

Bold Hearts Team Description

RoboCup 2015 Kid Size

Sander G. van Dijk, Drew Noakes, Daniel Barry,
Masood Ahmed, and Daniel Polani

Adaptive Systems Research Group
School of Computer Science
University of Hertfordshire, UK

Abstract. In this paper we describe the RoboCup Humanoid Kid Size division of team Bold Hearts, the RoboCup team of the University of Hertfordshire, in Hatfield UK. We discuss construction and aspects of our software architecture, as well as some of our solutions to important problems such as vision, localisation, action selection, and communication. Finally, we provide an overview of the relevant experience and achievements of our team.

1 Team and Robots

Team Bold Hearts, from the Adaptive Systems Research Group at the University of Hertfordshire, UK, has participated in RoboCup since 2002. Initially solely in the simulation leagues, we spread out to the kid-size humanoid hardware league in 2013. To facilitate this step, we have chosen the popular DARwIn-OP robotic platform to make up our robotic team. Currently we use this platform unchanged, i.e. they conform to their original specification; our focus has been on the software controlling them. Starting from the Robotis DARwIn-OP software supplied with the robots, we have replaced many parts with our own implementations, with the gait module being the main part still mostly in its original form. Additionally, we ported parts of the `libbats` framework originally developed for the 3D simulation league, which our team members released and maintain as an open-source project¹.

The history of the team in the simulation league has puts us in a good position to transfer experience between the leagues. With this in mind we strive to keep using a shared code-base and framework for both leagues. The remainder of this paper will outline some of the most important aspects of this framework as they apply to the kid-size league.

2 Software Architecture

The agent employs a modular, multi-threaded architecture implemented using both C++ and Python. With the first used for low level control and sensor

¹ <http://www.github.com/sgvandijk/libbats>

processing, and the latter for higher level control, this combination strikes a balance between efficiency and ease and speed of development.

The core of the framework consists of two control loops: one high frequency loop for motion control (125Hz), and one slower loop that controls vision and behaviour generation (30Hz). An overview of these two main threads of operation is provided in the following subsections, with more detailed discussions of some aspects given thereafter.

2.1 Cognition

The cognition thread loops through a set of operations at a rate governed by the camera, achieving 30Hz on our hardware and utilising every available image. Camera frame data is processed to find visual features, which are then projected into a 3D coordinate system relative to the agent. The localiser is updated using odometry derived from forward kinematics, filtered IMU data and the observed objects. With this revised posture estimate, the position of the agent and other dynamic objects are calculated in the World Frame.

Having integrated the latest perception of the environment, the behaviour subsystem runs, which results in target actions for the high frequency control loop. These actions consist of motor trajectories, and periodically also includes data to be transmitted to team mates via UDP.

A more detailed discussion of these different parts of this cognition process are given below.

2.2 Hardware Communication

The DARwIn-OP uses a CM730 subcontroller for motors, accelerometer, gyroscope, buttons and LEDs. A dedicated, real-time priority thread has exclusive access to this subcontroller, marshalling all communication efficiently along its bus.

With an update period of 8ms, the hardware loop runs with a frequency of 125Hz. Each cycle, the instantaneous position, temperature, load, voltage and alarm state is read from all 20 motors, along with the accelerometer and gyroscope readings. This data is provided to several motion modules, when active, which perform locomotion, head control and scripted motion playback. Target positions and PID gain values generated by these motion modules are transmitted to the motors. Any change to the status LEDs are also written at this time, as a means of communicating certain important states to the robot handler.

2.3 Configuration

We use a tiered configuration system divided into the following levels: Agent configuration, Location / Event configuration, and Team configuration. Configuration settings of higher levels can override those set at lower levels.

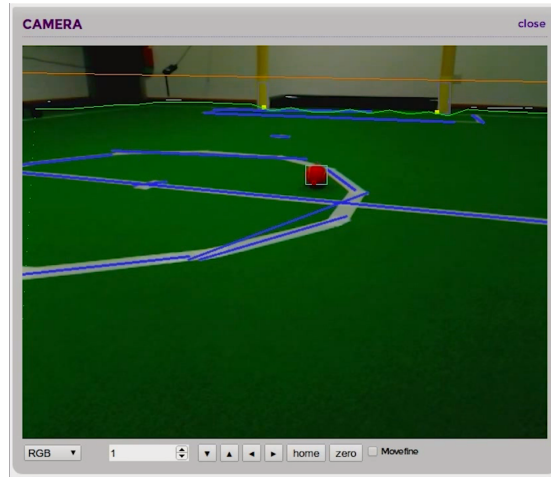


Fig. 1. Screenshot of vision debugging. Rectangles: detected objects; small filled squares: object ground positions; orange line: horizon; green line: field boundary; blue lines: detected lines.

Configuration values are strongly typed and limited to within sensible ranges. Values of most configuration settings may be modified at run time via our debugging tools, allowing convenient, interactive and rapid evaluation of parameters online.

2.4 Debugging

For debugging we built an HTML5 application called *Round Table* which runs in modern browsers and is served directly from the robot via HTTP. The core state of the agent is modelled as a set of immutable state objects, and each of these objects is exposed from the agent via WebSockets. Our debugger subscribes for updates to whatever state objects are required for the particular task at hand.

Further, most configuration settings are able to be modified using this browser-based tool. Examples include watching the live video feed, viewing real-time charts of IMU data, 2D and 3D models of the agent's local and world frames, behavioural reasoning, game state and detailed per-thread timing breakdowns just to name a few.

The website is served directly from the robot. This approach makes it possible to debug using any device with a recent web browser, with the debug interface always up to date with the software running on the robot.

3 Vision Pipeline

The vision system provides some of the most critical data for building the agent's belief about the state of its environment. To reduce the computational cost of

vision, we use a sampling technique that applies a uniform spacing across the ground plane relative to the agent. Consequently the greatest pixel density is maintained at great distances, while sparse sampling occurs in the foreground where objects are largest on the camera plane.

Since the 2014 world championship, we have further optimised our vision stack, such that it is able to fully process 640x480 pixels within 4ms. This allows us to increase the sampling resolution, and frees up computation time for more advanced algorithms that we are currently developing to deal with the more challenging environment this year.

From each image we extract the ball position, goal post positions, and field lines. Additionally, we have multiple techniques for estimating both the visual horizon and the edge of the field which assists in preventing incorrectly identifying objects outside the predictable, controlled environment of the field where objects must be assumed to be of all shapes, sizes and colours.

A colour look-up table is produced offline for use during competition matches. This year we have developed a colour table training interface in Round Table to ease the set-up process in a new environment. In this interface, a user clicks on colour patches to adjust colour ranges, rather than having to set them by hand in configuration files. During game play the pixels of each camera frame are labelled according to the look-up table, and these labelled pixels are processed by all vision subsystems in several passes.

For object detection we use a run-length encoding which is subsequently processed using a disjoint set to build up contiguous regions. Our line detection strategies include a progressive probabilistic Hough transform, and a segmented linear regression model. Each of these methods can be configured and swapped with different implementations on the fly using our debug tool, to make the process of experimentation and configuring during the set up phase of a RoboCup event as efficient as possible.

4 Localisation

Up to this point all the intelligence of our robots is solely based on local models: they do not maintain a global model of their location, or that of other objects. We have found that this prevents in some way common problems seen with popular localisation methods, such as drift over time, or permanent ‘flipping’ of the global model. This means there is no degradation of performance over time: our robots have the same accuracy, where those of other teams might get lost over time.

Of course, some method is needed to determine which direction to play at. This is achieved through team communication and vision, based on the assumption that our goalie is near to our own goal. When she sees the ball, our goalie announces its position relative to herself to her teammates, who are then able to deduce which goal to shoot for based on the observed relative position of a goal to the ball. Additionally, team markers are used by our field players to recognise our own goalie, to give further direction.

Although relatively stable, these methods slow down players as they constantly need to search out local features. Therefore we have continued our effort this year on developing global models to enhance decision making. We have developed a generic recursive Bayesian filtering module, on top of which we implemented standard filters such as a Kalman filter, particle filter, and an unscented Kalman filter, and are working on improving their reliability. Below we describe a method used to integrate sensor data for use in localisation.

4.1 Merging Gyroscope and Accelerometer

We integrate measurements of the tri-axes gyroscope and the accelerometer to keep track of the orientation of the robot, using a gradient descent based algorithm [4]. This algorithm operates on quaternion descriptions of the orientation of the robot and of the change thereof. Given a reading of the angular rate around each of the three axes $\boldsymbol{\omega} = [0 \ \omega_x \ \omega_y \ \omega_z]$, the change of rotation in quaternion form can be calculated as $\dot{\mathbf{q}}_{\boldsymbol{\omega}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}$, where \mathbf{q} is the current rotation and \otimes signifies the quaternion product. An orientation estimate can then be maintained by integrating this change over time. However, measurement noise and a limited measurement period will induce error of the estimate growing over time. To help overcome this, the accelerometer reading is incorporated into the estimate: it provides an error measurement as the difference between the observed field of gravity and that expected given the current orientation estimate. Posing this error as an objective function to minimise using gradient descent results in a simple update rule:

$$\mathbf{q}_{est} = \gamma \mathbf{q}_{\boldsymbol{\omega}} + (1 - \gamma) \mathbf{q}_{\nabla}, \quad (1)$$

where \mathbf{q}_{∇} is the orientation found by performing a gradient descent step, and $\gamma \in [0, 1]$ is a trade-off parameter. It turns out that a single gradient descent step at each cycle, rather than performing steps until convergence which would be too expensive, is sufficient to minimise the error over time [4].

5 Behaviour and Action Selection

In this section we will discuss our behavioural framework, which has grown out of the experience obtained in the simulation league where the focus lies heavily on this aspect of the RoboCup challenge. From this experience we have extracted some properties that a useful behaviour framework should have: it should be highly modular so separate parts can be developed in parallel, different pieces of behaviour can easily be reused, and behavioural solutions with different implementations can be interchanged; it must be as nonrestrictive as possible, as not to hit future limitations; hierarchy is important, to be able to build up complexity over time; and finally it must directly enable the use of well developed machine learning techniques.

Our behavioural framework is based on the so-called *Options* model[7]. The final outcome of a behavioural framework is the selection of primitive actions: actions that are instantaneously executed, usually once per some defined time

Algorithm 1 EXECUTEOPTION

Require: $\langle \mathcal{I}, \pi, \beta \rangle$

- 1: **repeat**
 - 2: Observe current state s
 - 3: Select sub-option $o \in \mathcal{I}$ according to probability distribution $\pi(s, \cdot)$
 - 4: EXECUTEOPTION(o)
 - 5: **until** $\beta(s) \geq \mathcal{U}(0, 1)$
-

step, or think cycle. In our scenario these actions consist of direct motor control, or acts of communication. More complex behaviour however consists of tightly connected series of such basic actions, and to make such behaviour feasible it must be described in the form of higher levels of abstraction. An *option* is such a higher level abstraction of an action, specifically forming an abstraction over time: an option is not necessarily instantaneous, it can be active over a length of time.

Formally, an option is described by three components: an *initiation set* \mathcal{I} that determines in what states the option can be selected; a *policy* π that describes the action selection when this option is selected; and a *termination condition* β that (stochastically) determines when the option ends. The case where the policy directly selects primitive actions amounts to a single level of abstraction. However, one can also allow the selection of other *sub-options*, incurring a recursion that forms an unlimited hierarchy. Finally, primitive actions can be modelled as options that always terminated after one step, ensuring we can use a single structure throughout the whole hierarchy. Algorithm 1 lists the full process of executing an option.

The Option model was first formulated in the context of Reinforcement Learning, and has been used and further developed in that area of research extensively for the last 15 years [5, 1, 2, 6], which will allow us to apply and further develop this work in the context of RoboCup. However, the framework is abstract enough to allow for a multitude of implementations of the options. For instance, the ubiquitous finite state machine is, in our framework, implemented as an option where the policy is determined by the state of the machine, and the termination condition is triggered by the state machine transiting into some final state. Other action selection mechanisms such as artificial neural nets or decision trees fit into this framework just as easily.

6 Communication

We incorporate a non-blocking open UDP standard of communication between robots that is developed by the FUManooids team under a grant from the RoboCup federation. This allows the bots to reliably transmit information about their current location and orientation relative to the global coordinate frame, observations made relative to the robot's local frame, the role in the team including confidence in those values and much more.



Fig. 2. Simulation of a team of DARwIn-OPs in a kid-size field, implemented on the RoboCup soccer simulator

This makes communication between players of different teams standardised, therefore allowing them to communicate and participate in mixed teams, which opens the way to new research directions of interest to our group and could support the growth of the hardware league to large teams with high level play. We have successfully played several of these mixed games, most recently in friendly games at the November 2014 RoHOW event in Hamburg, where our robots played side by side with some of the Hamburg BitBots.

7 Simulation

Using simulation can make testing and optimisation much faster. As an example, our team has previously used simulation to perform large scale optimisation of gait speed under energy constraints [3]. Given our background in the 3D simulation league, including providing active development of the official simulator as members of the league’s Maintenance Committee for many years, the obvious choice was to use this simulator, SimSpark/RCSSServer3D, within our hardware team. As such, we have ported a model of the DARwIn-OP, as well as the kid size arena, to this platform. Figure 2 shows a screen-shot of the simulator running a team of robots using this model.

Currently this integration is still in an early stage: the model needs to be improved further to be able to use the simulation for realistic physical testing and optimisation. This opens up several interesting paths of research, possibly involving bidirectional optimisation between simulation and in the physical scenario. Furthermore, this work may directly feed back into the simulation league, to support the effort of moving the leagues together. This year we are mostly using the system to try out localisation and high level behaviour logic.

8 Experience and Achievements

Team Bold Hearts has striven to build up as much experience as possible in the short period of time since we joined the humanoid league. During the competitions we have attended we performed our referee duties, with most of our current team members having this experience and full knowledge of the rules. Besides these official RoboCup events, we have also participated in several RoboCup workshops with German and UK teams. We anticipate attending a meet up in Berlin this March, as well as the German and Iranian Opens in April.

The following are the detailed achievements and contributions of team Bold Hearts of the last five years, both in the Simulation, and in the Kid-Size League.

- *Achievements Kid Size: 2nd Robocup World Championship 2014*, 3rd Iran Open 2014; 2nd German Open 2014, 3rd German Open 2013
- *Selected achievements Simulation: 2nd RoboCup World Championship 2009; 3rd RoboCup World Championship 2012; 1st German Open 2009, 2010; 1st Iran Open 2011; 2nd Dutch Open 2012*
- Members SL Technical Committee 2009, 2011, 2012
- Chair SL Organising Committee 2010, 2013
- Awarded RCF sponsored project ‘Boosting the 3D SSL Simulator’

References

1. George Konidaris and A.G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.
2. O Kozlova, O Sigaud, and C Meyer. Automated discovery of options in factored reinforcement learning. In *Proceedings of the ICML/UAI/COLT Workshop on Abstraction in Reinforcement Learning*, pages 24–29, Montreal, Canada, 2009.
3. Valerio Lattarulo and Sander G. van Dijk. Application of the “alliance algorithm” to energy constrained gait optimization. In *The 15th Annual RoboCup International Symposium*, pages 393–404, Istanbul, Turkey, 2011.
4. S.O. Madgwick, A.J. Harrison, and A. Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *IEEE International Conference on Rehabilitation Robotics*, page 2011:5975346, 2011.
5. A. McGovern and A.G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML 01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
6. Özgür Şimşek and Andrew G. Barto. Skill characterization based on betweenness. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1497–1504, 2009.
7. Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.