

تمرین ۵ بینایی مویک

سارا قوام پور

اطلاعات گزارش	چکیده
تاریخ:	در این سری ۵ از تمرینات هدف آشنایی با MultiResolution Processing در حوزه پردازش تصویر میباشد. تمرینات اولیه به باز کردن مفاهیم هرم های گوسین و لاپلاسن و تشکیل عکس اصلی با استفاده از این هرم ها میباشد. تمرینات آخر به بررسی تبدیل مویک و استفادت از آن برای Compression میپردازد.
واژگان کلیدی: هرم گوسین هرم لاپلاسن فشرده سازی تبدیل مویک haar transform	

۱- مقدمه

این داکيومنت شامل توضیحات فنی و تحلیل نتایج و پیاده سازی های تمرین ۵ مویک با زبان پایتون و با استفاده از کتابخانه cv2 میباشد.

۲- توضیحات فنی تمرینات و تحلیل نتایج

۱-۲ تمرین ۱-۱-۵

در این تمرین خواسته شده است تا هرم گوسی و هرم لاپلاسن تا ۵ سطح بر روی عکس مونالیزا اجرا شود. هرم گوسی به این گونه تشکیل میشود که در سطح ۰ خود عکس اصلی قرار دارد و سطح بعدی با **downsample** کردن سطح فعلی به دست می آید.

این **downsample** کردن در هرم گوسی، کانولوشن کرنل گوسی با عکس ورودی یا به عبارتی محاسبه کردن میانگین گوسی (وزن دار) عکس میباشد.

در این سوال هرم گوسی تا ۵ سطح خواسته شده است. چون سطح ۰ عکس اصلی است باید ۵ سطح دیگر با ۵ بار اعمال فیلتر گوسی بر روی سطح قبل اعمال شود تا هرم گوسی ۵ سطح به دست بیاید.

در ابتدا ارایه ای ۳ در ۳ به عنوان تخمین فیلتر گوسی ۳ در ۳ تشکیل میشود به نام **gaussian_kernel**.

برای پیاده سازی هرم گوسی تابع **gaussian_pyramid** نوشته شده است. این تابع عکس ورودی، فیلتر لازم برای **downsample** کردن و تعداد سطوح هرم را دریافت میکند.

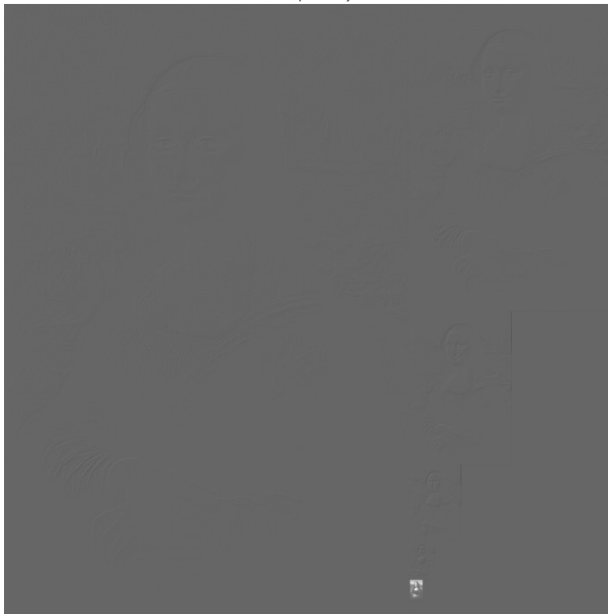
این تابع در یک حلقه به تعداد سطوحی که در ورودی دریافت کرد، کرنل گوسی را با سطح پیشین کانوالو میکند تا به سطح فعلی برسد.

چون در هر **Iteration** کرنل گوسی را با عکس کانوالو میکند و به عبارتی فرکانس عکس کاهش میابد باید طبق قضیه شانون پس از اعمال فیلتر نرخ نمونه برداری نیز کاهش پیدا کند و به این منظور از تابع **remove_row_col_downsampling** استفاده شده است. اگر میخواهیم عکس با نرخ ۲ **downsample** شود یعنی سائز آن در هر ۲ بعد نصف شود به این تابع ورودی را ۲ میدهم. پس از اعمال فیلتر و **downsaple** کردن با حذف ستون ها و ردیف ها، نتیجه به دست آمده حاصل هرم گوسی در این سطح میباشد و حلقه به همین صورت تا آخرین لایه ادامه پیدا میکند.

به منظور نمایش هرم به شیوه که در اسلایدها بیان شده است، یک آرایه **numpy** تمام صفر با ابعاد سطر برابر سطر های عکس تعداد ستون های ۱.۵ برابر عکس ایجاد میکنیم و هر یک از سطح های هرم را در ایندکس مربوطه آن قرار میدهم. تابع **guassian_pyramid** به عنوان خروجی عکس که همه سطوح در آن قرار گرفته شده اند و آرایه های شامل همه سطوح را برمیگرداند.

به همین دلیل تابعی با نام `pad_even` تعریف شده که با صفر گذاشت در بعد فرد تصویر، آن را زوج میکند. خروجی تابع `pad_even` بر روی سطح فعلی گوسین و تفریق آن با خروجی `upsample` سطح بعدی گوسین با فاکتور ۲، سطح فعلی هرم لاپلاسین را تشکیل میدهد. این مراحل در هر بار تکرار حلقه انجام میشود. به منظور نمایش هرم به شیوه که در اسلایدها بیان شده است، یک آرایه `numpy` تمام صفر با ابعاد سطر برابر سطر های عکس تعداد ستون های ۱.۵ برابر عکس ایجاد میکنیم و هر یک از سطح های هرم را در ایندکس مربوطه آن قرار میدهیم. تابع `laplacian_pyramid` به عنوان خروجی عکس که همه سطوح در آن قرار گرفته شده اند و آرایه های شامل همه سطوح را برمیگرداند.

Five level Laplacian Pyramid



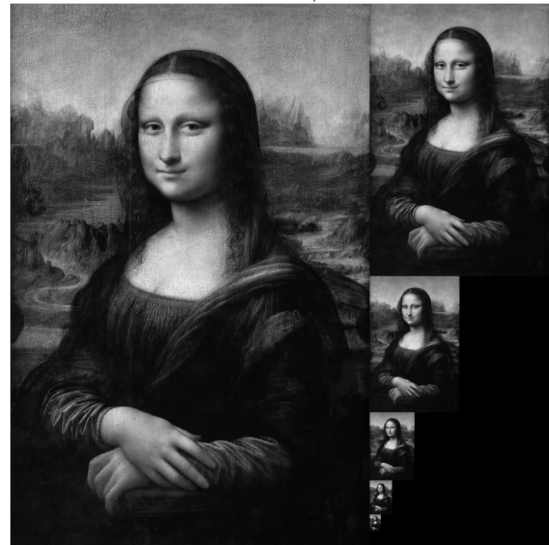
شکل ۲) هرم لاپلاسین با ۵ سطح روی تصویر مونالیزا

شکل ۲ هرم لاپلاسین مونالیزا تا ۵ سطح را نشان میدهد. سطح ۵ (سطح آخر) هرم لاپلاسین با سطح آخر هرم گوسی برابر است. سطح های ۰ تا سطح ۴ در واقع لبه های تصویر در `scale` های متفاوت میباشد.

۲-۲ تمرین ۱-۵

فیلتر جداپذیر یا `separable` که در تمرین های قبلی (سوال ۱ تمرین ۴) هم به آن اشاره شد، فیلتری است که ماتریس ۲ بعدی آن را بتوان به صورت حاصل ضرب یک بردار سطری در بردار ستونی نوشت. به عنوان مثال میتوان فیلتر ۳ در ۳ برای کرنل گوی را مثال زد که حاصل ضرب بردار ستونی و سطری [۱, ۲, ۱] است.

Five level Gaussian Pyramid



شکل ۱) هرم گوسین تصویر مونالیزا ۵ سطح

شکل ۱ هرم گوسین مونالیزا تا ۵ سطح را نشان میدهد. چون خود تصویر اصلی سطح ۰ را تشکیل میدهد در تصویر بالا ۶ عکس وجود دارد که تصویر اصلی در سطح ۰ همان تصویر سمت چپ است. تصاویر سمت راست از بالا به پایین به ترتیب سطح های ۱ تا ۵ هرم را تشکیل میدهند. هر سطح از اعمال گوسین و `downsample` روی سطح قبل به دست می آید.

بخش دیگر این تمرین به پیاده سازی هرم لاپلاسین ۵ سطحی بر روی مونالیزا اختصاص دارد. به منظور پیاده سازی هرم لاپلاسین تابع `laplacian_pyramid` پیاده سازی شده است.

این تابع به عنوان ورودی آرایه شامل تمام سطوح گوسین را دریافت میکند چون هرم لاپلاسین از روی هرم گوسی به دست می آید به این صورت که هر سطح لاپلاسین حاصل تفریق همان سطح هرم گوسی با `upsample` شده با فاکتور ۲ سطح بعدی هرم گوسین میباشد.

ورودی بعدی این تابع تعداد سطوح میباشد. در یک حلقه از ۰ تا تعداد سطح در هر بار تکرار حلقه، ابتدا با دریافت گوسین سطح بعدی از آرایه ورودی، آن را با استفاده از تابع `unsample_px_replication` با فاکتور ۲ `upsample` میکنیم با استفاده از روش `pixel replication` که اندازه تصویر را در فاکتور ضرب میکند و در سطر ها و ستون های خالی مقدار پیکسل قبلی را کپی میکند.

از آنجایی که خروجی این `upsample` باید از گوسین سطح بعد کم شود و در محاسبه هرم گوسین چون `downsample` صورت گرفته است ممکن است ابعاد گوسین فرد باشد و قابل تفریق با طرف دوم تفریق نباشد

اثر آن نیز در بهبود سرعت به جای استفاده یک باره از یک فیلتر بزرگ بحث شد.

۳-۲ تمرین ۳-۱-۵

در هرم ها در هر سطح اندازه تصویر نصف میشود، در نتیجه اگر سایز اولیه تصویر n در n باشد که n برابر است با 2^j بنابراین در هر سطح چون سایز آن تقسیم بر ۲ میشود از j یک مقدار کم میشود. این تقسیم بر ۲ و در نتیجه $j-1$ تا زمانی میتوان ادامه داد که $j=0$. بنابراین میتوان $j+1$ سطح از سطح j ساخت. مجموع تعداد پیکسل هایی که در هرم استفاده میشود از یک رابطه هندسی که در اسلاید شماره ۷ درس هم بیان میشود به دست می آید که به صورت زیر است:

$$N^2 + 1/4 N^2 + 1/16 N^2 + \dots = 4/3 N^2$$

در نتیجه تعداد پیکسل های استفاده شده در هرم تنها $1/3$ بیشتر از تعداد پیکسل های قبلی است. (۳۳ درصد بیشتر)

این فرمول هم برای هرم گوسی و هم برای لاپلاسین درست است.

فواید استفاده از هرم ها که با توجه به اینمه تعداد پیکسل ها ۳۳ درصد بیشتر میشود همچنان مقرون به صرفه و منطقی هستند:

برخی فواید هرم گوسی :

۱- course to fine : به این معنا است که از قسمتی با جزییات کم به شیوه ای تدریجی به قسمتی با جزییات مهم تر برسیم. در هرم ها این عمل به این صورت انجام میشود که در یک سطح با جزییات کم که بررسی شویم اگر در آن محدوده به جزییات دقیق تر نیاز داشته باشیم میتوان ززولوشن بیشتر آن را از سطوح قبلی (با رزولوشن بیشتر) بررسی کرد.

۲- search for correspondence : دیتکت کردن ابجکت در هرم گوسی میتوان با هزینه محاسباتی کمتر انجام شود و در زمان کمتر.

برخی فواید هرم لاپلاسین:

۱- image blending : زمانی که میخواهیم ۲ تصویر را با هم مخلوط کنیم که بین آن ها لبه مشخصی نباشد، لاپلاسین تصویر حاصل را با استفاده از لاپلاسین ۲ تصویر ورودی به دست می آوریم.

۲- image compression : در هرم لاپلاسین سطح آخر، سطح آخر هرم گوسی است اما بقیه سطح ها همه درواقع ماتریس های با مقدار زیاد ۰ (ماتریس saprse) هستند. در نتیجه نیاز به فضای کمتری برای ذخیره آن ها میباشد و چون میتوان دوباره از روی خود هرم لاپلاسین

اگر سایز تصویر n در n باشد و سایز فیلتر f در f انگار با اعمال فیلتر کل تصویر مرتبه زمانی به صورت توان ۲ از fn خواهد بود. اما اگر فیلتر جداپذیر باشد و در مرحله اول ابتدا حاصل تصویر را با بردار ستونی و بعد حاصل این حاصل ضرب را در بردار سطری ضرب کنیم، نتیجه نهایی مانند کانوالو کل فیلتر در تصویر است اما مرتبه زمانی در حالتی که اجزا فیلتر جدا پذیر را به صورت جدا در تصویر ضرب کنیم کمتر خواهد بود. با ضرب بردار ستونی در تصویر مرتبه زمانی به صورت توان ۲ f در n خواهد بود. مرتبه زمانی برای ضرب بردار سطری نیز همین خواهد بود به طور کلی ضرب تصویر در ۲ جز سطری و ستونی فیلتر از مرتبه توان ۲ f در n خواهد بود در حالی که کانوالو فیلتر اصلی در تصویر از مرتبه توان ۲ از fn خواهد بود. پس اگر به جای کانوالو یک فیلتر جداپذیر به طور کلی در تصویر اجزا آن را به ترتیب ضرب کنیم در عکس مرتبه زمانی کمتر خواهد شد.

اعمال فیلتر ها به صورت cascading به این معنا است که چندین فیلتر پشت سر هم به جای اعمال فیلتر بزرگتری است. به عنوان مثال هرم های گوسین را مثال زد که به جای اعمال یک گوسین بزرگ بر روی کل تصویر، در هر مرحله یک بار یک گوسین کوچکتر اعمال میشود. در هنگام کانوالو فیلتر بزرگ با عکس بدون cascading مرتبه زمانی به صورت توان ۲ f در n خواهد بود. در حالی که اگر همین فیلتر به صورت cascading از فیلتر های کوچک اعمال شود، چون سایز فیلتر ها کوچکتر میشود مرتبه زمانی در cascading کاهش میابد.

برای الگوریتمی که در قسمت دوم سوال خواسته شده (هرم سطح گوسین، انحراف معیار متفاوتی دارد) است میتوان به صورت زیر عمل کرد:

۱- در این سطح اول باید گوسین با انحراف معیار σ اعمال شود. چون فیلتر گوسین جدا پذیر است با ضرب کردم جز ستونی و سطری آن به صورت جداگانه باعث افزایش سرعت میشود.

۲- در سطح دوم باید گوسین با انحراف معیار $\sqrt{2}\sigma$ اعمال شود و مانند مرحله ۱ اگر آن را به بردار سطری و ستونی شکسته و بعد اعمال کنیم باعث کاهش مرتبه زمانی میشود.

۳- در سطح آخر باید باید گوسین با انحراف معیار 2σ اعمال شود. چون فیلتر گوسین جدا پذیر است با ضرب کردم جز ستونی و سطری آن به صورت جداگانه باعث افزایش سرعت میشود.

در هر یک از سه سطح از جداپذیری فیلتر گوسین و تاثیر آن در افزایش سرعت استفاده شد بعضافه اینکه عملاً تعدادی فیلتر پشت سر هم همان cascading است که

تصویر اولی را ساخت در نتیجه استفاده از این هرم برای فشرده سازی عکس مناسب است.

همچنین در توضیحات گفته شد که بر روی تصویر لنا نیز هرم گوسین و بعد از آن لاپلاسین استفاده شود و سپس با کمک این ۲ هرم دوباره تصویر reconstruct شود.

برای این عمل تابع reconstruct_ نوشته شده است. این تابع برای ورودی دو آرایه شامل عکس ها در سطح هرم گوسین و هرم لاپلاسین دریافت میکند. در یک حلقه به تعداد سطح ها تکرار میشود، در هر تکرار ، عکس در سطح فعلی گوسین با استفاده از تابع unsample_px_replication که در سوال ۱-۱-۵ توضیح داده شد، upsample میشود و خروجی آن عکس لاپلاسین سطح قبلی جمع میشود. در نهایت خروجی که تخمینی از عکس اصلی است به عنوان خروجی برگردانده میشود.



شکل ۴) تصویر لنا

با مقایسه شکل ۳ و ۴ مشاهده میشود که تصویر reconstruct شده شکل ۳ لبه های شارپ تری دارد که باعث شده است کیفیت شکل ۳ نسبت به تصویر اصلی (شکل ۴) افزایش پیدا کند.

۴-۲ تمرین ۴-۱-۵

در این تمرین خواسته شده است تا هرم ۳ سطحی approximation یا averaging (box filter) به دست آورده شود و سپس هرم ۳ سطحی residual یا laplacian از روی هرم که با استفاده از باکس فیلتر به دست آمده است، محاسبه شود. هرم ها باید بر روی تصویر لنا محاسبه شوند. برای محاسبه هرم باکس فیلتر تابعی به نام boxfilter_pyramid نوشته شده است که مانند تابع guassian_pyramid توضیح داده شده در سوال ۱-۱-۵ میباشد با این تفاوت که در مرحله کانوالو کرنل با عکس اینجا از کرنل باکی فیلتر استفاده میشود و بقیه مراحل مشابه است. برای محاسبه هرم لاپلاسین از روی هرم باکس فیلتر هم تابع box_filter_laplacian_pyramid نوشته شده است که مانند تابع laplacian_pyramid در سوال ۱-۵ عمل میکند با این تفاوت که اینجا هرم لاپلاسین را از روی هرم باکس فیلتر میسازد و نه گوسین.



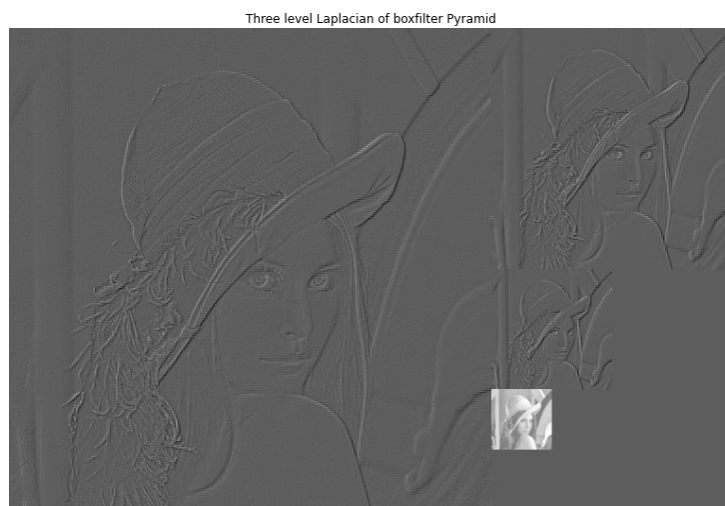
شکل ۳) تصویر construct شده لنا با استفاده از هرم گوسین و لاپلاسین

به عنوان ورودی عکس ورودی و تعداد سطح هایی که باید تبدیل موجک انجام شود را دریافت میکند. میدانیم که DWT در هر سطح از تبدیل ۴ خروجی بر میگرداند که یکی از آن ها LL یا approximation است و بقیه لبه های افقی و عمودی و ۴۵ درجه هستند. عملکرد این تابه به این صورت است در هر مرحله approximation را نگه میدارد تا بعدا با این ها یک approximation_pyramid بسازد تا بتوان آن را با boxfilter pyramid سوال ۴-۱-۵ مقایسه کرد. همچنین خروجی که بعد از ۳ بار تبدیل موجک به دست می آید که شامل ۹ قسمت حاوی لبه و یک approximation میباشد با laplacian pyramid سوال ۴-۱-۵ مقایسه میشود.

این تابع با استفاده از تابع wavedec2 از pywt که به آن ورودی haar برای هار انالیز داده شده و عدد ۳ به عنوان اینکه تبدیل تا چند سطح اجرا شود. Wavedec2 به عنوان خروجی اول خود ماتریس ضرایب موجک را بعد از ۳ سطح خروجی میدهد. خروجی تبدیل موجک بعد از ۳ سطح بر روی تصویر لنا:



شکل ۵) هرم ۳ سطحی باکس فیلتر بر روی تصویر لنا



شکل ۶) هرم ۳ سطحی لاپلاسین که از روی هرم باکس فیلتر شکل ۵ ساخته شده است.

بر اساس مقایسه شکل های ۵ و ۶ با شکل های ۱ و ۲ به و چون فیلتر گوسین به علت وزن دار بودن در مرکز میتواند نویز هارا نیز تقویت کند اما باکس فیلتر این عمل را انجام نمیدهد، مشاهده میشود که به لبه های شارپ تری در شکل ۶ رسیده ایم.

۲-۵ تمرین ۵-۱-۵

در این تمرین خواسته شده است که تبدیل موجک برای با استاده از آنالیز هار تا ۳ سطح بر روی عکس لنا انجام شود و هرم موجک به دست آمده با هرم تمرین ۳-۱-۵ مقایسه شود.

برای پیاده سازی تبدیل موجک تابع wavelet_transform پیاده سازی شده است. این تابع

شکل ۷ که خروجی و ضرایب موجک بعد از ۳ بار تبدیل موجک است تنها شامل ۱ عدد approximation که مربوط به سطح ۳ است میباشد و بقیه ۹ ناحیه مربوط به لبه های افقی و عمودی و ۴۵ درجه در سطح های متفاوت هستند. (تعداد دفعات مختلف تبدیل موجک)

شکل ۷) هرم موجک بعد از ۳ سطح تبدیل موجک بر روی تصویر لنا

در شکل ۷ دیده میشود که لبه هایی که بعد از بار سوم تبدیل موجک گرفتن محاسبه شده اند لبه های پهن تر و مشخص تر و ملموس تری نسبت به لبه ها در در سطح ۱ و ۲ هرم (بعد از ۱ و ۲ بار تبدیل موجک محاسبه کردن) این ضرایب (شکل ۷) را میتوان با laplacian pyramid سوال ۴-۵ (شکل ۶) مقایسه کرد.

در شکل ۷ لبه ها علاوه بر تفکیک شدن بر اساس سطحی از هرم که در آن قرار دارند بر اساس اینکه لبه عمودی یا افقی یا ۴۵ درجه هستند هم تفکیک شده اند. به این معنا که در هرم موجک اطلاعات تفکیک شده تر هستند. همچنین میتوان مشاهده کرد که لبه هایی که در شکل ۶ مبینیم بیشترین شباهت را به لبه های سطح ۳ از هرم موجک دارند.

تابع wavelet_transform پس از محاسبه ضرایب موجک بعد از ۳ تبدیل، با استفاده از یک حلقه به اندازه تعداد بار های انجام تبدیل موجک (تعداد سطوح هرم موجک)، در هر iteration یک تبدیل موجک را با استفاده از pywt.dwt2 اجرا میکند و قسمت approximation آن را ذخیره میکند به فرمت یک هرم. هدف از این حلقه ساختن یک approximation pyramid از approximation های موجک در سطح میباشد تا آن را با هرم باکس فیلتر سوال ۴-۵ بتوان مقایسه کرد.

در انتها نیز تابع wavelet_transform ضرایب موجک به شکل آرایه (چیزی که در شکل ۷ نمایش داده شده) و ضرایب موجک خام (قبل از استفاده از coeffs_to_array) و approximation pyramid تشکیل شده از approximation های هر تبدیل موجک را بر میگردداند. approximation pyramid تشکیل شده از approximation های هر تبدیل موجک:



شکل ۸ (approximation pyramid تشکیل شده از approximation های هر بار اجرای تبدیل موجک)

شکل ۸ را میتوان با شکل ۵ که هرم باکس فیلتر از ۵-۴ است مقایسه کرد. در شکل ۸ تصویر سمت چپ همان تصویر اصلی لنا است و سه تصویر سمت راست approximation های هر بار تبدیل موجک هستند که همانطور که مشاهده میشود از سطوح ۱ تا ۳ هرم باکس فیلتر شکل ۵ کیفیت بهتری دارند.

میدانیم که چه هرم ها و چه تبدیل موجک که به نوعی خود نیز هرم میسازد همه از روش های multi resolution processing هستند و همان طور که ذکر شد موجک لبه ها را تفکیک میکند بر اساس جهت و approximation ها هم کیفیت بهتری دارند نسبت به هرم های میانگین. پس میتوان نتیجه گرفت که تبدیل موجک روش multi resolution processing بهتری نسبت به هرم های میانگین و هرم لاپلاسین است.

۲-۶ تمرین ۶-۱-۵

در این سوال خواسته شده است تا کلیه ضرایب بعد از اعمال ۳ سطح تبدیل موجک که تابع wavelet_transform آن را به عنوان خروجی دوم خود (coeffs) برمیگرداند را طبق فرمول خواسته شده سوال Quantize کرده و سپس با ضرایب جدید عکس دروباره ساخته شود و مقدار PSNR برای عکس حاصل حساب شود.

خروجی coeffs که درواقع خروجی pywt.wavdec2 است، به فرم زیر میباشد:

$$[cAn, (cHn, cVn, cDn), \dots (cH1, cV1, cD1)]$$

ایندکس ۰ در coeffs ماتریس مربوط به عکس approximation میباشد و تاپل های ۳ تایی از n تا ۱، شامل لبه های افقی و عمودی و قطری در هر سطح از تبدیل موجک هستند. به ترتیب از سطح آخر تا سطح اول.

اگر c هر ضریب موجک باشد فرمول خواسته شده توسط سوال برای کوانتایز کردن ضرایب به صورت زیر است:

$$x' = y * \text{sgn}(c) * \text{floor}[\frac{|c|}{y}]$$

برای کوانتایز کردن ضرایب موجک طبق فرمول بالا تابع coefficient_quantize پیاده سازی شده است. این تابع ضرایب موجک بعد از سه سطح را به عنوان ورودی دریافت میکند.

فرمت این ضرایب در بالا توضیح داده شد که ایندکس ۰ آن مربوط به ضرایب approximation است و بقیه ایندکس ها تاپل های سه تایی از ضرایب لبه های افقی و عمودی و قطری از سطح آخر تا سطح اول.

PSNR به صورت گسترده برای اندازه گیری کیفیت فشرده سازی که روی عکس انجام شده است، به کار میرود.

کاری که در این سوال انجام شد نیز درواقع یک lossy compression بود که با PSNR میتوان کیفیت آن را سنجید.

مقدار PSNR برای شکل ۹ برابر است با 20.8 به این معنا که عکس نویزی (تصویر فشرده شده) که در واقع شکل ۹ میباشد با عکس اصلی db 20.8 تفاوت دارد. در واقع PSNR شبیه به MSE میباشد.

این تابع در ابتدا ماتریس ضرایب ایندکس ۰ را طبق فرمول بالا کوانتایز میکند و بعد خروجی را به عنوان ایندکس ۰ آرایه new_coeff قرار است شامل ضرایب کوانتایز شده باشد و درواقع خروجی این تابع است، قرار میدهد.

سپس یک حلقه از ۱ تا طول آرایه ضرایب وجود دارد که در هر تکرار این حلقه، تاپل حاوی سه آرایه ضرایب لبه های افقی و عمودی و قطری را دریافت کرده و هر کدام را جداگانه توسط فرمول بالا کوانتایز میکند و بعد خروجی هر سه را به شکل یه تاپل سه تایی در ایندکس مربوطه در new_coeffs قرار میدهد.

در نهایت این تابع new_coeffs را به عنوان خروجی (ضرایب کوانتایز شده که فرمت قرار گیری ضرایب کنار هم تغییر نکرده و به شکل توضیح داده در بالا میباشد) بر میگردد.

سوال میخواهد تا با این ضرایب دوباره عکس ساخته شود و تابعی با نام reconstruct_wawelet_coeff پیاده سازی شده است که به عنوان ورودی ضرایب جدید کوانتایز شده را دریافت میکند. این تابع با اعمال معکوس تبدیل موجک به وسیله pywt.idwt2 بر روی ضرایب کوانتایز شده جدید، با برگشت به حوزه مکان از حوزه تبدیل عکس را دوباره میسازد با ضرایب جدید کوانتایز شده ای که به عنوان ورودی دریافت کرد و عکس ساخته شده را به عنوان خروجی بر میگردد.



شکل ۹) عکس reconstruct شده توسط ضرایب موجک کوانتایز شده تصویر لنا

در ادامه سوال خواسته است تا مقدار PSNR برای این تصویر خواسته شده (شکل ۹) محاسبه شود.

۳- کد تمرینات