

تنظیم کنتراست

سحر شیخ الاسلامی

چکیده

اطلاعات گزارش

هیستوگرام یکی از ساده‌ترین، مفیدترین، و کم حجم ترین روش‌های نمایش اطلاعات عکس می‌باشد.

با پردازش هیستوگرام عکس می‌توان به اطلاعاتی همانند توزیع سطوح خاکستری دست پیدا کرد و علاوه آن هیستوگرام ابزار مناسبی برای افزایش کنتراست عکس می‌باشد. در این گزارش با روش‌های مختلف افزایش کنتراست تصویر، همانند همسان سازی هیستوگرام سراسری و محلی آشنا می‌شویم.

تاریخ: ۲۳/۸/۱۳۹۹

واژگان کلیدی:

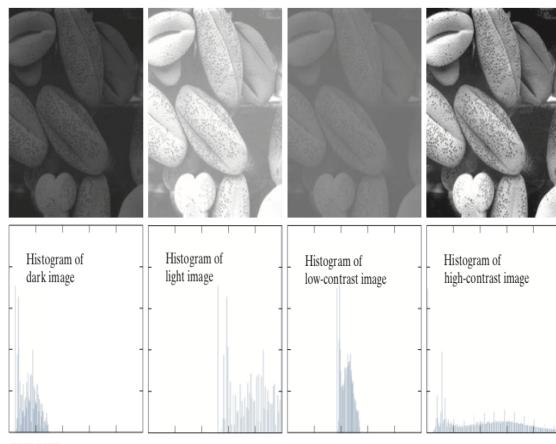
Histogram
Contrast
Histogram Processing
Image Adjustment
Histogram Equalization
AHE
CLAHE

که n_k تعداد پیکسل هایی است که شدت روشنایی r_k است.

به صورت مشابه، هیستوگرام نرمال شده به صورت زیر تعریف می‌شود.

$$P(r_k) = \frac{h(r_k)}{MN}$$

که M و N تعداد ستون و سطرهای عکس است.
محاسبه هیستوگرام ها ساده است و همچنین برای پیاده سازی بر روی سخت افزارهای ضعیف تر مناسب است، بنابراین تکنیک‌های مبتنی بر هیستوگرام، این روش را به ابزاری محبوب برای پردازش تصویر در سیستم‌های بلاذرنگ تبدیل می‌کند.



شکل ۱.۱.۱: هیستوگرام برای عکس با شدت رنگی کنتراست متفاوت

-۱ مقدمه

تصاویر داده‌های حجمی هستند و کار کردن با آنها نیازمند پردازنهای قوی و حافظه کافی می‌باشد. اما برای برخی کاربردها، نیازی نیست که آرایه‌ی دو بعدی تصویر را داشته باشیم و میتوان با استفاده از هیستوگرام عملیات مورد نظر را انجام داد. یکی از معروف ترین عملیات، افزایش کنتراست عکس است که معمولاً باعث خوانایی عکس می‌شود.

هیستوگرام حاوی اطلاعاتی درباره توزیع سطوح خاکستری است. اما درباره موقعیت مکانی پیکسل‌ها اطلاعاتی ندارد. هیستوگرام از نظر اندازه بسیار از عکس اصلی کوچکتر است و پردازش و انجام عملیات بر روی آن خیلی سریع است و نیاز به ابزارهای پیشرفته‌ای ندارد، به علاوه محاسبه آن از روی عکس اصلی ساده است، به همین دلیل به عنوان مشخصه‌ای محبوب از عکس بخش جدانشدنی در پردازش تصویر است.

۲- شرح تکنیکال

۲.۱.۱: هیستوگرام نمایشی از توزیع داده‌های عددی است.

هیستوگرام تصویر نیز، نوعی هیستوگرام است که به عنوان نمایش گرافیکی توزیع سطح رنگی در یک تصویر دیجیتال عمل می‌کند. هیستوگرام، تعداد پیکسل‌ها برای هر مقدار سطح رنگی رسم می‌کند. با مشاهده هیستوگرام برای یک تصویر خاص، یک بیننده قادر است در یک نگاه، دربارهٔ کل توزیع تصویر را نظر بدهد.

اگر بخواهیم یک تعریف ریاضی برای هیستوگرام ارائه بدهیم، به شکل زیر عمل کنیم:

برای r_k شدت رنگ پیکسل k تصویر در نظر می‌گیریم. هیستوگرام غیرنرمال به شکل زیر تعریف می‌شود.

$$h(r_k) = n_k \text{ for } k = 0, 1, 2, \dots, L-1$$

از نکات کلیدی این روش این است علاوه بر ساده بودن، قابل برگشت است. به این معنی که میتوان از روی هیستوگرام همسان شده، هیستوگرام اصلی را بدست آورد. نقطه ضعف این روش این است که می تواند باعث افزایش نویز در پیش زمینه شود در حالیکه اطلاعات مفید را از بین می برد.

همسان سازی باعث می شود که عکس غیر واقعی به نظر برسد و برای کاربردهای علمی مانند عکس برداری پژوهشکار، عکس برداری ماهواره ای بسیار ابزار قوی است. برای انجام همسان سازی هیستوگرام، به شکل زیر عمل می کنیم:

۱. هیستوگرام عکس را بدست می آوریم.
۲. هر کدام از مقادیر هیستوگرام را بر سایز تصویر تقسیم می کنیم. با این کار در واقع

۱. هیستوگرام عکس را بدست می آوریم.
۲. هر کدام از مقادیر هیستوگرام را بر سایز تصویر تقسیم می کنیم. با این تابع احتمال اینکه یک پیکسل برابر مقدار داده شده باشد را بدست می آوریم.

۳. هر کدام از مقادیر بدست آمده از مرحله قبل را برابر مجموع همه مقادیر با سطح خاکستری کمتر قرار می دهیم. با انجام این کار در واقع

(probability density function) pdf را بدست می آوریم. این تابع احتمال اینکه یک پیکسل برابر مقدار

فراآنی تجمعی را برای یک پیکسل محاسبه می کند.

۴. حال همه مقدار بدست آمده از مرحله قبل را در بیشترین سطح خاکستری (که در اینجا ۲۵۵ است) ضرب می کنیم. آرایه به دست آمده در واقع تابعی هست که شماره اندیس آن ورودی و خروجی سطح خاکستری در تصویر همسان شده را نشان می دهد.

با استفاده از الگوریتمی که در بالا توضیح داده شد، می توان همسان شده هیستوگرام را بدست آورد. علاوه بر آن با توجه به آنچه توضیح داده شد، تابعی داریم که به ازای هر سطح خاکستری، مقدار آن را در تصویر همسان شده به ما میدهد، پس تبدیل عکس نیز ممکن و ساده است.

۲.۱.۳: قسمت

درباره همسان سازی سراسری هیستوگرام در قسمت قبل به تفصیل توضیح دادیم. از آنجایی که این روش، روشی بسیار پرکاربرد در پردازش تصاویر دیجیتال است و متلب نیز یکی از محبوبترین ابزارهای پردازش تصویر می باشد، این روش توسط خود متلب پیاده سازی شده است.

این پیاده سازی، علاوه بر همسان سازی سراسری ویژگی های دیگری دارد که در ادامه توضیح می دهیم.

- همسان سازی سراسری: همسان سازی سراسری را انجام می دهد و سعی میکند که هیستوگرام عکس داده شده را صاف کند که به معنی توزیع یکنواخت است.

- همسان سازی سراسری با گرفتن نوع عکس: در این قسمت تابع علاوه بر عکس یک آرایه عددی می گیرد که مشخص می کند مقادیر عکس ورودی در چه بازه ای هستند، مثلا 8uint هستند

شکل هیستوگرام به شکل ظاهری تصویر مربوط می شود. به طور مثال در شکل ۱.۱.۱ هیستوگرام چهار عکس با شدت تیرگی و روشنی و کنتراست متفاوت را نشان می دهد.

همانطور که در شکل مشاهده شده در هیستوگرام تصویر تیره، مقادیر اولیه تکرار بیشتری دارند و در تصویر روشن، مقادیر انتهایی هیستوگرام تکرار بیشتری دارند. در تصویر با کنتراست بیشتر، توزیع مقادیر یکدست تر انجام شده است، در حالیکه در تصویر با کنتراست کمتر هیستوگرام دارای تجمع زیادی در بخشی از هیستوگرام است.

مشابه آنچه در قسمت ریاضی مفهوم هیستوگرام توضیح داده شد، برای بدست آوردن هیستوگرام به شکل زیر عمل می کنیم:

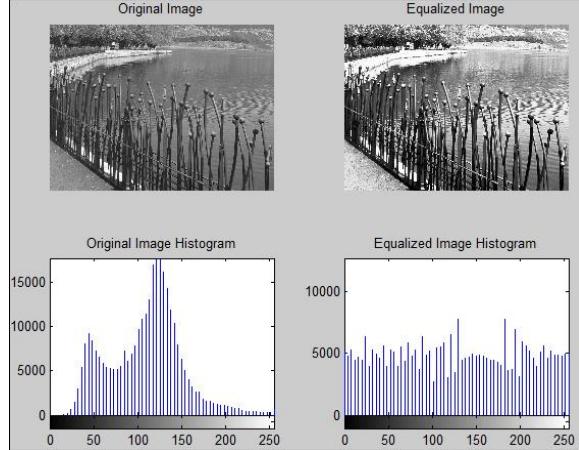
۱. به ازای همه ی پیکسل های عکس عملیات زیر را انجام بدده:

۱.۱ مقدار $[r_k]$ را یک عدد افزایش بده.

۲.۱.۲: قسمت

همسان سازی هیستوگرام روشی پرکاربرد در پردازش تصویر است که با استفاده هیستوگرام کنتراست را تنظیم می کند. این روش عموماً کنتراست کلی تصویر را افزایش می دهد، به خصوص اگر در عکس از سطوح خاکستری نزدیک به هم استفاده شده باشد.

با استفاده از این روش، سطوح خاکستری توزیع بهتری در هیستوگرام خواهند داشت.



شکل ۲.۱.۱: عکس و هیستوگرام بعد و قبل از همسان سازی هیستوگرام

این روش اجازه می دهد که مکان هایی که کنتراست کمتر دارند دارای کنتراست بیشتری شوند.

این متد برای عکس هایی که پیش زمینه عکس دارای سطوح خاکستری یکسانی با جزیات عکس است، بسیار کاربردی است.

به عنوان مثال، با استفاده از این روش، می توان تصویر استخوان را در عکسبرداری X-ray واضح تر کرد.

روش مناسبی برای افزایش کنتراست محلی تصویر می‌شود و باعث تقویت لبه‌ها در هر بخش از عکس می‌شود. با این همسان‌سازی محلی باعث افزایش noise در تصویر می‌شود.

در ادامه چند خاصیت همسان‌سازی محلی را بررسی می‌کنیم:

در همسان‌سازی محلی، سایز همسایگی که از آن به عنوان پنجره هم یاد می‌شود از پارامترهای مهم و تاثیرگذار است. هنگامی که یک همسایگی از نظر شدت رنگ نسبتاً همگن باشد، هیستوگرام آن دارای قله بلندی می‌باشد و با همسان‌سازی آن سعی داریم سطح کوچکی از سطوح خاکستری را به سطح وسیعی تبدیل کنیم. این باعث می‌شود که یک مقدار کم نویز در تصویر اصلی، در تصویر همسان شده شدت بگیرد.

برای پیاده سازی همسان‌سازی محلی روش‌های متعددی پیشنهاد شده است.

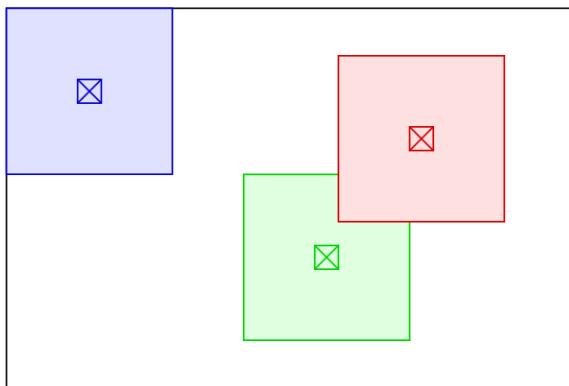
در روش اصلی، به شکل زیر عمل می‌کنیم:

۱. برای هر پیکسل عکس عملیات زیر را انجام بده:

۱.۱ برای آن پیکسل یک همسایگی انتخاب کن.

۱.۲ هیستوگرام را با توجه به بخش ۲.۱.۱ برای آن همسایگی بدست بیاور.

۱.۳ با استفاده از همسان‌سازی که در بخش بدست اوردهیم تابع تبدیل را بدست می‌آوریم و فقط مقدار همان پیکسل را عوض می‌کنیم.



عکس ۱.۴.۱: همسایگی هر پیکسل که بر اساس آن رنگ جدید را تشخیص می‌دهیم.

یکی از نکاتی که باید به آن توجه کرد، تعیین همسایگی برای نقاط روی بالایی و پایینی تصویر است. برای حل این مشکل روش‌های مختلفی ارائه شده، یکی از روش‌ها padding صفر است که به اندازه سایز پنجره به هر طرف از عکس پیکسل صفر اضافه می‌کند. اما این روش به شدت ضعیف است. روش بهتر برای انجام این mirror است. در این روش به اندازه‌ی سایز پنجره بالاترین، چپ ترین، پایینترین و راست ترین ردیف را با تکرار همان سطوح گسترش می‌دهیم.

با توجه به ماهیت همسان‌سازی، الگوریتم سریعتری از روش بالا وجود دارد. با توجه به اینکه با دانستن اینکه چند پیکسل در همسایگی مقداری کمتر یا مساوی پیکسل فعلی

و مقادیری بین ۰ تا ۲۵۵ unit هستند و مقادیر بین ۰ تا ۶۵۵۳۵ و ...

- همسان‌سازی سراسری با گرفتن تعداد bin: در این روش تابع تلاش می‌کند هیستوگرام را صاف کند، و هر چه n عدد کمتری از تعداد سطوح خاکستری عکس اصلی باشد باشد، هیستوگرام خروجی صاف تر خواهد بود.
- این تابع پیاده سازی‌های خاصی نیز برای عکس‌های رنگی دارد.

تابع دیگری که قصد بررسی آن را داریم تابع imadjust است. این تابع سطوح خاکستری و کنتراست تصویر را تغییر می‌دهد به شکلی که مقادیر جدید سطوح خاکستری، در یک درصد اولیه و نهایی (تیره ترین و روش‌ترین) باشد. به عبارت دیگه با دستکاری سطوح خاکستری، تجمع در هیستوگرام را در یک درصد ابتدایی و انتهایی هیستوگرام را افزایش می‌دهد و این کار باعث افزایش کنتراست خواهد شد.

همانند بخش قبل، این تابع نیز می‌تواند ورودی‌های دیگری بگیرد و عملیات پیچیده‌تری را انجام دهد.

- این تابع عکس را به عنوان گرفته و سطوح خاکستری را طوری تغییر می‌دهد که تجمع در یک درصد تاریک و روشن به شدت افزایش یابد.
- می‌توان علاوه بر عکس، یک بازه در ورودی داد و این تابع اعداد این بازه را به بازه $[0,1]$ تبدیل می‌کند. در واقع فقط سطوح خاکستری بین این بازه را تبدیل می‌کند و به بقیه سطوح کار ندارد.
- میتوان علاوه بر تعیین بازه در ورودی، بازه خروجی را نیز تعیین کرد. مثلاً میتوان گفت از $[l, r]$ را به $[lo, ro]$ تصویر کن.

با گرفتن یک تابع به عنوان ورودی خروجی را چنان تولید می‌کند که رابطه بین پیکسل های عکس ورودی و خروجی از آن تابع پیروی کند.

- این تابع همانند تابع قلبی، تنظیمان ویژه‌ای مخصوص عکس‌های سیاه و سفید دارد.

قسمت ۲.۲.۳:

همسان‌سازی محلی هیستوگرام، یک تکنیک در پردازش تصویر است که باعث افزایش کنتراست تصویر می‌شود.

این روش با همسان‌سازی هیستوگرام متفاوت است. همسان‌سازی هیستوگرام از یک تابع برای تمامی پیکسل‌ها استفاده می‌کند و برای تصاویری مناسب است که در که توزیع عکس در تمام عکس تقریباً یکسان است. اما در بسیاری از عکس‌ها، توزیع سطوح خاکستری در قسمت‌های مختلف تصویر، بسیار متفاوت است دارای بخش‌های خیلی تیره و یا خیلی روشن هستند و استفاده از یک تابع، تاثیر دلخواهی رو این نواحی نمی‌گذارد. در همسان‌سازی محلی، هیستوگرام‌های متعددی محاسبه می‌شود که هر کدام برای یک بخش مشخص از عکس است. و از این هیستوگرام‌ها برای تغییر سطح خاکستری تصویر استفاده می‌کنیم. برخلاف آنچه همسان‌سازی انجام می‌دهد، در همسان‌سازی محلی سطح خاکستری بر اساس همسایگی پیکسل انجام می‌شود و به همین علت این روش،

دارند، و با تقسیم عدد بدست آمده بر سایز همسایگی، فراوانی تجمعی پیکسل مرکزی بدست می آید پس نیازی به محاسبه تابع تبدیل برای کل سطوح خاکستری نیست و میتوان فقط برای سطح خاکستری پیکسل مرکزی این عملیات را انجام داد که سرعت بالاتری نسبت به الگوریتم اولیه دارد. خروجی الگوریتم زیر با پنجره های متفاوت در بخش نتایج به تفصیل توضیح داده شده است.

الگوریتم ارائه شده در بالا همچنان کند است و برای عکس های بزرگ و یا سایز پنجره های بزرگ ، سرعت محدود کننده ای دارد.

برای حل مشکل سرعت، از روش دیگری استفاده می کنند. در این روش عکس رو به صورت غیر همپوشان به پنجره هایی تقسیم کنند و پس از بدست آوردن هیستوگرام برای آن پنجره، با استفاده از الگوریتم همسان سازی، تابع را بدست می آورند و برای کل پنجره، با استفاده از همان تابع تبدیل را انجام می دهند. این روش نسبت به روش قبلی، سرعت قابل توجهی دارد اما باعث می شود خروجی مربع، مربع شود.

خروچی این بخش نیز در بخش نتایج توضیح داده شده است.

برای حل مشکل مربع مربع بودن، از پنجره های هم پوشان استفاده می کنند و میانگین وزن داری بر اساس فاصله آن از مرکز به دست آورده و این مقدار را به عنوان خروجی بر میگردانند.

اما روش های بالا همچنان مشکل بزرگ کردن نویز در یک ناحیه باشد رنگ همگن را دارند، بنابراین روش دیگری پیشنهاد شده که این مشکل را حل میکند.

از این روش به نام CLAHE(Contrast Limit Adaptive Histogram Equalization) شود. در این روش، تقویت کنتراس در مجاورت پیکسل داده شده با استفاده از شب تابع تبدیل محاسبه می شود. شب تابع تبدیل مناسب با شب تابع توزیع تجمعی محلی است و به همین دلیل با هیستوگرام را نیز تناسب دارد. در CLAHE شدت تقویت هر ناحیه از هیستوگرام را محدود می کنیم. به این معنی که بخشی از هیستوگرام را حذف کرده و سعی میکنیم روی بقیه ای هیستوگرام ، همسان سازی را انجام دهیم.

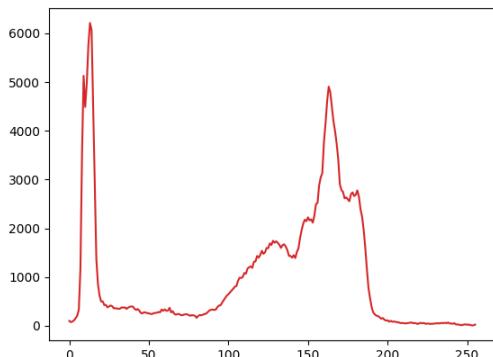
نتایج:

قسمت ۲.۱.۱:

همانطور که در قسمت شرح توضیح داده شد، با استفاده از الگوریتم هیستوگرام را بدست می آوریم.

شکل ۲.۱.۱ تصویر اصلی را با هیستوگرام نشان می دهد.

همانطور که مشاهده می شود نمودار شامل دو پیک می باشد. پیک اول در قسمت اولیه و بخش های با سطح خاکستری متواتر می باشد. همانطور که در تصویر هم مشاهده می شود لباس عکاس در عکس بخش زیادی از تصویر با تیرگی زیاد و پس زمینه نیز با سطح خاکستری متواتر می باشد.



شکل ۲.۱.۳: هیستوگرام بدست آمده از کتابخانه `opencv`

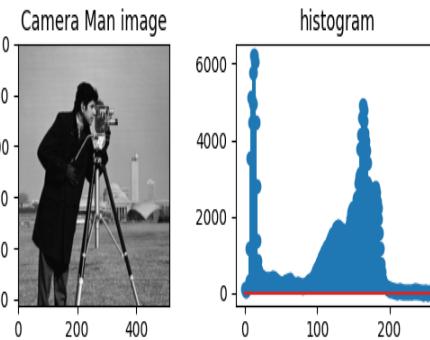
قسمت ۲.۱.۲:

در این قسمت از ما خواسته شده بود که همسان سازی هیستوگرام را روی تصویر انجام دهیم.

شکل ۲.۱.۲ تصویر اصلی و شکل ۲.۲.۲ خروجی را نشان می دهد.

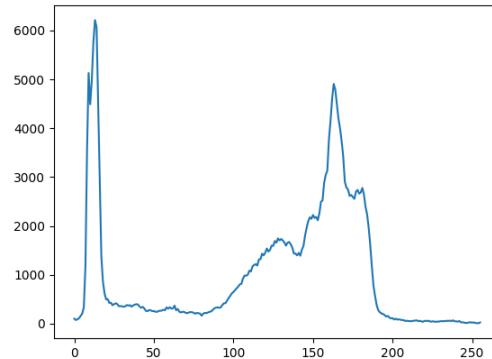


عکس ۲.۲.۱: عکس اصلی



شکل ۲.۱.۱: هیستوگرام عکس هماره عکس اصلی

در ادامه هیستوگرام بدست آمده از الگوریتم نوشته شده در شرح را با هیستوگرام نوشته شده در `opencv` مقایسه می کنیم. خروجی کاملاً یکسان است.



شکل ۲.۱.۲: هیستوگرام توسط الگوریتم در شرح

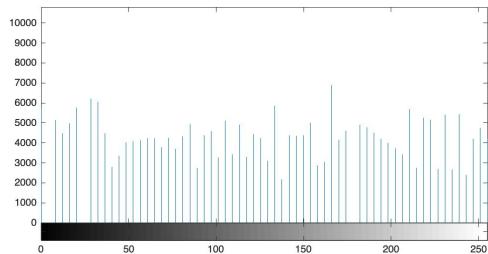
شکل ۲.۲.۴: عکس اصلی و خروجی به همراه هیستوگرام همانطور که در شکل ۲.۲.۴ مشاهده شده است هیستوگرام عکس خروجی، حالت uniform تری دارد و سعی شده همه سطوح خاکستری مقداری برابر داشته باشند. در هیستوگرام عکس خروجی مقادیر بین ۲۵ تا ۱۰۰ افزایش پیدا کرده اند و پیک ها نیز تعدیل شده اند.

قسمت ۲.۱.۳:

می خواهیمتابع های بررسی شده در قسمت شرح را با پارامتر های مختلف امتحان کنیم.
تابع: **histeq**



شکل ۲.۳.۱: تصویر بعد و قبل از اعمال همسان سازی سراسری



شکل ۲.۳.۲: هیستوگرام پس از اعمال همسان سازی سراسری

همانطور که در قسمت شرح توضیح دادیم، میتوان یک ورودی دیگر به این تابع داد که تعداد bin ها را مشخص می کند. هر چه این تعداد کمتر باشد، هیستوگرام صافتر است.

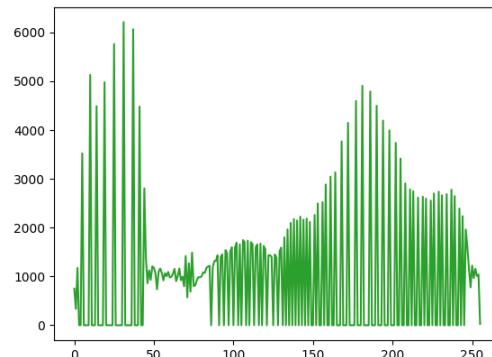


شکل ۲.۳.۴: تصویر پس از اعمال همسان سازی سراسری با bin برابر ۱۲۷

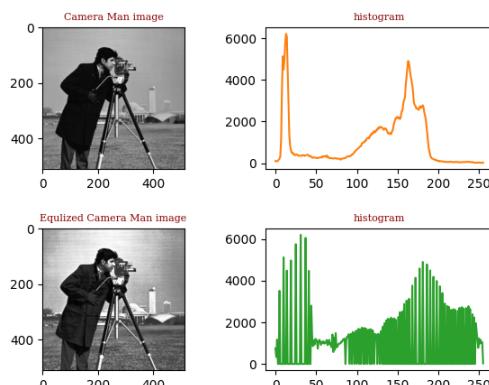


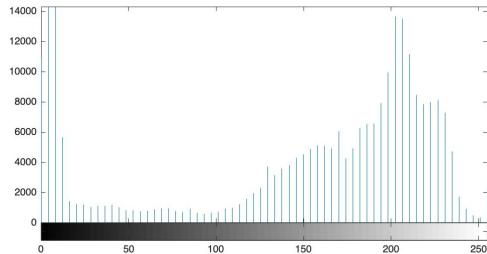
شکل ۲.۲.۲: تصویر بعد از همسان سازی هیستوگرام

همانطور که در عکس مشاهده شده است، کنترلاست تصویر افزایش یافته است. پس از همسان سازی هیستوگرام، جزییات بیشتری از عکس قبل مشاهده است. مثلاً دست عکاس در شکل ۲.۲.۲ واضحتر شده است. جیب کت عکاس در عکس اصلی قابل رویت نیست ولی پس از اعمال همسان سازی هیستوگرام، در شکل ۲.۲.۲ قابل رویت است.



شکل ۲.۳.۳: هیستوگرام پس از همسان سازی هیستوگرام



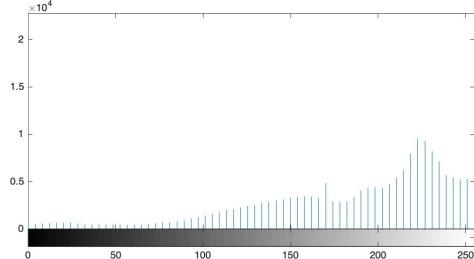


شکل ۲.۳.۹: هیستوگرام پس از اعمال تابع imadjust

با مقایسه هیستوگرام عکس اصلی (شکل ۲.۱.۲) و هیستوگرام بدست آمده، میبینیم که شاهد قله هایی در یک درصد ابتدایی و انتهایی هیستوگرام هستیم. همانطور که در بالا توضیح داده شد، این تابع را میتوان با ورودی های متنوع اجرا کرد. میتوان یک بازه از سطوح خاکستری را در ورودی به تابع بدهیم و بخواهیم این بازه به [۰,۱] تبدیل شود.



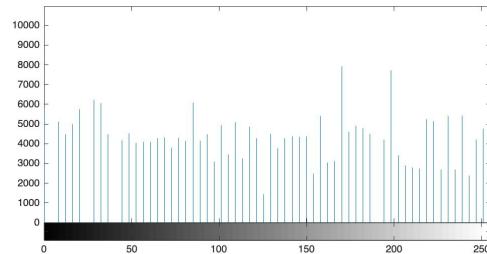
شکل ۲.۳.۱۰: تصویر پس از اعمال تابع با ورودی [۰.۰,۰.۷]



شکل ۲.۳.۱۱: هیستوگرام پس از اعمال تابع با ورودی [۰.۰,۰.۷]



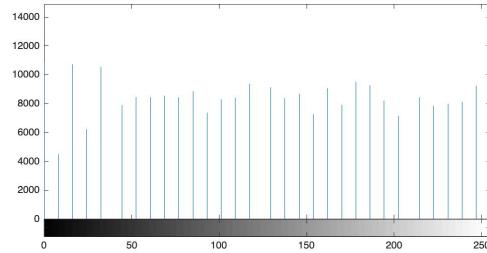
شکل ۲.۳.۱۲: تصویر پس از اعمال ورودی [۰.۰,۰.۵]



شکل ۲.۳.۵: هیستوگرام پس از اعمال همسان سازی سراسری با bin برابر ۱۲۷



شکل ۲.۳.۶: عکس پس از اعمال همسان سازی سراسری با bin برابر ۳۱

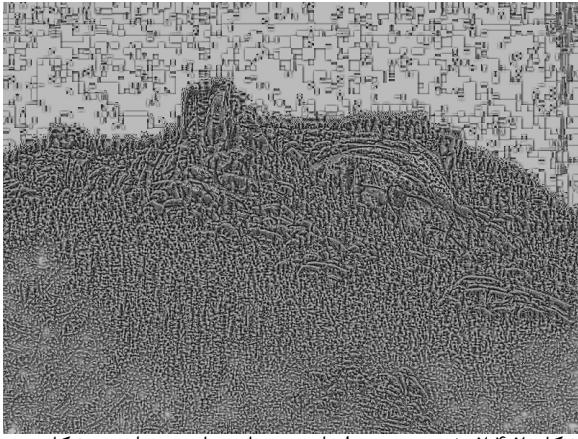


شکل ۲.۳.۷: هیستوگرام پس از اعمال همسان سازی سراسری با bin برابر ۳۱

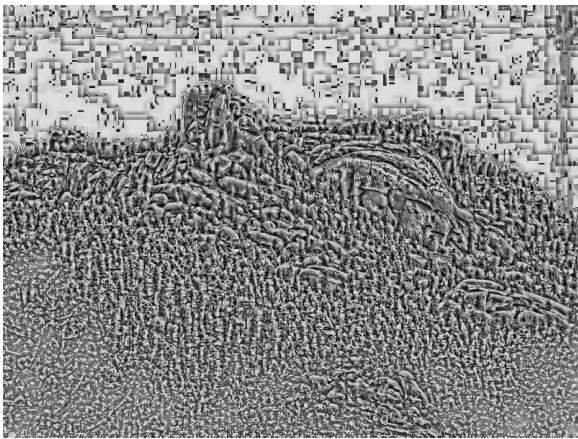
همانطور که مشاهده می شود در شکل ۲.۳.۷ نمودار صافتری نسبت به شکل ۲.۳.۵ داریم. تابع imadjust



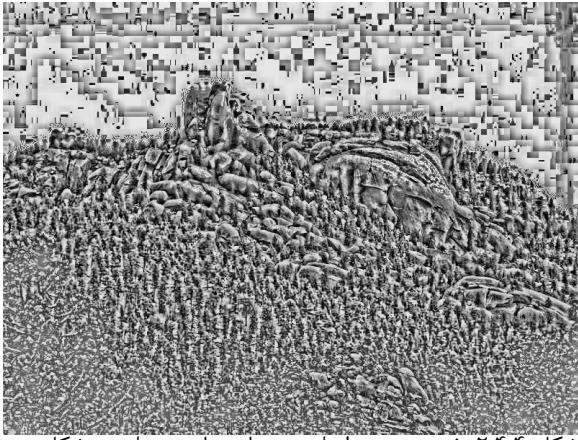
شکل ۲.۳.۸: عکس قبل و پس از اعمال تابع imadjust



شکل ۲.۴.۲: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱ سایز پنجره ۷



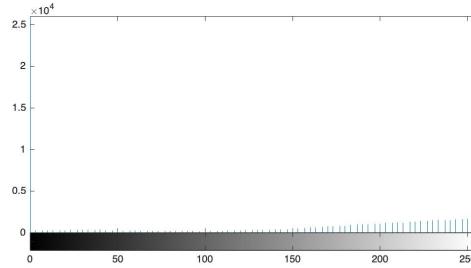
شکل ۲.۴.۳: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱ سایز پنجره ۱۵



شکل ۲.۴.۴: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱ سایز پنجره ۲۱

همانطور که مشاهده می شود با افزایش سایز پنجره، کیفیت تصویر خروجی بهتر شده و شاهد کاهش نویز هستیم.

همانطور که در شرح توضیح دادیم، در همسان سازی محلی و در همسایگی هایی که مقدار همگنی دارند، نویز بسیار بسیار زیاد است. خروجی های بالا نیز گواهی بر این حرف است.



شکل ۲.۴.۱۲: تصویر پس از اعمال ورودی [۰.۲۰۵]

همانطور که در تصاویر پیداست، وقتی بازه ی بزرگتری را از تصویر را در نظر میگیریم، جزئیات بیشتری را خواهیم داشت که کاملاً مطابق انتظار است.

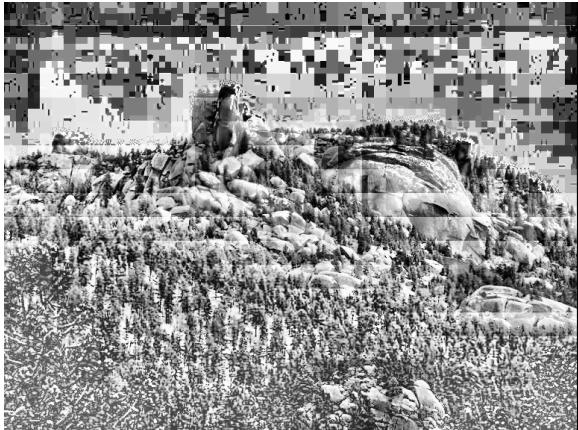
قسمت ۲.۲.۱:

همانطور که در شرح توضیح داده شد، برای اجرای همسان سازی محلی روش های متعددی وجود دارد. به دلیل اینکه بتوان مقایسه خوبی انجام داد، نتیجه اجرای الگوریتم های مختلف همسان سازی را بر هر تصویر پشت سر هم بررسی می کنیم.

در ابتدا خروجی روش اصلی، یعنی مقدار دادن پیکسل مرکزی در هر مرحله، را بررسی می کنیم. در شرح توضیح داده شد که این روش به شدت کند است و با افزایش سایز پنجره، زمان اجرا افزایش می یابد. به همین دلیل از پنجره های کوچکتری در این قسمت استفاده شده است و خروجی ها کیفیت قابل قبولی ندارند. در مرحله بعد روش سریعتر که از پنجره های غیر هم پوشان استفاده می شود را بررسی کرده و در نهایت خروجی CLAHE را بررسی می کنیم.



عکس ۲.۴.۱: عکس اصلی



شکل ۲.۴.۸: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۱ با سایز پنجره ۶۴

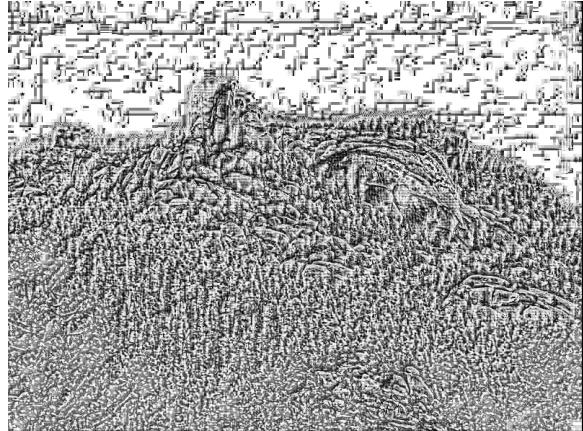


شکل ۲.۴.۹: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۱ با سایز پنجره ۱۲۸

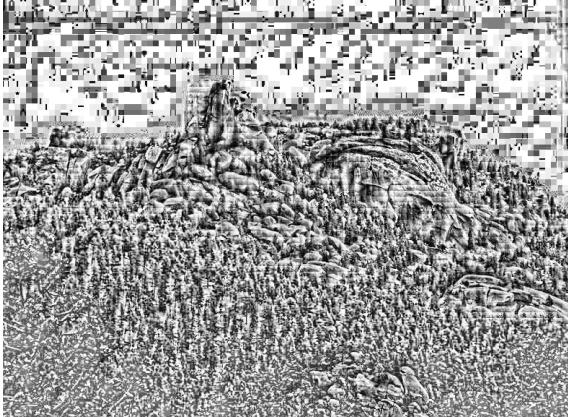
ایراد روش همسان سایزی با پنجره های غیر همپوشان، این است که باعث پنجره پنجره شدن عکس می شود. در شکل ها مشاهده میشود که با توجه به سایز پنجره، شکل به پنجره های کوچک و بزرگ تقسیم شده و کیفیت عکس مطلوب نیست. اما همچنان با افزایش سایز پنجره، به کیفیت عکس افزوده می شود.

در ادامه خروجی CLAHE را بررسی میکنیم که مشکل نویز را حل کرده است.

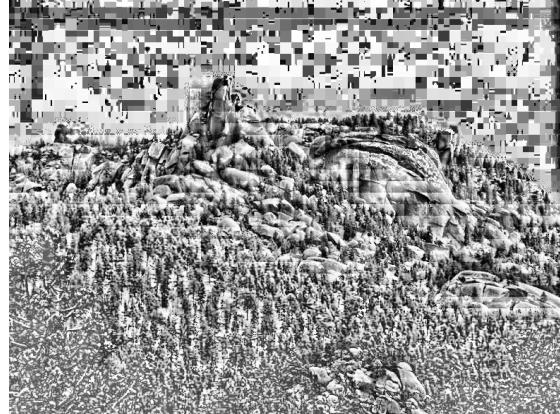
افزایش سایز پنجره، باعث کاهش همگان شدن همسایگی می شود زیرا ناحیه بزرگتری را بررسی می کنیم که امکان همگان بودن را پایین می آورد، پس احتمال تقویت نویز در پنجره های بزرگتر کمتر است. همانطور که مشاهده می شود، شکل ۲.۴.۲ کیفیت بدتری نسبت به ۲.۴.۴ و جزیبات در آن نا واضح است.



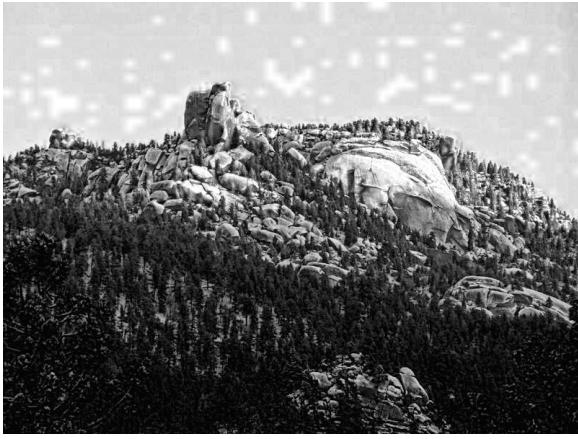
شکل ۲.۴.۵: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۱ با سایز پنجره ۸



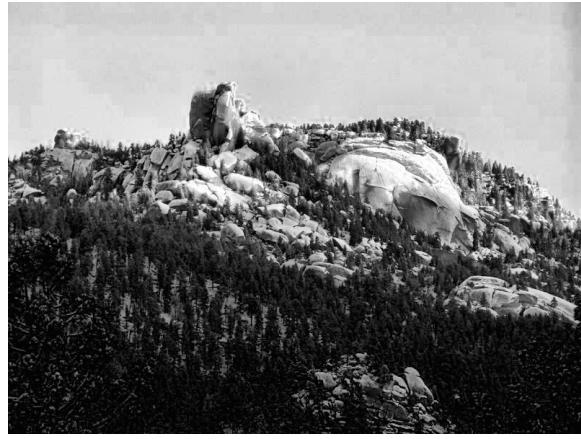
شکل ۲.۴.۶: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۱ با سایز پنجره ۱۶



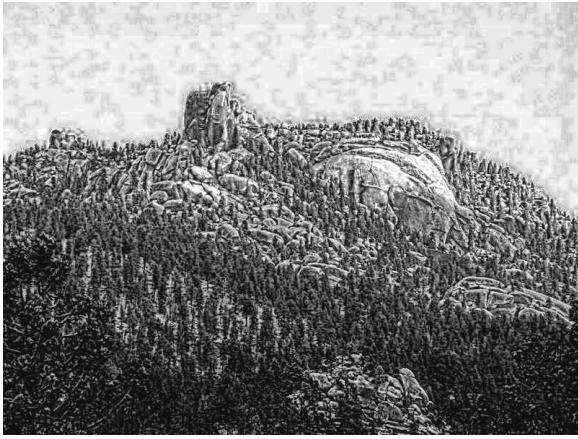
شکل ۲.۴.۷: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۱ سایز پنجره ۳۲



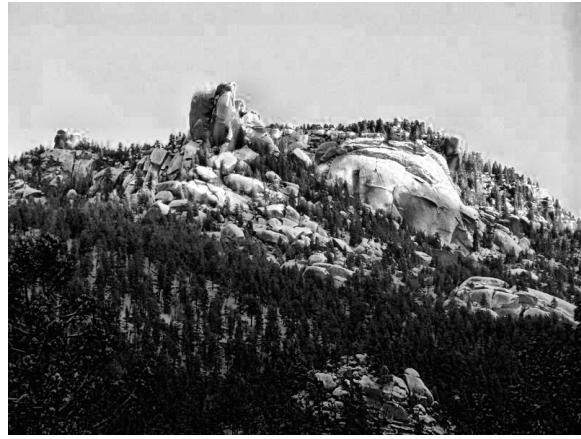
شکل ۲.۴.۱۳: خروجی متدهای CLAHE بر شکل ۲.۴.۱ سایز ۶۴ پنجره



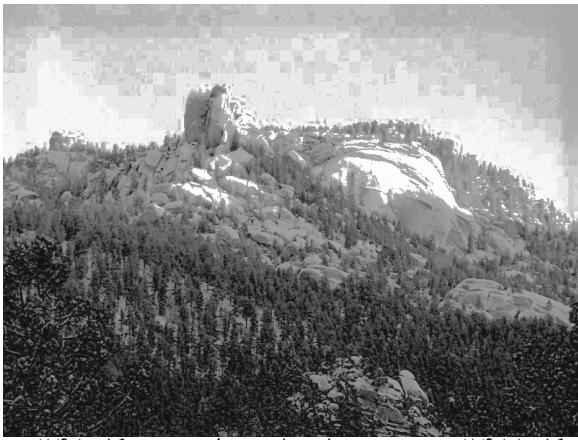
شکل ۲.۴.۱۰: خروجی متدهای CLAHE بر شکل ۲.۴.۱ سایز ۸ پنجره



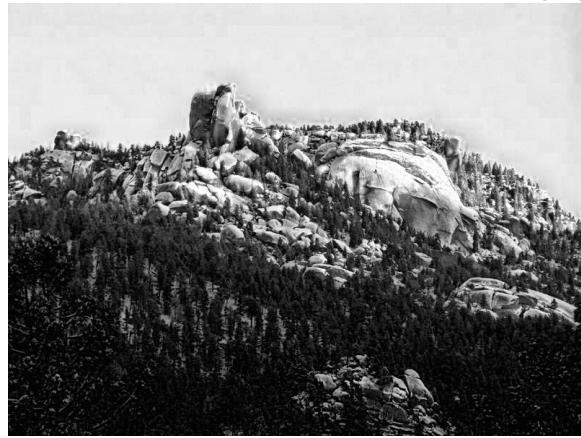
شکل ۲.۴.۱۴: خروجی متدهای CLAHE بر شکل ۲.۴.۱ سایز ۱۲۸ پنجره



شکل ۲.۴.۱۱: خروجی متدهای CLAHE بر شکل ۲.۴.۱ سایز ۱۶ پنجره



شکل ۲.۴.۱۵: خروجی همسان‌سازی سراسری بر شکل ۲.۴.۱



شکل ۲.۴.۱۲: خروجی همسان‌سازی سراسری بر شکل ۲.۴.۱ سایز ۳۲ پنجره



شکل ۲.۴.۱۸: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱۶ سایز پنجره ۱۵



شکل ۲.۴.۱۹: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱۶ سایز پنجره ۳۱

در این تصویر، در عکس اصلی تقریباً هیچ جزئیاتی دیده نمی‌شود و تنها قابل حدس است که دو آدم در عکس هستند. پس از انجام همسان سازی محلی، جزئیات کاملاً قابل مشاهده است و بنابراین کنتراست خروجی افزایش یافته است. همانطور که در عکس قبلی توضیح داده شده افزایش سایز پنجره باعث بهبود کیفیت تصویر و کاهش نویز می‌شود.

در ادامه روش دوم را بررسی میکنیم.

روش همسان سازی سراسری برای عکس هایی مناسب است که توضیح سطوح خاکستری در زمینه و محتوای اصلی عکس تقریباً مشابه باشد. اما در این عکس چنین نیست، به همین علت خروجی همسان سازی محلی، جزئیات بیشتری را نمایان میکند، هر چند در روش اصلی به علت وجود نویز کیفیت تصویر قابل قبول نیست، اما در صورتی که نیاز به فهمیدن جزئیاتی در عکس باشد، بررسی خروجی همسان سازی محلی راه حل مناسبتری است.



عکس ۲.۴.۱۶: عکس اصلی



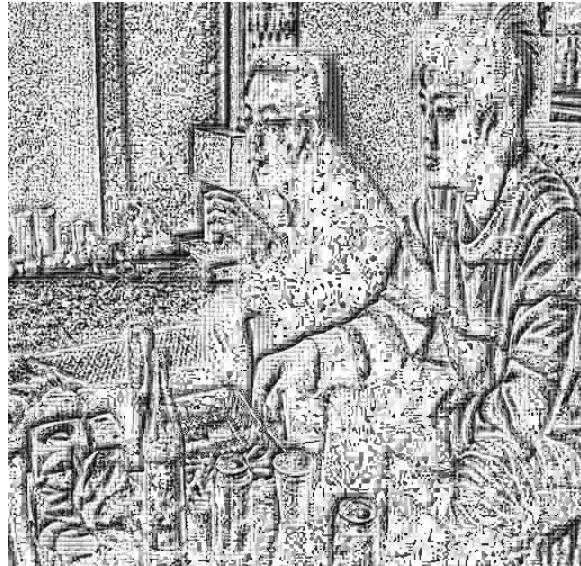
شکل ۲.۴.۱۷: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۱۶ سایز پنجره ۷



شکل ۲۰.۲۲: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲۰.۱۶ با سایز پنجره ۳۲



شکل ۲۰.۲۳: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲۰.۱۶ با سایز پنجره ۶۴



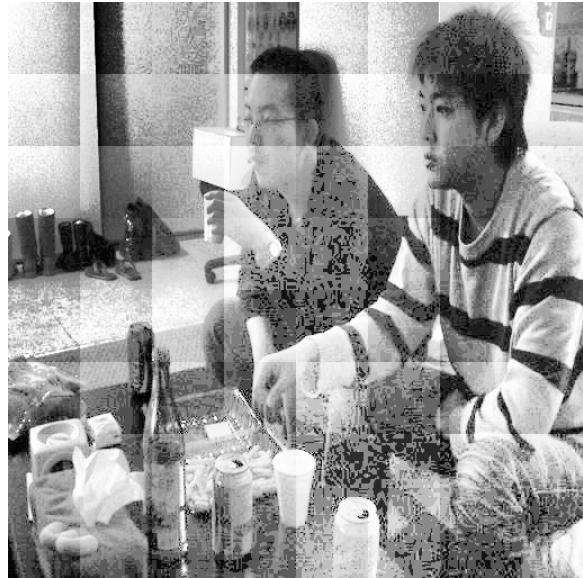
شکل ۲۰.۲۰: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲۰.۱۶ با سایز پنجره ۸



شکل ۲۰.۲۱: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲۰.۱۶ با سایز پنجره ۱۶



شکل ۲.۴.۲۶: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ سایز پنجره ۱۶



شکل ۲.۴.۲۷: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ با سایز پنجره ۱۶



شکل ۲.۴.۲۸: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ سایز پنجره ۸



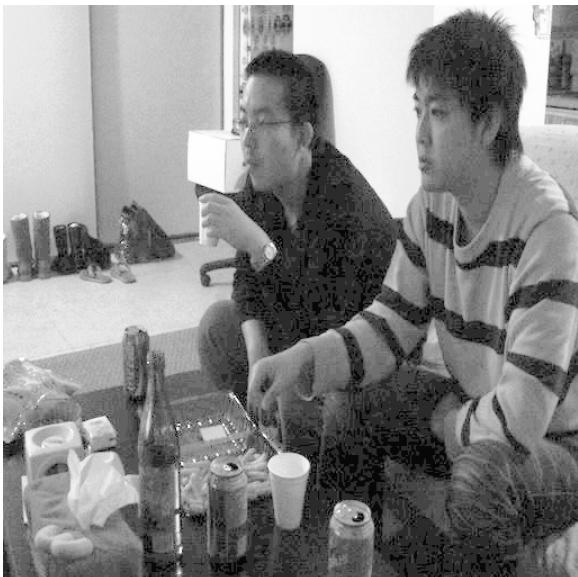
شکل ۲.۴.۲۹: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ سایز پنجره ۸

تاثیر افزایش سایز پنجره بر کیفیت در این سری از عکس‌ها کاملاً مشخص است.

عکس‌هایی که سایز پنجره در آنها کوچک است به شدت نویز دار بودند و با افزایش سایز پنجره کیفیت رفتہ بهبود می‌یابد.

عکس‌آخر به جز مشکل پنجره پنجره بودن از کیفیت خوبی برخودار است.

حال روش CLAHE را برای این عکس بررسی می‌کنیم.

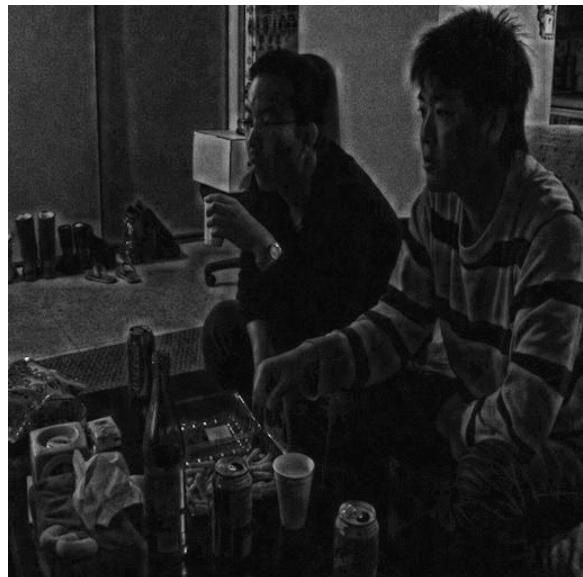


عکس ۲.۴.۳۰: خروجی همسان سازی سراسری foreground و background در این عکس توزیع در تقریباً برابر است و خروجی همسان سازی سراسری نیز تا حد خوبی قابل قبول است.

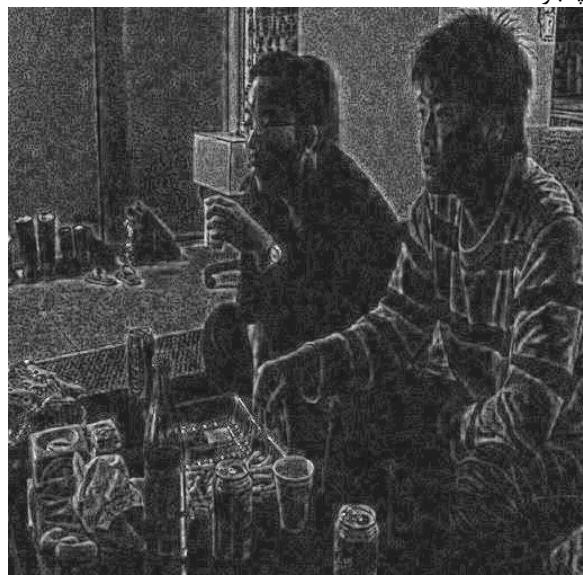


شکل ۲.۴.۳۱: عکس اصلی

همانطور که مشاهده می شود جزئیات عکس در عکس اصلی قابل تشخیص نیستند.

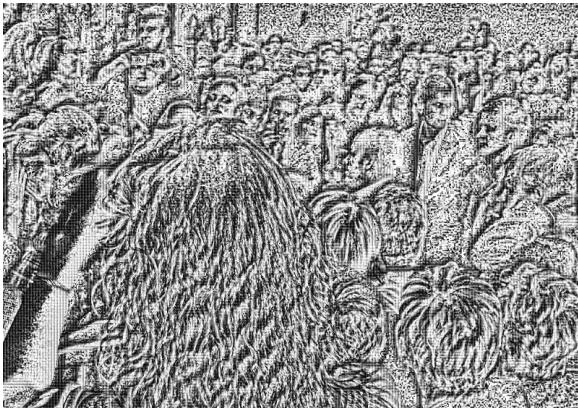


شکل ۲.۴.۲۸: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ سایز پنجره ۶۴

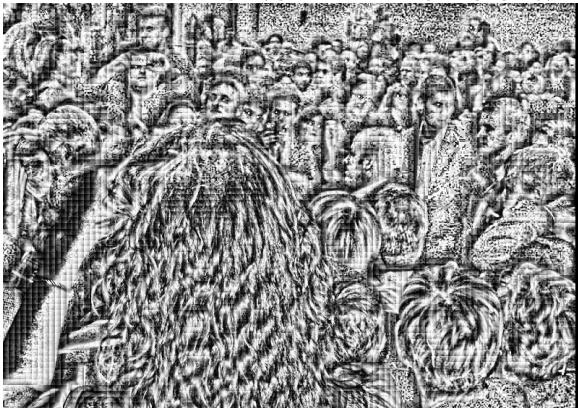


شکل ۲.۴.۲۹: خروجی متدهای CLAHE بر شکل ۲.۴.۱۶ سایز پنجره ۱۲۸

توضیحات این بخش، همانند بخش قبل است و افزایش پنجره تا حدی مفید است و بعد از آن باعث ایجاد نویز می شود.



شکل ۲.۴.۳۵: خروجی متده همسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۳۱ با سایز پنجره ۸



شکل ۲.۴.۳۶: خروجی متده همسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۳۱ با سایز پنجره ۱۶



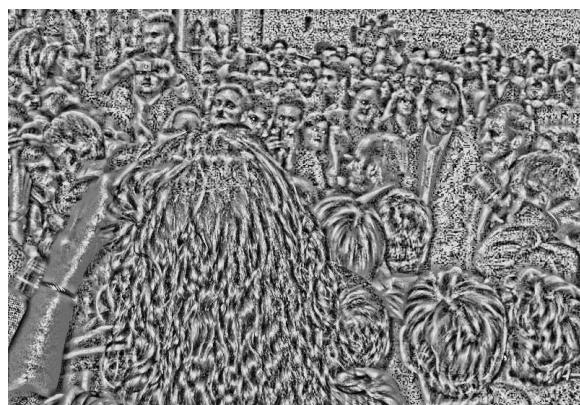
شکل ۲.۴.۳۷: خروجی متده همسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۳۱ با سایز پنجره ۳۲



شکل ۲.۴.۳۲: خروجی متده همسان سازی محلی بر شکل ۲.۴.۳۱ سایز پنجره ۷



شکل ۲.۴.۳۳: خروجی متده همسان سازی محلی بر شکل ۲.۴.۳۱ سایز پنجره ۱۵



شکل ۲.۴.۳۴: خروجی متده همسان سازی محلی بر شکل ۲.۴.۳۲ سایز پنجره ۲۱

جزییات تا حدی واضح شده است، متوجه آبوه جمعیت در عکس می شویم. اما کیفیت عکس مطلوب نیست.



شکل ۲.۴.۴۱: خروجی متدهای CLAHE بر شکل ۲.۴.۳۱ سایز پنجره ۱۶



شکل ۲.۴.۴۲: خروجی متدهای CLAHE بر شکل ۲.۴.۳۱ سایز پنجره ۲۲



شکل ۲.۴.۴۳: خروجی متدهای CLAHE بر شکل ۲.۴.۳۱ سایز پنجره ۶۴



شکل ۲.۴.۳۸: خروجی متدهای همسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۳۱ با سایز پنجره ۶۴

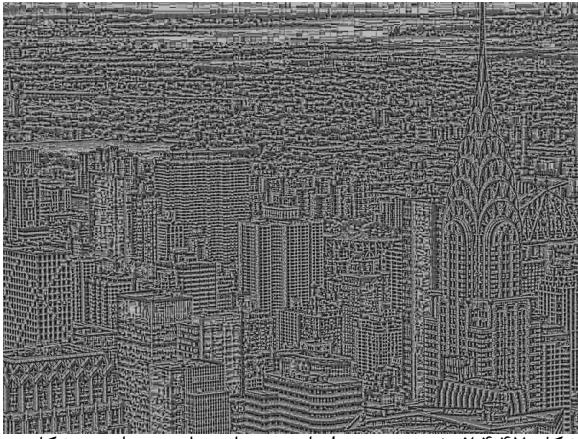


شکل ۲.۴.۳۹: خروجی متدهای همسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۳۱ با سایز پنجره ۱۲۸

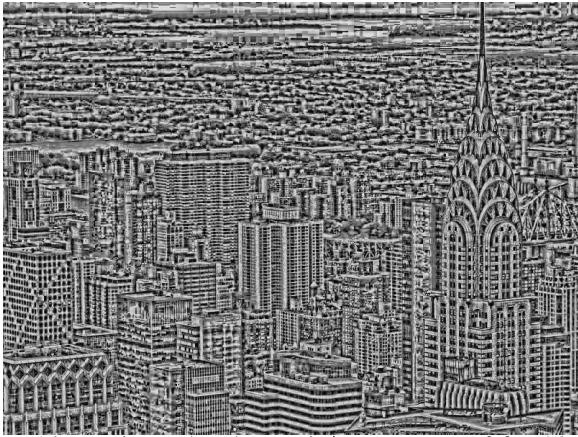
در این عکس نیز تاثیر افزایش پنجره بر کیفیت کاملاً مشخص است.



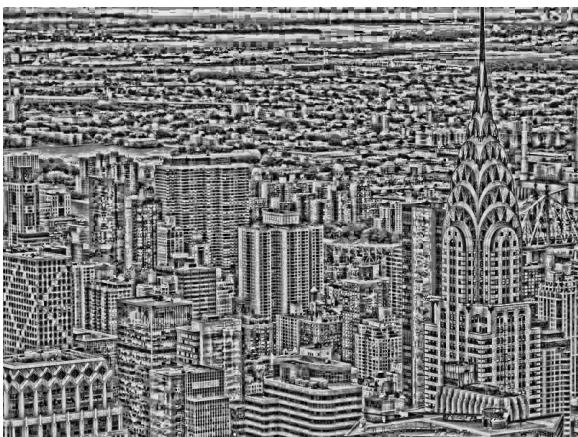
شکل ۲.۴.۴۰: خروجی متدهای CLAHE بر شکل ۲.۴.۳۱ سایز پنجره ۸



شکل ۲.۴.۴۷: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۴۶ سایز پنجره ۷

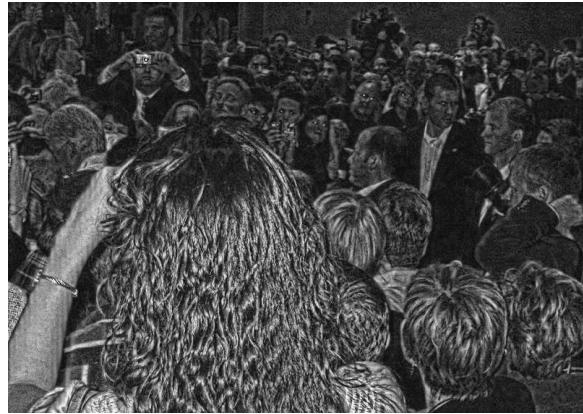


شکل ۲.۴.۴۸: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۴۶ سایز پنجره ۱۵



شکل ۲.۴.۴۹: خروجی متد اصلی همسان سازی محلی بر شکل ۲.۴.۴۶ سایز پنجره ۲۱

همانطور که این روش عکس های تیره را روشنتر می کند، به تیرگی عکس های روشن می افزاید. بر خلاف دو عکس قبلی این عکس، عکسی با سطح خاکستری بالاست و در پیش زمینه جزیيات واضح نیستند.



شکل ۲.۴.۴۴: خروجی متد CLAHE بر شکل ۲.۴.۳۱ سایز ۱۲۸ پنجره ۷

نتیجه گیری مشابه قسمت قبل است.



شکل ۲.۴.۴۵: همسان سازی سراسری هیستوگرام

در این عکس نیز همانند عکس ۲.۴.۱۶ به دلیل شباهت پیش زمینه و محتوای اصلی عکس، همسان سازی سراسری تاثیر قابل قبولی دارد.



عکس ۲.۴.۴۶: عکس اصلی



شکل ۲.۴.۵۳: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۴۶ با سایز پنجره ۶۴



شکل ۲.۴.۵۰: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۴۶ با سایز پنجره ۸



شکل ۲.۴.۵۴: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۴۶ با سایز پنجره ۱۲۸



شکل ۲.۴.۵۱: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۴۶ با سایز پنجره ۱۶



شکل ۲.۴.۵۵: خروجی متدهمسان سازی محلی با پنجره CLAHE بر شکل ۲.۴.۴۶ سایز پنجره ۸



شکل ۲.۴.۵۲: خروجی متدهمسان سازی محلی با پنجره غیر همپوشان بر شکل ۲.۴.۴۶ با سایز پنجره ۳۲



شکل ۲.۴.۵۹: خروجی متدهای CLAHE بر شکل ۲.۴.۴۶ سایز پنجره ۱۲۸



شکل ۲.۴.۵۶: خروجی متدهای CLAHE بر شکل ۲.۴.۴۶ سایز پنجره ۱۶



شکل ۲.۴.۶۰: تاثیر همسان‌سازی سراسری

جزئیات انتهای تصویر در همسان‌سازی محلی بهتر نمایان شده و به طور کل کنترast عکس پس از همسان‌سازی محلی افزایش چشم گیرتری داشته است.



شکل ۲.۴.۵۷: خروجی متدهای CLAHE بر شکل ۲.۴.۴۶ سایز پنجره ۳۲



شکل ۲.۴.۵۸: خروجی متدهای CLAHE بر شکل ۲.۴.۴۶ سایز پنجره ۶۴

```
plt.savefig("result/2.1.1/output.png")
```

قسمت ۲.۱.۳

```
clc; % Clear the command window.  
clear;  
  
I = imread('Camera Man.jpg');  
J = histeq(I,15);  
imshowpair(I,J,'montage');  
axis off  
  
% imhist(J,64)
```

```
clc; % Clear the command window.  
clear;  
  
I = imread('Camera Man.jpg');  
n = 2;  
Idouble = im2double(I);  
avg = mean2(Idouble);  
sigma = std2(Idouble);  
% J = imadjust(I,[0.1,0.5],[0.5,0.9]);  
J = imadjust(I,[avg-n*sigma  
avg+n*sigma],[]);  
imshowpair(I,J,'montage');  
axis off  
% imhist(J)
```

قسمت ۲.۲.۱

```
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
def get_histogram(img):  
  
    histogram = np.zeros((256) , int)  
    for i in range(0 , img.shape[0]):
```

Appendx:

قسمت ۲.۱.۱

```
import numpy as np  
import cv2 as cv  
  
def compute_histogram(img):  
    width , height = img.shape  
    histogram = np.zeros((256), int)  
    for i in range(0 , width):  
        for j in range(0, height):  
            histogram[img[i][j]] += 1  
    return histogram
```

قسمت ۲.۱.۲

```
import numpy as np  
import cv2 as cv  
import matplotlib.pyplot as plt  
import histogram as hst  
  
img = cv.imread("image/Camera Man.bmp"  
, cv.IMREAD_GRAYSCALE)  
hist = cv.calcHist([img] , [] , None ,  
[256] , [0,256])  
plt.figure("built_in fucntion")  
plt.plot(hist)  
#  
plt.savefig("result/2.1.1/Built_in_histogram.png")  
  
histo = hst.compute_histogram(img)  
  
fig = plt.figure()  
  
#show original image  
fig.add_subplot(221)  
plt.title("Camera Man image")  
plt.set_cmap('gray')  
plt.imshow(img)  
  
#show histogram  
fig.add_subplot(222)  
plt.title('histogram')  
plt.stem(histo)# ,  
use_line_collection=True)
```

```

        res = np.zeros((x - 2*hw , y - 2*hw) , np.uint8)

        for i in range(hw , x - hw ):
            for j in range(hw , y - hw ):
                rank = get_rank(img[i-hw:i+hw,j-hw:j+hw] , img[i][j])
                res[i - hw][j - hw] = np.uint8((rank * 255) / (window * window))

        return res

def AHE_v2(img , window):
    x,y = img.shape
    hw = int(window / 2)
    res = np.zeros((x , y) , np.uint8)

    for i in range(hw , x - hw + 1 , hw):
        for j in range(hw , y - hw + 1 , hw):
            histogram =
get_histogram(img[i-hw:i+hw,j-hw:j+hw])
            map_to_eqlz =
equilization(histogram , img[i-hw:i+hw,j-hw:j+hw])
            res[i-hw:i+hw,j-hw:j+hw] =
map_to_eqlz(img[i-hw:i+hw,j-hw:j+hw])
    return res

def HE(img):
    res = np.zeros((img.shape) ,
np.uint8)
    hstgm = get_histogram(img)
    trans = equilization(hstgm , img)
    res = trans(img)
    return res

def clahe(img, win_shape):
    largest_dim = np.argmax(win_shape)
    img = np.swapaxes(img, 0,
largest_dim)
    win_shape = list(win_shape)
    win_shape[0],
    win_shape[largest_dim] =
    win_shape[largest_dim], win_shape[0]

```

```

        for j in range(0 ,
img.shape[1]):
            histogram[img[i][j]] += 1
    return histogram

def equilization(histogram , img):
    size = img.shape[0] * img.shape[1]
    pdf = histogram / size
    cdf = np.cumsum(pdf)
    equlized = np.array(cdf * 255 ,
np.uint8)
    return lambda x: equlized[x]

def get_rank(img , intesity):
    rank = 0
    for i in range(0,img.shape[0]):
        for j in range(0,img.shape[1]):
            if img[i][j] <= intesity:
                rank = rank + 1
    return rank

def padding(img , w):
    x,y = img.shape
    pad_img = np.zeros((x + w, y + w) ,
np.uint8)

    h_w = int(w / 2)
    pad_img[h_w:x+h_w,h_w:y+h_w] = img
    for i in range(0, h_w):
        pad_img[i][h_w:y + h_w] =
img[0]
        pad_img[x + i + h_w][h_w:y +
h_w] = img[x - 1]

    for j in range(0 , h_w):
        pad_img[:, j] = pad_img[:,h_w]
        pad_img[:,j + y + h_w] =
pad_img[:,y - 1 + h_w]

    return pad_img

def AHE_v1(img , window):
    x,y = img.shape
    hw = int(window / 2)

```

```

        cv.imwrite(path, AHE_v1(pad_img ,
window_size))
if(typee == "v2"):
    path = "result/2.2.1/fake" +
str(img_nmbr) + "-" + str(window_size)
+ ".jpg"
    cv.imwrite(path, AHE_v2(img ,
window_size))

if(typee == "clahe"):
    path = "result/2.2.1/clahe" +
str(img_nmbr) + "-" + str(window_size)
+ ".jpg"
    cv.imwrite(path, clahe(img ,
window_size))

```

```

        img = np.pad(img, [((sz - 1) // 2,
sz // 2) for sz in win_shape],
"reflect")
        if _fast:
            if img.ndim == 2:
                res = _clahe_impl.clahe(
                    img[..., None],
*win_shape, 1, clip_limit)[..., 0]
            elif img.ndim == 3:
                res =
_clahe_impl.clahe(img, *win_shape,
clip_limit)

            else:
                res = clahe_nd(img, win_shape,
clip_limit)
        return np.swapaxes(
            res[tuple(np.s_[(sz - 1) // 2 :
-(sz // 2) or None]
                     for sz in
win_shape)], 0, largest_dim)

```

```

img_nmbr = int(input("Enter your input
image: "))
window_size = int(input("Enter window
size: "))
typee = input("Enter the method: ")

path = "image/HE" + str(img_nmbr) +
".jpg"
img = cv.imread(path ,
cv.IMREAD_GRAYSCALE)

pad_img = padding(img , window_size)

if(typee == "global"):
    path = "result/2.2.1/Global" +
str(img_nmbr) + ".jpg"
    cv.imwrite(path, HE(img))
if(typee == "v1"):
    path = "result/2.2.1/original" +
str(img_nmbr) + "-" + str(window_size)
+ ".jpg"

```