基于数字水印的图片泄露检测

实验目的

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。

实验过程

- OpenCV：用于图像处理基础操作
- DWT（离散小波变换）：用于水印嵌入的频域变换
- Python：作为主要开发语言

水印嵌入流程

1. 读取原始图像和水印信息
2. 对图像进行DWT变换
3. 在频域中嵌入水印
4. 进行IDWT逆变换
5. 保存含水印图像

代码块

```python
def embed_watermark(original_img_path, watermark_text, output_path, alpha=0.1):
    # 读取原始图像
    original_img = cv2.imread(original_img_path, cv2.IMREAD_GRAYSCALE)

    # 将水印文本转换为二进制
    watermark_binary = ''.join(format(ord(c), '08b') for c in watermark_text)

    # 进行DWT变换
    coeffs = pywt.dwt2(original_img, 'haar')
    LL, (LH, HL, HH) = coeffs

    # 在LL子带中嵌入水印
    rows, cols = LL.shape
    watermark_length = len(watermark_binary)

    if watermark_length > rows * cols:
        raise ValueError("水印信息过大，无法嵌入")
```

```
18
19      # 嵌入水印
20      watermarked_LL = LL.copy()
21      index = 0
22      for i in range(rows):
23          for j in range(cols):
24              if index < watermark_length:
25                  # 修改LL系数的LSB
26                  watermarked_LL[i,j] = LL[i,j] * (1 + alpha *
    (int(watermark_binary[index]) - 0.5))
27                  index += 1
28              else:
29                  break
30
31      # 逆DWT变换
32      watermarked_coeffs = (watermarked_LL, (LH, HL, HH))
33      watermarked_img = pywt.idwt2(watermarked_coeffs, 'haar')
34
35      # 保存含水印图像
36      watermarked_img = np.uint8(watermarked_img)  # 强制转换为CV_8U
37      cv2.imwrite(output_path, watermarked_img)
38
39      return watermarked_img
```

水印提取流程

1. 读取含水印图像
2. 进行DWT变换
3. 从频域提取水印信息
4. 验证水印完整性

代码块

```
1   def extract_watermark(watermarked_img_path, original_img_path,
    watermark_length, alpha=0.3):
2       # 读取含水印图像和原始图像
3       watermarked_img = cv2.imread(watermarked_img_path, cv2.IMREAD_GRAYSCALE)
4       original_img = cv2.imread(original_img_path, cv2.IMREAD_GRAYSCALE)
5
6       # 对两幅图像进行DWT变换
7       coeffs_watermarked = pywt.dwt2(watermarked_img, 'haar')
8       LL_w, _ = coeffs_watermarked
9
10      coeffs_original = pywt.dwt2(original_img, 'haar')
11      LL_o, _ = coeffs_original
```

```
    # 提取水印
    extracted_binary = ''
    rows, cols = LL_w.shape
    index = 0

    for i in range(rows):
        for j in range(cols):
            if index < watermark_length * 8:  # 每个字符8位
                # 比较系数差异
                diff = (LL_w[i,j] - LL_o[i,j]) / (alpha * LL_o[i,j]) + 0.5
                bit = '1' if diff > 0.5 else '0'
                extracted_binary += bit
                index += 1
            else:
                break

    # 将二进制转换为文本
    watermark_text = ''
    for i in range(0, len(extracted_binary), 8):
        byte = extracted_binary[i:i+8]
        watermark_text += chr(int(byte, 2))

    return watermark_text
```

鲁棒性测试

旋转：通过将图像绕中心点旋转一定角度，改变像素的空间分布，从而破坏水印的空间结构

裁剪：从图像中移除部分区域，仅保留剩余部分，可能导致水印数据的部分丢失甚至完全丢失

噪声：通过向图像添加随机扰动（如高斯噪声、椒盐噪声）修改像素值，破坏水印

压缩：JPEG 压缩通过 DCT 变换 + 量化 减少图像文件大小，引入误差，导致提取不准确

对比度调整：通过线性或非线性变换修改像素值范围，拉伸像素值分布，导致提取时误判

代码块

```
def robustness_test(watermarked_img_path, test_type, params):
    img = cv2.imread(watermarked_img_path)

    if test_type == 'rotate':
        # 旋转测试
        angle = params.get('angle', 30)
        rows, cols = img.shape[:2]
        M = cv2.getRotationMatrix2D((cols/2, rows/2), angle, 1)
        return cv2.warpAffine(img, M, (cols, rows))
```

```
10
11      elif test_type == 'crop':
12          # 裁剪测试
13          x, y, w, h = params.get('crop_area', (10, 10, 200, 200))
14          return img[y:y+h, x:x+w]
15
16      elif test_type == 'contrast':
17          # 对比度调整
18          alpha = params.get('alpha', 1.5)
19          return cv2.convertScaleAbs(img, alpha=alpha, beta=0)
20
21      elif test_type == 'noise':
22          # 添加噪声
23          mean = 0
24          var = params.get('var', 10)
25          sigma = var ** 0.5
26          gaussian = np.random.normal(mean, sigma, img.shape)
27          noisy_img = img + gaussian
28          return np.clip(noisy_img, 0, 255).astype(np.uint8)
29
30      elif test_type == 'compress':
31          # JPEG压缩
32          quality = params.get('quality', 50)
33          cv2.imwrite('temp.jpg', img, [int(cv2.IMWRITE_JPEG_QUALITY), quality])
34          return cv2.imread('temp.jpg')
35
36      else:
37          return img.copy()
```

实验结果

1. 正常水印提取成功：
   - `提取的水印: Hello,world!` 表明基础功能正常工作
2. 旋转攻击后提取失败：
   - `从旋转图像中提取的水印:` （空结果）
   - 原因：旋转操作破坏了DWT系数的空间关系，当前算法没有几何校正能力
3. 对比度调整后乱码：
   - `从对比度调整图像中提取的水印: ÿÿÿÿÿÿÿÿÿÿ`
   - 原因：对比度变化放大了系数差异，导致提取算法失效

```
C:\Users\Lenovo\Desktop\PYTHON-VSCode>C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/PYTHON-VSCode/watermark.py
提取的水印: Hello,world!
从旋转图像中提取的水印:
从对比度调整图像中提取的水印: ÿÿÿÿÿÿÿÿÿÿ
```