傅里叶分析和小波分析实验报告

课程名称：傅里叶分析和小波分析

实验项目名称：基于小波变换的图像融合 实验时间：2022.5.2

班级； 信计1901 姓名： 唐川淇 学号： 1131190111

**实验目的：**

* 如题目。

**实 验 环 境:**

Matlab

**实验步骤：**

该程序是用小波的方法对两幅图像进行融合。在低频系数对小波系数采用基于区域方差的加权融合方法处理,高频系数采用基于像素点小波系数绝对值取大的规则进行融合。该方法避免了图像融合过程中因平均化而出现的模糊现象, 融合后的图像内容更加清晰, 更容易识别

function y=imfus(I1,I2);

st=cputime; % 程序开始运行时的cpu时间

I1=imread('untitled3.jpg');

I2=imread('untitled4.jpg');

I1=I1(:,:,1);I2=I2(:,:,1); %读取彩色RGB图像的第一颜色分量

[r1,c1]=size(I1); % 读取图像分解后的小波分解系数矩阵的大小

[r2,c2]=size(I2);

if (r1~=r2)||(c1~=c2)

error('Images dimension mismatch.The image size must be the same!');

return;

end

dim=3;

figure; % 对模糊化后的两幅图像进行小波分解

y1=mywavedec2(I1,dim);

figure;

y2=mywavedec2(I2,dim);

[r,c]=size(y1); % 根据低频融合算法进行图像融合

for i=1:r % 首先取两幅源图像相应的小波分解系数最大值作为融合图像的分解系数

for j=1:c

if ( abs(y1(i,j)) >= abs(y2(i,j)) )

y3(i,j)=y1(i,j);

elseif ( abs(y1(i,j)) < abs(y2(i,j)) )

y3(i,j)=y2(i,j);

end

end

end

LLa=y1(1:r/(2^dim),1:c/(2^dim)); % 调用lowfrefus函数对低频部分的小波分解系数进行融合

LLb=y2(1:r/(2^dim),1:c/(2^dim));

y3(1:r/(2^dim),1:c/(2^dim))=lowfrefus(LLa,LLb);

yr=mywaverec2(y3,dim); % 调用mywaverec2函数重构融合图像

et=cputime-st % 计算程序运行所用的时间

averEntropy1=averEntropy(I1)

averEntropy2=averEntropy(I2)

averEntropy3=averEntropy(yr)

aveGrad1 = avegrad(I1)

aveGrad2 = avegrad(I2)

aveGrad3 = avegrad(yr)

function y=upspl(x);

N=length(x); % 读取输入序列长度

M=2\*N-1; % 输出序列的长度是输入序列长度的2倍再减一

for i=1:M % 输出序列的偶数位为0，奇数位按次序等于相应位置的输入序列元素

if mod(i,2)

y(i)=x((i+1)/2);

else

y(i)=0;

end

end

function [smat,mp,np] = submat(x,p,level);

% 函数 submat 取输入矩阵中以点P为中心、阶数为（2\*level+1）的方阵作为输出的子矩阵

[row,col]=size(x);

m=p(1); n=p(2);

if (m>row)||(n>col)

error('Point p is out of matrix X !');

return;

end

if ((2\*level+1)>row)||((2\*level+1)>col)

error('Too large sample area level !');

return;

end

% 设置子矩阵的边界值

up=m-level; down=m+level;

left=n-level; right=n+level;

% 若子矩阵的某一边界值超出输入矩阵的相应边界，就进行边界处理，

% 即超出边界后往相反方向平移，使其恰好与边界重合

if left<1

right=right+1-left;

left=1;

end

if right>col

left=left+col-right;

right=col;

end

if up<1

down=down+1-up;

up=1;

end

if down>row

up=up+row-down;

down=row;

end

% 获取作为输出的子矩阵，并计算点p在输出的子矩阵中的位置

smat = x(up:down,left:right);

mp=m-up+1;np=n-left+1;

function y=mywaverec2(x,dim)

xd=uint8(x); % 将输入矩阵的数据格式转换为适合显示图像的uint8格式

[m,n]=size(x); % 求出输入矩阵的行列数

for i=1:dim % 对转换矩阵xd进行分界线处理

m=m-mod(m,2);

n=n-mod(n,2);

xd(m/2,1:n)=255;

xd(1:m,n/2)=255;

m=m/2;n=n/2;

end

figure;

subplot(121);imshow(xd);title([ num2str(dim) ' 层小波分解图像']); % 画出带有分界线的分解图像

xr=double(x); % 将输入矩阵的数据格式转换回适合数值处理的double格式

[row,col]=size(xr); % 求出转换矩阵xr的行列数

for i=dim:-1:1 % 重构次序是从内层往外层进行，所以先抽取矩阵 xr 的最内层分解矩阵进行重构

tmp=xr(1:floor(row/2^(i-1)),1:floor(col/2^(i-1))); % 重构的内层矩阵的行列数均为矩阵xr的2^(i-1)

[rt1,ct1]=size(tmp); % 读取待重构矩阵 tmp 的行列数

rt=rt1-mod(rt1,2);ct=ct1-mod(ct1,2);

rLL=tmp(1:rt/2,1:ct/2); % 将待重构矩阵 tmp 分解为四个部分

rHL=tmp(1:rt/2,ct/2+1:ct);

rLH=tmp(rt/2+1:rt,1:ct/2);

rHH=tmp(rt/2+1:rt,ct/2+1:ct);

tmp(1:rt,1:ct)=myidwt2(rLL,rHL,rLH,rHH); % 将重构结果返回到矩阵 tmp

xr(1:rt1,1:ct1)=tmp; % 把矩阵 tmp 的数据返回到矩阵 xr 的相应区域，准备下一个外层的重构

end

y=xr; % 重构结束后得到的矩阵xr即为输出矩阵 y

yu=uint8(xr); % 将矩阵xr的数据格式转换为适合显示图像的uint8格式

subplot(122);imshow(yu);title('小波重构图像');

function y=mywavedec2(x,dim)

x=modmat(x,dim); % 首先规范化输入矩阵，使其行列数均能被 2^dim 整除，从而使分解顺利进行

subplot(121);imshow(x);title('原始图像'); % 画出规范化后的源图像

[m,n]=size(x); % 求出规范化矩阵x的行列数

xd=double(x); % 将矩阵x的数据格式转换为适合数值处理的double格式

for i=1:dim

xd=modmat(xd,1);

[dLL,dHL,dLH,dHH]=mydwt2(xd); % 矩阵小波分解

tmp=[dLL,dHL;dLH,dHH]; % 将分解系数存入缓存矩阵

xd=dLL; % 将缓存矩阵左上角部分的子矩阵作为下一层分解的源矩阵

[row,col]=size(tmp); % 求出缓存矩阵的行列数

y(1:row,1:col)=tmp; % 将缓存矩阵存入输出矩阵的相应行列

end

yd=uint8(y); % 将输出矩阵的数据格式转换为适合显示图像的uint8格式

for i=1:dim % 对矩阵 yd 进行分界线处理，画出分解图像的分界线

m=m-mod(m,2);

n=n-mod(n,2);

yd(m/2,1:n)=255;

yd(1:m,n/2)=255;

m=m/2;n=n/2;

end

subplot(122);imshow(yd);title([ num2str(dim) ' 层小波分解图像']);

function y=myidwt2(LL,HL,LH,HH);

lpr=[1 1];hpr=[1 -1]; % 默认的低通、高通滤波器

tmp\_mat=[LL,HL;LH,HH]; % 将输入的四个矩阵组合为一个矩阵

[row,col]=size(tmp\_mat); % 求出组合矩阵的行列数

for k=1:col % 首先对组合矩阵tmp\_mat的每一列，分开成上下两半

ca1=tmp\_mat(1:row/2,k); % 分开的两部分分别作为平均系数序列ca1、细节系数序列cd1

cd1=tmp\_mat(row/2+1:row,k);

tmp1=myidwt(ca1,cd1,lpr,hpr); % 重构序列

yt(:,k)=tmp1; % 将重构序列存入待输出矩阵 yt 的相应列，此时 y=[L|H]

end

for j=1:row % 将输出矩阵 y 的每一行，分开成左右两半

ca2=yt(j,1:col/2); % 分开的两部分分别作为平均系数序列ca2、细节系数序列cd2

cd2=yt(j,col/2+1:col);

tmp2=myidwt(ca2,cd2,lpr,hpr); % 重构序列

yt(j,:)=tmp2; % 将重构序列存入待输出矩阵 yt 的相应行，得到最终的输出矩阵 y=yt

end

y=yt;

function y = myidwt(cA,cD,lpr,hpr);

lca=length(cA); % 求出平均、细节部分分解系数的长度

lcd=length(cD);

while (lcd)>=(lca) % 每一层重构中，cA 和 cD 的长度要相等，故每层重构后，

% 若lcd小于lca，则重构停止，这时的 cA 即为重构信号序列 y 。

upl=upspl(cA); % 对平均部分系数进行上抽样

cvl=conv(upl,lpr); % 低通卷积

cD\_up=cD(lcd-lca+1:lcd); % 取出本层重构所需的细节部分系数，长度与本层平均部分系数的长度相等

uph=upspl(cD\_up); % 对细节部分系数进行上抽样

cvh=conv(uph,hpr); % 高通卷积

cA=cvl+cvh; % 用本层重构的序列更新cA，以进行下一层重构

cD=cD(1:lcd-lca); % 舍弃本层重构用到的细节部分系数，更新cD

lca=length(cA); % 求出下一层重构所用的平均、细节部分系数的长度

lcd=length(cD);

end % lcd < lca，重构完成，结束循环

y=cA; % 输出的重构序列 y 等于重构完成后的平均部分系数序列 cA

function [LL,HL,LH,HH]=mydwt2(x);

lpd=[1/2 1/2];hpd=[-1/2 1/2]; % 默认的低通、高通滤波器

[row,col]=size(x); % 读取输入矩阵的大小

for j=1:row % 首先对输入矩阵的每一行序列进行一维离散小波分解

tmp1=x(j,:);

[ca1,cd1]=mydwt(tmp1,lpd,hpd,1);

x(j,:)=[ca1,cd1]; % 将分解系数序列再存入矩阵x中，得到[L|H]

end

for k=1:col % 再对输入矩阵的每一列序列进行一维离散小波分解

tmp2=x(:,k);

[ca2,cd2]=mydwt(tmp2,lpd,hpd,1);

x(:,k)=[ca2,cd2]; % 将分解所得系数存入矩阵x中，得到[LL,Hl;LH,HH]

end

LL=x(1:row/2,1:col/2); % LL是矩阵x的左上角部分

LH=x(row/2+1:row,1:col/2); % LH是矩阵x的左下角部分

HL=x(1:row/2,col/2+1:col); % HL是矩阵x的右上角部分

HH=x(row/2+1:row,col/2+1:col); % HH是矩阵x的右下角部分

function [cA,cD] = mydwt(x,lpd,hpd,dim);

cA=x; % 初始化cA，cD

cD=[];

for i=1:dim

cvl=conv(cA,lpd); % 低通滤波，为了提高运行速度，调用MATLAB提供的卷积函数conv()

dnl=downspl(cvl); % 通过下抽样求出平均部分的分解系数

cvh=conv(cA,hpd); % 高通滤波

dnh=downspl(cvh); % 通过下抽样求出本层分解后的细节部分系数

cA=dnl; % 下抽样后的平均部分系数进入下一层分解

cD=[cD,dnh]; % 将本层分解所得的细节部分系数存入序列cD

end

function y=modmat(x,dim)

[row,col]=size(x); % 求出输入矩阵的行列数row,col

rt=row - mod(row,2^dim); % 将row,col分别减去本身模 2^dim 得到的数

ct=col - mod(col,2^dim); % 所得的差为rt、ct，均能被 2^dim 整除

y=x(1:rt,1:ct); % 输出矩阵 y 为输入矩阵 x 的 rt\*ct 维子矩阵

function y = lowfrefus(A,B);

[row,col]=size(A); % 求出分解系数矩阵的行列数

alpha=0.5; % alpha是方差匹配度比较的阈值

for i=1:row % 根据低频融合算法，先求出矩阵A,B中以点P为中心的区域方差和方差匹配度

for j=1:col % 再根据方差匹配度与阈值的比较确定融合图像的小波分解系数

[m2p(i,j),Ga(i,j),Gb(i,j)] = area\_var\_match(A,B,[i,j]);

Wmin=0.5-0.5\*((1-m2p(i,j))/(1-alpha));

Wmax=1-Wmin;

if m2p(i,j)<alpha % m2p表示方差匹配度

if Ga(i,j)>=Gb(i,j) % 若匹配度小于阈值，则取区域方差大的相应点的分解系数作为融合图像的分解系数

y(i,j)=A(i,j);

else

y(i,j)=B(i,j);

end

else % 若匹配度大于阈值，则采取加权平均方法得出相应的分解系数

if Ga(i,j)>=Gb(i,j)

y(i,j)=Wmax\*A(i,j)+Wmin\*B(i,j);

else

y(i,j)=Wmin\*A(i,j)+Wmax\*B(i,j);

end

end

end

end

function w = weivec(x,p);

[r,c]=size(x);

p1=p(1); p2=p(2);

sig=1;

for i=1:r

for j=1:c

w(i,j)=0.5\*(gaussmf(i,[sig p1])+gaussmf(j,[sig p2]));

end

end

function [m2p,Ga,Gb] = area\_var\_match(A,B,p);

level=1; % 设置区域的大小

[subA,mpa,npa]=submat(A,p,level); % submat 函数取输入矩阵中以点P为中心、阶数为（2\*level+1）的方阵作为子矩阵

[subB,mpb,npb]=submat(B,p,level);

[r,c]=size(subA);

w=weivec(subA,[mpa npa]); % 获取子矩阵的权值分布

averA=sum(sum(subA))/(r\*c); % 计算子矩阵的平均值

averB=sum(sum(subB))/(r\*c);

Ga=sum(sum(w.\*(subA-averA).^2)); % 计算子矩阵的区域方差

Gb=sum(sum(w.\*(subB-averB).^2));

if (Ga==0)&(Gb==0) % 计算两个子矩阵的区域方差匹配度

m2p=0;

else

m2p=2\*sum(sum(w.\*abs(subA-averA).\*abs(subB-averB)))/(Ga+Gb);

end

function y=flor(x);

y=x-mod(x,1);

function y=downspl(x);

N=length(x); % 读取输入序列长度

M=floor(N/2); % 输出序列的长度是输入序列长度的一半（带小数时取整数部分）

i=1:M;

y(i)=x(2\*i);

%^^^^^^^^^^^^^^^^^^ Compute Entropy Of Image ^^^^^^^^^^^^^

function AVERENTROPY=averEntropy(img)

% remove zero entries in I1

I1=double(img);

I1(I1==0) = [];

% normalize I1 so that sum(I1) is one.

I1 = I1./numel(I1);

Entropy = -sum(I1.\*log2(I1));

AVERENTROPY = sum(Entropy)/size(Entropy,2);

function AVEGRAD=avegrad(img)

%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% this function is used to calculate

%%%% the average gradient of an image.

%%%% 平均梯度可敏感地反映图像对微小细节反差表达的能力，可用来评价图像的模糊程度

%%%% 在图像中，某一方向的灰度级变化率大，它的梯度也就大。因此，可以用平均梯度值

%%%% 来衡量图像的清晰度，还同时反映出图像中微小细节反差和纹理变换特征。

img=double(img);

[M,N]=size(img);

gradval=zeros(M,N); %%% save the gradient value of single pixel

diffX=zeros(M,N); %%% save the differential value of X orient

diffY=zeros(M,N); %%% save the differential value of Y orient

tempX=zeros(M,N);

tempY=zeros(M,N);

tempX(1:M,1:(N-1))=img(1:M,2:N);

tempY(1:(M-1),1:N)=img(2:M,1:N);

diffX=img-tempX;

diffY=img-tempY;

diffX(1:M,N)=0; %%% the boundery set to 0

diffY(M,1:N)=0;

diffX=diffX.\*diffX;

diffY=diffY.\*diffY;

AVEGRAD=sum(sum(diffX+diffY));

AVEGRAD=sqrt(AVEGRAD);

AVEGRAD=AVEGRAD/((M-1)\*(N-1));

**实 验 心 得：**

重新修正代码：

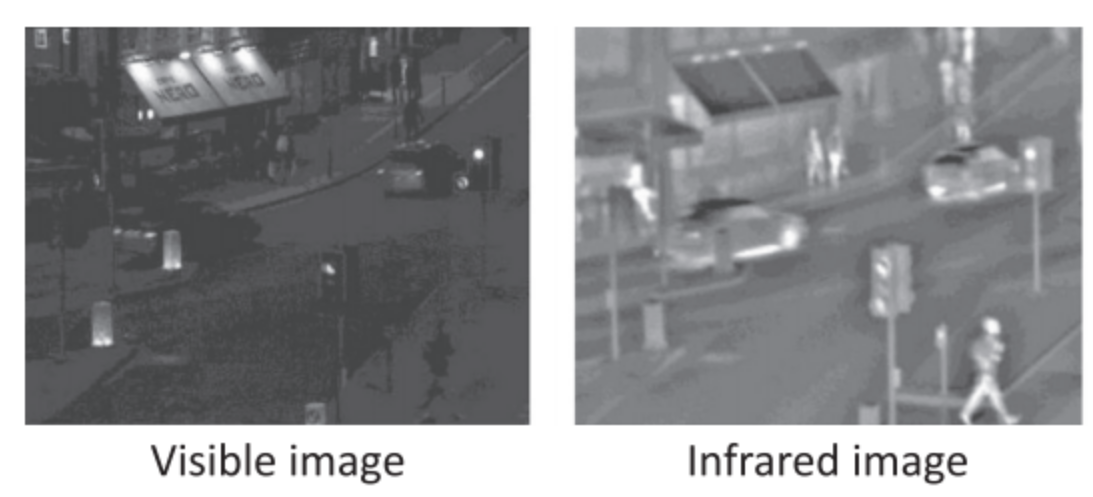
|  |
| --- |
|  |
| **clear;clc;close all;**  **I2=imread('ly.jpg');**  **I1=imread('ad.png');**    **figure;**  **subplot(1,3,1);imshow(I1);title('原始图像1');**  **subplot(1,3,2);imshow(I2);title('原始图像2');**    **C1=imfus(I1,I2,1);**  **C2=imfus(I1,I2,2);**  **C3=imfus(I1,I2,3);**  **subplot(1,3,3);imshow(cat(3,C1,C2,C3));title('融合图像');**    **function y=imfus(C,C\_,m)**  **I1=C(:,:,m);I2=C\_(:,:,m); %读取彩色RGB图像的颜色分量**    **[r1,c1]=size(I1); % 读取图像分解后的小波分解系数矩阵的大小**  **[r2,c2]=size(I2);**  **if (r1~=r2)||(c1~=c2)**  **error('Images dimension mismatch.The image size must be the same!');**  **return;**  **end**    **dim=3;**  **y1=mywavedec2(I1,dim);% 对模糊化后的两幅图像进行小波分解**  **y2=mywavedec2(I2,dim);**    **[r,c]=size(y1); % 根据低频融合算法进行图像融合**  **for i=1:r % 首先取两幅源图像相应的小波分解系数最大值作为融合图像的分解系数**  **for j=1:c**  **if ( abs(y1(i,j)) >= abs(y2(i,j)) )**  **y3(i,j)=y1(i,j);**  **elseif ( abs(y1(i,j)) < abs(y2(i,j)) )**  **y3(i,j)=y2(i,j);**  **end**  **end**  **end**    **LLa=y1(1:r/(2^dim),1:c/(2^dim)); % 调用lowfrefus函数对低频部分的小波分解系数进行融合**  **LLb=y2(1:r/(2^dim),1:c/(2^dim));**  **y3(1:r/(2^dim),1:c/(2^dim))=lowfrefus(LLa,LLb);**    **yr=mywaverec2(y3,dim); % 调用mywaverec2函数重构融合图像**  **y=uint8(yr);**  **end**    **function y=upspl(x)**  **N=length(x); % 读取输入序列长度**  **M=2\*N-1; % 输出序列的长度是输入序列长度的2倍再减一**  **for i=1:M % 输出序列的偶数位为0，奇数位按次序等于相应位置的输入序列元素**  **if mod(i,2)**  **y(i)=x((i+1)/2);**  **else**  **y(i)=0;**  **end**  **end**  **end**    **function [smat,mp,np] = submat(x,p,level)**  **% 函数 submat 取输入矩阵中以点P为中心、阶数为（2\*level+1）的方阵作为输出的子矩阵**  **[row,col]=size(x);**  **m=p(1); n=p(2);**    **if (m>row)||(n>col)**  **error('Point p is out of matrix X !');**  **return;**  **end**  **if ((2\*level+1)>row)||((2\*level+1)>col)**  **error('Too large sample area level !');**  **return;**  **end**  **% 设置子矩阵的边界值**  **up=m-level; down=m+level;**  **left=n-level; right=n+level;**  **% 若子矩阵的某一边界值超出输入矩阵的相应边界，就进行边界处理，**  **% 即超出边界后往相反方向平移，使其恰好与边界重合**  **if left<1**  **right=right+1-left;**  **left=1;**  **end**  **if right>col**  **left=left+col-right;**  **right=col;**  **end**  **if up<1**  **down=down+1-up;**  **up=1;**  **end**  **if down>row**  **up=up+row-down;**  **down=row;**  **end**  **% 获取作为输出的子矩阵，并计算点p在输出的子矩阵中的位置**  **smat = x(up:down,left:right);**  **mp=m-up+1;np=n-left+1;**  **end**    **function y=mywaverec2(x,dim)**  **xd=uint8(x); % 将输入矩阵的数据格式转换为适合显示图像的uint8格式**  **[m,n]=size(x); % 求出输入矩阵的行列数**    **xr=double(x); % 将输入矩阵的数据格式转换回适合数值处理的double格式**  **[row,col]=size(xr); % 求出转换矩阵xr的行列数**    **for i=dim:-1:1 % 重构次序是从内层往外层进行，所以先抽取矩阵 xr 的最内层分解矩阵进行重构**  **tmp=xr(1:floor(row/2^(i-1)),1:floor(col/2^(i-1))); % 重构的内层矩阵的行列数均为矩阵xr的2^(i-1)**  **[rt1,ct1]=size(tmp); % 读取待重构矩阵 tmp 的行列数**  **rt=rt1-mod(rt1,2);ct=ct1-mod(ct1,2);**  **rLL=tmp(1:rt/2,1:ct/2); % 将待重构矩阵 tmp 分解为四个部分**  **rHL=tmp(1:rt/2,ct/2+1:ct);**  **rLH=tmp(rt/2+1:rt,1:ct/2);**  **rHH=tmp(rt/2+1:rt,ct/2+1:ct);**  **tmp(1:rt,1:ct)=myidwt2(rLL,rHL,rLH,rHH); % 将重构结果返回到矩阵 tmp**  **xr(1:rt1,1:ct1)=tmp; % 把矩阵 tmp 的数据返回到矩阵 xr 的相应区域，准备下一个外层的重构**  **end**  **y=xr; % 重构结束后得到的矩阵xr即为输出矩阵 y**  **end**    **function y=mywavedec2(x,dim)**  **x=modmat(x,dim); % 首先规范化输入矩阵，使其行列数均能被 2^dim 整除，从而使分解顺利进行**  **[m,n]=size(x); % 求出规范化矩阵x的行列数**  **xd=double(x); % 将矩阵x的数据格式转换为适合数值处理的double格式**    **for i=1:dim**  **xd=modmat(xd,1);**  **[dLL,dHL,dLH,dHH]=mydwt2(xd); % 矩阵小波分解**  **tmp=[dLL,dHL;dLH,dHH]; % 将分解系数存入缓存矩阵**  **xd=dLL; % 将缓存矩阵左上角部分的子矩阵作为下一层分解的源矩阵**  **[row,col]=size(tmp); % 求出缓存矩阵的行列数**  **y(1:row,1:col)=tmp; % 将缓存矩阵存入输出矩阵的相应行列**  **end**  **end**    **function y=myidwt2(LL,HL,LH,HH)**  **lpr=[1 1];hpr=[1 -1]; % 默认的低通、高通滤波器**  **tmp\_mat=[LL,HL;LH,HH]; % 将输入的四个矩阵组合为一个矩阵**  **[row,col]=size(tmp\_mat); % 求出组合矩阵的行列数**    **for k=1:col % 首先对组合矩阵tmp\_mat的每一列，分开成上下两半**  **ca1=tmp\_mat(1:row/2,k); % 分开的两部分分别作为平均系数序列ca1、细节系数序列cd1**  **cd1=tmp\_mat(row/2+1:row,k);**  **tmp1=myidwt(ca1,cd1,lpr,hpr); % 重构序列**  **yt(:,k)=tmp1; % 将重构序列存入待输出矩阵 yt 的相应列，此时 y=[L|H]**  **end**    **for j=1:row % 将输出矩阵 y 的每一行，分开成左右两半**  **ca2=yt(j,1:col/2); % 分开的两部分分别作为平均系数序列ca2、细节系数序列cd2**  **cd2=yt(j,col/2+1:col);**  **tmp2=myidwt(ca2,cd2,lpr,hpr); % 重构序列**  **yt(j,:)=tmp2; % 将重构序列存入待输出矩阵 yt 的相应行，得到最终的输出矩阵 y=yt**  **end**  **y=yt;**  **end**    **function y = myidwt(cA,cD,lpr,hpr)**  **lca=length(cA); % 求出平均、细节部分分解系数的长度**  **lcd=length(cD);**    **while (lcd)>=(lca) % 每一层重构中，cA 和 cD 的长度要相等，故每层重构后，**  **% 若lcd小于lca，则重构停止，这时的 cA 即为重构信号序列 y 。**  **upl=upspl(cA); % 对平均部分系数进行上抽样**  **cvl=conv(upl,lpr); % 低通卷积**    **cD\_up=cD(lcd-lca+1:lcd); % 取出本层重构所需的细节部分系数，长度与本层平均部分系数的长度相等**  **uph=upspl(cD\_up); % 对细节部分系数进行上抽样**  **cvh=conv(uph,hpr); % 高通卷积**    **cA=cvl+cvh; % 用本层重构的序列更新cA，以进行下一层重构**  **cD=cD(1:lcd-lca); % 舍弃本层重构用到的细节部分系数，更新cD**  **lca=length(cA); % 求出下一层重构所用的平均、细节部分系数的长度**  **lcd=length(cD);**  **end % lcd < lca，重构完成，结束循环**  **y=cA; % 输出的重构序列 y 等于重构完成后的平均部分系数序列 cA**  **end**    **function [LL,HL,LH,HH]=mydwt2(x)**  **lpd=[1/2 1/2];hpd=[-1/2 1/2]; % 默认的低通、高通滤波器**  **[row,col]=size(x); % 读取输入矩阵的大小**    **for j=1:row % 首先对输入矩阵的每一行序列进行一维离散小波分解**  **tmp1=x(j,:);**  **[ca1,cd1]=mydwt(tmp1,lpd,hpd,1);**  **x(j,:)=[ca1,cd1]; % 将分解系数序列再存入矩阵x中，得到[L|H]**  **end**    **for k=1:col % 再对输入矩阵的每一列序列进行一维离散小波分解**  **tmp2=x(:,k);**  **[ca2,cd2]=mydwt(tmp2,lpd,hpd,1);**  **x(:,k)=[ca2,cd2]; % 将分解所得系数存入矩阵x中，得到[LL,Hl;LH,HH]**  **end**    **LL=x(1:row/2,1:col/2); % LL是矩阵x的左上角部分**  **LH=x(row/2+1:row,1:col/2); % LH是矩阵x的左下角部分**  **HL=x(1:row/2,col/2+1:col); % HL是矩阵x的右上角部分**  **HH=x(row/2+1:row,col/2+1:col); % HH是矩阵x的右下角部分**  **end**    **function [cA,cD] = mydwt(x,lpd,hpd,dim)**  **cA=x; % 初始化cA，cD**  **cD=[];**  **for i=1:dim**  **cvl=conv(cA,lpd); % 低通滤波，为了提高运行速度，调用MATLAB提供的卷积函数conv()**  **dnl=downspl(cvl); % 通过下抽样求出平均部分的分解系数**  **cvh=conv(cA,hpd); % 高通滤波**  **dnh=downspl(cvh); % 通过下抽样求出本层分解后的细节部分系数**  **cA=dnl; % 下抽样后的平均部分系数进入下一层分解**  **cD=[cD,dnh]; % 将本层分解所得的细节部分系数存入序列cD**  **end**  **end**    **function y=modmat(x,dim) %terrible function**  **[row,col]=size(x); % 求出输入矩阵的行列数row,col**  **rt=row - mod(row,2^dim); % 将row,col分别减去本身模 2^dim 得到的数**  **ct=col - mod(col,2^dim); % 所得的差为rt、ct，均能被 2^dim 整除**  **y=x(1:rt,1:ct); % 输出矩阵 y 为输入矩阵 x 的 rt\*ct 维子矩阵**  **end**    **function y = lowfrefus(A,B)**  **[row,col]=size(A); % 求出分解系数矩阵的行列数**  **alpha=0.5; % alpha是方差匹配度比较的阈值**  **for i=1:row % 根据低频融合算法，先求出矩阵A,B中以点P为中心的区域方差和方差匹配度**  **for j=1:col % 再根据方差匹配度与阈值的比较确定融合图像的小波分解系数**  **[m2p(i,j),Ga(i,j),Gb(i,j)] = area\_var\_match(A,B,[i,j]);**  **Wmin=0.5-0.5\*((1-m2p(i,j))/(1-alpha));**  **Wmax=1-Wmin;**  **if m2p(i,j)<alpha % m2p表示方差匹配度**  **if Ga(i,j)>=Gb(i,j) % 若匹配度小于阈值，则取区域方差大的相应点的分解系数作为融合图像的分解系数**  **y(i,j)=A(i,j);**  **else**  **y(i,j)=B(i,j);**  **end**  **else % 若匹配度大于阈值，则采取加权平均方法得出相应的分解系数**  **if Ga(i,j)>=Gb(i,j)**  **y(i,j)=Wmax\*A(i,j)+Wmin\*B(i,j);**  **else**  **y(i,j)=Wmin\*A(i,j)+Wmax\*B(i,j);**  **end**  **end**  **end**  **end**  **end**    **function w = weivec(x,p)**  **[r,c]=size(x);**  **p1=p(1);**  **p2=p(2);**  **sig=1;**  **for i=1:r**  **for j=1:c**  **w(i,j)=0.5\*(gaussmf(i,[sig p1])+gaussmf(j,[sig p2]));**  **end**  **end**  **end**    **function [m2p,Ga,Gb] = area\_var\_match(A,B,p)**  **level=1; % 设置区域的大小**  **[subA,mpa,npa]=submat(A,p,level); % submat 函数取输入矩阵中以点P为中心、阶数为（2\*level+1）的方阵作为子矩阵**  **[subB,mpb,npb]=submat(B,p,level);**    **[r,c]=size(subA);**  **w=weivec(subA,[mpa npa]); % 获取子矩阵的权值分布**    **averA=sum(sum(subA))/(r\*c); % 计算子矩阵的平均值**  **averB=sum(sum(subB))/(r\*c);**    **Ga=sum(sum(w.\*(subA-averA).^2)); % 计算子矩阵的区域方差**  **Gb=sum(sum(w.\*(subB-averB).^2));**  **if (Ga==0)&(Gb==0) % 计算两个子矩阵的区域方差匹配度**  **m2p=0;**  **else**  **m2p=2\*sum(sum(w.\*abs(subA-averA).\*abs(subB-averB)))/(Ga+Gb);**  **end**  **end**    **function y=downspl(x)**  **N=length(x); % 读取输入序列长度**  **M=floor(N/2); % 输出序列的长度是输入序列长度的一半（带小数时取整数部分）**  **i=1:M;**  **y(i)=x(2\*i);**  **end** |

程序是用小波的方法对两幅图像进行融合。在低频系数对小波系数采用基于区域方差的加权融合方法处理,高频系数采用基于像素点小波系数绝对值取大的规则进行融合。

该方法避免了图像融合过程中因平均化而出现的模糊现象, 融合后的图像内容更加清晰, 更容易识别。但在modmat函数处理中，丢掉了大量原始图像数据，使得最终的融合图像颜色部分缺失，尝试了几种方法效果都不太好。

****

通过组合红外和可见光源信息而产生的融合图像具有许多互补性。 例如[1]，这些图像通过传递某些纹理使红外图像更生动。 因此，融合算法可以嵌入许多应用程序并改进原始方法。 使用融合技术的主要应用包括识别，目标检测，跟踪，监视，彩色视觉和遥感。



如图所示，监视需要清晰，健壮和富有表现力的图像或视频。红外和可见图像的互补特性使它们成为完成此任务的良好融合源。给定红外和可见图像的图像特征不同，它们可以在各个方面提供对象信息。红外图像捕获不受光照，天气和其他干扰变化影响的物体的热辐射。然而，红外图像的空间分辨率低，因此很难区分细节。通过对比，可见图像的空间分辨率高并且色彩丰富，这使得其中的物体生动且易于识别。这两幅图像的融合显着提高了可视性，并有助于一些实际应用。监视方法通常用于处理视频，因此需要快速实施。下面使用小波工具对图像进行融合。



|  |
| --- |
| 代码 |
| close all  clear  clc  %% Read Images  % the size of images must be equal  a=imread('aaa.png');  b=imread('bbb.png');  %% Wavelet Transform  [a1,b1,c1,d1]=dwt2(a,'db2');  [a2,b2,c2,d2]=dwt2(b,'db2');  [k1,k2]=size(a1);  %% Fusion Rules  %% Average Rule  for i=1:k1  for j=1:k2  a3(i,j)=(a1(i,j)+a2(i,j))/2;  end  end  %% Max Rule  for i=1:k1  for j=1:k2  b3(i,j)=max(b1(i,j),b2(i,j));  c3(i,j)=max(c1(i,j),c2(i,j));  d3(i,j)=max(d1(i,j),d2(i,j));  end  end  %% Inverse Wavelet Transform  c=idwt2(a3,b3,c3,d3,'db2');  imshow(a)  title('First Image')  figure,imshow(b)  title('Second Image')  figure,imshow(c,[])  title('Fused Image')  fusedimage=(c-min(c(:)))/(max(c(:))-min(c(:)))\*255;  fusedimage1=uint8(fusedimage);  imwrite(fusedimage1,'fusedimage.bmp') |

**附 录：**

[1]钱小燕,韩磊,王帮峰.红外与可见光图像快速融合算法[J].计算机辅助设计与图形学学报,2011,23(7):1211-1216

备注：以上各项空白处若填写不够，可自行扩展