دانشگاه صنعتی شریف
دانشکده علوم ریاضی

پروژه درس محاسبات ماتریسی
رشته علوم کامپیوتر

فشرده سازی و بازسازی بهینه تصاویر مبتنی بر روش تجزیه مقادیر تکین و تبدیل فوریه ماتریسی

نگارش

نیما کلیدری

استاد درس

دکتر رزوان

بهمن ۱۴۰۲

بسم الله الرحمن الرحیم

چکیده

اشغال شدن زیاد فضای ذخیره سازی، ماهیت جدانشدنی دانش پردازش تصاویر است. از زمان گسترش این دانش، فعالان این حوزه به دنبال آن بودند که روش هایی کارا برای کاهش حجم اشغال شده در طی ذخیره تصاویر با استفاده روش های متنوع ارائه بدهند. در طی سالیان، روش های متفاوتی ارائه شدند که هر کدام ترکیب های مختلفی از چند روش اولیه را شامل می‌شد. از پر استفاده ترین آنها میتوان به روش هایی پرداخت که با استفاده از بلوک بندی تصاویر، تبدیل های فوریه و کسینوسی، و همچنین کدگذاری هافمن، به طور بهینه فشرده سازی تصاویر را انجام می‌دهند. این فشرده سازی ها میتواند با یا بدون از دست رفتن اطلاعات درون تصاویر باشند. در این پروژه، روش فشرده سازی ای معرفی می‌شود که با استفاده از تجزیه مقادیر تکین و تبدیل های فوریه ماتریسی، فشرده سازی ای را انجام می‌دهد که در کنار از دست رفتن اطلاعات و کاهش کیفیت کلی، کاهش حجم بسیار زیادی را می‌دهد. این تغییر حجم در سیستم هایی که نیاز به حفظ ابعاد تصاویر در کنار اشغال فضای ناچیز برای ذخیره تصویر دارند، میتواند به طور برجسته ای مفید ظاهر شود.

کلیدواژه‌ها: تجزیه مقادیر تکین، تبدیل فوریه، فشرده سازی، تصویر

فهرست مطالب

۱ مقدمه ۱

۱-۱ تعریف مسئله ۱

۱-۲ اهمیت موضوع ۲

۱-۳ اهداف پروژه ۲

۱-۴ ساختار پروژه ۳

۲ روش ۴

۲-۱ ساختار فنی پروژه ۴

۲-۱-۱ بخش اجرایی ۴

۲-۱-۲ بخش ذخیره ۴

۲-۱-۳ کتابخانه های استفاده شده ۵

۲-۲ بخش ذخیره سازی تصاویر ۵

۲-۲-۱ تعریف کتابخانه ها و توابع آنها ۵

۲-۲-۲ تعریف مقدار های ثابت درون برنامه ۶

۲-۲-۳ دریافت تصویر ۶

۲-۲-۴ پر کردن پوشه ساخته شده با اطلاعات درون تصویر ۷

۲-۳ بخش بازگردانی تصاویر ۸

۲-۳-۱ اجرای تابع تعریف شده ۸

۲-۳-۲ تعریف کتابخانه ها و توابع آنها ۸
۲-۳-۳ تعریف ثابت های درون برنامه ۸
۲-۳-۴ استخراج تصویر ۹
۲-۳-۵ بازگردانی کانال های رنگی تصویر ۹
۲-۳-۶ اجرای تابع بازگردانی ۱۰
۲-۳-۷ اعمال تبدیل فوریه روی هر کدام از کانال های رنگی و گذراندن فیلتر از روی آنها ۱۰
۲-۳-۸ شارپ کردن تصویر پایین گذر ۱۰
۲-۳-۹ تنظیم تصویر بالاگذر و اعمال آن بر رو نتیجه ۱۰
۲-۳-۱۰ نمایش ۱۰
۲-۴ کتابخانه های طراحی شده ۱۱
۲-۴-۱ کتابخانه Essential ۱۱
۲-۴-۲ کتابخانه Fourier ۱۱
۲-۴-۳ کتابخانه Filters ۱۲

۳ آزمایش و بررسی ۱۳
۳-۱ نتایج حاصل شده در حالت پیشفرض ۱۳
۳-۲ بررسی نتایج حاصل شده ناشی از تنظیمات گوناگون ۱۶
۳-۲-۱ تغییر نوع تصویر به ۸ بیتی ۱۶
۳-۲-۲ تغییر comp ۱۷
۳-۲-۳ تغییر scale ۱۷
۳-۲-۴ تغییر k ۱۸
۳-۲-۵ تغییر ضریب lowpass ۱۹
۳-۲-۶ تغییر ضریب highpass ۱۹
۳-۳ مقایسه حجم در تنظیمات متفاوت ۲۰

۳-۳-۱ تغییر تصویر ۱۶ بیتی به ۸ بیتی ۲۰
۳-۳-۲ تغیر comp و threshhold ۲۰
۳-۳-۳ تغیر scale ۲۱

۴ بحث **۳۲**

۵ نتیجه‌گیری **۳۴**

۶ مراجع **۳۵**

فهرست تصاویر

۳-۱ تصویر فلفل برای آزمایش برنامه ۱۴
۳-۲ خروجی پیشفرض برنامه در حوزه مکان ۱۵
۳-۳ خروجی پیشفرض برنامه در حوزه فرکانس ۱۶
۳-۴ فیلتر های پایین و بالاگذر پیشفرض برنامه ۱۷
۳-۵ تصویر های پایین گذر و بالاگذر اعمال شده روی تصویر فلفل ۱۸
۳-۶ تصاویر کاهش داده شده در حوزه فرکانس و حوزه مکان ۱۹
۳-۷ تصاویر بالا و پایین گذر ۲۰
۳-۸ تصویر بعد از فیلتر های مختلف و تصویر خام ۲۱
۳-۹ تصویر بعد و قبل از اضافه کردن عامل بالاگذر به تصویر ۲۲
۳-۱۰ تصویر اولیه و نهائی ۲۳
۳-۱۱ مقایسه تصویر حاصل از ذخیره تصویر ۱۶ و ۸ بیتی ۲۳
۳-۱۲ مقایسه تصویر حاصل از comp ۴۵۰ و ۱۲۰۰ . . . . . ۲۴
۳-۱۳ مقایسه تصویر حاصل از comp ۴۵۰ و ۱۵۰ . . . . . ۲۴
۳-۱۴ مقایسه تصویر حاصل از scale ۴.۱ و ۲.۲ . . . . . ۲۵
۳-۱۵ مقایسه تصویر حاصل از scale ۴.۱ و ۱ . . . . . ۲۵
۳-۱۶ مقایسه تصویر حاصل از k زیاد و متعادل ۲۶
۳-۱۷ مقایسه تصویر حاصل از k کم و متعادل ۲۶
۳-۱۸ مقایسه تصویر حاصل از lowpass ۱۷۵ و ۷۵ . . . . . ۲۷

۳-۱۹ مقایسه تصویر حاصل از lowpass ۳۵ و ۷۵ . . . . . ۲۷
۳-۲۰ مقایسه تصویر حاصل از highpass ۱۰۵ و ۲۱ . . . . . ۲۸
۳-۲۱ مقایسه تصویر حاصل از highpass ۵ و ۲۱ . . . . . ۲۸
۳-۲۲ مقایسه حجم اشغال شده توسط فایل ذخیره شده در ۳ حالت ذخیره متفاوت . ۲۹
۳-۲۳ نمودار حجم فایل بر حسب comp ۲۹
۳-۲۴ نمودار لگاریتم حجم فایل بر حسب comp ۳۰
۳-۲۵ نمودار لگاریتم حجم فایل بر threshhold ۳۰
۳-۲۶ نمودار scale بر حسب حجم فایل ۳۱
۳-۲۷ نمودار scale بر حسب لگاریتم حجم فایل ۳۱

۴-۱ تصویر با کیفیت بالا ۳۳
۴-۲ تصویر تخریب شده قلعه لیختن اشتاین ۳۳

فصل ۱

مقدمه

در این فصل سعی شده به فلسفه فشرده سازی تصاویر پرداخته شود. در ادامه سعی میشود نقاط پر اهمیت در این مسیر توضیح داده شوند. سپس به اهداف این پروژه و برتری های نسبی آن در مقایسه با رویکرد های مشابه پرداخته میشود. در پایان نیز ساختار پروژه و شمای کلی آن شرح داده میشوند.

۱-۱ تعریف مسئله

ذخیره سازی اطلاعات رکن اساسی مهندسی کامپیوتر می‌باشد. بخش زیادی از مبانی مهندسی و علوم کامپیوتر، پرداختن به نحوه ذخیره سازی اطلاعات می‌باشد. در این بخش ها سعی می‌شود اطلاعات به گونه ای ساختار پیدا کند و ذخیره شوند، که مجموع وزن دار زمان صرف شده برای ذخیره اطلاعات، دریافت اطلاعات و حافظه استفاده شده برای ذخیره آن‌ها به حداقل مقدار ممکن برسد. وزن اشاره شده بسته به کاربرد های مختلف می‌تواند متفاوت باشد. به این منظور، شاخه هایی از مهندسی و علوم کامپیوتر در طول سالیان به وجود آمده که با استفاده از تکنیک های ریاضی و آماری، سعی در به حداقل رساندن زمان و حافظه مصرفی برای انجام عملیات ذخیره و دریافت اطلاعات دارند.

۱-۲ اهمیت موضوع

امروزه بخش اعظم حافظه های موجود در دستگاه های الکترونیکی مختلف، توسط محتوای چند رسانه ای اعم از تصویر و صوت اشغال شده اند. از آنجا که افزایش ظرفیت برای بسیاری از وسایل الکترونیکی می‌تواند سخت، هزینه بر و حتی غیر ممکن باشد، گاها لازم است بجای صرف انرژی برای گسترش حافظه ها، با استفاده از تکنیک های مختلف نرم افزاری، سعی در کاهش حجم آن‌ها داشت. به این ترتیب میتوانیم به مزایای زیر دست پیدا کنیم:

- ایجاد فضای بیشتر برای قرار گیری محتوای بیشتر در فضای ذخیره سازی ثابت
- صرف انرژی و زمان کمتر برای توسعه های نرم افزاری بجای توسعه سخت افزاری
- یافتن راه حل های جدید تر برای حل مشکلات مشابه اعم از کاهش پیچیدگی محاسبه در مسائل خاص
- قابلیت استفاده وسیع تر از محتواهای چند رسانه ای که با محدودیت شدید محاسباتی و حافظه ای روبرو هستند

۱-۳ اهداف پروژه

در این پروژه، سعی بر آن بوده که هدف، فشرده سازی شدید تصویر و کاهش چند برابر حجم آن‌ها باشد. واضح است که این فشرده سازی شدید تصویر، نمیتواند بدون از دست رفتن اطلاعات باشد. از طرفی انتخاب اینکه کدام بخش هایی از اطلاعات دارای اهمیت کمتری در تصویر هستند، خود نیز یک موضوع چالش بر انگیز می‌باشد و هدف اول این پروژه پاسخ دادن به این سوال و حذف آن‌ها بدون از دست دادن اطلاعات مهم درون تصویر و بازسازی مناسب تصاویر با جزئیات حداکثری درون آن است. هدف دوم آن، کاهش حداکثری حجم فایل ذخیره شده است که اطلاعاتی از تصویر را در خود دارد. هدف سوم، این است که ذخیره و بخصوص بازگردانی تصویر در حداقل زمان ممکن اجرا شود. این به این دلیل میباشد که تصاویر به راحتی با سخت افزار های پیشرفته تر میتواند فشرده شود؛ این در حالیست که فایل مورد استفاده می‌تواند روی دستگاه هایی انتقال پیدا کند و استفاده شود که مشکل شدید تری در محدودیت سخت افزار و حافظه دارند.

۱-۴ ساختار پروژه

ساختار پروژه، در ابتدا بر اساس دو بخش ذخیره و بازگردانی بود که در فایل های متلب نوشته شد. لیکن در ادامه کار، به مشکلاتی برخوردیم که مانع استفاده مناسب از متلب شد. مشکلاتی مانند عدم تطابق توابع مختلف در حوزه فرکانس با آنچه کار شده بود و انتظار داشتیم، همچنین عدم وجود تعدادی کتابخانه پردازش تصویر که باعث شد نشود به نحو مناسب، تصاویر را به عنوان تصویر هایی ۱۶ بیتی با فرمت png ذخیره و بازگردانی کنیم، همه و همه دلایلی بودند که ما را از ادامه کار با متلب باز داشتند. هرچند فایل های SAVE.m و LOAD.m به نحوی که در ابتدا نوشته شده بود، قرار دارند که دارای ضعف های ساختاری اند و امکان اجرای مناسب را ندارند. ساختار پروژه، اساسا از دو بخش ذخیره و بازگردانی تشکیل شده که برای هر کدام یک فایل دفترچه ژوپیتر درون فایل پروژه قرار دارد.

- در بخش ذخیره ۳ کانال رنگی تصویر از آن استخراج میشود و روی هر سه کانال تجزیه مقادیر تکین انجام میشود. سپس مقادیر مهمتر با توجه به مقادیر تکین هر کانال جدا شده و همراه چند فایل کانفیگ و مشخصات تصویر درون یک فایل zip ذخیره میگردند.
- برای بخش بازگردانی تصویر نیز، در ابتدا فایل های کانفیگ از درون فایل zip خوانده شده و با توجه به آنها تصاویر حاوی اطلاعات مهم تصویر دریافت شده و تبدیل به اطلاعات قابل استفاده شده است. در مرحله بعد نیز با استفاده از انتقال هر کانال رنگی از حوزه مکان به حوزه فرکانس با تبدیل فوریه، میتوانیم به محاسبات بسیار کمتر و سرعت بیشتر، جزئیات تصویر بازگردانی شده را در آن ایجاد کرده و سعی کنیم تا حد زیادی آن ها را با استفاده از فیلتر ها و محاسبات مختلف در حوزه فرکانس، آن ها را به تصویر برگردانیم.

همچنین ۳ فایل پایتون به حالت کتابخانه درون فایل پروژه قرار دارند که شامل توابع استفاده شده درون این دو فایل دفترچه ژوپیتر هستند و برای اجرای صحیح آن‌ها الزامی هستند.

فصل ۲

روش

در این بخش، در ابتدا به ساختار فنی پروژه میپردازیم و سپس دو بخش ذخیره و بازگردانی آن را توضیحات کاملی میدهیم. در نهایت نیز به ساختار هر کدام از فایل های کتابخانه های آماده شده پایتون و توابع آن‌ها میپردازیم

۲-۱ ساختار فنی پروژه

۲-۱-۱ بخش اجرایی

این برنامه با زبان برنامه نویسی پایتون نوشته شده و در پایتون نسخه ۷.۹.۳ اجرا و آزمایش گردیده است. قالب فایل های آن در دو نوع فایل py و ipynb است. درون سه فایل py ، توابع مورد نیاز برای اجرای صحیح فایل های ipynb قرار دارد. همچنین دو فایل ipynb ، مرحله به مرحله اجرا میشوند و قبل از هر سلول توضیح مختصری در مورد کارکرد آن داده شده است.

۲-۱-۲ بخش ذخیره

تصویر باید درون مسیر ذخیره فایل های ipynb به عنوان تصویری با فرمت png ذخیره شده باشد. برنامه نام تصویر را در طول اجرای برنامه دریافت میکند و نیاز نیست نام خاصی داشته باشد. سپس اطلاعات درون تصویر به درون یک فولدر در همان مسیر و به نام تصویر ذخیره شده و سپس به عنوان یک فایل zip با همان نام تصویر ذخیره میگردد. در بخش بازگردانی هم این فایل zip ابتدا به

فولدر و در نهایت به یک تصویر با نام فایل zip تبدیل میگردد.

۲-۱-۳ کتابخانه های استفاده شده

در این پروژه از کتابخانه های خارجی زیر استفاده شده است:

۱. کتابخانه opencv : یک کتابخانه مربوط به پردازش تصاویر است و از آن برای کار هایی مانند دریافت و نوشتن تصاویر در حافظه استفاده میشود.

۲. کتابخانه numpy : برای انجام اعمال ماتریسی از جمله تجزیه مقدار تکین استفاده می‌شود.

۳. کتابخانه imageio : برای نوشتن تصاویر ۱۶ بیتی در فایل zip مورد استفاده قرار میگیرد.

۴. کتابخانه :os کتابخانه ای اساسی است برای کارهایی از قبیل ساخت و حذف پوشه ها.

۵. کتابخانه shuntil : کتابخانه ای برای ساختن فایل zip است.

۶. کتابخانه zipfile : یک کتابخانه برای خواندن و Extract کردن فایل zip می‌باشد.

۷. کتابخانه matplotlib : کتابخانه ای مشهور برای به تصویر کشیدن نتایج، داده ها و نمودارهاست.

۲-۲ بخش ذخیره سازی تصاویر

در این بخش به توضیح دقیق و مفصل بخش ذخیره سازی تصاویر می‌پردازیم. این بخش در یک فایل دفترچه با نام SAVE ذخیره شده است.

۲-۲-۱ تعریف کتابخانه ها و توابع آنها

در این بخش هم کتابخانه های توضیح داده شده را اضافه کرده و هم کتابخانه های داخلی را تعریف میکنیم و توابع مورد نیاز از آنها را به برنامه اضافه میکنیم.

۲-۲-۲ تعریف مقدار های ثابت درون برنامه

در این بخش ابتدا نام فایل از کاربر دریافت و ذخیره میشود، سپس مقادیر ثابتی تعریف میشود که در ادامه شرح داده میشوند:

۱. لیست :channels در این لیست نام کانال های رنگی ذخیره میشود. این لیست در فضای RGB برابر r و g و b است. همچنین میتواند فضای رنگی BGR یا حتی HSV باشد که به دلایلی میتواند پایداری RGB را نداشته باشد.

۲. عدد صحیح :comp این مقدار مشخص میکند که تا وقتی که مقدار تکین ها از این مقدار کمتر شدند، جدا شوند و فقط مقدار های بالاترشان ذخیره شوند. همچنین میتوانیم بجای مقدار دهی comp ، به threshhold مقدار دهی کنیم که به طور دقیق مشخص کنیم در هر کانال رنگی به همان اندازه برحسب تعداد مقدار تکین ها جدا کند و مقادیر بالاتر را بردارد. مشخصا به هر مقدار که comp بزرگتر باشد یا threshhold کوچک تر باشد، تصویر کم کیفیت تر بوده و جزئیات کمتری را شامل میشود و در عوض فضای کمتری برای ذخیره سازی اشغال خواهد کرد.

۳. مقدار ممیز شناور scale : این مقدار مشخص میکند که تصویر تا چه اندازه کوچک شود. این کوچک کردن ابعاد به طور چشمگیری باعث افزایش عملکرد الگوریتم فشرده سازی و کاهش حجم شدید فایل ذخیره شده میگردد.

۴. مقدار con_type : این مقدار دو عدد ۲ بتوان ۱۵ منهای ۱ و ۲ بتوان ۷ منهای ۱ میتواند باشد. دلیل این موضوع، این است که برای تبدیل مقادیر حاصل از تجزیه مقادیر تکین به تصویر، لازم است این اعداد بین ۰ تا ۲۵۵ و صفر تا ۲ بتوان ۱۶ قرار بگیرد که در حالت دوم حجم ذخیره بیشتر بوده ولی کیفیت تصویر در بازسازی نهایی رشد شدیدی خواهد کرد.

۲-۲-۳ دریافت تصویر

با استفاده از نام داده شده توسط کاربر، تصویر به عنوان یک ماتریس دریافت میشود. سپس ابعاد آن بنابر مقدار تعیین شده کوچک (یا بزرگ) شده و در نهایت پوشه ای هم نام با تصویر برای ذخیره اطلاعات درون آن بوجود می‌آید.

۲-۲-۴ پر کردن پوشه ساخته شده با اطلاعات درون تصویر

در این بخش تابعی به منظور استخراج و ذخیره داده های درون تصویر وجود دارد. در ابتدا ماتریسی تحت عنوان پیکر بندی درون پوشه ذخیره میشود. در این ماتریس کانال های رنگی، نوع تصاویر مولفه (۸ بیتی یا ۱۶ بیتی) و نیز ضریب بزرگی تصویر قرار میگیرند و درون فایل ذخیره میشوند. در ادامه الگوریتم زیر اجرا میشود.

الگوریتم ۱ تجزیه و بدست آوردن مقدار لازم فشرده سازی

ورودی: تصویر با ۳ کانال رنگی

خروجی: عوامل تکین بریده شده

۱: تا وقتی کانال های رنگی تصویر تمام نشده اند:

۲: کانال رنگی تصویر را جدا کن

۳: آن را به عوامل تکین تجزیه کن

۴: روی مقدار s آن پیمایش کن و مقدار آن را برابر با t قرار بده.

۵: تا وقتی t از comp بزرگتر است:

۶: ایندکس t را در threshhold قرار بده.

۷: threshhold مقدار اول s ، ستون اول u و سطر اول v را ذخیره کن.

۸: این سه ماتریس ذخیره شده را به تابع ذخیره بده.

تابع ذخیره نیز s و u و v بدست آمده را میگیرد، s را چون بردار است و فضای زیادی اشغال نمیکند، به همان صورتی که هست ذخیره میکند. برای u و v در ابتدا حداکثر مقدار مطلقشان را ذخیره میکنیم، سپس با تقسیم کل ماتریس بر آنها، ابتدا آنها را نرمال میکنیم، سپس ضربدر نصف حداکثر تصویر میکنیم (در اینجا ۲ بتوان ۱۵ منهای ۱) که تصاویر بین منفی و مثبت این دو مقدار قرار بگیرند، و در نهایت با جمع این مقدار با درایه های ماتریس به دست آمده، باعث آن میشود که مقادیر بین ۰ و ۲ بتوان ۱۶ منهای ۱ قرار بگیرند. حال با تبدیل آنها به عدد صحیح ۱۶ بیتی و ذخیره تصویر با فرمت ،png تصویری بسیار کم حجم تر از تصویر اصلی خواهیم داشت. در نهایت نیز حداکثر مقدار بدست آمده را برای هر ماتریس ذخیره میکنیم تا بعدا بتوانیم ماتریس را بازگردانی کنیم. این کار را برای هر کانال ذخیره میکنیم و در نهایت ۶ فایل تصویر، ۶ فایل ماتریس و ۱ فایل پیکره بندی خواهیم داشت.

علت ذخیره این ماتریس ها به عنوان تصویر، این بوده که تصاویر هم تصاویر را در صورت

کیفیت خیلی بالا به صورت ۱۶ بیتی ذخیره میکنند و این درحالیست که ممیز شناور حاصل شده از تجزیه، ۶۴ بیتی است و ماتریس آن حجم بسیار بالایی در صورت ذخیره خواهد داشت. از طرفی تصویر png خود نیز تبدیل ها و کدگذاری هایی برای کاهش حجم تصاویر دارد که اگرچه باعث از دست رفتن اطلاعات میشود، اما این تغییر جزئی بوده و در نتیجه حاصل شده تفاوت خاصی نمیکند. از طرفی اگر به تصاویر ۸ بیتی روی بیاوریم، از آنجا که هر عدد ممیز شناور با ممیز های زیاد باید به یک عدد بین ۰ تا ۲۵۵ برسد، در حین بازگردانی آن به نتیجه ی خوبی نخواهیم رسید، چراکه هر عدد ممیز شناور از مقدار واقعی خود میتواند اختلاف فاحشی داشته باشد و این باعث بوجود آمدن رگه هایی از رنگ های نادرست در تصویر است که در تصویر اولیه وجود نداشت. نکته جالب اینجاست که در این حالت هرچقدر مولفه های دریافتی تکین بیشتر شود و تصویر با کیفیت تر تجزیه شود، رگه های رنگه نادرست نیز به علت وضوح بیشتر، افزایش خواهند یافت. به این دلایل در ادامه کلا از تصاویر ۱۶ بیتی برای ذخیره تصاویر عوامل تکین بدست آمده استفاده میشوند.

۲-۳ بخش بازگردانی تصاویر

۲-۳-۱ اجرای تابع تعریف شده

این تابع با گرفتن نام و تصویر اجرا می‌شود و در پس از اجرا شدن، پوشه پر شده را تبدیل به فایل zip کرده و پوشه را حذف می‌کند. بدین ترتیب کار این فایل به پایان میرسد و تصویر دریافت شده را به صورت فایل zip ذخیره میکند.

۲-۳-۲ تعریف کتابخانه ها و توابع آنها

در این بخش هم کتابخانه های توضیح داده شده را اضافه کرده و هم کتابخانه های داخلی را تعریف میکنیم و توابع مورد نیاز از آنها را به برنامه اضافه میکنیم.

۲-۳-۳ تعریف ثابت های درون برنامه

در این بخش هم مانند بخش پیشین، تعدادی ثابت برای تنظیمات مختلف بازگردانی داریم که آنها را توضیح خواهیم داد: (لازم به ذکر است که نام فایل zip هم در ابتدا از کاربر گرفته میشود.)

۱. ضریب k : یک ضریب است که شدن شارپ شدن تصویر در مرحله تبدیل لاپلاسین روی تصویر فوریه را نشان میدهد. هرچقدر بیشتر باشد، تصویر شارپ تری دریافت خواهیم کرد.

۲. ۴ متغیر میبینیم که در قبل هم داشتیم، همگی خالی و صفر هستند و تعریف اولیه شده اند که بعدا توسط اطلاعات درون فایل zip پر شوند

۳. ضریب lowpass_coef : این ضریب تعیین کننده سیگمای کرنل گاوسی ای است که بر روی تصویر فوریه اعمال میشود. هرچه کمتر باشد، تصویر تار تر و هر چه بیشتر باشد، واضح تر است و نویز های بیشتری درون آن قرار میگیرند.

۴. ضریب highpass_coef : این ضریب تعیین کننده سیگمای کرنل گاوسی ای است که بر روی تصویر فوریه اعمال میشود.هر چه کمتر باشد، تصویر بالاگذر پرجزئیات تر است و گوشه ها را بهتر نمایان میکند، از طرفی هم هرچه بیشتر باشد، کم جزئیات تر است اما نویز ها در آن کمتر نمایان خواهند شد.

۲-۳-۴ استخراج تصویر

در ادامه فایل zip تبدیل به یک فولدر به همان نام میشوند.

۲-۳-۵ بازگردانی کانال های رنگی تصویر

تابع load در این بخش تعریف شده که با دریافت نام پوشه، کانال های رنگی آن را خروجی میدهد. در ابتدا پیکره بندی دریافت میشود، سپس عینا برعکس تابع save در بخش قبل، برای هم کانال رنگی، اول حداکثر آن کانال دریافت میشود و سپس u ، v برعکس روش ذخیره کردن، با با استفاده از ضریب و حداکثر مقادیر، بازسازی میشود. s هم که به طور عادی دریافت میشود و تبدیل به یک ماتریس قطری میشود که مقادیر قطر اصلی آن مقادیر s هستند. سپس u در s و حاصلشان در v ضرب ماتریسی میشود و مقدار آن برمیگردد. برای تضمین از بیشتر نشدن مقدار های حاصل از ۲۵۵ و کمتر نشدنشان از ۰ (به علت خطاهای مختلف) مقادیر بین این دو فیکس شده و به عنوان تصویر int، هر سه کانال با هم به عنوان پاسخ تابع خارج میشوند.

۲-۳-۶ اجرای تابع بازگردانی

تابع بازگردانی اجرا شده و تصویر بدست آمده در ابتدا به صورت سیاه و سفید در متغیری برای استفاده های بعدی قرار میگیرد.

۲-۳-۷ اعمال تبدیل فوریه روی هر کدام از کانال های رنگی و گذراندن فیلتر از روی آنها

در این بخش، در ابتدا هر سه کانال با استفاده از تابع fourier-transform به حوزه فرکانس برده میشوند. در ادامه هر یک از توابع lowpass و highpass ، با استفاده از ضریب پایین گذر و بالاگذر تعریف شده و دریافت تصویر ساخته شده از زیربخش قبل، یک فیلتر پایین گذر و یک فیلتر پایین گذر میسازد. با ضرب این فیلتر را روی تصاویر حوزه فرکانس، به ما تصویر پایین گذر و بالاگذر را خواهد داد.

۲-۳-۸ شارپ کردن تصویر پایین گذر

در این زیربخش در ابتدا سه کانال رنگی تصویر پایین گذر را به حوزه مکان برگردانده و از آن تصویر درست میکنیم. در این مرحله با داشتن ،scale ابعاد تصویر را به ابعاد اولیه برمیگردانیم. در ادامه نیز با دادن تصویر به تابع fourier-uv-sharpening به همراه ضریب شارپ کنندگی، آن را شارپ میکنیم.

۲-۳-۹ تنظیم تصویر بالاگذر و اعمال آن بر رو نتیجه

در انتها تصویر بالاگذر را به حوزه مکان برگردانده و با این فیلتر که هر مقدار زیر ۵۰ در آن برابر ۰ شوند، ضریبی از آن را با تصویر نتیجه شارپ شده برای معلوم شدن بریدگی ها و با کیفیت شدن تصویر جمع میکنیم.

۲-۳-۱۰ نمایش

در انتها نیز تمام تصویر های بخش های مختلف را نشان داده و تصویر نهایی را ذخیره میکنیم.

۲-۴ کتابخانه های طراحی شده

۲-۴-۱ کتابخانه Essential

این کتابخانه شامل توابع ضروری برای محاسبات بقیه کتابخانه ها است. توابع موجود در آن به شرح زیرند:

۱. normalize : این تابع یک ماتریس گرفته و مقادیر آن را به بین ۰ تا ۲۵۵ بطور خطی نگاشت میکند.

۲. normalize-binary : این تابع یک ماتریس گرفته و مقادیر آن را به بین ۰ تا ۱ بطور خطی نگاشت میکند.

۳. get-gaussian-kernel : این تابع، با گرفتن l و سیگما، یک تصویر به بزرگی l خروجی میدهد که در میانه آن یک فیلتر گاوسی با سیگمای داده شده قرار دارد.

۴. show-magnitude : این تابع، تصویر فوریه را گرفته و مقدار ویژه لگاریتم آن را برای نمایش مناسب تر بر میگرداند.

۵. show-magnitude-fourier: کار تابع قبلی را انجام میدهد با این تفاوت که خروجی را با استفاده از تابع اول نرمال میکند.

۶. clipping : تصویری را میگیرد و تمام مقادیر روی ۲۵۵ آن را برابر ۲۵۵ و تمام مقادیر منفی آن را برابر ۰ قرار میدهد.

۲-۴-۲ کتابخانه Fourier

در این کتابخانه تبدیل های فوریه انجام میشود. توابع آن بصورت زیر هستند:

۱. inverse-fourier: یک تصویر حوزه فرکانس گرفته و کلیپ شده نرمال قدر مطلق حوزه مکان آن را خروجی میدهد.

۲. fourier-transform : یک تصویر حوزه مکان گرفته و تصویر حوزه فرکانس را خروجی میدهد.

۳. fourier-transform-function : مانند تابع قبلی است.

۴. non-normalized-inverse-fourier-transform-function : مانند تابع اول است اما خروجی را کلیپ و نرمال نمیکند.

۵. fourier-uv-sharpening : با گرفتن تصویر، تبدیل فوریه همزمان سه کانال آن، و دادن آن به تابع uv-main-transform ، تصویر خروجی را شارپ میکند.

۲-۴-۳ کتابخانه Filters

در این کتابخانه فیلتر ساخته و استفاده میشود. توابع آن بصورت زیر هستند:

۱. highpass : این تابع ابعاد یک تصویر را گرفته و با استفاده از تابع ساختن کرنل گاوسی، درون یک صفحه سفید به ابعاد تصویر داده شده، یک دایره گاوسی مشکی رسم میکند.

۲. lowpass : این تابع ابعاد یک تصویر را گرفته و با استفاده از تابع ساختن کرنل گاوسی، درون یک صفحه سیاه به ابعاد تصویر داده شده، یک دایره گاوسی سفید رسم میکند.

۳. uv-main-transform : با گرفتن تصویر F ، $۴\pi^۲(v^۲ + u^۲)$ را خروجی میدهد که u و v مختصات صفحه اند و مرکز صفحه هردو صفر هستند.

فصل ۳

آزمایش و بررسی

در این فصل در ابتدا نتایج بدست آمده از اجرای برنامه در حالت پیشفرض را قرار میدهیم، در ادامه نتایج را روی تنظیمات مختلف به همراه حجم آنها نمایش میدهیم و در مورد آنها بحثی خواهیم داشت. در پایان نیز نمودار حجم تصاویر بر حسب ورودی های مختلف نشان میدهیم و خروجی آن ها را نیز بر اساس تنظیمات پیشفرض خواهیم داشت. در تمامی این تست ها، تصویر مشهور pepper.png استفاده شده است که یکی از تصاویر معروف در زمینه آزمایش های حوزه پردازش تصویر می‌باشد. این تصویر در شکل ۳-۱ نمایش داده شده است.

۳-۱ نتایج حاصل شده در حالت پیشفرض

در این بخش به تصاویر نتیجه بعد از اینکه برنامه را با تنظیمات پیشفرض اجرا میکنیم میپردازیم و آنها را تحلیل میکنیم. تصاویر همگی در قسمت بارگذاری ظاهر میشوند. در ابتدا بعد از اجرای برنامه و گرفتن داده، تصویر بارگذاری شده را در حوزه مکان مشاهده خواهیم کرد. این تصویر در شکل ۳-۲ نمایش داده شده است. همچنین بعد از انتقال آن به حوزه فرکانس، تصویر آن در این حوزه هم در شکل ۳-۳ نمایش داده میشود.

در ادامه، دو فیلتر بالا گذر و پایین ایجاد میکنیم و آن ها را در شکل ۳-۴ نمایش میدهیم.

در مرحله بعد، این دو فیلتر را روی تصویر در حوزه فرکانس ضرب میکنیم و نتایج آن ها را نیز در شکل ۳-۵ نمایش خواهیم داد.همانطور که میبینید، اطراف تصویر پایین گذر کمرنگ است و این همان جزئیات خیلی ریز تصویر هستند که فاصله زیادی تا مبدا دارند و تنها جزئیات را شامل میشوند.

شکل ۳-۱: تصویر فلفل برای آزمایش برنامه

فیلتر بالاگذر برعکس است و میانه تصویر که اطلاعات کلی زیادی آنجاست را نادیده میگیرد و تنها جزئیات را استخراج میکند.

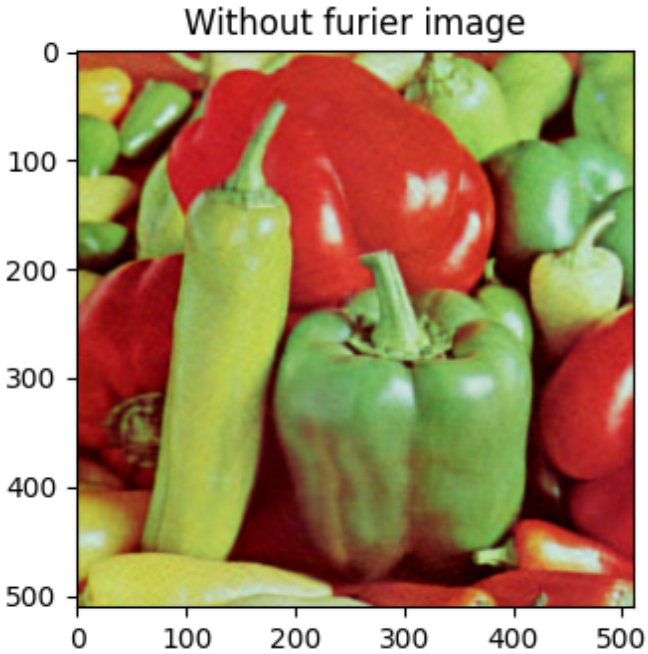
در تصویر ۳-۶ ، تصویر کاهش داده شده پایین گذر را میبینیم. در سمت چپ، این تصویر به حوزه مکان برده شده و تصویری خاکستری با چند رگه سیاه را میبینیم. این ها همان گوشه های اشیا در تصویر هستند. چون این گوشه ها خود از تصویر پایین گذر به دست آمده اند، دیگر نویز های مزاحم را شاهد نیستیم.

در ادامه تصاویر اصلی پایین و بالاگذر را مشاهده میکنیم. این تصاویر با عنوان ۳-۷ ذخیره شده است.

در ادامه، به تفاوت بین تصویر خام و تصویر پس از فیلتر های فوریه نگاهی میکنیم. این تفاوت در تصویر ۳-۸ به وضوح مشخص است. در این تصویر میتوان دید که این تغییرات، باعث همزمان کاهش نویز و بالارفتن کیفیت تصویر و بهبود جزئیات شده است.

سپس به تفاوت بین قبل از اضافه کردن عامل بالاگذر و بعد از آن توجه میکنیم. این تفاوت در تصویر ۳-۹ آمده است. نکته مهم در تفاوت بین این دو تفاوت در لبه های فلفل هاست که به توجه بیشتری میتوان متوجه آن ها شد.

در پایان نیز به تفاوت بین تصویر اولیه و نهایی میپردازیم. اختلاف بین این دو تصویر در شکل ۳-۱۰ قابل مشاهده است.

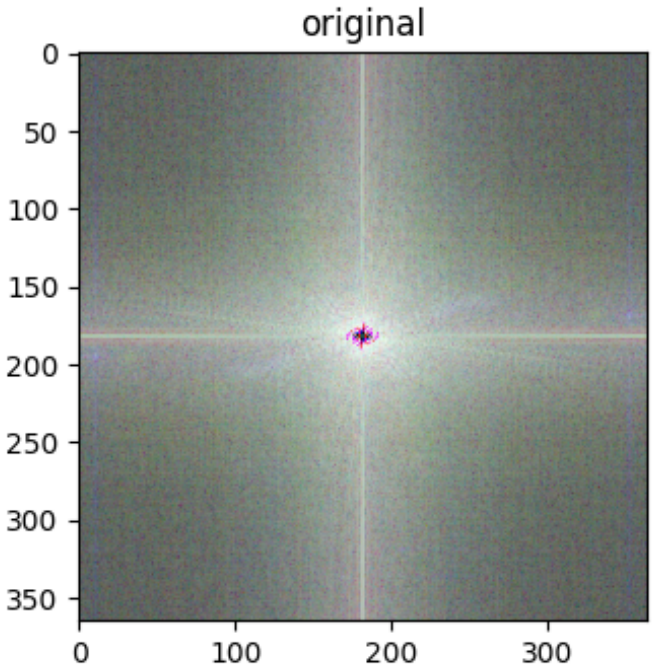


شکل ۳-۲: خروجی پیشفرض برنامه در حوزه مکان

نکته قابل توجه در مقایسه این دو تصویر، این است که تصویر نهایی در حجمی کمتر از یک سوم تصویر اولیه ذخیره شده است. پارامتر های تنظیم شده پیشفرض نیز به صورت زیر هستند:

- نوع تصویر : ۱۶ بیتی
- مقدار comp : ۴۵۰
- مقدار scale : ۴.۱
- مقدار k : ۹ * $(۱۰^{-۶})$
- ضریب lowpass : ۷۵
- ضریب highpass : ۲۱

لازم به ذکر است که این مقادیر از طریق آزمون و خطا به دست آمده اند و بهترین نتیجه نسبت به کمترین حجم قرار داده شده است.



شکل ۳-۳: خروجی پیشفرض برنامه در حوزه فرکانس

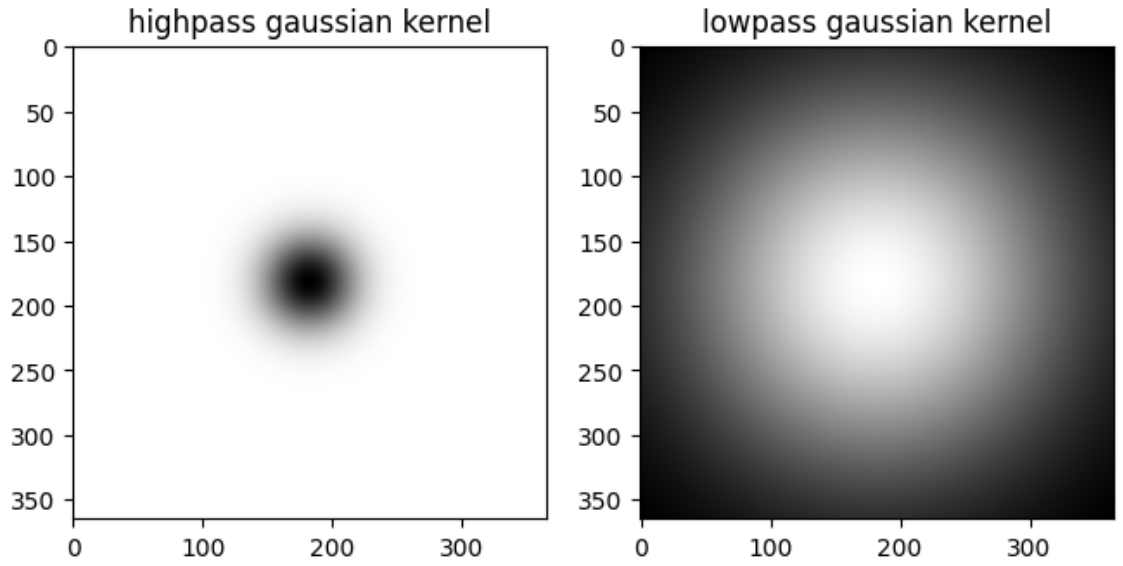
۳-۲ بررسی نتایج حاصل شده ناشی از تنظیمات گوناگون

در این بخش هر کدام از پارامتر های بالا را کم و زیاد میکنیم و تاثیر آن را با مقایسه کردن آن با تصویر پیشفرض در میابیم. لازم به ذکر است که بقیه پارامتر ها در هر آزمایش ثابت در نظر گرفته میشوند.

قابل توجه است که کانال رنگی HSV را کلا بررسی نمیکنیم، چرا که این طیف رنگی تعداد رنگ های بسیار کمتری از RGB دارد و به همین دلیل، تغییر آن و بازگردانی آن، بسیاری از دقت رنگی را از بین برده و باعث میشود تصویری با رنگ های بدتر داشته باشیم که کلا قابل استفاده نیست و همان حجم RGB را نیز به خود میگیرد. (این به این دلیل است که در HSV، مقدار H بین ۰ تا ۱۷۹ است و این مورد خیلی بدی به حساب می‌آید برای هدفی که داریم.)

۳-۲-۱ تغییر نوع تصویر به ۸ بیتی

از آنجا که تصویر ۳۲ بیتی حجم بسیار بالایی میگیرد و احتمالا برای ذخیره تصویرمان کارآمد نباشد، تنها تصویر را به ۸ بیتی تغییر داده و تاثیر آن را بررسی میکنیم. این تغییرات را در ۳-۱۱ میتوانیم



شکل ۳-۴: فیلتر های پایین و بالاگذر پیشفرض برنامه

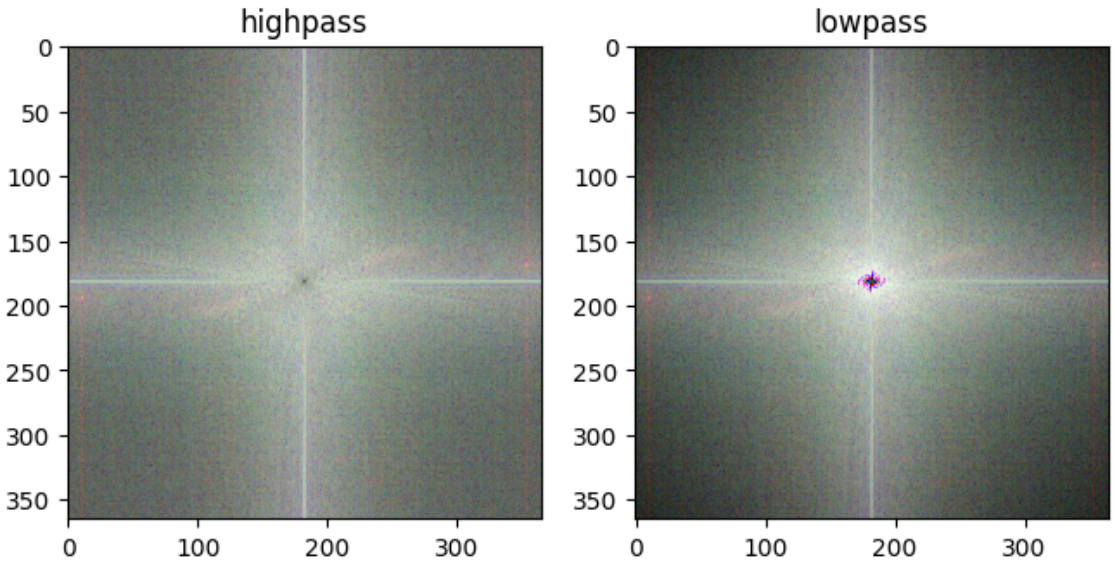
ببینیم.به وضوح میتوانیم چهارخانه های رنگی را در تصویر مشاهده کنیم که به دلیل تغییر شدید بعضی از عوامل تکین به وجود می آیند. (هرچند ما اینجا برای ذخیره سازی عوامل از فرمت png استفاده کردیم. اگر فرمت jpg بود، این راه های افقی و عمودی به دلیل ماهیت فشرده سازی ،jpg بسیار عمیق تر بودند)

۳-۲-۲ تغییر comp

در ابتدا مقدار comp را به ۱۲۰۰ تغییر میدهیم که مولفه های تکین بسیار کمتری را در خود جای میدهد. نتیجه در شکل ۳-۱۲ قابل مشاهده است که چقدر با افت کیفیت مواجع بوده ایم. سپس این مقدار را به ۱۵۰ تغییر میدهیم که از طرفی با افزایش حجم مواجه میشویم، ولی کیفیت تصویر رشد زیادی میکند. این تغییر نیز در شکل ۳-۱۳ نمایان است.(حجم فایل در حد حجم اصلی تصویر میشود)

۳-۲-۳ تغییر scale

طبیعتا با زیاد کردن این عامل، کیفیت خروجی و حجم کم شده و با زیاد کردن آن تا ۱، کیفیت افزایش پیدا میکند؛ چراکه در کوچک و بزرگ کردن تصویر، اطلاعات زیادی از بین میرود و جای

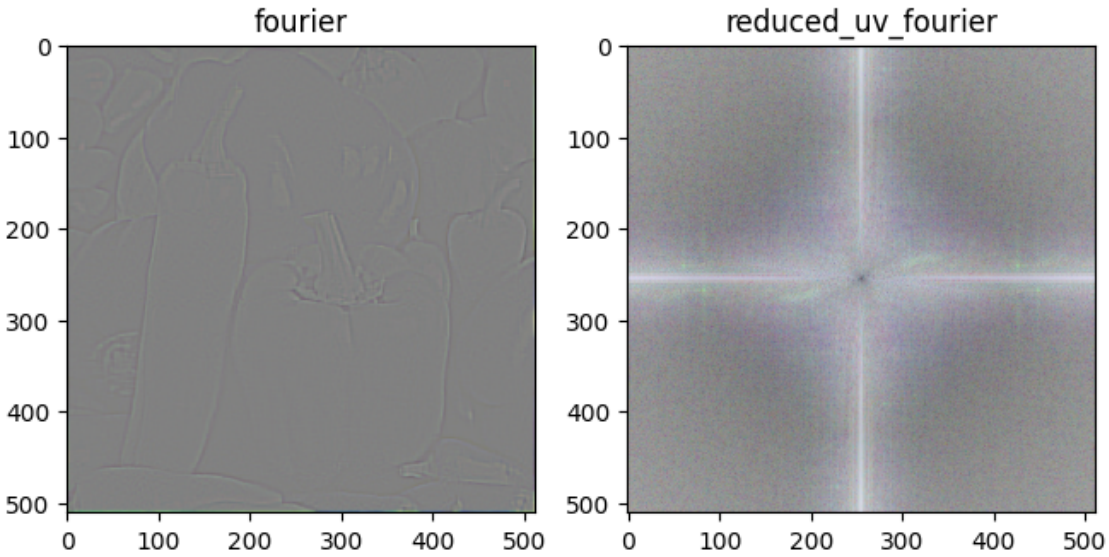


شکل ۳-۵: تصویر های پایین گذر و بالاگذر اعمال شده روی تصویر فلفل

خالی آن ها در بزرگ کردن، با درونیابی پر میشود که باعث بی کیفیت شدن تصویر میشود. هرچند که سعی شده با فیلتر های مختلف در حوزه فرکانس، شدن این اتفاق کمتر شود. زیاد شده آن تا ۲.۲ در تصویر ۳-۱۴ و کم شده آن تا ۱، در تصویر ۳-۱۵ مشخص است. میتوانیم متناظرا افت و افزایش کیفیت و بخصوص کاهش نویز را در این دو مقایسه شاهد باشیم.

۳-۲-۴ تغییر k

این پارامتر در بخش بازگردانی قرار دارد و تغییری دیگر روی حجم تصویر ندارد. برای این پارامتر در ابتدا مقدار $۹ * (۱۰^{-۴})$ و سپس مقدار $۹ * (۱۰^{-۸})$ در نظر میگیریم و آن را با حالت $۹ * (۱۰^{-۶})$ که پیشفرض است مقایسه میکنیم. این دو تصویر به ترتیب در ۳-۱۶ و ۳-۱۷ قابل مشاهده و مقایسه اند. میبینیم که در تصویر اول، شدن شارپ شدن زیاد شده و تصویر از حالت اصلی خارج شده و پر از نویز است. این تصویر اصلا تصویر خوبی نیست. در مقابل شکل ۳-۱۷ را داریم که جزئیات بسیار ناقصی دارد و مقدار زیادی تار است، چرا که گویا اصلا روی تصویر پایین گذر فیلتر شارپ کردنی اعمال نشده؛ این تصویر نیز عیب دارد و به خوبی تصویر پیشفرض نیست.



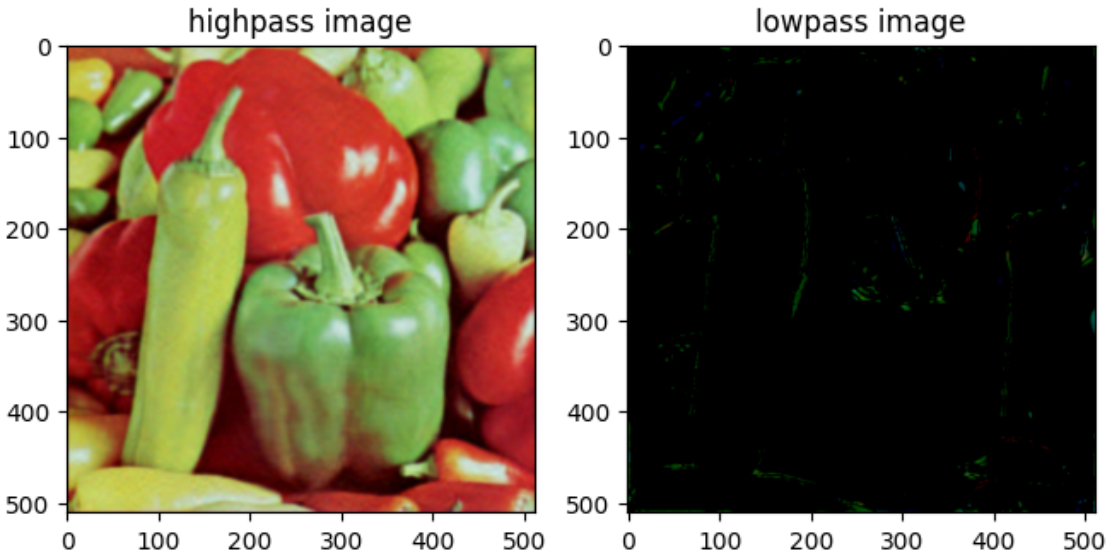
شکل ۳-۶: تصاویر کاهش داده شده در حوزه فرکانس و حوزه مکان

۳-۲-۵ تغییر ضریب lowpass

در ابتدا این ضریب را تا۱۷۵ زیاد میکنیم. میبینیم که نویز های تصویر دیگر از بین نمیروند و با شارپ کردن، زیاد تر هم میشوند. این تصویر در ۳-۱۸ قابل مشاهده است. این تصویر به وضوح تصویر خوبی به حساب نمی‌آیند. در ادامه ضریب را تا ۳۵ کم میکنیم و میبینیم که شدیدا تصویر تار میشود و فیلتر شارپ کردن هم نمیتواند آر را شارپ کند. این تصویر نیز جزئیات ندارد و تصویر مناسبی نیست.

۳-۲-۶ تغییر ضریب highpass

ابتدا ضریب را تا ۱۰۵ زیاد میکنیم و سپس آن را تا ۵ کم میکنیم. این دو تصویر در ۳-۲۰ و ۳-۲۱ ذخیره میکنیم و قابل مشاهده میباشند. در حالت اول، تفاوت عملا قابل دیدن نیست، تنها یک مقدار خیلی کوچک از شارپ بودن لبه های فلفل ها کم شده است.در مقابل، تغییر تصویر در حالت مخالف کاملا واضح است. لبه ها شارپ و پر رنگ شده اند و آنقدر پررنگ شده اند که در بعضی نقاط به حداکثر مقدار خود رسیده و مقدار ۲۵۵ به خود گرفته اند که اغراق شده است و منظره بصری جالبی ندارد. این دو تصویر نیز خیلی مناسب نمی‌باشند.



شکل ۳-۷: تصاویر بالا و پایین گذر

۳-۳ مقایسه حجم در تنظیمات متفاوت

این قسمت نیز به ۳ قسمت تقسیم میشود و در هر یک تغییر حجم فایل ذخیره شده بررسی میشود و نمودار آن نمایش داده میشود. دقت کنید در تمام نمودار ها حجم بر حسب کیلوبایت است.

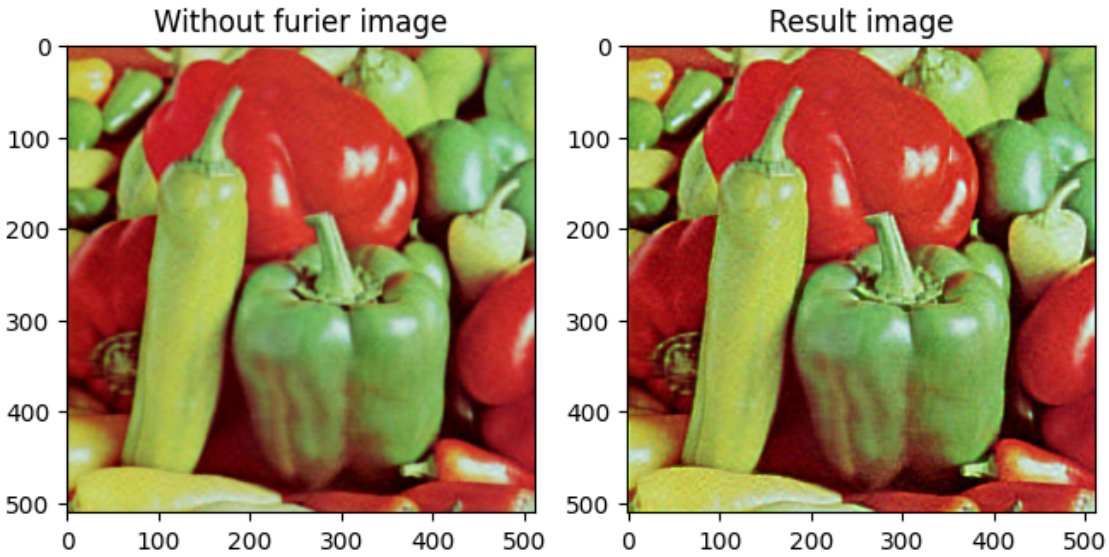
۳-۳-۱ تغییر تصویر ۱۶ بیتی به ۸ بیتی

در اینجا نمودار ۳ حالت مختلف ۱۶ بیتی، ۸ بیتی با ذخیره png و ۸ بیتی با ذخیره jpg بررسی میشود. این مقایسه در شکل ۲۲ قابل مشاهده است.(مقادیر بر حسب کیلوبایت هستند)

۳-۳-۲ تغیر comp و threshhold

در اینجا comp را از ۰ تا ۸۰۰۰ و threshhold را از ۵۱۲ تا ۰ تغییر میدهیم و تغییر حجم را بر اساس آن اندازه میگیریم و نموداش را رسم میکنیم. نمودار برای comp در شکل ۳-۲۳ قابل مشاهده است.

از آنجا که نمودار بنظر نمایی است، ما نمودار comp بر حسب لگاریتم حجم را نیز رسم میکنیم و بعنوان ۳-۲۴ نمایش میدهیم.



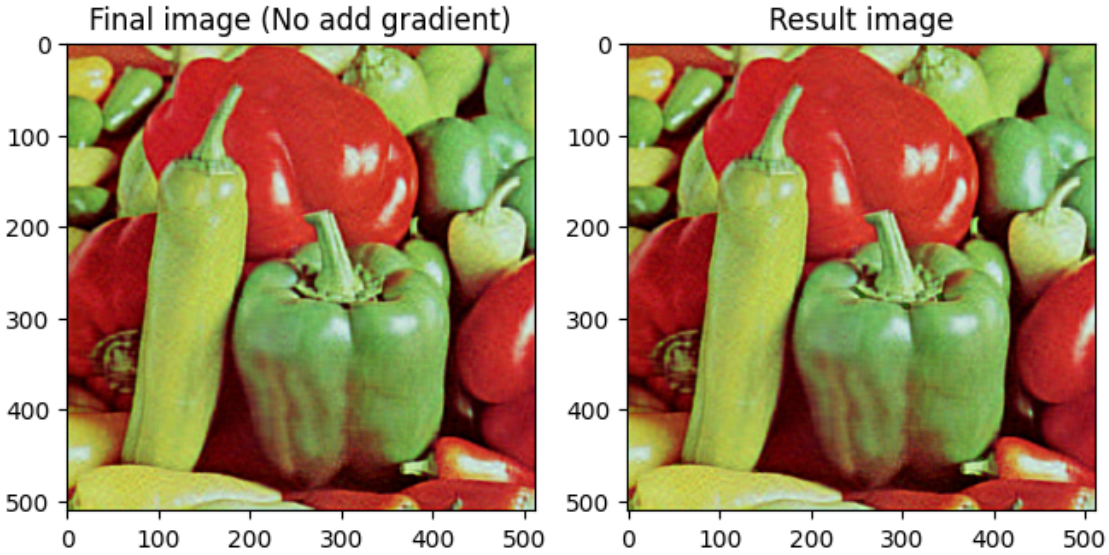
شکل ۳-۸: تصویر بعد از فیلتر های مختلف و تصویر خام

ما میتوانیم از این تصویر و برازش خطی آن که نشان داده شده است، نتیجه بگیریم که حجم فایل و comp و به نوعی کیفیت تصویر، رابطه لگاریتمی دارند. این مسئله از نمایی بودن الگوی مقادیر تکین ماتریس تصویر ناشی میشود.

حال میخواهیم حد روی مقدار مقادیر ویژه نگذاریم و تنها از تعدادشان کم کنیم. در این آزمایش، threshhold را تغییر میدهیم و حجم فایل را بر اساس آن رسم میکنیم. این نمودار را بعنوان شکل ۳-۲۵ نمایش میدهیم. این مقدار ماکزیمم ۵۱۲ دارد (به علت ابعاد تصویر) پس از ۵۱۲ شروع کرده و ۱۶ تا ۱۶ تا از آن کم میکنیم.در اینجا همانطور که انتظار میرفت شروع کم شدن حجم از ۳۵۲ بود به علت scale شدن تصویر. و بعد از آن نیز به طور خطی و منظرم کم شد، چراکه در هر بار مقدار مشابهی از ماتریس ها را داریم جدا میکنیم و به همان نسبت قبل حجم کم میشود. به نظر اینطور می‌آید که روش اول یعنی استفاده از comp، روش بهتری برای کنترل کیفیت تصویر و روش دوم یعنی استفاده از threshhold، روش مناسب روش برای کنترل حجم تصاویر است.

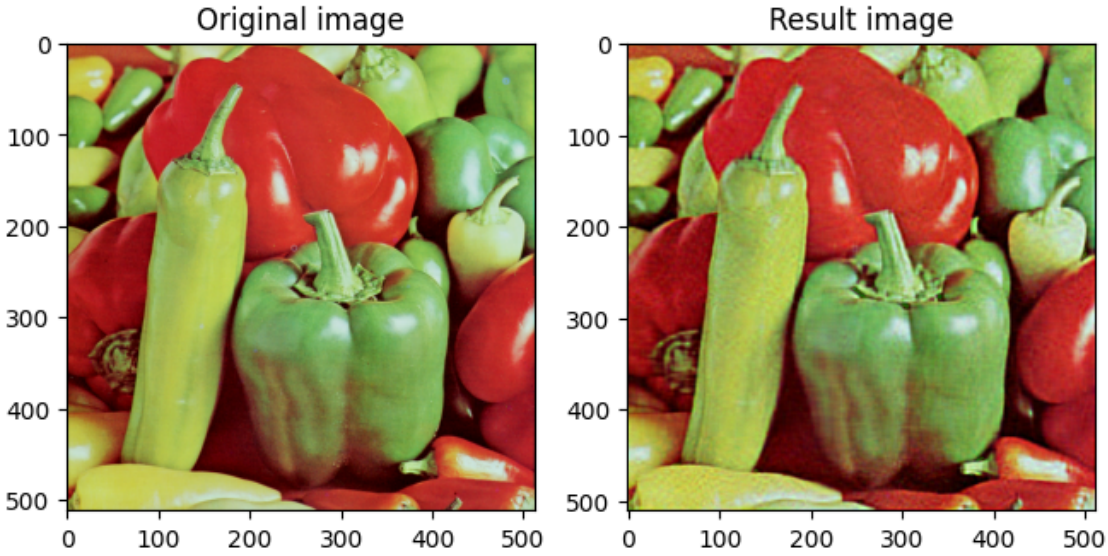
۳-۳-۳ تغیر scale

حال scale را از ۵.۰ تا ۳ تغییر میدهیم و تغییر حجم را بر حسب آن رسم میکنیم. این نمودار ۳-۲۶ است. این نمودار نمایی است، زیرا کم شدن ابعاد ماتریس، به نرخ نمایی از مقدار تکین ها کم میکند و از طرفی ما معیارمان برای جدا سازی عوامل اندازه مقدار های تکین هست. پس بنابراین،

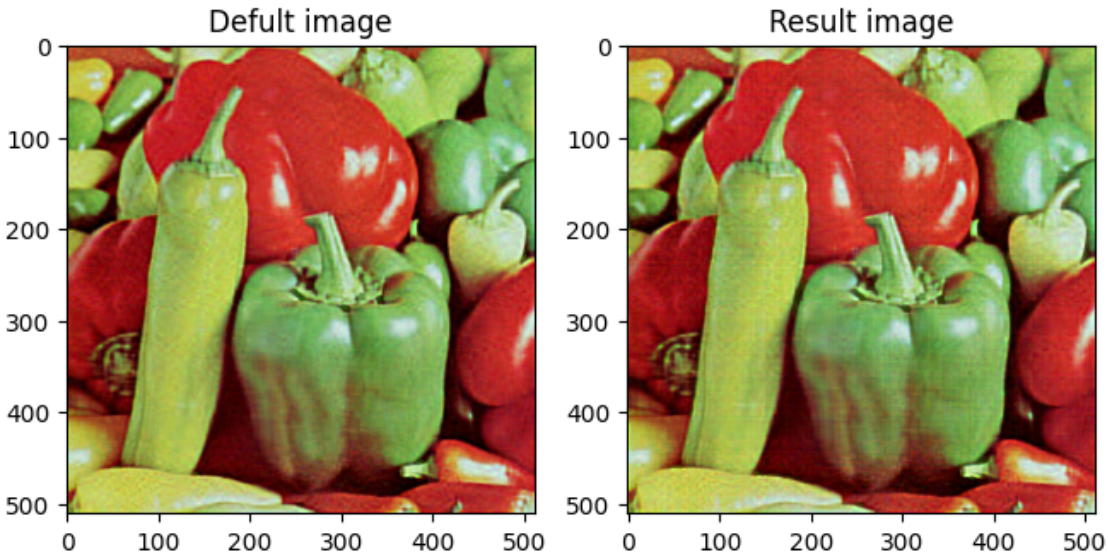


شکل ۳-۹: تصویر بعد و قبل از اضافه کردن عامل بالاگذر به تصویر

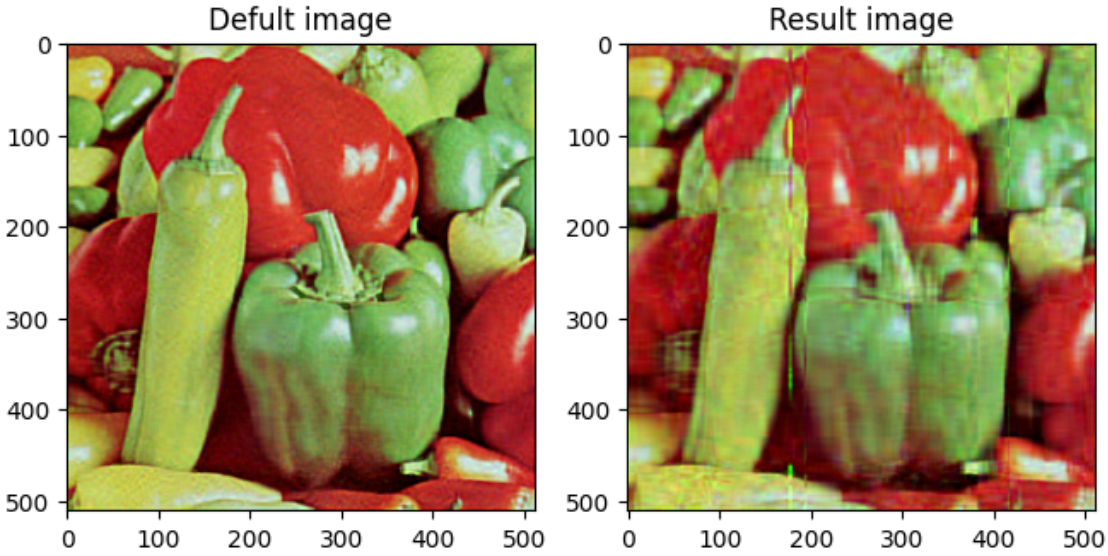
بصورت نمایی کم میشود حجم و نمودار نمایی آن را رسم و در ۳-۲۷ نشان میدهیم. از برازش خطی آن میتوان فهمید که احتمالا فرضیه مان در این مورد هم درست است.



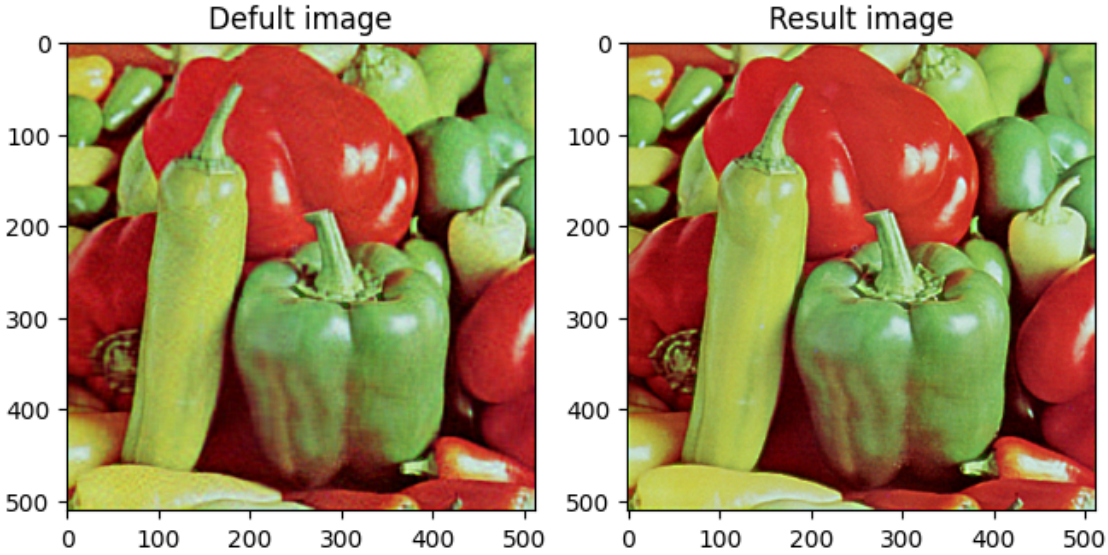
شکل ۳-۱۰: تصویر اولیه و نهائی



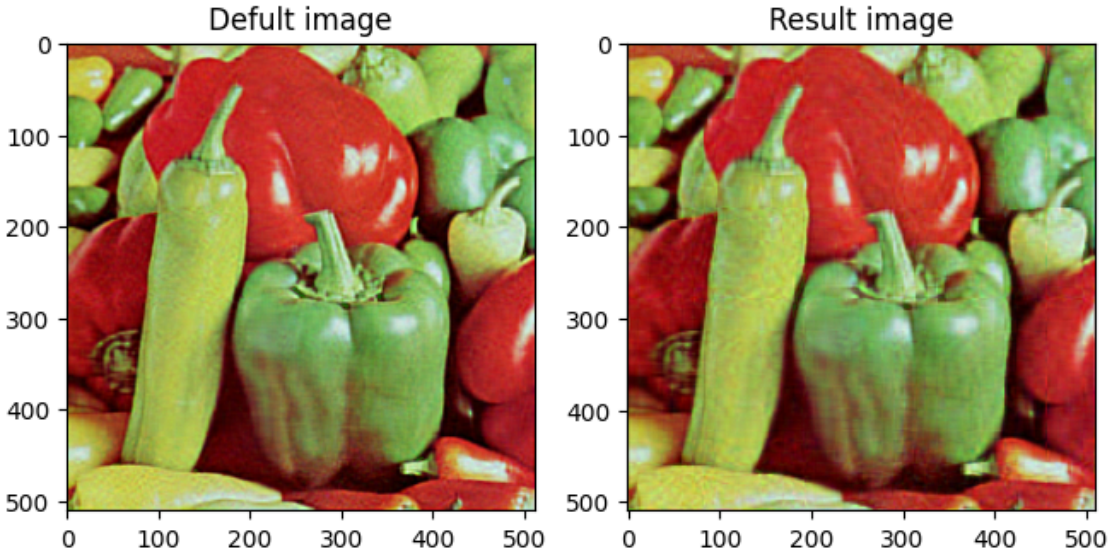
شکل ۳-۱۱: مقایسه تصویر حاصل از ذخیره تصویر ۱۶ و ۸ بیتی



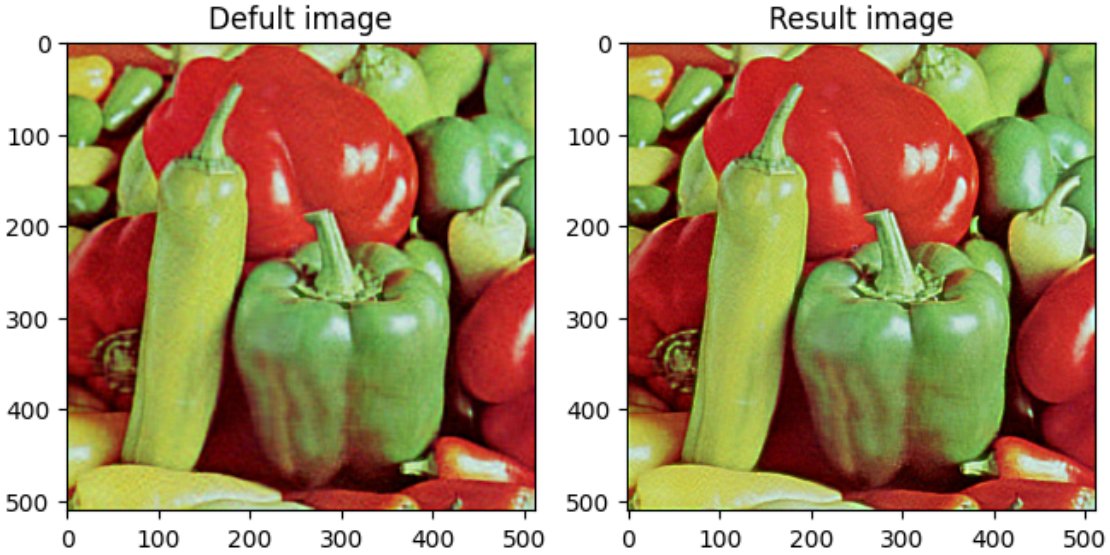
شکل ۳-۱۲: مقایسه تصویر حاصل از comp ۴۵۰ و ۱۲۰۰



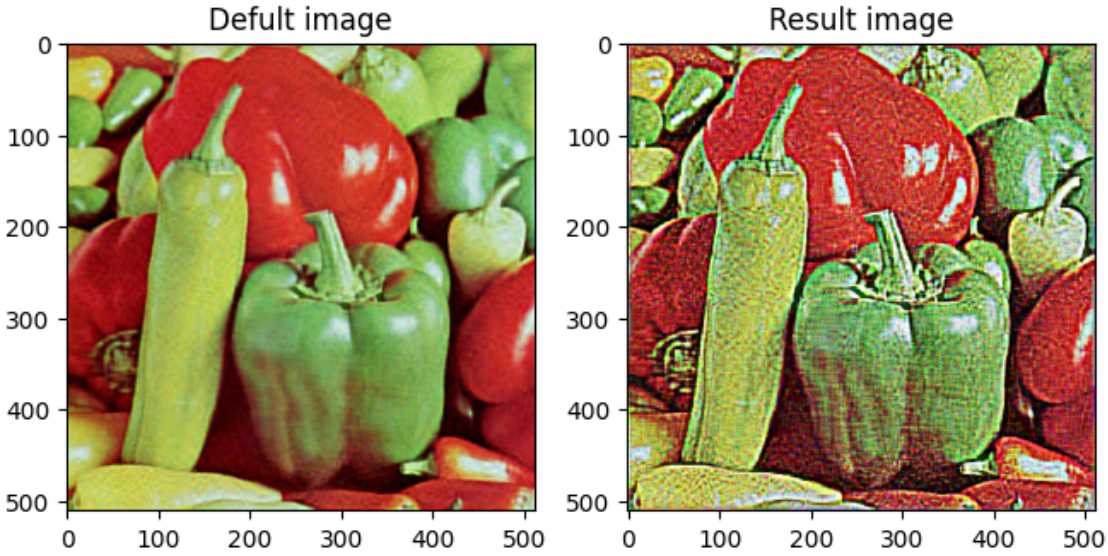
شکل ۳-۱۳: مقایسه تصویر حاصل از comp ۴۵۰ و ۱۵۰



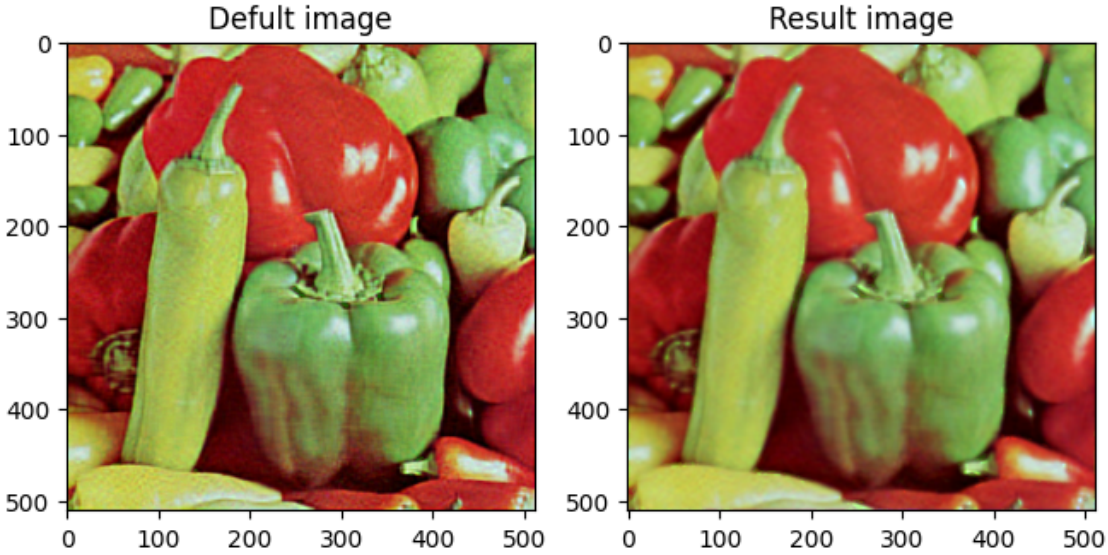
شکل ۳-۱۴: مقایسه تصویر حاصل از scale ۴.۱ و ۲.۲



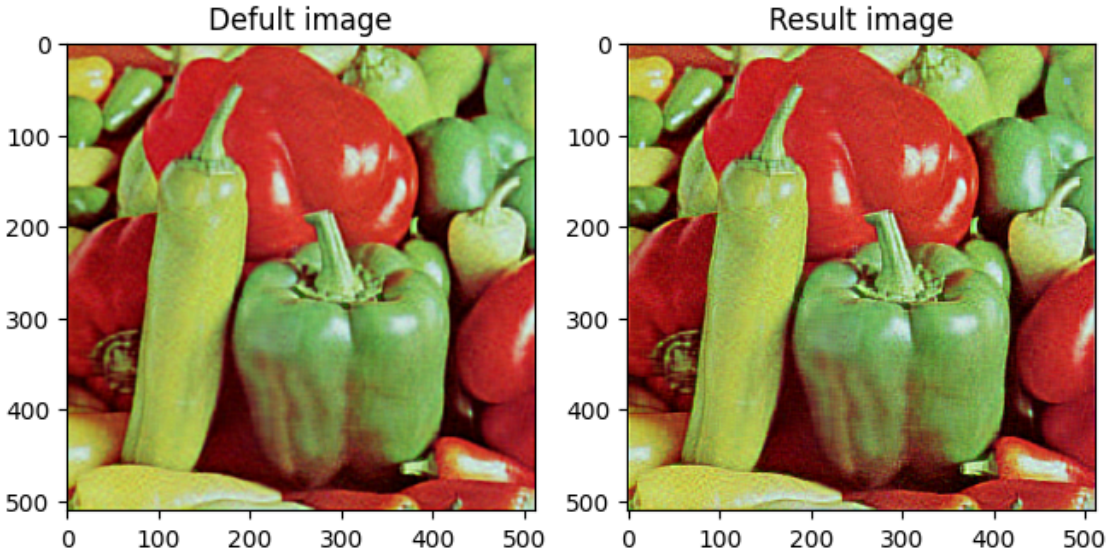
شکل ۳-۱۵: مقایسه تصویر حاصل از scale ۴.۱ و ۱



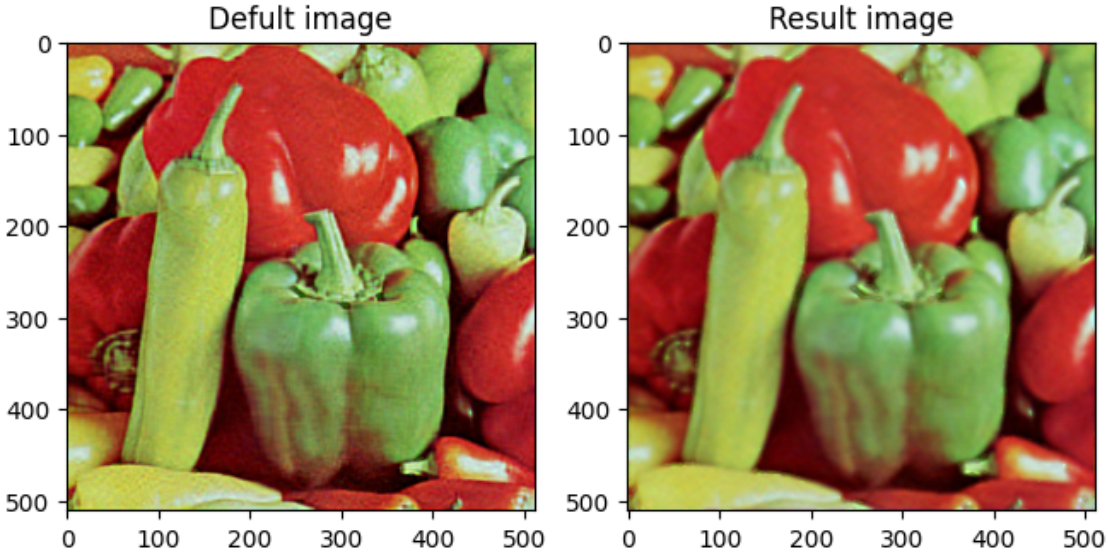
شکل ۳-۱۶: مقایسه تصویر حاصل از k زیاد و متعادل



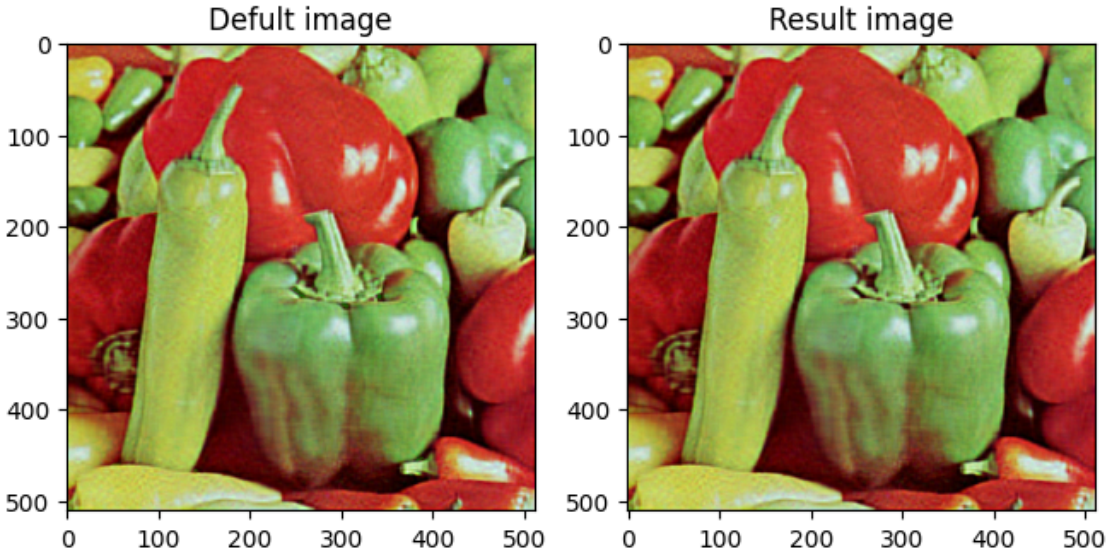
شکل ۳-۱۷: مقایسه تصویر حاصل از k کم و متعادل



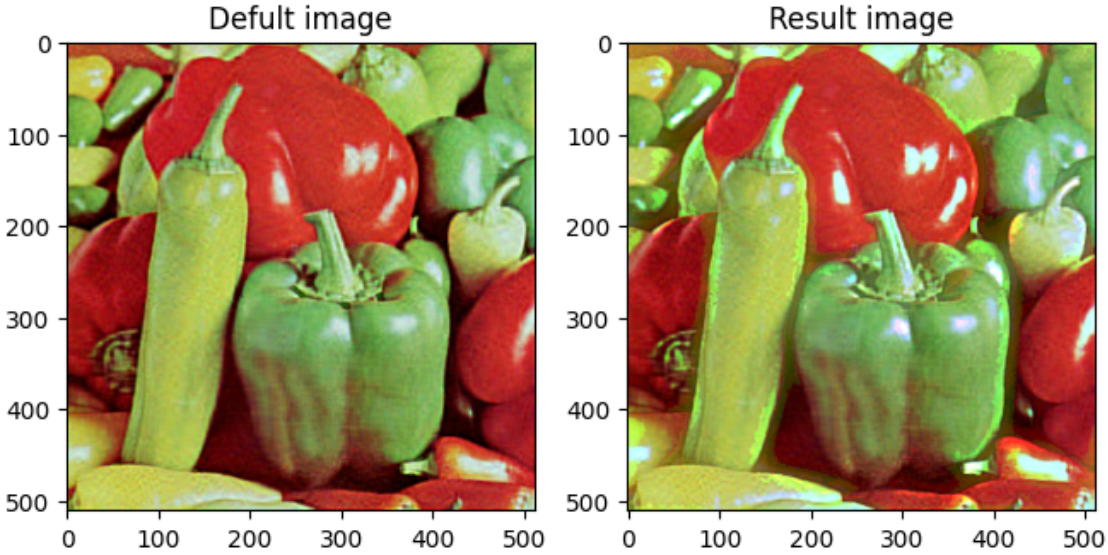
شکل ۳-۱۸: مقایسه تصویر حاصل از lowpass ۱۷۵ و ۷۵



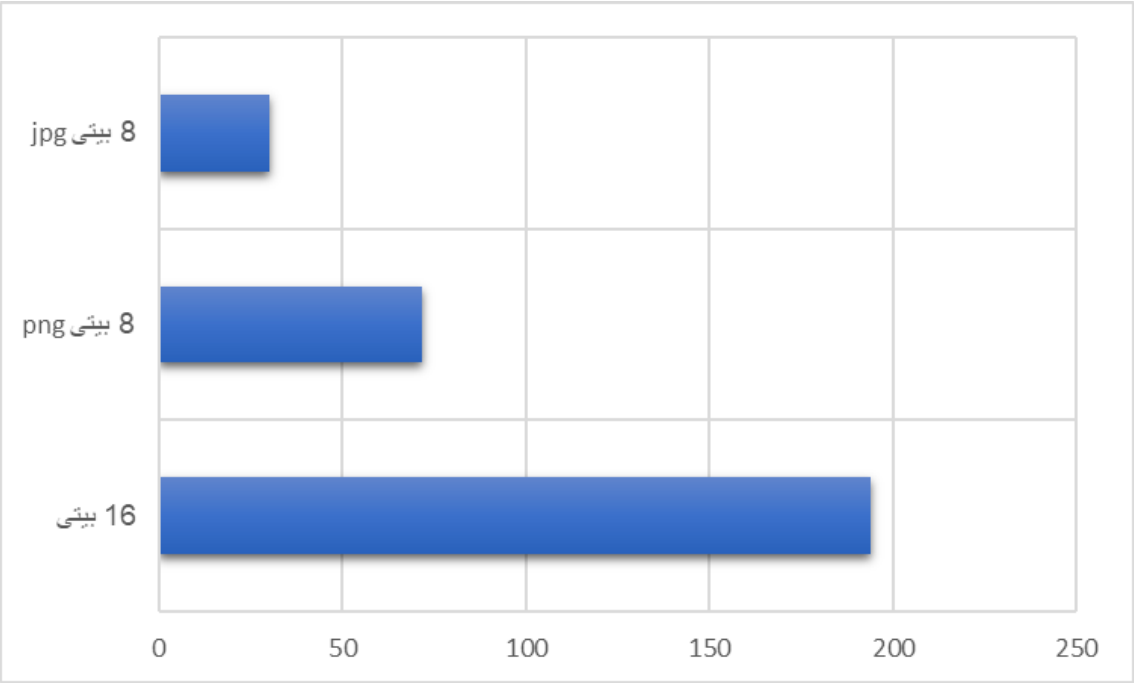
شکل ۳-۱۹: مقایسه تصویر حاصل از lowpass ۳۵ و ۷۵



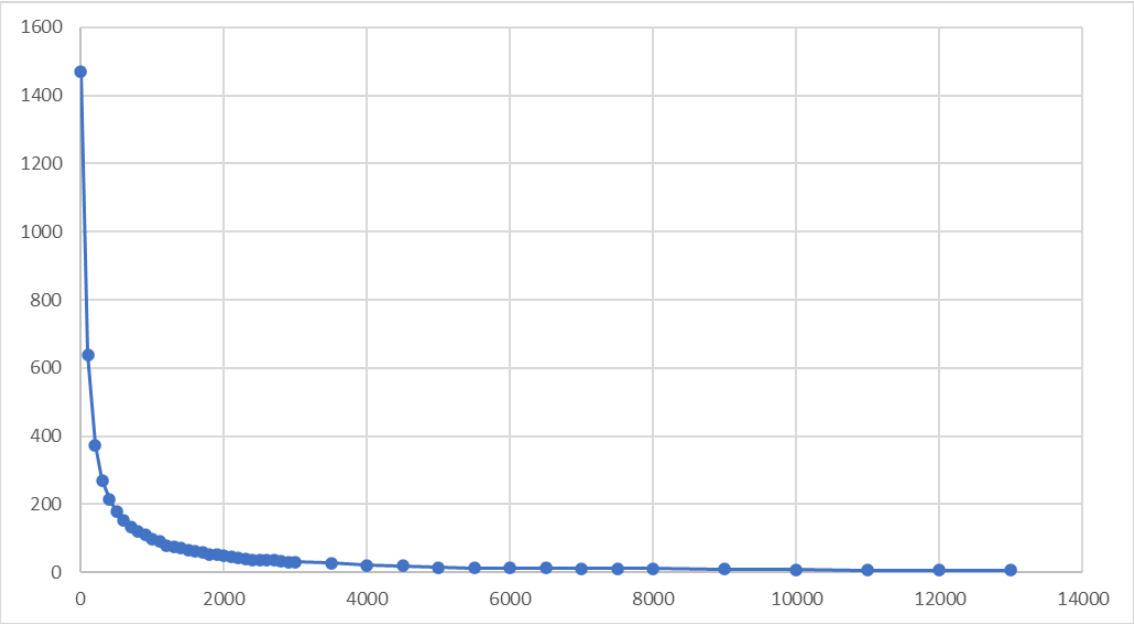
شکل ۳-۲۰: مقایسه تصویر حاصل از highpass ۱۰۵ و ۲۱



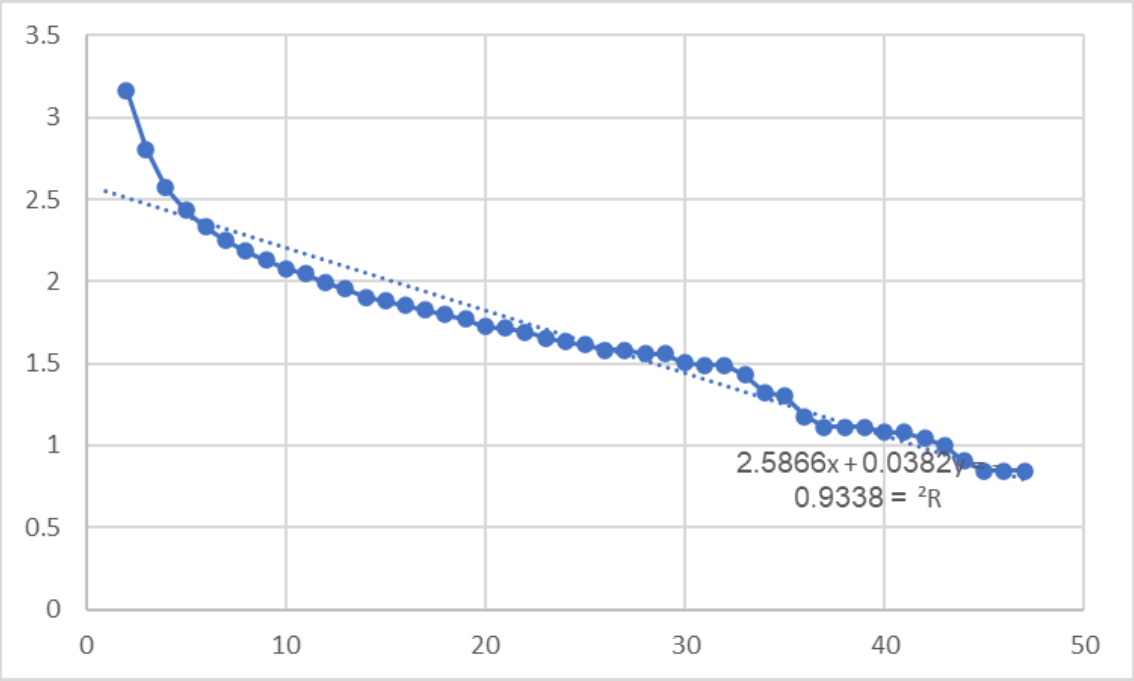
شکل ۳-۲۱: مقایسه تصویر حاصل از highpass ۵ و ۲۱



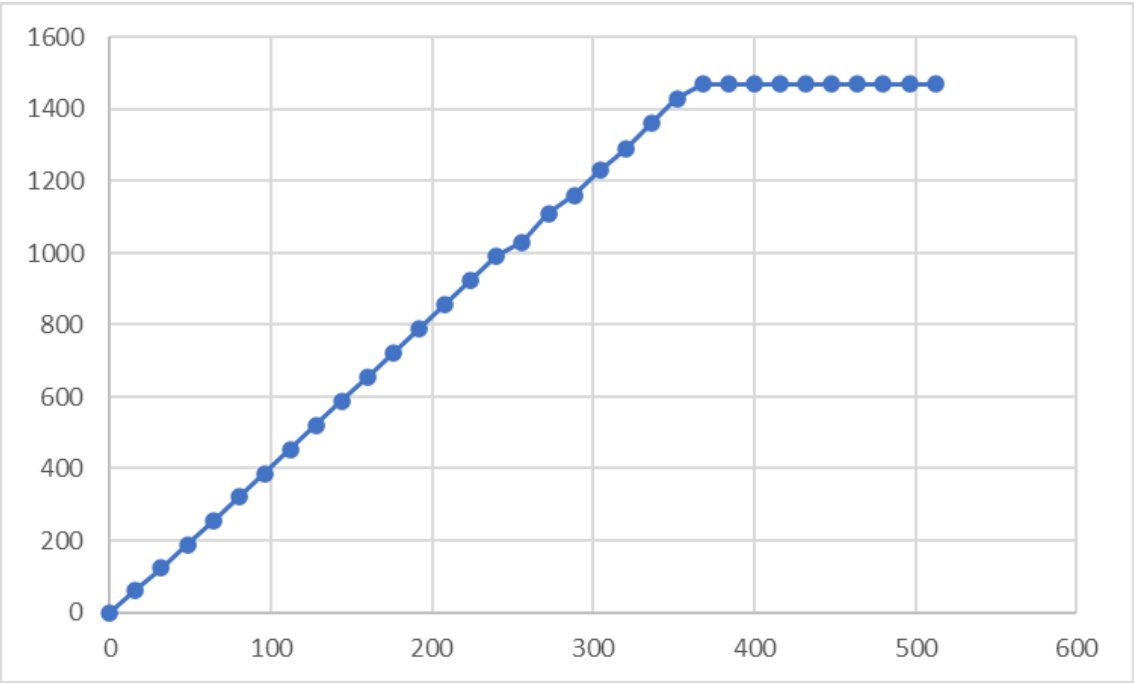
شکل ۳-۲۲: مقایسه حجم اشغال شده توسط فایل ذخیره شده در ۳ حالت ذخیره متفاوت



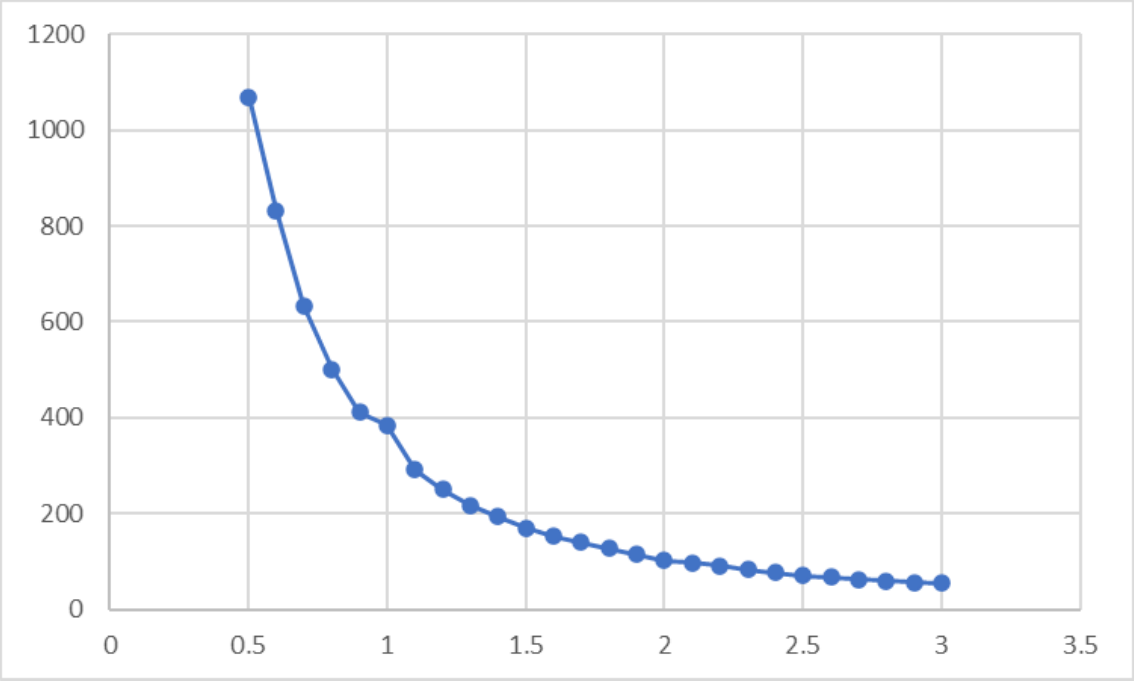
شکل ۳-۲۳: نمودار حجم فایل بر حسب comp



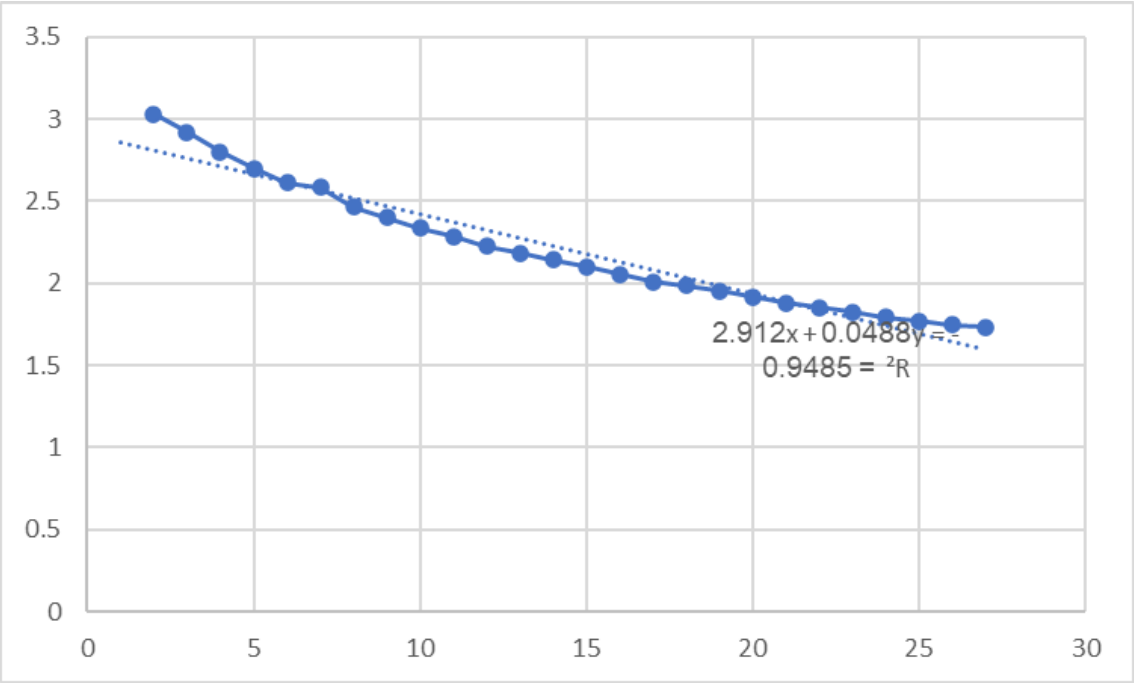
شکل ۳-۲۴: نمودار لگاریتم حجم فایل بر حسب comp



شکل ۳-۲۵: نمودار لگاریتم حجم فایل بر threshhold



شکل ۳-۲۶: نمودار scale بر حسب حجم فایل



شکل ۳-۲۷: نمودار scale بر حسب لگاریتم حجم فایل

فصل ۴

بحث

به چه دلیل از تبدیل فوریه برای افزایش کیفیت تصویر استفاده شد و نه کانوولوشن در حوزه مکان؟

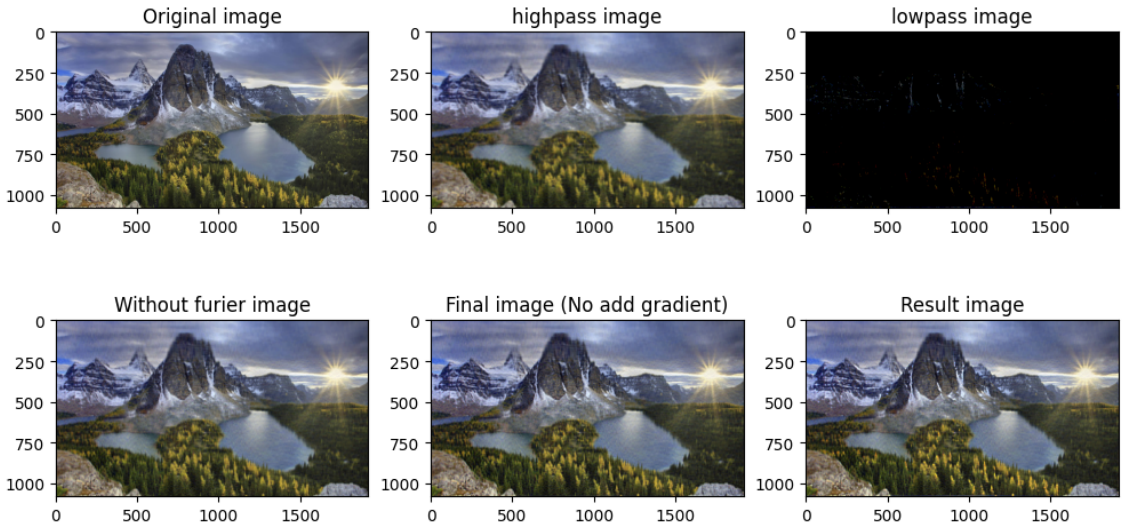
استفاده از فیلتر های مختلف در حوزه فرکانس به شدت سرعت پردازش را تسریع میکند و همانطور که گفته شد، این یکی از اهداف مهم پروژه بود.

آیا این برنامه روی تصاویر با کیفیت بالا هم جوابگوست؟

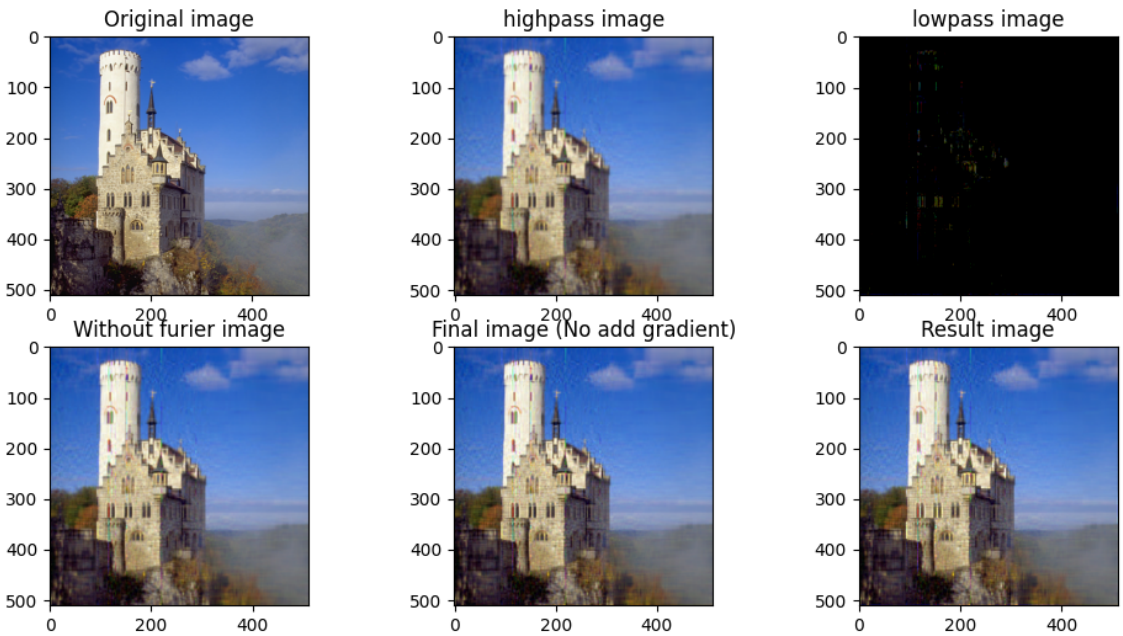
قطعا؛ این برنامه تصاویر با کیفیت های بالا را با ضریب بسیار بهتری نیز فشرده میکند، اما از طرفی این فشرده کردن میتواند بسیار زمان بر تر از عکس هایی با کیفیت معمولی باشد. برای مثال میتوانید تصویر ۴-۱ را مشاهده کنید. این تصویر در مقیاس حدود یک هفتم کوچکتر شده بدون اینکه لطمه بزرگی به تصویر وارد شود.

آیا این برنامه همیشه خوب عمل میکند؟

متاسفانه اینطور نیست. برای تصاویری خطوط عمودی و افقی زیادی دارند، حذف نقاط تکین مهم زودتر صورت گرفته و باعث از بین رفتن خطوط مهمی میشود. برای مثال میتوانید تصویر قلعه ۴-۲ را ببینید که اگرچه از حجم تصویر بسیار زیاد کاسته نشده، ولی باعث از بین رفتن نمای کلی تصویر شده است.



شکل ۴-۱: تصویر با کیفیت بالا



شکل ۴-۲: تصویر تخریب شده قلعه لیختن اشتاین

فصل ۵

نتیجه‌گیری

در این پروژه، توانستیم تا مقدار خوبی به هدفمان که در فصل مقدمه گفته شد، نزدیک بشویم. ایجاد یک بستر سریع و قابل دسترسی که بتواند باعث ذخیره یک تصویر با حجم بسیار کمتر از تصویر اصلی شود و در مدت نسبتاً کمی بتواند به‌صورتی بازگردانی شود که بخش‌های مهم تصویر حذف نشوند و مقدار کاهش کیفیت قابل قبولی داشته باشد، اهداف اصلی این پروژه بود. که دیده شد، توانستیم برای بازه گسترده‌ای از تصاویر، به این هدف دست پیدا کنیم. که دیدیم، با استفاده از تجزیه مقادیر تکین، توانستیم به‌طور مناسبی عوامل در تصویر را پیدا کنیم و با حذف عواملی که اهمیت کمتری دارند، حجم تصویر را بسیار کمتر کنیم، با وجود اینکه کمترین جزئیات را از آن حذف کنیم. این مورد به خوبی یکی از کاربردهای تجزیه مقادیر تکین را در دانش پردازش تصاویر نشان می‌دهد. سپس برای افزایش کیفیت، ملزم به استفاده از کانولوشن شدیم. اما با توجه به پیچیدگی محاسبه در حوزه مکان، توانستیم با صرف مقداری محاسبه آن را به حوزه فرکانس ببریم و تنها با ضرب عادی ماتریس در حوزه فرکانس، کانولوشن لازم را انجام دهیم. این هم یک مثال خوب از کاربردهای بی شمار محاسبات ماتریسی و تبدیل فوریه ماتریسی در پردازش تصاویر و کمک به بهبود تصاویر به شمار می‌آید. با این وجود، امکان فشردگی شدیدتر این تصاویر با تکنیک‌های گوناگونی وجود دارد و این پروژه صرفاً گوشه از یک فشردگی چندین برابری با از دست رفتن اطلاعات می‌باشد.

فصل ۶

مراجع

- https://www.techtarget.com/whatis/definition/image-compression
- https://www.frankcleary.com/svd/
- .html/۳۱۹۴۴https://python-forum.io/thread-
- /svd-based-image-compression۹۴https://github.com/Harkirat
- /fourier.htm۲https://homepages.inf.ed.ac.uk/rbf/HIPR
- /opencv-fast-fourier-transform-fft-۱۵/۰۶/۲۰۲۰https://pyimagesearch.com/ for-blur-detection-in-images-and-video-streams/
- https://www.shrink.media/blog/compression-algorithm
- https://www.frankcleary.com/svdimage/

Abstract

The excessive occupation of storage space is the inherent nature of image processing knowledge. Since the expansion of this knowledge, practitioners in this field have sought effective methods to reduce the volume occupied during image storage using various approaches. Over the years, different methods have been proposed, each incorporating various combinations of several primary techniques. Among the most useful are methods that optimally compress images using techniques such as image blocking, Fourier and cosine transforms, as well as Huffman coding. These compressions may occur with or without the loss of information within the images. In this project, a compression method is introduced that, through the use of singular value decomposition and matrix Fourier transforms, achieves compression resulting in a significant reduction in volume alongside some loss of information and overall quality. This volume reduction can prove notably beneficial in systems requiring the preservation of image dimensions while occupying minimal space for image storage.

Keywords: Singular Value Decomposition, Fast Fourier Transform, Image compression, Image Processing

Sharif University of Technology

Department of Mathematical science

Matrix computations project

Lossy image compression using Singular Value Decompostion and Fast Fourier Transform

By:

Nima Kelidari

Professor:

Dr. Razvan

January 2024