UTS Unleashed! RoboCup@Home SSPL Champions 2019

Sammy Pfeiffer, Daniel Ebrahimian, Sarita Herse, Tran Nhut Le, Suwen Leong, Bethany Lu, Katie Powell, Syed Ali Raza, Tian Sang, Ishan Sawant, Meg Tonkin, Christine Vinaviles, The Duc Vu, Qijun Yang, Richard Billingsley, Jesse Clark, Benjamin Johnston, Srinivas Madhisetty, Neil McLaren, Jonathan Vitale, Mary-Anne Williams (Team Leader) *

University of Technology Sydney, 15 Broadway, Ultimo NSW 2007, Australia
{Sammy.Pfeiffer, Sarita.Herse, TranNhut.M.Le, Su.W.Leong, Bethany.Lu, Katie.A.Binns, Tian.Sang-1, Ishan.Sawant, Margaret.Tonkin, Christine.Vinaviles, TheDuc.Vu, Qijun.Yang}@student.uts.edu.au
{Richard.Billingsley, Jesse.Clark, Daniel.Ebrahimian, Benjamin.Johnston, Srinivas.Madhisetty, Neil.McLaren, Syed.Raza, Jonathan.Vitale, Mary-Anne.Williams}@uts.edu.au

Abstract. This paper summarizes the approaches employed by Team *UTS Unleashed!* to take First Place in the 2019 RoboCup@Home Social Standard Platform League. First, our system architecture is introduced. Next, our approach to basic skills needed for a strong performance in the competition. We describe several implementations for tests participation. Finally our software development methodology is discussed.

Keywords: RoboCup@Home · RoboCup · Social Standard Platform League · Social Robotics · UTS Unleashed!

1 Introduction

UTS Unleashed! is the only Australian team to participate in 2017, 2018 and 2019 RoboCup@Home Social Standard Platform League. It was awarded the Human-Robot Interface Award in 2017, and able to achieve second place in the competition in both 2017 and 2018. In 2019 the team demonstrated significant problem solving and software development capability for developing sophisticated behaviours for Softbank Pepper robots to win the competition. To prepare for the 2019 RoboCup competition the team focused on system robustness by running all its code locally on the robot, while making fault-tolerant use of external computing resources when available and necessary.

2 System Architecture

Our system architecture was designed to tackle the specific challenges of the Pepper [1] robotic platform, the type of goals to be achieved in the different

* Authors are listed in alphabetical order after the student development lead, Sammy Pfeiffer: students are followed by researchers.

RoboCup@Home tests and the available network infrastructure constraints and performance at the RoboCup competition. It also takes into consideration the profile of our team composition which includes a mix of multi-disciplinary skills (Human-Robot Interaction, User Experience, Psychology, Machine Learning, Software Engineering, Design, Business and Law). Our diverse team members ranged from Undergraduate through Masters and PhD students and possessed a wide range of technical skills.

A diagram of our system architecture is illustrated in Fig. 1. All software modules were implemented with on-board deployment in mind and can be run externally as well. To run directly on the robot itself, dependencies and software needed to be compiled and configured for that purpose. This was accomplished by the base OS, `magiclab_pepper_os`. Services that benefit from running on an external computer or having a cabled internet connection are also run in the external computer and used automatically when the network is available using our sophisticated `magic_ros` framework.

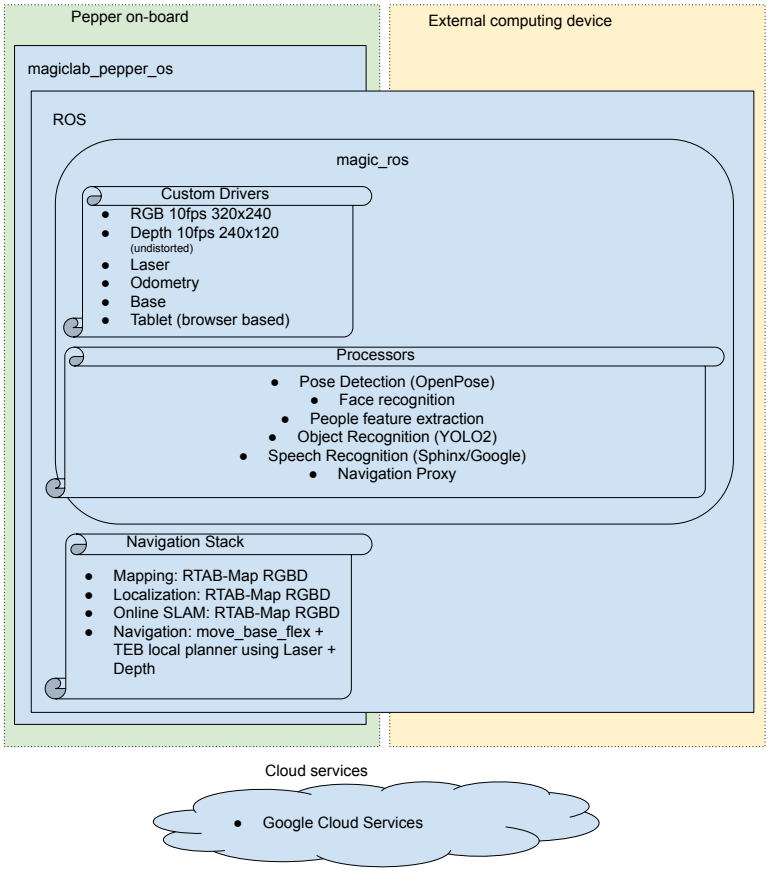

Fig. 1. UTS Unleashed! Software Architecture.

2.1 Base Operating System

A common challenge faced by all teams in the SSPL (Social Standard Platform League) is the need to build a large number of updated and new dependencies in order to enable usage of state-of-the-art algorithms and tools as found in the ROS middleware [2]. The challenge comes from the Pepper platform using a 32 bits Gentoo Operating System (OS)[1] with no root access, making developers unable to use standard software deployment strategies (e.g. `sudo apt-get install` as in a typical Ubuntu distribution). Building every package by hand was found to be extremely time consuming so a better solution was developed.

`pepper_os` [3] was designed and implemented, a 32bits Docker image based on a dump of the disk image of the Pepper robot[2] with Gentoo Prefix [4] and ROS_overlay [5] already built with a big set of commonly used ROS packages (e.g. `navigation`, `perception`, `rosbridge`, `naoqi_driver`...). Additionally, some extra useful (and hard to compile on a 32 bits OS) libraries like Tensorflow [7] can be found. A user must only extract the latest release onto the home folder of their robot and the robot will boot with a `roscore` ready to be used.

Furthermore `magiclab_pepper_os` [6] contains the full image used in our robots for the competition, including other hard to build libraries like PyTorch [8], RTAB-Map [9], OpenNI2 [10], dlib [11] and spatio_temporal_voxel_layer [12].

2.2 The `magic_ros` framework

During the previous years of the competition, the team identified difficulties for some team members to learn and use ROS quickly and effectively. Also, similarly to other robotics competitions, teams have to optimize to make the best use of wireless networking during the competition but not completely depend on it. Hence, the library `magic_ros` [13] was born, implementing the following features:

- ROS complexity is hidden: Auto initialization of uninitialized nodes; no need to use ROS message format as you can pass native Python datatypes around; autoconversion of images to cv2/numpy format.
- Helper informative messages, setting parameters such as `ROS_MASTER_URI`, `ROS_IP`, `init_node`,....
- Message Providers with subscription, query-able acquisition models and a buffer. This also enables easy logging and recording of data. They are network efficient, as `magic_ros` automatically detects if messages are to be delivered on-board or through the network, using information compression when necessary.
- Processors are nodes that enable remote calls, with the ability to have multiple nodes in the network doing the same task, different selection strategies are used to choose which processor to run first. Calls are guaranteed to go

[1] The OS image is dated as built in November 2016 with binaries and shared libraries dated from 2014.

[2] To be able to build in exactly the same conditions than the real robot, as the provided Virtual Machine by the manufacturer does not match the real system precisely.

through the network if it is possible. As an example a full OpenPose [14] call (containing RGB + Depth images and returning a dictionary with all the available 3D skeleton data and Regions Of Interest (ROI) of faces found and additional data) takes just 0.6-0.8s (±0.1s standard deviation) added to the processing time. Messages are compressed as much as possible usually fitting into a single TCP packet. All clients do not need to do any work to take advantage of nodes running in more powerful external computers when they are available, this is transparent to the user.
– Automatic diagnostics publication, making it easy to monitor and debug.

Custom Drivers Some drivers were re-written using `magic_ros` to have more control of the on-board processing pipeline. It was also an opportunity to double check and confirm if the drivers were correct. It is worth mentioning we found that the default laser driver provided by `naoqi_driver` provides inaccurate data. We implemented a better approximation on a virtual single laser which improved our navigation stack. A query-based interface was created for the camera images, which ran at 10Hz at low resolutions (320 × 240px RGB, 240 × 120px Depth) to minimize computing costs while still providing enough useful data. We also benchmarked the robot's odometry and discovered it was surprisingly good.

3 Skills

Some robot skills are of utmost importance to ensure a high performance in RoboCup@Home SSPL. The approach UTS Unleashed! took to prioritize foundational robot skills is outlined in this section.

3.1 Navigation

Autonomous navigation is necessary in all RoboCup tests, therefore, considerable effort was devoted to this topic.

Mapping RTAB-Map (Real-Time Appearance-Based Mapping) [9] is used for mapping with RGB and Depth images. The robot odometry is consumed at a rate of 10 Hz. RTAB-Map is able to provide 3D localisation but to minimise computational cost is constrained to 2D. A set of rosbags were recorded on the robot following trajectories that the robot is expected to pursue during the different tests while taking care to face the robot towards fixed landmarks. Examples of landmarks are walls, doors and other fixed furniture like kitchen sinks. We also positioned the head of the robot to look at a 45 degrees angle towards the floor to maximize the amount of close-range, visible features that can be recognized for localisation later. Some rosbags were used for the mapping process, while others were used for map verification.

Once the rosbag-recording stage is completed, we run the mapping process on a powerful external computer. A 2D projection of the map is created with the RTAB-Map library, then manually cleaned and obstacles are inflated for use in global path planning and visualisation purposes as shown in Fig. 2.

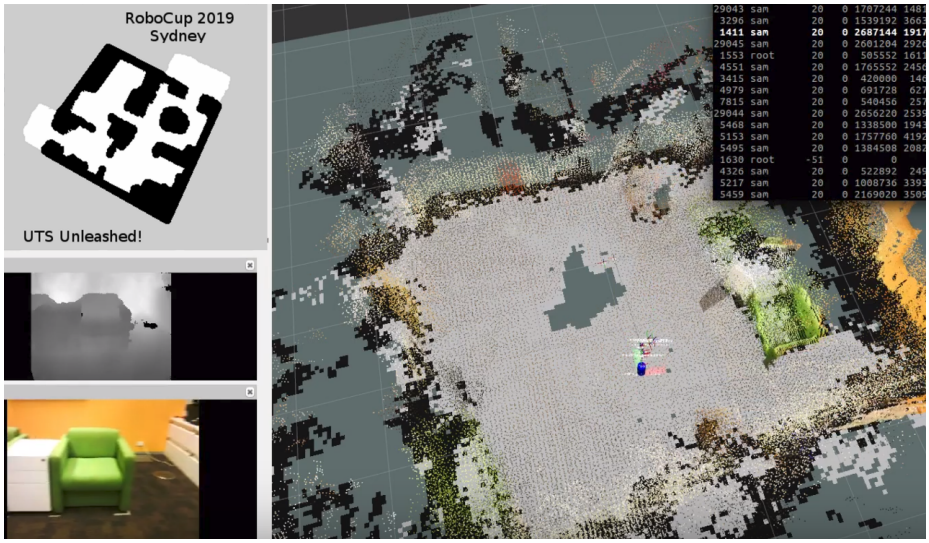

Fig. 2. RoboCup@Home SSPL arena gridmap used for global path planning and visualization on the top left corner. The rest of the image is an example of the 3D pointcloud and default gridmap provided by RTAB-Map with some real-time RGB+Depth data shown in the lab.

Localisation Localisation was also achieved using RTAB-Map, but running on-board at 10fps. We discovered that relying on laser data alone proved to be insufficient.

Navigation stack It is based on `move_base_flex` [18], hence generating a navigation state machine.

On receiving a navigation goal, a global path is queried to the standard ROS navigation stack, *global_planner/GlobalPlanner*. When a valid plan is found, it is executed by the local planner. We use Time-Elastic Bands (TEB) local planner [19] running on-board. To do real-time obstacle avoidance we use Pepper's laser (6Hz publishing frequency) and its depth camera (at 10Hz) as sensing sources. To be able to compute this information on-board we make use of spatio_temporal_voxel_layer [12] to feed pointcloud data to the plugin *costmap_converter::CostmapToPolygonsDBSMCCH* which converts this information into something processable in real-time.

We stop the navigation stack software while the robot is not moving to save CPU cycles by adapting the Linux approach of sending a signal of *SIGSTOP* to the RTAB-Map localisation process and the navigation state machine and sending *SIGCONT* when a new navigation goal is received until it is finished.

The specific configuration of our navigation stack and the TEB local planner will be available at our code release.

3.2 Speech Recognition

Speech recognition was performed by running multiple speech recognition engines in a concurrent fashion. On-board the robot, CMU PocketSphinx [15] provided a fallback recognition strategy when external cloud-based recognition was not available. When network conditions allowed cloud-based speech recognition, Google Cloud Speech to Text [16] was used. To improve performance over unreliable WiFi connections, audio data is streamed over the network using the compressed Opus [17] audio codec and decompressed on an external computing device before being forwarded to the cloud-based recognition service. As many cloud-based speech recognition services do not support controlled grammars, the recognized text is post-processed using a grammar. A dictionary is used to convert English transcriptions into phonemes, and then an approximate match is performed to find a high-probability parse.

3.3 Perception

Object Recognition Our object recognition module is built on YOLO [20] version 3. Named `dark_magic` because it is built on `darknet_ros` [21], a subset of YOLO for ROS, is able to produce rapid object detection with extrapolated 3D locations. On a machine with a GPU this is close to real-time; on Pepper's CPU, it detects objects within approximately 3.3 seconds.

Altogether, approximately 200-250 images are needed for each object to be trained. The number of images per object in the training data should be kept approximately equal, in order to prevent detection bias affecting the results. With the data sets prepared, we separated them into training and testing batches.

From an RGB image, the module will output the name of the detected object, the detection confidence, and a bounding box. With depth image added, 3D location in robot and world space of each object can also be extracted.

We improved the efficiency of the network by fine-tuning certain configurations such as mini-batch and network resolution. At times, this meant finding a compromise where the resolution of the images were small enough for fast detection, but still provided enough information for accuracy, due to computational limitation on CPU. A higher resolution was kept in the full (GPU) model.

People Perception We made extensive use of OpenPose [14] to detect people in an unconstrained environment like RoboCup@Home. As with other libraries, we created a `magic_ros` server and a client called `magic_pose`. The server automatically detects if the machine on which it is running has GPU access (external machine) or not (Pepper) and initializes the appropriate model (either GPU or CPU based). The client is the same for both, the GPU and CPU servers.

We also used OpenPose to estimate people's poses, one of which was if the person was sitting. This was achieved by measuring the height of the person and setting a threshold below which the person was estimated to be sitting.

Finally, we also used OpenPose to crop faces and further extract facial features, especially for non-frontal and far away faces which are harder to detect with our face detector.

When it was possible to make simpler assumptions on the location of the person (e.g. expecting a person within 1.5 meters and in front of the robot) we could rely on a different library to detect faces. We used the Python module `face_recognition` [24] that makes use of dlib [11]. As we did for OpenPose, we wrapped these APIs in client/server processors of our architecture. This library was also used to extract face encodings for face identity recognition tasks.

3.4 Human Robot Interaction

We developed our own web-based tablet interface with UTS-branded aesthetics including simple animations to express what the robot is doing. It became a useful debugging tool for programmers in addition to aiding the robot users in understanding what is happening and how to interact. We provided a simple API to increase adoption. A separate paper about this is to be published.

4 Test implementation highlights

We consider designing and implementing in a well-thought manner. Every test is important and here we show what we consider as key elements of our results in this edition of RoboCup@Home. We chose these tests by common decision in our team by taking into account our strengths, weaknesses, and opportunity for success. Most tests were implemented as State Machines. This section details some of these tests, while the full test descriptions can be found in the official RoboCup@Home rules website.

4.1 Stage I

Clean Up We used the `dark_magic` module for object recognition. Once the model arrived at an acceptable accuracy, we focused on HRI elements, particularly expressing the locations of the found objects to the operator in a clear and effective way, via both speech and tablet display and then ensuring the operator places the object on it's predefined location by detecting the object again.

Find My Mates We detected all the people in the room from a fixed set of safe viewpoints, which were selected to prevent blind spots. For each person detected we estimated an accurate position in the map, and achieved higher accuracy by correcting distortions of the depth camera images. People estimated within 20cm from a previously estimated person in the map were joined together as a single person. Face recognition was used to further reinforce this strategy. People detected outside the considered room were disregarded (e.g. audience).

After collecting the positions of all the people in the room, we estimated their pose (either standing or sitting), which was affirmed by checking if the person

was located near a seating landmark. We also cropped the face of the person and a portion of the torso.

To predict facial features we trained several SVM (Support Vector Machines) models to classify facial features by using the Large-scale CelebFaces Attributes (CelebA) dataset [22]. We first cropped the faces of the dataset and then extracted face encodings using the Python module `face_recognition` and its function `face_encodings`. We thought that the face encodings generated by the face_recognition module would most likely include information about features characterizing the physical appearance of the person's face to distinguish them. Our approach achieved 95% prediction accuracy when testing our models for facial features: facial hair, dark colored hair, gray hair, eyeglasses and gender.

To predict the color of the t-shirt we collect only the pixels in the torso with higher depth, to avoid occlusions. We then increased the luminance and saturation of the collected pixels and used a median filter with window size 3 to reduce the noise and unnecessary fine details. After this pre-processing we transformed the image patch from RGB space to CIELAB space. In the CIELAB space the distances reflect those in a perceptual space more closely, which is ideal when the task is to report colors to a human. The pixels of the patch were clustered together down to 3 colors through a k-means algorithm. Finally we employed a color dictionary with a limited set of colors we used to estimate distances of each pixel to a color in the dictionary. Each color belonged to a color group (e.g. blue, red, etc.). To compute perceptual distances between colors in our color dictionary we used the CIEDE2000 color difference distance [23]. This returned better results than employing a simple Euclidean distance or colors represented in HSV space. The likelihood of each detected color was calculated by the number of pixels bucketed in a specific color group over the number of total pixels of the patch. Finally, we estimated the color of the t-shirt using thresholds for the estimated likelihoods: a single color if a color reached 70% likelihood, multiple colors with over 50% likelihood, if any, or no color was returned otherwise.

In addition, we used Google Cloud Vision to detect text from the torso. If we detected any text from the torso we reported the collected characters back.

Our strategy revealed success in reporting the correct location and a unique description for one of the mates the robot was asked to find.

Receptionist Finding an empty seat was the hardest task in this test and the most crucial to score points. Our approach to finding an empty (sofa) seat was based on a simple heuristic that any surface with height within a range (say 40cm to 60cm) from the ground would be a seat. To differentiate seats from tables we also detect a seat back, by looking for a vertical surface just behind the horizontal surface. To return a location of an empty seat in the sitting area, the robot turned its head from left to right and calculated area for each empty seat found. Finally, the location was returned with the largest seat area as an empty seat.

Take Out The Garbage Given the placement of the trash cans is known beforehand the main challenge for this test was carrying rubbish bags over a few minutes. The bags may come loose and fall or the robot's arms may overheat. We asked an operator using speech and a descriptive tablet interface to have a bag placed between the hip of the robot and under the arm, then we continuously send commands to keep the arm in position. We repeat this approach with the second bag and finally navigate to the drop-off point.

4.2 Stage II

Find My Disk The robot provided specific instructions to the blind person about how to position the disk in front of its camera, and then compared text visible on the disk with the operator's desired disk. While disk positions were tracked using the robot's depth sensors, a naive strategy for disk tracking was not appropriate because of the depth sensor's limitations. In particular, the depth sensor is unable to measure the distance of objects that are held too close to the camera or highly glossy objects at angles that reflect the sensor's infrared laser light directly back into the sensor. In such cases, the result may contain depth ranges with no valid value and it is non-trivial to distinguish between objects that are either too close or objects that are at an appropriate distance but are simply glossy. Close-proximity objects can be handled by making the robot provide instructions to step backwards but glossy objects are a challenge to the sensor itself. Using in-lab experimentation, a set of simple heuristics was developed based on a binned histogram of depth ranges. These heuristics were simple if-then rules that mapped depth data into spoken commands to the operator.

The operator's desired disk was recognized verbally using Google Cloud Speech to Text. The currently held disk was recognized by using depth camera data to determine an appropriate crop of RGB camera data. The cropped RGB image was then sent to Google Vision for optical character recognition (OCR). The two transcriptions (speech and OCR) were then compared using a Levenshtein distance, The OCR transcription is modified by deletions at its start or end to make it zero cost.

Hand Me That The main problem was detecting pointing. We tried several methods in order to assess the direction that the person was pointing in. In the end we used data directly from our `magic_pose` module providing 3D joint poses. To see most of the human body in the frame, we used raw data from the deep learning module to assess the likelihood of pixel clusters representing elbows and wrists, which ended up improving accuracy. In the test, while detecting the person, our robot would get closer and look only at their hip area to find the arm joints.

Restaurant We scanned for waving people while turning around via `magic_poses`. The robot waves with both arms as an example of how we would like clients to wave. Naive users tend to wave with both arms imitating the robot increasing

the chances of a wave detection. If we find a client we navigate towards it by using our navigation stack.

If we don't find a waving person after a full turn of the robot looking for them, we repeat the strategy but taking HD images and scanning sections of them. This approach, while much slower, enabled us to find people waving up to 9 meters away (but without depth information given the range and noise of the depth camera). If someone waving far away is found, we navigate closer in the detection direction, and detect again.

The map location of the robot is saved for delivering the order as the customer is within reach, as it's known the robot can safely navigate to this point.

Due to the limited manipulation ability of Pepper's arms, all of the manipulation aspects of the test were managed by using HRI. The robot holds its arms outstretched in front, and the operator is instructed to place an off-the-shelf tray on the robot's arms, with the ordered items following.

Between orders, the robot interacts with the barman to remove the tray from its arms to allow the robot to return its arms to a neutral position. This step is required to manage the heat produced by the arm motors which tend to overheat very quickly especially when carrying weight. On top of this, while navigating with the tray the depth camera loses a major section of its field of view, therefore making navigation less safe. Speed limits are thus lowered.

Where is This? For this challenge, we developed a social navigation overlay on top of the global and local planner otherwise used for robot navigation. A dense map of socially meaningful way-points, socially meaningful connections between way-points and objects of interest was assigned to the arena. A human expert then assigned socially meaningful descriptions to the way-points and connections. When asked the location of an object or place, the robot first finds a shortest path through the social navigation overlay. A plain-English explanation is generated by concatenating the expert descriptions. The robot then uses the global and local planner to physically navigate through the arena along a path that approximates the social navigation overlay.

5 Software Development Methodology

We believe our success this year was aided thanks to our approach to the software development process. Some key elements of our approach are shown here.

5.1 Standard Software Development Practices

We made use of the available standard software development tools and practices such as Git, GitLab, automated testing, continuous integration and continuous deployment, coding standards and code reviews.

During different moments of the project the strictness to adhere to these practices needed to be re-evaluated in order to allow for smooth development across all team members.

We chose Python as our main programming language for its ease of learning and usage, and also the availability of useful libraries. We chose the ROS middleware for similar reasons.

5.2 Operational Readiness Tests

Three months out from the competition we conducted Operational Readiness Tests (ORT) every two weeks. Closer to the date of RoboCup we moved to weekly ORTs. Other teams have advised that they ran similar events weekly all year long. At RoboCup 2019 we simulated the conditions of the RoboCup@Home SSPL competition as closely as possible in our ORTs. For this exercise we used a house-like scenario inside of our laboratory similar to a real RoboCup@Home arena. This testing space was reconfigured from time to time both to practice for the competition setup days and to find new, unexpected challenges.

We believe this exercise helped in advancing the development of necessary robot skills and tests. It also allowed the team to brainstorm effectively and generally strengthen the team spirit.

5.3 Mentoring & Pair Programming

Initially each person was allocated a Stage 1 task to work on and at least one partner to work with. This was so that each member would be supported by each other, and to allow more discussions and creative solutions.

When the pairs were finished with their task, they could either choose to work with another pair who did not have as much progress, or begin working on a stage 2 task either in the same pair or with other people. This gave team members the chance to develop new skills, and to collaborate with people who they may not have interacted with much previously.

6 Summary and Outlook

In this paper we described the approach used to win the first place in RoboCup@Home SSPL 2019. The key elements and motivations of our architecture was discussed with special attention to some key skills that were implemented in our tests. Additionally, our software project implementation model was detailed, which includes regular readiness check, mentoring and pair programming, which we found extremely beneficial to ensure steady progress during development.

Our code release at https://utsunleashed.webnode.com/software contains our stack of software used for the competition.

6.1 Acknowledgements

We want to thank Cecilio Angulo and Bence Magyar for their help on polishing this paper and the Australian Research Council, WiseTech Global, NSW Chief Scientist and Engineer, Commonwealth Bank of Australia and University of Technology Sydney for the support and crucial funding for the team to compete.

References

1. Pandey AK, Gelin R. A mass-produced sociable humanoid robot: pepper: the first machine of its kind. IEEE Robotics & Automation Magazine. 2018 Jul 11;25(3):40-8.
2. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY. ROS: an open-source Robot Operating System. InICRA workshop on open source software 2009 May 12 (Vol. 3, No. 3.2, p. 5).
3. https://github.com/awesomebytes/pepper_os
4. https://wiki.gentoo.org/wiki/Project:Prefix
5. https://github.com/ros/ros-overlay
6. https://gitlab.com/uts-unleashed/magiclab_pepper_os
7. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M. Tensorflow: A system for large-scale machine learning. In12th USENIX Symposium on Operating Systems Design and Implementation ({OSDI} 16) 2016 (pp. 265-283).
8. Paszke A, Gross S, Chintala S, Chanan G. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration. 2017 May;6.
9. Labb M, Michaud F. RTABMap as an opensource lidar and visual simultaneous localization and mapping library for largescale and longterm online operation. Journal of Field Robotics. 2019 Mar;36(2):416-46.
10. https://github.com/OpenNI/OpenNI2
11. King DE. Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research. 2009;10(Jul):1755-8.
12. https://github.com/SteveMacenski/spatio_temporal_voxel_layer
13. magic_ros library https://gitlab.com/uts-unleashed/magic_ros
14. Cao Z, Hidalgo G, Simon T, Wei SE, Sheikh Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. arXiv preprint arXiv:1812.08008.
15. https://github.com/cmusphinx/pocketsphinx Huggins-Daines D, Kumar M, Chan A, Black AW, Ravishankar M, Rudnicky AI. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings 2006 May 14 (Vol. 1, pp. I-I). IEEE.
16. Google Cloud Speech To Text https://cloud.google.com/speech-to-text/
17. Valin JM, Maxwell G, Terriberry TB, Vos K. High-quality, low-delay music coding in the opus codec. arXiv preprint arXiv:1602.04845. 2016 Feb 15.
18. Ptz S, Simn JS, Hertzberg J. Move base flex. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018 Oct 1 (pp. 3416-3421).
19. C. Rsmann, W. Feiten, T. Wsch, F. Hoffmann and T. Bertram: Efficient trajectory optimization using a sparse model. Proc. IEEE European Conference on Mobile Robots, Spain, Barcelona, 2013, pp. 138143
20. J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. arXiv, 2018
21. https://github.com/leggedrobotics/darknet_ros
22. Z. Liu, P. Luo, X. Wang, and X. Tang. Proceedings of International Conference on Computer Vision (ICCV) 2015 December
23. https://en.wikipedia.org/wiki/Color_difference#CIEDE2000
24. https://github.com/ageitgey/face_recognition