FETPROACT-2-2014: Knowing, doing, being: cognition beyond problem solving

ACTION ACRONYM

TIMESTORM

ACTION FULL TITLE

"MIND AND TIME: INVESTIGATION OF THE TEMPORAL TRAITS OF HUMAN-MACHINE CONVERGENCE"

GRANT AGREEMENT NO:

641100

DELIVERABLE D2.2

ROBOTIC SIMULATOR

DUE DATE

JANUARY 1ST, 2016

RESPONSIBLE PARTNER

KARLSRUHE INSTITUTE OF TECHNOLOGY

AUTHORS

TAMIM ASFOUR, EREN ERDAL AKSOY

Table of Content

D2.2 Robotic Simulator

1 Executive Summary

This deliverable introduces the robot software framework ArmarX which provides a simulation environment for various robotics applications. In particular, we report here on the current status of ArmarX simulator and its possible applications which are at present being conducted by the Karlsruhe Institute of Technology (KIT) in collaboration with TimeStorm project partners.

The here introduced robotic simulator is a part of the ArmarX framework which is defined as an event-driven component-based robot development environment. In the context of the TimeStorm project, we extended the robotics simulation environment to allow generating human demonstrated manipulations by humanoid robots. In addition, we extended the software architecture to allow simultaneous simulation of multiple robots for collaborative manipulation tasks and introduced the virtual time concept within the simulator. For the experimental evaluation we set up new scene structures with various object models arranged in different contexts.

Further, we briefly describe the use of the simulator in two ongoing research activities and experiments in the project. The first experiment is performed in collaboration between KIT and the University of Groningen (UoG) and is concerned with measuring the temporal perception of real and simulated manipulation actions performed by humans. The second experiment is performed in collaboration between KIT and the Foundation for Research and Technology Hellas (FORTH) and covers the simulation of multi-agent collaborative task planning and execution with time constraints.

The deliverable is structured as follows. Section 2 gives a brief overview on the robot software framework ArmarX. In addition, we describe the ArmarX dynamic simulator, the virtual time concept, the setup of the simulation environment and object models. In Section 3 we introduce two applications that are currently being performed in the context of the TimeStorm project.

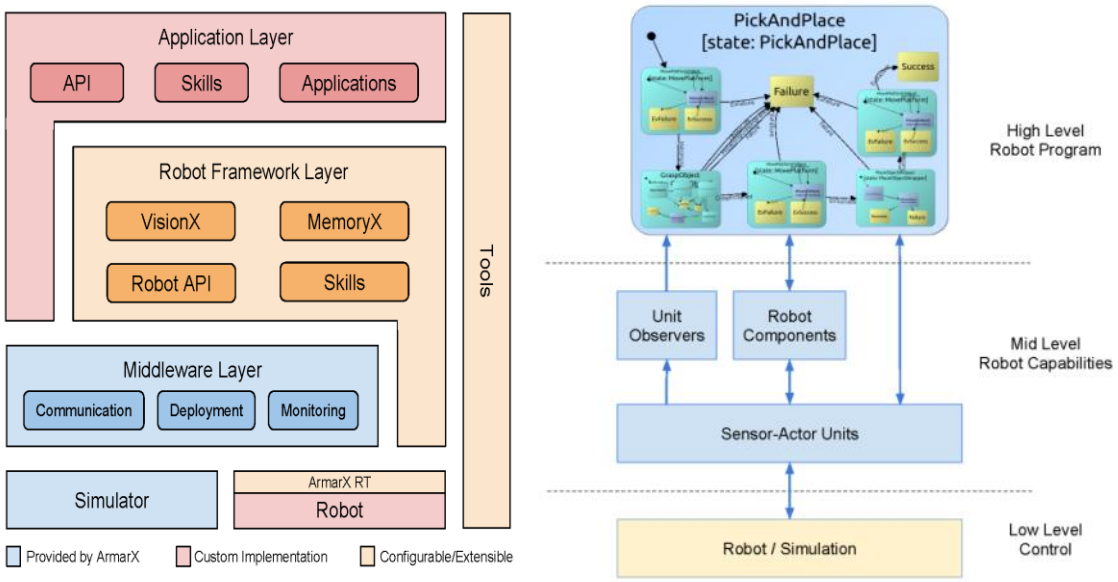
2 The ArmarX Software Framework

In cognitive systems, the representation of the perceived environment together with the agent's actual state information needs to be consistently stored in the internal memory. This requirement is demanded as an essential feature in the design of component-based robot software architectures. To explore such a vital feature, robot software architectures should provide comprehensive memory concepts allowing the incremental storage and fusion of a wide variety of multimodal sensory data, including low-level continuous sensory data as well as high-level discrete symbolic entities. These kinds of frameworks decouple the robot's high-level control system from the sensorimotor data processing and, thus, yield seamless operations on acquired data.

In this regard, ArmarX is a novel component-based robot software framework and has been recently developed at KIT in the context of the EU FP7 project

Xperience[1] to inherently support memory-based robot software architectures (see http://armarx.humanoids.kit.edu, [Vahrenkamp et al., 2015] and [Wächter et al., 2015]).

Figure 1(a) shows a conceptual view of the ArmarX robot software framework. The *Middleware Layer* consists of core capabilities required for implementing distributed applications and also has basic building blocks essential for robot software architecture operation. These building blocks are employed in the *Robot Framework Layer* which provides a generic robot Application Programming Interface (API) with more complex functionalities such as kinematics, memory, and perception. The here provided robot API modules can further be extended in order to implement robot specific APIs. Robot programs are implemented in the *Application Layer* as distributed applications by employing the generic and specific robot APIs. The ArmarX framework comes with a plugin-based Graphical User Interface (GUI) which can communicate with different components and visualize the produced data. The real time layer ArmarX RT provides a bridge that let specialized components interact with the ArmarX simulator or the robot hardware.



(a) **(b)**

Figure 1. The ArmarX robot software framework. (a) Conceptual view of ArmarX. (b) Three abstraction layers in ArmarX.

As illustrated in Figure 1(b), ArmarX has three abstraction layers: *low-*, *mid-*, and *high*-levels. These layers essentially cover the implementation of robot applications, extensions of robot APIs, and the realization of robot control algorithms. The *low-level layer* is abstracted through the Sensor-Actor Unit concept. This permits access to the hardware or the simulator to isolate the low level communication from higher levels of the robot software. The *mid-level layer*

[1] www.xperience.org

rather involves the implementation of robot capabilities such as perception, planning, and motion generation tasks in a network transparent fashion. At the *high-level layer*, a set of robot skills are designed as statecharts. These robot skills are mainly used for building complex robot programs as sequences of robot skills.

The ArmarX framework provides a memory structure, MemoryX consisting of prior knowledge unit, working (i.e. short-term) and long-term data representation units. Figure 2 depicts the overview of the MemoryX embedded in the Robot Framework Layer (see Figure 1(a)). The persistent memory supports long term memory structures and offers the possibility to store prior knowledge data. The robot's working memory (WM) is the central component and represents the current world state. The WM unit builds the actual representation of the world by continuous fusion of all sensor modalities. The WM is updated via an updater interface either by perceptual process or by prior knowledge that keeps known data such as models or features.

The existing memory units employ probabilistic reasoning concepts to ensure that the actual information about the certainty of perceived entities is available in the spatial and temporal domains. All memory units are organized in individually addressable segments containing arbitrary types or classes which are accessible within the distributed application. Based on these memories, robot skills are developed and encoded as event-driven and reusable statecharts. Appropriate interfaces allow attaching processes to the memory for updating and inference.

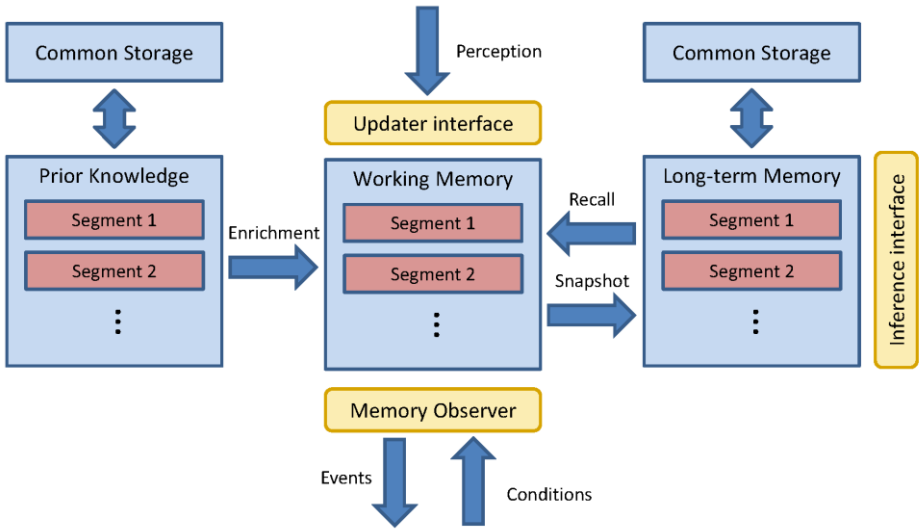


Figure 2. MemoryX: The memory structure in the ArmarX framework.

The ArmarX software packages are publicly available under https://gitlab.com/groups/ArmarX. Furthermore, KIT provides documentation with installation instructions, tutorials as well as frequently asked questions at http://armarx.humanoids.kit.edu.

In the following subsections, we will elaborate on some specific components required for generating actions in the robotic simulator.

2.1 The ArmarX Robot Skill Library

In ArmarX, robot skills are encoded as reusable generic statecharts. Here, each robot skill represents a unique robot behaviour, e.g. "*placing an object*" and is stored in the long-term memory. Existing robot skills can easily be parameterized and adapted to the actual robotic platform by configuring parameters in the ArmarX statechart editor. Figure 3 indicates a statechart editor for the *PlaceObject* skill. Here, each block defines one state and arrows represent transitions between state pairs. Run-time parameter settings can also be updated via configuration files which are directly accessible. This way, the same statechart can easily be altered from, for instance, "*placing a cup*" to "*placing a plate*". More details on the here briefly described statecharts can be found in [Wächter et al., 2015].

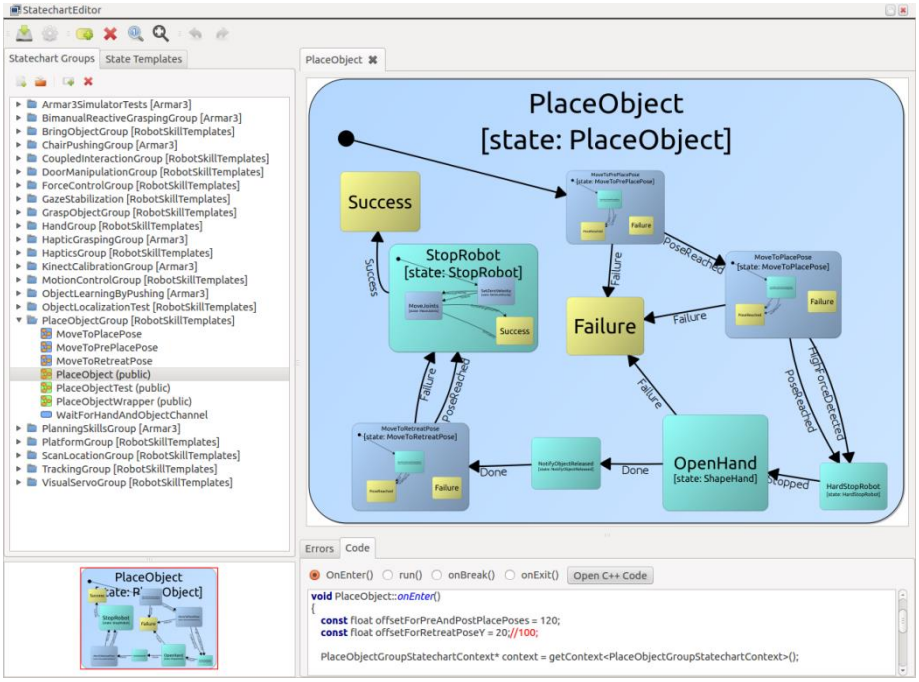


Figure 3. The ArmarX statechart editor.

In the context of the TimeStorm project, we define several perception and action skills in the simulation environment. These skills are then employed to implement sequence of various robot behaviours for different tasks, such as "*making cereal milk*". The following set of skills is currently available for the TimeStorm consortium and can directly be used in the simulation environment:

- *MoveJoints*: Move joints either in position or velocity control mode
- *MoveTCP*: Move the Tool Center Point (TCP) to a goal position
- *VisualServo*: Implementation of a position-based visual servo approach
- *MovePlatform*: Move a platform-based robot to a specific goal position

- *GraspObject*: Grasp an object with an end effector
- *BringObject*: Pick up an object and delivers it to a goal position
- *PlaceObject*: Put down a grasped object
- *PourObject*: Lift and tilt an object to pour
- *MixObject*: Move the end effector in a circular periodic format
- *ScanForObject*: Apply a scanning strategy to search for an object
- *TrackObject*: Try to track a given object
- *ViewSelection*: Change the view direction according to an automatic attention mechanism
- *Open/Close/Shape Hand*: Move a hand to specific shapes
- *StopRobot*: Stop all movements

2.2 The ArmarX Dynamic Simulator

The ArmarX dynamic simulator is essential for capturing the dynamics of the robot and its interaction with the environment. Figure 4 depicts the basic structure of the ArmarX simulation environment. The dynamic simulator follows a component-based approach and abstracts all communications with various units over well-defined interfaces. The initial version of the ArmarX Simulator has been extended to meet the requirements of the TimeStorm project. For this purpose, we first implemented an updated version of the hardware abstraction layer. This allows for transparent implementation of hardware interfaces through the ArmarX Sensor-Actor-Unit concept. We also decoupled the computation of the dynamic simulation unit from the visualization. This step lets the content of the observed world to be remotely displayed and, hence, yields an efficient simulation. In addition, we implemented localization interface providing consistent object and robot localization. The localization information is passed to the corresponding memory components of ArmarX.

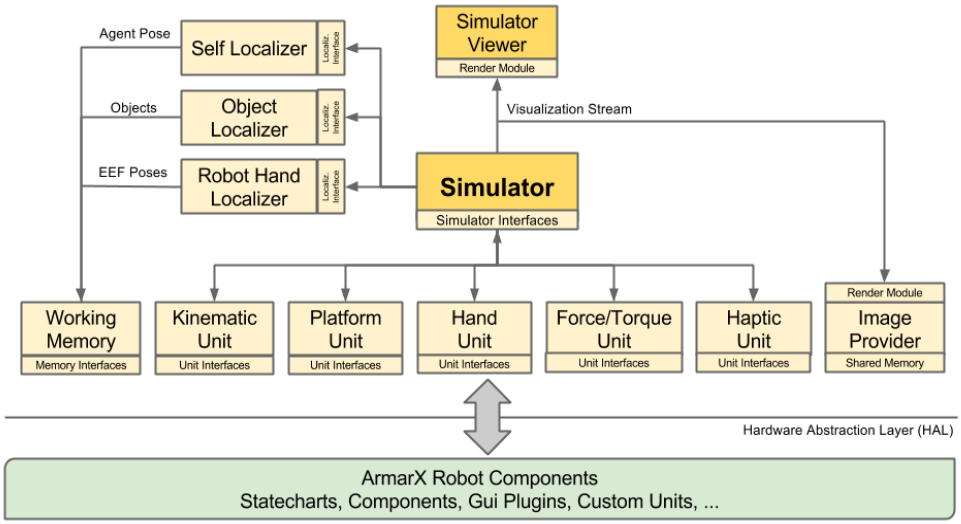


Figure 4. Structure of the ArmarX simulation environment.

2.3 The ArmarX Virtual Time Concept

In the context of the TimeStorm project, we implemented the concept of virtual time in ArmarX in order to influence the temporal duration required for each robot skill. The virtual time captures the elapsed time independent from the actual CPU time. It is also required to keep the execution of robot skills synchronized with the simulation. Calculating the properties of the dynamic simulation might take longer than executing the statecharts, thus leading to wrong calculations of control values.

The virtual time concept is based on a virtual clock sending out a heartbeat like messages. Each message advances the virtual time by a configurable time delta. For example, every time a clock message is received, the virtual clock advances by 10 milliseconds. The frequency for sending out these clock messages is configurable, too. It can either be synchronized with the system clock (tied to the CPU) or it can be slowed down or sped up by a user. This allows for the execution of robot actions to be faster or slower than regular. Additionally, it is possible to pause and resume the virtual clock and to perform single time steps.

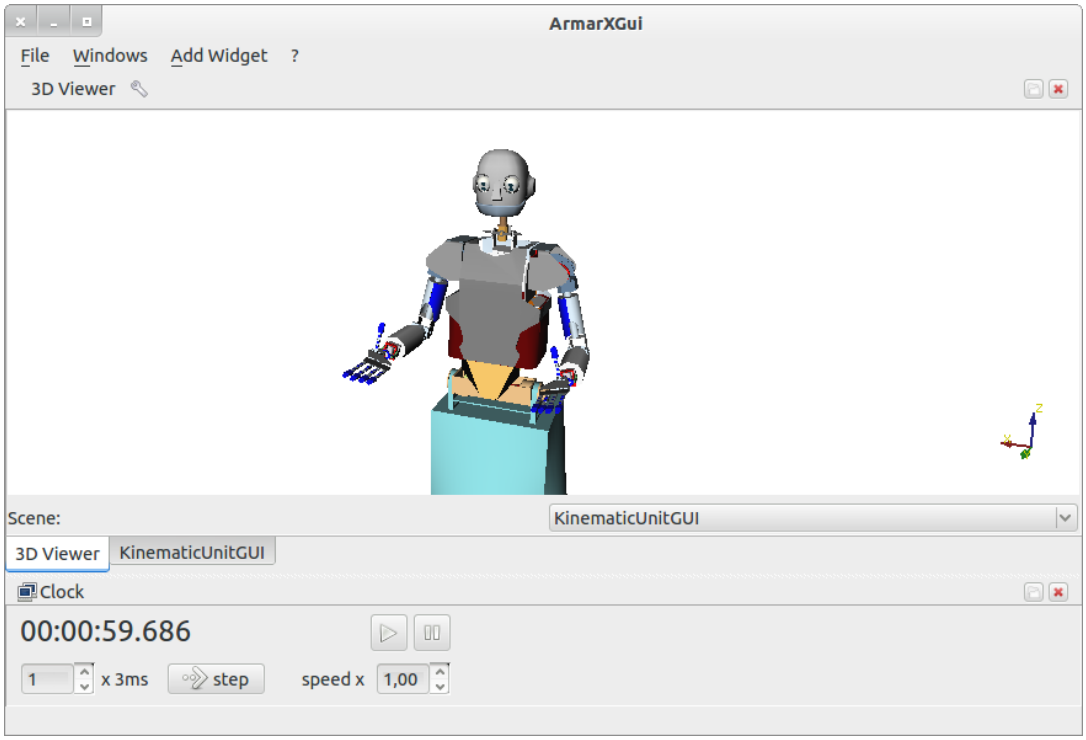


Figure 5. Snapshot of the ArmarX virtual time.

2.4 Scene, Object, and Humanoid Robot Models for TimeStorm

In the TimeStorm project, we aim e.g. at analyzing the temporal perception of human demonstrated actions and their execution with robots. In this regard, we simulated a kitchen environment in which the robot can perform various tasks, e.g. "*making cereal milk*", by manipulating different objects in different orders. Figure 6 shows a snapshot of a sample kitchen environment together with a set

of sample objects modeled in the ArmarX Simulator. KIT has a rich and publicly available 3D object model database [Kasper et al. 2012]. In TimeStorm related applications (see Section 3), we will use these scene and object models to accurately measure the temporal information during, for instance, the grasping process in the simulated environment.

KIT also provides two humanoid robots: Armar-3a [Asfour et al. 2006] and Armar-4 [Asfour et al. 2013]. Models of both robots are available in the ArmarX simulator. Armar-3a is a fully integrated autonomous humanoid robot with a mobile platform which allows for holonomic movability. Armar-3a has 43 DOFs and is equipped with position, velocity, and force-torque sensors. The next generation humanoid robot, i.e. Armar-4, is a full body torque controlled biped robot with 63 active degrees of freedom. Figure 7 illustrates both robots in the simulated environment.

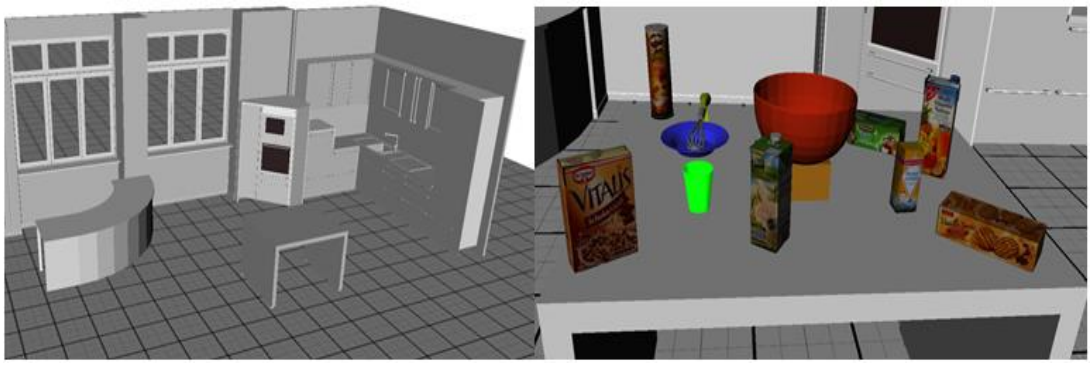


Figure 6. Snapshot of the kitchen environment (left) with a set of sample objects (right).

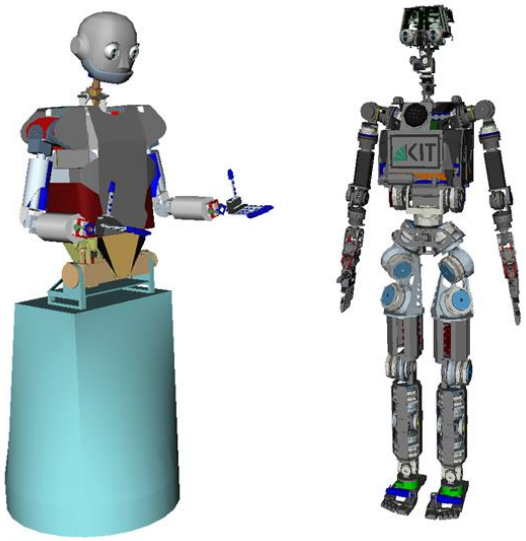


Figure 7. Armar-3a (left) and Armar-4 (right) humanoid robots.

3 Applications with the ArmarX Robotic Simulator

In this section, we provide two experimental setups that are designed for the TimeStorm project by using the ArmarX robot simulation environment. In the first setup, we aim at measuring the temporal perception of real and simulated manipulation actions performed by humans. This is a collaborative work between KIT and the University of Groningen (UoG). The second experiment covers the simulation of multi-agent collaborative task planning and execution with time constraints. This is a joint work between KIT and the Foundation for Research and Technology Hellas (FORTH).

3.1 Human Motion Simulation and Reproduction

The ArmarX simulator allows us to animate and reproduce observed human motion patterns with the Armar-4 humanoid robot. As also described in Deliverable 2.1., we introduced an experimental setup which investigates the extraction of motion primitives and their temporal lengths from a set of manipulation actions demonstrated by human subjects. For the experimental evaluation, we recorded several types of manipulations, such as “pick & place”, “push”, “mix”, “put in”, “put on”, “take down”, and “cut”, with a marker-based motion capture system VICON working at 100 Hz with the error of less than 1 mm. Each action type was demonstrated by a human subject at least 5 times by manipulating several objects (e.g. bowl, knife, whisk, sponge, bottle, etc.) at different speeds (fast, slow, and normal) in various scene contexts.

Captured human demonstrations were then encoded in Master Motor Map (MMM) framework, which provides a reference model of the human body and software tools for capturing representing, visualization and reproduction of whole-body human motion to humanoid robots with different morphologies (Terlemez et al., 2014). Given the MMM representation of the demonstrated action, we explored individual primitives (e.g. “approach”, “grasp”, “withdraw”, etc.) by considering both motion characteristics (such as trajectory and velocity profiles) of the hand and the semantic object-object and object-hand contact relations based on a hierarchical action segmentation methods introduced in (Wächter and Asfour, 2015). This segmentation process allows us to accurately estimate the duration and order of action primitives. Top row in Figure 8 depicts sample primitive frames extracted from a human demonstrated mixing action.

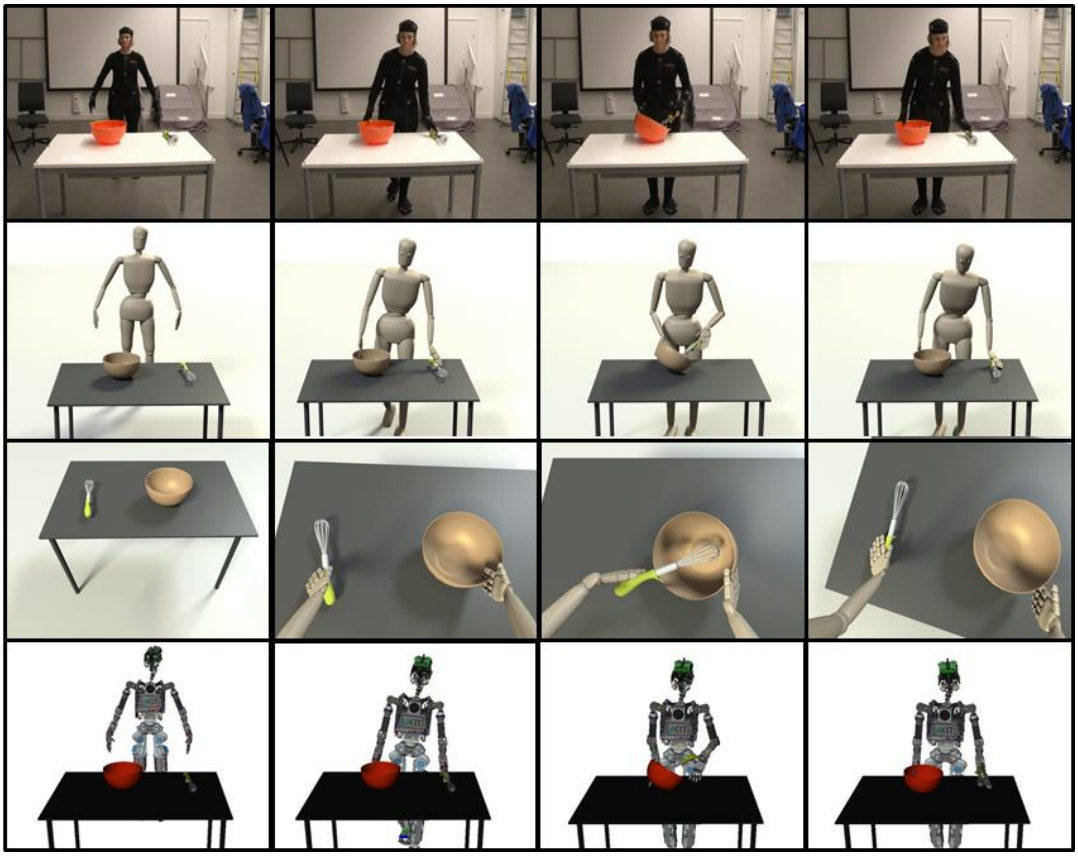


Figure 8. Human demonstrating manipulation actions (first row); Mapping of human demonstrations to the MMM in the first person viewpoint (second row) and third person viewpoint ((third row); Mapping to the demonstrations to the humanoid robot Armar-4 (fourth row)

In the ArmarX simulator, we can further reproduce the captured MMM motions of human demonstrations with an animated character from the first- and third-person viewpoints. Second and third rows in Figure 8 respectively show sample frames from the third- and first-person views of the animated mixing action. These animated actions are shown to humans to estimate the temporal length of each action. Our aim here is to provide a test-bed to investigate how time perception and motor timing in humans are influenced by the presence of biological movements in the real and animated visual scenes. We also intent to compare the temporal duration that humans perceive with that of our hierarchical segmentation method measures for each action. Note that we are currently collaborating with UoG to investigate these issues.

Furthermore, we can generate the motion profile of each primitive with our simulated robot Armar-4. Last row in Figure 8 indicates sample frames from the Armar-4 execution of the same mixing action. This reveals that with our framework we can transfer the motion profile of any observed action to different embodiments. This is a very important tool that can help us to understand the role of the embodiment in time perception and is subject to investigate in the future.

3.2 Multi-Agent Action Simulation

In the ArmarX simulator, we are also able to simultaneously simulate multi-agents for collaborative tasks, such as “preparing cereal milk”. In such tasks a higher level planner, implemented by FORTH, distributes different subtasks (such as bring, pour, mix) to each agent by considering agents’ skills with given time constraints. In this regard, we designed a scenario within the simulator which involves both of our humanoid robots, i.e. Armar-3a and Armar-4, in a simulated kitchen environment. In this scenario, we assume that both agents have different capabilities. For instance, Armar-3a is skilled at moving faster since it has a mobile platform, whereas Armar-4 can more efficiently perform the pouring and mixing actions due to its highly flexible kinematic design. In the simulation, each robot has its own working memory unit which leads to better cognitive reasoning. We are, on the other hand, synchronizing the working memories in order to achieve more accurate communication between agents. Figure 9 illustrates sample frames from a collaborative task in which Armar-3a is bringing a milk box while Armar-4 is placing a bowl on a table.

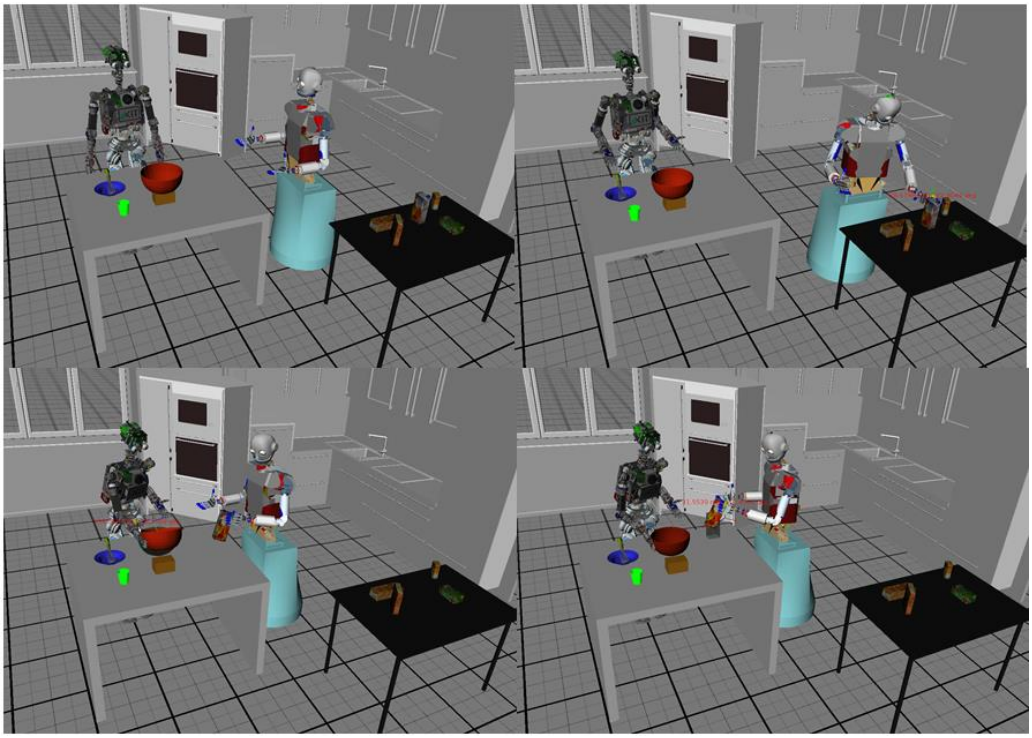


Figure 9. Multi-agent collaborative action execution in the ArmarX Simulator.

4 References

Asfour, T., Schill, J., Peters, H., Klas, C. , Bücker, J. , Sander, C., Schulz, S., Kargov, A., Werner, T. and Bartenbach, V. (2013), "ARMAR-4: A 63 DOF Torque Controlled Humanoid Robot", IEEE/RAS International Conference on Humanoid Robots.

Asfour, T, Regenstein, K., Azad, P., Schröder, J., Vahrenkamp, N., and Dillmann R. (2006), "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control", IEEE/RAS International Conference on Humanoid Robots

Kasper, A., Xue, Z., Dillmann, R. (2012) "The KIT object models web database: An object model database for object recognition, localization and manipulation in service robotics", The Inter. Journal of Robotics Research.

Terlemez, O., Ulbrich, S., Mandery, C., Do, M., Vahrenkamp, N., and Asfour, T. (2014), "Master Motor Map (MMM) - Framework and Toolkit for Capturing, Representing, and Reproducing Human Motion on Humanoid Robots", IEEE/RAS International Conference on Humanoid Robots.

Vahrenkamp, N., Wächter, M., Kröhnert, M., Welke, K. and Asfour, T. (2015) "The Robot Software Framework ArmarX", Information Technology, Vol. 57, No. 2, pp. 99 – 111.

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

Wächter, M., Ottenhaus, S., Kröhnert, M., Vahrenkamp, N., and Asfour, T. (2015) "The ArmarX Statechart Concept: Graphical Programming of Robot Behaviour", Frontiers in Robotics and AI (submitted).