山东大学
网络空间安全学院

创新创业实践
Project2

学生姓名：滕怀源

学号：202200460054)

学院：网络空间安全学院

专业班级： 2022 级网安 2 班

完成时间：2025 年 7 月 13 日

目录

1 基于 DCT 的彩色图像水印嵌入与提取

1.1 实验目的

本实验旨在实现一种基于离散余弦变换（Discrete Cosine Transform, DCT）的数字水印嵌入与提取方法。水印嵌入于图像的亮度通道中频区域，以确保其在保持图像感知质量的同时具有较好的鲁棒性。通过对嵌入图像施加翻转、平移、裁剪、对比度增强等常见图像攻击，评估水印的提取效果，分析其鲁棒性。

1.2 方法原理

1.2.1 颜色空间转换

考虑到人眼对亮度（Y）更敏感，对色度（Cr, Cb）不敏感，因此将彩色图像由 BGR 空间转换到 YCrCb 空间，并仅在 Y 通道中嵌入水印。

1.2.2 DCT 域嵌入

将图像亮度通道划分为 8×8 的子块，对每块进行 DCT 变换，选择一个固定的中频系数位置 (u, v) 进行修改：

$$D'(u, v) = D(u, v) + \alpha \cdot (2b - 1)$$

其中 $D(u, v)$ 为原始 DCT 系数，$b \in \{0, 1\}$ 是待嵌入的水印比特，α 为嵌入强度。修改后的 DCT 系数经 IDCT 变换重建图像。

1.2.3 水印提取

在接收端，将图像同样划分为 8×8 块，逐块进行 DCT，读取对应位置的系数判断符号：

$$\hat{b} = \begin{cases} 1, & D(u, v) > 0 \\ 0, & D(u, v) \leq 0 \end{cases}$$

恢复得到嵌入的水印图案。

1.3 鲁棒性攻击设计

为了测试算法的鲁棒性，实验对嵌入水印后的图像进行了以下常见图像处理操作：

- **水平翻转**：镜像处理，用于测试位置对称攻击。
- **平移**：图像整体向右下方平移若干像素。
- **裁剪**：裁去中心外区域并恢复为原尺寸。
- **对比度增强**：将图像对比度提升 $\alpha = 1.5$。

每种攻击后尝试从受损图像中提取出原始水印，并与原始水印图案进行对比分析。

1.4 实验结果与分析

实验结果显示，基于 DCT 的中频水印嵌入方法在不同图像攻击下均能成功提取出主要的水印结构，具有良好的鲁棒性。其中：

- 水平翻转对水印提取影响最小；
- 平移与裁剪会改变水印对应块的空间位置，提取结果略有扰动；
- 对比度增强会改变 DCT 系数的幅值，但符号未发生明显翻转，因此提取结果仍可识别。

该方法具有实现简单、嵌入隐蔽、对部分图像变换具备鲁棒性等优点，适合用于图像版权保护等场景。

2 系统实现分析

本实验基于 Python、OpenCV 和 NumPy 实现了一个完整的彩色图像数字水印系统，主要包括图像预处理、DCT 域嵌入、攻击模拟与水印提取四个核心部分。以下将结合实际代码逐段进行分析。

2.1 参数设置与水印设计

```
block_size = 8
embed_pos = (4, 4)
alpha = 10
```

此处设置水印嵌入的基本参数：每 8×8 的 DCT 块中嵌入一个比特，嵌入位置选择中频系数 $(4, 4)$，嵌入强度为 $\alpha = 10$，在保证嵌入效果的同时降低对图像质量的影响。水印本身为一个 4×4 的二值矩阵，实际应用中可替换为二维码或 logo。

2.2 图像加载与亮度通道提取

```
img_color = cv2.imread(...)
img_ycrcb = cv2.cvtColor(img_color, cv2.COLOR_BGR2YCrCb)
Y_channel = img_ycrcb[:, :, 0]
```

由于人眼对亮度更敏感，水印被嵌入 YCrCb 色彩空间中的 Y 通道，这一策略在视觉上更隐蔽，对人眼感知干扰较小。

图像随后被裁剪为 $block\_size$ 的整数倍大小，以保证可以完整分块进行 DCT 操作。

2.3 DCT 域水印嵌入

```
dct_block[embed_pos] += alpha if bit == 1 else -alpha
```

在每个 8×8 块的 DCT 变换结果中，选定位置的系数被调制以嵌入水印比特。嵌入公式如下：

$$D'(u, v) = D(u, v) + \alpha \cdot (2b - 1)$$

嵌入后的 DCT 块经逆变换（IDCT）重建后替换原图像块，实现水印写入。

2.4 彩色图像合成

```
img_ycrcb[:, :, 0] = Y_embed
watermarked_color = cv2.cvtColor(img_ycrcb, cv2.COLOR_YCrCb2BGR)
```

将嵌入水印后的 Y 通道重新与原 Cr、Cb 通道组合，并转换回 BGR 彩色图像，得到水印图像，保持了图像的色彩一致性和视觉可接受性。

2.5 攻击模拟

```
def attack_image(img, mode='flip'):
    ...
```

本实验设计了四种典型的图像攻击操作以测试水印鲁棒性：

- **flip**：水平翻转；
- **translate**：平移 10×10 像素；
- **crop**：中心裁剪后恢复尺寸；
- **contrast**：对比度增强。

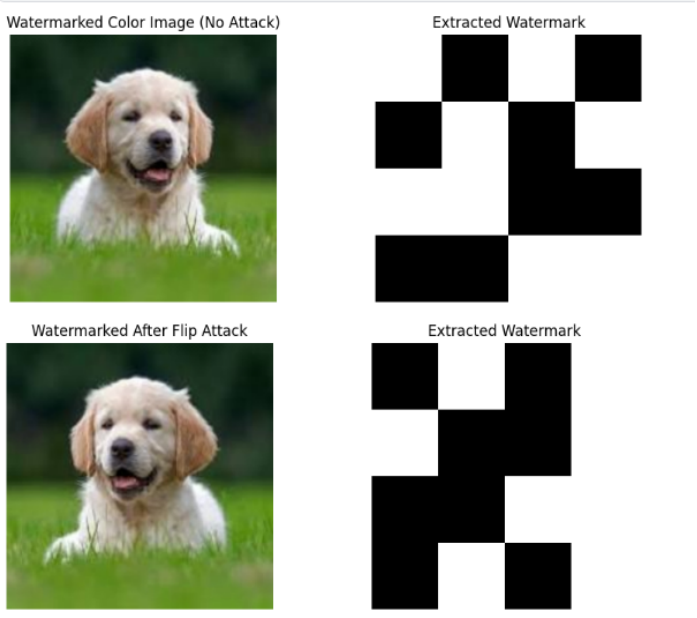
这些攻击模拟了图像在实际网络传播、编辑过程中可能遇到的干扰，有助于评估系统的鲁棒性。

2.6 水印提取与展示

```
dct_block = cv2.dct(block)
wm[idx // wm_w, idx % wm_w] = 1 if dct_block[embed_pos] > 0 else 0
```

提取过程同样基于 DCT，将图像划分为块后进行 DCT，读取对应嵌入位置系数的符号判断原始比特。该方法不依赖原图，属于**盲水印**提取方式。最终提取的水印以灰度图显示，与原始水印图案进行对比。

结果如下：



2.7 可视化与实验循环

```
for mode in ['flip', 'translate', 'crop', 'contrast']:
    attacked_color = attack_image(...)
    show_results(attacked_color, ...)
```

实验通过循环自动对嵌入图像施加不同攻击，并展示攻击后的图像与提取出的水印，便于观察鲁棒性表现。展示函数使用 Matplotlib 绘制原图和水印提取结果图，提升了可读性和分析效率。

2.8 实现总结

本系统采用模块化设计，将水印嵌入、攻击模拟和提取过程分离，实现逻辑清晰、实验可复现。DCT 域嵌入技术简单高效，且在多种图像攻击下具备一定鲁棒性，适合用于图像版权认证和数字签名等应用场景。

A DCT 代码

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from copy import deepcopy

# 参数设置
block_size = 8
embed_pos = (4, 4)
alpha = 10

# 二值水印矩阵（你可替换为二维码等）
watermark = np.array([
    [1, 0, 1, 0],
    [0, 1, 0, 1],
    [1, 1, 0, 0],
    [0, 0, 1, 1]
], dtype=np.uint8)

# 加载彩色图像
img_color = cv2.imread('/kaggle/input/dog22222/dog.jpg', cv2.IMREAD_COLOR)
if img_color is None:
    raise FileNotFoundError("图像加载失败，请检查路径。")

img_color = cv2.resize(img_color, (256, 256))

# 转换为 YCrCb 模式，提取 Y 通道
img_ycrcb = cv2.cvtColor(img_color, cv2.COLOR_BGR2YCrCb)
Y_channel = img_ycrcb[:, :, 0]

# 保证 Y 通道尺寸为 block_size 的整数倍
h, w = Y_channel.shape
Y_channel = Y_channel[:h // block_size * block_size, :w // block_size * block_size]

# DCT 域嵌入
def embed_watermark(Y, watermark, alpha):
    Y = np.float32(Y)
    wm_h, wm_w = watermark.shape
    idx = 0
    for i in range(0, Y.shape[0], block_size):
        for j in range(0, Y.shape[1], block_size):
            block = Y[i:i + block_size, j:j + block_size]
            dct_block = cv2.dct(block)
            if idx < wm_h * wm_w:
                bit = watermark[idx // wm_w, idx % wm_w]
                dct_block[embed_pos] += alpha if bit == 1 else -alpha
            block_idct = cv2.idct(dct_block)
```

```
            Y[i:i + block_size, j:j + block_size] = block_idct
            idx += 1
    return np.uint8(np.clip(Y, 0, 255))

#  提取水印
def extract_watermark(Y, shape, alpha):
    Y = np.float32(Y)
    wm = np.zeros(shape, dtype=np.uint8)
    wm_h, wm_w = shape
    idx = 0
    for i in range(0, Y.shape[0], block_size):
        for j in range(0, Y.shape[1], block_size):
            if idx >= wm_h * wm_w: break
            block = Y[i:i + block_size, j:j + block_size]
            dct_block = cv2.dct(block)
            wm[idx // wm_w, idx % wm_w] = 1 if dct_block[embed_pos] > 0 else 0
            idx += 1
    return wm

#  嵌入水印后合成回彩色图像
Y_embed = embed_watermark(Y_channel.copy(), watermark, alpha)
img_ycrcb[:, :, 0] = Y_embed
watermarked_color = cv2.cvtColor(img_ycrcb, cv2.COLOR_YCrCb2BGR)

#  攻击（彩色图）
def attack_image(img, mode='flip'):
    if mode == 'flip':
        return cv2.flip(img, 1)
    elif mode == 'translate':
        M = np.float32([[1, 0, 10], [0, 1, 10]])
        return cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
    elif mode == 'crop':
        h, w = img.shape[:2]
        cropped = img[h//4:3*h//4, w//4:3*w//4]
        return cv2.resize(cropped, (w, h))
    elif mode == 'contrast':
        return cv2.convertScaleAbs(img, alpha=1.5, beta=0)
    return img

# 显示结果
def show_results(attacked_img, title):
    ycrcb = cv2.cvtColor(attacked_img, cv2.COLOR_BGR2YCrCb)
    Y = ycrcb[:, :, 0]
    wm_extracted = extract_watermark(Y, watermark.shape, alpha)

    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    plt.title(title)
```

```
95      plt.imshow(cv2.cvtColor(attacked_img, cv2.COLOR_BGR2RGB))
96      plt.axis('off')
97      plt.subplot(1, 2, 2)
98      plt.title("Extracted Watermark")
99      plt.imshow(wm_extracted, cmap='gray')
100     plt.axis('off')
101     plt.show()
102
103 # 原图结果展示
104 show_results(watermarked_color, "Watermarked Color Image (No Attack)")
105
106 # 攻击测试
107 for mode in ['flip', 'translate', 'crop', 'contrast']:
108     attacked_color = attack_image(deepcopy(watermarked_color), mode)
109     show_results(attacked_color, f"Watermarked After {mode.capitalize()} Attack")
```