wavelet

فهیم جعفری

اطلاعات گزارش

تاریخ:

واژگان کلیدی:
Prediction pyramid
Approximation pyramid
Medin filter
Threshold
Box filter
Gaussian filter
Laplacian pyramid
Wavelet transform

چکیده

تبدیلات موجک یا ویولت (Wavelet) از جمله ابزارهایی هستند که کاربردهای فراوانی در شاخه های مختلف علمی و مهندسی به ویژه هوش مصنوعی، یادگیری ماشینی و پیش بینی سری زمانی و بازشناسی الگو دارد. تئوری موجک ها در واقع تعمیمی بر تئوری تبدیلات و سری های فوریه است و ضعف های آنالیز فوریه در عملکرد موضعی و مدل سازی رفتارهای کوتاه مدت را جبران می نماید.تبدیل موجک تجزیه یک تابع بر مبنای توابع موجک می‌باشد. موجک‌ها (که به عنوان موجک‌های دختر شناخته می‌شوند) نمونه‌های انتقال یافته و مقیاس شده یک تابع (موجک مادر) با طول متناهی و نوسانی شدیداً میرا هستند. چند نمونه موجک مادر در شکل زیر نمایش داده شده‌اند.

1-مقدمه

2-شرح تکنیکال

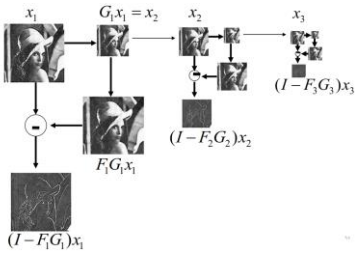
- **Pyramid**
  - Approximation pyramid

برای درست کردن این هرم به تصویر فیلتر گوسی اعمال میکنیم تا قضیه شانون را رعایت کرده باشیم و سپس سایز تصویر فیلتر شده به تصف سایز تصویر اصلی کاهش میدهیم سپس بر روی تصویر کوچک شده مراحل فوق را دوباره تکرار میکنیم و تصویری با سایز یک چهارم سایز تصویر اصلی بدست می اید و این مراحل را تا سطح دلخواه انجام میدهیم که این سطح دلخوه باید از $j+1$ کوچکتر باشد که 2 به توان j برابر با سایز تصویر اصلی میباشد

- Prediction pyramid

برای درست کردن این هرم پس از انکه هرم گوسی را بدست اوردیم از تصویر دوم شروع میکنیم و ان را upsample میکنیم حال این تصویر بزرگ شده که سایز برابر با تصویر اصلی دارد را از تصویر اصلی کم میکنیم و یک تصویر که تنها دارای لبه ها میباشد بدست می اید و این را عنوان اولین تصویر از هرم، در هرم لاپلاسین(prediction) قرار میدهیم و سپس به سراغ تصویر بعدی در هرم

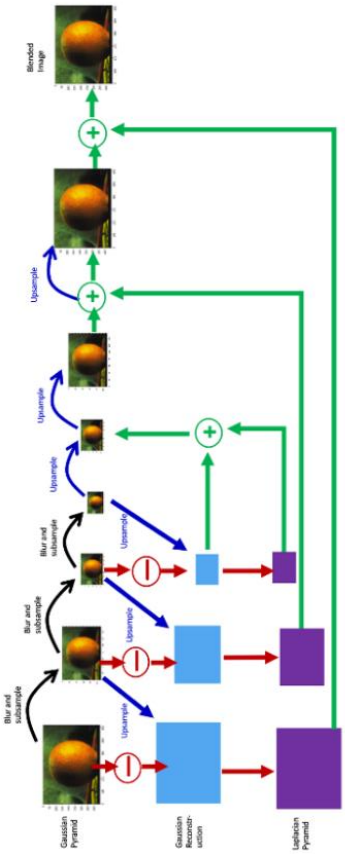
گوسی میرویم و مراحل فوق را تکرار میکنیم این مراحل را تا اندازه ای که در هرم گوسی گفته شد میتوانیم انجام دهیم که کوچکترین تصویر ممکن هرم گوسی و لاپلاسی با هم برابر و مساوی میانگین تصویر اصلی میباشد.



اگر بخواهیم از روی هرم گوسی و هرم لاپلاسی دوباره تصویر اصلی را بدست بیاوریم کافی است که تصویر لاپلاسی را با تصویر گوسی بزرگ شده مرحله بعد ان جمع کنیم که شروع این عمل از کوچکترین گوسی و یکی مانده به اخر کوچکترین لاپلاسی میباشد و به تصویر اصلی ختم میشود

- **بازسازی تصویر با هرم گوسی و هرم لاپلاسی**

اگر بخواهیم تصویر اصلی را بازسازی کنیم کافی است کوچکترین تصویر هرم گوسی را upsample کنیم و با تصویر لاپلاسی همسایز ان که از مرحله قبل در هرم لاپلاسی میباشد جمع کنیم که در این صورت تصویر گوسی مرحله قبل در هرم گوسی بدست میاید سپس مراحل فوق را برای تصویر بدست امده انجام میدهیم تا تصویر مرحله قبل هرم گوسی بدست بیاید این مراحل را تا رسیدن به تصویر اصلی که از جمع بزرگترین تصویر هرم لاپلاسی و تصویر خروجی که از این روند بدست امده انجام میدهیم که شمای کلی این روند به همراه ساخت هرم گوسی و لاپلاسی به صورت زیر میباشد



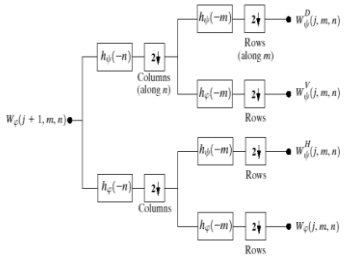
- هرم گوسی با میانگیری

میتوان در هرم گوسی به جای فیلتر گوسی از فیلتر میانگین استفاده کرد که به جای تصویر فیلتر شده با فیلتر گوسی ،تصویر فیلتر شده با فیلتر میانگین است برای هرم لاپلاسی مربوطه ان هم از تصاویر فیلتر شده با فیلتر میانگین مانند گوسی استفاده میکنیم که نتایج ان در قسمت شرح نتایج میباشد

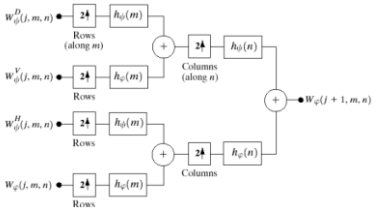
- Wavelet pyramid

در این هرم به تصویر اصلی 4 فیلتر اعمال میشود و سپس سایز ان به نصف کاهش میابد تا قضیه شانون رعایت شده باشد که شامل فیلتر

پایین گذر ،فیلتر بالاگذر در جهت x ،فیلتر بالاگذر در جهت y و فیلتر بالاگذر در جهت قطر ها میباشد که به ترتیب میانگین تصویر،لبه های عمودی،لبه های افقی و لبه های قطری را به ما میدهد و اگر بخواهیم تعداد سطح بیشتری ویولت بزنیم این مراحل فوق را بر قسمت میانگین که از مرحله قبل بدست امد دوباره انجام می دهیم .نکته جالب این هرم این است که سایز مجموع تمام تصاویر این هرم برابر سایز تصویر اصلی میباشد.



اگر بخواهیم تصویر اصلی را از روی هرم ویولت بدست بیاوریم کافی است از 4 کوچکترین تصایر شروع کرده و انها را upsample کنیم و با هم جمع کنیم و مراحل فوق را تا جایی که به تصویر اصلی برسیم ادامه دهیم



همه مراحل فوق توسط تابع dwt2وidwt2 انجام میشود

- Quantize wavelet coefficients

برای عمل کوانتایز کردن ضرایب ویولت کافی است تصاویر خروجی که dwt2 میدهد را کوانتایز کرده و برای رسیدن به تصویر اصلی به idwt2 بدهیم که فرمول کوانتایز کردن هر تصویر به صورت زیر میباشد

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

c represents the wavelet coefficient

برای اینکه چه پیکسلی به چه پیکسلی در هرم گوسی مپ میشود باید گفت که فرض کنید ما از اخرین سطح شروع کنیم که یک پیکسل است و میانگین 4 پیکسل سطح بالاتر ان میباشد و 4 پیکسل سطح بالاتر ان هر کدام خود میانگین 4 پیکسل همسایه سطح بالاتر از ان میباشد این چرخه همچنان ادامه دارد تا بالاترین سطح که به صورت بازگشتی محاسبه میشود

- **Denoising**

برای عمل حذف نویز بر پایه ویولت دو روش پیدا کردم از یک مقاله ای که در قسمت مرجع میباشد یکی این که ما بیایم بر روی ضرایب ویولت ترشولد بزنیم و از یک مقداری پایین را صفر قرار دهیم وبه بالاتر از ان مقدار کاری نداشته باشیم که بدست اوردن ترشولد تجربی میباشد

روش دیگر حذف نویز که بر پایه ویولت پیدا کردم فیلتر کردن بر روی ضرایب ویولت میباشد که سه فیلتر میانگین و میانه و گوسی بر روی ضرایب در قسمت شرح نتایج اعمال شده است که توضیح فیلتر ها در تمرین های قبلی توضیح دادم

3-شرح نتایج

- **Pyramid**
- 6.1.1

برای درست کردن هرم گوسی اگر خود تصویر اصلی را جزء سطح ها حساب کنیم بیشترین تعداد سطح ممکن که میتوان در هرم گوسی داشت برابر با j+1 که 2بتوان j اندازه تصویر می باشد. همینطور سایز کل هرم برابر با مجموع کل

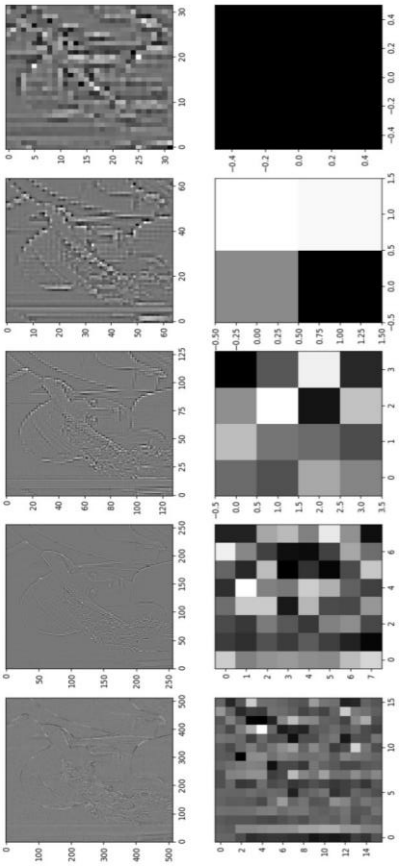
هرم که برابر با دوبرابر سایز تصویر اصلی منهای یک ((2^j+1)-1) میباشد

تصویر زیر هرم گوسی تصویر لنا میباشد که با خود تابع گوسی تا کوچکترین سطح ممکن که 10 میباشد انجام شده است

همانطور که مشاهده میشود تقریب تصویر در scale های مختلف در این هرم موجود میباشد که کوچکترین تصویر ان که سایز یک دارد برابر با میانگین تصویر میباشد همزمان که سایز تصویر کوچک میشود تصویر تیز نشده است چرا که فیلتر گوسی قبل از ان انجام شده است و بیشترین سطح ممکن برابر با j+1 که 2^j سایز تصویر میباشد و سایز کل هرم دوبرابر سایز تصویر منهای یک میباشد

تصویر زیر هرم لاپلاسی تصویر لنا میباشد که با تفاضل تابع گوسی و تصویر تا کوچکترین سطح ممکن که 10 میباشد انجام شده است

که برای اینکه جزییات بیشتر مشاهده شود به صورت زیر در امده است



همانطور که مشاهده میشود جزئیات تصویر در scale های مختلف در این هرم مشاهد میشود. کوچکترین تصویر این هرم برابر با کوچکترین تصویر هرم گوسی که میانگین تصویر است میباشد سایز این هرم و تعداد سطوح ممکن ان نیز برابر با هرم گوسی میباشد .اما سودمندی این روش این است که در هنگام ذخیره کردن میتوان از این روش به عنوان فشرده کردن عکس استفاده کرد و با حجم کمتری تصویر را ذخیره کرد.چرا که میتوان تصویر اصلی را با داشتن کوچکترین تصویر مناسب از هرم گوسی و همینطور کل هرم لاپلاسی در حد خوبی تقریب زد و بازسازی کردو از انجا که هرم لاپلاسی دارای مقادیر صفر زیادی میباشد که به ان ماتریس sparse میگویند میتوان ان را با حجم کمتری بیان و ذخیره کرد.که بازسازی تصویر اصلی از روی هرم لاپلاسی و کوچکترین تصویر گوسی در شرح تکنیکال بیان شده است .

تصویر زیر تصویر اصلی لنا میباشد

و تصویر زیر تصویر بازسازی شده از هرم لاپلاسی و گوسی ان میباشد

همانطور که با چشم هم مشاهده میشود تفاوتی بین ان دو نمیبینم برای اطمینان بیشتر بین ان دو mmse گرفتیم که برابر با صفر بود پس تصویر کاملا بازسازی شده است چرا که ما کاملا تمام اطلاعات حذف شده در downsampling را در هرم گوسی نگه میداریم

(برای اینکه چه پیکسلی به چه پیکسلی در هرم گوسی مپ میشود باید گفت که فرض کنید ما از اخرین سطح شروع کنیم که یک پیکسل است و میانگین 4 پیکسل سطح بالاتر ان میباشد و 4 پیکسل سطح بالاتر ان هر کدام خود میانگین 4 پیکسل همسایه سطح بالاتر از ان میباشد این چرخه همچنان ادامه دارد تا بالاترین سطح که به صورت بازگشتی محاسبه میشود)

- 6.1.2

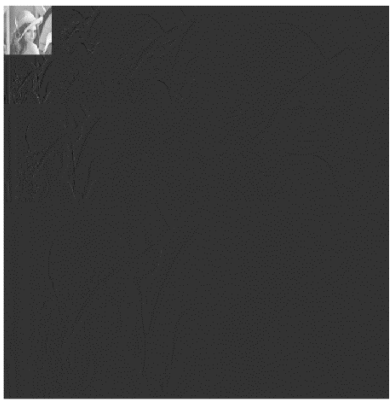
تصویر زیر هرم approximation میباشد که با فیلتر میانگین 2*2 و upsample با تکرار پیسکل میباشد که نحوه ی ان در شرح تکنیکال میباشد

این هرم دارای 3 سطح است.که تصاویر نسبت به حالت گوسی بلوری تر میباشد.

هرم زیر هرم لاپلاسی مربوط به مرحله قبل میباشد

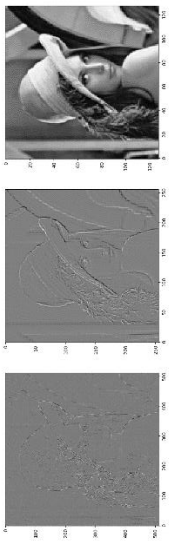
- 6.1.3

هرم زیر ویولت تصویر لنا میباشد

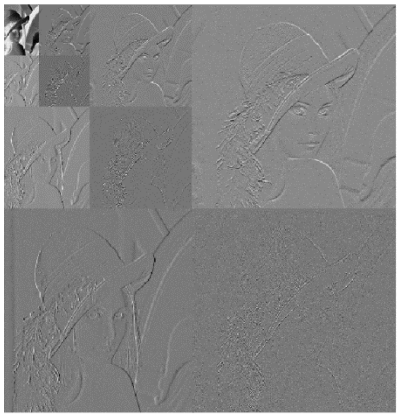


برای انیکه واضح تر شود با اسکیل کردن به شکل زیر درامده است

برای مشاهده بهتر جزییات ان را اسکیل کرده ایم



همانطور که مشاهده میشود کوچکترین تصویر ان به کوچکترین تصویر هرم گوسی یکی میباشد و قسمت هایی که در میانگین گیری و کوچک کردن از تصویر حذف شده است را در بر دارد که میتوانیم با داشتن تصویر سطح 3 هرم گوسی و کل هرم گوسی تصویر اصلی را تقریت خوبی بازسازی کنیم.



همانطور که مشاهده میشود در این هرم لبه های عمودی افقی و قطری در اسکیل های مختلف میباشد و همینطور تقریب تصویر در گوشه سمت چپ بالا میباشد. و در فرکانس های بالا که سه تصویر بزرگ میباشد رزلوشن مکانی خوب داریم و در فرکانس های پایین که (تصاویر کوچک) رزلوشن فرکانسی خوب داریم و دقیق تر است چرا که چند بار ویولت بر قسمت های فرکانس پایین زده ایم.

تفاوت بین هرم گوسی و لاپلاسی و هرم موجک:

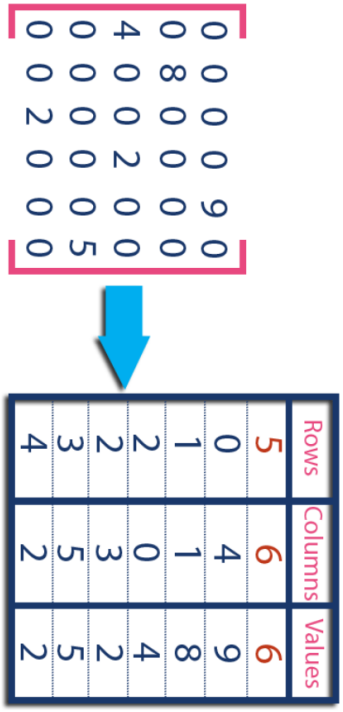
در هرم گوسی یک شی در تصویر را در اسکیل های مکانی مختلف داریم ولی در هرم موجک فقط در پایین ترین سطح شی در تصویر را به صورت فیزیکی داریم و اگر بخواهیم تقریب های سطح های دیگر را داشته باشیم مجبوریم ان را از روی سطح قبل از ان بسازیم و همینطور سایز کل هرم گوسی دو برابر تصویر میباشد و جزییات تصویر را نداریم در حالی که کل هرم موجک برابر سایز تصویر میباشد و جزییات تصویر در رزلوشن های فرکانسی و زمانی مختلف داریم.

در هرم لاپلاسی جزییات تصویر را در یک تصویر در اسکیل های مختلف داریم و تقریب تصویر را مانند هرم موجک در پایین ترین سطح فقط داریم و اگر بخواهیم در سطح های دیگر داشته باشیم باید ان را با سطح های پایین تر بسازیم ولی سایز هرم لاپلاسی دوبرابر سایز تصویر درحالی که سایز هرم موجک اندازه تصویر میباشد اما باید به این نکته توجه کرد که هرم موجک و هرم لاپلاسی را میتوان در حجم کمتر ذخیره کرد چرا که به شکل ماتریس sparse میباشد.

تصویر زیر تصویر بازسازی شده از ویولت تصویر لنا میباشد

همانطور که مشاهده میشود با تصویر اصلی تفاوتی ندارد که mmse ان با تصویر اصلی نیز صفر است پس تصویر کاملا با این روش بازسازی شده است و دلیل این دقت این است که جزییاتی که در downsample از دست میدادیم را همرو ذخیره کردیم.

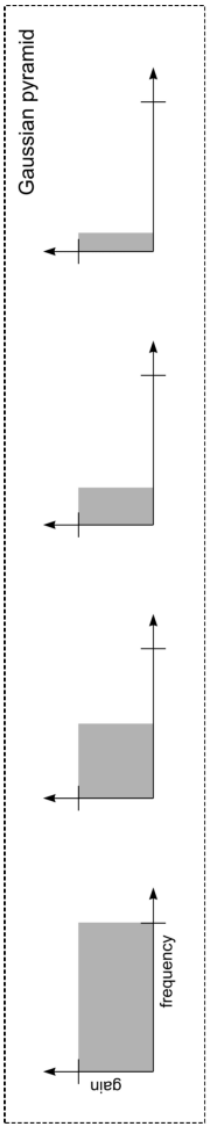
تفاوتی دیگر که بین هرم گوسی و هرم ویولت میباشد این است که در هرم گوسی بعضی جزییات را تکراری داریم در حالی که در هرم ویولت جزییات ذخیره شده در هرم با هم همپوشانی ندارند و تکراری نمیباشد به تصویر زیر برای دید بهتر دقت کنید



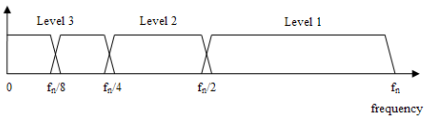
6.1.4

تصویر زیر هرم ویولت تصویر لنا میباشد که quantize شده است

که چون ضرایب با استپ مساوی 2 کوانتایز شدن پس میتوان با یک بیت کم تر ان ها را ذخیره کرد و تعداد سطح به نصف کاهش پیدا کرد پس علاوه بر اسپارس بودن ان که سایز ذخیده سازی را کم میکرد ما باز هم با کوانتایز کردن سایز ذخیره سازی را کمتر کردیم که این عمل کوانتایز کردن موجب خطا در تصویر میشود برای دید بهتر این موضوع تصویر را از روی این هرم دوباره بازسازی کردیم و psnr ان را با تصویر اصلی اندازه گرفتیم که تصویر بازسازی شده در زیر امده است



و تصویر زیر که مربوط به ویولت میباشد که همپوشانی بین فرکانسها نزدیک به صفر میباشد



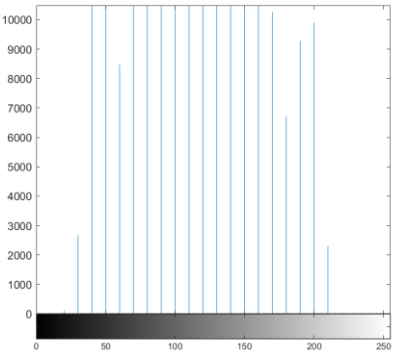
تصویر زیر نمایی از بخشی از یک ماتریس اسپارس که 30 عضو داشته است که 24 عضو ان صفر است میباشدکه فشرده شده به 27 عضو است

که در کل تغییری در تصویر مشاهده نمیشود اما اگر در ان زوم شود تغییرات را بیشتر حس میکنیم

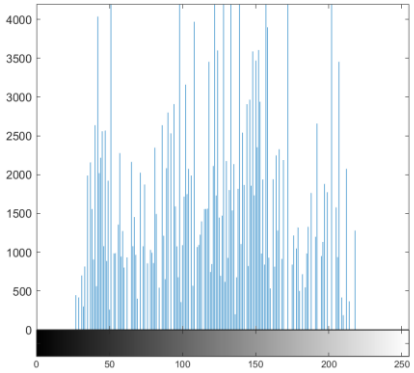
مثال به لبه های صورت لنا در زیر دقت کنید که خراب شده است و ان را با تصویر اصلی که در زیر ان امده است مقایسه کنید

همینطور psnr ان با تصویر اصلی برابر با 47.54 شد که نشان میدهد نسبت به نویز مقاوم است.

تصویر زیر هیستوگرام تصویر کوانتایز شده با استفاده از الگوریتم کوانتایز ضرایب ویولت با gamma=80 که مشاهد میشود با وجود step size = 80 در مقادیر در تصویر بازسازی شده از هم فاصله 80 ندارند چرا که ان گاما مساوی 80 در ضرایب ویولت انجام شد که psnr = 26



و این هم هیتسوگرام خود تصویر اصلی میباشد که جهت مقایسه با هیستوگرام بالا گذاشته شده است



همینطور یک کوانتایز بر روی خود تصویر اصلی نه ضرایب ویولت ان با gamma = 80 انجام دادیم که psnr برابر با 20 شد که این نشان میدهد که با وجود این که تعداد بیت های کم شده در هر دو روش یکی بود ولی کوانتایز در ضرایب ویولت psnr بهتری دارد و نسبت به نویز مقاوم تر است.

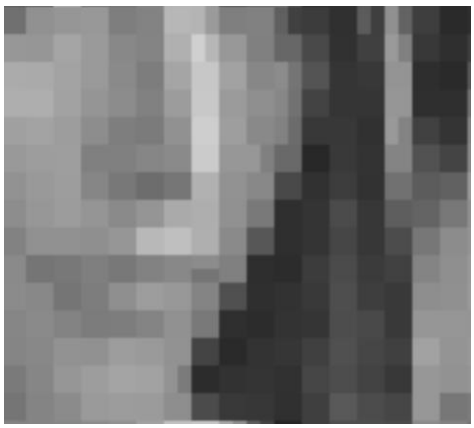
- **Denoising**

تصویر زیر تصویر اصلی لنا میباشد

تصویر زیر تصویر لنا میباشد که نویز گوسی به ان اعمال شده است

تصویر زیر تصویری است که بر روی ضرایب ویولت با 3 سطح ان ترشولد گرفته شده است

مشاهده میشود که نویز های در تصویر از بین رفته است ولی لبه ها و جزییات تصویر نیز از بین رفته اند به تصویر زیر نگاه بیاندازید که بر روی تصویر فوق زوم شده است



مشاهده میشود که که لبه ها از بین رفته اند و ان هم به خاطر این است که جزییاتی که زیر ترشولد بوده اند در عمل ترشولد صفر شده اند و عملا این تصویر بزرگ شده ی کوچکترین تصویر LL در هرم ویولت میباشد که جزییات کمی به ان اضافه شده است پس نتیجه میگیریم که عمل ترشولد علاوه بر نویز جزییات تصویر را نیز از بین میبرد

جدول زیر عمل دینویز کردن را تا ویولت ها ی سطح های مختلف نشان میدهد که بهترین سطح 2 میباشد و دلیل اینکه در یک سطح خاص بهترین نتیجه را میدهد بدلیل این است که فیلتر بالاگذری که در محاسبه ویولت زده میشود در سطحی خاص نویزها وجود دارد و ان فیلتر ان زا تشخیص میدهد

جدول زیر مربوط به median filter با سطح های مختلف میباشد

	psnr_col	mmse_col
1	24.8933	210.7428
2	25.5988	179.1421
3	23.3905	297.8733
4	20.8684	532.4043
5	18.8690	843.6916
6	16.8464	1.3441e+03
7	15.4540	1.8522e+03
8	15.0083	2.0524e+03
9	14.4654	2.3256e+03

همانطور که مشاهده میشود تا 2 سطح بهترین mmse را دارد چون در اون scale فیلتر میانه نویز بیشتری حذف میکند برای اطمینان تصویر دینویز شده با 2 سطح ویولت را در زیر میبینند که نسبت به 3 سطح که در بالا نشان داده شد بهتر است

	psnr_col	mmse_col
1	24.8811	211.3325
2	26.1095	159.2690
3	24.8453	213.0826
4	22.9905	326.6096
5	21.1754	496.0628
6	19.1178	796.7024
7	18.2663	969.2733
8	17.7875	1.0823e+03
9	17.1327	1.2584e+03

که تصویر دینویز شده سطح 2 با عمل ترشولد را در زیر میبینید

مشاهده میشود که کیفیت بسیار بهتری نسبت به 3 سطح که در عکس قبلی بررسی شد دارد تصویر زیر تصویر denoise شده با اعمال فیلتر medianبر روی ضرایب ویولت با 3 سطح میباشد

همانطور که مشاهده میشود تصویر جزییات خود را از دست داده است

مشاهده میشود که لبه های واقعی تصویر کمتر نسبت به حالات دیگر حذف شده است و نویز بیشتر حذف شده است

همانطور که مشاهده میشود عمل ترشولد کردن برای نویز گوسی از فیلتر میانه بهتر است دلیل ان هم این است که فیلتر میانه بر روی جزییات چون بیشتر مقادیر این ماتریس صفر است نمیتواند اطلاعات اماری بسیار خوبی بدهد که بتوان همسایه های ان را حدس زد

اما فیلتر میانه و ترشولد بر روی نویز نمک فلل را مشاهده کنیم

با فیلتر میانه

	psnr_col	mmse_col
1	23.4918	291.0022
2	25.1488	198.6997
3	23.3458	300.9530
4	20.9067	527.7232

با ترشولد

	psnr_col	mmse_col
1	21.5755	452.4093
2	22.6427	353.8397
3	22.1732	394.2379
4	21.0965	505.1578

مشاهده میشود که فیلتر میانه خیلی بهتر است از ترشولد نسبت به نویز نمک و فلل که چون اطلاعات اماری خوبی میتوان از تصویر نویزی در جزییات ویولت ان گرفت

فیلترهای میانگین و گوسی نیز بر روی ضرایب تصویر نویزی اعمال شده است که در زیر خروجی ها و جداول ان را مشاهده میکنید

جدول فیلتر میانگین

	psnr_col	mmse_col
1	24.9127	209.8021
2	25.6419	177.3750
3	23.4190	295.9208
4	20.8838	530.5137

همانطور که مشاهده میشود فیلتر میانگین توانسته نویز بیشتری نسبت به روش های قبلی حذف کند و اطلاعات اماری بهتری از ضرایب در اورده است

تصویر مربوطه به فیلتر میانگین که توضیحات ان داده شد

جدول فیلتر گوسی

	psnr_col	mmse_col
1	24.9137	209.7552
2	25.6627	176.5256
3	23.4484	293.9257
4	20.8837	530.5256

مشاهده میشود که فیلتر گوسی کمی نیز بهتر از میانگین عمل کرده است و اطلاعات اماری بهتری از ضرایب دریافت کرده است

عمل ترشودگرفتن خیلی سریع تر است از عمل فیلتر کردن چرا که فقط کافی است بر روی تمام عکس یکبار حلقه زده شود مرتبه زمانی n2 دارد اما مرتبه زمانی فیلتر کردن s2*n2 میباشد که s سایز فیلتر میباشد اما عمل ترشولد کردن فقط بر روی ضرایب ویولت قابلیت حذف نویز را دارد ولی فیلتر میانه قابلیت حذف نویز از روی تصویر اصلی را نیز دارد .

4-کد برنامه

کد قسمت 6.1.1

```
import cv2
import matplotlib.pyplot as plt
```

```
    pyramid_list.append(imgs[i])
    return pyramid_list


def
construct_img(laplacian_pyramid,gaus
sian):
    up_smple =
copy.deepcopy(gaussian)
    for i in
range(len(laplacian_pyramid)-2,-1,-
1):
        up_smple =
upsmple_pRreplication(up_smple)
        up_smple = up_smple +
laplacian_pyramid[i]
    return up_smple


def run(name):
    img =
cv2.imread('../image/'+name)
    img =
cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)

cv2.imwrite('../image/hw6.1.1/img.pn
g',img)
    img = img.astype(np.float)
    level = 9
    mode = 'gaussian'
    gaussian_pyramid =
get_approximation_pyramid(img,level,
mode)
    g_pyramid_im =
np.ones((img.shape[0],2*img.shape[1]
))*255
    frm = 0
    fig,axes =
plt.subplots(2,5,figsize=(20,8))
    for i,im in
enumerate(gaussian_pyramid):

axes[i//5,i%5].imshow(im,cmap='gray'
)

g_pyramid_im[0:im.shape[0],frm:frm+i
m.shape[1]] = im
        frm += im.shape[0]

fig.savefig('../image/hw6.1.1/approx
imation_pyramid({}).png'.format(mode
))

cv2.imwrite('../image/hw6.1.1/approx
imation_pyramid_im({}).jpg'.format(m
ode),g_pyramid_im)

    laplace_pyramid =
get_laplace_pyramid(gaussian_pyramid
)
    l_pyramid_im =
np.ones((img.shape[0],2*img.shape[1]
))*255
    frm = 0
```

```
import numpy as np
import copy

def PSNR(im1,im2):
    mse = np.mean((im1 - im2) ** 2)
    if(mse == 0):  # MSE is zero
means no noise is present in the
signal .
        # Therefore PSNR have no
importance.
        return np.inf,0
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel /
np.sqrt(mse))
    return psnr,mse

def
upsmple_pRreplication(dwn_smpl_img):
    width = dwn_smpl_img.shape[1]*2
    height = dwn_smpl_img.shape[0]*2
    up_smpl_img_pRplctn =
np.ones((height,width))
    for i in range(0,height):
        for j in range(0,width):
            up_smpl_img_pRplctn[i,
j] = dwn_smpl_img[int(i/2),
int(j/2)]

    return up_smpl_img_pRplctn


def get_blur(img,mode='gaussian'):
    ksize = (5,5)
    return
cv2.GaussianBlur(img,ksize,.1) if
mode == 'gaussian' else
cv2.blur(img,ksize)


def
get_approximation_pyramid(img,level,
mode='gaussian'):
    blur = get_blur(img,mode)
    blur =
cv2.resize(img,(int(blur.shape[0]/2)
,int(blur.shape[1]/2)))
    pyramid_list = [img,blur]
    for i in range(level-1):
        blur = get_blur(blur,mode)
        blur =
cv2.resize(blur,(int(blur.shape[0]/2
),int(blur.shape[1]/2)))
        pyramid_list.append(blur)
    return pyramid_list


def get_laplace_pyramid(imgs):
    pyramid_list = []
    for i in range(1,len(imgs)):
        upsample =
upsmple_pRreplication(imgs[i])
        laplace = imgs[i-1] -
upsample
        pyramid_list.append(laplace)
```

```
def
get_approximation_pyramid(img,level,
mode='gaussian'):
    blur = get_blur(img,mode)
    blur =
cv2.resize(img,(int(blur.shape[0]/2)
,int(blur.shape[1]/2)))
    pyramid_list = [img,blur]
    for i in range(level-1):
        blur = get_blur(blur,mode)
        blur =
cv2.resize(blur,(int(blur.shape[0]/2
),int(blur.shape[1]/2)))
        pyramid_list.append(blur)
    return pyramid_list


def get_laplace_pyramid(imgs):
    pyramid_list = []
    for i in range(1,len(imgs)):
        upsample =
upsmple_pRreplication(imgs[i])
        laplace = imgs[i-1] -
upsample
        pyramid_list.append(laplace)
    pyramid_list.append(imgs[i])
    return pyramid_list


def run(name):
    img =
cv2.imread('../image/'+name)
    img =
cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    img = img.astype(np.float)
    level = 2
    mode = 'average'
    gaussian_pyramid =
get_approximation_pyramid(img,level,
mode)
    g_pyramid_im =
np.ones((img.shape[0],2*img.shape[1]
))*255
    frm = 0
    fig,axes =
plt.subplots(1,4,figsize=(20,8))
    for i,im in
enumerate(gaussian_pyramid):

axes[i].imshow(im,cmap='gray')

g_pyramid_im[0:im.shape[0],frm:frm+i
m.shape[1]] = im
        frm += im.shape[0]

fig.savefig('../image/hw6.1.2/approx
imation_pyramid({}).png'.format(mode
))

cv2.imwrite('../image/hw6.1.2/approx
imation_pyramid_im({}).jpg'.format(m
ode),g_pyramid_im)
```

```
    fig,axes =
plt.subplots(2,5,figsize=(20,8))
    for i,im in
enumerate(laplace_pyramid):

l_pyramid_im[0:im.shape[0],frm:frm+i
m.shape[1]] = im

axes[i//5,i%5].imshow(im,cmap='gray'
)
        frm += im.shape[0]

fig.savefig('../image/hw6.1.1/laplac
e_pyramid({}).png'.format(mode))

cv2.imwrite('../image/hw6.1.1/laplac
e_pyramid_im({}).jpg'.format(mode),l
_pyramid_im)

    reconstruct_im =
construct_img(laplace_pyramid,gaussi
an_pyramid[len(gaussian_pyramid)-1])

cv2.imwrite('../image/hw6.1.1/recons
truct_im({}).png'.format(mode),recon
struct_im)
    psnr,mmse = PSNR(img,
reconstruct_im)

    print("mmse={}".format(mmse))
    print("psnr={}".format(psnr))


run('Lena.bmp')
```

کد قسمت 6.1.2

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from src.hw6_1_1 import *

def
upsmple_pRreplication(dwn_smpl_img):
    width = dwn_smpl_img.shape[1]*2
    height = dwn_smpl_img.shape[0]*2
    up_smpl_img_pRplctn =
np.ones((height,width))
    for i in range(0,height):
        for j in range(0,width):
            up_smpl_img_pRplctn[i,
j] = dwn_smpl_img[int(i/2),
int(j/2)]

    return up_smpl_img_pRplctn


def get_blur(img,mode='gaussian'):
    ksize = (2,2)
    return
cv2.GaussianBlur(img,ksize,.1) if
mode == 'gaussian' else
cv2.blur(img,ksize)
```

```
    laplace_pyramid =
get_laplace_pyramid(gaussian_pyramid
)
    l_pyramid_im =
np.ones((img.shape[0],2*img.shape[1]
))*255
    frm = 0
    fig,axes =
plt.subplots(1,3,figsize=(20,8))
    for i,im in
enumerate(laplace_pyramid):

l_pyramid_im[0:im.shape[0],frm:frm+i
m.shape[1]] = im

axes[i].imshow(im,cmap='gray')
        frm += im.shape[0]

fig.savefig('../image/hw6.1.2/laplac
e_pyramid({}).png'.format(mode))

cv2.imwrite('../image/hw6.1.2/laplac
e_pyramid_im({}).jpg'.format(mode),l
_pyramid_im)

    reconstruct_im =
construct_img(laplace_pyramid,gaussi
an_pyramid[len(gaussian_pyramid)-1])

cv2.imwrite('../image/hw6.1.2/recons
truct_im({}).jpg'.format(mode),recon
struct_im)
    psnr,mmse = PSNR(img,
reconstruct_im)
    print("mmse={}".format(mmse))
    print("psnr={}".format(psnr))

run('Lena.bmp')
```

کد قسمت 6.1.3

```
im = imread('../image/Lena.bmp');
im = rgb2gray(im);
im = double(im);

N = 3;
w = [];
f = figure(1);
A = im;
for i=1:N
    [A,H,V,D] = dwt2(A,'haar');
    w{4*(i-1)+1} = A;
    w{4*(i-1)+2} = H;
    w{4*(i-1)+3} = V;
    w{4*(i-1)+4} = D;
end
res = [[[w{4*(3-1)+1},w{4*(3-
1)+2};w{4*(3-1)+3},w{4*(3-
1)+4}],w{4*(2-1)+2};w{4*(2-
1)+3},w{4*(2-1)+4}],w{4*(1-
1)+2};w{4*(1-1)+3},w{4*(1-1)+4}];
imshow( res,[]);
```

```
saveas(f,'../image/hw6.1.3/dwt2_im',
'png');


%reconstruct image
A = w{4*(3-1)+1};
for i=N:-1:1
    A = idwt2(A,w{4*(i-1)+2},w{4*(i-
1)+3},w{4*(i-1)+4},'haar');
end
f = figure(1);
A = uint8(A);
im = uint8(im);
imshow(A);
saveas(f,'../image/hw6.1.3/idwt2','p
ng');
immse(A,im)


f2 = figure(1);
A = double(im);
for i=1:N
    [A,H,V,D] = dwt2(A,'haar');
    w{4*(i-1)+1} =
normalizeimg(A,max(A(:)),min(A(:)));
    w{4*(i-1)+2} =
normalizeimg(H,max(H(:)),min(H(:)));
    w{4*(i-1)+3} =
normalizeimg(V,max(V(:)),min(V(:)));
    w{4*(i-1)+4} =
normalizeimg(D,max(D(:)),min(D(:)));
end
res = [[[w{4*(3-1)+1},w{4*(3-
1)+2};w{4*(3-1)+3},w{4*(3-
1)+4}],w{4*(2-1)+2};w{4*(2-
1)+3},w{4*(2-1)+4}],w{4*(1-
1)+2};w{4*(1-1)+3},w{4*(1-1)+4}];
imshow( res,[]);
saveas(f2,'../image/hw6.1.3/dwt2_im_
NORM','png');



f = figure(1);
for i= 1:N*4
    subplot(N,4,i), imagesc(w{i});
    colormap gray(255);
end
saveas(f,'../image/hw6.1.3/dwt2','pn
g');
```

کد قسمت 6.1.4

```
im = imread('../image/Lena.bmp');
im = rgb2gray(im);
imwrite(im,'../image/hw6.1.4/im.png'
);
im = double(im);
gamma = 2;
N = 3;
```

```
imhist(A(:,:));
saveas(f,'../image/hw6.1.4/QHIST','p
ng');

f = figure(1);
q_im = quantize_im(im,gamma);
imshow(q_im);
saveas(f,'../image/hw6.1.4/q_im','pn
g');
psnr(im,q_im)
```

کد قسمت 6.2

```
im = imread('../image/Lena.bmp');
im = rgb2gray(im);
imwrite(im,'../image/hw6.2/im.png');
noisy_im =
imnoise(im,'gaussian',0,.01);
denoise_mode = "median_filter";
gamma_thresh = 85;
%just for threshold is this
ksize = [7,7];
%just for median filter is this
imwrite(noisy_im,'../image/hw6.2/noi
sy_im.png');
imshow(noisy_im);
pause
Num_wavelet_level = 4;
show_recon_level = 2;
w = [];
data_table = [];
psnr_col = [];
mmse_col = [];
levels = [];
for N=1:Num_wavelet_level
    levels = [levels;string(N)];
    %wavelet transform
    f = figure(1);
    A = noisy_im;
    for i=1:N
        [A,H,V,D] = dwt2(A,'haar');
        w{4*(i-1)+1} = A;
        w{4*(i-1)+2} = H;
        w{4*(i-1)+3} = V;
        w{4*(i-1)+4} = D;
    end
    res = w{4*(N-1)+1};
    for i=N:-1:1
        res = [res,w{4*(i-
1)+2};w{4*(i-1)+3},w{4*(i-1)+4}];
    end
    imshow( res,[]);

saveas(f,'../image/hw6.2/dwt2','png'
);

    %median filter
    if denoise_mode ==
"median_filter"
        for i=1:N
        %     w{4*(i-1)+1} =
medfilt2(w{4*(i-1)+1},ksize);
```

```
w = [];
f = figure(1);
A = im;
for i=1:N
    [A,H,V,D] = dwt2(A,'haar');
    w{4*(i-1)+1} = A;
    w{4*(i-1)+2} = H;
    w{4*(i-1)+3} = V;
    w{4*(i-1)+4} = D;
end
res = [[[w{4*(3-1)+1},w{4*(3-
1)+2};w{4*(3-1)+3},w{4*(3-
1)+4}],w{4*(2-1)+2};w{4*(2-
1)+3},w{4*(2-1)+4}],w{4*(1-
1)+2};w{4*(1-1)+3},w{4*(1-1)+4}];
imshow( res,[]);
saveas(f,'../image/hw6.1.4/dwt2','pn
g');


for i=1:N
    w{4*(i-1)+1} =
quantize_wavelet_coeff(w{4*(i-
1)+1},gamma);
    w{4*(i-1)+2} =
quantize_wavelet_coeff(w{4*(i-
1)+2},gamma);
    w{4*(i-1)+3} =
quantize_wavelet_coeff(w{4*(i-
1)+3},gamma);
    w{4*(i-1)+4} =
quantize_wavelet_coeff(w{4*(i-
1)+4},gamma);
end
f = figure(1);
res = [[[w{4*(3-1)+1},w{4*(3-
1)+2};w{4*(3-1)+3},w{4*(3-
1)+4}],w{4*(2-1)+2};w{4*(2-
1)+3},w{4*(2-1)+4}],w{4*(1-
1)+2};w{4*(1-1)+3},w{4*(1-1)+4}];
imshow( res,[]);
saveas(f,'../image/hw6.1.4/dwt2_quan
tize','png');

A = w{4*(3-1)+1};
for i=N:-1:1
    A = idwt2(A,w{4*(i-1)+2},w{4*(i-
1)+3},w{4*(i-1)+4},'haar');
end
f = figure(1);
A = uint8(A);
imshow(A);
saveas(f,'../image/hw6.1.4/idwt2','p
ng');
im = uint8(im);
psnr(im,A)


f = figure(1);
imhist(im(:,:));
saveas(f,'../image/hw6.1.4/IMHIST','
png');

f = figure(1);
```

```
        imshow( res,[]);

saveas(f,'../image/hw6.2/dwt2_avg','
png');
    end
    %gaussian_filter
    if denoise_mode ==
"gaussian_filter"
        sigma = 8;
        for i=1:N
        %     w{4*(i-1)+1} =
imgaussfilt(w{4*(i-1)+1},sigma);
            w{4*(i-1)+2} =
imgaussfilt(w{4*(i-1)+2},sigma);
            w{4*(i-1)+3} =
imgaussfilt(w{4*(i-1)+3},sigma);
            w{4*(i-1)+4} =
imgaussfilt(w{4*(i-1)+4},sigma);
        end
        f = figure(1);
        res = w{4*(N-1)+1};
        for i=N:-1:1
            res = [res,w{4*(i-
1)+2};w{4*(i-1)+3},w{4*(i-1)+4}];
        end
        imshow( res,[]);

saveas(f,'../image/hw6.2/dwt2_gaussi
an','png');
    end


    %reconstruct
    A = w{4*(N-1)+1};
    for i=N:-1:1
        A = idwt2(A,w{4*(i-
1)+2},w{4*(i-1)+3},w{4*(i-
1)+4},'haar');
    end
    A = uint8(A);
    im = uint8(im);
    if show_recon_level == N
        f = figure(1);
        imshow(A);

saveas(f,strcat('../image/hw6.2/idwt
2',denoise_mode),'png');
    end
    psnr_col =
[psnr_col(:);psnr(im,A)];
    mmse_col =
[mmse_col(:);immse(im,A)];

end


% report mmse table
T =
table(psnr_col,mmse_col,'RowNames',l
evels);
f = figure(1);
uitable('Data',T{:,:},'ColumnName',T
.Properties.VariableNames,...
```

```
            w{4*(i-1)+2} =
medfilt2(w{4*(i-1)+2},ksize);
            w{4*(i-1)+3} =
medfilt2(w{4*(i-1)+3},ksize);
            w{4*(i-1)+4} =
medfilt2(w{4*(i-1)+4},ksize);
        end
        f = figure(1);
        res = w{4*(N-1)+1};
        for i=N:-1:1
            res = [res,w{4*(i-
1)+2};w{4*(i-1)+3},w{4*(i-1)+4}];
        end
        imshow( res,[]);

saveas(f,'../image/hw6.2/dwt2_median
','png');
    end

    %threshold
    if denoise_mode == "threshold"
        for i=1:N
        %     w{4*(i-1)+1} =
threshold(w{4*(i-1)+1},gamma);
            w{4*(i-1)+2} =
threshold(w{4*(i-1)+2},gamma);
            w{4*(i-1)+3} =
threshold(w{4*(i-1)+3},gamma);
            w{4*(i-1)+4} =
threshold(w{4*(i-1)+4},gamma);
        end
        f = figure(1);
        res = w{4*(N-1)+1};
        for i=N:-1:1
            res = [res,w{4*(i-
1)+2};w{4*(i-1)+3},w{4*(i-1)+4}];
        end
        imshow( res,[]);

saveas(f,'../image/hw6.2/dwt2_thr','
png');
    end

    %avg filter
    if denoise_mode == "avg_filter"
        windowSize = 25;
        kernel = ones(windowSize) /
windowSize ^ 2;
        for i=1:N
%             w{4*(i-1)+1} =
avg_filter(w{4*(i-1)+1},kernel);
            w{4*(i-1)+2} =
avg_filter(w{4*(i-1)+2},kernel);
            w{4*(i-1)+3} =
avg_filter(w{4*(i-1)+3},kernel);
            w{4*(i-1)+4} =
avg_filter(w{4*(i-1)+4},kernel);
        end
        f = figure(1);
        res = w{4*(N-1)+1};
        for i=N:-1:1
            res = [res,w{4*(i-
1)+2};w{4*(i-1)+3},w{4*(i-1)+4}];
        end
```

```
'RowName',T.Properties.RowNames,'Uni
ts', 'Normalized', 'Position',[0, 0,
1, 1]);
saveas(f,strcat('../image/hw6.2/',de
noise_mode,'_table'),'png');
```

کد مربوط به تابع quantize

```
function res =
quantize_wavelet_coeff(c,gamma)
    res = gamma * sign(c) .*
floor(abs(c)/gamma);
end
```

مراجع

https://arxiv.org/pdf/1703.06499

کتاب گنزالس رافائل

اسلایدهای دکتر امیرحسین طاهری نیا

با کمک های حل تمرین درس اقای بلویان و خانم رستمی و اقای کرمی