گروه كامپيوتر - دانشكده مهندسي - دانشگاه فردوسي مشهد

مستندات گزارش تمرین اول

درس مبانی بینایی کامپیوتر

**ارائه ‌دهنده:**

**صالح شیروانی**

**استاد درس:**

**دکتر امیرحسین طاهری نیا**

**پاییز1401**

چکیده

در این گزارش به پیاده سازی مباحثی که تا کنون به صورت تئوری آنها را فرا گرفته ایم می‌پردازیم. این مباحث به طور کلی شامل مباحث پایه ای پردازش تصویر دیجیتال در حوزه مکان نظیر Quantization، Interpolation، Histogram equalization و... می‌شوند. در نهایت با پیاده سازی الگوریتم های مورد نظر و دریافت خروجی به مطالعه و تفسیر نتایج به دست آمده از پیاده سازی می‌پردازیم.

Technical Description

1. **Image Fundamentals**
   1. Quantization & Interpolation
      1. در این سوال از ما خواسته شده کار های زیر را انجام دهیم:

* نمایش تصویر با تعداد بیت متفاوت برای شدت روشنایی:

برای این کار تابعی تعریف میکنیم که تصویر و تعداد بیت مورد نظر را دریافت و سپس مقادیر شدت روشنایی (gray level) تمام پیکسل ها در تصویر دریافتی را طوری نرمال سازی ‌کند که با تعداد بیت دریافت شده در ورودی تابع ممکن باشد. برای این کار از فرمول زیر استفاده می‌کنیم:

* محاسبه و نمایش هیستوگرام تصویر:

برای به دست آوردن هیستوگرام تصویر یک لیست با 256 المان در نظر میگیریم که از آن برای شمارش پیکسل ها استفاده می کنیم. روش کار به این صورت است که تمام پیکسل های تصویر را پیمایش می کنیم و بر اساس شدت روشنایی که هر پیکسل خاص دارد المان متناظر با آن شدت روشنایی را در لیست مذکور یک واحد افزایش می دهیم. پس از شمارش پیکسل ها تمام داده هایی که برای رسم هیستوگرام تصویر نیاز داریم مهیا است.

* متعادل سازی هیستوگرام (histogram equalization):

برای این کار از تابع equalizeHist که در کتابخانه cv2 در پایتون وجود دارد استفاده می‌کنیم.

توجه: برای نرمال سازی هیستوگرام تابعی نوشته شده است که چون در این سوال مجاز به استفاده از تابع equalizeHist بودیم توضیح آن در قسمت مربوط به سوال 3-1-1-2 ارائه می‌شود.

* محاسبه MSE:

برای این کار از یک فرمول ساده استفاده می‌کنیم:

ابتدا تصویر متعادل شده را از تصویر اصلی به صورت آرایه ای کم می‌کنیم و سپس آرایه ی حاصل را به توان دو می رسانیم(منظور از این قسمت این است که تک تک درایه های ماتریس به توان دو می‌رسند). در نهایت میانگین درایه های ماتریس را حساب می کنیم که این مقدار همان MSE مطلوب ماست.

* + 1. کارهایی که در این سوال باید انجام شوند به صورت زیر است:
* Downsample کردن تصویر:

برای این کار تابعی تعریف می‌کنیم که در ورودی، تصویر مورد نظر و فرکانس مطلوب را دریافت کرده و در خروجی یک آرایه که در واقع تصویر مد نظر ماست را بر می‌گرداند.

نحوه کار تابع به این صورت است که مختصات هر پیکسل را بر فرکانس تقسیم می‌کند تا مختصات آن در تصویر downsample شده به دست آید. این کار باعث می‌شود تعداد سطر ها و ستون های تصویر با نرخی که به عنوان ورودی به تابع دادیم کاهش پیدا کند.

* پیاده سازی averaging filter:

برای این کار یک تابع می‌نویسیم که تصویری که قصد داریم روی آن فیلتر اعمال کنیم را در ورودی دریافت می‌کند. سپس یک ماسک با سایز 9 یعنی یک ماسک تعریف می‌کنیم و این ماسک را در تصویری که دریافت کردیم convolve می‌کنیم. خروجی تابع تصویر فیلتر شده است.

* *Upsample کردن تصویر:*

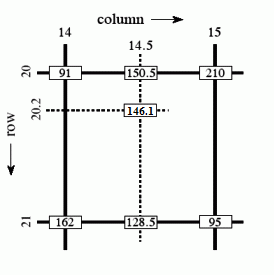
*تابعی تعریف می‌کنیم که در ورودی تصویر هدف و نرخ* upsampling *را دریافت ‌کند. برای* upsample *تمامی پیکسل های تصویر را پیمایش می‌کنیم. مختصات هر پیکسل را بر نرخی که در ورودی تابع دریافت کرده ایم تقسیم می‌کنیم تا مختصات پیکسل در تصویر upsample شده به دست آید و سپس شدت روشنایی این پیکسل را در تصویر upsample شده قرار می‌دهیم. در نهایت bilinear interpolation را روی تصویر upsample شده اعمال میکنیم و تصویر حاصل را به عنوان خروجی تابع در نظر می‌گیریم.*

* *پیاده سازی bilinear interpolation:*

*برای این کار تابعی تعریف می‌کنیم که مختصات پیکسلی که قصد داریم درونیابی کنیم و تصویری که این پیکسل به آن تعلق دارد را دریافت می‌کنیم. منطقی که این تابع پیاده می‌کند به صورت زیر است:*

*فرض می‌کنیم قصد داریم پیکسل p را درونیابی کنیم. اطراف این پیکسل چهار پیکسل قرار دارند. ما مقدار پیکسل را در دو گام محاسبه می‌کنیم.*

*گام اول) ابتدا و را درونیابی می‌کنیم تا پیکسل میانی حاصل شود. همین کار را برای و تکرار می‌کنیم تا پیکسل به دست آید. در نهایت برای محاسبه مقدار پیکسل کافیست پیکسل های و را درونیابی کنیم.*

**

* + 1. *در این سوال برای اینکه تصویر را به تصویری با تعداد بیت خواسته شده برای شدت روشنایی تبدیل کنیم از همان تابعی که در سوال 1-1-1 استفاده کردیم استفاده می کنیم. در همان قسمت توضیح دادیم که منطق این تابع چیست و از چه فرمولی استفاده می‌کند.*

1. ***Contrast Adjusment:***
   1. *Histogram Equalization:*
      1. *برای این قسمت باید تابعی طراحی کنیم که داده های لازم برای رسم هیستوگرام را به دست آورد. این تابع به تصویری که قصد داریم هیستوگرام آن را به دست آوریم دریافت می‌کند و سپس در یک آرایه 256 عنصری تعداد پیکسل های با شدت روشنایی را در اندیس شماره این آرایه قرار می‌دهد. خروجی تابع همین آرایه است که به کمک آن می‌توانیم هیستوگرام مورد نظر سوال را که دارای 256 bin است به دست آوریم.*

*2-1-1-1- برای حل این مساله کافیست تصویر مورد نظر را خوانده و شدت روشنایی هر پیکسل را بر 3 تقسیم کنیم. می‌دانیم وقتی یک تصویر grayscale را بخوانیم با یک آرایه 2 بعدی سروکار داریم. پس کافیست تمام المان های این آرایه را بر 3 تقسیم کنیم.*

*2-1-1-2- برای این قسمت باید هیستوگرام دو تصویر را به دست آوریم که توضیحات لازم در 2-1-1 ارائه شد.*

*2-1-1-3- برای این قسمت کارهای زیر را انجام می‌دهیم تا Histogram Equalization را پیاده سازی کنیم:*

* *محاسبه PDF:*

*برای این کار تابعی می‌نویسیم که آرایه ای که تعداد پیکسل های هریک از 256 bin هیستوگرام را دراختیار دارد به همراه سایز تصویر دریافت می کند و در نهایت تمام المان های آرایه را بر سایز تصویر تقسیم کرده و آرایه حاصل را خروجی می‌دهد.*

* *محاسبه CDF:*

*تابعی می‌نویسیم که آرایه حاصل از قسمت قبل (محاسبه PDF) را دریافت می‌کند و CDF را مطابق فرمول زیر محاسبه می کند:*

*خروجی این تابع یک آرایه است.*

* *در نهایت با ضرب آرایه حاصل از قسمت قبل که را برای ما حساب کرد در 255 مقادیر لازم برای محاسبه هیستوگرام equalize شده را به دست می‌آوریم.*

*2-1-1-4- برای این سوال گام‌های زیر را دنبال می‌کنیم:*

* *تابعی برای افزودن zero padding به تصویر هدف:*

*این تابع تصویر هدف و یک عدد که نشان دهنده تعداد سطر و ستونی های حاوی صفر است که باید به اطراف تصویر اضافه شوند.*

* *تابعی برای equalize کردن پنجره:*

*این تابع یک پنجره مربعی از تصویر مد نظر را دریافت می‌کند و روی درایه های این پنجره پیمایش کرده و احتمال وجود هر gray level در این پنجره را محاسبه کرده و سپس با ضرب این اعداد در 255 آن‌ها را به عددی در بازه ی [0-255] تبدیل می‌کند. در نهایت مقدار جدید عنصر مرکزی پنجره را مشخص کرده و به عنوان خروجی تابع برمی‌گرداند.*

* *در نهایت تابعی می‌نویسیم که تصویر و طول پنجره مدنظر را دریافت می‌کند و با استفاده از دو تابع قبل تصویر را equalize می‌کند و پس از این کار تصویر را به سایز اولیه خود برمی‌گرداند. خروجی این تابع تصویر equalize شده است.*

*2-1-1-5- global histogram equalization کنتراست کل تصویر را افزایش می‌دهد در حالی که local histogram equalization با در نظر گرفتن پنجره ای با سایز مشخص کنتراست تصویر را به صورت محلی افزایش می‌دهد که جزئیات بیشتری از تصویر را مشخص می‌کند.*

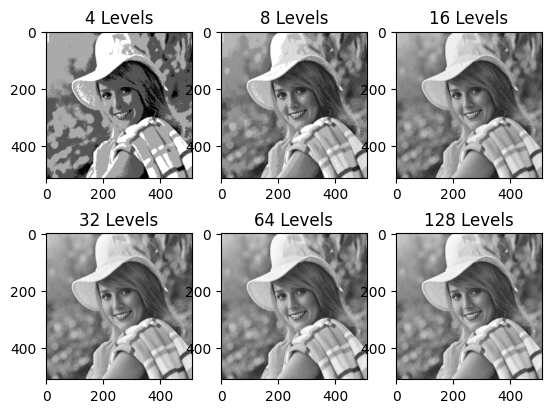
*2-1-1-6- برای این سوال توابعی به صورت زیر تعریف می‌کنیم:*

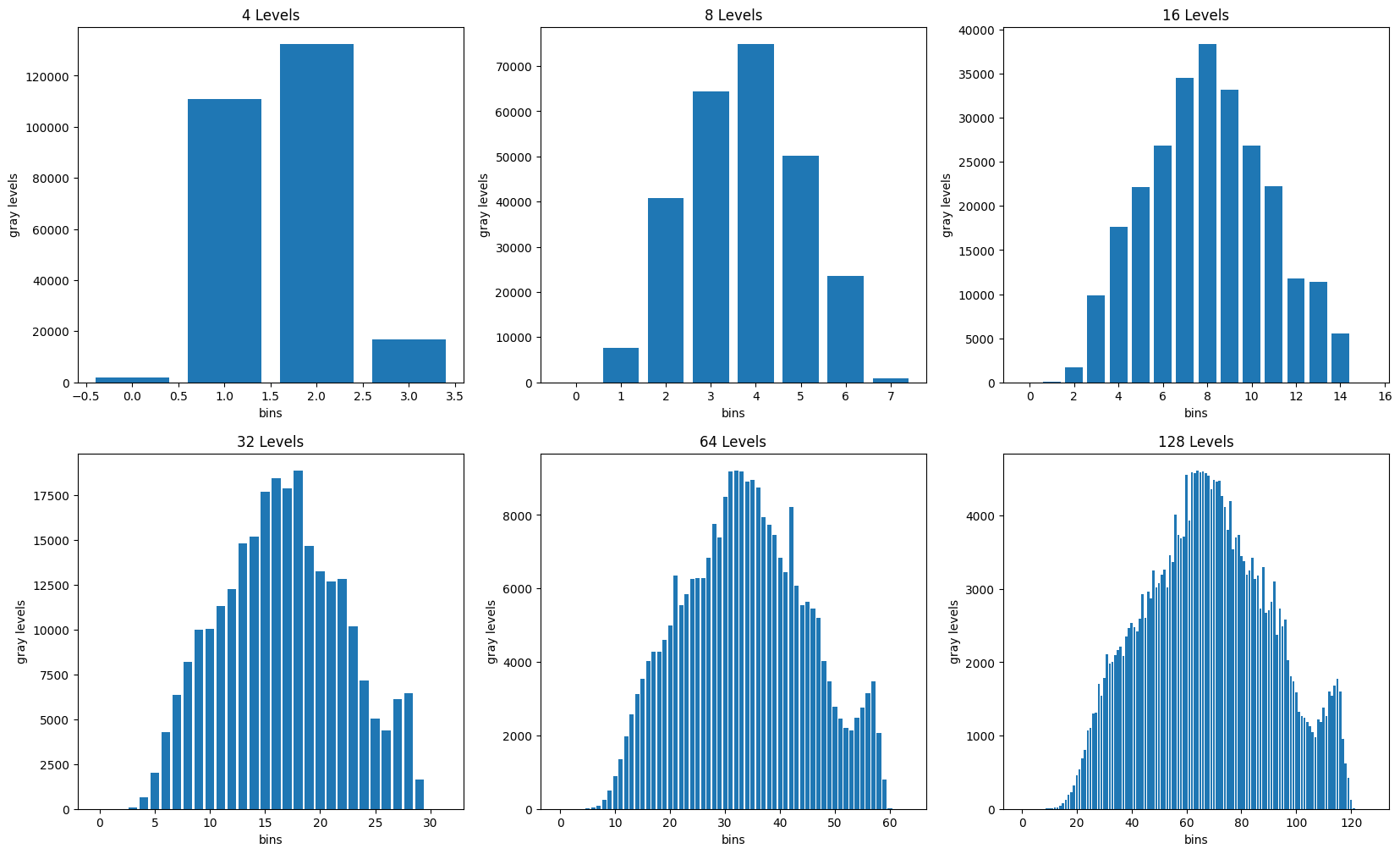
* *تابعی برای اعمال log transform: این تابع تصویر هدف و پارامتر c را دریافت می‌کند و سپس مقدار جدید gray level هر پیکسل را مطابق فرمول زیر محاسبه می‌کند:*
* تابعی برای اعمال *inverse log transform:* این تابع نیز مانند تابع قبل است ولی از فرمول زیر استفاده می‌کند:
* تابعی برای اعمال *power-law transform*: این تابع از فرمول زیر استفاده می‌کند:
  1. Local Histogram Equalization
     1. توضیحات مربوط به این سوال در قسمت 2-1-1-4 ذکر شد.

**Results**

1. **Image Fundamental**
   1. Quantization & Interpolation

* تصاویر *quantize* شده به صورت زیر است:

****

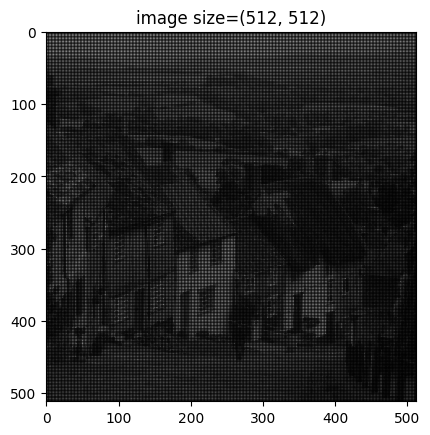
* هیستوگرام تصویر های quantize شده به صورت زیر است:
* ****
  + 1. تصویر *downsample* شده بدون *averaging filter:*

**

* تصویر *downsample* شده با استفاده از *averaging filter*:



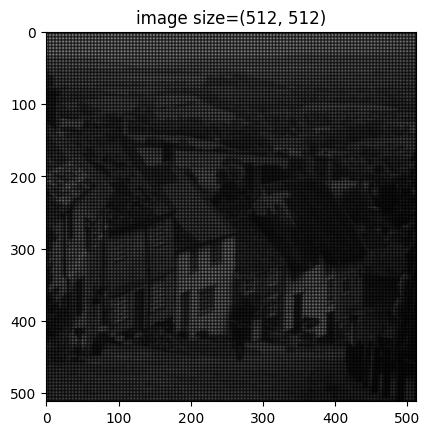
* تصویر upsample شده تصویر downsample شده بدون averaging filter:
* Bilinear interpolation:



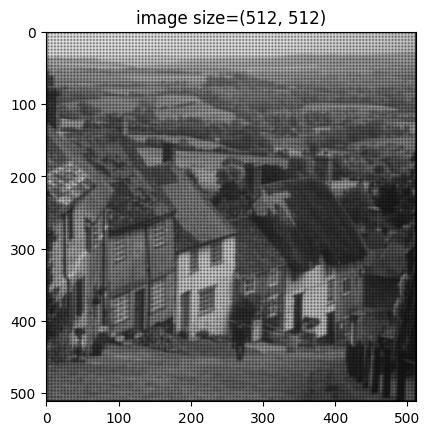
* Nearest neighbor:



* تصویر upsample شده تصویر downsample شده با استفاده از averaging filter:
* Bilinear Interpolation:

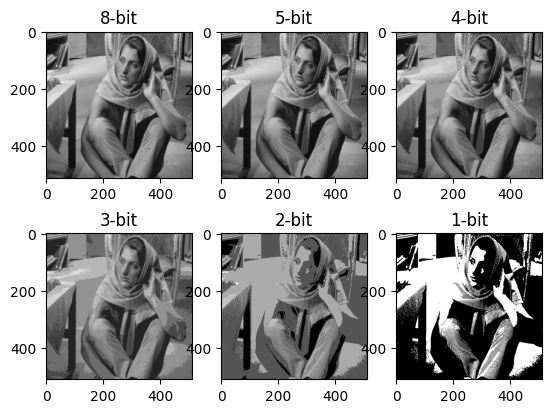


* Nearest Neighbor:



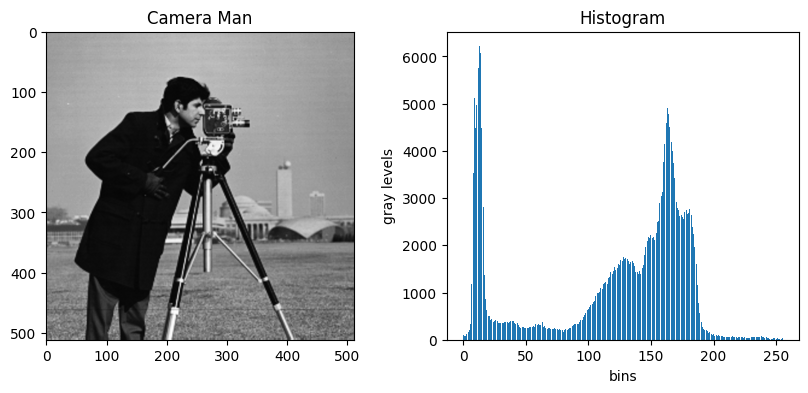
|  |  |  |
| --- | --- | --- |
| Bilinear interpolation | Pixel replication |  |
| 11329.0136 | 4005.0396 | Averaging |
| 11253.3671 | 3832.0093 | Remove row & column |

* + 1. تصویر Barbara با تعداد متفاوت بیت برای هر پیکسل به صورت زیر است:



با توجه به تصویر بالا به نظر می‌رسد 4 بیت برای هر پیکسل کیفیت خوبی دارد و می‌تواند راضی کننده باشد.

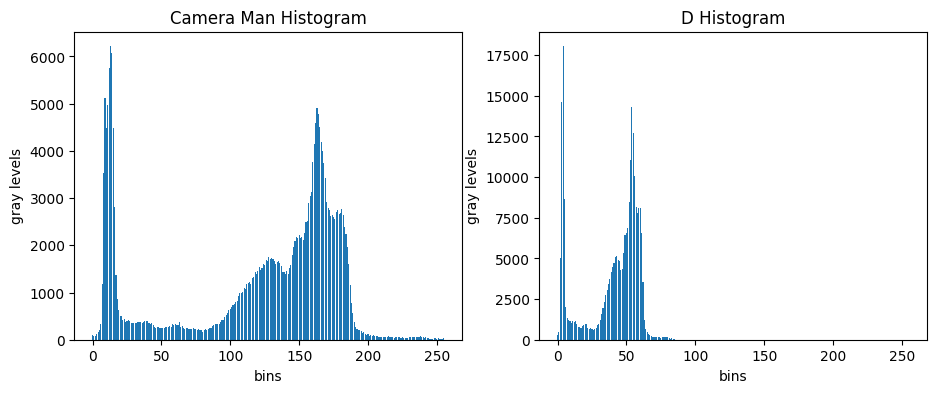
1. **Contrast Adjustment**
   1. Histogram Equalization



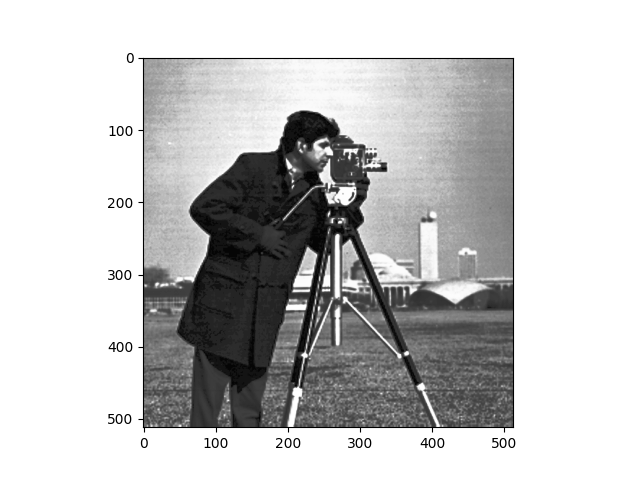
2-1-1-1- در تصویر زیر روشنایی تصویر camera man کاهش یافته است. روش این کار تقسیم تمام gray level ها بر 3 است. این تصویر همان تصویر D مد نظر سوال است.



2-1-1-2- با توجه به هیستوگرام ها واضح است که در هیستوگرام تصویر تاریک تعداد پیکسل های با gray level کمتر بسیار بیشتر شده که طبیعی هم هست. بنابراین هرچه تراکم در سمت چپ هیستوگرام بیشتر باشد به این معنی است که تعداد زیادی از پیکسل ها دارای gray level کمی هستند و تصویر تاریک تر است.



2-1-1-3- تصویر D پس از اعمال histogram equalization به صورت زیر است. این همان تصویر H است



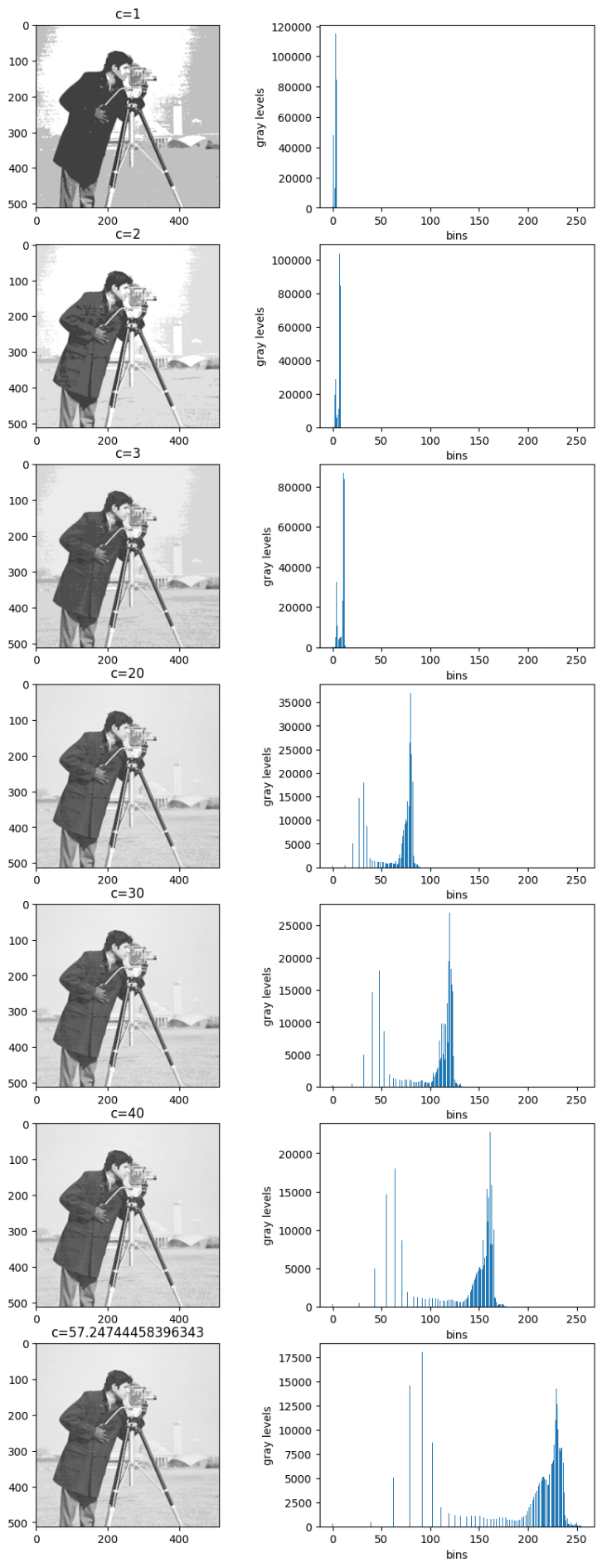
2-1-1-4- تصویر زیر حاصل اعمال local histogram equalization با اندازه پنجره 9 روی تصویر D است. این همان تصویر L می‌باشد.



2-1-1-5- *global histogram equalization کنتراست کل تصویر را افزایش می‌دهد در حالی که local histogram equalization با در نظر گرفتن پنجره ای با سایز مشخص کنتراست تصویر را به صورت محلی افزایش می‌دهد که جزئیات بیشتری از تصویر را مشخص می‌کند.*



2-1-1-6- log transform:



با توجه به تصویر های بالا مشخص می‌شود که با افزایش پارامتر c واریانس افزایش می‌یابد و در نتیجه کیفیت تصویر افزایش پیدا می‌کند. یک مقدار بسیار خوب برای c که در تصویر آخر وجود دارد به صورت زیر محاسبه می‌شود:

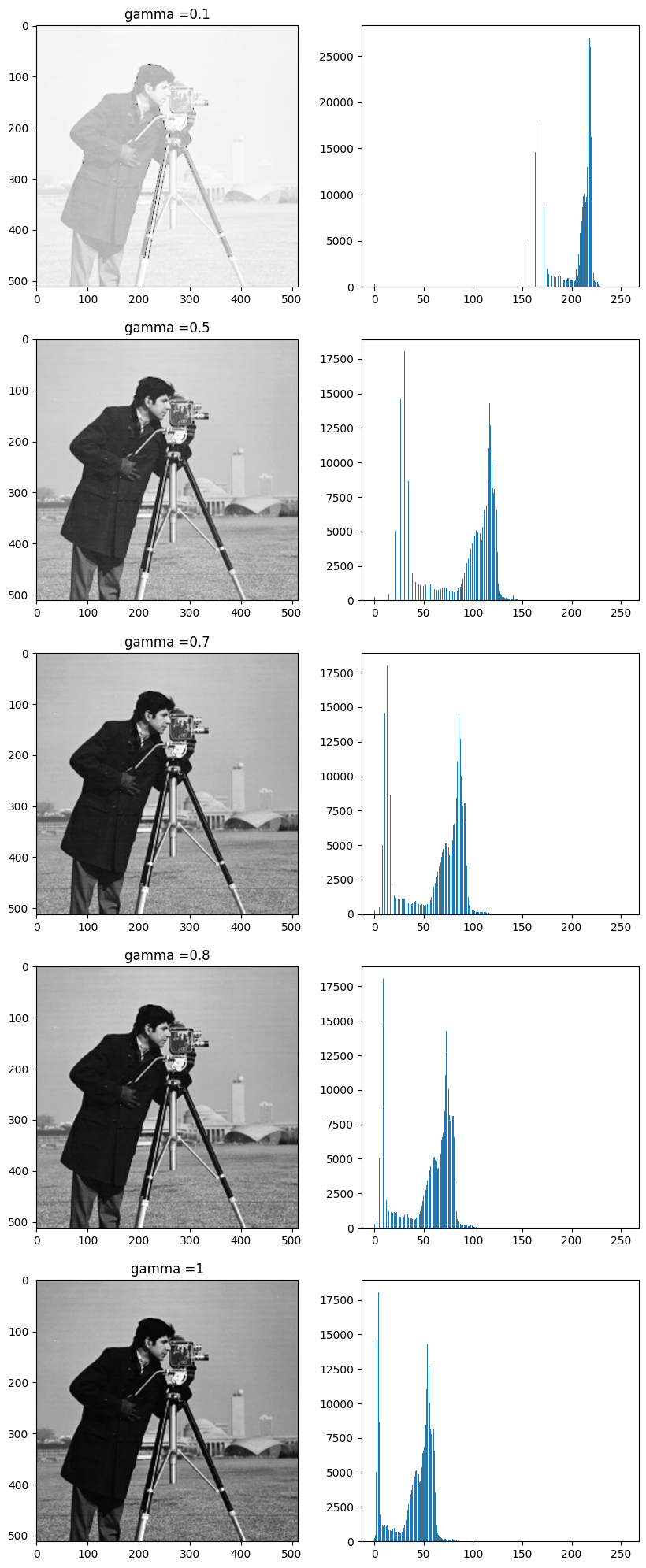
که r همان gray level های تصویر می‌باشد.

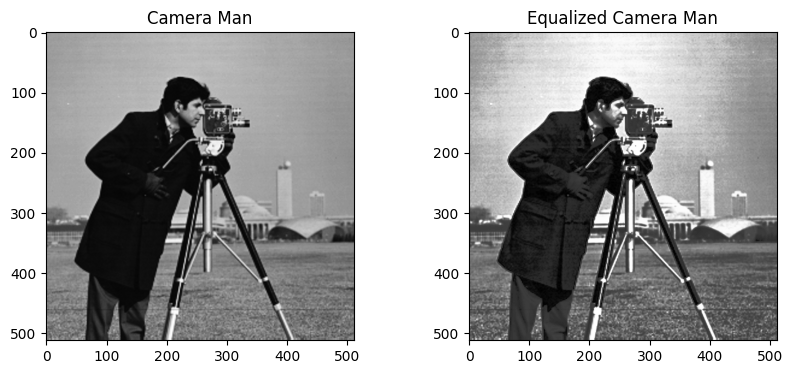
* همین نتایج برای inverse log transform نیز صادق است.



* Power-law transform:

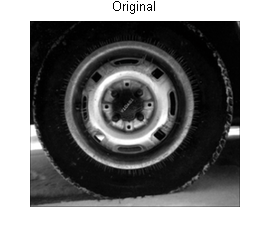
با توجه به هیستوگرام در واریانس و کنتراست تصویر بیشتر است.

**



* + 1. *هردوی این توابع برای بهبود کیفیت تصویر به کار می‌روند.* imadjust *کنتراست تصویر ورودی را با نگاشت مقادیر gray level ها به مقادیر جدید افزایش می‌دهد در حالی که histeq کنتراست تصویر را با اعمال histogram equalization روی تصویر افزایش می‌دهد.*

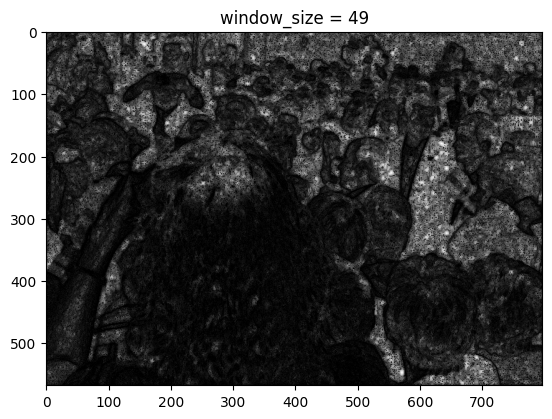
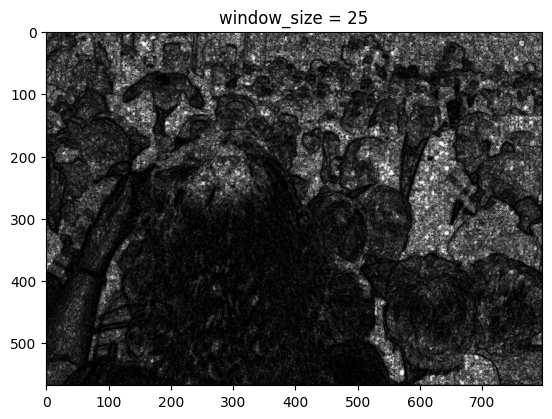
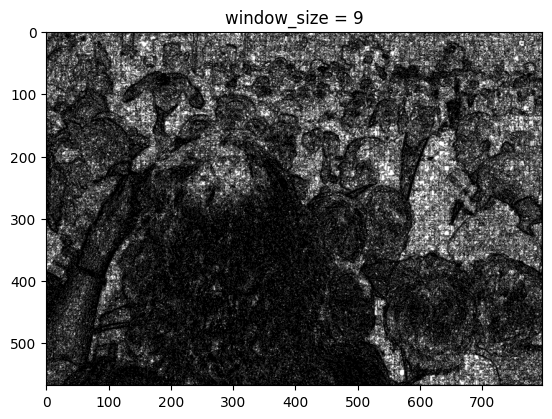
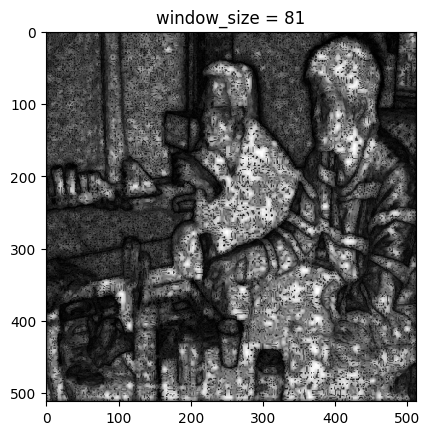
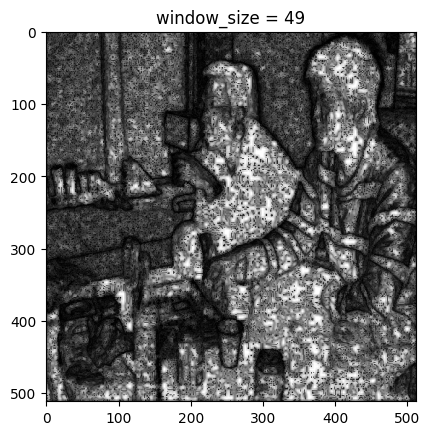
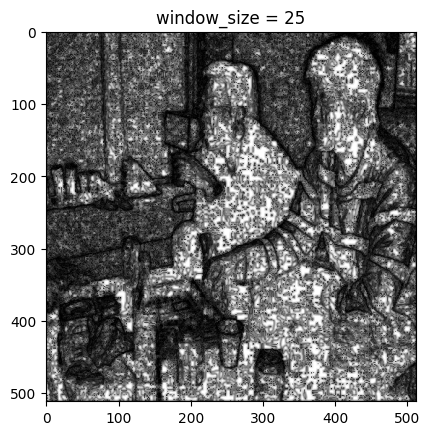
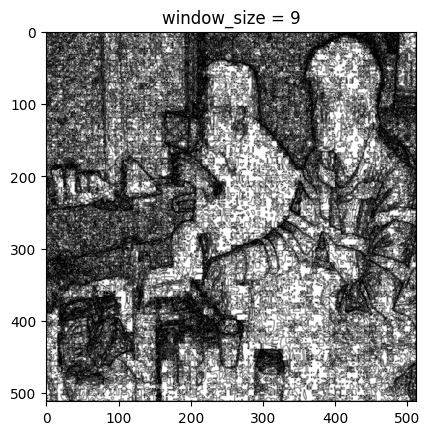
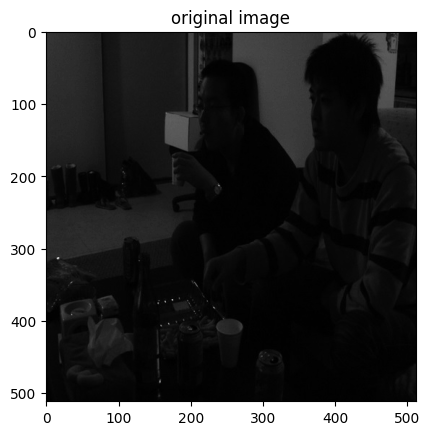
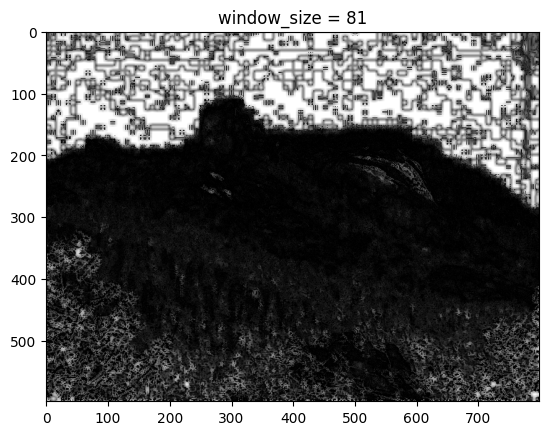
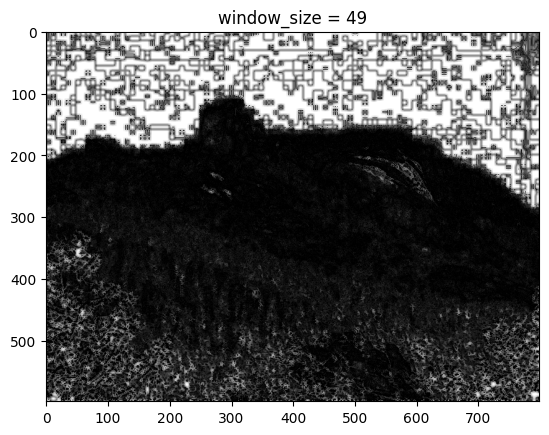
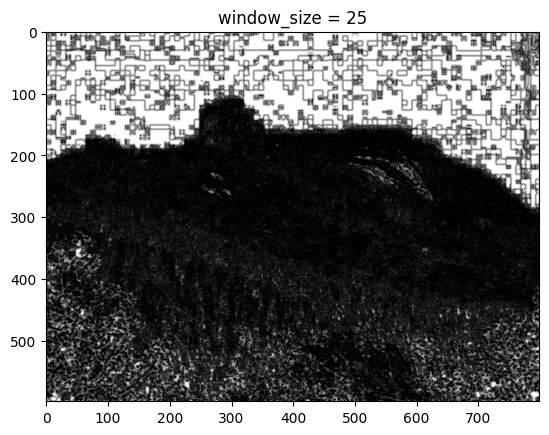
*این توابع از توابع متلب هستند بنابراین امکان تست آن ها برای ما وجود ندارد.(کد ها به زبان پایتون نوشته شده اند). تصاویر زیر برای مقایسه این توابع از اینترنت به دست آمده است:*

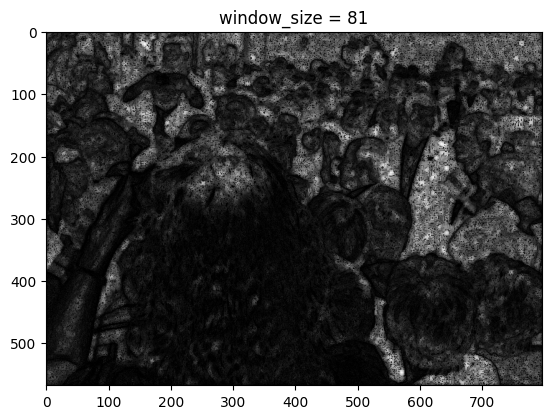
**

* 1. *Local Histogram Equalization*
     1. *هرچه سایز پنجره کوچک‌ تر باشد کیفیت تصویر خروجی بهتر است و جزئیات بیشتری ظاهر می‌شود. البته کاهش بیش از حد سایز پنجره نیز مطلوب نیست و زمان اجرای برنامه را بالا می‌برد.*

*با افزایش سایز پنجره تصویر خروجی شبیه تصویر حاصل از global histogram equalization می‌شود.*

*تصاویر خروجی به صورت زیر هستند:*

**

**

***پیوست***

* *کدها و تصاویر مربوط به این تمرین در گیتهاب از طریق این* [*لینک*](https://github.com/saleh-sh/Computer_Vision_FUM) *در دسترس است.*