Section I: Introduction

Ia. Abstract

Keepon is a robot whose movement and body lend themselves to dance-like motion. Of particular interest was the effect of pre-processing songs for the purpose of both predicting beats and automatically choreographing complex dance patterns that would be difficult to attain in real time. The implemented architecture employs specialized beat cleaning and Markov chaining to create natural dance patterns that appear superior to a merely beat-sensitive system. The results obtained confirm the advantage of foreknowledge of a song before dancing, and further indicate that Markov processes have utility in dance choreography generation. This work is continued in Part II.

Ib. Background

Keepon Platform
Keepon is a small robot developed by Hideki Kozima in Tokyo, Japan. Bright yellow in color and designed to appear nonthreatening, Keepon was designed to study social development in children, particularly in cases of autism spectrum disorders. Physically, Keepon is about a foot tall, the bottom half of which is a black base in which most of its mechanical parts are contained. The upper half of Keepon is the body of the robot, which is composed of two yellow spheres. Keepon has four degrees of freedom - it can lean side-to-side, scrunch up and down, pan left and right, and nod up and down. Keepon's flexible rubber skin allows for easy deformation (which in turn creates the perception of fluid motion), and its face consists of two small video cameras for eyes and a microphone nose/mouth. Despite its simple design, viewers consistently are able to perceive Keepon as having an emotional state.

Previous Work and Project Goal
Keepon has gained non-trivial notoriety through a viral YouTube video of it dancing to Spoon's "I Turn My Camera On". This video, constructed by Carnegie Mellon PhD student Marek Michalowski (formerly of Yale's own Social Robotics Lab), demonstrated the realism with which Keepon could dance, given its four mechanical degrees of freedom and flexible rubber exterior. However, the Michalowski video was based on a hand-scripted set of dance movements.

The project, undertaken by the two authors and Chris Riederer (SY '10) in Brian Scassellati's spring 2009 class (CPSC 473: Intelligent Robotics), set out to develop a dancing engine for Keepon that could dance to any song without human input. The three students ultimately combined simple movements that responded to music in real-time to elicit an
emergent pattern of dance. The results highlighted two important points. First, that simple sets of moves could be combined in unique and interesting ways, and second, that there were inherent limitations with an entirely real-time ("online") dance engine.

The authors hope to combine these two takeaways to construct a new dance engine that allows Keepon to both anticipate future beats in a song and also combine simple and complex moves into elegant and emergent patterns.

Section II: A Tripartite Model of Dance

IIa. Tripartite Model Overview

A sequence of three programs allows Keepon to dance to any song; they will be described in the next three sections. The song MP3 is input to the first program, a beat detector. This detector outputs a series of beats and their intensities, which acts as input to the second program, a beat processor. The move selector is the most complex program. It is the only analytic portion of the three programs, and outputs a song-specific script of moves for Keepon to perform to the given beat. Finally, this script is sent to an executor, which plays the song audio and issues commands to Keepon at appropriate times. This architecture is exciting because it allows interchangeability, should our group or another find a substantially better move selector or beat detector in the future.

IIb. Program I: Beat Detector

The beat detector translates MP3 files into a series of beats. Using a Java sound library called Minim, the beat detector uses an FFT to convert the sound energy to the frequency domain, where it then breaks down the range of sound into 27 channels. The detector measures the change in which channels are past a certain intensity threshold and reports the time the change occurred as well as how many channels have shifted (known in the program as “intensity”). The beat detector has an adjustable overall threshold, used to allow parsing of the softest songs and avoid beat over-detection in the loudest.

IIc. Program II: Beat Processor
See “Keep On, Keepon, Part I” by Matthew Du Pont

IId. Program III: Executor
See “Keep On, Keepon, Part I” by Matthew Du Pont

IIe. Advantages of the Model

Beat cleaning
See “Keep On, Keepon, Part I” by Matthew Du Pont

Simple / Complex Move Structure
The dance engine’s move structure is divided into two parts - simple and complex moves. Simple moves are merely movements of one of the four motors, and are encoded as relative moves (i.e.: "pan 40" pans 40 degrees to the right, rather than panning to position 40). Complex moves are written as sequences of simple moves, with timings attached. The alternative to this system would have been to send commands to Keepon separated by timed sleep commands. This system, however, would lead to Keepon occasionally stalling and moving in jerky fashion. Instead, by simply issuing commands a certain distance apart, Keepon’s motors do not have to stop and wait before the next command is issued.

Complex dance moves thus have three parts - first, the sequence of moves themselves, second, the timings associated with the move sequence, and third, a set of position vectors which indicate the total motor displacement of the movement. Some moves, such as a "bop" (down and then up again), have no total motor displacement, but obviously keeping track of motor displacement has implications for containing Keepon’s movements to in-bounds and aesthetically interesting dance.

**Randomization via Markov Chaining**

There were several ways to approach randomization of dance moves. The simplest, which was used as a baseline standard for our experimentation, was to randomly select dance moves. This is clearly better than wholly random motion, as a Keepon dancing with this selection mechanism will stay on beat and will still employ complex dance moves. However, the overall structure of the dance seems arbitrary with this selection mechanism. A more intuitively appealing option that was considered was assigning phenotypes to dance moves and selecting dance moves based on the circumstances - longer beats would warrant longer moves, while loud beats could warrant other sorts of moves. While this would have corrected the problem of apparently arbitrary move sequences, it would still be susceptible to less-than-fluid transitions between movements. Also, a phenotype-based system would require a vast library of dance moves to be non-deterministic (every phenotype would need to have several options to avoid the repetition of one or two dance moves to a repeated sound or refrain). A similar but slightly different system that would have had similar problems would have been a filter-system, which starts with a universe of dance moves and filters out moves for being inappropriate in a given situation. While this system is possible given the pre-processing time and computational power available to Keepon, it would still have required a vast library of dance moves to produce meaningful results, and would have required long and detailed hard-coding of rules that would have ultimately been specific to certain types of music.

Instead of these alternatives, the dance engine uses Markov chaining to obtain random yet meaningful dance patterns. Markov chains have been used in generation of music (for example, in Professor Hudak’s CPSC 431: Fundamentals of Computer Music: Algorithmic and Heuristic Composition), generation of superficially intelligible sentences, and generation of new Garfield comics (called "Garkov" by its creator).

A Markov chain is a memoryless stochastic process that can be thought of as a series of finite state machines. A better way to visualize a (finite state space) Markov chain made up of elements $X_1... X_n...$ is to consider an n-by-n matrix, where each row is made up of non-
negative values that sum to 1 (called a stochastic matrix). Then, the value of the ordered pair \((i, j)\) is \(P(X_{n+1} = j | X_n = i)\). Markov chaining is useful to the dance engine for at least three reasons. First, Markov chains allow for the creation of apparent patterns, including looping and cycling sequences. Second, in the absence of actual ‘seeding’ for the dance patterns of Keepon, fine-tuning the stochastic matrix is easy and intuitive. Third, Markov chains are memoryless, which means that each of Keepon’s dance moves are chosen based on merely what Keepon did last, rather than what Keepon did several moves ago. By discretizing this system, the fine-tuned stochastic matrix can produce aesthetically pleasing dance patterns for a variety of different musical types. While long chains of ten or twelve particular moves would look good for some types of music but horrendous for others, the probabilities of Keepon moving from Move A to Move B are now based entirely on what moving from Move A to Move B looks like.

**Improved User Feedback Integration**

The new architecture facilitated two important qualitatively derived improvements. First, the previous architecture outputted one move for every detected beat. In the beat cleaning section of this paper, it was shown that a lower rate than one move per detected beat is often desirable, as numerous detections lead to jerky motions and unfinished moves. However, in one case the group found it desirable to issue multiple instructions per beat (possible in emergent architectures, but omitted in the previous project). Informal discussions with a number of people who have watched Keepon dance led us to believe the “bop” motion, bobbing down and up quickly, makes a series of moves seem more dancelike. Responding to this feedback, the decision to bop under the new architecture is independent of other move selections – thus a bop is layered on to the other, often more complex, moves that Keepon undertakes.

Additionally, the Markov chain model of move selection in the new architecture made it simple to change the probability of moves and combinations quickly. By putting all probabilities in one matrix, rather than tagging moves into different categories or having a random system with selection probabilities scattered throughout the code, the group was able to quickly respond to informal feedback on what looked like natural dance and minimize combinations of incongruous moves: for example, when the group received repeated feedback that Keepon looking upwards appeared more natural, it was the work of minutes to adjust its move profile. This is valuable for future work as different Markov chains can be developed for different scenarios, perhaps to create the effect of “moods” or emulate certain people/styles of dance.

**Section III: Evaluation**

See “Keep On, Keepon, Part II” by Matthew Du Pont

**Section IV: Conclusion, Allocation of Responsibility, and Future Work**

**IVa. Conclusion**

The project’s goal was to construct a new dance engine taking advantage of both the natural appearance of emergence-based move selection and the refinement and planning of
a preprocessing architecture. Through a tripartite model of beat detection, beat processing, and execution, an emergent architecture was successfully modified: instead of merely reacting to the present, Keepon now draws upon the information from the song as a whole when selecting moves.

IVb. Allocation of Responsibility

Unmentioned areas of the project were done in collaboration.

Julian Rajeshwar designed the tripartite model of preprocessing (beat detection, beat processing, move execution). He was responsible for the creation of the beat detector used in the first stage of preprocessing. Julian also designed and implemented the Markov process used for move selection in the second half of preprocessing, along with creating the Simple/Complex Move classes used in creating complex moves. Julian wrote the move executor, which took the move instructions and executed them in coordination with the music.

Matthew Du Pont was responsible for designing and implementing the move selection architecture of the beat processing program. He also created the representation of the beat sequences that were passed from the first to the second stage of preprocessing. He was primarily responsible for survey design and analysis, as well as tuning timing of when moves take place relative to their beats.

IVc. Future Work

Demos
Though it is outside the scope of the project, both the authors and their advisor have expressed interest in creating publicly available demos, both for elementary school student participants in robotics outreach programs as well as YouTube demonstrations. These demos will be filmed in a time of lower academic pressure, and given to the advisor by January 15, 2010.

Extension of Tripartite Model
The model allows easy scripting of dances and other movements for the Keepon platform using only a text editor with little programming experience. This makes scripting more easily accessible to future programmers and users. Additionally, a different move selector or beat detector could be integrated with our other two programs, allowing future projects to focus solely on one of those components.

Dynamically Computed Markov Chains
As the user feedback section points out, different "personalities" could be manifested by changing move selection probabilities (based on genre information from the ID3 tag of the MP3 being played, for example). This adaptability could be further expanded by recomputing the probabilities as conditions change, perhaps with live user feedback on what moves are and are not desirable.
**Generated Markov Chains**

If given access to the scripted dances used by Michalowski in the aforementioned Youtube videos, the generation of Markov chains from these existing and aesthetically pleasing dance sequences could be invaluable in creating the appearance of natural dancing.
Works Referenced


Julian Rajeshwar, Matthew Du Pont, and Christopher Riederer. Keepon Dancing.

Appendix
User Survey Form Example

Name:

College:

Experience in (circle all that apply):

Robotics/AI

Music

Please write a few words or phrases you would use to describe Keepon's movements.

How would you rate Keepon's overall dancing ability?

1 2 3 4 5 6 7

Very poor

Very good

Did the timing of Keepon’s moves correspond with the timing of beats in the song?

1 2 3 4 5 6 7

No Correspondence

Perfect Correspondence

Would you characterize Keepon's moves as jerky or fluid?

1 2 3 4 5 6 7

Jerky

Fluid

How many distinct movements do you think Keepon made during the song?