EU FP7

AMARSi

Adaptive Modular Architectures for Rich Motor Skills

ICT-248311

Deliverable D.6.4

April 2014 (month 48)

Technical report on cognitive architecture admitting user interaction

Authors: Felix Reinhart (Bielefeld University), Arne Nordmann (Bielefeld University) and Jochen Steil (Bielefeld University)

Due date of deliverable	April 3, 2014
Actual submission date	April 3, 2014
Lead Partner	UniBi
Revision	Final
Dissemination level	Public

Contents

Related task from technical annex:

T.6.5 (UniBi (4PM)) Extension to user interaction and communication: This task prepares the final experimentation stage E 4. UniBi will provide constraints and modules derived from the Bielefeld robot architectures including that of iCub, which are centered at interaction and will allow to define interaction channels and modi for preparing the final architecture blueprint.

1 Introduction

UniBi developed modules for user interaction with the robots iCub and KUKA LWR. These modules cover different interaction channels, e.g. visual input [1, 2], gazing behaviors [3, 1], speech output [1, 2], tactile feedback [1, 2], and physical interaction modi [4, 5, 6]. For instance, AMARSi components for the visual detection of markers or colored blobs enable the intuitive interaction with robots like showing where to reach. Also, feedback from the robot to the user by a text-to-speech synthesis module enhances interaction, e.g. by conveying information about the robot's state. Tactile feedback from the user to the robot has been utilized for reward-based learning in [1, 2]. Concerning the physical interaction between humans and robots, the components developed in [4, 5, 6] enable intuitive programming of the KUKA LWR by enriching the active compliant control of the robot with motion constraints by means of previously taught redundancy resolutions of the inverse kinematics.
This deliverable provides an overview of the developed modules, their availability in the AMARSi component repository (`https://redmine.amarsi-project.eu/git/components.git`, accessible by the AMARSi partners), and their application. A short howto for handling the component repository can be found at `http://docs.cor-lab.de/cca-manual/trunk/html/comphowto.html`. Most of the components are implemented in the AMARSi software architecture. In applications with iCub, the software components utilize a feature of the middle-ware communication via RSB that enables the direct communication with YARP ports from the iCub software ecosystem (RSB YARP Transport). Most of the components are also integrated into AMARSi Domain-specific Language (DSL) by means of deployment descriptors. This enables the usage of interaction components when designing robotic experimentations in the DSL.

2 Overview

Table 1 gives an overview of interaction modules developed in AMARSi, their usage in demonstrations, software dependencies, and their integration into the AMARSi Domain-specific Language (DSL). Each module in Table 1 is presented by means of a short profile in the following sections. Dependencies are provided by hyperlinks to the respective software packages, where dependencies of those packages are not shown. Major dependencies are RSB [7], CCA [8], RCI [9], RSB YARP Transport [10], YARP [11], OpenCV [12], ICL [13], CBF [14] and `espeak` [15].

Module Name	**Demonstrations**	**Dependencies**	**DSL**	**Ref.**
CamGrabber	In examples	CCA, OpenCV	Yes	–
ImageViewer	In examples	CCA, OpenCV	Yes	–
BlobTracker	In examples	CCA, OpenCV	Yes	–
MarkerDetector	iCub: Classical and operant conditioning	CCA, OpenCV, ICL	Yes	[1, 2]
ESpeakServer	iCub: Classical and operant conditioning	CCA, `espeak`	Yes	[1, 2]
iCubSkinTouched	iCub: Classical and operant conditioning	CCA, RSB YARP Transport	Yes	[1, 2]
iCubGazing	iCub: Classical and operant conditioning	CCA, RSB YARP Transport	Yes	[1]
Random Head Movements	iCub: Classical and operant conditioning, Multi-modal interaction corpus	YARP	No	[1, 3]
Gravity Compensation	FlexIRob	CCA, RCI	Yes	[5]
Assisted Gravity Compensation	FlexIRob	CCA, RCI, CBF	Yes	[5, 6]
Collision Detection	FlexIRob	CCA, RCI	Yes	[4]

Table 1: Overview of interaction modules developed in AMARSi.

3 Modules for Computer Vision

CamGrabber <CCA::Node>	
Description:	This module grabs images from a camera and provides the image to other modules via its output port. The grabbed image is displayed in a window.
Inputs:	None.
Outputs:	out `rst::vision::Image`
Dependencies:	CCA, OpenCV
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

ImageViewer <CCA::Node>	
Description:	This module reads images from an input port and displays the image in a window.
Inputs:	inp_image `rst::vision::Image`
Outputs:	None.
Dependencies:	CCA, OpenCV
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

BlobTracker `<CCA::Node>`	
Description:	This module tracks colored blobs. Blob detection is based on a color filtering of the input image in HSV color space. Then, the resulting mask is postprocessed to calculate the Bary center of the detected color blobs. Color selection can be done interactively by a pointing device, or by configuring the component. The output contains the pixel coordinates of the Bary center relative to the center of the image. The module has a confidence threshold parameter $c_{thr} \in [0, 1]$, which prevents the module from sending pixel coordinates at the output port if blob detection is unreliable. Setting $c_{thr} = 0$ effectively turns off the reliability check.
Inputs:	inp_img `rst::vision::Image`
Outputs:	out_pxl `rci::Doubles`
Dependencies:	CCA, OpenCV
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

MarkerDetector <CCA::Node>	
Description:	This module receives an image and runs a marker detection algorithm. If any marker is detected, a list of marker indices, pixel positions, as well as the 3D estimate of the marker location is published at the output port. The module calls the very efficient and robust marker detection algorithms implemented in the ICL library. The module provides setter methods to specify marker parameters, as well as camera parameters for optimized 3D marker position detection. Results of the marker detection can be visualized with the additional module `MarkerDetectionViewer`. Parsing of the output containing detected marker indices and further properties is simplified by the `Marker` class contained in this software package.
Inputs:	inp_img `rst::vision::Image`
Outputs:	out_markers `rci::Doubles`
Dependencies:	CCA, ICL, OpenCV
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

4 Modules for Human-Robot Interaction

ESpeakServer <CCA::Node>	
Description:	This module receives string inputs and calls the text-to-speech synthesizer `espeak` to produce speech output. The voice of the speech synthesizer `espeak` can be configured when starting the module. The default voice configuration used by this module has been adopted to a young male robotic voice (iCub).
Inputs:	inp_text `std::string`
Outputs:	None.
Dependencies:	CCA, `espeak`
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

iCubSkinTouched <CCA::Node>	
Description:	This module detects whether a part of the artificial skin of iCub has been touched or not. Computation is based on a simple thresholding of the incoming skin readings $\mathbf{x}$. The scalar output o is computed according to $$o = \begin{cases} 1 & \text{if } \|\mathbf{x}\| > \theta \\ 0 & \text{otherwise} \end{cases},$$ where the threshold $\theta = 500$ by default.
Inputs:	inp_tactiles `rci::Doubles`
Outputs:	out_touched `rci::Doubles`

Dependencies:	CCA, RSB YARP Transport
Availability:	`https://redmine.amarsi-project.eu/git/components.git`
DSL Support:	DSL deployment descriptor

ICubGazing `<CCA::Node>`	
Description:	This module implements a proportional gain feedback controller for gazing at a target in pixel coordinates received at port in_target. The module controls pan and tilt of iCub's head joint angles in order to bring the target into the center of the image. In addition, the module compensates the roll of the head such that the head is stays upright. The module steers the head back into a home posture (straight forward viewing direction) if no targets are received. Furthermore, the module performs system calls to set iCub's facial expression according to target recognition status (smiling lips if target is tracked, neutral lips otherwise). Feedback of the current head configuration is read at port in_head. Outputs are provided in absolute joint angle positions by port out_head and as joint angle velocities by port out_vel. Gains for pan and tilt controllers, as well as for the roll compensation can be configured.
Inputs:	in_target `rci::Doubles` in_head `rci::JointAngles`
Outputs:	out_head `rci::JointAngles` out_vel `rci::JointVelocities`
Dependencies:	CCA, RSB YARP Transport
Availability:	`https://redmine.amarsi-project.eu/git/components.git`

DSL Support:	DSL deployment descriptor

RandomHeadMovements `<YARP Module>`	
Description:	This module performs randomized head movements on iCub. It utilizes the Modified VITE (vector integration to end-point, [16]) model for motion generation. MVITE is a motion primitive model formulated in terms of a second-order dynamical system. The MVITE primitive is used to control pan and tilt joint angles of iCub's head in order to achieve smooth, human-like head motions. The module generates randomized targets for the MVITE primitive, which gives iCub a natural looking background behavior. Randomized target generation can be turned on and off during runtime.
Inputs:	state:i
Outputs:	state:o
Dependencies:	YARP, iCub YARP interface
Availability:	Only UniBi.
DSL Support:	No.

5 Modules for Physical Human-Robot Interaction

Gravity Compensation <CCA::Component>	
Description:	Although gravity compensation is an internal control mode of the KUKA LWR control, we developed an additional gravity compensation. This mode allows finer tuning of interaction in terms of stiffness, damping and inertia [5].
Inputs:	in `rci::JointAngles`
Outputs:	out `rci::JointAngles`
Parameters:	`rci::JointStiffness`, `rci::JointDamping`, inertia
Dependencies:	CCA, RCI
Availability:	Only UniBi.
DSL Support:	DSL deployment descriptor

Assisted Gravity Compensation <CCA::Component>	
Description:	Hierarchical controller with primary task space controller and secondary joint space controller. Resulting in a gravity compensation that assists the user during interaction by supporting task space movements by resolving the redundancy according to the current environment. Redundancy resolution has to be provided by an additional component, e.g. by previous configuration in physical human robot interaction [5]. This control mode proved to facilitate easy Teach-In in constrained environments [6].

Inputs:	INPUT_CartCmd `rci::Pose` INPUT_JntMsr `rci::JointAngles` INPUT_JntCmd `rci::JointAngles`
Outputs:	out `rci::JointAngles`
Parameters:	3D/6D task, open/control loop
Dependencies:	CCA, RCI, CBF
Availability:	Only UniBi.
DSL Support:	DSL deployment descriptor

Collision Detection `<CCA::Component>`	
Description:	Collision detection component, monitoring joint torques or estimated joint torques of a robot limb, detecting and reporting a collision depending on a configurable torque threshold.
Inputs:	input_torques `rci::JointTorques`
Outputs:	output_collision `RobotCollision`
Parameters:	`rci::JointTorques` threshold
Dependencies:	CCA, RCI
Availability:	Only UniBi.
DSL Support:	DSL deployment descriptor

6 Conclusion

UniBi provided a large assortment of interaction modules which enhance the interaction capabilities of robotic experimentations conducted in the AMARSi project. Even though the AMARSi project is focussed on motion generation and explicitly excluded the development of complex perception components from the workplan, the integration of such components into the AMARSi software architecture has been accomplished successfully and thereby facilitates interaction through multiple channels. Most of the modules are available in the AMARSi Domain-specific Language, which supports the process of developing robotic experimentations with the AMARSi software toolchain.

References

[1] A. Soltoggio, A. Lemme, R.F. Reinhart, and J.J. Steil. Rare neural correlations implement robotic conditioning with delayed rewards and disturbances. *Frontiers in Neurorobotics*, 7(6), 2013.

[2] A. Soltoggio, R.F. Reinhart, A. Lemme, and J.J. Steil. Learning the rules of a game: Neural conditioning in human-robot interaction with delayed rewards. In *IEEE Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6, 2013.

[3] Judith Gaspers, Maximilian Panzner, Andre Lemme, Philipp Cimiano, Katharina J. Rohlfing, and Sebastian Wrede. A multi-modal corpus for evaluation of computational models for (grounded) language acquisition process. In *EACL Workshop on Cognitive Aspects of Computer Language Learning*, 2014. In press.

[4] Arne Nordmann, Christian Emmerich, Stefan Rüther, Andre Lemme, Sebastian Wrede, and Jochen J. Steil. Teaching Nullspace Constraints in Physical Human-Robot Interaction using Reservoir Computing. In *International Conference on Automation and Robotics*, 2012.

[5] C. Emmerich, A. Nordmann, A. Swadzba, J.J. Steil, and S. Wrede. Assisted gravity compensation to cope with the complexity of kinesthetic teaching on redundant robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4322–4328, 2013.

[6] Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen J Steil. A User Study on Kines-

thetic Teaching of Redundant Robots in Task and Configuration Space. *Journal of Human-Robot Interaction*, 2(Special Issue: HRI System Studies):56–81, 2013.

[7] Robotics Service Bus (RSB). `http://docs.cor-lab.org/rsb-manual/trunk/html/`. Version $\geq$ 0.9.

[8] Compliant Control Architecture (CCA). `https://code.cor-lab.org/projects/cca`. Version $\geq$ 0.4.

[9] Robot Control Interface (RCI). `http://docs.cor-lab.org/rci-manual/0.3/html/`. Version $\geq$ 0.4.

[10] YARP Transports with RSB. `http://docs.cor-lab.org/rsb-manual/trunk/html/specification-yarp.html`.

[11] Yet Another Robot Platform (YARP). `http://eris.liralab.it/yarp/`. Version 2.

[12] Open Source Computer Vision (Open CV). `http://opencv.org/`. Version 2.

[13] Image Component Library (ICL). `http://docs.cor-lab.de/icl-manual/trunk/html/index.html`. Version $\geq$ 8.2.

[14] Control Basis Framework (CBF). `https://github.com/norro/CBF`.

[15] eSpeak text to speech. `http://espeak.sourceforge.net/`.

[16] Biljana Petreska and Aude Billard. Movement curvature planning through force field internal models. *Biological Cybernetics*, 100:331–350, 2009. 10.1007/s00422-009-0300-2.