**Project 2:** 编程实现图片水印嵌入和提取（可依托开源项目二次开发），并进行鲁棒性测试，包括不限于翻转、平移、截取、调对比度等。

# 在网上找到一份blind-watermark开源项目[linyacool/blind-watermark](https://github.com/linyacool/blind-watermark/tree/master)，以他为基准二次开发。

原嵌入代码的工作原理：

1.将原始图像使用傅里叶变换转换到频域；

2.将水印信息以随机分布的方式嵌入到频域中；

3.通过傅里叶逆变换将修改后的频域数据转换回空域，得到带有水印的图像；

原提取代码的工作原理：

1.分别读取原始图像和含水印图像，对这两幅图像进行二维傅里叶变换，将它们从空域转换到频域。

**2.水印提取**：在频域中，利用原始图像与含水印图像的频域差异，通过公式 (原始图像频域 - 含水印图像频域) / alpha 计算得到水印的频域信息将计算结果转换为实数（去除傅里叶变换可能产生的虚部）。

**3.水印恢复**：使用与嵌入时相同的随机种子（图像尺寸）和打乱方式，重建随机索引，根据这些索引，将频域中提取到的水印信息恢复到正确位置。

尝试运行原代码，没有得到预期的结果，水印提取的图片根本看不清是什么。

**对两种代码进行鲁棒性测试，改进优化：**

1. 将加法嵌入改为基于幅度谱的乘法嵌入，提高鲁棒性：

magnitude = np.abs(img\_f\_shifted) *# 获取幅度谱*

phase = np.angle(img\_f\_shifted) *# 获取相位谱*

*# 在幅度谱上做乘法嵌入（1+α\*W保证非负）*

watermarked\_magnitude = magnitude \* (1 + alpha \* watermark\_pattern) *# 关键改进点*

*# 重建复数频域*

watermarked\_freq = watermarked\_magnitude \* np.exp(1j \* phase) *# 保持相位不变*

乘法嵌入通过修改幅度谱而非直接修改复数，能更好地抵抗JPEG压缩等攻击。

1. 增加了完整的频域中心化处理（fftshift/ifftshift）

img\_f = np.fft.fft2(img, axes=(0, 1))

img\_f\_shifted = np.fft.fftshift(img\_f) *# 低频移至中心*

*# 在中心化频域嵌入水印*

watermarked\_freq\_shifted = img\_f\_shifted \* (1 + alpha \* watermark\_pattern)

*# 逆中心化后逆变换*

watermarked\_freq = np.fft.ifftshift(watermarked\_freq\_shifted)

watermarked = np.fft.ifft2(watermarked\_freq, axes=(0, 1))

****可视化优势****：中心化后，低频集中在频谱中心，水印可精准嵌入中频区域，避免修改极低频导致可见失真。

3.使用归一化处理保证数值稳定性

def normalize\_image(img):

return img.astype(np.float32) / 255.0 *# 归一化到[0,1]*

img = normalize\_image(cv2.imread(img\_path)) *# 输入范围稳定*

watermark = normalize\_image(cv2.imread(wm\_path))

*# 处理完成后反归一化*

def denormalize\_image(img):

return (img \* 255).clip(0, 255).astype(np.uint8)

cv2.imwrite(res\_path, denormalize\_image(watermarked))

归一化后所有运算在[0,1]范围内进行，避免溢出（如FFT后实部可能超出255）。

未归一化时，逆变换后需手动裁剪值域（np.clip），而归一化后自动满足约束条件。

4.增加错误处理机制

5.由于个人习惯，将代码改为了使用python的IDLE上运行

**鲁棒性测试：**

robustness.py和robustness\_metrics.py分别是鲁棒性图片测试和数据结果测试。

分别测试了水平翻转，垂直翻转，旋转90度，旋转180度，平移，中心截取，提高对比度，降低对比度，添加高斯噪声，JPEG压缩攻击几种鲁棒性。

从测试结果综合来看，该算法的鲁棒性较差。在结构相似性和图像质量方面表现不佳，不同攻击的影响差异较大，虽然在nc指标上相对稳定，但整体而言在面对各种攻击时的表现有待提高。

测试结果分别是robustness\_tests和robustness\_metrics.csv。