第 3 周教案：图像预处理与增强

计算机视觉课程组

1 基本信息

周次	第 3 周
主题	图像预处理与增强
学时	3 学时（160 分钟）
故事问题	试卷拍照模糊怎么办？
OBE 目标	A1-图像处理：能对图像进行去噪、二值化、矫正操作
项目贡献	为所有题型识别提供清晰、规范的输入图像

2 教学目标

1. **知识目标：**
   - 理解图像噪声的类型与来源
   - 掌握图像滤波的基本原理
   - 理解二值化与阈值分割的原理
2. **能力目标：**
   - 能够实现图像去噪（高斯/中值滤波）
   - 能够实现图像二值化
   - 能够进行透视矫正
3. **素养目标：**
   - 理解预处理对后续识别的重要性
   - 培养参数调优的实验思维

3 教学重点与难点

教学重点

- 图像滤波：高斯滤波、中值滤波
- 图像二值化：全局阈值、自适应阈值
- 透视变换：四点变换矫正

教学难点

- 阈值的选择与优化
- 透视变换矩阵的理解

4 教学过程设计

4.1 环节一：图像预处理概述（10 分钟）

4.1.1 1.1 为什么需要预处理？（5 分钟）

现实问题：

- 拍摄角度不正
- 光照不均匀
- 纸张有折痕
- 背景有杂物

预处理的目标：

1. 去除噪声干扰
2. 增强目标特征
3. 规范图像格式
4. 提升识别准确率

4.1.2 1.2 预处理流水线（5 分钟）

$$\text{原图} \rightarrow \text{去噪} \rightarrow \text{二值化} \rightarrow \text{矫正} \rightarrow \text{增强输出}$$

4.2 环节二：图像去噪（40 分钟）

4.2.1 2.1 噪声类型（10 分钟）

常见噪声类型：

噪声类型	特征	典型场景
高斯噪声	随机分布的亮度变化	传感器噪声、低光拍摄
椒盐噪声	随机的黑点或白点	传输错误、老化的传感器
周期噪声	规则的干扰条纹	电气干扰

4.2.2 2.2 滤波方法（20 分钟）

1. 高斯滤波（Gaussian Blur）

- 原理：用高斯分布权重进行卷积
- 适合：高斯噪声、自然去噪

```
import cv2
import numpy as np

# 高斯滤波
# 参数：(图像, 核大小, 标准差)
blur = cv2.GaussianBlur(img, (5, 5), 0)

# 核大小必须是正奇数：3, 5, 7, 9...
# 核越大，模糊效果越强
# 标准差为0时自动计算
```

2. 中值滤波（Median Blur）

- 原理：用邻域像素的中值替换中心像素
- 适合：椒盐噪声、保持边缘

```
# 中值滤波
# 参数：(图像, 核大小)
median = cv2.medianBlur(img, 5)

# 中值滤波对椒盐噪声效果更好
# 能更好地保持边缘清晰
```

3. 双边滤波（Bilateral Filter）

- 原理：同时考虑空间距离和像素值差异
- 适合：保边去噪

```
# 双边滤波
# 参数：(图像, d, sigmaColor, sigmaSpace)
bilateral = cv2.bilateralFilter(img, 9, 75, 75)

# d: 像素邻域直径
# sigmaColor: 颜色空间的标准差
# sigmaSpace: 坐标空间的标准差
```

4.2.3 2.3 效果对比实验（10 分钟）

```
import cv2
import matplotlib.pyplot as plt

# 读取带噪声的图像
img = cv2.imread('noisy_exam.jpg')

# 应用不同滤波方法
gaussian = cv2.GaussianBlur(img, (5, 5), 0)
median = cv2.medianBlur(img, 5)
bilateral = cv2.bilateralFilter(img, 9, 75, 75)

# 可视化对比
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes[0, 0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axes[0, 0].set_title('原始噪声图像')
axes[0, 0].axis('off')

axes[0, 1].imshow(cv2.cvtColor(gaussian, cv2.COLOR_BGR2RGB))
axes[0, 1].set_title('高斯滤波')
axes[0, 1].axis('off')

axes[1, 0].imshow(cv2.cvtColor(median, cv2.COLOR_BGR2RGB))
axes[1, 0].set_title('中值滤波')
```

```
axes[1, 0].axis('off')

axes[1, 1].imshow(cv2.cvtColor(bilateral, cv2.COLOR_BGR2RGB))
axes[1, 1].set_title('双边滤波')
axes[1, 1].axis('off')

plt.tight_layout()
plt.show()
```

4.3 环节三：图像二值化（50 分钟）

4.3.1 3.1 什么是二值化？（10 分钟）

定义：将灰度图像转换为只有黑白两种颜色的图像

- 像素值 $>$ 阈值 $\rightarrow$ 白色（255）
- 像素值 $\leq$ 阈值 $\rightarrow$ 黑色（0）

应用场景：

- 文档扫描处理
- 填涂识别（OMR）
- 边缘检测预处理

4.3.2 3.2 全局阈值（15 分钟）

方法：对整幅图像使用同一个阈值

```
# 先转灰度图
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 全局阈值
# 参数：(图像, 阈值, 最大值, 阈值类型)
ret, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# ret: 实际使用的阈值
# binary: 二值化结果
```

阈值类型：

选择题识别的常用设置：

类型	公式	效果
THRESH_BINARY	$dst(x,y) = maxval$ if $src(x,y) > thresh$	大于阈值变白
THRESH_BINARY_INV	$dst(x,y) = 0$ if $src(x,y) > thresh$	大于阈值变黑（反色）
THRESH_TRUNC	$dst(x,y) = threshold$ if $src(x,y) > thresh$	超过阈值截断
THRESH_TOZERO	$dst(x,y) = src(x,y)$ if $src(x,y) > thresh$	小于阈值变 0

```
# 填涂识别：使用反色二值化
# 填涂部分（深色）变为白色，未填涂部分（浅色）变为黑色
ret, binary_inv = cv2.threshold(
    gray,
    127,            # 阈值
    255,            # 最大值
    cv2.THRESH_BINARY_INV
)
```

4.3.3 3.3 自适应阈值（15 分钟）

问题：全局阈值对光照不均的图像效果差

解决方案：自适应阈值——根据局部区域计算阈值

```
# 自适应阈值
# 参数：(图像，最大值，方法，阈值类型，邻域大小，常数C)
adaptive = cv2.adaptiveThreshold(
    gray, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  # 方法：高斯加权平均
    cv2.THRESH_BINARY,                # 阈值类型
    11,                               # 邻域大小（奇数）
    2                                 # 常数C（从均值减去）
)
```

两种方法对比：

方法	特点	适用场景
ADAPTIVE_THRESH_MEAN_C	邻域均值计算阈值	计算快
ADAPTIVE_THRESH_GAUSSIAN_C	高斯加权计算阈值	效果好，推荐

4.3.4 3.4 Otsu 自动阈值（10 分钟）

原理：自动寻找最佳阈值，使类间方差最大

```
# Otsu自动阈值
# 阈值设为0，添加cv2.THRESH_OTSU标志
ret, otsu = cv2.threshold(
    gray,
    0,                                  # 阈值（会被忽略）
    255,
    cv2.THRESH_BINARY + cv2.THRESH_OTSU
)

print(f"Otsu自动选择的阈值: {ret}")
```

4.4 环节四：透视矫正（40 分钟）

4.4.1 4.1 透视变换原理（10 分钟）

问题：拍照时试卷角度不正
解决：将梯形区域变换为矩形

歪斜的试卷 → 矫正后的标准矩形

四点变换：

1. 检测试卷的四个角点
2. 定义目标矩形的四个角点
3. 计算透视变换矩阵
4. 应用变换得到矫正图像

4.4.2 4.2 实现代码（20 分钟）

```
import cv2
import numpy as np

def four_point_transform(image, pts):
    """
    四点透视变换

    参数:
        image: 输入图像
```

```
        pts: 四个角点坐标 [tl, tr, br, bl]

    返回:
        变换后的图像
    """
    # 获取四个点坐标
    rect = order_points(pts)
    (tl, tr, br, bl) = rect

    # 计算新图像的宽度
    widthA = np.sqrt(((br[0] - bl[0])**2) + ((br[1] - bl[1])**2))
    widthB = np.sqrt(((tr[0] - tl[0])**2) + ((tr[1] - tl[1])**2))
    maxWidth = max(int(widthA), int(widthB))

    # 计算新图像的高度
    heightA = np.sqrt(((tr[0] - br[0])**2) + ((tr[1] - br[1])**2))
    heightB = np.sqrt(((tl[0] - bl[0])**2) + ((tl[1] - bl[1])**2))
    maxHeight = max(int(heightA), int(heightB))

    # 目标点（矩形）
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]
    ], dtype="float32")

    # 计算透视变换矩阵
    M = cv2.getPerspectiveTransform(rect, dst)

    # 应用变换
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))

    return warped

def order_points(pts):
    """对四个点排序: 左上、右上、右下、左下"""
    rect = np.zeros((4, 2), dtype="float32")

    # 计算点的和（左上和最小，右下和最大）
```

```
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]

    # 计算点的差（右上差最小，左下差最大）
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]

    return rect
```

4.4.3 4.3 完整示例（10 分钟）

```
# 读取图像
img = cv2.imread('skewed_exam.jpg')

# 假设已经检测到四个角点（实际需要用轮廓检测）
pts = np.array([
    [100, 150],   # 左上
    [450, 120],   # 右上
    [480, 380],   # 右下
    [80, 400]     # 左下
], dtype="float32")

# 应用透视变换
warped = four_point_transform(img, pts)

# 显示结果
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

axes[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axes[0].set_title('原始倾斜图像')
axes[0].axis('off')

axes[1].imshow(cv2.cvtColor(warped, cv2.COLOR_BGR2RGB))
axes[1].set_title('透视矫正后')
axes[1].axis('off')

plt.show()
```

4.5 环节五：综合实验（20 分钟）

任务：实现完整的试卷预处理流水线

```
def preprocess_exam_paper(image_path):
    """
    试卷图像预处理流水线

    参数:
        image_path: 试卷图像路径

    返回:
        预处理后的图像
    """
    # 1. 读取图像
    img = cv2.imread(image_path)

    # 2. 去噪
    denoised = cv2.medianBlur(img, 5)

    # 3. 转灰度
    gray = cv2.cvtColor(denoised, cv2.COLOR_BGR2GRAY)

    # 4. 二值化（用于后续处理）
    _, binary = cv2.threshold(
        gray,
        127,
        255,
        cv2.THRESH_BINARY
    )

    # 5. 透视矫正（需要先检测角点）
    # warped = perspective_transform(binary)

    return gray, binary

# 使用示例
gray, binary = preprocess_exam_paper('exam.jpg')
```

5 课后作业

5.1 作业内容

题目：实现试卷图像预处理完整流程

要求：

1. 对试卷图像进行去噪处理
2. 实现二值化（至少两种方法对比）
3. 提交处理前后对比图
4. 分析不同参数对结果的影响

5.2 评分标准

评分项	标准	分值
代码实现	功能完整，无错误	40 分
效果对比	清晰展示处理前后差异	30 分
参数分析	分析参数对结果的影响	20 分
代码规范	注释完整，结构清晰	10 分
合计		**100 分**

6 教学反思

6.1 重点提醒

- 预处理是后续识别的基础，必须重视
- 阈值选择需要根据具体图像调整
- 透视变换需要准确检测四个角点（下周内容）