تبدیل موجک

علی نصیری سروی

اطلاعات گزارش

تاریخ: 99/03/15

واژگان کلیدي:
تبدیل موجک
هرم لاپلاسی
هرم موجک
حذف نویز

چکیده

در این تمرین در ابتدا به معرفی تبدیل موجک و دلایل استفاده از آن به جای تبدیل فوریه میپردازیم.

سپس هرم لاپلاسین و هرم موجک را در مرحله بعد ایجاد کرده و در نهایت تلاش میکنیم سعی میکنیم به کمک تبدیل موجک،حذف نویز داشته باشیم.

1-مقدمه

تبدیل موجک (Wavelet Transform) یکی از تبدیلات مهم ریاضی است که در حوزه‌های مختلف علوم کاربرد دارد. ایده اصلی تبدیل موجک این است که بر ضعف‌ها و محدودیت‌های موجود در تبدیل فوریه غلبه کند. این تبدیل را بر خلاف تبدیل فوریه، می‌توان در مورد سیگنال‌های غیر ایستا و سیستم‌های دینامیک نیز مورد استفاده قرار داد.

2-شرح تکنیکال

سگینال‌های ایستا:حاوی اجزای طیفی هستند که با زمان تغییر نمی‌کنند.

- تمام اجزای طیفی همیشه وجود دارند.
- نیازی به اطلاعات زمانی نیست.
- FT برای سیگنالهای ایستا خوب عمل می‌کند.

سیگنال‌های غیرایستا : محتوای طیفی متغیر بازمان دارند.

- چگونه می‌توان فهمید که جزئیات طیفی کی ظاهر می‌شوند.
- FT تنها مشخص می‌کند که چه اجزایی در طیف وجود دارد و نه زمانی که آن طیف‌ها وجود دارند.
- نیاز به روش‌هایی برای تعیین زمانی اجزای طیفی است.

تبدیل فوریه (FT) تمامی اجزای موجود در دل سیگنال را شناسایی می‌کند، اما هیچ اطلاعاتی در خصوص مکان (زمان) این اجزا ارایه نمی‌کند.

6.1 هرم رزولوشن

ما به بیانی برای تصویر احتیاج داریم که بگوید چه چیزی در تصویر، کجا اتفاق افتاده است.

تبدیل فوریه اطلاعات موجود در تصویر (What?) را بیان می‌کند، اما پاسخ به محل وقوع (Where?) را ارایه نمی‌کند.

بیان تصویر در حوزه مکان به شما مکان وقوع را می‌دهد، ولی نمیدانید که آنجا چه اتفاق افتاده.

برای رفع این مشکل از تبدیل فوریه زمان کوتاه استفاده میشود اما مشکلی که دارد سایز پنجره ثابت است.

طبق اصل عدم قطعیت Heisenberg:

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi}$$

یعنی نمی‌توانیم هر دوی رزلوشن زمانی و فرکانسی را به دلخواه زیاد کنیم.

به همین دلیل از تبدیل موجک به جای تبدیل فوریه زمان کوتاه استفاده میکنیم.

در این تبدیل با استفاده از یک پنجره با طول متغیر می توان بر مشکل از پیش تعیین کردن رزلوشن غلبه کرد.

پنجره های با طول متغیر برای فرکانس های مختلف استفاده می‌شوند:

- آنالیز فرکانس های بالا استفاده از پنجره‌های باریک‌تر برای رزلوشن زمانی بهتر
- آنالیز فرکانس های پایین استفاده از پنجره‌های عریض برای رزلوشن فرکانسی بهتر

اصل عدم قطعیت Heisenberg همچنان در نظر گرفته می‌شود.

تابعی که برای پنجره‌ای کردن سیگنال استفاده می‌شود، موجک نامیده می‌شود.فرمول آن به صورت زیر است:

اطلاعات تصویر در رزلوشن های مختلف قرار دارد.با اعمال عملیات بر روی یک سطح رزولوشن تصویر و downsample کردن به سطح پایین تر میرویم.این سطوح در کنار هم هرم رزولوشن را تشکیل میدهند.

6.1.1 هرم لاپلاسی:

با استفاده از تفاضل بین تصویر در رزلوشن خاصی از هرم گوسی و نسخه ی بزرگ شده با رزلوشن پایین تر بدست می آید. در واقع در هر مرحله به این صورت عمل میکنیم:

- تصویر فعلی این مرحله را از یک فیلتر گاوسی (یا فیلتر پایین گذر) عبور میدهیم.
- تصویر که فرکانس های بالای آن حذف شده را downsmaple میکنیم.
- تصویر downsample شده را upsample کرده و آن را از تصویر این مرحله کم میکنیم.
- لاپلاسی این مرحله بدست می آید.
- تصویر downsample شده،تصویر مرحله بعد هرم می باشد.
- در نهایت تمام لاپلاسین ها+تصویر مرحله آخر را ذخیره میکنیم.

همچنین باید دقت شود در هر مرحله سایز تصویر نصف میشود.

اگر تصویرمان $N \times N$ باشد که $N = 2^J$، حداکثر تعداد مرحله ای که میتوان داشت برابر با $\log_2 N = J + 1$ میباشد.

مجموع تعداد پیکسل های تمام مراحل برابر

$$T = 1 \times 1 + 2 \times 2 + 4 \times 4 + 8 \times 8 + \cdots$$
$$= 2^0 + 2^2 + 2^4 + 2^6 + \cdots$$

می باشد که اگر فرمول دنباله هندسی را در نظر بگیریم:

$$Total\ pixels = 1 \times \frac{4^{J+1}-1}{4-1}$$

حال اگر تعداد پیکسل های تصویر اصلی را حساب کنیم

خواهیم داشت:

$$Total\ pixels\ original = 2^J \times 2^J = 2^{J+1}$$

این بدان معناست که تعداد پیکسل های هرم لاپلاسی بیشتر از حالت اورجینال میشود.

سوالی که پیش می آید این است که چرا به خودمان زحمت میدهیم و تصویر را در پیکسل های بیشتری ذخیره میکنیم؟

در صورتی که جزئیات را در نظر بگیریم،اکثر آنها ماتریس های *sparse* می باشند که حجم کمتری را اشغال میکنند.درست است که تعداد پیکسل ها بیشتر میشود، اما به دلیل استفاده از این ماتریکس ها فشرده سازی خواهیم داشت.

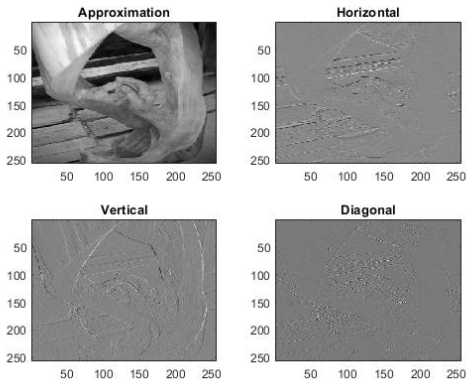
6.1.2 هرم لاپلاسی:

در مرحله قبل نحوه ایجاد هرم لاپلاسین را توضیح دادیم.حال به دنبال نحوه بازسازی تصویر اصلی از هرم هستیم.تصویر مرحله آخر را که جدا کرده ایم upsample کرده (به کمک درون یابی pixel replication) و سایز آن را دو برابر میکنیم.حال این تصویر را با لاپلاسین این مرحله جمع میکنیم. این دو مرحله را انقدر تکرار کرده تا تصویر با لاپلاسین مرحله اول جمع شود و به تصویر اصلی برسیم.

6.1.3 هرم موجک:

این هرم به کمک خاصیت جداپذیری تبدیل موجک و به کمک فیلترهای یک بعدی haar ایجاد میشود.

با هر بار اعمال فیلتر به تصویر،تصویر را به چهار بخش تقسیم میکنیم.



بخش approximation تصویر با رزولوشن کوچک تر می باشد.اگر دوباره به آن فیلتر را اعمال کنیم،به چهار بخش شکسته خواهد شد.

بخش های vertical,horizontal,dianogal هر کدام جزئیات عمودی،افقی و قطری تصویر را در آن رزولوشن خاص نشان می دهد.

از نظر تعداد پیکسل ها،تفاوتی با تصویر اصلی ندارد ولی به دلیل sparse بودن ماتریس های جزئیات،میتواند در فشرده سازی استفاده شود.

اگر بخواهیم آنرا با هرم لاپلاسی مقایسه کنیم،از نظر پیچیدگی حافظه ای هرم گاوسی شرایط بهتری دارد.

از نظر پیچیدگی زمانی با توجه به اینکه برای هرم لاپلاسی،یک اعمال فیلتر جعبه داریم اما در هرم موجک چهار با اعمال فیلتر haar داریم میتوان گفت هرم لاپلاسی شرایط بهتری دارد.

6.1.4 هرم موجک و چندی سازی:

به کمک چندی سازی هرم بدست آمده در بخش قبل میتوان به فشرده سازی تصویر کمک نمود.به این صورت که در هر مرحله تابع چندی سازی را بر هر 4 زیر باند اعمال میکنیم.یعنی 3 بخش جزئیات و بخش تقریب همه این تابع بر روی آنها اعمال می شوند.تابعی که اعمال میکنیم به صورت زیر می باشد.

$$c'(u,v) = \gamma \times sgn[c(u,v)] \times \left\lfloor \frac{|c(u,v)|}{\gamma} \right\rfloor$$

6.2.1 حذف نویز به کمک تبدیل موجک:

همانطور که گفته شد،بخش جزئیات در تبدیل موجک یک ماتریس sparse میباشد. معنی آن این است که اکثر مقادیر آن یا صفر بوده و یا بسیار نزدیک به صفر می باشند.در صورت وجود نویز اگر مقادیر خیلی کوچک را به کمک threshold صفر در نظر بگیریم،میتوان به حذف نویز کمک کرد.روش های مختلفی برای تعیین و اعمال threshold وجود دارد. ما از soft threshold استفاده میکنیم. فرمول آن برای سیگنال یک بعدی به صورت زیر می باشد که میتوان آنرا به فرم دو بعدی بسط داد:

$$y_{soft}(t)=\begin{cases}\operatorname{sgn}(x(t))\cdot(|x(t)-\delta|), & |x(t)|>\delta\\ 0, & |x(t)|<\delta\end{cases}$$

برای بدست آوردن مقدار T از فرمول های زیر استفاده میکنیم:

$$T=\frac{\beta\hat{\sigma}^2}{\hat{\sigma}_y}$$

که β برای هر رزولوشن جداگانه محاسبه میشود:

$$\beta=\sqrt{\log\left(\frac{L_k}{J}\right)}$$

که J تعداد تجزیه هایمان و L_k سایز تصویر در رزولوشن فعلی میباشد.

برای بدست آوردن دو پارامتر دیگر T از فرمول های زیر استفاده میکنیم:

$$\hat{\sigma}^2=\left[\frac{median(|Y_{ij}|)}{0.6745}\right]^2,\qquad Y_{ij}\in \text{ subband } HH_1$$

یعنی انحراف معیار در بخش HH

$\hat{\sigma}_y$ نیز انحراف معیار در هر زیر باند می باشد.

3-شرح نتایج

(ممکن است برای بهتر دیدن جزئیات نیاز باشد زوم کنید)

تصویر تست این تمرین،تصویر تصویر لنا است:

6.1.1 هرم لاپلاسی:

ابتدا تصویر هرم تا 9 مرحله ($2^9=512$) را ایجاد میکنیم.

تصویر هرم تا 9 مرحله تجزیه

در مرحله آخر صرفا به یک عدد می رسیم.

حال تصویر را به کمک روش شرح داده شده در بخش تکنیکال،بازسازی میکنیم.

تصویر بازسازی شده

تصویر بازسازی شده را با تصویر اصلی مقایسه میکنیم.

Psnr	Inf
Mse	0

نتایج عددی

نتایج نشان دهنده آن است که در تبدیل به کمک هرم لاپلاسی،هیچ داده ای از دست نمیرود.حتی اگر تا آخرین سطح تجزیه را انجام دهیم.

با توجه به اینکه downsampling و upsampling داریم ممکن است این نکته عجیب به نظر برسد،اما دلیل آن این است که جزئیات و داده های مهم را در لاپلاسین ها ذخیره میکنیم.زمانی که فرکانس های بالای تصویر را در لاپلاسین ها ذخیره کردیم و بالاترین فرکانس سیگنال کاهش یافت،قاعده شانون به ما اجازه میدهد نرخ نمونه برداری را بدون از دست دادن داده ای کاهش دهیم.

6.1.2 هرم لاپلاسی:

به مانند مرحله قبل تجزیه را انجام میدهیم.

تصویر هرم تا 3 مرحله تجزیه

حال تصویر را به کمک روش شرح داده شده در بخش تکنیکال،بازسازی میکنیم.

تصویر بازسازی شده

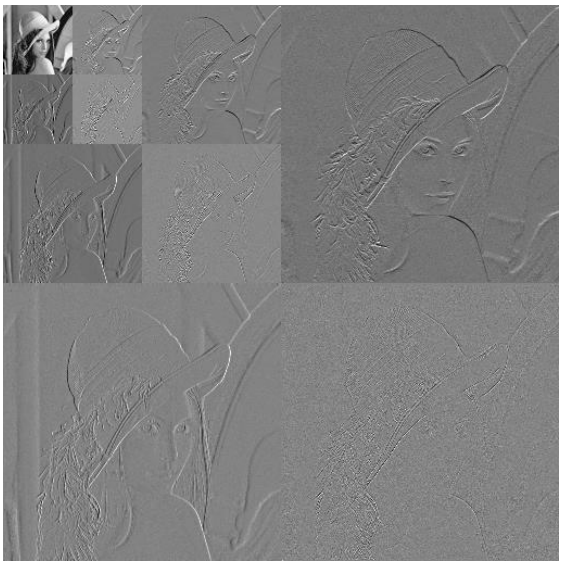
Psnr	Inf
Mse	0

نتایج عددی

همانطور که در بخش قبل دیدیم و در این بخش هم مورد انتظارمان بود که هیچ داده ای از دست نرود، خطایی پیش نیامد.

6.1.3 هرم موجک:

ابتدا هرم موجک را برای سه مرحله ایجاد میکنیم.



هرم بدست آمده برای 3 مرحله

حال تصویر را بازسازی میکنیم.

تصویر بازسازی شده

Psnr	Inf
Mse	0

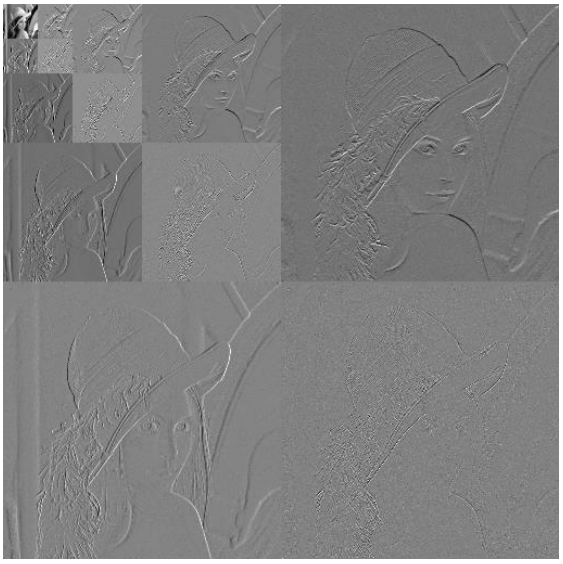
نتایج عددی

نتایج نشان دهنده آن است که در تبدیل به کمک هرم موجک،هیچ داده ای از دست نمیرود.دلیل آن این است که جزئیات مختلف تصویر برای هر رزولوشن به صورت جداگانه در سه بخش ذخیره میشود و در حین بازگشت از آنها برای ساخت تصویر اصلی استفاده میکنیم.

6.1.4 هرم موجک چندی سازی شده:

ابتدا هرم موجک را برای سه مرحله ایجاد میکنیم.

با توجه به اینکه تابع بازگشتی استفاده میشود،برای هر مرحله، به هر چهار زیر باند تابع چندی سازی را اعمال میکنیم.



هرم بدست آمده برای 3 مرحله

تصویر بازسازی شده.

Psnr	46.4378
Mse	1.4767

نتایج عددی

نتایج نشان دهنده آن است که در تبدیل به کمک هرم موجک و چندی سازی مقدار بسیار کمی از دست دادن داده داریم.با توجه به اینکه چندی سازی داریم،این مقدار خطا قابل چشم پوشی می باشد.این تابع چندی سازی در فشرده سازی Jpeg استفاده شده است.

6.2.1 **حذف نویز به کمک تبدیل موجک:**

ابتدا به کمک روش thresholding سعی میکنیم نویز را از بین ببریم.

ابتدا به تصویر لنا نویز گاوسین با پارامتر 0.01 اضافه میکنیم.

تصویر با نویز گاوسین

میزان تفاوت با تصویر اصلی را بدست می آوریم.

Psnr	20.0197
Mse	647.3018

نتایج عددی

حال بر روی آن soft threshold را اعمال میکنیم.

تصویر بعد از حذف نویز

میزان تفاوت با تصویر اصلی را بدست می آوریم.

Psnr	23.7350
Mse	275.1544

نتایج عددی

نویز کاهش داشته است.

به دلیل کمبود وقت و مشکل کد باقی نتایج تاثیر مثبتی نداشتند.

در نتیجه الکی وقت شما را در این بخش نمی گیرم :)

4-کدها

6.1.1 هرم لاپلاسی:

```
img =
imread('Homeworks/Images/6
/Lena.bmp');
img = rgb2gray(img);

levels = 9;
[pyramid,laplacians] =
laplace_pyramid(img,levels
);

[n,~] = size(laplacians);

last_img = laplacians{n};
for i=1:levels
    n = n-1;
    last_img =
pixel_rep(last_img);
    last_img = last_img +
laplacians{n};
end
last_img =
uint8(last_img);

p = psnr(img,last_img);
imshow(last_img);
m = immse(last_img,img);
figure
imshow(pyramid);

imwrite(last_img,'x.png');


function
[final,laplacians_last_img
] =
laplace_pyramid(img,levels
)

    avg_filter = [1 1;1
1]/4;
    [R,C]  = size(img);
    final =
zeros(2*R,2*C,'uint8');
    old_img = double(img);

    N = levels;
    laplacians_last_img =
cell(N,1);
    for i=1:levels

        r1 = R-(0.5)^(i-
1)*R+1;
        r2 = R;

        c1 = ((2^(i-1)-
1)/(2^(i-2)))*C+1;
        c2 = (2^(i)-
1)/(2^(i-1))*C;

        final(r1:r2,c1:c2)
= old_img;

        new_img =
average_filter(old_img,avg
_filter);


        laplace = old_img
- pixel_rep(new_img);


laplacians_last_img{i} =
laplace;

        laplace =
norm(laplace);
        laplace =
uint8(laplace);

        r1 =R+1;
        r2 = R+(0.5)^(i-
1)*R;


final(r1:r2,c1:c2)=laplace
;
        old_img = new_img;
    end

    i = levels+1;
```

```
    output =
zeros(R/2,C/2,'double');
    for i=1:2:R
        for j=1:2:C
            part =
double(image(i:i+1,j:j+1))
;
            mult =
part.*filter;
            out =
sum(mult,'all');

output(ceil(i/2),ceil(j/2)
) = out;
        end
    end
end

function output =
norm(img)
    output =
mat2gray(img);
    Max =
max(max(output));
    Min =
min(min(output));
    output = (255/(Max-
Min))*output;

end
```

6.1.2 هرم لاپلاسی:

```
img =
imread('Homeworks/Images/6
/Lena.bmp');
img = rgb2gray(img);

levels = 3;
    r1 = R-(0.5)^(i-
1)*R+1;
    r2 = R;
    c1 = ((2^(i-1)-
1)/(2^(i-2)))*C+1;
    c2 = (2^(i)-1)/(2^(i-
1))*C;
    final(r1:r2,c1:c2) =
old_img;
    r1 =R+1;
    r2 = R+(0.5)^(i-1)*R;
    final(r1:r2,c1:c2) =
old_img;


laplacians_last_img{levels
+1}=old_img;
end
function output =
pixel_rep(img)
    [r,c] = size(img);
    output =
zeros(2*r,2*c,class(img));
%// Change
    for x = 1:r %// Change
        for y = 1:c
            j = 2*(x-1) +
1; %// Change
            i = 2*(y-1) +
1; %// Change
            output(j,i) =
img(x,y); %// Top-left
            output(j+1,i)
= img(x,y); %// Bottom-
left
            output(j,i+1)
= img(x,y); %// Top-right

output(j+1,i+1) =
img(x,y); %// Bottom-right
        end
    end
end
function
output=average_filter(imag
e,filter)
    [R,C] = size(image);
```

```
        c1 = ((2^(i-1)-
1)/(2^(i-2)))*C+1;
        c2 = (2^(i)-
1)/(2^(i-1))*C;

        final(r1:r2,c1:c2)
= old_img;

        new_img =
average_filter(old_img,avg
_filter);


        laplace = old_img
- pixel_rep(new_img);


laplacians_last_img{i} =
laplace;

        laplace =
norm(laplace);
        laplace =
uint8(laplace);

        r1 =R+1;
        r2 = R+(0.5)^(i-
1)*R;


final(r1:r2,c1:c2)=laplace
;
        old_img = new_img;
    end

    i = levels+1;
    r1 = R-(0.5)^(i-
1)*R+1;
    r2 = R;
    c1 = ((2^(i-1)-
1)/(2^(i-2)))*C+1;
    c2 = (2^(i)-1)/(2^(i-
1))*C;
    final(r1:r2,c1:c2) =
old_img;
    r1 =R+1;
    r2 = R+(0.5)^(i-1)*R;
```

```
[pyramid,laplacians] =
laplace_pyramid(img,levels
);

[n,~] = size(laplacians);

last_img = laplacians{n};
for i=1:levels
    n = n-1;
    last_img =
pixel_rep(last_img);
    last_img = last_img +
laplacians{n};
end
last_img =
uint8(last_img);

p = psnr(img,last_img);
imshow(last_img);
m = immse(last_img,img);
figure
imshow(pyramid);


imwrite(pyramid,'x.png');


function
[final,laplacians_last_img
] =
laplace_pyramid(img,levels
)

    avg_filter = [1 1;1
1]/4;
    [R,C]  = size(img);
    final =
zeros(2*R,2*C,'uint8');
    old_img = double(img);

    N = levels;
    laplacians_last_img =
cell(N,1);
    for i=1:levels

        r1 = R-(0.5)^(i-
1)*R+1;
        r2 = R;
```

```
output(ceil(i/2),ceil(j/2)
) = out;
            end
        end
end

function output =
norm(img)
    output =
mat2gray(img);
    Max =
max(max(output));
    Min =
min(min(output));
    output = (255/(Max-
Min))*output;

end
```

6.1.3 هرم موجک:

```
img =
imread('Homeworks/Images/6
/Lena.bmp');
img = rgb2gray(img);


level = 3;
output = wt(img,level,0);

x = iwt(output,level);
%x = norm (x);
x = uint8(x);
imshow(output);
figure
imshow(x);
m = immse(img,x);
p = psnr(img,x);


imwrite(x,'x.png');
function output =
iwt(wt_pyramid,level)
    if(level==0)
        output =
wt_pyramid;
    final(r1:r2,c1:c2) =
old_img;


laplacians_last_img{levels
+1}=old_img;
end
function output =
pixel_rep(img)
    [r,c] = size(img);
    output =
zeros(2*r,2*c,class(img));
%// Change
    for x = 1:r %// Change
        for y = 1:c
            j = 2*(x-1) +
1; %// Change
            i = 2*(y-1) +
1; %// Change
            output(j,i) =
img(x,y); %// Top-left
            output(j+1,i)
= img(x,y); %// Bottom-
left
            output(j,i+1)
= img(x,y); %// Top-right

output(j+1,i+1) =
img(x,y); %// Bottom-right
        end
    end
end
function
output=average_filter(imag
e,filter)
    [R,C] = size(image);
    output =
zeros(R/2,C/2,'double');
    for i=1:2:R
        for j=1:2:C
            part =
double(image(i:i+1,j:j+1))
;
            mult =
part.*filter;
            out =
sum(mult,'all');
```

```
        cH = norm(cH);
        cD = norm(cD);
    end
    output(1:R/2,1:C/2) =
wt(cA,level-1,present);

output(R/2+1:R,1:C/2)=cV;

output(1:R/2,C/2+1:C)=cH;

output(R/2+1:R,C/2+1:C)=cD
;

    if (present)
        output =
uint8(output);
    end

end

function output =
norm(img)
    output =
mat2gray(img);
    Max =
max(max(output));
    Min =
min(min(output));
    output = (255/(Max-
Min))*output;

end
```

6.1.4 هرم موجک چندی سازی شده:

```
img =
imread('Homeworks/Images/6
/Lena.bmp');
img = rgb2gray(img);


level = 4;
output = wt(img,level,0);

x = iwt(output,level);
%x = norm (x);
x = uint8(x);
imshow(output);
figure
        return
    end
    output = wt_pyramid;
    [R,C] = size(output);
    dec_part =
output(1:R/(2^(level-
1)),1:C/(2^(level-1)));


    [r,c] =
size(dec_part);
    cA =
dec_part(1:r/2,1:c/2);
    cV =
dec_part(r/2+1:r,1:c/2);
    cH =
dec_part(1:r/2,c/2+1:c);
    cD =
dec_part(r/2+1:r,c/2+1:c);

     output(1:R/(2^(level-
1)),1:C/(2^(level-1))) =
idwt2(cA,cH,cV,cD,'haar');

    output =
iwt(output,level-1);

end

function output =
wt(img,level,present)

    if(level==0)
        output = img;
        return
    end
    [R,C] = size(img);
    output=
zeros(R,C,'double');

    %[LoD,HiD] =
wfilters('haar','d');
    [cA,cH,cV,cD] =
dwt2(img,'haar');

    if(present)
        cA = norm(cA);
        cV = norm(cV);
```

```
     output(1:R/(2^(level-
1)),1:C/(2^(level-1))) =
idwt2(cA,cH,cV,cD,'haar');

    output =
iwt(output,level-1);

end

function output =
wt(img,level,present)

    if(level==0)
        output = img;
        return
    end
    [R,C] = size(img);
    output=
zeros(R,C,'double');

    %[LoD,HiD] =
wfilters('haar','d');
    [cA,cH,cV,cD] =
dwt2(img,'haar');

    cA = quantizer(cA,2);
    cV = quantizer(cV,2);
    cH = quantizer(cH,2);
    cD = quantizer(cD,2);

    if(present)
        cA = norm(cA);
        cV = norm(cV);
        cH = norm(cH);
        cD = norm(cD);
    end
    output(1:R/2,1:C/2) =
wt(cA,level-1,present);

output(R/2+1:R,1:C/2)=cV;

output(1:R/2,C/2+1:C)=cH;

output(R/2+1:R,C/2+1:C)=cD
;

    if (present)
```

```
imshow(x);
p = psnr(x,img);
m = immse(x,img);

imwrite(x,'x.png');

function output= quantizer
(img,gamma)
    [M,N] = size(img);
    output =
zeros(M,N,'double');

    for i=1:M
        for j=1:N
            output(i,j) =
gamma*
sign(img(i,j))*floor(
abs(img(i,j))/gamma);
        end
    end
end
function output =
iwt(wt_pyramid,level)
    if(level==0)
        output =
wt_pyramid;
        return
    end
    output = wt_pyramid;
    [R,C] = size(output);
    dec_part =
output(1:R/(2^(level-
1)),1:C/(2^(level-1)));


    [r,c] =
size(dec_part);
    cA =
dec_part(1:r/2,1:c/2);
    cV =
dec_part(r/2+1:r,1:c/2);
    cH =
dec_part(1:r/2,c/2+1:c);
    cD =
dec_part(r/2+1:r,c/2+1:c);
```

```
    local_sigma =
std(double(img));
    T =
(beta*sigma)/local_sigma;
    for i=1:R
        for j=1:C
            x = img(i,j);

            if(abs(x)<T)
                x =0;
            else
                x =
sign(x)*abs(x-T);
            end

            output(i,j)=x;

        end
    end

end

function output =
wt(img,level)

    if(level==0)
        output = img;
        return
    end
    [R,C] = size(img);
    output=
zeros(R,C,'double');
    [cA,cH,cV,cD] =
dwt2(img,'haar');


    beta =
sqrt(log2(R/3));
    sigma =
median(abs(cD)./0.6745).^2
;

    cH =
soft_tresh(cH,beta,sigma);
    cD =
soft_tresh(cD,beta,sigma);
    cV =
soft_tresh(cV,beta,sigma);
        output =
uint8(output);
    end

end
function output =
norm(img)
    output =
mat2gray(img);
    Max =
max(max(output));
    Min =
min(min(output));
    output = (255/(Max-
Min))*output;

end
```

6.2.1 حذف نویز به کمک تبدیل موجک:

```
img =
imread('Homeworks/Images/6
/Lena.bmp');
img = rgb2gray(img);
noisy =
imnoise(img,'gaussian',0.0
1);

level =3;
output = wt(noisy,level);
x = iwt(output,level);


x = uint8(x);

imshow(noisy);
figure
imshow(x);
p1 = psnr(noisy,img);
m = immse(img,x);
p2 = psnr(img,x);

imwrite(noisy,'x.png');
function output=
soft_tresh(img,beta,sigma)
    [R,C] = size(img);
    output = zeros(R,C);
```

```
function output =
norm(img)
    output =
mat2gray(img);
    Max =
max(max(output));
    Min =
min(min(output));
    output = (255/(Max-
Min))*output;

end
```

```
    output(1:R/2,1:C/2) =
wt(cA,level-1);

output(R/2+1:R,1:C/2)=cV;

output(1:R/2,C/2+1:C)=cH;

output(R/2+1:R,C/2+1:C)=cD
;

end

function output =
iwt(wt_pyramid,level)
    if(level==0)
        output =
wt_pyramid;
        return
    end
    output = wt_pyramid;
    [R,C] = size(output);
    dec_part =
output(1:R/(2^(level-
1)),1:C/(2^(level-1)));

    [r,c] =
size(dec_part);
    cA =
dec_part(1:r/2,1:c/2);
    cV =
dec_part(r/2+1:r,1:c/2);
    cH =
dec_part(1:r/2,c/2+1:c);
    cD =
dec_part(r/2+1:r,c/2+1:c);

    output(1:R/(2^(level-
1)),1:C/(2^(level-1))) =
idwt2(cA,cH,cV,cD,'haar');

    output =
iwt(output,level-1);

end
```