DOI: https://doi.org/10.53052/9788367652124.16

Jakub KRAWCZYK[1], Marcin BERNAŚ[2]

Opiekun naukowy: Marcin BERNAŚ[2]

MONITOROWANIE I KLASYFIKACJA RUCHÓW UŻYTKOWNIKA W ŚRODOWISKU VR

Streszczenie: W artykule zaprezentowano autorskie narzędzie do zbierania danych w środowisku wirtualnej rzeczywistości (VR). Narzędzie to pozwala na przechwytywanie i zapisywanie danych w celu dalszej analizy. Poprawność zbieranych danych przeanalizowano analitycznie oraz ich użyteczność z zastosowaniem sieci neuronowych. Przeanalizowano cztery klasy ruchu użytkownika, które nie były bezpośrednio śledzone w środowisku VR. Do analizy sekwencji ruchów nóg wykorzystano rekurencyjną sieć neuronową z określonym oknem czasowym. Wyniki wskazują, że narzędzie to umożliwia precyzyjne śledzenie ruchów głowy i rąk, a także rozpoznawanie typowych ruchów, takich jak chodzenie, bieganie czy przysiady, z dużą dokładnością.

Słowa kluczowe: środowisko wirtualnej rzeczywistości, monitorowanie ruchu, sieci neuronowe

TRACKING AND CLASSIFICATION OF VR ENVIRONMENT USER

Summary: The article presents an original tool for collecting data in a virtual reality (VR) environment. This tool allows you to capture and save data for further analysis. The correctness of the collected data was analyzed, as well as their usefulness using neural networks. Four classes of user movement that were not directly tracked in the VR environment were analyzed. A recurrent neural network with a specific time window was used to analyze the sequence of leg movements. The results indicate that the tool can precisely track head and arm movements, as well as recognize common movements such as walking, running, and squatting with high accuracy.

Keywords: virtual reality environment, motion tracking, neural networks

[1] Uniwersytet Bielsko-Bialski, Wydział Budowy Maszyn i Informatyki
[2] Uniwersytet Bielsko-Bialski, Wydział Budowy Maszyn i Informatyki, mbernas@ubb.edu.pl

1. Wprowadzenie

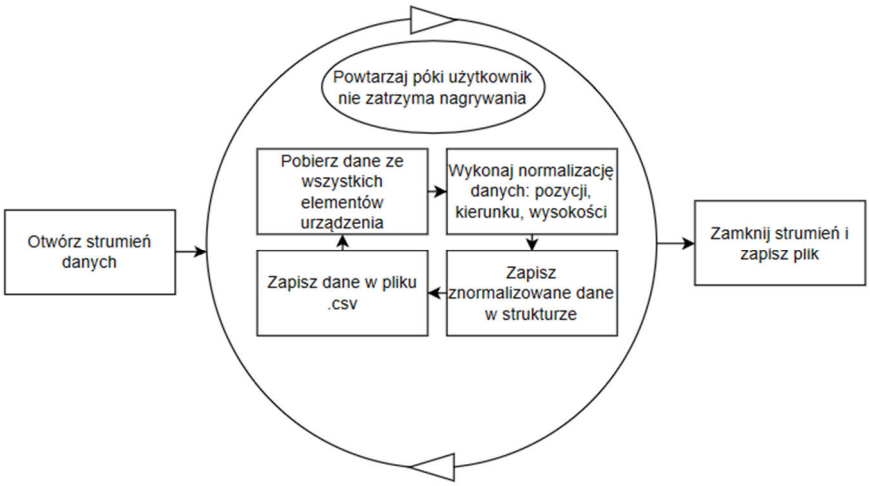
Wirtualna Rzeczywistość (z ang. Virtual Reality -VR) to obraz fikcyjnego świata nakładany na wycinek trójwymiarowej rzeczywistości przy wykorzystaniu technologii informatycznych. Polega na kreowaniu komputerowej wizji przedmiotów, przestrzeni oraz zdarzeń. Wewnątrz możemy zawrzeć zarówno elementy świata realnego (symulacje komputerowe), jak i fikcyjnego (gry komputerowe fantasy). Technologia ta ma szerokie zastosowanie w wielu dziedzinach takich jak na przykład szkolenie pilotów wojskowych i cywilnych [1], szkolenie chirurgów [2], oraz jako narzędzie do edukacji i rozrywki [3].

Rysunek 1. Przegląd gogli i manipulatorów VR

Imersję w wirtualnym świecie zapewnia możliwość rozglądania się czy interakcji ze światem. Poruszanie się po tym świecie odbywa się poprzez układy zainstalowane wewnątrz okularów oraz kontrolery takie jak np. akcelerometry (które mierzą przyśpieszenie liniowe), żyroskopy (które śledzą prędkość kontową) oraz magnetometry (które pozwalają określić orientację oraz eliminować tzw. „dryf"). Oprócz wymienionych przyrządów gogle (rys. 1) wyposażone są w zestaw kamer umieszczonych z przodu, które rejestrują i analizują otoczenie, co pozwala na bieżąco określać pozycję i orientację użytkownika. Poza fizycznymi czujnikami gogle wyposażone są w specjalny algorytm przewidywania, który na podstawie danych z czujników jest w stanie przewidzieć ruch użytkownika w niedalekiej przyszłości i na ich podstawie tworzy płynne i bardziej responsywne doświadczenie wirtualnego świata [4]. Wszystkie te przyrządy pozwalają przenieść ruch użytkownika do wirtualnego świata i wchodzić w interakcję z wirtualnymi elementami. Wirtualna rzeczywistość może także stanowić środowisko do analizy i diagnozy poszczególnych użytkowników [5]. Niniejszy artykuł proponuje narzędzie do pozyskiwania danych o użytkowniku VR oraz prezentuje ich użyteczność.

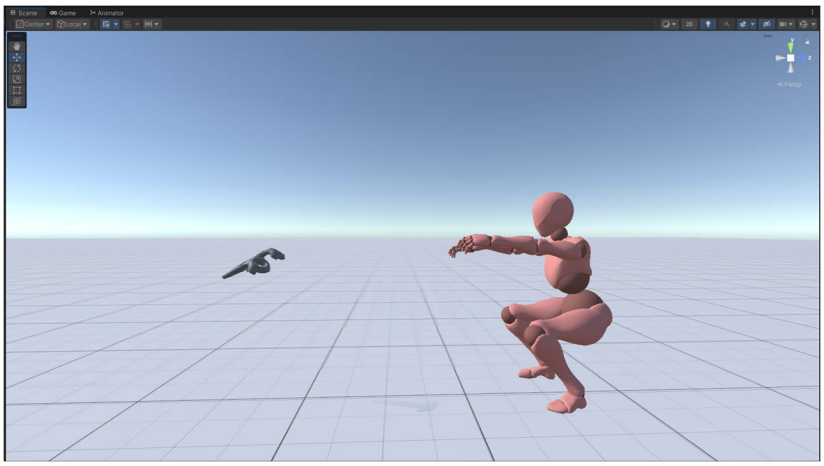
2. Rejestracja ruchu w środowisku VR

Aplikacja VR Recorder została stworzona, aby wykorzystać potencjał płynący z parametrów, które udostępniają nam okulary i kontrolery, takie jak np. pozycja, rotacja, wciśnięcie przycisków (ang. grip / thumbstick). Zwykły kontroler udostępnia nam tylko przyciski, na wciśnięcie których możemy wywołać daną funkcję. Dzięki śledzeniu ruchu użytkownika to samo oprogramowanie może zostać wykorzystane w znacznie szerszym spektrum takim jak na przykład rozpoznawanie schorzeń fizycznych bądź psychicznych [5], diagnostyka danych lub rozpoznawanie ruchu z wykorzystaniem modelu sztucznej inteligencji. Zaproponowane narzędzie umożliwia rejestrowanie ruchu w postaci szeregu czasowego zawierającego pozycję i stan poszczególnych elementów. Stan jest próbkowany ze stałą zadaną częstotliwością. Uzyskane dane można mapować na obiekt odwzorowując wykonany ruch. Narzędzie zostało wykonane jako moduł w ramach środowiska UNITY [6]. Dzięki takiemu podejściu moduł można zaimportować do dowolnego projektu (gry czy aplikacji) poprzez podpięcie pod odpowiedni obiekt w scenie. Uruchomienie nagrywania może odbywać się w sposób automatyczny (wyzwalany zdarzeniem) lub manualnie poprzez kombinację klawiszy. Schemat blokowy przedstawiający działanie aplikacji został przestawiony na rys. 2.

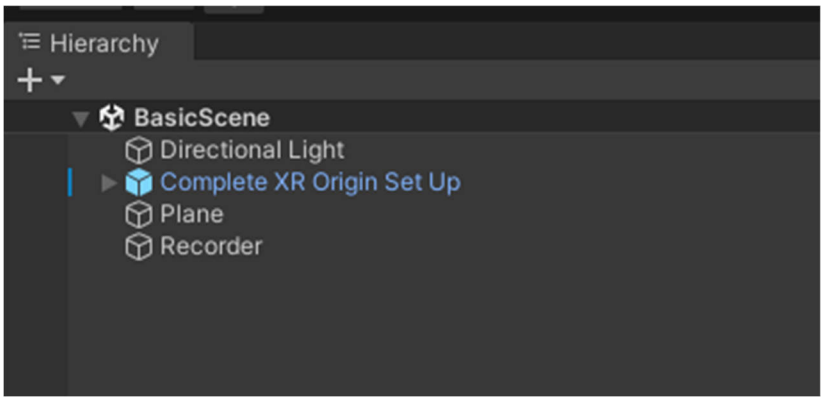


Rysunek 2. Schemat blokowy procedury nagrywania

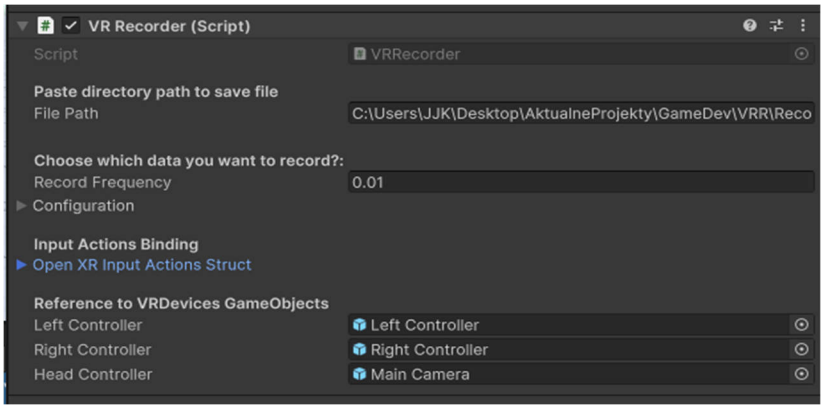
Moduł dzięki zastosowanej biblioteki XR wspiera szeroką gamę urządzeń (rys. 1). W tym artykule do rejestracji ruchu został wykorzystany zestaw VR Meta Quest 2. Na potrzeby prezentacji została utworzona scena z kontrolerem XR Origin wchodzącym w skład biblioteki open XR oraz obiekt „Recorder", jako autorska implementacja modułu „VRRecorder" oraz modelem wykonującym przysiad po którym należy naśladować. Scena wraz z ustawieniami przedstawiono na rys. 3 – 5.



Rysunek 3. Scena w Unity



Rysunek 4. Okno hierarchii z edytora Unity



Rysunek 5. Inspektor obiektu Recorder

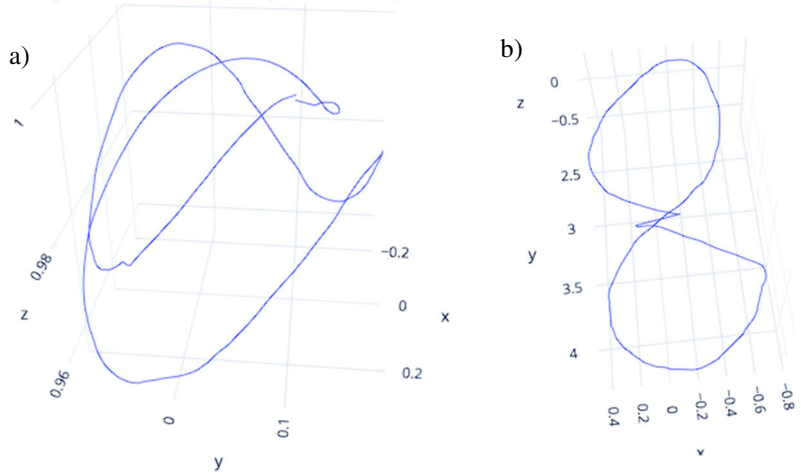
W inspektorze na rys. 5 przedstawiono elementy konfiguracyjne aplikacji nagrywającej. Są nimi:

- **File Path** – ścieżka zapisu plików z nagrywarki. Program zapisuje plik w formacie „record-rok-miesiąc-dzień-godzina-minuta-sekunda" i z rozszerzeniem **.csv**.
- **Record Frequency** – częstotliwość z jaką mają być zapisywane kolejne punkty parametrów.
- **Configuration** – tutaj możemy zaznaczyć, które dane chcemy przechwytywać.
- **OpenXR Input Action Struct** – struktura referencji do tzw. „Input Action". W Unity ten element pozwala podpinać pod konkretną akcję przycisku funkcję, którą chcemy żeby się wykonała. Możemy w taki sposób np. pod przyciskiem menu podpiąć otwarcie okna menu, ale my wykorzystamy „Input Action" do pobierania informacji o stanie danego elementu kontrolera, czyli np. czy jest wciśnięty przycisk i jak mocno.
- **Reference to VR Devices** – tutaj podpinamy obiekty ze sceny które reprezentują urządzenia, żebyśmy mogli zapisywać ich pozycję i rotację.

Elementy konfiguracyjne zostały zaprojektowane aby w najmniejszym stopniu obciążać użytkownika. Aplikacja ma z założenia nie obciążać nadmiernie aplikacji głównej oraz śledzić ruchy użytkownika oraz wybranych obiektów.

Proces rejestracji danych

Proces rejestracji rozpoczyna się automatycznie lub manualnie. Dane rejestrowane są z zadaną częstotliwością i zapisywane do pliku. Ze względu na specyfikę zestawów VR, uzyskiwane dane należy przetworzyć. Jest to związane z faktem, że uzyskane dane nie odwzorowują wprost ruchu wykonanego przez użytkownika we współrzędnych kartezjańskich. Na rys. 6 a przedstawiono surowe dane uzyskane przez wykonanie ósemki w powietrzu (próbkowanie 1000 na sekundę).



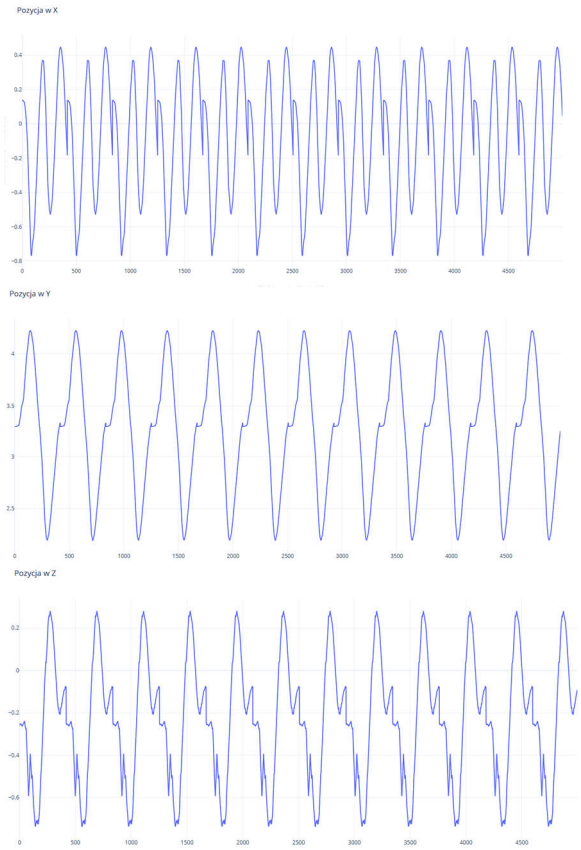
Rysunek 6. Ruch ręki przy narysowaniu 8 w powietrzu a) dane z urządzenia b) dane po przetworzeniu

Przejście do ww. współrzędnych wymaga dokonanie serii przekształceń obejmujących skalowanie pozycji, skalowanie rotacji i skalowanie wysokości

(rys. 6 b). Ponadto w momencie podpinania referencji urządzeń do narzędzia należy zwrócić uwagę, żeby obiekty kontrolerów były podpięte w scenie bezpośrednio pod goglami, przez co pozycja i rotacja staje się uzależniona od rodzica (gogli) tak jak jest w rzeczywistości. Wykonanie przekształcenia umożliwia prostszą analizę ruchu i rejestracje jego powtarzalności (okresowości). Rys. 7. przedstawia ruch ręki zrzutowany na osie X, Y oraz Z.

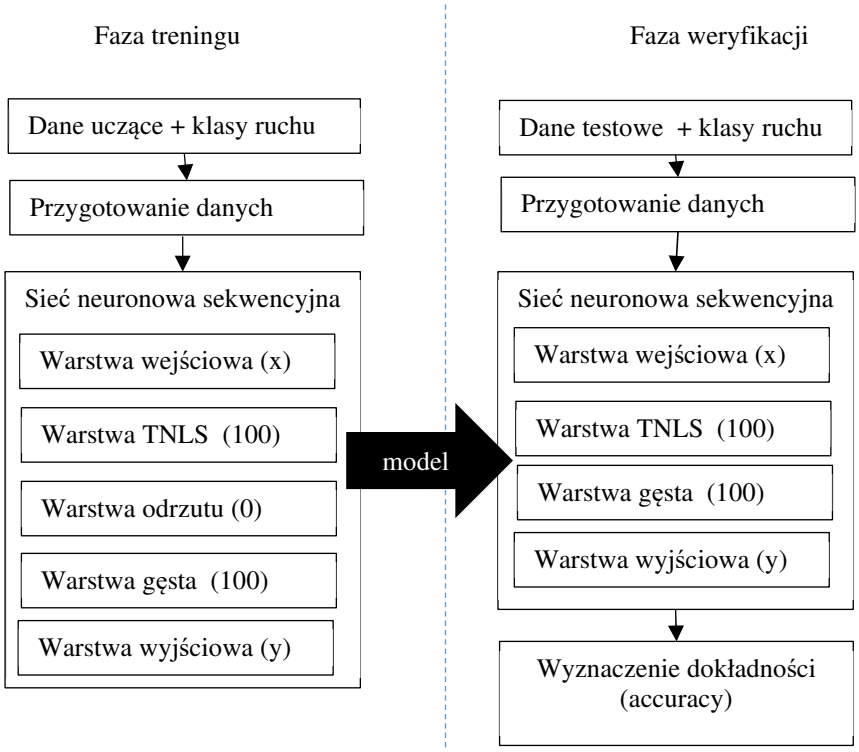
3. Model sieci neuronowej do klasyfikacji ruchu ciała w VR.

Przedstawione narzędzie, przy użyciu manipulatorów oraz gogli umożliwia pozyskiwanie danych o ruchu kończyn górnych oraz głowy z dużą częstotliwością próbkowania. W przypadku kończyn dolnych aktualnie wykorzystywana technologia nie śledzi ich bezpośrednio, jednak ich ruch może być wykryty pośrednio [7]. Do wykrywania ruchu kończyn dolnych mogą być stosowane różne algorytmy [8]. W tej pracy została zastosowana rekurencyjna sieć neuronowa (Recurrent Neural Networks - RNN), która umożliwia analizę szeregów czasowych.



Rysunek 7. Ruch ręki rzutowany na osie X,Y oraz Z

W zaproponowanej implementacji wykorzystano komórki typu Long Short-Term Memory (LSTM). W porównaniu do klasycznej sieci neuronowej, podejście umożliwia klasyfikację opartą na surowych (znormalizowanych) danych. W artykule zostanie zastosowana struktura dostosowana do danych pozyskiwanych z zestawu VR. Jej struktura sieci (przedstawiona na rys. 8) została dostosowana do charakterystyki danych wejściowych oraz rozpoznawanych klas ruchu.



Rysunek 8. Trenowanie oraz weryfikacja modelu sieci rekurencyjnej

Dane podawane na sieć są nieprzetworzone i podawane w postaci sekwencji na jej wejście. Ze względu na wybór danych o pozycji głowy oraz rąk z gogli i manipulatorów VR, wykorzystywane są dane z wideo detekcji, akcelerometru i żyroskopu, które są traktowane jako szeregi czasowe. Szeregi podzielone na sekwencje przy użyciu przesuwnych okien o stałej długości *w*. W celu uniezależnienia wyników od pozycji początkowej użyto filtru przyrostowego. Brane są pod uwagę jedynie wzrosty wartości położenia oraz rotacji. Proponowany model RNN przetwarza wiele wektorów wejściowych i generuje pojedynczy wektor wyjściowy. W związku z tym wykorzystywana jest architektura "wiele do jednego" gdzie przyjmowane są szeregi czasowe wektorów cech (jeden wektor na krok czasowy) i przekształcamy je w wektor prawdopodobieństwa na wyjściu w celu klasyfikacji.
Model RNN (rys. 3.1) został zaadaptowany na bazie publikacji [8, 9], przy czym proces dostosowywania i liczba operacji wstępnego przetwarzania danych zostały dostosowane do danych. Liczbę epok treningu dostosowano do analizowanej krzywej

uczenia, a warstwa odrzutu (ang. Drop out) została zastosowania w celu uniknięcia przetrenowania modelu. Aktywność o najwyższej wadze jest rozpoznawana jako rzeczywista aktywność. Zmodyfikowane parametry użyte w badaniach przedstawiono w kodzie źródłowym biblioteki Tensor Flow Keras [10].

```
def eval_model(trainData, trainClass, testData, testClass):
    verbose =0   #ukryj komunikaty
    epochs = 80 #maksymalna liczba epok
    n_timesteps = trainData.shape[1] #długość sekwencji w
    n_features, trainData.shape[2] #liczba cech x
    n_outputs =  trainClass.shape[1] #liczba klas y
    model = Sequential() #sekwencyjny model
    model.add(LSTM(100,   #definicja warst
    input_shape=(n_timesteps,n_features)))
    model.add(Dropout(0.5))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(n_outputs, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])#metoda uczenia
    # trenowanie sieci
    model.fit(trainData, trainClass, epochs=epochs,
    verbose=verbose)
    # ocena dokładności modelu
    acc = model.evaluate(testData, testClass, verbose=0)
    return acc
```

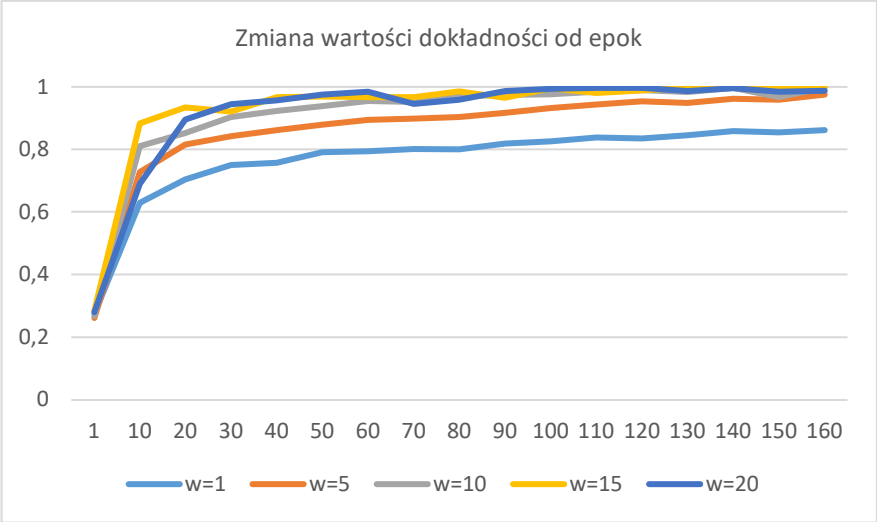
W przedstawionej strukturze aktywność o najwyższej wadze z warstwy wyjściowej jest rozpoznawana jako rzeczywista aktywność. Zmodyfikowane parametry użyte w badaniach przedstawiono w kodzie źródłowym biblioteki Tensor Flow Keras [10]. Na etapie treningu modelu wyłączono informację dla poszczególnych epok aby przyspieszyć działanie algorytmu. Został on włączony tylko do celów generowania postępów uczenia (verbose=1).

4. Analiza wyników modelu klasyfikacji ruchu

Narzędzie umożliwia śledzenie ponad 40 parametrów. Jednak do nauki modelu zostało wybrane x=18 z nich reprezentujących położenie głowy i dłoni w przestrzeni tj. 3 wartości pozycji oraz 3 wartości rotacji każdej z analizowanych części ciała. Pozostałe wartości opisujące głównie pozycje przycisków zostały pominięte. Zatem analizowany szereg czasowy pozyskany z urządzenia posiadał 18 cech oraz był próbkowany 10 razy na sekundę. Szereg czasowy następnie został podzielony w stosunku 70% do 30%, gdzie pierwsza wartość określa zbiór uczący a druga testowy. Każdy ze zbiorów został przetworzony za pomocą filtru przyrostowego. Następnie ciąg za pomocą okna został podzielony na sekwencje o długości okna w. Do oceny danego rozwiązania zastosowano cztery klasy gdzie oceniono ruch gracza obejmującego zaangażowanie dolnych partii ciała jak: bieg, marsz, przysiady czy stanie w miejscu. W sumie zdefiniowano y=4 analizowanych klas. W badaniu wykorzystano ponad 2800 wektorów uczących oraz ponad 800 wektorów testowych. Ich dokładana liczba zmieniała się w zależności od przyjętej wielkości okna.

4.1. Trenowanie modelu sieci

W celu poprawnego dobrania parametrów treningu sieci przeprowadzono próbne serie uczenia sieci oraz analizę otrzymanej krzywej nauki. Badanie miało na celu sprawdzenie czy przyjęta wartość graniczna epok nie jest zbyt mała. Krzywa uczenia została przedstawiona na rys. 9

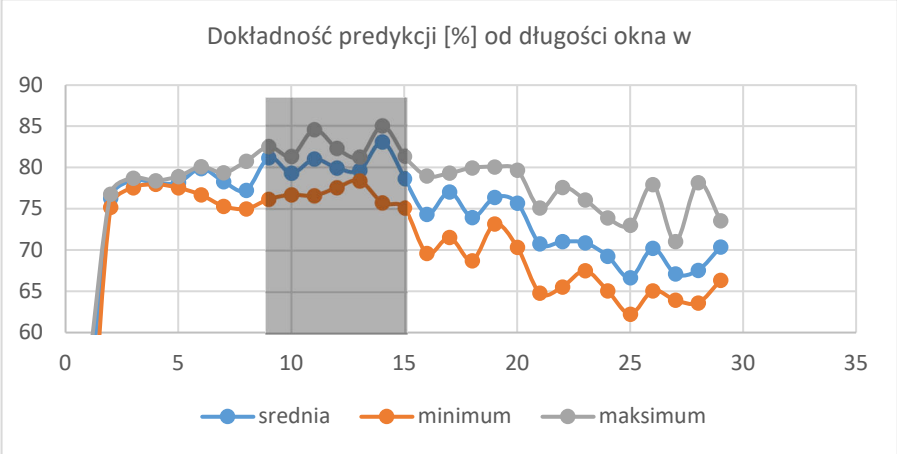


Rysunek 9. Krzywa uczenia sieci neuronowej

Wyniki wskazują, że zwiększenie liczby epok powyżej 120 nie wpływa na poziom wytrenowania sieci, dlatego wartość 120 przyjęto za wartość graniczną.

4.2. Weryfikacja modelu dla czterech klas.

Badania przeprowadzono dla różnych wielkości okien a tym samym różnej długości sekwencji wejściowej dla sieci. Przeanalizowano okna o długości w = [1, 30] odpowiadającej długości od 0,2 sekundy do 3 sekund. Wielkość graniczna została wybrana jako podwójna wartość okresu badanych czynności (umożliwiających wykrycie cykliczności). Wyniki uzyskanej dokładności (mierzonej miarą liczby poprawnie sklasyfikowanych przypadków do łącznej ich liczby) przedstawiono na rys. 10.



Rysunek 10. Wynik dokładności dla poszczególnych klas

Wyniki wskazują, że najlepsze wyniki i najlepiej dopasowany model otrzymujemy dla długości szeregów czasowych w przedziale od 9 do 15. Dla wartości w=3 można już zauważyć detekcję na stosunkowo wysokim poziomie. Jednak dla w=11 najlepiej odwzorowywana jest dynamika ruchu oraz jej cykliczność (dokładność równa 83%). W przypadku podziału na poszczególne klasy (rys. 11) można zauważyć pewne zależności.

w=4

	chód	bieg	przysiady	stanie
chód	226	2	7	0
bieg	12	169	53	1
przysiady	64	45	125	1
stanie	0	0	0	235

w=11

	chód	bieg	przysiady	stanie
chód	201	22	10	0
bieg	8	218	7	0
przysiady	55	64	114	0
stanie	0	0	0	233

Rysunek 11. Macierz błędów klasyfikacji

Bez względu na przyjęte okno najlepiej rozpoznawalnymi klasami są stanie w miejscu oraz chód. W przypadku biegu dokładność predykcji rośnie z wielkością okna, gdzie rozpoznawanie przysiadów nieznacznie maleje. W obu przypadkach dokładność jest na poziomie ponad 75%.

5. Podsumowanie

Zaproponowane narzędzie jest w stanie śledzić ruchy głowy i rąk z częstotliwością do 1000 razy na sekundę nie wpływając znacząco na szybkość działania głównej aplikacji. Narzędzie umożliwia dostosowanie częstotliwości próbkowania oraz zakres zbieranych danych do konkretnych potrzeb. Narzędzie może być aktywowane w reakcji na jakieś zdarzenie w aplikacji (automatycznie) lub manualnie poprzez wybraną sekwencje klawiszy. Analiza uzyskanych danych wskazuje na możliwość wyszukiwania zależności w ruchu w sposób analityczny a nawet monitorowanie czynności pośrednio za pomocą sieci neuronowej. Uzyskane wyniki wskazują, że

dane umożliwiają śledzenie czynności z udziałem kończyn dolnych z ponad 80% dokładnością dla okna w=11. Dalszy rozwój aplikacji będzie polegał na zastosowaniu mechanizmu do budowania matryc ruchowych (wzorców ruchowych) na potrzeby aplikacji VR a dane do analizy użytkowników oraz pośrednio poprawy imersji w grach opartych o tą technologię.

LITERATURA

1. DYMORA P. *et al* Deep recurrent neural networks for human activity recognition. 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* 1024 012099
2. MÜLLER, W., BOCKHOLT, U., LAHMER, A. et al. VRATS – Virtual-Reality-Arthroskopie-Trainingssimulator. Radiologe 40, 290–294 (2000).
3. KAVANAGH, S., LUXTON-REILLY, A., WUENSCHE, B., PLIMMER, B. (2017). A systematic review of Virtual Reality in education. Themes in Science and Technology Education, 10(2), 85-119.
4. ISLAM R., DESAI K. AND QUARLES J., Towards Forecasting the Onset of Cybersickness by Fusing Physiological, Head-tracking and Eye-tracking with Multimodal Deep Fusion Network, 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Singapore, Singapore, 2022, pp. 121-130, doi: 10.1109/ISMAR55827.2022.00026.
5. FLORIS T., Q., et al. Virtual reality in pediatrics, effects on pain and anxiety: A systematic review and meta-analysis update. Pediatric Anesthesia 32.12 (2022): 1292-1304.
6. JASON, J. et al. "Developing virtual reality applications with Unity." 2014 IEEE Virtual Reality (VR). IEEE, 2014.
7. MURAD, A.; PYUN, J.Y. Deep Recurrent Neural Networks for Human Activity Recognition. Sensors 2017, 17, 2556.
8. GUILLAUME C., LSTMs for Human Activity Recognition, 2016, https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition
9. BERNAŚ M., PŁACZEK, B., LEWANDOWSKI, M. Ensemble of RNN Classifiers for Activity Detection Using a Smartphone and Supporting Nodes. Sensors 2022, 22, 9451. https://doi.org/10.3390/s22239451
10. ABADI M., AGARWAL A., BARHAM P., ET. AL. TensorFlow: Large-scale machine learning on heterogeneous systems 2015. Software available from tensorflow.org.