计算机视觉实验

中山大学智能工程学院　陈俊周、梁小丹

2025年9月

第四讲　图像重定向算法实践

视觉显著性（Visual Saliency）

定义

图像中能够自动吸引人类视觉注意力的区域，可用于模拟人类视觉系统的选择性注意机制

作用

传统缩放：所有像素同等重要 → 内容变形

智能重定向：保护重要区域 → 内容感知的缩放

- 避开显著区域删除像素线
- 保留图像的视觉重点
- 重要区域少变形，背景多变形

颜色对比度显著性检测（Color Contrast Saliency）

假设　如果某个区域的颜色与整幅图像的平均颜色差异很大，那么这个区域就是显著的。

显著性定义　像素颜色与平均颜色的欧氏距离

优势

计算简单快速，对颜色鲜明的物体效果好

缺点

忽略空间位置信息，大面积同色物体可能被忽略

```python
def color_contrast_saliency(self, image):  1 usage
    """基于颜色对比度的显著性检测"""
    # 转换到LAB色彩空间
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
    # 计算图像的平均颜色
    mean_lab = np.mean(lab, axis=(0, 1))
    # 计算每个像素与平均颜色的欧氏距离
    saliency = np.sqrt(np.sum((lab - mean_lab) ** 2, axis=2))
    # 归一化到[0, 255]
    saliency = (saliency - saliency.min()) / (saliency.max() - saliency.min())
    saliency = (saliency * 255).astype(np.uint8)

    return saliency
```

中心先验显著性检测（Center Prior Saliency）

假设　图像中某个位置的显著性与该位置到图像中心的距离成反比关系

原理　使用二维高斯分布建模：$S(x,y) = \exp(-D^2(x,y)/(2\sigma^2))$，其中D为该位置到图像中心的欧氏距离，σ为标准差，控制衰减速度

优势

符合人眼注视习惯，计算极快

缺点

对边缘物体判断失误，过于依赖位置假设

```python
def center_prior_saliency(self, image):  1 usage
    """基于中心先验的显著性检测"""
    h, w = image.shape[:2]
    # 创建中心先验图
    center_x, center_y = w // 2, h // 2
    Y, X = np.ogrid[:h, :w]
    # 计算到中心的距离
    dist_from_center = np.sqrt((X - center_x) ** 2 + (Y - center_y) ** 2)
    max_dist = np.sqrt(center_x ** 2 + center_y ** 2)
    # 使用高斯函数建模中心先验
    center_prior = np.exp(-(dist_from_center ** 2) / (2 * (max_dist / 2) ** 2))
    center_prior = (center_prior * 255).astype(np.uint8)

    return center_prior
```

谱残差显著性检测（Spectral Residual）

假设 自然图像的对数幅度谱具有规律性（近似线性下降，1/f特性），显著区域会破坏这种规律性，产生谱残差

原理 谱残差 = 实际频谱 - 平滑频谱，高斯平滑操作去除了规律性部分，保留异常

优势

理论基础扎实，泛化能力最强

缺点

计算复杂度高，参数（sigma）敏感

```python
def spectral_residual_saliency(self, image):  1 usage
    """谱残差显著性检测"""
    # 转换为灰度图
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = gray.astype(np.float32) / 255.0
    # 傅里叶变换
    fft = np.fft.fft2(gray)
    log_amplitude = np.log(np.abs(fft) + 1e-7)
    phase = np.angle(fft)
    # 计算谱残差
    spectral_residual = log_amplitude - gaussian_filter(log_amplitude, sigma=3)
    # 重建图像
    combined = np.exp(spectral_residual + 1j * phase)
    saliency = np.abs(np.fft.ifft2(combined))
    # 后处理
    saliency = gaussian_filter(saliency ** 2, sigma=8)
    saliency = (saliency - saliency.min()) / (saliency.max() - saliency.min())
    saliency = (saliency * 255).astype(np.uint8)
```

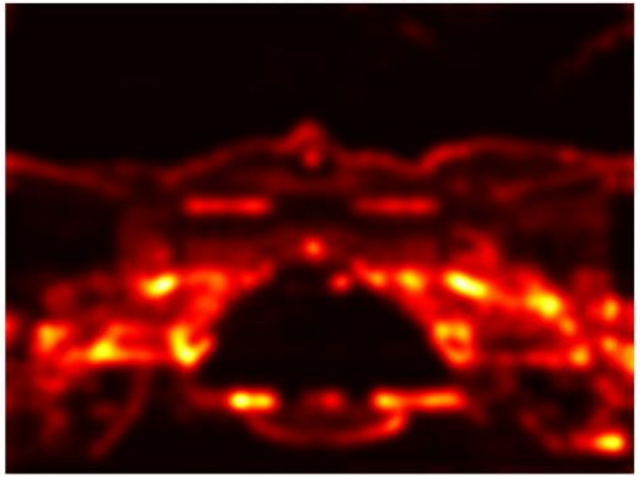
显著性图效果对比

Original

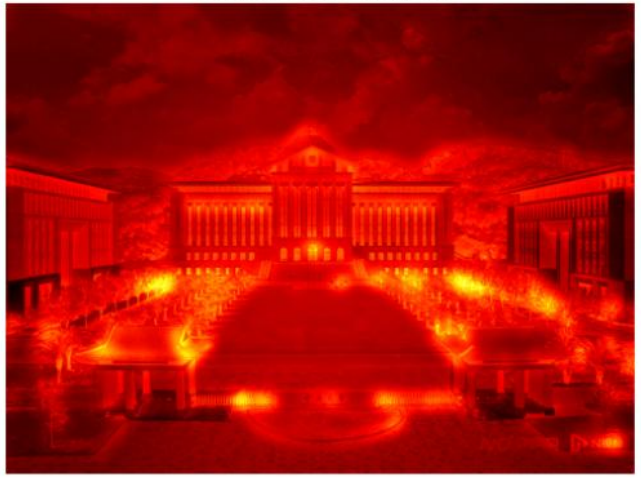
Color Contrast Saliency Map

Center Prior Saliency Map

Spectral Residual Saliency Map



Combined Saliency Map



Seam Carving 算法

假设 自然图像的对数幅度谱具有规律性（近似线性下降，1/f特性），显著区域会破坏这种规律性，产生谱残差

原理 谱残差 = 实际频谱 - 平滑频谱，高斯平滑操作去除了规律性部分，保留异常

```python
def calculate_energy(self, use_saliency=True):  3 usages
    """
    计算能量图

    Args:
        use_saliency: 是否使用显著性图增强能量
    """
    # 转换为灰度图
    gray = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
    # Sobel梯度
    dx = cv2.Sobel(gray, cv2.CV_64F, dx: 1, dy: 0, ksize=3)
    dy = cv2.Sobel(gray, cv2.CV_64F, dx: 0, dy: 1, ksize=3)
    # 梯度幅值作为基础能量
    energy = np.abs(dx) + np.abs(dy)
    # 如果有显著性图，将其融合到能量图中
    if use_saliency and self.saliency_map is not None:
        # 调整显著性图大小以匹配当前图像
        resized_saliency = cv2.resize(self.saliency_map,
                                      dsize: (self.image.shape[1], self.image.shape[0]))
        # 融合显著性信息（显著区域能量更高）
        saliency_weight = 1000  # 权重系数
        energy = energy + saliency_weight * (resized_saliency / 255.0)

    return energy
```

Seam Carving 算法

原理

图像中并非所有像素都同等重要。删除低能量的"接缝"(seam)对视觉影响最小，可以选择性删除不重要的像素来保护主体

Seam的定义

从上到下的8-连通路径，每行恰好经过一个像素，相邻行的像素列坐标差≤1（保证连续性）
8-连通保证了删除后图像仍然连续，不会断裂

能量函数

包括:

- 梯度能量：Sobel算子检测边缘，边缘处能量高
- 显著性增强：人脸、主体等重要区域能量高
- 权重平衡：λ控制显著性的影响程度（典型值1000）

能量高=重要=不能删，能量低=背景=可以删

优化算法：通过动态规划来找到从顶部到底部的最小能量路径，并迭代删除seam线

Seam Carving 算法—能量函数设计

Sobel梯度算子

Sobel算子本质是对图像进行卷积，梯度大的地方表示边缘，应该被保护。cv2.CV_64F使用64位浮点数避免梯度截断。

L1范数 vs L2范数

代码使用|dx| + |dy|(L1范数)而非sqrt(dx²+dy²)(L2范数)，因为L1计算更快且效果相当。

显著性融合策略

energy = gradient + λ·saliency，其中λ=1000是超参数。这种线性组合使得显著区域的能量值远高于背景，即使背景有边缘也会优先保护显著区域。

```python
def calculate_energy(self, use_saliency=True):  3 usages
    """
    计算能量图

    Args:
        use_saliency: 是否使用显著性图增强能量
    """
    # 转换为灰度图
    gray = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
    # Sobel梯度
    dx = cv2.Sobel(gray, cv2.CV_64F, dx: 1, dy: 0, ksize=3)
    dy = cv2.Sobel(gray, cv2.CV_64F, dx: 0, dy: 1, ksize=3)
    # 梯度幅值作为基础能量
    energy = np.abs(dx) + np.abs(dy)
    # 如果有显著性图，将其融合到能量图中
    if use_saliency and self.saliency_map is not None:
        # 调整显著性图大小以匹配当前图像
        resized_saliency = cv2.resize(self.saliency_map,
                                      dsize: (self.image.shape[1], self.image.shape[0]))
        # 融合显著性信息（显著区域能量更高）
        saliency_weight = 1000  # 权重系数
        energy = energy + saliency_weight * (resized_saliency / 255.0)

    return energy
```

Seam Carving 算法—动态规划

状态定义

dp[i,j]表示从第一行到达位置(i,j)的最小累积能量，化简为有向无环图(DAG)的最短路径问题。

8-连通约束

seam必须是连续的，相邻行的像素列坐标差不超过1。这保证了删除seam后图像仍然连续。

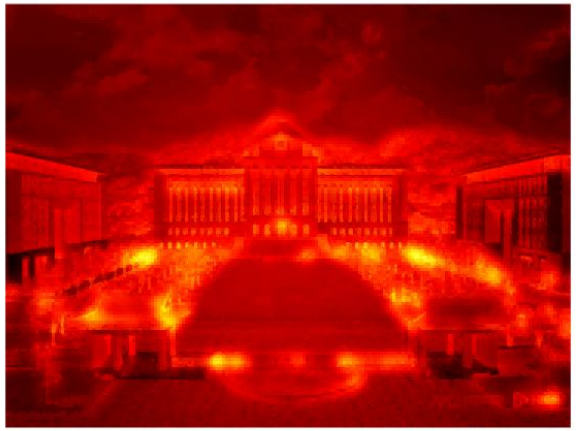
```python
def find_vertical_seam(self, energy):  3 usages
    Args:
        energy: 能量图

    Returns:
        seam: 垂直缝的列索引数组
    """
    h, w = energy.shape
    # 动态规划表
    dp = energy.copy()
    # 从第二行开始，计算累积最小能量
    for i in range(1, h):
        for j in range(w):
            # 检查上一行的三个可能位置
            if j == 0:
                dp[i, j] += min(dp[i - 1, j], dp[i - 1, j + 1])
            elif j == w - 1:
                dp[i, j] += min(dp[i - 1, j - 1], dp[i - 1, j])
            else:
                dp[i, j] += min(dp[i - 1, j - 1], dp[i - 1, j], dp[i - 1, j + 1])
    # 回溯找到最小能量缝
    seam = np.zeros(h, dtype=np.int32)
    seam[-1] = np.argmin(dp[-1])
    for i in range(h - 2, -1, -1):
        j = seam[i + 1]
        if j == 0:
            seam[i] = j + np.argmin(dp[i, j:j + 2])
        elif j == w - 1:
            seam[i] = j - 1 + np.argmin(dp[i, j - 1:j + 1])
        else:
            seam[i] = j - 1 + np.argmin(dp[i, j - 1:j + 2])
    return seam
```

Seam Carving 算法—动态规划

Original Image

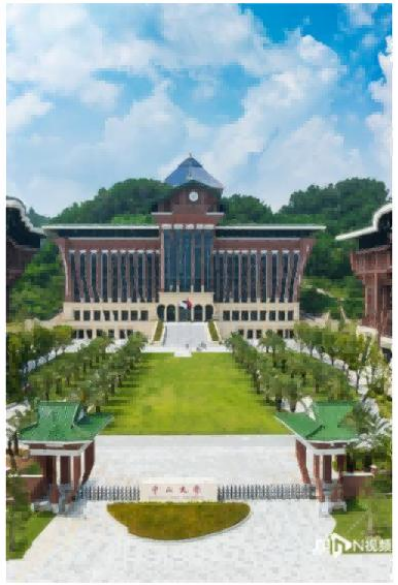
Saliency Map



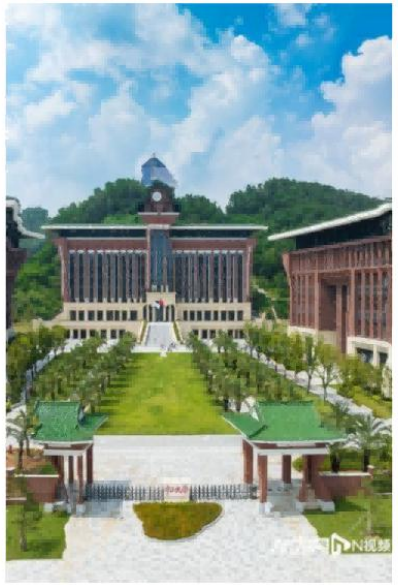
Energy Map

First Seam

Resized Image (400x600)



No Saliency Map Resized Image



质量评估

PSNR 峰值信噪比

衡量图像失真程度

值越大质量越好 (>30dB)

Step 1：计算MSE（均方误差）

差异 = (原图像素值 − 处理后像素值)²

MSE = 所有像素差异的平均值

Step 2：计算PSNR

$PSNR = 10 \times \log_{10}(MAX^2/MSE)$

其中：

- MAX = 像素最大可能值（8位图像是255）
- MSE = 均方误差
- $\log_{10}$ = 以10为底的对数

SSIM 结构相似性

综合亮度、对比度和结构

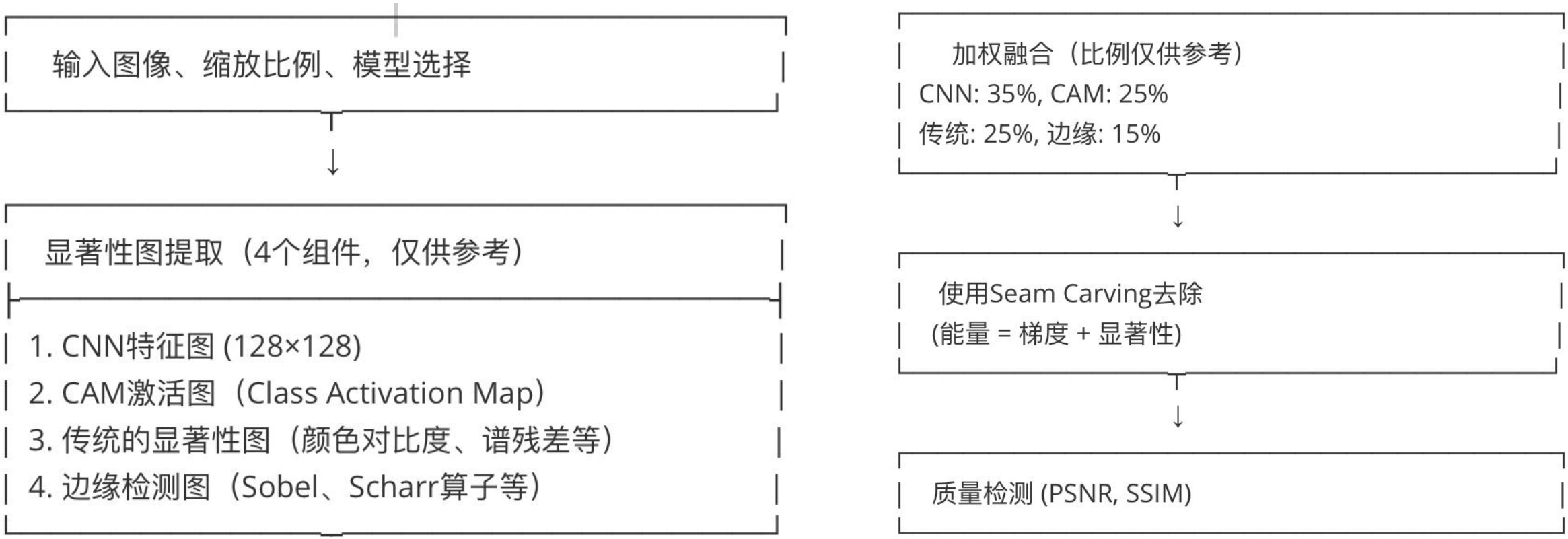
范围 [0,1]，接近1最好

$$SSIM(x,y) = l(x,y) \times c(x,y) \times s(x,y)$$

其中：

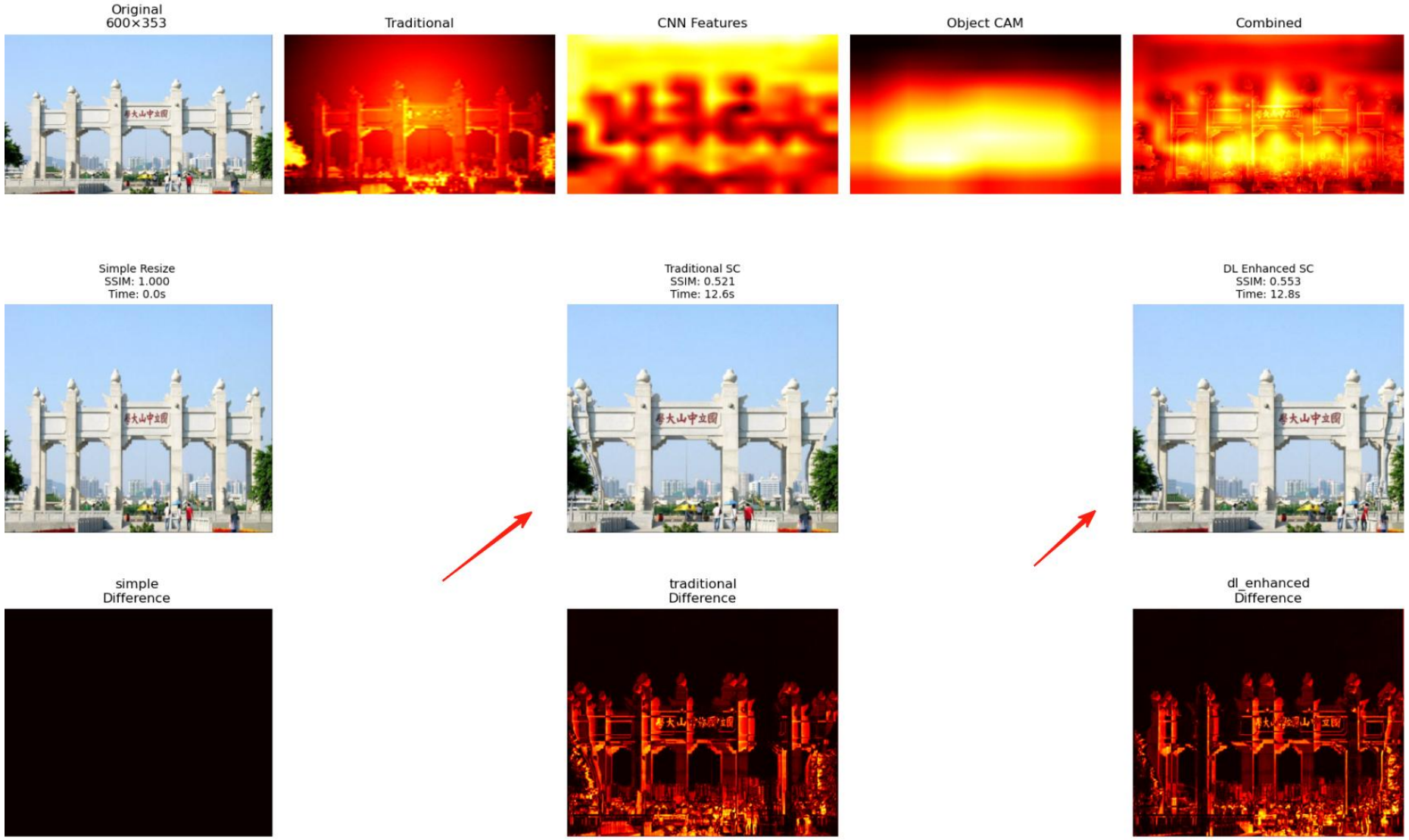
- $l(x,y)$ = 亮度相似度
- $c(x,y)$ = 对比度相似度
- $s(x,y)$ = 结构相似度

课后作业：如何使用神经网络来增强重定向质量？



模型选择：MobileNetV2，SqueezeNet，ResNet，Vision Transformer等

课后作业：如何使用神经网络来增强重定向质量？



代码与报告发送至：cvexp2025@163.com

题目描述

随着多种显示设备的普及（手机、平板、电脑、电视），图像需要适应不同的宽高比。传统的缩放方法会导致内容变形或重要信息丢失。Seam Carving是一种内容感知的图像缩放技术，但原始算法缺乏对图像语义内容的理解。本实验将探索如何使用轻量级深度学习模型增强Seam Carving算法，在有限的计算资源下实现更智能的图像重定向。本题目要求你**结合卷积神经网络**来增强显著性图效果，**使用Seam Carving算法进行图片重定向**，并观察效果和定量计算前后**PSNR、SSIM得分。**

要求

1.掌握常规显著性检测和CNN显著性图在图像重定向中的作用，考虑结合多种显著性图来增强特征

2.理解并实现基础Seam Carving算法，利用上一步的显著性图进行重定向

3.计算使用插值算法和基于显著性图的重定向算法应用后的SSIM和PSNR得分，列出表格进行比较。

请在完成后提交您的**代码**、**实机演示视频**和**实验报告**。邮件需命名为：**学号-姓名-第四次作业**，且**附件不超过30M**

DDL: 2025年10月20日

谢谢大家