The Memory Game: Creating reactive, interruptible, turn-taking, human-robot interaction for ASIMO

Victor Ng-Thow-Hing , Jongwoo Lim, Joel Wormer,
Ravi Kiran Sarvadevabhatla, Carlos Rocha,
Kikuo Fujimura and Yoshiaki Sakagami
Honda Research Institute USA, Inc.
United States of America

1. Introduction

Recently, humanoid robots have begun to demonstrate an array of impressive motor skills, such as climbing stairs (Honda Motor Co., Ltd., 2000), whole body manipulations of large objects (Stilman et al., 2007), and dancing (Nakaoka et al., 2004). On the perceptual side, continued effort has been invested in object recognition and geometry reconstruction to provide robots a model of their 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 (Ng-Thow-Hing et al., 2009) to meet these needs.

In this paper, we present our attempt to utilize our system architecture and component modules to create a well-defined 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 (Honda Motor Co., Ltd., 2000) and a human opponent.

1.1. The 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 its 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 4.5).

1.2. Reusability of architecture and modules

Having the 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 was important to make sure the architecture and modules did not become too specialized to the memory game as we wished 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.

1.3. Interaction characteristics

Our main research goal is to enable rich human-robot interactive experiences. The memory game is our first attempt at creating multi-modal, turn-taking based communication with an

application that requires state information to be stored and used to direct the content and style of communication. Another goal is to create robust experiences that allow people to interact with the robot in the least-constrained manner. For example in the memory game, 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 needed to focus on activities of interest, while ignoring unimportant events. Players were 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 reacted as a result of speech or activities from the user. ASIMO was able to proactively perform behaviors while waiting for the player to take his or her turn. This included asking players for help if they were taking too long to pick a card. It also included expressing the same semantic message different ways to avoid monotony. For example, if ASIMO was winning the game, it could choose to say things in a more gentle manner so as not to upset the player. To solve these problems, we provided timeout mechanisms for ASIMO to proactively initiate behaviors, idle motions such as nodding acknowledgment whenever the player spoke to ASIMO, and a multi-modal communication module to generate new phrases and gestures based on requests for semantic message communication from other modules.

1.4. Previous Work

Robots interacting with humans have come in a variety of forms, ranging from baby seals like PARO (Wada and Shibata, 2007), small personal robots like PaPeRo (Osada et al., 2006), and human-like androids (Ishiguro, 2006). The form of the robot is a key factor in the design of its intended use and in people's expectations of its behaviour and capabilities. Humanoid robots by virtue of their body design being similar to our own, come with the increased expectations of human-like behaviour. In cases when the robot's behaviour did not coincide with the expectations of its visual appearance, a phenomenon of uneasiness called the *uncanny valley* (Ishiguro, 2006) has been observed in people's impressions of the robot. Consequently, humanoid robots such as the WE-4RII (Itoh et al., 2004) exhibit unique functionalities and behavior like facial expressions and arm gestures that mainly serve a communicative role rather than a utilitarian one. Combined with good perception, human-robot interaction can be seen as a novel way for people to accomplish a task with a machine using the natural communication modalities of speech, gesture and other recognized visual cues. In (Sakagami et al., 2002), a system architecture for ASIMO is introduced to connect a behavior planning system to the vision and auditory systems of the robot, enabling key capabilities important for interaction such as gesture recognition, sound localization and obstacle avoidance during navigation.

Communication between humans and embodied agents has been explored in virtual reality and in human-robot interaction. Virtual reality systems featuring synthetic actors have the advantage of more expressive computer graphics and perfect knowledge of the environment (Zhang et al., 2008). In contrast, human-robot interaction requires the additional problem of developing robust sensing and control algorithms. The Kismet system (Breazeal and Scassellati, 1999) specialized in facial expressions and featured a perceptual attention model to direct head gaze for short-term interactions.

There have been several robotic systems designed for long-term interaction with humans. The HERMES system (Bischoff and Graefe, 2002) 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 (Gockley et al., 2005) 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.

In the remainder of this paper, we will give an overview of our robot architecture in Section 2, followed by our design process from conceptualization of the interactive scenario to system realization in Section 3. Individual component modules are described in Section 4. Several case studies highlighting how information flows through our architecture are covered in Section 5. Discussions of several important lessons and limitations are in Section 6 with the final conclusion in Section 7.

2. System Overview

Details of our system architecture are described in (Ng-Thow-Hing et al., 2009). 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 based on a blackboard architecture (Hayes-Roth, 1985). Component modules can be distributed on different machines and operating systems or co-located on the same 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. Fig. 1 depicts the main 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 CObject) that describes its properties. CObjects 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.

Fig. 1. Overview of the Cognitive Map robot architecture: Representative components depicted showing connectivity to the central Cognitive Map server. Component module types are labeled above name. Arrow directions depict dependencies on various interfaces (not message directions).

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, 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-based or expression modules can produce perceivable robot activity such as speech utterances, gestures, or other complex motor tasks. Fig. 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 5, we describe the communication flow for several situations that can occur during the game. Section 4 describes each module used in the system in more detail.

3. Design

Our design process begins with the construction of a branching storyboard (Fig. 2) that depicts the desired observed behavior of both the robot and humans involved in the scenario. Every attempt is made to predict different conditional behaviors and responses to those situations. For example, the storyboard shows how the robot would behave if it won, what happens if it failed to see a detected card or how to respond if the human player asks for help. This process allows us to identify what technical component modules are required and what information they should publish or subscribe to for accomplishing their role within the system. Relating to the Cognitive Map, the storyboard aids both in the design of individual modules and their message interfaces between each other. For example, if dialog is present in the storyboard, this implies the need for both speech recognition and speech synthesis components. The storyboard in Fig. 2 further suggests that a simple input text

4. 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.

Fig. 3. Left: Environment map, Right: Projector highlighting of card (enhanced for easier visibility in image)

4.1. Environment Map

The Environment Map collects object pose information and robot configuration 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 Fig. 3 illustrates the robot and the detected table and cards in the environment map.

4.2. 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 top row of Fig. 4, we use color and edge information to build a response map (Fig. 4b) 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 (Forsyth and Ponce, 2002) (Fig. 4c). The convex hull of the border

Fig. 4. 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.

Bottom row: Same steps showing views when person and cards are present.

points are then computed (Fig. 4d) to generate table edge candidates. For the n convex hull points, we consider the $n-k$ lines that pass through 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 (Fig. 4f), then we compute the homography H (Hartley and Zisserman, 2004) which maps $[-w/2:w/2] \times [-l/2:l/2]$ to the area bordered 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 (Fig. 4f).

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.

4.3. Card Recognition

The card recognition module was one of the most important modules for the scenario because of its responsibility for detecting the key elements of the game. Consequently, in the course of its development, we went through several iterations. The initial version was adapted from general object recognition software. The recognition was reasonably robust, especially when applied to the homography table-top view that eliminated perspective effects. However, the detection of events such as card flipping and removals was noisy and required the use of lengthy timeouts to ensure stable detection of events which decreased the responsiveness of the game.

The second version of the card recognizer used blob segmentation and direct template matching to segment and identify cards. Training images were stored as image patches. For each frame of incoming video, holes in the image blob corresponding to the table were detected as potential cards from the homography-corrected mask obtained from the table detector (Section 4.2). If the area in pixels of a hole was too large compared to the training card patches, it was split into multiple pieces using K-means clustering. For each card blob,

the center and 2-D orientation were 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 was cropped and compared to the training patches.

When the person interacted with cards, there were significant occlusions by the player. To handle this occlusion, only the holes completely surrounded by the table blob were considered as valid card blobs. The other holes were considered as occluded regions. The cards in the occluded regions were not updated until the occluding blob disappeared. If a card was removed, the region it was in before would be in the table blob, so we could detect the card had been removed during the occlusion. If a card was not touched, it would remain in the same position after the occlusion. This detector worked when there were approximately 10-20 cards with mild overlapping between cards. When there were more cards, the holes in the mask tended to become connected to each other, and this made segmentation of connected blobs harder. Another problem in the mask-based method is that the holes near the farther edge of the table tended to be larger than the nearer ones. This made the location estimate on the far cards inaccurate and sometimes resulted in incorrect detection of cards.

Fig. 5. In our latest card detector, the edge response maps on the left are used to detect and recognize cards (front and back) on the right. The outlined card is an example of a 'gray blob' that needs subsequent processing to determine if it is a card.

To resolve these issues, our latest card detector (Fig. 5) uses edge information extracted only from the homography-corrected image. After non-maximal suppression of sobel edge detection, we build the edge response map, whose pixel values from 0 to 1 represent the

likelihood of edge existence on the pixel. Initially edge blobs higher than a certain threshold are detected as the candidates for cards. If an edge blob fits in size and shape, it is classified as a card. After this initial classification, the remaining blobs are merged and tested whether they are from one card or not. To find the proximity between edge blobs, the concept of a "gray blob" is used. Instead of the binary representation in usual blob detection methods, we find the weakest link between two blobs, and that value represents the proximity between blobs. From the strongest pairs of blobs we test if the assembled blob looks like a card, and repeat until there is no likely combination left. After detecting cards, the identification of each card is determined by comparing pixel values of the region with the trained templates.

Fig. 6. 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 and proximity regions (lighter regions closer to table).

4.4. 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 4.2, 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 Fig. 6a), 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. Fig. 6d shows the touch regions corresponding to Figure 6c.

4.5. 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 to highlight cards. 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 Fig. 3, right).

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

4.6. 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 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 5.

4.7. Text-to-Speech

The first version of our 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 4.8) to convert text to an audible voice.

We recently updated our speech engine to a newer version that allows more control over intonation of any syllable in a sentence and has more emotionally expressive phrases such as "Hey!" and nonverbal expressions such as laughter. We have found that players enjoyed the more expressive speech engine, even if the functional robustness of the application

suffered. We suspect that greater expression provided a stronger illusion of ASIMO having a real personality. We intend to confirm and exploit this observation in future work.

4.8. 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 reusable 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 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.

4.9. Task Matrix

The Task Matrix module, described in detail in (Drumwright and Ng-Thow-Hing, 2006) 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. Tasks are run in their own control loops and can react to changes in the environment. The main purpose of the Task Matrix is to allow 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)* and shown in the bottom right image of Fig. 9.

4.10. 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 received by other modules different from the originator of the timeout request.

4.11. Game Behavior

The Game Behavior module is responsible for maintaining the game state during the entire scenario. It can be thought 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 5).

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 (Smed & Hakonen, 2006).

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 (Ierusalimsky, 2006) 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.

The initial version of the memory game was implemented with a single finite state machine (Fig. 7a). However, we found that this approach was not scaleable when we attempted to add more scenarios to the game since the number of new transitions that had to be added to

Fig. 7. Interaction component's finite state machines (simplified for clarity). Transitions denote condition/action pairs. (a) Original finite state machine. (b) Refactored finite state machine.

each newly added state also grew increasingly larger. Furthermore, any failure to perceive a condition event would result in failure of a transition to occur, often leaving the game behavior module indefinitely waiting for an event that had already occurred. We were able to refactor the game behavior module to keep multiple state machines for different entities in the game (Fig. 7b) featuring one state machine for the state of cards on the table and one for the rules of the game. This approach simplified the structure of the game, and increased robustness by allowing the card table state machine to focus on valid card layouts while the game rules state machine separately monitored whether it was currently the robot or the player's turn.

4.12. Multiple Interactive Tasks

In addition to the memory game application, we expanded the robot's ability to handle other active interaction modules simultaneously. For example, we would like the robot to answer off-topic questions from people (e.g., frequently-asked questions) while playing the memory game. There is a potential conflict if both the memory game and frequently-asked questions (FAQ) scenarios attempt to say something at the same time. Initially, we created a module called the *focus arbiter* that sorted all incoming events to one of the active interaction modules to be serviced. However, this design did not scale well as the focus arbiter would need detailed application-specific knowledge of which events each module could handle. As incoming perceived events could only be redirected to a single module by the focus arbiter, this design would not allow an event to be handled by two or more interaction modules simultaneously, creating a somewhat arbitrary restriction.

Fig. 8. The mediator module is used to arbitrate robot requests from multiple interactive applications. All messages are routed through the Cognitive Map server (not shown for simplicity).

When multiple interactive scenarios are present, the constraint should be placed on the robot's own resources for speaking and performing certain motor tasks (i.e., the robot cannot say two things at once or walk to two different locations at once), and not on the incoming perceived events. Consequently, in our revised design all interactive modules are

free to receive all message events that they subscribe to, and can generate speech or task requests (see Fig. 8) that they wish the robot to service. These requests are then sent to the *mediator module* to be placed in a priority queue, with priority pre-assigned based on the relative importance of each interactive application. In Fig. 8, the memory game is assigned higher priority than the FAQ module. Requests of higher priority can pre-empt lower priority requests. The mediator also sends acknowledgement messages back to the interaction modules when their speech or task commands have been serviced or pre-empted, so that the individual modules can decide what appropriate action to take. For example, if the frequently asked questions knows it was interrupted while answering a specific question, it can choose to initiate a new speech request repeating the answer, but can also precede it with a transition phrase like "Sorry, as I was saying...".

5. 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.

5.1. 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 CObject 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 CObject, 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 (Fig. 9). Finally, a *Say("I choose this*

card") message is sent to the Text-To-Speech module to generate speech on ASIMO's audio speakers.

5.2. 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 do 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 (Fig. 9).

5.3. 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 picking up that card?", prompting the human player to be filled with self-doubt (Fig. 9).

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 CMOBJ for the card. The game behaviour 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.

6. Discussion

Several important issues arose during development of the memory game scenario. We discuss each of these in the following subsections.

6.1. 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.

6.2. 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 (Murphy, 2002) 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.

6.3. 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

Pro-active behavior: Warning about touched cards.

Activity detection

Speech recognition and simple conversation: Offering help to player

Pointing actions: uses spatial information in environment map

Multi-modal communication with speech and gestures

External device control: projector highlighting

Situational awareness

Custom motions for expressive behavior

Fig. 9. Various features of the memory game application.

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 4.11) 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.

6.4. 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 use the Cognitive Map architecture to allow the Task Matrix to send warning messages to perception-based modules of impending head motion to allow these modules to go into standby if needed during head motion so as not to 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 at run-time. These inherent problems in module functionality need to be resolved during development with thorough system-wide testing to identify failure cases early and then to test solutions to fix them.

The selection of finite state machines for decision-making features actions that are only triggered on event occurrences, creating moments of robot activity alternating with periods of inactivity. Other ways of organizing the mapping from perception to action are needed to creating complex, overlapping behaviour whose elements run over different time scales. 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 picked after the incident has already occurred. The solution involves ASIMO examining the incoming event queue in an efficient manner prior to processing responses so that it can decide to skip notifications if it notices the player has already performed the event.

7. 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. Modules can be created for many different roles, including pattern recognition, decision-making and robot behaviour. Modules can control the behavior of other modules as seen with the mediator module (Section 4.12) prioritizing which interactive modules will have their robot commands serviced. Our design decision to make most modules application-independent allows us to reuse modules to rapidly create novel interactive applications in the future.

7.1. Future Work

The memory game has been tested informally with approximately 20 different players, but in order to systematically quantify which aspects of multi-modal communication are effective, detailed user interaction studies are planned. We are exploring new ways to infer state and organize behavior to handle uncertainty and to improve responsiveness in the robot. Many interactive sequences in the Game Behavior module can be modeled with a combination of events, state machines and behavior actions. We can potentially use the same representation as a learning template for dynamically building new state machines from observed events and responses. With the completion of this first version of the memory game, we intend to use this scenario as a research platform for ASIMO to learn new complex behaviour and to improve the richness of communication between robots and humans.

8. References

- Bischoff, R. & Graefe, V. (2002). Demonstrating the humanoid robot HERMES at an exhibition: A long-term dependability test, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, Workshop on Robots at Exhibitions., 2002.
- Breazeal, C. & Scassellati, B. (1999). How to build robots that make friends and influence people, *Proceedings in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, October 1999, Kyongju, Korea.
- Drumwright, E. & Ng-Thow-Hing, V. (2006). The task matrix: An extensible framework for creating versatile humanoid robots, *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '06)*, Orlando, USA.
- Forsyth, D. & Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall
- Gockley, R. ; Bruce, A. ; Folizzi, J. ; Michalowski, M. ; Mundell, A. ; Rosenthal, S. ; Sellner, B. ; Simmons, R. ; Snipes, K. ; Schultz, A. & Wang, J. (2005). Designing robots for long-term social interaction, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 2199-2204, Edmonton, Canada.
- Hartley, R. & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press.
- Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, vol. 26, pp. 251-321.
- Honda Motor Co., Ltd. (2000), ASIMO year 2000 model humanoid robot.

- Ierusalimschy, R. (2006). Programming in Lua, 2nd Edition, Lua.org., March 2006.
- Ishiguro, H. (2006). Interactive Humanoids and Androids as Ideal Interfaces for Humans. *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI'06)*, pp. 2-9, Sydney, Australia.
- Itoh, K. ; Miwa, H. ; Matsumoto, M. ; Zecca, M ; Takanobu, H. ; Roccella, S. ; Carrozza, M.C., Dario, P. & Takanishi, A. (2004), Various emotional expressions with emotion expression humanoid robot WE-4RII, in *Proceedings of the 1st IEEE Technical Exhibition Based Conference on Robotics and Automation (TExCRA '04)*, November 2004, pp.35-36.
- Murphy, K. (2002). Dynamic bayesian networks: Representation, inference and learning, Ph.D. dissertation, UC Berkeley.
- Nakaoka, S.; Nakazawa, A.; Yokoi, K. & Ikeuchi, K. (2004). Leg Motion Primitives for a Dancing Humanoid Robot, *Proceedings of IEEE 2004 International Conference on Robotics and Automation*, pp. 610-615, April 2004, New Orleans, USA.
- Ng-Thow-Hing, V.; Thórisson, K. ; Sarvadevabhatla, R. K. ; Wormer, J. & List, T. (2009) Cognitive Map Architecture: Facilitation of Human-Robot Interaction in Humanoid Robots. *IEEE Robotics and Automation Magazine*, vol. 16, No. 1, Mar. 2009, pp. 55-66.
- Osada, J. ; S. Ohnaka & M. Sato. (2006) Scenario and Design Process of Childcare Robot PaPeRo. *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, vol. 266, no. 92, Hollywood, USA.
- Sakagami, Y. ; Watanabe, R. ; Aoyama, C. ; Matsunaga, S. ; Higaki, N. ; & Fujimura, K. (2002) Intelligent ASIMO: System overview and integration, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, pp. 2478-2483, Genova, Switzerland.
- Smed, J. & Hakonen, H. (2006). Algorithms and Networking for Computer Games. John Wiley & Sons Inc.
- Stilman, M.; Nishiwaki, K. & Kagami, S. (2007). Learning object models for whole body manipulation, *Proceedings of IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids 2007)*, November 2007.
- Wada, K. & Shibata, T. (2007). Robot Therapy in a Care House – Change of Relationship among the Residents and Seal Robot during a 2-month Long Study, *Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN '07)*, August 2007, pp. 107-112.
- Zhang, L. ; Gillies, M. & Barnden, J. (2008). Emma: an automated intelligent actor in e-drama, *Proceedings of ACM International Conference on Intelligent User Interfaces 2008 (IUI 2008)*, pp. 409-412.

Robotics 2010 Current and Future Challenges

Edited by Housseem Abdellatif

ISBN 978-953-7619-78-7

Hard cover, 494 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

Without a doubt, robotics has made an incredible progress over the last decades. The vision of developing, designing and creating technical systems that help humans to achieve hard and complex tasks, has intelligently led to an incredible variety of solutions. There are barely technical fields that could exhibit more interdisciplinary interconnections like robotics. This fact is generated by highly complex challenges imposed by robotic systems, especially the requirement on intelligent and autonomous operation. This book tries to give an insight into the evolutionary process that takes place in robotics. It provides articles covering a wide range of this exciting area. The progress of technical challenges and concepts may illuminate the relationship between developments that seem to be completely different at first sight. The robotics remains an exciting scientific and engineering field. The community looks optimistically ahead and also looks forward for the future challenges and new development.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Victor Ng-Thow-Hing , Jongwoo Lim, Joel Wormer, Ravi Kiran Sarvadevabhatla, Carlos Rocha, Kikuo Fujimura and Yoshiaki Sakagami (2010). The Memory Game: Creating Reactive, Interruptible, Turn-Taking, Human-Eobot Interaction for ASIMO, Robotics 2010 Current and Future Challenges, Housseem Abdellatif (Ed.), ISBN: 978-953-7619-78-7, InTech, Available from: <http://www.intechopen.com/books/robotics-2010-current-and-future-challenges/the-memory-game-creating-reactive-interruptible-turn-taking-human-robot-interaction-for-asimo>

INTeCH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.