The Memory Game: Creating a human-robot interactive scenario for ASIMO

Victor Ng-Thow-Hing, Jongwoo Lim, Joel Wormer, Ravi Kiran Sarvadevabhatla,
Carlos Rocha, Kikuo Fujimura, and Yoshiaki Sakagami
Honda Research Institute USA
Mountain View, CA, 94041, USA
{vngthowhing,jlim,jwormer,rsarvadevabhatla,crocha,kfujimura,ysakagami}@hra.com

Abstract— **We present a human-robot interactive scenario consisting of a memory card game between Honda's humanoid robot ASIMO and a human player. The game features perception exclusively through ASIMO's on-board cameras and both reactive and proactive behaviors specific to different situational contexts in the memory game. ASIMO is able to build a dynamic environmental map of relevant objects in the game such as the table and card layout as well as understand activities from the player such as pointing at cards, flipping cards and removing them from the table. Our system architecture, called the Cognitive Map, treats the memory game as a multi-agent system, with modules acting independently and communicating with each other via messages through a shared blackboard system. The game behavior module can model game state and contextual information to make decisions based on different pattern recognition modules. Behavior is then sent through high-level command interfaces to be resolved into actual physical actions by the robot via a multi-modal communication module. The experience gained in modeling this interactive scenario will allow us to reuse the architecture to create new scenarios and explore new research directions in learning how to respond to new interactive situations.**

I. INTRODUCTION

Recently, humanoid robotics have begun to demonstrate an array of impressive motor skills, such as climbing stairs [1], whole body manipulations of large objects [2], and dancing [3]. On the perceptual side, continued effort has been invested in object recognition and geometry reconstruction to provide robots a model of its environment. The goal for many autonomous robots is to be able to use many of these component motor and perceptual skills to accomplish complex tasks involving interaction with both objects and humans in the environment. This is especially important for humanoid robots because of their anthropomorphic body design and capability to perform many different tasks.

In addition to the motor and perceptual problems, the robot needs to be able to perform decision-making tasks that allow it to map pattern recognition modules on the perception side to the behavior generating components on the motor side. This involves the need to recognize situational context and requires a communication subsystem for interaction with humans. For humanoid robots, speech and gesture are the main modalities available.

The ability to perform complex tasks requires the support of an established software architecture to ease integration and inter-communication of these disparate components. The successful execution of these tasks requires a system-level approach rather than focusing on individual components. Decision-making modules must make use of information or events from pattern recognition perceptual modules and be able to select appropriate behavior for the robot. Furthermore, even though components may operate error-free when tested in isolation, they can suffer unforeseen failures when operating concurrently with other modules. For example, any head-related motion may jeopardize the robustness of computer vision algorithms that assume a static input image. For researchers, the integration process should be simple and require minimal modification to existing software. The architecture should also encourage components to be reusable, modular and exchangeable to maximize flexibility and extensibility of the robot's capabilities. On the performance side, the architecture should be fault-tolerant so that the failure of any component module will not require restarting the entire system. Recently, we have developed a blackboard-based architecture known as the Cognitive Map [4] to meet these needs.

In this paper, we present our attempt to utilize our system architecture and component modules to create a well-define human-robot interactive scenario that demonstrates situational context recognition, perception and reconstruction of the environment and multi-modal communication via speech and gestures, including deictic gestures that make use of spatial information obtained from the perceived environment. This scenario is realized in the form of a memory card game between Honda's humanoid robot ASIMO [1] and a human opponent.

A. Memory Game Problem

The *Memory Game* features a deck of matching pairs of cards that are shuffled and placed face down on a table. Players take alternate turns picking two cards in an attempt to find and collect matching pairs. If a player successfully finds a match, he or she can take another turn. If they do not find a match, the other player starts his or her turn. When all the cards are gone, whoever has the most pairs of matches wins. A tie can occur if both players have an equal number of card pairs. In our scenario, we feature ASIMO playing against a human player, where the cards can be arranged anywhere on a table.

The choice of modeling the memory game as an interactive scenario was made due to its relatively simple rules, strategies and goals. The game requires a robust perceptual component to detect and recognize the identity of cards. ASIMO must communicate with the player using speech or gestures, including asking the player to flip cards on its behalf since ASIMO's manipulators are not dexterous enough to turn over flat cards on a table. ASIMO's perception is restricted to the two cameras located in the head. This not only makes the system more portable to different environments, but it motivates embodied behaviors such as ASIMO having to look down at the table to see the cards. The one exception to the self-contained ASIMO constraint is our use of an external projector to highlight cards that ASIMO would like to flip over (Section III-E).

1) Reusability of architecture and modules: Having a concrete goal of the memory game enabled us to focus on the desired features needed for our system architecture. Modules were written in different computer languages and on different operating systems and needed to communicate with each other. The nature of interactions dictated the type of messages that modules must pass to each other. It is important to make sure the architecture and modules do not become too specialized to the memory game as we wish to reuse the system for other types of interactive applications in the future. This criteria influenced the choice of design and selection of modules to use. For example, our card detector was deliberately designed to be unaware of any game-specific rules.

2) Interaction characteristics: Another goal was to consider robustness in the memory game on several levels. For example, perceptual robustness is required as cards change orientation or position on the table and undergo different illumination changes. Table detection should be robust to small changes in ASIMO's head orientation while it executes body gestures. In the memory game itself, ASIMO needs to focus on activities of interest, while ignoring unimportant events. The player should be free to communicate with ASIMO using natural means such as speech and gestures.

We did not want to create a scripted experience where ASIMO only reacts as a result of speech or activities from the user. ASIMO should be able to proactively perform behaviors while waiting for the player to take his or her turn. This includes asking players for help if they are taking too long to pick a card. It also includes expressing the same semantic message different ways to avoid monotony. For example, if ASIMO is winning the game, it may choose to say things in a more gentle manner so as not to upset the player. To solve these problems, we provide timeout mechanisms for ASIMO to proactively initiate behaviors, idle motions such as nodding acknowledgment whenever the player speaks to ASIMO, and a multi-modal communication module to generate new phrases and gestures based on requests for semantic message communication from other modules.

B. Previous Work

Communication between humans and embodied agents have been explored in virtual reality and human-robot interaction. Virtual reality systems featuring synthetic actors have the advantage of more expressive computer graphics and perfect knowledge of the environment [5]. In contrast, human-robot interaction requires the additional problem of developing robust sensing and control algorithms. The Kismet system [6] specialized in facial expressions and movement for short-term interactions. There have been several robotic systems designed for long-term interaction with humans. The HERMES system [7] was operated over a period of six months in a public museum. It performed tasks such as pouring water into a glass, explaining museum exhibits and engaging in dialogs with people. Valerie the Roboceptionist [8] is a stationary robot designed to interact with visitors over an extended period of time through social dialog. Although the robots are active for long periods of time in these systems, the duration of task interactions were relatively short. For example, the robot would respond to a query or command from the user and perform the task, ending the interaction and subsequently waiting for the next interactive event. These tasks typically require a small number of states to characterize them. In contrast, the memory game requires the modeling of many states or contexts in order for the robot to determine appropriate behavior, including exceptions where a player may ask for help or ASIMO proactively engages the player.

II. System Overview

Details of our system architecture are described in [4]. However, several details are reviewed here for completeness. The implementation of the memory game can be considered to be a multi-agent system, where the component modules act as independent agents that can process information concurrently with other modules. The agents can communicate with each other using messages or data streams through a shared information space using a blackboard architecture [9]. Component modules can be distributed on different machines and operating systems or co-located on the computer, subject to computational load constraints. Since communication is restricted to specific message types, different implementations of a module can be substituted as long as the same message types are posted and subscribed to. Modules can be stopped and restarted (in case of failure) without requiring other modules to restart. Figure 1 depicts all the components of our memory game system.

Messages are typically used to denote events in the system and are accompanied by information about the properties of the related object (if applicable). For example, the card recognizer reports messages: *card-flip*, *card-appear*, *card-remove*, *card-touch* to notify modules of the respective activities involving cards on the table. Each event is accompanied by a data object (called a CMobject) that describes its properties. CMobjects can also represent commands not involving any physical objects such as *Say-text* and *Do-Task* messages for text-to-speech and task commands respectively.

In general, modules should have a clearly defined role in the system. For the memory game, modules usually fall into one of three categories: pattern recognition, decision-making,

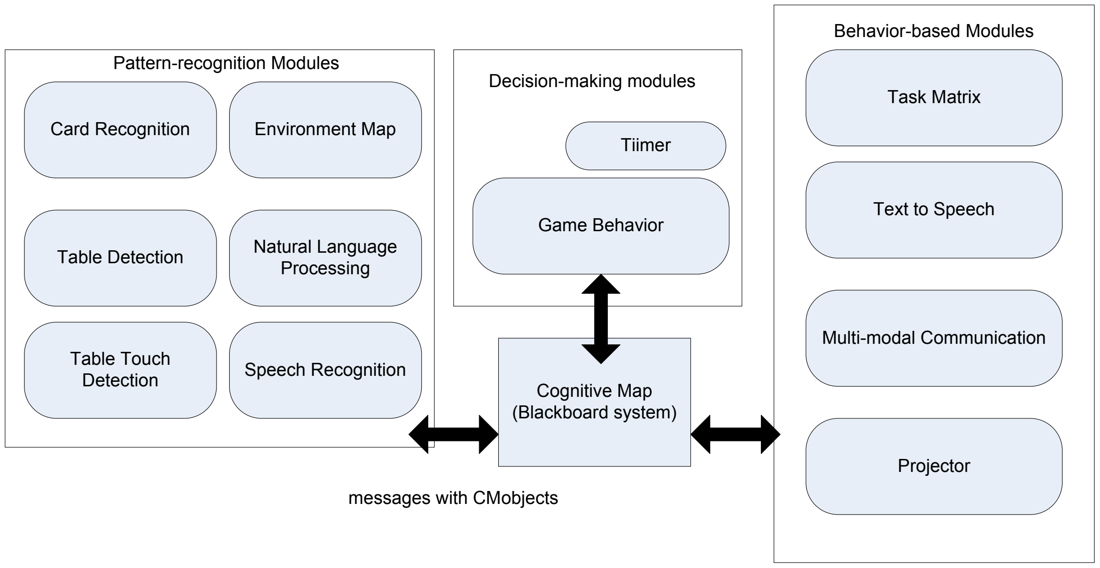

Fig. 1. Cognitive-map multi-agent architecture for the Memory Game

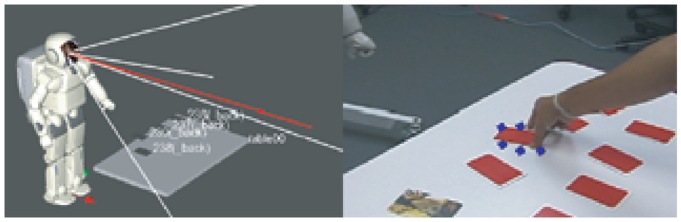

Fig. 2. Left: Environment map, Right: Projector highlighting of card (enhanced for easier visibility)

and behavior-based. Pattern recognition modules typically consist of modules which take sensor input and produce perceptual features in the scene, such as card location and identity. Pattern recognition modules can also work with temporal streams of information to detect activities such as card removals or card flips. Decision-making modules accept events posted from the pattern recognition modules and consider internal state of the robot and game to generate appropriate behavior. Behavior modules can produce perceivable robot activity such as speech utterances, gestures, or other complex motor tasks. Figure 1 illustrates the different categories of the modules used in the memory game. Information exchange can happen in both directions between any modules in the system. In Section IV, we describe the communication flow for several situations that can occur during the game. The next section describes each module used in the system in more detail.

III. Modules

An interactive scenario like the memory game requires a variety of different component functionalities that when assembled together enables complex behavior of the robot with its environment.

A. *Environment Map*

The Environment Map collects object pose information and robot configurational state to perform the necessary coordinate transformations to bring both the robot and its environment in the same frame of reference. A visual display allows monitoring of ASIMO's current estimate of its environment. The positional information is used for finding the physical locations of objects that are required to complete deictic gestures such as pointing at cards.

Incoming coordinate information can be either 2-D (such as table-top or stage coordinates) or 3-D (pose of the table). For example, the table's pose is reported in camera coordinates which is in turn transformed into the world reference system because we know the position of the camera on the robot. Cards are reported as 2-D coordinates relative to the table-top and are transformed into world coordinates since we know the table's position and orientation in the world. The left side of Figure 2 illustrates the robot and the detected table and cards in the environment map.

B. *Table Detection*

Since the memory game is played on a table, robustly detecting and tracking the game table is very important for uninterrupted play. We use a single camera to detect and track a table. Due to large radial distortion of the wide field-of-view lens in our camera, it is quite a challenging task to extract accurate information about the table, cards and the player's actions.

As shown in the first row of Figure 3, we use color and edge information to build a response map (Figure 3 b) from a raw image frame. Once the biggest blob is detected, its border pixels are mapped to normalized image coordinates after compensating for the radial distortion of the lens [10] (Figure 3 c). The convex hull of the border points are then

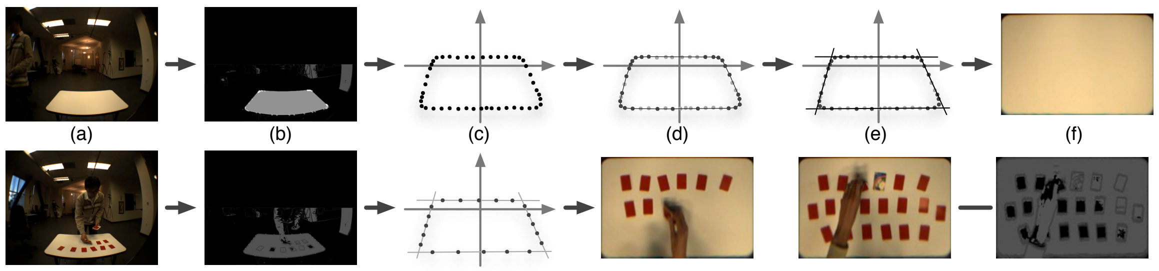

Fig. 3. Steps for table detection: a) table view from camera, b) table blob detection, c) table border points, d) table convex hull, e) table edges, f) table homography transform

computed (Figure 3 d) to generate table edge candidates. For the n convex hull points, we consider $n \times k$ lines that passes one point and its k neighbors, compute the support for each line by the table border points, and pick local-maximum lines. If four lines are detected correctly (Figure 3 f), then we compute the homography H [11] which maps $[-w/2 : w/2] \times [-l/2 : l/2]$ to the area surrounded by the four lines where w and l correspond to the width and length of the table. By applying this homography H, we can create a virtual top-view image of the table (Figure 3 f).

Once the table is detected, we can efficiently test whether the table is in the same detected position in subsequent frames. We spread a few test points on each edge of the table, and test whether the points are lying on the boundary between the table blob and the background. If sufficiently many points are on the table edge, we use the previous homography. Otherwise, we re-detect the table. This method gives us robustness over occlusions occurring when the player's hand touches or flips cards on the table.

If we know the geometry of the table, we can reconstruct its 3-D orientation and position from the estimated homography. When the head moves and if the motion trajectory is known, it can be combined with the previously estimated homography to generate an estimated current homography without detecting the table from the image. This option is useful when some parts of the table are out of the field of view.

C. Card Recognition

The card recognizer uses blob segmentation and direct template matching to segment and identify cards. Training images are stored as image patches. For each frame of incoming video, holes in the image blob corresponding to the table are detected as potential cards from the homography-corrected mask obtained from the table detector (Section III-B). If the area in pixels of a hole is too large compared to the training card patches, it is split into multiple pieces using K-means clustering. For each card blob, the center and 2-D orientation is estimated by taking the mean of the pixel coordinates and the eigenvectors of the covariance matrix respectively. Using this center and rotation, the image patch from the homography-corrected image is cropped and compared to the training patches.

When the person interacts with cards, there are significant occlusions by the player. To handle this occlusion, only the holes completely surrounded by the table blob are considered as valid card blobs. The other holes are considered as occluded regions. The cards in the occluded regions are not updated until the occluding blob disappears. If a card is removed, the region it was in before will be in the table blob, so we can detect the card has been removed during the occlusion. If a card was not touched, it will remain in the same position after the occlusion.

D. Table Touch Detection

ASIMO has a stereo camera pair that we use for detecting the presence of the player's hands over the table. Generally, stereo works poorly in the table region or other textureless regions. However, we have very good estimates of 3-D position and orientation of the table from the homography from Section III-B, so we can default to the table plane when encountering ambiguous stereo matching pairs. We set a volume-of-interest (VOI) on the detected table (marked as black and white dots in Figure 4 a), and only consider 3-D blobs in the volume. The resulting depth map we get from the stereo contains only the body parts or other objects on the table. With this information we can detect whether the player is touching a card or his hand is hovering over the table by measuring the distance from the table plane to the 3-D blobs in the VOI. Figure 4 (c) and (d) shows touch regions as red and proximity regions as green.

E. Projector

When ASIMO points to a region of the table containing many cards, it may be difficult to tell which card it is pointing at. We contemplated having ASIMO hold a laser pointer, but the limited degrees of freedom in its arm did not provide enough precision control. Consequently to reduce ambiguity, we installed a ceiling-mounted video projector. The projector's display is controlled by a module that interacts with the Cognitive Map. All modules are able to interact with the projector by sending drawing commands, which will be displayed on top of the table. During the game, the projector is used to highlight selected cards with an animated pattern, complementing ASIMO's pointing gesture and verbal indications (see Figure 2, right).

The projector system requires an initial calibration step to manually adjust the corners of a projected grid to match

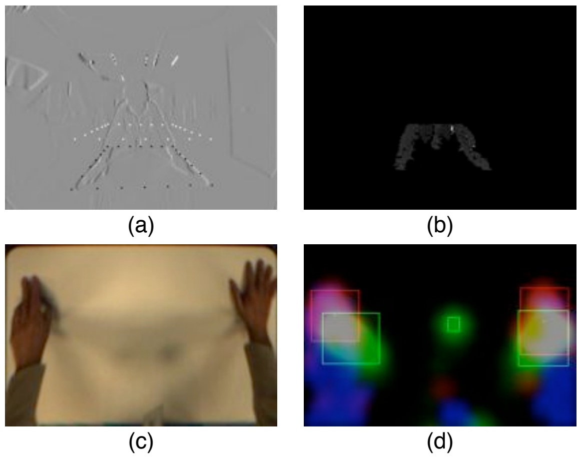

Fig. 4. Detecting hand-table touch and proximity events: (a) volume of interest (VOI) bounded by black and white dots, (b) cropped depth map in VOI, (c) hands touching table and (d) corresponding touch regions (red) and proximity regions (green).

the corners of the table using the computer's mouse. This ensures that the projection viewport is aligned and scaled to the projection surface. This method allows the projector system to be used on a variety of different rectangular tables.

F. Speech Recognition/Natural Language Processing

Recognizing speech utterances is important for realistic and satisfying human-robot interaction. The memory game consists of several scenarios where a player can be expected to speak. For instance, the player could seek help in determining the location of a particular card. In order to recognize the words and phrases used, game-specific grammars can be defined and employed. Robustness of recognition can be achieved by re-defining the grammar to contain only relevant phrases that are expected in different contexts of the game. In order to assign semantics to the spoken utterances, messages containing the utterances from the speech-recognition module are sent and processed by the Natural Language Processing(NLP) module. This module maps all the variations of an utterance to a unique set of semantic directives. These directives are passed to the game behavior modules for processing and high-level decision making. Further examples of how this process works are described in Section IV.

G. Text-to-Speech

The Text-to-Speech module uses a customized commercial engine to produce ASIMO's boyish English-speaking voice. The speech engine is capable of changing intonation when posing questions and offers some speed control as well as the ability to interrupt speech in progress (barge-in capability). This module services *say* messages sent from other modules like Multi-modal Communication (Section III-H) to convert text to an audible voice (see accompanying video).

H. Multi-modal Communication (MC)

Human beings tend to use different modalities such as speech and body gestures in their interactions with each other. To this end, we developed a multi-modal communication (MC) module capable of communicating using body gestures and speech simultaneously to improve ASIMO's expressive ability. This is especially important as ASIMO lacks an expressive face. Since the content and style of a message are specified separately, we can avoid repetitive, identical behaviors when ASIMO wishes to communicate information with the same semantic content.

In order to allow the MC module to be reuseable in different application scenarios, care was made to avoid building in application-specific knowledge within the module. The module must act on information solely from directives it receives from other modules, as well as any shared information accessible in the Cognitive Map. The MC module takes as input high-level communication directives from other modules such as *Indicate(card10)* or *Offer(help)* and converts the directives into physical gestures and speech. Directives can be accompanied by style tags so an application module can give hints to the MC module on how to deliver the contents of the communication directive. For example, ASIMO may choose to use polite phrasing, or short curt sentences to convey friendliness or impatience. For cases where an application wants complete control over how communication is delivered, there is a special *Verbatim* directive which allows a module to specify the exact words and gestures to use.

I. Task Matrix

The Task Matrix module, described in detail in [12], collects different motor programs that can execute motor tasks ranging from pointing to an object, playing pre-generated motion trajectories or navigation with motion planning around obstacles. It allows other modules to launch motor tasks using a high-level command interface. For example, the MC module can send the command *Point(target=Card12)* to the Task Matrix. The corresponding pointing motor program automatically resolves the 3-D position of Card12 and selects which arm to use based on proximity of the hand to the card. Other tasks include playing pre-generated motions like a victory celebration sequence, parametrized by duration, *Victory(duration=5 secs)*. In the accompanying video, these described behaviors can be observed.

J. Timer

The timer module produces a message echoing service that allows modules to schedule coarse-grained (minimum resolution 1 second) timeout events that can be used to trigger proactive behaviors. A module can specify when they want the message to be sent back to them (timeout interval) as well as the message type and contents. The timer module manages all timeout requests and at the appropriate time posts the messages to the Cognitive Map for the modules to receive. Since the timer module is implemented within the Cognitive Map framework, the timeout messages can be recognized by other modules different from the originator of the timeout request.

K. Game Behavior

The Game Behavior module is responsible for maintaining the game state during the entire scenario. It can be though of as a more specific instance of an *interaction module* that encapsulates the application-specific details of an interactive scenario. It also includes functions for handling game strategy and related interaction scenarios such as asking for help, or proactively warning players of potential bad moves (see Section IV).

Finite state machines are used to handle the recognition of the game situational context as well as decide which behaviors to perform based on perceived events from other modules in the system. The finite state machine is a collection of transition sequences defined by a current state, an event message that triggers a transition to a new state, additional optional Boolean conditions or *guards* that must be true for the transition to occur, and a set of functions that are executed upon transition. This arrangement allows for a wide variety of interesting behaviors and flexibility in describing when they should be chosen. A good review of finite state machines in the context of creating games can be found in [13].

In addition to the transition sequences, local game state information is also maintained. In particular, a *tableau* is maintained of the current state of cards on a table, *card memory* keeps track of ASIMO's current knowledge of the identity of cards (those whose identification is revealed in the past), and turn-based *ply* information that monitors the current cards revealed for each player during a turn in the memory game. Various events such as card flips initiate updates to ASIMO's internal card memory.

During the construction of the memory game, we were able to build up the complexity of ASIMO's interactive behavior by simply adding new sequences to the finite state machine. The sequences are defined in the Lua scripting language [14] which can be run-time interpreted to allow for quick evaluation of newly added interactions. Since all the application-specific state transitions and game-related functions of the finite state machine are implemented in the Lua scripting language, new interactive scenarios can be modeled simply by replacing this single script definition file without needing to re-compile the generic portion of the game behavior module that implements the execution of the finite state machine.

IV. Interaction Case Studies

ASIMO's style of play during the memory game focuses on offering a friendly, interactive experience for the human opponent. With this in mind, we illustrate how different modules interact with each other during the course of the memory game. Three different scenarios that can occur during the game will be described: ASIMO pointing at cards, ASIMO offering help, and ASIMO proactively warning a player if they are about to make a mistake.

A. Pointing at cards

Since ASIMO does not have hands dexterous enough to pick up flat cards on a table, it must utilize its speech and gesture modalities to indicate to its human opponent which card to select on its behalf. In particular, once ASIMO decides which card to select, it verbally informs the user it wants to select the card while pointing at the card and highlighting it with the projector module.

The Game Behavior module recognizes through its state machine that it is ASIMO's turn. ASIMO then refers to its local game state memory to identify any potential unrevealed card matches on the table. In the event there are no matches, it selects an unknown card. If the card reveals a known match, ASIMO will promptly select the matching card. Otherwise, it deliberately chooses a known card so as to minimize the risk of revealing new information to its opponent. The sophistication of game strategy can be adjusted for different levels of players.

When ASIMO decides to select a card, it sends a high-level directive, *Indicate(cardID)* along with the CMobject for *cardID* containing the object's properties to the MC module. The MC module recognizes the *Indicate* directive and determines that a pointing gesture and projector highlighting are appropriate, as well as a speech utterance like: "I choose this card.". In this example, since the MC module can access the object's type through the sent CMobject, it can use this information to indicate that it is a card. Alternatively, ASIMO can choose to say something like, "I choose this one" or "This." depending on style flags as well as other state information it can access from the Cognitive Map. For pointing, a *Point(cardID)* is sent to the Task Matrix which resolves the 3-D position of cardID from the Environment Map and runs the motor program to generate a pointing trajectory at the 3-D location. The MC module also sends a *Project(cardID)* to the Projector module which computes the 2-D location of the card on the table to highlight the card. Finally, a *Say("I choose this card")* message is sent to the Text-To-Speech module to generate speech on ASIMO's audio speakers. See Figure 5.

B. Offering help

Several game-related conversations can occur in the course of the game. For example, if ASIMO notices the player is taking a long time to perform his or her turn, it may proactively prompt the player if he or she needs help. If the player answers "No", ASIMO may apologetically say, "Sorry, just asking." . However, if the player says "Yes", ASIMO examines the state of the player's turn to determine how to proceed. If one card has already been exposed, ASIMO automatically can check its card memory to determine the location of the other card. If ASIMO does not know, ASIMO politely declines saying, "I have no idea myself." If it does know, it proceeds to indicate two cards with the MC module as described above, with one being the correct card. The rationale behind this behavior is that we did not want ASIMO to give the answer outright, but to reduce the probability the opponent will select the wrong card.

To accomplish this, when the game module enters states corresponding to the player's turn, it initiates a message to the Timer module to timeout after a predetermined number of seconds. At the end of the timeout period, the Timer module sends a message back to the Game Behavior module, prompting the state machine to enter a dialog offering help. The Offer(help) directive is sent to the Multi-modal communication module causing ASIMO to produce an open arm gesture while saying a phrase like: "Do you need help?". Meanwhile, the game behavior module is now awaiting a response from the user. The Speech Recognition module listens for utterances which upon receiving, it sends to the Natural Language Processing module which can convert the response to a Reject directive (if negative) or an Acknowledge directive (if positive). In the latter case, the NLP module can append style tags like "simple" or "lengthy" to indicate if the player responded in a curt or lengthy manner to provide hints to another module on how to appropriately respond. These directives are sent back to the Game Behavior module. Finally, if the answer is positive and ASIMO is capable of offering assistance, *Indicate* or *Verbatim* directives are sent back to the Multi-modal module to point out card choices for the user (Figure 5).

C. Proactive warning of touched cards

Another proactive behavior we built into the memory game is having ASIMO warn or tease a player if they are about to flip an incorrect card. If it is the player's turn and the second card is about to be picked, ASIMO can detect the initial card touch event prior to flipping and determine if the card is the correct one to pick. If a pending mismatch is about to occur, ASIMO playfully says, "Are you sure about that card?", prompting the human player to be filled with self-doubt (Figure 5).

The touch event is created from composite information from the table detector and card recognition modules. The table detector module detects a generic table touch event, communicating position information to the card recognizer which can resolve the position information to a specific card. The card recognizer subsequently posts a message to the game behavior module indicating a card touch has occurred along with the corresponding CMobject for the card. The game behavior module uses the submitted card information to verify if the pending card is the correct one and if not, sends a warning by issuing a *Verbatim* directive to the MC module. The game behavior state machine then returns back to the state awaiting the player's second move.

V. Discussion

A. Pattern recognition

The pattern recognition modules in our system range from the stand-alone table detector to more complex modules that combine information from several other perceptual modules. For example, identification of touched cards requires combining table detector information with card detectors. Patterns are not restricted to the spatial domain. Time-series information is an important characteristic of many activity detectors and is utilized in the memory game for identifying arrival, flipping and removal of cards.

B. Uncertainty

When dealing with real-world environments and sensors, handling of uncertainty becomes an issue. In our scenario, most of the uncertainty in perception is handled by individual pattern recognition modules. However, uncertainty at higher levels, related to interpretation of events is also handled by the Game Behavior module. For example, a player may simply flip a card over, or remove a card, examine it, and place it back on the table. The finite state machine can use its knowledge of the current game state to interpret the proper meaning of the sequence of events observed.

Nevertheless, our implementation of the memory game can still suffer from critical failure conditions. If a critical event is missed by any of the pattern recognition modules, the robot may be left waiting indefinitely for the missed event to occur. This problem can be handled by extending the finite state machine to include timeout events that trigger subsequent action to reassess the game state. Alternatively, using probabilistic methods to analyze time series events and sequence actions like dynamic Bayesian networks [15], [16] can be used to robustly estimate the current game state. We also experienced robustness issues with the speech recognizer while testing the memory game with children due to the higher pitch in their voices. This was due to the fact that our speech recognizer was designed with adult voice acoustic models. Creating a new voice model from children's voice data should help alleviate the problem.

C. Limitations

There can be unwanted interactions between modules that may jeopardize the performance of modules, especially when dealing with moving robots. For example, if ASIMO has excessive head motion, most pattern recognition modules may fail because they either incorrectly detect removal of objects as the field of view changes or the resulting motion blur degrades the images to be processed. We intend to use the Cognitive Map architecture to allow the Task Matrix to send warning messages to modules of impending head motion to allow perception modules to go into standby during head motion so as to not post faulty information to other modules.

As with any large distributed system, the failure of critical components can still halt the application. For example, any errors in the Game Behavior module will obviously not allow the game to proceed. Although one could restart the problematic module, if the error is systematic, there is no easy way to solve the problem beyond standard debugging.

Another problem with finite state machine decision-making is that the occurrence of events in a live system can outpace the handling of those events. In our testing, we often rapidly played out the game and ASIMO's reactions sometimes lagged with the current action of the game. For example, if a player already picked two cards, it does not make sense for ASIMO to prompt for the second card to be

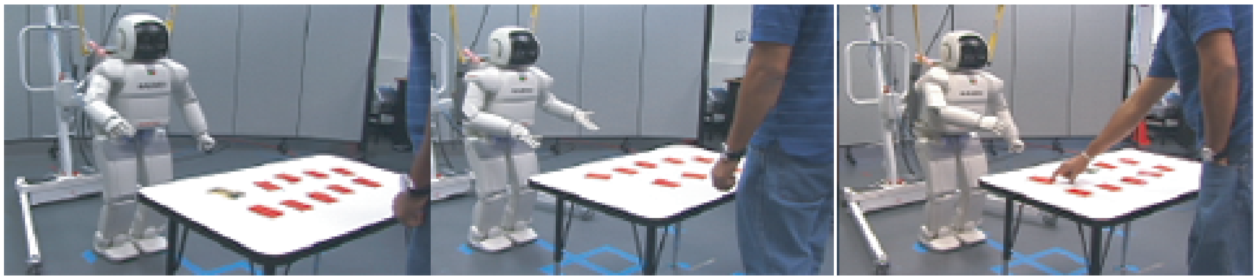

Fig. 5. Interactive scenarios: Pointing at cards (left), Offering help (middle), Proactively warning of touched cards (right)

picked after the incident has already occurred. The solution involves ASIMO examining the message queue prior to processing responses so that it can decide to skip notifications if it notices the player has already performed the event.

D. Reusability

The Cognitive Map architecture facilitates reuse of many modules created for the memory game for other interactive scenarios. Since modules operate as independent agents and communicate with each other only via messages, a new interactive application can be created by changing or replacing a few key modules while keeping the rest of the system intact. Most modules should be designed to offer generic services while isolating application-specific details to a single module. Generic communication behavior embodied in the Task Matrix, Text-to-Speech, and Multi-Modal communication modules can be reused without modification as they are designed with no application-specific knowledge. Since the card recognizer detects generic activities such as flipping and removing, it can be reused for different card-related games. The Game Behavior module (Section III-K) is an instance of an interaction module that encapsulates all the application-specific details of the memory game. Different card games can be developed by changing the rules and game states in this module. Messages sent from this module can contain application details used to parametrize the behavior of a generic module. For example, the Game Behavior module can send a message to the speech recognizer to configure grammar files that describe game-specific vocabulary. This incremental strategy for building new applications by reusing modules allows faster development times by focusing efforts on only new functionality.

VI. Conclusion

The architectural framework we have developed based on the Cognitive Map and related modules enables us to model the memory game as a continuous, complex scenario involving two-way communication between ASIMO and its human opponent. Our design decisions to make most modules application-independent allow us to reuse modules to rapidly create novel interactive applications in the future. The memory game has been tested informally with approximately 20 different players, but in order to quantify what aspects of multi-modal communication are effective, detailed user interaction studies should be conducted. Many interactive sequences in the Game Behavior module can be modeled with a combination of events, state machines and behavior actions. We would like to use the same representation as a learning template to attempt to dynamically build new state machines from observed events and responses. By going through the process of attempting to reach a clear, focused goal of ASIMO playing the memory game, we can now proceed to the next step of learning sequences of interaction in novel situations of similar complexity.

References

[1] Honda Motor Co., Ltd., "Asimo year 2000 model," http://world.honda.com/ASIMO/technoloogy/spec.html, 2000.

[2] M. Stilman, K. Nishiwaki, and S. Kagami, "Learning object models for whole body manipulation," in *IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids 2007)*, 2007.

[3] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi, "Leg motion primitives for a dancing humanoid robot," in *IEEE 2004 International Conference on Robotics and Automation*, 2004.

[4] V. Ng-Thow-Hing, E. Drumwright, K. Hauser, Q. Wu, and J. Wormer, "Expanding task functionality in established humanoid robots," in *IEEE-RAS 7th International Conference on Humanoid Robots*, 2007.

[5] L. Zhang, M. Gillies, and J. Barnden, "Emma: an automated intelligent actor in e-drama," in *ACM International Conference on Intelligent User Interfaces 2008 (IUI 2008)*, 2008, pp. 409–412.

[6] C. Breazeal and B. Scassellati, "How to build robots that make friends and influence people," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, 1999.

[7] R. Bischoff and V. Graefe, "Demonstrating the humanoid robot hermes at an exhibition: A long-term dependability test," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02); Workshop on Robots at Exhibitions.*, 2002.

[8] R. Gockley, A. Bruce, J. Folizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, R. Simmons, K. Snipes, A. Schultz, and J. Wang, "Designing robots for long-term social interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, 2005, pp. 2199–2204.

[9] B. Hayes-Roth, "A blackboard architecture for control." *Artificial Intelligence*, vol. 26, pp. 251–321, 1985.

[10] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[12] E. Drumwright and V. Ng-Thow-Hing, "The task matrix: An extensible framework for creating versatile humanoid robots," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2006, Orlando, FL, USA.

[13] J. Smed and H. Hakonen, *Algorithms and Networking for Computer Games*. John Wiley & Sons Inc., 2006.

[14] R. Ierusalimschy, *Programming in Lua*. Lua.org, 2006.

[15] K. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, UC Berkeley, 2002.

[16] B. Laxton, J. Lim, and D. Kriegman, "Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video," in *Computer Vision and Pattern Recognition (CVPR 2007)*, 2007.