

WrightEagle Team Description for RoboCup@Home 2009

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 WrightEagle team for the RoboCup@Home 2009 Competitions. We report what we have done and what we are doing in the effort, including the base robot 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 years (see below). Now WrightEagle@home team is established¹ 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.

Simulation2D	Simulation3D	4-Legged	MSRS	
2nd place	2nd place	2nd place	3rd place	RoboCup2008 Suzhou
2nd place	1st place	4th place	1st place	RoboCup2007 Atlanta
1st place	2nd place	Top 8		Robocup2006 Bremen

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

¹ Other team members besides the authors are: Jiexi Chen, Min Cheng, Haoxiang Li, Feng Wang, and Jiongkun Xie.

2 The Base Robot

The base of the robot that we use currently is provided by Ingenious², a Chinese robot company.



- 2 driving wheels with a DSP to control them.
- equipped with 6 ultrasonic transducer and 6 PSD for distance detection and avoidance.
- A two-freedom Sony EVI-D70P camera used for face detection, recognition and other object detection.
- The size of the robot is 49.5cm x 48.0cm x 48.0cm, and the weight is about 35Kg.
- A Pentium-M 1.8GHz PC used for fusing the sensor data and building the world model.
- Two laptops for face recognition and natural language processing.

We will be using an advanced version of this robot around April, 2009.

3 Software Architecture

Our Software architecture is shown in fig: 1.

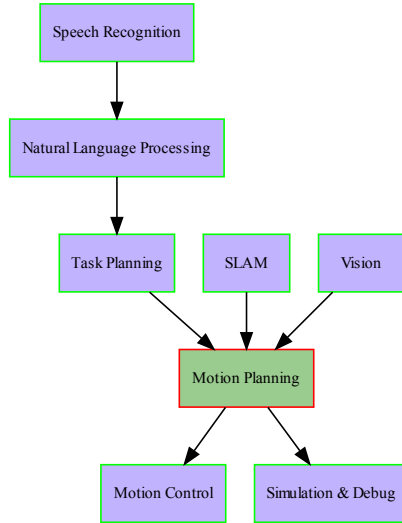
We are using MicroSoft Speech API 5.1 for speech recognition and synthesis. Currently, we use a small vocabulary and will try to enlarge it gradually.

4 Natural Language Processing

In order to make reasoning and planning in our home robot, input information in natural language (NL) is transformed to an inner representation. Our NLP system uses a Stanford parser to extract a grammar tree from an arbitrary given NL sentence. We are developing a new, more powerful semantic analyser, which recognises all predicates and entities in the sentence from the grammar tree and produces a logical formula representing the meaning of the sentence.

The system calls a parser[10] developed by Stanford University to get the grammar tree of an input sentence. 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,

² <http://www.ingenious.cn>



- Speech: from speech to text and vice versa.
- Natural Language Processing: from text to logic formulas.
- Task Planning: from logic formulas to tasks, also world modeling.
- Motion Planning: from tasks to control sequence.
- Motion Control: for smoothly action switch.
- Debug: global localization and simulation.
- Vision: for face (object) detecting & recognition , also gesture recognition.
- SLAM: dynamically map building with multi-sensor fusion.

Fig. 1. Architecture of Our Software System

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.

A first version of the semantic analyser described above has been implemented, which can handle any single sentence from the given small vocabulary. It will be upgraded to handle more complex sentences, with more robustness. With realisation of SDRT, inter-sentence anaphora and some other complicated NL phenomena that cannot be resolved by DRT will be treated and our robot would be able to make more complicated conversations in the settings of @Home tasks.

5 Task Planning

We follow the perspective proposed in [9] 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 [6], planning [11], modeling and rescheduling of the propulsion system of the NASA Space Shuttle [12], multi-agent systems [4, 9] and Semantic Web and web-related technologies [13, 15].

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:

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

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 [11], 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:

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

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:

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

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

Markov Decision Processes(MDPs[14]) is a popular framework in building dynamic systems and has been studied to support the development of WrightEagle 2D simulation team [7, 17].

For our home robot, suppose that the mapping is given by SLAM and the task is given by task planning. The motion planning can be carried out with MDPs techniques.

1. *States*: a state is a tuple, containing all the features which are necessary to capture a "snapshot" of the environment.
2. *Actions*: an action is characterised by a probabilistic transition function. Due to the imprecision, for any action a , there is a probability p (< 1) for a 's successful execution and probability $1 - p$ for its unsuccessful execution.
3. *Rewards*: The function of motion planning is to work out a course of action for each subtask generated by task planning. For this purpose, a certain reward is assigned to each state, which reflects how desirable the state is wrt the subtask at hand.

The basic decision process of our home robot is as follows. The agent analyzes the situation, makes sure the current state and confirms the goal state. The action generator generates many actions as candidates. Each candidate has its reachable states and the transformation probability. The long-term expected values of the states are computed by an approximate method iteratively. Based on these values, a course of action with the great expected utility is chosen.

Though, as is known that most of decision-making in this league is based on rules or logic, empirically, we believe that introduction of MDPs techniques would be meaningful and worth of persuing. The adopting of MDP in our 2D simulation achieves a lot[17, 7], 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 hark 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 Motion Control, Simulation and Debug

We've built a base system for motion control which can produce some basic motion such as "forward", "backward" and "gotoPosition" when given map and destination. Now the encoder of the moter mainly acts as the odometer. We've already done the calibration.

Motion control system contains the motion prediction, motion execution and motion feedback. motion prediction is used to smooth the motion execution, and meanwhile, motion feedback gives some localization information, also, it is a mainly aspect of motion execution.

We are intend to develop a simulation system which can represent the state of the robot in real-time, for this, we have modified the robocup simulation 3D league's server to simulate the sensor data. So, we can test our motion control and decision-making before we port it to our robot. The ultimate effect is that we may directly employ the code in our robot with little modification. Now the system is still in progress.

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[16] 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 [8], which is used as HRI command to the robot.

9 Summary

The first versions of most of modules described above have been implemented. However, neither NLP nor task planning module has been used actually in the current running for the demo (See <http://wrighteagle.org/en/robocup/atHome/>

for our qualification Video.). There are much more work to do even for making these modules to work, not less to say for our long-term goal. Besides what we mentioned above, 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[5]. 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, and F. Lin. Computing loops with at most one external support rule. *Proceedings of KR 08*, Sept. 16–19, 2008.
6. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. The diagnosis frontend of the dlvs system. *AI Communications*, 12(1-2):99–111, 1999.
7. C. Fan and X. Chen. Bounded incremental Real-Time dynamic programming. *IEEE Proceedings of FBIT 2007*, 2007.
8. W. Freeman and M. Roth. Orientation histogram for hand gesture recognition. In *Int'l Workshop on Automatic Face-and Gesture-Recognition*, 1995.
9. 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.
10. D. Klein and C. Manning. Fast exact inference with a factored model for natural language parsing. *MIT; 1998*, pages 3–10, 2003.
11. V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.
12. 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.
13. 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.
14. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.
15. 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.
16. 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.
17. F. Wu and X. Chen. Solving Large-Scale and Sparse-Reward DEC-POMDPs with Correlation-MDPs. *Proceedings of RoboCup Symposium 2007*, July 2007.