Gliwice, 22.04.2022

Przetwarzanie Obrazów Cyfrowych

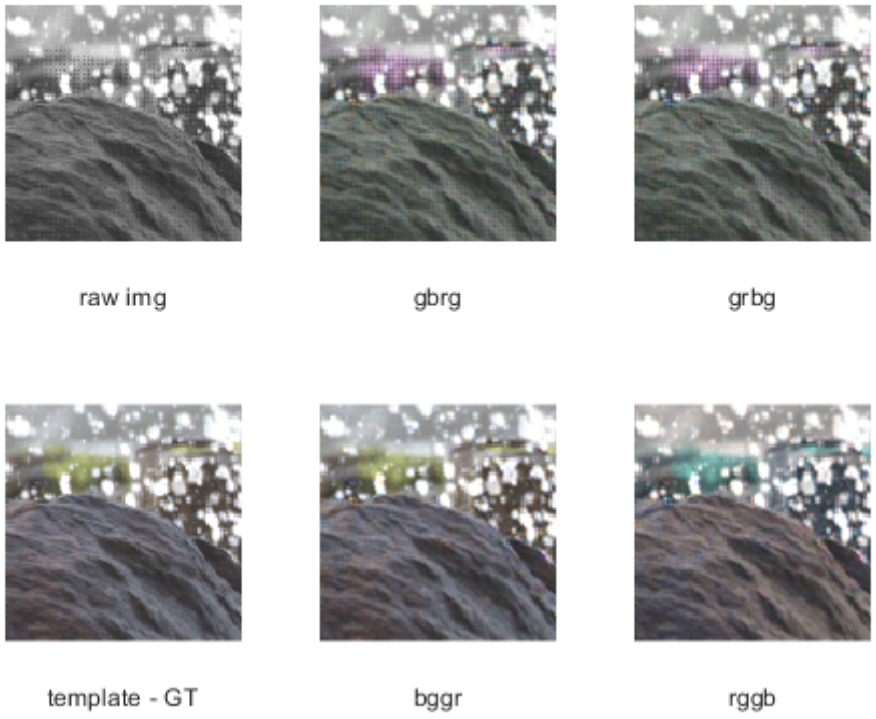
Akwizycja obrazów barwnych – demosaicing

Politechnika Śląska

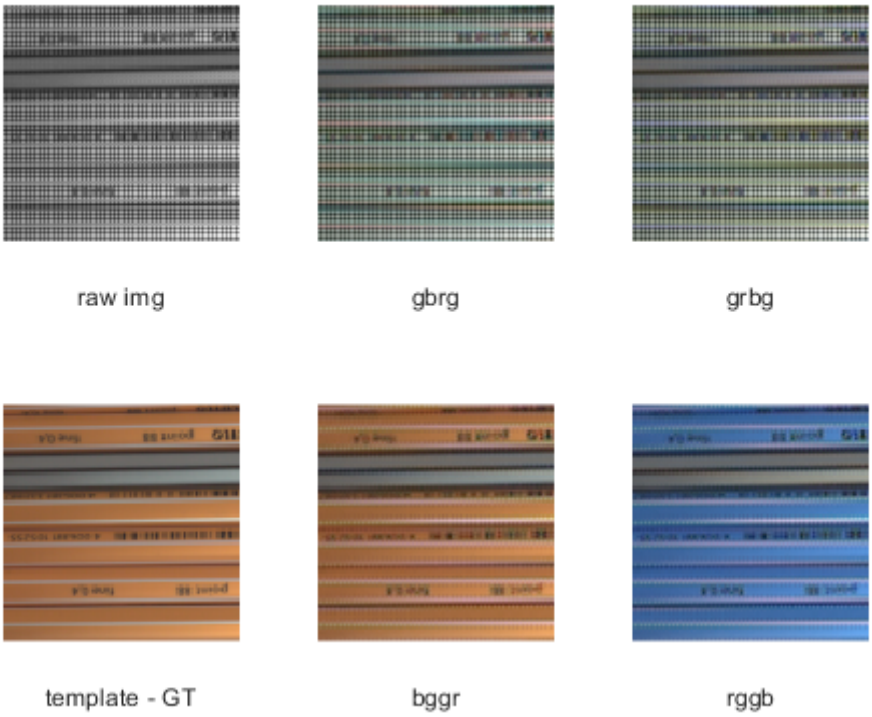
Jakub Zeifert

1 Identyfikacja układu matrycy Bayera

- Proszę wybrać kilka "surowych" obrazów ze zbioru testowego. Podczas normalnej pracy oprogramowanie kamery lub aparatu cyfrowego ze konkretnej matrycy i układ pikseli jest z góry zdefiniowany. Jednak w naszym przypadku obrazy testowe nie posiadają tej informacji. Przed przystąpieniem do interpolacji musimy więc zidentyfikować układ pikseli w przetwarzanych obrazach. Najprościej skorzystać z polecenia Matlab służącego do interpolacji obrazów mozaikowych - demosaic. W przypadku poprawnego doboru układu pikseli uzyskamy obraz o poprawnych barwach, pozbawiony artefaktów "mozaikowych"
- W odpowiedzi proszę przesłać odpowiednią sekwencję poleceń Matlaba i uzyskane układ pikseli.



Rysunek 1: Pierwszy wybrany obraz



Rysunek 2: Drugi wybrany obraz

Układ użytej matrycy to **BGGR**. Zdecydowanie widać, że pozostałe znacznie odróżniają się dokładnością barw co do oryginalnego obrazu.

Kod programu:

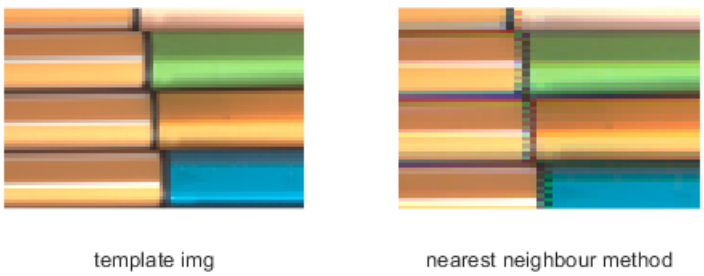
```
clc;clear;

raw_img_path = 'cfa.png';
color_img_path = 'srgb.png';


raw_img = imread(raw_img_path);  %upload raw img
color_img = imread(color_img_path);  %upload color img

d1 = demosaic(raw_img,'gbrg');    %using demosaic function for each matrice combination
d2 = demosaic(raw_img,'grbg');
d3 = demosaic(raw_img,'bggr');
d4 = demosaic(raw_img,'rggb');

figure(); %plotting results
title('Bayer Matrice');
subplot (2,3,1); imshow(raw_img(80:180,90:190)); xlabel('raw img');
subplot (2,3,2); imshow(d1(80:180,90:190,:)); xlabel('gbrg');
subplot (2,3,3); imshow(d2(80:180,90:190,:)); xlabel('grbg');
subplot (2,3,4); imshow(color_img(80:180,90:190,:)); xlabel('template - GT')
subplot (2,3,5); imshow(d3(80:180,90:190,:)); xlabel('bggr');
subplot (2,3,6); imshow(d4(80:180,90:190,:)); xlabel('rggb');

% imwrite(d3, 'demosaic.png'); %saving correct result in file

[PSNR] = psnr(d3, color_img) %PSNR check
```

zad1.m

2 Interpolacja metodą najbliższego sąsiada

- Zaimplementuj metodę interpolacji obrazów mozaikowych metodą najbliższego sąsiada (proste kopiowanie pikseli),
- zaprezentuj jej działanie na kilku przykładowych plikach,
- prześlij działający skrypt Matlaba i kilka wyników jego działania - wybierz tylko zbliżenia istotnych detali obrazu, zaobserwuj artefakty powstające na ostrych krawędziach obrazu

Rysunek 3: skały:
PSNR = 25.2255



Rysunek 4: flamastry:
PSNR = 25.4501

W przypadku tej metody zdecydowanie można zauważyć charakterystyczne ząbkowanie przy przejściu między krawędziami. Jeżeli nie planujemy mocno przybliżać zdjęcia lub ukazywać tylko wycinka,

to metoda powinna być wystarczająca. Jeśli zależy nam na dokładności przy krawędziach ta metoda nie daję za dobrych wyników.

Kod programu:

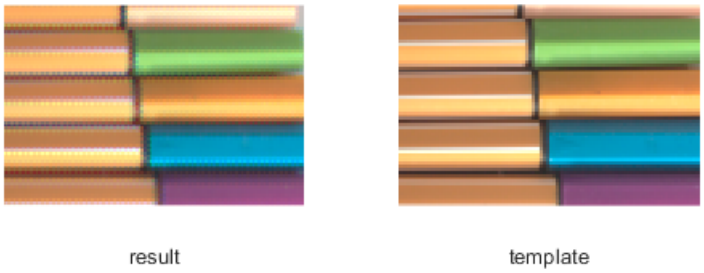
```
clear; clc;

raw_img_path = 'cfa.png';
color_img_path = 'srgb.png';


raw_img = imread(raw_img_path);  %upload raw img
color_img = imread(color_img_path);  %upload color img
[column,rows] = size(raw_img);
final_result = uint8(zeros(column,rows,3));
test1 = separate_color(raw_img, 'r');
test2 = separate_color(raw_img, 'g');
test3 = separate_color(raw_img, 'b');

final_result(:,:,1) = nearest_neighbour(separate_color(raw_img, 'r'),'r');
final_result(:,:,2) = nearest_neighbour(separate_color(raw_img, 'g'),'g');
final_result(:,:,3) = nearest_neighbour(separate_color(raw_img, 'b'),'b');
final_result=cast(final_result,'uint8');



figure;
%     subplot (1,3,1);
%     imshow(raw_img);
%     xlabel('raw img')
    subplot (1,2,1);
    imshow(color_img);
    xlabel('template img')
    subplot (1,2,2);
    imshow(final_result);
    xlabel('nearest neighbour method')


[PSNR] = psnr(final_result, color_img)

imwrite(final_result,'neighbour.png');
%% Functions

function separated_img  = separate_color(img, color) %BGGR system
    [row,col] = size(img);
    separated_img = uint8(zeros(row,col));
    for i = 1:2:row
        for j = 1:2:col
            if color == 'b'
                separated_img(i,j) = img(i,j);
            elseif color == 'g'
                separated_img(i,j+1) = img(i,j+1);
                separated_img(i+1,j) = img(i+1,j);
            elseif color == 'r'
                separated_img(i+1,j+1) = img(i+1,j+1);
            else
                fprintf('error occured, wrong color imput!!!');
                break
            end
        end
    end
end
```

```
function result_img = nearest_neighbour(img,color)
    [row,col] = size(img);
    result_img = double(zeros(row,col));
    for i=1:2:(row-1)
            for j=1:2:(col-1)
                val_check = double(img(i:i+1,j:j+1));
                if color == 'r'
                    result_img(i:i+1,j:j+1) = val_check(2,2);
                elseif color == 'g'
                    result_img(i,j) = val_check(1,2); %/2 + val_check(2,1)/2;
                    result_img(i,j+1) =val_check(1,2);
                    result_img(i+1,j) = val_check(2,1);
                    result_img(i+1,j+1) = val_check(2,1);
                elseif color == 'b'
                    result_img(i:i+1,j:j+1) = val_check(1,1);
                else
                    fprintf("error");
                end
            end
    end
end
```

zad2.m

3 Interpolacja biliniowa

- Zaimplementuj interpolację biliniową dla obrazów mozaikowych,
- przetestuj jej działanie na kilku przykładowych plikach,
- prześlij działający skrypt Matlaba i kilka wyników jego działania - wybierz tylko zbliżenia istotnych detali obrazu.



Rysunek 5: flamastry:
PSNR = 28.5290

Rysunek 6: skały:
PSNR = 29.3396

Pierwsze co mocno się rzuca w oczy to wzrost PSNR względem poprzedniej metody, co świadczy o lepszej jakości odwzorowania barw. Nie występuje już ząbkowanie, ale jest niestety występuje lekko rozmazanie między przejściami oraz obraz interpolowany posiada nadal sporo niedokładności w

odwzorowywaniu kolorów. Dobrze można to zauważyć na skałach, gdzie widać sporo pikseli bardziej fioletowych czy zielono-zółtych, co zdecydowanie nie powinno się znajdować. Można również zauważyć rozmycie obrazu w stosunku do oryginału. Mimo to nadal zdjęcie w przybliżeniu prezentuje się lepiej niż interpolowane metodą najbliższego sąsiada.

Kod programu:

```
clear; clc;

% raw_img_path = 'cfa.png';
% color_img_path = 'srgb.png';

raw_img_path = 'cfa_2.png';
color_img_path = 'srgb_2.png';


raw_img = imread(raw_img_path);  %upload raw img
color_img = imread(color_img_path);  %upload color img
[row,col] = size(raw_img);


r = double(separate_color(raw_img, 'r'));
b = double(separate_color(raw_img, 'b'));
g = double(separate_color(raw_img, 'g'));

r_channel = r;
b_channel = b;
g_channel = g;

% R
r_channel(1:2,1:2) = r(2,2); %filling first corner of a picture
for i=2:1:row-1
    for j=2:1:col-1
        if r_channel(i,j) == 0
            if mod(j,2) == 0 && mod(i,2) == 1
                r_channel(i,j) = r(i-1,j)/2 + r(i+1,j)/2;
            elseif  mod(j,2) == 1 && mod(i,2) == 0
                r_channel(i,j) = r(i,j+1)/2 + r(i,j-1)/2;
            elseif mod(j,2) == 1 && mod(i,2) == 1
                r_channel(i,j) =  r(i-1,j-1)/4 + r(i+1,j-1)/4 + r(i+1,j+1)/4 + r(i+1,j
                    -1)/4;
            end
        end
    end
end
% R filling edges
for i=2:1:row
    if mod(i,2) == 0
        r_channel(i,1) = r(i,2);

    else
        r_channel(i,1) = r(i-1,2)/2 + r(i+1,2)/2;
        r_channel(i,col) = r(i-1,col)/2 + r(i+1,col)/2;
    end
end
for j=2:1:col
    if mod(j,2) == 0
        r_channel(1,j) = r(2,j);
    else
        r_channel(1,j) = r(2,j-1)/2 + r(2,j+1)/2;
        r_channel(row,j) = r(row,j-1)/2 + r(row,j+1)/2;
    end
end


```

```
% B
for i=1:1:row-1
    for j=1:1:col-1
        if b_channel(i,j) == 0
            if mod(j,2) == 0 && mod(i,2) == 1
                b_channel(i,j) = b(i,j-1)/2 + b(i,j+1)/2;
            elseif  mod(j,2) == 1 && mod(i,2) == 0
                b_channel(i,j) = b(i+1,j)/2 + b(i-1,j)/2;
            elseif mod(j,2) == 0 && mod(i,2) == 0
                b_channel(i,j) =  b(i-1,j-1)/4 + b(i+1,j-1)/4 + b(i+1,j+1)/4 + b(i+1,j
                    -1)/4;
            end
        end
    end
end
% B filling edges
b_channel(row,col) = b(row-1,col-1);
b_channel(1,col) = b(1,col-1);
for i=2:1:row-1
    if mod(i,2) == 0
        b_channel(i,col) = b(i-1,col-1)/2 + b(i+1,col-1)/2;
    else
        b_channel(i,col) = b(i,col-1);
    end
end
for j=1:1:col-1
    if mod(j,2) == 1
       b_channel(row,j) = b(row-1,j);
    else
       b_channel(row,j) = b(row-1,j-1)/2 + b(row-1,j+1)/2;
    end
end



% G
g_channel(1,1) = g(1,2)/2 + g(2,1)/2;
for i=2:1:row-1
    for j=2:1:col-1
        if( (mod(i,2)==0 && mod(j,2)==0) || ( mod(i,2)==1 && mod(j,2)==1))
            g_channel(i,j)= g(i-1,j)/4+ g(i,j-1)/4+ g(i+1,j)/4+ g(i,j+1)/4;
        end
    end
end
% G filling edges
for i=2:1:row-1
    if mod(i,2) == 1
       g_channel(i,1) = g(i-1,1)/3 + g(i+1,1)/3 + g(i,2)/3;
    elseif mod (i,2) == 0
       g_channel(i,col) = g(i-1,col)/3 + g(i+1,col)/3 + g(i,col-1)/3;
    end
end
for j=2:1:col-1
    if mod(j,2) == 1
       g_channel(1,j) = g(1,j-1)/3 + g(1,j+1)/3 + g(1+1,j)/3;
    elseif mod(j,2) == 0
        g_channel(row,j) = g(row,j-1)/3 + g(row,j+1)/3 + g(row-1,j)/3;
    end
end



final_result(:,:,1) = r_channel;
final_result(:,:,2) = g_channel;
final_result(:,:,3) = b_channel;
```

```
final_result=cast(final_result,'uint8');
figure;
    subplot (1,2,1);
    imshow(final_result);
    xlabel('result')
    subplot (1,2,2);
    imshow(color_img);
    xlabel('template')
[PSNR] = psnr(final_result, color_img)

imwrite(final_result,'bilinieal.png');
%% Functions

function separated_img  = separate_color(img, color) %BGGR system
    [row,col] = size(img);
    img = cast(img,'double');
    separated_img = double(zeros(row,col));
    for i = 1:2:row
        for j = 1:2:col
            if color == 'b'
                separated_img(i,j) = img(i,j);
            elseif color == 'g'
                separated_img(i,j+1) = img(i,j+1);
                separated_img(i+1,j) = img(i+1,j);
            elseif color == 'r'
                separated_img(i+1,j+1) = img(i+1,j+1);
            else
                fprintf('error occured, wrong color imput!!!');
                break
            end
        end
    end
end
```

zad3.m

4 Porównanie metod

- Porównaj działanie zaimplementowanych metod oraz wbudowanej funkcji Matlaba, pod kątem jakości i szybkości działania,
- zwróć uwagę na powstające artefakty i ilość szczegółów w obrazie wynikowym,
- dodatkowo skuteczności algorytmów demozaikowania oceń za pomocą za pomocą wskaźników jakości takich jak PSNR czy NCD

Czas działania poszczególnych metod:

- Czas działania programu liczącego metodą najbliższego sąsiada: 0,662 s
- Czas działania programu liczącego metodą interpolacji biliniowej: 0,537 s
- Czas działania programu liczącego metodą Demosaic wbudowaną w Matlab: 0.472 s

Wartości wahają się w zależności od próby, ale różnica między metodami jest cały czas równa około 0,1 sekundy na korzyść interpolacji biliniowej. Jest to raczej mniej spodziewany wynik. Jednakże biorąc pod uwagę moc obliczeniową teraźniejszych komputerów, to różnicę mogą wynikać z działania samego programu Matlab niż konkretnie różnicy metod. Najlepszy wynik osiąga metoda wbudowana, które o kolejne 0,1 sekundy szybciej wykonuje operację.

Porównanie wartości PSNR poszczególnych metod

- Metoda najbliższego sąsiada:
  flamastry : PSNR = 25.4501, skały: PSNR = 25.2255
- Metoda interpolacji biliniowej:
  flamastry : PSNR = 28.5290, skały: PSNR = 29.3396
- Metoda Demosiac wbudowana w Matlab:
  flamastry : PSNR = 30.9308, skały : PSNR = 34.7963

Wyniki są zdecydowanie bardziej zadowalające dla interpolacji biliniowej. Jednakże metoda wbudowana w Matlab radzi sobie jeszcze lepiej.

Podsumowując, metoda interpolacji biliniowej radzi sobie lepiej niż metoda najbliższego sąsiada, natomiast metoda wbudowana jest w tym jeszcze lepsza w dokładności jak i w czasie. Jeśli planujemy użyć zdjęcia nie powiększając go ani nie wycinając kawałków, to każda z metod zda test. Jednakże w przypadku chęci uzyskania jak najlepszej kopii, to zdecydowanie najlepiej korzystać z metody Demosaic, gdyż najwieriniej odzwierciedli kopię.