AKADEMIA GÓRNICZO-HUTNICZA

im. St. Staszica w Krakowie

WEAIiE, Katedra Automatyki i Robotyki

Przedmiot: Interfejsy człowiek-komputer HCI2025_4_OPENPOSE

Temat projektu: Sterowanie aplikacją za pomocą dłoni.

Spis treści:

Wykonali: Mateusz Cierpik, Tymoteusz Domagała, Bartłomiej Bańka

4 rok *AiR*.

konsultant: *dr inż. Jaromir Przybyło*

Wersja 1.0

Kraków, maj 2025.

1. Abstrakt

Celem projektu była implementacja sterowania do gry na bazie ruchów i gestów dłoni. Konkretnie zadanie związane było z detekcją i klasyfikacją aktualnego wskazania gracza oraz następnie dostarczeniem sygnałów sterujących do gry. Wykorzystano w tym celu bibliotekę MediaPipe, która zapewniła detekcję punktów charakterystycznych dłoni. Na podstawie wykrytych punktów przeprowadzana była analiza kierunku palca wskazującego, uwzględniająca także kierunki pośrednie jak np. wskazanie “góra-lewo”. Ponadto na tej samej klatce obrazu przeprowadzano jeszcze klasyfikację gestu. Istotny aspekt projektu stanowiła także obsługa komunikacji modułu z głównym kodem gry, uruchamianym w innym miejscu. Ustalono połączenie sieciowe oparte na protokole UDP oraz skonfigurowano odpowiednio ramkę wysyłanych danych. Stworzony moduł przetestowano w różnych warunkach i wprowadzono drobne modyfikacje, wynikające z zaobserwowanych w międzyczasie problemów. Ostatecznie po wprowadzonych zmianach i testach całościowych z głównym modułem gry, stwierdzono dobrą skuteczność i poprawność działania sterowania.

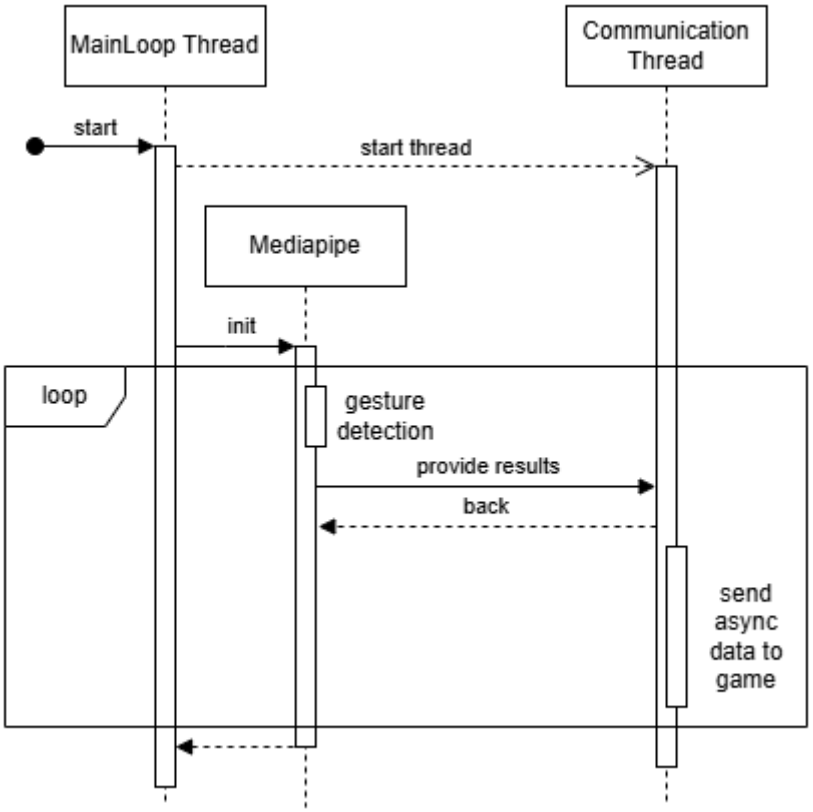
2. Wstęp i koncepcja proponowanego rozwiązania

W projekcie zdecydowaliśmy się na wykorzystanie biblioteki MediaPipe, która stanowi jedno z najlepszych dostępnych rozwiązań w swojej kategorii. Jest to biblioteka typu open-source, udostępniona na licencji Apache 2.0, co umożliwia jej bezpłatne wykorzystanie — również w projektach komercyjnych — oraz modyfikowanie zgodnie z indywidualnymi potrzebami. MediaPipe zapewnia wysoką skuteczność działania, dużą niezawodność oraz znacząco przyspiesza proces wdrożenia w porównaniu do tworzenia własnej implementacji od podstaw.

Do realizacji komunikacji z modułem grupy GAME zdecydowaliśmy się zastosować połączenie sieciowe oparte na protokole UDP (ang. User Datagram Protocol). Jego główną zaletą jest wysoka szybkość transmisji danych oraz brak konieczności ustanawiania i utrzymywania stałego połączenia z serwerem, co pozwala na przesyłanie informacji z dużą częstotliwością — kluczowe w przypadku dynamicznych aplikacji, takich jak gry. Niemniej jednak, UDP ma również swoje ograniczenia: nie gwarantuje dostarczenia wszystkich pakietów, ich kolejności ani integralności danych. W kontekście gier komputerowych, gdzie dane są wysyłane wielokrotnie w ciągu sekundy, niedoskonałości te są zazwyczaj akceptowalne i nie wpływają znacząco na jakość działania aplikacji.

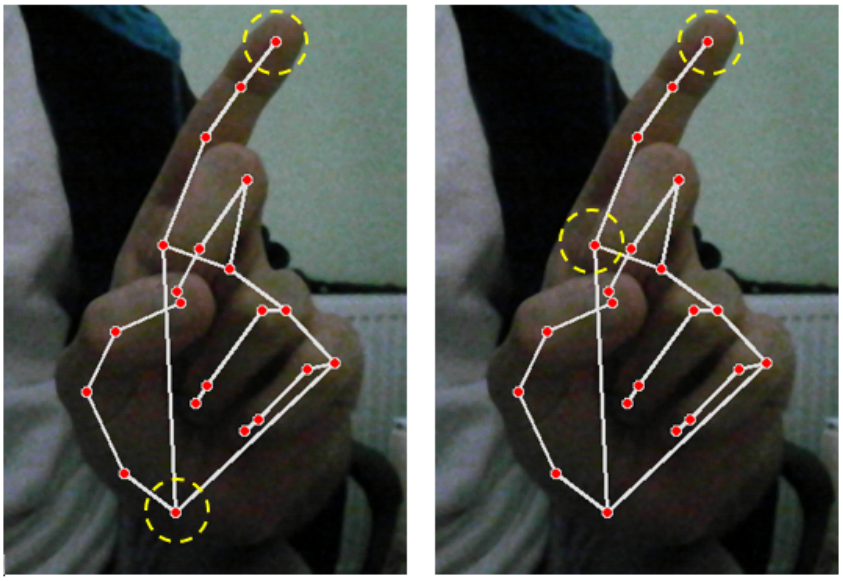
W języku Python dostępna jest bogata biblioteka do obsługi komunikacji sieciowej, w tym z użyciem protokołu UDP. Dodatkowo, MediaPipe oferuje wsparcie dla Pythona poprzez udostępnione API, co umożliwia jego pełną integrację w naszym środowisku programistycznym. Python to język cieszący się ogromną popularnością, znany ze swojej czytelności, prostoty i szybkości tworzenia rozwiązań. Choć nie należy do najszybszych pod względem wydajności, jego elastyczność oraz dostępność gotowych narzędzi pozwala na szybkie budowanie złożonych aplikacji. Dodatkowym atutem jest wysoki poziom znajomości tego języka w naszym zespole, co przekłada się na efektywność pracy i jakość kodu.

Poniżej został zaprezentowany wstępny schemat działania naszego systemu:



3. Rezultaty i wyniki testów

Na etapie testowania algorytmu określającego kierunek wskazania palcem przeprowadzono porównanie dwóch podejść. W pierwszym z nich wykorzystane punkty charakterystyczne to czubek palca wskazującego oraz punkt nadgarstka. W drugim podejściu wybrano skrajne punkty palca wskazującego.



Na podstawie przeprowadzonych testów stwierdzono, że użycie skrajnych punktów samego palca wskazującego (czubka i podstawy) daje wyraźnie lepsze rezultaty w kontekście dokładności wykrywania kierunku (lepiej oddaje rzeczywisty kierunek, ponieważ uwzględnia lokalne ułożenie palca, niezależnie od ogólnego ułożenia dłoni w przestrzeni).

4. Realizacja proponowanego rozwiązania

Opis działania algorytmu (część „sterowanie na podstawie wskazania palcem”):

Działanie tej części programu opiera się na określeniu orientacji czubka palca wskazującego w stosunku do jego podstawy oraz wyznaczenia na tej podstawie kierunku sterowania. Zostało to zrealizowane w oparciu o punkty charakterystyczne wykrywane przez model MediaPipe Hands. Szczegółowo wydzielone kolejne etapy działania programu to:

- Pobieranie obrazu z kamery
odczyt klatka po klatce (biblioteka OpenCV)

- Wykrywanie dłoni i punktów charakterystycznych
wykorzystanie biblioteki MediaPipe do detekcji dłoni oraz określenia pozycji „landmarków” – w tym przypadku czubka palca wskazującego (INDEX_FINGER_TIP), jego podstawy (INDEX_FINGER_MCP) oraz nadgarstka (WRIST).

- Obliczenie kąta kierunku

Na podstawie różnicy wektorowej między pozycją czubka palca wskazującego i jego podstawą obliczany jest kąt:

$$\theta = \left(arctan\frac{dy}{dx} \cdot \frac{180}{\pi} + 360\right) mod\ 360$$

gdzie dx i dy to przesunięcia w osi X i Y między skrajnymi punktami palca.

- Określenie sterowania

Wyliczony kąt zamieniany jest na odpowiednie sterowanie w formacie „wsad". Jest to określane na podstawie predefiniowanych zakresów kątów przypisanych do kierunków:

Góra (W): 75°–105°
Dół (S): 255°–285°
Lewo (A): 165°–195°
Prawo (D): 345°–360° i 0°–15°

Pozostałe – Kierunki pośrednie (np. Góra-Prawo)

Wyznaczone sterowanie kodowane jest do ustalonego formatu ["w", "s", "a", "d"], gdzie aktywne sterowanie sygnalizowane jest wartością 1. Przykładowo sterowanie „Góra-Prawo" przesyłane będzie zgodnie z tym formatem jako "1001".
Dodatkowo logika działania tej części została rozszerzona o przypadek braku sterowania (gracz nie wskazuje żadnego kierunku). Będzie to zatem sytuacja gdy palec wskazujący jest schowany, a przechodząc do logiki programu wtedy gdy odległość między czubkiem palca a jego podstawą jest mniejsza niż określony próg (0.12 jednostek znormalizowanych).

- Wizualizacja wyników

Wykryty kierunek jest wyświetlany na obrazie w postaci tekstowej.

Opis wysyłania danych (networking):

Zadaniem tej części systemu jest przesyłanie danych do modułu GAME z odpowiednio wysoką częstotliwością, zapewniającą płynność działania aplikacji. W trakcie realizacji projektu

zmodyfikowano pierwotne podejście, opisane wcześniej w części dotyczącej koncepcji rozwiązania.

Początkowo planowano wykorzystanie osobnego wątku (ang. *thread*) do realizacji wysyłania danych, jednak na podstawie przeprowadzonych testów i analiz zdecydowano się uprościć ten mechanizm. Ostatecznie dane są przesyłane bezpośrednio po każdej przetworzonej klatce obrazu. Takie rozwiązanie okazało się wystarczające pod względem wydajnościowym, a jednocześnie znacząco uprościło architekturę programu, eliminując potrzebę zarządzania współbieżnością i potencjalnymi problemami synchronizacji wątków.

Przed wysłaniem danych do modułu GAME konieczne jest ich odpowiednie przekształcenie — z formatu wewnętrznego (np. *Góra*, *Dół*, *Góra-Prawo* itd.) na wymagany przez grupę GAME format, czyli ciąg pięciu znaków ASCII reprezentujących wartości sterowania. Pierwszy znak reprezentuje grupę od której wysyłane jest sterowanie (w naszym przypadku wartość 1), pozostałe 4 znaki oznaczają zakodowane dane.

Poniżej przedstawiono przykład konwersji dla przypadku, w którym wykryto sterowanie w dół:

1_ _ _ _ +

Format	
wewnętrzny	grupa GAME
None	0000
Gora	1000
Dol	0100
Lewo	0010
Prawo	0001
Gora-Prawo	1001
Gora-Lewo	1010
Dol-Lewo	0110
Dol-Prawo	0101

= 10100

5. Podsumowanie i wnioski

Podczas realizacji projektu okazało się, że analiza oraz praktyczne testowanie systemu w rzeczywistych warunkach użytkowania jest kluczowe, zarówno od strony dokładności działania samego modułu jak i jego synchronizacji z całą grą. Istotnym aspektem była zatem ścisła współpraca z zespołem GAME. Synchronizacja pomiędzy systemem sterowania a logiką gry wymagała wspólnego ustalania formatów danych, oczekiwanych efektów działania itp. Od strony

implementacji, zastosowanie w projekcie gotowych bibliotek (w szczególności MediaPipe) znacząco przyspieszyło prace nad projektem. Możliwość skorzystania z dobrze działających modeli detekcji punktów charakterystycznych dłoni pozwoliło skoncentrować się na implementacji logiki sterowania, a nie na budowie systemu od podstaw.

6. Literatura

[1] "MediaPipe Legacy Solutions", https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

[2] Indriani Moh.Harris, Ali Suryaperdana Agoes, „Applying Hand Gesture Recognition for User Guide Application Using MediaPipe", 2021

[3] Shu Nakamura, Yasutomo Kawanishi, Shohei Nobuhara, Ko Nishino, „DeePoint: Visual Pointing Recognition and Direction Estimation."

[4] https://virtualenvwrapper.readthedocs.io/en/latest/install.html - Instalacja rozrzeszenia virtualenvwrapper

7. Dodatek A: Instalacja i uruchomienie programu

Projekt jest kompatybilny tylko z systemami typu linux (np. Linux Mint, Ubuntu, Arch Linux). Jest to spowodowane tym że jedna z zależności TensorFlow od wersji 2.10, utraciła wsparcie dla systemów Windows. Również aby uniknąć błędów zalecane jest korzystanie z wirtualnego środowiska języka Python, aby odizolować projekt od zainstalowanych wcześnie bibliotek systemowych i uniknąć błędów. Jednym z takich narzędzi jest virutalenvwrapper [4], którego kroki instalacji znajdują się w rozdziale 6.

- Wymagany system operacyjny: Linux
- Wymagana wersja języka Python: 3.10.X
- Wymagane zależności
  - opencv_python
  - mediapipe
  - pyautogui
  - requests

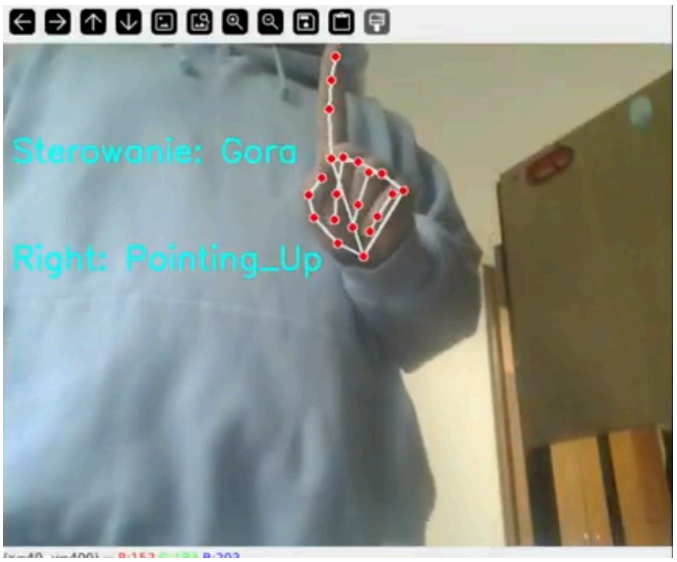
Za uruchomienie programu odpowiada plik main_movement_control.py, a w pliku util/config.py możemy ustawić adres i port do którego będziemy wysyłać dane poprzez protokół UDP. Poniżej przedstawiamy kroki instalacji korzystając z virutalenvwrapper i Visual Studio Code.

1. Tworzymy odizolowane środowisko za pomocą komendy:

   mkvirtualenv ICK_OP -p 3.10

   (W systemie operacyjnym musi zostać zainstalowana wersja Pythona 3.10 jak i virtualenvwrapper)
2. Instalujemy wymagane zależności:

   pip3 install opencv_python mediapipe pyautogui requests
3. Aby wyjść ze środowiska korzystamy z komendy *deactivate* (Aby pracować w danym środowisku: *workon NAZWA*; Aby usunąć dany venv: rmvirtualenv NAZWA)
4. Uruchamiamy program Visual Studio Code i otwieramy folder zawierający projekt
5. Za pomocą skrótu CTRL+SHIFT+P, wpisujemy Python: Select Interpreter i z listy wybieramy utworzone wcześniej środowisko (Dla virtualenvwrapper domyślnym folderem jest ~/.virutalenvs, czyli dla naszego przykładu to: ~/.virtualenvs/ICK_OP/bin/python)
6. Po uruchomieniu programu wyskoczy okno na którym będzie widać obraz z kamery z nałożonymi punktami orientacyjnymi na dłoni. Domyślnie jesteśmy w stanie wykryć czy dłoń jest lewa czy prawa, oraz jaki gest wykonuje dłoń. Poruszając palcem wskazującym prawej dłoni jesteśmy w stanie wysłać sygnał sterujący w zależności jaki kąt tworzy on z osią OX.



Rysunek 1 Główne okno programu main_movement_control.py