Dokumentacja aplikacji “BeHeard”

Programowanie Zespołowe 2021/22
UMK WMiI

Członkowie zespołu

dr. Marta Burzańska - Opiekun Zespołu
Maksymilian Rękawek - Kierownik Zespołu
Hubert Nowakowski - programista
Michał Gulczyński - programista
Szymon Smolik - programista
Karol Wróblewski - programista
Piotr Kulig - sekretarz

Czym jest BeHeard?

Pomysł na projekt pojawił się w głowach studentów UMK pod koniec 2021 roku. Studenci kierunku informatyki pod okiem dr Marty Burzańskiej realizowali program w 6-cio osobowym zespole.

Za cel tworzonej aplikacji postawiono ułatwienie kontaktu między osobami głuchoniemymi a osobami nie znającymi języka migowego. W szczególności kładąc nacisk na czynności związane z bezpośrednią obsługą klienta.

Dla kogo przeznaczona jest aplikacja?

Naszym profilem klienta jest każde biuro bezpośredniej obsługi klienta, które w obsłudze posiada użytkowników głuchoniemych.
Aplikacja służy wsparciu pracownikom biura w sytuacjach obsługi osoby z problemem mowy posługującej się językiem migowym.

Wykorzystane technologie

Do wykonania aplikacji zostały wykorzystane następujące technologie:

1. OpenCV
2. Tensorflow
3. Keras
4. SK_Learn
5. MediaPipe
6. Numpy

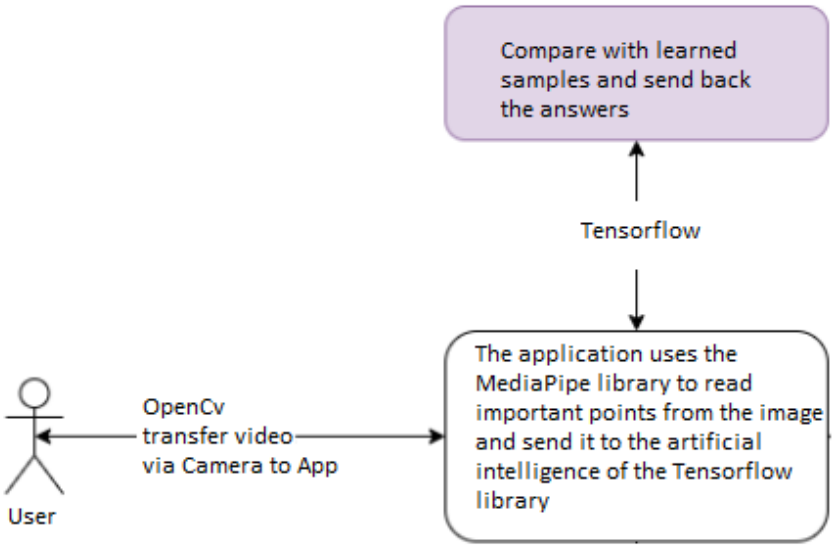
Wymagania do uruchomienia aplikacji:

- Środowisko: Windows 7+, Linux, MacOs
- Kamera minimum 420p
- CPU minimalny: AMD Ryzen 7 4800H
- Ram: 2400 MHz 8GB
- GPU minimalna: zintegrowana
- HDD: 5GB
- Nie wymagane podłączenie do sieci

Licencja

Demoware - wersja o ograniczonej funkcjonalności w stosunku do wersji pełnej. Do uzyskania pełnej wersji prosimy o kontakt przez formularz u dołu strony.

Schemat architektury



Opis głównych metod aplikacji

Funkcja `landmarkSnapshot` zapisująca pozycje wszystkich punktów w aktualnej klatce i zwracajaca tablice xyz

Funkcja `makeFolders` tworząca podkatalogi dla gestów do ćwiczenia.

Funkcja `labelAndTrain` tworzy mapę etykiet, a następnie ćwiczymy model.

Funkcja `restoreModel` przywraca model z pliku i go zwraca

Funkcja `predictFromModel` do przewidywania gestów

Opis zmiennych globalnych aplikacji

no_sequences = Ilość sekwencji dla jednego gestu

sequence_length = Długość sekwencji w klatkach obrazu

snaptime = Ilość czasu w ms pomiędzy snapshotami próbek

winX, winY = Wymiary okna z wyświetlanym obrazem

mdc = Prog wykrywania punktow ciala (niska wartość = wiele nadinterpretacji)

mtc = Dokladnosc wykrywania punktow ciala (mniej = lepsza wydajnosc kosztem jakosci)

teS = Procent próbki przeznaczony na testowanie

epo = Ilość prób dokonanych przez model (więcej = wieksza skutecznosc przy dłuższym czasie ćwiczenia)

counterFrames = Po ilu klatkach następuje zmiana wyświetlanego tekstu

actions = np.array(['xyz', 'test'**])** - Lista gestów/znaków, dopisujemy po przecinku w tablicy

Aby nagrać sekwencje gestów uruchamiamy metody:

makeFolders() - Tworzy katalogi
readSequences() - Zapisuje w nich sekwencje w postaci tablic koordynatów

Aby wyćwiczyć model uruchamiamy metody:

trainedModel = labelAndTrain() - Ćwiczy model i przypisuje go do zmiennej trainedModel
predictFromModel() - Przewiduje gesty na podstawie modelu ze zmiennej trainedModel

Aby wgrać i uruchomić wyćwiczony wcześniej model uruchamiamy metody:

trainedModel = restoreModel() - Wcześniej wyćwiczony model przypisuje do zmiennej trainedModel
predictFromModel() - Przewiduje gesty na podstawie modelu ze zmiennej trainedModel