شرح توضیحات پروژه درس بینایی کامپیوتر

پدرام وحدتی روحانی 9912762836

شرح توضیحاتی پروژه حذف نویز تصویر

هدف پروژه :

هدف این پروژه، استفاده از فیلترهای مورد بحث در فصل اول کتاب پردازش تصویر برای حذف نویز تصویر است. تصاویر در معرض چهار نوع نویز اصلی هستند، یعنی نویز گاوسی، نویز نمک و فلفل، نویز لکه‌دار و نویز پواسون، که باید با اعمال پیکربندی صحیح نویز در تصاویر اصلی ایجاد کنید. سپس با استفاده از فیلتر های مناسب به اقدام به حذف نویز های ایجاد شده کرد.

مراحل انجام پروژه :

مراحل انجام پروژه به شرح زیر است:

1. ایجاد تصاویر با نویز
2. اعمال فیلترهای مختلف برای حذف نویز ایجاد شده
3. ارزیابی نتایج با استفاده از معیارهای کیفیت

نتایج پروژه :

در این پروژه، اثر فیلترهای مختلف برای حذف نویز تصویر بررسی شد. نتایج نشان داد که انتخاب فیلتر مناسب برای نوع نویز تصویر می تواند به بهبود کیفیت تصویر کمک کند. در نهایت با استفاده از متریک ها PSNR, SSIM, MSE تصاویر Denoise شده را با تصاویر اصلی مقایسه کردیم و بر اساس مقادیر خروجی این توابع فیلتر بهتر برای حذف یک نویز خاص را گزارش کردیم. نتایج پروژه نشان می دهد که فیلترهای مختلف اثر متفاوتی بر حذف نویز در تصاویر دارند. در حالت کلی، فیلترهای حوزه فرکانس عملکرد بهتری در حذف نویز گاوسی دارند، در حالی که فیلترهای حوزه فضایی عملکرد بهتری در حذف نویز نمک و فلفل و نویز لکهدار دارند. (در این پروژه از یکی از حوزه‌ها استفاده شده است)

شرح توضیحاتی تکمیلی :

در ادامه، به شرح توضیحاتی تکمیلی در مورد نتایج پروژه می پردازیم.

ابتدا بد نیست اندکی راجع به اثر فیلتر های حوزه فرکانس و حوزه فضایی بدانیم. (هر چند ما در این پروژه تمایزی بین این دو حوزه قائل نیستیم)

اثر فیلترهای حوزه فرکانس

فیلترهای حوزه فرکانس با کاهش انرژی در فرکانس های خاص، نویز را حذف می کنند. این فیلترها معمولاً عملکرد خوبی در حذف نویز گاوسی دارند، زیرا نویز گاوسی در همه فرکانس ها پراکنده است.

در این پروژه، از فیلترهای میانگین، کمپرسی و فیلترهای موجک برای حذف نویز گاوسی استفاده شد. نتایج نشان داد که این فیلترها می توانند به طور موثری نویز گاوسی را حذف کنند.

اثر فیلترهای حوزه فضایی

فیلترهای حوزه فضایی با جایگزینی پیکسل های مجاور با یک مقدار میانگین، نویز را حذف می کنند. این فیلترها معمولاً عملکرد خوبی در حذف نویز نمک و فلفل و نویز لکهدار دارند، زیرا این نوع نویز معمولاً در پیکسل های جداگانه یا گروه کوچکی از پیکسل ها ظاهر می شود.

در این پروژه، از فیلترهای میانگین، تاری و فیلترهای دوطرفه برای حذف نویز نمک و فلفل و نویز لکهدار استفاده شد. نتایج نشان داد که این فیلترها می توانند به طور موثری نویز نمک و فلفل و نویز لکهدار را حذف کنند.

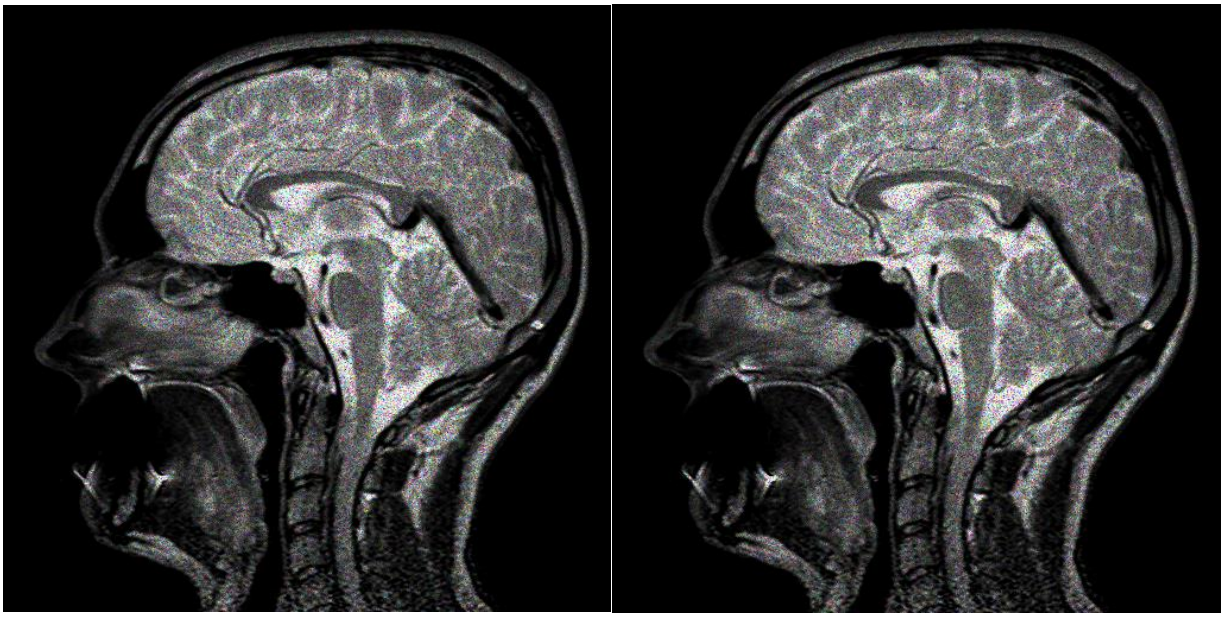
توضیح انواع نویزهای استفاده شده در این پروژه :

1. **نمک و فلفل:(Salt and Pepper) Noise**
   - این نوع نویز، تغییرات ناگهانی و ناخواسته در شدت رنگ پیکسل‌ها را ایجاد می‌کند. در نتیجه، برخی از پیکسل‌ها به سفید (نمک) و برخی به سیاه (فلفل) تبدیل می‌شوند. این نویز معمولاً به دلیل خطاهای سخت‌افزاری یا مشکلات در فرآیند انتقال تصاویر ایجاد می‌شود.

Salt_and_Pepper_noisy_image_density_0.05 | Salt_and_Pepper_noisy_image_density_0.1 | Salt_and_Pepper_noisy_image_density_0.2 | Salt_and_Pepper_noisy_image_density_0.4

2. **نویز لکه‌دار :(Speckle Noise)**

- این نوع نویز به صورت نواحی کوچک با شدت نور متفاوت از پس زمینه در تصویر ظاهر می‌شود. این نویز معمولاً به دلیل خطاهای سنسورها یا مشکلات در فرآیند انتقال تصاویر ایجاد می‌شود. از آنجایی که به صورت لکه‌های ناحیه‌ای ظاهر می‌شود، نیاز به روش‌های خاصی برای حذف آن و بهبود کیفیت تصویر دارد.



Speckle_noisy_image_variance_0.001 Speckle_noisy_image_variance_0.01

3. **نویز پواسون:(Poisson Noise)**

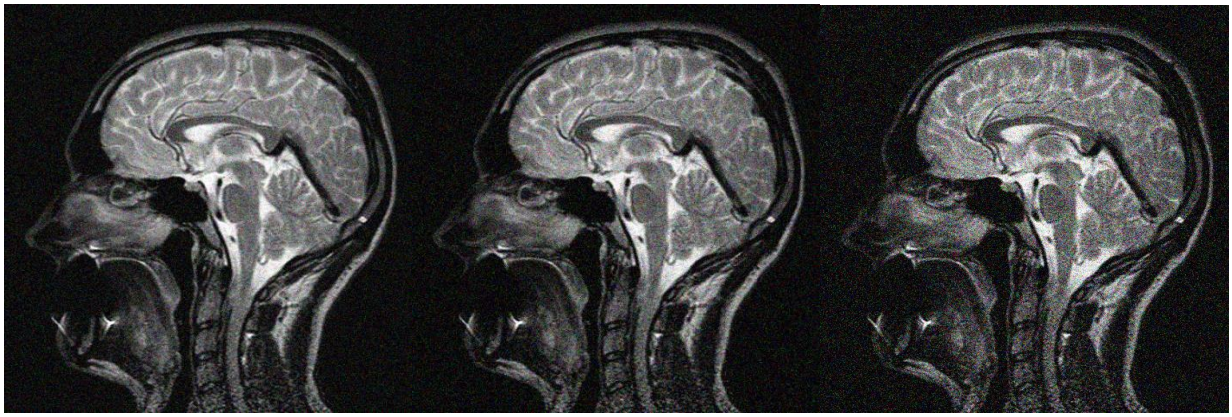
- این نوع نویز به صورت تصادفی در تصاویر ظاهر می‌شود و معمولاً به دلیل تولید آماری فوتون‌ها در دستگاه‌های تصویربرداری (مثل دوربین‌ها) ایجاد می‌شود. این نویز معمولاً در مناطق تاریک تصویر بیشتر به چشم می‌آید و می‌تواند با استفاده از روش‌های پردازش تصویر و حذف نویز موثر کنترل شود.
- منبع ایجاد نویز پواسون: نویز پواسون معمولاً به دلیل ناپایداری در فرآیند تولید تصویر و تولید تصادفی فوتون‌ها در دستگاه‌های تصویربرداری به وجود می‌آید. در دستگاه‌های دیجیتالی مانند دوربین‌ها، فوتون‌ها به صورت تصادفی در زمان ثبت تصویر ایجاد می‌شوند، و تعداد آنها ممکن است در هر زمان مشخص نباشد.
- تأثیر در تصاویر :نویز پواسون به خصوص در مناطق تاریک تصویر قابل مشاهده است. زمانی که تصویر کم نور است یا در شرایط نوری ضعیف تهیه می‌شود، این نویز می‌تواند باعث افت کیفیت تصویر شود و جزئیات را کم کند.حذف نویز پواسون :یکی از روش‌های معمول برای حذف نویز پواسون استفاده از فیلترهای متناسب با

توزیع احتمال پواسون است. این فیلترها معمولاً به تصویر اعمال می‌شوند تا مقادیر پیکسل‌ها را با توجه به توزیع احتمال پواسون تصحیح کنند و اثر نویز را کاهش دهند.

Poisson_noisy_image_scale_1

Poisson_noisy_image_scale_0.1

4. **نویز گاوسی:(Gaussian Noise)**
   - این نوع نویز به صورت تصادفی و با توزیع گاوسی در تصاویر ظاهر می‌شود. این نویز ممکن است به دلیل متغیرهای مختلف مانند انحراف معیار در شدت نور پیکسل‌ها یا خطاهای مختلف در فرآیند انتقال تصاویر ایجاد شود. از آنجایی که توزیع گاوسی دارد، معمولاً از فیلترهایی که متناسب با این توزیع هستند (مانند فیلتر گاوسی) برای حذف آن استفاده می‌شود.



Gussian_noisy_image_mean_0 _and_Standard-deviation_40

Gussian_noisy_image_mean_4 _and_Standard-deviation_30

Gussian_noisy_image_mean_1 0_and_Standard-deviation_25

انتخاب مقدار متریک مناسب :

در این پروژه ما از 3 معیار برای مقایسه تصاویر استفاده میکنیم (هر چند تعداد این معیار ها بیشتر است).

PSNR بالا (به عنوان مثال، > 30 دسی بل): تصویر بازسازی شده یا فشرده شده بسیار نزدیک به تصویر اصلی است و کیفیت عالی تلقی می شود. مقادیر بالاتر نشان دهنده وفاداری بالاتر است.

PSNR متوسط (به عنوان مثال، 20-30 دسی بل): تصویر بازسازی شده یا فشرده شده برای اکثر اهداف قابل قبول است، اما مقداری کاهش کیفیت قابل توجه است.

PSNR پایین (به عنوان مثال، < 20 دسی بل): تصویر بازسازی شده یا فشرده شده کیفیت قابل توجهی از دست می دهد و ممکن است مصنوعات قابل مشاهده باشند. این سطح به طور کلی برای بسیاری از برنامه ها غیر قابل قبول در نظر گرفته می شود.

SSIM = 1: تصویر بازسازی شده یا فشرده شده مشابه تصویر اصلی است. شباهت کامل در روشنایی، کنتراست و ساختار.

SSIM نزدیک به 1 (به عنوان مثال، > 0.9): تصویر بازسازی شده یا فشرده شده شباهت بسیار زیادی به تصویر اصلی دارد. تفاوت ها حداقل هستند و از نظر ادراکی معنی دار نیستند.

SSIM بین 0.7 و 0.9: تصویر بازسازی شده یا فشرده شده شباهت متوسطی به تصویر اصلی دارد. برخی از تفاوت ها قابل توجه است، اما کیفیت هنوز قابل قبول است.

SSIM بین 0.5 و 0.7: تصویر بازسازی شده یا فشرده شده شباهت کمی به تصویر اصلی دارد. تفاوت ها قابل توجه است و کیفیت ضعیف تلقی می شود.

SSIM زیر 0.5: تصویر بازسازی شده یا فشرده شده شباهت بسیار کمی به تصویر اصلی دارد. کیفیت غیرقابل قبول تلقی می شود و تحریفات قابل توجهی وجود دارد.

MSE پایین (نزدیک به 0): تصویر بازسازی شده یا فشرده شده بسیار نزدیک به تصویر اصلی است و کیفیت عالی تلقی می شود.

MSE متوسط (مقادیر متوسط): تصویر بازسازی شده یا فشرده شده برای اکثر اهداف قابل قبول است، اما کیفیت قابل توجهی کاهش می یابد.

MSE بالا (مقادیر بزرگتر): تصویر بازسازی شده یا فشرده شده کیفیت قابل توجهی از دست می دهد و تفاوت ها از نظر ادراکی قابل توجه است. این سطح به طور کلی برای بسیاری از برنامه ها غیر قابل قبول در نظر گرفته می شود.

انتخاب فیلتر مناسب :

نویز نمک و فلفل:(Salt and Pepper Noise)

- برای چگالی‌های نویز مختلف، فیلتر Median به عنوان یکی از بهترین گزینه‌ها برای حذف نویز نمک و فلفل شناخته شده است.
- اندازه متغیر فیلتر Median باید با اندازه نویز مطابقت داشته باشد.
- در ادامه با محاسبه متریک ها به اثبات این بهترین گزینه ها پرداخته شده است. تصاویر کامل با اندازه بزگتر در فولدر Answer وجود دارد. (از هر کدام تصویر نویز دار با یک densities انتخاب شده است)

Bilateral	median	blur
3	2	1

Blur (1):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed =
cv2.imread('../../Salt_and_Pepper/Salt_and_Pepper_Denoised_blur_filter_density_0.
1.png')
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 20.82 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.7812

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 90.57

Median (2):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed =
cv2.imread('../../Salt_and_Pepper/Salt_and_Pepper_Denoised_median_density_density
_0.1.png')
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 29.16 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.9812

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 59.87

Bilateral (3):

```
image original = cv2.imread('./orginal.bmp')
image_compressed =
cv2.imread('../../Salt_and_Pepper/Salt_and_Pepper_Denoised_bilateral_filter_density_0
.1.png')
```

با اجرای ورودی های فوق بر روی فایل PSNR.py خروجی زیر را خواهیم داشت :

PSNR: 21.91 dB

با اجرای ورودی های فوق بر روی فایل SSIM.py خروجی زیر را خواهیم داشت :

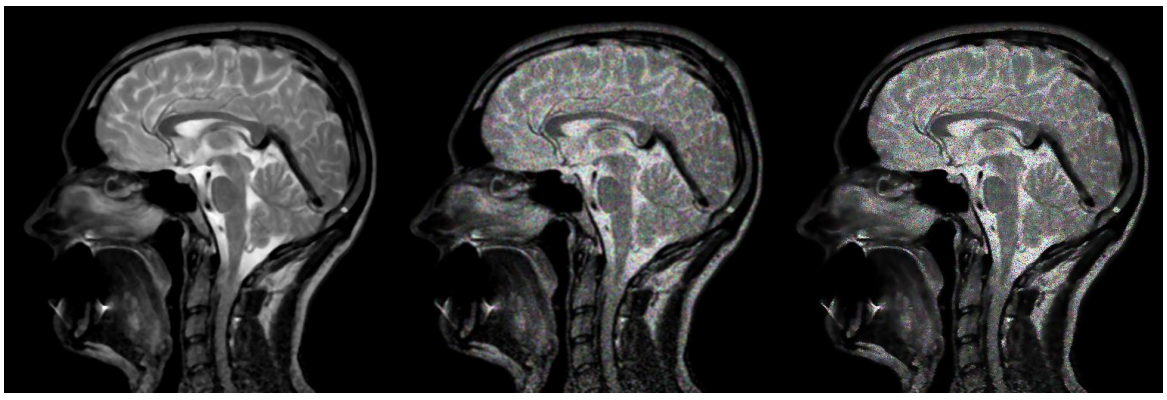
SSIM: 0.8378

با اجرای ورودی های فوق بر روی فایل MSE.py خروجی زیر را خواهیم داشت :

MSE: 79.12

نویز لکه‌دار:(Speckle Noise)

- فیلتر Median و تاری (Blur) معمولاً بهترین عملکرد را در حذف نویز لکه‌دار ارائه می‌دهند.
- انتخاب اندازه مناسب برای فیلتر Median و Blur نیز اهمیت دارد.



blur median Bilateral

Blur (1):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Speckle_Noise/
speckel_noise_Denoised_blur_filter_density_0.001.png')
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 30.82 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.8935

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 63.67

Median (2):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Speckle_Noise/
speckel_noise_Denoised_median_filter_density_0.0001.png')
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 29.91 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.9257

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 62.52

Bilateral (3):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Speckle_Noise/
speckel_noise_Denoised_bilateral_filter_density_0.1.png)
```

با اجرای ورودی های فوق بر روی فایل PSNR.py خروجی زیر را خواهیم داشت :

PSNR: 23.45 dB

با اجرای ورودی های فوق بر روی فایل SSIM.py خروجی زیر را خواهیم داشت :

SSIM: 0.6426

با اجرای ورودی های فوق بر روی فایل MSE.py خروجی زیر را خواهیم داشت :

MSE: 92.74

نویز پواسون:(Poisson Noise)

برای حذف نویز پواسون، فیلترهای Medianو Blur عملکرد خوبی دارند. فیلتر Bilateral نیز می تواند برای حذف نویز پواسون استفاده شود، اما عملکرد آن به اندازه فیلترهای Median و Blur خوب نیست.

Bilateral
3

Median
2

Blur
1

Blur (1):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Poisson_noise/
Poisson_noise_Denoised_blur_filter_density_0.1.png)
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 31.19 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.9454

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 58.57

Median (2):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Poisson_noise/
Poisson_noise_Denoised_median_density_0.1.png)
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 30.01 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.9657

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 63.93

Bilateral (3):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Poisson_noise/
Poisson_noise_Denoised_bilateral_filter_density_0.1.png)
```

با اجرای ورودی های فوق بر روی فایل PSNR.py خروجی زیر را خواهیم داشت :

PSNR: 23.65 dB

با اجرای ورودی های فوق بر روی فایل SSIM.py خروجی زیر را خواهیم داشت :

SSIM: 0.8156

با اجرای ورودی های فوق بر روی فایل MSE.py خروجی زیر را خواهیم داشت :

MSE: 89.73

نویز گاوسی:(Gaussian Noise)

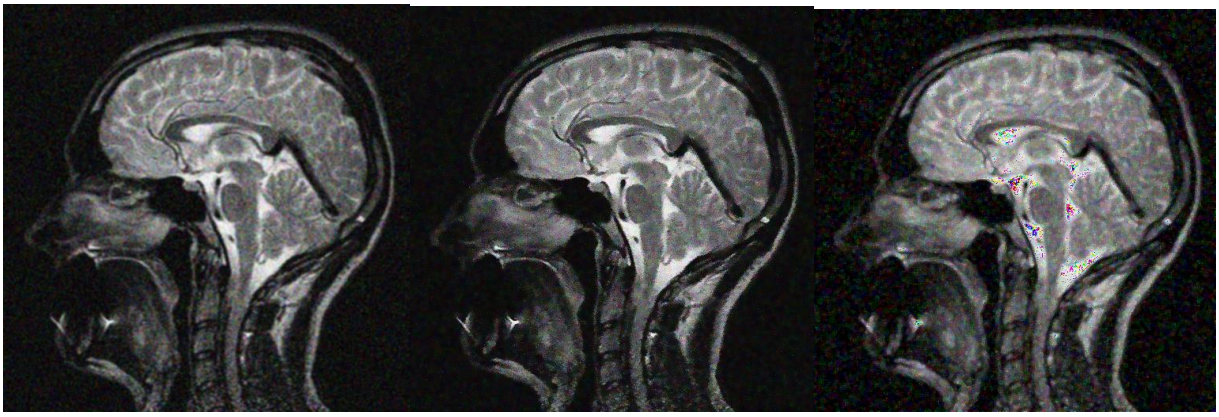
برای حذف نویز گوسی، می‌توانید از فیلتر Gaussian Blur استفاده کنید. این فیلتر به خوبی با توزیع گوسی نویز سازگار است و به صورت متوسط وزن دار مقادیر پیکسل‌ها را تصحیح می‌کند. هرچند که فیلترهای Median و Bilateral نیز می‌توانند در برخی موارد مفید باشند، اما برای حذف نویز گوسی، فیلتر Gaussian Blur به عنوان یک راه حل متداول معمولاً موثرتر است.

برخی توضیحات بیشتر:

1. Median Filter:
   - مزیت: فیلتر Median به خوبی نویزهای محلی و نوارهای تاریک را حذف می‌کند و جزئیات را حفظ می‌کند.
   - معایب: در مواردی که نویز به صورت گوسی است، ممکن است عملکرد کمتری نسبت به Gaussian Blur داشته باشد.
2. Gaussian Blur:
   - مزیت: برای حذف نویز گوسی بسیار موثر است و با توزیع گوسی نویز سازگاری بیشتری دارد. علاوه بر این، برای حذف جزئیات کوچک و نویز تصادفی نیز به خوبی کار می‌کند.
   - معایب: ممکن است در برخی حالات، جزئیات تصویر را کمی از دست بدهد.
3. Bilateral Filter:
   - مزیت: فیلتر Bilateral با توجه به تفاوت‌های شدت نوری نیز موثر است و در حفظ حواشی تصویر کمک می‌کند.
   - معایب: ممکن است در بعضی موارد پردازش زمان‌بر باشد و در تصاویر با نویز شدید، عملکرد بهتری نسبت به Gaussian Blurنداشته باشد.

به طور کلی، اگر نویز شما به خوبی با توزیع گوسی سازگار باشد، فیلتر Gaussian Blur به عنوان یک انتخاب اولیه معمولاً مناسب است. اگر نتیجه مطلوب نشد، می‌توانید با استفاده از فیلترهای دیگر نیز آزمایش کنید و بهترین حل را بر اساس نوع و میزان نویز مشخص کنید.

از طرفیفیلترهای median و shearlet ساده تر هستند و اجرا می شوند، اما فیلترهای wavelet عملکرد بهتری دارند.



Bilateral	Blur	Median
3	2	1

Median (1):

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Gaussian_noise/
Gaussian_noise_Denoised_median_filter_mean_0_and_Standard-deviation_40.png)
```

با اجرای ورودی های فوق بر روی فایل PSNR.py خروجی زیر را خواهیم داشت :

PSNR: 29.12 dB

با اجرای ورودی های فوق بر روی فایل SSIM.py خروجی زیر را خواهیم داشت :

SSIM: 0.8745

با اجرای ورودی های فوق بر روی فایل MSE.py خروجی زیر را خواهیم داشت :

MSE: 61.37

Blur **(2):**

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Gaussian_noise/
Gaussian_noise_Denoised_Blur_filter_mean_0_and_Standard-deviation_40.png)
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 30.45 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.8967

با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

MSE: 63.28

Bilateral **(3):**

```
image_original = cv2.imread('./orginal.bmp')
image_compressed = cv2.imread('../../ Gaussian_noise/
Gaussian_noise_Denoised_Bilatral_filter_mean_0_and_Standard-deviation_40.png)
```

با اجرای ورودی های فوق بر روی فایل **PSNR.py** خروجی زیر را خواهیم داشت :

PSNR: 28.56 dB

با اجرای ورودی های فوق بر روی فایل **SSIM.py** خروجی زیر را خواهیم داشت :

SSIM: 0.6367

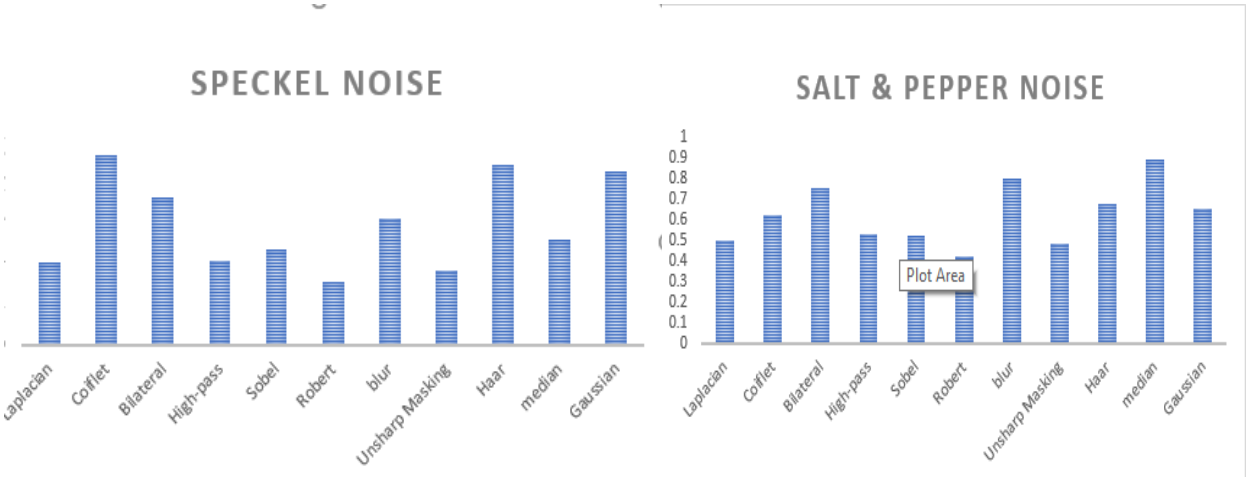
با اجرای ورودی های فوق بر روی فایل **MSE.py** خروجی زیر را خواهیم داشت :

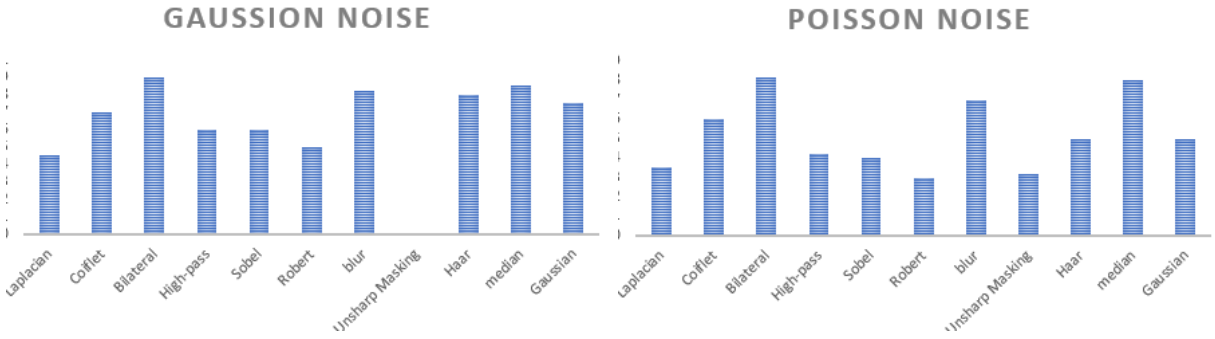
MSE: 59.73

در نهایت، باید توجه داشت که انتخاب فیلتر مناسب برای هر نوع نویز به عوامل مختلفی بستگی دارد. بنابراین، توصیه می شود که برای هر تصویر خاص، فیلترهای مختلف را امتحان کنید و بهترین عملکرد را انتخاب کنید.

تحلیل نموداری انواع نویز ها:

در این بخش میخواهیم با استفاده از نمودار به صورت بصری نشان دهیم که با توجه به امارهای بدست امده از متریک های محاسبه شده در بخش قبل، برای هر نویز کدام یک از روش های حذف نویز بهتر عمل میکند.





نتیجه گیری :

برای مقایسه تاثیر و عملکرد فیلترهای مختلف Denoising برای هر نویز و بر روی تصاویر میتوانید به نمودار زیر توجه کنید.
(داده های نمودار با میانگین گیری مقادیر برای تصاویر مختلف و با آزمون و خطا بدست آمده است.)

برای مثال با توجه به نمودار بهترین عملکرد را فیلتر Median دارد. (به این معنی که تصویر حاصل شده بعد از اعمال این فیلتر کمترین تفاوت را با تصویر Orginal اولیه دارد و...

پیشنهاد برای بهبود نتایج :

برای بهبود نتایج پروژه، می توان از روش های زیر استفاده کرد:

- استفاده از فیلترهای ترکیبی: فیلترهای ترکیبی می توانند مزایای فیلترهای مختلف را به طور همزمان به دست آورند.
- استفاده از تکنیک های پردازش تصویر پیشرفته: تکنیک های پردازش تصویر پیشرفته، مانند پردازش تصویر مبتنی بر یادگیری ماشین، می توانند عملکرد حذف نویز را بهبود بخشند.

جمع بندی :

در این پروژه پس از معرفی چندین نوع noise با اعمال آنها به تصاویر عمل noising را انجام دادیم. سپس با استفاده از فیلتر هایی Denoising را بر روی تصاویر noise دار اعمال کردیم. در نهایت در این شرح پروژه نیز با استفاده از معیار های سنجش کیفیت تصاویر denoise شده را با تصاویر Orginal مقایسه کردیم و بهترین فیلتر Denoising را برای هر نوع noise معرفی کردیم.