Filter

فهیم جعفری

اطلاعات گزارش

تاریخ:

واژگان کلیدی:
Box filter
Median filter
Mmse
Gain factor
Edge detection
Robert filter
Sobel filter
Unsharp masking

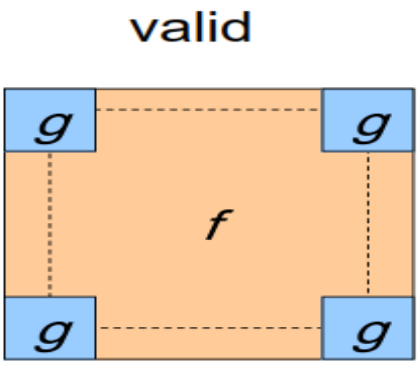
چکیده

واژه مکانی به خود تصویر اطلاق میشود و روش های پردازش تصویر در این حوزه،مبتنی بر دستکاری مستقیم پیکسل ها در تصویر است.این موضوع عکس پردازش تصویر در حوزه تبدیل است، دو دسته اصلی از پردازش مکانی ،تبدیلات شدت و فیلتر کردن مکانی است. فیلتر کردن مکانی به نام های نقاب های مکانی ،هسته ها،قالب ها و پنجره ها نیز خوانده می شود.تناظر یک به یک بین فیلترهای مکانی خطی و فیلترها در حوزه فرکانس وجود دارد اما فیلترهای مکانی توع زیادی دارند ،زیرا همان طور که خواهید دید ،انها میتوانند برای فیلتر کردن غیرخطی به کار روند ، چیزی که نمی توان در حوزه فرکانس انجام داد.فیلتر کردن ،پیکسل جدیدی با مختصاتی مساوی با مختصات مرکز همسایگی ایجاد میکند و مقدارش برابر با نتیجه عملیات فیلتر کردن است.اگر عملیات اجرا شده روی پیکسل های تصویر خطی باشد ،انگاه فیلتر مکانی خطی می نامند. وگرنه ،فیلتر غیر خطی است

1-مقدمه

2-شرح تکنیکال

در همه ی فیلتر ها برای حاشیه تصویر از تکنیک valid استفاده شده است که به صورت کل پایین میباشد که حاشیه ها ازتصویر ورودی به تصویر خروجی بدون تغییر انتقال یافته است



- **Box filter**

در این نوع فیلترها پنجره ای مربعی که هر کدام از عضو ها ضریب یکسان دارند در نظر گرفته میشود و بر روی قسمتی از تصویر ورودی قرار میگیرد و در ان ضرب میشود سپس خروجی ان

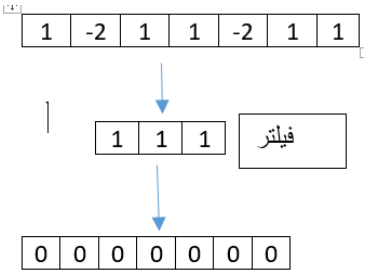
بر مجموع ضرایب تقسیم میشود و به جای عنصر مرکزی پنجره در تصویر خروجی قرار میگیرد و پنجره به پیکسل مجاور انتقال پیدا میکند . به این فیلترها،فیلترهای میانگین نیز میگویند چرا که در واقع میانگین شدت پیکسل های مجاور و مرکز در پیکسل مرکزی قرار میگیرد.

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

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

لیست بدی های این فیلتر:

1. لبه های تصویر در این نوع فیلتر از بین میرود
2. ابجکت هایی که شدت نزدیک به شدت پس زمینه دارند با پس زمینه مخلوط میشوند
3. اگر از zero-padding استفاده شود سیاهی درحاشیه تصویر تاثیر داده میشود و تصویر در ان ناحیه تاریک میشود
4. در صورت بزرگ انتخاب کردن پنجره ابجکت های کوچک چه با واریانس بالا و چه کم از بین میروند
5. ساختار کلی تصویر را از بین میبرد



6. هیستوگرام تصویر عوض میشود

اگر این فیلتر چندین بار بر روی تصویر زده شود:

لبه ها بیشتر محو میشوند پس این ویژگی بد تقویت میشود

ابجکتهای با شدت نزدیک به پس زمینه کاملا محو میشوند پس این ویژگی بد نیز تقویت میشود

صفرهای حاشیه برای zero-padding در ناحیه های بیشتری دخالت داده میشود

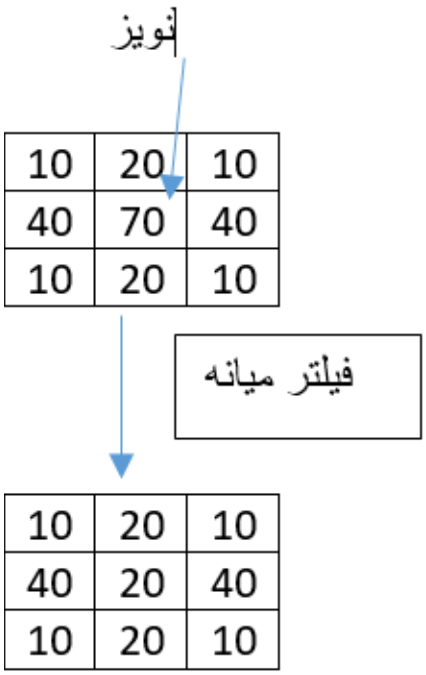
تصویر بیشتر مات میشود

ابجکت های کوچک محو میشوند پس این ویژگی های بد نیز تقویت میشوند

و تصویر به سمت یک تصویر یکنواخت متمایل میشود که شدت برابر با میانگین تصویر دارد

- **Median filter**

این فیلتر مقدار پیکسل را با میانه مقادیر شدت در همسایگی ان پیکسل جایگزین میکند. این نوع فیلتر ها قابلیت کاهش نویز هوشمندی را فراهم میکند که نسبت به فیلترهای هموارسازی خطی با اندازه مشابه مات کردن انها کمتر است.برای اجرای فیلتر کردن میانه در نقطه ای از تصویر ابتدا مقادیر پیکسل ها موجود در همسایگی را مرتب میکنیم ،میانه انها را تعیین میکنیم وسپس ان مقدار ان را به پیکسل متناظر در تصویر فیلترشده نسبت میدهیم مثال برای فیلتر 3*3 به شرح زیر میباشد:



کار اصلی فیلتر میانه این است که نقاطی که با سطوح شدت مجزا خیلی شبیه به همسایه های خود شوند

برای مقایسه فیلترهای جعبه ای و فیلترهای میانه میتوان از mmse بین تصویر فیلترشده و تصویر اصلی استفاده کرد و با هم مقایسه کرد که کدام mmse کمتری دارد که در بخش شرح نتایج اورده شده است.

- **Sharpening,bluring,and nois removal**

تکنیک هایی که قبل از این و بعد از این توضیح داده شده است استفاده شده است غیر از فیلتر لاپلاسین که مشتق دوم میباشد و پیاده سازی پنجره ان به شکل زیر است

$\frac{1}{16}$

-1	-1	-1
-1	8	-1
-1	-1	-1

که ضریب ان gain factor میباشد که توضیح gain factor در قسمت بعدی (edge detection) امده است

- **Edge Detection**

مشتق های اول در پردازش تصویر ، با استفاده از بزرگی گرادیان پیاده سازی میشوند برای تابع f(x,y) ،گرادیان در مختصات(x,y) به صورت بردار ستونی دو بعدی تعریف میشود:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

تعریف پایه مشتق مرتبه اول مربوط به تابع دو بعدی f(x,y) ، تفاضل زیر است

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$
$$\frac{\partial f}{\partial y} = f(y+1) - f(y)$$

که برای تابع یک بعدی فقط

$$\frac{df}{dx} = f(x+1) - f(x)$$

مطرح میباشد این مشتق ها (1) باید در نواحی با شدت ثابت ،صفر باشد(2)در اغاز سراشیبی نباید صفر باشد (3) در طول سراشیبی باید غیر صفر باشد که میتوان در راستای x با پنجره های زیر انها را پیاده سازی کرد

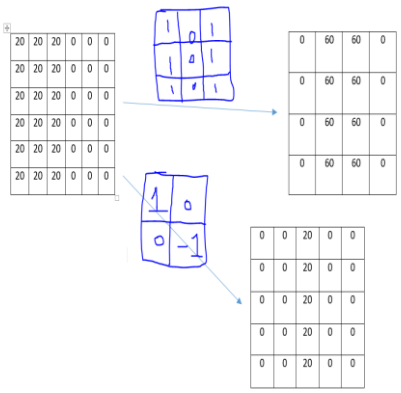
$$a)\frac{1}{2}[1 \quad 0 \quad -1], \quad b)\frac{1}{6}\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix}, \quad c)\frac{1}{8}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

که ضرایب پنجره ها gain factor پنجره میباشد تا بتوان مقاومت پنجره نسبت به نویز را کنترل کرد و پنجره اخر که در مکان (2,1) و (2,2) عدد 2 و -2 دارد برای این میباشد که در این فیلتر به عناصر نزدیک به مرکز توجه بیشتری دارد تا عناصر قطری به این پنجرها ،پنجره های روبرتز میگویند

دو پیاده سازی دیگر هم برای مشتق اول وجود دارد که به پنجره های سوبل معروف میباشد ان ها به شکل زیر میباشد که در راستای قطری مشتق میگیرند

d) $\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$, e) $\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$

این پیاده سازی نسبت به پیاده سازی قبل لبه های نازک تری تولید میکنند در پیاده سازی قبلی به ازای هر لبه دو پیکسل میگزارد ولی در این پیاده سازی فقط یک پیکسل حایگزاری میشود



اما نکته اخر اینکه گرادیان برای بعضی از لبه ها منفی میشود و هنگامی که ماشین ان را خروجی میدهد مقادیر کوچکتر از صفر را صفر در نظر میگیرد پس یک قدر مطلق هنگام پنجره زدن لازم است تا خروجی مثبت شود ولبه از بین نرود

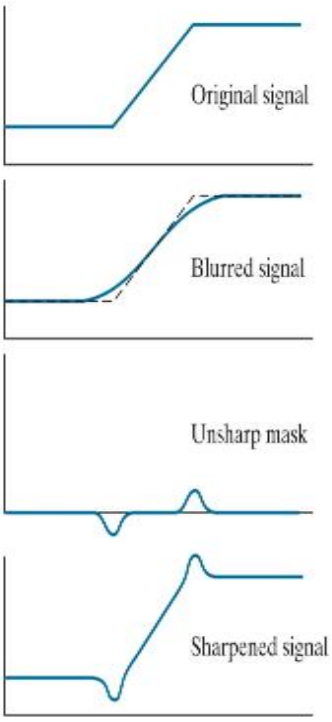
- **Unsharp masking**

این فیلتر که برای تیز کردن تصویر میباشد مراحل زیر انجام میشود

1. تصویر اصلی با فیلتر مات میشود
2. تصویر مات شده از تصویر اصلی تفریق میشود و حاصل در نقاب ریخته میشود
3. نقاب بدست امده از مرحله قبل به تصویر اصلی اضافه میشود

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

تصویر بالا معادله این فیلتر را نشان میدهد که در ان I' تصویر مات شده میباشد و I تصویر اصلی و α ضریبی میباشد که در نقاب ضرب میشود و سپس به نصویر اصلی اضاف میشود و هر چه این ضریب بیشتر باشد تصویر تیزتر میشود که توجیه ان به شکل زیر است



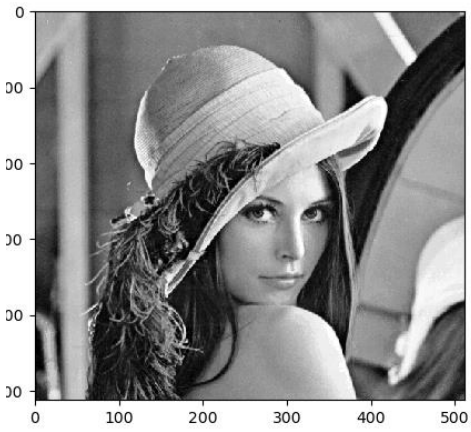
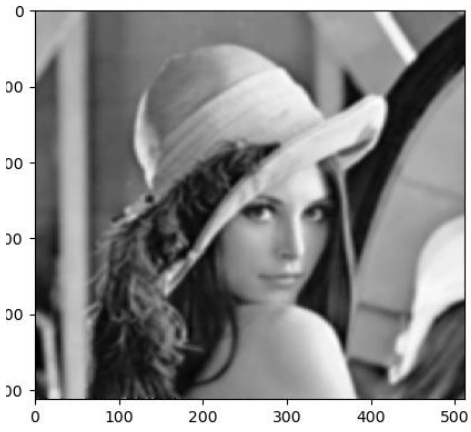
اگر ضریب الفا از یک حدی بزرگ باشد نتیجه نهایی شدت ها ممکن است منفی یا بزرگتر از 255 باشد تصویر خروجی خراب میشود

3-شرح نتایج

- **Box filter**

تصویر زیر تصویر اصلی و نتیجه فیلتر 3در3 جعبه ای میباشد همانطور که در تصویرخروجی مشخص است لبه ها کمی از حالت تیزی در امده اند و یک لکه سفید که در تصویر اصلی مشاهده میشود در تصویر خروجی کوچکتر شده است

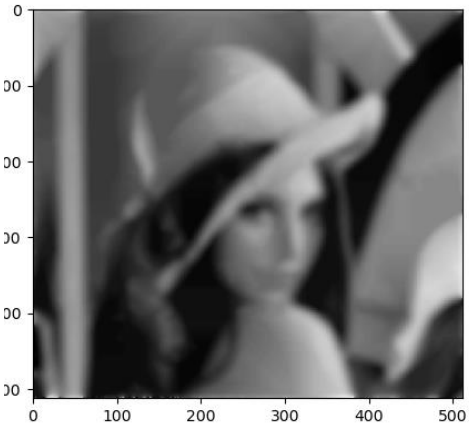

تصویر زیر تصویر اصلی و نتیجه فیلتر 3در3 جعبه ای با 8 بار تکرار میباشد همانطور که مشاهده میشود این تصویر به صورت بلوری در امده است و لکه ی سفیدی که در تصویر میباشد بسیار کوچکتر شده است و با پس زمینه مخلوط شده است پس تا اینجای کار میتوان نتیجه گرفت که از فیلترهای جعبه ای میتوان به عنوان حذف نویز استفاده کرد اما تصویر خروجی بلوری خواهد بود و نتیجه خوبی نمیدهد و همینطور رنگ پس زمینه عوض شده است

اما تصویر زیر که پس از 64 بار تکرار فیلتر جعبه ای 3در3 را نشان میدهد مشاهده میشود که علاوه بر موارد قبلی اجسام کوچک در تصویر با پس زمینه محو شده اند و دیده نمیشوند مانند تار موی خانم لنا که بر روی شانه ی او میباشد و اختلاف شدت زیادی با پس زمینه ان دارد ، همینطور شدت پیکسل های اجسام تا 64 پیکسل در راستای حرکت پنجره بر روی همسایه هایش به طور نزولی تاثیر گزاشته است و فقط اجسام بزرگ با جزییات پایین قابل مشاهده میباشند. پس اگر فیلتر جعبه چندین بار اجرا شود:

1. لبه های تصویر در این نوع فیلتر از بین میرود
2. ابجکت هایی که شدت نزدیک به شدت پس زمینه دارند با پس زمینه مخلوط میشوند

3. اگر از zero-padding استفاده شود سیاهی درحاشیه تصویر تاثیر داده میشود و تصویر در ان ناحیه تاریک میشود
4. در صورت بزرگ انتخاب کردن پنجره ابجکت های کوچک چه با واریانس بالا و چه کم از بین میروند
5. ساختار کلی تصویر را از بین میبرد



- **Median filter**

تصویر زیر مربوط به جدول msse میباشد که بین نتیجه فیلتر اعمال شده بر روی تصویر نویزی فلل و نمک و تصویر بدون نویز گرفته شده است

	3 X 3	5 x 5	7 x 7	9 x 9
P = 0.05	16.983001708984375	24.297760009765625	28.707027435302734	32.197990417480 47
P = 0.1	18.472309112548828	25.228126525878906	29.447780609130 86	32.834480285644 53
P = 0.2	21.536758422851562	26.846218109130 86	30.632778167724 61	33.771320343017 58

تصویر اصلی

تصویر مربوط به density=0.05

تصویر زیر مربوط به density=0.05 و فیلتر 3در3 میباشد

تصویر مربوط به density=0.2

تصویر مربوط به density = 0.1

همانطور که مشاهده میشود باز هم فیلتر 3در3 تنوانست تمامی نویزهای تصویر را حذف کند و همانطور که مشاهده میشود تمام نویز موجود در تصویر کاملا حذف شده است پس نیازی به انتخاب سایز بزرگتر فیلتر نمیباشد و بهترین سایز برای این چگالی نویز 3در 3 میباشد چرا که اگر در تصویر خروجی دقت شود و با تصویر اصلی مقایسه شود تصویر کمی مات شده است و اگر سایز بزرگتر انتخاب کنیم این مات شدگی همانطور که در قسمت های بعد خواهید دید بیشتر میشود و همینطور اگر به جدول نگاه کنید مقدار mmse با سایز بزرگتر بیشتر میشود اگر مات شدگی در تصویر را نمیتوانید تشخیص دهید نگران نباشید در قسمت های بعد به صورت واضحتر ان را مشاهده خواهید کرد

تصویر زیر تصویر فیلتر شده 3در3 با چگالی نویز 0.1 میباشد

باز هم بهترین سایز برای چگالی 0.1 ،3در3 میباشد پس بریم سراغ نویز بیشتر بر روی تصویر.

تصویر زیر فیلتر 3در3 باچگالی نویز 0.2 میباشد

در این تصویر مشاهده میشود که مقداری نویز در تصویر باقی مانده است برای همین باید سایز بزگتر انتخاب کرد تا مقدار نویز در پنجره انتخاب شده کمتر باشد و هنگام انتخاب میانه در پنجره نویز انتخاب نشود.

تصویر زیر نتیجه ی فیلتر 5در5 بر روی تصویر نویزی با چگالی 0.2 میباشد

همانطور که مشاهده میشود این تصویر دارای نویز سیاه وسفیدی که ایجاد شده بود نمیباشد ولی تصویر مات تر(بلوری) شده است و همین طور در جدول مقدار mmse ان نسبت به حالت 3در3 کاهش یافته است ولی خوب از لحاظ نویزی بهتر است که این نتیجه را به عنوان بهترین خروجی در نظر بگیریم که دارای نویز نمیباشد و اگر به لبه ها در تصویر دقت کنید لبه های نویزی و کاذبی به وجود امده است و این هم یکی از بدی های فیلتر میانه میباشد که باید در نظر گرفت

برای مشاهد واضحتر این که با افزایش سایز پنجره تصویر بلورتر میشود و mmse نیز بیشتر میشود تصویر زیر که با پنجره 7در7 گرفته شده و جدول مربوط به گزارش mmse را مشاهده کنید البته سرعت افزایش mmse با توجه افزایش سایز پنجره به مرور کم تر میشود این یکی از نتایجی است که میتوان از جدول گرفت

حاشیه تصویر فقط در خروجی کپی شده است و هیچ عملیاتی بر روی ان انجام نشده است

و اما بریم سراغ نویز گوسی و مقایسه دو فیلتر جعبه ای و میانه

جدول mmse فیلتر جعبه ای بر روی تصویر نویز گوسی

تصویر نویزی با واریانس 0.01

تصویر نویزی با واریانس0.05

تصویر نویزی با واریانس 0.1

این تصاویر از ان جهت گزاشته شده اند که در ادامه به انها برای مقایسه مرجع داده میشود

جدول mmse فیلتر میانه بر روی تصویر نویز گوسی

	3 X 3	5 x 5	7 x 7	9 x 9
sigma = 0.01	224.4844111218864	343.12453083496086	494.03862781040006	648.9497095360077
sigma = 0.05	475.55967349770634	445.49102589721673	555.8575680536909	694.0297684380965
sigma = 0.1	738.2986191172658	572.2367412780762	647.3248713047292	771.7153109850053

	3 X 3	5 x 5	7 x 7	9 x 9
sigma = 0.01	67.6969807861328	53.486820220947266	48.473529815673830	48.031459808349610
sigma = 0.05	92.6385765075683 6	80.808452606201170	72.644207000732420	67.436363220214840
sigma = 0.1	99.7595405578613 3	89.889110566185550	82.009418487548830	76.852310180664060

هر چه سایز فیلتر میانه تا یک اندازه ی خاص بر روی تصویر نویزی گوسی بزرگ تر باشد میزان mmse ان کمتر میشود و همینطور نویز های تصویر کمتر میشود و تصویر بلوری میشود البته بعد از یک سایز هر چه بزرگتر شود میزان mmse بیشتر میشود که در جدول اورده نشده است ولی دلیل ان اینست که اطلاعات اماری در پنجره با سایز بزرگ، معتبر و مناسب برای جایگزینی نیست که برای sigma=.01وsigma=.05وsigma=.1 به ترتیب سایز های 7*7 و 9*9 و9*9 مناسب میباشد علت انیکه برای پنجره 7*7 برای sigma=.01 انتخاب شد اینست مقدار mmse تفاوت خاصی با پنجره 9*9 ندارد و اگر پنجره بزرگتر انتخاب شود تصویر بلوری میشود

برای فیلتر جعبه ای هم همینطور با افزایش سایز پنجره ،مقدار mmse تا یک اندازه رابطه معکوس و سپس رابطه مستقیم دارد که این اندازه خاص بستگی به میزان نویزگوسی در تصویر دارد که برای sigma=.01 و sigma-.05 وsigma=.1 به ترتیب سایز های 3*3 و 5*5 و 5*5 مناسب میباشد که علت انتخاب این سایزها برای اینست که سایز های بزرگتر و کوچکتر از انها یا میزان نویز انها بیشتر بوده یا میزان mmse و بلوری انها بیشتر است

همانطور که در جدول ها مشاهده میشود میزان mmse فیلتر های جعبه ای بیشتراست از mmse فیلتر های میانه در تصاویر زیر هم که میبینیم که خروجی های فیلتر جعبه ای بلوری تر از فیلتر های میانه میباشد پس فیلتر های میانه قوی تر از فیلتر های جعبه ای در حذف نویز میباشند به خصوص در حذف نویز فلل و نمک که در قسمت های قبلی مشاهده کردیم

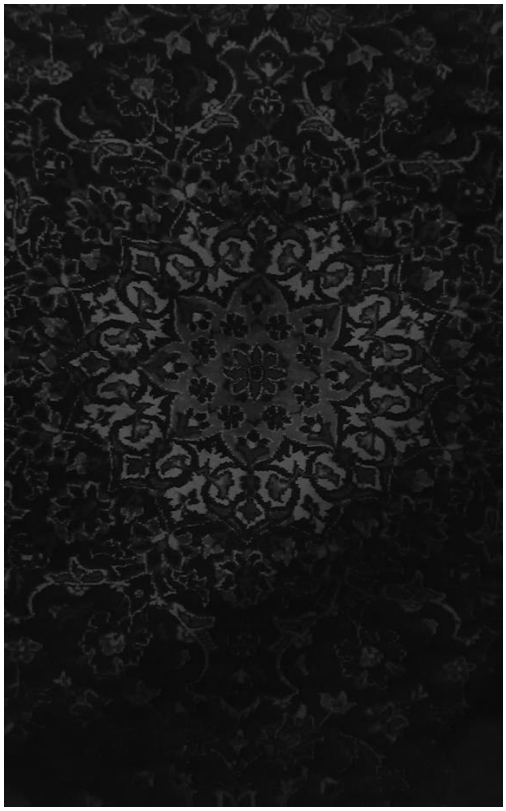
خروجی فیلتر جعبه ای سایز 9*9 و نویز با واریانس 0.1

خروجی فیلتر میانه سایز 9*9 و نویز با واریانس 0.1

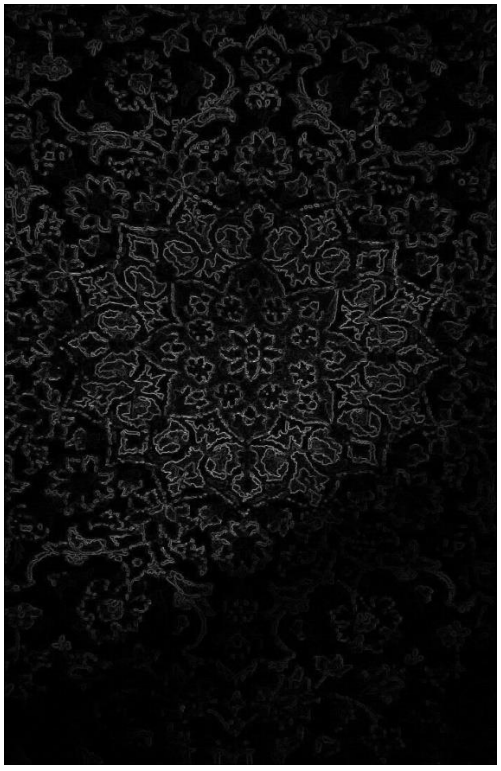
لبه های کاذبی در این تصویر ایجاد شده است

- **Sharpening,noise removal, bluring**

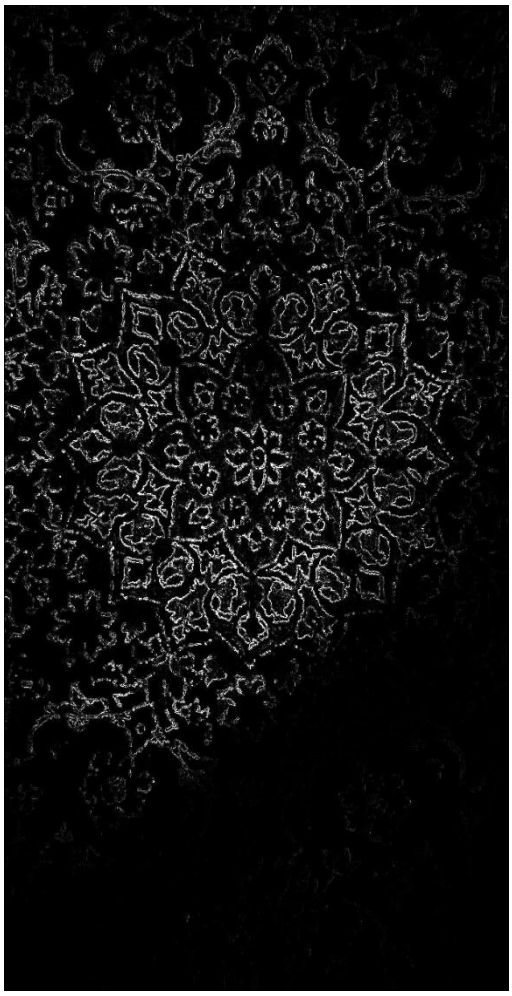
تصویر زیر تصویری از فرش میباشد که در نور کم گرفته شده است و دارای نویز و بلوری میباشد



برای اینکه تصویر را از حالت بلوری به حالت شارپ نزدیک کنیم از فیلتر های روبرتز استفاده کردیم که نقاب scale شده ان به شکل زیر میباشد



در شکل بالا تمام لبه ها دیده نمیشود برای اینکه لبه های بیشتر تشخیص داده شود یک لاپلاسین فیلتر بر روی تصویر زده میشود و با نقاب قبلی که حاصل فیلتر روبرتز بود ضرب میشود که خروجی ان به شکل زیر است



اما در این تصویر نویز هم تقویت شده برای رفع نویز یک فیلتر میانه با سایز5*5 بر روی این نقاب زده میشود که خروجی ان به شکل زیر است

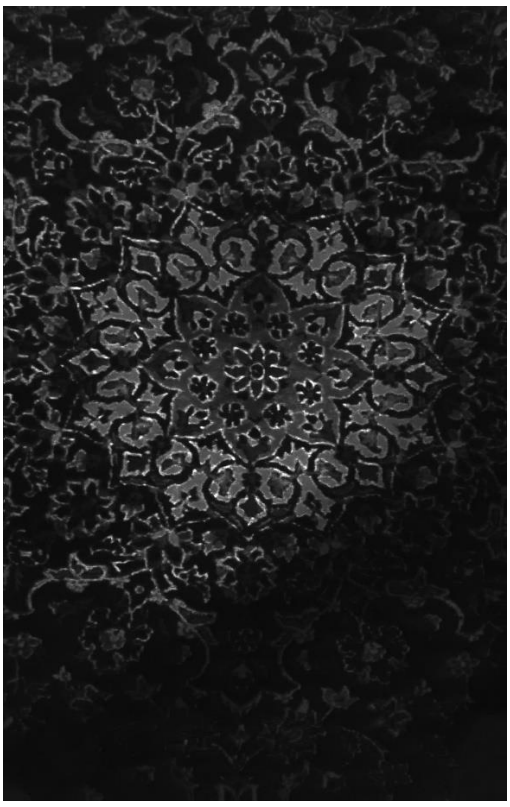
- **Edge detection**

فیلترهای گرادیان برای تشخیص لبه ها و تشخیص عیب ها و ارتقای انها میباشد.

تصویر زیر مربوط به نقاب گرادیان با پنجره a که در شکل ان در شرح تکنیکال امده است میباشد

اما همانطور که دیده میشود چیز زیادی مشخص نیست چرا که شدت های لبه کوچک است و بازه ای کوچک را از هیستوگرام تشکیل داده اند برای این مثال شدت ها بین بازه ی (0,89) میباشد برا اینکه بتوان لبه ها را مشاهده کرد شدت ها را بین بازه ی (0,255) گسترش میدهیم که خروجی زیر نتیجه ان میباشد

در تصویر بالا فقط لبه ها وجود دارد حال این نقاب را با تصویر اصلی جمع مکینیم تا لبه ها تقویت شده و و نویز ها در برابر لبه ها ضعیف میشوند که خروجی به شکل زیر در امده است



اما نویز هایی نیز در این نقاب به عنوان لبه تشخیص داده شده اند که به صورت خال های کوچک در تصویر دیده میشوند برای اینکه این مشکل را تا حدی حل کنیم به سراغ فیلتر b میرویم که در شرح تکنیکال پیاده سازی ان امده است در این فیلتر همسایگی های بیشتری از مرکز پنجره در نظر گرفته میشود و در نتیجه نسبت به نیز مقاوم تر است برای اینکه بتوانیم تصویر نقاب را خوب مشاهده کنیم نقاب را بین بازه (0,255) گسترش میدهیم که خروجی ان به شکل زیر میباشد

این تصویر دیگر نویزهای تصویر قبلی را ندارد ولی متاسفانه بعضی از لبه ها را نتوانسته به خوبی تشخیص دهد برای همین به سراغ فیلتر c میرویم که به عناصر نزدیک به مرکز توجه بیشتری دارد

این فیلتر نقایص a و b را ندارد ولی باید به این نکته توجه کرد که فیلتر a با وجود اینکه نویز ها را تشخیص میداد ولی از لحاظ محاسباتی سریع تر بود چرا که فقط 2 پیکسل مجاور را در نظر میگرفت

جدول زیر mmse اندازه گرفته شده بین یک نقاب که از تصویر نویزی بدست امده و نقابی که از تصویر بدون نویزی بدست امده گرفته شده است که نشان میدهد بیشترین مقاومت در برابر نویز را فیلتر bدارد

	window a	window b	window c
mmse	257.4482431411743	83.70010534922278	94.26447004079819

تصویر زیر مربوط به نقاب گرادیان با فیلتر d که پیاده سازی ان در شرح تکنیکال امده است میباشد

همانطور که در تصویر مشاهده میشود این فیلتر لبه هایی که تقریبا موازی با خطی با شیب منفی یک را نتواسته تشخیص بدهد و همینطور نویز هایی را به عنوان لبه تشخیص داده است برای انیکه بتوانیم شیب های منفی را تشخیص بدهیم از پنجره e که پیاده سازی ان در شرح تکنیکال امده است استفاده میکنیم و برای واضحی کار شدت های ان را بین بازه ی (0,255) بسط میدهیم

اما این تصویر لبه هایی که تقریبا موازی با خطی با شیب مثبت یک را تشخیص نداده است و همچنان نویز ها هم باقی است

برای اینکه بتوان هم لبه های با شیب منفی و مثبت 1 را داشت میتوان در نقاب را با فاصله اقلیدوسی با هم جمع کرد که این کار بار محاسباتی زیادی دارد اگر دوست داشتید بار محاسباتی ان کم شود میتوانید از جمع انها استفاده کنید. نتیجه scale شده فاصله اقلیدوسی دو فیلتر به شکل زیر میباشد

همانطور که مشاهده میشود نویز های موجود در این نقاب ها خیلی بیشتر است از نقابهای فیلتر سوبل و ذکر این نکته هم باقی نماند که در فیلتر های سوبل ما از یک ضریب به اسم gain factor استفاده کردیم که موجب کاهش نویز میشود ولی خوبی که این فیلتر ها نسبت به فیلتر های قبلی(سویل) دارد این است که لبه ها در این فیلترها نازک تر میباشد و دقت تصویر را بالا میبرد

شکل زیر ادامه جدول قبلی میباشد که بر روی این فیلترهای روبرت انجام شده است

window a	window b	window A&B
1052.1536827087402	1063.1350440979004	1920.2616626797976

بله همانطور که انتظار میرفت اعداد وحشتناک میباشد و مقاومت این فیلتر ها نسبت به نویز از فیلتر های سوبل کمتر است ولی بار محاسباتی انها به علت کوچکتر بودن پنجره کمتر میباشد

در زیر نتیجه چند خروجی که نقاب ها با خود عکس جمع شد اند را میبیند

تصویر با فیلتر c

تصویر با فیلتر d

مشاهده میکنید که نویز در ان تقویت شده و در جهت

با شیب منفی 1 تقویت نشده

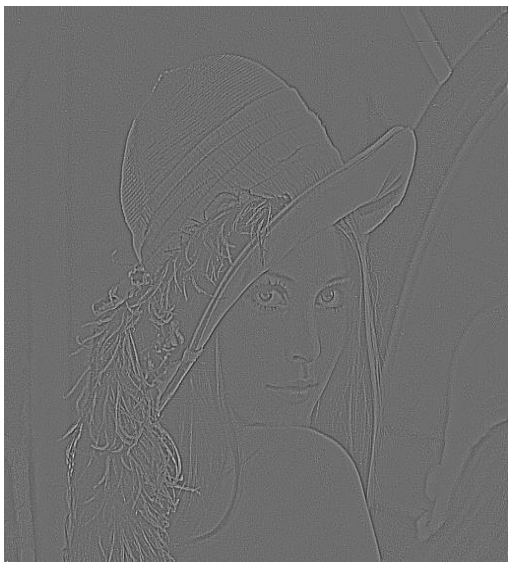
تصویر بافیلتر d,e

نویز بیش از همه در این تصویر تقویت شده گویا که نویز گوسی به تصویر اعمال شده است و لبه ها در همه جهت ها تقویت شده است

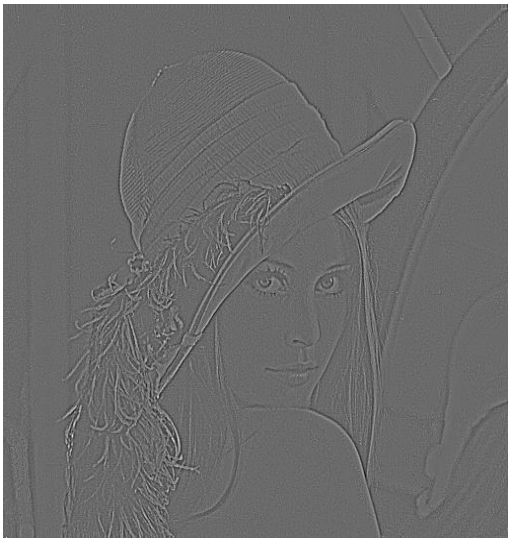
- **Unsharp masking**

تصاویر زیر مربوط به نقاب های unsharp mask میباشد یعنی $I - I'$ با سایزهای مختلف فیلتر گوسی میباشد که برای تحلیل بر روی اندازه سایز فیلتر گوسی اورده شده اند

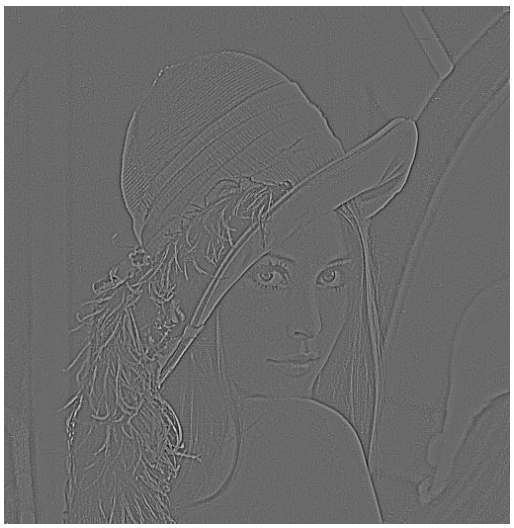
تصویر با سایز فیلتر گوسی 3*3



تصویر با سایز فیلتر گوسی 5*5

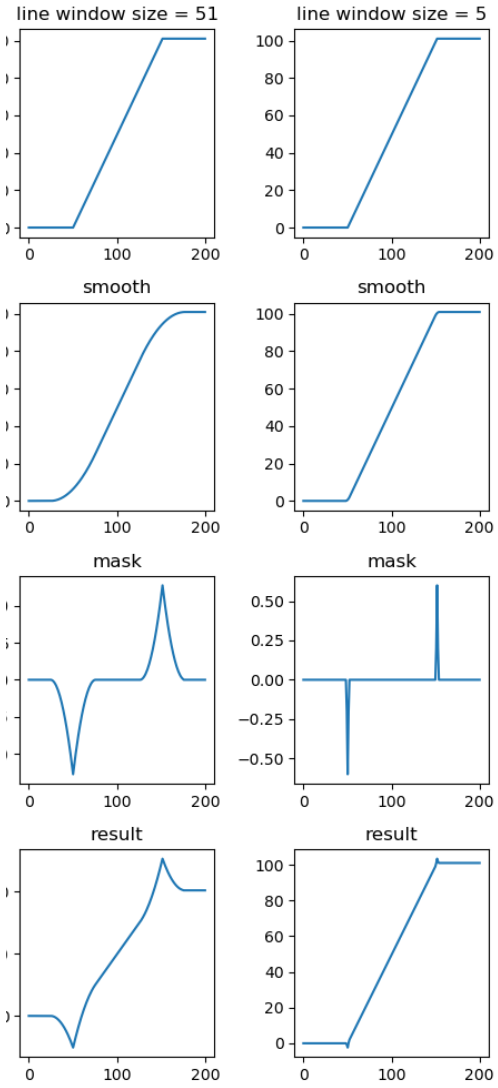


تصویر با سایز فیلتر گوسی 7*7



تصویر با سایز فیلتر گوسی 9*9

اگر به لبه های تصویر دقت شود مشاهده میکنید که با افزایش سایز پنجره فیلتر گوسی ضخامت لبه ها بیشتر میشود و میزان شدت انها بیشتر میشود و همینطور لبه های نزدیک به هم با هم یکی مشوند برای اینکه این امر را ثابت کنیم یک ازمایش یک بعدی که به واضحی ان را نشان میدهد انجام شده است که در زیر اورده شده است



همانطور که میبینید با افزایش سایز فیلتر گوسی لبه ها به هم نزدیک و با هم میکس میشوند و همینطور میزان تقویت لبه ها بیشتر میشوند ولی اگر نویزی وجود داشته باشد تاثیر ان کمتر میشود در واقع با افزایش سایز فیلتر گوسی رابطه عکس دارد چرا که همسایه های بیشتری در فیلتر تاثیر داده میشوند و تاثیر نویز کمتر میشود

بد نیست به نتایج زیر هم یک نگاه بیاندازید که مشاهده میشود با افزایش alpha میزان نویز و لبه ها با هم بیشتر در تصویر تقویت میشوند

تصویر unsharp شده با سایز 3*3 و alpha=.1 و sigma = 1

تصویر unsharp شده با سایز 3*3 و alpha=1 و sigma=1

4-کد برنامه

کد مربوط به فیلتر جعبه ای و اعمال ان بر روی تصویر لنا

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import copy

def box_filter(img,window,sumWndwW8):
    res = copy.deepcopy(img)
    #indicate area of filter
    paddingW2cntri = int(window.shape[0]/2)
    paddingW2cntrj = int(window.shape[1]/2)
    for i in range(paddingW2cntri,img.shape[0]-paddingW2cntri):
        for j in range(paddingW2cntrj,img.shape[1]-paddingW2cntrj):
            # convolve filter with image
            res[i,j] = (1/sumWndwW8)*sum(np.asarray(img[i-paddingW2cntri:i+paddingW2cntri+1,j-paddingW2cntrj:j+paddingW2cntrj+1] * window).flatten())
    return res


window = np.ones((3,3))
sumWndwW8 = 9

img = cv2.imread('../Lena.bmp')
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
res = copy.deepcopy(img)
num = 8
for i in range(0,num):
    res = box_filter(res,window,sumWndwW8)

fig,axs = plt.subplots(2,figsize=(5,10))
axs[0].imshow(img,cmap='gray',aspect='auto')
axs[1].imshow(res,cmap='gray',aspect='auto')
fig.savefig('../images/hw3.1/res{}.png'.format(num))
```

کد مربوط به فیلتر میانه

```
ax.table(cellText=data,
colLabels=col_labels, loc='center')
fig.savefig('../images/hw3.2/resTable.pn
g')
```

کد مربوط به به مقایسه ی فیلتر میانه و فیلتر جعبه ای

```
import numpy as np
import cv2
import skimage
import copy
import matplotlib.pyplot as plt

def median_filter(img,sizeWindow):
    res = copy.deepcopy(img)
    paddingW2cntri =
int(sizeWindow[0]/2)
    paddingW2cntrj =
int(sizeWindow[1]/2)
    for i in
range(paddingW2cntri,img.shape[0]-
paddingW2cntri):
        for j in
range(paddingW2cntrj,img.shape[1]-
paddingW2cntrj):
            window =
np.sort(np.asarray(img[i-
paddingW2cntri:i+paddingW2cntri+1,j-
paddingW2cntrj:j+paddingW2cntrj+1]).flat
ten())
            res[i,j] =
window[int(sizeWindow[0]*sizeWindow[1]/2
)]
    return res


img = cv2.imread('../Lena.bmp')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
densities = [0.05,0.1,0.2]
sizeWndws = [(3,3),(5,5),(7,7),(9,9)]
mmse_matric = {}
for density in densities:
    noisy_img =
skimage.util.random_noise(img,
mode='s&p', seed=None,
clip=True,amount=density)
    noisy_img =
skimage.img_as_ubyte(noisy_img)

cv2.imwrite('../images/hw3.2/imagesaltde
nsity={}.jpg'.format(density),noisy_img)
    temp_dict = {}
    for sizeW in sizeWndws:
        res =
median_filter(noisy_img,sizeW)

cv2.imwrite('../images/hw3.2/resd={}{}.j
pg'.format(density,sizeW),res)
        temp_dict[sizeW] =
np.square(np.subtract(img,res)).mean()
    mmse_matric[density] = temp_dict

data = []
counter = 0
for dict in mmse_matric.values():
    temp_list = []
    temp_list.append('P =
{}'.format(densities[counter]))
    for mmse in dict.values():
```

```
import numpy as np
import cv2
import skimage
import copy
import matplotlib.pyplot as plt

def median_filter(img,sizeWindow):
    res = copy.deepcopy(img)
    paddingW2cntri =
int(sizeWindow[0]/2)
    paddingW2cntrj =
int(sizeWindow[1]/2)
    for i in
range(paddingW2cntri,img.shape[0]-
paddingW2cntri):
        for j in
range(paddingW2cntrj,img.shape[1]-
paddingW2cntrj):
            window =
np.sort(np.asarray(img[i-
paddingW2cntri:i+paddingW2cntri+1,j-
paddingW2cntrj:j+paddingW2cntrj+1]).flat
ten())
            res[i,j] =
window[int(sizeWindow[0]*sizeWindow[1]/2
)]
    return res


img = cv2.imread('../Lena.bmp')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
densities = [0.05,0.1,0.2]
sizeWndws = [(3,3),(5,5),(7,7),(9,9)]
mmse_matric = {}
for density in densities:
    noisy_img =
skimage.util.random_noise(img,
mode='s&p', seed=None,
clip=True,amount=density)
    noisy_img =
skimage.img_as_ubyte(noisy_img)

cv2.imwrite('../images/hw3.2/imagesaltde
nsity={}.jpg'.format(density),noisy_img)
    temp_dict = {}
    for sizeW in sizeWndws:
        res =
median_filter(noisy_img,sizeW)

cv2.imwrite('../images/hw3.2/resd={}{}.j
pg'.format(density,sizeW),res)
        temp_dict[sizeW] =
np.square(np.subtract(img,res)).mean()
    mmse_matric[density] = temp_dict

data = []
counter = 0
for dict in mmse_matric.values():
    temp_list = []
    temp_list.append('P =
{}'.format(densities[counter]))
    for mmse in dict.values():
        temp_list.append(mmse)
    counter+=1
    data.append(temp_list)

col_labels = (" ", "3 X 3","5 x 5","7 x
7","9 x 9")
fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
ax.axis('off')
```

```
def gradient(img,num_window):
    res = np.zeros(img.shape)
    num_window = num_window
    if num_window == 'a':
        #window a
        window = np.ones((1,3))
        window[0,0] = 1
        window[0,1] = 0
        window[0,2] = -1
        sumw = 2
    elif num_window == 'b':
        # window b
        window = np.ones((3,3))
        window[0,0] = 1
        window[0,1] = 0
        window[0,2] = -1
        window[1,0] = 1
        window[1,1] = 0
        window[1,2] = -1
        window[2,0] = 1
        window[2,1] = 0
        window[2,2] = -1
        sumw = 6
    else:
        #window c
        window = np.ones((3,3))
        window[0,0] = 1
        window[0,1] = 0
        window[0,2] = -1
        window[1,0] = 2
        window[1,1] = 0
        window[1,2] = -2
        window[2,0] = 1
        window[2,1] = 0
        window[2,2] = -1
        sumw = 8
    paddingW2cntri =
int(window.shape[0]/2)
    paddingW2cntrj =
int(window.shape[1]/2)
    for i in
range(paddingW2cntri,img.shape[0]-
paddingW2cntri):
        for j in
range(paddingW2cntrj,img.shape[1]-
paddingW2cntrj):
            res[i,j] =
(1/sumw)*abs(sum(np.asarray(img[i-
paddingW2cntri:i+paddingW2cntri+1,j-
paddingW2cntrj:j+paddingW2cntrj+1] *
window).flatten()))
    return res


img = cv2.imread('../Lena.bmp')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
noisy_img =
skimage.util.random_noise(img,
mode='gaussian', seed=None,
clip=True,var=.01)
noisy_img =
skimage.img_as_ubyte(noisy_img)

num_windowa = 'a'
edgea = gradient(img,num_windowa)
noisy_edgea =
gradient(noisy_img,num_windowa)
mmsea =
np.square(np.subtract(edgea,noisy_edgea)
```

```
        temp_list.append(mmse)
    counter+=1
    data.append(temp_list)

col_labels = (" ", "3 X 3","5 x 5","7 x
7","9 x 9")
fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
ax.axis('off')
ax.table(cellText=data,
colLabels=col_labels, loc='center')
fig.savefig('../images/hw3.2/resTable.pn
g')
```

کد مربوط به بخش noise removal , bluring , sharpening

```
import numpy as np
import cv2
import skimage
import copy
import matplotlib.pyplot as plt

from src.hw3_2_1 import median_filter
from src.laplac import laplcian_filter
from src.hw3_4_2 import *


num = 5
img = cv2.imread('hg{}.jpg'.format(num))
img =
cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
img = img.astype(np.float)

edgeA = gradient_robert(img,'a')
edgeB = gradient_robert(img,'b')

edge = combine(edgeA,edgeB)
laplace_edge = laplcian_filter(img)

# edge = combine(laplace_edge,edge)
edge = laplace_edge * edge
edge = median_filter(edge,(5,5))

res = img + edge

# res =
normalizeimg(res,np.min(res),np.max(res)
)
cv2.imwrite('images/test/edge{}.jpg'.for
mat(num),edge)
cv2.imwrite('images/test/hg{}.jpg'.forma
t(num),res)
cv2.imwrite('images/test/hgimg{}.jpg'.fo
rmat(num),img)
```

کد مربوط به بخش فیلتر های سوبل

```
import numpy as np
import cv2
import skimage
import matplotlib.pyplot as plt


def normalizeimg(img,min,max):
    img = (img-min)/(max-min)*255
    return img
```

```
def combine(A,B):
    res = copy.deepcopy(A)
    for i in range(res.shape[0]):
        for j in range(res.shape[1]):
            res[i,j] = ((A[i,j]**2) +
(B[i,j]**2))**(1/2)
    return res
def gradient robert(img,num window):
    res = np.zeros(img.shape)
    num_window = num_window
    if num_window == 'a':
        #window a
        window = np.ones((2,2))
        window[0,0] = 1
        window[0,1] = 0
        window[1,0] = 0
        window[1,1] = -1
        sumw = 1
    elif num_window == 'b':
        # window b
        window = np.ones((2,2))
        window[0,0] = 0
        window[0,1] = 1
        window[1,0] = -1
        window[1,1] = 0
        sumw = 1
    paddingW2cntri =
int(window.shape[0]/2)
    paddingW2cntrj =
int(window.shape[1]/2)
    for i in
range(paddingW2cntri,img.shape[0]-
paddingW2cntri):
        for j in
range(paddingW2cntrj,img.shape[1]-
paddingW2cntrj):
            res[i,j] =
abs(sum(np.asarray(img[i:i+paddingW2cntr
i+1,j:j+paddingW2cntrj+1] *
window).flatten()))
    return res


img = cv2.imread('../Lena.bmp')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
noisy_img =
skimage.util.random_noise(img,
mode='gaussian', seed=None,
clip=True,var=.01)
noisy_img =
skimage.img_as_ubyte(noisy_img)

num_windowa = 'a'
edgea = gradient_robert(img,num_windowa)
noisy_edgea =
gradient_robert(noisy_img,num_windowa)
mmsea =
np.square(np.subtract(edgea,noisy_edgea)
).mean()

num_windowb = 'b'
edgeb = gradient_robert(img,num_windowb)
noisy_edgeb =
gradient_robert(noisy_img,num_windowb)
mmseb =
np.square(np.subtract(edgeb,noisy_edgeb)
).mean()

edgeAPlusB = combine(edgea,edgeb)
noisy_edgeAPlusB =
combine(noisy_edgea,noisy_edgeb)
).mean()

num_windowb = 'b'
edgeb = gradient(img,num_windowb)
noisy_edgeb =
gradient(noisy_img,num_windowb)
mmseb =
np.square(np.subtract(edgeb,noisy edgeb)
).mean()

num_windowc = 'c'
edgec = gradient(img,num_windowc)
noisy_edgec =
gradient(noisy_img,num_windowc)
mmsec =
np.square(np.subtract(edgec,noisy_edgec)
).mean()

resa = img + edgea
resb = img + edgeb
resc = img + edgec

# min = np.min(edgea)
# max = np.max(edgea)
# edgea = normalizeimg(edgea,min,max)
# min = np.min(edgeb)
# max = np.max(edgeb)
# edgeb = normalizeimg(edgeb,min,max)
# min = np.min(edgec)
# max = np.max(edgec)
# edgec = normalizeimg(edgec,min,max)

cv2.imwrite('../images/hw3.4.1/edge{}.jp
g'.format(num_windowa),edgea)
cv2.imwrite('../images/hw3.4.1/edge{}.jp
g'.format(num_windowb),edgeb)
cv2.imwrite('../images/hw3.4.1/edge{}.jp
g'.format(num_windowc),edgec)

cv2.imwrite('../images/hw3.4.1/res{}.jpg
'.format(num_windowa),resa)
cv2.imwrite('../images/hw3.4.1/res{}.jpg
'.format(num_windowb),resb)
cv2.imwrite('../images/hw3.4.1/res{}.jpg
'.format(num_windowc),resc)
cv2.imwrite('../images/hw3.4.1/img.jpg',
img)

data = [["mmse",mmsea,mmseb,mmsec]]
col labels = (" ", "window a","window
b","window c")
fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
ax.axis('off')
ax.table(cellText=data,
colLabels=col_labels, loc='center')
fig.savefig('../images/hw3.4.1/resTableA
verage.png')
```

کد مربوط به فیلترهای روبرتز

```
import numpy as np
import cv2
import skimage
import matplotlib.pyplot as plt
import copy

def normalizeimg(img,min,max):
    img = (img-min)/(max-min)*255
    return img
```

```
range(paddingW2cntrj,img.shape[1]-
paddingW2cntrj):
            res[i,j] =
(1/sumWndwW8)*sum(np.asarray(img[i-
paddingW2cntri:i+paddingW2cntri+1,j-
paddingW2cntrj:j+paddingW2cntrj+1] *
window).flatten())
    return res


def
unsharp_masking(img,k,window,sumWndwW8):
    smoothimg =
box_filter(img,window,sumWndwW8)
    mask = img - smoothimg
    res = img + k*mask
    return res,mask


windowList = [(3,3),(5,5),(7,7),(9,9)]
sumWndwW8 = 9
alphaList = [.1,.2,.4,.8,1]
img = cv2.imread('../Lena.bmp')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img = img.astype(np.float)
noisy_img =
skimage.util.random_noise(img,
mode='gaussian', seed=None,
clip=True,var=.3)
noisy_img =
skimage.img_as_ubyte(noisy_img)
noisy_img = noisy_img.astype(np.float)

mmseMatric = []
for window in windowList:
    mmseList =
["window={}".format(window)]
    #unsharp masking for img
    sumWndwW8 = np.sum(window)
    smoothimg =
cv2.GaussianBlur(img,window,1)
    mask = img - smoothimg
    #unsharp masking for noisy_img
    smoothimg_noisy =
cv2.GaussianBlur(noisy_img,window,.4)
    mask_noisy = noisy_img -
smoothimg_noisy
    for alpha in alphaList:
        res = img + (alpha*mask)
        res_noisy = noisy_img +
(alpha*mask_noisy)
        mmse =
np.square(np.subtract(res_noisy,res)).me
an()
        mmseList.append(mmse)

cv2.imwrite('../images/hw3.5/res{}(alpha
={}).jpg'.format(window,alpha),res)

    min = np.min(mask)
    max = np.max(mask)
    mask = normalizeimg(mask,min,max)

cv2.imwrite('../images/hw3.5/mask{}(alph
a={}).jpg'.format(window,alpha),mask)
    mmseMatric.append(mmseList)


data = mmseMatric
col_labels = ("alpha ",
"0.1","0.2","0.4","0.8","1")
```

```
mmseAPlusB =
np.square(np.subtract(edgeAPlusB,noisy_e
dgeAPlusB)).mean()


resa = img + edgea
resb = img + edgeb
resAPlusB = img + edgeAPlusB


min = np.min(edgea)
max = np.max(edgea)
edgea = normalizeimg(edgea,min,max)
min = np.min(edgeb)
max = np.max(edgeb)
edgeb = normalizeimg(edgeb,min,max)
min = np.min(edgeAPlusB)
max = np.max(edgeAPlusB)
edgeAPlusB =
normalizeimg(edgeAPlusB,min,max)


cv2.imwrite('../images/hw3.4.2/edge{}.jp
g'.format(num_windowa),edgea)
cv2.imwrite('../images/hw3.4.2/edge{}.jp
g'.format(num_windowb),edgeb)
cv2.imwrite('../images/hw3.4.2/edge{}.jp
g'.format('AB'),edgeAPlusB)

cv2.imwrite('../images/hw3.4.2/res{}.jpg
'.format(num_windowa),resa)
cv2.imwrite('../images/hw3.4.2/res{}.jpg
'.format(num_windowb),resb)
cv2.imwrite('../images/hw3.4.2/res{}.jpg
'.format('AB'),resAPlusB)

cv2.imwrite('../images/hw3.4.2/img.jpg',
img)

data = [["mmse",mmsea,mmseb,mmseAPlusB]]
col_labels = (" ", "window a","window
b","window A&B")
fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
ax.axis('off')
ax.table(cellText=data,
colLabels=col_labels, loc='center')
fig.savefig('../images/hw3.4.2/resTableA
verage.png')
```

کد مربوط به unsharp masking

```
import numpy as np
import cv2
import skimage
import matplotlib.pyplot as plt


def normalizeimg(img,min,max):
    img = (img-min)/(max-min)*255
    return img


def box_filter(img,window,sumWndwW8):
    res = np.zeros(img.shape)
    paddingW2cntri =
int(window.shape[0]/2)
    paddingW2cntrj =
int(window.shape[1]/2)
    for i in
range(paddingW2cntri,img.shape[0]-
paddingW2cntri):
        for j in
```

```
for i in range(151,200):
    line[i] = 101
x = np.linspace(0,200,200)
fig,axs =
plt.subplots(4,2,figsize=(5,10))
for i in range(0,len(windows)):

    axs[0,i].plot(x,line)
    axs[0,i].set_title("line window size
= {}".format(windows[i].shape[0]))
    smooth =
filter(line,windows[i],sizes[i])
    axs[1,i].plot(x,smooth)
    axs[1,i].set_title("smooth")

    mask = line - smooth
    axs[2,i].plot(x,mask)
    axs[2,i].set_title("mask")

    res = line + (k *mask)
    axs[3,i].plot(x,res)
    axs[3,i].set_title("result")
fig.tight_layout()
fig.savefig("fahim.png")
```

```
fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
ax.axis('off')
ax.table(cellText=data,
colLabels=col_labels, loc='center')
fig.savefig('../images/hw3.5/resTableAve
rage.png')


def filter(img,window,sumWndwW8):
    res = copy.deepcopy(img)
    paddingW2cntrj = int(len(window)/2)
    for j in
range(paddingW2cntrj,len(img)-
paddingW2cntrj):
        res[j] =
(1/sumWndwW8)*sum((img[j-
paddingW2cntrj:j+paddingW2cntrj+1] *
window))
    return res


windows = [np.ones(51),np.ones(5)]
line = np.zeros(200)
sizes = [51,5]
k = 4
for i in range(50,151):
    line[i] = i-50
```

مراجع

کتاب گنزالس رافائل

اسلایدهای دکتر امیرحسین طاهری نیا

با کمک های حل تمرین درس اقای بلویان و خانم رستمی و اقای امیر