POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Szymon Cyperski
Nr albumu: 180408
Poziom kształcenia: Studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Automatyka, cybernetyka i robotyka
Profil: Systemy decyzyjne i robotyka

PROJEKT DYPLOMOWY INŻYNIERSKI

Tytuł projektu w języku polskim: System asystenta treningu siłowego przy użyciu wizji komputerowej

Tytuł projektu w języku angielskim: Strength training assistant system using computer vision

Opiekun pracy: dr inż. Mariusz Domżalski

POLITECHNIKA GDAŃSKA | WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: System asystenta treningu siłowego przy użyciu wizji komputerowej

Imię i nazwisko studenta: Szymon Cyperski
Data i miejsce urodzenia: 02.04.1999, Gdańsk
Nr albumu: 180408

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki
Kierunek: automatyka, cybernetyka i robotyka

Poziom kształcenia: pierwszy
Forma studiów: stacjonarne

Typ pracy: projekt dyplomowy inżynierski

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),[1] a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

04.01.2023, Szymon Cyperski

Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG

**) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

[1] Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

STRESZCZENIE

W związku z popularyzacją zdrowego trybu życia coraz więcej osób decyduje się na trening siłowy. Poprawne wykonywanie niektórych ćwiczeń siłowych może być trudne dla osób niedoświadczonych, a popełnianie błędów technicznych niesie za sobą ryzyko utraty zdrowia. W ramach niniejszej pracy został stworzony prototyp aplikacji, która pomaga osobom początkującym podczas treningu siłowego poprzez wykrywanie błędów w technice wykonywania ćwiczeń. Praca składa się z ośmiu rozdziałów, które dokładnie opisują proces tworzenia aplikacji. Dokonano przeglądu podobnych rozwiązań dostępnych na rynku globalnym. Omówiono zagadnienie estymacji pozy człowieka, które jest fundamentem do realizacji całego projektu. Przedstawiono także proces przygotowania zbioru danych zebranego specjalnie na potrzeby niniejszej pracy. Zaprezentowano sposób na realizację systemu asystenta treningowego, który potrafi wykrywać popełnione przez użytkownika błędy techniczne oraz inne zdarzenia w czasie rzeczywistym. Opisano implementację aplikacji komputerowej oraz sposób jej integracji ze stworzonym systemem. Praca zawiera również opis przeprowadzonych testów skuteczności gotowego rozwiązania oraz przedstawia możliwości jego rozwoju w przyszłości.

Słowa kluczowe: estymacja pozy człowieka, uczenie maszynowe, detekcja błędów, trening siłowy

Dziedzina nauki i techniki zgodna z OECD: Nauki o komputerach i informatyka, Robotyka i automatyka, Nauka o sporcie i sprawności fizycznej

ABSTRACT

With the popularization of healthy lifestyles, more and more people are opting for strength training. Correctly performing some strength exercises can be difficult for inexperienced people, and making technical mistakes carries the risk of losing health. Within this work, a prototype of an application has been created to help beginners during strength training by detecting errors in exercise techniques. This paper consists of eight chapters, which describe the process of developing the app in detail. Similar solutions available on the global market are reviewed. The task of human pose estimation, which is the foundation for the implementation of the entire project, is discussed. The process of preparing a dataset collected specifically for this work is also presented. A method to implement a training assistant system, which can detect technical errors made by the user and other events in real-time, is proposed. The implementation of the computer application is described, as well as how to integrate it with the created system. The paper also includes a description of the tests carried out on the effectiveness of the final solution and presents possibilities for its future development.

Keywords: human pose estimation, machine learning, error detection, strength training

OECD consistent field of science and technology classification: Computer and information sciences, Robotics and automatic control, Sport and fitness sciences

SPIS TREŚCI

1. WSTĘP I CEL PRACY

Trening siłowy to rodzaj aktywności sportowej polegającej na obciążaniu mięśni ciężarem własnego ciała lub ciężarem zewnętrznym. Zwiększenie wytrzymałości i siły, korekta postawy ciała czy zmniejszenie ryzyka wystąpienia kontuzji to tylko niektóre z korzyści, jakie może on zapewnić osobom ćwiczącym. W szczególności ćwiczenia wielostawowe z dodatkowym obciążeniem, takie jak przysiad ze sztangą czy martwy ciąg, uwydatniają wymienione wyżej zalety.

Dla osób bez odpowiedniej wiedzy i umiejętności ćwiczenia te są trudne do wykonania przy zachowaniu poprawnej techniki. Natomiast realizowanie ich w sposób nieprawidłowy, zamiast korzyści może nieść za sobą zagrożenie utraty zdrowia. W takich sytuacjach przydatna może być pomoc doświadczonego trenera, lecz zazwyczaj jest ona związana z niemałymi kosztami. W konsekwencji osoby początkujące często decydują się pomijać ćwiczenia wielostawowe w swoim treningu lub zastępują je mniej efektywnymi ćwiczeniami, które izolują poszczególne partie mięśniowe.

1.1. Cel pracy

Celem pracy jest stworzenie prototypu aplikacji mającej pomagać osobom początkującym podczas treningu siłowego. Aplikacja ma być wyposażona w system asystenta treningowego, który będzie w stanie wykrywać w czasie rzeczywistym podstawowe błędy w technice wykonywania ćwiczenia oraz w prosty sposób doradzać jak użytkownik może je poprawić. System będzie potrafił także wykrywać moment rozpoczęcia i zakończenia ćwiczenia, oraz zliczać wykonane przez użytkownika powtórzenia. Działanie systemu zostało ograniczone do jednego ćwiczenia siłowego - przysiadu ze sztangą z tyłu.

System asystenta będzie podejmował decyzje na podstawie analizy położenia punktów kluczowych na ciele osoby ćwiczącej. Informacje o położeniu punktów kluczowych będą dostarczane przez algorytm estymujący pozę człowieka na nagraniu z pojedynczej kamery RGB.

1.2. Zakres pracy

zakres pracy obejmuje następujące zagadnienia:

1. zapoznanie się z istniejącymi algorytmami estymującymi pozę człowieka i wybór rozwiązania optymalnego na potrzeby pracy
2. zapoznanie się z zasadami poprawnego wykonywania ćwiczeń siłowych i przygotowanie podzbioru błędów technicznych potencjalnie możliwych do wykrycia przy pomocy dostępnych narzędzi
3. zebranie i przygotowanie zbioru danych z nagraniami treningowymi, zawierającego przykłady z wybranymi błędami technicznymi
4. zaprojektowanie i implementacja systemu asystenta treningowego za pomocą modeli uczenia maszynowego wytrenowanych na przygotowanym zbiorze danych
5. integracja i wdrożenie systemu asystenta treningowego do aplikacji komputerowej
6. testy wytworzonego systemu na rzeczywistych nagraniach treningowych.

2. PRZEGLĄD ROZWIĄZAŃ

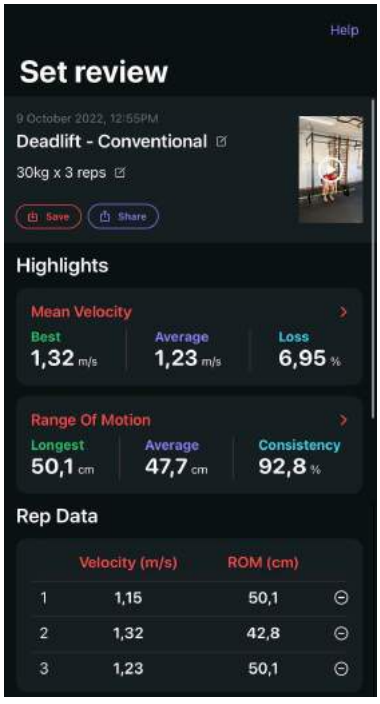
W związku z popularyzacją zdrowego trybu życia w ostatnim czasie na rynku pojawia się coraz więcej aplikacji i narzędzi wspierających osoby ćwiczące siłowo. Oprócz aplikacji umożliwiających prowadzenie dzienników dietetycznych i treningowych powstają także rozwiązania pozwalające analizować jakość wykonywania ćwiczeń na podstawie nagrania z kamery. Przygotowany został wgląd w kilka rozwiązań dostępnych na rynku globalnym, które są w największym stopniu zbliżone do systemu, będącego tematem niniejszej pracy.

2.1. Metric VBT [1]

Aplikacja mobilna dostępna dla systemu iOS, pozwalająca użytkownikowi na nagranie wykonania jednego z wielu dostępnych ćwiczeń. System na podstawie nagrania z kamery analizuje zrealizowane przez użytkownika powtórzenia. Po zakończeniu serii wyświetlane jest podsumowanie widoczne na rys. 2.1. Zawiera ono następujące informacje i statystyki:

- liczbę wykonanych powtórzeń
- prędkość wykonanego ruchu dla każdego powtórzenia
- zakres ruchu dla każdego powtórzenia
- statystki zbiorcze dla całej wykonanej serii powtórzeń (m.in. średnią prędkość i średni zakres ruchu)

Dodatkowo użytkownik może zapisać w pamięci nagranie z wykonanej serii wraz z jej datą w celu późniejszych analiz i porównań.



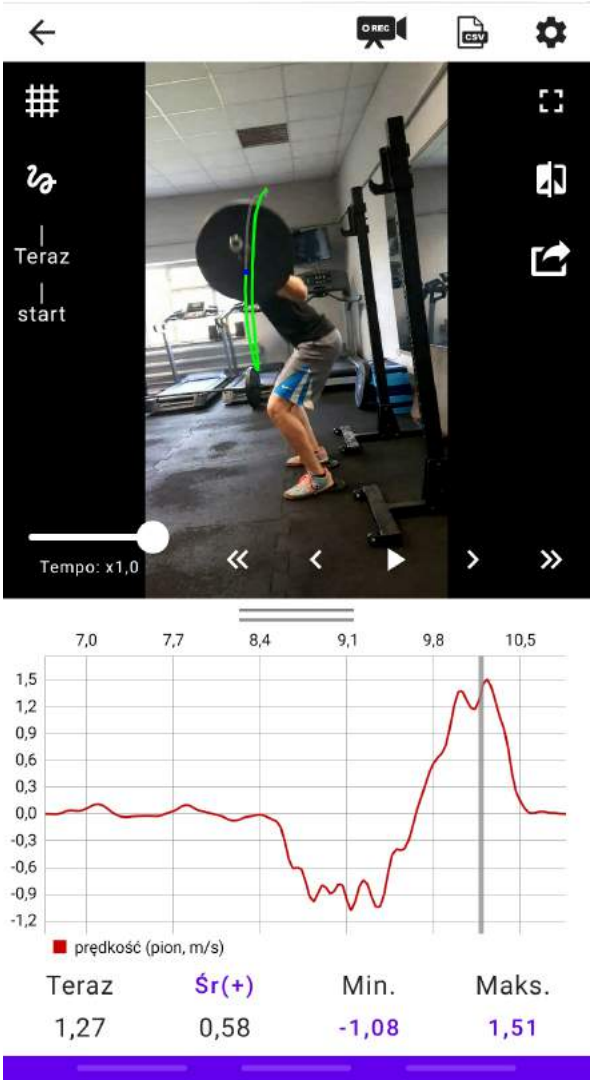
Rysunek 2.1: Widok z aplikacji Metric VBT

Źródło: opracowanie własne

2.2. WL Analysis [2]

Aplikacja mobilna dostępna dla systemów Android oraz iOS, służąca do analizy trajektorii ruchu sztangi podczas wykonywania ćwiczeń. Wymagane jest usytuowanie kamery z boku osoby ćwiczącej. Aplikacja automatycznie wykrywa bliższą końcówkę sztangi i śledzi jej trajektorię, jednocześnie wyliczając podstawowe parametry kinematyczne ruchu. Po wykonanym ćwiczeniu użytkownik może odtworzyć nagranie wraz z naniesioną trajektorią ruchu sztangi, co zostało przedstawione na rys. 2.2. W aplikacji można także wyświetlić wykres czasowy zmieniających się prędkości i przyspieszenia sztangi. Produkt oferuje także inne funkcjonalności wymienione poniżej (niektóre z nich są dodatkowo płatne):

- porównywanie dwóch lub więcej nagrań jednocześnie
- eksport nagrania do pliku wraz z dodatkowymi informacjami o wykonanym ćwiczeniu
- eksport wyliczonych parametrów kinematycznych do pliku z rozszerzeniem "csv".



Rysunek 2.2: Widok z aplikacji WL Analysis

Źródło: opracowanie własne

2.3. Kemtai [3]

Platforma webowa umożliwiająca także dostęp z poziomu API. Aplikacja za pomocą nagrania z kamery śledzi położenie do 44 punktów kluczowych na ciele użytkownika i na tej podstawie ocenia każde wykonane przez niego powtórzenie w czasie rzeczywistym. W przypadku wykrycia błędu użytkownikowi są wyświetlane oraz głosowo odtwarzane wskazówki, co należy poprawić. Liczba błędów obsługiwanych przez aplikację jest znacznie ograniczona. Dodatkowo niektóre z nich są mocno uzależnione od indywidualnych parametrów użytkownika, których aplikacja nie uwzględnia, przez co często sygnalizowane są fałszywe popełnienia błędów (np. widoczny na rys. 2.3 błąd związany z szerokością rozstawu stóp, który w znacznej mierze zależy od budowy anatomicznej osoby wykonującej ćwiczenie).

Aplikacja stworzona jest głównie do użytku w warunkach domowych. Swoim zakresem obejmuje kilkaset różnych ćwiczeń – m.in. rozciągające, fizjoterapeutyczne oraz siłowe. Jednak w przypadku tych ostatnich dostępne są głównie ćwiczenia wykonywane z ciężarem własnego ciała. Użytkownik nie ma możliwości dobrania interesujących go ćwiczeń ani określenia liczby powtórzeń do wykonania - dostępne są jedynie gotowe treningi z ustalonymi ćwiczeniami. Ponadto, znaczna większość z nich wymaga zakupu płatnej subskrypcji.

Rysunek 2.3: Widok z aplikacji Kemtai

Źródło: opracowanie własne

2.4. Podsumowanie

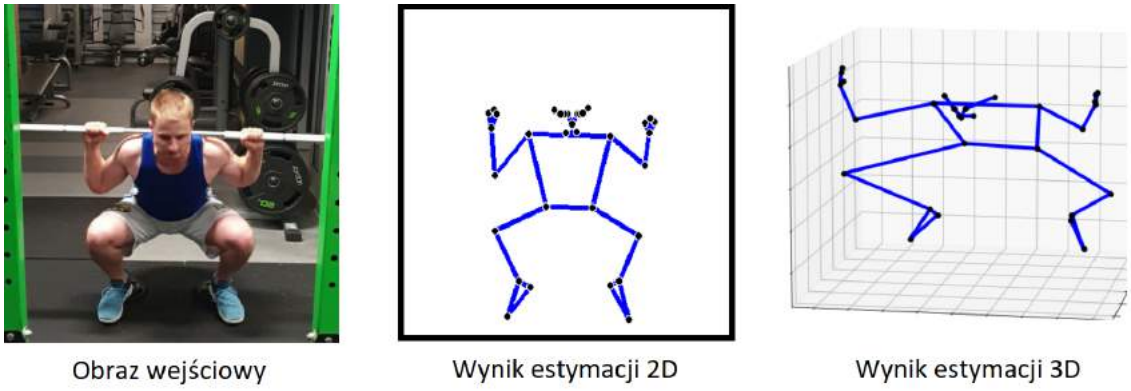
Spośród wszystkich przedstawionych powyżej aplikacji jedynie platforma Kemtai oferuje funkcjonalności podobne do tych, które ma zapewniać system, będący tematem niniejszej pracy. Pozostałe aplikacje, pomimo iż mogą przyczynić się do ogólnej poprawy jakości wykonywania ćwiczeń, nie uchronią osób początkujących przed popełnianiem błędów technicznych. Brak większej liczby bliźniaczych rozwiązań na rynku świadczy o innowacyjności takiego systemu w skali globalnej.

3. ESTYMACJA POZY CZŁOWIEKA

Rozdział ma na celu przedstawienie zagadnienia estymacji pozy człowieka, które jest fundamentem do realizacji systemu asystenta treningowego. Zostały omówione podstawy teoretyczne oraz opisano niektóre algorytmy, w tym także model wykorzystany w niniejszej pracy.

3.1. Wprowadzenie teoretyczne

Estymacja pozy człowieka jest popularnym zagadnieniem z dziedziny wizji komputerowej. Dotyczy ona problemu identyfikacji oraz wyznaczania położenia punktów kluczowych ciała człowieka na zdjęciach lub nagraniach. Przykładowe rezultaty estymacji pozy ilustruje rys. 3.1. Zasadniczo algorytmy estymacji pozy można podzielić według trzech kryteriów [4]:

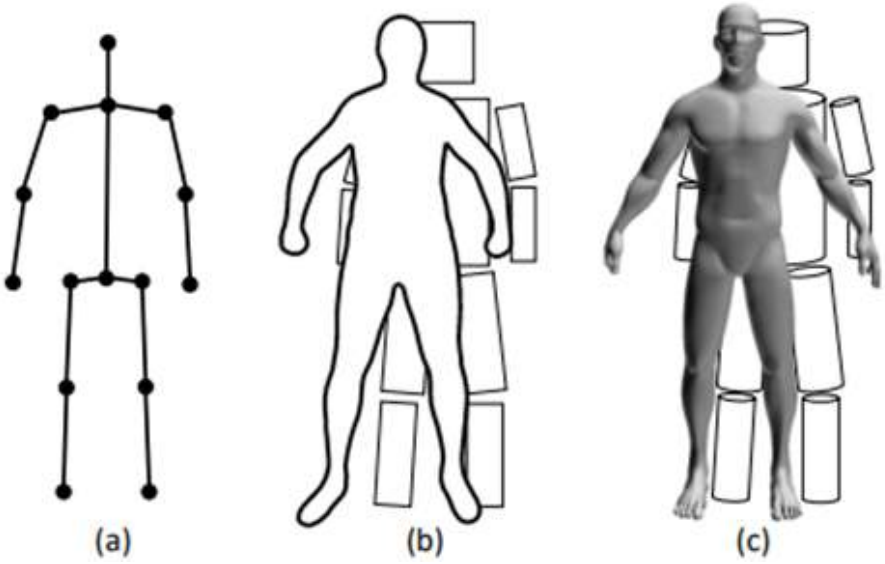


Rysunek 3.1: Estymacja pozy człowieka

Źródło: opracowanie własne

- Liczba wymiarów przestrzeni wynikowej:
  - Estymacja 2D – szacowanie położenia punktów kluczowych w przestrzeni dwuwymiarowej (współrzędne X i Y).
  - Estymacja 3D – transformacja punktów kluczowych pozyskanych z płaskiego obrazu do przestrzeni trójwymiarowej (dodatkowy wymiar Z określający odległość punktów od kamery).
- Liczba estymowanych sylwetek:
  - Pojedyncza sylwetka – szacowane jest położenie punktów kluczowych na ciele jedynie jednej osoby.
  - Wiele sylwetek – estymowane są położenia wszystkich sylwetek, które zostaną znalezione na obrazie.
- Typ modelu ludzkiego ciała (przedstawione na rys. 3.2):
  - Model kinematyczny (szkieletowy) – struktura szkieletu oparta o punkty kluczowe na ciele, będące najczęściej stawami kostnymi (np. stawy kolanowe). Liczba punktów i ich umiejscowienie na ciele jest kwestią indywidualną dla danego algorytmu.

- Model konturowy (planarny) – używany tylko w przypadku estymacji 2D. Jest wykorzystywany w celu uzyskania kształtu i konturu sylwetki. Części ciała są często reprezentowane za pomocą wielu prostokątów.
- Model objętościowy – używany tylko w przypadku estymacji 3D. Sylwetka człowieka jest reprezentowana za pomocą jednej z wielu predefiniowanych geometrycznych siatek dopasowanych do ludzkiego ciała.



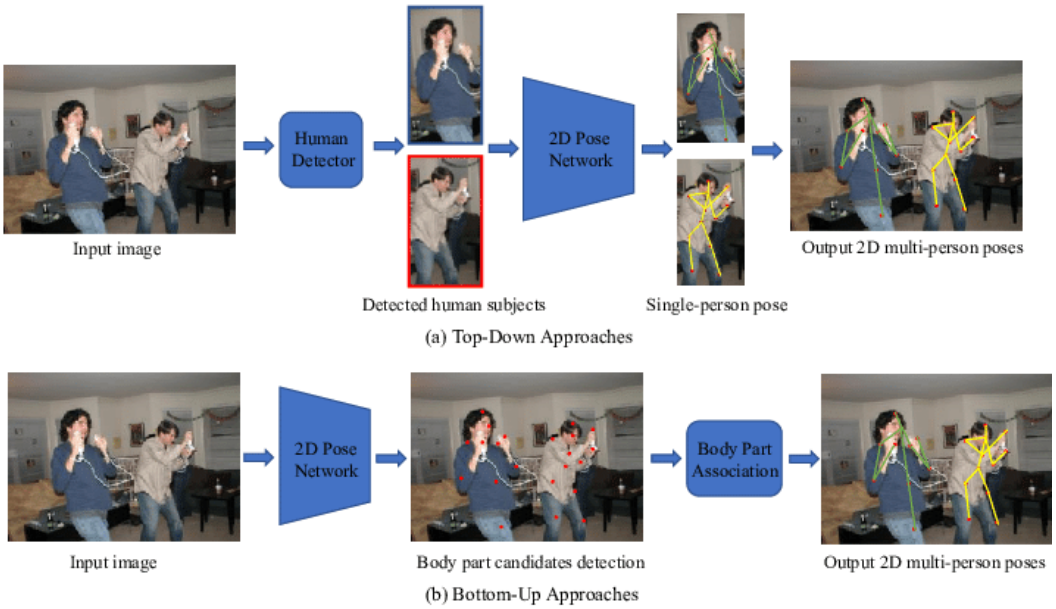
Rysunek 3.2: Typy modelu ludzkiego ciała wykorzystywane w algorytmach estymacji pozy: (a) kinematyczny; (b) konturowy; (c) objętościowy

Źródło: [4]

Nowoczesne modele do estymacji pozy opierają się głównie na uczeniu głębokim i konwolucyjnych sieciach neuronowych (CNN - convolutional neural network). W szczególności można spotkać 2 podstawowe metody ich działania [4]:

- Metoda *top-down* – najpierw wykrywany jest prostokąt okalający człowieka na obrazie, a następnie wewnątrz niego szacowane są położenia punktów kluczowych na ciele. Estymacja pozy jest wykonywana dla każdej sylwetki, dlatego czas przetwarzania rośnie proporcjonalnie do liczby wykrytych osób na obrazie. Krytycznym etapem w tej metodzie jest detekcja człowieka – jej skuteczność i dokładność znacząco wpływa na wynik estymacji pozy.
- Metoda *bottom-up* – na całym obrazie od razu wykrywane są wszystkie punkty kluczowe bez identyfikacji poszczególnych osób. Następnie punkty te są przypisywane do właściwych sylwetek za pomocą różnych technik. Zagadnienie to jest równoważne problemowi *K-dimensional matching* [1], o którym wiadomo, że jest NP-trudny. Etap grupowania znalezionych punktów ma największy wpływ na szybkość i skuteczność całej estymacji.

[1] *K-dimensional matching* - problem znalezienia największego skojarzenia w k-dzielnym r-jednolitym hipergrafie [5].



Rysunek 3.3: Porównanie metod działania algorytmów estymacji pozy: (a) metoda top-down; (b) metoda bottom-up

Źródło: [4]

Podział na metody działania przedstawione na rys. 3.3 dotyczy estymacji wielu sylwetek jednocześnie. W przypadku estymacji jednej sylwetki metody działania są uproszczone i zazwyczaj sprowadzają się do:

1. Detekcji człowieka na obrazie - wyznaczenia prostokąta okalającego
2. Estymacji punktów kluczowych wewnątrz wyznaczonego prostokąta.

Istnieją także dwa podstawowe kryteria przy wyborze odpowiedniego modelu do konkretnego zastosowania:

- Szybkość dokonywania estymacji – podstawową stosowaną metryką jest liczba przetworzonych klatek na sekundę (fps - frames per second). Wartość tej metryki jest silnie uzależniona od specyfikacji urządzenia, na którym uruchamiany jest model, oraz od rozmiaru obrazu wejściowego.
- Dokładność estymacji – istnieje wiele metryk określających dokładność modelu. Jedną z najczęściej używanych jest Percentage of Correct Key-points (PCK), wedle której punkt jest uznawany za poprawnie przewidziany, jeśli odległość pomiędzy estymacją a prawdziwym położeniem punktu jest mniejsza niż ustalona wartość progowa. Dla jednej z wersji tej metryki, PCK@0.2, próg ten wynosi: 0.2 · odległość pomiędzy lewym barkiem a prawym biodrem.

3.2. Przegląd popularnych modeli

Przygotowany został wgląd w niektóre z najpopularniejszych modeli estymujących pozę człowieka.

3.2.1. OpenPose [6]

Model estymacji pozy o otwartym kodzie źródłowym zaimplementowany w języku C++. Umożliwia estymację 2D 135 punktów kluczowych na ciałach wielu osób w czasie rzeczywistym. Jego działanie opiera się na metodzie *bottom-up*. Rysunek 3.4 przedstawia proces uzyskiwania predykcji - można go podsumować w następujących krokach:

1. Wygenerowanie map cech z obrazu wejściowego za pomocą sieci konwolucyjnej, bazującej na 10 pierwszych warstwach modelu VGG-19[2].

2. Jednoczesne przetworzenie uzyskanych map cech za pomocą dwóch gałęzi wieloetapowej sieci CNN:

   (a) Górna gałąź przewiduje Part Confidence Maps (PCM) – dwuwymiarowe mapy pewności reprezentujące przekonanie o lokalizacjach wystąpienia punktu kluczowego na obrazie. Generowana jest jedna mapa dla każdego rodzaju punktu kluczowego.

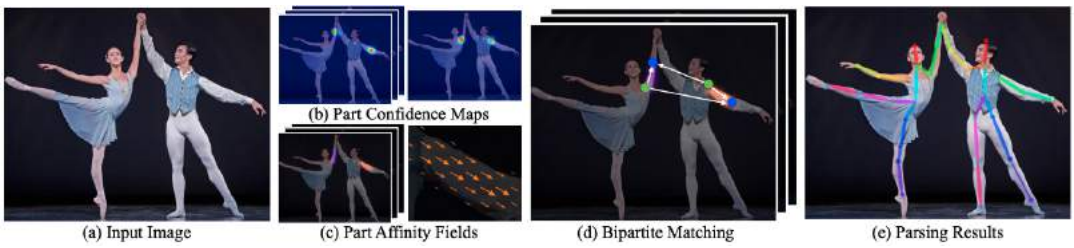
   (b) Dolna gałąź przewiduje Part Affinity Fields (PAF) – pola powinowactwa o elementach będących wektorami dwuwymiarowymi, które kodują kierunek z jednego końca kończyny do drugiego. Generowane jest jedno pole dla każdej pary punktów (kończyny), która stanowi część predefiniowanego szkieletu człowieka.

3. W celu uzyskania lepszych predykcji wyniki z obu gałęzi wraz pierwotnymi mapami cech są następnie przetwarzane za pomocą kilku kolejnych etapów sieci CNN, również zawierających dwie gałęzie opisane powyżej.

4. Uzyskanie lokalizacji punktów kluczowych z PCM za pomocą tłumienia niemaksymalnego[3].

5. Dla każdego PAF obliczane są miary powiązania pomiędzy parami punktów (kandydaci do rzeczywistej kończyny) za pomocą całki krzywoliniowej.

6. Grupowanie punktów do właściwych sylwetek za pomocą skojarzeń w grafach dwudzielnych.



Rysunek 3.4: Potok przetwarzania w modelu OpenPose: (a) obraz wejściowy; (b) Part Confidence Maps; (c) Part Affinity Fields; (d) grupowanie punktów; (e) wynik estymacji

Źródło: [6]

3.2.2. AlphaPose [8]

Model estymacji pozy dla wielu sylwetek o otwartym kodzie źródłowym. Został zaimplementowany za pomocą popularnego pakietu PyTorch, wykorzystywanego głównie do uczenia głębokiego. Podstawowym zamierzeniem projektu było przeciwdziałanie wpływowi niedokładnych oraz nadmiarowych detekcji człowieka na estymację pozy w metodzie *top-down*. Proces dokonywania estymacji jest przedstawiony na rys. 3.5 i składa się z następujących etapów:

[2]VGG-19 - 19-warstwowy wariant popularnej sieci konwolucyjnej VGG stworzonej przez zespół z laboratorium Visual Geometry Group (VGG). Model został wytrenowany na zbiorze ImegeNet, zawierającym kilkanaście milionów zdjęć różnych obiektów, i zajął drugie miejsce w 2014 roku w konkursie ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) [7].

[3]Tłumienie niemaksymalne (NMS - non-maximum suppression) - algorytm stosowany jako końcowy komponent w rozwiązaniach z dziedziny detekcji obiektów. Jest odpowiedzialny za redukcję nadmiarowych detekcji, czyli wybór najbardziej właściwego prostokąta okalającego dany obiekt.

1. Detekcja sylwetek ludzi za pomocą modelu SSD[4].

2. Estymacja proponowanych póz za pomocą modułu, działającego w następujących krokach:

   (a) Transformacja regionów z wykrytymi sylwetkami (prostokątów okalających) za pomocą sieci STN[5] w celu poprawienia ich jakości.

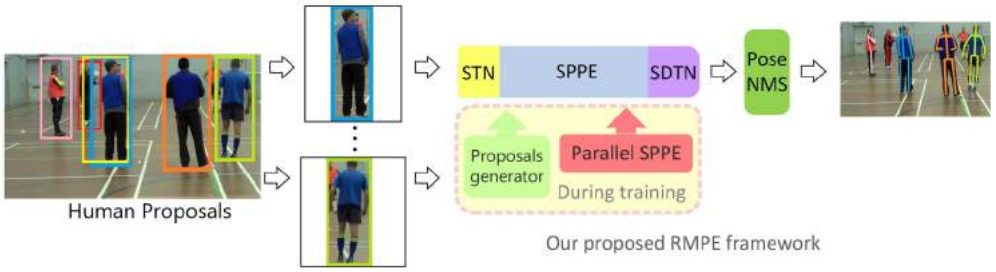
   (b) Estymacja punktów kluczowych wewnątrz skorygowanych prostokątów za pomocą algorytmu SPPE (single person pose estimation), który dokonuje predykcji dla pojedynczej sylwetki.

   (c) Zmapowanie uzyskanej estymacji pozy do współrzędnych pierwotnego obrazu za pomocą Spatial De-Transformer Network (SDTN), czyli sieci dokonującej przekształceń odwrotnych do tych zrealizowanych przez STN.

   W trakcie treningu wykorzystano dodatkową, równoległą gałąź z modułem SSPE. Pozwoliła ona uzyskiwać dokładniejsze transformacje (centrowanie) regionów z wykrytymi sylwetkami za pomocą STN, dzięki lepszemu unikaniu lokalnych minimów.

3. Eliminacja nadmiarowych, wyestymowanych póz za pomocą algorytmu parametric pose non-maximum suppression (p-Pose NMS), który pełni rolę analogiczną do algorytmu NMS w trakcie detekcji obiektów, wykorzystując to tego celu miarę podobieństwa pomiędzy pozami.

Dodatkowo, w trakcie uczenia wykorzystano Pose-guided Proposals Generator (PGPG), czyli generator pozwalający uzyskiwać dużą liczbę próbek treningowych symulujących wyjście z detektora osób z etapu 1. Taka augmentacja danych pozwoliła modułowi dokonującemu estymacji na lepszą adaptację do niedokładnych regionów z wykrytymi sylwetkami dostarczanych przez detektor.



Rysunek 3.5: Potok przetwarzania w modelu AlphaPose

Źródło: [8]

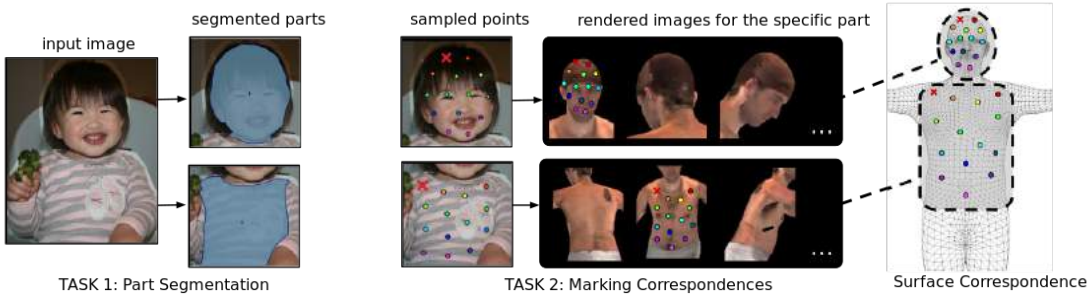
3.2.3. DensePose [11]

Projekt stworzony przez zespół badawczy firmy Facebook. Opracowany przez nich model potrafi w czasie rzeczywistym mapować wiele sylwetek wykrytych na pojedynczym, dwuwymiarowym obrazie na trójwymiarowe powierzchnie reprezentujące ludzkie ciała. Zatem, w odróżnieniu od rozwiązań opisanych w poprzednich podpunktach, DensePose do reprezentacji sylwetki człowieka wykorzystuje model objętościowy zamiast modelu szkieletowego.

[4]SSD (Single Shot Detector) - model bazujący na sieci VGG przeznaczony do detekcji obiektów w czasie rzeczywistym. Charakteryzuje się znacznie krótszym czasem dokonywania predykcji przy zachowaniu skuteczności porównywalnej do konkurencyjnych modeli [9].

[5]STN (Spatial Transformer Network) - sieć konwolucyjna zawierająca moduł transformatora przestrzennego (Spatial Transformer), który jest w stanie dokonywać przekształceń na obrazach lub mapach cech (m.in. skalowanie, rotacja, przycinanie). Włączenie transformatora do architektury sieci pozwala uzyskać model bardziej niezmienniczy względem przekształceń na danych wejściowych [10].

Podstawą do prac nad tym modelem był zbiór danych COCO-DensePose, stworzony specjalnie na potrzeby tego projektu. Zawiera on powiązania sylwetek ludzi na obrazach z trójwymiarowymi modelami objętościowymi dla 50 tysięcy zdjęć ze zbioru danych COCO[6]. Przedstawiony na rys. 3.6 ręczny proces oznaczania danych składał się z dwóch etapów - najpierw adnotatorzy wyznaczali obszary odpowiadające różnym częściom ciała (np. głowa, tułów), a następnie wewnątrz każdego z nich dokładnie odwzorowywali położenie zestawu punktów próbkujących dany obszar. W trakcie przygotowywania zbioru przywiązywano wagę do dokładności oznaczeń, stosując różne metryki jej ewaluacji.



Rysunek 3.6: Proces oznaczania danych w zbiorze COCO-DensePose

Źródło: [11]

Architektura modelu bazuje na dwóch innych gotowych rozwiązaniach:

- sieci konwolucyjnej DenseReg, w której zadanie mapowania twarzy z obrazów na ich trójwymiarowe szablony zdefiniowano jako problem regresyjny [13]
- modelu Mask R-CNN [14], który dzięki dodaniu dodatkowej gałęzi do architektury sieci Faster R-CNN [15] przeznaczonej do detekcji obiektów w czasie rzeczywistym, jest w stanie również realizować segmentację semantyczną obiektów.

W modelu DensePose za dokonywanie predykcji odpowiedzialna jest pełna sieć konwolucyjna (FCN - fully convolutional network), która realizuje zadania klasyfikacji oraz regresji. Najpierw pikselom z obrazu przypisywana jest etykieta odpowiedniego obszaru ciała lub tła, a następnie gałąź regresyjna sieci przewiduje dokładne współrzędne umiejscowienia pikseli wewnątrz tego obszaru. Ze względu na skomplikowaną strukturę ludzkiego ciała, dla każdego obszaru został dobrany lokalny, sparametryzowany układ współrzędnych.

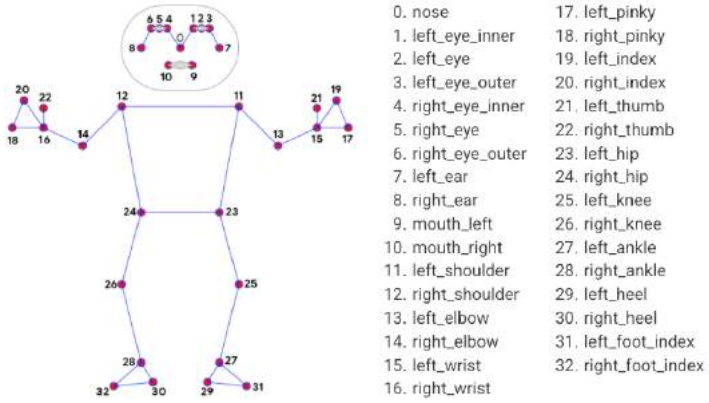
Aby usprawnić działanie modelu, na początku jego architektury wykorzystano moduł FPN[7] do wydobycia map cech. W dalszej kolejności zastosowano podejście oparte o regiony zainteresowań (ROI - region of interest), czyli rozbicie map cech na propozycje najbardziej trafnych obszarów. Pozwoliło to na zdekomponowanie całego zadania, ponieważ sieć FCN dokonująca predykcji jest umieszczona na samym końcu architektury modelu.

[6]COCO - jeden z największych zbiorów danych wykorzystywany do tworzenia modeli z obszaru wizji komputerowej. Zawiera ponad 200 tysięcy zdjęć wraz z etykietami przeznaczonymi do różnych zadań (m.in. segmentacja semantyczna, detekcja, klasyfikacja) [12].

[7]FPN (Feature Pyramid Network) - komponent wykorzystywany do początkowej ekstrakcji cech z obrazu wejściowego w modelach przeznaczonych do detekcji obiektów. FPN generuje mapy cech zawierające dokładniejsze informacje przestrzenne, dzięki dodaniu dodatkowych połączeń pomiędzy jego dwoma ścieżkami złożonymi z warstw konwolucyjnych [16].

3.3. Model wykorzystany w pracy - BlazePose [17]

Lekki model do estymacji pozy człowieka, który działa w czasie rzeczywistym i jest dedykowany na urządzenia mobilne. Został stworzony przez zespół badawczy firmy Google razem z bliźniaczymi rozwiązaniami przeznaczonymi do estymacji twarzy (BlazeFace [18]) oraz estymacji dłoni (MediaPipe Hands [19]). Algorytm jest w stanie szacować położenie (w przestrzeni 2D oraz 3D) 33 punktów kluczowych na ciele człowieka, które są przedstawione na rys. 3.7. Model, wraz z wieloma innymi rozwiązaniami przeznaczonymi do różnych zadań z obszaru wizji komputerowej, jest dostępny na bazie licencji Open Source w kilku różnych językach (m.in. Python, C++, Android) za pośrednictwem pakietu MediaPipe.



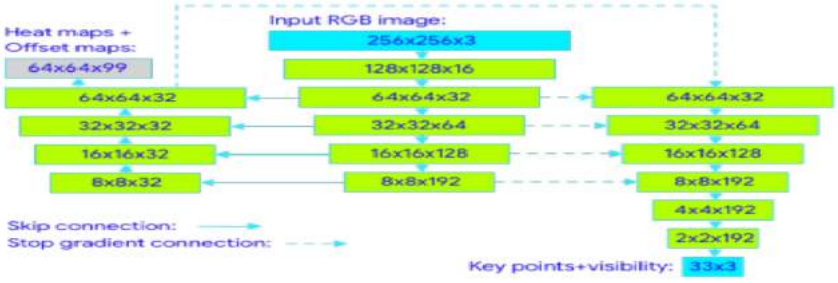
Rysunek 3.7: Punkty kluczowe na ciele człowieka estymowane przez model BlazePose

Źródło: [17]

Estymacja pozy jest wykonywana za pomocą dwóch modułów - detektora, który jest odpowiedzialny za zlokalizowanie ROI z sylwetką człowieka na obrazie, oraz estymatora, który przewiduje położenie punktów kluczowych wewnątrz zwróconego obszaru. Autorzy, chcąc maksymalnie ograniczyć wagę i przyspieszyć działanie modelu, zamiast detektora całej sylwetki wykorzystali detektor twarzy z projektu [18], który dodatkowo przewiduje następujące parametry:

- położenie punktu środkowego pomiędzy ludzkimi biodrami
- promień koła otaczającego sylwetkę człowieka
- kąt pomiędzy odcinkiem łączącym biodra oraz odcinkiem łączącym barki.

Te trzy wartości pozwalają opisać środek, rotację i rozmiar ciała. Jednak skutkiem takiego uproszczenia jest konieczność, by twarz człowieka była widoczna na obrazie.

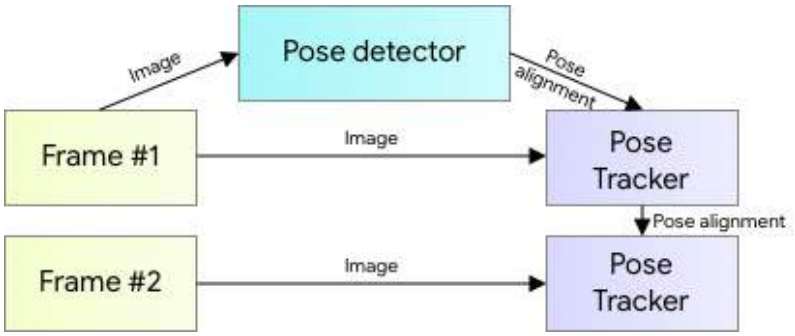


Rysunek 3.8: Architektura sieci konwolucyjnej estymującej położenie punktów kluczowych w modelu BlazePose

Źródło: [17]

Widoczna na rys. 3.8 architektura estymatora zawiera blok wykorzystujący mapy ciepła oraz przesunięcia punktów kluczowych, jednak jego warstwy są usuwane po zakończeniu treningu w celu przyspieszenia fazy uzyskiwania estymacji. Za obliczanie predykcji odpowiedzialny jest blok regresyjny widoczny z prawej strony. W trakcie uczenia zrezygnowano z rozsyłania jego gradientów podczas propagacji wstecznej, co pozwoliło uzyskać jeszcze lepsze rezultaty.

BlazePose posiada także duże usprawnienie widoczne na rys. 3.9, które skraca jego czas obliczeń podczas użytkowania go na nagraniach. Mianowicie, detektor jest uruchamiany wyłącznie dla pierwszej klatki nagrania. Dla kolejnych klatek ROI jest wnioskowane na podstawie wyniku estymacji punktów kluczowych z klatek poprzedzających. Ewentualne, ponowne wykorzystanie detektora następuje w momencie spadku wartości oznaczających widoczność punktów kluczowych na obrazie poniżej określonego progu (widoczność punktów kluczowych jest również estymowana i zwracana przez model).



Rysunek 3.9: Potok przetwarzania w modelu BlazePose

Źródło: [17]

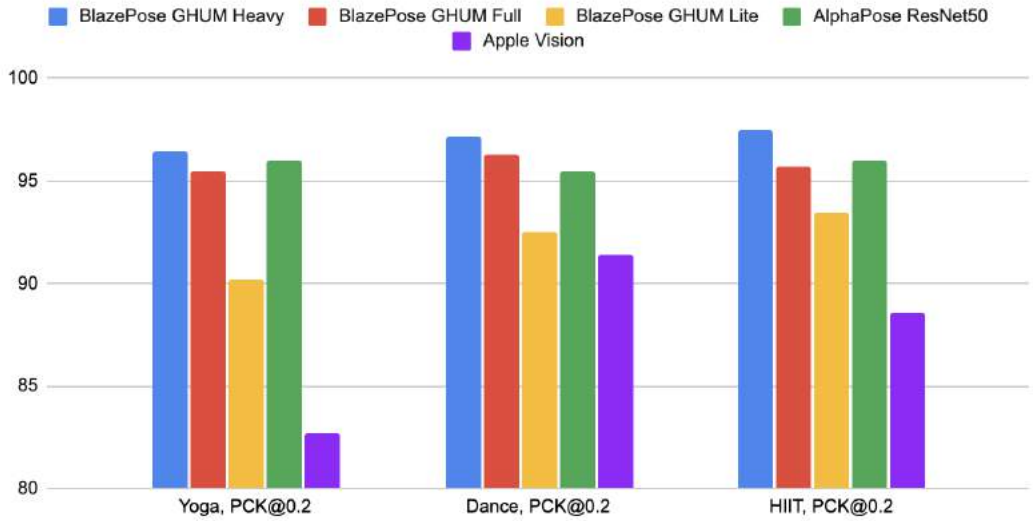
Aby umożliwić estymację pozy w 3D [20], model BlazePose został ponownie wytrenowany na syntetycznych danych trójwymiarowych. Zbiór treningowy pozyskano dzięki dopasowaniu algorytmu GHUM [21] do istniejących już zbiorów 2D, co zostało przedstawione na rys. 3.10. GHUM jest przeznaczony do trójwymiarowej rekonstrukcji ludzkiego ciała za pomocą jednej z dwóch geometrycznych siatek o różnych rozdzielczościach (dla GHUM siatka składa się z 10168 wierzchołków; w przypadku GHUML(ite) z 3194 wierzchołków). Bazą do stworzenia tego algorytmu był zbiór danych GHS3D, który zawiera ponad 60 tysięcy dokładnych, trójwymiarowych skanów ludzkich sylwetek w różnych pozycjach.

Rysunek 3.10: Przykładowa rekonstrukcja 3D ludzkiego ciała za pomocą algorytmu GHUM

Źródło: [20]

Trzecia współrzędna zwracana przez model (współrzędna z) oznacza głębokość, czyli odległość punktów kluczowych od kamery. Dokładność estymacji w tym wymiarze jest znacznie gorsza niż dla pozostałych dwóch współrzędnych (x i y). Początek osi Z jest usytuowany w punkcie środkowym pomiędzy biodrami estymowanej pozy. Punkty występujące pomiędzy biodrami a kamerą przyjmują wartości ujemne dla tej współrzędnej, natomiast te z tyłu bioder - dodatnie.

Dodatkowo model BlazePose jest dostępny w trzech wersjach (Heavy, Full oraz Lite) różniących się liczbą wytrenowanych parametrów (w konsekwencji także rozmiarem) oraz dokładnością i czasem predykcji. Na rys. 3.11 można zauważyć stosunkowo niewielkie różnice w skuteczności dla wersji Heavy oraz Full wyrażone za pomocą metryki PCK@0.2. Również popularny model AlphaPose (szerzej opisany w Sekcja 3.2.2) charakteryzuje się dokładnością zbliżoną do obu wspomnianych wersji. Z kolei najmniejsza wersja BlazePose (wersja Lite), a także rozwiązanie zaproponowane przez firmę Apple, skutecznością znacząco odbiegają od topowej trójki.



Rysunek 3.11: Porównanie wartości metryki PCK@0.2 uzyskanych przez modele do estymacji pozy na trzech różnych zbiorach testowych

Źródło: [22]

W przypadku drugiego ważnego kryterium oceny dla modeli estymacji pozy, jakim jest szybkość obliczeń, ranking trzech wersji BlazePose odwraca się o 180 stopni. W Tabela 3.1 możemy zauważyć drastyczną różnicę w liczbie przetwarzanych klatek na sekundę pomiędzy wersjami Heavy oraz Full. Biorąc pod uwagę zbliżone dokładności oraz ponad dwukrotnie większą szybkość obliczeń na urządzeniu mobilnym, to właśnie wersja Full modelu BlazePose zostanie wykorzystana w niniejszej pracy jako algorytm estymujący pozę człowieka.

Tabela 3.1: Porównanie czasów predykcji dla różnych wersji modelu BlazePose

Model	**Czas obliczeń [ms] (FPS)**	
	Smartfon Pixel 3	*MacBook Pro 2017*
BlazePose GHUM Heavy	53 (18,9)	38 (26,3)
BlazePose GHUM Full	25 (40)	27 (37)
BlazePose GHUM Lite	20 (50)	25 (40)

4. ZBIÓR DANYCH

Najprostszym sposobem na stworzenie systemu przeznaczonego do wykrywania błędów technicznych w treningu siłowym byłaby jego implementacja w oparciu o reguły, które w sposób jednoznaczny opisywałyby występowanie błędów w danym ćwiczeniu. Reguły te mogłyby być opisane np. na podstawie różnych relacji kątowych i odległościowych pomiędzy kończynami osoby wykonującej ćwiczenie. Niestety w literaturze sportowej na próżno szukać takich matematycznych definicji reguł, ponieważ w rzeczywistości trenerzy sportów siłowych fakt wystąpienia błędu technicznego stwierdzają na podstawie własnego doświadczenia i wiedzy dziedzinowej, którą trudno wyrazić w tak jednoznaczny sposób. Nie pomaga tutaj również fakt, że stwierdzenie wystąpienia niektórych błędów technicznych jest uzależnione od znajomości m.in. budowy ciała czy długości kończyn osoby wykonującej ćwiczenie.

Innym sposobem na realizację systemu asystenta treningowego, na który zdecydowano się w niniejszej pracy, jest wykorzystanie do tego celu uczenia maszynowego. Jednak stworzenie modeli klasyfikujących położenie punktów kluczowych osoby ćwiczącej na etykiety wystąpienia lub braku wystąpienia danego błędu wymaga oznaczonego zbioru danych z przykładami, gdzie te błędy rzeczywiście wystąpiły. Niestety brakuje publicznie dostępnych zbiorów danych z nagraniami lub zdjęciami z treningu siłowego, które w znacznej mierze zawierałyby przykłady z błędami technicznymi.

Potencjalnym sposobem na zebranie takich nagrań mogłoby być wykorzystanie do tego celu Web Scraping'u[8], jednak zrezygnowano z tego pomysłu z uwagi na niską jakość znacznej części nagrań w Internecie, brak kontroli nad usytuowaniem kamery oraz brak gwarancji wystąpienia wybranych błędów technicznych na nagraniach. Druga metoda na zebranie zbioru danych polegała na przedstawieniu pomysłu na system, który jest tematem niniejszej pracy, wewnątrz grup zrzeszających sympatyków sportów siłowych na popularnych portalach społecznościowych i zwrócenie się z prośbą o wysyłanie prywatnych nagrań treningowych poprzez specjalnie stworzony do tego celu, prosty formularz internetowy. Pomimo wyraźnego zainteresowania ideą i działaniem takiej aplikacji, niewiele osób zdecydowało się udostępnić swoje materiały wideo. W dalszej kolejności zbiór danych próbowano zgromadzić prosząc pojedyncze osoby trenujące na jednej z siłowni o zgodę na nagranie wykonywanych przez nie ćwiczeń siłowych. Niestety podejście to okazało się nieefektywne, ponieważ zgodę na zarejestrowanie swoich ćwiczeń wyrażały głównie osoby zaawansowane treningowo, których technika była bardzo dobra. Skutkiem tego zebranie odpowiedniej liczby przykładów z błędami technicznymi pochłonęłoby mnóstwo czasu.

Ostatecznie na potrzeby wyuczenia modeli klasyfikacyjnych, wchodzących w skład systemu asystenta treningowego, zdecydowano się na zgromadzenie zbioru danych od podstaw w sposób zorganizowany. Proces zbierania nagrań treningowych, ich przygotowania i oznaczania został omówiony w niniejszym rozdziale.

[8] Web Scraping - technika automatycznego pozyskiwania danych ze stron internetowych poprzez kopiowanie ich zawartości do lokalnych baz danych.

4.1. Błędy techniczne w treningu siłowym

Przed przystąpieniem do zbierania nagrań treningowych należało wybrać podzbiór najczęstszych błędów technicznych wykonywanych przez osoby początkujące podczas przysiadu ze sztangą z tyłu. Jako cel obrano sobie wyłonienie takich błędów, których wykonywanie jest niewskazane i/lub może być szkodliwe bez względu na indywidualne parametry osoby ćwiczącej. Po konsultacjach z kilkoma trenerami personalnymi wyłoniono następujące błędy techniczne (zilustrowane na rys. 4.1):

Rysunek 4.1: Wizualizacja wyłonionych błędów technicznych wykonywanych podczas przysiadu ze sztangą z tyłu

Źródło: opracowanie własne

- Niepełny zakres ruchu (Błąd 1)
- Niesymetryczne chwycenie sztangi (Błąd 2)
- Niepoprawna pozycja głowy (Błąd 3)
- Zbyt mocne wychylenie łokci do tyłu (Błąd 4)
- Zapadanie się sylwetki do przodu (Błąd 5)
- Zapadanie się kolan do środka (Błąd 6)
- Asymetryczna praca bioder (Błąd 7)
- Unoszenie pięt (Błąd 8)
- Zbyt szybkie, niekontrolowane opuszczanie w dół (Błąd 9)

Błąd dotyczący wykonywania niepełnych, zbyt płytkich powtórzeń, a także błąd dotyczący zbyt szybkiej fazy ekscentrycznej w przysiadzie wynikają z trajektorii wykonanego powtórzenia i będą wykrywane na podstawie analizy całej sekwencji klatek nagrania. Natomiast wystąpienie wszystkich pozostałych, wymienionych wyżej błędów da się stwierdzić obserwując jedynie pojedynczą klatkę, dlatego to właśnie te błędy będą etykietowane podczas oznaczania zbioru i to dla nich zostaną wytrenowane modele klasyfikacyjne.

4.2. Zbieranie nagrań treningowych

W celu zapewnienia zbliżonej liczby przykładów dla każdego z wyłonionych błędów technicznych zdecydowano się na nagrywanie materiałów treningowych w sposób zorganizowany - w grupach składających się z kilku osób średnio zaawansowanych treningowo, którzy potrafili w sposób poprawny wykonywać podstawowe ćwiczenia siłowe.

Uczestników poproszono o wykonanie jednej poprawnej serii powtórzeń dla każdego nagrywanego ćwiczenia (oprócz przysiadu ze sztangą z tyłu, którego obejmuje działanie systemu asystenta treningowego w ramach niniejszej pracy, zbierano także nagrania martwego ciągu klasycznego, które mogą posłużyć do rozbudowy systemu w przyszłości). Następnie każda osoba wykonywała jedną serię powtórzeń starając się imitować kolejne błędy techniczne przedstawione w Sekcja 4.1. Każda seria składała się z około 5 wykonanych powtórzeń. Oprócz samej realizacji ćwiczenia na nagraniach rejestrowano także podejścia uczestników pod sztangę, moment rozpoczęcia wykonywania ćwiczenia, oraz moment zakończenia ćwiczenia i odłożenie sztangi z powrotem na stojaki. Te dodatkowe materiały umożliwią dodanie do systemu funkcji wykrywania momentów rozpoczęcia i zakończenia serii.

W sumie zebrano materiały treningowe 22 osób, które w znaczny sposób różnią się od siebie budową ciała oraz techniką wykonywania rejestrowanych ćwiczeń. Przekłada się to na blisko 300 nagrań wideo (dla przysiadu ze sztangą z tyłu), trwających od około 10 do 30 sekund każde. Nagrania rejestrujące ćwiczenia wykonywane przez autora pracy występują w zebranym zbiorze w większej liczbie niż nagrania pozostałych uczestników.

Każdą serię rejestrowano za pomocą trzech kamer umiejscowionych pod różnymi kątami względem osoby wykonującej ćwiczenie:

1. Na przeciwko osoby ćwiczącej
2. Pod kątem około 45° względem osoby ćwiczącej

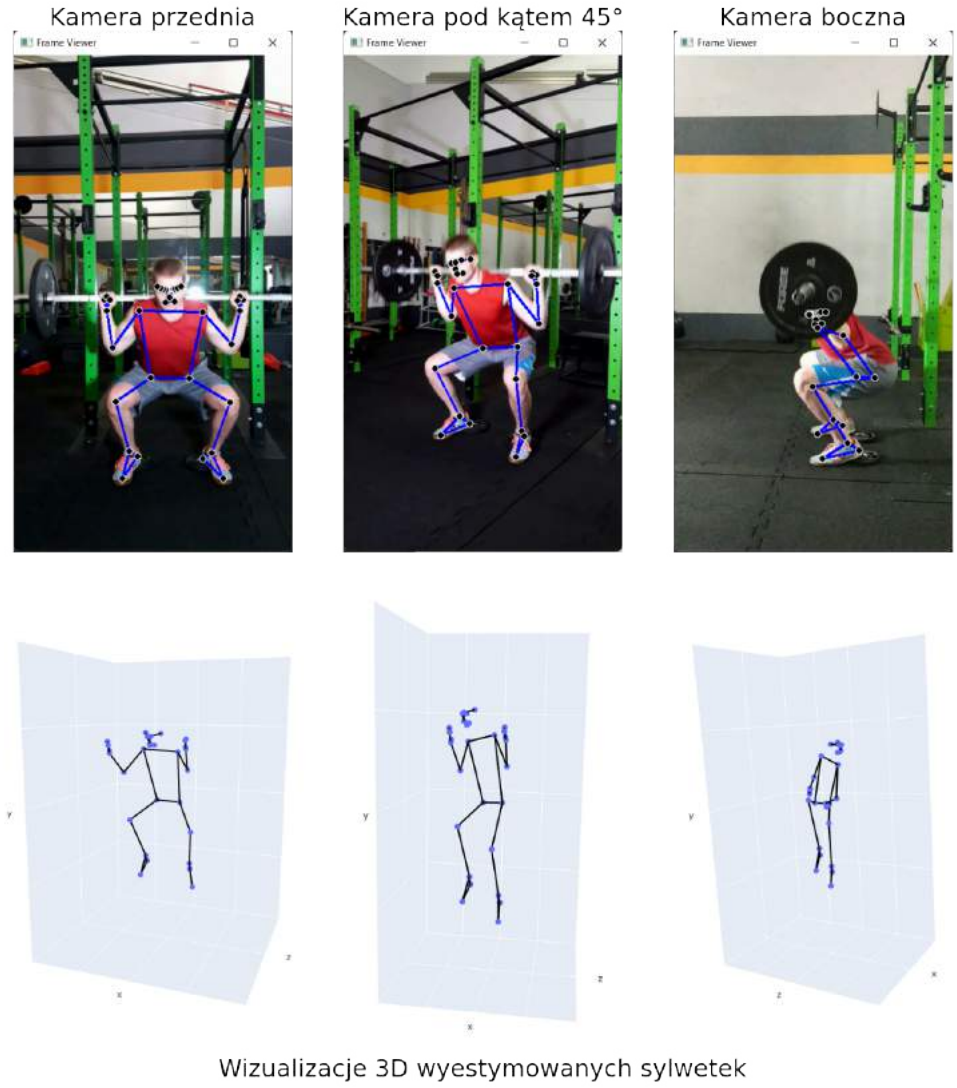
3. Od boku (pod kątem około 90°) osoby ćwiczącej.

Jako kamery zostały wykorzystane aparaty fotograficzne następujących telefonów komórkowych: Samsung Galaxy S7, Samsung Galaxy S6, Xiaomi Redmi Note 5. Każde urządzenie zostało umieszczone na nieruchomym stojaku, na wysokości około 1 metra. Nagrania rejestrowano w rozdzielczości 1080 x 1920 (orientacja pionowa) przy szybkości przechwytywania obrazu wynoszącej 30 klatek na sekundę. Widok z każdej z kamer przedstawiony został na rys. 4.2.

Rysunek 4.2: Widok z trzech kamer wykorzystywanych podczas gromadzenia zbioru danych
Źródło: opracowanie własne

W celu zautomatyzowania i synchronizacji procesów rozpoczynania oraz zatrzymywania nagrań, a także ustrukturyzowania zapisu zarejestrowanych materiałów na dysku, został wykorzystany program napisany w języku Python specjalnie na potrzeby niniejszej pracy. Pozwala on na równoległe uruchomienie i wyłączenie nagrywania trzema kamerami za pomocą 2 klawiszy. Po zakończeniu automatycznie zapisuje materiały wideo we wskazanych miejscach na dysku. Aby umożliwić obsługę aparatów fotograficznych urządzeń mobilnych z poziomu komputera, na którym uruchamiany był program do rejestracji nagrań, wykorzystano oprogramowanie Iriun Webcam. Po jego instalacji na obu urządzeniach końcowych pozwala ono wykorzystać telefon komórkowy jako kamerę internetową. Przesył strumienia wideo odbywa się za pomocą transmisji bezprzewodowej (wspólna sieć Wi-Fi) lub transmisji szeregowej (złącze USB). W tym przypadku korzystano z transmisji przewodowej, aby zapewnić większą stabilność przesyłu i lepszą jakość nagrań.

Ostatecznie w pracach nad systemem asystenta treningowego skorzystano wyłącznie z nagrań pochodzących z kamery ustawionej na przeciwko osoby ćwiczącej. W przypadku nagrań z kamery bocznej, algorytm BlazePose błędnie dokonywał estymacji pozy osoby wykonującej ćwiczenie z powodu przysłonięcia jej głowy przez obciążenie znajdujące się na sztandze oraz przez przysłonięcia kończyn znajdujących w się dalszej odległości od kamery. Z kolei przy nagraniach z kamery ustawionej pod kątem 45° wybrany algorytm miał tendencję do niesymetrycznego szacowania położenia rąk i nóg, pomimo ich identycznej pozycji względem osi przechodzącej przez środek ciała osoby na nagraniu. Błąd ten wynika z niedokładności w estymacji trzeciego wymiaru, jakim jest odległość punktów od kamery. Rysunek 4.3 wizualizuje opisane zjawiska.



Rysunek 4.3: Porównanie wyników estymacji pozy algorytmem BlazePose dla różnych pozycji ustawienia kamery

Źródło: opracowanie własne

4.3. Oznaczanie nagrań treningowych

Przed przystąpieniem do pracy nad systemem pojedyncze klatki z zebranych nagrań treningowych należało oznaczyć odpowiednimi etykietami. Pomimo założenia przyjętego w trakcie gromadzenia zbioru danych, że każde nagranie zawiera imitację tylko jednego błędu technicznego, niektóre klatki zawierają po kilka z nich, co trzeba uwzględnić w procesie ich etykietowania.

Na kolejnych klatkach, zawierających wykonywane przez uczestników powtórzenia, nasilenie danego błędu technicznego znacząco się zmienia. Aby umożliwić rozróżnianie pewności i widoczności błędów podczas etykietowania, wprowadzona została czterostopniowa skala od 0 do 3, gdzie 0 oznacza brak wystąpienia danego błędu, a 3 oznacza ewidentne jego wystąpienie. Przykład zastosowania skali pewności dla klatek pochodzących ze zbioru danych został przedstawiony na rys. 4.4.

Oprócz etykiet mówiących o pewności wystąpienia poszczególnych błędów technicznych, z perspektywy tworzenia systemu asystenta ważna jest również informacja jaki jest stan wykonywanej przez użytkownika serii powtórzeń danego ćwiczenia. Sytuacje, w których system zwracałby użytkownikowi komunikaty o błędzie jeszcze w trakcie jego ustawiania się do pozycji wejściowej danego ćwiczenia lub już po jego zakończeniu, byłyby niedopuszczalne. Co więcej, informacja o stanie wykonywanej serii

Rysunek 4.4: Porównanie pewności wystąpienia błędów technicznych na przykładzie błędu zapadania się kolan do środka

Źródło: opracowanie własne

jest również kluczowa, aby umożliwić zliczanie powtórzeń wykonanych przez użytkownika. Mając to na uwadze, wprowadzony został kolejny rodzaj oznaczenia klatek z nagrań treningowych (zilustrowany na rys. 4.5), który zawiera 4 możliwe wartości:

- Przed rozpoczęciem lub po zakończeniu ćwiczenia
- Pozycja startowa
- Faza opuszczania w dół lub wstawania do góry
- Pozycja dolna (pełny przysiad).

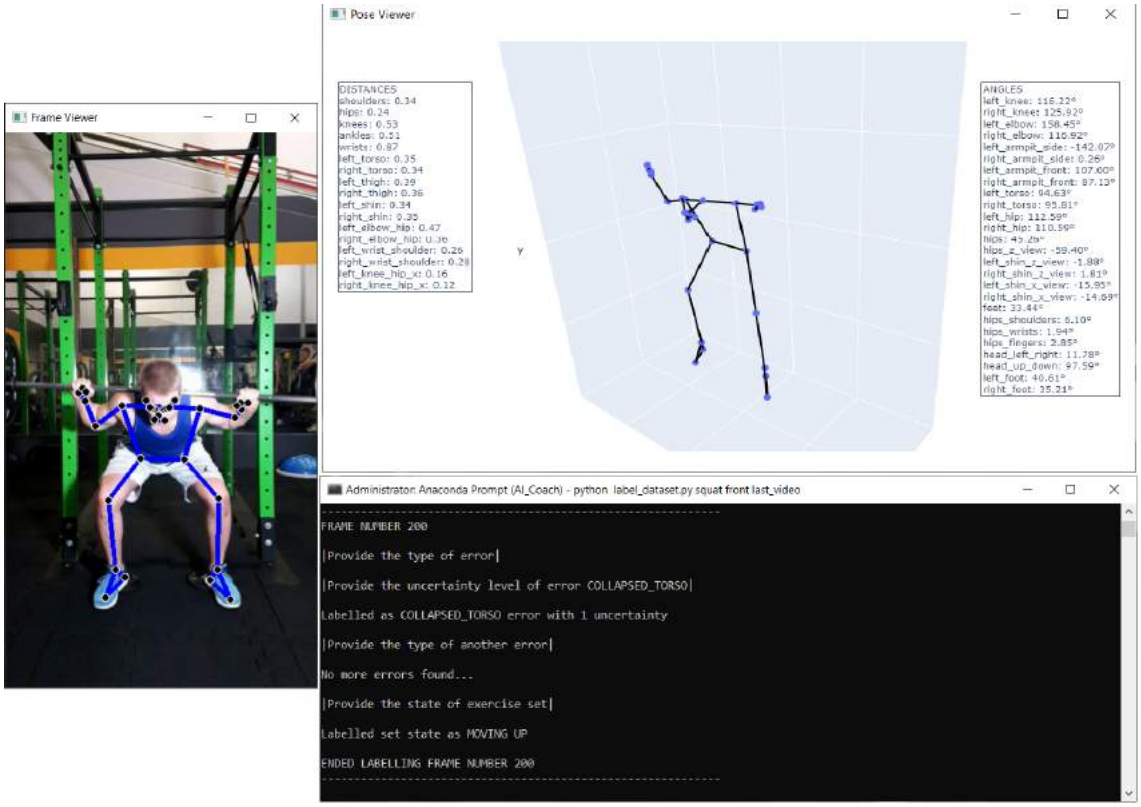
Granice pomiędzy możliwymi wartościami oznaczeń zarówno dla pewności danego błędu, jak i stanu wykonywanej serii są umowne. Przy podejmowaniu decyzji w dyskusyjnych przypadkach kierowano się doświadczeniem treningowym autora pracy.

Rysunek 4.5: Porównanie stanów serii powtórzeń w przysiadzie ze sztangą z tyłu

Źródło: opracowanie własne

Aby spełnić wszystkie wymienione założenia, które dotyczą oznaczania zgromadzonych nagrań treningowych, zdecydowano się na napisanie dedykowanego programu w języku Python, który automatyzuje proces etykietowania. Interfejs programu został przedstawiony na rys. 4.6. Składa się on z trzech oddzielnych okien:

- Okna wyświetlającego aktualną klatkę nagrania wraz z naniesionym położeniem szkieletu oszacowanym przez algorytm BlazePose
- Okna z interaktywnym wykresem wyświetlającym położenie oszacowanego szkieletu w przestrzeni trójwymiarowej
- Okna konsoli, gdzie znajdują się informacje o numerze nagrania oraz numerze aktualnej klatki, a także o dokonanych przez użytkownika oznaczeniach.



Rysunek 4.6: Widok interfejsu dedykowanego programu do oznaczania nagrań treningowych

Źródło: opracowanie własne

Sterowanie oznaczeniami jest wykonywane z poziomu klawiatury. Po wyświetleniu nowej klatki nagrania użytkownik iteracyjnie ją oznacza błędami technicznymi, które są powiązane z predefiniowanymi klawiszami. Po wybraniu konkretnego błędu należy także podać jego pewność w opisanej wcześniej skali. Następnie użytkownik stwierdza wystąpienie kolejnego błędu lub przechodzi do ostatniego etapu, który dotyczy wyznaczenia stanu serii powtórzeń. Postanowiono oznaczać co piątą klatkę każdego nagrania, ponieważ zmiana w położeniu punktów kluczowych pomiędzy pojedynczymi klatkami jest nieznaczna. Pomimo pomijania oznaczeń klatek pośrednich, algorytm BlazePose dokonywał dla nich estymacji, ponieważ jego predykcje w trybie użytkowania go na nagraniach są zależne od wyników uzyskanych na klatkach poprzedzających.

Wyniki estymacji oraz oznaczenia wykonane przez użytkownika po każdej klatce są dodawane jako nowy wiersz do ramki danych będącej w pamięci programu. Po skończeniu oznaczania całego nagrania dane te są zapisywane do pliku w formacie .csv pod wskazanym adresem w pamięci urządzenia.

W oknie z wizualizacją 3D sylwetki osoby wykonującej ćwiczenie wyświetlane są także wartości różnych kątów oraz odległości pomiędzy punktami kluczowymi. Podczas oznaczania zbioru monitorowano te wartości oraz z czasem dodawano do nich nowe. Ułatwiło to i znacznie przyspieszyło procesy inżynierii oraz selekcji cech wykonane podczas projektowania modeli klasyfikacyjnych. Pozwoliło to także poznać najczęstsze sytuacje, w których algorytm BlazePose dokonuje błędnych estymacji.

Pomimo maksymalnej automatyzacji proces ręcznego oznaczania nagrań (a także ich gromadzenia) jest bardzo czasochłonny. Dodatkowym czynnikiem negatywnie wpływającym jest rozbudowany system etykietowania zaproponowany na potrzeby niniejszej pracy. Łącznie oznaczono około 20 tysięcy klatek, co przy średnim czasie poświęconym na pojedynczą klatkę wynoszącym kilkanaście sekund, przekłada się na blisko 100 godzin pracy.

W celu zwiększenia liczebności zbioru danych zastosowano augmentację na wszystkich oznaczonych klatkach z nagrań treningowych. Wykonano transformację w postaci odbicia lustrzanego, co podwoiło liczbę przykładów uczących. Zrezygnowano z innych, bardziej zaawansowanych sposobów na przekształcenia obrazu, ponieważ mogłyby one powodować zmniejszenie dokładności estymacji punktów kluczowych osoby wykonującej ćwiczenie, co w konsekwencji prowadziłoby do pogorszenia skuteczności uzyskiwanej przez projektowany system.

5. SYSTEM ASYSTENTA TRENINGU SIŁOWEGO

Docelowy system asystenta treningowego swoim działaniem powinien obejmować wiele ćwiczeń siłowych. Ze względu na różną specyfikę każdego ćwiczenia oraz różnice w popełnianych przy nich błędach technicznych powinien być on podzielony na odrębne moduły, z których każdy obsługiwałby pojedyncze ćwiczenie. Niniejszy rozdział opisuje założenia oraz proces tworzenia modułu treningowego, który ma za zadanie wspierać osoby początkujące podczas wykonywania przysiadu ze sztangą z tyłu.

5.1. Założenia

Jak wspomniano we wstępie do rozdziału, system asystenta powinien być podzielony na moduły, obsługujące poszczególne ćwiczenia siłowe. Każdy moduł jako dane wejściowe powinien przyjmować położenia punktów kluczowych osoby znajdującej się na nagraniu, które są wynikiem obliczeń wybranego algorytmu do estymacji pozy. Moduł, na podstawie analizy ułożenia szkieletu na kolejnych klatkach nagrania, ma stwierdzać fakt popełnienia błędów technicznych przez osobę ćwiczącą. Podzbiór błędów obsługiwanych przez moduł powinien zawierać takie błędy, których wykonywanie jest niewskazane bez względu na indywidualne parametry osoby ćwiczącej. Możliwe jest także dodanie obsługi spersonalizowanych błędów technicznych, jednak ich wykrywanie powinno być dodatkowo uzależnione od różnych indywidualnych parametrów podawanych przez użytkownika systemu. Działanie systemu, będącego tematem niniejszej pracy, jest ograniczone wyłącznie do pierwszego rodzaju błędów.

Oprócz obsługi błędów, wynikających z nieprawidłowego ułożenia szkieletu w danym momencie, system powinien być w stanie określać, jaki jest aktualny stan wykonywanej przez użytkownika serii powtórzeń. Taka informacja umożliwi detekcję momentów rozpoczęcia części właściwej ćwiczenia oraz jego zakończenia. Pozwoli to uniknąć sytuacji, w których system zwracałby użytkownikowi informacje o popełnieniu błędu w momentach, w których on dopiero ustawiałby się do pozycji startowej danego ćwiczenia lub tuż po jego zakończeniu. Takie rozwiązanie umożliwi także wykrywanie błędów, które wynikają z trajektorii całego ruchu i jej parametrów kinematycznych (m.in. prędkość ruchu i jego zakres). Informacja o stanie wykonywanej serii pozwoli również na dodanie innych funkcjonalności do systemu, jak chociażby licznika wykonanych powtórzeń.

W ogólności na jednej klatce nagrania (w konsekwencji także w trakcie pojedynczego powtórzenia) może wystąpić więcej niż jeden błąd techniczny popełniony przez użytkownika. Implikuje to zdefiniowanie problemu wykrywania błędów jako klasyfikacji wieloetykietowej, gdzie liczba możliwych klas jest uzależniona od liczby wyłonionych błędów technicznych obsługiwanych przez konkretny moduł systemu asystenta (z pominięciem dodatkowych błędów wynikających z trajektorii całego ruchu). Skuteczność wykrywania błędów przez system powinna być mierzona na poziomie pojedynczego powtórzenia, a nie na poziomie pojedynczych klatek nagrania.

Aplikacja, do której zostanie wdrożony system asystenta treningowego, powinna działać w czasie rzeczywistym na urządzeniu końcowym, co w sposób naturalny wpływa na możliwości jego rozbudowy i wybór algorytmów do jego implementacji. Dodatkowym czynnikiem utrudniającym spełnienie tego wymagania jest czas konieczny do przetworzenia klatki przez algorytm estymujący pozę. Oczywiście szybkość działania całego sytemu jest uzależniona od specyfikacji urządzenia, na którym jest on uruchamiany. W ramach niniejszej pracy założenie działania w czasie rzeczywistym zdefiniowano jako zdolność do przetwarzania minimum 15 klatek na sekundę na urządzeniu, którego szczegółowa specyfikacja została podana w Tabela 5.1.

Tabela 5.1: Specyfikacja urządzenia wykorzystanego do testów systemu asystenta treningowego

Komponent	Specyfikacja
Procesor	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70 GHz
RAM	16 GB DDR4 3200 MHz

5.2. Detekcja stanu wykonywanego ćwiczenia

Problem detekcji stanu zdefiniowano jako zadanie klasyfikacji wieloklasowej, gdzie liczba docelowych klas jest równa liczbie różnych wartości, którymi oznaczano klatki nagrań treningowych podczas przygotowywania zbioru uczącego (szczegółowo przedstawione w Sekcja 4.3). Liczebność przykładów oznaczonych poszczególnymi etykietami nie jest idealnie zbalansowana - w Tabela 5.2 znajduje się szczegółowy udział klas w zebranym zbiorze. Próbki z klasy ’0’ zajmują jedynie 8% całego zbioru, ponieważ nagrania rozpoczynano z opóźnieniem i kończono z wyprzedzeniem, aby zaoszczędzić czas podczas etykietowania.

Tabela 5.2: Udział poszczególnych klas, oznaczających stany wykonywanego ćwiczenia, w zbiorze danych

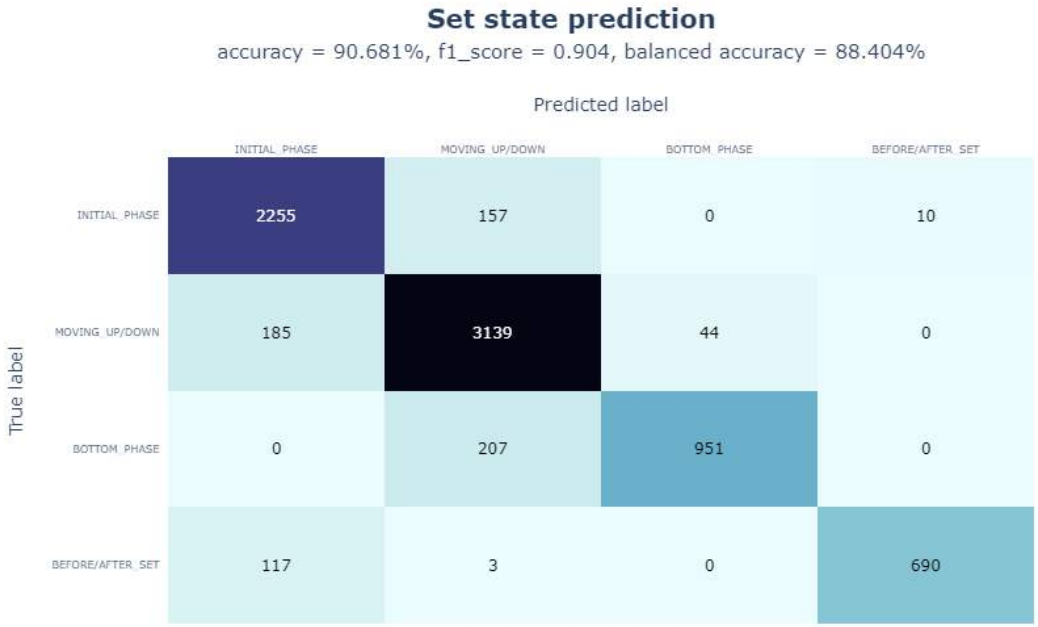
Klasa	Nazwa	Udział w zbiorze danych
0	Przed rozpoczęciem lub po zakończeniu ćwiczenia	8%
1	Pozycja startowa	32%
2	Faza opuszczania w dół lub wstawania do góry	44%
3	Pozycja dolna (pełny przysiad)	16%

Do budowy klasyfikatora wykorzystano algorytm drzew decyzyjnych wzmacnianych gradientowo, którego efektywna implementacja jest dostępna w wielu językach programowania za pośrednictwem pakietu XGBoost [23]. Jest to metoda, polegająca na sekwencyjnym dodawaniu kolejnych estymatorów do zespołu (w tym przypadku drzew decyzyjnych), gdzie każdy kolejny z nich jest dopasowywany do błędu resztowego popełnianego przez poprzedników. Wspomniana implementacja algorytmu zawiera kilka dodatkowych elementów regularyzacyjnych, hamujących rozrost drzew, dodanych w celu kontrolowania złożoności modelu i zapobiegania jego przetrenowaniu. Zdecydowano się na wybór tego algorytmu z uwagi na szybkość uzyskiwania predykcji oraz istnienie wielu mechanizmów, przeciwdziałających problemom występującym w posiadanym zbiorze danych.

Jako cechy wejściowe do nauki modelu wyłoniono zestaw kilkudziesięciu kątów pomiędzy różnymi częściami ciała oraz odległości pomiędzy parami różnych punktów kluczowych wyestymowanej sylwetki. Niektóre kąty są wyliczane z różnych perspektyw w przestrzeni trójwymiarowej, natomiast niektóre odległości tylko w wybranych osiach układu współrzędnych.

Do optymalizacji hiperparametrów wybranego modelu, dotyczących m.in. liczby estymatorów, maksymalnej głębokości drzew czy szczegółów mechanizmów regularyzacyjnych, wykorzystano pakiet Optuna [24], który jest dostępny w języku Python za pomocą prostego w użyciu API. Biblioteka zawiera efektywną implementację metody optymalizacji bayesowskiej, specjalnie dostosowaną do potrzeb tuningu hiperparametrów modeli predykcyjnych. Działanie metody polega na próbach oszacowania początkowo nieznanej funkcji, której dziedziną jest podana przestrzeń hiperparametrów, a przeciwdziedziną przyjęta metryka błędu. Po znalezieniu jej przybliżenia zwracany taki zestaw wartości szukanych hiperparametrów, który będzie minimalizował błąd predykcji. Zdecydowano się na wybór tej metody, kosztem chociażby metod przeszukiwania siatki czy metod losowych, z uwagi na jej większą efektywność czasową.

Finalna wersja klasyfikatora osiągnęła ponad 90% skuteczności na zbiorze walidacyjnym, zawierającym przykłady z 5 osobami, które nie wystąpiły w zbiorze treningowym. Rysunek 5.1 przedstawia macierz błędów, na której można zauważyć klasy najczęściej ze sobą mylone. Przejrzano część błędnie sklasyfikowanych klatek i wszystkie z nich przedstawiały sytuacje sporne z pogranicza obu stanów, które klasyfikator ze sobą pomylił.



Rysunek 5.1: Macierz błędów klasyfikatora stanu wykonywanego ćwiczenia

Źródło: opracowanie własne

Dokonano także wglądu w najczęstsze cechy wybierane przez model podczas generowania kolejnych drzew decyzyjnych. Najbardziej wartościowymi zmiennymi wejściowymi okazały się być następujące cechy:

- Średni kąt w stawach kolanowych (płaszczyzna YZ)
- Średni kąt pomiędzy kośćmi udowymi a torsem (płaszczyzna YZ)
- Odległość pomiędzy stawami skokowymi (oś X)
- Kąt pomiędzy kośćmi udowymi (płaszczyzna XY)
- Odległość pomiędzy stawami skokowymi (oś Z).

Podczas uruchomienia modelu na rzeczywistych nagraniach okazało się, że klasyfikator często błędnie zmienia stany z klasy "1" na "0" oraz z klasy "1" na "2" w trakcie drobnych ruchów wykonywanych przez osoby ćwiczące podczas stania w pozycji startowej. Problem ten wynika ze zbyt drobiazgowego oznaczenia stanów w zbiorze danych w stosunku do dokładności algorytmu estymującego pozę. Przykładowo, początkowe klatki powtórzenia, na których osoba wykonująca ćwiczenie miała tylko minimalnie ugięte kolana, oznaczano już etykietą klasy "2". Niestety niedokładność algorytmu BlazePose nie pozwala dostrzec tych drobnych różnic w wartościach obliczonych kątów i odległości. Aby temu zaradzić, zwiększono wartości minimalnych progów prawdopodobieństwa dla klas "0" oraz "2" koniecznych, aby model przewidział którąś z wymienionych etykiet. Jednocześnie zmniejszono jednakowy próg dla

klasy "1". Dodatkowym mechanizmem, zabezpieczającym przed nadmiarowymi zmianami stanów, jest implementacja pamięci, która przechowuje N ostatnich predykcji dokonanych przez klasyfikator. Jako prawdziwy, aktualny stan przyjmuje się klasę najczęściej występującą w tym zbiorze. Rozwiązanie to wprowadza nieznaczne opóźnienie, jednak jest ono prawie niezauważalne. Wartość parametru N została dobrana empirycznie.

Na podstawie informacji o aktualnym stanie system może przewidzieć różne zdarzenia. Jako rozpoczęcie części właściwej ćwiczenia uznaje się moment pierwszego wystąpienia klasy "2" lub "3". Z kolei zakończenie ćwiczenia (jeśli wcześniej stwierdzono jego start) zgłaszane jest w chwili pojawienia się klasy "0". Zmiana stanu z klasy "1" na "2" oznacza rozpoczęcie nowego powtórzenia. Analogicznie, przejście z klasy "2" na "1" oznacza jego zakończenie, co daje sygnał do inkrementacji licznika wykonanych przez użytkownika powtórzeń.

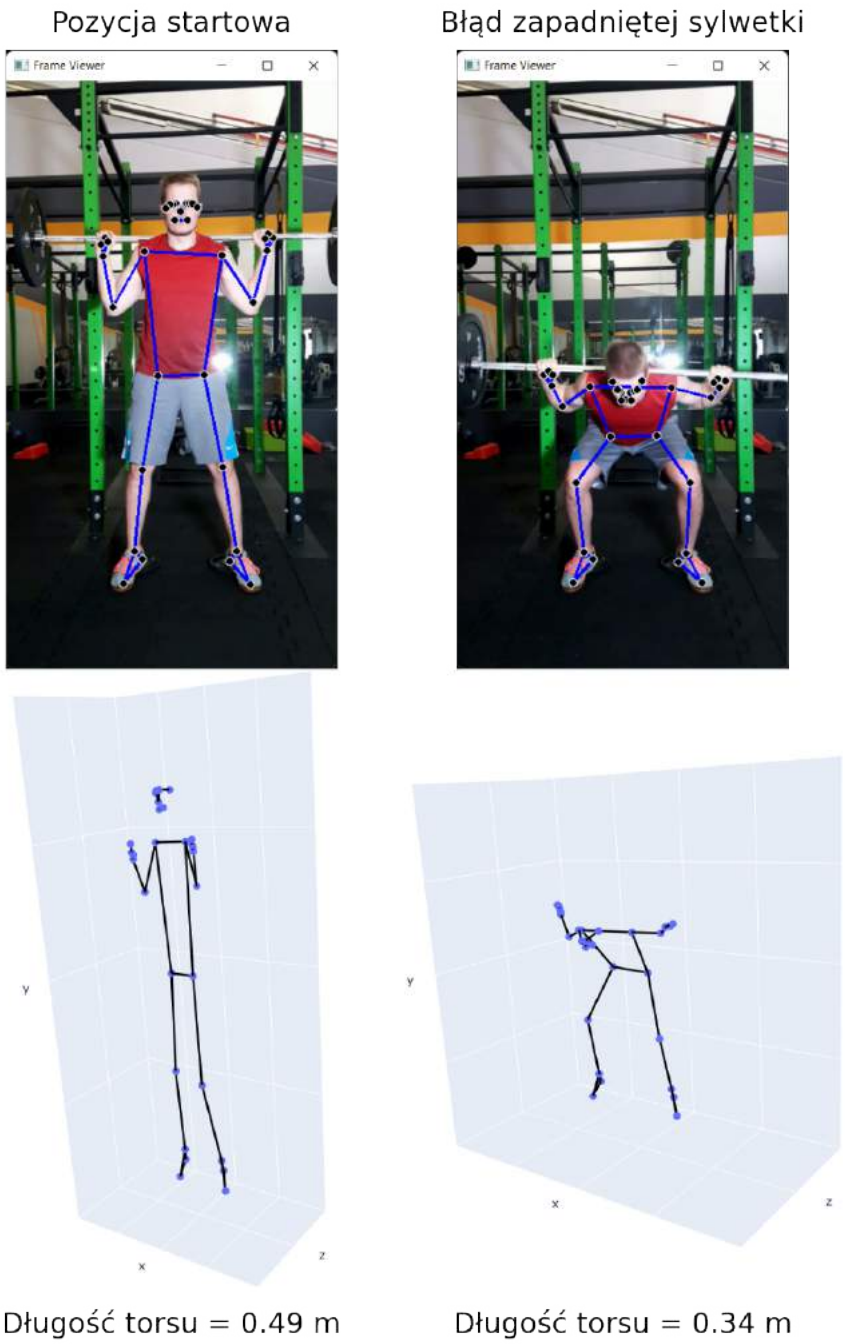
5.3. Wykrywanie błędów technicznych

Wystąpienie Błędu 1 oraz Błędu 9 da się stwierdzić na podstawie informacji o stanie wykonywanego ćwiczenia. Podczas oznaczania zbioru granicę pomiędzy klasami "2" i "3" wyznaczono jako umowną barierę różniącą pełny, poprawny przysiad od przysiadu zbyt płytkiego. Umożliwia to uzależnienie detekcji Błędu 1 od faktu wystąpienia klasy "3" w trakcie wykonywania pojedynczego powtórzenia. Jeśli klasa "3" nie wystąpiła, zwracana jest informacja o wykonaniu niepełnego przysiadu. Znając dodatkowo parametr w postaci liczby klatek na sekundę w analizowanym nagraniu (lub wartość tego parametru dla kamery w przypadku treningu na żywo), można obliczyć czas przejścia z pozycji startowej do pozycji dolnej przysiadu. Jeśli ten czas będzie mniejszy niż minimalny, wymagany próg, to oznacza, że faza ekscentryczna przebiegła w sposób niekontrolowany. W tej sytuacji system zwróci informację o wystąpieniu Błędu 9.

Detekcja pozostałych błędów technicznych jest tożsama z problemem klasyfikacji wieloetykietowej. Zamiast jednego modelu zdecydowano się na wyuczenie 7 klasyfikatorów binarnych, z których każdy przewiduje wystąpienie pojedynczego błędu. Takie podejście umożliwia wybór uszczuplonego zestawu indywidualnych cech wejściowych dla każdego klasyfikatora, co, przy stosunkowo niewielkim rozmiarze zbioru treningowego, zmniejsza ryzyko przeuczenia poszczególnych modeli. Do budowy wspomnianych klasyfikatorów zdecydowano się po raz kolejny wykorzystać algorytm drzew decyzyjnych wzmacnianych gradientowo z biblioteki XGBoost.

Znacznym utrudnieniem w trakcie projektowania poszczególnych modeli jest fakt silnego niezbalansowania klas w zbiorze treningowym. W zależności od typu błędu klasa pozytywna (klatki przedstawiające błąd techniczny) stanowi jedynie kilka procent wszystkich przykładów. Dodatkowo, co opisano w Sekcja 4.3, tylko część z tych przykładów przedstawia błędy ewidentne o najwyższej skali pewności. Aby zniwelować problem niezbalansowania klas, wykorzystano mechanizm zaimplementowany w algorytmie XGBoost, który polega na zwiększeniu wag dla błędów popełnianych przez model na próbkach pozytywnych. W konsekwencji model koryguje je w większym stopniu podczas dodawania kolejnych estymatorów do zespołu.

W celu uwzględnienia skali pewności stosowanej podczas etykietowania, wprowadzono dodatkowe ważenie próbek pozytywnych. Przykłady z błędami pewniejszymi otrzymały większe wartości wag, przez co miały większy wpływ na proces dopasowywania modelu. Część z projektowanych klasyfikatorów uzyskiwała lepsze wyniki na zbiorze testowym po usunięciu najmniej pewnych przykładów pozytywnych ze zbioru uczącego, co po raz kolejny świadczy o niezdolności algorytmu BlazePose do odwzorowywania niewielkich zmian położenia sylwetki na obrazie w zwracanych estymacjach. W konsekwencji, w trakcie treningu tych modeli, wyrzucono te przykłady ze zbioru uczącego.



Rysunek 5.2: Wizualizacja zjawiska zmiany proporcji estymowanej sylwetki przez algorytm BlazePose podczas wystąpienia błędu technicznego

Źródło: opracowanie własne

Do zestawów zmiennych wejściowych niektórych klasyfikatorów dodano także cechy, będące różnicami pomiędzy aktualną długością a długością w trakcie pozycji startowej tych samych części ciała. Podczas oznaczania zbioru zauważono tendencję algorytmu BlazePose do zmiany proporcji estymowanej sylwetki przy wystąpieniu różnych błędów technicznych. Rysunek 5.2 wizualizuje opisane zjawisko na przykładzie błędu zapadniętej sylwetki. Podczas pochylenia osoby wykonującej ćwiczenie algorytm w wyniku estymacji skrócił znacznie długość jej torsu, zamiast zmniejszyć kąt pomiędzy kośćmi udowymi a torsem. Błąd ten wynika z niedokładnie oszacowanych odległości punktów kluczowych od kamery. Dodanie wspomnianych cech wejściowych sprawia, że klasyfikatory są w stanie się uodpornić na tego

typu błędy dokonywane przez algorytm estymujący pozę. Zastosowanie takiego rozwiązania skutkuje koniecznością obliczenia różnych parametrów początkowych na sylwetce, znajdującej się w pozycji startowej, co uniemożliwia wykrywanie niektórych błędów przed pierwszym wystąpieniem stanu "1" (dotyczy to przyciętych nagrań, które rozpoczynają się już w momencie wykonywania powtórzeń przez osobę ćwiczącą).

Do tuningu hiperparametrów wszystkich klasyfikatorów binarnych po raz kolejny wybrano metodę optymalizacji bayesowskiej z biblioteki Optuna. Jednak tym razem zdecydowano się wykorzystać optymalizację wielokryterialną, w której maksymalizowano wskaźniki precyzji i pełności uzyskiwane przez poszczególne modele na zbiorze walidacyjnym. Precyzja stanowi miarę dokładności predykcji pozytywnych. Z kolei pełność to odsetek przykładów pozytywnych, które zostały prawidłowo rozpoznane przez klasyfikator. Z powodu zastosowania dwóch kryteriów optymalizacyjnych wybór finalnych wartości hiperparametrów polegał na znalezieniu kompromisu pomiędzy wspomnianymi metrykami. Aby zapobiec sytuacjom, w których system fałszywie informowałby o wystąpieniu błędu, w większej mierze kierowano się maksymalizacją precyzji konkretnego klasyfikatora kosztem potencjalnie niewykrytych błędów. W zależności od złożoności danego błędu, finalne wersje modeli uzyskiwały precyzję w przedziale (0.8; 0.95) przy pełności w granicach (0.55; 0.9) na przykładach walidacyjnych z 5 nowymi osobami ćwiczącymi.

Podobnie jak przy detekcji stanu wykonywanego ćwiczenia, tutaj także zastosowano mechanizmy pamięci, aby uchronić system przed zwracaniem informacji o fałszywych błędach z powodu pojedynczych źle sklasyfikowanych klatek. Każda taka pamięć gromadzi N poprzednich predykcji dokonanych przez konkretny klasyfikator. Aby system zwrócił informację o danym błędzie, wymagane jest co najmniej M predykcji pozytywnych (gdzie $M \leq N$). Wartości parametrów N oraz M dobrano w sposób doświadczalny indywidualnie dla każdego rodzaju błędu technicznego.

Podczas wstępnych testów na rzeczywistych nagraniach zauważono, że system podczas wystąpienia jednego błędu często zwracał również fałszywe informacje o innym błędzie. Przykładowo system często informował o zbyt mocnym wychyleniu łokci do tyłu w trakcie błędu zapadniętej sylwetki do przodu. Problem ten wynikał z korelacji niektórych zmiennych wejściowych, wykorzystanych do predykcji fałszywie zgłaszanego błędu, na klatkach przedstawiających zarówno pierwszy, jak i drugi błąd. Zdecydowano się nie zmuszać modeli do nauki tych zależności poprzez dodawanie kolejnych cech, ponieważ najpewniej zakończyłoby się to niepowodzeniem z uwagi na ograniczony rozmiar zbioru treningowego. Zamiast tego, dla wcześniej zdiagnozowanych par błędów (błąd prawdziwy i fałszywy), wprowadzono rozwiązanie w postaci bufora. Po stwierdzeniu błędu fałszywego jego zwrócenie przez system jest zamrażane na czas około 1 sekundy. Jeśli w tym okresie zostanie wykryty błąd prawdziwy, to informacja o błędzie fałszywym jest usuwana. W przeciwnym razie, błąd ten jest zwracany z lekkim opóźnieniem.

Rozwiązanie to nie jest idealne, ponieważ teoretycznie niektóre z obsłużonych w ten sposób par błędów mogłyby się rzeczywiście pojawić. Jednak z punktu widzenia użytkownika, popełniającego oba te błędy jednocześnie, w dłuższej perspektywie nie ma to większego znaczenia. Jeśli w trakcie pierwszego powtórzenia popełni on oba błędy, system zwróci informację tylko o wystąpieniu pierwszego z nich. Osoba ćwicząca, po zauważeniu informacji o błędzie (oraz usłyszeniu głosowej wskazówki, mówiącej jak dany błąd poprawić), będzie starała się go wyeliminować. Jeśli się jej powiedzie i w następnym powtórzeniu popełni już tylko drugi błąd, to system tym razem już ją o nim poinformuje.

Detekcja każdego z obsługiwanych błędów technicznych jest blokowana na czas trwania aktualnego powtórzenia po jednokrotnym stwierdzeniu ich wystąpienia. Zapobiega to sytuacjom, w których strumień informacji zwracanych użytkownikowi byłby zablokowany przez wielokrotne sygnały o tym samym błędzie. Po wystąpieniu zdarzenia, oznaczającego rozpoczęcie nowego powtórzenia, detekcje wszystkich błędów są na nowo odblokowywane.

6. APLIKACJA

Gotowy system asystenta treningowego powinien być częścią aplikacji komputerowej, umożliwiającej mu swobodną interakcję z użytkownikiem końcowym. W niniejszym rozdziale przedstawiono prototyp takiej aplikacji oraz sposób jej integracji z systemem.

6.1. Założenia

Głównym zadaniem aplikacji powinno być umożliwienie osobom ćwiczącym intuicyjnego korzystania z funkcjonalności oferowanych przez system asystenta treningowego. Aplikacja powinna zawierać prosty w obsłudze interfejs graficzny. Użytkownik powinien mieć także możliwość zapoznania się z instrukcją użytkowania systemu.

W przypadku systemów, obsługujących większą liczbę ćwiczeń siłowych, powinna być także zapewniona możliwość wyboru ćwiczenia. Oprócz treningów na żywo aplikacja mogłaby także umożliwiać uruchomienie systemu na nagraniach, znajdujących się w pamięci urządzenia. Dzięki temu użytkownik mógłby analizować i porównywać ze sobą pod kątem popełnionych błędów różne serie wykonane przez niego w przeszłości.

Aplikacja powinna zapewniać obsługę kamery urządzenia, na którym zostanie uruchomiona, oraz możliwość wczytywania nagrań zapisanych w jego pamięci. Ponadto, aplikacja ma obowiązek zagwarantowania transferu wczytywanych klatek do algorytmu estymującego pozę, a następnie dostarczenie wyników jego obliczeń do systemu asystenta treningowego.

Na podstawie informacji otrzymywanych od systemu asystenta treningowego, aplikacja powinna sygnalizować błędy techniczne popełnione przez użytkownika oraz wystąpienia różnych zdarzeń, takich jak rozpoczęcie oraz zakończenia ćwiczenia czy wykonanie kolejnego powtórzenia. W przypadku treningu na żywo, ważnym jest, aby wymienione komunikaty były także odtwarzane głosowo. Dzięki temu użytkownik, będąc oddalonym od urządzenia, mógłby się z nimi bez problemu zapoznać. W przypadku informacji o wykrytym błędzie aplikacja powinna odtwarzać krótkie głosowe wskazówki, dzięki którym użytkownik mógłby je jak najszybciej wyeliminować.

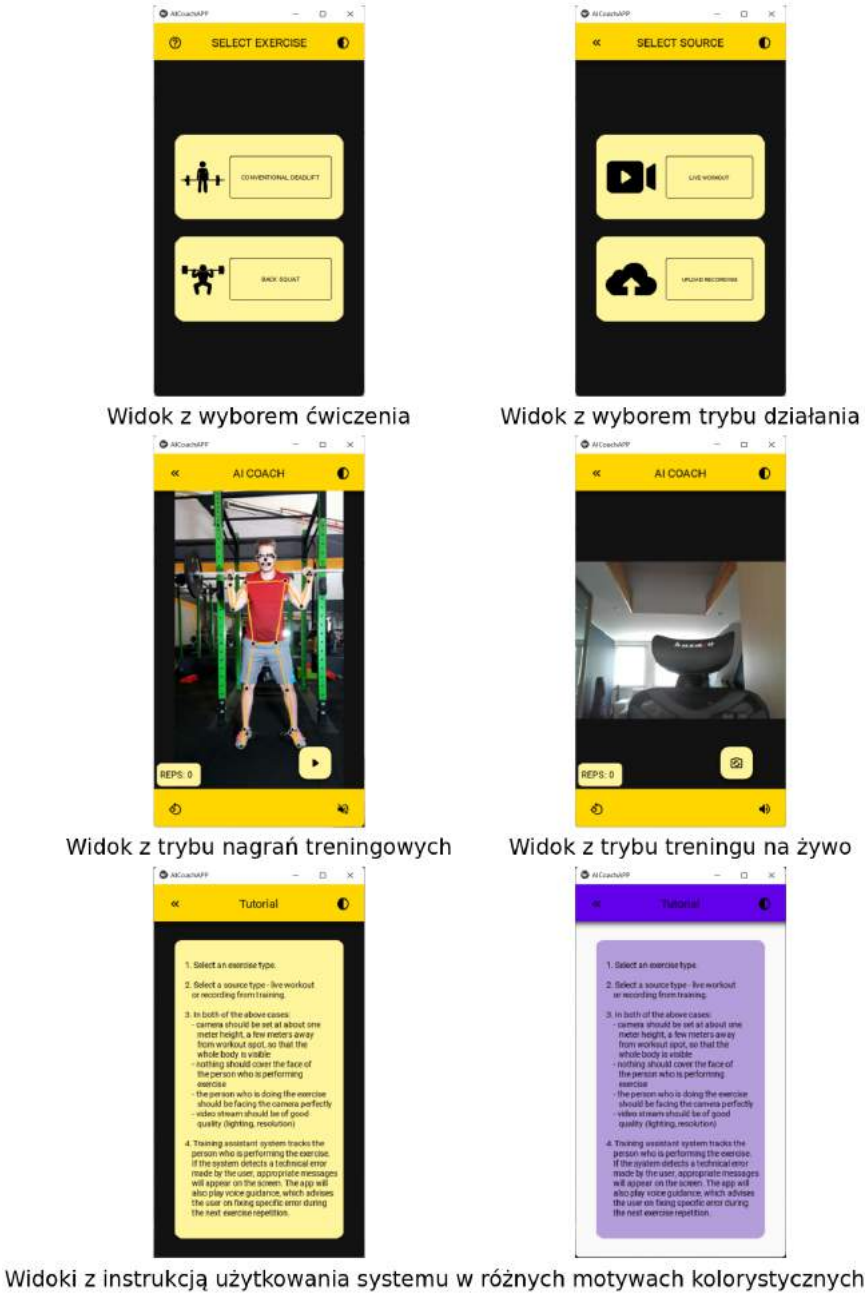
6.2. Implementacja

Do implementacji aplikacji zdecydowano się wykorzystać platformę Kivy, która jest dostępna z poziomu języka Python. Kivy jest projektem społecznościowym o otwartym kodzie źródłowym. Umożliwia tworzenie aplikacji okienkowych na wiele platform docelowych - urządzenia mobilne (Android, iOS) oraz różne systemy operacyjne (Windows, Linux, macOS). Powodami wyboru biblioteki Kivy na potrzeby niniejszej pracy były następujące czynniki:

- bezproblemowa integracja z systemem asystenta treningowego, który również został zaimplementowany w języku Python
- możliwość potencjalnego zbudowania aplikacji na różne platformy końcowe niskim kosztem.

Do przygotowania szaty graficznej aplikacji wykorzystano wewnątrzplatformowy język projektowania o nazwie Kivy Design Language. Znacząco ułatwia on wizualne projektowanie interfejsu aplikacji oraz zapewnia jego separację od logiki jej działania. Skorzystano także z rozszerzenia o nazwie KivyMD (Kivy Material Design), które jest zbiorem dodatkowych widżetów. Rozszerzenie umożliwia ich wykorzystanie wewnątrz interfejsu graficznego aplikacji tworzonej za pomocą Kivy.

Na interfejs graficzny napisanej aplikacji składają się następujące widoki (zilustrowane na rys. 6.1):



Widok z wyborem ćwiczenia

Widok z wyborem trybu działania

Widok z trybu nagrań treningowych

Widok z trybu treningu na żywo

Widoki z instrukcją użytkowania systemu w różnych motywach kolorystycznych

Rysunek 6.1: Widoki składające się na interfejs graficzny aplikacji

Źródło: opracowanie własne

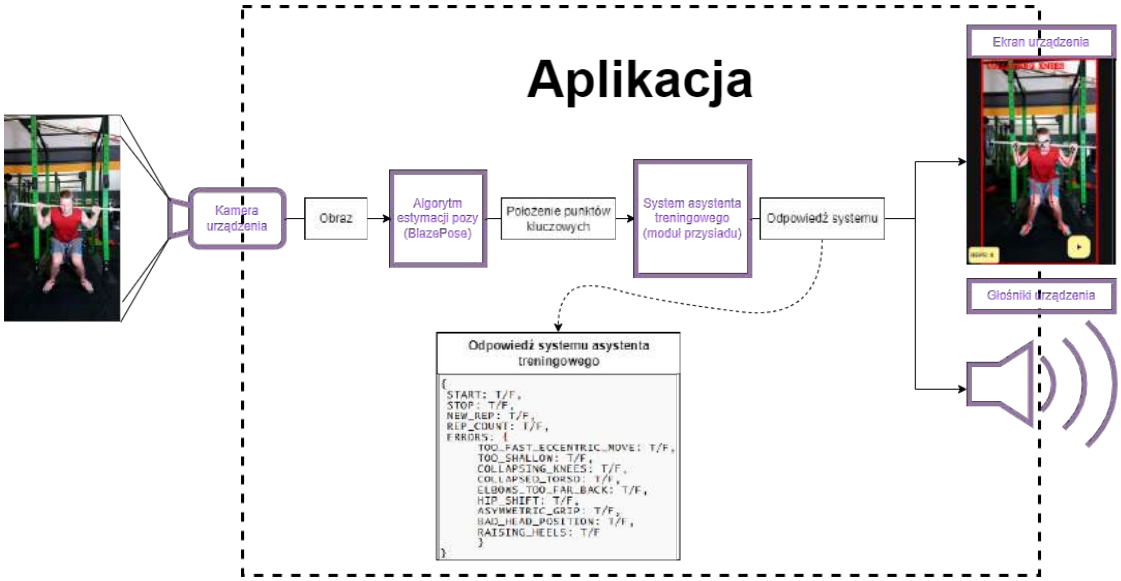
- Widok początkowy aplikacji, który umożliwia użytkownikowi wybór ćwiczenia siłowego. Z powodu ograniczenia działania systemu do jednego ćwiczenia, istnieje możliwość wyboru jedynie przysiadu ze sztangą z tyłu.
- Widok z wyborem trybu działania, który pojawia się po zaznaczeniu ćwiczenia. Użytkownik może wybrać trening na żywo lub wczytać przygotowane nagranie treningowe z pamięci urządzenia.
- Widok z trybu nagrań treningowych, który pojawia się po zaznaczeniu odpowiadającego mu przycisku w poprzednim widoku. Składa się z ekranu wyświetlającego klatki nagrania oraz przycisku zatrzymującego i wznawiającego jego odtwarzanie.

- Widok z trybu treningu na żywo - prowadzi do niego analogiczny przycisk, znajdujący się w widoku z wyborem trybu działania. Również zawiera ekran, który wyświetla klatki przechwycone z kamery urządzenia. Zamiast przycisku startu/stopu znajduje się tutaj przycisk, umożliwiający zmianę wykorzystywanej kamery (w przypadku, gdy urządzenie jest wyposażone w więcej niż jedną kamerę).
- Widok z instrukcją użytkowania aplikacji, do którego można się dostać, wciskając ikonę znaku zapytania, która znajduje się na panelu górnym widoku początkowego aplikacji. Instrukcja opisuje sposób korzystania z interfejsu aplikacji, poprawne przygotowanie kamery i miejsca treningowego, a także skrótowo zasadę działania systemu treningowego.

Dodatkowo interfejs graficzny aplikacji dostępny jest w dwóch wersjach kolorystycznych. Zmiany motywu można dokonać wciskając ikonę, która znajduje się po prawej stronie panelu górnego na każdym z opisanych powyżej widoków. Do nawigacji pomiędzy widokami i powrotu do widoku początkowego służą przyciski, które znajdują się po lewej stronie na panelach górnych. Widoki treningowe (tryb nagrań lub tryb na żywo) zawierają dodatkowy panel dolny. Po lewej stronie panelu znajduje się przycisk, umożliwiający rotację klatek na ekranie (niektóre kamery domyślnie rejestrują odwrócony obraz). Z kolei po prawej stronie panelu znajduje się przycisk umożliwiający włączenie i wyłączenie dźwięku. Na ekranach widoków treningowych widoczne jest także pole, które wyświetla liczbę wykonanych powtórzeń przez użytkownika.

6.3. Integracja z systemem asystenta treningowego

Podzielony na moduły system asystenta treningowego został zaimplementowany, korzystając z obiektowego paradygmatu programowania. Moduł przysiadu, stworzony w ramach niniejszej pracy, stanowi jedną, oddzielną klasę. Po wyborze ćwiczenia oraz trybu działania tworzona jest instancja tej klasy wewnątrz aplikacji. Do jej konstruktora podawana jest tylko informacja o szybkości przechwytywania obrazu (liczba klatek na sekundę), która jest potrzebna do poprawnego wykrywania jednego z błędów technicznych. Równolegle inicjowany jest także algorytm BlazePose, który również został opakowany w oddzielną klasę, aby ułatwić jego wykorzystanie.

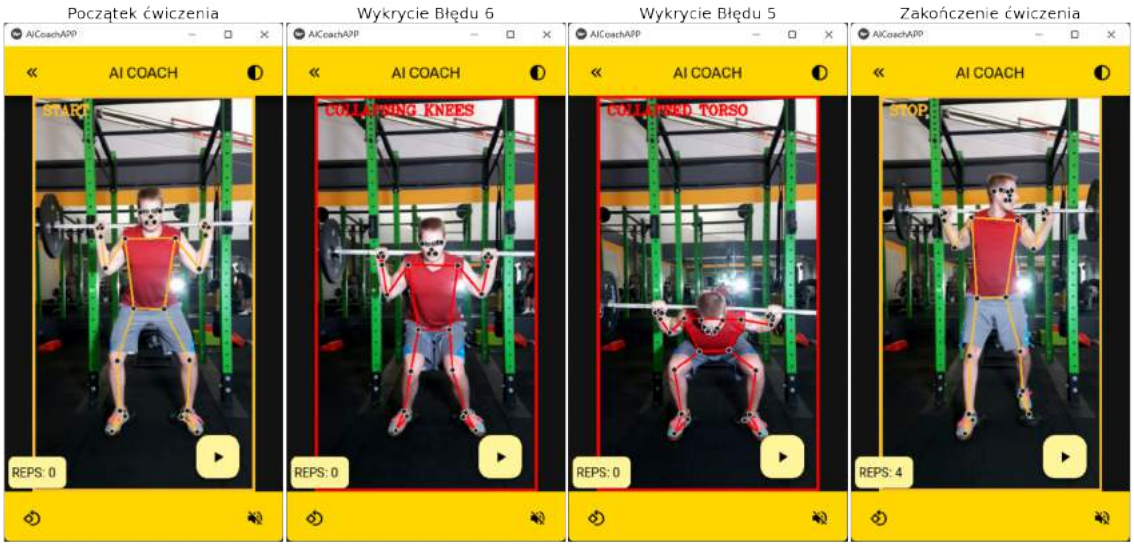


Rysunek 6.2: Schemat blokowy przedstawiający działanie systemu asystenta treningowego wewnątrz aplikacji komputerowej

Źródło: opracowanie własne

W momencie przejścia do któregoś z widoków treningowych, wewnątrz aplikacji inicjowany jest proces cyklicznego przechwytywania klatek z kamery urządzenia lub z wczytanego nagrania treningowego. Jego częstotliwość jest uzależniona od parametrów kamery lub nagrania (liczby klatek na sekundę). Po każdym przechwyceniu rozmiar klatki jest pomniejszany, a następnie algorytm BlazePose dokonuje na niej estymacji. Wynik tej estymacji jest później podawany na wejście metody zaimplementowanej w klasie modułu treningowego. Wewnątrz tej metody położenia punktów kluczowych są przekształcane na cechy wejściowe do wszystkich modeli opisanych w Rozdział 5. Następnie przeprowadzana jest detekcja stanu wykonywanego ćwiczenia oraz detekcje wszystkich obsługiwanych błędów technicznych. Moduł wynik swoich obliczeń zwraca w postaci ustrukturyzowanej odpowiedzi, informującej o wystąpieniu różnych zdarzeń oraz błędów technicznych. Szczegółowa struktura odpowiedzi modułu treningowego oraz ogólny schemat działania systemu asystenta treningowego wewnątrz aplikacji zostały przedstawione na rys. 6.2.

Aplikacja odpowiedź otrzymaną od modułu treningowego przekształca na różne komunikaty wizualne oraz dźwiękowe. Zdarzenia oznaczające początek i koniec ćwiczenia sygnalizowane są na ekranie oraz za pomocą charakterystycznych sygnałów dźwiękowych. W przypadku wykrycia błędu technicznego na ekranie pojawia się napis z nazwą popełnionego błędu, a jego obramowanie i zaznaczony szkielet osoby ćwiczącej zmieniają kolor na czerwony. Równolegle aplikacja odtwarza także kilkusekundowe nagrania dźwiękowe, które informują użytkownika o potencjalnym sposobie na poprawę danego błędu technicznego. Zdarzenie, oznaczające zakończenie pojedynczego powtórzenia, inkrementuje wartość licznika powtórzeń, który jest widoczny na ekranie. Przykłady komunikatów wizualnych zostały przedstawione na rys. 6.3.



Rysunek 6.3: Przykłady komunikatów wizualnych informujących o wystąpieniu różnych zdarzeń i popełnieniu błędów technicznych

Źródło: opracowanie własne

Wykrycie dużej liczby zdarzeń w krótkim czasie mogłoby powodować nakładanie się komunikatów na siebie, co prowadziłoby do dezinformacji. Aby tego uniknąć, wprowadzono dwa systemy ich kolejkowania, oddzielnie dla komunikatów wizualnych oraz dźwiękowych. Po pojawieniu się nowego zdarzenia, jego komunikaty (wizualne oraz dźwiękowe) trafiają na końce odpowiednich kolejek FIFO[8].

[8]Kolejka FIFO (First In First Out) - abstrakcyjna struktura danych, w której nowe elementy dodawane są zawsze na koniec kolejki, natomiast pobieranie danych ma miejsce na jej początku.

Komunikaty, znajdujące się wewnątrz kolejki, są chronologicznie sygnalizowane użytkownikowi. Każdy komunikat wizualny jest wyświetlany na okres około 1 sekundy i po jego zakończeniu pobierany jest następny komunikat z kolejki. Z kolei komunikaty dźwiękowe mają różną długość czasu trwania, dlatego nowy komunikat jest rozpoczynany dopiero w momencie zakończenia poprzedniego nagrania. Aby uniknąć odtwarzania zaległych komunikatów dopiero w trakcie wykonywania następnych powtórzeń, kolejki są zerowane w momencie pojawienia się zdarzenia, które oznacza rozpoczęcie nowego powtórzenia. Podobna reakcja ma miejsce w przypadku zdarzenia, oznaczającego zakończenie ćwiczenia - aplikacja automatycznie ucina wszystkie aktualne komunikaty (zarówno wizualne, jak i dźwiękowe) i sygnalizuje użytkownikowi wykryty koniec serii powtórzeń. Jeśli w momencie wystąpienia zdarzenia, mówiącego o zakończeniu powtórzenia, w kolejce komunikatów dźwiękowych nie oczekują żadne nagrania, to użytkownikowi odtwarzany jest krótki komunikat, mówiący o aktualnym stanie licznika powtórzeń.

7. TESTY

Rozdział opisuje przeprowadzone testy skuteczności stworzonego systemu asystenta treningowego. Uzyskane wyniki zostały przeanalizowane i przedstawione w formie tabelarycznej.

7.1. Testy systemu asystenta treningowego

Aby sprawdzić skuteczność działania wszystkich funkcji oferowanych przez stworzony system asystenta treningowego, zdecydowano się zebrać dodatkowe nagrania treningowe. Podobnie jak w przypadku zbioru uczącego, do nagrań zaproszono 5 osób średniozaawansowanych treningowo, które były zaznajomione z poprawną techniką wykonywania przysiadów ze sztangą z tyłu. Nagrania żadnego z uczestników nie były częścią zbioru danych, który został wykorzystany podczas projektowania systemu. Każdego z uczestników poproszono o wykonanie dwóch serii powtórzeń. W pierwszej z nich użytkownicy wykonali po około 5 powtórzeń przy zachowaniu ich domyślnej techniki wykonywania ćwiczenia. Natomiast w drugiej każdy z nich wykonał najpierw jedno poprawne powtórzenie, a następnie starał się imitować poszczególne błędy techniczne w kolejnych powtórzeniach.

Podczas testów sprawdzano skuteczność wykrywania zdarzeń (początek i koniec ćwiczenia oraz zliczanie wykonanych powtórzeń) oraz skuteczność detekcji błędów technicznych. Poprawność zgłaszania błędów była oceniana na poziomie pojedynczego powtórzenia, a nie na poziomie poszczególnych klatek nagrania. Błędy wykonane przez uczestników zostały dodatkowo sklasyfikowane, korzystając ze skali pewności zaproponowanej podczas oznaczania oryginalnego zbioru. Fakt wystąpienia błędów oraz skala ich pewności były oceniane na podstawie doświadczenia treningowego autora pracy. Monitorowana była także liczba nadmiarowych detekcji dla poszczególnych zdarzeń oraz błędów.

Tabela 7.1: Wyniki testów skuteczności systemu asystenta treningowego na nagraniach z domyślną techniką wykonywania ćwiczeń

Zdarzenie	**Skuteczność (poprawne detekcje / rzeczywiste wystąpienia)**			**Nadmiarowe detekcje**
	Pewność 1	**Pewność 2**	**Pewność 3**	
Początek ćwiczenia	-	-	5/5	0
Koniec ćwiczenia	-	-	2/2	0
Wykonanie powtórzenia	-	-	25/25	0
Niepełny zakres ruchu (Błąd 1)	-	-	0/0	0
Zbyt szybkie opuszczanie w dół (Błąd 9)	-	-	0/0	0
Niesymetryczne chwycenie sztangi (Błąd 2)	0/0	0/0	0/0	0
Niepoprawna pozycja głowy (Błąd 3)	0/0	0/0	0/0	0
Wychylenie łokci do tyłu (Błąd 4)	0/0	0/0	0/0	1
Zapadanie się sylwetki do przodu (Błąd 5)	0/0	0/0	0/0	0
Zapadanie się kolan do środka (Błąd 6)	0/0	0/0	0/0	0
Asymetryczna praca bioder (Błąd 7)	0/0	0/0	0/0	0
Unoszenie pięt (Błąd 8)	0/0	0/0	0/0	0

Testy podzielono na dwie części - nagrania z domyślną techniką uczestników oraz nagrania, które przedstawiają imitację błędów technicznych. W Tabela 7.1 znajdują się wyniki z pierwszej części. Żaden z uczestników nie popełniał błędów technicznych, a system jeden raz zwrócił fałszywą informację o błędzie zbyt mocnego wychylenia łokci do tyłu. Wszystkie wykonane powtórzenia zostały poprawnie policzone, a także prawidłowo wykryto momenty rozpoczęcia oraz zakończenia ćwiczenia (te ostatnie pojawiły się tylko na 2 nagraniach, ponieważ zbyt szybko stopowano nagrywanie).

Z kolei Tabela 7.2 zawiera wyniki testów na nagraniach, na których uczestnicy starali się imitować błędy techniczne. System omyłkowo zliczył 6 wykonanych powtórzeń oraz jedno zakończenie ćwiczenia w połowie wykonywania serii, co następnie skutkowało nadmiarową detekcją rozpoczęcia ćwiczenia. Przy błędach, wynikających z trajektorii ruchu (Błąd 1 i Błąd 9) nie popełniono żadnych pomyłek. Z pozostałych błędów najlepiej wypadły błędy 2, 5 oraz 6, dla których system pominął pojedyncze ich wystąpienia. Błędy 3 oraz 4 system wykrył ze 100% skutecznością, jednak dodatkowo sygnalizował ich fałszywe wystąpienia. W przypadku błędów 7 oraz 8, oprócz niewykrytych przypadków ich popełnienia, system kilka razy nadmiarowo zgłaszał ich wystąpienie.

Tabela 7.2: Wyniki testów skuteczności systemu asystenta treningowego na nagraniach zawierających imitację błędów technicznych

Zdarzenie	Skuteczność (poprawne detekcje / rzeczywiste wystąpienia)			Nadmiarowe detekcje
	Pewność 1	Pewność 2	Pewność 3	
Początek ćwiczenia	-	-	6/6	1
Koniec ćwiczenia	-	-	6/7	1
Wykonanie powtórzenia	-	-	52/52	6
Niepełny zakres ruchu (Błąd 1)	-	-	4/4	0
Zbyt szybkie opuszczanie w dół (Błąd 9)	-	-	10/10	0
Niesymetryczne chwycenie sztangi (Błąd 2)	0/0	0/1	4/4	0
Niepoprawna pozycja głowy (Błąd 3)	1/1	0/0	5/5	2
Wychylenie łokci do tyłu (Błąd 4)	3/3	1/1	4/4	3
Zapadanie się sylwetki do przodu (Błąd 5)	0/0	0/0	4/5	0
Zapadanie się kolan do środka (Błąd 6)	0/0	1/1	3/4	0
Asymetryczna praca bioder (Błąd 7)	0/1	0/1	2/3	3
Unoszenie pięt (Błąd 8)	1/1	2/2	3/5	5

W trakcie testów sprawdzono także szybkość przetwarzania klatek przez system asystenta treningowego wewnątrz aplikacji na urządzeniu, którego specyfikacja znajduje się w Tabela 5.1. Zarówno dla trybu nagrań treningowych oraz trybu treningu na żywo na ekranie urządzenia klatki były odświeżane od 18 do 23 razy na sekundę. Ich liczba zależała od aktualnego stanu wykonywanego ćwiczenia, ponieważ klasyfikatory błędów dokonują predykcji tylko w trakcie wykonywania powtórzenia. Uzyskany wynik spełnia założenie działania w czasie rzeczywistym, które zostało zdefiniowane jako zdolność do przetwarzania minimum 15 klatek na sekundę.

7.2. Analiza błędów popełnionych przez system

W przypadku nagrań z poprawną techniką wykonywania ćwiczenia system działał bez zarzutu. Poza pojedynczym wyjątkiem, system nie dezinformował użytkowników komunikatami o fałszywych pojawieniach się błędów, co starano się osiągnąć w trakcie projektowania poszczególnych klasyfikatorów poprzez maksymalizację ich precyzji. Na nagraniach z błędami, wyniki pod tym względem były już nieco gorsze. O ile w przypadku Błędu 3 oraz Błędu 4 nadmiarowe detekcje pojawiały się głównie w trakcie występowania innych błędów, to dla błędów 7 oraz 8 fałszywe wystąpienia były sygnalizowane bez najmniejszych powodów.

Nadmiarowe detekcje dotyczyły również zdarzeń wnioskowanych na podstawie predykcji klasyfikatora stanu (rozpoczęcie/zakończenie ćwiczenia oraz wykonanie powtórzenia). Wszystkie ich fałszywe zgłoszenia były spowodowane zbyt pochopnymi zmianami stanu, których dokonywał wytrenowany klasyfikator. O tym problemie informowano także w Sekcja 5.2 - w zbiorze uczącym oznaczenia stanu ćwiczenia są wykonane zbyt drobiazgowo względem dokładności algorytmu BlazePose. Próbowano temu zaradzić, zmieniając minimalne progi prawdopodobieństwa dla różnych klas oraz dodając mechanizm pamięci o poprzednich predykcjach, jednak mimo to zdarzają się pomyłki widoczne w wynikach przeprowadzonych testów. Najpewniejszym rozwiązaniem tego problemu byłoby ponowne oznaczenie zbioru poprawionymi etykietami stanu, jednak jest to proces bardzo czasochłonny.

Najgorszą skuteczność detekcji (2/5) system uzyskał w przypadku błędu związanego z asymetryczną pracą bioder przy przysiadzie. Dodatkowo, błąd ten został także trzykrotnie zasygnalizowany w sposób fałszywy. Jest to błąd najbardziej złożony i większość osób ćwiczących popełnia go w inny sposób. Biorąc to pod uwagę, istnieje duża szansa, że klasyfikator odpowiedzialny za ten błąd został przetrenowany z powodu zbyt małej liczby przykładów treningowych. Dodatkowym czynnikiem, negatywnie wpływającym na jego przewidywanie, jest zapewne fakt, że większość zmiennych wejściowych wspomnianego klasyfikatora dotyczy symetryczności ułożenia różnych części ciała osoby ćwiczącej. W przypadku, gdy użytkownik nie ustawi się idealnie na przeciwko kamery urządzenia, wartości tych cech są zakłamywane z powodu pogorszenia się dokładności algorytmu BlazePose.

Drugim najgorzej obsługiwanym błędem technicznym jest błąd dotyczący unoszenia pięt w trakcie przysiadu. O ile skuteczność jego wykrywania nie jest najgorsza (6/8), to system aż pięciokrotnie zwrócił fałszywy komunikat o jego wystąpieniu. Błąd ten często jest trudny do wykrycia na podstawie pojedynczej klatki nawet dla ludzkiego oka. Z drugiej strony, posiadając informacje o położeniu szkieletu osoby ćwiczącej, można łatwo obliczyć nachylenie stóp względem podłoża. Niestety algorytm BlazePose bardzo niedokładnie szacuje ich położenie. Przykładowo, podczas stania w pozycji startowej na pełnych stopach, wyestymowane kąty ich nachylenia względem podłoża wynoszą zazwyczaj kilkadziesiąt stopni. Co gorsza, jak zauważono w trakcie oznaczania zbioru danych, jakość oświetlenia, typ obuwia i kolor podłoża znacząco wpływają na rozbieżności w ich estymacjach. Będąc tego świadomym, w trakcie projektowania klasyfikatora dodano także inne cechy wejściowe (m.in. nachylenie piszczeli na płaszczyźnie YZ, które zazwyczaj wzrasta w momencie unoszenia pięt). Jednak najprawdopodobniej rozmiar zbioru treningowego był zbyt mały, aby nauczyć model tych bardziej złożonych zależności.

Błędy 2, 3, 4, 5 oraz 6 system wykrywał z co najmniej 80% skutecznością. Dodatkowo kilka z wystąpień tych błędów była nieoczywista i tylko trochę akcentowana przez uczestników.

8. PODSUMOWANIE

W ramach niniejszej pracy stworzono prototyp aplikacji, która pomaga osobom początkującym podczas wykonywania przysiadów ze sztangą z tyłu. Pracę rozpoczęto od przeglądu zbliżonych rozwiązań dostępnych na rynku globalnym. W dalszej kolejności zapoznano się z zagadnieniem estymacji pozy człowieka i wybrano jeden z gotowych algorytmów. Następnie zaznajomiono się z poprawną techniką wykonywania przysiadów i wyłoniono podzbiór błędów technicznych najczęściej popełnianych przez osoby ćwiczące. Na potrzeby pracy stworzono także od podstaw zbiór danych - zebrano kilkaset rzeczywistych nagrań treningowych, które później ręcznie oznaczono odpowiednimi etykietami. Po przygotowaniu zbioru danych rozpoczęto pracę nad systemem asystenta treningowego.

Stworzony system potrafi stwierdzić, jaki jest aktualny stan wykonywanego ćwiczenia, oraz dokonuje detekcji błędów technicznych popełnionych przez użytkownika. System został wdrożony do aplikacji komputerowej, która wykryte zdarzenia sygnalizuje w postaci komunikatów wizualnych oraz dźwiękowych. Na sam koniec przeprowadzono testy skuteczności gotowego rozwiązania. Ich wyniki okazały się być dobre pomimo stosunkowo niewielkiego rozmiaru zbioru treningowego. Finalny rezultat projektu świadczy o potencjalnej możliwości stworzenia bliźniaczej aplikacji w wersji komercyjnej.

8.1. Potencjalny rozwój systemu asystenta treningowego

Głównym czynnikiem, który limituje skuteczność wykrywania zdarzeń przez stworzony system jest wielkość zbioru treningowego. Aby umożliwić jego lepsze działanie, należałoby przede wszystkim zwiększyć jego rozmiar. Jest to szczególnie konieczne w przypadku bardziej złożonych błędów technicznych. Nie jest to łatwe zadanie, ponieważ wymaga dużych zasobów czasowych.

Drugim czynnikiem, znacząco wpływającym na popełniane przez system błędy w predykcjach, jest niedokładność algorytmu estymującego pozę. Ciężko byłoby od podstaw stworzyć model o lepszej skuteczności niż algorytmy zaproponowane przez firmy takie jak Google czy Facebook, jednak można by spróbować doszkolić istniejące rozwiązania na zbiorze danych przygotowanym na potrzeby niniejszej pracy. Wymagałoby to jednak ponownego oznaczenia nagrań zgodnie ze standardami przyjętymi podczas projektowania tamtych algorytmów, co również byłoby bardzo czasochłonne.

System asystenta treningowego można także rozszerzyć o dodatkowe moduły, obsługujące inne ćwiczenia siłowe. Wiązałoby się to z koniecznością poznania zasad ich poprawnego wykonywania oraz z wyłonieniem dla nich najczęściej popełnianych błędów technicznych. Dla każdego dodanego ćwiczenia należałoby również zebrać nagrania treningowe, które licznie przedstawiałyby przypadki występowania tych błędów. Klatki zebranych nagrań powinny być także ręcznie oznaczone odpowiednimi etykietami. Dopiero po tych czynnościach można by rozpocząć pracę nad projektowaniem modeli predykcyjnych, które wykrywałyby różne zdarzenia w trakcie wykonywania tych ćwiczeń.

Kolejnym, bardziej zaawansowanym etapem rozwoju systemu mogłoby być dodanie obsługi spersonalizowanych błędów technicznych, których wystąpienia zależą również od indywidualnych parametrów osoby ćwiczącej. To rozwiązanie wymagałoby jednak jeszcze większego poszerzenia zbioru nagrań treningowych oraz zebrania dodatkowych informacji o parametrach uczestników, którzy znajdowaliby się na tych materiałach wideo. Przygotowany w ten sposób zbiór danych powinien zapewniać różnorodność pod względem różnych cech uczestników (powinien być reprezentatywny). Podczas projektowania klasyfikatorów do tych błędów należałoby uwzględnić parametry użytkownika w zbiorze ich cech wejściowych.

Do systemu asystenta treningowego można by także dodać funkcjonalności, które są dostępne w aplikacji MetricVBT, szerzej przedstawionej w Rozdział 2. Wyliczenie statystyk w postaci prędkości oraz średniego zakresu ruchu nie powinno być problematyczne, posiadając informację o położeniu szkieletu osoby wykonującej ćwiczenie. Możliwe jest także rozszerzenie systemu o funkcję analizy trajektorii ruchu sztangi, z której użytkownik mógłby skorzystać, ustawiając kamerę od boku. Takie rozwiązanie zostało zaimplementowane w aplikacji WL Analysis i jest przydatne dla osób bardziej zaawansowanych w treningu siłowym.

8.2. Potencjalny rozwój aplikacji

Aplikacja stworzona na potrzeby niniejszej pracy działa na urządzeniach stacjonarnych. Dzięki wykorzystaniu do jej implementacji platformy Kivy, istnieje możliwość przerzucenia jej także na urządzenia mobilne stosunkowo niskim kosztem. Dostępność z poziomu telefonu komórkowego znacząco ułatwiłaby użytkownikom korzystanie z niej w miejscach przeznaczonych do wykonywania ćwiczeń.

Z perspektywy osób początkujących dobrym rozwiązaniem byłoby umieszczenie wewnątrz aplikacji nagrań instruktażowych, demonstrujących poprawną technikę wykonywania poszczególnych ćwiczeń siłowych. Mogłyby one także ostrzegać i informować o najczęściej popełnianych błędach technicznych. Osoby niedoświadczone, po obejrzeniu takiego nagrania, byłyby już zaznajomione z podstawami teoretycznymi, co z pewnością przekładałoby się na mniejszą liczbę popełnianych przez nie błędów w trakcie wykonywania pierwszej serii powtórzeń danego ćwiczenia.

Do aplikacji można by także dodać funkcje dzienników dietetycznych oraz treningowych. Wiele osób ćwiczących siłowo w trakcie swoich treningów zapisuje osiągnięte wyniki, aby mieć lepszą kontrolę nad progresowaniem w poszczególnych ćwiczeniach. Możliwość zapisu informacji o przebiegu treningu wewnątrz aplikacji byłaby dla nich znacznym ułatwieniem. Analogicznie jest w przypadku prowadzenia dzienników dietetycznych - znaczna część osób aktywnych fizycznie liczy kalorie oraz makroskładniki spożyte w ciągu dnia, dlatego obecność takiej funkcji wewnątrz jednej aplikacji mogłaby zaoszczędzić im sporo czasu.

SPIS RYSUNKÓW

SPIS TABEL

BIBLIOGRAFIA

[1] Aplikacja Metric VBT [online], https://www.metric.coach/, (data dostępu 09.12.2022 r.)

[2] Aplikacja WL Analysis [online], https://wlanalysis.com/, (data dostępu 09.12.2022 r.)

[3] Aplikacja Kemtai [online], https://kemtai.com/, (data dostępu 09.12.2022 r.)

[4] Yucheng Chen, Yingli Tian, Mingyi He, 2020, "Monocular human pose estimation: A survey of deep learning-based methods"

[5] E. Hazan, S. Safra, O. Schwartz, 2003, "On the Hardness of Approximating k-Dimensional Matching"

[6] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh, 2016, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields"

[7] Simonyan, Andrew Zisserman, 2014, "Very Deep Convolutional Networks for Large-Scale Image Recognition"

[8] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, Cewu Lu, 2016, "RMPE: Regional Multi-Person Pose Estimation"

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, 2015, "SSD: Single Shot MultiBox Detector"

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, 2015, "Spatial Transformer Networks"

[11] Rıza Alp Güler, Natalia Neverova, Iasonas, Kokkinos, 2018, "DensePose: Dense Human Pose Estimation In The Wild"

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár, 2014, "Microsoft COCO: Common Objects in Context"

[13] Rıza Alp Güler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, Iasonas Kokkinos, 2016, "DenseReg: Fully Convolutional Dense Shape Regression In-the-Wild"

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017, "Mask R-CNN"

[15] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, 2016, "Feature Pyramid Networks for Object Detection"

[17] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, Matthias Grundmann, 2020, "BlazePose: On-device Real-time Body Pose tracking"

[18] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, Matthias Grundmann, 2019, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs"

[19] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias Grundmann, 2020, "MediaPipe Hands: On-device Real-time Hand Tracking"

[20] Ivan Grishchenko, Valentin Bazarevsky, Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, Richard Yee, Karthik Raveendran, Matsvei Zhdanovich, Matthias Grundmann, Cristian Sminchisescu, 2022, "BlazePose GHUM Holistic: Real-time 3D Human Landmarks and Pose Estimation"

[21] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T. Freeman, Rahul Sukthankar, Cristian Sminchisescu, 2020, "GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models"

[22] Dokumentacja pakietu MediaPipe [online], `https://google.github.io/mediapipe/`, (data dostępu 09.12.2022 r.)

[23] Tianqi Chen, Carlos Guestrin, 2016, "XGBoost: A Scalable Tree Boosting System"

[24] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama, 2019, "Optuna: A Next-generation Hyperparameter Optimization Framework"

A. SKRÓTOWY OPIS DYPLOMU

A.1. Tytuł dyplomu

System asystenta treningu siłowego przy użyciu wizji komputerowej.

A.2. Cel i przeznaczenie

Celem pracy jest stworzenie prototypu aplikacji pomagającej osobom początkującym podczas treningu siłowego. Aplikacja ma zapobiegać popełnianiu błędów technicznych w trakcie wykonywania ćwiczeń, ponieważ mogą one powodować negatywne skutki zdrowotne oraz obniżać efektywność treningu.

A.3. Funkcjonalność

Aplikacja na podstawie nagrania z pojedynczej kamery RGB potrafi w czasie rzeczywistym wykrywać momenty rozpoczęcia i zakończenia ćwiczenia oraz zliczać powtórzenia wykonane przez osobę ćwiczącą. System rozpoznaje także błędy techniczne popełnione przez użytkownika. Wszystkie wykryte zdarzenia są sygnalizowane użytkownikowi za pomocą komunikatów wizualnych oraz dźwiękowych.

A.4. Ogólna architektura systemu

System asystenta treningowego jako dane wejściowe przyjmuje informacje o położeniu punktów kluczowych na ciele osoby ćwiczącej, które są dostarczane przez wybrany algorytm estymujący pozę człowieka. W pierwszej kolejności wewnątrz systemu dokonywana jest klasyfikacja aktualnego stanu ćwiczenia, co umożliwia detekcję różnych zdarzeń oraz niektórych błędów technicznych. Pozostałe błędy są wykrywane za pomocą kilku klasyfikatorów binarnych, z których każdy obsługuje osobny typ błędu technicznego. Każdy z poszczególnych modeli podejmuje decyzje w oparciu o indywidualny zbiór kątów pomiędzy różnymi częściami ciała oraz odległości pomiędzy parami różnych punktów kluczowych na sylwetce użytkownika.

A.5. Architektura oprogramowania

Do implementacji aplikacji wykorzystano bibliotekę Kivy dostępną z poziomu języka Python. Aplikacja potrafi obsługiwać kamerę urządzenia w przypadku treningu na żywo lub działać na nagraniach treningowych wczytanych z pamięci urządzenia. Aplikacja zapewnia przepływ informacji pomiędzy obsługiwanym nagraniem, algorytmem estymującym pozę oraz systemem asystenta treningowego. Zdarzenia oraz popełnione błędy techniczne wykryte przez system aplikacja sygnalizuje użytkownikowi na ekranie urządzenia oraz odtwarza w postaci krótkich komunikatów dźwiękowych.

A.6. Przebieg pracy nad dyplomem

A.6.1. Założenia i sformułowanie zadania

Zadaniem określonym w pracy jest stworzenie systemu asystenta treningowego, który będzie w czasie rzeczywistym wykrywał różne zdarzenia oraz błędy techniczne popełniane podczas treningu siłowego. System ma być zaimplementowany za pomocą modeli uczenia maszynowego, które będą dokonywać predykcji na podstawie informacji o położeniu punktów kluczowych osoby ćwiczącej. Informacje o położeniu punktów kluczowych będą dostarczane przez wybrany algorytm estymujący pozę człowieka. Zakłada się wdrożenie gotowego systemu do aplikacji komputerowej, zapewniającej swobodną interak-

cję z użytkownikiem. Działanie aplikacji zostało ograniczone do jednego ćwiczenia siłowego - przysiadu ze sztangą z tyłu.

A.6.2. Przegląd rozwiązań

Na rynku globalnym coraz częściej pojawiają się rozwiązania pozwalające analizować jakość wykonywania ćwiczeń na podstawie nagrania z kamery. Aplikacje Metric VBT oraz WL Analysis umożliwiają analizę parametrów kinematycznych wykonanego ćwiczenia. Pierwsza z nich oblicza różne statystyki ruchu na podstawie analizy zrealizowanych przez użytkownika powtórzeń. Natomiast drugi program wykrywa końcówkę sztangi, śledzi jej trajektorię i prezentuje podstawowe parametry kinematyczne na wykresie. W odniesieniu do automatycznego wykrywania błędów technicznych, liczba dostępnych aplikacji jest bardzo ograniczona, co świadczy o innowacyjności takiego rozwiązania. Jedną z nielicznych jest płatna platforma Kemtai, która potrafi ocenić każde wykonane powtórzenie i zasygnalizować popełnione błędy, dzięki śledzeniu kilkudziesięciu punktów na ciele użytkownika.

A.6.3. Estymacja pozy człowieka

Fundamentem do realizacji projektu jest estymacja pozy człowieka, czyli zagadnienie z dziedziny wizji komputerowej, które dotyczy problemu problemu identyfikacji oraz wyznaczania położenia punktów kluczowych sylwetki człowieka na zdjęciach. Zapoznano się z istniejącymi algorytmami estymującymi pozę i na potrzeby pracy wybrano model BlazePose, który jest dedykowany na urządzenia mobilne i potrafi dokonywać predykcji w przestrzeni trójwymiarowej. Zdecydowano się na to rozwiązanie z uwagi na krótszy czas inferencji i porównywalną dokładność estymacji względem innych topowych algorytmów.

A.6.4. Zbiór danych

Do stworzenia modeli klasyfikacyjnych, wchodzących w skład systemu asystenta treningowego potrzebny był odpowiedni zbiór danych. Z uwagi na brak publicznie dostępnych zbiorów z materiałami z treningu siłowego, zdecydowano się przygotować taki zbiór od podstaw. Zebrano nagrania treningowe 22 osób średnio zaawansowanych treningowo, którzy w kolejnych seriach powtórzeń imitowali popełnianie różnych błędów technicznych. Zaproponowano także rozbudowany system oznaczania klatek z zebranych nagrań. Oprócz pojedynczej etykiety mówiącej o popełnieniu danego błędu stosowano także skalę, która rozróżniała pewność i widoczność jego wystąpienia na konkretnej klatce. Dodatkowo wprowadzono oznaczenia dotyczące stanu wykonywanego ćwiczenia. Finalnie przygotowany zbiór danych zawierał około 20 tysięcy przykładów uczących.

A.6.5. Implementacja

System asystenta treningowego składa się z kilku klasyfikatorów, do implementacji których wykorzystano algorytm drzew decyzyjnych wzmacnianych gradientowo z biblioteki XGBoost. Do każdego modelu indywidualnie dobrano zbiór cech wejściowych, zawierający kąty pomiędzy różnymi częściami ciała oraz odległości pomiędzy parami różnych punktów kluczowych sylwetki. Odpowiedź systemu zawiera informacje o wystąpieniu różnych zdarzeń oraz błędów technicznych wykrytych na aktualnej klatce nagrania. System jest wdrożony do aplikacji komputerowej, która integruje go z algorytmem estymującym pozę i obsługuje kamerę urządzenia lub gotowe nagranie wczytane z jego pamięci. Wewnątrz aplikacji zaimplementowano funkcję przetwarzającą odpowiedź systemu na komunikaty wizualne wyświetlane na ekranie oraz dźwiękowe odtwarzane przez głośniki urządzenia.

A.6.6. Testowanie

System został poddany testom na nagraniach z 5 nowymi osobami wykonującymi ćwiczenia. Każdy uczestnik wykonał dwie serie powtórzeń - pierwszą z domyślną techniką wykonywania ćwiczenia i drugą starając się imitować różne błędy techniczne. W trakcie testów sprawdzano dokładność zdarzeń i błędów technicznych zgłaszanych przez system oraz monitorowano szybkość przetwarzania klatek.

A.6.7. Ocena rozwiązania i wnioski

Gotowe rozwiązanie spełnia przyjęte założenia. System większość błędów technicznych i zdarzeń wykrywa z wysoką skutecznością, a sama aplikacja działa w czasie rzeczywistym. Finalny rezultat projektu świadczy o potencjalnej możliwości stworzenia bliźniaczej aplikacji w wersji komercyjnej.

A.6.8. Perspektywy dalszych prac

Dalsze prace nad systemem powinny skupiać się przede wszystkim na zwiększeniu rozmiaru zbioru danych, ponieważ jest to główny czynnik ograniczający dokładność uzyskiwanych predykcji. System można także rozszerzyć o dodatkowe moduły obsługujące pozostałe ćwiczenia siłowe. Natomiast do samej aplikacji można dodać kilka niezależnych funkcjonalności, które byłyby atrakcyjne z perspektywy osób aktywnych fizycznie - np. funkcje dzienników dietetycznych i treningowych oraz możliwość odtwarzania nagrań instruktażowych.

B. INSTRUKCJA DLA UŻYTKOWNIKA

B.1. Wymagania

Przed uruchomieniem aplikacji należy zainstalować na urządzeniu stacjonarnym język Python w wersji 3.10 wraz ze wszystkimi pakietami wymienionymi w pliku "requirements.txt", który znajduje się w katalogu głównym zawierającym kod projektu. Po przygotowaniu środowiska wystarczy uruchomić program "main.py", który włącza aplikację. Aby system działał w czasie zbliżonym do rzeczywistego, wymagane są odpowiednie zasoby obliczeniowe. Zaleca się uruchamianie aplikacji na urządzeniach o specyfikacji co najmniej zbliżonej do tej, która została przedstawiona w Tabela 5.1.

Aby umożliwić działanie aplikacji w trybie treningu na żywo, konieczne jest, aby urządzenie było wyposażone w kamerę. Zaleca się, aby wykorzystywana kamera była o jak najwyższej jakości, ponieważ ma to pośredni wpływ na skuteczność działania systemu asystenta treningowego. Urządzenie powinno być także wyposażone w głośniki lub słuchawki, aby pozwolić aplikacji na odtwarzanie różnych komunikatów głosowych, które ułatwiają wykonywanie ćwiczeń siłowych.

B.2. Użytkowanie

Poniżej przedstawiono kilka wskazówek, opisujących korzystanie z interfejsu aplikacji oraz efektywne użytkowanie systemu asystenta treningowego:

1. Wybierz rodzaj ćwiczenia (aktualnie obsługiwany jest jedynie przysiad ze sztangą z tyłu)
2. Wybierz tryb działania aplikacji - trening na żywo lub zapisane nagranie treningowe
3. W obu wspomnianych powyżej przypadkach:
    - kamera powinna być ustawiona na wysokości około jednego metra, w odległości kilku metrów od miejsca wykonywania ćwiczenia, tak aby cała sylwetka osoby trenującej była widoczna
    - zadbaj, aby nic nie zasłaniało twarzy oraz ciała osoby wykonującej ćwiczenie
    - osoba ćwicząca powinna być zwrócona idealnie w stronę kamery
    - zadbaj o dobre oświetlenie miejsca, w którym wykonywane jest ćwiczenie
4. Po uruchomieniu któregoś z widoków treningowych, system zacznie śledzić osobę widoczną na nagraniu. Aplikacja zasygnalizuje rozpoczęcie wykonywania ćwiczenia za pomocą napisu "START", który pojawi się na ekranie, oraz za pomocą krótkiego sygnału dźwiękowego. Po rozpoczęciu serii powtórzeń system będzie sprawdzał poprawność techniki wykonywania ćwiczenia. W przypadku wykrycia błędu technicznego, na ekranie wyświetli się jego nazwa oraz zostanie odtworzona wskazówka głosowa, mówiąca o tym, jak go wyeliminować w następnym powtórzeniu. System będzie także zliczał wykonane powtórzenia, co będzie można zauważyć w lewym dolnym rogu ekranu urządzenia oraz usłyszeć w postaci krótkich komunikatów. W momencie zakończenia ćwiczenia na ekranie pojawi się napis "STOP" oraz zostanie odtworzony krótki sygnał dźwiękowy.

Wskazówki o podobnej treści są również zamieszczone wewnątrz aplikacji. Aby je otworzyć, należy nacisnąć ikonę znaku zapytania, która znajduje się na panelu górnym w początkowym widoku aplikacji.

C. INSTRUKCJA DLA PROGRAMISTY

C.1. Struktura projektu

Poniżej przedstawiono finalną strukturę projektu wraz z opisem poszczególnych plików i katalogów:

- ***AI_COACH*** - katalog główny
  - ***data_utils*** - katalog zawierający narzędzia pomocnicze do przygotowania zbioru danych
    - ***capture_pose_from_videos.py*** - program odtwarzający nagrania wraz z naniesioną estymacją pozy z wybranej, zarejestrowanej wcześniej serii powtórzeń
    - ***label_dataset.py*** - program do etykietowania nagrań treningowych
    - ***record_videos.py*** - program do automatycznego nagrywania ćwiczeń trzema kamerami
  - ***assets*** - katalog zawierający zasoby wykorzystywane w projekcie
    - ***images*** - katalog zawierający ikony wykorzystane w interfejsie graficznym aplikacji
    - ***models*** - katalog zawierający wytrenowane modele uczenia maszynowego wykorzystywane wewnątrz systemu asystenta treningowego
      - ***squat*** - katalog zawierający modele wykorzystywane w module obsługującym przysiad ze sztangą z tyłu
    - ***sounds*** - katalog zawierający pliki dźwiękowe odtwarzane przez aplikację
  - ***AICoach.kv*** - plik zawierający deklarację interfejsu graficznego aplikacji
  - ***config.py*** - plik konfiguracyjny do aplikacji i systemu asystenta treningowego
  - ***exercise_modules.py*** - plik zawierający deklarację modułów systemu asystenta
  - ***main.py*** - plik główny aplikacji komputerowej
  - ***pose_module.py*** - plik zawierający deklarację modelu BlazePose jako oddzielnej klasy
  - ***requirements.txt*** - plik tekstowy zawierający listę wymaganych bibliotek
  - ***utils.py*** - plik z deklaracjami pomocniczych klas oraz funkcji wykorzystywanych w projekcie

C.2. Wprowadzanie zmian w projekcie

Zarówno system asystenta treningowego, jak i samą aplikację można łatwo rozwijać. Kod źródłowy systemu znajduje się w pliku "exercise_modules.py". Aktualnie system składa się z jednego modułu obsługującego przysiad ze sztangą z tyłu, który został zaimplementowany jako osobna klasa o nazwie "SquatModule". Oprócz konstruktora klasa posiada jedną metodę o nazwie "process_estimation_results", wewnątrz której dokonywana jest detekcja różnych zdarzeń oraz błędów technicznych. W tym miejscu należy dokonywać wszelkich zmian dotyczących logiki uzyskiwania predykcji. We wspomnianym pliku można także deklarować inne moduły, które obsługiwałyby pozostałe ćwiczenia siłowe.

Na kod źródłowy aplikacji komputerowej składają się dwa oddzielne pliki. Plik "AICoach.kv" zawiera deklarację interfejsu graficznego aplikacji przy pomocy wewnątrzplatformowego języka projektowania o nazwie Kivy Design Language. Z kolei w pliku "main.py" zawarta jest logika działania aplikacji, która obejmuje m.in. przepływ informacji pomiędzy nagraniem a różnymi modułami oraz przetwarzanie odpowiedzi systemu na komunikaty wizualne oraz dźwiękowe. We wspomnianych plikach można przeprowadzać zmiany w aplikacji oraz dokonywać rozszerzeń o dodatkowe funkcjonalności.