Echtzeit und Nicht-Echtzeit Integration von MATLAB und Simulink in das neue Robotik Software Framework aRDx des DLR

Berthold Bäuml

Autonomous Learning Robots Lab
Institute of Robotics and Mechatronics
German Aerospace Center (DLR)
berthold.baeuml@dlr.de
“Agile Justin” — An Advanced Robotic System

sensing
• stereo cameras (2MPixel/25Hz)
  RGB-D sensor (0.5MPixel/33Hz)
• torque sensor (all DOF, 1kHz)
  tactile skin on hands (3000taxel/750Hz)
• IMU (6D, 500Hz)

acting
• 53 DOF = 8 (platform) + 19 (torso) + 26 (hands)
• torque control over all DOF
  • 1kHz, <3ms latency, <100us Jitter

computing
• 4x Core i7 Quadcore (onboard)
• CPU cluster with 64 cores
• GPGPU cluster 16 NVidia K20
“Playing Balls”: Perception and Action at the Limits
“Mars Habitat”: Dexterous Mobile Manipulation
Planning and Object Recognition

Optimization Based Planning

Collision Objective based on Euclidian Distance Transform

initial
Tactile Skin

1500 Taxel/Hand, 2mm Resolution
Software Architecture for Advanced Robotic Systems

• communicating components central abstraction in robotic frameworks
• packet/message based communication
• dynamic connecting and disconnecting of components
• concurrent, parallel execution
• distributed over network of computing resources
• components decoupled
  • as modules for team development
  • as process for robustness at runtime
• popular examples: **ROS**, YARP, OROCOS
Software Architecture for Advanced Robotic Systems

big challenge: **wide range of software domains**

<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>microcontroller/FPGA</td>
<td>realtime PC (QNX)</td>
<td>CPU/GPGPU cluster (Linux)</td>
<td>cluster, internet server cloud</td>
</tr>
<tr>
<td>communication</td>
<td>hardware bus (SPI, I2C, …)</td>
<td>hard realtime, distributed, QoS, up to 1kHz, 5MB</td>
<td>“optimal” transport, distributed, QoS, ~10Hz, up to 1GB</td>
<td>“fast” transport, distributed &lt;10Hz, &lt;10MB</td>
</tr>
<tr>
<td>language</td>
<td>hard realtime, small footprint, HDL</td>
<td>hard realtime, efficient, parallel</td>
<td>efficient on large data, parallel, OOP</td>
<td>parallel, high level (functional, declarative, …)</td>
</tr>
</tbody>
</table>
ROS (Robot Operating System, www.ros.org)

- quasi-standard in robotics
- communication
  - no hard realtime
  - no QoS
  - not optimal (e.g., copy-once for inter-process)
  - nested structs and dynamic 1D arrays
  - but no multidimensional arrays and beyond (e.g., recursive types)
- languages
  - C/C++ for low level, planning/modelling
  - Python as “high level” language
    - not compiled, no parallel execution
    - -> bad performance (50x slower than C)
    - -> lot of higher level stuff has to implemented in C/C++
  - bindings to high level languages (diverse Lisps and Prolog, …)
<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>microcontroller/FPGA</td>
<td>realtime PC (QNX)</td>
<td>CPU/GPGPU cluster (Linux)</td>
<td>cluster, internet server cloud</td>
</tr>
<tr>
<td>communica-</td>
<td>hardware bus (SPI, I2C, …)</td>
<td>hard realtime, distributed, QoS, up to 1kHz, 5MB</td>
<td>“optimal” transport, distributed, QoS, ~10Hz, up to 1GB</td>
<td>“fast” transport, distributed &lt;10Hz, &lt;10MB</td>
</tr>
<tr>
<td>language</td>
<td>hard realtime, small footprint, HDL</td>
<td>hard realtime, efficient, parallel</td>
<td>efficient on large data, parallel, OOP</td>
<td>parallel, high level (functional, declarative, …)</td>
</tr>
</tbody>
</table>
aRDx (Agile Robot Development “Next Generation”)

- communication
  - no single stack can fulfill all demands -> two stacks!
    - highly performant hard real-time stack
      - minimal latency and jitter, detailed QoS
      - optimal transport (zero-copy for IPC, copy-once to each host)
    - nested structs and static multidimensional arrays
  - high level directly by modern programming language
    - automatic serialization of
      - arbitrary complex (recursive data types)
      - code (sending closures)
    - (almost) optimal transport (copy-once to each VM on each host)

- language
  - C/C++ for low level
  - Racket (http://racket-lang.org): modern language from Lisp family
    - functional, compiled, parallel (to some extent)
    - only 5x slower than C (compared to Python’s 50x slower than C)
    - “programmable programming language” -> OOP, DSLs, Prolog, …
A Comparison of the Raw Communication
Performance of Robotic Frameworks

We presented the design considerations and implementa-
...
### Domain
- **low level driver, joint controller**
- **robot controller, sensor preprocessing**
- **world modeling, path planning**
- **“AI” (logic planner, cognitive model)**

### Computer
- **microcontroller/FPGA**
- **realtime PC (QNX)**
- **CPU/GPGPU cluster (Linux)**
- **cluster, internet server cloud**

### Communication
- hardware bus (SPI, I2C, …)
- hard realtime, distributed, QoS, up to 1kHz, 5MB
- “optimal” transport, distributed, QoS, ~10Hz, up to 1GB
- “fast” transport, distributed <10Hz, <10MB
- bits/bytes & simple structs
- nested static structs & arrays
- nested static structs & dynamic arrays
- flexible, recursive data types & program snips

### Language
- hard realtime, small footprint, HDL
- hard realtime, efficient, parallel
- efficient on large data, parallel, OOP
- parallel, high level (functional, declarative, …)
MathWorks Toolchain

has much to offer for wide range of domains!

- model-based design of complex controllers
- automatic realtime (!) code generation -> HIL (Hardware-In-the-Loop)
- signal and image processing
- thorough numerics
- data analysis & visualization
- simulation of complex mechatronic systems
- ...

<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>system modeling, data analysis</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulink,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink Coder -&gt; HIL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDL Coder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stateflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control System Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Processing Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Processing Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Vision System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbolic Math Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SimMechanics, Sim…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MathWorks Toolchain has much to offer for a wide range of domains!

- Model-based design of complex controllers
- Automatic realtime (!) code generation -> HIL (Hardware-In-the-Loop)
- Signal and image processing
- Thorough numerics
- Data analysis & visualization
- Simulation of complex mechatronic systems
- ...

<table>
<thead>
<tr>
<th>Domain</th>
<th>Low level driver, joint controller</th>
<th>Robot controller, sensor preprocessing</th>
<th>System modeling, data analysis</th>
<th>World modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulink,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink Coder -&gt; HIL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDL Coder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stateflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control System Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Processing Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Processing Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Vision System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbolic Math Toolbox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SimMechanics, Sim…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• problem: How to bring to robotic system with >50DOF & distributed computation?
• -> binding to aRDx

<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>system modeling, data analysis</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>microcontroller/FPGA</td>
<td>realtime PC (QNX)</td>
<td>workstation (Linux)</td>
<td>CPU/GPGPU cluster (Linux)</td>
<td>cluster, internet server cloud</td>
</tr>
<tr>
<td>communication</td>
<td>hardware bus (SPI, I2C, …)</td>
<td>hard realtime, distributed, QoS, up to 1kHz, 5MB</td>
<td>data logging</td>
<td>“optimal” transport, distributed, QoS, ~10Hz, up to 1GB</td>
<td>“fast” transport, distributed &lt;10Hz, &lt;10MB</td>
</tr>
<tr>
<td></td>
<td>bits/bytes &amp; simple structs</td>
<td>nested static structs &amp; arrays</td>
<td>numerical data (structs and arrays)</td>
<td>nested static structs &amp; dynamic arrays</td>
<td>flexible, recursive data types &amp; program snips</td>
</tr>
<tr>
<td>language</td>
<td>hard realtime, small footprint, HDL</td>
<td>hard realtime, efficient, parallel</td>
<td>scripting, numerics &amp; computer algebra</td>
<td>efficient on large data, parallel, OOP</td>
<td>parallel, high level (functional, declarative, …)</td>
</tr>
</tbody>
</table>
Simulink - aRDx Binding

features

- automatic S-Function block generation
- packets with 1D-arrays of basic types
  (future: multi-arrays and nested structs)
- supports interpreted Simulink and generated code on realtime target
- channel specification in S-Function parameter

robot-sfun.rkt

```racket
#lang racket/base
(require generator/simulink)
(require robot-packet)

(ardx-simulink-bridge
 #:ardx->simulink robot_state_packet
 #:simulink->ardx robot_control_packet)
```

robot-packets.rkt

```racket
#lang generator/idesc

(define-gstruct robot_state_packet
  ([status gint]
   [angles (garray gdouble 19)]
   [torques (garray gdouble 19)]))

(define-gstruct robot_control_packet
  ([torques (garray double 19)])
```

Simulink Coder

realtime executable
• problem: How to bring to robotic system with >50DOF & distributed computation?
• -> binding to aRDx

<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>system modeling, data analysis</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>microcontroller/FPGA</td>
<td>realtime PC (QNX)</td>
<td>workstation (Linux)</td>
<td>CPU/GPGPU cluster (Linux)</td>
<td>cluster, internet server cloud</td>
</tr>
<tr>
<td>communication</td>
<td>hardware bus (SPI, I2C, …)</td>
<td>hard realtime, distributed, QoS, up to 1kHz, 5MB</td>
<td>data logging</td>
<td>“optimal” transport, distributed, QoS, ~10Hz, up to 1GB</td>
<td>“fast” transport, distributed &lt;10Hz, &lt;10MB</td>
</tr>
<tr>
<td>language</td>
<td>hard realtime, small footprint, HDL</td>
<td>hard realtime, efficient, parallel</td>
<td>scripting, numerics &amp; computer algebra</td>
<td>efficient on large data, parallel, OOP</td>
<td>parallel, high level (functional, declarative, …)</td>
</tr>
</tbody>
</table>
Matlab - Racket Binding

import racket.*
require('my-module')

v = to_racket([1 2 3 4], 'vector')
sv = racket('scale', 0.2, v)
robots_id = racket('make-robots', 5, sv)
robots = from_racket(robots_id)
q = s(3).arm.joint_angle(2)

(define (scale s v)
  (vector-map (lambda (e) (* e s)) v))

(define (make-robots num v)
  (for/vector ([i num])
    (dot (Robot) 'arm 'joint-angle v)))

features
- require (load) any Racket module
- obeys Matlab’s lexical scope
- call any racket function or macro
- transparent conversion of basic types
- references to any Racket objects
- explicit conversion to/from Racket for arbitrarily nested arrays, structs

implementation
- Racket process for each Matlab process
- command/result (basic) by aRDx channels
- complex Matlab data by MAT-Files
• **problem**: How to bring to robotic system with >50DOF & distributed computation?

• -> binding to aRDx

<table>
<thead>
<tr>
<th>domain</th>
<th>low level driver, joint controller</th>
<th>robot controller, sensor preprocessing</th>
<th>system modeling, data analysis</th>
<th>world modeling, path planning</th>
<th>“AI” (logic planner, cognitive model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>microcontroller/FPGA</td>
<td>realtime PC (QNX)</td>
<td>workstation (Linux)</td>
<td>CPU/GPGPU cluster (Linux)</td>
<td>cluster, internet server cloud</td>
</tr>
<tr>
<td>communication</td>
<td>hardware bus (SPI, I2C, …)</td>
<td>hard realtime, distributed, QoS, up to 1kHz, 5MB</td>
<td>data logging</td>
<td>“optimal” transport, distributed, QoS, ~10Hz, up to 1GB</td>
<td>“fast” transport, distributed &lt;10Hz, &lt;10MB</td>
</tr>
<tr>
<td>language</td>
<td>hard realtime, small footprint, HDL</td>
<td>hard realtime, efficient, parallel</td>
<td>scripting, numerics &amp; computer algebra</td>
<td>efficient on large data, parallel, OOP</td>
<td>parallel, high level (functional, declarative, …)</td>
</tr>
</tbody>
</table>
Example: Automatic Calibration of a Multisensorial Upper Body of a Humanoid Robot

5 min “Push-a-Button” Calibration

L,R: Stereo Cameras
K: RGB-D Camera
I: IMU
W: Wrist with Marker
Example: Automatic Calibration of a Multisensorial Upper Body of a Humanoid Robot

- Simulink model for HIL control of full humanoid
- Matlab script for flow control of the calibration
  - during recording (commanding robot movements, start/stop of recorders, ..)
  - batch processing of recorded data
- Matlab for processing data and optimization
  - Image Processing Toolbox (marker detection)
  - Signal Processing Toolbox (noise filtering)
  - Matlab based MTK - Manifold ToolKit (openslam.org/MTK.html) (model fit)
- Optimization Toolbox (finding optimal robot configurations)

-> whole application can be implemented in Matlab/Simulink universe

- fast development
- dramatically less error-prone than, e.g., C/C++
- numerically rock-solid
Conclusions

- advanced robotic systems span **wide range of software domains**
  - “from hardware driver to artificial intelligence”
- DLR’s new **aRDx software framework** tackles this with
  - two communication stacks: static & realtime — flexible & fast
  - Racket as language
- MathWorks Toolchain could cover large part of domains
- **tight binding of MathWorks Toolchain to aRDx** allows use on advanced robotics systems (>50DOF, distributed computing, …)
  - Simulink & Coder to aRDx: hard realtime
  - Matlab to Racket: flexible, all functionality accessible
- **MathWorks Toolchain and aRDx perfect fit** esp. in research:
  - rapid prototyping
  - even students can work with complex robots because they can often stay completely in the MathWorks universe (known from their studies)
"It shows off ... capabilities of what's arguably ... the most, capable dual-armed mobile humanoid robots in existence." (IEEE Spectrum 2014)

Best Video Award
ICRA 2014

“aRDx/Racket & MathWorks Tools inside”