Wrocław, 24 listopada 2024

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 nr 5. Podczas tego etapu nacisk zostanie położony na dodatkowy trening modelu detekcji w celu poprawy jego skuteczności w wykrywaniu i rozpoznawaniu wyuczonych gestów oraz następnie wdrożenie jego na moduł bazujący na `Raspberry Pi`

2 Zadania

Przypomnienie kompletnej listy zadań w ramach obu laboratoriów:

1. Zebranie zdjęć do treningu.
2. Oznaczenie zdjęć.
3. Trening modelu.
4. Wdrożenie systemu do detekcji gestów na `Raspberry Pi`.

*. Wdrożenie systemu w urządzeniu mobilnym.

3 Opis zadań

3.1 Zadanie 4

Celem zadania jest wdrożenie wcześniej wytrenowanego modelu w module. Przed przystąpieniem do zadania będzie niezbędna instalacja odpowiednich bibliotek, w tym do obsługi kamery np.: `OpenCV` lub `PiCamera`.

Kamera

Na początku 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.

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()
```

3.2 Zadanie *.

Cel zadania jest analogiczny jak w zadaniu poprzednim, ale model należy wdrożyć w urządzeniu mobilnym. Detekcja i rozpoznanie musi działać w czasie rzeczywistym. Dozwolone jest używanie dowolnych bibliotek, oczywiście w zależności od wybranej platformy (`Android` lub `iOS`).

Zadanie opracowanie wspólnie z *dr inż. Teodorem Niżyńskim.*