libARMANI – Version 1.0
Autonomous Robot Manipulation Simulator

Tobias Lang and Marc Toussaint
http://userpage.fu-berlin.de/tlang/armani/

June 12, 2012

When using this library, please cite Lang and Toussaint [2010].

Contents

1 Introduction **1**

2 Installation **1**
2.1 Required Software 1
2.2 Compilation 1

3 ORS Configuration Files **1**
3.1 Cubes and balls 2
3.2 Boxes 2

4 User's Guide **2**
4.1 Simulator Core 2
4.2 Symbolic Relational Representations 2
4.3 Demos 2

5 Trouble Shooting **2**
5.1 ODE 2

1 Introduction

libARMANI is a C++-simulator for **Autonomous Robot MANIpulation of objects**. It simulates a robot which manipulates cubes, balls and boxes of different sizes scattered on a table. This provides a physically realistic extension of one of the most popular scenarios in Artificial Intelligence, namely of the good, old "blocks world".

libARMANI has been developed to serve as an easy-to-use test-bed for **model-based relational reinforcement learning in stochastic domains** [Lang and Toussaint, 2010]: the actions of the simulated robot are affected by noise; relational representations are appropriate to learn and reason with the simulator objects.

2 Installation

2.1 Required Software

- Freeglut (tested with v2.4.0): probably already installed on your system (Debian/Ubuntu: freeglut3-dev)
- QHull (tested with v2003.1): probably already installed on your system (Debian/Ubuntu: libqhull-dev)
- ODE (tested with v0.11): probably needs to be installed by hand (make sure to configure ODE with `--enable-double-precision`, which isn't the default)
- SWIFT++ (tested with v1.2): comes with libARMANI in directory `extern`
- ANN (tested with v1.1.1): probably already installed on your system (Debian/Ubuntu: libann-dev)

2.2 Compilation

First, please compile SWIFT++ in the directory `extern/` and ensure that the correct pathes are set for Freeglut, QHull, ODE and ANN in `make-generic`.

Then, please type `make` in the root-directory `libARMANI/`. This should compile the library `lib/libARMANI.so` and the demos in `test/`. If you have problems getting this working, please contact me. If compilation succeeded, you can check out the demos in `test/`.

You can clean everything via `libARMANI/make cleanAll`.

3 ORS Configuration Files

Configuration files specify all objects and robot body parts used in the simulator: from the robot's individual fingers over the table to the manipulated balls. The configuration files reflect Marc Toussaint's graph-based representation Open Robot Simulator Toolkit (ORS). ORS is the backbone data representation of libARMANI. For more information, please take a look at `http://userpage.fu-berlin.de/mtoussai/source-code/`.

Example configuration files are `test/relational_armani_basic/situation_simple.ors` and `test/relational_armani_basic/situation_box.ors`.

The important part for your modifications is the list of manipulated objects at the end of the file. Please note that these object definitions need to stay at the end of the file.

3.1 Cubes and balls

How to specify cubes (blocks) and balls is best explained by means of an example:

```
body o1 { X=<t(−0.3 −0.7 0.8)> type=1 mass=.1
  size=[.03 .03 .03 .03] color=[0.4 0 .5] contact }
```

`o1` is the name of the object. Objects have to be named `oX` where `X` is a number in ascending order.

`X=<t(-0.3 -0.7 0.8)>` specifies the position in (x/y/z)-coordinates. (To get an idea of the coordinate system, play with the parameters.)

`type` specifies the type of the object which can either be a cube (0) or a ball (1).

`size=[.03 .03 .03 .03]` specifies the size of the object. Use the following sizes for objects:

- Big cube: [.06 .06 .06 .06]
- Small cube: [.04 .04 .04 .04]
- Big ball: [.045 .045 .045 .045]
- Small ball: [.03 .03 .03 .03]

The mentioned requirements need to be met to be certain that all our implementations work correctly. In principle, you are not obliged to them, of course, in order to get the simulator run. But then you might need to modify some of our provided implemented methods.

3.2 Boxes

Boxes are more complicated to specify: they consist of combining an ORS-body with several ORS-shapes (the individual sides of the box). I recommend to simply look at the example file `test/relational_armani_basic/situation_box.ors`.

4 User's Guide

There are two important header files in `src/relational/`:

- `robotManipulationSimulator.h` declares the simulator. It provides methods (i) to query the current state of the simulator and (ii) to trigger actions of the simulated robot. The main actions are grabbing objects and dropping them above other objects.
- `robotManipulationInterface.h` provides a simple high-level interface between the simulator and symbolic relational representations.

4.1 Simulator Core

At its heart, the simulator uses Marc Toussaint's graph-based representation Open Robot Simulator Toolkit (ORS) and approximate inference for control (AICO) [Toussaint, 2009] to control the robot. For more information, please take a look at `http://userpage.fu-berlin.de/mtoussai/source-code/`, in particular `libAICO`, where you'll find a guide explaining all this in detail.

The simulator uses unsigned integer variables to identify objects.

The behavior of the robot is defined in `robotManipulationSimulator`. For example, if we command the robot to grab an object it first puts the inhand object on the table; if it grabs an object below some other object, it will fail to grab this object with 40% probability. You may choose to change all these behaviors according to your own ideas.

The concept of **noise** may be of particular interest for you to implement your own ideas of stochastic worlds. Of course, the simulator by itself is unnoisy and you have to specify explicitly sources of noise. For example, we chose as a major source of noise the calculation of a target location in a puton action (take a look at the method `calcTargetPositionForDrop`). We deliberately add Gaussian noise to the target location. You may want to define differents sources and/or degrees of noise.

4.2 Symbolic Relational Representations

libARMANI provides the interface `robotManipulationInterface.h` to symbolic relational representations. This interface uses relational representations as implemented in `libPRADA`, a library for model-based relational reinforcement learning. You can download this library together with a guide from `http://userpage.fu-berlin.de/tlang/prada/`.

4.3 Demos

The following demos are provided in `test/`:

- `relational_armani_basic/`:
  This example program shows you how to directly control the simulator via `robotManipulationSimulator.h`. Objects are managed by means of their ids (which are unsigned integers), which are used to retrieve state information and to trigger actions.
- `relational_armani_symbolic/`:
  This example shows you how to use an abstract relational representation to control the robot via the interface in `robotManipulationInterface.h`. This uses the extensive logic machinery (see the various files in `src/relational/`) of our experiments in Lang and Toussaint [2010] which is implemented in libPRADA. It provides many convenience functions (writing to files, specifying symbols etc.). See Sec. 4.2.

5 Trouble Shooting

5.1 ODE

Make sure to configure ODE with `--enable-double-precision`, which isn't the default.

In `ors_ode.cpp` and `robotManipulationSimulator.cpp`, you might want to change the following includes

```
# include <ode/../internal/objects.h>
# include <ode/../internal/joints/joints.h>
# include <ode/../internal/collision_kernel.h>
# include <ode/../internal/collision_transform.h>
```

to:

```
# include <ode/../../ode/src/objects.h>
# include <ode/../../ode/src/joints/joints.h>
# include <ode/../../ode/src/collision_kernel.h>
# include <ode/../../ode/src/collision_transform.h>
```

You might also want to change the LinuxLibs ODE library specification appropriately to your needs.

References

Tobias Lang and Marc Toussaint. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49, 2010.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2009.