Urządzenia peryferyjne Laboratorium 5 – Kamery cyfrowe	
Michał Lewandowski, Dominik Kilijan Grupa F	Czwartek 17:30 – 20:30 TNP 23.11.2023

1 Wstęp teoretyczny

1.1 Budowa kamery, matryce

1.1.1 CMOS (Complementary Metal-Oxide-Semiconductor):

- **Zasada działania:** Każdy piksel w matrycy CMOS zawiera własny wzmacniacz i przetwornik analogowo-cyfrowy. Każdy piksel działa niezależnie, co umożliwia równoczesne przetwarzanie danych dla wielu pikseli.
- **Budowa:** Struktura CMOS zawiera matrycę pikseli, a każdy piksel ma własny wzmacniacz i przetwornik ADC. Wzmacniacze są umieszczone bezpośrednio przy pikselach.
- **Zastosowanie:** Matryce CMOS są obecnie powszechnie stosowane w większości cyfrowych aparatów fotograficznych, kamerach wideo, smartfonach i innych urządzeniach.

1.1.2 HAD CCD (Hole-Accumulation Diode CCD):

- **Zasada działania:** HAD CCD to ulepszona wersja tradycyjnego CCD, w której dziurawe diody akumulacyjne zastępują warstwę izolacyjną w tradycyjnym CCD. To pozwala na lepszą efektywność kwantową, co oznacza, że więcej fotonów jest rejestrowanych jako sygnał.
- **Budowa:** HAD CCD ma strukturę podobną do CCD, ale zastosowano dziurawe diody akumulacyjne.
- **Zastosowanie:** HAD CCD były popularne w profesjonalnych kamerach wideo, szczególnie w zastosowaniach wymagających wysokiej jakości obrazu w warunkach słabego oświetlenia.

1.1.3 CCD (Charge-Coupled Device):

- Zasada działania: CCD opiera się na przenoszeniu ładunku elektrycznego przez matrycę pikseli, które są ułożone w siatkę. Kiedy światło pada na piksel, generuje ono ładunek elektryczny proporcjonalny do natężenia światła.

- Budowa: CCD składa się z matrycy pikseli, które są połączone strukturą przenoszenia ładunku. Istnieją trzy główne warstwy w strukturze CCD: warstwa p-substratu, warstwa izolacyjna i warstwa pionowego przesyłająca ładunek.
- Zastosowanie: CCD były pierwotnie popularne w profesjonalnych kamerach wideo i aparatach fotograficznych, ale obecnie są często zastępowane przez matryce CMOS.

1.2 Kolory cyfrowe

1.2.1 Filtry i Balans Bieli:

- **Filtry:** Filtry w fotografii cyfrowej mogą być używane do zmiany charakterystyki kolorów lub do filtrowania niektórych rodzajów światła. Na przykład, filtr UV może pomóc w redukcji efektów niechcianego promieniowania UV. Filtry kolorowe mogą wpływać na balans bieli lub uzyskiwać określone efekty artystyczne.
- **Balans Bieli:** Balans bieli jest kluczowym ustawieniem pozwalającym dostosować kolorystykę zdjęcia w zależności od warunków oświetleniowych. Dostępne są różne ustawienia balansu bieli, takie jak automatyczny, predefiniowane ustawienia (np. dla światła dziennego, sztucznego oświetlenia) oraz możliwość ręcznego dostosowania.

1.2.2 Rozdzielczość Fotografowania:

- Rozdzielczość fotografowania odnosi się do liczby pikseli na zdjęciu. Wyższa rozdzielczość oznacza większą ilość szczegółów, ale także generuje większe pliki. Wybór odpowiedniej rozdzielczości zależy od zamierzonego zastosowania zdjęcia - czy będzie to drukowanie w dużym formacie, prezentacja online czy też archiwizacja.

1.2.3 De-mozaikowanie:

- W przypadku matryc Bayera, które są powszechnie stosowane w matrycach CMOS, piksele są zwykle ustawione w układzie mozaiki, gdzie są one albo czerwone, zielone lub niebieskie. De-mozaikowanie to proces przetwarzania obrazu, który konwertuje ten układ mozaiki na pełny kolor dla każdego piksela. To pozwala na uzyskanie pełnokolorowych obrazów.

1.2.4 Zoom Cyfrowy/Optyczny:

- **Zoom Optyczny:** Zoom optyczny wykorzystuje ruch fizycznych soczewek, aby zbliżyć obraz. Jest to preferowany rodzaj zoomu, ponieważ nie wpływa negatywnie na jakość obrazu.
- **Zoom Cyfrowy:** Zoom cyfrowy polega na elektronicznym powiększeniu obrazu, ale bez fizycznego zbliżania obiektywu. To zwykle prowadzi do utraty jakości, ponieważ obraz jest interpolowany, a nie faktycznie zwiększany za pomocą optyki.

1.3 Techniki HDR, anaglify, stabilizacja drgań, bokeh.

1.3.1 Obraz HDR:

- High dynamic range imaging – technika w fotografii polegająca na wykonaniu kilku ekspozycji tego samego kadru, z których część jest niedoświetlona, a część prześwietlona. Pozwala ona otrzymać obraz sceny charakteryzującej się dużą rozpiętością tonalną.
- Używając tej techniki najczęściej wykonuje się trzy do pięciu fotografii (czasami więcej – w zależności od rozpiętości tonalnej kadru oraz dynamiki aparatu) z poprawnie naświetlonymi cieniami, elementami pośrednimi oraz światłami, z różnicą np. 2 EV (jednak może to być 1 EV albo nawet 3 EV), a następnie łączy się obrazy w jeden plik graficzny. Tak złożony obraz HDR poddawany jest mapowaniu luminancji, dzięki czemu kompresji ulega globalny kontrast zdjęcia, ale zachowane są lokalne zmiany w jasności

1.3.2 Anaglif:

- Jeden z typów rysunku lub fotografii stereoskopowej, dający złudzenie trójwymiaru podczas oglądania za pomocą specjalnych, najczęściej czerwono-cyjanowych (turkusowych), okularów. Sporządzenie anaglifów polega na nałożeniu na siebie dwóch zdjęć, wykonanych z lekkim poziomym przesunięciem, odpowiadającym obrazom dla lewego i prawego oka

1.3.3 Stabilizacja drgań:

A. **Stabilizacja Optyczna (OIS):**
   - W systemie stabilizacji optycznej, elementy optyczne w obiektywie lub samej matrycy aparatu są fizycznie przesuwane, aby zrównoważyć drgania aparatu. To umożliwia uzyskanie stabilniejszego obrazu, nawet przy ruchu kamery.

B. **Stabilizacja Sensora (In-Body Image Stabilization - IBIS):**
   - W aparatach z matrycą, stabilizacja sensora polega na fizycznym przesuwaniu samej matrycy w celu skompensowania drgań. To umożliwia korzystanie z efektu stabilizacji nawet w przypadku obiektywów, które nie posiadają wbudowanej stabilizacji.

C. **Stabilizacja Hybrydowa:**
   - Niektóre systemy łączą stabilizację optyczną z stabilizacją sensora, co nazywane jest stabilizacją hybrydową. To połączenie obu metod może zapewnić jeszcze skuteczniejszą kompensację drgań.

D. **Stabilizacja Cyfrowa:**
   - Niektóre kamery oferują również stabilizację cyfrową, która jest realizowana poprzez obróbkę obrazu w czasie rzeczywistym. Jednak stabilizacja cyfrowa może wpływać na jakość obrazu, ponieważ zazwyczaj obejmuje przycinanie krawędzi kadru.

E. **Użycie Statywu:**

- Oczywiście, jednym z najbardziej skutecznych sposobów unikania drgań jest użycie statywu. To eliminuje ruch kamery i pozwala na uzyskanie ostrego obrazu nawet przy długich czasach naświetlania.

1.3.4 Bokeh:

- **Bokeh** to efekt estetyczny w fotografii, który odnosi się do jakości rozmycia obszarów poza głębią ostrości w obrazie. Charakteryzuje się miękkimi, płynnymi przejściami między obszarami ostrości a obszarami rozmytymi, co tworzy atrakcyjny efekt wizualny.
- Kluczowym elementem bokeh jest umiejętne kontrolowanie głębi ostrości, co oznacza, że fotograf musi skoncentrować się na odpowiednim ustawieniu aparatu. Oto kilka kluczowych aspektów związanych z efektem bokeh:
- Bokeh może być wykorzystywane w celu wyodrębnienia głównego obiektu na pierwszym planie, a także w celu uzyskania estetycznego tła. Warto eksperymentować z różnymi obiektywami, otworami przysłony oraz odległościami, aby osiągnąć pożądany efekt bokeh w zależności od konkretnych potrzeb fotograficznych.

2 Przebieg zadania

Celem zadania było napisanie programu, który będzie umożliwiał zrobienie zdjęcia i filmu oraz zapisania go na inne urządzenie, następnie prowadzący zdecydował o zaimplementowaniu detektora ruchu.

W pierwszej kolejności sprawdziliśmy czy dostępne kamery usb w laboratorium działają oraz jakie są ich nazwy w menadżerze urządzeń, po weryfikacji zabraliśmy się do pisania kodu.

3 Kod programu

```
import cv2
import imutils
from pygrabber.dshow_graph import FilterGraph

# Funkcja do wyświetlania dostępnych kamer
def print_cameras():
    graph = FilterGraph()
    devices = graph.get_input_devices()
    if devices:
        print("Dostępne urządzenia wejściowe:")
        for index, device in enumerate(devices, start=0):
            print(f"{index}. {device}")
    else:
        print("Nie znaleziono urządzeń wejściowych.")
    camera_choice = int(input())
    return camera_choice
```

```python
# Funkcja do ustawiania właściwości kamery
def set_camera_properties(camera, width=None, height=None):
    if width is not None:
        camera.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    if height is not None:
        camera.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

# Funkcja do przechwytywania zdjęcia z kamery
def capture_photo(photo_path='photo.jpg', width=640, height=480):
    # Otwórz kamerę
    camera_index = print_cameras()
    cap = cv2.VideoCapture(camera_index, cv2.CAP_DSHOW)

    if not cap.isOpened():
        print("Błąd: Nie można otworzyć kamery.")
        return

    set_camera_properties(cap, width, height)

    brightness = cap.get(cv2.CAP_PROP_BRIGHTNESS)
    contrast = cap.get(cv2.CAP_PROP_CONTRAST)
    saturation = cap.get(cv2.CAP_PROP_SATURATION)

    while True:
        # Przechwyć klatkę
        ret, frame = cap.read()

        if not ret:
            print("Błąd: Nie można odczytać klatki.")
            break

        # Wyświetl klatkę
        cv2.imshow('Obraz z kamery', frame)

        print(
            f"Aktualne właściwości kamery - Szerokość: {width}, Wysokość:
{height},"
            f" Jasność: {brightness}, Kontrast: {contrast}, Nasycenie:
{saturation}")

        key_pressed = cv2.waitKey(30)
        if key_pressed == ord("q"):
            break
        elif key_pressed == ord('w'):
            brightness += 10
            cap.set(cv2.CAP_PROP_BRIGHTNESS, brightness)
        elif key_pressed == ord('s'):
            brightness -= 10
```

```
            cap.set(cv2.CAP_PROP_BRIGHTNESS, brightness)
        elif key_pressed == ord('e'):
            contrast -= 10
            cap.set(cv2.CAP_PROP_CONTRAST, contrast)
        elif key_pressed == ord('d'):
            contrast += 10
            cap.set(cv2.CAP_PROP_CONTRAST, contrast)
        elif key_pressed == ord('r'):
            saturation -= 10
            cap.set(cv2.CAP_PROP_SATURATION, saturation)
        elif key_pressed == ord('f'):
            saturation += 10
            cap.set(cv2.CAP_PROP_SATURATION, saturation)
        if key_pressed == ord("p"):
            # Zapisz zdjęcie
            cv2.imwrite(photo_path, frame)

            # Zwolnij kamerę
            cap.release()
            print(f"Zdjęcie zapisane jako {photo_path}")

# Funkcja do przechwytywania wideo z kamery
def capture_video(video_path='video.avi', duration_seconds=10, width=640,
height=480):
    # Otwórz kamerę
    camera_index = print_cameras()
    cap = cv2.VideoCapture(camera_index, cv2.CAP_DSHOW)

    if not cap.isOpened():
        print("Błąd: Nie można otworzyć kamery.")
        return

    set_camera_properties(cap, width, height)

    # Pobierz domyślne właściwości kamery
    brightness = cap.get(cv2.CAP_PROP_BRIGHTNESS)
    contrast = cap.get(cv2.CAP_PROP_CONTRAST)
    saturation = cap.get(cv2.CAP_PROP_SATURATION)

    while True:
        # Przechwyć klatkę
        ret, frame = cap.read()

        if not ret:
            print("Błąd: Nie można odczytać klatki.")
            break

        # Wyświetl klatkę
        cv2.imshow('Obraz z kamery', frame)
```

```
        print(
            f"Aktualne właściwości kamery - Szerokość: {width}, Wysokość:
{height},"
            f" Jasność: {brightness}, Kontrast: {contrast}, Nasycenie:
{saturation}")

        key_pressed = cv2.waitKey(30)
        if key_pressed == ord("q"):
            break
        elif key_pressed == ord('w'):
            brightness += 10
            cap.set(cv2.CAP_PROP_BRIGHTNESS, brightness)
        elif key_pressed == ord('s'):
            brightness -= 10
            cap.set(cv2.CAP_PROP_BRIGHTNESS, brightness)
        elif key_pressed == ord('e'):
            contrast -= 10
            cap.set(cv2.CAP_PROP_CONTRAST, contrast)
        elif key_pressed == ord('d'):
            contrast += 10
            cap.set(cv2.CAP_PROP_CONTRAST, contrast)
        elif key_pressed == ord('r'):
            saturation -= 10
            cap.set(cv2.CAP_PROP_SATURATION, saturation)
        elif key_pressed == ord('f'):
            saturation += 10
            cap.set(cv2.CAP_PROP_SATURATION, saturation)
        if key_pressed == ord("p"):
            # Zdefiniuj kodek i utwórz obiekt VideoWriter
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
            out = cv2.VideoWriter(video_path, fourcc, 20.0, (width, height))

            # Przechwyć wideo przez określony czas
            start_time = cv2.getTickCount()
            while (cv2.getTickCount() - start_time) / cv2.getTickFrequency() <
duration_seconds:
                ret, frame = cap.read()
                if not ret:
                    break

                # Zapisz klatkę do pliku wideo
                out.write(frame)

            # Zwolnij kamerę i obiekt VideoWriter
            cap.release()
            out.release()
            print(f"Wideo zapisane jako {video_path}")
```

```
# Funkcja do detekcji ruchu
def motion_detection():
    camera_index = print_cameras()
    cap = cv2.VideoCapture(camera_index, cv2.CAP_DSHOW)

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

    _, start_frame = cap.read()
    start_frame = imutils.resize(start_frame, width=500)
    start_frame = cv2.cvtColor(start_frame, cv2.COLOR_RGB2GRAY)
    start_frame = cv2.GaussianBlur(start_frame, (21, 21), 0)

    alarm = False
    alarm_mode = False
    alarm_counter = 0

    while True:
        _, frame = cap.read()
        frame = imutils.resize(frame, width=500)

        if alarm_mode:
            frame_bw = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
            frame_bw = cv2.GaussianBlur(frame_bw, (5, 5), 0)

            difference = cv2.absdiff(frame_bw, start_frame)
            threshold = cv2.threshold(difference, 25, 255,
cv2.THRESH_BINARY)[1]
            start_frame = frame_bw

            if threshold.sum() > 300:
                alarm_counter += 1
            else:
                if alarm_counter > 0:
                    alarm_counter -= 1

            cv2.imshow("Kamera", threshold)
        else:
            cv2.imshow("Kamera", frame)

        if alarm_counter > 20:
            if not alarm:
                alarm = True
                print("Wykryto ruch!")
                for _ in range(5):
                    if not alarm_mode:
                        break
                    print("ALARM")
                alarm = False
```

```
        key_pressed = cv2.waitKey(30)
        if key_pressed == ord("t"):
            alarm_mode = not alarm_mode
            alarm_counter = 0
        if key_pressed == ord("q"):
            alarm_mode = False
            break

    cap.release()
    cv2.destroyAllWindows()

# Funkcja menu
def menu():
    while True:
        print("Wybierz opcję:")
        print("0. Detekcja ruchu")
        print("1. Zdjęcie")
        print("2. Nagranie wideo")
        print("3. Wyjście")

        choice = input("Podaj swój wybór (0-3): ")

        if choice == "0":
            motion_detection()
        elif choice == "1":
            nazwa_zdjecia = input("Podaj nazwę zdjęcia: ")
            szerokosc = int(input("Podaj szerokość: "))
            wysokosc = int(input("Podaj wysokość: "))
            capture_photo(nazwa_zdjecia, szerokosc, wysokosc)
        elif choice == "2":
            nazwa_wideo = input("Podaj nazwę pliku wideo: ")
            czas_nagrania = int(input("Podaj czas trwania (w sekundach): "))
            szerokosc = int(input("Podaj szerokość: "))
            wysokosc = int(input("Podaj wysokość: "))
            capture_video(nazwa_wideo, czas_nagrania, szerokosc, wysokosc)
        elif choice == "3":
            print("Zamykanie programu.")
            break
        else:
            print("Nieprawidłowy wybór. Podaj liczbę od 0 do 3.")

if __name__ == "__main__":
    menu()
```

4 Efekty kodu na zajęciach

```
C:\Users\1312\PycharmProjects\camera\venv\Scripts\python.exe C:\Users\1312\PycharmProjects\camera\main.py
Input:
0. Motion Detection
1. Capture Photo
2. Capture Video
3. Exit
Enter your choice (0-3): 0
Available input devices:
0. HP TrueVision HD Camera
1. USB2.0 Camera
0
```

Zdjęcie 1. Przedstawia linie konsolowe uzyskane na laboratoriach podczas działania programu

5 Wnioski i podsumowanie

Kod po laboratoriach został przetłumaczony na polski i zostały dodane komentarze.

Dzięki laboratoriom nauczyliśmy się korzystać z OpenCV potężnej biblioteki do przetwarzania obrazów i wideo, która oferuje wiele funkcji gotowych do użycia, umożliwia łatwe przechwytywanie, analizę i manipulację danymi wizualnymi z kamery.

Bibliotekę imutils użyliśmy do zmiany rozmiaru klatek natomiast pygrabber.dshow_graph do uzyskania informacji o kamerach.

Dzięki tym bibliotekom obsługa kamer oraz implementacja funkcjonalności z nimi związana jest bardzo wygodna i intuicyjna dzięki czemu udało nam się zrealizować wszystkie zadania w trakcie zajęć.