UNSW+RMIT RoboCup@Home Team Description Paper

A. Golding, A. Tran, Z. Li, M. Pagnucco, Y. Song, C. Sammut, and T. Wiley*

School of Computer Science and Engineering & Creative Robotics Lab,
The University of New South Wales, Sydney, Australia,
https://robotics.cse.unsw.edu.au/runsweep/

*School of Computing Technologies, RMIT University, Melbourne, Australia

1 Introduction

This team description paper is accompanied by a video that demonstrates some of our capabilities on the Toyota HSR and Fourier GR-1 robots. The videos, taken in our HRI laboratory, shows a "clean the table" demonstration involving planning in our Teleo-reactive language, superquadric pose estimation for grasping, manipulation, navigation, all using a world model. This is followed by a short clip on multimodal reasoning, and preliminary results with the Fourier GR-1 humanoid robot, including SLAM, navigation and reinforcement learning for grasping.

The current work is mainly focused on moving to the Fourier GR-1 humanoid robot. While the locomotion and manipulation are very different from the Toyota HSR, most of our vision pipeline, world model and task planner are platform independent. For this year, we will continue to use the walk provided by Fourier, but have had to implement our own mapping, navigation and manipulation capabilities. The latter includes sim-to-real development of grasping behaviours using Isaac Lab. This will later be extended to create new walks that are better capable of getting around a cluttered home environment.

Continuing work includes further development of the robot's world model to support perspective taking. The world model serves as the central repository of the robot's short-term and long-term memory. Somewhat like SOAR [1] and KnowRob [2], the world model is a symbolic representation that provides knowledge to the planner and the dialogue system. The planner is based on an extended version of Nilsson's Teleo-Reactive planner [3], developed in collaboration with Prof. Keith Clark at Imperial College and Dr. Peter Robinson at the University of Queensland [4]. The world model is also used for "common sense reasoning". For example, if language alone is not sufficient to disambiguate between a reference to an object; proximity, function or recency can be used as reasonable guesses to resolve the reference.

The transition to the GR-1 has not been easy, largely because the system software provided by Fourier is very immature and successive updates have been unstable. However, we now have vision, mapping and navigation and basic manipulation working. Since the the world model, planner and dialogue systems are mostly platform independent, they port relatively easily. In addition, these have been run on a PAL Robotics Ari robot, with little change.

2 System Components

As in previous years, our world model is the central node that stores perceptual and spatial information, maintains a history of interactions, and serves as the belief store for the TR planning system [5]. The remaining components, such as object recognition, are derived from existing open source software libraries. We have had to change some of these with the transition to the GR-1.

2.1 World Model

The world model represents regions, objects, and people and stores them in a deductive database, making it easy to access the locations of objects relative to specific regions and other objects. The object database is automatically filled as the robot autonomously explores its surroundings using a vision pipeline with inputs from YOLO. It also detects people in the environment and captures features that allow re-recognition. Meanwhile, the region database is populated from a topological map program that is able to identify regions in the occupancy grid created by SLAM. The world model also captures interactions with uniquely identified people that is used by the planner.

The world model is central to the robot's intelligent behaviour as it stores the robot's current beliefs about the world. A current area of research is also exploring the maintenance of beliefs of each person in the environment. These provide the context for reasoning when, for example, the dialogue refers to objects or locations that are not directly visible. The world model represents the current state of the world required by the planner. It is implemented as a PostgreSQL database[1]. The tables in the database map to predicates that are easily imported into Prolog, and therefore can be used for deductive reasoning or planning, as described in Section 3.1

2.2 Vision Pipeline

The vision pipeline provides a generalised image processing module. The pipeline is designed to reduce computational overhead by only processing frames that contain objects of interest. The modular plug-and-play approach allows for easy expansion of the core functions enabling research and development of new perception methods. The pipeline consists of object and human pose detection, object annotation including position within the map, size, shape, colour, and person recognition. Specific vision functions such as bag handle detection or empty seat checker can draw from the data generated by the pipeline. Ultralytic's YOLOv11 is used in our vision pipeline to handle object and human pose estimation. The Ultralytics framework provides seamless deployment capability and enable us to swap to newer YOLO models if appropriate.

2.3 Person Re-Identification

Person re-identification is an important part of human-robot interaction. Our person re-identification process *(Figure 1)* extends current state-of-the-art methods of object detection, classification, and tracking. If a person is detected in the scene, the object track is verified and updated, and the viewpoint of the person from the robot is determined using one of 8 defined viewpoints. Features from the face and head are extracted when available and compared to known people, if the person is recognised, the features are updated in the world model, if not, a new person is added to the world model.

2.4 Multimodal Perception

The multimodal perception fuses speech, gesture, and gaze data. Applying visual perspective taking and linguistic grounding integrated with the world model generates grounded references that are used by the planner. The reference resolver can also convert ambiguous references to grounded objects, for example, a person might say "move that over there" while pointing at the object of interest and confirming the target location. Each event is associated with a uniquely identified person which provides a per-agent episodic memory.

[1] https://introlab.github.io/rtabmap/

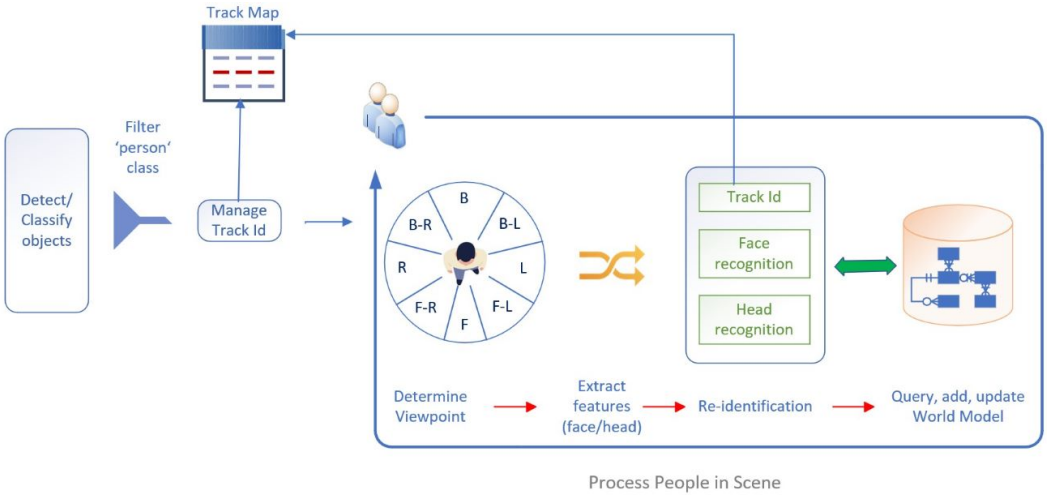


Fig. 1. Person Re-Identification High Level Architecture

2.5 Conversational Agent

The current dialogue system for the robot uses RASA, an open-source language understanding and dialogue management system [6]. RASA Core can be executed on a local host, with a ROS node serving as the interface that connects the RASA model and user input text. The interface executes in a node, constantly passing the input text to the NLP model (converted from speech by VOSK, a speech-to-text system, described in 2.8). The output from the model is converted into goals for the *TeleoR* planner.

The conversational agent is also connected to the robot's path planner to be able to output directions for the "where is" task. The parser is augmented by the world model, which has knowledge of relations such as "contained in", so that the robot can correctly respond to utterances like "I want a bag of chips". The system knows that the chips are in the kitchen cupboard so directions are given to the cupboard. The SLAM map is labelled with the locations of objects and spaces, which correspond to the symbolic spatial relations stored in the world model. This provides the information required to turn the path generated by the movement planner into words that make sense to the human user.

2.6 SLAM and Navigation

On the HSR, we have used standard SLAM libraries such as Cartographer[2] and GMapping[3]. Transitioning to the GR-1, which only has RealSense RGB-D cameras, we first experimented with ORB-SLAM [7], but have now settled on RTABMAP[4], as it works well with a depth camera.

2.7 Object Recognition and Grasping

For object recognition, we use YOLOv11 [8] to detect objects in the scene, placing bounding boxes around them, and then using a point cloud generated from the RGB-D camera to locate the object in space. As described in Section 2.1, once the recognition system has done its job, the object's properties, pose, and location are stored in the world model so that this information is available to the language and planning nodes.

[2] https://github.com/cartographer-project/cartographer$_t oyota_h sr$
[3] https://wiki.ros.org/gmapping
[4] https://introlab.github.io/rtabmap/

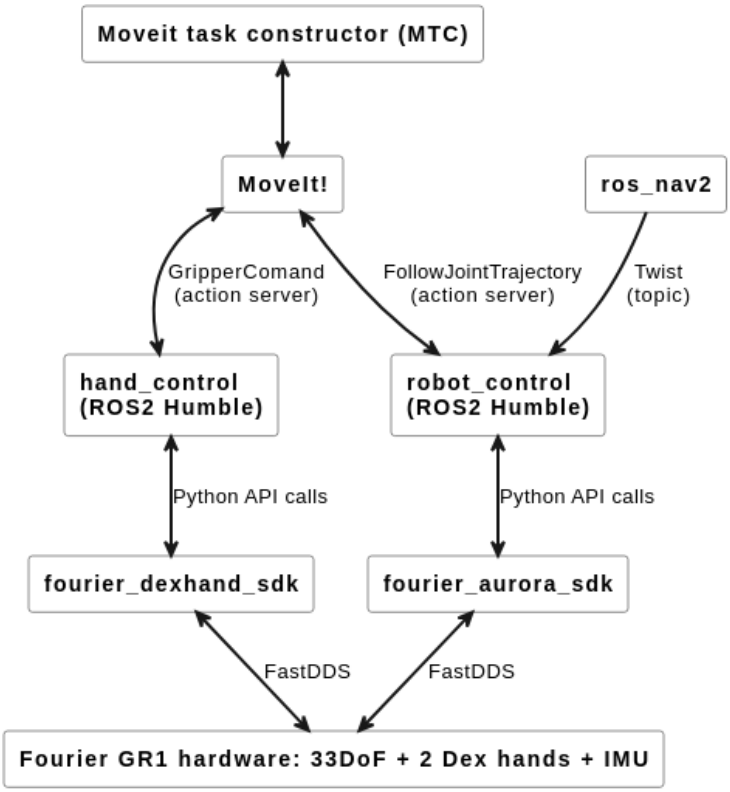


Fig. 2. Control flow for the Fourier GR-1

We have developed a model-based approach to 3D object recognition for grasping, using RGB-D cameras [9]. The vision system extracts shape primitives, represented as superquadrics [10], from the point cloud. Given the superquadrics, we estimate the pose of the object and determine appropriate grasping points. This method is fast, with low memory usage. It is demonstrated in the accompanying video on the Toyota HSR robot and is in the process of being ported to the Fourier GR-1 humanoid robot.

In the videos, motion planning for both the HSR and GR-1 is done using the ROS MoveIt package. In addition, we are training the GR-1 to grasp objects in Isaac Lab, to then transfer the learned policy to the real robot. The control flow for motion planning on the GR-1, including grasping, is shown in Figure 2. Fourier provides two Python SDKs, Dexhand and Aurora. Dexhand controls both 6DoF hands independently of the rest of the body. Aurora provides basic functionality for the IMU, standing, walking, arms joint control, and full body joint control (33DoF). We built two wrappers using ROS2 Humble (hand_control and robot_control) that expose topics and action servers to the rest of the ROS ecosystem. We configured the necessary services to use MoveIt! and ros_nav2 to control the robot. The accompanying video shows a pick object example using the Moveit Task Constructor (MTC) to demonstrate full connectivity between ROS and the low level hardware control.

2.8 Externally Available Components

As above, some components are derived from existing open source software, especially ROS packages. We use the MoveIt or Agile Grasp ROS packages for calculating inverse kinematics and performing manipulation tasks. For face recognition and person tracking we use tools in OpenCV 3.0, and the OpenNI/NiTE skeleton tracking library. The speech-to-text voice recognition software currently being used is Vosk. Additionally, we use some off the shelf

cloud APIs such as Micorosoft Azure's Face API for gender detection. The planning rules are written in *TeleoR*, which is a built in QU-Prolog[5].

3 Knowledge Representation, Planning and Learning

The central theme of our research is combining high-level reasoning with real-time, low-level sensing and control to improve the capabilities of autonomous robots.

As previously mentioned, the heart of the system is the world model, implemented using the PostgreSQL database system. Dialogue scripts can access and update the database with knowledge of people, places, objects and actions. This database implements the system's working memory and it's long term memory. Scripts can incorporate input and output in different modalities, e.g. gestures as well as spoken dialogue. Responses are formulated as goals and passed to the TeleoR planner.

Tables in PostgreSQL store the current state of the robot, the types and locations of objects that have been observed, information about other agents (mainly people in the environment), past actions and interactions. From this, we can extract objects that are currently in view and remember objects out of view. All database items are timestamped so that past events can be recalled and so that old information can be forgotten. Object information is used to re-identify people and objects if they pass out of view and then re-enter the scene. A collection of pre-built SQL queries allows us to write Prolog programs that can interact with the database in a way that is easy to map database items to Prolog facts. This is especially useful for the task planner, which is implemented on top of Prolog.

3.1 Teleo-Reactive Planner

In collaboration with Clark and Robinson, we have implemented a planner that makes use of the Teleo-Reactive (TR) programming paradigm. The main difference between a TR planner and a more traditional PDDL planner is that the TR system attempts to implement a universal plan that is reactive rather than predictive. A TR program consists of a collection of procedures, each of which contains a set of rules that implement what Nilsson called "circuit semantics". That is, the system behaves as if all the conditions in all the rules are continuously being evaluated just as in an electronic circuit. A TR procedure has the general form:

```
proc(<arguments>)
    <goal> -> <do nothing>
    <conditionN> -> <actionN>
    ...
    <condition1> -> <action1>
    true -> <default action>
```

where the actions may be calls to other TR procedures. The condition-action rules are arranged such that the action of a rule lower in the procedure should result in the conditions of a higher rule being satisfied. The lowest level rule is the default rule when no other condition is satisfied and it invokes a default action. The top rule's condition is satisfied when the goal for that procedure has been satisfied. Because the conditions in all rules are, conceptually, being re-evaluated continuously, the planner is very reactive, and thus, capable of responding quickly to changes in the environment.

As an example, a simplified set of TR rules for cleaning the kitchen table is:

```
clean_the_table(){
    holding(ObjName) & Dest = goes_to(ObjName) & near_static(Dest)
```

[5] https://staff.itee.uq.edu.au/pjr/HomePages/QuPrologHome.html

```
        -> place(ObjName, Dest)
    holding(ObjName) & Dest = goes_to(ObjName)
        -> goto_object(Dest)
    near_static(table) & can_see_object(ObjName, ObjId) & is_ooi(ObjName)
        -> pickup(ObjName, ObjId)
    true
        -> goto_region(dining_area)
}
```

Percepts, such as *holding*, are implemented as queries to the world model which includes the current robot state and are also implemented as QuProlog relations and functions. Actions, such as *goto* are implemented as messages sent to a ROS action server that communicates to the appropriate nodes, like navigation.

At present, the TR rules must be hand-coded, but a subject of research is how TR can be learned, either from demonstration or by experimentation. The aim is to learn from far fewer examples than required by reinforcement learning.

3.2 Robot Learning

We continue to develop an episodic memory system that enables the robot to recall past events that may be relevant to the interaction or to solving a present problem. This was the subject of a PhD thesis [11] and is incorporated into the current world model. Event frames are stored in memory with the two primary problems being how to know what events should be remembered or forgotten and what is an appropriate metric to use to determine what a relevant memory is. We have used a novel knowledge acquisition system called "Ripple-Down Rules" to interactively acquire rules for identifying similar events and for determining which past events, stored in the episodic memory, are most relevant to the current situation.

Although we use public machine learning environments, such as Isaac Lab, for reinforcement learning, we also have our own learning systems that are aimed at data efficiency. A major drawback of reinforcement learning systems is that they generally require a very large number of training examples, hence the popularity of simulators. However, if a good simulation mode is not available or if there is a need for continuous, interactive learning, more data efficient methods are needed. We combine learning abstract qualitative models with reinforcement learning, where the abstract layer constrains search in the lower-control layer to greatly reduce the number of trials required [12]. Learning occurs in two phases. In the first, the robot experiments with its available actions to learn qualitative action models. That is, it learns approximately, the effects on its state and the state of the world when it performs some action. This is sufficient to create a high level plan to achieve a goal, but the plan may not have the exact parameters (e.g. joint angle or motor speed) required to actually perform the task. However, the qualitative plan can constrain the search space of a reinforcement learning algorithm so that it is consistent with the qualitative actions of the plan. A current PhD project is aimed at refining this method so that it is applicable to motion planning for our humanoid robot.

4 Experiments and Results

The accompanying videos demonstrate results obtained using the Toyota HSR and the Fourier GR-1. These include:

- Natural speech interaction using a dialogue manager that understands the context of a conversation and uses the context to disambiguate utterances.
- The robot's sensors give it an awareness of its surroundings and, coupled with mapping and knowledge contained in the world model, an awareness of space and objects in the environment.

- The vision system is capable of object recognition and tracking, human pose estimation, and ray tracing of arms to provide real-world context to dialogue. The robot can differentiate between human users through facial recognition and texture matching.
- A teleo-reactive planner takes input from the world model to generate robot behaviours. Planning actions combine vision, knowledge from the world model, person recognition, natural language reasoning, path planning, and manipulation.
- Manipulation makes use of superquadrics to represent objects and pose estimation. This has been demonstrated on the HSR and is being ported to the GR-1.
- Mapping and navigation has been demonstrated on the GR-1, currently using the Fourier walk engine.

5 Conclusion

The major focus of our research in @Home has been:

- transitioning to a humanoid robot platform;
- developing the world model with its connection to the robot's perceptual system and the episodic memory to enhance the robot's understanding of time and space;
- incorporation planning through the TR system;
- applying existing and newly developed machine learning algorithms to enhance the robot's capabilities.

.

We will be one of the first teams to attempt the @Home competition with a humanoid robot. We hope that our experience with the new platform will inform other teams that may be planning to move to a humanoid robot.

References

1. John E Laird, Keegan R Kinkade, Shiwali Mohan, and Joseph Z Xu. Cognitive Robotics Using the Soar Cognitive Architecture. In *8th International Conference on Cognitive Robotics, (Cognitive Robotics Workshop, Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 46–54, Toronto, 2012. ACM Press.
2. Moritz Tenorth and Michael Beetz. KnowRob – knowledge processing for autonomous personal robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266. IEEE, 2009.
3. N. J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158, 1994.
4. Keith L. Clark and Peter J. Robinson. Robotic agent programming in teleor. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 5040–5047, 2015.
5. Keith L. Clark and Peter J. Robinson. *TeleoR User Guide*, 2023.
6. Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management, 2017.
7. Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, October 2015.
8. Mohammad Javad Shafiee, Brendan Chywl, Francis Li, and Alexander Wong. Fast YOLO: A fast you only look once system for real-time embedded object detection in video. *CoRR*, abs/1709.05943, 2017.
9. Jayden Kanbour. Efficient grasp using superquadrics for human service robots. Final year thesis, School of Computer Science and Engineering, The University of New South Wales, 2025.
10. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

11. Colm Flanagan. *Episodic Memory for Cognitive Robots in Dynamic, Unstructured Environments*. Phd thesis, School of Computer Science and Engineering, The Uniersity of New South Wales, 2021.
12. Timothy Wiley and Claude Sammut. Data efficient online learning of robot behaviours via qualitative planning and reinforcement learning. *Robotics and Autonomous Systems*, 194:105122, 2025.

Annex

The foreground software used in 2025 has been made available at `https://robotics.cse.unsw.edu.au/gitlab/toyota-hsr/runsweep2025`. The software is described in the README.md file of the git repository. An excerpt of the readme is provided below.

The ROS packages of the foreground software are described below.

- **manipulation**
  - *hsrb-interfaces-unsw*: A fork of the toyota HSRB interface package
  - *object-grasper*: Package contains services for manipulation actions
- **msgs**
  - *unsw_action_msg*: Custom msgs for action dispatch
  - *unsw_vision_msgs*: Custom msgs for unsw vision pipeline
- **navigation**
  - *gmapping*: Fork of hsrb gmapping package used for mapping
  - *nav_utils*: Contains navigation scripts such as person-following. Intention to refactor to action_servers
  - *rosnav*: Edited fork of HSRB rosnav repository
- **planning**
  - *action_clients*: State machines to dispatch planner outputted actions to action action_servers
  - *action_servers*: Servers for all base actions. Containing manipulation actions at the moment
  - *state_machine*: Package used as proof of concept for continuous feedback using smach library
  - *TR*: Teleo-reactive planner
- **speech**
  - *sound_stream*: Package to publish microphone stream to ROS topic
  - *speech_to_text*: Vosk based model for speech to speech_to_text
  - NOT INCLUDED:
    - *nlp-rasa*: Package containing scripts for nlp using rasa model
- **tasks**
  - *carry_my_luggage*: State machine/behaviour tree based control for carry my luggage task using services
  - *clean_the_table*: State machine based control for clean the table task, using action_clients and action_servers
  - *inspection_task*: State machine based control for inspection_task
  - *pick_up_apple*: State machine based control for simple integrated vision and manipulation task using services
  - *serve_breakfast*: State machine based control for serve_breakfast written at RoboCup 2024, unchanged and untested since competition. Will be refactored
  - *store_groceries*: State machine based control for store_groceries task written at RoboCup 2024, unchanged untested since competition. Will be refactored
- **tools**
  - *common_utils*: Scripts that don't really belong anywhere such as "door checker" to check if a door has opened

- **vision**
  - *person-recognition*: Person recognition and reidentification inserted in vision pipeline
  - *utils*: Specific vision nodes used for small tasks such as identifying a bag handle or a person pointing
  - *yolov11*: Wrapper of yolov11 for ROS
- **world model**
  - *world-model*: World model and scripts used to maintain belief store

We are using an external device for additional processing, compliant with the rules of the Domestic Standard Platform League: Dell Mobile Precision Workstation (Intel Core Ultra 9, NVIDIA GeForce 4090, 64Gb RAM) connected via Ethernet to the HSR, mounted on the standard backpack mount.

UNSW uses the following third party software and libraries for the competition:

- Vision Processing: YOLO, Face Recognition
- Grasping: MoveIt, Grasp Pose Detection (GPD)
- SLAM and Navigation: GMapping, ROS navigation stack
- Speech: Vosk
- Database: Postgresql