به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش تمرین اول

دکتر فاطمی زاده

امیررضا حاتمی پور

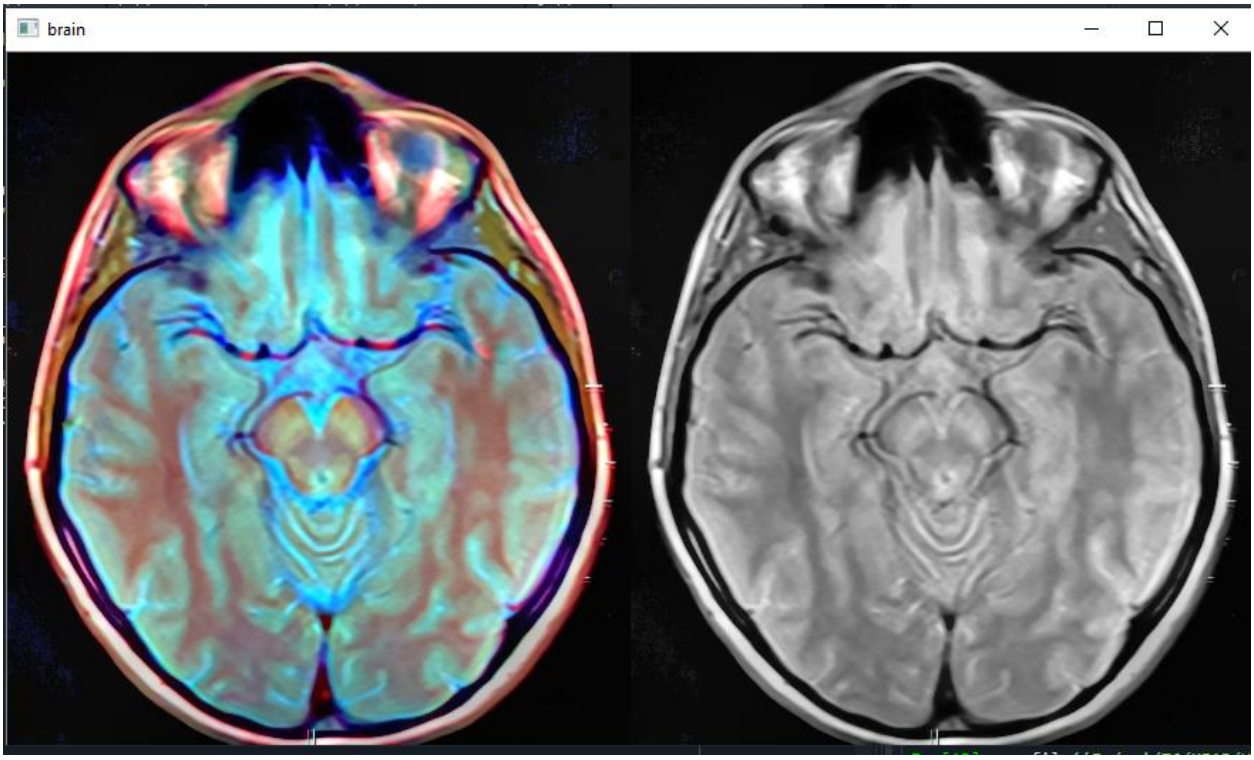
97101507

سوال اول) a)

ابتدا با استفاده از تابع cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) در پایتون تصویر مغز را به gray_scale می بریم . بعد با برای نشان دادن هر دو تصویر با هم از دستور np.concatenate استفاده می کنیم .

فقط باید به این نکته توجه شود از آنجایی که برای نمایش این دو تصویر با استفاده از np.concatenate باید هم اندازه باشن لذا تصویر gray_scaleرا به یک تصویر سیاه سفید سه کاناله تبدیل می کنیم تا بتوانیم آنرا نمایش دهیم.

نتیجه به صورت زیر می باشد:



شکل 1 : الف)سمت راست:تصویر سیاه سفید ب) سمت چپ: تصویر رنگی مغز

(b

بصورت خلاصه به بررسی کاربرد هر کدام از تایپ های مختلف تصویر می پردازیم:

Double *: این نوع تصاویر مقادیر بین 0 تا 1 را می گیرن که 0 نماد سیاه مطلق و 1 نماد سفید می باشد و بقیه gray_level ها بین این دو مقدار قرار می گیرن.*

Uint16 *: این نوع تصاویر رنج بیشتری از اعداد قرار دارند بطوری که اعداد ما بجای 0 تا 255 مه همان 8 بیت در تصاویر عادی میتونن بین 0 تا 2^{16} قرار بگیرن که خب بازه بسیار گسترده تری می باشد . در تصاویر عادی این مقدار سبب اندازه بیشتر می شود که از آنجایی که سیستم بینایی ما تفاوت انچنانی را بین این دو تا روش نمی بیند به همین دلیل معمولا از 8 بیت استفاده می شود. اما در کاربرد های پزشکی ممکن است تصاویر 16 بیتی کاربرد خاصی داشته باشن.*

uint8 *: مرسوم ترین نوع ذخیره تصویر می باشد که دارای 8 بیت و بین 0 تا 255 می باشد.اکثر مواقع برای نمایش تصاویر از این فرمت برای نمایش استفاده می شود.*

Binary *: تصاویر باینری یا تصاویر صفر و یکی بسیار دارای کاربرد فراوانی می باشند . مثلا می توان از هر نوع تصویری که داریم با توجه به ترشهولدی که برای آن ها تعیین می کنیم آنهارا به تصاویر باینری تبدیل کنیم. در خیلی از مواقع استفاده از این نوع تصاویر بسیار سرعت پردازش مارا بالا می برد.*

بطور مثال برای درست کردن ماسک از تصویر در کاربرد های مثل image blending , hybrid image و ... از تصاویر باینری استفاده می شود .

RGB *: یک شیوه بیان برای تصوایر رنکی این نوع تصاویر می باشد که هر تصویر را بر اساس اینکه در هر یک از کانال های آبی ، سبز و قرمز چه رنگ های را دارد ، رنگ تصویر را تشکیل می دهند.*

شیوه های دیگری هم برای نمایش تصاویر رنگی وجود دارد مثل Lab یا Hsv که در جاهای خاص خود کاربرد زیادی دارن

Gray_scale *: تصاویر یک کاناله می باشند که با فرمول ساده ایی از تصاویر رنگی بدست می آیند .*

در جاهای مختلف کاربردهای فراوانی دارن مثلا میتوان برای template matching تصاویر رنگی را سیاه سفید کرد و سپس پردازش را روی آن انجام داد.

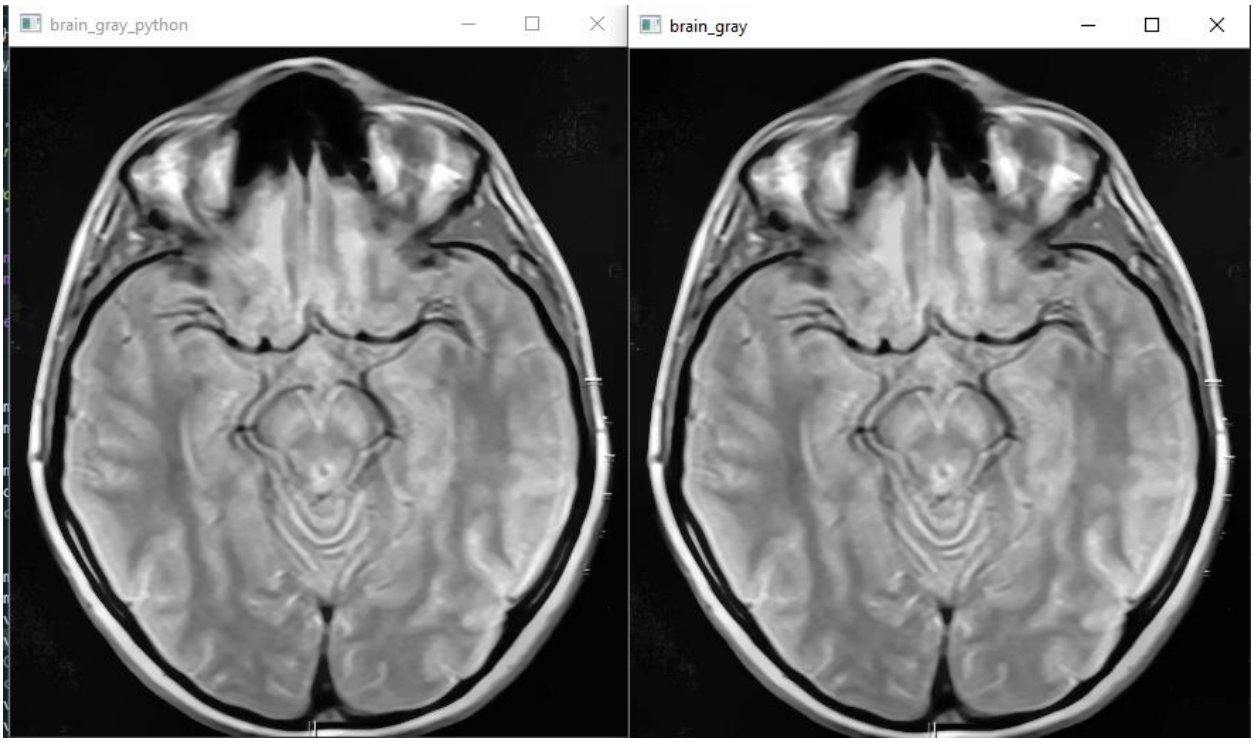
حال به پیاده سازی تابع rgb2gray در پایتون می پردازیم

میدانیم تصاویر رنگی را به استفاده از فرمول زیر می توان به سیاه سفید تبدیل کرد:(باتوجه به help متلب)

$$Grayscale=0.299 * R + 0.587 * G + 0.114 * B$$

حال با توجه به این نکته که تصاویری که توسط opencv خوانده می شوند به ترتیب BGR هستن فرمول بالا را روی کانال های مربوطه اعمال کرده و در نهایت برای اینکه قابل نمایش باشد به uint8 تبدیل می کنیم.

نتیجه حاصل از تابع خود ما و تابع آماده بصورت زیر می باشد:



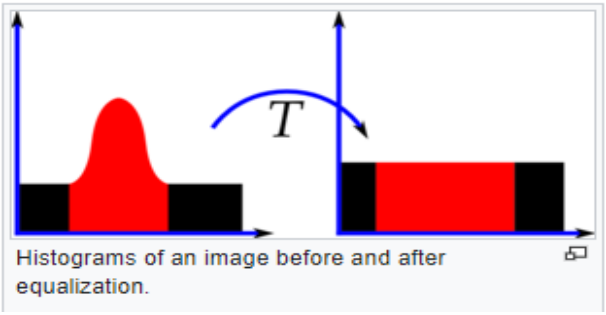
شکل 2 : الف)سمت راست:تصویر سیاه سفید مغز با تابع نوشته شده ب) سمت چپ: تصویر سیاه سفید مغز با تابع آماده پایتون

که مشاهده می شود تفاوت چندانی با هم ندارن.

توجه: می توانستیم بجای استفاده از فرمول فوق از فرمول میانگین‌گیر در سه کانال استفاده کنیم .

C)حال با استفاده از دستور cv2.imwrite تصویر حاصل را با نام Brain_MRI_grayscale.png ذخیره می کنیم. تصویر در فایل result موجود می باشد.

2) Histogram equalization :



Histograms of an image before and after equalization.

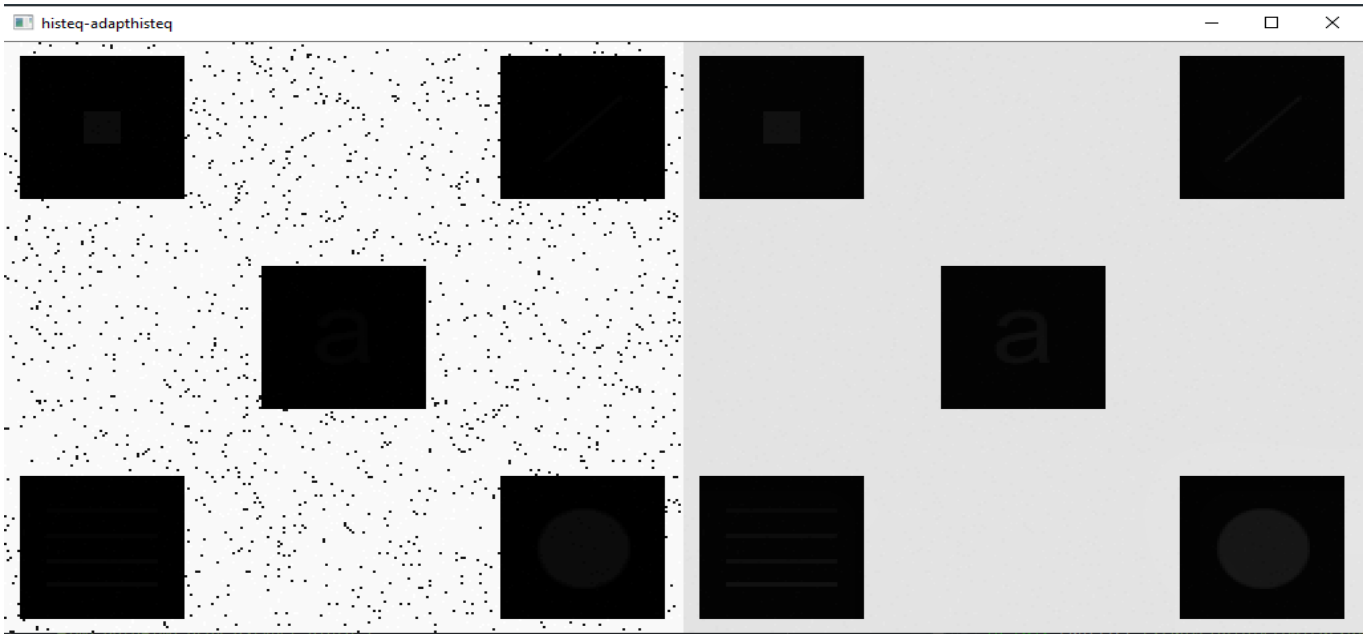
شکل3: نمودار هیستوگرام تصویر قبل و بعد از Histogram equalization

همانطور که در کلاس هم مطرح شد یکی از راه های افزایش contrast تصاویر استفاده از Histogram equalization می باشد. به این ترتیب که اگر یک هیستوگرام داشته باشیم مقدار فراوانی آن را در تمام intensity ها تقریبا برابر می کنیم.که بطور خلاصه می شود گفت که مقدار contrast را در کل تصویر در نظر می گیرد و سعی در افزایش contrast را دارد.

Adaptive Histogram equalization:

در خیلی از تصاویر اینکه کل تصویر را در نظر بگیریم و بعد آنرا برابر کنیم ایده خوبی نمی باشد. لذا میایم تصویر را بلاک های کوچک در نظر می گیریم و در هر کدام از ان بلاک های کوچک عمل equal کردن را انجام می دهیم.

حال همانطور که در خواسته شده هر دو تابع گفته شده را روی تصویر اعمال می کنیم و نتایج بصورت زیر می باشد:

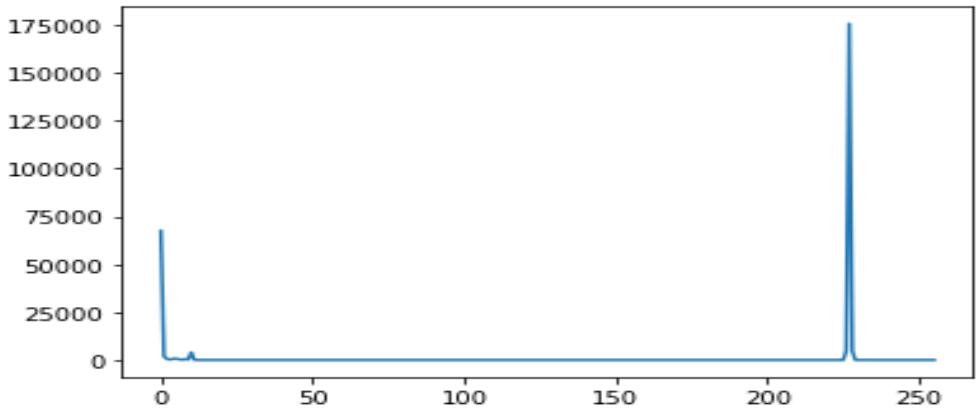


شکل 4 : الف)سمت راست:تصویر حاصل از Adaptive Histogram equalization ب) سمت چپ: تصویر حاصل از Histogram equalization

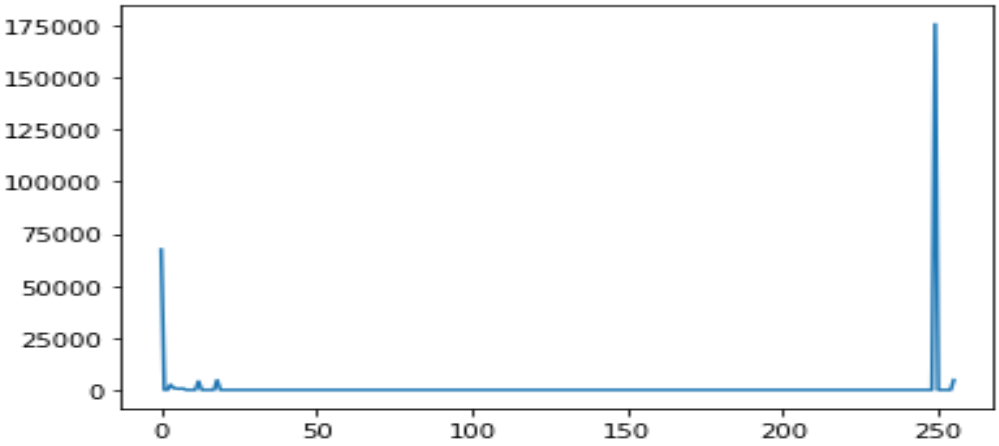
از آنجایی که تصویر سیاه سفید می باشد کافیست دو تابع cv2.createCLAHE و cv2.equalizeHist تنها در یکی از کانال های تصویر خوانده شده اعمال کنیم . اگر تصویر رنگی بود می توانستیم این کار را جداگانه و بصورت موازی برای سه کانال تصویر انجام دهیم و در نهایت با هم ترکیب کنیم.

همانطور که مشاهده می شود تصویر حاصل از Adaptive بسیار بهتر است تا تصویر حاصل از equalization که مشاهده می شود کل تصویر را خراب تر کرده است.

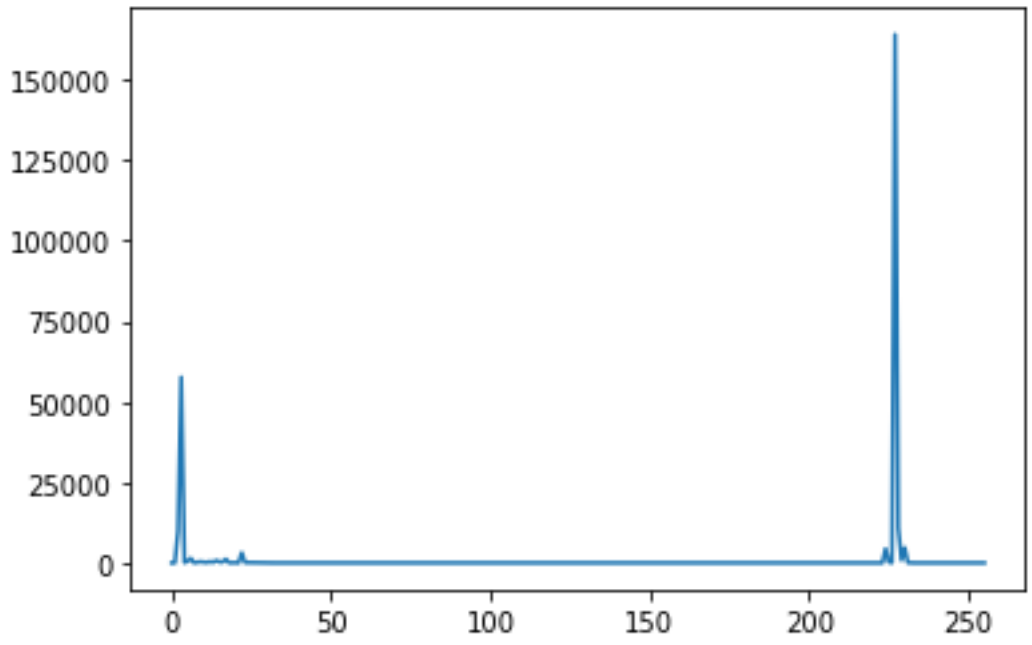
حال هیستوگرام سه تصویر را بدست می آوریم.



شکل 5 : هیستوگرام تصویر اصلی



شکل 6 : هیستوگرام حاصل از Histogram equalization



شکل 7 : هیستوگرام حاصل از Adaptive Histogram equalization

همانطور که مشاهده می شود هیستوگرام تصویر اول فقط حول دو مقدار ثابت فراوانی دارد و در بقیه نقاط صفر می باشد.

وقتی از equalization استفاده می کنیم هم مشاهده می کنیم اندکی در intensity های دیگر هم این اتفاق افتاده است.

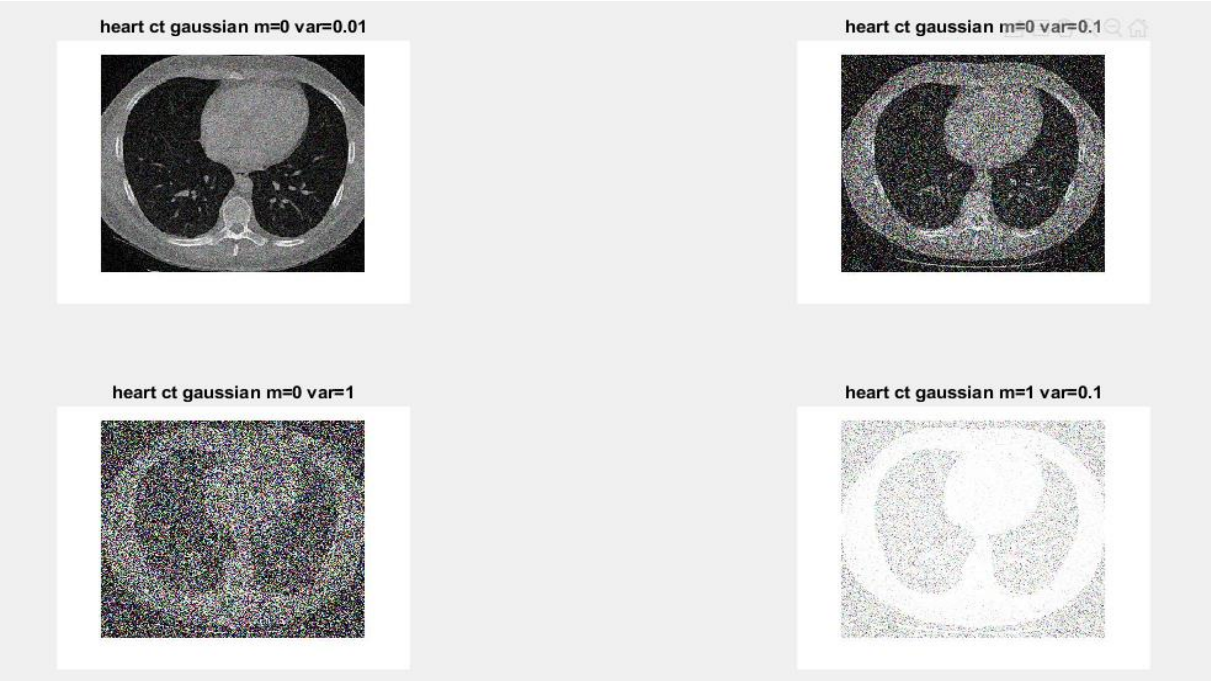
بهترین نتیجه برای adaptive بود که آنرا مشاهده می کنیم.

3) A تصویر را میخوانیم

شکل 8 : تصویر قلب پیش از پردازش

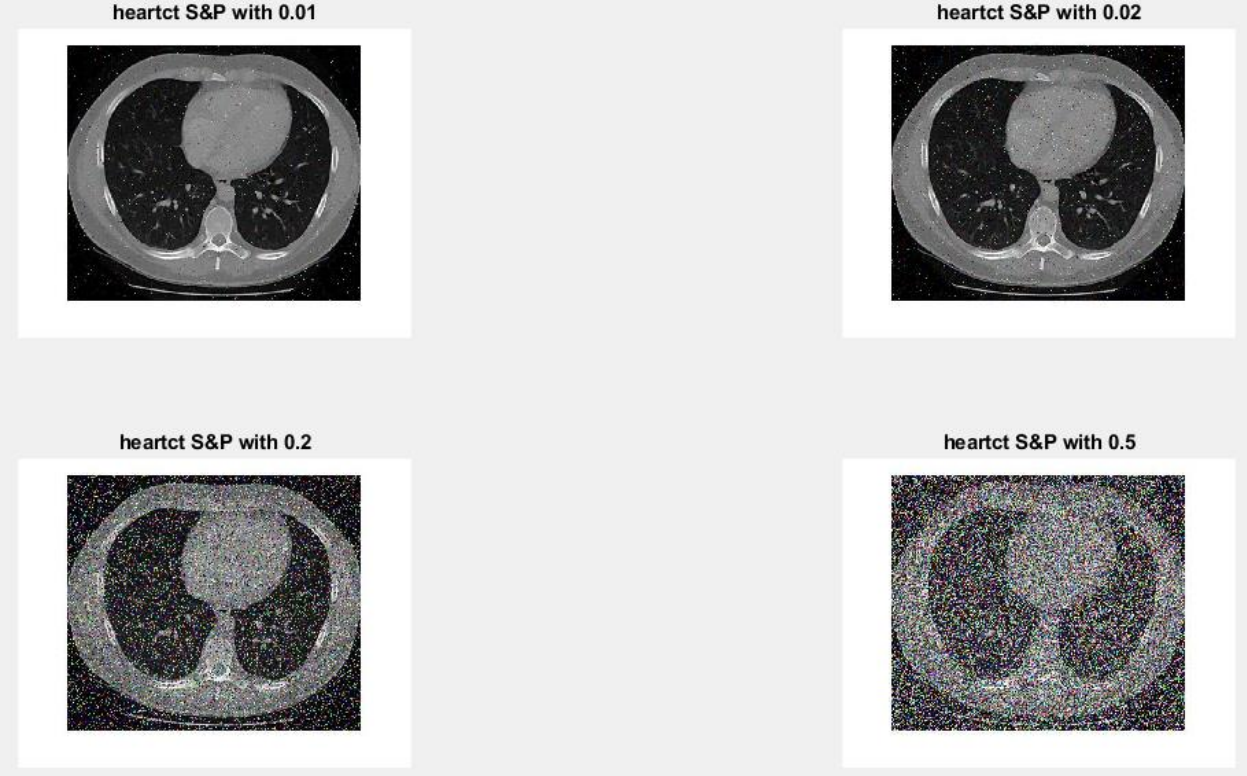
B) با استفاده از دستور imnoise نویز گاوسی و S&P را اعمال می کنیم به تصاویر و نتایج زیر بدست می آید:

ابتدا تاثیر نویز گاوسی با پارامتر های مختلف را می بینیم:



شکل9: الف)بالا سمت چپ:تصویر قلب با نویز گاوسی میانگین صفر و واریانس 0.01 . ب) بالا سمت راست:تصویر قلب با نویز گاوسی با میانگین صفر و واریانس 0.1 . ج)پایین سمت چپ:تصویر قلب با نویز گاوسی میانگین صفر و واریانس1 . د) الف)پایین سمت راست:تصویر قلب با نویز گاوسی میانگین یک و واریانس 0.1 .

حال تاثیر نویز S&P را نشان می دهیم:

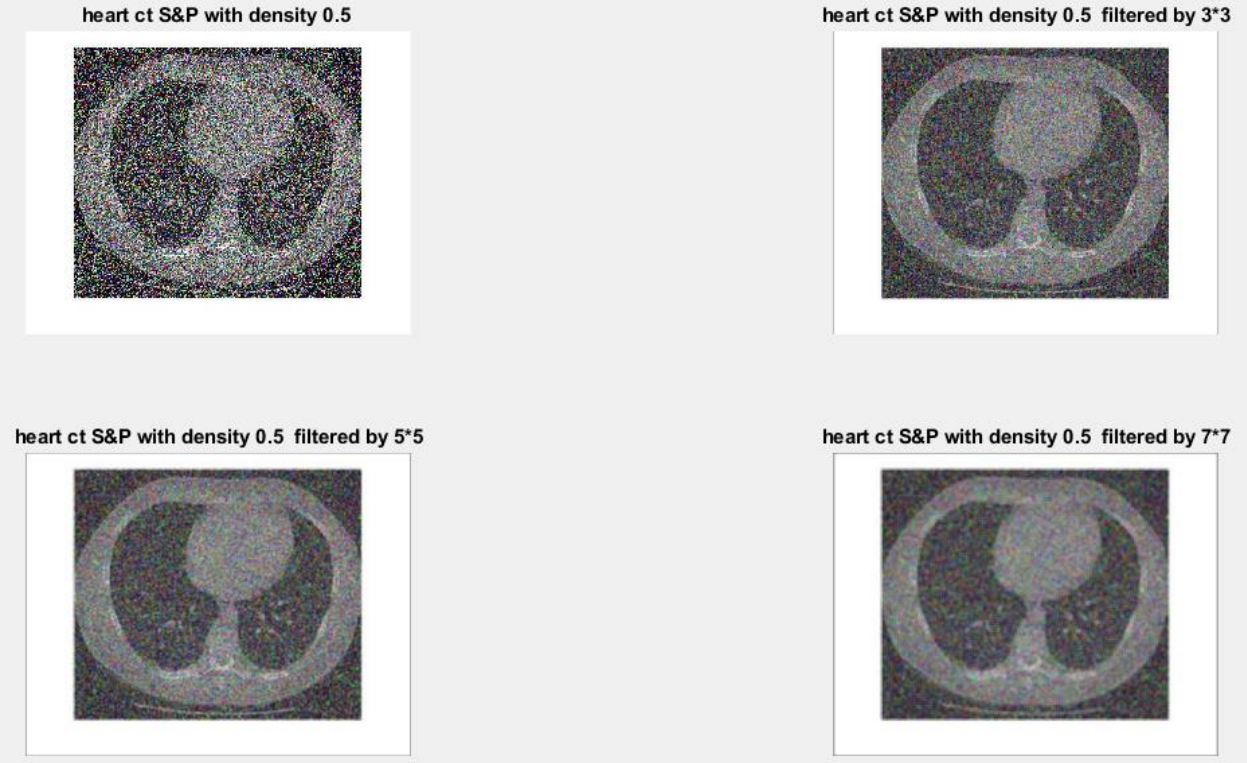


شکل10: الف)بالا سمت چپ:تصویر قلب با نویز نمک/فلفل با چگالی 0.01 . ب) بالا سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.02. ج) پایین سمت چپ: تصویر قلب با نویز نمک/فلفل با چگالی 0.2. د) الف) پایین سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.5.

C)حال روی تصاویر بالا فیلتر

سه فیلتر میانگین گیر با اندازه های گفته شده تعریف می کنیم

برای هر کدام از تصاویر داریم:

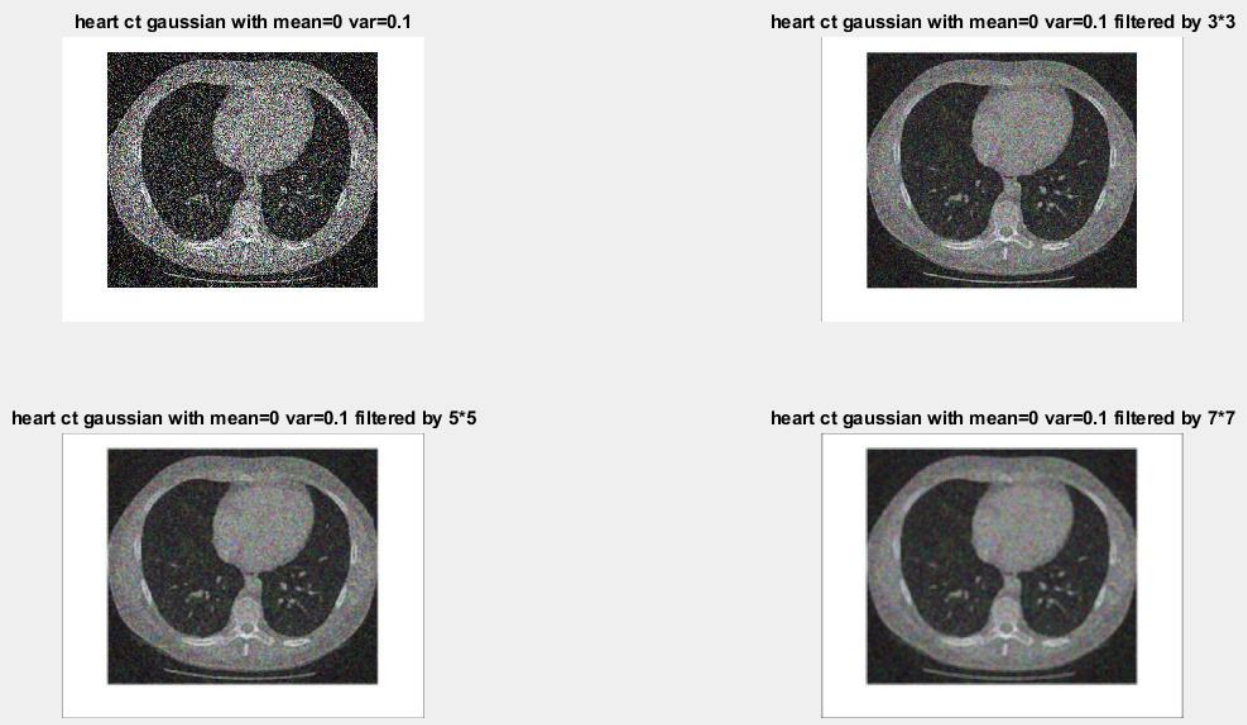


شکل 11 : الف)بالا سمت چپ:تصویر قلب با نویز نمک/فلفل با چگالی 0.5 . ب) بالا سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.5 بعداز اعمال فیلتر 3*3 ج)پایین سمت چپ: تصویر قلب با نویز نمک/فلفل با چگالی 0.5 بعداز اعمال فیلتر 5*5 .د) الف)پایین سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.5 بعداز اعمال فیلتر 7*7.

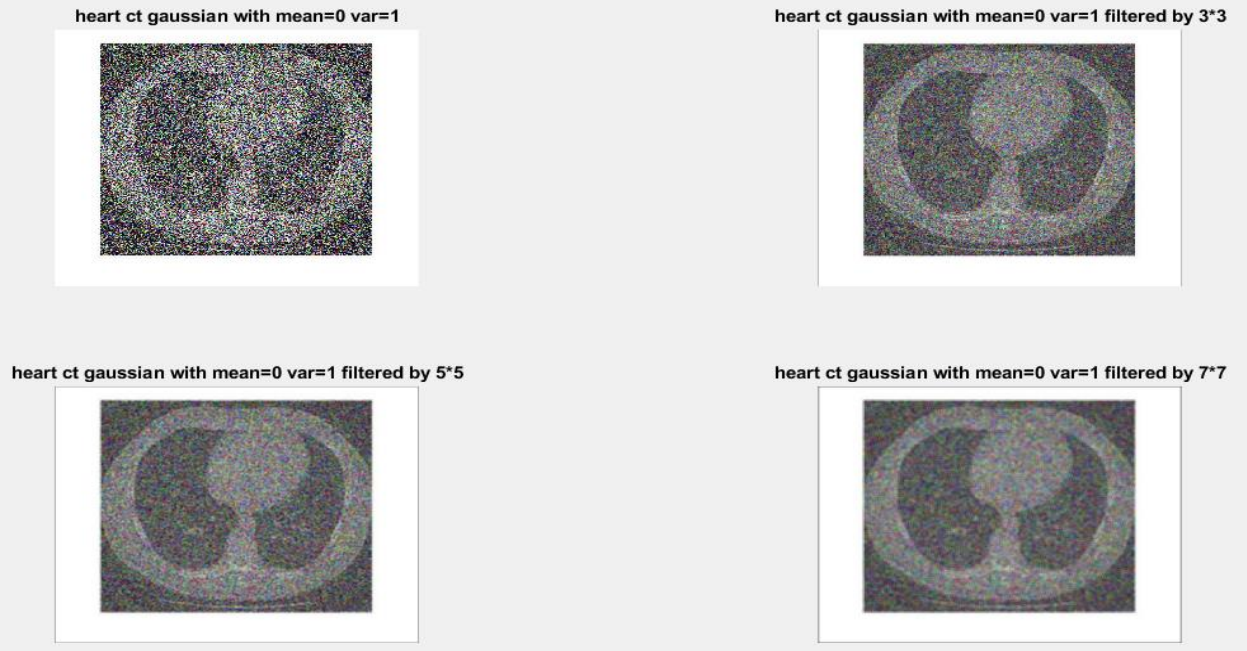
شکل 12 : الف)بالا سمت چپ:تصویر قلب با نویز نمک/فلفل با چگالی 0.2 . ب) بالا سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.2 بعداز اعمال فیلتر 3*3 ج)پایین سمت چپ: تصویر قلب با نویز نمک/فلفل با چگالی 0.2 بعداز اعمال فیلتر 5*5 .د) الف)پایین سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.2 بعداز اعمال فیلتر 7*7.

شکل 13 : الف)بالا سمت چپ:تصویر قلب با نویز نمک/فلفل با چگالی 0.02 . ب) بالا سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.02 بعداز اعمال فیلتر 3*3 ج) پایین سمت چپ: تصویر قلب با نویز نمک/فلفل با چگالی 0.02 بعداز اعمال فیلتر 5*5 .د) الف) پایین سمت راست: تصویر قلب با نویز نمک/فلفل با چگالی 0.02 بعداز اعمال فیلتر 7*7.

شکل 14 : الف)بالا سمت چپ:تصویر قلب با نویز گاوسی با واریانس 0.01 . ب) بالا سمت راست: تصویر قلب با نویز گاوسی با واریانس 0.01 بعداز اعمال فیلتر 3*3 ج)پایین سمت چپ: تصویر قلب با نویز گاوسی با واریانس 0.01 بعداز اعمال فیلتر 5*5 .د) الف)پایین سمت راست: تصویر قلب با نویز گاوسی با واریانس 0.01 بعداز اعمال فیلتر7*7.



شکل 15 : الف)بالا سمت چپ:تصویر قلب با نویز گاوسی با واریانس 0.1 . ب) بالا سمت راست: تصویر قلب با نویز گاوسی با واریانس 0.1 بعداز اعمال فیلتر 3*3 ج) پایین سمت چپ: تصویر قلب با نویز گاوسی با واریانس 0.1 بعداز اعمال فیلتر 5*5 د) الف) پایین سمت راست: تصویر قلب با نویز گاوسی با واریانس 0.1 بعداز اعمال فیلتر7*7.



شکل 16 : الف)بالا سمت چپ:تصویر قلب با نویز گاوسی با واریانس 1 . ب) بالا سمت راست: تصویر قلب با نویز گاوسی با واریانس 1 بعداز اعمال فیلتر 3*3 ج) پایین سمت چپ: تصویر قلب با نویز گاوسی با واریانس 1 بعداز اعمال فیلتر 5*5 .د) الف) پایین سمت راست: تصویر قلب با نویز گاوسی با واریانس 1 بعداز اعمال فیلتر 7*7.

در تصاویر بالا دیده میشود که اعمال فیلتر میانگین گیر در برخی از موارد می تواند در حذف نویز بسیار مفید و موثر می باشد. اما اگر نویز ما در تصویر بیشتر از یک حد باشد دیگر از این روش نمی توان برای حذف نویز استفاده کرد . همانطور که در تصویر ها می بینیم هم در نویز گاوسی و هم در نویز S&P اگر بیشتر از حد نویز وجود داشته باشد تصویر کاملا از بین رفته و دیگر قابل بازیابی نمی باشد.

حتی در مقادیر density یا variance کم میزان اثر گذاری فیلتر میانگین گیر تاثیر خوبی در نویز S&P ندارد در مقایسه بر نوع نویز گاوسی بهتر عمل می کند چون در تصاویر S&P نقاط می توانند تاثیری زیادی را روی میانگین نقاط همسایه ها بگذارن و تمام نقاط اطرافشان هم خراب کنند . پس اگر واریانس نویز گاوسی ما در حد قابل قبولی باشد می شود از این نوع فیلتر برای حذف نویز استفاده کرد. نکته دیگری که وجود دارد تاثیر اندازه فیلتر اعمالی بر روی تصویر می باشد. هرچه اندازه فیلتر بزرگ تر باشد نویز بیشتری حذف می شود اما تصویر blur تر میشود . بسته به کاربرد ما می توان این امر حائز اهمیت باشد یا خیر.

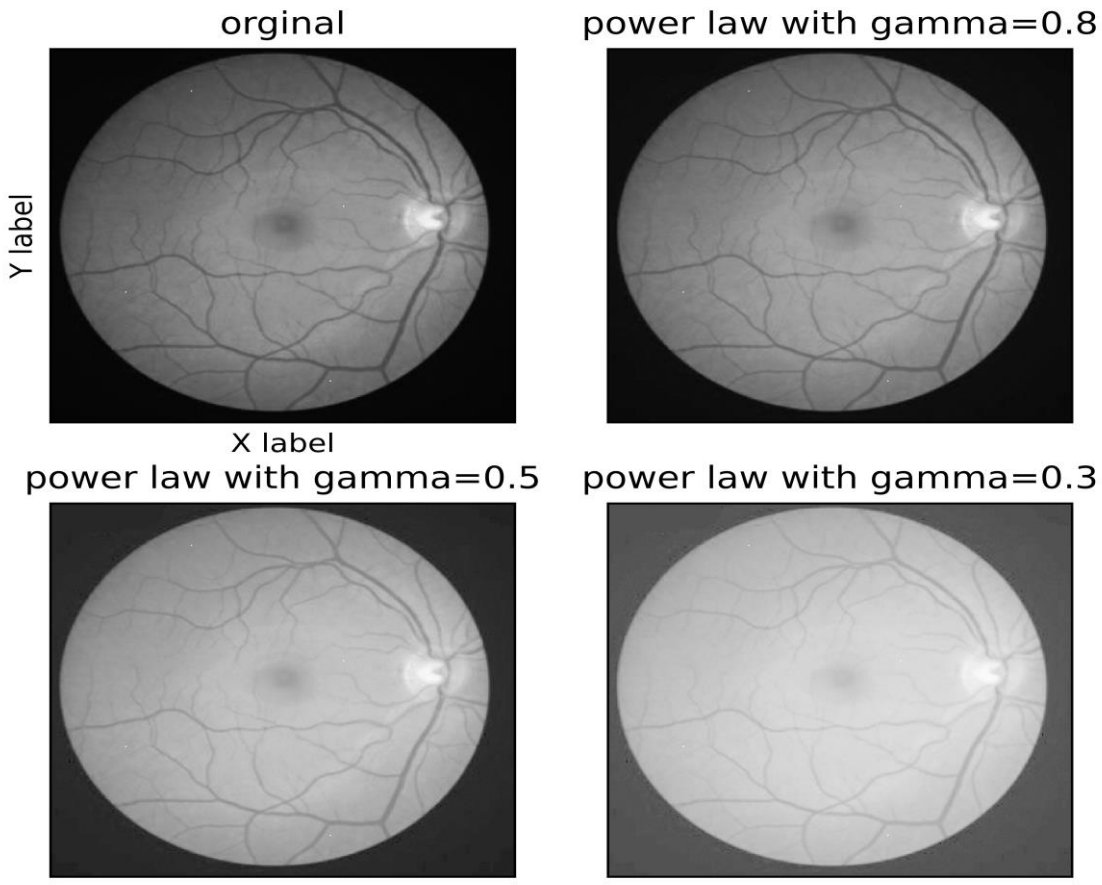
یکی دیگر از فیلتر هایی که برای حذف نویز استفاده کرد فیلتر median می باشد. که میتواند بجای میانگین گیر استفاده شود و نتیجه بهتری هم می دهد.

برای نویزهای S&P یکی از بهترین روش هایی که برای حذف آنها پیشنهاد شده است استفاده از روش Selective adaptive median filter می باشد .

4)

ابتدا تابع power_law با پارامترهای مختلف اعمال می کنیم.

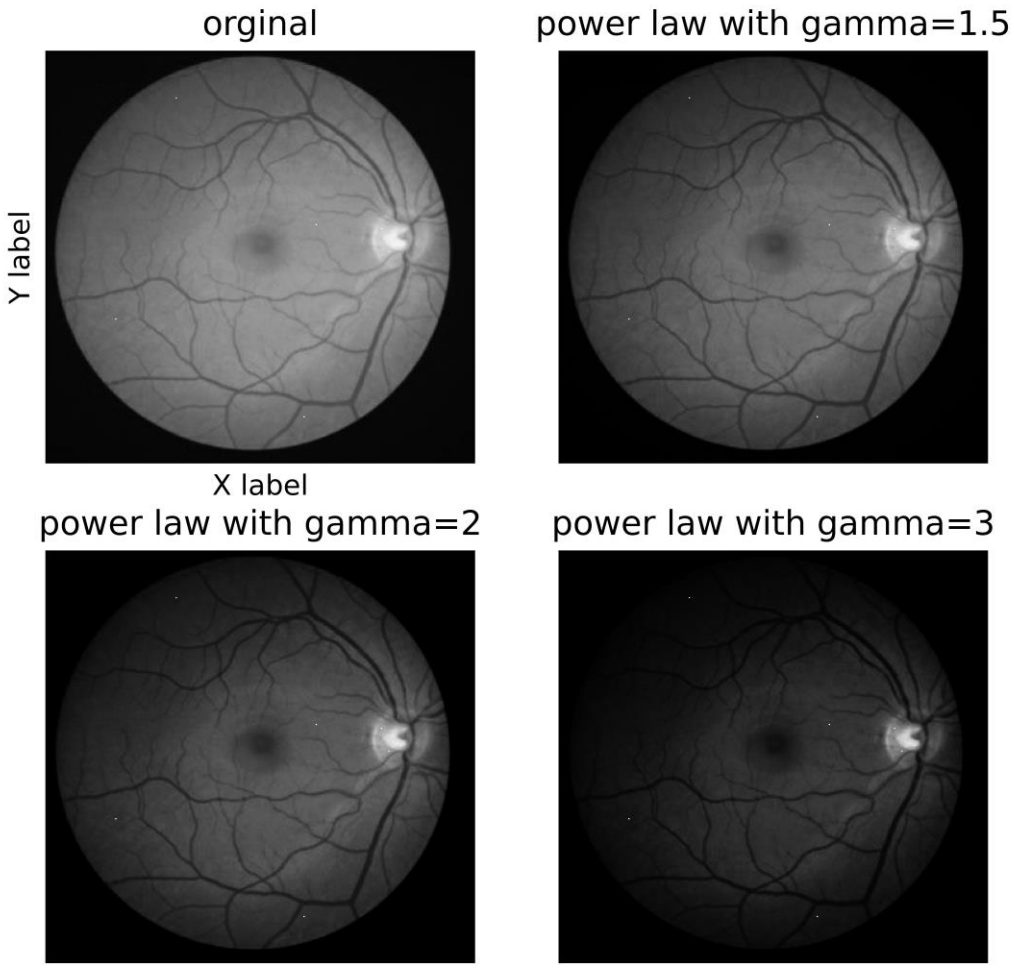
با مقادیر گاما کوچک تر از یک داریم:



شکل 17 : الف)بالا سمت چپ:تصویر اصلی چشم . ب) بالا سمت راست: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 0.8 .ج) پایین سمت چپ: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 0.5 .د) الف) پایین سمت راست: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 0.3 .

همانطور که در درس هم اشاره شد از مقادیر کوچکتر از یک گاما برای روشن کردن تصاویر تیره استفاده می شود. هنگام استفاده از این تابع برای روشن کردن تصاویر قسمن های روشن تصویر تقریبا بدون تغییر باقی می مانند اما قسمت های تیره آن روشن تر می شوند. حال براساس کاربردی که ما در نظر داریم می توانیم از گاما با مقدار مورد نیاز خودمان استفاده کینم.

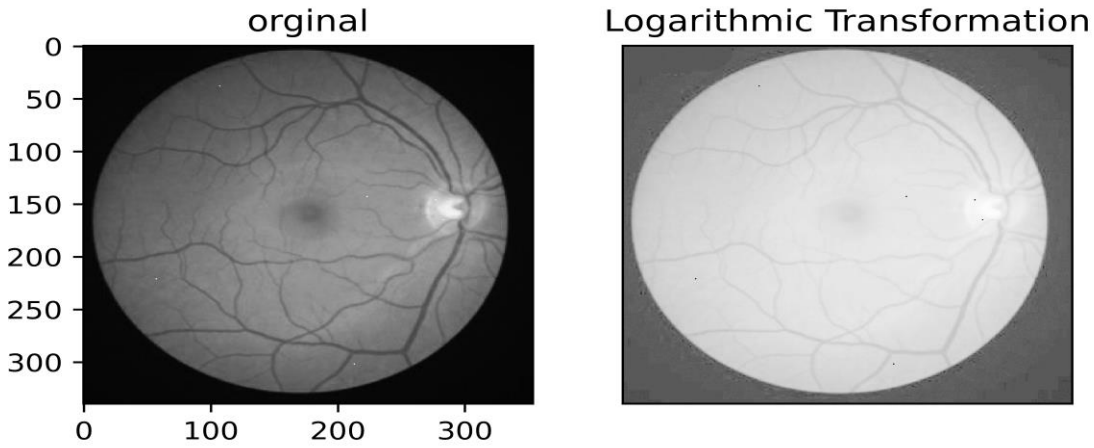
به ازای گاما های بزرگتر از یک داریم:



شکل 18: الف)بالا سمت چپ:تصویر اصلی چشم . ب) بالا سمت راست: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 1.5 .ج) پایین سمت چپ: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 2 .د) الف) پایین سمت راست: تصویر اصلی چشم بعد از اعمال تابع توانی با گاما 3

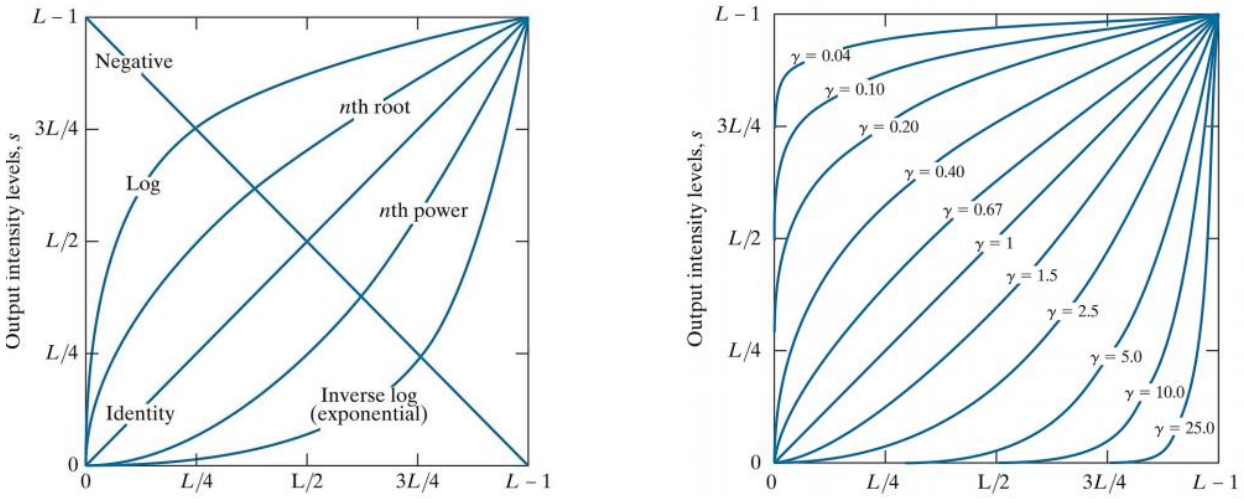
از گاما های بزرگتر از یک برای تیره کردن تصاویر استفاده می شود . باز هم میتوان برحسب کاربرد از این روش استفاده کرد.

برای تابع لگاریتمی هم داریم:



شکل 19 : الف)سمت چپ: تصویر اصلی چشم ب) سمت راست: تصویر اصلی چشم بعد از اعمال تابع لگاریتمی

با توجه به نمودار زیر می توان گفت که با اعمال تابع لگاریتمی پیکسل هایی که در ناحیه تاریک قرار دارن(بین 0 تا L/4) روشن تر می شوند.

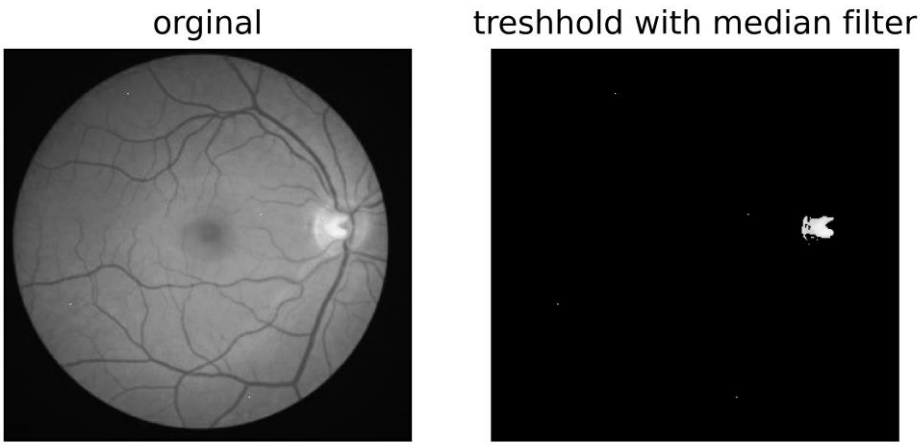


شکل 19 : الف)سمت چپ: نمودار تابع توانی ب) سمت راست: نمودار تابع لگاریتمی

در نهایت باز هم بسته به اینکه چه نوع پردازشی را می خواهیم روی تصویر انجام بدهیم ممکن است هرکدام از این روش ها کاربردی باشن .

5)

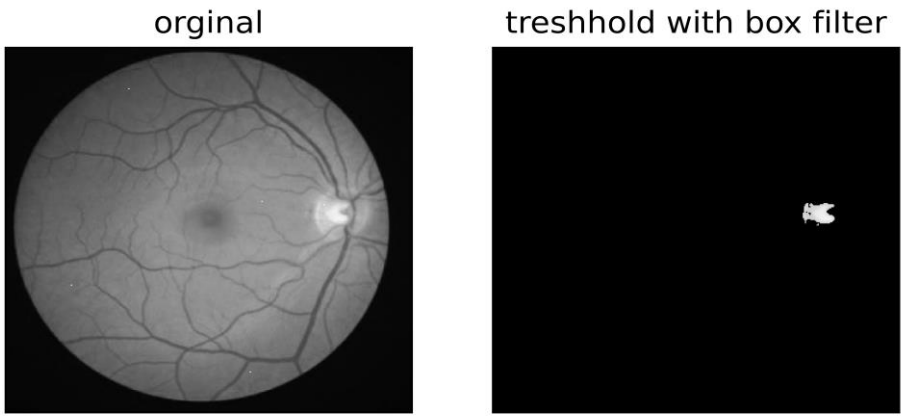
ابتدا تصویر را می‌خوانیم. اگر بدون حذف نویز آستانه گذاری را انجام دهیم شکلی مانند زیر بدست می آید:



شکل 20 : الف)سمت چپ: تصویر اصلی چشم ب) سمت راست: تصویر اصلی چشم بعد از اعمال حدآستانه بدون حذف نویز

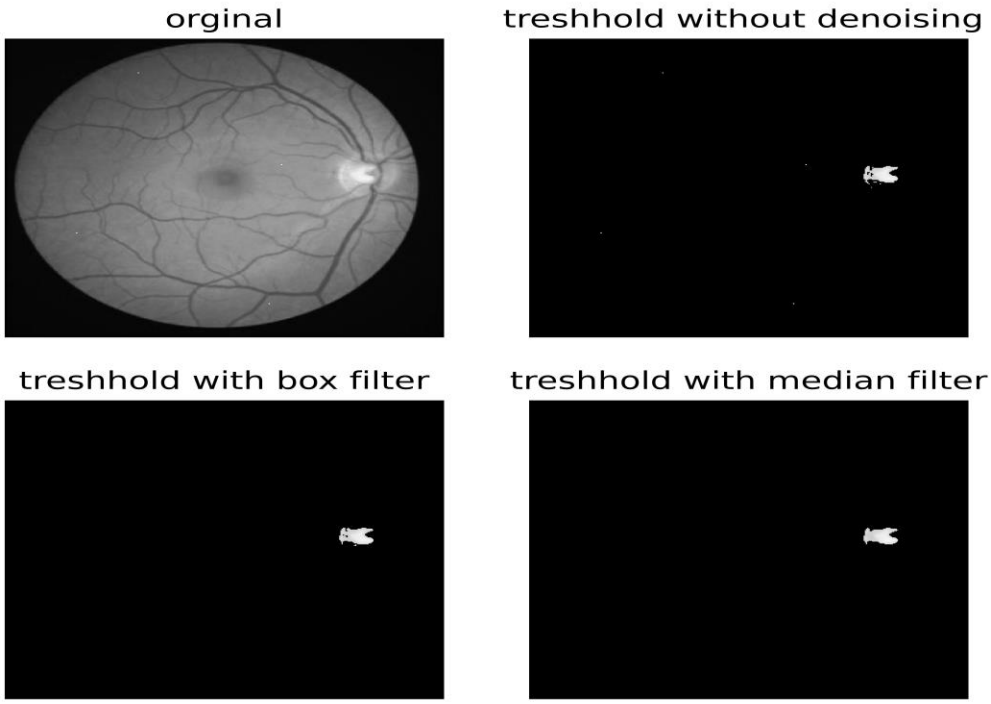
مشاهده می کنیم به دلیل وجود نویز اندکی خطا داریم. چون مقدار نویز موجود بسیار کم می باشد پس می‌توانیم با اعمال فیلتر میانگین گیر به خواسته سوال برسیم.

*یک 3*3 box filter را روی تصویر اعمال می کنیم . نتیجه زیر بدست می آید:*



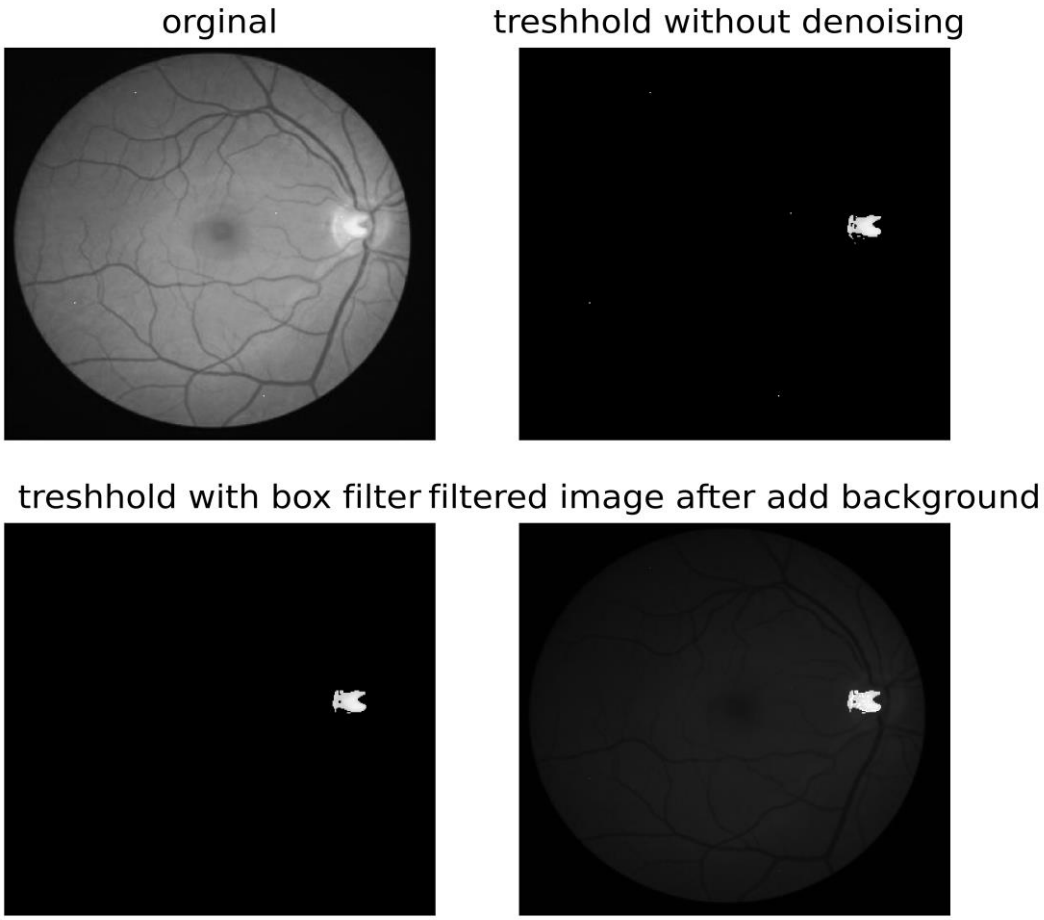
شکل 21 : الف)سمت چپ: تصویر اصلی چشم ب) سمت راست: تصویر اصلی چشم بعد از اعمال حدآستانه با اعمال Box filter

که تمام نویز های موجود در تصویر حذف شده است . می توانستیم بجای این فیلتر از فیلترهای median یا گاوسی یا میانگین گیر وزن دارهم استفاده کنیم اما به دلیل ساده بودن و کم بودن نویز استفاده Box filter مناسب می باشد.



شکل 22: الف)بالا سمت چپ:تصویر اصلی چشم . ب) بالا سمت راست تصویر اصلی چشم بعد از اعمال حدآستانه بدون حذف نویز .ج)پایین سمت چپ: تصویر اصلی چشم بعد از اعمال حدآستانه با اعمال Box filter د)پایین سمت راست: تصویر اصلی چشم بعد از اعمال فیلتر Median

حال بک گراند تصویر را هم به همان تصویر فیلتر شده با box filter اضافه می کنیم:



شکل 23: الف)بالا سمت چپ:تصویر اصلی چشم . ب) بالا سمت راست تصویر اصلی چشم بعد از اعمال حدآستانه بدون حذف نویز .ج) پایین سمت چپ: تصویر اصلی چشم بعد از اعمال حدآستانه با اعمال Box filter د) پایین سمت راست: تصویر اصلی چشم بعد از اعمال فیلتر Box filter و اضافه کردن بک گراند

که تصویر آخر همان خواسته سوال می باشد.

منابع:

1. https://towardsdatascience.com/image-filters-in-python-26ee938e57d2
2. https://www.geeksforgeeks.org/python-intensity-transformation-operations-on-images/
3. https://medium.com/fullstackai/why-is-plotting-figures-so-difficult-in-python-b3754f5d4c60