Michał Słodowy	Informatyka Przemysłowa	II		II	28.03.2025
Imię i Nazwisko	*Kierunek*	*Grupa*	*Sekcja*	*semestr*	*Data Wykonania ćwiczenia*

OpenCV III

DR HAB. INŻ. SŁAWOMIR GOLAK

Prowadzący

Zadanie 1

- Pobierz obraz *https://media.gettyimages.com/id/200149285-004/photo/crowd-of-women-portrait-elevated-view.jpg?s=2048x2048&w=gi&k=20&c=0__eFQ6_Zlx9f_n7meEW-DtWd7BOsnpHPYLHJYPVQTg=*
- Korzystając z przykładu *https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81* oznacz osoby kółkami i policz ile na obrazie jest osób
- W sprawozdaniu umieść kod, obrazy i wynik
- Obróć obraz o 90 stopni i sprawdź działanie swojego program

```
import cv2
import matplotlib.pyplot as plt


image_path = "resources/images/zd1.jpg"
image = cv2.imread(image_path)


image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Obraz po odczytniu
plt.imshow(image_rgb)
plt.axis("off")
plt.show()

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Wykrywanie twarzy
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))


for (x, y, w, h) in faces:
    center = (x + w // 2, y + h // 2)
    radius = w // 2
    cv2.circle(image_rgb, center, radius, (255, 0, 0), 3)

# Wyświetlenie obrazu z zaznaczonymi twarzami
plt.imshow(image_rgb)
```

```
plt.axis("off")
plt.title(f"Wykryto twarzy: {len(faces)}")
plt.show()




# Obrót
rotated_image = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)

rotated_gray = cv2.cvtColor(rotated_image, cv2.COLOR_BGR2GRAY)

faces_rotated = face_cascade.detectMultiScale(rotated_gray, scaleFactor=1.1, minNeighbors=5, minSize=(30,
30))

rotated_image_rgb = cv2.cvtColor(rotated_image, cv2.COLOR_BGR2RGB)
for (x, y, w, h) in faces_rotated:
    center = (x + w // 2, y + h // 2)
    radius = w // 2
    cv2.circle(rotated_image_rgb, center, radius, (255, 0, 0), 3)

plt.imshow(rotated_image_rgb)
plt.axis("off")
plt.title(f"Po obrocie wykryto twarzy: {len(faces_rotated)}")
plt.show()
```


gettyimages
Credit: Ryan McVay
200149285-004

Wykryto twarzy: 60

Po obrocie wykryto twarzy: 2

Zadanie 2

Tak przerobić program z zadania 1, aby wykryć twarze w dowolnej orientacji.

- Dodać margines do obrazka, żeby w trakcie obrotu nie ginęły twarze (funkcja copyMakeBorder)
- Wykrywać twarze dla kolejnych wersji obrazu pierwotnego obracanego, co 5 stopni
- Po wykryciu na obrócony obrazie twarzy należy jej pozycję przeliczyć na pozycję na obrazie pierwotnym i oznaczyć krzyżykiem.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# PARAMETRY DO EDYCJI
MIN_NEIGHBORS = 4  # Im wyższa wartość, tym mniej fałszywych twarzy (zalecane 8-10)
SCALE_FACTOR = 1.5  # Jak szybko zmniejszamy obraz przy skanowaniu (zalecane 1.2-1.3)
CLUSTER_THRESHOLD = 30  # Jak blisko siebie wykryte twarze są łączone (zalecane 20-30)
MIN_FACE_SIZE = (100, 100)  # Minimalny rozmiar wykrywanej twarzy

# Wczytaj obraz
image_path = "resources/images/zd2.jpg"
image = cv2.imread(image_path)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Dodanie marginesu, aby twarze nie znikały podczas rotacji
border_size = 100
image_with_border = cv2.copyMakeBorder(image, border_size, border_size, border_size, border_size,
cv2.BORDER_CONSTANT, value=(0, 0, 0))

# Klasyfikator do wykrywania twarzy
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Funkcja do obrotu obrazu i przeliczania pozycji wykrytych twarzy
def detect_faces_in_rotations(image, step=5):
    original_h, original_w = image.shape[:2]
    detected_faces = []

    for angle in range(0, 360, step):
        # Obracanie obrazu
        center = (original_w // 2, original_h // 2)
        rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
        rotated_image = cv2.warpAffine(image, rotation_matrix, (original_w, original_h))
```

```
        # Wykrywanie twarzy
        gray = cv2.cvtColor(rotated_image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=SCALE_FACTOR,
minNeighbors=MIN_NEIGHBORS, minSize=MIN_FACE_SIZE)

        # Transformacja pozycji twarzy na oryginalny obraz
        for (x, y, w, h) in faces:
            rotated_point = np.array([[x + w // 2, y + h // 2]], dtype=np.float32)
            rotated_point = np.append(rotated_point, np.ones((1, 1)), axis=1)
            inverse_matrix = cv2.invertAffineTransform(rotation_matrix)
            original_point = np.dot(inverse_matrix, rotated_point.T).T
            detected_faces.append(tuple(original_point[0]))

    # Usuwanie duplikatów twarzy
    final_faces = cluster_faces(detected_faces)
    return final_faces

# Funkcja do grupowania twarzy (eliminuje duplikaty)
def cluster_faces(faces):
    if not faces:
        return []

    clustered_faces = []
    faces = np.array(faces)

    while len(faces) > 0:
        base_face = faces[0]
        distances = np.linalg.norm(faces - base_face, axis=1)
        close_points = faces[distances < CLUSTER_THRESHOLD]

        # Średnia pozycja grupy twarzy
        mean_x = int(np.mean(close_points[:, 0]))
        mean_y = int(np.mean(close_points[:, 1]))
        clustered_faces.append((mean_x, mean_y))

        # Usuwanie już przetworzonych punktów
        faces = faces[distances >= CLUSTER_THRESHOLD]

    return clustered_faces
```

```
# Wykrywanie twarzy
detected_faces = detect_faces_in_rotations(image_with_border)

# Rysowanie wykrytych twarzy
for (x, y) in detected_faces:
    x, y = int(x - border_size), int(y - border_size)  # Przesunięcie do oryginalnych wymiarów
    cv2.drawMarker(image_rgb, (x, y), (255, 0, 0), cv2.MARKER_CROSS, thickness=2)

# Wyświetlenie obrazu
plt.imshow(image_rgb)
plt.axis("off")
plt.title(f"Wykryte twarze: {len(detected_faces)}")
plt.show()
```

Zadanie 3

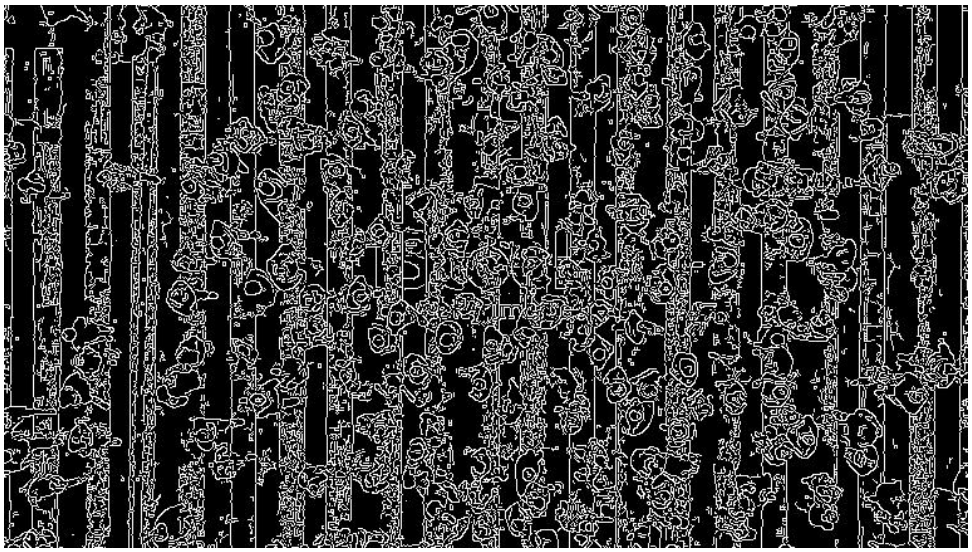
1. Wczytaj wideo osób przechodzących przez przejście
2. Wygeneruj video z klatkami pokazującemu krawędzie obiektów bazując na przykładzie w pliku video.py. W sprawozdaniu zamieść kod i 3 klatki z filmu pierwotnego i wynikowego.
3. Wykreśl na statycznym obrazie trajektorię ruchu mężczyzny w seledynowej koszulce znajdującej się na pierwszej klatce po lewej. W sprawozdaniu zamieść kod i trajektorię.

```
import cv2
import numpy as np
import os

# Ścieżki do plików
input_video_path = "resources/videos/zd3.mp4"
trajectory_output_path = "output/videos/trajectory_farneback.jpg"

# Tworzenie katalogu wyjściowego
os.makedirs("output/videos", exist_ok=True)

# Wczytanie wideo
cap = cv2.VideoCapture(input_video_path)
if not cap.isOpened():
    print(f"Nie udało się otworzyć wideo: {input_video_path}")
    exit()

# Definicja zakresu koloru w przestrzeni HSV (dla koloru R77 G183 B153)
lower_bound = np.array([72, 144, 117])  # Dolna granica HSV
upper_bound = np.array([110, 179, 187])  # Górna granica HSV
```

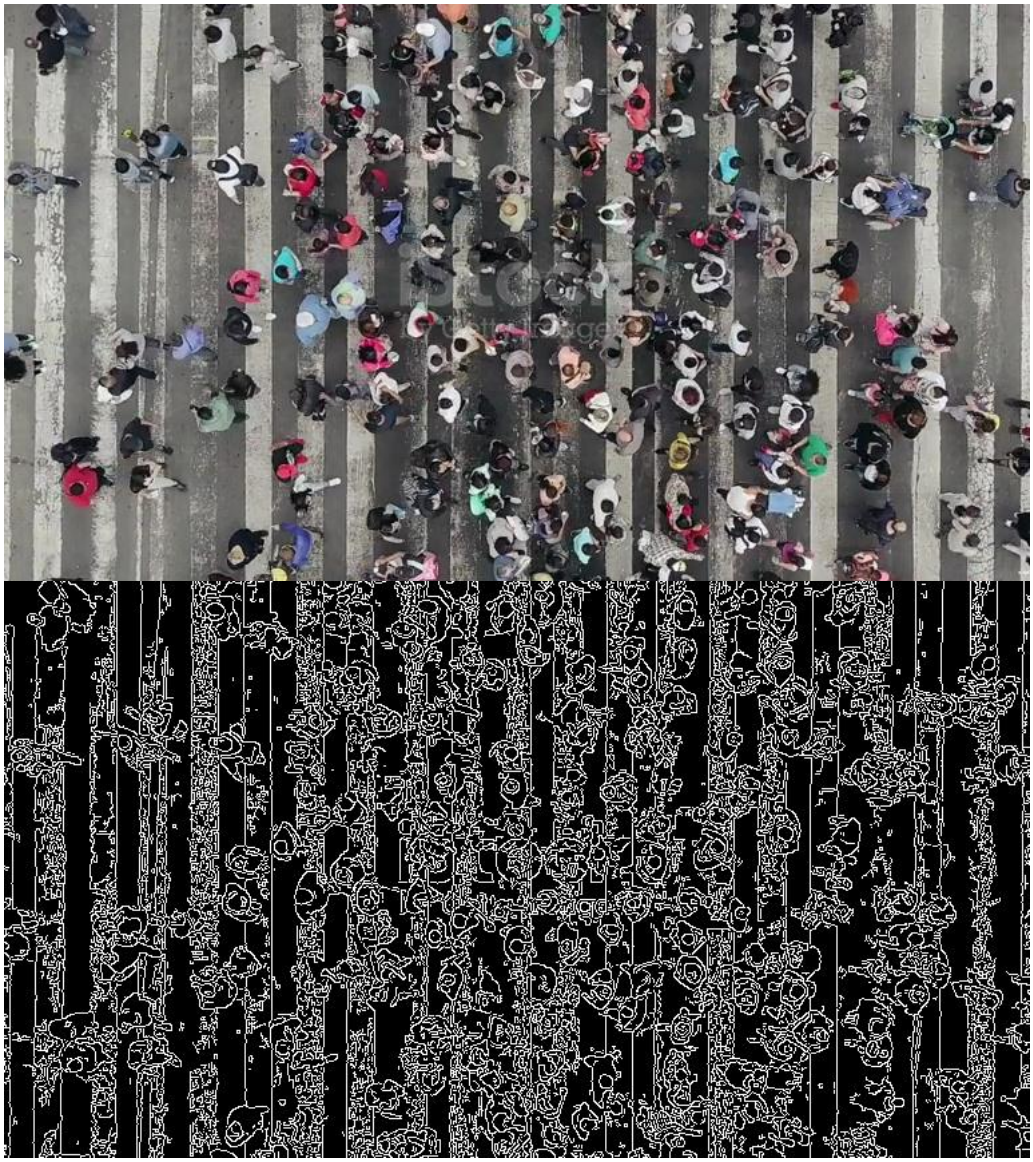
```
# Pobranie pierwszej klatki
ret, frame = cap.read()
if not ret or frame is None:
    print("Nie udało się wczytać pierwszej klatki wideo.")
    cap.release()
    exit()

# Konwersja pierwszej klatki do HSV i wykrycie obiektu na podstawie koloru
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_frame, lower_bound, upper_bound)

# Znajdowanie konturów w masce
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if len(contours) == 0:
    print("Nie znaleziono obiektu o podanym kolorze.")
    cap.release()
    exit()

# Wybór największego konturu (zakładamy, że to obiekt)
largest_contour = max(contours, key=cv2.contourArea)
x, y, w, h = cv2.boundingRect(largest_contour)

# Rysowanie prostokąta wokół obiektu
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 255), 2)

# Lista przechowująca trajektorię ruchu
trajectory_points = [(x + w // 2, y + h // 2)]

# Konwersja pierwszej klatki do skali szarości
prev_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# Przetwarzanie każdej klatki
while cap.isOpened():
    ret, frame = cap.read()
    if not ret or frame is None:
        print("Koniec wideo lub błąd odczytu.")
        break

    # Konwersja bieżącej klatki do HSV i wykrywanie obiektu
```

```
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv_frame, lower_bound, upper_bound)

    # Znajdowanie konturów w masce
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if len(contours) > 0:
        largest_contour = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(largest_contour)
        center = (x + w // 2, y + h // 2)
        trajectory_points.append(center)

    # Rysowanie trajektorii na bieżącej klatce
    for i in range(1, len(trajectory_points)):
        cv2.line(frame, trajectory_points[i - 1], trajectory_points[i], (0, 255, 255), 2)

    # Wyświetlanie wyniku
    cv2.imshow("Optical Flow Tracking", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Rysowanie trajektorii na pierwszej klatce i zapis
for i in range(1, len(trajectory_points)):
    cv2.line(frame, trajectory_points[i - 1], trajectory_points[i], (0, 255, 255), 2)

cv2.imwrite(trajectory_output_path, frame)
print(f"Trajektoria zapisana w pliku {trajectory_output_path}.")

# Zapewnia, że okno z wideo nie zamknie się automatycznie
print("Naciśnij dowolny klawisz, aby zamknąć okno...")
while True:
    cv2.imshow("Optical Flow Tracking", frame)
    if cv2.waitKey(0) & 0xFF:
        break

cap.release()
cv2.destroyAllWindows()
```

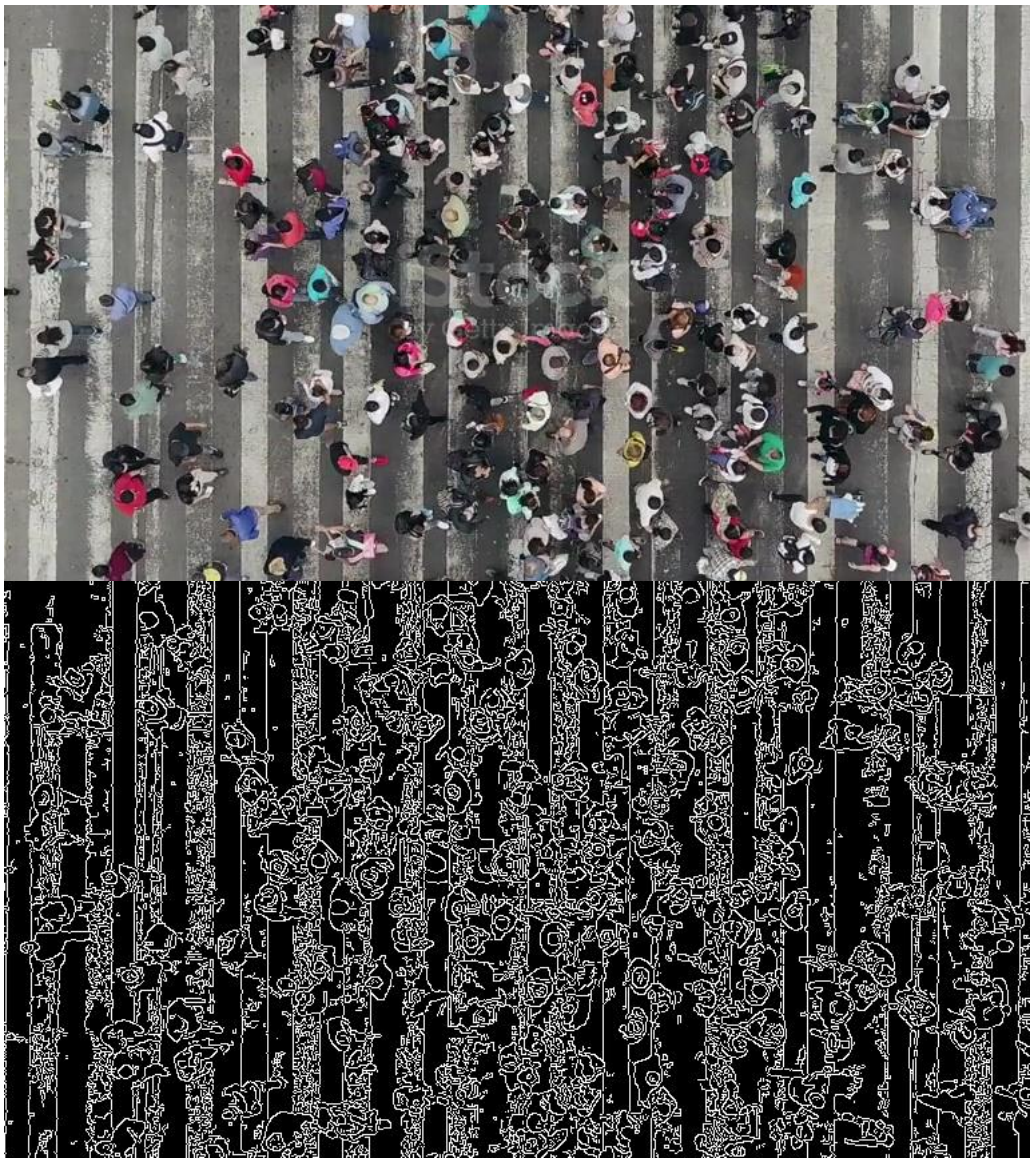
Klatka 0



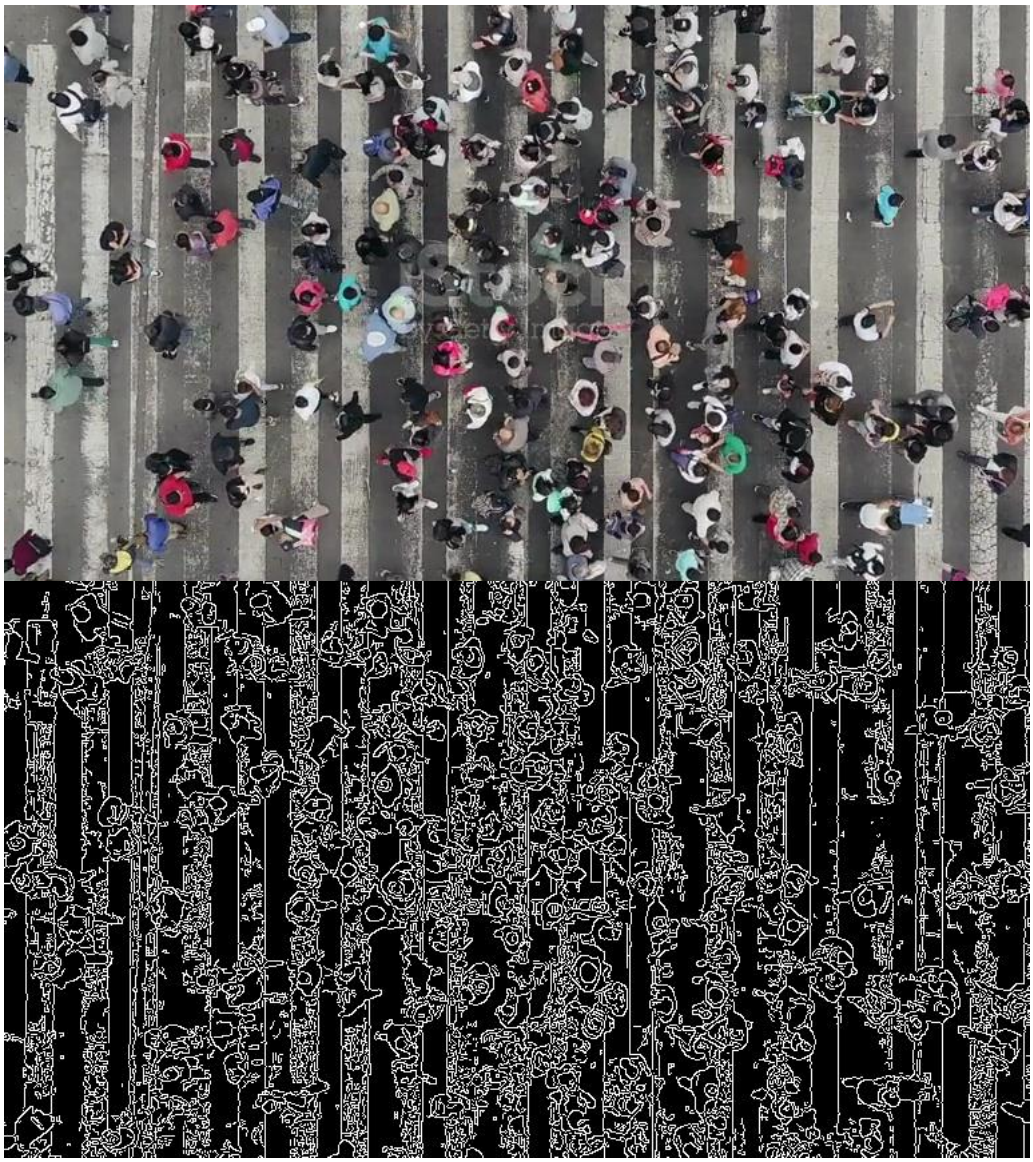
Klatka 30



Klatka 60



Klatka 90



Trajektoria **Gunnar Farneback's algorithm**

(Dense Optical Flow)

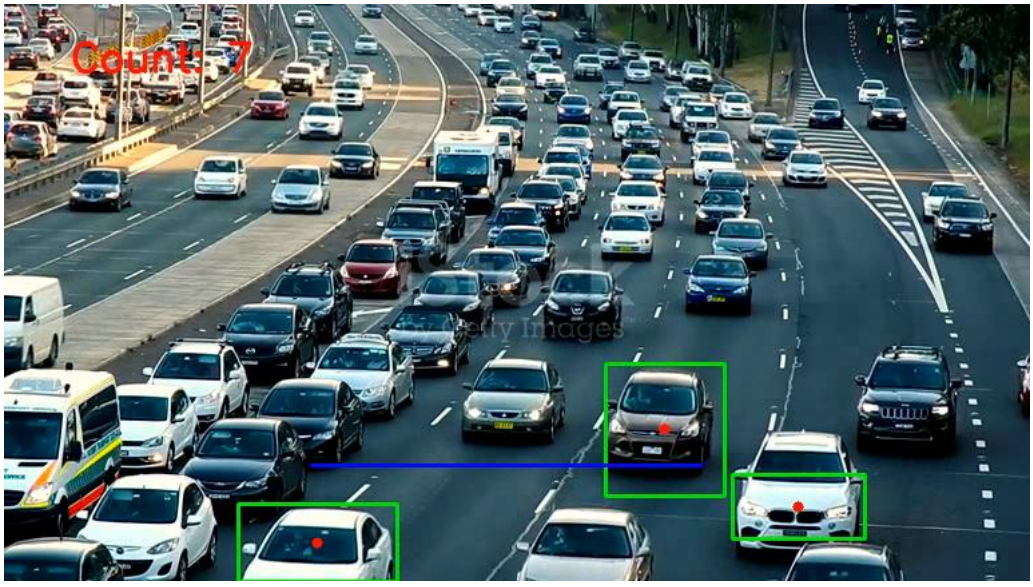
Zadanie 4

1. Wczytaj film. Wygeneruj film zawierający klatki różnicowe (abs) pomiędzy sąsiadującymi klatkami. Weź pod uwagę, że wczytany obraz to uint8 (od 0 do 255).
2. Uzyskaj stałe tło bez samochodów uśredniając wszystkie klatki filmu. Do sprawozdania dodaj kod i efekt.
3. Wyznacz korzystając z operacji morfologicznych i binaryzacji obiekty samochodów. Sprawozdanie powinno zawierać kod, 3 klatki filmu wejściowego i wyjściowego.
4. Bazując na pojawianiu i znikaniu obiektów pojawiających się w obszarze testowym (z progiem liczby pikseli) policz automatycznie ile samochodów przejechało środkowym pasem. Zamieść w sprawozdaniu kod i wynik.

```
import cv2
import numpy as np
import os

# Parametry konfiguracyjne
FLOW_THRESHOLD = 2.0  # Próg dla wartości przepływu optycznego

# Tworzenie folderów na wyniki
os.makedirs("output/videos", exist_ok=True)
os.makedirs("output/images", exist_ok=True)

# Ścieżki do plików
input_video_path = "resources/videos/zd4.mp4"
output_diff_video_path = "output/videos/diff_output.mp4"
output_avg_image_path = "output/images/avg_background.png"

# Wczytanie wideo
cap = cv2.VideoCapture(input_video_path)

# Pobranie właściwości wideo
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Inicjalizacja zapisu wideo różnicowego
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out_diff = cv2.VideoWriter(output_diff_video_path, fourcc, fps, (frame_width, frame_height), isColor=False)
```

```
# Inicjalizacja zmiennych do uśredniania
avg_frame = np.zeros((frame_height, frame_width, 3), dtype=np.float32)
ret, prev_frame = cap.read()
prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY) if ret else None

frame_index = 0

while ret:
    # Dodawanie klatki do uśredniania
    avg_frame += prev_frame.astype(np.float32)

    # Wczytanie kolejnej klatki
    ret, frame = cap.read()
    if not ret:
        break

    # Konwersja na skalę szarości
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Obliczenie różnicy między kolejnymi klatkami
    diff_frame = cv2.absdiff(prev_gray, gray)

    # Zapis klatki różnicowej
    out_diff.write(diff_frame)

    # Aktualizacja poprzedniej klatki
    prev_gray = gray.copy()
    prev_frame = frame.copy()

    frame_index += 1

# Uśrednianie tła
avg_frame /= frame_index
avg_frame = np.uint8(avg_frame)  # Konwersja do uint8

# Zapis uśrednionego tła
cv2.imwrite(output_avg_image_path, avg_frame)

# Zwolnienie zasobów
cap.release()
```

```
out_diff.release()


print(f"Zakończono przetwarzanie. Wygenerowano klatki różnicowe oraz obraz uśredniony tła.")
```




Count: 13


```
import cv2
import numpy as np
import os

# Konfiguracja parametrów
MIN_CONTOUR_AREA = 3000   # Minimalna powierzchnia konturu
KERNEL_SIZE = (5, 5)      # Rozmiar kernela morfologicznego
HISTORY = 500             # Liczba klatek do zapamiętania przez odejmowanie tła
VAR_THRESHOLD = 25        # Czułość na zmiany w tle
MOG_SHADOWS = False       # Włącz/wyłącz detekcję cieni w MOG2
TRACKED_LIFETIME = 3      # Czas życia obiektów w sekundach
# Pozycja linii liczenia
MIDDLE_AREA_TOP_RATIO = 0.8
MIDDLE_AREA_LEFT_RATIO = 0.3
```

```
MIDDLE_AREA_RIGHT_RATIO = 0.68

# Ścieżki do plików
INPUT_VIDEO_PATH = "resources/videos/zd4.mp4"
OUTPUT_VIDEO_PATH = "output/videos/cars_detected.mp4"

# Tworzenie katalogów wyjściowych
os.makedirs("output/videos", exist_ok=True)
os.makedirs("output/images", exist_ok=True)

# Inicjalizacja wideo
cap = cv2.VideoCapture(INPUT_VIDEO_PATH)
if not cap.isOpened():
    raise IOError("Nie udało się otworzyć wideo.")

frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)
frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Obliczenie pozycji linii liczenia
line_y = int(frame_height * MIDDLE_AREA_TOP_RATIO)
line_left = int(frame_width * MIDDLE_AREA_LEFT_RATIO)
line_right = int(frame_width * MIDDLE_AREA_RIGHT_RATIO)

# Inicjalizacja zapisu wideo
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(OUTPUT_VIDEO_PATH, fourcc, fps, (frame_width, frame_height), isColor=True)

# Inicjalizacja algorytmu odejmowania tła
bg_subtractor = cv2.createBackgroundSubtractorMOG2(history=HISTORY, varThreshold=VAR_THRESHOLD,
detectShadows=MOG_SHADOWS)

# Zmienna do liczenia pojazdów oraz lista śledzonych obiektów
car_count = 0
tracked_objects = {}
frame_index = 0

# Ustalanie klatek do zapisania (początek, środek, koniec)
save_frames = [0, frame_count // 2, frame_count - 1]
```

```
saved_frames = 0

while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame_index += 1

    # Zastosowanie algorytmu odejmowania tła
    fgmask = bg_subtractor.apply(frame)
    _, fgmask = cv2.threshold(fgmask, 150, 255, cv2.THRESH_BINARY)  # Zmniejszony próg

    # Operacje morfologiczne
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, KERNEL_SIZE)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel, iterations=2)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_DILATE, kernel, iterations=2)

    # Znalezienie konturów
    contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for cnt in contours:
        if cv2.contourArea(cnt) > MIN_CONTOUR_AREA:
            x, y, w, h = cv2.boundingRect(cnt)
            cx, cy = x + w // 2, y + h // 2

            # Rysowanie prostokąta i środka obiektu
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.circle(frame, (cx, cy), 4, (0, 0, 255), -1)

            # Detekcja przecięcia linii liczenia
            if (y < line_y <= y + h) and (line_left <= cx <= line_right):
                if not any(abs(cx - obj[0]) < 40 and abs(cy - obj[1]) < 40 for obj in tracked_objects):
                    tracked_objects[(cx, cy)] = frame_index  # Rejestracja pojazdu
                    car_count += 1

    # Usuwanie starych wpisów z listy śledzonych obiektów
    tracked_objects = {k: v for k, v in tracked_objects.items() if frame_index - v < fps * TRACKED_LIFETIME}

    # Rysowanie linii liczenia
    cv2.line(frame, (line_left, line_y), (line_right, line_y), (255, 0, 0), 2)
```

```
    cv2.putText(frame, f"Count: {car_count}", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2,
cv2.LINE_AA)

    # Zapisywanie klatek w ustalonych momentach
    if frame_index in save_frames and saved_frames < 3:
        cv2.imwrite(f"output/images/frame_{saved_frames+1}.png", frame)
        cv2.imwrite(f"output/images/mask_{saved_frames+1}.png", fgmask)
        saved_frames += 1

    # Zapis przetworzonej klatki do wideo
    out.write(frame)

# Zwolnienie zasobów
cap.release()
out.release()

print(f"Przetwarzanie zakończone. Wykryto łącznie {car_count} pojazdów.")
```

Źródła

https://medium.com/@andresberejnoi/computer-vision-with-opencv-building-a-car-counting-system-andres-berejnoi-8bcc29fc256

https://pl.freepik.com/darmowe-zdjecie/srednio-strzal-usmiechnieci-ludzie-lezacy-na-trawie_16689019.htm

https://media.gettyimages.com/id/200149285-004/photo/crowd-of-women-portrait-elevated-view.jpg?s=2048x2048&w=gi&k=20&c=0__eFQ6_ZIx9f_n7meEW-DtWd7BOsnpHPYLHJYPVQTg=

https://www.istockphoto.com/pl/filmy/widok-z-lotu-ptaka-na-pieszych-przechodz%C4%85cych-przez-zat%C5%82oczony-ruch-gm1140364351-305135790

https://www.istockphoto.com/pl/filmy/godziny-szczytu-ruchu-w-sydney-gm511613486-87515053