第 9 周教案：系统架构与分组开发

计算机视觉课程组

1 基本信息

周次	第 9 周
主题	系统架构与分组开发
学时	3 学时（160 分钟）
故事问题	把所有模块组合起来
OBE 目标	A4-系统集成：能将各模块整合为完整系统
项目阶段	架构设计 + 分组开发启动

2 教学目标

1. **知识目标：**
   - 理解软件系统的模块化架构
   - 掌握系统集成的方法
   - 了解项目开发流程
2. **能力目标：**
   - 能够设计系统架构
   - 能够整合各功能模块
   - 能够进行团队协作开发
3. **素养目标：**
   - 培养工程化思维
   - 提升团队协作能力
   - 建立项目管理意识

3 教学重点与难点

教学重点

- 系统架构设计
- 模块接口定义
- 代码集成方法

教学难点

- 模块间的数据流转
- 错误处理与调试

4 教学过程设计

4.1 环节一：项目回顾与架构设计（40 分钟）

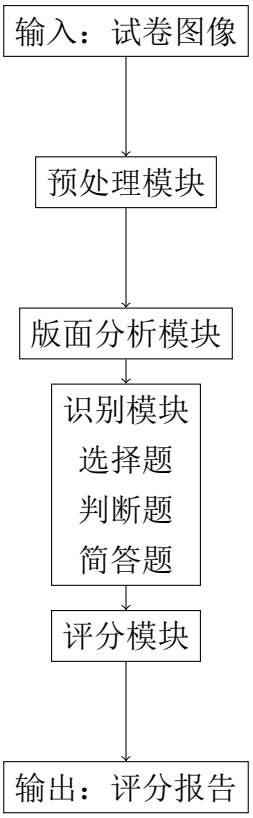
4.1.1 1.1 已学模块回顾（15 分钟）

我们已实现的模块：

周次	模块	功能
第 1-2 周	基础工具	OpenCV 基础、AI 辅助编程
第 3 周	图像预处理	去噪、二值化、透视矫正
第 4 周	版面分析	边缘检测、轮廓检测、区域定位
第 5 周	选择题识别	填涂检测、OMR 识别
第 6 周	判断题识别	符号匹配、形状识别
第 7-8 周	文字识别	OCR 印刷识别、手写识别

4.1.2 1.2 系统架构设计（25 分钟）

自动阅卷系统架构：



模块接口设计：

模块	输入	输出	接口
预处理	原始图像	预处理图像	`preprocess(img)`
版面分析	预处理图像	区域坐标	`analyze(img)`
选择题	选项区域图像	答案 (A/B/C/D)	`recognize_choice(roi)`
判断题	符号区域图像	答案 (T/F)	`recognize_judge(roi)`
简答题	答题区域图像	文字内容	`recognize_essay(roi)`
评分	所有答案	分数报告	`grade(answers)`

4.2 环节二：项目框架说明（40 分钟）

4.2.1 2.1 代码结构（15 分钟）

```
auto_grading_system/
    README.md                  # 项目说明
    requirements.txt           # 依赖包
    config.py                  # 配置文件
    main.py                    # 主程序入口
    modules/                   # 功能模块
        __init__.py
```

```
        preprocess.py          # 预处理模块（已实现）
        layout.py              # 版面分析模块（已实现）
        choice_recognizer.py   # 选择题识别（学生实现）
        judge_recognizer.py    # 判断题识别（学生实现）
        essay_recognizer.py    # 简答题识别（学生实现）
        grading.py             # 评分模块（学生实现）
    utils/                     # 工具函数
        __init__.py
        visualization.py       # 可视化工具
        logger.py              # 日志工具
    data/                      # 数据目录
        input/                 # 输入试卷
        output/                # 输出结果
        templates/             # 标准答案模板
        test/                  # 测试数据
    tests/                     # 单元测试
        test_preprocess.py
        test_choice.py
        test_judge.py
        test_essay.py
```

4.2.2 2.2 已实现模块说明（10 分钟）

预处理模块（modules/preprocess.py）：

```python
"""
图像预处理模块
已实现：去噪、二值化、透视矫正
"""

import cv2
import numpy as np

class Preprocessor:
    """图像预处理器"""

    def __init__(self, config=None):
        self.config = config or {}

    def denoise(self, image):
```

```
        """去噪"""
        return cv2.medianBlur(image, 5)

    def binarize(self, image, method='otsu'):
        """二值化"""
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) if len(image.
            shape) == 3 else image

        if method == 'otsu':
            _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY
                + cv2.THRESH_OTSU)
        else:
            _, binary = cv2.threshold(gray, 127, 255, cv2.
                THRESH_BINARY)

        return binary

    def process(self, image):
        """完整预处理流程"""
        # 去噪
        denoised = self.denoise(image)

        # 二值化
        binary = self.binarize(denoised)

        return {
            'gray': cv2.cvtColor(denoised, cv2.COLOR_BGR2GRAY) if len
                (denoised.shape) == 3 else denoised,
            'binary': binary,
            'denoised': denoised
        }
```

版面分析模块（modules/layout.py）：

```
"""
版面分析模块
已实现：检测试卷边界、定位题型区域
"""

import cv2
import numpy as np
```

```python
class LayoutAnalyzer:
    """版面分析器"""

    def __init__(self, config=None):
        self.config = config or {}

    def detect_paper_boundary(self, binary_image):
        """检测试卷边界"""
        contours, _ = cv2.findContours(
            binary_image,
            cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE
        )

        # 找到最大的轮廓（假设是试卷）
        if contours:
            paper_contour = max(contours, key=cv2.contourArea)
            return cv2.boundingRect(paper_contour)

        return None

    def locate_question_areas(self, binary_image):
        """
        定位题目区域

        返回: {
            'choice': [(x,y,w,h), ...],  # 选择题区域
            'judge': [(x,y,w,h), ...],   # 判断题区域
            'essay': [(x,y,w,h), ...]    # 简答题区域
        }
        """
        # 简化实现: 基于投影法的区域分割
        # 学生需要根据实际试卷版面调整

        h, w = binary_image.shape

        # 示例: 固定位置分区
        # 实际应该通过版面分析自动检测
        return {
```

```
            'choice': [(50, 200, w-100, 300)],     # 选择题区域
            'judge': [(50, 520, w-100, 200)],      # 判断题区域
            'essay': [(50, 740, w-100, 300)]       # 简答题区域
        }

    def analyze(self, image):
        """完整版面分析"""
        # 转换为二值图
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) if len(image.
            shape) == 3 else image
        _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY +
            cv2.THRESH_OTSU)

        # 检测边界
        boundary = self.detect_paper_boundary(binary)

        # 定位区域
        areas = self.locate_question_areas(binary)

        return {
            'boundary': boundary,
            'areas': areas,
            'binary': binary
        }
```

4.2.3 2.3 待实现模块说明（15 分钟）

选择题识别模块模板：

```
"""
选择题识别模块
学生需要实现填涂检测和答案识别
"""

import cv2
import numpy as np

class ChoiceRecognizer:
    """选择题识别器"""
```

```
    def __init__(self, config=None):
        self.config = config or {}
        self.threshold = self.config.get('threshold', 0.3)

    def calculate_density(self, roi):
        """
        计算填涂密度

        TODO: 学生需要实现
        """
        pass

    def recognize_question(self, question_img, option_positions):
        """
        识别单道选择题

        TODO: 学生需要实现

        参数:
            question_img: 题目图像
            option_positions: 选项位置列表 [(x,y,w,h), ...]

        返回:
            答案 ('A', 'B', 'C', 'D' 或 None)
        """
        pass

    def recognize_all(self, image, choice_areas):
        """
        识别所有选择题

        TODO: 学生需要实现

        参数:
            image: 完整图像
            choice_areas: 选择题区域列表

        返回:
            答案列表 [{'question': 1, 'answer': 'A'}, ...]
        """
```

```
        pass
```

4.3 环节三：分组与任务分配（30 分钟）

4.3.1 3.1 分组原则（10 分钟）

分组建议：

- 每组 3-4 人
- 包含不同专业背景
- 设立组长和技术负责人

角色分工：

角色	职责	适合学生
组长	统筹协调、进度管理、对外沟通	组织能力强的
技术负责人	架构设计、核心算法、代码审查	CS/EE 专业
模块开发 A	选择题 + 判断题模块实现	有编程基础的
模块开发 B	简答题 + 评分模块实现	有编程基础的
前端/测试	界面展示、测试用例、文档编写	设计/文科专业

4.3.2 3.2 任务分解（20 分钟）

开发任务清单：

任务	优先级	预计工时
搭建开发环境	高	1h
实现选择题识别	高	4h
实现判断题识别	高	3h
实现简答题识别	中	4h
实现评分模块	中	2h
模块集成测试	高	2h
界面开发	低	2h
文档编写	中	2h

4.4 环节四：开发指导（40 分钟）

4.4.1 4.1 开发环境搭建（10 分钟）

```
# 1. 克隆/创建项目
mkdir auto_grading_system
cd auto_grading_system

# 2. 创建虚拟环境
python -m venv venv
source venv/bin/activate  # Linux/Mac
# 或 venv\Scripts\activate  # Windows

# 3. 安装依赖
pip install opencv-python
pip install paddlepaddle
pip install paddleocr
pip install numpy pillow matplotlib

# 4. 创建项目结构
mkdir -p modules utils data/{input,output,templates,test} tests
```

4.4.2 4.2 开发工作流（10 分钟）

推荐开发流程：

1. **搭建框架**
   - 创建项目结构
   - 配置开发环境
   - 编写基础代码框架
2. **模块开发**
   - 选择一个模块开始（建议从选择题开始）
   - 编写单元测试
   - 调试优化
3. **模块集成**
   - 整合各模块

- 测试整体流程
- 修复接口问题

4. **测试优化**

- 使用测试集验证
- 优化识别准确率
- 完善错误处理

4.4.3 4.3 调试技巧（10 分钟）

```
"""
调试工具函数
"""

import cv2
import numpy as np
from datetime import datetime

def save_debug_image(image, name, prefix='debug'):
    """保存调试图像"""
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    filename = f"data/output/{prefix}_{name}_{timestamp}.jpg"
    cv2.imwrite(filename, image)
    print(f"Debug image saved: {filename}")

def draw_debug_boxes(image, boxes, labels=None, color=(0, 255, 0)):
    """绘制调试框"""
    debug_img = image.copy()

    for i, box in enumerate(boxes):
        x, y, w, h = box
        cv2.rectangle(debug_img, (x, y), (x+w, y+h), color, 2)

        if labels and i < len(labels):
            cv2.putText(debug_img, labels[i], (x, y-5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

    return debug_img
```

```python
def print_debug_info(info_dict, title="Debug Info"):
    """打印调试信息"""
    print(f"\n{'='*50}")
    print(f"{title}")
    print(f"{'='*50}")
    for key, value in info_dict.items():
        print(f"{key}: {value}")
    print(f"{'='*50}\n")
```

4.4.4 4.4 常见问题与解决方案（10 分钟）

Q1: 模块导入错误

```python
# 错误: ModuleNotFoundError: No module named 'modules'

# 解决方案: 确保项目根目录在Python路径中
import sys
sys.path.append('.')  # 或项目的绝对路径

from modules.choice_recognizer import ChoiceRecognizer
```

Q2: 图像路径问题

```python
import os

# 使用相对路径时, 注意当前工作目录
base_dir = os.path.dirname(os.path.abspath(__file__))
image_path = os.path.join(base_dir, 'data', 'input', 'exam.jpg')

img = cv2.imread(image_path)
if img is None:
    print(f"Error: Cannot load image from {image_path}")
```

Q3: 数组索引越界

```python
# 安全地裁剪ROI
def safe_crop(image, x, y, w, h):
    """安全裁剪, 防止越界"""
    img_h, img_w = image.shape[:2]

    # 确保坐标在图像范围内
    x = max(0, min(x, img_w - 1))
```