دانشگاه تهران

دانشکده علوم و فنون نوین

پردازش سیگنال های تصویری دیجیتال

تمرین شماره دو

نام و نام خانوادگی	محمدحسین نژادهندی
شماره دانشجویی	۸۳۰۴۰۲۰۷۸
تاریخ ارسال گزارش	۰۲/۰۷/۲۳

فهرست گزارش سوال‌ها

سوال ۱ – رنگی کردن تصاویر سیاه و سفید

تصویر فایل Mand.tiff تصویری است که از یک شبکه حسگر دارای الگوی بایر گرفته شده است.

الف) برنامه ای بنویسید که با استفاده از روش نزدیکترین همسایه تصویر رنگی حاصل را تولید نماید. تصویر حاصل را ترسیم نمایید.

ب) برنامه ای بنویسید که با استفاده از درون یابی خطی تصویر رنگی حاصل را تولید نماید. تصویر حاصل را ترسیم نمایید.

ج) نتیجه قسمت الف و ب را با هم و همچنین با حالت استفاده از تابع demosaic در متلب مقایسه نمایید. چه تفاوتی را احساس می‌کنید.

پاسخ :

الف) در ابتدا ما تصویر Mand.tiff را با استفاده از دستور imread از دیسک خوانده و در متغیر mandImage ذخیره می‌کنیم. سپس به تعیین اندازه تصویر میپردازیم. تصویر رنگی حاصل اندازه ای مشابه تصویر اصلی Mand.tiff دارد. ما یک متغیر بنام colorImage ایجاد نموده که ابعاد آن به اندازه تصویر Mand.tiff است. در اینجا ما تصور میکنیم که تصویر رنگی سه کانال رنگی دارد، بنابراین تصویر رنگی ما ۳ بعد دارد که عبارتند از : ارتفاع، عرض و کانال رنگ.

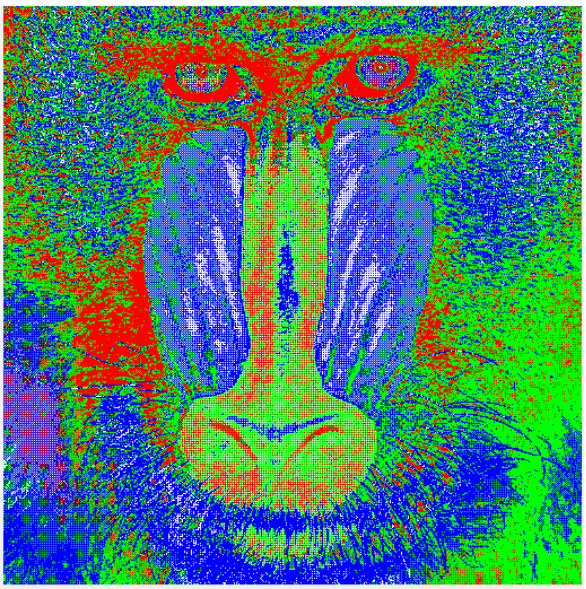
گام بعدی ترسیم تصویر رنگی حاصل با استفاده از روش نزدیکترین همسایه است. برای هر پیکسل در تصویر Mand.tiff، ما مقدار پیکسل را بررسی میکنیم و یک رنگ متناظر با آن مقدار را تعیین میکنیم. در اینجا ما چندین بازه مقدار پیکسل تعریف کرده ایم و برای هر بازه یک رنگ (قرمز،سبز،آبی و سفید) انتخاب مینماییم.

سپس نوبت به نمایش تصویر رنگی حاصل می‌رسد که برای این مهم از دستور imshow استفاده می‌کنیم.

کد را در خطوط پایین تر مشاهده می‌نمایید:

```
mandImage = imread('Mand.tiff');

colorImage = uint8(zeros(size(mandImage, 1), size(mandImage, 2), 3));

% Generate the Color Image Using the Nearest Neighbor Method
for i = 1:size(mandImage, 1) % iterate on rows :)
    for j = 1:size(mandImage, 2) % iterate on columns :)
        pixelValue = mandImage(i, j);
        if pixelValue < 64
```

```
            color = [255, 0, 0]; % Red
        elseif pixelValue < 128
            color = [0, 255, 0]; % Green
        elseif pixelValue < 192
            color = [0, 0, 255]; % Blue
        else
            color = [255, 255, 255]; % White
        end

        colorImage(i, j, :) = color;
    end
end

imshow(colorImage);
```

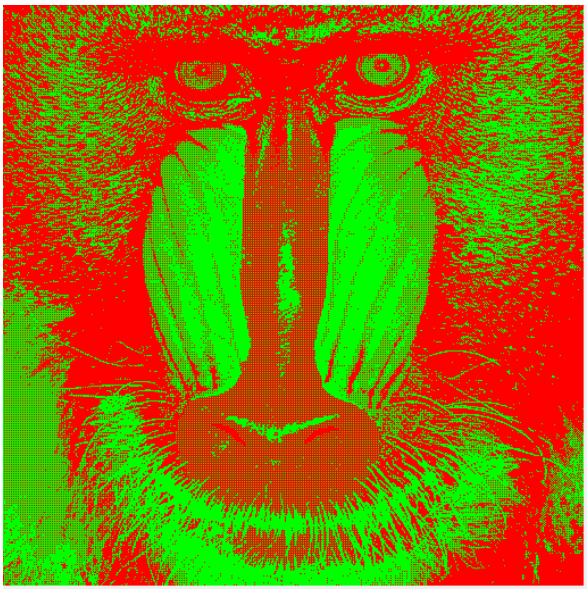


شکل ۱: تصویر بدست آمده از خروجی در قسمت الف

ب) در این مرحله همانطور که در کد مشاهده می‌نمایید، تصویر Mand.tiff را تبدیل به یک تصویر رنگی میکنیم و بر اساس مقادیر پیکسل ها، رنگ های مختلفی را به پیکسل ها اختصاص میدهیم. این کد از چند بازه مقداری برای تعیین رنگ استفاده کرده است. میتوانیم این بازه‌ها را تغییر داده تا به نتیجه دلخواه برسیم.

```
mandImage = imread('Mand.tiff');

[rows, cols] = size(mandImage);

R = zeros(rows, cols);
G = zeros(rows, cols);
B = zeros(rows, cols);

for i = 1:rows
    for j = 1:cols
```

```
        if mandImage(i, j) < 128
            R(i, j) = 255;
        else
            G(i, j) = 255;
        end
    end
end

colorImage = cat(3, R, G, B);

imshow(colorImage);
```



شکل۲ : تصویر بدست آمده از خروجی در قسمت ب

ج) تفاوت بین این ۳ روش به دلیل معماری و الگوریتم های مختلفی که در هر یک از آنها استفاده میشود ممکن است به نتایج متفاوتی منجر شود. همچنین این نتایج ممکن است بسته به ویژگی های تصویر Mand.tiff تغییر کند. در ادامه مقایسه ای کلی بین این ۳ روش ارائه می‌شود:

- استفاده از الگوریتم نزدیکترین همسایه :
  - این الگوریتم بسادگی هر پیکسل سیاه و سفید را به یک مقدار رنگی تبدیل میکند.
  - نتیجه معمولا یک تصویر رنگی با رنگ های ساده و نادرست خواهد بود.
- استفاده از رونیابی خطی:
  - این روش با استفاده از الگوی بایر و درونیابی خطی مقادیر پیکسل ها را به تصویر رنگی تبدیل می‌کند.

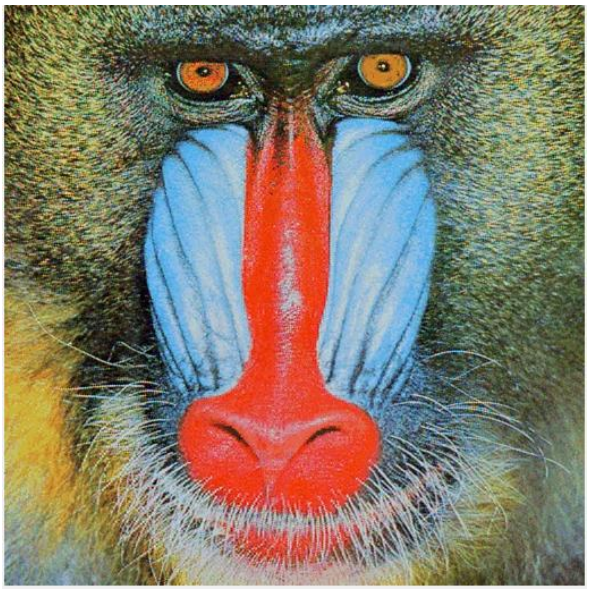
- نتیجه بهتر از الگوریتم نزدیکترین همسایه خواهد بود و رنگ ها به صورت صحیح تر بازتولید می‌شود.
- اما این روش نیاز به پیاده سازی دقیق الگوی بایر دارد.

- استفاده از تابع demosaic:
  - تابع demosaic با استفاده از الگو های رنگی مختلف مانند ayer تصویر سیاه و سفید را به تصویر رنگی تبدیل می‌کند.
  - نتیجه نزدیک به رنگ های واقعی خواهد بود و بصورت خودکار تبدی انجام می‌شود.
  - این روش نیاز به پیاده سازی دقیق الگو های رنگی ندارد و به طور کلی نتیجه بهتری نسبت به دو روش قبلی دارد.

پس در کل استفاده از تابع demosaic به طور معمول نتایج بهتری در تبدیل تصویر سیاه و سفید به تصویر رنگی ارائه می دهد. اگر چه درونیابی خطی نیز نتایج بهتری نسبت به الگوریتم نزدیکترین همسایه ارائه میدهد اما پیاده سازی الگوی بایر و درونیابی خطی به اندازه تابع demosaic پیچیده تر است.

در ادامه کد قسمت ج را مشاهده میکنید:

```
mandImage = imread('Mand.tiff');
colorImage = demosaic(mandImage, 'grbg');
imshow(colorImage);
```



شکل ۳ : تصویر بدست آمده از خروجی در قسمت ج

سوال ۲ – نمایش تصویر از فایل باینری

فایل باینری imgdrv را در نظر بگیرید.

الف) با استفاده از دستور fread، داده این فایل را به صورت کاراکتر بدون علامت ۸ بیتی بخوانید. تصویر دارای ۴۳۵ خط است و طور هر خط برابر ۵۸۰ می‌باشد. تصویر را نمایش دهید.

ب) در تصویر قبل، بجای ۸ بیت، از ۵، ۴، ۳ ، ۲ و ۱ بیت برای نمایش هر نقطه استفاده نمایی. برای حفظ کیفیت تصویر، چند بیت کافی است؟

پاسخ :

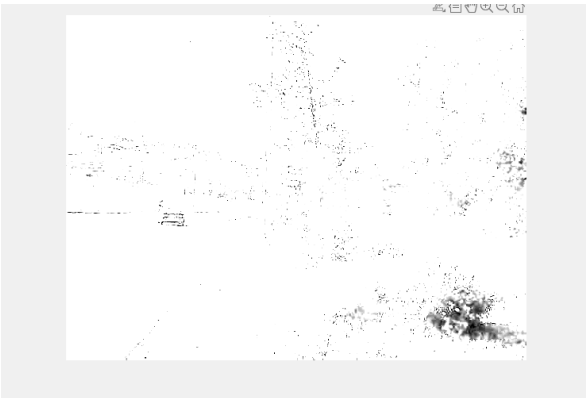
الف) در این قسمت با استفاده از دستور fopen فایل باینری را برای خواندن باز کردیم. سپس از موفقیت باز شدن آن باخبر میشویم و پس از آن به سراغ خواندن داده ها به صورت کاراکتر بدون علامت ۸ بیتی با استفاده از دستور fread میرویم. سپس داده ها را با اسفاده از تابع $uint^{8}$ به تصویر تبدیل میکنیم و پس از آن با استفاده از تابع imshow آن را نمایش میدهیم. کد توسعه داده شده در ادامه قابل مشاهده است:

```
fileID = fopen('imgdrv.txt', 'rb');

if fileID == -1
    error('نمی‌توان فایل را باز کرد.');
end

numLines = 435;
lineLength = 580;

data = fread(fileID, [lineLength, numLines], 'uchar=>char')';

fclose(fileID);

imageData = uint8(data);

imshow(imageData);
title('Image 😊')
```

همچنین تصویر تولید شده نیز در ادامه مشاهده می‌فرمایید:

شکل ۴ : تصویر بدست آمده از خروجی در قسمت الف

ب) درقسمت ب به دو روش راه حل هایی یافت شده است که یکی با استفاده از تابع bitshift است و دیگری با استفاده از تقسیم است. مراحل خواندن فایل که تکراری است و بهنگام نمایش در حالت اول از دستور imshow(uint8(bitshift(data, 8 - n)), []); استفاده میکنیم که nبیانگر مقدار چند بیت برای نمایش است و میتوان اعداد تا ۵ را قرار داد که بعنوان مثال نتیجه $n=^{5}$ خواهد بود:



شکل ۵ : تصویر بدست آمده از خروجی در قسمت ب حالت اول

کد این حالت را در ادامه مشاهده میکنید:

```
file_path = 'imgdrv.txt';

fid = fopen(file_path, 'rb');

if fid == -1
    error('خطا در باز کردن فایل.');
end

num_rows = 435;
num_columns = 580;

data = fread(fid, [num_columns, num_rows], 'uint8')';

fclose(fid);

imshow(uint8(bitshift(data, 8 - 5)), []);
```

اما در حال دوم به جای استفاده از بیت شیفت ، از تقسیم استفاده کرده ایم :

```
file_path = 'imgdrv.txt';

fid = fopen(file_path, 'rb');

if fid == -1
    error('خطا در باز کردن فایل.');
end

num_rows = 435;
num_columns = 580;

data = fread(fid, [num_columns, num_rows], 'uint8')';

fclose(fid);

imshow(data, []);

bits_per_pixel = 2;

new_image = uint8(double(data) / 2^(8 - bits_per_pixel));

figure;
imshow(new_image, []);
```

همانطور که مشاهده میفرمایید با تغییر مقدار متغیر `bits_per_pixel` میتوان به تصاویری دست یافت. در این حالت اگر مقدار متغیر برابر ۵ باشد تصویر حاصل برابر زیر خواهد بود :

شکل۶ : تصویر بدست آمده از خروجی در قسمت ب حالت دوم

هانطور که مشاهده میفرمایید در حالت دوم حتی با عدد ۳ و یا حتی کمتر نیز میتوان تصاویری بدست آورد . اما در حالت اول حتی با ۵ بیت هم نمیتوان تصویری دقیق و خوانا بدست آورد.

با تشکر از توجه تان

نژادهندی

۸۳۰۴۰۲۰۷۸