

فیلتر های حوزه مکان

سحر شیخ الاسلامی

چکیده

اطلاعات گزارش

فیلتر های حوزه مکان، به طور مستقیم با پیکسل های عکس اصلی درگیر می شوند و از این طریق سعی بر اثر گذاری بر تصویر را دارند. در این گزارش با این فیلتر ها، محدوده اثر گذاری آنها آشنا می شویم.

تاریخ: ۸/۹/۱۳۹۹

واژگان کلیدی:

Spatial Filtering
Smoothing Spatial Filter
Box Filter
Median Filter
Sharpening Spatial Filter
Edge detection
Roberts Filter
Sobel Filter
Noise Reduction

۱- مقدمه

۲- شرح تکنیکال

۳.۱.۱: قسمت

فیلتر های smoothing (که از آنها با نام فیلتر های میانگین گیر و یا پایین گذر (low pass) نیز یاد می شود)، فیلتر هایی در حوزه مکان هستند که برای کاهش تغییرات شدید شدت رنگی در تصویر استفاده می شوند. به علت اینکه نویز ها غالبا شامل تغییرات شدید شدت رنگی هستند، از این فیلتر ها به عنوان فیلتر هایی برای کم کردن نویز نیز یاد می شود. این فیلتر ها عموما برای کاهش جزئیات نامرتب یا ناخواسته استفاده می شوند. این فیلتر ها معمولاً با روش های دیگر تقویت کیفیت عکس استفاده می شوند.

فیلتر های خطی convolution، در عکس smoothing شده و تاثیر مات کننده (blur) کننده دارند. میزان مات شدن و واپسیت به سایز فیلتر دارد، هر چی فیلتر بزرگتر باشد، اثر بیشتر است.

Box filter ساده ترین فیلتر پایین گذر است، که همه ضرایب آن مقداری برابر دارند (معمولایک). یک باکس فیلتر با سایز $m \times n$ از یک ماتریس $m \times n$ تشکیل شده است، که همه خانه های آن مقداری برابر یک دارند، و در انتهای نرمال سازی شده است. این نرمال سازی، که در همه فیلتر های low pass وجود دارد، معمولًا دو دلیل دارد:

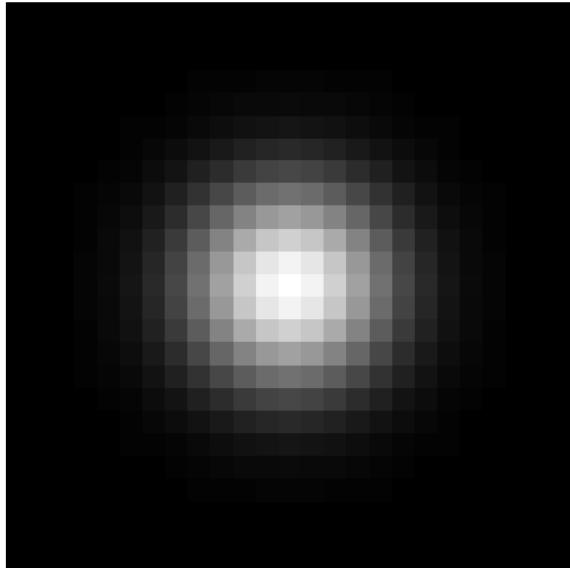
فیلتر کردن یک تکنیک است که به وسیله ای آن می توان یک عکس را اصلاح کرد و یا کیفیت آن را افزایش داد. بعنوان مثال شما می توانید یک عکس را فیلتر کنید تا برخی از ویژگی هایش را مهمتر جلوه دهید و یا اینکه برخی از ویژگی های آن را حذف کنید.

فیلتر کردن، در واقع یک عملیات تقریبی و محدوده ای است به طوری که مقدار به دست آمده برای هر پیکسل از عکس خروجی به وسیله ای اعمال یک یا چند الگوریتم بر پیکسل هایی که در همسایگی پیکسل ورودی قرار دارند، به دست می آید.

محدوده یا همسایگی یک پیکسل، در واقع یک مجموعه از پیکسل ها است که موقعیت آنها نسبت به این پیکسل تعیین شده است.

اعمال فیلتر خطی بر روی یک عکس، از طریق یک عملیات convolution (convolution) انجام می پذیرد. نوعی عملیات محدوده ای است، که به وسیله ای آن هر پیکسل خروجی، از جمع وزنی (weighted sum) پیکسل های در همسایگی پیکسل ورودی به دست می آید.

به ماتریسی که حاوی این وزن ها است، convolution kernel گفته می شود و با نام فیلتر، شناخته می شود



شکل ۱.۱.۲: خروجی دوربین فوکوس نشده



شکل ۱.۱.۳: خروجی مدل شده دوربین فوکوس نشده با استفاده از box filter

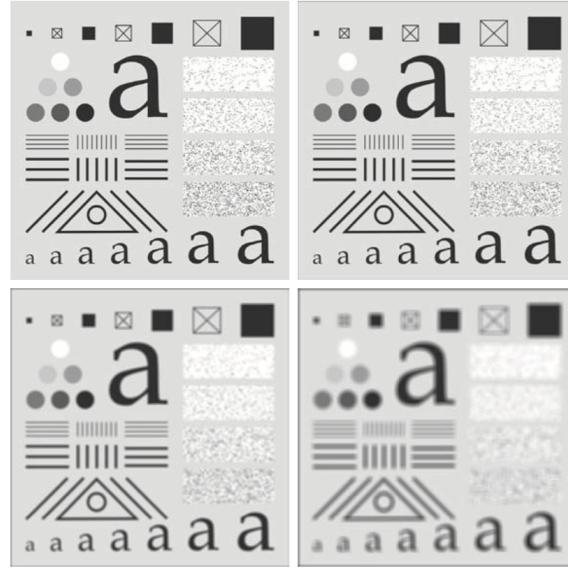
محدودیت دیگر این روش این است که این روش ها، در درجه های عمودی و یا افقی مات می کنند. در برخی تصاویر با جزییات بالا که علاقه مند به محو کردن همانند فیلتر گوسی داریم، نتیجه این فیلتر نامطلوب است. در شکل های هندسی پیچیده نیز، محدود بودن باکس فیلتر به مات کردن عمودی، نتیجه غیر دلخواه خواهد داد.

قسمت ۱.۱.۲:

اعمال کردن فیلتر برای از بین بردن ویژگی های نامطلوب عکس روشی بسیار مفید است. معمولاً پس از یکبار اعمال فیلتر، نتیجه به شکل کاملاً قابل حسی قابل مشاهده است. با افزایش تعداد دفعاتی که یک فیلتر را به یک عکس اعمال می کنیم، ویژگی منفی مورد نظر مان کم کم کمتر می شود. اما نکته قابل توجه این است که بین این دو رابطه مستقیم وجود ندارد. به این معنی که اگر ۱۰۰ بار فیلتر را اعمال کنیم، جواب لزوماً ۱۰۰ برابر بهتر شود. بله با افزایش تعداد اعمال فیلتر، اثر بخشی آن کاهش می یابد. به طور مثال اگر خروجی ۳ بار اعمال فیلتر را با ۱۰ بمقایسه کنیم، احتمالاً شاهد تغییری چشم گیر در این

- مقدار خروجی یک فیلتر برای یک ناحیه با شدت رنگی ثابت، برابر مقدار شدت رنگی تصویر اولیه شود.(باید چنین شود)

چنین نرمال سازی از بوجود آمدن bias در هنگام فیلترینگ جلوگیری می کند، زیرا مجموع پیکسل های داخل فیلتر قبل و بعد از اعمال فیلتر ثابت است.



شکل ۱.۱.۱: تصویر اصلی و تاثیر اعمال box filter با اندازه ۳*۳ و ۵*۵ و ۷*۷ و ۱۱*۱۱ بر آن

همانطور که شکل ۱.۱.۱ مشاهده می شود، با افزایش سایز فیلتر، میزان مات شدن بسیار افزایش می باید.

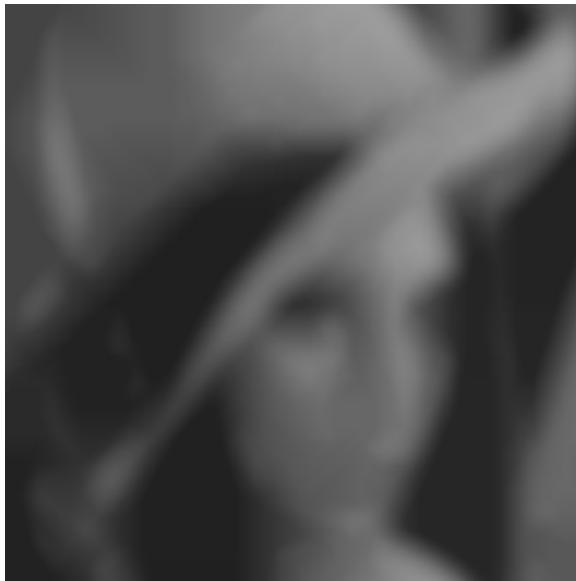
به علت سادگی باکس فیلتر و پیچیدگی محاسباتی کمی که دارد، معمولاً برای کارهایی استفاده می شود که سرعت در آنها بسیار مهم است و معمولاً خروجی آنها از دید ناظر انسانی قابل قبول است. همچنین وقتی میخواهیم لبه های یک عکس را روان تر کنیم، گزینه بسیار خوبی هستند. هر چند که باکس فیلتر ها محدودیت هایی دارند، که آنها را در برخی کاربردها به شدت ضعیف کرده است.

به طور مثال لنز دوربین فوکوس نشده (مانند شکل ۱.۱.۲) را عموماً با یک فیلتر پایین گذر مدل می کنند. اما باکس فیلتر تخمین سیار ضعیفی برای نحوه مات کردن دوربین فوکوس نشده است.

برای درک بهتر، یک نور شدید را در نظر بگیرید که با یک دوربین که تنظیم نشده است، در حال عکسبرداری از آن هستیم. چیزی که مشاهده می کنیم یک توده دایره ای شکل سفید غیر واضح است(شکل ۱.۱.۲). در حالیکه با اعمال باکس فیلتر، خروجی توده ای مربعی شکل است.(شکل ۱.۱.۳)



شکل ۱.۳.۱: تصویر پس از ۵ بار اعمال فیلتر با اندازه ۵x۵



شکل ۱.۳.۲: تصویر پس از ۱۰۰ بار اعمال فیلتر با اندازه ۵x۵ کاهش یافته

تصویر خواهیم بود. در صورتی که خروجی بار ۱۰۰۰ و ۲۰۰۰ با اینکه فاصله بسیار بیشتری دارند، به هم بسیار نزدیک است.

نکته‌ی دیگری که باید به آن توجه کرد، این است که با افزایش تعداد دفعات اعمال فیلتر، همانطور که ویژگی منفی مورد نظر کم می‌شود، برخی از نقاط قوت عکس هم از بین می‌رود. ممکن است این عمل رفته رفته باعث از بین رفتن عکس بشود، به طوری که در نهایت مجموعه از پیکسل های نامربوط بماند. پس نکته مهم دیگری که قابل توجه است این است که به ازای از بین بردن یک ویژگی منفی، تا چه حد ویژگی های مثبت و ویژگی های مهم عکس را نیز از بین می‌بریم.

قسمت ۳.۱.۳:

در مورد box filter در قسمت های قبل به صحبت کردیم. همانطور که در قسمت اولیه توضیح دادیم، این فیلتر به جای هر پیکسل، میانگینی ثابت از پیکسل های همسایه می‌گذارد. همانطور که قابل حدس است، انجام این کار باعث می‌شود پیکسل هایی که در یک همسایگی هستند، مقادیری نزدیک به هم پیدا کنند، که یعنی جزیيات تصویر و لبه ها کمرنگ می‌شود.

برای اعمال فیلتر، پس از مشخص کردن سایز فیلتر، کافی است مرکز فیلتر را بر روی همه پیکسل ها قرار دهیم و به جای پیکسل مرکزی، میانگین مقدار تمام پیکسل های داخل آن پنجره را جایگزین کنیم.

تنها مشکلی که در این قسمت وجود دارد، پیکسل های لبه است که بخشی از همسایگی آنها خارج از تصویر اصلی می‌افتد.

در حالت عادی و با فیلتر های کوچک این مسئله شاید مشکل بزرگی نباشد، اما هنگامی که که مانند این مسئله نیاز به اعمال چند باره ای فیلتر داریم، انجام ندادن این کار باعث کاهش سایز عکس پس از هر مرحله می‌شود و با توجه به اینکه تعداد مراحل زیاد است، در آخر سایز عکس به شدت کوچک می‌شود. شکل ۱.۳.۱ و ۱.۳.۲ نتیجه را در صورتی که این مشکل را رفع نکنیم نشان می‌دهد. همانطور که مشاهده می‌شود، شکل ۱.۳.۲ بسیار کوچکتر شده است.

به همین علت خوب است از padding استفاده کنیم. عملیاتی است که در آن برای هر عکس Padding خاصی در نظر می‌گیریم.

برای این سوال اگر هر بار پس از اعمال فیلتر، padding با اندازه یک را دور تصویر اعمال کنیم، مشکل کوچک شدن عکس را حل کرده ایم.

برای انجام padding، روش های مختلفی وجود دارد. یکی از محبوب ترین روش ها zero-padding است. این روش دور عکس را پیکسل هایی با شدت حاکستری صفر می‌پوشاند.

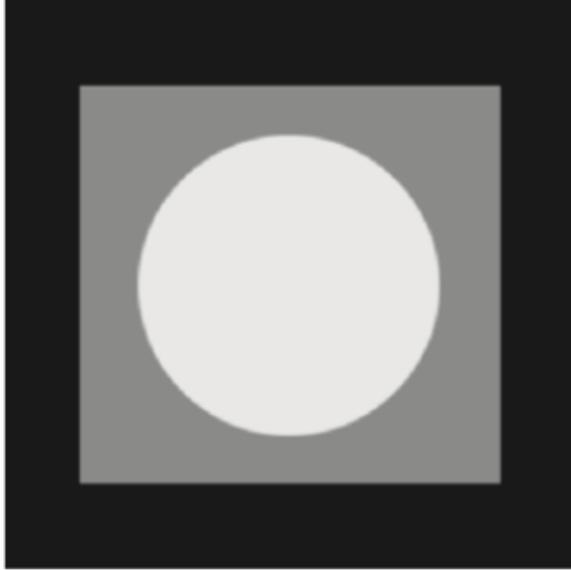
$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \quad -\infty < z < \infty$$

که z نمایانگر شدت خاکستری، σ میانگین شدت خاکستری و برابر مقدار standard deviation می باشد.

در هنگام تصویربرداری، امکان بوجود آمدن نویز گاووسی وجود دارد. این نویز می تواند به دلیل وجود اختلالاتی در سنسورهای دوربین یا تاریکی بیش از حد فضای باشد. به دلیل انعطاف پذیری بالای این نویز، چه در حوزه مکان و چه در حوزه فرکانس، این نویز به شکل گسترش ای در مدل کردن استفاده می شود. در نویز گاووسی، یکتابع گاووسی که در معادله قبل آن را توضیح دادیم به عکس اضافه می شود:

$$g(x,y) = f(x,y) + \eta(x,y)$$

شكل های پایین قبل و پس از افزودن این نویز به عکس را نشان می دهد.



شکل ۱.۵.۱: تصویر اصلی

روش های دیگری نیز مانند mirror و reflective وجود دارد. در این روش ها پیکسل های خارج از عکس را با مقادیری از خود عکس پر می کنیم.

قسمت ۳.۲.۱

order-statistic فیلتر های غیر خطی هستند که خروجی آنها بر اساس ترتیب (rank) پیکسل ها در آن محدوده‌ی تحت پوشش فیلتر می باشد. از کاربردهای این فیلتر ها، smoothing هستند که با جایگذاری مقدار پیکسل مرکزی با مقدار حاصل از ranking بدست می آید.

از معروفترین فیلترها در این حوزه، هستند، که همانطور که از نامشان مشخص است، مقدار پیکسل مرکزی را با میانه ای اعداد تحت پوشش فیلتر جایگزین می کند.

میانه مجموعه ای از اعدادی، عددی است که دقیقاً نصف اعداد مقداری کمتر از آن و نصف دیگر اعداد مقداری بیشتر از آن دارند.

فیلتر median توانایی قابل توجهی در کاهش میزان نویزهای تصادفی یک عکس دارد. به علاوه با وجود کاهش شدید نویز از فیلتر های خطی، تصویر را کمتر مات می کند. فیلتر median تاثیر قابل توجهی بر روی عکس هایی با نویز salt-and-paper دارد.

نویز salt-and-paper یا برفکی نوعی از نویز هست که غالباً بر روی تصویر دیده می شود. به این نویز impulse هم گفته می شود.

این نویز عموماً به علت اغتشاشات آنی و سریع بر روی سیگنال تصویر رخ می دهد. این نویز خودش را به شکل پیکسل های سفید و سیاه به صورت پراکنده در تصویر نشان می دهد.

دلیل تاثیر گذاری این فیلتر بر روی نویز salt-and-paper این است که با توجه به اینکه پیکسل های نویزی، کمترین مقدار و بیشترین مقدار ممکن را دارند، احتمال انتخاب آنها به عنوان عضو میانه به شدت پایین است. بنابراین به جای این پیکسل های نویزی، عموماً پیکسلی از همان منطقه استفاده می شود، که باعث افزایش کیفیت عکس و رفع نویز می شود. تاثیر افزایش سایز فیلتر و میزان نویز تصویر به تفصیل در ادامه نتایج داده می شود.

قسمت ۳.۲.۲

در مورد فیلتر های order-statistic در قسمت قبل توضیح دادیم.

در قسمت های قبلی نیز درباره فیلترهای هموار کننده، به خصوص box filter صحبت کردیم.

نویز گاووسی، نویزی است تراکمی برابر با توزیع نرمال دارد. تابع چگالی نویز برای یک متغیر تصادفی گاووسی از رابطه زیر بدست می آید:

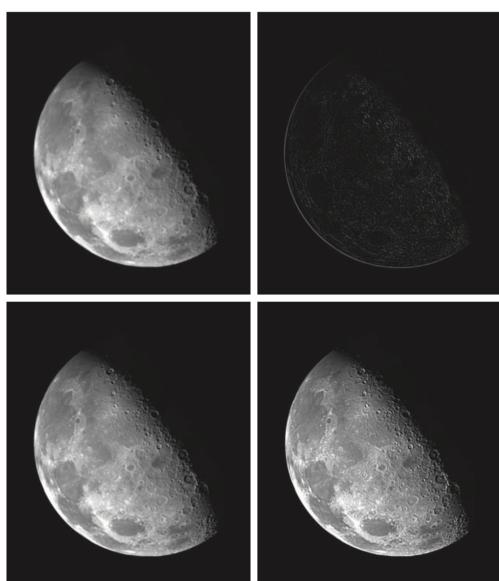
در این قسمت می تواند کاری مفید باشد در مورد فیلترهای میانگین گیر در قسمت های قبل به تفصیل توضیح دادیم:

- افزایش کیفیت تصویر با استفاده از لاپلاسین: پس از انجام عملیات رفع نویز با استفاده از فیلتر میانگین گیر، تصویر کمی مات می شود. علاوه بر آن با توجه به شرایط نوری بدی که عکس در آن گرفته شده است عکس خروجی از مرحله قبل به شدت مات است. برای رفع این مات بودن می توان از فیلتر Laplacian استفاده کرد. لاپلاسین فیلتری به شکل زیر است که جز دسته فیلترهای لبه یاب با استفاده از مشتق می باشد:

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

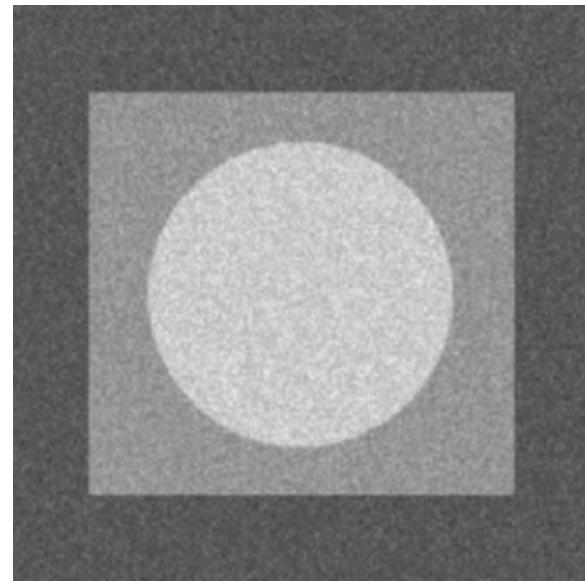
شکل ۱۶.۱: دو فیلتر لاپلاسین

خروجی این فیلتر تصویری است که در آن لبه ها مقدار زیادی دارند و بقیه قسمت ها مقادیر کمتری دارند. بنابراین افزودن این تصویر به تصویر اصلی باعث کاهش مات بودن می شود. شکل ۱۶.۲ این عملیات را نشان می دهد.



شکل ۱۶.۱: مراحل sharp کردن تصویر

قسمت ۳.۴.۱: تشخیص لبه های عمودی و افقی و فطری از مهمترین کاربردهای پردازش تصویر است.



شکل ۱۶.۲: تصویر پس از افزودن نویز گاووسی

برای حذف این نویز از فیلتر های متفاوتی می توان استفاده کرد. در این مسئله ما دو فیلتر box و میانه را بررسی می کیم.

توضیحات این دو فیلتر در بخش های بالاتر آمده است و نتیجه در قسمت نتایج به تفصیل توضیح داده شده است.

قسمت ۳.۳.۱:

در این قسمت از ما خواسته شده یک تصویر مات با نویز گرفته با استفاده از روش های مختلف سعی در بهبود آن داشته باشیم.

یکی از راه های انجام این کار گرفتن عکس در شرایط نوری بد (بسیار تاریک) است. دوربین هایی معمولی در شرایط نوری بد عموما تصاویر به شدت مات و با نویز زیاد می گیرند. در ادامه سوال خواسته شده است که این تصویر را با استفاده از روش هایی که تا به حال با آنها آشنا شده ایم، بهبود ببخشیم.

برای بهبود عکس و افزایش کیفیت آن عملیات زیر را انجام می دهیم:

- انجام عملیات همسان سازی هیستوگرام: همانطور که توضیح داده شده، عکس در شرایط نوری نامناسبی گرفته شده است. بنابراین ابتدا و قبل از هر عملی برای واضح شدن عکس و افزایش کنتراست ان نیاز به همسان سازی هیستوگرام داریم. قبل از مورد اینکه چرا همسان سازی هیستوگرام باعث افزایش کنتراست می شود به تفصیل توضیح داده ایم.

- رفع نویز با استفاده از فیلتر میانگین گیر: می دانیم که تصویر پس از افزایش کنتراست، دارای نویزی های زیادی است. برای از بین بردن این نویز، باید از فیلتر هایی که این کار را انجام میدهند استفاده کنیم. پس از افزایش کنتراست، مشاهده شد که تصویر، نویزی مشابه به نویز گوسی دارد. بنابراین و با توجه به نتیجه گیری قسمت ۳.۲.۲، استفاده از فیلترهای میانگین گیر

$$\frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

این فیلتر نیز همانند فیلتر قبلی در همسایگی به فاصله یک افقی و عمودی عمل می کند، تنها تفاوت این فیلتر این است به همسایگی نزدیکتر، وزن بیشتری می دهد.
پس از اعمال این فیلتر نیاز به عملیات scale می باشد.

مشتق اول در پردازش تصویر را با استفاده از ضرایب بدست آمده از گرادیان محاسبه می کنیم.
گرادیان یک تصویر مانند f در مختصات (x,y) یک بردار با اندازه ۲ به شکل زیر است:

$$\nabla f \equiv \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

این بردار ویژگی های هندسی مهمی دارد و به بیشترین تغییر تصویر f در مختصات (x,y) اشاره می کند.
استفاده از مشتق برای تشخیص لبه بسیار رایج است.
با استفاده از دو فرمول زیر، فیلتر های سوبل برای تشخیص لبه بدست می آید که همان فیلتر قسمت ۳ می باشد.
برای تشخیص لبه افقی:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

برای تشخیص لبه عمودی:

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

که در نهایت فیلتر های آن شکل مشابه پایین دارند:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

شکل ۱.۷.۱: فیلتر های سوبل
بنابراین از بین فیلتر های ورودی میتوان از ۵ به عنوان محاسبه گر مشتق استفاده کرد.

برای انجام این کار، روش های متفاوتی است از جمله تبدیل فوریه و کار کردن در حوزه فرکانس.
اما در حوزه مکان هم، فیلتر هایی هستند که این کار را انجام می دهند.
در این سوال به بررسی سه فیلتر از مهمترین فیلترهای تشخیص لبه افقی می پردازیم:
۱:

$$\frac{1}{2} [1 \quad 0 \quad -1]$$

همانطور که از ظاهر این فیلتر پیدا است، این فیلتر میانگین تفاضل دو پیکسل در همسایگی افقی را بر میگرداند. می دانیم از ویژگی های لبه این است که جدا کننده نوعی اختلاف است، به این معنی که در پیکسل های مجاور شاهد تغییر شدید سطح خاکستری هستیم.
به همین دلیل این فیلتر، با انجام این عملیات لبه ها را شناسایی می کند. اگر پیکسلی لبه باشد، خروجی بزرگتری را بر میگرداند (تفاضل های پیکسل اطرافش بیشتر است).
نکته ای که باید به آن توجه کرد، این است که خروجی این قسمت ممکن است خارج از محدود ۰ تا ۲۵۵ باشد. به همین دلیل نیاز به scale کردن پس از انجام عملیات می باشیم. در عملیات scale کردن، بازه ای که پیکسل ها در آن قرار دارند را به صفر تا ۲۵۵ تبدیل می کنیم.
۲:

$$\frac{1}{6} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

این فیلتر نیز همانند فیلتر یک است، فقط همسایگی بزرگتری را تحت پوشش قرار می دهد. این فیلتر می تواند تاثیر بیشتری برای پیدا کردن لبه داشته باشد، چرا که این کار را برای ناحیه بزرگتری انجام می دهد. با بررسی ماهیت لبه، میدانیم که فقط یک پیکسل لبه افقی نمی باشد و پیکسل های مجاوار افقی از نیز حتما لبه هستند. با در نظر گرفتن این ویژگی این فیلتر تهیه شده است که این نکته را در نظر گرفته است. پس از اعمال این فیلتر نیاز به عملیات scale می باشد.
۳:

قسمت ۳.۵.۱:

یکی از قدیمی ترین کاربردهای پردازش تصویر، sharp کردن تصویر با استفاده از فیلتر های smothe کننده است. در این روش به شکل زیر عمل می کنیم:

۱. تصویر اصلی را blur می کنیم. این کار را میتوان با استفاده از فیلترهای میانگین گیر انجام داد.
۲. تصویر مات شده را از تصویر اصلی کم می کنیم. با توجه به اینکه در تصویر مات شده لبه ها کمرنگ شده است، این تفاضل دارای لبه های پررنگی است. باید توجه کرد که blur کردن تصویر فقط لبه ها را از بین نمی برد، بلکه ممکن است برخی از جزییات را نیز از بین ببرد که در نتیجه عکس حاصل تفاضل شامل برخی جزییات ناخواسته نیز می باشد.
۳. این تفاضل را با وزن دلخواه به عکس اضافه می کنند. در نتیجه لبه ها در عکس تقویت می شود. عملیات بالا در قابل دو معادله در پایین آمده است:

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

که عملیات بدست آوردن mask یا همان تفاضل است.

$$g(x, y) = f(x, y) + kg_{\text{mask}}(x, y)$$

که افزودن mask با ضریبی بین [۰, ۱] به تصویر است.

شکل ۱.۹.۱ نیز نشان دهنده خروجی این عملیات است:



شکل ۱.۹.۱: تصویر اصلی - تصویر پس از smooth کردن - mask - خروجی با ضریب یک - خروجی با ضریب بیشتر از یک در بخش نتایج، خروجی این بخش را بیشتر بررسی می کنیم.

قسمت ۳.۴.۲:

همانطور که در قسمت قبل توضیح دادیم، با استفاده از مشتق می توان لبه های یک تصویر را بدست آورد. تعریف زیر توسط Roberts پیشنهاد شده است، و از آن به عنوان cross differences یاد می شود:

$$g_x = (z_9 - z_5) \quad \text{and} \quad g_y = (z_8 - z_6)$$

با ساده کردن عبارت بالا، به عبارت زیر خواهیم رسید:

$$M(x, y) = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2 \right]^{1/2}$$

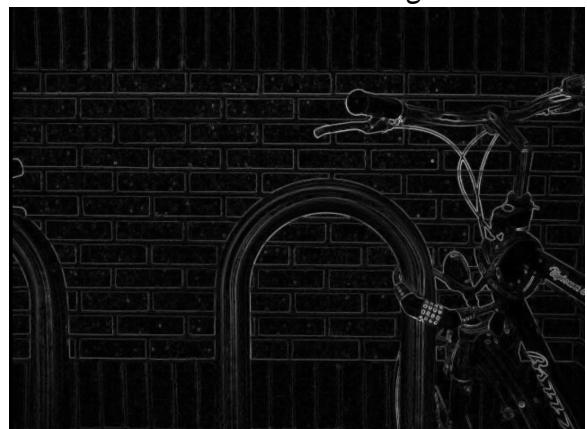
که برابر است با :

$$M(x, y) \approx |z_9 - z_5| + |z_8 - z_6|$$

بنابراین این دو فیلتر را می توان به شکل ۱.۸.۱ زیر نوشت:

-1	0	0	-1
0	1	1	0
0	1	1	0

شکل ۱.۸.۱: فیلتر های روبرت
این فیلتر ها برای پیدا کردن لبه های قطری استفاده می شوند. البته این فیلتر ها لبه های عمودی و افقی را نیز پیدا میکنند، اما در نهایت وزن لبه های قطری بسیار بزرگتر است. (همانند شکل ۱.۸.۲)



شکل ۱.۸.۲: تصویر پس از اعمال فیلتر روبرت

در قسمت نتایج، نتیجه این فیلتر را بر روی عکس Lena بررسی می کنیم و خروجی این قسمت را با خروجی قسمت قبل مقایسه می کنیم.

نتایج:

قسمت ۳.۱.۳:

همانطور که در شرح توضیح داده شد، box filter یکی از ساده ترین فیلترها برای انجام عملیات smoothing است. در حین این عملیات، عکس مات شده و جزیات آن از بین می رود. اجرای چند باره این عملیات باعث می شود عکس رفته رفته مات ترشود، لبه ها کمرنگ شود و جزیات ناواضح شود.

شکل ۲.۱.۱ تصویر اصلی را نشان می دهد. از مقایسه تصویر اصلی با شکل ۲.۱.۲ که تصویر را پس از ۵ بار اعمال باکس فیلتر نشان می دهد، متوجه می شویم، اعمال چند باره این

فیلتر باعث افزایش ماتی تصویر می شود.

همانطور که در تصاویر پیداست، افزایش تعداد اعمال فیلتر ها باعث کاهش جزیبات و افزایش ماتی می شود. اما این روند رفته رفته کند می شود. به طور مثال شکل ۲.۱.۲ و ۲.۱.۳ تفاوت محسوسی دارند، در حالیکه یکی با ۵ بار اعمال

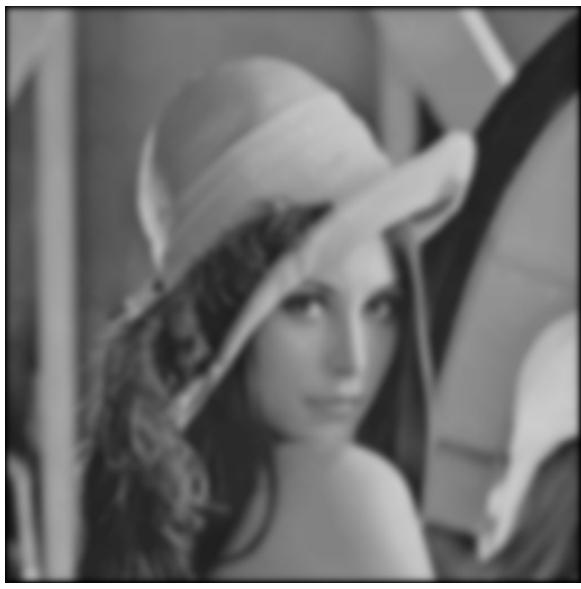
فیلتر، و دیگری با ۲۰ بار اعمال فیلتر است.

این در حالی است که عکس ۲.۱.۵ و ۲.۱.۴ تفاوت آنچنانی ندارند، در حالی که برای یکی ۱۰۰ بار فیلتر بیشتر اعمال شده است.

بنابراین افزایش تعداد اعمال فیلتر تا جایی موثر است و پس از آن تأثیر چندانی ندارد.



شکل ۲.۱.۲: خروجی عکس پس از ۲۰ بار اعمال کردن box filter با سایز ۳



شکل ۲.۱.۳: خروجی عکس پس از ۴۰ بار اعمال کردن box filter با سایز ۳



شکل ۲.۱.۱: خروجی عکس پس از ۵ بار اعمال کردن box filter با سایز ۳

شدت پایین می آید. این موضوع در شکل ۲.۲.۱ و ۲.۲.۲ و ۲.۲.۳ قابل پیگیری است.

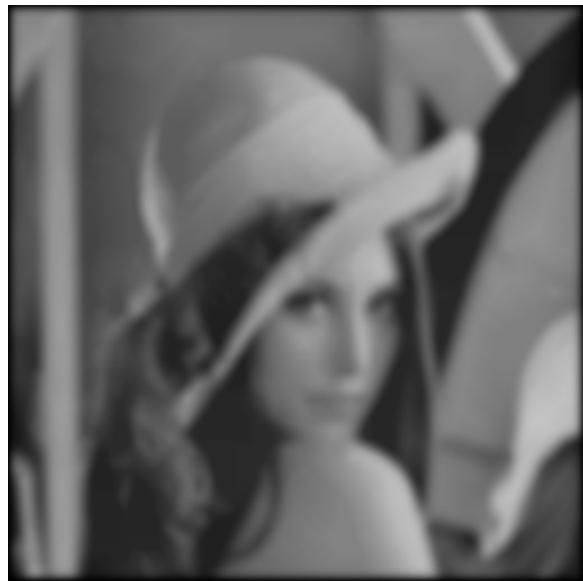
همانطوری که دیده می شود، با نویز بیشتر، عکس کیفیت بدتری دارد و شباهت کمتری به عکس اصلی دارد.



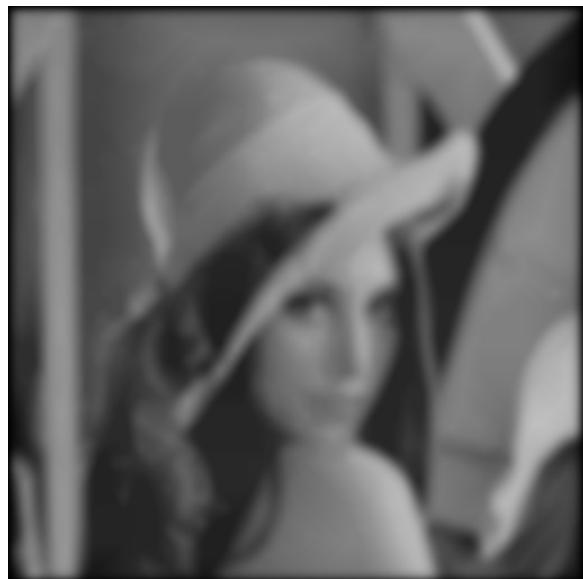
شکل ۲.۲.۱: تصویر با نویز salt-and-paper با چگالی ۰۰۵



شکل ۲.۲.۲: تصویر با نویز salt-and-paper با چگالی ۰۱



شکل ۲.۱.۴: خروجی عکس پس از ۸۰ بار اعمال کردن box filter با سایز ۳



شکل ۲.۱.۵: خروجی عکس پس از ۲۰۰ بار اعمال کردن box filter با سایز ۳

قسمت ۳.۲.۱:

در قسمت شرح به بررسی ویژگی های نویز salt-and-paper پرداختیم. درباره ای فیلتر median که فیلتری قدرتمند برای حذف نویز است نیز صحبت کردیم. در این قسمت مشخصات هر یک از این دو مورد بحث قرار می دهیم.

یکی از نکات مهم در نویز salt-and-paper، این است که چه مقدار از عکس در گیر این نویز شده است. طبیعتاً هر چه درصد بیشتری از عکس دچار نویز باشد، کیفیت عکس به



شکل ۲.۲.۵: تصویر با نویز salt-and-paper با چگالی ۰.۱ و اعمال فیلتر میانه با سایز ۳



شکل ۲.۲.۶: تصویر با نویز salt-and-paper با چگالی ۰.۰ و اعمال فیلتر میانه با سایز ۳

با اندازه‌ی فیلتر ۳، کیفیت تصویر حفظ شده است و تصویر به شکل نامحسوسی تار شده است، اما برای چگالی‌های بالاتر همانند ۰.۶۲۶ اگر دقت شود، نویز به شکل کامل از بین نرفته است. برای چگالی‌ها پایینتر این سایز فیلتر کاملاً مناسب است. اما اگر چگالی نویز باشد، استفاده از فیلتر‌های بزرگتر توصیه می‌شود



شکل ۲.۲.۳: تصویر با نویز salt-and-paper با چگالی ۰.۲

اعمال روی این قسمت، ابتدا خروجی را با فیلتر با سایز های متفاوت را مشاهده می‌کنیم. در انتهای با بررسی مقدار MSE نتیجه را مورد بحث قرار می‌دهیم.



شکل ۲.۲.۴: تصویر با نویز salt-and-paper با چگالی ۰.۰۵ و اعمال فیلتر میانه با سایز ۳



شکل ۲.۲.۹: تصویر با نویز salt-and-paper با چگالی ۰.۲ و اعمال فیلتر میانه با سایز ۵



شکل ۲.۲.۷: تصویر با نویز salt-and-paper با چگالی ۰.۰۵ و اعمال فیلتر میانه با سایز ۵

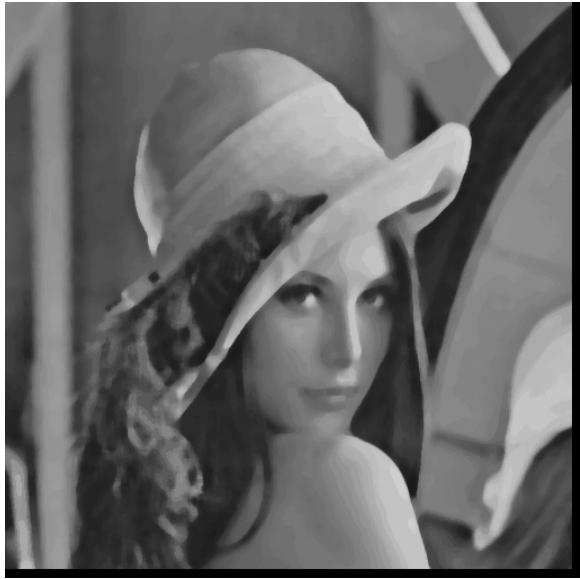


شکل ۲.۲.۱۰: تصویر با نویز salt-and-paper با چگالی ۰.۰۱ و اعمال فیلتر میانه با سایز ۵

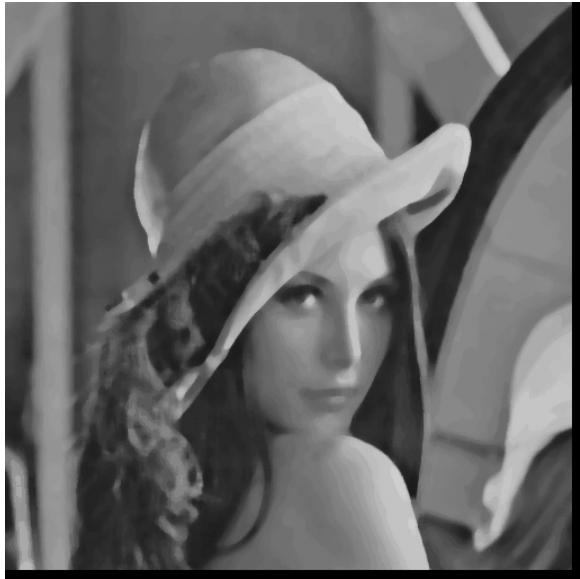


شکل ۲.۲.۸: تصویر با نویز salt-and-paper با چگالی ۰.۰۰۵ و اعمال فیلتر میانه با سایز ۷

با این سایز پنجره، ماتی تصویر کاملا مشهود است. به همین دلیل استفاده از این سایز فیلتر خیلی پیشنهاد نمی شود. مگر چگالی خیلی خیلی بالا باشد.



شكل ۲.۱۳: تصویر با نویز salt-and-paper با چگالی ۰.۰۵ و اعمال فیلتر میانه با سایز ۹



شكل ۲.۱۴: تصویر با نویز salt-and-paper با چگالی ۰.۱ و اعمال فیلتر میانه با سایز ۹



شكل ۲.۱۱: تصویر با نویز salt-and-paper با چگالی ۰.۱ و اعمال فیلتر میانه با سایز ۷



شكل ۲.۱۲: تصویر با نویز salt-and-paper با چگالی ۰.۲ و اعمال فیلتر میانه با سایز ۷

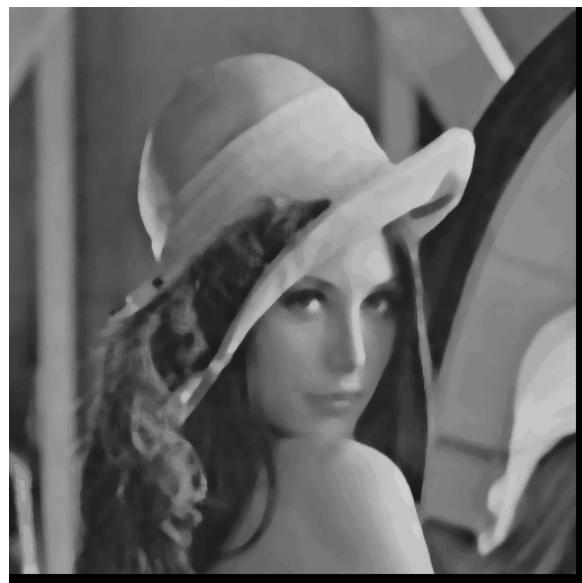
همانطور که در شرح در مورد این نویز توضیح دادیم، این نویز یک توزیع گوسی را به عکس می‌افزاید. در ابتدا عکس‌های اصلی و بدون اعمال فیلتر را بررسی می‌کنیم.



شکل ۲.۳.۱: شکل اصلی با نویز gaussian با میانگین ۰.۱



شکل ۲.۳.۲: شکل اصلی با نویز gaussian با میانگین ۰.۵



شکل ۲.۲.۱۵: تصویر با نویز salt-and-paper با چگالی ۰.۰۹ و اعمال فیلتر میانه با سایز ۹

	3	5	7	9	0
0.05	37.0647583	48.5505791	54.8483925	59.3949699	5.24665833
0.1	37.087265	48.4616966	54.8757057	59.2766609	5.21737289
0.2	37.1128006	48.4633484	54.9268875	59.3373985	5.17616272

جدول ۲.۱: مقایسه mse سایز فیلتر های متفاوت و چگالی های متفاوت

با افزایش سایز فیلتر میانه، تصویر مات می‌شود. همانطور که مشاهده می‌شود، رفته رفته عکس‌ها تارتر می‌شود و به ماتی آن افزوده می‌شود.

واضح است که با افزایش سایز پنجره، یک پیکسل ممکن است مقدار پیکسلی در همسایگی دورتر از خودش را اختیار کند، به همین دلیل عکس با فیلتر های بزرگتر مات تر است. اما به همین دلیل، با انتخاب همسایگی کوچک، ممکن است بیش از نصف پیکسل های همسایگی نویز باشند، که در اینصورت یک پیکسل همچنان نویز می‌ماند.

همانطور که در جدول ۲.۱ قابل مشاهده است، با افزایش سایز فیلتر تصویر از تصویر اصلی دورتر می‌شود. نکته قابل مشاهده از جدول ۲.۱ این موضوع است که با فیلتر های بزرگتر، میزان نویز تاثیر کمتری دارد. به بیان دیگر سه تصویر خروجی به ازای مقادیر مختلف چگالی بسیار به هم شبیه هستند. بنابراین هرچه سایز فیلتر بزرگتر شود، خروجی فارغ از میزان نویز تارتر و مات تر می‌شود.

با توجه به مقایسه نتایج، میتوان نتیجه گرفت در صورتی که حجم نویز بالا باشد(بیشتر از ۰.۰۵) استفاده از فیلتر با اندازه متوسط (۵ مثلا) گزینه مناسبی است، در حالیکه برای حجم نویز های پایین (کمتر از ۰.۰۵) فیلتر با سایز ۳ بسیار مناسب است.

قسمت ۳.۲.۳ در این قسمت خواسته شده است تاثیر فیلتر های میانه و باکس را بر روی نویز گاووسی بررسی کنیم.



شکل ۲.۳.۵: اعمال box filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۶: اعمال box filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۰۱

مشاهده می کنیم که اعمال فیلتر smooth روی نویز گاوسی تاثیر مطلوبی نداشته است و تصویر تغییر قابل مشاهده ای نکرده است.



شکل ۲.۳.۳: شکل اصلی با نویز gaussian با میانگین ۰.۰۱ همانطور که مشخص است با افزایش میزان توزیع نویز در عکس، عکس دارای کیفیت بدتری خواهد بود. در شکل ۲.۳.۳ تصویر تا حد قابل قبولی واضح است و جزیات مشخص است. اما در تصویر ۲.۳.۱ که دارای توزیع بیشتری می باشد بافت تصویر بسیار زیاد از بین رفته است و جزیات تصویر ابدا قابل تشخیص نیست.



شکل ۲.۳.۴: اعمال box filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۹: اعمال box filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۰۱ با افزایش سایز فیلتر به ۵، نتیجه کمی بهتر شده است اما نتیجه همچنان مطلوب نیست.



شکل ۲.۳.۱۰: اعمال box filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۷: اعمال box filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۸: اعمال box filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۱۳: اعمال box filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۱۴: اعمال box filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۱۱: اعمال box filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۱۲: اعمال box filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۰۱
استفاده از فیلتر باکس با اندازه ۷ نیز نتیجه قابل قبولی نداشته است.



شکل ۲.۳.۱۷: اعمال median filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۱۸: اعمال median filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۱۹: اعمال box filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۰۱
استفاده از فیلتر باکس با اندازه ۹ نیز نتیجه قابل قبولی نداشته است.
در ادامه به بررسی فیلتر میانه می پردازیم.



شکل ۲.۳.۲۰: اعمال median filter با اندازه ۳ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۲۱: اعمال median filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۰۱



شکل ۲.۳.۲۲: اعمال median filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۱۹: اعمال median filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۰۱



شکل ۲.۳.۲۰: اعمال median filter با اندازه ۵ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۲۵: اعمال median filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۱



شکل ۲.۳.۲۶: اعمال median filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۰۱
فیلتر میانه نیز تاثیر قابل توجهی بر روی تصاویر با نویز گوسی ندارد.
به طور کل هیچ کدام از فیلترهای اعمال شده نتیجه را به طور قابل توجهی تغییر نمی دهند.



شکل ۲.۳.۲۳: اعمال median filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۰۵



شکل ۲.۳.۲۴: اعمال median filter با اندازه ۷ بر روی عکس میانگین نویز ۰.۰۱

یک شدت نویزی شده اند(حدودا) به همین دلیل استفاده از فیلتر با سایز بزرگتر توصیه می شود(مثلا ۵).

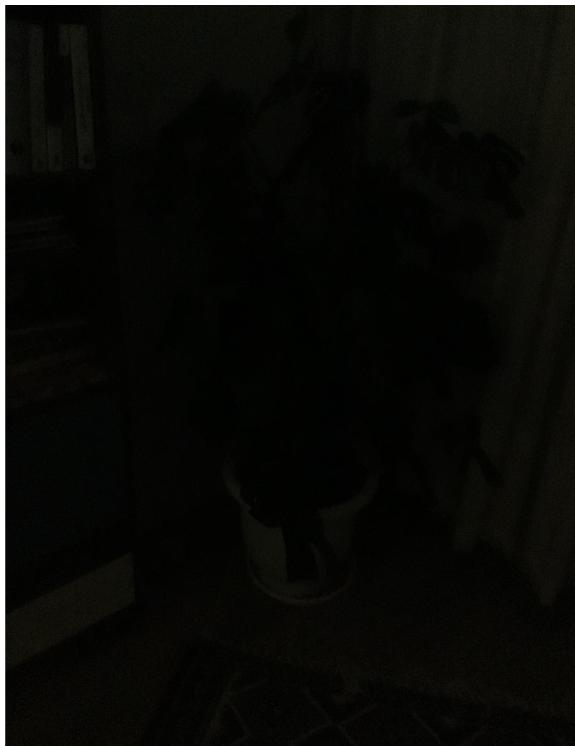
اما بزرگتر کردن سایز فیلتر همیشه نمی تواند راهکار مناسبی باشد، زیرا همانطور که می تواند پیکسل ها با شدت نویز زیاد را با میانگین گیری بهتر کند، می تواند پیکسل هایی که خیلی درگیر نویز نیستند را با میانگین گیری خراب کند. به همین دلیل سایز ۵ برای این فیلتر سایز ایده آل و قابل قبولی است.

همانطور که توضیح داده شد، فیلتر میانه در این نویز قوی عمل نمی کند و شاهد نتیجه های بدتری برای این فیلتر هستیم.

برای این فیلتر می تواند حدودا گفت که با افزایش سایز فیلتر، شاهد افزایش کیفیت هستیم. دلیل این امر این است که با گرفتن همسایگی بزرگتر، امکان انتخاب پیکسل هایی که آسیب کمتری دیده اند بیشتر می شود. البته این بیشتر شدن تا حدی باعث افزایش کیفیت می شود و افزایش بیش از حد آن نیز می تواند باعث به هم ریختن عکس شود.

۳.۱.۳: قسمت

همانطور که در شرح توضیح داده شد، تصویر اولیه در محیطی تاریک گرفته شد. شکل ۲.۴.۱ این تصویر را نشان می دهد.



شکل ۲.۴.۱: تصویر اصلی



شکل ۲.۴.۲: اعمال median filter با اندازه ۹ بر روی عکس میانگین نویز ۰.۰۱

	box 3	box 5	box 7	box 9
0.01	66.3806229	61.1397705	62.2559128	64.6738014
0.05	87.4296532	78.8724556	76.0367928	75.260437
0.1	94.9003372	87.9262428	85.1278839	84.398716

جدول ۳.۱: مقایسه اعمال فیلتر باکس با سایز های مختلف بر روی نویز با توزیع مختلف گاوی

	no filter	median 3	median 5	median 7	median 9
0.01	94.2108116	72.4542122	65.154686	64.3581047	65.6634369
0.05	105.180576	93.5499268	84.2264442	79.035759	76.3841858
0.1	106.147224	100.25882	92.2835999	86.8811798	83.6309814

جدول ۳.۲: مقایسه اعمال فیلتر میانه با سایز های مختلف بر روی نویز با توزیع مختلف گاوی

در ابتدا مشاهده می کنیم که تاثیر اعمال نویزی به شدت زیاد است و عکس نویزی با عکس اصلی دارای تفاوت زیادی

است. اعمال هر کدام از فیلتر های نیز نتیجه در mse به شکل قابل توجهی بهبود می بخشند.

همانطور که از مقایسه دو جدول درک است، برای نویز گوسی فیلتر باکس عملکرد بهتری دارد. دلیل این عمل این است که برخلاف نویز نمک-فلقل این نویز تقریبا بر تمام پیکسل های عکس تاثیر میگذارد و نمی توان مطمئن بود که میانه ی یک همسایگی، پیکسلی سالم است. چیزی که برای نویز نمک-فلقل صادق بود.

میان فیلتر های باکس با سایز های مختلف، فیلتر با سایز ۵ بهترین نتیجه را می دهد. با توجه به اینکه همسایگی سه، همسایگی کوچکی است، تمام پیکسل ها در این منطقه با



شکل ۲.۴.۲: تصویر پس از حذف نویز ها و اعمال فیلتر میانگین گیر



شکل ۲.۴.۳: تصویر پس از اعمال لابلسین و جمع آن با تصویر اصلی



شکل ۲.۴.۱: تصویر اصلی پس از انجام عملیات نرمال کردن هیستوگرام سپس با استفاده از فیلتر های میانگین سر که فیلتر های مناسبی برای حذف نویزهای شبه گاوی هستند، فیلتر ها را حذف می کنیم.

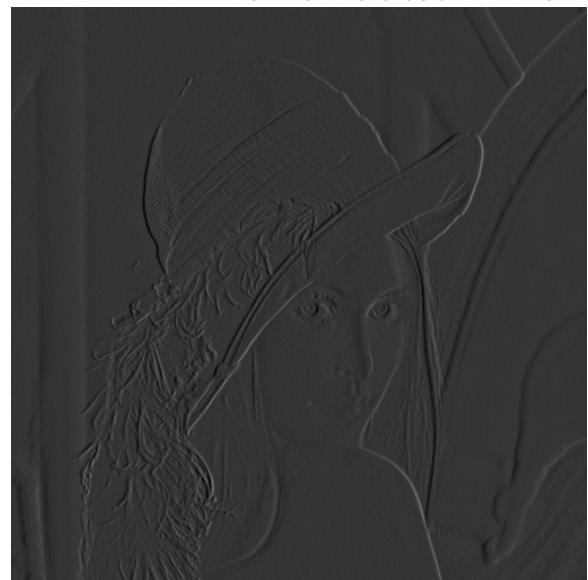
همانطور که مشاهده می شود، پس از اعمال لابلائین و جمع آن با تصویر اصلی، تصویر دارای لبه های بسیار واضح تری است و به طور کل از مات و تار بودن آن کم شده است.

۳.۴.۱: قسمت

در قسمت شرح توضیح دادیم که این فیلترها لبه های افقی را پیدا می کند.



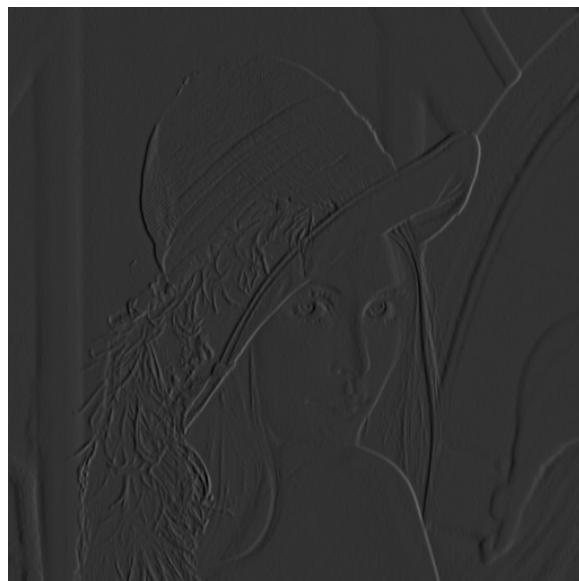
شکل ۲.۵.۱: تصویر پس از اعمال فیلتر a



شکل ۲.۵.۲: تصویر پس از scale کردن تصویر ۲.۵.۱



شکل ۲.۵.۳: تصویر پس از اعمال فیلتر b



شکل ۲.۵.۴: تصویر پس از scale کردن تصویر ۲.۵.۳

قسمت ۳.۴.۲:



شکل ۲.۶.۱: اعمال فیلتر a بر تصویر Lena و scale کردن



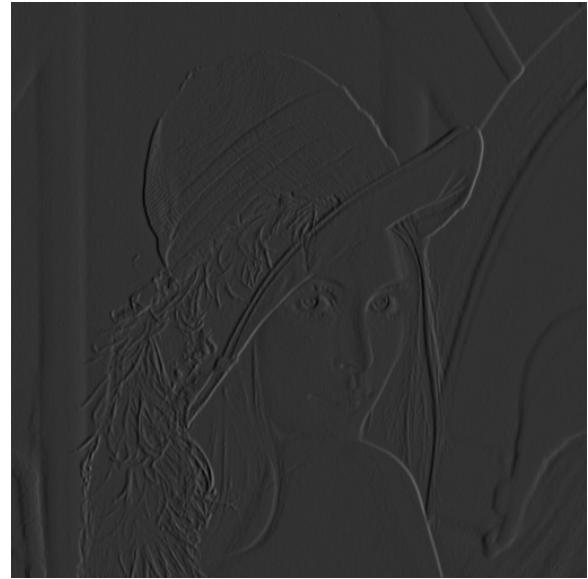
شکل ۲.۶.۲: اعمال فیلتر b بر تصویر Lena و scale کردن

شانه‌ی دختر یک لبه قطري اصلی است و آن را در دو عکس ۲.۶.۲ و عکس ۲.۶.۱ بررسی ميکنیم. همانطور که مشخص است که در شکل ۲.۶.۲ اين لبه به شکل مشخص تری آمده است. اما کلاه که لبه قطري فرعی است در شکل ۲.۶.۳ به شکل مشخص تری آمده است.

مقایسه خروجی ۲.۶.۱ با ۲.۵.۲ نشان می دهد که شکل ۲.۶.۱ لبه‌های قطري را واضح تر نشان می دهد در حالیکه سه فیلتر قبلی بيشتر لبه‌های افقی را تشخيص داده اند.



شکل ۲.۵.۵: تصویر پس از اعمال فیلتر c



شکل ۲.۵.۶: تصویر پس از scale کردن تصویر ۲.۵.۵

همانطور که مشاهده می شود، همه‌ی این سه فیلتر لبه‌ها را پیدا می کنند و لبه‌های افقی را بهتر پیدا میکنند. در این میان دو فیلتر b و c عملکردی مشابه دارند. در حالیکه لبه‌های پیدا شده در تصویر حاصل از اعمال فیلتر a، کمی ضعیف تر می باشد.

قسمت : ۳.۵.۱

همانطور که قابل مشاهده است، با افزایش الگوهای عکس به شدت پررنگ می شود. زیرا لبه‌ی پررنگ تری از عکس به عکس اصلی اضافه می شود.



شکل ۳.۹.۳: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه‌ی ۳ و آلفای ۰.۵



شکل ۳.۹.۱: تصویر mask شده با استفاده از فیلتر gaussian با اندازه‌ی ۳



شکل ۳.۹.۴: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه‌ی ۳ و آلفای ۰.۷



شکل ۳.۹.۲: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه‌ی ۳ و آلفای ۰.۱



شکل ۳.۹.۷: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۵ و آلفای ۰.۵



شکل ۳.۹.۵: تصویر mask شده با استفاده از فیلتر gaussian با اندازه ۳



شکل ۳.۹.۸: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۵ و آلفای ۰.۷



شکل ۳.۹.۶: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۵ و آلفای ۰.۱



شکل ۳.۹.۱۱: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۷ و آلفای ۰.۵



شکل ۳.۹.۱۲: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۷ و آلفای ۰.۷



شکل ۳.۹.۹: تصویر mask شده با استفاده از فیلتر gaussian با اندازه ۷



شکل ۳.۹.۱۰: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۷ و آلفای ۱



شکل ۳.۹.۱۵: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۹ و آلفای ۰.۵



شکل ۳.۹.۱۳: تصویر mask شده با استفاده از فیلتر gaussian با اندازه ۹



شکل ۳.۹.۱۶: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۹ و آلفای ۰.۷



شکل ۳.۹.۱۴: خروجی عملیات unsharp با استفاده از فیلتر gaussian با اندازه ۹ و آلفای ۰.۱

می دانیم با افزایش سایز فیلتر میانگین، تصویر smoother شده و به بیان دیگر، لبه ها در آن به شدت کمنگ می شود. به همین دلیل تصویر mask دارای لبه های پرنگ تری است. افزودن لبه های پرنگ باعث می شود که تصویر شارپتر شود. بنابراین هرچه سایز فیلتر گوسی (فیلتر میانگین گیر در این قسمت) بیشتر شود، تصویر خروجی شارپ تر می شود. با مقایسه تصاویر نیز این نتیجه کاملاً تایید است.

```

import cv2 as cv
import numpy as np
import pandas
from skimage.util import *
import pandas as pd

def
apply_median_filter(img,window_size):
    w,l = img.shape
    result = np.zeros((w ,l) ,
np.uint8)
    hw = int(window_size / 2)

    for i in range(hw , w - hw):
        for j in range( hw,l - hw):
            result[i - hw][j - hw] =
np.median(np.squeeze(np.asarray(img[i-
hw:i+hw+1,j-hw:j+hw+1])))
    print(result)
    return result

def save_result(size , variance , img ,
n_img):
    path = "result/3.2.1/var" +
str(variance) + "_size" + str(size) +
".jpg"
    mse = np.square(n_img - img).mean()
    result.set_value(str(variance),
str(size) , mse)
    cv.imwrite(path , n_img)

img =
cv.imread("image/Lena.bmp",cv.IMREAD_GR
AYSCALE)

noise_var = np.array([0.05 , 0.1 ,
0.2])
window_sizes = np.array([3,5,7,9])

result = pd.DataFrame(index=['0.05' ,
'0.1' , '0.2'],columns=['nf' , '3' ,
'5' , '7' , '9'])

for noise in noise_var:
    noise_img = random_noise(img ,
mode="s&p" , salt_vs_pepper = 0.1)

```

APPENDIX:

:٣.١.٣ قسمت

```

import cv2 as cv
import numpy as np

def apply_box_filter(img,size):
    hs = int(size / 2)
    w,l = img.shape
    blur_img = np.zeros((w-size + 1,l-
size + 1 ) , np.uint8)
    for i in range(hs,w-hs):
        for j in range(hs,l-hs):
            blur_img[i - hs][j - hs] =
np.uint8(np.sum(img[i-hs:i+hs+1,j-
hs:j+hs+1]) / (size*size))
    return blur_img

size = int(input("Enter the filter
size: "))
iteration = int(input("Enter Iteration
number:"))
path = "image/Lena.bmp"

img = cv.imread(path ,
cv.IMREAD_GRAYSCALE)
w,l = img.shape
pad_img = np.zeros((w+size-1,l+size-1)
, np.uint8)
hs = int(size / 2)
pad_img[hs:w+hs,hs:l+hs] = img

path = "result/3.1/imag" +
str(iteration) + ".jpg"
for i in range(0 , iteration):
    print(i)
    blur_img = apply_box_filter(pad_img
, size)
    pad_img = np.zeros((w+size-
1,l+size-1) , np.uint8)
    pad_img[hs:w+hs,hs:l+hs] = blur_img

pad_img[hs:w+hs,hs:l+hs] = img
cv.imwrite(path , blur_img)
cv.waitKey(0)

```

:٣.٢.٢ قسمت

```

def save_result(filter, size , variance
, img , n_img):
    # path = "result/3.2.2/img_" +
filter + "_var" + str(variance) +
"_size" + str(size) + ".jpg"
    mse = np.square(n_img - img).mean()
    result.set_value(str(variance),
filter , mse)
    # cv.imwrite(path , noised_img)

path = "image/Lena.bmp"
img = cv.imread(path ,
cv.IMREAD_GRAYSCALE)
noise_var = np.array([0.01 , 0.05 ,
0.1])
window_sizes = np.array([3,5,7,9])

result = pd.DataFrame(index=['0.01' ,
'0.05' , '0.1'],columns=['nf' , 'median
3' , 'median 5' , 'median 7' , 'median
9' ,
'box 3' , 'box 5' , 'box 7' , 'box
9'])

for noise in noise_var:

    noised_img = random_noise(img ,
mode="gaussian" , var = noise)
    noised_img = np.array(noised_img *
255 , np.uint8)
    save_result('nf' , 0 , noise , img
, noised_img)
    for ws in window_sizes:
        box_img =
apply_box_filter(noised_img, ws)
        save_result('box ' + str(ws) ,
ws , noise ,img , box_img)
        median_img =
apply_median_filter(noised_img , ws)
        save_result('median ' + str(ws),
, ws , noise ,img , median_img)
print()
print(result)
result.to_csv("result.csv")

```

```

noise_img = np.array(noise_img *
255 , np.uint8)
save_result(0 , noise , img,
noise_img)
for win in window_sizes:
    n_img =
apply_median_filter(noise_img,win)
    save_result(win , noise , img ,
n_img)

result.to_csv("result1.csv")
print(result)

```

قسمت :۳.۲.۲

```

import cv2 as cv
import numpy as np
import pandas as pd
from skimage.util import *

def
apply_median_filter(img,window_size):
    w,l = img.shape
    result = np.zeros((w ,l) ,
np.uint8)
    hw = int(window_size / 2)

    for i in range(hw , w - hw):
        for j in range( hw,l - hw):
            result[i - hw][j - hw] =
np.median(np.squeeze(np.asarray(img[i-
hw:i+hw+1,j-hw:j+hw+1])))
    return result

def apply_box_filter(img,size):
    hs = int(size / 2)
    w,l = img.shape
    blur_img = np.zeros((w,l) ,
np.uint8)
    for i in range(hs,w-hs):
        for j in range(hs,l-hs):
            blur_img[i - hs][j - hs] =
np.uint8(np.sum(img[i-hs:i+hs+1,j-
hs:j+hs+1]) / (size*size))
    return blur_img

```

قسمت :۳.۳.۱

```
path = "image/room6.jpg"

image = cv.imread(path , cv.IMREAD_GRAYSCALE)
img = eqlized.equalize(image)
cv.imwrite("result/3.3.1/eqlized.jpg" , img)
img = apply_median_filter(img,5)

#apply laplacian
filters = np.array([[0,1,0],[1,-4,1],[0,1,1]] , np.uint8)
laplacian = apply_filter(img , filters)
cv.imwrite("result/3.3.1/laplacian.jpg" , laplacian)

# Apply Unsharp masking
result = np.array(laplacian + img , np.uint8)
cv.imwrite("result/3.3.1/result.jpg" , result)
# cv.imshow("gig" , gauss)
# cv.waitKey(0)
```

قسمت :۳.۴.۱

```
import cv2 as cv
import numpy as np

def dots(m1 , m2):
    if(m1.shape != m2.shape):
        print("WRONGE")
    w , l = m1.shape
    ans = 0
    for i in range(w):
        for j in range(l):
            ans += int(m2[i][j] *
m1[i][j])
    return ans

def apply_filter(img , ifilter,
weight):
    w,l = img.shape
    wf, lf = ifilter.shape
    hwf , hlf = int(wf/2) , int(lf/2)
    result = np.zeros((w - wf + 1 , l -
lf + 1) , int)
    maxi , mini = 0 , 255
    for i in range(hwf , w - hwf):
        for j in range(hlf , l - hlf):
            val = int(dots(ifilter ,
img[i - hwf: i + hwf + 1 , j - hlf : j +
hlf + 1]) / weight)
            maxi , mini = max(maxi ,
val) , min(mini , val)
            result[i - hwf][j - hlf] = val
```

```
def
apply_median_filter(img,window_size):
    w,l = img.shape
    result = np.zeros((w ,l) ,
np.uint8)
    hw = int(window_size / 2)

    for i in range(hw , w - hw):
        for j in range( hw,l - hw):
            result[i - hw][j - hw] =
np.median(np.squeeze(np.asarray(img[i-
hw:i+hw+1,j-hw:j+hw+1])))
            return result
```

```

if(m1.shape != m2.shape):
    print("WRONG")
w , l = m1.shape
ans = 0
for i in range(w):
    for j in range(l):
        ans += int(m2[i][j] *
m1[i][j])
return ans

def apply_filter(img , ifilter,
weight):
    w,l = img.shape
    wf, lf = ifilter.shape
    hwf , hlf = int(wf/2) , int(lf/2)
    result = np.zeros((w - wf + 1 , l -
lf + 1) , int)
    maxi , mini = 0 , 255
    for i in range(hwf , w - hwf):
        for j in range(hlf , l - hlf):
            val = int(dots(ifilter ,
img[i - hwf: i + hwf , j - hlf : j +
hlf]) / weight)
            maxi , mini = max(maxi ,
val) , min(mini , val)
            result[i - hwf][j - hlf] = val

    alpha = (maxi - mini) / 255
    scale = lambda x : (x - mini) /
alpha
    applied = np.array(scale(result) ,
np.uint8)
    return applied

filter_num = int(input())
img = cv.imread("image/Lena.bmp" ,
cv.IMREAD_GRAYSCALE)
filters = []
filters.append(np.array([[1, 0 , -1]] ,
int))
filters.append(np.array([[1 , 0 , -1] ,
[1 , 0 , -1] , [1 , 0 , -1]]) , int))
filters.append(np.array([[1 , 0 , -1] ,
[2 , 0 , -2] , [1 , 0 , -1]]) , int))
weight = [2 , 6, 8]

result = apply_filter(img ,
filters[filter_num] ,
weight[filter_num])

# cv.imshow("img" ,result)
# cv.imshow("sobel" ,sobelx)
# cv.waitKey(0)

path = "result/3.4.1/img" +
str(filter_num) + ".jpg"
cv.imwrite(path , result)

```

```

for i in range(hwf , w - hwf):
    for j in range(hlf , l - hlf):
        val = int(dots(ifilter ,
img[i - hwf: i + hwf + 1 , j - hlf : j +
hlf + 1]) / weight)
        maxi , mini = max(maxi ,
val) , min(mini , val)
        result[i - hwf][j - hlf] = val

    alpha = (maxi - mini) / 255
    scale = lambda x : (x - mini) /
alpha
    applied = np.array(scale(result) ,
np.uint8)
    return applied

filter_num = int(input())
img = cv.imread("image/Lena.bmp" ,
cv.IMREAD_GRAYSCALE)
filters = []
filters.append(np.array([[1, 0 , -1]] ,
int))
filters.append(np.array([[1 , 0 , -1] ,
[1 , 0 , -1] , [1 , 0 , -1]]) , int))
filters.append(np.array([[1 , 0 , -1] ,
[2 , 0 , -2] , [1 , 0 , -1]]) , int))
weight = [2 , 6, 8]

result = apply_filter(img ,
filters[filter_num] ,
weight[filter_num])

# cv.imshow("img" ,result)
# cv.imshow("sobel" ,sobelx)
# cv.waitKey(0)

path = "result/3.4.1/img" +
str(filter_num) + ".jpg"
cv.imwrite(path , result)

```

قسمت :۳.۴.۲

```

import cv2 as cv
import numpy as np

def dots(m1 , m2):

```

```
path = "result/3.4.2/img" +
str(filter_num) + ".jpg"
cv.imwrite(path , result)
```

:٣.٥.١ قسمت

```
import cv2 as cv
import numpy as np

def scale(img):
    maxi , mini = np.max(img) ,
    np.min(img)
    a = (maxi - mini) / 255
    scl = lambda x : (x - mini) / a
    result = np.array(scl(img) ,
    np.uint8)
    return result

filter_size = int(input("Enter input
size: "))
alpha = float(input("Enter alpah: "))
img = cv.imread("image/Lena.bmp" ,
cv.IMREAD_GRAYSCALE)
blur_img =
np.array(scale(cv.GaussianBlur(img,(fil
ter_size,filter_size),0)) , np.uint8)
result = np.array(img + (alpha *
scale(blur_img - img)) , np.uint8)
neg = np.array(blur_img - img ,
np.uint8)
path = "result/3.5.1/" +
str(filter_size) + "_" + str(alpha) +
".jpg"
cv.imwrite(path , result)
path = "result/3.5.1/blue" +
str(filter_size) + "_" + str(alpha) +
".jpg"
cv.imwrite(path , neg)
cv.waitKey(0)
```