Wrocław, 19 listopada 2023

Przemysł 4.0

Laboratorium 6

System detekcji gestów cz. 2

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

Celem laboratorium jest dokończenie systemu detekcji gestów, rozpoczętego w ramach poprzedniego laboratorium. Podczas tego etapu nacisk zostanie położony dodatkowy trening modelu detekcji oraz przeniesienie go na `Raspberry Pi` w celu stworzenia samodzielnego systemu.

2 Zadania

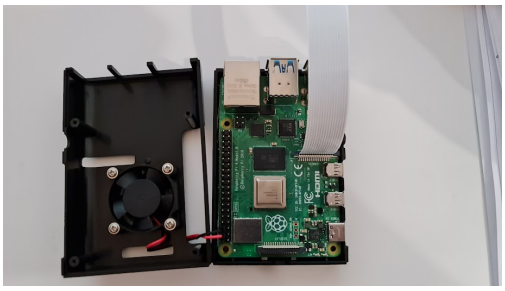
Przypomnienie kompletnej listy zadań w ramach obu laboratoriów:

1. Zebranie zdjęć do treningu.
2. Oznaczenie zdjęć.
3. Trening modelu.
4. Budowa systemu do detekcji gestów z wykorzystaniem `Raspberry Pi`.

3 Opis zadań

W ramach zadania należy zbudować układ detekcji gestów z wykorzystaniem `Raspberry Pi` oraz kamery `HD F Night Vision OV5647 5Mpx - IR`. Dla ułatwienia, zdjęcie kamery przedstawiono na Rys, 1. Moduł kamery ze regulacją ostrości, czyli możliwością regulacji ostrości podłączany do dedykowanego złącza minikomputera `Raspberry Pi`. Urządzenie posiada matrycę `CCD` 1/4" o rozdzielczości 5 `Mpx`, wspiera tryb `HD 1080p`. `Raspberry Pi` posiada sprzętowe wsparcie dla obsługi tej kamery, dzięki temu urządzenie nie zużywa mocy obliczeniowej procesora. Dzięki dołączonym do zestawu modułom podświetlania `IR 850` nm o mocy $3W$ zestaw umożliwia nagrywanie w nocy, nawet w bardzo zaciemnionym miejscach. Ta wersja posiada możliwość zmiany ostrości za pomocą pokrętła na obiektywie.

Rysunek 1: Kamera `HD F Night Vision OV5647 5Mpx - IR`



(a) Podłączenie do `Raspberry Pi`

(b) Podłączenie do kamery przy użyciu taśmy

Rysunek 2: Montaż kamery

Kamera

Etap należy rozpocząć od **poprawnego** podłączenia kamery przy użyciu złącza `CSI`. W celu podłączenia taśmy do `Raspberry Pi` trzeba w pierwszej kolejności otworzyć obudowę (odkręcamy 4 śrubki na rogach podstawy) jak na Rys. 2a. Aby odblokować złącze należy delikatnie unieść do góry czarny plastik, dopiero wtedy możemy wsunąć taśmę (proszę zwrócić uwagę na złącza). Taśmę należy wsunąć do końca i ponownie zablokować blokadę delikatne naciskając plastik do dołu. Analogicznie postępujemy z drugą końcówką taśmy, aby wpiąć ją bezpośrednio do bliźniaczego złącza przy kamerze (Rys. 2b), z tą różnicą, że w tym wypadku należy pociągnąć blokadę do siebie (w poziomie). **Wciskanie na siłę taśmy**, bez odblokowania blokady, może prowadzić do **zniszczenia złącz na taśmie**: Po poprawnym podłączeniu kamery, trzeba jeszcze ją **włączyć w ustawianiach systemu**.

```
sudo raspi-config
```

W menu głównym należeń wybrać opcję `Interfacing Options`, a potem `Camera`. Następnie trzeba potwierdzić chęć włączenia kamery. Teraz warto przetestować działanie kamery prostym poleceniem:

```
raspistill -n -o test.jpg
```

W bieżącym folderze powinien się pojawić plik o nazwie `test.jpg` z naszym zdjęciem. Opcja `-n` wyłącza podgląd. Domyślnie zdjęcie zostanie zrobione po $5s$. Inne przydatne parametry:

- `-t 100` – zmiana opóźnienia na $100ms$, zalecana minimalna wartość,
- `-rot 180` – obrót o 180^o,
- `-hf` – odbicie w poziomie,
- `-vf` – odbicie w pionie,
- `-w 640` – ustawienie szerokości na $640px$,
- `-h 480` – ustawienie wysokości na $480px$,
- `-dt` – dodanie daty.
- `-q 100` – ustawienie jakości na 100.

Jeśli kamera działa poprawnie może przejść do dalszej części zadania. W przypadku gdy obraz jest niewyraźny, należy dostosować ostrość kamery. Ostrością kamery sterujemy poprzez obrót obiektywu na kamerce. Proszę pamiętać, że celem zadania jest wykrywanie gestów wykonywanych dłonią, więc przy ustawianiu kamery należy zadbać o to, żeby dłoń była widoczna w pewnym zakresie odległości.

YOLOv5

Należy zacząć od pobrania odpowiedniego modelu `best.pt` z naszego `Google Colab`. Kolejnym krokiem jest zainstalowanie niezbędnych bibliotek:

```
git clone https://github.com/ultralytics/yolov5  # clone
cd yolov5
pip install -r requirements.txt  # install
sudo pip3 install opencv-contrib-python
sudo pip3 install opencv-python
```

Następnie wystarczy uruchomić poniższy kod. **Uwaga!!** Prawdopodobnie do poprawnego działania, biblioteki `openCV` oraz `PyTorch` mogą wymagać systemu w architekturze `64 bit`.

```
from importlib.resources import path
from time import time
import torch
from matplotlib import pyplot as plt

import numpy as np
import cv2

model = torch.hub.load('ultralytics/yolov5', 'custom', path='best100.pt',
    force_reload=True)

cap = cv2.VideoCapture(0)
while cap.isOpened():
    start = time()
    ret , frame = cap.read()
    result = model(frame)
    cv2.imshow('Screen', np.squeeze(result.render()))
    if cv2.waitKey(10) & 0xff == ord('x'):
        break
    if cv2.getWindowProperty("Screen", cv2.WND_PROP_VISIBLE) < 1:
        break
    end = time()
    fps = 1/(end - start)
    print(fps)
cap.release()
cv2.destroyAllWindows()
```