第 6 周：判断题识别（符号匹配）

怎么看到是 ✓ 还是 ×？

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：

- 判断题识别概述
- 符号特征提取理论
- 模板匹配方法
- 机器学习方法
- Live Coding 实战
- 案例分析

智能阅卷系统进度：

图像采集与预处理
答题卡区域定位
选择题填涂检测
1. **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 判断题识别概述（15min）

00:45 特征提取理论 + 演示（20min）

第 2 学时（45 分钟）：

01:05 模板匹配理论（20min）

01:30 Live Coding：特征提取与匹配（25min）

第 3 学时（45 分钟）：

01:30-01:55 Live Coding：完整识别流程（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：轮廓特征、模板匹配原理（5 分钟视频）

预备知识回顾

上周内容：选择题填涂检测

- OMR 技术：基于像素密度判断填涂状态
- 形态学操作：腐蚀、膨胀、开闭运算
- 连通域分析：findContours 与 contourArea
- 填涂检测：阈值判断 + 位置定位

本周不同之处：

- 选择题：**密度**判断（填涂 vs 空白）
- 判断题：**形状**判断（对号 vs 错号 vs 圆圈）

学习目标

知识目标

- 理解判断题识别与选择题识别的区别
- 掌握轮廓特征的提取与计算方法
- 理解模板匹配的基本原理
- 了解机器学习在符号识别中的应用

能力目标

- 能够使用 OpenCV 提取符号的形状特征
- 能够实现基于特征的符号分类器
- 能够实现模板匹配算法
- 能够构建完整的判断题识别流程

预备知识（课前 5 分钟视频）

相似度度量方法：

- 欧氏距离（Euclidean Distance）
- 余弦相似度（Cosine Similarity）
- 归一化相关系数

轮廓特征提取基础：

- 轮廓查找的基本概念
- 轮廓层级结构
- 基础轮廓特征（面积、周长、长宽比）

观看要求

请在课前观看预备知识视频，为本周学习做好准备

分组策略与角色分工

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	实现轮廓特征、模板匹配	有编程基础的
特征调优者	调整圆度阈值、凸性阈值	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

并行学习路径

观察者路径：

- 理解判断题识别原理
- 看老师演示轮廓特征提取、模板匹配
- 完成基础任务：运行示例代码

使用者路径：

- 使用示例代码处理自己的判断题图像
- 调整圆度阈值、凸性阈值
- 完成核心任务：识别对号和错号

创造者路径：

- 设计自己的符号分类器
- 处理不同手写风格的符号
- 完成挑战任务：实现自适应符号识别

多屏协同设计

双屏协作：

- **主屏**：显示 PPT 理论和讲解
- **侧屏**：实时演示代码运行效果
- **移动设备**：互动答题、查看代码

Live Coding 演示流程：

1. 教师展示问题需求
2. 用 AI 生成代码框架
3. 师生共同完善关键代码
4. 实时运行验证效果

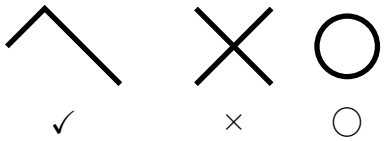
互动方式

使用手机扫码参与实时投票，反馈理解情况

判断题的特点

常见符号类型：

- ✓（对号/正确）
- ×（错号/错误）
- √（根号/正确）
- ○（圆圈/正确）



与选择题的本质区别：

- 选择题：关注**填涂密度**（连续区域）
- 判断题：关注**符号形状**（笔画结构）

判断题识别的应用场景

教育考试

- 标准化考试判断题
- 问卷调查判断题
- 课堂测验快速批改

其他场景

- 表单勾选识别
- 质检合格/不合格标记
- 审批通过/驳回识别

技术挑战

- 符号多样性：不同人书写习惯差异大
- 书写质量：笔画粗细、深浅不一
- 位置偏移：符号位置可能偏离预期
- 模糊符号：擦除修改留下的痕迹

识别方案对比

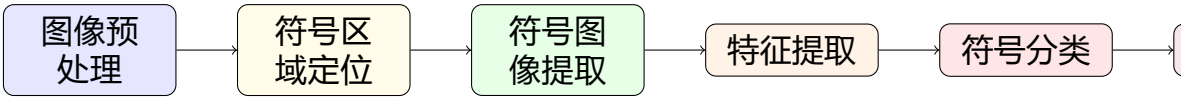
方案	优点	缺点
轮廓特征法	速度快、无需训练	规则复杂、泛化弱
模板匹配法	简单直观、易于实现	对形变敏感、需要模板
机器学习	泛化能力强、准确率高	需要训练数据、计算复杂
深度学习	准确率最高	需要大量数据、资源消耗大

表: 判断题识别方案对比

本周学习策略:

1. 先掌握轮廓特征法（理解形状本质）
2. 再学习模板匹配法（了解经典方法）
3. 了解机器学习方法（开阔技术视野）

判断题识别流程



关键步骤说明：

- **符号区域定位**：找到判断题符号所在位置（类似选择题定位）
- **符号图像提取**：裁剪出单个符号的图像
- **特征提取**：计算符号的形状特征（重点）
- **符号分类**：根据特征判断是 ✓ 还是 ×

为什么选择轮廓特征?

轮廓（Contour）的定义：

- 连接所有连续边界点的曲线
- 具有相同的颜色或强度
- 是形状分析的基础

轮廓特征的优点：

- 直观反映形状的几何特性
- 对光照变化相对鲁棒
- 计算效率高
- OpenCV 提供完善的 API 支持

基础轮廓特征

特征	物理意义	OpenCV 函数
面积（Area）	轮廓所围区域大小	cv2.contourArea()
周长（Perimeter）	轮廓边界长度	cv2.arcLength()
边界矩形	包围轮廓的最小矩形	cv2.boundingRect()
最小外接矩形	旋转的最小矩形	cv2.minAreaRect()
最小外接圆	包围轮廓的最小圆	cv2.minEnclosingCircle()
凸包	包围轮廓的凸多边形	cv2.convexHull()

长宽比与占空比

1. 长宽比（Aspect Ratio）

$$AR = \frac{Width}{Height}$$

- 细长形状：AR 较大或较小
- 方形/圆形：AR 接近 1
- 对号的 AR 通常大于错号的 AR

2. 占空比（Extent）

$$Extent = \frac{ContourArea}{BoundingBoxArea}$$

- 反映轮廓填充边界矩形的程度
- 圆形：接近 $\pi/4 \approx 0.785$
- 稀疏形状：值较小

圆度（Circularity）

定义：

$$C = \frac{4\pi \times Area}{Perimeter^2}$$

物理意义：

- 形状接近圆的程度
- 圆形的圆度 $= 1$（周长 $2\pi r$，面积 πr^2）
- 其他形状的圆度 < 1

判断题符号的圆度特征：

- ◯：接近 1
- ✓：较低（开口形状，周长大）
- ×：更低（两线交叉，周长更大）

三个理解层级：圆度概念

基础概念

- 什么是圆度?
- 圆度的公式是什么?
- 为什么圆度能区分不同符号?

可视化演示

- 对比对号、错号、圆圈的圆度值
- 观察圆度值的差异
- 调整阈值观察分类效果

扩展应用

- 设计自适应圆度阈值
- 处理不同手写风格的符号
- 优化符号分类准确率

理解层级建议：观察者掌握基础概念，使用者完成可视化演示，创造者探索扩展应用

凸性（Convexity）

凸包（Convex Hull）：

- 包围轮廓的最小凸多边形
- 类似" 橡皮筋" 包裹形状

凸性定义：

$$Convexity = \frac{ContourArea}{ConvexHullArea}$$

判断题符号的凸性特征：

- ◯：接近 1（本身就是凸的）
- ✓：明显小于 1（有凹陷）
- ×：接近 1（近似凸的）

关键区分点

凸性可以有效区分对号和错号！

Hu 矩（Hu Moments）

什么是矩?

- 描述图像分布的统计特征
- 类似于物理学中的" 矩"
- 具有旋转、缩放、平移不变性

Hu 矩的特点:

- 7 个不变量
- 对形状变换高度鲁棒
- 适合识别不同角度/大小的符号

OpenCV 使用:

```
moments = cv2.moments(contour)
hu_moments = cv2.HuMoments(moments)
```

特征选择策略

区分对号和错号的特征组合：

圆度优先

- 对号：中等圆度
- 错号：低圆度

凸性辅助

- 对号：明显凹陷
- 错号：近似凸

特征工程建议：

- 计算多个特征，构建特征向量
- 使用决策树或规则组合判断
- 通过实验确定最优特征组合和阈值

模板匹配原理

基本思想：

- 在图像中滑动模板窗口
- 计算每个位置的相似度
- 找到相似度最高的位置

工作流程：

1. 准备标准符号模板图像
2. 将模板在待识别图像上滑动
3. 计算每个位置的匹配分数
4. 选择分数最高的位置和符号类型

相似度度量方法

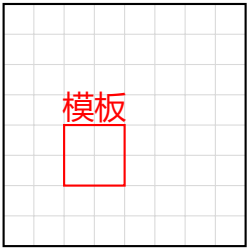
OpenCV 提供 6 种匹配方法：

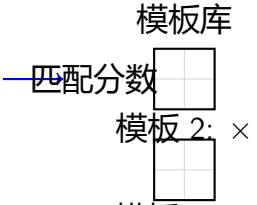
方法	特点
TM_SQDIFF	差值平方和，越小越匹配
TM_SQDIFF_NORMED	归一化差值平方和，范围 [0,1]
TM_CCORR	相关性，越大越匹配
TM_CCORR_NORMED	归一化相关性，范围 [0,1]
TM_CCOEFF	相关系数，范围 [-1,1]
TM_CCOEFF_NORMED	归一化相关系数，范围 [-1,1]

推荐使用

TM_CCOEFF_NORMED：1 表示完美匹配，-1 表示完全不匹配

模板匹配演示





模板匹配的优势与局限

优势：

- 简单直观，易于理解和实现
- 无需训练过程
- 识别速度快

局限性：

- 对尺度变化敏感（需要 resize）
- 对旋转敏感（需要多角度模板）
- 对形变敏感（手写符号变化大）
- 需要准备足够的模板

多尺度模板匹配

问题：手写符号大小不一
解决方案：图像金字塔

- 构建多尺度图像金字塔
- 在每个尺度上进行模板匹配
- 选择所有尺度中最佳匹配结果

实现思路：

1. 生成不同尺度的图像（0.8 倍递减）
2. 对每个尺度执行模板匹配
3. 记录最佳匹配的尺度、位置、分数
4. 返回最高分数对应的符号类型

模板库构建策略

模板采集原则：

- 覆盖不同书写风格
- 包含不同大小和角度
- 数量适中（每类 10-20 个）

模板标准化：

1. 统一尺寸（如 32×32 像素）
2. 统一粗细（形态学处理）
3. 去除噪声（滤波处理）
4. 归一化颜色（转灰度、二值化）

为什么需要机器学习?

特征法的局限:

- 需要手工设计特征和规则
- 对复杂符号难以设计有效规则
- 泛化能力有限

机器学习的优势:

- 自动学习特征组合
- 泛化能力强
- 可以处理更复杂的符号

传统机器学习方法

1. **KNN（K 近邻分类器）**
   - 简单直观
   - 无需训练过程
   - 适合小数据集
2. **SVM（支持向量机）**
   - 适合高维特征
   - 泛化能力强
   - 对小样本效果好
3. **决策树**
   - 可解释性强
   - 类似手工规则的自动化
   - 适合特征工程

深度学习方法

CNN（卷积神经网络）：

- 自动提取特征
- 准确率最高
- 需要大量训练数据

轻量级网络设计：

- LeNet-5：经典小型 CNN
- MobileNet：移动端优化
- 自定义浅层 CNN

实际应用建议

对于判断题识别，传统方法已足够。深度学习适合更复杂的符号识别场景。

方法选择建议

场景	推荐方法	原因
标准印刷符号	模板匹配	简单高效
手写规范符号	轮廓特征 + 规则	特征明显
手写多样符号	SVM/KNN	泛化能力强
复杂手写符号	深度学习	准确率最高

表: 不同场景的方法选择

轮廓相关 API

轮廓查找：

```
contours, hierarchy = cv2.findContours(
    binary_image,
    cv2.RETR_EXTERNAL,  # 只检测外轮廓
    cv2.CHAIN_APPROX_SIMPLE  # 压缩轮廓点
)
```

轮廓特征计算：

```
area = cv2.contourArea(contour)
perimeter = cv2.arcLength(contour, True)
x, y, w, h = cv2.boundingRect(contour)
```

模板匹配 API

单模板匹配：

```
result = cv2.matchTemplate(
    image,           # 待搜索图像
    template,        # 模板图像
    cv2.TM_CCOEFF_NORMED  # 匹配方法
)

min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
```

多模板匹配（阈值过滤）：

```
locations = np.where(result >= threshold)
for pt in zip(*locations[::-1]):
    # 处理每个匹配位置
    pass
```

矩特征 API

计算 Hu 矩:

```
# 计算图像矩
moments = cv2.moments(contour)

# 计算Hu矩（7个不变量）
hu_moments = cv2.HuMoments(moments)

# Hu矩范围差异大，通常取对数
hu_moments = -np.sign(hu_moments) * np.log10(abs(hu_moments))
```

应用场景:

- 形状匹配
- 物体识别
- 姿态估计

AI 辅助编程：本周使用指南

AI 工具推荐

- **Cursor**: 强烈推荐，AI 辅助编程 IDE
- **ChatGPT/Claude**: 通用 AI 编程助手
- **通义灵码**: 国内可用的 AI 编程工具

本周 AI 辅助任务：

- 用 AI 解释圆度和凸性的数学原理
- 用 AI 生成轮廓特征提取代码框架
- 用 AI 调试特征阈值参数
- 用 AI 优化模板匹配性能

Prompt 工程：RTF 框架

RTF 框架示例：

Prompt 模板

[角色] 你是 OpenCV 和图像处理专家
任务
帮我解释什么是 Hu 矩，以及它如何用于符号识别
格式
用通俗语言配合具体示例说明

场景示例：

```
# 示例1：理解概念
请解释圆度的概念，并用简单的比喻帮助我理解
为什么对号的圆度较低而圆圈的圆度较高。

# 示例2：代码生成
请用Python和OpenCV实现计算轮廓的圆度，几何特征，
并解释每个参数的含义。
```

AI 辅助调试：问题诊断

常见问题与 AI 求助方式：

问题：符号识别错误

Prompt 模板： 我的判断题符号识别代码将对号误识别为错号。
运行环境：Windows 11, Python 3.9, OpenCV 4.8
特征值：圆度 =0.65，凸性 =0.85
代码：[粘贴关键代码片段]
请帮我分析问题并提供解决方案。

AI 调试三部曲（回顾 Week 2）：

1. **问题定位**：向 AI 描述现象，请求可能原因
2. **代码分析**：粘贴代码，请求 AI 检查 bug
3. **解决方案**：请 AI 提供修复代码和解释

AI 辅助学习：代码脚手架使用

代码脚手架说明

以下是带有 TODO 标记的代码框架，请使用 AI 助手完成 TODO 部分。

```python
def extract_features(contour):
    """提取轮廓特征"""
    features = {}

    # TODO: 使用AI助手完成以下代码
    # Prompt: 请用Python和OpenCV实现轮廓特征提取

    # TODO: 计算轮廓面积
    # 提示: 使用cv2.contourArea()
    features['area'] = ______

    # TODO: 计算轮廓周长
    # 提示: 使用cv2.arcLength(contour, True)
    features['perimeter'] = ______

    # TODO: 计算圆度
    # 提示: 圆度公式 C = 4π × Area / Perimeter²
```

AI 辅助学习：高级 Prompt 技巧

场景 1：优化模板匹配

Prompt 示例

我的模板匹配识别准确率不高，特别是对于手写差异较大的符号。当前方法：cv2.matchTemplate(roi, template, cv2.TM_CCOEFF_NORMED) 问题：手写对号和标准模板的匹配度较低。请提供优化方案的实现代码。

场景 2：设计自适应阈值

Prompt 示例

我想设计一个自适应阈值系统，能够根据不同的手写风格自动调整圆度和凸性的阈值。请提供设计思路和 Python 实现代码。

AI 辅助编程：特征提取

AI 辅助提示

你可以使用 Cursor、ChatGPT、Claude 等 AI 工具来帮助你实现轮廓特征提取。

推荐 Prompt： 请用 Python 和 OpenCV 实现轮廓特征提取函数，并解释每个特征的含义。

```python
import cv2
import numpy as np

def extract_features(contour):
    """提取轮廓特征"""
    features = {}

    # TODO: 使用AI助手完成以下代码
    # 提示: 使用cv2.contourArea()计算轮廓面积
    features['area'] = cv2.contourArea(contour)

    # TODO: 计算轮廓周长
    # 提示: 使用cv2.arcLength(contour, True), True表示轮廓是闭合的
```

特征可视化

任务：可视化展示特征提取结果

```python
def visualize_features(image, contour, features):
    """可视化特征"""
    result = image.copy()

    # TODO: 绘制轮廓（绿色）
    # 提示: cv2.drawContours(图像, 轮廓列表, 轮廓索引, 颜色, 线宽)
    cv2.drawContours(result, [contour], -1, (0, 255, 0), 2)

    # TODO: 绘制边界矩形（蓝色）
    # 提示: cv2.rectangle(图像, 左上角, 右下角, 颜色, 线宽)
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(result, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # TODO: 绘制凸包（红色）
    # 提示: 先获取凸包，再绘制
    hull = cv2.convexHull(contour)
    cv2.drawContours(result, [hull], -1, (0, 0, 255), 2)

    # TODO: 显示特征值文本
    # 提示: 使用cv2.putText()在图像上添加文字
```

基于特征的分类器

任务：实现基于特征的符号分类器

```python
def classify_by_features(features):
    """基于特征的符号分类"""
    # TODO: 提取特征值
    circularity = features['circularity']
    convexity = features['convexity']

    # TODO: 实现分类规则
    # 规则1: 圆度接近1 -> 圆圈
    if circularity > 0.8:
        return 'circle'

    # TODO: 添加凸性判断规则
    # 提示: 对号的凸性明显小于1（因为有凹陷）
    if convexity < 0.85:
        return 'check'

    # TODO: 添加错号判断规则
    # 提示: 错号的圆度较低
    if circularity < 0.5:
        return 'cross'
```

模板匹配实现

```python
def match_symbol(roi, templates, threshold=0.7):
    """
    使用模板匹配识别符号

    Args:
        roi: 待识别的区域图像
        templates: 模板字典 {类型: 模板图像}
        threshold: 匹配阈值

    Returns:
        (符号类型, 匹配分数)
    """
    best_match = 'unknown'
    best_score = -1

    for symbol_type, template in templates.items():
        # 调整模板大小到ROI大小
        if template.shape[:2] != roi.shape[:2]:
            template = cv2.resize(template, (roi.shape[1], roi.shape[0]))

        # 模板匹配
        result = cv2.matchTemplate(roi, template, cv2.TM_CCOEFF_NORMED)
```

多尺度模板匹配

```python
def multi_scale_match(roi, templates, scales=[0.8, 1.0, 1.2], threshold=0.7):
    """多尺度模板匹配"""
    best_match = 'unknown'
    best_score = -1

    for scale in scales:
        # 缩放ROI
        scaled_roi = cv2.resize(roi, None, fx=scale, fy=scale)

        for symbol_type, template in templates.items():
            # 调整模板大小
            if template.shape[:2] != scaled_roi.shape[:2]:
                template = cv2.resize(template, (scaled_roi.shape[1], scaled_roi.shape
                    [0]))

            # 匹配
            result = cv2.matchTemplate(scaled_roi, template, cv2.TM_CCOEFF_NORMED)
            score = result[0, 0]

            if score > best_score:
                best_score = score
                best_match = symbol_type
```

完整识别流程

```python
def recognize_tf_symbol(image, symbol_position):
    """
    判断题符号识别完整流程

    Args:
        image: 输入图像
        symbol_position: 符号位置 (x, y, w, h)

    Returns:
        符号类型和置信度
    """
    # 1. 提取符号区域
    x, y, w, h = symbol_position
    roi = image[y:y+h, x:x+w]

    # 2. 预处理
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # 3. 查找轮廓
    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

置信度计算

```python
def calculate_confidence(features, symbol_type):
    """计算识别置信度"""
    confidence = 0.0

    if symbol_type == 'check':
        # 对号：高凸性区分度
        confidence = 1.0 - features['convexity']
    elif symbol_type == 'cross':
        # 错号：低圆度区分度
        confidence = 1.0 - features['circularity']
    elif symbol_type == 'circle':
        # 圆圈：圆度接近1
        confidence = features['circularity']

    return max(0, min(1, confidence))

def result_with_confidence(symbol_type, confidence):
    """带置信度的结果输出"""
    if confidence < 0.5:
        return f'{symbol_type}(低置信度:{confidence:.2f})'
    return f'{symbol_type}(置信度:{confidence:.2f})'
```

案例 1：标准符号识别

场景：考试答题卡上的标准打印符号

特点：

- 符号清晰规范
- 位置准确
- 无噪声干扰

推荐方案：轮廓特征法
识别率： 99%+

案例 2：手写符号识别

场景：学生手写的判断题符号

挑战：

- 符号大小不一
- 形状变化较大
- 可能有修改痕迹

推荐方案：

- 优先：多尺度模板匹配
- 备选：SVM 分类器

识别率： 85-95%

案例 3：模糊符号处理

场景：擦除修改后留下的模糊符号

挑战：

- 多个符号重叠
- 符号不完整
- 特征不明显

处理策略：

1. 检测多个轮廓
2. 对每个轮廓分别识别
3. 根据置信度选择结果
4. 低置信度标记为" 人工审核"

识别失败案例分析

常见失败原因

- 符号太轻（阈值问题）
- 符号被遮挡
- 多个符号重叠
- 扫描质量问题

解决方案

- 自适应阈值
- 形态学处理
- 多轮廓检测
- 预处理增强

课堂 Quiz

问题 1：判断题识别和选择题识别的核心区别是什么？

A 处理速度不同

B 密度 vs 形状

C 图像大小不同

D 算法复杂度不同

课堂 Quiz

问题 1：判断题识别和选择题识别的核心区别是什么？

A 处理速度不同

B 密度 vs 形状

C 图像大小不同

D 算法复杂度不同

答案： B

选择题关注填涂密度，判断题关注符号形状

课堂 Quiz

问题 2：圆度（Circularity）公式正确的是?

A $C = \frac{Perimeter^2}{4\pi \times Area}$

B $C = \frac{4\pi \times Area}{Perimeter^2}$

C $C = \frac{Area}{Perimeter}$

D $C = \frac{Perimeter}{Area}$

课堂 Quiz

问题 2：圆度（Circularity）公式正确的是?

A $C = \frac{Perimeter^2}{4\pi \times Area}$

B $C = \frac{4\pi \times Area}{Perimeter^2}$

C $C = \frac{Area}{Perimeter}$

D $C = \frac{Perimeter}{Area}$

答案： B

圆形的圆度 $=1$，其他形状 <1

课堂 Quiz

问题 3：凸性为什么可以区分 ✓ 和 ×？

课堂 Quiz

问题 3：凸性为什么可以区分 ✓ 和 ×？
答案：

- ✓ 有明显凹陷（checkmark 的钩）
- × 近似凸（两直线交叉）
- 凸性 = 轮廓面积/凸包面积
- ✓ 的凸性明显小于 1，× 的凸性接近 1

课堂 Quiz

问题 4：模板匹配的主要缺点是什么？

课堂 Quiz

问题 4：模板匹配的主要缺点是什么？

答案：

- 对尺度变化敏感
- 对旋转敏感
- 对形变敏感（手写差异）
- 需要准备足够多的模板

本周知识要点

判断题识别核心技术：

轮廓特征法

- 面积、周长
- 圆度、凸性
- Hu 矩
- 规则分类

模板匹配法

- 相似度度量
- 多尺度匹配
- 模板库构建
- 阈值判断

完整流程： 预处理 → 定位 → 提取 → 特征 → 分类 → 输出

方法对比总结

方法	准确率	速度	适用场景
轮廓特征	中	快	规范符号
模板匹配	高	中	标准符号
机器学习	高	慢	手写符号

实践建议：

- 先用特征法快速实现
- 效果不好再加模板匹配
- 复杂场景考虑机器学习

课后作业

题目：判断题符号识别模块

实现一个能够识别对号和错号的程序

基础要求（60 分）：

- 实现轮廓特征提取函数（20 分）
- 实现基于规则的分类器（20 分）
- 能够区分标准对号和错号（20 分）

进阶要求（30 分）：

- 支持圆圈识别（10 分）
- 实现模板匹配方法（10 分）
- 输出识别置信度（10 分）

加分项（10 分）：

- 可视化展示识别过程（10 分）

作业提交要求

提交内容：

1. Python 代码（.py 文件）
2. 测试结果截图
3. 简要说明文档

评分标准：

- 代码规范性（20 分）
- 功能完整性（40 分）
- 识别准确率（30 分）
- 文档与展示（10 分）

截止时间：下周上课前

下节预告

第 7 周：OCR 基础与文字识别

故事问题：怎么让机器" 阅读" 文字?

你将学会：

- OCR 技术原理
- PaddleOCR 使用
- 印刷文字识别
- 手写文字识别

实践项目：

- 学号识别
- 姓名识别
- 简答题识别

谢谢！

下节课见！