گزارش تمرین Image Fundamentals و Contrast Adjustment

مریم واقعی

اطلاعات گزارش

تاریخ:
۱۳ آبان ۱۴۰۱

واژگان کلیدی:
گسسته سازی
متعادل سازی
خطای MSE
واریانس
کنتراست
Downsampling
Upsampling
Averaging
Pixel Replication
Bilinear Interpolation
متعادل سازی
متعادل سازی محلی

چکیده

ما در این گزارش پیاده سازی تمرین Image Fundamentals و Contrast Adjustment را با استفاده از زبان برنامه نویسی متلب انجام داده ایم. در تمرین اول ابتدا گسسته سازی را بدون متعادل سازی و سپس با متعادل سازی انجام داده ایم. سپس downsampling به دو روش میانگین گیری و حذف سطروستون و همچنین upsampling به روش pixel replication و bilinear interpolation انجام داده ایم. در بخش آخر تمرین اول ما داده را با تعداد کمتری بیت نمایش داده ایم که ابتدا بیت های کم ارزش را نگه داشتیم و بیت های با ارزش را حذف کردیم و خروجی را نمایش دادیم و سپس بیت های با ارزش را نگه داشتیم و بیت های کم ارزش را حذف کردیم و خروجی را نمایش دادیم و با هم مقایسه کردیم.

در تمرین دوم ابتدا متدی برای محاسبه هیستوگرام تصویر پیدا کردیم سپس توابع histogram equalization و local histogram equalization را پیاده کردیم. همچنین توابع تبدیل log و inverse log و تابع نمایی را روی تصاویر با مقادیر مختلف پارامتر های ورودی اعمال کردیم و نتایج را باهم مقایسه و بررسی کردیم.

فهرست مطالب

۱-مقدمه

در ابتدا ما نحوه پیاده سازی بخش های مختلف تمرین را توضیح داده ایم. سپس در بخش بعدی نتایج را نمایش میدهیم و با هم مقایسه میکنیم که برای مقایسه برخی نتایج مانند نتایج گسسته سازی با استفاده از متعادل سازی و بدون متعادل سازی و همچنین downsampling و upsampling ما از خطای MSE استفاده کرده ایم. در نهایت در بخش پیوست ما کد های هر بخش از تمرین را قرار داده ایم.

۲- توضیحات فنی

۲-۱- تمرین Image Fundamentals

۲-۱-۱- بخش اول تمرین

در این بخش ما باید در سطوح ۴، ۸، ۱۶، ۳۲ و ۶۴ تصویر را گسسته سازی کنیم که برای انجام این کار من از روش نرمال سازی که در درس گفته شده استفاده کردم یعنی برای اینکه داده ها را از بازه ی [min, max] به بازه ی [new_min, new_max] ببریم از فرمول زیر استفاده میکنیم:

(۱)

$$Value = \lfloor (new\_\max - \text{new\_min}) * (Value - \min)/(\max - min) \rfloor + new\_min$$

که این فرمول ابتدا اعداد را از بازه [min, max] به بازه [0,max-min] می برد سپس عدد را تقسیم بر max-min میکند تا بازه ی اعداد به [0,1] منتقل شود و پس از آن در new_max-new_min ضرب میشود و در نهایت با new_min جمع میشود تا به بازه ی جدید منتقل شود.

ما میخواهیم بازه ی اعداد را از ۰ تا ۲۵۵ به بازه ی ۰ تا Level-1 که Level همان سطوح مشخص شده در بالا می باشد منتقل کنیم که با استفاده از فرمول بالا این تبدیل را انجام میدهیم.

پس از انجام تبدیل نتیجه را در متغیر quantized_image ذخیره کردیم.

نکته ای که در اینجا هست این است که چون مقادیر تصویر تبدیل یافته محدود به بازه ی ۰ تا Level-1 می باشد، برای نمایش تصویر با همین مقادیر به مشکل برمی‌خوریم چون در نمایش تصویر بازه اعداد بین ۰ تا ۲۵۵ هست ولی مقادیر ما بازه محدودتری از اعداد را دارند. مثلا اگر تعداد سطوح ما ۴ باشد بازه مقادیر بین ۰ تا ۳ است و در نتیجه اگر تصویر را با همین مقادیر ۰ تا ۳ نمایش دهیم یک تصویر تماما سفید خواهیم داشت چون بازه ای که در نمایش تصویر در نظر گرفته شده ۰ تا ۲۵۵ است.

پس در این مرحله کاری که میکنیم این است که همین تعداد سطوح را به بازه ی ۰ تا ۲۵۵ میبریم تا تصویر قابل نمایش باشد و برای اینکار از فرمول زیر استفاده میکنیم:

(۲)

$$a = \left(\frac{quantizedimage}{\max}\right) * 255$$

حال خروجی ما یک تصویر در بازه ی ۰ تا ۲۵۵ می باشد که تعداد مقادیری که از این بازه اتخاذ کرده ایم به اندازه Level می باشد.

مثلا مقادیر ۰ تا ۳ برای Level=4 به مقادیر ۰ و ۸۵ و ۱۷۵ و ۲۵۵ در بازه ی ۰ تا ۲۵۵ نگاشت شده است.

حال روی خروجی نهایی بدست آمده histeq را اعمال میکنیم تا هیستوگرام خروجی متعادل شود.

پس از اینکار خطای MSE[1] برای هر دو تصویر equalize شده و غیر equalize شده در مقایسه با تصویر اصلی با استفاده از متد immse بدست آمده که خروجی را در جدول ۱ گزارش داده ایم.

در نهایت برای هر Level خروجی تصویر گسسته شده بدون equalization و با equalization به همراه هیستوگرام مربوط به هر یک نمایش داده شده و در فایل png ذخیره میشود.

۲-۱-۲- بخش دوم تمرین

در این بخش ابتدا downsamlpling را بدون میانگین گیری انجام داده ایم که برای اینکار سطر و ستون ها را یکی در میان حذف کرده ایم.

سپس downsampling را با میانگین گیری انجام داده ایم که برای اینکار ابتدا فیلتر میانگین را به صورت [0.25 0.25; 0.25 0.25] تعریف کرده ایم سپس تصویر را به بلاک های ۲ در ۲ تقسیم میکنیم و هر بلاک را در این فیلتر ضرب میکنیم و مقادیر نهایی را با هم جمع میکنیم و عدد حاصل را در خانه متناظر در تصویر downsample شده قرار میدهیم و در نهایت خروجی بدون میانگین گیری را در تصویر ۱۴ و خروجی با میانگین گیری را در تصویر ۱۵ نمایش داده ایم.

Pixel replication:

همچنین برای upsampling به روش pixel Replication مقدار هر پیکسل را در ۴ خانه مجاور هم قرار میدهیم. به این صورت که روی تصویر upsample پیکسل ها را یکی در میان حرکت میکنیم و به ازای هر مکان i و j در تصویر upsample مقدار پیکسل (i/2,j/2) از تصویر downsample را در مکان های (i-1,j-1) و (i,j-1) و (i-1,j) و (i,j) قرار میدهیم.

به عبارتی ما در میانگین گیری میانگین ۴ خانه مجاور را در پیکسل متناظر در تصویر downsample قرار دادیم و در upsample به روش pixel replication مقدار پیکسل مورد نظر در تصویر downsample را در ۴ خانه مجاور در تصویر upsample قرار میدهیم.

Bilinear interpolation:

در upsample به روش replication ابتدا به صورت یکی در میان سطر و ستون به تصویر اضافه کردیم. سپس برای آسانی محاسبات padding به تصویر اضافه کردیم.

[1] Mean Squared Error

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	228	0	236	0	228	0	228	0	228	0	236
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	228	0	236	0	236	0	228	0	228	0	228
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	228	0	236	0	236	0	236	0	236	0	236
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	236	0	228	0	228	0	228	0	236	0	228
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	228	0	228	0	228	0	228	0	236	0	236
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	236	0	228	0	228	0	228	0	236	0	228

تصویر ۱- بخشی از ماتریس تصویر پس از اضافه کردن سطر و ستون و padding

پس از اینکار با step=2 از خانه ی (3,3) تصویر که در مرکز ۴ مقدار قطری اطراف آن قرار دارد روی ماتریس تصویر حرکت میکنیم تا مقادیر پیکسل ها را پرکنیم.

نکته ای که وجود دارد این است که ۴ خانه های مجاوری که برای محاسبه مقدار هر پیکسل درنظر میگیریم با خود پیکسل فاصله یکسانی دارند پس وزن ۴ پیکسل با هم برابر است و تنها باید بین مقادیر این پیکسل ها میانگین گرفته شود.

ما دو دسته پیکسل داریم که روش مقدار دهی پیکسل های آنها را در زیر توضیح میدهیم:

۱. دسته اول پیکسل های مرکزی هستند که همسایه های قطری آن ها مقدار دارند و با حلقه روی این پیکسل ها حرکت میکنیم مثل پیکسل (3,3) در ماتریس بالا.

   برای محاسبه مقدار این پیکسل ما از فیلتر [0.25 0 0.25; 0 0 0; 0.25 0 0.25] استفاده میکنیم و این ماتریس را با ماتریس ۳*۳ از تصویر که پیکسل مرکزی در مرکز آن قرار دارد ضرب داخلی میکنیم و بعد مقادیر را با هم جمع میکنیم تا مقدار پیکسل مرکزی بدست آید.

۲. دسته دوم پیکسل های سمت چپ و بالای پیکسل مرکزی هستند که این پیکسل ها همسایه های ۴ گانه شان مقدار دارند و به همین دلیل از فیلتر [0 0.25 0; 0.25 0 0.25; 0 0.25 0] استفاده میکنیم و با ماتریس ۳*۳ که این پیکسل ها در مرکز آن قرار دارند ضرب داخلی میکنیم و پس از آن مقادیر را جمع میکنیم و در پیکسل متناظر قرار میدهیم.

به عبارتی در هر step من پیکسل مرکزی و سمت چپی و بالایی آن را مقدار میدهم و به همین ترتیب در تصویر حرکت میکنیم تا در نهایت تمام پیکسل ها مقدار بگیرند.

۲-۱-۳- بخش سوم تمرین

بازه ی مقادیر تصویر بین ۰ تا ۲۵۵ می باشد و ما میخواهیم به ترتیب ابتدا کم ارزش ترین بیت تا در نهایت ۵ تا کم ارزش ترین بیت ها را نگه داریم و بقیه بیت ها را حذف کنیم و پس از آن با ارزش ترین بیت تا ۵ تا با ارزش ترین بیت ها را نگه داریم و بقیه بیت ها را حذف کنیم تا ببینیم خروجی حاصل چه خواهد بود.

برای اینکار ما تصویر را برای ۱ بیت کم ارزش تا ۵ بیت کم ارزش به ترتیب با مقادیر ۱، ۳، ۷، ۱۵ و ۳۱ and بیتی میکنیم تا بیت های پر ارزش حذف شود و تنها تعداد بیت مدنظر ما از مقادیر پیکسل های تصویر باقی بماند که خروجی تصاویر در تصویر ۲۲ تا ۲۶ می باشد.

برای با ارزش ترین بیت تا ۵ تا با ارزش ترین بیت هم به ترتیب پیکسل های تصویر را با مقادیر ۸، ۲۴، ۵۶، ۱۲۰ و ۲۴۸ and بیتی میکنیم تا بیت های کم ارزش حذف شوند و تنها تعداد بیت مدنظر ما باقی بماند و خروجی نتایج در تصاویر ۲۷ تا ۳۱ می باشد، همچنین نتایج را در بخش بررسی نتایج با هم مقایسه و تحلیل کرده ایم.

۲-۲- تمرین Contrast Adjustment

۲-۲-۱- بخش اول تمرین

محاسبه هیستوگرام تصویر:
در این بخش ابتدا یک تابع به اسم computeHist نوشته ایم که ورودی آن تصویر و تعداد سطوح خاکستری می باشد.
سپس یک آرایه یک بعدی با مقدار دهی اولیه صفر تعریف کردیم که تعداد خانه های آن برابر با تعداد سطوح خاکستری می باشد. سپس روی پیکسل های تصویر حرکت میکنیم و خانه ی متناظر با مقدار هر پیکسل را یکی اضافه میکنیم. مثلا اگر مقدار پیکسل ۰ است، مقدار خانه ی اول آرایه را یکی افزایش میدهیم، یا مثلا اگر مقدار پیکسل ۱۲۳ است، مقدار خانه ی ۱۲۴ ام آرایه را یکی اضافه میکنیم و به همین ترتیب تعداد مقادیر پیکسل ها در خانه های متناظر با آن مقدار ذخیره می شود.
در نهایت در فایل دیگری ما تصویر Camera Man را خوانده و ارایه ی مربوط به هیستوگرام تصویر را از تابع computeHist دریافت میکنیم و درنهایت تصویر Camera Man را در کنار نمودار هیستوگرام آن که با متد bar رسم شده نمایش داده ایم. خروجی را میتوانید در تصویر ۳۲ مشاهده کنید.
کاهش روشنایی تصویر:
در بخش بعدی مقادیر پیکسل های تصویر اصلی را تقسیم بر ۳ کردیم و آن را در متغیری به نام D ذخیره کردیم و پس از آن هیستوگرام تصویر را با استفاده از تابع از پیش نوشته شده ی computeHist بدست آوردیم و در نهایت هیستوگرام تصویر اصلی و تصویر D را رسم کردیم و آن را در تصویر ۳۳ مشاهده میکنید.

توابع تبدیلات هندسی:
در ابتدا تابع تبدیل لگاریتم را با فرمول ۳ روی مقادیر پیکسل ها اعمال میکنیم. اما قبل از اعمال فرمول مقادیر پیکسل ها را تقسیم بر ۲۵۵ میکنیم تا مقادیر نرمالایز شده و بازه مقادیر بین ۰و۱ قرار بگیرد و پس از آن توابع تبدیل را روی مقادیر اعمال میکنیم.
(۳)

$$c * \log(1 + r)$$

مقدار c در فرمول بالا را ۰.۵ و ۱ و ۲ قرار دادیم که خروجی آن را در تصویر میتوانید مشاهده کنید.
برای تابع تبدیل inverse log از فرمول ۴ استفاده کردیم که مشابه تابع تبدیل لگاریتم مقدار c را ۰.۵ و ۱ و ۲ قرار دادیم.
(۴)

$$(e^r)^{1/c}$$

برای تابع تبدیل نمایی از فرمول ۵ استفاده کردیم که مقدار c را ۰.۵ و ۱و ۱.۵ قرار دادیم و مقدار گاما را ۰.۳، ۰.۵، ۱، ۳ و ۵ قرار دادیم.

(۵)

$$c * r^{\gamma}$$

متعادل سازی محلی و سراسری:

متعادل سازی سراسری به این صورت نوشته شده که ابتدا تصویر ورودی به همراه تعداد سطوح خاکستری تصویر دریافت میشود سپس در گام اول هیستوگرام تصویر با استفاده از متد computeHist() نوشته شده در بخش های قبلی محاسبه میشود سپس تعداد پیسکل های هر سطح خاکستری تقسیم بر تعداد کل پیکسل های تصویر میشود تا احتمال رخداد هر سطح خاکستری یعنی pdf بدست آید. پس از آن با بدست آوردن مقدار cdf، توزیع تجمعی را محاسبه میکنیم.

اگر مقادیر cdf را در ماکزیمم مقدار gray level ضرب کنیم و مقدار آن را گرد کنیم در حقیقت آرایه ای که بدست می آید که نگاشت سطوح خاکستری به مقادیر متعادل سازی شده را به ما میدهد که با استفاده از این آرایه مقادیر پیکسل های تصویر را به مقادیر متعال سازی شده نگاشت میشود و تصویر متعادل سازی شده را بدست میدهد.

در متعادل سازی محلی ما تصویر را به بلاک هایی تقسیم کردیم و هر بلاک را به تابع histEqualization می دهیم و متعادل سازی را روی هر بلاک اعمال میکنیم و آن بلاک را پس از متعادل سازی در مکان متناظر در ماتریس تصویر خروجی قرار میدهیم. نکته ای که هست این است که برای آنکه تصویر خروجی نرم تر شود و بخش بندی ها در تصویر دیده نشود دو کار انجام دادیم:

۱. سایز بلاک ها را 1/64 ام سایز تصویر درنظر گرفتیم و بلاک ها را برزگتر از این در نظر نگرفتیم چون در این صورت هر بلوک هر تصویر خروجی مشخص میشد و همین موضوع کیفیت تصویر را کاهش میداد.
۲. بلاک ها را همپوشانی داده ایم تا خروجی نرم تر باشد و نویز کمتری در تصویر خروجی باشد.

پس از پیاده سازی این دو خروجی متعادل سازی سراسری را در متغیر H و خروجی متعادل سازی محلی را در L قرار دادیم که در تصویر ۳۹ میتوانید مشاهده کنید.

۲-۲-۲- بخش دوم تمرین

در این بخش پیاده سازی equalization سراسری را باید انجام داده باشیم که به دلیل استفاده از توابع آن در بخش قبل، آن را به طور کامل در انتهای بخش اول تمرین توضیح داده ایم که میتوانید آن را مشاهده کنید. در این بخش تنها تابع histEqualization را روی تصویر Camera Man اعمال کردیم که خروجی آن را در بخش بررسی نتایج میتوانید مشاهده کنید.

۲-۲-۳- بخش سوم تمرین

تابع imadjust کنتراست تصویر را با نگاشت مقدار شدت ورودی به مقادیر جدید افزایش میدهد، به طوری که به طور پیش فرض ۱ درصد از داده ها در شدت های کم و زیاد اشباع میشوند.

اما تابع histeq متعادل سازی هیستوگرام را انجام میدهد. کنتراست تصاویر را با تبدیل مقادیر در یک تصویر افزایش میدهد تا هیستوگرام تصویر خروجی تقریبا با هیستوگرام مشخص شده (که به طور پیش فرض تابع uniform است) مطابقت داشته باشد.

همچنین تابع imadjust هیستوگرام را به صورت خطی scale میکند در حالی که histeq اینطور نیست و به همین دلیل خروجی imadjust طبیعی تر به نظر میرسد.

۳- بررسی نتایج

۳-۱- تمرین Image Fundamentals

۳-۱-۱- بخش اول تمرین

در جدول زیر خطا برای تصویر equalized و nonequalized برای تعداد سطوح خاکستری مختلف با معیار خطای MSE گزارش شده است:

جدول ۱- گزارش خطای MSE برای تصاویر equalized و nonequalized در سطوح خاکستری مختلف

128	64	32	16	8	4	Levels
40.0525	29.6005	17.92	32.3378	247.8818	1981.042	Without equalization
971.2648	986.1608	961.4193	973.123	1042.2789	996.1926	With equalization

همانطور که مشاهده میکنید مقدار خطا در حالتی که هیستوگرام متعادل سازی شده خیلی بیشتر از حالتی است که هیستوگرام متعادل سازی نشده و به نظرم این به این دلیل است که چون در متعادل سازی سعی میشود تا نمودار هیستوگرام نرمال سازی شود و داده هایی که داریم فقط مربوط به چند سطح خاکستری خاصِ خود تصویر نباشد، به همین دلیل مقادیر پیکسل ها در سطوح مختلف پخش شده و هیستوگرام متعادل سازی شده تطابق کمتری با هیستوگرام تصویر اصلی دارد در نتیجه اختلاف تصویر اصلی با تصویر متعادل سازی شده بیشتر است.

همچنین هرچه تعداد سطوح خاکستری بیشتر میشود خطای MSE کمتر میشود که آن هم به این دلیل است که هرچه تعداد سطوح خاکستری بیشتر شود تصویر را با مقادیر بیشتری میتوان نمایش داد و درنتیجه کیفیت تصویر به تصویر اصلی نزدیکتر خواهد بود.

این متعادل سازی و عدم تمرکز بیشتر پیکسل ها روی چند سطح خاص در نمودار ۶ به خوبی قابل مشاهده است.

خروجی تصاویر برای هر یک از سطوح مشخص شده در حالت equalized و nonequalized به صورت زیر می باشد:

تصویر ۲- تصویر با ۴ سطح خاکستری nonequalized

تصویر ۳- تصویر با ۴ سطح خاکستری equalized

تصویر ۴ - تصویر با ۸ سطح خاکستری nonequalized

تصویر ۵ - تصویر با ۸ سطح خاکستری equalized

تصویر ۶- تصویر با ۱۶ سطح خاکستری nonequalized

تصویر ۷- تصویر با ۱۶ سطح خاکستری equalized

تصویر ۸- تصویر با ۳۲ سطح خاکستری nonequalized

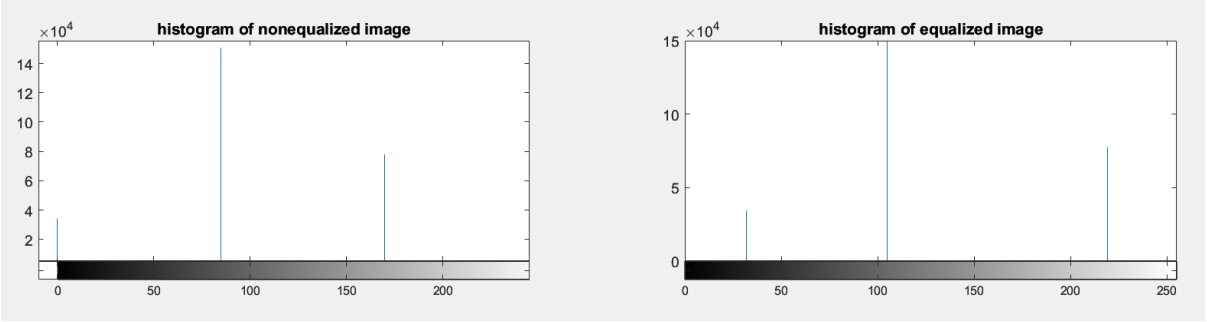
تصویر ۹- تصویر با ۳۲ سطح خاکستری equalized

تصویر ۱۰- تصویر با 64 سطح خاکستری nonequalized

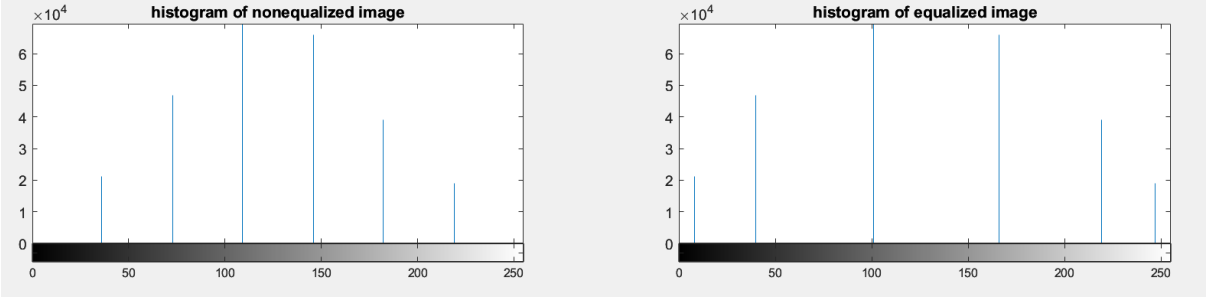
تصویر ۱۱- تصویر با ۶۴ سطح خاکستری equalized

تصویر ۱۲- تصویر با ۱۲۸ سطح خاکستری nonequalized

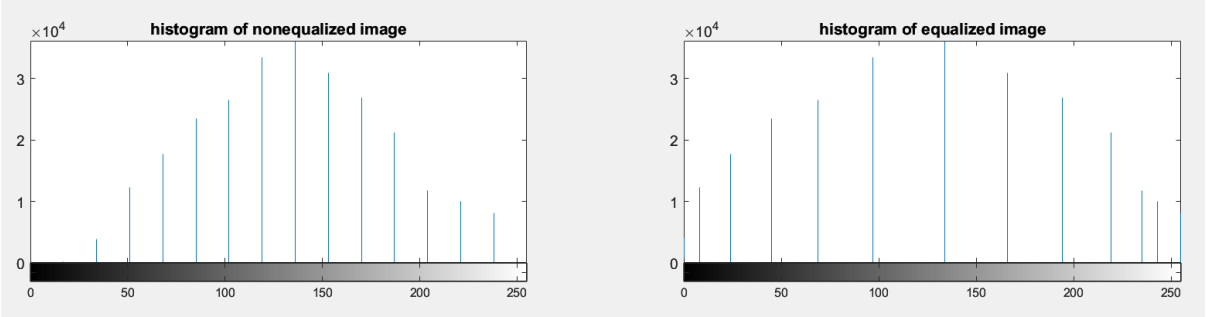
تصویر ۱۳- تصویر با ۱۲۸ سطح خاکستری equalized



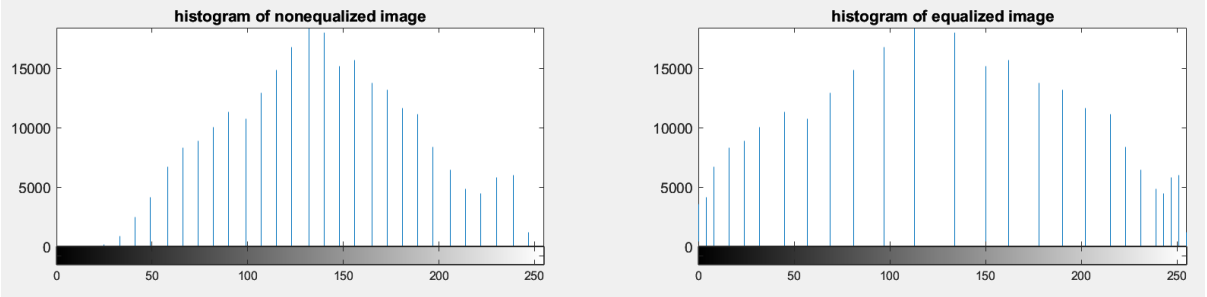
نمودار ۱- نمودار هیستوگرام تصویر با ۴ سطح خاکستری



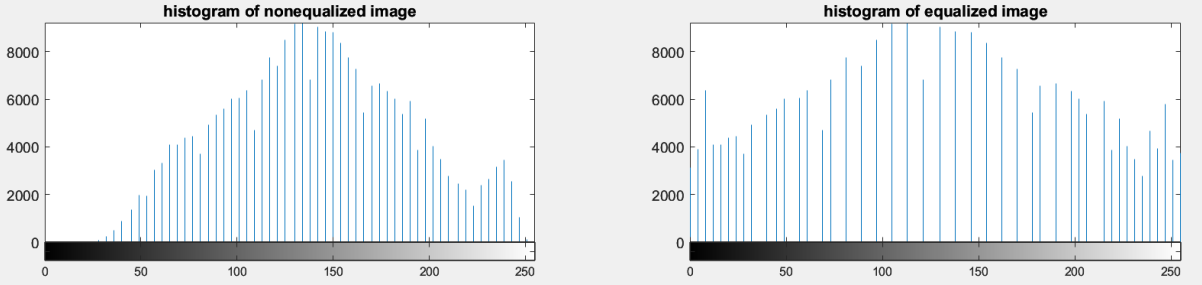
نمودار ۲- نمودار هیستوگرام تصویر با ۸ سطح خاکستری



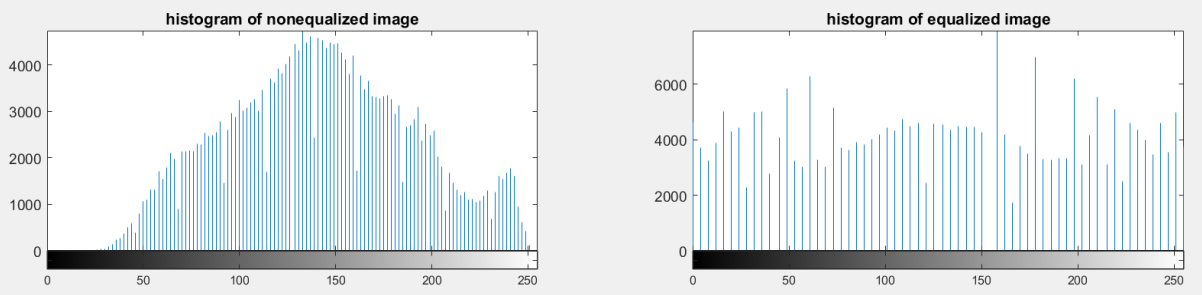
نمودار ۳- نمودار هیستوگرام با ۱۶ سطح خاکستری



نمودار ۴- نمودار هیستوگرام با ۳۲ سطح خاکستری



نمودار ۵- نمودار هیستوگرام تصویر با ۶۴ سطح خاکستری



نمودار ۶- نمودار هیستوگرام با ۱۲۸ سطح خاکستری

همانطور که در خروجی تصاویر بالا مشاهده میکنیم اولا هرچه تعداد سطوح خاکستری بیشتر میشود ما با تعداد سطح بیشتری میتوانیم تصویر را نمایش دهیم و در نتیجه کیفیت تصویر بیشتر میشود مثلا کیفیت تصویر در حالتی که ما ۱۲۸ سطح خاکستری مختلف داریم خیلی بیشتر از حالتی است که تنها ۴ سطح خاکستری متفاوت داریم، دوما در هیستوگرام تصویر متعادل شده مشاهده میکنیم که سعی شده تا پیکسل ها بین مقادیر مختلف پخش شوند و تمرکز پیکسل ها روی تنها چند پیکسل خاص نباشد و این باعث افزایش کیفیت و همچنین افزایش واریانس و در نتیجه افزایش کنتراست تصویر شده است.

۳-۱-۲- بخش دوم تمرین

خروجی تصاویر downsample شده به صورت زیر می باشد:

تصویر ۱۴- تصویر downsample شده با روش حذف سطر و ستون

تصویر ۱۵- تصویر downsample شده با فیلتر میانگین گیر ۲ در ۲

همانطور که در تصاویر بالا مشاهده میکنیم، تصویری که با روش میانگین گیری downsample شده نرم تر است و خروجی با کیفیت تر است.

تصاویر زیر بخشی زوم شده از تصاویر بالا هستند که به وضوح مشاهده میکنید تصویر downsample شده با روش حذف سطر و ستون کیفیت کمتر دارد و لبه ها مشخص تر است اما در تصویر downsample شده با فیلتر میانگین گیر چون میانگین هر ۴ پیکسل مجاور گرفته شده و در مکان متناظر قرار داده شده در نتیجه تصویر نرم تر و کیفیت آن بهتر است.

تصویر ۱۶- تصویر زوم شده downsample شده با روش حذف سطر و ستون

تصویر ۱۷- تصویر زوم شده downsample شده با فیلتر میانگین گیر ۲ در ۲

همچنین جدول خطای mse به صورت زیر می باشد:

	Pixel Replication	Bilinear Interpolation
Averaging	136.9358	165.6964
Remove Row&Column	133.0759	79.8993

همانطور که مشاهده میکنید خطای MSE در حالتی که تصویر را با روش حذف سطر و ستون downsample کنیم نسبت به روش میانگین گیری در هر دو روش pixel replication و Bilinear Interpolation کمتر است که نشان میدهد روش حذف سطر و ستون خروجی که میدهد به تصویر اصلی نزدیک تر است که به نظرم دلیلش این است که در روش حذف سطر و ستون لبه های تصویر حفظ میشود اما در روش میانگین گیری تصویر نرم میشود و در نتیجه بسیاری از لبه ها حذف میشوند.

در حالت میانگین گیری وقتی تصویر را upsample میکنیم خطا در روش pixel replication به نسبت bilinear interpolation کمتر است و دلیل آن این است که در روش bilinear چون میانگین وزن دار گرفته میشود تصویر نرم تر میشود و همین امر باعث میشود که تصویر به نسبت روش دیگر blur تر شود.

در حالت حذف سطر و ستون وقتی تصویر را upsample میکنیم خطا در روش bilinear interpolation نسبت به pixel replication کمتر است که به نظرم دلیل آن این است که در روش pixel replication چون ۴ خانه مجاور هم از تصویر upsample مقدار یک خانه متناظر در تصویر dounsample را دارند در نتیجه تصویر شطرنجی میشود و همین امر باعث میشود که کیفیت تصویر نسبت به روش bilinear که برای خانه های خالی از میانگین وزن دار استفاده میکند کمتر شود.

پس بهترین کیفیت را زمانی داریم که به روش حذف سطر و ستون downsample کردیم و با روش دو خطی upsample کردیم.

تصویر ۱۸- downsample به روش میانگین گیری و upsample به روش pixel replicaion

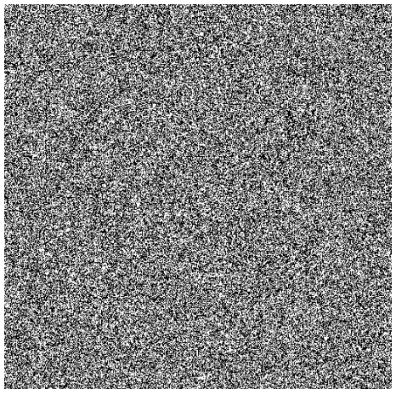
تصویر ۱۹- downsample به روش حذف سطر و ستون و upsample به روش pixel replicaion

تصویر ۲۰- downsample به روش میانگین گیری و upsample به روش bilinear interpolation

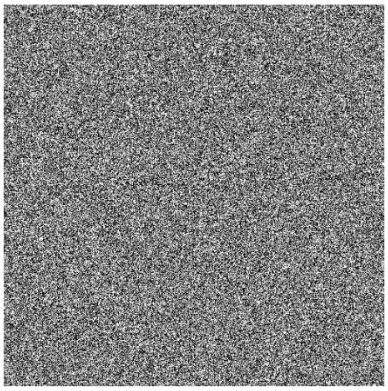
تصویر ۲۱- downsample به روش حذف سطر و ستون و upsample به روش bilinear interpolation

۳-۱-۳- بخش سوم تمرین

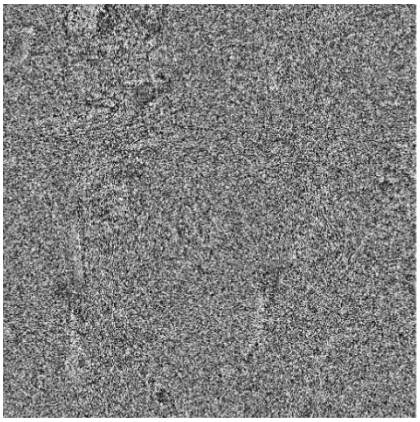
در پایین شما خروجی تصویر حاصل از حفظ ۱ تا ۵ بیت کم ارزش و ۱ تا ۵ بیت با ارزش را مشاهده میکنید:



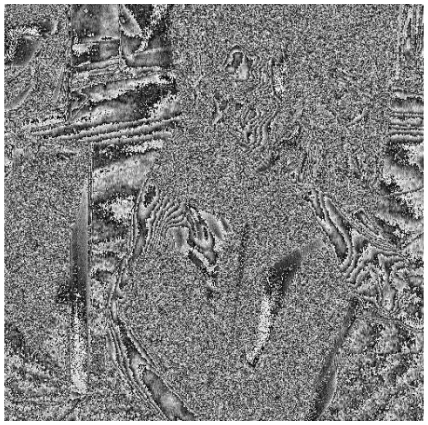
تصویر ۲۲ - خروجی تصویر حاصل از نگهداری bit-plane 0



تصویر ۲۳- خروجی تصویر حاصل از نگهداری bit-plane 0,1



تصویر ۲۴- خروجی تصویر حاصل از نگهداری bit-plane 0,1,2



تصویر ۲۵- خروجی تصویر حاصل از نگهداری bit-plane 0,1,2,3

تصویر ۲۶- خروجی تصویر حاصل از نگهداری bit-plane 0,1,2,3,4

تصاویر ۲۲ و ۲۳ صرفا یکسری تصویر نویزی میباشند که جزئیاتی از تصویر اصلی را به ما نشان نمیدهند و هرچه به سمت تصویر۶۲۱ می رویم تصاویر دارای خط و لبه هایی میشود که ما را به نمایش تصویر اصلی نزدیک میکند اما همچنان در تصویر۲۶ مشاهده میکنیم که باز هم نگهداری ۵ بیت کم ارزش برای بازنمایی تصویر اصلی کافی نیست.
حال در پایین تصاویری را مشاهده میکنیم که بیت های پر ارزش آن ها را نگه داشته و بیت های کم ارزش آن ها را حذف کرده ایم. همانطور که میبینیم کلیات تصویر حتی با یک بیت با ارزش هشتم هم مشخص است و هرچه بیت های با ارزش بیشتری برای نمایش تصویر نگه داریم کیفیت رنگ ها در تصویر بهتر میشود به طوری که در تصویر۳۰ و ۳۱ کیفیت تصویر همانند کیفیت تصویر اصلی می باشد. بنابراین نتیجه میگیریم که جزئیات تصویر در بیت های با ارزش ذخیره شده و کلیات و خط و لبه های تصویر در بیت های کم ارزش تر ذخیره شده، همچنین برای حذف بیت بهتر است بیت های کم ارزش مثلا بیت های اول تا چهارم را حذف کنیم تا کیفیت تصویر تا حد بسیار خوبی حفظ شود و همچنین حذف بیت های با ارزش مثل بیت هشتم و هفتم و ششم که خروجی آن ها را در تصاویر بالا مشاهده کردیم عملا به حفظ جزئیات تصویر کمک نمیکند و کیفیت تصویر حفظ نمیشود چون جزئیات در بیت های با ارزش ذخیره میشود و حذف بیت های با ارزش به معنای حذف جزئیات تصویر می باشد.

تصویر ۲۷- خروجی تصویر حاصل از نگهداری bit-plane 7

تصویر ۲۸- خروجی تصویر حاصل از نگهداری bit-plane 6,7

تصویر ۲۹- خروجی تصویر حاصل از نگهداری bit-plane 5,6,7

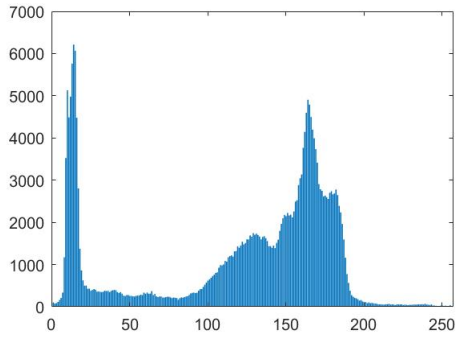
تصویر ۳۰- خروجی تصویر حاصل از نگهداری bit-plane 4,5,6,7

تصویر ۳۱- خروجی تصویر حاصل از نگهداری bit-plane 3,4,5,6,7

۳-۲- تمرین Contrast Adjustment

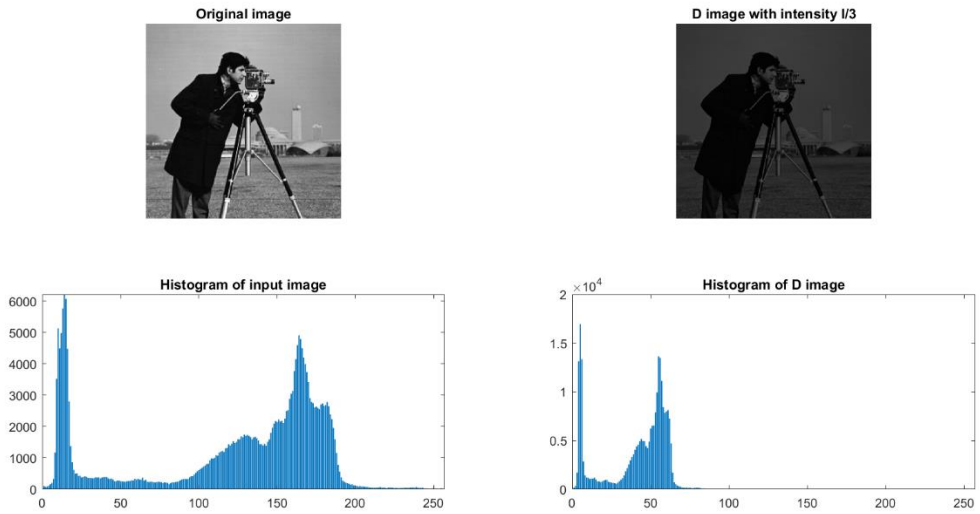
۳-۲-۱- بخش اول تمرین

محاسبه هیستوگرام تصویر:
خروجی تصویر Camera Man را به همراه نمودار هیستوگرام آن که به وسیله ی متد bar نمایش داده شده در زیر مشاهده میکنید:



تصویر ۳۲- تصویر Camera Man به همراه نمودار هیستوگرام آن

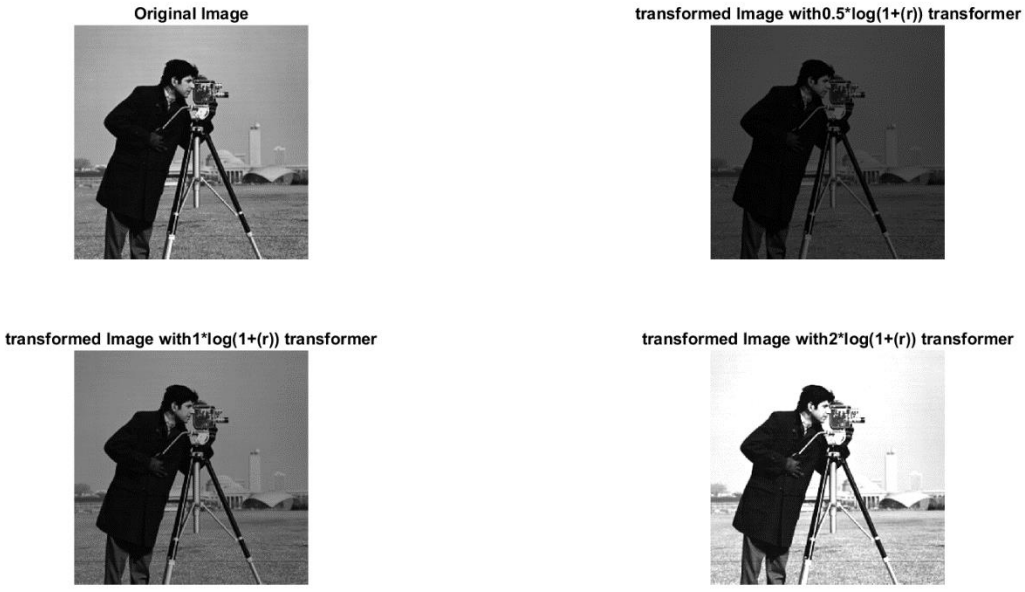
کاهش روشنایی تصویر:
در تصویر زیر هیستوگرام تصویر Camera Man اصلی و تصویر D که روشنایی آن کاهش داده شده را مشاهده میکنید.
همانطور که میبینید شکل هیستوگرام تصویر اصلی و تصویر D تقریبا مشابه است با این تفاوت که در تصویر D هیستوگرام آن فشرده تر شده و مقادیر تصویر اصلی در بازه فشرده تری از اعداد قرار گرفته اند و اگر در تصویر اصلی بازه مقادیر بین ۰ تا ۲۵۵ است در تصویر D این بازه به ۰ تا ۸۵ تبدیل شده است و در نهایت در تصویر D شما مشاهده میکنید که روشنایی پیکسل ها به یک نسبت کاهش یافته و تصویر تیره تر شده است.



تصویر ۳۳- تصویر Camera Man به همراه هیستوگرام تصویر اصلی و تصویر D

توابع تبدیل هندسی:

همانطور که در تصویر زیر مشاهده میکنید خروجی زیر مربوط به اعمال تابع تبدیل لگاریتم روی تصویر Camera Man هست که مقدار c در تصویر بالا سمت راست ۰.۵ و در تصویر پایین سمت چپ ۱ و در تصویر پایین سمت راست ۲ می باشد. همانطور که میبینیم مقدار c در حقیقت تمام پیکسل های تصویر را به یک نسبت افزایش یا کاهش میدهد یعنی وقتی c بین ۰و۱ است، مقدار همه پیکسل ها را به یک نسبت کاهش میدهد و وقتی c بیشتر از ۱ است مقدار تمام پیکسل ها را یک نسبت افزایش میدهد و در نهایت خود تابع لگاریتم به این صورت عمل میکند که واریانس نواحی تیره را افزایش میدهد و کنتراست تصویر در نواحی تیره افزایش میابد و در نواحی روشن کنتراست و واریانس تصویر کاهش پیدا میکند.

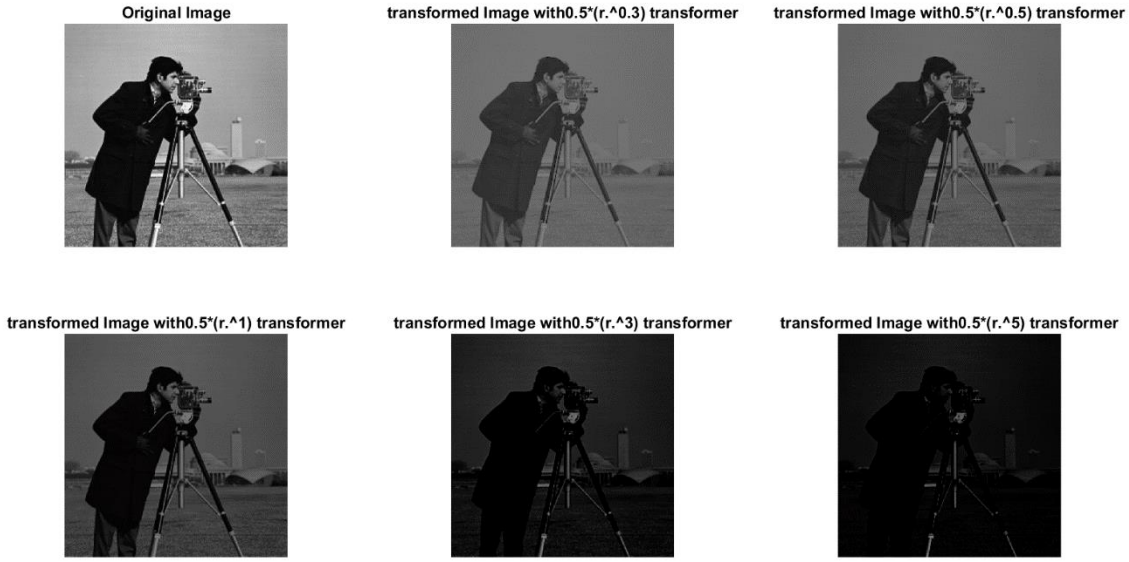


تصویر ۳۴- اعمال تابع تبدیل لگاریتم روی تصویر با مقادیر مختلف c

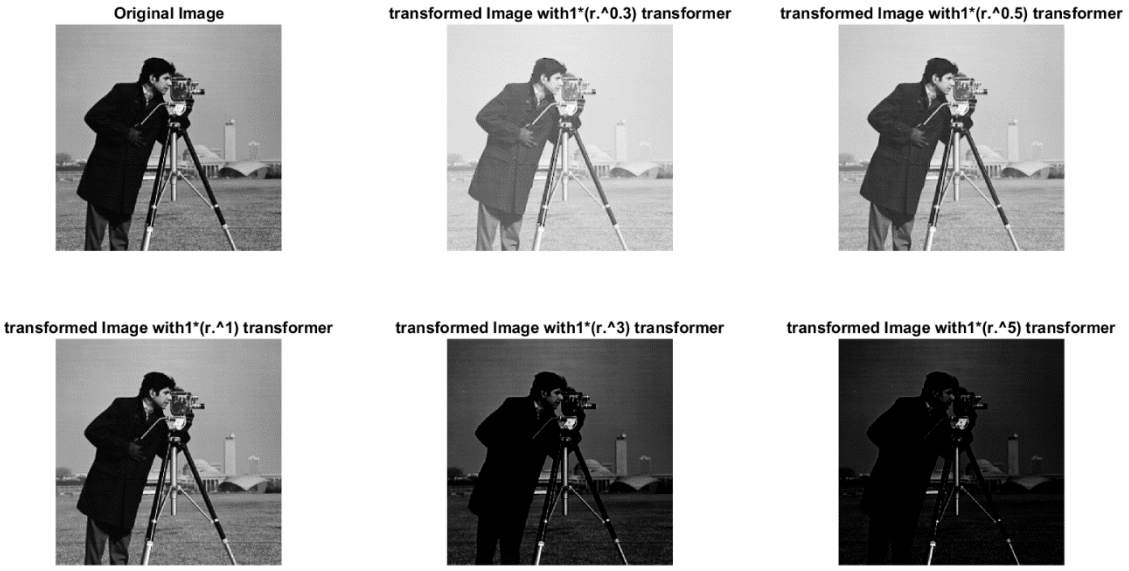
در تصویر زیر تابع معکوس لگاریتم با مقادیر مختلف c اعمال شده. در اینجا مقدار c با افزایش و کاهش مقدار پیکسل ها رابطه عکس دارد چون c در مخرج توان قرار گرفته. پس اگر مقدار c بین ۰و۱ باشد، مقادیر پیکسل ها به یک نسبت افزایش میابند و اگر مقدار c بیشتر از ۱ باشد، مقدار پیکسل ها به یک نسبت کاهش میابند.
خود تابع معکوس لگاریتم نیز برعکس تابع لگاریتم عمل میکند به این صورت که در نواحی تیره کنتراست و واریانس تصویر کاهش میابد و در نواحی روشن کنتراست و واریانس تصویر افزایش میابد.

تصویر ۳۵- اعمال تابع تبدیل معکوس لگاریتم روی تصویر با مقادیر مختلف c

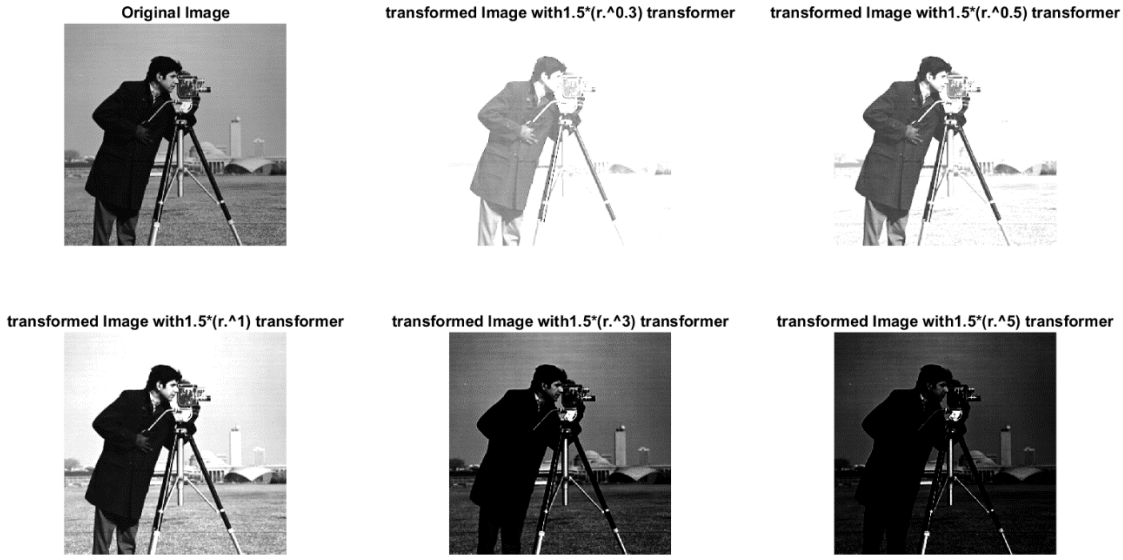
تصویر ۳۶ تا ۳۸ نیز مربوط به اعمال تابع نمایی روی تصویر است که با بررسی این تصاویر میتوانیم به تاثیر مقدار c و گاما پی ببریم.
به تصاویر گوشه پایین سمت چپ در تصاویر ۳۶ تا ۳۸ دقت کنید. در این تصاویر مقدار گاما برابر با ۱ است و مقدار c متغیر می باشد. همانطور که میبینیم مقدار c باعث میشود که مقدار پیکسل ها به یک اندازه افزایش یا کاهش یابد به طوری که اگر c در بازه ۰ تا ۱ باشد مقدار همه پیکسل ها به یک نسبت کاهش میابد و تصویر از حالت معمول تیره تر میشود و اگر c بیشتر از ۱ باشد مقدار همه پیکسل ها به یک نسبت افزایش میابد و تصویر از حالت معمول روشن تر میشود.
همچنین مقدار c اگر ۱ باشد اثر خنثی روی تصویر خواهد داشت.
برای مقدار گاما باید به یکی از تصاویر ۳۶ تا ۳۸ دقت کنیم. مثلا در تصویر ۳۶ مشاهده میکنیم که اگر مقدار گاما بین ۰و۱ باشد، تابع تبدیل نمایی مشابه تابع معکوس لگاریتم عمل میکند یعنی در نواحی تیره کنتراست و واریانس کاهش میابد و در نواحی روشن کنتراست و واریانس افزایش میابد و اگر مقدار گاما بیشتر از ۱ باشد تابع تبدیل نمایی مشابه تابع لگاریتم عمل میکند یعنی در نواحی تیره کنتراست و واریانس افزایش میابد و در نواحی روشن کنتراست و واریانس کاهش میابد.
همچنین در حالتی که c و گاما هر دو برابر با ۱ هستند ما خود تصویر اصلی را در خروجی خواهیم داشت.



تصویر ۳۶- اعمال تابع تبدیل نمایی روی تصویر با گاما های مختلف و c=0.5



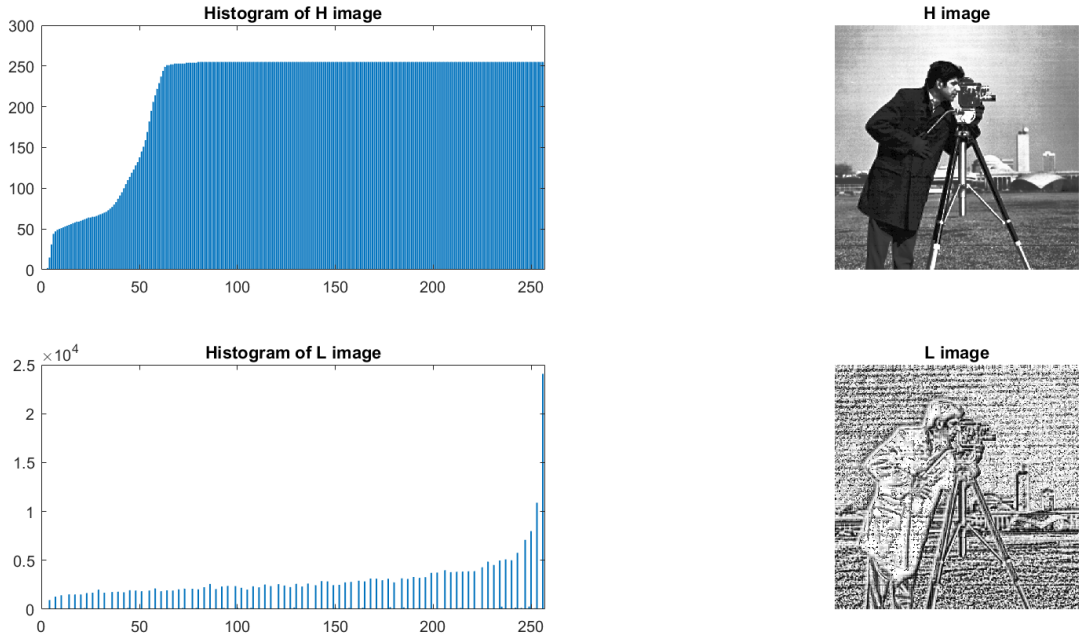
تصویر ۳۷- اعمال تابع تبدیل نمایی روی تصویر با گاما های مختلف و c=1



تصویر ۳۸- اعمال تابع تبدیل نمایی روی تصویر با گاما های مختلف و c=1.5

متعادل سازی محلی و سراسری:

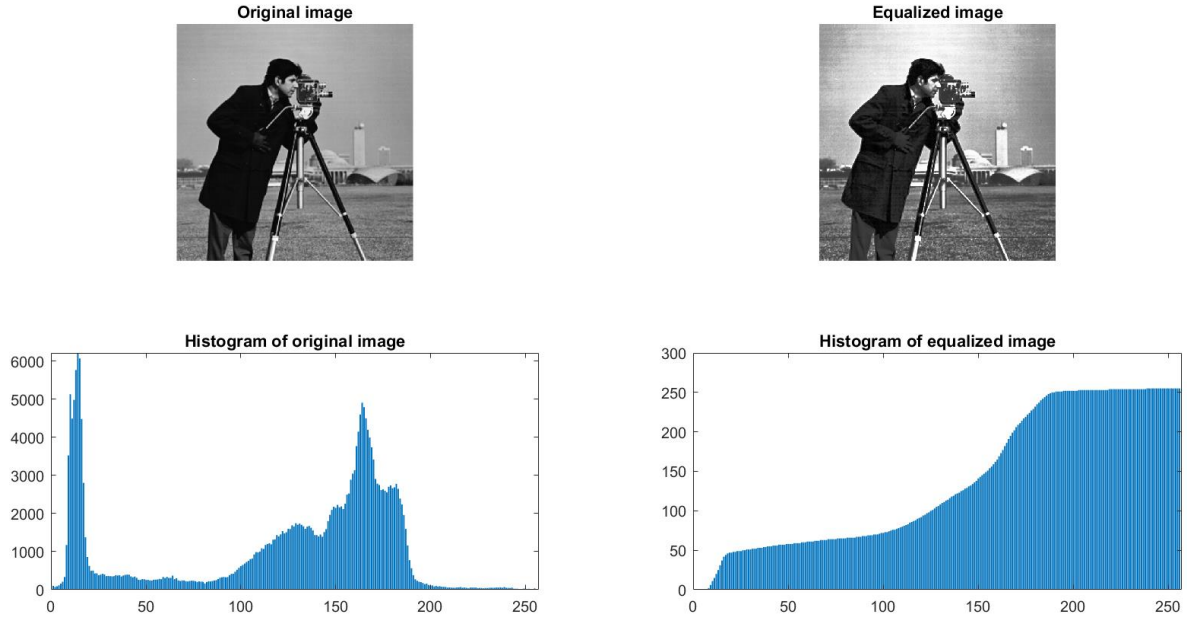
در تصویر زیر شما خروجی متعادل سازی محلی و سراسری را مشاهده میکنید. ما تصویر D را که روشنایی آن یک سوم تصویر Camera Man اصلی می باشد، که آن را در تصویر ۳۳ مشاهده میکنید، را به صورت سراسری و محلی متعادل سازی کرده ایم. خروجی متعادل شده سراسری تصویر D را H و خروجی متعادل شده محلی تصویرD را L نامیده ایم. همانطور که مشاهده میکنید به نظر میرسد که در تصویری که متعادل سازی محلی شده جزئیاتی از تصویر D قابل مشاهده است که در روش متعادل سازی سراسری قابل مشاهده نیست یا به سختی قابل مشاهده است و از طرفی لبه های تصویر در حالتی که به صورت محلی متعادل سازی شده نمایان تر می باشد. در روش متعادل سازی محلی کنتراست تصویر افزایش یافته و کیفیت تصویر به لحاظ بصری بهبود یافته اما در متعادل سازی محلی بیشتر به لبه ها و جزئیاتی که به راحتی در تصویر D قابل مشاهده نبوده اند توجه شده و آن ها را نمایان کرده است. مثلا در تصویر D انگشت های Camera Man به سختی قابل مشاهده است و در تصویر H تا حدودی قابل مشاهده است اما در تصویر L به وضوح لبه های دست فرد قابل مشاهده می باشد.



تصویر ۳۹- متعادل سازی سراسری و محلی تصویر D که روشنایی آن نسبت به تصویر اصلی یک سوم می باشد.

۳-۲-۱- بخش دوم تمرین

در تصویر زیر شما تصویر equalize شده ی Camera Man به همراه نمودار هیستوگرام آن را مشاهده میکنید. همانطور که میبینید کنتراست تصویر افزایش یافته اما همچنان نویز هایی در تصویر میبینیم که ناشی از این است که متعادل سازی به صورت سراسری انجام شده است.



تصویر ۴۰- خروجی متعادل سازی شده تصویر Camera Man

۳-۲-۳- بخش سوم تمرین

در اینجا خروجی تابع histeq و imadjust را مشاهده میکنید که به دلایلی که در بخش قبل گفتیم خروجی تابع imadjust طبیعی تر به نظر میرسد.

تصویر ۴۱- خروجی توابع histeq() و imadjust() روی تصویر Camera Man

۴- پیوست

۴-۱- تمرین Image Fundamentals

۴-۱-۱- کد بخش اول تمرین

```
A = imread("E:\Dars\Masters\digital image processing\Homeworks\Images\1\Elaine.bmp");
Levels = [4 8 16 32 64 128];
for i=1:size(Levels,2)
   %resize figure to 1000*600 pixel
   fig = figure;
   width=1000;
   height=600;
   set(fig,'position',[0,0,width,height])
   %quantize image
   image_copy = double(A);
   fm = image_copy - min(image_copy(:));
   quantized_image = floor((Levels(i)-1) * (fm/max(fm(:))));
   % we want to show quantized image so we should convert [0:3] value range to [0:255] range
   a = quantized_image/max(quantized_image(:));
   b = uint8(255 * (a));
   %d = unique(c(:));
   % after quantization we equalize image with histeq and put the result in equalized_image variable
   equalized_image = histeq(b);
   % compute mse for equalized and non equalized images
   mse_without_eq = immse(A, b);
   mse_with_eq = immse(A, equalized_image);
   % here we show the result in figure
   subplot(2,2,1), imshow(b);
   title(strcat(int2str(Levels(i)),' Levels without equalization with mse=', num2str(mse_without_eq)));
   subplot(2,2,2), imshow(equalized_image);
   title(strcat(int2str(Levels(i)),' Levels with equalization with mse=', num2str(mse_with_eq)));
   subplot(2,2,3), imhist(b);
   title('histogram of nonequalized image');
   subplot(2,2,4), imhist(equalized_image);
   title('histogram of equalized image');
   %save figure and image results
   saveas(fig,strcat(int2str(Levels(i)),'Levels_result.png'));
   imwrite(b,strcat(int2str(Levels(i)),'L_noneq_img.png'));
   imwrite(equalized_image,strcat(int2str(Levels(i)),'L_eq_img.png'));
   pause(1);
```

```
end
```

۴-۱-۲- کد بخش دوم تمرین

کد فایل main:

```
A = imread("E:\Dars\Masters\digital image processing\Homeworks\Images\1\Goldhill.bmp");
%downsample without average filtering by removing one among the columns and rows
ds_without_avg = A(1:2:end,1:2:end);
% downsample with applying average filter on image
average_filter = ones(2)/4;
downsample_size = size(A)/2;
ds_with_avg = zeros(downsample_size);
for i = 2:2:size(A,1)-1
   for j = 2:2:size(A,2)-1
      temp = double(A(i-1:i,j-1:j)) .* average_filter;
      ds_with_avg(floor(i/2),floor(j/2)) = sum(temp(:));
   end
end
ds_with_avg = uint8(ds_with_avg);
imwrite(ds_without_avg,"down_sample_without_averaging.bmp");
imwrite(ds_with_avg,"down_sample_with_averaging.bmp");
% upsample with pixel replication
img = ds_without_avg;
upsampled = zeros(size(img)*2);
for i=2:2:size(img,1)-1
   for j=2:2:size(img,2)-1
      upsampled(i-1,j-1) = img(floor(i/2),floor(j/2));
      upsampled(i-1,j) = img(floor(i/2),floor(j/2));
      upsampled(i, j-1) = img(floor(i/2),floor(j/2));
      upsampled(i,j) = img(floor(i/2),floor(j/2));
   end
end
% upsample with Pixel Replication
pr_without_avg = pixelReplication(ds_without_avg);
pr_with_avg = pixelReplication(ds_with_avg);
imwrite(pr_without_avg,"pr_without_avg.bmp");
imwrite(pr_with_avg,"pr_with_avg.bmp");
% compute mse for Pixel Replication
mse_pr_without_avg = immse(A, pr_without_avg)
mse_pr_with_avg = immse(A, pr_with_avg)
% upsample with Bilinear Interpolation
bi_without_avg = bilinearInterpolation(ds_without_avg);
bi_with_avg = bilinearInterpolation(ds_with_avg);
imwrite(bi_without_avg,"bi_without_avg.bmp");
imwrite(bi_with_avg,"bi_with_avg.bmp");
% compute mse for Bilinear Interpolation
mse_bi_without_avg = immse(A, bi_without_avg)
mse_bi_with_avg = immse(A, bi_with_avg)
```

کد pixel replication:

```
function upsampled = pixelReplication(img)
upsampled = zeros(size(img)*2);
for i=2:2:size(upsampled,1)
   for j=2:2:size(upsampled,2)
      upsampled(i-1,j-1) = img(floor(i/2),floor(j/2));
      upsampled(i-1,j) = img(floor(i/2),floor(j/2));
```

```
      upsampled(i, j-1) = img(floor(i/2),floor(j/2));
      upsampled(i,j) = img(floor(i/2),floor(j/2));
   end
end
upsampled = uint8(upsampled);
```

کد bilinear interpolation:

```
function upsampled = bilinearInterpolation(img)
angle_filter = [1 0 1; 0 0 0; 1 0 1]/4;
hv_filter = [0 1 0; 1 0 1; 0 1 0]/4;
upsampled = zeros(size(img)*2);
% placing values of img in upsampled matrix in odd rows and columns
upsampled(1:2:size(upsampled,1),1:2:size(upsampled,2)) = img;
% first we add padding to img2 matrix to make interpolation easy
img2 = zeros(size(upsampled)+2);
img2(2:size(img2,1)-1,2:size(img2,2)-1) = upsampled;
for i=3:2:size(img2,1)
   for j=3:2:size(img2,2)
      % computation for center pixel: we should use angle_filter
      img_block = img2(i-1:i+1,j-1:j+1);
      mask_apply = img_block.*angle_filter;
      pixel_value = sum(mask_apply,'all');
      img2(i,j) = pixel_value;
      % computation for left and up pixels: we should use hv_filter
      indexes = [i,j-1;i-1,j;];%i,j+1; i+1,j
      for k=1:2
         img_block = img2(indexes(k,1)-1:indexes(k,1)+1,indexes(k,2)-1:indexes(k,2)+1);
         mask_apply = img_block.*hv_filter;
         pixel_value = sum(mask_apply,'all');
         img2(indexes(k,1),indexes(k,2)) = pixel_value;
      end
   end
end
upsampled = img2(2:size(img2,1)-1,2:size(img2,2)-1);
upsampled = uint8(upsampled);
```

۴-۱-۳- کد بخش سوم تمرین

```
image8bit = imread('E:\Dars\Masters\digital image processing\Homeworks\Images\1\Barbara.bmp');
image8bit = rgb2gray(image8bit);
% for 1 to 5 LSB
number = 0;
index = 1;
for bit=0:4
   number = number + 2.^bit;
   disp(number);
   decrease_bits = double(bitand(image8bit,number));
   imwrite(decrease_bits, strcat('barbara_',int2str(index),'bit_LSB.bmp'));
   figure; imshow(decrease_bits,[]), title(strcat('with  ',int2str(index),' least significant bits'));
   index = index + 1;
end

% for 1 to 5 MSB
number = 0;
```

```
index = 1;
for bit=7:-1:3
   number = number + 2.^bit;
   disp(number);
   decrease_bits = double(bitand(image8bit,number));
   imwrite(decrease_bits, strcat('barbara_',int2str(index),'bit_MSB.bmp'));
   figure; imshow(decrease_bits,[]), title(strcat('with  ',int2str(index),' most significant bits'));
   index = index + 1;
end
```

۴-۲- تمرین Contrast Adjustment

۴-۲-۱- کد بخش اول تمرین

کد فایل main:

```
img = imread('E:\Dars\Masters\digital image processing\Homeworks\Images\2\Camera Man.bmp');
% histogram of Camera Man image
hist_array = computeHist(img, 256);
fig = figure;
set(fig,'position',[0,0, 1000,300])
subplot(1,2,1), imshow(img);
subplot(1,2,2), bar(hist_array);

% histogram of Camera Man  and D image
D = uint8(img/3);
fig = figure;
set(fig,'position',[0,0, 1000,1000])
subplot(1,2,1), bar(computeHist(img,256));
title('Histogram of input image');
subplot(1,2,2), bar(computeHist(D,256));
title('Histogram of D image');

% do histogram equalization and local histogram equalization on D
[H, eq_hist_H] = histEqualization(D,256);
[L, eq_hist_L] = LocalHistEqualization(D,256,int32(size(D,1)/64));
subplot(2,2,1), bar(eq_hist_H), title('Histogram of H image');
subplot(2,2,2), imshow(H), title('H image');
subplot(2,2,3), bar(eq_hist_L), title('Histogram of L image');
subplot(2,2,4), imshow(L), title('L image');

% log transformation
fig = figure;
set(fig,'position',[0,0, 1000,1000])
r = double(img)/255; % Normalized Image
subplot(2,2,1),imshow(img),title('Original Image');
c=[0.5; 1; 2];
for index=1:3 % Constant
   f1 = c(index).*log(1 + (r)); % Log Transform
   subplot(2,2,index+1),imshow(f1),title(strcat('transformed Image with ',num2str(c(index)),'*log(1+(r))
transformer'));
end

% inverse log transformation
fig = figure;
set(fig,'position',[0,0, 1000,1000]);
```

```
subplot(2,2,1),imshow(img),title('Original Image');
L = 256;
c=[0.5; 1; 2];
for index=1:3
   f2 = (exp(r) .^ (1/c(index))) - 1;
   subplot(2,2,index+1), imshow(f2), title(strcat('transformed Image with (exp(r).\^(1/',num2str(c(index)),'))-1
transformer'));
end

% power law transformation
gamma =[0.3 0.5 1 3 5];
for c=0.5:0.5:1.5
   fig = figure;
   set(fig,'position',[0,0, 1000,1000]);
   subplot(2,3,1),imshow(img),title('Original Image');
   for index=1:5
      f3 = c * double(r).^ gamma(index);
      subplot(2,3,index+1), imshow(f3), title(strcat('transformed Image
with',num2str(c),'*(r.\^',num2str(gamma(index)),') transformer'));
   end
end
```

کد محاسبه نمودار هیستوگرام:

```
function hist_array = computeHist(img, gray_levels)
hist_array = zeros(1,gray_levels);
for i=1:size(img,1)
   for j=1:size(img,2)
      index = img(i,j)+1;
      hist_array(index) = hist_array(index) + 1;
   end
end
```

کد محاسبه histogram equalization:

```
function [eq_image, eq_histogram] = histEqualization(input_image, gray_level)
image_hist = computeHist(input_image, gray_level);
total_pixels = size(input_image,1) * size(input_image,2);
pdf = image_hist ./ total_pixels;
cdf = pdf;
for i=2:gray_level
   cdf(i) = cdf(i) + cdf(i-1);
end
eq_histogram = round(cdf * (gray_level-1));
eq_image = zeros(size(input_image));
for i=1:size(input_image,1)
   for j=1:size(input_image,2)
      eq_image(i,j) = eq_histogram(input_image(i,j)+1);
   end
end
eq_image = uint8(eq_image);
```

کد محاسبه local histogram equalization:

```
function [leq_image, leq_histogram] = LocalHistEqualization(input_image, gray_level, windowsize)
leq_image = zeros(size(input_image));
windowsize = int32(windowsize);
```

```
step_size = int32(windowsize/2);
for i=1:step_size:size(input_image,1)
   for j=1:step_size:size(input_image,2)
      row_begin = i;
      row_end = min(size(input_image,1), i + windowsize);
      col_begin = j;
      col_end = min(size(input_image,2), j + windowsize);
      img_block = input_image(row_begin:row_end, col_begin:col_end);
      [block_eq_image, block_eq_hist] = histEqualization(img_block, gray_level);
      leq_image(row_begin:row_end, col_begin:col_end) = block_eq_image;
   end
end
leq_histogram = computeHist(leq_image, gray_level);
leq_image = uint8(leq_image);
```

۴-۲-۲- کد بخش دوم تمرین

کد اعمال equalization روی تصویر Camera Man

```
fig = figure;
set(fig,'position',[0,0, 1000,1000])
[eq_image, eq_hist] = histEqualization(img,256);
subplot(2,2,1), imshow(img), title('Original image');
subplot(2,2,2), imshow(eq_image), title('Equalized image');
subplot(2,2,3), bar(computeHist(img,256)), title('Histogram of original image');
subplot(2,2,4), bar(eq_hist), title('Histogram of equalized image');
```

۴-۲-۳- کد بخش سوم تمرین

```
img = imread('E:\Dars\Masters\digital image processing\Homeworks\Images\2\Camera Man.bmp');
figure;
subplot(1,3,1);imshow(img); title('original image');
subplot(1,3,2);imshow(histeq(img,256)); title('histeq()');
subplot(1,3,3);imshow(imadjust(img)); title('imadjust()');
```