

بسمه تعالیٰ



دانشگاه صنعتی شریف
دانشکده مهندسی برق

دکتر کربلایی آقاجان

گزارش تمرین سری پنجم متاب سیگناال و سیستم

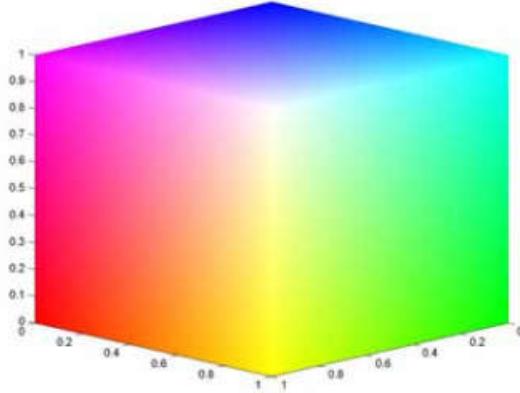
طاهها انتصاری

95101117

سوال اول

تصاویر رنگی در متلب به صورت ماتریس های $M \times N \times 3$ ذخیره می شوند که در آن بعد سوم همان نمایانگر ویژگی های رنگی و RGB است. که درایه i در هر یک از 3 صفحه i بعد سوم نمایانگر قسنتی از رنگ این پیکسل می باشد.

هر درایه از این ماتریس عددی بین 0 تا 1 است که با توجه به شکل زیر، با مشخص شدن 3 عدد برای هر پیکسل رنگ آن به طور یکتا مشخص می شود. صفحه ای اول برای رنگ قرمز(R)، صفحه ای دوم برای رنگ سبز(G) و صفحه ای سوم برای رنگ آبی(B) است.



RGB مکعب

در عکس بالا نقطه $(0,0,0)$ نمایانگر رنگ سیاه و نقطه $(1,1,1)$ نمایانگر رنگ سفید است. بقیه رنگ ها نیز به صورت نقطه ای در این فضا مشخص می شوند.

أنواع نویزها

- نویز ضربه (Impulse noise) ، که بیشتر در صوت مشهود است، به تغییر شدید و لحظه ای، تقریباً شبیه تابع ضربه، می گویند. این نوع از نویز در اثر آزاد شدن مقدار زیادی انرژی در مدت زمان کم به وجود می آید مثل انفجار بمب. برای از بین بردن این نوع نویز میتوان از فیلتر "میانه" استفاده کرد.

- نویز جمع شونده (Additive noise) نویز را در اصل می توان به دو دسته تقسیم کرد. دسته اول نویز های جمع-شونده و دسته دوم نویز های ضرب شونده. معروف ترین نویز در این حوزه AWGN و یا همان نویز سفید گوسی جمع شونده است. اصل این نویز، قضیه حد مرکزی

در احتمالات است چرا که بنابراین قانون حد مرکزی جمع تعداد زیادی توزیع احتمال، توزیع گاووسی دارد. با توجه به این قضیه، نویز AWGN را می‌توان مجموعی از تمامی نویزهای موجود از گرمای زمین گرفته تا اثر امواج خورشید دانست.

Salt-and-Pepper noise -

این نوع نویز که تحت عنوان impulse noise نیز شناخته شده است، تحت تغییرات سریع و زیاد در سیگنال تصویر به وجود می‌آید. اثر این نویز مثل ذرات نمک و فلفل بر روی عکس می‌باشد. از روش‌های کاهش این نویز فیلتر میانه است.



Shot noise (Poisson noise) -

نوعی نویز که اثر آن را می‌توان با فرایند پواسون مدل کرد. این نویز از ذات کوانتیده بودن ذرات نور و فوتون‌ها نشات می‌گیرد. این نوع نویز بیشتر وقتی که شدت نور کمتر است نمایان می‌شود چرا که با توجه به نکته گفته شده، در این حالت فوتون‌های محدود نمی‌توانند عکس را به طور کامل نشان دهند.



در عکس بالا با افزایش نور در تصویر اثر نویز از بین می‌رود.

Speckle noise (Multiplicative noise) -

همانطور که گفته شد نویز را می‌توان به دو دسته جمع‌شونده و ضرب‌شونده تقسیم کرد که نویز ذکر شده از نوع دوم هست. این نویز در اثر اختلال سیگنال بازتابی از سطح سخت در ورودی لنز دوربین است. اثر این نوع نویز به صورت نقاط سیاه و سفید در تصویر نمایان می‌شود.

انواع فیلتر ها

Linear smoothing filters -

این دسته از فیلترها سعی در ذخیره الگوهای اصلی شکل، و در عین حال حذف نویز و یا تغییرات سریع هستند. از آنجایی که لبه‌های شکل جزو تغییرات سریع به حساب می‌آیند این فیلترها اگر با شدت زیادی به شکل اعمال شوند لبه‌ها را "نرم" می‌کنند

از جمله این فیلترها می‌توان فیلتر گاوی را نام برد. فیلترهای خطی نرمکننده دسته‌ای از این فیلترها است که رابطه بین خروجی و داده‌ی اولیه به صورت تبدیلی خطی باشد.

Wiener filter -

این نوع فیلتر با استفاده از کمینه کردن مریع اختلافات (MMSE)، با فرض ثابت بودن سیگنال و اثر نویز و همچنین این فرض که نویز به صورت جمع‌شونده بوده، تخمینی از سیگنال اصلی ارائه می‌دهد. این فیلتر سعی در ارائه تخمینی آماری از سیگنال اصلی دارد. البته برای استفاده از این فیلتر بایستی اطلاعاتی از حوزه فرکانسی سیگنال اصلی داشته باشیم.

Median filter -

اساس این الگوریتم، جایگزینی هر خانه با میانه‌ی داده‌های همسایه است. داده‌های همسایه بر اساس پنجره‌ای دلخواه تعریف می‌شوند. همانگونه که مشهود است، این نوع فیلتر، فیلتری غیرخطی است. از ویژگی‌های مهم این فیلتر، حفظ لبه‌های تصویر است

در ادامه عکس‌ها و نویزهای اعمال شده و نتایج فیلترینگ آمده‌است

عکس اول:



Original image



Uniform additive noise



Gaussian noise



Poisson multiplicative noise



Salt & pepper noise



Speckle noise



Gaussian noise



Median filter applied



Gaussian filter applied



Wiener filter applied



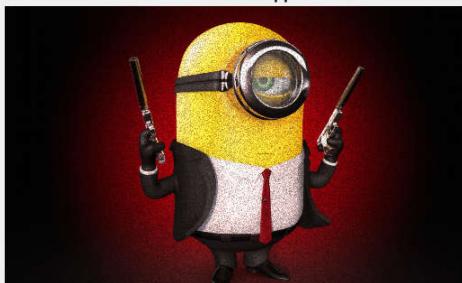
Poisson noise



Median filter applied



Gaussian filter applied



Wiener filter applied



Salt & pepper noise



Median filter applied



Gaussian filter applied



Wiener filter applied



Speckle noise



Median filter applied

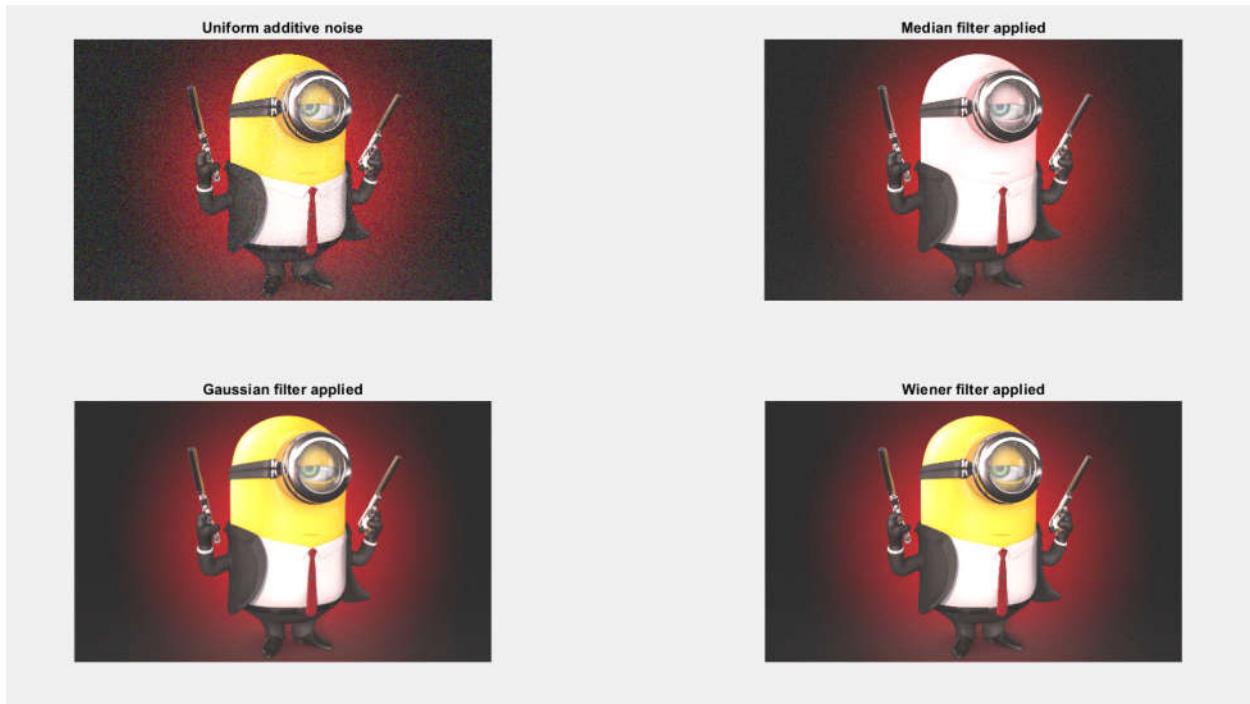


Gaussian filter applied



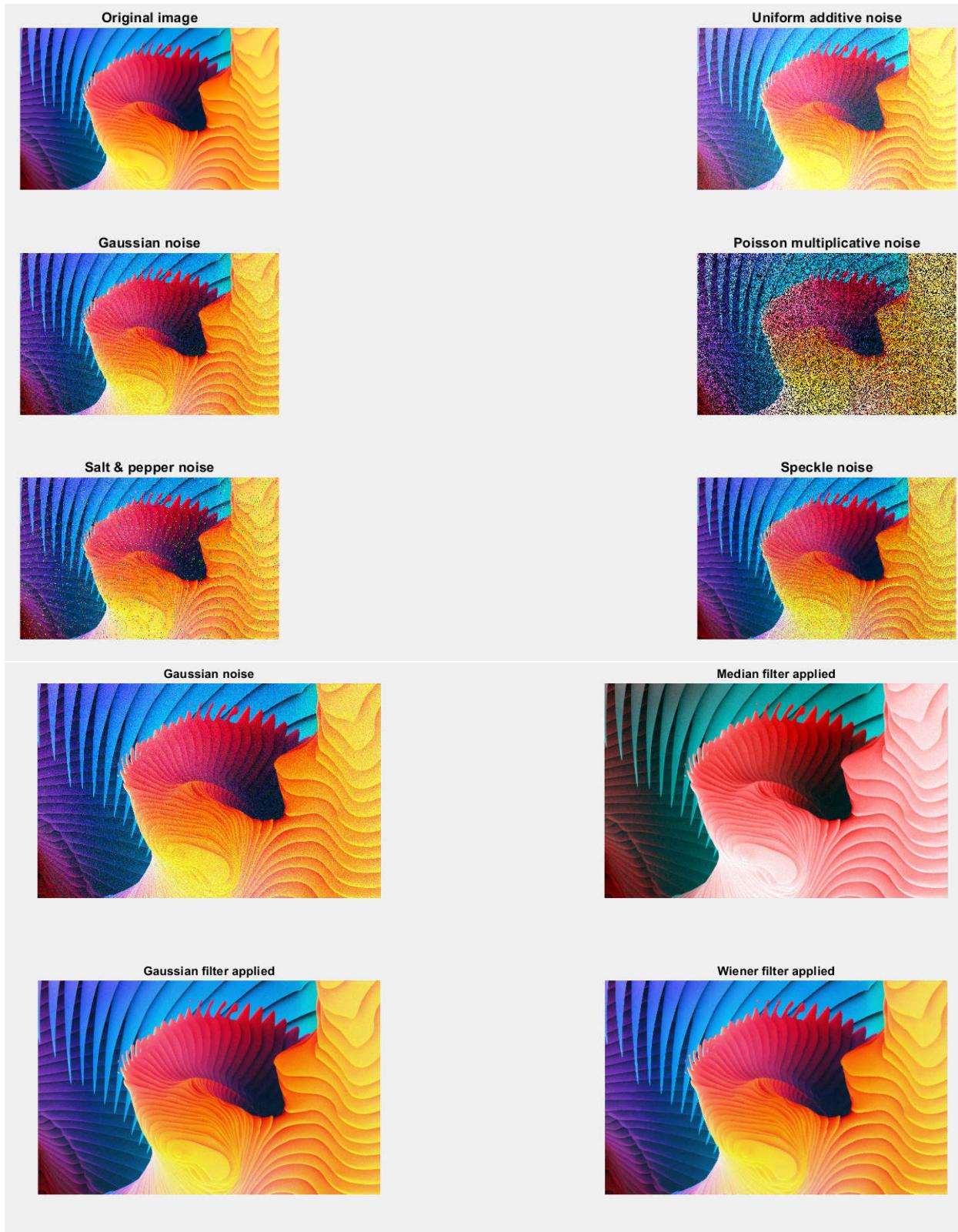
Wiener filter applied



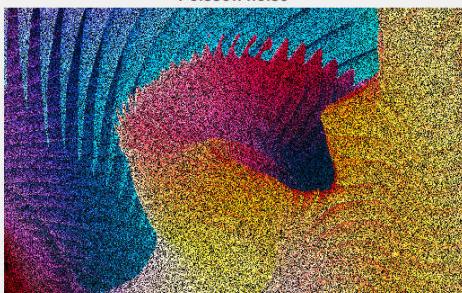


عکس دوم:

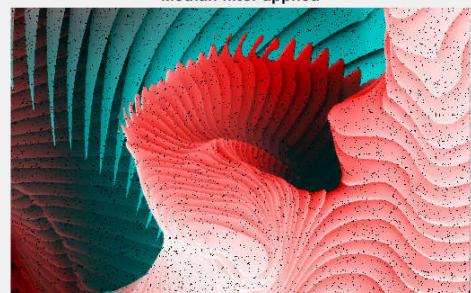




Poisson noise



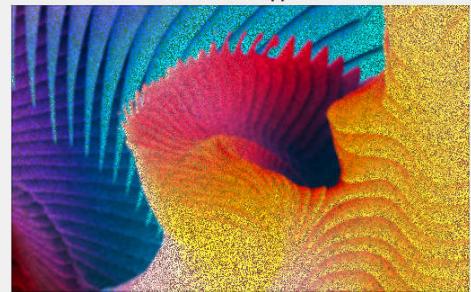
Median filter applied



Gaussian filter applied



Wiener filter applied



Salt & pepper noise



Median filter applied



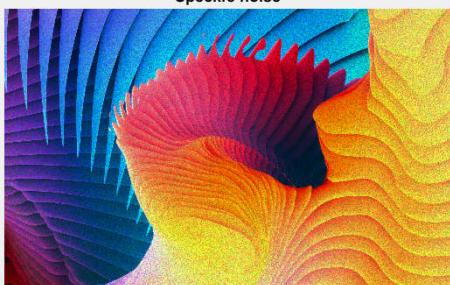
Gaussian filter applied



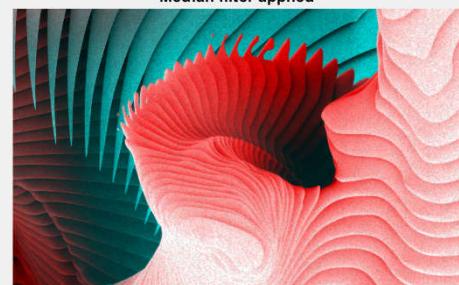
Wiener filter applied



Speckle noise



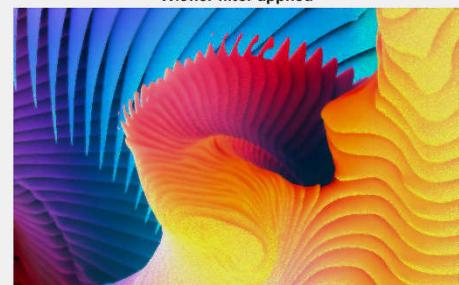
Median filter applied



Gaussian filter applied



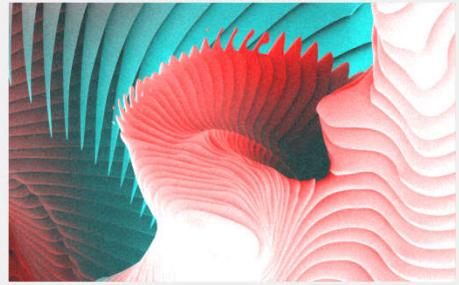
Wiener filter applied



Uniform additive noise



Median filter applied



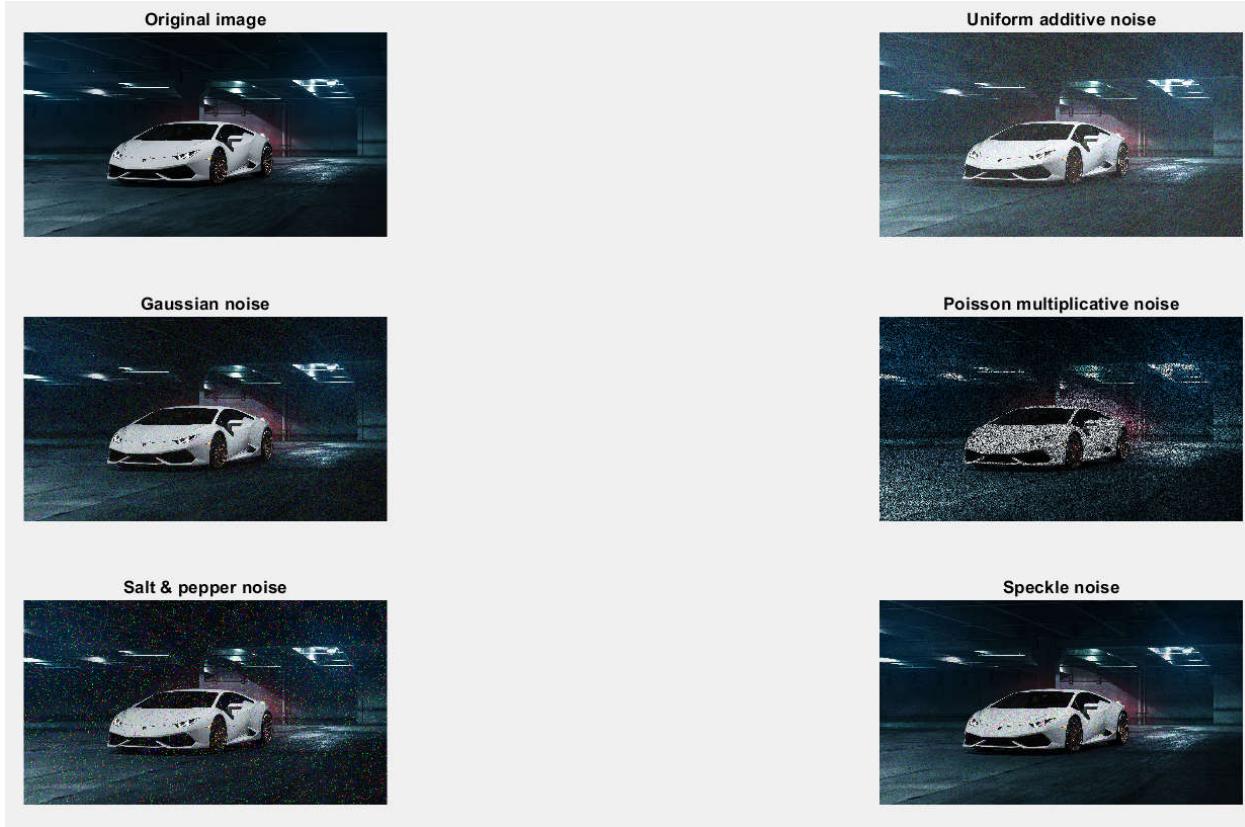
Gaussian filter applied



Wiener filter applied



عکس سوم:



Gaussian noise



Median filter applied



Gaussian filter applied



Wiener filter applied



Poisson noise



Median filter applied



Gaussian filter applied



Wiener filter applied



Salt & pepper noise



Median filter applied



Gaussian filter applied



Wiener filter applied



Speckle noise



Median filter applied

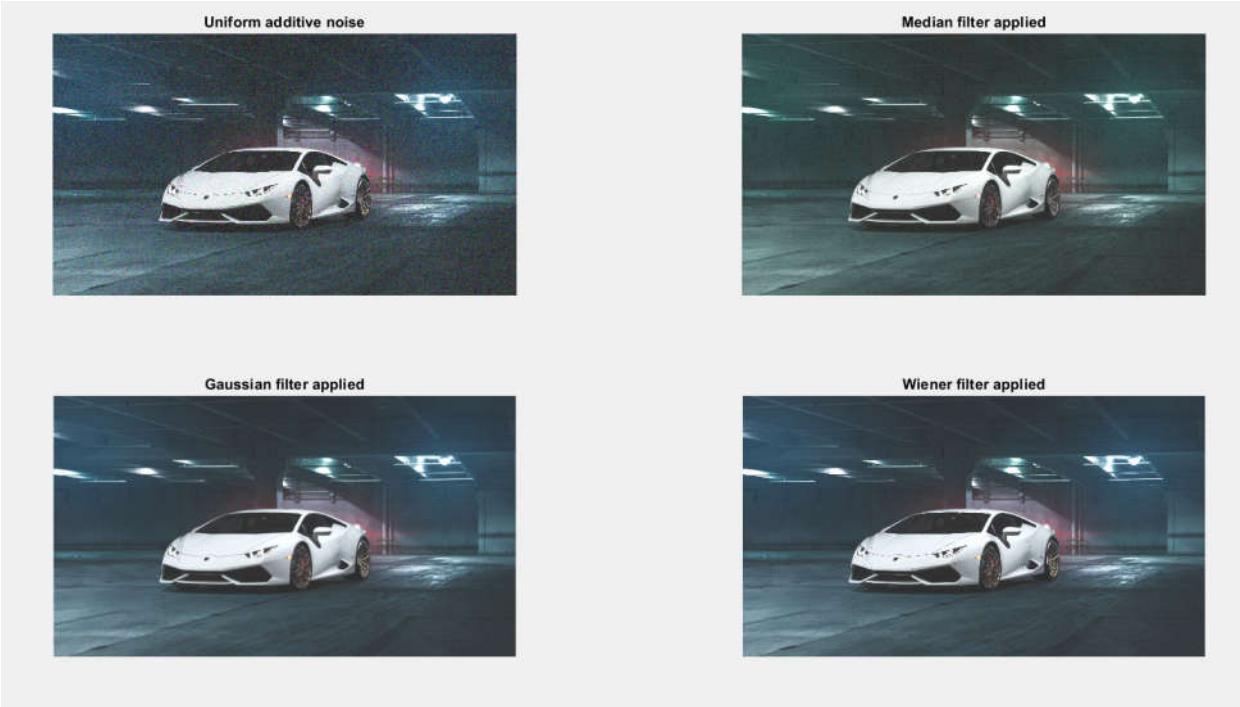


Gaussian filter applied



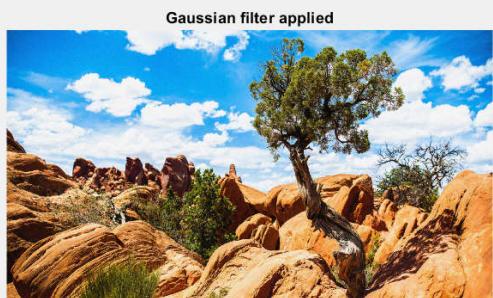
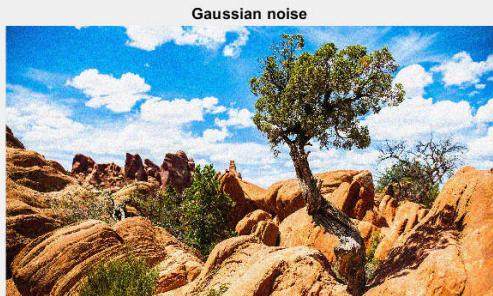
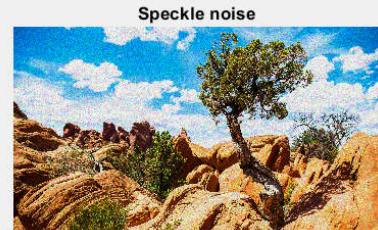
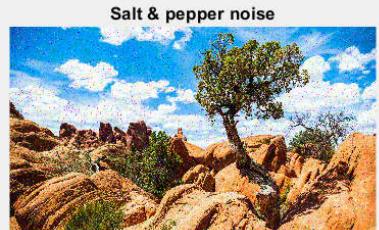
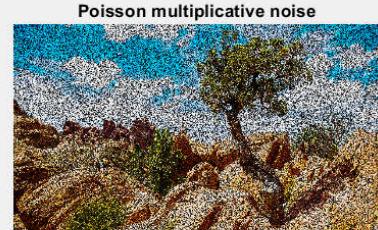
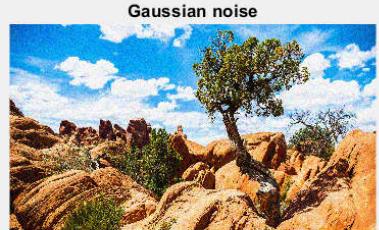
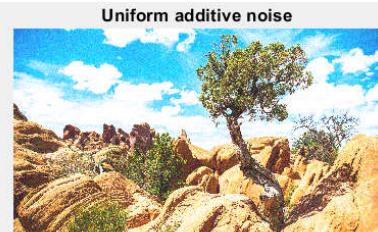
Wiener filter applied





عکس چهارم





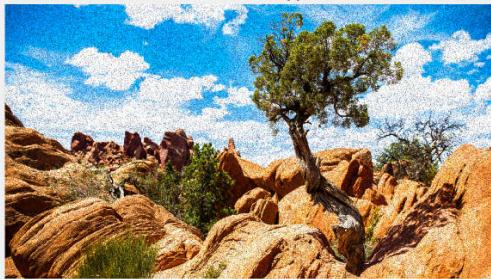
Poisson noise



Median filter applied



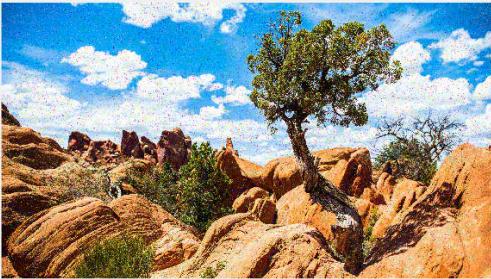
Gaussian filter applied



Wiener filter applied



Salt & pepper noise



Median filter applied



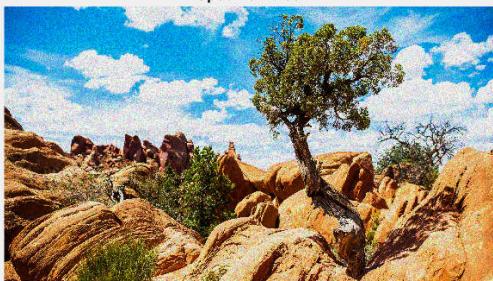
Gaussian filter applied



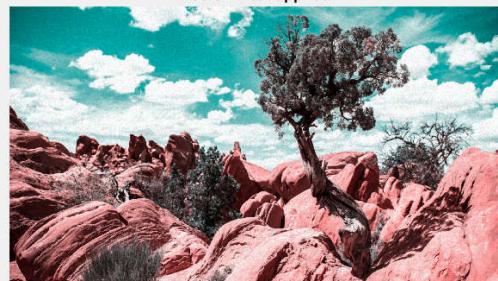
Wiener filter applied



Speckle noise



Median filter applied



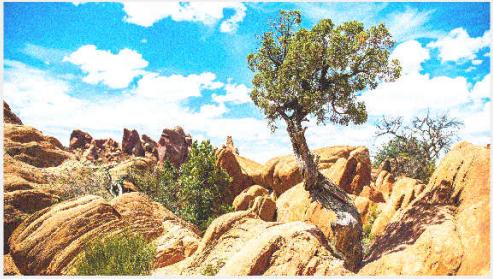
Gaussian filter applied



Wiener filter applied



Uniform additive noise



Median filter applied



Gaussian filter applied



Wiener filter applied



در بین نویزهای اعمال شده به عکس‌ها اثر Poisson و uniform additive noise از همه بیشتر بوده و رفع این نویز‌ها نیز به آسانی بقیه نیست و بعد از اعمال فیلترها نیز همچنان اثر نویز به خوبی مشهود است.

فیلتر wiener در کل بیشتر از بقیه فیلترها کیفیت عکس را کاهش می‌داد در حالی که فیلتری گاوی کیفیت را در بهترین حالت نگه می‌داشت. فیلتر میانه‌گیر در عمل با عکس‌های رنگی باعث تغییر ترکیب رنگ‌ها می‌شود. البته باید ذکر شود کهتابع که همان تابع میانه‌گیر خود متلب است این ایراد را ندارد.

فیلتر wiener علاوه بر کاهش کیفیت عکس باعث تارشدن تصویر نیز می‌شود(با اعمال پنجره‌ی بزرگ برای کار فیلتر).

Noise type	Best filter(s)
Uniform additive	1.Gaussian 2.wiener
Gaussian	1.Gaussian (the other two were not that bad either)
Poisson multiplicative	None (wiener was no good Gaussian was so so Median converted the noise to pepper noise)
Salt & pepper	1.Median 2.Gaussian
Speckle	Gaussian

سوال دوم

برای اجرای عملیات بر روی داده ها ابتدا فرکانس سمپلینگ آنها را یکی کردم. برای این کار فایل sound که فرکانس سمپلینگ بزرگتری دارد را ابتدا تا مقدار کم مرکعبی می‌دهیم و سپس فرکانس سمپلینگ را کاهش می‌دهیم. همچنین فایل دوم که طول بیشتری دارد را بعد از اعمال تغییرات سمپلینگ، به طول داده ی singing می‌کنیم. محاسبات از این به بعد روی این داده‌ها انجام شده است.

اثر صفر کردن فاز

استفاده از اندازه تبدیل فوریه: با صفر کردن فاز اگرچه میتوان آهنگ اصلی را به حالت گنگ و نصفه شنید اما در کل صوت کاملاً تغییر پیدا کرده است. نتیجتاً می‌توان گفت که فاز صوت تاثیر بسزایی در صدا و فایل کلی دارد.

استفاده از قسمت حقیقی تبدیل فوریه: در این صوت، صدای اصلی واضح‌تر است اما صوت به گونه است که گویا بر روی صدای اصلی، همان صدا را با تاخیر هایی پخش می‌کنیم.

اثر تغییر یکسان فاز

اگر سیگنال با همان اندازه ولی فاز شیفت یکسان یافته را پخش کنیم متوجه می‌شویم که تغییر یکسان فاز تاثیری بر سیگنال صوتی ندارد.

مشاهده می‌شود که در تغییر فاز یک سیگنال صوتی اگر به صورت یکنوا و یکسان تغییر فاز اعمال نشود، به دلیل اعوجاج، سیگنال صوتی قابل بازسازی و بازیابی نیست. اما اگر تغییر فاز اعمال شده به صورت یکسان به تمامی عناصر اعمال شده باشد سیگنال حاصل فرقی با سیگنال اولیه ندارد.

استفاده از فاز صوت دیگر

صداهای حاصل اختلاف زیادی با صوت اولیه دارند و به جز صدای نمایانی که شاید بتوان گفت که یک آهنگ را با آن می‌شناسیم باقی مانده است

تبدیل فوریه تصاویر

بر خلاف صوت، به نظر می‌رسد که فاز در تصاویر تا حدی برای بازسازی داده‌ها کافی باشد چرا که با انجام مراحل داده شده و نشان دادن عکسی که اندازه آن، اندازه عکس اول و فاز آن، فاز عکس دوم است، می‌توانیم عکس دوم را تشخیص دهیم و اثری از عکس اول در آن نیست. این تصاویر البته به‌گونه‌ای نیستند که همان تصویر اصلی باشند و به خاطر تغییر اندازه آنها تصاویر کمی تغییر کرده اند اما با این وجود شکل حاصل، شکل تصویری است که فاز آن در بازسازی تصویر استفاده شده است.

توجه شود که برای انتقال داده‌های تصاویر به حوزه فوریه بایستی از دستور fft2 استفاده شود و در صورت استفاده از دستور fft ، داده‌های نادرستی حاصل خواهند شد.

نکته‌ی قابل توجه در بازسازی تصاویر، این بود که اگر از عکسی تبدیل fft2 می‌گرفتیم و سپس از آن تبدیل فوریه معکوس ifft2 می‌گرفتیم، در خروجی صفحه‌ای سفید با تعدادی نقاط بر روی شکل می‌بینیم. با تقسیم خروجی ifft2 یک عکس بر خودش که برای گرفتن خروجی درست بعد از اعمال ifft2 داده‌ها بایستی بر 255 نرمالیزه شوند. در صورتی که داده‌ها نرمالیزه نشوند، از آنجایی که در فرمت RGB عدد 1 نمایانگر رنگ سفید است و تمامی داده‌های فعلی بزرگتر از 1 هستند، صفحه عموماً به رنگ سفید درمی‌آید.

در زیر عکس‌های حاصل بعد از تغییر فاز و نرمالیزه کردن آمده‌اند توجه شود که عکس سمت راست، عکس حاصل از اندازه عکس شماره 01 و فارعکس شماره 02 است که نهایتاً عکس 02 قابل تشخیص است و عکس سمت چپ، عکس حاصل از اندازه عکس شماره 02 و فاز عکس 01 است که در نهایت عکس 01 دیده می‌شود.



در صورتی که عکس ها را نرمالیزه نمی کردیم خروجی ها به شکل زیر بودند:



سؤال سوم

کد این قسمت با دو روش پیاده سازی شده است. روش اول استفاده از تابع آماده متلب برای شمارش تعداد دایره ها در یک صفحه باینری است. روش دوم استفاده از الگوریتم Hough برای یافتن مرکز دایره ها و پس از آن یافتن شعاع دایره ها است.

روش اول

در این روش بایستی ابتدا تصویر را باینری کنیم. بعد از تبدیل عکس رنگی به سیاه و سفید، داده ها به صورت اعدادی بین 0 تا 255 بر حسب شدت نور آن نقطه رده-بندی می شوند. با اعمال معیاری بین 0 تا 255، اعداد بزرگتر از آن را 1 و کمتر از آن را به صفر تبدیل می کنیم و سپس خرجی این مرحله را به تابع `regionprops` داده و دایره ها را پیدا می کنیم. در این تبدیل باینری، در ابتدا و انتهای بازه به مشکل می خوریم چرا که در ابتدای بازه تعداد نقاط سیاه در پس زمینه سفید خیلی زیاد هستند و این تابع متلب سعی می کند تمامی این نقاط و یا مجموعه نقاط را به عنوان دایره شناسایی بکند. در انتهای این بازه نیز تعداد نقاط سفید در پس زمینه سیاه زیاد می شوند و همان مشکل دوباره به وجود می آید پس برای درست عمل کردن این الگوریتم مرز باینری را بین 10 تا 150 انتخاب می کنیم و از بین تمامی خروجی ها مقدار مینیمم غیر صفر را انتخاب می کنیم.

روش دوم

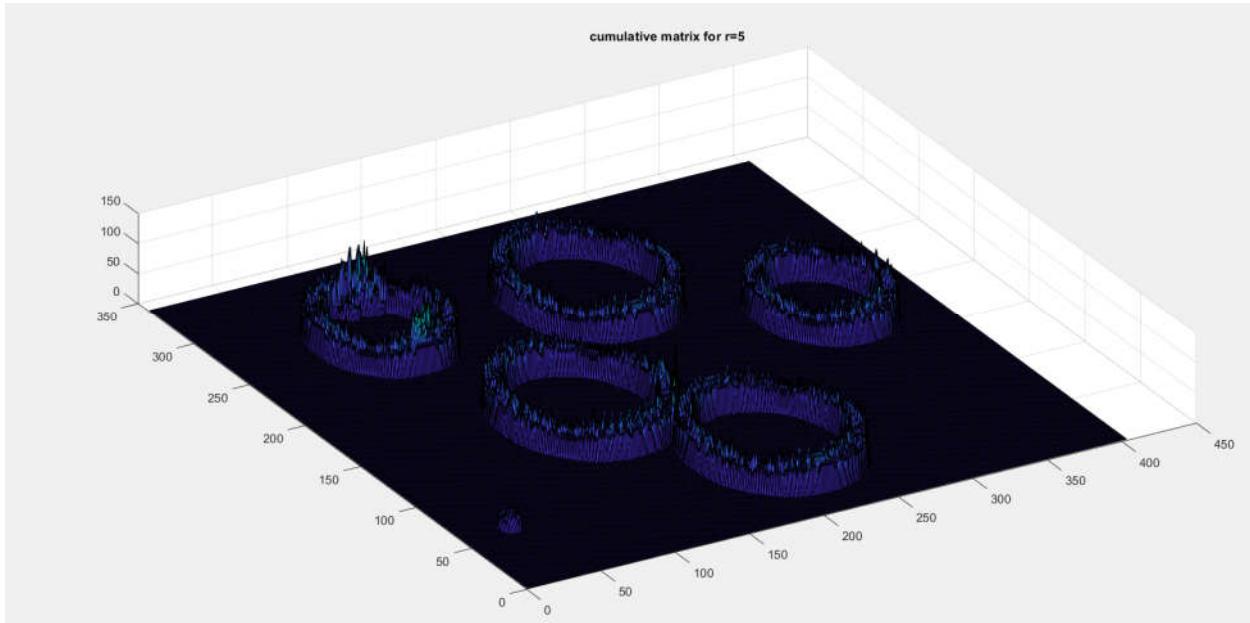
استفاده اصلی الگوریتم Hough برای شناسایی خط است ولی میتوان با تعمیم آن دایره را نیز شناسایی کنیم. میدانیم که معادله دایره به مرکز (a,b) و شعاع r به شکل زیر است. پس میتوانیم معادله مرکز دایره را به صورت زیر بر حسب نقاط روی دایره و شعاع دایره بنویسیم.

$$\begin{cases} x = a + r \cos \theta \\ y = b + r \sin \theta \end{cases} \rightarrow \begin{cases} a = x - r \cos \theta \\ b = y - r \sin \theta \end{cases}$$

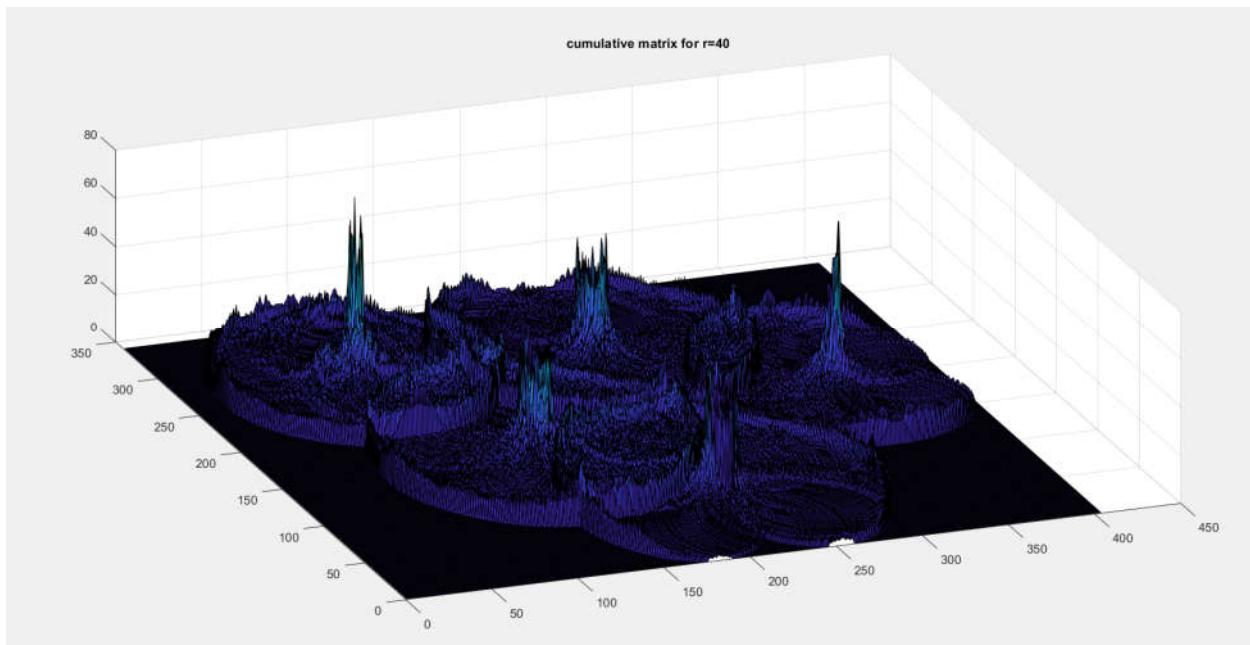
سری معادلات سمت راست در بالا خود نمایانگر دایره هایی به مرکز (y,x) و شعاع r هستند. اکنون الگوریتم را به صورت زیر پیاده سازی می کنیم.

بعد از طی مراحلی مطابق مراحل طی شده در روش اول برای باینری کردن عکس و سپس یافتن لبه های (edge) تصویر به ازای نقاط روی لبه ها و به ازای شعاع های مختلف، دایره هایی رسم می کنیم و در ماتریسی به تمامی نقاطی که به ازای یک شعاع خاص دایره از آنها گذشته است، یک واحد اضافه می کنیم. توجه شود که به ازای هر شعاع، بایستی یک ماتریس جداگانه داشته باشیم و گرنه الگوریتم پاسخ گو نیست. می توان نشان داد که به ازای شعاع های مناسب مرکز دایره ها

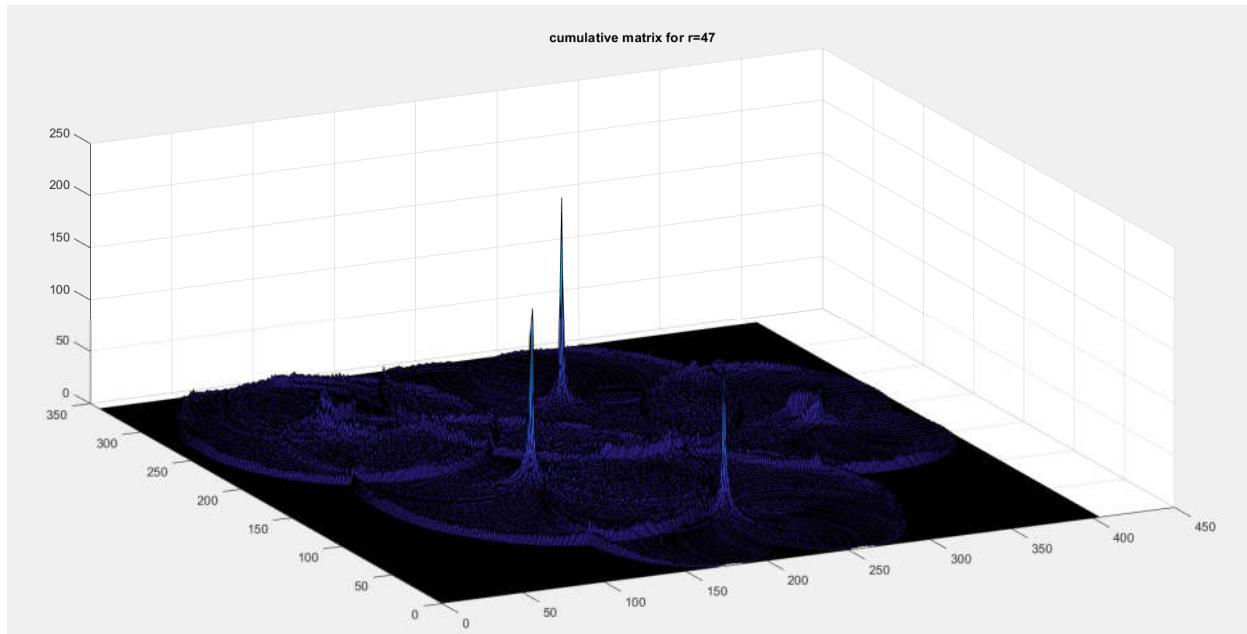
چگالی بیشتری نسبت به سایر نقاط داردند. برای مثال در زیر شکل سه سری از این ماتریس ها برای شعاع های مختلف آمده است:



عکس بهارای شعاع 5



بهارای شعاع 40



بهازی شعاع 47

درباره سه شکل بالا نکات زیر حائز اهمیت است:

وقتی که شعاع انتخابی مقدار کم و نامناسبی باشد، با این الگوریتم تنها آمده‌ایم و تعدادی دایره تودرتو کشیده‌ایم. اما بهازی شعاع‌های بزرگتر و مناسب، دایره‌های ساخته‌شده بر روی لبه‌های شکل به صورت بیشینه در مرکز دایره‌های که بر لبه آن قرار دارند با یکدیگر تقاطع داشته‌اند و چگالی این نقطه بسیار بیشتر از بقیه نقاط است. برای مثال عکس سوم که با شعاع 47 کشیده شده است دقیقاً 3 عدد از سکه‌ها را که خودشان نیز شعاع 47 دارند را به خوبی مشخص می‌کند و مرکز این سکه‌ها دارای چگالی بسیار بیشتری نسبت به بقیه نقاط است و می‌توانیم از این ماکریزم موضعی بودن برای شناسایی مرکز دایره‌ها استفاده کنیم. البته از آنجایی که در ابتدا ما نمیدانیم که شعاع دایره چند است نمیتوانیم فقط به شعاع خاص اتکا کنیم. در این قسمت بین بازه‌ای از شعاع‌های مناسب ماتریس‌ها را با هم جمع می‌کنیم و عملیات بعدی را بر روی این ماتریس انجام می‌دهیم. برای انتخاب این بازه مناسب معلوم است که این بازه بایستی شامل کوچکترین شعاع دایره‌های موجود در شکل و بزرگترین شعاع ممکن نیز باشد. برای این کار، بالاگوریتمی در ابتدای شروع کار و بعد از شناسایی لبه‌های تصویر، تقریبی از شعاع‌های موجود به دست می‌آوریم. برای این کار ماتریس حاصل بعد از یافتن لبه‌ها را در هر بار در یک راستا پرمایش می‌کنیم و فاصله‌ی بین اولین 1 تا آخرین 1 را پیدا می‌کنیم. بدیهی است که برخی از سطرها و ستونها هیچ عدد یکی در آنها ظاهر نشده است. سپس از بین تمامی سطر و ستون‌های غیرصفر میانگین می‌گیریم. این عدد را به عنوان حد بالای شعاع در نظر می‌گیریم.

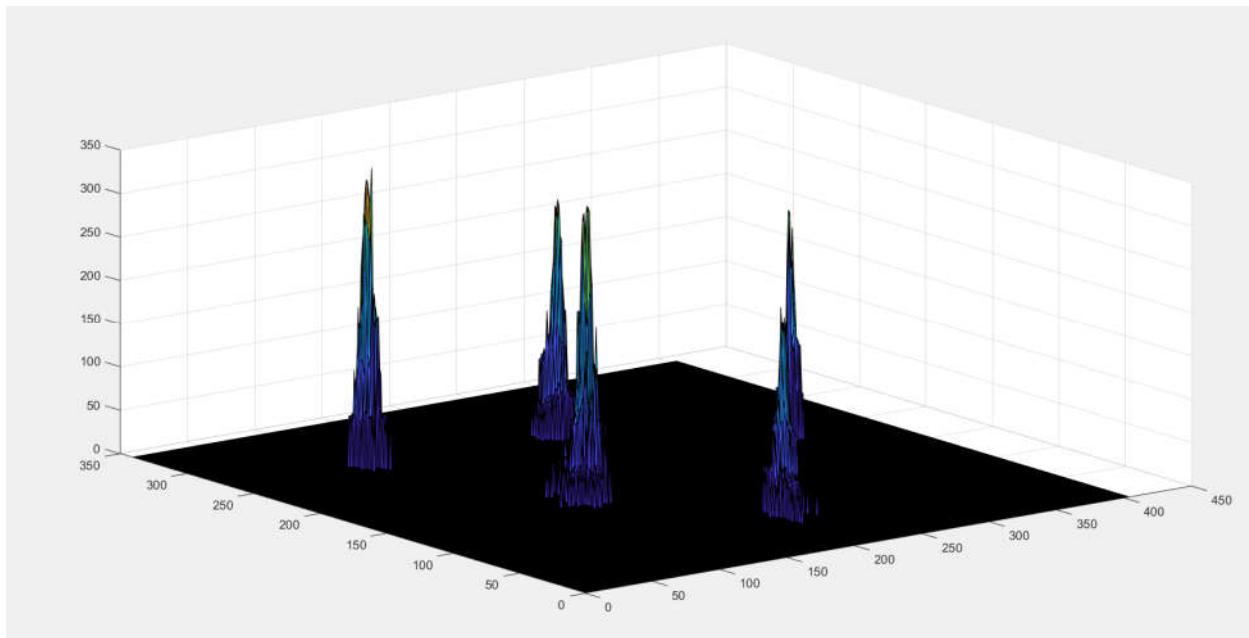
برای مثال اکنون حد بالای شعاع برای تصویر داده شده 72 بدست آمده که به مراتب بزرگتر از بزرگ‌ترین شعاع موجود در شکل، 47، است.

این مساله با فرض اینکه تمامی دایره‌های ما حداقل 20 پیکسل شعاع دارند و فاصله‌ی مرکز آنها نیز حداقل 20 پیکسل است حل شده است.

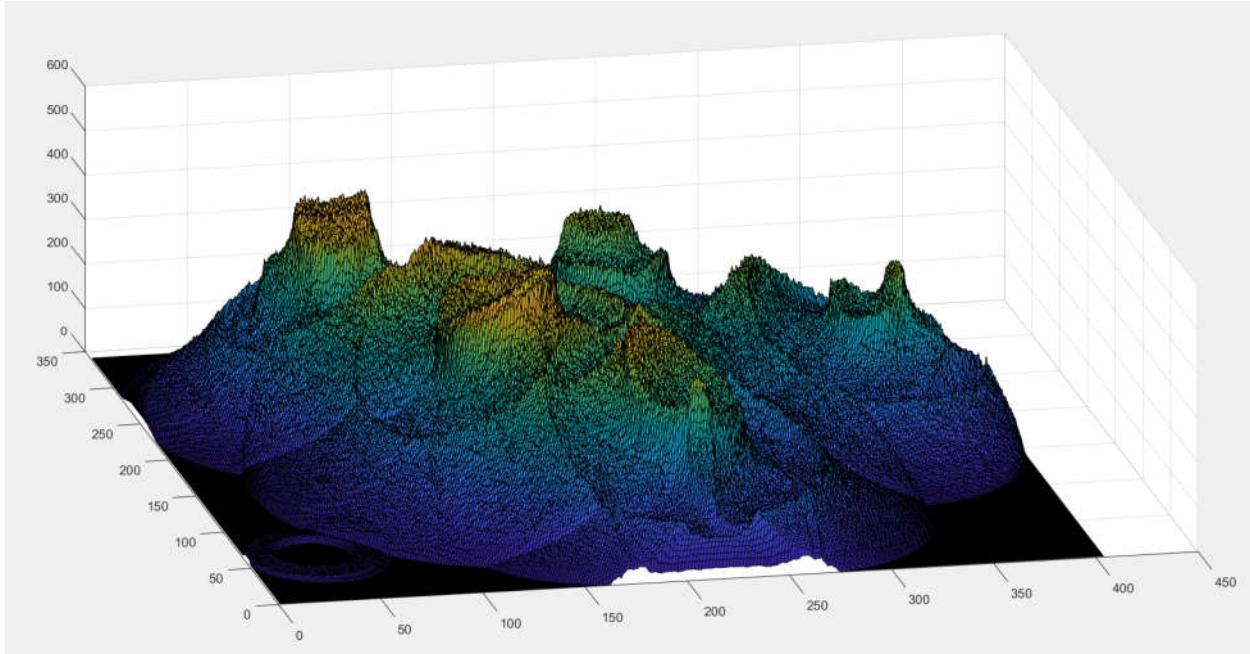
اکنون برای ادامه کار در تمامی ماتریس‌ها معیار زیر را اعمال می‌کنیم: تمامی داده‌هایی که کمتر از میانگین آن سری داده‌ها به اضافه 10 برابر انحراف معیار هستند را صفر می‌کنیم. از آنجایی که بنایه الگوریتم پیشنهاد شده می‌دانیم که مرکز دایره‌ها دارای چگالی بیشتری هستند، با این کار سعی داریم داده‌های اضافی را حذف کنیم.

اگر بیاییم و ماتریس‌های بالا را در شعاع بین 20، حد پایینی که برای شعاع دایره‌ها گرفتیم، و شعاع 72 با یکدیگر جمع کنیم، شکل زیر حاصل خواهد شد که محاسبات از این مرحله به بعد با این ماتریس انجام گرفته‌اند.

این حاصل جمع، بعد از معیار صفر کردن داده‌های اضافی ماتریسی حاصل می‌دهد که می‌توانیم با استفاده از آن مرکز دایره‌ها را تشخیص دهیم که شکل زیر که شکل همین ماتریس است نیز بیانگر این مشاهده است:



توجه شود که اگر معیار صفر کردن داده‌های اضافی اعمال نمی‌شد این ماتریس به شکل زیر درمی‌آمد که نمی‌توانیم از آن برای تشخیص مرکز دایره‌ها استفاده کنیم چرا که چگالی مرکز دایره‌ها کمتر از چگالی بسیاری از نقاط دیگر است.



همانگونه که مشاهده می‌شود نمی‌توانیم از ماتریس شکل بالا برای تشخیص مرکز دایره‌ها استفاده کنیم. البته اگر بتوان تخمین بهتری از شعاع دایره بدست آورد نتایج بهتر می‌شوند که در حال حاضر الگوریتمی برای آن نتوانستم پیدا کنم

بعد از مراحل بالا، طی الگوریتمی نقاط داده شده را دسته بندی می‌کنیم. نحوه دسته بندی نیز به صورت زیر است:

ابتدا یک بعد ماتریس را مرتب می‌کنیم. اولین نقطه غیر صفر را به عنوان مرکز دایره در نظر می‌گیریم. سپس در ادامه با پرمایش ماتریس فاصله بین نقطه‌ی جدید و مرکز دایره‌ای که گرفته‌ایم را می‌سنجدیم و اگر این فاصله بزرگ‌تر از 20 پیکسل، همان حد پایین شعاع دایره که گرفته‌ایم، باشد این نقطه به عنوان مرکز دایره جدید می‌گیریم و بقیه داده‌ها را با این مرکز مقایسه می‌کنیم. اما اگر این فاصله کمتر از 20 پیکسل باشد میانگین وزن‌دار بر حسب چگالی نقاط در ماتریس اولیه بین مرکز دایره، که چگالی آن را مجموع چگالی تمامی نقاطی می‌گیریم که با آن میانگین گرفته شده است، و نقطه‌ی جدید را به عنوان مرکز دایره‌ی فعلی انتخاب می‌کنیم. بعد از تمام شدن یک سری مرحله بالا برخی دایره‌های مشابه و نزدیک هستند که اکنون باید با مرتب کردن بعد دیگر این ماتریس، آنها را باهم ادغام کنیم.

برای یافتن شعاع دایره، میانگین فاصله تمامی نقاطی که در یافتن یک مرکز دایره دخیل بودند را تا آن مرکز می‌یابیم

در آخر نیز برای اطمینان دایره‌های نزدیک را با یک معیار اضافه در هم ادغام کردیم.

خروجی الگوریتم برای تصویر داده شده به صورت زیر است:



مختصات یافته شده برای تصویر بالا بدین صورت است:

the centers of the circles and the corresponding radius are:

Center Coordinates	Radius
260.7137 102.2324	35.5911
149.1840 152.7922	46.4471
68.9039 215.3530	47.4243
254.3266 245.8995	48.2150
190.0915 338.7362	36.8813

و یا برای تصاویری دیگر به شکل زیر است:



منبع سوال سوم و الگوریتم :Hough

Circle recognition through a 2D Hough Transform and radius histogramming

Dimitrios Ioannou, Walter Huda, Andrew F. Laine