**《数字内容安全大作业》**

**实验报告**

**实验名称：** 常用图像置乱算法matlab实现  **任课教师： 付宇**

**学号：**200340164  **姓名：**张博文  **提交日期：**2022/11/5

**一、实验目标：**

使用matlab实现常用图像置乱算法进行图像置乱

**二、实验设计：**

常用置乱算法分三种：

1、对二维图像矩阵进行行置乱和列置乱，或交叉进行行、列置乱。

2、将二维图像展开成一维行向量或一维列向量，对该向量进行位置置乱。

3、借助2x2置乱矩阵变换二维图像的各个像素点的位置。

**三、实验记录：**

第一种

1、二维图像直接行置乱与列置乱

（1）随机置乱

行/列随机：借助于混沌系统产生长度为M/N的随机数向量X/Y，然后将图像矩阵的第i行/第j列与第Xi行/第Yj列互换。

实现：

这里使用的超混沌Lorenz系统:

% https://blog.csdn.net/dollar\_jen/article/details/115335268?spm=1001.2101.3001.6650.4&utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-4-115335268-blog-115381954.pc\_relevant\_3mothn\_strategy\_and\_data\_recovery&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-4-115335268-blog-115381954.pc\_relevant\_3mothn\_strategy\_and\_data\_recovery&utm\_relevant\_index=9

clc,clear;

P=imread('1.jpg');P=rgb2gray(P);% 读入图像 转化为灰度图像

iptsetpref('imshowborder','tight');% 设置图像显示无边框

figure(1);imshow(P);% 显示

[M,N]=size(P);P=double(P);% 获得图像行列数M N

n=M+N;% 设定计算的混沌序列长度

h=0.002;t=800;% 以下

a=10;b=8/3;c=28;r=-1;

x0=1.1;y0=2.2;z0=3.3;w0=4.4;

s=zeros(1,n);

for i=1:n+t

K11=a\*(y0-x0)+w0;K12=a\*(y0-(x0+K11\*h/2))+w0;

K13=a\*(y0-(x0+K12\*h/2))+w0;K14=a\*(y0-(x0+h\*K13))+w0;

x1=x0+(K11+K12+K13+K14)\*h/6;

K21=c\*x1-y0-x1\*z0;K22=c\*x1-(y0+K21\*h/2)-x1\*z0;

K23=c\*x1-(y0+K22\*h/2)-x1\*z0;K24=c\*x1-(y0+h\*K23)-x1\*z0;

y1=y0+(K21+K22+K23+K24)\*h/6;

K31=x1\*y1-b\*z0;K32=x1\*y1-b\*(z0+K31\*h/2);

K33=x1\*y1-b\*(z0+K32\*h/2);K34=x1\*y1-b\*(z0+h\*K33);

z1=z0+(K31+K32+K33+K34)\*h/6;

K41=-y1\*z1+r\*w0;K42=-y1\*z1+r\*(w0+K41\*h/2);

K43=-y1\*z1+r\*(w0+K42\*h/2);K44=-y1\*z1+r\*(w0+h\*K43);

w1=w0+(K41+K42+K43+K44)\*h/6;

x0=x1;y0=y1;z0=z1;w0=w1;

if i>t

s(i-t)=x1;

if mod((i-t),3000)==0% 每3000次迭代后加扰动

x0=x0+h\*sin(y0);

end

end

end% 以上 生成混沌序列

X=mod(floor((s(1:M)+100)\*10^10),M)+1;% 生成行置乱向量

Y=mod(floor((s(M+1:M+N)+100)\*10^10),N)+1;% 生成列置乱向量

A=P;

for i=1:M

t=A(i,:);A(i,:)=A(X(i),:);A(X(i),:)=t;

end

figure(2);imshow(uint8(A));% 行置乱后的图像

B=A;

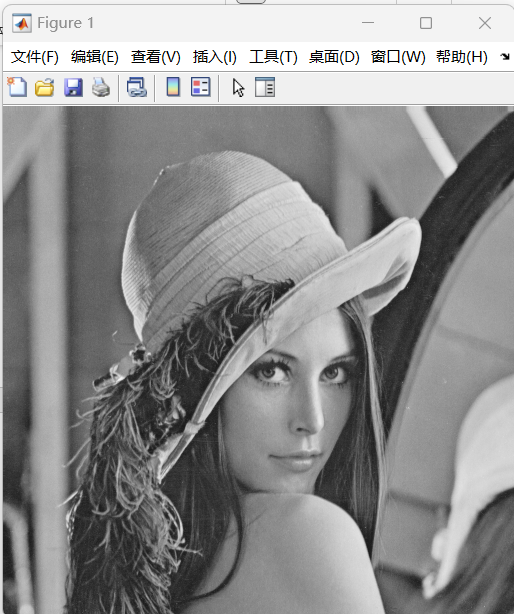
for j=1:N

t=B(:,j);B(:,j)=B(:,Y(j));B(:,Y(j))=t;

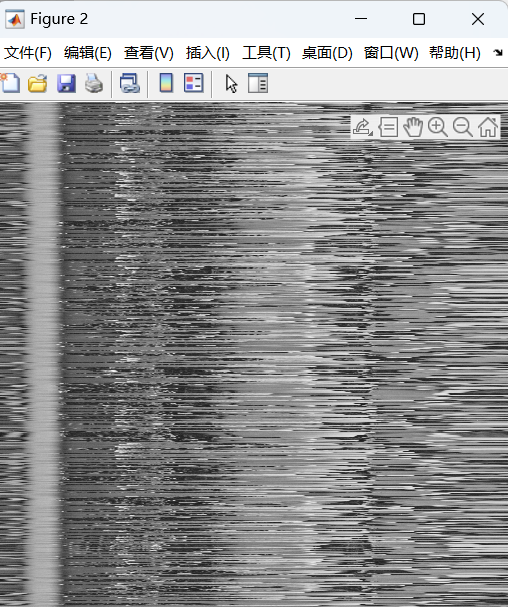
end

figure(3);imshow(uint8(B));% 列置乱后的图像

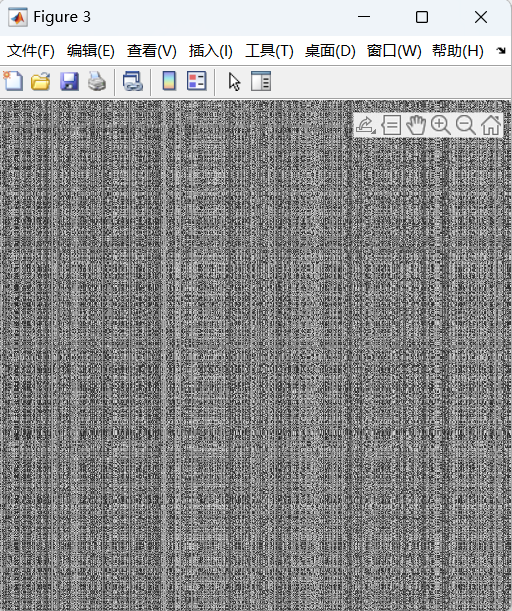
转化的灰度图像



行置乱后的图像：



列置乱后的图像：



第二种：将二维图像展开成一维行/列向量，对向量进行位置置乱。

将二维图像展开成一维向量后进行置乱，再将置乱后的一维向量还原成二维矩阵。比起整行整列操作，效果更好，但运算量更大。

无重复置乱：

把由混沌系统产生的随机向量中重复的随机数只保留一个，然后在末尾按从小到大的顺序补齐没有出现过的元素。

% https://blog.csdn.net/dollar\_jen/article/details/115425450

clc,clear;

P=imread('1.bmp');% P=rgb2gray(P);

iptsetpref('imshowborder','tight');

figure(1);subplot(1,2,1);imshow(P);title('明文图像');

[M,N]=size(P);P=double(P);

n=M\*N;

h=0.002;t=800;

a=10;b=8/3;c=28;r=-1;

x0=1.1;y0=2.2;z0=3.3;w0=4.4;

s=zeros(1,n);

for i=1:n+t

K11=a\*(y0-x0)+w0;K12=a\*(y0-(x0+K11\*h/2))+w0;

K13=a\*(y0-(x0+K12\*h/2))+w0;K14=a\*(y0-(x0+h\*K13))+w0;

x1=x0+(K11+K12+K13+K14)\*h/6;

K21=c\*x1-y0-x1\*z0;K22=c\*x1-(y0+K21\*h/2)-x1\*z0;

K23=c\*x1-(y0+K22\*h/2)-x1\*z0;K24=c\*x1-(y0+h\*K23)-x1\*z0;

y1=y0+(K21+K22+K23+K24)\*h/6;

K31=x1\*y1-b\*z0;K32=x1\*y1-b\*(z0+K31\*h/2);

K33=x1\*y1-b\*(z0+K32\*h/2);K34=x1\*y1-b\*(z0+h\*K33);

z1=z0+(K31+K32+K33+K34)\*h/6;

K41=-y1\*z1+r\*w0;K42=-y1\*z1+r\*(w0+K41\*h/2);

K43=-y1\*z1+r\*(w0+K42\*h/2);K44=-y1\*z1+r\*(w0+h\*K43);

w1=w0+(K41+K42+K43+K44)\*h/6;

x0=x1;y0=y1;z0=z1;w0=w1;

if i>t

s(i-t)=x1;

if mod((i-t),3000)==0

x0=x0+h\*sin(y0);

end

end

end

X=mod(floor((s+100)\*10^10),M\*N)+1;

[~,idx]=unique(X);L=length(idx);X1=zeros(1,M);

X1(1:length(idx))=X(sort(idx));

X1(length(idx)+1:M\*N)=setdiff(1:M\*N,X1);X=X1;

A=P;

for i=1:floor(M\*N/2)

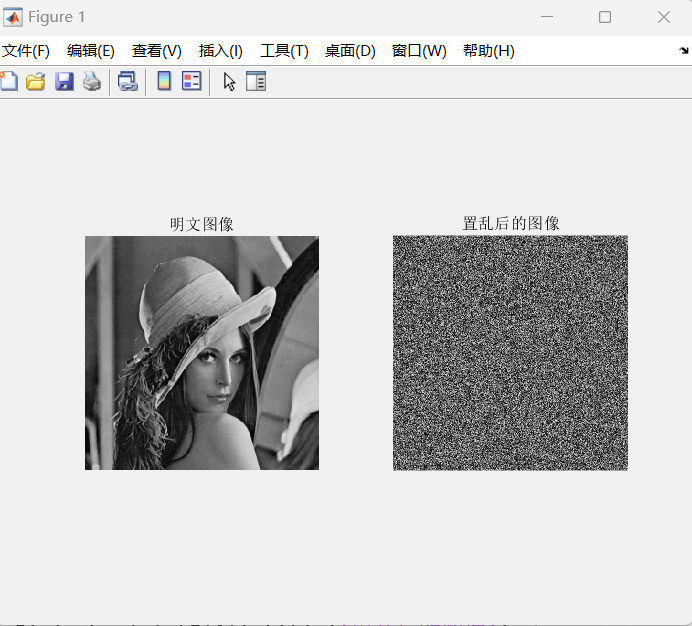
t=A(X(i));A(X(i))=A(X(M\*N-i+1));A(X(M\*N-i+1))=t;

end

A=reshape(A,M,N);

subplot(1,2,2);imshow(uint8(A));title('置乱后的图像');

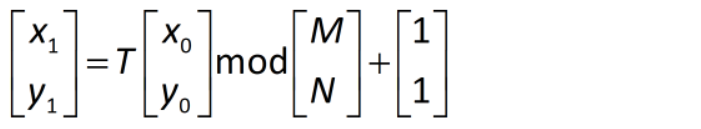
执行结果：



第三种：借助2x2伪随机矩阵进行二维图像置乱

前两种置乱方法都是可逆的，而使用伪随机矩阵进行置乱时，任意随机矩阵都是可用的，包括不可逆的矩阵，只要置乱算法本身是可逆的，就是合法的置乱算法。即：算法可逆并不要求随机矩阵可逆。

原理：



T是变换矩阵，对图像像素点位置（x0,y0）进行变换，得到新的位置（x1,y1），由于像素点位置只可能是非负整数，所以要求变换矩阵T是整数矩阵。比较经典的是Arnold矩阵，它是整数矩阵，并且其逆矩阵仍然是整数矩阵。  
Arnold矩阵长这样：

图片包含 游戏机, 物体, 钟表

描述已自动生成

a,b是整数。

实现：

Arnold伪随机矩阵置乱算法

% https://blog.csdn.net/dollar\_jen/article/details/115425450

clc,clear;

P=imread('1.bmp');% P=rgb2gray(P);

iptsetpref('imshowborder','tight');

figure(1);subplot(1,3,1);imshow(P);title('明文图像');

[M,N]=size(P);P=double(P);

n=2\*M\*N;

h=0.002;t=800;

a=10;b=8/3;c=28;r=-1;

x0=1.1;y0=2.2;z0=3.3;w0=4.4;

s=zeros(1,n);

for i=1:n+t

K11=a\*(y0-x0)+w0;K12=a\*(y0-(x0+K11\*h/2))+w0;

K13=a\*(y0-(x0+K12\*h/2))+w0;K14=a\*(y0-(x0+h\*K13))+w0;

x1=x0+(K11+K12+K13+K14)\*h/6;

K21=c\*x1-y0-x1\*z0;K22=c\*x1-(y0+K21\*h/2)-x1\*z0;

K23=c\*x1-(y0+K22\*h/2)-x1\*z0;K24=c\*x1-(y0+h\*K23)-x1\*z0;

y1=y0+(K21+K22+K23+K24)\*h/6;

K31=x1\*y1-b\*z0;K32=x1\*y1-b\*(z0+K31\*h/2);

K33=x1\*y1-b\*(z0+K32\*h/2);K34=x1\*y1-b\*(z0+h\*K33);

z1=z0+(K31+K32+K33+K34)\*h/6;

K41=-y1\*z1+r\*w0;K42=-y1\*z1+r\*(w0+K41\*h/2);

K43=-y1\*z1+r\*(w0+K42\*h/2);K44=-y1\*z1+r\*(w0+h\*K43);

w1=w0+(K41+K42+K43+K44)\*h/6;

x0=x1;y0=y1;z0=z1;w0=w1;

if i>t

s(i-t)=x1;

if mod((i-t),3000)==0

x0=x0+h\*sin(y0);

end

end

end

X=mod(floor((s+100)\*10^10),10\*max(M,N))+1;

a=reshape(X(1:M\*N),M,N);b=reshape(X(M\*N+1:2\*M\*N),M,N);

A=P;tic;

for i=1:M

for j=1:N

k=mod([1 a(i,j);b(i,j) a(i,j)\*b(i,j)+1]\*[i;j],[M,N])+[1;1];

t=A(i,j);A(i,j)=A(k(1),k(2));A(k(1),k(2))=t;

end

end

toc;subplot(1,3,2);imshow(uint8(A));title('置乱后的图像');

% 从以下可看出，逆过程仍然使用的Arnold矩阵，而不是其逆矩阵。

B=A;

for i=M:-1:1

for j=N:-1:1

k=mod([1 a(i,j);b(i,j) a(i,j)\*b(i,j)+1]\*[i;j],[M,N])+[1;1];

t=B(i,j);B(i,j)=B(k(1),k(2));B(k(1),k(2))=t;

end

end

subplot(1,3,3);imshow(uint8(B));title('还原后的图像');

执行结果：

QR 代码

中度可信度描述已自动生成

上面算法的运行时间是：

图形用户界面

描述已自动生成

**四、实验思考与体会**

单纯的置乱算法无论设计多么复杂，都只是改变像素位置的“纯”加密处理，无法对抗选择明文攻击和已知明文攻击，因此其安全性并不高。现已多使用高效率的置乱算法与扩散结合，或在扩散的过程中同步使用置乱算法。