

گزارش تمرین Filters

مریم واقعی

اطلاعات گزارش	چکیده
تاریخ:	ما در این گزارش پیاده سازی تمرین Filters را با استفاده از زبان برنامه نویسی متلب و در بخشی با زبان پایتون انجام داده ایم. در بخش اول تمرین ابتدا فیلتر box و لاپلاسین را روی تصویر اعمال میکنیم و نتایج خروجی را بررسی کردیم. سپس در بخش بعدی median با سایز های مختلف را روی تصاویر نویزی شده با نویز نمک و فلفل با ضریب نویز مختلف اعمال کردیم و خروجی را با تصویر اصلی مقایسه کردیم و خطای MSE را برای آنها محاسبه کردیم.
واژگان کلیدی:	پس از آن ما نویز گوسی را روی تصویر اصلی با واریانس های مختلف اعمال کردیم و پس از آن فیلتر box و median را روی آن با سایز های مختلف اعمال کردیم و نتیجه را با تصویر اصلی با معیار MSE مقایسه کردیم.
فیلتر گوسی	در بخش بعد ما لبه های تصویر را به کمک یکسری فیلتر مثل فیلتر Robert و first order difference بدست آوردیم و نتایج را با هم مقایسه کردیم.
فیلتر روبرت	
median	
فیلتر laplacian	
نويز نمک و فلفل	
MSE	
ضریب آلفا	
لبه تصویر	
Unsharp mask	

فهرست مطالع

٤	۱-مقدمه
٤	۲- توضیحات فنی
٤	Box Filter -۱-۲
٤	۲-۱-۱- بخش اول
٤	۲-۱-۲- بخش دوم
٤	۲-۱-۳- بخش سوم
٤	۲-۱-۴- بخش چهارم
۵	۲-۱-۵- بخش پنجم
۵	۲-۱-۶- بخش ششم
۵	Median Filter -۲-۲
۵	۲-۲-۱- بخش اول
۵	۲-۲-۲- بخش دوم
۶	Edge Detection -۳-۲
۶	۳-۱-۱- بخش اول
۶	۳-۱-۲- بخش دوم
۶	Unsharp Masking -۴-۲
۶	۳-بررسی نتایج
۶	Box Filter -۱-۳
۶	۱-۱-۱- بخش اول
۶	۱-۱-۲- بخش دوم
۷	۱-۱-۳- بخش سوم
۸	۱-۱-۴- بخش چهارم
۱۰	۱-۱-۵- بخش پنجم
۱۰	۱-۱-۶- بخش ششم
۱۵	Median Filter -۲-۳
۱۵	۲-۱-۱- بخش اول
۱۷	۲-۱-۲- بخش دوم
۲۱	Edge Detection -۳-۳
۲۱	۳-۱-۱- بخش اول

۲۳ بخش دوم ۲-۳-۳
۲۳ Unsharp Masking ۴-۳
۲۶ پیوست ۴
۲۶ Box Filter ۱-۴
۲۶ ۱-۱- بخش اول ۴
۲۶ ۲-۱- بخش دوم ۴
۲۶ ۳-۱- بخش سوم ۴
۲۷ ۲-۱-۲- بخش چهارم ۴
۲۷ ۵-۱- بخش پنجم ۴
۲۷ ۶-۱- بخش ششم ۴
۲۸ Median Filter ۲-۴
۲۸ ۱-۲-۱- بخش اول ۴
۲۸ ۲-۲-۲- بخش دوم ۴
۲۹ Edge Detection ۳-۴
۲۹ ۱-۳-۱- بخش اول ۴
۳۰ ۲-۳-۲- بخش دوم ۴
۳۰ Unsharp Masking ۴-۴

۱- مقدمه

در ابتدا ما نحوه پیاده سازی بخش های مختلف تمرین را توضیح داده ایم. سپس در بخش بعدی نتایج را نمایش میدهیم و با هم مقایسه میکنیم که برای مقایسه برخی نتایج مانند نتایج اعمال فیلتر median و box روی تصویر ما از خطای MSE استفاده کرده ایم. در نهایت در بخش پیوست ما کد های هر بخش از تمرین را قرار داده ایم.

۲- توضیحات فنی

Box Filter - ۱-۲

۱-۱-۲- بخش اول

Box filter به همه پیکسل های اطراف پیکسل مرکزی، وزن یکسان میدهد و به عبارتی بین پیکسل ها میانگین با وزن یکسان میگیرد. این باعث میشود که فیلتر بین لبه ها و دیگر بخش های تصویر تفاوتی قائل نمیشود در نتیجه لبه ها از بین میروند. همچنین بین اختلاف رنگ پیکسل ها با پیکسل مرکزی و یا فاصله پیکسل ها از پیکسل مرکزی تفاوتی قائل نمیشود به عبارتی بهتر میشد اگر این فیلتر به پیکسل هایی که به پیکسل مرکزی نزدیکتر هستند و یا رنگ آنها به این پیکسل نزدیکتر است وزن بیشتری بدهد. در نتیجه تصویر خروجی smooth box filter مناسبی برای تصویر اصلی نمی باشد.

۲-۱-۲- بخش دوم

بله برخی فیلتر ها مثل فیلتر median را اگر روی تصویر با نویز salt&peper چندین بار اعمال کنیم باعث میشود نویز بیشتری در هر مرحله حذف شود. همچنین در مورد box filter با اعمال چند باره آن تصویر smooth تر میشود و در نتیجه برای برخی کاربرد ها مثلا حذف مقادیر پرت با چندین بار میانگین گیری از مقادیر اطراف مناسب است. مثلا برای کاهش نویز در تصاویر میتوان با اعمال چندین بار box filter روی تصویر، این نویز ها را کاهش داد و تصویر را smooth تر کرد.

۳-۱-۲- بخش سوم

در این بخش ما box filter ۳ در ۳ را روی تصویر Elaine اعمال کردیم. به این صورت که ابتدا padding به اطراف تصویر اضافه کردیم سپس متده apply_filter را call میکنیم و پارامتر های آن یعنی تصویر و فیلتر را به عنوان ورودی میدهیم. نکته: padding به اندازه نصف فیلتر به هر سمت تصویر اضافه میشود یعنی به اندازه $\text{floor}(\text{size(filter,1)}/2)$ در متده apply_filter ابتدا یک متغیر برای تصویر خروجی در نظر میگیریم. سپس بلاک های ۳ در ۳ از تصویر padding دار بر میداریم و box filter را در آن بلاک ضرب داخلی میکنیم و پس از آن مقادیر بدست آمده را با هم جمع میکنیم و در پیکسل متناظر در تصویر خروجی قرار میدهیم.

همچنین هر بار اعمال box filter را به تصویر اعمال میکنیم خروجی را در متغیری که filter را روی آن اعمال میکنیم ذخیره میکنیم، با اینکار فیلتر هر بار روی خروجی مرحله قبل اعمال میشود. خروجی در تصویر ۱ تا ۵ نمایش داده شده است.

۴-۱-۲- بخش چهارم

در این بخش ما box filter با سایز های ۳ و ۵ و ۷ و ۹ و ۱۱ را روی تصویر Elaine اعمال کردیم. به این صورت که ابتدا به اندازه size(filter,1)-1 یعنی در هر طرف به اندازه نصف اندازه فیلتر به تصویر padding اضافه میکنیم.

سپس از متدها apply_filter استفاده میکنیم و فیلتر را روی تصویر اعمال میکنیم و پس از آن تصویر را ذخیره میکنیم و نمایش میدهیم.

۵-۱-۲-بخش پنجم

با توجه به تصاویر ۶ تا ۱۱ سایز ۳ در ۳ tradeoff بین blur کردن تصویر و حذف نویز را دارد، زیرا تصویر به اندازه ای که جزئیات تصویر حفظ شود blur شده و از طرف دیگر نویز تصویر نیز کاهش پیدا کرده است. البته این بستگی به تصویر نیز دارد. اگر تصویری جزئیات زیاد و نویز کمتری دارد بهتر است از box filter با سایز کوچک استفاده شود و اگر تصویر جزئیات کم و نویز بیشتری دارد بهتر است از box filter با سایز بزرگتر استفاده شود، به عبارتی هر جا برای ما جزئیات مهم باشد فیلتر با سایز کوچکتر و هر جا حذف نویز برای ما مهم باشد فیلتر بزرگتر لازم است.

۶-۱-۲-بخش ششم

در این بخش ما باید فیلتر لاپلاسین را روی تصویر اعمال کنیم. برای اینکار ابتدا فیلتر لاپلاسین را به صورت $[0 \cdot 1 \cdot 0; 1 \cdot 0 \cdot 0]$ تعریف میکنیم تا همزمان خروجی تصویر با اعمال فیلتر لاپلاسین محاسبه شود و روی تصویر اصلی اعمال شود. در صورت سوال گفته شده که این فیلتر را روی تصاویر خروجی بخش سوم، که box filter روی آنها اعمال شده، چندین بار اعمال کنیم. برای این کار تصاویری که box filter روی آنها اعمال شده، که این تصاویر ۵ تا است و از ۱ بار تا ۵ بار box filter روی آنها اعمال شده، را میخوانیم سپس بر روی هر تصویر ۵ بار فیلتر لاپلاسین را اعمال میکنیم که برای اعمال فیلتر لاپلاسین از همان تابع از قبل نوشته شده `apply_filter` استفاده میکنیم.

Median Filter -۲-۲

۱-۲-۲-بخش اول

در این بخش ما باید ابتدا به تصویر با ضرایب تراکم مختلف نویز salt & peper را اضافه کنیم سپس سایز های مختلف فیلتر median را روی تصویر اعمال کنیم و خطای MSE را برای آن ها محاسبه کنیم. برای همین ابتدا ما نویز salt & peper را با ضریب تراکم $0.05, 0.1, 0.2, 0.4$ به تصویر اصلی اضافه میکنیم. سپس بر روی هر یک از تصاویر نویزی فیلتر median با سایز پنجره ۳، ۵، ۷، ۹ و ۱۱ را اعمال میکنیم. به این صورت که روی تمام پیکسل های تصویر دارای padding حرکت میکنیم و برای هر بلاک از تصویر داده‌ی میانی را بدست می‌آوریم و در پیکسل متناظر در تصویر بهبود یافته قرار میدهیم. پس از آن خطای MSE را برای هر یک از تصاویر بدست آمده از اعمال فیلتر median با تصویر اصلی را محاسبه میکنیم.

۲-۲-۲-بخش دوم

در این بخش ما نویز gaussian را با ضرایب تراکم $0.05, 0.1, 0.05$ به تصویر اصلی اعمال میکنیم سپس فیلتر median و box را با سایز پنجره ۳، ۵، ۷، ۹ و ۱۱ روی تصویر نویزی اعمال میکنیم و خطای MSE خروجی فیلترها را با تصویر اصلی میسنجهیم. برای اعمال فیلتر median از کد بخش قبل استفاده کردیم و برای اعمال فیلتر box از متدهای `apply_filter` که در تمرین box filter پیاده شده است استفاده کردیم.

Edge Detection - ۳-۲

۱-۳-۲- بخش اول

در این بخش ما ۳ فیلتر زیر را برای پیدا کردن لبه ها در جهت \times روی تصویر Elaine اعمال کردیم:

```
filter_a = [1 0 -1]./2;
filter_b = [1 0 -1; 1 0 -1; 1 0 -1]./6;
filter_c = [1 0 -1; 2 0 -2; 1 0 -1]./8;
```

این ۳ فیلتر را روی تصویر اعمال کردیم و خروجی را در بخش بررسی نتایج نمایش داده ایم و بررسی کردیم.

۳-۳-۲- بخش دوم

در این بخش دو فیلتر روبرت به صورت زیر را روی تصویر Elaine اعمال کردیم:

```
robert1 = [1 0; 0 -1];
robert2 = [0 1; -1 0];
```

این ۲ فیلتر را روی تصویر اعمال کردیم و خروجی را در بخش بررسی نتایج نمایش داده ایم و بررسی کردیم.

Unsharp Masking - ۴-۲

توجه: لپ تاپ بنده در حین کار با متلب دچار مشکل شد و من مجبور شدم با لپ تاپ دیگری ادامه تمرین را با پایتون در محیط colab انجام دهم.

در این بخش ما ابتدا یک متد unsharp_mask_filter نوشته‌یم که تصویر اصلی و تصویر smooth شده با فیلتر گوسی و ضریب آلفا را به عنوان ورودی می‌گیرد و سپس خروجی این فیلتر را که با فرمول زیر بدست می‌آید را به عنوان خروجیتابع در نظر می‌گیریم. فرمول به صورت زیر می‌باشد:

فرمول (۱)

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

ما سایز های ۳ و ۵ و ۷ و ۹ و ۱۱ کرنل را برای فیلتر گوسی در نظر می‌گیریم و فیلتر گوسی را روی تصویر اصلی اعمال می‌کنیم و نتایج را در یک آرایه ذخیره می‌کنیم. سپس برای تمام مقادیر آلفا فیلتر unsharp mask را روی تمام تصاویر smooth شده به وسیله فیلتر گوسی اعمال می‌کنیم و نتایج را در یک آرایه ذخیره می‌کنیم.

پس از این نیز برای هر سایز کرنل فیلتر گوسی، تصویر اصلی و تصویر smooth شده و تصویر فیلتر شده با unsharp mask با آلفا های مختلف را نمایش داده ایم که در بخش بررسی نتایج آنها را مشاهده می‌کنیم.

۳- بررسی نتایج

Box Filter - ۱-۳

۱-۱-۳- بخش اول

این بخش دارای خروجی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۲-۱-۳- بخش دوم

این بخش دارای خروجی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۳-۱-۳- بخش سوم

ما ۵ بار روی تصویر اعمال کردیم و خروجی هر بار در تصاویر زیر نمایش داده شده و مشاهده میکنید که در هر مرحله تصویر smooth تر میشود.



تصویر ۱ - خروجی تصویر Eliane با اعمال ۱ بار box filter



تصویر ۲ - خروجی تصویر Eliane با اعمال ۲ بار box filter



تصویر ۳ - خروجی تصویر Eliane با اعمال ۳ بار box filter



تصویر ۴- خروجی تصویر Eliane با اعمال ۴ بار box filter



تصویر ۵- خروجی تصویر Eliane با اعمال ۵ بار box filter

۴-۱-۳- بخش چهارم

همانطور که در تصاویر زیر میبینیم هر چه اندازه box filter بزرگتر باشد، برای محاسبه مقدار پیکسل مرکزی، پیکسل های بیشتری از اطراف آن در نظر گرفته میشود در نتیجه تصویر blur تر میشود و به دنبال آن نویز بیشتر کاهش می یابد.



تصویر ۶- خروجی تصویر Eliane با اعمال ۳ در ۳ box filter



تصویر ۷- خروجی تصویر Eliane با اعمال ۵ در ۵ box filter



تصویر ۸- خروجی تصویر Eliane با اعمال ۷ در ۷ box filter



تصویر ۹- خروجی تصویر Eliane با اعمال ۹ در ۹ box filter



تصویر ۱۰ - خروجی تصویر Eliane با اعمال ۱۱ در ۱۱ box filter



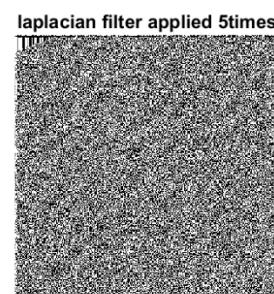
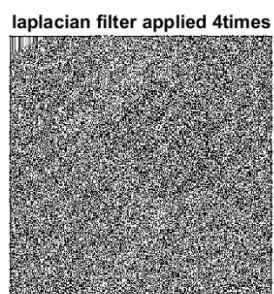
تصویر ۱۱ - خروجی تصویر Eliane با اعمال ۱۳ در ۱۳ box filter

۱-۵-بخش پنجم

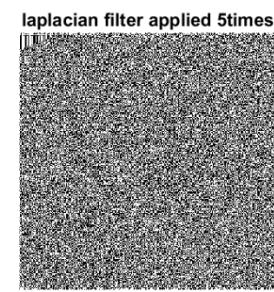
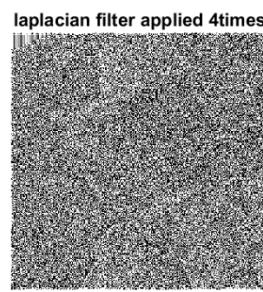
این بخش دارای خروجی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۱-۶-بخش ششم

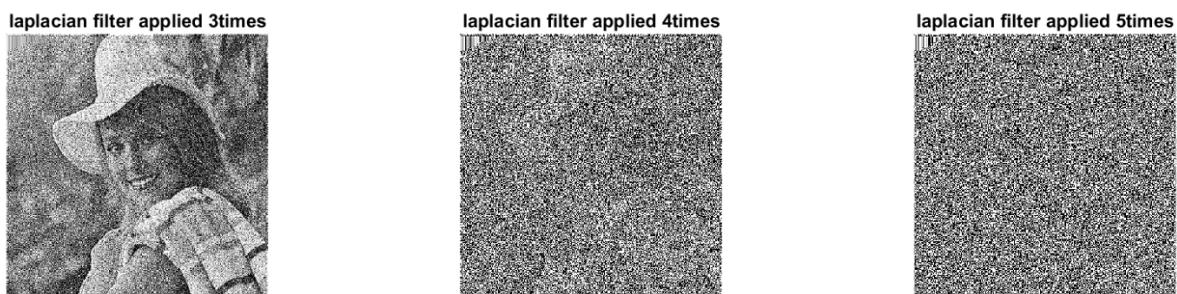
همانطور که میبینید به ازای هر تصویر که روی آنها اعمال شده ما چندین بار فیلتر لایپلاسین را اعمال کرده ایم و با دقت در تصاویر مشاهده میکنید که با یک بار اعمال لایپلاسین تصویر از حالت blur خارج شده و خروجی مناسب نمایش داده میشود اما هر چه لایپلاسین را بیشتر روی تصویر اعمال کنیم نویز تصویر بیشتر میشود به گونه ای که در حالتی که در ۴ یا ۵ بار box filter روی تصویر اعمال شده، عملا خروجی نویز های salt&peper است. در حالتی که تعداد دفعات بیشتری فیلتر روی تصویر اعمال شده است مثلا ۴ یا ۵ بار، برای اینکه حالت blur تصویر از بین برود، بهتر است به جای یکبار اعمال لایپلاسین، دو بار آن را روی تصویر اعمال کنیم اما بیشتر از آن تصویر نویزی میشود.



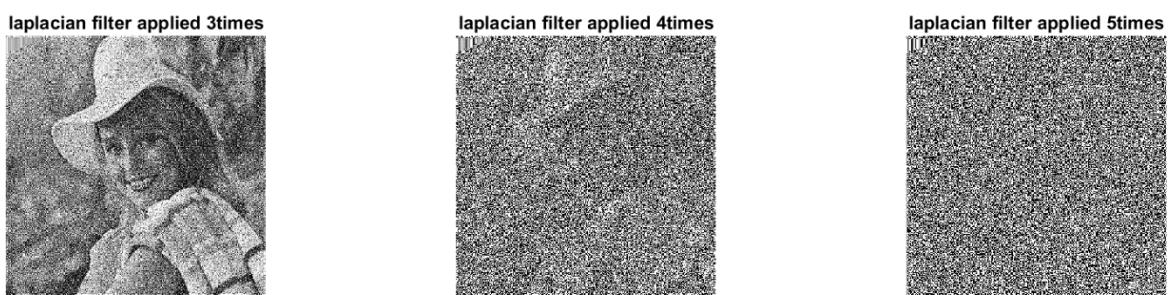
تصویر ۱۲- اعمال فیلتر لاپلاسین روی تصویری که ۱ بار روی آن **box filter** اعمال شده



تصویر ۱۳- اعمال فیلتر لاپلاسین روی تصویری که ۲ بار روی آن **box filter** اعمال شده



تصویر ۱۴- اعمال فیلتر لاپلاسین روی تصویری که ۳ بار روی آن box filter اعمال شده



تصویر ۱۵- اعمال فیلتر لاپلاسین روی تصویری که ۴ بار روی آن box filter اعمال شده

box filter applied 5 times



laplacian filter applied 1times



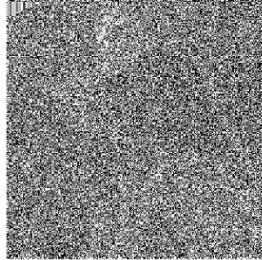
laplacian filter applied 2times



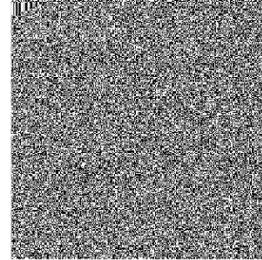
laplacian filter applied 3times



laplacian filter applied 4times



laplacian filter applied 5times



تصویر ۱۶- اعمال فیلتر لاپلاسین روی تصویری که ۵ بار روی آن box filter اعمال شده

من برای حل مشکل نویزی شدن سریع خروجی لاپلاسین از نرمال سازی استفاده کردم و تصویر خروجی فیلتر را نرمال کردم و به بازه $[0, 255]$ تبدیل کردم به جای اینکه مستقیم خروجی را به تابع uint8 پاس بدهم اما نتیجه ای که داد بدتر از حالتی بود که مستقیماً تصویر را به تابع uint8 پاس میدادم و تصاویر نسبت به حالت اصلی خود تیره تر نمایش داده است به خصوص در تصویر ۱۷ و ۱۸ اما این نرمال سازی در کاهش نویز salt & pepper تاثیر داشته و این نویز تا حد خیلی خوبی در تصاویر کاهش پیدا کرده است. خروجی در حالتی که نرمال سازی انجام شده است به صورت زیر می باشد:

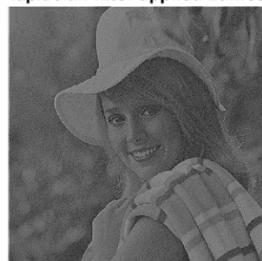
box filter applied 1 times



laplacian filter applied 1times



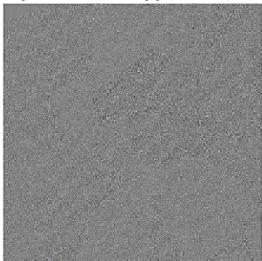
laplacian filter applied 2times



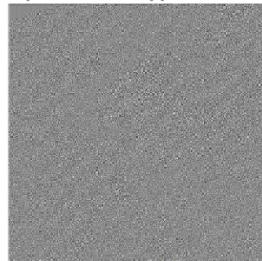
laplacian filter applied 3times



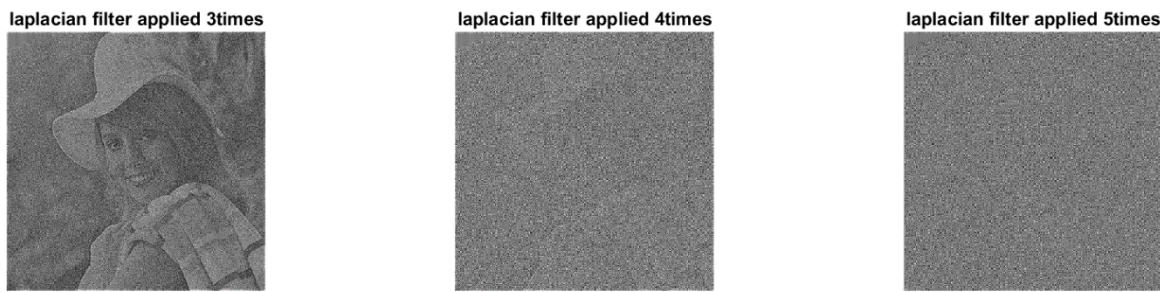
laplacian filter applied 4times



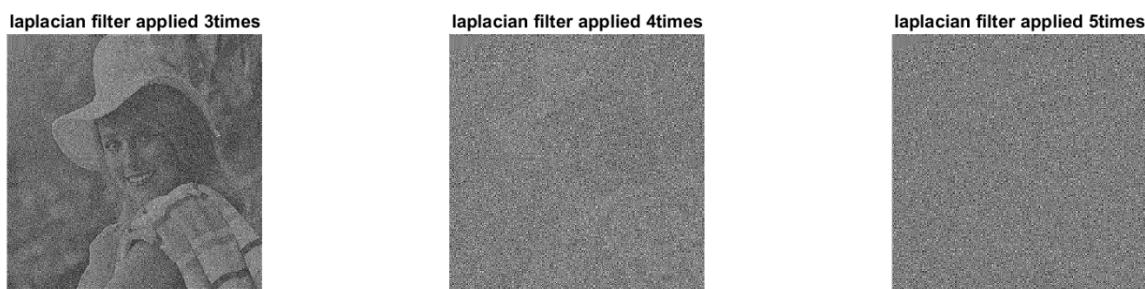
laplacian filter applied 5times



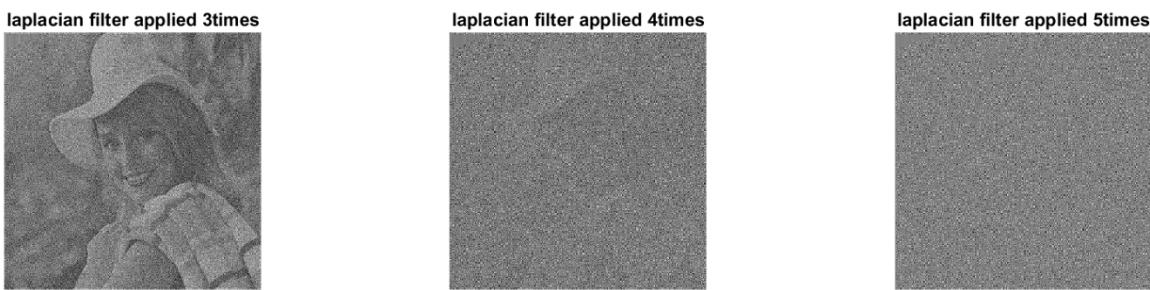
تصویر ۱۷- اعمال فیلتر لاپلاسین روی تصویری که ۱ بار روی آن box filter اعمال شده به همراه نرمال سازی خروجی



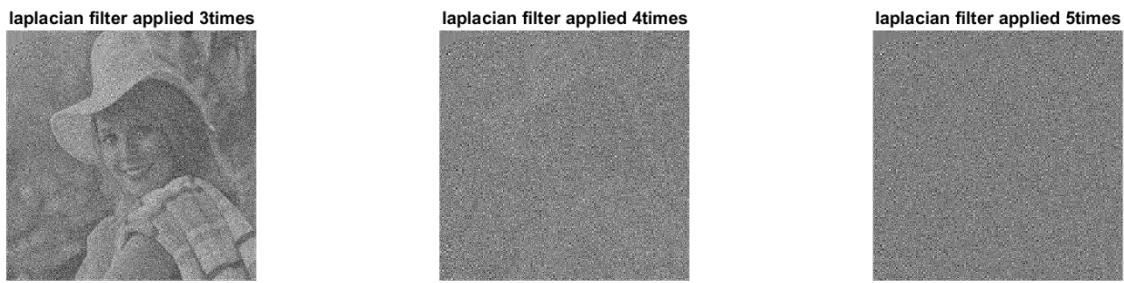
تصویر ۱۸- اعمال فیلتر لاپلاسین روی تصویری که ۲ بار روی آن **box filter** اعمال شده به همراه نرمال سازی خروجی



تصویر ۱۹- اعمال فیلتر لاپلاسین روی تصویری که ۳ بار روی آن **box filter** اعمال شده به همراه نرمال سازی خروجی



تصویر ۲۰- اعمال فیلتر لاپلاسین روی تصویری که ۴ بار روی آن box filter اعمال شده به همراه نرمال سازی خروجی



تصویر ۲۱- اعمال فیلتر لاپلاسین روی تصویری که ۵ بار روی آن box filter اعمال شده به همراه نرمال سازی خروجی

Median Filter - ۲-۳

۱-۲-۳- بخش اول

در جدول زیر ما خطای MSE را برای هر یک از حالات نویز و فیلتر Median نمایش داده ایم:

جدول ۱- خطای MSE برای نویز salt & peper با ضریب تراکم متفاوت و فیلتر Median با سایز های مختلف

ضریب	۰.۱	۰.۵	۳۶.۲۸۳۴	۴۶.۲۳۵۵	۵۵.۹۷۵۹	۷۱.۰۸۵۳	۸۸.۵۳۹۱	۱۱۰.۴۵۲۴	پنجره ۳ در ۳	پنجره ۵ در ۵	پنجره ۷ در ۷	پنجره ۹ در ۹	پنجره ۱۱ در ۱۱
ضریب	۰.۱	۰.۵	۳۶.۲۸۳۴	۴۶.۲۳۵۵	۵۵.۹۷۵۹	۷۱.۰۸۵۳	۸۸.۵۳۹۱	۱۱۰.۴۵۲۴	پنجره ۳ در ۳	پنجره ۵ در ۵	پنجره ۷ در ۷	پنجره ۹ در ۹	پنجره ۱۱ در ۱۱

۱۹۰.۷۱۲۷	۱۴۷.۹۸۶۷	۱۱۱.۷۸۶	۸۲.۳۵۷۹	۱۰۳.۱۹۱۹	ضریب ۰.۲
۳۱۵.۱۳۴	۲۶۱.۱۳۸۷	۲۱۴.۶۷۳۴	۱۸۳.۳۰۷۱	۸۲۲.۹۷۵۹	ضریب ۰.۴

همانطور که در جدول بالا مشاهده میکنید برای هر ضریب تراکم، سایز فیلتر Median مناسب متفاوت است. مثلا برای ضریب تراکم ۰.۰۵ و ۰.۱ سایز فیلتر ۳ در ۳ کمترین خطای را دارد و مناسب است و برای ضریب تراکم ۰.۲ و ۰.۴ سایز فیلتر ۵ در ۵ مناسب است. به عبارتی هر چه میزان نویز salt & peper بیشتر باشد، بهتر است که سایز فیلتر Median بزرگتر باشد اما نه خیلی بزرگ که از کیفیت تصاویر بکاهد.

در تصاویر زیر نیز نتایج بدست آمده از اعمال فیلتر Median روی تصاویر نویزی را مشاهده میکنید:

noisy image with density 0.05



median size:3 immse:37.3547



median size:5 immse:46.1634



median size:7 immse:54.724



median size:9 immse:69.7895



median size:11 immse:87.3298



تصویر ۲۲- تصویر نویزی با ضریب تراکم ۰.۰۵ و اعمال فیلتر های median با سایز های ۳ و ۵ و ۷ و ۹ و ۱۱

noisy image with density 0.1



median size:3 immse:45.3759



median size:5 immse:52.1357



median size:7 immse:64.0969



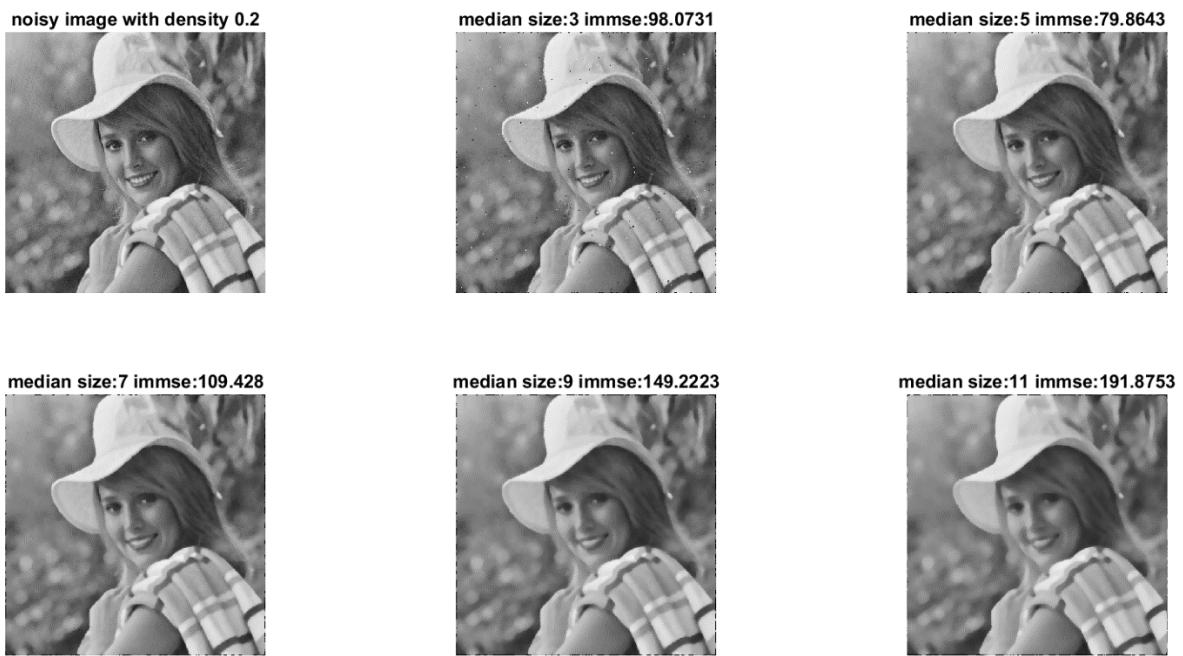
median size:9 immse:83.1314



median size:11 immse:108.6003



تصویر ۲۳- تصویر نویزی با ضریب تراکم ۰.۱ و اعمال فیلتر های median با سایز های ۳ و ۵ و ۷ و ۹ و ۱۱



تصویر ۲۴- تصویر نویزی با ضریب تراکم ۰.۲ و اعمال فیلتر های median با سایز های ۳ و ۵ و ۷ و ۹ و ۱۱



تصویر ۲۵- تصویر نویزی با ضریب تراکم ۰.۴ و اعمال فیلتر های median با سایز های ۳ و ۵ و ۷ و ۹ و ۱۱

۲-۲-۳- بخش دوم

در دو جدول زیر مقدار خطای MSE برای اعمال فیلتر های median و box روى تصویر نویزی شده با نویز gaussian گزارش شده است:

جدول ۲- خطای MSE برای نویز gaussian با ضریب تراکم مختلف و فیلتر Median با سایز های مختلف

Median filter	

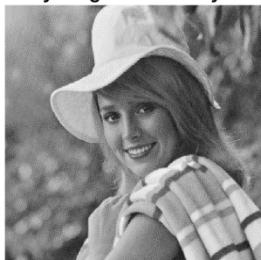
۱۱ در ۱۱	۹ در ۹	۷ در ۷	۵ در ۵	۳ در ۳	
۱۳۲.۹۳۹۸	۱۱۲.۳۱۲۵	۹۸.۶۸۹۸	۱۰۲.۲۷۷۴	۱۵۳.۶۹۱۴	ضریب ۰.۰۱
۲۷۵.۹۸۱۹	۲۵۸.۰۳۸۹	۲۴۶.۳۷۸۵	۲۵۳.۱۷۲	۳۰۷.۰۳۲۲	ضریب ۰.۰۵
۷۴۲.۶۹۶	۷۲۷.۹۹۱۷	۷۱۹.۳۲۳	۷۲۷.۱۹۳۷	۷۷۹.۶۳۲۹	ضریب ۰.۱

جدول ۳- خطای MSE برای نویز gaussian با ضریب تراکم متفاوت و فیلتر box با سایز های مختلف

Box filter					
۱۱ در ۱۱	۹ در ۹	۷ در ۷	۵ در ۵	۳ در ۳	
۲۱۱.۶۷۰۶	۱۷۲.۷۲۲	۱۳۸.۰۷۴۸	۱۱۷.۰۰۱۸	۱۲۸.۶۷۲۱	ضریب ۰.۰۱
۳۴۱.۱۲۸۳	۳۰۶.۱۲۰۵	۲۷۵.۳۴۶۳	۲۵۷.۷۹۸۹	۲۷۳.۳۲۱۹	ضریب ۰.۰۵
۷۷۲.۷۳۶۱	۷۴۳.۳۲۳۱	۷۱۸.۰۲۱۹	۷۰۵.۸۰۷۹	۷۲۶.۵۶۲۹	ضریب ۰.۱

به طور کلی برای حذف نویز gaussian این دو فیلتر مناسب نمی باشند و دقت را به اندازه مناسبی افزایش نمی دهند و هر چه ضریب تراکم نویز افزایش می یابد خطای که از بهبود تصاویر با دو روش فیلتر median و box بدست می آید افزایش می یابد و تصویر بهبود یافته با تصویر اصلی اختلاف زیادی دارد. اما با مقایسه مقادیر MSE به نظر می آید که median با سایز پنجره ۷ تا حدودی عملکرد بهتری دارد و میتواند به دقت مناسبی برسد به خصوص برای ضرایب تراکم ۰.۰۱ و ۰.۰۵.

noisy image with density 0.01



median size:3 immse:153.6914



median size:5 immse:102.2774



median size:7 immse:98.6898



median size:9 immse:112.3125



median size:11 immse:132.9398



تصویر ۲۶- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۰۱ با فیلتر median با سایز پنجره متفاوت

noisy image with density 0.01



box size:3 immse:128.6721



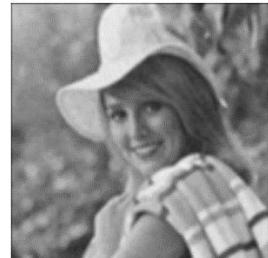
box size:5 immse:117.0018



box size:7 immse:138.0748



box size:9 immse:172.722



box size:11 immse:211.6706



تصویر ۲۷- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۰۱ با فیلتر box با سایز پنجره متفاوت

noisy image with density 0.05



median size:3 immse:307.0322



median size:5 immse:253.172



median size:7 immse:246.3785



median size:9 immse:258.0389



median size:11 immse:275.9819



تصویر ۲۸- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۰۵ با فیلتر median با سایز پنجره متفاوت

noisy image with density 0.05



box size:3 immse:273.3219



box size:5 immse:257.7989



box size:7 immse:275.3463



box size:9 immse:306.1205



box size:11 immse:341.1283



تصویر ۲۹- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۰۵ با فیلتر box با سایز پنجره متفاوت

noisy image with density 0.1



median size:3 immse:779.6329



median size:5 immse:727.1937



median size:7 immse:719.323



median size:9 immse:727.9917



median size:11 immse:742.696

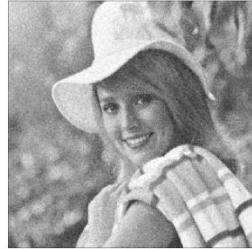


تصویر ۳۰- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۱ با فیلتر median با سایز پنجره متفاوت

noisy image with density 0.1



box size:3 immse:726.5629



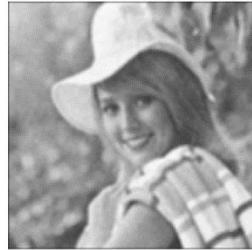
box size:5 immse:705.8079



box size:7 immse:718.0219



box size:9 immse:743.3231



box size:11 immse:772.7361



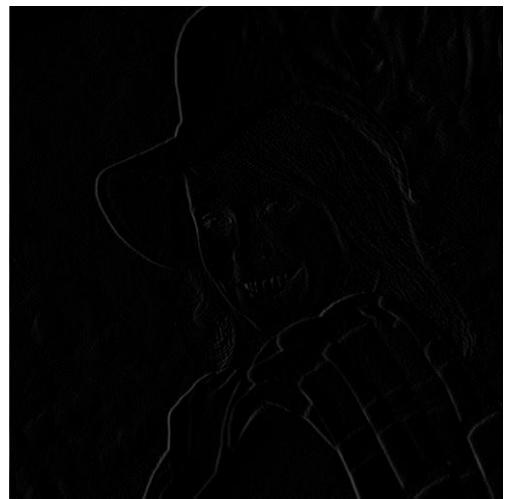
تصویر ۳۱- بهبود تصویر با نویز gaussian و ضریب تراکم ۰.۱ با فیلتر box با سایز پنجره متفاوت

Edge Detection - ۳-۳

۱-۳-۳- بخش اول

۳ تصویر زیر خروجی اعمال فیلتر برای تشخیص لبه های عمودی تصویر می باشد همانطور که مشاهده میکنیم خروجی این ۳ فیلتر برای تشخیص لبه های تصویر مشابه هم می باشد، اما این ۳ فیلتر تفاوت هایی با هم دارند:
فیلتر تصویر ۳۲: این فیلتر تنها دو طرف پیکسل مرکزی را در نظر میگیرد و از اختلاف پیکسل های قطری صرف نظر میکند و در نتیجه لبه های بیشتری تشخیص میدهد که ممکن است برخی از این لبه ها غیر واقعی باشند.
فیلتر تصویر ۳۳: این فیلتر علاوه بر بدست آوردن اختلاف دو پیکسل دو طرف پیکسل مرکزی، اختلاف دو به دوی پیکسل های قطری را نیز بدست می آورد، در نتیجه نسبت به قبلی لبه های کمتری را بدست می آورد و نکته دیگری که وجود دارد این است که به تمام پیکسل ها وزن یکسانی میدهد.

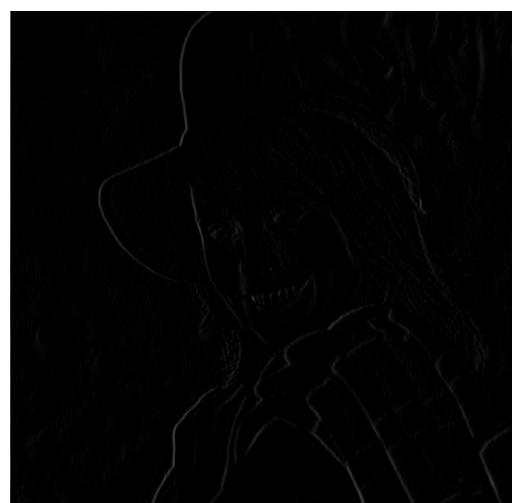
فیلتر تصویر ۳۴: این فیلتر علاوه بر بدست آوردن اختلاف دو طرف پیکسل مرکزی، اختلاف دو به دوی پیکسل های قطری را نیز بدست می آورد اما تفاوت آن با فیلتر تصویر ۳۳ این است که این فیلتر به دو پیکسل دو طرف پیکسل مرکزی وزن بیشتری نسبت به پیکسل های قطری میدهد چون پیکسل های دو طرف به پیکسل مرکزی نزدیکترند.
بنابراین فیلتر تصویر ۳۴ بهتر از تصویر ۳۳ عمل میکند زیرا به دو پیکسل دو طرف پیکسل مرکزی وزن بیشتری میدهد و فیلتر تصویر ۳۳ بهتر از تصویر ۳۲ می باشد چون علاوه بر دو پیکسل دو طرف پیکسل های قطری را نیز در نظر میگیرد.



تصویر ۳۲- اعمال فیلتر $[1\ 0\ -1]/2$ برای تشخیص لبه های تصویر در جهت x



تصویر ۳۳- اعمال فیلتر $[1\ 0\ -1; 1\ 0\ -1; 1\ 0\ -1]/6$ برای تشخیص لبه های تصویر در جهت x



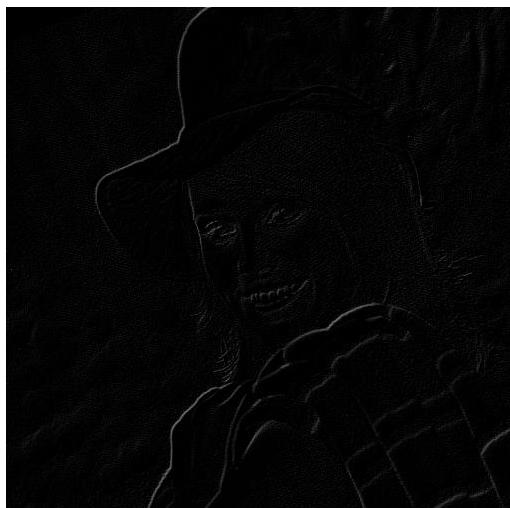
تصویر ۳۴- اعمال فیلتر $[1\ 0\ -1; 2\ 0\ -2; 1\ 0\ -1]/8$ برای تشخیص لبه های تصویر در جهت x

۲-۳-۳-بخش دوم

دو تصویر زیر خروجی اعمال فیلتر روبرت روی تصویر Elaine می باشد:



تصویر ۳۵- اعمال فیلتر [۰; ۱ ۰;-۱] روی تصویر Elaine



تصویر ۳۶- اعمال فیلتر [۰; ۱ ۰;-۱] روی تصویر Elaine

فیلتر تصویر ۳۵ در حقیقت لبه های تصویر در راستای قطر اصلی را بdst می آورد، همچنین فیلتر تصویر ۳۶ لبه های تصویر در راستای فطر فرعی را بdst می آورد.

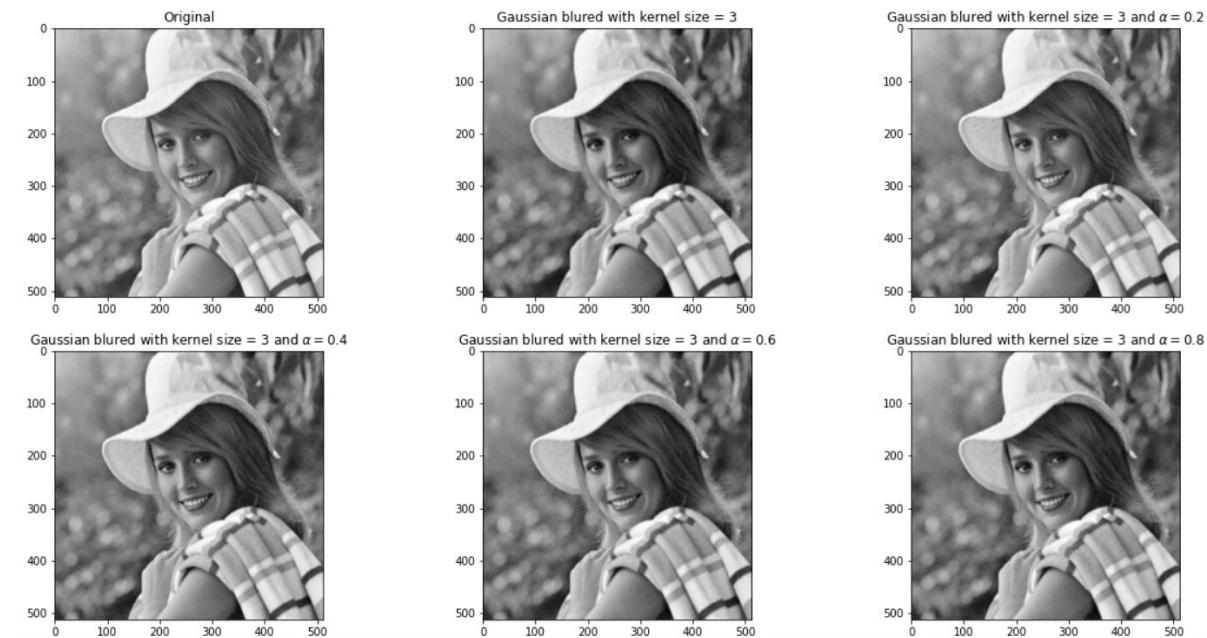
همچنین فیلتر های این بخش نسبت به بخش قبل لبه های بیشتری را تشخیص میدهد زیرا در بخش قبل فیلتر تنها لبه های افقی را تشخیص میدادند اما در اینجا فیلتر Robert علاوه بر لبه های قطری بخش کمی از لبه های افقی و عمودی را نیز تشخیص میدهند.

4-3 Unsharp Masking

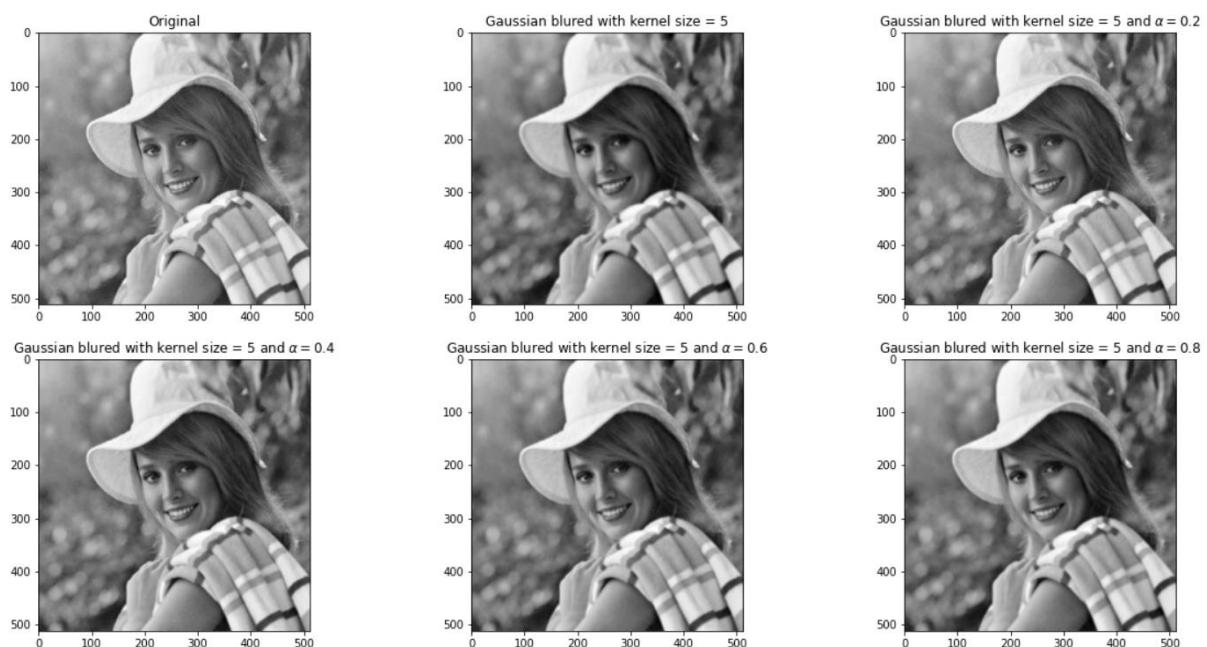
همانطور که میبینیم سایز کردن گوسی هر چه بزرگتر میشود، تصویر blur تر میشود. با بررسی نتایج فیلتر unsharp mask متوجه میشویم در آلفا با مقادیر ۰.۲ و تا حدودی ۰.۴ خروجی تا حد خوبی به تصویر اصلی نزدیک است و میزان تاری تصویر کمتر است، اما هرچه مقدار آلفا بیشتر میشود قدرت sharp کردن تصویر و وضوح تصویر کاهش میابد.
به عبارتی در فرمول (۱) مشاهده میکنیم که آلفا، ضریب تصویر smooth شده با فیلتر گوسی و یک منهای آلفا، ضریب تصویر اصلی می باشد. بنابراین هرچه مقدار آلفا کمتر باشد وزن و تاثیر تصویر اصلی در خروجی فیلتر unsharp mask بیشتر است و در مقابل تاثیر تصویر smooth شده کمتر میشود، لذا تصویر خروجی به تصویر اصلی نزدیکتر است و تصویر به نسبت تصویر

گویی sharp تر است و هرچه مقدار آلفا بیشتر باشد تاثیر تصویر اصلی کمتر و تاثیر تصویر smooth شده بیشتر میشود بنابراین خروجی unsharp mask نرم تر و به تصویر گویی نزدیکتر است.

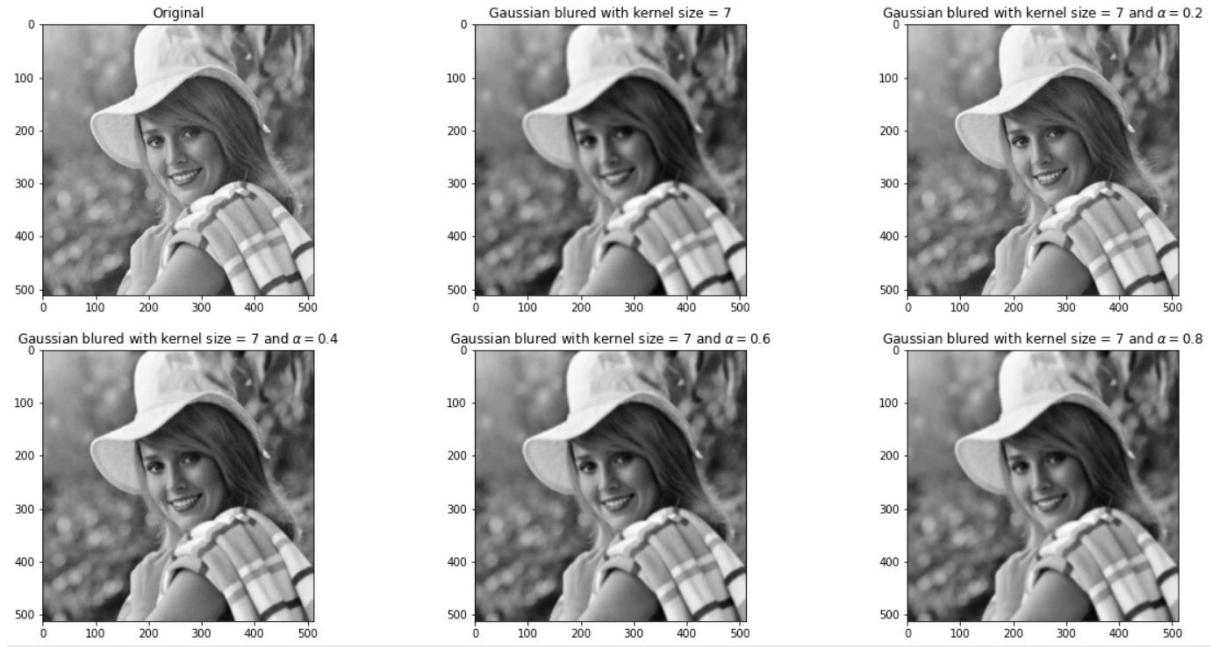
بنابراین براساس اینکه میخواهیم تصویر smooth تر باشد یا sharp تر باشد یا انتخاب شود و اگر میخواهیم تصویر blur تر باشد مقدار آلفا باید بزرگتر از ۰.۵ باشد و در مقدار ۰.۵ وزن تصویر اصلی و وزن تصویر blur شده یکسان میباشد.



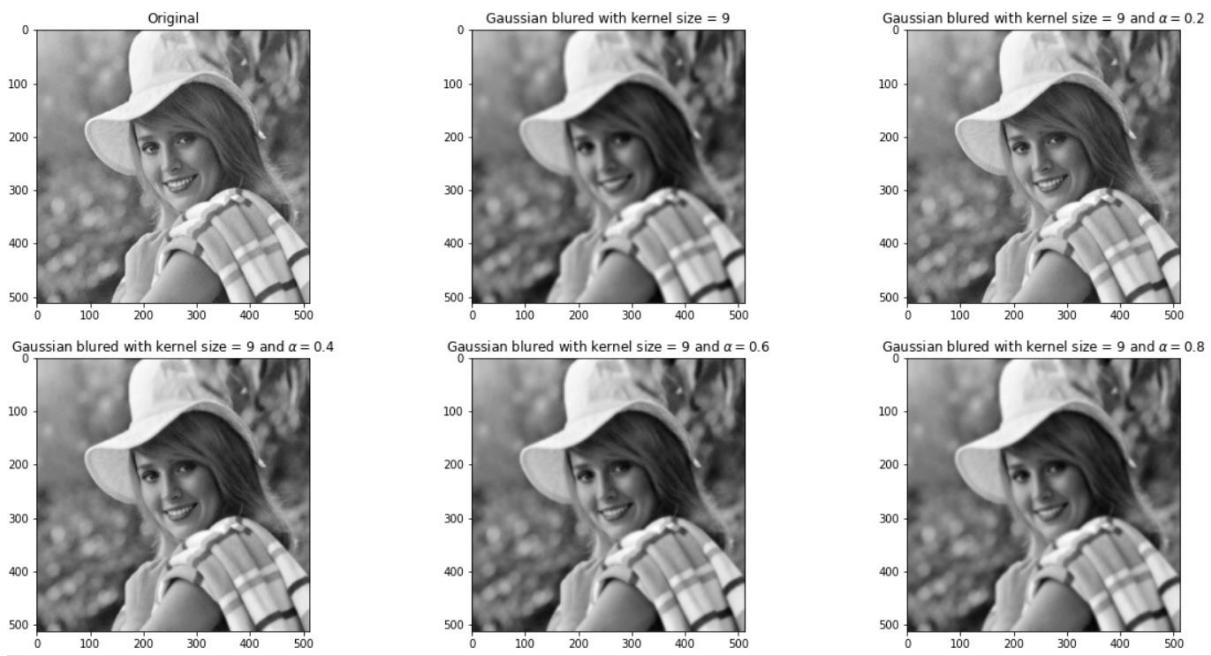
تصویر ۳۷- اعمال فیلتر unsharp mask با آلفا های مختلف روی تصویر smooth شده با فیلتر گویی با $kernel_size = 3$



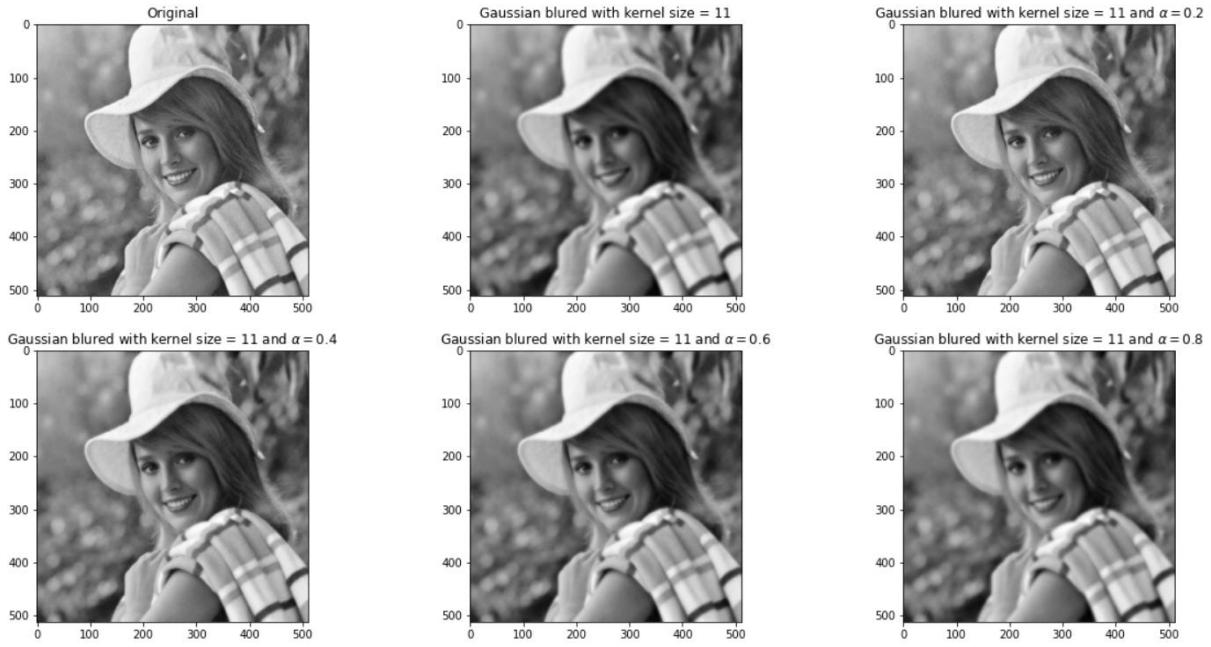
تصویر ۳۸- اعمال فیلتر unsharp mask با آلفا های مختلف روی تصویر smooth شده با فیلتر گویی با $kernel_size = 5$



تصویر ۳۹- اعمال فیلتر unsharp mask با آلفا های مختلف روی تصویر smooth شده با فیلتر گوسی با $kernel_size = 7$



تصویر ۴۰- اعمال فیلتر unsharp mask با آلفا های مختلف روی تصویر smooth شده با فیلتر گوسی با $kernel_size = 9$



تصویر ۴۱- اعمال فیلتر unsharp mask با آلفا های مختلف روی تصویر smooth شده با فیلتر گوسی با $\text{kernel_size} = 11$

۴- پیوست

Box Filter - ۱-۴

۱-۱-۴- بخش اول

این بخش دارای کد پیاده سازی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۲-۱-۴- بخش دوم

این بخش دارای کد پیاده سازی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۳-۱-۴- بخش سوم

کد فایل `:apply_filter`

```
function filtered_img = apply_filter(img, filter, window_size)
padding = floor(window_size/2);
padding_img = zeros(size(img)+(padding*2));
padding_img((padding+1):size(padding_img,1)-
padding,(padding+1):size(padding_img,2)-padding) = img;
filtered_img = zeros(size(img));
for i=1:size(img,1)
    for j=1:size(img,2)
        img_block = padding_img(i:i+(padding*2),j:j+(padding*2));
        filter_apply = img_block.*filter;
        pixel_value = sum(filter_apply, 'all');
        filtered_img(i,j) = pixel_value;
    end
end
```

:main کد فایل

```
img = imread("E:\Dars\Masters\digital image processing\Homeworks\Images\3\Elaine.bmp");
boxfilter = ones(3)/9;
fig = figure;
subplot(2,3,1); imshow(img); title("Original Image");
filtered_img = img;
for i=1:5
    filtered_img = apply_filter(filtered_img,boxfilter,3);
    imwrite(uint8(filtered_img),strcat("box_filter_",int2str(i),"times.bmp"));
    subplot(2,3,i+1); imshow(uint8(filtered_img)); title(strcat("box filter applied ",int2str(i),"times"));
end
```

۲-۱-۴-بخش چهارم

```
img = imread("E:\Dars\Masters\digital image processing\Homeworks\Images\3\Elaine.bmp");
fig = figure;
subplot(2,3,1); imshow(img); title("Original Image");
padding_img = zeros(size(img)+2);
for i=3:2:11
    idx = floor(i/2)+1;
    boxfilter = ones(i)/(i.^2);
    filtered_img = apply_filter(img, boxfilter, i);
    imwrite(uint8(filtered_img),strcat("box_filter_",int2str(i),"_size.bmp"));
    subplot(2,3,idx); imshow(uint8(filtered_img)); title(strcat("box filter applied ",int2str(i)," size"));
end
```

۵-۱-۴-بخش پنجم

این بخش دارای کد پیاده سازی نمی باشد و پاسخ سوال در بخش توضیحات فنی آورده شده است.

۶-۱-۴-بخش ششم

:main کد فایل

```
laplacianfilter = [0 -1 0; -1 5 -1;0 -1 0];
for i=1:5
    img_path = strcat("E:\Dars\Masters\digital image processing\Homeworks\Filters\1\box_filter_",int2str(i),"times.bmp");
    img = imread(img_path);
    fig = figure;
    subplot(2,3,1); imshow(img); title(strcat("box filter applied ",int2str(i),"times"));
    filtered_img = img;
    for j=1:5
        filtered_img = apply_filter(filtered_img,laplacianfilter,3);
        % we want to normalize output before passing to uint8 method
        %fm = filtered_img - min(filtered_img(:));
        %a = 255*(fm/max(fm(:)));
    end
    imwrite(uint8(filtered_img),strcat("box_",int2str(i),"_laplacian_",int2str(j),"_times.bmp"));
    subplot(2,3,j+1); imshow(uint8(filtered_img)); title(strcat("laplacian filter applied ",int2str(j),"times"));
    %filtered_img = a;
```

```
    end
end
```

Median Filter -۲-۴

۱-۲-۴ - بخش اول

:main کد

```
img = imread("E:\Dars\Masters\digital image
processing\Homeworks\Images\3\Elaine.bmp");
noise_density = [0.05; 0.1; 0.2; 0.4];
window_size = [3; 5; 7; 9; 11];
for ns=1:size(noise_density)
    noisy_img = imnoise(img, 'salt & pepper', noise_density(ns));
    imwrite(uint8(noisy_img), strcat("noise_", num2str(noise_density(ns)), ".bmp"));
    fig = figure;
    subplot(2,3,1); imshow(img); title(strcat("noisy image with density
", num2str(noise_density(ns))));

    for mf=1:size(window_size)
        padding = floor(window_size(mf)/2);
        padding_img = zeros(size(noisy_img)+(padding*2));
        padding_img((padding+1):size(padding_img,1)-
padding,(padding+1):size(padding_img,2)-padding) = noisy_img;
        improved_img = zeros(size(noisy_img));
        for i=1:size(noisy_img,1)
            for j=1:size(noisy_img,2)
                img_block = padding_img(i:i+(padding*2),j:j+(padding*2));
                improved_img(i,j) = median(img_block(:));
            end
        end
        improved_img = uint8(improved_img);
        error = immse(img,improved_img);

        imwrite(improved_img, strcat("noise_", num2str(noise_density(ns)), "_median_",
int2str(window_size(mf)), ".bmp"));
        subplot(2,3,mf+1); imshow(improved_img); title(strcat("median
size:", int2str(window_size(mf)), " immse:", num2str(error)));
    end
end
```

۲-۲-۴ - بخش دوم

:main کد

```
img = imread("E:\Dars\Masters\digital image
processing\Homeworks\Images\3\Elaine.bmp");
noise_density = [0.01; 0.05; 0.1];
window_size = [3; 5; 7; 9; 11];
for ns=1:size(noise_density)
    % apply gaussian noise with different densities
    noisy_img = imnoise(img, "gaussian", noise_density(ns));

    imwrite(uint8(noisy_img), strcat("gaussian_noise_", num2str(noise_density(ns)), ".bmp"));
    fig = figure;
    subplot(2,3,1); imshow(img); title(strcat("noisy image with density
", num2str(noise_density(ns))));
```

```

% apply median filter on noisy image
for ws=1:size(window_size)
    padding = floor(window_size(ws)/2);
    padding_img = zeros(size(noisy_img)+(padding*2));
    padding_img((padding+1):size(padding_img,1)-
padding,(padding+1):size(padding_img,2)-padding) = noisy_img;
    improved_img = zeros(size(noisy_img));
    for i=1:size(noisy_img,1)
        for j=1:size(noisy_img,2)
            img_block = padding_img(i:i+(padding*2),j:j+(padding*2));
            improved_img(i,j) = median(img_block(:));
        end
    end
    improved_img = uint8(improved_img);
    error = immse(img,improved_img);

imwrite(improved_img,strcat("guussian_noise_",num2str(noise_density(ns)),"_median_"
,int2str(window_size(ws)), ".bmp"));
    subplot(2,3,ws+1); imshow(improved_img); title(strcat("median
size:",int2str(window_size(ws)), " immse:",num2str(error)));
end
fig = figure;
subplot(2,3,1); imshow(img); title(strcat("noisy image with density
",num2str(noise_density(ns))));
% apply box filter on noisy image
for ws=1:size(window_size)
    boxfilter = ones(window_size(ws))/(window_size(ws).^2);
    improved_img = apply_filter(noisy_img, boxfilter, window_size(ws));
    improved_img = uint8(improved_img);
    error = immse(img,improved_img);

imwrite(improved_img,strcat("guussian_noise_",num2str(noise_density(ns)),"_box_",
int2str(window_size(ws)), ".bmp"));
    subplot(2,3,ws+1); imshow(improved_img); title(strcat("box
size:",int2str(window_size(ws)), " immse:",num2str(error)));
end
end

```

Edge Detection -٣-٤

١-٣-٤-بخش اول

کد فایل main

```

img = imread("E:\Dars\Masters\digital image
processing\Homeworks\Images\3\Elaine.bmp");
filter_a = [1 0 -1]./2;
filter_b = [1 0 -1; 1 0 -1; 1 0 -1]./6;
filter_c = [1 0 -1; 2 0 -2; 1 0 -1]./8;
fig = figure;
subplot(2,2,1); imshow(img); title("Original Image");
a = conv2(img,filter_a,"same");
imwrite(uint8(a), 'apply_filter_a.bmp');
subplot(2,2,2); imshow(uint8(a)); title("filter [1 0 -1]./2");
b = conv2(img,filter_b,"same");
imwrite(uint8(b), 'apply_filter_b.bmp');
subplot(2,2,3); imshow(uint8(b)); title("filter [1 0 -1; 1 0 -1; 1 0 -1]./6");
c = conv2(img,filter_c,"same");

```

```

imwrite(uint8(c),'apply_filter_c.bmp');
subplot(2,2,4); imshow(uint8(c)); title("filter [1 0 -1; 2 0 -2; 1 0 -1]./8");

```

٤-٣-٢-بخش دوم

:main كد فایل

```

img = imread("E:\Dars\Masters\digital image
processing\Homeworks\Images\3\Elaine.bmp");
robert1 = [1 0; 0 -1];
robert2 = [0 1; -1 0];
fig = figure;
subplot(1,3,1); imshow(img); title("Original Image");
a = conv2(img,robert1,"same");
imwrite(uint8(a),'apply_robert2.bmp');
subplot(1,3,2); imshow(uint8(a)); title("filter [1 0; 0 -1]");
b = conv2(img,robert2,"same");
imwrite(uint8(b),'apply_robert1.bmp');
subplot(1,3,3); imshow(uint8(b)); title("filter [0 1; -1 0]");

```

Unsharp Masking -4-4

```

src_path = './Images/3/Elaine.bmp'
src = cv2.imread(src_path, cv2.IMREAD_GRAYSCALE)
def unsharp_mask_filter(I, I_prime, alpha):
    return (1 - alpha) * I + alpha * I_prime

alpha_values = [0.2, 0.4, 0.6, 0.8]
kernel_sizes = [3, 5, 7, 9, 11]
border_type = cv2.BORDER_DEFAULT
# apply gaussian filter on original image with different kernel size
gaussian.blur_results = []
for i in range(len(kernel_sizes)):
    current_kernel_size = kernel_sizes[i]
    gaussian.blur_results.append(cv2.GaussianBlur(src, (current_kernel_size, current_kernel_size), border_type))
# apply unsharped mask filter with different alpha values
unsharped.masked.filter_results = []
for i in range(len(alpha_values)):
    current_alpha = alpha_values[i]
    for j in range(len(kernel_sizes)):
        current_src_blurred = gaussian.blur_results[j]
        unsharped.masked.filter_results.append(unsharp_mask_filter(src, current_src_blurred, current_alpha))

gaussian.blur_3 = gaussian.blur_results[0]
gaussian.blur_5 = gaussian.blur_results[1]
gaussian.blur_7 = gaussian.blur_results[2]
gaussian.blur_9 = gaussian.blur_results[3]
gaussian.blur_11 = gaussian.blur_results[4]
unsharped.masked.filter_gaussian_02_3 = unsharped.masked.filter_results[0]

```

```

unsharped_masked_filter_gaussian_02_5 = unsharped_masked_filter_results[1]
unsharped_masked_filter_gaussian_02_7 = unsharped_masked_filter_results[2]
unsharped_masked_filter_gaussian_02_9 = unsharped_masked_filter_results[3]
unsharped_masked_filter_gaussian_02_11 = unsharped_masked_filter_results[4]

unsharped_masked_filter_gaussian_04_3 = unsharped_masked_filter_results[5]
unsharped_masked_filter_gaussian_04_5 = unsharped_masked_filter_results[6]
unsharped_masked_filter_gaussian_04_7 = unsharped_masked_filter_results[7]
unsharped_masked_filter_gaussian_04_9 = unsharped_masked_filter_results[8]
unsharped_masked_filter_gaussian_04_11 = unsharped_masked_filter_results[9]

unsharped_masked_filter_gaussian_06_3 = unsharped_masked_filter_results[10]
unsharped_masked_filter_gaussian_06_5 = unsharped_masked_filter_results[11]
unsharped_masked_filter_gaussian_06_7 = unsharped_masked_filter_results[12]
unsharped_masked_filter_gaussian_06_9 = unsharped_masked_filter_results[13]
unsharped_masked_filter_gaussian_06_11 = unsharped_masked_filter_results[14]

unsharped_masked_filter_gaussian_08_3 = unsharped_masked_filter_results[15]
unsharped_masked_filter_gaussian_08_5 = unsharped_masked_filter_results[16]
unsharped_masked_filter_gaussian_08_7 = unsharped_masked_filter_results[17]
unsharped_masked_filter_gaussian_08_9 = unsharped_masked_filter_results[18]
unsharped_masked_filter_gaussian_08_11 = unsharped_masked_filter_results[19]

plt.figure(figsize=(20,10))
plt.subplot(2, 3, 1)
plt.title("Original")
plt.imshow(src, cmap='gray')

plt.subplot(2, 3, 2)
plt.title(r"Gaussian blured with kernel size = 3")
plt.imshow(gaussian_blur_3, cmap='gray')

plt.subplot(2, 3, 3)
plt.title(r"Gaussian blured with kernel size = 3 and $\alpha = 0.2$")
plt.imshow(unsharped_masked_filter_gaussian_02_3, cmap='gray')

plt.subplot(2, 3, 4)
plt.title(r"Gaussian blured with kernel size = 3 and $\alpha = 0.4$")
plt.imshow(unsharped_masked_filter_gaussian_04_3, cmap='gray')

plt.subplot(2, 3, 5)
plt.title(r"Gaussian blured with kernel size = 3 and $\alpha = 0.6$")
plt.imshow(unsharped_masked_filter_gaussian_06_3, cmap='gray')

plt.subplot(2, 3, 6)
plt.title(r"Gaussian blured with kernel size = 3 and $\alpha = 0.8$")
plt.imshow(unsharped_masked_filter_gaussian_08_3, cmap='gray')
plt.show()

```

