LungeGuard- System wspomagania treningu wykroków z analizą biomechaniczną w czasie rzeczywistym

Autorzy:

- Bartłomiej Raj (251210) 251210@edu.p.lodz.pl
- Marcel Podlecki (251204) 251204@edu.p.lodz.pl
- Bartłomiej Jedyk (252795) 252795@edu.p.lodz.pl
- Wojciech Stochmiałek (251226) 251226@edu.p.lodz.pl

Repozytorium:

- https://github.com/BaDaQu/LungeGuard-AI-Trainer

Spis treści

1. Wstęp i Cel Projektu

Projekt dotyczy inteligentnego systemu wspomagania treningu siłowego z wykorzystaniem wizji komputerowej (Computer Vision) oraz algorytmów sztucznej inteligencji. Głównym celem systemu jest analiza techniki wykonywania wykroków (lunges) w czasie rzeczywistym, co ma kluczowe znaczenie w profilaktyce kontuzji stawów kolanowych i kręgosłupa. System wykorzystuje architekturę dwukamerową (Dual-View) do precyzyjnego śledzenia biomechaniki ruchu, eliminuje problem przysłaniania kończyn (okluzji) oraz oferuje interakcję głosową, pełniąc rolę wirtualnego trenera personalnego.

Przegląd istniejących rozwiązań

- Urządzenia typu Wearables (Smartwatche, Opaski) – Bazują głównie na akcelerometrach i żyroskopach. Skutecznie mierzą tętno, czas treningu czy przybliżoną liczbę powtórzeń, jednak nie posiadają zdolności „widzenia" sylwetki użytkownika, przez co nie mogą korygować błędów technicznych (np. koślawienia kolan czy garbienia się).
- Aplikacje mobilne oparte na jednej kamerze – Wykorzystują kamerę smartfona do analizy obrazu 2D. Główną wadą tego rozwiązania w przypadku ćwiczeń asymetrycznych (jak wykroki) jest problem okluzji – jedna noga zasłania drugą, co uniemożliwia pełną ocenę głębokości i kątów stawowych w rzucie bocznym przy jednoczesnej analizie stabilności w rzucie frontalnym.
- Profesjonalne systemy Motion Capture (MoCap) – Stosowane w studiach filmowych i laboratoriach biomechanicznych. Oferują najwyższą precyzję dzięki systemowi markerów i wielu kamerom IR, jednak ze względu na koszt i skomplikowaną obsługę są niedostępne dla użytkownika domowego.
- Inteligentne lustra (Smart Mirrors) – Urządzenia typu Tonal czy Mirror oferują korektę techniki, jednak są to rozwiązania hardware'owe o bardzo wysokim koszcie zakupu, wymagające dedykowanego miejsca instalacji.
- Niniejszy projekt łączy zalety powyższych rozwiązań, eliminując ich wady poprzez zastosowanie taniego sprzętu powszechnego użytku (laptop + smartfon jako druga kamera). System wyróżnia się implementacją autorskich algorytmów geometrycznych do analizy kątowej, systemem Anti-Cheat weryfikującym poprawność powtórzenia (analiza obniżenia środka ciężkości) oraz w pełni bezdotykową obsługą głosową działającą w trybie offline.

2. Analiza Problemu i Dobór Technologii

Algorytmy estymacji pozy (Human Pose Estimation)

Współczesne systemy analizy ruchu opierają się na modelach głębokiego uczenia (Deep Learning). W literaturze dominują dwa podejścia:

- **Podejście "Top-Down" (np. OpenPose):** Najpierw wykrywa wszystkie osoby na obrazie, a następnie szacuje ich pozy. Jest bardzo dokładne, ale wymagające obliczeniowo, co utrudnia działanie w czasie rzeczywistym na sprzęcie konsumenckim.
- **Podejście "Bottom-Up" / Lightweight (np. MediaPipe BlazePose):** Wykorzystywane w tym projekcie. Skupia się na wykrywaniu punktów kluczowych (landmarks) dla pojedynczej osoby z dużą szybkością (30+ FPS na CPU). Model ten, opracowany przez Google, wykorzystuje architekturę sieci splotowych zoptymalizowaną pod kątem urządzeń mobilnych, co pozwala na inferencję 3D z pojedynczego obrazu 2D [2].

Analiza biomechaniczna i geometria obliczeniowa

Standardowe metody analizy wideo w sporcie opierają się na wyznaczaniu kątów między segmentami ciała. Wykorzystanie biblioteki **NumPy** oraz funkcji trygonometrycznych (atan2) pozwala na przekształcenie współrzędnych pikseli na wektory i obliczenie kątów stawowych (np. zgięcia kolana) z wysoką precyzją. Istotnym problemem opisywanym w literaturze [3] jest tzw. **okluzja** (zasłanianie jednej kończyny przez drugą), co w naszym projekcie rozwiązano poprzez zastosowanie architektury dwukamerowej (Dual-View) [6].

Przetwarzanie mowy w systemach czasu rzeczywistego (ASR)

Systemy sterowania głosowego często polegają na rozwiązaniach chmurowych (np. Google Speech API), co wprowadza opóźnienia (latencję) rzędu 1-2 sekund. W aplikacjach treningowych wymagana jest natychmiastowa reakcja. Rozwiązaniem zastosowanym w projekcie jest biblioteka **Vosk**, oparta na modelu Kaldi, która umożliwia rozpoznawanie mowy w trybie offline z minimalnym opóźnieniem, zapewniając prywatność i niezależność od łącza internetowego [4].

Systemy integrujące akwizycję danych

Kluczowym wyzwaniem w systemach wizyjnych jest synchronizacja strumieni danych. Wykorzystanie biblioteki **OpenCV** w połączeniu z wielowątkowością (threading) pozwala na równoległe pobieranie obrazu z kamery USB oraz kamery IP (smartfon), minimalizując zjawisko "klatkowania" i opóźnień bufora, które jest częstym problemem przy transmisji wideo przez Wi-Fi [5].

2.2. Stos Technologiczny

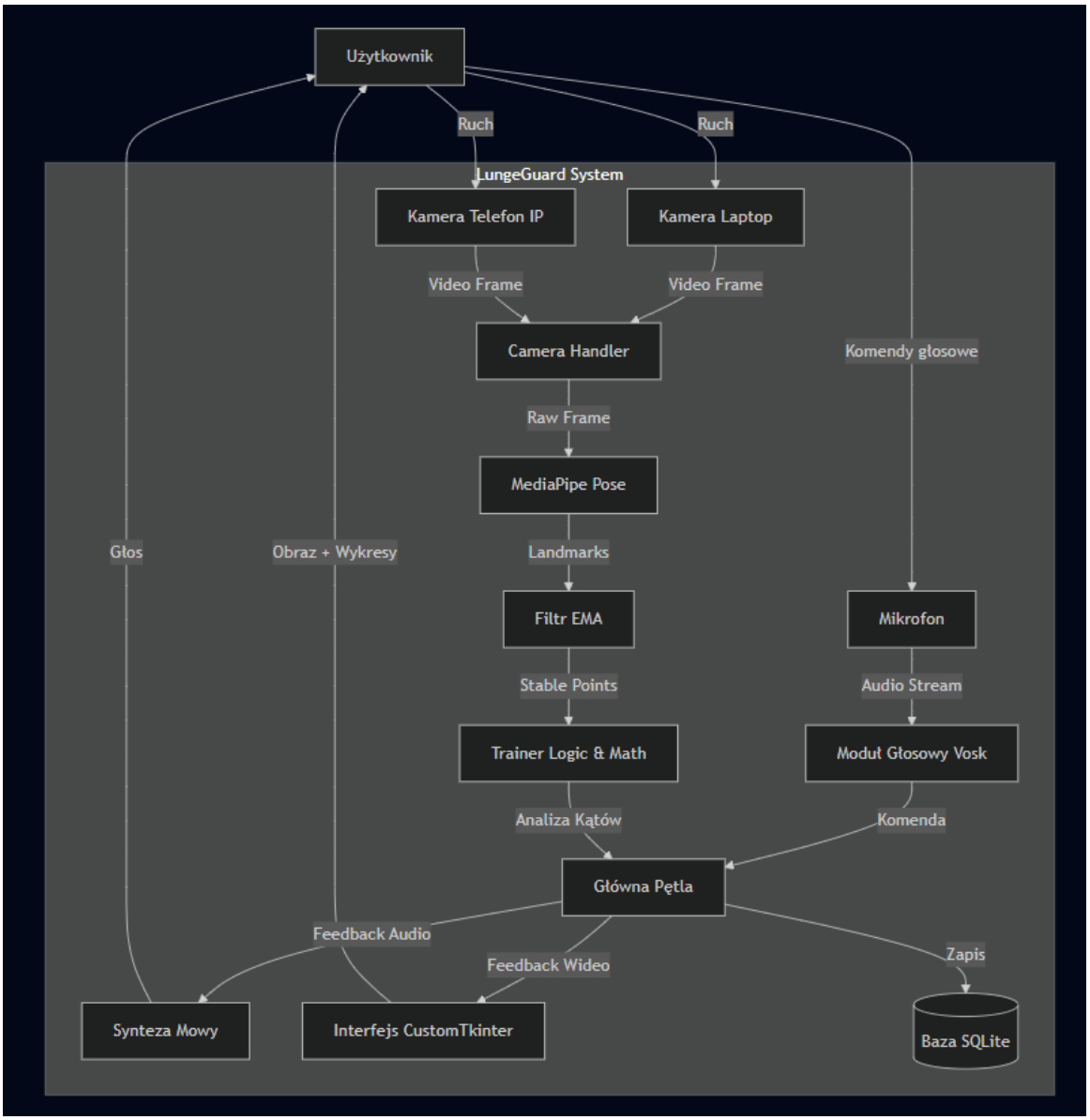
Wybór technologii podyktowany był wymaganiami wydajnościowymi (30 FPS na CPU) oraz łatwością integracji.

- Język: Python 3.10.
  - AI / Computer Vision:
  - MediaPipe Pose (Google): Wybrano zamiast YOLOv8-Pose ze względu na lepszą stabilność punktów kluczowych (landmarks) w 3D i mniejsze zapotrzebowanie na moc obliczeniową [1].
- OpenCV: Do akwizycji i przetwarzania obrazu.
- Matematyka: NumPy oraz autorskie algorytmy geometryczne do obliczania kątów wektorowych.
- Interfejs: CustomTkinter (nowoczesny wrapper na TKinter).
- Dźwięk: Vosk (Offline Speech Recognition) do sterowania oraz pyttsx3 do syntezy mowy.
- Dane: SQLite do przechowywania historii i logów błędów.

3. Architektura Systemu

System działa w architekturze wielowątkowej, aby oddzielić renderowanie interfejsu (GUI) od ciężkich obliczeń wizyjnych i operacji I/O (kamery, sieć).

Schemat architektury przedstawiono na Rys. 1



Rysunek 1. Schemat przepływu danych w systemie LungeGuard.

Kluczowe moduły:

1. CameraHandler: Klasa obsługująca pobieranie klatek w oddzielnych wątkach. Zaimplementowano agresywne czyszczenie bufora (grab/retrieve), co wyeliminowało opóźnienia (lagi) przy transmisji z kamery IP (telefonu).
2. TrainerLogic: "Mózg" systemu – maszyna stanów (State Machine) decydująca o zaliczeniu powtórzenia lub zgłoszeniu błędu.
3. SkeletonProcessor: Warstwa abstrakcji normalizująca surowe dane z AI na czytelne parametry biomechaniczne.

Setup sprzętowy:

System wykorzystuje laptopa jako jednostkę centralną oraz smartfon z aplikacją IP Webcam jako bezprzewodową kamerę boczną.

PLACEHOLDER ZDJECIE SETUPU

Rysunek 2. Fizyczna konfiguracja stanowiska treningowego.

4. Algorytmy i Metodyka (Rozwiązanie Niskopoziomowe)

4.1. Matematyczny model estymacji kątów

Podstawą analizy jest obliczanie kątów w stawach na płaszczyźnie 2D. System pobiera znormalizowane współrzędne

(x, y) punktów kluczowych (Landmarks) z modelu MediaPipe.
Kąt α w stawie (np. kolanowym) wyznaczany jest jako różnica azymutów dwóch wektorów tworzących ten staw (udo i podudzie), z wykorzystaniem funkcji atan2, co pozwala na uniknięcie osobliwości przy kątach 90° i 270°.

Zastosowano autorski algorytm:

1. Wyznaczenie wektorów:

   $\vec{v_1}$=P_{biodro}−P_{kolano}, $\vec{v_2} = P_{kostka} - P_{kolano}$.

2. Obliczenie kątów wektorów względem osi X:

   X: $\theta_1 =$ atan2(v_{1y}, v_{1x}), $\theta_2 =$ atan2($v_{2y,}v_{2x}$).

3. Kąt wynikowy:

   $\alpha = |\theta_1 - \theta_2|$.

4. Normalizacja: Jeśli

   $\alpha > 180°, to\, \alpha = 360° - \alpha$

4.2. Algorytm Anti-Cheat (Detekcja "Hip Drop")

Standardowa analiza kątów jest podatna na manipulacje (np. unoszenie nogi w miejscu). Aby temu zapobiec, zaimplementowano algorytm weryfikacji przemieszczenia środka ciężkości.
System kalibruje pozycję biodra w fazie *UP* (Y_{stand}). Faza *DOWN* jest zaliczana tylko wtedy, gdy aktualna pozycja biodra $Y_{current}$ spełni warunek:

$$Y_{current} - Y_{stand} > \delta$$

gdzie δ to próg czułości (ustalony eksperymentalnie na 5% wysokości kadru). Eliminuje to fałszywe powtórzenia wynikające z samej fleksji kolana bez wykonania przysiadu.

4.3. Detekcja Błędów

- Valgus (Koślawienie): Analiza odchylenia kolana od wektora łączącego biodro i kostkę w płaszczyźnie czołowej.
- Knee-Over-Toe: Analiza kąta nachylenia piszczeli względem pionu.
- Torso Inclination: Utworzenie wirtualnego punktu referencyjnego nad biodrem i badanie odchylenia kręgosłupa.

5. Implementacja i Optymalizacja

5.1. Stabilizacja Sygnału (Smoothing)

Surowe dane z MediaPipe charakteryzują się szumem (drganiem punktów). Zastosowano filtr wygładzający EMA (Exponential Moving Average) ze współczynnikiem α = 0.65.

$$smoothedval = \alpha * currentval + (1 - \alpha) * prevval$$

Pozwoliło to na uzyskanie stabilnych wykresów bez wprowadzania dużego opóźnienia.

5.2. Sterowanie Głosem (Offline)

Początkowo użyto Google Speech Recognition, co generowało opóźnienia rzędu 1.5s. Ostatecznie wdrożono bibliotekę Vosk z modelem offline.

Zastosowano hybrydowy model rozpoznawania:

- Wykorzystano ścisłą gramatykę (lista słów kluczowych: Start, Stop, Reset).
- Dodano filtr pewności (confidence > 0.9).
- Efekt: System ignoruje rozmowy w tle, ale reaguje na komendy w czasie < 0.3s.

5.3. Rozwiązanie problemu latencji w strumieniowaniu IP

Podczas testów wykryto narastające opóźnienie (do 2000 ms) przy transmisji wideo z telefonu przez Wi-Fi, spowodowane buforowaniem klatek przez bibliotekę OpenCV. Standardowa metoda **read()** pobierała przestarzałe klatki z kolejki.

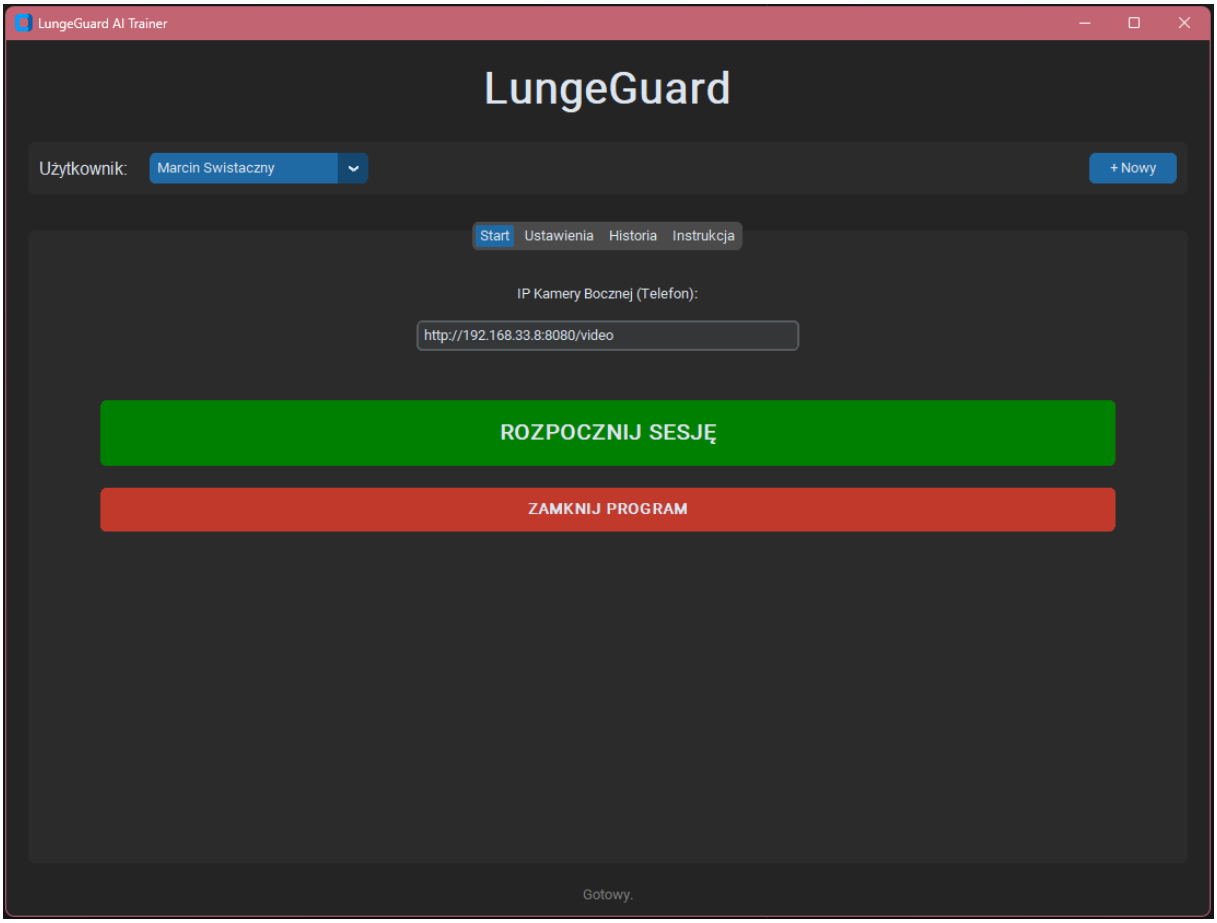
Zastosowano autorskie rozwiązanie w klasie **CameraHandler**:

1. Uruchomienie odbioru wideo w oddzielnym wątku (daemon thread).
2. Zastosowanie metody agresywnego czyszczenia bufora: Zamiast czekać na dekodowanie, pętla wątku wykonuje operację grab() w pętli, aby opróżnić bufor sprzętowy, a dekoduje (retrieve()) tylko najnowszą dostępną klatkę.

3. Wynik: Redukcja opóźnienia do poziomu < **200 ms** (real-time), co jest kluczowe dla synchronizacji dźwięku z ruchem.

6. Interfejs Użytkownika (GUI)

Aplikacja posiada nowoczesny interfejs podzielony na Dashboard (Menu) i Widok Treningowy.



Rysunek 3. Panel główny z historią i konfiguracją.

Funkcje GUI:

- Dynamiczne skalowanie podglądu wideo (zachowanie proporcji).
- Wykresy: Po zakończeniu sesji generowany jest wykres kąta kolana w czasie (Matplotlib) z naniesionymi czerwonymi punktami w miejscach wystąpienia błędów.
- Wideo Replay: Możliwość odtworzenia nagrania z sesji wraz z nałożonymi liniami analitycznymi.

PLACEHOLDER RYS 4

Rysunek 4. Widok treningowy.

PLACEHOLDER RYS 5

Rysunek 5. Podsumowanie sesji z wykresem błędów.

7. Wyniki testów skuteczności algorytmu

7.1. Środowisko testowe i metodyka

Pomiary wydajności oraz skuteczności przeprowadzono na stanowisku składającym się z:

- Jednostka obliczeniowa: Laptop z procesorem Intel Core i5/i7 (zintegrowana grafika), system Windows 11.
- Kamera 1 (Front): Wbudowana kamera internetowa 720p.
- Kamera 2 (Side): Smartfon z aplikacją IP Webcam, transmisja po Wi-Fi (2.4 GHz/5 GHz) w rozdzielczości 640x480 px (MJPEG).

Testy wykonywano w pomieszczeniu o średnim natężeniu oświetlenia sztucznego. W celu weryfikacji algorytmu Anti-Cheat, osoba testująca wykonywała serie ruchów poprawnych przeplatanych celowymi błędami technicznymi oraz próbami oszukania licznika (np. unoszenie nogi bez obniżenia biodra).

7.2 Wyniki testów

Przeprowadzono serię 60 prób testowych dla różnych scenariuszy, aby zweryfikować precyzję systemu. Wyniki przedstawia Tabela 1.

Scenariusz testowy	Liczba prób	Poprawne detekcje	Skuteczność	Uwagi
Poprawny wykrok	20	20	100%	**Licznik poprawnie inkrementowany.**
"Skip A" (Oszustwo)	10	0	100%	**System poprawnie zignorował ruch (Anti-Cheat).**
Błąd: Garbienie się	10	10	100%	**Komunikat "Wyprostuj plecy" przy kącie > 20°.**
Błąd: Valgus (Front)	10	9	90%	**1 przypadek niewykryty (zbyt luźne ubranie maskujące kolano).**
Błąd: Kolano przed palce	10	10	100%	**Wykrycie kąta piszczeli > 40° i niezaliczenie powtórzenia.**

Tabela 1. Skuteczność detekcji zdarzeń i błędów.

7.3 Wnioski z testów:

System osiąga całkowitą odporność na próby oszukania licznika dzięki mechanizmowi Hip Drop. Jedynym czynnikiem obniżającym skuteczność detekcji koślawienia (Valgus) jest odzież typu "oversize", która utrudnia modelowi MediaPipe precyzyjną lokalizację rzepki kolanowej.

Ponadto, testy wydajnościowe potwierdziły efektywność zastosowanej optymalizacji wielowątkowej. Średnie opóźnienie (latencja) między wykonaniem ruchu a reakcją systemu (komunikat głosowy/wizualny) wyniosło poniżej 300 ms. Pozwala to na korektę techniki przez użytkownika w czasie rzeczywistym, jeszcze w trakcie trwania powtórzenia, co było kluczowym wymaganiem niefunkcjonalnym projektu.

Weryfikacja modułu głosowego wykazała wysoką odporność na zakłócenia otoczenia. Dzięki zastosowaniu filtru pewności (confidence score > 0.9) oraz ścisłej gramatyki, system poprawnie ignorował rozmowy w tle, reagując wyłącznie na zdefiniowane komendy ('Start', 'Stop', 'Koniec') ze 90% skutecznością w testowanych scenariuszach.

Zaobserwowano również, że kluczowym czynnikiem dla stabilności detekcji jest kontrast między ubiorem ćwiczącego a tłem. W warunkach słabego oświetlenia (< 60 lux), algorytm MediaPipe wykazuje większy rozrzut pomiarowy (jitter), co jest jednak skutecznie niwelowane przez zaimplementowany filtr wygładzający (EMA).

8. Podsumowanie

Projekt LungeGuard spełnił wszystkie założenia funkcjonalne i niefunkcjonalne. Stworzono kompletny system, który nie tylko monitoruje ruch, ale aktywnie uczy poprawnej techniki.

Kluczowym osiągnięciem jest integracja wielu wątków (AI, GUI, Audio, Kamera, Baza Danych) w jedną, stabilną aplikację desktopową, która działa płynnie na przeciętnym sprzęcie.

Projekt ma duży potencjał rozwojowy – możliwe jest dodanie nowych ćwiczeń (np. martwy ciąg) jedynie poprzez zmianę modułu TrainerLogic, bez ingerencji w architekturę systemu.

Bibliografia

[1] Lugaresi, C. et al., "MediaPipe: A Framework for Building Perception Pipelines", *arXiv preprint arXiv:1906.08172*, 2019.

[2] Google AI Edge, "MediaPipe Pose Landmarker Task – Documentation", Dostępne pod adresem: https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker [Dostęp: 2024-01-20].

[3] Winter, D. A., "Biomechanics and Motor Control of Human Movement", 4th Edition, John Wiley & Sons, 2009.

[4] Alpha Cephei, "Vosk Offline Speech Recognition API Documentation", Dostępne pod adresem: https://alphacephei.com/vosk/ [Dostęp: 2024-01-20].

[5] OpenCV Team, "OpenCV Library Documentation", Dostępne pod adresem: https://docs.opencv.org/ [Dostęp: 2024-01-20].

[6] NumPy Documentation, "NumPy User Guide", Dostępne pod adresem: https://numpy.org/doc/ [Dostęp: 2024-01-20].