

Progress of Ke Jia Project

Xiaoping Chen, Jianmin Ji, Jiehui Jiang and Guoqiang Jin

Multi-Agent Systems Lab., Department of Computer Science and Technology,
University of Science and Technology of China, HeFei, 230027, China
xpchen@ustc.edu.cn
<http://wrighteagle.org/en/robocup/atHome>

Abstract. This paper describes some progress of the Ke Jia project, in particular as the team description of Wrighteagle for the RoboCup@Home 2010 Competitions. We report what we have done and what we are doing in the effort, including the hardware and the software modules such as those for NLP, planning, vision, and robot control, etc.

1 Introduction

RoboCup@Home is developing very fast to be a common testbed for domestic and service robots. It includes a wide spectrum of possible real-world applications and of technical and scientific challenges.

WrightEagle is the first Chinese team that ever entered the international RoboCup competitions and got good achievements in recent years¹. The WrightEagle@home team is established² with a two-fold goal. On one hand, we are trying to integrate some state-of-the-art Robotics and AI techniques into our home robot system, in order to test whether or not, or to which extent, these techniques are powerful enough for building a home robot in the settings of @home league. Currently we focus in this efforts on reasoning about actions, planning, natural language processing and computational linguistics. On the other hand, we are trying to investigate into the challenges from this league that introduce new chances and/or requirements to further Robotics and AI techniques, especially in those domains mentioned above.

The paper describes what we have done and what we are trying to do with Ke Jia. Section 2 and 3 describe the hardware and software architecture of keJia, respectively. Section 4 to 8 present the main modules of our robot system one by one. We make a summery in Section 9.

2 Hardware of Ke Jia

Compared to last year, we've made a big improvement in the hardware of Ke Jia, as shown in fig 1.

¹ See <http://wrighteagle.org/en/robocup>

² Other team members besides the authors are: Jiexi Chen, Min Cheng, Haoxiang Li, Feng Wang, Jiongkun Xie and Guangyu Zhang.

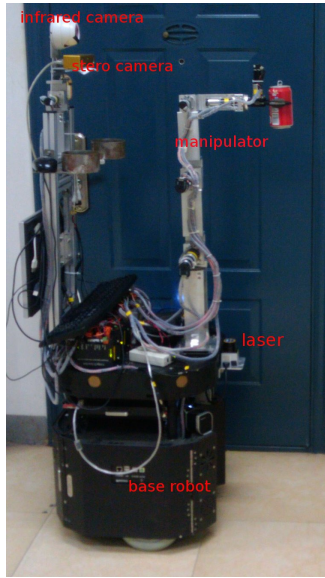


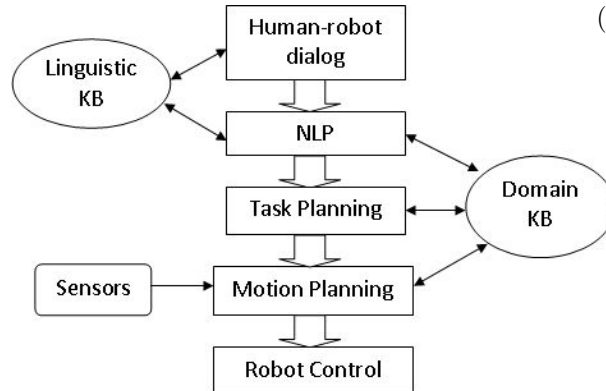
Fig. 1. The Hardware of Ke Jia

- 2 driving wheels with a DSP to control them.
- A hokuyo urg-04lx laser range finder for distance detection and obstacle avoidance.
- A pointgrey bumblebee 2 with pan-tilt unit used for face detection, object detect and manipulation.
- An Infrared camera which is used, for face recognition.
- A 6-DOF manipulator produced by a china company for manipulation.
- A Pentium-M 1.8GHz PC for motor the robot, obstacle avoidance and manipulation.
- Two laptops for face recognition and natural language processing.
- The size of the robot is 49.5cm x 48.0cm x 148.0cm, and the weight is about 45Kg.

3 Software Architecture

We proposed an approach based on the state-of-the-art NLP and common-sense reasoning techniques, integrated with human-robot dialog and motion planning. The main ideas are described below.

- (1) We take some *limited segments of natural language* (LSNLs [5]) as RHI languages. A specific LSNL is formed with a fixed vocabulary and a simplified syntax, a subset of the syntax of some natural language. With these LSNLs, service queries, descriptions about the states of environments, knowledge of the world, instructions about new tasks and so on can be expressed in similar ways as in everyday spoken language dialog and teaching at classes. Accordingly, we employ and develop some NLP techniques for the robot to “understand” the dialog in these LSNLs.
- (2) We introduced Answer Set Programming (ASP [11]) as knowledge representation and reasoning tool for Ke Jia. ASP is a logic language with Prolog-like syntax and the stable model semantics, and thus a non-monotonic reasoning mechanism [12]. This feature makes it suitable for handling underspecification and further supporting knowledge accumulation and human-robot collaboration through instructions.



- (3) We proposed a layered architecture (See left [7]) to integrate all the techniques and separate the task and motion planning. This is crucial for reducing computational costs, since current ASP solvers are not efficient enough for motion planning.

We are using MicroSoft Speech API 5.1 for speech recognition and synthesis. Currently, we use a small vocabulary and will try to enlarge it gradually.

4 Natural Language Processing

In order to make reasoning and planning in Ke Jia, input information in natural language (NL) is transformed to an inner representation. Our NLP system uses a Stanford parser [13] to extract a grammar tree from LSNLs. It returns three parts of information that may be used for further analysing: a tree structure representing the grammatical structure of the sentence, a part of speech (POS) tag on each node of the tree, and a set of grammatical relations depicting whether a phrase is the subject, object, or modifier of another phrase.

With a grammar tree, the semantic analyser walks over all words (leaf nodes of the tree) and checks their POS tags. For a verb or an adjective, there should be a predicate to carry its meaning, with arguments undetermined as “slots”. The arguments of the predicates should be entities of the sentence, which in turn should be described by the nouns and pronouns in the sentences. For each noun or pronoun, an entity is created as candidate for the filler of some slot in predicates.

Then the semantic analyser tries to find a right slot for each existing entity in the sentence, and fill slots with entities. As a result, each predicate expression composed by a predicate name and some slots is properly assigned arguments and becomes a meaningful predicate formula as defined in first-order logic. When the process is over, all the predicates are connected with proper connectives, resulting in a logical formula, and anaphora relations within the sentence are correctly handled, too.

Adverbs and some prepositional phrases modify verbs. Normally an adjective predicate encloses the entity it modifies as its argument, and normally an adverb predicate should have the verb it modifies as its argument, too. When a verb itself is already represented as a predicate, the problem seems to involve a second order language. To confine the solution in a controllable fashion, we use the representation of the Segmented Discourse Representation Theory (SDRT) [2], which assigns a label to each verb (in fact all words and phrases), and this label

is used to fill the argument slot of the adverbial modifier. The introduction of SDRT is the main innovation of this NLP system, while all the implemented semantic analysers are based on the Discourse Representation Theory (DRT) or the like.

An SDRT formula is not ready for efficient nonmonotonic reasoning, so further work needs to be done to transcribe SDRT formulae into strict first-order forms. The grammatical structure information is slightly affected, but essential information is totally preserved, and the output is the correct representation of the semantic meaning of the input sentence.

An usable version of the semantic analyser described above has been implemented, which can handle any single sentence from the given LSNLs. With realisation of SDRT, inter-sentence anaphora and some other complicated NL phenomena that cannot be resolved by DRT will be treated and Ke Jia would be able to make more complicated conversations in the settings of @Home tasks.

5 Task Planning

We follow the perspective proposed in [11] to use Answer Set Programming (ASP [3]) for the design and implementation of deliberative agents, which captures reasoning, planning and acting in a changing environment.

ASP has been used to tackle a variety of problems, including: diagnosis [8], planning [14], modeling and rescheduling of the propulsion system of the NASA Space Shuttle [15], multi-agent systems [4, 11] and Semantic Web and web-related technologies [16, 18].

ASP is a form of declarative logic programming, whose syntax is similar to standard Prolog and semantics is based on stable model semantics, which is a model based semantics for logic programs with negation as failure.

Specifically, an answer set logic program is a finite set of rules of the form:

$$H \leftarrow p_1, \dots, p_k, \text{not } q_1, \dots, \text{not } q_m, \quad (1)$$

where p_i , $1 \leq i \leq k$, and q_j , $1 \leq j \leq m$, are atoms, and H is either empty or an atom. If H is empty, then this rule is also called a *constraint*. A logic program with variables is viewed as shorthand for the set of all ground instances of its rules. The computational result of an ASP program is a set of answer sets[3].

For example, given the following program:

$$\begin{aligned} fly(X) &\leftarrow bird(X), \text{not } nfly(X). \\ nfly(X) &\leftarrow penguin(X). \\ &\leftarrow fly(X), nfly(X). \\ bird(tweety) &\leftarrow . \\ penguin(tweety) &\leftarrow . \end{aligned}$$

The only answer set of the program is $\{ penguin(tweety), bird(tweety), nfly(tweety) \}$.

The computation of a program is currently composed of two components, a *grounder* which removed the variables from the program by instantiation and

an *answer set solver* which computes answer sets of the propositional program. Lparse and Gringo are the grounders most commonly used, and clasp, cmodels, smodels, ASSAT and DLV represent the state of the art of solver development.

Following the proposal given in [14], the initial state, domain knowledge of the environment and descriptions of robot’s actions are declaratively expressed in ASP as the knowledge base of the robot. For example, the robot’s ability of ‘catch’ and the corresponding predicates *hold* and *empty* are expressed as follows:

$$\begin{aligned}
\text{catch}(A, T) &:- \text{not } n_catch(A, T), \text{small_object}(A), \text{time}(T), T < \text{lasttime}. \\
n_catch(A, T) &:- \text{not } catch(A, T), \text{small_object}(A), \text{time}(T), T < \text{lasttime}. \\
\text{hold}(A, T + 1) &:- \text{catch}(A, T), \text{small_object}(A), \text{time}(T), T < \text{lasttime}. \\
n_catch(A, T) &:- \text{location}(A, X, T), \text{not } \text{location}(\text{agent}, X, T), \text{small_object}(A), \\
&\quad \text{number}(X), \text{time}(T). \\
n_catch(A, T) &:- \text{not } \text{empty}(T), \text{small_object}(A), \text{number}(X), \text{time}(T). \\
\text{empty}(T + 1) &:- \text{empty}(T), \text{not } n_empty(T + 1), \text{time}(T), T < \text{lasttime}. \\
n_empty(T + 1) &:- \text{hold}(A, T + 1), \text{small_object}(A), \text{time}(T), T < \text{lasttime}. \\
\text{hold}(A, T + 1) &:- \text{hold}(A, T), \text{not } n_hold(A, T + 1), \text{small_object}(A), \text{time}(T), \\
&\quad T < \text{lasttime}. \\
n_hold(A, T + 1) &:- \text{empty}(T + 1), \text{small_object}(A), \text{time}(T), T < \text{lasttime}.
\end{aligned}$$

If the robot catches an object at time T , then she holds the object at time $T + 1$, but if the robot is not at the same position with the object or the hand of the robot is not empty, then she can not catch the object. The last four rules are inertia rules for predicates *hold* and *empty*, which concern the frame problem.

As an example of a goal of the robot, “give Jim the book”, can be expressed as follows:

$$\begin{aligned}
&g_give(\text{agent}, \text{Jim}, \text{book}) :- \\
&:- \text{not } \text{location}(A2, X, \text{lasttime}), \text{location}(A, X, \text{lasttime}), g_give(\text{agent}, A, A2), \\
&\quad \text{object}(A), \text{small_object}(A2), \text{number}(X). \\
&:- \text{not } \text{empty}(\text{lasttime}), g_give(\text{agent}, A, A2), \text{object}(A), \text{small_object}(A2).
\end{aligned}$$

The task “give Jim the book” is explained as the goal state: the object *book* is at the location of *Jim* and the hand of the robot is empty.

In the end, the task planning problem is reduced to the problem of finding an answer set of the ASP program. If the goal is achievable, then a sequence of actions (one action at each state) is contained in one answer set of the program, which stands for a plan to achieve the goal. We use Lparse to ground the program, and cmodels, one of the most efficient ASP solvers, to compute answer sets in our system.

Task planning is related to natural language processing and motion planning. User’s commands and provided information can be accessed from natural language processing in the form of facts. For example, the command “give Jim the

book” can be translated to the fact $g_give(agent, Jim, book)$. With the help of the part of the program which translates commands to the corresponding goal states, these facts can be directly added to the knowledge base of the robot, thus an answer set of the whole program is a solution to fulfill the command with the help of information provided by the user. The sequence of actions computed from the ASP program is then passed to motion planning part.

6 Motion Planning and Robot Control

A set of elementary actions are defined for Ke Jia’s motion planning. Each elementary action is pre-defined with a fixed set of parameters, similar to a command in command recognition in some aspects. Unlike commands, however, elementary actions are determined by the capabilities of a robot’s hardware to a great extent. An elementary action is full-specified in the sense that if only the values of parameters of an elementary action are assigned, the robot control module will execute the elementary action “blindly”. For instance, once the robot gets the position information of the bottle while performing the elementary action “catch the bottle”, it will try to catch the object in that position, no matter what object it is. In fact, it is not the “responsibility” of robot control module and/or elementary actions to identify the “right” objects for acting.

For each atomic action in a high-level plan generated by Ke Jia’s task planning module, the motion planning module will try to make a low-level plan composed of elementary actions, and execute the low-level plan autonomously. This is a special form of hierarchical planning introduced in Ke Jia for gaining its computational efficiency. Currently, we employ heuristic methods for Ke Jia’s motion planning due to the same reason.

Though, as is known that most of efforts on decision-making in this league is based on rules or logic, empirically, we believe that introduction of MDPs [17] techniques would be meaningful and worth of pursuing. The adopting of MDP in our 2D simulation achieves a lot [20, 9], but when porting it into physical robot, we come up against two problems: firstly, a model comprising information related to perception and actions is required, which is actually a hard and long-term job; secondly, how to simplify and optimize the process, and decrease the time in computation is also a hard work.

A simplified version of decision with probability has been tried out. Though the result is not good, we will continue the work on the aspect.

7 Simulation and Debug

We’ve developed a simulation system like robocup simulation 3D league’s sever, which simulates the robot and the environment to a large extent in real-time. So, we can test our motion control and decision-making before we port them to our robot. The ultimate effect is that we may directly employ the code in our robot with little modification. Now most functions have been realized.

For debug, firstly we record the log of each execution and do some replay when necessary. Secondly, we have a interface that can receive the real time sensor data and the global localization information of the robot when running in the real environment. And then, we draw it in our simulator, which act as a monitor when robot is running.

8 Vision

The vision system mainly consists of face detection & recognition, object detection & recognition and gesture recognition.

Face recognition[1] consists of 3 steps, first is face detection, then feature extraction and training, last recognition.

In detection, the robot does continuous analysis images from the camera, using the predefined feature to find faces. When detected, the robot give a brief greeting, and do some feature extraction of the face, compare the feature with the database, then give the results. Also the database is updated dynamically based on the results.

Object recognition[19] consists of 3 steps also. With fixed objects such as television, cupboard, which are sampled when building the scenario map, just goto the right place when navigation. With other small objects such as bottles, cups, still we do some feature extraction and training.

For now, we have done some simple gesture recognition (mainly hand gesture) base on [10], which is used as HRI command to the robot.

9 Summary

The modules described above have been implemented and the whole system was succesfully used in RoboCup China Open 2009. Also we've done some demos (See our demo page <http://wrighteagle.org/en/demo/>). Though, there are more challenges we must confront. One of them is large-scale knowledge acquisition, in particular for NLP and task planning. Another big challenge is the efficiency of general-purpose systems of planning and reasoning[6]. We believe that it is these and other challenges that makes @home league so appealing, remarkable, and fruitful.

References

1. T. Ahonen, A. Hadid, and M. Pietikainen. Face recognition with local binary patterns. *Springer*, pages 469–481, 2004.
2. N. Asher and A. Lascarides. *Logics of conversation*. Cambridge University Press, 2003.
3. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
4. C. Baral and M. Gelfond. Reasoning agents in dynamic domains. *Logic-Based Artificial Intelligence*, pages 257–279, 2000.

5. X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie. Developing high-level cognitive functions for service robots. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems(AAMAS 2010)*, To appear., 2010.
6. X. Chen, J. Ji, and F. Lin. Computing loops with at most one external support rule. *Proceedings of KR 08*, Sept. 16–19, 2008.
7. 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 (IAT-09)*, pages 137–140, 2009.
8. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. The diagnosis frontend of the dlvs system. *AI Communications*, 12(1-2):99–111, 1999.
9. C. Fan and X. Chen. Bounded incremental Real-Time dynamic programming. *IEEE Proceedings of FBIT 2007*, 2007.
10. W. Freeman and M. Roth. Orientation histogram for hand gesture recognition. In *Int'l Workshop on Automatic Face-and Gesture-Recognition*, 1995.
11. M. Gelfond. Answer set programming and the design of deliberative agents. In *Proceedings of Twentieth International Conference on Logic Programming (ICLP'04)*, pages 19–26, 2004.
12. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming (ICLP-88)*, pages 1070–1080, 1988.
13. D. Klein and C. Manning. Fast exact inference with a factored model for natural language parsing. *MIT; 1998*, pages 3–10, 2003.
14. V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.
15. M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry. An A-Prolog decision support system for the space shuttle. In *Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages*, pages 169–183, 2001.
16. A. Polleres. Semantic web languages and semantic web services as application areas for answer set programming. In *Nonmonotonic Reasoning, Answer Set Programming and Constraints*, 2005.
17. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.
18. M. Ruffolo, N. Leone, M. Manna, D. Sacc, and A. Zavatto. Exploiting ASP for semantic information extraction. In *Proceedings of the Third Intl. ASP'05 Workshop*, 2005.
19. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *In the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 905–910, 2001.
20. F. Wu and X. Chen. Solving Large-Scale and Sparse-Reward DEC-POMDPs with Correlation-MDPs. *Proceedings of RoboCup Symposium 2007*, July 2007.