Software development environment for HRP series

Fumio Kanehiro (AIST)
Outline

1. Overview
2. Software platforms
   a. RT-Middleware
   b. Choreonoid
3. Continuous integration using dynamics simulation
Layer 1: Joint Level Control

Layer 2: Whole Body Level Control

Shared Memory

Layer 3: Perception and Planning

Layer 4: User Interface

External computers

Onboard computers

TCP/IP

Physical location and communication between layers
Robot Hardware

Whole-body motion controller

Impedance Control
Balance Control
Prioritized Inverse Kinematics Solver

Standing up motion generator

Impedance Control

Object detection

Robot Hardware

Simulator (Choreonoid)

Real-time OS (RT-Preempt Linux)

User Interface Layer (sporadic)

Graphical User Interface

Walk destination

Non-real Time Layer1

(period=100ms)

Motion Planners

Real-time Layer1

(joint level, period=1ms)

Sensor

Joint position

Joint Servo

User Interface (Choreonoid)

Reaching Motion Planner

Footstep Planner

Posture Generator

Standing position Planner

Standing up motion generator

3D Laser Scanner

Statistical Outlier Removal Filter

Moving Least Square Filter

Resize Image

Undistort Image

Image Grabber

Environment mapping

Object detection

Occupancy 3D Grid Map

Object Detector

Landing State Estimator

Planned footsteps whole-body motions

Current robot state

Real Time Layer2

(whole body level, period=2ms)

Middleware (RT-Middleware)

Processed images

Processed point cloud

Object motion

Real-time OS (RT-Preempt Linux)

Joint servos

Robot Hardware

Graphical User Interface

Data Logger

Middleware (RT-Middleware)

User Interface (Choreonoid)
RT-Middleware [Ando IROS05]

- http://www.openrtm.org
- RT = Robot Technology
- A software platform to develop RT system as a network of software components (RT-component, RTC)
- OpenRTM-aist is one of implementations
- RT components can be deployed on a computer network
RT-Component Architecture

Activity, Execution context

Data Port
- Data centric communication
- Continuous data transfer
- Dynamic connection/disconnection

Data centric communication

Service Port
- User defined interface
- Access to detailed functionality of RTC
  - Getting/setting parameters
  - Changing modes
  - etc…

Service oriented interaction

Configuration
- Function for internal parameter
- Multiple parameter sets
- They can be changed from remote in run-time

RTC can have several configuration sets. Runtime reconfiguration and dynamic switching are supported.
Component execution in RTM/ROS

- **RTC**
  - Business logic
  - Non-real-time EC (Execution Context)
    - Execute by time
  - Real-time EC
    - Execute in real-time
  - Ext. trigger EC
    - Execute by external trigger
  - EC can be attached/ detached in runtime

- **ROS**
  - Business logic
  - Non-real-time thread
    - Execute by time
  - Real-time thread
    - Execute by real-time
  - Ext. trigger thread
    - Execute by external trigger

Logic execution type is fixed in compile time
Combination of execution contexts and RTCs

One EC and one RTC (default)

One EC and multiple RTCs
Sequential execution of RTCs ex) image processing

Multiple ECs and one RTC
Parallel execution using shared data ex) short cycle control and long cycle visualization
Real-time/composite execution

- **RTC architecture** realizes composition, real-time execution for multiple RTCs
  - Execution and logic are separated, and various execution type can be realized

- **ROS:** 1-node = 1-process
  - Sequential execution, close coupled composition are impossible
  - Some tools such as ros_control, realtime-tools can supports such requirement
  - However, node must be designed different way from normal ROS node
New communication features

• Topic connection
  – DDS, ROS like connection scheme
  – Topics are registered and matched on naming servers

• Direct connection
  – OutPort directly write into InPort’s variable
  – Two RTCs must be in a same process
  – Thread-safe implementation. Execution context isn't necessarily shared RTCs

• Shared memory connection
  – Same node, but different process/language RTCs can communicate.
  – Marshalled data are stored/read into/from shared memory area.
Other features

- **Master-slave manager**
  - Master: Frontend process to application, slave management
  - Slave: It actually hosts RTCs.

- **Secure communication (SSL)**
  - CORBA’s SSL features are used

- **DDS port implementation** will be included
  - ROS2 compatibility might be realized
Choreonoid

Choreonoid is an extensible framework for robot applications.

- www.choreonoid.org
- Windows and Linux are supported
- Open source software (MIT license)
- Basic functions to handle robot models are included
- Dynamics simulator is embedded
- Users can extend by developing/adding plugins
- Lightweight and efficient single process architecture
Use cases of Choreonoid

1. Robot world simulator
   The official simulator of JVRC
   (Japan Virtual Robotics Challenge)

2. Teleoperation interface
   User interface for supervised autonomy used at DRC Finals

3. Robot choreographer
   CG software-like interface and automatic balance compensation
Choreonoid as a simulator

- Joints
  - Free, fixed, rotate, slide
- Sensors
  - Force/torque sensor, gyrometer, accelerometer, camera, RGBD camera, range finder
- Shape description
  - VRML97, COLLADA, STL
- Middleware
  - RTM, ROS
- Physics engines
  - AIST, ODE, PhysX, AgX, Bullet
- Not implemented
  - Deformable objects, cable, aerial robots, radio wave, sound, …
Choreonoid as a User Interface

- Markers
  - Walk destination marker
  - Body part marker
  - Manipulation marker
  - Measurement marker

- Task sequence system
  - Task description by Python
Example: turning a valve

Robot’s views          Main view          Simulation view

Task sequence view

<table>
<thead>
<tr>
<th>Valve</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>キャンセル</td>
<td>0 / 42</td>
<td>&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rotating valves</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Continuous Integration using dynamics simulation

- Jenkins: Continuous integration tool
- GitHub: Version control
- Developer
- Update
- Log and screenshot
- Movie
- E-mail
- Build test
- Task execution test
- Web page
- Polling
- Ubuntu
- Debian
- Netlify
- YouTube
Summary page of test results

Jenkins CI report

Last update: 2016/05/24 17:04:51

Job Summary

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Latest Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>build-debian7-32</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>build-ubuntu1404-64</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>build-ubuntu1604-64</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>task-hrp2kai-balancebeam</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>task-hrp2kai-button</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>task-hrp2kai-door</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>task-hrp2kai-terrain</td>
<td>build</td>
<td>passing</td>
</tr>
<tr>
<td>task-hrp2kai-valve</td>
<td>build</td>
<td>running</td>
</tr>
<tr>
<td>task-hrp2kai-wall</td>
<td>build</td>
<td>passing</td>
</tr>
</tbody>
</table>

Latest 10 test results

Link
History page of test results

Task: hrp2kai-valve-building2-6

Build Stability: 36%

Build History:

<table>
<thead>
<tr>
<th>#</th>
<th>Status</th>
<th>Time</th>
<th>Duration</th>
<th>Slave</th>
<th>Inspection</th>
<th>Test</th>
<th>Coverage</th>
<th>Changes</th>
<th>Logs</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2456</td>
<td>SUCCESS</td>
<td>2016/11/01 15:57</td>
<td>14 min.</td>
<td>slave8(Ubuntu 16.04.1 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>2307744KB used 1039004KB change</td>
</tr>
<tr>
<td>2455</td>
<td>SUCCESS</td>
<td>2016/11/01 14:57</td>
<td>14 min.</td>
<td>slave8(Ubuntu 16.04.1 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>2318820KB used 1027500KB change</td>
</tr>
<tr>
<td>2454</td>
<td>SUCCESS</td>
<td>2016/11/01 13:57</td>
<td>17 min.</td>
<td>slave8(Ubuntu 16.04.1 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>2322916KB used 1009376KB change</td>
</tr>
<tr>
<td>2453</td>
<td>SUCCESS</td>
<td>2016/11/01 12:57</td>
<td>21 min.</td>
<td>slave3(Ubuntu 14.04.5 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>1987244KB used 927176KB change</td>
</tr>
<tr>
<td>2452</td>
<td>SUCCESS</td>
<td>2016/11/01 11:57</td>
<td>29 min.</td>
<td>slave3(Ubuntu 14.04.5 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>1995596KB used 916359KB change</td>
</tr>
<tr>
<td>2451</td>
<td>SUCCESS</td>
<td>2016/11/01 10:57</td>
<td>21 min.</td>
<td>slave3(Ubuntu 14.04.5 LTS)</td>
<td>0 err.</td>
<td></td>
<td></td>
<td></td>
<td>console log task.png task.log</td>
<td>1991880KB used 935458KB change</td>
</tr>
</tbody>
</table>

Link to a build log and a screenshot on Google Drive

Link to GitHub pages

Link to a movie on YouTube
Links

• Choreonoid
  http://www.choreonoid.org

• OpenRTM-aist
  http://www.openrtm.org