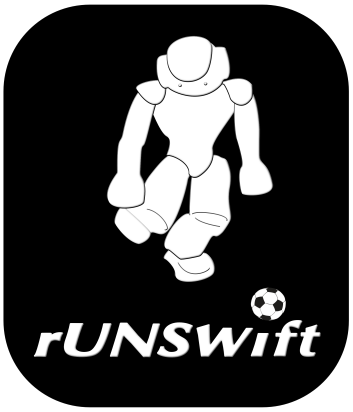

Team Description Paper 2025

Joshua Chan, Reilly Fox, Quoc Tuan Pham, Stefan Immaraj, Zijie Li, Mikhail Asavkin, Claude Sammut, Peter Schmidt

School of Computer Science and Engineering, The University of New South Wales, Sydney, New South Wales, 2052

1 Team Information

Team Name:	rUNSWift
Team Leader:	Claude Sammut, Joshua Chan
Email Address:	robocup.spl@cse.unsw.edu.au
Team Website:	https://robotics.cse.unsw.edu.au/runswift/
Country of Origin:	Australia
University affiliation:	UNSW Sydney

2 Code Usage

By transitioning to ROS2 Humble, all of our dependencies have been changed substantially.

The general approach to the vision system remains the same in 2025 as it was in 2017-19 [4], [5], [3]. However, the code was completely rewritten from the ground up to meet our needs for the ROS2 transition. We used OpenCV C++ for image processing and the ROS2 usb_cam node to handle camera drivers.

Localisation and state estimation retain the same algorithm as in 2018 [5] and 2019 [3], but the code was partially modified when ported to the ROS2 architecture.

The rUNSWift motion package is based on Hengst's walk generator [6] developed in-house and used since the 2014 RoboCup competition. It was further developed within the team to be used in junction with the nao_lola_client ROS2 package.

By leveraging rviz2 and rqt, the tooling has changed significantly compared to our 2019-2024 codebases [3].

It is also worth mentioning that parts of the codebase have been linted or generated using Generative AI tools such as Copilot, Claude 3.5 and ChatGPT-4o.

3 Own Contribution

Although rUNSWift retains a research focus, our efforts have lately been on transitioning to ROS2 architecture, tidying our codebase and rewriting major modules. These changes better position us to test ideas within a modular codebase, following industry trends in ROS.

ROS2 Humble's modularity gives us a solid base to continue our investigations into improving our computer vision capabilities, training a walk with reinforcement learning and sensor-informed predictive maintenance - all of these build towards future contributions.

3.1 Architecture

Our experience at Eindhoven 2024 demonstrate that our codebase suffered from several issues, but coupling and synchronisation seemed to be large factors preventing the team from running adequate integration tests that may have assisted us.

By restructuring our architecture around packages, each containing nodes, we were able to take advantage of ROS2's modularity. Moreover, ROS2 provides message headers that include timestamp data and a queue buffer to coordinate nodes, as well as a protocol for intra-process communication.

This all said, the structure of each package needed to be optimised so as to balance between our needs for modularity and a minimal ROS2 overhead, whilst adequately adjusting for concurrency issues.

For example, our multi-stage vision pipeline is roughly linear - first we read the image, then we preprocess the image, etc. This means that for the sake of modularity, we can comfortably treat these nodes as independent steps in a sequence, passing the image frame between nodes through a shared message.

On the other hand, motion control modules require greater consideration of concurrency because the control module needs to be able to address multiple incoming motion commands with precise timing. As such, we treat the control node using ROS2 components, which allow for multiprocessing similar to ROS nodelets.

3.2 Vision

Currently, the team has transitioned to ROS2 using a multistage vision pipeline with a mixture of ML-based and algorithm-based hueristic checks. This follows the rough structure of the previous codebases.

We first began by calling a webcam topic created by the usb_cam node, intentionally downsampled to a lower resolution so that we do not sacrifice processing time resizing images.

Afterwards, we use OpenCV's adaptive thresholding function to generate a binary image. Next, we apply Connected Component Analysis (CCA) to segment the image into distinct 'islands'. To ensure consistent processing, we impose constraints on bounding box sizes, splitting large connected components, and discarding small noise. Finally, stray noise is eliminated by masking out disconnected components based on the CCA, meaning that some regions are purely noise, whereas others are practically noiseless.

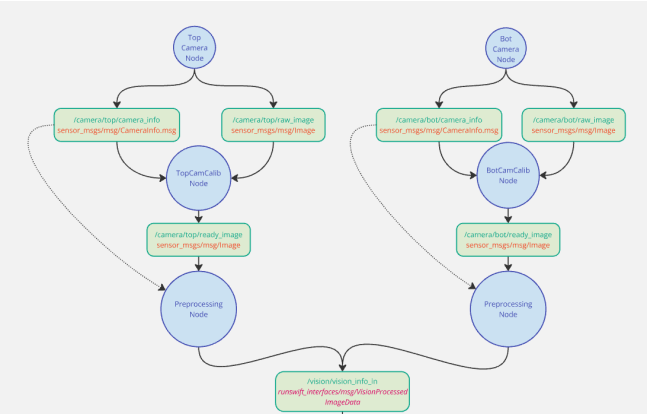

Fig. 1. Image preprocessing ROS2 architectural diagram

These 'regions of interest' (ROIs) are passed through to the two detectors for future analysis, represented both as bounding box coordinates and as a cropped image, masked according to the binarised CCA's value.

The field feature detector begins by performing heuristic checks, relying on the 'blob edge' functionalities maintained from the legacy code. These blobs are identified by scanning the borders of binary images for continuous white segments, with different combinations (2 or 3 blobs) potentially indicating different field features like lines, curves, corners, or T-junctions. These candidates are then passed to a neural network classifier, which was trained using binary image data on a manually-labelled dataset of 'regions of interest' that had undergone the relevant heuristics. The training data was curated using a custom GUI tool that categorised potential features as 'Lines', 'Curves', 'T-junctions', 'Crosses', 'L-Corners', and 'None'. The system combines traditional techniques with machine

learning in a client-server architecture, allowing the system to use heuristics for classifying features and leveraging the neural network to verify.

Rather than just using the top camera, the ball detector uses both the top and bottom cameras, relying on a CNN trained with greyscale images rather than binary ones.

First, it divides the bottom camera image into six regions and the top camera image using the ROIs created by the preprocessing node.

The top camera ROIs are initially filtered using Hough circles to identify ball candidates, reducing false positives, and then resized to 32x32 pixels. The bottom camera begins with a downsampled input and does not undergo any heuristics.

Our ball detector is based on a greyscale data set sourced from the B-Human ball and penalty cross data set [1]. Using this data set we trained a CNN-based classifier (using transfer learning methods) that outputs three confidence scores per region, representing ball, penalty cross, and ambiguous cases. The system also includes post-processing to combine detections when two cameras spot the same ball, and coordinate-transforms to convert pixel coordinates to real-world coordinates relative to the robot's reference frame (this was achieved by migrating the kinematics module into its own package, decoupled from motion). This approach brings the ball detection range up to 7.4 meters while maintaining real-time performance.

We are further developing experimental approaches that would allow us to train the entire vision pipeline using a single ML model, rather than passing preprocessed regions of interest to multiple classifiers. However, this requires us to build tools for capturing autolabeled data during game-like situations. In the 2022-2024 period we had been working on tools to enable the data capture and auto-labeling for future training, and with ROS2 tools available to us, we have also developed tools to convert these dump files into rosbags.

4 Past History

Team *rUNSWift* has been competing in the Standard Platform League (SPL) since 1999. Every year, we strive to improve our system and adapt it to the new rules and challenges conferred by the SPL technical committee.

The 2024 RoboCup SPL competition was a wake up call for rUNSWift, reminding us that to be serious competitors within the evolving world championship requires consistent progress and a testing cycle beginning more than just a few months prior. Accordingly, this experience has predisposed every single member in our team to subject themselves to rigorous training and development over the course of the year leading up to the championship.

This year, we only reached the quarter-finals [2], which was a step down from our third-place finishes in Bordeaux 2023, Thailand 2022 and Sydney 2019 [1].

[1] https://b-human.informatik.uni-bremen.de/public/datasets/BallAndPenaltyMarkPerceptor/

[2] https://spl.robocup.org/results-2024/

Fig. 2. The rUNSWift team at RoboCup 2024 in Eindhoven, Netherlands. *From left to right:* Mikhail Asavkin, Reilly Fox, Quoc Tuan Pham, Stefan Immaraj, Peter Schmidt, Joshua Chan, Ryan Samarakoon

Although disappointing, our 2024 performance highlighted numerous issues and changes to be made, if indeed we wished to remain viable within the evolving challenges of the SPL. The time had demonstrably come for the long foreshadowed transition to ROS2 [2].

We are confident that these lessons will strengthen the team in the long run and lay the groundwork for more competitive performances in the future within the Champion's Cup. Having completed the transition, our focus remains on further refining the vision pipeline and updating the behaviours engine to make better decisions under uncertainty.

5 Impact

5.1 On SPL

The Hengst walk engine [6] was a significant contributing factor to winning 2014 and 2015, and reached the 2016 finals as part of the UT Austin Villa system. It was also integrated into the 2017 B-Human code release.[3]. In 2021 a labeled dataset for field segmentation consisting of 20 videos was published [7].
rUNSWift has in previous years had several members elected to the technical and organising committees, with Claude Sammut serving as past President and continuing as a Trustee.

We also hope that our ROS2 developments will provide a good opportunity for prospective teams to investigate the tools and utilities offered therein. We intend to release the entire architecture behind our modules so that any interested

[3] See Section 8.3: Walking https://b-human.de/downloads/publications/2017/coderelease2017.pdf

parties can 'plug and play' modules as they wish. We hope this helps facilitate easier ROS2 transitions for other teams that wish to leverage its benefits.

5.2 On UNSW & local community

rUNSWift organises demonstrations throughout the year on occasions such as Open Day, including Naos playing on a small field, inspiring prospective students to consider Computer Science/STEM careers, serving also as entertainment for the public and fostering positive associations with the robotics field. We also use these events as a platform for future recruitment, refereeing, training, and to inform students about RoboCup and SPL.

Regarding coursework, students are given the option to work on a project related to RoboCup as a part of a 'Robotics Software Architecture' course offered at UNSW, offering a chance to expand their knowledge within the context of a real project.

rUNSWift often visits schools to inspire students to engage with the future of robotics. We have historically had an association with Kensington Primary School[4] and Holy Cross College, Ryde[5]. By taking Naos on-site to Sydney's regional-level RoboCup junior, we aim to inspire the next generation of RoboCuppers to continue pursuing their interest in robotics.

rUNSWift also took the main stage at Sydney's South by Southwest (SXSW) technology expo, showcasing SPL to a wide audience consisting of enthusiasts, professionals, and researchers. The presentation highlighted rUNSWift's latest developments in motion control, localisation, and team coordination, demonstrating real-time decision-making and adaptive behaviours. By engaging with attendees and discussing the challenges of SPL, we sought to bridge the gap between academic research and public interest in autonomous robotics.

rUNSWift looks forward to any opportunity to showcase and develop both our university's (as well as our broader community's) technical aptitude. With the redevelopments completed on the UNSW Village Green synthetic soccer field (now meeting the FIFA accreditation standard), we hope to do so in concert with the objectives of the RoboCup SPL.[6]

6 Other

Acknowledgements

The 2025 team wish to acknowledge the legacy left by previous rUNSWift teams and deeply thank the School of Computer Science and Engineering, University of New South Wales for their continued administrative, financial and laboratory support to our team. We'd also like to warmly thank additional sponsors and

[4] https://kensington-p.schools.nsw.gov.au/
[5] https://hccryde.syd.catholic.edu.au/
[6] https://www.estate.unsw.edu.au/village-green-redevelopment-0

contributors, including though not limited to ANT61, the Chief Scientist of NSW and CR8. We also wish to pay tribute to all RoboCup teams, in particular RoboCup SPL teams that inspire and drive our innovations in the spirit of friendly competition.

References

1. Asavkin, M., Bell, N., Gopikrishnan, N., Immaraj, S., Lizura, M., Pillay, M., Sammut, C., Schmidt, P.: runswift team report 2024. Tech. rep., The University of New South Wales (2024)
2. Asavkin, M., Gopikrishnan, N., Lizura, M., Sammut, C., Schmidt, P., Vijayan, A.: runswift team report 2023. Tech. rep., The University of New South Wales (2023)
3. Ashar, J., Brameld, K., Jones, E.R., Kaur, T., Li, L., Lu, W., Pagnucco, M., Sammut, C., Sheh, Q., Schmidt, P., Wells, T., Wondo, A., Yang, K.: runswift team report 2019. Tech. rep., The University of New South Wales (2019)
4. Bai, G., Brady, S., Brameld, K., Chamela, A., Collette, J., Collis-Bird, S., Hall, B., Hendriks, K., Hengst, B., Jones, E., Pagnucco, M., Sammut, C., Schmidt, P., Smith, H., Wiley, T., Wondo, A., Wong, V.: runswift 2017 team report and code release. Tech. rep., The University of New South Wales (2017)
5. Brameld, K., Hamersley, F., Jones, E., Kaur, T., Li, L., Lu, W., Pagnucco, M., Sammut, C., Sheh, Q., Schmidt, P., Wiley, T., Wondo, A., Yang, K.: runswift 2018 team report and code release. Tech. rep., The University of New South Wales (2018)
6. Hengst, B.: rUNSWift Walk2014 report. https://github.com/UNSWComputing/rUNSWift-2014-release/blob/master/20140930-Bernhard.Hengst-Walk2014Report.pdf, University of New South Wales (2014)
7. Lu, W.: The rUNSWift SPL Field Segmentation Dataset (08 2021)