A Framework for Advanced Robot Programming in the RoboCup Domain

Hayato Kobayashi\textsuperscript{1}, Akira Ishino\textsuperscript{2}, and Ayumi Shinohara\textsuperscript{3}

\textsuperscript{1}Department of Informatics, Kyushu University, Japan
\textsuperscript{2}Office for Information of University Evaluation, Kyushu University, Japan
\textsuperscript{3}Graduate School of Information Science, Tohoku University, Japan
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Outline

- **Background**
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
RoboCup Soccer

Team Jolly Pochie [dʒɔliːˈpɒtʃiː]

Small size robot league

Middle size robot league

Four-legged robot league

Simulation league

Humanoid league

https://www.robocup.org/
Many difficulties

- Soccer programs is complex
- Full compiling takes more than 10 minutes
- Booting of AIBO takes about 30 seconds
- Debug via wireless LAN
- Batteries only last about 30 minutes
- Team development can cause conflicts
- Cute shape is not suited for playing soccer
- AIBOs can faint because of motor load
- AIBOs can break their legs
How to solve?

- Hardware problems
  - We can’t solve!

- Software problems
  - Maybe, we can solve!!!
How to solve?

- Hardware problems
  - We can’t solve!

- Software problems
  - We want a “framework” that makes it easy to create robot programs
  - Maybe, we can solve!!!
Outline

- Background
- **Related work**
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Related Work

- **Tekkotsu**
  - A development framework for AIBO
  - Created at Carnegie Mellon University
  - Consists of C++ Libraries wrapping OPEN-R.
  
  (We must use OPEN-R for creating AIBO programs)

http://www.tekkotsu.org/
Related Work in RoboCup Symposium


Outline

- Background
- Related work
- **Proposed framework**
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Proposed Framework

- Two techniques are integrated
  - **Plug-in system** (easy to extend)
    - Effective for team development
    - No need to know the whole system
  - **Scripting language** (easy to use)
    - Effective for creating strategic programs
    - No need to recompile and reboot
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Concept of our framework

**Plug-in system**

OVirtualRobot can control actuators and sensors of AIBO

OVirtualRobot (OPEN-R Object)
Concept of our framework

Plug-in system

Strategic scripts
Concept of our framework

For Constructing a player program

- ball recognition
- beacon recognition
- shoot motion
- self-localization
- quadrupedal locomotion

Strategic scripts

Plug-in system
Concept of our framework

Strategic scripts

Easy to plug

Plug-in system
Concept of our framework

Strategic scripts

Scripts access these modules

Plug-in system
Concept of our framework

- Strategic scripts
- Scripts access these modules
- Plug-in system
- Sound effect
Concept of our framework

- Strategic scripts
- Scripts access these modules
- Easy to extend
- Plug-in system

Easy to extend
Concept of our framework

Strategic scripts

Plug-in system

A better beacon recognition module

Ver. 2
Concept of our framework

Strategic scripts

Ver. 2

Easy to replace

Plug-in system

Without any changes
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - **Plug-in system**
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Plug-in system

- Plug-in system has often been used in recent applications

Firefox
https://www.mozilla.org/
Web browser

eclipse
https://www.eclipse.org/
IDE

Adobe Illustrator
https://www.adobe.com/
Drawing software

We don’t need to know the whole system
Concept of Plug-in system

Plug-in system

OVirtualRobot
Concept of Plug-in system

- A recognition module
- A team-play module
- A touch sensor module
- A locomotion module

Plug-in system

OVirtualRobot
Concept of Plug-in system

- A recognition module
- A team-play module
- A touch sensor module
- A locomotion module

Plug-in system

OVirtualRobot

Camera event
Sensor event
UDP event
Action event
Abstract modules

The design of our plug-in system

JPObject manages the instance of created modules
### Abstract modules

We can override `cameraNotify()` and describe image processing.

<table>
<thead>
<tr>
<th>Abstract Module</th>
<th>Special method</th>
<th>When is the method called?</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPCameraModule</td>
<td>cameraNotify()</td>
<td>Every 40 ms in sync with the CCD-camera</td>
</tr>
<tr>
<td>JPMindModule</td>
<td>mindNotify()</td>
<td>The same as <code>cameraNotify()</code></td>
</tr>
<tr>
<td>JPActionModule</td>
<td>actionReady()</td>
<td>When a set of joint angles are achieved</td>
</tr>
<tr>
<td>JPSensorModule</td>
<td>sensorNotify()</td>
<td>When sensor data is detected</td>
</tr>
<tr>
<td>JPUDPModule</td>
<td>udpNotify()</td>
<td>When UDP data is received</td>
</tr>
<tr>
<td>JPTCPModule</td>
<td>tcpNotify()</td>
<td>When TCP data is received</td>
</tr>
<tr>
<td>JPModule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abstract Module</td>
<td>Special method</td>
<td>When is the method called?</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>JPCameraModule</td>
<td>cameraNotify()</td>
<td>Every 40 ms in sync with the CCD</td>
</tr>
<tr>
<td>JPMindModule</td>
<td>mindNotify()</td>
<td>The same as cameraNotify()</td>
</tr>
<tr>
<td>JPActionModule</td>
<td>actionReady()</td>
<td>When a set of joint angles are achieved</td>
</tr>
<tr>
<td>JPSensorModule</td>
<td>sensorNotify()</td>
<td>When sensor data is detected</td>
</tr>
<tr>
<td>JPUDPModule</td>
<td>udpNotify()</td>
<td>When UDP data is received</td>
</tr>
<tr>
<td>JPTCPModule</td>
<td>tcpNotify()</td>
<td>When TCP data is received</td>
</tr>
</tbody>
</table>

recognized modules

strategy modules

locomotion modules

localization modules, etc
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Lua  (http://www.lua.org/)

- Designed for embedding into C/C++
  - Easy to embed into C/C++
  - Faster than Python
  - Uses less memory than Python
  - Has a simple and powerful syntax
  - Smaller footprint than Python (about 1/10)

By  http://lua-users.org/wiki/LuaVersusPython
A simple example

- Returns the summation of arguments

```plaintext
function sum(…)  
  local s = 0  
  for i=1, arg.n do  
    s = s + arg[i]  
  end  
  return s  
end
```
Luabind
(http://www.luabind.sourceforge.net/)

- A library that helps us create bindings between C++ and Lua
  - Utilizing template meta programming, we can easily register C++ functions and call Lua functions
An example of Binding in modules

- For lua scripts to use C/C++ functions

```cpp
void BasicMotion6JPM::init() {
    module(JPLua::L) [
        class_<BasicMotion6JPM>("BasicMotion6JPM")
            .def("swingHead", &BasicMotion6JPM::swingHead)
            .def("stopSwingHead", &BasicMotion6JPM::stopSwingHead)
        ...
    ];

    get_globals(JPLua::L)["basicMotion"] = this;
}
```
An example of Binding in modules

- For lua scripts to use C/C++ functions

```c++
void BasicMotion6JPM::init()
{
    module(JPLua::L)
        .def("swingHead", &BasicMotion6JPM::swingHead)
        .def("stopSwingHead", &BasicMotion6JPM::stopSwingHead)
        ....
};

getGlobals(JPLua::L)["basicMotion"] = this;
```

```lua
basicMotion:swingHead(0,0,0)
```

(in Lua scripts)
An example of robot script

- Swings its head from side to side

```lua
function swingLeft()
    basicMotion:swingHead(0, d2ur(80), 0, "swingRight")
    stree:setState("swingWait")
end

function swingRight()
    basicMotion:swingHead(0, d2ur(-80), 0, "swingLeft")
    stree:setState("swingWait")
end

require "JPLib/Syslog.lua"
require "JPLib/Units.lua"
require "JPLib/STree.lua"

function init()
    stree:setState("swingLeft")
end

function mindNotify()
    stree:doAction()
end
```
Other teams

- **MicroPerl** by team *UPennalizers*
- **Python** by team *rUNSWift* and *CMDash*
- **Scheme** by team *ASURA*

Definition of global wrapping functions for binding to C/C++
/* Get the project for ball. When this function is called,
   it is assumed that the robot can see the ball. */
static PyObject * VisionLink_getProjectedBall(PyObject * /*self*/, PyObject *args) {
    // Track visual ball.
    int inpoints[2] = {((int) (vision->vob[vobBall].cx) - WIDTH / 2), -1 * ((int) (vision->vob[vobBall].misc) - HEIGHT / 2)};
    double outpoints[2] = {-1, -1};
    vision->projectPoints(inpoints, 1, outpoints, 0);
    double ballx = -outpoints[0];
    double bally = outpoints[1];

    PyObject *t;
    t = PyTuple_New(2);
    PyTuple_SetItem(t, 0, PyFloat_FromDouble(ballx));
    PyTuple_SetItem(t, 1, PyFloat_FromDouble(bally));
    return t;
}
Our choice

- **Lua & Luabind** library

Simple function calls for binding to C/C++
An example of Binding in modules

- For lua scripts to use C/C++ functions

```cpp
void BasicMotion6JPM::init() {
    module(JPLua::L) [
        class_<BasicMotion6JPM>("BasicMotion6JPM")
            .def("swingHead", &BasicMotion6JPM::swingHead)
            .def("stopwingHead", &BasicMotion6JPM::stopSwingHead)
        ....
    ];
    get_globals(JPLua::L)["basicMotion"] = this;
}
```
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Our programming method

Create modules by C++
-> Need to compile

Create a binary
-> Need to send the binary and reboot AIBO

Create a script by Lua
-> Need to only send the script

Test on AIBO
Our programming method

- Create modules by C++
  - Need to compile
  - Create a binary
    - Need to send the binary and reboot AIBO
  - Create a script by Lua
    - Need to only send
  - Test on AIBO

For professionals

For beginners
Demonstrations

- Available online at following URL

  https://youtu.be/mehBFIsW4lQ

  https://youtu.be/_EgW_0Isx8U

About 5 minutes

Create a binary and script

About 7 minutes

Create a module and bind it
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- **Discussion**
- Conclusions and future work
Discussion

- Rough Comparison of working efficiency
  - In RoboCup 2004 (before using our framework)
  - In RoboCup 2005 (after using our framework)
Experimental result?

- Me as a beginner (in RoboCup 2004)
  - 3 days to run a sample program
  - 6 months to create a player program for actual games

(Censored image)

me
(I was an undergraduate student)
Experimental result?

- Beginner A and B (in RoboCup 2005)
  - 1 hour to run a sample program
  - 1 month to create a player program for actual games

Beginner A
(they are still undergraduate students)

Beginner B
Jolly Pochie in RoboCup 2005

- 130 modules and 350 scripts
- Top 8 in Soccer competition
- 7th in Technical Challenge
Outline

- Background
- Related work
- Proposed framework
  - Concept
  - Plug-in system
  - Scripting language
- Demonstrations
- Discussion
- Conclusions and future work
Conclusions

- We proposed a framework that makes it easy to create robot programs
  - No need to know the whole system
  - No need to compile and reboot
- Our framework is very useful in the RoboCup Soccer competitions
- Coming soon at following URL
  - [http://www.shino.ecei.tohoku.ac.jp/jollypochie/](http://www.shino.ecei.tohoku.ac.jp/jollypochie/)
Future work

- Create a framework for robots other than AIBO, especially humanoids!
  - We hope that we can use some modules from AIBO, such as recognition modules.
Thank you for your attention!