Controlling a Humanoid Robot in Home Environment with a Cognitive Architecture

KangGeon Kim, Dongkyu Choi, Ji-Yong Lee, Jung-Min Park and Bum-Jae You

Abstract— **Latest humanoid robots mimic the human body in realistic ways, but their control mechanisms fall short of the human mind. More often than not, the robots simply have a fixed set of routine codes that govern their behavior, and even the ones with some learning capabilities are not human-like. Since cognitive architectures aim for general intelligence and provide infrastructure for modeling human cognition, we can benefit from this line of research by adapting such architectures as control mechanisms in our robots. Recent studies revealed that this strategy is, in fact, very useful. Previously, we used one such architecture,** ICARUS**, to control a humanoid robot for Blocks World tasks. In this paper, we extend the system to perform more complex tasks autonomously on a similar platform. Despite the challenges we encountered in system integration, sensory updates, and navigation, the overall success of this application indicates that the system is capable of more complex tasks to provide services in home environments.**

I. INTRODUCTION

The field of humanoid robotics has recently made some great advances. Robots look more human-like, and they can perform more tasks at better precisions than before. However, we see considerably less advance in their control mechanisms. In many cases, these robots simply have a fixed set of routine codes that control their behaviors in rigid reaction to their surroundings. There are some learning systems, but often they are not based on psychological findings and it is extremely hard for a human to interpret the resulting knowledge that is acquired. We believe cognitive architectures [1] can be a promising solution to this problem, because they are software systems that aim for general intelligence and provide infrastructure for modeling human cognition. Humanoid robots can certainly benefit from the rich cognitive capabilities these architectures provide.

There is a long tradition of research on cognitive architectures. Over the years, researchers developed a variety of architectures, with Soar [2], ACT-R [3], EPIC [4], PRODIGY [5], and CLARION [6] being some famous examples. Some of these architectures are undoubtedly focused on a particular aspect of human cognition, but, in principle, they all aim for general intelligence providing infrastructure for modeling various human capabilities like perception, inference, performance, and learning. Many of them have sophisticated learning modules that acquire new knowledge from a variety of sources.

Previously, we used one such architecture, ICARUS [7], to control a humanoid, Mahru. The system handled blocks world tasks successfully in both the simulated environment [8] and the real world [9]. ICARUS controlled Mahru to build towers of different colors from an arbitrarily arranged set of blocks on a table. We intended this task to be a simplified version of dish arranging tasks found in a typical home environment. However, this previous work did not involve the robot moving from its standing position using its legged locomotion. Mahru used only its upper body to manipulate blocks on the table. In this work, we extend the domain significantly by adding the navigational component to the task. The robot should find its way to the table and complete the block sorting task once it gets there. We believe this richer domain is an important step toward a real home environment, providing the similar challenges the robot might encounter. For this task, we use a slightly different version of the robot, Mahru-M [10], that uses a wheeled platform instead of a two-legged locomotion.

As expected, there are many challenges to overcome for this extension. Unlike in our previous work where object sensing is restricted to the blocks on the table, Mahru should perceive any objects that are in front. Some of the objects are obstacles blocking its way to the table, while others are blocks sitting on the table that the robot can move. Especially, the obstacles require more dynamic recognition and reaction, since Mahru should avoid them while it is moving toward the table.

Once we resolve the issue related to sensing, we should program ICARUS for the navigation task, in addition to the existing program for block manipulations from our previous work. Based on the perceived information, the system should detect more complicated situations like its goal being in one direction and the immediate obstacle being in the other. The knowledge of such a situation then triggers ICARUS's selection among its general schemes for avoiding obstacles and moving toward its goal, which, in turn, issues more basic movements like left or right turns to the robot hardware.

In addition, the problem of the vision-based manipulation continues to challenge us. Synchronizing the sensor input, ICARUS's recognize-act cycles, and the robot's manipulation is a task that requires our constant attention. The current version of ICARUS has certain limitations on its speed of cyclic operation, and this requires adjustments on the robot side to coordinate movements.

This work was supported by the Global Frontier R&D Program from the National Research Foundation of Korea (Ministry of Education, Science, and Technology; NRF-M1AXA003-2010-0029746)

K. Kim, J.-Y. Lee, J.-M. Park, and B.-J. You are with Interaction and Robotics Research Center, Korea Institute of Science and Technology, Seoul, Korea. {danny, wisrage, pjm, ybj}@kist.re.kr

D. Choi is with Institute for the Study of Learning and Expertise, Palo Alto, California, USA. dongkyuc@uic.edu

In the next sections, we review ICARUS and Mahru-M platform briefly and describe the extended task domain. We then provide some detailed explanation of our strategy for the two tasks involved. We also provide results from our experiment before we conclude.

II. REVIEW OF ICARUS AND MAHRU-M

Previously, we developed an integrated system of ICARUS and the Mahru humanoid. We use ICARUS with a minimal set of modifications, and Mahru interfaces with the architecture through a TCP connection. This integration does not require any serious adaptations on the both sides, each of which we briefly review in this section.

A. ICARUS, *a Cognitive Architecture*

ICARUS is a framework that grew out of the cognitive architecture movement. It makes commitments to its representation, interpretation, and acquisition of knowledge, as well as the memory structures that support these activities. The architecture distinguishes conceptual and procedural knowledge bases, and it has separate memories for them. ICARUS further differentiates long-term and short-term knowledge such that a long-term conceptual memory stores the definition of its conceptual knowledge, or *concepts*, while a short-term conceptual memory houses instances of these concepts that are true in the current state. Similarly, a long-term skill memory stores the definitions of ICARUS' procedural knowledge, or *skills*, and another short-term memory records instantiated skills used for ICARUS's goals.

During runtime, the architecture operates in distinct cycles. On each cycle, it performs a series of cognitive processes as shown in Fig. 1. ICARUS first perceives its surroundings and deposits the sensory input in its perceptual buffer. Based on this information, the architecture infers concept instances that are currently true. Then it instantiates skills that are executable in the current state by matching the preconditions of them against the inferred concept instances. ICARUS's skills are hierarchically organized, and each skill instance can call upon another skill instance. As a result, ICARUS creates executable skill paths that consist of multiple skill instances at different levels of abstraction, rather than a single executable instance. Among these skill paths, the architecture selects a subset for execution following its selection criteria. The default criteria include logical precedence and recency.

Once an executable skill path is found, ICARUS executes actions implied by the skill path to change its environment. This, in turn, changes ICARUS's perception on the next cycle. It is notable that this cyclic operation gives reactivity to the architecture while maintaining the goal-oriented property. For this reason, ICARUS agents can readily adapt to unexpected situations or outcomes, for instance, an ICARUS-controlled robot can avoid obstacles dynamically even if they appear at some arbitrary times, and the robot can perform repeated trials if it fails to pick up a block.

B. *Mahru-M, a Mobile Humanoid Platform*

For our integrated system, we use a network-based humanoid, Mahru, with ICARUS. The version of Mahru we use in this paper has a mobile platform with wheels, and we call it Mahru-M (Fig. 2). This robot has 6-DOF for each arm, 2-DOF for its neck, and 3-DOF for each hand. Mahru-M operates in a network-based manner, and it sends various sensor data to external servers over the wireless network to perform high-level recognition, inference, and decision-making. It then receives control inputs from its servers to perform actions. This strategy facilitates updates for different applications on the server side, while maintaining the robot side relatively untouched. For example, we used two different PCs for the current work: a system with a Pentium Core 2 Duo T7700 2.4GHz CPU and 1GB RAM running Windows XP for sensory data processing and another with a Pentium M 1.73GHz CPU and 1GB RAM running Linux for ICARUS generating control inputs.

To process environment information, Mahru-M has three different types of sensors. A stereo camera is mounted in the robot's head to obtain depth images. The robot sends these images to its server for perceptual processing. It also uses the

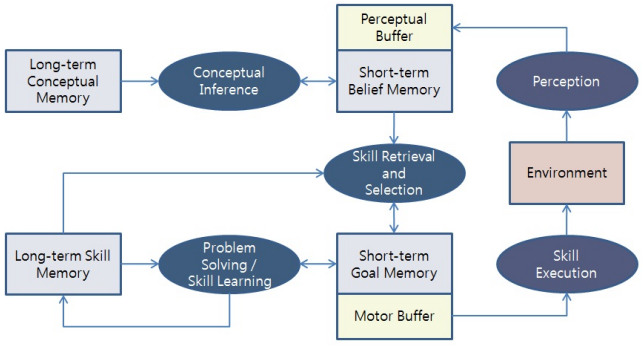


Fig. 1. ICARUS's cognitive processes on each cycle. Rectangular shapes denote memories and buffers, while oval shapes represent processes that work over them.

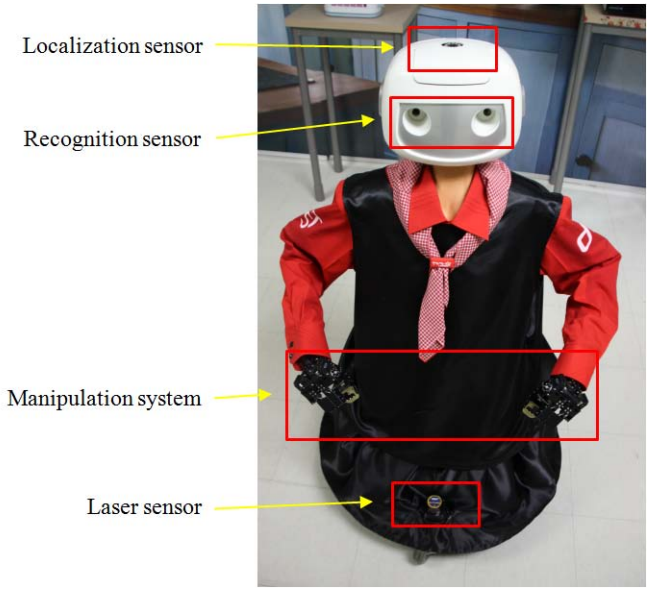


Fig. 2. Mahru-M, a mobile humanoid platform

system of an infra-red beam emitter and a matching image camera that detects and recognizes landmarks attached to the ceiling, allowing the computation of the robot's current location and heading. In addition, Mahru-M has a laser scanner attached to its mobile platform toward the front of the robot. This sensor detects obstacles within a designated range, providing the distance and angle of such objects. In the next two sections, we describe the task environment and our strategy to solve various challenges we encountered in the domain.

III. TASK ENVIRONMENT

Our goal is to simulate a typical home environment, and we devised a space with obstacles that resembles a living room and a connected kitchen. Initially, Mahru-M is standing on one side of this space, and we give it a goal to be on the other side where a table is. There are various objects blocking its direct path to the goal, and the robot should go around these obstacles to get to the table. Once Mahru-M arrives at the table, it should stack blocks of different colors into towers. We describe this task environment in detail below.

A. Navigation Task

Mahru-M starts at an initial location on one corner of the room. As shown in Fig. 3, the robot's goal is the table on the other side, on which blocks of different colors exist. It should navigate itself through a space filled with obstacles that simulate people moving around and various living room furniture like sofas and tables. Since these objects are movable, there is no fixed passage that is known to be safe in advance. The robot should react to the current situation and cope with the obstacles to avoid collision.

To facilitate the localization of the robot in this environment, we placed a landmark array on the ceiling. Fig. 4 illustrates the array used in our experiment, which consists of 30 landmarks and covers an approximate area of five meters by four meters. Each landmark is unique and acts as a local coordinate system, and the robot uses transform functions to convert locations in each local coordinate system into those in the global coordinate system.

B. Block Sorting Task

Once Mahru-M arrives at the table in the goal location after navigating around obstacles, the robot should stack blocks that are arbitrarily scattered on the table into towers of different colors. This task is similar in spirit to the blocks world task we have previously reported with another version of the Mahru humanoid [9]. We view this as a simulation of organizing tasks at home like dish sorting.

(a) (b)

Fig. 3. (a) The task environment with obstacles and (b) the goal position of the robot.

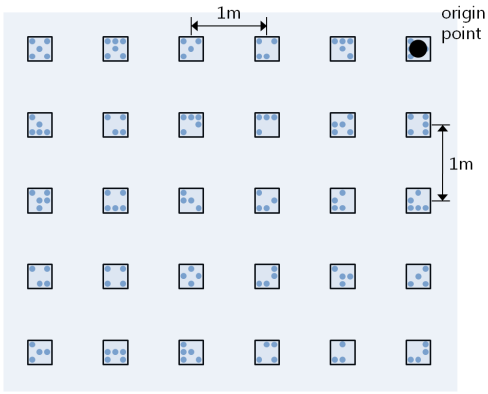


Fig. 4. The landmark array used to localize Mahru-M in the environment.

There are five blocks that are either red or blue. Initially, these blocks are randomly set on the table as shown in Fig. 5(a). Mahru-M should use its arms and hands to manipulate the blocks and stack them into two different colored towers like those shown in Fig. 5(b). The size of the table is comparable to the operating range of Mahru-M's manipulators. But the width is wider than what a single arm can reach, and the robot should switch hands to move a block from the left side of the table to the right side or vice versa.

(a) (b)

Fig. 5. (a) The initial and (b) the goal positions of Mahru-M for the block sorting task.

IV. CHALLENGES AND SOLUTIONS

While working to make Mahru-M capable of finishing the two tasks we just introduced, we encountered several challenges. These include implementing reactive navigation in ICARUS, synchronizing the recognition process between the robot and ICARUS, distinguishing obstacles and the table, and so on. In this section, we describe some of the most notable issues and our solutions to them.

A. Sensing for Navigation

There are at least two important aspects of reactive navigation we should consider. The first relates to the issue of sensing. Since Mahru-M should move toward its goal location without a map of the room, having localizing information on the robot itself and the goal in a global coordinate system helps the process considerably. Furthermore, the robot should detect any obstacles that block its way to the goal, so that it can avoid any collision.

(a) (b)

Fig. 6. A sample arrangement of obstacles (a) and the obtained image from the laser scanner (b)

For the localization problem, we use a StarGazer sensor system attached to the top of the robot's head. The system emits infra-red light onto the ceiling, where a landmark array exists. By recognizing the reflection from the printed patterns, Mahru-M compute its position and heading in the global coordinate system. The robot provides this information to ICARUS as attributes of its self-perception, so that the architecture can use them to infer about the current situation.

On the other hand, Mahru-M detects obstacles with a Hokuyo laser scanner[1] mounted on its mobile platform facing the front of the robot. The sensor measures the distance to any object that is blocking the view within the specified range of the robot's heading $\pm$ 90 degrees. Fig. 6 shows a sample arrangement of obstacles and the corresponding sensor output. A preprocessor filters out any noisy data and computes the distances and the angles of an obstacle's left and right extremes. It provides information on obstacles within the range of 0.3 to 2.0 meters from the robot to ICARUS as percepts.

ICARUS and Mahru-M communicates at an approximate speed of 5 Hertz. Upon request from the architecture, the robot assembles data from both the StarGazer sensor and the laser scanner and send the processed information as shown in Table I. Therefore, any changes to the situation are immediately included in the perceptual information provided to ICARUS at its next cycle, enabling ICARUS to perform reactive decision making.

TABLE I

SAMPLE PERCEPTS MAHRU-M PROVIDES TO ICARUS

```
(SELF ME X -0.26 Y -1.6719 HEADING 130.77)
(OBJECT OBJECT0 DIST1 0.825 DIST2 0.692
        ANGLE1 -31.644007 ANGLE2 -41.84041)
(OBJECT OBJECT1 DIST1 0.337 DIST2 0.396
        ANGLE1 -42.192005 ANGLE2 -85.790405)
```

B. Reactive Control for Navigation

Another important aspect of reactive navigation is the ICARUS program that provides rules for inference and decision making. Using its concepts about different obstacle configurations and its skills for maneuvers to avoid collisions, ICARUS reactively controls the robot to move it toward the given goal. The architecture has access to basic actions it can take in the world for the forward movement and turns without the control over speeds.

Table II show some sample ICARUS concepts for the navigation task. The first concept, *heading-to-target*, matches against Mahru-M's self-perception represented as an object of type *self*. It also checks the goal predicate as a sub-relation and retrieves the goal location. The concept then compares the vector direction from the robot's current location to the goal and the robot's heading, to see if Mahru-M is heading toward its goal. The second concept, *target-on-left*, uses the first concept as a negated sub-relation and performs an inequality test between the same vector direction and the heading. The last concept, *obstacle-avoidable-to-left*, matches against an object and retrieves the distances and the angles of its two extremities. It performs four different inequality tests to find objects that allow avoidance to the left. To prevent this concept from matching against the table at the goal location, a negated sub-relation is added.

TABLE II

SAMPLE ICARUS CONCEPTS FOR NAVIGATION

```
((heading-to-target ?me)
 :percepts  ((self ?me x ?x0 y ?y0 heading ?angle0))
 :relations ((goal ?x ?y ?angle))
 :tests     ((same-angle (vector-angle ?x ?y ?x0 ?y0)
                         ?angle0)))

((target-on-left ?me)
 :percepts  ((self ?me x ?x0 y ?y0 heading ?angle0))
 :relations ((goal ?x ?y ?angle)
             (not (heading-to-target ?me)))
 :tests     ((> (vector-angle ?x ?y ?x0 ?y0)
                ?angle0)))

((obstacle-avoidable-to-left ?me ?object)
 :percepts  ((self ?me)
             (object ?object dist1 ?dist1
              angle1 ?angle1 angle2 ?angle2))
 :relations ((not (table ?object)))
 :tests     ((< ?dist1 0.80)     (>= ?dist1 0.30)
             (> ?angle1 ?angle2) (>= ?angle1 -70.0)))
```

Meanwhile, Table III show some sample ICARUS skills for this task. The first skill, *heading-to-target* achieves the concept with the same name, namely, the first concept shown in Table II. The skill can start executing and continue to do so when the goal is to the left of the robot's current heading, and it invokes a left turn action. The second skill achieves *clear-front* when an obstacle is avoidable to the left side by executing a left turn. The remaining two skills are different from the first two, in that they call upon other skills instead of invoking actions directly. They provide an ordered list of subgoals that describe how these skills are executed in what order. For instance, the last skill specifies a strategy to avoid obstacles while moving toward the goal, which gives priority to collision avoidance over the movement to the goal by listing *clear* before *heading-to-target* in its subgoal specification.

C. Object Recognition for Block Sorting

Another important problem to solve in our extended task domain is that of recognizing individual blocks before

[1]For detailed specifications and characteristics of the Hokuyo laser scanner, see [11].

TABLE III

SAMPLE ICARUS SKILLS FOR NAVIGATION

```
((heading-to-target ?me)
 :percepts ((self ?me))
 :start    ((goal ?x ?y ?angle) (target-on-left ?me))
 :requires ((target-on-left ?me))
 :actions  ((*turn-left-target)))

((clear-front ?me)
 :percepts ((self ?me))
 :requires ((obstacle-avoidable-to-left ?me ?object))
 :actions  ((*turn-left-obstacle)))

((clear ?me)
 :percepts ((self ?me))
 :subgoals ((clear-front ?me) (clear-side ?me)))

((at-target-location ?me)
 :percepts ((self ?me))
 :start    ((goal ?x ?y ?angle))
 :requires ((obstacle-avoidable-to-left ?me ?any))
 :subgoals ((clear ?me)
            (heading-to-target ?me)
            (at-target-location ?me)))
```

Mahru-M can start manipulating them. The robot should provide the position, the pose, and the color of each block to ICARUS, so that the architecture can perform the block sorting task properly.

For this, the robot has a bumblebee stereo camera and uses the vision information to detect individual blocks on table. We put color-specific patterns on top of each block as shown in Fig. 7. By analyzing the stereo image, our vision algorithm can detect the color of each block from the type of the pattern, get the position from the location of the pattern, and recognize the pose of each block from the direction of the pattern. To implement this algorithm, we use ARToolKit, an open source library for pattern recognition [12].

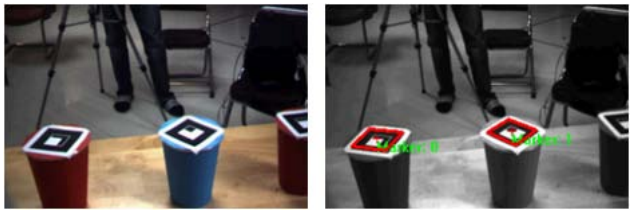
Although this algorithm reliably detects different blocks, the process is relatively time consuming. For this reason, we do not invoke the algorithm every time ICARUS requests perceptual input. Instead, the system updates only the information on the recently moved block to verify that Mahru-M manipulated it without any issues. When the system detects the block at its intended location with some predefined tolerance, the algorithm limits the update to the recently moved block. However, when it detects an error and the block is not at the intended location, the system invokes a complete recognition process from scratch to provide up-to-date perceptual information to ICARUS. For more details, see our previous work [9] that takes a similar approach.

V. EXPERIMENTAL RESULTS

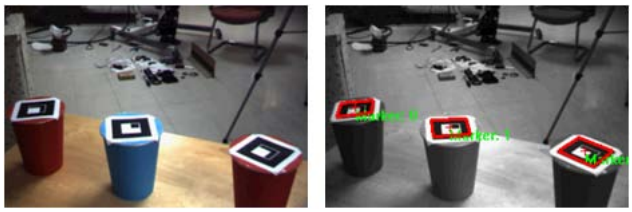
To verify that our solutions work properly for the two tasks in this extended domain, we performed a series of trial runs. In this section, we show a typical example of such trials.

Our basic strategy for obstacle avoidance is to move in the opposite direction from the detected obstacle. Fig. 8 and 9 show the result from a typical navigation trial. When Mahru-M starts its navigation task, it detects a collection of obstacles ahead to the left (a). ICARUS commands the robot to turn right until it sees the obstacles to its left side (b).

(a) right side of the table



(b) center of the table



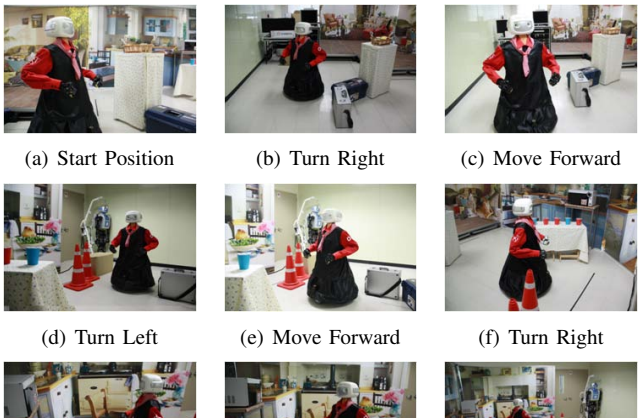
(c) left side of the table

Fig. 7. Sample patterns on our block. Mahru-M recognizes the color, the position, and the pose of blocks using these patterns. The original color images are shown on the left column and the recognition results are shown on the right column.

Then the architecture moves Mahru-M forward to avoid these obstacles (c). But the robot detects another set of obstacles in the front, and ICARUS turns to the left to avoid them (d). Once these obstacles are to the right of the robot, the architecture commands Mahru-M to move forward (e). When the path ahead becomes clear, ICARUS turns the robot to the right in the direction of the table (f). Once the robot reaches the table, it turns left to face the table (g).

When Mahru-M is in front of the table, ICARUS starts the block sorting task. Fig. 10 shows a series of images that captures this task. Initially, the robot recognizes five different blocks on the table that are set at random (a). It picks up the blue block in the center (b) and stacks the block on top of another blue block to its right (c). It continues the similar procedure for red blocks (d through i).

Despite the deliberate control of the robot, ICARUS sometimes faces unexpected outcomes like failures in grasping a block or putting a block at a target location. In such cases, the architecture reacts to the unexpected situation and makes persistent attempts to fix the failure, or proceeds with a different skill that is known to work in the new situation. Such failures cause an inevitable increase in ICARUS's cognitive cycles to complete the task. For instance, at 10% probability of action failures, it took 17.1 cycles on average for ICARUS to complete the block sorting task, which is an increase of approximately 30% from fail-free situations.



(a) Start Position (b) Turn Right (c) Move Forward

(d) Turn Left (e) Move Forward (f) Turn Right

(g) Turn Left (h) End Position (i) Whole View

Fig. 8. Mahru-M is performing the navigation task.

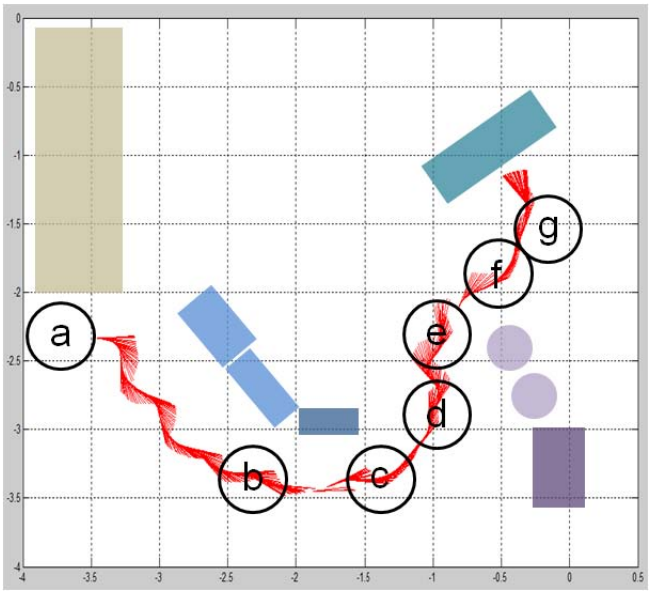
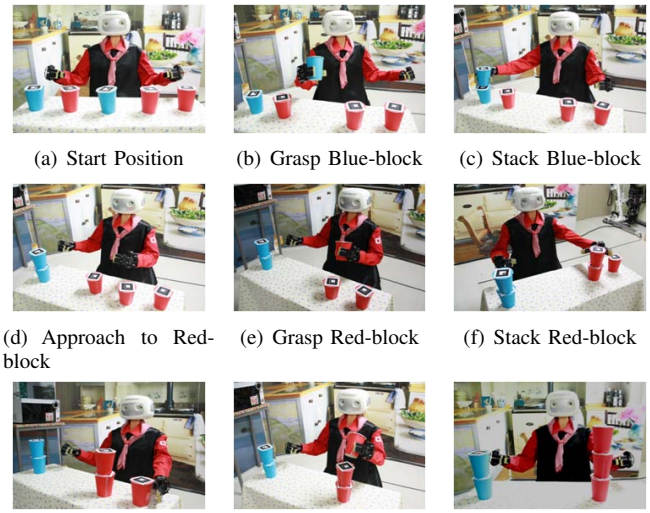


Fig. 9. Mahru-M's moving path. The alphabets represent the position of the robot in Fig. 8. The red lines show the heading of the robot.

VI. CONCLUSIONS

Our previous work reported a successful attempt to use a cognitive architecture, ICARUS, to control a humanoid robot for a blocks world task. In this paper, we used an extended task environment that resembles typical homes and challenged the architecture with a more complex set of tasks that include navigation and block sorting. Despite a variety of problems in sensing, localization, and reactive control, we programmed ICARUS to perform reactive navigation through a space with obstacles and finish the block sorting task. This is a promising result that shows the potential use of cognitive architectures with more human-like capabilities for humanoid robots that operate in home environment.



(a) Start Position (b) Grasp Blue-block (c) Stack Blue-block

(d) Approach to Red-block (e) Grasp Red-block (f) Stack Red-block

(g) Approach to Red-block (h) Stack Red-block (i) End Position

Fig. 10. Mahru-M is performing the block sorting task.

REFERENCES

[1] A. Newell, *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA; 1990.

[2] J. Laird, P. Rosenbloom, and A. Newell, Chunking in Soar: The anatomy of a general learning mechanism, *Machine Learning*, vol. 1, 1986, pp. 11-46.

[3] J.R. Anderson, *Rules of the Mind*, Lawrence Erlbaum, Hillsdale, NJ; 1993.

[4] D. Kieras, and D.E. Meyer, An overview of the EPIC architecture for cognition and performance with application to human-computer interaction, *Human-Computer Interaction*, vol. 12, 1997, pp. 391-438.

[5] S. Minton, J.G. Carbonell, C.A.Knoblock, D.R. Kuokka, O. Etzioni, and Y. Gil, Explanation-based learning: A problem solving perspective, *Artificial Intelligence*, vol. 40, 1989, pp. 63-118.

[6] R. Sun, *Duality of the Mind: A Bottom-up Approach Toward Cognition*, Lawrence Erlbaum, Mahwah, NJ; 2002.

[7] P. Langley, D. Choi, and S. Rogers, Acquisition of hierarchical reactive skills in a unified cognitive architecture, *Cognitive Systems Research*, vol. 10, 2009, pp. 316-332.

[8] D. Choi, Y. Kang, H. Lim, and B.-J. You, "Knowledge-based control of a humanoid robot", *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 2009.

[9] K. Kim, J.-Y. Lee, D. Choi, J.-M. Park, and B.-J. You, "Autonomous task execution of a humanoid robot using a cognitive model", *in Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Tianjin, China, 2010.

[10] K. Kim, J.-Y. Lee, S.-J. Kim, M.-H. Jeong, and B.-J. You, "Network-based humanoid operation in home environment", *in Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Tianjin, China, 2010.

[11] L. Kneip, F. Tache, G. Caprari, and R. Siegwart, "Characterization of the compact Hokuyo URG-04LX 2D laser range scanner", *in Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 1447-1454.

[12] "ARToolKit", http://www.hitl.washington.edu/artoolkit/