

第 5 周：选择题识别（填涂检测）

怎么知道选了 A 还是 B?

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

北京石油化工学院



本周内容:

- OMR 技术概述
- 图像形态学处理
- 连通域分析
- 填涂区域定位
- 填涂检测算法
- 完整识别系统

智能阅卷系统进度:

图像采集与预处理

答题卡区域定位

① 选择题填涂检测

— 判断题符号识别

— 手写文字 OCR

本周时间分配 (135 分钟 = 3 学时)

第 1 学时 (45 分钟):

- 00:10 预备知识回顾 (10min)
- 00:25 OMR 技术概述 (15min)
- 00:45 形态学理论 + 演示 (20min)

第 2 学时 (45 分钟):

- 01:05 连通域分析理论 (20min)
- 01:30 Live Coding: 形态学与连通域 (25min)

第 3 学时 (45 分钟):

- 01:30-01:55 Live Coding: 填涂检测 (25min)
- 01:55-02:10 案例分析与讨论 (15min)
- 02:10-02:25 课堂 Quiz (15min)

课前准备

预习：二值图像操作、形态学运算原理
(5 分钟视频)

三个理解层级：本周学习路径

基础概念：

- 什么是 OMR?
- 像素密度的概念
- 形态学基本运算
- 连通域的原理

可视化演示：

- 对比填涂与未填涂的密度差异
- 观察形态学处理效果
- 连通域标记可视化
- 调整阈值观察识别效果

扩展应用：

- 设计自适应 OMR 识别器
- 处理不同填涂程度答题卡
- 优化识别准确率
- 实现完整阅卷系统

每个知识点都按照“理解 → 观察 → 创造”的层次递进

预备知识（课前 5 分钟视频）

本周需要的前置知识

- **二值图像的基本操作**

- 什么是二值图像？
- 阈值处理：全局阈值 vs 自适应阈值

- **形态学运算原理**

- 腐蚀（Erosion）与膨胀（Dilation）
- 开运算与闭运算

[课前视频] 扫码观看 5 分钟预备知识视频

三种学习路径

观察者

- 理解 OMR 原理
- 看教师演示
- 完成基础思考题

适合：无编程基础

使用者

- 运行示例代码
- 调整参数观察效果
- 完成核心任务

推荐：大多数学生

创造者

- 修改算法逻辑
- 优化阈值策略
- 实现挑战任务

挑战：有编程基础

分组与角色

分组原则 (4 人/组):

- 不同专业背景混合
- 至少 1 人有编程基础

角色分工:

组长 协调进度, 分配任务

技术负责人 把关代码质量

模块开发 A 负责形态学与定位

模块开发 B 负责检测与识别

AI 辅助编程本周重点

用 AI 理解复杂概念

- [AI 提示] " 请用直观的比喻解释形态学腐蚀和膨胀 "
- [AI 提示] " 什么是连通域? 如何用图像找连通区域? "

用 AI 生成代码框架

- [AI 提示] " 帮我创建一个 ChoiceRecognizer 类, 包含 $\text{calculate}_{density}$ "
- [AI 提示] " 如何用 OpenCV 统计二值图像中的白色像素数量? "

AI 调试技巧

- ① 贴出完整的错误信息 (Traceback)
- ② 说明你的代码意图

多屏协同设计

教室布局与屏幕分工：

主屏（理论）	侧屏（演示）	设备（互动）
<ul style="list-style-type: none">● 核心概念讲解● 算法原理展示● 知识点梳理	<ul style="list-style-type: none">● 实时代码演示● AI 辅助编程展示● 处理效果对比	<ul style="list-style-type: none">● Quiz 实时答题● 代码片段查看● 小组讨论记录

动静结合原则

每个知识点：理论讲解（15min）→ 代码演示（15min）→ 实践操作（15min）

什么是 OMR?

OMR: Optical Mark Recognition

光学标记识别——通过检测填涂区域来识别答案

核心思想：填涂区域与空白区域的**像素密度差异**



填涂 → 密度高 空白 → 密度低

特性	OMR (光学标记识别)	OCR (光学字符识别)
识别对象	填涂标记、气泡	文字、字符
判断依据	像素密度	特征匹配、深度学习
典型应用	标准化考试答题卡	文档数字化、车牌识别
技术复杂度	较低	较高
准确率	极高 (99%+)	取决于场景

表: OMR 与 OCR 的对比

OMR 技术发展简史

1930s 第一台 OMR 机器用于考试评分

1960s IBM 推出大型 OMR 系统

1980s 个人电脑 + 扫描仪实现桌面 OMR

2000s 数字相机 + 图像处理实现便携 OMR

2020s 手机拍照 + AI 实现智能 OMR

本周目标：用 Python+OpenCV 实现一个基础的 OMR 系统

标准化考试答题卡技术

答题卡的关键元素：

- **定位标记 (Timing _ Marks)**

- 黑色方块/线条
- 用于定位和校正

- **选项气泡 (Bubbles)**

- 圆形或椭圆形
- 规则排列

- **填涂区域**

- 铅笔/碳素笔填涂
- 深度影响识别

- **导出块**

- 考号填写区
- 需特殊识别

选择题识别的挑战

填涂质量问题

- 填涂深浅不一
- 填涂范围不完整
- 笔迹颜色差异

图像质量

- 扫描/拍摄角度偏差
- 光照不均匀
- 纸张折痕/污损

擦除与修改

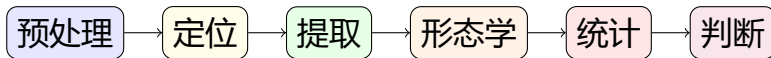
- 擦除不干净的痕迹
- 多次填涂的混乱

多选与漏选

- 多选项填涂（多选题）
- 所有选项未填（空题）

识别流程概述

- ① **图像预处理**: 去噪、二值化、几何校正
- ② **选项区域定位**: 找到每个选项的位置
- ③ **提取选项图像**: 裁剪出单个选项区域
- ④ **形态学处理**: 去除噪点、填充孔洞
- ⑤ **统计像素密度**: 计算深色像素占比
- ⑥ **阈值判断**: 密度超过阈值则认为已填涂



OMR 技术在教育领域的应用

标准化考试:

- 高考: 全国千万考生, 选择题占比 60-70%
- 研究生考试: 政治、英语等科目
- 英语四六级: 每年超过 1800 万考生
- 公务员考试: 行测部分全部为选择题
- 职业资格认证: 医师、会计师、建造师等

在线教育:

- 在线作业与测验
- 智慧课堂互动答题
- 企业培训考核
- 职业技能鉴定

市场规模:

- 中国 OMR 市场年增长率约 15%

答题卡的历史演进

第一代：穿孔卡片 (1930s-1960s)

- IBM 开发，用于人口普查和考试评分
- 需要专用读卡设备
- 速度慢、成本高

第二代：光标阅读卡 (1960s-1990s)

- 使用石墨铅笔填涂
- 光电传感器识别
- 速度快、准确率高

第三代：数字化 OMR (1990s-2010s)

- 扫描仪 + 计算机处理
- 图像处理算法优化
- 成本大幅降低

第四代：智能 OMR (2010s 至今)

答题卡标准化设计原则

设计原则：

- ① **唯一性**：每个填涂位置唯一对应一个选项
- ② **独立性**：各填涂区域互不干扰
- ③ **可检测性**：提供定位标记便于自动识别
- ④ **容错性**：允许一定程度的填涂偏差

常见答题卡格式：

● 横版答题卡

- 左侧为选择题区域
- 右侧为考号填涂区
- 顶部为定位标记

● 竖版答题卡

- 上部为选择题区域
- 下部为姓名考号区

印刷质量要求：

- 定位标记精度： $\pm 0.5\text{mm}$
- 选项气泡位置： $\pm 1\text{mm}$
- 线条粗细： $0.8\text{-}1.2\text{mm}$
- 颜色对比度： $>70\%$

国内外 OMR 系统发展现状

国外主要厂商:

- **Scantron** (美国): 市场领导者, 服务全球教育机构
- **Pearson VUE** (英国): 考试评估综合服务
- **Datacard** (美国): 身份证件 + 答题卡识别

国内主要厂商:

- **海云天**: 教育考试信息化龙头
- **鸥玛**: OMR 设备制造商
- **新开普**: 智慧考试综合解决方案
- **科大讯飞**: AI+OMR 智能阅卷

技术趋势:

- 硬件向软件服务转型
- 传统 OMR 向 AI-OMR 升级
- 离线处理向云端实时处理发展

为什么需要形态学处理？

填涂图像的常见问题：

- 铅笔痕迹不连续（有孔洞）
- 填涂边缘不规则
- 纸张上有噪点/污渍
- 扫描时有颗粒噪声

形态学处理的作用：

- **膨胀**：填充孔洞，连接断点
- **腐蚀**：去除噪点，分离目标
- **开运算**：先腐后胀，去噪保形
- **闭运算**：先胀后腐，填充闭孔

结构元素 (Structuring Element)

结构元素：形态学操作的“探针”，决定操作的范围和形状

常见形状：

- 矩形（最常用）
- 十字形
- 椭圆形

大小选择：

- 太小：效果不明显
- 太大：过度处理
- 推荐：(3,3) 或 (5,5)

OpenCV 创建结构元素

```
kernel = cv2.getStructuringElement(shape, ksize)
```