The magmaOffenburg 2012 RoboCup 3D Simulation Team

Klaus Dorer, Stefan Glaser[1]

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany

Abstract. This paper describes the magmaOffenburg 3D simulation team trying to qualify for RoboCup 2012. While last year's TDP focused on the tool set created for 3D simulation and the support for heterogeneous robot models, this year we focus on the different ways how robot behavior can be defined in the magmaOffenburg framework and how those behaviors can be improved by learning.

1 Introduction

This year our focus is on how robot behavior can be defined in the magmaOffenburg framework and how those behaviors can be improved by learning. Two recent bachelor thesis together with the ongoing efforts of the team have extended the possibilities considerably. The possibilities include hard coded robot movements (section 2.1), behaviors defined by functions like splines or sins (section 2.2), behaviors defined using functions for body part positions using inverse kinematics to reach them (section 2.3), behaviors created from translating motion captured files (section 2.4) and behaviors life recorded using a Kinect (section 2.5)).

2 Robot Behavior

One of the most difficult tasks in soccer simulation 3D league is the proper choreography of moving 22 joints in a way that a desired behavior of the robot is achieved. In an effort to simplify the creation of new behaviors, a couple of methods and frameworks have been evolved in the magmaOffenburg framework.

2.1 Movement Framework

The most basic, but nevertheless useful method to create robot behaviors in the magmaOffenburg framework is provided by the so called movement framework. The model is shown in Figure 1.

Each movement is subdivided into movement phases which are composed of single joint movements. Special features include an automatic conversion of right body parts movements into their symmetric left body part movement. Also movements allow to define speeds for each joint so that a joint not necessarily

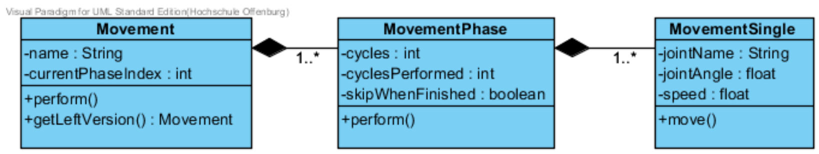

Fig. 1. Model of the movement framework.

has to move until the end of a movement phase. Behaviors based on movements can be closed loop by defining conditions at any phase of the movement allowing to prolong or skip movement phases or changing single movements completely. Example behaviors using the movement framework are getting up and kicking, both above average in the league. The movements are specified as hard coded java code (Fig. 2) though it would be simple to read the corresponding values from xml files or similar.

```
keep = new Movement("keepRight");
keep.add(new MovementPhase("phase1", 200) //
      .add(INaoConstants.LShoulderPitch, 90, 7f) //
      .add(INaoConstants.LShoulderYaw, 0, 7f) //
      .add(INaoConstants.LArmYaw, 0, 7f) //
      .add(INaoConstants.LArmRoll, 0, 7f) //
```

Fig. 2. Example implementation (part) of a movement.

2.2 Function Behaviors

Static behaviors can be defined using functions that model the joint angle over time. In order to simplify the definition of such functions, a function editor has been created. It supports piecewise linear functions, sinus functions and splines (see Figure 3). Support points can be moved in any direction between its neighbor support points. Whole movement phases can be selected and shifted, stretched or removed. The behavior can be tested in real speed from within the tool. And finally a cursor can be moved at any speed and direction to test each part of the behavior or to move support points while the robot is following the movement. The result is stored in a behavior file and can be used by the robots.

2.3 Inverse Kinematics Control

In contrast to static or semi-static robotic movement definitions, modelling dynamic movements involving further sensor information is essential in reaching

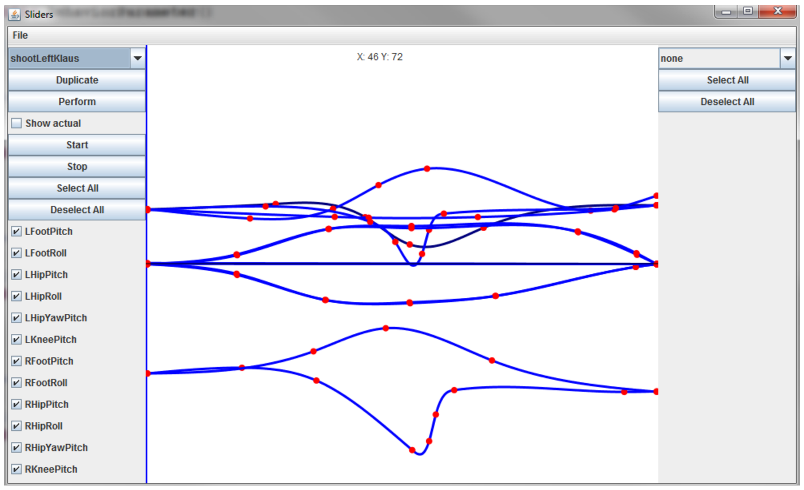

Fig. 3. Function Behavior Editor.

human like movement skills. These, usually already complex behaviors get further blown up by calculations regarding direct joint control. Furthermore, with the introduction of heterogeneous robot models, all behaviors controlling joints directly are most likely to fail when porting them on a new robot model and have to be rewritten or modified.

Since the magmaOffenburg framework already provides a dynamic robot model, described by the last TDP, using this model for inverse kinematics calculations was the natural consequence. For this purpose each body part dynamically provides a method to create the Jacobian matrix along the chain of body parts to the root body part. A body part instance provides anything needed to perform the typical basic cycle of inverse kinematics methods between requesting the Jacobian matrix and performing delta movements on the corresponding joints. This way different inverse kinematics methods can be used to optimize the position/orientation of a specific body part, of a body model instance representing an arbitrary (hierarchical) physical robot model, towards a target position/orientation in space. Current effort is spent on investigating different inverse kinematics methods and their applicability in the humanoid robot domain.

Defining inverse kinematic controlled behaviors can be done similar to most other methods described in this TDP by simply replacing the term “target angles to joints” with “target position/orientation to body parts”. However, the real power behind inverse kinematics lies in the abstraction of the physical properties of the robot. With this framework, behaviors can calculate target positions and orientations dynamically using further sensor information and simply hand this information to the inverse kinematics framework to move the robot appropriately. Behavior creation in general is much easier and with a scalable design it should also be portable to other robot models. First results show that movements calculated by inverse kinematics are well competitive to their equivalents using direct joint control, but the portability of such behaviors to other robot models is not yet verified.

2.4 Motion Capturing

As a first step towards human like behavior, Matthias Kurth [2] has created a converter that is able to convert motion capturing files in Acclaim ASF/AMC format to movement files of a Nao robot. A core component of this work is the mapping logic between the different skeletons defined Acclaim and the Nao robot (see Figure 4).

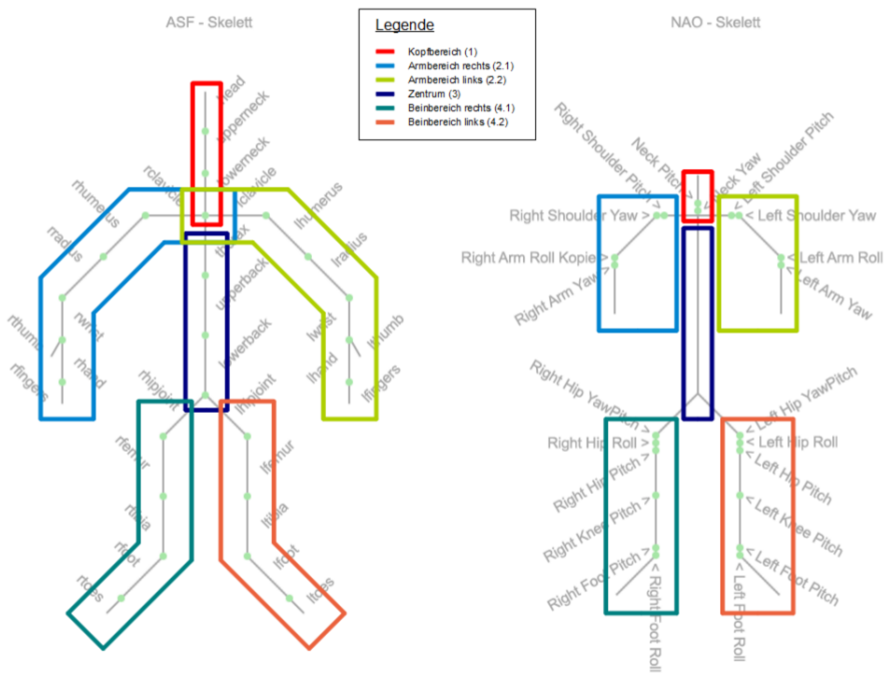

Fig. 4. Acclaim ASF/AMC and Nao skeleton.

2.5 Kinect Control

The next step towards human controlled behavior has been done by Björn Ritter [1] who has created a framework that allows life control of a simulated Nao robot through a Kinect device. Motions are captured by the Kinect and transformed into a Nao skeleton in a piece of code written in C#. Since the magmaOffenburg framework is written in Java, the access to the data is done through a JNI Bridge (see Figure 5).

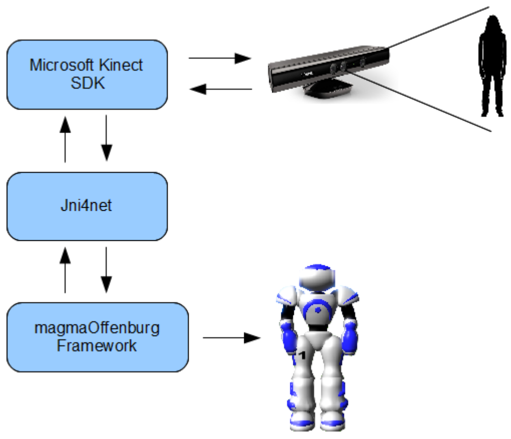

Fig. 5. Architecture for human to Nao robot control through a Kinect.

A snapshot of the result can be seen in Figure 6. Motions of interacting humans are translated life to motions of the simulated Nao robot.

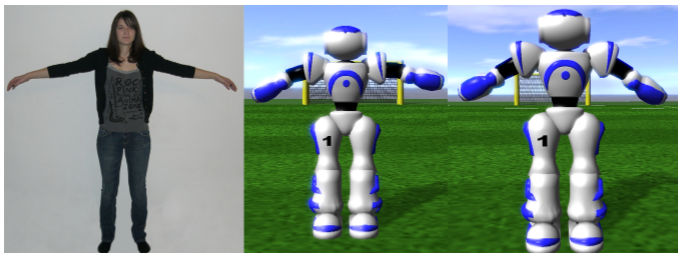

Fig. 6. Example of a motion created life by interaction with a Kinect. Left the human motion, middle Nao robot using forward kinematics, right Nao robot using inverse kinematics.

Current limitations include that leg movements almost always result in the robot falling down. Also the Kinect is not able to detect foot angles or arm rotations. To achieve some smoothness in motions, Kinect percepts have to be filtered to some degree, causing roughly 0.5 s latency (including the latency of the Kinect itself).

3 Behavior Optimization

Walking of our robots is based on the work of [8] and described in [3]. In this approach, successful walking depends on roughly 15 parameters that have to be fine tuned to fit to the robot model. Two methods proved to be useful to explore valuable parameter sets: genetic algorithms and tabu search. While the genetic algorithms proved to be more successful in exploring initial valuable areas of the search space, tabu search was used to fine tune parameter sets found by genetic algorithms. For both approaches the 'as fast as possible' server mode was used to run simulation about eight times faster than real time. Optimization has been spread on a cluster of six computers.

3.1 Genetic Optimization

Classic genetic optimization was used to find promising areas in the search space. The population typically consisted of 20 individuums, selection was done using Monte-Carlo selection, reproduction was done using multi-crossover and no mutation was used. The fitness function included the distance reached (most important), the deviation from desired angle and from walking straight. Training was immediately interrupted if the robot fell down or turned more than 45 degrees from walking straight. The utility was averaged over five runs that started with a standing robot and ended after 500 cycles at most. The results of such a run can be seen in Figure 7. The trend of the fitness of all individuums shows an improvement of roughly 40% after running fitness measurements for roughly 2000 robots, i.e. 100 generations.

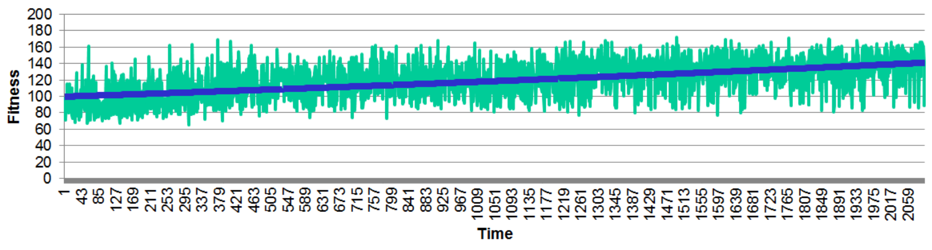

Fig. 7. Improvement of individuums over time using genetic optimization.

3.2 Tabu Search

Since calculation of fitness is quite expensive in this domain taking several seconds for each individuum, final fine tuning of parameters in interesting areas of the search space was performed by tabu search. The neighborhood has been defined to consist of 10% variation steps for each parameter. Because utilities are quite similar for those small kinds of changes, 20 runs have been performed for each position in the neighborhood to average out fluctuations. The small size of the chromosoms allowed for tabu list sizes that rendered cycles in search space impossible. Roughly another 10% improvement with respect to fitness values of the best individuums could be achieved.

4 Team

The magmaOffenburg team:

- Klaus Dorer (Team leader)
- Stefan Glaser
- Fabian Korak
- Maximilian Krög
- Ingo Schindler

References

1. Ritter, B.: Steuerung simulierter Roboter mit Kinect. Bachelor thesis, Hochschule Offenburg, Germany (2012)
2. Kurth, M.: Steuerung eines simulierten Roboters mit Hilfe von Motion Capturing Files. Bachelor thesis, Hochschule Offenburg, Germany (2011)
3. Schindler, I.: Laufen auf zwei Beinen in der simulierten RoboCup 3D-Umgebung. Bachelor thesis, Hochschule Offenburg, Germany (2009)
4. Dorer, K.: Modeling Human Decision Making using Extended Behavior Networks. J Baltes et al. (Eds.): RoboCup 2009, LNAI 5949, pp. 81–91. Springer, Heidelberg (2010)
5. Dorer, K.: Extended Behavior Networks for Behavior Selection in Dynamic and Continuous Domains. In: U. Visser, et al. (Eds.) Proceedings of the ECAI workshop Agents in dynamic domains, Valencia, Spain (2004)
6. Dorer, K.: Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten. PhD thesis, Albert-Ludwigs University (2000)
7. Dorer, K.: Behavior Networks for Continuous Domains using Situation–Dependent Motivations. Proceedings of the Sixteenth International Conference of Artificial Intelligence (1999) 1233–1238
8. Huang Qiang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hi- rohiko Arai, Noriho Koyachi und Kazuo Tanie: Planning Walking Patterns for a Biped Robot. IEEE Tranactions on Robotics and Automation, 17:280–289, 2001.
9. Maes, P.: The Dynamics of Action Selection. Proceedings of the International Joint Conference on Artificial Intelligence (1989) 991–997
10. Veenstra, A., Neijt, B., Vermeulen, F., Veenstra, G., Prins, J., Kuypers, J., Stollenga, M., vd Sanden, M., Klomp, M., Platje, M., van Dijk, S. The Little Green Bats at http://www.littlegreenbats.nl/ (2008)