



网络安全创新创业实践第二次实验

数字图像水印系统

学院: 网络空间安全学院

专业: 密码科学与技术

姓名: 金鑫悦

学号: 202200460042

目录

1 实验任务	2
2 实验环境	2
3 实验原理	2
3.1 数字水印技术概述	2
3.2 离散余弦变换 (DCT)	2
3.3 水印嵌入与提取	3
3.3.1 水印嵌入	3
3.3.2 水印提取	3
3.4 鲁棒性测试	3
4 代码实现	4
5 结果分析	7
6 总结与感悟	14

1 实验任务

基于数字水印的图片泄露检测编程实现图片水印嵌入和提取，并进行鲁棒性测试，包括不限于翻转、平移、截取、调对比度等

2 实验环境

硬件环境：

12th Intel (R) Core (TM) i7-12700H

软件环境：

windows11、python3.12 (64-bit)、pycharm2024.1.4

3 实验原理

3.1 数字水印技术概述

数字水印技术是在数字媒体文件中嵌入一段信息，使得该信息可以在不影响文件质量的情况下持续存在。数字水印有两个主要应用：

- 版权保护：防止未授权复制或盗用，提供内容的可追溯性。
- 内容认证：确保内容没有被篡改，验证内容的完整性。

数字水印通常分为两类：

- 显性水印：可以直接察觉的水印，通常用于版权保护。
- 隐性水印：嵌入的水印不可被察觉，通常用于内容认证。

水印的嵌入通常会利用图像的冗余信息（如视觉上不易察觉的低频信息）来存储水印信息，同时保证水印的鲁棒性和透明性。鲁棒性是指水印在经过多种图像处理操作后依然能够有效地提取，而透明性是指水印嵌入后对原始图像的影响应尽量小。

3.2 离散余弦变换 (DCT)

离散余弦变换 (DCT) 是一种常用于信号处理的变换方法，尤其适用于图像和视频压缩（如 JPEG）。DCT 能够将图像从空间域转换到频域。频域中的图像可以分为低频和高频部分，其中低频部分包含了图像的主要视觉信息，而高频部分包含了图像的细节信息。

在水印嵌入中，DCT 的主要优势是能够在低频部分嵌入水印，而这些低频部分对图像质量的影响较小。图像的低频部分是视觉感知中最重要的部分，它通常包含图像的大致结构和主要特征。因此，水印可以在这些频率上嵌入而不会显著影响图像的视觉效果。

DCT 的步骤：

1. 将图像从空间域转换到频域。
2. 对频域图像中的低频部分进行操作，比如嵌入水印。
3. 将操作后的图像通过逆 DCT 变换恢复到空间域。

3.3 水印嵌入与提取

3.3.1 水印嵌入

在本项目中，水印嵌入的过程依赖于 DCT。原始图像通过两次 DCT 变换被转换到频域后，水印信息被嵌入到低频部分（图像的前 32x32 的频域系数）。具体步骤如下：

- 读取原图像和水印：首先，将灰度图像格式的原图和水印图像读取进来，并将水印图像调整为固定大小 (32x32)。
- 二值化水印：将水印图像进行二值化处理，使得每个像素的值为 0 或 255，以便在频域中嵌入水印。
- DCT 变换：对原图像进行 DCT 变换，将图像从空间域转换到频域。变换后的图像由低频和高频组成，其中低频包含了图像的大部分信息。
- 水印嵌入：水印被嵌入到 DCT 变换后的低频部分。嵌入方式是将水印值加到低频系数中，水印的强度由一个系数 alpha 控制。
- 逆 DCT：将修改后的频域图像通过逆 DCT 恢复为空间域图像，得到嵌入水印后的图像。

3.3.2 水印提取

水印提取的过程是从已嵌入水印的图像中恢复出水印信息。提取过程如下：

- 读取水印图像和原图：首先，读取已嵌入水印的图像和原始图像。
- DCT 变换：对水印图像和原图分别进行 DCT 变换，得到频域图像。
- 水印恢复：通过对水印图像和原图的 DCT 变换结果，提取出低频部分的差异，并将其除以 alpha（嵌入时使用的强度系数）。通过这种方式恢复出水印。
- 二值化处理：恢复的水印经过二值化处理，最终提取出水印图像。

3.4 鲁棒性测试

鲁棒性测试是检验水印是否能够抵抗各种图像处理操作（如旋转、缩放、裁剪、压缩等）。常见的攻击方法包括：

- 翻转攻击：水平翻转图像。翻转通常不会显著影响低频信息，因此水印在这种攻击下通常能够成功提取。
- 平移攻击：对图像进行平移（即平移图像中的像素）。平移操作会影响图像的像素位置，但不会改变图像的主要信息，因此水印依然可以被提取。
- 裁剪攻击：裁剪图像的边缘部分。裁剪会丢失图像的部分内容，尤其是低频部分。裁剪过多可能导致水印信息丢失，水印提取的精度降低。
- 对比度调整：调整图像的对比度。对比度调整改变了图像的亮度和对比度，但对低频信息的影响较小，因此水印可以在一定程度上抵抗对比度调整。

4 代码实现

1. attacks.py, 包含了一些常见的图像攻击方法, 旨在模拟攻击对水印的影响。

```
1 import cv2
2 import numpy as np
3
4 def apply_flip(image):
5     return cv2.flip(image, 1)
6
7 def apply_translation(image, dx=10, dy=10):
8     M = np.float32([[1, 0, dx], [0, 1, dy]])
9     return cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
10
11 def apply_crop(image, crop_size=30):
12     h, w = image.shape[:2]
13     cropped = image[crop_size:h-crop_size, crop_size:w-crop_size]
14     return cv2.resize(cropped, (w, h)) # 还原尺寸以便提取
15
16 def apply_contrast(image, alpha=1.5, beta=0):
17     return cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
```

2. embed_watermark.py, 实现了水印的嵌入, 使用 DCT (离散余弦变换) 来将水印嵌入到图像的频域中。

```
1 import cv2
2 import numpy as np
3 from scipy.fftpack import dct, idct
4
5 def embed_watermark(cover_path, watermark_path, output_path, alpha=10):
6     # 读取原图和水印
7     cover = cv2.imread(cover_path, cv2.IMREAD_GRAYSCALE)
8     watermark = cv2.imread(watermark_path, cv2.IMREAD_GRAYSCALE)
9
10    # 调整水印尺寸
11    watermark = cv2.resize(watermark, (32, 32))
12    watermark = (watermark > 128).astype(np.float32) # 二值化
13
14    # DCT
15    dct_cover = dct(dct(cover.astype(float), axis=0, norm='ortho'),
16                     axis=1, norm='ortho')
16    dct_cover[:32, :32] += alpha * watermark
17
```

```
18     # 逆 DCT
19     watermarked = idct(idct(dct_cover, axis=1, norm='ortho'), axis=0,
20                           norm='ortho')
21
22     watermark = np.clip(watermarked, 0, 255).astype(np.uint8)
23
24     cv2.imwrite(output_path, watermark)
25
26     print(f"水印嵌入完成: {output_path}")
```

3. extract_watermark.py, 实现了从水印图像中提取水印的功能。

```
1  import cv2
2  import numpy as np
3  from scipy.fftpack import dct
4
5  def extract_watermark(watermarked_path, original_path, alpha=10):
6      w_img = cv2.imread(watermarked_path, cv2.IMREAD_GRAYSCALE)
7      o_img = cv2.imread(original_path, cv2.IMREAD_GRAYSCALE)
8
9      dct_w = dct(dct(w_img.astype(float), axis=0, norm='ortho'), axis=1,
10                  norm='ortho')
11
12     dct_o = dct(dct(o_img.astype(float), axis=0, norm='ortho'), axis=1,
13                  norm='ortho')
14
15
16     wm = (dct_w[:32, :32] - dct_o[:32, :32]) / alpha
17     wm = (wm > 0.5).astype(np.uint8) * 255
18
19     return wm
```

4. test.py, 进行整个流程的测试，包括嵌入水印、攻击图像、提取水印和保存结果。

```
1  import cv2
2  from embed_watermark import embed_watermark
3  from extract_watermark import extract_watermark
4  from attacks import apply_flip, apply_translation, apply_crop,
5                                apply_contrast
6
7  # 文件路径
8  original_image = r"D:\pythonProject1\project2\lenna.png"
9  watermark_image = r"D:\pythonProject1\project2\watermark.png"
10
11 # Step 1: 嵌入水印
12 embed_watermark(original_image, watermark_image, watermarked_output)
13
```

```
14 # Step 2: 加载嵌入后的图像
15 wm_img = cv2.imread(watermarked_output, cv2.IMREAD_GRAYSCALE)
16
17 # Step 3: 模拟攻击
18 attacks = {
19     "flip": apply_flip(wm_img),
20     "translate": apply_translation(wm_img),
21     "crop": apply_crop(wm_img),
22     "contrast": apply_contrast(wm_img)
23 }
24
25 # Step 4: 对每种攻击提取水印并保存
26 for name, attacked_img in attacks.items():
27     attacked_path = f"{name}.png"
28     cv2.imwrite(attacked_path, attacked_img)
29
30     extracted = extract_watermark(attacked_path, original_image)
31     cv2.imwrite(f"extracted_{name}.png", extracted)
32     print(f"{name} 攻击后的水印已提取并保存为 extracted_{name}.png")
```

5 结果分析

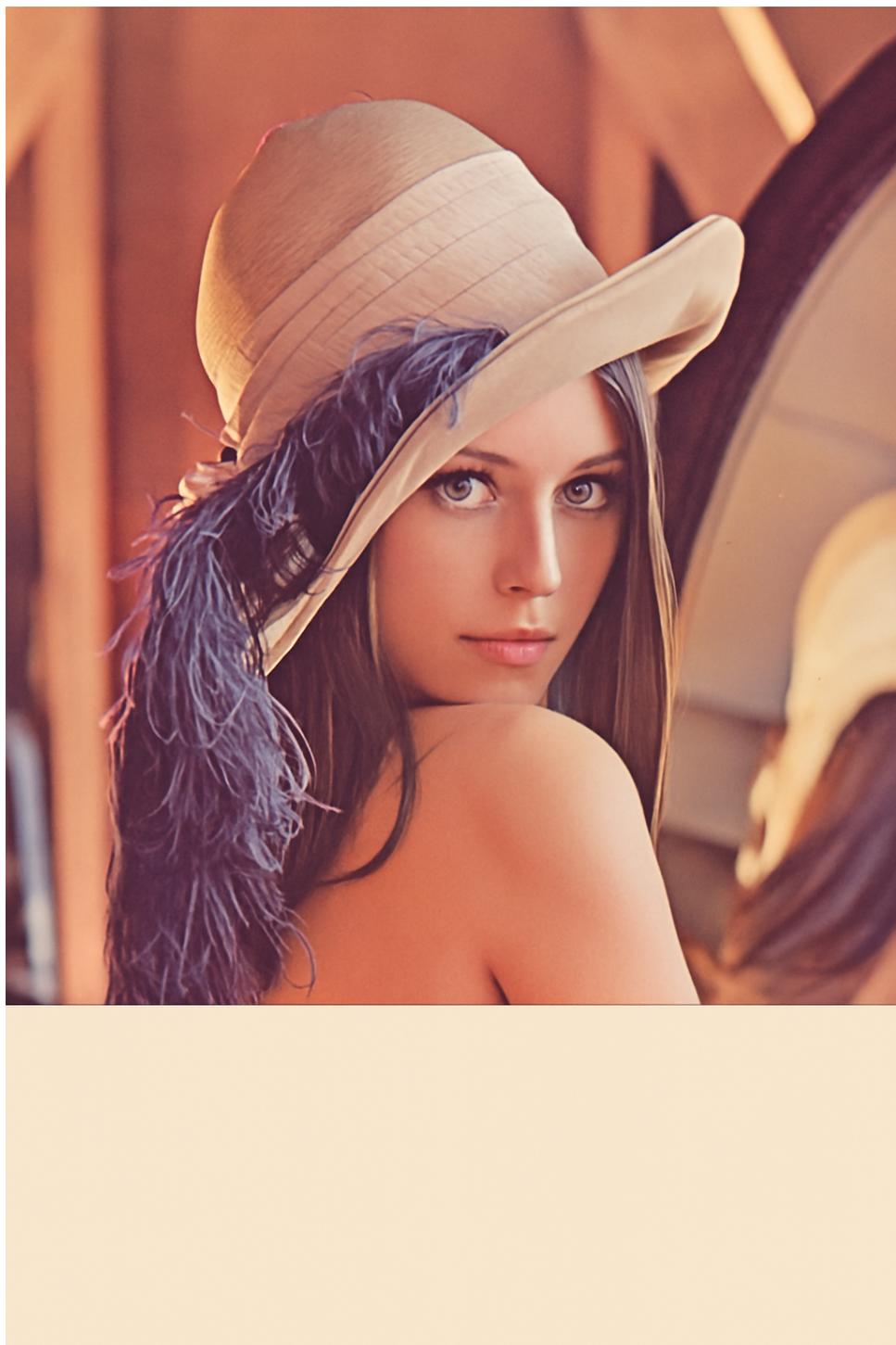


图 1: 原始图像

原图 (水印未受攻击)。

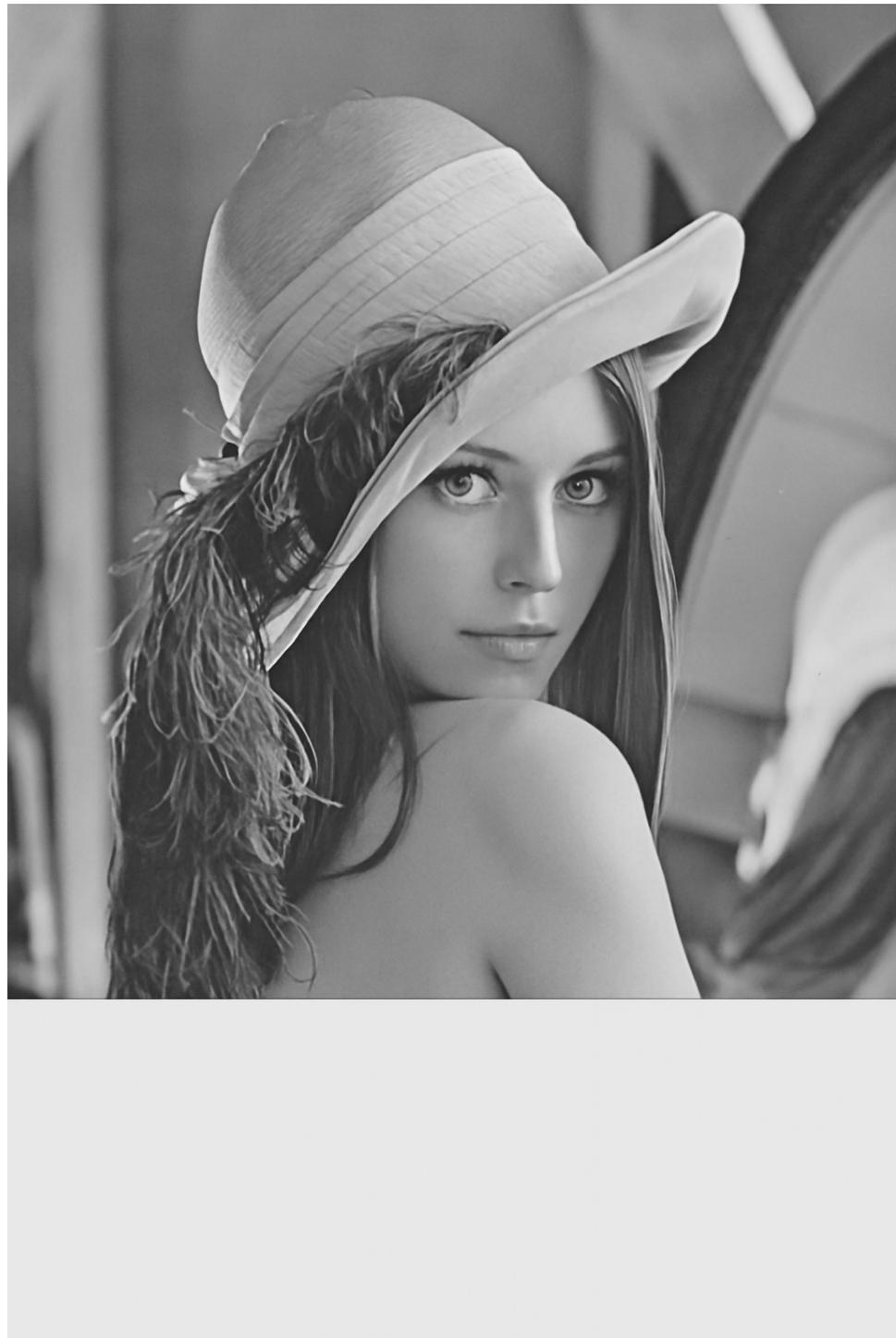


图 2: 嵌入水印后

完成了水印嵌入。

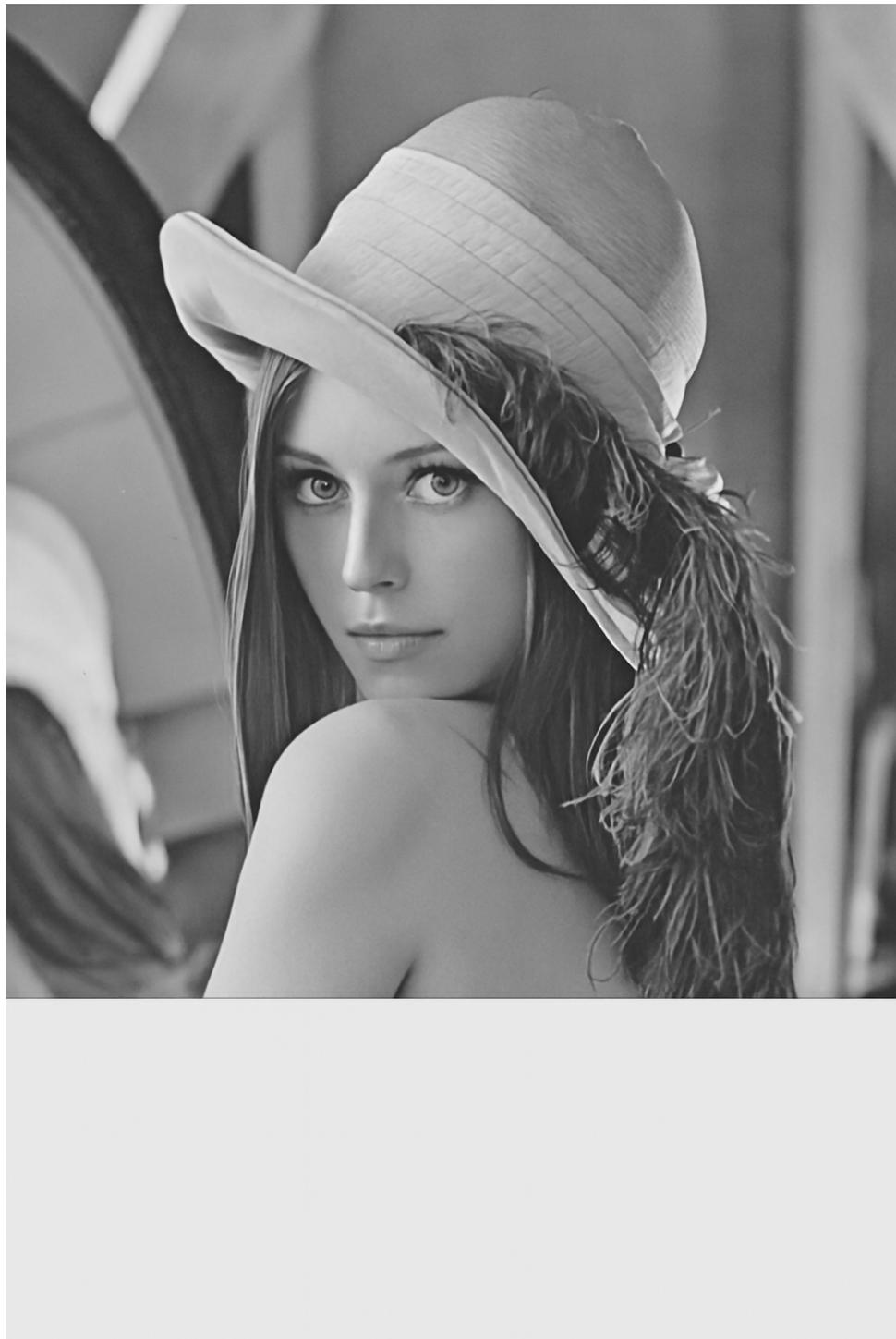


图 3: 翻转图像

翻转图像：水印图像进行了翻转操作。

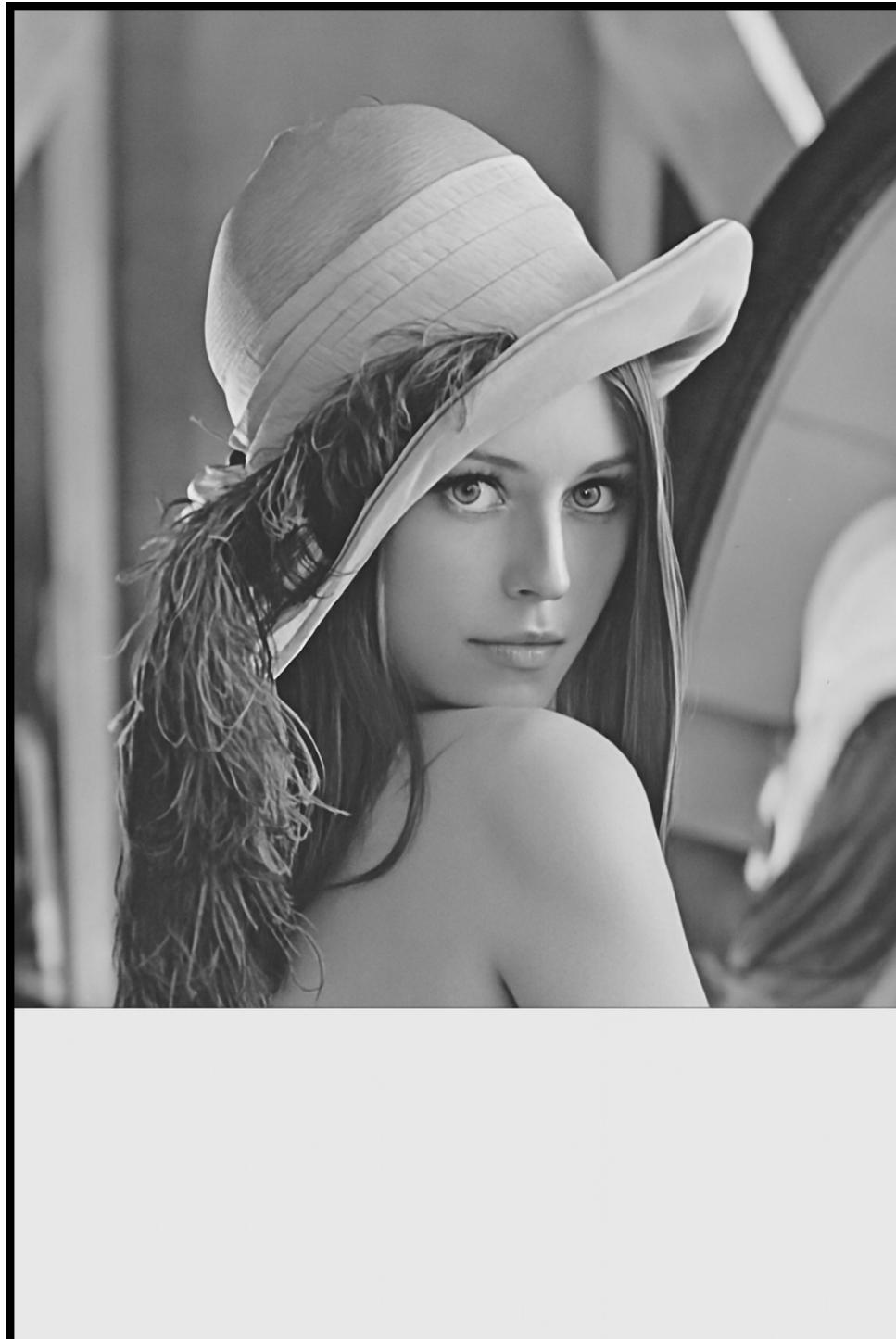


图 4: 平移图像

平移图像：水印图像进行了平移操作。

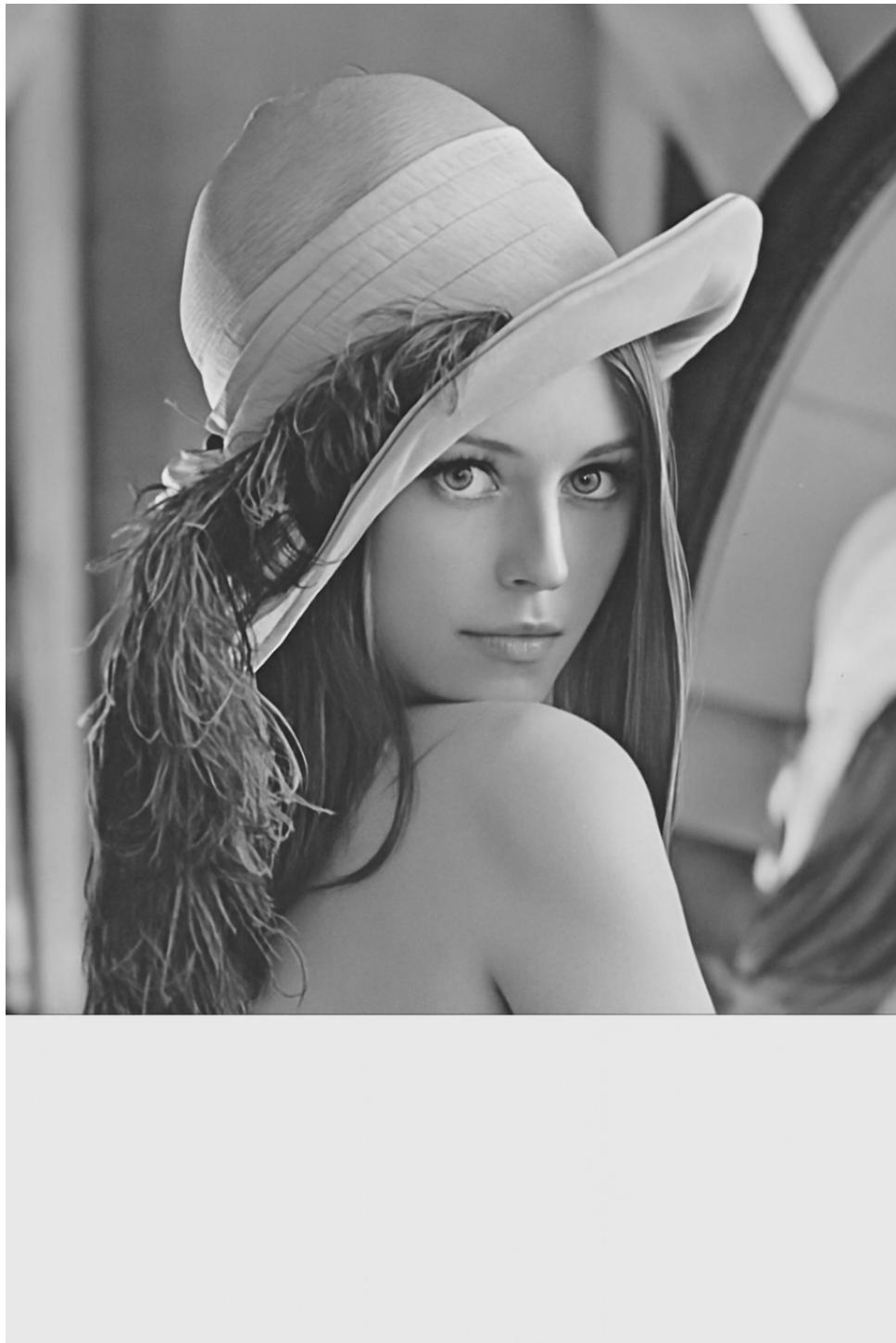


图 5: 裁剪图像

裁剪图像：图像的边缘被裁剪。



图 6: 对比度调整

对比度调整：图像的对比度进行了调整。



图 7: 水印图像

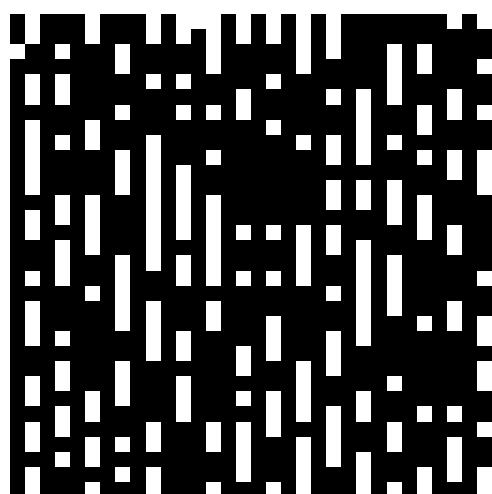


图 8: 翻转攻击

翻转攻击：水印成功提取，整体效果较好，翻转操作没有显著影响水印的提取。

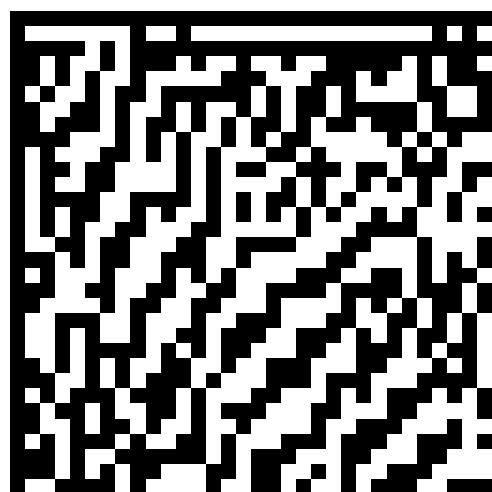


图 9: 平移攻击

平移攻击：水印仍然可以提取，平移后的水印清晰度较高，几乎没有损失。

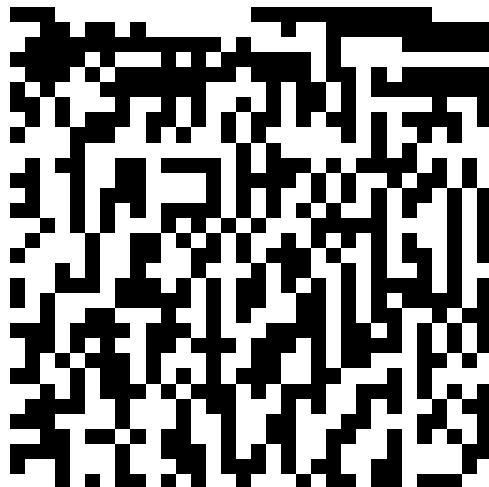


图 10: 裁剪攻击

裁剪攻击：水印仍然可以被提取，但部分区域存在明显的缺失，水印的形态有所变化。

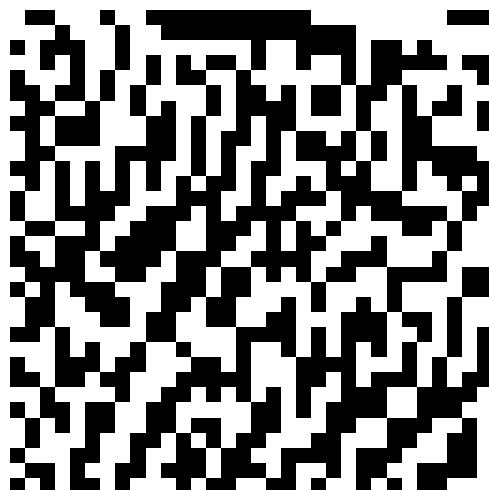


图 11: 对比度调整

对比度调整：水印提取效果较好，虽然有部分模糊，但整体可以辨识。

6 总结与感悟

实验过程中的问题：

1. **数字水印嵌入与提取的鲁棒性：**在实现数字水印嵌入和提取的过程中，最大的挑战之一是如何确保水印在面对不同攻击时的鲁棒性。特别是在图像进行翻转、平移或裁剪等操作后，水印仍然能够被有效地提取。这个过程中的关键在于如何选择适合的频域（如 DCT）来嵌入水印，并确保水印的鲁棒性。
2. **图像处理技术的应用问题：**在图像的平移、翻转、裁剪及对比度调整等攻击过程中，我发现图像处理的一些细节可能会影响水印提取的效果。例如，裁剪过多的图像可能会导致水印的低频信息丢失，从而无法成功提取水印。因此，在图像处理和水印嵌入之间的平衡是一个挑战。

实验亮点:

1. **DCT 频域水印嵌入的应用:** 在本项目中, 我使用 DCT (离散余弦变换) 来进行水印的嵌入。DCT 的低频特性确保了水印在图像的视觉质量上不会产生过大影响, 同时仍能有效嵌入水印。通过频域的处理, 水印不容易受到图像噪声的影响, 尤其是在翻转、平移和对比度调整等常见攻击下, 水印仍能成功提取。
2. **鲁棒性测试的成功验证:** 我成功模拟了多种图像攻击, 包括翻转、平移、裁剪、对比度调整等, 并通过提取水印验证了算法的鲁棒性。实验结果表明, 虽然裁剪和过度的图像变形会对水印的提取造成影响, 但其他大多数攻击 (如翻转、平移等) 不会显著影响水印的提取效果。通过这种测试, 我对水印算法在不同攻击下的表现有了深入的理解。

实验收获: 本次实验让我深入理解了数字水印技术, 特别是频域水印嵌入和提取的过程。通过学习 DCT 的使用和水印在图像中的嵌入技巧, 我更清晰地掌握了如何将水印信息嵌入到图像中而不会影响图像质量, 同时保持较高的鲁棒性。