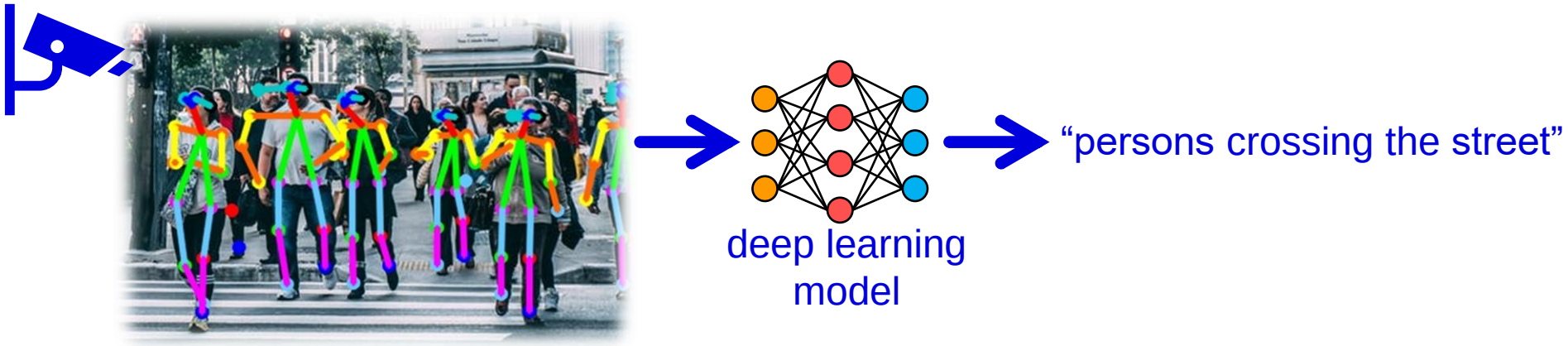


Research in Processing Human Motion from Video Data using Deep Learning

Jan Sedmidubský

Department of Machine Learning and Data Processing

Masaryk University

Brno, Czechia

Laboratory of Data Intensive Systems and Applications
disa.fi.muni.cz

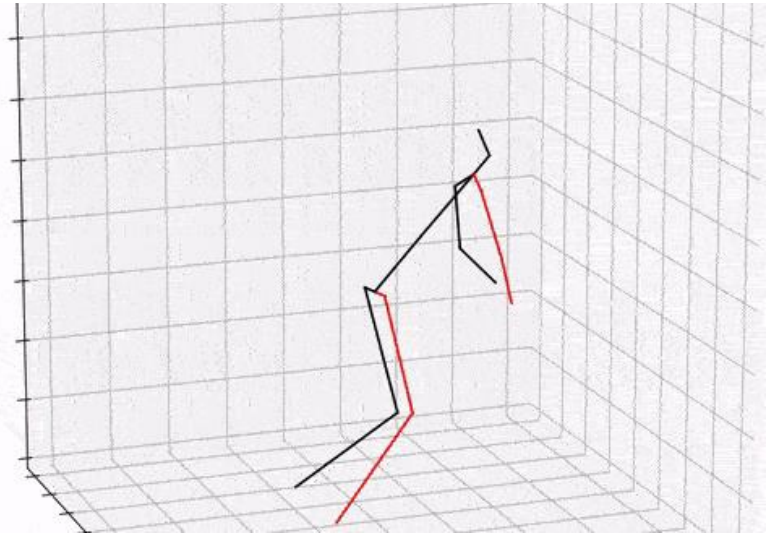
Digitization of human motion

- Skeleton-data representation
  - Simplified spatio-temporal representation of human motion
  - Sequence of 3D skeletons ~ a set of 3D trajectories of key body joints
  - Better structured and easier to store than video-based representation

Video-based representation

Source: https://blog.usejournal.com/3d-human-pose-estimation-ce1259979306

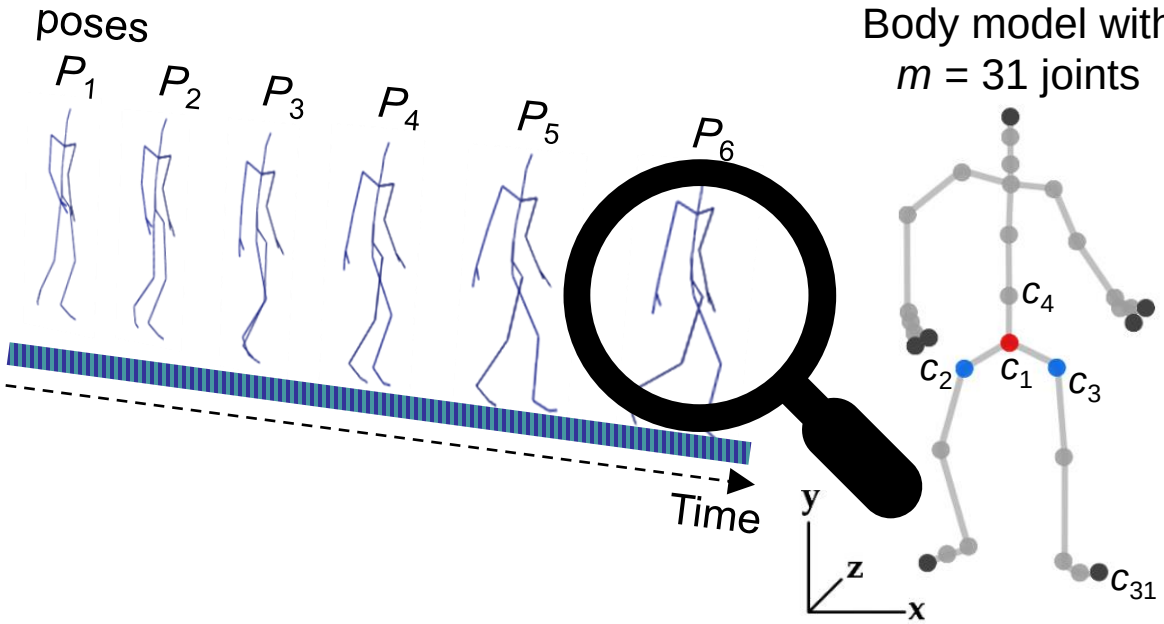
Skeleton-based representation



Skeleton data

Skeleton-based representation

- Sequence $S = (P_1, \ldots, P_n)$ of skeleton poses $P_1, \ldots, P_n$
  - Pose $P_i = (c_1, \ldots, c_m)$ consists of 2D/3D coordinates $c_1, \ldots, c_m$ of selected key body points, that usually correspond to significant joints
  - Bones are just “artificial lines” between pairs of key points



Matrix representation of 3D skeleton data

		P_1	P_2	P_3	P_4	P_5	…	P_n
c_1	x	7.2	7.3	7.4	7.2	6.9	…	5.4
	y	0.9	0.9	0.8	0.7	0.8	…	0.7
	z	2.4	2.6	2.8	2.5	2.5	…	2.2
⋮								
		…	…	…	…	…	…	
c_m	z	9.0	8.9	9.1	9.3	9.6	…	8.2

Trajectory of c_1

Capturing technologies

- **Acquisition of skeleton data**

2000 Optical sensors (e.g., Vicon)

Inertial sensors (e.g., xSens)

RGB + depth sensors (e.g., Kinect, Xtion)

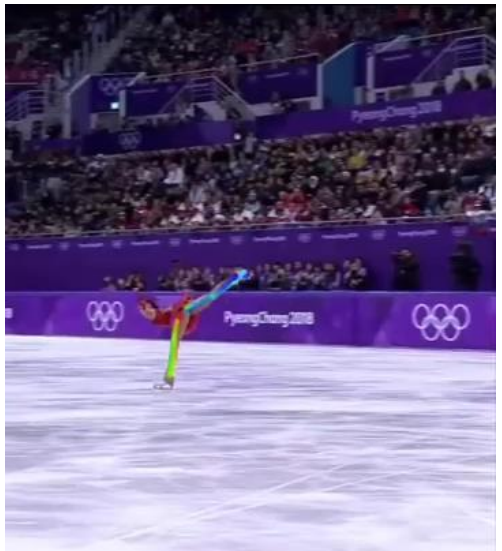
2023 Ordinary video camera (e.g., HRnet, STAF, XNect)

Type	Technologies	Sensors	Joints	Framerate	Error	Cost	Mobility	Invasivity
Optical sensors	Vicon, OptiTrack, Qualisys	10–40	22–32	120–420	mm	$$$	–	Markers
Inertial sensors	Xsens, Vicon	~20	~20	~120	mm–cm	$$	✓	Sensors
RGB + depth sensors	Kinect, Xtion	3	25	~30	>cm	$	–	–
Pose estimation	OpenPose, STAF, XNect	1	14–16	~video	>cm	–	✓	–

Pose estimation example

Great application potential

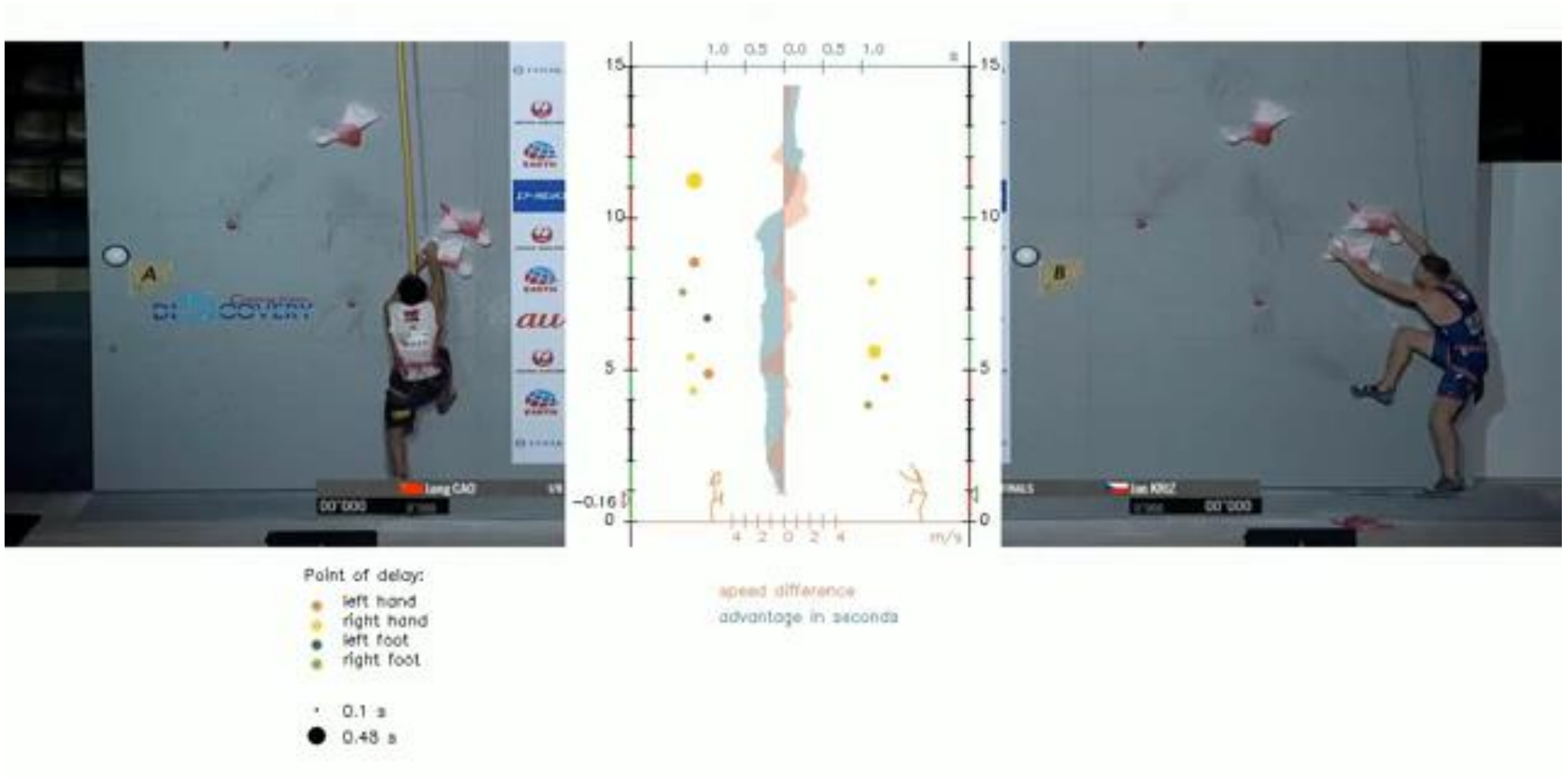
- **A wide variety of possible application domains**
  - Sports – digital referees assessing the quality of performance
  - Virtual reality – recognizing player movements in real time
  - Smart-cities – detecting falls of (elderly) people
  - Healthcare – evaluating the rehabilitation progress remotely



Source: https://www.youtube.com/watch?v=5cI-JibDEMA

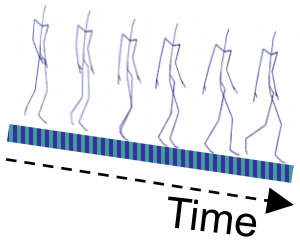
Source: https://blog.usejournal.com/3d-human-pose-estimation-ce1259979306

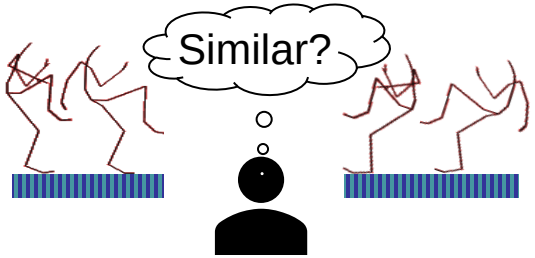
Example application – analysis of speed-climbing performances

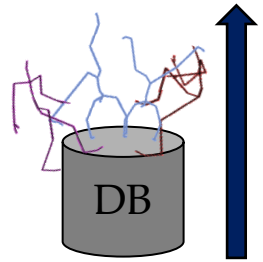


Research objectives

- **Research objective**
  - Effective and efficient content-based access to skeleton data to make them “findable” and thus reusable
  - Content-based access operations:
    - Searching | Subsequence searching
    - Action recognition (classification) | Action detection
    - Motion generation (synthesis)
  - Challenges: Data complexity | Similarity-based comparison | Data volume





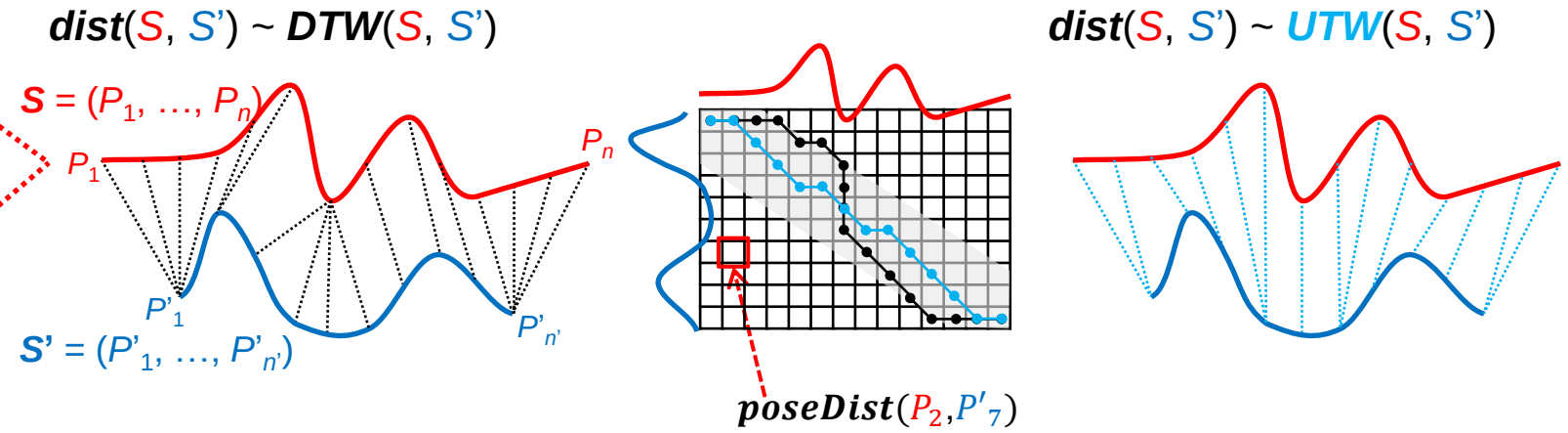


Similarity models – handcrafted

- **Similarity of actions** $dist(S, S') \rightarrow \mathbb{R}_0^+$
  - Actions (sequences of poses) have different lengths → similarity can be determined by time-warping functions
    - Dynamic Time Warping (DTW) – quadratic time complexity $O(n \cdot n')$
    - Uniform Time Warping (UTW) – linear time complexity $O(n + n')$

Matrix representation of 3D skeleton data

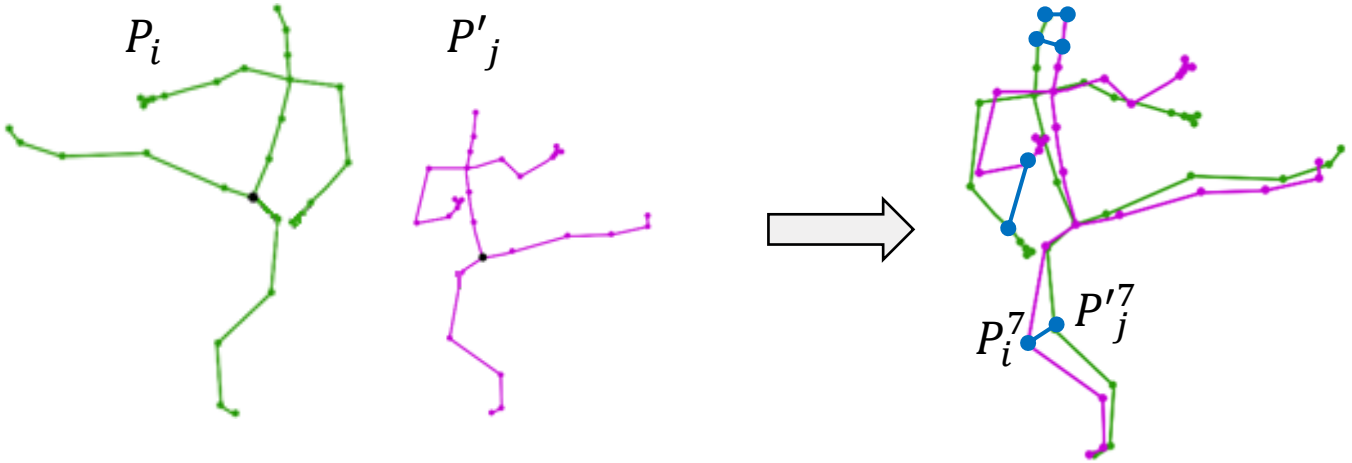
		P_1	P_2	P_3	P_4	P_5	…	P_n
c_1	x	7.2	7.3	7.4	7.2	6.9	…	5.4
	y	0.9	0.9	0.8	0.7	0.8	…	0.7
	z	2.4	2.6	2.8	2.5	2.5	…	2.2
⋮		…	…	…	…	…	…	
c_m	z	9.0	8.9	9.1	9.3	9.6	…	8.2



- Time warping requires a pose-based similarity function $poseDist(P_i, P'_j) \rightarrow \mathbb{R}_0^+$

Similarity models – handcrafted

- **Similarity of poses on raw joint coordinates**
  - Optional pre-processing step – normalization of coordinates:
    - Normalization of position, orientation, and skeleton size



  - Several possibilities, e.g., sum of the Euclidean distances between corresponding 3D joint coordinates:

$$\textbf{\textit{poseDist}}(P_i, P'_j) = \sum_{c=1}^{m} \left\| P_i^c - P'^c_j \right\|$$

Similarity models – deep learning

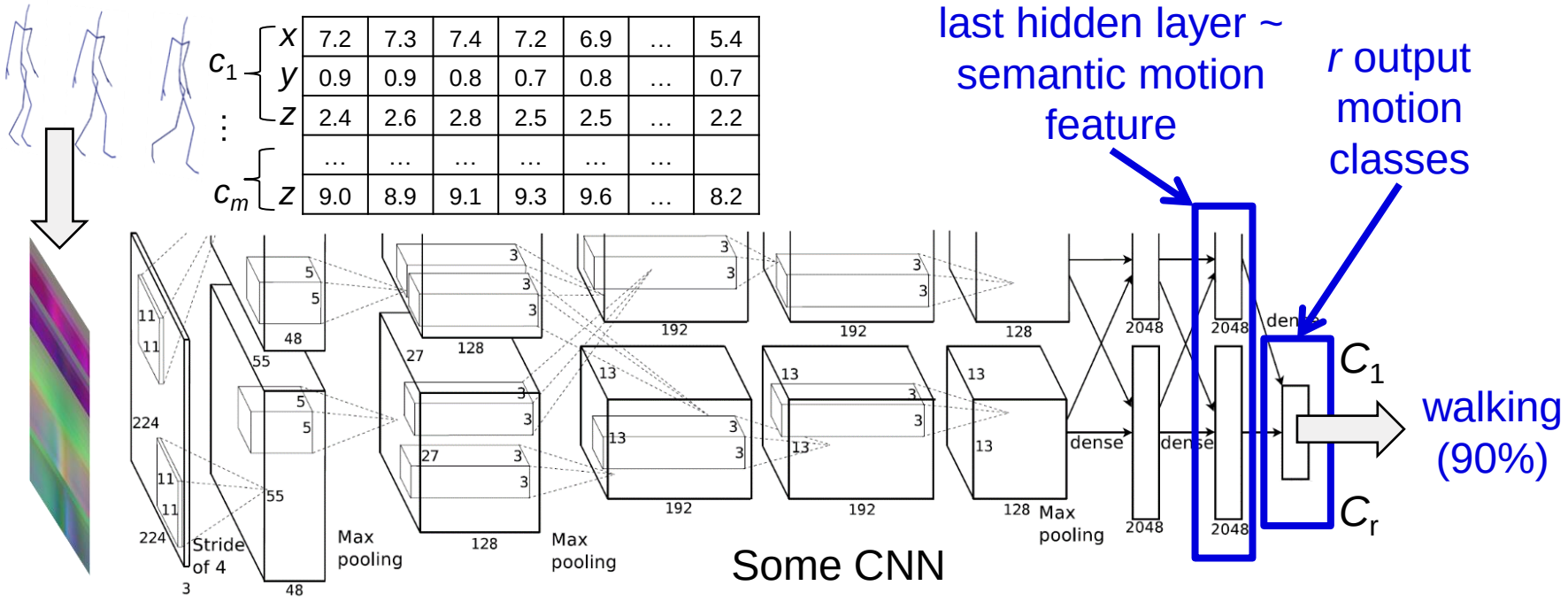
MUNI FI

- Handcrafted models hardly capture semantics of skeleton data
- A variety of deep-learning architectures for learning semantics:
  - 2D/3D Convolutional neural networks (CNNs)
  - Recurrent neural networks (RNNs)
  - Graph convolutional networks (GCNs)
  - Transformers

- Learning:
  - Supervised (~classifiers)
  - Self-supervised
  - Unsupervised

Similarity models – motion images + CNN

- Input skeleton data transformed into a 2D motion image
- Training a CNN to learn semantic motion features (fixed-size vectors)
  - Example of training – training for classification in a supervised way
  - Similarity – features efficiently compared by the Euclidean/Cosine distance



Similarity models – motion images + CNN

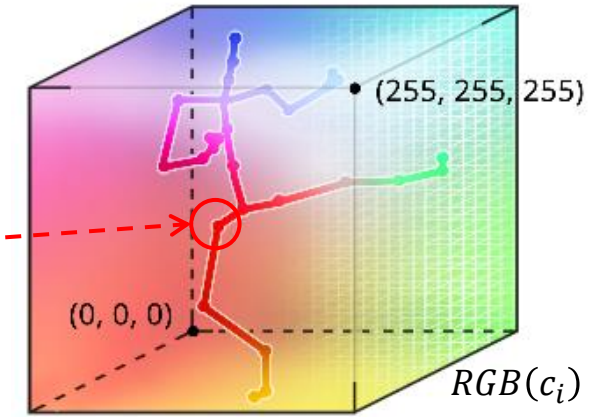
- Construction of motion images:
  - Cover all skeleton poses by a virtual 3D cube
  - Split the cube into 256x256x256 cells and assign a color to each cell based on the RGB color space (16.8M colors)

    => 3D joint position is approximated by a specific color

    => Spatially similar joint coordinates get similar colors

Matrix representation of 3D skeleton data

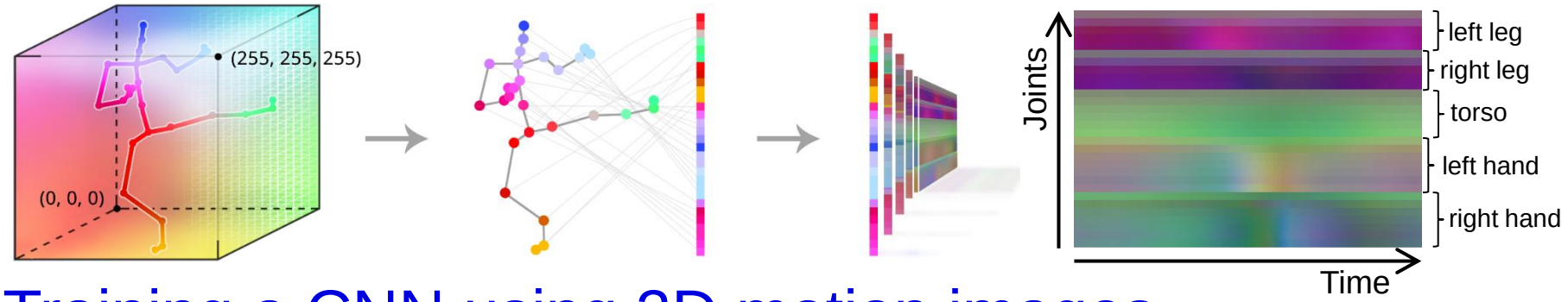
		P_1	P_2	P_3	P_4	P_5	…	P_n
c_1	x	7.2	7.3	7.4	7.2	6.9	…	5.4
	y	0.9	0.9	0.8	0.7	0.8	…	0.7
	z	2.4	2.6	2.8	2.5	2.5	…	2.2
⋮		…	…	…	…	…	…	
c_m	z	9.0	8.9	9.1	9.3	9.6	…	8.2



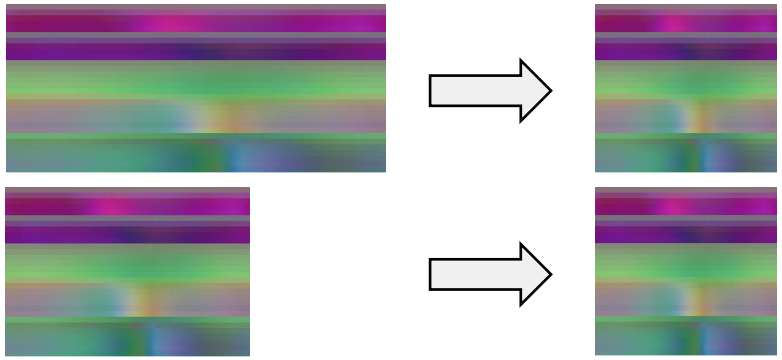
$$RGB(c_i) = \left\lfloor \frac{255}{max - min} \cdot \begin{pmatrix} c_i[x] - min \\ c_i[y] - min \\ c_i[z] - min \end{pmatrix} \right\rfloor$$

Similarity models – motion images + CNN

- Transforming skeleton data into a 2D motion image

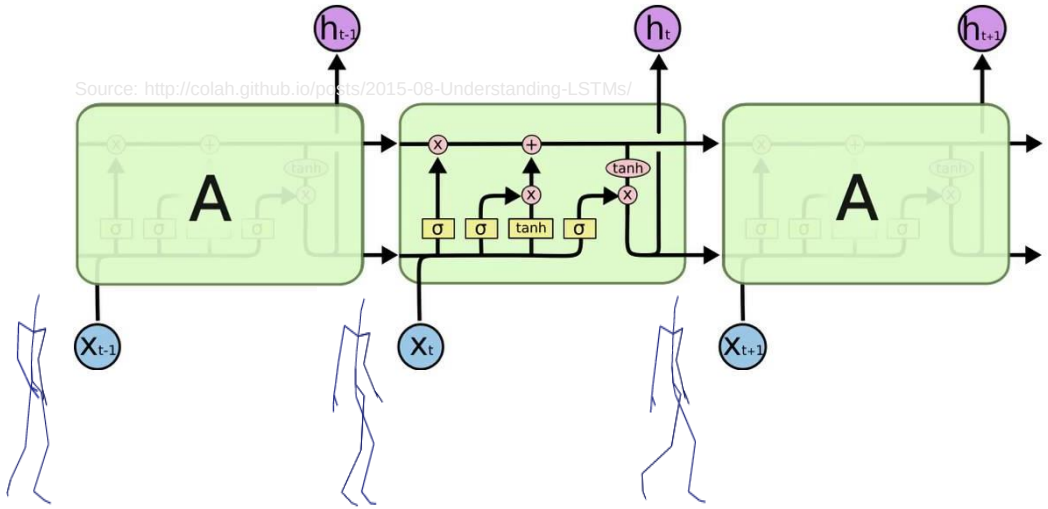


- Training a CNN using 2D motion images
  - Resizing motion images to a fixed size (e.g., 224x224 pixels)
    - Temporal deformations – slower/faster action executions become very similar



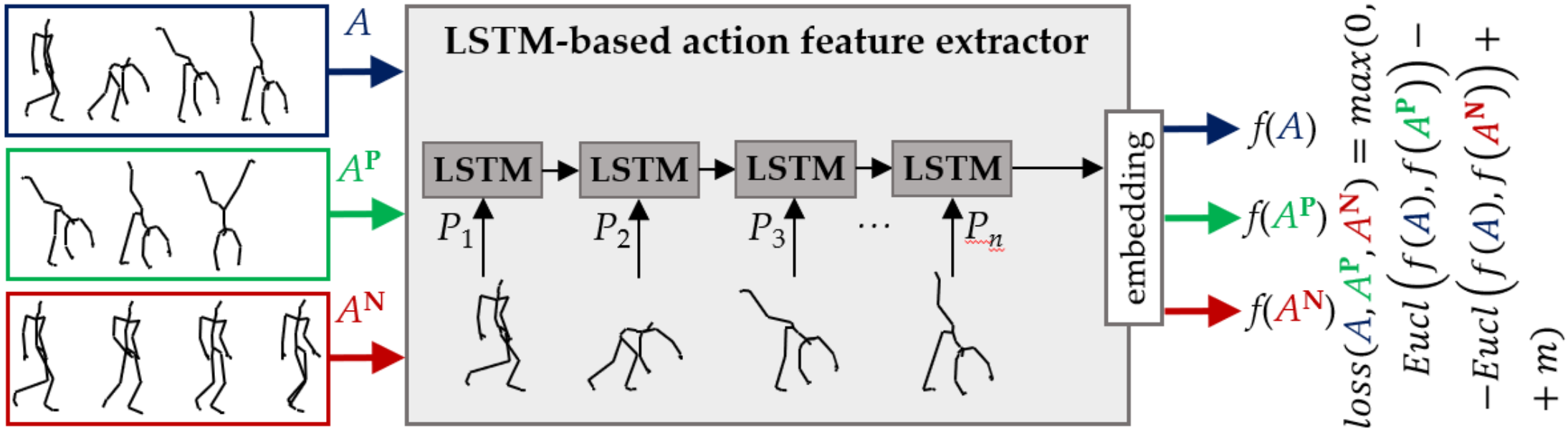
Similarity models – RNNs

- An RNN with GRU/LSTM cells – suitable for learning temporal data
  - Number of states/cells corresponds to the number of poses
- Semantic feature vector – output of the last cell (h_{t+1})
  - Size of each state h_i is a user-defined parameter (e.g., 512 dimensions)
  - Features compared by the Euclidean/Cosine distance



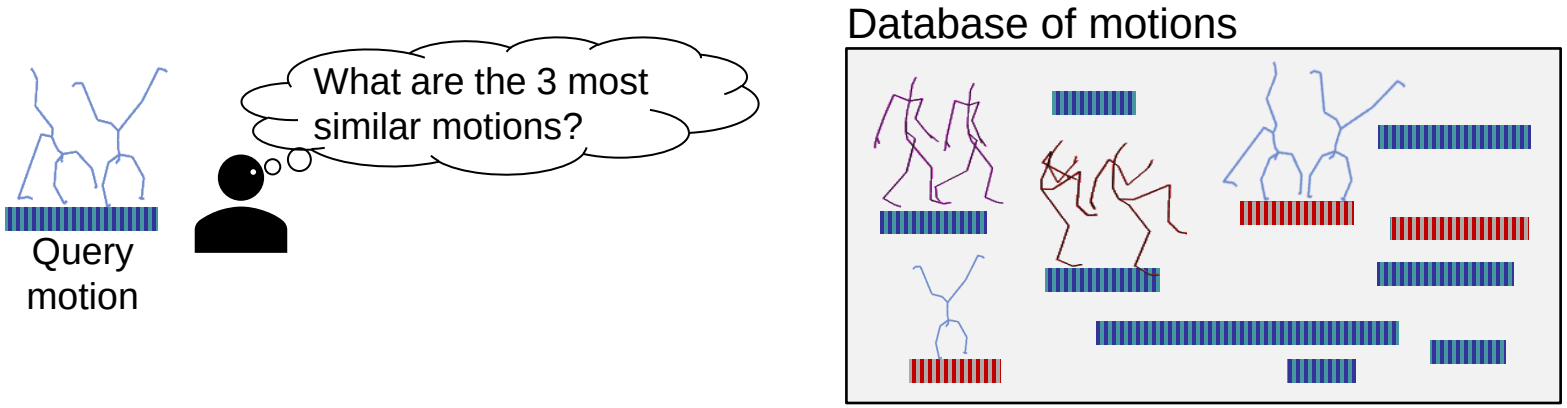
Similarity models – RNNs

- Examples of training:
  - Supervised (for classification)
  - Self-supervised using pairs of similar/dissimilar actions



Searching – query-by-example paradigm

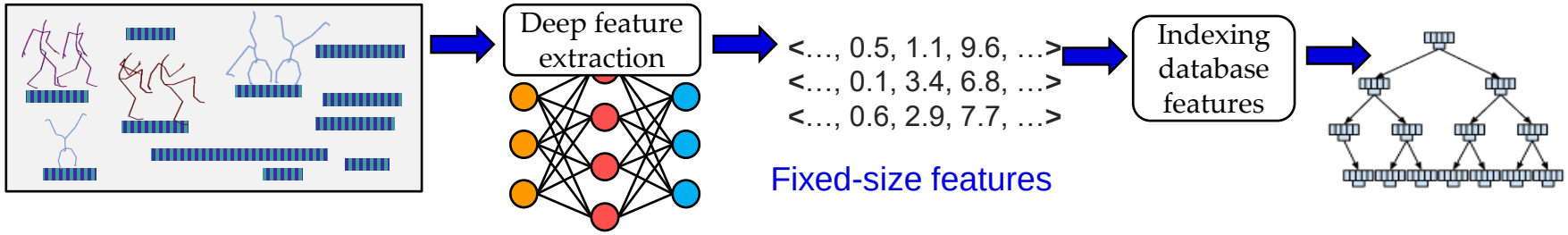
- Query-by-example searching
  - The most fundamental operation – finding the k ($k \in \mathbb{N}$) database motions that are the most similar to a query motion



- Main challenges:
  - Effective similarity model to compare the query and database motion
  - Efficient retrieval algorithm to provide the k most similar motions

Searching – query-by-example paradigm

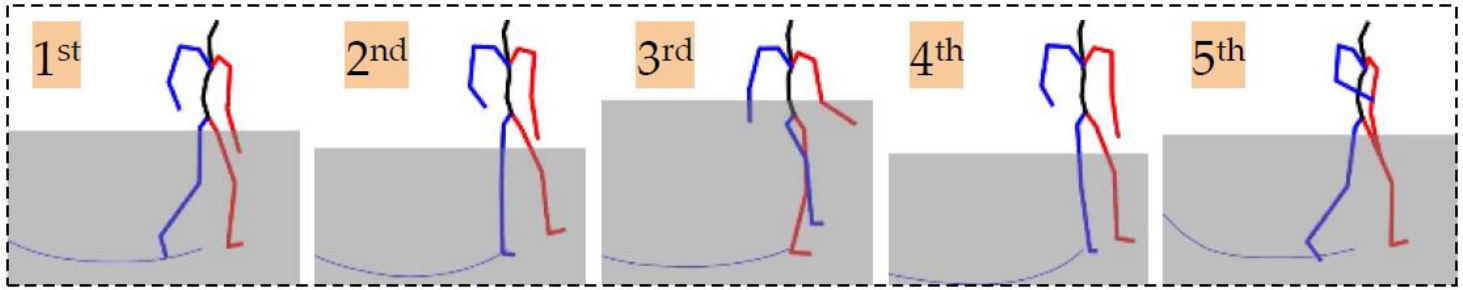
- Query-by-example searching
  - Transforming complex motions to fixed-size feature vectors (~effectiveness)
    - E.g., CNN-based or RNN-based features
  - Indexing feature vectors (~efficiency)
    - E.g., FAISS or PPP-codes



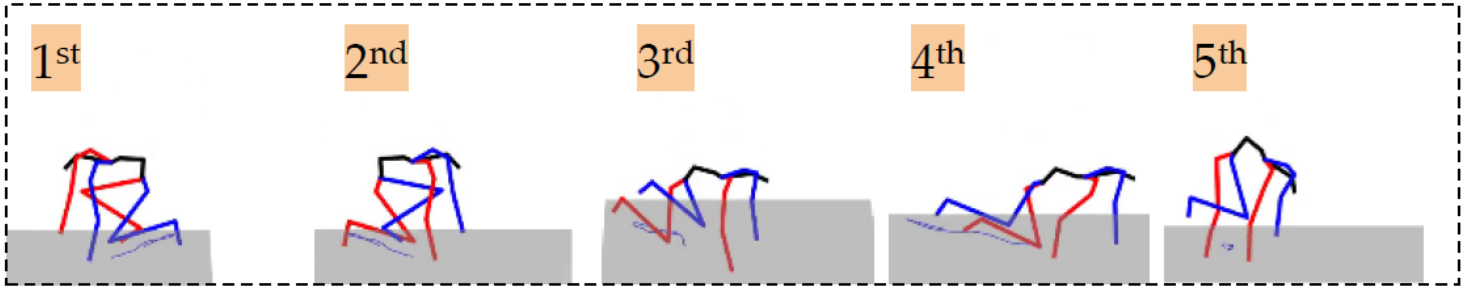
Searching – text-to-motion paradigm

- Text-to-motion searching:
  - Idea – replace the motion example query by a text query
  - Text query specified by a natural language description

"A person walks in a counterclockwise circle."

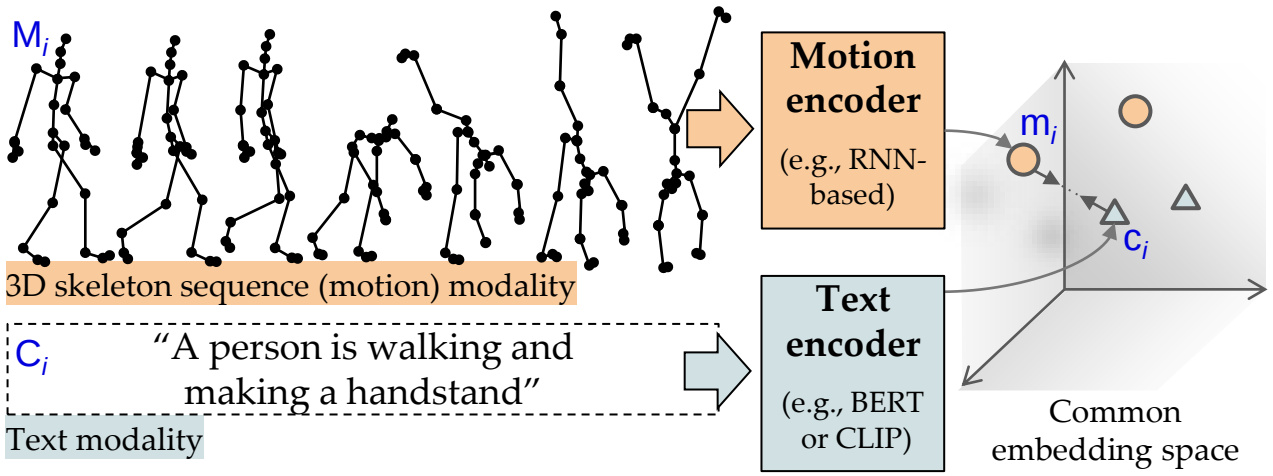


"The person is kneeling down on all fours to begin to crawl."



Searching – text-to-motion paradigm

- Learning a common text-motion space



i-th training motion-text pair (M_i, C_i):
m_i – motion feature vector
c_i – motion description feature vector

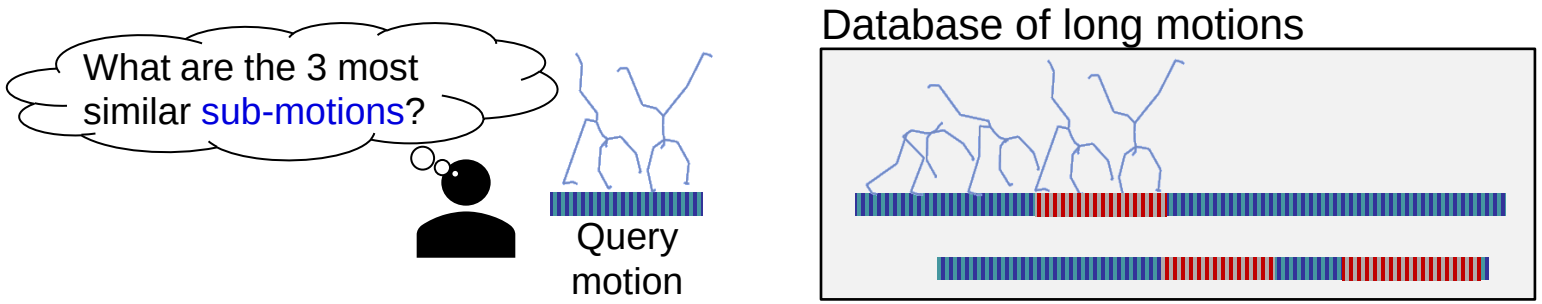
batch *B*=128

(m_1, c_1)
(m_2, c_2)
…
(m_{128}, c_{128})

$$\frac{1}{B}\sum_{i}^{B}\max_{j,j\neq i}[\alpha+S(\mathbf{m}_i,\mathbf{c}_j)-S(\mathbf{m}_i,\mathbf{c}_i)]_+ + \max_{j,j\neq i}[\alpha+S(\mathbf{m}_j,\mathbf{c}_i)-S(\mathbf{m}_i,\mathbf{c}_i)]_+$$

Subsequence searching

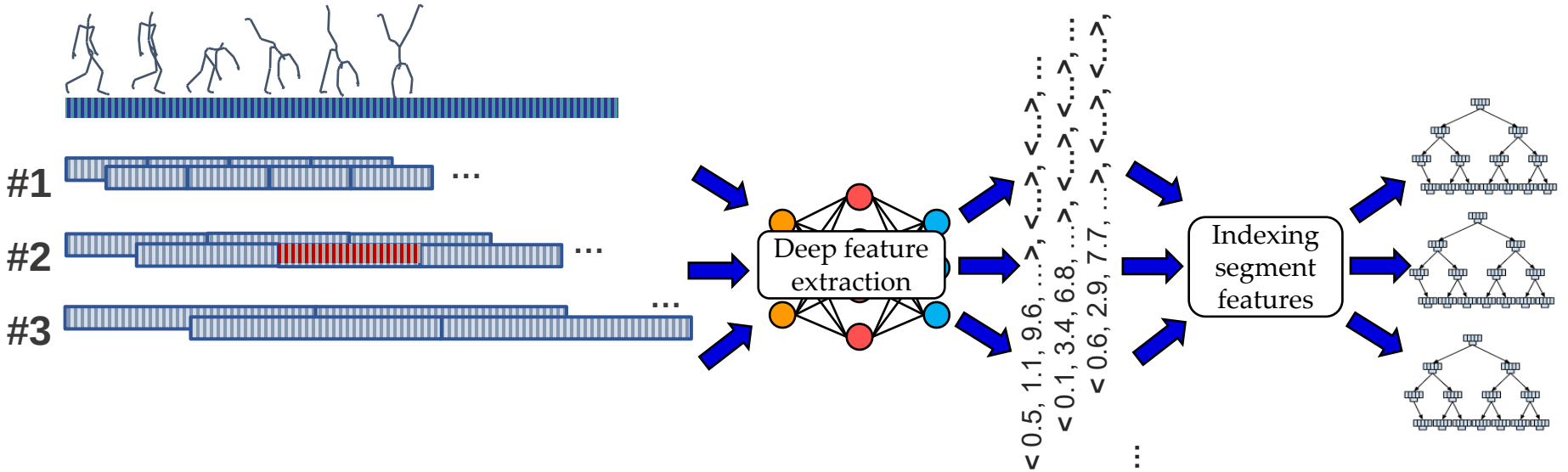
- Query-by-example subsequence searching
  - Inspecting the long database motions to find their *k* subsequences that are the most similar to a query motion



  - Additional challenge to the search task:
    - Efficient retrieval algorithm to localize candidate subsequences

Subsequence searching

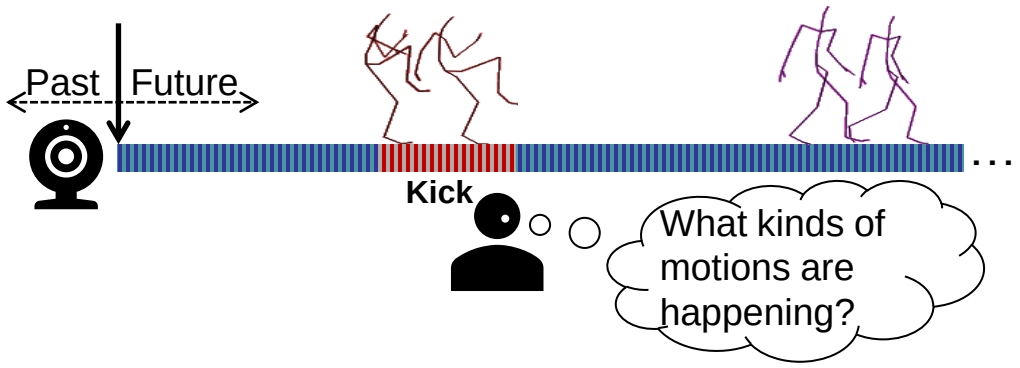
- Possible solution:
  - Partitioning long database motions into many overlapping segments of different sizes
  - Indexing segments of the same size within a single index structure
  - Searching in a “suitable” index for the most query-similar segments

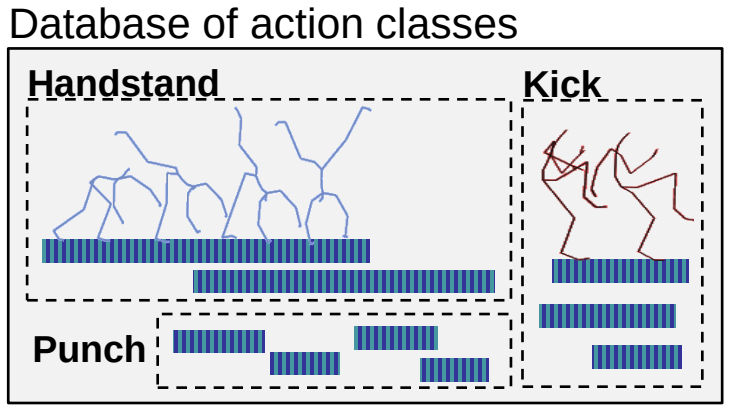


DEMO

Action detection

- Detecting actions (events) in skeleton-data streams
  - Processing a motion stream and detecting its subsequences that correspond to the provided action classes

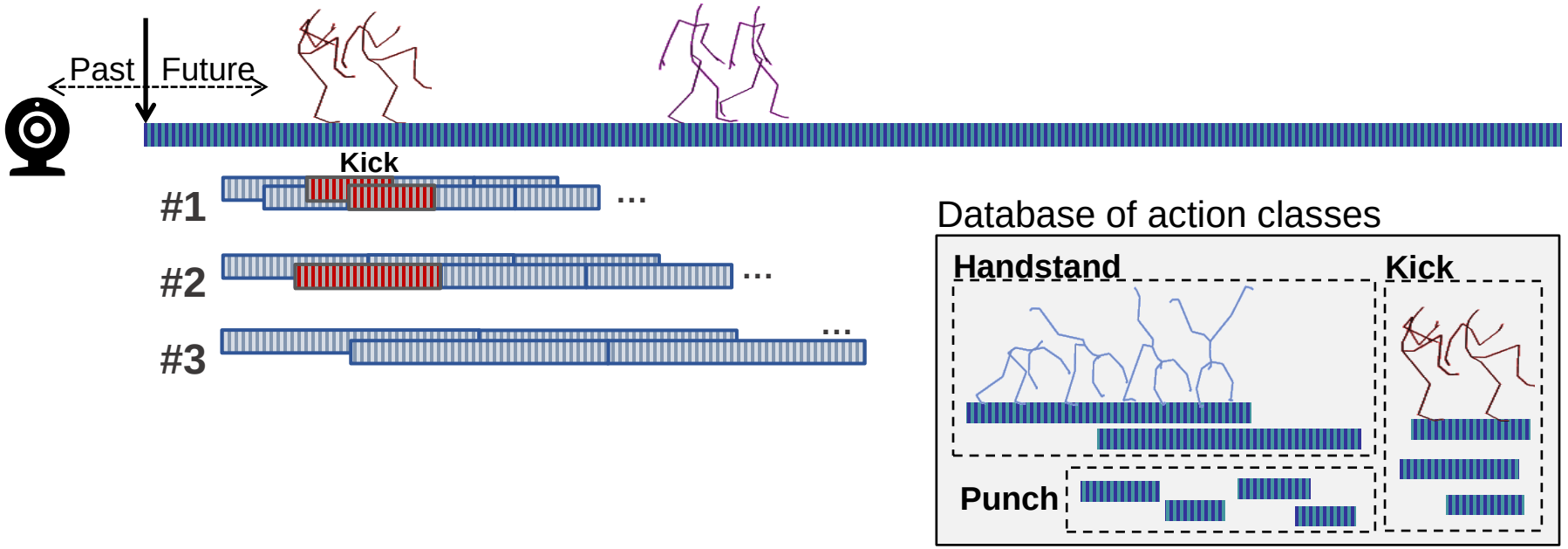




  - Main challenges:
    - Learning effective representations of individual action classes
    - Identifying beginnings and endings of to-be-detected actions

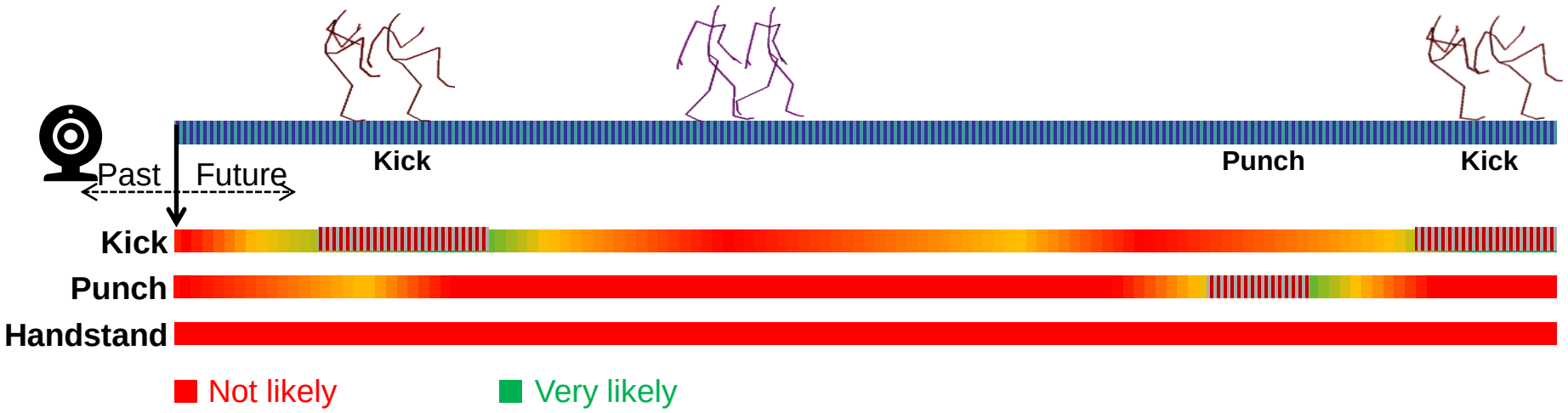
Action detection

- Detecting actions in streams – segment-based principle
  - Applying the “subsequence-search” segmentation
  - Determining similarity between segments and action classes
  - Merging segments



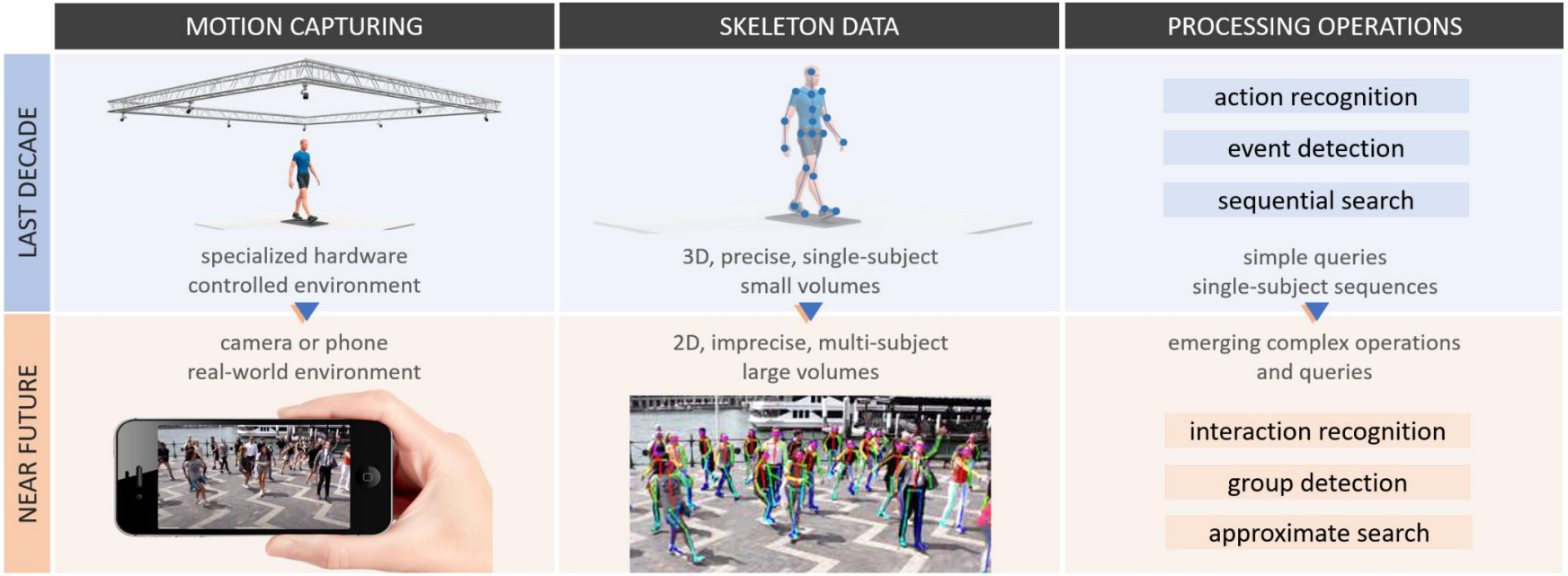
Action detection

- Detecting actions in streams – frame-based principle
  - Adopted LSTM-based recurrent neural network to estimate a probability for each class and frame



Future research

- Paradigm shift:



- Possible topics:
  - Indexing mechanisms for very large skeleton-data collections
  - Explainability of similarity models (e.g., of LSTM-based models)
  - New retrieval models (e.g., regex-based or text-to-motion models)
  - Analyzing interactions of more persons (e.g., detection of groups of people)