RedbackBots Team Report RoboCup Soccer SPL 2023

Timothy Wiley, John Thangarajah, Jasper Avice Demay, Tom Ellis, Samuel Griffiths, David Hermanto, Adhiraj Jain, Lewis Matthews, Luke Mullan, Murray Owens, Kah Hie Toh, Ryan Samarakoon, James Sun

Artificial Intelligence Innovation Lab,
School of Computing Technologies, STEM College,
RMIT University, 124 La Trobe St, Melbourne, Victoria, Australia.
w: http://www.rmit.edu.au/ailab e: stem.redbackbots@rmit.edu.au

1. Introduction

RedbackBots competed in our first on-stie competition for the RoboCup Soccer Standard Platform League (SPL) in the 2023 international competition held at Bordeaux, France. We placed 5th in the Challenger shield. Our team is part of the AI Innovation Lab at RMIT University aims to develop and extend solutions to further and enhance human capabilities, improving our quality of life using artificial intelligence. We see RoboCup Soccer SPL as an excellent platform for research and innovation, and as an education platform for students in RMIT University to learn and apply skills in the field of Artificial Intelligence.

The 2023 was our first full attendance in the RoboCup Soccer SPL, and our first experience in international 5v5 Soccer competition. We previously competed remotely in 2022 in the technical challenges. Our goal for 2023 was to put forward a competitive team and codebase and score goals in a competitive game. We have exceeded our expectations, by narrowly missing out on the Challenger Shield semi-finals. We also strongly support the research aims of the Soccer SPL in each team bringing novel developments. Our codebase in derived from the rUNSWift 2019/2020 code release (UNSW Computing 2019). In 2023, aside from developing our own Soccer playing behaviours, we have continued our development from 2022 in the Visual Referee Challenge, our own whistle detection, and porting the rUNSWift codebase into bootable Linux image structure developed by the B-Human and Nao Devils teams (BHuman 2022). This year's competition has provided invaluable experience to our team, and we strive to return in 2024.

2. Team Organisation

RedbackBots was first established in 2019 and competed for the first time in 2022 after a long series COVID lockdowns in our city of Melbourne. Our is formed of Undergraduate and Postgraduate Students from Computing and Engineering programs at RMIT University. The team is supervised by Dr. Timothy Wiley and Prof. John Thangarajah who have been involved in RoboCup for 15 years. Our team is coordinated by final year 'Capstone' students who lead the administration and development work of the team as part of their final semester major industry linked project course. This means that our student's keep a strong ownership over the direction and running of the RedbackBots team.

Our 2023 consisted of 4 students (Jasper Avice Demay, Samuel Griffiths, Luke Mullan, and Ryan Samarakoon), and 1 staff member (Timothy Wiley). The team was also remotely supported by 5 students remotely in Melbourne (Tom Ellis, Adhiraj Jain, Lewis Matthews, Murray Owens, and Kah Hie Toh).

Figure 1: 2023 RedbackBots Travelling Team (Bordeaux, France)

RedbackBot's participation in RoboCup Soccer SPL is summarised below.

Year	**Participation**	**Results**
2023	RoboCup23 Challenger Shield	5th Place
2022	Technical Challenges	8th Place Overall 3rd Place Visual Referee Challenge
2021	Qualified	Withdrew due to COVID-19 travel restrictions
2020	Qualified	Competition cancelled due to COVID-19 pandemic

The RedbackBots team convenes weekly where members collectively engage in codebase development, organisation, and hold regular practice matches. Concurrently, team members who are attend remotely via MS Teams. Our team leverages Discord as a communication platform, fostering an environment where members can both seek assistance and collaborate effectively. We maintain our codebase on internal Bitbucket repositories. We have made our 2023 code release available through the RMIT GitHub Organisation.

3. Codebase

The codebase for our 2023 code release consists of our own contributions build on top of the previous work of RoboCup SPL teams. We acknowledge the following code-use as part of the development of our RedbackBots software:

- Primarily we have used the rUNSWift 2019/2020 code release as the basis of our codebase for the software architecture, locomotion, vision, and communication modules (UNSW Computing 2019).
- We have incorporated into our codebase updated Machine Learning models from the

rUNSWift 2022 code release (UNSW Computing 2022).

- The B-Human (and Nao Devils) 2022 code release, for the structure and creation of a bootable Nao V6 Ubuntu 20.04 image, along with general build configuration scripts, and general robot configuration management scripts (BHuman 2022).
- The B-Human 2022 code release for the Nao V6 camera driver compatible with the Nao V6 Ubuntu 20.04 image (BHuman 2022).

Our codebase is divided into the following modules, derived from rUNSWift:

- Motion, implemented in C++, that includes an integrated walk engine and kick engine.
- Perception, that includes Vision for all object recognition tasks, and audio detection primarily used for whistle detection.
- Localisation, that provides the state estimation of the robot including it's on-field position and a shared-ball estimation.
- Behaviour, that provides the decision making and action selection of the soccer playing software.
- Infrastructure, that provides data structures and software tools for supporting the other modules, including a C++/Python interface in Boost, and libraries for loading Machine Learnt models, and build-chain tools. The module includes our

In 2023, we have made contributions to:

- Perception, in our approach to the Visual Referee Challenge.
- Perception, in our approach for Whistle Detection.
- Infrastructure, in our bootable image of the 'rUNSWift-derived' codebase.
- Behaviours, in our developed of soccer playing capable code.

We describe our contributions in this team report, and refer to the published work of rUNSWift, B-Human and Nao Devils for the work on which we depend.

3.1. RedbackBots 2023 Code Release

Our code release for 2023 is available on the RMIT University GitHub Organisation at:

https://github.com/rmit-computing-technologies/redbackbots-coderelease

The 2023 code release is located under the tag "coderelease2023".

4. Perception

Our contributions within Perception have been focused on continuing our work from the 2022 in the Visual Referee Challenge, and our work on Whistle Detection.

4.1. Visual Referee Challenge

We have extended upon our participation in the 2022 Visual Referee Challenge (Lohani & Wiley 2022; Lohani 2022). Our previous approach leverages a hybrid symbolic and neural network methodology, shown in Figure 2 along with an illustration of each stage in Figure 3. First, we down-sample and pre-process the camera image. Next, we use two ``light-weight'' machine learning algorithms to locate bounding boxes for the face and hands of the human referee. From these bounding boxes, we compute various heuristic symbolic measurements for properties of the human pose, such as the angles between the human referee's gloved hands and face, which are passed through a rule set to determine the referee signal. Our approach trades single-image accuracy for processing speed with a video sequence in a methodology that approaches the accuracy of the slower single-image methods. We have evaluated our method on both single image instances, and on a video sequence of images, which, published in Lohani & Wiley 2022. A key result is that across a video frame sequence of 25 seconds length containing 3 separate referee signals, our hybrid methods require an accumulative 1.5 second of processing time on the Nao V6. This is compared to two Neural Network based approaches Stacked Hourglass (Newell *et. al.* 2016) and OpenPose (Cao *et.al.* 2019), which require an average of 5.27 and 7.30 seconds is required to class each of the three signals. This demonstrated that the hybrid approach provided a beneficial trade-off between accuracy and speed of operation which is an important consideration in RoboCup Soccer games where the processing capacity of the Nao is at a premium.

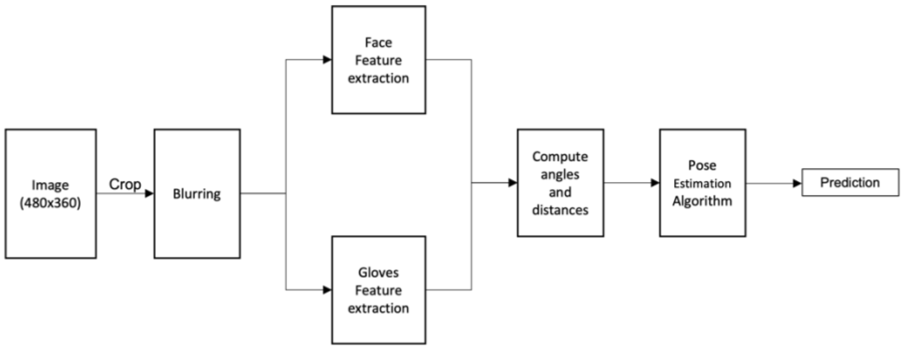
Image (480x360)
Crop
Blurring
Face Feature extraction
Gloves Feature extraction
Compute angles and distances
Pose Estimation Algorithm
Prediction

Figure 2: Architecture of our hybrid symbolic and ``light-weight'' machine learning method

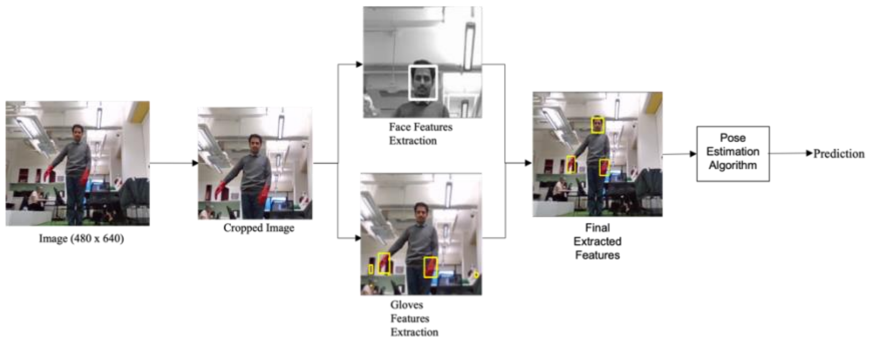

Figure 3: Visual illustration of our hybrid symbolic and ``light-weight'' machine learning method

In 2023, we have transitioned to the use of lightweight Recurrent Neural Networks (RNN) to recognise dynamic signals in the Visual Referee Challenge (Owens 2023). First, we use colour segmentation to locate the position of the referee's gloves. We find the minimum and maximum position of the gloves along the horizontal axis and mask the sections of the image outside this area. This isolates the referee in the image, reducing the overall amount of background noise. Next, we use the lightweight CNN pose detector Blazepose Lite (Bazarevsky et. Al. 2020) to quickly extract skeletal-keypoint features from the referee. To allow for real-time detection on low-powered systems, Blazepose Lite trades-off accuracy for quicker processing speed. Finally, we use a lightweight RNN to recognize the referee gesture. We collect the referee features for 20 frames, sampled over 4-5 second to ensure we obtain enough information to classify the dynamic signals. We developed two RNN architectures: a 2-layer LSTM (Hochreiter & Schmidhuber 1997) with 30 units each, and a 2-layer SimpleRNN with 128 and 64 units respectively. To compensate for the reduced complexity, our 30-unit LSTM also uses the angles of the shoulder, elbow and wrist to better recognize the goal and pushing free kick gestures. Real-time testing shows our approach has a strong performance, with both architectures having an accuracy of 93%. The average time needed to classify the gestures is 5.23s. We trade-off the quick speed of the heuristic method for increased accuracy and the ability to accurately classify the dynamic gestures. Our approach is capable of quickly locating and classifying the gestures from many different locations on the field, allowing it to be integrated into an official RoboCup SPL match.

4.2. Whistle Detection

For whistle detection we have applied a Fourier Transform algorithm as frequency peaks in audio signals from a whistle are an effective method of recognising a whistle. Short Time Fourier Transform (STFT) analyses the frequency and phase content of signal over time by dividing the signal into short overlapping segments (Allen 1977). We have chosen to use STFT as the short overlapping segments can better identify the time varying frequency content. The frequency and phase content provide distinct frequency peaks and patterns that the machine learning model can use to identify whistles. We have implemented STFT in both C++ and Python. We have leverage the Scipy library in Python to conduct the initial training of the Machine Learnt model, while we leverage C++ to execute the trained model.

However, our STFT approach had challenges with false positives from other noises such as talking. This is problematic as the RoboCup fields have significant background noise. We have used a Band Pass Filter (BPF). The BPF isolates specific signals at set frequencies which can help to remove noise and further highlight the high frequency peaks from the whistle. Unlike other filtering and smoothing techniques, the BPF retains time variations in the signal which means important whistle characteristics are not lost.

The output of the filters is used to determine regions of a whistle from the audio signals.

To test and adjust our filters, we collected consists of 200 audio samples, with 50 audio samples for 0, 1, 2 and 3 whistles blow respectively. Each sample is recorded from the NAO robot’s microphone with a duration of 3 seconds and sample rate of 8000. To increase variability in the dataset, 2 different whistles are used when recording the audio, the first whistle being of plastic material and the other being a metal whistle. We have also simulated background crowd noise, using a recording created by FootballViewHD (2021) that is played during data collection.

5. Behaviours

We have implemented Python Based Behaviours encompassing various roles such as Striker, Defender, Goalkeeper, and more. These roles will define how the robot acts at any given situation during the match by deciding what skill to use. There are various skills implemented, from lining up and kicking the ball to moving to defensive positions.

5.1. Striker

Striker is a dynamically selected role, implemented for any robot that is designated to kick a goal. Whilst striker is a non-static role, only one robot is active in this role at any given point in time. The robot selected as striker is determined by the closest, non-goalie robot to the ball. To avoid role oscillations, the striker role is set for a minimum time and after this has lapsed it is able to switch out.

5.2. Defender

Defender is the role designated to guard the defensive half of the field. The positioning of the defender is focused on the penalty box. Defenders will aim to remain in the path from the centre of the goal to the ball, to intercept any possible attempts at a goal. The defender will track the ball with horizontal movement, until the ball enters their designated zone, when it will start to directly intercept the ball.

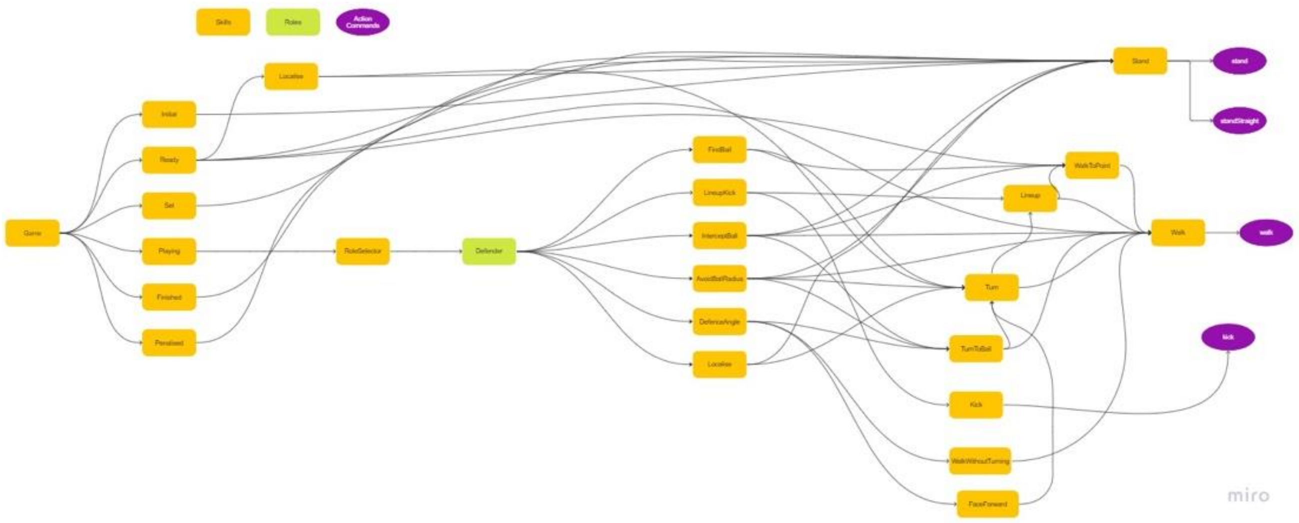

Figure 4: Defender Behaviour Tree

5.3. Goalkeeper

The current implementation of Goalkeeper is a sentry like role mainly tasked with staying in the goal box and tracking the ball, similarly to defender. Unlike defender, the goalie will not approach the ball as readily, instead only when the ball enters the penalty box. Upon the decision to intercept the ball, the Goalkeeper will determine if a dive is necessary, regarding the ball velocity. If a dive is not necessary, the Goalkeeper will make it's ground footprint as wide as possible, increasing any chance of blocking an attempt at the goal.

5.4. Supporter

Supporter's role in the team is to supplement the striker. They patrol the attacking half of the field, when they lock on to the ball, they follow at a reasonable distance allowing for the striker to approach and kick the ball. If the supporter is closer than the striker and the striker's minimum role time has elapsed it will swap to striker and communicate this swap, leading the previous striker to become the supporter. Supporter also aims to keep the ball in the attacking half, attempting to intercept any downfield kicks.

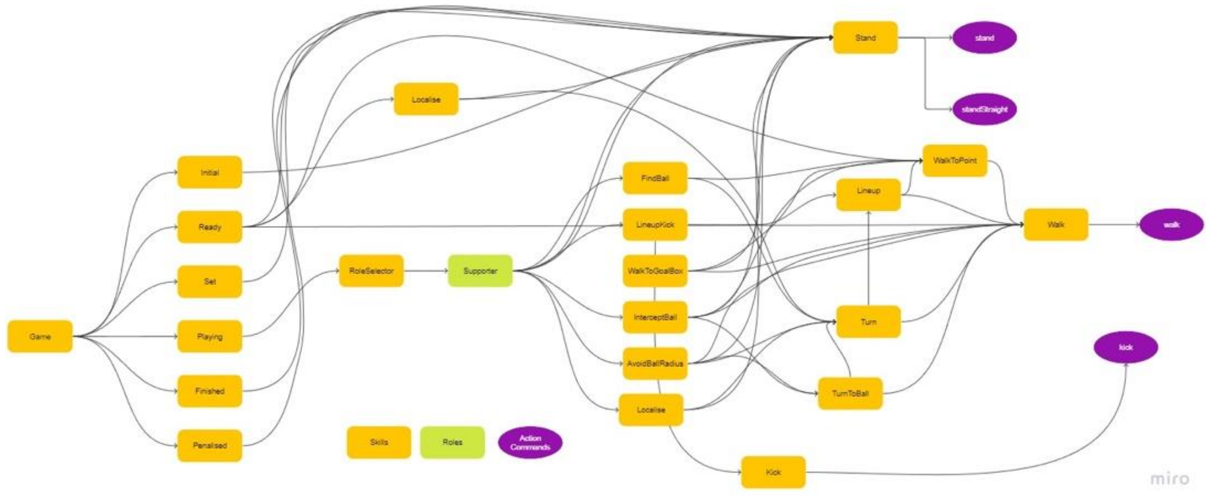

Figure 5: Supporter Behaviour Tree

5.5. Midfielder:

Midfielder is a similar role to that of defender, except it is allowed to encroach more on the enemy goals to better support the striking half. Midfielder acts as a hybrid of both supporter and defender.

6. Software Infrastructure

During the COVID-19 pandemic periods, the Soccer SPL became more invested in the ability to deploy a team's code remotely more easily to another robot. One approach developed by the B-Human and Nao Devils teams (BHuman 2022) investigated are bootable USB images, that enable a robot to be "flashed" with a complete operating system which includes Soccer SPL capable software. This enables a newly flashed robot to be placed onto a field and join a Soccer SPL game without any additional configuration.

We have ported our codebase to the buildable Nao V6 Ubuntu 20.04 image developed by teams including B-Human and Nao Devils. As our codebase is based on the rUNSWift 2019/2020 code release, this enables another independent codebase to be deployed as an independent image following the same structure as other teams. This increases the diversity of different codebases available in the SPL.

The structure of our updated infrastructure shared similarities with both the B-Human structure and the rUNSWift infrastructure. The root level structure is described in the table.

Folder	**Purpose**
Build	The location where C++ build files, and the bootable Image is placed.
Config	Software and robot configuration files for the RedbackBots team. In our code release, configuration files have been replaced with default values.
Install	Software and scripts for installing the RedbackBots software on a Nao robot. Static files that are deployed to the Nao. These include operating system configurations, static robot configuration files for use in the RedbackBots software, audio files, and machine learnt models.
Make	Software and scripts for building the RedbackBots software, primarily consisting of CMake configuration files. Also includes scripts for generating the bootable USB image.
Src	C++ and Python source files of the RedbackBots codebase.
Util	Utility files, at present only containing pre-compiled libraries for various operating systems architectures, including x86_64 and arm64.

Contained within the SRC folder, noted above, is the bulk implementation of our soccer capable code, divided into subdirectories, summarises in the table.

SRC Folder	**Purpose**
behaviour	All python files/modules that implement our robot behaviours implemented in a python sub-system called from within the C++ code. The entry point into the Python sub-system is the behaviour.py file.
boost_test	Standalone C++ program for testing Boost functionality on the Nao robot and linked against the provided Boost Buildchain. By default, this program is not compiled, but can be enabled within the CMake configuration files.
lola_test	Standalone C++ program for testing LOLA connectivity on the Nao robot. By

	default, this program is not compiled, but can be enabled within the CMake configuration files.
offnao	Standalone local program, written in C++, for connecting to the RedbackBots software running on a Nao robot and observing the robot state. Currently offnao supports viewing of processed camera images, vision features, localisation, shared ball information, and the Blackboard state.
robot	C++ source files for the robot soccer code, that is cross-compiled and deployed onto the Nao robot.

Further information about the software infrastructure is documented within our code release.

Acknowledgements

RedbackBots acknowledges the support of RMIT University has made a significant investment in Artificial Intelligence and Robotics. The Artificial Innovation Intelligence Lab was launched in late 2019, and our RedbackBots RoboCup team is a key element of the lab strategic direction. RMIT University also launched the Centre for Industrial Artificial Intelligence and Research Innovation (CIAIRI) in late 2021 with a focus on AI applications for industry which has provided the funding support for our RedbackBots team since 2019, including our 2023 participation.

RedbackBots also acknowledges the large number of teams who have contributed to the various codebases of which our team has developed our code, and the resources provided by the RoboCup Soccer SPL.

References

Allen, J. (1977), *Short term spectral analysis, synthesis, and modification by discrete Fourier transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, no. 3, pp. 235-238.

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang F. and Grundmann, M. (2020). *BlazePose: On-device Real-time Body Pose tracking*. https://arxiv.org/abs/2006.10204.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A. (2019). *Openpose: Real-time multi-person 2d pose estimation using part affinity fields*. IEEE Transactions on Pattern Analysis and Machine Intelligence. pp. 72-91.

BHuman (2022). *BHumanCodeRelease, Tag: coderelease2022*. GitHub repository. https://github.com/bhuman/BHumanCodeRelease/releases/tag/coderelease2022. Last accessed October 2023.

FootballViewHD (2021). *10 Hours Of Stadium Crowd Sound | Stadium Crowd Noise Sound Effect | Cheering Crowd*. Youtube. https://www.youtube.com/watch?v=Nsi3zUriYgc. Last accessed October 2023.

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation. 9(8) pp. 1735-1780

Lohani, P. (2022). *Visual Referee Challenge for RoboCup Soccer*. Master's Minor Thesis.

Lohani, P. & Wiley, T (2022). *Hybrid Methods for Real-time Video Sequence Identification of Human Soccer Referee Signals*. Proceedings of the RoboCup Symposium (to appear).

Melbourne, Australia: School of Computing Technologies, RMIT University.

Newell, A., Yang, K., Deng, J. (2016). *Stacked hourglass networks for human pose estimation*. European conference on computer vision. pp. 483-499.

Owens, M. (2023). Visual Referee Signals for RoboCup Standard Platform League. Master's Minor Thesis (to appear).

UNSW Computing (2019). *rUNSWift-2019-release*. GitHub repository. https://github.com/UNSWComputing/rUNSWift-2019-release. Last accessed October 2023.

UNSW Computing (2022). *rUNSWift-2022-release*. GitHub repository. https://github.com/UNSWComputing/rUNSWift-2022-release. Last accessed October 2023.