数字水印系统使用说明

1. 环境准备

1.1 安装依赖库

打开终端或命令提示符，导航到项目目录，执行以下命令安装所需依赖：
pip install -r requirements.txt
所需的主要依赖库包括：

- opencv-python：用于图像处理和DCT变换
- numpy：用于数值计算
- matplotlib：用于结果可视化

2. 基本使用流程

2.1 水印嵌入

要在图像中嵌入水印，请按照以下步骤操作：

1. 导入Watermark类：

```
from src.watermark import Watermark
```

2. 创建Watermark实例，指定密钥和水印大小：

```
# 使用自定义密钥
wm = Watermark(secret_key=12345, watermark_size=(8, 8))
```

3. 生成水印（可以使用随机水印或基于消息生成）：

```
# 生成随机水印
watermark = wm.generate_watermark()

# 或使用消息生成水印
# message = "MySecretMessage"
# watermark = wm.generate_watermark(message)
```

4. 嵌入水印：

```
# 嵌入水印并保存结果
wm.embed("input_image.png", watermark, "watermarked_image.png")
```

2.2 水印提取

要从图像中提取水印，请按照以下步骤操作：

1. 创建与嵌入时使用相同参数的Watermark实例：

```
# 必须使用与嵌入时相同的密钥
wm = Watermark(secret_key=12345, watermark_size=(8, 8))
```

2. 提取水印：

```
extracted_watermark = wm.extract("watermarked_image.png")
```

3. （可选）与原始水印比较相似度：

```
similarity = wm.calculate_similarity(original_watermark, extracted_watermark)
print(f"水印相似度: {similarity:.4f}")
```

3. 运行示例程序

项目提供了一个完整的示例程序，展示水印的嵌入、提取和鲁棒性测试过程：
python examples/example.py
示例程序将执行以下操作：

1. 创建或使用现有的示例图像
2. 生成水印并嵌入到图像中
3. 从带水印的图像中提取水印
4. 显示原始图像和带水印图像的对比
5. 对带水印的图像应用各种攻击
6. 从攻击后的图像中提取水印并显示相似度结果

示例程序的输出包括：

- 保存带水印的图像到`examples/watermarked.png`
- 保存各种攻击后的图像到`examples/attacked/`目录
- 保存图像对比结果到`examples/comparison.png`
- 保存鲁棒性测试结果到`examples/robustness_demo.png`

4. 运行自动化测试

项目提供了自动化测试程序，用于验证系统功能和水印的鲁棒性：
python -m src.test
测试程序将执行以下测试：

1. 验证水印嵌入和提取功能的正确性
2. 对带水印的图像应用各种攻击
3. 评估在不同攻击下水印的提取效果
4. 验证关键攻击下的水印相似度是否达到预期阈值

5. 自定义参数

您可以通过调整以下参数来优化水印性能：

1. **密钥（secret_key）**：

- 用于生成水印嵌入位置的伪随机序列
- 不同的密钥会导致不同的嵌入位置
- 必须在嵌入和提取时使用相同的密钥

2. **水印大小（watermark_size）**：
   - 默认为(8, 8)，即64位
   - 更大的水印可以携带更多信息，但可能降低鲁棒性
   - 建议保持为正方形，且边长为8的倍数
3. **水印强度（alpha）**：
   - 在Watermark类中定义，默认为0.05
   - 增大alpha可以提高鲁棒性，但可能降低图像质量
   - 减小alpha可以提高图像质量，但可能降低鲁棒性

调整示例：# 创建具有自定义参数的Watermark实例
wm = Watermark(secret_key=98765, watermark_size=(16, 16))

修改水印强度

wm.alpha = 0.07

6. 实际应用建议

1. **选择合适的图像**：
   - 建议使用分辨率较高的图像（至少256x256像素）
   - 避免使用过于简单的图像（如纯单色图像）
2. **密钥管理**：
   - 使用唯一的密钥为不同的用户或图像生成水印
   - 妥善保管密钥，防止未授权的水印提取或伪造
3. **水印验证**：
   - 设定合适的相似度阈值（通常建议0.7以上）
   - 低于阈值的结果可能表示图像被严重篡改或水印不存在
4. **多重水印**：
   - 对于重要应用，可以考虑嵌入多个水印（使用不同密钥）
   - 这可以提高安全性，并在部分水印被破坏时提供冗余
5. **性能权衡**：
   - 根据具体应用场景调整水印强度和大小
   - 对图像质量要求高的场景，使用较低的水印强度

- 对鲁棒性要求高的场景，使用较高的水印强度