فیلترها

رضا عباس زاده

اطلاعات گزارش	چکیده
تاریخ:	به منظور بهبود کیفیت تصاویر و حذف نقاط ضعف آن، از فیلترها استفاده می‌کنیم. در این تمرین با انواع نویزهای ننمک و فلفل و گاوسین آشنا می‌شویم و برای حذف آن‌ها از فیلترهای جعبه‌ای و میانه استفاده می‌کنیم. سپس با انواع فیلترها برای تشخیص و تقویت لبه‌ها در تصویر آشنا می‌شویم و نقاط ضعف و قوت هر یک را بررسی می‌کنیم.
واژگان کلیدی: فیلتر نویز Box filter Median filter Noise removal Edge detection Unsharp masking	

۱-مقدمه

نوشتار حاضر، به بررسی انواع فیلترهای مکانی و روش های پیاده سازی آن‌ها به منظور بهبود کیفیت تصویر، حذف نویز، تقویت لبه‌ها و... می‌پردازد.

۲-توضیح تکنیکال

۱-۱-۳ box filter

در این فیلتر، یک پنجره مربعی با مقادیر تماما یک، روی تصویر حرکت می‌دهیم(convolution). درواقع پنجره در هر موقعیتی که قرار گرفت، میانگین بدون وزن تمام عناصر موجود در پنجره محاسبه می‌شود که مقدار عنصر مرکز پنجره در تصویر فیلترشده را مشخص می‌کند.

Box filter

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

از این فیلتر برای هموارسازی (smoothing) تصاویر استفاده می‌شود. با استفاده از این فیلتر، جزییات کوچک تصویر حذف می‌شوند و تصویر تار یا blur می‌شود.

هرچه ابعاد پنجره بزرگتر باشد، میزان تاری تصویر افزایش می‌یابد.

مشکلات این فیلتر:

۱. لبه ها را به اندازه نصف طول ضلع پنجره از بین می‌برد. با استفاده از padding می‌توان این مشکل را حل کرد، اما این فاصله با هر سطح

خاکستری که ایجاد شود، در رنگ لبه‌ها تاثیر خواهد گذاشت و سایه ایجاد می‌کند.

۲. جزییات کوچک تصویر و جزییاتی که همرنگ محیط اطراف هستند را از بین می‌برد.

۳. در فیلترهای با اندازه پنجره بزرگ، بار محاسبات بسیار زیاد خواهد بود. به عنوان مثال با اعمال یک فیلتر ۱۰۰*۱۰۰ ، برای هر پیکسل موجود در تصویر،۱۰۰۰۰ عمل جمع انجام می‌شود که حتی در تصاویر کوچک باعث ایجاد بار محاسباتی می‌شود.

۳-۱-۲

با اعمال چندین باره این فیلتر:

- تصویر با هر بار استفاده تار تر می‌شود. به طوری که اگر این فیلتر را به تعداد دفعات زیادی روی تصویر اعمال کنیم، کل تصویر به یک رنگ تبدیل می‌شود و عملا تصویر از بین می‌رود.
- اگر از padding استفاده نشود، لبه‌های تصویر هربار از بین می‌روند و تصویر کوچکتر می‌شود.
- و اگر از padding استفاده کنیم تاثیر سایه ایجاد شده روی تصویر بیشتر می‌شود و لبه های تصویر متمایل به همان رنگ می‌شوند.
- باعث کاهش کنتراست تصویر می‌شود.

۳-۱-۳

برای پیاده سازی فیلتر جعبه‌ای مراحل زیر را انجام می-دهیم:

۱. به اندازه نصف اندازه پنجره پدینگ صفر در اطراف تصویر ایجاد می‌کنیم.

۲. فیلتر را روی تصویر حرکت می‌دهیم و convolve می‌کنیم.

۳. پدینگ ایجاد شده را حذف می‌کنیم.

۳-۱-۴

برای بررسی تاثیر اندازه پنجره در نتیجه فیلتر جعبه‌ای، با اندازه‌های ۳و ۹و ۱۵و ۲۷ فیلتر اعمال شده و نتیجه در بخش بعدی قرار داده شده است.

۳-۱-۵

برای بررسی تاثیر این فیلتر بر نویز، ابتدا با استفاده از کتابخانه skimage ، نویز نمک و فلفل را به تصویر اضافه می‌کنیم.

نویز نمک و فلفل:

این نویز به برخی از پیکسل‌های تصویر را به صورت رندوم، مقدار ۰ یا ۲۵۵ می‌دهد.

۳-۱-۶

برای اعمال فیلتر مشتق مرتبه دو(لاپلاسین):

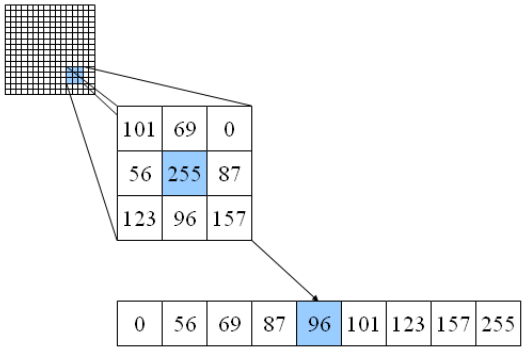
۱. ابتدا فیلتر را با عکس convolve می‌کنیم و نتیجه را نومال‌سازی می‌کنیم. با این کار جزییات تصویر باقی می‌ماند.

۲. سپس نتیجه مرحله قبل را با تصویر اصلی جمع می‌کنیم و نرمال‌سازی انجام می‌دهیم. در نتیجه جزییات تصویر واضح‌تر می‌شوند.

۳-۲

۳-۲-۱ فیلتر میانه

این فیلتر مقدار هر پیکسل را برابر با میانه سطح خاکستری همسایه‌ها (با توجه به اندازه پنجره) قرار می‌دهد. با این کار نویزها (مخصوصا نویز ننمک و فلفل) که معمولا مقدار دورتری از میانگین همسایه‌هایشان دارند به ابتدا یا انتهای بازه منتقل می‌شوند و در محاسبه میانه تاثیر نخواهند گذاشت. با این ترتیب نویز تصویر حذف می‌شود اما نسبت به فیلتر جعبه‌ای تصویر بسیار واضح‌تر باقی می‌ماند.



۲-۳-2 نویز گاوسین

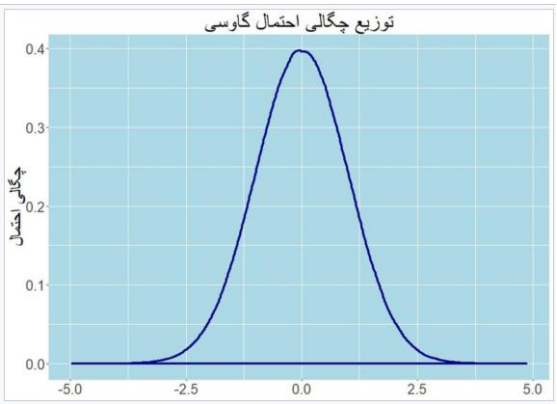
یک نویز استاتیک است که از تابع توزیع چگالی احتمال نرمال(تابع گاوسی) پیروی می‌کند؛ یعنی، مقادیر این نویز، توزیع گاوسی دارند.

نویز گاوسی در تصاویر و ویدیوها توزیعی تصادفی از آرتیفکت‌ها است که باعث می‌شود تصاویر کم‌رنگ و تار به نظر برسند. اگر از نزدیک و با دقت به تصاویر روی فیلم‌های گرفته شده با دوربین‌های قدیمی‌تر یا تصاویر نوارهای ویدیویی که مدت طولانی آرشیو شده‌اند نگاه شود، ذرات ریز با الگوهای تصادفی قابل مشاهده است.

برای ایجاد این نویز باید به صورت رندوم تابع گاوسین را به تصویر اضافه کنیم:

$$p(g) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$

که g مقدار خاکستری، σ انحراف معیار و μ برابر میانگین است.



در این تمرین ابتدا با استفاده از کتابخانه skimage نویز گاوسی را به تصویر اضافه می‌کنیم و سپس با استفاده از فیلترهای جعبه‌ای و میانه سعی می‌کنیم این نویز را از بین ببریم و نتیجه mse تصویر اصلی با تصاویر رفع نویز شده در قالب جدول گزارش شده است.

۳-۲-۳

برای خذف نویز گاوسی و نمک و فلفل باید سه مرحله زیر انجام شود:

۱. ابتدا پیکسل های لبه توسط یک فیلتر لبه یاب در کل تصویر شناسایی می شوند.
۲. در اطراف یک پیکسل معین ، یک منحنی تکه تکه خطی از پیکسل های لبه شناسایی شده توسط یک الگوریتم ساده اما کارآمد تخمین زده می شود تا تقسیم بندی قسمت لبه زیرین در آن منطقه از تصویر انجام شود.
۳. شدت تصویر مشاهده شده در همان سمت قطعه لبه برآورد شده ، مثل پیکسل داده شده ، برای برآورد شدت واقعی تصویر در پیکسل داده شده ، با روش linear kernel smoothing محلی به دست می آید.

۳-۳-۱

با استفاده از دوربین گوشی همراه، تصویری تار و نویزی در نور کم گرفتم و سعی کردم کیفیت آن را بهبود دهم.

ابتدا هیستوگرام تصویر را همسان‌سازی کردم و با استفاده از فیلتر میانه نویز های کوچک تصویر را از بین بردم. سپس با استفاده از فیلتر های لبه یاب عمودی و افقی سوبل، لبه-های تصویر اصلی را شناسایی کردم و آن ها را با ضریب ۰.۸ به نتیجه مرحله قبل اضافه کردم. درنهایت به منظور smooth تر شدن تصویر به خصوص در نواحی پر از نویز، از فیلتر جعبه‌ای استفاده کردم.

۳-۴ تشخیص لبه edge detection

۳-۴-۱ سوبل

هدف آشکارسازی لبه نشان‌گذاری نقاطی از یک تصویر است که در آنها شدت روشنایی به تندی تغییر می‌کند. تغییرات تند در خصوصیات تصویر معمولاً نماینده رویدادهای مهم و تغییرات در خصوصیات محیط هستند.

با توجه به این توضیح، در لبه های تصویر تابع شدت تصویر دچار تغییر می‌شود. بنابراین قله ها در مشتق این تابع نشانگر لبه ها می‌باشد.

برای گرفتن مشتق اول تصویر، یک فیلتر را روی آن convolve می‌کنیم. با اعمال فیلتر های زیر تصویر به صورت افقی مقایسه می‌شود و لبه‌های عمودی تشخیص داده می‌شوند. برای لبه‌های افقی باید فیلترهای داده شده را ۹۰ درجه بچرخانیم و جداگانه convolve کنیم.

فیلتر a: این فیلتر اختلاف دو پیکسل در سمت چپ و راست (افقی) را محاسبه می‌کند. به دلیل اینکه محدودیت‌های کمی اعمال می‌کند ممکن است در تشخیص لبه‌ها دچار خطا شود.

فیلتر b: این فیلتر پیکسل‌های قطری را هم درنظر می‌گیرد. بنابراین دقت مناسبتری دارد اما ضرایب پیکسل‌های قطری و افقی برابر هستند.

فیلتر c: مانند فیلتر b است اما وزن همسایه‌های افقی نسبت به قطری بیشتر است.

در این فیلترها، نیاز به انتخاب یک حدthreshold وجود دارد تا حداقل تفاوت دو سطح برای تشخیص یک لبه مشخص شود.

۳-۴-۲ رابرت

دو فیلتر دیگر برای تشخیص لبه ها با استفاده از مشتق اول، فیلترهای رابرت هستند:

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

این فیلتر لبه‌های نازکتری ایجاد خواهد کرد. چراکه به ازای یک لبه، یک ردیف پیکسل ایجاد می‌کند. برعکس سوبل که به ازای هر لبه، دو ردیف پیکسل ایجاد می‌کند.

فیلترهای رابرت بار محاسباتی بسیار کمتری نسبت به سوبل دارند. همچنین نیاز به پیدا کردن پارامتر threshold ندارند.

در هر دو نوع فیلترها، از نتیاج منفی قدرمطلق گرفته می‌شود.

۳-۵ Unsharp masking

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

با توجه به فرمول بالا، ابتدا یک نسخه smooth شده از تصویر ایجاد می‌کنیم که جززیات تصویر در آن از بین رفته است. سپس نتیجه را از تصویر اصلی کم می‌کنیم. به این ترتیب تنها جزییاتی که در مرحله قبل حذف شده بودند، باقی می‌مانند. این جزییات را با ضریبی مناسب به تصویر اصلی اضافه می‌کنیم تا جزییات تصویر اصلی تقویت شوند. این ضریب باید به گونه‌ای انتخاب شود که مقادیر آن بسیار منفی یا بزگتر از ۲۵۵ نشوند تا اطلاعات تصویر حفظ شوند.

۳-شرح نتایج

۳-۱-۳

تصویر اصلی

نتیجه اعمال فیلتر ۳*۳ جعبه ای با تعداد دفعات مختلف:

۱ مرتبه

۳ مرتبه

۵ مرتبه

۱۰ مرتبه

۱۵ مرتبه

۳-۱-۴

تصویر اصلی به همراه نویز نمک و فلفل با تراکم ۰.۰۵:

نتایج اعمال فیلتر جعبه‌ای با اندازه پنجره های مختلف روی تصویر با نویز نمک و فلفل:

3x3 window

9x9 window

15x15 window

27x27 window

همانطور که مشخص است هرچه ابعاد پنجره فیلتر بزرگتر باشد، میزان تاری تصویر افزایش می‌یابد. از طرفی نویزهای اضافه شده به تصویر کم کم از بین می‌روند به طوری که با فیلتر ۱۵*۱۵ تقریبا نویزی وجود ندارد اما تصویر به میزان زیادی تار شده است.

۳-۱-۵

با توجه به نتایج سوال قبل، برای حفظ اطلاعات تصویر و حذف نویز، به نظر می‌رسد فیلتر ۹*۹ مناسب باشد.
چراکه هم جزییات تصویر کاملا از بین نرفته است و هم نویز ها کاهش یافته اند.
در مجموع می‌توان نتیجه گرفت که فیلتر جعبه‌ای برای حذف نویز مناسب نیست و بیشتر باعث از دست رفتن اطلاعات تصویر می‌شود.

۳-۱-۶

نتیجه اعمال فیلتر لاپلاسین به تصویر تار شده در تمرین ۳-۱-۳:

تصویر اصلی

تصویر با ۲ بار اعمال فیلتر لاپلاسین

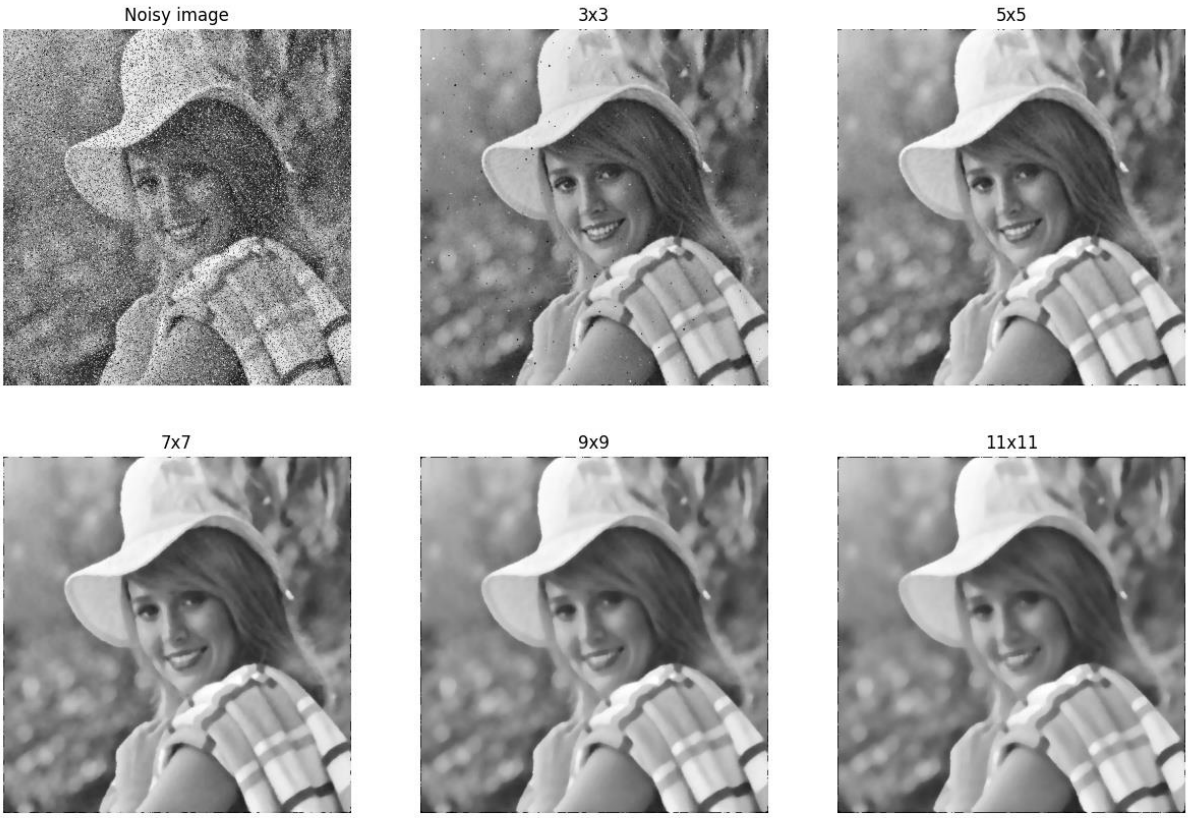
تصویر با ۱ بار اعمال فیلتر لاپلاسین

تصویر با ۳ بار اعمال فیلتر لاپلاسین

۳-۲-۱

نتیجه اعمال فیلتر میانه بر روی تصویر با نویز نمک و فلفل با تراکم ۰.۰۵

نتیجه اعمال فیلتر میانه بر روی تصویر با نویز نمک و فلفل با تراکم ۰.۱



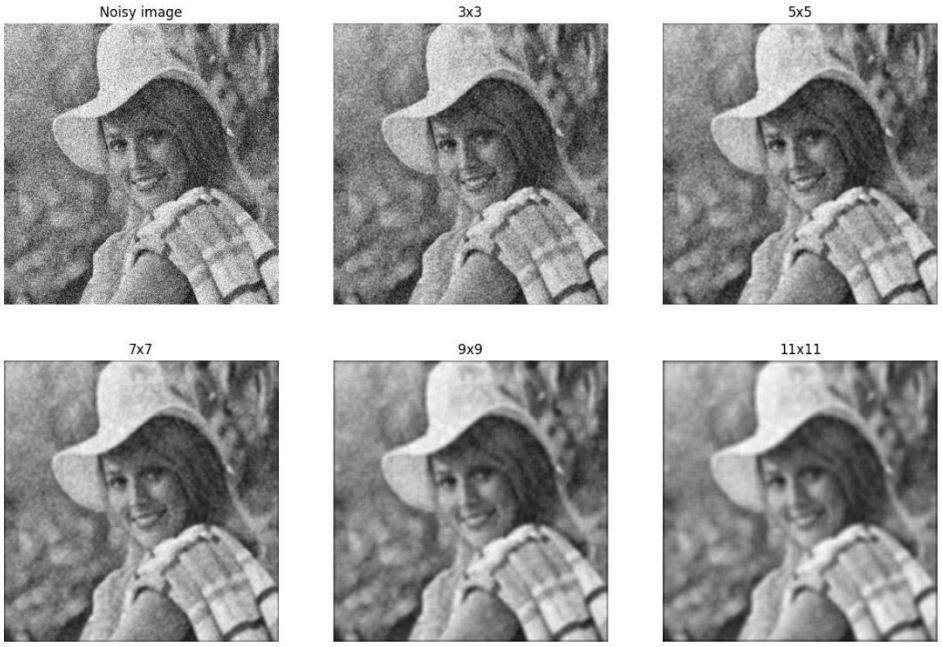
نتیجه اعمال فیلتر میانه بر روی تصویر با نویز نمک و فلفل با تراکم ۰.۲

به طور کلی فیلتر میانه، برای از بین بردن این نوع نویز بسیار موثر است. در چگالی پایین نویز، استفاده از پنجره کوچکتر بهتر است زیرا جزییات تصویر کمتر از بین می‌رود. و در چگالی بالا نویز، استفاده از پنجره‌های بزرگتر مناسبتر است.

	3X3	5x5	7x7	9x9	11x11
p = 0.05	28.192302703857422	32.937171936035156	35.22077178955078	38.842952728271484	41.86066818237305
p = 0.1	29.320789337158203	33.65595626831055	35.788509368896484	39.368858337402344	42.27154541015625
p = 0.2	31.772769927978516	34.90841293334961	36.75908279418945	40.095306396484375	42.92881774902344

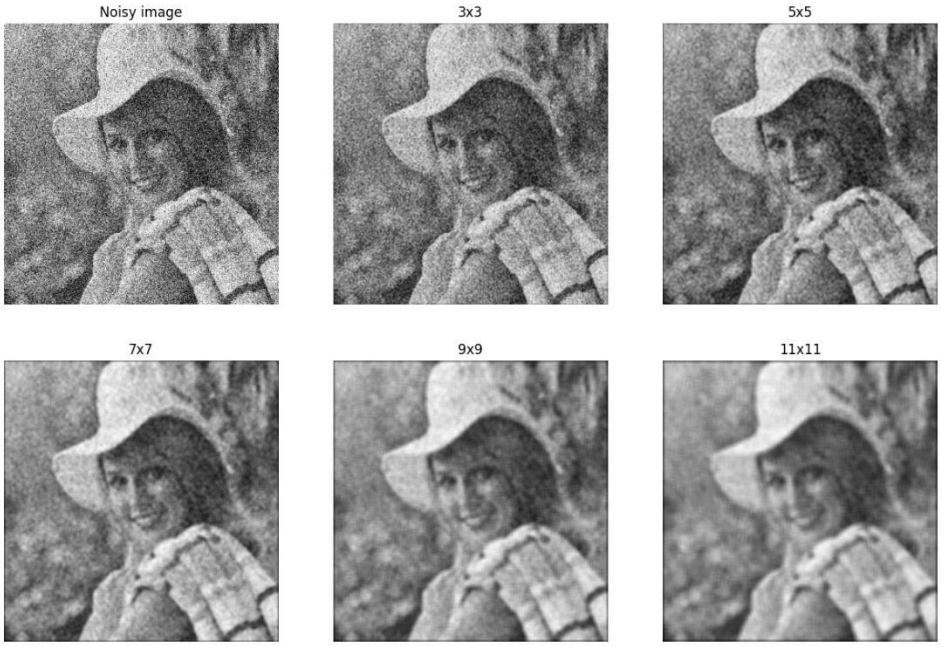
با توجه به جدول می‌توان نتیجه گرفت که هرچه اندازه پنجره کوچکتر باشد، نتیجه به تصویر اصلی نزدیکتر خواهد بود.

۳-۲-۲

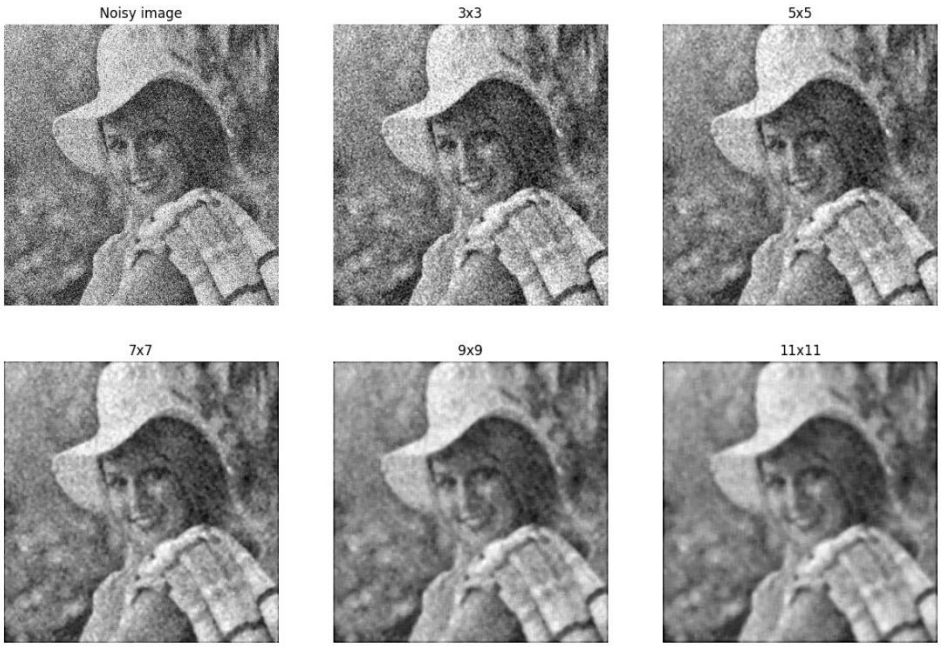


نتیجه اعمال فیلتر **جعبه‌ای** بر روی تصویر با نویز گاوسی با تراکم ۰.۰۵

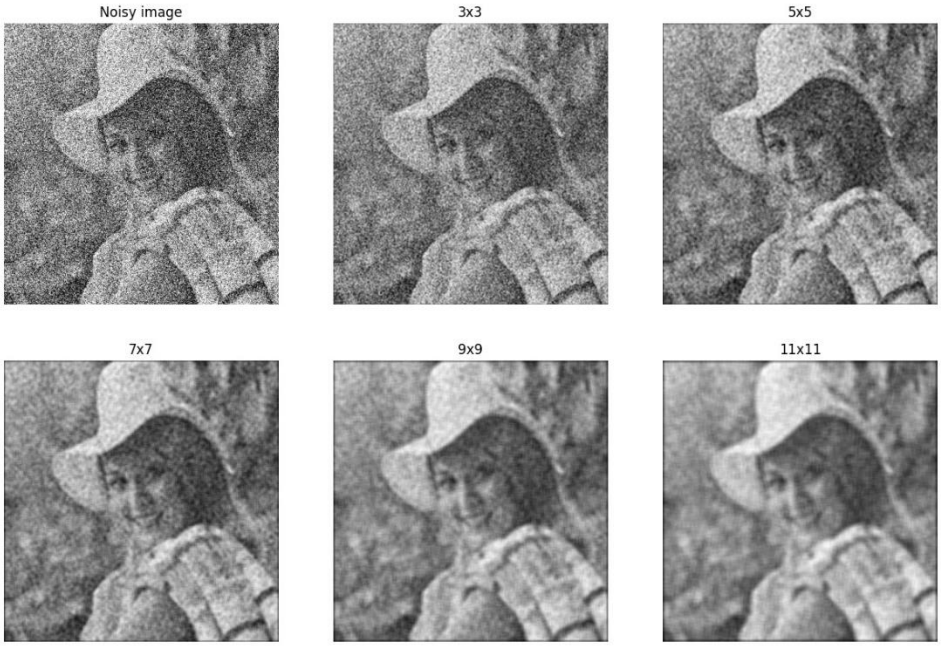
نتیجه اعمال فیلتر **میانه** بر روی تصویر با نویز گاوسی با تراکم ۰.۰۵



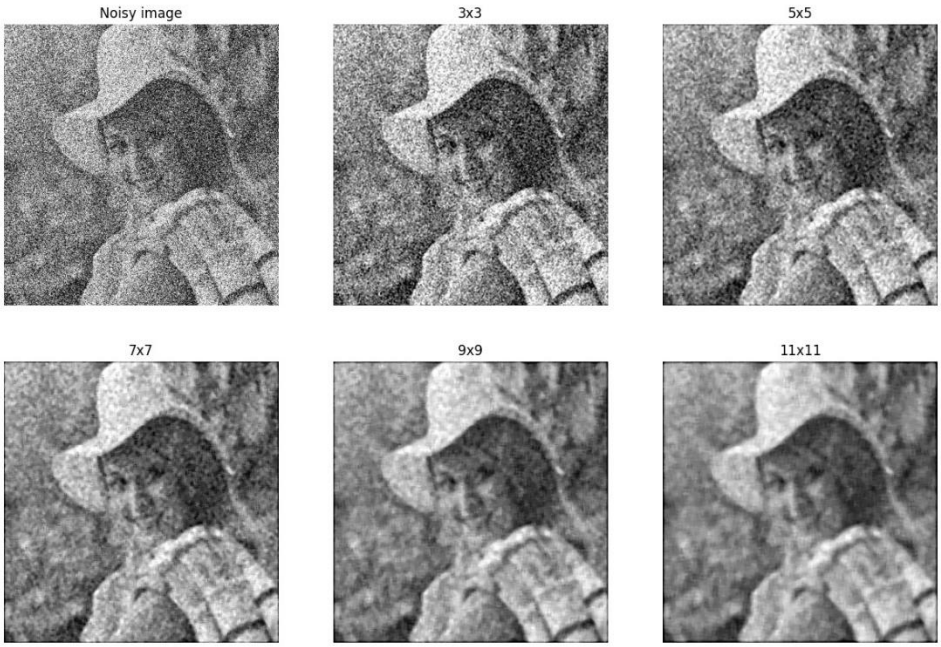
نتیجه اعمال فیلتر **جعبه‌ای** بر روی تصویر با نویز گاوسی با تراکم ۰.۱



نتیجه اعمال فیلتر **میانه** بر روی تصویر با نویز گاوسی با تراکم ۰.۱



نتیجه اعمال فیلتر **جعبه‌ای** بر روی تصویر با نویز گاوسی با تراکم ۰.۲



نتیجه اعمال فیلتر **میانه** بر روی تصویر با نویز گاوسی با تراکم ۰.۲

	3X3	5x5	7x7	9x9	11x11
p = 0.05	86.33770370483398	74.55221557617188	69.00286102294922	67.28031921386719	67.40365600585938
p = 0.1	93.73309707641602	84.51886749267578	79.79100799560547	77.82550430297852	77.29903793334961
p = 0.2	99.14425659179688	92.61767578125	89.49714660644531	87.92359161376953	87.42602157592773

جدول خطای فیلتر **جعبه ای** با ابعاد مختلف

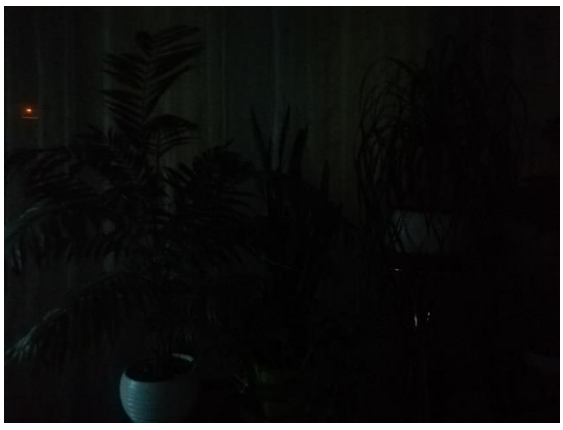
	3X3	5x5	7x7	9x9	11x11
p = 0.05	92.76668548583984	80.1616096496582	71.82543182373047	67.13553619384766	65.42524337768555
p = 0.1	99.6280746459961	89.86278533935547	81.64241790771484	76.32162094116211	73.20489120483398
p = 0.2	104.04870223999023	97.30248641967773	90.4967269897461	85.3046760559082	81.67327499389648

جدول خطای فیلتر **میانه** با ابعاد مختلف

همانطور که از جدول‌ها مشخص است، فیلتر جعبه‌ای عمدتا با ابعاد پنجره کوچک عملکرد بهتری از فیلتر میانه دارد. اما با ابعاد بزرگ پنجره عملکرد فیلتر میانه نسبت به فیلتر جعبه‌ای بهتر می‌شود.

۳-۳-۱

تصویر اولیه:



تصویر بعد از equalization:

نویز زیادی در تصویر مشاهده می‌شود.

بعد از اعمال فیلتر میانه:

تشخیص لبه‌ها با استفاده از فیلتر سوبل افقی و عمودی و اضافه کردن آن به تصویر بالا:

Smooth کردن تصویر با کمک فیلتر جعبه‌ای:

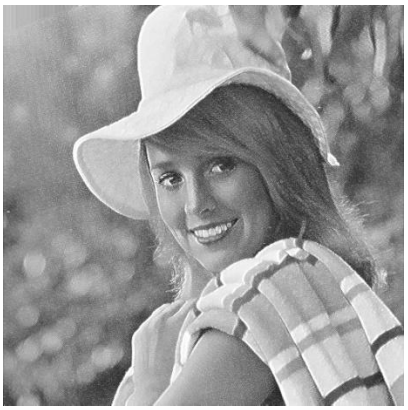
نتیجه نهایی نویز بسیار کمتری دارد و جزییات در آن قابل مشاهده هستند.

۳-۴-۱

فیلترها برای نمایش بهتر، نرمال‌سازی شده اند.

لبه های تشخیص داده شده با استفاده از فیلترهای مختلف و نتایج اضافه کردن آن‌ها به تصویر اصلی:

فیلتر a:



فیلتر b:

فیلتر a:

فیلتر c:

فیلتر b:

۳-۴-۲

فیلترها برای نمایش بهتر، نرمال‌سازی شده اند.

لبه های تشخیص داده شده با استفاده از فیلترهای مختلف و نتایج اضافه کردن آن‌ها به تصویر اصلی:

با بررسی این دو فیلتر متوجه می‌شویم که فیلترهای رابرت جزییات بیشتری را در نظر می‌گیرند و نسبت به نویز حساس‌تر هستند.

بنابراین در این مثال، فیلتر های c و b عملکرد بهتری دارند.

۳-۵-۱

نتیجه unsharp masking با اندازه پنجره ۳*۳ و آلفاهای مختلف

نتیجه unsharp masking با اندازه پنجره ۵*۵ و آلفاهای مختلف

نتیجه unsharp masking با اندازه پنجره ۷*۷ و آلفاهای مختلف

نتیجه unsharp masking با اندازه پنجره ۹*۹ و آلفاهای مختلف

نتیجه unsharp masking با اندازه پنجره ۱۱*۱۱ و آلفاهای مختلف

با مقایسه تصاویر بالا به این نتیجه می‌رسیم که برای شارپ شدن بیشتر تصویر دو راه داریم. ۱. افزایش ابعاد پنجره ۲. افزایش آلفا. با افزایش سایز پنجره اختلاف دو تصویر بیشتر می‌شود و با افزایش آلفا می‌توان تاثیر این اختلاف را بیشتر کرد.

پیوست

۳-۱

تابع‌های اضافه و حذف کردن padding:

```
def add_padding(img, padSize, padValue=0):
    padded = np.ones((img.shape[0] + padSize * 2, img.shape[1] + padSize *
2), np.uint8) * padValue
    padded[padSize:img.shape[0] + padSize, padSize:img.shape[1] + padSize]
= img
    return padded


def cut_padding(img, padSize):
    return img[padSize: img.shape[0] - padSize,
           padSize: img.shape[1] - padSize]
```

تابعی که یک پنجره را با یک تصویر convolve می‌کند:

```
def convolve_filter(img, window):
    padding_size = int(window.shape[0] / 2)
    padded_img = add_padding(img, padding_size)
    filtered = np.zeros(padded_img.shape, np.uint8)
    sum_of_filter = sum(window.flatten())

    for i in range(padding_size, img.shape[0] + padding_size):
        for j in range(padding_size, img.shape[1] + padding_size):
            filtered[i, j] = (1 / sum_of_filter) * sum(
                np.multiply(
                    np.asarray(padded_img[i - padding_size:i + padding_size
+ 1,
                                          j - padding_size:j + padding_size + 1]
                                          ),
                    window
                )
                    .flatten()
            )
    return cut_padding(filtered, padding_size)
```

تابع فیلتر جعبه‌ای و لاپلاسین:

```
def box_filter(img, window_size):
    return convolve_filter(img, np.ones((window_size, window_size)))


def laplacian_filter(img, window):
    return convolve_filter(img, window)
```

تابع نرمال‌سازی تصویر:

```
def normalize(img):
    min = np.min(img)
    max = np.max(img)
    img = (img - min) / (max - min) * 255
    return img
```

سوالات ۳ تا ۶ با کامنت نشانه‌گذاری شده اند.

```
import numpy as np
import cv2
import copy
import skimage

from common import box_filter, laplacian_filter, normalize

img = cv2.imread('Elaine.bmp', cv2.IMREAD_GRAYSCALE)
box_filtered = copy.deepcopy(img)

# 3-1-3
repeats = 10
for i in range(0, repeats):
    box_filtered = box_filter(box_filtered, window_size=3)
    cv2.imwrite('3-1/box-filtered-3x3-it{}.jpg'.format(i + 1),
box_filtered)

# 3-1-4
window_sizes = [3, 9, 15, 27]
noisy_img = skimage.util.random_noise(img, mode='s&p', seed=None,
clip=True, amount=0.05)
noisy_img = skimage.img_as_ubyte(noisy_img)
cv2.imwrite('3-1/noise.jpg', noisy_img)
for size in window_sizes:
    res = box_filter(noisy_img, window_size=size)
    cv2.imwrite('3-1/box-filtered-{}x{}.jpg'.format(size, size), res)

# 3-1-5
filtered_noisy = box_filter(noisy_img, 5)
cv2.imwrite('3-1/noise-reduction.jpg', filtered_noisy)


# 3-1-6
sharpened = copy.deepcopy(img)
for i in range(0, 3):
    filtered = laplacian_filter(sharpened, np.asarray([[0, -1, 0], [-1, 5,
-1], [0, -1, 0]]))
    filtered_norm = normalize(filtered)
    sharpened = normalize(filtered_norm + img)
    cv2.imwrite('3-1/laplacian-filtered-{}.jpg'.format(i), sharpened)
```

۳-۲-۱

تابع فیلتر میانه:

```
def median_filter(img, windowSize):
    padding_size = int(windowSize / 2)
    padded_img = add_padding(img, padding_size)
    filtered = np.zeros(padded_img.shape, np.uint8)

    for i in range(padding_size, img.shape[0] + padding_size):
        for j in range(padding_size, img.shape[1] + padding_size):
            filtered[i, j] = np.median(np.asarray(padded_img[i -
padding_size:i + padding_size + 1,
                                                             j - padding_size:j +
padding_size + 1]
                                                             ))
    return cut_padding(filtered, padding_size)
```

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

from common import median_filter

img = cv2.imread('Elaine.bmp', cv2.IMREAD_GRAYSCALE)
densities = [0.05, 0.1, 0.2]
window_sizes = [3, 5, 7, 9, 11]
MSEs = []
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('3-2-1/3-2-1-den{}.jpg'.format(density), noisy_img)
    MSEs_of_density = ['p = {}'.format(density)]

    fig, axs = plt.subplots(2, 3, figsize=(15, 10))
    axs[0][0].imshow(noisy_img, cmap='gray', aspect='auto')
    axs[0][0].set_title('Noisy image')
    axs[0][0].axis('off')
    i = 0
    for size in window_sizes:
        res = median_filter(noisy_img, size)
        MSEs_of_density.append(np.square(np.subtract(img, res)).mean())

        i += 1
        axs[int(i / 3)][i % 3].imshow(res, cmap='gray', aspect='auto')
        axs[int(i / 3)][i % 3].set_title('{}x{}'.format(size, size))
        axs[int(i / 3)][i % 3].axis('off')
    fig.savefig('3-2-1/3-2-1-den{}-res.jpg'.format(density))
    MSEs.append(MSEs_of_density)

col_labels = ("       ", "3X3", "5x5", "7x7", "9x9", "11x11")
fig, ax = plt.subplots(dpi=300, figsize=(5, 1))
ax.axis('off')
ax.table(cellText=MSEs, colLabels=col_labels, loc='center')
fig.savefig('3-2-1/3-2-1-resTable.png')
```

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

from common import median_filter, box_filter

img = cv2.imread('Elaine.bmp', cv2.IMREAD_GRAYSCALE)
variances = [0.05, 0.1, 0.2]
sizeWndws = [3, 5, 7, 9, 11]
MSEs_median = []
MSEs_box = []
for variance in variances:
    noisy_img = skimage.util.random_noise(img, mode='gaussian', seed=None,
clip=True, var=variance)
    noisy_img = skimage.img_as_ubyte(noisy_img)
    cv2.imwrite('3-2-2/noisy-den{}.jpg'.format(variance), noisy_img)
    MSEs_of_density_median = ['p = {}'.format(variance)]
    MSEs_of_density_box = ['p = {}'.format(variance)]

    fig_box, axs_box = plt.subplots(2, 3, figsize=(15, 10))
    axs_box[0][0].imshow(noisy_img, cmap='gray', aspect='auto')
    axs_box[0][0].set_title('Noisy image')
    axs_box[0][0].axis('off')

    fig_median, axs_median = plt.subplots(2, 3, figsize=(15, 10))
    axs_median[0][0].imshow(noisy_img, cmap='gray', aspect='auto')
    axs_median[0][0].set_title('Noisy image')
    axs_median[0][0].axis('off')
    i = 0
    for size in sizeWndws:
        i += 1
        median = median_filter(noisy_img, size)
        MSEs_of_density_median.append(np.square(np.subtract(img,
median)).mean())
        axs_median[int(i / 3)][i % 3].imshow(median, cmap='gray',
aspect='auto')
        axs_median[int(i / 3)][i % 3].set_title('{}x{}'.format(size, size))
        axs_median[int(i / 3)][i % 3].axis('off')

        box = box_filter(noisy_img, size)
        MSEs_of_density_box.append(np.square(np.subtract(img, box)).mean())
        axs_box[int(i / 3)][i % 3].imshow(box, cmap='gray', aspect='auto')
        axs_box[int(i / 3)][i % 3].set_title('{}x{}'.format(size, size))
        axs_box[int(i / 3)][i % 3].axis('off')

    fig_median.savefig('3-2-2/den{}-median.jpg'.format(variance))
    fig_box.savefig('3-2-2/den{}-box.jpg'.format(variance))

    MSEs_median.append(MSEs_of_density_median)
    MSEs_box.append(MSEs_of_density_box)


col_labels = ("       ", "3X3", "5x5", "7x7", "9x9", "11x11")
fig, ax = plt.subplots(dpi=300, figsize=(5, 1))
ax.axis('off')
ax.table(cellText=MSEs_median, colLabels=col_labels, loc='center')
fig.savefig('3-2-2/table_median.png')
```

```
col_labels = ("       ", "3X3", "5x5", "7x7", "9x9", "11x11")
fig, ax = plt.subplots(dpi=300, figsize=(5, 1))
ax.axis('off')
ax.table(cellText=MSEs_box, colLabels=col_labels, loc='center')
fig.savefig('3-2-2/table_box.png')
```

۳-۳

```
import cv2
import numpy as np
from common import globalHistEq, median_filter, convolve_filter,
unsharp_masking, normalize, box_filter

def gradient(img, window):
    res = np.zeros(img.shape)
    padding_v = int(window.shape[0] / 2)
    padding_h = int(window.shape[1] / 2)
    is_odd = 1
    if window.shape[0] == 2:
        is_odd = 0

    for i in range(padding_v, img.shape[0] - padding_v):
        for j in range(padding_h, img.shape[1] - padding_h):
            res[i, j] = abs(sum(np.asarray(img[i - padding_v:i + padding_v
+ is_odd,
                                           j - padding_h:j + padding_h +
is_odd] * window).flatten()))
    return res

sobel_v = (1 / 8) * np.array([
    [1, 0, -1],
    [2, 0, -2],
    [1, 0, -1]
])
sobel_h = (1 / 8) * np.array([
    [1, 2, 1],
    [0, 0, 0],
    [-1, -2, -1]
])

for i in [5]:
    img = cv2.imread('noisy{}.jpg'.format(i), cv2.IMREAD_GRAYSCALE)
    eq = globalHistEq(img)
    cv2.imwrite('3-3/{}eq.jpg'.format(i), eq)

    median_filtered = median_filter(eq, 5)
    cv2.imwrite('3-3/{}median.jpg'.format(i),
globalHistEq(median_filtered))

    edge_v = normalize(gradient(img, sobel_v))
    edge_h = normalize(gradient(img, sobel_h))
    edged = normalize(median_filtered + 0.8 * (edge_h +
edge_v)).astype(np.uint8)
    cv2.imwrite('3-3/{}edge.jpg'.format(i), globalHistEq(edged))

    box_filtered = box_filter(edged, 3)
    cv2.imwrite('3-3/{}filtered.jpg'.format(i), globalHistEq(box_filtered))
```

```
import numpy as np
import cv2

from common import normalize


def gradient(img, window):
    res = np.zeros(img.shape)
    padding_v = int(window.shape[0] / 2)
    padding_h = int(window.shape[1] / 2)
    is_odd = 1
    if window.shape[0] == 2:
        is_odd = 0
    for i in range(padding_v, img.shape[0] - padding_v):
        for j in range(padding_h, img.shape[1] - padding_h):
            res[i, j] = abs(sum(np.asarray(img[i - padding_v:i + padding_v
+ is_odd,
                                               j - padding_h:j + padding_h +
is_odd] * window).flatten()))

    return res

a = (1 / 2) * np.array([
    [1, 0, -1]
])

b = (1 / 6) * np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])

c = (1 / 8) * np.array([
    [1, 0, -1],
    [2, 0, -2],
    [1, 0, -1]
])

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

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

filter_names = ['a', 'b', 'c', 'd', 'e']
filters = [
    a, b, c, d, e
]

img = cv2.imread('Elaine.bmp', cv2.IMREAD_GRAYSCALE)

for i in range(len(filters)):
    edge = normalize(gradient(img, filters[i]))
    cv2.imwrite('3-4-1/edge-{}.jpg'.format(filter_names[i]), edge)
    cv2.imwrite('3-4-1/res-{}.jpg'.format(filter_names[i]), img + edge)
```

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

from common import convolve_filter, normalize


def unsharp_masking(img, k, window):
    smoothed = convolve_filter(img, window)
    mask = img - smoothed
    res = img + k * mask
    return normalize(res)


windowList = [(3, 3), (5, 5), (7, 7), (9, 9), (11, 11)]
alphas = [.1, .3, .6, .8, 1]
img = cv2.imread('Elaine.bmp', cv2.IMREAD_GRAYSCALE).astype(np.float)

for window in windowList:
    fig, axs = plt.subplots(2, 3, figsize=(15, 10))
    axs[0][0].imshow(img, cmap='gray', aspect='auto')
    axs[0][0].set_title('Normal image')
    axs[0][0].axis('off')
    i = 0
    for alpha in alphas:
        res = unsharp_masking(img, alpha, np.ones(window))
        i += 1
        axs[int(i / 3)][i % 3].imshow(res, cmap='gray', aspect='auto')
        axs[int(i / 3)][i % 3].set_title('alpha = {}'.format(alpha))
        axs[int(i / 3)][i % 3].axis('off')

    fig.savefig('3-5/res{}.jpg'.format(window))
```