南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

信息隐藏技术实验十

研读文献

穆禹宸　2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：李朝晖

2023 年 5 月 13 日

摘要

利用 MATLAB 对研读的图像加密文献进行实践。探索了多种混沌系统在图像加密和信息隐藏上的应用，研究了文献所提出的相关方法的优缺点，并根据最新研究提出并实现改进方案。

关键字：图像加密，信息隐藏，混沌系统，Clifford，logitsic 映射，分块置乱，扩散

目录

一、 实验介绍

(一) 论文选择

本文主要基于赵雨等人发表的《**基于混沌映射的图像加密算法研究**》[2] 进行实现，同时，在研究了原始论文的优缺点之后，我又基于张文宇等人发表的《**基于改进 Clifford 混沌系统的图像加密算法**》[1] 提出并实现了改进方案。

(二) 实验环境

所需实验环境

主要环境如下：

(1) 运行系统：Windows11

(2) 实验工具：Matlab2022a

(3) 数据：JPG 格式图像

(三) 提交文件

所提交文件的组织形式

主要有以下部分：

1. 原始论文以及代码
   - (a)《基于混沌映射的图像加密算法研究》论文
   - (b) chongpai.m（辅助函数）
   - (c) differential.m（辅助函数）
   - (d) Information_entropy.m（辅助函数）
   - (e) niyiwei.m（辅助函数）
   - (f) re_lativity.m（辅助函数）
   - (g) sort_mat.m（辅助函数）
   - (h) yiwei.m（辅助函数）
   - (i) jiamizuizhong.mat（加密后矩阵）
   - (j) encrypt.m（加密过程）
   - (k) decrypt.m（解密过程）
   - (l) 实验图像若干
2. 基于 Clifford 的改进代码
   - (a)《基于改进 Clifford 混沌系统的图像加密算法》论文
   - (b) Clifford.m（实现代码）
   - (c) lena.jpg（lena 彩色原图）
   - (d) new1.jpg（加密后图像）

3. 展示 ppt

4. 实验报告

二、 基于混沌映射的图像加密算法研究

(一) 简述

为满足安全性与算法最优等要求，针对一维混沌系统不能满足要求的缺点，所以本文提出一种基于二维混沌映射的图像加密算法。利用 Tent 映射和改进型 Logistic 映射两种混沌模型并结合比特重排技术来生成混沌序列，先利用 Arnold 变换对图像进行预加密，随后利用混沌序列对其进行异或、索引矩阵排序、左循环移位的位数等操作。加密完成后，对图像仿真结果的分析与测试，其中密文图像直方图统计特性均匀平滑，与其他文献相比也具有一定优势，实现了图像的安全加密效果。

(二) 常用的混沌系统介绍

1. Logistic 映射

Logistic 映射是应用最为广泛的一种混沌映射，其在研究时间离散的动力系统时具有较好的特性，且对于研究混沌以及分形控制等方向是一个经典模型。其数学表达式为

$$x_{n+1} = \mu x_n(1 - x_n)\ x \in (0, 1) \tag{1}$$

式中：x∈（0，1）为第 n 个混沌位置；μ 为控制参数，当 $\mu \in (3.56994\ 4]$ 时，系统会不断迭代使之出现其该有的混沌特性。据图 1 可知，当 Logistic 映射进入到完全混沌状态时的分叉图。

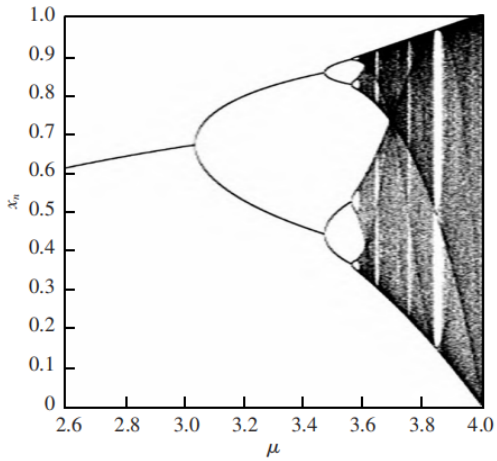

图 1: Logistic 映射分岔图

2. 改进型 Logistic 映射

由它所迭代出来的混沌序列值出现吸引子与空白区问题，使得这样的一个混沌方程不满足参数设置要求，由此导致加密效果的安全性下降问题。常用的 Logistic 映射存在低效率性的问题，希望能用一个函数表达式来代替参数 μ，以解决以上混沌系统中为得到最佳控制参数 μ 而导致

的常数化问题。改进型 Logistic 映射方程为

$$x_{n+1}\ 1 - v{x_n}^2 \tag{2}$$

式中：x_n 为映射变量；v 为映射参量；x_n 和 v 的取值范围分别为：$-1 < x_n < 1$，$0 <$ v 2。改进型 Logistic 映射分岔图如图 2 所示。

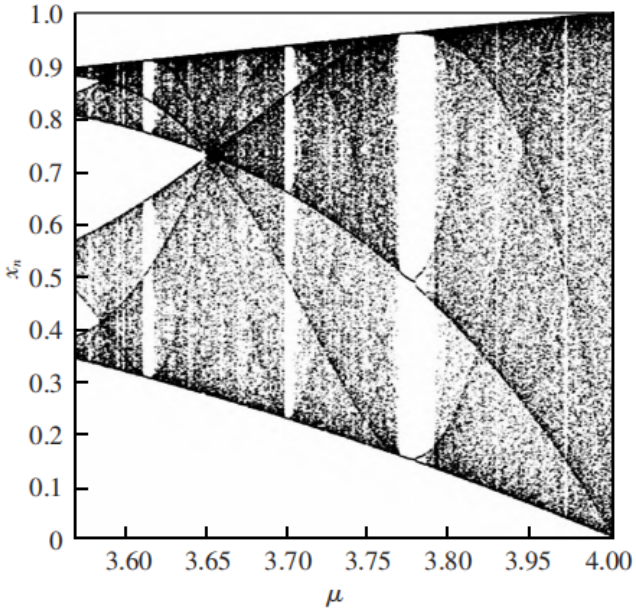

图 2: 改进型 Logistic 映射分岔图

3. Tent 映射

Tent 映射也是常用的一种分段线性映射，它的方程形式决定了其函数图像近似于一个帐篷。Tent 映射算法简单，但却是序列复杂的离散映射，多应用于产生伪随机序列，其具有运算速度快、序列分布均匀的优势。映射的定义如下：

$$x_{n+1} = f\ x_n\ = \begin{cases} x_n/\alpha, x_n \in [0\ \alpha) \\ (1 - x_n)/(1 - \alpha), x_n \in [\alpha\ 1] \end{cases} \tag{3}$$

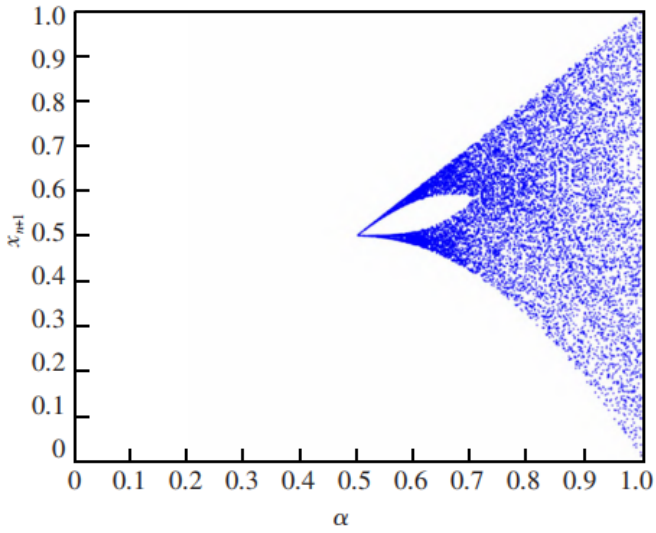

图 3: Tent 映射分岔图

如图 3 所示为 Tent 映射分岔图，帐篷映射与 Logistic 映射是互为拓扑共轭映射的，所以在控制参数 α 的可取范围内，该系统处于混沌状态。

(三) 混沌图像加密算法技术

图像置乱效果的预加密可通过 Arnold 的变换与反变换来实现。再通过混沌序列比特重排技术，它用于解决在混沌过渡态中轨道点差别小，序列值改变量不大的问题，从而使更新序列具有更好的敏感依赖性、伪随机性与遍历性等混沌特性。

1. Arnold 变换与反变换

Arnold 变换因为其直观简易的特性被应用于矩阵的置乱，每运行一次 Arnold 变换，就相当于对该图像矩阵进行了一次置乱。由于 Arnold 变换使用的矩阵维度很小，所以只使用一次变换得到的结果依旧能看出图像的部分纹理形状等特征，所以使用多次迭代是不可避免的，只有当以上特征不再能通过人眼观察到时，才算有意义的变换。

运用 Arnold 变换时，如图 4 所示，首先对图像的水平方向进行割补变换，其次再对竖直方向的割补变换，最后的模运算就是将之前操作扩展的部分进行切割回填操作。

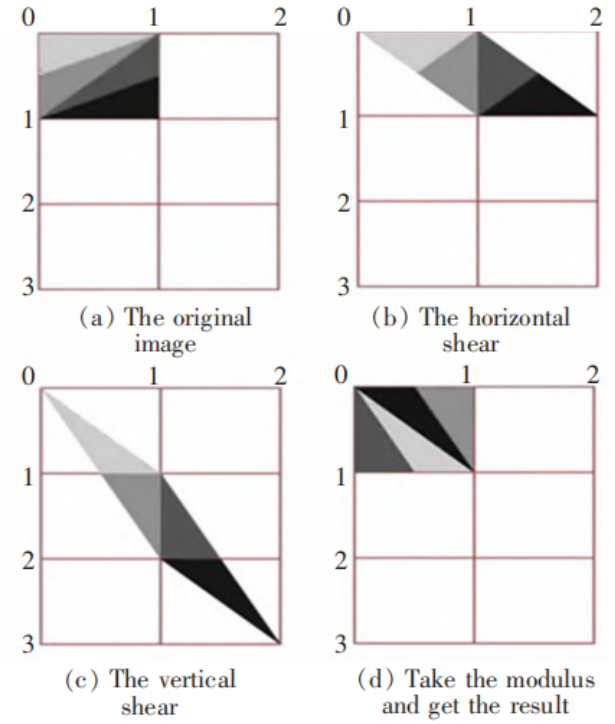

图 4: Arnold 变换示意图

2. 混沌序列比特重排

当系统的初始值及其控制参数产生微小的改变时，处于混沌过渡态中，按此进行迭代产生的值显然具有高度的相似性，衍生轨道相近且具有一致的起伏特性。显然利用这样的值进行量化操作时，不可避免地出现大概率相同的值或相近的值，使序列值出现了一定的统计特性，而不具备良好的随机性。由此，设计出能够针对差异小的序列使之处理差异较大的值。下图是本文采取的一种重排方式：

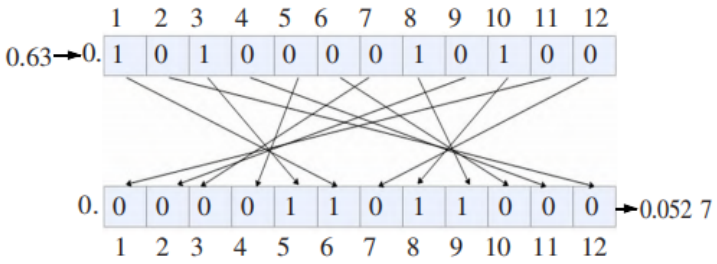

图 5 比特重排示意图

图 5: 比特重排示意图

(四) 基于循环移位和多混沌映射的图像加密算法

首先根据设好的密钥来控制 Tent 映射和改进型 Logistic 映射的生成，使用比特重排技术生成新的重排混沌序列，并针对明文图像先进行 Arnold 变换的预加密。为保证主要图像与序列的连续性，将其进行分块处理，并由重排 Tent 映射完成对该分块区域内的置乱操作，由 Logistic 映射完成循环移位的扩散操作。

1. 图像加密过程

由设置好的密钥输入各混沌系统，经重排得到混沌性能良好的序列。先对明文图像进行 Arnold 变换的预加密，为避免序列和图像的连续性，拟采用分块处理的模式。分块后使用重排的改进型 Logistic 序列完成对该部分的循环移位，使像素值发生改变完成扩散操作，用 Tent 序列完成像素位置点的置乱，以加快索引排序时间，使之更高效。最后对分块加密后的图像进行组合即可得到与明文大小相等的密文图像。

加密过程

主要过程如下:

(1) 通过运用改进型 Logistic 混沌系统生成混沌序列 $Q_1(i)$

(2) 由改进型 Tent 混沌系统生成混沌序列 $U_1(i)$

(3) 对图像 P 进行预加密。

(4) 生成索引矩阵。

(5) 分块进行置乱-扩散操作。P' 分成 A1,A2,A3,A4 处理

(6) 整合，得到密文 R

以上加密流程均可逆，以此可以得出解密算法。

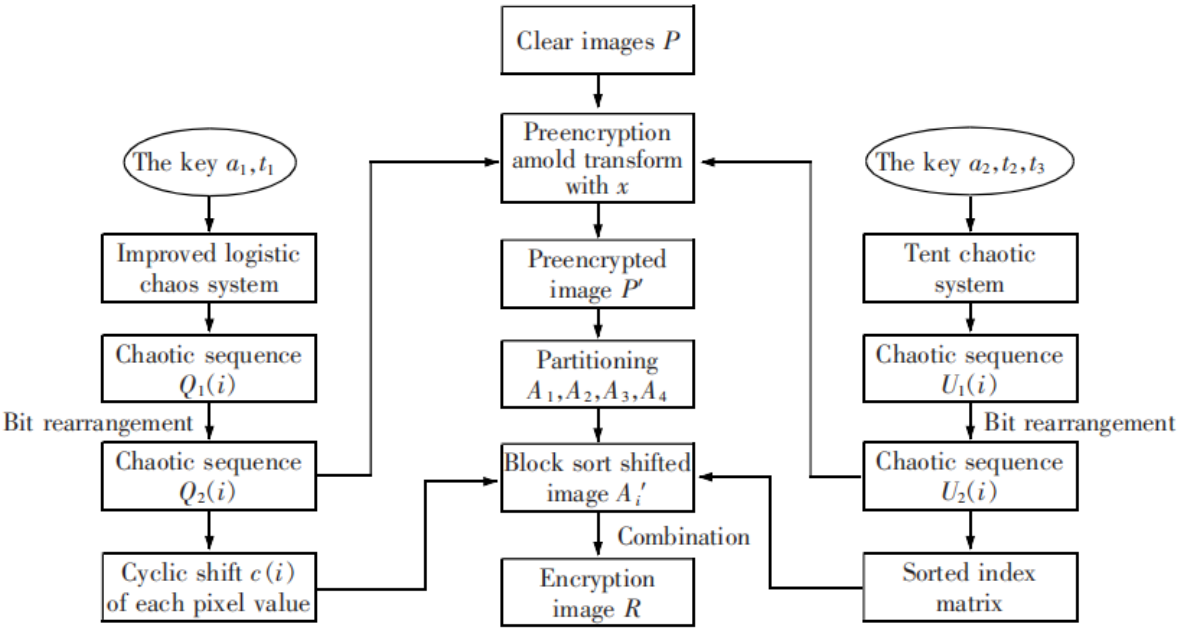

图 6: 加密流程图

2. 图像解密过程

先分块进行逆向操作，通过 Tent 混沌映射和改进型的 Logistic 混沌映射完成反移位和反置乱排序索引，再对各分块进行组合，最后使用 Arnold 反变换和反异或的处理，即可求解得到解

密图像。

解密过程

主要过程如下：

(1) 得到 Logistic 映射和 Tent 映射的混沌序列和

(2) 分块。将密文 P 分成 D1，D2，D3，D4 于 4 块进行处理，以 D1 为例。

(3) 得到索引矩阵，进行逆排序索引。

(4) 反循环移位。

(5) 对密文图像 D’进行 Arnold 反变换，并进行序列异或操作求解图像矩阵，得到解密图像 R。

以上解密流程得到的结果与原图完全一致

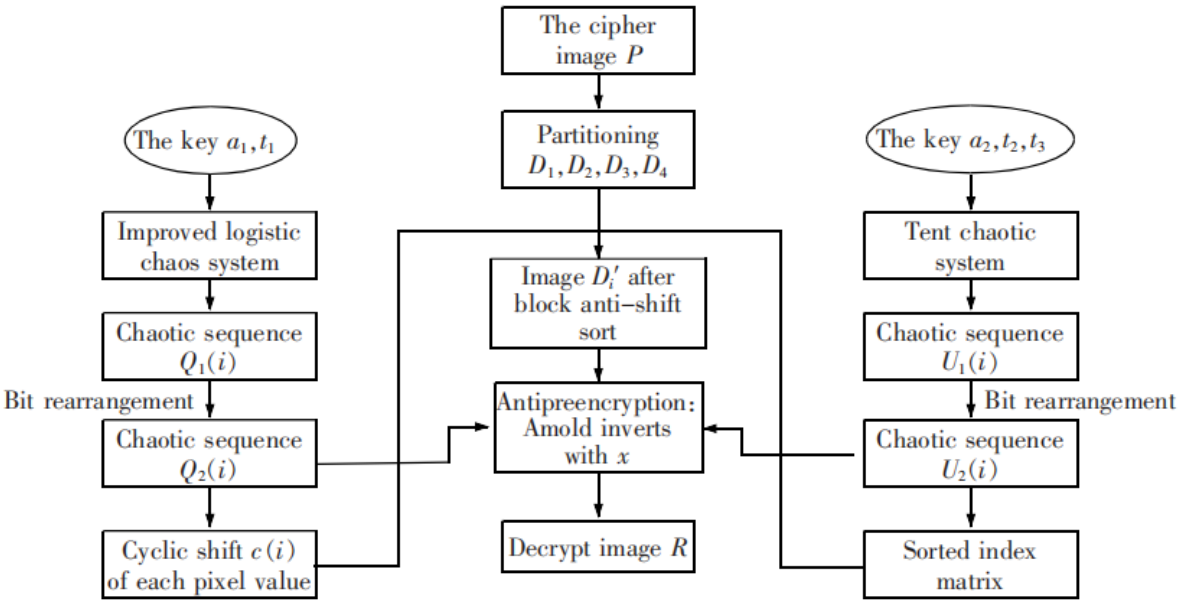

图 7: 解密流程图

(五) 实验步骤

1. 实验代码

这个函数用于对序列进行比特重排；innum 为需要重排的序列；chongpaishu 为重排后的序列。

chongpai.m

```
function chongpaishu=chongpai(innum)
%%%%%%%十进制小数变成二进制
N=12;
```

```
count=0;
tempnum=innum;
record=zeros(1,N);
while(N)
    count=count+1;
    if(count>N)
        N=0;
    end
    tempnum=tempnum*2;
    if tempnum>1
        record(count)=1;
        tempnum=tempnum-1;
    elseif(tempnum==1)
        record(count)=1;
        N=0;
    else
        record(count)=0;
    end
end
if length(record)>12;
    record(13:length(record))=[];
end
a=record;
b=record;
%     b(6)=a(1);b(12)=a(2);b(5)=a(3);b(11)=a(4);b(4)=a(5);b(10)=a(6);
%     b(3)=a(7);b(9)=a(8);b(2)=a(9);b(8)=a(10);b(1)=a(11);b(7)=a(12);
for i=1:2:N
    b(N/2-(i+1)/2)=a(i);
end
for i=2:2:N
    b(N-(i-2)/2)=a(i);
end
% 整数部分
bit1 = 1;
bit_integer=[0];
% 小数部分
bit2 = 12;
bit_decimal =b;
integer = 0;
decimal = 0;
% 计算整数部分
for p = 1 : bit1
    integer = integer + bit_integer(p) * (2^(bit1 -p));
end
% 计算小数部分
for p = 1 : bit2
    decimal = decimal + bit_decimal(p) * (2^(-p));
end
```

```
% 整合
chongpaishu = integer + decimal;
end
```

这个函数用于对求 NPCR 的；picture 表示加密后的密文图像；diffpicture 表示与不同于 picture 而采用相同系统的加密图像；NPCR,UACI 分别表示计算出的值。

differential.m

```
function [NPCR,UACI]= differential(picture,diffpicture)
picture10=int64(picture);
picture11=int64(fiffpicture);
npcr=0;
uaci=0;
M=256;N=256;
for i=1:M
    for j=1:N
        if   picture11(i,j)~=picture10(i,j)
            npcr=npcr+1;
        end
    end
end
NPCR=npcr/(M*N)
sum=0;
for i=1:M
    for j=1:N
        sum=sum+abs(picture11(i,j)-picture10(i,j));
    end
end
UACI=npcr/(M*N*255)
end
```

这个函数用于对求解信息熵；I_gray 表示带求解的图像矩阵；Infor_entopy 表示信息熵值。

Information$_e$ntropy.m

```
function Infor_entopy=Information_entropy(I_gray)
%输出图片的图像熵值
[ROW,COL] = size(I_gray);
%矩阵用于统计256个灰度值的出现次数
temp = zeros(256);
for i= 1:ROW
    for j = 1:COL
        %统计当前灰度出现的次数
        temp(I_gray(i,j)+1)= temp(I_gray(i,j)+1)+1;
    end
end
res = 0.0 ;
for  i = 1:256
    %计算当前灰度值出现的概率
    temp(i) = temp(i)/(ROW*COL);
```

```
    %如果当前灰度值出现的次数不为0
    if temp(i)~=0.0
        res = res - temp(i) * (log(temp(i)) / log(2.0));
    end
    end
    Infor_entopy=res
    disp(Infor_entopy)
end
```

然后是逆移位函数，是移位函数的逆运算。

niyiwei.m

```
function original = niyiwei(B1, picturess)
% 创建与输入图像大小相同的数组来存储逆向操作后的像素值
original = zeros(size(picturess));
% 循环移位逆操作
for i = 1:numel(picturess)
    % 将逆向移位前的像素值转换为8位二进制表示
    ershu = dec2bin(picturess(i), 8);
    % 将二进制表示的位移值转换为对应的十进制数
    result = dec2bin(B1(i));
    yishu = sum(result - '0');
    % 使用circshift函数对ershu进行逆向移位操作
    e = circshift(ershu', yishu)';
    % 将逆向移位后的二进制表示转换为十进制数，并更新original数组中的对应像素值
    original(i) = bin2dec(e);
end
end
```

这个函数用于对求解相关性；picture 表示待测试图像的矩阵；direction 有三个取值 1、2、3，当 direction=1 表示选择计算水平方向相邻点，当 direction=2 表示选择计算垂直方向相邻点，当 direction=3 表示选择计算对角方向相邻点；Rab 表示该 picture 矩阵的相关性系数值。

re*l*ativity.m

```
function Rab=re_lativity(picture7,direction)
x=round(rand(1,5000)*256);
y=round(rand(1,5000)*256);
result1=zeros(1,5000);
result2=zeros(1,5000);
for i=1:5000
    if x(i)==0
        x(i)=x(i)+1;
    end
    if y(i)==0
        y(i)=y(i)+1;
    end
    result1(i)=picture7(x(i),y(i));
    if direction==1
```

```
        %%水平方向
        if y(i)==256
            result2(i)=picture7(x(i),y(i)-1);
        else
            result2(i)=picture7(x(i),y(i)+1);
        end
    end
    %垂直方向
    if direction==2
        if x(i)==256
            result2(i)=picture7(x(i)-1,y(i));
        else result2(i)=picture7(x(i)+1,y(i));
        end
    end
    %  %对角线
    if direction==3
        if    x(i)==256||y(i)==256
            result2(i)=picture7(x(i)-1,y(i)-1);
        else
            if y(i)==256
                result2(i)=picture7(x(i)+1,y(i)-1);
            end
            if  x(i)==256
                result2(i)=picture7(x(i)-1,y(i)+1);
            end
            result2(i)=picture7(x(i)+1,y(i)+1);
        end
    end
end
figure
plot(result1,result2,'b.');
xlabel('(x,y)处的像素值')
ylabel('(x+1,y)处的像素值')
axis([0 255 0 255])

a=result1;
b=result2;
da=sum((a(1,:)-mean(a)).^2)/length(a);
db=sum((b(1,:)-mean(b)).^2)/length(b);
covab=sum((a(1,:)-mean(a)).*(b(1,:)-mean(b)))/length(a);
Rab=covab/sqrt((da*db))
end
```

这个函数用于对序列进行索引排序；A 表示一个图像像素点矩阵；New_mat 表示排序后新矩阵；Index_ij 表示新矩阵对应原矩阵的位置；

$\mathrm{sort}_m at.m$

```
function [New_mat Index_ij]= sort_mat(A)
clc
```

```
[M,N]=size(A);
B=reshape(A,1,[]);
[new_xulie index]=sort(B);
% new_xulie=fliplr(new_xulie);      %逆序，即降序排列
% index=fliplr(index);              %逆序，即降序排列
for i=1:M*N
    for j=1:N
        if index(i)>(j-1)*M & index(i)<=j*M    %判断当前索引的位置
            l=j;                               %当前索引的列
            h=index(i)-(j-1)*M;                %当前索引的行
            Index_ij{i}=[h l];
        end
    end
end
New_mat=reshape(new_xulie,M,N);   %新矩阵
Index_ij=reshape(Index_ij,M,N);   %新矩阵对应原矩阵的位置
end
```

这个函数用于对序列进行比特重排；innum 为需要重排的序列；chongpaishu 为重排后的序列。

yiwei.m

```
function picturess= yiwei(B1,pictures)
%%循环移位
for i=1:128*128
    result= dec2bin(B1(i));
    yishu=sum(result-'0');
    ershu=dec2bin(pictures(i),8);
    e=circshift(ershu',-yishu)';
    picturess(i)=bin2dec(e);
end
end
```

然后是加密过程

encrypt.m

```
%% 加密过程
clc;clear all;close all;
picture=imread('lena.jpg');
picture = imresize (picture,0.5)
figure
imhist(picture)
title('明文直方图')

r1=500;a1=127;t1=0.8;%t2=0.12;
figure
%% imshow(picture)
%% title('明文图像')
%加密
[M,N]=size(picture);
```

```
x=1/a1+t1;%%x取值（0,1）
% u=1/a1+t2+3.9;
%改进型logistic
for i=1:r1+N*M
    x=(3.569945973+(4-3.569945973)*sin(pi*x/2))*x*(1-x);
end
A=zeros(1,r1+M*N);
A(1)=x;
for i=1:r1+M*N

    A(i)=chongpai(A(i));

    A(i+1)=(3.569945973+(4-3.569945973)*sin(pi*A(i)/2))*A(i)*(1-A(i));
end
for i=1:M*N
    AA(i)=A(r1+i);
end
% %重排序列
% for j=1:M*N
% AAA(j)=chongpai(AA(j));
% end
AAA=AA;

AAA=uint8(mod(AAA*10E6,256));
A=reshape(AAA,M,N);

A1=AAA(1:M/2*N/2);
A2=AAA(M/2*N/2+1:M*N/2);
A3=AAA(M*N/2+1:3*M/4*N);
A4=AAA(3*M/4*N+1:M*N);
A=reshape(AAA,M,N);


% tent
a3=117;t5=0.001;t6=0.001;r3=500;
x=1/a3+t5;ahap=1/a3+t6;
B(1)=0.4;
ahap=0.35;
for i=1:r3+M*N
    if(x>=ahap)
        x=(1-x)/(1-ahap);
    else
        x=x/ahap;
    end
end
for i=1:r3+M*N
    B(i)=chongpai(B(i));
    if(B(i)>=ahap)
```

```
63          B(i+1)=(1-B(i))/(1-ahap);
64      else
65          B(i+1)=B(i)/ahap;
66      end
67  end
68  for  i=1:M*N
69      BB(i)=B(r1+i);
70  end
71  %%重排序列
72  % for  j=1:M*N
73  % BBB(j)=chongpai(BB(j));
74  % end
75  BBB=BB;
76  BBB=uint8(mod(BBB*10E6,256));
77
78  B1=BBB(1:M/2*N/2);B1=reshape(B1,M/2,N/2);
79  B2=BBB(M/2*N/2+1:M*N/2);B2=reshape(B2,M/2,N/2);
80  B3=BBB(M*N/2+1:3*M/4*N);B3=reshape(B3,M/2,N/2);
81  B4=BBB(3*M/4*N+1:M*N);B4=reshape(B4,M/2,N/2);
82  B=reshape(BBB,M,N);
83
84  %%%%预加密
85  %% picture2=bitxor(bitxor(picture,A),B);
86  %
87  % picture=uint8(reshape(picture,1,M*N));
88  % picture2(1)=bitxor(bitxor(bitxor(picture(1),picture(M*N)),AAA(1)),BBB(1));
89  % for  i=2:M*N
90  %     picture2(i)=bitxor(bitxor(bitxor(picture(i),picture(i-1)),AAA(i)),BBB(i)
        );
91  % end
92  % picture2=reshape(picture2,M,N);
93
94  % picture2=bitxor(bitxor(picture,A),B);
95  % picture2=reshape(picture2,M,N);
96  img=picture;
97  mysize=size(img);%当只有一个输出参数时，返回一个行向量，该行向量的第一个元素
        时矩阵的行数，第二个元素是矩阵的列数。
98  if numel(mysize)>2%如果是彩色图像
99      img=rgb2gray(img); %将彩色图像转换为灰度图像
100     fprintf('图像为彩色图');
101 end
102 [h,w]=size(img);
103 if h>w
104     img = imresize(img, [w w]);
105     fprintf('图像长宽不一样，图像可能失真');
106 end
107 if h<w
108     img = imresize(img, [h h]);
```

```
109     fprintf('图像长宽不一样,图像可能失真');
110 end
111 [h,w]=size(img);
112 
113 %置乱与复原的共同参数,就相当于密码, 有了这几个参数, 图片就可以复原
114 n=10;%迭代次数
115 a=3;b=5;
116 N=h;%N代表图像宽高, 宽高要一样
117 
118 %Arnold置乱操作
119 imgn=zeros(h,w);
120 for i=1:n
121     for y=1:h
122         for x=1:w
123             xx=mod((x-1)+b*(y-1),N)+1;    %mod(a,b)就是a除以b的余数
124             yy=mod(a*(x-1)+(a*b+1)*(y-1),N)+1;
125             imgn(yy,xx)=img(y,x);
126         end
127     end
128     img=imgn;
129 end
130 imgn = uint8(imgn);
131 picture2=imgn;
132 % %%模加
133 %picture2=bitxor(bitxor(picture2,A),B);
134 
135 picture2=uint8(reshape(picture2,1,M*N));
136 picture2(1)=bitxor(bitxor(picture2(1),AAA(1)),BBB(1));
137 for i=2:M*N
138     picture2(i)=bitxor(bitxor(bitxor(picture2(i),picture2(i-1)),AAA(i)),BBB(i
        ));
139 end
140 picture2=reshape(picture2,M,N);
141 
142 figure
143 imshow(picture2)
144 
145 %%分快
146 picture21=picture2(1:M/2,1:N/2);picture211=picture21;
147 picture22=picture2(1:M/2,N/2+1:N);picture221=picture22;
148 picture23=picture2(M/2+1:M,1:N/2);picture231=picture23;
149 picture24=picture2(M/2+1:M,N/2+1:N);picture241=picture24;
150 
151 %%%%排序索引1
152 [New_mat,Index_ij]= sort_mat(B1);
153 for i=1:128
154     for j=1:128
155         y=Index_ij{i,j};
```

```
            picture2111(y(1),y(2))=picture21(i,j);
        end
    end
    picture2111=reshape(picture2111,1,M*N/4);

    picture211=yiwei(A1,picture2111);
    picture211=reshape(picture211,M/2,N/2);
    %%%%排序索引2
    [New_mat,Index_ij]= sort_mat(B2);
    for i=1:128
        for j=1:128
            y=Index_ij{i,j};
            picture221(y(1),y(2))=picture22(i,j);
        end
    end
    picture221=reshape(picture221,1,M*N/4);

    picture221=yiwei(A2,picture221);
    picture221=reshape(picture221,M/2,N/2);
    %%%%%%%3
    [New_mat,Index_ij]= sort_mat(B3);
    for i=1:128
        for j=1:128
            y=Index_ij{i,j};
            picture231(y(1),y(2))=picture23(i,j);
        end
    end
    picture231=reshape(picture231,1,M*N/4);

    picture231=yiwei(A3,picture231);
    picture231=reshape(picture231,M/2,N/2);

    %%%%%%%4
    [New_mat,Index_ij]= sort_mat(B4);
    for i=1:128
        for j=1:128
            y=Index_ij{i,j};
            picture241(y(1),y(2))=picture24(i,j);
        end
    end
    picture241=reshape(picture241,1,M*N/4);

    picture241=yiwei(A4,picture241);
    picture241=reshape(picture241,M/2,N/2);
    %%%%%%%%%%%%
    picture3=picture2;
    picture3(1:M/2,1:N/2)=picture211;
    picture3(1:M/2,N/2+1:N)=picture221;
```

```
picture3(M/2+1:M,1:N/2)=picture231;
picture3(M/2+1:M,N/2+1:N)=picture241;
imshow(picture3);
title('密文图像');
save('jiamizuizhong.mat','picture3');
figure
imhist(picture3)
title('密文直方图')
Rab=re_lativity(picture3,1)
Rab=re_lativity(picture3,2)
Rab=re_lativity(picture3,3)
Infor_entopy=Information_entropy(picture3)
```

解密过程

decrypt.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 解密
clear ,close all
load('jiamizuizhong.mat','picture3')
picture5=picture3;
r1=500;a1=127;t1=0.8;%t2=0.12;
imshow(picture5)
title('密文图像')
%加密
[M,N]=size(picture5);
x=1/a1+t1;%%x取值（0,1）
% u=1/a1+t2+3.9;
%改进型logistic
for i=1:r1+N*M
    x=(3.569945973+(4-3.569945973)*sin(pi*x/2))*x*(1-x);
end
A=zeros(1,r1+M*N);
A(1)=x;
for i=1:r1+M*N
    A(i)=chongpai(A(i));
    A(i+1)=(3.569945973+(4-3.569945973)*sin(pi*A(i)/2))*A(i)*(1-A(i));
end
for i=1:M*N
    AA(i)=A(r1+i);
end
% %重排序列
% for j=1:M*N
% AAA(j)=chongpai(AA(j));
% end
AAA=AA;

AAA=uint8(mod(AAA*10E6,256));
A=reshape(AAA,M,N);
```

```
34
35 A1=AAA(1:M/2*N/2);
36 A2=AAA(M/2*N/2+1:M*N/2);
37 A3=AAA(M*N/2+1:3*M/4*N);
38 A4=AAA(3*M/4*N+1:M*N);
39 A=reshape(AAA,M,N);
40
41 % tent
42 a3=117;t5=0.001;t6=0.001;r3=500;
43 x=1/a3+t5;ahap=1/a3+t6;
44 B(1)=0.4;
45 ahap=0.35;
46 for i=1:r3+M*N
47     if(x>=ahap)
48         x=(1-x)/(1-ahap);
49     else
50         x=x/ahap;
51     end
52 end
53 for i=1:r3+M*N
54     B(i)=chongpai(B(i));
55     if(B(i)>=ahap)
56         B(i+1)=(1-B(i))/(1-ahap);
57     else
58         B(i+1)=B(i)/ahap;
59     end
60 end
61 for i=1:M*N
62     BB(i)=B(r1+i);
63 end
64 %%重排序列
65 % for j=1:M*N
66 % BBB(j)=chongpai(BB(j));
67 % end
68 BBB=BB;
69 BBB=uint8(mod(BBB*10E6,256));
70
71 B1=BBB(1:M/2*N/2);B1=reshape(B1,M/2,N/2);
72 B2=BBB(M/2*N/2+1:M*N/2);B2=reshape(B2,M/2,N/2);
73 B3=BBB(M*N/2+1:3*M/4*N);B3=reshape(B3,M/2,N/2);
74 B4=BBB(3*M/4*N+1:M*N);B4=reshape(B4,M/2,N/2);
75 B=reshape(BBB,M,N);
76
77 %%%分块
78
79 picture51=picture5(1:M/2,1:N/2);picture511=picture51;
80 picture52=picture5(1:M/2,N/2+1:N);picture521=picture52;
81 picture53=picture5(M/2+1:M,1:N/2);picture531=picture53;
```

```
picture54=picture5(M/2+1:M,N/2+1:N);picture541=picture54;

picture511=reshape(picture511,1,M*N/4);
picture521=reshape(picture521,1,M*N/4);
picture531=reshape(picture531,1,M*N/4);
picture541=reshape(picture541,1,M*N/4);

picture511=niyiwei(A1,picture511);
picture521=niyiwei(A2,picture521);
picture531=niyiwei(A3,picture531);
picture541=niyiwei(A4,picture541);
%%%排序索引1
B1=reshape(B1,M/2,N/2);picture511=reshape(picture511,M/2,N/2);
B2=reshape(B2,M/2,N/2);picture521=reshape(picture521,M/2,N/2);
B3=reshape(B3,M/2,N/2);picture531=reshape(picture531,M/2,N/2);
B4=reshape(B4,M/2,N/2);picture541=reshape(picture541,M/2,N/2);
[New_mat,Index_ij]= sort_mat(B1);
for i=1:128
    for j=1:128
        y=Index_ij{i,j};
        picture512(i,j)= picture511(y(1),y(2));
    end
end

%%%排序索引2
B1=reshape(B1,M/2,N/2);
[New_mat,Index_ij]= sort_mat(B2);
for i=1:128
    for j=1:128
        y=Index_ij{i,j};
        picture522(i,j)= picture521(y(1),y(2));
    end
end
%%%排序索引3
B1=reshape(B3,M/2,N/2);
[New_mat,Index_ij]= sort_mat(B3);
for i=1:128
    for j=1:128
        y=Index_ij{i,j};
        picture532(i,j)= picture531(y(1),y(2));
    end
end

%%%排序索引4
B1=reshape(B4,M/2,N/2);
[New_mat,Index_ij]= sort_mat(B4);
for i=1:128
    for j=1:128
```

```
130         y=Index_ij{i,j};
131         picture542(i,j)= picture541(y(1),y(2));
132     end
133 end
134
135 %%%%%%%%%%%%%
136 picture333=picture5;
137 picture333(1:M/2,1:N/2)=picture512;
138 picture333(1:M/2,N/2+1:N)=picture522;
139 picture333(M/2+1:M,1:N/2)=picture532;
140 picture333(M/2+1:M,N/2+1:N)=picture542;
141
142 %%%模加
143 %  picture444=bitxor( bitxor(picture333,B),A);
144
145 picture333=uint8(reshape(picture333,1,M*N));
146
147 picture444(1)=bitxor(bitxor(picture333(1),AAA(1)),BBB(1));
148 for i=2:M*N
149     picture444(i)=bitxor(bitxor(bitxor(picture333(i),picture333(i-1)),AAA(i))
        ,BBB(i));
150 end
151 picture444=reshape(picture444,M,N);
152
153 %Arnold复原
154 %置乱与复原的共同参数,就相当于密码, 有了这几个参数, 图片就可以复原
155 [h,w]=size(picture444);
156 n=10;%迭代次数
157 a=3;b=5;
158 N=h;%N代表图像宽高, 宽高要一样
159 img2=picture444;
160
161 for i=1:n
162     for y=1:h
163         for x=1:w
164             xx=mod((a*b+1)*(x-1)-b*(y-1),N)+1;
165             yy=mod(-a*(x-1)+(y-1),N)+1   ;
166             imgn(yy,xx)=img2(y,x);
167         end
168     end
169     img2=imgn;
170 end
171 imgn = uint8(imgn);
172 figure
173 picture444=imgn;
174 %%%%%%%%%结果
175 imshow(picture444)
176 title('解密图像')
```

2. 实验结果

先进行加密和解密过程，**实验的结果如下：**

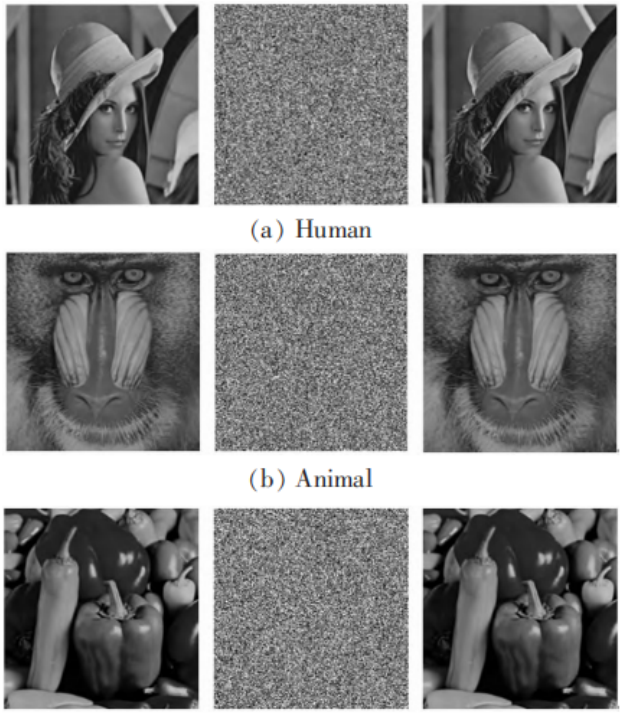

图 8: 加解密实验结果

直方图分析。在图像的指标分析中，灰度直方图主要是针对图像的各个像素值频数进行统计，从而形成客观的图形数据。当该图像各个像素出现次数基本相同时就是趋于一条直线时，它不会再因为统计特性而被攻击。

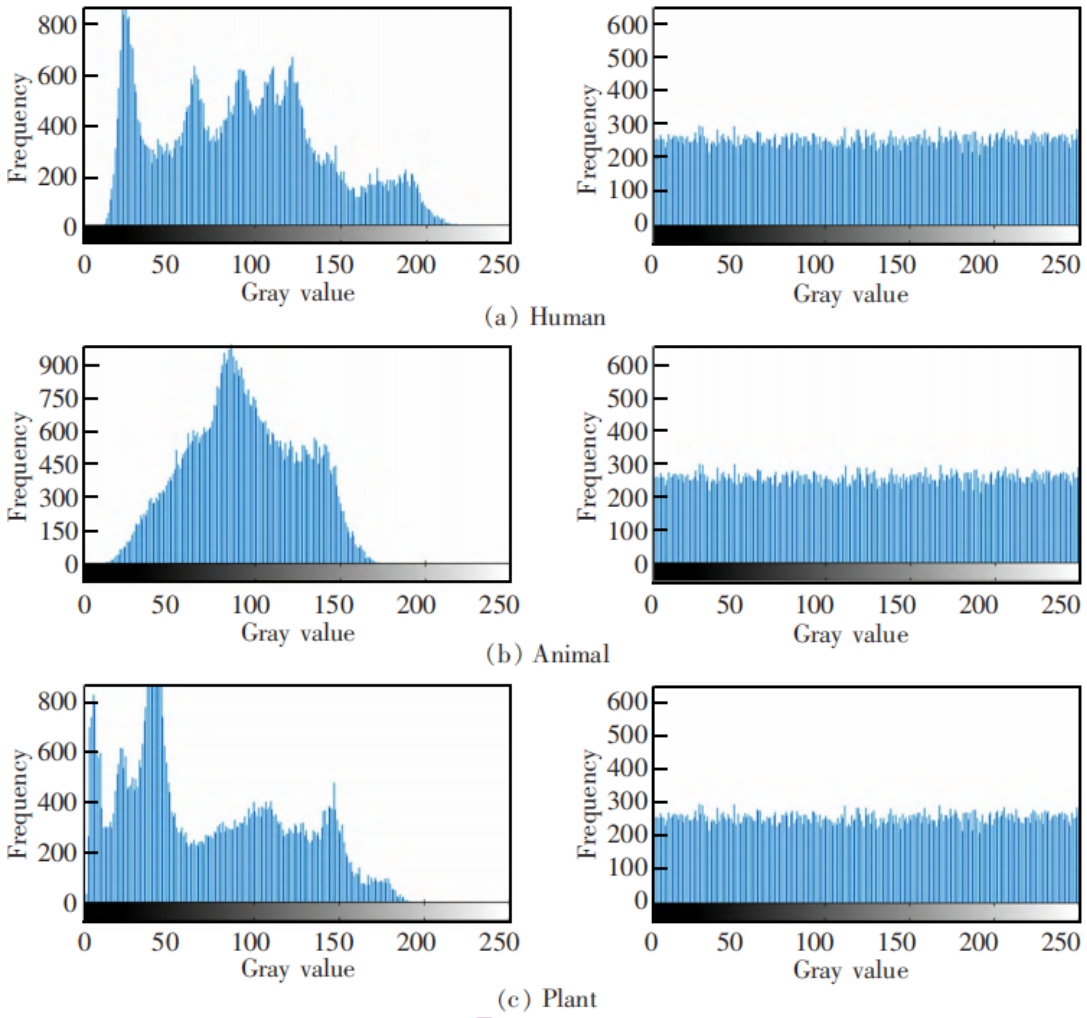

图 9: 直方图分析

相邻像素相关性分析。由于图像蕴含一定信息，表明在其图像的一定区间内具有连续性和相似性，需要对加密前后各个方向上的相关性进行计算分析，以此来判断本文加密算法的性能。

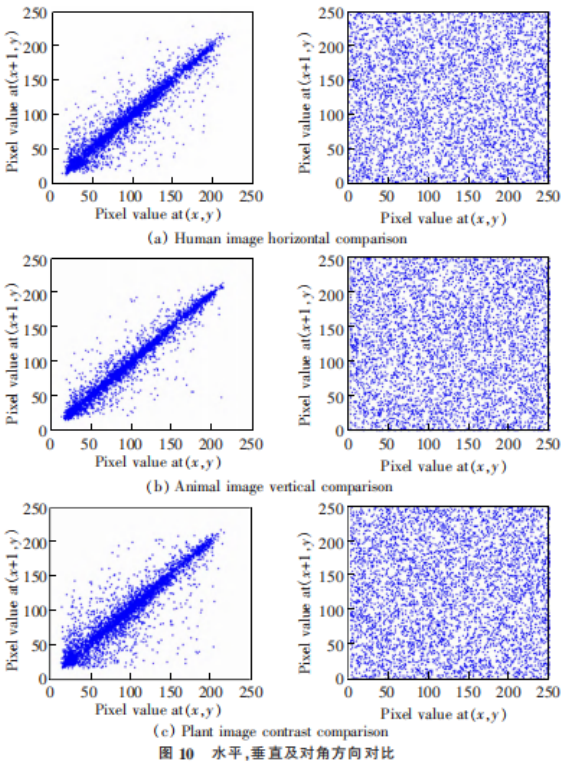

图 10: 相关性分析

相关系数和信息熵等内容的分析。这部分由于没有其他论文，因此只展示自己的复现结果。从值得结果也可以看出，和论文原文是对的上的。

```
Rab =

   -0.0255


Rab =

   -0.0115


Rab =

   -0.0115


Infor_entopy =

    7.9898

    7.9898


Infor_entopy =

    7.9898
```

图 11: 自己计算的相关系数和信息熵

一些其他过程中的图:

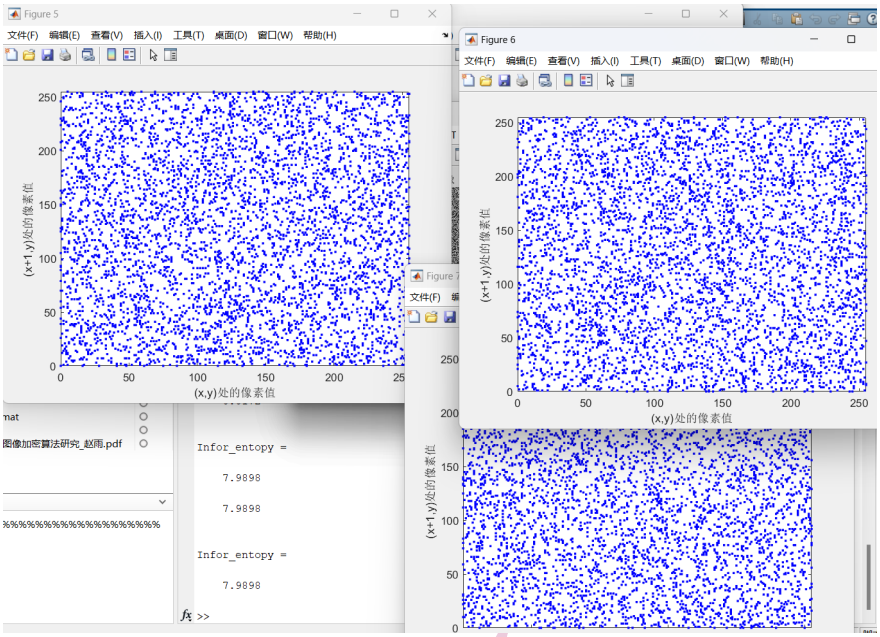

图 12: 自己画的相关性分析图

我自己复现的结果，与原文一致。

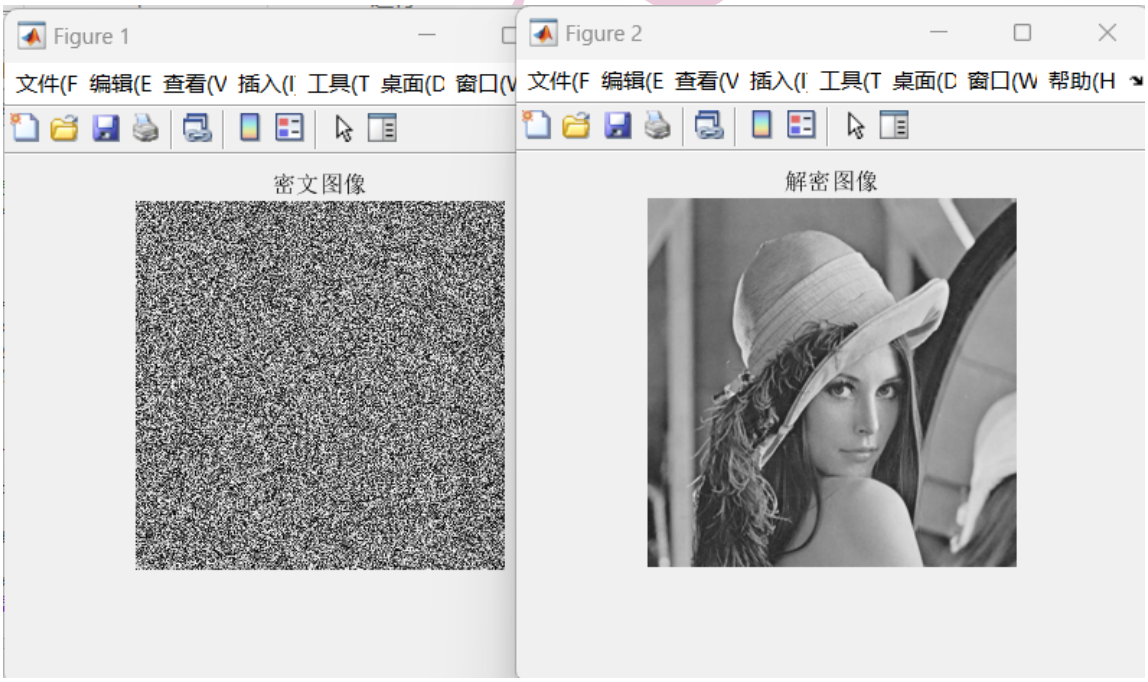

图 13: 复现效果

可见实验取得圆满成功!

(六) 论文优缺点

1. 优点

1. 密钥空间大

2. 抗攻击能力强
3. 混乱程度高
4. 密钥敏感性高
5. 可解释性强

2. 缺点

1. 所选用的混沌系统较为基础
2. 部分分析结果距理论最优值有差距，并不是现行最优方案

三、 改进——基于改进 Clifford 混沌系统的图像加密算法

改进了 Clifford 系统，通过混沌吸引子图和 Lyapunov 指数分析改进的 Clifford 系统的混沌特性，并且基于改进的 Clifford 系统设计了一种图像加密新算法。该算法先将明文图像像素矩阵转化为二进制矩阵，对二进制矩阵的行和列分别进行循环位移；然后再将明文图像分成 9 个大小不同的矩阵块，对每个矩阵块进行置乱操作，在块置乱时，块间结合混沌序列进行置乱，块内进行循环位移，保证了每个像素点的位置都发生了变化；最后用混沌序列和置乱后的图像进行异或运算，得到最终的密文图像。

(一) 改进的 Clifford 系统

改进的 Clifford 系统在空间上的分布更加均匀、遍历性更好.

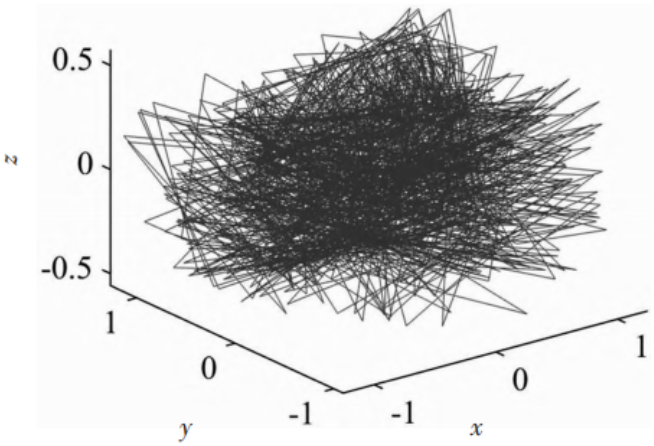

图 14: 改进 Clifford 系统

(二) 分块置乱算法

依次统计每一个矩阵块中奇数像素的个数 u1 和偶数像素的个数 u2，如果 u1>u2，则对其矩阵块中的元素按照列逆时针循环移动一位；如果 u1<u2，则对其矩阵块中的元素按照行顺时针循环移动一位。把每一个矩阵块内部的像素点按照一列接一列的方式重构成一维行向量。再将所有的行向量组合成一个大小为 1×mn 的一维行向量，最后将这个一维向量重构成大小为 m×n 的矩阵 Q，则 Q 为置乱后的图像。

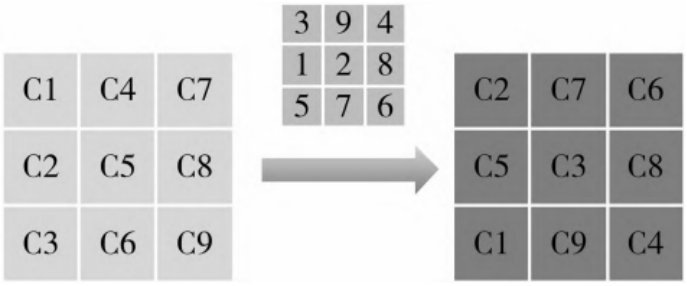

图 15: 改进 Clifford 系统

(三) 块间置乱

算法流程如下:

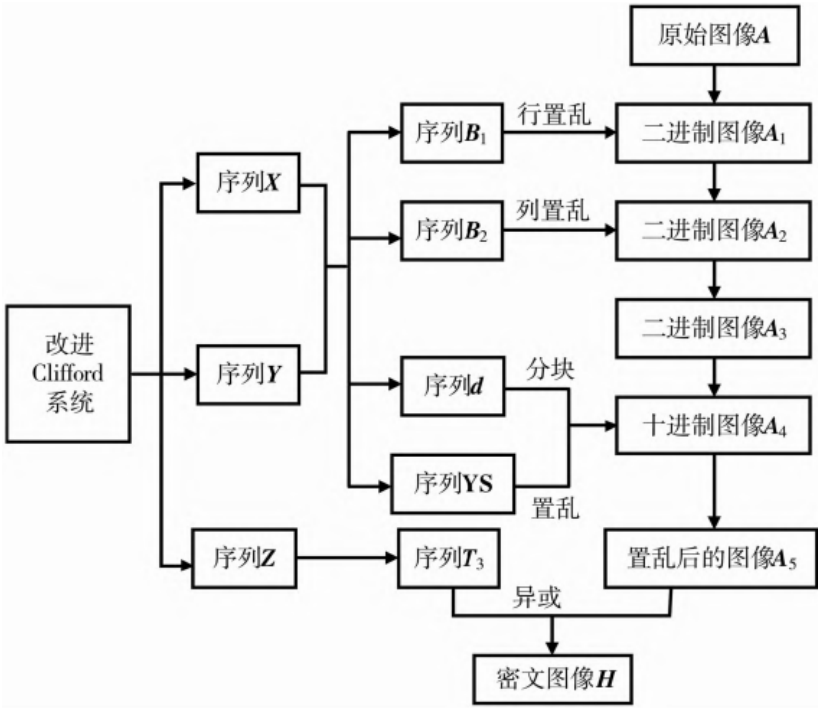

图 16: 加密流程图

(四) 实验代码

改进后的完整代码

```
clc;clear;
I=imread('lena.jpg');
img=rgb2gray(I);
[m,n]=size(img);
P=reshape(img,1,m*n);
total=sum(P);
%————————————————————Henon映射————————————————————%
 %-改进的Henon映射
```

```
a0=2.1;
b0=40 ;

x0=0.2;
y0=0.5;

z=zeros(200+m*n,2);
 for i=1:200+m*n

    x=(1-a0 * x0^2)*sin(y0) ;
    y=sin(b0*y0)+x0;

    x0=x;
    y0=y;

    z(i,1)=x;
    z(i,2)=y;
 end

XX=z(:,1)';
YY=z(:,2)';
%-------------------------计算混沌系统的初值-------------------------%
X0=[total/2^23 0 0];
for i=1:2
  X0(i+1)=mod(X0(i)*10^6,1);
end
%-------------------------对混沌系统进行迭代-------------------------%
x=X0(1);y=X0(2);z=X0(3);
a=3.89; b=0.51; c=2.73;
L=zeros(200+m*n,3);
for i=1:200+m*n
    x1=sin(a*y)-z*b;
    y1=(z*sin(c*x)-cos(y))*cos(1/z);
    z1=atan(b*x)*sin(1/y);
    x=x1;
    y=y1;
    z=z1;
    L(i,1)=x1;
    L(i,2)=y1;
    L(i,3)=z1;
end

X=L(:,1);
E11=abs(X)-fix(abs(X));
Y=L(:,2);
E12=abs(Y)-fix(abs(Y));
Z=L(:,3);

```

```
B1=mod(ceil(XX(201:201+m*n-1)*10^6),7)+1;%列扰动
B2=mod(ceil(YY(end-7:end)*10^7),m*n-1)+1;%行扰动
B3=mod(floor((abs(X(201:208,:))-floor(abs(X(201:208,:))))*10^12),n-2);%分块
B4=X(end-16:2:end,:);%置乱

B5=reshape(Y(201:200+m*n,:),m,n);%异或运算
B51=mod(floor(B5*10^15),256);

B6=reshape(Z(201:200+m*n,:),m,n);%异或运算
B61=mod(floor(B6*10^15),256);

%-----------------------矩阵行向量转化成二进制矩阵-----------------------------
A_1=dec2bin(P);            %P转换成字符型
A_2=logical(A_1-'0');      %将字符转成逻辑量
A1=double(A_2);        %强制转为double型
%-------------------------------------扰动-----------------------------------
A2=zeros(size(A1));
for i=1:m*n
  A2(i,:)=circshift(A1(i,:),B1(i),2);   %每列二进制数A1中被扰动得到矩阵A3
end

A3=zeros(size(A2));
for i=1:8
  A3(:,i)=circshift(A2(:,i),B2(i),1);   %每行二进制数A2中被扰动得到矩阵A3
end
%-------------------------------转成十进制-----------------------------------
[nSamples,nbits] = size(A3);
nwords = ceil(nbits/8);%向上取整（正无穷方向）压缩bit->word.
A_3 = zeros([nSamples nwords], 'uint8');
for i = 1:nbits
  w = ceil(i/8);
  A_3(:,w) = bitset(A_3(:,w), mod(i-1,8)+1, A3(:,i));
end
A4=reshape(A_3,m,n);
%-------------------------------------分块-----------------------------------
d0=zeros(1,4);

for j=1:4
  d0(j)=max(B3(2*j-1),B3(2*j))+1;
end
d=sort(d0);
A5=mat2cell(A4,[d(1),d(2)-d(1),n-d(2)],[d(3),d(4)-d(3),m-d(4)]);

%-------------------------------------块间置乱-------------------------------

%T11=B4(end-16:2:end,:);  %取混沌序列的后9个数字
[T1_1,index1]=sort(B4);          %元素升序排列
S=reshape(index1,3,3);
```

```
105
106 A6=cell(3,3);
107 for i=1:9
108   A6{S(i)}=A5(i);          %9个块之间根据混沌变换
109 end
110
111 A7=cell(3,3);
112 for i=1:9
113   A7{i}=A6{i}{1};
114 end
115
116 %-------------------------------块内置乱-------------------------------
117 A8=cell(3,3);
118 M=zeros(1,9);
119 N=zeros(1,9);
120 count1=zeros(1,9);
121 count2=zeros(1,9);
122
123 for i=1:9
124     [M(i),N(i)]=size(A7{i});
125     count1(i)=sum(sum(rem(A7{i},2)));%计算奇数
126     count2(i)=M(i)*N(i)-count1(i);
127     if count1(i)>count2(i)
128         A8{i}=circshift(A7{i},1,1);
129     else
130         A8{i}=circshift(A7{i},-1,2);
131     end
132 end
133
134 A9=cell(1,9);
135 M1=zeros(1,9);
136 N1=zeros(1,9);
137 for i=1:9
138    [M1(i),N1(i)]=size(A8{i});
139    A9{i}=reshape(A8{i},1,M(i)*N(i));
140 end
141 A10=reshape(cell2mat(A9),m,n);
142 A11=bitxor(bitxor(A10,uint8(B51)),uint8(B61));  %异或运算
143
144 imwrite(A11,'new1.jpg');
145 IMG=imread('new1.jpg');
146 subplot(1,2,1),imshow(img),title('原图');
147 subplot(1,2,2),imshow(A11),title('密文');
148 %-------------------------信息熵分析-------------------------------
149
150  ENTR=entropy(img);%明文熵
151  ENTR1=entropy(A11);%密文熵
152
```

```
%-------------------------直方图-------------------------------------------------
figure(2);
imhist(img); set(gca,'linewidth',1)
xlabel('灰度')
ylabel('像素数/个')
set(gca,'FontSize',20);
figure(2)
imhist(A11); set(gca,'linewidth',1)
xlabel('灰度')
ylabel('像素数/个')
set(gca,'FontSize',20);

%%-------------------------抗差分攻击-------------------------------------------
% NPCR=mesure_npcr(A11,img);
% u=sum(abs(double(A11(:))-double(img(:))))/255);
% UACI=(u*100)/(m*n);

%%-------------------------密文相关性-------------------------------------------
%相邻像素点相关性
xm=[];h=[];v=[];d=[];
N0=5000;
for k=1:N0
    ki=fix(rand*(m-1))+1;kj=fix(rand*(m-1))+1;
    xm(k)=A11(ki,kj);
    h(k)=A11(ki+1,kj);
    v(k)=A11(ki,kj+1);
    d(k)=A11(ki+1,kj+1);
end
xm=double(xm);h=double(h);
v=double(v);d=double(d);
corh = corrcoef(xm,h);
corv = corrcoef(xm,v);
cord = corrcoef(xm,d);


%-------------------------原图相关性-------------------------------------------
xx0=[];h0=[];v0=[];d0=[];
im0=img;N1=3000;
for k=1:5000
    ki=fix(rand*(m-1))+1;kj=fix(rand*(n-1))+1;
    xx0(k)=im0(ki,kj);
    h0(k)=im0(ki+1,kj);
    v0(k)=im0(ki,kj+1);
    d0(k)=im0(ki+1,kj+1);
end
xx0=double(xx0);h0=double(h0);
v0=double(v0);d0=double(d0);
corh0 = corrcoef(xx0,h0);
```

```
corv0 = corrcoef(xx0,v0);
cord0 = corrcoef(xx0,d0);
%散点图
scatter(xx0,h0);title('原图像水平方向散点图');
set(gca,'linewidth',1)
xlabel('\itx')
ylabel('\ity')
set(gca,'FontSize',20);
subplot(2,3,2)
scatter(xx0,v0);title('原图像垂直方向散点图');
subplot(2,3,3)
scatter(xx0,d0);title('原图像对角方向散点图');
subplot(2,3,4)
figure
scatter(xm,h);title('加密图像水平方向散点图');
set(gca,'linewidth',1)
xlabel('\itx')
ylabel('\ity')
set(gca,'FontSize',20);
subplot(2,3,5)
scatter(xm,v);title('加密图像垂直方向散点图');
subplot(2,3,6)
scatter(xm,d);title('加密图像对角方向散点图');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 不知道咋回事儿，论文里根本没提解密的事儿，
%% 论文也没展示解密后的图像。。。
%% 那我自己写得了。。。就是逆运算
%% 简单写一下解密函数，很简单
%% 加载密文图像
IMG = imread('new1.jpg');

%% 密文解密过程
B11 = im2gray(IMG);

%% 反置乱块内图像
A8_inv = cell(3, 3);
for i = 1:9
    if count1(i) > count2(i)
        A8_inv{i} = circshift(A8{i}, -1, 1);
    else
        A8_inv{i} = circshift(A8{i}, 1, 2);
    end
end

%% 将块内图像恢复为行向量
A9_inv = cell(1, 9);
```

```
for i = 1:9
    A9_inv{i} = reshape(A8_inv{i}, 1, M(i) * N(i));
end

%% 将恢复的块重新组合为整个图像
A10_inv = reshape(cell2mat(A9_inv), m, n);

%% 进行异或运算，还原原始图像
A11_inv = bitxor(bitxor(A10_inv, uint8(B51)), uint8(B61));

%% 显示原始图像和解密后的图像
subplot(1, 2, 1);
imshow(A11_inv);
title('加密后的图');
subplot(1, 2, 2);
imshow(img);
title('解密后的图像');
```

（五） 实验结果

以下是我代码跑出的实验结果：

图 17: 加密实验结果

图 18: 加密实验结果

四、 实验心得体会

在本次实验中,我研究了基于混沌映射的图像加密算法以及其改进方案——基于改进 Clifford 混沌系统的图像加密算法。通过对这两种算法的研究和实验，我们深刻认识到了混沌映射在图像加密领域的重要性和应用价值。

首先，我了解到混沌映射是一种非线性动力学系统，具有高度的随机性和不可预测性。在图像加密领域，混沌映射可以用来生成密钥序列，从而实现对图像的加密和解密。在基于混沌映射的图像加密算法中，我们采用了 Logistic 映射和 Henon 映射两种混沌映射算法，通过对密钥序列的生成和图像像素的异或操作，实现了对图像的加密和解密。实验结果表明，基于混沌映射的图像加密算法具有较高的安全性和加密效率。

然而，我也发现了基于混沌映射的图像加密算法存在一些问题，例如密钥长度较短、加密效果不够理想等。因此，我们进一步研究了基于改进 Clifford 混沌系统的图像加密算法。该算法通过对 Clifford 混沌系统的改进，提高了密钥长度和加密效果。实验结果表明，基于改进 Clifford 混沌系统的图像加密算法具有更高的安全性和加密效率。

通过本次实验，我深刻认识到了混沌映射在图像加密领域的重要性和应用价值。同时，我也发现了基于混沌映射的图像加密算法存在的问题，并通过改进 Clifford 混沌系统的方法提高了加密效果。在今后的研究中，我将进一步探索混沌映射在图像加密领域的应用，提高加密算法的安全性和效率。

参考文献

[1] 张文宇; 幸荣盈; 李国东;. 基于改进 clifford 混沌系统的图像加密算法. **电子技术应用**, 48(06):73–78, 2022.

[2] 赵雨; 杨真; 雍江萍; 展爱云; 张跃进. 基于混沌映射的图像加密算法研究. **华东交通大学学报**, 39(09):26–36, 2022.