数字水印（Digital Watermarking）是一种将特定信息（水印）嵌入数字媒体内容（如图像、音频、视频）中的技术，这些信息在正常使用中不可察觉，但可通过专用算法提取。它是信息安全领域的重要分支，广泛应用于版权保护、内容认证和隐蔽通信。

本次实验采用 DCT 变换域方法，在图像的离散余弦变换（DCT）系数中嵌入水印。DCT 数字水印算法是首先把图像分成 8×8 的不重叠像素块，在经过分块 DCT 变换后，即得到由 DCT 系数组成的频率块，然后随机选取一些频率块，将水印信号嵌入到由密钥控制选择的一些 DCT 系数中。该算法是通过对选定的 DCT 系数进行微小变换以满足特定的关系，以此来表示一个比特的信息。

水印嵌入

```
1个用法
def embed_dct_watermark(cover_img, watermark_bin, strength=10.0):
    """
    嵌入水印到载体图像

    参数:
    cover_img: BGR格式的载体图像 (uint8)
    watermark_bin: 二值水印数组 (0/1值)
    strength: 水印嵌入强度

    返回:
    watermarked_img: 含水印图像 (uint8)
    """
    # 确保载体图像尺寸为块大小的整数倍
    height, width = cover_img.shape[:2]
    adj_height = (height // BLOCK_SIZE) * BLOCK_SIZE
    adj_width = (width // BLOCK_SIZE) * BLOCK_SIZE
    cover = cover_img[:adj_height, :adj_width].copy()

    # 转换到YCrCb空间并提取Y通道
    ycrcb_img = cv2.cvtColor(cover, cv2.COLOR_BGR2YCrCb).astype(np.float32)
    y_channel = ycrcb_img[:, :, 0]

    # 计算块数量
    block_rows = adj_height // BLOCK_SIZE
    block_cols = adj_width // BLOCK_SIZE
```

实现过程为实现图片转化为灰度格式，返回灰度图像的函数，实现计算两张二值图的归一化相关系数的函数。

水印核心函数

使用 DCT 变换和 QIM 方法嵌入水印。

1. 将图像分成 8×8 块，对每块进行 DCT 变换。
2. 选择一个中频系数（如(4,3)位置），根据水印比特（0 或 1）使用 QIM 方法修改该系数。
3. 对修改后的块进行逆 DCT 变换，得到含水印的图像块。
4. 提取时，同样分块进行 DCT 变换，根据系数值的小数部分判断水印比特。

水印提取

```python
def extract_dct_watermark(watermarked_img, wm_shape):
    """
    从含水印图像中提取水印

    参数:
    watermarked_img: 含水印图像（BGR格式，uint8）
    wm_shape: 目标水印尺寸（高度，宽度）

    返回:
    extracted_bits: 提取的二值水印（0/1数组）
    """
    height, width = watermarked_img.shape[:2]
    adj_height = (height // BLOCK_SIZE) * BLOCK_SIZE
    adj_width = (width // BLOCK_SIZE) * BLOCK_SIZE
    img = watermarked_img[:adj_height, :adj_width].copy()

    # 提取Y通道
    ycrcb_img = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb).astype(np.float32)
    y_channel = ycrcb_img[:, :, 0]

    # 计算块数量
    block_rows = adj_height // BLOCK_SIZE
    block_cols = adj_width // BLOCK_SIZE
```

将图像同样分为 8x8 块，对每个块做 dct，读取对应中频系数，进行判断，最后将所有位组合得到二值水印图像

鲁棒性测试

```python
def run_evaluation(cover_path, wm_path, output_dir='output', strength=10.0):
    """运行水印嵌入、提取与鲁棒性评估流程"""
    os.makedirs(output_dir, exist_ok=True)

    # 读取输入图像
    cover_img = cv2.imread(cover_path)
    if cover_img is None:
        raise ValueError("无法读取载体图像")
    wm_gray = cv2.imread(wm_path, cv2.IMREAD_GRAYSCALE)
    if wm_gray is None:
        raise ValueError("无法读取水印图像")

    # 水印二值化
    _, wm_binary = cv2.threshold(wm_gray, thresh: 127, maxval: 1, cv2.THRESH_BINARY)

    # 嵌入水印
    watermarked_img, wm_grid = embed_dct_watermark(cover_img, wm_binary, strength=strength)
    output_path = os.path.join(output_dir, 'watermarked.png')
    cv2.imwrite(output_path, watermarked_img)
    print(f"已保存含水印图像至: {output_path}")

    # 无攻击提取
    extracted_clean = extract_dct_watermark(watermarked_img, wm_binary.shape)
    cv2.imwrite(os.path.join(output_dir, 'extracted_clean.png'), extracted_clean * 255)
    acc_clean = calculate_accuracy(wm_binary, extracted_clean)
```

鲁棒性测试要抵抗常见的几何攻击（翻转，旋转），像素值攻击（对比度）
测试方式为在修改过的图像提取水印，与原本水印比较归一化相关系数

进行测试

```
ValueError: 无法读取载体图像
PS D:\实验文件\shujia\1> py watermark.py --cover cover.jpg --watermark wm.png --strength 10.0
已保存含水印图像至: output\watermarked.png
无攻击提取准确率: 92.97%  载体与含水印图像PSNR: 36.92 dB
水平翻转攻击后提取准确率: 33.90%  已保存攻击图像与提取结果
垂直翻转攻击后提取准确率: 66.77%  已保存攻击图像与提取结果
平移攻击后提取准确率: 56.42%  已保存攻击图像与提取结果
80%裁剪攻击后提取准确率: 76.02%  已保存攻击图像与提取结果
高对比度攻击后提取准确率: 91.99%  已保存攻击图像与提取结果
低对比度攻击后提取准确率: 92.96%  已保存攻击图像与提取结果
高斯噪声攻击后提取准确率: 92.17%  已保存攻击图像与提取结果
JPEG(Q70)攻击后提取准确率: 92.77%  已保存攻击图像与提取结果
JPEG(Q50)攻击后提取准确率: 86.42%  已保存攻击图像与提取结果

评估结果汇总:
水平翻转        准确率:  33.90%
垂直翻转        准确率:  66.77%
平移          准确率:  56.42%
80%裁剪       准确率:  76.02%
高对比度        准确率:  91.99%
低对比度        准确率:  92.96%
高斯噪声        准确率:  92.17%
JPEG(Q70)  准确率:  92.77%
JPEG(Q50)  准确率:  86.42%
```