KeJia: The Intelligent Service Robot for RoboCup@Home 2014

Xiaoping Chen, Dongcai Lu, Kai Chen, Yingfeng Chen and Ningyang Wang

Multi-Agent Systems Lab, Department of Computer Science and Technology,
University of Science and Technology of China, Hefei, 230027, China
xpchen@ustc.edu.cn, {ludc, chk0105, chyf, wny257}@mail.ustc.edu.cn
http://wrighteagle.org/en/robocup/atHome

Abstract. This paper aims at reporting the recent progress of research on and development of our intelligent robot KeJia. The long-term goal of this effort is to develop human-level intelligence for domestic robots. The effort covers research issues ranging from hardware design, perception and high-level cognitive functions. RoboCup@Home competition, besides other case studies, is taken as a main test-bed for these techniques and the whole robot system.

1 Introduction

More and more researchers in robotics and AI are showing their interest in intelligent robots [4, 5, 14, 18]. Research on intelligent service robots, which aims to fulfill a fundamental goal of Artificial Intelligence, is drawing much more attention than ever. Yet there are still challenges lying between the goal and reality. There are several essential abilities that a robot should have in order to make it intelligent and able to serve humans. Firstly, the robot should be able to perceive the environment through on-board sensors. Secondly, the robot has to independently plan what to do under different scenarios. Thirdly and most importantly, the robot is expected to be able to communicate with humans through natural languages, which is the core difference between service robots and traditional robots. As a result, developing an intelligent service robot requires huge amount of work in both advancing each aspect of abilities, and system integration of all such techniques.

The motivation of developing our robot KeJia is twofold. First, we want to build an intelligent robot integrated with advanced AI techniques, such as natural language processing [6], hierarchical task planning [7] and knowledge acquisition [5, 8]. Second, by participating RoboCup@Home League, all these techniques could be tested in real-world like scenarios, which in return helps the development of such techniques. In previous RoboCup@Home competitions, our robot KeJia got two 2nd places in 2013 and 2011, respectively. Other demo videos are available on our website[1].

[1] http://wrighteagle.org/en/demo/index.php

In this paper, we present our latest research progress with our robot KeJia. Section 2 gives an overview of our robot's hardware and software system. The low-level functions for the robot are described in Section 3. Section 4 presents techniques for complicated task planning and Section 5 elaborates our approach to dialogue understanding. Finally we conclude in Section 6.

2 Hardware Design and Architecture

The hardware architecture of our robot KeJia was designed in 2012 and has shown its stableness since RoboCup@Home 2012. Our robot is based on a two-wheels driving chassis. It is equipped with a lifting system that could adjust the height of its upper body quickly. A five degrees-of-freedom (DOF) arm makes our robot agile to fulfill manipulating tasks under indoor environments. It has a reach of over 83 centimeters and is able to hold a payload of up to 500 grams while fully extended. Our robot is about 1.6 meters height and weighs about 40 kilograms. For supporting the real-time environmental perception, our robot is equipped with a Kinect camera, a high-resolution RGB camera, two laser range finders and a microphone. The 20AH battery makes a guarantee that our robot could be reliably running in continuous applications. The computational capability of KeJia is powered by a laptop setting on the back of robot. The robot is shown in Fig. 1.

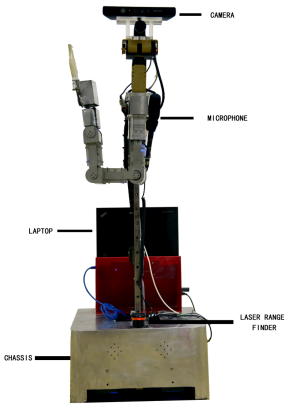

Fig. 1: The Robot KeJia

As for the software system, Robot Operating System (ROS)[2] has been employed as the infrastructure supporting the communication between modules in our KeJia robot. In general service scenarios, our robot is driven by human speech orders, as input of the robot's Human-Robot Dialogue module. Through the Speech Understanding module, the utterances from users are translated into the internal representations of the robot. These representations are in the form of Answer Set Programming (ASP) language [10] which is a Prolog-like logical language. An ASP solver is employed in the Task Planning module to automatically make decisions given the translated results. The Task Planning module then generates the high-level plans for users' tasks. The generated course of actions is fed into the Motion Planning module. Each action is designed as a primitive for KeJia's Task Planning module and could be carried out by the Motion Planning module and then autonomously executed by the Hardware Control module. A figure describing the architecture is shown in Fig. 2. In case of simple tasks or pre-defined ones, a state machine is used instead of the Task Planning module.

[2] http://www.ros.org/wiki/

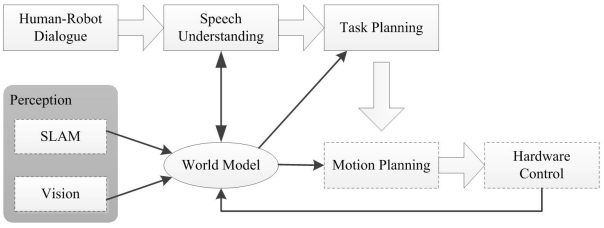

Fig. 2: Software architecture of KeJia

3 Perception

3.1 Self-Localization and Navigation

For self-localization and navigation, a 2D occupancy grid map is generated first from the raw data collected by laser scanners through a round travel within the rooms beforehand[11]. Then the map is manually annotated with the approximate location and area of rooms, doors, furniture and other interested objects. Finally, a topological map is automatically generated, which will be used by the global path planner and imported as a part of prior world model. With such map, scanning match and probabilistic techniques are employed for localization. Moreover, VFH+[19] is adopted to avoid local obstacles while the robot is navigating in the rooms. Frontier-based exploration strategy[21] and GMapping[11] algorithm are used to explore unknown environment. We also create the 3D environment representation using octo-tree structure[12], the system receive the point cloud information from the Kinect device, and then process the data with the localization provided by 2D grid map, eventually we get an effective and efficient 3d map, the map can be used in avoiding obstacles in all height and motion planning.

3.2 Vision

Sensors of our vision system consist of a *Microsoft* Kinect and a high-resolution 1394 RGB camera from *PointGrey*. With the pre-calibrated intrinsic and extrinsic camera parameters, we obtain an aligned RGB-D image by combining the RGB image from 1394 camera with the depth image from Kinect. With such aligned RGB-D image, our vision module is capable of detecting, tracking people and recognizing different kinds of objects.

People Awareness The aligned RGB-D image is transformed into the robot's coordinate using ROS *tf* API. Since human will occupy a continuous and almost fixed-size space, we segment the point cloud into multiple connected-components, and analyze the shape of each component. Each candidate is then passed into a pre-trained HOD [17] upper body detector to decide whether it is human or not. Then a HAAR [20] face detector from *OpenCV* [3] is used to find and localize human face. If present, the *VeriLook* SDK will be used to identify whether it is known via face recognition.

Object Recognition We follow the approach as proposed in [16] to detect and localize table-top objects including bottles, cups, etc. The depth image is first transformed and segmented, then the largest horizontal plane is extracted using Point Cloud Library (PCL) [15], and point clouds above it are clustered into different pieces. After that the SURF feature matching against the stored features are applied to each piece [1]. The one with highest match above certain threshold is considered as a recognition. At last, to further enhance the detection performance and decrease FP rate, we check each recognized cluster and filter out those vary too much in size. Recognition result is shown in Fig. 3.

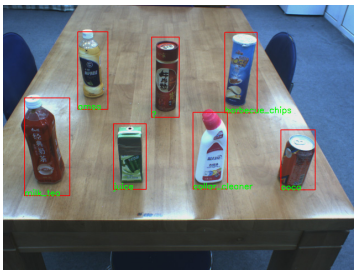

Fig. 3: Object recognition

4 Integrated Decision-making

One of the most challenging tests in the RoboCup@Home competition is GPSR, where a robot is asked to fulfill multiple requests from an open-ended set of user tasks. This ability is generally required for real-world applications of service robots. We are trying to meet this requirement by developing a set of techniques that can make use of open knowledge, i.e., knowledge from open-source knowledge resources, including the Open Mind Indoor Common Sense (OMICS) database, whose knowledge was input by Internet users in semi-structured English. This section provides a brief report on this effort.

In the KeJia project, the integrated decision-making [9] module is implemented using Answer Set Programming (ASP), a logic programming language with Prolog-like syntax under stable model semantics originally proposed by Gelfond & Lifschitz (1988). The module implements a growing model $\mathrm{M} = < A, C^*, P^*, F^* >$, the integrated decision-making mechanism, and some auxiliary mechanisms as an ASP program M^{Π}. The integrated decision making in M is then reduced to computing answer sets of M^{Π} through an ASP solver. When the robots Dialogue Understanding module extracts a new piece of knowledge and stores it into M, it will be transformed further into ASP-rules and added into the corresponding part of M^{Π}.

4.1 Representing growing models in ASP

Given any growing model $\mathrm{M} = < A, C^*, P^*, F^* >$, where A, C^*,P^*, and F^* represents the robot's action model and the conceptual, procedural and functional knowledge, respectively. All the components can be expressed in ASP with the following conventions. The underlying language includes three pairwise disjoint symbol sets: a set of action names, a set of fluent names, and a set of time names. The atoms of the language are expressions of the form $occurs(a, t)$ or

$true(f,t)$, where a, f and t are action,fluent, and time name, respectively. Intuitively, $occurs(a,t)$ is true if and only if the action a occurs at time t, and $true(f,t)$ is true if and only if the fluent f holds at time t. Based on these conventions, an ASP-rule is of the form

$$H \leftarrow p_1, ..., p_k, notq_1, ..., notq_m \tag{1}$$

where $p_i, 1 \leq i \leq k$, and $q_i, 1 \leq i \leq m$ are literals, and H is either empty or a literal. A literal is a formula of the form p or $\neg p$, where p is an atom. If H is empty, then this rule is also called a constraint. An ASP-rule consisting of only H is called an ASP-fact. An ASP program is a finite set of ASP-rules. There are two kinds of negation in ASP, the classical negation $\neg$ and non-classical negation not. Roughly, not q in an ASP program means that q is not derivable from the ASP program. Similarly, a constraint that $\leftarrow p_1, ..., p_k$ specifies that p $p_1, ..., p_k$ are not jointly derivable from the ASP program.

4.2 Integrated decision-making in ASP

Since any ASP solver innately possesses a general-purpose goal-directed planning schema, we embed a general-purpose task-directed action selection schema into the existing schema, so that the augmentation becomes a general-purpose decision-making mechanism that integrates both schemas and guarantees the executability of every plan it generates when there is sufficient knowledge for the corresponding task. Technically, the augmentation is built on the basis of M^{Π}.

First of all, we name a class of entities called sequence as follows: (i) an action a is a sequence; (ii) a task T is a sequence; and (iii) if $P_i(1 \leq i \leq m)$ are sequences, then $p_1; ...; p_m$ is a sequence. Let $\tau =< s_0, a_0, s_1, ..., a_{n-1}, s_n >$ be any trajectory. That τ satisfies a sequence p is defined recursively as follows:

1. If p = a, where a is an action, then $a_0 = a$;
2. If p = T , where T is a task such that there is an HRDS rule in P^* that decomposes T into a sequence of sub-tasks, then τ satisfies this sequence of sub-tasks, or where T is a task such that it is designated as a set of literals in F^*, then this set is a subset of the state s_n;
3. If p = $p_1; ...; p_m$, where $P_i(1 \leq i \leq m)$ are sequences, then there exist $0 \leq n^1 \leq n^2 \leq ... \leq n^{m-1} \leq n$ such that:
   - the trajectory $< s_0, a_0, ..., s_{n^1} >$ satisfies p_1;
   - the trajectory $< s_{n^1}, a_{n^1}, ..., s_{n^2} >$ satisfies p_2;
   - ...;
   - the trajectory $< s_{n^{m-1}}, a_{n^{m-1}}, ..., s_n >$ satisfies p_n;

According to the definitions above, if a trajectory $< s_0, a_0, s_1, ..., a_{n-1}, s_n >$ satisfies a sequence $a; a^{'}$ where a and $a^{'}$ are actions, and $a^{'}$ is not executable in s_1, then $a_0 = a$ and there exists a state $s_m(1 \leq m \leq n)$ such that s_m satisfies the preconditions of $a^{'}$ and $a_m = a^{'}$. In other words, the sub-trajectory $< s_1, a_1, ..., s_m >$ fills the gap between a and $a^{'}$.

A sequence S specifies how to complete a task T step by step. If a trajectory contains a sub-trajectory which satisfies S, then the corresponding task T is also completed in this trajectory. Now we consider how to specify a sequence S in ASP. Given an growing model M, we want to obtain a set of ASP-rules of S, Π_S, such that a trajectory $< s_0, a_0, s_1, ..., a_{n-1}, s_n >$ satisfies both M and S if and only if $\{true(\sigma, i) | \sigma \in s_i, 0 \leq i \leq n\} \bigcup \{\neg true(\sigma, i) | \neg\sigma \in s_i, 0 \leq i \leq n\} \bigcup \{occurs(a_i, i) | 0 \leq i \leq n-1\}$ is an answer set of $M^{\Pi} \bigcup \Pi_S$.

5 Dialogue Understanding

The robot's Dialogue Understanding module for Human-Robot Interaction contains Speech Recognition module and Natural Language Understanding module, it provides the interface for communication between users and the robot.

The Speech Recognition module uses the Speech Application Programming Interface (SAPI) which developed by iFLYTEK and on this basis, we have developed a Speech Recognition system for KeJia. Once a user's utterance is captured by the Speech Recognition module, it is converted into a sequence of words. The embedded dialogue manager then classifies the dialogue contribution of the input utterance by keeping track of the dialogue moves of the user. Fig. 4 shows our implementation (i.e., finite state machine) of managing a simple human-robot dialogue in which the user tells the robot facts that he/she has observed or tasks, and the robot asks for more information if needed.

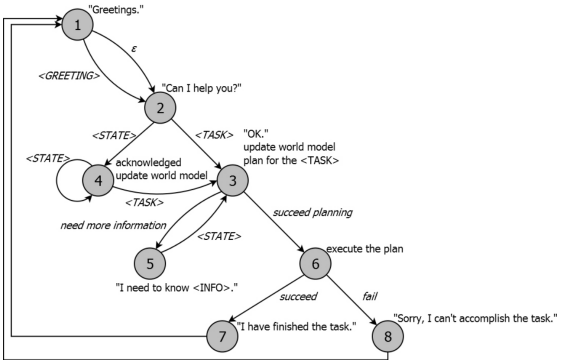

Fig. 4: The finite state machine for a simple human-robot dialogue

The natural Language Understanding module is used for the translation to its semantic representation. With the Speech Recognition module move and the semantic information of the speech, the Natural Language Understanding module decides to update the World Model, which contains the information from the perceptual model and of the robot's internal state, and/or to invoke the Task Planning module for fulfilling a task.

The translation from Speech Recognition Results to semantic representation consists of the syntactic parsing and the semantic interpretation. In the syntactic parsing, the Stanford parser [13] is employed to obtain the syntax tree of the speech. The semantic interpretation using λ-calculus [2] is then applied on the syntax tree to construct the semantics. Fig. 5 shows an example of semantic interpretation.

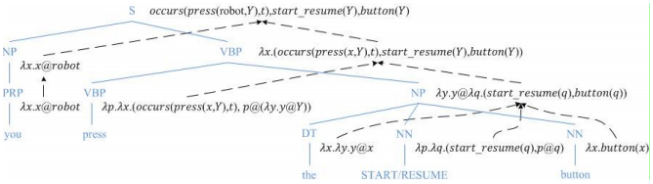

Fig. 5: An example of semantic interpretation

6 Conclusion

In this paper we present our recent progress with our intelligent service robot KeJia. Our robot is not only capable of perceiving the environment, but also equipped with advanced AI techniques which make it able to understand human speech orders and solve complex tasks. Furthermore, through automated knowledge acquisition, KeJia is able to fetch knowledge from open source knowledge base and solve tasks it has not met before.

Acknowledgments

This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150, the Natural Science Foundations of China under grant 60745002 and 61175057, the 985 project and the core direction project of USTC. Other team members beside the authors are: Zhiqiang Lin, Zhe Zhao, Zhang Liu, Wei Shuai and Jiangchuan Liu.

References

1. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
2. P. Blackburn and J. Bos. *Representation and inference for natural language: A first course in computational semantics*. CSLI Publications, 2005.
3. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
4. R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. Tell me when and why to do it!: Run-time planner model updates via natural language instruction. In *Proceedings of the 7th ACM/IEEE International Conference on Human-Robot Interaction*, pages 471–478. ACM, 2012.

5. X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie. Developing high-level cognitive functions for service robots. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 989–996, 2010.
6. X. Chen, J. Jiang, J. Ji, G. Jin, and F. Wang. Integrating nlp with reasoning about actions for autonomous agents communicating with humans. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 137–140, 2009.
7. X. Chen, G. Jin, and F. Yang. Correct reasoning. chapter Extending action language C+ by formalizing composite actions, pages 134–148. Springer-Verlag, 2012.
8. X. Chen, J. Xie, J. Ji, and Z. Sui. Toward open knowledge enabling for human-robot interaction. *Journal of Human-Robot Interaction*, 1(2):100–117, 2012.
9. X. Chen, J. Xie, J. Ji, and Z. Sui. Toward open knowledge enabling for human-robot interaction. In *Journal of Human-Robot Interaction*, pages 100–117, No. 2, 2012.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080, 1988.
11. G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.
12. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34(3):189–206, 2013.
13. D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL-03)*, pages 423–430. ACL, 2003.
14. S. Rosenthal, M. Veloso, and A. Dey. Learning accuracy and availability of humans who help mobile robots. In *Proceedings of the 25th Conference on Artificial Intelligence*, pages 60–74. AAAI Press, 2011.
15. R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.
16. R. B. Rusu, A. Holzbach, G. Bradski, and M. Beetz. Detecting and segmenting objects for mobile manipulation. In *Proceedings of the 12th IEEE International Conference on Computer Vision: Workshop on Search in 3D and Video*, 2009.
17. L. Spinello and K. O. Arras. People detection in RGB-D data. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843. IEEE, 2011.
18. M. Tenorth and M. Beetz. KnowRob-knowledge processing for autonomous personal robots. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266. IEEE, 2009.
19. I. Ulrich and J. Borenstein. Vfh+: reliable obstacle avoidance for fast mobile robots. In *1998 IEEE International Conference on Robotics and Automation*, volume 2, pages 1572–1577 vol.2, 1998.
20. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.
21. B. Yamauchi. A frontier-based approach for autonomous exploration. In *1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.