دانشگاه تهران

دانشکده علوم و فنون نوین

پردازش سیگنال های تصویری دیجیتال

تمرین شماره دو

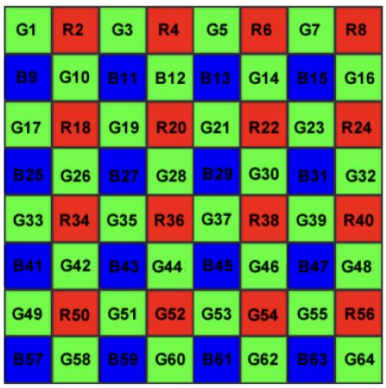
نام و نام خانوادگی	فاطمه چیت ساز
شماره دانشجویی	830402092
تاریخ ارسال گزارش	21 مهر 1402

فهرست گزارش

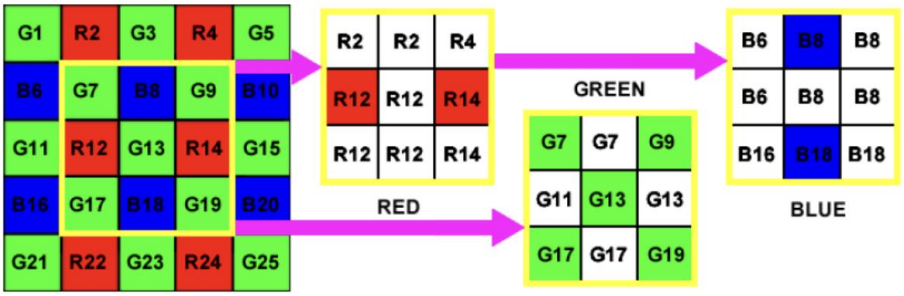
سوال 1 – الگوی بایر

برای قسمت الف از ما خواسته شده که تصویری که با الگوی بایر گرفته شده است را خوانده سپس آن تصویر را به تصویر رنگی تبدیل کنیم

میدانیم برای انجام این فرایند روش های متفاوتی وجد دارد که یکی از این روش ها استفاده از روش نزدیکترین همسایه است که در این روش ما مقادیر قرمز یا سبز یا آبی یک پیکسل را میدانیم حال باید بقیه مقادیر آن پیکسل را حدس بزنیم برای این کار پیکسل های اطراف آن پیکسل را نگاه میکنیم و با استفاده از آنها بقیه رنگ های پیکسل انتخابی را حدس میزنیم در انتهای کار ما سه تصویر داریم که یکی تمامی پیکسل های آن قرمز دیگری سبز و بعدی آبی است وقتی این سه تصویر را روی همدیگر قرار دهیم تصویر رنگی ما به دست میاید .



Nearest Neighbor



کد این کار :

برای نوشتن این فرایند در متلب ما ابتدا فایل تصویر خود را با imread میخوانیم و آرایه ای از پیکسل های خود را داریم حال وقتی به هر خانه از این ارایه میرسیم باید ببینیم طبق الگوی بایر که در صورت سوال داده شده ما در چه خانه ای هستیم (سبز یا قرمز یا آبی)

حال با توجه به شما خانه ما با استفاده از خانه های اطراف آن پیکسل بقیه رنگ های آن را به دست میاوریم پس به وسیله این کار ما red channel و blue channel و green channel خود را میسازیم

```
% Perform demosaicing using nearest neighbor interpolation
for row = 2:rows-1
    for col = 2:cols-1
        % Determine the position in the Bayer mask
        maskRow = mod(row, 2) + 1;
        maskCol = mod(col, 2) + 1;

        % Calculate the interpolated color values
        if bayerMask(maskRow, maskCol) == 1
            blueChannel(row, col) = bayerImage(row, col);
            greenChannel(row, col) = bayerImage(row-1, col);
            redChannel(row, col) = bayerImage(row-1, col-1);
        elseif bayerMask(maskRow, maskCol) == 2
            greenChannel(row, col) = bayerImage(row, col);
            if(maskRow== 2)
            blueChannel(row, col) = bayerImage(row-1, col);
            redChannel(row, col) = bayerImage(row, col-1);
          else
            blueChannel(row, col) = bayerImage(row, col-1);
            redChannel(row, col) = bayerImage(row-1, col);
            end
        elseif bayerMask(maskRow, maskCol) == 3
            redChannel(row, col) = bayerImage(row, col);
            greenChannel(row, col) = bayerImage(row-1, col);
            blueChannel(row, col) = bayerImage(row-1, col-1);
        end
    end
end
```

حال که سه مجموعه رنگی خود را ساختیم زمان آن رسیده که آنها را ترکیب کنیم تا عکس رنگی نهایی تولید شود

برای این کار کافیست که با استفاده از cat این سه مجموعه را ترکیب کرده و در نهایت با imshow تصویر رنگی خود را نمایش دهیم

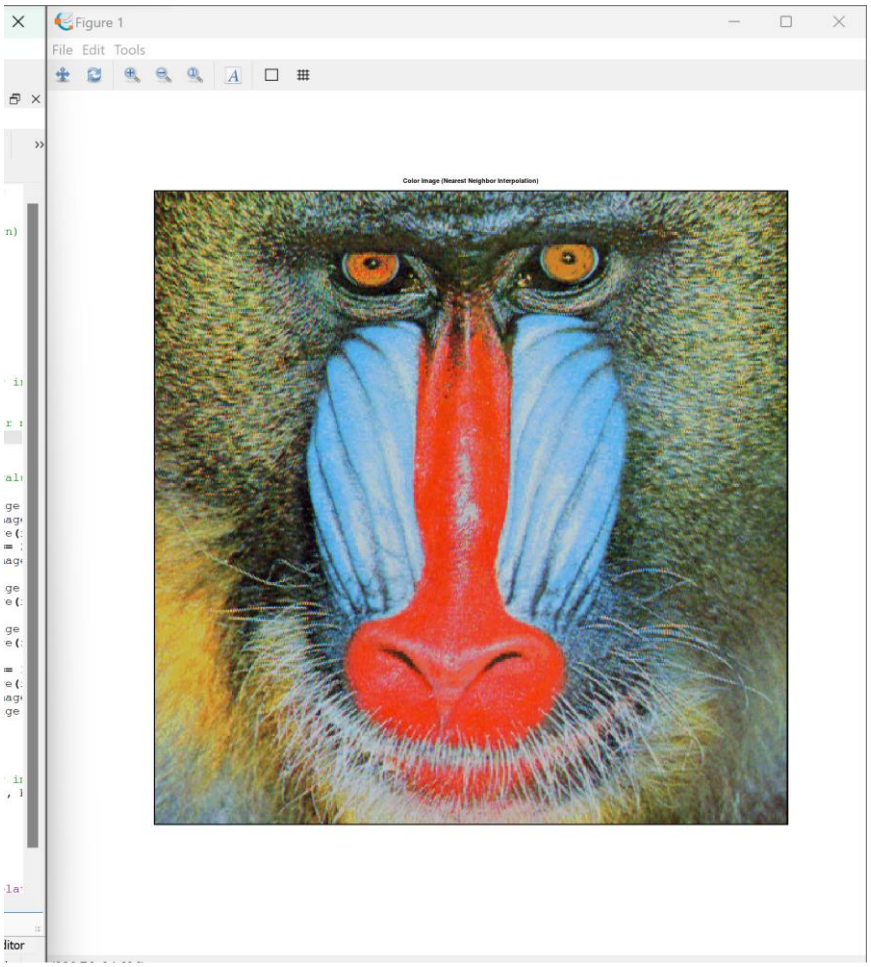
```
% Combine color channels into a single color image
colorImage = cat(3, redChannel, greenChannel, blueChannel);

% Convert to uint8 format (0-255)
colorImage = uint8(colorImage);

% Display the resulting color image
imshow(colorImage);
title('Color Image (Nearest Neighbor Interpolation)');
```

فایل کد در پوشه شماره یک و سپس پوشه A با نام q.m موجود است

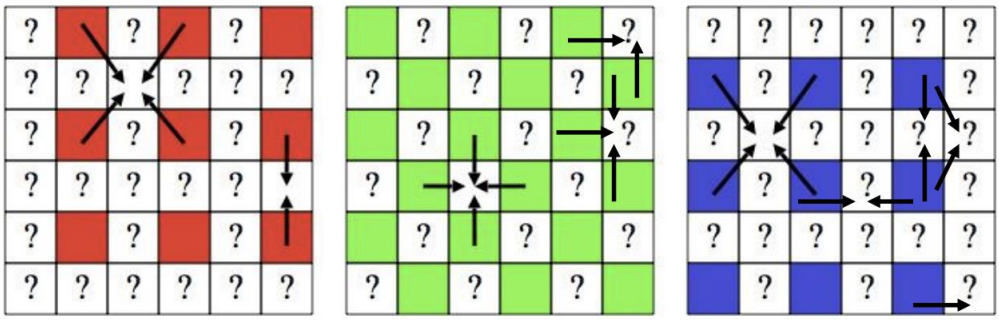
نتیجه کد:



برای قسمت بعدی ما باید با استفاده از درونیابی خطی همین فرایند را جلو ببریم در درون یابی خطی مشابه نزدیکترین همسایه ما در هر پیکسل یکی از رنگ ها را داریم و میخواهیم بقیه رنگ ها را پیدا کنیم اما اینجا برای این کار به جای اینکه یکی از همسایه ها را جایگزین کنیم میانگینی از همسایگان آن پیکسل میگیریم تا آن رنگی که نداریم را پیدا کنیم

Bilinear Interpolation

Averaging the four (or less) neighboring values



کد این قسمت در پوشه یک و سپس پوشه B با نام q2.m موجود است

کد این قسمت :

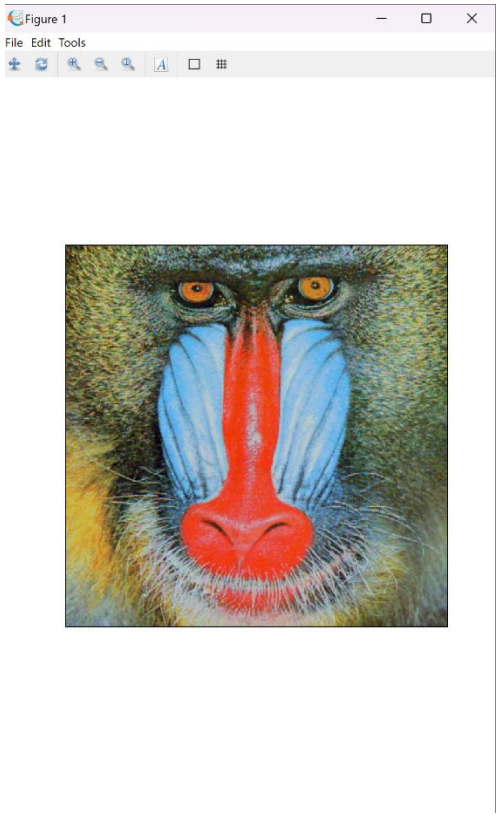
برای پیاده سازی این قسمت نیز ابتدا فایل را با imread خوانده و سپس باید روی همه پیکسل های فایل یک for بزنیم وقتی به هر پیکسل رسیدیم با توجه به موقعیت آن و پترن بایر که در صورت سوال داده شده میدانیم پیکسل ما چه رنگی را دارد حال بقیه رنگ های آن را با توجه به موقعیت آن از میانگین رنگ های اطراف بدست میاوریم

```
% Apply bilinear interpolation to the green channel
for i = 2:height-1
    for j = 2:width-1
        if mod(i, 2) == 1 % Odd rows
            if mod(j, 2) == 1 % Odd columns
                greenChannel(i, j) = bayerImage(i, j);
                redChannel(i, j) = (bayerImage(i, j - 1) + bayerImage(i, j + 1)) / 2;
                blueChannel(i, j) = (bayerImage(i - 1, j) + bayerImage(i + 1, j)) / 2;
            else % Even columns
                greenChannel(i, j) = (bayerImage(i, j - 1) + bayerImage(i, j + 1) + bayerImage(i - 1, j) + bayerImage(i + 1, j)) / 4;
                redChannel(i, j) = bayerImage(i, j);
                blueChannel(i, j) = (bayerImage(i - 1, j - 1) + bayerImage(i - 1, j + 1) + bayerImage(i + 1, j - 1) + bayerImage(i + 1, j + 1)) / 4;
            end
        else % Even rows
            if mod(j, 2) == 0 % Even columns
                greenChannel(i, j) = bayerImage(i, j);
                blueChannel(i, j) = (bayerImage(i, j - 1) + bayerImage(i, j + 1)) / 2;
                redChannel(i, j) = (bayerImage(i - 1, j) + bayerImage(i + 1, j)) / 2;
            else % Odd columns
                greenChannel(i, j) = (bayerImage(i, j - 1) + bayerImage(i, j + 1) + bayerImage(i - 1, j) + bayerImage(i + 1, j)) / 4;
                redChannel(i, j) = (bayerImage(i - 1, j - 1) + bayerImage(i - 1, j + 1) + bayerImage(i + 1, j - 1) + bayerImage(i + 1, j + 1)) / 4;
               blueChannel(i, j) = bayerImage(i, j);
            end
        end
    end
end
```

بعد از انکه هر سه گروه رنگی خود را به دست اوردیم زمان آن رسیده که آنها را ترکیب کنیم و عکس رنگی خود را بدست بیاوریم

```
% Combine the channels to create the full-color image
colorImage = cat(3, redChannel, greenChannel, blueChannel);
% Convert to uint8 format (0-255)
t = uint8(colorImage);
% Display the resulting color image
imshow(t);
```

تصویر خروجی این کد :



در قسمت آخر این سوال از ما خواسته شده که همین کار را با تابع demosaic انجام دهیم و سه نتیجه را با هم مقایسه کنیم برای استفاده از تابع demosaic نیز ابتدا ما فایل را میسازیم سپس ارایه تصویر خود را به demosaic میدهیم باید توجه کرد چون پترنی که در صورت سوال داده شده به صورت GRBG است ما این را باید به تابع demosaic خود به عنوان ورودی بدهیم

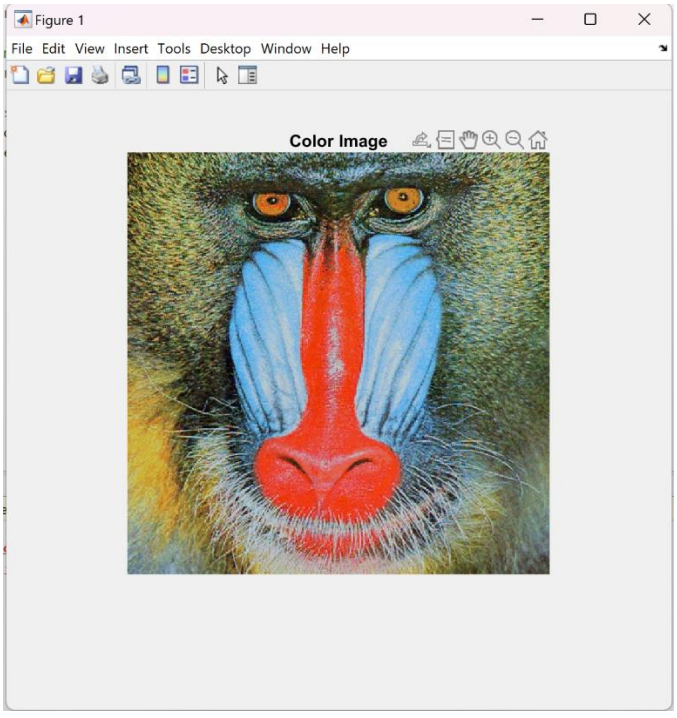
* کد این قسمت در پوشه یک سپس پوشه A با نام q3.m میباشد

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

% Demosaic the image using the AHD algorithm
colorImage = demosaic(bayerImage, "grbg");

% Display the color image
imshow(colorImage);
title('Color Image');
```

خروجی کد :



پس از بررسی نهایی عکس متوجه میشویم کیفیت عکسی که با درونیابی خطی و demosaic بدست آمده از نزدیکترین ترین همسایه بهتر است

سوال ۲ – کاهش رزولوشن عمق بیت

در قسمت اول این سوال به ما یک imgdrv داده شده است که شامل محتوای باینری است و شامل 435 سطر و 580 ستون میباشد ما باید ابتدا این فایل را خوانده و سپس به عنوان یک تصویر نمایش دهیم برای این کار از تابع fread استفاده کرده و فایل خود را که با fopen خوانده ایم را به آن میدهیم حال برای مشخص کردن سایز خروجی خود باید تعداد ستون ها و سطر های خود را به عنوان ورودی به fread پاس دهیم برای اینکه مشکلی در خواندن تصویر به وجود نیاید ما فایل تصویر را به صورت {تعداد سطر و تعداد ستون } میخوانیم سپس آن را پریم میکنیم تا تصویر درست بدست آید نکته بعدی که در صورت سوال مطرح شده آن است که ما فایل را به صورت هشت بیتی بی علامت بخوانیم برای این کار ما uint8 => unit8 را به عنوان یکی از پرامتر های fread میدهیم تا هشت بیتی بدون علامت داشته باشیم

* کد این سوال در پوشه دو سپس پوشه A با نام q2.m موجود است

r - C:\Users\ASUS\Desktop\Hw2\2\A\q2.m

m | q1.m | q2.m | q3.m | q2.m | +

```
% Image dimensions
numRows = 435;
numCols = 580;

% Open the binary file for reading
fid = fopen('imgdrv.txt', 'rb');

% Read the data directly into the image matrix
imageMatrix = fread(fid, [numCols numRows], 'uint8=>uint8')';

% Close the file
fclose(fid);

% Display the image
imshow(imageMatrix, []);
title('Binary Image');
```

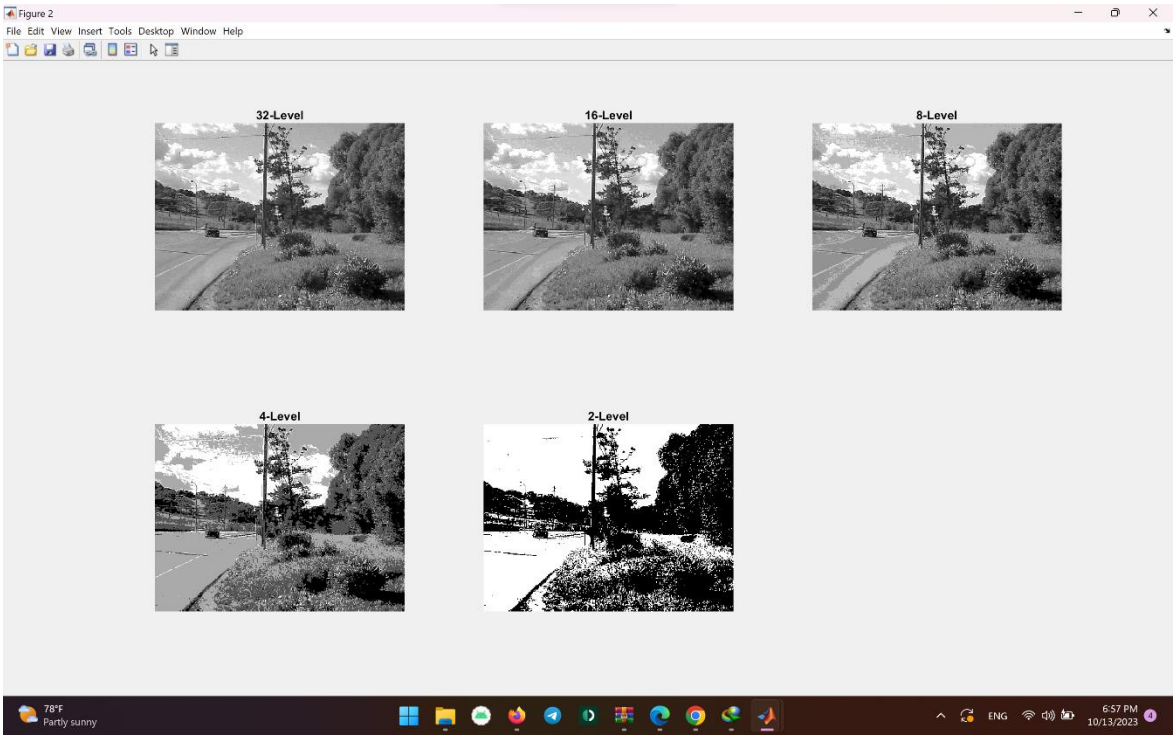
تصویر خروجی:

در قسمت بعدی از ما خواسته شده که کاهش رزولوشن عمق بیت انجام دهیم یعنی هر یک از پیکسل های خود را به جای نمایش با هشت بیت بی علامت با پنج و چهار و ... بیت نمایش دهیم و تاثیر آن را کیفیت ببینیم

برای این کار میتوانیم از کدی که در کلاس برای این کار معرفی شد استفاده کنیم تا به وسیله فرمول زیر عمق بیت های تصویر خود را عوض کنیم

```
% Open the binary file for reading
fid = fopen('imgdrv.txt', 'rb');
I_8bit =  fread(fid,[numCols numRows])';
f2j = @(f,J) max(min(round(J*(f-1/(2*J))),J-1),0);
j2r = @(j,J) (j+1/2)/J;
figure
cnt = 0;
for J=[32 16 8 4 2]
I1 = j2r(I_8bit,256);
I2 = f2j(I1,J);
cnt = cnt+1;
subplot(2,3,cnt);
imshow(I2,[0 J-1])
title(sprintf('%d-Level',J))
end
```

به وسیله این روش ما پیکسل های خود را به تعداد بیتی که میخواهیم میرسانیم و حال برای پنج و چهار و سه و... بیت تصاویر زیر بدست میآید



در سه لول اول که برای پنج بیت و چهار بیت و سه بیت است کیفیت تصویر تقریبا حفظ شده ولی وقتی به سمت دو بیت و یک بیت رفته ایم کیفیت تصویر به شدت پایین آمده است

- کد این قسمت در فایل شماره دو و سپس پوشه B با نام q22.m موجود است