دانشکده مهندسی کامپیوتر

انتقال داده‌ها

پاییز ۱۴۰۰

پروژه اول

حذف نویز

استاد درس . . . . . دکتر دیانت

نام . . . . . علی صداقی

شماره دانشجویی ۹۷۵۲۱۳۷۸

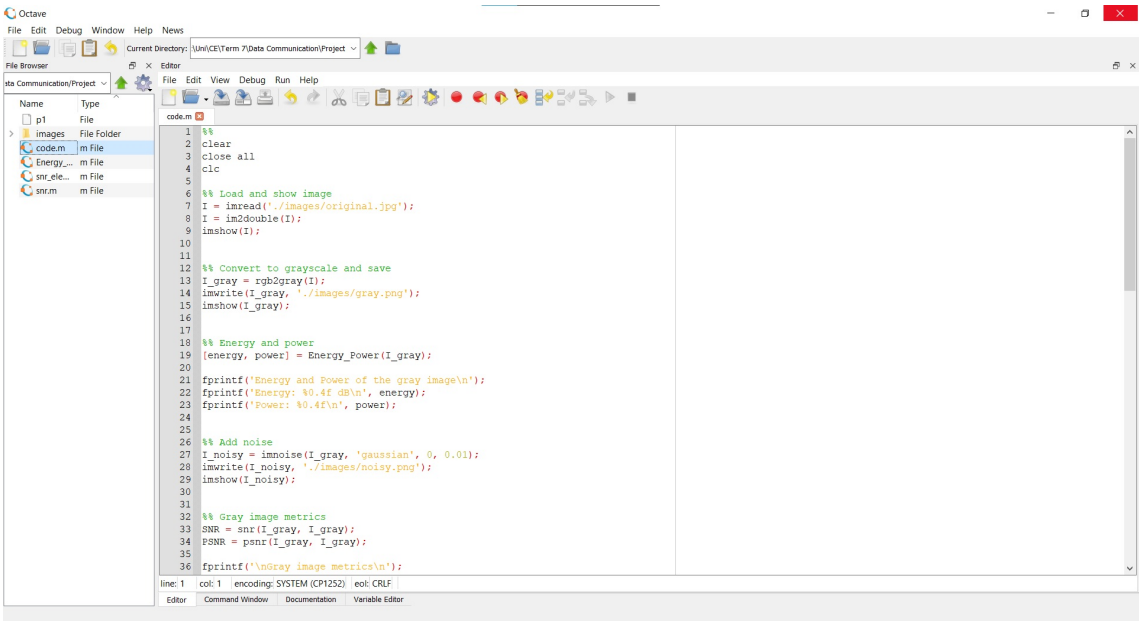
۱ گام اول

با توجه به حجم زیاد نرم‌افزار متلب و کمبود فضا، از نرم‌افزار GNU Octave استفاده کردم که دارای دو محیط کاربری متنی و گرافیکی است.

این برنامه تمامی دستورات متلب را پشتیبانی می‌کند و قابلیت بارگذاری پکیج‌ها و تولباکس‌های متلب را دارد.

برای مثال با دستور زیر می‌توانیم پکیج مربوط به تصاویر متلب را بارگذاری کنیم:

```
pkg load image
```



شکل ۱: محیط Octave

۲ گام دوم

با استفاده از دستور imread تصویر زیر را بارگذاری می‌کنیم.

برای راحت شدن محاسبات با استفاده از دستور im2double تایپ آن را به double تبدیل می‌کنیم.

سپس با دستور imshow تصویر بارگذاری شده را نمایش می‌دهیم.

شکل ۲: تصویر اصلی

۳ گام سوم

size تصویر اصلی که RGB است به صورت 1920x1080x3 می‌باشد.

با استفاده از دستور rgb2gray آن را تبدیل به یک تصویر grayscale با سایز 1920x1080 می‌کنیم.

در واقع تصویر اصلی دارای ۳ کانال رنگی بوده است اما تصویر جدید دیگر عمق و کانال ندارد و هر پیکسل آن با ۸ بیت یعنی عددی بین ۰ تا ۲۵۵ نمایش داده می‌شود. در صورتی که در دنیای RGB نیاز به ۲۴ بیت داریم.

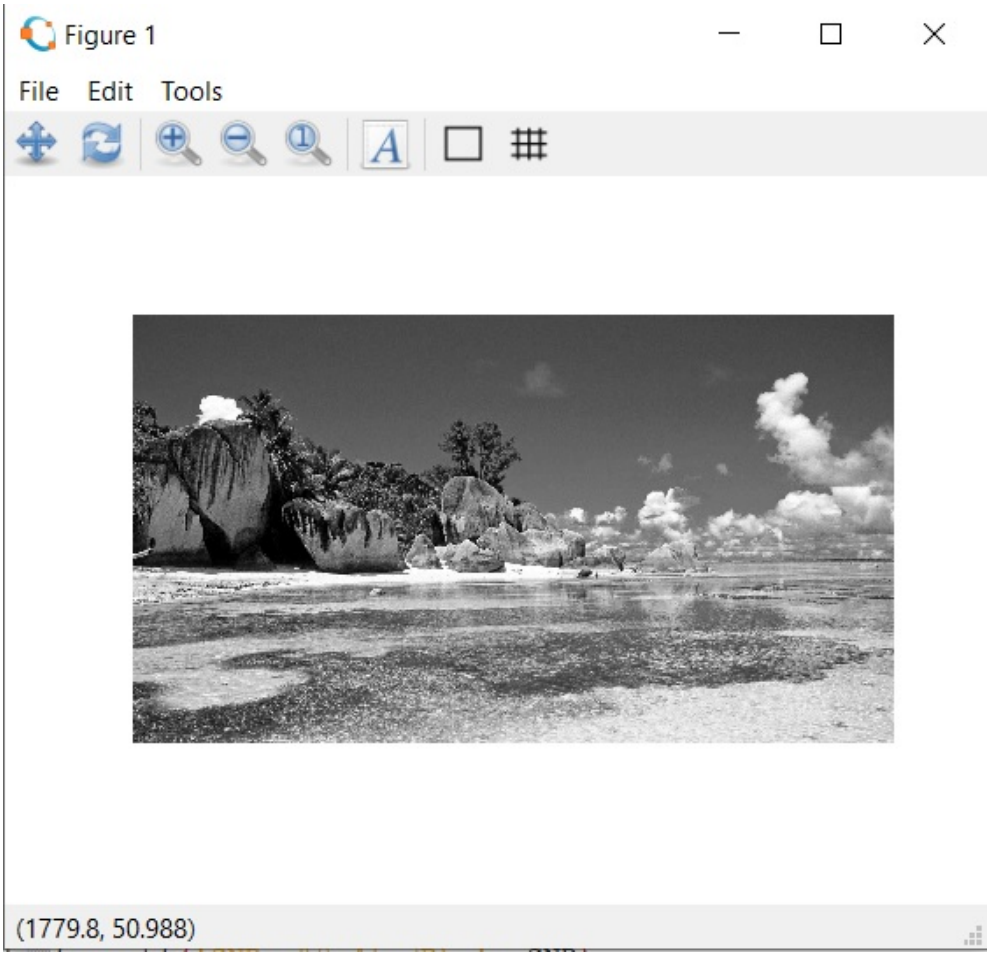
عدد ۰ معادل سیاه و عدد ۲۵۵ معادل سفید است و مقادیر بین آن در واقع روشنایی بین این دو مقدار می‌باشند. فرمول تبدیل به صورت زیر است.

$$0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

شکل ۳: تصویر خاکستری

۴ گام چهارم

با استفاده از دو دستور imwrite و imshow تصویر خاکستری شده را ذخیره می‌کنیم و نمایش می‌دهیم.



شکل ۴: دستور imshow

فرمت ذخیره‌سازی تصاویر را می‌توان در ۳ دسته زیر تقسیم کرد:

۱. Lossy compression

۲. Lossless compression

۳. Uncompressed

فرمت JPG از نوع اول می‌باشد و هنگام فشرده‌سازی بخشی از اطلاعات عکس حذف می‌شوند و دیگر قابل برگشت نیستند. این تصاویر دارای حجم کمتری هستند و مناسب دنیای اینترنت هستند.

فرمت PNG از نوع دوم می‌باشد و هنگام فشرده‌سازی اطلاعاتی از دست نمی‌دهد و می‌توان تصویر فشرده شده را به تصویر اولیه برگرداند. بنابراین حجم بیشتری نسبت به حالت بالا دارد.

فرمت BMP از نوع سوم می‌باشد و هنگام ذخیره‌سازی هیچ فشرده‌سازی انجام نمی‌دهد. حجم تصاویر در این حالت بسیار بالا بوده زیرا تمامی جزئیات و اطلاعات تصویر در این فرمت حفظ و ذخیره می‌شود. البته در بعضی حالت‌های این فرمت می‌توان فشرده‌سازی نیز داشت. فرمت ذخیره‌سازی TIFF نیز از همین نوع می‌باشد.

نکته: برای کاهش حجم پروژه ارسالی تصاویر را در حالت JPG ذخیره کردیم. اما حالت بدون فشرده‌سازی BMP است.

۵ گام پنجم

انرژی تصویر را برابر مجموع مربعات هر پیکسل تعریف می‌کنیم و در نهایت مقدار به دست آمده را به واحد دسی‌بل تبدیل می‌کنیم.

پس از محاسبه انرژی تصویر با تقسیم کردن آن بر بینهایت توان عکس را محاسبه می‌کنیم.

پیاده‌سازی این قسمت در فایل تابع Energy_Power.m موجود است.

با توجه به اینکه مقدار انرژی عکس خاکستری برابر 57.59 dB و مقدار توان آن برابر صفر شد می‌توان گفت این تصویر یک سیگنال انرژی است.

$$E = \sum_{i,j} I(i,j)^2$$
$$E_{dB} = 10 \times log_{10}(E)$$
$$P = \lim_{T\to\infty} \frac{1}{2T} \times \sum_{i,j} I(i,j)^2$$

۶ گام ششم

به تصویر خاکستری با استفاده از دستور imnoise یک نویز گاوسی با میانگین صفر و پراش 0.01 اضافه می‌کنیم.

شکل ۵: تصویر خاکستری نویزی شده

نسبت سیگنال به نویز SNR نامیده می‌شود. هرچه این مقدار بیشتر باشد یعنی نویز کمتر می‌باشد و بلعکس.

برای محاسبه این نسبت دو تابع به نام‌های snr.m و snr_elementwise.m نوشتیم که با روابط زیر این نسبت را محاسبه می‌کنند.

$$SNR = \frac{\sum_{i,j} Ref(i,j)^2}{\sum_{i,j} Noise(i,j)^2}$$
$$SNR_{ElementWise} = Average(\sum_{i,j} \frac{Ref(i,j)^2}{Noise(i,j)^2})$$

سپس مقادیر بدست آمده را به دسی‌بل تبدیل واحد می‌کنیم.

Noise برابر با تفاضل سیگنال اصلی Ref و سیگنال نویزی Noisy می‌باشد.

در حالت Element Wise بعضی از المان‌های مخرج برابر صف می‌شوند و این موضوع باعث می‌شود در آن المان این نسبت به بینهایت میل پیدا کند. به‌همین دلیل از حالت اول استفاده کردیم.

نکته: یک پیاده‌سازی دیگر برای محاسبه نسبت سیگنال به نویز وجود دارد که فقط یک ورودی می‌گیرد. نحوه کار به این صورت است که با محاسبه ۳ مقدار کمینه، بیشینه و انحراف معیار در سیگنال ورودی نسبت سیگنال به نویز را محاسبه می‌کند. اما این روش مناسب سیگنال‌هایی است که توزیع نرمال گاوسی دارند. اما تصویر یک سیگنال تصادفی است و هیچ توزیع نرمالی ندارد.

نکته: می‌توانستیم از تابع آماده snr در متلب نیز استفاده کنیم.

SNR gray image: ∞

PSNR gray image: ∞

SNR noisy image: 14.4273 dB

PSNR noisy image: 19.9957 dB

در تصویر اصلی مقادیر بینهایت بدست آمده زیرا مخرج کسر برابر صفر می‌شود.

۷ گام هفتم

مطابق کدی که در اسلایدها بود تصویر خاکستری و نویزی را به حوزه فرکانس می‌بریم.

با استفاده از دستور fft2 می‌توانیم عکس را از حوزه Spatial به حوزه فرکانس ببریم.

این دستور بر روی یک ماتریس یک تبدیل فوریه گسسته DFT می‌گیرد. و رابطه آن به صورت زیر است:

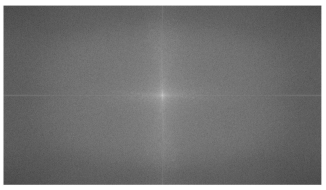
$$Y_{p+1,q+1} = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \omega_m^{jp} \omega_n^{kq} X_{j+1,k+1}$$

ω_m and ω_n are complex roots of unity:

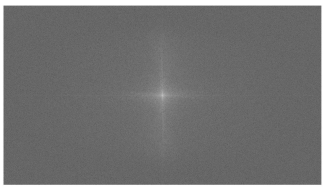
$$\omega_m = e^{-2\pi i/m}$$

$$\omega_n = e^{-2\pi i/n}$$

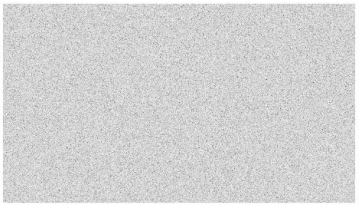
شکل ۶: رابطه تبدیل فوریه دو بعدی



شکل ۷: تصویر خاکستری در حوزه فرکانس



شکل ۸: تصویر نویزی در حوزه فرکانس



شکل ۹: تصویر نویز در حوزه فرکانس

این تصاویر در واقع یک محور مختصات دو بعدی هستند که محورهای آن شامل Vertical frequency و Horizontal frequency است. مرکز تصویر مبدا این دستگاه مختصات است. مقدار روشنایی هر نقطه در این دستگاه مختصات، دامنه (Amplitude) فرکانس در آن نقطه است.

در مرکز تصاویر که مبدا محورهای فرکانس است و بزرگی فرکانس در آن نقطه کم است نقاط نورانی‌تر هستند و دامنه بیشتری دارند. این نشان می‌دهد اطلاعات تصویر در فرکانس‌های پایین یعنی مرکز تصویر ذخیره شده است.

هر چه از مرکز دورتر می‌شویم و به اطراف می‌رویم بزرگی فرکانس‌ها افزایش می‌یابند ولی دامنه آن کمتر می‌شود و نقاط کم نورتر می‌شوند. این امر نشان می‌دهد اطلاعات کمی از تصویر در فرکانس‌های بالا ذخیره شده اند.

با توجه به اینکه نویز اضافه شده از نوع گاوسی بوده و توزیع اطلاعات در آن یکسان است، یعنی نقاط سفید در تصویر نویز پخش هستند (شکل ۹)، پس از اضافه شدن نویز تصویر فرکانسی روشن تر شده است (شکل ۸)

در واقع می‌توان گفت نقاط سفید در مرکز مربوط به تصویر اصلی بوده که دامنه بالا، فرکانس کم

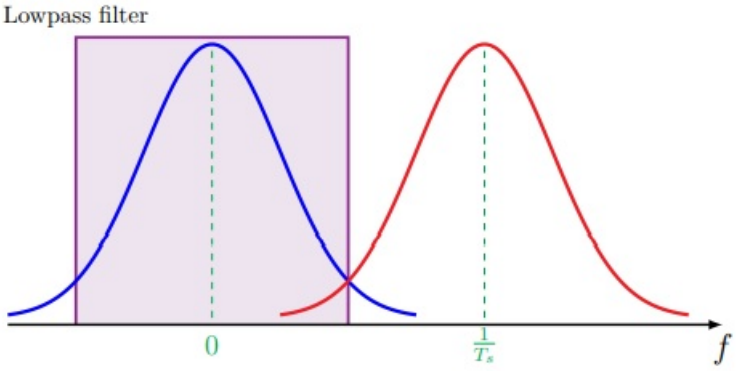
و اطلاعات زیاد دارد. نقاط سفید اطراف مربوط به نویز افزوده شده هست که فرکانس بالا، دامنه کم و اطلاعات کم دارد.

بالاترین فرکانس‌ها در اطراف است که دامنه کمی دارد و کم نور است و پایین‌ترین فرکانس‌ها در مرکز وجود دارد که دامنه بالایی دارد و پر نور است.

نکته: دستور شیفت باعث شده است که نقطه صفر فرکانس بر روی مرکز تصویر قرار گیرد.

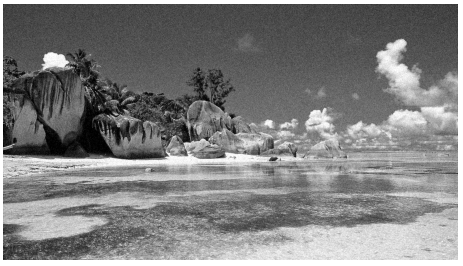
۸ گام هشتم

مطابق آنچه در درس آموختیم برای حذف نویز از فیلتر استفاده کنیم.

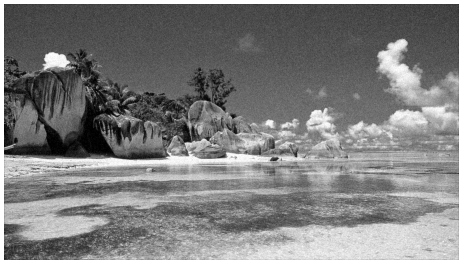


شکل ۱۰: تصویر خاکستری نویزی شده

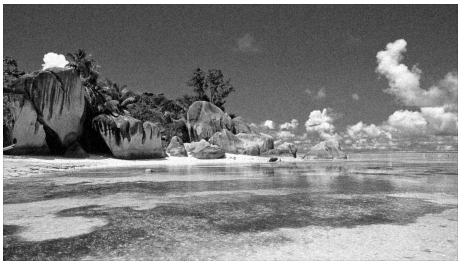
برای حذف نویز از ۴ فیلتر wiener2، filter2، conv2 و medfilt2 استفاده کردیم. سپس مقدار SNR و PSNR را برای هر یک محاسبه کردیم.



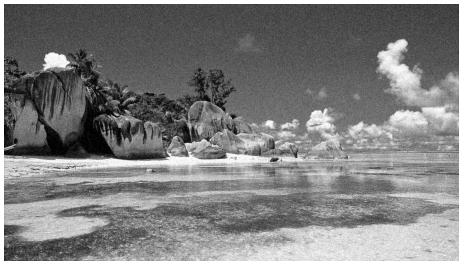
شکل ۱۱: Wiener Filter



شکل ۱۲: Filter



شکل ۱۳: Conv Filter



شکل ۱۴: Median Filter

	SNR	PSNR
WEINER	20.39	25.96
FILTER	19.97	25.54
CONV	19.97	25.54
MEDIAN	19.25	24.82

شکل ۱۵: مقایسه انواع فیلترها

با توجه به مقادیر جدول می‌توان گفت:

۱. فیلتر Weiner بهترین عملکرد را داشته.

۲. دو فیلتر Conv و Filter عملکرد یکسانی دارند.

۳. فیلتر Median بدترین عملکرد را دارد.

۴. پس از حذف نویز، نویز واقعا کاهش یافته است.

۱.۸ مفهوم PSNR

عبارت است از نسبت بین بیشینه توان سیگنال اصلی و توان سیگنال نویز. و عموما در مقیاس دسی‌بل بیان می‌شود.

هر چه مقدار آن بیشتر باشد یعنی عملکرد بهتری داشتیم.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

The PSNR (in dB) is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

شکل ۱۶: PSNR formula

نکات

۱. منابع استفاده شده در فایل resources.txt موجود است.

۲. لینک پروژه لاتک درون فایل latex link.txt موجود است.

۳. برای کم شدن حجم پروژه تصاویر را با فرمت JPG ذخیره کردم.

۴. تصاویر خروجی در پوشه src->images موجود است.

```
Command Window
Energy and Power of the gray image
Energy: 57.5988 dB
Power: 0.0000

Gray image metrics
SNR: Inf dB
PSNR: Inf dB

Noisy image metrics
SNR: 14.4302 dB
PSNR: 19.9987 dB

Removing noise with wiener filter
SNR: 20.3999 dB
PSNR: 25.9683 dB

Removing noise with filter
SNR: 19.9848 dB
PSNR: 25.5532 dB

Removing noise with conv filter
SNR: 19.9848 dB
PSNR: 25.5532 dB

Removing noise with median filter
SNR: 19.2687 dB
PSNR: 24.8371 dB
>>
```

شکل ۱۷: تصویر خروجی‌ها