Wavelet

نسیم فانی

اطلاعات گزارش

تاریخ: 99/10/17

واژگان کلیدی:
موجک
فوریه
هرم لاپلاسین
هرم گوسین
هرم موجک
موجک هار
حذف نویز

چکیده

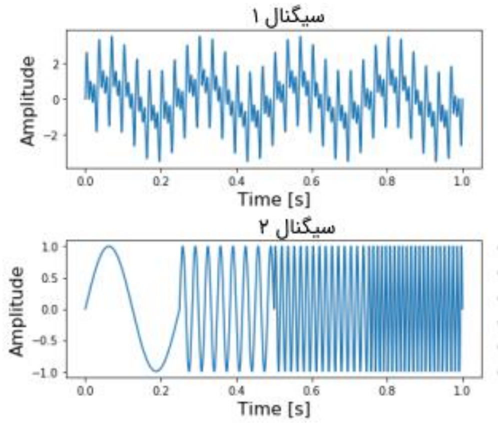
تبدیل موجک (Wavelet Transform) یکی از تبدیلات مهم ریاضی است که در حوزه‌های مختلف علوم کاربرد دارد. ایده اصلی تبدیل موجک این است که بر ضعف‌ها و محدودیت‌های موجود در تبدیل فوریه غلبه کند. این تبدیل را بر خلاف تبدیل فوریه، می‌توان در مورد سیگنال‌های غیر ایستا و سیستم‌های دینامیک نیز مورد استفاده قرار داد.

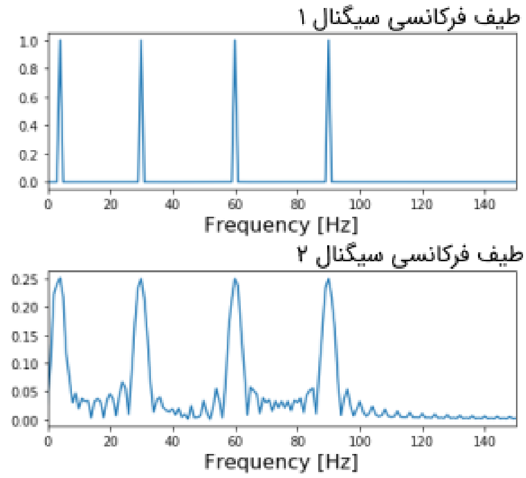
1-مقدمه

از تبدیل فوریه به تبدیل موجک

همان طور که می‌دانیم، تبدیل فوریه از طریق ضرب کردن سیگنال مورد پردازش در قطاری از سیگنال‌های سینوسی با فرکانس‌های مختلف عمل می‌کند. در واقع، از این راه می‌توانیم تعیین کنیم که کدام فرکانس‌ها در سیگنال مورد پردازش وجود دارند. اگر عملگر ضرب نقطه‌ای بین سیگنال مورد نظر و یک سیگنال سینوسی با فرکانس مشخص، برابر با یک عدد با دامنه بزرگ شود، آن‌گاه می‌توان نتیجه گرفت که هم‌پوشانی زیادی بین این دو سیگنال وجود دارد و در نتیجه آن فرکانس مشخص در طیف فرکانسی سیگنال مورد نظر نیز مشاهده خواهد شد. قطعا دلیل این امر از آنجایی ناشی می‌شود که عملگر ضرب نقطه‌ای معیاری برای اندازه‌گیری میزان هم‌پوشانی و شباهت بین دو بردار یا دو سیگنال است.

نکته‌ای که در مورد تبدیل فوریه می‌توان به آن اشاره کرد این است که در حوزه فرکانس دارای رزولوشن بالایی است، در حالی که در حوزه زمان از رزولوشن صفر برخوردار است. به عبارت دیگر، تبدیل فوریه این توانایی را دارد که به ما بگوید دقیقا چه فرکانس‌هایی در یک سیگنال وجود دارند، اما نمی‌توان با استفاده از آن تعیین کرد که فرکانس مورد نظر در چه لحظه‌ای از زمان در سیگنال اتفاق می‌افتد.





در تصاویر بالا، در ردیف اول سیگنال اصلی به همراه طیف فرکانسی آن دیده می‌شود که شامل چهار فرکانس در تمام زمان‌ها است و در ردیف پایین می‌توان دید که سیگنال دوم نیز چهار فرکانس مختلف اما در چهار زمان متفاوت دارد.

در این تصویر، اولین سیگنال از سمت چپ، بر اساس تصویر طیف فرکانسی خود، چهار فرکانس مختلف در ۶۰، ۳۰، ۴ و ۹۰ هرتز دارد. این فرکانس‌ها در تمام زمان‌ها وجود دارند. در سیگنال دوم نیز چهار فرکانس مشابه وجود دارد، اما تفاوتی که دارد این است که فرکانس اول فقط در ربع اول از سیگنال، فرکانس دوم در ربع دوم سیگنال، فرکانس سوم در ربع سوم و فرکانس چهام فقط در ربع آخر سیگنال وجود دارند.

نکته مهمی که در اینجا باید به آن توجه شود این است که هر دو نمودار طیف فرکانسی در تصویر بالا، دارای چهار پیک دقیقا یکسان در ۴ فرکانس ذکر شده هستند. بنابراین با استفاده از تبدیل فوریه نمی‌توان به این واقعیت پی برد که فرکانس در سیگنال اصلی دقیقا در کجا اتفاق می‌افتد. دقیقا به همین دلیل است که در مثال بالا نیز تبدیل فوریه نتوانست به ما در تمایز و تشخیص دو سیگنال از یکدیگر کمک کند. توجه کنید که وقوع لبه‌های گذرا جانبی در طیف فرکانسی سیگنال دوم، به دلیل ناپیوستگی بین چهار فرکانس در سیگنال است.

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

- سیگنال‌های 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
- این در حالی است که 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

به عبارتی دیگر:

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

برای غلبه بر این مشکلات، روش تبدیل فوریه زمان کوتاه (Short-Time Fourier Transform) یا STFT مورد استفاده قرار می‌گیرد. در این روش، سیگنال اصلی به چندین بخش با طول یکسان تقسیم می‌شوند. این بخش‌ها ممکن است با یکدیگر هم‌پوشانی داشته باشند و یا فاقد هم‌پوشانی باشند. با استفاده از پنجره‌های لغزشی (Sliding Window)، سیگنال را قبل از اعمال تبدیل فوریه به چندین بخش تقسیم می‌کنیم. ایده اصلی در این روش بسیار ساده است. اگر سیگنال را مثلا به 10 قسمت تقسیم کرده باشیم و تبدیل فوریه در بخش دوم فرکانس خاصی را تشخیص دهد، بنابراین با قطعیت بالا می‌توان گفت که این فرکانس در بازه بین 2/10 و 3/10 از سیگنال اصلی به وقوع می‌پیوندد.

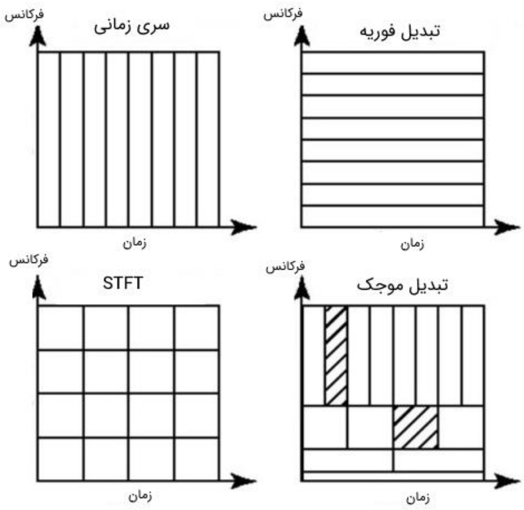
اما عیب اصلی که در این روش وجود دارد این است که با یک محدودیت فیزیکی در تبدیل فوریه رو به رو خواهد شد که عدم قطعیت نام دارد:

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

در این روش، هرچه اندازه پنجره‌ها را کوچک‌تر کنیم، قادر خواهیم بود به صورت دقیق‌تر تعیین کنیم که یک فرکانس در چه زمانی از سیگنال اصلی به وقوع پیوسته است، اما از طرف دیگر اطلاعات کم‌تری را راجع به مقدار فرکانس سیگنال اصلی به دست خواهیم آورد. به صورت مشابه، هر چه اندازه پنجره‌ها را بزرگ‌تر انتخاب کنیم، اطلاعات بیشتری راجع به مقدار فرکانس و اطلاعات کمتری راجع به زمان وقوع فرکانس به دست خواهیم آورد.

روش بهتری که برای آنالیز یک سیگنال با طیف فرکانسی دینامیک وجود دارد، استفاده از تبدیل موجک است. تبدیل موجک هم در حوزه زمان و هم در حوزه فرکانس دارای رزولوشن بالایی است. این تبدیل نه تنها مقدار فرکانس‌های موجود در سیگنال را مشخص می‌کند، بلکه تعیین می‌کند که آن فرکانس‌ها در چه زمانی از سیگنال به وقوع می‌پیوندند. تبدیل موجک این توانایی را از طریق کار کردن در مقیاس‌های (Scale) مختلف به دست می‌آورد.

در تبدیل موجک، ابتدا سیگنال را با مقیاس یا پنجره بزرگ در نظر می‌گیریم و ویژگی‌های بزرگ (Large Features) آن را آنالیز می‌کنیم. در گام بعد، با پنجره‌های کوچک به سیگنال نگاه می‌کنیم و ویژگی‌های کوچک سیگنال را به دست می‌آوریم. در تصویر زیر رزولوشن حوزه زمان و فرکانس در روش تبدیل‌های مختلف به نمایش درآمده است.



در تصـویر بـالا، انـدازه و جهـت بلوک‌هـا نشـان‌دهنده مقـدار رزولوشـن در آن تبـدیل اسـت، بـه عبـارت دیگـر بلوک‌هـا در هـر تبـدیل تعیـین می‌کننـد کـه در حـوزه زمـان و فرکـانس می‌تـوان ویژگی‌هـای تـا چـه مقـدار کوچـک را بـا اسـتفاده از آن تبـدیل تشـخیص داد. سـیگنال اصـلی کـه در حـوزه زمـان اسـت، دارای رزولوشـن بـالا در حـوزه زمـان و رزولوشـن صـفر در حوزه فرکـانس اسـت. ایـن ویژگـی بـه ایـن معنـی اسـت کـه می‌تـوانیم ویژگی‌هـای بسـیار کـوچکی را در حـوزه زمـان تشـخیص دهـیم، امـا در حـوزه فرکـانس هـیچ ویژگی قابل تمایز نیست.

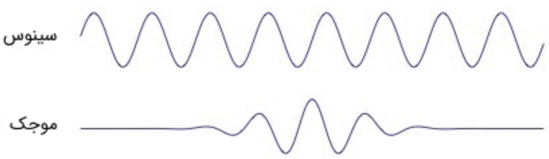
اما تبدیل فوریـه زمـان کوتـاه، دارای رزولوشـن بـا انـدازه متوسـط در هـر دو حـوزه زمـان و فرکـانس اسـت. رزولوشن تبدیل موجک به صورت زیر تغییر می‌کند:

- بـرای مقـادیر فرکانس‌هـای کوچـک، رزولوشـن بـالا در حـوزه فرکـانس و رزولوشـن پـایین در حوزه زمان دارد.
- بـرای مقـادیر فرکانس‌هـای بـالا، رزولوشـن پـایین در حـوزه فرکـانس و رزولوشـن بـالا در حوزه زمان دارد.

بنـابراین در حالـت کلـی می‌تـوان گفـت تبـدیل موجـک بـه صـورت مصـالحه‌ای عمـل می‌کنـد. در مقیاس‌هـایی کـه مشخصـه‌های وابسـته بـه زمـان جـذاب‌تر هسـتند، تبـدیل موجـک دارای رزولوشـن بـالاتر در حـوزه زمـان و در مقیاس‌هـایی کـه مشخصـه‌های وابسـته بـه فرکـانس جـذاب‌تر هسـتند، دارای رزولوشـن بـالاتر در حـوزه فرکـانس اسـت. ایـن نـوع مصـالحه دقیقـا همـان هـدفی است که در پردازش سیگنال مورد نظر است.

نحوه عملکرد تبدیل موجک

تبدیل فوریـه بـرای آنـالیز سـیگنال از یـک سـری امـواج سینوسـی بـا فرکانس‌هـای مختلـف اسـتفاده می‌کنـد. در ایـن حالـت، سـیگنال بـه صـورت ترکیبـی خطـی از سـیگنال‌های سینوسـی نمـایش داده می‌شـود. امـا تبـدیل موجـک از تعـدادی توابـع بـه نـام موجـک استفاده می‌کنـد کـه هـر کـدام مقیـاس متفـاوتی دارنـد. همـان طـور کـه مـی‌دانیم معنـی واژه موجـک، مـوج کوچـک اسـت و توابـع موجـک نیـز دقیقـا بـه همـین صـورت کوچـک هسـتند. در تصـویر زیـر تفـاوت بـین یـک سـیگنال سینوسـی و یـک موجـک نشـان داده شـده است.



بـا دقـت در تصـویر بـالا، کـاملا مشـخص اسـت کـه سـیگنال سینوسـی در یـک لحظـه خـاص از زمـان واقـع نشده است. ایـن سـیگنال از بی‌نهایـت شـروع می‌شـود و تـا بی‌نهایـت ادامـه می‌یابـد، در حـالی کـه یـک موجـک در لحظه خاصی از زمـان واقـع شـده اسـت. ایـن ویژگـی به تبدیل موجـک اجـازه می‌دهـد تـا عـلاوه بـر اطلاعـات فرکانسی، اطلاعات زمانی را نیز به دست آورد.

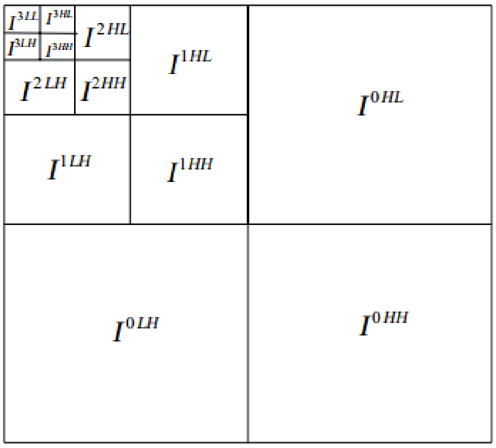
چون موجک در زمان واقع شده است، در نتیجه می‌توان سیگنال اصلی را در لحظات مختلف از زمان در موجک ضرب کرد. در گام نخست، با نقاط ابتدایی سیگنال شروع می‌کنیم و به تدریج موجک را به سمت انتهای سیگنال حرکت می‌دهیم. این عمل را کانولوشن (Convolution) می‌گویند. بعد از این که کانولوشن را با سیگنال موجک اصلی (موجک مادر) انجام دادیم، می‌توانیم آن را به نحوی مقیاس‌دهی کنیم که بزرگ‌تر شود و دوباره فرایند را تکرار کنیم. این فرایند در تصویر متحرک زیر نشان داده شده است.

تبدیل موجک پیوسته و تبدیل موجک گسسته

تبدیل موجک دارای دو نوع مختلف گسسته و پیوسته است. از لحاظ ریاضی، یک تبدیل موجک پیوسته را می‌توان توسط تابع زیر توصیف کرد:

$$X_\omega(a,b) = \frac{1}{|a|^{\frac{1}{2}}} \int_{-\infty}^{\infty} x(t)\bar{\psi}\left(\frac{t-b}{a}\right)dt$$

با توجه به اینکه تصاویر دارای دو بعد می‌باشند، اگر یک تصویر توسط تبدیل موجک گسسته مورد تجزیه قرار گیرد، چهار تصویر بدست می آید : یک تصویر مربوط به کلیات و سه تصویر مربوط به جزئیات (جزئیات افقی، عمودی وقطری).



2-توضیحات تکنیکال

هرم ها:

هرم تصویر یا Image Pyramid به طور گسترده در بسیاری از کاربردهای بینایی ماشین کاربرد دارد. یک هرم تصویر مجموعه ای از تصویرها است که همگی ناشی از یک تصویر اصلی هستند به طوری که به صورت پیوسته تا یک نقطه دلخواه به چندین نمونه کاهش (Downsample) پیدا می کنند. به طور کلی دو دسته از هرم های تصویر به نام گوسی (Gaussian) و لاپلاسین (Laplacian) وجود دارد.

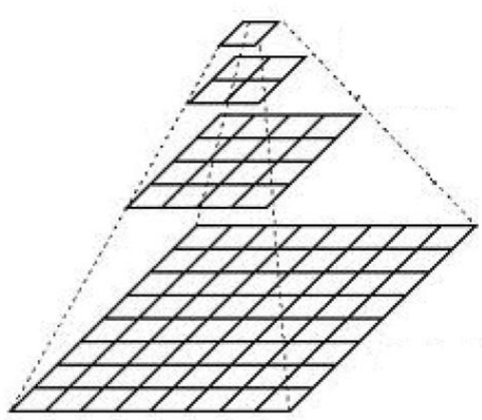
واژه Downsampled به فرایندی گفته می شود که تصویرهای کوچکتر و قاعدتا با ابعاد کوچکتر از تصویر اولیه ساخته می شود و این کاهش ابعاد تا یک نقطه دلخواه صورت خواهد گرفت. ولی بلعکس، Upsampled فرایندی است که در آن تصویرهای بزرگتر و قاعدتا با ابعادی بزرگتر از تصویر کوچکتر ساخته می شود.

هرم گوسی که در این نوشته در مورد آن گفتگو می کنیم برای انجام Downsampled به کار گرفته می شود.

- هرم گوسی: در این روش برای ساخت هرم از میانگین‌گیری گوسی استفاده می‌شود.
- هرم لاپلاسی: مشابه هرم گوسی است با این تفاوت که از یک تبدیل لاپلاس برای ساخت هرم استفاده می‌کند.

❖ هرم گوسی:

هرم را به صورت یک مجموعه از لایه‌ها در نظر بگیرید که هر چه لایه در سطح بالاتری قرار گرفته باشد، کوچک‌تر خواهد بود.



لایه‌ها از پایین به بالا شماره گذاری شده‌اند. پس لایه i+1 که با G_{i+1} نشان داده می‌شود، از لایه i که با G_i نشان داده می‌شود کوچکتر است.

در هرم گوسی برای تولید لایه i+1 از روی لایه i، به شیوه زیر عمل می‌کنیم:

G_i را با یک کرنل گوسی زیر کانوالو می‌کنیم:

$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

همه سطرها و ستون‌های زوج را پاک می‌کنیم.

عکس به دست آمده دقیقاً یک چهارم عکس پیشین خواهد بود. با تکرار فرایند بالا روی عکس اصلی (یعنی G_0) ، می‌توان تمام هرم را درست کرد.

رویه بالا به منظور کوچک نمایی تصویر بود. برای بزرگ‌نمایی تصویر باید مراحل زیر را انجام دهیم:

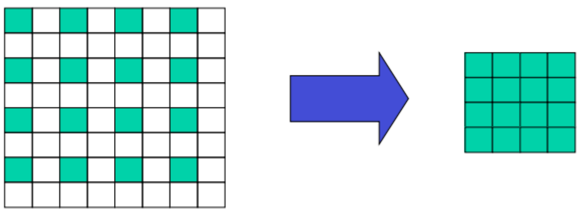
ابتدا ابعاد لایه i را دو برابر کرده و سطرها و ستون‌های زوج آن را با صفر پر می‌کنیم.

تصویر به دست آمده در مرحله قبل را در کرنل بالا (البته با درایه‌های آن که در 4 ضرب شده‌اند) کانوالو می‌کنیم. با این کار مقدار پیکسل‌های جدید معرفی شده در مرحله قبل که با صفر پر شده بودند، تخمین زده می‌شوند.

تولید هرم گاوسی:

توابع پایه:

1. Blur (کانوالو با گوسین برای smooth کردن تصویر) : این کار در تمارین قبلی مفصلا بررسی شده است.
2. DownSample (کاهش اندازه تصویر به نصف) :



همان طور که پیداست، Subsampling ایده بدی است مگر اینکه قبلاً تصویر را blurred/smoothed کرده باشیم. (زیرا منجر به aliasing می‌شود)

در تصاویر زیر این موضوع مشهود است:

original image
262x195

downsampled (left)
vs. smoothed then
downsampled (right)

original image
262x195

downsampled (left)
vs. smoothed then
downsampled (right)
131x97

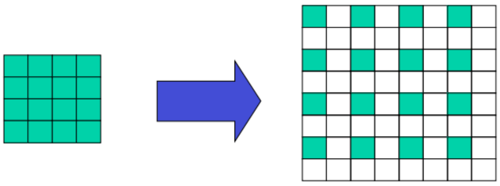
original image
262x195

downsampled (left)
vs. smoothed then
downsampled (right)
65x48

original image
262x195

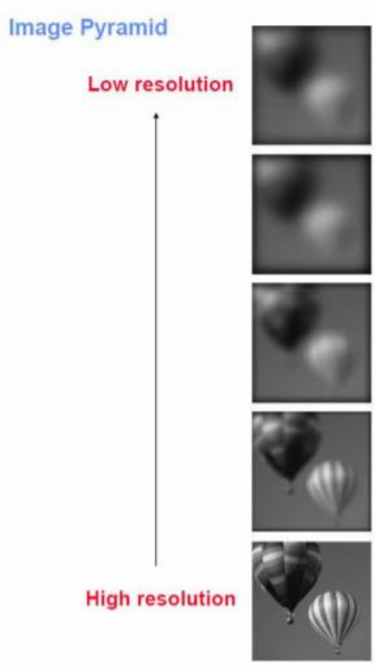
downsampled (left)
vs. smoothed then
downsampled (right)
32x24

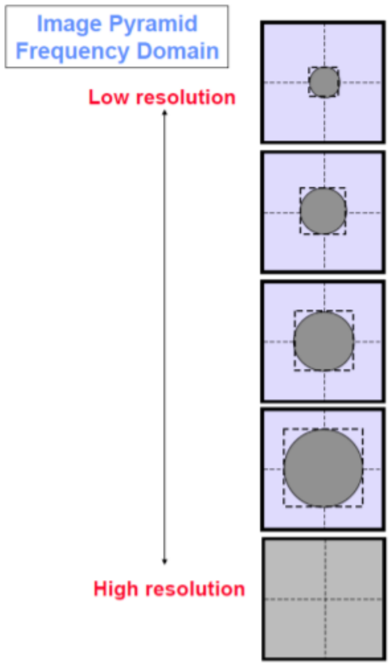
Upsample (دوبرابر کردن اندازه تصویر):



برای پر کردن مقادیر خالی، با روش درون‌یابی، ابتدا مقادیر خالی را صفر می‌دهیم. سپس تصویر upsampled را با فیلتر گاوسین کانوالو می‌کنیم (مثلا 5x5 kernel with sigma = 1.)

درنهایت باید در 4 ضرب نماییم.





هزینه محاسباتی:

Memory:

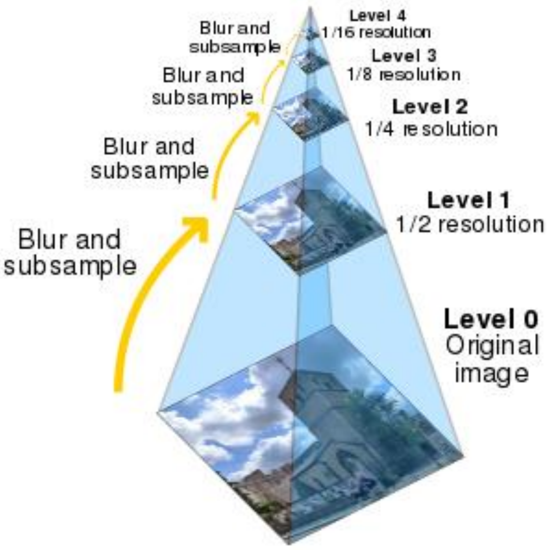
$$2^n * 2^n(1 + ¼ + 1/16 + \ldots) = 2^n * 2^n * 4/3$$

Computation:

هر مرحله با یک کانولوشن به دست می‌آید.

❖ **هرم لاپلاسی:**

هرم لاپلاسی بسیار شبیه به هرم گاوسی است اما تصویر تفاوت نسخه های blurred بین هر سطح را ذخیره می کند. فقط کوچکترین سطح برای ایجاد امکان بازسازی تصویر با وضوح بالا با استفاده از تصاویر اختلاف در سطوح بالاتر ، یک تصویر حاصل از تفاضل(difference image) نیست. از این روش می توان در فشرده سازی تصویر استفاده کرد.



تولید هرم لاپلاسین:

روش اول:

Synthesize:
P_L(0) = X – Blur(X)
P_L(1) = Reduce(Blur(X))

Reconstruct:
X = Expand(P_L(1)) + P_L(0) =
Expand(Reduce(Blur(X)) + X – Blur(X) != X

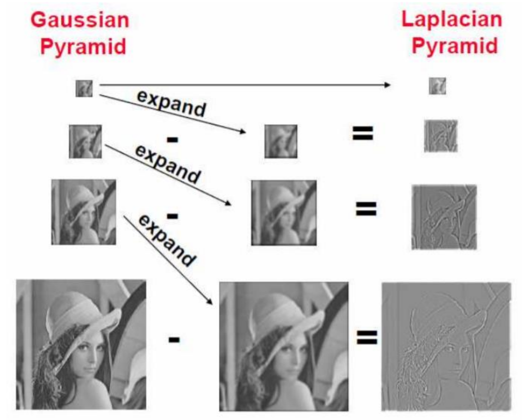
روش دوم:

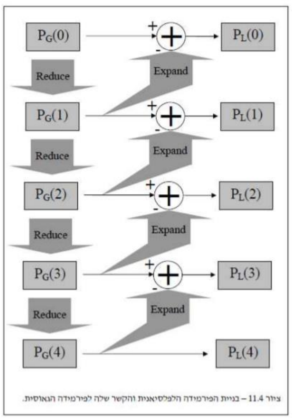
Synthesize:
P_G(0) = X
P_G(1) = Reduce(Blur(P_G(0))
P_L(0) = P_G(0) – Expand(P_G(1))
P_L(1) = P_G(1)

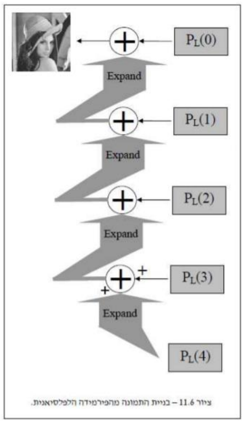
Reconstruct:
X = Expand(P_L(1)) + P_L(0)
= Expand(P_G(1)) + P_G(0) – Expand(P_G(1))
= Expand(Reduce(Blur(P_G(0))) + P_G(0) –
Expand(Reduce(Blur(P_G(0))) = P_G(0) = X

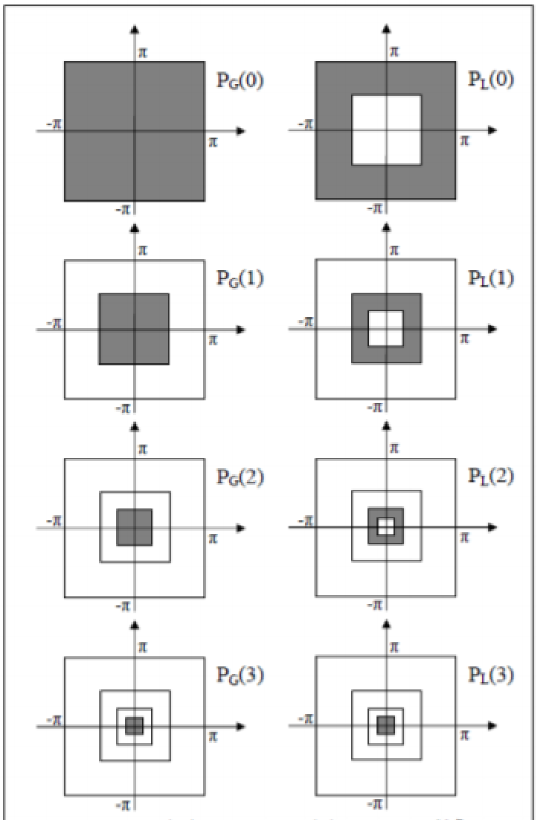
الگوریتم حل سوال:

- Given image X
- Set $P_G(0) = X$
- For i=1:numLevels
  - $P_G(i)$ = smooth and sub-sample $P_G(i-1)$
  - Define $P_L(i-1) = P_G(i-1)$ - expand $P_G(i)$
- Set $P_L(\text{numLevels}) = P_G(\text{numLevels})$









ציור 11.7 –הפירמידה הלפלסיאנית והקשר שלה לפירמידה הגאוסית – ייצוג בתדר.

هزینه محاسباتی:

Memory:

$$2^n * 2^n (1 + 1/4 + 1/16 + \ldots) = 2^n * 2^n * 4/3$$

اما ضرایب می توانند فشرده شوند.

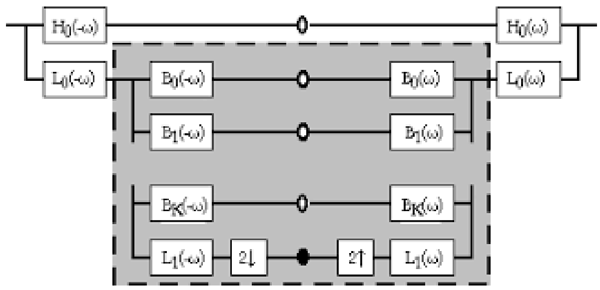
Computation:

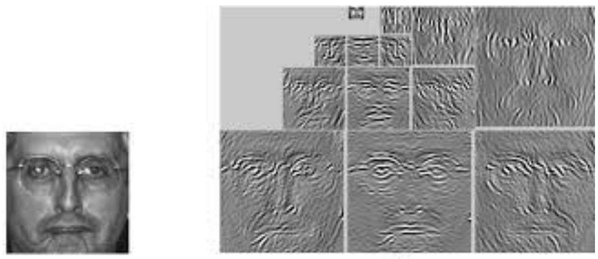
هر مرحله با یک کانولوشن به دست می‌آید.

❖ **هرم steerable:**

یک هرم قابل هدایت ، که توسط Simoncelli و دیگران ساخته شده است ، پیاده سازی یک بانک فیلتر باند گذر چند منظوره و چند جهته است که برای کاربردهایی از جمله فشرده سازی تصویر ، سنتز بافت و تشخیص شی استفاده می شود. می توان آن را به عنوان یک نسخه انتخابی جهت گیری از هرم لاپلاس در نظر گرفت که در آن از بانک

فیلترهای قابل استفاده در هر سطح از هرم به جای یک فیلتر لاپلاسی یا گاوسی استفاده می شود.



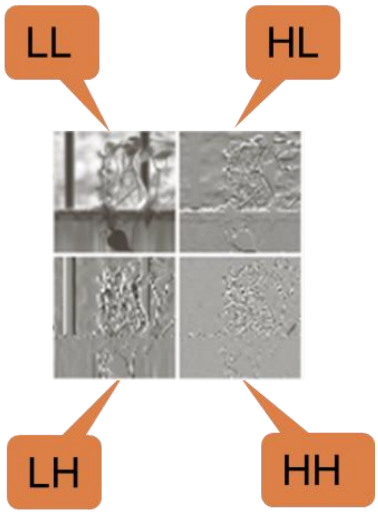


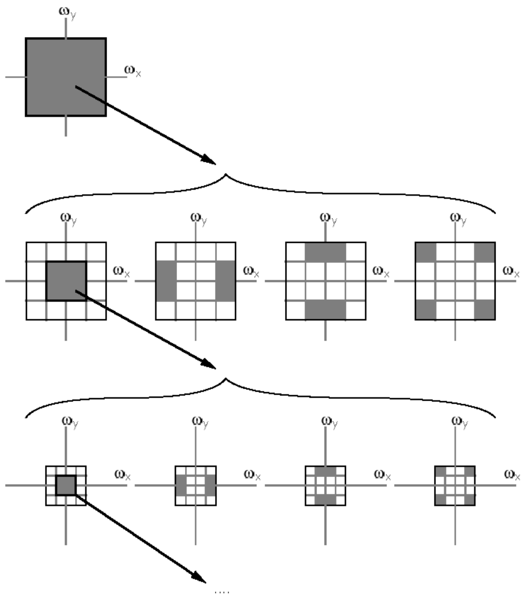
a

b

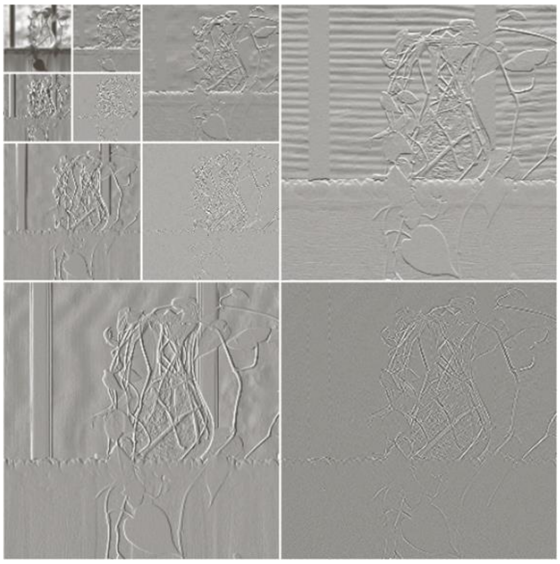
❖ هرم موجک:

با استفاده از خاصیت جدایی‌پذیری، به کمک فیلترهای یک بعدی پیاده‌سازی می‌شود:





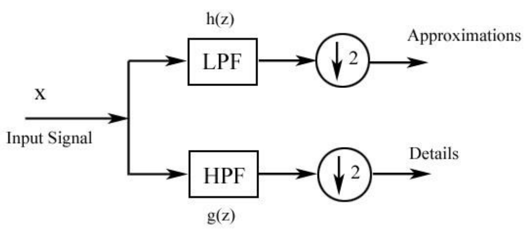
برای مثال:



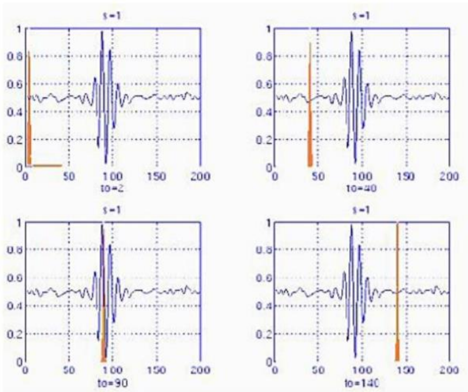
مزایا:

- قابلیت پیاده سریع تبدیل موجک
- عدم نیاز به بافر کمکی در بدست اوردن تبدیل
- عدم نیاز به تبدیل فوریه

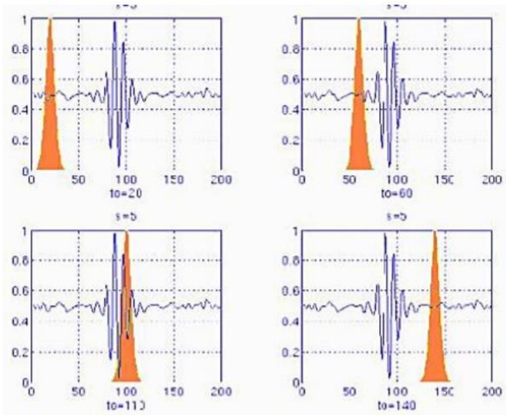
پیاده‌سازی تبدیل موجک به روش Mallat:



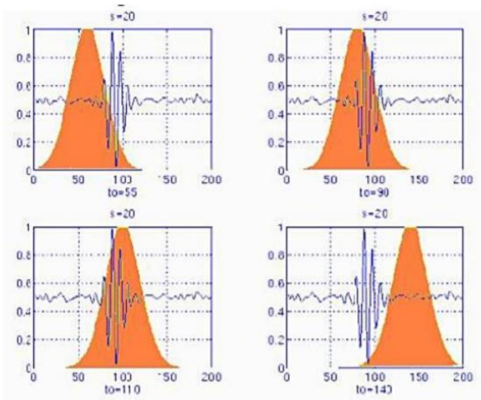
تبدیل موجک در مقیاس‌های مختلف (پایین)



تبدیل موجک در مقیاس‌های مختلف (متوسط)



تبدیل موجک در مقیاس‌های مختلف (بالا)



فرض کنیم توالی داده های ورودی زیر به ما داده شده است:

$$\{x_{n,i}\} = \{10, 13, 25, 26, 29, 21, 7, 15\}$$

تبدیل موجک هار: به جای توالی اصلی، میانگین هر دو عنصر متوالی $x_{n-1,i}$ و تفاضل آن دو $d_{n-1,i}$ را جایگزین می‌کند:

$$x_{n-1,i} = \frac{x_{n,2i} + x_{n,2i+1}}{2}$$

$$d_{n-1,i} = \frac{x_{n,2i} - x_{n,2i+1}}{2}$$

تفاضل ها و میانگین ها فقط روی جفت های پشت سرهم سری داده ای اعمال می شوند که اندیش اولین عضو آن‌ها زوج باشد. بنابریان، تعداد اعضا در هر سری $\{x_{n-1,i}\}$ و $\{d_{n-1,i}\}$ دقیقاً نصف تعداد اعضا توالی اصلی است.

دنباله جدیدی که طول آن برابر طول دنباله اصلی است با به هم پیوستن دو دنباله $\{x_{n-1,i}\}$ و $\{d_{n-1,i}\}$ ایجاد می‌شود:

$$\{x_{n-1,i}, d_{n-1,i}\} = \{11.5, 25.5, 25, 11, -1.5, -0.5, 4, -4\}$$

این دنباله دقیقاً همان تعداد اعضا دنباله داده ورودی را دارد. عمل تبدیل مقدار داده را افزایش نداده است.

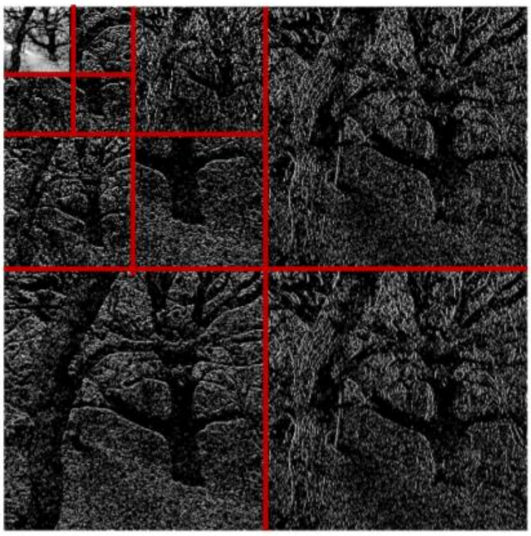
از آنجایی که نیمه اولیه دنباله بالا حاوی میانگین های دنباله اصلی است، می توانیم آن را به عنوان تقریب بزرگتری از سیگنال اصلی در نظر بگیریم. نیمه دوم، در واقع جزئیات یا خطاهای تقریب نیمه اول است.

درستی این قضیه که دنباله اصلی می تواند از روی دنباله مبدل با استفاده از این روابط دوباره ساخته شود، به راحتی قابل اثبات است.

$$x_{n,2i} = x_{n-1,i} + d_{n-1,i}$$

$$x_{n,2i+1} = x_{n-1,i} - d_{n-1,i}$$

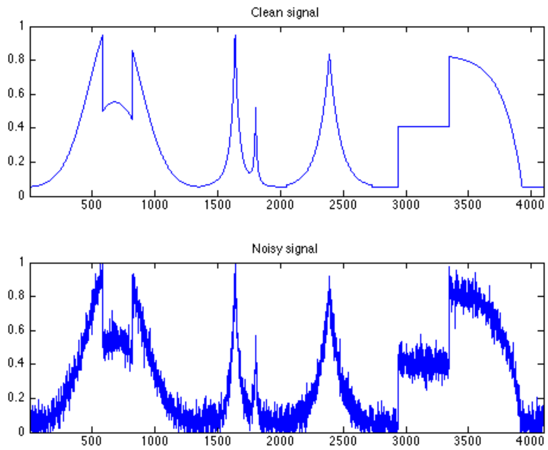
این تبدیل، موجک با تبدیل هار Haar نام دارد.



تجزیه Haar wavelet در سه مرحله

❖ **حذف نویز به کمک تبدیل موجک:**

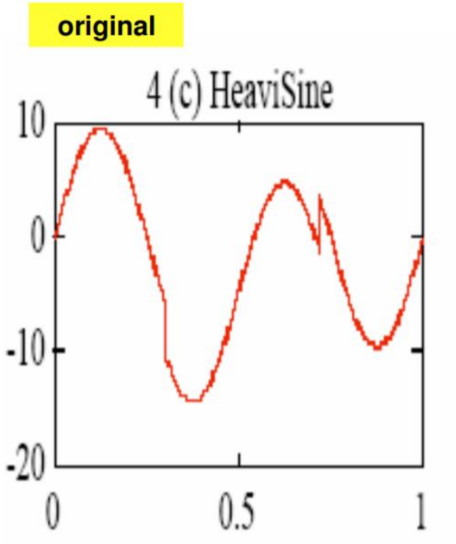
حذف نویز روندی است که ما با آن یک سیگنال را از یک سیگنال noisy بازسازی می‌کنیم.

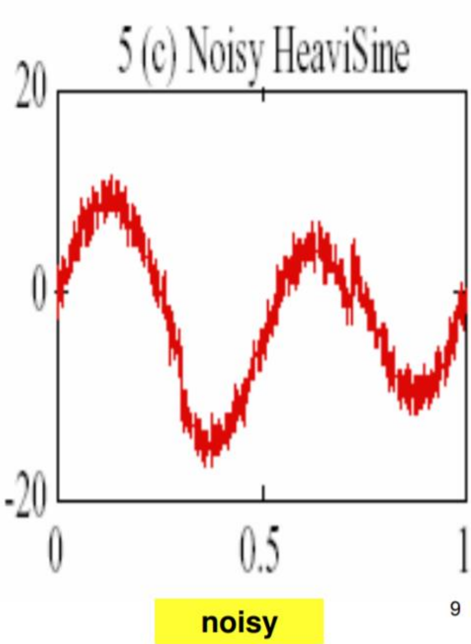


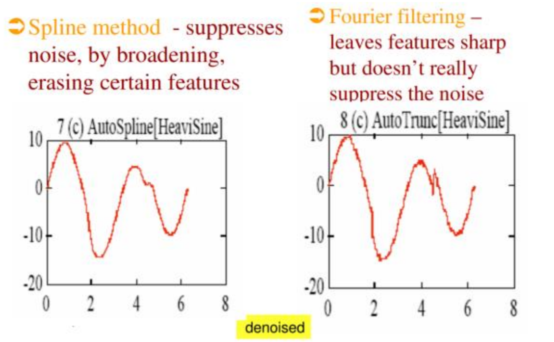
مشکلات روش‌های قبلی:

- هنوز ساختارهای محلی به اندازه کافی حل و فصل نشده اند.. این در هنگام برخورد با سیگنال هایی که حاوی ساختارهایی از مقیاس ها و دامنه های مختلف اند لازم است مانند سیگنال های مغز و اعصاب.

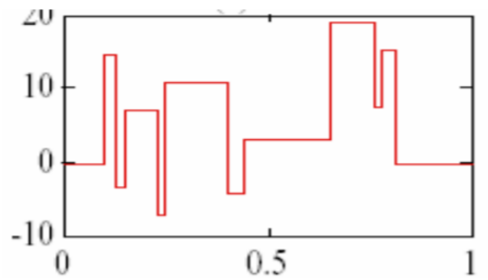
- ما سیگنالهای خود را به گونه ای مرتب می کنیم که سیگنالها و هر نوع همپوشانی نویز تا حد ممکن در دامنه فرکانس و فیلترخطی زمان ثابت تقریبا آن ها را از هم جدا کند. این روش فیلتر کردن خطی نمی تواند نویز را از سیگنالی که طیف فوریه آنها با هم همپوشانی دارند جدا کند.

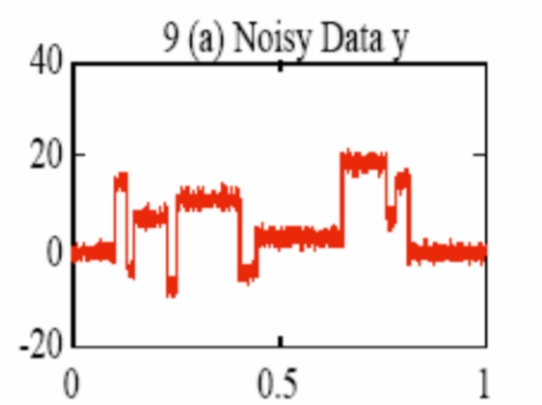


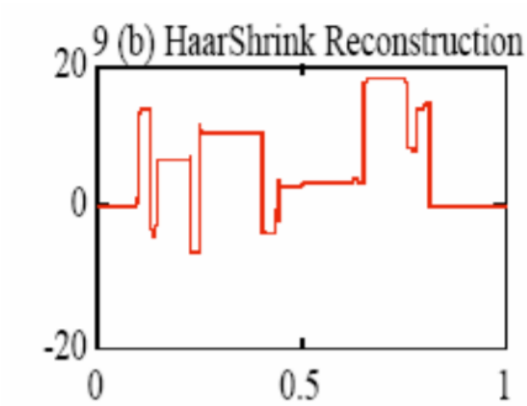




در سیگنال زیر میخواهیم نویز را به روش Haar-basis shrinkage حذف نماییم:







تبدیل Wavelet یک تجزیه و تحلیل همبستگی را انجام می دهد ، بنابراین انتظار می رود که خروجی حداکثر زمانی باشد که سیگنال ورودی بیشتر شبیه موجک مادر باشد.

اگر سیگنال انرژی خود را در تعداد کمی از ابعاد WL متمرکز کند ، ضرایب آن در مقایسه با هر سیگنال یا نویز دیگری که انرژی آن در تعداد زیادی ضریب پخش می شود ، نسبتاً زیاد خواهد بود.

این بدان معنی است که کوچک شدن تبدیل WL باعث حذف نویز با دامنه کم یا سیگنال نامطلوب در حوزه WL می شود ، و یک تبدیل موجک معکوس پس از آن سیگنال مورد نظر را با کمبود جزئیات بازیابی می کند.

معمولاً همان خصوصیاتی که باعث می شود سیستم با استفاده از روش های غیرخطی برای تخلیه یا جداسازی مناسب باشد ، آن را برای فشرده سازی مناسب می کند ، که این نیز فرایندی غیرخطی است.

فرآیند حذف نویز:

denoising موجک برای نویز افزودنی(additive) کار می کند از آنجا که تبدیل موجک خطی است:

$$W(a,b)[f+\eta;\psi] = W(a,b)[f;\psi] + W(a,b)[\eta;\psi]$$

1. تجزیه سیگنال با استفاده از DWT ؛
   a. انتخاب موجک و تعداد سطح تجزیه.
   b. محاسبه‌ی Y = Wy
2. آستانه گذاری را در حوزه Wavelet
   a. ضرایب کوچک شدن توسط آستانه (سخت / نرم)

3. بازسازی سیگنال از ضرایب آستانه DWT

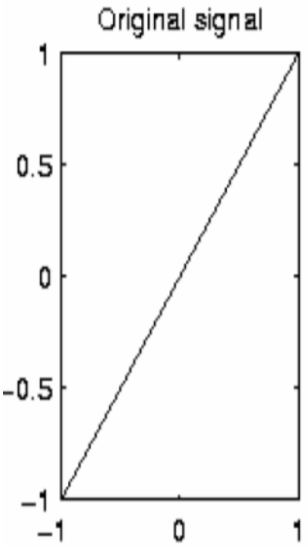
a. محاســـــــــــــــــــــــــــبه:

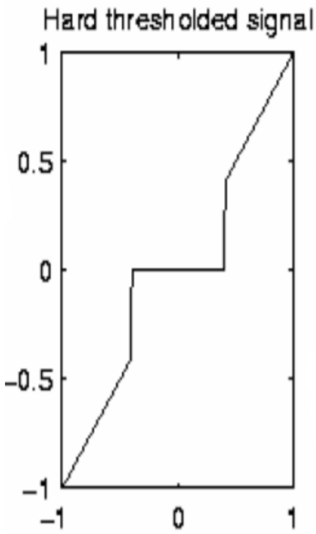
$$\hat{x} = W^{-1}\hat{X}$$

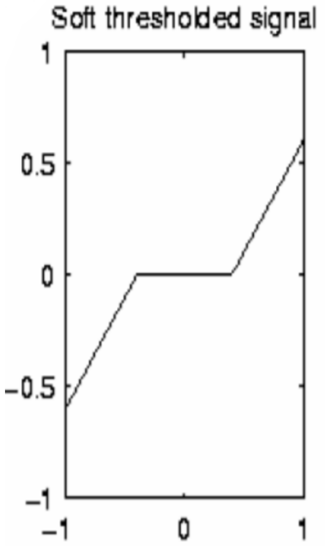
روش‌های آستانه گذاری:

به طور کلی روش **soft** و**hard** وجود دارد.

با کمک نمـودترهـا ی زیـر مـی‌تـوانیم مقایســه‌ای کلـی بین دو روش بکنیم:

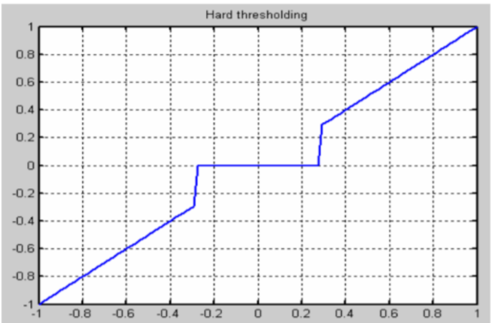






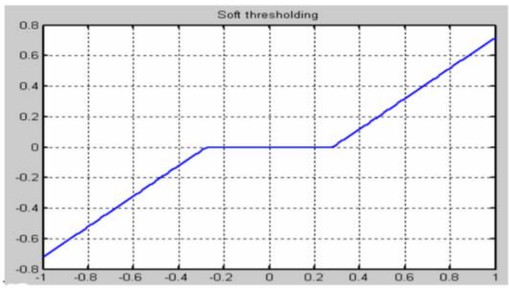
❖ **Hard thresholding:**

$$y_{hard}(t) = \begin{cases} x(t), & |x(t)| > \delta \\ 0, & |x(t)| < \delta \end{cases}$$



❖ **Soft trsholding:**

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



مقایسه بین این دو روش:

- گفته می‌شود که آستانه نرم در مقایسه با آستانه سخت نتایج smooth تری را ارائه می دهد.
- از نظر بصری تصاویر دلپذیرتر هستند ، زیرا پیوسته است.
- با این وجود آستانه سخت در مقایسه با نرم، باعث حفظ بهتر لبه می شود.
- بعضی اوقات ممکن است خوب باشد که آستانه نرم را روی تعداد کمی از سطوح جزئیات اعمال کنیم و بر روی بقیه سطوح از آستانه سخت استفاده نماییم.

3-شکل‌ها، جدول‌ها و روابط (فرمول‌ها)

تصویر ورودی:

هرم لاپلاسین 9 مرحله ای:

تصویر بازیابی شده از این هرم:

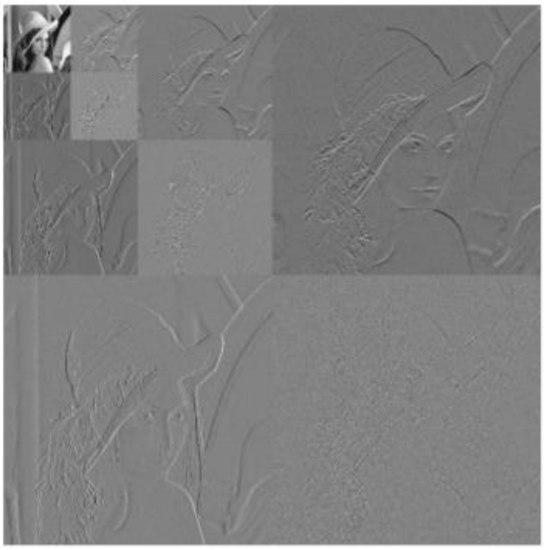
برای مقایسه این تصویر با تصویر اصلی می‌توان psnr و mse را محاسبه کرد.

مقدار حاصل برای psnr، Inf خواهد بود که نشان می‌دهد تصویز حاصل از بازسازی با تصویر اولیه identicalهستند.
همچنین mse نیز مقدار صفر دارد.
مرحله آخر تنها شامل یک پیکسل است.
پس برای تصویر n*n که $n = 2^j$است j+1 سطح تجزیه میکنیم.
هرم لاپلاسین 3 مرحله ای:

تصویر بازیابی شده از این هرم:

همانند آنچه در بخش قبل دیدیم این تصویر با تصویر اولیه identical هستند.
پس نتیجه می‌گیریم که هرم لاپلاسین در نگهداری اطلاعات مهم تصویر بسیار دقیق عمل می‌کند.

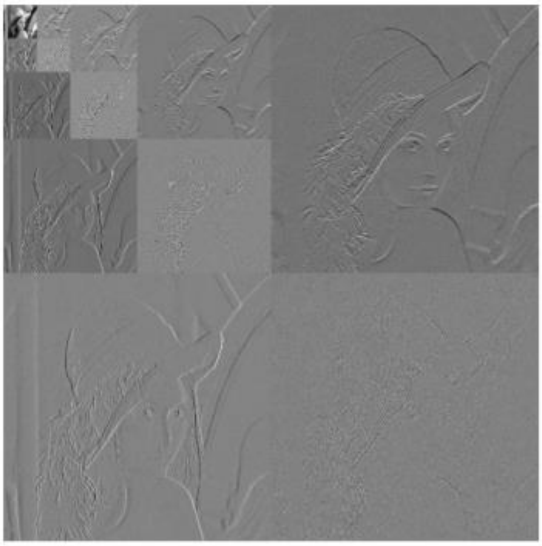
هرم موجک سه مرحله ای:



مطابق آنچه گفته شد تصویر بالا شامل hh،lh،hl همه مراحل و ll مرحله‌ی آخر است.
تصویر بازیابی شده از این هرم:

با محاسبه مقادیر mse = 0 و psnr =idf در می‌یابیم که این هرم نیز در نگهداری اطلاعات مهم تصویر بسیار عملکرد خوبی دارد.

هرم موجک چندی سازی شده:



تصویر بازیابی شده از این هرم:

همان طور که توقع داشتیم مقادیر mse = 1.53 و psnr =45.2 نشان می‌دهند که این هرم با وجود عملکرد بسیار بالا، از هرم لاپلاسین و موجک اندکی ضغیف تر است.

حذف نویز از تصویر:

با توجه به اینکه از آستانه گذازی نرم استفاده کرده ایم، لبه ها تضعیف شده اند اما به طور کلی نویز کمتری در تصویر وجود دارد.

البته با توجه به آنچه در توضیحات تکنیکال بیان شد استفاده‌ی مطلق از یکی از روش‌های حذف نویز پیشنهاد نمی‌شود و در صورتی-که هر دو روش را ترکیب نماییم، نتایج بهتری حاصل خواهد شد.

4- کدها:

هرم لاپلاسین:

```
function lp(img,levels)
avg_filter = [1 1;1 1]/4;
[pyramid,laplacians] = lpyramid(img,levels,avg_filter);
[n,~] = size(laplacians);
recovered = laplacians{n};
for i=1:levels
    n = n-1;
    recovered = pixel_rep(recovered);
    recovered = recovered + laplacians{n};
end
recovered = uint8(recovered);
imshow(recovered);
figure ,imshow(pyramid);
```

تابع lpyramid:

```
function [final,laplacians_recovered] = lpyramid(img,levels,avg_filter)
    [n,m] = size(img);
    final = zeros(2*n,2*m,'uint8');
    img = double(img);
    N = levels;
    laplacians_recovered = cell(N,1);
    for i=1:levels
        r1 = n-(0.5)^(i-1)*n+1;
        r2 = n;
        c1 = ((2^(i-1)-1)/(2^(i-2)))*m+1;
        c2 = (2^(i)-1)/(2^(i-1))*m;
        final(r1:r2,c1:c2) = img;
        new_img = average_filter(img,avg_filter);
        laplace = img - pixel_rep(new_img);
        laplacians_recovered(i) = laplace;
        laplace = norm(laplace);
        laplace = uint8(laplace);
        r1 =n+1;
        r2 = n+(0.5)^(i-1)*n;
        final(r1:r2,c1:c2)=laplace;
        img = new_img;
    end
    i = levels+1;
    r1 = n-(0.5)^(i-1)*n+1;
    r2 = n;
    c1 = ((2^(i-1)-1)/(2^(i-2)))*m+1;
    c2 = (2^(i)-1)/(2^(i-1))*m;
    final(r1:r2,c1:c2) =img;
    r1 =n+1;
```

```
    r1 =n+1;
    r2 = n+(0.5)^(i-1)*n;
    final(r1:r2,c1:c2) = img;
    laplacians_recovered(levels+1)=img;
end
```

تابع محاسبه تصویر average:

```
function output=average_filter(image,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');

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

```
function output = pixel_rep(img)
    [r,c] = size(img);
    output = zeros(2*r,2*c,class(img));
    for x = 1:r
        for y = 1:c
            j = 2*(x-1) +1;
            i = 2*(y-1) +1;
            output(j,i) = img(x,y);
            output(j+1,i)= img(x,y);
            output(j,i+1)= img(x,y);
            output(j+1,i+1) =img(x,y);
        end
    end
end
```

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

end
```

هرم موجک:

```
function wav_pyr(img,level)
out = wavelet(img,level);
res = wt(out,level);
imshow(out);
figure,imshow(uint8(res));
end
```

توابع:

```
function output = wt(pyramid,level)
    output = pyramid;
    [R,C] = size(output);
    x = output(1:R/(2^(level1)),1:C/(2^(level-1)));
    [r,c] = size(x);
    A = x(1:r/2,1:c/2);
    H = x(1:r/2,c/2+1:c);
    V = x(r/2+1:r,1:c/2);
    D = x(r/2+1:r,c/2+1:c);
    output(1:R/(2^(level1)),1:C/(2^(level-1))) = idwt2(A,H,V,D,'haar');
    output = wt(output,level-1);
end

function output = wavelet(img,level)
    [R,C] = size(img);
    output= zeros(R,C,'double');
    wfilters('haar','d');
    [A,H,V,D] = dwt2(img,'haar');
    output(1:R/2,1:C/2) = wavelet(A,level-1,present);
    output(R/2+1:R,1:C/2)=V;
    output(1:R/2,C/2+1:C)=H;
    output(R/2+1:R,C/2+1:C)=D;
end
```

هرم موجک چندی سازی شده:

```
function [out] = wav2(img,level)
out = wavelet(img,level);
x = wt(out,level);
imshow(out);
figure,imshow(uint8(x));
end
```

```
function output = wavelet(img,level)
    [R,C] = size(img);
    output= zeros(R,C,'double');
    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);
    output(1:R/2,1:C/2) = wavelet(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 out = quantizer(img,g)
    [M,N] = size(img);
    out = zeros(M,N,'double');
    for i=1:M
        for j=1:N
            out(i,j) = g*sign(img(i,j))*floor(abs(img(i,j))/g);
        end
    end
end

function output =wt(wt_pyramid,level)
    output = wt_pyramid;
    [R,C] = size(output);
    x = output(1:R/(2^(level1)),1:C/(2^(level-1)));
    [r,c] = size(x);
    A = x(1:r/2,1:c/2);
    V = x(r/2+1:r,1:c/2);
    H = x(1:r/2,c/2+1:c);
    D = x(r/2+1:r,c/2+1:c);
    output(1:R/(2^(level1)),1:C/(2^(level-1))) = idwt2(A,H,V,D,'haar');
    output = wt(output,level-1);
end
```

Denoising:

```
function [output] = untitled3(img,level)
noisyImg = imnoise(img,'gaussian',0.01);
output = wt(noisyImg,level);
x = iwt(output,level);
imshow(noisyImg);
figure,imshow(uint8(x));
end

    function output= soft_treshold(img,beta,sigma)
        [R,C] = size(img);
        output = zeros(R,C);
        local_sigma = std(double(img));
        T = (beta*sigma)/local_sigma;
        for i=1:R
            for j=1:C
                res = img(i,j);
                if(abs(res)<T)
                    res =0;
                else
                    res = sign(res)*abs(res-T);
                end
                output(i,j)=res;
            end
        end

    end
```

```
function output = wt(img,level)
    [R,C] = size(img);
    output = zeros(R,C,'double');
    [A,H,V,D] = dwt2(img,'haar');
    beta = sqrt(log2(R/3));
    sigma = median(abs(cD)./0.6745).^2;
    H = soft_treshold(H,beta,sigma);
    D = soft_treshold(D,beta,sigma);
    V = soft_treshold(V,beta,sigma);
    output(1:R/2,1:C/2) = wt(A,level-1);
    output(R/2+1:R,1:C/2)=V;
    output(1:R/2,C/2+1:C)=H;
    output(R/2+1:R,C/2+1:C)=D;
end
```

مراجع

- https://blog.faradars.org/%D8%AA%D8%A8%D8%AF%DB%8C%D9%84-%D9%85%D9%88%D8%AC%DA%A9/#%D8%A7%D8%B2_%D8%AA%D8%A8%D8%AF%DB%8C%D9%84_%D9%81%D9%88%D8%B1%DB%8C%D9%87_%D8%A8%D9%87_%D8%AA%D8%A8%D8%AF%DB%8C%D9%84_%D9%85%D9%88%D8%AC%DA%A9
- http://fadaei.semnan.ac.ir/uploads/13Wavelet.pdf

- http://www.eng.tau.ac.il/~ipapps/Slides/lecture05.pdf
- https://slideplayer.com/slide/15477142/
- http://www.cse.psu.edu/~rtc12/CSE486/lecture10.pdf
- http://www.numerical-tours.com/matlab/denoisingwav_1_wavelet_1d/
- http://cs.haifa.ac.il/hagit/courses/seminars/wavelets/Presentations/Lecture09_Denoising.pdf