浙江大学

本科实验报告

课程名称:	Matlab 图像处理编程实践
实验名称:	图像无损压缩
姓　　名:	付靖
学　　号:	3240102168
学　　院:	计算机科学与技术学院
专　　业:	计算机科学与技术
报告日期:	2025 年 9 月 16 日

实验任务简介

本次实验的任务是对图像进行压缩编码与解码，具体流程如下：

- 输入：一张灰度或真彩色图像；
- 功能：
  1. 利用 Haar 小波变换 对图像进行编码，并保存中间数据文件；
  2. 使用 Huffman 编码 对中间数据进行二进制压缩，并保存为压缩文件；
  3. 解码压缩文件，恢复图像并展示。

实验还将对比不同的压缩方法，包括：

1. 原图 -> Huffman 编码；
2. 原图 -> 差分编码 -> Huffman 编码；
3. 原图 -> Haar 小波编码 -> Huffman 编码。

通过对比这些方法，分析不同编码方式的压缩效率。

程序框架与技术细节

一、 总体框架

在本实验中，我们对比了三种不同的编码方式：

- 原图 → Huffman 编码：对图像的像素进行直接 Huffman 编码。这种方式适用于简单的无变换压缩，压缩效率较低，实现位于 huffman_length 函数中(huffman_length.m)。
- 原图 → 差分编码 → Huffman 编码：先对图像进行差分编码，再进行 Huffman 编码。这种方法通过消除图像中相邻像素间的差异，进一步提高 Huffman 编码的压缩效率。差分编码的实现位于 diff_encode 函数中(diff_encode.m)。
- 原图 → Haar 小波编码 → Huffman 编码：先对图像进行 Haar 小波变换，然后对变换后的系数进行 Huffman 编码。这种方法能够有效减少图像的数据冗余，提供最高的压缩效率，位于主文件（haar_main.m)。

主文件模块的具体功能：

本程序主要分为以下几个部分：

1. Haar 小波编码部分：使用 Haar 小波变换对图像进行编码。每个图像通道（灰度或彩色）会分别进行小波变换，得到变换系数。此部分代码实现位于 Haar_encode 函数中。

2. Huffman 编码部分：对小波编码后的图像系数使用 MATLAB 自带的 huffmandict 和 huffmanenco 函数进行压缩。通过计算每个系数的频率，构建 Huffman 字典，并进行编码。 此部分代码实现位于 Huffman_encode 函数中
3. 解码部分：从压缩的二进制文件中读取数据，使用 Huffman 解码恢复小波系数，再通过 Haar 小波逆变换恢复图像，分别实现于 Huffman_decode 和 Haar_decode 函数中。

函数结构与实现

- `Haar_encode`：对图像进行 Haar 小波变换，并对变换后的系数进行阈值处理（去除低频噪声）。

  输入文件：输入图像数据（从用户指定路径加载）。

  输出文件：

  coeffs_all.mat：包含所有图像通道（RGB 或灰度）的 Haar 小波变换系数。此文件保存了小波变换后的系数，并记录了图像的尺寸和 Haar 矩阵。

- `Huffman_encode`：将 Haar 编码后的系数使用 Huffman 编码压缩成二进制文件。

  输入文件：

  coeffs_all.mat：Haar 小波编码后的系数文件（由 Haar_encode 输出）。

  输出文件：

  output.bin：存储 Huffman 编码后的压缩数据（二进制文件）。

  dict.mat：存储生成的 Huffman 字典，用于编码和解码。

  len_file.mat：存储每个通道 Huffman 编码后的比特数（用于压缩率计算）。

- `Huffman_decode`：对压缩文件进行解码，恢复 Huffman 编码的系数。

  输入文件：

  output.bin：Huffman 编码后的压缩数据（二进制文件）。

  dict.mat：Huffman 字典，用于解码过程。

  len_file.mat：存储每个通道的 Huffman 编码长度，用于解码时定位不同通道的比特流。

  输出：解码后的图像系数（每个通道的系数），这些系数用于后续的 Haar 逆变换。

- `Haar_decode`：对解码后的系数进行逆 Haar 小波变换，恢复图像。

  输入文件：解码后的系数（由 Huffman_decode 输出）。

  输出：重建的图像。

- `diff_encode`：对图像进行差分编码，减少数据冗余。

- `huffman_length`：一个用于计算给定数据 Huffman 编码后的比特长度的辅助函数。它的主要作用是计算图像数据（或者差分编码后的数据）通过 Huffman 编码后的总比特数，从而帮助计算压缩率。

二、 技术细节

- 图像读取与处理：使用 imread 读取图像并用 imresize 将图像调整为指定的尺寸 L x L。这样可以确保图像大小一致，适合后续的 Haar 小波变换。
- Haar 小波变换：通过构造 Haar 矩阵并对图像进行二维矩阵乘法（左乘和右乘），实现 Haar 小波变换。
- Huffman 编码：通过 huffmandict 函数生成 Huffman 字典，然后使用 huffmanenco 进行编码。
- 差分编码：通过 diff_encode 函数对图像进行差分编码。
- 阈值处理: 为了压缩系数并去除不重要的高频信息，我们对小波变换后的系数应用了阈值处理。

  排序系数：对 Haar 小波变换后的系数按绝对值从大到小排序。

  设定阈值：选取最大的 50% 系数，其他的被丢弃。

  阈值化：小于阈值的系数置为零，保留有用信息。

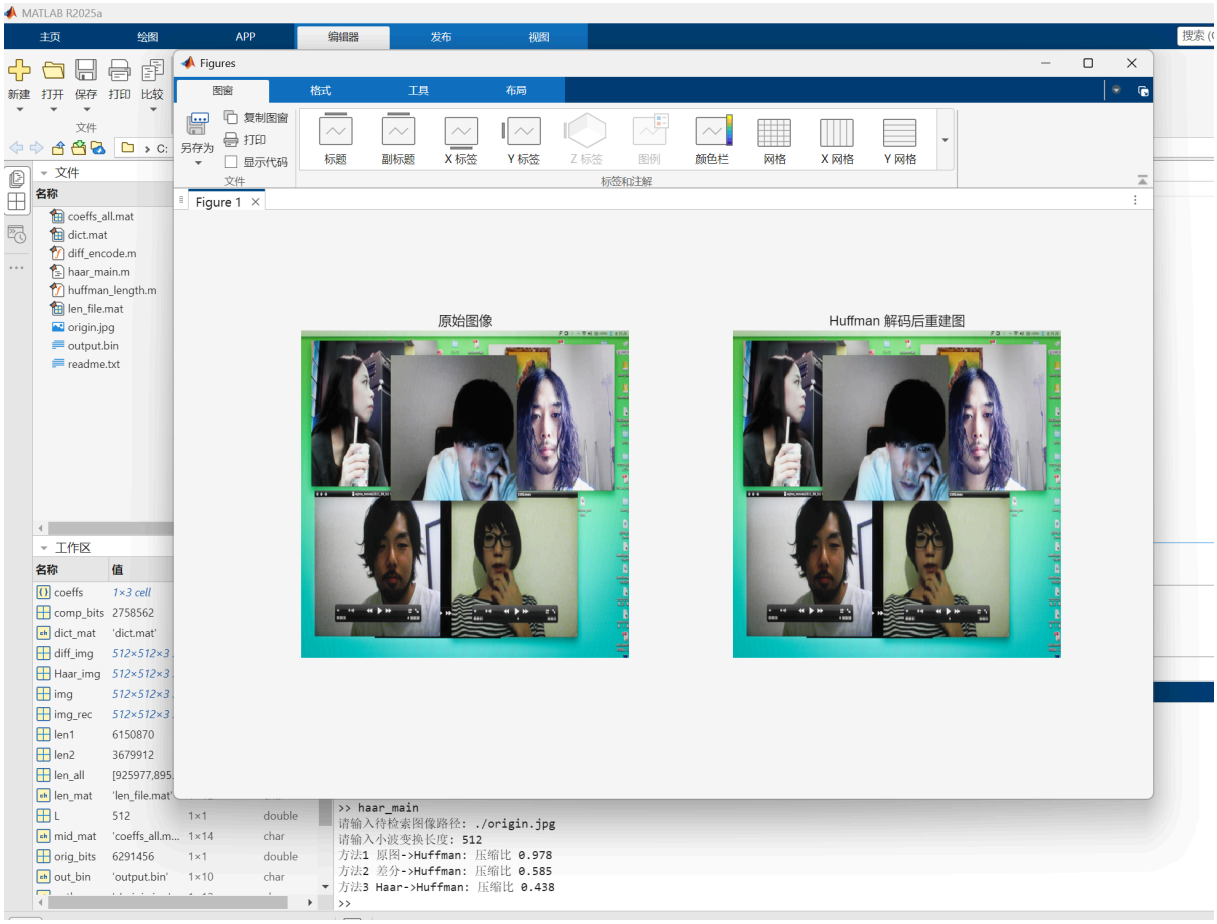
程序运行示例

- 运行主脚本。
- 输入图像路径以及小波变换长度（2^n），程序会根据此值调整图像大小。

```
请输入待检索图像路径：./origin.jpg
请输入小波变换长度：512
```

- 结果显示：程序会自动运行三种不同的压缩方法，并展示结果
  - 原图：原始图像。
  - 压缩图像：经过 Huffman 编码后，解码并重建的图像。
  - 压缩比：不同压缩方法的比率（在命令行）。

```
方法1 原图->Huffman：压缩比 0.978
方法2 差分->Huffman：压缩比 0.585
方法3 Haar->Huffman：压缩比 0.438
```



实验结果分析

在本实验中，我们对三种不同的图像压缩方法进行了比较：

压缩效率对比：

1. 原图 → Huffman 编码：

   这种方法直接对图像的像素进行编码，压缩比一般。压缩效果有限，尤其是在图像具有较高的冗余时。

2. 原图 → 差分编码 → Huffman 编码：

   通过对图像进行差分编码，消除了图像的平滑部分（例如，相邻像素差异较小的区域），提高了 Huffman 编码的效率。压缩比通常较好。

3. 原图 → Haar 小波编码 → Huffman 编码：

   这种方法通过 Haar 小波变换提取图像的低频信息，再进行 Huffman 编码。由于小波变换后系数的高频部分大多被压缩或去除，最终的压缩比最好，且对图像质量的影响最小。

结果分析：

1. 压缩比：从实验结果可以看出，采用 Haar 小波变换后的压缩比最小，这意味着它能显著减少文件的大小，尤其适用于图像压缩。
2. 图像质量：差分编码和 Haar 小波编码后的图像恢复效果较好，且可以通过调整阈值来控制压缩比与图像质量的平衡。
3. 计算复杂度：Haar 小波变换和 Huffman 编码结合的处理相对较为复杂，但它的压缩效果最好。差分编码相对简单，且能提供不错的压缩比。

总结：

通过比较三种压缩方法，我们可以得出：

- Haar 小波编码 + Huffman 编码是最优解，它在保持较高图像质量的同时，能达到较高的压缩比。
- 差分编码 + Huffman 编码在某些场合下也能提供较好的压缩效果，但压缩比较 Haar 方法略低。
- 对于一般的图像压缩任务，建议采用 Haar 小波变换和 Huffman 编码的结合方法。