

第 2 周教案：AI 辅助编程工具实战

计算机视觉课程组

1 基本信息

周次	第 2 周
主题	AI 辅助编程工具实战
学时	3 学时（160 分钟）
故事问题	怎么让 AI 帮我写代码？
OBE 目标	A5-AI 协作：能用 AI 工具辅助编程开发

2 教学目标

1. 知识目标：

- 了解主流 AI 编程工具（ChatGPT、Claude、Copilot 等）
- 理解 Prompt 工程的基本原理

2. 能力目标：

- 能够编写有效的 CV 领域 Prompt
- 能够用 AI 辅助调试和优化代码
- 能够利用 AI 快速实现 CV 功能原型

3. 素养目标：

- 建立 AI 协作意识，提高学习效率
- 培养批判性思维，验证 AI 生成代码的正确性

3 教学重点与难点

教学重点

- Prompt 工程：如何向 AI 提问 CV 问题
- AI 辅助代码调试技巧
- AI 生成代码的验证与优化

教学难点

- 编写精准有效的 Prompt
- 判断 AI 生成代码的正确性
- 将 AI 生成的代码整合到项目中

4 教学过程设计

4.1 环节一：AI 编程工具介绍（30 分钟）

4.1.1 1.1 为什么需要 AI 辅助编程？（10 分钟）

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

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

4.1.2 1.2 主流 AI 工具对比（15 分钟）

工具选择建议：

- 学习理解：ChatGPT/Claude
- 实时编码：GitHub Copilot/Cursor
- 国内使用：通义千问/DeepSeek

工具	特点	适用场景
ChatGPT	对话能力强，代码生成准确	学习、调试、解释
Claude	代码分析深入，长文本处理好	代码审查、架构设计
GitHub Copilot	IDE 集成，实时补全	日常编码
通义千问/文心一言	中文友好，国内可用	中文问题咨询

4.1.3 1.3 演示：AI 工具的基本使用（5 分钟）

现场演示：用 ChatGPT/Claude 生成一个简单的 OpenCV 代码

4.2 环节二：Prompt 工程实战（40 分钟）

4.2.1 2.1 什么是 Prompt？（5 分钟）

定义：Prompt 是给 AI 的指令或提示词

好 Prompt 的标准：

1. 具体明确：不说模糊的话
2. 有上下文：提供足够的背景信息
3. 有约束：明确输出格式要求

4.2.2 2.2 CV 领域专用 Prompt 模板（30 分钟）

模板 1：代码生成

请用 Python 和 OpenCV 实现以下功能：

功能描述：[详细描述要实现的功能]

输入：[描述输入数据格式]

输出：[描述期望的输出格式]

要求：

- 使用 OpenCV 库
- 代码有详细注释
- 包含使用示例

示例：

请用 Python 和 OpenCV 实现以下功能：

功能描述：将彩色图像转换为灰度图像

输入：一张 JPG 格式的彩色图像文件

输出：灰度图像，并保存为PNG格式

要求：

- 使用OpenCV库
- 代码有详细注释
- 包含使用示例

模板 2：代码解释

请解释以下 OpenCV 代码的含义：

代码：

[粘贴代码]

要求：

- 逐行解释代码功能
- 说明关键参数的作用
- 指出可能的错误用法

模板 3：调试求助

我的 OpenCV 代码运行出错，请帮我分析：

代码：

[粘贴代码]

错误信息：

[粘贴报错信息]

预期行为：[描述期望的结果]

实际行为：[描述实际发生的情况]

模板 4：性能优化

如何优化以下 OpenCV 代码的性能？

当前实现：

[粘贴代码]

问题：[描述性能瓶颈]

目标：[期望达到的性能指标]

4.2.3 2.3 互动练习（5 分钟）

练习：给学生一个 OpenCV 代码片段，让学生尝试编写 Prompt 让 AI 解释

4.3 环节三：AI 辅助实战——人脸检测（50 分钟）

4.3.1 3.1 任务说明（5 分钟）

任务目标：用 AI 辅助实现一个人脸检测程序

实现步骤：

1. 用 Prompt 生成人脸检测代码
2. 理解 AI 生成的代码
3. 运行并测试代码
4. 优化和改进

4.3.2 3.2 步骤 1：生成代码（10 分钟）

Prompt 示例：

请用Python和OpenCV实现一个人脸检测程序：
功能：从图片中检测所有人脸，并用矩形框标注
输入：图片文件路径
输出：标注了人脸框的图片
要求：
- 使用OpenCV的Haar级联分类器
- 在每个人脸周围绘制绿色矩形框
- 显示检测到的人脸数量
- 代码有详细中文注释

预期 AI 输出：

```
import cv2

def detect_faces(image_path):
    """
    检测图像中的人脸并标注

    参数：
        image_path: 输入图像路径

    返回：
        标注后的图像
    """
    # 加载预训练的人脸检测模型
```

```

face_cascade = cv2.CascadeClassifier(
    cv2.data.harcascades + 'haarcascade_frontalface_default.xml'
)

# 读取图像
img = cv2.imread(image_path)
if img is None:
    print(f"错误：无法读取图像 {image_path}")
    return None

# 转换为灰度图（Haar分类器需要灰度输入）
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 检测人脸
faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor=1.1,      # 图像缩放比例
    minNeighbors=5,       # 每个检测框的最小邻居数
    minSize=(30, 30)      # 最小人脸尺寸
)

# 绘制检测框
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

print(f"检测到 {len(faces)} 个人脸")

return img

# 使用示例
result = detect_faces('test.jpg')
if result is not None:
    cv2.imwrite('faces_detected.jpg', result)
    cv2.imshow('Faces', result)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

4.3.3 3.3 步骤 2：代码理解（15 分钟）

关键概念解析：

1. Haar 级联分类器

- 一种基于机器学习的目标检测方法
- OpenCV 提供预训练模型文件 (.xml)

2. detectMultiScale 参数

- scaleFactor: 图像金字塔缩放比例
- minNeighbors: 检测框合并的阈值
- minSize: 最小目标尺寸

学生活动:

- 运行代码, 观察效果
- 修改参数, 观察检测效果变化
- 用自己的照片测试

4.3.4 3.4 步骤 3: 调试与优化 (20 分钟)

常见问题与 AI 辅助解决:

问题 1: 检测不到人脸

我的代码检测不到侧脸, 请帮我分析:
[粘贴代码]

AI 建议:

- 调整 scaleFactor 和 minNeighbors 参数
- 尝试使用更先进的模型 (如 DNN 人脸检测)

问题 2: 误检测 (非人脸被识别成人脸)

我的代码把背景中的圆形也识别成人脸了:
[粘贴代码]

AI 建议:

- 增加 minNeighbors 值
- 增加 minSize, 过滤小目标
- 使用眼睛检测作为二次验证

学生活动:

1. 记录遇到的问题
2. 用 AI 工具寻求解决方案
3. 验证 AI 建议的有效性

4.4 环节四：代码调试技巧（30 分钟）

4.4.1 4.1 AI 辅助调试流程（10 分钟）

1. 复制错误信息
 - 完整复制报错的 traceback
 - 注意错误类型和行号
2. 准备上下文
 - 粘贴相关代码片段
 - 说明输入数据
3. 向 AI 提问
 - 使用模板化 Prompt
 - 明确预期 vs 实际行为
4. 验证解决方案
 - 不要盲目信任 AI
 - 理解修改原理
 - 测试修改效果

4.4.2 4.2 常见错误案例（15 分钟）

案例 1：图片路径错误

```
# 错误代码
img = cv2.imread('test.jpg') # 返回 None

# AI 诊断：路径不存在或格式错误
# 解决方案：
import os
path = 'test.jpg'
if os.path.exists(path):
```



```
img = cv2.imread(path)
else:
    print(f"文件不存在: {path}")
```

案例 2：颜色顺序混乱

```
# 错误：OpenCV是BGR，matplotlib是RGB
plt.imshow(img) # 颜色异常

# AI诊断：颜色空间不匹配
# 解决方案：
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

案例 3：数据类型溢出

```
# 错误：uint8溢出
result = img + 50 # 大于255的值会循环

# AI诊断：uint8范围是0-255
# 解决方案：
result = cv2.add(img, 50) # 自动截断
# 或者
result = np.clip(img.astype(int) + 50, 0, 255).astype(np.uint8)
```

4.4.3 4.3 练习（5 分钟）

给学生一段有 bug 的代码，让学生用 AI 辅助找出并修复 bug

4.5 环节五：AI 工具使用规范（10 分钟）

4.5.1 5.1 使用原则

1. 验证优先

- AI 生成代码必须测试
- 理解代码再使用

2. 学习导向

- 不仅问”怎么写”
- 更要问”为什么”

3. 安全意识

- 不要输入敏感信息
- 注意代码安全性

4.5.2 5.2 最佳实践

- 从简单 Prompt 开始，逐步迭代优化
- 保持对话的连贯性
- 保存有用的对话记录
- 建立个人 Prompt 库

5 课后作业

5.1 作业内容

题目：用 AI 辅助实现答题卡边界检测程序

项目关联：这是自动阅卷系统的第一步——定位答题卡在图像中的位置。通过完成此作业，学生将为后续的版面分析和题目识别模块奠定基础。

要求：

1. 用 Prompt 向 AI 询问题卡边界检测的实现方法
2. 记录完整的 AI 对话过程（至少 3 轮交互）
3. 运行并测试代码（可使用试卷图片）
4. 撰写反思报告

5.2 提交内容

1. AI 对话记录（截图或复制文本）
2. 最终代码
3. 测试结果图片（标注出检测到的边界）
4. 反思报告（包含以下内容）
 - AI 给你的帮助有哪些？
 - 你遇到的问题如何解决的？

- AI 生成的代码有哪些需要改进的地方？
- 此功能对自动阅卷系统的作用是什么？

5.3 评分标准

评分项	标准	分值
AI 对话质量	Prompt 清晰有效，交互充分	20 分
代码实现	能正确检测答题卡边界	30 分
测试完整	有多场景测试记录，可视化结果清晰	20 分
反思深度	深入思考 AI 辅助的利与弊，理解项目关联	30 分
合计		100 分

6 教学反思

6.1 预期挑战

- 学生可能完全依赖 AI，不理解代码
 - 应对：设置代码理解考核环节
- AI 工具访问问题（网络限制）
 - 应对：推荐国内可用工具，准备备用方案
- Prompt 编写能力差异大
 - 应对：提供模板库，多练习