

فیلترها

شیوا رادمنش

اطلاعات گزارش	چکیده
تاریخ: ۱۳۹۹/۹/۷	در این تکلیف به بررسی فیلترهای مختلف خطی و غیر خطی و تاثیر آنها بر نویزهای مختلف، فیلترهای تشخیص لبه، روش های بهبود کیفیت تصویر مات و نویزی، یک روش unsharpening mask پرداخته شده است. این روش ها، روش هایی برای پردازش تصویر در حوزه مکان هستند.
واژگان کلیدی:	فیلتر میانه
	فیلتر میانگین
	فیلتر گوسی
	salt and pepper
	Unsharp masking

بهبود داده می شود. این عملیات به طور ویژه بر روی نواحی اعمال می شود که اطلاعات بیشتری در بر دارند و با بهبود کیفیت آنها پردازش راحت تر می شود.

۱- مقدمه

نوشتار حاضر به بررسی فیلترهای حوزه مکان مانند فیلترهای میانگین و میانه و تاثیر آنها بر نویز گوسی و salt and pepper، فیلترهای مشتق گیر و تشخیص لبه، فیلترهای شارپ کننده می پردازد.

تمام پیاده سازی های این تمرین با استفاده از زبان پایتون انجام شده است.

۲- شرح تکنیکال

هر تصویر بررسی شده در این گزارش با استفاده ازتابع imread کتابخانه opencv پایتون خوانده شده است. فیلترها در هر بخش به تصاویر در حالت grayscale اعمال شده اند. برای اعمال فیلترها ابتدا عمل padding انجام شده است. این عمل به این دلیل انجام می شود که فیلتر مورد نظر به پیکسل های

این پیش پردازشها یکی از مراحل اصلی در پردازش تصویر در حوزه های مختلف که از پردازش تصویر کمک می گیرند می باشند. که کیفیت تصویر مورد نظر را به منظور استفاده ی مستقیم یا به عنوان ورودی به بخش پردازش در روش های بینایی کامپیوتر

بسیار متفاوت با سایر پیکسل‌های تصویر می‌باشد ممکن است حتی میانگین به سمت نویز متمایل شود و در واقع نویز در در محدوده بزرگتری گسترش یابد.

در این فیلترها وزن‌ها به جای اینکه به تدریج با افزایش فاصله از مرکز صفر شوند به طور ناگهانی قطع و صفر می‌شوند که این باعث عدم پیوستگی در تصویر نهایی می‌شود.

یکی دیگر از دلایل این است که box filter ها تمايل به محور کردن تصویر در جهت عمود دارند و در واقع isotropic نیستند. این ویژگی باعث می‌شود در برخی برنامه‌ها که شامل تصویری با جزئیات زیاد و یا مولفه‌های هندسی قوی هستند، نتایج مطلوبی حاصل نشود.

یک کاربرد دیگر که این فیلترها در آن عملکرد ضعیفی دارند لنزهای defocus می‌باشد. این لنزاها معمولاً با فیلترهای پایین گذر مدل می‌شوند. در حالی که box filter ها تقریب ضعیفی نسبت به مشخصه‌های محوری و تار شدن لنزاها دارند.

محدوده حاشیه‌های تصویر هم اعمال شود. در این تمرین از reflect padding استفاده شده است. پس از اعمال padding فیلتر با شروع از نقطه‌ی (۰،۰) تصویر، ضرب آرایه ای فیلتر بر روی آن قسمت از تصویر که فیلتر روی آن قرار گرفته است انجام شده و سپس حاصل جمع ضرایب محاسبه می‌شود. و این حاصل جمع به مقدار پیکسل مرکزی فیلتر اعما می‌شود. سپس پیکسل بر روی تصویر حرکت کرده و این عملیات برای هر پیکسل تصویر انجام می‌شود.

۲.۱.۱ - سوال

این فیلتر مقدار میانگین پیکسل‌های همسایه را به پیکسل مرکزی اعمال می‌کند. اما این میانگین وزن دار نیست و در واقع بین پیکسل‌های نزدیکتر به پیکسل مرکزی و پیکسل‌های دور تر تمایزی قائل نمی‌شود و مقدار پیکسل مرکزی از هر دو به یک اندازه تاثیر می‌پذیرد. (در صورت بزرگ بودن سایز فیلتر این مشکل بیشتر مشاهده می‌شود).

همچنین در صورت وجود نویز در همسایه‌های پیکسل مرکزی، مقدار سطح خاکستری پیکسل نویز نسبت به پیکسل‌های اطراف آن بسیار متفاوت است و این فیلتر به مقدار نویز وزنی به اندازه‌ی سایر پیکسل‌های همسایه می‌دهد و مقدار میانگین متاثر از نویز می‌شود و با توجه به اینکه مقدار نویز معمولاً

۲.۱.۲ - سوال

این ویژگی‌ها ممکن است در ابتدا افزایش یابند ولی با تکرار فیلتر پس از مدتی، تصویر تقریباً دیگر تغییر نمی‌کند و ثابت می‌ماند. بر اساس قضیه‌ی Central

در این بخش در ابتدا با استفاده ازتابع random_noise کتابخانه scikit image نویز salt and pepper با چکالی 0.05 و 0.02 به تصویر Lena اعمال شده است. سپس به تصاویر نویزی بدست آمده، فیلتر میانه با ابعاد 3×3 و 5×5 و 7×7 و 9×9 اعمال شده است.

۲-۴-۱- فیلتر میانه

فیلتر میانه از جمله فیلترهایی می‌باشد که برای بهبود تصاویر با نویز salt and pepper مناسب می‌باشد. اعمال این فیلتر همانند سایر فیلترها با لغزاندن و حرکت دادن فیلتر بر روی تصویر انجام می‌شود ولی در این فیلتر به جای ضرب کردن مقادیر فیلتر در مقادیر پیکسل‌های قطعه تصویر مورد نظر، پیکسل‌های قطعه تصویر را مرتب کرده و مقدار میانی (مقدار وسط) را به پیکسل مرکزی قطعه اعمال می‌کند.

۳.۲.۲- سوال ۲-۵

در این بخش در ابتدا با استفاده ازتابع random_noise کتابخانه scikit image نویز گوسی با مقدار واریانس 0.01 و 0.05 و 0.1 به تصویر Lena اعمال شده است. سپس به تصاویر نویزی حاصل، فیلتر میانه و box filter با ابعاد 3×3 و 5×5 و 7×7 و 9×9 اعمال شده است.

با تکرار فیلتر سطح خاکستری پیکسل‌ها به سمت توزیع نرمال همگرا می‌شود. و در واقع $k \rightarrow \infty$ بار تکرار فیلتر، معادل اعمال نوعی فیلتر گوسی می‌باشد.

۳.۱.۳- سوال ۲-۳

یک فیلتر خطی است که برای یک فیلتر Box filter با اندازه $n \times n$ ضرایب فیلتر مقدار ثابت $\frac{1}{n^2}$ می‌باشد.

در این بخش به تصویر Lena در حالت grayscale، یک box filter (فیلتر میانگین گیر) با ابعاد 3×3 با تکرارهای $1, 5, 10, 20, 50, 100, 200, 300$ و 400 بار اعمال شده است. این فیلتر به صورت زیر می‌باشد.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

پس از اعمال این فیلتر ممکن است برخی پیکسل‌های تصویر حاصل دارای مقادیر سطح خاکستری با مقادیر اعشاری باشند. این مقادیر به پایین رند شده تا عدد حاصل int باشد.

۳.۲.۱- سوال ۲-۴

همچنین برای اینکه فیلتر باعث مات شدن تصویر نشود به جای استفاده از فیلتر با ابعاد بزرگتر، چند بار فیلتر میانه با ابعاد کوچک اعمال شده است.

سایز فیلتر میانه در اینجا 7×7 بوده و تعداد تکرار آن ۶ است.

۲-۷- سوال ۳.۴.۱

در این بخش برای تشخیص لبه از فیلترهای زیر استفاده شده است.

$$a) \frac{1}{2} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, \quad b) \frac{1}{6} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \quad c) \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

هر سه فیلتر لبه‌های عمودی تصویر را شناسایی می‌کنند.

فیلترهای b و c بسیار شبیه هستند با این تفاوت که در فیلتر c وزن بیشتری برای پیکسل‌های مجاور پیکسل مرکزی در نظر گرفته شده است.

پیکسل a بار محاسباتی کمتری دارد، زیرا برای هر پیکسل ۳ عمل جمع و ۲ عمل ضرب انجام می‌دهد در حالی که در فیلتر c، b برای محاسبه مقدار هر پیکسل ۹ عمل ضرب و ۸ عمل جمع انجام می‌دهند.

پس از اعمال فیلتر مقدار محاسبه شده برای هر فیلتر ممکن است عدد اعشاری باشد یا در بازه‌ی ۰ تا ۲۵۵ نباشد. بدین منظور پس از اعمال فیلتر، عملیات نرمال‌سازی خطی بر روی تصویر انجام شده

توضیحات مربوط به فیلتر میانه در بخش ۱-۴-۲ و توضیحات مربوط به box filter در ۳.۱.۳ آورده شده است.

۲-۶- سوال ۳.۳.۱

در این بخش تصویری مات و دارای نویز وجود دارد که باید با روش‌هایی کیفیت آن را بهبود داد. برای ایجاد چنین تصویری در یک فضای تاریک عکس گرفته شده است.

برای رفع مات بودن از روشی که در ادامه آورده شده، استفاده شده است.

در این روش ابتدا به کمک فیلتر میانگین 11×11 یک تصویر تار از تصویر اصلی ایجاد کرده. سپس تصویر تار را از تصویر اصلی کم می‌کنیم تا لبه‌های تصویر بدست آید. حال با افزون حاصل تغیریق با ضریب مناسب به تصویر اصلی تصویری شارپ تر خواهیم داشت.

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

$$g(x,y) = f(x,y) + k \cdot g_{mask}(x,y)$$

برای حذف نویز از فیلتر میانه استفاده شده است. حذف نویز با استفاده از فیلتر میانه در دو حالت بررسی شده است. قبل از شارپ کردن و پیش از آن.

انتظار می‌رود فیلتر a لبه‌های قطری با شیب مثبت و فیلتر b لبه‌های قطری با شیب منفی را تشخیص دهد.

اعمال این فیلترها بار محاسباتی کمتری نسبت به فیلترهای b و c در بخش ۲-۷ دارد زیرا برای هر پیکسل ۴ عمل ضرب و ۳ عمل جمع انجام می‌دهند.

۲-۹-سوال ۳.۵.۱

یکی از روش‌های unsharp masking به شکل زیر است.

$$(1 - \alpha)I + I' = I + \alpha(I' - I)$$

در این رابطه I تصویر اصلی می‌باشد و I' تصویر smooth شده می‌باشد. α نیز پارامتری است که مقدار آن بین ۰ و ۱ می‌باشد.

در اینجا مقادیر α برابر ۰.۱ و ۰.۵ و ۰.۸ و ۱ در نظر گرفته شده است.

در این بخش برای smooth کردن تصویر از فیلتر گوسی با اندازه‌های ۳*۳ و ۵*۵ و ۷*۷ و ۹*۹ استفاده شده است. فیلتر گوسی همچون فیلتر میانگین باعث مات شدن تصویر می‌شود ولی تفاوت آن با فیلتر میانگین آن است که در فیلتر گوسی مقادیر پیکسل‌های نزدیکتر به مرکز وزن بیشتری دارند.

است. برای این کار از رابطه‌ی زیر استفاده شده است.

$$I_N = \frac{(I - MIN)}{Max - Min} \times 255$$

در این رابطه I_N برابر با مقدار نرمال شده (در بازه‌ی ۰ تا ۲۵۵)، I مقدار اولیه‌ی پیکسل و Max و Min به ترتیب بیشترین و کمترین مقدار پیکسل عکس در بازه‌ی اولیه است.

*فیلتر sobel نام دارد.

۲-۱۰-سوال ۳.۴.۲

در این بخش برای تشخیص لبه از فیلترهای زیر استفاده شده است.

$$a) \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b) \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

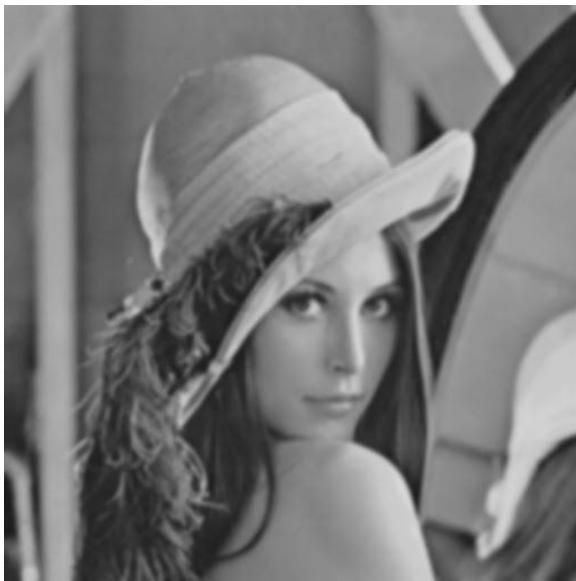
این فیلتر robert نام دارد. فیلتر a تخمینی از مشتق افقی را پیدا می‌کند و فیلتر b تخمینی از مشتق عمودی. در واقع حاصل ضرب فیلتر a در مقادیر پیکسل‌ها در یک قطعه تصویر برابر با g_x و حاصل ضرب فیلتر b در مقادیر پیکسل‌ها در یک قطعه تصویر برابر با g_y در آن قطعه تصویر می‌باشد. برای اعمال هر دو مشتق به پیکسل مرکزی گرادیان را محاسبه می‌کنیم.

$$\nabla f = \left[(g_x)^2 + (g_y)^2 \right]^{0.5}$$



شکل ۱- تصویر Lena پس از ۱ باز اعمال box filter با

اندازه‌ی 3×3



شکل ۲- تصویر Lena پس از ۵ باز اعمال box filter با

اندازه‌ی 3×3

می‌دانیم $I' - I$ لبه‌های تصویر را به ما می‌دهد که در صورت افروden آن به تصویر اصلی تصویر شارپتر می‌شود(به بخش ۲-۶) مراجعه شود. حال در این بخش مقدار $I' - I$ با ضریب α با تصویر اصلی جمع شده. در واقع گویی مقدار لبه‌ها $(I - I')$ از تصویر اصلی کم شده است. با کم کردن لبه‌ها از تصویر اصلی انتظار داریم تصویر smooth تر شود.

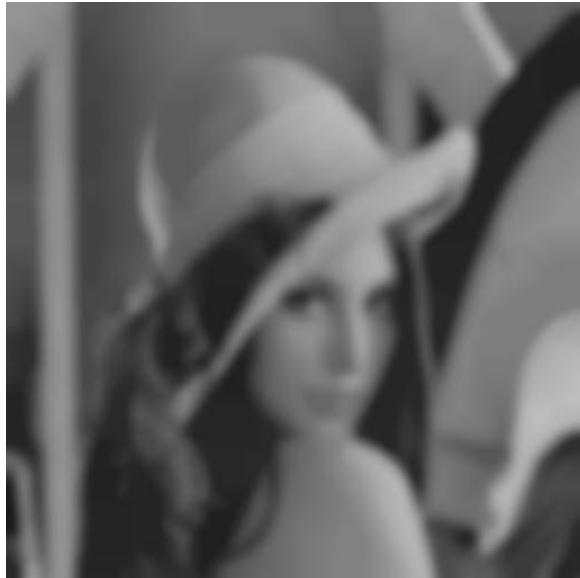
۳- شرح نتایج

۳-۱- سوال ۲۰۲

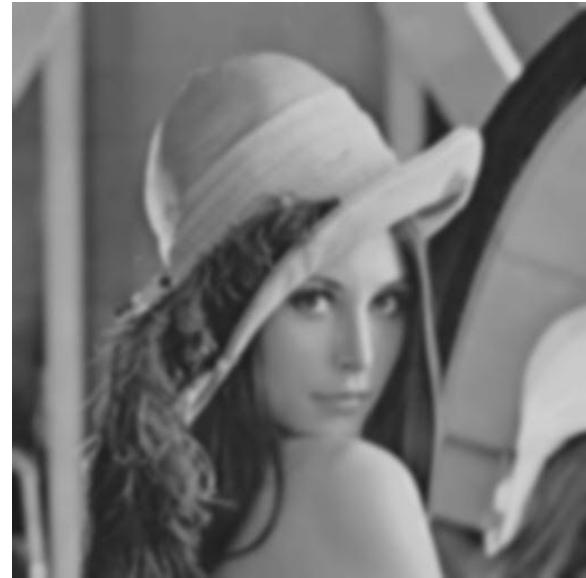
نتیجه‌ی تکرار اعمال box filter بر روی تصویر Lena در ادامه قابل مشاهده است.



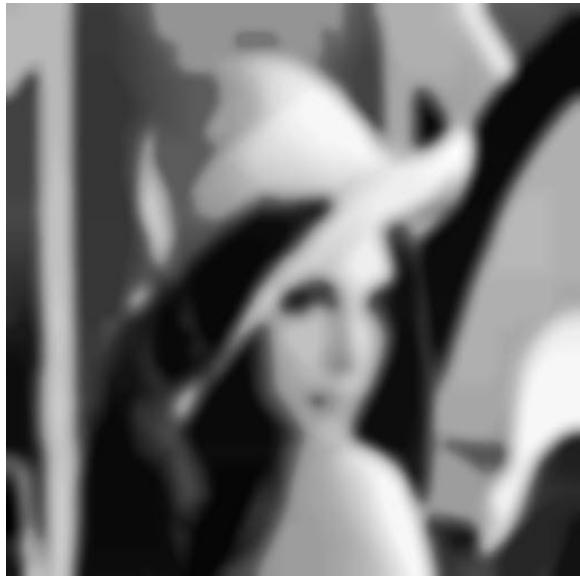
شکل ۱- تصویر Lena بدون اعمال فیلتر



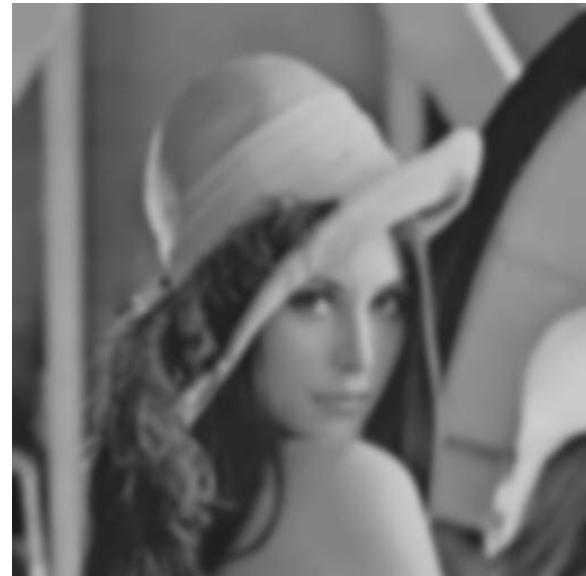
شکل۶- تصویر Lena پس از ۵۰ باز اعمال box filter
با اندازه‌ی 3×3



شکل۴- تصویر Lena پس از ۱۰ باز اعمال box filter
با اندازه‌ی 3×3



شکل۷- تصویر Lena پس از ۱۰۰ باز اعمال box filter
با اندازه‌ی 3×3



شکل۵- تصویر Lena پس از ۲۰ باز اعمال box filter
با اندازه‌ی 3×3



شکل ۹- تصویر Lena پس از ۴۰۰ باز اعمال box filter با اندازه‌ی 3×3



شکل ۸- تصویر Lena پس از ۲۰۰ باز اعمال box filter با اندازه‌ی 3×3

مشاهده می‌شود که در ابتدا با تکرار فیلتر تصویر مات تر می‌شود و شفافیت و جزئیات آن کم می‌شود این روند با مقایسه‌ی شکل ۱ و ۲ و ۳ و ۴ و ۵ و ۶ و ۷ قابل مشاهده است. در ادامه هرچه تعداد تکرارها بیشتر می‌شود و به سمت بی نهایت میل می‌کند تقریباً تغییر محسوسی در نتیجه مشاهده نمی‌شود با مقایسه‌ی شکل ۸ و ۹ و ۱۰ می‌توان این نتیجه را مشاهده نمود. در واقع با تکرار k بار فیلتر وقتی $\rightarrow \infty$ ، سطح خاکستری پیکسل‌های تصویر به سمت توزیع نرمال همگرا می‌شوند(بر اساس قضیه‌ی آماری .) (Central Limit Theorem



شکل ۹- تصویر Lena پس از ۳۰۰ باز اعمال box filter با اندازه‌ی 3×3

۲.۲.۱- سوال ۳-۲

حاصل اعمال نویز salt and pepper به تصویر Lena در شکل های ۱۰ و ۱۱ و ۱۲ قابل مشاهده می باشد.



شکل ۱۱- نتیجه اعمال نویز salt and pepper با

چگالی ۰.۱



شکل ۱۰- نتیجه اعمال نویز salt and pepper با

چگالی ۰.۰۵

نتیجه اعمال فیلتر میانه بر روی تصاویر نویزی در

ادامه قابل مشاهده است.





شکل ۱۵- نتیجه‌ی اعمال فیلتر میانه با سایز $۳*۳$ بر روی تصویر نویزی با چگالی ۰.۲ .



شکل ۱۳- نتیجه‌ی اعمال فیلتر میانه با سایز $۳*۳$ بر روی تصویر نویزی با چگالی ۰.۵ .

همانطور که قابل مشاهده است فیلتر میانه $۳*۳$ در تصویر با چگالی نویز کم، نویز را تا حد خوبی از بین برده است اما با افزایش چگالی نتوانسته نویز را به طور کامل از بین برد مثلًا آثار نویز در شکل ۱۵ به وضوح قابل مشاهده می‌باشد.



شکل ۱۴- نتیجه‌ی اعمال فیلتر میانه با سایز $۳*۳$ بر روی تصویر نویزی با چگالی ۰.۱ .



شکل-۱۸- نتیجه‌ی اعمال فیلتر میانه با سایز $5*5$ بر روی تصویر نویزی با چگالی 2.0 .



شکل-۱۶- نتیجه‌ی اعمال فیلتر میانه با سایز $5*5$ بر روی تصویر نویزی با چگالی 0.5 .

با افزایش سایز فیلتر عمل از بین بردن نویز به خوبی انجام شده است ولی با مقایسه شکل‌های $13, 14$ ،
 15 با شکل‌های $16, 17, 18$ مشاهده می‌شود که با افزایش سایز فیلتر تصویر مقداری مات شده است.



شکل-۱۷- نتیجه‌ی اعمال فیلتر میانه با سایز $5*5$ بر روی تصویر نویزی با چگالی 0.1 .



شکل ۲۱- نتیجه‌ی اعمال فیلتر میانه با سایز ۷*۷ بر روی تصویر نویزی با چگالی ۰.۲.



شکل ۱۹- نتیجه‌ی اعمال فیلتر میانه با سایز ۷*۷ بر روی تصویر نویزی با چگالی ۰.۰۵.

مشاهده می‌شود که فیلتر میانه ۷*۷ تاثیر نویز را تا حد خوبی از بین برد و لی باعث از بین رفتن جزئیاتی (جزئیات لبه‌ها) در تصویر شده و تصویر مات‌تر شده است.



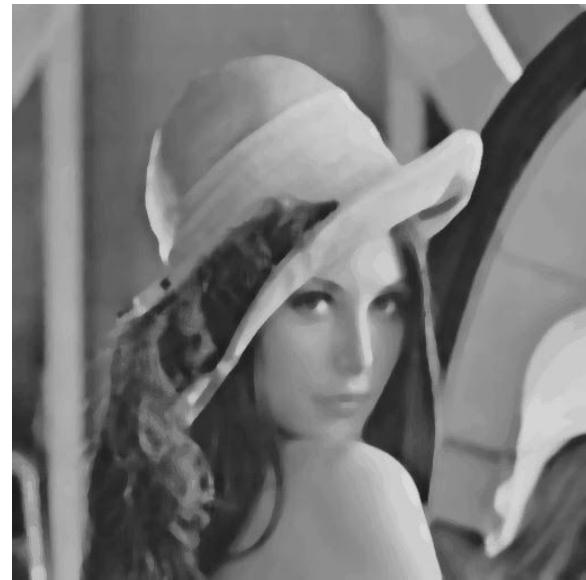
شکل ۲۰- نتیجه‌ی اعمال فیلتر میانه با سایز ۷*۷ بر روی تصویر نویزی با چگالی ۰.۱.



شکل-۲۲- نتیجه‌ی اعمال فیلتر میانه با سایز $9*9$ بر روی تصویر نویزی با چگالی 0.2 .

مشاهده می‌شود

در جدول ۱ هر یک از تصاویر با استفاده از معیار MSE با هم مقایسه شده اند.



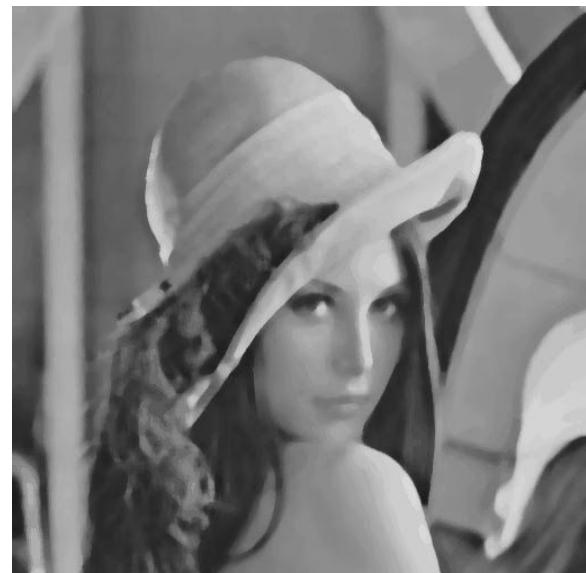
شکل-۲۱- نتیجه‌ی اعمال فیلتر میانه با سایز $9*9$ بر روی تصویر نویزی با چگالی 0.05 .

	3×3	5×5	7×7	9×9
$\rho = 0.05$	28.49	59.11	90.29	120.03
$\rho = 0.1$	39.48	65.62	100.59	129.47
$\rho = 0.2$	94.49	93.61	135.39	185.8

جدول-۱- MSE تصاویر حاصل از اعمال فیلتر میانه بر

salt and pepper نویز

همانطور که انتظار می‌رفت فیلتر با اندازه‌ی کوچک‌تر ویژگی‌های تصویر را بهتر حفظ می‌کند و با افزایش اندازه‌ی فیلتر تصویر مات‌تر شده و برخی جزئیات را



شکل-۲۲- نتیجه‌ی اعمال فیلتر میانه با سایز $9*9$ بر روی تصویر نویزی با چگالی 0.1 .



شکل ۲۳- نتیجه‌ی اعمال نویز گوسی با واریانس ۱۰۰۰



شکل ۲۴- نتیجه‌ی اعمال نویز گوسی با واریانس ۵۰۰

از دست می‌دهد. تنها در یک مورد با افزایش سایز فیلتر MSE افزایش نیافته است. که مربوط به تصویر حاصل از اعمال فیلتر 3×3 به تصویر نویزی با چگالی ۲۰٪ می‌باشد. همانطور که در شکل ۱۵ قابل مشاهده است. در اینجا پس از اعمال فیلتر آثار نویز به طور کامل از بین نرفته است. به همین دلیل با افزایش سایز فیلتر به 5×5 مقدار MSE خیلی کم کاهش یافته است.

همچنین مشاهده می‌شود با افزایش چگالی نویز مقدار MSE افزایش یافته است. اگرچه از نظر بصری خیلی مشهود نیست اما از نظر معیار تصاویر با چگالی نویز بیشتر اطلاعات بیشتری را از دست داده اند.

۳.۲.۲- سوال ۳

حاصل اعمال نویز گوسی به تصویر Lena در شکل‌های ۲۳ و ۲۴ و ۲۵ قابل مشاهده است.



شکل ۲۷- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی 3×3 بر تصویر دارای نویز گوسی با واریانس ۰۰۱

مشاهده می‌شود که هر دو فیلتر نویز را تا حدی کاهش داده اند اما همچنان نویز وجود دارد. از نظر بصری به نظر می‌رسد که فیلتر میانه نسبت به box filter تصویر نویزی را کمتر بهبود داده است.



شکل ۲۵- نتیجه‌ی اعمال نویز گوسی با واریانس ۰۰۱

در ادامه نتیجه‌ی اعمال فیلتر میانه و box filter بر این تصاویر نویزی قابل مشاهده است.



شکل ۲۶- نتیجه‌ی اعمال box filter با اندازه‌ی 3×3 بر تصویر دارای نویز گوسی با واریانس ۰۰۱

با مقایسه شکل ۲۸ و ۲۹ مشاهده می‌شود که تصویر حاصل از اعمال box filter مقداری مات تر است و در تصویر حاصل از اعمال فیلتر میانه جزئیات box filter بیشتر قابل تشخیص می‌باشند. در واقع بیشتر عمل smoothing را انجام می‌دهد.



شکل ۲۰- نتیجه اعمال box filter با اندازه $۳*۳$ بر تصویر دارای نویز گوسی با واریانس ۰.۱.



شکل ۲۸- نتیجه اعمال box filter با اندازه $۳*۳$ بر تصویر دارای نویز گوسی با واریانس ۰.۵.



شکل ۲۹- نتیجه اعمال فیلتر میانه با اندازه $۳*۳$ بر تصویر دارای نویز گوسی با واریانس ۰.۰۵.



شکل-۳۲- نتیجه‌ی اعمال box filter با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 100 .



شکل-۳۱- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $3*3$ بر تصویر دارای نویز گوسی با واریانس 10 .



شکل-۳۳- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 1000 .

با مقایسه‌ی شکل-۳۰ و ۳۱ مشاهده می‌شود که box filter از نظر بصری تصویر را بیشتر بهبود داده است و همچنین تصویر را smooth کرده است.



شکل-۳۵- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 0.05



شکل-۳۶- نتیجه‌ی اعمال box filter با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 0.1

با مقایسه‌ی شکل‌های ۲۶ و ۲۷ با شکل‌های ۳۲ و ۳۳ مشاهده می‌شود که با افزایش اندازه‌ی فیلتر تصویر خروجی نویز کمتری دارد اما تصویر مقداری مات تر شده است.

همچنین مشاهده می‌شود که box filter نست به فیلتر میانه از نظر بصری نویز را بیشتر کاهش داده است.



شکل-۳۴- نتیجه‌ی اعمال box filter با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 0.05



شکل-۳۸- نتیجه‌ی اعمال box filter با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس 100 .



شکل-۳۷- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $5*5$ بر تصویر دارای نویز گوسی با واریانس 1 .



شکل-۳۹- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس 1 .

با مقایسه شکل‌های ۳۰ و ۳۱ با شکل‌های ۳۶ و ۳۷ مشاهده می‌شود که با افزایش سایز فیلتر نویز بیشتر کاهش یافته است اما تصویر مات‌تر شده و جزئیات مربوط به لبه‌ها کمتر شده است.



شکل ۴۲- نتیجه‌ی اعمال box filter با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس ۱۰.



شکل ۴۰- نتیجه‌ی اعمال box filter با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس ۰۰۵.



شکل ۴۳- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس ۱۰.



شکل ۴۱- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $7*7$ بر تصویر دارای نویز گوسی با واریانس ۰۰۵.



شکل-۴۵- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی $9*9$ بر تصویر دارای نویز گوسی با واریانس 0.01



شکل-۴۶- نتیجه‌ی اعمال box filter با اندازه‌ی $9*9$ بر تصویر دارای نویز گوسی با واریانس 0.005

مشاهده می‌شود که با افزایش سایز فیلتر از $5*5$ به $7*7$ نویز تصویر کاهش یافته است اما تصویر به طور قابل توجهی مات شده و جزئیاتی مانند لبها در تصویر کمتر قابل مشاهده است.
همچنین قابل مشاهده است که box filter از نظر بصری نسب به فیلتر میانه نویز را بیشتر کاهش داده است.



شکل-۴۴- نتیجه‌ی اعمال box filter با اندازه‌ی $9*9$ بر تصویر دارای نویز گوسی با واریانس 0.01



شکل ۴۹- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی 9×9 بر تصویر دارای نویز گوسی با واریانس ۰.۱.



شکل ۴۷- نتیجه‌ی اعمال فیلتر میانه با اندازه‌ی 9×9 بر تصویر دارای نویز گوسی با واریانس ۰.۰۵.

با مقایسه‌ی شکل ۴۸ و ۴۹ مشاهده می‌شود که در یک تصویر نویزی با چگالی بالا box filter از نظر بصری نسبت به فیلتر میانه بهتر کلیات شکل را حفظ کرده است.

در جدول ۲ و جدول ۳ هر یک از تصاویر از نظر معیار MSE با هم مقایسه شده اند.



شکل ۴۸- نتیجه‌ی اعمال box filter با اندازه‌ی 9×9 بر تصویر دارای نویز گوسی با واریانس ۰.۱.

	3×3	5×5	7×7	9×9
$\sigma = 0.01$	150.1	158.1	214.8	290.4
$\sigma = 0.05$	574.2	278.5	235.1	266.4

۴-۳-۱ سوال

تصویر اصلی دارای نویز می‌باشد و مات است در شکل ۵۰ قابل مشاهده است.



شکل ۵۰- تصویر اصلی(مات و نویزی)

تصویر شارپ شده بدون اعمال فیلتر میانه (برای رفع نویز) در شکل ۵۱ قابل مشاهده است.

$\sigma = 0.1$	1089.8	481.8	328.1	291.1
----------------	--------	-------	-------	-------

جدول ۲- MSE تصاویر حاصل از اعمال فیلتر میانه بر نویز گوسی

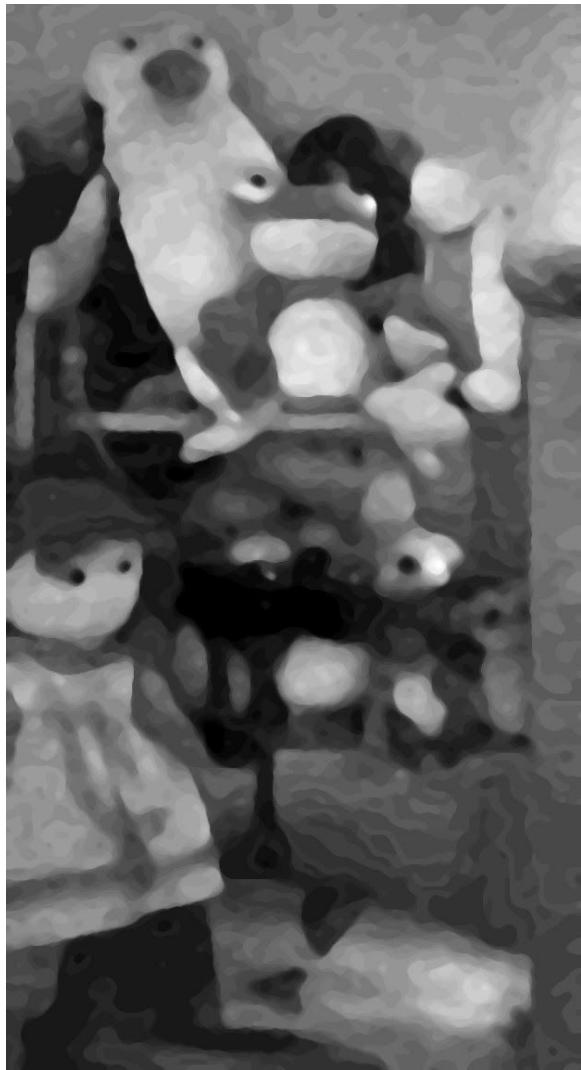
	3×3	5×5	7×7	9×9
$\sigma = 0.01$	137.7	198.2	282.1	340.6
$\sigma = 0.05$	373.7	242.1	280.5	313.6
$\sigma = 0.1$	639.1	372.5	283.3	291.8

جدول ۳- MSE تصاویر حاصل از اعمال box filter بر نویز گوسی

همانطور که انتظار می‌رفت به طور کلی نتایج اعمال box filter بر نویز گوسی بهتر از نتایج فیلتر میانه است.

به طور کلی با افزایش اندازه فیلتر در هر دو فیلتر مقدار MSE کاهش می‌یابد. دلیل این اتفاق این است که با افزایش سایز فیلتر نویز کاهش می‌یابد.

در هر دو فیلتر با افزایش سایز فیلتر از 7×7 به 9×9 مقدار MSE خیلی کم افزایش می‌یابد زیرا با افزایش سایز فیلتر با وجود کاهش نویز، تصویر مات تر می‌شود و جزئیات آن کاهش می‌یابد.



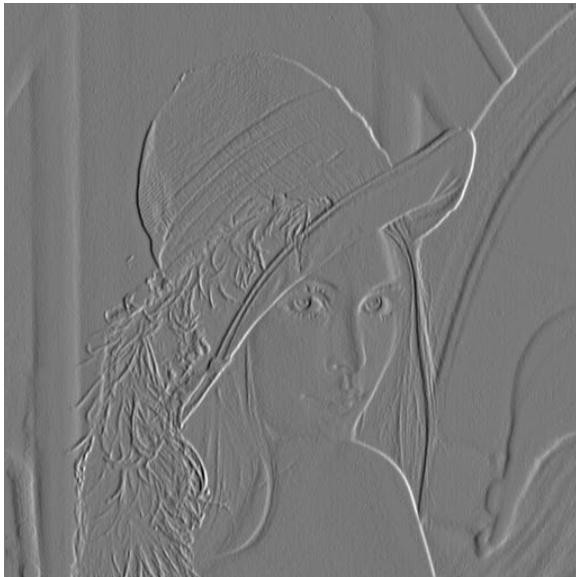
شکل ۵۲- اعمال ۴ بار فیلتر میانه پس از شارپ کردن

تصویر



شکل ۵۱- تصویر شارپ شده بدون اعمال فیلتر میانه

نتایج حاصل از اعمال فیلترها بر رو تصویر Lena در ادامه قابل مشاهده است.



شکل ۵۴- حاصل اعمال فیلتر a به تصویر Lena



شکل ۵۳- اعمال ۴ بار فیلتر میانه قبل از شارپ کردن تصویر



شکل ۵۵- حاصل اعمال فیلتر b به تصویر Lena

با مقایسه شکل ۵۲ و ۵۳ مشاهده می‌شود که اگر فیلتر میانه را قبل از شارپ کردن تصویر اعمال کنیم نویز کمتری خواهیم داشت و جزئیات بیشتری قابل مشاهده است.

۴.۳- سوال ۲

نتایج حاصل از اعمال فیلتر robert به تصویر Lena در ادامه قابل مشاهده است.



شکل ۵۷- حاصل اعمال فیلتر robert_a بر تصویر Lena



شکل ۵۸- حاصل اعمال فیلتر robert_b بر تصویر Lena



شکل ۵۹- حاصل اعمال فیلتر c به تصویر Lena

تقریبا نتیجه‌ی هر ۳ فیلتر مشابه می‌باشد.

مشاهده می‌شود که فیلتر a نسبت به فیلتر b به مقدار خیلی جزئی لبه‌های بیشتری پیدا کرده است، مثلا در قسمت کلاه و در گوش‌های پایین سمت راست تصویر. دلیل این تفاوت به این علت است که فیلتر a برای فعال شدن فقط دو پیکسل مجاور افقی خود را در نظر می‌گیرد اما فیلتر b علاوه بر فیلترهای مجاور افقی ۴ پیکسل مجاور مورب را هم در نظر می‌گیرد. در واقع حتی ممکن است فیلتر a جایی که واقعاً لبه نیست را لبه تشخیص دهد.

همچنین مشاهده می‌شود نتیجه‌ی اعمال فیلتر b و فیلتر c بسیار نزدیک به هم می‌باشد و از نظر بصری خیلی متفاوت نیستند.

به طور کلی فیلتر robert نسبت به فیلتر sobel بیشتر به نویز حساس است. فیلتر sobel نسبت به لبها را بهتر تشخیص می‌دهد.

۴.۵.۱- سوال ۳-۷



شکل ۶۰- تصویر حاصل از مقدار $a=0$



شکل ۵۹- حاصل اعمال فیلتر robert (هردو فیلتر a و b) بر تصویر Lena

همانطور که انتظار می‌رفت مشاهده می‌شود که فیلتر robert_b لبهايی قطری با شبیه منفی بیشتری نسبت به فیلتر robert_a تشخیص داده و فیلتر robert_a لبهايی مورب با شبیه مثبت بیشتری تشخیص داده است.

از مقایسه نتیجه اعمال فیلتر sobel (شکل ۵۶) با نتیجه اعمال فیلتر robert مشاهده می‌شود که فیلتر sobel لبها را نسبت به فیلتر robert بهتر و با کیفیت بیشتری تشخیص می‌دهد. در مورد سایر فیلترهای بخش قبل نیز همینطور است.



شکل ۶۳- تصویر حاصل از مقدار $a=0.8$ و سایز فیلتر
گوسی $3*3$



شکل ۶۱- تصویر حاصل از مقدار $a=0.1$ و سایز فیلتر
گوسی $3*3$



شکل ۶۴- تصویر حاصل از مقدار $a=1$ و سایز فیلتر
گوسی $3*3$



شکل ۶۲- تصویر حاصل از مقدار $a=0.5$ و سایز فیلتر
گوسی $3*3$



شکل ۶۶- تصویر حاصل از مقدار $a=0.5$ و سایز فیلتر ۵*۵ گوسی



شکل ۶۷- تصویر حاصل از مقدار $a=0.8$ و سایز فیلتر ۵*۵ گوسی

مشاهده می شود که با افزایش مقدار آلفا تاثیر تصویر smooth شده بر تصویر بیشتر می شود. و در واقع لبه های تصویر با ضریب بیشتری از تصویر کم می شوند و باعث مات تر شدن تصویر می شوند.



شکل ۶۸- تصویر حاصل از مقدار $a=0.1$ و سایز فیلتر ۵*۵ گوسی



شکل ۶۹- تصویر حاصل از مقدار $a=0.1$ و سایز فیلتر
گوسی ۷*۷



شکل ۶۸- تصویر حاصل از مقدار $a=1$ و سایز فیلتر
گوسی ۵*۵



شکل ۷۰- تصویر حاصل از مقدار $a=0.5$ و سایز فیلتر
گوسی ۷*۷

مشاهده می شود که با افزایش اندازه فیلتر، تصویر در محدوده لبه ها حاصل smooth تر شده است که همینطور انتظار می رفت.



شکل ۷۳- تصویر حاصل از مقدار $a=0.1$ و سایز فیلتر
گوسی $9*9$



شکل ۷۱- تصویر حاصل از مقدار $a=0.8$ و سایز فیلتر
گوسی $7*7$



شکل ۷۴- تصویر حاصل از مقدار $a=0.5$ و سایز فیلتر
گوسی $9*9$



شکل ۷۲- تصویر حاصل از مقدار $a=1$ و سایز فیلتر
گوسی $7*7$

به طور کلی قابل مشاهده است که با افزایش مقدار آلفا، تاثیر تصویر smooth شده بر تصویر بیشتر می‌شود و لبها با ضریب بزرگتری از تصویر اصلی کم می‌شوند.

با افزایش اندازه‌ی فیلتر گوسی همانطور که انتظار می‌رود، تصویر نهایی در محدوده‌ی لبها smooth تر می‌شود.

وقتی آلفا برابر با ۱ است، طبق رابطه‌ی ذکر شده در بخش ۲-۹ تصویر حاصل برابر با تصویر smooth شده می‌شود.



شکل ۷۵- تصویر حاصل از مقدار $a=0.8$ و سایز فیلتر ۹*۹ گوسی



شکل ۷۶- تصویر حاصل از مقدار $a=1$ و سایز فیلتر ۹*۹ گوسی

منابع

[1] Digital Image Processing, Rafael C. Gonzalez, Rechard E. Wood

[2]Digital Image Processing, Berned Jahne

[3] <https://dsp.stackexchange.com/questions/18225/applying-a-filter-several-times-on-data>

Appendix

```
import cv2
import numpy as np
import math
from skimage.util import random_noise
#-----
# this function applys the box filter on the input image
# the size of the filter is filter_dim*filter_dim
# returns the results of applying the the filter on original_img
def box_filter(original_img, filter_dim):
    img = np.pad(original_img, (filter_dim -1, filter_dim-1), 'reflect')

    box_filter = np.ones([filter_dim, filter_dim], dtype= float)
    box_filter = np.true_divide(box_filter, filter_dim*filter_dim)
    output = filtering(img, box_filter)
    output = unpad(output, filter_dim)
    output = normal(output)
    return output

#-----
# this method applys the input filter matrix on the input image
# my_filter is the matrix of filter
# returns the result of the input my_filter on img
def filtering(img, my_filter):
    filter_dim = np.size(my_filter, 0)
    output = np.zeros_like(img, dtype = int)
    filter_center = int(filter_dim/2)

    for i in range(np.size(img, 0)-filter_dim+1):
```

```

    for j in range(np.size(img, 1)-filter_dim+1):
        img_part = img[i:i+filter_dim, j:j+filter_dim]
        value = np.sum(np.multiply(img_part, my_filter))
        output[i+filter_center, j+filter_center] = value
    return output

#-----
# this method removes the padding
# takes the image with padding(as img) and size of the filter(as filter_dim)
# returns a image with the original size
def unpad(img, filter_dim):
    original_img = img[filter_dim-1 : np.size(img, 0)-filter_dim+1, filter_dim-1
    : np.size(img, 1)-filter_dim+1]
    return original_img

#-----
# this method applys box filter several times
# n is the number of times that method applys box filter on the image
# filter_dim is the filter size
def several_time_boxfilter(original_img, filter_dim, n):
    new_img = original_img
    for i in range(n):
        new_img = box_filter(new_img, filter_dim)
    return new_img

#-----
def gaussian_noise(img, var):

    mean = 0
    sigma = var**0.5
    image = img

    row,col = image.shape
    print(image.shape)
    gauss = np.random.normal(mean,sigma,(row,col))
    gauss.dtype = 'uint8'
    print(gauss.shape,"gav")
    gauss = gauss.reshape(row,col)

    noisy = np.add(image, gauss)

    print(noisy.shape)
    return noisy

```

```

#-----
#-----#
def salt_and_pepper(image, amount):
    row,col = image.shape
    s_vs_p = 0.5
    out = np.copy(image)
    # Salt mode
    num_salt = np.ceil(amount * image.size * s_vs_p)
    coords = [np.random.randint(0, i - 1, int(num_salt))for i in image.shape]
    out[coords] = 255

    # Pepper mode
    num_pepper = np.ceil(amount* image.size * (1. - s_vs_p))
    coords = [np.random.randint(0, i - 1, int(num_pepper))for i in image.shape]
    out[coords] = 0
    out.dtype = 'uint8'
    return out
#-----
# this method applys median filter on the input noisy image
# size of the filter is filter_dim*filter_dim
def medain_filter(noisy, filter_dim):
    img = np.pad(noisy, (filter_dim -1, filter_dim-1), 'reflect')
    output = np.zeros_like(img)
    filter_center = int(filter_dim/2)
    for i in range(np.size(img, 0)-filter_dim+1):
        for j in range(np.size(img, 1)-filter_dim+1):
            img_part = img[i:i+filter_dim, j:j+filter_dim]
            value = np.median(img_part)
            output[i+filter_center, j+filter_center] = value
    output = unpad(output, filter_dim)
    output = normal(output)
    return output
#-----
# this method applys differnce filter on the input image
# 3 filters are available:
# enter "a" as dif_filter for (1/2)*[1, 0, -1]
# enter "b" as dif_filter for (1/6)*[[1, 0, -1],[1, 0, -1],[1, 0, -1]]
# enter "c" as dif_filter for (1/8)*[[1, 0, 1],[2, 0, -2],[1, 0, -1]]
def difference_filter(img):
    dif_a = np.array([1, 0, -1])
    dif_a = np.true_divide(dif_a, 2)

    dif_b = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
    dif_b = np.true_divide(dif_b, 6)

    dif_c = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])

```

```

dif_c = np.true_divide(dif_c, 8)

out_a = filtering(img, dif_a)
out_b = filtering(img, dif_b)
out_c = filtering(img, dif_c)

out_a = normal(out_a)
out_b = normal(out_b)
out_c = normal(out_c)

return out_a, out_b, out_c
#-----
def robert_filter(img):
    robert_v = np.array([[0, 0, 0],
                        [0, 1, 0],
                        [0, 0, -1]])

    robert_h = np.array([[0, 0, 0],
                        [0, 0, 1],
                        [0, -1, 0]])

    img = np.pad(img, (2, 2), 'reflect')

    vertical = filtering(img, robert_v)
    horizontal = filtering(img, robert_h)

    output = np.sqrt(np.square(vertical) + np.square(horizontal))
    output = unpad(output, 3)
    output = normal(output)
    vertical = normal(vertical)
    horizontal = normal(horizontal)

    return output, vertical, horizontal
#-----
def normal(img):
    min_val = np.amin(img)
    max_val = np.amax(img)
    print(min_val, max_val)
    output = np.zeros_like(img)

    for i in range(np.size(img, 0)):
        for j in range(np.size(img, 1)):
            new_val = math.floor(((img[i, j] - min_val)*255) / (max_val -
min_val))
            output[i, j] = new_val

```

```
    return output
#
def mse(true, pred):
    MSE = np.square(np.subtract(true,pred)).mean()
    return MSE
```