RedbackBots Team Report
RoboCup Soccer SPL 2024

Timothy Wiley, John Thangarajah, Rithvik Abothu, Jasper Avice Demay, Hirday Bajaj, Hin Yeung Samuel Chan, Mason Cao, Tom Ellis, Mark Field, Roohollah Ganji, Samuel Griffiths, Christopher Lamb, Prawal Lohani, Luke Mullan, Murray Owens, Jed Pauckner, Rafael Perez, Mark Putter, Arturo Sandoval Rodriguez, Joel Stephens, James Thom.

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 second on-site competition for the RoboCup Soccer Standard Platform League (SPL) in the 2024 international competition held at Eindhoven, The Netherlands. We placed 3rd in the Challenge Shield. Our team is part of the AI Innovation Lab at RMIT University which aims to develop and extend solutions to further 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.

In 2024, we improved upon our 2023 results, placing 3rd in the Challenge Shield and exceeded our pre-comp expectations knowing that RoboCup SPL is so competitive. Our codebase, originally derived from rUNSWift's 2020 code release (UNSW Computing 2019), continues to diverge significantly, with a revised behaviour framework, Python 3 integration, and a bootable Linux image structure we have ported from the B-Human and Nao Devils teams (BHuman 2022).

2. Team Organisation

RedbackBots was established in 2019. Our team 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 Work Integrated Learning course. Thus, our student's retain a strong ownership over the direction and running of the RedbackBot's team.

Our 2024 travelling team consisted of 5 students (Murray Owens, Samuel Griffiths, Tom Ellis, Mark Field and Jasper Avice Demay), and 1 staff member (Timothy Wiley). The team was supported remotely in Melbourne by 2 students (Prawal Lohani and Arturo Sandoval Rodriguez) and a recent graduate (Luke Mullan).

Figure 1: 2024 RedbackBots Travelling Team (Eindhoven, Netherlands)

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

Year	Participation	Results
2024	RoboCup 2024 Challenger Shield	3rd Place
2023	RoboCup 2023 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. Our team leverages Discord as a communication platform, fostering an environment where members can both seek assistance and collaborate effectively. We maintain our codebase using Bitbucket. We have made our 2024 code release available through the RMIT GitHub Organisation.

3. Codebase

The codebase for our 2024 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 the rUNSWift 2019/2020 code release is the original 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:

- 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 its 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.

In 2024, we have made contributions to:

- Perception:
  - In our approach to the Visual Referee Challenge.
  - In our approach for Whistle Detection.
- Behaviours:
  - In our dynamic role & position selectors.
- Behaviours:
  - In “Mine” calls for team communication.
- Camera calibration:
  - To improve localisation consistency.
- Infrastructure:
  - Updating of internal Blackboard representations.
  - Updating Offnao.
  - In moving our supported OS build to Ubuntu 22.04 on x86 and Arm64.
  - In moving the behaviour framework to Python 3.

Additionally, in 2024 we have developed a unique Augmented Reality (AR) visualisation system, GameSight, for the MetaQuest device. GameSight enables visualising GameController packets and the information available to the Team Communication Monitor in real-time projected in AR onto the field with full colour feed-through rendering.

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 2024 Code Release

Our code release for 2024 is available on the RMIT University GitHub Organisation at:

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

The 2024 code release is located under the tag “coderelease2024”.

3.2. GameSight 2024 Code Release

We have released our GameSight AR visualisation tool for 2024 on the RMIT University GitHub Organisation at:

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

The 2024 version of the code release is located under the tag “coderelease2024”.

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 2023; 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 lightweight 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. 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.

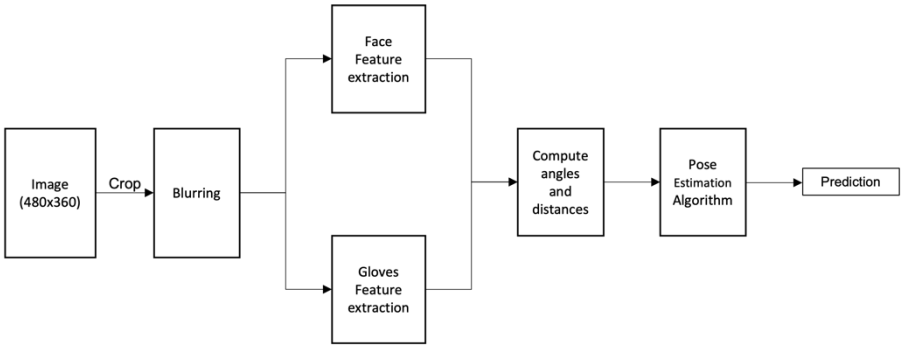

Figure 2: Architecture of our hybrid symbolic and ``light-weight'' machine learning method

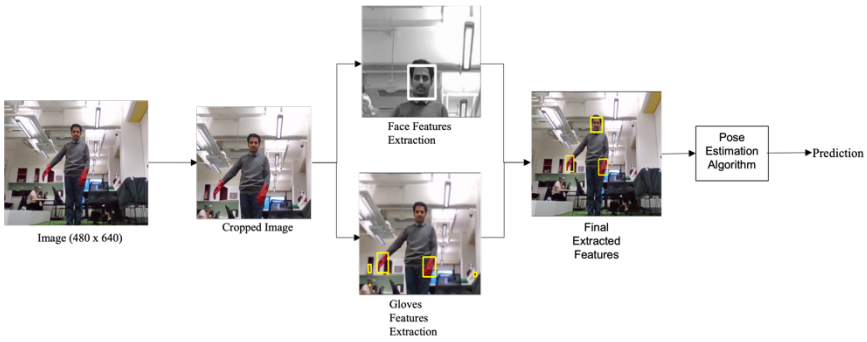

Figure 3: Visual illustration of our hybrid symbolic and ``light-weight'' machine learning method

In 2023, we 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 RNN 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 reimplemented a standalone Python-based whistle detection algorithm. The detector uses 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). These segments are then subjected to a series of tests to determine whether they match a pre-calibrated spectrum established through manual tuning. If a sufficient number of tests pass consecutively the algorithm concludes that a whistle has been detected.

The primary challenge with our current whistle detector is achieving accurate calibration. When the detector is not properly calibrated, the likelihood of false positives or false negatives significantly increases. Additionally, the algorithm is sensitive to changes in background noise levels, which can significantly impact its reliability. This poses a problem in environments where background noise levels are higher than normal and continually fluctuate, like the RoboCup SPL fields during competition. To mitigate this, we have incorporated 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.

At RoboCup 2024, the constantly changing background noise caused significant issues with our current whistle detector. Calibration took too long to complete before each game, forcing us to rely on predefined settings that were often in limbo between either being too sensitive or not sensitive enough. Consequently, after being inspired by a presentation at the RoboCup symposium from the Nao-Devils SPL team, we are now working on developing a self-calibrating neural-network based whistle detector. This new detector will be implemented in C++ to improve performance and allow seamless integration into our existing codebase.

5. Behaviours

We have separated our field positions from our roles. Any robot can go into any role (e.g. striker, defence) from any assigned position.

5.1. Position Selector

We dynamically assign each robot's position depending on the active status of other robots. The field is divided into four quadrants: two attacking and two defending. If all robots are active, they are each assigned one quadrant. If a robot becomes inactive, the neighbouring robot will expand its zone to cover the inactive robot's quadrant. When only one robot is active, it is assigned the special position of 'Superstar', whose quadrant covers the entire field.

5.2. Role Selector

We also dynamically assign roles to each robot. Robots will enter the striker role if the ball is inside its quadrant, or it is a half centre circle distance away from the ball. We use a scoring system (described in 5.3) to ensure that only one robot goes into striker at a given time. Non-striker robots are either assigned to defence or standby, depending on whether the ball is in the attacking or defending half. Defending robots will move between the ball and the goal to intercept it. When the team has lost sight of the ball for a certain amount of time, each robot will return to its default position and turn to search for the ball.

5.3. "Mine" calls – Team Communication

We implemented a coordinated strategy where each robot calculates its relative "ability to kick the ball," known as the "ball score." This score reflects how well-aligned, close, and visible the ball is to the robot.

The robot with the lowest ball score sends a packet to confirm to the team that it is the best candidate to kick the ball. This approach minimises packet use, as robots do not need to send "heartbeat" packets at regular intervals. Instead, packets are sent only when a robot believes it is the new best kicker or when other robots overestimate its ability (e.g., after the ball is kicked away).

When comparing ball scores, robots' factor in the time since the last packet was sent, adding a set amount for each second elapsed. For example, if a packet was sent four seconds ago, the score is adjusted accordingly. If a robot decides it is superior and sends a packet, but another robot remains the best by a margin, it quickly sends another packet to reaffirm its status. This method reduces overall packet usage, especially when a single robot is alone with the ball, mimicking the "mine" call in a real soccer game with real-time play negotiation.

6. Camera Calibration

Camera calibration proved essential for our robots' vision systems in the inconsistent lighting conditions at RoboCup 2024. Before every game and on each field, we performed calibrations to ensure the robots could localise on the field while identifying the ball. We sometimes had to compromise line perception to prioritise ball detection when shadows caused blending issues.

7. Software Infrastructure

Our infrastructure is originally derived from rUNSWift and we have incorporated B-Human's build system. The root level structure of our infrastructure is described below.

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.
Docs	Contains some pages from our internal team wiki. It is stripped down to include only steps on getting our code running on a robot.
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.

8. League Contributions

8.1. League Tools

RedbackBots developed an Augmented Reality (AR) visualiser, GameSight, compatible with TCM and GC. This enables visualisation of localised game information through AR platforms including the Meta Quest. This was demonstrated at the 2024 competition, is available for teams to download and use themself and will form part of our code release. We plan to continue developing this platform to enable a more diverse range of toolsets that can be used league wide and to explore unique human robot interaction research.

8.2. Committee Roles

For the 2024 season, Tom Ellis represented RedbackBots by serving as the Chair of the RoboCup SPL 2024 Organising Committee where he contributed to the organisation and running of the SPL tournament.

9. Research Publications and Articles

RedbackBots supports the research initiatives of the SPL and views our league as an important avenue of research in autonomous robotics. We also support the active public promotion of RoboCup and the SPL. Our team publications and articles are below.

9.1. Thesis Publications

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

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

9.2. Research Papers

Owens, M. & Wiley, T. (2024). *Lightweight Real-Time Gesture Recognition for Dynamic Soccer Referee Signals*. Proceedings of the RoboCup Symposium (to appear).

Lohani, P. & Wiley, T (2023). *Hybrid Methods for Real-time Video Sequence Identification of Human Soccer Referee Signals*. Proceedings of the RoboCup Symposium

9.3. Media Articles

AIHub. *Feature in: RoboCup2024 – daily digest: 21 July*. (2024) RedbackBots, were featured on the AIHub's final day digest for RoboCup 2024 for our innovative AR live visualisation system of RoboCup games - GameSight.

Acknowledgements

RedbackBots acknowledges the support of RMIT University, along with the School of Computing Technologies in the STEM College for their significant investment in Artificial

Intelligence and Robotics. The Artificial Innovation Intelligence Lab, under which our team resides, was launched in late 2019. Our RedbackBots RoboCup team is a key part of the lab's 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 2024 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.

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

BHuman (2023). *BHumanCodeRelease, Tag: coderelease2023*. GitHub repository. https://github.com/bhuman/BHumanCodeRelease/releases/tag/coderelease2023. Last accessed October 2024.

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.

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

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

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.