Rozpoznawanie i przetwarzanie obrazów - Projekt dokumentacja

Prowadzący *Mgr inż. Tomasz Serafin*	**Termin zajęć** *WT 11:15 TP*
Grupa *Jan Zemło 259194,* *Karol Rakicki 259088*	**Kierunek** *Informatyka Techniczna*
Temat projektu *Kamera bezpieczeństwa w pojeździe campingowym (#vanlife)*	**Data** *20th September 2023*

Contents

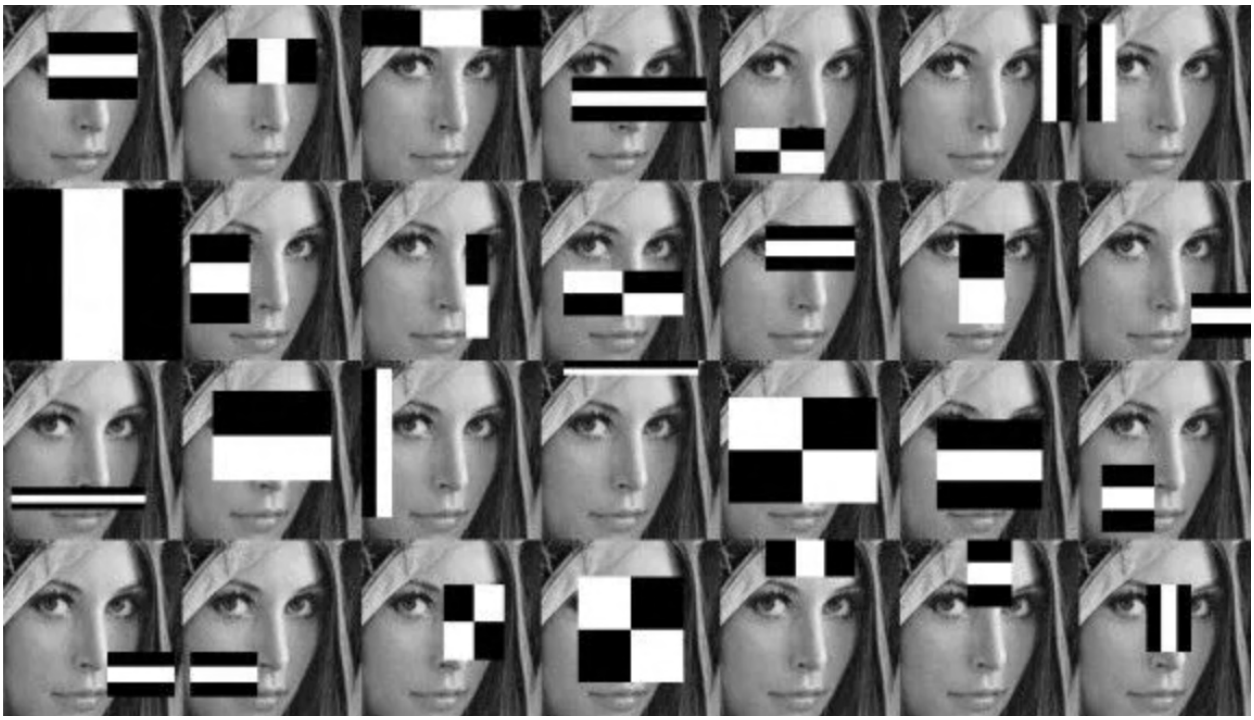
1 Temat projektu

Zaprojektowanie programu, który na podstawie nagrania video z kampera ma za zadanie wykryć sytuacje niebezpieczne i zgłosić odpowiedni alarm.

2 Analiza dziedziny

Projekt będzie implementowany w języku Python, dzięki bibliotece OpenCV. Biblioteka ta daje nam szereg funkcji takich jak:
- wykrywania i rozpoznawania twarzy,
- identyfikowania obiektów,
- klasyfikowania ludzkich działań,
- śledzenia ruchomych obiektów.

Nasz program zaimplementujemy za pomocą klasyfikatora Haar Cascade, który wykrywa obiekty na obrazach niezależnie od ich skali. Algorytm Haar jest stosunkowo łatwy do zaimplementowania oraz może działać w czasie rzeczywistym.



Rysunek 1: Sposób działania funkcji Haar

Idea wykrywanie obiektów opiera się na ekstrakcji cech z zadanego zdjęcia, gdzie każda cecha to różnica pomiędzy sumą pikseli pod białym prostokątem i sumą pikseli pod czarnym prostokątem. Tak przygotowane cechy następnie wykorzystuje się do trenowania modeli kaskadowych, gdzie dane wejściowe to próbka zdjęć zawierających lub nie zadany obiekt (np. twarze). Następnie wybierane są cechy, które najdokładniej wychwytują pożądane obiekty. Cechy podzielone są na odpowiednie grupy, gdzie każda z grup to kolejny etap klasyfikacji. Wybrane okno zaklasyfikowane jest jako zawierające obiekt kiedy przechodzi pozytywnie przez wszystkie etapy.

3 Analiza problemu

Oprogramowanie będzie musiało w czasie rzeczywistym analizować następujące po sobie klatki dostarczone przez kamerę, aby nie doprowadzić do wypadku. Jednym z niebezpieczeństw w czasie podróżowania jest zmęczenie kierowcy. Różne badania sugerują, że około 20% wszystkich wypadków drogowych jest związanych ze zmęczeniem, a na niektórych drogach stosunek ten rośnie nawet do 50%. Kolejne zagrożenie pojawia się, gdy właściciele kampera oddalą się od niego bądź śpią w środku. W tym czasie kamper jest narażony na zniszczenie, a niezabezpieczony ekwipunek na skradzenie.

Dwa pomysły realizowania projektu:

- Wykrywanie poziomu skupienia kierowcy- oprogramowanie analizowałoby czas na jaki kierowca zamyka oczy, gdy przekracza on 5 sekund o potencjalnym zagrożeniu informować będzie kierowcę za pomocą sygnału dźwiękowego i komunikatu tekstowego np. "czas na przerwę”
- Wykrywanie potencjalnego przestępstwa- po wykryciu osoby w pobliżu kampera i po przeanalizowaniu jej zachowania jako podejrzane oprogramowanie rozpoczynałoby nagrywanie w celu późniejszej identyfikacji sprawcy

4 Cele projektu

1. Przetwarzanie wideo transmitowanego z kamery w celu polepszenia jakości obrazu oraz wyeliminowaniu szumów.
2. Zaimplementowanie algorytmu oraz wyuczenie go rozpoznawania twarzy ludzi.
3. Wykrywanie ruchu w pobliżu samochodu oraz monitorowanie twarzy kierowcy.
4. Wykrycie zagrożenia, takiego jak zaśnięcie kierowcy albo zbliżenia się osoby blisko samochodu.
5. Interwencja w razie zagrożenia, w zależności od przypadku zaczęcie nagrywania albo wydanie sygnału głosowego.

5 Założenia projektowe

1. Kamera będzie zainstalowana wewnątrz kampera.
2. Do kamery będzie podłączone urządzenie z zaimplementowanych algorytmem.
3. Program zostanie zaimplementowany w języku Python z użyciem biblioteki OpenCV.
4. Program ma za zadanie zidentyfikować zagrożenie.
5. Program będzie zawiadamiał użytkownika o wystąpieniu zagrożenia.

6 Etap 1

6.1 Cel etapu

W etapie pierwszym mieliśmy za zadanie zebranie materiałów wideo potrzebnego do późniejszych testów działania naszego programu. Materiał wideo nagraliśmy z kampera. Wideo jest w jakości 1080p przy 60 klatkach na sekunde.

6.2 Kadry z zebranych materiałów

6.2.1 Kadr okna

Program ma wykrywać otwarte okno oraz rękę sięgająca do środka.

Rysunek 2: Otwarte okno

Rysunek 3: Otwarte okno z widoczną ręką

Rysunek 4: Zamknięte okno

6.2.2 Kadr drzwi

Program ma wykrywać otwieranie drzwi.

Rysunek 5: Zamknięte drzwi

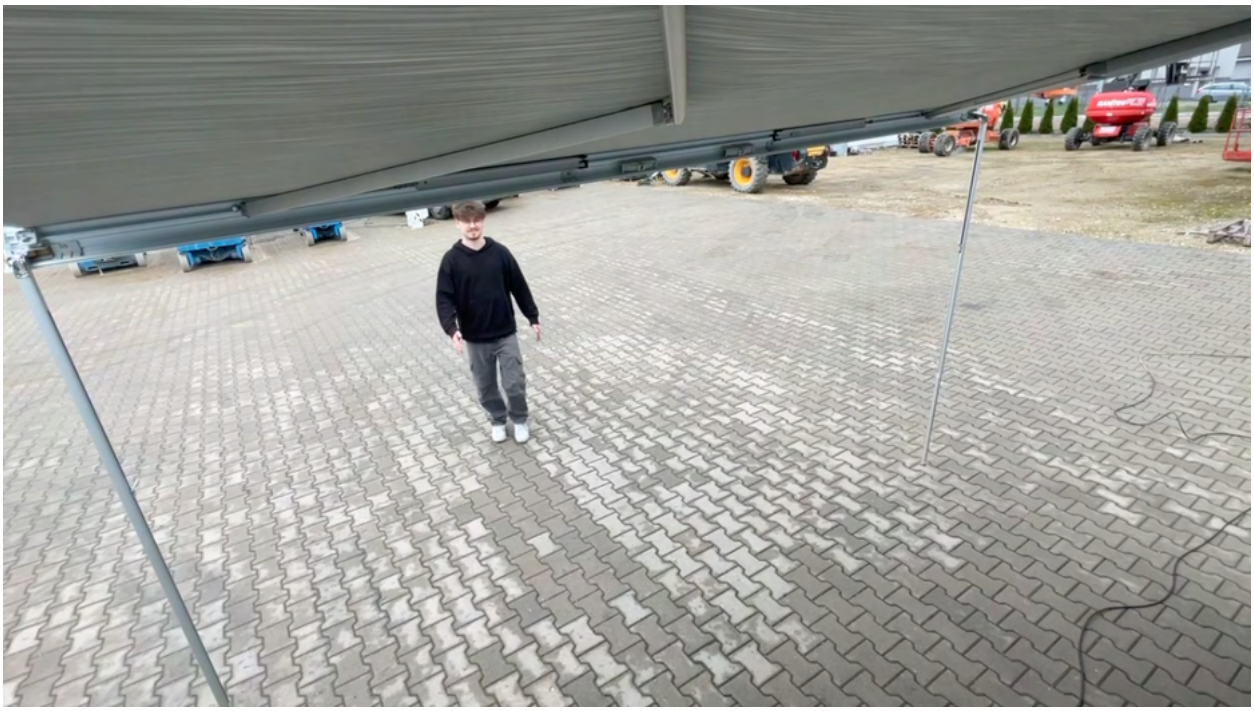
Rysunek 6: Uchylone drzwi

6.2.3 Kadr markizy

Program ma wykrywać wejście na teren pod markizą kampera.

Rysunek 7: Pusty obszar

Rysunek 8: Osoba na obszarze



Rysunek 9: Osoba poza obszarem

6.3 Podsumowanie etapu

Program ma zadanie wykrywać ruch z trzech kadrów. Nagraliśmy materiał wideo o sumarycznej długości około 15 minut, będą one symulować realistyczne przypadki użycia naszego programu.

7 Etap 2

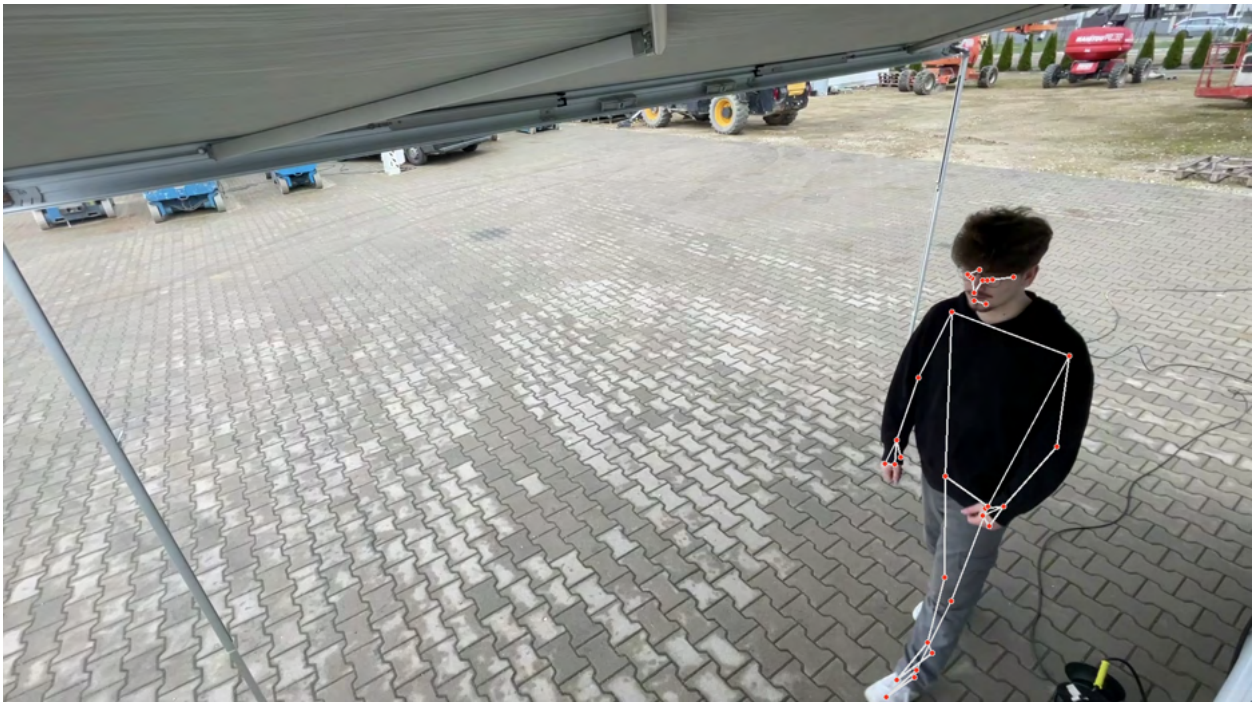
7.1 Cel etapu

Celem etapu 2 było zbudowanie szkieletu programu oraz pierwsze próby działania.

7.2 Wykrywanie ręki i osoby

Do wykrywania rąk w przypadku sceny z oknem oraz do wykrywania osoby znajdującej się w pobliżu markizy została wykorzystana biblioteka MediaPipe. MediaPipe to biblioteka open-source opracowana przez Google, która umożliwia budowanie aplikacji opartych na uczeniu maszynowym (w tym sieci neuronowe i modele głębokiego uczenia), przetwarzaniu wideo i analizie multimediów. Biblioteka MediaPipe zapewnia narzędzia i modele do przetwarzania multimediów, w tym modeli do śledzenia twarzy, detekcji ręki, rozpoznawania gestów, segmentacji obiektów, analizy postawy, rozpoznawania emocji i wiele innych. Biblioteka ta jest dostępna jako wolne oprogramowanie na licencji Apache 2.0, co oznacza, że jest to narzędzie dostępne dla każdego programisty, który chce z niej skorzystać.

Rysunek 10: Markiza



Rysunek 11: Markiza z wykrytą osobą

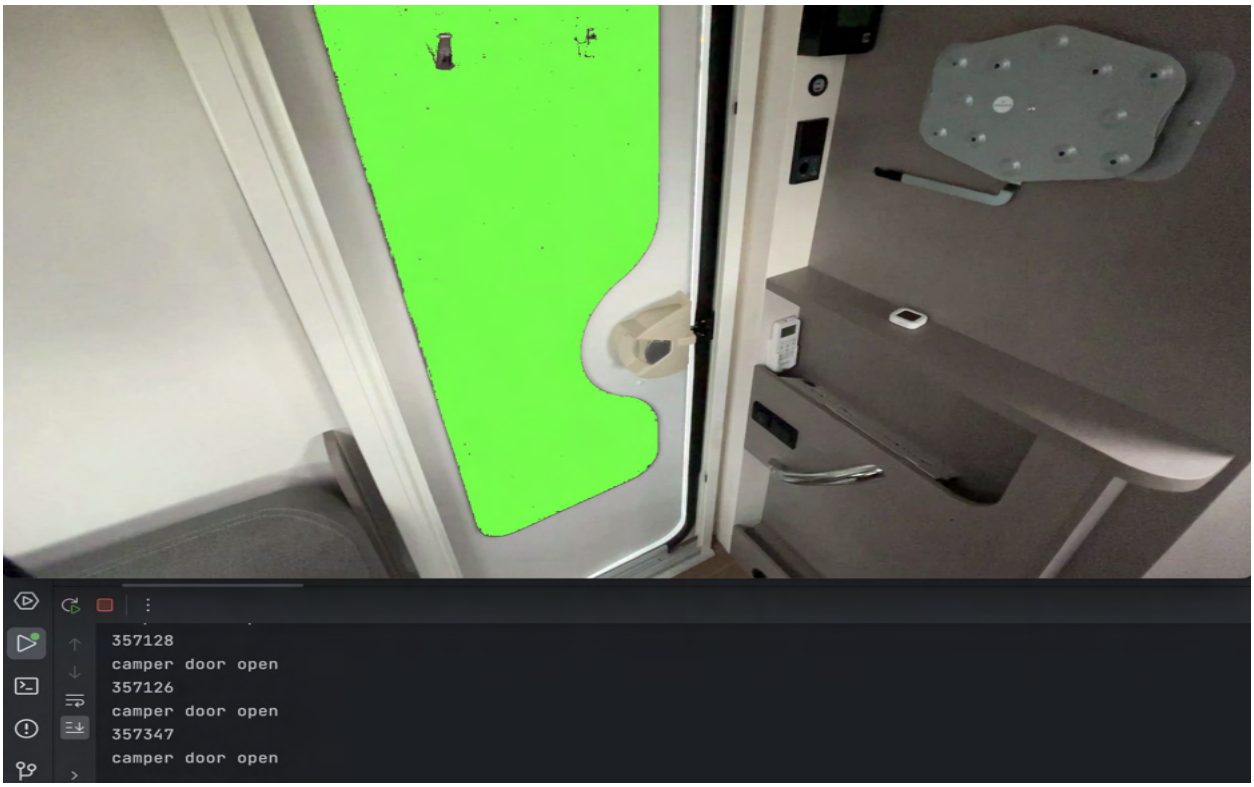
Rysunek 12: Okno

Rysunek 13: Okno z widoczną ręką

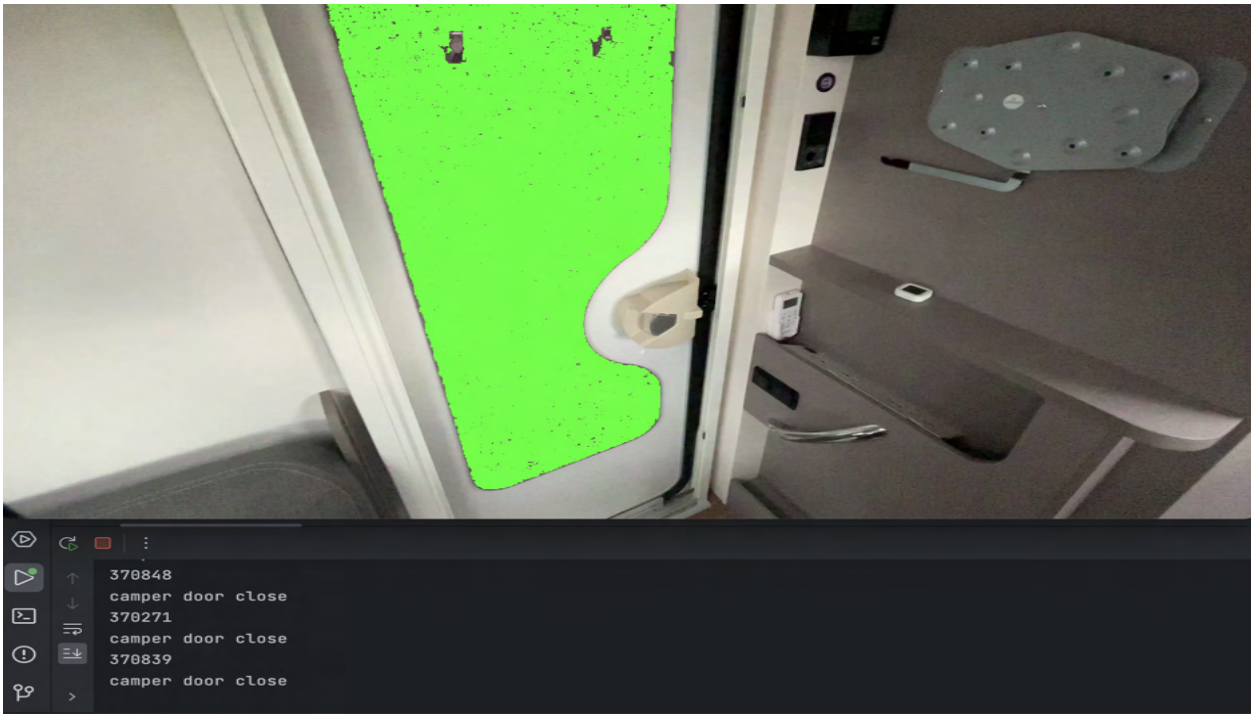
7.3 Wykrywanie otwierania drzwi kampera

Wykrywanie otwartych drzwi polega na wykorzystaniu algorytmu flood fill, który jest stosowany do wykrywania wnętrza drzwi w obrazie. Algorytm flood fill działa poprzez wypełnienie pikseli w określonej odległości od piksela startowego tym samym kolorem. W tym przypadku pikselem startowym jest górny piksel drzwi kampera, a piksele są wypełniane kolorem zielonym.

Następnie porównywany jest rozmiar obszaru zalania z wymiarami drzwi kampera. Jeśli rozmiar obszaru zalania jest mniejszy niż wymiary drzwi, to oznacza, że drzwi są otwarte. W przeciwnym przypadku oznacza to, że drzwi są zamknięte. Algorytm ten jest zastosowany w pętli while, która pozwala na ciągłe wykrywanie otwartych i zamkniętych drzwi w czasie rzeczywistym na podstawie wideo z kamery.



Rysunek 14: Drzwi otwarte



Rysunek 15: Drzwi zamknięte

7.4 Posumowanie etapu

Działanie programów zostało przetestowane na zebranych materiałach wideo z etapu 1. Program w czasie rzeczywistym wykrywa i podąża za ręką oraz osobą. Otwarte drzwi także są poprawnie wykrywane.

8 Etap 3

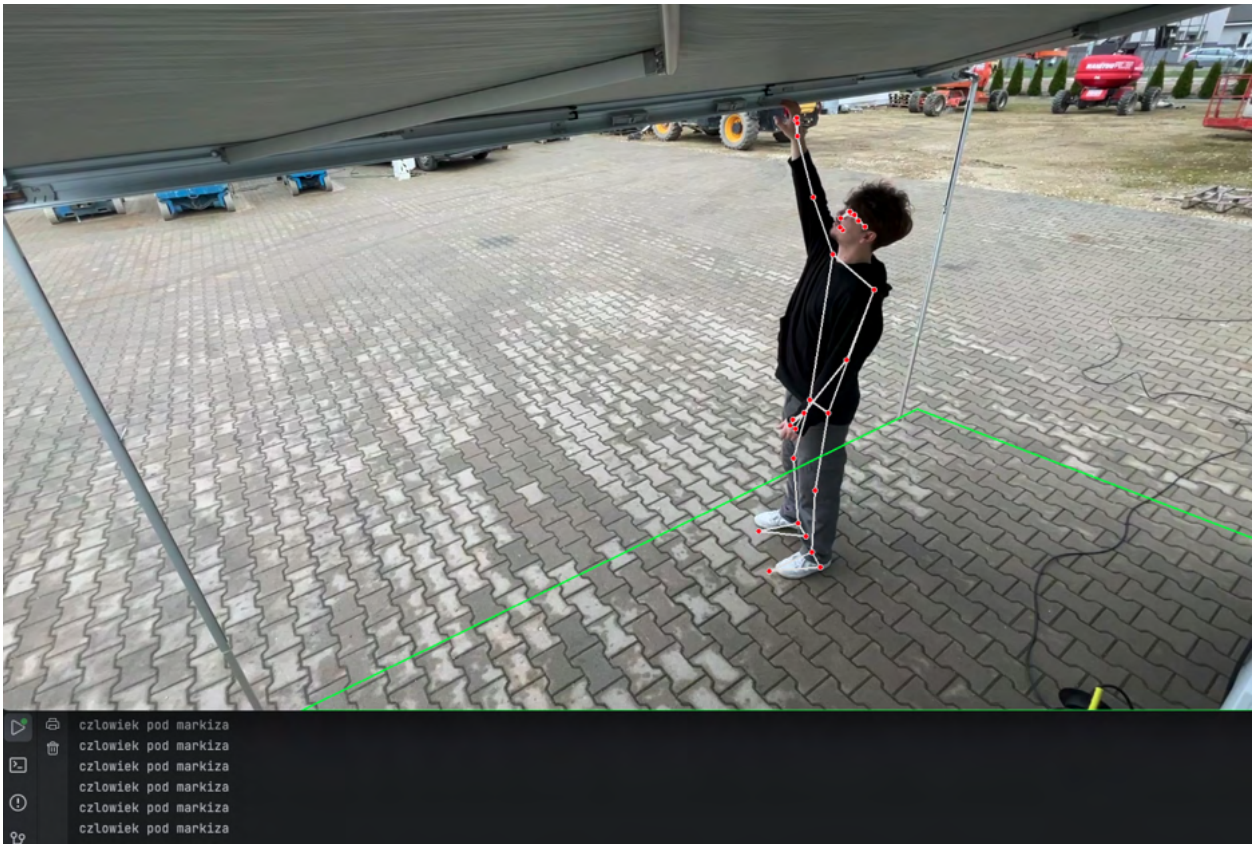
8.1 Cel etapu

Celem etapu 3 było udoskonalenie programów oraz wykrycie ewentualnych błędów.

8.2 Wykrywanie osoby pod markizą

W scenie pod markizą po zmianach w kodzie osoba jest wykrywana przez cały czas. Jednak program sygnalizuje, gdy osoba znajdzie się pod markizą. Interesujący nas obszar jest widoczny na obrazach jako zielony czworokąt. W tej scenie nie znaleźliśmy żadnych problemów.

Rysunek 16: Markiza



Rysunek 17: Markiza z wykrytą osobą

8.3 Wykrywanie ręki przez okno

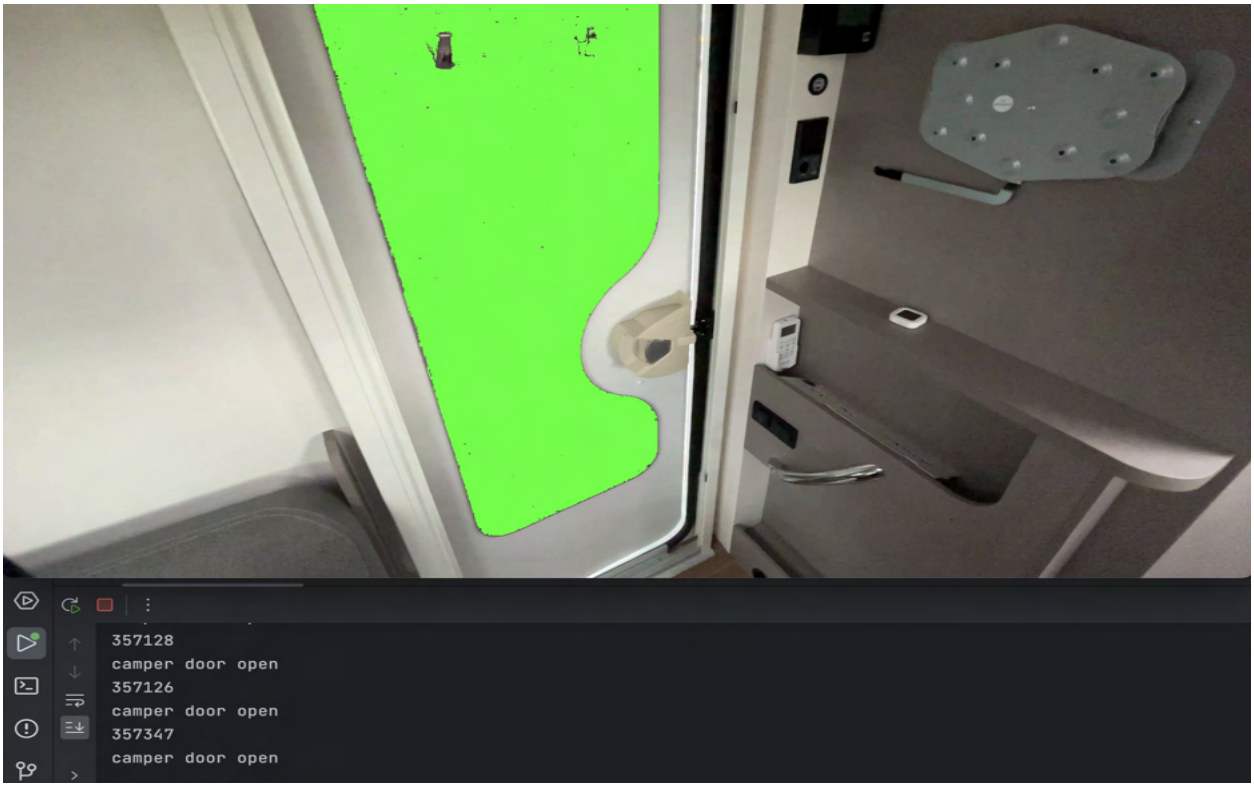
W tej scenie problemem było, że nie zawsze wykrywało rękę, teraz zmniejszyliśmy stopień podobieństwa potrzebny zakwalifikowania obiektu jako ręki, ale pojawił się nowy błąd, rękę wykrywa też w odbiciu okna. Problem ten rozwiążemy poprzez wyłączenia z obszaru wykrywania okna.

Rysunek 18: Ręka wykryta

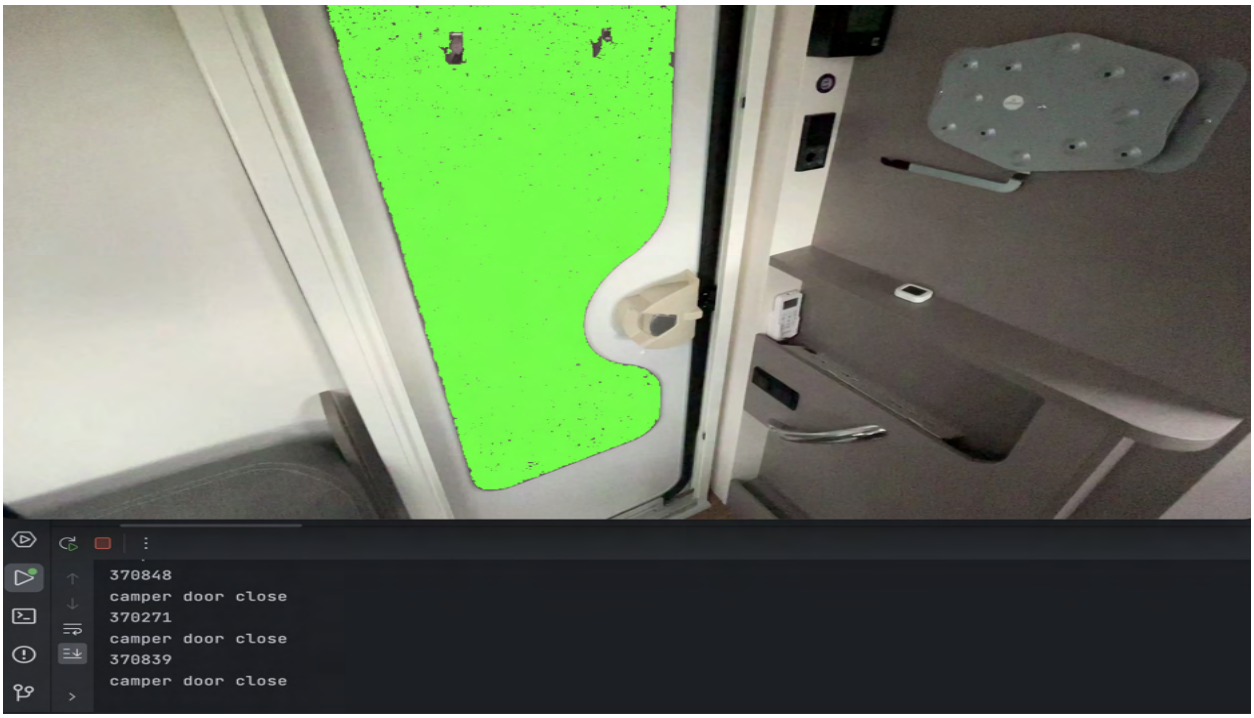
Rysunek 19: Ręka wykryta w oknie

8.4 Wykrywanie otwierania drzwi kampera

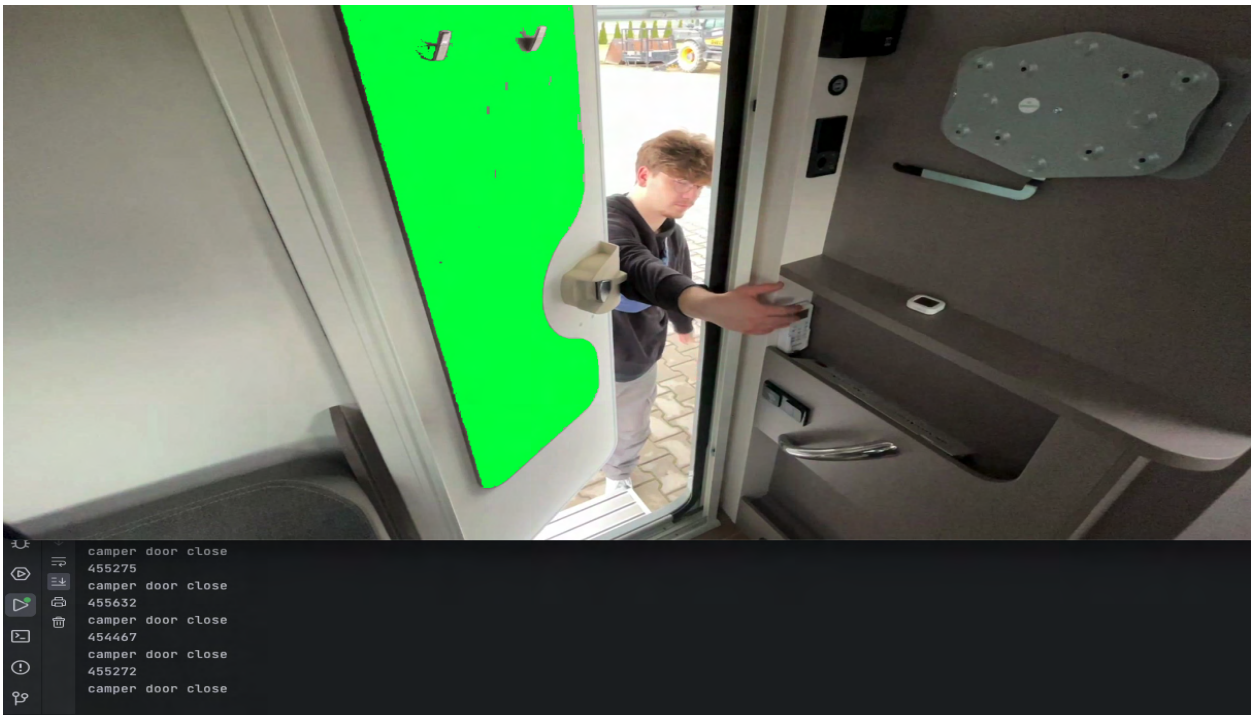
W wykrywaniu drzwi poprawiliśmy dokładną lokalizację początkowego punktu, od którego liczymy piksele podobne kolorem. Następnie porównujemy czy jest to rozmiar większy niż liczba podobnych pikseli przy zamkniętych drzwiach. Niestety program czasami nie działa jak drzwi są otwarte prawie w całości, ponieważ wtedy oświetlenie pada na resztę drzwi i obszar jest większy niż powinien



Rysunek 20: Drzwi otwarte



Rysunek 21: Drzwi zamknięte



Rysunek 22: Drzwi z błędem

8.5 Posumowanie etapu

Programy działają w większości, będziemy je jeszcze udoskonalać. Scena markizy pozostanie już taka, bo nie znaleźliśmy żadnych błędów. W scenie wykrywania dłoni będziemy mieć dynamiczny obszar, gdzie jest ona wykrywana.

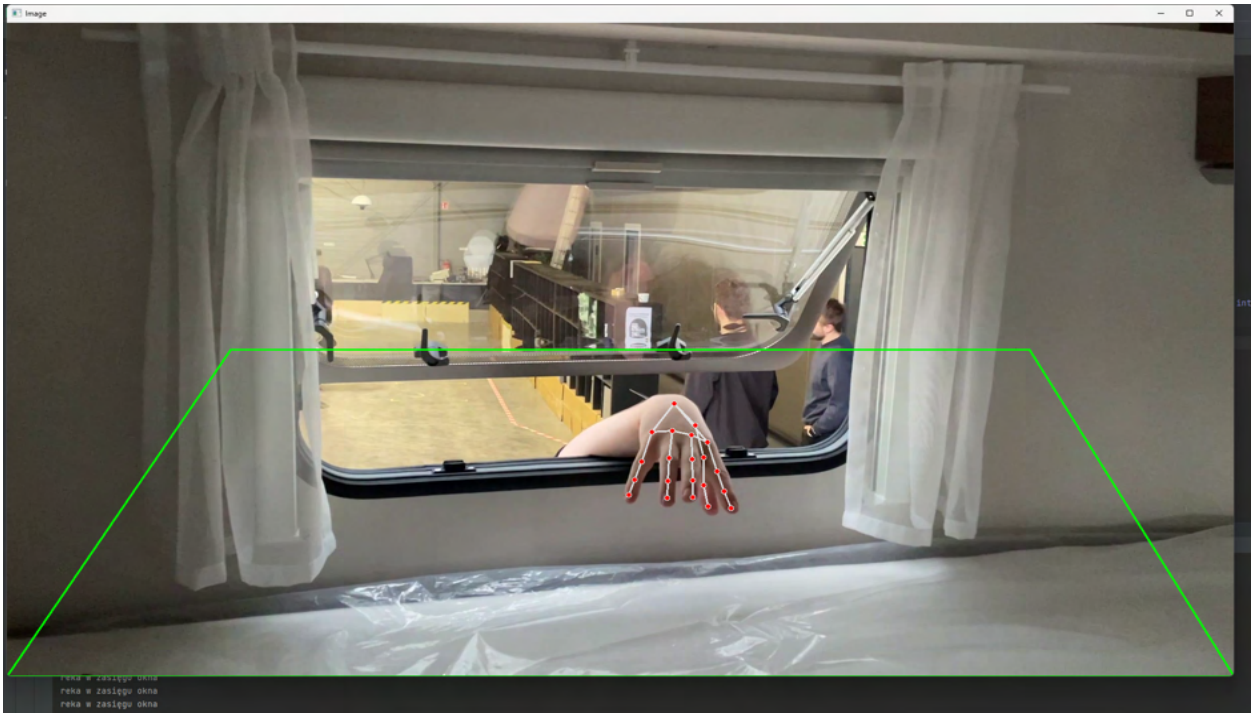
9 Etap 4

9.1 Cel etapu

Celem etapu było usprawnienie działania wykrywania ręki na scenie z oknem.

9.2 Implementacja

W scenie z oknem podobne jak w scenie z markizą ustaliśmy interesujący nas teren i gdy ręka zostanie w nim wykryta program to sygnalizuje. Teren ten jest zaznaczony zielonym czworokątem na obrazku.



Rysunek 23: Działanie programu

9.3 Podsumowanie etapu

Dzięki ustaleniu odpowiedniego interesującego nas terenu rozwiązujemy problem związany z wykrywaniem odbicia ręki w oknie.

10 Podsumowanie projektu

Celem projektu było stworzenie programu, który będzie wykrywał i analizował różne sceny na podstawie zebranych materiałów wideo. Projekt składał się z czterech etapów.

W etapie 1 zebrano materiały wideo, które symulowały realistyczne przypadki użycia programu. Materiały obejmowały różne sceny, takie jak otwarte okno z ręką sięgającą do środka, otwierane drzwi kampera oraz wejście na teren pod markizą kampera. Materiały były nagrywane z kamery o jakości 1080p przy 60 klatkach na sekundę.

W etapie 2 zbudowano szkielet programu i przeprowadzono pierwsze próby działania. Wykorzystano bibliotekę MediaPipe do wykrywania ręki i osoby na zebranych materiałach wideo. Program w czasie rzeczywistym wykrywał ruch ręki i osoby, oraz otwieranie drzwi kampera.

W etapie 3 dokonano udoskonaleń programów oraz wykrywania ewentualnych błędów. Poprawiono wykrywanie osoby pod markizą i wykrywanie ręki przez okno. W przypadku wykrywania ręki przez okno zastosowano zielony czworokąt, który oznaczał interesujący nas obszar. Wykrywanie otwierania drzwi kampera również zostało usprawnione.

W etapie 4 skoncentrowano się na dalszym usprawnianiu wykrywania ręki na scenie z oknem. Wykorzystano ustalony interesujący nas obszar, w którym ręka jest wykrywana, aby rozwiązać problem związany z wykrywaniem odbicia ręki w oknie.

W rezultacie projekt zakończył się sukcesem, programy są w stanie wykrywać i analizować różne sceny na zebranych materiałach wideo. Dzięki iteracyjnemu podejściu udało się wprowadzić udoskonalenia i rozwiązać napotkane problemy.

11 Listing kodu

11.1 Scena wyrkywania rąk - main.py

```
import cv2
import mediapipe as mp
import numpy as np

def sprawdz_wspolrzedne(x, y, shp):
    num_vertices = len(shp)
    inside = False

    # sprawdzanie czy punkt jest w zasi gu okna
    # shp: lista zawieraj ca wierzcho ki obszaru w kolejno ci przeciwnie do ruchu wskaz wek
    # dlatego sprawdzamy po kraw dziach czy punkt jest na zewn trz
    for i in range(num_vertices):
        x1, y1 = shp[i]
        x2, y2 = shp[(i + 1) % num_vertices]

        if (y1 < y and y2 >= y) or (y2 < y and y1 >= y):
            if x1 + (y - y1) / (y2 - y1) * (x2 - x1) < x:
                inside = not inside

    return inside

cap = cv2.VideoCapture("IMG_8456.mov")

mpHands = mp.solutions.hands
hands = mpHands.Hands(False, 2, 1, 0.15, 0.3)
mpDraw = mp.solutions.drawing_utils

# Wsp  rz dne pikseli punkt w charakterystycznych
punkty = []

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    #print(results.multi_hand_landmarks)
```

```
    # czyszczenie listy punkt w
    punkty.clear()

    if results.multi_hand_landmarks:
        for handsMulti in results.multi_hand_landmarks:
            mpDraw.draw_landmarks(img, handsMulti, mpHands.HAND_CONNECTIONS)
            a, b = int(handsMulti.landmark[mpHands.HandLandmark.RING_FINGER_TIP].x * img.shape[1
            punkty.append((a, b))

    shp = [(0, 1080), (1920, 1080), (1600, 540), (350, 540)]

    #sprawdzanie czy reka jest w zasiegu okna
    reka = False
    for x, y in punkty:
        if sprawdz_wspolrzedne(x, y, shp):
            reka = True
    if reka:
        print("reka␣w␣zasi gu␣okna")

    # rysowanie pola okna
    shp = np.array(shp, np.int32).reshape((-1, 1, 2))
    cv2.polylines(img, [shp], True, (0, 255, 0), 2)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

11.2 Scena wykrywania osób pod markizą - scena_markiza.py

```
import cv2
import mediapipe as mp
import numpy as np


def sprawdz_wspolrzedne(x, y, shp):
    num_vertices = len(shp)
    inside = False

    # sprawdzanie czy punkt jest pod markiza
    # shp: lista zawieraj ca wierzcho ki obszaru w kolejno ci
    # przeciwnie do ruchu wskaz wek zegara
    # dlatego sprawdzamy po kraw dziach czy punkt jest na zewn trz
    for i in range(num_vertices):
        x1, y1 = shp[i]
        x2, y2 = shp[(i + 1) % num_vertices]

        if (y1 < y and y2 >= y) or (y2 < y and y1 >= y):
            if x1 + (y - y1) / (y2 - y1) * (x2 - x1) < x:
                inside = not inside

    return inside


cap = cv2.VideoCapture("IMG_8460.mov")

mpPose = mp.solutions.pose
pose = mpPose.Pose()
mpDraw = mp.solutions.drawing_utils
```

```
# Wsp  rz dne pikseli punkt w charakterystycznych
punkty = []

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = pose.process(imgRGB)

    # czyszczenie listy punkt w
    punkty.clear()

    # dodawanie punkt w do listy
    if results.pose_landmarks is not None:
        for landmark in results.pose_landmarks.landmark:
            a, b = int(landmark.x * img.shape[1]), int(landmark.y * img.shape[0])
            punkty.append((a, b))

    #rysowanie lini cz owieka
    mpDraw.draw_landmarks(img, results.pose_landmarks, mpPose.POSE_CONNECTIONS)

    shp = [(460, 1080), (1920, 1080), (1920, 820), (1400, 620)]

    #sprawdzanie czy czlowiek jest pod markiza
    czlowiek = False
    for x, y in punkty:
        if sprawdz_wspolrzedne(x, y, shp):
            czlowiek = True
    if czlowiek:
        print("czlowiek␣pod␣markiza")

    #rysowanie pola markizy
    shp = np.array(shp, np.int32).reshape((-1, 1, 2))
    cv2.polylines(img, [shp], True, (0, 255, 0), 2)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

11.3 Scena wykrywania otwieranych drzwi - drzwi.py

```
import cv2
from numpy import *

cap = cv2.VideoCapture('/Users/janzemlo/Desktop/ripo/video/IMG_8463.mov')


while True:
    success, img = cap.read()
    height, width, channels = img.shape
    mask = zeros((height+2, width+2), uint8)

    #maximum distance to start pixel:
    diff = (2, 2, 2)


    door_top_pixel = (570, 30)

    # wykryj wn trze drzwi kampera
    retval, rect, _, _ = cv2.floodFill(img, mask, door_top_pixel, (0, 255, 0), diff, diff)
```

```
print(retval)
# por wnaj rozmiar obszaru zalania z wymiarami drzwi kampera
if (retval > 361000):
    print("camper door close")
else:
   print("camper door open")

cv2.imshow("Image", img)
cv2.waitKey(1)
```

12 Bibliografia

References

https://miroslawmamczur.pl/wykrywanie-twarzy-real-time-w-15-liniach-kodu-w-python/
https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d
https://developers.google.com/mediapipe
https://developers.google.com/mediapipe/solutions/guide