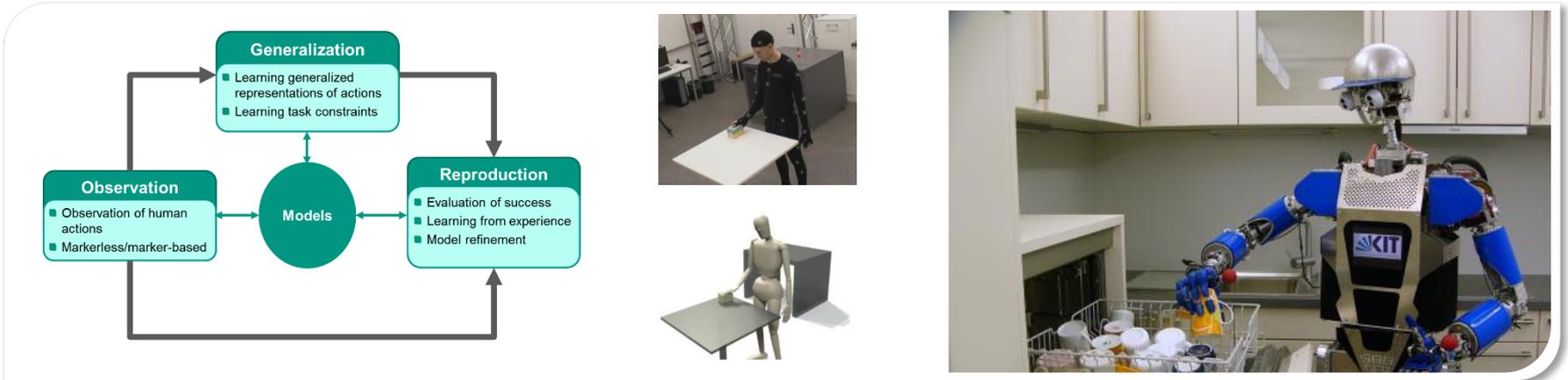


Robotics II: Humanoid Robotics

Chapter 4 – Imitation Learning

Tamim Asfour

<http://www.humanoids.kit.edu>



Imitation Learning in Human

- Human babies or children learn how to use objects from visual demonstrations, by imitating others.
- Only very small number of demonstrations are required.



source: <https://www.youtube.com/watch?v=9FhImmxcwbs>
Andrew N. Meltzoff's research group. <https://ilabs.uw.edu/meltzoff>

What is Imitation?

- Imitation (from Latin *imitatio*, “a copying, imitation”) is an advanced behavior whereby an individual observes and replicates another’s behavior. Imitation is also a form of social learning that leads to the “development of traditions, and ultimately our culture. It allows for the transfer of information (behaviors, customs, etc.) between individuals and down generations without the need for genetic inheritance.” <https://en.wikipedia.org/wiki/Imitation>
- The word imitation can be applied in many contexts, ranging from animal training to politics

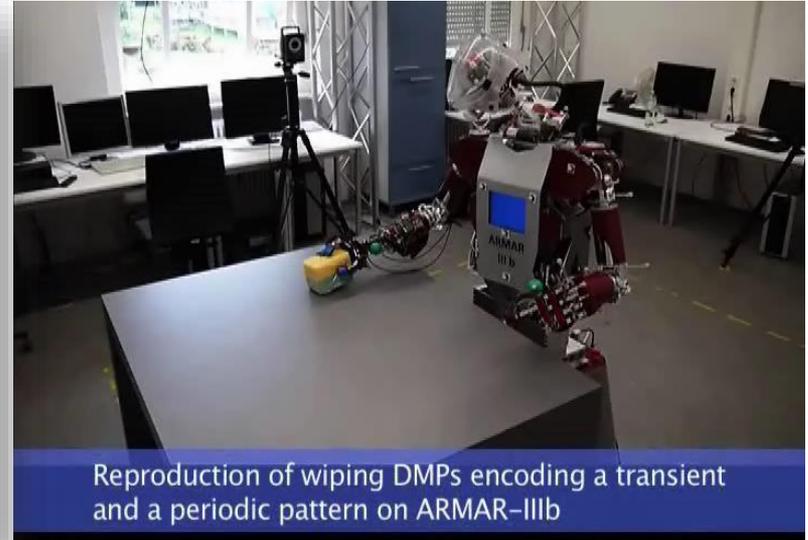


<https://www.compellingconversations.com/teaching-matters-adult-esl-students-learn-children>



<http://oscareducation.blogspot.com/2013/01/observational-learning-or-learning-by.html>

Motivation: Learning From Human Observation



Motivation: Learning From Human Observation



Imitation Learning

- Imitation learning from the viewpoint of
 - **Computer science**
 - Cognitive science
 - Neuroscience

Computer Science View

- Three phases model
 - Decomposition into sub-problems
 - Solution to different problems
 - Integration of the solution



Imitation Learning

- Imitation learning from the viewpoint of
 - Computer science
 - **Cognitive science**
 - Neuroscience

Cognitive Science View (I)

- Focus on “**software**” of brains
- Until 1970, movement imitation did not receive widespread attention anymore, partially due to the prejudice that “imitating” or “mimicking” **is not an expression of higher intelligence**
- True imitation, if...
 - ... the imitated behavior is absolutely new for the imitator
 - ... the same strategy is employed as that of the demonstrator
 - ... the same task goal is achieved

Stefan Schaal “Is Imitation Learning the Route to Humanoid Robots?” *Trends in Cognitive Sciences* (1999)

Cognitive Science View (II)

- Is Imitation an innate ability?
- **Can newborns imitate? → yes!**
- Meltzoff and Moore (Meltzoff 1997, Meltzoff 1983)
 - They reported on the ability of 12-21 day old and, later, even less than an hour old infants to imitate both facial and manual gestures
 - Young infants of this age had neither seen their own faces nor been exposed to viewing faces of other humans for any significant amount of time
 - Thus, the ability to map a perceived facial gesture to their own gestures was concluded to be innate and contradicted Piaget's ontogenetic account of imitation (developmental theory)

Cognitive Science View (III)

- In the light of this new interest in imitation learning, it was discovered that many animals are unable to learn by imitation
- These findings contributed to today's view of imitation as an important **expression of higher intelligence**

Cognitive Science View (IV)

■ Innate Releasing Mechanism (IRM), Meltzoff and Moore:

Model to explain the ability of newborn to imitate

- Every newborn has a certain repertoire of innate movement patterns
- By observing others' movements, these pattern will be activated and a correspondent movement is released
- The model is very interesting because it supports the idea of implementing a **set of pre-programmed behaviors** in a technical system (robot) to bootstrap learning by imitation

Cognitive Science View (V)

- IRM is unlikely because
 - On the one hand, many movements can be imitated, which means that a large number of these movements should be pre-programmed
 - On the other hand, newborns are constantly trying to **improve the imitated movements**, which also speaks against a firm anchorage

J. Demiris and Hayes, G. “Imitative learning mechanisms in robots and humans” In Klingspor, V., editor, *Proceedings of the 5th European Workshop on Learning Robot* (1996)

■ Active Intermodal Mapping (AIM) Modell:

- The visual perception of the teacher's movement is converted into a higher level representation that can be matched against appropriately transformed proprioceptive information about one's own movement
- If this matching space is given, imitation can be seen as **learning to achieve a target representation**, a problem that can be tackled with techniques from supervised learning

■ Active Intermodal Mapping (AIM) Modell:

- The active nature of the matching process is captured by the proprioceptive feedback loop. The loop allows infants' motor performance to be evaluated against the seen target and serves as a basis for correction
- Neonates can detect equivalences between their own acts and ones they see → innerer Abgleichmechanismus
- The perceived and produced human acts are coded within a common (supramodal) framework, an intermodal mechanism for imitation.
- Evidences: several minutes old neonates can imitate a broad range of facial and manual gestures even when they cannot sense their faces.
- **Mirror neuron**: Active not only during observation but also during the action.

Imitation Learning

- Imitation learning from the viewpoint of
 - Computer science
 - Cognitive science
 - **Neuroscience**

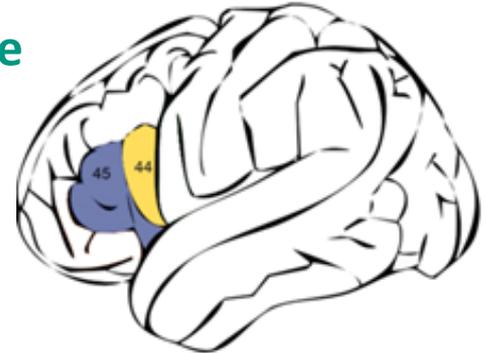
Neuroscience and Cognitive Neuroscience (I)

- Focus on „**hardware**“ of brains
- Imitations-region in brains (F5)
 - Active during observation **and** movement
 - Active during **whole** observation
 - Essential prerequisite for imitation is a **connection between the sensory systems and the motor systems** such that percepts can be mapped onto appropriate actions. This mapping is a difficult computational process as visual perception takes place in a different coordinate frame to motor control.
- Connection to some neurons in F5, **the mirror neurons**, which are active *both* when the monkey *observed* a specific behavior and when it *executed* it itself

G. Rizzolatti, R. Camarda, L. Fogassi, M. Gentilucci, G. Luppino, M. Matelli “Functional organization of inferior area 6 in the macaque monkey. II. Area F5 and the control of distal movements” *Exp. Brain Res* (1988)

Neuroscience and Cognitive Neuroscience (II)

- Mirror neurons are active during the whole observation and action and not only during a certain subsequence
→ **neurons encode complete movement and concrete types of movements, not only their subsequences**
- This result plays a very important role for finding an appropriate representation of observed movements
- A special area in human's brain (**Broca-Area**) is found, which has almost the same function as mirror neurons



https://en.wikipedia.org/wiki/Broca%27s_area

M. Jeannerod, M. A. Arbib, G. Rizzolatti, H. Sakata "Grasping objects: the cortical mechanisms of visuomotor transformation" *Trends Neuroscience* (1995)

A. Murata, L. Fadiga, L. Fogassi, V. Gallese, V. Raos, G. Rizzolatti "Object representation in the ventral premotor cortex (area F5) of the monkey" *J. Neurophysiology*, (1997)

Terms, Literature

Red: relevant for the exam

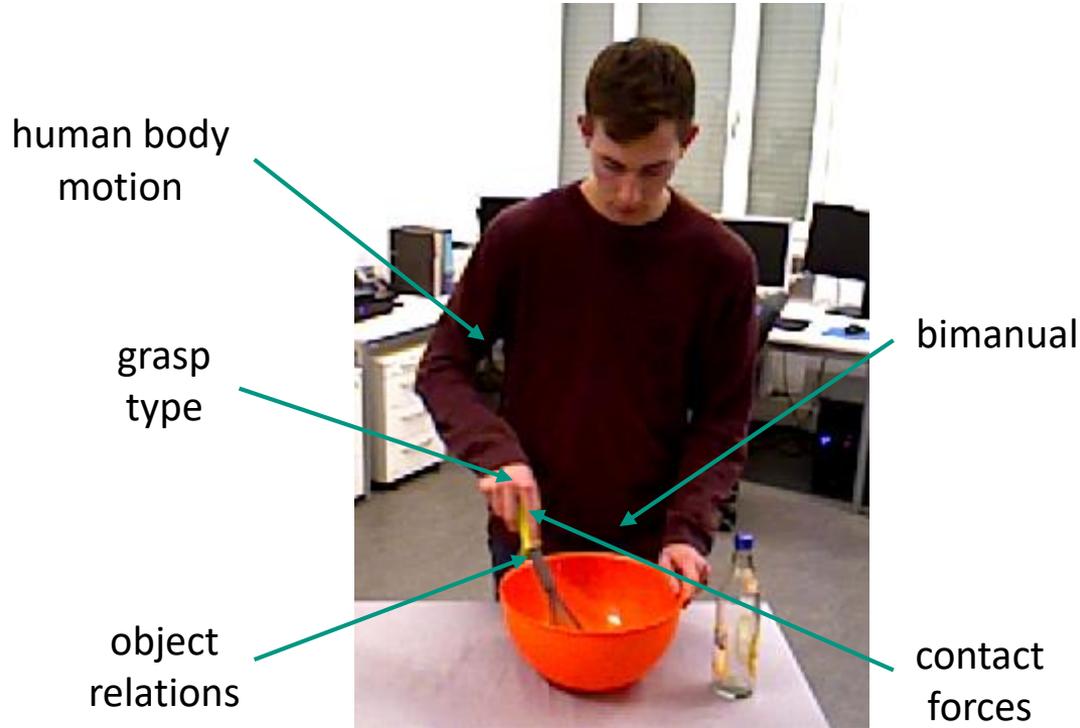
- Learning from Human Observation
- Learning from Human Demonstration (LfD)
- Robot Programming from/by Demonstration (PbD)
- Apprenticeship learning
- Imitation Learning (IL)

Billard, S. Calinon, R. Dillmann and S. Schaal, Robot Programming by Demonstration, In Handbook of Robotics, Springer, pp. 1371-1394, 2008

B. Argall, S. Chernova, M. Veloso and B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems, 2009

J. Romero, From human to robot grasping, PhD Thesis, 2011

Example: What is Important?



Key Questions of Imitation and PbD – Overview

■ WHAT to imitate

- ... essential
- ... start and end by sequencing
- ... goal

■ HOW to imitate

- ... match movement of demonstrator to own
- ... movement

■ WHOM to imitate

■ WHEN to imitate

Learning a skill

Largely unexplored!

Key Questions of Imitation and PbD – “Whom”

1. Whom to imitate?

- Choosing a demonstrator whose behavior can benefit the imitator
(**teacher selection**)

Key Questions of Imitation and PbD – “When”

2. **When** to imitate?

- The imitator has to **segment** and identify the beginning and end of a shown behavior
- The imitator has to decide if the observed behavior is **appropriate in the current context**, and also how many times this behavior should be repeated

Key Questions of Imitation and PbD – “What” (I)

3. What to imitate?

- Find out what aspects of the demonstration are of interest
 - In the case of **actions**, the demonstrator’s **movements (trajectories)** are relevant
 - In other situations, the results and the **effects** of actions are considered more important
- For a given task, certain observable or affectable properties may be **irrelevant** and safely ignored
 - Example: if the demonstrator always approaches a location from the north, is it necessary for the robot to do the same?

Key Questions of Imitation and PbD – “What” (II)

3. What to imitate?

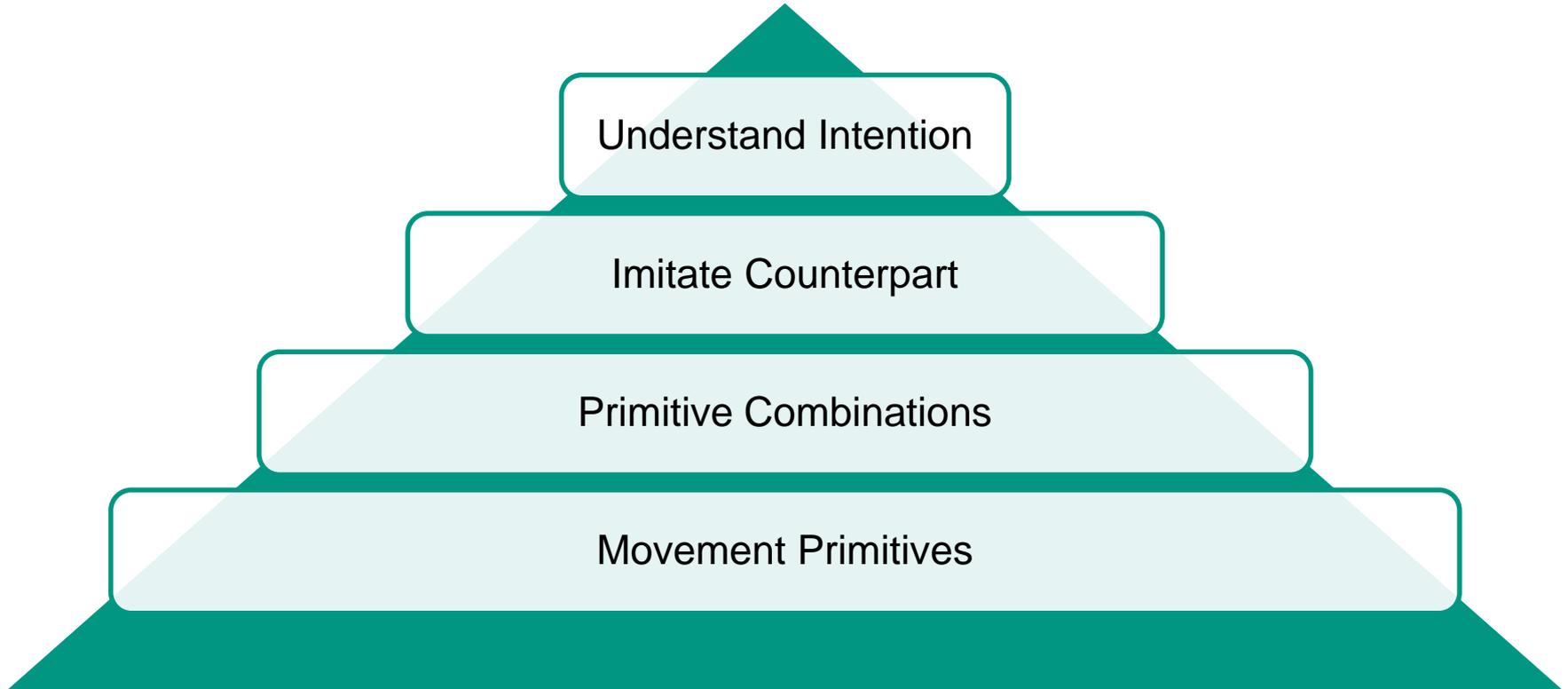
- In continuous control tasks, “what to imitate” relates to the problem of automatically defining the **feature space** for learning, as well the **constraints** and the **cost function**
- In discrete control tasks, such as those treated by reinforcement learning and symbolic reasoning, “what to imitate” relates to the problem of how to define the **state** and **action space** and of how to automatically **learn** the **pre/post conditions** in an autonomous decision system

Key Questions of Imitation and PbD – “How”

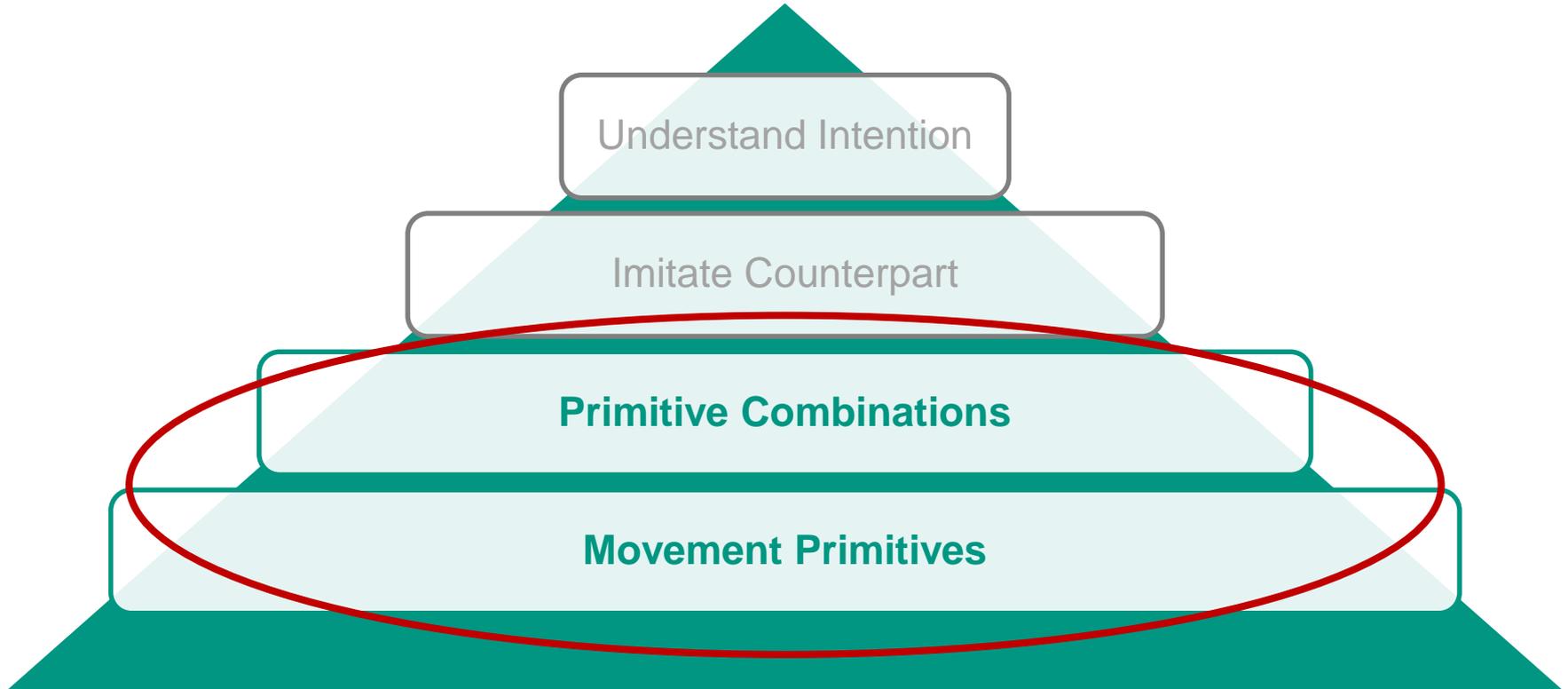
4. How to imitate?

- Determine how the robot will actually perform the learned behaviors to maximize the metric found when solving the “what to imitate” problem
 - A robot cannot act exactly in the same way as a human, due to differences in physical embodiment
 - E.g., if the demonstrator uses a foot to move an object, is it acceptable for a wheeled robot to bump it, or should it use a gripper instead?
- The robot has to learn how to imitate by **mapping perception into a sequence of motion actions related to its own body**
 - Embodiment of the robot and its body constraints determine how observed action can be imitated (**correspondence problem**)

Meltzoff's Imitation Learning Model (I)



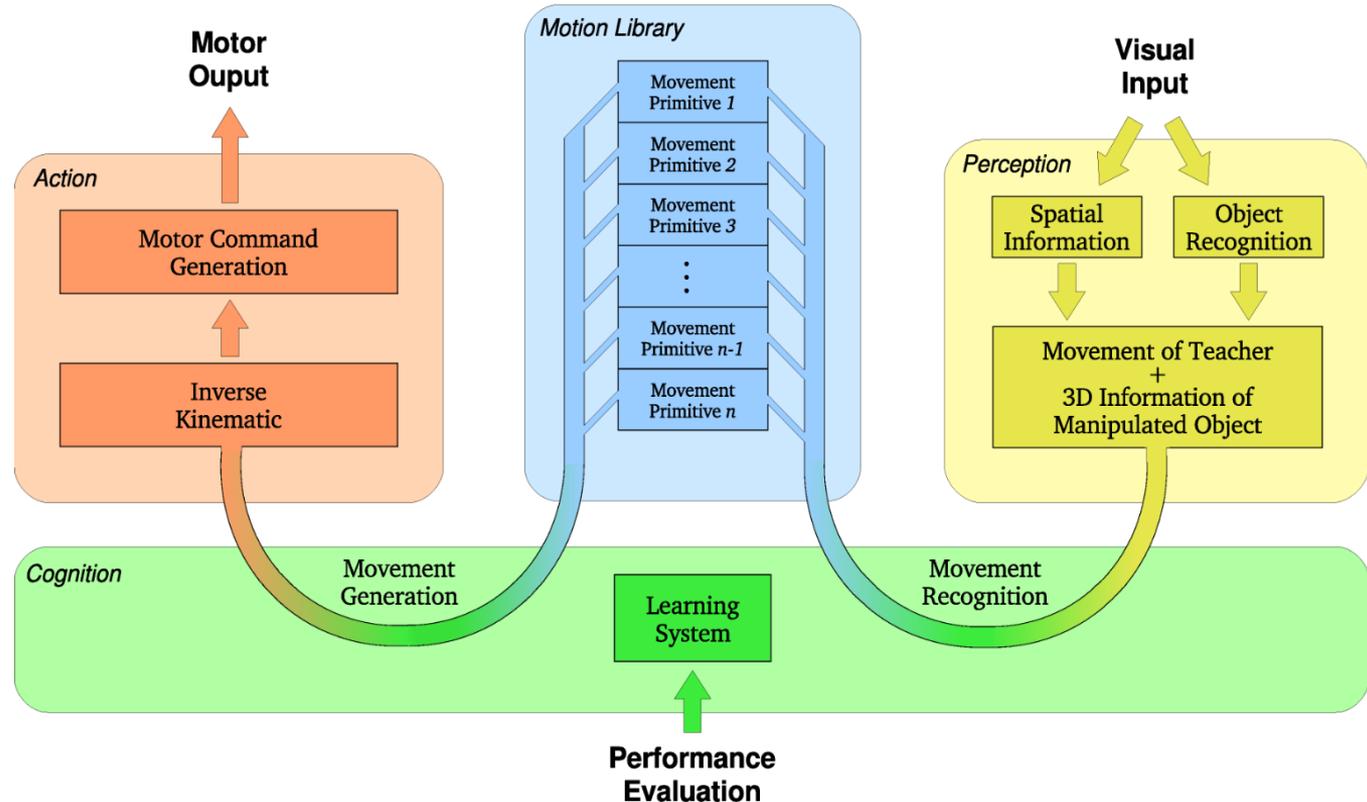
Meltzoff's Imitation Learning Model (II)



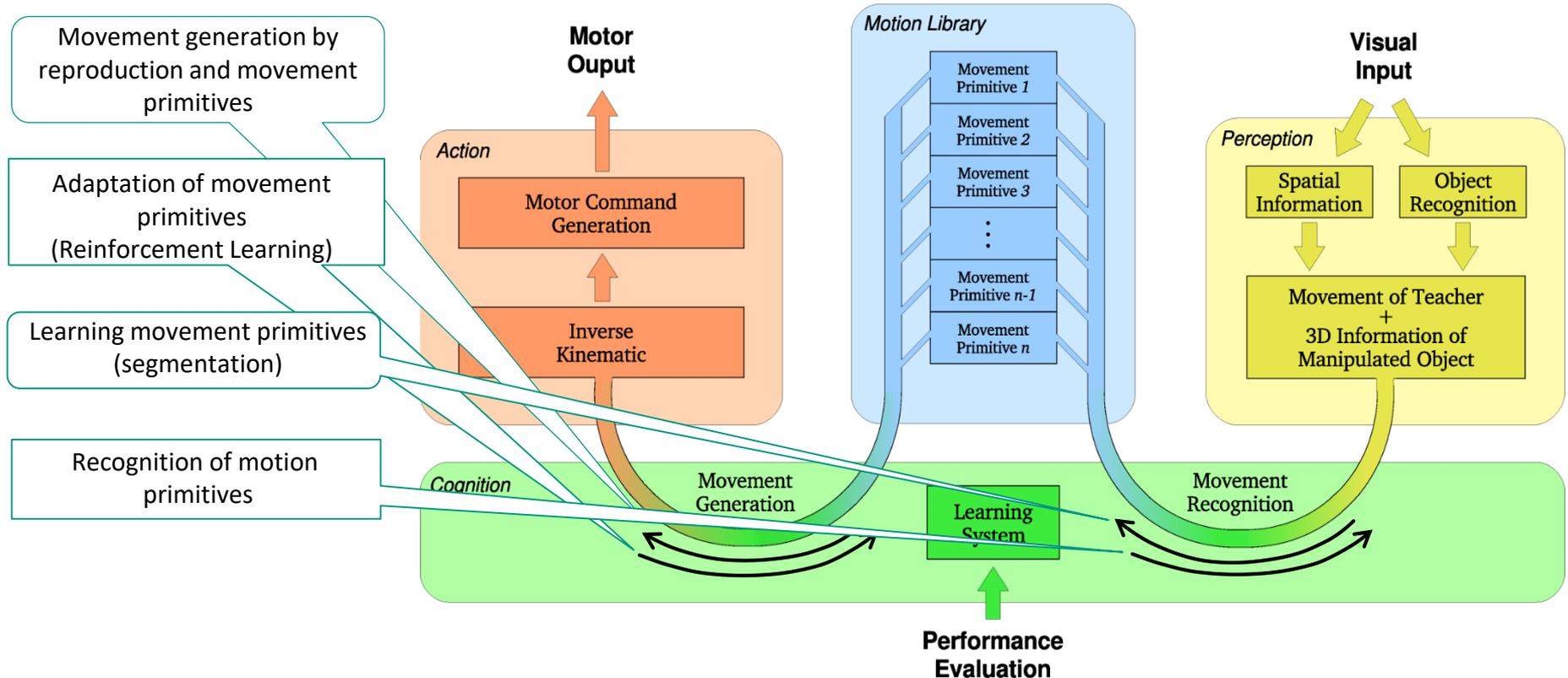
- Three reasons for IL & PbD:
 - PbD is a powerful mechanism for **reducing the complexity of search spaces for learning**
 - Positive examples: Start search from positive observations
 - Negative examples: Eliminating negative observations from search space
 - PbD offers an **implicit means of training a robot**, such that explicit and tedious programming by a human user can be minimized or eliminated
 - Studying and modeling the **coupling of perception and action**, which is at the core of imitation learning, helps us to understand the mechanisms by which the self-organization of perception and action could arise during development.

Imitation Learning

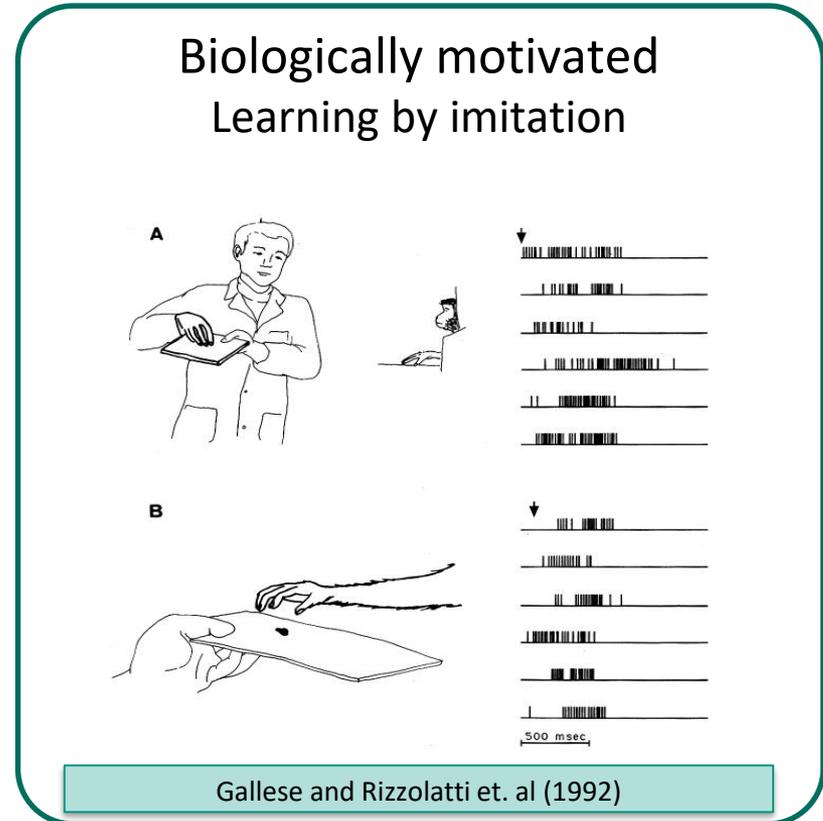
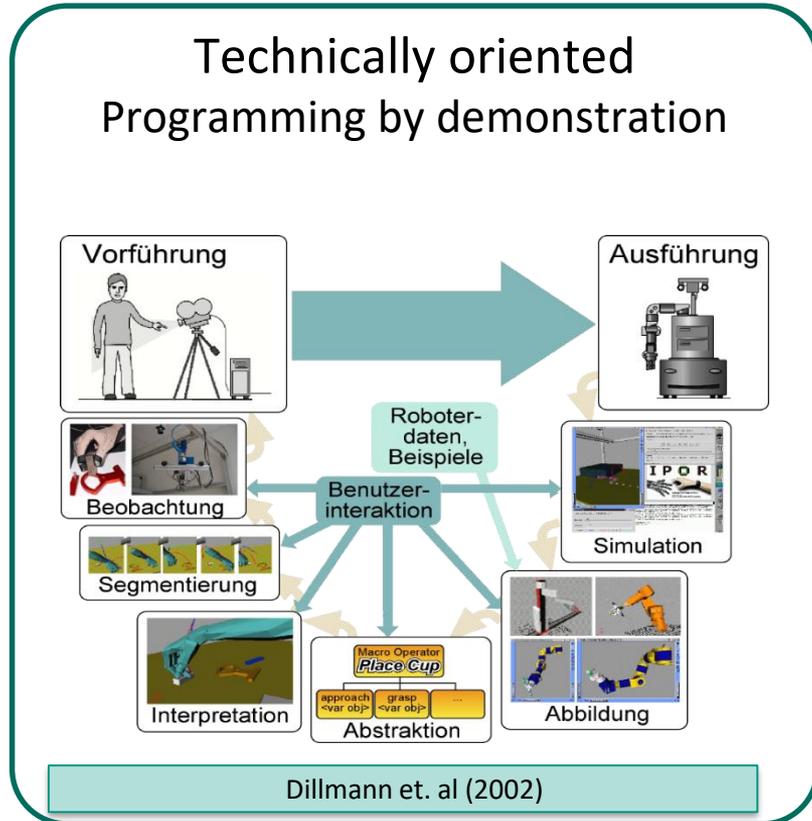
Idea: Reproduction of mirror neurons by movement primitives



Imitation Learning



Motivation: Learning From Demonstration



Proposal for a Procedure

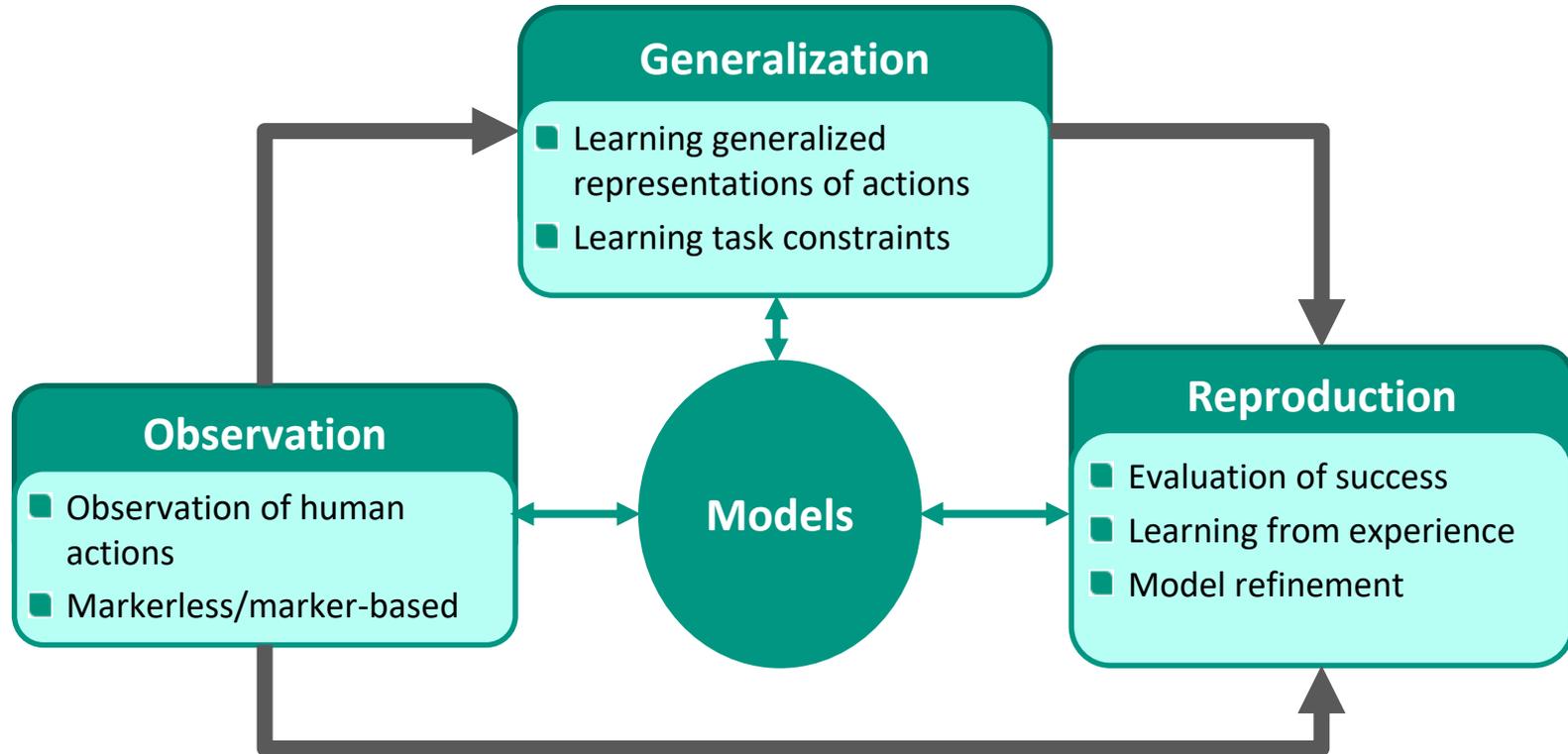
- Build a **motion library** by observing human's action
 - Elements of motion library: **Motion primitives**
 - Focus on goal-oriented movements, consider other constraints, such as objects and forces...
 - Methods of recording human movements (motion capture techniques)
- Application of motion primitives to generate more complex whole body movements
- **Key questions:** Representation and adaptation of motion primitives

Learning a Motion Library

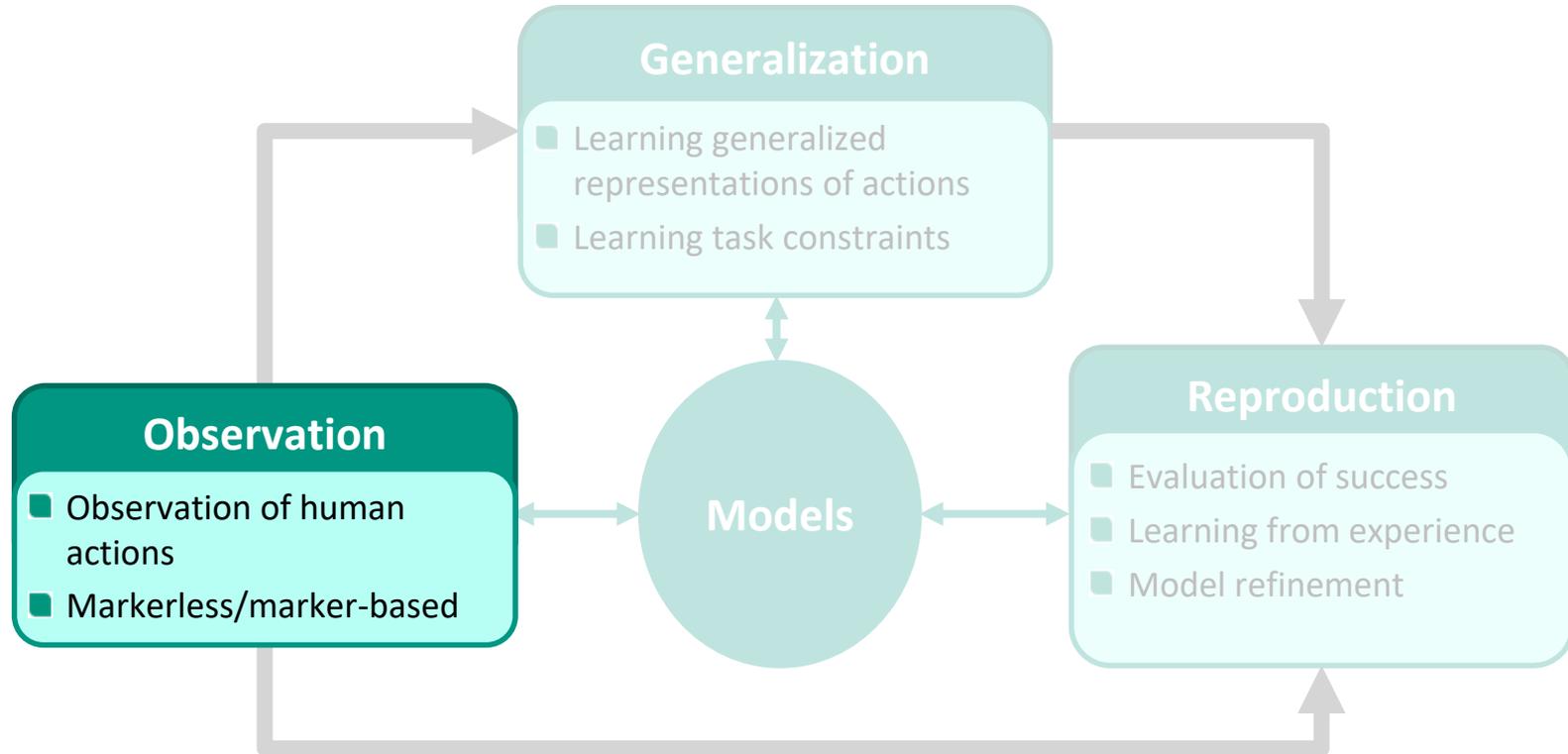
- Motion alphabet, similar to natural language
- Library of motion primitives
- Tasks as sequences of motion primitives



Learning From Human Observation



Learning From Human Observation



Observation

■ Human motion capture techniques

■ Marker-based systems

- Passive markers (VICON)
- Active markers
- IMU based

■ Markerless systems

- From Images (RGB, RGB-D)
- Exoskeletons
- Deep learning-based approaches

■ Observation of object motions

- Object detection, segmentation
- Object pose estimation and tracking
- Object feature correspondence detection
- Scene graph

Observation

■ Human motion capture techniques

■ Marker-based systems

- Passive markers (VICON)
- Active markers
- IMU based

■ Markerless systems

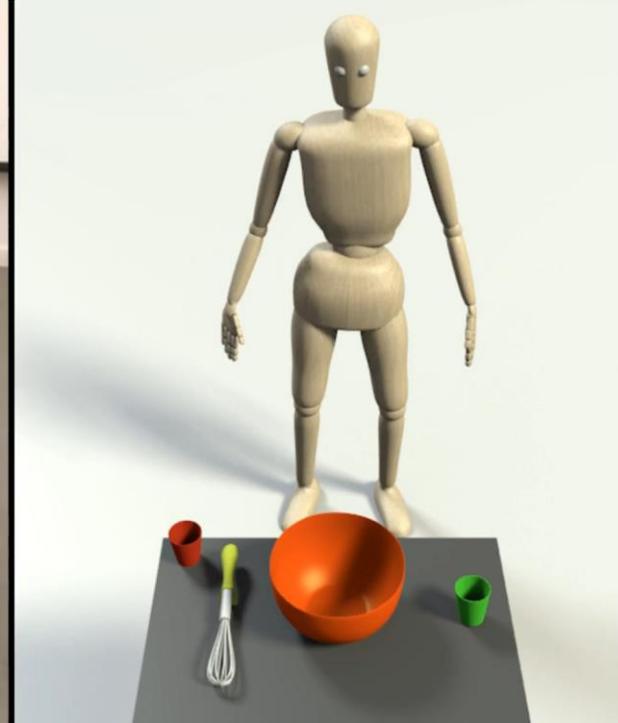
- From Images (RGB, RGB-D)
- Exoskeletons
- Deep learning-based approaches

■ Observation of object motions

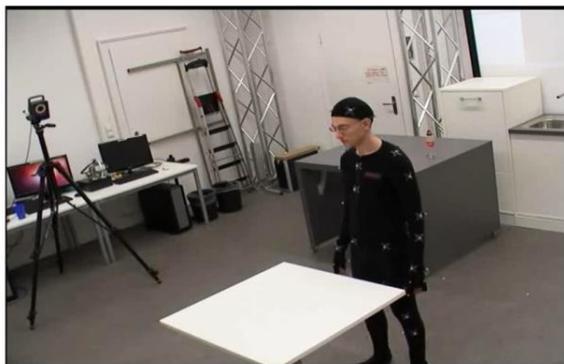
- Object detection, segmentation
- Object pose estimation and tracking
- Object feature correspondence detection
- Scene graph

Capturing and Reconstruction

- ... of whole-body **and** objects motions



■ Data for learning a robot motion alphabet



Conversion of Human and Object Motions with the MMM Framework

Statistics

- **42 hours** of manually labeled human motion data (including object information)
- **13198** motions
- **232** (110/41) subjects
- **158** objects
- **2.1 TB**

motion-database.humanoids.kit.edu
<https://git.h2t.iar.kit.edu/sw/mmm>

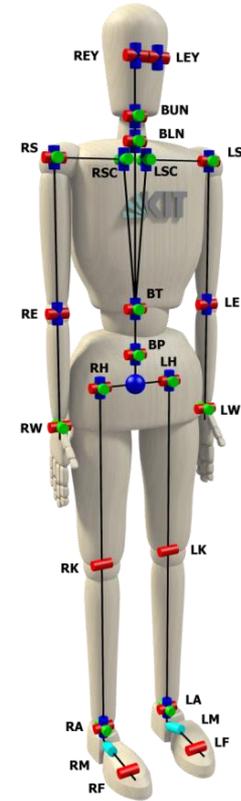
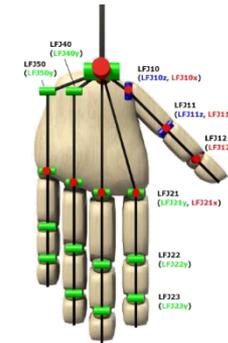
The Master Motor Map (MMM)

- Framework for the **unifying representation** of whole-body motion, motion analysis and its transfer to humanoid robots

- **Reference model of the human body with currently 104 DoF**
 - Kinematics and dynamics model
 - Joint Limits (min/max)
 - Center of Mass
 - Mass percentage of total weight
 - Inertia Tensor

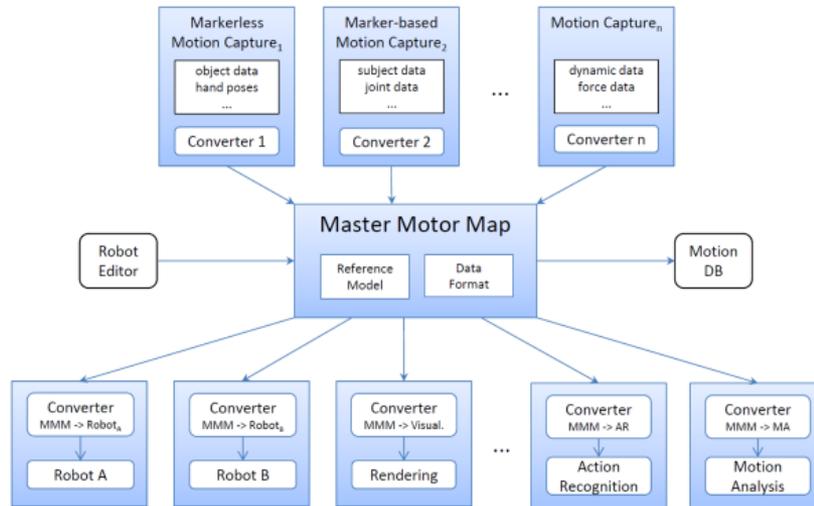
 - Kinematics/Dynamics properties scalable regarding subject height and weight (scaled by MMM model processor)

 - **Subject-specific** parameters



The Master Motor Map (MMM)

- **Unifying framework** for capturing, representation, visualization and whole body human motion and mapping/converting to different embodiments



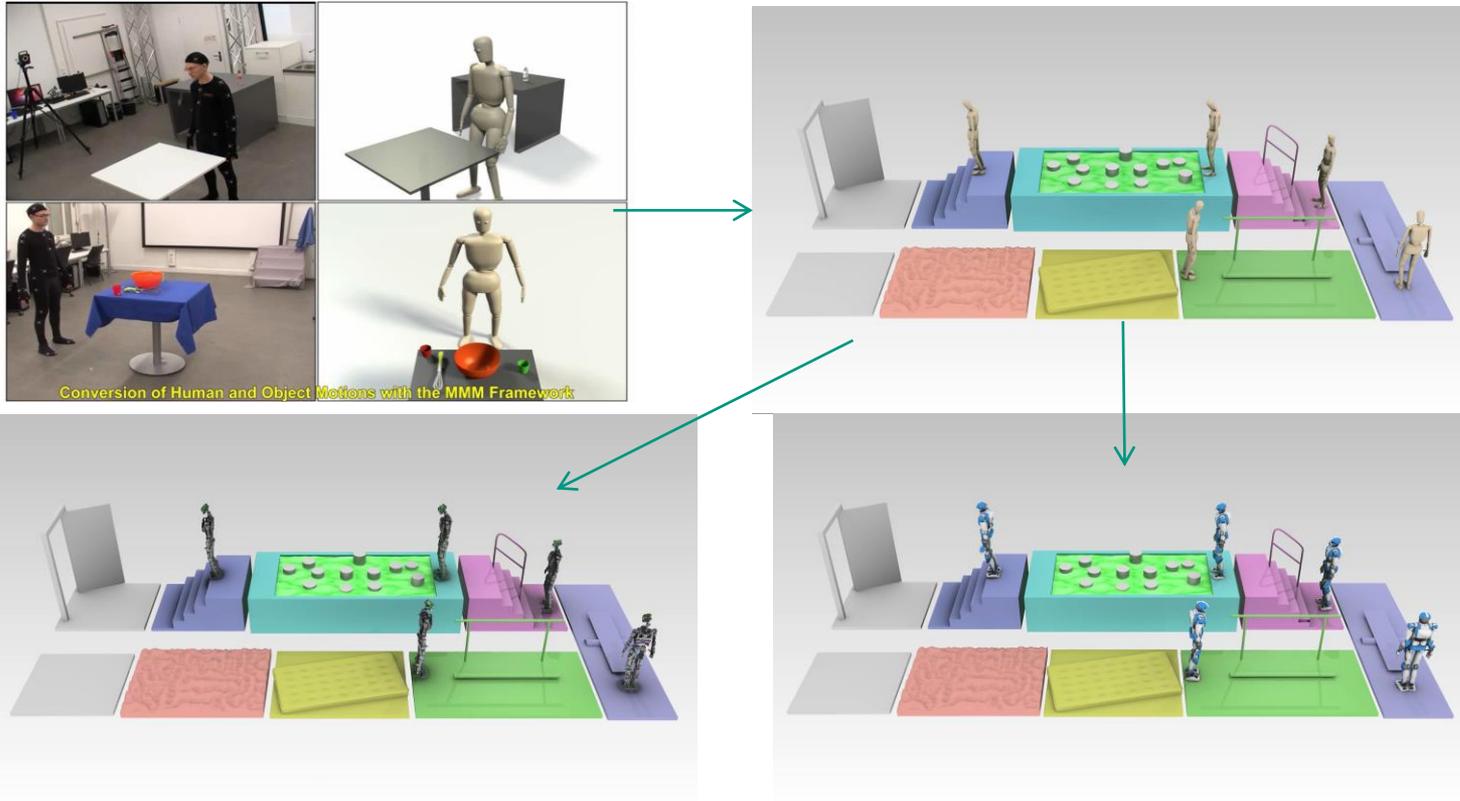
<https://mmm.humanoids.kit.edu>
<https://mmm.humanoids.kit.edu>

Mandery, C., Terlemez, Ö., Do, M., Vahrenkamp, N. and Asfour, T., *Unifying Representations and Large-Scale Whole-Body Motion Databases for Studying Human Motion*, IEEE Transactions on Robotics, vol. 32, no. 4, pp. 796-809, 2016

Motion Reproduction with MMM



From Human to Humanoid Motion with MMM



Observation

■ Human motion capture techniques

■ Marker-based systems

- Passive markers (VICON)
- Active markers
- IMU based

■ Observation of object motions

- Object detection, segmentation
- Object pose estimation and tracking
- Object feature correspondence detection
- Scene graph

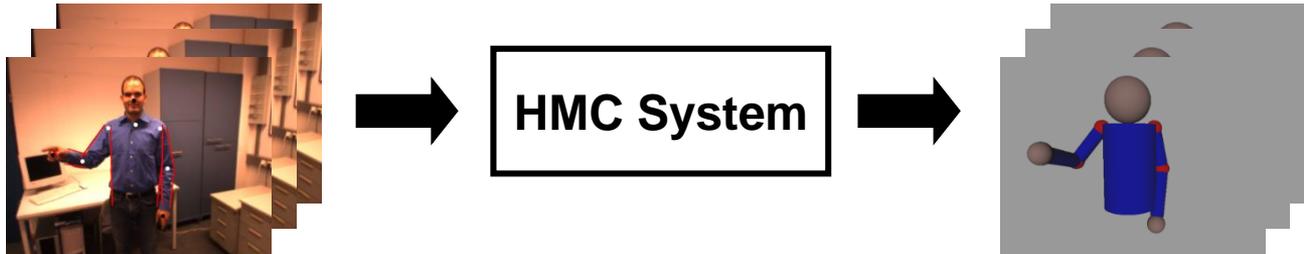
■ Markerless systems

- From Images (RGB, RGB-D)
- Exoskeletons
- Deep learning-based approaches

Markerless Human Motion Capture

- Human Motion Capture (HMC):
 - The system operates on a simplified 3D human model
 - Output is a sequence of configuration vectors of this model, one for each frame
- Markerless:
 - The only input to the system is a sequence of stereo image pairs
 - No markers are used

Azad, P., Asfour, T. and Dillmann, R., *Robust Real-time Stereo-based Markerless Human Motion Capture*, IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 700-707, December, 2008



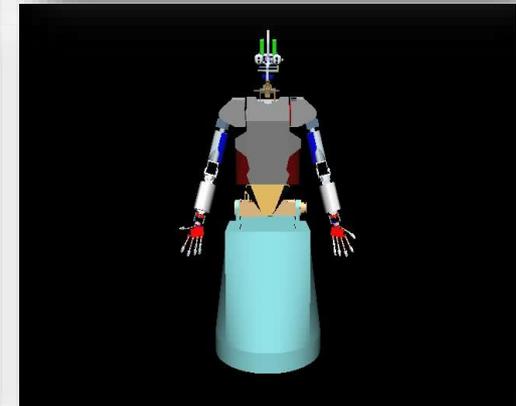
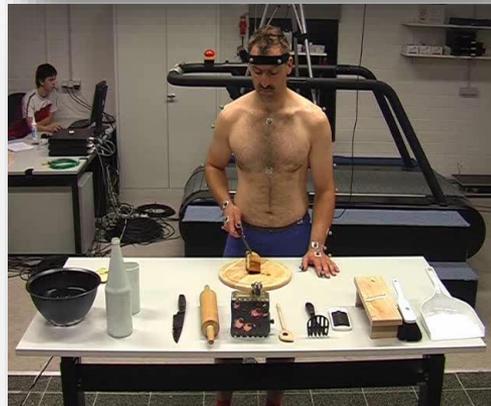
Stereo-Based 3D Human Motion Capture (HMC)

- Capture 3D human motion based on the image input from the cameras of the robot's head **only**
- **Approach**
 - Hierarchical Particle Filter framework
 - Localization of hands and head using color segmentation and stereo triangulation
 - Fusion of 3d positions and edge information
 - Half of the particles are sampled using inverse kinematics
- **Features**
 - Automatic Initialization
 - 30 fps real-time tracking on a 3 GHz CPU, 640x440 images
 - Smooth tracking of real 3D motion

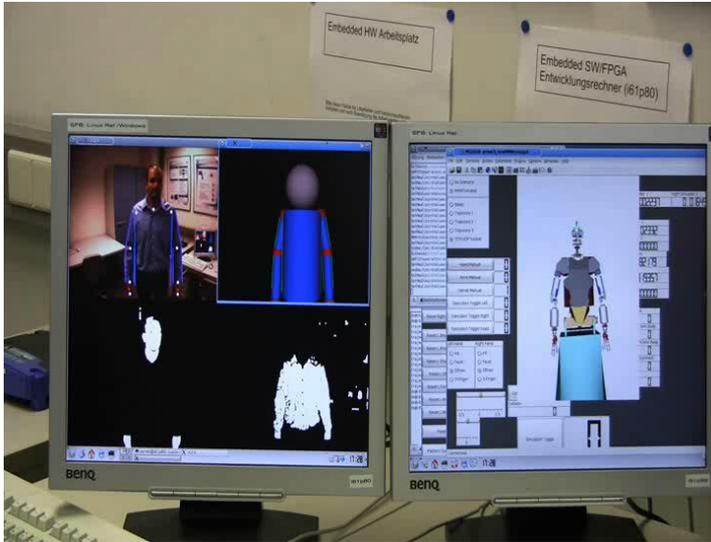


Motion Reproduction with MMM (II)

- Data from stereo-based marker-less human motion capture system
- Data from VICON system (SFB 588)



Motion Reproduction with MMM on ARMAR-IIIb



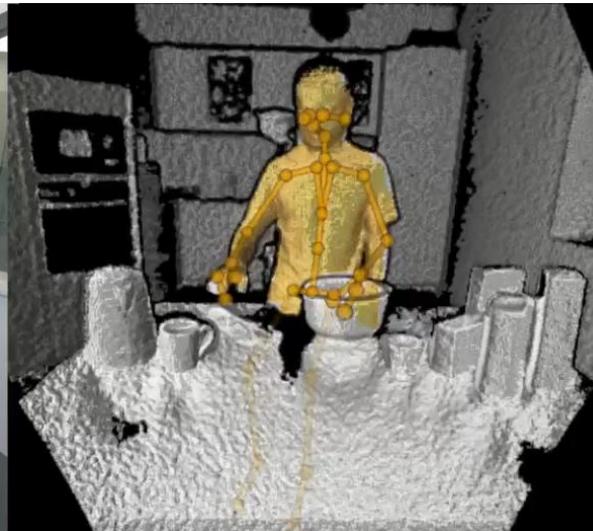
Azad, P., Asfour, T. and Dillmann, R., *Robust Real-time Stereo-based Markerless Human Motion Capture*, IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 700-707, December, 2008

Deep Learning based Human Pose Estimation

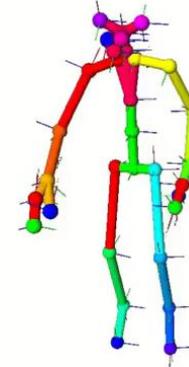
- Azure Kinect body tracking (in real-time) <https://learn.microsoft.com/en-us/azure/kinect-dk/body-sdk-download>
 - Deep Neural Network for human pose estimation
 - Nonlinear optimization for motion retargeting to MMM model



RGB



Depth



Human pose

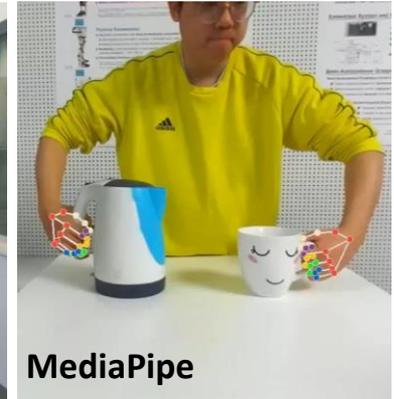


Motion retargeting

Deep Learning based Human Pose Estimation

Other deep learning-based approaches (in real-time):

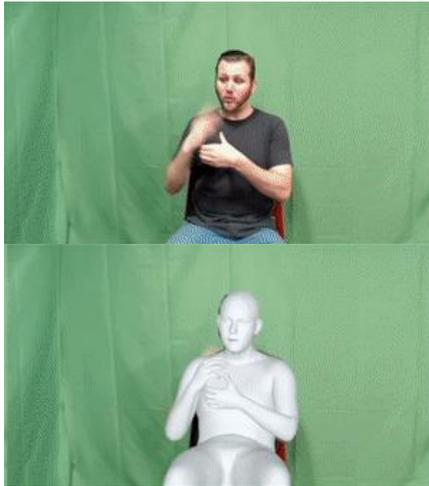
- OpenPose – CMU <https://github.com/CMU-Perceptual-Computing-Lab/openpose>
- MediaPipe – Google <https://developers.google.com/mediapipe>



Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y. A. „**OpenPose**: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields“, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019)
C. Lugaresi et. al, “**MediaPipe**: A Framework for Building Perception Pipelines,” *CoRR* (2019)

Deep Learning based Human Pose Estimation

- Mesh recoveries from monocular images (not real-time capable)
- Whole-body mesh recovery (can also output the skeleton or only the hand mesh)



Whole-body mesh recovery [1]



Hand mesh recovery [2]

[1] Lin, Jing, et al. "One-Stage 3D Whole-Body Mesh Recovery with Component Aware Transformer." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023. <https://osx-ubody.github.io/>

[2] Lin, Kevin, Lijuan Wang, and Zicheng Liu. "Mesh graphormer." *ICCV*. 2021.

Observation

■ Human motion capture techniques

■ Marker-based systems

- Passive markers (VICON)
- Active markers
- IMU based

■ Markerless systems

- From Images (RGB, RGB-D)
- Exoskeletons
- Deep learning-based approaches

■ Observation of object motions

- Object detection and segmentation
- Object pose estimation and tracking
- Object feature correspondence detection
- Scene graph

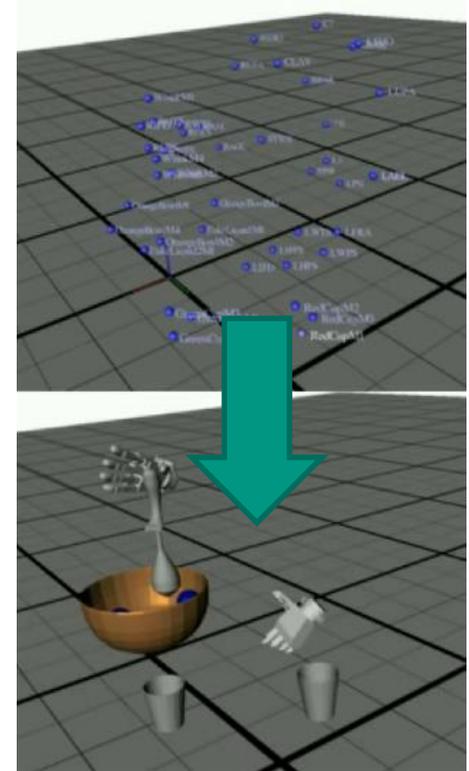
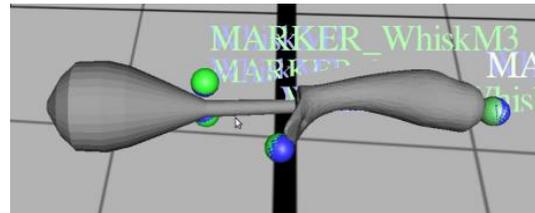
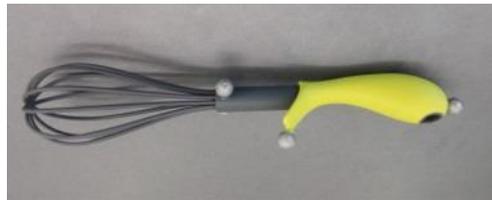
Observation of Object Motions

- Marker-based object motion capture
 - Pros: high-quality and precise motions
 - Cons: expensive equipment, time-consuming postprocessing (manual annotation, labeling and fine-tuning), 3D object models

- Markerless object motion capture
 - Common techniques required for object motion capture
 - Object detection and segmentation
 - Object pose estimation and tracking
 - Object feature correspondence detection
 - Scene graph generation

Marker-based object motion capture

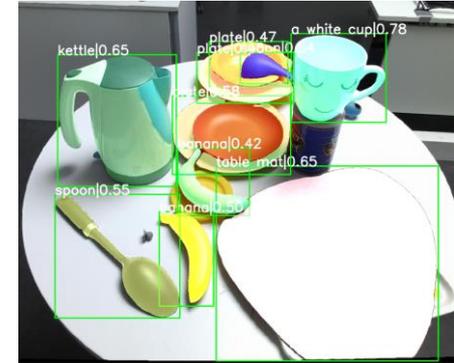
- Recordings of **action sequences** with marker-based motion capture
- Conversion to **6D object pose trajectories** with simplified MMM (Master Motor Map) models
- **Input** for segmentation algorithm (see later)



Object Detection and Segmentation

■ Object detection

- **Closed-set:** only detects object categories that appear in the training dataset
- **Open-set:** also detects object categories that are not in the training dataset

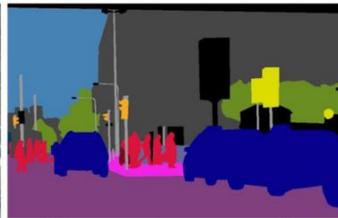


■ Segmentation

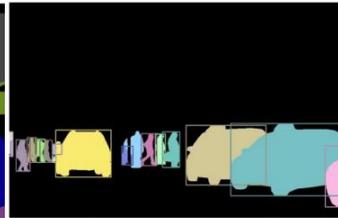
- **Instance segmentation:** assigns each object instance a unique identifier
- **Semantic segmentation:** assigns each object category a unique label
- **Panoptic segmentation:** a combination of instance and semantic segmentation



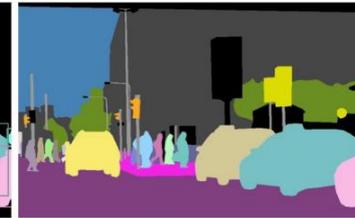
(a) image



(b) semantic segmentation



(c) instance segmentation



(d) panoptic segmentation

<http://www.cs.cornell.edu/courses/cs6670/2021fa/lec08-segmentation.pdf>

Object Detection and Segmentation

- Real-world examples using Grounding DINO and Segment Anything

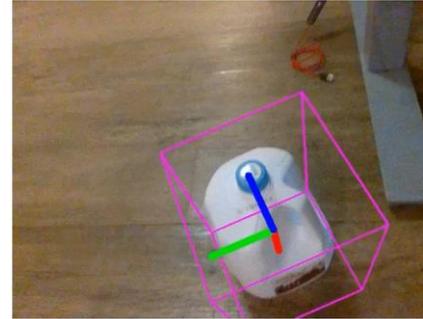
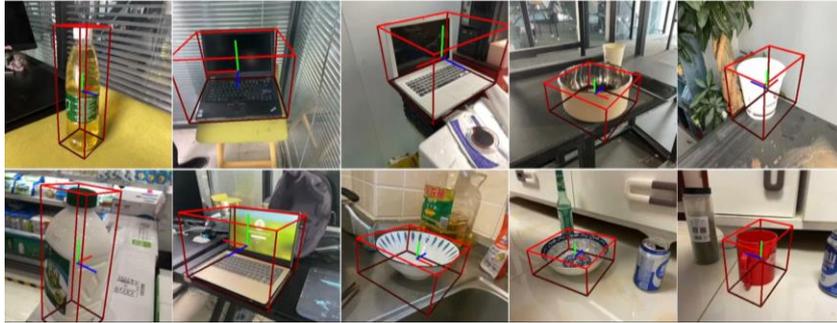


Liu, Shilong, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection." *arXiv preprint arXiv:2303.05499* (2023).

Kirillov, Alexander, et al. "Segment anything." *arXiv preprint arXiv:2304.02643* (2023).

Object Pose Estimation and Tracking

- Object pose estimation
 - Estimating the object 6D pose, spatial dimension given the RGB image
- Object-level SLAM, e.g. BundleSDF
 - Simultaneously optimize and track the object pose and reconstruct the object shape (e.g. using signed distance function, SDF)



BundleSDF: 10 Hz

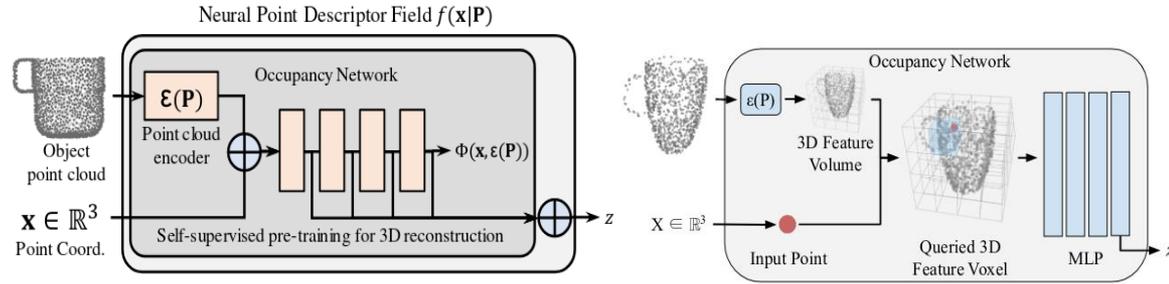
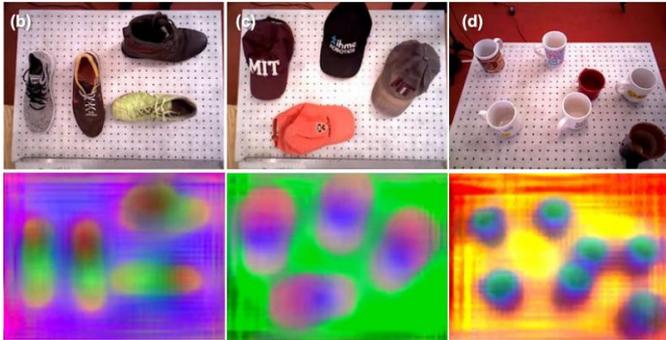


Zhang, Kaifeng, et al. "Self-Supervised Geometric Correspondence for Category-Level 6D Object Pose Estimation in the Wild." preprint arXiv:2210.07199 (2022).
 Wen, Bowen, et al. "BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

Object Feature Correspondence Detection

Object spatial descriptor obtained from self-supervised learning

- RGB-based approach
 - Maps RGB images to descriptor space
 - Dense Object Net (DON) [1]
- Point-cloud-based approach
 - Maps 3D query points to descriptor space conditioned on partially observed point cloud
 - Neural Descriptor Field (NDF) [2] encodes the global shape
 - Local-NDF (L-NDF, [3]) encodes local geometry into a grid-structured feature voxels



[1] P. Florence, L. Manuelli, and R. Tedrake, "Dense Object Nets: Learning dense visual object descriptors by and for robotic manipulation," CoRL, 2018

[2] A. Simeonov, et. al, "Neural Descriptor Fields: SE(3)-equivariant object representations for manipulation," ICRA, 2022

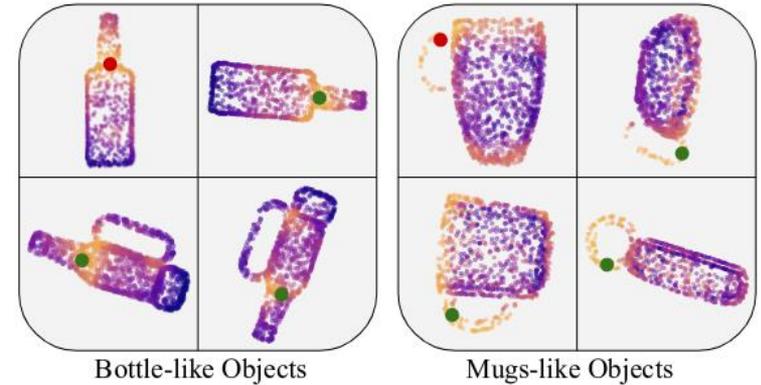
[3] E. Chun, et. al, "Local Neural Descriptor Fields: Locally Conditioned Object Representations for Manipulation." Preprint 2023. <http://arxiv.org/abs/2302.03573>

Object Feature Correspondence Detection

- Object spatial descriptor can be used to find **dense correspondences** or **local frames of reference** between categorical objects or similar parts
- It facilitates generalizable task constraints and manipulation skills



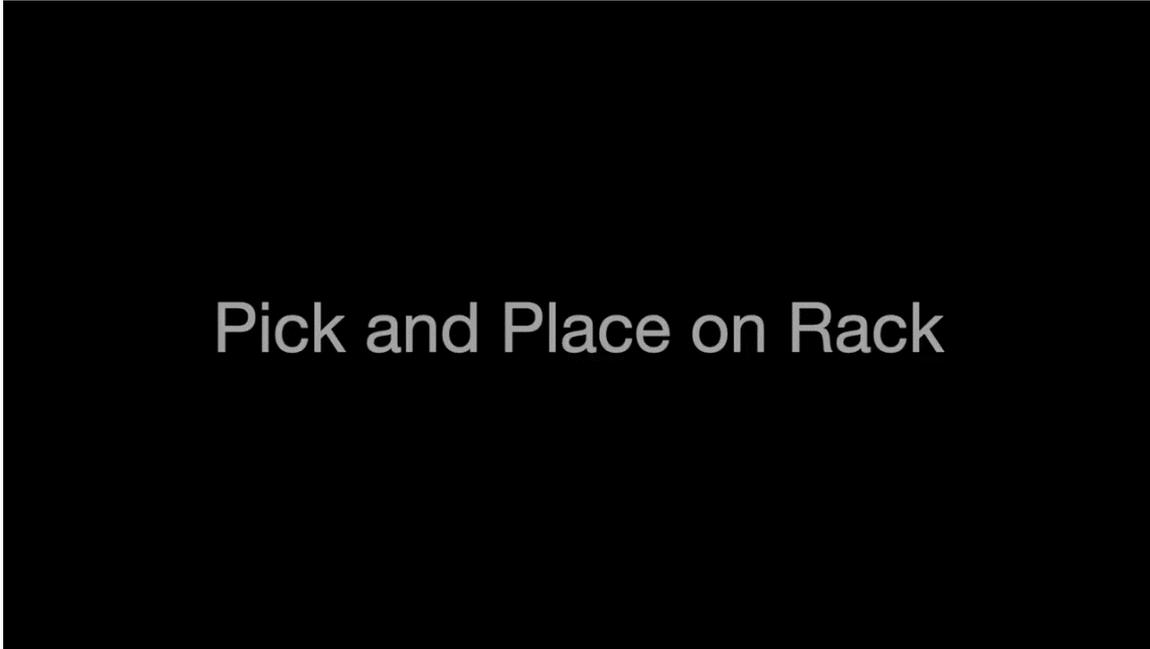
Examples of DON



Examples of L-NDF

Object Feature Correspondence Detection

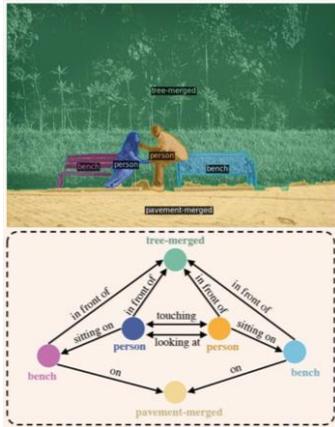
- Learning from human demonstration using L-NDF and transfer to categorical objects



Pick and Place on Rack

Scene Graph

- Scene graph is a **graph-structured representation of objects and their pairwise relationships**



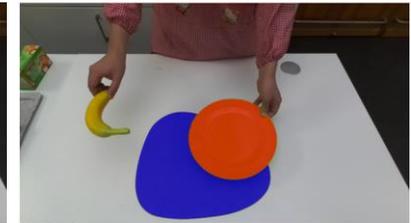
Panoptic scene graph (PSG)



Red holding Blue



Red holding Blue



Red on Blue



Red on Blue



Red beside Blue



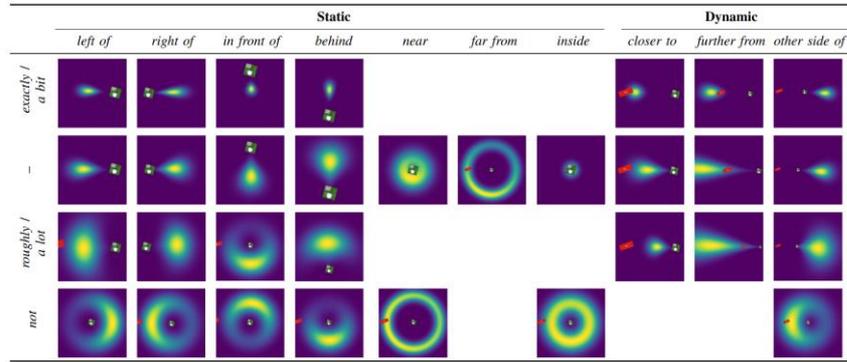
Red on Blue

Relate Anything (Segment Anything + PSG)

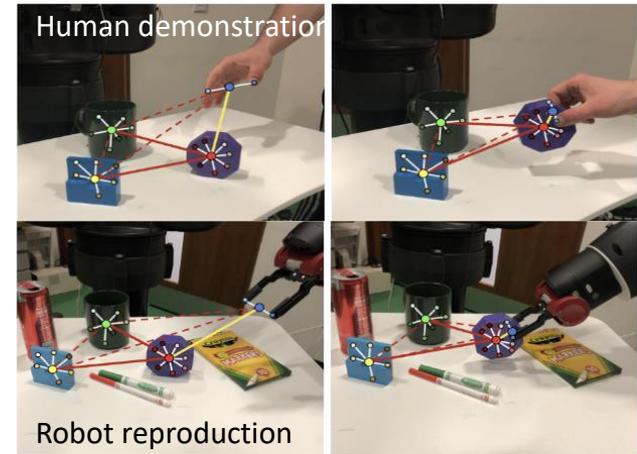
Yang, Jingkang, et al. "Panoptic scene graph generation." ECCV, 2022.
Relate Anything: <https://github.com/Luodian/RelateAnything>

Scene Graph

- Object relationship can be **static** or **dynamic**
- Scene graph can encode both **symbolic** and **subsymbolic information**
 - Symbolic:** spatial relation
 - Subsymbolic:** object pose, distribution of objects



Static and dynamic object relation



Visual Entity Graph (VEG)

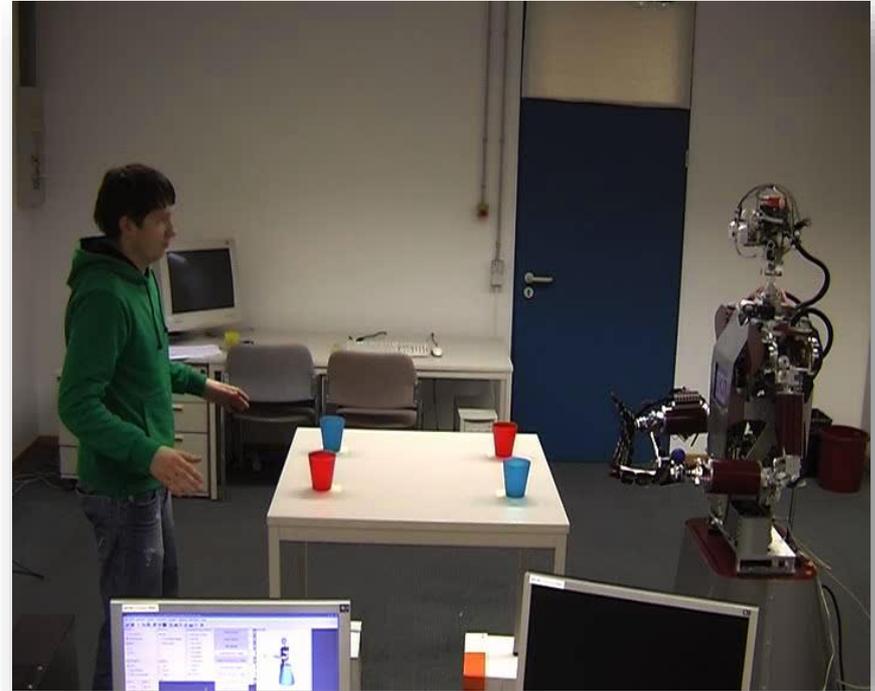
Kartmann, Rainer, et al. "Representing spatial object relations as parametric polar distribution for scene manipulation based on verbal commands." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

Sieb, Maximilian, et al. "Graph-structured visual imitation." Conference on Robot Learning. PMLR, 2020.

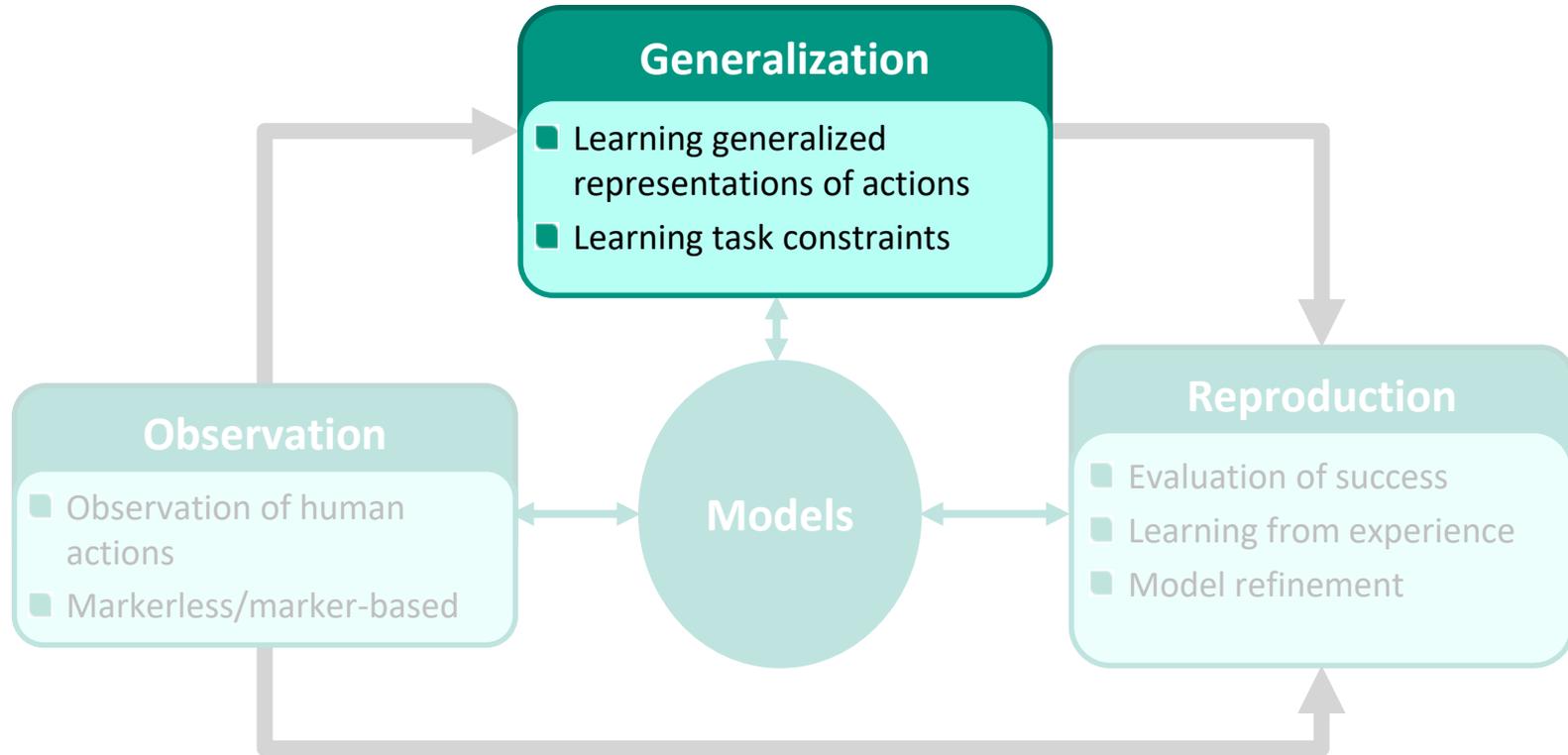
Human Motion to Robot Motion

- Tracking of human and object motion
- Visual servoing for grasping

Generalization?



Learning From Human Observation



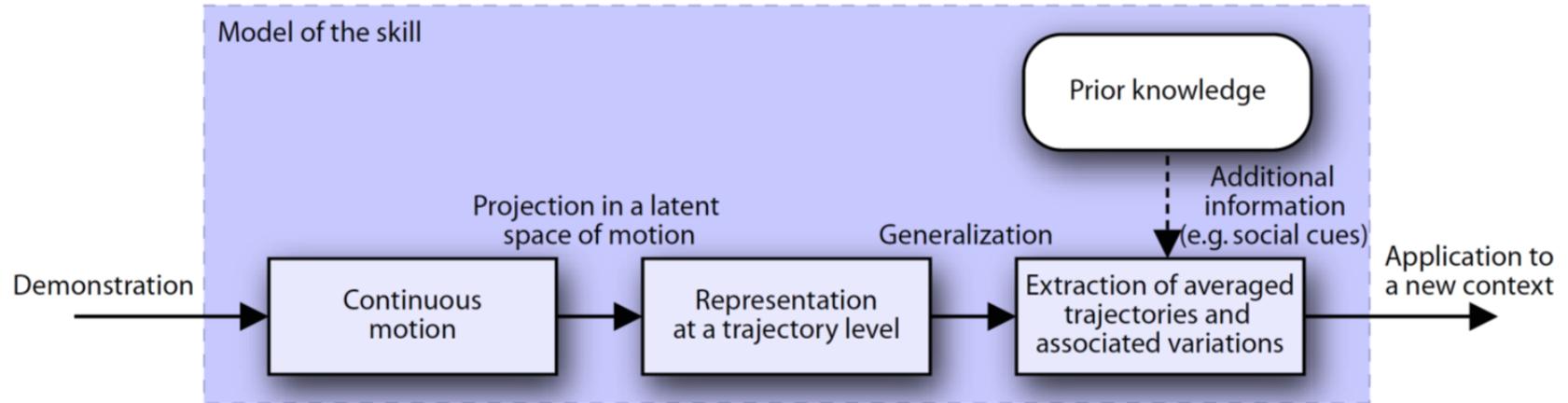
Generalization

- Decompose the demonstration into pieces (segmentation)
- Representation of these pieces in a parameterizable way

Two Levels of Representation

Red: relevant for the exam

- **Trajectory level encodings:** A low-level representation of the skill, taking the form of a non-linear mapping between sensory and motor information

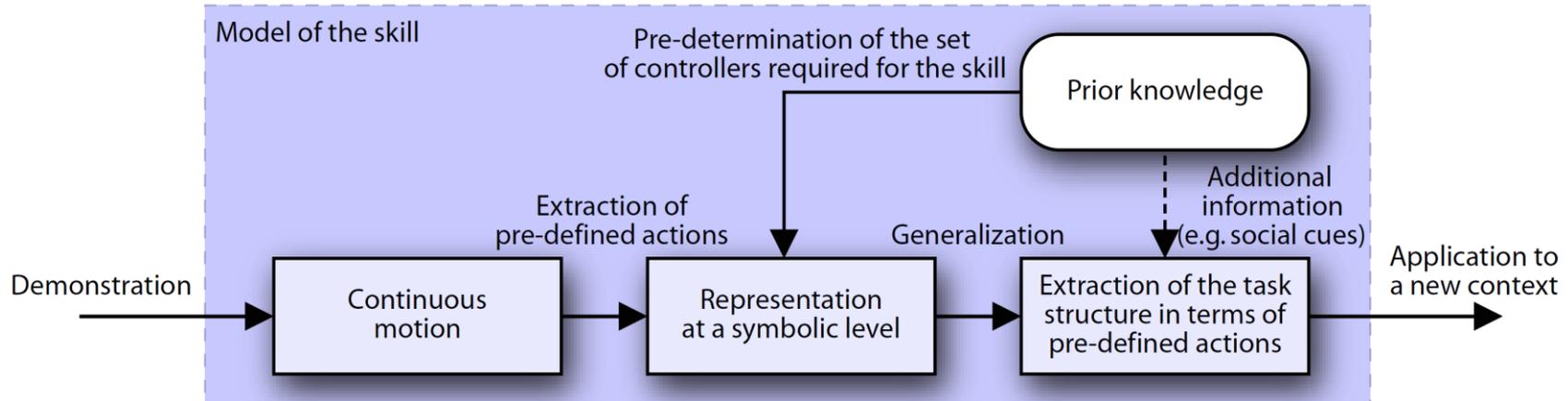


Billard, A., Calinon, S. and Dillmann, R. (2016). Learning From Humans. Siciliano, B. and Khatib, O. (eds.). Handbook of Robotics, 2nd Edition, Chapter 74, pp. 1995-2014. Springer

Two Levels of Representation

Red: relevant for the exam

- **Symbolic level encodings:** A high-level representation of the skill that decomposes the skill in a sequence of action-perception units



Billard, A., Calinon, S. and Dillmann, R. (2016). Learning From Humans. Siciliano, B. and Khatib, O. (eds.). Handbook of Robotics, 2nd Edition, Chapter 74, pp. 1995-2014. Springer

Generalization

■ Trajectory Level: Generalization of movements

- Generic representation of motion which allows encoding different types of signals/gestures
- Addresses the “What is important to imitate?” question
- Does not allow to reproduce complicated high-level skills ☹️

■ Symbolic Level: Sequential organization of pre-defined elements (actions)

- Allows to learn hierarchies and rules
- Relies on large amount of prior knowledge to predefine symbolic representations (of actions)
- Requires to pre-define a set of basic controllers for reproduction ☹️

■ In both cases: **Segmentation is needed**

- Trajectory segmentation
- Task segmentation

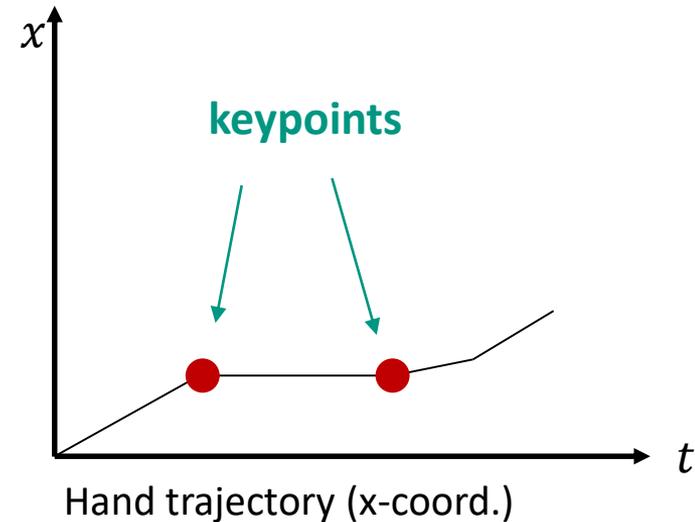
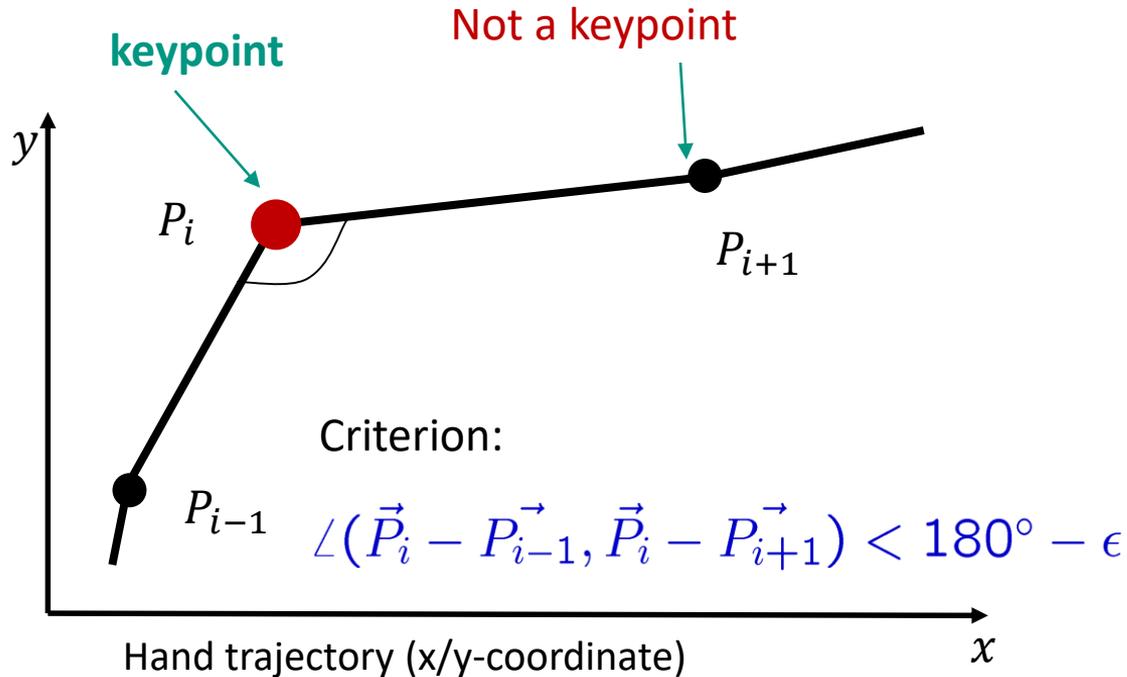
Segmentation of Human Demonstrations

Segmentation

- Segmentation means to **divide** a motion into **meaningful** segments, i.e., find prominent/distinctive points, **the key points**, of the demonstration
 - Examples: reaching phase of grasping, single steps of a walking motion
- "Meaningful" depends on the application and is subjective
 - Each step in a walking motion?
 - Segmenting between dancing and walking?
 - Repetition of an exercise?
- Input data
 - **Joint angle** trajectories (mostly used)
 - **Cartesian** trajectories of end-effectors
 - **Semantic relations** between objects and hands of demonstrator

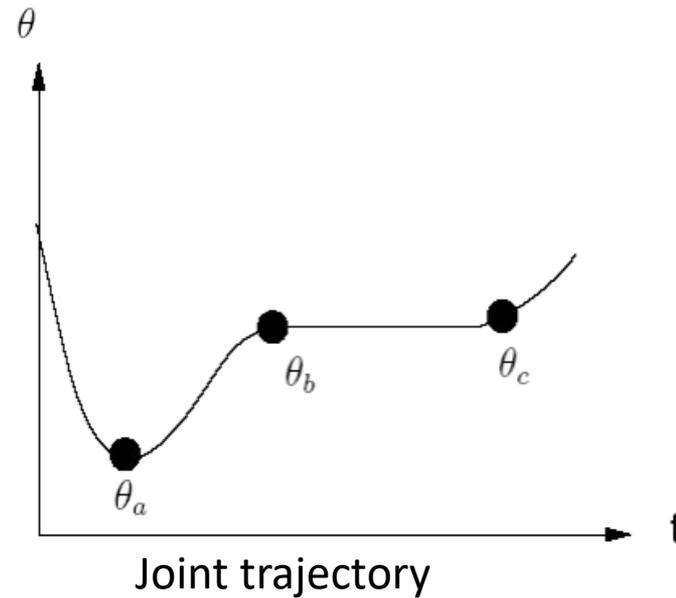
Keypoints (I)

Local minima and maxima and also pauses



Keypoints (II)

Local minimum and maximum and also pauses
(in position, velocity, ... trajectories)



Motion Segmentation

■ Approaches

- **Zero-Velocity Crossings** (Fod et al., 2002)
- **Principal Component Analysis** (Barbić et al., 2004)
- **Clustering** (Zhou et al., 2013)
- **Machine learning** (Lin et al., 2014)
- **Object relation changes** (Aksoy et al., 2011)
- **Object relation changes** and **motion characteristic** (Wächter and Asfour, 2015)
- ...

■ Applications

- Extraction of actions in a human demonstration (Segmentation into single manipulation action)
- Extraction of single activities, as walking, dancing, running, ...
- Rehabilitation progress evaluation (Segmentation into exercise repetition)
- ...

Zero-Velocity Crossings (ZVCs) – Idea

- Movement direction change implies joint angle velocity crossing zero
- If this happens, point in time is labelled as “zero-velocity crossing” (ZVC)
- **Assumption:** Motion primitives transition into each other when multiple ZVCs occur
- **Idea:** Segment, if several ZVCs occur simultaneously in a time window

A. Fod, M. J. Matarić, and O. C. Jenkins “Automated derivation of primitives for movement classification”
Autonomous robots, vol. 12, no. 1, pp. 39–54 (2002)

Zero-Velocity Crossings (ZVCs) – Algorithm

- For each degree of freedom: Identify all ZVCs
- If at least n joints have a ZVC within t seconds: **Segment**
- **Goal:** Identify movement primitives → discard segments with only minor movements / oscillations (defined by a threshold parameter m)
- Parametrization (in this work):
 - $n = 2$ (i.e., 2 joints of a 4 DoF arm)
 - $t = 300$ ms
 - m not reported (“empirically determined”)

A. Fod, M. J. Matarić, and O. C. Jenkins “Automated derivation of primitives for movement classification”
Autonomous robots, vol. 12, no. 1, pp. 39–54 (2002)

Zero-Velocity Crossings (ZVCs) – Discussion

■ Pros

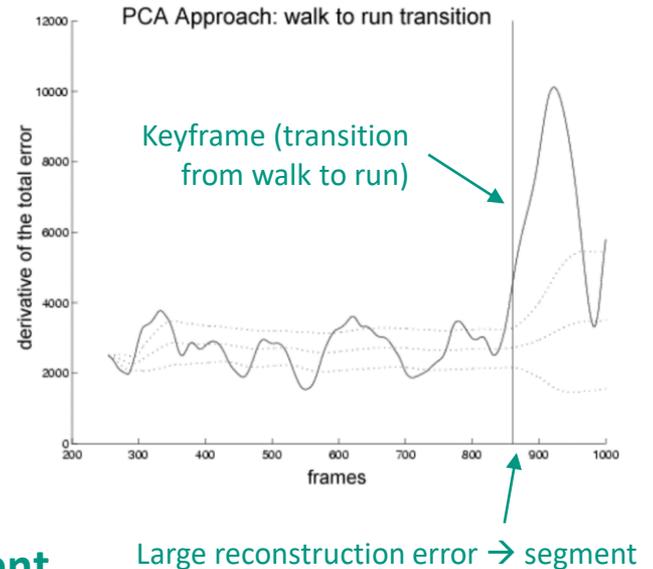
- Simple, easy to implement
- No model required
- Online capable

■ Cons

- Tends to miss segmentation points
- Does not find segments with smooth transitions
- Only tested on 4 DoF arm → Impractical for systems with higher number of DoF (whole-body motion)
- Unable to segment a whole class of specific movements
- Doesn't consider co-articulation
- Threshold parameters: t , m
- Segmentation not complete

Principal Component Analysis (PCA) – Idea

- **Goal:** Segment long motion sequences automatically into distinct behaviors (walk, jump, wash the window, wipe, cut, ...)
- **Idea:**
 - Perform PCA on sliding window
 - Explain following movement with identified principal components based on **a reconstruction error**
 - If reconstruction error exceeds threshold: **Segment**



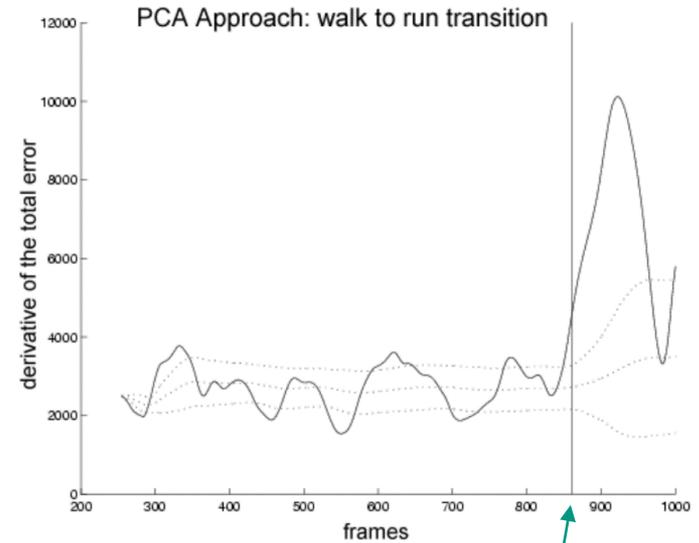
J. Barbić, A. Safonova, J. Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, “Segmenting motion capture data into distinct behaviors,” Proceedings of Graphics Interface, 2004, pp. 185–194

Principal Component Analysis (PCA) – Algorithm

- Input: Sequence of 56-dimensional vectors
- Error function e (reconstruction error when only r dimensions are considered)

■ Algorithm (in loop)

1. Use k frames and PCA to identify dimensionality r
2. With $i \triangleq$ current frame index:
 Compute derivative $d_i = e_i - e_{i-l}$ with l large enough to compensate noise
3. If d_i is more than three standard deviations from average: **Segment**, go to 1.
4. Move sliding window, go to 2.



Large reconstruction error, segment

Principal Component Analysis (PCA) – Discussion

■ Pros

- High accuracy
- No model required

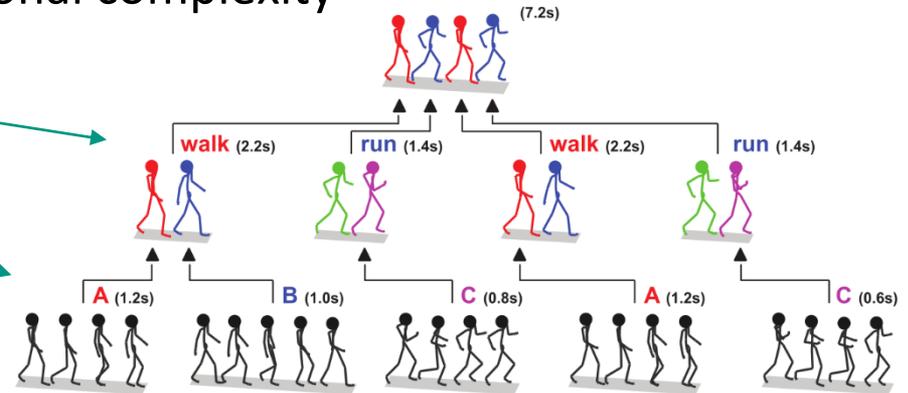
■ Cons

- Parameters: k, l, \dots
- Sliding window (delay-accuracy-tradeoff)
- Motion must be longer than $k + l$ frames
- Tested for high-level motions (walk, run, wash the window, ...)
- Tends to produce additional segmentation points

Hierarchical Aligned Cluster Analysis (HACA) – Idea

- **Idea:** motion segmentation as temporal clustering problem (time series data)
- Cluster motion segments based on their frame-wise similarities
- Two hierarchies to reduce computational complexity

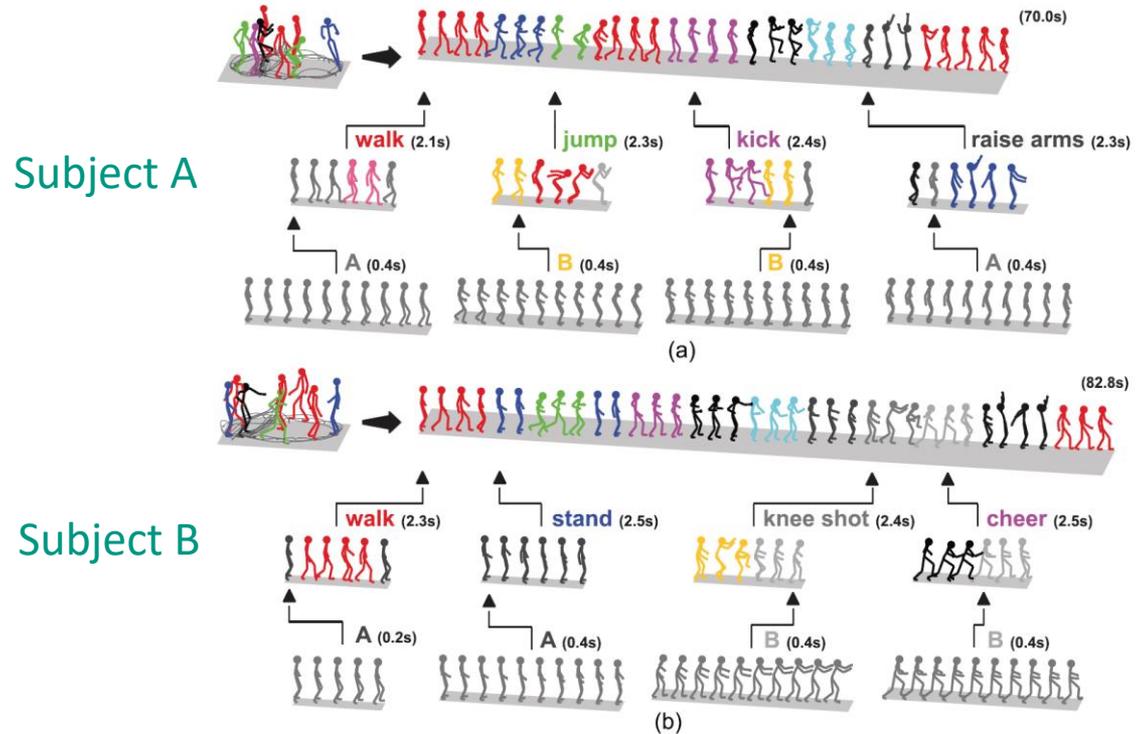
- Action level
(e.g., running vs. walking)
- Motion primitive level
- Several temporal scales



F. Zhou, F. de la Torre, and J. K. Hodgins. "Hierarchical Aligned Cluster Analysis for Temporal Clustering of Human Motion". Transactions on Pattern Analysis and Machine Intelligence 35, no. 3 (2013): 582–96.

Hierarchical Aligned Cluster Analysis (HACA) – Example

- Several actions (walk, jump, kick, , etc.) share common components at lower temporal granularity level



Hierarchical Aligned Cluster Analysis (HACA) – Discussion

■ Pros

- Unsupervised
- It is known which segments belong to the same cluster

■ Cons

- Offline
- Number of actions (= clusters) in the data should be known in advance
- Computationally expensive

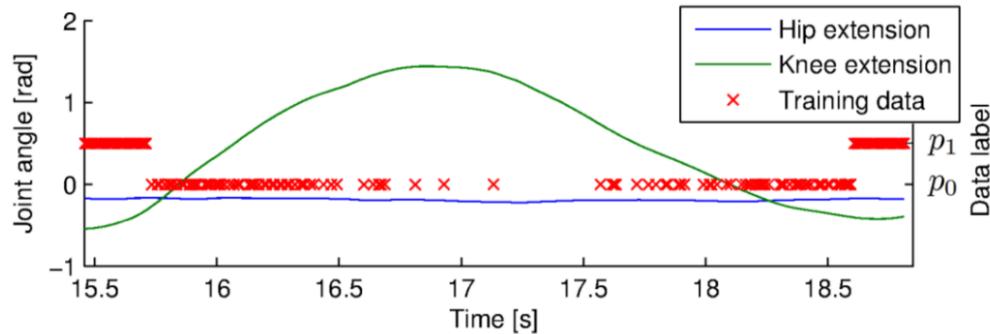
Segmentation Point Classification

- Previous approaches perform segmentation only when **one motion type transitions to another**
- Exercise supervision algorithms (e.g. in rehabilitation) requires the ability to segment when the **same** action is performed multiple times?
- **Idea:** Interpret motion segmentation as a binary classification problem

J. F.-S. Lin, V. Joukov, and D. Kulić, “Human motion segmentation by data point classification,” in IEEE Engineering in Medicine and Biology Conference, 2014, pp. 9–13

Segmentation Point Classification – Idea

- **Idea:** Interpret motion segmentation as a binary classification problem
- Data points are either **segmentation point (p_1)** or **non-segment point (p_0)**
- Train classifiers for p_0 and p_1 on annotated data

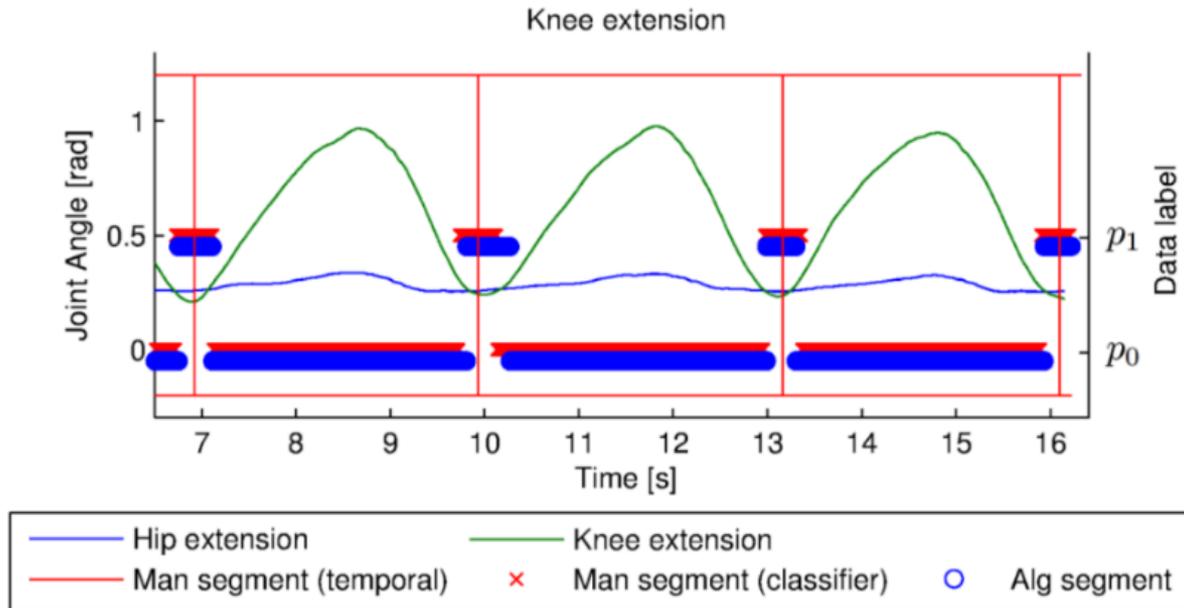


Segmentation Point Classification – Classifiers

- Different classifiers and additions to them were considered
- Dimensionality reduction
 - PCA
 - FDA (Fisher's Discriminant Analysis)
- Classifiers
 - k -nearest neighbors
 - Radial basis function
 - SVM + kernel trick
 - Neural network
- Aggregators
 - Boosting, Bagging, None

Segmentation Point Classification – Example Result

- Classify data points as for p_0 and $p_1 \rightarrow$ Temporal segmentation as classification problem



Segmentation Point Classification – Discussion

■ General results

- No big difference between PCA and FDA
- SVM seems to be sufficient; neural network not needed
- Aggregators have little influence

■ Pros

- Able to classify segment points of unknown motion primitives
- Classification usually fast after training phase, online

■ Cons

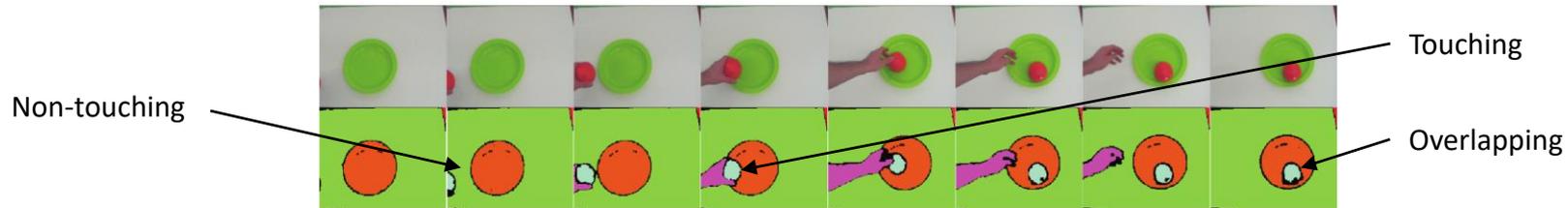
- Learning needs to be done offline
- Some overfitting can be observed

J. F.-S. Lin, V. Joukov, and D. Kulić, “Human motion segmentation by data point classification,” in IEEE Engineering in Medicine and Biology Conference, 2014, pp. 9–13

Object Relation Changes – Idea

Red: relevant for the exam

- **Assumption:** Contact relation changes between objects occur in the same order for the same actions
- **Idea:** For each action, build a representation of contact relation changes → identify executed action by comparing observation with action representations
- Relations between 2D image segments (colored areas) are considered since real contact relations are hard to recognize from vision:



E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. “Learning the Semantics of Object-Action Relations by Observation”. *International Journal of Robotics Research (IJRR)* 30, no. 10 (2011): 1229–49.

Object Relation Changes – Approach (I)

- Representation is called **Semantic Event Chain (SEC)**

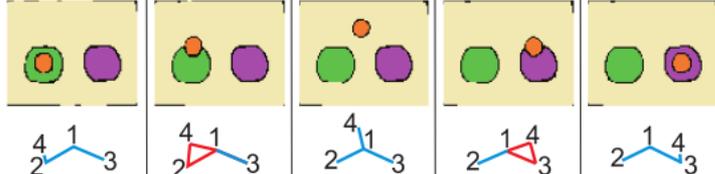
- $\rho_{x,y} \triangleq$ Relation type between x and y
 - “non-touching” (0), “overlapping” (1, blue), and “touching” (2, red)

- Identify 2D image segments

- Evaluate the relations for each pair of image segments \rightarrow Add column to SEC

- For each change in relations \rightarrow Add new column to SEC

SEC in tabular form



$\rho_{2,1}$	1	2	1	1	1
$\rho_{3,1}$	1	1	1	2	1
$\rho_{4,1}$	0	2	1	2	0
$\rho_{3,2}$	0	0	0	0	0
$\rho_{4,2}$	1	2	0	0	0
$\rho_{4,3}$	0	0	0	2	1

E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. “Learning the Semantics of Object-Action Relations by Observation”. International Journal of Robotics Research (IJRR) 30, no. 10 (2011): 1229–49.

Object Relation Changes – Approach (II)

- **Goal:** Learn SECs model which represent an action by searching for similar common rows and columns observed in all demonstrations
- Use a similarity measure defined on SECs for action recognition → Most similar model SEC is most likely the observed action

E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. “Learning the Semantics of Object-Action Relations by Observation”. International Journal of Robotics Research (IJRR) 30, no. 10 (2011): 1229–49.

Object Relation Changes – Discussion

■ Pros

- Intuitive approach
- Lightweight, can be incrementally learned
- Simultaneous action segmentation and recognition
- Many additions to this work in consecutive publications

■ Cons

- An action must be completed before it can be recognized
- Can suffer from bad initial examples when learned online
- Highly dependent on perfect image segmentation

E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. “Learning the Semantics of Object-Action Relations by Observation”. *International Journal of Robotics Research (IJRR)* 30, no. 10 (2011): 1229–49.

Task Segmentation Based on Object-Hand Relations

Hierarchical Segmentation

Human Demonstration



Converted Demonstration



Hierarchical Segmentation

No contact	Cup in left hand			No contact
Grasp	Lift	Pour	Place	Retreat

Red: relevant for the exam

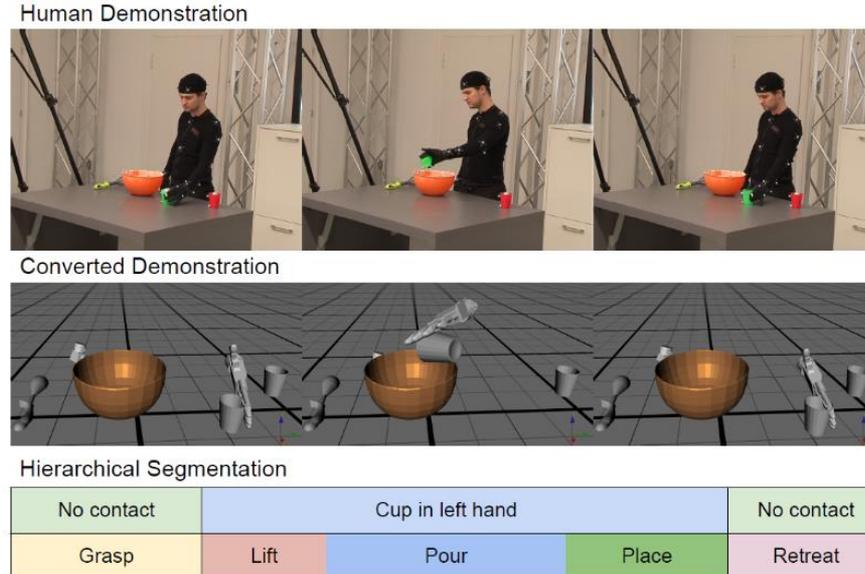
M. Wächter and T. Asfour, Hierarchical Segmentation of Manipulation Actions based on Object Relations and Motion Characteristics, International Conference on Advanced Robotics (ICAR), July, 2015

M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter and R. Dillmann, Action Sequence Reproduction based on Automatic Segmentation and Object-Action Complexes, IEEE/RAS International Conference on Humanoid Robots (Humanoids), October, 2013

Hierarchical Action Segmentation

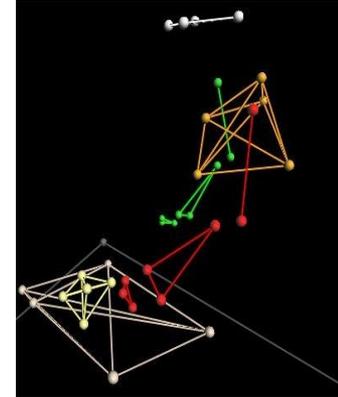
- Segmentation of human demonstration on two levels
 - **Semantic** segmentation based on object relation changes
 - **Motion** segmentation based on trajectory characteristics

Wächter and Asfour, 2015



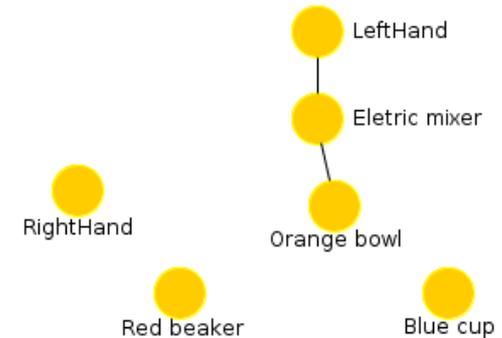
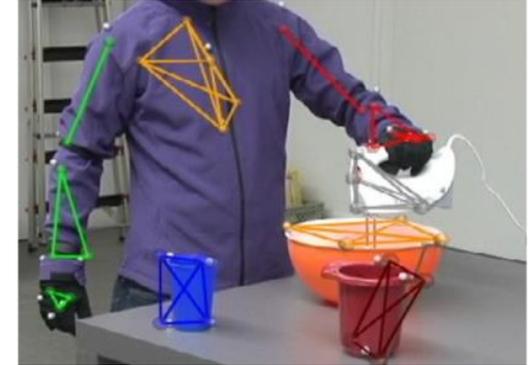
Capturing of the Human Demonstration

- Human motion capturing with VICON
- The human and all objects have several **markers** attached
- All markers are **labeled** and **grouped** by the attached object
- Extraction of marker trajectories



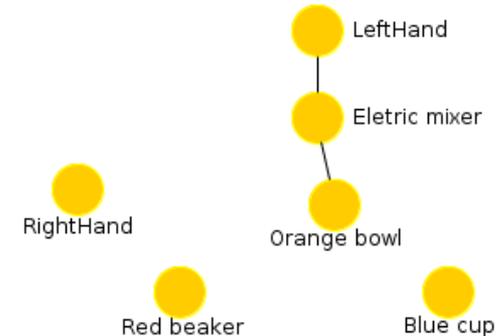
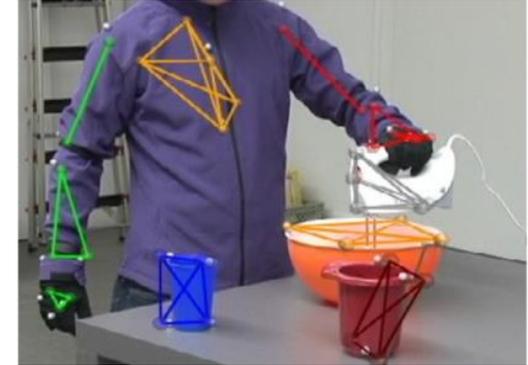
Action Segmentation: Idea (I)

- Use **contact/non-contact relations** to segment the action, i.e. to extract important key frames out of object interactions
- Cartesian distance of markers is used to **detect contact and key frames**
- Result: segmentation of an action sequence into several **action primitives**
 - E.g., “Preparing a dough” is divided into *grasping*, *pouring*, *placing*, etc.
- But these action primitives are still unlabeled



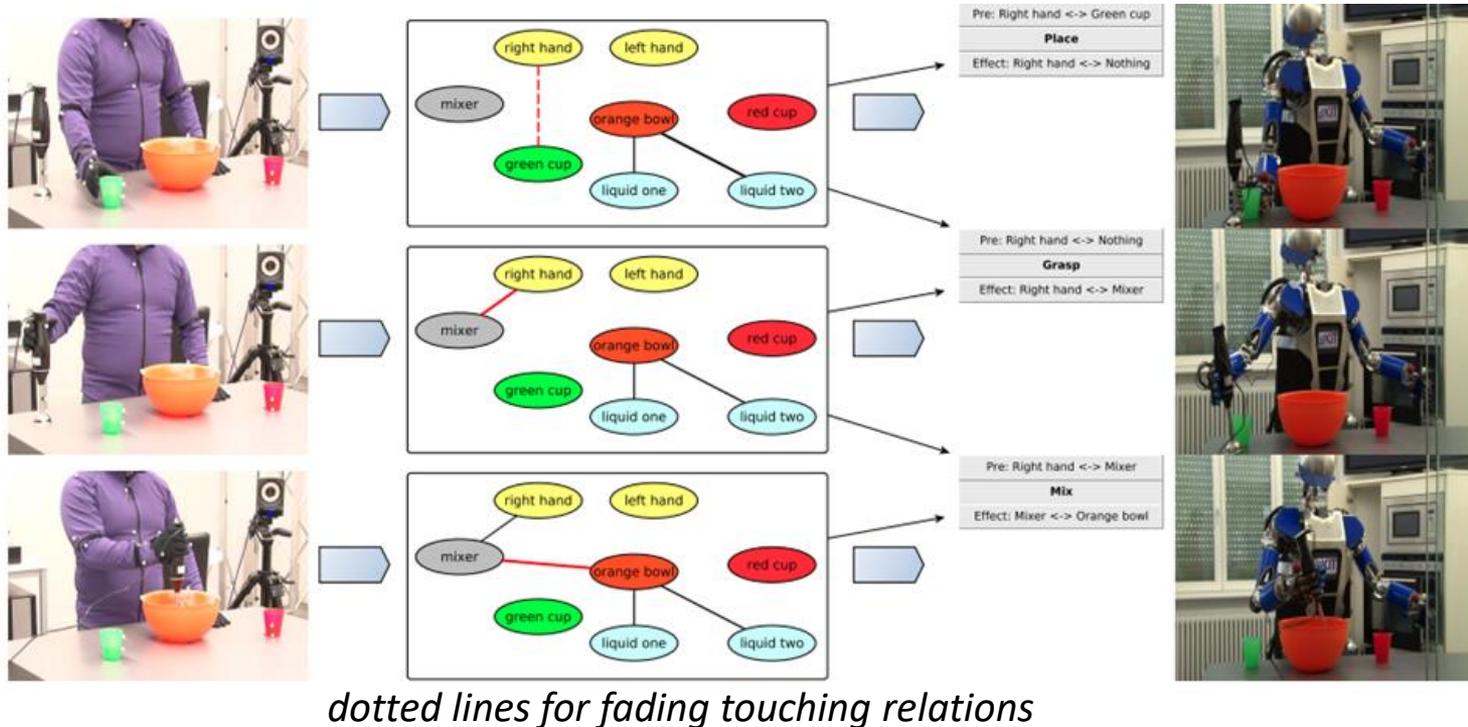
Action Segmentation: Idea (II)

- **Object relations** support the extraction of **general pre-/post-conditions** of an action primitive
 - Grasping pre-condition:
Lefthand no contact
 - Grasping post-condition:
LeftHand touches BlueCup
 - **BlueCup is grasped**



Action Segmentation – World State Examples

World state and pre-/post-conditions



The Whole Process

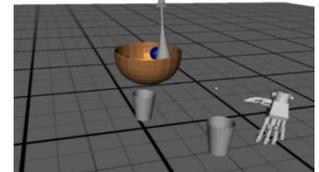
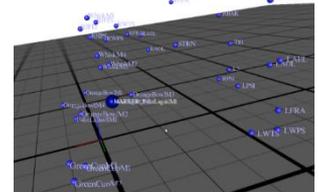


Detecting Contact between Hand and Object

- Using vision: difficult
- Using haptics: require contact sensors on the human hand

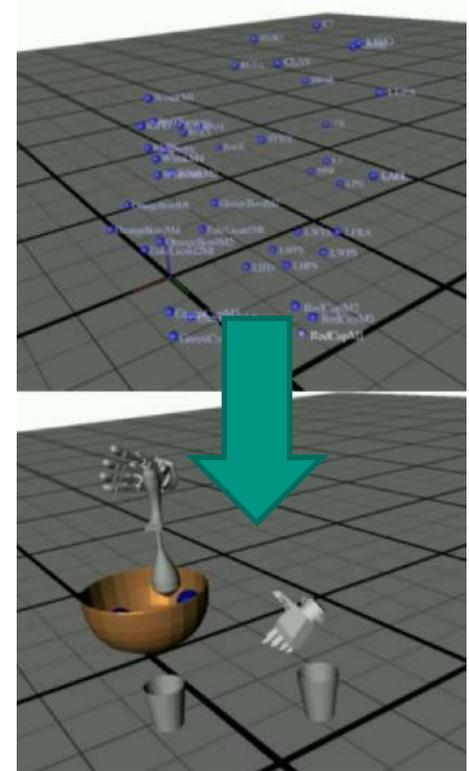
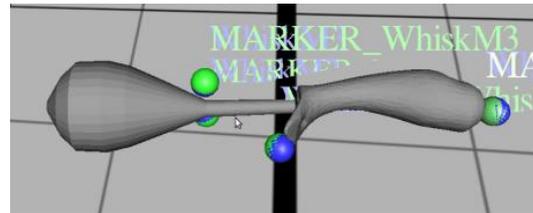
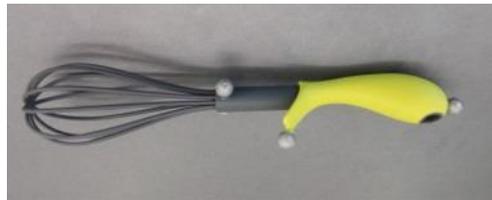
Instead:

- Mapping of marker representation into geometric model representation
 - 3D mesh model for all objects with virtual markers
 - Registration of recorded markers with virtual markers for 6D pose estimation
 - 6D trajectories analyzed by mesh collision detection algorithms



6D Object Trajectories

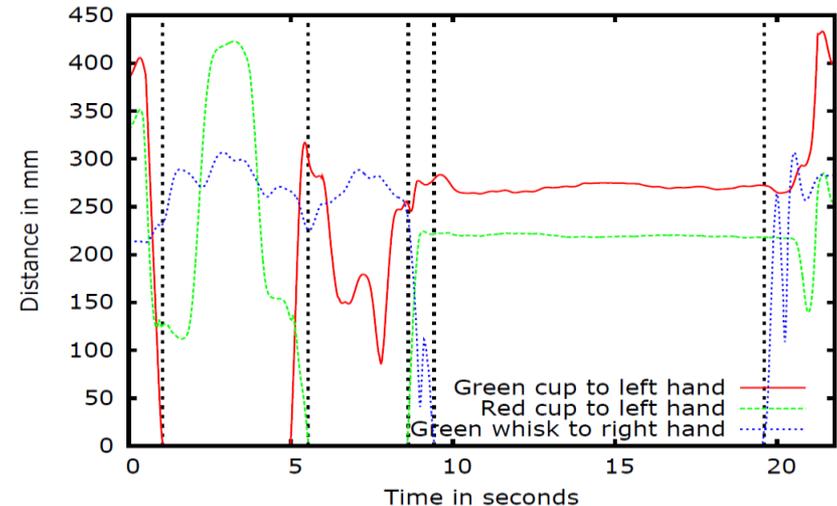
- Recordings of **action sequences** with marker-based motion capture
- Conversion to **6D object pose trajectories** with simplified MMM (Master Motor Map) models
- Input** for segmentation algorithm



Top Level: Semantic Segmentation



- Extraction of hand-objects and object-object **contact relation changes** using **3D mesh models** and collision detection algorithms

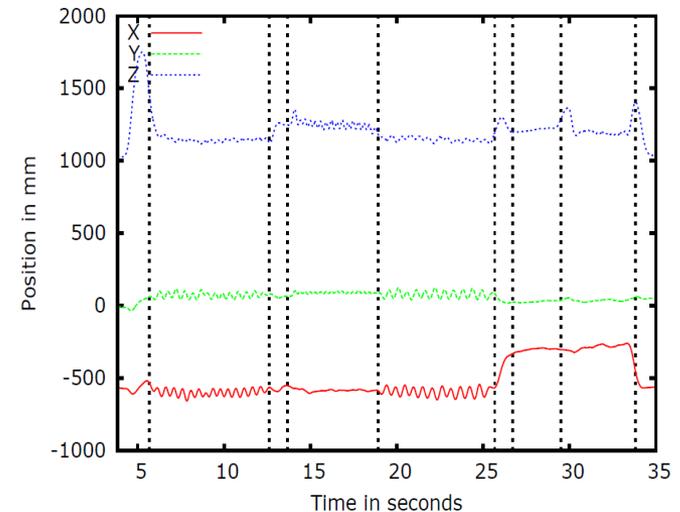


Bottom Level: Motion Segmentation

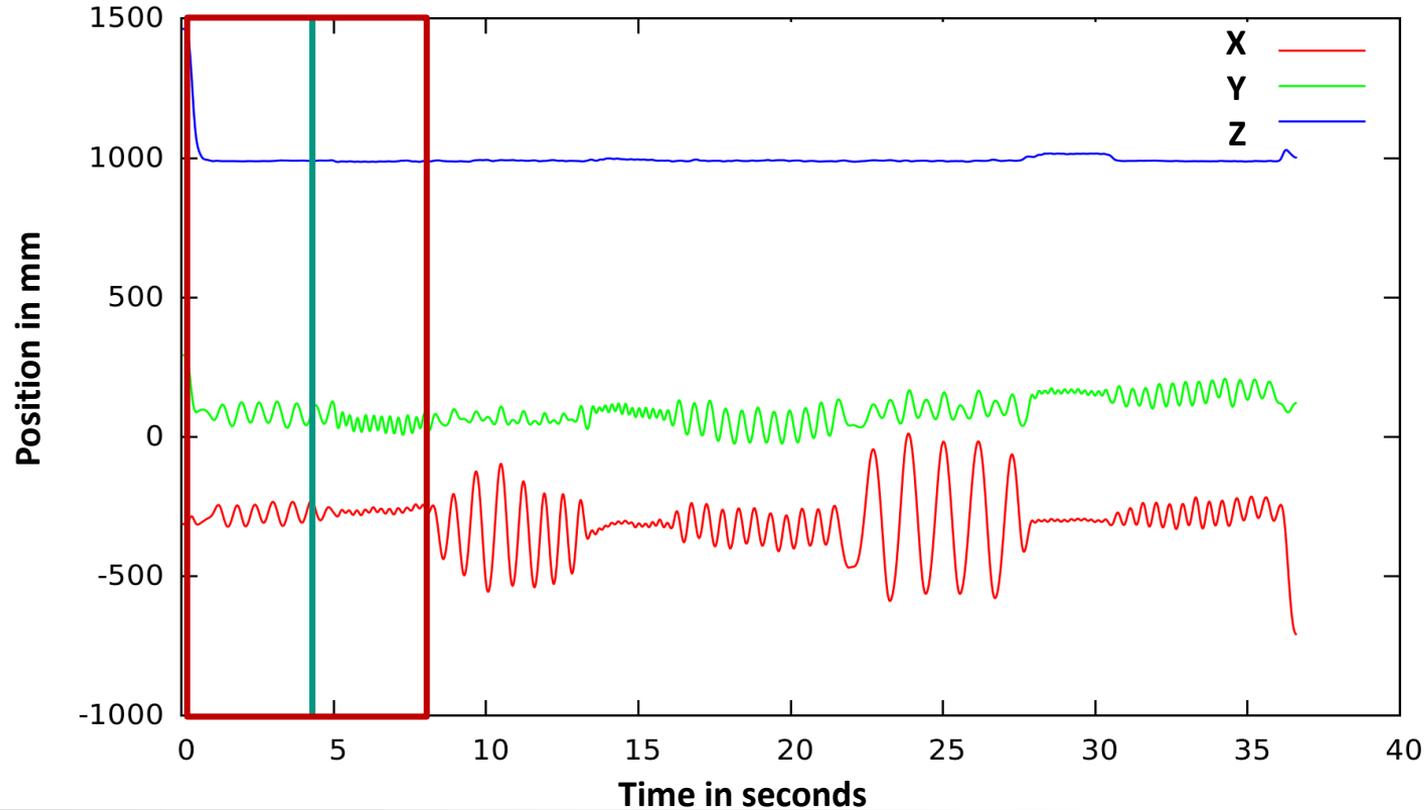


- Segmenting of semantic segments into **most distinctive parts** based on the motion characteristic

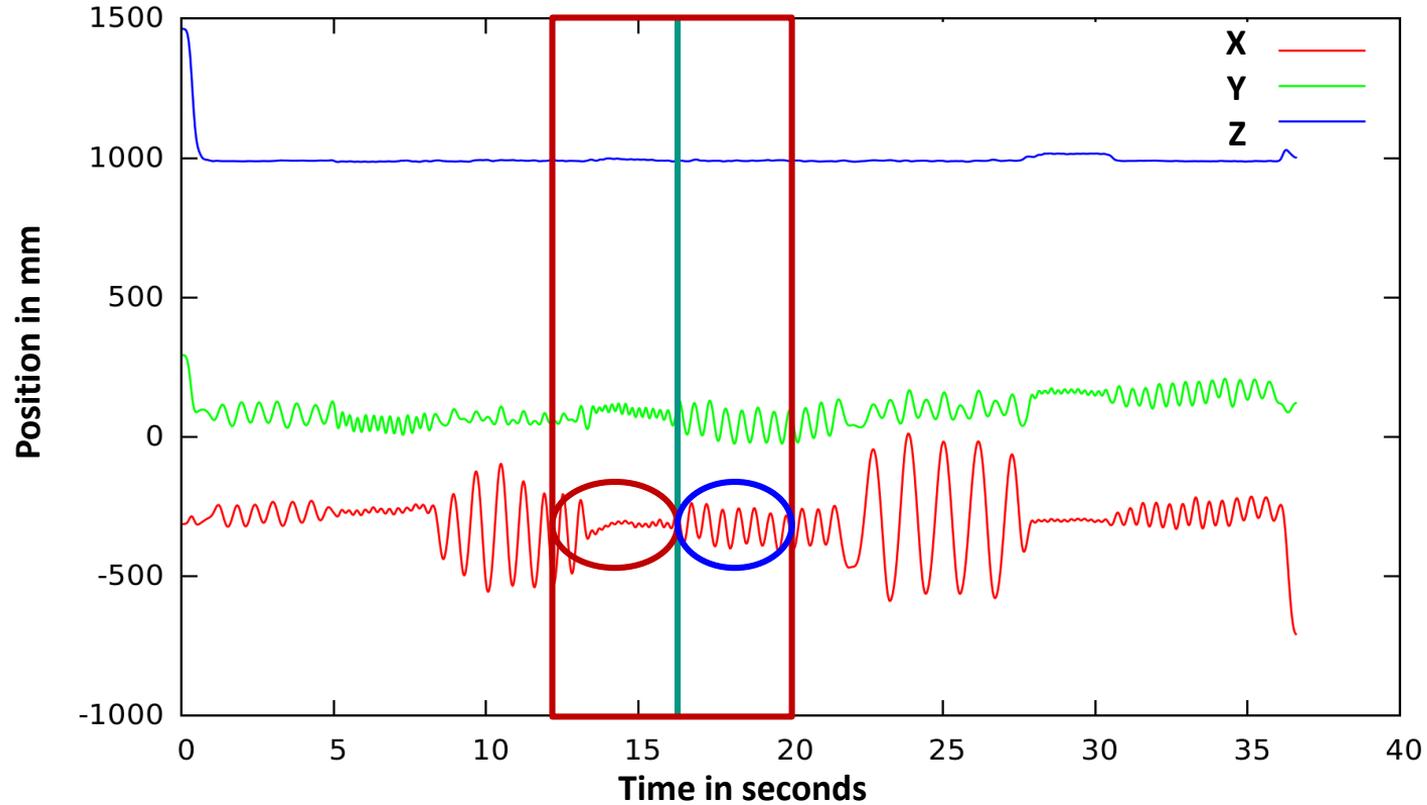
- New **heuristic** based on **acceleration profile** to segment motion
- Iterative search for best key frame in the current semantic segment
- Recursive segmentation until segments either **too small** or **too similar**



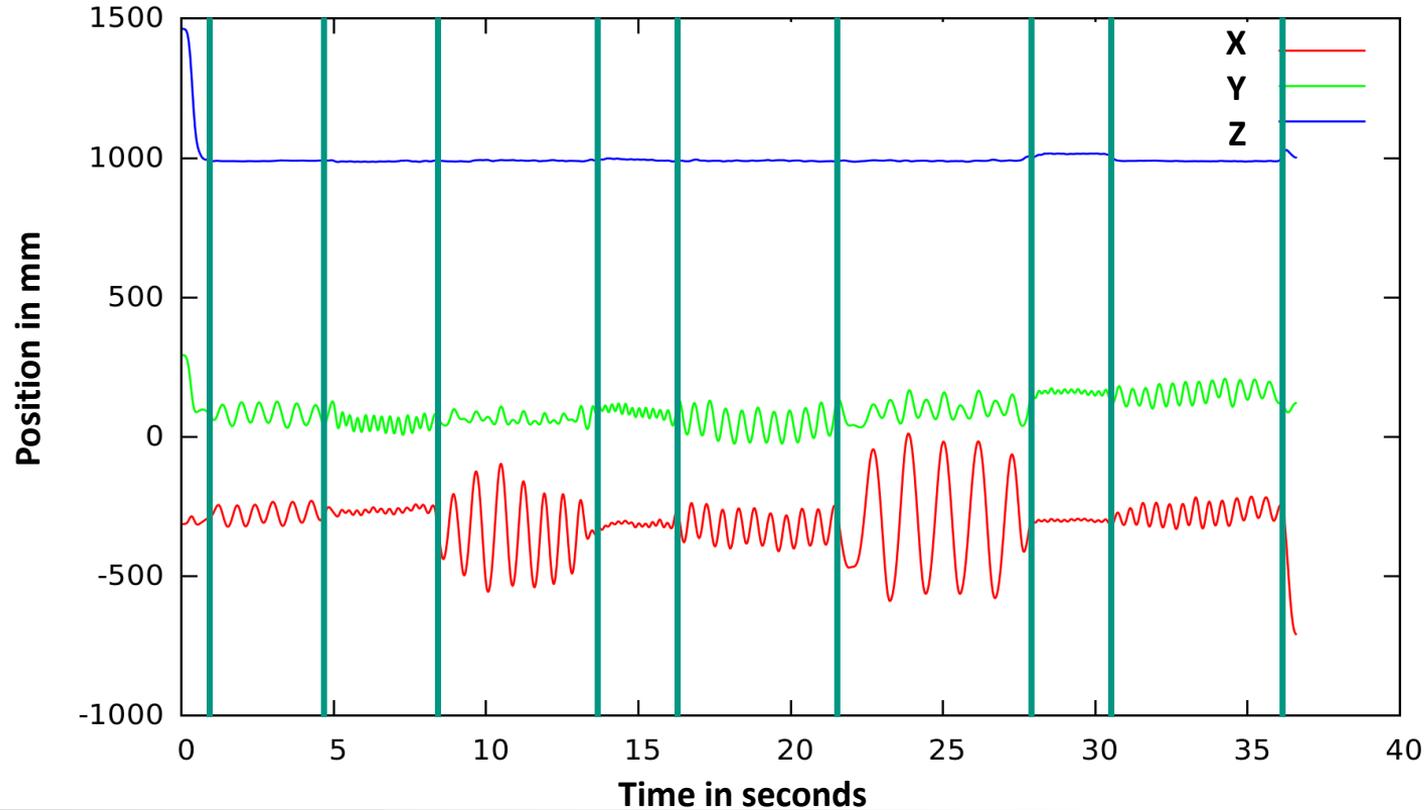
Motion Characteristics Heuristic



Motion Characteristics Heuristic



Motion Characteristics Heuristic



Motion Characteristics Heuristic (IV)

- **Heuristic** based on the **acceleration profile**, measuring the **length** of the **acceleration curve**

$$A_d(t) = |a_d(t + 1) - a_d(t)|$$

$$s_{l,d}(t_c) = \sum_{t=t_c-\frac{w}{2}}^{t_c-1} A_d(t) \left(\frac{\hat{U}_l}{\hat{U}_r} \right)^2$$

$$s_{r,d}(t_c) = \sum_{t=t_c}^{t_c+\frac{w}{2}-1} A_d(t) \left(\frac{\hat{U}_r}{\hat{U}_l} \right)^2$$

Motion Characteristics Heuristic (V)

- **Iterative search** for best candidate key frame with a **sliding window**
 - Comparison of segments left and right of candidate with score function s
 - Key frame quality: $q_d = \begin{cases} \frac{s_{l,d}}{s_{r,d}} & s_{l,d} > s_{r,d} \\ \frac{s_{r,d}}{s_{l,d}} & s_{l,d} \leq s_{r,d} \end{cases}$
- **Recursive** subdividing until **segment size** or **quality** too small

Evaluation of Segmentation Results

- New metric for assessment of segmentation results: **Mean squared error** with **penalties** for missing and additional key frames

$$e = (m + f) * p + \sum_i \min_j (k_{r,i} - k_{f,j})^2$$

m: missed key frames
f: false positives
p: penalty

- **Comparison to manual reference segmentation**

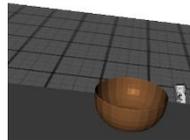
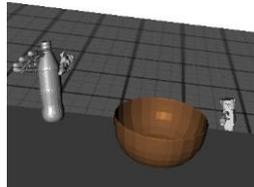
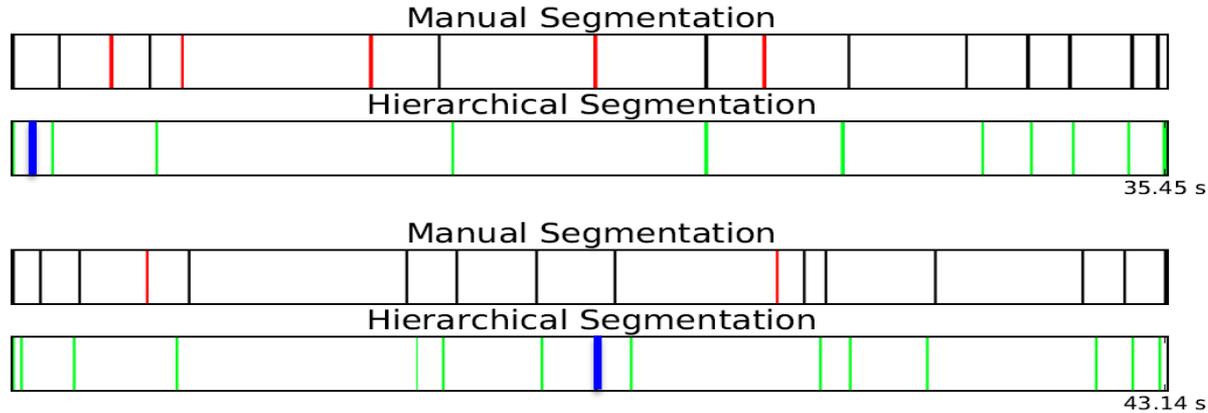
- HS: Hierarchical Segmentation
- ZVC: Zero Velocity Crossing
- PCA: Principal Component Analysis

- 13 action sequences à ≈ 30 seconds:

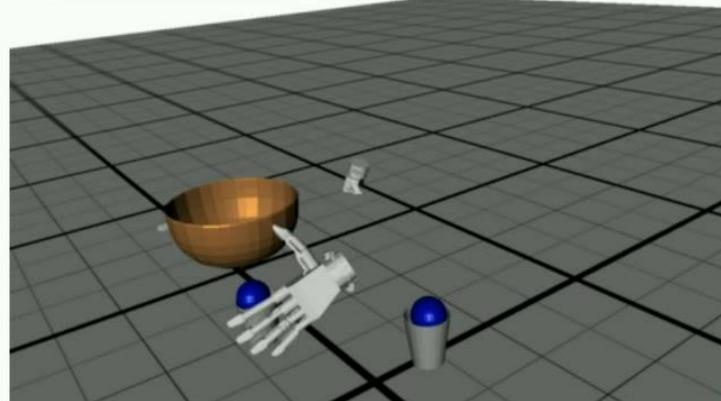
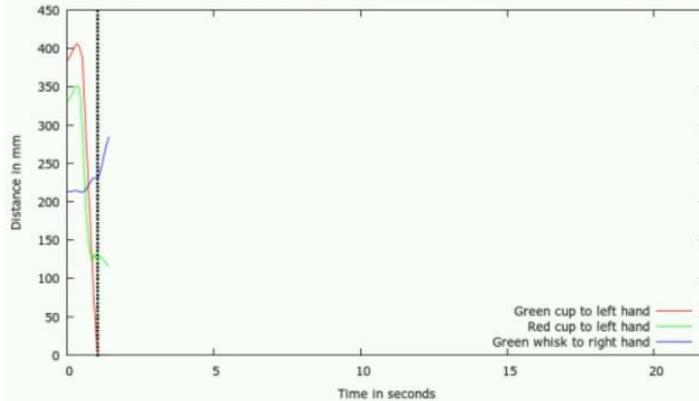
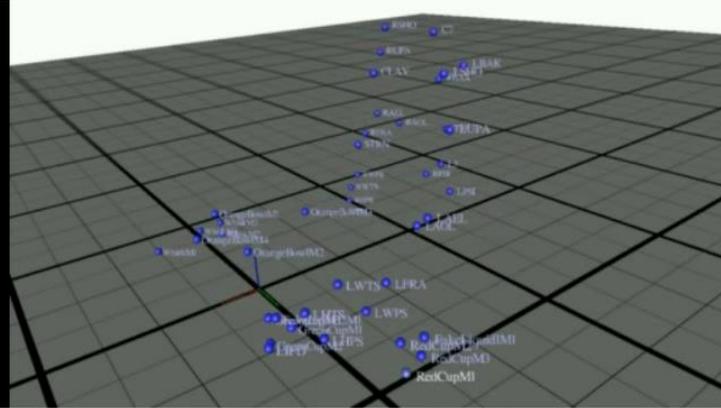
Average Results	HS	ZVC	PCA
Error	3.35 s ²	7.01 s ²	20.18 s ²
Accuracy	0.27 s	0.1 s	0.36 s
Unmatched key frames	2	0.6	27.9
Missed key frames	3.54	12.27	3.5

Evaluation: Comparison to Reference Segmentation

Two repetitions of action sequences of pouring-shaking actions



Hierarchical Action Segmentation



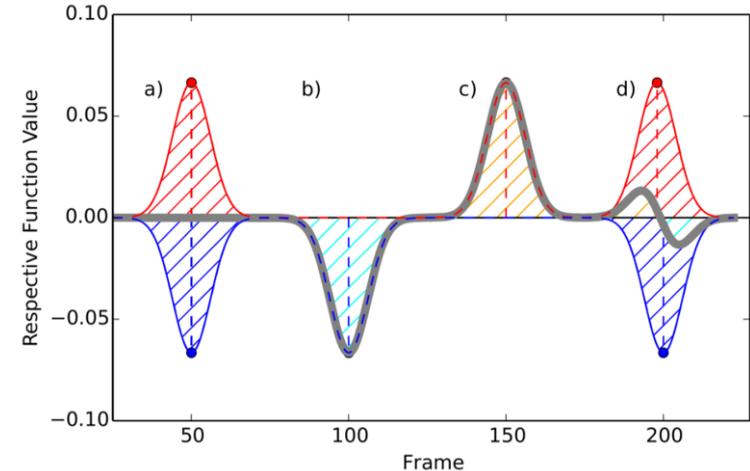
Evaluating Motion Segmentation Algorithms

- How to assess the quality of a segmentation?
- How to compare different segmentation algorithms?
- **An answer:** Integrated Kernel Approach (Dreher et al., 2017)

C. R. G. Dreher, N. Kulp, C. Mandery, M. Wächter and T. Asfour, A Framework for Evaluating Motion Segmentation Algorithms, IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2017

Evaluating Motion Segmentation Algorithms

- Open source framework to **evaluate motion segmentation algorithms**
- Novel way to assess the quality of the segmentation of human motion recordings, called the **Integrated Kernel** approach
- The **Motion Segmentation Point Hierarchy** to discriminate different granularities on which motion segmentation algorithms operate



a) Perfect segmentation
 b) / c) False negative / False positive
 d) Close segmentation

C. R. G. Dreher, N. Kulp, C. Mandery, M. Wächter and T. Asfour, A Framework for Evaluating Motion Segmentation Algorithms, IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2017

- For task understanding it is important to identify task constraints
 - Preconditions of actions
 - Order of actions
 - Temporal constraints
 - Spatial constraints (object-hand relations)
 - Execution related constraints (contact, forces, position, velocities, ...)

Task Constraints – Sequential Order of Actions

- Some actions, keypoints, sub-tasks, ... may depend on each other, some don't
- Consider the task “making muesli”
 - **Ingredients:** Milk, cereals, banana pieces
 - **Actions:** cut banana, pour banana pieces, milk, and cereals into a bowl
 - **Order of pouring the ingredients into the bowl is irrelevant, but the banana must be cut before pouring it into the bowl**

Making Muesli – First Demonstration



cut
banana



“pour”
banana

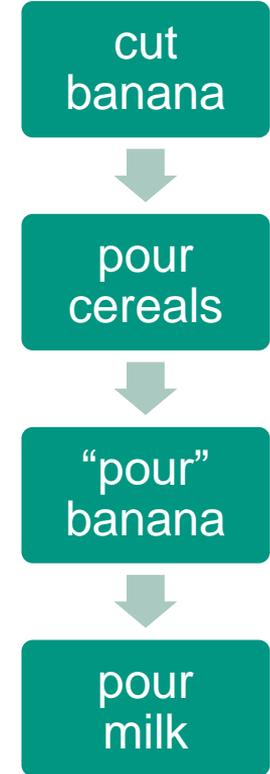


pour milk



pour
cereals

Making Muesli – Second Demonstration

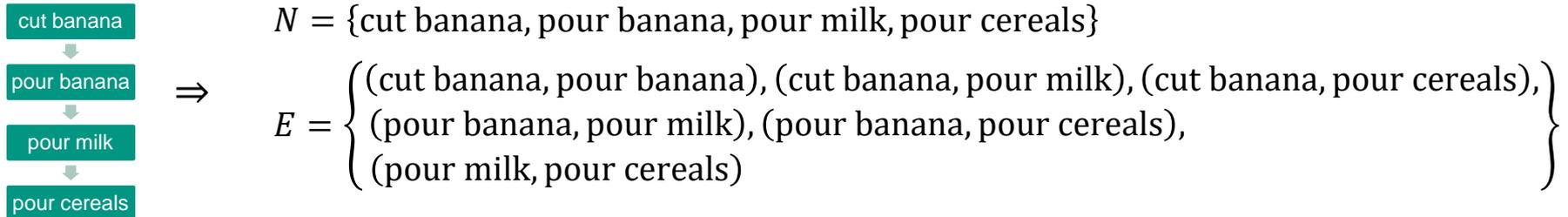


Task Precedence Graph – Definition

■ How to find out which sub-tasks/actions really depend on each other?

⇒ A **task precedence graph** is a graph $G = (N, E)$, where

- Nodes N is the set of sub-tasks/actions required to fulfill a task
- Edges E is the set of sequential dependencies of the sub-tasks/actions (Edge from X to $Y \rightarrow X$ must be finished before Y starts)



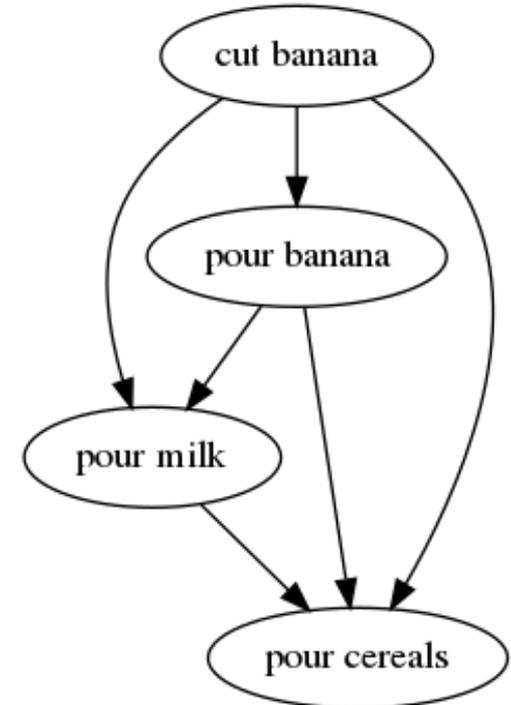
M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zöllner. “Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments”. Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37, no. 2 (2007): 322–32.

Task Precedence Graph – First Demonstration

- Without further knowledge, a robot has to assume that this is the only way to fulfill the task
- Task precedence graph is the transitive closure of the demonstration sequence:



⇒

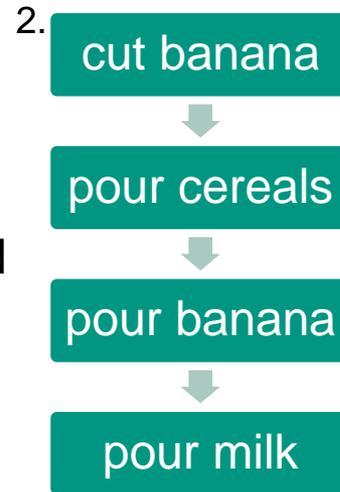


Task Precedence Graph – Second Demonstration

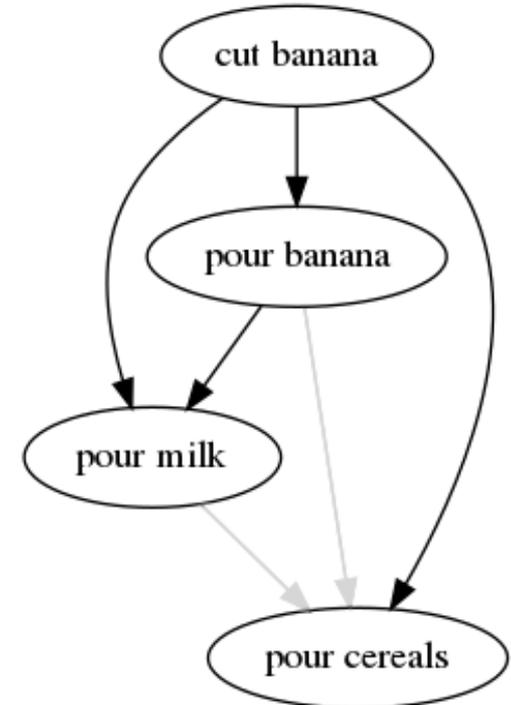
- The second demonstration is not consistent with the task precedence graph, as there are contradicting edges \Rightarrow remove them
- Result: More general task precedence graph



and

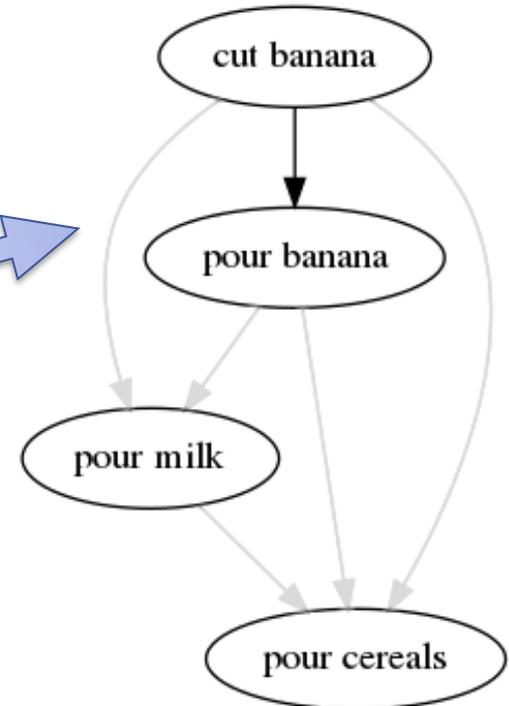
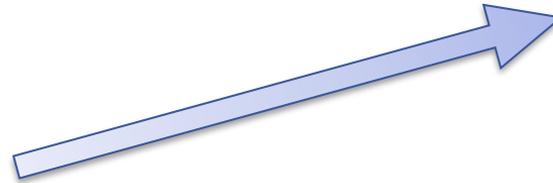


\Rightarrow



Task Precedence Graph – n -th Demonstration

- After enough demonstrations, more and more contradictions can be found if variations in sub-task order are observed
- Contradicting edges can then be removed from the task precedence graph
- Minimal task precedence graph



Task Precedence Graph – Discussion

Pros

- Can be incrementally learned as new demonstrations are observed
- Lightweight, easy to interpret

Cons

- Perfect data required
- Number of sub-tasks/actions fixed (cannot be changed)
- No possibility to determine whether the minimal task precedence graph has been found (without further knowledge)

■ **There are more types of constraints than only temporal constraints**

Other Types of Constraints

- Cartesian constraints
 - Position constraints
 - Orientation constraints
 - Direction constraints
 - Force constraints
 - Moment constraints
- Contact constraints
- Configuration constraints
- Additional constraints
 - Collision constraints
 - Grasp quality constraints
 - ...

Rainer Jäkel “Learning of Generalized Manipulation Strategies in Service Robotics”,
Ph.D. dissertation, Karlsruhe Institute of Technology, 2013.

Generalization

- Decompose the demonstration into pieces (segmentation)
- **Representation of these pieces in a parameterizable way**

Action Representations

■ Actions are **symbolic**

- grasp (object)
- pour (water)
- place (cup)
- wipe (table)
- cut (carrot)
- throw (ball)
- walk
- ...

■ Movement primitives specify the motion to execute an actions

Action Representation – Robotics

- Mathematical models to represent robot motions are needed
→ action representation
- Desired properties of such an action representation
 - **Reusability**: Reuse learned movement primitives (actions) to solve different tasks
 - **Flexibility**: Execution under different conditions, such as changing start and goal positions, velocity, acceleration
 - **Adaptivity**: Adaptation to dynamic situations, e.g. for obstacle avoidance
 - **Interoperability**: Use the same action representation on different robots

Movement Primitives - Biological Systems

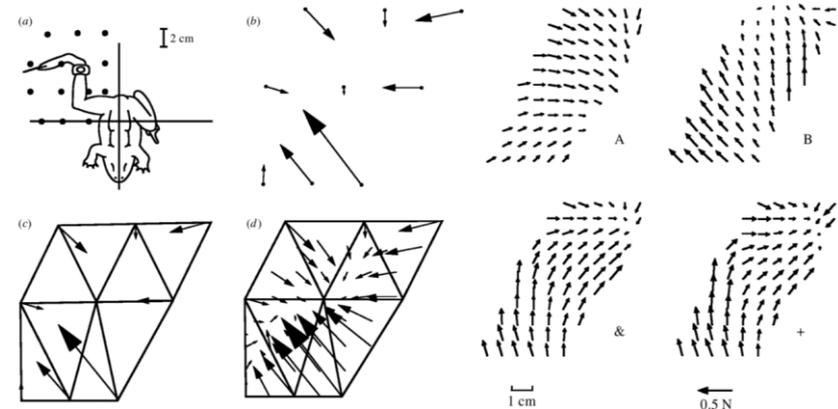
- Motor control in biological systems
 - Biological systems exhibit a continuous stream of movements in their daily activities
 - Movements can be
 - Rhythmic (e.g., locomotion, juggling)
 - Discrete (e.g., tennis swing)
- Assumption: Movement sequences consist of **segments** (→ motion primitives), which can be **executed** either in **sequence** or with **overlap**

Movement Primitives– Evidence from Neurobiology

■ Frog experiment

- stimulate spinal cord and record the force exerted by its leg at nine different locations (a-b)
- Interpolate the whole area with triangles (c-d)
- A, B: two different force fields generated by stimulating different sites
- +: superposition of A and B
- &: stimulate both sites

- Force fields in (+) and (&) are similar
- The movement is based on combinations of a few basic elements called ***motion primitives***



- Bizzi, E., Mussa-Ivaldi, F.A., Giszter, S. Computations underlying the execution of movement: a biological perspective“; Science , Vol. 253, Issue 5017, pp. 287-291, 19 Jul 1991, DOI: 10.1126/science.1857964
- Mussa-Ivaldi, F. A. and Bizzi, E. Motor learning through the combination of primitives. Philosophical transactions of the Royal Society of London. Series B, Biological sciences vol. 355,1404 (2000): 1755-69. doi:10.1098/rstb.2000.0733

Dynamic Movement Primitives (DMP)

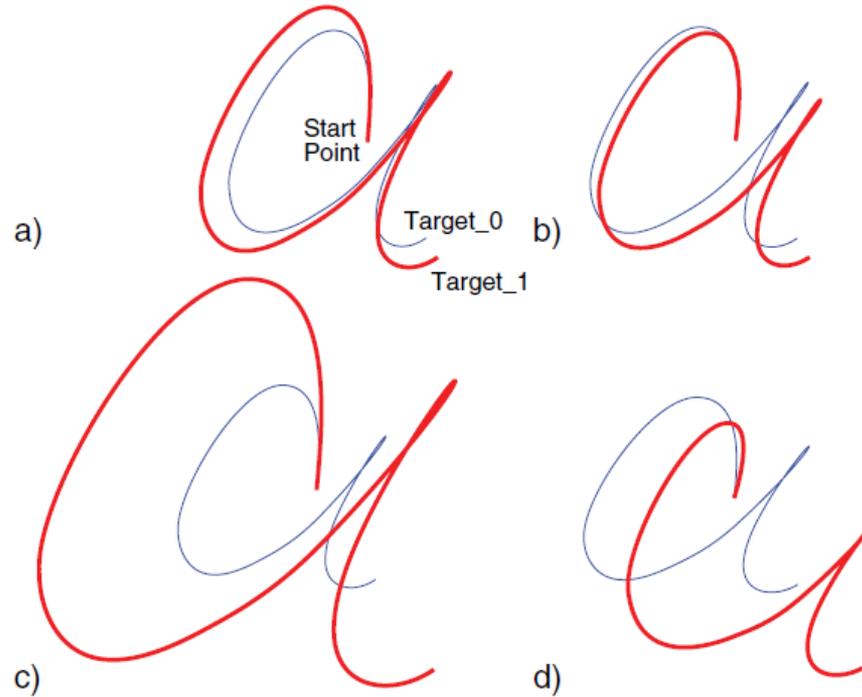
Red: relevant for the exam

- Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors; *Neural computation* 25 (2), 328-373, 2013
- A. J. Ijspeert, J. Nakanishi, and S. Schaal “Learning Attractor Landscapes for Learning Motor Primitives” *In Advances in Neural Information Processing Systems 15 (NIPS)*, 2003
- P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal “Learning and generalization of motor skills by learning from demonstration” *In Proc. IEEE Int. Conf. Robotics and Automation, Kobe, Japan, 2009*, pp. 763-769
- Ales Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto “Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives” *IEEE Transactions on Robotics*, 26(5): 800-815, 2010
- A. J. Ijspeert, J. Nakanishi, and S. Schaal “Learning Attractor Landscapes for Learning Motor Primitives” *In Advances in Neural Information Processing Systems 15 (NIPS)*, 2003

Dynamic Movement Primitives (DMP)

- Two types of motions
 - Discrete (e.g., tennis swing)
 - Periodic/Rhythmic (e.g., locomotion, juggling)
- Different stability requirements
 - **Discrete**: A discrete movement is **stable**, if the movement shows **point attraction behavior**, which means that it will stop somewhere when time goes infinite
 - **Periodic**: A periodic movement is **stable**, if the movement shows **path attraction behavior**, which means that it will keep the same path when time goes infinite

Dynamic Movement Primitives (DMPs): Motivation



Dynamic Movement Primitives

■ Given

- **Demonstration** by human (**Position** and **velocity** for each time step): y_D, v_D
- Arbitrary **start position** y_0 , **velocity** v_0
- Arbitrary **goal position** g
- Arbitrary **duration** (i.e., temporal factor τ)

■ Desired

- Trajectory from y_0 to g in duration τ with **shape characteristic similar to the demonstration** y_D

■ **Solution: Critically damped spring-mass system with perturbation**

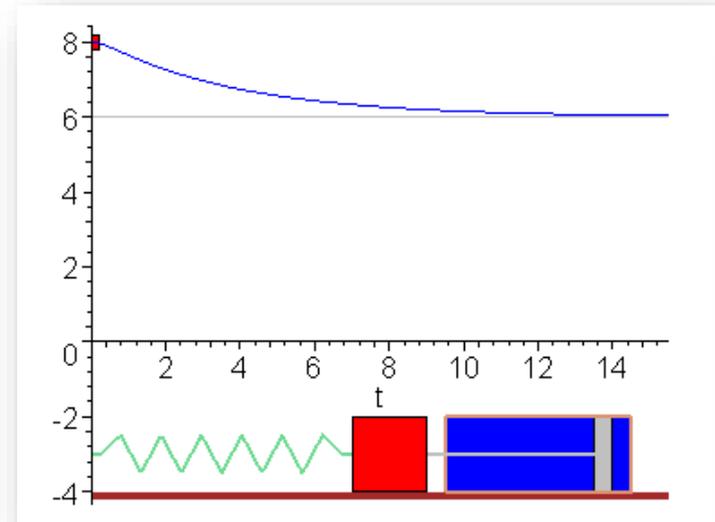
Damped Spring-Mass System

$$\ddot{x} = K(g - x) - D\dot{x}$$

Spring term: goal attractor
 Damping term: limits acceleration

- 2nd order ordinary differential equation (ODE)
- Globally stable attractor system which converges to g

g : goal
 x : current position
 D : damping constant
 K : spring constant



Damped Spring-Mass System

For easier computation, the ODE is transformed into a 1st order system:

Substitution of variables so that only the first derivative appears

$$\dot{v} = K(g - x) - Dv$$

$$\dot{x} = v$$

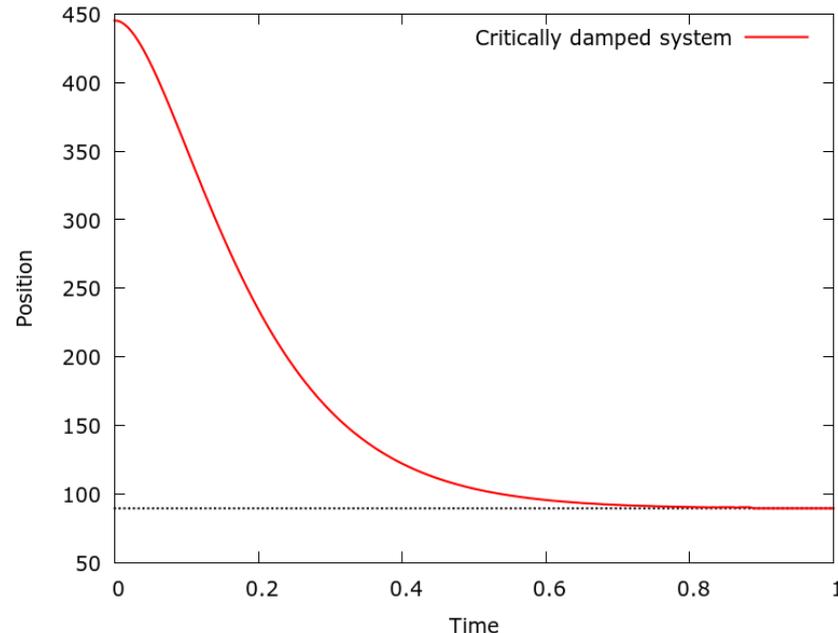
g : goal

x : current position

v : current velocity

D : damping constant

K : spring constant



Transformation System (I)

To shape the trajectory a **perturbation force term** is introduced:

Temporal factor

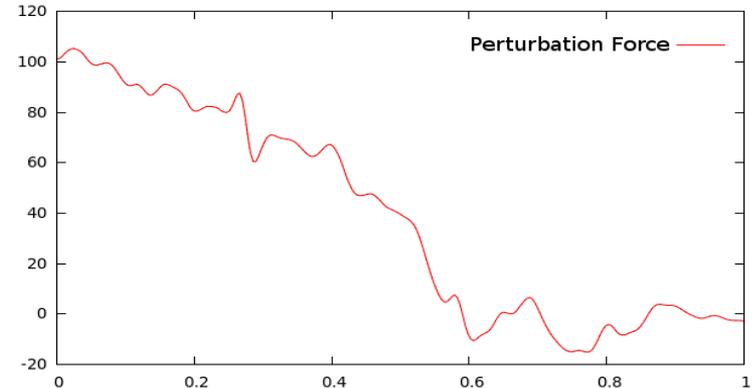
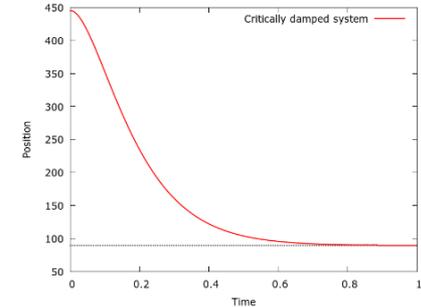
Scaled perturbation
Term (nonlinear)

$$\tau \dot{v} = K(g - x) - D \cdot v + (g - x_0) \cdot f(u)$$

$$\tau \dot{x} = v$$

Perturbation force

- This non-linear dynamic system is called the **transformation system** of a DMP
- The **perturbation force term** is learned from **demonstration**



Transformation System (II)

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0) \cdot f(u)$$

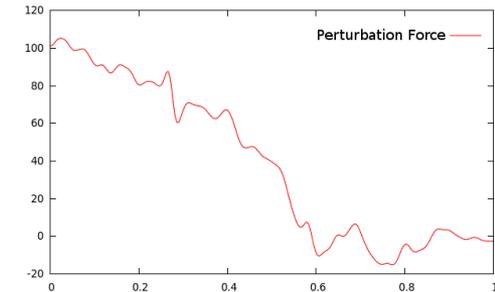
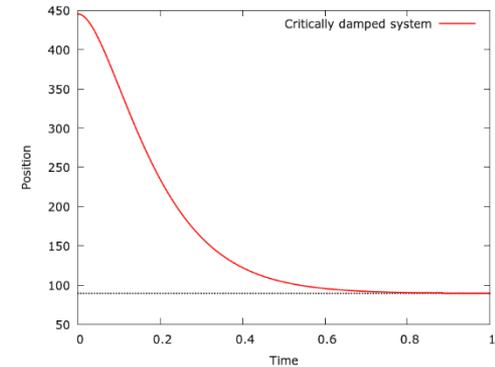
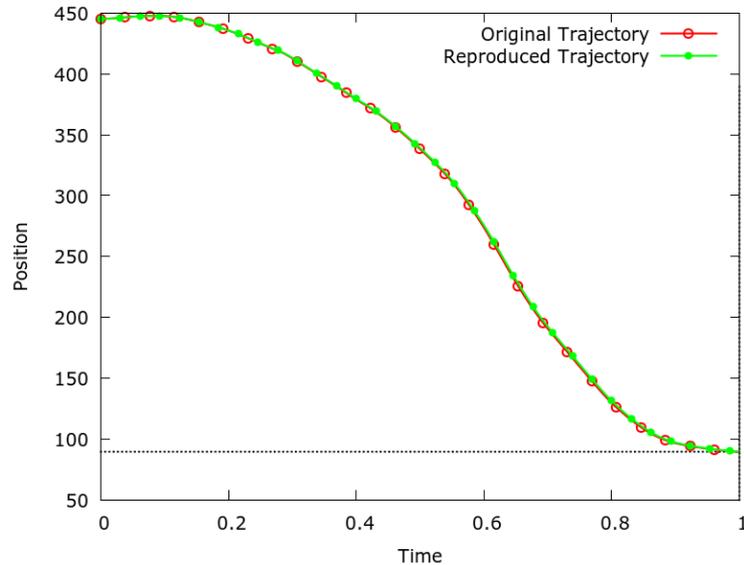
$$\tau \dot{x} = v$$

- Why 2nd order system?
 - position, velocity and acceleration needed for control

- $f(u)$
 - $f(u) = 0 \rightarrow$ globally stable 2nd order linear dynamic system with a unique attractor point g (converge to g). This is a trivial pattern generator to a single attractor point
 - $f(u)$ **phasic** (active in a finite time window) \rightarrow point attractor (discrete movement)
 - $f(u)$ **periodic** \rightarrow Oscillator (rhythmic movement)

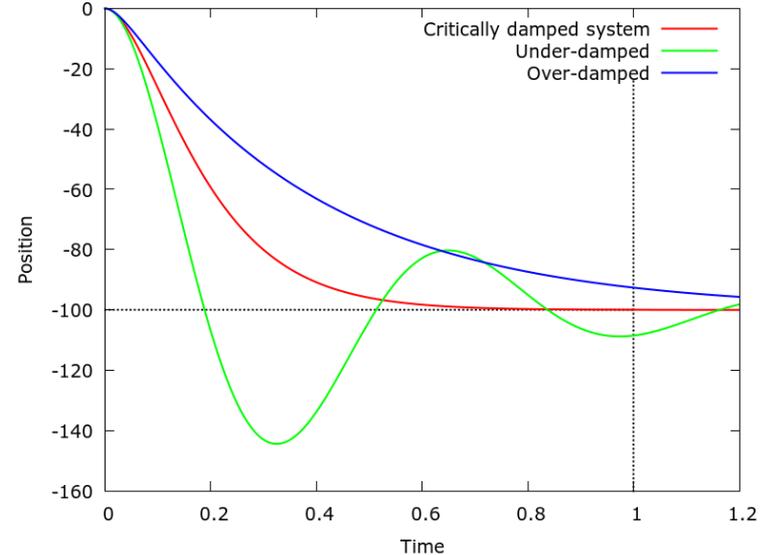
Transformation System (III)

Resulting trajectory of transformation system



Value Selection of Constants D and K

- A damped spring-mass system oscillates around the goal: not desired for movement generation
- Solution: Critically damped spring-mass system
 - Special relation of spring constant K and damping constant D
 - Damping is just as strong to compensate overshooting over goal
 - But: goal is still reached as fast as possible



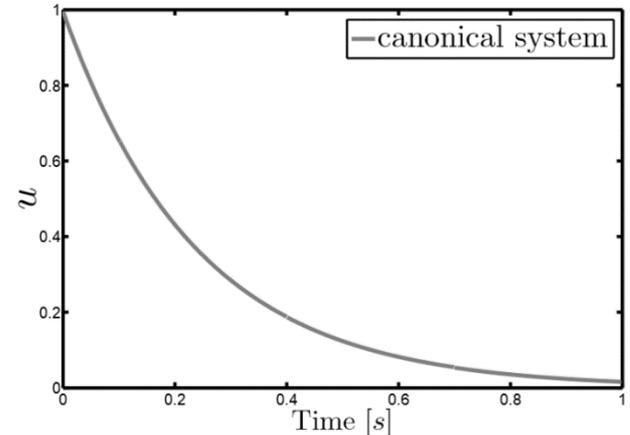
The Canonical System: Inducing Time Independence

- If the **perturbation force term** f depends explicitly on time, the complete execution depends on time; In order to change the speed of the motion, we need to change the trajectory.
- Solution: Use a **canonical system (phase variable)** instead of explicit time dependence (relates time and the state u)

Temporal factor

$$\tau \dot{u} = -\alpha u$$
$$\alpha = -\log(u_{min}) \tau$$
$$u_{min} = 0.001$$
$$u(0) = 1$$

- With the canonical system,
 - Changing $\tau \Rightarrow$ change the motion speed



DMP: Recap

- The **canonical system**
 - State of the DMP in time
 - Drives the perturbation to control the transformation system
- The **transformation system** and the Perturbation
 - Are adapted to the demo trajectory
 - Generate the robot motion
 - Can generalize the learned trajectory to new conditions

Canonical system

$$\tau \dot{u} = -\alpha u$$

Transformation system

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(u)$$

$$\tau \dot{x} = v$$

Perturbation: Shaping the Trajectory

- The **perturbation function** f determines the **shape** of the trajectory
 - It “pushes” the spring-mass-system away from it’s original path
 - Additional term in the acceleration calculation
- **Non-linear function** representing the **demonstrated trajectory**
- f depends on the canonical system $u(t)$
- Resolve the transformation system to f

$$f = \frac{-K(g - x) + Dv + \tau\dot{v}}{(g - x_0)}$$

- How do we find f and how do we store it?

How to Learn the Trajectory of the Demonstration

- Given: **Demonstration** y (sequence of position, velocities and acceleration with timestamps) of length T
- Wanted: **Perturbation force** for complete demonstration
- Calculate f for all timestamps with

$$\begin{aligned}g &= y(T) \\x_0 &= y(0) \\x, v, \dot{v} &= y, \dot{y}, \ddot{y} \\ \tau &= T\end{aligned}$$

- These are only the perturbation forces at the given timestamps
 - Continuous representation of the perturbation force desired

How to Approximate the Perturbation Force

- **Approximate** the discrete values for perturbation function f with a **continuous representation**
- Approximation can be done with any **function approximation algorithm**
 - Weighted Radial Basis Functions (RBF)
 - Splines
 - LWPR
 - ...
- **Here: RBF**

Radial Basis Function (RBF) Networks (I)

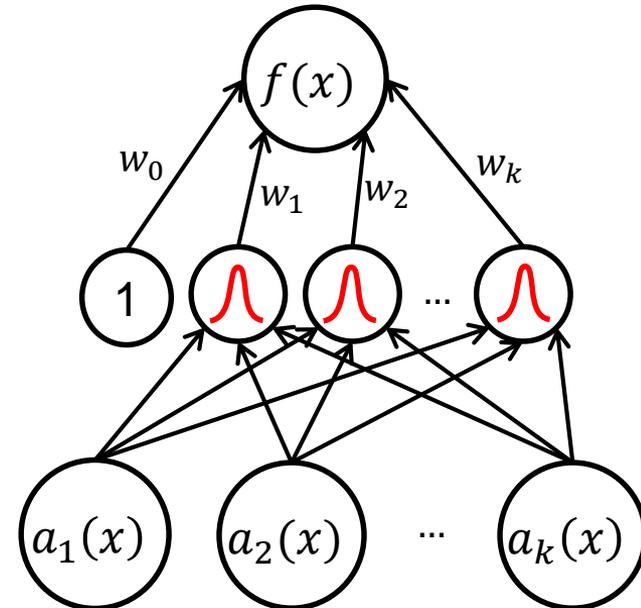
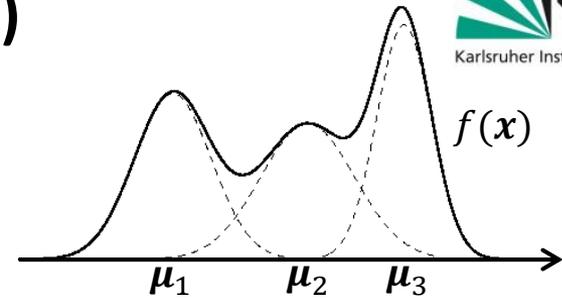
- Function approximation

$$\hat{f}(x) = w_0 + \sum_{u=1}^k w_u \cdot K_u(x)$$

- Kernel function

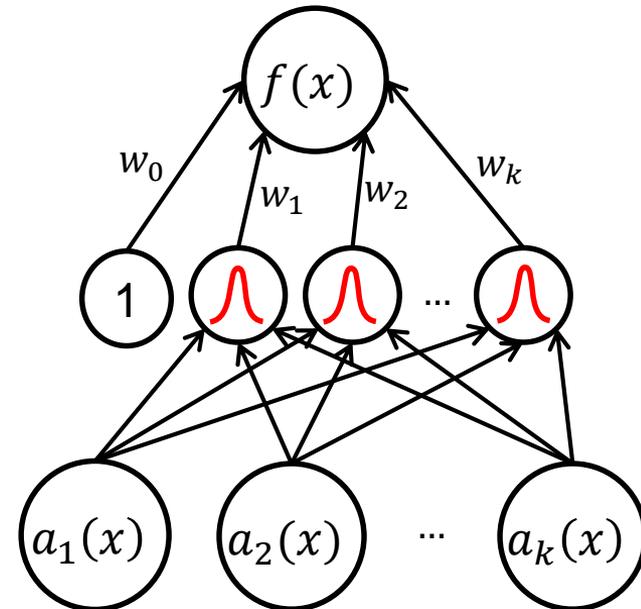
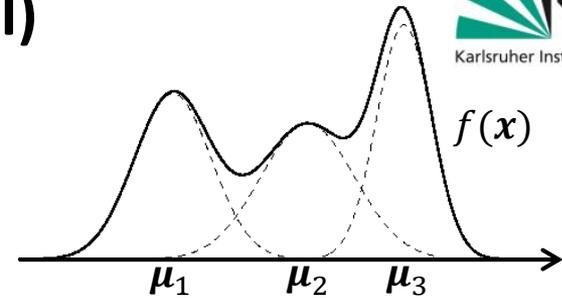
$$K_u(x) = e^{-\frac{1}{2\sigma_u^2} \|\mu_u - x\|^2}$$

- Modelled as a neural network
 - One hidden layer
 - One output neuron
- Related to locally weighted regression



Radial Basis Function (RBF) Networks (II)

- Training a RBF network
- Two steps
 - **1. Determine the receptive fields**
 - Number k of hidden neurons
 - Parameters μ_u and σ_u
 - One neuron per sample or
 - Uniformly distributed in X or
 - Based on clustering of the training data or
 - Expectation Maximization (EM)
(finds optimal μ_u and σ_u)
 - ...
 - **2. Determine the weights w_u**
 - As in regular neural networks
 - While μ_u and σ_u are fixed



How to Approximate the Perturbation Force (I)

■ Here: **Weighted RBFs**:

- Locally weighted regression

$$f_{approx.}(u) = \frac{\sum_{i=1}^N w_i \psi_i(u)}{\sum_{i=1}^N \psi_i(u)} \cdot u$$

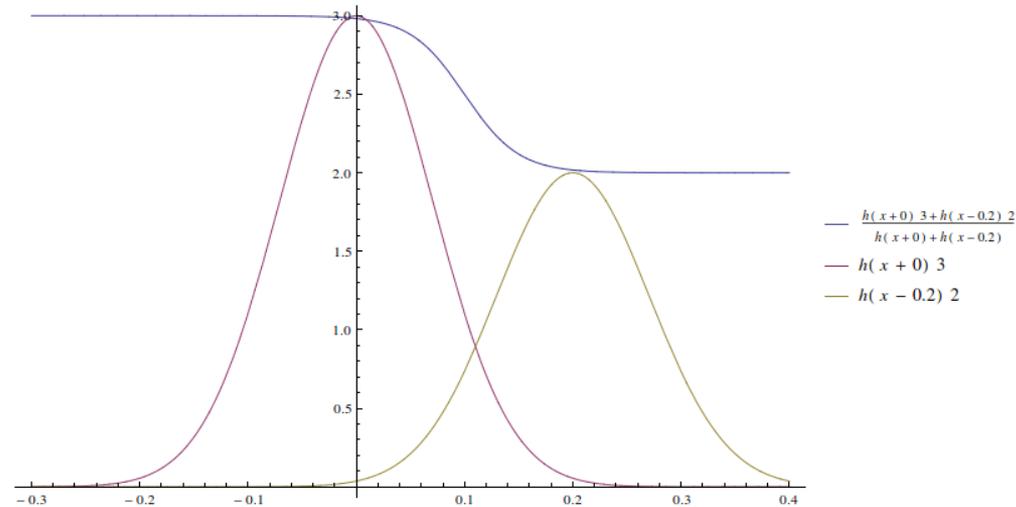
- Basis function - RBF:

$$\psi_i(u) = e^{-h_i (u - c_i)^2} \left\{ \begin{array}{l} h_i: \text{width of RBF} \\ c_i: \text{center of RBF} \\ w_i: \text{weight of RBF} \end{array} \right.$$

- One dimensional optimization problem to find the weights w_i with fixed h_i and c_i that fit the discrete values of f best
- Weights w_i are stored for the execution of the DMP, h_i and c_i have fixed values

Example

2 weighted RBFs, proximity to support points is determined by RBF width

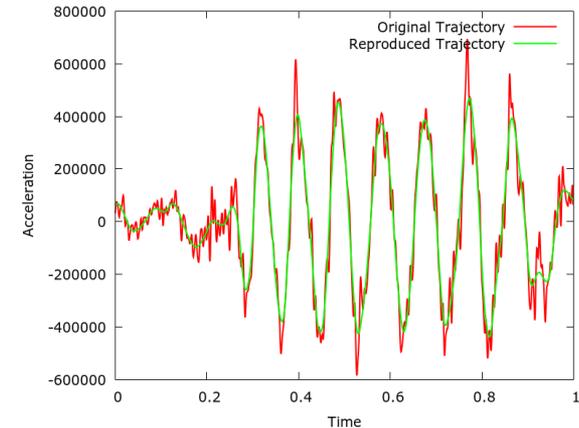
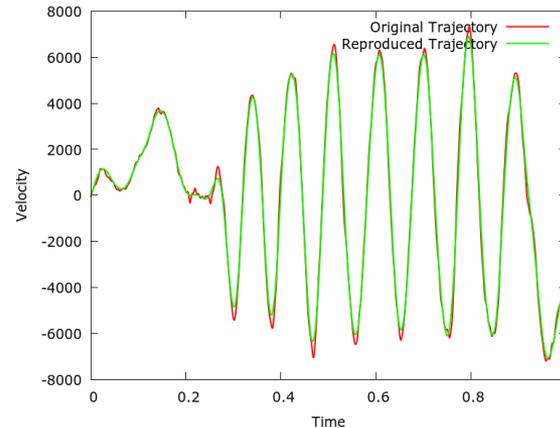
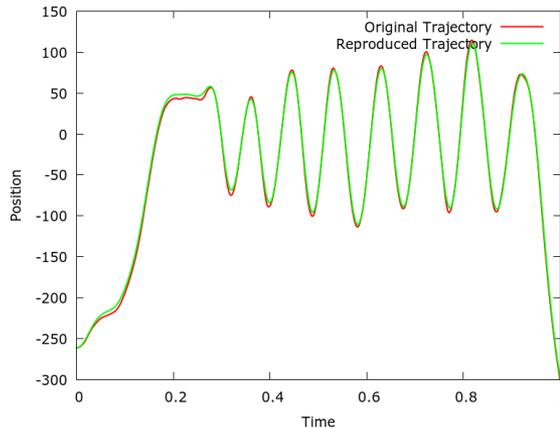


Approximation accuracy depends on the number of basis functions, the more basis functions, the less approximation error

Benefits of Approximation

■ Weighted RBFs

- remove jitter from original demonstration
- reduce memory consumption: 70 RBFs instead of 1500 position and velocity values (in the example below)



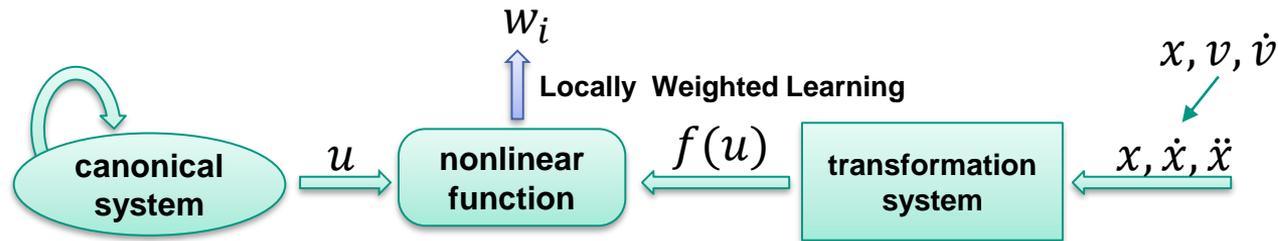
DMP Formalization (I)

canonical system: $\tau \dot{u} = -\alpha u$

nonlinear function: $f(u) = \frac{\sum_i w_i \psi_i(u)}{\sum_i \psi_i(u)} \cdot u$ $\psi_i(u) = e^{-h_i(u-c_i)^2}$

transformation system: $\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(u)$
 $\tau \dot{x} = v$

$$f_{target} = \frac{-K(g - x) + Dv + \tau \dot{v}}{(g - x_0)}$$



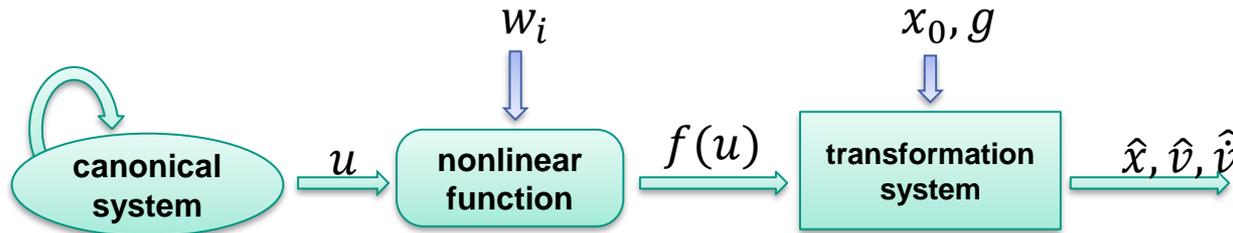
DMP Formalization (II)

canonical system: $\tau \dot{u} = -\alpha u$

nonlinear function: $f(u) = \frac{\sum_i w_i \psi_i(u)}{\sum_i \psi_i(u)} \cdot u$ $\psi_i(u) = e^{-h_i(u-c_i)^2}$

transformation system: $\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(u)$
 $\tau \dot{x} = v$

$$f_{target} = \frac{-K(g - x) + Dv + \tau \dot{v}}{(g - x_0)}$$

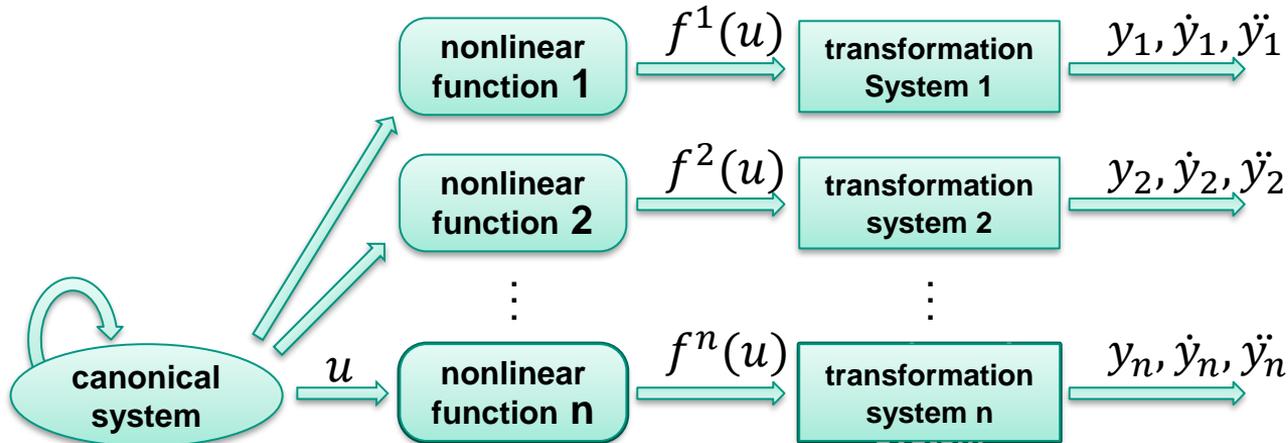


n-dimensional DMPs (I)

canonical system: $\tau \dot{u} = -\alpha u$

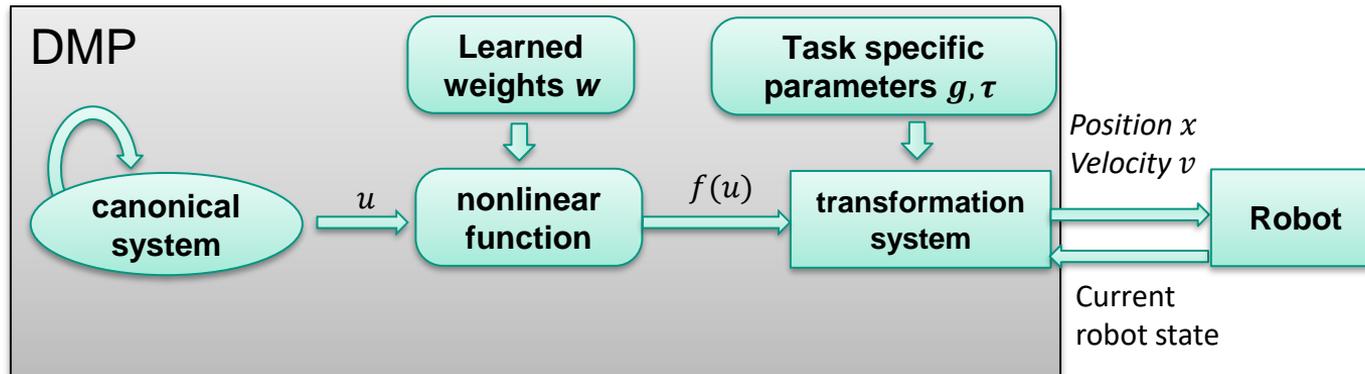
nonlinear function: $f(u) = \frac{\sum_i w_i \psi_i(u)}{\sum_i \psi_i(u)} \cdot u \quad \psi_i(u) = e^{-h_i(u-c_i)^2}$

transformation system: $\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(u)$
 $\tau \dot{x} = v$

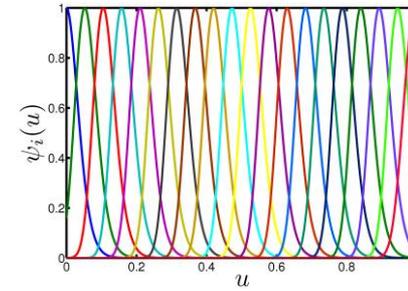
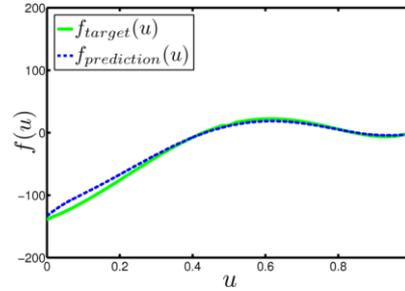
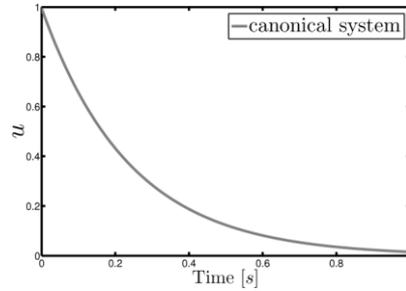
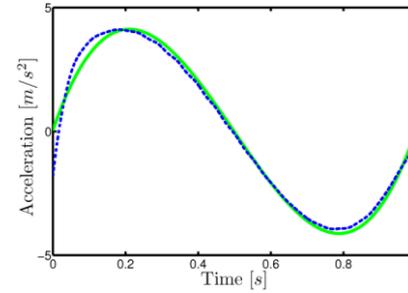
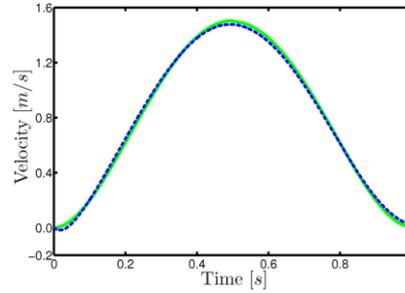
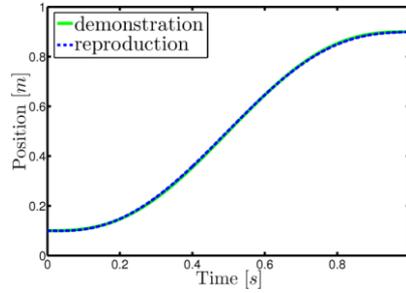


Execution of a DMP

- Parameterize the DMP
 - Initial state (position & velocity): y_0, \dot{y}_0 (x_0, v_0)
 - Goal position: g
 - Temporal factor: τ
 - Choose a demonstration (stored as approximation of perturbation force)
- Integrate the ordinary differential equation

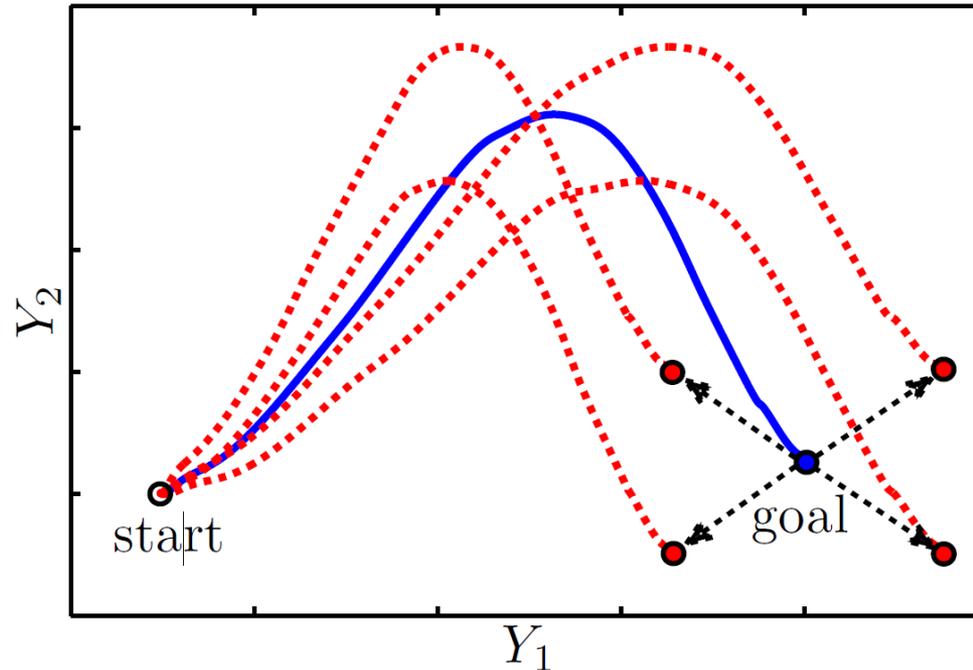


Example



Changing the Goal

- DMPs can generalize to new goal while keeping the characteristic shape of the trajectory



Examples



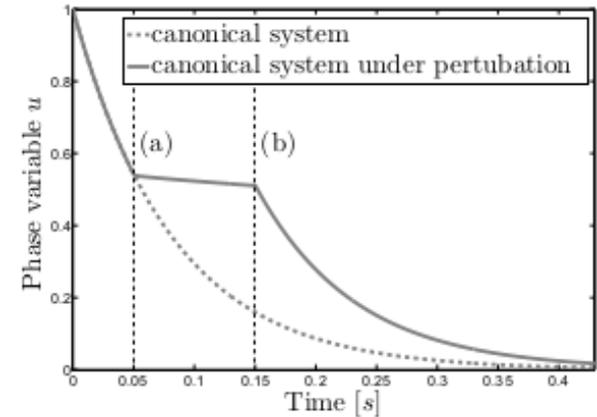
Online Feedback: External Perturbations (I)

- One way to deal with an external perturbation is **phase stopping**
 - Phase slows down until external perturbation force is gone
 - E.g. a person holds the arm of the robot
- Modifying the canonical system for phase stopping:

$$\tau \dot{u} = \frac{-\alpha u}{1 + \alpha_{pu} |\tilde{x} - x|}$$

x : desired position
 \tilde{x} : actual position

- α_{pu} controls how fast the phase slows down
- Increasing error in position slows the phase down



Online Feedback: External Perturbations (II)

- One way to deal with an external perturbation is **phase stopping**
 - Phase slows down until external perturbation force is gone
 - E.g. a person holds the arm of the robot

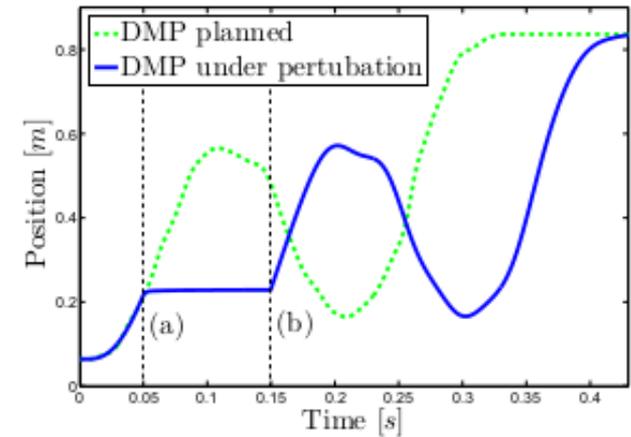
- Similarly the transformation system is changed:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0) f(u)$$

$$\tau \dot{x} = v + \alpha_{px} |\tilde{x} - x|$$

x : desired position

\tilde{x} : actual position



An error in position moves the desired position towards the actual position

Online Feedback: Changing Goal

- Simple online changing of the goal would lead to a jump in the acceleration

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(u)$$

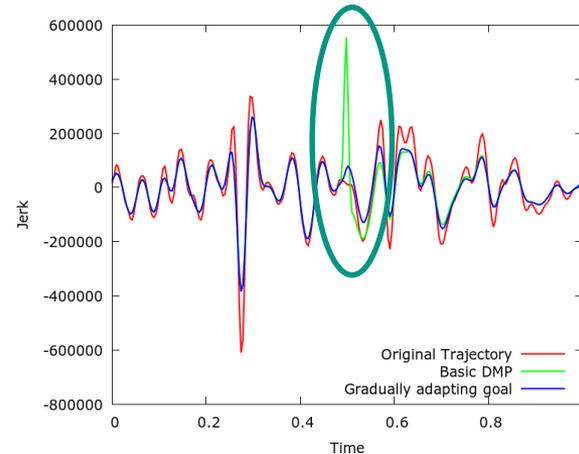
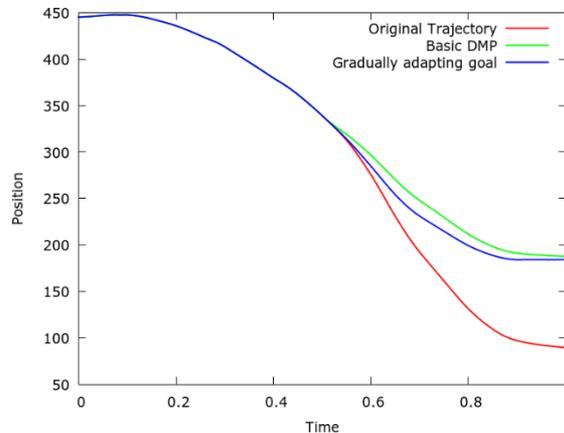
$$\tau \dot{x} = v$$

- Filtering the goal with the ODE:

$$\dot{g} = \lambda(g - g_d)$$

g_d : Desired goal

λ : Goal change rate factor



Summary

- DMPs provide a simple mathematical formulation of movement primitives
- DMPs can adapt to new start and goal
- DMPs can adapt to external perturbation

- **However, only one single demonstration is used!**

- How to learn from multiple demonstrations and take variability into account?

Actions as Trajectories distribution

- Probabilistic Movement Primitives (ProMPs)
 - Represent trajectories distribution with a Gaussian distribution
- Via-points Movement Primitives (VMPs)
 - Represent trajectories distribution with a structured formulation and a Gaussian distribution

Probabilistic Movement Primitives (ProMP)

- Represent actions using probabilistic linear regression model

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \Psi_t \mathbf{w} + \epsilon_x$$

$\epsilon_x \sim \mathcal{N}(0, \Sigma_x)$ is Gaussian noise

$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ follows a Gaussian distribution

- N Basis functions:

$$\mathbf{w} = [w_1, w_2, \dots, w_N]^T$$

$$\Psi_t = \begin{bmatrix} \boldsymbol{\psi}_t \\ \dot{\boldsymbol{\psi}}_t \end{bmatrix}$$

$$\boldsymbol{\psi}_t = [\psi_1(t) \quad \psi_2(t) \quad \dots \quad \psi_N(t)]$$

- Recall the basis function for the perturbation force in DMP

$$\psi_i(t) = e^{-h_i(t-c_i)^2}$$

Paraschos, A., Daniel, Ch. Peters, J. and Neumann, G.
 Probabilistic Movement Primitives. Probabilistic Movement Primitives. NIPS, 2013

Probabilistic Movement Primitives (ProMP)

- The parameters of a ProMP are

$$\theta = \{\mu_w, \Sigma_w\}$$

- How to learn the parameters

Probabilistic Movement Primitives (ProMP)

$$\theta = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$$

- How to learn the parameters (here, we only consider the position):

- Collect M demonstrations $\{\xi_i\}_{i=1}^M$, each trajectory can be represented by T samples such as: (t_k is the k -th time point)

$$\xi_i = [x_{t_1}, x_{t_2}, \dots, x_{t_T}]^T = [\boldsymbol{\psi}_{t_1} \mathbf{w}_i, \boldsymbol{\psi}_{t_2} \mathbf{w}_i, \dots, \boldsymbol{\psi}_{t_T} \mathbf{w}_i]^T = \boldsymbol{\Psi} \mathbf{w}_i$$

- For each trajectory, use pseudo-inverse:

$$\mathbf{w}_i = (\boldsymbol{\Psi}^T \boldsymbol{\Psi})^{-1} \boldsymbol{\Psi}^T \cdot \xi_i$$
$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_1(t_1) & \psi_2(t_1) & \dots & \psi_N(t_1) \\ \psi_1(t_2) & \psi_2(t_2) & \dots & \psi_N(t_2) \\ \dots & \dots & \dots & \dots \\ \psi_1(t_T) & \psi_2(t_T) & \dots & \psi_N(t_T) \end{bmatrix}$$

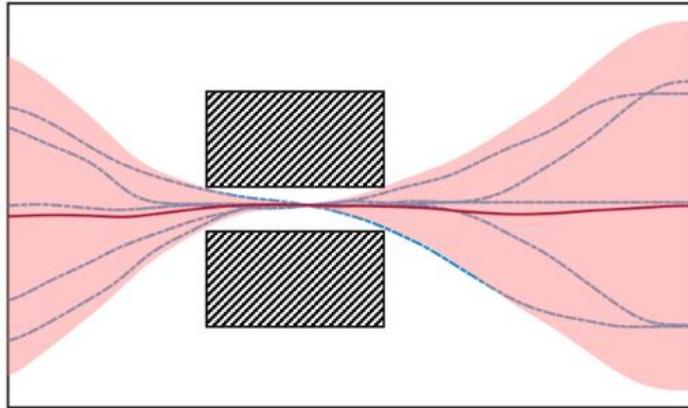
- Calculate the empirical mean and variance

$$\hat{\boldsymbol{\mu}}_w = \frac{1}{M} \sum_{i=1}^M \mathbf{w}_i$$
$$\hat{\boldsymbol{\Sigma}}_w = \frac{1}{M} \sum_{i=1}^M (\mathbf{w}_i - \boldsymbol{\mu}_w)(\mathbf{w}_i - \boldsymbol{\mu}_w)^T$$

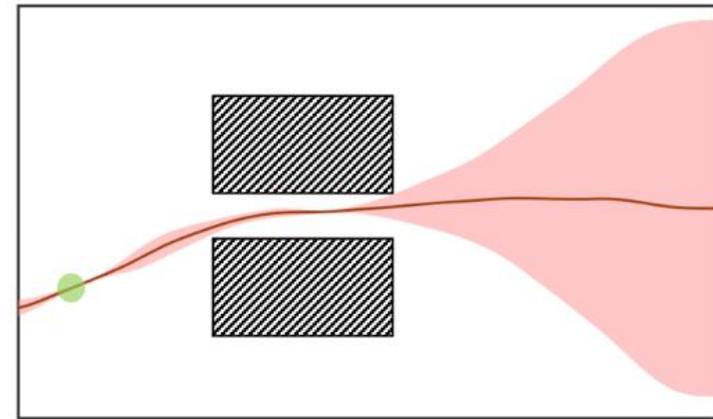
Advantage of ProMPs

■ Via-points adaptation

- Via-points are points that the generated trajectories should go through.



blue curves: demonstrations
 red region: ProMP $\xi = \Psi \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{\mathbf{w}}, \hat{\boldsymbol{\Sigma}}_{\mathbf{w}})$
 red curve: $\xi = \Psi \hat{\boldsymbol{\mu}}_{\mathbf{w}}$
 shadowed rectangles: obstacles



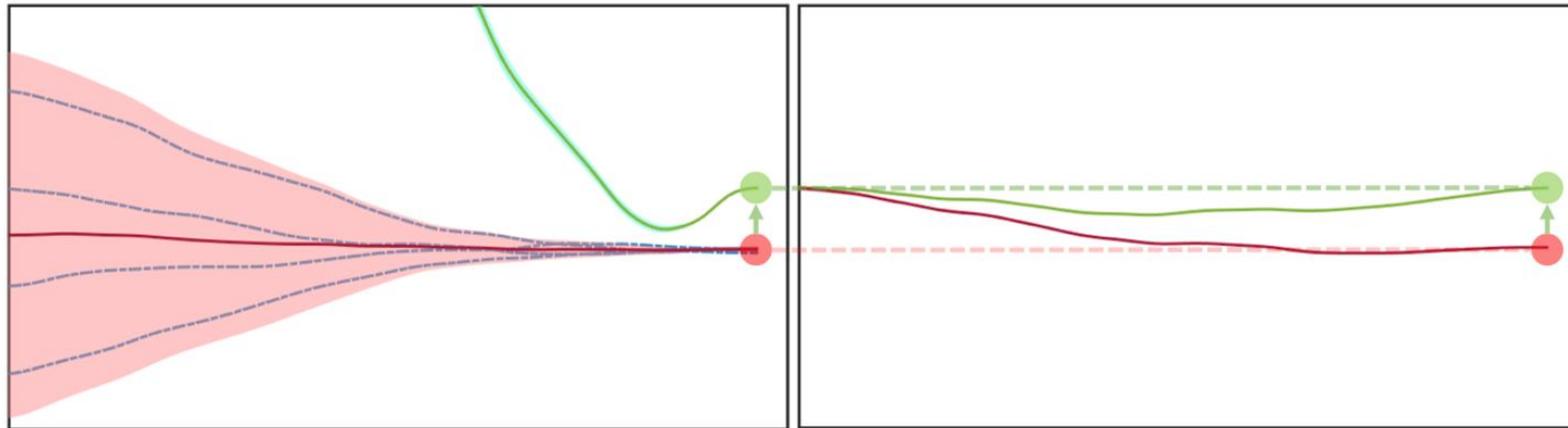
green dot: via-point
 red region: ProMP $\xi^* = \Psi \mathbf{w}^*$ with $\mathbf{w}^* \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{\mathbf{w}|x_{via}}, \hat{\boldsymbol{\Sigma}}_{\mathbf{w}|x_{via}})$
 red curve: $\xi^* = \Psi \hat{\boldsymbol{\mu}}_{\mathbf{w}|x_{via}}$

Comparison between DMP and ProMP

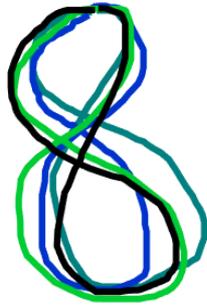
ProMP encodes multiple demonstrations \Leftrightarrow DMP can only learn one demonstration

ProMP can adapt to intermediate via-points \Leftrightarrow DMP cannot adapt to intermediate via-points

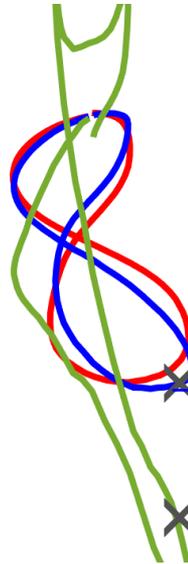
ProMP generates infeasible motions for the goals \Leftrightarrow DMP can adapt to the goal far away from the one in the demonstration.



Via-points Adaptation Problem of ProMPs



demonstrations

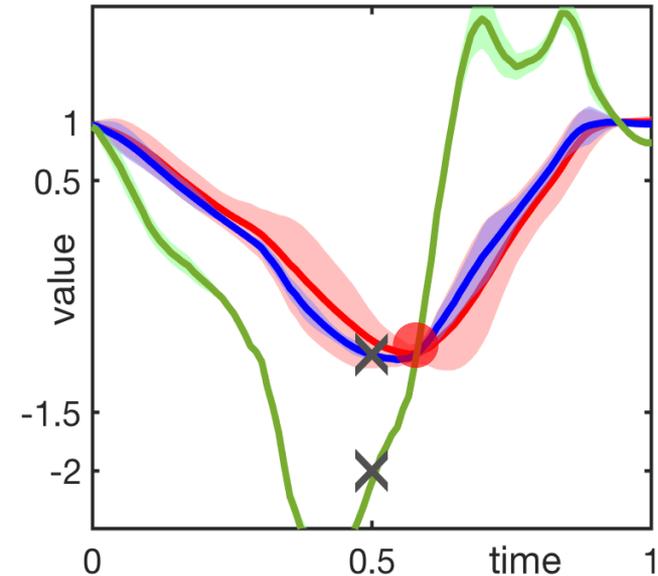
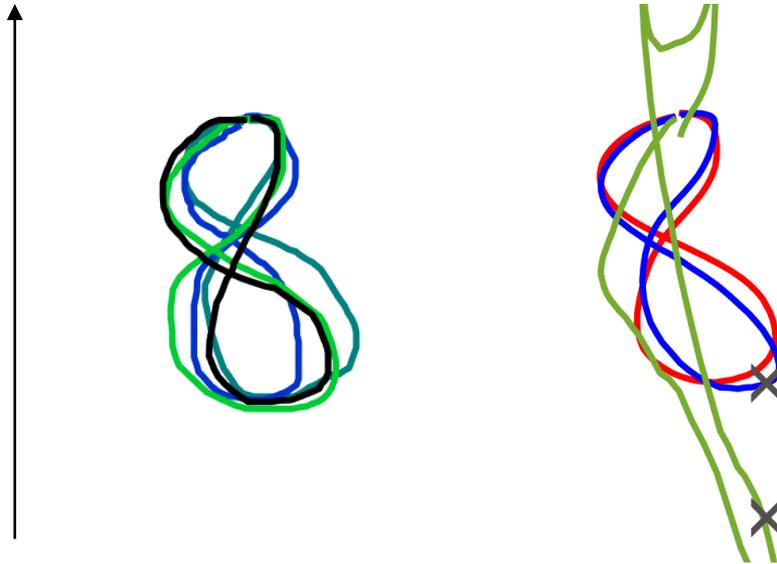


via-point near to the demonstrations ✓

via-point far away from demonstrations ✗

Via-points Adaptation Problem of ProMPs

- Check the motion in the vertical direction

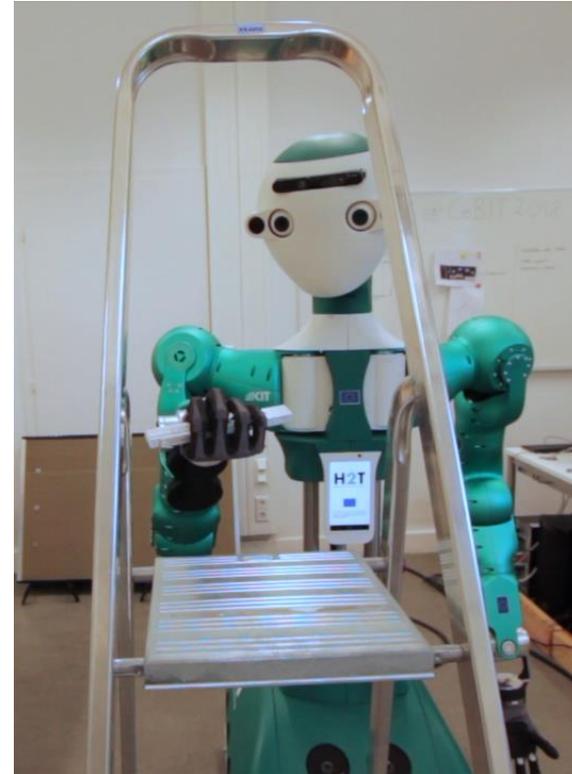


- The learned low variance region causes the problem.

Via-points Adaptation

- Why we need via-points adaptation?

The robot learns the motion to pass a short ruler through the ladder

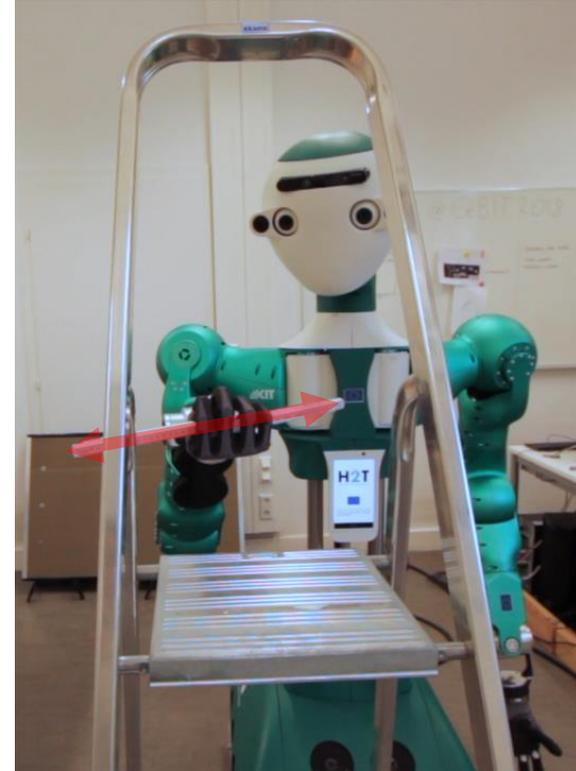


Via-points Adaptation

■ Why we need via-points adaptation?

For a longer ruler, it makes more sense to integrate a via-point instead of learning a new motion

Via-point should encode both position and orientation.



Via-points Adaptation

- Why we need via-points adaptation?

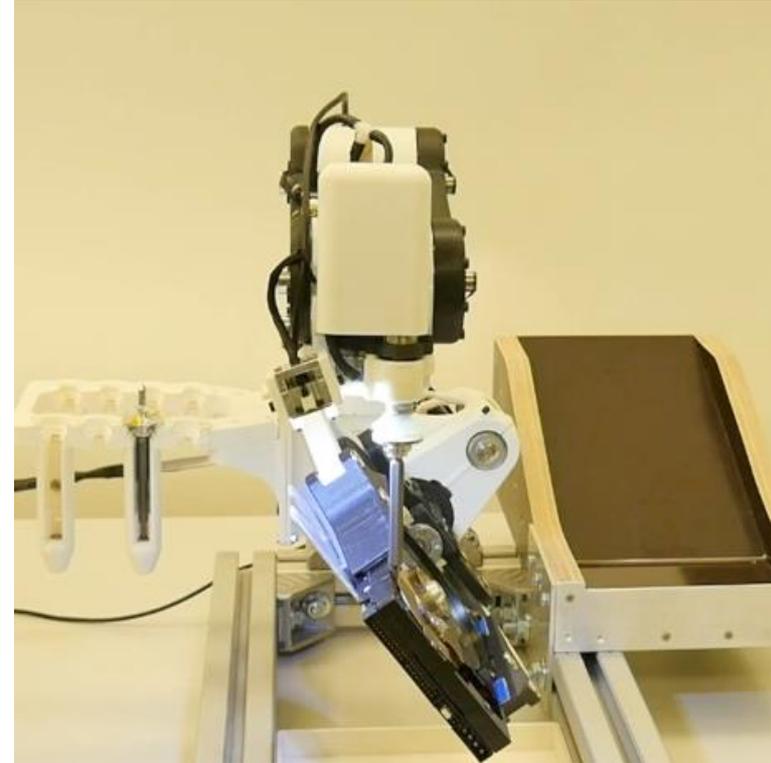
The robot needs to guarantee the position accuracy at some place



Via-points Adaptation

- Why we need via-points adaptation?

A via-point guarantees the accuracy



Via-point Movement Primitives (VMP)

$$x(u) = h(u) + f(u)$$



u is the phase variable and determined by a canonical system as DMP

elementary trajectory: a spatial trajectory that connects the successive via-points directly, e.g.

$$h(u) = a_1 u + a_0$$

$$h(u) = \sum_{i=0}^5 a_i u^i$$

shape modulation: a probability distribution that models the offset of $x(u)$ to $h(u)$

$$f(u) = \psi_u w \quad \psi_u = [\psi_1(u), \dots, \psi_N(u)]$$

$$w \sim \mathcal{N}(\mu_w, \Sigma_w) \quad w = [w_1, \dots, w_N]^T$$

Similar to ProMP

Y. Zhou, J. Gao and T. Asfour, "Learning Via-Point Movement Primitives with Inter- and Extrapolation Capabilities," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4301-4308

Via-point Movement Primitives (VMP)

- Learning VMP is similar to learning ProMP, however, we need to subtract the elementary trajectory from each demonstration. Each weights vector takes the form:

$$\mathbf{w}_i = (\Psi^T \Psi)^{-1} \Psi^T \cdot (\xi_i - \mathbf{h}_i)$$

Via-points Adaptation with $f(x)$

- Similar to ProMP, we can calculate the conditional Gaussian distribution for the shape modulation based on the via-point (u_{via}, x_{via}) as

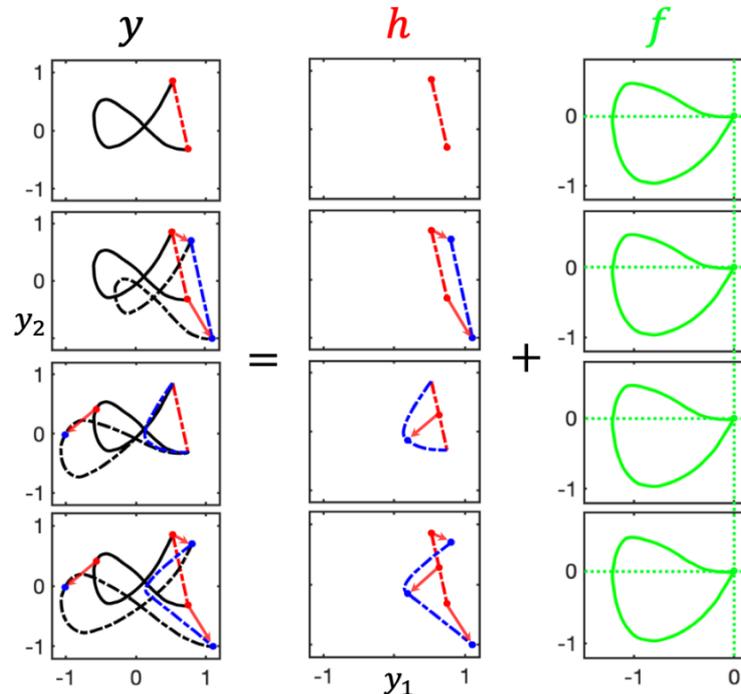
$$x^*(u) = h(u) + f^*(u)$$

$$f^*(u) = \psi_u w^* \quad w^* \sim \mathcal{N}(\mu_w | u_{via}, x_{via}, \Sigma_w | u_{via}, x_{via})$$

u_{via} is the phase variable that indicates when the generated trajectory should go through the via-point

Via-points Adaptation with $h(x)$

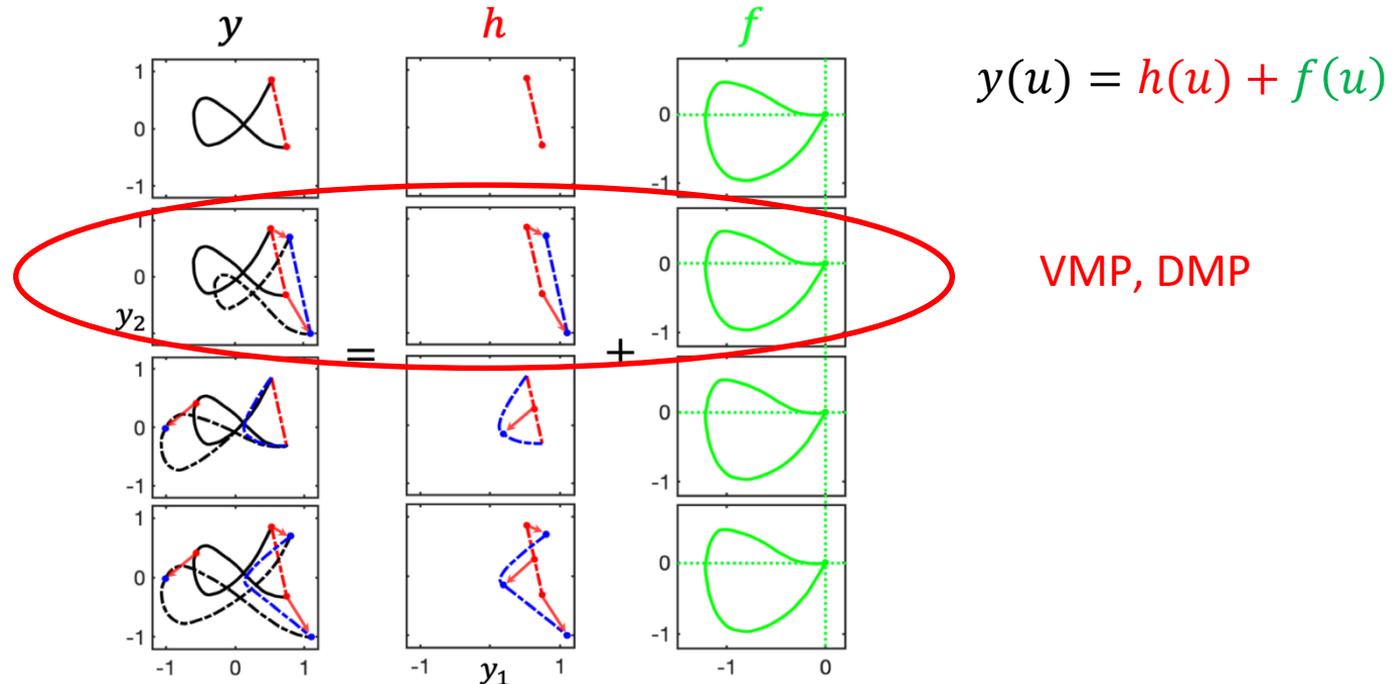
- How can VMPs adapt to via-points using the elementary trajectory $h(u)$?



$$y(u) = h(u) + f(u)$$

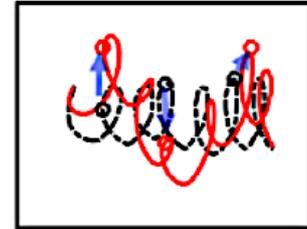
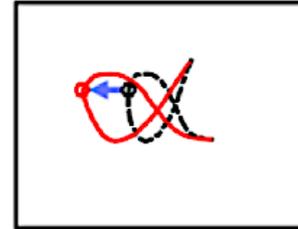
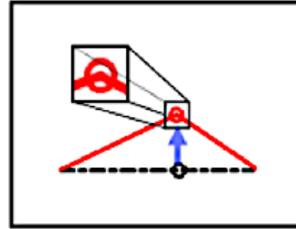
Via-points Adaptation with $h(x)$

- If we use $h(u) = a_1u + a_0$ this can be achieved similar to the DMP goal adaptation

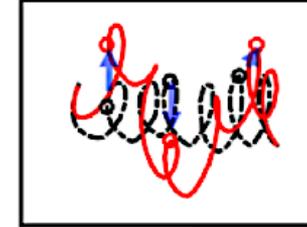
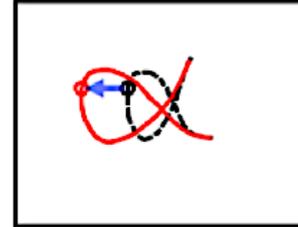
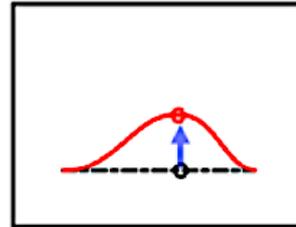


Via-points Adaptation with $h(x)$

If $h(u) = a_1u + a_0$



If $h(u) = \sum_{i=0}^5 a_i u^i$



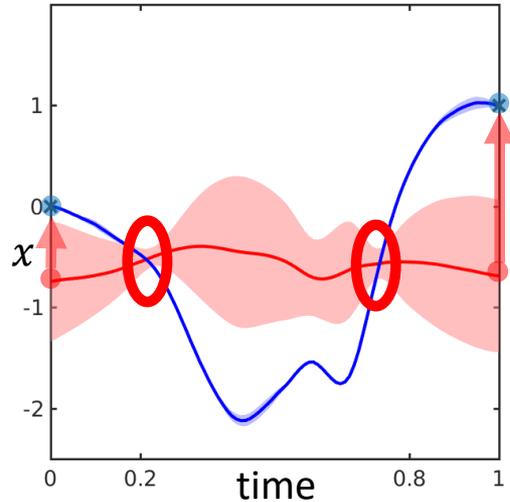
Via-points Adaptation with VMP

- For each via-point (u_{via}, x_{via}) , calculate $p(u_{via}, x_{via} | \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mathbf{h})$

Interpolation: $p(u_{via}, x_{via} | \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mathbf{h}) > \eta$  adjust $f(u)$

Extrapolation: $p(u_{via}, x_{via} | \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mathbf{h}) < \eta$  adjust $h(u)$

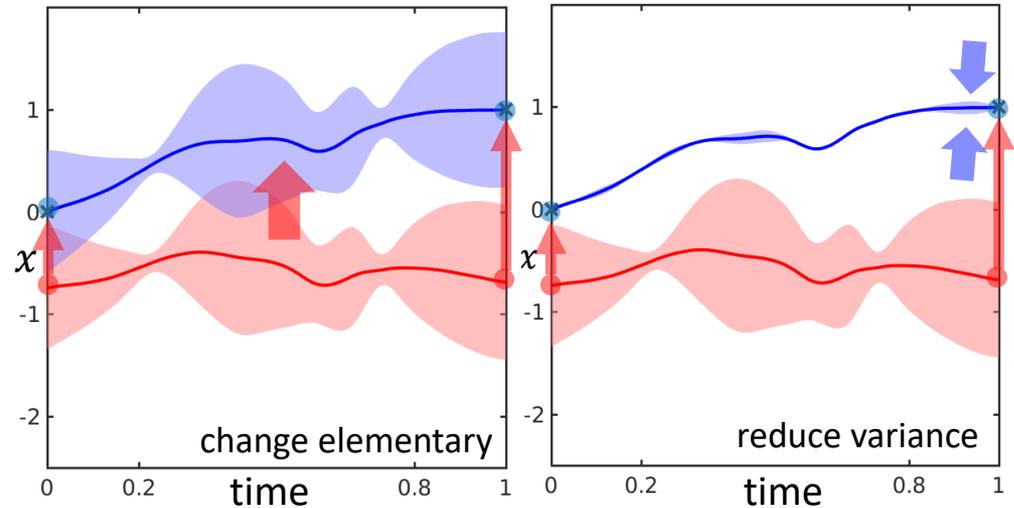
ProMP vs VMP for Via-points Adaptation



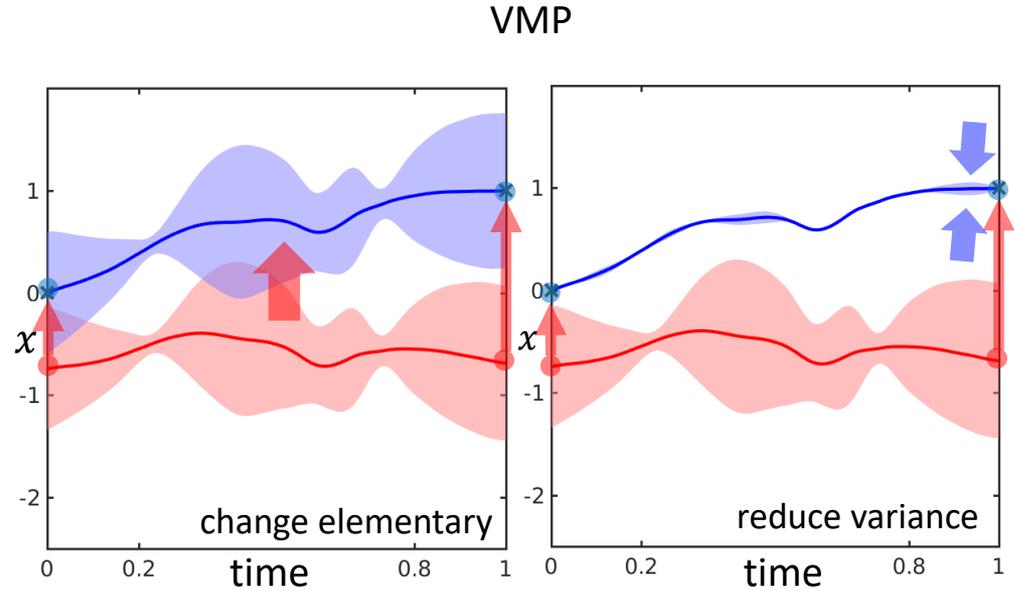
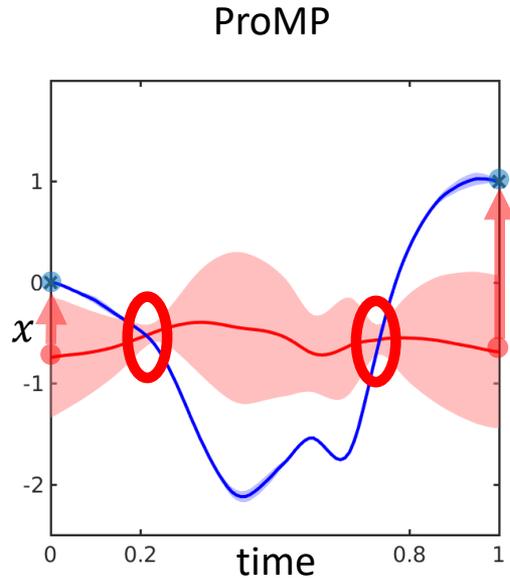
Because ProMP sticks to the low variance regions indicated by the red circles.

ProMP vs VMP for Via-points Adaptation

Then, VMP reduces the variance.
lets the mean trajectory meet the new start and goal.



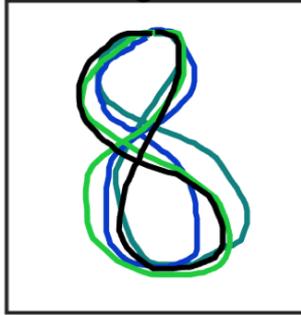
ProMP vs VMP for Via-points Adaptation



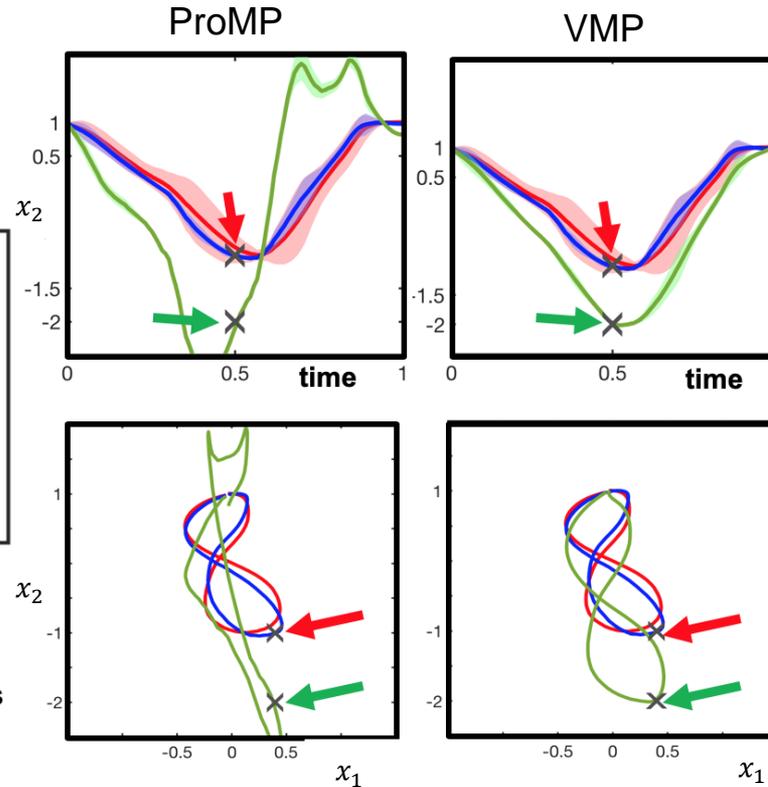
ProMP vs VMP for Via-points adaptation

- Trained MP
- Interpolation
- Extrapolation

Training data



- ➔ nearby via-points
- ➔ far away via-points



Action Generalization

So far, we introduced different action representations, but did not answer the question:
How can we generalize the action to different tasks?

- Represent different tasks with **task parameters**,
e.g.

$$\mathbf{q} = [q_1, q_2, \dots]^T$$

task: throw the ball to a basket.

task parameter: the location of the basket



Movement Primitive Generalization using MNDs

task: throw ball; task parameter: target location



mode 1: throw the ball directly to the target



mode 2: bounce the ball back from the wall

Y. Zhou, J. Gao and T. Asfour, "Movement Primitive Learning and Generalization: Using Mixture Density Networks," in *IEEE Robotics & Automation Magazine*, pp. 22-32, June 2020

Movement Primitive Generalization using MNDs

task: throw ball; task parameter: target location



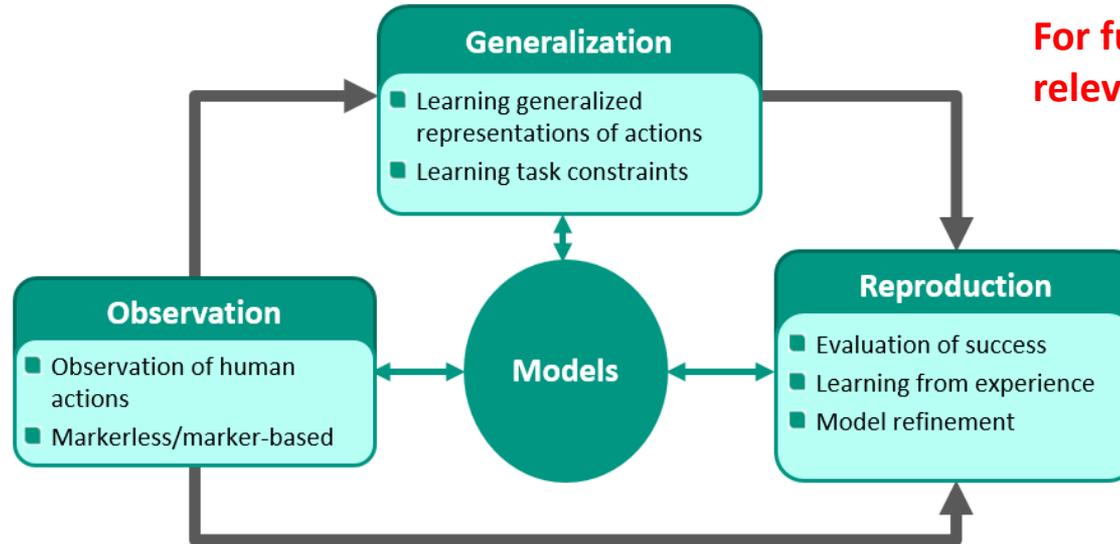
mode 1: throw the ball directly to the target



mode 2: bounce the ball back from the wall

Y. Zhou, J. Gao and T. Asfour, "Movement Primitive Learning and Generalization: Using Mixture Density Networks," in *IEEE Robotics & Automation Magazine*, pp. 22-32, June 2020

From Observation to Reproduction with HMMs



For further reading; not relevant for the exam

T. Asfour et al. (2006, 2008). Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots. International Journal on Humanoid Robots, 2008. (International Conference on Humanoid Robots, 2006)

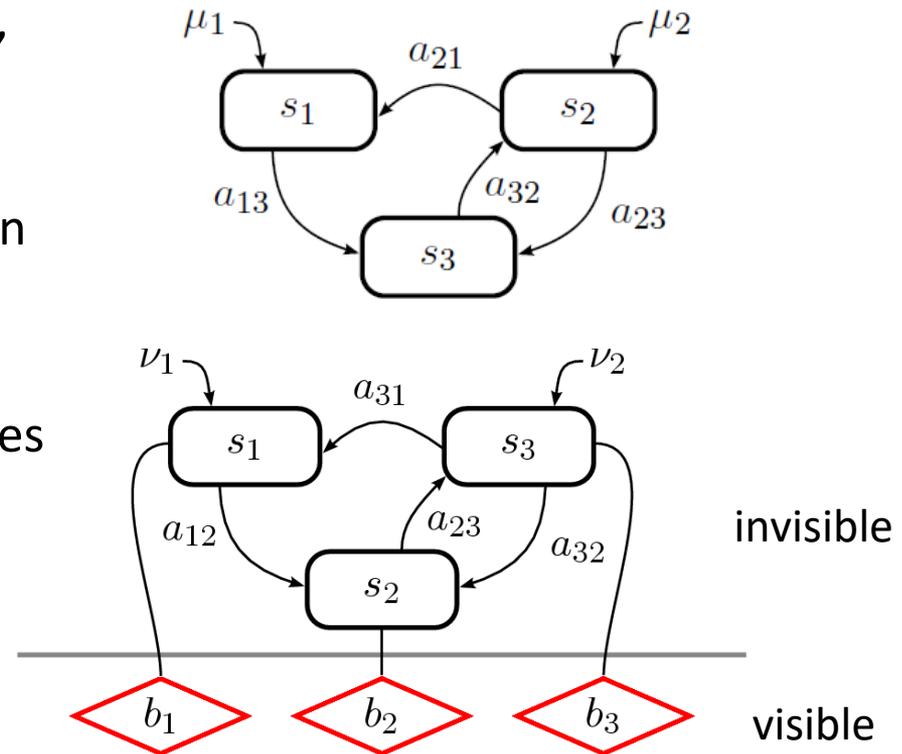
Action Representation with HMMs

- Stochastic representation using Hidden Markov Models (HMM)
 - Extract keypoints (KP) in the demonstration
 - Determine keypoints that are common in multiple demonstrations (common keypoints: CKP)
- HMMs for segmentation, generalization and reproduction

Asfour, T., Azad, P., Gyurfas, F. and Dillmann, R., *Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots*, International Journal of Humanoid Robotics (IJHR), vol. 5, no. 2, pp. 183-202, December, 2008

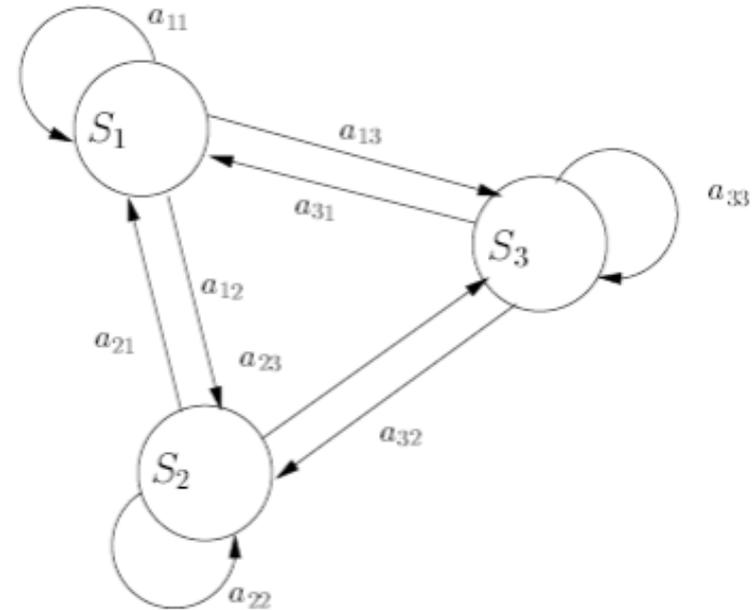
Hidden-Markov-Model (I)

- HMMs are first order Markov chains, i.e., the subsequent state depends exclusively on the current state
- States are not observable, but we can observe the output they generate. Therefore “hidden”
- In speech: sounds are known, syllables are hidden



Hidden-Markov-Model (II)

- Suitable for the classification of **time series data**, such as speech or gestures signals
- One HMM contains
 - States S_i
 - Transition probabilities a_{ij}
 - Start probabilities π_{ij}
 - Observation probabilities b_i for each state S_i
 - Discrete / continuous
 - Continuous case: probability density with mean μ_i und covariance matrix U_i



Example: One Day of a Baby

- **Markov-Chain:** States set $S = \{\text{wach, hungrig, schlafend}\}$

A	$X_{i+1} = \text{wach}$	$X_{i+1} = \text{hungrig}$	$X_{i+1} = \text{schlafend}$
$X_i = \text{wach}$	0,5	0,4	0,1
$X_i = \text{hungrig}$	0,2	0,1	0,7
$X_i = \text{schlafend}$	0,3	0	0,7

μ	$X_0 = \text{wach}$	$X_0 = \text{essend}$	$X_0 = \text{schlafend}$
	0,7	0	0,3

- **HMM:** A neighbor does not know what the baby is doing

	B	ruhig	schreiend
$X_i = \text{wach}$		0,5	0,5
$X_i = \text{hungrig}$		0,1	0,9
$X_i = \text{schlafend}$		1	0

Three Problems

Given a model λ and one observation sequence $O = O_1, \dots, O_n$

- How to efficiently calculate $P(O|\lambda)$, i.e., the probability that O is generated by λ ?
- How to determine the most likely sequence of states which generated the observation sequence O ?
- How to find the parameters λ that maximize $P(O|\lambda)$?

HMM Basic Algorithms

■ Forward-Algorithm

- Given an observation sequence $O = O_1, \dots, O_n$ and a model λ
- How to efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

■ Viterbi-Algorithm

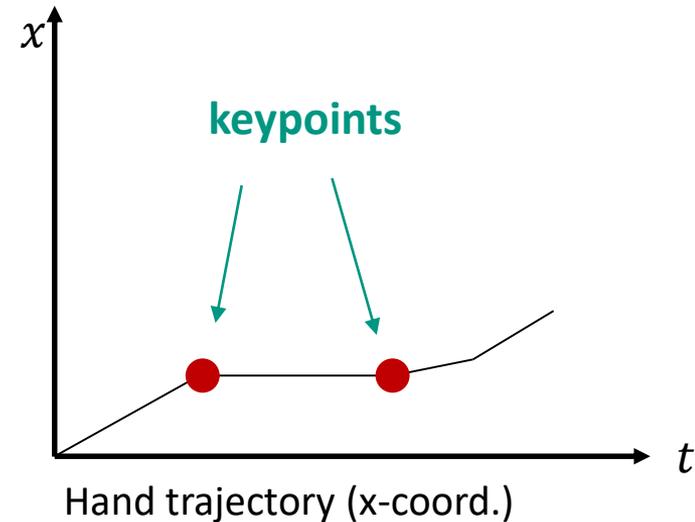
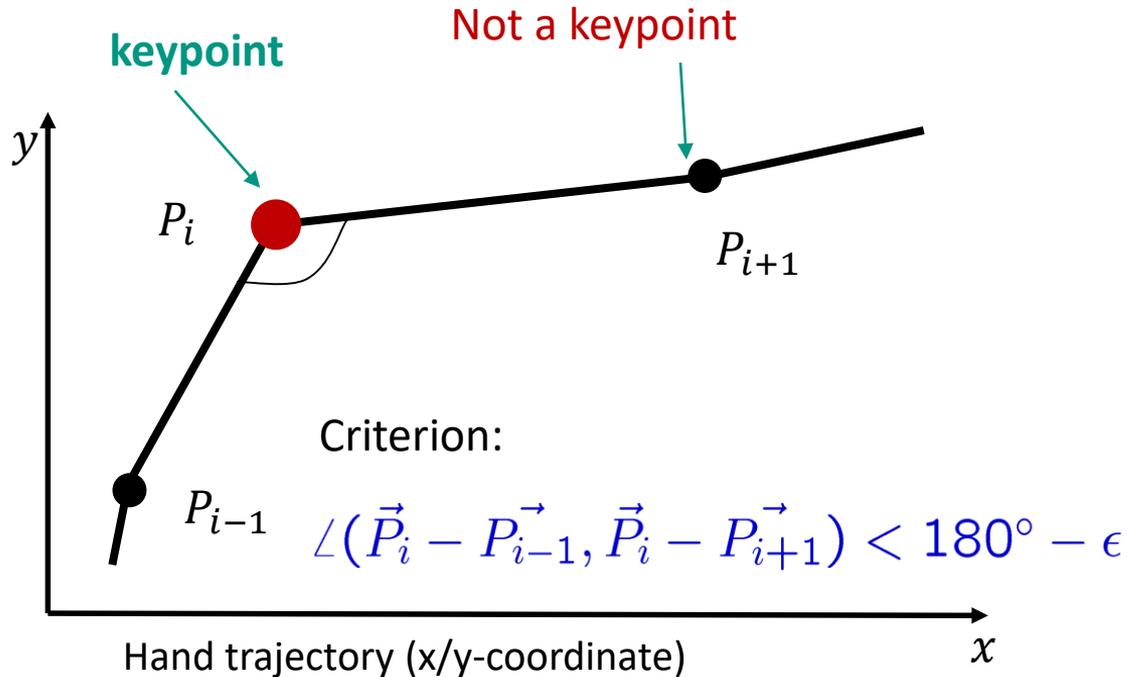
- Given an observation sequence $O = O_1, \dots, O_n$ and a model λ
- How to find a corresponding state sequence $S = S_1, \dots, S_n$ which is optimal in some sense (e.g., which “explains” the observation sequence best)

■ Baum-Welch-Algorithm

- How to adjust the model parameters λ (**training of the HMM**) to maximize $P(O|\lambda)$

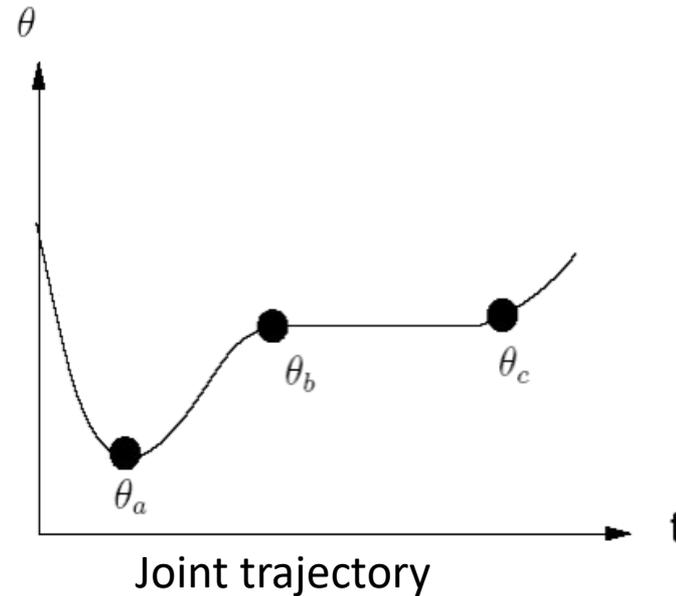
Segmentation - Keypoints (I)

Local minima and maxima and also pauses



Segmentation - Keypoints (II)

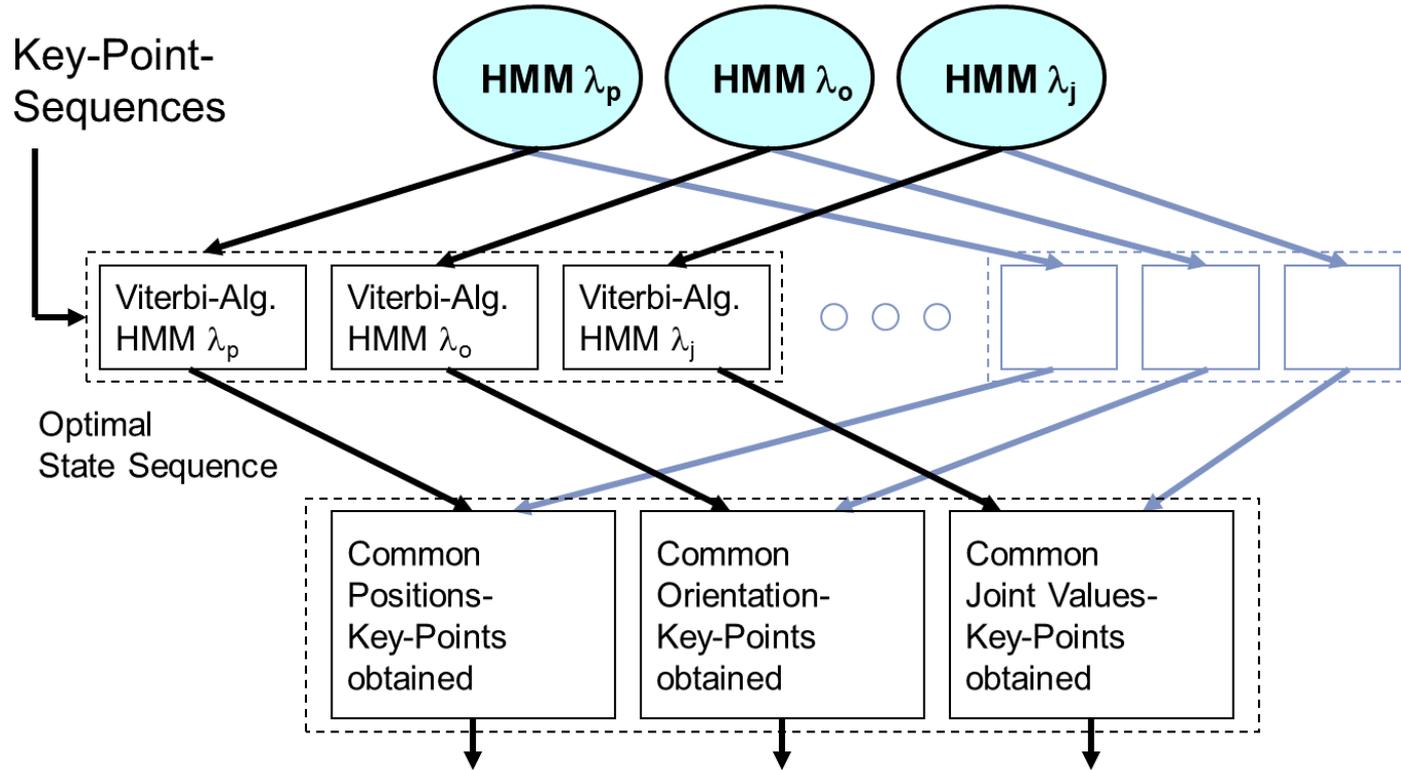
Local minimum and maximum and also pauses
(in position, velocity, ... trajectories)



Common Keypoints

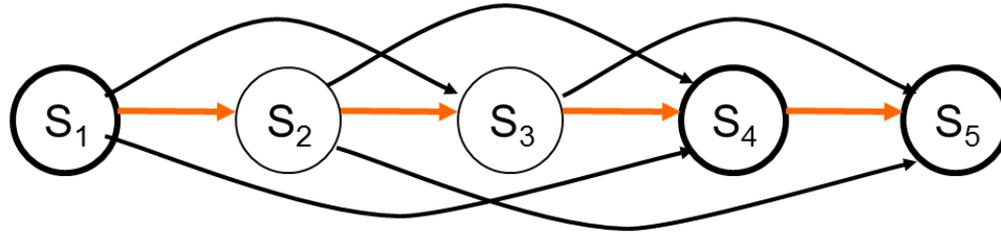
- Relevant to a generalized movement: Keypoints that occur in (almost) all demonstrations
- Similar keypoints from the various demonstrations are **common keypoints** $C_i = (K_1, \dots, K_D)$
- How to find common keypoints?
 - Usually used for such problems: Dynamic Programming Matching (Dynamic-Time-Warping, DTW)
 - Alternative: HMM
 - Viterbi algorithm provides optimal state sequences for the keypoint sequences of the various demonstrations
 - Common keypoints are obtained from states that occur in all sequences

Common Keypoints



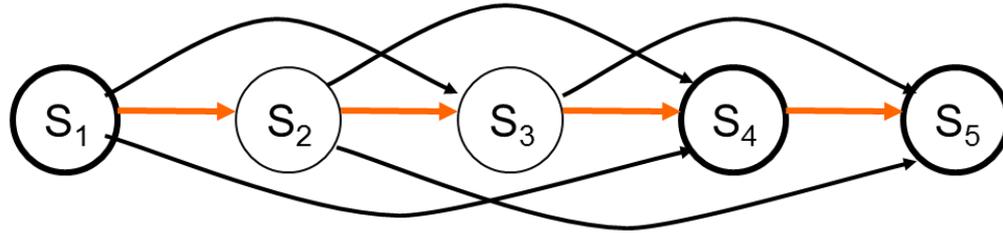
Common Keypoints (Example)

Demonstration 1:

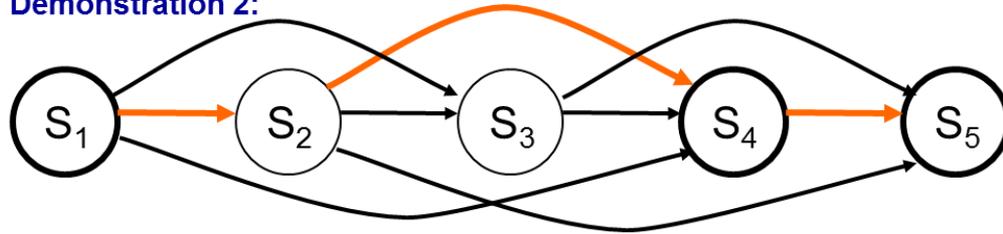


Common Keypoints (Example)

Demonstration 1:

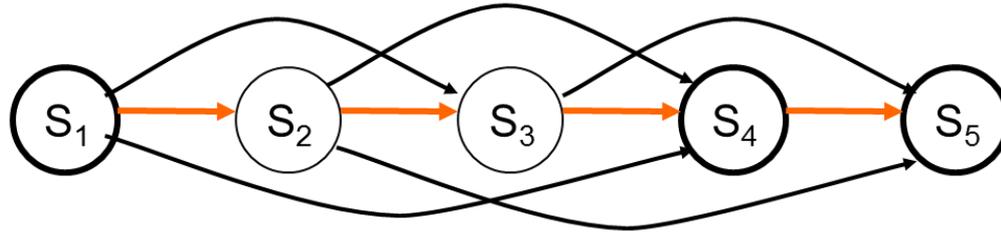


Demonstration 2:

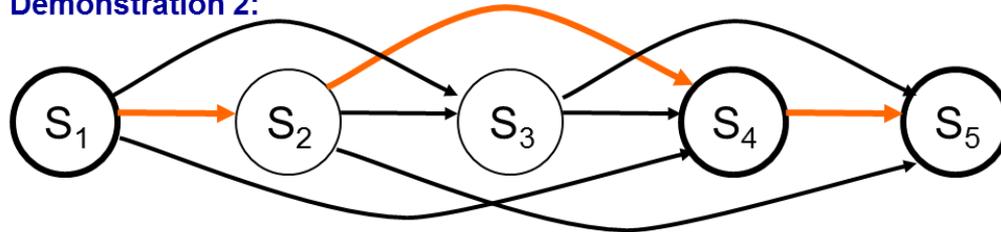


Common Keypoints (Example)

Demonstration 1:

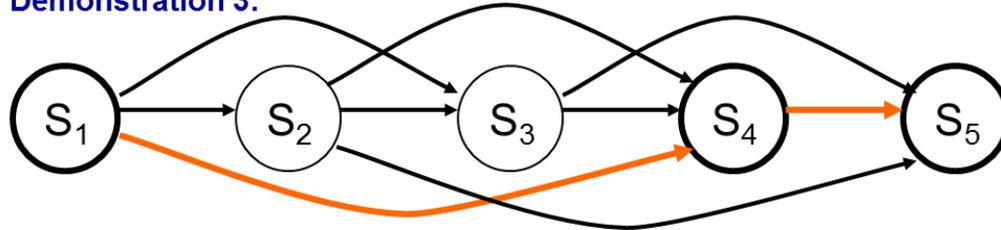


Demonstration 2:



Only the states S_1 , S_4 and S_5 represent common keypoints

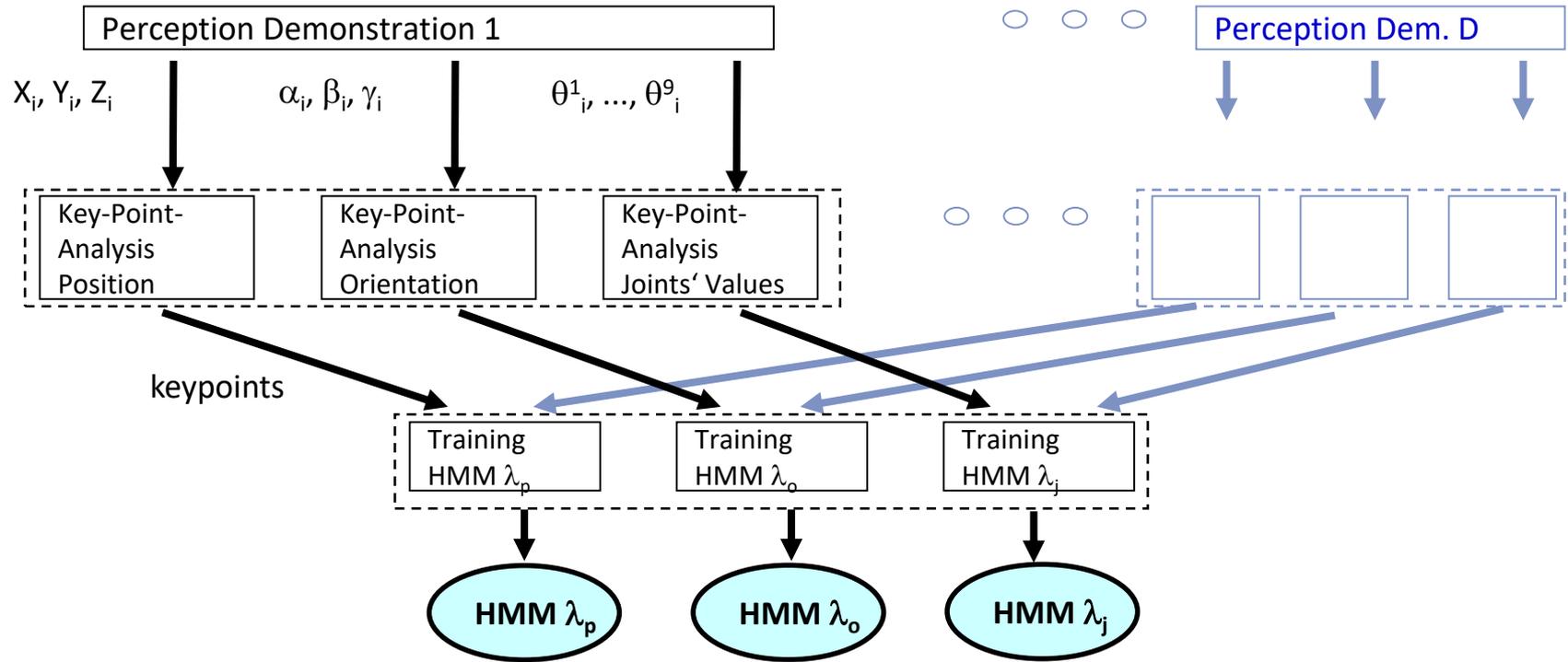
Demonstration 3:



Approach

- Both joint angles as well as positions and orientations of the hand are recorded and three different HMMs (TCP position, TCP orientation and joint angles) are trained
- Model of a human arm with 9 DOF
- Depending on the priority – imitation of the exact TCP trajectory or, if possible, arm positions – the influence of the respective HMM on the reproduction can be controlled by a set of weighting factors

Analysis in Detail: Training HMMs



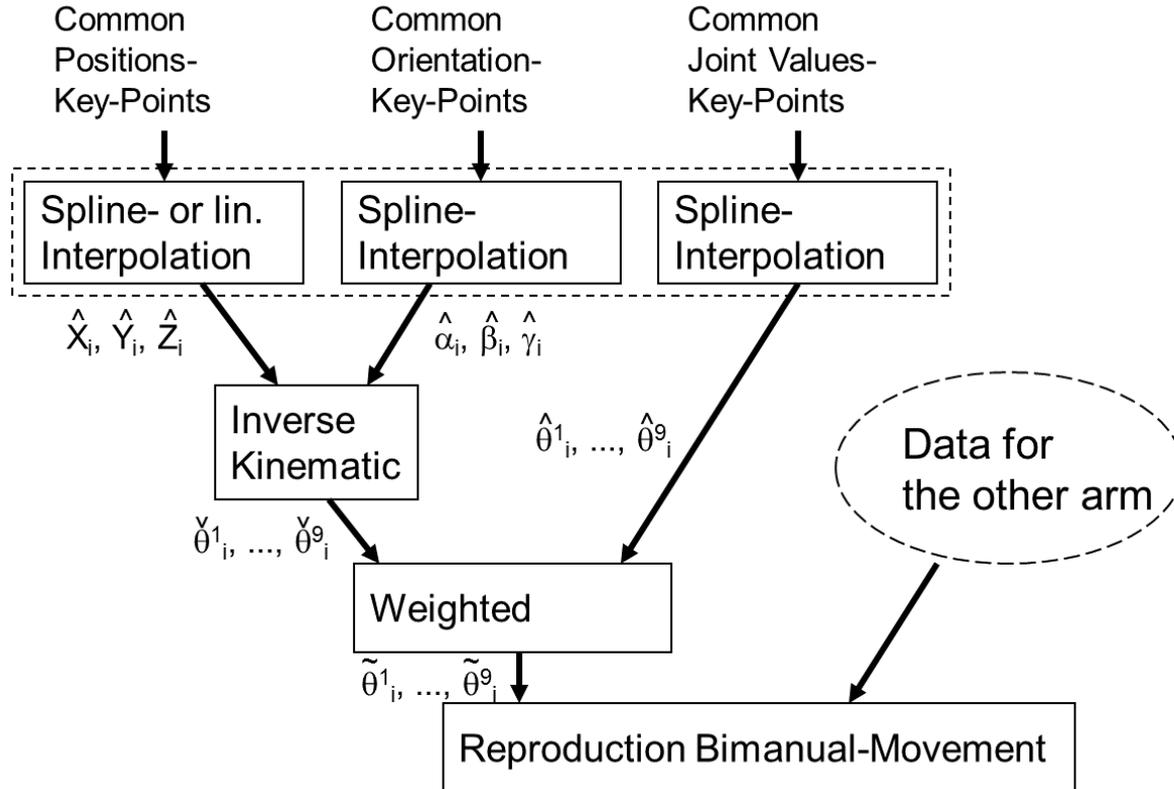
Recap: Keypoints

- Keypoints are prominent/distinctive points of a movement
- Purpose to determine these points
 - Data reduction (important for HMM)
 - Movement is represented exclusively by characteristic points
→ Better comparability of trajectories
- Similar keypoints from the various demonstrations are **common keypoints** $C_i = (K_1, \dots, K_D)$

Training of HMM

- HMM is trained by multiple demonstrations
- **Continuous HMM:** Output probabilities are calculated from probability densities of normal distribution
- After training, mean values of these distributions can be calculated to form an “average” trajectory
- Which states should be considered for the reproduction?
→ **Common Keypoints**

Extraction of Common Keypoints



Generalization/Reproduction

- Interpolation between common keypoints (separately for position CKP, orientation CKP and joint angle CKP)
- Subsequently, inverse kinematics to calculate joint angle θ_i^j from position and orientation

- Reproduction:
$$\tilde{\theta}_i^j = \omega \cdot \hat{\theta}_i^j + (1 - \omega) \cdot \check{\theta}_i^j \quad \omega \in [0, 1]$$

 **Joints**  **IK**

- Irrespective of the approach presented here, interesting question:

What temporal relations must be taken into account when imitating a two-arm movement?

- Example: Pour (with two arms)

Temporal Coordination

- But the demonstrations offer more relevant information than just the common key points: **What temporal relations must be taken into account?**
- For each common keypoint $C_i = (\kappa_1, \dots, \kappa_D)$, a vector of timestamps of the individual keypoints can be specified: (τ_1, \dots, τ_D)
- A common keypoint C_i is strongly ordered before another common keypoint C_j if:

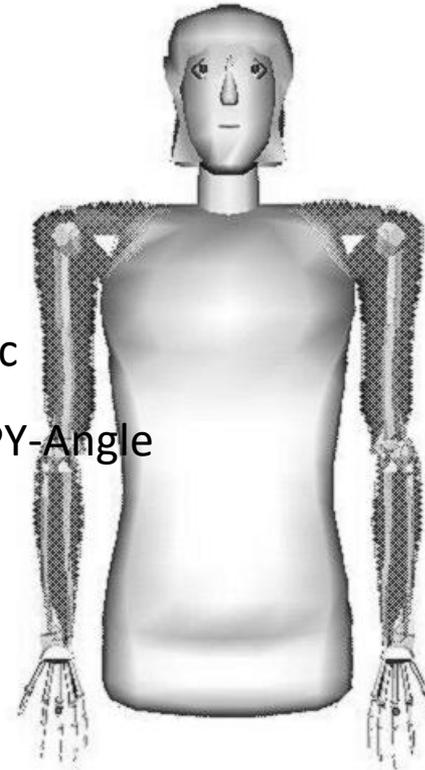
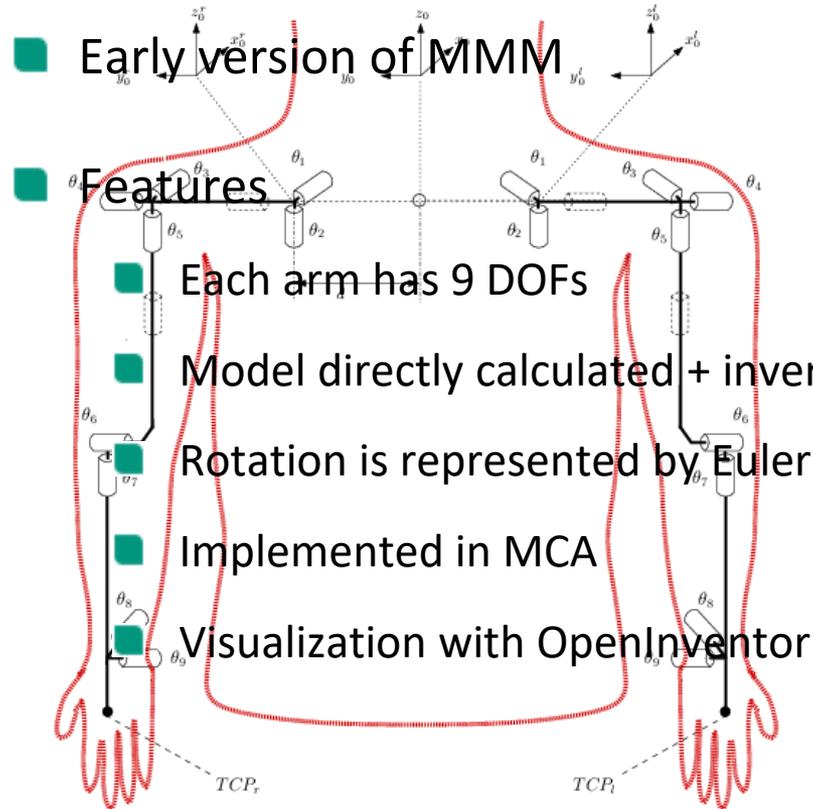
$$\forall k: \tau_i^k < \tau_j^k$$

k is the number of the demonstrations

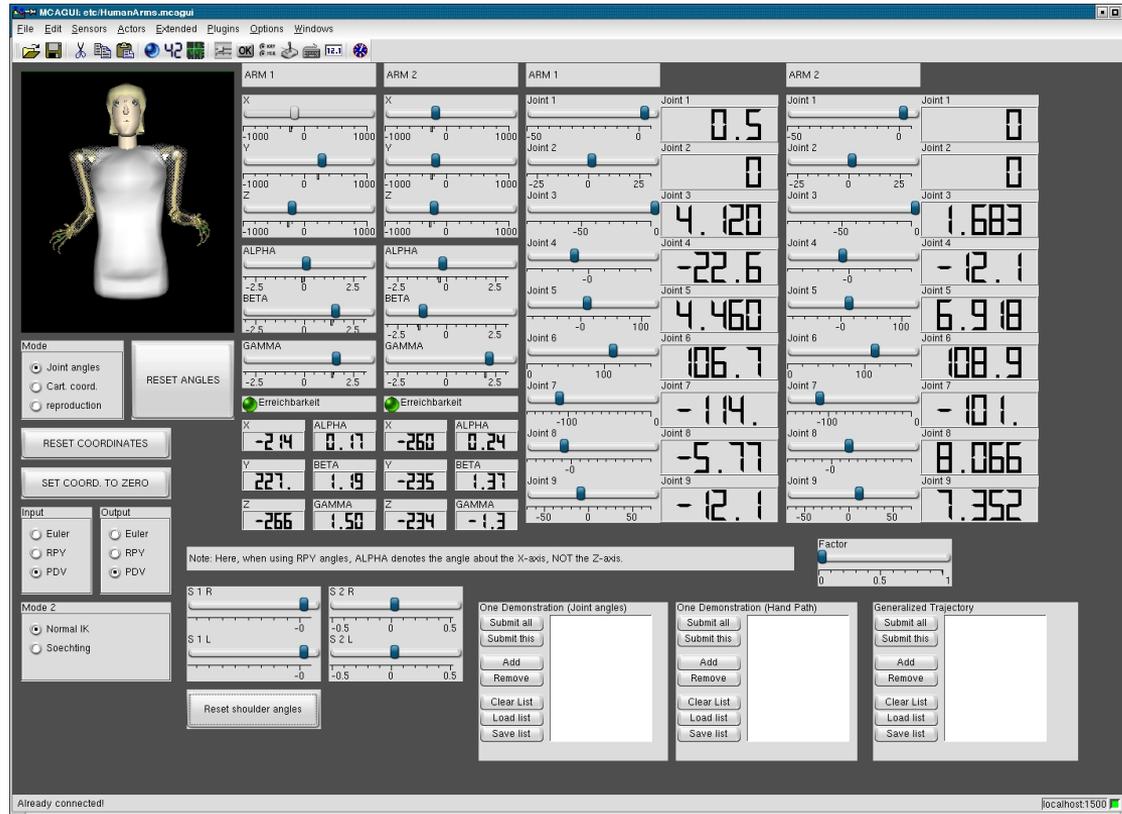
Reproduction: The Kinematic Model (I)

- Early version of MMM
- Features
 - Each arm has 9 DOFs
 - Model directly calculated + inverse kinematic
 - Rotation is represented by Euler-Angle or RPY-Angle
 - Implemented in MCA
 - Visualization with OpenInventor

Reproduction: The Kinematic Model (I)



Reproduction: MCA-User Interface



The screenshot shows the MCAGUI software interface for a humanoid robot arm. The interface is divided into several sections:

- 3D Viewport:** Displays a 3D model of a humanoid robot's upper body and arms.
- ARM 1 and ARM 2 Controls:** Each arm has a set of sliders for X, Y, and Z coordinates, and three joint angle sliders (ALPHA, BETA, GAMMA).
- Joint Angle Readouts:** A central panel displays the current values for Joint 1 through Joint 9. For example, Joint 1 is 0.5, Joint 2 is 0, Joint 3 is 0, Joint 4 is 4.120, Joint 5 is -22.6, Joint 6 is 4.460, Joint 7 is 106.7, Joint 8 is -5.77, and Joint 9 is -12.1.
- Mode Selection:** Includes options for 'Joint angles', 'Cart. coord.', and 'reproduction'. There are also 'RESET ANGLES' and 'RESET COORDINATES' buttons.
- Coordinate Readouts:** Displays current X, Y, Z coordinates for both arms and their corresponding ALPHA, BETA, and GAMMA angles. For example, Arm 1 X is -2.14, Y is 227, Z is -266.
- Input/Output Selection:** Allows switching between Euler, RPV, and PDV control methods.
- Shoulder Angle Controls:** Sliders for S 1 R, S 2 R, S 1 L, and S 2 L, with a 'Reset shoulder angles' button.
- Demonstration Lists:** Three panels for 'One Demonstration (Joint angles)', 'One Demonstration (Hand Path)', and 'Generalized Trajectory', each with 'Submit all', 'Submit this', 'Add', 'Remove', 'Clear List', 'Load list', and 'Save list' buttons.
- Factor Control:** A slider for adjusting a 'Factor' from 0 to 1.
- Status Bar:** Shows 'Already connected!' and 'localhost:1500'.

Problem: No Joint Angles

- Problem: Joint angles were not present (in 2006), which are necessary for the approach
- Solution?
 - Now: KIT human motion database and MMM
 - In 2006: How to create human-like joint angles from end-effector trajectories?
Sensorimotor transformation model for human-like arm positions according to Soechting and Flanders

J.F. Soechting, M. Flanders. Errors in Pointing are Due to Approximations in Targets in Sensorimotor Transformations. *Journal of Neurophysiology*, 62(2):595-608, 1989

J.F. Soechting, M. Flanders. Sensorimotor Representations for Pointing to Targets in Three-Dimensional Space. *Journal of Neurophysiology*, 62(2):582-594, 1989

Soechting-Angle (I)

■ Calculation of Soechting-Angle

$$r^2 = x^2 + y^2 + z^2$$

$$\tan \chi = \frac{x}{y}$$

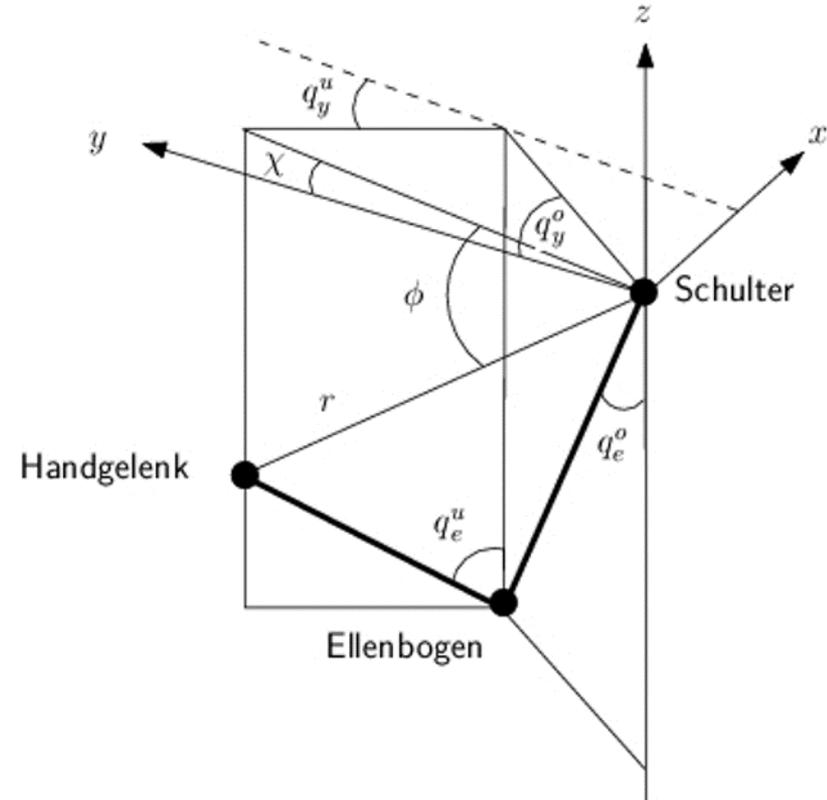
$$\tan \phi = \frac{z}{\sqrt{x^2 + y^2}}$$

$$q_e^o = -4.0 + 1.10r + 0.90\phi$$

$$q_e^u = 39.4 + 0.54r - 1.06\phi$$

$$q_y^o = 13.2 + 0.86\chi + 0.11\phi$$

$$q_y^u = -10.0 + 1.08\chi - 0.35\phi$$



Soechting-Angle (II)

How to get angles from the Soechting-angles?

- Set θ_1 and θ_2 to 0
- Calculate $\theta_3 - \theta_6$ from Soechting-angles:

$$\theta_3 = \arcsin\left(\frac{\sin(q_e^o) \cdot l_o \cdot \sin(q_e^u)}{\cos(\theta_4) \cdot l_o}\right)$$

$$\theta_4 = \arcsin\left(\frac{\sin(q_e^o) \cdot l_o \cdot \cos(q_e^u)}{l_o}\right)$$

$$\theta_6 = \arccos(\sin(q_e^o) \cdot \sin(q_y^o) \cdot \sin(q_e^u) \cdot \sin(q_y^u) + \sin(q_e^o) \cdot \cos(q_y^o) \cdot \sin(q_e^u) \cdot \cos(q_y^u) - \cos(q_e^o) \cdot \cos(q_e^u))$$

Soechting-Angle (III)

$$\vec{p} = (\vec{l}_u \times \vec{l}_o) \times \vec{l}_o$$

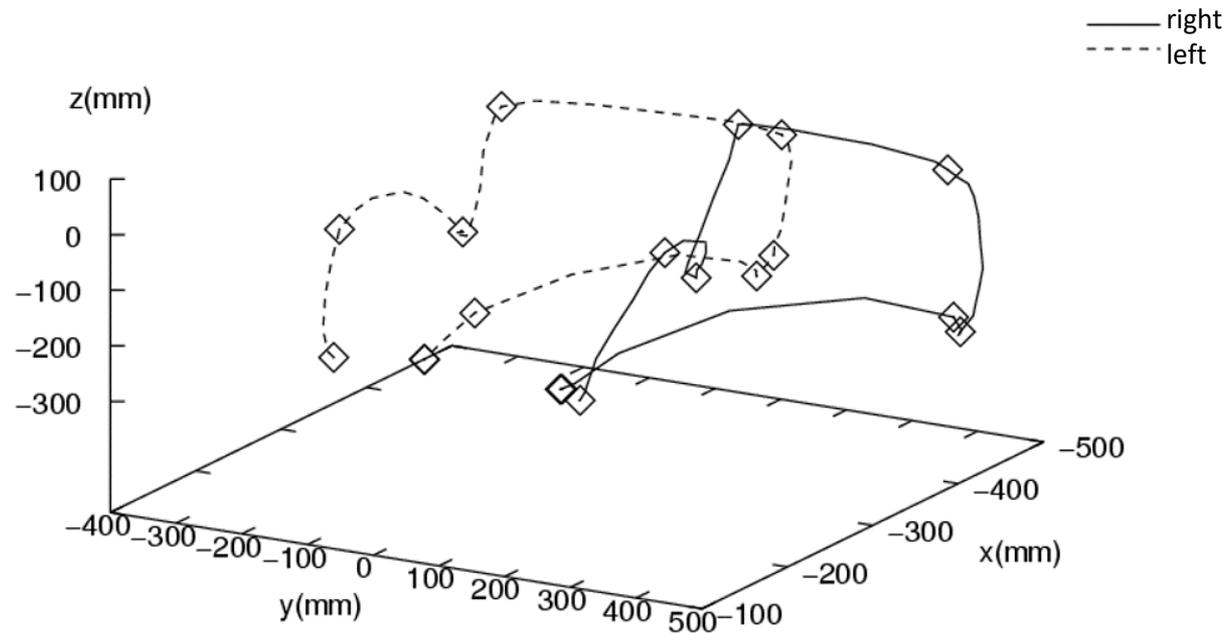
$$\vec{q} = (\vec{l}_{u'} \times \vec{l}_o) \times \vec{l}_o$$

$$\theta_5 = \arccos \left(\frac{\langle \vec{p}, \vec{q} \rangle}{\|\vec{p}\| \cdot \|\vec{q}\|} \right)$$

- Joint angle $\theta_7 - \theta_9$ can be calculated by inverse kinematics
- If the target position is not reached, $\theta_3 - \theta_6$ are changed step by step until an acceptable solution is found

Experiment (I)

Sample demonstration pick-and-place with box (Keypoints are also drawn)

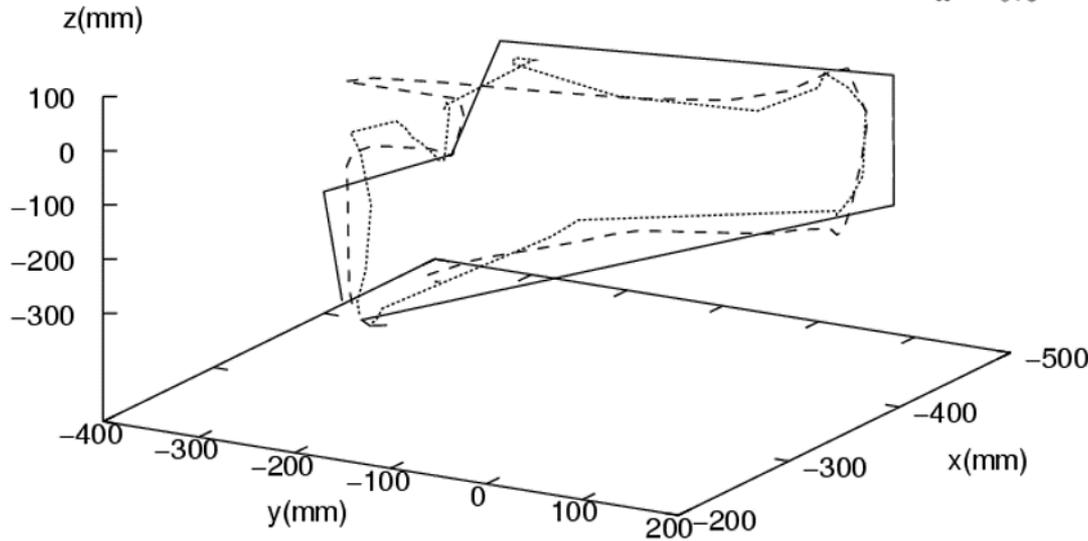


Experiment (II)

Comparison of the generalized trajectories of the left hand for different weighting factors

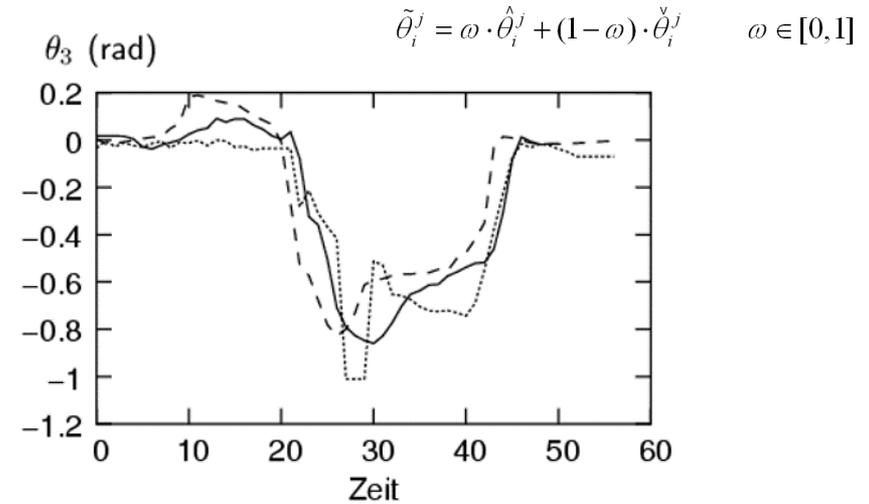
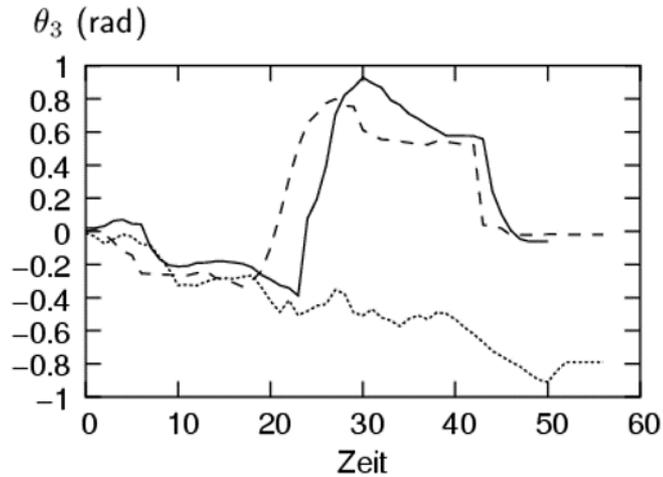
$$\tilde{\theta}_i^j = \omega \cdot \hat{\theta}_i^j + (1-\omega) \cdot \check{\theta}_i^j \quad \omega \in [0,1]$$

$\omega = 0$ ——— (solid line)
 $\omega = 1$ (dotted line)
 $\omega = 0.5$ - - - - (dashed line)



Experiment (III)

Joint angle trajectories for a selected joint (θ_3)



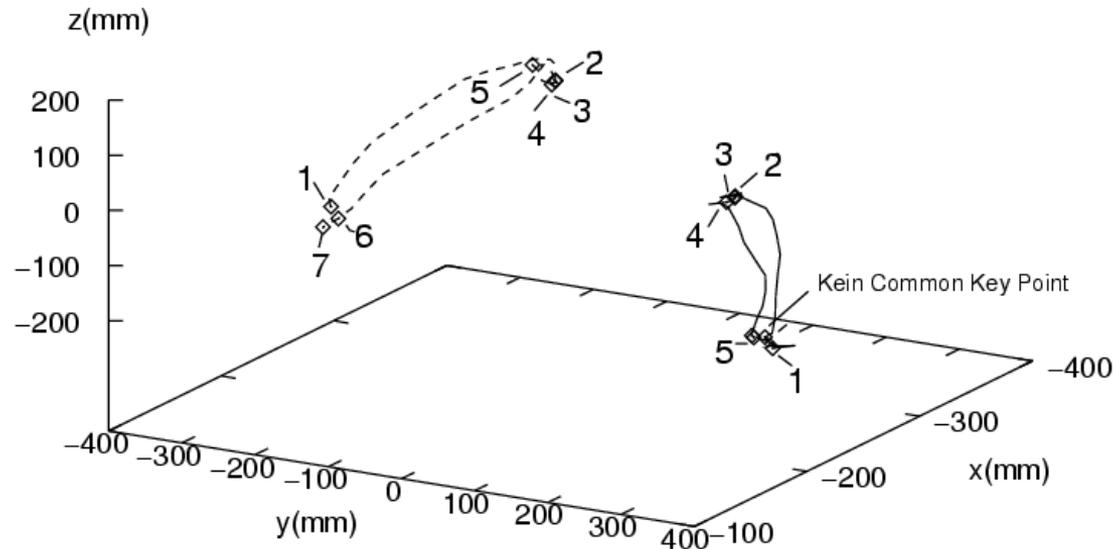
— Beispiel-Demonstration $\omega = 0$ - - - - $\omega = 1$

Experiment (IV)

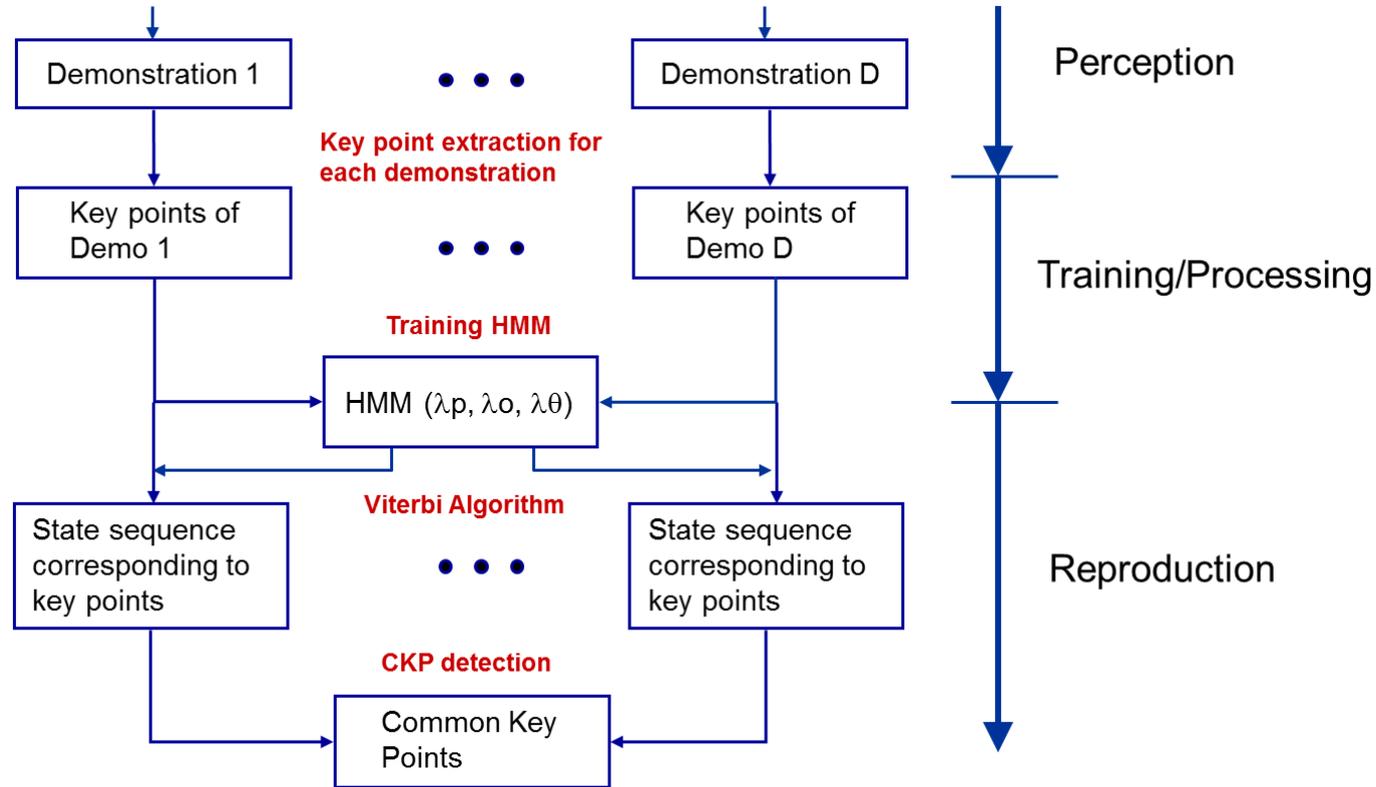
Insertion movement (right hand holds glass, left hand bottle)

— right
- - - left

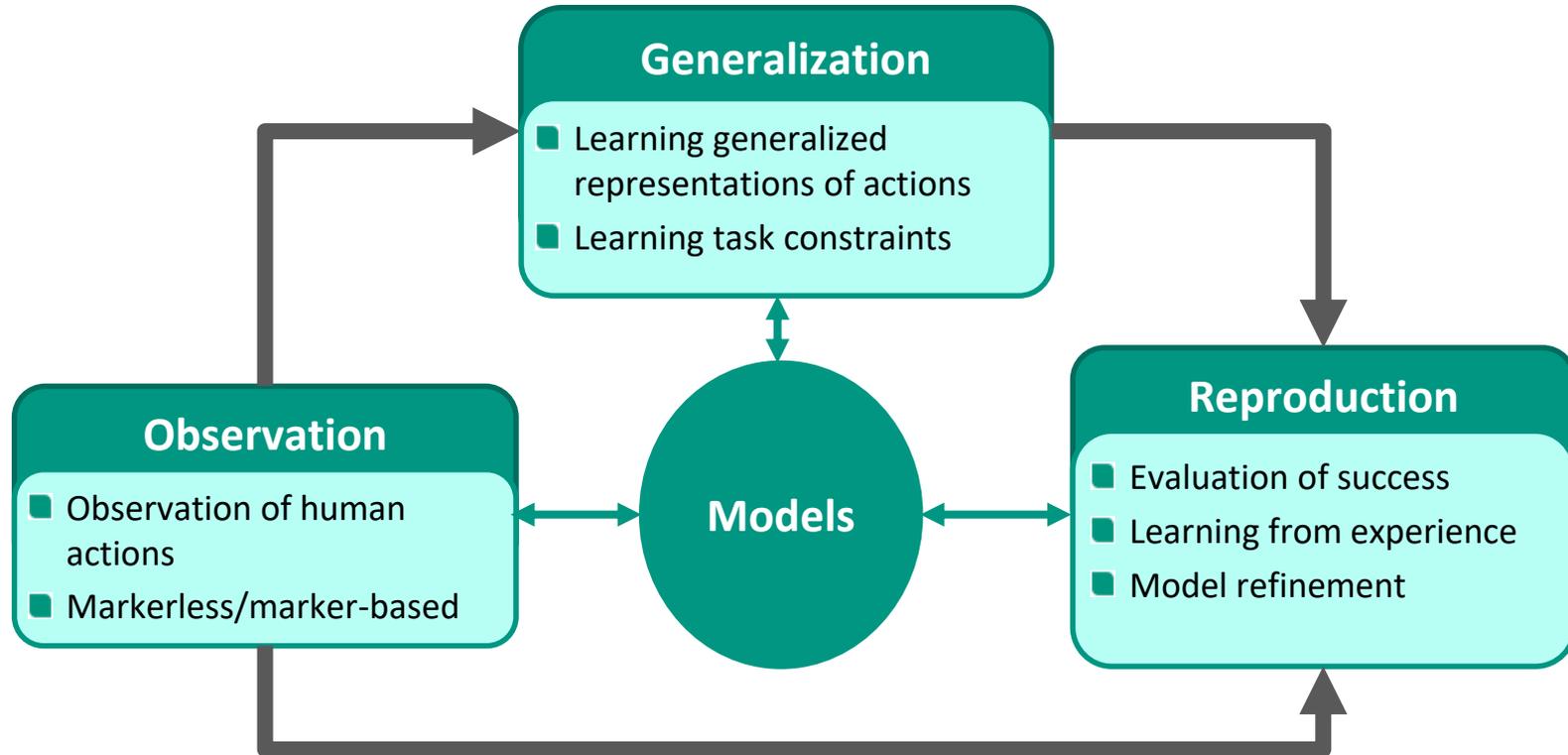
$$\tilde{\theta}_i^j = \omega \cdot \hat{\theta}_i^j + (1 - \omega) \cdot \check{\theta}_i^j \quad \omega \in [0, 1]$$



Approach: From Perception to Reproduction



Ende!

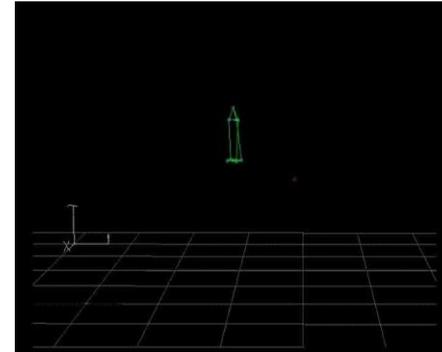
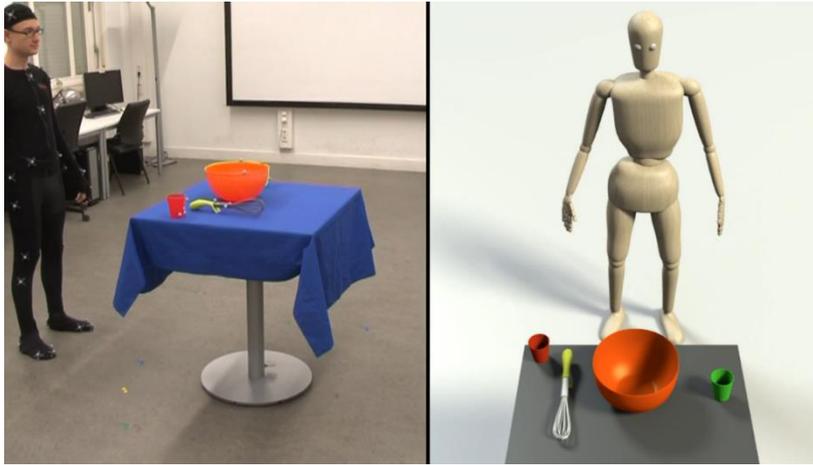


Observation

- Motion capture techniques
 - Marker-based systems
 - Passive markers (VICON)
 - Active markers
 - IMU based
 - Markerless
 - From Images (RGB, RGB-D)
 - Exoskeletons

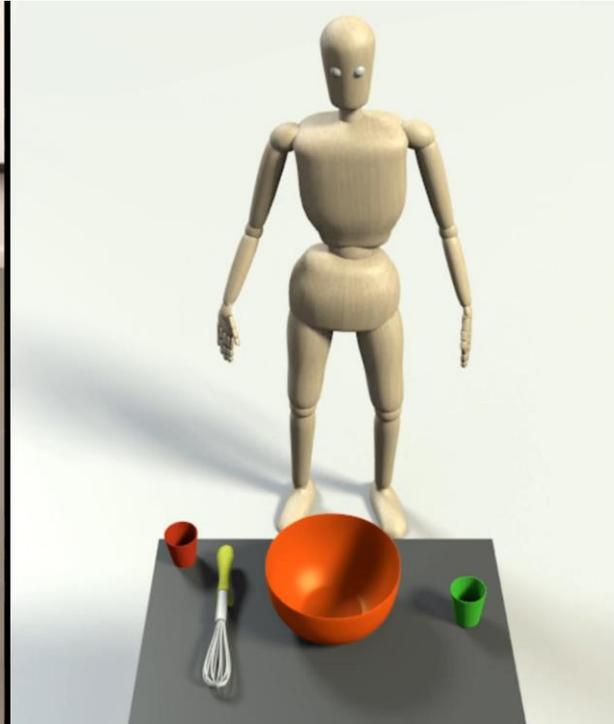
Observation

■ Marker-based motion capture techniques



Capturing and Reconstruction

- ... of whole-body **and** objects motions



KIT Whole-Body Human Motion Database

<https://motion-database.humanoids.kit.edu/>



■ Data for learning a robot motion alphabet

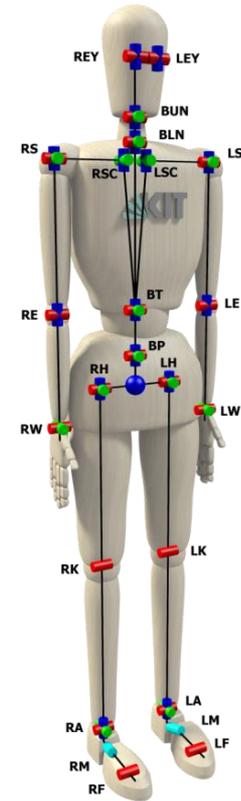
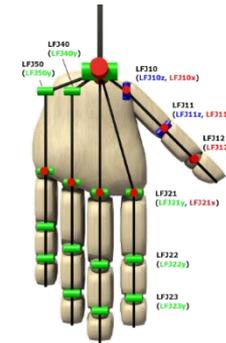


- **38,4 hours** of manually labeled human motion data (including object information)
- **9375** motions
- **224** (106/37) subjects
- **151** objects
- **2.0 TB**

motion-database.humanoids.kit.edu
<https://gitlab.com/mastermotormp>

The Master Motor Map (MMM)

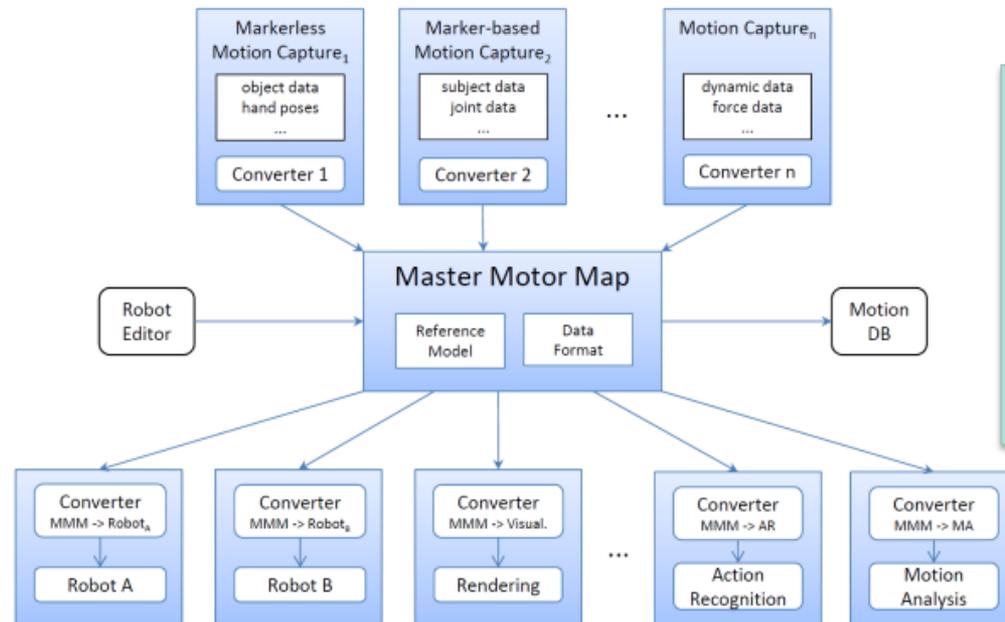
- Framework for the **unifying representation** of whole-body motion, motion analysis and its transfer to humanoid robots
- **Reference model of the human body with currently 104 DoF**
 - Kinematics and dynamics model
 - Joint Limits (min/max)
 - Center of Mass
 - Mass percentage of total weight
 - Inertia Tensor
 - Kinematics/Dynamics properties scalable regarding subject height and weight (scaled by MMM model processor)
 - **Subject-specific** parameters



T-RO 2016

The Master Motor Map (MMM), see Chapter 2

- **Unifying framework** for capturing, representation, visualization and whole body human motion and mapping/converting to different embodiments



Red: relevant for the exam

O. Terlemez et al., “Master Motor Map (MMM) – Framework and Toolkit for Capturing, Representing, and Reproducing Human Motion on Humanoid Robots”, IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2014

Mandery, C., Terlemez, Ö., Do, M., Vahrenkamp, N. and Asfour, T., *Unifying Representations and Large-Scale Whole-Body Motion Databases for Studying Human Motion*, IEEE Transactions on Robotics, vol. 32, no. 4, pp. 796-809, 2016

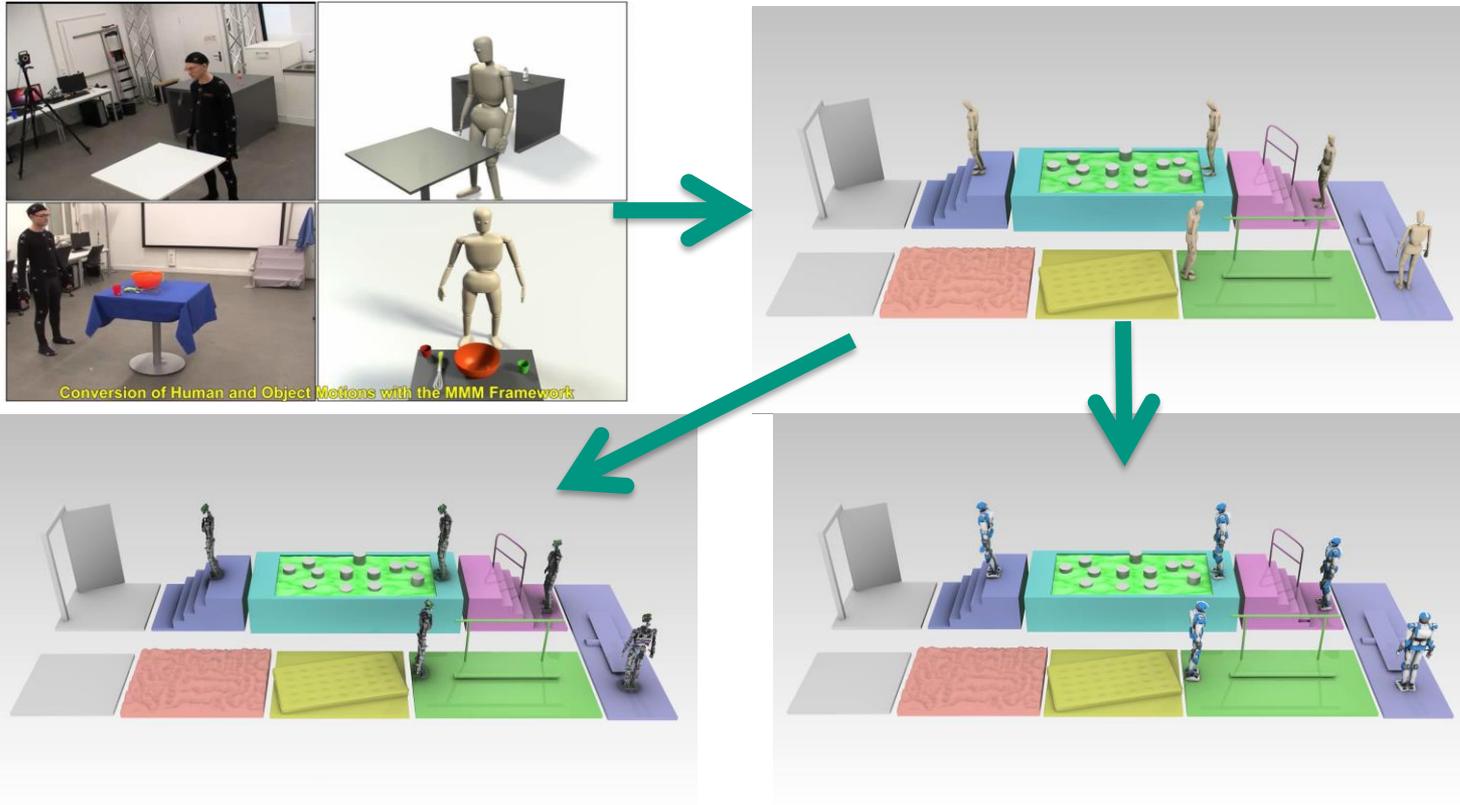
<https://mmm.humanoids.kit.edu>

<https://mmm.humanoids.kit.edu>

Motion Reproduction with MMM



From Human to Humanoid Motion with MMM



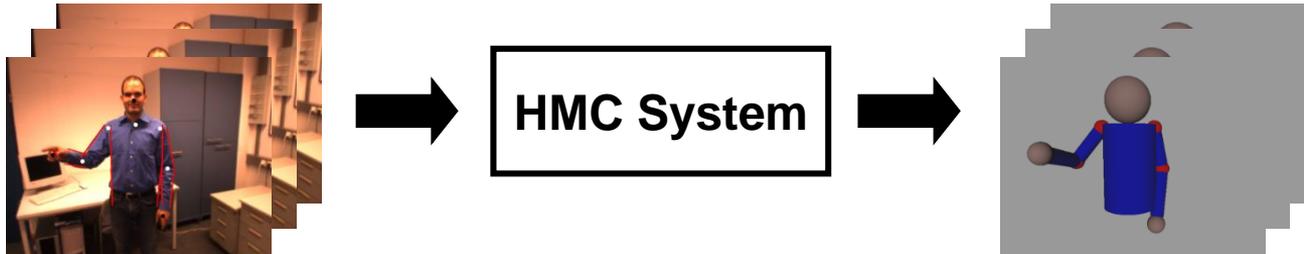
Observation

- Motion capture techniques
 - Marker-based systems
 - Passive markers (VICON)
 - Active markers
 - IMU based
 - **Markerless**
 - From Images (RGB, RGB-D)
 - Exoskeletons

Markerless Human Motion Capture

- Human Motion Capture (HMC):
 - The system operates on a simplified 3D human model
 - Output is a sequence of configuration vectors of this model, one for each frame
- Markerless:
 - The only input to the system is a sequence of stereo image pairs
 - No markers are used

Azad, P., Asfour, T. and Dillmann, R., *Robust Real-time Stereo-based Markerless Human Motion Capture*, IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 700-707, December, 2008

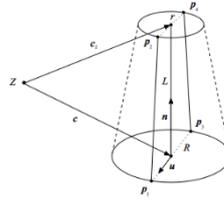


Stereo-Based 3D Human Motion Capture (HMC)

- Capture 3D human motion based on the image input from the cameras of the robot's head **only**
- **Approach**
 - Hierarchical Particle Filter framework
 - Localization of hands and head using color segmentation and stereo triangulation
 - Fusion of 3d positions and edge information
 - Half of the particles are sampled using inverse kinematics
- **Features**
 - Automatic Initialization
 - 30 fps real-time tracking on a 3 GHz CPU, 640x440 images
 - Smooth tracking of real 3D motion

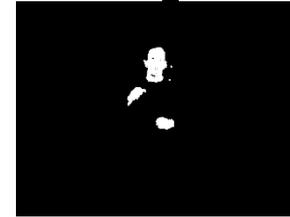


HMC Cues



Model projection

Image processing pipeline



Edge cue:

Operates on 2D image positions along the projected contour

$$w_g(I_g, P) = 1 - \frac{1}{|P|} \sum_{i=1}^{|P|} I_g(\mathbf{p}_i)$$

Distance cue:

Operates on 3D positions of the hands and the head

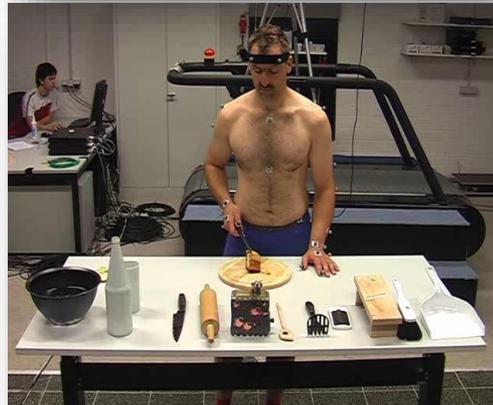
$$w_d(I_d, P) = \sum_{i=1}^{|P|} |\mathbf{p}_i - \mathbf{p}'_i(I_d)|^2$$

Basic fusion:

$$p(I_g, I_d | \mathbf{s}) \propto \exp \{-s_g w_g(I_g, f_g(\mathbf{s}))\} \cdot \exp \{-s_d w_d(I_d, f_d(\mathbf{s}))\}$$

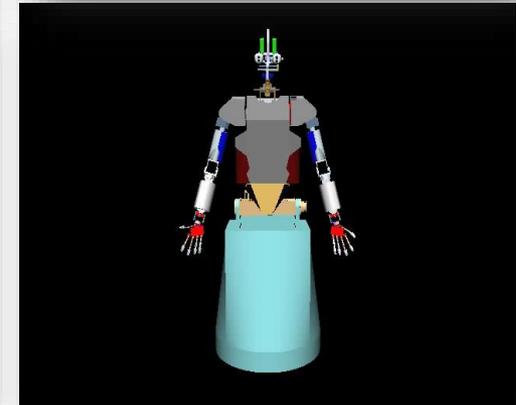
Motion Reproduction with MMM (I)

- Data from stereo-based marker-less human motion capture system
- Data from VICON system (SFB 588)

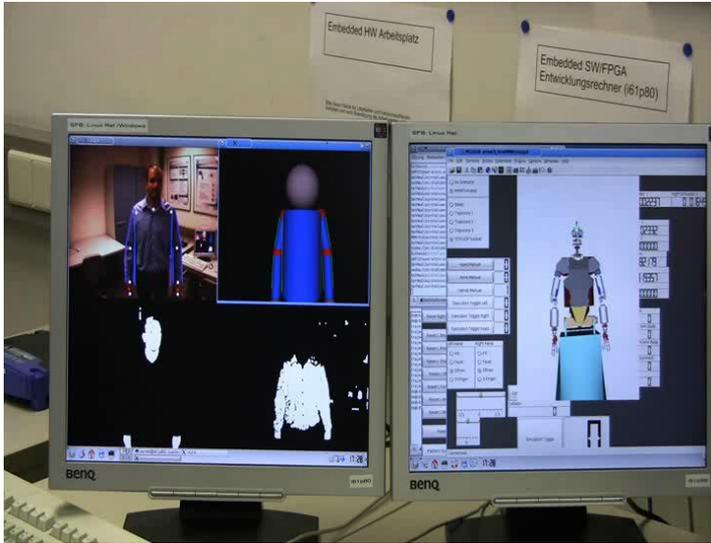


Motion Reproduction with MMM (II)

- Data from stereo-based marker-less human motion capture system
- Data from VICON system (SFB 588)



Motion Reproduction with MMM on ARMAR-IIIb



Azad, P., Asfour, T. and Dillmann, R., *Robust Real-time Stereo-based Markerless Human Motion Capture*, IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 700-707, December, 2008

OpenPose – CMU

- Real-time multi-person keypoint detection library for body, face, and hands estimation
- Code: <https://github.com/CMU-Perceptual-Computing-Lab/openpose>



<https://www.youtube.com/watch?v=C1Sxk6zxWLM>



Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y. A. „OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields“, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019)