A Layered Architecture for Lifelike Robotic Motion

Scott Snibbe, Mark Scheeff and Krisnawan Rahardja
Interval Research Corporation
1801 Page Mill Road, Building C
Palo Alto, CA 94304
{snibbe, scheeff, rahardja}@interval.com

ABSTRACT

In this paper we present an architecture for the synthesis of realistic and emotionally expressive robotic motion. Our work borrows heavily from the computer graphics and classical animation communities which emphasize perceptual realism of motion, rather than mechanical efficiency. Principles of traditional animation, as articulated over the last century, are algorithmically embodied in our work, serving to create a sense of character and presence in a robot.

Our architecture represents motion as layers of periodic function primitives. These primitives form more complex behaviors which are combined and sequenced by the *Motion Compositor*, a module that is parameterized by control weights determined by higher level abstractions. A signal-processing module modifies the overall synthesized motion, providing a complementary method for global control of affect[1].

Our layered motion architecture is currently being used to explore the emerging field of personal robotics. In particular, we are experimenting with emotional communication between a physically embodied robot and humans. We have built a small pet-like robot which we operate via high level remote control to perform experiments in emotional interaction.

Keywords

Personal Robotics, Motor Control, Motion Control, Emotion, Affect, Animation, Digital Signal Processing

MOTIVATION

Our project at Interval is concerned with using real, physical devices to portray affect. In our current work, we have created a mobile robot that uses gesture, action and sound to communicate with people. We are primarily interested in understanding human reaction to this type of creature.

We view this work as addressing fundamental issues in intuitive, friendly human-computer interface technologies (HCI). In many ways, this robot is the direct opposite of the traditional workstation paradigm of HCI: it can move through space, it has a rich set of senses and physical actions, and it communicates with affect instead of syntax and is thus inherently social.

Currently, the robot is controlled at a high level by a remote human operator who gives commands for gross orientation, movement and emotional state. Because we are focused on expression and the consequent responses of people, we have emphasized fine motor control in our design. With such a framework we can demonstrate a rich variety of emotional states. Besides studying the dominant affective phenomena such as “happy” and “cute” present in today’s toys, we can make our robot appear angry, depressed, sleepy, nervous, surprised, etc. Additionally, since we occupy the same physical space as a person, we can test behaviors impossible with on-screen characters – approach, avoidance, touching, hiding, and even threatening are possible. This richness facilitates a broad range of responses from our human subjects.

In programming this robot's expressions and movements, we found that no current methodologies in robotics could give us the lifelike motion we were seeking. This paper covers our work in developing and implementing algorithms that resulted in the emotional character-based movement we required.

PRIOR WORK

The majority of research in robotic motion control prior to 1985 focused on efficient path and motion planning primarily for industrial and scientific applications. In the area of path planning, research focused on avoiding collisions with obstacles when navigating through space with a mobile robot or when moving a robotic appendage towards a given target [2]. Dynamics and kinematics provided the two main techniques for computing robotic movement. In *forward* kinematics and dynamics, the goal is predicting the actions of a robot given impulse forces and torques and a model of the environment. In *inverse* kinematics and dynamics the forces necessary to result in a robot following a given trajectory or reaching a given target are computed a priori [3].

[1] We use the term *affect* as something that relates to, arises from or deliberately influences emotions. Rosalind Picard has articulately explored this domain as it relates to Human-Computer interaction in her book Affective Computing [1].

In the Computer Graphics and Animation community, both kinematics and dynamics have been used to generate realistic solid-body animation, referred to as *Physically Based Modeling* [4]. Additionally, there have been several interesting systems developed for the integrated control of real-time characters. The mechanics of human motion have been explored to successfully create directable characters following the spirit of kinematics and dynamics research [5].

Recently, several "bottom-up" alternatives to traditional motion and behavior planning architectures have been proposed. Rodney Brooks introduced the *Subsumption Architecture* [6], a layered scheme for control in which multiple semi-autonomous behaviors run simultaneously and compete for control of the actuators. Behaviors communicate sensory and actuator information to each other through low-bandwidth communication channels. In the event of conflicts, a fixed-priority arbitration scheme determines the winner. Brooks' robots have exhibited emergent high-level behaviors that qualitatively exceeded those of prior modular and structured approaches. Masahiro Fujita of Sony has used a Subsumption Architecture to create an autonomous pet-like robot for entertainment [7].

Ken Perlin introduced another departure from kinematics and dynamics related to the Subsumption Architecture. His *Improv* system [8,9] uses a generative approach for motion synthesis. In similar spirit to Brooks' work, Perlin ignores kinematics and dynamics to produce motion from simple primitives. Perlin's primitive behaviors, however, are based on continuous periodic noise functions that when combined to drive a jointed computer graphic creature, result in natural and smooth real-time animations. His higher level abstractions of behavior control are not as strictly hierarchical as Brooks' architecture, allowing arbitrary inter-relationships between behaviors, including mechanisms such as lateral inhibition, mutual exclusion (seen much earlier in the work of Ludlow [10]) and constraint satisfaction.

Other non-traditional approaches include Blumberg's work on motor control and ethologically-inspired action-selection for directable characters [11], and Karl Sims' work on feedback learning methods based on evolution [12].

Recent work in modifying an arbitrary motion to alter its emotional content has had some promising results. Several computer graphics researchers have found preliminary success in using digital signal processing algorithms to alter the emotional quality of sampled or generated computer animation [13,14]. A simple observation, for example, is that high-pass filters can make movement appear more nervous or exaggerated and low-pass filters result in sad or sluggish perceived behavior.

Finally, the principles of traditional animation as developed over the last century are a major source of inspiration for creating character and emotion in an inanimate being. Thomas and Johnson of Disney Studios have made one of the best reference books for understanding the techniques of traditional animation [15]. Of the many principles, ones that immediately stand out as lessons for the robotics community include:

Overlapping Action. Actions overlap in space and time. For example, while a person walks towards a door, he is reaching for the doorknob at the same time. This term also refers to the time lag between body parts – for example, a character's eyes will follow the point he is tracking much more quickly than his head moves.

Follow Through. Actions do not come to an immediate halt but continue on after the end-goal of the motion is achieved, as in a baseball swing.

Anticipation. Before entering into an action, a character will anticipate such action, giving the impression of thought or preparation.

Arcs. Objects follow curved paths in the real world, due to body geometry and basic physics.

Ambient Motion. Real creatures never come to a complete stop – there is continual movement and rhythm such as nodding, blinking, bobbing and shifting of weight.

Ease In/Ease Out. Motions begin and end with a ramp up to speed and a slow down to a halt. It is impossible for real creatures to make immediate and precise changes in position or velocity.

The vast majority of modern robotics architectures routinely violate these principles. For example, the work of Brooks exhibits discontinuities in both motion and behavior as layers are subsumed. It can be argued that these principles represent the most important challenge to the development of a life-like affective robot.

LAYERED MOTION SYNTHESIS

As we approached the problem of affective motion control, we were heavily influenced by lessons learned from classical animation. Ken Perlin's work offered us the most promising set of tools for applying the principles of animation to an emotive robotic character. Architecturally, our system is derived substantially from his published accounts of creating directable computer graphics characters with the Improv system. However, we have built our own system from the ground up, incorporating important additions necessary for applying motion synthesis to a mechanical system.

The principle of layered motion synthesis is fundamentally different from goal-directed path planning, kinematics and dynamics. Layered motion synthesis represents a *generative* model for movement based on the weighted composition of low-level periodic functions. On the surface, a generative system, which is not physically or mechanically based, might seem to be inappropriate for modeling physical behavior. However, in order to create the illusion of intelligence and sentience, a system based on rhythmic functions makes sound theoretical sense, since periodic functions, or rhythms, have been both theorized and

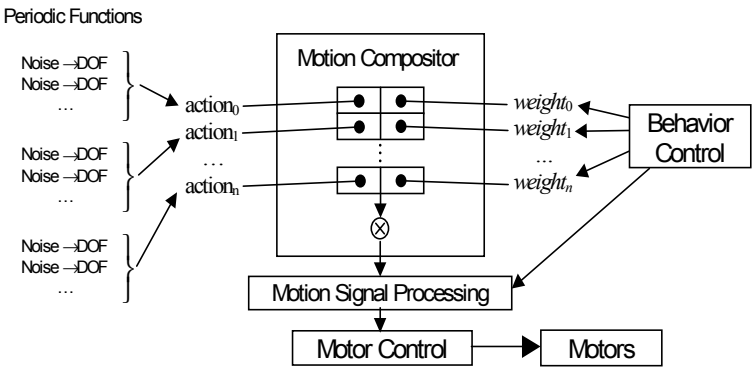


Figure 1. Motion synthesis architecture overview. The Motion Compositor serves as the engine for motion synthesis. The primitives of motion consist of actions, which in turn consist of a series of mappings from periodic functions to degrees of freedom. The top-level Behavior Control module continuously varies a set of weights, which combine with actions in the Motion Compositor to generate target values for the degrees of freedom. A Motion Signal processing layer can add final overall changes to affect before the values are sent to the Motor Control layer which delivers actual signals to the motors of the robot.

qualitatively observed to lie at the core of animate movement and cognition [16].

Our periodic functions are generated with statistically controlled noise ("Perlin Noise" [17]) to mimic the unpredictability and noisiness of animate motor control. This approach is in stark opposition to traditional mechanical approaches, since deliberately introducing noise makes a mechanical system less efficient.

The synthesis process consists of the following hierarchically arranged functional elements (Figure 1). First *Actions* are constructed based on the periodic function primitives. Each action consists of a list of individual mappings from a periodic function to a mechanical degree of freedom. At the next level, a *Motion Compositor* takes a combination of actions and their corresponding control weights to produce the base synthesized motion. The *Motion Signal Processing* module modifies the base motion with a digital filterbank, producing overall changes to affect. Finally, at the topmost level, the *Behavioral Control* module conducts the entire process by sending the appropriate time varying control weights to the Motion Compositor.

The Behavioral Control module requires a method for changing the action weights over time to produce shifts in the robot's behavior. We drive our robot as a puppet with a human operator as the behavioral engine. The operator makes discrete changes to the robot's state which are integrated to compute continuously changing action weights. In this way we are able to carry out experiments in mediated emotive communications. The existing literature in behavioral control is extensive and most of these methods could be used to drive our motion synthesis engine. Of particular interest to us is the recent work by Kline and Blumberg on the continuous modification of behavior weights using functional networks [18].

SOFTWARE ARCHITECTURE DESIGN

The layers of our software architecture consist of distinct modules in a hierarchical organization. The details of each layer are described below, from the lowest to the highest level of abstraction.

MOTOR CONTROL

The lowest level of our system consists of a mechanism for delivering motion commands to a mechanical robot with multiple degrees of freedom (DOF). This layer handles the real-time delivery of target values to the mechanical degrees of freedom, computation of derivatives for smooth motion, calibration, error correction, time keeping and other utilities.

At this software layer, mechanical degrees of freedom can be registered as either *absolute* or *relative*. Absolute degrees of freedom, such as a neck angle or eyebrow position treat the incoming values as position targets. Relative degrees of freedom, such as wheels, treat the incoming target values as velocities.

The target values are queued in a small cache to compute numerical first and higher order derivatives – required signals that ensure smooth motor movements.

PERIODIC FUNCTION GENERATORS

The functions out of which all motions are synthesized consist of three simple primitives, all of which provide values between 0 and 1 inclusive. The Raised Sine and Cosine functions are normalized sine and cosine values with a period of one:

$$\mathrm{rsin}(t) = \frac{1 + \sin(2t\pi)}{2}$$

In each of these functions, t refers to the real-time clock in units of seconds. The Noise function is similarly a continuous periodic function with zero-crossings at integral values of t. However, the maximum value and derivative between crossings varies smoothly and randomly (Figure 2). There are many ways of generating such noise functions. We chose a fast method which involves caching gradients to a Hermite spline at each zero crossing, then computing Hermite spline values for particular values of t on the fly. Several tables of gradients can be pre-computed to provide an assortment of smooth noise from which to choose.

From these primitives, other simple functions can be constructed. As an example, consider a function which can

be used for blinking eyes. We call this function *Random Triggered Sine* and it can be expressed algorithmically as follows:

```
function rTrigSin(t)

if (NOT blinking AND random value > trigger
     threshold AND time since last blink > minimum
     time between blinks)
then
        blinking = TRUE
        startTime = t
else if (t-startTime > 1)
        blinking = FALSE

if (blinking)
        return rsin(t-startTime)
else
        return 1.0
```

The output of this and the other periodic functions is used to interpolate between high and low joint values in each action.

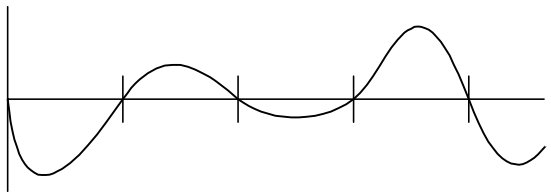


Figure 2. Periodic noise function. Our basic periodic noise primitive has zero crossings at integer values with smoothly varying randomness in derivative and amplitude.

ACTION REPRESENTATION

Actions consist of a list of mappings from periodic functions to degrees of freedom. An action consists of high and low values for a joint combined with a standard or custom function based on the periodic primitives, which is used to interpolate between the two extreme values. An action need not involve all degrees of freedom and most only use a subset. An action can be succinctly describes as a short list:

```
(DOF, LOW VALUE, HIGH VALUE, FUNCTION)
```

As an example, consider our `NEUTRAL` action:

```
(NECK, 0.6, 0.7, NOISE2)
(EYEBROW, 0.8, 0.85, NOISE1)
(UPPERLIP, 0.4, 0.5, NOISE3)
(LOWERLIP, 0.4, 0.5, NOISE3)
```

The various noise functions `NOISE1`, `NOISE2` and `NOISE3` represent different cached sets of coherent noise to avoid undesired perceptual coupling between mechanical degrees of freedom.

MOTION COMPOSITOR

The Motion Compositor takes a list of actions and a set of control weights and computes joint angles and motor velocities at each temporal frame. These values are then delivered to the robot via the Motor Control layer. Actions are composited using two techniques. The first is a *channel mask*. Unused degrees of freedom (channels) in a particular action are not averaged into the final composite motion, unduly influencing other Actions' effects. Instead, the unused channels are selectively masked out, so that only those channels with a defined mapping are averaged into the final signal. This technique is analogous to the process of visual masking familiar to users of Adobe Photoshop™ and other digital compositing tools for images where parts of an image are selectively allowed to show through using a visual mask. The second technique is the method for summing channel values. A *convex sum* is computed, where final joint values are computed by dividing the sum contribution of values for each particular degree of freedom by the sum of the weights for that degree of freedom. At a given instant in time, the value of a particular joint *j* can be expressed as:

$$\frac{\sum_i Mask_{ij} \times Weight_{ij} \times Action_{ij}}{\sum_i Mask_{ij} \times Weight_{ij}}$$

Where $Action_i$ refers to the vector $[v_1, v_2, \ldots v_n]$ of target joint values generated by action *i* for joint *j* and $Mask_i$ is a vector whose components have a value of 1 for a joint used by this action and 0 otherwise. This combination results in a simple and scalable technique for mediating multiple and changing numbers of actions, varying weights of actions and actions influencing only particular parts of the robot's body.

When actions are registered with the Motion Compositor, they are given a *tempo* which is a multiplier for the base frequency of the periodic functions. In order for actions to blend smoothly into the final composited motion, *tempo matching* is used to gradually warp from the time of the action to the time of the master real-time clock so that phases of actions always match.

MOTION SIGNAL PROCESSING

A post-processing layer above the compositor allows us to experiment with overall changes to affect. We have currently implemented a bandpass filter as described by Bruderlin and Williams [13]. This allows us to set positive or negative gains on seven frequency bands, resulting in overall changes to affect. As predicted, accentuating high frequencies results in nervous, anxious behavior. Accentuating low frequencies results in a more lugubrious or relaxed perceived motion.

One problem with motion signal processing is that we must introduce a small delay into the output signal in order to compute our digitally filtered output. In our current case this window is 5 samples, which introduces a significant delay of 170ms. However, we get around this problem by using predictive methods to compute ahead. Since all of our periodic functions are explicit functions of time, we can

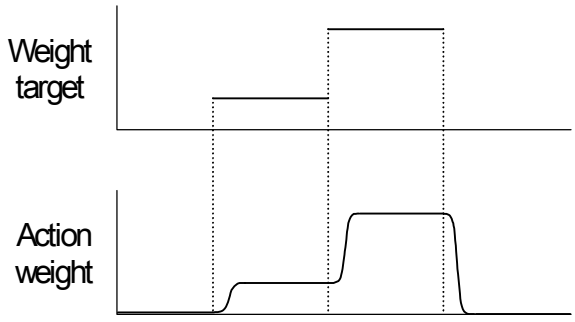


Figure 3. Integrating weight targets. Discrete weight target values are integrated using an exponential filter to compute continuously changing action weights.

compute future values using either the current action weights, or extrapolations to future weights using a technique such as Kalman filtering. Thus we can remove all of the delay and perform our signal processing on the predicted sample values.

BEHAVIORAL CONTROL

This is the topmost layer as well as the highest abstraction level of our motion synthesis engine. This layer sends control weights to the Motion Compositor and Motion Signal Processing layers to trigger the different motion behaviors. In a typical application, this layer has a set of 20-30 actions from which 3-6 actions typically have non-zero weights at any given time. A behavior is defined as set of concurrent actions and their weights, for example:

```
{[RANDOMBLINK, 0.05],
 [NEUTRAL BODY, 0.05],
 [NEUTRAL HEAD, 0.05],
 [HAPPY FACE, 1.0],
 [PIGEON WALK, 1.0]}
```

Actions with low weights are ones which can be easily suppressed by heavier-weighted actions.

Our research to date has focused on driving the robot's behavior with medium-level control over emotion and movement. Essential to our solution is the smooth varying of weights over time to produce smooth behavioral transitions. We have adapted Ken Perlin's method of integrating weights to change discrete commands into continuously varying signals. Our particular method is to introduce desired weight targets at discrete temporal intervals which are gradually integrated into the actual Action weights using an exponential filter (Figure 3). This guarantees continuity of position and velocity and enhances the sense of perceived character by avoiding abrupt transitions.

Perlin has also observed that simple suppression of actions via higher weight values does not work for more complex sets of behaviors, which we have also verified in our initial experiments. This occurs because multiple actions begin to compete for the same DOF, with their average contribution resulting in nonsensical movements. In such cases, we have considered several approaches towards mediating competing behaviors:

Lateral inhibition. We borrow this common technique from neural network research. As the weight for one behavior rises, it inhibits the rise of the weight for a second behavior. E.g. the happy behavior suppresses the sad behavior, since these two cannot sensibly co-exist at once.

Functional feedback and feed-forward systems. Feedback and feed-forward systems can be used to create complex and naturalistic transitions, even chaotic non-repeating systems of behavior. Recent promising work in this domain comes from the Synthetic Characters Group at MIT where all behavior is boiled down to continuously changing real numbers which represent drives, emotions and actions [18]. By using a strictly numerical approach, functional networks are constructed from accumulators, sensors and transducers to create a sophisticated interplay between internal motivation and external action.

Subsumption Architectures. As in Rodney Brooks' work, finite-state-automata can be used for making discrete changes to desired state. Coming even closer to his work, competing finite-state-automata could communicate using the current action weights as the "model" corresponding directly to the robot's behavior and blurring the program/data distinction.

To date, we have only experimented with the first two techniques (lateral inhibition and feedback/feed-forward) in our research.

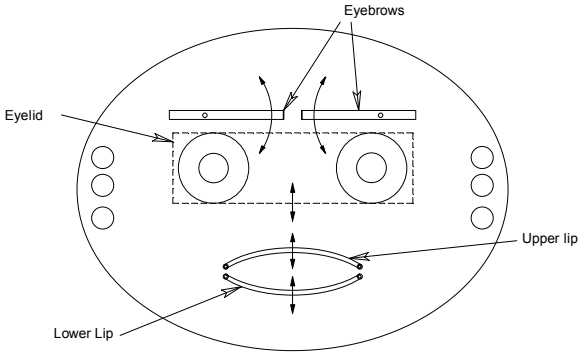
IMPLEMENTATION DETAILS

ROBOT DESIGN

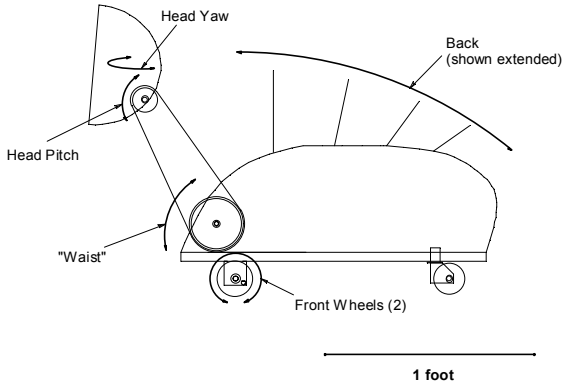
Our robot is a mobile platform with all of its sensing, actuating and computing resources on-board. The robot has a Pentium 200 PC/104 stack with cards for networking, sound production and power regulation. Attached to the stack are three Motion Engineering 104/DSP motion control cards. These control cards, together with custom amplifiers built here at Interval, give us 10 channels of motion control on the robot.

The first four degrees of freedom are dedicated to facial expression (Figure 4a). The eyebrows move together and are capable of portraying an emotional range from surprise to extreme anger. The eyelids are similarly a single DOF and allow blinking but not winking. Each lip has a single DOF. The lips can bend in both directions, allowing us to make, for instance, a true frown or a thin smile.

The remaining six degrees of freedom are dedicated to larger, "whole body" motions (Figure 4b). There are two degrees of freedom in the head that allow for 180 degrees of yaw and 100 degrees of pitch. At the "waist," where the long neck meets the body, is a single degree of freedom. By using this joint we can portray various postures such as sitting up straight or leaning forward. Two degrees of freedom are employed in the front of the robot, one for each drive wheel. Finally, there is a DOF implemented in the back to allow it rise up or flutter down in a movement reminiscent of a cat whose back can arch in fear or pleasure. An up-and-active state conveys fear, anger, dominance, etc. while a gentle fluttering shows calm or ease, like a bird ruffling its feathers. The back also serves to amplify the posture created by the waist.



(a)



(b)

Figure 4. Schematic diagram of the robot. In (a) the four facial degrees of freedom are shown: eyebrow, eyelid, upper and lower lip. In (b) the six degrees of freedom for the body are illustrated: head yaw and pitch, neck, wheels and back. The scale is only accurate for the body illustration.

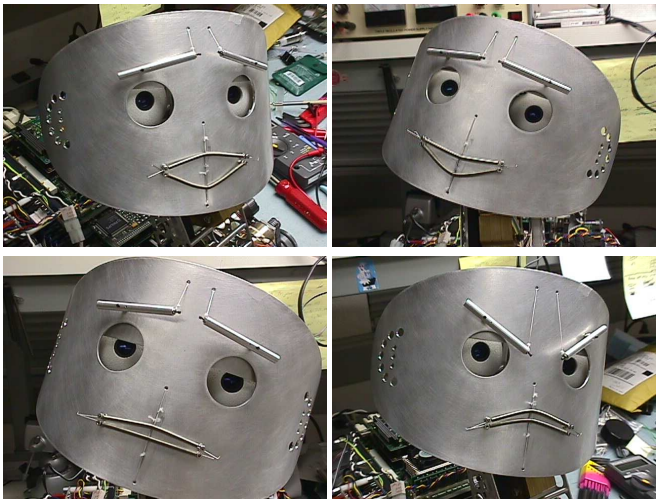


Figure 5. Range of facial expressions. Clockwise from upper left: surprise, happiness, melancholy, anger.

SOFTWARE INFRASTRUCTURE

Our software is implemented as a set of C++ classes running on the QNX operating system. QNX has provided us with a stable and robust real-time architecture for delivering continuous control commands to our robot. Motor control is performed by sending velocity frames at 30Hz to the motor control cards.

The software consists of the six layers as described in this paper: Motor Control, Periodic Function Generation, Actions, Motion Compositor, Motion Signal Processing and Behavior Control. These layers run in two processes – a dedicated process runs the Motor Control layer to continuously update the motor control cards, while the other process runs the remaining motion synthesis layers as a single serial process.

PUPPET CONTROL OF THE ROBOT

The focus of this research has been the development of an underlying architecture that is capable of generating subtle and sophisticated changes to perceived emotional states. We have not yet addressed autonomy. To enable immediate testing of our emotive communication ideas, we arrived at the concept of treating our robot as a puppet.

Figure 6. Our robot in action. In the top row, left to right: afraid and inquisitive. Below: happy and angry.

There are three main puppetry controls that we found to be necessary: the robot's gaze direction, mobility controls, and emotional states. These target motion control parameters are sent to the Behavior Control layer, and processed such that the human operator can tell the robot where to go and look, but the robot performs these actions based on its "mood." A video game controller retrofitted with a small microcontroller to allow RS-232 serial access turned out to be a great match for these tasks (Figure 7).

In our pilot application, we use the left joystick to control gaze direction, the right joystick for mobility movement of the robot, right buttons for facial emotions, left buttons for overall body moods and a few of the other buttons for miscellaneous control (kill, reset, etc.) Our experiments

Figure 7. Videogame controller. We use a retrofitted off-the-shelf video game controller for high level control of gaze, movement and emotion. The well-designed physical interface allows the operator to quickly master the capabilities of the robot.

with this platform will be described in more detail in future papers, however we found that having a hand-held, well-designed controller radically altered our testing and debugging experience, allowing us to work in the same space as our robot and use physical memory[2] for the control.

RESULTS

We have found our system to be a robust method for implementing complex emotionally expressive movement of a robot's face and body. Some images of the robot in action can be found in Figures 5 and 6. However, to appreciate this work, one must see the motion performed over time. We have several movies of our robot available for download on the World Wide Web at:

http://www.interval.com/papers/1999-033/index.html

FUTURE WORK

We are currently in the process of performing cognitive experiments with our robot in controlled and open spaces to test peoples' responses to an affective robotic character. Also planned are public excursions, to understand better the effectiveness of mediated emotive communication with a group.

We will continue to improve the quality of our behaviors through handcrafting of the actions by animators. On the physical form, we would like to explore the expressiveness of a robot without a face, as animators have previously indicated that emotion can be expressed purely through

[2] *Physical memory* is the body's ability to "remember" kinesthetic experience or spatial arrangements, without explicit cognition. For example, riding a bike, knowing the controls to your car or the keyhole and doorknob to your home, etc.

movement even without facial cues [15]. We would like to experiment with different body materials. In particular, we think that a softer robot might be more appealing and approachable, as evidenced by the success of recent robotic plush toys available to consumers such as Microsoft ActiMates™ Barney™ and Tiger Electronics' Furby®. These toys currently rely more on sound than movement for their emotional communication. We believe our architecture might make a strong addition to the sense of character projected by such toys.

Most notably lacking in our research to date is autonomy. We are looking forward to exploring the possibility of interfacing cognitive and autonomous behavior architectures to the current system in order to create the illusion of sentience. The overall design of our motion synthesis engine allows the system to be driven by any behavioral models, including the traditional modular AI, a Subsumption Architecture, or a canned sequence as seen in Disney's animatronic characters. However, we're especially interested in models that include internal notions of drive and emotion. We believe that these models could benefit by showing their internal state as visible mood. This idea seems in line with Brooks' thinking – "use the world as your model" can be flipped around to "use your body as your storage of state."

ACKNOWLEDGEMENTS

We would like to acknowledge Rob Tow, who initiated the original robotics project at Interval and provided the founding principles for exploring an emotionally expressive robot. Jesse Dorogusker, John Ananny, Paul Korff, Gerald Rogerson, Brad Niven, Dan Psomas and the rest of the Interval Shop staff have all made strong contributions to the robot. Jesse is also responsible for our game controller retrofit. Finally, we would like to thank Chris Kline, an Interval fellow from the MIT Media Laboratory's Synthetic Characters group, with whom we had lots of productive discussions during his one-month stay at Interval.

REFERENCES

[1] Rosalind W. Picard. *Affective Computing,* MIT Press, 1997.

[2] Jean-Claude Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[3] M. Brady, J. Hollerbach, T. Johnson, T. Lozano-Pérez, M. Mason, Eds., *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.

[4] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics* 26(2): 303-308, 1992.

[5] N. Badler, B. Barsky, D. Zeltzer, *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[6] R. Brooks. A Robust Layered Control for a Mobile Robot, *IEEE Journal of Robotics and Automation*, 2(1):14-23, 1986.

[7] M. Fujita and H. Kitano. Development of an Autonomous Quadruped Robot for Robot Entertainment, *Autonomous Robots*, 5, 7-18, 1998.

[8] K. Perlin, Real Time Responsive Animation with Personality, IEEE Transactions on Visualization and *Computer Graphics* (SIGGRAPH '95 Proceedings), August 1995.

[9] K. Perlin, A. Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds, *Computer Graphics*: 29(3), 1996.

[10] A. Ludlow. The Behavior of a Model Animal. *Journal of Behavior* 58. 1976.

[11] B. Blumberg, T. Galyean, Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments, *Computer Graphics* 30(3): 47-54, 1995.

[12] Sims, Karl. Evolving Virtual Creatures, *Computer Graphics* (SIGGRAPH '94 Proceedings), July 1994

[13] A. Bruderlin, L. Williams. Motion Signal Processing, *Computer Graphics*: 30(3), 1995.

[14] K. Amaya, A. Bruderlin. Emotion from Motion, in *Graphics Interface '96*, Proceedings, Toronto, Canada, May 1996, pp. 222-229.

[15] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Hyperion. 1981.

[16] Thaddeus L. Bolton. Rhythm, *The American Journal of Psychology* (VI. 2), Jan. 1894: 146-47.

[17] K. Perlin. An image synthesizer. *Computer Graphics*: 19(3), 1985, pp. 287-293.

[18] C. Kline and B. Blumberg. The Art and Science of Synthetic Character Design. *Convention of the Society for the Study of Artificial Intelligence and the Simulation of Behavior (AISB), Symposium on AI and Creativity in Entertainment and Visual Art,* Proceedings, Edinburgh, Scotland, April, 1999.