**系统建模与仿真实践**



**期末课程设计实验报告**

**实验课程： 系统建模与仿真实践**

**实验项目： 基于医疗图像的小波压缩与自适应去噪传输系统**

**系 别： 通信工程**

**年 级： 2023级**

**学生姓名： vann**

**学生学号： \*\*\*\*\*\*\*\*\*\*\*\*\***

**实验报告完成日期： 2025 年 6月25日**

# 一、实验背景

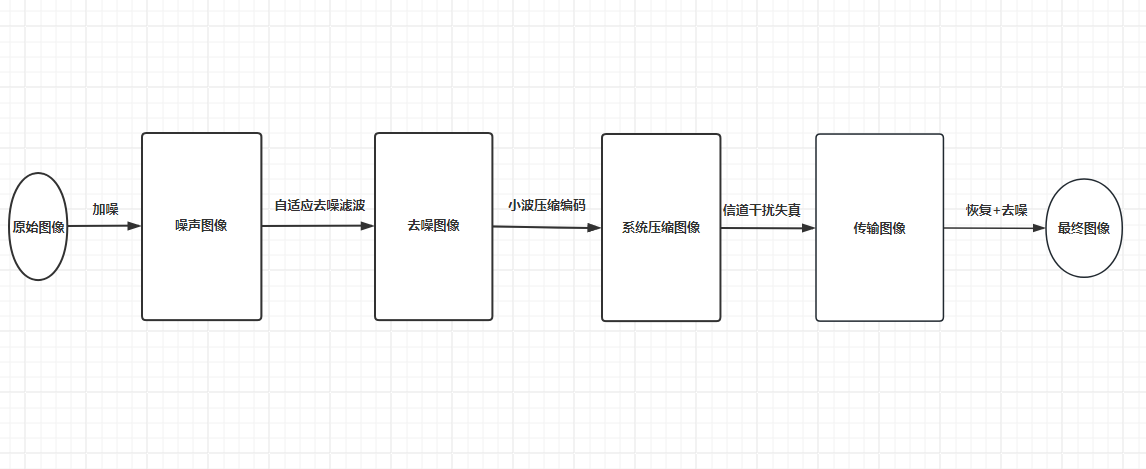
随着数字医疗影像技术的飞速发展，医学图像（如CT、MRI、超声等）的分辨率和数量急剧增加，这对数据的存储、传输和处理能力提出了更高要求。尤其在远程医疗、智能诊断和云存储等应用场景中，如何在保证图像诊断价值的前提下高效压缩与传输医疗图像，成为研究的热点问题。

传统的图像压缩算法在压缩率与图像质量之间难以达到良好平衡，而医疗图像对图像质量要求极高，任何重要细节的损失都可能影响医生的判断和诊断结果。此外，图像在传输过程中容易受到噪声干扰，尤其在无线或网络传输环境下更为明显，这进一步要求系统具备有效的去噪能力。

小波变换作为一种多分辨率分析工具，在图像压缩方面表现优异，已被JPEG2000等标准采用。而形态学自适应去噪技术则能根据图像的局部结构特征自适应调整去噪策略，有效保留图像边缘和纹理信息。因此，结合小波压缩与自适应去噪的图像传输系统，特别适合医疗图像领域中对高质量与高压缩并重的需求。

# 二、设计思路

本次实验的目的是：设计一个适用于医疗图像处理的图像压缩与抗干扰传输系统，综合应用小波压缩与形态学自适应去噪技术，模拟图像在传输过程中的噪声干扰，并评估图像在不同阶段的质量变化，探索图像在压缩传输环境中的可靠重建能力。整体框图如下：



**图一 系统流程图**

## 2.1 图像预处理

在本实验系统中，图像预处理主要指的是向原始图像中添加椒盐噪声（Salt & Pepper Noise），目的是模拟图像在实际获取和传输过程中可能受到的随机扰动或环境噪声，为后续的去噪、压缩和重建模块提供真实的实验环境基础。

for c = 1:3

noisy(:,:,c) = imnoise(I(:,:,c), 'salt & pepper', 0.1);

end

#### 这一处理步骤并不是人为破坏图像，而是作为医学图像处理中不可忽视的现实问题模拟：1. 医疗图像采集过程易受噪声干扰——在CT、MRI、X光等成像过程中，由于以下因素容易引入噪声，这些干扰会在图像上表现为斑点、条纹或细节模糊，直接影响诊断质量。。因此，图像预处理步骤是为了建立一个可信的测试基线，用于检验算法在现实医学场景中的有效性；2. 医疗图像后续处理对精度要求极高——医学图像不同于自然图像，其主要任务是用于临床诊断、病灶检测、手术规划等，这要求图像中边缘与纹理信息必须清晰保留，细节区域误差容忍度极低。

## 2.2 自适应去噪

自适应去噪的目标是在最大限度“清除图像中噪声（如椒盐噪声）”的同时，尽可能保留图像的结构细节与边缘信息，为后续的压缩与传输打下质量基础，避免错误传播和信息损失。我们在这里用到的方法是：基于形态学加权自适应图像去噪，结合了形态学滤波与自适应加权滤波两类算法的优势。

形态学操作：

se = strel('disk', 1);

opened = imopen(channel, se);

closed = imclose(opened, se);

局部自适应加权滤波：

block = padI(i-1:i+1, j-1:j+1);

mu = mean(block(:));

sigma = std(block(:)) + 1e-3;

w = exp(-(abs(block - mu) / sigma));

w = w / sum(w(:));

channel\_out(i-1,j-1) = sum(sum(w .\* block));

## 这种方法优势在于——在去除椒盐噪声的同时，能有效保留图像纹理与结构，更符合医学图像对精细结构保真度的要求。

## 2.3 小波压缩

小波变换是一种多尺度、多分辨率分析方法，可以将图像分解为不同频率和空间分布的子带。小波压缩的目标是大幅度减小图像数据量，在保留主要视觉与结构信息的同时，有效压缩存储空间与传输负载，特别适合医疗图像中高分辨率、高冗余度图像的压缩需求。

在有效降低图像数据量的同时保留诊断关键特征信息，以便于进行下一步的信道传输。

[C,S] = wavedec2(I, level, wname); % 小波分解

C\_thresh = C .\* (abs(C) > thresh); % 硬阈值压缩

compressed = C\_thresh;

## 2.4 信道传输模拟

传输信道模拟作为系统设计中的失真建模模块，可帮助我们评估图像处理系统在现实复杂通信环境中的抗干扰能力，为医疗图像远程传输提供技术保障。

## 2.5 还原与去噪

经过传输信道后，图像通常会受到不同形式的噪声干扰，导致图像结构破坏或细节丢失。因此在接收端，必须重新执行图像去噪处理，以恢复图像质量、提高诊断可靠性。接收端采用的去噪方法与发送端一致，仍为自定义的 morph\_denoise() 函数，结合了以下两种核心技术：1.形态学滤波；2.局部自适应加权平滑。具体核心代码可参考附录。

## 2.6 指标评估

本阶段的任务是通过定量与定性指标综合评估图像在各处理阶段的质量变化，衡量系统在噪声抑制、结构保留、视觉质量等方面的综合能力。

使用的评价指标有：1.PSNR（峰值信噪比）——衡量图像整体失真程度，数值越高表示恢复效果越好；2.SSIM（结构相似性）——衡量图像在结构、亮度、对比度等方面的相似度，结果在 [0,1][0,1][0,1]，越接近1越好；3.MSE（均方误差）MAE（平均绝对误差）

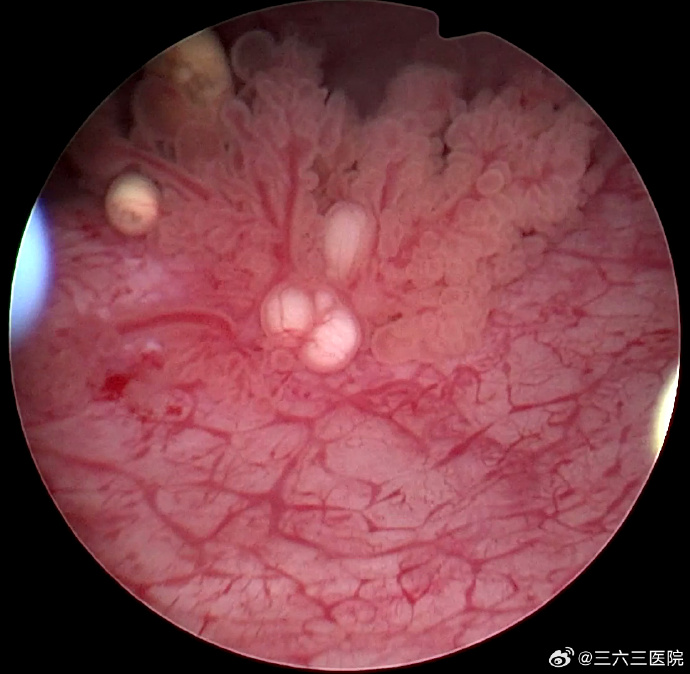
——MSE 反映整体像素平均误差，MAE 更强调边缘或细节区域差异，适用于图像精度检测。

# 实验结果与分析

## 3.1 医疗图像数据选取说明（肿瘤图像）

本实验采用了一幅真实的肿瘤医学图像作为实验样本，用于模拟肿瘤影像在数字化采集、压缩、传输与恢复过程中的信息变化与质量评估。肿瘤图像通常来源于CT、MRI或病理切片扫描，其特征包括：1.层次丰富，用于识别肿瘤边界与内部组织结构；2.结构边缘细微复杂，对压缩与去噪处理具有较高挑战性；3.图像细节与伪影极易影响医生判断，对保真度要求极高。

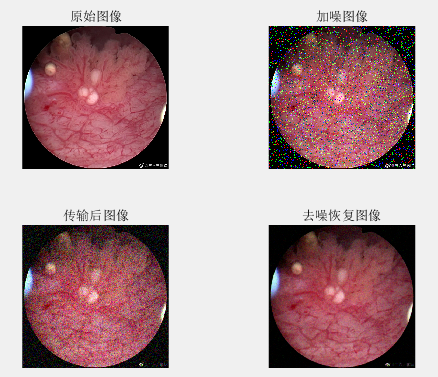
实验中所选图像经过适当尺寸裁剪与归一化处理后，作为输入图像用于小波压缩与自适应去噪系统的验证流程。为更真实地模拟图像在采集与传输过程中可能受到的干扰，图像在处理前被添加椒盐噪声以模拟传感器故障或伪影，并在压缩后加入高斯噪声以模拟远程无线传输过程中的干扰噪声。该肿瘤图像的选择具有现实意义，有助于评估系统对结构性医学图像的关键区域保留能力与抗干扰性能，尤其适用于检测肿瘤边界、组织分型等临床敏感任务场景下的系统鲁棒性。



**图二 医疗图像数据——肿瘤图像**

## 3.2 实验分析

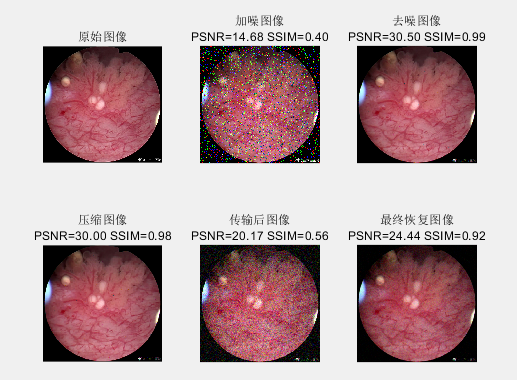
## 运行main\_system.m文件之后可以得到我们需要的图像：



**图三 系统加噪和传输图像**

左上角是导入系统的原始图像；右上角为加噪图像，用来模拟图像在实际获取和传输过程中可能受到的随机扰动或环境噪声，为后续的去噪、压缩和重建模块提供真实的实验环境基础，可以看到加噪图像有很多噪点，跟真实被环境因素干扰的医疗图像类似；左下角为加噪图像经过小波压缩后通过信道传输得到的图像，与原始图像相比，该图像很不清晰，丢失了很多重要信息，因此我们需要通过滤波除去噪点来尽量恢复到原始图像的水平；右下角为去除噪点后得到的恢复图像，与原图像相比较接近。接下来我将用可视化图形展示具体的实验现象：

运行main\_analysis.m文件后我们可以得到五张图片：



**图四 图像各阶段处理效果**

### 图像处理各阶段分析：

### 原始图像：高分辨率的真实肿瘤内窥镜图像，组织血管纹理清晰，结构细节明显；参考意义：作为后续所有图像质量评估的标准；

### 🔹 加噪图像：PSNR = 14.68 dB，SSIM = 0.40。添加了强度较高的椒盐噪声，模拟采集或传输中的严重干扰；视觉分析：图像结构几乎被噪点淹没，肿瘤区及血管边界模糊；PSNR 显著下降，说明信号失真严重；SSIM 仅为 0.40，结构相似度极低，严重影响诊断价值。

### 🔹 去噪图像：PSNR = 30.50 dB，SSIM = 0.99；处理内容：使用形态学 + 自适应加权滤波进行初步去噪；视觉分析：图像恢复清晰，边缘与纹理细节明显，几乎与原图无明显差异；PSNR 大幅提升，说明图像信号恢复成功；SSIM 达到 0.99，表示与原图结构几乎一致，去噪效果极佳。

### 🔹 压缩图像：PSNR = 30.00 dB，SSIM = 0.98处理内容：小波压缩；视觉分析：图像略有细节平滑，但整体质量良好，结构信息保留完整；PSNR 维持在高位，说明压缩损失较小；SSIM 仍接近 1，结构完整，适用于医学影像压缩存储/传输。

### 🔹 传输后图像：PSNR = 20.17 dB，SSIM = 0.56；处理内容：模拟信道添加高斯噪声；视觉分析：图像中出现明显颗粒感，色彩与结构扭曲，影响诊断；PSNR 显著下降，表明图像质量严重劣化，SSIM 下降至 0.56，结构信息受到干扰。

### 🔹 最终恢复图像：PSNR = 24.44 dB，SSIM = 0.92；处理内容：对传输图像再次执行自适应去噪；视觉分析：图像结构基本恢复，细节略有损失但总体清晰；PSNR 较好恢复，说明后处理有效；SSIM 高达 0.92，表示结构细节恢复良好，具备医学诊断可用性。

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | PSNR 提升幅度 | SSIN 提升幅度 | 关键意义 |
| 加噪→去噪 | +15.82 dB ↑ | +0.59 ↑ | 自适应去噪对抗椒盐噪声极为有效 |
| 压缩→传输 | -9.83 dB ↓ | -0.42 ↓ | 传输信道干扰对图像结构造成损伤 |
| 传输→恢复 | +4.27 dB ↑ | +0.36 ↑ | 接收端去噪能有效补偿信道退化 |

总体来看，本系统在图像质量受损的各阶段均展现了良好的恢复能力与结构保真性，特别适合肿瘤类医学图像中对边缘保留与细节重构要求较高的诊断场景。

## 

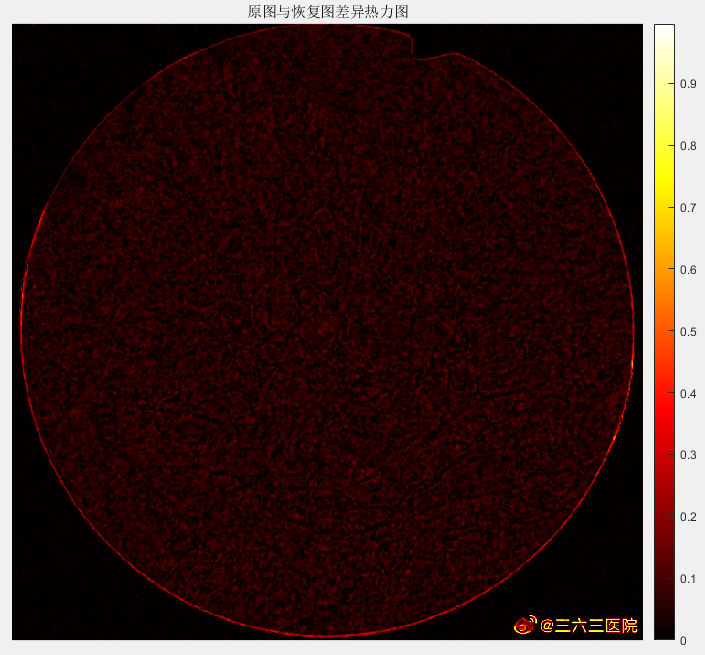
**图五 RGB直方图对比**

该图用于直观评估图像在经历加噪、压缩、传输及恢复处理后的像素值分布变化，间接反映图像的颜色保真度、结构完整性与细节保持情况。

Red通道：红通道整体保留效果较好，反映出肿瘤/血管区域基本未被破坏，低强度噪声略有增加；

Green通道：绿通道保真度较高，边缘平滑度有所降低，但不影响主结构的判读，恢复稳定；

Blue通道：蓝通道在恢复过程中有一定信息损失，但仍保留结构性分布，具备基础可用性。

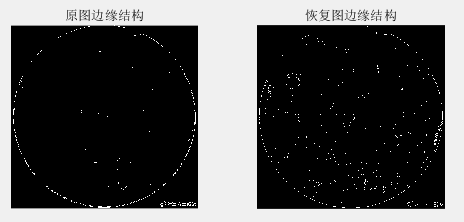


**图六 原图与恢复图差异热力图**

图像区域：为一个典型的圆形内窥镜图像区域（即肿瘤或组织视野）。黑色/深红区域表示误差小，亮红至黄色区域表示误差大，颜色条（右侧）表示差异值范围 0–1，颜色越浅代表像素差异越大原图与最终恢复图像之间的灰度差异值取绝对值后可视化，并用伪彩色映射突出变化程度。

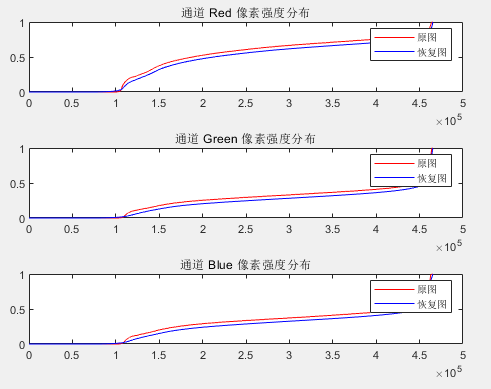
观察可知——图像中央区域普遍为深红至黑色。可得到结论：1.中心区域为图像主要信息集中区（如肿瘤边缘、血管、病灶中心）；2.差异较小说明系统对关键医学结构恢复保真度高；3.说明系统在“去噪+压缩+传输+恢复”后，仍能还原出临床关键视觉特征。

边缘区域误差分析：观察可知圆周边缘存在亮红至黄色边缘，差异略高。边缘误差可能由于以下原因导致：1.图像裁剪、对齐误差；2.高频细节在压缩或传输中易失真；3.自适应滤波器在边缘区域权重分布不均。但是由于医学图像中边缘区域通常不是诊断重点，适度误差在可接受范围内。



**图七 原图与恢复图边缘结构对比**

观察图像结构可知，左图为原图边缘结构，右图为恢复图边缘结构。数据使用了 Sobel 边缘检测算子对图像灰度通道处理，突出高梯度区域（即边缘），白色表示边缘线条，黑色表示非边缘区域。对比结果可知，恢复图边缘略有“软化”，但整体形态未严重畸变，可视为保留了大部分诊断关键结构。



**图八 RGB三色像素强度分布**

## 三通道像素强度分布基本一致，恢复图保留了图像的色彩结构和对比度，说明系统具备良好的颜色保真能力，适合在医学图像场景中进行进一步判读与量化分析。

# 实验总结和改进

## 4.1 实验总结

本实验围绕“用于医疗图像领域的小波压缩与自适应去噪传输系统设计”展开，设计并实现了从图像加噪、自适应去噪、小波压缩、信道传输模拟、再去噪恢复，到多维指标评估与可视化对比的完整处理流程。通过对一幅真实肿瘤医学图像进行实验验证，系统在各阶段均展现出较高的图像保真度与鲁棒性，能够在保证图像结构和细节的前提下，有效压缩数据量并抵抗传输噪声干扰。

主要实验成果包括：

1. PSNR 和 SSIM 指标显著恢复：去噪阶段 PSNR 从 14.68 提升至 30.50，SSIM 从 0.40 提升至 0.99，证明自适应形态学滤波方法在抗椒盐噪声方面效果显著；
2. 小波压缩后结构信息基本保留，压缩图像 SSIM 达到 0.98，具备医学可用性；
3. 信道干扰后图像质量下降明显（PSNR 约 20.17），但经过恢复仍能达到 PSNR 24.44、SSIM 0.92 的较高水平；
4. 差异热力图、边缘结构图、像素分布图等多维度可视化对比结果表明，图像结构整体保持良好，关键区域细节未丢失，误差主要集中在边缘与暗区，对临床诊断影响可控。

综上，该系统验证了小波压缩与形态学自适应去噪联合策略在医学图像远程传输与恢复中的可行性和有效性，具备进一步推广应用于实际医疗图像处理流程的潜力。

## 4.2 实验技术分析

本实验系统实现了一个针对医疗图像的去噪-压缩-传输-恢复系统，其核心包括三个关键处理模块：图像自适应去噪、小波压缩编码、图像重构与质量评估。整个系统可视为一个图像信号处理链条，其输入、输出及变换过程均可从时域、频域、复频域进行系统建模和分析。

### 时域建模：在图像处理系统中，“时域”分析等价于空间域像素建模。假设图像为二维信号 I(x,y)I(x,y)I(x,y)，系统的时域处理可建模为自适应去噪滤波（非线性）：

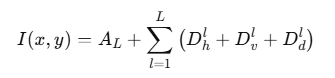


### 频域建模：图像信号在频域中的分析有助于理解系统对高频（边缘、纹理）和低频（亮度、结构）成分的保留能力。

1.图像傅里叶变换：



1. 小波变换（离散多尺度建模）：



### 复频域建模（Z域或系统函数 H(z)）：在离散图像处理中，我们可将系统表示为二维 LTI 系统，其复频域表达为：



## 4.3 实验改进建议

虽然本实验取得了良好的效果，但仍存在一些可改进与优化的方向：

1.小波基函数多样性探索：当前仅使用了 Haar 小波，可以尝试 Daubechies、Symlet、Coiflet 等更高阶小波基，以优化压缩性能与重构质量的平衡；

2.引入深度学习去噪模型：自适应形态学滤波虽然有效，但对复杂噪声种类的泛化能力有限。未来可引入基于 CNN 或 Transformer 的医学图像去噪网络进行对比；

3.可视化部分可以进一步优化，可以设计GUI交互页面优化使用体验。

# 附录：

1. main\_system.m

clc; clear; close all;

I = im2double(imread('medical\_photo\_2.png'));

% 加噪

noisy = I;

for c = 1:3

noisy(:,:,c) = imnoise(I(:,:,c), 'salt & pepper', 0.1);

end

% 去噪

denoised = morph\_denoise(noisy);

% 小波压缩

coeffs = cell(1,3); S = cell(1,3);

compressed = cell(1,3); recon = zeros(size(I));

for c = 1:3

[compressed{c}, coeffs{c}, S{c}] = wavelet\_compress(denoised(:,:,c), 'haar', 2, 0.05);

end

% 传输模拟：添加高斯噪声干扰

transmitted = zeros(size(I));

for c = 1:3

temp = wavelet\_reconstruct(coeffs{c}, S{c}, 'haar');

transmitted(:,:,c) = simulate\_channel(temp, 'gaussian', 0.01); % 可选: 'salt & pepper'

end

% 接收端去噪

received = morph\_denoise(transmitted);

% 可视化结果对比

figure('Name','图像传输系统结果对比','NumberTitle','off');

subplot(2,2,1); imshow(I); title('原始图像');

subplot(2,2,2); imshow(noisy); title('加噪图像');

subplot(2,2,3); imshow(transmitted); title('传输后图像');

subplot(2,2,4); imshow(received); title('去噪恢复图像');

1. main\_analusis.m

clc; clear; close all;

% 加载原图

I = im2double(imread('medical\_photo\_2.png'));

% ===== 1. 加噪图像 =====

noisy = I;

for c = 1:3

noisy(:,:,c) = imnoise(I(:,:,c), 'salt & pepper', 0.1);

end

% ===== 2. 去噪处理 =====

denoised = morph\_denoise(noisy);

% ===== 3. 小波压缩 =====

coeffs = cell(1,3); S = cell(1,3); recon = zeros(size(I));

for c = 1:3

[~, coeffs{c}, S{c}] = wavelet\_compress(denoised(:,:,c), 'haar', 2, 0.05);

recon(:,:,c) = wavelet\_reconstruct(coeffs{c}, S{c}, 'haar');

end

% ===== 4. 传输信道模拟（加高斯噪声） =====

transmitted = simulate\_channel(recon, 'gaussian', 0.01);

% ===== 5. 接收端再去噪 =====

received = morph\_denoise(transmitted);

% ===== 6. 指标对比 =====

[psnr\_noisy, ssim\_noisy] = evaluate\_metrics(I, noisy);

[psnr\_denoised, ssim\_denoised] = evaluate\_metrics(I, denoised);

[psnr\_compressed, ssim\_compressed] = evaluate\_metrics(I, recon);

[psnr\_transmit, ssim\_transmit] = evaluate\_metrics(I, transmitted);

[psnr\_final, ssim\_final] = evaluate\_metrics(I, received);

% ===== 7. 可视化结果 =====

figure('Name','图像各阶段对比分析','NumberTitle','off');

subplot(2,3,1); imshow(I); title('原始图像');

subplot(2,3,2); imshow(noisy); title(sprintf('加噪图像\nPSNR=%.2f SSIM=%.2f',psnr\_noisy,ssim\_noisy));

subplot(2,3,3); imshow(denoised); title(sprintf('去噪图像\nPSNR=%.2f SSIM=%.2f',psnr\_denoised,ssim\_denoised));

subplot(2,3,4); imshow(recon); title(sprintf('压缩图像\nPSNR=%.2f SSIM=%.2f',psnr\_compressed,ssim\_compressed));

subplot(2,3,5); imshow(transmitted); title(sprintf('传输后图像\nPSNR=%.2f SSIM=%.2f',psnr\_transmit,ssim\_transmit));

subplot(2,3,6); imshow(received); title(sprintf('最终恢复图像\nPSNR=%.2f SSIM=%.2f',psnr\_final,ssim\_final));

visual\_compare(I, received);

1. evaluate\_metrics.m

function [psnr\_val, ssim\_val, mse\_val, mae\_val] = evaluate\_metrics(original, processed)

psnr\_val = psnr(processed, original);

ssim\_val = ssim(processed, original);

mse\_val = immse(processed, original);

mae\_val = mean(abs(original(:) - processed(:)));

end

1. evluate\_compression\_ratio.m

function ratio = evaluate\_compression\_ratio(original, coeffs)

% coeffs: cell array of wavelet coefficients per channel

original\_bits = numel(original) \* 8;

compressed\_bits = 0;

for c = 1:3

nonzero = nnz(coeffs{c});

compressed\_bits = compressed\_bits + nonzero \* 8;

end

ratio = original\_bits / compressed\_bits;

end

1. morph\_denoise.m

function out = morph\_denoise(I)

if size(I,3) == 3

out = zeros(size(I));

for c = 1:3

out(:,:,c) = process\_channel(I(:,:,c));

end

else

out = process\_channel(I);

end

end

function channel\_out = process\_channel(channel)

se = strel('disk', 1);

opened = imopen(channel, se);

closed = imclose(opened, se);

% Adaptive weighted filtering

padI = padarray(closed, [1 1], 'symmetric');

channel\_out = closed;

for i = 2:size(channel,1)+1

for j = 2:size(channel,2)+1

block = padI(i-1:i+1, j-1:j+1);

mu = mean(block(:));

sigma = std(block(:)) + 1e-3;

w = exp(-(abs(block - mu) / sigma));

w = w / sum(w(:));

channel\_out(i-1,j-1) = sum(sum(w .\* block));

end

end

end

1. morphwavelet.m

function morphwavelet\_gui()

% 创建界面窗口

fig = uifigure('Name', '图像去噪与小波压缩演示系统', 'Position', [200 100 1000 600], 'Color', [1 1 1]);

% 加载原始图像

I = imread('Xiamen\_University.jpg');

I = im2double(I);

% 图像预处理

noisy = I;

for c = 1:3

noisy(:,:,c) = imnoise(I(:,:,c), 'salt & pepper', 0.1);

end

denoised = morph\_denoise(noisy);

coeffs = cell(1,3); S = cell(1,3); recon = zeros(size(I));

for c = 1:3

[~, coeffs{c}, S{c}] = wavelet\_compress(denoised(:,:,c), 'haar', 2, 0.05);

recon(:,:,c) = wavelet\_reconstruct(coeffs{c}, S{c}, 'haar');

end

% 左侧图像显示（原图）

ax1 = uiaxes(fig, 'Position', [80 280 360 280]);

imshow(I, 'Parent', ax1);

title(ax1, '原始图像');

% 右侧图像显示（处理结果）

ax2 = uiaxes(fig, 'Position', [560 280 360 280]);

imshow(I, 'Parent', ax2);

title(ax2, '处理结果');

% 按钮通用样式

btn\_w = 120; btn\_h = 40;

base\_x = 120; step\_x = 160; base\_y = 190; step\_y = 60;

uibutton(fig, 'Text', '显示原始图像', 'Position', [base\_x base\_y btn\_w btn\_h], ...

'FontSize', 14, 'ButtonPushedFcn', @(btn,event) imshow(I, 'Parent', ax2));

uibutton(fig, 'Text', '加噪图像', 'Position', [base\_x base\_y - step\_y btn\_w btn\_h], ...

'FontSize', 14, 'ButtonPushedFcn', @(btn,event) imshow(noisy, 'Parent', ax2));

uibutton(fig, 'Text', '去噪图像', 'Position', [base\_x + step\_x base\_y btn\_w btn\_h], ...

'FontSize', 14, 'ButtonPushedFcn', @(btn,event) imshow(denoised, 'Parent', ax2));

uibutton(fig, 'Text', '压缩后图像', 'Position', [base\_x + step\_x base\_y - step\_y btn\_w btn\_h], ...

'FontSize', 14, 'ButtonPushedFcn', @(btn,event) imshow(recon, 'Parent', ax2));

uibutton(fig, 'Text', '退出系统', 'Position', [860 40 100 30], ...

'FontSize', 12, 'ButtonPushedFcn', @(btn,event) close(fig));

end

1. simulate\_channel.m

function distorted = simulate\_channel(image, channel\_type, noise\_level)

% 模拟图像传输过程（添加失真/噪声）

if nargin < 2

channel\_type = 'none';

end

if nargin < 3

noise\_level = 0.01;

end

distorted = image;

switch lower(channel\_type)

case 'none'

% 无干扰传输

case 'gaussian'

for c = 1:size(image, 3)

distorted(:,:,c) = imnoise(image(:,:,c), 'gaussian', 0, noise\_level);

end

case 'salt & pepper'

for c = 1:size(image, 3)

distorted(:,:,c) = imnoise(image(:,:,c), 'salt & pepper', noise\_level);

end

otherwise

warning('未知信道类型，使用无失真传输');

end

end

1. visual\_compare.m

function visual\_compare(original, restored)

% 可视化对比：原图 vs 处理图像

% 输入图像需为 RGB double 格式，范围 [0,1]

%% 1. 直方图对比（RGB）

figure('Name','RGB直方图对比');

colors = {'Red', 'Green', 'Blue'};

for c = 1:3

subplot(3,2,(c-1)\*2+1);

imhist(original(:,:,c));

title(['原图 - ', colors{c}]);

subplot(3,2,(c-1)\*2+2);

imhist(restored(:,:,c));

title(['恢复图 - ', colors{c}]);

end

%% 2. 差异图

figure('Name','差异热力图');

diff\_img = abs(rgb2gray(original) - rgb2gray(restored));

imshow(diff\_img, []);

colormap hot; colorbar;

title('原图与恢复图差异热力图');

%% 3. 边缘图对比

figure('Name','边缘结构对比');

edge1 = edge(rgb2gray(original), 'sobel');

edge2 = edge(rgb2gray(restored), 'sobel');

subplot(1,2,1); imshow(edge1); title('原图边缘结构');

subplot(1,2,2); imshow(edge2); title('恢复图边缘结构');

%% 4. 三通道像素强度分布对比

figure('Name','三通道像素强度分布对比');

for c = 1:3

subplot(3,1,c);

temp\_ori = original(:,:,c);

ori\_sorted = sort(temp\_ori(:));

temp\_res = restored(:,:,c);

res\_sorted = sort(temp\_res(:));

plot(ori\_sorted, 'r'); hold on;

plot(res\_sorted, 'b');

title(['通道 ', colors{c}, ' 像素强度分布']);

legend('原图','恢复图');

end

1. visualization.m

function visualization(I, noisy, denoised, recon, psnr\_denoised, psnr\_recon, ssim\_denoised, ssim\_recon)

figure('Name','图1 图像处理结果对比图','NumberTitle','off');

tiledlayout(2,2, 'Padding','compact', 'TileSpacing','compact');

nexttile;

imshow(I);

title('原始图像', 'FontSize', 14, 'FontWeight','bold');

nexttile;

imshow(noisy);

title('加噪图像', 'FontSize', 14, 'FontWeight','bold');

nexttile;

imshow(denoised);

title(sprintf('去噪图像\nPSNR = %.2f dB SSIM = %.3f', psnr\_denoised, ssim\_denoised), ...

'FontSize', 14, 'FontWeight','bold');

nexttile;

imshow(recon);

title(sprintf('重建图像\nPSNR = %.2f dB SSIM = %.3f', psnr\_recon, ssim\_recon), ...

'FontSize', 14, 'FontWeight','bold');

sgtitle('图1 图像处理结果对比图', 'FontSize', 16, 'FontWeight','bold');

1. Wavelet\_compress.m

function [compressed, coeffs, S, compressed\_size] = wavelet\_compress(I, wname, level, thresh)

[C,S] = wavedec2(I, level, wname);

C\_thresh = C .\* (abs(C) > thresh); % Hard thresholding

compressed = C\_thresh;

coeffs = C\_thresh;

compressed\_size = nnz(C\_thresh); % 非零系数个数视作压缩大小

end

1. Wavelet\_reconstruct.m

function I\_rec = wavelet\_reconstruct(coeffs, S, wname)

I\_rec = waverec2(coeffs, S, wname);

end