سید محمد طاها طباطبایی – 9812762838

گزارش تمرین 1

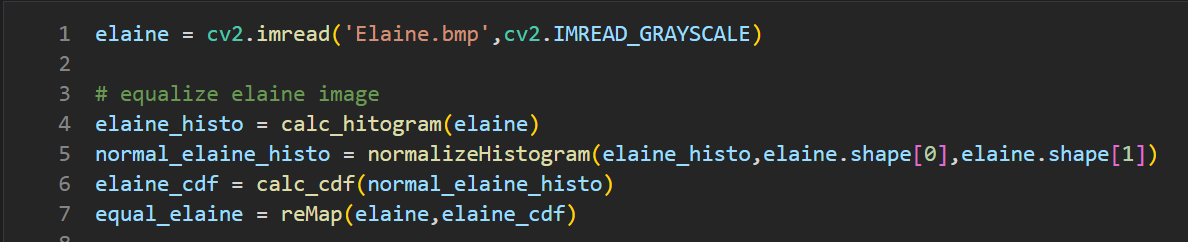
چکیده:

در این تمرین، به بررسی 3 عملیات پایه روی تصاویر می پردازیم. در بخش اول، با گسسته سازی مقادیر پیکسل های تصویر و نحوه تاثیر آن بر کیفیت عکس آشنا می‌شویم. در بخش دوم، دو متد کاهش ابعاد تصویر، یکی حذف سطر و ستون و دیگری میانگین گیری را بررسی می‌کنیم. همچنین برای افزایش ابعاد تصویر نیز، دو متد کپی و تکرار و تبدیل دو خطی را پیاده‌سازی می‌کنیم. در بخش آخر نیز، به بررسی تاثیر تعداد بیت های مورد استفاده در ذخیره سازی تصویر، بر روی کیفیت عکس می‌پردازیم.

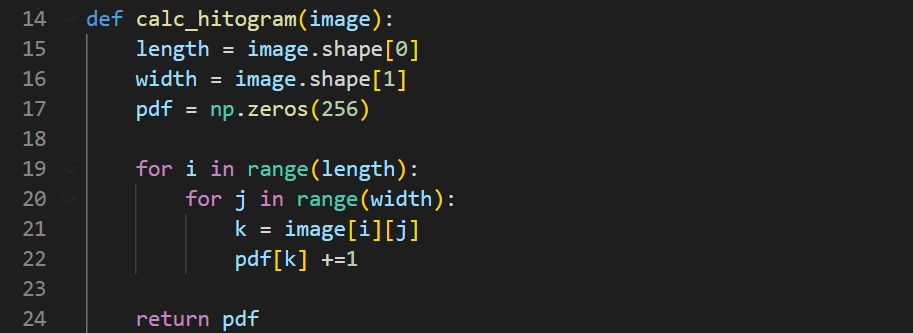
1.1.1

توضیح فنی:

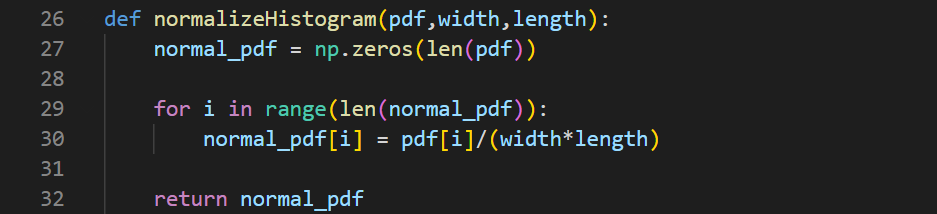
در سلول اول، ابتدا به محاسبه هیستوگرام محلی برای تصویر elaine پرداخته ایم.



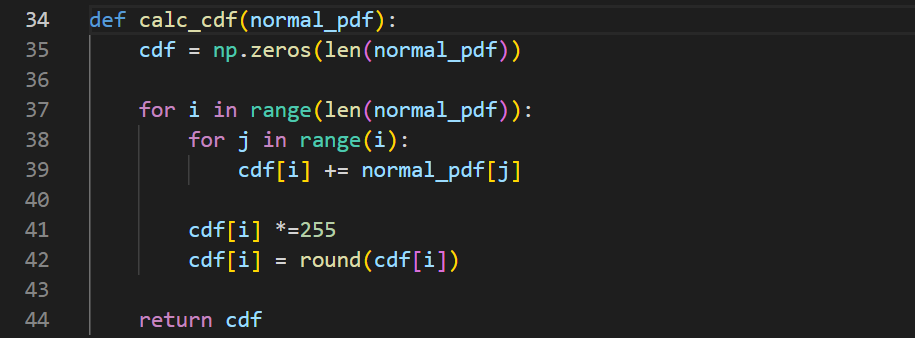
در اینجا، ابتدا با تابع calc\_histogram ، هیستوگرام تصویر را محاسبه می‌کنیم. سپس با تابع normalizationhistogram ، تصویر را نرمال‌سازی می‌کنیم. در گام بعد، تابع calc\_cdf برای محاسبه cdf تصویر استفاده می‌شود. در انتها، تصویر جدید مطابق cdf محاسبه شده، توسط تابع reMap ساخته می‌شود. تابع calc\_histogram به شکل زیر تعریف می‌شود:

 در این تابع، عکس مورد نظر به عنوان ورودی تابع استفاده می‌شود. در دو حلقه تو در تو، فراوانی رخ داد هر پیکسل که مقداری بین 0 تا 255 دارد را، در یک آرایه به نام pdf ذخیره می‌کنیم. طول این آرایه 256 است، و اندیس k این آرایه، نشان دهنده فراوانی پیکسل هایی با مقدار k است.

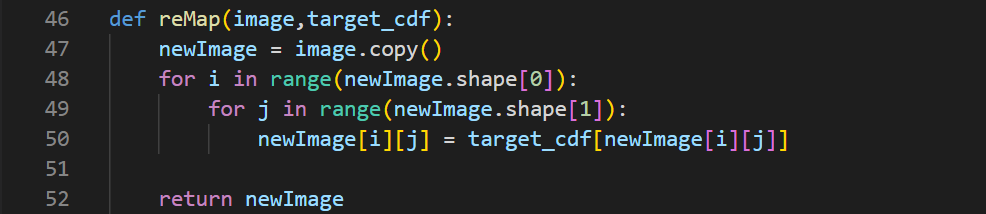
برای اجرای روال متعادل سازی هیستوگرام، در ابتدا نیاز داریم تا طبق الگوریتم ارائه شده در درس، هیستوگرام فعلی را نرمالایز کنیم. برای اینکار یک تابع به نام normalizeHistogram نوشته ایم.



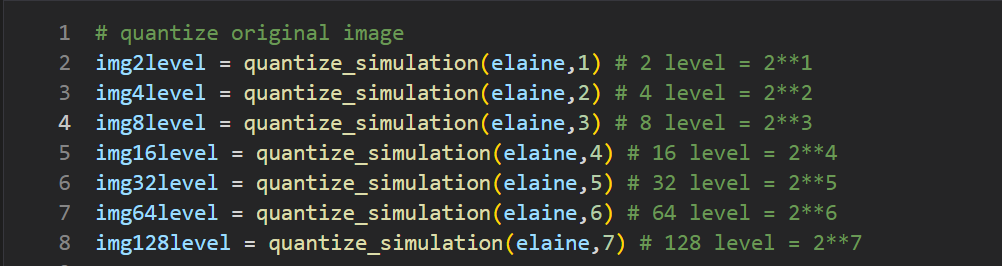
در این تابع، مقادیر pdf را به رنج بین 0 تا 1 مپ می‌کنیم. چون ماکسیمم مقدار فراوانی برابر حاصل ضرب طول در عرض تصویر است، برای تقسیم کردن در مخرج از این عدد استفاده می‌کنیم.

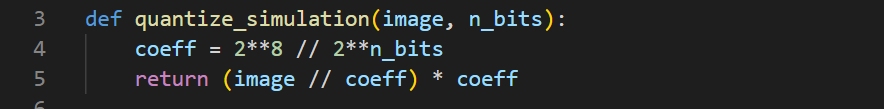


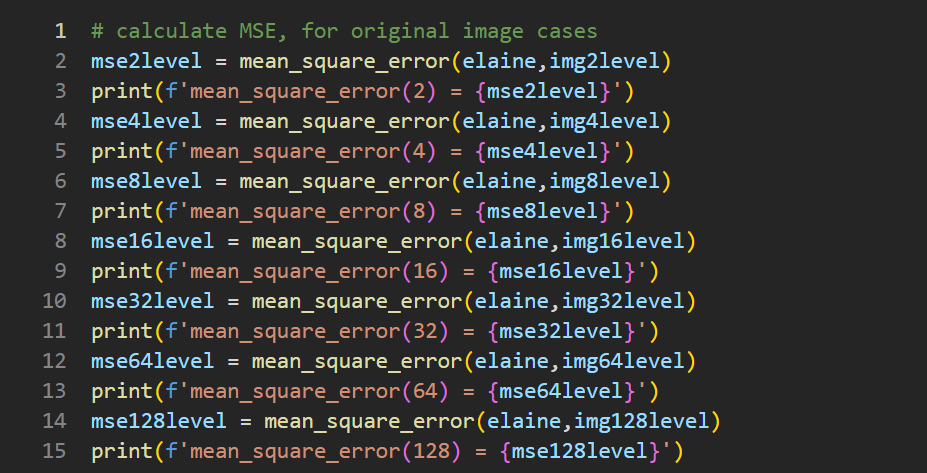
در این تابع، به ازای مقدا i ام از آرایه نرمال شده، تمام اندیس های 0 تا i از آرایه نرمال شده را با هم جمع می‌کنیم. طبق فرمول محاسبه cdf، در حلقه بیرونی، که در واقع انتهای سیگمای جمع مقادیر نرمالایز شده تا آن اندیس است، مقدار آن اندیس را در 255 ضرب می‌کنیم. در انتها چون این مقدار ممکن است یک عدد اعشاری باشد، آن را به مقدار صحیح نزدیک تر به آن گرد می‌کنیم.

 این تابع با دریافت عکس هدف و cdf مورد نظر، پیکسل های عکس را با مقادیر جدید بر اساس cdf داده شده، آپدیت می‌کند.

در ادامه، ابتدا تصویر اصلی را با لول های مختلف، از 2 تا 128 لول، نمایش می‌دهیم.

این قطعه کد، پیکسل های عکس elaine را با کمک تابع quantize\_simulation ، در لول های مشخص شده، تقسیم بندی می‌کند.

این تابع، با تقسیم256 بر ، تعداد دسته های مورد نظر را محاسبه می‌کند. سپس پیکسل های تصویر را بر این مقدار تقسیم و به پایین گرد می‌کند. در نهایت حاصل را دوباره در coeff ضرب می‌کنیم. منطق این عملیات این است که ابتدا پیکسل ها به اعدادی بین 0 تا level-1 مقداری که میخواهیم نرمالایز می‌شوند، و در ادامه دوباره در coeff ضرب می‌کنیم تا به اعداد ابتدای بازه ها مپ شوند.

در نهایت در قطعه کد بالا، mean square error را بین عکس اصلی و عکس های گسسته سازی شده محاسبه می‌کند.

در سلول های بعد، همین روال را برای عکس متعادل شده تکرار کرده ایم.

نتیجه:

در ابتدا، با مقایسه عکس اصلی و متعادل شده، با توجه به عکس های حاصل، تاثیر متعادل سازی در افزایش کنتراست تصویرو جدا شدن لبه های سیاه از سفید و تشخیص بهتر جزئیات را مشاهده می‌کنیم.

در مورد گسسته سازی(quantize)، می‌بینیم که پس از 8 یا 16 سطح، دیگر تفاوت قابل تشخیص با جشم نیست.

در مورد محاسبه تفاوت بین عکس ها، اعداد زیر به‌دست آمد:

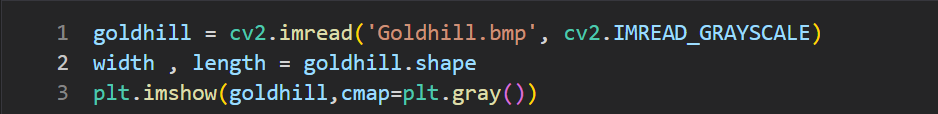
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Level** | **128** | **64** | **32** | **16** | **8** | **4** | **2** |
| **Without Histeq** | 0.5093 | 3.4688 | 17.3819 | 77.3164 | 325.0481 | 1346.5181 | 5436.5977 |
| **With Histeq** | 0.5847 | 3.9539 | 17.8455 | 76.2318 | 325.0791 | 1318.1048 | 5296.0716 |

واضح است که هرچه تعداد سطوح به ۲۵۶ نزدیکتر باشد، مقدار خطا کمتر است. در مقایسه بین حالت متعادل شده و نشده نیز، تفاوت ارور برای عکس متعادل شده با عکس های گسسته اش، فقط در تعداد سطح پایین، خطای کمتری دارد، که البته تفاوت فاحشی نیست.

۱.۱.۲

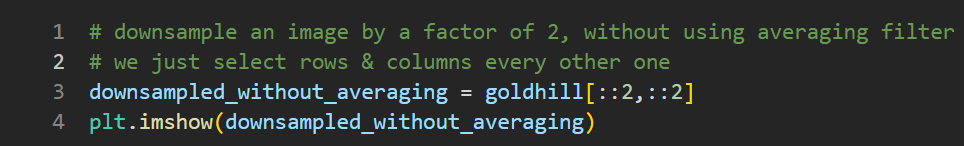
توضیح فنی:

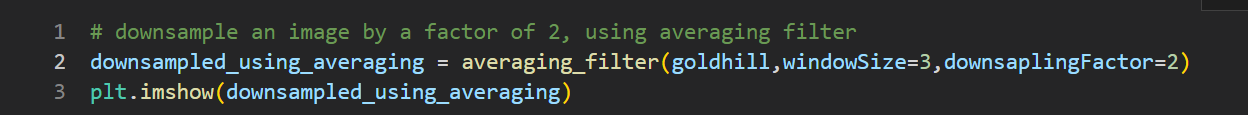
در ابتدا عکس را لود می‌کنیم.

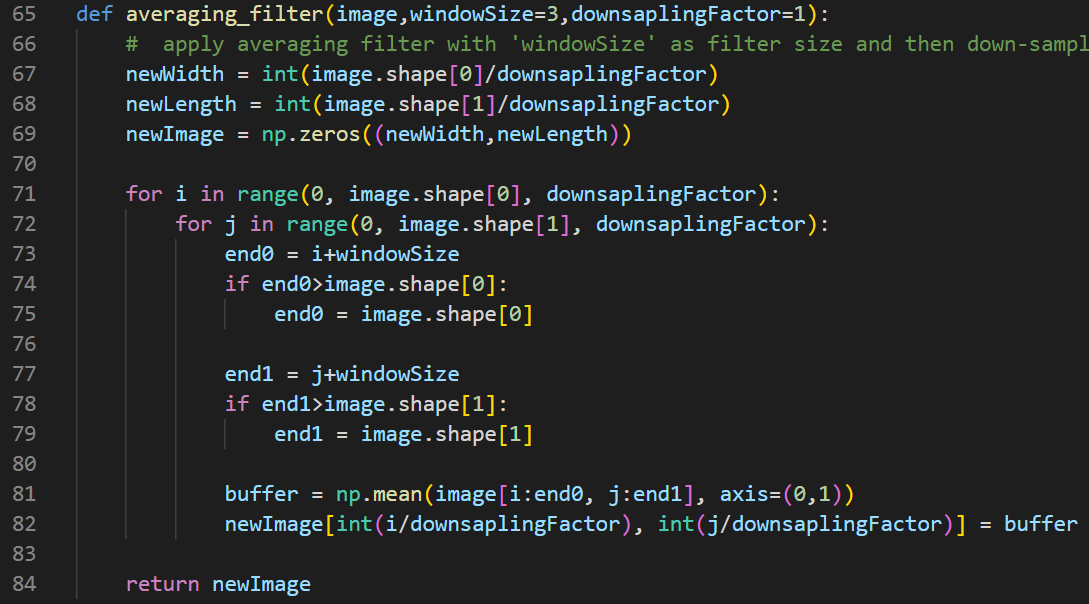


ابعاد عکس (512,512) است.

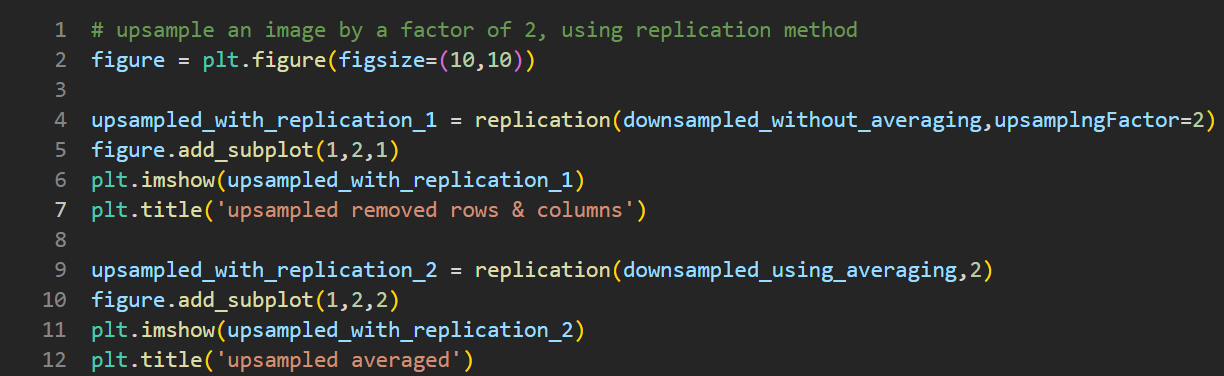
در حالت اول، عکس را با حذف سطر و ستون ها به صورت یکی در میان، کاهش ابعاد می‌دهیم.

سپس، کاهش ابعاد را با استفاده از فیلتر میانگین گیر، انجام می‌دهیم.

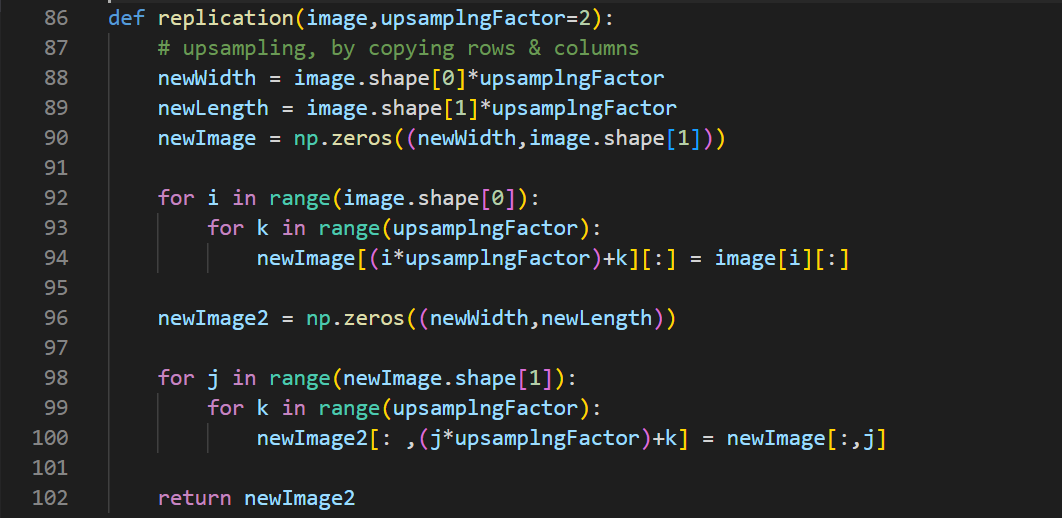
در اینجا، از تابع averaging\_filter استفاده شده است، که به شکل زیر پیاده‌سازی شده:

ورودی این تابع به ترتیب، عکس ورودی مورد نظر جهت کاهش، سایز پنجره فیلتر میانگین گیر و نرخ کاهش حجم است.

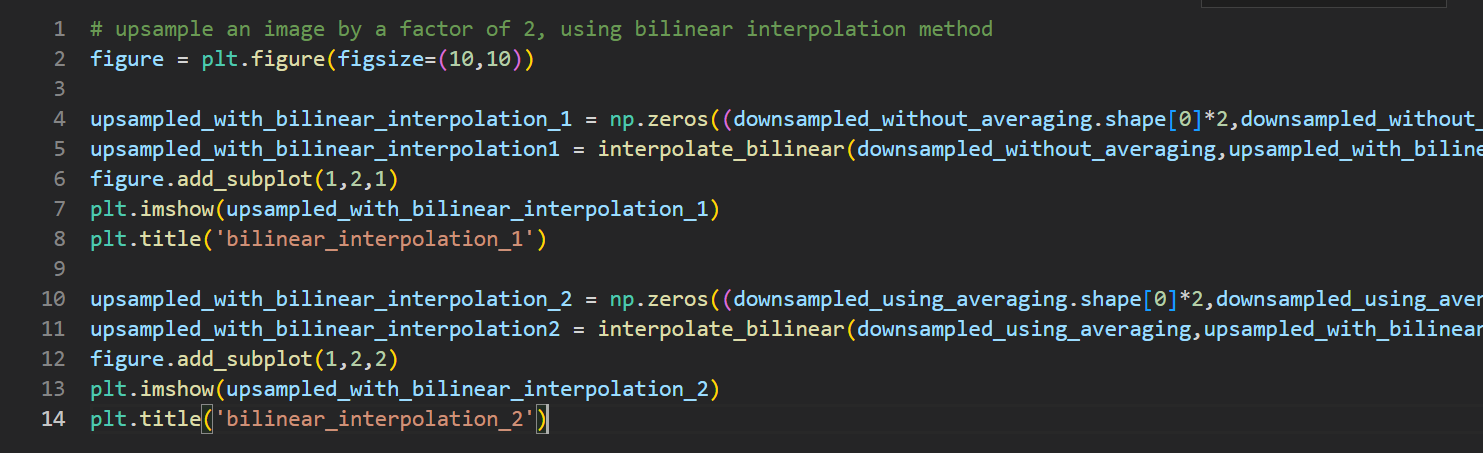
در ابتدا، یک آرایه جهت نگهداری عکس خروجی می‌سازیم. سپس در دو حلقه تو در تو، ابتدا، یک برش از عکس در ابعاد فیلتر را انتخاب می‌کنیم، سپس میانگین مقادیر پیکسل های این بخش را محاسبه می‌کنیم که در متغییر buffer ریخته می‌شود، و در انتها این مقدار را در پیکسل متناظر در آرایه عکس خروجی قرار می‌دهیم.



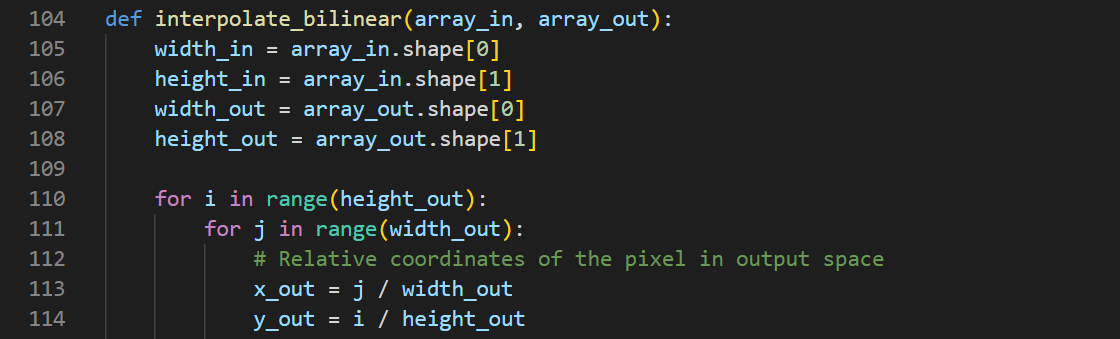
در قطعه کد بالا، با متد replication، ابعاد تصاویر را دو برابر می‌کنیم. پیاده‌سازی این متد به شکل زیر است:

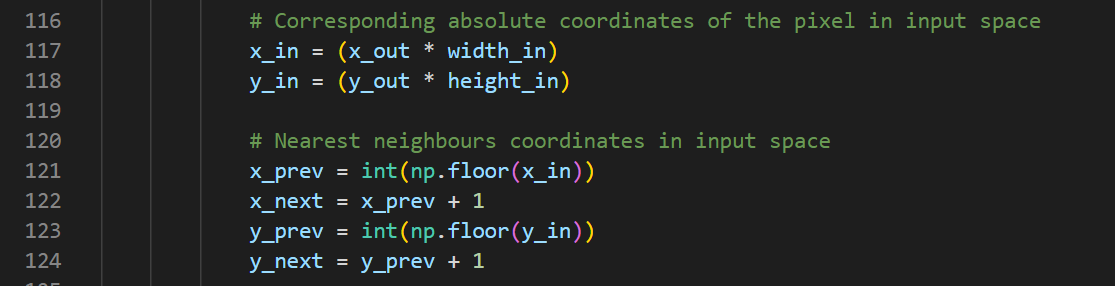
پارامتر های این تابع، تصویر ورودی برای تغییر ابعاد و نرخ تبدیل است. در این تابع، ابتدا ستون های عکس اصلی را در ستون های عکس خروجی، کپی و تکرار می‌کنیم، و سپس همین کار را برای سطرهای تصویر، تکرار می‌کنیم.

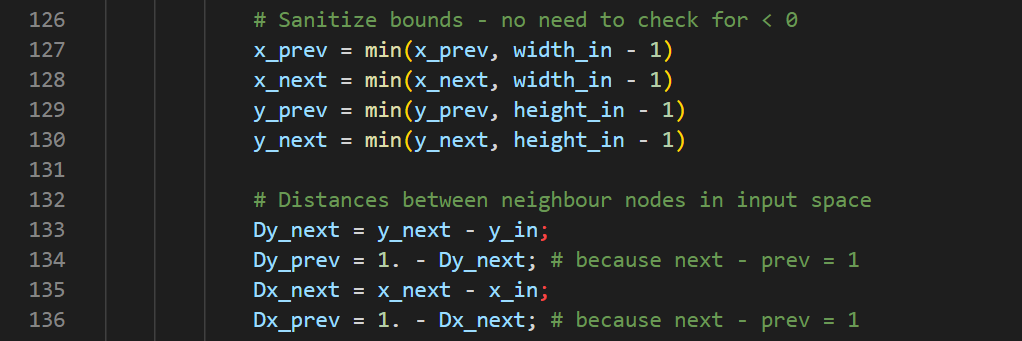
در ادامه اینبار با استفاده از متد دوخطی، افزایش ابعاد را انجام دادیم.

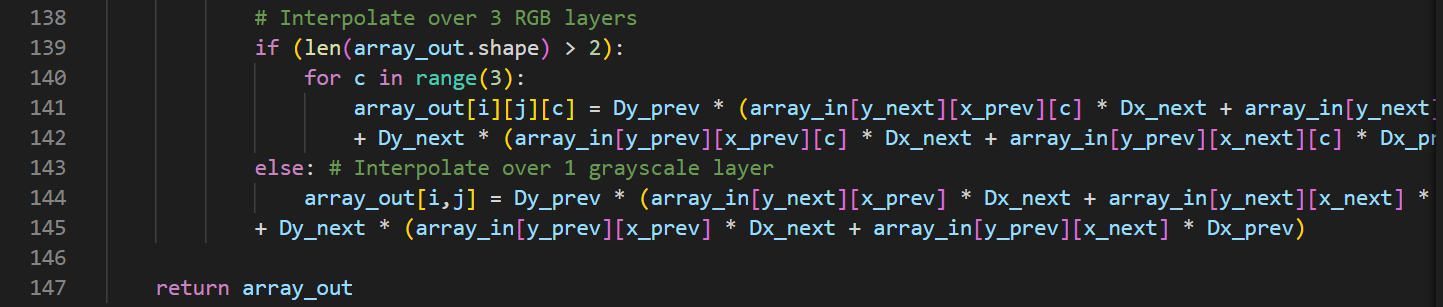


پیاده‌سازی متد interpolate\_bilinear با کمک یک منبع خارجی، به شکل زیر است.

ورودی این تابع به ترتیب، عکس ورودی و آرایه با ابعاد مدنظر برای عکس خروجی است.

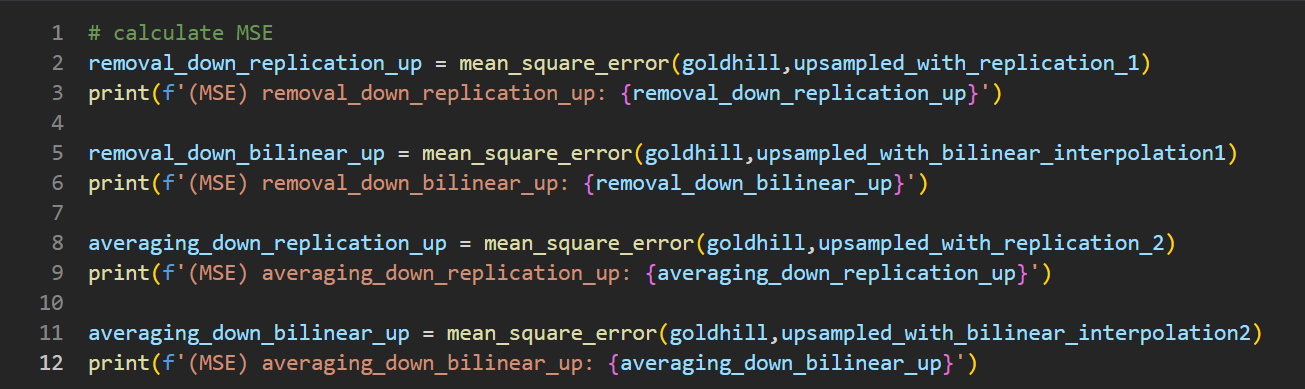






در هر بخش از کد، کامنت ها، گویای عملیات ساده ای است که رخ می‌دهد، پس توضیح اضافه ای ندارد.

در انتها، تفاوت عکس ها را با تصویر اصلی محاسبه کرده‌ایم.



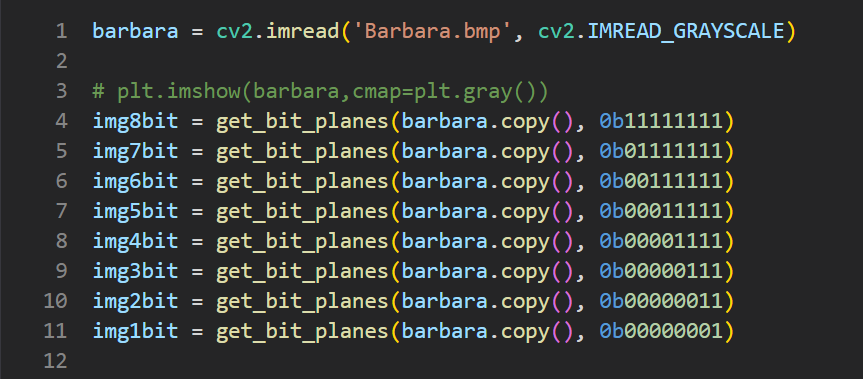
نتیجه:

|  |  |  |
| --- | --- | --- |
|  | Bilinear Interpolation | Pixel Replication |
| Averaging | 142.4705 | 99.0850 |
| Remove Row&Column | 65.0079 | 133.0759 |

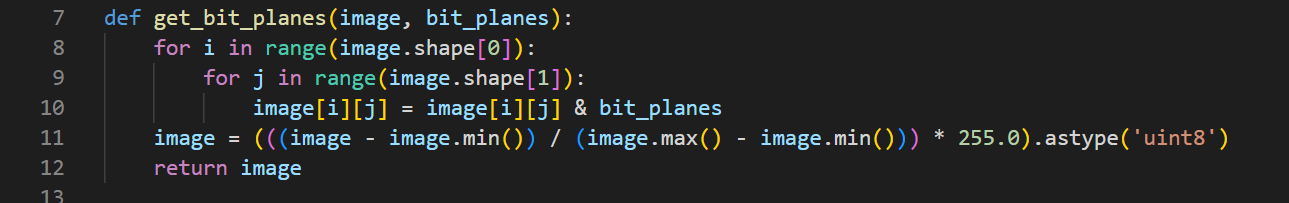
با توجه به مقادیر خطا محاسبه شده، بهترین روش برای کاهش ابعاد و سپس افزایش مجدد ابعاد یک تصویر، استفاده از روش حذف سطر و ستون و در ادامه تبدیل دوخطی است.

1.1.3

توضیح فنی:

پس از لود تصویر، با استفاده از تابع، get\_bit\_planes ، تصاویر را با تعداد بیت های به ترتیب 8 تا 1 می‌سازیم.

پیاده‌سازی تابع get\_bit\_planes به شکل زیر است:

در این تابع، رشته بیتی مدنظر را با مقدار پیکسل های تصویر، اند منطقی می‌گیریم تا فقط اطلاعات بیت های مدنظر، باقی بماند.

نتیجه:

با توجه به نتیجه عکس های خروجی، می‌توان گفت، با 4 یا 5 بیت، دیگر نمی توان اطلاعات خوبی از تصویر داشت. البته در پاسخ به سوال دوم مطرح شده، می‌توان گفت در بخش هایی که فرکانس تغییرات پایین تر است، یا به بیانی، با بخش های یکدست تصویر رو به رو هستیم، می توان تصویر را با تعداد بیت های کمتر، و با کیفیت قبل نشان داد، اما بخش هایی که فرکانس تغییرات زیادتر است، به سرعت تصویر نویزی و خراب می‌شود(حالت برفکی پیدا می‌کند)