# System wizyjnej analizy pozy człowieka wykorzystujący dane RGB-D

Michał Makoś

Marzec 2020

# 1 Opis systemu

## 1.1 Human Pose Classification

Tworzony system jest oparty o powszechnie znany problem określany jako *Human Pose Classification* czyli *Klasyfikacja pozy człowieka*. System ten ma za zadanie w czasie rzeczywistym wykrywać ludzi znajdujących się w badanym obszarze oraz wykrywać ich pozycje statyczne tj. stanie, siedzenie, leżenie; ale także pozycje dynamiczne tj. chodzenie czy też bieganie. Jako dane wejściowe przyjmuje obraz z kamery RGB-D, czyli standardowe dane kolorowych obrazów 2D wzbogacone o dodatkowy kanał *Depth* czyli kanał głębokości.

System ten w odróżnieniu od systemów znanych jako *Human Action Recognition* czyli *Rozpoznawanie czynności człowieka* skupia się na rozpoznawaniu podstawowych pozycji i ruchów, a nie na rozpoznawaniu czynności złożonych. Stąd takie zachowanie jak mycie naczyń powinno zostać sklasyfikowane jako stanie a np. granie w piłkę nożną jako bieganie.

## 1.2 Zastosowanie

Systemy klasyfikacji pozy człowieka mają zastosowanie przy monitorowaniu osób starszych i niepełnosprawnych. Szeroko stosowanym elementem takiego systemu jest wykrywanie upadków człowieka. Dzięki wykrywaniu nie tylko póz statycznych możemy rozszerzyć zastosowanie do wykrywanie różnych innych anomalii i niepożądanych w danym miejscu zachowań człowieka.

# 2 Istniejące rozwiązania

Większość istniejących rozwiązań klasyfikowania pozy opiera się wcześniejszym wykryciu tak zwanych keypointów, czyli charakterystycznych punktów na ciele człowieka, które połączone ze sobą tworzą szkielet. Problem tan określany jest jako *Human Pose Estiomation*. Klasyfikacja pozy następuje poprzez analizę otrzymanych punktów. W mojej pracy skupiać się będę na takim właśnie podejściu.

Istnieją klasyczne metody rozpoznawania póz człowieka, które nie wykorzystują sieci neuronowych. Takie rozwiązanie zostało przedstawione w artykule [9]. Autorzy proponują przedstawienie szkieletu człowieka jako szereg danych opisujących położenie, orientację oraz wzajemne relacje poszczególnych punktów. Tak przygotowane dane są następnie klasyfikowane przy użyciu klasyfikatora SVM.

Większość rozwiązań bazuje na rekurencyjnych sieciach neuronowych (RNN) z długoterminową pamięcią (LSTM). Takie rozwiązanie zostało użyte w systemie ActionXPose [5]. System ten wykrywa złożone czynności człowieka takie jak granie w piłkę czy mycie naczyń dlatego adekwatne jest zastosowanie sieci neuronowej z długoterminowym profilem pamięci.

Mój system nie potrzebuje jednak takich sieci z racji tego, że do wykrycia dynamicznych, ale nie złożonych póz nie ma potrzeby korzystania z długoterminowej pamięci. Wystarczy analizować kilkanaście lub kilkadziesiąt klatek, aby sklasyfikować pozę człowieka. W tym celu można w dość niekonwencjonalny sposób skorzystać ze splotowych sieci neuronowych (CNN). Takie rozwiązanie zostało przedstawione w systemie PoTion [12] oraz w artykule [3]. W mojej pracy będę bazował głównie na rozwiązaniu przedstawionym w artykule [3].

# 3 Zastosowane rozwiązanie

## 3.1 Human Pose Estimation

W zastosowanym przeze mnie rozwiązaniu klasyfikacja pozy bazuje na wykrytym szkielecie człowieka czyli problemie znanym jako *Human Pose Estimation*. Ten etap realizowany będzie przy pomocy biblioteki OpenPose. Biblioteka ta jako dane wejściowe przyjmuje stream z kamery RGB (2D) i zwraca dla każdego wykrytego człowieka współrzędne punktów jego szkieletu (niekoniecznie pełnego). Z racji tego, że biblioteka OpenPose działa na obrazach 2D, konieczne jest wcześniejsze wycięcie kanału głębokości z obrazu.

W pracy użyję następujących 15 keypointów w odpowiedniej kolejności: nos, szyja, prawe ramię, prawy łokieć, prawy nadgarstek, lewe ramię, lewy łokieć, lewy nadgarstek, punkt między biodrami, prawe biodro, prawe kolano, prawa kostka, lewe biodro, lewe kolano, lewa kostka.

## 3.2 2D do 3D

Z biblioteki OpenPose otrzymujemy współrzędne poszczególnych szkieletów w 2D. W przestrzeni dwuwymiarowej nie zawsze da się precyzyjnie określić pozę człowieka mając do dyspozycji tylko keypointy. Na przykład kiedy człowiek idzie prosto na kamerę nie unosząc zbytnio nóg, to analizując tylko kilka kroków człowieka możemy stwierdzić, że człowiek stoi w miejscu. Do rozwiązania tego problemu i poprawienia wyników klasyfikacji użyję dostępnego kanału głębokości poprzez proste zmapowanie keypointów z przestrzenie dwuwymiarowej na przestrzeń trójwymiarową.

Takie rozwiązanie ma jedną wadę: takie zmapowanie nie zawsze będzie odwzorowywało rzeczywiste położenie tych punktów. OpenPose jest w stanie estymować także zakryte

części ciała. Wobec tego, jeśli będziemy analizować pozę osoby stojącej bokiem do kamery i zmapujemy keypointy nogi zakrytej przez drugą nogę to po zmapowaniu otrzymamy takie współrzędne jakby obie nogi znajdowały się w tej samej odległości od kamery. Jednak proponowane przeze mnie rozwiązanie mapowania punktów na przestrzeń 3D ma na celu jedynie poprawić wyniki klasyfikacji. Jest to uzupełnienie poprzedniego problemu. W przypadkach kiedy ruch postaci może być niewykrywalny na przestrzeni 2D mapowanie może umożliwić wykrycie tego ruchu natomiast w przypadkach kiedy mapowanie 2D niedokładnie odwzorowuje rzeczywiste położenie punktów to ruch postaci może być dobrze widoczny w przestrzeni 2D. Taką sytuację możemy zaobserwować analizując człowieka idącego w kierunku kamery oraz bokiem do kamery.

Zastosuję jednak pewien próg - maksymalną odległość jaką może mieć dany keypoint od szkieletu gdyż może zdarzyć się sytuacja, że np. dłoń zostanie zasłonięta przez obiekt znajdujący się dużo bliżej kamery niż szkielet a wtedy taka anomalia może znacznie wpłynąć na wynik klasyfikacji.

## 3.3 Śledzenie szkieletu

Ponieważ system ma wykrywać także niestatyczne pozy człowieka tj. chodzenie a te mogą być trudne do odróżnienia analizując jedną klatkę należy zastosować analizowanie kilku klatek dla każdego analizowanego szkieletu. W tym celu zastosuję algorytm śledzenia szkieletu przedstawiony w artykule [3] rozszerzony o trzecią współrzędną keypointów. Algorytm dla każdego szkieletu $a$ z analizowanej klatki $n$ próbuje przyporządkować szkielet $b$ z klatki $n-1$. W tym celu oblicza podobieństwo $S_{ab}$ szkieletu $a$ do szkieletu $b$.

Na początku dla szkieletu $a$ wyznacza bounding box czyli wierzchołki prostopadłościanu obejmującego dany szkielet. Tutaj zastosuję dodatkowy warunek na odległość bounding box'ów branych pod uwagę, ponieważ dla dwóch szkieletów znajdujących się po dwóch stronach obrazu, daleko od kamery, prawdopodobieństwo, że zostaną one zaklasyfikowane jako ten sam szkielet jest bardzo małe, więc dla oszczędności obliczeń można pominąć porównywanie tych dwóch szkieletów.

Następnie liczy współczynnik $\Delta_a$ określający maksymalny dystans pomiędzy dwoma danymi punktami $i$ szkieletu, powyżej którego prawdopodobieństwo $S_{ab_i}$ należenia punktów do tego samego szkieletu będzie wynosiło zero oraz jednocześnie oznaczający proporcjonalność $S_{ab_i}$ do odległości $\delta_{ab_i}$ pomiędzy dwoma punktami:

$$\Delta_a = F\sqrt{x_a^2 + y_a^2 + z_a^2} \tag{1}$$

gdzie $x_a$, $y_a$, $z_a$ są wymiarami bounding box'a szkieletu $a$ a $F$ jest hiperparametrem, który zostanie dobrany podczas testowania systemu.

Następnie dla każdej pary keypoint'ów (tego samego typu $i$) liczy podobieństwo tego punktu ze szkieletów $a$ i $b$ jako:

$$S_{ab_i} = \begin{cases} 1 - \frac{\delta_{ab_i}}{\Delta_a} & \text{dla } \delta_{ab_i} < \Delta_a \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{2}$$

gdzie $\delta_{ab_i}$ jest odległością punktów $a_i$ i $b_i$ liczoną jako odległość euklidesowa $\delta_{ab_i} = ||a_i - b_i||$.

Podobieństwo $S_{ab}$ szkieletów jest liczone jako średnia średnia arytmetyczna wszystkich podobieństw punktów $S_{ab_i}$, przy czym punkty nie wykryte (bo nie zawsze mamy pełne szkielety) nie są brane pod uwagę do średniej. Szkielety $a$ oraz $b$ zostają zaklasyfikowane jako ten sam szkielet kiedy podobieństwo $S_{ab}$ jest największe spośród wszystkich podobieństw dla szkieletu $a$ oraz jeśli $S_{ab}$ jest większe niż ustalony threshold.

## 3.4 Splotowa sieć neuronowa - CNN

Ponieważ jako dane wejściowe otrzymaliśmy współrzędne wielu punktów z wielu klatek (bo nie stosujemy sieci z pamięcią), a splotowe sieci neuronowe (CNN) w ostatnich latach bardzo dobrze się rozwinęły dając dobre rezultaty, to mając dane szkieletu śledzonego na przestrzeni wielu klatek możemy sklasyfikować jego pozę przy pomocy niekonwencjonalnego wykorzystania CNN. Pomysł przedstawiony w artykule [3] polega na mapowaniu danych wejściowych z postaci współrzędnych w przestrzeni trójwymiarowej punktów z wielu klatek na obraz RGB w następujący sposób:

- kolejne klatki śledzonego szkieletu są reprezentowane na obrazie jako kolumny
- kolejne keypointy danego szkieletu w danej klatce są reprezentowane jako kolejne wiersze kolumny dla danej klatki
- współrzędne keypointu są reprezentowane jako odpowiedni kolor piksela w kolumnie oraz wierszu odpowiadającemu danemu punktowi z danej klatki.

Kolor RGB określany jako kolejne składowe kolorów czerwonego, zielonego oraz niebieskiego przedstawia kolejne współrzędne $x$, $y$ oraz $z$ danego punktu zrzutowane na wartość z przedziału $[0, 1]$.

## 3.5 Architektura sieci

Początkowo zostanie zaimplementowana nieduża sieć zaproponowana w artykule [3] bazująca na sieciach VGG, charakteryzujących się małymi filtrami 3x3 z minimalnym przesunięciem oraz operacji MaxPooling co 3 sieć. Nasze obrazy wejściowe będą jednak dużo mniejszych rozmiarów, bo na przykład dla 16 keypointów i 16 klatek będzie to obraz 16x16 podczas gdy dane wejściowe w sieci VGG mają rozmiar 224x244. Dodatkowo ilość klas będzie zdecydowanie mniejsza, dlatego możemy zastosować mniejszą wersję tej sieci:

- 6 warstw splotowych z filtrem 3x3 i przesunięciem 1
- po 2 i 4 warstwie operacja MaxPooling 2x2 aby zmniejszyć wymiary map
- po 6 warstwie operacja GlobalAveragePooling - zastosuję operację global pooling zamiast operacji flatten, ponieważ nie potrzebujemy dużej ilości klas, dodatkowo average pooling zapobiega przetrenowaniu sieci
- na końcu warstwa w pełni połączona
- jako funkcja aktywacji zostanie użyta funkcja ReLU.

# 4 Zbiór danych

Zbiór danych powinien składać się z sekwencji video w formacie RGB-D wraz z anotacją odpowiedniej pozy. Najlepiej jakby posiadały też anotacje keypointów w 2D lub 3D, wtedy można pominąć przetwarzanie kanałów RGB video lub całości video.

Nie udało mi się znaleźć zbioru danych, który odpowiadałby powyższym wymaganiom oraz założeniom projektowym. Zbiory danych odpowiadające założeniom, czyli posiadające klasy takie jak stanie, bieganie, chodzenie, leżenie itd. są jedynie w przestrzeni 2D. Przykładem takiego zbioru jest zbiór HMDB [7] oraz zbiór JHMDB [6] (Joint HMDB) będący rozszerzeniem zbioru HMDB o anotacje keypointów. Zbiory danych w 3D np. Human3.6M [8] oraz NET RGB+D [1] nie do końca odpowiadają założeniom projektowym gdyż składają się głównie z akcji takich jak jedzenie czy rozmawianie, a nie z póz. Zawierają one tylko pojedyncze kategorie interesujących nas póz.

W ramach tej pracy przygotuję własny zbiór uczący oraz testowy składający się w części z pojedynczych sekwencji z biblioteki Human3.6M [8] i NET RGB+D [1], które odpowiadają założeniom projektowym; a w części z własnoręcznie przygotowanego zbioru testowego.

## 4.1 Przygotowanie własnego zbioru

Przygotowanie własnego zbioru będzie składało się z etapów:

- Nagranie sekwencji video z danymi pozami oraz podzielenie sekwencji na kategorie.
- Uzyskanie estymowanych keypointów przy pomocy biblioteki OpenPose i zmapowanie ich na przestrzeń 3D.
- Przekonwertowanie współrzędnych keypointów poszczególnych klatek na obrazy gotowe do trenowania sieci.

W przypadku danych z wyżej wymienionych gotowych zbiorów konieczne jest zastosowanie tylko ostatniego etapu przygotowania zbiorów.

# 5 Technologie

- Python - system zostanie napisany w języku Python 3.
- TensorFlow + keras - biblioteka, w której zostanie zaimplementowany model sieci neuronowej.
- OpenCV - biblioteka ta zostanie użyta do wszystkich operacji na obrazach oraz do wyświetlania wyników. Jest ona wyposażona w moduł do obsługi obrazów RGB-Depth.
- OpenPose - biblioteka, za pomocą której zostanie zrealizowany moduł wykrywania szkieletu człowieka.

# 6 Kod źródłowy

Repozytorium GitHub: https://github.com/mmakos/HPC

# Literatura


[1] T.-T. N. G. W. Amir Shahroudy, Jun Liu. Ntu rgb+d: A large scale dataset for 3d human activity analysis. 2016.

[2] A. F. J. D. Christoph Käding, Erik Rodner. Fine-tuning deep neural networks in continuous learning scenarios. 2017.

[3] C. C. Dennis Ludl, Thomas Gulde. Simple yet efficient real-time pose-based action recognition. 2019.

[4] T. D. Evan SHelhamer, Jonathan Long. Fully convolutional networks for semantic segmentation. 2016.

[5] Y. L. L. S. Federico Angelin, Zeyu Fu, S. M. Naqvi. A novel 2d multi-view pose-based algorithm for real-time human action recognition. 2018.

[6] S. Z. C. S. H. Jhuang, J. Gall, M. J. Black. Towards understanding action recognition. Dec. 2013.

[7] E. G. T. P. H. Kuehne, H. Jhuang, T. Serre. Hmdb: A large video database for human motion recognition. 2019.

[8] C. Ionescu, D. Papava, V. Olaru, C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[9] J. G. Martin Garbade. Handcrafting vs deep learning:an evaluation of ntraj+ features for posebased action recognition. 2016.

[10] A. B. P. L. Rizlene Raoui-Outach, Cecile Million-Rousseau. Deep learning for automatic sale receipt understanding. 2017.

[11] A. B. P. L. Rizlène Raoui-Outach, Cécile Million-Rousseau. Lecture automatique d'un ticket de caisse par vision embarquée sur un téléphone mobile. 2016.

[12] J. R. C. S. Vasileios Choutas, Philippe Weinzaepfel. Potion: Pose motion representation for action recognition. 2018.