Bold Hearts
Team Description for RoboCup 2019
(Humanoid Kid Size League)

Marcus M. Scheunemann, Sander G. van Dijk, Rebecca Miko, Daniel Barry, George M. Evans, Alessandra Rossi, and Daniel Polani

School of Computer Science
University of Hertfordshire, Hatfield AL10 9AB, UK
marcus@mms.ai
https://robocup.herts.ac.uk

Abstract. We participated in the RoboCup 2018 competition in Montreal with our newly developed BoldBot based on the Darwin-OP and mostly self-printed custom parts. This paper is about the lessons learnt from that competition and further developments for the RoboCup 2019 competition. Firstly, we briefly introduce the team along with an overview of past achievements. We then present a simple, standalone 2D simulator we use for simplifying the entry for new members with making basic RoboCup concepts quickly accessible. We describe our approach for semantic-segmentation for our vision used in the 2018 competition, which replaced the lookup-table (LUT) implementation we had before. We also discuss the extra structural support we plan to add to the printed parts of the BoldBot and our transition to ROS 2 as our new middleware. Lastly, we will present a collection of open-source contributions of our team.

1 Bold Hearts

The team Bold Hearts has been founded as part of the Adaptive Systems Research Group at the University of Hertfordshire. The team started participating in RoboCup in 2003 in the simulation leagues and made a transition to the Humanoid KidSize League in 2013. We hope to participate in that league in 2019 for the seventh year in a row.

The following are the main achievements of team Bold Hearts in the Humanoid League over the last few years.

- Quarter-finalist RoboCup World Championchip 2017 (1st in group)
- 2nd round RoboCup World Championship 2016 (1st in group)
- 1st Iran Open 2016
- 2nd round RoboCup World Championship 2015 (1st in group)
- 3rd German Open 2015
- 2nd RoboCup World Championship 2014

2 Introducing New Members Gradually

Recruiting new members is a crucial task, as with most RoboCup teams. It is important for the team's overall success to continuously recruit new members and transfer knowledge to the new generations.

We have always kept a well maintained wiki for new members to read up on the given infrastructure. Additionally, we have made it easier to set up the code by using tools such as Ansible and Docker[1]. However, there has been a lack of students with C++ knowledge. This is the language of choice for our custom framework, which is presented in previous team description papers [4]. Additionally, robotics itself is very complex, therefore working with real robots without pre-knowledge is a challenge in itself.

We decided to tackle this issue on different levels. Firstly, we plan on moving to a framework which allows modular development in Python and C++, see section 5 for more details.

Students then need to understand the level of complexity of robotic tasks. Some known contributors to the RoboCup community approached this issue gradually themselves, by firstly participating in, e.g., simulation league and only later entering the hardware league. We want to emulate this locally, by offering students a very simple and accessible idea of RoboCup with a simple, standalone 2D simulator called PythoCup. It is written in Python and was firstly developed for the Humanoid soccer school 2013 by our team. It has now been adapted for the use of PyGame and is published on GitLab[2].

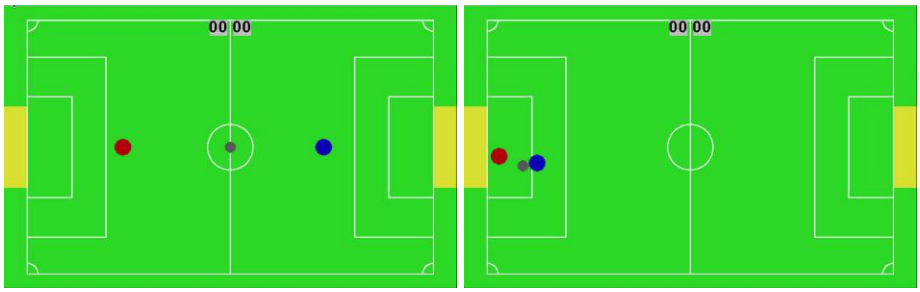

Fig. 1. Two scenes of a PythoCup game. The left screenshot shows the moment before the game starts. The right screenshot shows the blue player attacking the goal of the red player.

We expect several benefits for new students/participants and existing team members. Setting up PythoCup is simple and manipulating the behaviour of the robots can be achieved in a few steps, yet it offers the possibility to achieve already quite sophisticated agents. We hope that this will ease the process for

[1] https://www.ansible.com/ and https://www.docker.com/
[2] https://gitlab.com/boldhearts/pythocup

new members joining the team. Additionally, we expect that some important robotic related problems will derive naturally and therefore we hope that new members get a glimpse of RoboCup related problems quickly. Another benefit is that those who have learnt these skills can then help introduce new members, creating a pyramid of experience.

After mastering PythoCup, the next step will be to allow students to set up our code for the humanoid robots. They will be given an isolated modular problem to solve. Testing will be done in the simulator (e.g. Gazebo) and already small changes will yield different, visible output.

3 Robotic Hardware and Design

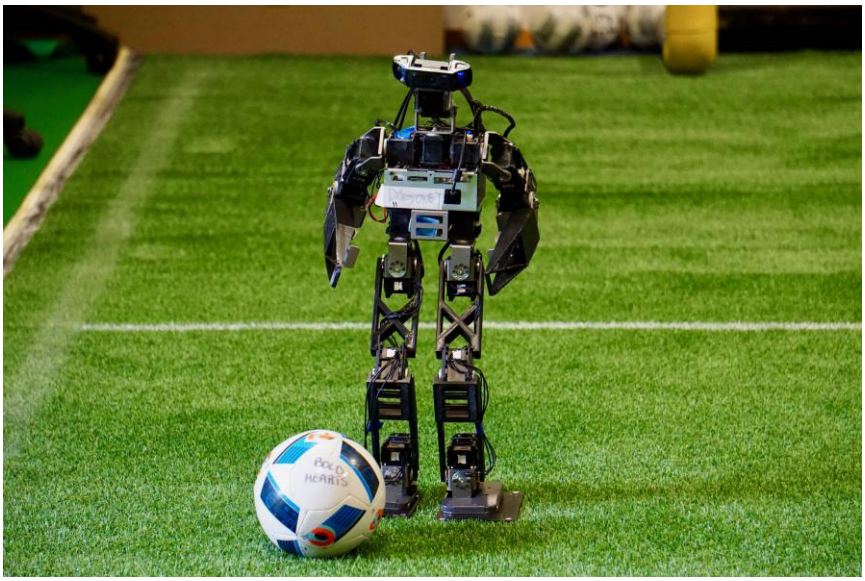

Fig. 2. The BoldBot robot in its second version. It is incrementally developed with a Darwin-OP as its base. The torso has been scaled up to fit an Odroid-XU4, the new main processing unit. The shin, thigh, arm, head bracket and the foot plate have been redesigned, scaled up and 3D printed.

As described in our last years' team description paper, we started with incrementally developing a new robot platform based on the Darwin-OP [4].

The main processing unit has been replaced with an Odroid-XU4. Shins, legs, foot plates, head bracket and arms have been redesigned and 3D printed. Figure 2 shows one of our robots with its newly designed parts. At the RoboCup 2018 competition, we participated with 4 robots of that configuration. The robots were equipped with four different webcam models: Logitech C910, C920, C920c and

C930e. For this year's configuration, we decided on using Logitech C920 Pro HD webcam for all robots.

For the self-printed limbs, we mainly used PLA and ABS and a range of different printers. PLA seems to be most sturdy and well suited for our needs. One of our biggest challenges during the RoboCup 2018 competition was mounting the printed parts to the servos. The plastic parts had to resist a lot of stress on a small area of contact of the screws. When the plastic parts broke, they usually did so around the horn mounting area.

To help address this, we use the outer horn disc as additional support for the motors, as seen in Fig. 3. This gives greater support in 180 degrees (towards the model) where the parts are typically stressed. Despite being thinner, the parts are stronger as the force is spread more evenly across the model.

In the near future, we plan to investigate the use of metal inserts to further increase the strength of printed parts in the form of: small washers per screw, a larger washer inserted into the model and an embedded nut per screw. For all

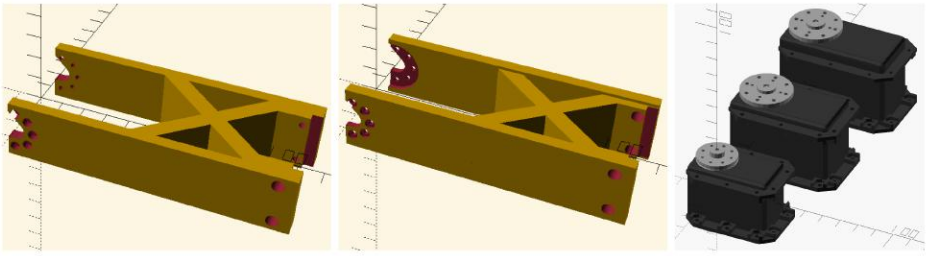

Fig. 3. Depicted is the design of the thigh of the BoldBot model used in RoboCup 2018 (left).We redesigned this part with some additional support structure (middle). We will investigate whether the support for the horn and bearing will help to reduce the stress on the screws. The picture on the right shows the design of Dynamixel servos. These models can be used in OpenSCAD for designing limbs.

our designs we utilise OpenSCAD, a tool which allows for parametric designs, enabling us to adapt the length of a limb without redesign. Like the Darwin, all BoldBot servos are Dynamixel MX-28. It turned out that, with increasing the robot size, these servos are already too weak for the robot to stand up or to locomote. We therefore also parametrised Dynamixel servos for the models MX-28, MX-64 and MX-106[3] with OpenSCAD. This enables us to redesign limbs with reference to the used model more easily.

4 Vision

In previous years, our object recognition methods were based on a lookup-table (LUT) approach. The LUT was created based on thresholds in HSV colour

[3] Open-sourced here: https://gitlab.com/boldhearts/dynamixel-scad

space, that were manually tweaked for each separate competition and/or field. Besides being time consuming during setup, the method was no longer very applicable in the modern non-colour-coded RoboCup scenario.

The hardware upgrade that our robots received allows the application of more advanced computer vision methods, however it is not yet feasible to run some of the latest large-scale deep learning models. We managed to scale such models down to be able to run fast enough on our mobile hardware with sufficient accuracy, the full details of which were presented at the RoboCup 2018 symposium [6]. Here we will summarise this work.

Rather than using a direct object recognition approach, similar to the popular YOLO and RCNN family of CNNs, which are too complex to run or need a highly optimised domain-specific candidate selection process, we use the more general method of semantic segmentation. Besides being able to process a full resolution frame faster, without requiring specific domain knowledge, this method has the additional benefit of a single network being able to handle multiple image resolutions without retraining. Finally, the output is a per-pixel labelling of the image, equivalent to the output of traditional LUT based methods, so it fits seamlessly into our existing pipeline.

The neural networks that we use have an ecoder-decoder structure similar to other, large-scale, segmentation networks in the literature, such as U-Net [3] and SegNet [1]. In such networks, a first series of convolution layers encode the input into successively lower resolution but higher dimensional feature maps, after which a second series of layers decode these maps into a full-resolution pixelwise classification. This architecture is shown in Fig. 4.

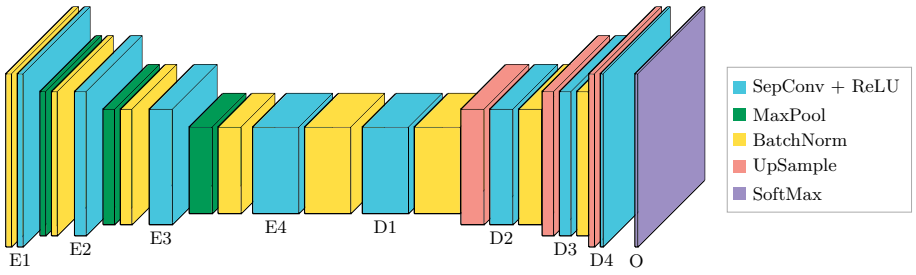

Fig. 4. The architecture of the network consists of a series of fully convolutonal encoding (E1–E4) and decoding (D1–D4) steps. A pixelwise softmax output layer provides the final classifications.

The main techniques that make it possible to process frames at full resolution and high rate are:

Depthwise Separable Convolution By breaking up a full 3D convolution operation into separate per-layer 2D convolutions plus a 1x1 depthwise convolution, the number of computations per layer is reduced significantly, without a great loss of performance.

Stride Using a stride of 2 reduces the number of computations in a convolution layer by 4, again without great loss (and sometimes even slight increase) of performance.

Figure 5 shows typical results of these networks. At the RoboCup 2018 competition we were able to train and run such networks by collecting and labelling imagery at location (using the Bit-Bots' Imagetagger[4]). Training of the network took a few hours on a laptop with a GeForce GT 750M GPU.

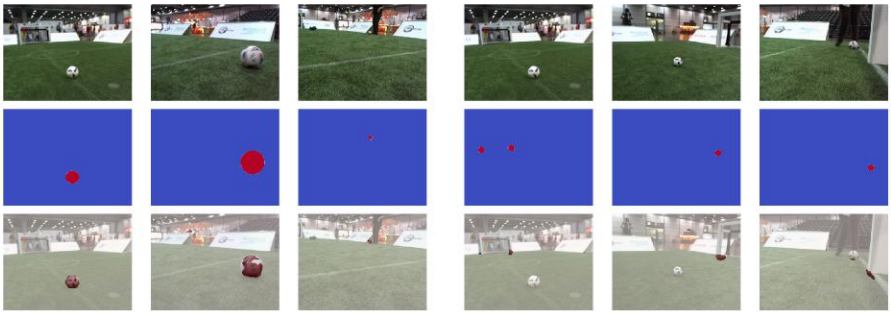

Fig. 5. Examples of input (top), target (middle) and segmentation outputs (bottom) of the segmentation network. Left: ball detection, right: goal post detection.

During development of these methods, several open sets from the Imagetagger were used. Additionally, we have created and released two datasets: one with the bottom of the goalposts annotated to train goal detection and one with ball anotations created and used during the RoboCup 2018 competition in Montreal (See section 6).

5 Using ROS 2 as Middleware

From 2013, we have developed and used our own software framework, with all modules created from scratch, except for some that were partly based on the source code originally supplied with the Darwin-OP platform [5]. Although shown to be capable of performing well, over the years the framework has become more and more complex, and being completely custom it is now difficult for new members to get into it. We have opted to replace the base framework completely, and have reviewed several options: a new framework from scratch again, the NUClear framework[5] [2], ROS 1[6], and ROS 2[7].

[4] https://imagetagger.bit-bots.de/
[5] https://github.com/Fastcode/NUClear
[6] http://www.ros.org/
[7] https://index.ros.org/doc/ros2/

ROS 1 was discarded as an option, as we have learned throughout the years that efficiency of the framework is crucial for our limited hardware, and seeing that multi-robot teams, small platforms and real-time systems are explicit use cases that ROS 1 is not ideal for that sparked the development of ROS 2. NUClear does offer a very efficient, modular platform, that has been proven in the RoboCup scenario. However, ROS 2 is currently in a state that offers most of the same benefits, while having much wider support, including from large entities from the industry, such as Intel and Amazon. With an eye on the future and the transferability of skills learned by our (student) members outside/after participation in the team, we opted to use ROS 2.

ROS 2 is based on Data Distribution Services (DDS) for real-time systems. This connectivity framework aims for enabling scalability and real-time data exchange using a publisher-subscriber architecture. ROS 2 sits on top of that, providing standard messages and tools to adapt DDS for robotic needs. Publishers and subscribers can be written in C++ or Python.

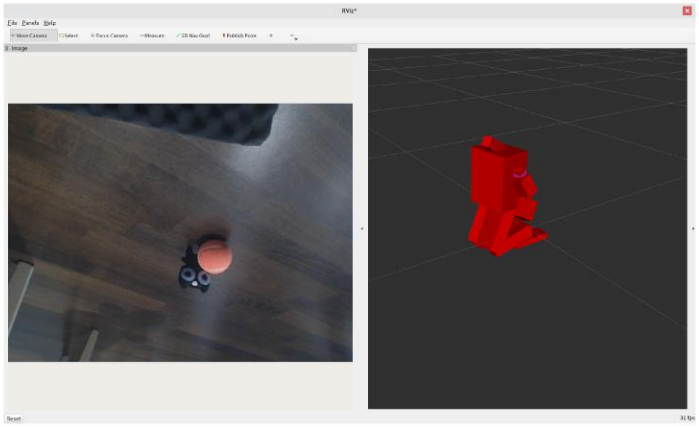

Fig. 6. Screenshot of RViz2, showing camera feed and robot model built from joint states read from subcontroller.

Our main efforts to move to use ROS 2 as our middleware consist of:

Creating hardware specific interfaces Our robots are based on the Robotis CM-730 sub-controller. Robotis has released ROS 1 packages for their products, but at the moment there is no ROS 2 effort. We have created and are finalising a ROS 2 driver for the CM-730, which we will release in the near future. Figure 6 (right) shows the result of the robot model built using the output of this driver.

Porting ROS 1 packages Only a subset of existing ROS packages has been ported to ROS 2. We are porting several missing packages, such as a USB camera driver, which we will contribute upstream (Sec. 6). Figure 6 (left) shows the output of the camera driver.

Porting our modules Once all hardware interfaces are in place, we can port our existing modules over to the new platform.

6 Research and Open-Source Contributions 2018

We presented our recent vision research at the RoboCup symposium 2018, it was "Deep Learning for Semantic Segmentation on Minimal Hardware" [6] and it is briefly described in section 4. We also published the related annotated image sets created and used during the RoboCup 2018 competition in Montreal:

- goal-posts: https://imagetagger.bit-bots.de/images/imageset/233
- ball: https://imagetagger.bit-bots.de/images/imageset/12/

The following is a list of open-source contributions:

- SCAD models for Dynamixel's MX-28, MX-64 and MX-106 servos, horns and bearings: https://gitlab.com/boldhearts/dynamixel-scad
- porting the ROS 1 usb_cam package to ROS 2: https://github.com/ros-drivers/usb_cam/pull/106
- a simple, standalone 2D simulator: http://gitlab.com/boldhearts/pythocup

7 Acknowledgements

Team Bold Hearts would like to acknowledge the crucial open source projects used to develop our team: ROS 2, OpenSCAD, GitLab, TensorFlow and Bit-Bots Imagetagger[8].

References

1. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.
2. Trent Houliston, Jake Fountain, Yuqing Lin, Alexandre Mendes, Mitchell Metcalfe, Josiah Walker, and Stephan K. Chalup. Nuclear: A loosely coupled software architecture for humanoid robot systems. *Frontiers in Robotics and AI*, 3:20, 2016.
3. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
4. Marcus M. Scheunemann, Sander G. van Dijk, Alessandra Rossi, Daniel Barry, and Daniel Polani. Bold Hearts Team Description RoboCup 2018 Kid Size. techreport, School of Computer Science, University of Hertfordshire, College Lane, AL10 9AB, UK, December 2018.
5. Sander van Dijk, Drew Noakes, Daniel Barry, and Daniel Polani. Bold Hearts Team Description RoboCup 2014 Kid Size, 2014.
6. Sander G. van Dijk and Marcus M. Scheunemann. Deep Learning for Semantic Segmentation on Minimal Hardware, 2018. To appear in RoboCup 2018: Robot World Cup XXII, Online: https://arxiv.org/abs/1807.05597.

[8] https://www.openscad.org, https://gitlab.com/, https://www.tensorflow.org/, https://imagetagger.bit-bots.de