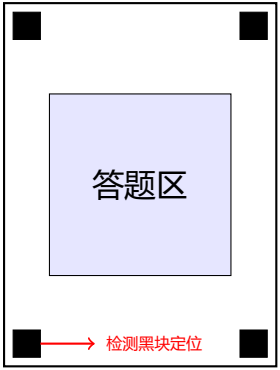
阅卷系统将人工阅卷的准确率从 **95%** 提升到 **99.9%**，效率提升 **100 倍**

项目深度：答题卡的”定位锚点”(Timing Marks)

问题：如果试卷皱了或者拍摄角度极度倾斜，如何准确定位答题卡？

Timing Marks 的作用：

- 答题卡边缘的黑色小方块
- 用于精确定位答题卡区域
- 类似二维码的定位图案



设计规范：

- 位置：四个角或边缘
- 大小：固定的几何尺寸
- 对比度：黑色 vs 白纸
- 排列：特定模式（如 L 形）

工程价值

Timing Marks 是阅卷系统的”眼睛”，有它们即使试卷皱了也能准确定位

定位锚点检测：从轮廓到坐标

核心思路：检测黑色方块 → 计算中心 → 透视变换

```
import cv2
import numpy as np

# 1. 读取答题卡
img = cv2.imread('answer_sheet.
    jpg')
gray = cv2.cvtColor(img, cv2.
    COLOR_BGR2GRAY)

# 2. 二值化
_, binary = cv2.threshold(gray,
    127, 255, cv2.THRESH_BINARY)

# 3. 查找黑色方块轮廓
```

检测流程：

1. **二值化**：分离黑色定位块和白色背景

为什么需要 AI 编程助手?

传统编程的痛点:

- API 参数复杂，记不住
- 报错信息看不懂
- 算法原理理解困难
- 编程基础薄弱

AI 辅助的优势:

- **快速生成代码框架**
- **解释错误原因**
- **提供优化建议**
- **降低学习门槛**

本学期 AI 工具:

- **Claude Code**：代码生成与解释
- **通义千问**：中文友好，国内可用
- **ChatGPT**：通用编程助手

课程特色

用 AI 工具辅助学习，聚焦**理解原理**而非**记忆 API**

下周我们将专门学习如何用 AI 辅助编程!

AI 工具使用规范

本课程 AI 工具：

- **Claude Code**：代码生成与解释
- **通义千问**：中文友好，国内可用
- **ChatGPT**：通用编程助手

使用规范：

✓ **允许**：用 AI 解释概念、调试错误、优化代码

✓ **鼓励**：用 AI 生成对比实验、可视化结果

× **禁止**：直接复制完整代码、用 AI 完成全部作业

核心理念

理解原理 > 复制代码

本学期 AI 工具使用计划

AI 工具应用场景：

1. **Week 2**：AI 辅助编程实战
   - 学习 Prompt 工程
   - 用 AI 生成人脸检测代码
2. **Week 3-4**：图像预处理与版面分析
   - 用 AI 解释复杂算法
   - 调试二值化参数
3. **Week 5-6**：选择题与判断题识别
   - 用 AI 生成代码框架
   - 优化识别算法
4. **Week 7-8**：OCR 与手写识别
   - 用 AI 理解深度学习模型
   - 调试模型训练过程

AI 工具使用规范：

- ✓ 允许：用 AI 解释概念、调试错误、优化代码
- ✓ 鼓励：用 AI 生成对比实验、可视化结果
- × 禁止：直接复制完整代码、用 AI 完成全部作业

重要原则

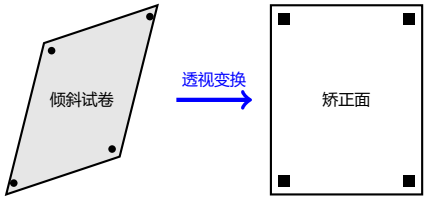
理解原理 > 复制代码
AI 是助手，不是替代者

引出下周：透视变换的力量

场景：检测到定位点后，如何矫正倾斜的试卷？

当前问题：

- 手机拍照难免倾斜
- 试卷可能有透视变形
- 直接处理会降低识别率



下周解决方案：

1. **透视变换**：将任意四边形矫正为矩形
2. **特征匹配**：自动找到定位点

关键技术

`cv2.getPerspectiveTransform()` + `cv2.warpPerspective()`

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)。

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

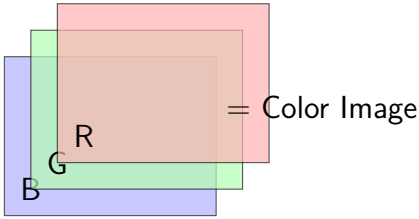
(矩阵数值 $\rightarrow$ 图像亮度)

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**

像素级操作：NumPy 数组操作

基础索引与切片：

```
# 获取单个像素
pixel = img[100, 200]  # 返回[B, G, R]

# 获取红色通道
red_channel = img[:, :, 2]

# 获取左上角100x100区域
top_left = img[0:100, 0:100]

# 水平翻转（左右颠倒）
flipped = img[:, ::-1, :]
^^I^^I
```

统计操作：

```
# 计算平均值
mean_val = np.mean(img)

# 计算标准差
std_val = np.std(img)

# 找最大最小值
max_val = np.max(img)
min_val = np.min(img)

# 计算非零像素数量
nonzero_count = np.count_nonzero(img)
^^I^^I
```

条件操作：

```
# 找出所有暗像素（值 < 128）
dark_pixels = img < 128

# 将暗像素增强
img[dark_pixels] = img[dark_pixels] * 1.2
^^I^^I
```

重要提示

NumPy 切片是**视图（view）**而非副本，修改切片会影响原图！如果需要独立副本，使用 `img.copy()`

像素级操作：手动实现灰度化

原理： $Gray = R \times 0.299 + G \times 0.587 + B \times 0.114$

方法 1：手动循环（学习用，不推荐）

```
def manual_grayscale(img):
    """手动实现灰度化"""
    h, w, c = img.shape
    gray = np.zeros((h, w), dtype=np.uint8)

    for i in range(h):
        for j in range(w):
            b, g, r = img[i, j]
            gray[i, j] = int(0.299*r +
                             0.587*g +
                             0.114*b)
    return gray
^^I^^I
```

缺点：速度慢，不推荐生产环境使用

方法 2：向量化操作（推荐）

```
def grayscale_vectorized(img):
    """向量化实现灰度化"""
    # 方法1：矩阵运算
    b, g, r = cv2.split(img)
    gray = 0.299*r + 0.587*g + 0.114*b
    return gray.astype(np.uint8)

    # 方法2：点积（更简洁）
    weights = np.array([0.114, 0.587, 0.299])
    gray = img.dot(weights).astype(np.uint8)
    return gray
^^I^^I
```

优点：速度快，利用 *NumPy* 向量化加速

性能对比： 手动循环 200ms vs 向量化操作 5ms（快 40 倍）

像素级操作：亮度调整的”溢出”陷阱

错误做法：

```
img = cv2.imread('exam.jpg')

# 直接相加
bright = img + 50

# 问题：如果像素值是220，
# 220 + 50 = 270
# 但uint8的范围是0-255
# 270会截断为14（或绕回）
# 导致图像出现噪点！
^^I^^I
```

正确做法：

```
# 方法1：使用np.clip
bright = np.clip(
    img.astype(np.int32) + 50,
    0, 255
).astype(np.uint8)

# 方法2：使用cv2.add（推荐）
bright = cv2.add(
    img,
    np.array([50.0])
)

# 方法3：使用convertScaleAbs
bright = cv2.convertScaleAbs(
    img,
    alpha=1.0,  # 对比度
    beta=50     # 亮度增量
)
^^I^^I
```

为什么？

uint8 类型：8 位无符号整数

范围：$[0, 255]$

溢出：截断到边界值

核心原理

图像的统计特性：直方图 (Histogram)

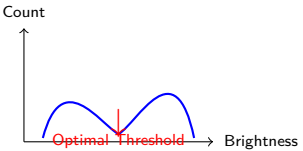
什么是直方图?

- 横坐标：亮度级别 (0-255)
- 纵坐标：该亮度像素出现的频次

在阅卷中的意义：

- **曝光检查**：判断照片是否太暗或过曝
- **二值化参考**：寻找波谷作为分割阈值

```
import cv2
import matplotlib.pyplot as plt

# 计算直方图
img = cv2.imread('exam.jpg',
                 cv2.IMREAD_GRAYSCALE)
hist = cv2.calcHist([img], [0], None,
                    [256], [0, 256])

plt.plot(hist)
plt.title('Pixel Intensity Distribution')
plt.show()
^^I^^I
```



直方图形态分析：试卷照片的曝光诊断

场景：自动判断试卷照片的质量

1. **正常曝光（双峰分布）**
   - 白纸（高亮度）+ 黑字（低亮度）
   - 波谷在中间，适合二值化
   - **阅卷理想状态**
2. **欠曝（左偏分布）**
   - 大部分像素集中在暗部
   - 可能是光照不足
   - 需要亮度增强
3. **过曝（右偏分布）**
   - 大部分像素集中在亮部
   - 可能是闪光灯太强
   - 需要降低亮度

```python
def check_exposure(img):
    """检查图像曝光情况"""
    gray = cv2.cvtColor(img,
                        cv2.COLOR_BGR2GRAY)

    # 计算平均亮度
    mean_brightness = np.mean(gray)

    # 判断曝光状态
    if mean_brightness < 80:
        return "欠曝，建议增强"
    elif mean_brightness > 200:
        return "过曝，建议降低"
    else:
        return "曝光正常"

# 使用
status = check_exposure(img)
print(f"曝光状态：{status}")
^^I^^I
```

图像缩放：插值算法对比

问题：cv2.resize 时，像素是如何" 凭空产生" 或" 消失" 的?

```python
import cv2
import numpy as np

img = cv2.imread('exam.jpg')
h, w = img.shape[:2]

# 1. 最近邻插值（最快）
# 取最近的像素值，会有马赛克
near = cv2.resize(img, (w*2, h*2),
                  interpolation=cv2.INTER_NEAREST)

# 2. 双线性插值（默认）
# 周围4个像素加权平均，较平滑
linear = cv2.resize(img, (w*2, h*2),
                    interpolation=cv2.INTER_LINEAR)

# 3. 双三次插值（最慢但最好）
# 周围16个像素加权平均
cubic = cv2.resize(img, (w*2, h*2),
                   interpolation=cv2.INTER_CUBIC)
```

插值效果对比：

- **最近邻**：
  - 速度：最快
  - 质量：有锯齿
  - 应用：像素风游戏
- **双线性**：
  - 速度：中等
  - 质量：较平滑
  - 应用：日常缩放
- **双三次**：
  - 速度：最慢
  - 质量：最平滑
  - 应用：高质量缩放

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)
^^I
```

常用 Flag

工程实战：处理中文路径的”终极方案”

真实场景

阅卷系统中，学生姓名经常是中文，文件名如：张三 _ 试卷.jpg
cv2.imread() 直接读取会失败（返回 None）

问题原因：

- OpenCV 的 C++ 库不支持 Unicode 路径
- 中文路径会乱码

错误做法：

```
# 直接读取，返回 None
img = cv2.imread('张三_试卷.jpg')
if img is None:
    print("读取失败！")
^^I^^I
```

正确做法：

```
import cv2
import numpy as np

def imread_chinese(path):
    """读取中文路径的图片"""
    img_array = np.fromfile(path,
                            dtype=np.uint8)
    img = cv2.imdecode(img_array, -1)
    return img

# 使用
img = imread_chinese('张三_试卷.jpg')
cv2.imshow('成功！', img)
^^I^^I
```

工程实战：保存中文路径图片

配套技巧：保存时也支持中文路径

```python
import cv2
import numpy as np

def imwrite_chinese(path, img):
    """保存中文路径的图片"""
    is_success, img_buf = cv2.imencode(
        ".jpg", img)
    if is_success:
        img_buf.tofile(path)
        return True
    return False

# 使用
img = cv2.imread('exam.jpg')
imwrite_chinese('处理结果_张三.jpg', img)
print("保存成功！")
^^I^^I
```

原理说明：

1. **imdecode**：从内存中的字节数组解码
2. **imencode + tofile**：编码为字节数组后写入

阅卷系统应用：

- 批量处理学生试卷
- 生成带学生姓名的结果文件

AI 辅助实战示例：解决中文路径问题

场景: OpenCV 无法读取中文路径，如何用 AI 解决?

[colback=blue!5!white,colframe=blue!50!black,title= 用户提问] " 我用 cv2.imread 读取中文路径的图片返回 None，怎么解决？ "

[colback=green!5!white,colframe=green!50!black,title= AI 回答] "OpenCV 的 imread 不支持中文路径。解决方法：

1. 用 numpy.fromfile 读取为字节数组
2. 用 cv2.imdecode 解码"

```
import numpy as np
import cv2

# 支持中文路径的读取方式
img = cv2.imdecode(np.fromfile(path, dtype=np.uint8), -1)
^^I
```

AI 的价值: 告诉你问题的本质（OpenCV 的 C++ 底层限制），直接给出可用代码，节省搜

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()
^^I
```

注意: 灰度图显示时需要设置 cmap='gray', 否则会变成" 原油色"。

```
plt.imshow(gray_img, cmap='gray')
^^I
```

OpenCV 进阶：几何变换

1. 平移（Translation）

```
# 向右平移100，向下平移50
M = np.float32([
    [1, 0, 100],  # x位移
    [0, 1, 50]    # y位移
])
translated = cv2.warpAffine(
    img, M, (w, h)
)
^^I^^I
```

2. 旋转（Rotation）

```
# 逆时针旋转30度
center = (w//2, h//2)  # 旋转中心
M = cv2.getRotationMatrix2D(
    center,
    30,      # 角度（度）
    1.0      # 缩放比例
)
rotated = cv2.warpAffine(
    img, M, (w, h)
)
^^I^^I
```

3. 缩放（Scaling）

```
# 放大2倍
scaled = cv2.resize(
    img,
    None,
    fx=2.0,
    fy=2.0,
    interpolation=cv2.INTER_CUBIC
)

# 指定尺寸缩放
resized = cv2.resize(
    img,
    (800, 600),  # 宽，高
    interpolation=cv2.INTER_LINEAR
)
^^I^^I
```

插值方法选择

- 放大：INTER_CUBIC（质量最好）
- 缩小：INTER_AREA（抗锯齿）

OpenCV 进阶：仿射变换与透视变换

仿射变换（Affine Transform）

- 保持平行线的平行性
- 需要 3 个点对应

```python
# 原图像的3个点
src_pts = np.float32([
    [50, 50],    # 左上
    [200, 50],   # 右上
    [50, 200]    # 左下
])

# 目标图像的3个点
dst_pts = np.float32([
    [10, 10],
    [200, 20],
    [10, 200]
])

# 计算变换矩阵
M = cv2.getAffineTransform(
    src_pts, dst_pts
)
```

透视变换（Perspective Transform）

- 不保持平行性
- 需要 4 个点对应
- **下周重点！**

```python
# 原图像的4个角点
src_pts = np.float32([
    [100, 150],  # 左上
    [450, 120],  # 右上
    [480, 380],  # 右下
    [80, 400]    # 左下
])

# 目标矩形
width, height = 400, 300
dst_pts = np.float32([
    [0, 0],
    [width-1, 0],
    [width-1, height-1],
    [0, height-1]
])

# 计算变换矩阵
M = cv2.getPerspectiveTransform(
```

OpenCV 进阶：形态学操作基础

形态学操作：基于图像形状的变换，常用于二值图像

1. 腐蚀（Erosion）

- 膨胀白色，收缩黑色
- 去除小噪点

```
kernel = np.ones((3, 3), np.uint8)
eroded = cv2.erode(binary, kernel,
                   iterations=1)
^^I^^I
```

2. 膨胀（Dilation）

- 收缩白色，膨胀黑色
- 填充小孔洞

```
dilated = cv2.dilate(binary, kernel,
                     iterations=1)
^^I^^I
```

4. 闭运算（Closing）

- 先膨胀后腐蚀
- 填充小孔洞，连接近邻物体

```
closing = cv2.morphologyEx(
    binary, cv2.MORPH_CLOSE, kernel
)
^^I^^I
```

5. 形态学梯度

- 膨胀 - 腐蚀
- 提取边缘

```
gradient = cv2.morphologyEx(
    binary, cv2.MORPH_GRADIENT,
    kernel
```

形态学操作在阅卷中的应用

场景 1：去除填涂噪点

- 问题：填涂边缘有细小噪点
- 解决：开运算去除小噪点

```
# 去除小噪点
kernel = np.ones((2, 2), np.uint8)
clean_bubble = cv2.morphologyEx(
    binary,
    cv2.MORPH_OPEN,
    kernel
)
^^I^^I
```

场景 2：连接断开的笔迹

- 问题：手写数字断开
- 解决：闭运算连接

```
# 连接断开部分
kernel = np.ones((3, 3), np.uint
```

场景 3：提取轮廓边缘

- 问题：需要清晰的轮廓
- 解决：形态学梯度

```
# 提取边缘
gradient = cv2.morphologyEx(
    binary,
    cv2.MORPH_GRADIENT,
    kernel
)
^^I^^I
```

可视化对比

- 原始：有噪点的填涂
- 开运算：干净的填涂
- 闭运算：连接的笔迹
- 梯度：清晰的边缘

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。)

```
# 方法1: OpenCV 函数
# TODO 1: 使用 cv2.cvtColor 将彩色图像转为灰度图
# 提示: 参数为 cv2.COLOR_BGR2GRAY
gray = cv2.cvtColor(img, __________)

# 方法2: 手动计算（不推荐）
# TODO 2: 填写正确的权重系数
# 提示: 人眼对绿色最敏感，绿色权重最大
gray = ____ * r + ____ * g + ____ * b
```

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。

```
# 方法1: NumPy 运算
# TODO 1: 实现图像反色
# 提示: 反色公式 NewValue = 255 - OldValue
inverted = ____ - img

# 方法2: OpenCV 位运算
# TODO 2: 使用 OpenCV 函数实现反色
# 提示: 使用 cv2.bitwise_not
inverted = _________________

# 方法3: NumPy 按位取反(选做)
```

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
# TODO 1: 安全地增加亮度，防止溢出
# 提示：先转为int32，加50后用np.clip限制到[0,255]，再转回uint8
bright_img = np.clip(img.astype(np.int32) + ___, ___, ___).astype
    (______)

# 或者使用 OpenCV 内置函数（推荐，速度更快）
# TODO 2: 使用 cv2.add 实现安全加法
# 提示：cv2.add 会自动处理溢出
bright_img = cv2.add(img, np.array([______]))
```

代码实战（1/5）：图像翻转与旋转

场景：阅卷时试卷可能被倒置，需要自动旋转

```python
import cv2
import numpy as np

img = cv2.imread('exam.jpg')

# 方法1: NumPy 数组切片
# TODO 1: 实现垂直翻转（上下颠倒）
# 提示：使用切片 [::-1, :, :] 来反转第一个维度（高度）
flip_v = img[____, :, :]

# TODO 2: 实现水平翻转（左右颠倒）
# 提示：使用切片 [:, ::-1, :] 来反转第二个维度（宽度）
flip_h = img[:, ____, :]

# TODO 3: 实现水平+垂直翻转（旋转180度）
# 提示：同时反转高度和宽度两个维度
flip_both = img[____, ____, :]
```

```python
# 方法2: OpenCV 函数（推荐）
flip_v = cv2.flip(img, 0)       # 垂直
flip_h = cv2.flip(img, 1)       # 水平
flip_both = cv2.flip(img, -1)   # 两者

# 显示对比
cv2.imshow('Original', img)
cv2.imshow('Flip V', flip_v)
cv2.imshow('Flip H', flip_h)
cv2.waitKey(0)
```

性能对比： NumPy 切片比 cv2.flip 快约 20%，但 cv2.flip 更易读

[colback=green!5!white,colframe=green!50!black,title= 预期结果] 成功运行后，应看到：

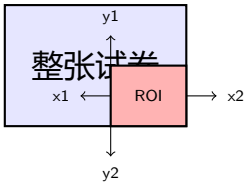
代码实战（2/5）：提取答题卡区域（ROI）

场景：从整张试卷中提取答题卡区域

```python
import cv2
import numpy as np

exam = cv2.imread('exam.jpg')
h, w = exam.shape[:2]

# 假设答题卡在右下角
# TODO 1: 计算 ROI 的坐标范围
# 提示：答题卡位于右下区域，宽度从60%到末尾
x1, x2 = int(w * ______), ___
# 提示：高度从50%到末尾
y1, y2 = int(h * ______), ___

# TODO 2: 使用 NumPy 切片提取 ROI
# 提示：图像切片的格式是 img[y_start:y_end, x_start:x_end]
roi = exam[____:____, ____:____]

# 保存 ROI
cv2.imwrite('answer_sheet.jpg', roi)

print("原图大小:", exam.shape)
print("ROI 大小:", roi.shape)
```

坐标系统回顾：

- 原点在左上角 (0, 0)
- `img[y1:y2, x1:x2]`
- y 是行（高度），x 是列（宽度）



代码实战（3/5）：通道分离与合并

场景：提取特定颜色通道

```python
import cv2

img = cv2.imread('exam.jpg')

# 方法1：使用 split 函数
b, g, r = cv2.split(img)

# 只保留红色通道，其他设为0
zeros = np.zeros_like(b)
img_r = cv2.merge([zeros, zeros, r])
```

```python
# 方法2：直接索引（更快）
img_r = img.copy()
img_r[:, :, 0] = 0  # B通道
img_r[:, :, 1] = 0  # G通道
# R通道保持不变

cv2.imshow('Red Only', img_r)
cv2.waitKey(0)
```

通道顺序：

- OpenCV: **BGR**
- matplotlib: **RGB**
- PIL: **RGB**

常见错误

使用 `plt.imshow(img)` 显示 OpenCV 图像时，颜色会异常！

解决方案：

```python
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

代码实战（4/5）：阅卷系统核心代码

场景：检测答题卡填涂位置

```python
import cv2
import numpy as np

# 1. 读取答题卡区域
roi = cv2.imread('answer_sheet.jpg',
                 cv2.IMREAD_GRAYSCALE)

# 2. 二值化
_, binary = cv2.threshold(roi, 127, 255,
                          cv2.THRESH_BINARY)

# 3. 定义选项位置
positions = [
    (100, 100, 120, 120),  # A
    (100, 130, 120, 150),  # B
    (100, 160, 120, 180),  # C
    (100, 190, 120, 200)   # D
]
```

```python
# 4. 检测每个选项是否被填涂
answers = []
for (x1, y1, x2, y2) in positions:
    option = binary[y1:y2, x1:x2]

    # 计算黑色像素比例
    black_pixels = np.sum(option == 0)
    total_pixels = option.size
    ratio = black_pixels / total_pixels

    # 判断是否填涂（阈值30%）
    if ratio > 0.3:
        answers.append('填涂')
    else:
        answers.append('未填')

print(answers)
```

核心思想：填涂区域黑色像素占比显著高于未填涂区域

[colback=green!5!white,colframe=green!50!black,title= 预期结果] 成功运行后，应输出：

代码实战（5/5）：图像增强对比

场景：答题卡光照不均，需要增强对比度

```python
import cv2
import numpy as np

img = cv2.imread('exam.jpg')

# 方法1：线性对比度调整
# new = alpha * old + beta
enhanced = cv2.convertScaleAbs(
    img, alpha=1.5, beta=30
)
```

```python
# 方法2：直方图均衡化
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
equalized = cv2.equalizeHist(gray)
```

```python
# 方法3：CLAHE（自适应）
clahe = cv2.createCLAHE(
    clipLimit=2.0,
    tileGridSize=(8,8)
)
enhanced_clahe = clahe.apply(gray)
```

效果对比：

- **线性调整**：简单但效果有限
- **直方图均衡化**：全局优化
- **CLAHE**：局部自适应，效果最好

阅卷推荐： CLAHE 适合光照不均场景

挑战任务：智能裁剪答题卡

任务描述：给定一张试卷照片，自动识别并裁剪出答题卡区域

挑战目标

- 找到答题卡四个角点
- 透视变换矫正
- 输出标准的答题卡图像

难度：

需要综合运用：

- 边缘检测
- 轮廓分析
- 几何变换

提示：

- 使用 `cv2.Canny` 检测边缘
- 使用 `cv2.findContours` 找轮廓
- 使用 `cv2.getPerspectiveTransform` 透视变换

预期结果：

- 输入：任意角度拍摄的试卷照片
- 输出：正面朝上的答题卡图像
- 测试：用自己手机拍的试卷验证

挑战任务：答题卡填涂密度热图

任务描述：可视化全班答题卡每个选项的填涂频率

```
import matplotlib.pyplot as plt
import numpy as np

# 假设有全班同学的答题数据
# answers[i][j] 表示第i题第j个选项的填涂次数
answers = [
    [25, 5, 2, 0],  # 第1题：A最多选
    [3, 20, 8, 1],  # 第2题：B最多选
    # ... 更多题目
]

# TODO：生成热图
# 1. 转换为numpy数组
data = np.array(answers)

# 2. 使用plt.imshow显示热图
# 提示：使用cmap='YlOrRd' colormap
plt._______(data, cmap='______')

# 3. 添加颜色条
plt._________()

# 4. 添加题号和选项标签
plt.xticks([0,1,2,3], ['A','B','C','D'])
# TODO：添加y轴标签（题号）
plt.yticks(range(len(data)), [_____ for ___, _ in enumerate(data)])
```

教学价值

帮助教师快速发现：

- 哪些题目太简单（A 选项 90
- 哪些题目太困难（分散度高）
- 迷惑选项是否有效（B/C 选了多少）

预期结果：

- 热图清晰展示填涂分布
- 颜色越红表示填涂越多

Live Coding：完整的阅卷预处理流程

目标：从照片到可识别的图像

```python
def preprocess_exam(image_path):
    """试卷预处理完整流程"""

    # 1. 读取图像（支持中文路径）
    img = imread_chinese(image_path)

    # 2. 转为灰度
    gray = cv2.cvtColor(img,
                        cv2.COLOR_BGR2GRAY)

    # 3. 去噪
    denoised = cv2.GaussianBlur(
        gray, (5, 5), 0)

    # 4. 对比度增强（CLAHE）
    clahe = cv2.createCLAHE(2.0, (8, 8))
    enhanced = clahe.apply(denoised)

    # 5. 二值化
    binary = cv2.adaptiveThreshold(
        enhanced, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY, 11, 2)

    return img, gray, enhanced, binary

# 使用
```

流程图：

展示结果：

- 原始照片

Live Coding：阅卷系统核心检测（基础版）

功能 1：填涂检测

```
def detect_bubble(binary, position):
    """检测单个气泡的填涂状态 - 基础版"""
    x1, y1, x2, y2 = position

    # TODO 1: 提取气泡区域
    # 提示: 使用NumPy切片提取矩形区域 [y1:y2, x1:x2]
    bubble = binary[____:____, ____:____]

    # TODO 2: 计算填涂密度
    # 提示: 黑色像素值为0, 用 np.sum(bubble == 0) 统计
    black_pixels = ________________
    total_pixels = ________.____
    fill_ratio = black_pixels / total_pixels

    # TODO 3: 根据填涂密度判断状态
    # 提示: >0.6为已填涂, <0.2为未填涂, 中间为不确定
    if _____________:
        return 'filled'
    elif _____________:
        return 'empty'
    else:
        return 'uncertain'
^^I
```

功能 2：多选检测与警告（进阶版选做）

Live Coding：图像质量检测函数

目标：自动判断试卷照片是否适合识别

```python
def check_image_quality(img):
    """检测图像质量"""

    h, w = img.shape[:2]

    # 1. 分辨率检查
    if min(h, w) < 1000:
        return False, "分辨率过低"

    # 2. 曝光检查
    gray = cv2.cvtColor(img,
                        cv2.COLOR_BGR2GRAY)
    mean_brightness = np.mean(gray)

    if mean_brightness < 80:
        return False, "曝光不足"
    elif mean_brightness > 200:
        return False, "过曝"

    # 3. 清晰度检查
    laplacian_var = cv2.Laplacian(
        gray, cv2.CV_64F
    ).var()

    if laplacian_var < 100:
        return False, "图像模糊"
```

使用示例：

```python
img = imread_chinese('exam.jpg')

is_good, msg = check_image_quality(img)

if is_good:
    print(f"图像质量：{msg}")
    # 继续处理
    result = process_image(img)
else:
    print(f"图像质量：{msg}")
    print("提示用户重新拍照")
^^I^^I
```

质量标准：

- 分辨率： 1000px
- 曝光：80-200
- 清晰度：方差 100

Live Coding：批量处理与结果输出

批量处理函数：

```python
import os
import json

def batch_process_exams(folder_path, output_path):
    """批量处理试卷"""
    results = []

    for filename in os.listdir(folder_path):
        if not filename.endswith(('.jpg', '.png')):
            continue

        input_path = os.path.join(folder_path, filename)

        # 1. 质量检查
        is_good, msg = check_image_quality(
            imread_chinese(input_path))
        if not is_good:
            print(f"X {filename}: {msg}")
            continue

        # 2. 预处理
        img, gray, enhanced, binary = \
            preprocess_exam(input_path)

        # 3. 检测答题
        answers = detect_all_answers(binary)
```

硬件知识（1/3）：卷帘快门 vs 全局快门

快门：控制传感器曝光时间的机制

卷帘快门

- 逐行曝光，像" 扫描" 一样
- 优点：便宜、功耗低
- 缺点：拍摄快速物体会变形
- **斜坡效应**
  - 照片上下部分时间不同
  - 快速物体会倾斜

应用：手机摄像头、普通相机

全局快门

- 所有像素同时曝光
- 优点：无变形
- 缺点：昂贵、功耗高
- 工业相机标配

应用：工业检测、高速摄影

阅卷系统建议

硬件知识（2/3）：CMOS vs CCD 传感器

图像传感器：将光信号转换为电信号的芯片

CMOS 传感器

- 每个像素有独立的放大器
- 优点：
  - 速度快（并行读出）
  - 功耗低
  - 成本低
  - 集成度高
- 缺点：
  - 噪声较大
  - 填充因子低

应用：手机、网络摄像头、消费级相机

CCD 传感器

- 所有电荷通过一个放大器
- 优点：
  - 图像质量高
  - 噪声低
  - 动态范围大
  - 一致性好
- 缺点：
  - 速度慢（串行读出）
  - 功耗高
  - 成本高

应用：天文摄影、科学成像、高端相机

硬件知识（3/3）：图像传感器关键参数

选择相机时需要关注的参数

1. 分辨率
- 像素数量：1920×1080, 4K 等
- **不是越高越好！**
- 阅卷：300dpi 扫描足够

2. 帧率
- 每秒传输图像数
- 高速扫描：30-60 fps

3. 像素尺寸
- 单个像素物理大小（ m）
- 越大 = 进光量越大 = 噪声越低

4. 动态范围
- 能同时捕捉的最亮和最暗细节
- 单位：dB 或 stops
- 高端：60-80 dB
- 普通：40-60 dB

5. 信噪比（SNR）
- 信号与噪声的比值
- 越高越好
- >40dB 为优秀

6. ISO 感光度
- 对光的敏感程度

工业相机选型指南

阅卷系统相机选型决策树

根据吞吐量选择：

- **小型（<1000 张/天）**：USB3.0 相机
- **中型（1000-10000 张/天）**：GigE 相机
- **大型（>10000 张/天）**：Camera Link 高速相机

根据精度选择：

- **普通阅卷**：300dpi　5MP
- **手写识别**：600dpi　10MP
- **图表分析**：1200dpi　20MP

热门品牌对比：

品牌	价格	质量	支持
Basler	中	高	好
IDS	中	高	好
海康	低	中	中
大华	低	中	中

接口选择：

- **USB3.0**：短距离（<3m），即插即用
- **GigE**：长距离（<100m），需交换机
- **Camera Link**：超高速，专用线缆

阅卷系统硬件配置建议

完整硬件方案示例

方案 A：经济型（学校用）

- 相机：USB3.0 工业相机（5MP，30fps）
- 镜头：定焦镜头（8mm, f/2.8）
- 光源：LED 环形光（可调亮度）
- 传输：USB3.0 线缆（2 米）
- 预算：约 5000-8000 元

方案 B：专业型（考试中心）

- 相机：GigE 全局快门（10MP，60fps）
- 镜头：远心镜头（低畸变）
- 光源：条形光源阵列

关键配置要点：

1. **全局快门**：避免文字倾斜
2. **镜头畸变 $<1\%$**：保证定位精度
3. **均匀照明**：光照变化 $<10\%$
4. **固定焦距**：避免自动对焦抖动
5. **稳定支架**：减少震动影响

避坑指南

- 不要用手机摄像头（卷帘快门）
- 不要用网络摄像头（分辨率低）

小测验时间（1）：NumPy 综合测试

问题

给定形状为 $(100, 100, 3)$ 的彩色图像，如何提取中心 50×50 的红色通道值？

A. `img[25:75, 25:75, 0]`

B. `img[25:75, 25:75, 2]`

C. `img[50:100, 50:100, 2]`

D. `img[25:75, 25:75]`

小测验时间（1）：NumPy 综合测试

问题

给定形状为 (100, 100, 3) 的彩色图像，如何提取中心 50×50 的红色通道值?

A. `img[25:75, 25:75, 0]`

B. `img[25:75, 25:75, 2]`

C. `img[50:100, 50:100, 2]`

D. `img[25:75, 25:75]`

答案： **B. img[25:75, 25:75, 2]**

解析：

- 100×100 图像的中心 50×50：索引从 25 到 75
- OpenCV 中 BGR 顺序，红色通道索引为 2
- `img[y1:y2, x1:x2, channel]` 格式

Live Coding（1/3）：5 分钟闪电编程任务

编程挑战

给定一张”脏”试卷图像（含噪声、光照不均）
任务：在 5 分钟内，用 **3 行代码** 提取出学号区的均值

提示：

1. 读取图像（已有）
2. 切片学号区
3. 计算均值

学号区位置：

- 假设在左上角
- 坐标范围：[50:150，100:300]
- 高度：100px，宽度：200px

```python
# 给定代码（不要修改）
import cv2
import numpy as np

img = cv2.imread('dirty_exam.jpg')

# ===== 你的任务：补全这3行 =====
# 第1行：灰度化
gray = ???

# 第2行：切片学号区
id_region = ???

# 第3行：计算均值
mean_value = ???

# ==============================

print(f"学号区均值：{mean_value}")
```

Live Coding（2/3）：参考答案与解析

参考答案：

```python
import cv2
import numpy as np

img = cv2.imread('dirty_exam.jpg')

# 第1行：灰度化
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 第2行：切片学号区
id_region = gray[50:150, 100:300]

# 第3行：计算均值
mean_value = np.mean(id_region)

print(f"学号区均值：{mean_value:.2f}")
```

代码解析：

- **灰度化**：`cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)` 将三维彩色图转为二维灰度图
- **切片**：`gray[y1:y2, x1:x2]`，y 在前，x 在后，提取高度方向 [50:150]，宽度方向 [100:300]

Live Coding（3/3）：即时反馈与扩展

课堂互动：分享你的切片坐标

学生分享环节：

- 你用的是哪个坐标范围？
- `img[50:150, 100:300]`？
- 还是 `img[0:100, 0:200]`？
- 坐标不同，结果如何？

观察要点：

1. 不同坐标范围，均值会不同
2. 越亮的区域，均值越大
3. 越暗的区域，均值越小

进阶挑战（可选）：

```python
# 挑战1：计算标准差
std_value = np.std(id_region)
print(f"标准差：{std_value:.2f}")

# 挑战2：判断是否填涂
if mean_value < 100:
    print("学号区可能已填涂")
else:
    print("学号区未填涂")

# 挑战3：可视化切片
cv2.imshow('学号区', id_region)
cv2.waitKey(0)
```

阅卷应用：

- 均值：判断填涂密度
- 标准差：判断填涂均匀性
- 阈值：自动识别填涂状态

小测验时间（5）：综合应用

问题

在阅卷系统中，要将答题卡的填涂区域（黑色）从白色纸张中分离出来，应该使用哪种阈值类型？

A. `cv2.THRESH_BINARY`
B. `cv2.THRESH_BINARY_INV`
C. `cv2.THRESH_TRUNC`
D. `cv2.THRESH_TOZERO`

小测验时间（5）：综合应用

问题

在阅卷系统中，要将答题卡的填涂区域（黑色）从白色纸张中分离出来，应该使用哪种阈值类型？

A. `cv2.THRESH_BINARY`

B. `cv2.THRESH_BINARY_INV`

C. `cv2.THRESH_TRUNC`

D. `cv2.THRESH_TOZERO`

答案： **A. cv2.THRESH_BINARY**

解析：

- 填涂区域是黑色（低像素值）
- 纸张是白色（高像素值）
- THRESH_BINARY 会将低于阈值的设为 0（黑），高于阈值的设为 255（白）

阅卷系统完整流程（1/3）：图像预处理

目标：将拍摄的试卷图像转换为适合处理的格式

```
import cv2
import numpy as np

# 1. 读取图像
img = cv2.imread('exam_paper.jpg')

# 2. 灰度化
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 3. 去噪
denoised = cv2.GaussianBlur(gray, (5, 5), 0)

# 4. 对比度增强
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
```

阅卷系统完整流程（2/3）：定位答题卡

目标：在试卷中找到答题卡区域

```
# 1. 边缘检测
edges = cv2.Canny(binary, 50, 150)

# 2. 查找轮廓
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)

# 3. 筛选矩形轮廓
for cnt in contours:
    peri = cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)
    if len(approx) == 4:
        x, y, w, h = cv2.boundingRect(cnt)
        answer_sheet = binary[y:y+h, x:x+w]
```

阅卷系统完整流程（3/3）：识别填涂

目标：识别每个选项是否被填涂

```
# 定义每个选项的位置
options = [
    (50, 30, 80, 60),    # 第1题 A
    (90, 30, 120, 60),   # 第1题 B
]

results = []
for (x1, y1, x2, y2) in options:
    option = answer_sheet[y1:y2, x1:x2]
    black_ratio = np.sum(option == 0) / option.size
    is_filled = black_ratio > 0.3  # 阈值30%
    results.append(is_filled)
```

识别算法优化：

快速问答环节

问题 1：OpenCV 默认读取的彩色图像是什么顺序?

A RGB

B **BGR**（正确）

C HSV

D LAB

问题 2：如何判断一个图像是否读取成功?

A if img != None

B **if img is not None**（正确）

C if img.exists()

D if len(img) > 0

问题 3：uint8 类型的像素值范围是?

A 0-1023

B **0-255**（正确）

C -128-127

D 0-65535

问题 4：图像像素相加时，如何避免溢出?

A 直接相加

B **cv2.add() 或 np.clip()**（正确）

C 转为 float 后相加

D 无需处理

快速问答环节

问题 1：OpenCV 默认读取的彩色图像是什么顺序?

A RGB

B **BGR**（正确）

C HSV

D LAB

问题 2：如何判断一个图像是否读取成功?

A if img != None

B **if img is not None**（正确）

C if img.exists()

D if len(img) > 0

问题 3：uint8 类型的像素值范围是?

A 0-1023

B **0-255**（正确）

C -128-127

D 0-65535

问题 4：图像像素相加时，如何避免溢出?

A 直接相加

B **cv2.add() 或 np.clip()**（正确）

C 转为 float 后相加

D 无需处理

小测验时间（2）：代码找错挑战

找出以下代码中的 3 个错误：

```
import cv2
import numpy as np

# 读取图像
img = cv2.imread('张三试卷.jpg')  # 错误1

# 亮度增加50
bright_img = img + 50  # 错误2

# 显示
plt.imshow(img)  # 错误3
plt.show()
```

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

知识点网络与下周预告

本周核心知识点：

- 计算机视觉基本概念
- 图像的数字表示（矩阵、RGB）
- OpenCV 基础操作
- 图像预处理（灰度、二值化、去噪）
- 阅卷系统入门

下周预告（Week 2）：AI 辅助编程工具实战

- **ChatGPT/Claude**：学习编程的 AI 助手
- **Prompt 工程**：如何让 AI 帮我们写代码
- **实战演练**：用 AI 辅助实现人脸检测

跨周链接

- Week 1：图像基础
- Week 2：AI 工具
- Week 3：图像预处理（深度）
- Week 4：版面分析
- Week 5：选择题识别

重点提示

Week 2 我们将学习如何用 **AI 工具**来加速 Week 1 学到的 OpenCV 代码开发！

总结与问答

Q & A 准备好进入计算机视觉的世界了吗?