دانشکده مهندسی کامپیوتر

استاد درس: دکتر ابوالفضل دیانت

پاییز ۱۴۰۲

گزارش پروژه اول
درس انتقال داده

زهراسادات طباطبائی ـ ستاره باباجانی
شماره دانشجویی: ۹۹۵۲۱۴۱۵ - ۹۹۵۲۱۱۰۹

پروژه

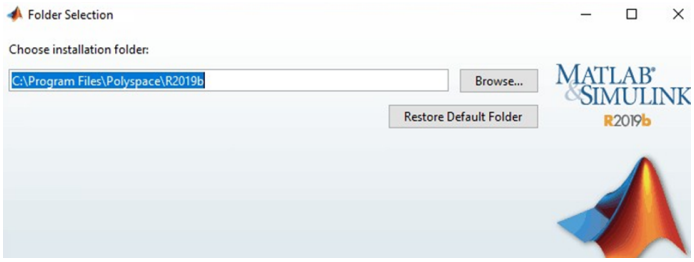
مقدمه

در این پروژه قصد داریم یک فایل تصویری را وارد نرم افزار MATLAB کرده و در ابتدا به آن نویز اضافه کنیم. و سپس نویز را حذف کنیم. برای انجام این کار، هشت گام نام برده شده در صورت پروژه را طی می‌کنیم و در هر مرحله توضیحاتی می‌دهیم.

مراحل

۱ گام اول

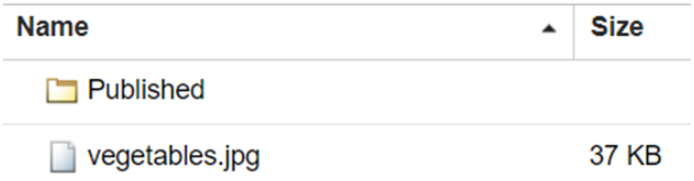
در این گام به نصب نرم افزار متلب MATLAB می‌پردازیم. از سایت soft۹۸.ir آن را دانلود کرده و مراحل نصب را طی می‌کنیم.(آدرس آن را همان مقدار دیفالت در پوشه files program گذاشتم.)



شکل ۱: نصب MATLAB

۲ گام دوم

در این مرحله، ابتدا یک عکس دلخواه از اینترنت دانلود کرده و سپس عکس را وارد نرم افزار متلب می‌کنیم.



شکل ۲: وارد کردن عکس

```
% Position of figure
set(gcf, 'Position',  [100, 40, 900, 600])
% reading image
pic = imread('vegetables.jpg');
subplot(2,3,1), imshow(pic), title('Original Picture');
```

شکل ۳: خواندن و نشان دادن عکس

شکل ۴: عکس

۳ گام سوم

در این گام قصد داریم فایل وارد شده را از RGB به یک فایل Gray-Scale bit ۸ تبدیل کنیم. به همین منظور از دستور `rgb2gray` مطابق زیر استفاده کرده‌ام.

```
% Convert to grayscale
grayPic = rgb2gray(pic);
```

شکل ۵: تبدیل به Gray-Scale bit ۸

نکته ۱) تصاویر در حالت کلی به ۳ دسته باینری، رنگی، scale gray تقسیم می‌شوند:

- تصاویر باینری، تنها شامل دو رنگ سیاه و سفید هستند. به طوری که هر پیکسل آنها اگر بیشتر از یک مقداری باشد سفید در نظر گرفته می‌شود. یعنی عدد ۱ و اگر کمتر از آن باشد، سیاه یعنی صفر در نظر گرفته می‌شود.

- تصاویر Gray-Scale شامل سطح‌های مختلفی از رنگ خاکستری هستند. این تصاویر دارای ۶۵۲ سطح خاکستری هست.
- تصاویر رنگی نیز شامل سه باند رنگ هستند که شامل RGB می‌شود که هرکدام رنگ متفاوتی دارند. مقایسه این تصاویر در کل ۲۴ bit/pixel است.

۴ گام چهارم

حال با استفاده از دستورات imshow و imwrite تصویر حاصل را مشاهده می‌کنیم و آن را در کامپیوتر خود ذخیره می‌کنیم.
نکته ۲) در این بخش برای ذخیره تصویر به صورت کامل و بدون هیچ فشرده سازی با اتلاف در کامپیوتر، از فرمت png استفاده میکنیم. نتیجه:

```
% Convert to grayscale
grayPic = rgb2gray(pic);

subplot(2,3,2), imshow(grayPic), title('Grayscale Picture');
imwrite(grayPic, 'gray_pic.png');
```

شکل ۶: ذخیره تصویر

شکل ۷: نتیجه

۵ گام پنجم

در این گام تصویر rgb ذخیره شده را دوباره با دستور imread می‌خوانیم و به تصویر خاکستری تبدیل می‌کنیم. سیگنال تصویر حاصل، از نوع سیگنال انرژی هست (و توان آن صفر است). مقدار انرژی آن اینگونه محاسبه می‌شود که تمام سطح های خاکستری عکس را باهم جمع می‌زنیم.

```
% Energy of grayscale picture
energy = sum(grayPic(:));
fprintf('The energy of gray phote is: %d\n', energy);
```

```
The energy of gray phote is: 709547509
>>
```

شکل ۸: به دست آوردن انرژی

۶ گام ششم

حال به تصویر خاکستری ای که در مرحله قبل ایجاد کردیم، یک نویز گاوسی با میانگین صفر و پراش (variance) دلخواه (اینجا ۰.۰۱) اضافه میکنیم. مطابق قطعه کد زیر، این کار را با دستور imnoise انجام داده ایم.

```
% Noise
gaussian_picture = imnoise(grayPic, 'gaussian', 0, 0.01);
imwrite(gaussian_picture, 'gaussian_picture.png');
subplot(2,3,3), imshow(gaussian_picture), title('Grayscale gaussian noise');
```

شکل ۹: به دست آوردن انرژی

شکل ۱۰: به دست آوردن انرژی

برای محاسبه snr از یک تابع آماده خود متلب به نام snr استفاده کرده ایم.

```
pic_snr = snr(double(grayPic(:)), double(grayPic(:)) - double(grayPic(:)));
fprintf('this is before snr: %d\n', pic_snr);

pic_snr2 = snr(double(grayPic(:)), double(grayPic(:)) - double(gaussian_picture(:)));
fprintf('this is after snr: %d\n', pic_snr2);
```

```
this is before snr: Inf
this is after snr: 1.514939e+01
```

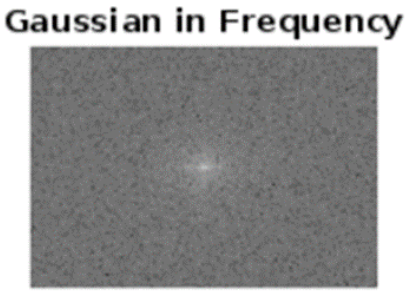
شکل ۱۱: به دست آوردن انرژی

نکته۳) در کل snr ، حاصل تقسیم توان سیگنال به نویز است. که یکای آن dB است. در حالتی که نویزی نداشته باشیم، حاصل این متغیر، بینهایت است. زیرا مقدار نویز صفر بوده و طبق گفته بالا، صفر در مخرج کسر قراره گرفته و در نتیجه حاصل را بینهایت میکند.

۷ گام هفتم

```
% Frequency domain
freqdomain = fftshift(log(abs(fft2(gaussian_picture))));
subplot(2,3,4), imshow(freqdomain, []), title('Gaussian in Frequency');
```

شکل ۱۲: به دست آوردن انرژی



شکل ۱۳: به دست آوردن انرژی

نکته۴) در FFT هر تابع به صورت جمع توابع سینوسی و کسینوسی نوشته میشود. هدف کلی FFT جابه جایی عکس بین حوزه های spatialوfrequency هست. از آن برای بردن یک عکس به حوزه فرکانس استفاده میکنیم. بیشترین فرکانس در کناره ها و کمترین فرکانس در وسط اتفاق می افتد.

۸ گام هشتم

حال قصد داریم نویز ایجاد شده را از تصویر حذف کنیم. دو روش برای اینکار وجود دارد، که یکی روش average filter و دیگری روش filter median که هر دو را روی تصویر انجام داده ایم. هر دوی این روش ها در تابع آماده متلب هستند و مطابق زیر انجام داده ام:

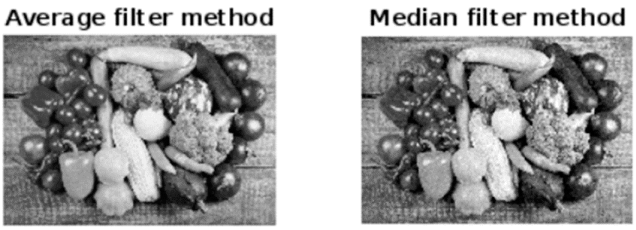
```
% Frequency domain
freqdomain = fftshift(log(abs(fft2(gaussian_picture))));
subplot(2,3,4), imshow(freqdomain, []), title('Gaussian in Frequency');

X=ones(5,5)/25;
method1_pic=imfilter(gaussian_picture, X);
subplot(2,3,5), imshow(method1_pic), title('Average filter method');
imwrite(method1_pic, 'method1_pic.png');

method2_pic=medfilt2(gaussian_picture);
subplot(2,3,6), imshow(method2_pic), title('Median filter method');
imwrite(method2_pic, 'method2_pic.png');

% Method1 psnr:
[psnr1, snr1] = psnr(gaussian_picture, method1_pic);
fprintf('\n Average filter psnr: %0.3f', psnr1);
% Method2 psnr:
[psnr2, snr2] = psnr(gaussian_picture, method2_pic);
fprintf('\n median filter psnr: %0.3f', psnr2);
```

شکل ۱۴: به دست آوردن انرژی



شکل ۱۵: به دست آوردن انرژی

```
Average filter psnr: 19.712
median filter psnr: 20.244
>>
```

شکل ۱۶: به دست آوردن انرژی

توابع آماده هر دو روش در شکل بالا استفاده شده است. برای روش اول از تابع imfilter و برای روش دوم از تابع medfilt۲ استفاده میشود. و در آخر نیز طبق نکته۵، مقدار psnr محاسبه شده است. که از همان تابع آماده psnr در متلب استفاده میشود. مطابق شکل آخر، طبق اعداد نشان داده شده، دقت روش دوم بالاتر است.