«بسم الله الرحمن الرحیم»

مخابرات دیجیتال

گزارش تمرین اول کامپیوتری

دانشجو: سیدسعید جزائری شوشتری

شماره دانشجویی: ۹۸۱۰۴۸۸۵

دانشکده مهندسی برق

پاییز ۱۴۰۲

۱) آشنایی با الگوریتم های فشرده سازی تصویر

۱. در اصل، الگوریتم های فشرده سازی تصویر به دو دسته Lossless و Lossy تقسیم میشوند. در فشرده سازی های Lossless هیچگونه اطلاعاتی از بین نمیرود اما به روش های مختلف و استفاده از ویژگی های تصویر، حجم ذخیره سازی آن کاهش میابد. این گونه الگوریتم ها معمولا در تصاویر پزشکی که جزئیات حائز اهمیت میباشند استفاده میشود.

در مقابل، در روش های فشرده سازی Lossy بهرحال بخشی از اطلاعات تصویر از بین میرود اما حجم ذخیره سازی بسیار کاهش میابد. این راه معمولا در فضا هایی که نرخ انتقال داده محدود میباشد و در مقابل جزئیات تصویر چندان حائز اهمیت نیست مانند تصاویر دیجیتال روزمره استفاده میشود.

روش های مختلف فشرده سازی Lossy به شرح زیر میباشند:

- Transform Coding : به طور کلی، بهینه ترین روش فشرده سازی تصویر میباشد و بر پایه تبدیل های مرتبط به فوریه میباشد. حالت خاص آن یعنی Discrete Cosine Transform در روش JPEG استفاده میشود.
- Color Quantization : به طور خلاصه، با استفاده از کاهش تعداد سطوح کوانتیزاسیون شدت رنگ های اصلی قرمز، سبز و آبی میباشد به این معنا که بجای ۲۵۶ سطح مختلف از سطوح کمتری استفاده میکند.
- Chroma Subsampling : با استفاده از این واقعیت که سیستم فیزیکی چشم انسان به برخی جزئیات حساسیت کمتری دارد به فشرده سازی میپردازد.
- روش های مبتنی بر هوش مصنوعی که اخیرا با رویکردی متفاوت به این مساله میپردازند.

در مقابل، روش های زیر معروف ترین ها در فشرده سازی Lossless هستند:

- Run-length Encoding
- Are Image Compression
- Entropy Encoding
- Diffusion Models
- ...

در مورد مزایای استاندارد فشرده سازی JPEG به موارد زیر میتوان اشاره نمود:

i. بسیار قابل فشرده سازی است. در واقع با حفظ مناسب کیفیت و جزئیات تصویر، بخش قابل توجهی از حجم فایل تصویری را میکاهد.

ii. تطابق پذیری بالایی دارد. در واقع این فرمت به سادگی قابلیت دیکودینگ دارد و سخت افزار های مختلف دیکودینگ و نرم افزار های ساده قابلیت استفاده و پشتیبانی از آن را دارا میباشند.

iii. نرم و پروضوح است. به این معنا که طیف رنگی بسیار زیاد و حالت کلی تصویر از قبیل شفافیت و رزولوشن و ترکیب رنگی را حفظ میکند.

iv. قابلیت شخصی سازی دارد. یعنی با این روش فشرده سازی بر اساس مورد نیاز میتوان تصویر را به میزان دلخواه کم حجم نمود.

۲. درمورد روش بیت‌مپ میتوان گفت یک روش ذخیره سازی بدون اتلاف اطلاعات مستقل از پلتفرم است، به نحوی که هر دستگاهی میتواند از آن استفاده کرده تا تصویر را نمایش دهد. ساختار یک فایل بیت‌مپ (بدون در نظر گرفتن اطلاعات اختیاری) به صورت زیر میباشد:

الف) هدر فایل بیت مپ که ۱۴ بایت اندازه دارد. این رشته ثابت در ابتدای هر فایل بیت مپ می آید تا نشان دهد استاندارد این فایل از نوع بیت مپ میباشد و یک سری اطلاعات کلی دیگر در اختیار نرم افزاری که قرار است از آن استفاده کند قرار میدهد.

ب) هدر DIB یا همان اطلاعات فایل بیت مپ که هرچند حالت های مختلفی دارد، اما در هر حالت این هدر نیز اندازه فیکس و ثابتی دارد و به دستگاه اطلاع میدهد که نحوه نمایش تصویر چگونه است. روش متداول و امروزی آن، ۱۲۴ بایت اندازه دارد.

ج) قسمت Pixel Array که قسمت اصلی یا همان اطلاعات تصویر است. در این قسمت اطلاعات پیکسل-به-پیکسل تصویر ذخیره میشود. معمولا اطلاعات با شروع از پیکسل پایین سمت چپ تصویر شروع شده و به اطلاعات پیکسل بالا سمت راست تصویر خاتمه میابد. اطلاعات بیت مپ میتواند به صورت ۱بیت به ازای هر پیکسل، ۲بیت به ازای هر پیکسل، ۴بیت به ازای هرپیکسل، ۸بیت به ازای هر پیکسل، ۲۴ بیت به ازای هر پیکسل و ۳۲ بیت به ازای هرپیکسل باشد که که متناظر با یک رنگ خواهند بود. واضح است که هرچه تعداد بیت متناظر با آن پیکسل بیشتر باشد اطلاعات دقیق تری از آن تصویر ثبت خواهد شد.

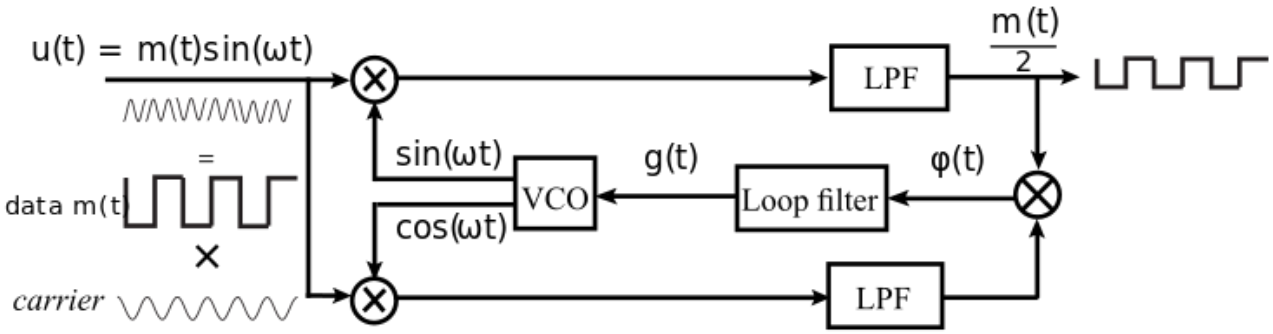
معمولا تصاویر بیت مپ ۲۴بیت به ازای هر پیکسل هستند به این معنا که ۸بیت یا ۲۵۶ درجه مختلف برای رنگ قرمز، ۸بیت یا ۲۵۶ درجه مختلف برای رنگ سبز و ۸بیت یا ۲۵۶ درجه مختلف برای رنگ آبی را ثبت میکنند تا با ترکیب مختلف آن ها ۱۶۷۷۷۲۱۶ رنگ مختلف را بسازند.

منبع: ویکی پدیا۱، ویکی پدیا۲ و اجوکیتیو.

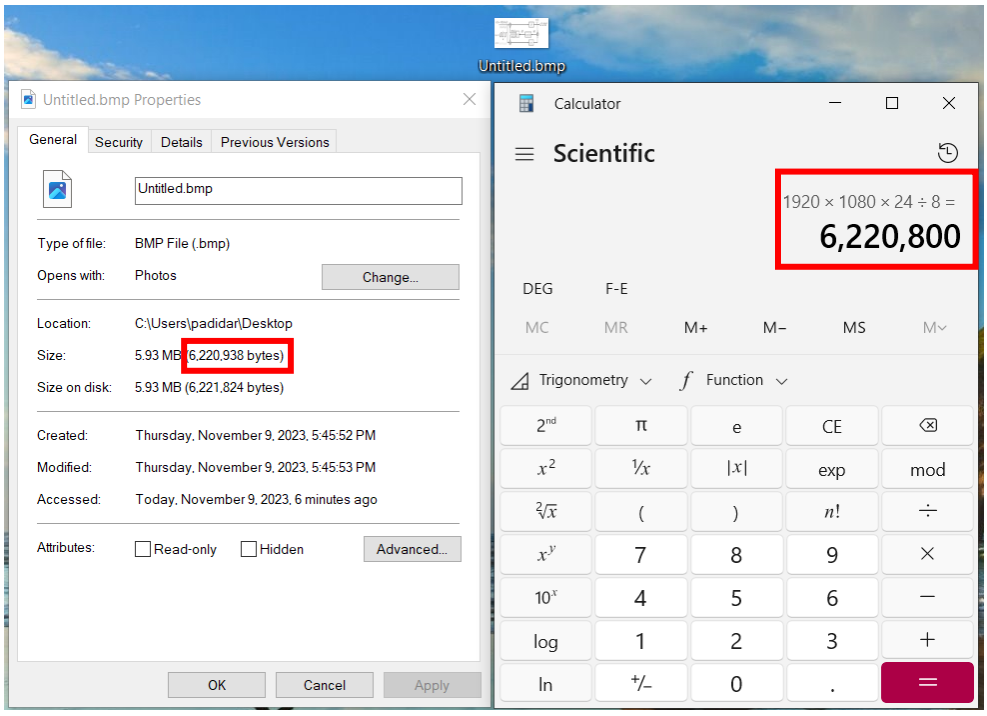
۲) تولید تصویر بیت‌مپ

در این بخش قرار است ساختار بلوکی Costas Loop را به صورت یک تصویر بیت مپ ذخیره کنیم. با توجه به دشواری ترسیم بلوک دیاگرام در نرم افزار GIMP و نا آشنایی با آن، تصویر بلوکی این روش را دانلود کرده و به کمک این نرم افزار آن را به فرمت bmp که افزونه بیت مپ است ذخیره میکنیم. دقت کنید مطابق با خواسته سوال، ما تصویر نهایی را در ابعاد $1920 \times 1080\ pxl$ ذخیره میکنیم. از طرفی هر پیکسل را با دقت ۲۴ بیت ذخیره میکنیم.

ساختار بلوکی که ترسیم کرده ایم به صورت زیر است:



تصویر اصلی (تصویر فوق) با افزونه png و ترنسپرنت بوده و سایز بسیار کمی در حدود چند کیلوبایت دارد. اما وقتی به تصویر ذخیره شده به فرمت بیت مپ دقت میکنیم میبینیم که همه چیز مطابق انتظار است:



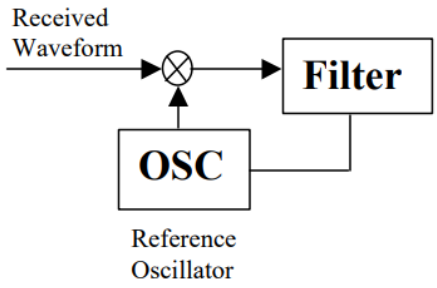
واضح است که حجم تصویر مطابق استاندارد ذخیره سازی ما و آنچه به عنوان اندازه هدر ها گفتیم کاملا مطابق انتظار است:

$$Size = \underbrace{1920 \times 1080 \times 24 \div 8}_{Image\ data\ size} + \underbrace{124}_{DIB\ Header} + \underbrace{14}_{BMP\ Header} = 6220938\ MB$$

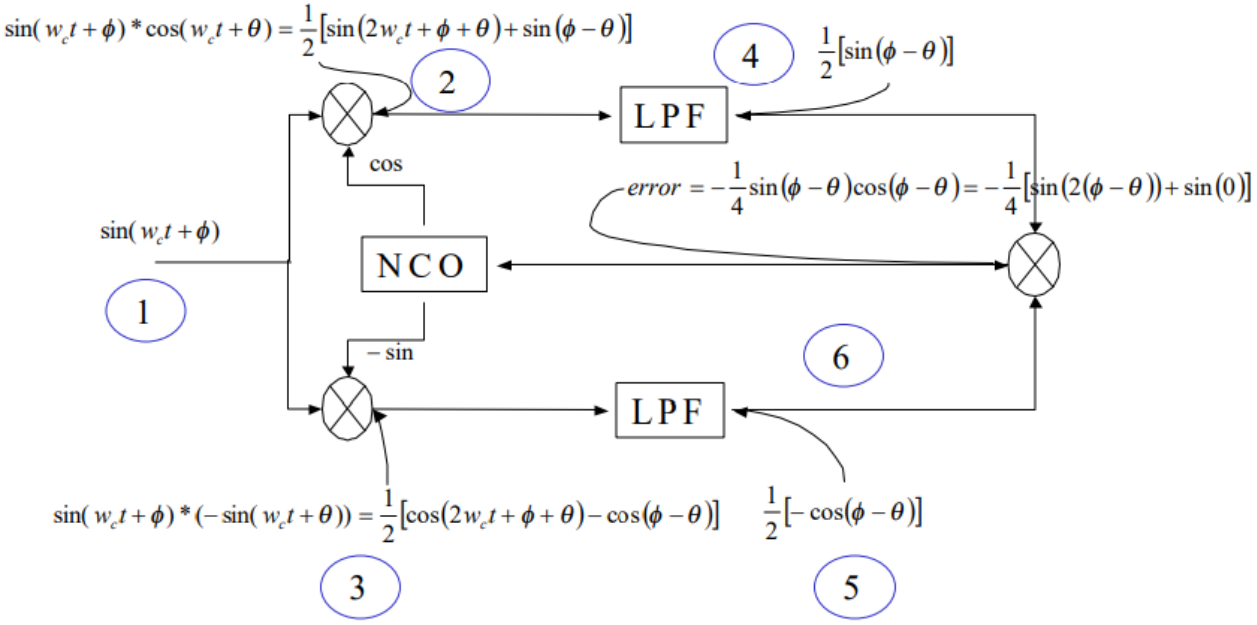
که ترم اول همان است که در ماشین حساب بدست آورده ایم و برابر با حجم دیتای خام پیکسل هاست.

برای پایان این بخش، توضیح مختصر و کلی از عملکرد Costas Loop میدهیم:

در واقع حلقه Costas یک حلقه قفل فاز برای بازیابی فرکانس حامل سیگنال های مدوله شده میباشد. بنابر این خوب است یک توضیح مختصری در رابطه با ساختار حلقه قفل فاز یا همان PLL بدهیم. در PLL یک سیگنال رفرنس با یک سیگنال دریافت میکس (ضرب) میشود تا یک سیگنال خطا حاصل گردد. با فیلتر مناسب سیگنال خطا، شکل موج مناسبی ایجاد میشود تا تولید کننده سیگنال رفرنس هم فاز و سنکرون با سیگنال دریافتی گردد این پروسه به نحوی ادامه دار میشود که سیگنال دریافتی و سیگنال رفرنس در فرکانس و فاز سنکرون میشوند.



حال با توجه به شکل زیر، ساختار حلقه Costas و بخش های مختلف آن را توضیح میدهیم:



1 در این قسمت یک شکل موج ورودی تک تن (سینوسی) با فرکانس و فاز ناشناخته وارد سیستم میشود. البته باید یک حد بالا برای فرکانس این سیگنال بدانیم.

2 در قسمت دو و سه، سیگنال ورودی در قسمت ۱، با دو سیگنال سینوسی و کسینوسی به صورت مستقل
3 میکس(ضرب) میشود. با این کار، شکل موج هایی که در خروجی این دو ضرب کننده در شکل بالا نوشته شده است تولید میشود. هر کدام از این خروجی ها از دو ترم تشکیل شده اند که یکی در فرکانس دو برابر ورودی است و دیگری مستقل از زمان و یک ترم ثابت است. نکته مهم در اینجا مطمئن شدن از نرخ نمونه برداری برای حذف اثر aliasing میباشد.

4 با گذردادن خروجی های قسمت های ۲ و ۳ از یک فیلتر پایین گذر، واضح است که صرفا ترم تفاضلی یعنی عبارت
5 ثابت مستقل از زمان باقی خواهد ماند.

6 در این قسمت عبارات بدست آمده از دو فیلتر پایین گذر قسمت های ۴ و ۵ در هم ضرب میشوند. در اینجا نیز یک ترم حاصل از جمع آرگومان ها داریم و یک ترم حاصل از تفریق آن ها. واضح است ترم حاصل از تفریق برابر صفر بوده و حاصل این قسمت یک عبارت سینوسی با دو برابر اختلاف فاز سیگنال ورودی و سیگنال مرجع است. در واقع مزیت عملکردی این سیستم نیز همین است که حساسیت حلقه به اختلاف فاز را تا دو برابر افزایش میدهد.

در پایان باتوجه به close loop بودن این سیستم، اختلاف فاز به مرور تصحیح شده و سیگنال رفرنس هم فاز با سیگنال ورودی تولید میگردد.

۳) فشرده سازی و تبدیل به فرمت JPEG

الف)

۱. مرحله Color Space Conversion

در این مرحله تصویر از فضای RGB به فضای Y-Cb-Cr تبدیل میشود. در این فضا Y میزان درخشندگی، Cb درجه آبی تصویر و Cr درجه قرمز تصویر را نشان میدهد. هدف از این بخش آن است که بر اساس میزان حساسیت چشم انسان به رنگ ها و مشخصه های مختلف بتوانیم بخشی از اطلاعات غیر قابل استفاده را حذف کنیم.

۲. مرحله Subsampling

در این مرحله، از اجزای کرومیک (یعنی Cb و Cr) نمونه برداری میشود تا باز هم حجم تصویر کاهش یابد. مثلا به صورت ۴:۲:۲ یعنی نصف داده های این دو کامپوننت را دور بینداز و ...

۳. مرحله Splitting

در این مرحله تصویر به بلوک های n*n تقسیم میشود و هر بلوک به عنوان یک پیکسل تلقی میشود

۴. مرحله Discrete Cosine Transform

این مرحله روی هر بلوک مرحله قبل اجرا میشود و هدف انتقال تصویر از فضای تفسیر فضایی به فضای فرکانسی است. به این نحو که با انجام این تبدیل تصویر به صورت جمع سینوسی ها با دامنه ها و فرکانس های مختلف تعبیر میشود و کمک میکند تا حجم تصویر باز هم کاهش یابد.

۵. مرحله Quantization

در این مرحله ضرایب مرحله قبل را کوانتایز میکنیم. علت هم از آنجا می آید که چشم ما به یک فرکانسی و بالاتر از آن کمتر حساس است و با کوانتایز کردن این ضرایب میتوان حجم راه کاهش داد.

۶. مرحله Entropy Coding یا Huffman Coding

کد کردن در این مرحله منجر میشود که ضرایب بدست آمده در مرحله قبل در رشته بیت با طول کمتری قرار گیرند. روش هم همانطور که میدانیم مقدار دهی رشته های بیتی با طول کمتر به مقادیر پرتکرار تر است.

۷. مرحله Zigzag Scanning

ضرایب نمونه برداری شده را به صورت زیگزاگی پشت سر هم میچیند و علت آن اینطور که بیان میشود استفاده از ویژگی های بینایی انسان است.

۸. مرحله Run-Length Encoding

این انکودینگ، منجر میشود تا رشته های بیتی ۰ طولانی، در تعداد بیت کمتری قابل نمایش باشند.

۹. مرحله JPEG File Format

در این مرحله دیتای فشرده شده را با هدر های مربوط به استاندارد JPEG ترکیب میکنیم تا تصویر به فرمت خواسته شده ذخیره گردد.

همانطور که واضح است فاکتور های زیر میتوانند بر کیفیت و حجم خروجی نهایی تاثیر بگذارند:

- **نسبت نمونه برداری در مرحله ۲:** با کاهش یا افزایش این نسبت نمونه برداری میتوانیم به صورت دلخواه بین حم تصویر خروجی و کیفیت تصویر تریداف ایجاد کنیم.
- **سایز تقسیم بندی در مرحله ۳:** مشخصا انتخاب n در انتخاب بلوک های n*n و تعبیر همه آن ها به صورت یک تک پیکسل میتواند اثرگذار باشد.
- **مدل کوانتیزاسیون در مرحله ۵:** بدیهی است کوانتیزاسیون های یونیفرم یا غیریونیفورم، تعداد سطح نمونه برداری و به صورت کلی حالات مختلف کوانتیزاسیون روی میزان از دست رفت اطلاعات اثر میگذارند.
- **کدینگ هافمن در مرحله ٦:** استفاده از جدول های مختلف هافمن کد مخصوص روش jpeg منجر به تغییر کیفیت و حجم خروجی نهایی میشود.

منابع: گیکس فور گیکس، جاوا تی پوینت

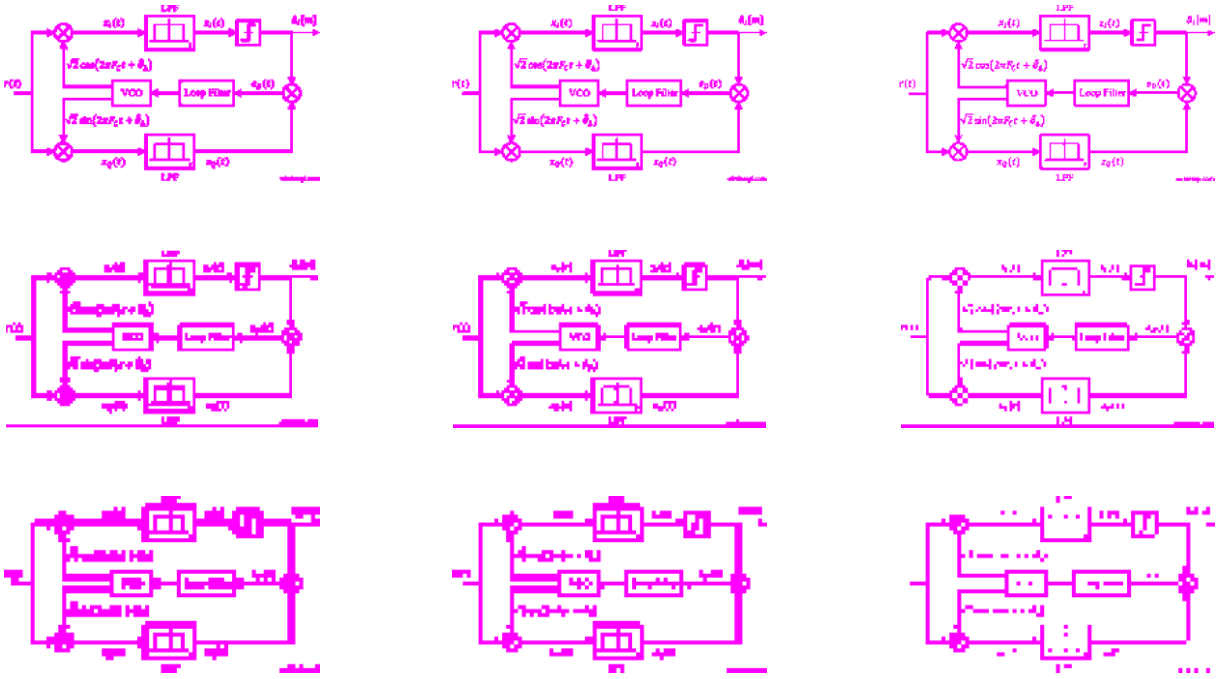
ب)

همانطور که در قسمت قبل توضیح داده شد، کانال سبز رنگ حذف شده و صرفا اثری از کانال های رنگی آبی و قرمز باقی میمانند. علت نیز در قسمت قبل بیان شد که به ویژگی های حساسیت چشم به رنگ های کرومیک برمیگردد. مابقی خواسته های این قسمت به صورت فایل های متلب قرار داده شده اند.

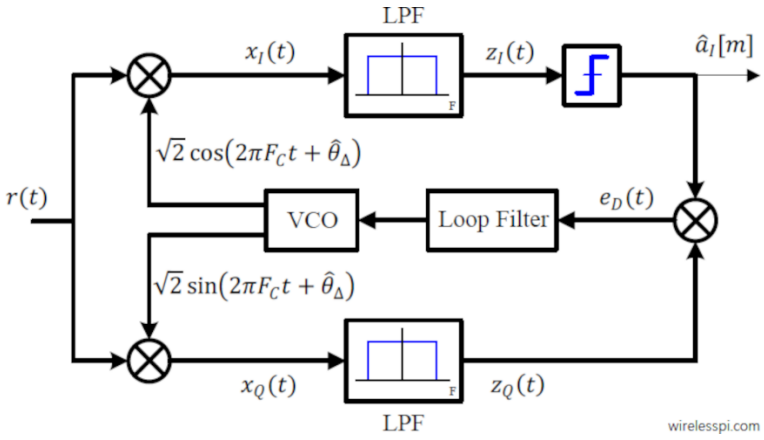
در این بخش سه پارامتر qualityFactor و SubsamplingRatio و blockSize به عنوان پارامتر های کنترلی قرار داده شده اند که البته چون جدول های کوانتیزاسیون در اینترنت فقط برای حالت ۸*۸ موجود بودند، پارامتر blockSize را همواره مجبور هستیم ۸ قرار دهیم.

نتایج به شرح زیر است:

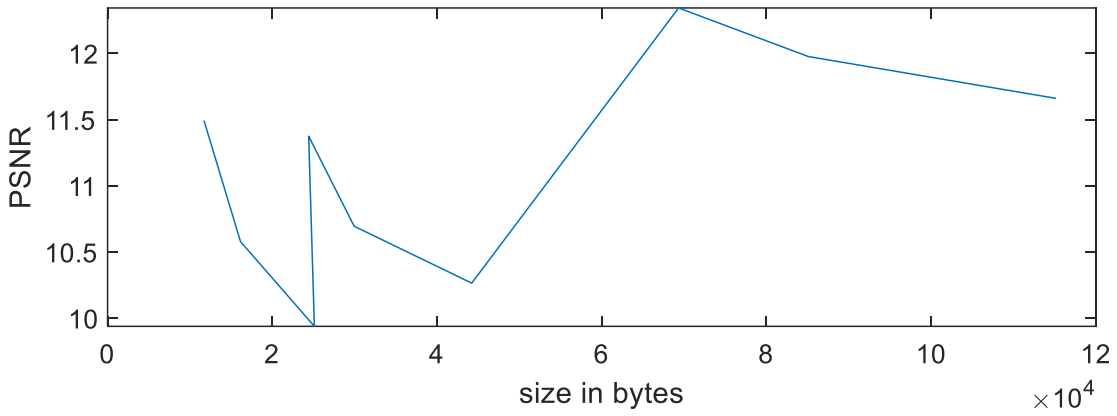
شکل زیر توسط متلب پلات شده و خروجی برای ۹ کیفیت مختلف (که پارامتر های کنترلی آن را در متلب میتوانید ببیند) بدست آمده است:

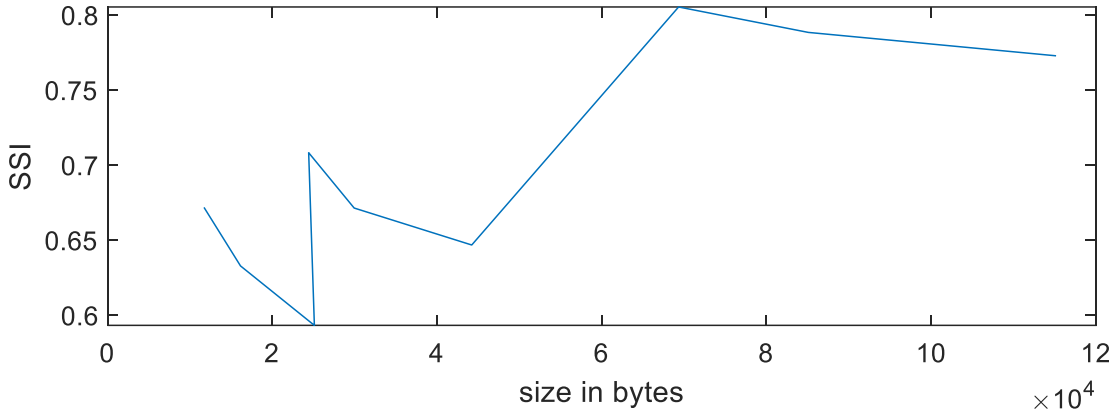


و تصویر اصلی:



نتایج معیار ها:


PSNR
12
11.5
11
10.5
10
0
2
4
6
8
10
12
size in bytes
×10^4


SSI
0.8
0.75
0.7
0.65
0.6
0
2
4
6
8
10
12
size in bytes
×10^4