به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

تجزیه تحلیل سیستم‌ها

پروژه پایانی

نوید اکبری

سیده مهتاب عامری

شماره دانشجویی

810894034

810895023

تیر ماه 1396

فهرست

چکیده

در طی این پروژه برای آشنایی بیشتر با مفهوم پردازش تصویر و رفع نویزها با دو مدل نویز سینوسی و حرکتی آشنا شدیم و با انتخاب متغیرهای مناسب اقدام به برطرف کردن این نویزها کردیم و در ادامه با مفهوم فشرده سازی عکس بصورت jpeg و فشرده زدایی تصویر آشنا شدیم و تصویر دینویز شده از سوال اول را ابتدا فشرده کردیم و از حالت فشرده خارج ساختیم.

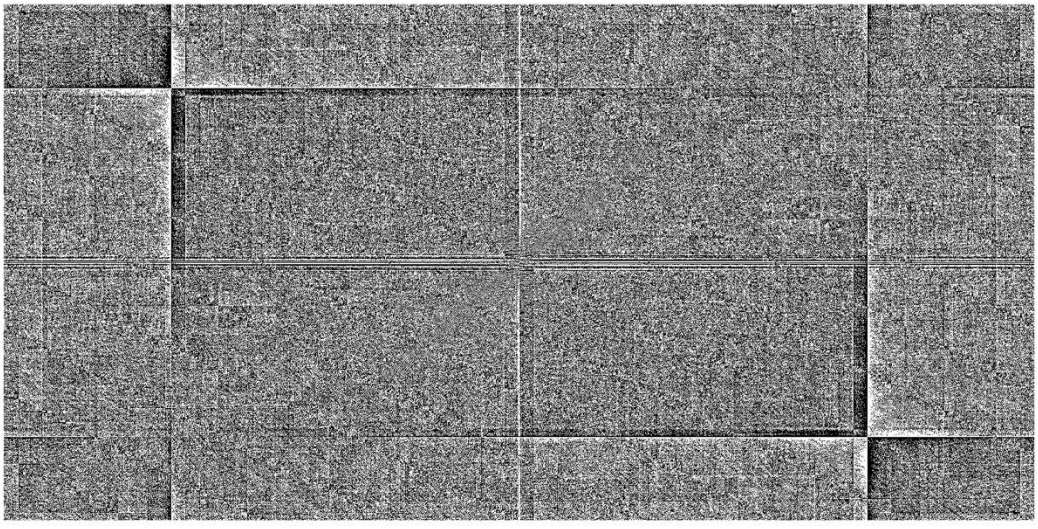
سوال 1

همانطور که در صورت پروژه گفته شده تصویر داده شده دارای نویزهای سینوسی و نویزهای حرکتی است.

که هر مرحله به ترتیب در ادامه توضیح داده می‌شود.

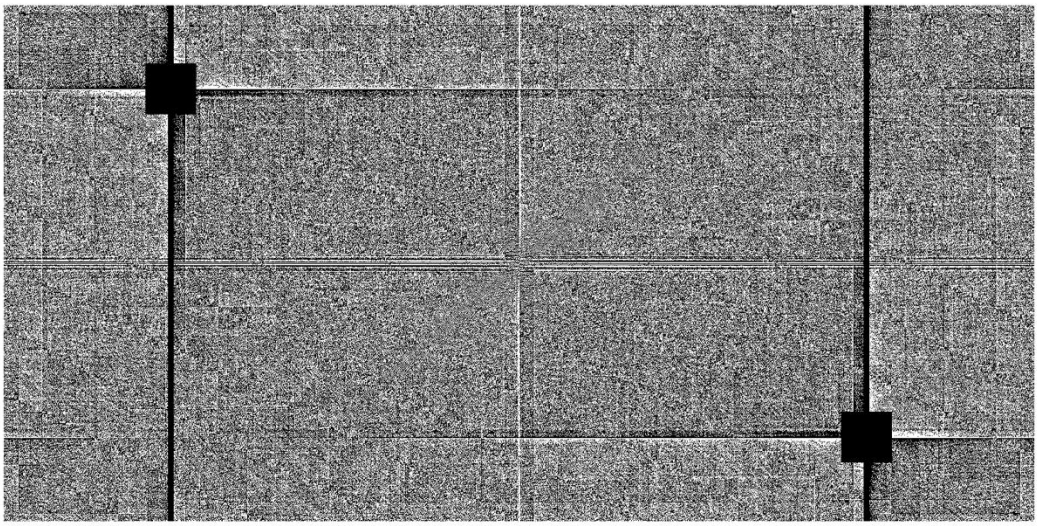
شکل شماره ۱ تصویر اصلی

برای برطرف کردن نویز سینوسی از تصویر اصلی که fft2 و fftshift می‌گیریم تصویر به دست آمده دارای ۲ تقاطع می‌باشد و باید آن نقاط را سیاه کنیم.



شکل شماره 2 تبدیل فوریه تصویر اصلی

برای از بین بردن نقاط تقاطع مختصات نقاط را بدست آوردیم و آن‌ها را به صفر تغییر دادیم.



شکل شماره ۳ تبدیل فوریه شکل پس از برطرف کردن نویز سینوسی

شکل شماره ۴ تصویر اصلی پس از برطرف کردن نویز سینوسی

شکل حاصل بعد از این تغییر به صورت زیر می‌باشد.

مرحله بعدی از بین بردن نویز حرکتی است که برای این کار از توابع fspecial و deconvwnr استفاده می‌کنیم. تابع fspecial دارای سه پارامتر می‌باشد که اولین پارامتر نشان‌دهنده نوع نویز است که ما از نوعmotion استفاده میکنیم پارامتر دوم نشان‌دهنده میزان حرکت و پارامتر سوم نشان‌دهنده

نوع حرکت می‌باشد. که زاویه حرکت به دلیل این که شکل اصلی به میزان ۳۰- یا ۳۳۰ ما باید عکس این زاویه یعنی ۱۵۰ درجه استفاده کنیم و میزان حرکت با امتحان کردن به دست آمده است. شکل به دست آمده از این کار برابر شکل زیر می‌باشد.

شکل شماره ۵ شکل حاصل از برطرف کردن نویز حرکتی

مقدار MSE و SNR به دست آمده برابر مقادیر زیر می‌باشد.

SNR = 0.0266 MSE = 101.4800

همانطور که در شکل پیداست پلاک ماشین برابر "C4RWOW" می‌باشد و تعداد سرنشینان آن برابر یک یعنی خود راننده می‌باشد.

سوال 2

هدف کلی در این سوال آشنایی با روند فشرده سازی jpeg در عکس‌هاست. برای توضیح بهتر روند کار آنرا به چند مرحله تقسیم کرده‌ایم که در کد هم این شماره‌ها رعایت شده‌اند و در کد هر مرحله در سکشنی جدا با همین شماره نوشته شده.

.1

در ابتدا بعد از خواندن عکس، چون میخواهیم عکس را به زیر تصویرهای ۸ * ۸ تقسیم کنیم چک میکنیم اگر ابعاد عکس مناسب نبود با اضافه کردن ستون یا سطرهای صفر به عکس ابعاد آنرا تنظیم میکنیم. برای حساب کردن خطا این عکس را با عنوان SourceImage ذخیره میکنیم.

.2

سپس روشنایی عکس را تنظیم میکنیم. به اینصورت که از تمامی درایه ها ۱۲۸ واحد کم میکنیم و چون میزان روشنایی باید بین ۰ تا ۲۵۵ باشد درایه‌هایی که منفی شدن را ۰ در نظر میگیریم.

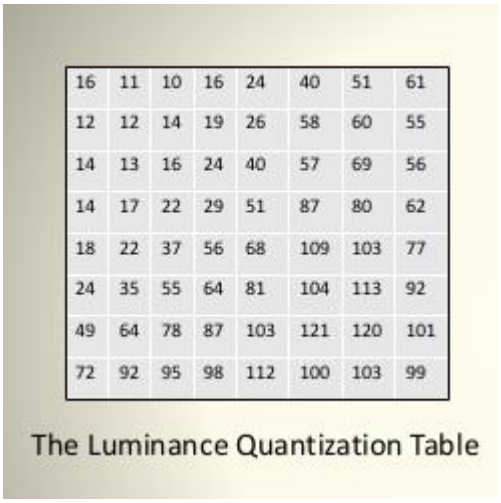
.3

برای تمامی زیرتصویرهای ۸*۸ تبدیل گسسته کسینوسی را با کمک دستور dct2 در متلب انجام میدهیم تا تصویر را برای بخش کوانتیده شدن آماده کنیم. همچنین چون زیر تصویرهای ۸*۸ داریم از تبدیل دوبعدی استفاده میکنیم.

.4

حال برای کوانتایز کردن عکس از با استفاده از کیفیتی که کاربر انتخاب میکند و با کمک جدول The Luminance Quantization Table ماتریس ۸*۸ Q را تشکیل میدهیم و با تقسیم آن بر هر زیرتصویر عکس را کوانتیده میکنیم.

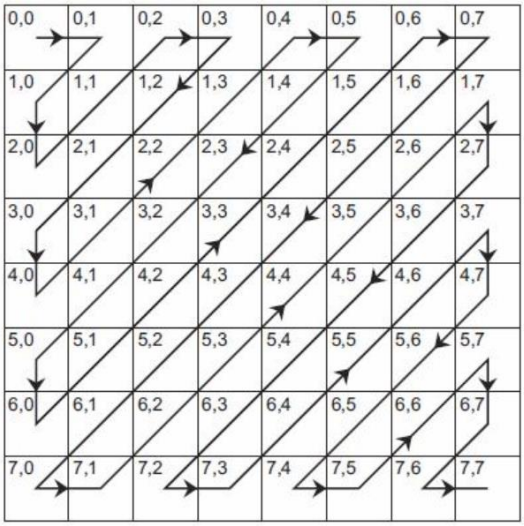
اگر کیفیت انتخابی ۵۰ باشد ماتریس Q برابر ماتریس Luminance Quantization میشود.



شکل ۶ ماتریس استفاده شده برای کوانتایز کردن

.5

در این مرحله هر کدام از زیرتصویر های ۸*۸ را با کمک الگوی زیگ زاگ به یک توالی عددی تبدیل میکنیم. و این توالی ها را جایگزین هر زیرتصویر میکنیم. سپس کل ماتریس توالی‌ها هم با کمک الگوی زیگزاگ به یک توالی عددی تبدیل میکنیم.



شکل ۷ الگوی زیگ زاگ **(Zig-Zag pattern)**

برای تبدیل کردن ماتریس توالی‌ها به بردار چون فرمت نگهداری آنها در برنامه بصورت cell بود ابتدا در یک ماتریس هم اندازه کمکی برای هر سلول یک شماره منظور کردیم و با کمک آن ماتریس اصلی را به توالیی از سلول ها تبدیل کردیم.

برای تبدیل یک ماتریس m*n به توالی عددی تابعی نوشته شده که در پوشه سوال قرار داده شده است.

.6

حال میخواهیم با کمک الگوریتم هافمن توالی عددی را به توالی باینری تبدیل کنیم.

در این قسمت ابتدا با محاسبه تعداد تکرار عناصر توالی و احتمال‌ها Huffman dictionary را تشکیل میدهیم تا هم برای تبدیل توالی عددی به باینری و هم برای برگرداندن توالی باینری به عددی از آن استفاده کنیم.

.7

در این قسمت با کمک دستور huffmanenco و دیکشنریی که تعریف کردیم توالی عددی که در quantized_vector ذخیره شده را به توالی باینری تبدیل میکنیم.

تا اینجا مراحل فشرده سازی تصویر کاملا طی شده و از این مرحله به بعد روند خارج کردن تصویر از حالت فشرده را آغاز میکنیم.

.8

در اولین مرحله خروج از حالت فشرده، توالی باینری را به کمک دستور huffmandeco و با کمک مرجع یا دیکشنری تعریف شده در مرحله ۶، به توالی عددی تبدیل میکنیم.

.9

حال با کمک بازگشت از الگوی زیگ‌زاگ این توالی عددی را به ماتریسی تبدیل میکنیم که در هر خانه‌ی آن توالی عددی زیرتصویرها قرار داشته باشد.

این الگو که عکس الگوی زیگ زاگ است در تابعی به نام invzigzag در پوشه سوال قرار دارد.

حال دوباره هرکدوم از توالی های زیرتصویرها را به حالت اولیه باز میگردانیم و دوباره زیرتصویرهای ۸*۸ را تشکیل میدهیم.

.10

در این مرحله با ضرب ماتریس Q در هر زیر تصویر، تصویر را دیکوانتایز میکنیم و عملا عکس مرحله ۴ را انجام میدهیم.

.11

از عکس بدست آمده از مرحله قبل عکس تبدیل گسسته کوسینوسی دوبعدی میگیریم. اینکار را با کمک دستور idct2 در متلب انجام میدهیم.

.12

در آخرین مرحله هم با اضافه کردن 128 میزان روشنایی تصویر را تنظیم میکنیم و عکس را از حالت فشرده خارج میکنیم.

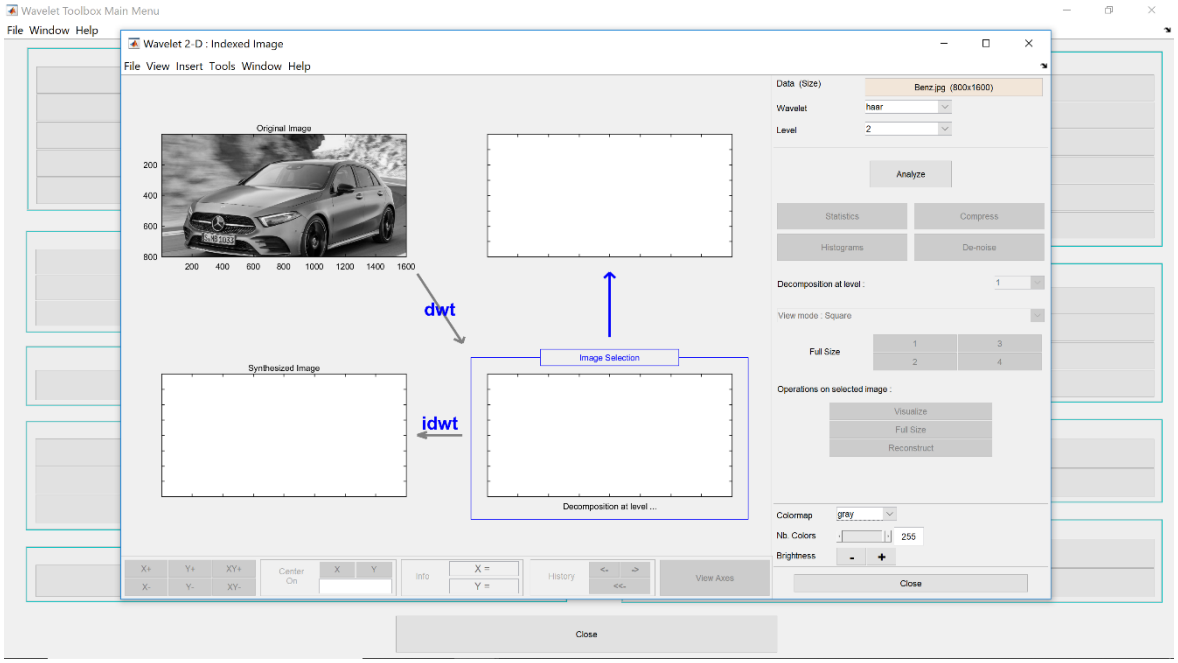
.13

در این قسمت عکس ابتدایی و عکسی که یکبار فشرده شده و از فشردگی خارج شده را نمایش میدهیم و همچنین با کمک تابع‌های داده شده در صورت پروژه میزان خطا را بدست می آوریم و نمایش میدهیم.

تصویر بدست آمده و تصویر ابتدایی کمی با هم تفاوت دارند و این موضوع به این دلیل است که هنگام استفاده از تبدیل گسسته کوسینوسی ممکن از یک سری از داده ها از دست بروند.

سوال سوم

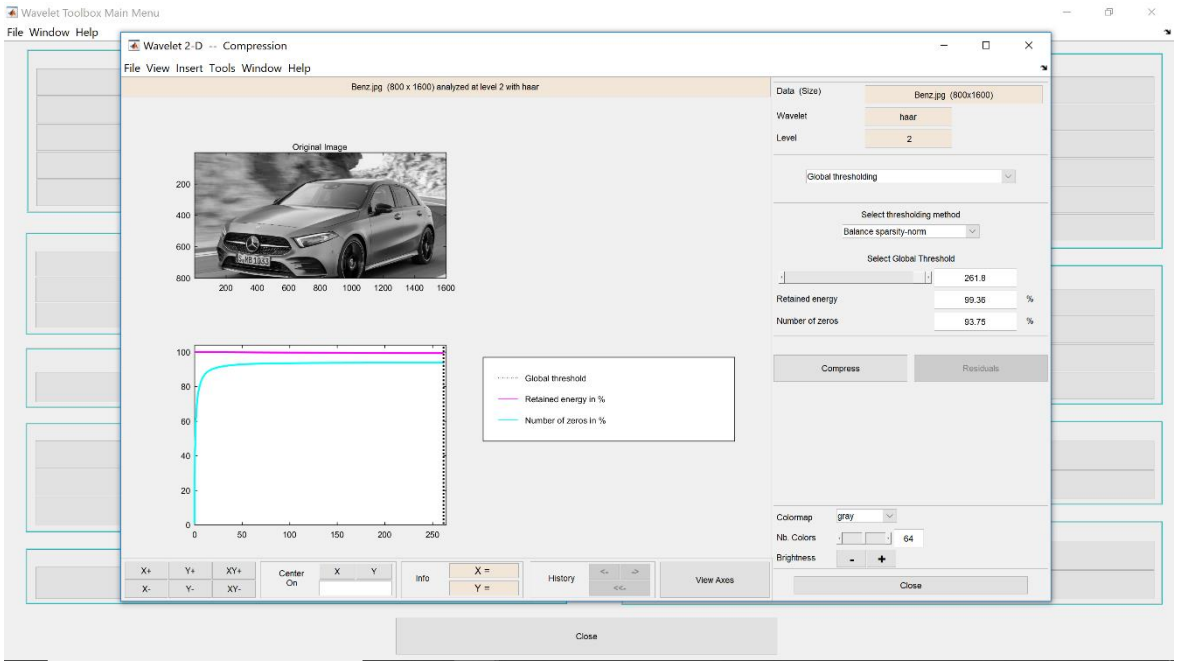
پس از باز کردن جعبه ابزار wavelet ، انتخاب گزینه wavelet 2-D در متلب و بارگذاری عکس مورد نظر صفحه به حالت زیر در میاید.



شکل ۸ بارگذاری عکس در **wavelet toolbox**

در قسمت بالا و سمت چپ میتوان نوع موجک را تغییر داد و پس از انتخاب سطح و موجک با زدن گزینه آنالیز میتوان تصویر را آنالیز نمود.

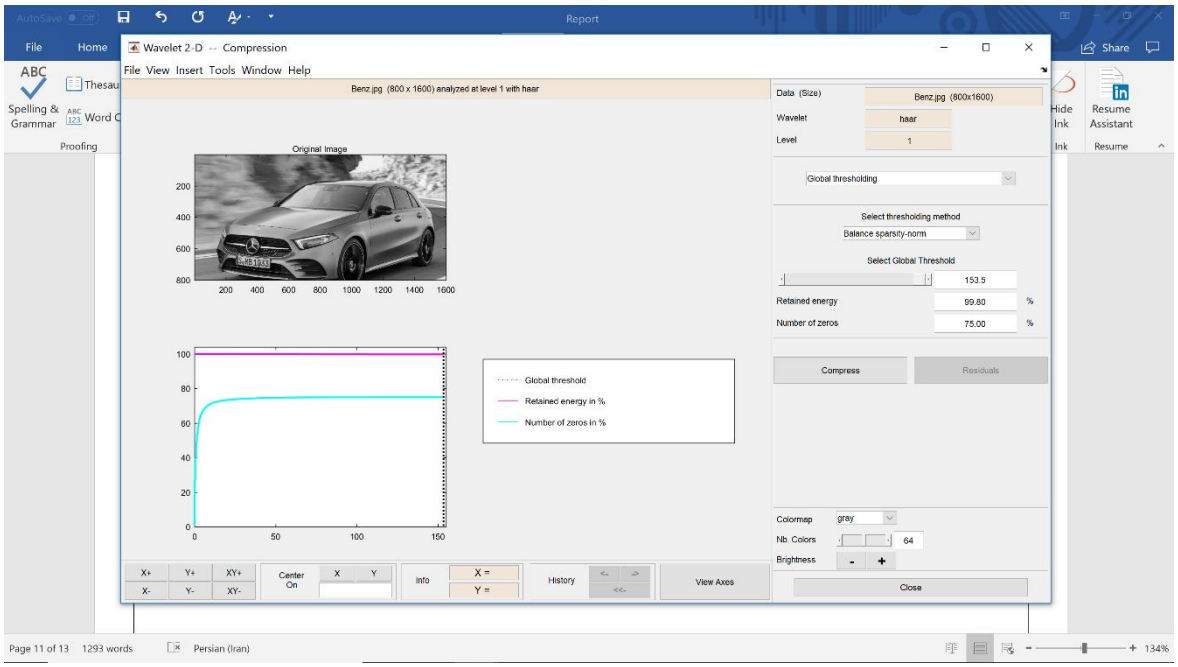
پس از آنالیز و انتخاب گزینه compress وارد محیط زیر میشویم.



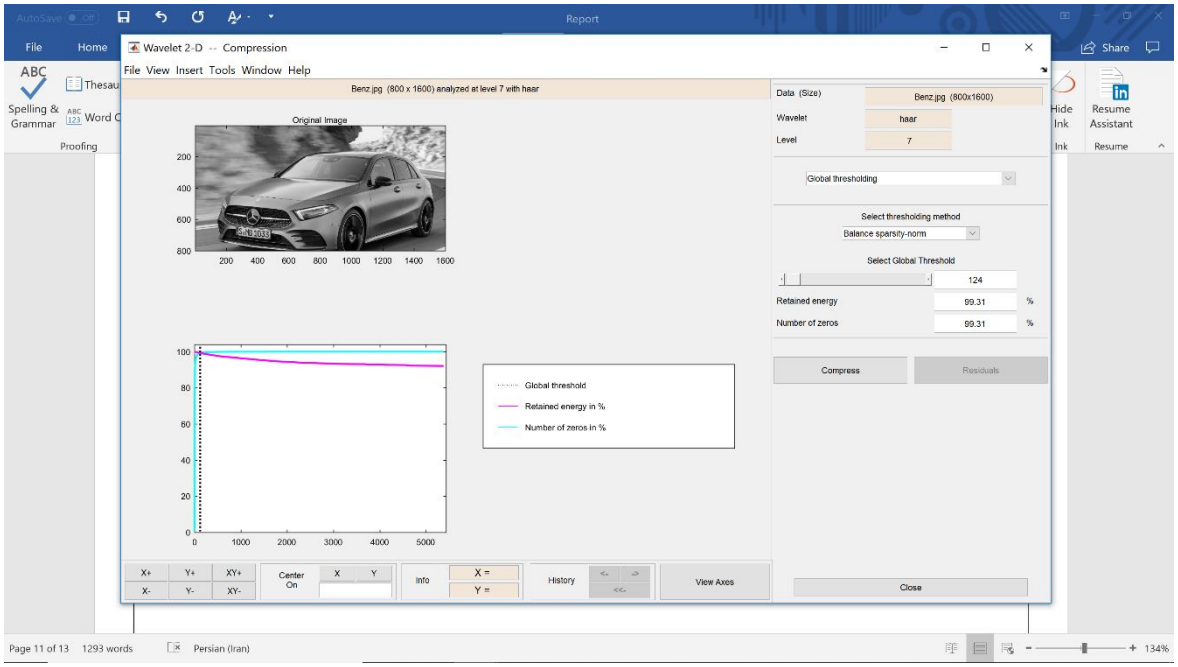
شکل ۹ محیط **compress** در **wavelet toolbox**

در نمودار ایجاد شده در پایین سمت راست به صورت خودکار سطوح آستانه را برای ارائه تعادل بین حفظ انرژی تصویر درحالیکه حداقل تعداد ضرایب مورد نیاز برای نشان دادن تصویر حفظ شود. این آستانه بصورت دستی هم تغییر میکند و میزان حفظ انرژی و تعداد ضرایب نشان داده میشود.

با انتخاب سطح بالاتر در هر موجک این آستانه در محل پایین‌تری انتخاب میشود.



شکل ۱۰ فشرده سازی تصویر در سطح ۱ و موجک **hear**



شکل ۱۱ فشرده سازی تصویر در سطح ۷ موجک **hear**

همینطور که در عکس‌ها مشاهده میکنید آستانه انتخابی در سطح ۷ در محل پایین‌تری (۱۲۴) انتخاب شده و ۹۹/۳۱ درصد انرژی تصویر حفظ شده است. اما در سطح ۱ آستانه در محل ۱۵۳ انتخاب شده و ۹۹/۸۰ درصد انرژی حفظ شده.

هرچه مقدار آستانه بیشتر باشد یعنی مقدار بیشتری صفر میتواند ساخته شود اما مقدار بیشتری از انرژی از دست میرود. مقدار انرژی حفظ شده نشان از این دارد که چه مقدار از جزئیات تصویر نگه داشته میشود و در واقع کیفیت تصویر را نشان میدهد. مقدار صفرهای ساخته شده نشان میدهد چقدر میشود عکس را فشرده کرد. هر چه صفرها بیشتر باشد میتوان عکس را بیشتر فشرده کرد.

پیوست : روند اجرای برنامه

برای اجرای برنامه درون پوشه با نام code برای هر سوال پوشه‌ای جدا با شماره همان سوال در نظر گرفته شده است. برای سوال اول در پوشه Q1 فایل Q1 را اجرا نمایید.نمودارهای تبدیل فوریه عکس ورودی، تبدیل فوریه عکس پس از رفع نویز سینوسی و عکس پس از رفع نویز سینوسی و حرکتی نشان داده میشوند.

برای سوال دوم در پوشه Q2 فایل Q2 را اجرا کنید و سپس quality of compression را وارد کنید. سپس عکس اولیه و خروجی برنامه نمایش داده میشود و در قسمت کامند ویندو مقدار خطا را میتوانید ببینید.

مراجع

مراجع استفاده شده برای انجام این پروژه عبارتند از

1. https://www.youtube.com/watch?v=aFbGqXFT0Nw
2. https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348