Obraz zawierający tekst, Czcionka, logo, Grafika

Opis wygenerowany automatycznie

Sprawozdanie 3

Ćwiczenie 3. KAMERY CYFROWE

Autorzy: Krzysztof Zalewa, Wiktor Wojnar

Spis treści

[1.Wstęp 2](#_Toc182935587)

[1.1 Matryce CCD, CMOS 2](#_Toc182935588)

[1.1.1 Budowa 2](#_Toc182935589)

[1.1.2 Wady i zalety 4](#_Toc182935590)

[1.2 Kolory cyfrowe 4](#_Toc182935591)

[1.2.1 Filtry 4](#_Toc182935592)

[1.2.2 De-mozaikowanie 5](#_Toc182935593)

[1.2.3 Zoom cyfrowy/optyczny 5](#_Toc182935594)

[1.3 HDR 6](#_Toc182935595)

[1.5 Stabilizacja drgań 7](#_Toc182935596)

[2.Zadanie Laboratoryjne 9](#_Toc182935597)

[2.1 Treść zadania 9](#_Toc182935598)

[2.2 Opis działania programu 9](#_Toc182935599)

[2.3 Kod programu 9](#_Toc182935600)

[3.Wnioski 14](#_Toc182935601)

[4. Źródła 14](#_Toc182935602)

# 1.Wstęp

## 1.1 Matryce CCD, CMOS

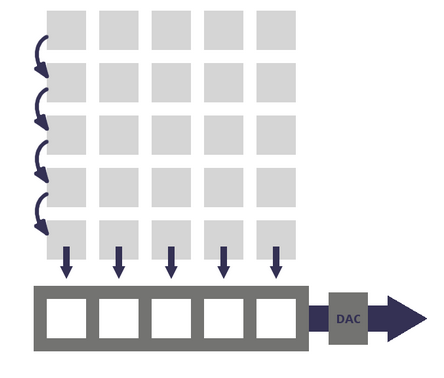
Matryca jest najważniejszą częścią kamery. To ona zamienia światło na sygnał elektryczny. Obecnie dwa najbardziej popularne typy tych matryc to CCD (ang. Charge Coupled Device) i CMOS (ang. complementary metal-oxide-semiconductor)

### Budowa

Ogólna zasada działania obu matryc jest taka sama. Są to płytki krzemowe podzielone na odizolowane elektrycznie obszary czyli piksele. Piksele gromadzą w sobie ładunek elektryczny (proporcjonalny do intensywności padającego na nie światła). Różnicą między CCD a CMOS jest sposób odczytu danych z matrycy.

**CCD**

W matrycach typu CCD dane odczytywane są wierszami (Rys. 1). Dane z wiersza zostają przesunięte do kanału odczytu. Następnie wartości ładunków zostają przetworzone na proporcjonalne napięcie. Tak otrzymane napięcie przechodzi przez konwerter A/D. Dane powstałe w wyniku konwersji zostają zapisane w pamięci kamery.



*Rysunek 1. Schemat pracy matrycy CCD [1]*

**CMOS**

W przeciwieństwie do matryc CCD, w których występował tylko jeden konwerter ładunku na napięcie, w matrycach CMOS każdy piksel ma własny konwerter (Rys 2). Otrzymane w ten sposób napięcie przechodzi przez konwerter A/D i w ten sposób otrzymywane danych

Obraz zawierający kwadrat, Prostokąt, zrzut ekranu, design

Opis wygenerowany automatycznie

*Rysunek 2. Schemat pracy matrycy CMOS [1]*

### Wady i zalety

**Powierzchnia światłoczuła:** W matrycach CMOS występuje więcej elementów niż same piksele (Jak w CCD) dlatego też można ich zmieścić mniej niż w CCD.

**Prędkość działania:** W matrycy CMOS każdy piksel przekształca światło wejściowe na napięcie co znacznie przyspiesza działanie układu.

**Zużycie energii:** W matrycy CMOS tranzystory ułożone są w taki sposób że w danym momencie tylko jeden z nich przewodzi prąd. Oznacza to że matryce CMOS zużywają znacznie mniej prądu niż CCD.

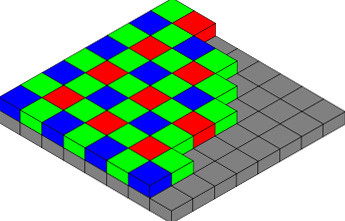
**Koszt produkcji:** Produkcja matryc CMOS jest prosta i tania (technologia CMOS jest używana nie tylko do produkcji matryc).

**Ilość zakłóceń:** W matrycach CMOS odległość między fotodiodą a przetwornikiem A/C jest mniejsza niż w CCD więc jest mniejsza szansa na zniekształcenie.

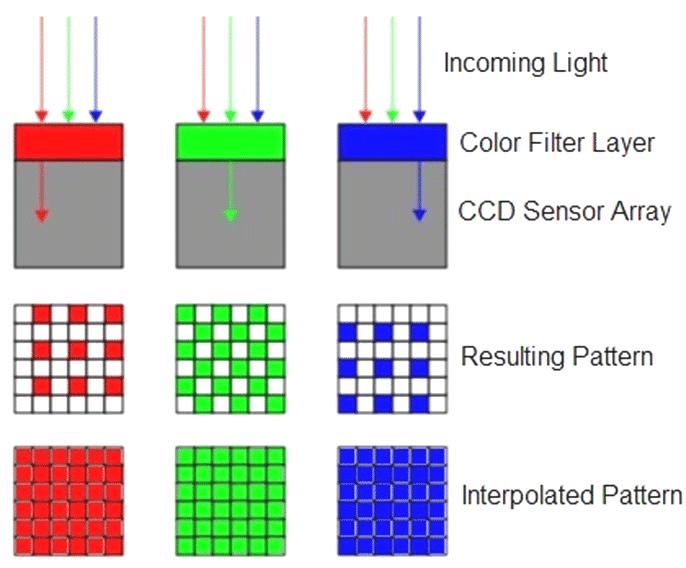
## 1.2 Kolory cyfrowe

### 1.2.1 Filtry

Matryce CMOS i CCD same w sobie nie dają informacji o kolorze obrazu. Żeby obraz nie był czarno biały na matrycę nakłada się filtr który przepuszcza tylko światło o wybranej długości fali. Najczęściej używany jest filtr Bayera (siatka Bayera) widoczny na Rysunku 3. Każdy kwadrat przepuszcza tylko jedną z trzech barw RGB. Kwadraty rozmieszczone są w proporcji 1:2:1 ( Jest 2 razy więcej kwadratów przepuszczających kolor zielony ponieważ ludzkie oko jest bardzo czułe na ten kolor). Aby utworzyć realistyczny obraz stosuje się interpolację Bayera (Rys 4). Polega ona na szacowaniu informacji dla brakującego koloru na podstawie natężenia światła zmierzonego przez sąsiednie piksele. Jednakże ta metoda nie jest idealna i często w obiektach z wysokim kontrastem dochodzi do rozmazania krawędzi.



Rysunek 3. Filtr Bayera 1[2]



Rysunek 4. Filtr Bayera 2 [3]

### 1.2.2 De-mozaikowanie

De-mozaikowanie to proces przetwarzania danych zapisanych w pamięci aparatu na obraz. Dwa proste przykłady procedur demozaikowania to:

* W pierwszej procedurze jeden punkt zdjęcia odtwarzany jest z natężenia światła zarejestrowanego na 4 sąsiadujących z nim pikseli. Maryca **4x4**, czyli 16 pikselowa daje zdjęcie o rozdzielczości **2x2**.
* W drugiej procedurze jest podobna ale punkty dla tworzenia zdjęcia są brane tak, że część pikseli (poza zupełnie zewnętrznymi) brana jest do demozaikowania kilka razy. Np. natężenie zarejestrowane przez pierwszy (licząc od lewej i od góry) "niebieski" piksel brane jest do odtworzenia kolorów punktów **1, 2 4 i 5**. To daje zdjęcie o rozdzielczości **3x3**.

W praktyce demozaikowanie jest bardziej skomplikowane jednakże szczegóły przebiegu tych procedur są trzymane w ścisłej tajemnicy.

### 1.2.3 Zoom cyfrowy/optyczny

Zoom optyczny polega na wydłużaniu ogniskowej aparatu (Rys 5). Dzięki temu po przybliżeniu widać wiele szczegółów niewidocznych na zdjęciu nie przybliżonym. Zoom cyfrowy polega na „rozciąganiu” obrazu uzyskanego na matrycy (Rys 6). W tym przypadku aparat wykonuje normalne zdjęcie, z którego potem wycina odpowiednio powiększony obszar i go rozciąga tak by zapełnił całą przestrzeń. Przy rozciąganiu powstają puste przestrzenie pomiędzy oryginalnymi pikselami które zapełniane są programowo (Na Rys 6 czerwone paski to piksele oryginalnymi, a żółte to piksele wygenerowane). Dlatego też zdjęcia stworzone przy użyciu zooma cyfrowego są gorszej jakości od tych stworzonych za pomocą zooma optycznego

Obraz zawierający linia, diagram, Równolegle, szkic

Opis wygenerowany automatycznie

*Rysunek 5. Schemat zoomowania optycznego [7]*

Obraz zawierający linia, diagram, szkic, design

Opis wygenerowany automatycznie

*Rysunek 6. Schemat zoomowania cyfrowego [7]*

## 1.3 HDR

HDR (ang. High Dynamic Range) to technika polegająca na zwiększaniu zakresu tonalnego zdjęć w fotografii cyfrowej. Dzięki niej jesteśmy w stanie na zdjęciu odwzorować obiekty o bardzo różnych jasnościach. HDR polega na wykonaniu kilku lub kilkunastu ekspozycji o tych samych parametrach wartości przesłony i czułości ISO ale różnych czasach ekspozycji. Tak powstałe pliki są na siebie nakładane co daje dobry obraz wynikowy.

Obraz zawierający niebo, drzewo, na wolnym powietrzu, zrzut ekranu

Opis wygenerowany automatycznie

*Rysunek 7. Przykład działania HDR [9]*

## 1.5 Stabilizacja drgań

Drgania apartu podczas wykonywania zdjęcia mogą spowodować pogorszenie jakość zdjęcia. By temu zapobiec wiele aparatów wyposażonych jest w system stabilizacji drgań. W takim aparacie zamontowany jest czujnik drgań który wykrywa i przekazuje procesorowi kierunki i wielkości drgań. Następnie system stabilizacji drgań wykonuje odpowiednie działania by drganiom przeciwdziałać. Jest to realizowane na dwa główne sposoby:

* Przesuwanie matrycy – aparat jest wyposażony w system który przesuwa matrycę tak by zniwelować drgania (Rys 8)
* Stabilizacja optyczna – elementem kompensującym drgania jest specjalna soczewka



*Rysunek 8. Konstrukcja modelu Pentax K10D [11]*

*Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie*

*Rysunek 8. obiektyw modelu Sony Cyber-shot T30 [11]*

# 2.Zadanie Laboratoryjne

## 2.1 Treść zadania

W ramach zadania laboratoryjnego należało napisać program który wykona zdjęcia/filmy z podłączonej kamery USB i zapisze je w pamięci komputera. Program ten miał wykryć kamery USB, połączyć się z wybranym urządzeniem, zarejestrować obraz w formie pojedynczej klatki. W drugiej części zadania należało wybrać jedną z kilku opcji. My wybraliśmy wykrywanie ruchu.

## 2.2 Opis działania programu

Po włączeniu program wyszukuje wszystkie podłączone kamery. Należy wybrać jedną z nich. Po wybraniu kamery zaczyna się przechwytywanie obrazu i sprawdzanie czy w wideo jest ruch.

## 2.3 Kod programu

import os

import cv2

import ctypes

import numpy as np

from pygrabber.dshow\_graph import FilterGraph

def call\_analyze\_bitmap(frame):

    """

    Calls the C analyze\_bitmap function with a given OpenCV RGB frame.

    Args:

        frame (np.ndarray): The OpenCV frame to process (assumes an RGB frame).

    Returns:

        tuple:

            - bool: True if the previous frame was the same, False otherwise.

            - np.ndarray: The converted frame as an image.

    """

    # Ensure the frame is an RGB image

    if len(frame.shape) != 3 or frame.shape[2] != 3:

        raise ValueError("Input frame must be a 3-channel RGB image.")

    height, width, channels = frame.shape

    if channels != 3:

        raise ValueError("Input frame must have exactly 3 channels (RGB).")

    # Convert the frame to a contiguous array if it's not already

    frame\_data = np.ascontiguousarray(frame, dtype=np.uint8)

    # Allocate memory for the converted pixels (output buffer)

    converted\_pixels = np.zeros\_like(frame\_data, dtype=np.uint8)

    # Call the C function

    result = analyze\_bitmap\_lib.analyze\_bitmap(

        width,

        height,

        frame\_data.ctypes.data\_as(ctypes.POINTER(ctypes.c\_ubyte)),

        converted\_pixels.ctypes.data\_as(ctypes.POINTER(ctypes.c\_ubyte)),

    )

    # Reshape the converted\_pixels back into the image format

    converted\_image = converted\_pixels.reshape((height, width, channels))

    # Return the result and the converted image

    return bool(result), converted\_image

def destroy():

    analyze\_bitmap\_lib.destroy()

def get\_available\_cameras() :

    devices = FilterGraph().get\_input\_devices()

    available\_cameras = {}

    for device\_index, device\_name in enumerate(devices):

        available\_cameras[device\_index] = device\_name

    return available\_cameras

def select\_camera(connected\_cameras):

    print("Dostepne kamery:")

    for index, name in connected\_cameras.items():

        print(f"{index}: {name}")

    while True:

        try:

            selected\_index = int(input("Wybierz kamere: \n>"))

            if selected\_index in connected\_cameras:

                return selected\_index

            else:

                print("Wybrano niepoprawny numer")

        except ValueError:

            print("Wybrano niepoprawny numer")

def main():

    connected\_cameras = get\_available\_cameras()

    if connected\_cameras:

        print("Connected cameras found.")

        selected\_camera = select\_camera(connected\_cameras)

        print(f"You selected Camera {selected\_camera}.")

        cap = cv2.VideoCapture(selected\_camera)

        if cap.isOpened():

            print("Rozpoczynanie przechwytywania. Nacisnij 'q' by wyjsc")

            while True:

                ret, frame = cap.read()

                if not ret:

                    print("Nie udalo sie wykonac zdjecia")

                    break

                move, converted = call\_analyze\_bitmap(frame)

                if move:

                    print("Wykryto ruch")

                cv2.imshow(f"Camera {selected\_camera}", converted)

                if cv2.waitKey(1) & 0xFF == ord('q'):

                    break

            cap.release()

            cv2.destroyAllWindows()

        else:

            print(f"Nie udalo sie otwrzyc kamery {selected\_camera}.")

    else:

        print("Nie znaleziono kamer.")

if \_\_name\_\_ == '\_\_main\_\_':

    print(os.getcwd())

    # Replace 'analyze\_bitmap.dll' with the correct file path on your system

    analyze\_bitmap\_lib = ctypes.CDLL('./Lab3/libBitmapAnalyzer.dll')

    # Define the function signature for the C function

    analyze\_bitmap\_lib.analyze\_bitmap.argtypes = [

        ctypes.c\_int,  # width

        ctypes.c\_int,  # height

        ctypes.POINTER(ctypes.c\_ubyte),  # currentFrame

        ctypes.POINTER(ctypes.c\_ubyte)  # convertedPixels

    ]

    analyze\_bitmap\_lib.analyze\_bitmap.restype = ctypes.c\_char  # return type

    main()

**Plik libBitmapAnalyzer.dll**

**Library.h**

#ifndef UNTITLED2\_LIBRARY\_H

#define UNTITLED2\_LIBRARY\_H

#include <stdio.h>

#include <malloc.h>

#include <string.h>

static unsigned char\* previousFrame = NULL;

static int previousHeight, previousWidth;

\_\_declspec(dllexport)

char analyze\_bitmap(int width, int height, unsigned char \*currentFrame, unsigned char \*convertedPixels);

#endif //UNTITLED2\_LIBRARY\_H

**Library.c**

#include <math.h>

#include "library.h"

void rgbToLab(unsigned char r, unsigned char g, unsigned char b, float \*L, float \*a, float \*b2) {

    // Normalize RGB values to [0, 1]

    float R = r / 255.0f;

    float G = g / 255.0f;

    float B = b / 255.0f;

    // Convert to linear RGB

    R = (R > 0.04045f) ? pow((R + 0.055f) / 1.055f, 2.4f) : R / 12.92f;

    G = (G > 0.04045f) ? pow((G + 0.055f) / 1.055f, 2.4f) : G / 12.92f;

    B = (B > 0.04045f) ? pow((B + 0.055f) / 1.055f, 2.4f) : B / 12.92f;

    // Convert linear RGB to XYZ

    float X = R \* 0.4124564f + G \* 0.3575761f + B \* 0.1804375f;

    float Y = R \* 0.2126729f + G \* 0.7151522f + B \* 0.0721750f;

    float Z = R \* 0.0193339f + G \* 0.1191920f + B \* 0.9503041f;

    // Normalize for D65 illuminant

    X /= 0.95047f;

    Y /= 1.00000f;

    Z /= 1.08883f;

    // Convert XYZ to Lab

    X = (X > 0.008856f) ? pow(X, 1.0f / 3.0f) : (7.787f \* X) + (16.0f / 116.0f);

    Y = (Y > 0.008856f) ? pow(Y, 1.0f / 3.0f) : (7.787f \* Y) + (16.0f / 116.0f);

    Z = (Z > 0.008856f) ? pow(Z, 1.0f / 3.0f) : (7.787f \* Z) + (16.0f / 116.0f);

    \*L = (116.0f \* Y) - 16.0f;

    \*a = 500.0f \* (X - Y);

    \*b2 = 200.0f \* (Y - Z);

}

/// Calculate perceptual color difference

float calculateDeltaE(unsigned char r1, unsigned char g1, unsigned char b1,

                      unsigned char r2, unsigned char g2, unsigned char b2) {

    float L1, a1, b1Lab;

    float L2, a2, b2Lab;

    // Convert both colors to Lab

    rgbToLab(r1, g1, b1, &L1, &a1, &b1Lab);

    rgbToLab(r2, g2, b2, &L2, &a2, &b2Lab);

    // Calculate ΔE (Euclidean distance in Lab space)

    return sqrt(pow(L2 - L1, 2) + pow(a2 - a1, 2) + pow(b2Lab - b1Lab, 2));

}

///\* Analyzes given bitmap, returns the new table ( comparation), as well as the boolean whether there was a change;

///  width, height -> image constraints, currentFrame -> pointer to rgb pixel array;

char analyze\_bitmap(int width, int height, unsigned char \*currentFrame, unsigned char \*convertedPixels) {

    char hasChanged;

    if (previousFrame == NULL) {

        previousFrame = malloc(sizeof(char) \* width \* height \* 3);

        previousHeight = height;

        previousWidth = width;

        memcpy(convertedPixels,currentFrame,sizeof(char)\*width\*height);

        return 1;

    } else if (width != previousWidth || height != previousHeight) {

        free(previousFrame);

        previousFrame = malloc(sizeof(char) \* width \* height \* 3);

        previousHeight = height;

        previousWidth = width;

        memcpy(convertedPixels,currentFrame,sizeof(char)\*width\*height);

        return 1;

    }

    ///Assert malloc successfull

    if (convertedPixels == NULL) {

        return 0;

    }

    ///Declare maximal accepted difference, as well as current diff

    int maxDiff = width\*height/7, diff = 0;

    ///Iterate through pixel array

    for (int i = 0; i < width \* height \* 3; i += 3) {

        const float MOTION\_THRESHOLD = 70.0f;  // Adjust for sensitivity

        /// Updated motion detection logic using perceptual difference

        float pixDiff = calculateDeltaE(

                previousFrame[i], previousFrame[i + 1], previousFrame[i + 2],

                currentFrame[i], currentFrame[i + 1], currentFrame[i + 2]

        );

        if(pixDiff<MOTION\_THRESHOLD){

            memcpy(convertedPixels+i, currentFrame+i, sizeof(char)\*3\*3);

        }

        ///

        else{

            ///Change color of the pixel on red

            convertedPixels[i] = 15; convertedPixels[i+1] = 2;convertedPixels[i+2] = 166;

            diff++;

        }

    }

    if(maxDiff<=diff)

        hasChanged = 1;

    else

        hasChanged = 0;

    return hasChanged;

}

void destroy() {

    free(previousFrame);

}

# 3.Wnioski

Na zajęciach nie udało dokończyć programu. Mimo tego że poprawnie wykrywał urządzenia wykonanie zdjęcia było niemożliwe. Po pracy w domu udało się uzyskać pożądane wyniki

# 4. Źródła

1. <https://avicon.pl/dzialalnoscbr/technologie-wizyjne/cmos-ccd/>
2. <https://en.wikipedia.org/wiki/Bayer_filter>
3. <https://www.researchgate.net/figure/a-RGB-color-space-b-a-Bayer-Color-Filter-Array_fig2_311422563>
4. <https://www.optyczne.pl/14.3-artyku%C5%82-CCD_vs_CMOS_Kr%C3%B3tko_o_CMOS.html>
5. <https://avicon.pl/dzialalnoscbr/technologie-wizyjne/cmos-ccd/>
6. <https://www.optyczne.pl/115-s%C5%82ownik-Matryca_filtr%C3%B3w.html>
7. <https://www.fotopolis.pl/warsztat/porady-fotograficzne/4929-abc-fotografii-cyfrowej-cz-11-zoom-cyfrowy-kontra-zoom-optyczny>
8. <https://www.fotoporadnik.pl/demozaikowanie-en.html>
9. <https://www.optyczne.pl/94-s%C5%82ownik-HDR.html>
10. <https://www.fotopolis.pl/warsztat/porady-fotograficzne/4645-abc-fotografii-cyfrowej-cz-2-stabilizacja-drgan-aparatu>