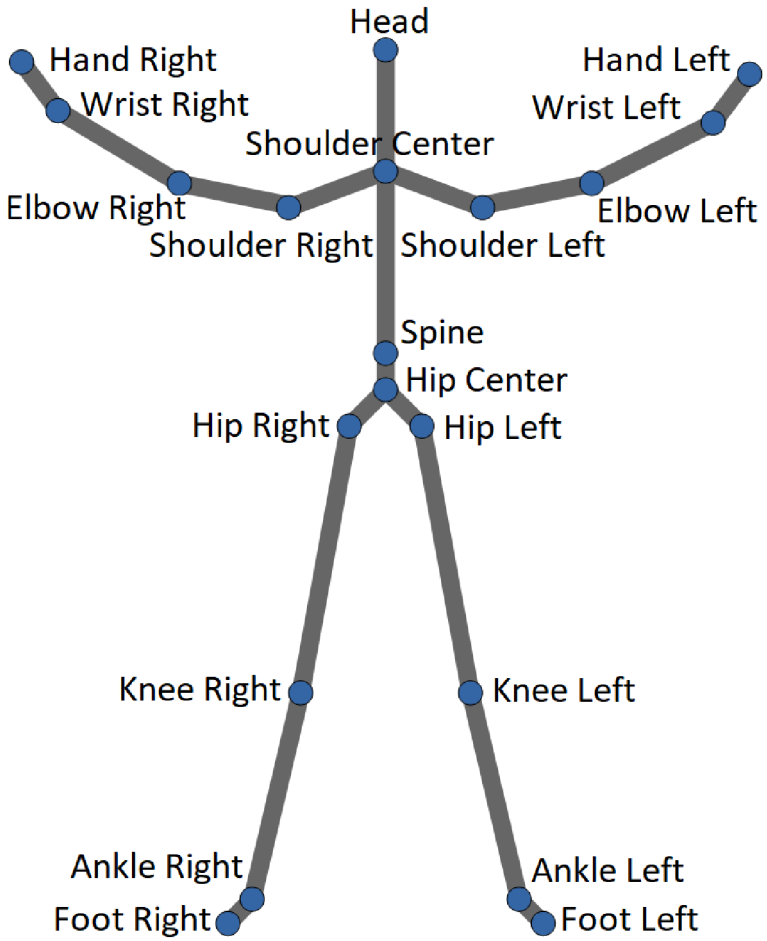
Rozpoznawanie akcji wykonywanych przez ludzi m.in. na podstawie danych szkieletowych

Wykład w ramach przedmiotu „Metody rozpoznawania obiektów i analizy ruchu”

Dane szkieletowe postaci ludzkiej

- Dane składają się ze współrzędnych, w przestrzeni 3D lub na płaszczyźnie 2D, przegubów i innych punktów charakterystycznych (węzłów, ang. joint) postaci ludzkiej.

- Rysunek przedstawia przykładowy szkielet z zestawem węzłów, który można pozyskać z kamery Kinect.



Sposoby pozyskiwania danych szkieletowych postaci ludzkiej

Dobre kamery, niestety już nie wspierane.

Kinect (Xbox 360)

Kinect (Xbox One)

Azure Kinect DK

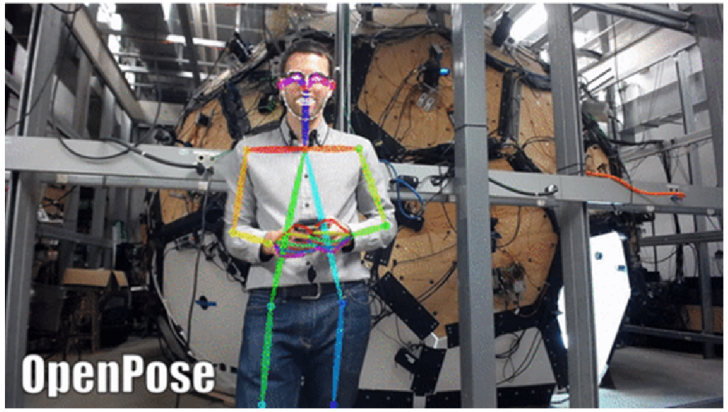
Nowsze urządzenia, wspierane do dziś.

Stereolabs ZED 2i

Ultraleap Stereo IR 170
(szkielety samych dłoni)

Sposoby pozyskiwania danych szkieletowych postaci ludzkiej, c.d. – biblioteka OpenPose

- Biblioteka w językach C++ oraz Python do wyznaczania szkieletów 2D (postaci ludzkiej wraz z palcami dłoni i mimiką twarzy) na podstawie obrazów kolorowych: *https://github.com/CMU-Perceptual-Computing-Lab/openpose*
- Ostatnia wersja biblioteki jest z roku 2020.



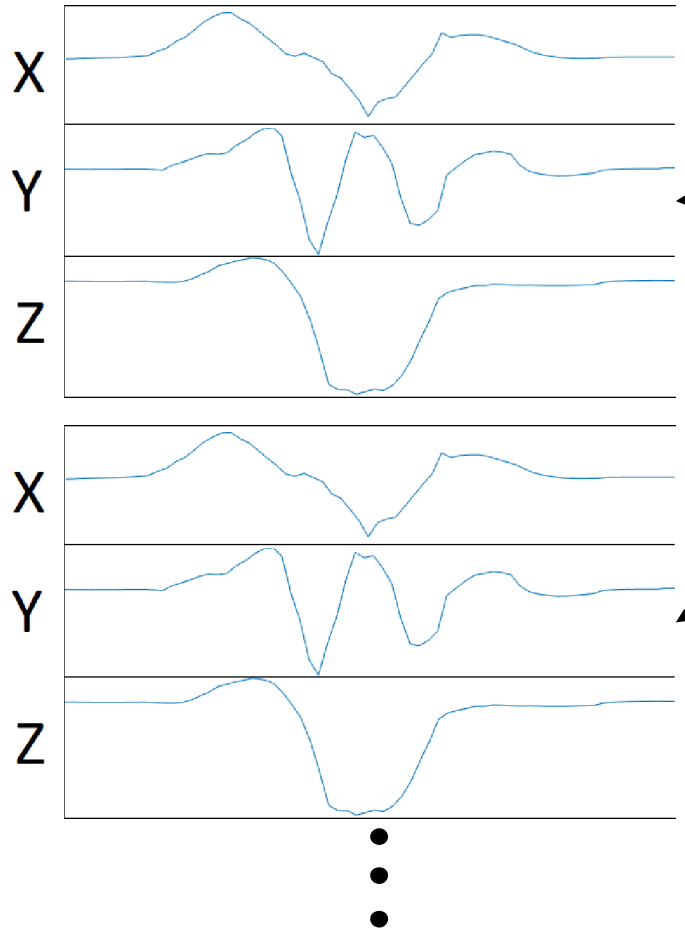
Sposoby pozyskiwania danych szkieletowych postaci ludzkiej, c.d. – biblioteka MediaPipe

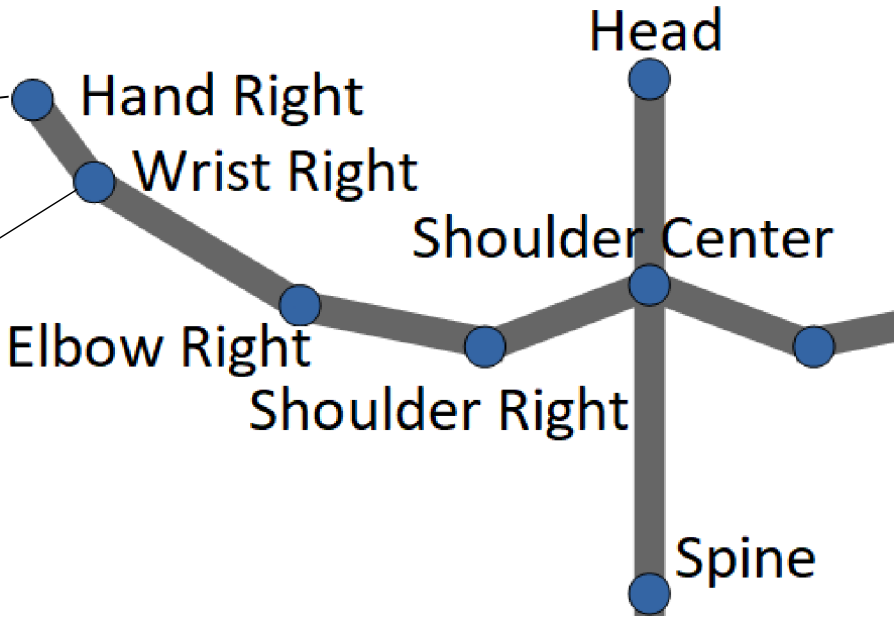
- Biblioteka w językach C++, Python, Java (na Androidzie) oraz JavaScript (nieoficjalnie) do wyznaczania szkieletów 2D **lub 3D** (postaci ludzkiej wraz z palcami dłoni i mimiką twarzy) na podstawie obrazów kolorowych: *https://google.github.io/mediapipe*
- Ostatnia wersja biblioteki jest z roku 2023.

Sposoby pozyskiwania danych szkieletowych postaci ludzkiej, c.d. – inne biblioteki

- Istnieją również inne, mniej znane biblioteki do pozyskiwania szkieletów 2D. Wśród nich warto wyróżnić dwie, ze względu na to, że są oferują dużą szybkość detekcji szkieletów:
  - MoveNet (Python, JavaScript)
  - Lightweight OpenPose (Python, C++)

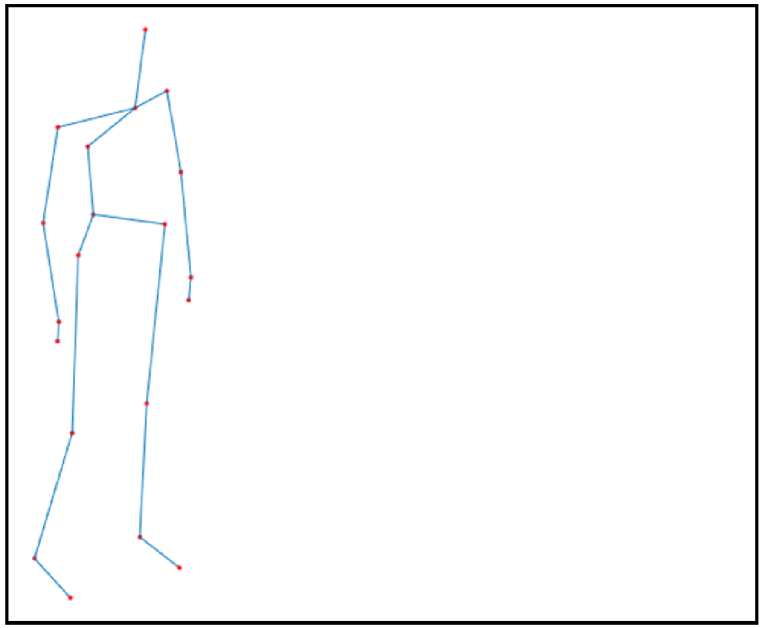
Akcje (gesty dynamiczne) – dane sekwencyjne

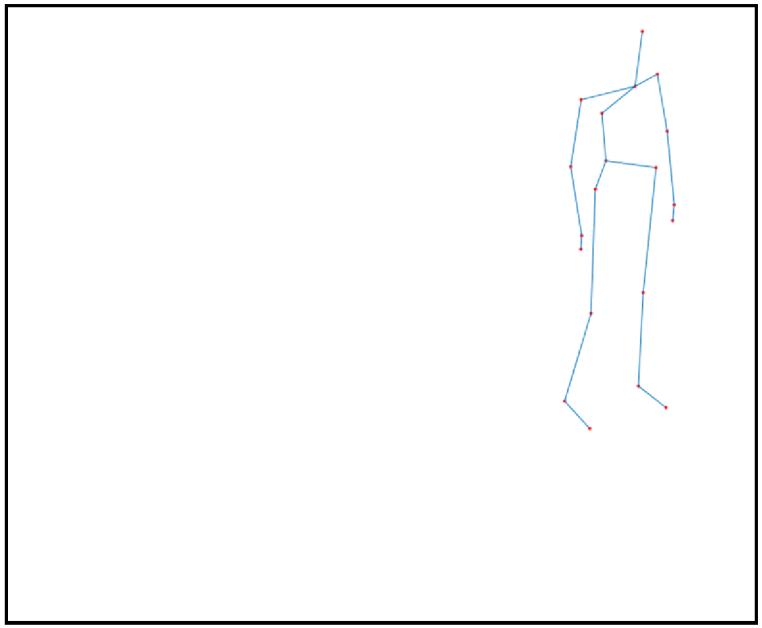




Zależność cech współrzędnych od położenia i rozmiaru szkieletu

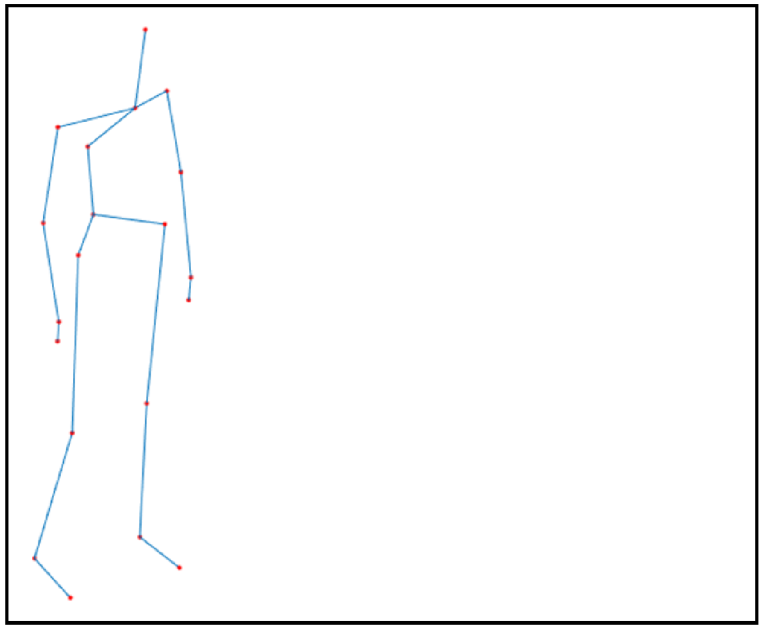
- Obrazki przedstawiają kadry kamery zawierające szkielet.
- Pozycje szkieletu w obu kadrach są identyczne (pozycja spoczynkowa).
- Szkielety różnią się jednak położeniem (względem kamery) i rozmiarem (wzrost osoby, odległość od kamery).

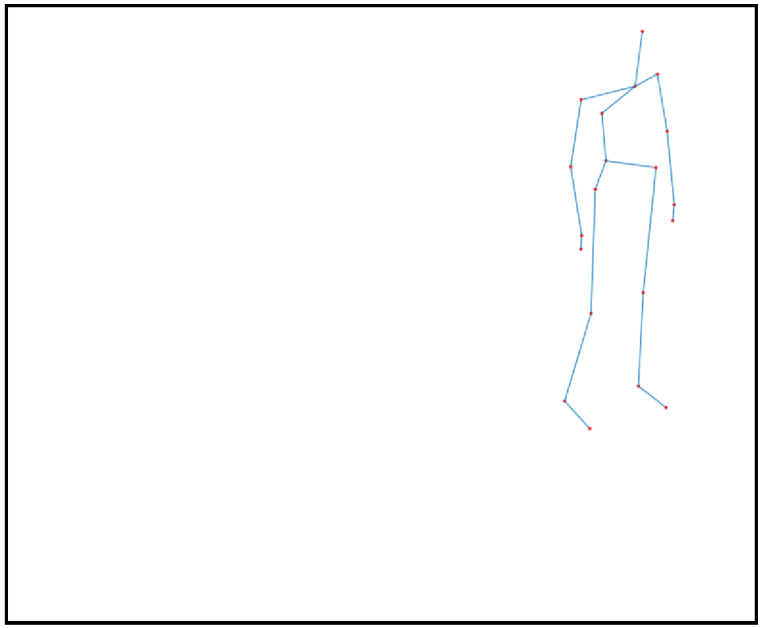




Zależność cech współrzędnych od położenia i rozmiaru szkieletu

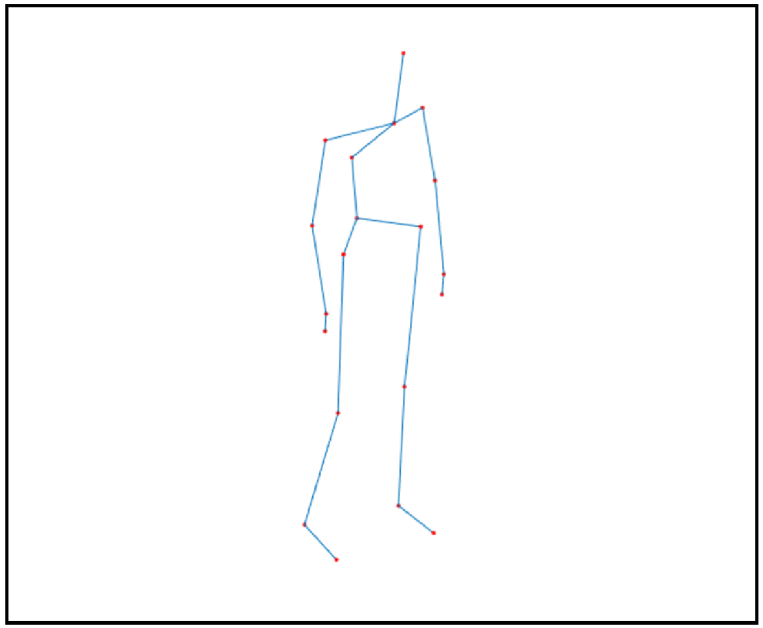
- Ze względu na różnice w położeniu i skali wektory cech współrzędnych uzyskane z tych szkieletów będą się znacząco różniły.
- Co należy zrobić ze szkieletami, żeby w obu przypadkach uzyskać podobne wektory cech?

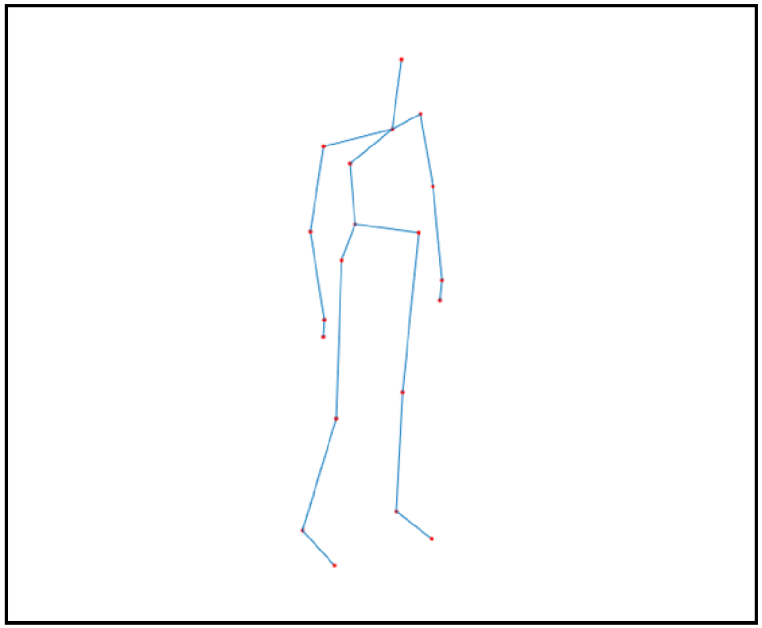




Zależność cech współrzędnych od położenia i rozmiaru szkieletu

- Ze względu na różnice w położeniu i skali wektory cech współrzędnych uzyskane z tych szkieletów będą się znacząco różniły.
- Co należy zrobić ze szkieletami, żeby w obu przypadkach uzyskać podobne wektory cech?





Uniezależnienie cech współrzędnych od położenia szkieletu

- Aby uniezależnić cechy od położenia szkieletu należy go przesunąć tak, aby jeden z węzłów środkowych (np. hip center – środek biodra) mieścił się zawsze w ustalonym punkcie (np. [0, 0, 0]).
- Czy należy przesuwać tylko pierwszą klatkę, czy wszystkie klatki?
- Odpowiedź: wszystkie, ale jeśli akcje uwzględniają przemieszczanie się, a nie tylko zmianę pozycji węzłów względem siebie, to pozycja środkowego węzła w kolejnych klatkach powinna być aktualizowana względem ustalonego punktu [0, 0, 0], a nie pozostać zawsze w tym punkcie.

Przykład (akcje uwzględniająca przemieszczanie się):

Współrzędna *x* węzła środkowego w kolejnych klatkach: [3.0, 3.5, 3.7, 3.8].

Kolejne klatki powinniśmy przesunąć tak, aby współrzędna miała wartości: [0.0, 0.5, 0.7, 0.8].

Nieprawidłowe przesunięcia: [0.0, 0.0, 0.0, 0.0], [0.0, 3.5, 3.7, 3.8].

Uniezależnienie cech współrzędnych od rozmiaru (skali) szkieletu

- Aby uniezależnić cechy od rozmiaru szkieletu należy go przeskalować do ustalonego rozmiaru, wspólnego dla wszystkich szkieletów.
- Jak wykonać takie skalowanie?
- Najlepiej podzielić każdą współrzędną wszystkich klatek przez odległość pomiędzy dwoma relatywnie długimi węzłami, np. spine (dolny koniec kręgosłupa) i shoulder center (środek barku).
- Wykorzystujemy fakt, że odległość pomiędzy sąsiednimi węzłami nie powinna się znacząco zmieniać.

Deskryptory szkieletów

- Cechami szkieletów nie muszą być koniecznie współrzędne węzłów. Na podstawie współrzędnych można obliczyć bardziej skomplikowane cechy.

- Przykładowe deskryptory, które umożliwiają obliczenie takich cech:
  1. Distance descriptor (deskryptor odległościowy)
  2. Bone pair descriptor (deskryptor par kości)

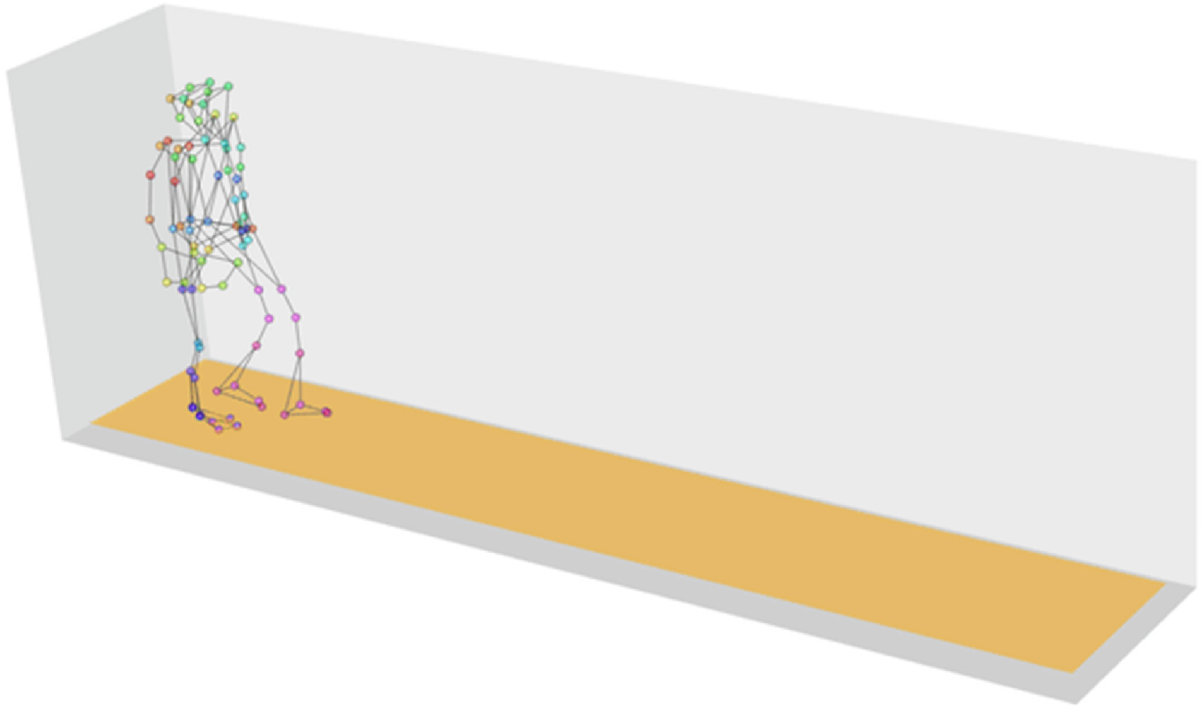
- Danymi wejściowymi powyższych deskryptorów są współrzędne węzłów pojedynczego szkieletu (pojedynczej klatki sekwencji). Współrzędne nie muszą być uniezależnione od położenia i rozmiaru, ponieważ deskryptory wykazują taką niezależność.

Kody źródłowe (wraz z przykładami zastosowania) w języku Matlab omawianych deskryptorów można pobrać ze strony: *http://vision.kia.prz.edu.pl/action/action.zip*

Klasyfikacja danych sekwencyjnych

- Standardowe klasyfikatory, takie jak SVM, drzewa decyzyjne, nie nadają się do rozpoznawania akcji, gestów, czynności (i większości innych rodzajów danych sekwencyjnych). Dlaczego?
- Ponieważ akcje mogą być wykonywane z różną szybkością, a więc zwykłe porównywanie cech z kolejnych klatek nie ma sensu.
- Czy rozwiązaniem może być zwykłe liniowe przyspieszenie (np. usunięcie co drugiej klatki) lub spowolnienie (np. zdublowanie każdej klatki), żeby wyrównać długość wszystkich klasyfikowanych sekwencji?
- Odpowiedź: Nie. Dlaczego?
- Ponieważ akcje mogą nie być wykonywane ze stałą prędkością. Może być obecne przyspieszenie lub spowolnienie (nagłe lub ciągłe).

Te same akcje wykonane z różną prędkością



Źródło: https://doi.org/10.1111/rssc.12276

Proponowane klasyfikatory danych sekwencyjnych

- **k-najbliższych sąsiadów** z odległością **DTW** (Dynamic Time Warping) – standardowy klasyfikator kNN, w którym zamiast klasycznych metryk użyto miarę podobieństwa DTW umożliwiającą porównanie dwóch sekwencji.
- **LDMLT** (LogDet divergence-based Metric Learning with Triplet constraints) – uczenie z metryką opartą na dywergencji z ograniczeniami trójek.
- **Delay Embedding** – skuteczny i szybki klasyfikator transformujący sekwencje do tzw. przestrzeni osadzania (ang. embedding space).
- Metody oparte o głębokie uczenie: sieci **LSTM** (Long Short-Term Memory) oraz **BiLSTM** (Bidirectional Long Short-Term Memory).
- **TSM** (Time Shift Module) – popularna w ostatnich latach metoda rozpoznawania akcji nie wymagająca podawania cech ani szkieletów (wprowadza się sekwencję obrazów), posiadająca elementy głębokiego uczenia.

Porównanie klasyfikatorów

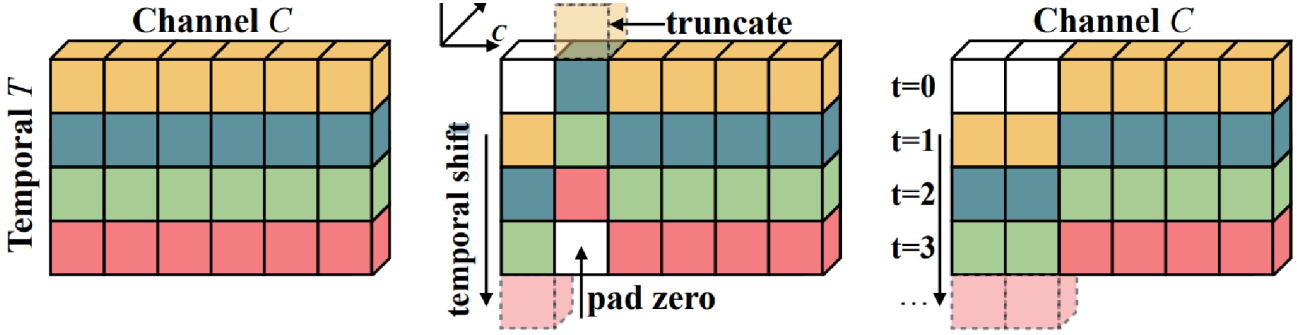
- **k-najbliższych sąsiadów** z odległością **DTW** – długi czas klasyfikacji, nie wymaga uczenia.
- **LDMLT** – krótki czas klasyfikacji, długi czas uczenia, wyniki lepsze od DTW.
- **Delay Embedding** – długi czas klasyfikacji, krótki czas uczenia, wyniki różne (czasem gorsze od DTW, czasem lepsze od LDMLT).
- Metody oparte o głębokie uczenie – zazwyczaj dobre wyniki, ale trzeba je odpowiednio dostroić (duża liczba parametrów) i wymagają większej ilości danych treningowych niż pozostałe wymienione metody.

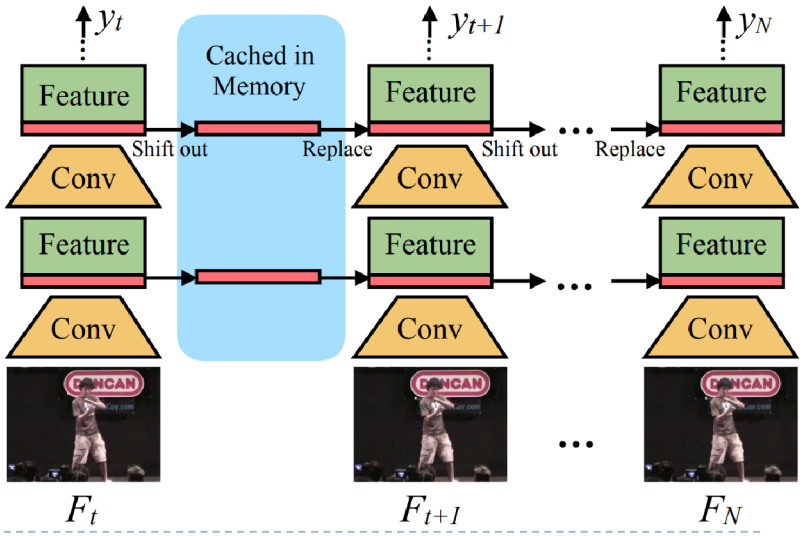
Przykładowe implementacje proponowanych klasyfikatorów w Matlabie

- **DTW** – funkcja *dtw*. Wymaga zainstalowania rozszerzenia Signal Processing Toolbox w Matlabie.
- **LDMLT** – *https://www.mathworks.com/matlabcentral/fileexchange/47928-ldmlt_multivariate_time_series_classification-zip*
- **Delay Embedding** – *https://github.com/ZZUTK/Delay_Embedding*
- Sieci **LSTM** oraz **BiLSTM** – *https://www.mathworks.com/help/deeplearning/ug/classify-sequence-data-using-lstm-networks.html*
- **DTW** oraz **LSTM/BiLSTM** są dostępne również w języku Python. LSTM/BiLSTM są w module TensorFlow. DTW w Pythonie będzie dotyczył ostatni slajd.

TSM – Temporal Shift Module

- Oprócz klasyfikatorów danych operujących na wyznaczonych cechach (np. węzłach szkieletów), w ostatnich latach opracowano również algorytmy rozpoznające akcje na podstawie sekwencji zwykłych obrazów.
- Wartą polecenia jest metoda **TSM** (Temporal Shift Module).
- Bazuje na głębokim uczeniu. Wykorzystuje operacje konwolucji i ideę algorytmu przesuwnego okna.





Źródło: https://doi.org/10.48550/arXiv.1811.08383

Implementacja TSM

- Algorytm jest zaimplementowany w Pythonie i można go pobrać wraz z przykładami użycia ze strony:

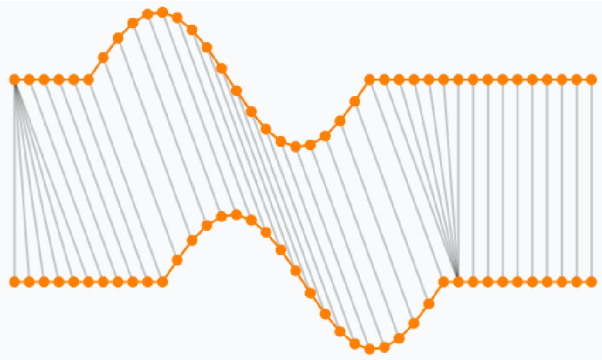
  https://github.com/mit-han-lab/temporal-shift-module

DTW

- DTW – ang. Dynamic Time Warping, dynamiczne dopasowanie czasowe, nieliniowa transformata czasowa.
- DTW jest metodą umożliwiającą porównywanie dwóch dyskretnych ciągów czasowych (przebiegów, danych liczbowych sekwencyjnych).
- Na wejściu DTW podaje się dwa szeregi czasowe, a na wyjściu metoda zwraca ich odległość (stopień niepodobieństwa).
- Szeregi czasowe porównywane metodą DTW mogą być wielowymiarowe (ang. multivariate time-series), czyli opisane przez więcej niż jedną zmienną.
- Oznacza to, że za pomocą DTW możemy porównać np. akcje ludzkie opisane przez więcej niż jedną cechę.

Na czym polega DTW?

- DTW dokonuje nieliniowej transformacji obu porównywanych przebiegów tak, aby odległość (np. euklidesowa, miejska) między nimi była minimalna.

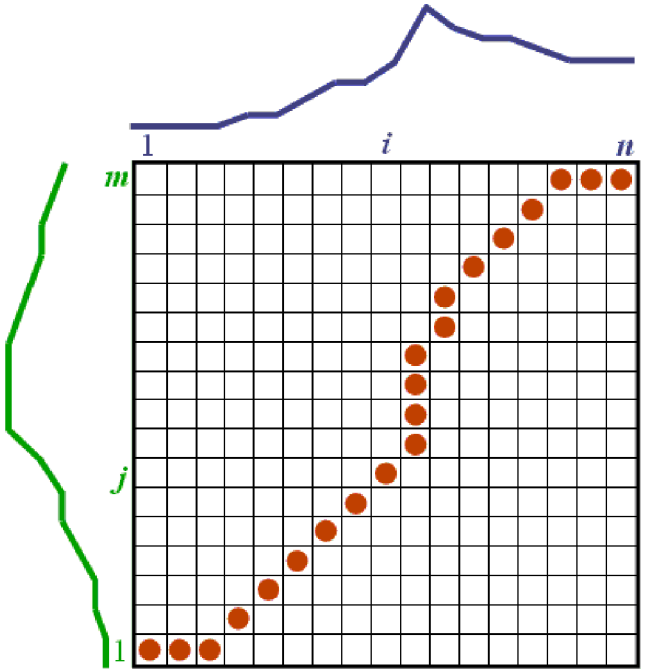


- Na powyższym rysunku widzimy dwa przebiegi, które zostały do siebie dopasowane. Linią przerywaną zaznaczono pasujące do siebie punkty obu przebiegów.
- Warto zwrócić uwagę, że niektóre pojedyncze punkty jednego wykresu zostały dopasowane do wielu punktów drugiego wykresu.

Źródło: Tavenard R., *An introduction to Dynamic Time Warping* - https://rtavenar.github.io/blog/dtw.html

Jak DTW wykonuje dopasowanie przebiegów?

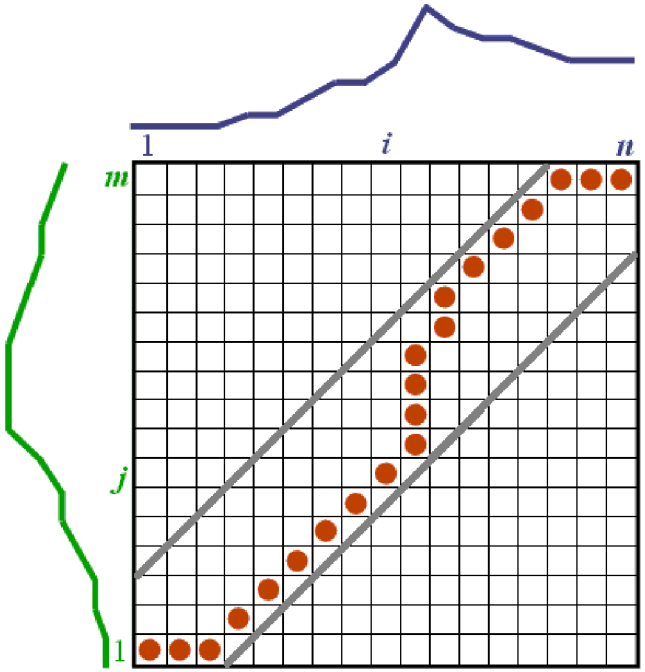
- Należy utworzyć tzw. macierz odległości lokalnych o wymiarach $n \times m$, gdzie n i m są długościami porównywanych przebiegów.
- Następnie tworzy się ścieżkę dopasowania (ang. warping path, czerwona ścieżka na rysunku).
- Ścieżka dopasowania powinna mieć jak najmniejszą sumę wartości (odległości), jednak muszą zostać spełnione trzy warunki:
  I. musi zacząć się w punkcie w lewym dolnym rogu i skończyć w punkcie w prawym górnym rogu,
  II. nie może mieć przerw,
  III. nie może się cofać (po żadnej z osi).



Źródło: https://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm

Jak DTW wykonuje dopasowanie przebiegów?

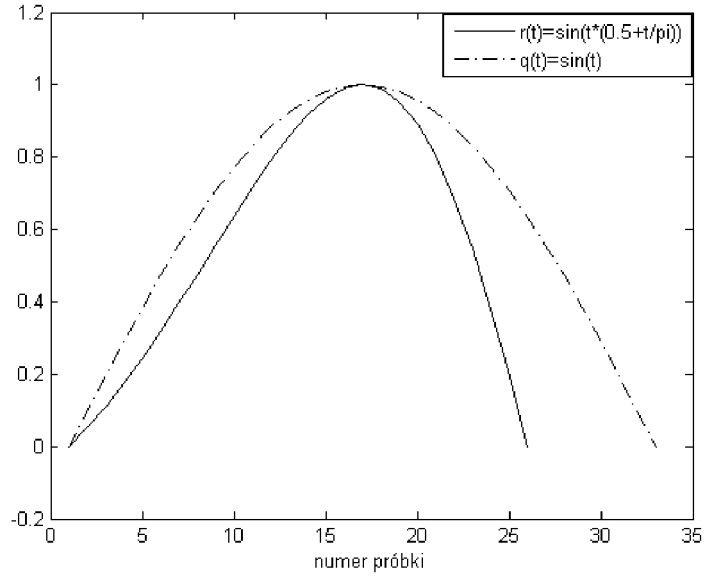
- Ścieżkę tworzy się za pomocą programowania dynamicznego (metoda optymalizacji).
- Aby uniknąć niepożądanych sytuacji, w których krótki fragment jednego przebiegu zostanie dopasowany do długiego fragmentu drugiego przebiegu, wprowadza się dodatkowe ograniczenie: rozmiar okna transformacji.
- Ścieżka musi mieścić się w granicach okna transformacji, które są zaznaczone szarymi liniami na rysunku.

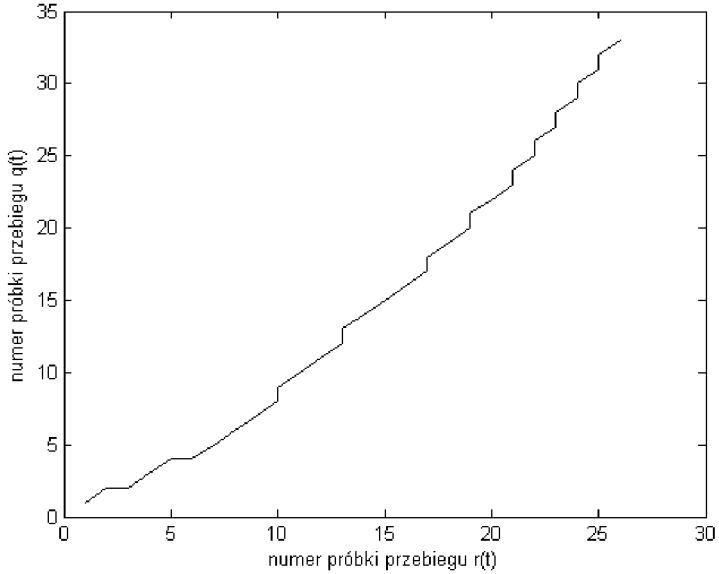


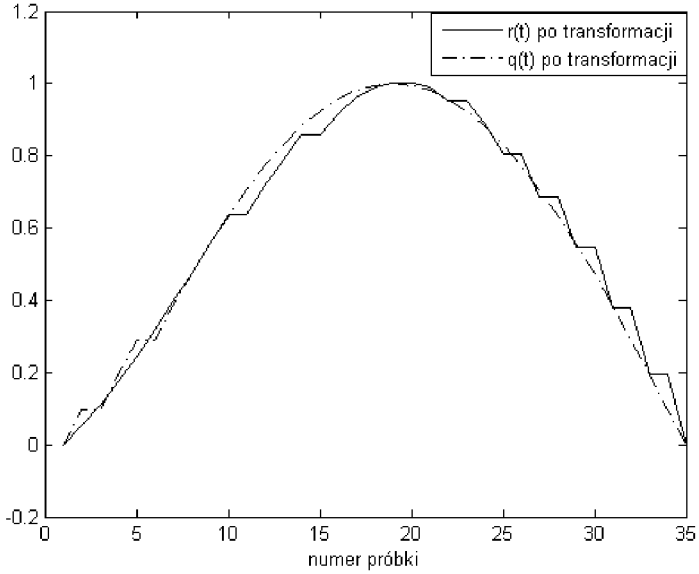
Na podstawie: https://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm

Przykład DTW

- Na pierwszym rysunku od lewej widzimy dwa porównywane przebiegi.
- Środkowy rysunek przedstawia utworzoną ścieżkę dopasowania.
- Ostatni rysunek przedstawia przebiegi po transformacji.







Źródło: M. Wysocki, T. Kapuściński, J. Marnik, M. Oszust, *Rozpoznawanie gestów wykonywanych rękami w systemie wizyjnym*. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów, 2011.

Czy DTW jest metryką?

- Metryka jest funkcją, która określa odległość dwóch elementów.
- Żeby funkcja mogła być metryką, musi spełniać trzy warunki:
  I. Identyczność nierozróżnialnych – odległość równa 0 występuje tylko w przypadku, gdy oba porównywane elementy są identyczne.
  II. Symetria – odległość między elementami A i B jest taka sama jak między elementami B i A.
  III. Nierówność trójkąta – odległość między elementami A i C nie może być większa niż suma odległości pomiędzy A i B oraz B i C.
- DTW spełnia warunki I i II, ale nie spełnia warunku III. Nie może być więc uznana za metrykę w ścisłym tego słowa znaczeniu.
- Nie przeszkadza to jednak w używaniu jej jako pseudometryki korzystając z klasyfikatora k-najbliższych sąsiadów.

Implementacja DTW w języku Python

- W Pythonie istnieje kilka modułów realizujących metodę DTW. Są one jednak znacznie wolniejsze od funkcji matlabowej *dtw*.
- Jeśli komuś zależy na szybkości, warto zastosować moduł *fastdtw* (*https://github.com/slaypni/fastdtw*). Metoda tam zastosowana jest szybką aproksymacją DTW.
- Autorzy *fastdtw* deklarują, że ma złożoność czasową i pamięciową O(N).
- Uwaga na parametr *radius*, który powinien odpowiadać rozmiarowi okna transformacji. Może nie zachowywać się tak jak powinien.