به نام خدا

تمرین سری اول- درس مبانی بینایی کامپیوتر

سید محمد علی نخاری- شماره دانشجویی: 99521496

سوال اول)

سؤال اول)

الف) اگر فرض کنیم شی با سرعت به سمت چپ در حال حرکت باشد، دوربین rolling shutter با ثابت باشد؛ تصویر ثبت شده به صورت زیر خواهد بود:

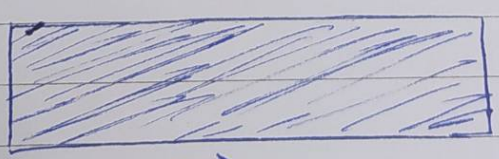
دلیل این امر آن است که اگر فرض کنیم، دوربین ما از بالا به پایین تصویر را ثبت می‌کند، پس در لحظه اول ما اولین خط از شکل را ثبت می‌کنیم و با حرکت شی به سمت چپ این خطوط به شکل بالا درآمده و تصویر نهایی به صورت یک متوازی الاضلاع خواهد شد.

ب) وقتی گفته می‌شود دوربین global shutter است، یعنی اینکه صفحه آرایه با ابعاد mxn می‌تواند نور بازتاب شده از اجسام را در یک بازه زمانی بسیار کوتاه ثبت نماید.

منظور از سرعت shutter پایین این است که shutter دوربین مدت زمانی بیشتری باز است در نتیجه تصویر ثبت شده از محیط با جزئیات بیشتری خواهد بود.

+ حال در این سؤال گفته شده، شی با سرعت به سمت راست در حرکت است و چون دوربین global shutter سرعت shutter پایینی دارد پس تصویر با جزئیات کامل ثبت خواهد شد.

تصویر ثبت شده:



* در واقع، مبلغ داده شده در سؤال به سمت راست کشیده می‌شود.

سوال دوم)

الف)

سؤال (دوم)

الف)

object ⟷ u ⟷ Lens ⟷ v ⟷ film $\quad \frac{1}{f} = \frac{1}{u} + \frac{1}{v}$

if $v = 10\,cm$ and $u = 70\,cm$

- شیء مورد نظر توپ بسکتبال باشد که در فاصلهٔ ده سانتی متر از صفحهٔ film قرار دارد.

+ ابتدا باید مقدار فاصلهٔ کانونی لنز دوربین را از طریق معادلهٔ لنز نازک محاسبه کنیم:
برای بدست آمدن f از اطلاعات مربوط به توپ بسکتبال استفاده می کنیم چون کیفیت خوبی دارد:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} =$$

$$f = \frac{u.v}{u+v} = \frac{70\,cm \cdot 10\,cm}{80\,cm} = \frac{35}{4} = 8.75$$

+ حال فاصلهٔ کانونی مناسب برای ثبت توپ بسکتبال را بدست می آوریم:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \Rightarrow \frac{1}{f} = \frac{u+v}{uv} \rightarrow f = \frac{u.v}{u+v}$$

$$u + v = 50\,cm \rightarrow u = 50 - v = 40\,cm$$

$$f = \frac{40\,cm \times 10\,cm}{50\,cm} = 8$$

+ در این حالت فاصلهٔ کانونی، باید به اندازهٔ ۰/۷۵ کاهش یابد. تا بتوانیم از توپ بسکتبال به خوبی تصویر برداری کنیم.

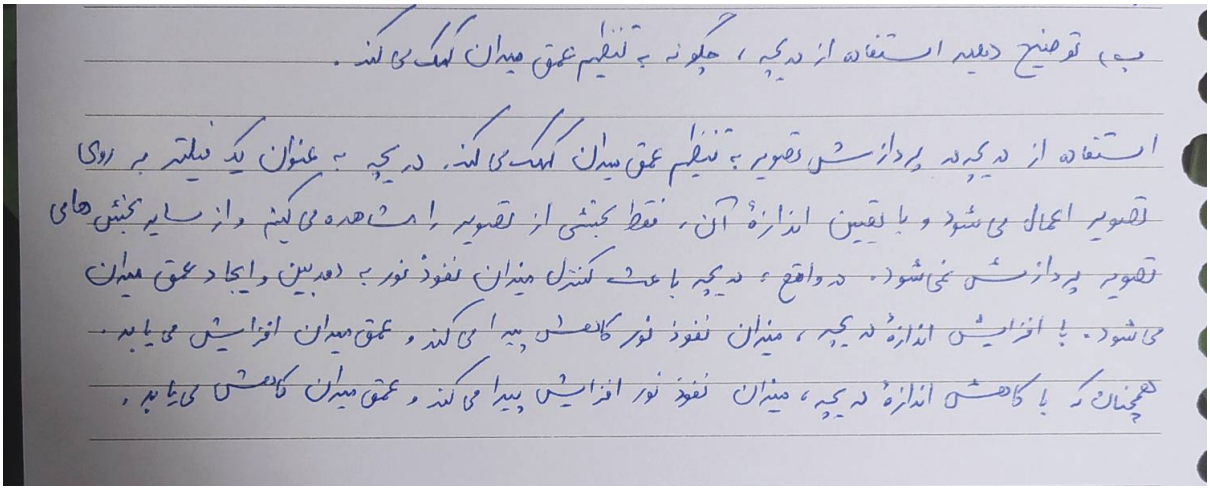
- شیء مورد نظر توپ فوتبال باشد که در فاصلهٔ ۶۰ سانتی متر از توپ بسکتبال قرار دارد.

$$u + v = 50\,cm + 60\,cm = 110\,cm \quad , \quad v = 10\,cm \quad , \quad u = 100\,cm$$

$$f = \frac{u.v}{u+v} = \frac{100\,cm \times 10\,cm}{110\,cm} \simeq 9.09$$

+ در این حالت فاصلهٔ کانونی به حدود ۰/۳۴ افزایش پیدا می کند.

ب)

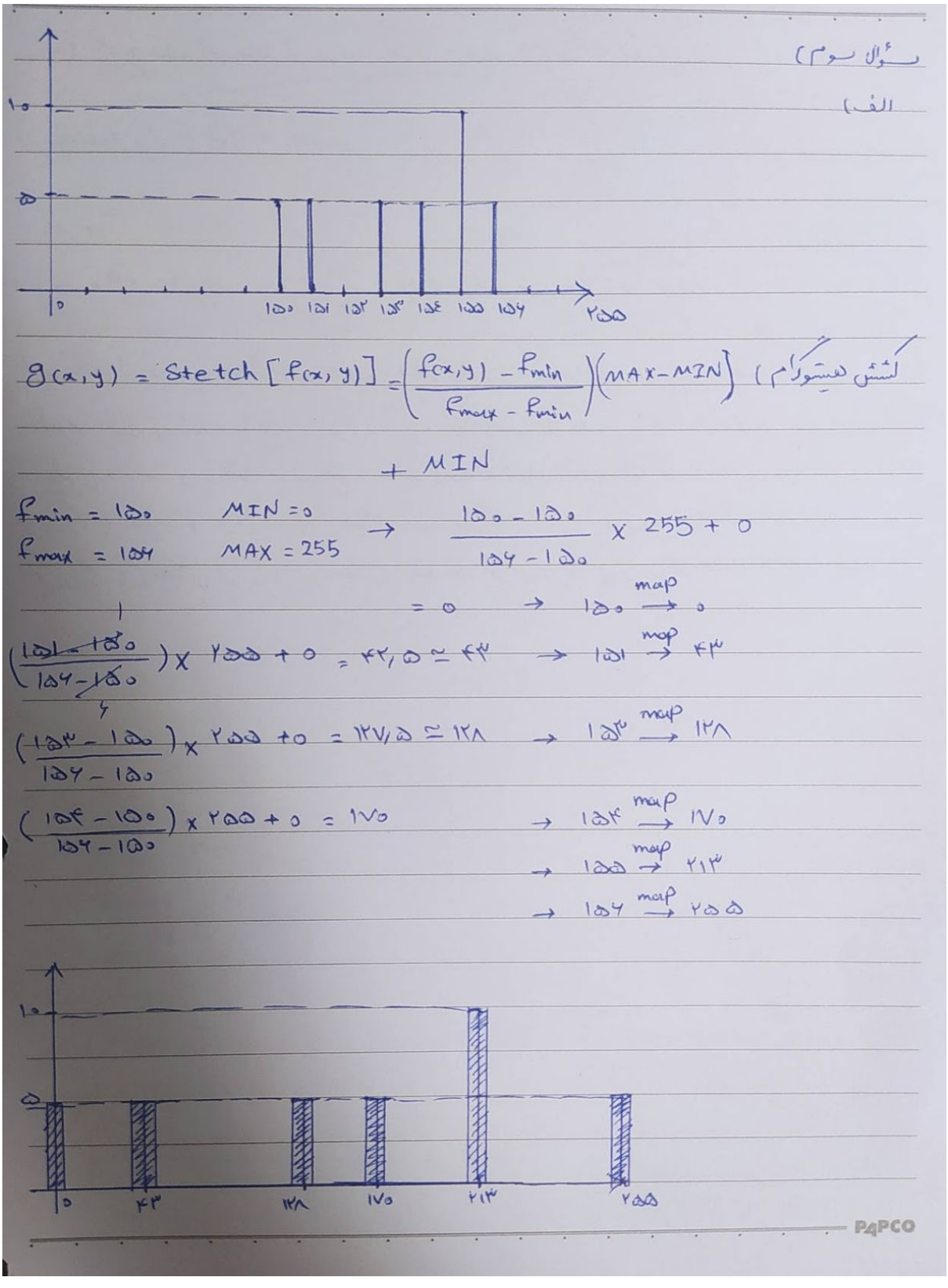


ب) توضیح دهید استفاده از دریچه، چگونه به تنظیم عمق میدان کمک می‌کند.

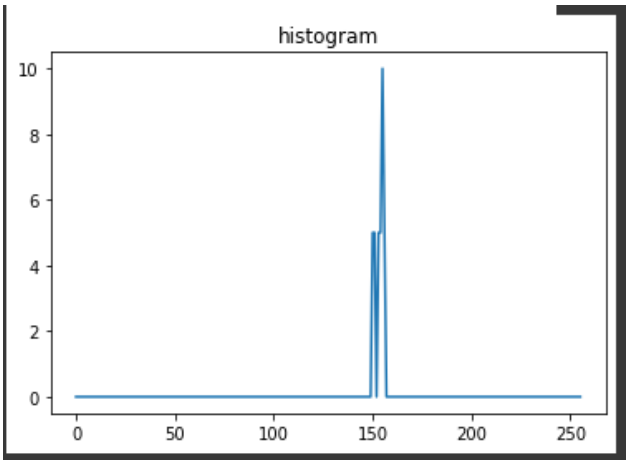
استفاده از دریچه در پردازش تصویر به تنظیم عمق میدان کمک می‌کند. دریچه به عنوان یک فیلتر بر روی تصویر اعمال می‌شود و با تعیین اندازهٔ آن، فقط بخشی از تصویر را مشخص می‌کنیم و از سایر بخش‌های تصویر پردازش نمی‌شود. در واقع، دریچه باعث کنترل میزان نفوذ نور به دوربین و ایجاد عمق میدان می‌شود. با افزایش اندازهٔ دریچه، میزان نفوذ نور کاهش پیدا می‌کند و عمق میدان افزایش می‌یابد. همچنان که با کاهش اندازهٔ دریچه، میزان نفوذ نور افزایش پیدا می‌کند و عمق میدان کاهش می‌یابد.

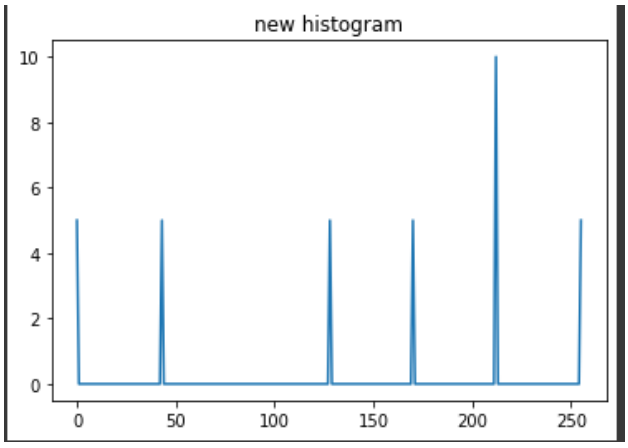
سوال سوم

الف)

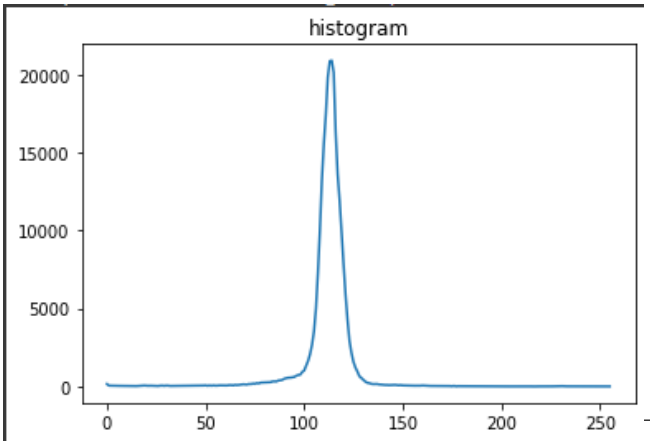

سوال سوم
الف)
۱۰
۵
۰
۱۵۰ ۱۵۱ ۱۵۲ ۱۵۳ ۱۵۴ ۱۵۵ ۱۵۶
۲۵۵
g(x,y) = Stetch [f(x,y)] = ((f(x,y) - fmin) / (fmax - fmin)) (MAX-MIN) + MIN
کشش هیستوگرام
fmin = ۱۵۰ MIN = ۰
fmax = ۱۵۶ MAX = 255
(۱۵۰ - ۱۵۰) / (۱۵۶ - ۱۵۰) x 255 + ۰
= ۰ → ۱۵۰ map → ۰
(۱۵۱ - ۱۵۰) / (۱۵۶ - ۱۵۰) x ۲۵۵ + ۰ = ۴۲,۵ ≃ ۴۳ → ۱۵۱ map → ۴۳
(۱۵۳ - ۱۵۰) / (۱۵۶ - ۱۵۰) x ۲۵۵ + ۰ = ۱۲۷,۵ ≃ ۱۲۸ → ۱۵۳ map → ۱۲۸
(۱۵۴ - ۱۵۰) / (۱۵۶ - ۱۵۰) x ۲۵۵ + ۰ = ۱۷۰
→ ۱۵۴ map → ۱۷۰
→ ۱۵۵ map → ۲۱۳
→ ۱۵۶ map → ۲۵۵
۱۰
۵
۰ ۴۳ ۱۲۸ ۱۷۰ ۲۱۳ ۲۵۵
PqPCO

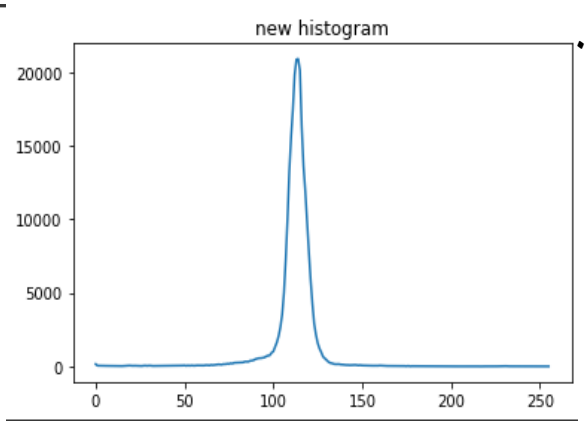
ب) در این قسمت باید هیستوگرام عکس دو بعدی قسمت قبل را با استفاده از توابع موجود در کتابخانه opencv بدست بیاوریم. برای این کار image1 یک را به صورت یک ماتریس دوبعدی نگهداری میکنیم. در قسمت بعد تابع calc_hist را تعریف می کنیم که در ورودی عکس را به صورت آرایه دو بعدی گرفته و هیستوگرام محاسبه شده را برمی گرداند. با اجرای سلول بعدی عکس ورودی و نمودار هیستوگرام آنرا مشاهده خواهیم کرد. در سلول بعدی باید تابع stretch_hist را تعریف کنیم که وظیفه آن کشش هیستوگرام عکس ورودی است. در داخل این تابع طبق فرمول نوشته شده در اسلاید ها کوچکترین و بزرگترین پیکسل های عکس ورودی را پیدا میکنیم. سپس با استفاده از تابع np.nditer بر روی عکس ورودی پیمایش کرده مقدار جدید هر پیکسل را محاسبه می کنیم. با اجرای این تابع عکس جدید و نمودار هیستوگرام آن قابل مشاهده خواهد بود.





ج) ابتدا image2 را از محیط گوگل درایو خود خوانده و آن را بوسیله تابع cv2_imshow نمایش میدهیم. حال در سلول بعدی همچون قسمت قبل ابتدا هیستوگرام آن را نمایش داده و سپس با استفاده از تابع پیاده سازی شده در قسمت قبل سعی در بهبود آن به کمک روش کشش هیستوگرام میکنیم. پس از گرفتن خروجی و مشاهده نمودار هیستوگرام جدید، تغییری در بهبود آن نمیکند.





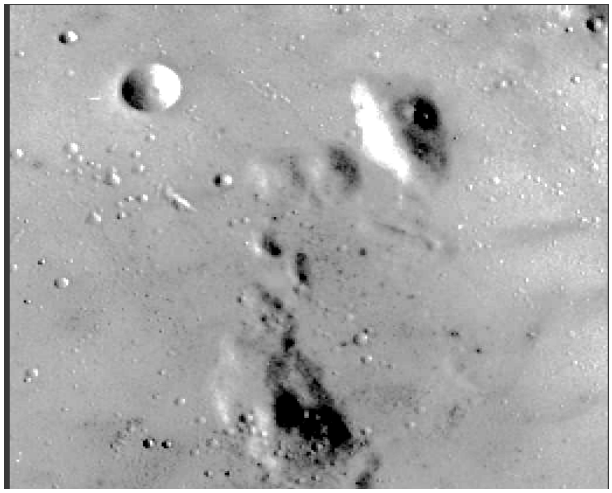
همانطور که از نمودار های هیستوگرام دو تصویر قابل مشاهده است هیچ تغییری در بهبود کیفیت آن رخ نداده است. دلیل آن هم بخاطر این است که در تصویر داده شده تعدادی از پیکسل های رنگ سیاه(0) و رنگ سفید(255) را دارند و با توجه به فرمول گفته شده برای کشش هیستوگرام مخرج کسر (fmax-fmin) با مقدار (MAX-MIN) ساده شده و چون مقدار fmin و MIN هردو صفر هستند تغییری در پیکسل ها داده نمیشود.

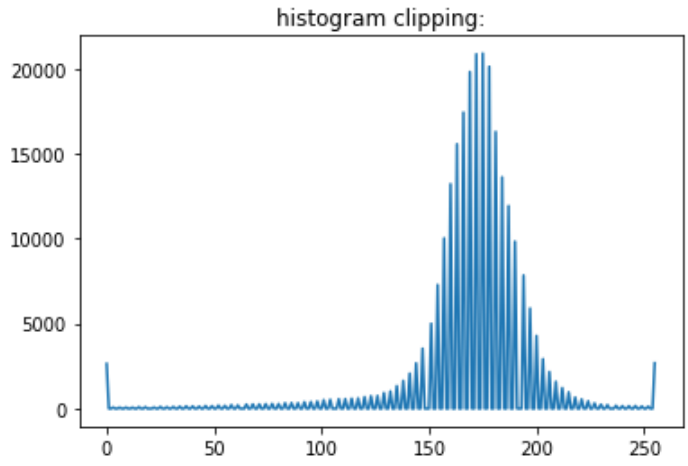
$$g(x,y) = stretch[f(x,y)] = \left(\frac{f(x,y) - f_{min}}{f_{max} - f_{min}}\right)(MAX - MIN) + MIN$$

د) در این قسمت میخواهیم مشکل گفته شده در روش کشش هیستوگرام را حل کنیم. برای این کار روش های مختلفی وجود دارد که میتوان به برش هیستوگرام، متعادل سازی هیستوگرام و ... اشاره کرد. در این قسمت هر دو روش گفته شده پیاده سازی شده است. در ابتدا روش برش هیستوگرام را توضیح میدهیم:

برای برش هیستوگرام نیاز است که تعدادی از پیکسل های ابتدایی و پایانی را صرف نظر کنیم. در اینجا تعداد پیکسل های صرف نظر شده یک درصد هستند. ابتدا تصویر داده شده را بر اساس رنگ های آن مرتب میکنیم. سپس یک درصد از پیکسل های ابتدایی و پایانی مرتب شده را صرف نظر میکنیم و از میان پیکسل های باقی مانده مینیمم و ماکسیمم میگیریم. که این مقادیر همان f1 و f99 هستند. در پایان بر روی عکس پیمایش کرده و پیکسل هایی که مقدار کمتر از f1 دارند را برابر با min و پیکسل هایی که مقدار بیشتر از f99 دارند را برابر با max قرار داده و باقی پیسکل ها را بر اساس فرمول نوشته شده در اسلاید ها محاسبه میکنیم.

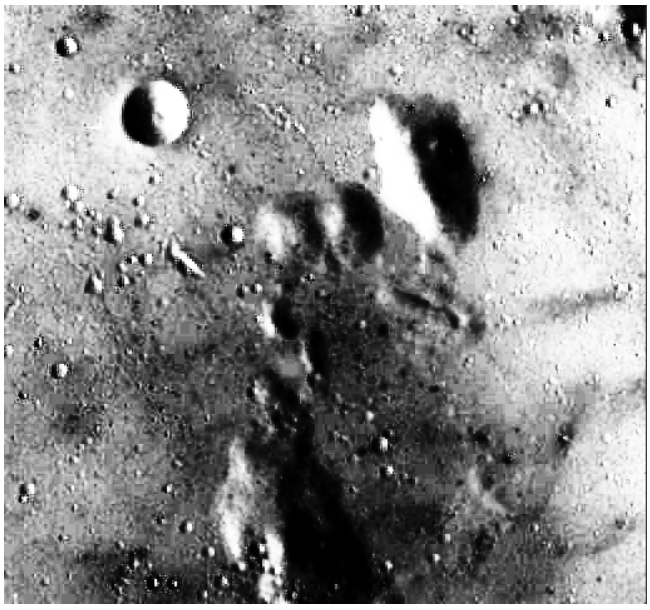
$$g(x,y) = clip[f(x,y)] = \left(\frac{f(x,y) - f_1}{f_{99} - f_1}\right)(MAX - MIN) + MIN$$

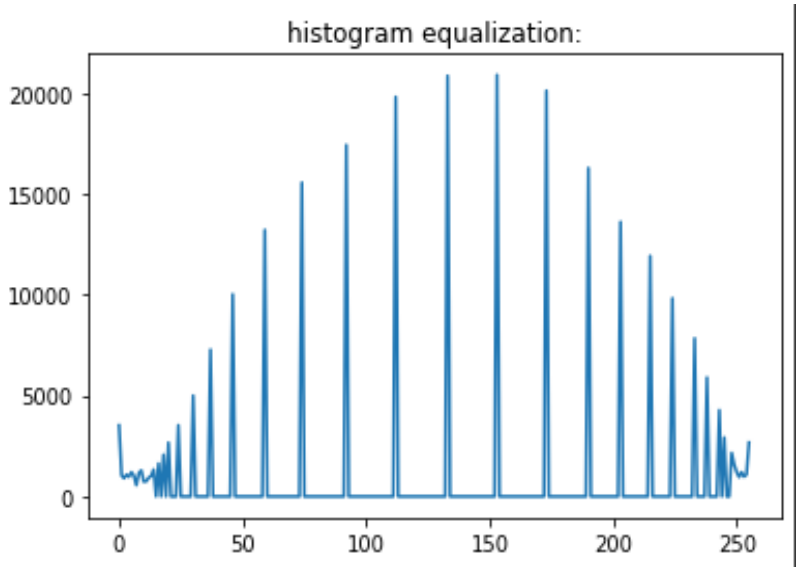




همانطور که دیدیم کیفیت عکس کمی بهبود پیدا کرده و هیستوگرام آن در نواحی روشن بهتر کشیده شده است.

حال در ادامه روش متعادل سازی هیستوگرام را پیاده سازی میکنیم. در ابتدا هیستوگرام عکس داده شده را به دست می آوریم و از روی آن cdf را با استفاده از یک حلقه محاسبه میکنیم. پس از محاسبه cdf باید آن را normalized کنیم که برای این کار هر مقدار cdf را بر تعداد کل پیکسل های عکس تقسیم کرده و مقدار L-1 را در آنها ضرب میکنیم. در ادامه بر روی عکس پیمایش کرده و مقادیر روشنایی هر پیکسل را بر اساس آرایه به دست آمده از cdf های متعادل سازی شده به دست می آوریم. در آخر نتیجه به دست آمده را به تابع stretch داده و آن را میکشیم. نتیجه به صورت زیر خواهد بود:



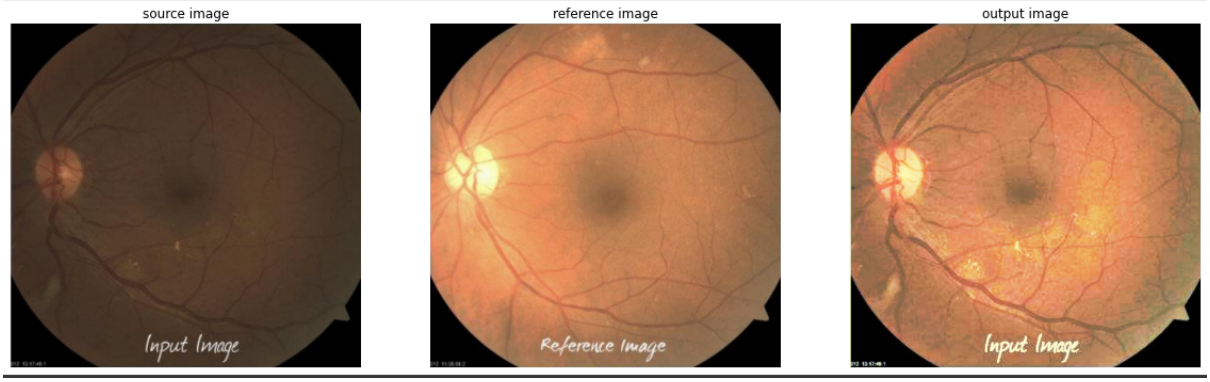


همانطور که مشاهده میشود کیفیت عکس در این حالت نسبت به حالت قبل بهتر شده است و نمودار هیستوگرام یکنواخت تری از آن میبینیم.

سوال چهارم)

الف) در این سوال بر خلاف سوال قبلی برای محاسبه هیستوگرام باید از حلقه فور استفاده کنیم و بر روی عکس داده شده که آن را به یک آرایه یک بعدی تبدیل میکنیم پیمایش کرده و تعداد تکرار رنگ های هر پیکسل را بدست آورده و به عنوان یک آرایه یک بعدی بر میگردانیم.

در سلول بعدی تابع محاسبه cdf را پیاده میکنیم که این تابع هر کدام از کانال های یک عکس را گرفته و ابتدا هیستوگرام آن را به کمک تابع نوشته شده در قسمت قبل محاسبه کرده و در اخر cdf آن را با پیمایش بر روی آرایه هیستوگرام تصویر محاسبه میکند. در سلول بعدی تابع hist_matching را پیاده کردیم. در این تابع ابتدا با استفاده از یک حلقه for بر روی کانال های عکس پیمایش کرده و برای هرکدام از کانال ها جداگانه به محاسبه cdf پرداخته و سپس با استفاده از حلقه های for و while مشخص میکنیم که هرکدام از پیکسل های عکس src باید به کدام یک از پیکسل های عکس ref تبدیل شوند. در آخر هم با دو حلقه for مشخص میکنیم هر کدام از پیکسل های موجود در عکس src چه مقداری باید به آن ها نسبت داده شود.



در آخر برای بهبود سرعت اجرا برنامه میتوانیم از توابع موجود در پایتون استفاده و از حلقه ها کمتر استفاده کنیم. به عنوان نمونه چندین خط گفته شده برای مپ کردن پیکسل ها از عکس src به عکس ref میتوانیم دو خط زیر را استفاده کنیم:

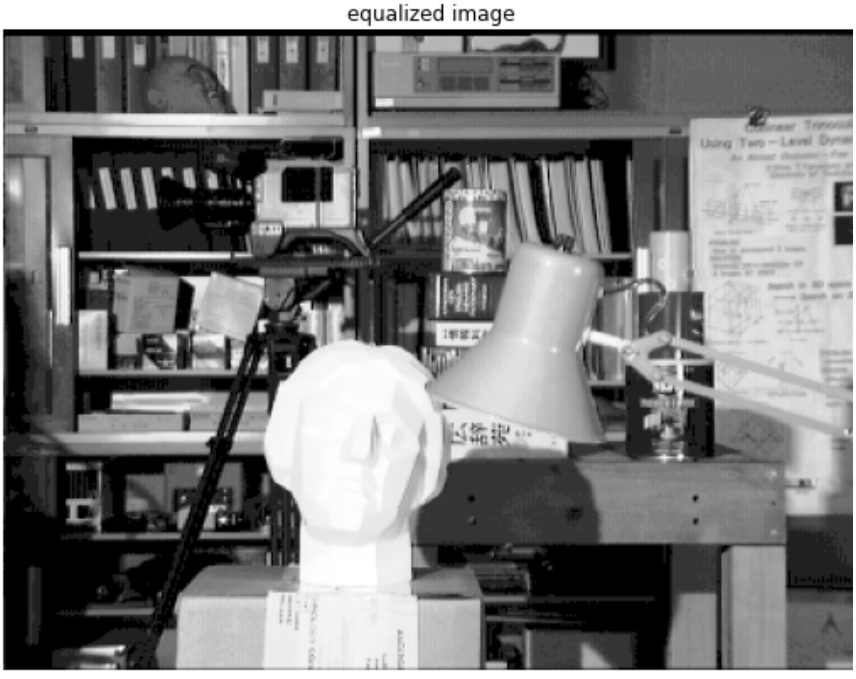
```
# implement without for and while loop
#-------------------------------------------------------------------------------
# Create the mapping table
src_to_ref = np.searchsorted(ref_cdf, src_cdf).astype('uint8')
# Apply the mapping table to the source image to create the output image
#-------------------------------------------------------------------------------
output_image[:, :, channel] = src_to_ref[src_image[:, :, channel]]
```

در کد بالا ابتدا با استفاده از تابع searchsorted از کتابخانه numpy دو آرایه ref_cdf و src_cdf به عنوان ورودی میدهیم و در خروجی آرایه ای یک بعدی که ایندکس پیکسل هایی است که از آرایه src_cdf در آرایه ref_cdf بزرگتر مساوی با آن پیدا شده است و در نهایت با استفاده از این ویژگی پایتون میتوانیم که به عنوان ورودی آرایه یک بعدی src_to_ref، آرایه ای دو بعدی داده و مقدار جدیدی به هرکدام نسبت میدهیم.

ب) منطبق سازی هیستوگرام یا histogram-matching روشی است که در آن به افزایش contrast تصویر و روشن تر شدن آن میپردازیم. از جمله کاربرد های آن میتوان به افزایش کیفیت تصویر در زمینه های پزشکی برای تشخیص ویژگی ها عکس بیماری ، تصاویر ماهواره ای و ... اشاره کرد.

سوال پنجم)

الف) در این قسمت برای بهبود تصویر از تابع آماده کتابخانه opencv استفاده میکنیم. این تابع یک عکس به عنوان ورودی گرفته و عکس بهبود یافته را در خروجی برمیگرداند. عکس بهبود یافته به صورت زیر خواهد بود:



همانطور که مشخص است کیفیت عکس بهبود نیافت چرا که تابع بهبود هیستوگرام را بر روی کل عکس اعمال کرده و به همین دلیل برخی نواحی که روشن تر هستند روشن تر شده چرا که کلیت تصویر را رنگ تیره تشکیل میدهد که برای بهبود آن ناچار هستیم تصویر را روشن تر کنیم.

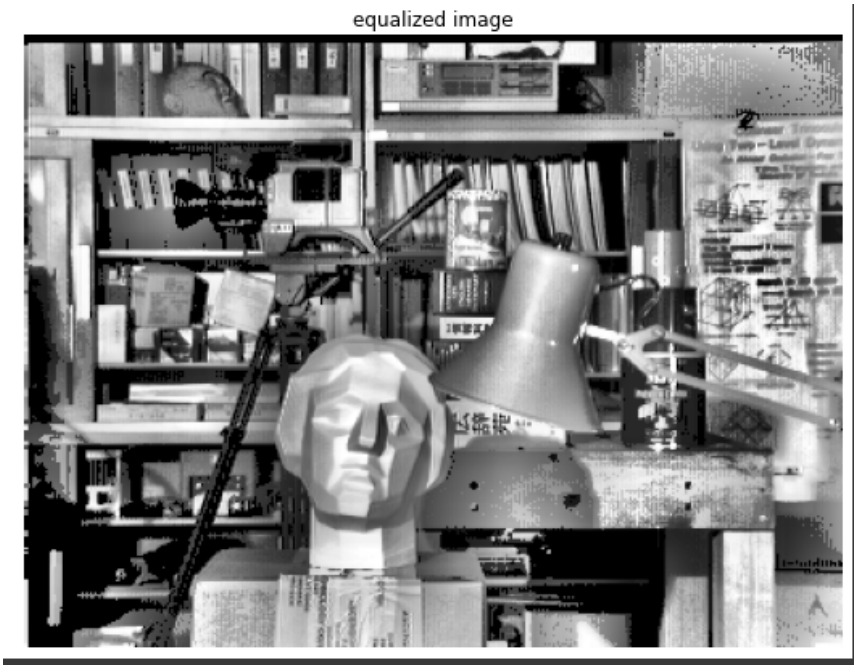
ب) در روش اول عکس ورودی را به چندین گرید تقسیم کرده و هر گرید را جداگانه توسط تابع آماده موجود در کتابخانه opencv بهبود میدهیم.

تصویر بهبود نیافت چرا که در این روش هر گرید به صورت جداگانه بهبود پیدا میکنند و از آنجایی که برخی نواحی گرید ها با یکدیگر همپوشانی دارند و بهبود یافته آن ها با یکدیگر تفاوت پیدا میکند اثر آنها باقی مانده و تصویر را به صورت مجموعه ای از گرید ها خواهیم داشت.

ج) در روش دوم برای اینکه مشکل قسمت قبلی را نداشته باشیم بر روی عکس پیمایش کرده و برای بهبود هر پیکسل یک گرید به دور هر پیکسل در نظر گرفته و آن گرید را بهبود میبخشیم.

نکته ای که باید به آن توجه کنیم لبه های آن تصویر است که باید برای بهبود لبه های آن به تصویر اصلی حاشیه اضافه کنیم. برای این کار از کتابخانه copyMakeBorder استفاده کرده و بالا، پایین، چپ و راست تصویر به یک مقدار حاشیه اضافه میکنیم.

نتیجه نهایی این روش به صورت زیر خواهد بود:

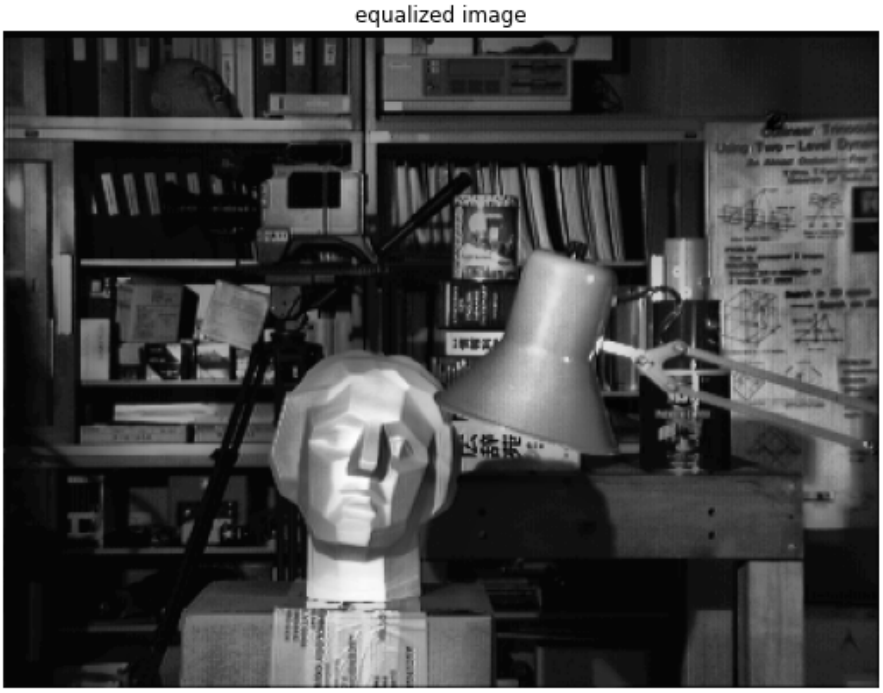


در این روش عکس نسبت به روش های قبلی بهبود بهتری داشت اما مشکل بزرگ این روش در آن است که نویز موجود در تصویر را تقویت میکند. به عنوان نمونه در ناحیه سمت راست بالای تصویر این تقویت نویز قابل مشاهده خواهد بود. دلیل آن هم بخاطر این است که در این ناحیه تمامی پیکسل ها رنگ تیره دارند و وقتی از بهبود هیستوگرام استفاده میکنیم، فاصله این رنگ ها بیشتر شده و نویز در تصویر پدید می آید.

د) در این قسمت میخواهیم روش CLAHE را پیاده سازی کنیم. ایده اصلی در این روش این است که به ازای هر پیکسل از تصویر ورودی، گرید هایی را در نظر گرفته و ابتدا نمودار هیستوگرام آن گرید را کشیده و بر اساس یک مقدار clip_limit، پیکسل هایی که از حد مشخصی از یک رنگ خاص دارند را تشخیص و مقدار اضافه رنگ های آن ها را به شکل یکسانی میان تمام 256 سطح رنگ پخش میکنیم. برای این کار هم تعداد رنگ های بیشتر از حد مشخص در طول کل تصویر را محاسبه و بر تعداد کل پیکسل های تصویر تقسیم میکنیم. حاصل به دست آمده را به تمامی رنگ های 0 تا 255 اضافه میکنیم.

در این روش هرچه مقدار clip limit را کوچکتر در نظر بگیریم، هیستوگرام حاصل از این کار برای هر گرید یکنواخت تر خواهد شد.

علاوه بر این هرچه اندازه گرید ها را بزرگتر در نظر بگیریم کیفیت نهایی عکس به کیفیت عکس در حالتی که عکس را به صورت سراسری بهبود میدهیم نزدیک تر میشود.



منابع استفاده شده در حل تمرین:

- سایت های رسمی کتابخانه های numpy, opencv و اسلاید های تدریس شده در کلاس