انتقال داده‌ها

موضوع پروژه : اضافه و حذف کردن نویز به تصویر

استاد درس: ابولفضل دیانت

نام دانشجو: محمدجواد مهدی‌تبار

نیم‌سال اول

سال تحصیلی ۱۴۰۱-۱۴۰۰

فهرست مطالب

فهرست تصاویر

پروژه ۱

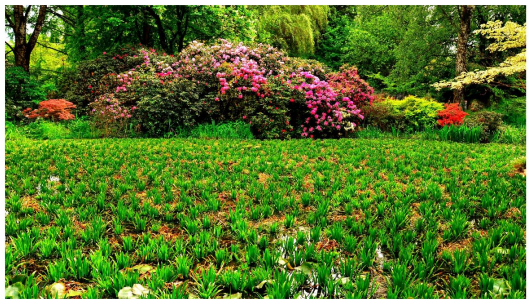
سیگنال تصویر و اضافه کردن نویز

۱.۱ گام اول

با توجه به توضیحات در مستند داده شده متلب را در سیستم خود نصب کرده و به ادامه کار پرداختم.

۲.۱ گام دوم

در این گام عکسی را به عنوان ورودی برنامه‌مان انتخاب می‌کنیم .



شکل ۱.۱: عکس ورودی

۱.۲.۱ بررسی کد

حال کد متلب این گام را بررسی می‌کنیم :

```
clc;
close all;

% STEP 2
filePath = 'D:\Full HD Desktop Wallpapers\amusement.jpg';
img = imread(filePath);
figure;
imshow(img)
% ----------------
```

حال به بررسی خطوط کد می‌پردازیم:

- اول مسیر فایل داده‌شده را مشخص میکنیم
- سپس با دستور *imread* آن فایل را می‌خوانیم
- و با دستور *imshow* فایل مرحله قبل را در خروجی نشان می‌دهیم.

۳.۱ گام سوم

در این گام با توجه به راهنمایی گفته شده در داک پروژه از تابع پیش تعریف‌شده `rgb2gray` استفاده می‌کنیم.

۱.۳.۱ بررسی کد

حال کد متلب این گام را بررسی می‌کنیم :

```
% STEP 3
grayImg = rgb2gray(img);
% ----------------
```

۲.۳.۱ انواع تصاویر

- `binary images` : این نوع تصاویر در یک ماتریس $m * n$ ذخیره می‌شوند که رنگ سیاه در آن معادل صفر و رنگ سفید معادل یک می‌باشد
- `indexed images`
- `grayscale images` : در این نوع تصاویر که در یک ماتریس $m * n$ ذخیره می‌شوند هر عضو آن شدت رنگ آن پیکسل را نشان می‌دهد. که بزرگترین عدد برای رنگ سفید و کوچکترین عدد برای رنگ سیاه می‌باشد که با توجه به تایپ داده ها می‌توانند رنج مختلفی برای خود بگیرند
  - [0,1] : single or double
  - [0,..,255] : uint8
  - [0,...,65535] : uint16
  - [-32768,...,32768] : int
- `truecolor images(rgb images)` : این نوع تصاویر در یک ماتریس سه بعدی یعنی $m * n * 3$ ذخیره می شوند که به جای ذخیره کردن عدد در مرحله قبل یک ماتریس ۳ عضوی متشکل از شدت رنگ های rgb را در خود نگه می دارد.

۴.۱ گام چهارم

تصویر تبدیل‌شده در مرحله قبل را در نشان داده و آن را ذخیره می‌کنیم.

۱.۴.۱ بررسی کد

```
% STEP 4
figure;
imwrite(grayImg, 'GrayImage.bmp');
imshow(grayImg);
% ----------------
```

شکل ۲.۱: تصویر خاکستری‌شده

۲.۴.۱ تفاوت فرمت های متفاوت عکس

- `jpg` : این نوع تصاویر از نوع فشرده‌سازی با اتلاف [۱] می‌باشند. در واقع حجم فایل را تا حد زیادی کاهش می‌دهد این نوع داده برای نگه‌داری و فرستادن مناسب است
- `png` : این نوع تصایر از نوع فشرده‌سازی بدون اتلاف [۲] می‌باشند. در واقع این نوع فشرده‌سازی مخالف فشرده‌سازی قبلی می‌باشد و می‌توان به بازسازی دوباره داده اصلی از فایل فشرده رسید.
- `bmp` : توسط شرکت Microsoft توسعه یافته‌شده است. حجم فایل بیشتری دارد و از نوع فشرده‌سازی بدون اتلاف می‌باشد.

[۱] lossy compression
[۲] lossless compression

- `tiff` : این نوع نسبت به نوع های‌های قبل بیشترین حجم فایل را دارد و از نوع فشرده‌سازی بدون اتلاف می‌باشد و معمولا در صنعت هنور و عکس‌های حرفه‌ای استفاده می‌شود

اگر بخواهیم از بین این فرمت ها یک فرمت را به عنوان فشرده‌سازی بدون اتلاف انتخاب کنیم کدام فرمت می‌باشد؟ `.tiff`

۵.۱ گام پنجم

با توجه به جستجوهای صورت‌گرفته تصویرها از نوع سیگنال توان نیستند اما از نوع سیگنال انرژی می‌باشند. در پی جستجوهای مختلف متوجه شده‌ام هر عضو ماتریس نشان‌دهنده انرژی آن می‌باشد که با جمع آن‌ها می‌توان به انرژی کل تصویر رسید. دو راه را برای این‌کار انجام داده‌ام یکی که طبق نکته ذکرشده بالا و دیگری طبق رابطه پارسوال اول تبدیل فوریه آن را محاسبه کرده و سپس انرژی این سینگال را در حوزه فرکانس حساب می‌کنیم. اما جواب های متفاوتی کسب کرده‌ام .

۱.۵.۱ بررسی کد

```
% STEP 5
totalEnergy = sum(grayImg(:));
display(totalEnergy);
% alternative
F = fft2(grayImg);
magImage = abs(F).^2;
energy = sum(magImage(:));
display(energy);
% ----------------------
```

```
totalEnergy : 190679380, energy : 5.1564e+16
```

۶.۱ گام ششم

در این گام نیز با توجه به تابع از پیش تعریف‌شده `imnoise` که مقادیر پیش‌فرض 0.01 برای واریانس و صفر برای میانگین در نظر گرفته‌شده است می‌توان به تصویر نویز اضافه کرد. و به عنوان ورودی دوم تابع نویز `gaussian` را به آن می‌دهیم و در ادامه نویز اضافه شده به تصویر را نمایش می‌دهیم.

۱.۶.۱ بررسی کد

```
% STEP 6
figure;
noisedPicture = imnoise(grayImg,'gaussian');
imshow(noisedPicture);
% ----------------
```

شکل ۳.۱: تصویر با نویز

۲.۶.۱ SNR(signal to noise ratio)

در این قسمت نوبت به محاسبه نسبت سینگال به نویز اضافه شده‌است.

`SNR` : میزان قدرت یک سیگنال نسبت به نویز پس‌زمینه آن می‌باشد. که با با واحد db اندازه‌گیری می‌شود.

$$SNR = 10 \cdot \log_{10} \frac{P_{signal}}{P_{noise}}$$

این نسبت می‌تواند هر عددی باشد که اعداد بزرگتر از صفر نشان دهنده این است که سطح سیگنال اصلی بیشتر از سطح noise است و اعداد کوچکتر برعکس. در واقع هر چه این نسبت بزرگتر باشد کیفیت سیگنال بهتر است.

۳.۶.۱ بررسی کد

```
% SNR
x = im2double(grayImg);
y = im2double(abs(noisedPicture - grayImg));
signalNoiseRatio = snr(x,y);
display(signalNoiseRatio);
% ----------------
```

در این بخش اگر دو سیگنال را به صورت مستقیم به تابع از پیش تعریف‌شده `snr` میدادم به `syntax error` برمی‌خوردم که می‌گفت باید ورودی های تابع `snr` باید `double` باشد در نتیجه هر دو سیگنال را به این نوع تبدیل کردم .
علت اینکه دو نویز را از هم کم کرده‌ام این بود که باید نویز خالص را از این روش به دست می‌آوردم.

```
signalNoiseRatio = 15.76
```

۷.۱ گام هفتم

در این گام سعی به گرفتن تبدیل فوریه از تصویر خاکستری شده و آن را طبق راهنمایی گفته شده رسم می‌کنیم.

۱.۷.۱ بررسی کد

```
% STEP 7
figure;
ft = fftshift(log(abs(fft2(grayImg))));
imshow(ft, []);
% ------------------------
```

شکل ۴.۱: تصویر تبدیل فوریه گرفته شده

۲.۷.۱ تحلیل عکس بالا

طی جست‌وجو های پی‌درپی نتایج دریافت‌شده از آنها را در قالب چند نکته اشاره می‌کنم:

- هر پیکسل در عکس تبدیل فوریه نشان دهنده یک موج سینوسی دو بعدی با فرکانس وابسته به فاصله از مرکز می‌باشد.
- نتیجه تصویر نشان می‌دهد که عکس حاوی تمامی فرکانس‌ها می‌باشد اما بزرگی آن هر چه از مرکز تصویر دورتر می‌شویم کمتر می‌شود و در نتیجه فرکانس بیشتر می‌شود.
- فرکانس‌های کمتر حاوی اطلاعات بیشتری نسبت به فرکانس های بالاتر هستند

۳.۷.۱ پاسخ به سوال‌ها

۱. **مرکز تصویر چرا از همه نقاط دیگر نورانی تر است؟** به دلیل اینکه بخش زیادی از عکس فرکانس کمتری دارند و مرکز تصویر حاوی اطلاعات بیشتری است. به تعبیری دیگر بخش زیادی از تصویر اصلی ما داری تغییرات کم فرکانس هستند یعنی رنگ آنها به یک‌باره از سفید به سیاه تغییر نمی‌کند

۲. **چرا هر چه از مرکز دورتر می‌شویم نقاط کم نورتر می شوند؟** چون این نقاط فرکانس بیشتر را در بر می‌گیرند در واقع این نفاط نشان‌دهنده تغییر ناگهانی در عکس اصلی می‌باشند.

۳. **بالا و پایین‌ترین فرکانس در تصویر کدام نقاط است؟** مرکز دارای کمترین فرکانس و هرچه از مرکز دورتر می‌شویم فرکانس بیشتر می‌شود.

۸.۱ گام هشتم

با کمک از مستند اصلی سایت متلب از روش `median` استفاده می‌کنم که این متد با کمک $3-by-3$ $neighborhood3$ پیاده‌سازی شده‌است.

۱.۸.۱ بررسی کد

```
% STEP 8
figure;
Kmedian = medfilt2(noisedPicture);
imshow(Kmedian);
% ----------------
```

شکل ۵.۱: تصویر با نویز رفع‌شده

۲.۸.۱ عملکرد رفع نویز

با توجه به راهنمایی گفته‌شده و تابع ازپیش‌تعریف‌شده $PSNR$ به بررسی این تابع می‌پردازیم . هرچه مقدار خروجی این تابع بیشتر باشد ما بهتر عمل کرده‌ایم

range PSNR

- برای تصویر ها و ویدئو های فشرده شده بین ۳۰ تا ۵۰ دسی‌بل می‌باشد که ۸ بیتی است.
- برای ۱۲ بیتی خروجی از ۶۰ به بالا خوب است
- مقدارهای قابل‌قبول برای انتقال‌های بی‌سیم بین ۲۰ تا ۲۵ می‌باشد.

```
% PEAK SNR
[peaksnr, outputSNR] = psnr(noisedPicture , grayImg);
fprintf('\n The Peak-SNR value is %0.4f', peaksnr );
% ----------------------------
```

```
The Peak-SNR value is 20.4105
```