Austin Villa@Home 2020 Team Description Paper

Yuqian Jiang Haresh Karnan Gilberto Briscoe-Martinez
Dominick Mulder Ryan Gupta Stone Tejeda Justin Hart
Luis Sentis Peter Stone

November 1, 2019

Abstract. UT Austin Villa has participated in four RoboCup@Home competitions, performing respectably in each. What is more exciting, however, is that we have begun a strong program of research that has been in part inspired by our efforts in this competition. It is our intention to build a comprehensive service robot system which is used in our laboratories, in a real-world deployment in our CS Department, and to compete in RoboCup@Home. In this Team Description Paper, you will find the highlights of our efforts and our plans for 2020.

1 Introduction

Using the RoboCup@Home team as a focal point for inter-department and inter-laboratory collaboration, UT Austin Villa@Home has pursued an ambitious research program towards the goal of the development of a comprehensive service robot system. We want to enter RoboCup@Home not with a suite of different programs for each round, but with a single program which is capable of competing and winning.

UT Austin Villa@Home is a collaborative effort between PIs and students in the Computer Science, Mechanical Engineering and Aerospace Engineering departments at the University of Texas at Austin. We have competed in four RoboCup@Home events. In 2007, we took second place. In 2017, we entered into the newly-formed Domestic Standard Platform League (DSPL) and took third place, having received our robot only a couple of months before the competition. In 2018, the team developed a design intended to allow us to develop a single system which would enter into all of the stages of the competition, encompassing knowledge representation, mapping, and architectural aspects. The team advanced to the second stage and was able to score in difficult tasks such as Enhanced General Purpose Service Robot (EGPSR). In 2019, we improved the system with better perception and manipulation modules. While we were unable to demonstrate our capabilities fully at the 2019 competition due to hardware issues,[1] the progress made leading up to the competition would set a good starting for 2020 and open many research opportunities. Our efforts have resulted in five publications [1,2,3,4,5]. Going into 2020, we plan to further improve the core components of our system and develop more rigorous approaches to the tasks.

2 Software and Scientific Contributions

This section describes the component technologies we developed across multiple tasks for our robot architecture, knowledge representation, semantic perception, and object manipulation on top of the HSR software stack. To the extent possible, we built our approach in a manner consistent with our ongoing Building-Wide Intelligence project [6]. While using a different hardware platform, many of the objectives and capabilities are the same. Indeed we have previously designed an underlying architecture that is common to the two platforms [5].

2.1 Robot Architecture

Our architecture is designed for service robots to handle dynamic interactions with humans in complex environments. The three-layer architecture, as shown in Figure 1, outlines integration of the robot's skill components, such as perception and manipulation, with high-level reactive and deliberative controls. The top layer sequences and executes skills, and is reactive during execution to respond to changes. A central knowledge base facilitates knowledge sharing from all the components. The deliberative control layer uses the knowledge base to reason about the environment, and can be invoked to plan for tasks that cannot be statically decomposed. Details on implementation of these layers can be found in our recent paper [5].

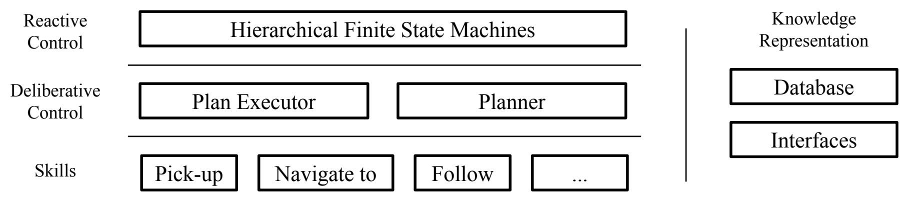

Fig. 1. Implementation of our robot architecture on HSR.

[1] The robot we shipped to the competition did not clear customs in time, and the replacement we were loaned had some faulty hardware.

2.2 Knowledge Representation and Planning

Our KR subsystem stores knowledge in a SQL database in order to allow for fast access and easy querying. Queries can be formed using a custom C++ library. The library can be interfaced through a simple predicate logic form which can be then imported for planning. Core to our KR subsystem is the ability to reason about hypothetical objects, as can be seen in our team's qualification video. In the video, when the person asks for a piece of fruit, the system is both able to reason about incomplete information (the non-specificity of "fruit") and to reason about a hypothetical apple that it has not yet witnessed. It is also able to address the potential inaccuracy of this information (there is no apple), and to report this error to the user. Details on our knowledge representation and planning system can be found in our recent paper [2].

2.3 Perception

We employ a semantic perception module whose purpose is to process raw video and depth data from the robot's sensors and extract information that can be processed by the manipulation, navigation, and knowledge reasoning modules. The main output representations are a query-able point cloud of objects in the environment and a partial 3D map of the world.

The main input to our semantic perception module is RGBD camera data. Compressed RGB and depth images from the robot are streamed to an offboard computer that runs the perceptual system. This image data is then consumed by finding objects via the YOLO object detection network [7], and constructing a point cloud. Next, semantic information about the world is synthesized in two main ways: a partial 3D environmental map and object cloud. For the former, regions of the point cloud corresponding to detected objects are fused together over time in a probabilistic Octree representation based on Octomap [8], which allows for the realtime construction of a partial 3D map of the world. For the latter, point estimates of the locations of objects are stored in a KD-Tree and wrapped with an efficient querying interface that integrates with our knowledge representation system. The synthesized semantic information is then made available to plugins in an event-based model, where a plugin can request access to semantic information that it wants to operate on. Plugins used include custom RANSAC edge detectors used to detect surfaces, and bounding box fitting on the 3D map for use in manipulation.

A significant limitation is the partial nature of the 3D environmental map. Only a partial map is constructed due to the realtime processing constraint; namely, full views of the world cannot be stitched together at framerate using the Octomap technology. Alternatively, GPU-based techniques for combining full point clouds could potentially overcome this limitation, and thus provides a direction for future development. Benefits of having full 3D environmental maps include the ability to directly localize objects with respect to the robot.

2.4 Manipulation

The purpose of our manipulation system is to enable the pick up and put down of diverse objects of different shapes and sizes. Our manipulation stack consists of three main components which we describe below: grasp pose generation, parallel motion planning, and closed-loop correction.

First, our semantic perception system provides 3D bounding boxes for objects worth manipulating. Based on these bounding boxes, potential grasp poses are computed that place the gripper on the top of the object as well as on all sides, with multiple possible rotations of the wrist. Of these poses, invalid configurations are filtered out by projecting the gripper onto the object and seeing if there is a collision.

Once grasp poses are determined, motion plans need to be determined in order for the robot to achieve a desired grasp pose. In order to do this quickly, we employ a parallel motion planning architecture built on top of the Moveit framework [9]. Our motion planning architecture is comprised of primary and secondary nodes. The secondary nodes handle generating motion plans for each potential grasp pose, while the primary node coordinates and handles executing motion plans. Specifically, secondary nodes plan in parallel, and the first motion plan found is what is executed. The rationale behind this is that different grasp poses will require different yet unknown amounts of time for finding motion plans. Since motion planning takes a significant amount of time, reducing this bottleneck greatly speeds up the entire manipulation pipeline. Furthermore, the Moveit framework can sometimes crash when trying to find plans. In our setup, this problem is mitigated: If a secondary node dies from such a crash, then the other secondary nodes are still present, allowing the system to continue functioning.

Next, executing a motion plan precisely is usually not feasible. This is because, as the plan is executed, the software solely uses odometry to control its position and the resultant drift can cause errors in how much the robot thinks it has moved. To overcome this obstacle, we slightly modify desired grasp poses by having the gripper be some offset away from the object. This way, after a motion plan is generated and executed, the robot's gripper is close to the object, but there remains a small gap. We take advantage of this small gap by employing a proportional controller based on object detections from the robot's hand camera to correct for odometry drift. This practically means that the robot shifts slightly to align the gripper perfectly with the centroid of the object. The gap is then closed by moving in a straight line towards the object, leading to a successful grasp.

3 2019 Task Approaches and Results

This section provides, for the main tasks we attempted in 2019, a summary of our approach, the challenges overcome, successes and limitations of the approach, and directions for future improvements.

3.1 Storing Groceries

Fast perception and manipulation are crucial in this time-constrained task, which has shaped our approach. First, the robot identifies and localizes the kitchen table by exploiting two known pieces of information: the location of the pantry cupboard and the height of the table. Namely, the robot navigates to the pantry cupboard, and then scans around for an edge that is exactly the height of the table. Next, as objects have been passively perceived throughout the previous step, our semantic perception system is queried for all objects that are on the table. Of these, a random object is chosen and pushed through our manipulation system which causes the robot to pick up the object. Subsequently, the pantry cupboard is localized in a similar way as the table, and we query for all objects that are currently in the pantry. The final component of the task is deciding where to place the grasped object in our gripper. The simplest case is when the knowledge base knows a priori that two objects are part of the same category (e.g. sprite and ginger ale). Otherwise, we use a word2vec [10] model fine-tuned on a custom corpus to decide the similarity between our grasped object and the objects in the pantry.

Our manipulation system is designed to work with a variety of object shapes and sizes. However, increased speed and reliability can come from exploiting the fact that most objects are quite small. Specifically, complex motion planning could be abandoned in favor of positioning the robot's gripper above the table and executing a simple motion downwards until the gripper hits the table. While not all objects can be picked up this way, many reliably can. Our qualification video shows the robot successfully executing the above behavior.

3.2 Take out the Garbage

This task relies mainly on quick navigation and manipulation, with emphasis on speed and accuracy. A particular challenge is precise localization near the trash cans. As the locations of the trash cans are known beforehand, the HSR is able to navigate to a position in the arena where it is directly facing the trash can. Once facing the trash can, the HSR reaches out its arm and points its arm with the hand camera facing directly downward. From there, a 2D bounding box of the target is generated by YOLO object detection on the hand camera image. If the lid is on the can, it will be detected and become the target for the HSR to grab. Otherwise, the trash bag with the trash can will be detected and used as the grab target.

The HSR uses the position of the generated 2D bounding box to align its hand with the target. A proportional controller is used to publish a velocity command to the robot base based on the distance between the center of the hand camera image and the center of the bounding box. Once this distance is within a certain tolerance, the hand is directly above the target and the velocity command is set to zero. With the height of the trash can also known ahead of time, the HSR can then lower its arm straight down until it is at the height of

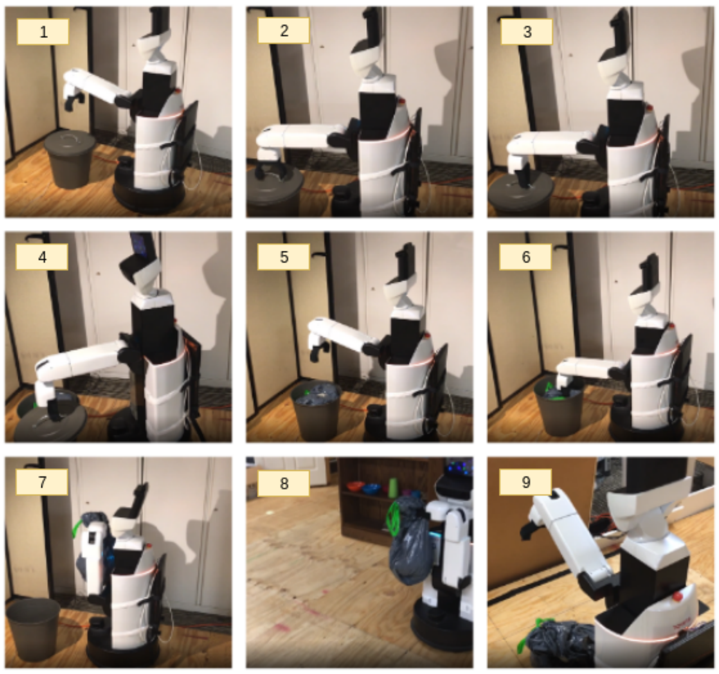

Fig. 2. Sequence of actions for taking out a trash bag to the collection zone. Ordered from top left to bottom right.

the lid handle. If the lid is already removed, the HSR will instead lower its hand down into the trash can to grab the bag.

Figure 2 shows the sequence of states for the robot to pickup and deposit trash. In Step 1 , the robot navigates to the trash can and places its hand facing downward. In Step 2 and 3, the robot performs closed loop control using proportional controller to reduce the 2D translational error between hand camera plane and trash can lid plane. The robot then picks up the trash can lid, followed by the trash bag in steps 4,5,6 and 7. In step 8, the robot navigates to the collection zone, navigating through the arena avoiding obstacles along the way. Finally in step 9, the robot deposits the trash bag. We noticed that in step 8, the trash bag sometimes blocks the HSR's LIDAR and hence the navigation stack is unable to charter a plan to the goal as a blocked LIDAR is equivalent to a static obstacle infront of the robot. To solve this problem, we raise the height at which the HSR holds the trash bag while transporting it. In Step 9, the robot performs a bidirectional roll motion on the wrist, which helps in dropping the bag from the gripper and placing it successfully on the ground. Our qualification video shows the above behavior including a pick up of one of the trash lids for extra points.

3.3 Serving Drinks

This task presents perception and manipulation challenges alongside HRI. First, the robot navigates to the bar to check which drinks are available. Once this is done, we utilize OpenPose [11] to detect and then navigate to people in the living room that require beverages. The closest person is asked for their name and drink order through Google Cloud's speech-to-text service. The speech recognition often misinterpreted the order or the name and to catch when a robot misheard their order, the person's speech was checked against a dictionary of rhyming words (e.g. to correct Santa to Fanta). However, this method has potential limitations should two drinks off the menu have similar names and improved recognition would facilitate human-robot interaction in this task. The robot proceeds to query our semantic perception system for the requested drink and then attempts to grab it. Ideally, the drink should be delivered to the same person that requested it, but facial recognition proved to be highly unreliable and therefore there was failure in delivering the drink back to the same person. Our qualification video shows the robot's execution of the above behavior.

3.4 Restaurant

As the most dynamic task, Restaurant requires navigation, perception, and manipulation in an unseen and chaotic environment. For increased reliability, we bypass all manipulation in this task and focus our efforts on the navigational and human interaction challenges this task has to offer. First, the bar is detected by asking the bartender to raise his or her hands. Next, waiting customers are detected by employing a velocity-based hand waving gesture classifier; arms from OpenPose skeletons are identified, and the velocity of the wrist relative to the shoulder is checked over a few frames. This allows the robot to see which customers are waving. Once detected, the customer must be maplessly navigated towards.

A challenge however is determining *where* to move to. After all, moving to exactly the customer's location would be equivalent to running them over, which would lead to immediate disqualification. Instead, just like a normal waiter, the robot should move close to the customer, such as right beside their table. Though, since we are in a previously unseen area, the robot has no knowledge of where tables are or what areas would be appropriate to move to. To that end, the robot looks at its local obstacle map, and finds the "island" the the customer is on. By island we mean an occupied region surrounded by free space. Generally, the customer, the chairs they are sitting on, and their table, will be an island surrounded by free space that the robot can move to. The shortest path to the customer is planned, and the farthest point on that path which does not collide with the island is where the robot moves to.

4 Conclusion

UT Austin Villa@Home has been a strong competitor and has a tradition of synergistic research our RoboCup@Home team and our other research efforts.

RoboCup@Home has become a driving force in robotics research at UT Austin. We look forward to seeing everyone in Bordeaux this summer.

References

1. Rishi Shah, Yuqian Jiang, Haresh Karnan, Gilberto Briscoe-Martinez, Dominick Mulder, Ryan Gupta, Rachel Schlossman, Marika Murphy, Justin Hart, Luis Sentis, and Peter Stone. Solving service robot tasks: Ut austin villa@home 2019 team report. In *AAAI Fall Symposium on Artificial Intelligence and Human-Robot Interaction for Service Robots in Human Environments (AI-HRI 2019)*, November 2019.
2. Yuqian Jiang, Nick Walker, Justin Hart, and Peter Stone. Open-world reasoning for service robots. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS 2019)*, July 2019.
3. Justin W. Hart, Rishi Shah, Sean Kirmani, Nick Walker, Kathryn Baldauf, Nathan John, and Peter Stone. Prism: Pose registration for integrated semantic mapping. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.
4. Justin Hart, Harel Yedidsion, Yuqian Jiang, Nick Walker, Rishi Shah, Jesse Thomason, Aishwarya Padmakumar, Rolando Fernandez, Jivko Sinapov, Raymond Mooney, and Peter Stone. Interaction and autonomy in robocup@home and building-wide intelligence. In *Proceedings of the AAAI Fall Symposium on Artificial Intelligence and Human-Robot Interaction (AI-HRI)*, October 2018.
5. Yuqian Jiang, Nick Walker, Minkyu Kim, Nicolas Brissonneau, Daniel S Brown, Justin W Hart, Scott Niekum, Luis Sentis, and Peter Stone. Laair: A layered architecture for autonomous interactive robots. In *Proceedings of the AAAI Fall Symposium on Reasoning and Learning in Real-World Systems for Long-Term Autonomy (LTA)*, October 2018.
6. Piyush Khandelwal, Shiqi Zhang, Jivko Sinapov, Matteo Leonetti, Jesse Thomason, Fangkai Yang, Ilaria Gori, Maxwell Svetlik, Priyanka Khante, Vladimir Lifschitz, et al. BWIBots: A platform for bridging the gap between AI and human–robot interaction research. *The International Journal of Robotics Research*, 36(5-7):635–659, 2017.
7. Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
8. Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.
9. David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.
10. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
11. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

HSR Software and External Devices [DSPL]

We use a standard Human Support Robot (HSR) from *Toyota*. No modifications have been applied.

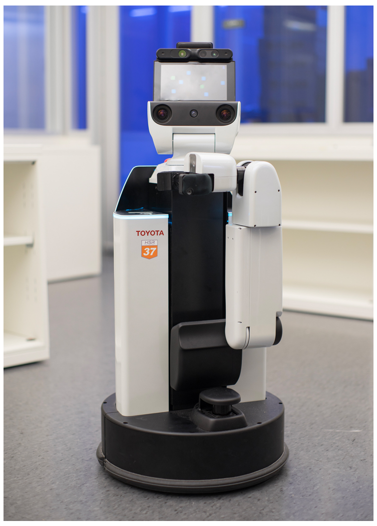

Fig. 3. HSR

Robot's Software Description

We are using the following 3rd party software:

- Object recognition: YOLO
- People and activity recognition: OpenPose
- Manipulation: MoveIt
- Planning and reasoning: Clingo (answer set solver)

External Devices

We are using the following external devices:

- Alienware 17 Laptop (Backpack)
- MSI Laptop (Backpack)

Cloud Services

We are using the following cloud services:

- Speech recognition: Google Cloud Speech API