AKADEMIA GÓRNICZO-HUTNICZA
im. St. Staszica w Krakowie
WEAIiIB, Katedra Automatyki i Robotyki

Przedmiot: Interfejsy człowiek-komputer HCI2024_4_OPENFACE

Temat projektu: Sterowanie grą poprzez ruchy głowy

Spis treści:

Wykonali: Kamila Kopacz, Stefan Kowalczyk, Mateusz Smolarczyk
IV rok *AiR Inteligentne Systemy Sterowania*
konsultant: *Jaromir Przybyło*

Kraków, kwiecień 2024.

1. Abstrakt

Celem projektu była realzacja sterowania, przy pomocy ruchów głowy w grze zręcznościowej typu „Subway Surfers”, gdzie celem gracza jest omijanie przeszkód i uzyskanie jak najwyższego wyniku.

Program napisany został w języku Python, a komunikacja z grą zaimplementowana została przy pomocy protokołu WebSocket. Sterowanie opiera się na bilbiotece MediaPipe, służącej do wykrywania ruchów twarzy, oraz odpowiednim przetwarzaniu danych na dyskretne komendy ruchów w 4 kierunkach kardynalnych.

Realizacja sterowania powiodła się, lecz jakość sterowania była niższa od oczekiwanej ze względu na ograniczony zakres wykrywanych przez program ruchów.

2. Wstęp i badania literaturowe

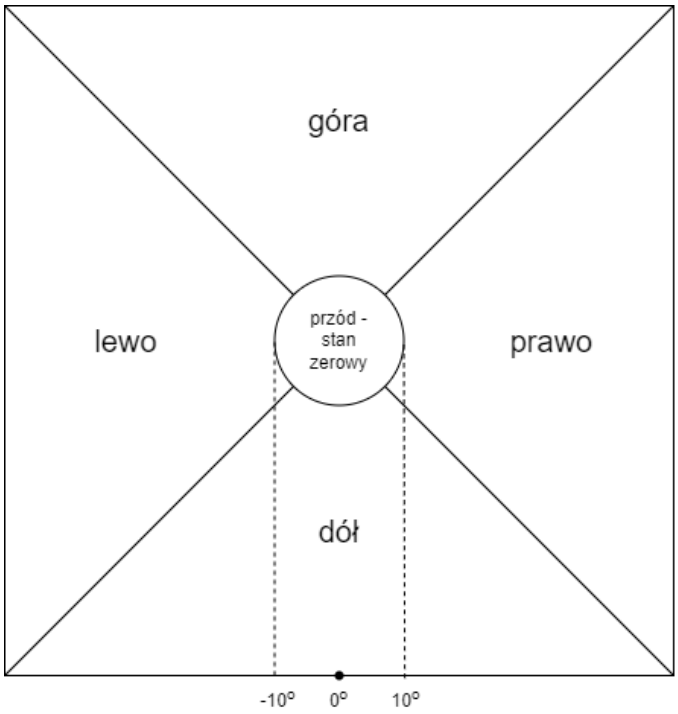
a) Celem zrealizowanego projektu było wykorzystanie istniejących bibliotek do implementacji sterowania grą poprzez ruchy głowy, mimikę twarzy bądź kierunek patrzenia. Grupa odpowiedzialna za samą implementację gry podała sposób komunikacji oraz wymaganą ilość sygnałów do sterowania równą 4, zatem została podjęta realizacja sterowania ruchami głowy w 4 kierunkach: góra, dół, prawo i lewo. Gra, do jakiej sterowanie należało zaimplementować, opierała się na popularnej grze mobilnej Subway Surfers[1]. Jest to gra typu „niekończący się biegacz”, gdzie gracz ma za zadanie omijać nadciągające przeszkody poprzez poruszanie się po 3 liniach skacząc, schylając się lub zmieniając tor lewo-prawo.

b) Zespół podjął się realizacji zadania poprzez implementację sterowania ruchami głowy przy użyciu biblioteki Pythonowej[6] Mediapipe[2]. Początkową propozycją było użycie biblioteki Openface[3], lecz została ona odrzucona przez zespół na rzecz lepszej znajomości biblioteki Mediapipe.

3. Koncepcja proponowanego rozwiązania

Wybrana modalność związana jest z obrotem głowy. Potrzebne są 4 sygnały sterujące. Są one realizowane obrotem głowy: w lewo, prawo, w górę i w dół. Patrzenie prosto przed siebie nie daje żadnego sterowania. Sygnał sterujący jest impulsowy – trwa jedną iterację pętli. Ma to na celu uniknięcie sytuacji, że gdy głowa byłaby zwrócona przez dłuższy czas w jedną stronę, to wyemitowany byłby długi sygnał ciągły. Sygnał sterujący zostanie wysłany, po obrocie głowy w

danym kierunku. Aby wysłać kolejny sygnał, kierunek głowy musi się zresetować – zmienić się na inny lub wrócić do stanu zerowego.



Rys. 3.1 Wizualizacja sygnału w zależności od kąta obrotu głowy. Oś x – kąt w stopniach w poziomie, oś y – w pionie.

3.1 Ogólny schemat algorytmu

Proponowany algorytm składa się z następujących kroków:

Cześć pierwsza – ekstrakcja cech, uzyskanie kątów obrotu twarzy.

a) Pobranie ramki obrazu z kamery
b) Wykrycie twarzy i zwrócenie punktów charakterystycznych - wykorzystując bibliotekę Mediapipe
c) Wybranie kilku punktów odpowiadających za nos, oczy, uszy i usta
d) Przeskalowanie punktów do skali obrazu
e) Estymacja położenia i obrotu twarzy na podstawie podanych punktów. Dzięki temu uzyskuje się wektor obrotu.
f) Wyliczenie, na podstawie wektora obrotu, kątów obrotu twarzy w trzech osiach

Część druga – filtracja i wyznaczenie sterowania na podstawie obrotu głowy.

g) Wybór kąta o wyższej bezwzględnej wartości, z pośród osi X, Y.
h) Progowanie – sprawdzenie, czy wartość jest spoza zakresu [-10°, 10°]
i) Filtracja – zwracany sygnał sterujący powinien być impulsowy, a nie ciągły przez cały czas, gdy twarz jest obrócona. Dlatego algorytm zwróci sygnał, jeżeli przez 3 kolejne ramki twarz będzie zwrócona w tym samym kierunku. Licznik zresetuje się, jeżeli głowa zmieni

kierunek, dzięki czemu pojedynczy obrót, nieważne jak długo trwający, zawsze zwróci jeden impuls.

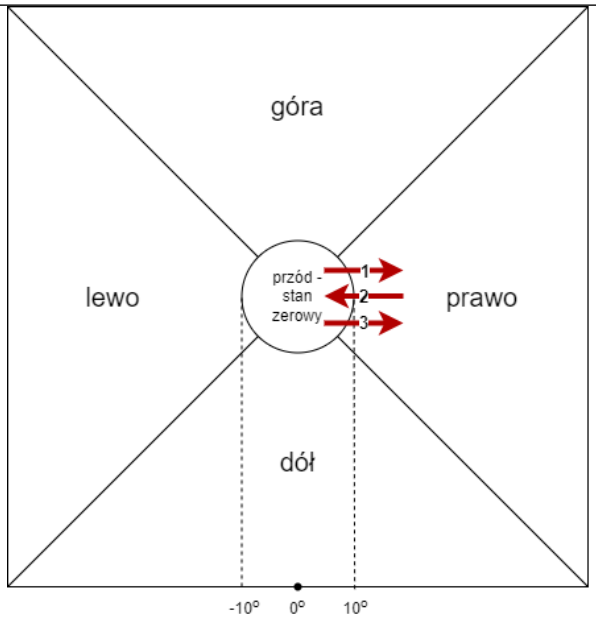
Cześć trzecia – komunikacja.

Każdorazowe wygenerowanie impulsu powoduje nadanie wiadomości po łączu WebSocket do serwera gry. W zależności od wykrytego kierunku wysyłane są odpowiednie wiadomości: "UP", "DOWN", "LEFT", "RIGHT"; zgodne z protokołem stworzonym przez twórców gry.

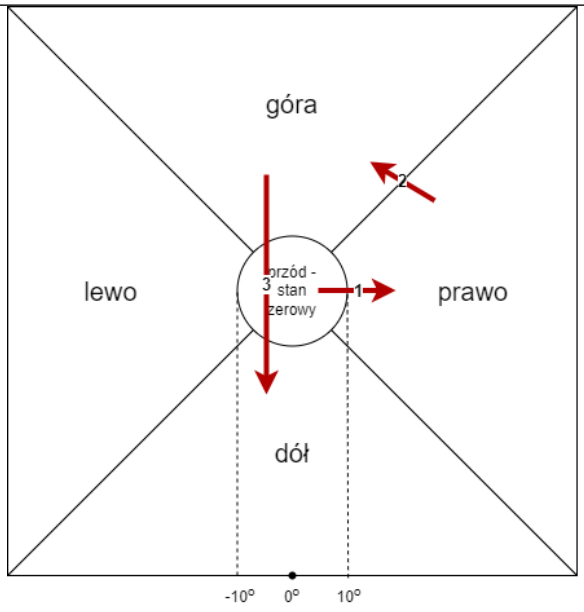
4. Rezultaty i wyniki testów

W celu przetestowania algorytmu wykorzystano kamerę internetową wbudowaną w laptopa. Wyświetlano pobierane ramki oraz wypisywano w konsoli sygnał sterujący. Na podstawie przeprowadzonych testów potwierdzono działanie algorytmu.

Jak wspomniano wcześniej sygnał sterujący jest impulsowy, a nie ciągły. Wymagane jest zresetowanie licznika. Na Rys. 4.1 oraz 4.2 pokazano rezultat dla przykładowych ruchów głowy.



Rys. 4.1 Schemat ruchów głowy. Prawo, środek, prawo.



Rys. 4.2 Schemat ruchów głowy. Prawo, góra, dół.

W przypadku pokazanym na Rys. 4.1 sygnał wyjściowy będzie: **prawo, prawo**. W przypadku drugim sterowanie przyjmie wartości: **prawo, góra, dół.** Jak można zauważyć, aby dwa razy zadać ten sam sygnał należy dwa razy obrócić głowę w zadaną stronę. Jednak jeżeli użytkownik chce zadać kolejno inne sygnały, nie ma przymusu za każdym razem resetować licznika patrząc na wprost.

Zakres kątów [-10, 10] wybrano doświadczalnie. W tym wypadku należało znaleźć kompromis, gdyż zbyt duży kąt sprawiałby, że algorytm mógłby nie wykrywać ruchu (FN), a zbyt mały skutkowałby fałszywymi detekcjami (FP).

Zauważono, że kąt obrotu twarzy w osi pionowej jest lekko przesunięty względem intuicji. Wynika to z położenia kamery. Nie jest to błąd działania programu, więc postanowiono tego nie korygować.

5. Realizacja proponowanego rozwiązania

5.1 Język programowania i biblioteki

Projekt został zrealizowany z wykorzystaniem języka Python W celu realizacji zadania wykrywania ruchów głowy wykorzystano bibliotekę MediaPipe. Biblioteka MediaPipe jest open-source-owym zestawem narzędzi i bibliotek stworzonym przez Google, zaprojektowanym do analizy multimodalnych danych wejściowych, takich jak obrazy, wideo i dane dźwiękowe. Jest to narzędzie, które ma na celu ułatwienie budowy aplikacji związanych z przetwarzaniem multimediów, w szczególności z wykorzystaniem technik uczenia maszynowego. MediaPipe zawiera moduły do detekcji i śledzenia obiektów w czasie rzeczywistym na obrazach i wideo. Można go wykorzystać do wykrywania twarzy, rąk, ciała, gestów, a nawet konkretnych obiektów. Wybrano właśnie tę bibliotekę, ze względu na to, że jest idealnym narzędziem do rozwiązania podanego problemu. Do tego jest dostępna w języku Python, co ułatwia znacząco jej wykorzystanie.

Do wykonania zadania potrzebne były również biblioteki numpy[5], OpenCV[4] oraz WebSocket.

5.2 Szczegółowy opis algorytmu

Pierwszym krokiem programu jest połączenie się z kamerą. W tym celu wykorzystano funkcję VideoCapture z OpenCV. Pobiera ona ramki obrazu z kamery internetowej laptopa. Stworzono również obiekt klasy *FaceMesh* wykorzystując Mediapipe. To właśnie dzięki niemu będzie możliwa detekcja twarzy.

Główny program znajduje się w nieskończonej pętli, przerywanej interwencją użytkownika. Na początku każdej pętli pobierana jest ramka z kamery. Następnie obraz jest obracany wokół osi y oraz zmieniana jest przestrzeń barw z BGR na RGB. W kolejnym kroku wykorzystywany jest obiekt klasy *FaceMesh*, który po uruchomieniu metody *process* zwraca punkty charakterystyczne wykrytej twarzy. Spośród wszystkich punktów zostaje wybrane kilka charakterystycznych odpowiadających za części ciała (usta, nos, oczy, uszy). Tworzony jest z nich wektor 2D oraz 3D.

Następnie wyznaczona zostaje dystorcja kamery. Potrzebna jest ona do funkcji *solvePnP* z biblioteki OpenCV. Funkcja ta zwraca, między innymi, wektor obrotu, na podstawie którego, dzięki wykorzystaniu funkcji *Rodrigues* i *RQDecomp3x3* (OpenCV), można uzyskać wartości kątów obrotu twarzy w trzech osiach. Zwrócone wartości są unormowane, więc trzeba je przemnożyć razy 360. Następnie następuje progowanie i generowanie sygnału sterującego. Proces ten został wyczerpująco opisany w rozdziale 3.

6. Podsumowanie i wnioski

Zespołowi udało się zaimplementować zaproponowane sterowanie ruchami głowy. Ostateczny test oprogramowania został przeprowadzony na zajęciach stacjonarnych, gdzie udało się sukcesywnie połączyć z serwerem wystawionym przez grupę odpowiedzialną za implementację gry i możliwe było sterowanie poruszaniem się w grze, a podjęta próba jest widoczna na nagraniu wspomnianym w rozdziale Dodatek B. Poza zaimplementowanymi czterema sygnałami, możliwe by było rozszerzenie ich ilości poprzez np. wykrywanie ilości ruchów głowy w danej jednostce czasu lub szybkości obrotu.

7. Literatura

[1] Gra Subway Surfers (https://subwaysurf.io/home)

[2] Biblioteka Mediapipe 0.10.11 (https://pypi.org/project/mediapipe/)

[3] Biblioteka OpenFace (https://github.com/TadasBaltrusaitis/OpenFace)

[4] Biblioteka Opencv-Python 4.9.0.80 (https://pypi.org/project/opencv-python/)

[5] Biblioteka Numpy 1.24.3 (https://numpy.org/)

[6] Python 3.10.13 (https://www.python.org/)

8. DODATEK A: Opis opracowanych narzędzi i metody postępowania

Celem uruchomienia oprogramowania, należy sklonować repozytorium z podanego w rozdziale Dodatek B linku, następnie w sklonowanym folderze zainstalować wymagane bibilioteki wymienione w pliku requirements.txt. Następnie należy uruchomić skrypt języka Python face_direction.py. Otworzy się okno kamery, którą wcześniej należy podłączyć do urządzenia, jeśli nie posiada ono wbudowanej, na podstawie którego program będzie wysyłał dane do serwera o wykrytym kierunku ruchu głowy.

9. DODATEK B. Spis zawartości załączników

- https://github.com/hyunmila/hcl_project.git – repozytorium z kodem na GitHubie
- https://drive.google.com/file/d/1DUU3jmcE2jXQNjbsa4XVlCCqud7pD0aK/view – nagranie z przykładu sterowania przy użyciu zaimplementowanego algorytmu