گزارش تمرین Wavelets

مریم واقعی

اطلاعات گزارش

تاریخ:
۱۱ دی ۱۴۰۱

واژگان کلیدی:
هرم موچک
تبدیل موجک گسسته
هرم گوسی
هرم لاپلاسین
موجک
فیلتر هار

چکیده

معایب تبدیل فوریه و تبدیل فوریه زمان کوتاه، منجر به استفاده از تبدیل موک شد. در این تبدیل می‌توان آنالیز محلی سیگنال را برای یافتن مولفه های فرکانسی به طور پویا در دامنه سیگنال انجام داد. با استفاده از تبدیل موجک گسسته و اعمال آن بر روی تصویر، می‌توان رزولوشن های مختلف از تصویر را در مقیاس های مختلف ذخیره سازی نمود که به آن هرم رزولوشن می‌گویند.

فهرست مطالب

۱-مقدمه ۳

۲- توضیحات فنی ۳

۲-۱- Pyramid ۳

۲-۱-۱- بخش اول ۳

۲-۱-۲- بخش دوم ۳

۲-۱-۳- بخش سوم ۳

۲-۱-۴- بخش چهارم ٤

۲-۱-۵- بخش پنجم ٤

۲-۱-۶- بخش ششم ٤

۳- بررسی نتایج ٥

۳-۱- Pyramid ٥

۳-۱-۱- بخش اول ٥

۳-۱-۲- بخش دوم ٦

۳-۱-۳- بخش سوم ٦

۳-۱-۴- بخش چهارم ٦

۳-۱-۵- بخش پنجم ۷

۳-۱-۶- بخش ششم ۹

۴- پیوست ۱۰

۴-۱- Pyramid ۱۰

۴-۱-۱- بخش اول ۱۰

۴-۱-۲- بخش دوم ۱۱

۴-۱-۳- بخش سوم ۱۱

۴-۱-۴- بخش چهارم ۱۱

۴-۱-۵- بخش پنجم ۱۲

۴-۱-۶- بخش ششم ۱۳

۱-مقدمه

تبدیل فوریه، به بیان ساده مشخص کننده این بود که در کل طول بازه زمانی تا بی نهایت، چه فرکانس هایی در سیگنال وجود دارند. از این رو مزیت عمده این تبدیل، یافتن تمامی مولفه های فرکانسی سیگنال بود.با این حال، مکان یا زمان وقوع مولفه‌های فرکانسی در سیگنال های غیرایستا، یکی از چالش هایی بود که تبدیل فوریه در مشخص کردن آن ناتوان بود.
تلاش برای یافتن بازنمایی هایی که مشخص می‌کنند چه اجزای فرکانسی‌ای در چه مکانی (یا زمانی) اتفاق افتاده، منجر به معرفی تبدیل فوریه زمن کوتاه و پس از آن، تبدیل موجک شد.
در این گزارش به معرفی تبدیل موجک و چندی از کاربردهای آن می‌پردازیم، همچنین به بازنمایی تصویر به صورت چند رزولوشنی خواهیم پرداخت و کاربرد های تبدیل موجک در بازنمایی چندرزولوشنی و کاهش نویز را بررسی خواهیم کرد.

۲- توضیحات فنی

Pyramid -۱-۲

۲-۱-۱- بخش اول

ابتدا یک cell array با نام gussian_pyr تعریف میکنیم که خروجی هرم گوسی برای هر سطح را در آن ذخیره میکنیم.
سطح اول هرم گوسی را تصویر اصلی قرار میدهیم. سپس در یک حلقه با استفاده از خروجی سطح قبلی هرم گوسی و متد impyramid، سطح بعدی هرم گوسی را می سازیم و در آرایه گفته شده ذخیره میکنیم.
پس از ساخت تمام سطوح هرم گوسی نوبت به ساخت هرم لاپلاسین از روی هرم گوسی میرسد.
برای بدست آوردن سطح i ام هرم لاپلاسین (L_i)، لازم است ابتدا سطح i+1 ام هرم گوسی (G_{i+1}) که ابعاد آن نصف ابعاد سطح i ام هرم گوسی (G_i) است را upsample میکنیم. پس از upsample کردن G_{i+1} آن را از G_i کم میکنیم تا L_i بدست آید. به عبارتی برای بدست آوردن L_i از فرمول زیر استفاده میکنیم:
فرمول (۱):

$$L_i = G_i - expand(G_{i+1})$$

۲-۱-۲- بخش دوم

۲-۱-۳- بخش سوم

هرم approximation:
الف) اگر یک تصویر N*N داشته باشیم که $N = 2^J$ باشد، اگر فرض کنیم با فیلتر ۲*۲ میانگین گیر، هرم approximation را بسازیم در این صورت در هر مرحله اندازه تصویر نصف حالت قبل میشود. پس ماکزیمم تعداد دفعاتی که میتوان فیلتر را اعمال کرد برابر با J می باشد و در نتیجه ماکزیمم تعداد سطح approximation برابر با J+1 می باشد.
ب) تعداد کل پیکسل ها در هرم برابر است با:

$$N^2 + (\frac{N}{2})^2 + (\frac{N}{4})^2 + \cdots + \left(\frac{N}{2^J}\right)^2 = N^2 + \left(\frac{1}{4}\right)N^2 + \left(\frac{1}{16}\right)N^2 + \cdots + 1 = \left(\frac{4}{3}\right)N^2$$

ج) تفاوت تعداد پیکسل های هرم با تصویر اصلی: در تصویر اصلی تعداد پیکسل ها برابر با N^2 است، در صورتی که تعداد پیکسل های هرم تقریبا برابر با $\left(\frac{4}{3}\right)N^2$ می باشد که یعنی یک سوم برابر تصویر اصلی، فضای ذخیره سازی بیشتری نیزا دارد.

د) مزایای هرم approximation: میتوانیم در هر مقیاس از تصویر که میخواهیم، پردازش ها را انجام دهیم چون در این هرم اندازه تصویر در هر سطح با نسبت نمایی از اندازه تصویر اصلی کمتر است، پس پردازش ها در سطوح این هرم نسبت به تصویر اصلی سریعتر است.

هرم residual:

الف) مشابه پاسخ الف برای هرم approximation

ب) مشابه پاسخ ب برای هرم approximation

ج) مشابه پاسخ ج برای هرم approximation، تعداد کل پیکسل ها در این هرم برابر با $\left(\frac{4}{3}\right)N^2$ می باشد، اما نکته ای که وجود دارد این است که این هرم تنها لبه های تصویر را ذخیره میکند و تعداد زیادی از پیکسل ها در هر سطح مقدار صفر را دارند و ماتریس تصویر در هر سطح تُنک میباشد. بنابراین فضای ذخیره سازی به شدت کمتری نسبت به تصویر اصلی نیاز دارد.

د) مزایای هرم residual: برای یافتن لبه ها در هر سطح هرم میتوانیم پردازش ها را داشته باشیم. همچنین برای ذخیره سازی تصویر اصلی با حجم بسیار کمتر میتوانیم از این هرم استفاده کنیم چون این هرم تنها لبه ها را حفظ میکند و با ذخیره این لبه ها و تصویر اصلی در مقیاس کوچکتر میتوان تصویر اصلی را بازسازی کرد.

۲-۱-۴- بخش چهارم

در این بخش برای تولید هرم approximation و residual ابتدا متد downsample_by_average_filter() را پیاده میکنیم که از بلاک های ۲*۲ از تصویر اصلی میانگین میگیرد و در ماتریس خروجی در درایه نظیر قرار میدهد. سپس متد pixel_replication() را پیاده میکنیم که هر مقدار از تصویر ورودی داده شده به این متد را داخل بلاک های ۲*۲ از تصویر خروجی تکرار میکند.

حال پس از پیاده سازی این دو متد، در یک حلقه روی خروجی مرحله قبل approximation، متد downsample_by_average_filter() را اعمال میکنیم تا سطح بعدی از هرم approximation بدست آید. پس از بدست آمدن هر سطح approximation آن را با استفاده از متد pixel_replication()، upsample میکنیم و سپس از سطح قبل هرم کم میکنیم تا residual این دو سطح بدست آید.

۲-۱-۵- بخش پنجم

در این بخش از ما خواسته شده تا تبدیل موجک را با استفاده از موجک haar اعمال کنیم. برای اینکار ابتدا فیلتر های تجزیه lowpass و highpass مربوط به موجک haar را با استفاده از متد wfilters تولید میکنیم. سپس با استفاده از این فیلتر ها و متد dwt2() تبدیل موجک haar را روی تصویر approximation هر سطح (برای سطح اول روی تصویر اصلی) اعمال میکنیم و تصویر approximation و لبه های افقی، عمودی و مورب تصویر را در برای هر سطح نمایش میدهیم.

۲-۱-۶- بخش ششم

ابتدا برای کوانتایز کردن مقادیر ضرایب موجک، متد quantization() را تعریف میکنیم. به این صورت که با استفاده از فرمول زیر مقادیر ضرایب را کوانتایز میکند:

فرمول(۲)

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

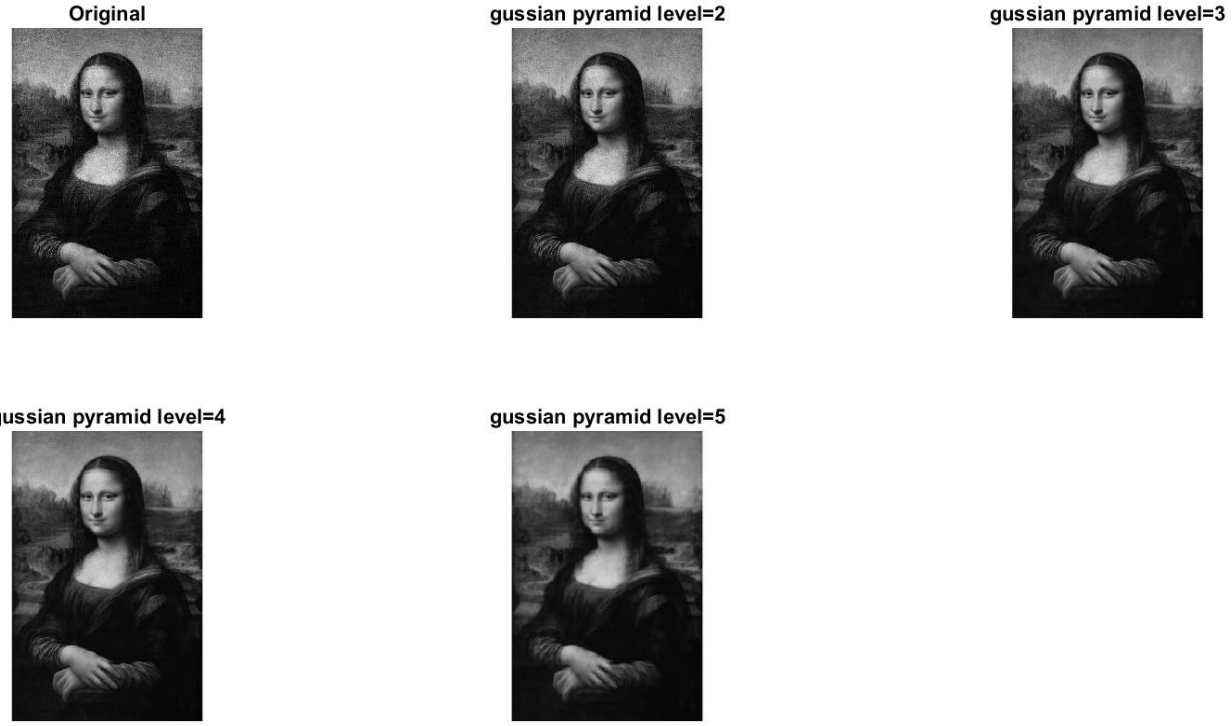
پس از تعریف این متد، داخل یک حلقه ابتدا تبدیل موجک haar را (مشابه بخش پنجم تمرین) روی approximation سطح قبل (که در ابتدا همان تصویر اصلی می باشد) اعمال میکنیم. سپس با استفاده از متد ()quantization گفته شده در بالا، مقادیر ضرایب approximation و horizontal و vertical و diagonal را کوانتایز میکنیم و پس از آن با استفاده از متد idwt2 معکوس تبدیل موجک را اعمال میکنیم تا تصویر بازسازی شود. سپس با استفاده از متد psnr مقدار Peak signal-to-noise ratio را برای تصویر محاسبه میکنیم که این مقادیر در title هر تصویر در خروجی آمده است.

۳- بررسی نتایج

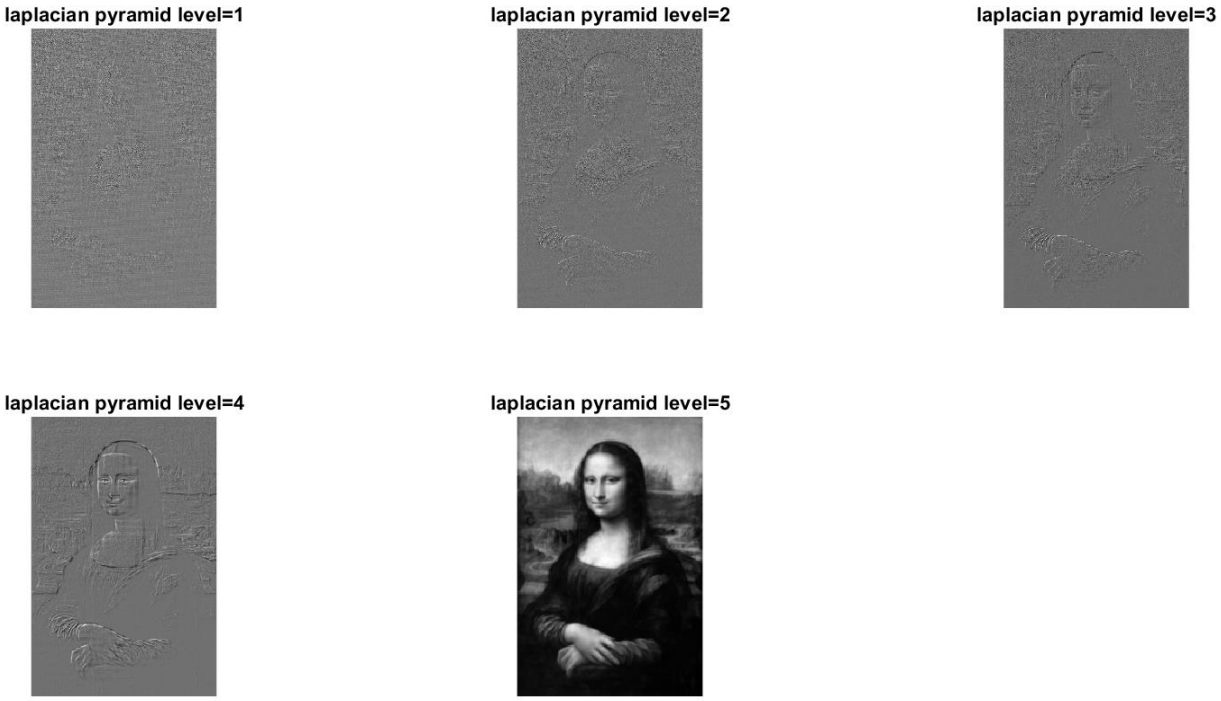
۳-۱- Pyramid

۳-۱-۱- بخش اول

در تصاویر زیر شما خروجی هرم گوسی و هرم لاپلاسین برای ۵ سطح را مشاهده میکنید:



تصویر ۱ - خروجی هرم گوسی برای ۵ سطح که سطح اول آن همان تصویر اصلی می باشد.



تصویر ۲- خروجی هرم لاپلاسین برای ۵ سطح که سطح آخر آن برابر سطح آخر هرم گوسی می باشد.

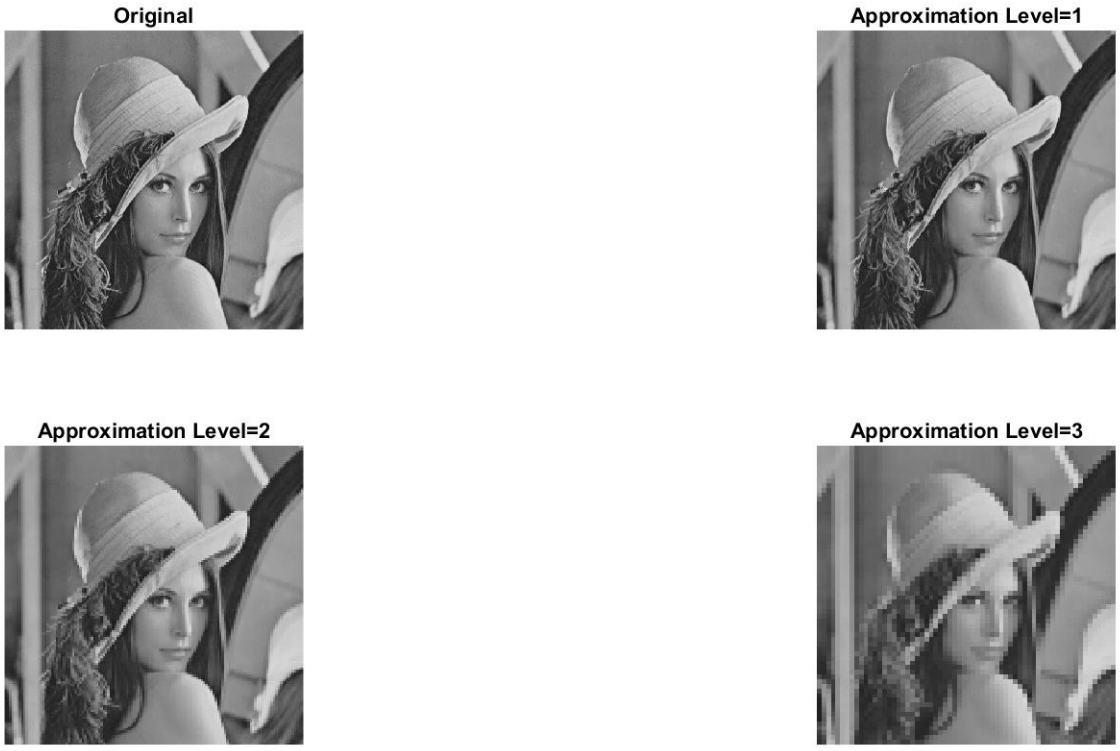
۳-۱-۲- بخش دوم

۳-۱-۳- بخش سوم

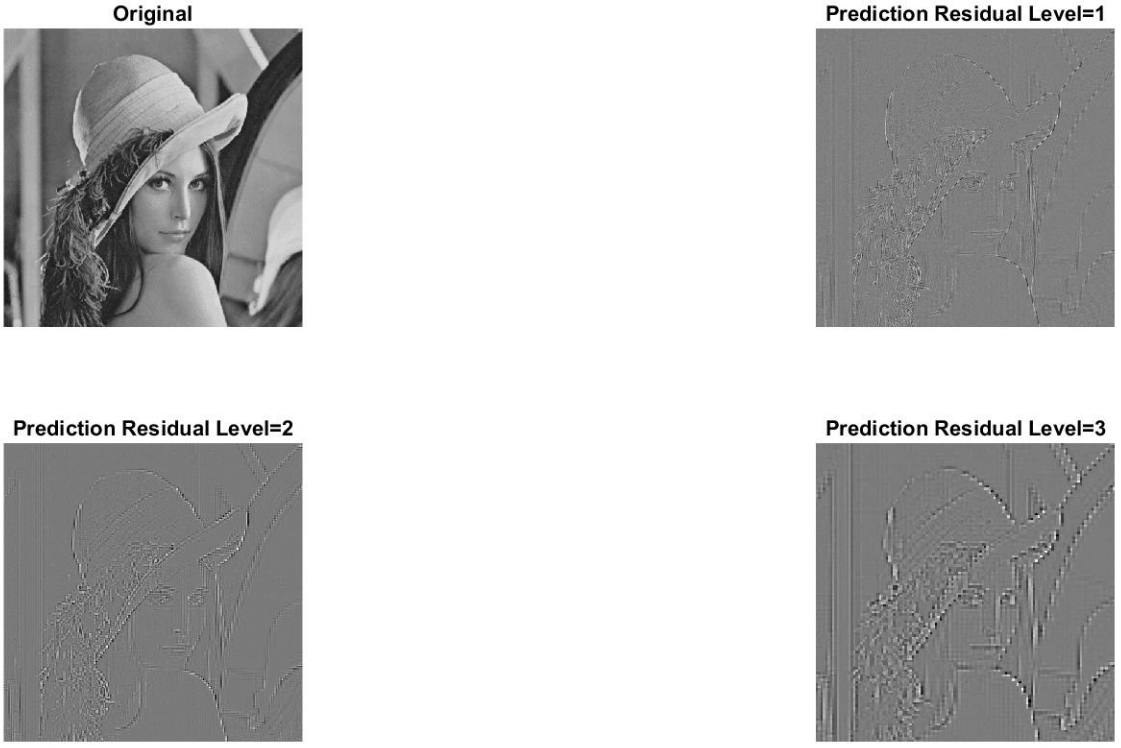
این بخش نتیجه ای برای نمایش ندارد. پاسخ سوال در بخش توضیحات فنی آورده شده است.

۳-۱-۴- بخش چهارم

در تصاویر زیر شما خروجی هرم approximation و residual برای ۳ سطح را مشاهده میکنیم.



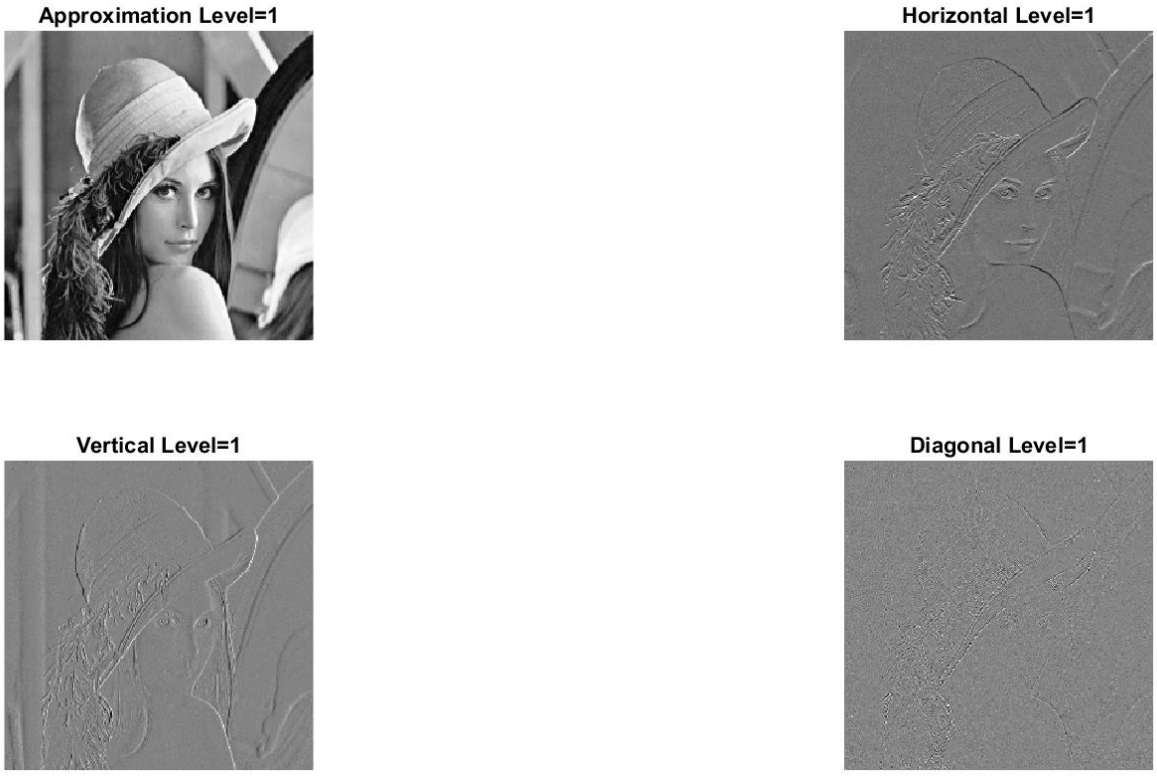
تصویر ۳ - هرم approximation تا سطح سوم (تصویر اصلی برای مقایسه با سطوح هرم آورده شده است)



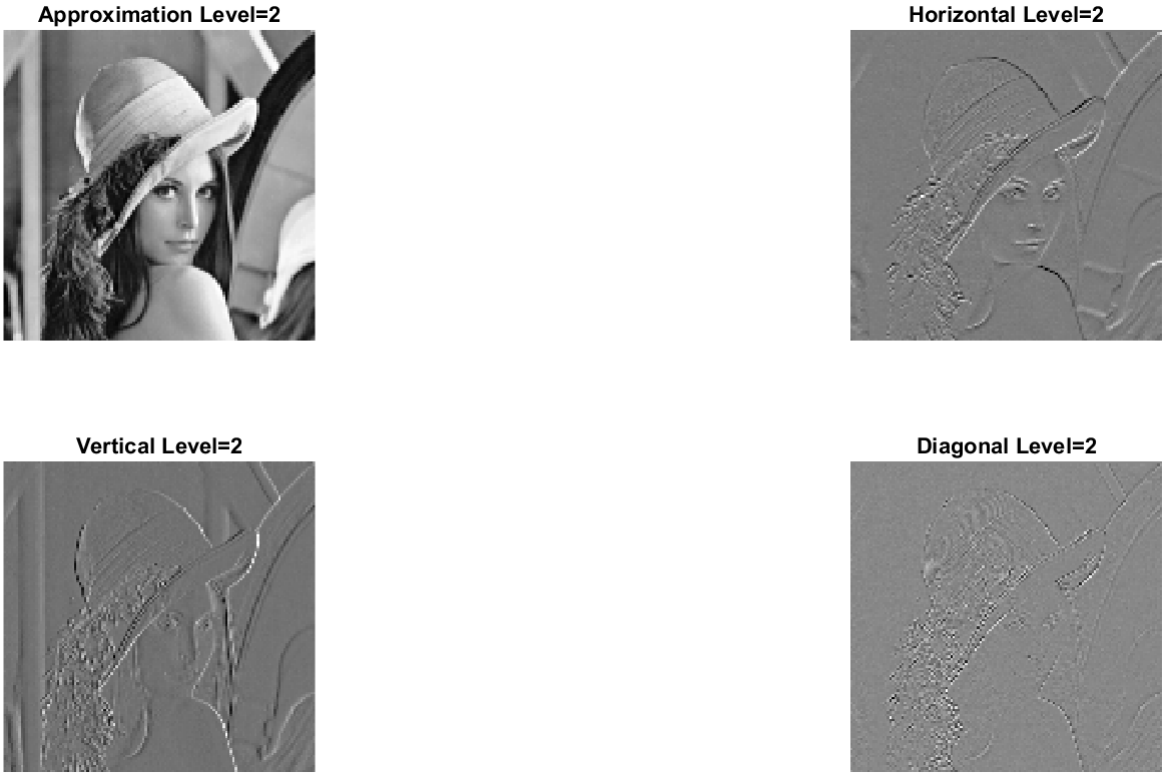
تصویر ۴- هرم residual تا سطح سوم (تصویر اصلی برای مقایسه با سطوح هرم آورده شده است)

۳-۱-۵- بخش پنجم

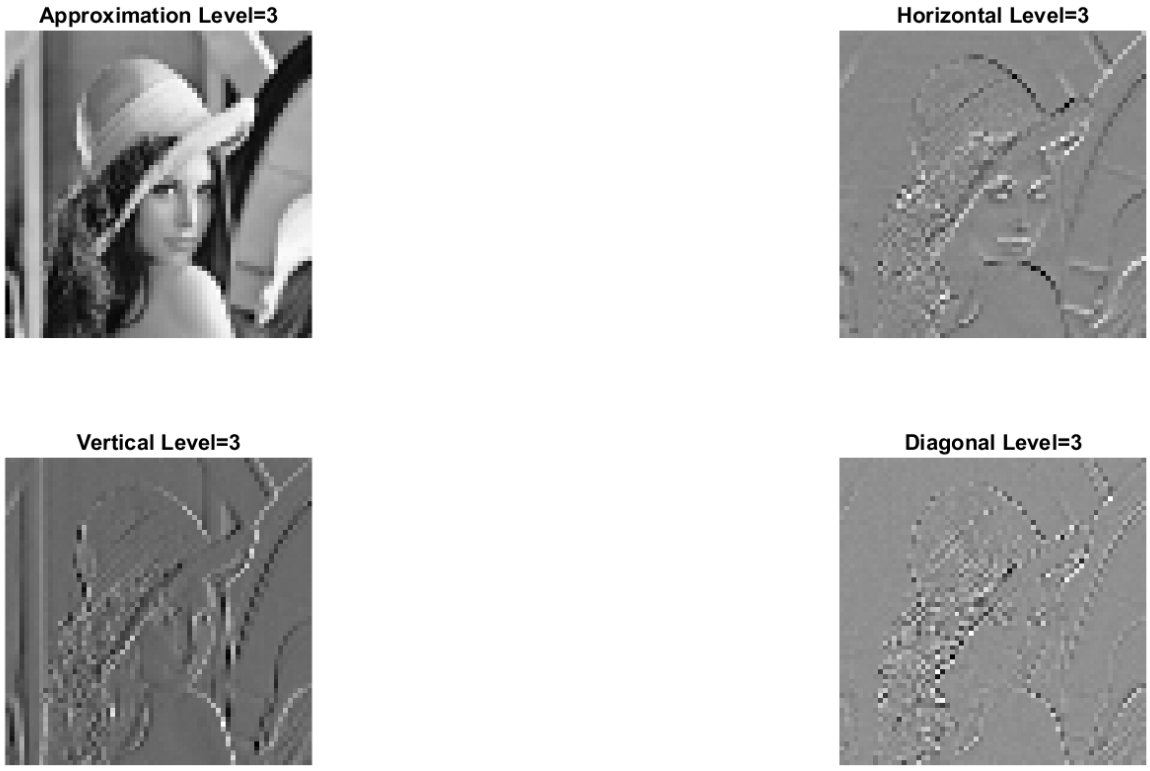
در تصاویر زیر شما خروجی تبدیل موجک haar را برای ۳ سطح مشاهده میکنید:



تصویر ۵- سطح اول تبدیل موجک **haar** روی تصویر **lena**



تصویر ۶- سطح دوم تبدیل موجک **haar** روی تصویر **lena**



تصویر ۷- سطح سوم تبدیل موجک haar روی تصویر lena

مقایسه سطوح approximation سوال بخش قبل با این بخش:

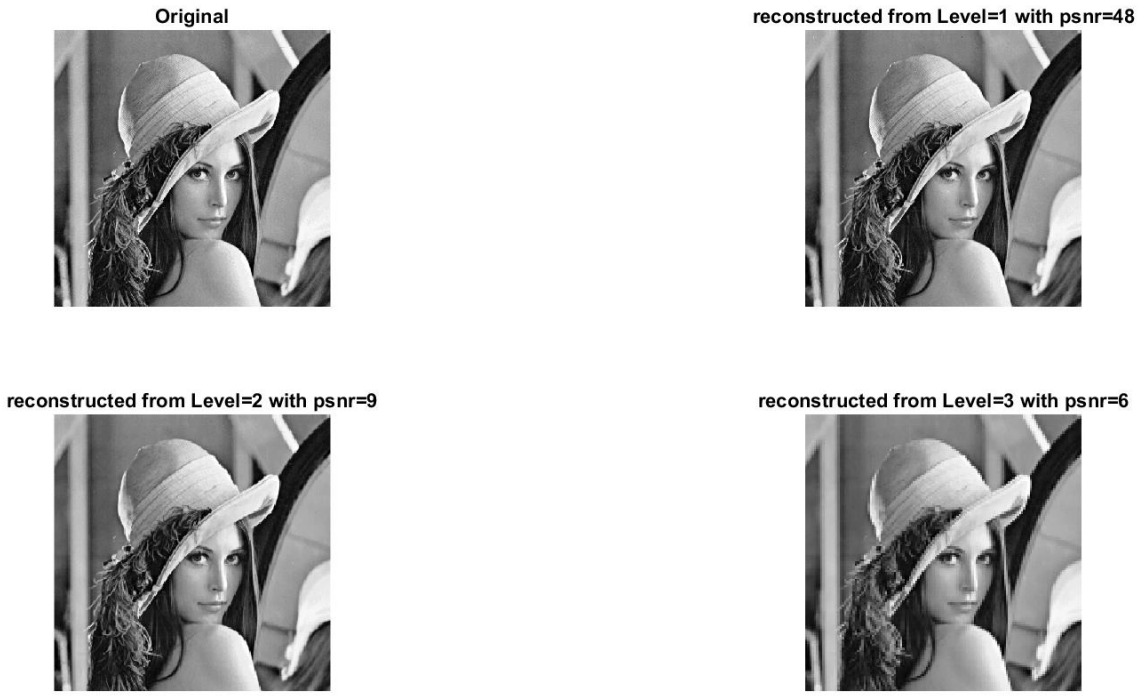
همانطور که مشاهده میکنید سطح approximation در سوال قبل نسبت به سطح approximation نظیر آن در تبدیل موجک، smooth تر می باشد زیرا از میانگین گیری برای بدست آوردن آن استفاده شده است و approximation مربوط به تبدیل موجک sharp تر از آن است.

مقایسه سطوح residual سوال بخش قبل با این بخش:

در سوال قبل در سطوح هرم residual لبه های عمودی تصویر را مشاهده میکنید. در صورتی که در residual مربوط به تبدیل موجک هر ۳ نوع لبه افقی، عمودی و مورب به ازای هر سطح هرم موجک ذخیره شده است.

۳-۱-۶- بخش ششم

در تصاویر زیر شما خروجی بازسازی شده تصویر کوانتایز شده در سطوح مختلف را مشاهده میکنید:



تصویر ۸- تصاویر بازسازی شده از هر سطح تبدیل موجک که مقادیر ضرایب موجک آنها کوانتایز شده است.

برای مقادیر psnr تصویر اصلی lena به عنوان مرجع درنظر گرفته شده و محاسبه شده است. همانطور که مشاهده میکنید هرچقدر به سطح بالاتری از موجک میرویم مقدار psnr کاهش میابد که نشان میدهد کیفیت تصویر بازسازی شده کمتر میشود و تصویر بازسازی شده با تصویر اصلی اختلاف بیشتری دارد.

۴- پیوست

۴-۱- Pyramid

۴-۱-۱- بخش اول

کد main:

```
mona_lisa = rgb2gray(imread('D:\Dars\Masters\digital image
processing\Homeworks\Images\5\mona lisa.jpg'));
level = 5;
%GUSSIAN PYRAMID CREATION
gussian_pyr = cell(1,level);
gussian_pyr{1} = mona_lisa;
figure;
subplot(2,3,1);imshow(gussian_pyr{1});title('Original');
for i=2:level
    gussian_pyr{i} = impyramid(gussian_pyr{i-1},'reduce');
    subplot(2,3,i);imshow(gussian_pyr{i}); title(strcat('gussian pyramid
level=',int2str(i)));
end

laplacian_pyr = cell(1,level);
% LAPLACIAN PYRAMID CREATION
current_img = gussian_pyr{1};% cur_img = Gi
figure;
for i=1:level-1
```

```
    expanded = pixel_replication(gussian_pyr{i+1});%expand(Gi+1)
    if size(current_img,1) ~= size(expanded,1)
        expanded = expanded(1:size(expanded,1)-(size(expanded,1)-
size(current_img,1)),:);
    end
    if size(current_img,2) ~= size(expanded,2)
        expanded = expanded(:,1:size(expanded,2)-(size(expanded,2)-
size(current_img,2)));
    end
    laplacian_pyr{i} = double(current_img) - double(expanded); % Li = Gi -
expanded (Gi+1)
    current_img = gussian_pyr{i+1};
    subplot(2,3,i);imshow(double(laplacian_pyr{i}),[]); title(strcat('laplacian
pyramid level=',int2str(i)));
end
laplacian_pyr{level} = gussian_pyr{level};
subplot(2,3,level);imshow(double(laplacian_pyr{level}),[]);
title(strcat('laplacian pyramid level=',int2str(level)));
```

کد متد pixel_replication():

```
function output = pixel_replication(img)
[X,Y] = size(img);
%make sure output is 2* the size of the input
output = zeros(2*X,2*Y);
for x = 1:X
    for y = 1:Y
        %first we find the top-left, and then find others in respect to it
        j = 2*(x-1) + 1;
        i = 2*(y-1) + 1;
        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
output = uint8(output);
```

۴-۱-۲- بخش دوم

۴-۱-۳- بخش سوم

این بخش کدی برای نمایش ندارد. پاسخ سوال در بخش توضیحات فنی آورده شده است.

۴-۱-۴- بخش چهارم

کد main:

```
lena = rgb2gray(imread('D:\Dars\Masters\digital image
processing\\Homeworks\Images\5\Lena.bmp'));
temp = lena;
fig2 = figure;
figure(1); subplot(2,2,1); imshow(lena); title('Original');
figure(2); subplot(2,2,1); imshow(lena); title('Original');
```

```
for i=1:3
    approx_img = downsample_by_average_filter(temp);
    prediction_residual = double(temp) - pixel_replication(approx_img);
    temp = approx_img;
    figure(1); subplot(2,2,i+1); imshow(uint8(temp)); title(strcat('Approximation
Level=',int2str(i)));
    figure(2); subplot(2,2,i+1); imshow(prediction_residual,[]);
title(strcat('Prediction Residual Level=',int2str(i)));
end
```

کد متد downsample_by_average_filter():

```
function dst = downsample_by_average_filter(img)
[height, width] = size(img);
dst_height = floor(height/2);
dst_width = floor(width/2);
dst = zeros(dst_height,dst_width);
for i=1:dst_height
    for j=1:dst_width
        block = img(2*i-1: 2*i, 2*j-1: 2*j);
        dst(i, j) = mean(block,'all');
    end
end
```

کد متد pixel_replication():

```
function output = pixel_replication(img)
[height,width] = size(img);
%make sure output is 2* the size of the input
output = zeros(2*height,2*width);
for x = 1:height
    for y = 1:width
        %first we find the top-left, and then find others in respect to it
        j = 2*(x-1) + 1;
        i = 2*(y-1) + 1;
        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
```

۴-۱-۵- بخش پنجم

```
lena = rgb2gray(imread('D:\Dars\Masters\digital image
processing\Homeworks\Images\5\Lena.bmp'));
cA = lena;
[LoD, HiD] = wfilters('haar','d');
for i=1:3
    [cA,cH,cV,cD] = dwt2(cA, LoD, HiD, 'mode', 'symh');
    figure;
    subplot(2,2,1); imshow(cA,[]); title(strcat('Approximation
Level=',int2str(i)));
    subplot(2,2,2); imshow(cH,[]); title(strcat('Horizontal Level=',int2str(i)));
    subplot(2,2,3); imshow(cV,[]); title(strcat('Vertical Level=',int2str(i)));
    subplot(2,2,4); imshow(cD,[]); title(strcat('Diagonal Level=',int2str(i)));
end
```

۴-۱-۶- بخش ششم

کد main:

```
lena = rgb2gray(imread('D:\Dars\Masters\digital image
processing\Homeworks\Images\5\Lena.bmp'));
cA = lena;
[LoR, HiR] = wfilters('haar','r');
figure;
subplot(2,2,1); imshow(lena,[]); title('Original');
gamma = 2;
for i=1:3
    % compute dwt of approximation image of previous level
    [cA,cH,cV,cD] = dwt2(cA, LoD, HiD, 'mode', 'symh');
    %now quantize values of coeficients
    quantized_cA = quantization(cA, gamma);
    quantized_cH = quantization(cH, gamma);
    quantized_cV = quantization(cV, gamma);
    quantized_cD = quantization(cD, gamma);
    reconstructed = idwt2(quantized_cA, quantized_cH, quantized_cV, quantized_cD,
LoR, HiR);
    reconstructed_Resized = imresize(reconstructed, size(lena), 'nearest');
    peak_snr = psnr(uint8(reconstructed_Resized),uint8(lena));
    subplot(2,2,i+1); imshow(reconstructed,[]); title(strcat('reconstructed from
Level=',int2str(i),' with psnr=',int2str(peak_snr)));
end
```

کد متد quantization():

```
function quantized_coefs = quantization(coef, gamma)
[height ,width] = size(coef);
quantized_coefs = zeros(height,width);
for u=1:height
    for v=1:width
        quantized_coefs(u,v) = gamma * sign(coef(u,v)) *
floor(abs(coef(u,v))/gamma);
    end
end
```