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 ۳ کتابخانه cv2، 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 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.

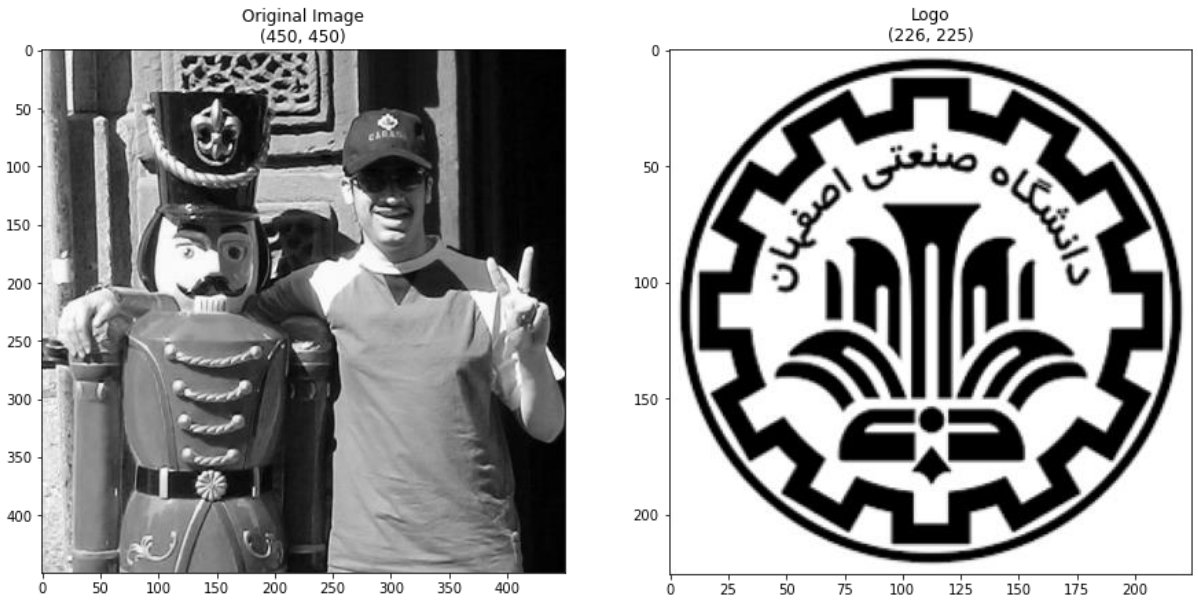
```
pip install numpy

pip install 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 تصویر اصلی و لوگو و نمایش آنها

در این قسمت ابتدا به وسیله تابع imread کتابخانه cv2 تصاویر Hi.tif و iut.tif را خوانده و درون src_img و logo 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.



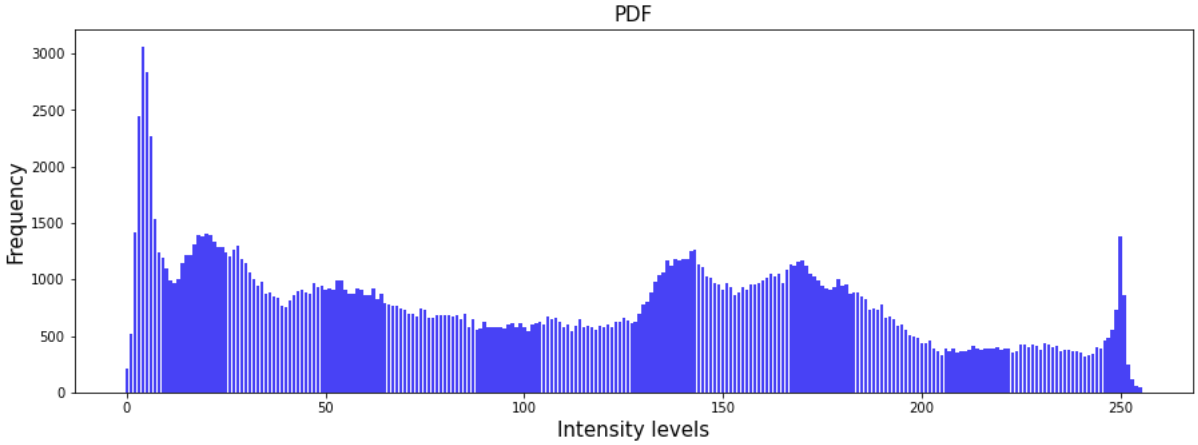
شکل (۱)

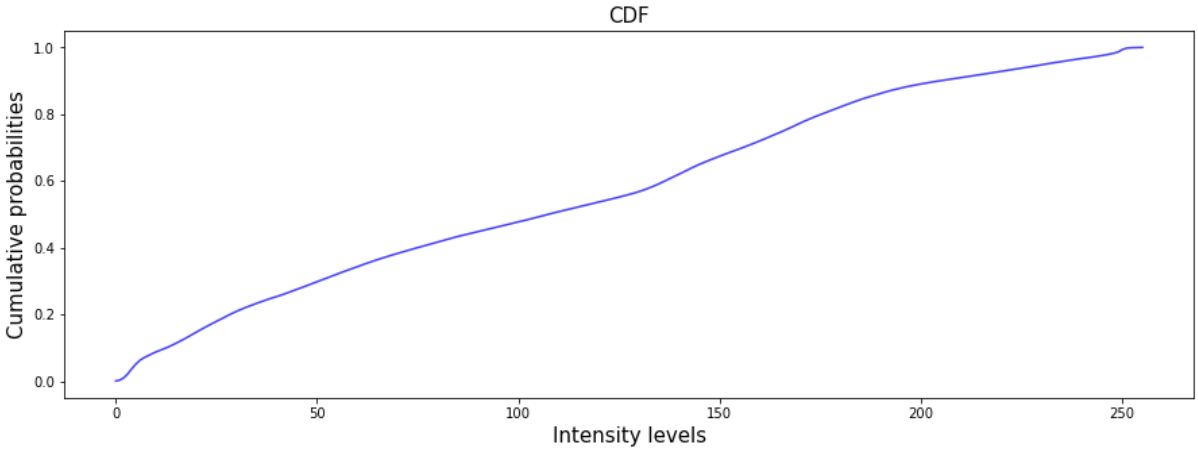
بلوک [4] فایل ipynb : تابع Hist_CDF()

در این قسمت تابعی تحت عنوان Hist_CDF پیاده سازی شده است که تصویری را به عنوان ورودی دریافت کرده، ابتدا آرایه pdf تصویر را محاسبه کرده و نمودار original histogram یا همان pdf را نمایش می دهد. سپس از روی pdf بدست آمده ارایه cdf تصویر را محاسبه کرده و نمودار cdf را نیز نمایش می دهد.

بلوک [5] فایل ipynb :فراخوانی تابع Hist_CDF() و نمایش تصویر خروجی

در این قسمت تابع Hist_CDF فراخوانی شده و src_img به عنوان تصویر اولیه به این تابع پاس داده می شود سپس نمودارهای حاصل از خروجی تابع نمایش داده شده است. نمودار های pdf و cdf تصویر اصلی Hi.tif مطابق شکل (۲) می باشد.





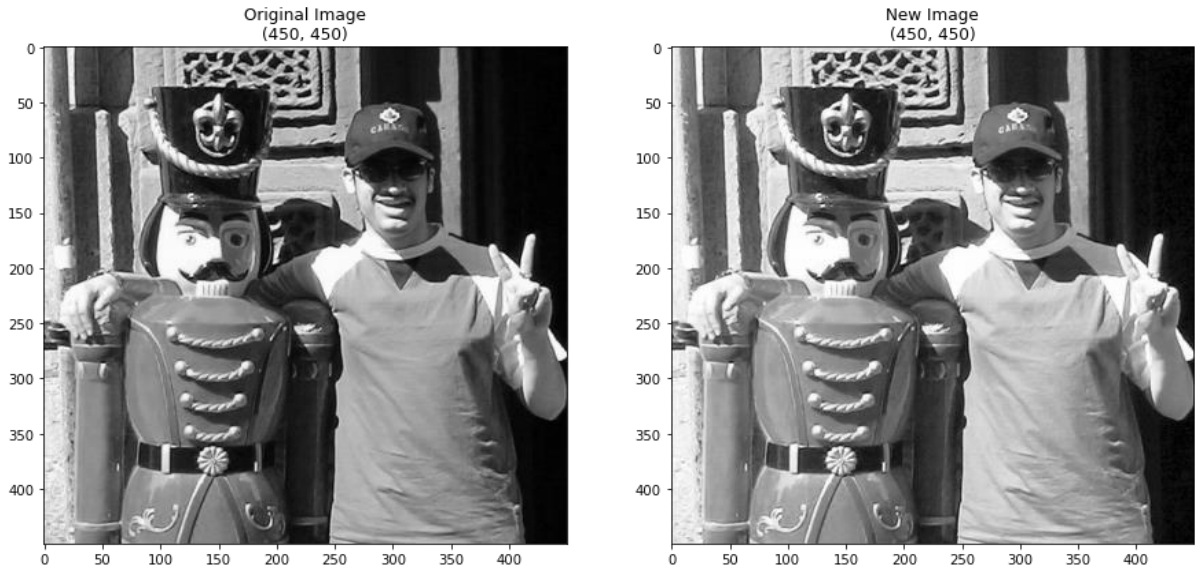
شکل (۲)

بلوک [6] فایل ipynb : تابع HW2_Histeq()

در این قسمت تابعی تحت عنوان HW2_Histeq پیاده سازی شده است که تصویری را به عنوان ورودی دریافت کرده، ابتدا آرایه pdf تصویر را محاسبه کرده و سپس از روی pdf بدست آمده آرایه cdf تصویر را محاسبه کرده و در انتها مقدار هر پیکسل در تصویر اصلی را در cdf نظیر آن پیکسل ضرب کرده و مقدار جدید پیکسل ها را محاسبه می کند. 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 [7] فایل ipynb : فراخوانی تابع HW2_Histeq() و نمایش تصویر خروجی

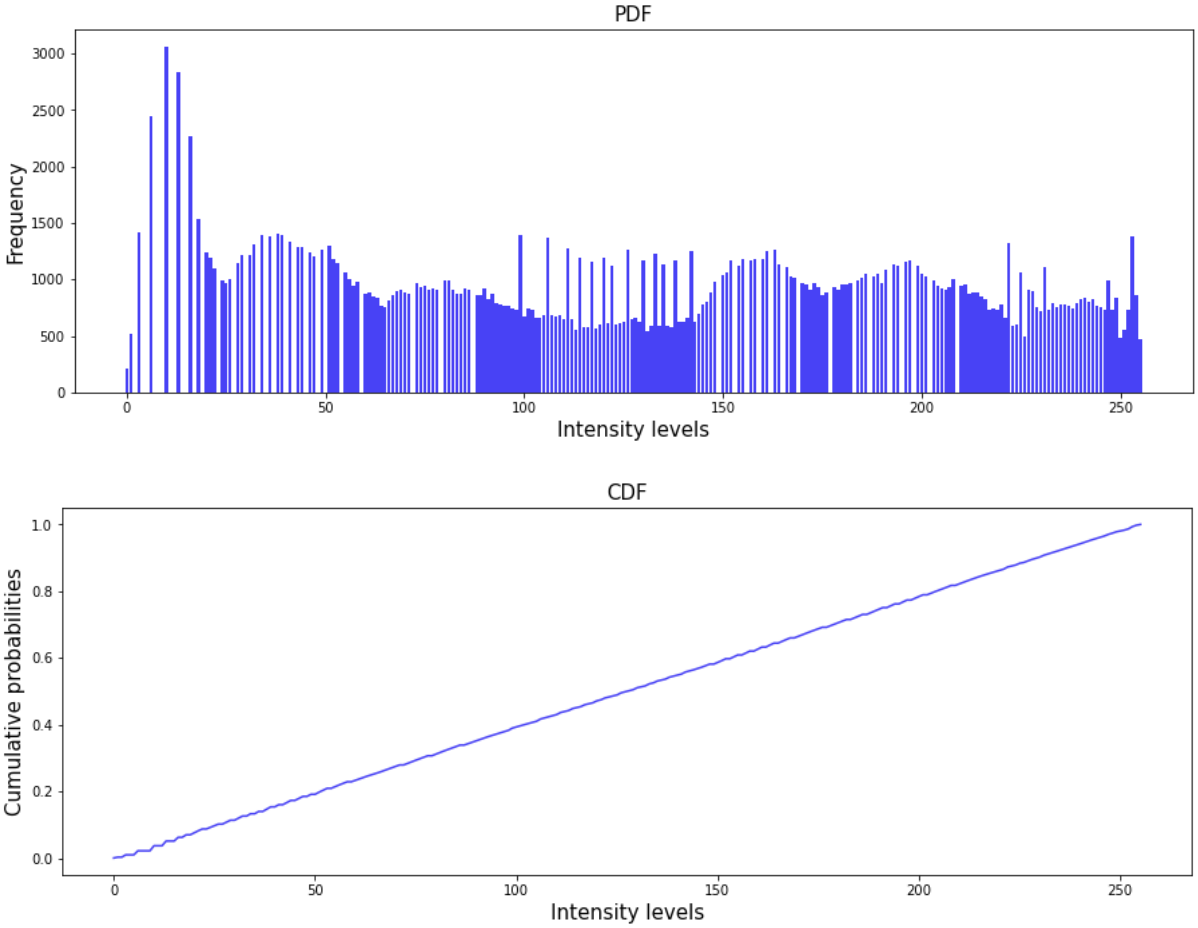
در این قسمت تابع HW2_Histeq فراخوانی شده و src_img به عنوان تصویر اولیه به این تابع پاس داده می شود سپس تصویر حاصل از خروجی تابع نمایش داده شده است. تصویر خروجی با سطح روشنایی جدید مطابق شکل (۳) می باشد.



شکل (۳)

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

در این قسمت تابع Hist_CDF فراخوانی شده و تصویر خروجی قسمت قبل یعنی تصویر حاصل از تابع HW2_Histeq به این تابع پاس داده می شود سپس نمودارهای حاصل از خروجی تابع نمایش داده شده است. نمودار های pdf و cdf تصویر Hi.tif پس از اعمال global equalization مطابق شکل (۴) می باشد.



شکل (۴)

بلوک [9] فایل ipynb : تابع Padding()

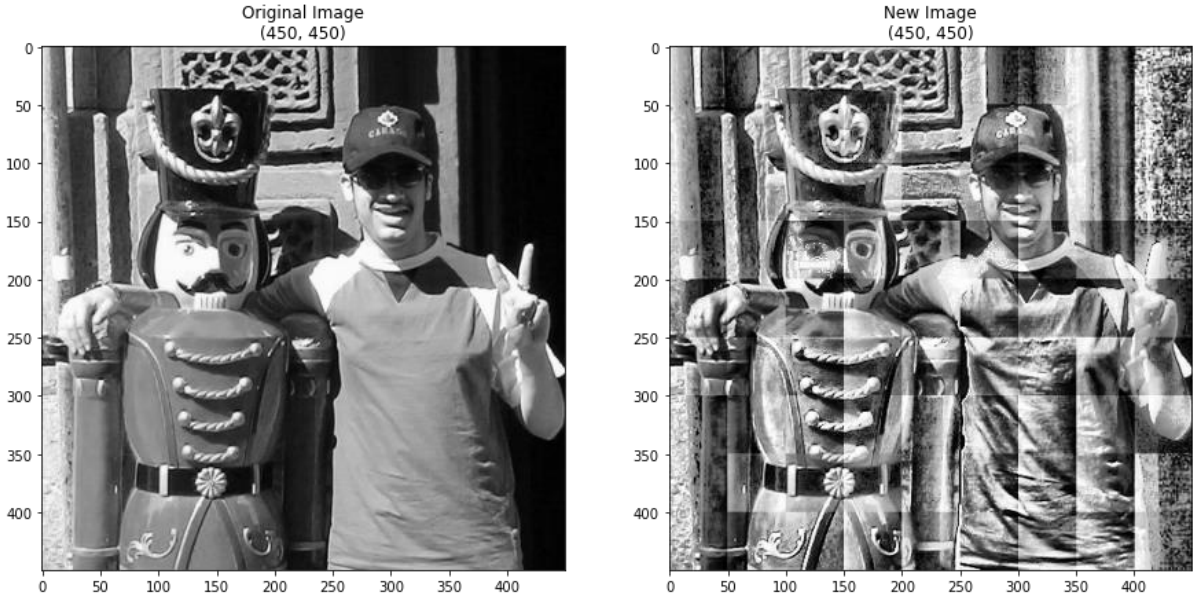
در این قسمت تابعی تحت عنوان Padding پیاده سازی شده است که تصویر اصلی را به میزان دلخواه pad کرده و ابعاد تصویر را گسترش می دهد. این تابع به عنوان ارگومان ورودی یک تصویر src_img و پارامتر pixel را دریافت کرده و یک تصویر پد شده به اندازه (طول تصویر اصلی + pixel) * (عرض تصویر اصلی + pixel) ایجاد می نماید. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [10] فایل ipynb : تابع HW2_Local_Histeq()

در این قسمت تابعی تحت عنوان HW2_Local_Histeq پیاده سازی شده است که تصویر اصلی را به اندازه های n * n تبدیل کرده و سپس بر روی هر قسمت local equalization را اعمال می کند. چنانچه ابعاد تصویر مضرب n نباشد ابتدا به میزان مورد نیاز تصویر اصلی را به وسیله تابع padding پد کرده و سپس تقسیم بندی می کنیم. هر قسمت n * n از تصویر را به تابع HW2_Histeq پاس داده و قسمت equalized شده را از خروجی تابع دریافت کرده و با همان بخش از تصویر پیکسل به پیکسل جایگذاری می نماییم. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند. نهایتاً پیکسل های پد شده از تصویر حذف گشته و تصویری با ابعاد اولیه به عنوان تصویر local equalized شده برگردانده می شود.

بلوک [11] فایل ipynb : فراخوانی تابع HW2_Local_Histeq() و نمایش تصویر خروجی

در این قسمت تابع HW2_Local_Histeq فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان n با مقدار ۵۰ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۵) می باشد.

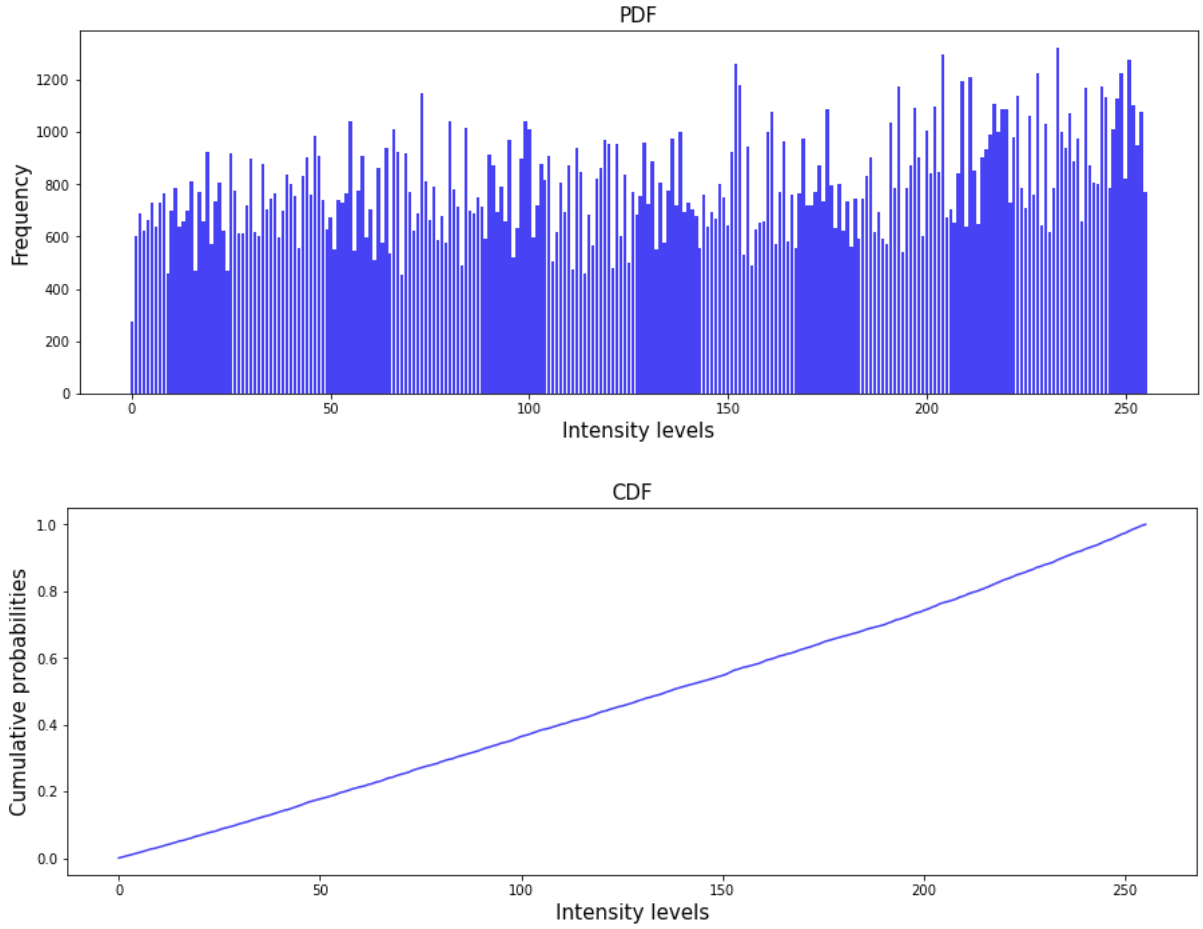


شکل (۵)

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

در این قسمت تابع Hist_CDF فراخوانی شده و تصویر خروجی قسمت قبل یعنی تصویر حاصل از تابع HW2_Local_Histeq به این تابع پاس داده می شود سپس نمودارهای حاصل از خروجی تابع نمایش داده

شده است. نمودار های pdf و cdf تصویر Hi.tif پس از اعمال local equalization مطابق شکل (۶) می باشد.



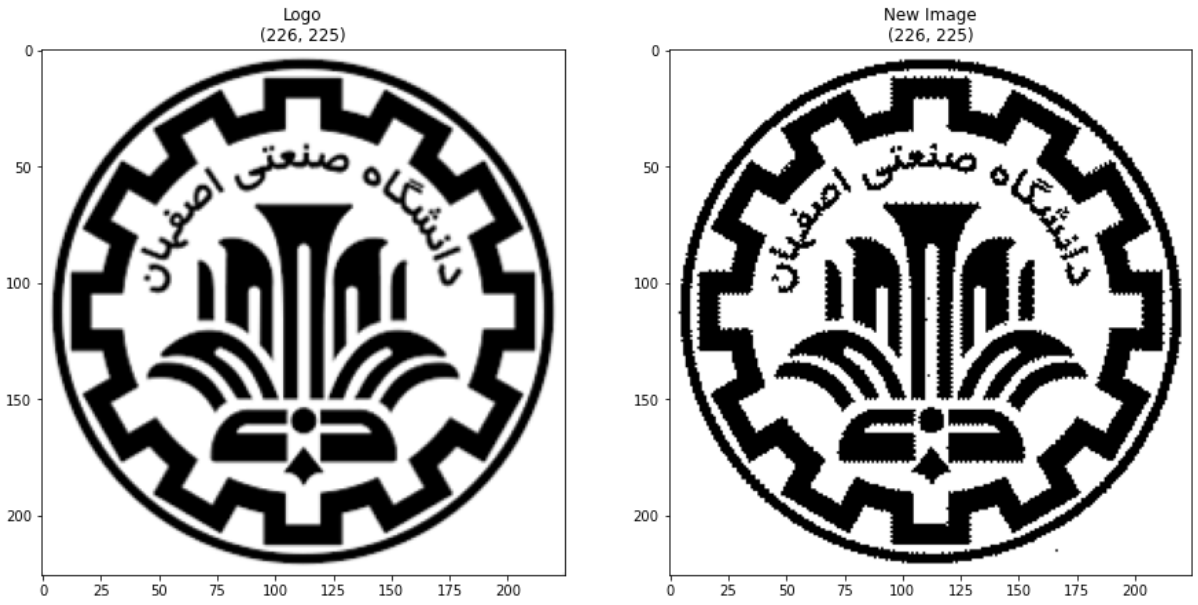
شکل (۶)

بلوک [13] فایل ipynb : تابع Make_Onebit()

در این قسمت تابعی تحت عنوان Make_Onebit پیاده سازی شده است که تصویری به عنوان ورودی دریافت کرده و تک بیتی شده تصویر را ایجاد می نماید. در این تابع از روش فلوید استاینبرگ برای تک بیتی کردن تصویر استفاده شده است. در این روش ارور ایجاد شده ضمن تک بیتی کردن هر پیکسل به پیکسل های همسایه منتقل خواهد شد.

بلوک [14] فایل ipynb : فراخوانی تابع Make_Onebit() و نمایش تصویر خروجی

در این قسمت تابع Make_Onebit فراخوانی شده و logo به عنوان تصویر اولیه به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۷) می باشد.



شکل (۷)

بلوک [15] فایل ipynb : تابع HW2_MSE()

در این قسمت تابعی تحت عنوان HW2_MSE برای بدست آوردن MSE دو تصویر پیاده سازی شده است. برای پیاده سازی این تابع کافی است تک تک پیکسل های دو تصویر از یکدیگر کم شده و حاصل به توان ۲ رسانده شود سپس مجموع تمام مربعات بدست آمده بر تعداد کل پیکسل های یک تصویر تقسیم گردد. لازم به ذکر است که ابعاد دو تصویر می بایست یکسان باشد.

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

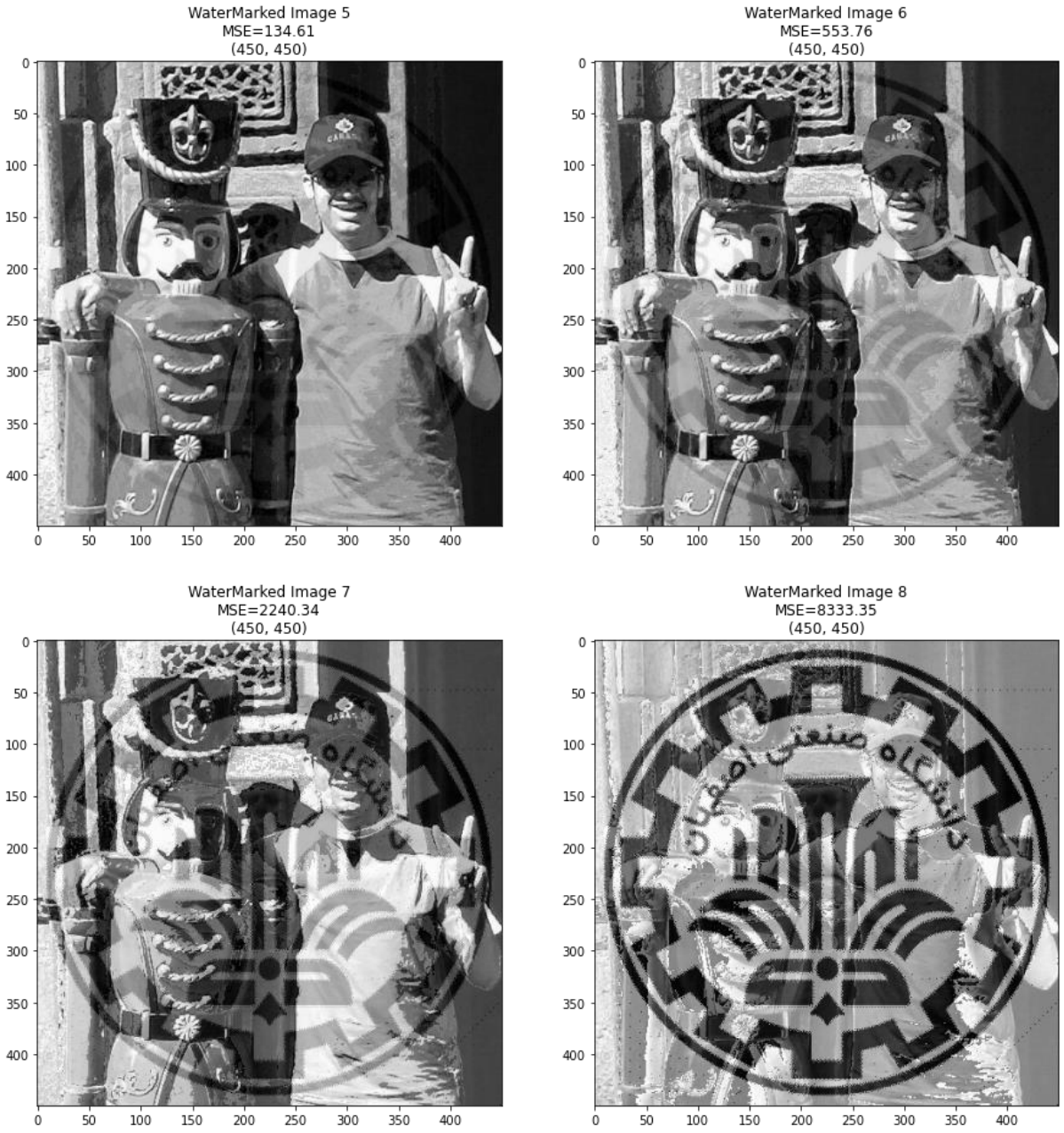
نکته قابل ملاحظه این است که پیکسل های تصاویر از نوع داده unit8 هستند و لذا برای جلوگیری از سرریز ضمن عملیات تفریق ابتدا نوع داده آن ها را به int تبدیل کرده و سپس تفاضل را صورت می دهیم.

بلوک [16] فایل ipynb : تابع HW2_Hide()

در این قسمت تابعی تحت عنوان HW2_Hide پیاده سازی شده است که یک تصویر را درون تصویر اصلی و در لایه L ام آن پنهان سازی می نماید. برای پیاده سازی این تابع نخست لازم است سایز تصویر لوگو با استفاده از تابع cv2.resize تغییر یافته و منطبق بر سایز تصویر اصلی شود سپس تصویر لوگو بدست آمده تک بیتی شده و در لایه L ام تصویر اولیه قرار داده می شود. تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [17] فایل ipynb : فراخوانی تابع HW2_Hide() و نمایش تصاویر خروجی

در این قسمت تابع HW2_Hide فراخوانی شده و src_img به عنوان تصویر اولیه logo به عنوان تصویری که قرار است پنهان سازی شود به تابع پاس داده می شود. تابع را به ازای L های از ۱ تا ۸ فراخوانی کرده و هر بار تصویر logo را در یکی از لایه های اول تا هشتم تصویر اصلی پنهان می کنیم و در انتها هر ۸ تصویر بدست آمده به همراه MSE هرکدام در خروجی نمایش داده شده است که در شکل (۸) قابل مشاهده می باشد.


WaterMarked Image 5
MSE=134.61
(450, 450)
WaterMarked Image 6
MSE=553.76
(450, 450)
WaterMarked Image 7
MSE=2240.34
(450, 450)
WaterMarked Image 8
MSE=8333.35
(450, 450)

شکل (۸)