باسمه تعالی

تکلیف سری دوم درس سیستم های چندرسانه ای

سارا برادران (شماره دانشجویی: ۹۶۲۴۱۹۳)

بلوک [1] فایل ipynb : کتابخانه ها

در این قسمت کتابخانه های به کار رفته در کد import شده است. به طور کلی از ۴ کتابخانه cv2، scipy، numpy و matplotlib استفاده نموده ایم که نصب هر یک از این کتابخانه ها به کمک دستورات زیر قابل انجام است. کتابخانه matplotlib برای نمایش تصاویر، کتابخانه cv2 برای اعمالی از جمله خواندن تصاویر، کتابخانه scipy برای استفاده از توابع dct و idct و کتابخانه numpy برای انجام برخی عملیات ها برروی تصاویر مورد استفاده قرار گرفته است که در ادامه به تفصیل به آن ها می پردازیم.

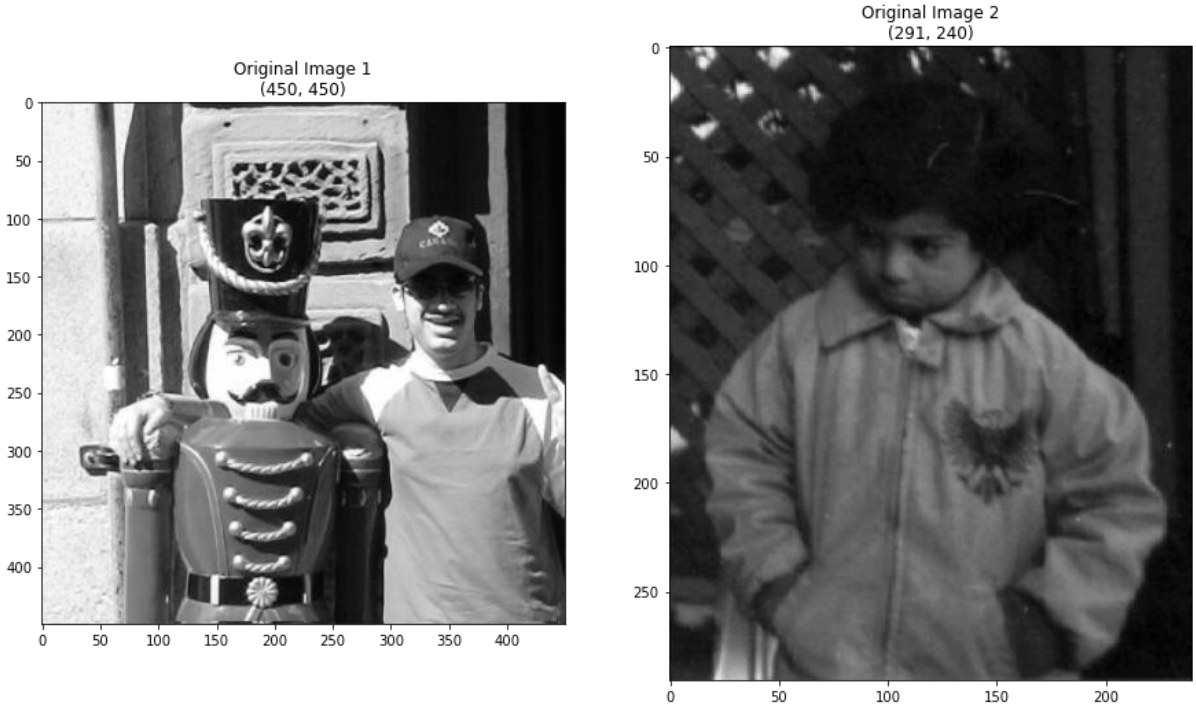
```
pip install numpy

pip install scipy

pip install matplotlib

pip install opencv-python
```

بلوک [2] فایل ipynb : تابع Show_Images()

در ابتدا یک تابع تحت عنوان show_images برای نمایش تصاویر به صورت تکی و چندتایی ایجاد شده است. برای نمایش تصویر و پیاده سازی این تابع از کتابخانه matplotlib و دستور imshow استفاده کرده‌ایم. همچنین این تابع به عنوان آرگومان ورودی لیستی از تصاویر، برچسب هر تصویر، و سایز مورد نیاز برای نمایش تصاویر را دریافت می نماید. به علاوه این تابع ابعاد تصاویر دریافتی را در کنار برچسب نام هر تصویر نمایش می دهد. از تابع پیاده سازی شده در مراحل بعدی و برای نمایش تصویر خروجی حاصل از توابع پیاده سازی شده استفاده می کنیم.

بلوک [3] فایل ipynb : خواندن تصاویر و نمایش آنها

در این قسمت ابتدا به وسیله تابع imread کتابخانه cv2 تصاویر 1.tif و 2.tif را خوانده و درون src_img1 و src_img2 ذخیره می نماییم سپس به وسیله تابع Show_Images نوشته شده در قسمت های پیشین، تصاویر را نمایش داده ایم. تصاویر اولیه مطابق شکل (۱) می باشند.



شکل (۱)

بلوک [4] فایل ipynb : تابع Hist_Median()

در این قسمت تابعی تحت عنوان Hist_Median پیاده سازی شده است که تصویری را به عنوان ورودی دریافت کرده، ابتدا آرایه pdf تصویر را محاسبه کرده و نمودار هیستوگرام مربوطه را نمایش می دهد، سپس از روی pdf بدست آمده آرایه cdf تصویر را محاسبه نموده و برای به دست آوردن ۳ میانه مورد نظر بر روی آرایه cdf حرکت می کند. در ابتدا پیکسل با سطح روشنایی T0 را به گونه ای جستجو می کند که مجموع تعداد پیکسل های با سطح روشنایی ۰ تا T0 از ۲۵ درصد کل پیکسل ها بیشتر باشد (لازم به ذکر است اولین پیکسل با ویژگی مربوطه به عنوان میانه مد نظر است برای مثال اگر سطح روشنایی ۲۳ به عنوان میانه نخست انتخاب گردد بدین معناست که مجموع تعداد پیکسل های با سطح روشنایی ۰ تا ۲۳ از ۲۵ درصد کل پیکسل های تصویر بیشتر یا مساوی بوده اما مجموع تعداد پیکسل های با سطح روشنایی ۰ تا ۲۲ کمتر از ۲۵ درصد کل پیکسل های تصویر است.) به همین ترتیب میانه دوم یا T1 به گونه ای انتخاب می گردد که مجموع تعداد پیکسل های با سطح روشنایی ۰ تا T1 بزرگتر یا مساوی ۵۰ درصد از کل پیکسل ها باشد و نهایتا میانه سوم یا T2 به گونه ای انتخاب می شود که مجموع تعداد پیکسل های با سطح روشنایی ۰ تا T2 بزرگتر یا مساوی ۷۵ درصد از کل پیکسل ها باشد. به وسیله متد axvline سه میانه یافت شده بر روی هیستوگرام تصویر نمایش داده خواهند شد. و آرایه میانه های تصویر به عنوان خروجی تابع بازگردانده می شود. این تابع در حقیقت تصویر ورودی را با ۴ سطح روشنایی کوانتیزه کرده و به این طریق تصویر ۸ بیتی اولیه را در قالب یک تصویر ۲ بیتی به عنوان خروجی تبدیل می کند.

بلوک [5] فایل ipynb : Median_Quantization()

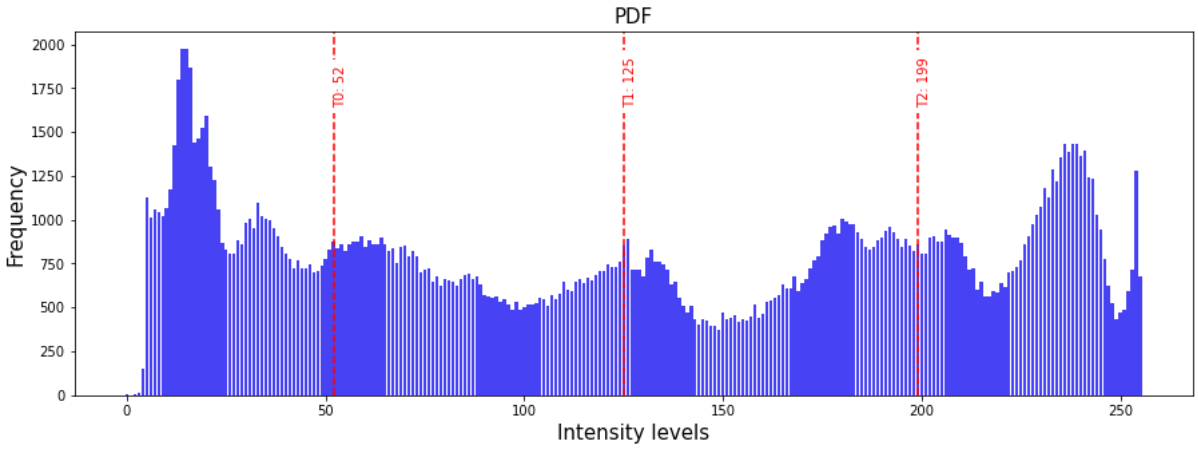
در این قسمت تابعی تحت عنوان Median_Quantization پیاده سازی شده است که یک تصویر و آرایه حاوی میانه های جستجو شده از پیکسل های تصویر را به عنوان ورودی دریافت کرده، سپس بر روی تصویر ورودی حرکت کرده و هر یک از پیکسل های تصویر که سطح روشنایی آن ها از میانه نخست کمتر باشد را با سطح روشنایی T(start)/2 تعویض کرده و چنانچه سطح روشنایی پیکسلی از میانه آخر بزرگتر باشد آن را با سطح 255+T(end)/2 تعویض نموده و در غیر اینصورت چنانچه سطح روشنایی پیکسلی بین دو میانه باشد آنگاه مقدار جدید آن را برابر میانگین دو میانه مربوطه تنظیم کرده و نهایتاً پیکسل های جدید توسط تابع np.round گرد شده و نوع داده آن ها به uint8 تغییر داده می شود تا تصویر نهایی توسط imshow قابل نمایش باشد.

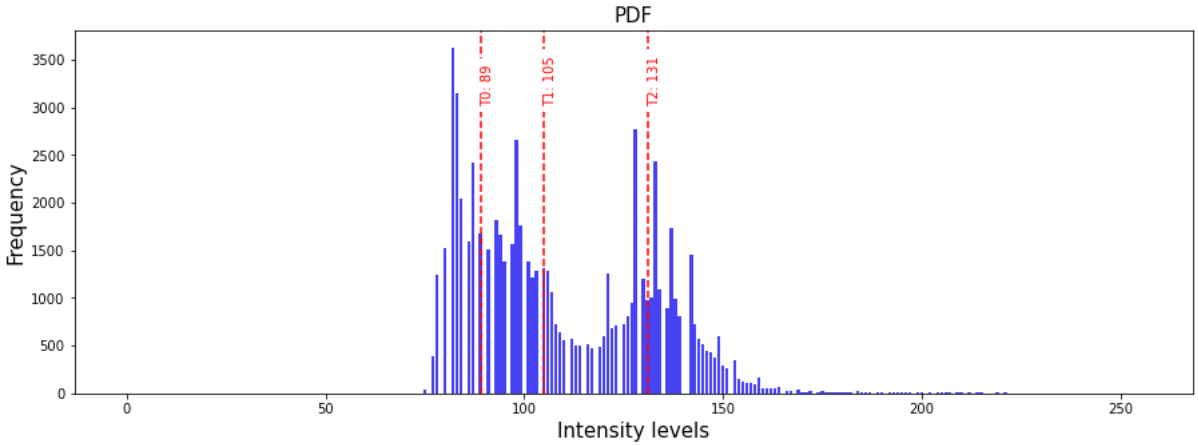
بلوک [6] فایل ipynb : تابع Simple_Quantization()

در این قسمت تابعی تحت عنوان Simple_Quantization پیاده سازی شده است که تصویری را به عنوان ورودی دریافت کرده، از روی آن تصویر جدیدی ایجاد می نماید به این صورت که ۶ بیت کم ارزش هر یک از پیکسل های تصویر اصلی را حذف کرده و تنها ۲ بیت پر ارزش را حفظ می کند و به این ترتیب رنج ۲۵۶ تایی سطوح روشنایی را به یک رنج ۴ تایی نگاشت می کند همچنین نیاز است ضمن تقسیم پیکسل ها به عدد ۶۴، حاصل به سمت پایین گرد شده و سپس در ۶۴ ضرب شود تا ۶ بیت کم ارزش به طور کامل از بین برود. . در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود و نوع داده آن ها به uint8 تغییر داده می شود تا تصویر نهایی توسط imshow قابل نمایش باشد.

بلوک [7] فایل ipynb : فراخوانی تابع Hist_Median() و نمایش هیستوگرام های خروجی

در این قسمت تابع Hist_Median فراخوانی شده و src_img1 و src_img2 به عنوان تصویر اولیه به این تابع پاس داده می شود سپس تصویر هیستوگرام حاصل از خروجی تابع نمایش داده شده و آرایه میانه های تصاویر به ترتیب در T1 و T2 ذخیره شده است هیستوگرام های بدست آمده مطابق شکل (۲) می باشند.

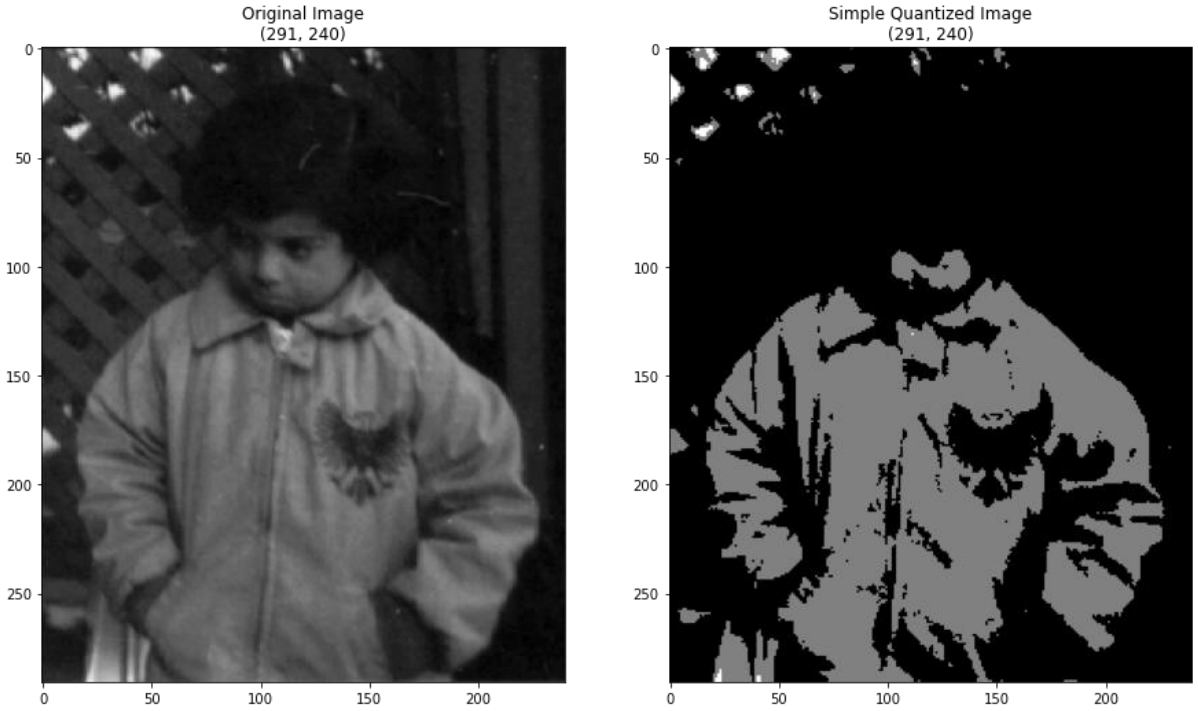


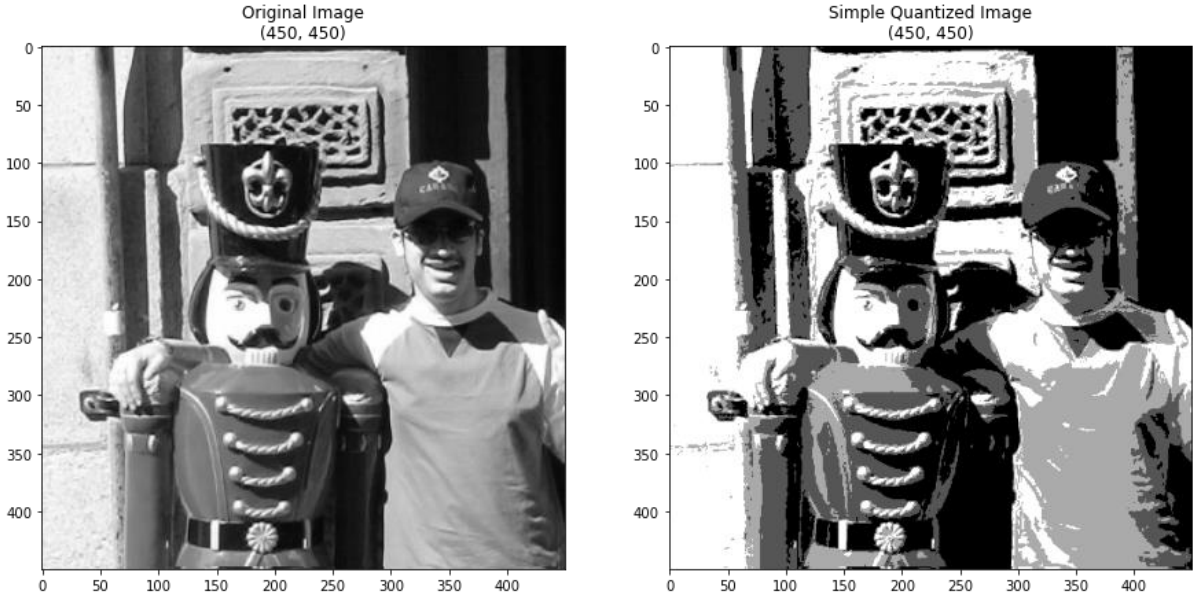


شکل (۲)

بلوک [8] فایل ipynb : فراخوانی تابع Simple_Quantization() و نمایش تصویر خروجی

در این قسمت تابع Simple_Quantization فراخوانی شده و تصاویر src_img1 و src_img2 به عنوان ورودی به آن پاس داده شده است. تصاویری که به صورت ساده کوانتیزه شده اند به عنوان خروجی تابع بازگردانده شده و مطابق شکل (۳) می باشد.

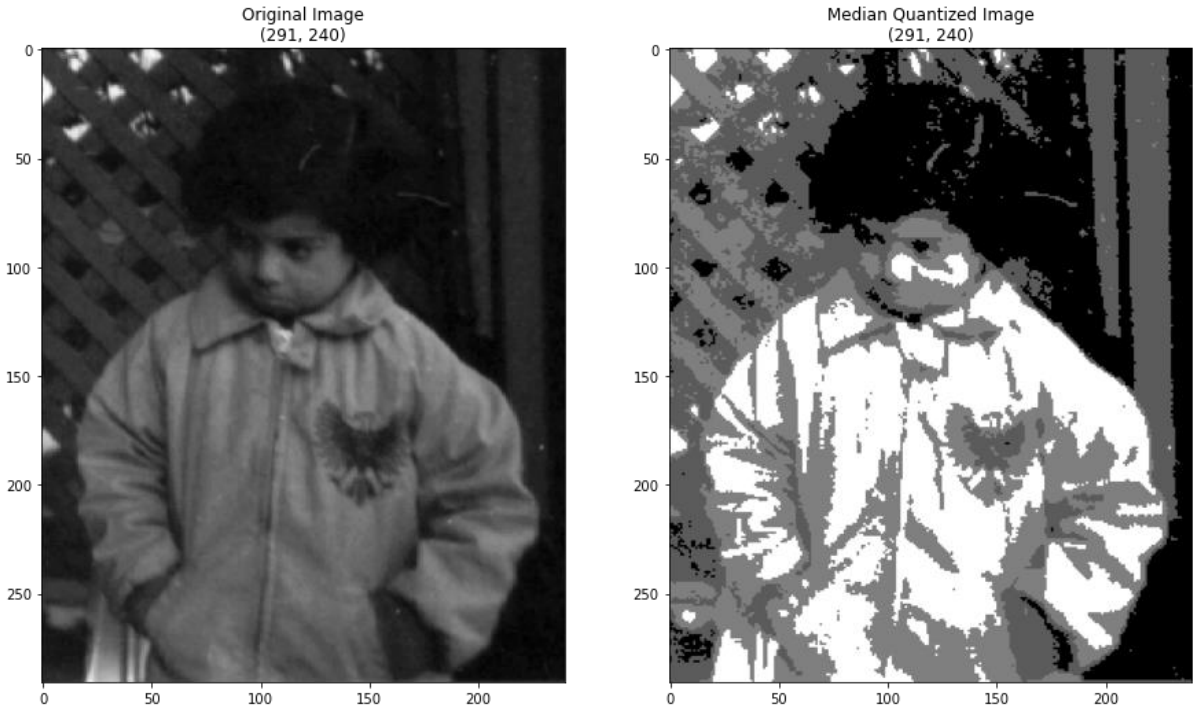


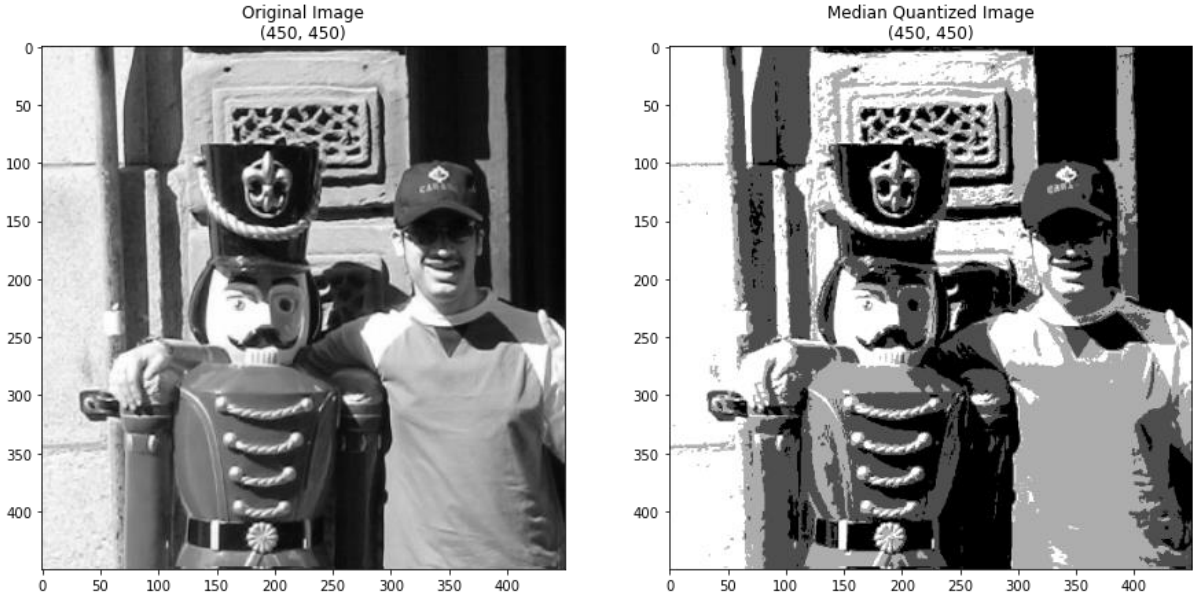


شکل (۳)

بلوک [9] فایل ipynb : فراخوانی تابع ()Median_Quantization و نمایش تصویر خروجی

در این قسمت تابع Median_Quantization فراخوانی شده و تصاویر src_img1 و src_img2 و آرایه میانه های T1 و T2 به ترتیب به عنوان ورودی به آن پاس داده شده است. تصاویری که به صورت میانه یابی کوانتیزه شده اند به عنوان خروجی تابع بازگردانده شده و مطابق شکل (۴) می باشد.





شکل (۴)

مقایسه دو روش کوانتیزه سازی:

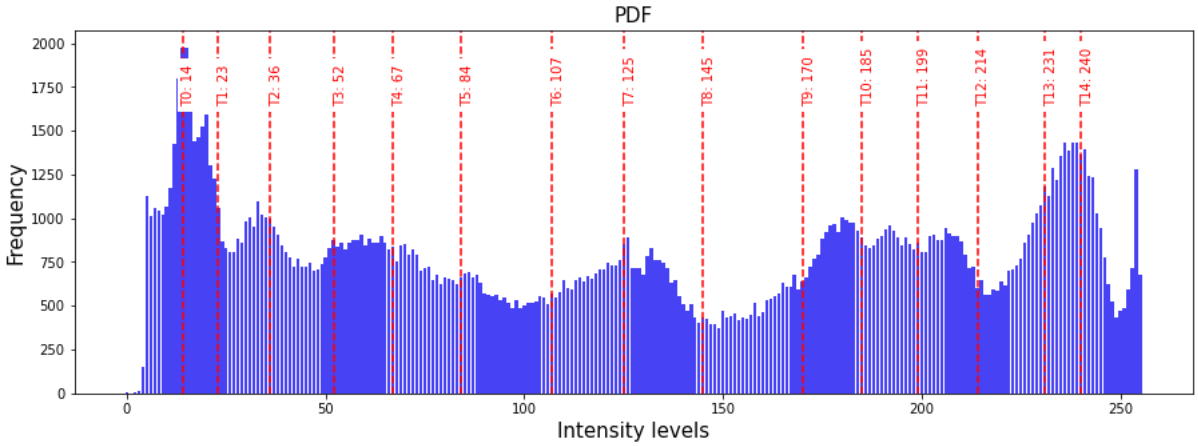
واضح است که برای تصویر src_img2 خروجی روش کوانتیزه سازی median_cut از نظر بصری بهتر از کوانتیزه سازی ساده عمل کرده است. و برای تصویر src_img1 از لحاظ بصری خروجی دو تصویر تفاوت چندانی ندارد.

بلوک [10] فایل ipynb : تابع Hist_Median()

در این قسمت تابعی تحت عنوان Hist_Median پیاده سازی شده است که مشابه تابع پیاده سازی شده در قسمت قبل می باشد با این تفاوت که عدد n را نیز به عنوان ورودی دریافت کرده و تعداد $2^n - 1$ میانه را در هیستوگرام تصویر جستجو می کند. برای پیاده سازی آن نیز کافی است هر بار میانه m ام (T_m) متناظر با یک سطح روشنایی به گونه ای جستجو شود که تعداد پیکسل های با سطح روشنایی از ۰ تا T_m بزرگتر و یا مساوی $100 * m/2^n$ درصد از کل پیکسل های تصویر باشد.

بلوک [11] فایل ipynb : فراخوانی تابع Hist_Median() و نمایش هیستوگرام خروجی

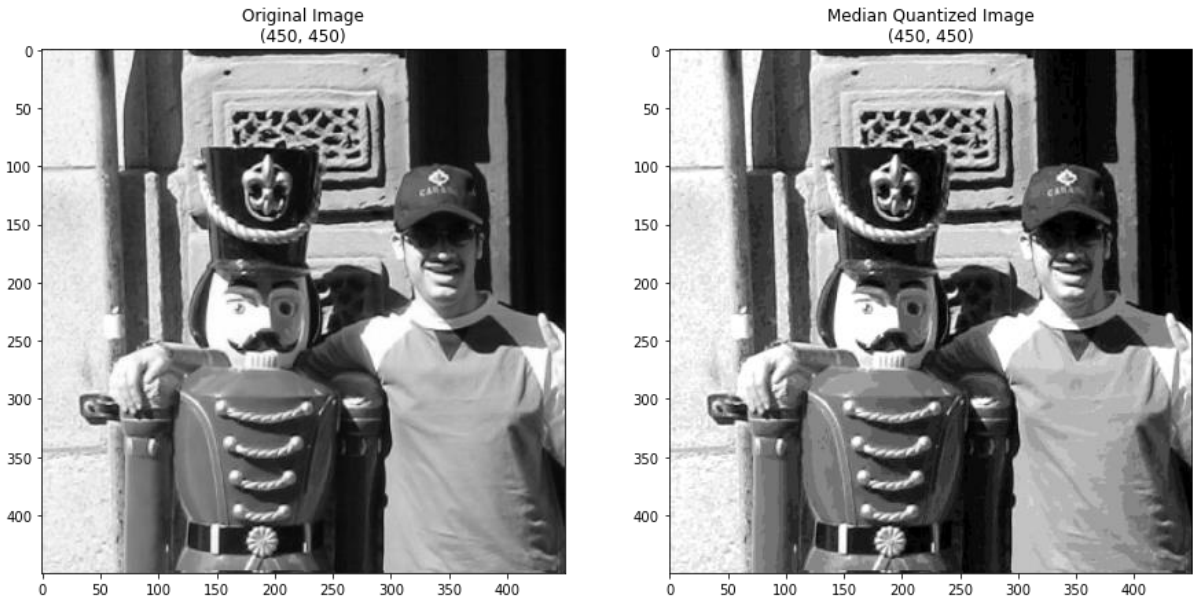
در این قسمت تابع Hist_Median فراخوانی شده و src_img1 به عنوان تصویر اولیه و آرگومان n با مقدار ۴ به این تابع پاس داده می شود سپس هیستوگرام تصویر به همراه میانه های بدست آمده نمایش داده شده است. هیستوگرام مربوطه مطابق با شکل (۵) می باشد.



شکل (۵)

بلوک [12] فایل ipynb : فراخوانی تابع Median_Quantization() و نمایش تصویر خروجی

در این قسمت تابع Median_Quantization فراخوانی شده و تصویر src_img1 و آرایه میانه های T1 به عنوان ورودی به آن پاس داده شده است. خروجی تصویری که به صورت میانه یابی کوانتیزه شده و ۴ بیتی است (۱۶ سطح روشنایی دارد) به عنوان خروجی تابع بازگردانده شده و مطابق شکل (۶) می باشد. واضح است هرچه تعداد بیت تصویر بیشتر باشد و یا به عبارت دیگر پارامتر n در تابع Hist_Median بزرگتر بوده و در نتیجه T ورودی Median_Quatization حاوی تعداد بیشتری میانه باشد تصویر خروجی به تصویر اصلی نزدیک تر خواهد بود. به همین دلیل است که تصویر کوانتیزه شده ۴ بیتی در شکل (۶) نسبت به تصویر کوانتیزه شده ۲ بیتی در شکل (۴) نزدیکتر به تصویر اصلی می باشد.



شکل (۶)

بلوک [13] فایل ipynb : تابع PSNR() ، dct2() و idct2()

در این قسمت تابعی تحت عنوان PSNR برای بدست آوردن peak signal-to-ratio پیاده سازی شده است. برای پیاده سازی این تابع کافی است ابتدا mse میان دو تصویر اصلی و بازسازی شده را محاسبه کرده و در گام بعد از فرمول زیر برای محاسبه PSNR استفاده نماییم. با توجه به فرمول نوشته شده مشخص است که هرچه مقدار PSNR بزرگتر باشد به این معناست که mse میان دو تصویر کمتر بوده و لذا تصویر بازسازی شده به تصویر اصلی نزدیک تر است.

$$
\begin{aligned}
PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\
&= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\
&= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)
\end{aligned}
$$

```
mse = (np.square(imageA.astype(int) - imageB.astype(int))).mean()

return 100 if mse == 0 else 10 * np.log10((255 * 255) / mse)
```

برای پیاده سازی توابع dct2 و idct2 به نحوی که عملکردی مشابه توابع متلب داشته باشد کافی است بر روی تصویر ورودی دو مرتبه dct نوع دوم و همچنین دو مرتبه idct نوع دوم گرفته شود. به صورت پیش فرض این توابع از type = 2 استفاده می نمایند و پارامتر norm می تواند دو مقدار None و ortho اخذ کند که فرمول تبدیلات مربوط به هر norm متفاوت می باشد و ما در پیاده سازی توابع ذکر شده از ortho normalization mode استفاده نموده ایم که فرمولی مشابه زیر دارد :

$$
f = \begin{cases} \sqrt{\frac{1}{4N}} & \text{if } k = 0, \\ \sqrt{\frac{1}{2N}} & \text{otherwise} \end{cases}
$$

برای اطلاعات بیشتر پیرامون توابع dct و idct پیاده سازی شده در پایتون از داکیومنت های زیر می توان استفاده نمود:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.dct.html

https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.idct.html

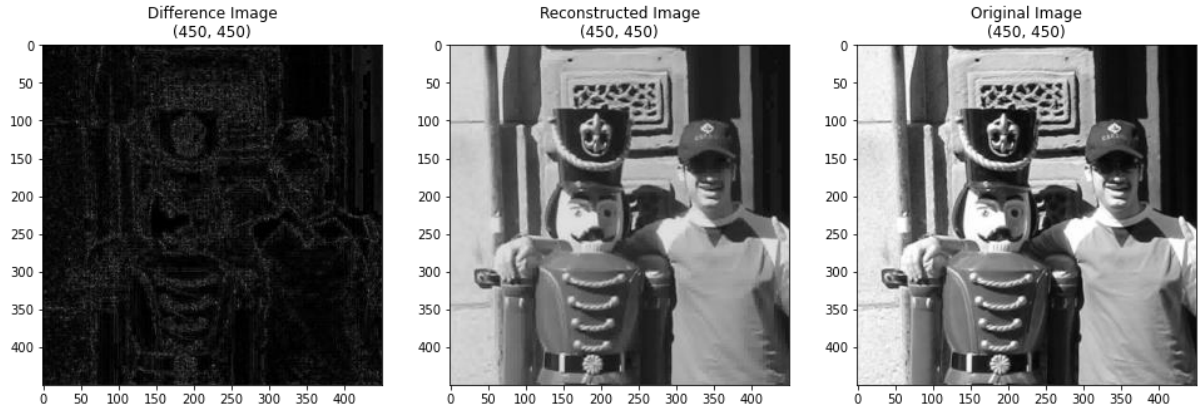
https://stackoverflow.com/questions/40104377/issiue-with-implementation-of-2d-discrete-cosine-transform-in-python

بلوک [14] فایل ipynb : تابع HW3_DCT()

در این قسمت تابعی تحت عنوان HW3_DCT پیاده سازی شده است که یک تصویر و پارامتر های k و t را به عنوان ورودی دریافت کرده و ابتدا بررسی می نماید که طول و عرض تصویر بر k بخش پذیر باشد. در غیر اینصورت تصویر را به گونه ای resize می کند که هم طول و هم عرض آن مضربی از k باشد. در گام بعد تصویر دریافتی را به بلوک های k تایی تقسیم کرده و سپس هر بلوک از پیکسل ها را به تابع dct2 پاس داده و در آرایه دریافتی حاصل تمام پیکسل هایی که قدرمطلق مقدار آن ها از t کمتر باشد را صفر کرده و تعداد این پیکسل ها را در شمارنده counter ذخیره می نماید سپس آرایه آستانه گذاری شده را به تابع idct2 پاس داده و حاصل را در تصویر جایگذاری می کند. به همین ترتیب در ادامه تصویر حاصل از تفاضل تصویر اولیه و حاصل idct2 را بدست آورده و درصد پیکسل های صفر شده را نیز محاسبه می نماید. برای dct2 و idct2 از توابع نوشته شده در قسمت پیشین استفاده شده است.

بلوک [15] فایل ipynb : فراخوانی تابع HW3_DCT() و نمایش تصاویر خروجی و PSNR

در این قسمت تابع HW3_DCT فراخوانی شده و src_img1 به عنوان تصویر اولیه و آرگومان t با مقدار ۵۰ و آرگومان k با مقدار ۵ به این تابع پاس داده می شود سپس تصویر اصلی، تصویر بازسازی شده و تصویر تفاصل نمایش داده شده است که مطابق با شکل (۷) می باشد.

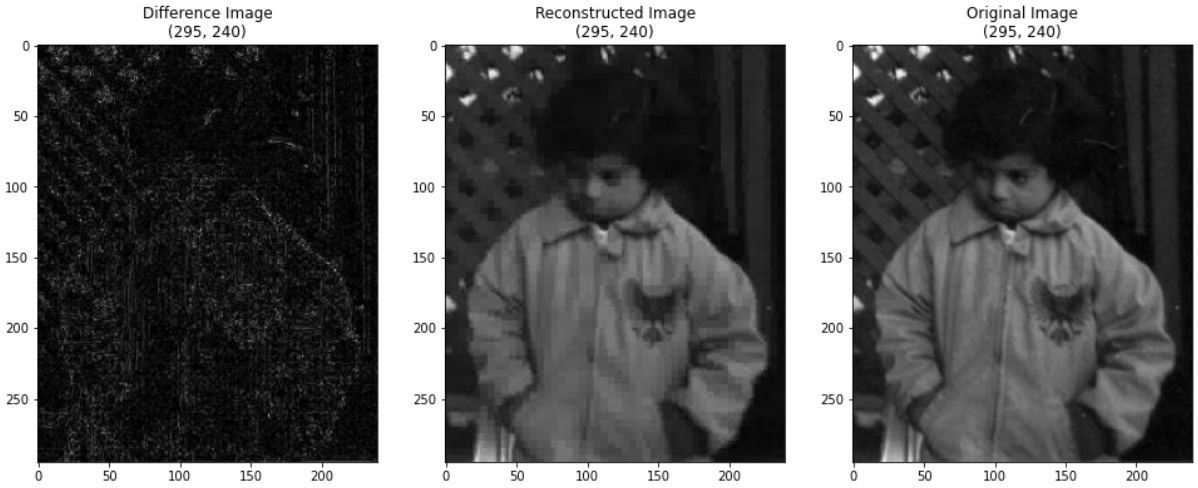



```
Zero DCTs = 93.39 %
My PSNR = 29.56
OpenCV PSNR = 29.56
```

شکل (۷)

بلوک [16] فایل ipynb : فراخوانی تابع HW3_DCT() و نمایش تصاویر خروجی و PSNR

در این قسمت تابع HW3_DCT فراخوانی شده و src_img2 به عنوان تصویر اولیه و آرگومان t با مقدار ۲۰ و آرگومان k با مقدار ۵ به این تابع پاس داده می شود سپس تصویر اصلی، تصویر بازسازی شده و تصویر تفاصل نمایش داده شده است که مطابق با شکل (۸) می باشد.




```
Zero DCTs = 95.59 %
My PSNR = 37.89
OpenCV PSNR = 37.89
```

شکل (۸)

با توجه به نتایج بدست آمده مشخص است که با وجود آنکه در تصویر اول حدود ۹۳ درصد و در تصویر دوم حدود ۹۵ درصد از تمامی dct ها صفر شده اند اما تصاویر بازسازی شده از کیفیت قابل قبولی برخوردار است. به علاوه نتیجه ی تابع PSNR پیاده سازی شده و نتیجه تابع PSNR آماده در کتبخانه cv2 هردو محاسبه و نمایش داده شده است.