AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Techniki Wizyjne

Raport

Nikodem Dybiński

Imię i nazwisko

Inżynieria Mechatroniczna, semestr VI

Kierunek studiów

Obraz HDR wykonany na podstawie szeregu zdjęć zrobionych bez statywu

Temat projektu

dr inż. Ziemowit Dworakowski

Prowadzący

Kraków, rok 2017

1 WPROWADZENIE

Poniższy projekt zrealizowany został w oparciu o wiedzę zdobytą w ramach przedmiotu *Techniki Wizyjne* oraz o informacje pozyskane samodzielnie. Skorzystano zarówno z gotowych algorytmów, jak i rozwiązań autorskich. Baza zdjęć, stanowiąca wejście do programu, została przygotowana w większości własnoręcznie, choć skorzystano również z materiałów zewnętrznych. Kod źródłowy został w całości napisany korzystając z oprogramowania *MATLAB R2016a* firmy *The MathWorks, Inc*.

1.1 ZADANIE PROJEKTOWE

Głównym zadaniem projektowym, mającym zostać zrealizowanym przez napisany kod, jest wygenerowanie obrazu, o szerokim zakresie dynamiki (HDR, *ang. High Dynamic Range*), na podstawie szeregu dostarczonych zdjęć o różnej wartości ekspozycji, wykonanych bez użycia statywu. Zdjęcia takie, choć przedstawiające jedną scenę, są w mniejszym lub większym stopniu względem siebie przesunięte, co wiąże się z koniecznością wzajemnego ich dopasowania.

Rys. 1.1. Obraz HDR, wraz ze zdjęciami źródłowymi

1.2 FOTOGRAFIA HDR

HDR (*ang. High Dynamic Range Imaging*) to technika w fotografii, polegająca na wykonaniu serii zdjęć przedstawiających tą samą scenę, wykonanych jednak z różną wartością ekspozycji, a następnie ich połączenie, w celu uzyskania obrazu o dużej wartości rozpiętości tonalnej. Przez rozpiętość tonalną rozumiemy różnicę pomiędzy najjaśniejszym i najciemniejszym punktem fotografii. Powyżej (*Rys. 1*.1) przedstawiony został przykład działania omawianego procesu. Zostały tutaj użyte cztery obrazy źródłowe. Obszary prześwietlone lub niedoświetlone na jednych zdjęciach są doskonale widoczne na innych. Technika ta pozwala uzyskać obrazy zbliżone do tych obserwowanych ludzkim okiem.

Przeprowadzając algorytm HDR ręcznie jesteśmy w stanie dobrać jego parametry tak, aby uzyskany efekt spełniał w jak największym stopniu nasze oczekiwania. Automatyzacja tego procesu, będąca przedmiotem prowadzonego projektu, nie zawsze daje zadowalające rezultaty. Pomimo teoretycznego uzyskania szerokiego zakresu dynamiki uzyskany obraz może okazać się pozbawiony walorów estetycznych lub przekłamywać rzeczywistość w nieakceptowalnym stopniu. Ocena jego działania jest więc subiektywna, dlatego poniższy projekt ma charakter eksperymentalny, porównując kilka możliwych rozwiązań postawionego problemu.

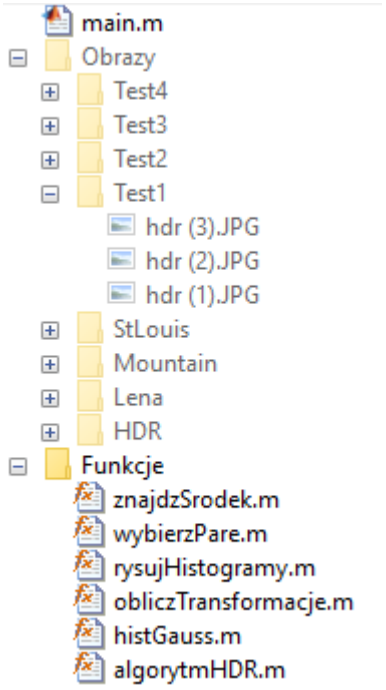
2 OPIS METODY

Przed rozpoczęciem pracy z programem należy przygotować bazę zdjęć wykonanych zgodnie z założeniami projektu. Każde zdjęcie powinno zawierać różną wartość ekspozycji. Osiągnąć to można operując parametrami czasu naświetlania, otwarcia przesłony obiektywu, a także wielkości współczynnika *ISO*. Najlepsze efekty uzyskamy robiąc zdjęcia częściowo prześwietlone oraz częściowo niedoświetlone. Fotografie powinny zostać wykonane bez użycia statywu.

Przygotowane pliki należy umieścić w folderze utworzonym wewnątrz folderu *Obrazy*. Nazwa utworzonego katalogu nie powinna zawierać polskich znaków, ani spacji. Wszystkie zdjęcia powinny mieć rozszerzenie *jpg* (w innym przypadku należy zmodyfikować wartość zmiennej *rozszerzenie* (28 linijka kodu *main.m*). Warto mieć na uwadze, że rozmiar obrazów wpływa bezpośrednio na czas wykonywania algorytmu, dlatego dla celów testowych nie zaleca się wprowadzania obrazów o dłuższym boku większym niż 1200 pikseli.

2.1 STRUKTURA PROJEKTU

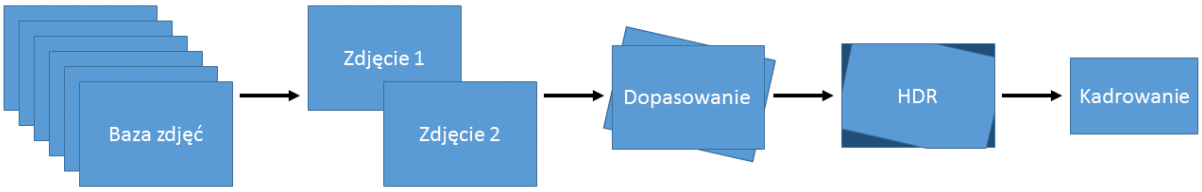
W folderze projektowym znajdują się: kod główny programu (*main.m*) oraz dwa podfoldery: *Obrazy* i *Funkcje*. W pierwszym z nich znajdują się kolejne podfoldery, zawierające przygotowane zestawy zdjęć, w drugim zaś kody użytych w programie funkcji. Struktura projektu zwizualizowana została poniżej w postaci drzewa katalogowego.



Rys 2.1 Drzewo katalogowe projektu

2.2 SCHEMAT DZIAŁANIA PROGRAMU

Program zrealizowany został w postaci kodu *MATLAB*. Użytkownik, przed rozpoczęciem jego działania, proszony jest o wskazanie bazy oraz wybór algorytmu doboru pary zdjęć. Możliwe jest również określenie sposobu prezentacji wyników. Poniżej (*Rys. 2.1*) przedstawiony został zarys pracy kodu.



Rys. 2.2. Schemat działania programu

Ze schematu odczytać można następujące po sobie etapy:

1. Wczytanie do projektu bazy zdjęć.
2. Wybór pary zdjęć, mającej dać najlepsze rezultaty.
3. Dopasowanie do siebie zdjęć.
4. Przeprowadzenie algorytmu HDR.
5. Kadrowanie części wspólnej dopasowanych zdjęć.

Dla celów testowych możliwe jest wyświetlenie efektów pracy poszczególnych etapów. Ponadto wyświetlane są również histogramy pary obrazów wejściowych oraz obrazu wyjściowego. Poszczególne algorytmy zostaną opisane w następnych podrozdziałach.

2.3 WCZYTYWANIE ZDJĘĆ

Po uruchomieniu programu użytkownik proszony jest o wskazanie paczki zdjęć, mającej zostać zaimportowaną do programu. Po dokonaniu wyboru wczytywane są wszystkie pliki zdjęciowe zawierające się wewnątrz wybranego katalogu. Domyślnie wyszukiwanym formatem jest *jpg*. Zdjęcia zapisywane są wewnątrz struktury *im*, gdzie przechowywane są zarówno dane graficzne, jak i nazwy plików.

2.4 WYBÓR PARY ZDJĘĆ

Algorytm wyboru pary zdjęć zdaje się być najistotniejszym elementem programu, ponieważ od jego efektywności zależą bezpośrednio wszystkie następujące po nim operacje. Dokonuje się on przy pomocy funkcji *wybierzPare*(). Jako argumenty przyjmuje ona strukturę zawierającą zdjęcia, zmienną *method*, zawierającą wybór użytkownika dotyczący typu użytych wewnątrz niej algorytmów, oraz zmienną logiczną *show*, określającą sposób prezentacji jej pracy. Funkcja zwraca indeksy wybranej pary zdjęć.

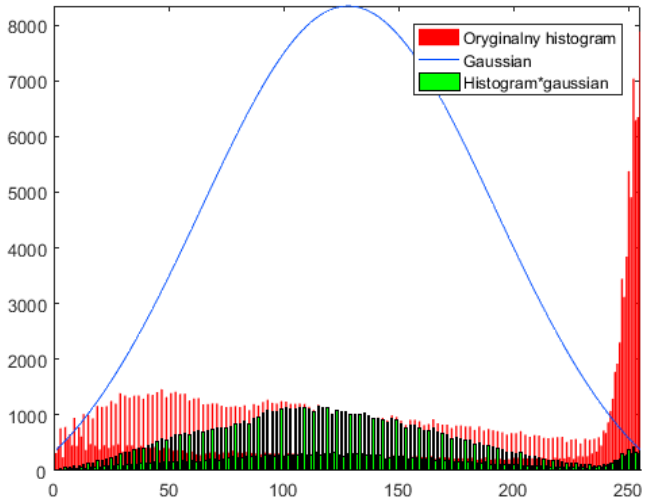
W programie zaimplementowane zostały trzy wersje powyższej funkcji.

2.4.1 Wersja pierwsza

W pierwszym kroku, za pomocą funkcji *znajdzSrodek()*, określana jest średnia pozycja „środka" histogramów wszystkich wczytanych zdjęć. Odbywa się to poprzez odnalezienie wartości na osi odciętych dla histogramu pojedynczego obrazu, dla której sumy wysokości słupków histogramów po jej lewej i prawej stronie są równe. Proces powtarzany jest dla wszystkich wczytanych obrazów, a następnie sumowany i dzielony przez ich ilość.

Następnie tworzymy histogram danego obrazu, po czym określamy sumę wysokości jego słupków znajdujących się po lewej i po prawej stronie obliczonego środka. Stosunek tych wielkości decyduje, czy zdjęcie jest „jasne", czy „ciemne". Powtarzając ten proces na wszystkie wczytane zdjęcia otrzymujemy dwa wektory indeksów zdjęć: o krótkiej oraz o długiej ekspozycji. Takie działanie w teorii zawsze zwróci przynajmniej jeden indeks „ciemny" i przynajmniej jeden „krótki", jednak ze względu na przyjęte zaokrąglenia oraz zabezpieczając się przed skrajnie nieprzychylnym przypadkiem (wczytanie jedynie dwóch zdjęć o zbliżonej ekspozycji), wprowadzony został jeszcze algorytm sztucznie rozdzielający jeden z wektorów pomiędzy dwa z nich bazując na odczytywanych poziomach jasności.

Kolejnym etapem jest zsumowanie wszystkich obrazów o krótkiej ekspozycji ze wszystkimi obrazami o długiej ekspozycji. Sumy te stanowią wejście do funkcji *histGauss()*, która sprzęża ich histogram z gaussianem o środku pokrywającym się ze środkiem tego histogramu, a następnie sumuje otrzymane wartości. Poniżej (*Rys 2.3)* przedstawione zostało jej działanie. Wartość zwracana przez *histGauss()* jest częściowym wyznacznikiem zakresu dynamiki zdjęcia, ponieważ im mniej na obrazie prześwietleń lub niedoświetleń, tym jest ona większa.



Rys 2.3 Przykładowe działanie funkcji histGauss()

Tworzona jest macierz wartości zwróconych przez omawianą powyżej funkcję. Następnie odszukiwana jest jej największa wartość. Indeks wiersza i kolumny tej wartości stanowi wyjście z algorytmu – odszukaną parę zdjęć.

2.4.2 Wersja druga

Wersja ta stanowi uproszczenie wersji poprzedniej – nie przeprowadzany jest podział obrazów na „jasne" i „ciemne". Od razu tworzona jest macierz wartości zwróconych przez funkcję *histGauss()*, czyli sumy tonów średnich histogramu sumy pary zdjęć.

Odszukiwana jest największa wartość oraz odczytywane są indeksy. Na koniec zdjęcia szeregowane są względem poziomu jasności.

2.4.3 Wersja trzecia

Na każdej parze zdjęć przeprowadzany jest algorytm HDR. Otrzymany obraz wprowadzamy do funkcji *histGauss()*. Zwrócona przez nią wartość jest wyznacznikiem efektywności przeprowadzonej operacji.

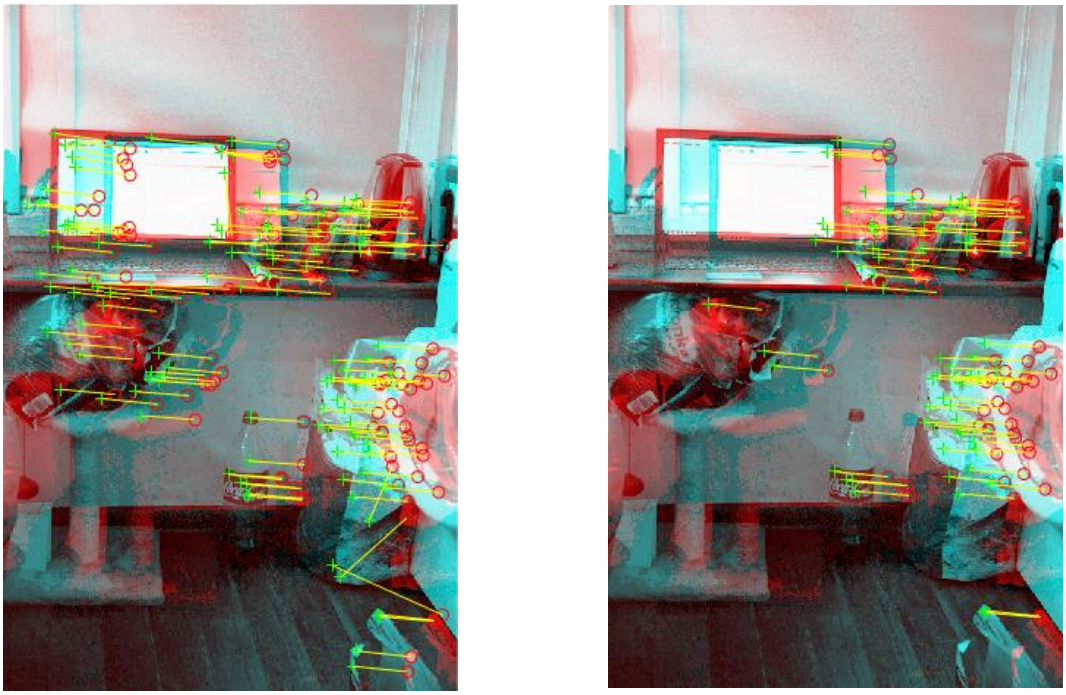
2.5 DOPASOWANIE PARY ZDJĘĆ

Przeprowadzenie fotografii bez użycia statywu wiąże się nieodłącznie z przesunięciami pomiędzy kolejnymi kadrami. Dopasowanie polega na odnalezieniu na obu zdjęciach odpowiadających sobie elementów, a następnie manipulację jednym z obrazów do momentu nałożenia ich na siebie. O ile sama ekstrakcja cech danej fotografii nie stanowi większego problemu, o tyle dobór ich par odpowiadających sobie na obu zdjęciach oraz określanie wymaganej transformacji jest zadaniem wymagającym skomplikowanych obliczeń optymalizacyjnych. Z tego powodu zdecydowano się na skorzystanie z gotowego algorytmu realizowanego dostosowaną na potrzeby projektu funkcją *obliczTransformacje()*.

2.5.1 Algorytm SURF

Algorytm SURF (*ang. Speeded-Up Robust Features*) jest algorytmem przy pomocy którego odnajdywać możemy na obrazach poszczególne obiekty. Jest on częściowo wzorowany na algorytmie SIFT, mającym takie samo zastosowanie, jest jednak od niego kilkukrotnie szybszy (kosztem odporności na błędy i dokładności). Szybkość ta wynika z jego sposobu pracy opierającego się na tworzeniu pośrednich reprezentacji obrazu (tzw. obrazów zintegrowanych), będących lokalnymi sumami wartości otaczających pikseli. W związku z tym prędkość jego działania nie zależy od rozmiaru obrazów wejściowych.

W pierwszej kolejności, na obu obrazach, odnajdywane są punkty charakterystyczne oraz określane są wartości opisujące je (tzw. deskryptory). Następnie wartości deskryptorów dla punktów obu zdjęć są porównywane, dzięki czemu są one parowane. Z zestawu par odrzucane są te nie pasujące do większości. Ostatecznie obliczana jest wartość macierzy transformacji. Macierz ta stanowi wyjście z funkcji *obliczTransformacje()* oraz wejście do funkcji *imwarp()* będącej częścią biblioteki *MATLAB*.



Rys 2.4 Wszystkie pary punktów odnalezione przez algorytm SURF (lewo) oraz pary pozostawione po odrzuceniu wyznaczonych błędnie (prawo)

2.6 ALGORYTM HDR

Podobnie jak w poprzednim przypadku – algorytm HDR do prawidłowego działania wymaga przeprowadzenia szeregu złożonych obliczeń optymalizacyjnych. W związku z tym skorzystałem z kodu dostępnego w bibliotece *MATLAB*. Algorytm ten realizowany jest przez funkcję *algorytmHDR()*, będącą dostosowaną na potrzeby projektu kopią funkcji testującej design *mlhdlc_hdr.m*, odpowiedzialny bezpośrednio za łączenie obrazów. Wejście do niej stanowią dwa obrazy: jeden o krótkiej ekspozycji, drugi o długiej.

2.7 KADROWANIE

Po dopasowaniu do siebie obrazów otrzymujemy jeden obraz w niezmienionym kształcie prostokąta oraz drugi, przekształcony w czworokąt. Najbardziej pożądanym kadrem będzie więc największy prostokąt wpisany w część wspólną obu obrazów. Odnalezienie współrzędnych tego prostokąta jest odrębnym problemem optymalizacyjnym, więc ze względu na swoją złożoność proces kadrowania został uproszczony. Użytkownik proszony jest o ręczne wskazanie docelowego kadru.

3 WERYFIKACJA EKSPERYMENTALNA

Część projektu stanowią dostarczone paczki zdjęć, przeznaczone do testowania poprawności działania kodu. W ich skład wchodzą foldery:

- **HDR** – seria osiemnastu własnoręcznie zrobionych zdjęć, paczka najlepiej spełniająca założenia dotyczące pożądanego wejścia do programu,
- **Lena** – klasyczna Lena w ośmiu odsłonach, różniących się między sobą jasnością oraz kadrowaniem,
- **Mountain** – cztery dopasowane zdjęcia chatki górskiej o różnej ekspozycji, pozyskane z Internetu,
- **StLouis** – j.w.,
- **Test1** – trzy wykonane własnoręcznie fotografie o różnej ekspozycji,
- **Test2** – dwie wykonane własnoręcznie fotografie o stałej ekspozycji, ale różnym kadrowaniu,
- **Test3** – j.w.

HDR, *Lena* oraz *Test1* służą do całościowego testowania aplikacji, *Mountain* i *StLouis* to sprawdzenie algorytmów doboru pary zdjęć oraz HDR, *Test2* i *Test3* zaś pokazują działanie algorytmu SURF.

3.1 ZAŁOŻENIA

Zakłada się, że wejście do programu stanowić będzie pakiet zdjęć przygotowanych zgodnie z tematem projektu, ponadto:

- wszystkie zdjęcia powinny przedstawiać tą samą scenę,
- poruszenia pomiędzy kolejnymi fotografiami powinny jak najmniejsze (minimum 60% powierzchni wspólnej po dopasowaniu),
- należy unikać zdjęć o takiej samej ekspozycji,
- wszystkie pliki powinny być tej samej rozdzielczości oraz tego samego rozszerzenia.

W interakcji z aplikacją należy wprowadzać tylko wartości sugerowane przez program w postaci zakresów podanych w nawiasach.

3.2 STANDARDOWY TRYB PRACY

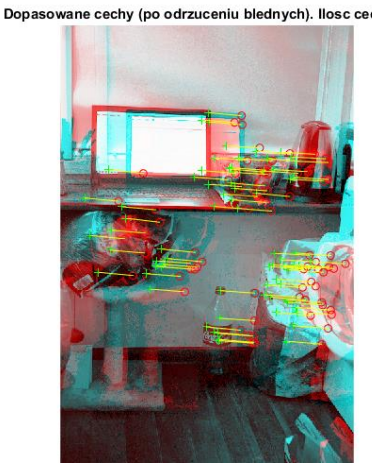
3.2.1 Dziennik okna komend

```
HDR bez statywu - Nikodem Dybinski
----------------------------------
Dostepne bazy zdjec:
   1. HDR
   2. Lena
   3. Mountain
   4. StLouis
   5. Test1
   6. Test2
   7. Test3
Wybierz baze zdjec (1-7): 5
Dostepne algorytmy:
   1. Podzial obrazow na jasne i ciemne, wybor najwiekszej sumy srodkow
histogramow.
   2. Wybor najwiekszej sumy srodkow histogramow.
   3. Przeprowadzenie oceny jakosci dzialania algorytmu HDR dla kazdej
pary. (dlugi czas trwania)
Wybierz rodzaj algorytmu wyboru pary zdjec (1-3): 1
Wyswietlic kolejne kroki dzialania programu? (0-1): 1
----------------------------------
Wybrane parametry programu:
              Zestaw zdjec: "Test1" (rozszerzenie "jpg")
                  Algorytm: 1
----------------------------------
Rozpoczecie pracy programu.
Wczytano 3 zdjecia.
Obrazy wybrane jako wejscie do algorytmu HDR: hdr (3).JPG oraz hdr (1).JPG
Wybierz obszar wspolny dopasowanych zdjec...
Zakonczenie pracy programu.
```

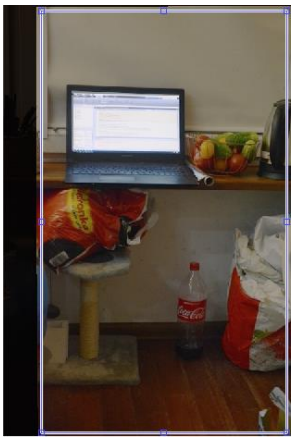
3.2.2 Działanie programu

Rys 3.1. Figura I

Rys 3.2. Figura II

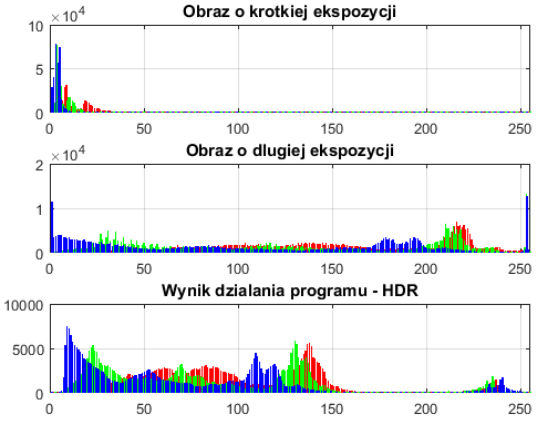


Rys 3.3. Figura III



Rys 3.4. Figura IV

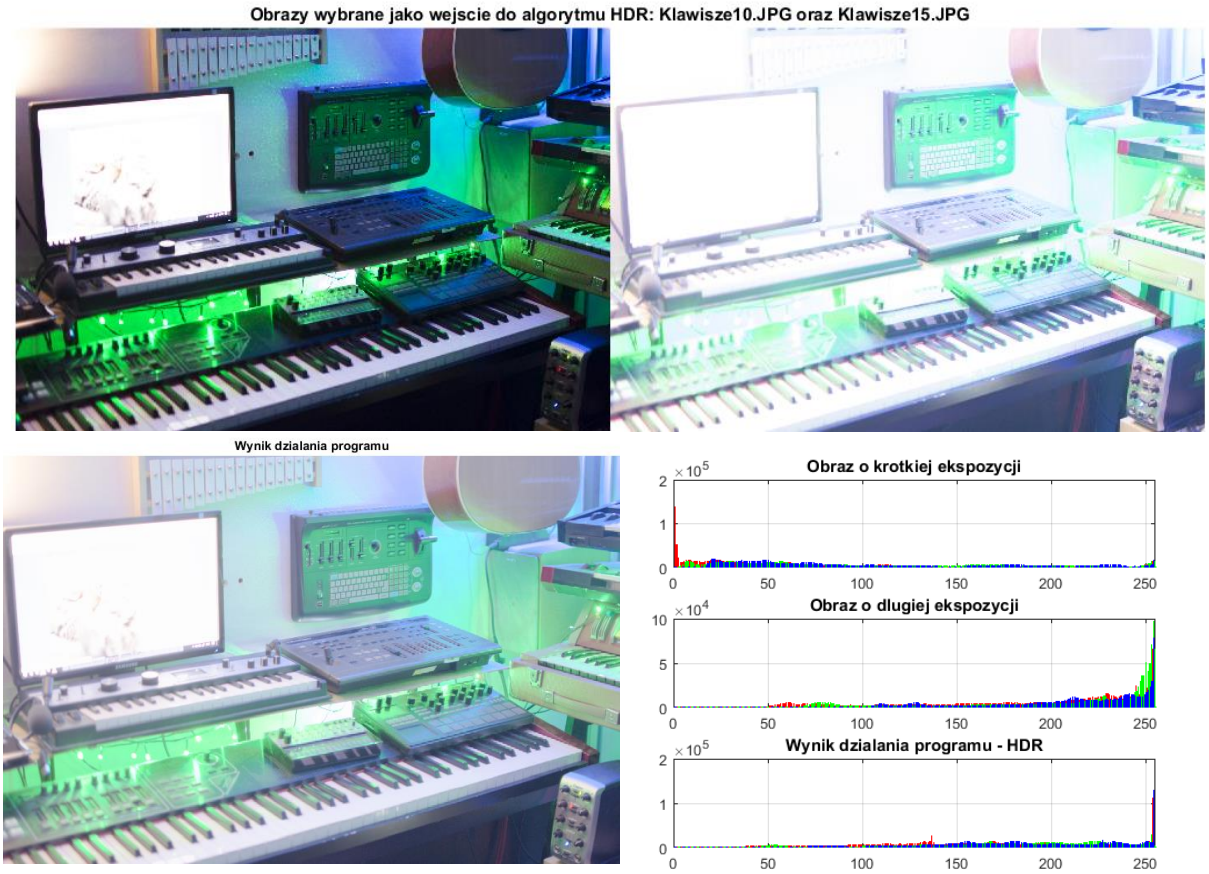
Rys 3.5. Figura V



Rys 3.6. Figura VI

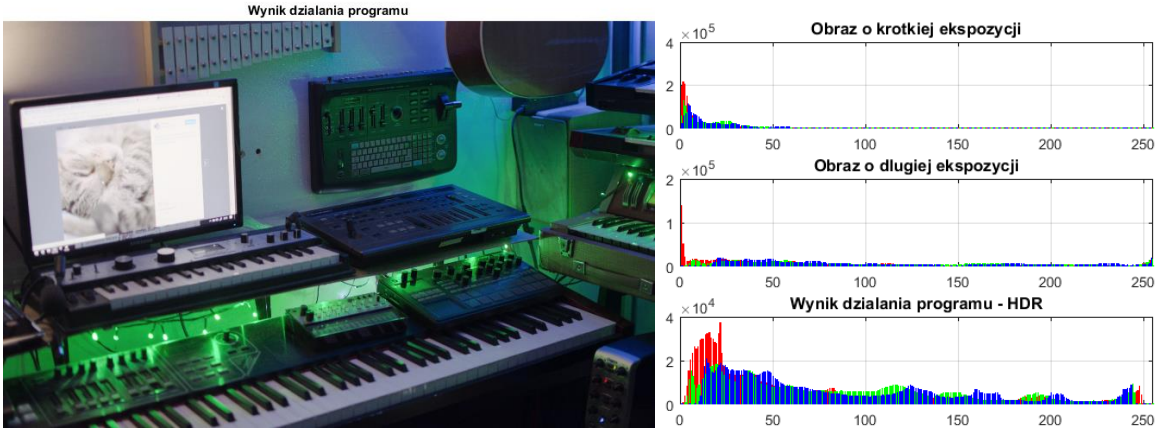
3.3 PORÓWNANIE DZIAŁANIA ALGORYTMÓW

3.3.1 Algorytm I



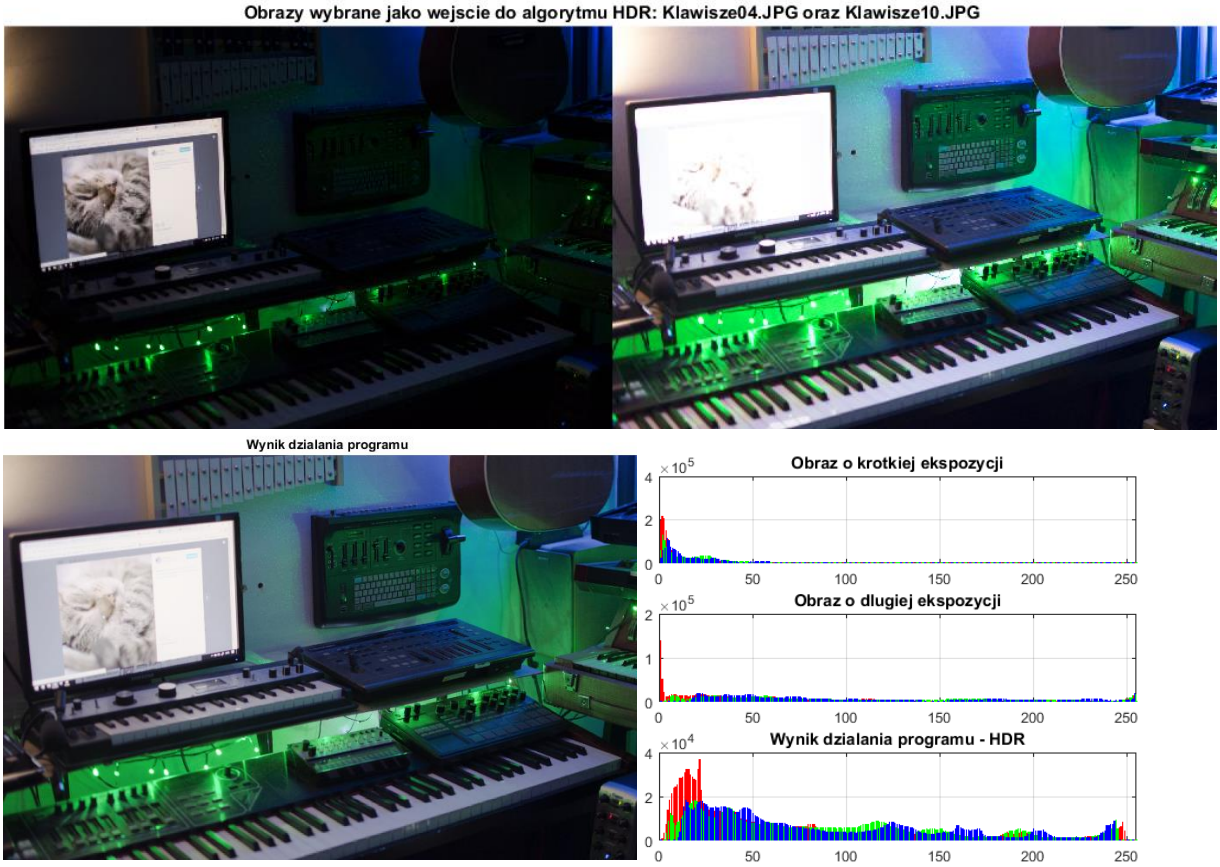
Rys. 3.7. Czas wykonania: 40 sekund

3.3.2 Algorytm II



Rys. 3.8. Czas wykonania: 38 sekund

3.3.3 Algorytm III



Rys. 3.9. Czas wykonania: 45 minut

4 WNIOSKI

4.1 ALGORYTM DOBORU PARY ZDJĘĆ

Pierwszy algorytm doboru pary zdjęć obarczony jest zgrubnym doborem parametrów, z czym wiąże się utrata skuteczności, zwłaszcza przy dużej ilości danych wejściowych. Trzeci zaś, oprócz podobnych wad, jest ekstremalnie czasochłonny. Czas jego wykonania rośnie bowiem wykładniczo, zależnie od ilości wczytanych zdjęć.

Optymalnym rozwiązaniem okazuje się zastosowanie algorytmu drugiego. Jest to metoda najprostsza, dająca najlepsze rezultaty przy dobrze przygotowanych materiałach wejściowych. Czas obliczeń również jest możliwie najkrótszy. Problem pojawia się w momencie wprowadzenia do programu zdjęć o takiej samej ekspozycji. Zostaną one wówczas zakwalifikowane jako para najbardziej efektywna, co nie jest rzeczą pożądaną. Aby tego uniknąć należy przestrzegać założeń projektowych.

4.2 MOŻLIWE UDOSKONALENIA

Praca nad projektem zaowocowała w obserwację szeregu możliwych udoskonaleń, wpływających korzystnie na pracę z programem oraz jego efektywność. Należą do nich m.in.:

- zaprojektowanie interfejsu graficznego,
- możliwość wczytywania zdjęć o dowolnym rozszerzeniu z dowolnego miejsca na dysku,
- możliwość ręcznego doboru pary zdjęć,
- optymalizacja algorytmów doboru pary zdjęć (chociażby pod kątem uwzględnienia poruszenia pomiędzy kolejnymi fotografiami),
- automatyzacja procesu kadrowania,
- implementacja algorytmu HDR operującego na dowolnej ilości zdjęć,
- możliwość zapisania wyniku działania programu do pliku.

5 MATERIAŁY ŹRÓDŁOWE

1. Przykład obrazu HDR (*Rys 1.1*) [Online]. Dostępny w Internecie: *https://commons.wikimedia.org/wiki/File:StLouisArchMultExpToneMapped.jpg*
2. Algorytm SURF. Część biblioteki *MATLAB R2016a, The MathWorks®*
3. Algorytm HDR. Część biblioteki *MATLAB R2016a, The MathWorks®*
4. Przykładowy pakiet zdjęć *„Mountain”* [Online]. Dostępny w Internecie: http://farbspiel-photo.com/learn/hdr-before-and-after/hdr-cookbook-before-and-after-mountain-shed-hdr
5. Przykładowy pakiet zdjęć *„StLouis”* [Online]. Dostępny w Internecie: *https://commons.wikimedia.org/wiki/File:StLouisArchMultExpToneMapped.jpg*

6 KOD PROGRAMU

6.1 MAIN.M

```
%%
% Projekt:    Program tworzacy zdjecie o szerokim zakresie dynamicznym
%             (HDR) z szeregu zdjec wykonanych bez statywu.
% Autor:      Nikodem Dybinski
%             Inzynieria Mechatroniczna, semestr VI
%             Wydzial Inzynierii Mechanicznej i Robotyki
%             Akademia Gorniczo Hutnicza im. Stanislawa
%             Staszica w Krakowie
% Przedmiot:  Techniki Wizyjne
% Prowadzacy: dr inz. Ziemowit Dworakowski
%             Katedra Robotyki i Mechatroniki
%%
clear all
close all
clc
addpath('Funkcje')
obrazy=dir('Obrazy');
disp('HDR bez statywu - Nikodem Dybinski')
disp('-----------------------------------')
%% Parametry
disp('Dostepne bazy zdjec:')
for i=3:size(obrazy,1)
    disp(['    ' num2str(i-2) '. ' obrazy(i).name]);
end
wybor=input(['Wybierz baze zdjec (1-' num2str(size(obrazy,1)-2) '): ']);
name=obrazy(wybor+2).name;        %nazwa folderu zawierajacego zestaw zdjec
rozszerzenie='jpg'; %szukane rozszerzenie zdjec
disp('Dostepne algorytmy:')
disp('   1. Podzial obrazow na jasne i ciemne, wybor najwiekszej sumy
srodkow histogramow.')
disp('   2. Wybor najwiekszej sumy srodkow histogramow.')
disp('   3. Przeprowadzenie oceny jakosci dzialania algorytmu HDR dla
kazdej pary. (dlugi czas trwania)')
algorytm=input(['Wybierz rodzaj algorytmu wyboru pary zdjec (1-3): ']);
wybor=input('Wyswietlic kolejne kroki dzialania programu? (0-1): ');
if wybor == 1
    show = true;
else
    show = false;
end
clear wybor
disp('-----------------------------------')
disp(['Wybrane parametry programu:'])
disp(['               Zestaw zdjec: "' name '" (rozszerzenie "' rozszerzenie
'")'])
disp(['                   Algorytm: ' num2str(algorytm)])
disp('-----------------------------------')
disp('Rozpoczecie pracy programu.')
%% Wczytanie zdjec
file = dir(fullfile('Obrazy',name,['*.' rozszerzenie])); %wczytanie obrazow
z pliku
NF = length(file);  %ilosc wczytanych obrazow
if NF<2
    error('Nie odnaleziono wystarczajacej ilosci zdjec! (minimum 2)')
```

```
end
im = cell(NF,2);    %definiowanie przestrzeni danych
for i=1:1:NF        %wczytanie obrazow do przestrzeni danych
    im{i,1} = imread(fullfile('Obrazy',name, file(i).name)); %wczytanie
obrazow
    im{i,2} = file(i).name;    %wczytanie nazw plikow
end, clear i file ;
if NF<5
    disp(['Wczytano ' num2str(NF) ' zdjecia.'])
else
    disp(['Wczytano ' num2str(NF) ' zdjec.'])
end
clear NF name rozszerzenie
%% Selekcja obrazow oraz ich prezentacja
[short,long]=wybierzPare(im,algorytm,false);
disp(['Obrazy wybrane jako wejscie do algorytmu HDR: ' im{short,2} ' oraz '
im{long,2}])
if(show)
    figure
    imshowpair(im{short,1},im{long,1},'montage')
    title(['Obrazy wybrane jako wejscie do algorytmu HDR: ' im{short,2} '
oraz ' im{long,2}])
    drawnow
end
%% Dopasowanie obrazow
outputView = imref2d(size(im{short,1}));
trans=obliczTransformacje(im{short,1},im{long,1},show); %liczenie
transformacji
long_adjusted = imwarp(im{long,1},trans,'OutputView',outputView);
%wykonanie transformacji
%% Przeprowadzenie algorytmu HDR
HDR=algorytmHDR(im{short,1},long_adjusted);
%% Kadrowanie
disp('Wybierz obszar wspolny dopasowanych zdjec...')
figure
HDR = imcrop(HDR);
%% Prezentacja wynikow
figure
imshow(HDR);
title('Wynik dzialania programu')
drawnow
if(show)
    rysujHistogramy(im{short,1},im{long,1},HDR);
end
disp('Zakonczenie pracy programu.')
```

6.2 WYBIERZPARE.M

```
function [shortIdx,longIdx] = wybierzPare(im,method,show)
%% Wstepne definicje
NF=size(im,1);      %odczytanie ilosci zdjec
switch(method)
    case 1      %podzial obrazow na jasne i ciemne, wybor najwiekszej sumy
srodkow histogramow
        brightIdx=[];
        darkIdx=[];
        %% Podzial obrazow na jasniejsze i ciemniejsze
        srodek=znajdzSrodek(im);
```

```
        for m=1:NF
            hist = imhist(rgb2gray(im{m,1}));
            low = sum(hist(1:srodek));
            high = sum(hist((srodek+1):256));
            if low < high   %wiecej wartosci jasnych niż ciemnych
                brightIdx=[brightIdx m];
                if(show)
                    figure
                    bar(hist)
                    xlim([0 255])
                    ylim([0 max(hist)+100])
                    grid on
                    hold on
                    plot([srodek srodek],[0 max(hist)+100],'r--')
                    hold off
                    title(['Bright (' num2str(low) ':' num2str(high) ')'])
                end
            else                                %wiecej wartosci ciemnych niz
jasnych
                darkIdx=[darkIdx m];
                if(show)
                    figure
                    bar(hist)
                    xlim([0 255])
                    ylim([0 max(hist)+100])
                    grid on
                    hold on
                    plot([srodek srodek],[0 max(hist)+100],'r--')
                    hold off
                    title(['Dark (' num2str(low) ':' num2str(high) ')'])
                end
            end
        end, clear m;
        if isempty(brightIdx)
            brightness=0;
            for x=1:length(darkIdx)
                hist = imhist(rgb2gray(im{darkIdx(x),1}));
                high = sum(hist((srodek+1):256));
                if high > brightness
                    brightness = high;
                    brightIdx=darkIdx(x);
                end
            end

        end
        if isempty(darkIdx)
            brightness=0;
            for x=1:length(brightIdx)
                hist = imhist(rgb2gray(im{brightIdx(x),1}));
                low = sum(hist(1:srodek));
                if low > brightness
                    brightness = low;
                    darkIdx=brightIdx(x);
                end
            end
        end

        %% Przeprowadzenie klasyfikacji dla kazdej pary obrazow
        sumMatrix=zeros(length(brightIdx),length(darkIdx));
%zdefiniowanie macierzy klasyfikacyjnej obrazow
        for m=1:length(brightIdx)
```

```
            for n=1:length(darkIdx)
                if brightIdx(m)~=darkIdx(n)

%sumMatrix(m,n)=histGauss(im{m,1},false)+histGauss(im{n,1},false);
                    sumMatrix(m,n)=histGauss(im{m,1}+im{n,1},false);
                end
            end
        end, clear m n;
        %% Odczyt najlepszej klasyfikacji
        [~,I]=max(sumMatrix(:));
        [bIdx, dIdx]=ind2sub(size(sumMatrix),I);
        shortIdx=darkIdx(dIdx);
        longIdx=brightIdx(bIdx);
    case 2  %wybor najwiekszej sumy srodkow histogramow
        sumMatrix=zeros(NF);    %zdefiniowanie macierzy klasyfikacyjnej
obrazow
        %% Przeprowadzenie klasyfikacji dla kazdej pary obrazow
        for m=1:NF
            for n=1:NF
                if m~=n
                    sumMatrix(m,n)=histGauss([im{m,1}+im{n,1}],false);
                end
            end
        end, clear m n;
        %% Odczyt najlepszej klasyfikacji
        [~,I]=max(sumMatrix(:));
        [im1, im2]=ind2sub(size(sumMatrix),I);
        %% Uszeregowanie wybranej pary wzgledem czasu ekspozycji
        hist1=imhist(rgb2gray(im{im1,1}));
        hist2=imhist(rgb2gray(im{im2,1}));
        if(sum(hist1(1:50))>sum(hist2(1:50)))
            shortIdx=im1;
            longIdx=im2;
        else
            shortIdx=im2;
            longIdx=im1;
        end
    case 3  %Przeprowadzenie oceny jakosci dzialania algorytmu HDR dla
kazdej pary
        sumMatrix=zeros(NF);    %zdefiniowanie macierzy klasyfikacyjnej
obrazow
        %% Przeprowadzenie algorytmu HDR dla kazdej pary obrazow
        for m=1:NF
            for n=1:NF
                if m~=n
                    disp(['m: ' num2str(m) ', n: ' num2str(n)])
                    tryHDR=algorytmHDR(im{m,1},im{n,1});
                    sumMatrix(m,n)=histGauss(tryHDR,false);
                end
            end
        end, clear m n;
        %% Odczyt najlepszej klasyfikacji
        [~,I]=max(sumMatrix(:));
        [im1, im2]=ind2sub(size(sumMatrix),I);
        %% Uszeregowanie wybranej pary wzgledem czasu ekspozycji
        hist1=imhist(rgb2gray(im{im1,1}));
        hist2=imhist(rgb2gray(im{im2,1}));
        if(sum(hist1(1:50))>sum(hist2(1:50)))
            shortIdx=im1;
            longIdx=im2;
```

```
        else
            shortIdx=im2;
            longIdx=im1;
        end
end
end
```

6.3 ZNAJDZSRODEK.M

```
function srodek = znajdzSrodek(im)
sr=zeros(1,size(im,1));
a=zeros(1,254);
for i=1:size(im,1)
    hist=imhist(rgb2gray(im{i,1}));
    for j=2:254  %szukamy punktu, wzgledem ktorego sumy lewej i prawej
czesci histogramu sa najbardziej zblizone
        a(j)=abs(sum(hist((j+1):256))-sum(hist(1:j)));
    end
    [~,sr(i)]=min(a(31:226));
end
sr=sr+30;
srodek=round(sum(sr)/length(sr));
end
```

6.4 HISTGAUSS.M

```
function klasyfikacja = histGauss(obraz,show)
hist=imhist(rgb2gray(obraz));
if(show)
    figure
    bar(hist,'r')
    hold on
end
mean=127;
sigma=64;
x=0:255;
fx=1/sqrt(2*pi)/sigma*exp(-(x-mean).^2/2/sigma/sigma);
fx=fx-max(fx)/10;
fx=fx/max(fx);
for i=1:size(fx,2)
    if fx(i)<0
        fx(i)=0;
    end
end, clear i;

for i=1:size(hist,1)
    histfx(i)=hist(i)*fx(i);
end, clear i;
if(show)
    plot(fx*max(histfx),'Color',[0 0.3 1])
    bar(histfx,'g')
    ylim([0 max(histfx)])
    xlim([0 255])
    hold off
end
```

```
klasyfikacja=sum(histfx);
end
```

6.5 OBLICZTRANSFORMACJE.M

```
function tform = obliczTransformacje(im1,im2,show)
im1=histeq(rgb2gray(im1));
im2=histeq(rgb2gray(im2));
%%
% Wykryj i wyekstraktuj cechy z obu obrazow
pts1  = detectSURFFeatures(im1);
pts2 = detectSURFFeatures(im2);
[cechy1,validpts1] = extractFeatures(im1, pts1);
[cechy2,validpts2] = extractFeatures(im2,pts2);
%%
% Dopasuj cechy
pary = matchFeatures(cechy1,cechy2);
dopasowanepts1  = validpts1(pary(:,1));
dopasowanepts2 = validpts2(pary(:,2));
if(show)
    figure; showMatchedFeatures(im1,im2,dopasowanepts1,dopasowanepts2);
    title(['Dopasowane cechy. Ilosc cech: '
num2str(length(dopasowanepts1))]);
    drawnow
end
%%
% Odrzuc niepoprawne dopasowania i oblicz transformacje ('similarity')
[tform,poprawnepts1,poprawnepts2] =
estimateGeometricTransform(dopasowanepts2,dopasowanepts1,'similarity');
if(show)
    figure; showMatchedFeatures(im1,im2,poprawnepts2,poprawnepts1);
    title(['Dopasowane cechy (po odrzuceniu blednych). Ilosc cech: '
num2str(length(poprawnepts1))]);
    drawnow
end
end
```

6.6 ALGORYTMHDR.M

```
function output = algorytmHDR(short, long)
% define HDR output variable
HDR = zeros(size(short));
[height, width, color] = size(HDR);
%% Create the Lum(Y) channels LUTs
% Pre-process
% Lumminance short LUT
ShortLut.x = [0    16    45    96   255];
ShortLut.y = [0    20    38    58   115];
% Lumminance long LUT
LongLut.x = [ 0 255];
LongLut.y = [ 0  140];

% Take the same points to plot the joined Lum LUT
plot_x = 0:1:255;
plot_y_short = interp1(ShortLut.x,ShortLut.y,plot_x); %LUT short
```

```
plot_y_long = interp1(LongLut.x,LongLut.y,plot_x); %LUT long


%% Create the HDR Lum channel
% The HDR algorithm
% read the Y channels

YIQ_short = rgb2ntsc(short);
YIQ_long = rgb2ntsc(long);

%% Stream image through HDR algorithm

for x=1:width
    for y=1:height
        YShort1 = round(YIQ_short(y,x,1)*255); %input short
        YLong1 = round(YIQ_long(y,x,1)*255); %input long

        YShort2 = YIQ_short(y,x,2); %input short
        YLong2 = YIQ_long(y,x,2); %input long

        YShort3 = YIQ_short(y,x,3); %input short
        YLong3 = YIQ_long(y,x,3); %input long

        valid_in = 1;

        [valid_out, x_out, y_out, HDR1, HDR2, HDR3] = mlhdlc_hdr(YShort1,
YShort2, YShort3, YLong1, YLong2, YLong3, plot_y_short, plot_y_long,
valid_in, x, y);

        % use x and y to reconstruct image
        if valid_out == 1
            HDR(y_out,x_out,1) = HDR1;
            HDR(y_out,x_out,2) = HDR2;
            HDR(y_out,x_out,3) = HDR3;
        end
    end
end
%% output HDR
output = ntsc2rgb(HDR);
end
```

6.7 RYSUJHISTOGRAMY.M

```
function fig = rysujHistogramy(im1,im2,HDR)
im1Hist=[imhist(im1(:,:,1)) imhist(im1(:,:,2)) imhist(im1(:,:,3))];
im2Hist=[imhist(im2(:,:,1)) imhist(im2(:,:,2)) imhist(im2(:,:,3))];
HDRHist=[imhist(HDR(:,:,1)) imhist(HDR(:,:,2)) imhist(HDR(:,:,3))];

fig=figure;
subplot(3,1,1)
a=bar(im1Hist);
a(1).EdgeColor = 'red';
a(2).EdgeColor = 'green';
a(3).EdgeColor = 'blue';
xlim([0 255])
title('Obraz o krotkiej ekspozycji')
```

```
grid on

subplot(3,1,2)
b=bar(im2Hist);
b(1).EdgeColor = 'red';
b(2).EdgeColor = 'green';
b(3).EdgeColor = 'blue';
xlim([0 255])
title('Obraz o dlugiej ekspozycji')
grid on

subplot(3,1,3)
c=bar(HDRHist);
c(1).EdgeColor = 'red';
c(2).EdgeColor = 'green';
c(3).EdgeColor = 'blue';
xlim([0 255])
title('Wynik dzialania programu - HDR')
grid on

drawnow
end
```