Autonomous Animation of Humanoid Robots

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Mechanical Engineering
(NTU-CMU Dual Ph.D. Degree Programme)

Junyun Tay

B.Comp., Information Systems, National University of Singapore
M.S., Mechanical Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA
May, 2016

To my loving husband, Somchaya Liemhetcharat,
for his unwavering support and encouragement.

Acknowledgments

First and foremost, I would like to express my utmost gratitude to my advisers, Manuela Veloso (co-chair) and I-Ming Chen (co-chair) for their invaluable guidance and advice. The two of them were vital to the process for gaining entry to the NTU-CMU Dual Ph.D. Programme in Engineering (Robotics). Manuela provided many opportunities for me to participate in RoboCup competitions and outreach programmes like Creative Technology Nights, which led to my growing interest in AI and robotics. These opportunities laid the foundation for my Ph.D. by equipping me with the knowledge about the NAO humanoid robots and prompted me to discover the exciting worlds of AI and robotics. Manuela's passion for AI and robotics has also been infectious and I will never forget her reminding me of Allen Newell's "The science is in the details!". I was also extremely fortunate to meet and talk to I-Ming during RoboCup 2010 held in Singapore and found out about the NTU-CMU Dual Ph.D. Programme. The conversation with I-Ming was key to gaining entry to the two Ph.D. programmes at NTU and CMU. I-Ming had also generously provided me with resources to support my Ph.D. work.

I would also like to thank Jonathan Cagan and Song Huat Yeo for graciously agreeing to be on my thesis committee, and providing valuable feedback for the thesis.

I am forever indebted to my husband, Somchaya Liemhetcharat, who supported me in all possible ways during my Ph.D. The countless hours of research discussions and sharing of experiences throughout this challenging and exhilarating journey have been cathartic. No amount of words can describe my gratitude to him as a companion in life and research and I hope we have many years to go. My daughter, Dhanaphon Liemhetcharat, has also been extremely supportive with her mantra, "You can do it!" that she learned from her dad. Her cheeky antics have provided much joy and relief. It is my regret that I cannot spend enough hours with her through her early childhood, but hope that she will eventually benefit from the contributions I made with this thesis.

I am also grateful to my parents and family for putting up with my decision to follow Somchaya to CMU with no idea of what I would be doing in Pittsburgh and giving up on a job that provided financial support. I thank them for their continuous love, encouragement, understanding and support in my crazy decision to do a Ph.D., particularly in the field of Mechanical Engineering, given that I had little background in it.

I am also thankful for the help and support of many friends and colleagues in the CORAL group at CMU and the Robotics Research Centre at NTU over the years. They are in no particular order: Mike Phillips, Brian Coltin, Susana Brandão, Çetin Meriçli, Stephanie Rosenthal, Joydeep Biswas, Prashant Reddy, Richard Wang, Rachel Jackson, Josh Caputo and Myunghee Kim from CMU, and Albert Causo, Emily Toh, Charles Ng, Qilong Yuan, Bingbing Li, Huixu Dong, Teguh Lambono, Conghui Liang and Lili Liu from NTU. I am also grateful for the administrative support given by Christina Contreras and Chris Hertz at CMU, Agnes Tan, Eng Cheng Lim and Meow Chng Soh at NTU, which has enabled me to focus on my research.

I would like to express my gratitude and appreciation to Changjiu Zhou, Gerald Seet and Nadine Aubry for their advice regarding the NTU-CMU Dual Ph.D. programme. I am also grateful to enjoy the support of my friends in the NTU-CMU Dual Ph.D. programme: Chun Fan Goh, Guo Zhan Lum, Juanjuan Zhang and Wei Sin Ang, and the Singaporeans in CMU: Ying Ying Wu, Wee Liat Ong and Matthew Lee. I am very grateful to the Economic Development Board of Singapore that graciously funds my graduate studies at NTU and CMU.

Last but not least, I am thankful for the Singaporean community in Pittsburgh that made Pittsburgh a second home to us: Rodney and Suzanne, Jiaqi and Weiling, Chwee Ming and Andrea, Kenneth and Linli, Bryan and Cocoa. Your love and help have been forthcoming and readily extended in times of need and support.

Abstract

Gestures and other body movements of humanoid robots can be used to convey meanings which are extracted from an input signal, such as speech or music. For example, the humanoid robot waves its arm to say goodbye or nods its head to dance to the beats of the music. This thesis investigates how to autonomously animate a real humanoid robot given an input signal. This thesis addresses five core challenges, namely: **R**epresentation of motions, **M**appings between meanings and motions, **S**election of relevant motions, **S**ynchronization of motion sequences to the input signal, and **S**tability of the motion sequences (R-M-S^3). We define parameterized motions that allow a large variation of whole body motions to be generated from a small core motion library and synchronization of the motions to different input signals. To assign meanings to motions, we represent meanings using labels and map motions to labels autonomously using motion features. We also examine different metrics to determine similar motions so that a new motion is mapped to existing labels of the most similar motion. We explain how we select relevant motions using labels, synchronize the motion sequence to the input signal, and consider the audience's preferences. We contribute an algorithm that determines the stability of a motion sequence. We also define the term relative stability, where the stability of one motion sequence is compared to other motion sequences. We contribute an algorithm to determine the most stable motion sequence so that the humanoid robot animates continuously without interruptions. We demonstrate our work with two input signals – music and speech, where a humanoid robot autonomously dances to any piece of music using the beats and emotions of the music and also autonomously gestures according to its speech. We describe how we use our solutions to R-M-S^3, and present a complete algorithm that captures the meanings of the input signal and weighs the selection of the best sequence using two criteria: audience feedback and stability. Our approach and algorithms are general to autonomously animate humanoid robots, and we use a real NAO humanoid robot and in simulation as an example.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A humanoid robot is designed to resemble a human, with a head, torso, two arms and two legs. As such, a humanoid has multiple degrees of freedom (DOFs) that can be actuated to form whole body motions. Since humanoid robots share a similar appearance as humans, the whole body motions of humanoid robots can serve as non-verbal behavior in social interactions with humans. For example, an ASIMO interacts with a live host in the show, "Say 'Hello' to Honda's ASIMO", at a Disneyland park [Honda, 2005], and a NAO humanoid robot acts as a concierge to guests at the Hilton McLean hotel and uses its motions as gestures to complement its speech that is powered by IBM's Watson program [Statt, 2016]. Since the articulated AIBO robots, followed by QRIO humanoids, we have many demonstrations of a variety of artistic expressions. For example, a QRIO humanoid conducts the Tokyo Philharmonic Orchestra that played Beethoven's Fifth Symphony [Geppert, 2004] and 540 humanoid robots perform a synchronized dance routine during the 2016 Chinese New Year-themed variety show [Reich, 2016].

When a humanoid robot is animated using whole body motions, the goal is to convey the meanings of an input signal, such as speech or music. A good animation requires that the motions are synchronized to the input signal. Whole body motions of humanoid robots are complex and are carefully configured so that the humanoid robots remain stable throughout the animation given the multiple DOFs. Thus, the motions of humanoid robots are mostly pre-programmed by motion choreographers or programmed to mimic human motions through motion capture data, e.g., [Nakaoka et al., 2005]. Moreover, when the input signal changes, e.g., a new speech or piece of music, the motion choreographers manually create new stable whole body motions that

express the new input signal and synchronize the motions to the input signal. Such manual animation does not easily enable the general use of robots. Similarly, new motion capture data are collected for a new input signal, modified so that the motions are stable, and also satisfy the physical constraints of the humanoid robot (i.e., its joint angle and velocity limits). *This thesis investigates how to autonomously animate a humanoid robot given an input signal while ensuring that the robot remains stable throughout the animation.*

The input signal is used as a guide to plan the humanoid robot's whole body motions. In this thesis, the input signal is pre-processed offline to identify the meanings to express. We represent meanings using labels and extract labels and their timings by pre-processing the input signal. The pre-processing of the input signal is synonymous to how humans study the script of a play to understand what a character needs to portray, before acting in synchrony with the character's speech; similarly, dancers analyze the dance music before dancing to the beats and mood of the music. *This thesis explains how an input signal is pre-processed and used to select and plan the whole body motions of a humanoid robot.*

Whole body motions are smooth and continuous and are represented in different ways. For example, whole body motions are represented as continuous joint trajectories using Kochanek-Bartels (TCB) cubic splines [Ng et al., 2010] or as a set of key poses that are interpolated with fixed times [Zheng and Meng, 2012]. The choice of the representation of whole body motions affects how motions are modified to be synchronized with the input signal, and whether the same motion is used in different input signals to convey the same meaning. Also, the representation of motions affects the number of motions defined in a motion library. For example, suppose there are two similar head nod motions – one motion where the head of the robot nods from 5° to -5° to 5°, and another motion where the head nods from 10° to -10° to 10°. Should similar motions be represented as separate motions or as a single parameterized motion? *This thesis uses parameterized motions, to reduce the size of the motion library and to synchronize the motions to the input signal.*

In this thesis, motions are mapped to labels in order for relevant motions to be autonomously selected to animate the input signal. Motions and labels have a many-to-many relationship, i.e., a motion is mapped to multiple labels and a label is mapped to multiple motions. Manually labeling the motions becomes laborious as the number of motions and labels increases. *This thesis investigates how to automatically map motions to labels, by extracting the features of the motion, and by measuring the similarity between a new motion and existing motions.*

For every label in the input signal, there are multiple motions mapped to the same label. When there is no exact match between the label of the motions and the label in the input signal, we identify labels with similar meanings to the label in the input signal. Hence, there are multiple choices of motions to convey the same or similar meaning(s) represented by each label in the input signal. *This thesis describes how we select relevant motions autonomously based on the labels of the motions and the labels identified in the pre-processed input signal to form a motion sequence, and contributes an algorithm to autonomously synchronize the motion sequence to the input signal.*

Actors in a play adapt their acting to the preferences of the audience using the audience's feedback such as loud cheers or applause. This thesis uses the audience feedback to learn the preferences of motions so as to execute the most preferred animation on the humanoid robot. *This thesis investigates how the audience preferences are modeled, uses the learned model to improve the selection of preferred sequences, and also models the effects of boredom where the audience gets bored when they are shown the same motions repeatedly.*

Continuous animation of the humanoid robot requires the humanoid robot to remain stable. When the humanoid robot falls, the animation is disrupted. Moreover, falling may result in disastrous consequences where the humanoid robot breaks or damages its joint(s) and the humanoid robot is no longer capable of performing the motions using its whole body. Even if the robot is able to recover from a fall without any damage, a disruption to the animation of the input signal is still undesirable. *This thesis describes how we identify unstable sequences that should not be executed, and also selects the most stable sequence to execute, so that the robot is able to continue to execute other sequences continuously without interruption.*

When there are multiple motions with the same label, and multiple labels in the input signal, multiple sequences of motions are feasible to animate the input signal. The multiple sequences of motions are analogous to the different animations created by different motion choreographers for the same input signal. Similarly, actors act differently given the same script as there are multiple motions that convey the same meaning. The humanoid robot is only capable of executing a single sequence of motions to animate the input signal at a particular instance. *This thesis contributes a complete algorithm that considers how to select the best motion sequence by taking into account the meaning of the input signal, the synchronization of the motion sequence to the input signal, the audience preferences, and the stability of the motion sequence.*

Our approach is general for any pre-processed input signal that comprises labels and the

timings of the labels, and we select two input signals to demonstrate how autonomous animation of humanoid robots is achieved. Specifically, we use stories that are converted from text to speech, and various pieces of music. The approach of this thesis is applicable to humanoid robots that have a torso, two arms and two legs, and we choose a commercially-available humanoid robot – the NAO humanoid robot – to demonstrate our work.

1.1 Thesis Question and Approach

After introducing the challenges involved in autonomously animating a humanoid robot given an input signal, we present the thesis question, provide an overview of our approach and briefly describe our solutions to the five core challenges, namely Representation, Mappings, Selection and Synchronization, and Stability (R-M-S^3).

The thesis question is:

> In order to autonomously animate a humanoid robot given an input signal, how do we ***r**epresent* motions, automate ***m**appings* between motions and meanings, ***s**elect* the relevant motions and consider the audience's preferences, ***s**ynchronize* motions to the input signal, and determine a ***s**table* sequence of motions?

Overview of our approach

The autonomous animation of humanoid robots involves solving five core challenges we term as Representation-Mappings-Selection-Synchronization-Stability (R-M-S^3). We present our approach to the thesis question and summarize our approach in Figure 1.1. We describe our solutions to R-M-S^3 and also contribute a complete algorithm that utilizes these solutions – AAMPS – **A**utonomous **A**nimation that conveys the **M**eaning of the input signal and considers audience **P**references and **S**tability of the motion sequences.

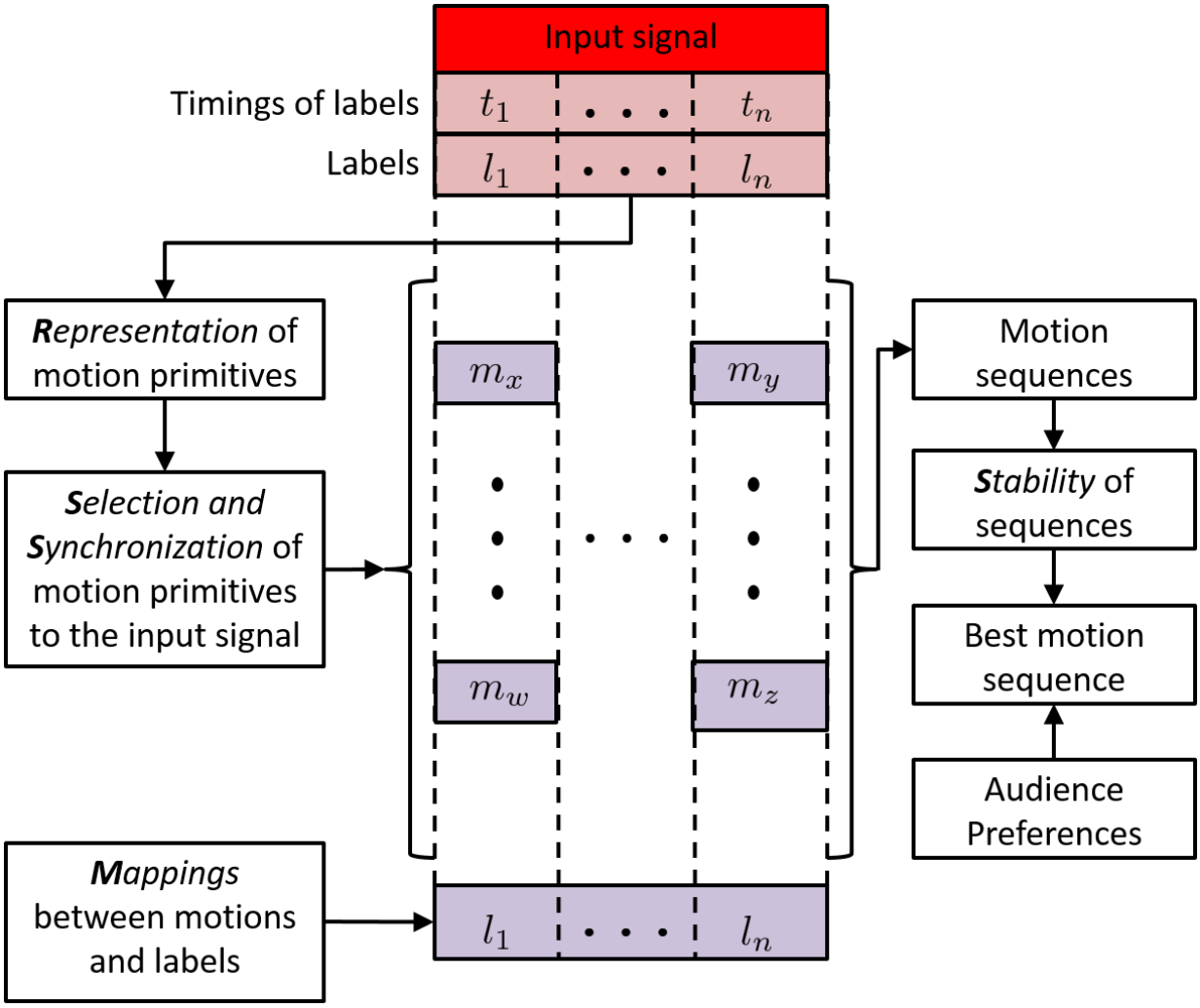


Figure 1.1: Overview of our approach.

Representation

We formally define robot motions in a motion library, where the motions are parameterized such that motions are varied and synchronized to different input signals. Motions are also categorized based on the features of the motions and we demonstrate how these categories reduce redundancies in the motion library but yet, still be able to create many variations of motions. We also introduce the concept of a spatially targeted motion, i.e., a motion that is directed at a target of interest. For example, a storytelling robot waves hello to a friend when meeting a friend in the

story by using a spatially targeted motion.

We also formally define a pre-processed input signal where labels and the timings of the labels are extracted. We discuss the relationship between labels of the motions and the labels of the input signal.

Mappings

We contribute an approach to autonomously assign labels to motions based on the features of the motions. This approach is useful as it becomes more tedious to manually create mappings between motions and labels whenever there is a new label or motion in a large motion library.

When features of the new motion to be added to the library are not available, we examine different metrics to determine similar motions. We find an existing motion in the library that is the most similar to the new motion and propose possible mappings between the new motion and the labels of the existing motion.

Selection and Synchronization

We present an approach to select motions based on the similarity between the labels of the motions and the labels of the input signal, and synchronize the motions to the input signal. We demonstrate how motions are selected probabilistically, or by selecting the highest ranked sequence.

Many motion sequences are feasible to animate the same input signal since there are multiple motions with the same label and there are many labels in the input signal. The humanoid robot executes one motion sequence and we aim to select the most preferred sequence according to the audience's preferences of the motions. We contribute an approach to model the audience's preferences of the motions and determine the most preferred motion sequence based on the feedback of the audience at the end of the motion sequence. Given that the audience may get bored of watching different sequences of motions for the same input signal, we do not execute all possible sequences to determine the most preferred sequence. In our approach, the robot learns from the feedback of executed sequences and determines the next sequence to execute, balancing exploration (collecting feedback on a new sequence) and exploitation (selecting the best sequence in its model). We also show that we do not execute all possible motion sequences to find the most preferred sequence of motions for the input signal.

Stability

We aim to determine the stability of motion sequences so that the humanoid robot animates the input signal continuously without interruption. We contribute an algorithm that predicts if a motion sequence is stable without executing the motion sequence, nor modeling the dynamics of the humanoid robot. Although each motion in the motion library is assumed to be stable, a sequence of various motions may be unstable. Hence, the stability of a sequence of motions before executing the sequence on the robot must be known. The stability of a sequence of motions is important for two reasons:

1. The meanings of the input signal are successfully conveyed by the robot when the robot remains stable after executing the sequence of motions.
2. If the sequence of motions executed by the robot causes the robot to fall, severe wear and tear of the robot may occur. If the robot repeatedly falls, the robot may not be able to execute whole body motions and is only capable of actuating some joints.

We contribute an algorithm to determine the most stable sequence of motions by comparing the relative stability of a sequence among other possible sequences of motions generated for the same input signal. We choose the most stable sequence so that the robot continues to execute more sequences of motions without interruption.

Complete algorithm – AAMPS

We contribute a complete algorithm – AAMPS – **A**utonomous **A**nimation that conveys the **M**eaning of the input signal and considers audience **P**references and **S**tability of motion sequences. AAMPS plans the best sequence of motions for a labeled input signal, that fulfills these conditions:

1. Motions are relevant given that the labels of the motions are similar to the labels of the input signal.
2. Motions are synchronized to the timings of the input signal's labels.
3. Motions are stable and do not cause the robot to fall.
4. Motions that are preferred by the audience are rated higher and selected.

AAMPS selects relevant motions, generates multiple sequences of motions that are synchro-

nized to the input signal, ranks the sequences based on a weighted criteria consisting audience preferences and stability so as to select the best sequence of motions.

We assume that the autonomous generation of whole body motions for a humanoid robot is possible when the following conditions are met:

- The physical constraints of the robot are known, e.g., joint angular limits and joint velocity limits.
- The motions in the motion library are defined for the humanoid robot.
- The motions in the motion library are labeled.
- The input signal is pre-processed to determine labels that represent the meaning of the input signal and timings of the labels.
- There exists a list of criteria to determine the best sequence of motions.
- Each motion in the motion library is stable.
- The body angle trajectories for each motion and interpolations between pairs of motions are collected using the inertial measurement unit of the robot.

Throughout the thesis, the Aldebaran NAO humanoid robot is used to demonstrate our algorithms and approaches. The Aldebaran NAO humanoid robot is a fully autonomous robot with an internal CPU and sensors. Though we use the NAO robot in this thesis, our algorithms and approaches are general for humanoid robots that are similar to the NAO robots. Besides collecting data on the NAO humanoid robot, we also use Webots 7 [Webots, 2014], a real-time simulator that simulates the dynamics of the NAO robot in physically realistic worlds.

To evaluate this thesis, two input signals are considered: stories that are converted to speech using a text-to-speech program, and various pieces of music. We demonstrate that we generate stable relevant motions synchronized to speech for a story-telling robot, and stable relevant motions synchronized to music for a dancing robot. Our approach and algorithms are general to be used for humanoid robots that are similar to the NAO humanoid robots, and input signals that are pre-processed to determine the labels and the timings of the labels.

1.2 Contributions

The main contributions of this thesis are:

- A formal definition of robot motions which allows many variations of motions that are used in different input signals and the motions are synchronized to the input signals;
- A formal definition of pre-processed input signals;
- An algorithm to automatically map labels to motions;
- A metric to determine similar motions to map existing labels to new motions;
- An approach to generate variations of sequences of motions probabilistically and synchronize the motions to the input signal;
- An approach to select the best sequence by ranking different sequences of motions based on different criteria;
- An algorithm to select the best sequence based on audience preferences;
- An algorithm to predict the stability of a sequence of motions (i.e., whether the robot is stable after executing the sequence) without a model of the robot or prior executions of the sequence;
- A formal definition of relative stability where the stability of a sequence is compared to the stability of other sequences;
- An algorithm to predict the most stable sequence by comparing the relative stability of a sequence among other sequences using executions of other sequences of motions;
- A complete algorithm of autonomous robot animation that captures signal meaning and weighs the criteria of audience preferences and stability.

1.3 Thesis Document Outline

Figure 1.2 presents an overview of the chapters in this thesis. The outline below presents a summary of the following chapters:

- Chapter 2 presents an overview of the five core challenges to autonomous animation of humanoid robots, namely **R**epresentation, **M**appings, **S**election, **S**ynchronization and

Stability (R-M-S^3). We also explain how we utilize our solutions to R-M-S^3 in the complete algorithm for **a**utonomous robot **a**nimation to capture signal **m**eaning and weigh the criteria of audience **p**reference and **s**tability – AAMPS.

- Chapter 3 describes how we formalize robot motions and pre-processed input signals. This chapter illustrates the relationship between the labels of the robot motions and the labels of the input signals. This chapter also describes how we create the motion library and input signals we use to demonstrate our work.
- Chapter 4 explains how we autonomously map labels to motions and reduce the work to manually label motions. This chapter also discusses the metrics used to determine similar motions so as to map existing labels to new motions.
- Chapter 5 explains how we select relevant motions based on labels that capture the meanings of the input signal and audience preferences and synchronize motions to the input signal.
- Chapter 6 presents an approach to predict the stability of a sequence of motions (whether the robot is stable or falls after the execution of the sequence) with no prior execution data and no model of the dynamics of the humanoid robot. This chapter also presents an approach to predict the relative stability among a set of sequences and determine the most stable sequence.
- Chapter 7 discusses related research with respect to the different aspects of R-M-S^3 and how they relate to this thesis.
- Chapter 8 concludes with a summary of the thesis' contributions and a discussion of future work.

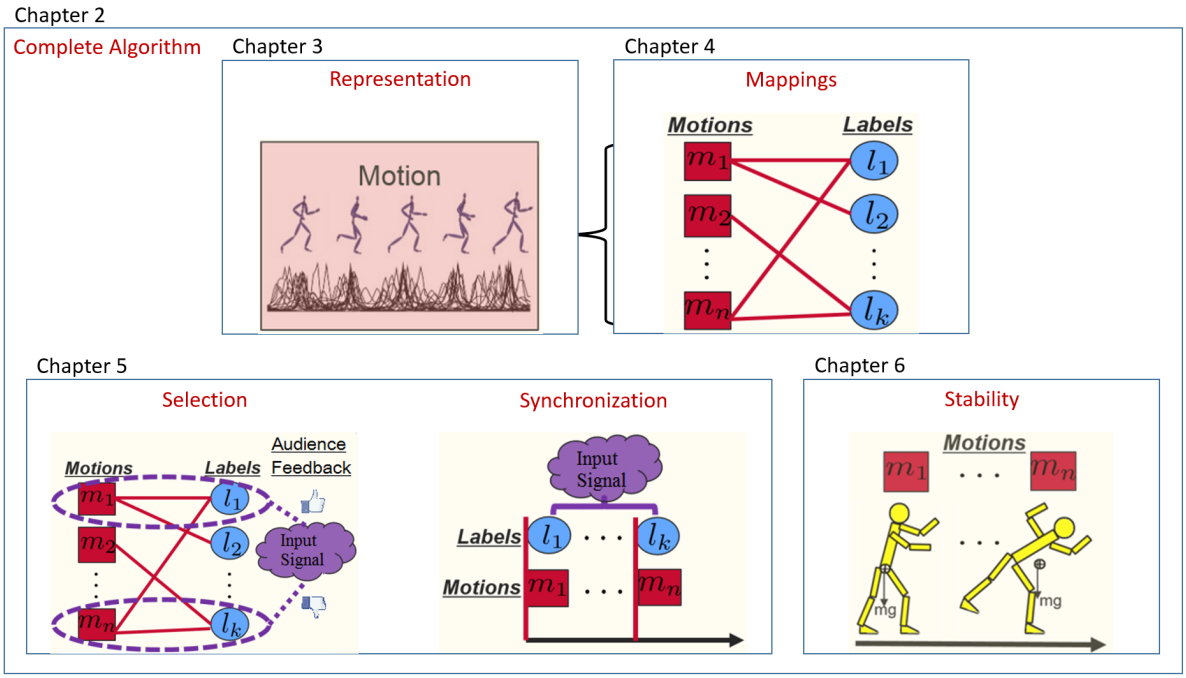


Figure 1.2: Overview of the thesis chapters.

Chapter 2

AAMPS – The Complete Algorithm

This chapter formally defines the five core challenges – "Representation", "Mapping", "Selection", "Synchronization" and "Stability" (R-M-S^3) and how we use our proposed solutions in the complete algorithm for autonomous robot animation that captures signal meaning, audience preference and stability. We also list the assumptions made for each core challenge. We explain our complete algorithm – **A**utonomous robot **A**nimation that captures signal **M**eaning, audience **P**reference and **S**tability of motion sequences (AAMPS). The following chapters will describe how we address each core challenge in detail.

2.1 Problem Statement

We formally define the core challenges in R-M-S^3 to lay the foundation of how AAMPS provides a complete solution and addresses these challenges.

2.1.1 Representation of Robot Motions and Input Signals

We have a motion library that contains parameterized motion primitives and an input signal to animate. A motion is an instantiated motion primitive where all the parameters of the motion primitive are defined. The representations of parameterized motion primitives and input signals are illustrated in Chapter 3.

Definition 2.1.1. *Let a parameterized motion primitive be m. Let l^m be a label assigned to the motion primitive m. Let the set of parameterized motion primitives in the motion library be M,*

and let the set of labels be L. Let the set of labeled parameterized motion primitives in the motion library be M^L.

An input signal is pre-processed to determine the labels and the timings of the labels.

Definition 2.1.2. *Let a pre-processed input signal be s, and the set of all signals be S.*

Assumptions

We assume that the motion primitives in the motion library are defined for the humanoid robot used to animate the input signal. The labels of the motion primitives are either manually defined or mapped using our autonomous mapping algorithms, which require examples of labeled motions to map existing labels to the new motions.

2.1.2 Mapping between Motions and Labels

If the mappings between motions and labels are manually defined, it becomes increasingly challenging when the library of motions and the labels increase. This thesis investigates how to autonomously map motion primitives to labels with a function $\mathbb{X}$. With the function $\mathbb{X}$, we generate mappings between the parameterized motion primitives in the motion library and labels. We explain the function $\mathbb{X}$ in detail in Chapter 4.

Definition 2.1.3. *Let the function to map motion primitives to labels be $\mathbb{X} : M \times L \rightarrow [0, 1]$, where $\mathbb{X}(m, l)$ determines if the motion m is mapped to the label l.*

Assumptions

We assume that there exists examples of labeled motions that our autonomous mapping algorithms can use to map existing labels to the new motions. These examples of labeled motions are defined using our representation of motion primitives.

2.1.3 Selection of Motions based on Labels and Audience Preferences

Selection of relevant motions requires a match between the label of the motion primitive to the label of the input signal. We also consider audience preferences of the motions. We explain in detail how we select the motions based on labels and audience preferences in Chapter 5.

There may not be a perfect match in the meanings between the labels of motion primitives and the labels of the input signal. To determine how well the labels match, we define a function $\mathbb{S}$ that determines the similarity of the labels in meaning.

Definition 2.1.4. *The function $\mathbb{S} : L \times L \rightarrow [0, 1]$ determines the similarity in meaning between two labels. $\mathbb{S}$ returns a value of 1 when there is a perfect match in the meaning between two labels or when the two labels are the same. $\mathbb{S}$ returns a value of 0 when there is no similarity in the meaning of the label.*

Besides selecting motion primitives based on the similarity between labels, we also consider the audience preferences of motions. We consider that feedback of the audience is given at the end of a performance. We are not able to get feedback for every motion primitive in the sequence, but at the end of a sequence, we observe the audience feedback, i.e., a preference value.

Definition 2.1.5. *A sequence of motion primitives $u^s = (m_1, \ldots, m_d)$ is an ordered set of d motion primitives for a pre-processed input signal s, where $d \geq 2$. Let U be the set of all possible sequences of motion primitives.*

We define a function $\mathbb{A}$ that returns the audience feedback of a sequence.

Definition 2.1.6. *The audience preference value of a sequence of motion primitives is determined by the function $\mathbb{A} : U \times S \rightarrow \mathbb{R}^+$.*

Assumptions

We assume that for all the labels in the input signal to be animated, there are motions in the motion library with the same label or similar labels depending on the similarity function defined. We provided examples of the similarity functions for emotion labels and text labels in Section 5.1 and Section 5.2.

We created a model of the audience preferences of motion sequences and assume that the audience provides feedback using the model. The model is described in Section 5.3.

2.1.4 Synchronization of Motions to Input Signal

A sequence of motion primitives is synchronized to the input signal to animate the input signal, otherwise it is awkward to see a robot animate out of sync with the input signal. The function $\mathbb{H}$ synchronizes a sequence of motion primitives to the input signal. We describe the function $\mathbb{H}$ in detail in Chapter 5.

Definition 2.1.7. *The function $\mathbb{H} : U \times S \rightarrow U$ synchronizes a sequence of motion primitives to the input signal. If the sequence of motion primitives is not synchronized due to the constraints on the duration of the motions and labels, $\mathbb{H}$ returns $\emptyset$.*

Assumptions

We assume that the interpolation time is manually defined or determined using the fastest velocity of the motors involved. We also assume that the function $\mathbb{H}$ is defined, e.g., to synchronize the start of a motion to the start of the corresponding label or to start within a certain time at the start of the label.

2.1.5 Stability of a Sequence of Motion Primitives

In Chapter 6, we investigate how to determine if a sequence of motions is stable to execute using a humanoid robot and also find the most stable sequence from the list of possible sequences.

Definition 2.1.8. *The function $\mathbb{F} : U \rightarrow \{0, 1\}$ computes the **stability** of a sequence of motion primitives, where $\mathbb{F}(u^s) = 1$ if and only if u^s is feasible. A sequence of motion primitives is **feasible** if and only if the robot is able to execute the keyframes whilst being **stable**.*

Being stable means that the humanoid robot remains on its two feet, where only the base of the robot's feet remains in contact with the ground.

We also investigate the problem of determining the relative stability of a sequence given the set of possible sequences. Relative stability of a sequence refers to how stable a sequence is as compared to other sequences in the set of possible sequences. We determine the most stable sequence from the relative stability of the sequences. By choosing the most stable sequence to execute, the robot maximizes the chance that the robot continues to animate other input signals without interruption.

Definition 2.1.9. *The function $\mathbb{U}(u^s, U^s) \rightarrow [0, 1]$ determines the relative stability of a sequence u^s as compared to other sequences in U^s, where U^s is the set of possible sequences for the pre-processed input signal s. The function $\mathbb{U}$ returns a value of 0 when u^s is the least stable sequence in U^s and returns a value of 1 when u^s is the most stable sequence in U^s.*

Assumptions

We assume the following:

- The robot starts each motion sequence with the same keyframe.
- Individual motions in the motion library are stable. We also assume that the body angles X (roll) and Y (pitch) sensor readings are available via the inertial measurement unit of the humanoid robot.
- Data are collected on the same humanoid robot that is animating the motion sequences and that the prediction of the stability of the motion sequence is made for the same humanoid robot.
- There is no wear and tear.

2.2 AAMPS – The Complete Algorithm

After formalizing R-M-S^3, we introduce the complete algorithm – AAMPS shown in Figure 2.1. We describe how AAMPS utilizes the solutions to the following aspects for **a**utonomous robot **a**nimation:

1. ***M**eanings of signal* are captured by determining relevant motion primitives based on the labels of these motion primitives and the labels in the input signal;
2. ***P**references of the motion primitives* from the audience feedback;
3. ***S**tability of sequences* of motion primitives.

Meanings of signal

To determine relevant motion primitives, we use the function $\mathbb{S} : L \times L \rightarrow [0, 1]$ defined previously. There may not be a perfect match between the label of a motion primitive to the label of the pre-processed input signal, meaning that $\mathbb{S}$ returns a value less than 1 for all motion primitives in the library for a particular label in the signal, l^s. Hence, we select motion primitives where the function $\mathbb{S}$ returns a value larger or equals to μ, where $\mu \in [0, 1]$. For example, if we set $\mu = 0.8$, only motions with labels that have a similarity value of at least 0.8 with the signal's label are selected.

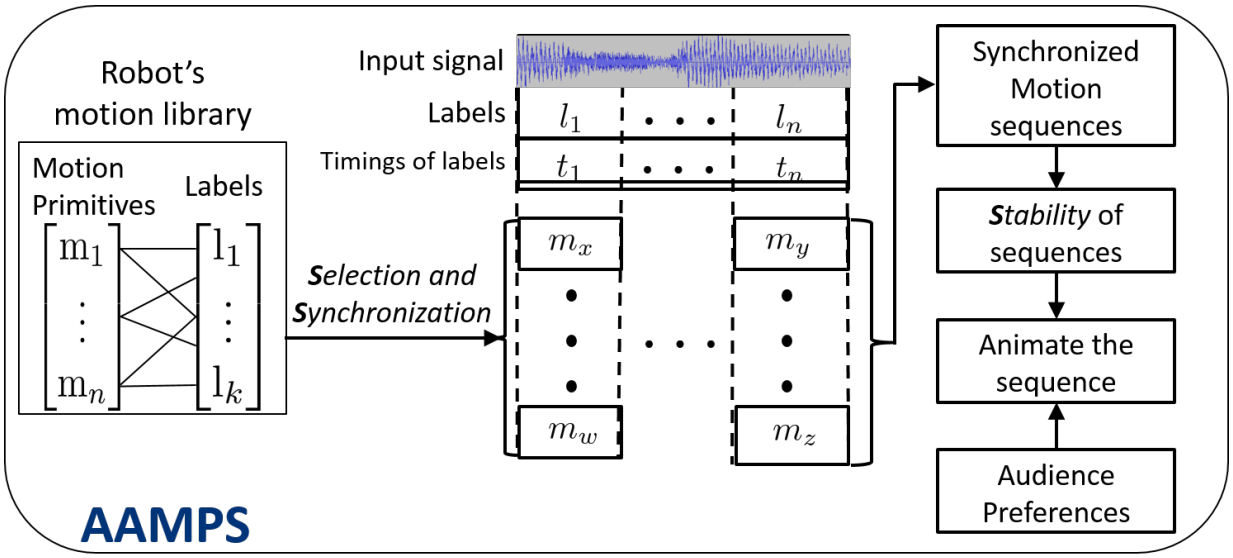


Figure 2.1: Overview of AAMPS.

By identifying relevant motion primitives for each label in the pre-processed input signal, we generate combinations of possible sequences for all labels in the pre-processed input signal s. After generating these combinations, we synchronize each sequence of motion primitives to the input signal using the function $\mathbb{H}$. If a sequence of motion primitives is not synchronized to the input signal, we discard that sequence of motion primitives.

Definition 2.2.1. *The function $\mathbb{SS}(M^L, s, \mu) = U^s$ selects the relevant motion primitives from a library of labeled motion primitives, M^L using μ, generates all possible combinations of sequences, synchronizes each sequence to the input signal s using $\mathbb{H}$ and returns a set of synchronized sequences of motion primitives, U^s.*

Preferences of the motion primitives

We use $\mathbb{A} : U \times S \rightarrow \mathbb{R}^+$ to determine the audience preference (ratings) of stable sequences U^{ss}. We use the terms audience preference and audience rating interchangeably from here on. We normalize the ratings of these sequences using the highest and lowest rating in U^{ss}.

Definition 2.2.2. *Let $\hat{A}^{max} = \max_{u^{ss} \in U^{ss}} \mathbb{A}(u^{ss})$ be the maximum rating in U^{ss} for a sequence u^{ss}. Let $\hat{A}^{min} = \min_{u^{ss} \in U^{ss}} \mathbb{A}(u^{ss})$ be the minimum rating in U^s.*

Definition 2.2.3. *Let $\hat{A}^{u^{ss}} = \frac{\mathbb{A}(u^{ss}) - \hat{A}^{min}}{\hat{A}^{max} - \hat{A}^{min}}$ be the normalized rating of sequence $u^{ss} \in U^{ss}$. The function $\mathbb{P}$ returns the normalized rating, i.e., $\mathbb{P}(u^{ss}, U^{ss}) = \hat{A}^{u^{ss}}$. If all the ratings of the sequences are the same, $\hat{A}^{u^{ss}}$ is set to 1.*

Stability of sequences

We use $\mathbb{F} : U \rightarrow \{0, 1\}$ to determine if a sequence is feasible (stable). We discard sequences in U^s that are unstable.

Definition 2.2.4. *Let u^{ss} be a synchronized sequence of motion primitives that is stable, i.e., $\mathbb{F}(u^{ss}) = 1$. Let U^{ss} be the set of stable and synchronized sequences of instantiated motion primitives.*

Next, we determine the relative stability of each sequence u^{ss} among the sequences in U^{ss} using the function $\mathbb{U}$. The relative stability of the least stable sequence is 0 whereas the relative stability of the most stable sequence is 1. If all sequences are just as stable, the relative stability of each sequence is 1.

AAMPS

The inputs to the complete algorithm, AAMPS (Algorithm 1), are a labeled motion library, M^L, and a pre-processed input signal, s. AAMPS considers all three aspects – **S**ignal **M**eaning, **S**tability and **A**udience **P**reference – by using the following parameters:

- μ changes the number of relevant motion primitives.
- γ is the weight assigned to the relative stability of a sequence, u^{ss}, and $1 - \gamma$ is the weight assigned to the normalized rating, $\hat{A}^{u^{\text{ss}}}$, where $\gamma \in [0, 1]$.

First, AAMPS determines U^s, a set of synchronized sequences of relevant motion primitives, using the function $\mathbb{SS}$. AAMPS goes through each sequence in U^s to determine if the sequence is stable. Next, AAMPS calculates the score for each sequence using the weights assigned to the normalized audience rating and the normalized relative stability. Lastly, AAMPS finds the best sequence with the highest score.

Algorithm 1 **A**utonomous robot **A**nimation that captures signal **M**eaning, audience **P**reference and **S**tability (AAMPS).

AAMPS(M^L, s, μ, γ)
  $U^s \leftarrow \mathbb{SS}(M^L, s, \mu)$
  $U^{\text{ss}} \leftarrow \emptyset$
  for $i = 1$ to $|U^s|$ **do**
    if $\mathbb{F}(u_i^s) = 1$ **then**
      $U^{\text{ss}} \leftarrow U^{\text{ss}} \cup \{u_i^s\}$ // Determine stable sequences
    end if
  end for
  scores $\leftarrow \emptyset$
  for $i = 1$ to $|U^{\text{ss}}|$ **do**
    score$_i \leftarrow \gamma \cdot \mathbb{U}(u_i^s, U^{\text{ss}}) + (1 - \gamma) \cdot \mathbb{P}(u_i^s, U^{\text{ss}})$ // u_i^s is the i^{th} stable sequence in U^{ss}
    scores $\leftarrow$ scores $\cup \{\text{score}_i\}$
  end for
  best $\leftarrow \text{argmax}_{i \in \{1,\ldots,|U^{\text{ss}}|\}}$ score$_i$
  return $u^{\text{ss}}_{\text{best}}$

Assumptions

We assume that γ is defined and that we only consider relative stability and audience preferences to select a motion sequence for the humanoid robot to animate. We assume that there is only one label to be animated at any instance, i.e., we do not animate multiple labels at once.

Discussion

In this section, we discuss how AAMPS is used. If we only want the most stable sequence, we set $\gamma = 1$. If we only want the most preferred sequence, we set $\gamma = 0$. If we want to achieve a balance between the two criteria – relative stability and the audience preference, we set $\gamma = 0.5$. We note that the lower the value of μ, the higher the number of sequences generated.

We do not include the evaluation of how relevant each motion primitive is as part of the criteria. Even though we select motion primitives based on the similarity between the labels of the motion primitives and the labels of the input signal, there is no guarantee that the motion primitives truly express the meaning of the labels. There may be nuances in the input signal that are not determined just by using the labels in the input signal. For example, when the input signal

is the sentence – "He pretends to be happy, but he is really sad". We will generate sequences that will animate the words "happy" and "sad" when these words are associated to motions in the library. However, the true preference may be only on sequences that animate the word "sad". In such cases, we rely on the criterion of the audience preference to determine the best sequence.

Using Algorithm 1, we select the best sequence using the highest score. If there are multiple sequences with the highest score, we select a sequence randomly. We can also use the scores to select a sequence probabilistically by converting the scores into probabilities. The higher the score of a sequence, the higher the probability that the sequence is selected. Using a probabilistic approach, we select different sequences for the same input signal at different instances. This probabilistic approach is useful when we present variations in the animations, e.g., to generate a dance for a piece of music. However, as the values for relative stability and audience preference are normalized, there will be a sequence with a score of 0 for the relative stability and another sequence with a score of 0 for the audience rating. If we set $\gamma = 1$ or $\gamma = 0$, there will always be a sequence that is not selected using the probabilistic approach.

The reader may think that γ should be set to 0 since that all sequences in U^{ss} are stable. However, we highlight that no algorithms are foolproof. The function $\mathbb{F}$ may not be able to accurately determine whether all the sequences are stable, i.e., there may be some sequences that are misclassified. By using the function $\mathbb{U}$ to determine relative stability of sequences, we minimize the probability that a sequence that is unstable and deemed stable by $\mathbb{F}$ will be executed. When γ is not set to 0, we consider the relative stability of sequences and maximize the probability that the robot will continue to execute more stable sequences without interruption.

2.3 Chapter Summary

This chapter presents an overview of the five core challenges and formalizes each challenge – **R**epresentation of motions, **M**appings of motions to meanings where meanings are represented as labels, **S**election of relevant motions that considers the similarity between labels and audience preferences, **S**ynchronization of motions to the input signal to form motion sequences, and **S**tability of the motion sequences (RMS^3). We contribute an algorithm – AAMPS, that is the complete algorithm for autonomous robot animation to capture signal meaning, the audience preferences and stability. We explain how AAMPS makes use of the proposed solution for the five core challenges and discuss how AAMPS is used.

Chapter 3

Representation of Robot Motions and Input Signals

This chapter formally defines robot motions and input signals, the inputs to the complete algorithm – AAMPS. We provide an overview of the instantiated robot motions and input signals we use throughout the thesis to illustrate the algorithms and approaches we propose to solve the problem. This chapter does not explain any algorithms or approaches to solve the problem; algorithms and approaches are detailed in the rest of the thesis. The symbols used in this chapter are also used in the rest of the thesis. Appendix A contains a summary of the list of symbols used.

3.1 Representation of Robot Motions

In this section, we formally define motions for robots, starting from the definition of a keyframe, which is a building block to form a robot motion. Next, we explain how a series of keyframes forms a motion primitive, and how the parameter to a motion primitive adjusts its duration.

Following that, we formalize the labels of motion primitives and describe motion primitive categories, which are useful in the selection of relevant motions based on the input signal. Categories also consist features of the motion primitives. These features are introduced as part of the model of the preferences of the audience.

Lastly, we explain how our representation of robot motions reduces the number of expressive motion primitives defined and yet creates many interesting variations of robot motions.

We first formally define a robot that is used to animate the input signal.

Definition 3.1.1. *A robot R has a series of D actuated joints or degrees of freedom with corresponding joint limits and velocities $\{(\theta_1^{min}, \theta_1^{max}, \dot{\theta}_1^{max}), \ldots, (\theta_D^{min}, \theta_D^{max}, \dot{\theta}_D^{max}\}$. The joint index is $d \in \{1, \ldots, D\}$, the minimum and maximum angle of the joint is θ_d^{min}, θ_d^{max} and the maximum velocity is $\dot{\theta}_d^{max}$. Let ζ be the D-dimensional configuration space of R.*

3.1.1 Keyframes

A keyframe (static pose) stores the joints and corresponding angles at a particular time step. For a robot to perform a motion, several keyframes are stored at different time steps and interpolated to form a continuous motion.

Definition 3.1.2. *A keyframe $k \in \zeta$ is a vector of D real numbers for each joint angle of of R. A keyframe $k \in \zeta$ is valid if it is collision-free (the robot has no self-collisions) and the joint angles stay within joint angular limits.*

While having keyframes with clearly defined joint angles enables motion designers or users to know the exact pose of a motion, it does not allow flexibility in defining motions that have different starting positions. For example, Fig. 3.1 shows a motion of nodding the head at different yaw angles with the same pitch angle changes. If keyframes with clearly defined joint angles are used, all combinations of different yaw angles have to be defined. This problem is solved by defining relative changes for certain joints to the previous keyframe. We define keyframes with clearly defined joint angles as fixed keyframes and keyframes with relative changes for certain joints as variable keyframes.

Definition 3.1.3. *A keyframe with fixed joint angles is $k^f = \{(J_1, \theta_1), \ldots, (J_n, \theta_n)\}$, $J_i \neq J_j$ and $n \leq D$. The joint index is J_d and θ_d is the corresponding joint angle. Let the set of keyframes with fixed joint angles be $K^f = \bigcup k^f$.*

We formally define a variable keyframe with the parameter, α, that changes the amplitude of the relative changes for certain joints of R. A relative change in joint index J_d is denoted as $\tilde{\theta}_d$.

Definition 3.1.4. *A variable keyframe, $k^v(\alpha) = \{(J_1, \tilde{\theta}_1^{min}, \tilde{\theta}_1^{max}), \ldots, (J_n, \tilde{\theta}_n^{min}, \tilde{\theta}_n^{max})\}$ where $\tilde{\theta}_d^{min}$ and $\tilde{\theta}_d^{max}$ contains the minimum and maximum relative change for the joint with index J_d, $J_i \neq J_j$ and $n \leq D$. Let $K^v = \bigcup k^v$ be the set of all variable keyframes.*

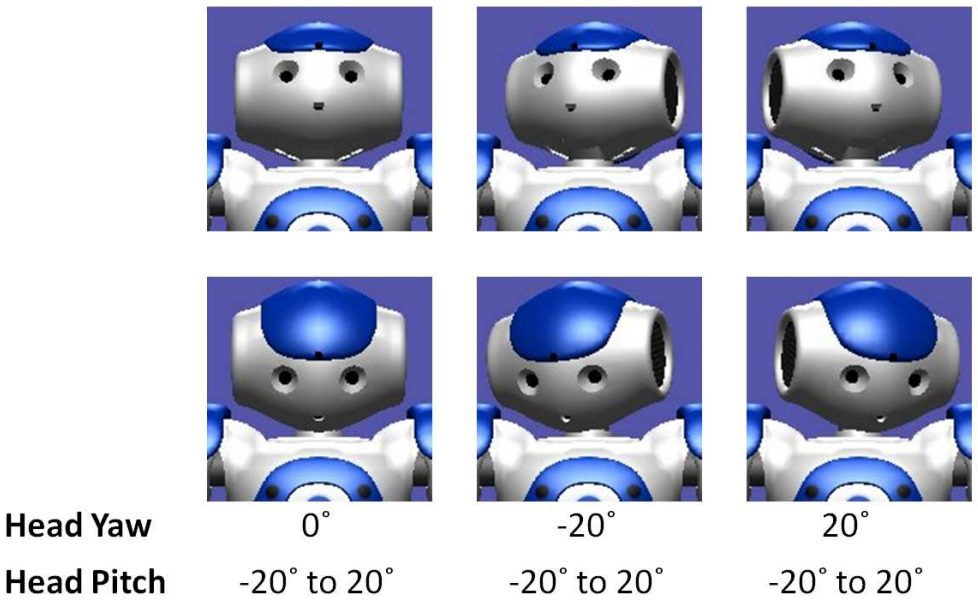


Figure 3.1: Head nods - different yaw angles, same pitch angle changes [Tay and Veloso, 2012].

To determine the joint angle in k^v for joint index J_d, the parameter $\alpha \in [0, 1]$ is used to determine the amplitude of the relative change of J_d, i.e., $\tilde{\theta}_d = \alpha \cdot (\tilde{\theta}_d^{max} - \tilde{\theta}_d^{min}) + \tilde{\theta}_d^{min}$. Hence, with α, a variable keyframe k^v becomes a clearly defined keyframe k^f, so k^v is a parameterized form of a keyframe.

Therefore, we have 2 different types of keyframes:

Definition 3.1.5. *Let $K = K^f \cup K^v$ be the entire set of keyframes (fixed and variable).*

3.1.2 Motion Primitives (MPs)

Motions are movements that convey meanings when synchronized to an input signal. To execute motions on a humanoid robot, the joints of the robot are actuated. A robot is only capable of actuating its joints within the angular joint limits and speeds. A motion is made up of several instantiated *motion primitives*, which we define below. A motion primitive m is a general motion primitive m^g or a spatially targeted motion primitive m^{st}.

General Motion Primitive

A general motion primitive uses only fixed keyframes and does not use any variable keyframes. A general motion primitive is parameterized to allow the motion to be synchronized with the task.

Definition 3.1.6. *A **general motion primitive** m^g is a tuple of $\mathtt{G}$ primitives – $\mathcal{M}^g$, and parameterized with β and N, i.e., $m^g(\beta, N) = (\mathcal{M}^g_1, \ldots, \mathcal{M}^g_{\mathtt{G}})^N$ and $\mathtt{G} \in \mathbb{Z}^+$ and $N \in \mathbb{Z}^+$.*

The primitive $\mathcal{M}^g_n$ is a tuple of 2 keyframes, k_{n-1} and k_n, and the time to interpolate between these two keyframes, $t_{n-1,n}$, where $\mathcal{M}^g_n = (k_{n-1}, \beta t_{n-1,n}, k_n)$. k_0, the first keyframe in $\mathcal{M}^g_1$, is the initial pose of the robot R, which contains the joint angles for the D joints. Let $M^g = \bigcup m^g$ be the set of all general motion primitives.

The motion primitive is parameterized with β, where $\beta \in \mathbb{R}$ and $\beta \geq 1$. β is determined by the duration required to complete the motion primitive based on factors such as the duration of the word, and is used as a multiplying factor. As some motions are repeated, such as waving from side to side for a few times, the parameter N indicates the number of times the general motion primitive is repeated. When $N > 1$, to repeat the motion primitive, the last keyframe k_n interpolates to the first keyframe k_0. When the last iteration of the motion primitive is executed, the final pose of the robot will be k_n.

The interpolation method to interpolate between pairs of keyframe is defined. In this thesis, we use the linear interpolation method. There are other interpolation methods such as bezier interpolation. We assume the motions generated by a motion planner fulfill the following conditions:

1. are collision-free;
2. are within physical limits, e.g., joint angular and velocity limits.

The time to interpolate between two keyframes, k_n and k_{n+1}, is determined by the interpolation time computation function $\mathbb{T} : \zeta \times \zeta \rightarrow \mathbb{R}^+$, i.e., $t_{n,n+1} = \mathbb{T}(k_n, k_{n+1})$. $t_{n,n+1}$ specifies the minimum duration required to interpolate from the joint angles in k_n to the respective joint angles defined in k_{n+1}. The minimum duration depends on the interpolation method defined and is calculated using the maximum joint angular velocities. $t_{n,n+1}$ can also be pre-defined by the motion choreographer. However, if $t_{n,n+1}$ is shorter than the minimum duration required to interpolate from one keyframe to another (i.e., the duration specified by the choreographer is too short for the robot to feasibly execute), then the minimum duration is used for $t_{n,n+1}$.

Spatially Targeted Motion primitive (STM)

Many motions are directed at a point of interest or target. To our knowledge, existing formalizations of motions do not automatically direct the motions at a target based on the parameters of the motion. Therefore, we formally define another type of motion primitive, spatially targeted motion primitive (STM), m^{st}, which uses more parameters to define the direction. A STM is directed at a point or a vector in a particular direction, e.g., to look at the clouds in the sky, thus the robot turns its head to look up at a point in space. In the case of facing someone, the target is defined as a vector and not a point, and the robot is orientated towards the person as the robot looks at the face of the person.

Definition 3.1.7. *A **spatially targeted motion primitive** (STM) m^{st} is a tuple of* $\mathtt{S}$ *primitives and parameterized with β, N and $\mathcal{V}$, i.e., $m^{st}(\beta, N, \mathcal{V}) = (\mathcal{M}_1^{st}, \ldots, \mathcal{M}_{\mathtt{S}}^{st})^N$ where $\mathtt{S} \in \mathbb{Z}^+$ and $N \in \mathbb{Z}^+$. $\mathcal{V}$ is a vector defining the direction of the STM's first keyframe.*

$\mathcal{V}$ determines the direction of the body part that is directed at a point of interest or target and is found in the first pose of the robot. $\mathcal{V}$ consists of two ego-centric coordinates, P^s and P^e.

The primitive $\mathcal{M}_u^{st}$ is a tuple of 2 keyframes, k_{u-1}^v and k_u^v, and the time to interpolate between these two keyframes, $t_{u-1,u}$, where $\mathcal{M}_u^{st} = (k_{u-1}^v, \beta t_{u-1,u}, k_u^v)$. The parameter N indicates the number of times the spatially targeted motion primitive is repeated. Let $M^{st} = \bigcup m^{st}$ be the set of spatially targeted motion primitives.

Similar to the repeats of the general motion primitive, when $N > 1$, to repeat the motion primitive, the last keyframe k_n^v interpolates to the first keyframe k_0^v using the time computation function $\mathbb{T}$.

To instantiate a spatially targeted motion, we need the target's pose and the robot's current pose. Figure. 3.2 illustrates examples of adjusting the robot's pose based on the target.

With $\mathcal{V}$, the pose of the robot is calculated so as to execute the STM and the STM is directed at its desired target. We define two parameters – D^{min} and D^{max}, the minimum and maximum distance. These two distances define the range that the STM is able to execute. Proximity studies are useful as a guide to define these 2 parameters, D^{min} and D^{max}. If the robot's position is out of the range, the robot moves within the range so that the STM is executed. The range is also useful for motions that require a certain distance to the target, e.g., shaking hands with someone.

Algorithm 2 determines the robot's global pose using its original global position, $\mathcal{P}_o^R$, and original global orientation, O_o^R, given a known STM, m^{st}, and a target, $\mathcal{T}$, so as to direct the

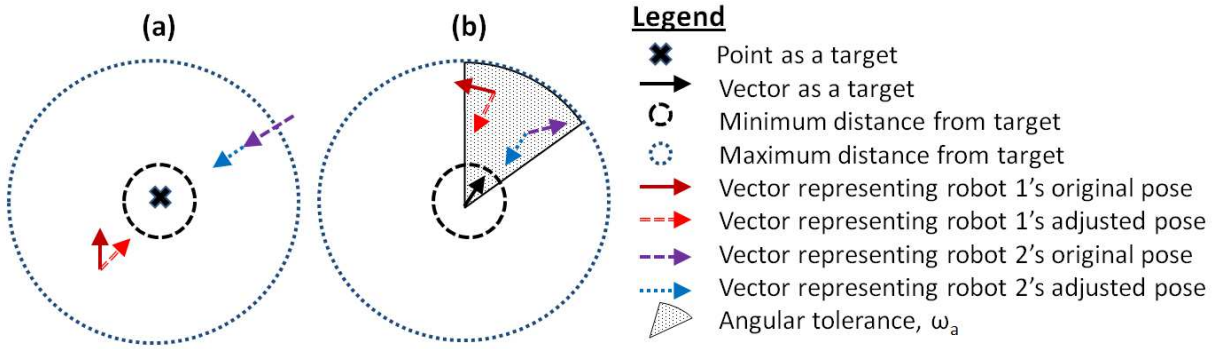


Figure 3.2: Examples of robot's adjusted poses to face a point or vector target [Tay and Veloso, 2012].

robot to face the target. A target is a point, $\mathtt{t}^s$, or a vector, $\overrightarrow{\mathtt{t}^s\mathtt{t}^e}$, defined in global coordinates. ω_a in Fig. 3.2 provides an angular tolerance, where $|O^{st} - O^{\mathcal{T}}| \leq \omega_a$, where O^{st} and $O^{\mathcal{T}}$ are the current and desired STM orientations respectively.

The function convertRelativeToGlobal(P) converts any point relative to the robot to global coordinates. The function canUpdateSTM$(m^{st}, O^{\mathcal{T}})$ performs several checks and updates to determine the final robot position $\mathcal{P}_f^R$ and orientation O_f^R: (a) It checks if m^{st} is updated to face the target at a global orientation angle of $O^{\mathcal{T}}$ and the function canUpdateSTM returns True if it is possible and updates m^{st}, otherwise it returns False and the robot's orientation is updated. (b) Since m^{st} includes variable keyframes, canUpdateSTM$(m^{st}, O^{\mathcal{T}})$ determines if the motion is able to execute with the parameters specified. (c) canUpdateSTM(m^{st}, O^{st}) checks if the joint angular changes are within the joints' angular limits. For example, if the head pitch joint only actuates from $-25°$ to $25°$, and the current head pitch angle is $-20°$ and the variable keyframe contains a relative change of $-10°$, the head pitch joint cannot actuate to $-30°$. Therefore, we update the robot's orientation when we cannot execute the STM.

After determining the global orientation, we check if the position of the robot needs to be changed given the range of the minimum and maximum distance of the motion primitive m^{st} can be executed. If the robot's position has to be updated, the robot will be placed at a distance of $D^{mean} = \frac{D^{min}+D^{max}}{2}$. Algorithm 2 is written for a 2-dimensional space scenario, but the algorithm can be extended to a 3-dimensional space.

Algorithm 2 Determines the final position and orientation of the robot [Tay and Veloso, 2012].

DeterminePoseForSTM($m^{st}, \mathcal{T}, \mathcal{P}_o^R, O_o^R, \omega_a$)

$P^{gs} \leftarrow$ convertRelativeToGlobal(P^s) // P^s is from m^{st}.
$P^{ge} \leftarrow$ convertRelativeToGlobal(P^e) // P^e is from m^{st}.
$O^{st} \leftarrow$ atan2($P^{ge}.y - P^{gs}.y, P^{ge}.x - P^{gs}.x$)
if $\mathcal{T}$ is a point **then**
    $O^{\mathcal{T}} \leftarrow$ atan2($\mathtt{t}^s.y - P^{gs}.y, \mathtt{t}^s.x - P^{gs}.x$)
else if $\mathcal{T}$ is a vector **then**
    $O^{\mathcal{T}} \leftarrow 2\pi -$ atan2($\mathtt{t}^e.y - \mathtt{t}^s.y, \mathtt{t}^e.x - \mathtt{t}^s.x$)
end if
if $|O^{st} - O^{\mathcal{T}}| \leq \omega_a$ **then**
    $O_f^R \leftarrow O_o^R$
else if canUpdateSTM($m^{st}, O^{\mathcal{T}}$) **then**
    $O_f^R \leftarrow O_o^R$
else
    $O_f^R \leftarrow O_o^R + (O^{\mathcal{T}} - O^{st})$
end if
dist $\leftarrow \sqrt{(P^{gs}.x - \mathtt{t}^s.x)^2 + (P^{gs}.y - \mathtt{t}^s.y)^2}$
$D^{mean} \leftarrow \frac{D^{max}+D^{min}}{2}$
if $\mathcal{T}$ is a point **then**
    if dist $\geq D^{min}$ and dist $\leq D^{max}$ **then**
        $\mathcal{P}_f^R \leftarrow \mathcal{P}_o^R$
    else
        $\mathcal{P}_f^R.x \leftarrow (\mathtt{t}^s.x - D^{mean} * \cos(O^{\mathcal{T}})) - P^{gs}.x + \mathcal{P}_o^R.x$
        $\mathcal{P}_f^R.y \leftarrow (\mathtt{t}^s.y - D^{mean} * \sin(O^{\mathcal{T}})) - P^{gs}.y + \mathcal{P}_o^R.y$
    end if
else if $\mathcal{T}$ is a vector **then**
    if dist $\geq D^{min}$ and dist $\leq D^{max}$ and
    |atan2($\mathtt{t}^s.y - P^{gs}.y, \mathtt{t}^s.x - P^{gs}.x$) $- O^{\mathcal{T}}| \leq \omega_a$ **then**
        $\mathcal{P}_f^R \leftarrow \mathcal{P}_o^R$
    else
        $\gamma \leftarrow \frac{\mathtt{t}^e.y - \mathtt{t}^s.y}{\mathtt{t}^e.x - \mathtt{t}^s.x}$
        $\mathcal{P}_f^R.x \leftarrow (\mathtt{t}^s.x - \frac{D^{mean} * D^{mean}}{\gamma^2+1}) - P^{gs}.x + \mathcal{P}_o^R.x$
        $\mathcal{P}_f^R.y \leftarrow (\mathtt{t}^s.y - \gamma\frac{D^{mean} * D^{mean}}{\gamma^2+1}) - P^{gs}.y + \mathcal{P}_o^R.y$
    end if
end if
return $\mathcal{P}_f^R, O_f^R$

The formalization of variable keyframes in spatially targeted motion primitives reduces the number of motion primitives in M that had to be defined for variations of similar motions, where $M = M^g \cup M^{st}$ and M is the set of all motion primitives.

3.1.3 Motion Primitives Categories

Previously, we explained how we formalize motion primitives. In this section, we illustrate with examples how we reduce the number of motion primitives stored by categorizing motion primitives. Motion primitives categories are useful for the following reasons:

- There are many motion primitives stored in the robot's motion library. Organizing motion primitives into categories allows indexing of motion primitives so as to enable a fast search to relevant motion primitives.
- Characteristics of the motion primitives also act as categories and make selection of relevant motion primitives easier. For example, audiences may prefer motion primitives that are faster versus those that are slower.
- Motion primitive categories also reduce repetitive motions to be stored and will be illustrated in Section 3.3.

Definition 3.1.8. *A motion primitive, m, is associated with a set of k motion primitives features $(f_1, \ldots, f_k)$. A motion primitive is assigned to a category, c, due to a particular feature(s), f_i or a set of features, $(f_1, \ldots, f_j)$ of the motion primitive, where $j \leq k$. Let C be the set of all categories and F be the set of all features.*

Definition 3.1.9. *The function $\mathbb{C} : F \times C \rightarrow \{0, 1\}$ determines if the feature f is assigned to a category, c, i.e., $\mathbb{C}(f, c) = 1$ if the feature f is assigned to the category c.*

We assume that all the motion primitives are categorized based on a particular feature(s). In Section 3.3, we describe one of the motion primitive categories we use in this thesis. An example of possible features is the speed of the motion etc.

After formalizing motion primitives and the motion primitive categories, we discuss the pre-processed input signals, including labels in the pre-processed input signal and the labels of motion primitives.

3.2 Representation of Input Signals

To autonomously animate the robot based on an input signal, the input signal is pre-processed to identify the labels (meanings) and the timings of these labels. Motion primitives are selected based on the labels and synchronized to the input signal based on the timings of these labels.

Definition 3.2.1. *A pre-processed input signal $s = (\mathcal{S}_1, \ldots, \mathcal{S}_\mathcal{I})$ is a tuple of $\mathcal{I}$ primitives, where each primitive $\mathcal{S}_i = (l_i^s, t_i^{ss}, t_i^{se})$ is a tuple consisting l_i^s, the label; t_i^{ss}, the starting time of the label l_i^s; and t_i^{se}, the ending time, where $t_i^{se} > t_i^{ss}$. Let $d^{l_i^s} = t_i^{se} - t_i^{ss}$ be the duration of the label l_i^s. Let the set of signals be S.*

Labels

Definition 3.2.2. *A label, l, is assigned to identify meaning. Let L be the set of all labels.*

Labels are useful to identify relevant motion primitives for the input signal. Labels are assigned to motion primitives such that the labels embody the meaning of the motion primitive. A label is mapped to many motion primitives and a motion primitive is mapped to many labels. Labels are used to identify relevant motion primitives based on the similarity between the labels assigned to motion primitives and the labels identified in the signal. We define the labels of the pre-processed signal and labels of motion primitives to explain their relationship.

Labels of the Pre-processed Input Signal

An input signal is pre-processed to identify labels and the times of the occurrences of these labels. The labels of the input signal are used to determine the relevant motions.

Definition 3.2.3. *Let l^s be a label assigned to the signal s. Let the set of labels assigned to the signal s be L^s.*

For example, to identify the semantic meaning of the pre-processed signal, s, of a story, text labels are used.

Labels of Motion Primitives (MPs)

Definition 3.2.4. *A label, l^m, is assigned to a motion primitive, $m \in M$. Let the set of labels assigned to motion primitives be L^M.*

To determine if a mapping exists between a label and a motion primitive, the function $\mathbb{X}$ is used.

Definition 3.2.5. *The function* $\mathbb{X} : M \times L^M \rightarrow \{0, 1\}$ *determines the mapping between the motions and the labels, i.e.,* $\mathbb{X}(m, l) = 1$ *if the motion* $m \in M$ *is mapped to the label* $l \in L^M$.

3.3 Instantiations of Robot Motions and Input Signals

In this section, we present a discussion of the robot motions and the input signals we use to demonstrate our work. First, we consider how motions are created for two domains — music and text. Next, we explain how we use motion primitive categories, specifically using body parts, to create interesting variations of motions. There are different possible categories that we come up with using different features of the motion primitives.

Robot Motions

In this section, we explain how we generate robot motions for the domain of music and the domain of text as input signals.

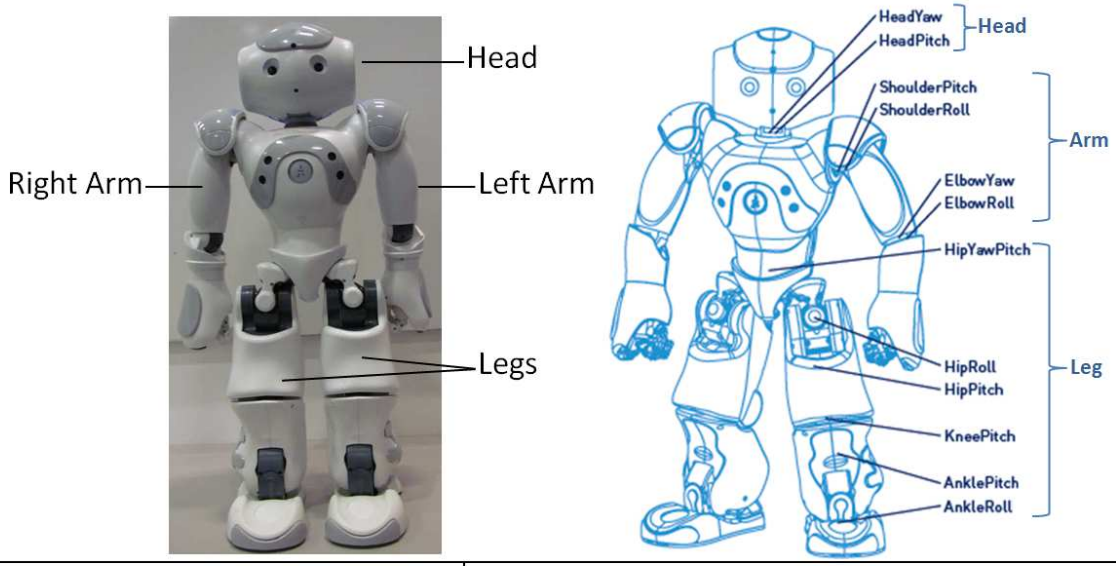
Music

For the domain of music, we wanted to create an interesting variation of motion primitives from a small number of motion primitives defined in the motion library. To attain that goal, we grouped the joints into four categories based on the body parts shown in Figure 3.3 as each body part is actuated independently, assuming that the effects of dynamics caused by the actuation of other body parts are ignored. Moreover, by categorizing the joints, we define motion primitives for each category and create a large number of interesting variations for whole body motions. We do not separate the legs into left leg and right leg categories as the robot loses its balance if the left and right legs are actuated independently.

Joints of the robots are grouped according to the body part category, so the joints of the robots are used as a feature f. A motion primitive is automatically categorized to be in a body part category based on the categories of the joints and whether the joints are actuated in this motion primitive.

Definition 3.3.1. *Let $c^b = (J_1^{c^b}, \ldots, J_k^{c^b})$ be a body part category, where J^{c^b} is the name or index of the joint in the category c^b, and k is the total number of joints in the category c^b.*

We also denote $c^b \in \{\text{Head}, \text{LArm}, \text{RArm}, \text{Legs}\}$. For example, in the body part category "Head", we have Head $=$ (HeadYaw, HeadPitch), where HeadYaw and HeadPitch are the corresponding joint indices of the robot. By categorizing the joints, each keyframe in a motion primitive is associated with one or more categories. Hence, the motion primitive is associated with the union of all the categories the keyframes of the motion primitive are associated with.



Body Part Categories	Joints
Head	HeadYaw, HeadPitch
Left arm (LArm)	LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll
Right arm (RArm)	RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll
Legs	LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll

Figure 3.3: NAO humanoid robot's body parts and joints [Tay and Veloso, 2012].

For the NAO, we group the joints into these 4 categories: Head, LArm (left arm), RArm (right arm) and Legs (Figure 3.3). With these categorizations, we select motion primitives to execute simultaneously and emphasize what the robot is expressing. For example, with a left arm motion primitive shaking the fist angrily and a right arm motion primitive shaking the fist angrily, we combine both motion primitives to emphasize anger. A motion primitive (general or spatially targeted) may be categorized into more than one body part category (Figure 3.4). E.g., a single motion primitive that expresses anger by staring at someone is composed of a head movement and each arm moving to the side of the hips.

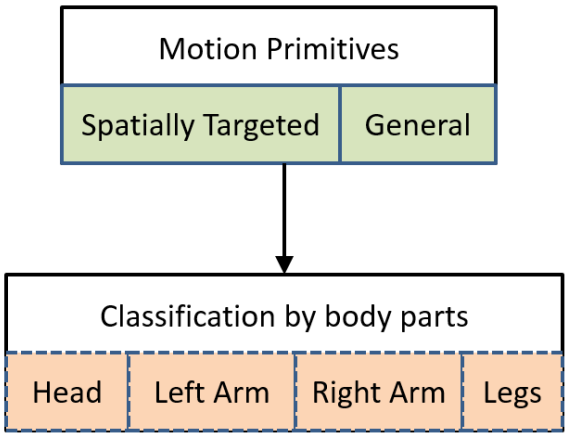


Figure 3.4: Classification of motion primitives [Tay and Veloso, 2012].

To address the goals of automatically generating motions, we define motion primitives as building blocks for a motion, since combinations of motion primitives enable a greater variety of motions. For example, the motion of shaking one's head and waving two hands, indicating no, is made up of three motion primitives and these motion primitives are applicable in other situations as shown in Fig. 3.5.

To generate interesting variations of dances for the domain of music, we manually generated 52 parametrized motion primitives — 8 (Head) $\times$ 9 (LArm) $\times$ 9 (RArm) $\times$ 26 (legs) $= 16,848$ whole body motion combinations. The number of combinations is actually much larger because motions primitives of different categories do not necessarily start and end synchronously.

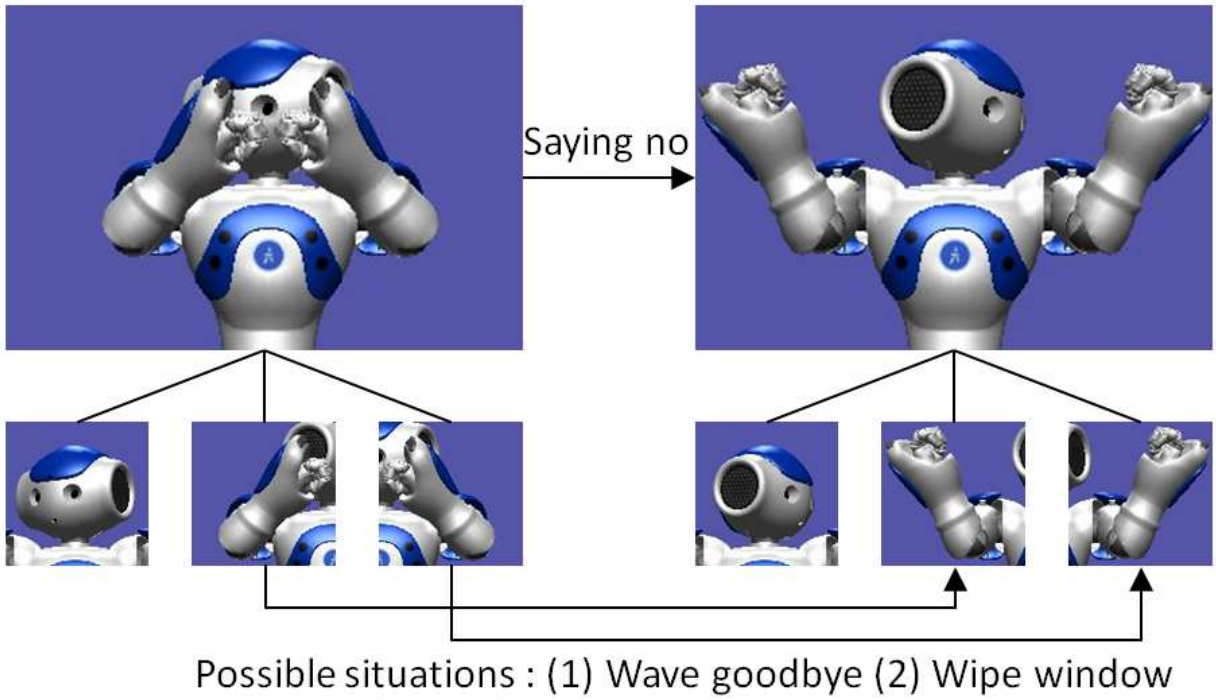


Figure 3.5: Motion composition and possible combinations for other situations [Tay and Veloso, 2012].

Text

For the domain of text, to create a motion library, we collected fifty two words that were taken from a list of words that toddlers should know [Laura Mize, 2008], the Dolch word list, "a list of frequently used English words compiled by Edward William Dolch" [Wikipedia, 2015] and Paul Ekman's six basic emotions as labels.

Following that, we trained a group of students to create motions using the NAO humanoid robot and Choregraphe [Aldebaran Robotics, 2014c], a software to create keyframe motions. The students were instructed to create at least one motion for each label, and were encouraged to create more motions for each label.

Each motion in the motion library is an instantiated motion primitive, where the motion primitive's parameters, such as β and N, are defined. The motions for Paul Ekman's six basic motions were modified from the motions available at `http://hcm-lab.de/projects/shr` [Haring et al., 2011a]. In total, there are 161 motions for the fifty-two words. The list of

words and the number of motions for the list of words are listed in Appendix B.

Each motion primitive in the motion library is instantiated with the parameters $\beta = 1.0$ and $N = 1$ and tested in the simulator, Webots 7 [Webots, 2014] to ensure that the NAO humanoid robot is stable after executing the motion (instantiated motion primitive).

Webots 7 [Webots, 2014] is a real-time simulator that stimulates the dynamics of the NAO humanoid robot. If the motion is unstable, we use Algorithm 3 to determine the shortest duration such that the motion is stable.

We determine the function $\mathbb{T}$ to determine the shortest interpolation time between keyframes. We use the function Simulate to determine the index of the last keyframe the robot is stable.

The robot is determined to be unstable when the robot's body angle exceeds a threshold (robot's body is on the ground) three seconds after the robot has executed the motion. The amount of time to wait to determine the body angle is attained empirically as the robot's body angle reaches an equilibrium after three seconds. The threshold of the body angle is also attained empirically when the robot's body is on the ground.

ε is the amount of time added to the interpolation time between the pair of keyframes to test if the interpolation from one keyframe to the next keyframe will be stable. We use $\varepsilon = 10$ since the shortest time the next keyframe is executed is 10 milliseconds. After determining the shortest time that after executing the pair of keyframes and the robot is stable, we update the interpolation time between the pair of keyframes using the function UpdateInterpolationTime.

After ensuring the stability of each motion, a video of the NAO humanoid robot executing each motion was shown to students and they were asked to provide labels for each motion. Hence, more labels were added, resulting in 161 motions and 319 labels.

We do not evaluate how well the motion(s) expresses the meaning of the label(s) collected. We consider how well the motion expresses the label using the ratings collected from the audience since the audience ratings reflect how well the motion expresses the meaning of a label. We show how we select motion primitives based on the audience preferences in Section 5.3.

Input Signals

In this section, we explain how the input signals are pre-processed to determine the labels and the timings of the labels.

Algorithm 3 Shortest Duration of a Stable Motion.

ShortestDuration(m)

1: $i \leftarrow$ Simulate(m) // Execute motion primitive in simulator, returns the index of the last keyframe that it is stable
2: minStableTime $\leftarrow 0$
3: // $|m|$ is the number of keyframes in m
4: **while** $i \neq |m|$ **do**
5: **if** minStableTime $= 0$ **then**
6: minStableTime $\leftarrow \mathbb{T}(k_i, k_{i+1}) + \varepsilon$
7: **else**
8: minStableTime $\leftarrow$ minStableTime $+ \varepsilon$
9: **end if**
10: $m \leftarrow$ UpdateInterpolationTime(m, k_i, k_{i+1}, minStableTime) // Update the interpolation time between k_i and k_{i+1} to minStableTime
11: $i \leftarrow$ Simulate(m)
12: **end while**
13: **return** m

Music

A piece of music is pre-processed using SMERS [Han et al., 2009], a music emotion recognition system that maps seven features extracted from the music to eleven emotion categories. Each of the eleven emotion categories was assigned a 2-dimensional value using Thayer's 2-dimensional Activation-Valence model [Thayer, 1989]. These activation-valence values act as labels of the signal. As emotions change over time within a piece, we use a 30-second sliding window with a 15-second overlap. Therefore, the i^{th} label represents the emotion of the music from time $15i$ to $15i + 30$ seconds, where $i \geq 0$.

Besides the emotion of the music, we are also interested in having the robot motions follow the beats of the music. We extract the beats of the music using the approach of [Ellis, 2006].

Text

Using the fifty two labels used to create motions for the motion library, we asked a group of students to create stories using at least two labels per sentence and five sentences per story. Twenty stories were written and listed in Appendix C.

To determine the timings of the labels in the stories, we used an open source text-to-speech

engine, Festival [The Centre for Speech Technology Research, The University of Edinburgh, 2010] that provides the start time and end time of each label in the text-to-speech output.

3.4 Chapter Summary

This chapter presented the representation of robot motions, specifically keyframes which are the building blocks of a motion primitive. A motion primitive is parameterized so as to be able to synchronize the motion primitive to the input signal. A motion primitive is a general motion primitive or a spatially targeted motion primitive. We also explained how we reduce the number of motion primitives stored with motion primitives categories using features of the motion primitives.

This chapter explained how the input signal is pre-processed to determine the labels of the input signal. We also discussed the relationship between the labels of the pre-processed input signal and labels of the motion primitives.

After formalizing robot motions and the input signals, we explained how we instantiate these motions and describe the input signals, such as music and speech used in the thesis.

Chapter 4

Mappings between Motions and Labels

This chapter presents two approaches to automatically map motions to labels, since manually labeling new motions in the library becomes tedious when we expand the motion library. The first approach automatically maps new labels to motions based on the features of the motions. The second approach automatically maps existing labels to motions, based on the similarity of the new motion to existing motions in the motion library.

First, we consider how to automatically map motion to labels based on the features of the motions. We use music as the input signal, and describe how we collect emotional poses that are labeled. We contribute an algorithm that automatically labels a motion primitive with an emotional label, given that the motion has similar features to the labeled emotional pose. We use Thayer's 2-dimensional Activation-Valence (AV) model [Thayer, 1989] as the emotional label.

Second, we consider the scenario that features of the motions are unavailable, so we use the existing library of motion primitives which are already mapped to labels. There are three cases:

1. When a new motion primitive is added, mappings to existing labels are to be established.
2. When a new label is added, mappings from existing motion primitives to the new label are to be added.
3. When a new motion primitive and a new label are added, where the new motion primitive is mapped to the new label, mappings between existing motion primitives and the new label, and mappings between the new motion primitive and existing labels are to be generated.

For this thesis, we specifically look at the first case, i.e., when a new motion primitive is added. We explore different metrics to determine the similarity of the motions. We use the

library of pre-defined motions primitives where the motion primitives are already labeled and instantiated. We vary the motions in the motion library by changing the angles and/or duration.

4.1 Mapping Motions to Labels

For the domain of music, we pre-process the input signal, a piece of music, using SMERS [Han et al., 2009], a music recognition system that maps seven features of the music to eleven emotion categories, namely: Anger, Excited, Happy, Nervous, Pleased, Bored, Calm, Relaxed, Sad, Sleepy, Peaceful.

Thayer proposed a two-dimensional Activation-Valence (AV) model that is used to describe emotions on the dimension of activation (also known as arousal, which represents the level of energy) and the dimension of valence (the dimension of stress) [Thayer, 1989]. We assigned activation-valence values to the eleven emotion categories based on the eleven emotion categories plotted on Thayer's two-dimensional emotion model [Han et al., 2009]. Figure 4.1 shows the eleven emotions marked with crosses and their corresponding AV values. For each emotion, there is an activation-valence value (a, v) assigned to the label, where $a \in [-1, 1], v \in [-1, 1]$.

4.1.1 Approach – LeDAV

To assign emotional labels using Thayer's two-dimensional emotion model, we collect static emotional poses in order to create a reference to how emotional labels are mapped to motions. We assume that when the characteristics of these static poses are shown in motions, they express the same emotion as the static pose.

Paul Ekman concluded from his research that there are six basic emotions, namely happy, sad, anger, fear, surprise and disgust [Paul Ekman and Wallace V Friesen, 1975] and claimed that other emotions can be classified into these six emotions [Ekman, 1992]. We label Ekman's six basic emotions with circles and their corresponding AV values which are shown in a grey background in Figure 4.1. SMERS outputs one of eleven emotions, and we show these eleven emotions in Figure 4.1 with crosses. Three of the eleven emotions (Happy, Sad, and Angry) overlap with Paul Ekman's six basic emotions.

Paul Ekman's six basic emotions are clearly separate, discrete emotional states, and we want to determine their characteristics via static poses on a robot. The emotional static poses were

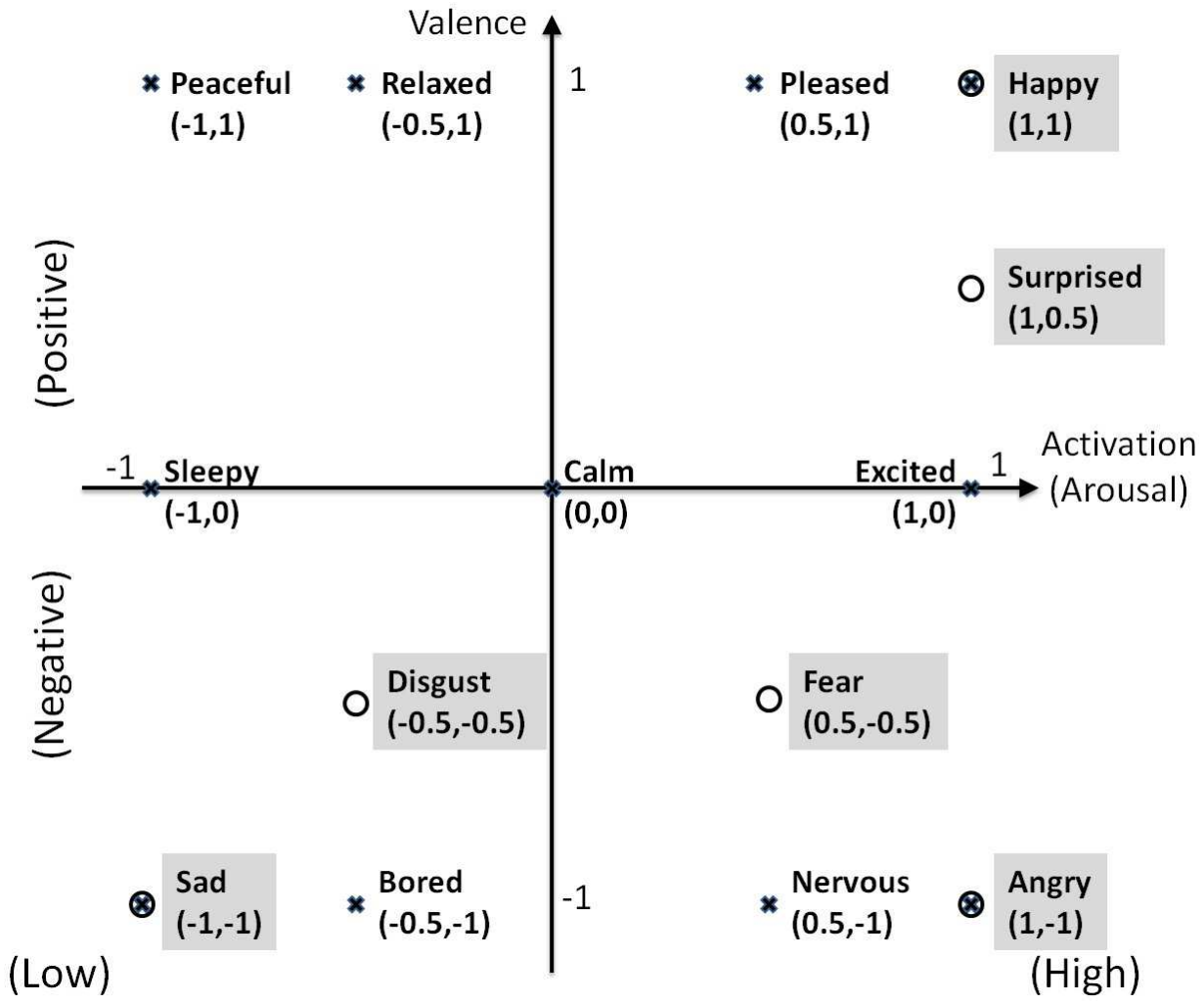


Figure 4.1: Emotions labeled with Thayer's 2-dimensional AV model. [Thayer, 1989]

obtained from 24 girls, aged between 11 to 16, each using the NAO humanoid robot independently. Each girl is asked to express one emotion out of Ekman's six basic emotions using a NAO humanoid robot. They are allowed to freely position the head and arms. As the legs of the robot are difficult to be freely adjusted without the NAO falling over, we do not allow the legs of the robot to be freely adjusted. Instead, we allow the participants to vary the heights and tilts of the robots. Figure 4.2 shows how each participant can vary the legs with five different heights and five different tilts.

In total, we collected a total of 24 static poses, i.e., four static body poses for each of Ekman's

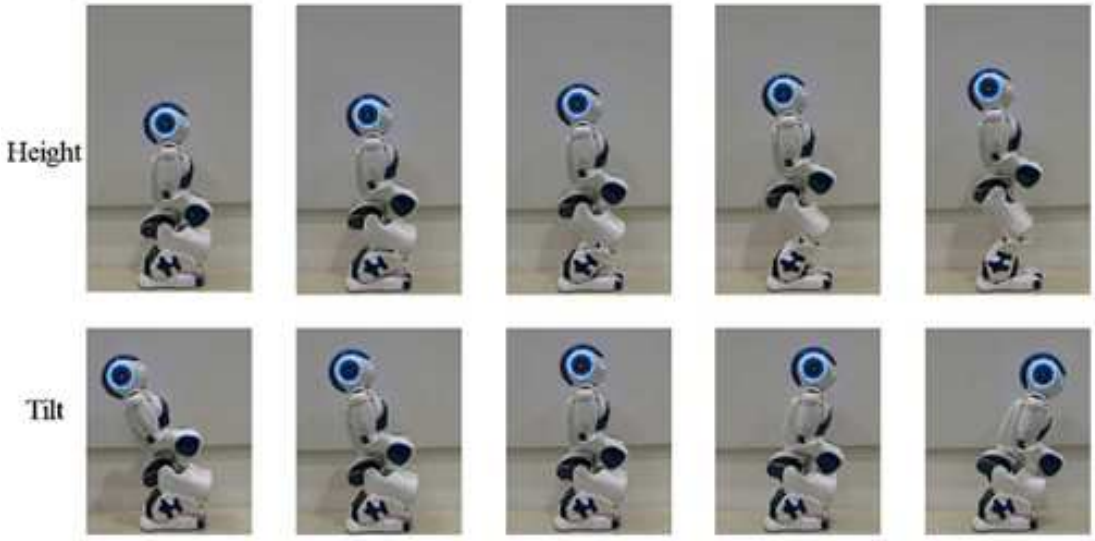


Figure 4.2: 5 heights and 5 tilts of the NAO robot [Xia et al., 2012].

six basic emotions. These 24 static poses are listed with a summary of the height and tilt in Appendix D. Figure 4.3 shows a subset of the static poses data for each emotion. Table 4.1 summarizes our observations of the characteristics of the static poses in terms of height, tilt and arms.

Definition 4.1.1. *Let EM* $= \{$*Happy, Sad, Angry, Disgusted, Fear, Surprised*$\}$ *be the set of Ekman's six emotions, and let em* $\in$ *EM be the one of Paul Ekman's six basic emotions.*

Table 4.1: Summary of the characteristics of the static emotional poses collected.

Emotions	Height	Tilt	Head	Arms
Happy	High	Neutral	Neutral	Raised up above shoulders
Sad	Low	Forward	Forward	Side / In front of eyes
Anger	High	Forward	Neutral	Arms out to the front / At the hip
Surprise	High	Back	Neutral / Back	Arms out to the front / In front of the face
Fear	Low	Forward	Forward	Arms raised and in front of the face
Disgust	-	-	-	-

Using these 24 emotional static postures, we contribute an approach – LeDAV, made up of three algorithms, Algorithms 4 - 6 autonomously assign an AV label to the motion primitive m. Algorithm 4 uses the 3-dimensional positions of the points of interest (POIs) shown as red circles

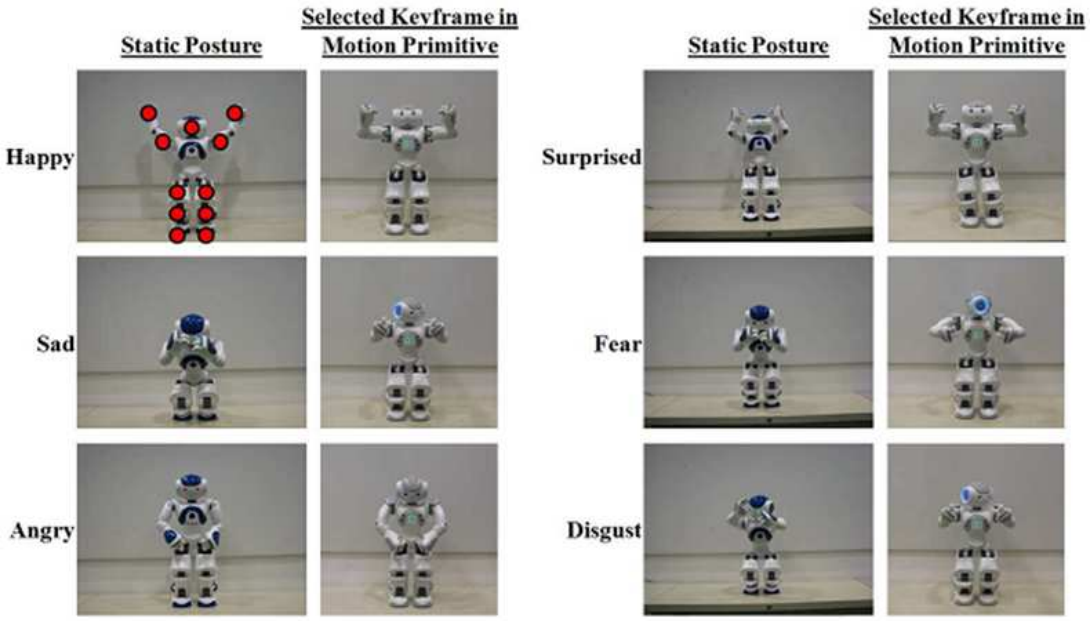


Figure 4.3: Examples of emotional static poses collected and selected keyframes from motion primitives that convey the emotion. Red circles indicate the points of interest (POI) [Xia et al., 2012].

in Figure 4.3. The POIs are used to calculate the least sum of Cartesian distances between each of the four emotional static poses and the keyframes of the motion primitive m. The POIs are placed in the middle of each rigid body link, so a static pose of the robot is reproduced using POIs. Thus, we determine the similarity of each emotional static pose and the motion primitive using the Cartesian positions of the POIs.

Algorithm 5 computes the weights for each of Ekman's six emotions based on the exponential weighting of the rankings and values of the least sum of Cartesian distances from each emotion. Using the least sum of distances returned by Algorithm 4 for each emotion, Algorithm 6 estimates the AV value of the motion primitive m with the weights calculated from Algorithm 5 and the AV values of Ekman's six basic emotions. Figure 4.3 shows examples of keyframes selected from the motion primitives that are assigned the AV value closest to the AV value of the corresponding emotion. The keyframes from the motion primitives labeled with similar emotions look similar to the static poses. Figure 4.4 shows the estimated AV values of the motion primitives in the motion library we created and Ekman's six emotions.

Algorithm 4 Calculate the least sum of Cartesian distances of points of interest of a motion primitive m and the emotional static pose in SP_{em}, the set of four emotional static postures for the emotion em [Xia et al., 2012].

Least**D**iff(m, em)

for sp $\in \text{SP}_{\text{em}}$ **do**
  // Sum the distances between the keyframes in m and the emotional static pose sp
  $\text{DIST}_{\text{sp}} \leftarrow \sum_{\text{kf} \in m} \text{GetDist}(\text{kf}, \text{sp})$
end for
return $\min_{\text{sp} \in \text{SP}_{\text{em}}}(\text{DIST}_{\text{sp}})$

Algorithm 5 Calculate the vector of weights based on the ranking of the Euclidean distances [Xia et al., 2012].

GetWeights(distances)

1: **for** $i = 1$ to $|\text{distances}|$ **do**
2: $\quad \text{flippedDistances}_i \leftarrow (\sum_j \text{distances}_j) - \text{distances}_i$
3: **end for**
4: sorted $\leftarrow$ sortAscending(flippedDistances)
5: meanValue $\leftarrow$ mean(flippedDistances)
6: **for** $i = 1$ to $|\text{flippedDistances}|$ **do**
7: $\quad \text{weights}_i \leftarrow e^k + \frac{\text{flippedDistances}_i}{\text{meanValue}}$ where $\text{sorted}_k = \text{flippedDistances}_i$
8: **end for**
9: **for** $i = 1$ to $|\text{weights}|$ **do**
10: $\quad \text{weights'}_i \leftarrow \frac{\text{weights}_i}{\sum_j \text{weights}_j}$
11: **end for**
12: **return** weights'

4.2 Mapping Existing Labels to New Motions

In this section, we consider adding new motions into an existing labeled motion library, where each motion in the motion library has one or more labels. Without an autonomous way to map labels to new motions, all existing labels have to be examined manually to determine the mappings between labels and the new motion. We contribute an algorithm that autonomously determines mappings between a new motion and existing labels, by finding similar motions and using the labels of the similar motions as the labels for the new motion. We investigate how to associate labels with a new motion by determining effective metrics to compute the similarity between two motions so as to use the labels of the most similar motion.

Algorithm 6 Estimate activation-valence value of m [Xia et al., 2012].

AV(m)
  for em $\in$ EM **do**
    $\text{emDiff}_{\text{em}} \leftarrow \text{LeastDiff}(m, \text{em})$
  end for
  $\text{weights} \leftarrow \text{GetWeights}(\text{emDiff})$
  $\text{act} \leftarrow \sum_{\text{em}\in\text{EM}}(\text{weights}_{\text{em}} \cdot \text{em}_a)$
  $\text{val} \leftarrow \sum_{\text{em}\in\text{EM}}(\text{weights}_{\text{em}} \cdot \text{em}_v)$

A similar motion has generally been defined as having similar joint angles or postures. We explore using joint angles and postures as measures to determine similarity. We also investigate two general distance metrics – Euclidean and Hausdorff distances. We introduce the concept of a mirrored motion, where a motion is symmetrical to another motion, e.g., where a motion involving the left hand or the right hand is mapped to the same label. We incorporate all these approaches into eight distance metrics and compare the efficacy of each metric using precision and recall.

We conduct experiments in Webots, a real-time simulator to determine the efficacy of the eight distance metrics. We explain how we create two motion libraries to compare a motion library with mirrored motions versus a library without. We also generate variants of the motions in each motion library to evaluate the distance metrics. We determine the mappings of existing labels to new motions using the eight distance metrics and the nearest neighbor algorithm. We determine the best distance metric based on the precision, recall and computational complexity.

4.2.1 Motion Library

We have a total of 161 motions in the motion library and we term these as the Initial set. The Initial set contains motions with no modifications and are labeled. 35 out of these 161 motions in Initial are mirrored motions. Since our motion library consists of 161 motions, we expand Initial to 1610 motions by varying the joint angles and/or interpolation times. We assume that by varying the joint angles and/or interpolation times, the same labels are still applicable to these variations.

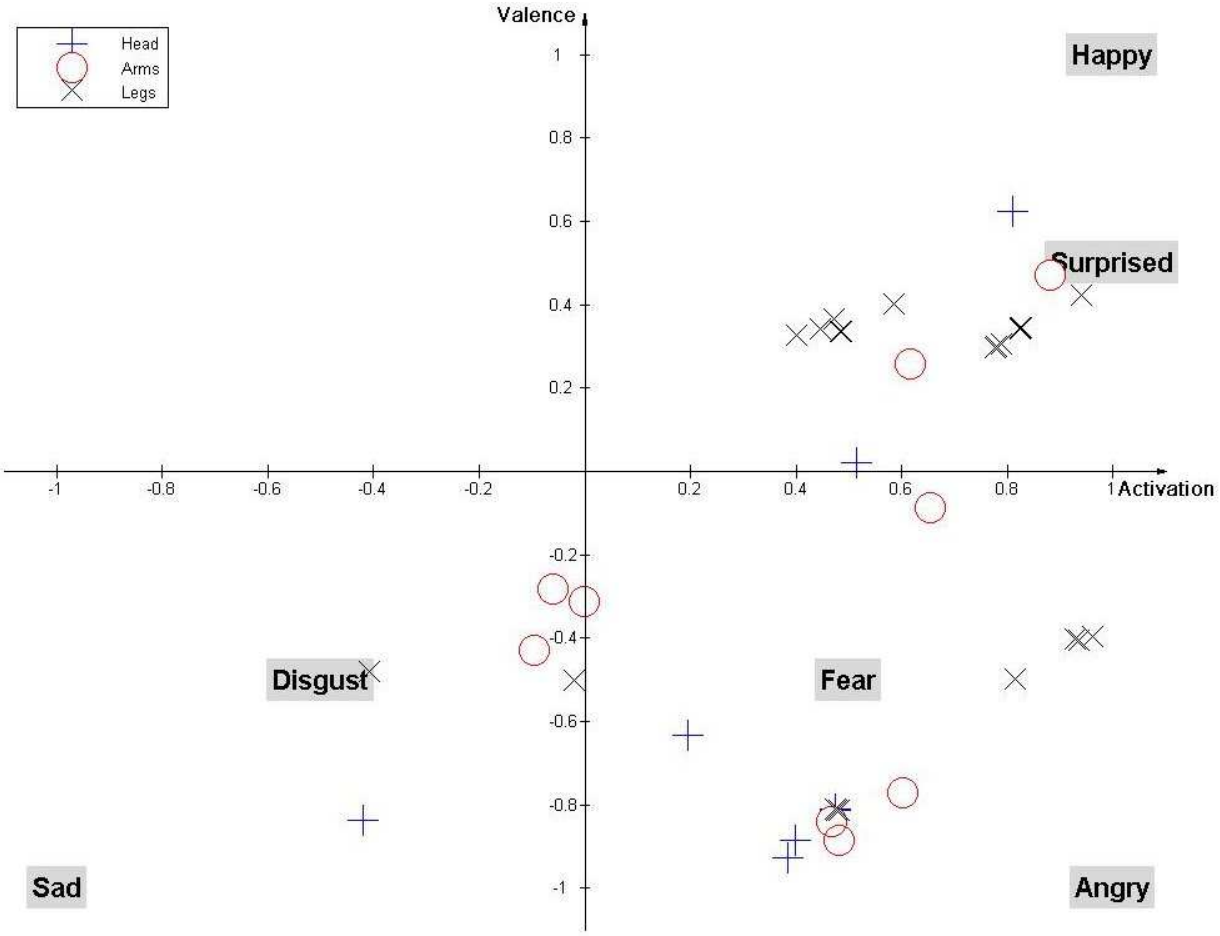


Figure 4.4: Activation-valence values of labeled motion primitives

Varying Interpolation Time and Joint Angles

To create similar motions, we vary the following features of the motions in Initial, and assume each variant of a motion, m_n, shares the same labels assigned to m_n in Initial:

- ModJoints: We only modify the joint angles of each motion where each joint angle for each keyframe in the motion primitive is modified with a 50% probability. If the joint angle is modified, the joint angle will be changed within a range of -5° to 5°, so $\theta_d = \theta_d + \tilde{\theta}$ and $\tilde{\theta} \in \{-5^\circ, -4^\circ, -3^\circ, \ldots, 5^\circ\}$.
- ModTime: We only vary the interpolation times by changing β, $\beta \in \{1.25, 1.5, 1.75, 2\}$.
- ModJointsAndTime: We change both joint angles and interpolation time of each motion

by combining the first and second features. We use the motions that were modified in ModJoints and modify the interpolation time by $\beta \in \{1.25, 1.5, 1.75, 2\}$.

Creating Mirrored Motions

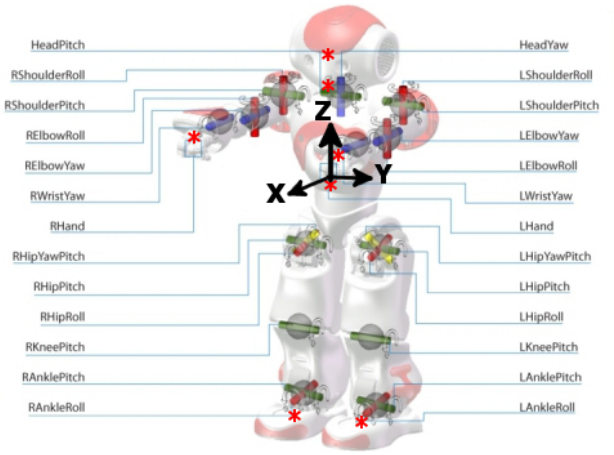


Figure 4.5: Joints, POIs and coordinate frame of the NAO robot. Edited image from [Aldebaran Robotics, 2014b].

The NAO H25 humanoid robot has 25 actuated joints. Figure 4.5 shows the positions of all the joints in the NAO humanoid robot. Although Fig. 4.5 shows a total of 26 joints, only 25 joints are actuated as the LHipYawPitch and RHipYawPitch "share the same motor so they move simultaneously and symmetrically" and in the case of "conflicting orders, LHipYawPitch always takes the priority" [Aldebaran Robotics, 2014a].

Some motions are a mirror image of another motion in the library; we termed them *mirrored motions*. The meanings of a motion are similar when a motion is a mirror image of another motion. For example, the motions of waving with the left hand and waving with the right hand are mapped to the same label – "wave". Also, the motions of kicking with the left leg and kicking with the right leg are also mapped to the label – "kick". However, waving with the left hand is labeled with the phrase "wave with left hand" and waving with the right hand is labeled with the phrase "wave with right hand". Therefore, though they share most of their labels (and meanings), they are also mapped to different labels.

Definition 4.2.1. *Let $\theta_d^{original}$ be the joint angle of joint d in the motion m. Let θ_d^{mirror} be the joint angle of joint d in the mirrored motion m_n.*

Paired Joints		**Corresponding Mirrored Joints**	
HeadYaw	HeadPitch	-HeadYaw	HeadPitch
LShoulderPitch	RShoulderPitch	RShoulderPitch	LShoulderPitch
LShoulderRoll	RShoulderRoll	-RShoulderRoll	-LShoulderRoll
LElbowYaw	RElbowYaw	-RElbowYaw	-LElbowYaw
LElbowRoll	RElbowRoll	-RElbowRoll	-LElbowRoll
LWristYaw	RWristYaw	-RWristYaw	-LWristYaw
LHand	RHand	RHand	LHand
LHipRoll	RHipRoll	-RHipRoll	-LHipRoll
LHipPitch	RHipPitch	RHipPitch	LHipPitch
LKneePitch	RKneePitch	RKneePitch	LKneePitch
LAnklePitch	RAnklePitch	RAnklePitch	LAnklePitch
LAnkleRoll	RAnkleRoll	-RAnkleRoll	-LAnkleRoll
LHipYawPitch		LHipYawPitch	

Table 4.2: Paired joints and corresponding mirrored joints.

We compute a mirrored motion by looking at pairs of joints that are symmetrical to each other by the Z axis in Fig. 4.5 using the function mirror, where $\text{mirror}(\theta_d^{\text{original}}) = \theta_d^{\text{mirror}}$. Table 4.2 shows the list of 25 joints and the corresponding mirrored joints. For example, for the joint angle of joint HeadYaw in the original motion, $\theta_{\text{HeadYaw}}^{\text{original}}$, the joint angle for HeadYaw in the mirrored motion will be negative. Therefore, in order to find $\theta_{\text{HeadYaw}}^{\text{mirror}}$, we use the function mirror, i.e., $\theta_{\text{HeadYaw}}^{\text{mirror}} = \text{mirror}(\theta_{\text{HeadYaw}}^{\text{original}}) = -\theta_{\text{HeadYaw}}^{\text{original}}$.

4.2.2 Metrics for Motion Similarities

Besides joint angles, we also consider the differences in the three-dimensional positions of the joints with respect to the robot's torso as the joint differences may not reflect the differences in posture. Hence, we compute the three-dimensional (3D) position of each joint of the robot and term each position as a point of interest (POI). Besides each joint, Fig. 4.5 shows the points of interest (POIs) with seven red asterisks (*). We add these seven POIs because their 3D positions vary with joint angles changes in the head, wrists and ankle joints, whereas the 3D positions of the head, wrists and ankle joints are invariant to joint angle changes. E.g., the 3D position of the HeadYaw joint remains unchanged when HeadYaw's joint angle changes. Therefore, there are $|\text{POI}| = 25 + 7 = 32$ POIs. Besides using Euclidean distance, Erdogan and Veloso also chose

"the Hausdorff metric for its generality and efficiency" [Erdogan and Veloso, 2011a]. Eight distance metrics are varied across three axes – Euclidean versus Hausdorff, mirrored versus non mirrored and joint angles versus POIs:

1. EuclideanJoint: We compute the average absolute joint difference between the same joint for two different motions for each time step. If a motion m_1 is longer in duration than the other motion m_2, we use the joint angles at the last time step of m_2 to compare with the rest of the joint angles of m_1, and vice versa. Let the duration of m_1 be dt_1 and the duration of m_2 be dt_2. Since D is the number of the degrees of the freedom of the robot R and we are using the NAO humanoid robot with 25 joints, $D = 25$. Let $\theta_s^{(m_i,d)}$ be the joint angle of joint d at time step s of the motion i. We determine the average joint difference:

$$\text{EuclideanJoint}(m_1, m_2) = \frac{\sum_{d=1}^{D} \sum_{s=1}^{\max(\text{dt}_1,\text{dt}_2)} |\theta_s^{(m_1,d)} - \theta_s^{(m_2,d)}|}{\max(\text{dt}_1, \text{dt}_2)}$$

2. EuclideanMirrorJoint: We compute the average absolute joint difference in joint angles for motion m_1 and a mirrored motion of another motion m_2 using the function mirror which calls $\text{mirror}(\theta_s^{(m_1,d)})$ on each joint angle $\theta_s^{(m_1,d)}$ of m_1 in each timestep. We also compute the average absolute joint difference for motions m_1 and m_2 and use the smaller difference:

$$\begin{aligned}\text{EuclideanMirrorJoint}(m_1, m_2) = \min(&\text{EuclideanJoint}(m_1, m_2),\\ &\text{EuclideanJoint}(m_1, \text{mirror}(m_2)))\end{aligned}$$

3. EuclideanPOI: We compute the average absolute Euclidean distance of the 3D position of the same POI for two different motions for each time step. We also repeat the computations for each of the 32 POIs. If a motion m_1 is longer in duration than the other motion m_2, we use the 3D position of the POI at the last time step of m_2 to compare with the rest of the 3D position of the same POI of m_1, and vice versa. Let the duration of m_1 be dt_1 and the duration of m_2 be dt_2. Let $\text{POI}_s^{(m_i,p)}$ be the p^{th} POI in the i^{th} motion at time step s. We determine the average Euclidean POI difference:

$$\text{EuclideanPOI}(m_1, m_2) = \frac{\sum_{p=1}^{|\text{POI}|} \sum_{s=1}^{\max(\text{dt}_1,\text{dt}_2)} |\text{POI}_s^{(m_1,p)} - \text{POI}_s^{(m_2,p)}|}{\max(\text{dt}_1, \text{dt}_2)}$$

4. EuclideanMirrorPOI: We compute the average absolute Euclidean distance of the three dimensional position of the same POI for two different motions for each time step. We also compute the average absolute Euclidean distance of the first motion m_1 to the mirrored motion of the second motion m_2 and take the minimum:

$$\text{EuclideanMirrorPOI}(m_1, m_2) = \min(\text{EuclideanPOI}(m_1, m_2), \\ \text{EuclideanPOI}(m_1, \text{mirror}(m_2)))$$

5. HausdorffJoint: Instead of determining Euclidean distances between joints or POIs, we use the Hausdorff metric where the function ED computes the Euclidean distance between two joints using the joint angles, i.e., $\text{ED}(\theta^{m_1}, \theta^{m_2})$:

$$\text{HausdorffJoint}(m_1, m_2) = \max(\max_{\theta^{m_1} \in m_1} \min_{\theta^{m_2} \in m_2} \text{ED}(\theta^{m_1}, \theta^{m_2}), \\ \max_{\theta^{m_2} \in m_2} \min_{\theta^{m_1} \in m_1} \text{ED}(\theta^{m_1}, \theta^{m_2}))$$

6. HausdorffMirrorJoint: We use HausdorffJoint to find the minimum of the two Hausdorff measures – joint angles for m_1 and m_2 and joint angles for m_1 and $\text{mirror}(m_2)$:

$$\text{HausdorffMirrorJoint}(m_1, m_2) = \min(\text{HausdorffJoint}(m_1, m_2), \\ \text{HausdorffJoint}(m_1, \text{mirror}(m_2)))$$

7. HausdorffPOI: Instead of joint angles, we look at Hausdorff measures for POIs and the function $\text{EP}(\text{POI}^{m_1}, \text{POI}^{m_2})$ returns the Euclidean distance between the two POIs of m_1 and m_2:

$$\text{Hausdorff}_{\text{POI}}(m_1, m_2) = \max(\max_{\text{POI}^{m_1} \in m_1} \min_{\text{POI}^{m_2} \in m_2} \text{EP}(\text{POI}^{m_1}, \text{POI}^{m_2}), \\ \max_{\text{POI}^{m_2} \in m_2} \min_{\text{POI}^{m_1} \in m_1} \text{EP}(\text{POI}^{m_1}, \text{POI}^{m_2}))$$

8. HausdorffMirrorPOI: We use HausdorffPOI to compute the minimum of the two Hausdorff measures – POIs for motion m_1 and m_2 and POIs for m_1 and mirror(m_2):

$$\text{HausdorffMirrorPOI}(m_1, m_2) = \min(\text{HausdorffPOI}(m_1, m_2), \text{HausdorffPOI}(m_1, \text{mirror}(m_2)))$$

Adding a new motion to the motion library

We use the nearest neighbor algorithm to select the closest motion to the new motion using the output of a distance metric and map its labels to the new motion:

First, given a new motion m^+, and the existing motion library M, using DM(m^+, m), where DM is one of the metrics described earlier. For example, using one of the distance metrics, i.e., DM = EuclideanJoint, we find:

$$m^* = \text{argmin}_{m \in M} \text{DM}(m^+, m)$$

Second, we create an updated motion library $M^+ = M \cup \{m^+\}$. Third, a new motion m^+ is mapped to m^*'s labels and use the updated mapping function $\mathbb{X}^+$:

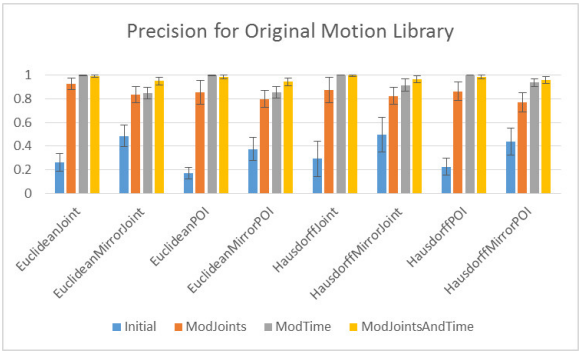
$$\mathbb{X}^+(m, l) = \begin{cases} \mathbb{X}(m, l) & \text{if } m^+ \neq m \\ \mathbb{X}(m^*, l) & \text{otherwise} \end{cases}$$

Thus, the new motion and its corresponding labels are represented in the updated motion library M^+ and the updated mapping function $\mathbb{X}^+$.

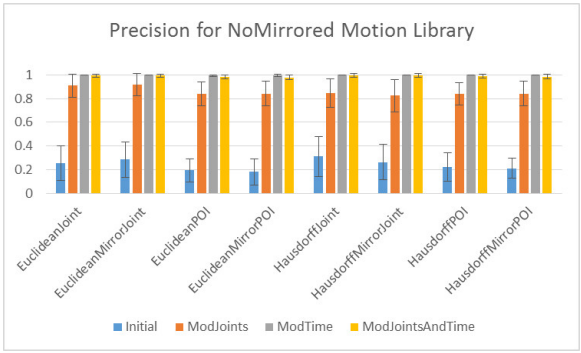
4.2.3 Experiments

In this section, we describe our experiments to evaluate the eight distance metrics and the nearest neighbor algorithm that autonomously maps motions to labels.

We compared the different distance metrics to determine similarities of motion trajectories. We used the motions from an existing motion library used by a NAO humanoid robot to animate stories – Original – and created another motion library – NoMirrored – by removing mirrored



(a) Precision for Original Motions



(b) Precision for NoMirrored Motions

Figure 4.6: Precision for 2 motion libraries.

motions from Original. These two motion libraries enable us to understand the efficacy of including the function mirror in the distance metrics:

- Original: Original has a set of 161 motions and 319 associated labels.
- NoMirrored: NoMirrored has a set of 126 motions and 265 associated labels.

Next, for each of the 2 motion libraries: Original and NoMirrored, we create the variants described in Section 4.2.1: Initial, ModJoints, ModTime and ModJointsAndTime.

To evaluate the eight distance metrics, we use Precision $= \frac{\text{true positives}}{(\text{true positives}+\text{false positives})}$ and Recall $= \frac{\text{true positives}}{(\text{true positives}+\text{false negatives})}$ to measure the performance of assigning labels to motions. The term positive means the motion is assigned a label and negative means the motion is not assigned a label. The term true means the label assigned is right and false means the label assigned is wrong. We term true positives TP, false positives FP, false negatives FN and each term is indexed by v – the index of the label. The equation to compute the precision is $\sum_{v=1}^{|L|} \text{TP}_v / (\sum_{v=1}^{|L|} \text{TP}_v + \sum_{v=1}^{|L|} \text{FP}_v)$ and the equation for recall is $\sum_{v=1}^{|L|} \text{TP}_v / (\sum_{v=1}^{|L|} \text{TP}_v + \sum_{v=1}^{|L|} \text{FN}_v)$, where $|L|$ is the number of labels in the library.

We perform 10-fold cross validation, where the motions are randomly divided into 10 folds and we iteratively use 1 fold as test data and the rest as training data. Next, we determine the labels of each motion in the test data using a distance metric and the nearest neighbor algorithm. We perform the cross validation 10 times for each distance metric, find the precision and recall for each variant of motions in each motion library and summarize the results with a mean and standard deviation in Figure 4.6 and Figure 4.7.

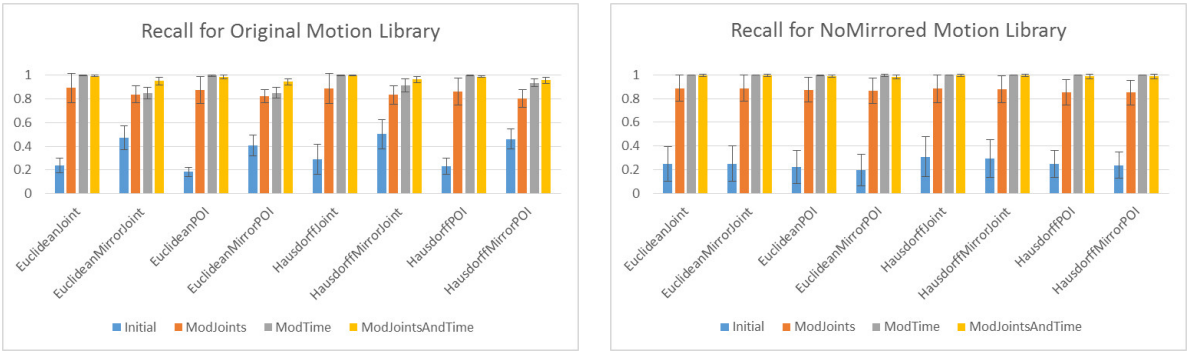
The precision and recall for the Initial motions are low as compared to other variants of

motions, e.g., ModTime, which is expected as the motions in Initial do not have many similar motions and are mostly distinct except for mirrored motions. Hence, the nearest neighbor algorithm is unable to find an exact match of the labels for a new motion.

The precision and recall for the Initial motions improve for the distance metrics that include the function mirror. In contrast, when the library is expanded with ModTime for example, the nearest neighbor algorithm returns a similar motion with the exact labels. Hence, the precision and recall is much higher.

For the Original motion library, distance metrics that include the function mirror perform worse than metrics that do not include the function, except for the Initial motions. Since distance metrics that include the function mirror compute the similarity between a motion m_1 and the mirrored motion of m_2, m_1 is treated as the mirrored of m_2 given that the distance metric returns the lowest value. Therefore, mapping the wrong labels to the new motion often occurs, resulting in a lower precision and recall. Also, though most labels of the mirrored motions are the same as the labels of the Initial motions, some of them are different as they include the word, "right" instead of "left" or vice versa. By removing mirrored motions in the NoMirrored motion library, the precision and recall for the NoMirrored motion library are similar for metrics without the function mirror. This finding supports our explanation of why distance metrics that include the function mirror perform worse for the Original motion library than the NoMirrored motion library.

The distance metrics that involve Euclidean distances perform as well as the distance metrics that involve Hausdorff distances in terms of precision and recall. However, Hausdorff distances are computationally more expensive and runs in $O(t^2)$, whereas Euclidean distances run in $O(t)$, where t is the number of time steps of the longer motion. The distance metrics that involve the joints perform as well as distance metrics that involve the POIs. However, distance metrics that involve the POIs use more computations (absolute difference between a pair of 3D points) than the distance metrics that involve the joints as we take the absolute difference between each pair of joint angles. Hence, EuclideanJoint is the best distance metric for motions such as ModJoints, ModTime and ModJointsAndTime in terms of precision, recall and computational complexity.



(a) Recall for Original Motions

(b) Recall for NoMirrored Motions

Figure 4.7: Recall for 2 motion libraries.

Analysis of Results

We define eight distance metrics to determine the similarities of motions. We create two motion libraries and explain how we created variants of the motions to conduct experiments. We determine mappings of existing labels to new motions using the eight distance metrics and the nearest neighbor algorithm. We present the efficacy of each distance metric using precision and recall. We find that EuclideanJoint is the best distance metric in terms of precision, recall and computational complexity.

We observe that distance metrics with the mirror function have a lower precision and recall. Precision and recall can be increased by looking through the labels, and replacing the words associated with the mirrored motion, e.g., changing “left” to “right”, but this approach requires a dictionary of such pairs of words.

4.3 Chapter Summary

This chapter presents LeDAV, that uses the weighted similarity between the emotional static poses collected and the motion to be labeled. LeDAV assigns an emotional label with an activation-valence value based on the computed weights. LeDAV autonomously maps motions to labels based on the features of the motions using music as an input signal.

This chapter also explains how there may be cases where the features of the motions are not available to automatically map new motions to labels. This chapter explores different distance

metrics to determine the similarity between motions so that labels of existing motions are used for a new motion.

Chapter 5

Selection and Synchronization of Motion Primitives

In this chapter, we consider how to autonomously select relevant motion primitives and synchronize the motion primitives to the input signal. Selection of relevant motion primitives is not a simple task of just selecting the motion primitives based on a match between labels. There are two approaches that we use to select relevant motion primitives. First, we investigate probabilistically selecting relevant motion primitives as we step through each label of the input signal. The second approach is to consider all possible combinations of the sequences of motion primitives, use a set of weighted criteria, rank the sequences, and choose the best sequence.

For each approach, we explain how we synchronize the motion primitives to the input signal. We use two domains to illustrate the two different approaches. First, in Section 5.1, we consider the domain of music, where we probabilistically select relevant motion primitives based on the emotions of the music and synchronize the motion primitives to the beats of the music. Next, in Section 5.2, using the domain of text, we explain how we select relevant motions and generate synchronized motion sequences that are valid. We rank the motion sequences based on a set of weighted criteria and select the best sequence to execute.

Lstly, in Section 5.3, we explore how to use the audience feedback of previous motion sequences to improve the selection of motions and determine the most preferred sequence. The audience provides feedback at the end of a performance, i.e., a numerical rating is given at the end of a motion sequence. We explain how we model the ratings of the individual motions using

the feedback of the audience. We also discuss how we model the effects of boredom when the audience repeatedly views the same motion.

5.1 Probabilistic Selection and Synchronization

Dancing motions for robots are usually created by choreographers and designed for a particular piece of music. If the piece of music changes, the dance movements of the robot will have to be recreated. We are interested in automating the task of robot dance choreography by generating sequences of dance movements from a motion library. The automatically generated choreography should satisfy several goals. First, the choreography should reflect the emotional character of the music. Peaceful music should be choreographed differently from music that sounds angry. Second, the dance should be synchronized to the music. Lastly, the dance should not be deterministic. Even when the emotion and tempo of the music remain constant, the dance should contain interesting variations.

We represent emotion using a two-dimensional activation-valence emotion space, which is commonly used to describe emotional states. We generate many variations of motion primitives, by dividing the joints of the NAO humanoid robot into 4 body part categories, where each category of joints is actuated independently. We describe how we generate many variations of motions in Section 3.3. Motion primitives from each body part category are selected to match the emotional state of the music.

To synchronize a dance to the music, we adjust the duration of each motion primitive so that the duration will be an integer multiple of beats. To create interesting variations in the dance, we use a first-order Markov model to generate dances stochastically. States correspond to motion primitives. The state transition probabilities are designed to produce smooth motion sequences by favoring next states that begin with a keyframe near the final keyframe of the current state. The state transition probabilities also depend upon the current emotion in the music, such that at any given time, state transition probabilities will prefer states that reflect the current emotion in the music. Figure 5.1 summarizes the process we described.

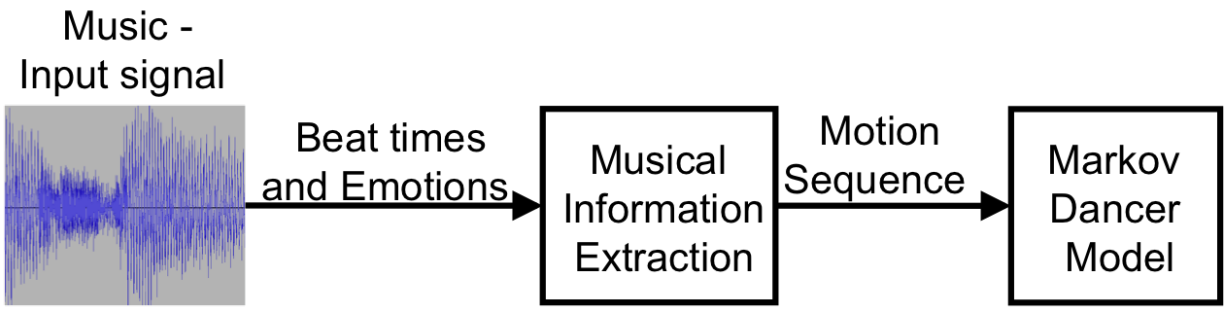


Figure 5.1: Overview of probabilistic selection and synchronization.

5.1.1 Approach – CEN

Previously, in Section 3.3, we discussed body part categories. We group the joints into 4 categories:

1. Head (*Head*): HeadYaw, HeadPitch;
2. Left Arm (*LArm*): LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll;
3. Right Arm (*RArm*): RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll;
4. Legs (*Legs*): LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll

Each body part category is defined to be $c^b = (J_1^{c^b}, \ldots, J_k^{c^b})$, where J^{c^b} is the name or index of the joint in the category c^b and $c^b \in \{\text{Head}, \text{LArm}, \text{RArm}, \text{Legs}\}$. k is the total number of joints in the category c^b. For example, $\textit{Head} = (1, 2)$ where 1 is the index of HeadYaw and 2 is the index of HeadPitch. We drop the superscript b for notation simplicity for the body part category.

We have a labeled library of motion primitives that are categorized by body parts. We describe how we label these motion primitives in Section 4.1. We have 52 parametrized motion primitives categorized by body parts, 8 for the head, 9 for the left arm, 9 for the right arm and 26 for the legs. Although the library of the motion primitive seems small, when we combine the motion primitives into a full body motion, we generate $8 \times 9 \times 9 \times 26 = 16,848$ whole

body combinations. There are even more whole body combinations especially since the motion primitive from each category does not have to start or end at the same time.

We aim to generate a sequence of motions that mimick a dancer who strives to reflect the emotion, dance to the beats of the music and achieve continuity of motions. Continuity of motions means that there is no jerk in the motions and that the motions are fluid and continuous. We model this problem as a Markov chain, which is a generative stochastic motion model. A separate model (Figure 5.2) is used for each category, e.g., Head.

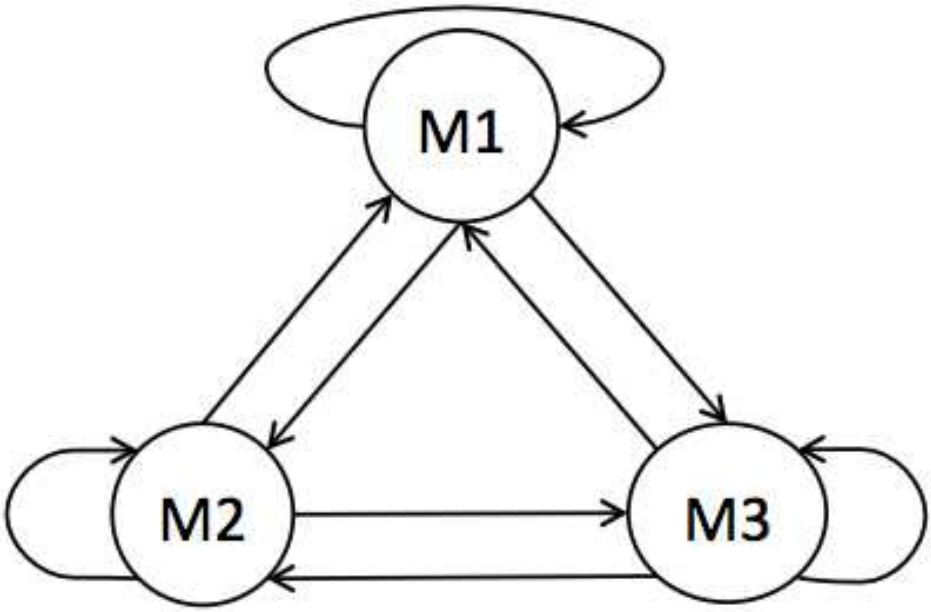


Figure 5.2: Markov model shown with 3 motion primitives [Xia et al., 2012].

We plan four sequences of motion primitives – $u^{\text{Head}}, u^{\text{LArm}}, u^{\text{RArm}}, u^{\text{Legs}}$ – independently according to the emotions and beats of the music. A Markov chain is used to select the motion primitives m_i^c for each sequence u^c. We want to generate m_i^c with the probability $P(m_i^c|m_{i-1}^c, \text{em}_{i-1})$, where em_{i-1} is the emotion detected at the end of m_{i-1}^c. em_{i-1} represents the activation-valence label for the emotion. As a special case, when $i = 1$, we select m_1^c according to $P(m_i^c|\text{em}_i)$.

The motion primitive sequence generated by this model should (i) be continuous, (ii) reflect the musical emotion, and (iii) be interestingly non-deterministic. We set the probability function according to Equation 5.1.

$$P(m_i^c|m_{i-1}^c, \text{em}_{i-1}) = \mathcal{C} \cdot \mathcal{E} \cdot N \tag{5.1}$$

Here, we call $\mathcal{C}$ and $\mathcal{E}$ the *continuity factor* and *emotion factor*, respectively. They are based on the transition between different motion primitives and the emotion-motion primitive relationships. N is a constant normalizing factor. We term this approach CEN.

Continuity factor: The continuity factor is designed to encourage continuity from each motion primitive to the next. Specifically, we want a quick and smooth interpolation from the last keyframe of the current motion primitive to the first keyframe of the next motion primitive. We denote the minimum required time interval computed from Algorithm 7 of this interpolation using $\mathbb{T}(k^{m_i}_{|m_i|}, k_1^{m_{i+1}})$ in Equation 5.2.

$$\mathcal{C} = \exp\left(-\frac{(\mathbb{T}(k^{m_i}_{|m_i|}, k_1^{m_{i+1}}))^2}{2\sigma_m^2}\right) \tag{5.2}$$

Here, σ_m^2 is a constant. The continuity factor is big when the minimum interpolation time is short.

Algorithm 7 Calculate time t to interpolate from a keyframe k_n to the next keyframe k_{n+1}.

$\mathbb{T}(k_n, k_{n+1})$
  for ji $= 1$ to $|k_n|$ **do**
    time[ji] $\leftarrow$ $\frac{|\theta^n_{\text{ji}} - \theta^{n+1}_{\text{ji}}|}{\dot{\theta}^{max}_{\text{ji}}}$ // θ^n_{ji} is joint angle of the jith joint from keyframe k_n, $\dot{\theta}^{max}_{\text{ji}}$ is the maximum joint velocity of the jith joint
  end for
  maxTime $\leftarrow$ max(time)
  avgTime $\leftarrow$ average(time)
  if maxTime $= 0$ **then**
    return 0
  end if
  timeMultiplier $\leftarrow$ $e^{\frac{\text{avgTime}}{\text{maxTime}} \cdot \lambda} \cdot \gamma$
  return maxTime $\cdot$ timeMultiplier

Emotion factor: The emotion factor is designed to select motion primitives whose emotions are similar to the musical emotion. This emotion factor is an example of how the function $\mathbb{S}$ described in Chapter 2 is defined. The emotion factor is defined in Equation 5.3.

Definition 5.1.1. *$AV(m_i^c)$ returns the activation-valence label of m_i^c.*

Definition 5.1.2. *DE*$(AV(m_i^c), em_{i-1}) = \sqrt{(a_i^m - a_{i-1}^{em})^2 + (v_i^m - v_{i-1}^{em})^2}$ *is the Cartesian distance between* $AV(m_i^c)$ *and* em_{i-1}.

$$\mathcal{E} = \exp\left(-\frac{(\text{DE}(AV(m_i^c), \text{em}_{i-1}))^2}{2\sigma_{\text{em}}^2}\right) \tag{5.3}$$

Here, σ_{em}^2 is a constant. The emotion factor is big when the emotional difference is small. Again, em_{i-1} refers to the detected emotion at the end of m_{i-1}^c.

After describing the process to select the sequence of motion primitives, we provide an algorithm to synchronize the schedule of motion primitives with the detected beat times, where each motion primitive in the schedule should end on a beat time. When a motion primitive ends, we begin interpolating to the first keyframe of the next motion primitive.

Calculate Time to Interpolate Between Motion Primitives

Algorithm 7 calculates the time needed to interpolate from the last keyframe, k_j of the previous motion primitive m_{i-1}^c, to the first keyframe, k_1, of the motion primitive m_i^c, using the joint angles of k_j and joint angles of k_l. Although we interpolate between two keyframes with maximum joint angular speeds given the joint angles, we want the robot to dance stably. As we do not implement the controller for the actuators of the robot to account for dynamics, we weight the minimum duration for the interpolation with a multiplier in Algorithm 7. We define λ as the maximum time multiplier, where $\lambda = 0.4$ ($e^{0.4} \approx 1.5$) so that the maximum time multiplier $\leq 1.5\eta$. We define η for each category (Table 5.1). For example, we assign a higher η of 3 for the legs and 1.5 for the head, so that the robot's legs move slower than the head and the robot is more stable at the bottom. We weighted the time multiplier more heavily when the avgTime $\approx$ maxTime which implies that all the joints move almost equally fast.

Table 5.1: η values for joint categories [Xia et al., 2012].

Category	Head	Arm	Leg
η	1.5	2	3

Calculate Timing Parameter β

After describing the process to select the sequence of motion primitives, we provide an algorithm to synchronize the schedule of motion primitives with the detected beat times, where each motion primitive in the schedule should end on a beat time. When a motion primitive ends, we begin interpolating to the first keyframe of the next motion primitive.

The time required for each motion primitive includes the interpolation time between two primitives computed from Algorithm 7 and the times between the keyframes in the motion primitive. If only one beat-time interval is insufficient to execute the motion primitive, we add subsequent beat-time intervals until the total time offered is long enough for execution (Figure 5.3). To make each motion primitive end at a beat time, we stretch the duration by increasing the parameter, β, in each motion primitive to fill the time interval from its starting beat time to the next beat time. In practice, the schedule of motion primitives for each body part is planned independently and executed simultaneously.

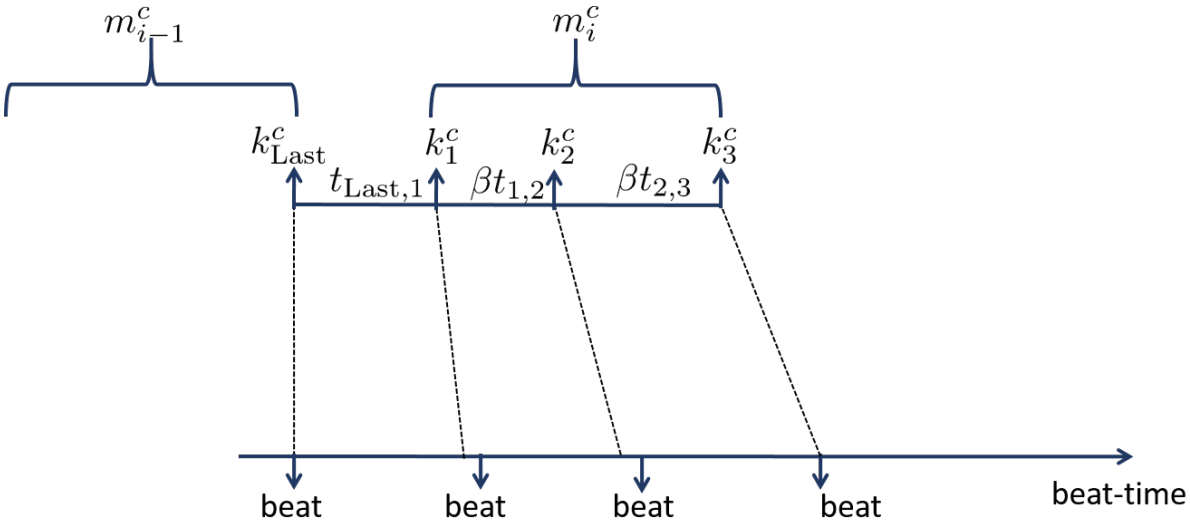


Figure 5.3: Synchronizing motion primitive with beat times.

Emotion For Next Motion Primitive

Motion primitives are selected sequentially and stretched to fill a whole number of beat times. To choose the next motion primitive, we use the emotion em_{i-1} at the end of the previous motion

primitive. We estimate the emotion at each beat time by linearly interpolating the (a, v) values, as the emotions are determined at 15-second intervals.

5.1.2 Experiments

A piece of music is preprocessed by SMERS [Han et al., 2009] to determine the emotions of the music at fixed intervals as described in Section 4.1. Emotional labels of the input signal enable us to select relevant motion primitives. To synchronize the motion primitives to the input signal, we also require the beats of the music. Otherwise, changing motion primitives at fixed intervals will not reflect the beats of the music, which is something that a dancer normally uses as a guide to dance to.

Beat detection is based on audio features associated with changes in amplitude. Peaks in the amplitude mark likely candidates for beat locations. Since beats mostly occur with an overall stable frequency (tempo), these candidate locations are filtered by looking for the ones that are regularly spaced. We use an approach that estimates the global tempo by analyzing the audio features and find the best beat times by using dynamic programming [Ellis, 2006].

Figure 5.4 shows a planned schedule for the right arm motion primitives for a snippet of Peaceful music and Angry music. The motion primitives are synchronized to the beat times. Although there are no motion primitives associated with the emotion Peaceful, the motion primitives m_7 and m_1 are chosen as their AV values are close to the AV value of the emotion Peaceful. We show that we are able to generate dance movements as long as the emotion is assigned an AV value and there are motion primitives labeled with AV values. m_6 is the motion primitive that best corresponds to the emotion Angry and is often selected for Angry music.

We perform an experiment to show how the continuity and emotion factors affect the plan for a Pleased piece of music with right arm motion primitives. We ran 100 iterations for each trial and the results are summarized in Table 5.2. Smaller numbers for the average time to interpolate indicate greater continuity and smaller numbers for the emotion distance indicate greater correspondence between the emotion of the motions and the music's emotion. The results show that both continuity and emotion factors are beneficial as compared to only having one factor or random dancing. Using both continuity and emotion factors strikes a balance between continuity of movements and correspondence between the motion emotion and the music emotion. Although the results shown are for a Pleased piece of music with RArm motion primitives, we ran the same

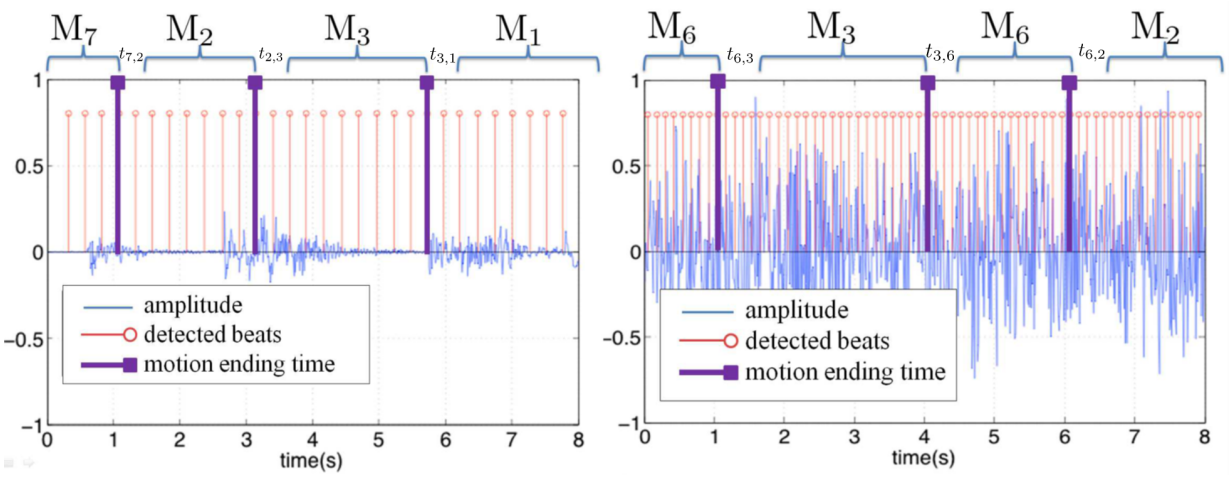


Figure 5.4: RArm motion primitives schedule for peaceful music (left) and angry music (right).

trials on music with different emotions and motion primitives of other categories and arrived at the same conclusion.

Trial	**Average time to interpolate** $\mathbb{T}(k_{\lvert m_i \rvert}^{m_i}, k_1^{m_{i+1}})$	**Average emotion distance** $\mathrm{DE}(AV(m_{i+1}^c), \mathrm{em}_{i+1})$
Continuity and emotion factors	0.6134 ± 0.0930	0.8310 ± 0.0938
Continuity factor	0.6105 ± 0.0842	0.9956 ± 0.1042
Emotion factor	0.6750 ± 0.0875	0.8174 ± 0.0966
Random dancing	0.6581 ± 0.0773	0.9960 ± 0.0959

Table 5.2: A contrast experiment to show the effects of continuity and emotion factors.

We investigate the automatic generation of motions within the context of having a robot dance to any music. We autonomously generate many dance motions combinations for the NAO humanoid robot for any piece of music using a small set of motion primitives. We create smooth movements that reflect the emotions using the continuity and emotion factors to probabilistically select relevant motion primitives. We ensure that the motions are synchronized to the beat times by adjusting the parameters of motion primitives. The stochastic process creates interesting variations given the same piece of music. We successfully fulfilled all the goals to achieve for the task of automating dance choreography for a humanoid robot.

5.2 Selection and Synchronization using Weighted Criteria

Input text converted to speech during human-robot interaction provides a guide to determine relevant gestures. Robots convey the meanings of the speech using relevant gestures (motions) at the right moments. We are interested in automating the selection of relevant gestures and synchronizing gestures based on the timings of the corresponding spoken text. The autonomously generated gestures should satisfy the following goals: First, gestures should convey the meanings represented by labels extracted from the input text. Second, gestures that are directed at a target of interest should be automatically generated given the target's pose (position and orientation). We demonstrate how spatially targeted motion primitives (STM) defined in Chapter 3 are used. Finally, the sequences of gestures generated should be synchronized to the speech generated from the input text and ranked so that we are able to select the best sequence.

5.2.1 Approach – TAMARS

We describe a process to analyze the input signal – text, select relevant motion primitives, generate synchronized sequences of motion primitives, and rank these synchronized sequences of motion primitives based on a weighted list of criteria proposed. We divide the process into three phases as shown in Fig. 5.5.

We term this approach using these three phases – "**T**ext **A**nalysis", "**M**otion **A**nalysis" and "**R**anking of **S**equences", TAMARS. We describe what each phase does in detail and then explain how each phase works using a particular text input.

Phase 1: Text Analysis

A text-to-speech system is used to convert the input signal, text, into speech. The text-to-speech system also produces the start and times of each word in the text.

Phase 2: Motion Analysis

After pre-processing the input signal – text, we find the the sequence of labels (words) – $l_1 \ldots l_n$. We compare each label in the input signal with the labels associated with the motion primitives in the motion library. There are different motion primitives associated with each label.

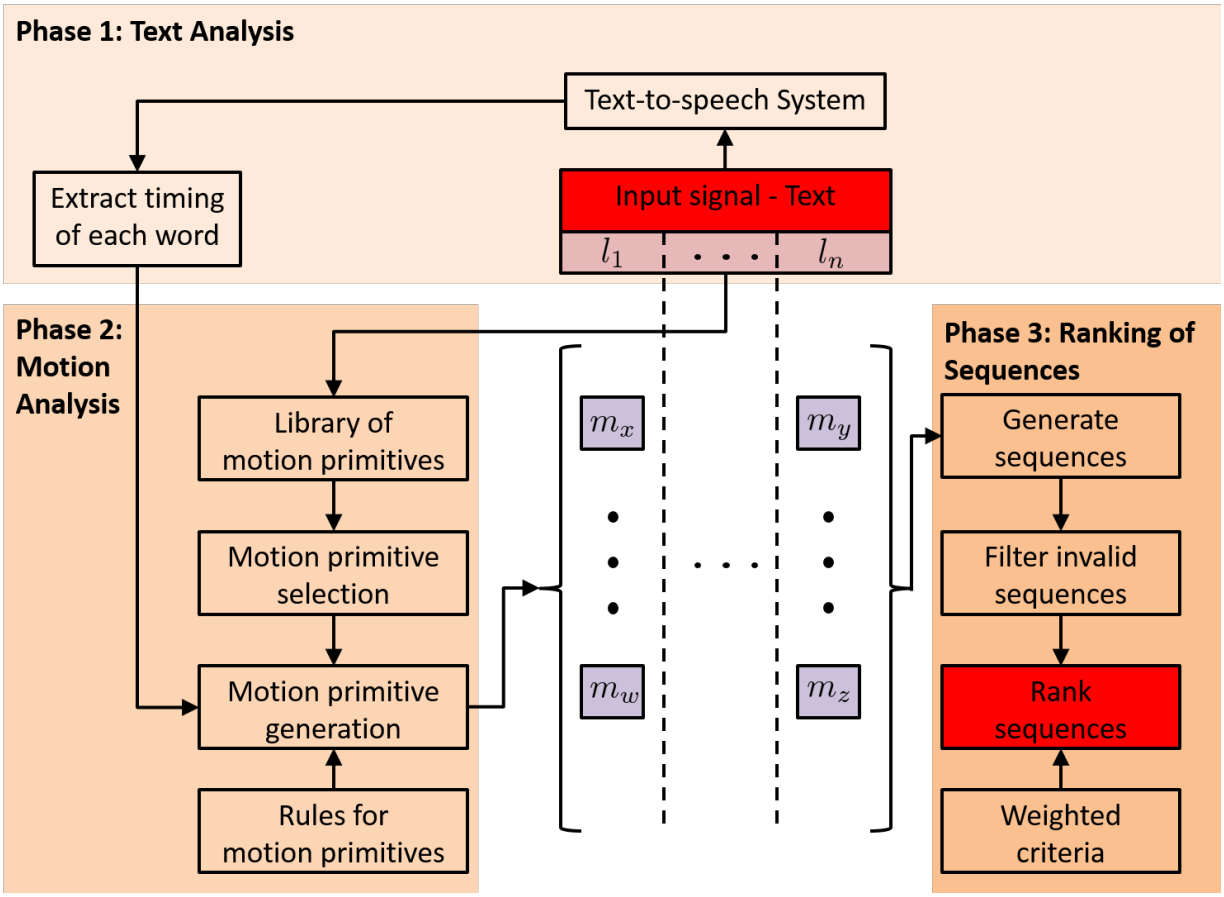


Figure 5.5: Process to rank sequences of motions and the process starts from the red box, "Input signal - Text" and ends at the red box, "Rank sequences".

We determine the similarity of labels using word2vec [Mikolov et al., 2013], which outputs a value from 0 to 1, where a value of 0 means that the pair of labels has no meaning associated with one another, whereas a value of 1 means that the pair of labels has a perfect match in meaning. In this case, the function $\mathbb{S}$ described in Chapter 2 is defined using word2vec [Mikolov et al., 2013]. We select relevant motion primitives where the function $\mathbb{S}$ returns a value larger or equals to μ, where $\mu \in [0, 1]$.

Next, after selecting the relevant motion primitives, we determine the values for each motion primitive's parameters. The values are determined using the rules for motion primitives and the duration of each label. The rules for motion primitives include the target's information for each spatially targeted motion and the number of times a motion primitive should be repeated. We

generate the motion primitive and determine the robot's pose using Algorithm 2 in Chapter 3.

Phase 3: Ranking of Sequences

There is a list of possible motion primitives for each label that is made up of word(s) from the input text. These motion primitives are considered as choices for a particular label. We include the choice to do nothing for each label in the input text. Once we determine the motion primitives for each label, we generate all possible combinations of sequences. We note that multiple motion primitives for each label can be executed simultaneously if they are from different motion primitives categories, $c \in \{\text{Head, LArm, RArm, Legs}\}$.

For each motion sequence, motion primitives are synchronized to the labels when the starting time of a motion primitive corresponds to the starting time of the label and the motion primitive is mapped to the particular label.

Definition 5.2.1. *The starting time of the i^{th} motion primitive, m_i, in the sequence u^s is defined as $t_i^{ms} = \mathbb{T}(k_0, k_1^{m_1}) + \sum_{x=1}^{x=i} \sum_{j=1}^{j=|m_x|-1} \mathbb{T}(k_j^{m_x}, k_{j+1}^{m_x})$. where $k_j^{m_x}$ is the j^{th} keyframe in m_x and k_0 is the initial pose of the robot. The interpolation time computation function $\mathbb{T}$ is defined in Chapter 3.*

Definition 5.2.2. *A sequence of motion primitives, u^s, is synchronized to the pre-processed input signal s when $\forall m_i \in u^s$, $t_i^{ms} = t_i^{ss}$, where t_i^{ss} is the starting time of the i^{th} label.*

We discard the motion sequence if the sequence of motions is not synchronized to the input signal. When the time between two labels in the input signal is longer than the time to interpolate from one motion primitive to the next motion primitive, we have two choices. First, we interpolate from one motion primitive to the initial pose of the robot before interpolating to the next motion primitive such that the next motion primitive starts at the start time of the corresponding label in the input signal. Second, we hold the pose in the last keyframe of the motion primitive so that the next motion primitive starts at the start time of the corresponding label in the input signal. We attempt the first choice before trying the second choice. If both methods fail, the motion sequence is discarded.

After that, we filter invalid motion sequences which cause the robot to fall. We explain how we determine unstable sequences in Section 6.1. After finding the sequences that are valid and executable, we propose the following criteria to rank them:

- Labels: We use the function $\mathbb{S}$ that outputs the similarity between the semantic meanings

of the motion primitive's label and the label in the input signal. We normalize the output of $\mathbb{S}$, where the value is from 0 to 1 and the ranking is fairly weighted.

- Stability: The more stable a sequence is, the higher the ranking of the sequence. We explain how we determine the relative stability of sequences in Section 6.2. The values for the relative stability of sequences are also normalized.

The ranking for each criterion is weighted based on the user's requirements. To determine the ranking of a sequence, $\mathcal{R}_i = \sum_{j=1}^{|\chi|} w_j \mathcal{R}_{i,j}$, where i is the i$^{\text{th}}$ sequence, $|\chi|$ is the total number of criteria, j is the index of the criterion, $\mathcal{R}_{i,j}$ is the normalized value for the sequence i under criterion j and w_j is the weight of the criterion j. The higher the weight assigned to a criterion, the more important the criterion is to the user. The best sequence has a ranking of the least $\mathcal{R}_i$.

5.2.2 Experiment

To demonstrate how the process works, we used the text input, "Little Red Riding Hood looked at her grandmother and gasped out in surprise, 'Oh! Grandmother, what a big mouth you have!' " as an example. We describe each phase in the process:

Phase 1: Text Analysis

We use the text-to-speech system, Festival [The Centre for Speech Technology Research, The University of Edinburgh, 2010], and show the starting time of each word in Table 5.3. We extract labels that correspond to the labels of the motion primitives in the library.

Table 5.3: Timings of words in text input in seconds [Tay and Veloso, 2012].

Little	Red	Riding	Hood	looked	at	her	grandmother
0.18	0.54	0.80	1.20	1.42	1.72	1.83	2.02
and	gasped	out	in	surprise,	"Oh!	Grandmother,	what
2.97	3.14	3.65	3.84	3.96	4.77	4.93	5.92
a	big	mouth	you	have!"			
6.092	6.15	6.39	6.76	6.92	7.12		

Phase 2: Motion Analysis

After extracting the labels and the timings of each label from the input text, we select motion primitives from the database where $\mu \geq 0.8$. In Table 5.4, the motion primitives found are listed with other relevant information such as the labels associated with the motion primitive. There are two types of motion primitives – m^g is a general motion primitive and m^{st} is a spatially targeted motion described in Chapter 3.

Table 5.4: Motion primitives selected [Tay and Veloso, 2012].

Word	Motion primitive	Labels	Total minimum duration (s)	Body part categorization
looked	m_1^{st}	look stare	0.06	Head
looked	m_2^{st}	peer	0.1	Head, left and right arms
surprise	m_1^g	surprise	1.5	Head, left and right arms
surprise	m_2^g	surprise	0.5	Head and Legs
big	m_3^{st}	big	0.3	Left and right arms
big	m_4^{st}	big	1	Legs

Each motion primitive is instantiated based on the rules defined. The rules are that the target of interest is represented by a vector targeted towards the character, "Grandmother" and that each motion primitive is only executed once. We instantiate spatially targeted motions with Algorithm 2.

Phase 3: Ranking of Sequences

After we generate each motion primitive, we determine the list of sequences. For this example, we have a total of $3 \times 3 \times 4 = 36$ sequences as we include the choice to do nothing for each word, and for the word, "big", we execute m_3^{st} and m_4^{st} simultaneously, hence adding another choice. After generating all sequences, we filter for invalid sequences by checking for collisions. We discard 4 sequences that involve m_2^{st} and m_1^g as the arms collide with the head. We also discard 3 sequences that include m_2^g and m_4^{st}, and 3 sequences that contain m_2^g and m_3^{st}, m_4^{st} as the robot is unstable after executing these sequences. There are 6 other sequences that are not synchronized due to the constraints on the duration of the motions. Hence, we are left with only

$36 - 4 - 3 - 3 - 6 = 20$ possible sequences. Lastly, we rank each motion sequence based on the criteria listed in Section 5.2.1 and use a weighting of 1 for each criteria since all the criteria are equally important in this case. Fig. 5.6 shows snapshots of NAO executing the highest ranked motion sequence. The NAO looks in the direction where the character "Grandmother" is at, expresses surprises and expresses how big her mouth is.

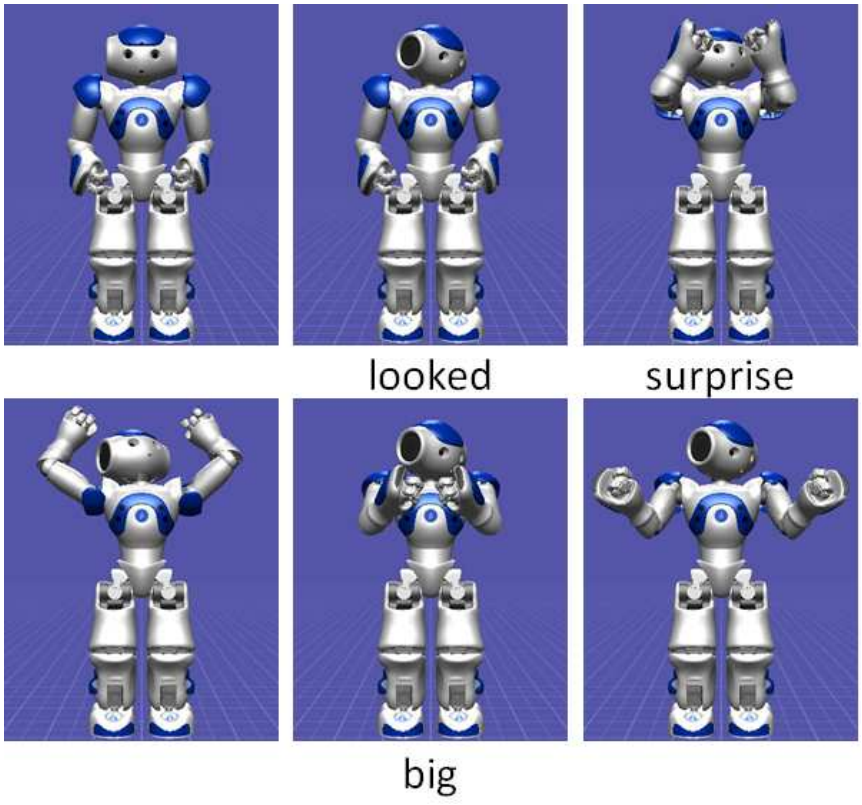


Figure 5.6: Snapshots of the NAO executing the highest ranked motion sequence [Tay and Veloso, 2012].

We categorize motion primitives and show how various categorizations are used to form a motion. We contribute a process to analyze the text input, select the relevant motion primitives based on the analysis of the input, generate the motion primitives and combine them to form motion sequences. We do not vary the parameter, β, and set it to 1 as the durations of the motions are generally longer or equal to the time allowed for a motion to execute. The motion sequences are synchronized to the text-to-speech and if they are not synchronized, the motion sequence is discarded. The valid sequences of motion sequences are ranked based on the set of

criteria proposed. Selection of relevant motions is highly dependent on the accuracy of the text analysis performed, the richness of the motion primitives database and the associated labels.

5.3 Selection using Audience Preferences

A robot is tasked to animate an input signal with a sequence of motions autonomously. For example, a robot dances to a piece of music [Xia et al., 2012] or animates a story [Tay and Veloso, 2012]. Given a library of labeled motions where multiple motions are mapped to each label, multiple sequences are feasible to animate the input signal. In this section, we aim to select the sequence which is most preferred by the audience through the feedback of some sequences.

5.3.1 Problem Description and Assumptions

In this section, we describe the motivating scenarios, and present the formal problem definition and assumptions.

Motivating Scenario

Suppose that a humanoid robot is tasked with animating a story. The story comprises sentences, where some words in each sentence are animated and are the labels from the pre-processed input signal, e.g., in "John *waved* at the *bird*", the words in italics are *labels* to be animated.

For each label in a sentence, there may be multiple motions that are applicable, e.g., to animate "*waved*", the robot can wave with its left/right arm, with its palm open/closed, and move quickly/slowly. As such, for each sentence, with multiple labels and multiple motions per label, there are multiple unique sequences of motion that are feasible.

Each *label-motion pair* (e.g., *"waved"*-Wave Slowly With Open Left Hand) has a unique audience preference value, and as such, each sequence of motions for the sentence has an audience preference value. Further, the audience preference value may degrade each time the audience views a motion, as the audience may get increasingly bored with seeing the same motion multiple times. The goal is to select the sequence of motions for the sentence with the highest audience preference value, while minimizing the number of times the audience is queried so that the degradation is minimal.

The audience preference value for each sequence of motions is observed at the end of the motion sequence. The audience preference value is a numerical rating that is determined via a device which measures the audience feedback. The audience preference value for each label-motion pair is not observed and is unknown. Given that the audience preference value for a motion sequence is dependent on the audience preference values of the label-motion pairs used, if we are able to determine the audience preference values for the label-motion pairs, we can determine the audience preference value for all motion sequences.

When there are multiple labels in an input signal, not all labels can be animated due to reasons such as the duration of a motion is too long and the label is not animated, or stability issues such as when a motion is used to animate the label, the motion sequence becomes unstable. To determine the preference of which label is to be animated in such situations, the audience assigns weights to indicate their preference of the labels to be animated.

Formal Problem Definition

Definition 5.3.1. *Let lm_k be the k^{th} label-motion pair, and LM be the set of all label-motion pairs.*

There exists different sequences of motions for the robot to animate the signal s, where the labels of the signal match the corresponding labels in the motions and the motions are synchronized to the starting times of the labels in the signal.

Definition 5.3.2. *Let $u^s = (lm_1, \ldots, lm_D)$ be an ordered set of D label-motion pairs for a pre-processed input signal s, where $D \geq 2$. Let U^s be the set of all possible sequences of motion primitives for s.*

There exists a unique audience preference rating for every motion sequence. The audience preference rating for a motion sequence is provided as feedback at the end of the motion sequence. For example, the audience claps or raises a colored paddle (a green paddle to indicate yes or a red paddle to indicate no) to indicate their preferences [Knight et al., 2011]. The weights assigned to the ratings of the label-motion pairs are the weights of the labels in the input signal to indicate the audience preference of the labels to be animated.

Definition 5.3.3. *Let am^k be the audience preference rating of a label-motion pair lm_k. The audience rating of a sequence of motions is $\mathbb{A} : U^s \rightarrow \mathbb{R}^+$, where $\mathbb{A}(u^s) = \sum_{lm_k \in u^s} w_k^s am^k$, for some weights w_k^s. Let the set of weights assigned to the ratings of the individual label-motion*

pairs in all the possible sequences, U^s, be W^U.

The audience feedback is captured via a device, e.g., a sound detector that estimates the number of people who clapped or a camera that determines the number of paddles of a particular color. However, the numerical rating outputs of these devices are noisy. For example, the sound detector may not capture all the claps or the camera may not see all the colored paddles raised.

Definition 5.3.4. *Let as^i be the noisy observation of the rating for sequence u_i^s (the i^{th} sequence for signal s), i.e., $as^i \sim \mathcal{N}(\mathbb{A}(u_i^s), R_k)$ for some noise variance R_k.*

The goal is to find the best sequence of label-motion pairs, i.e., $\operatorname{argmax}_i \mathbb{A}(u_i^s)$.

One approach would be to repeatedly try all possible sequences multiple times to determine the best sequence and account for the noise in the observation. However, people get bored when viewing the same animation multiple times.

As such, we define a model that simulates the effects of boredom, when viewing a label-motion pair repeatedly. We term it the degradation model, where the rating for an individual label-motion pair in the sequence degrades by a factor each time the individual label-motion pair is viewed. This degradation to the rating means that the audience prefers the individual label-motion pair a little less each time the label-motion pair is seen.

Definition 5.3.5. *Let the degradation factor be DF $\in [0, 1]$.*

Assumptions

- The rating for each label-motion pair is independent.
- The observation noise R_k is known.
- The weights $w_k^s \in W^U$ are known.
- The degradation factor DF is known.
- The audience preference value of a motion sequence is a weighted sum of the audience preference values of the label-motion pairs based on the weights and degradation factors.

5.3.2 Approach – MAK

We model the problem as a multi-armed bandit, where each arm represents a motion sequence. At each iteration, we pull an arm and observe a noisy rating of the sequence.

Next, we use a Kalman filter to estimate the ratings of the individual label-motion pairs by using the series of noisy observations for different sequences over time.

We model the estimated ratings of each label-motion pair as a distribution with a mean and variance. The lower the variance of the estimated rating, the more confident we are about the mean ("true value") of the rating of the label-motion pair.

Using our model of the audience preferences, by determining the rating of each label-motion pair, we can calculate the audience preference value of a motion sequence and do not need to continuously ask the audience for feedback.

Definition 5.3.6. *Let $\widetilde{am}^i$ be the estimate of the rating of the label-motion pair lm_i and the mean of the rating in our model. Let $\widetilde{vam}^i$ be the variance of the estimated rating of the label-motion pair lm_i. Let $\widetilde{AM}_t$ be the set of estimated ratings for all label-motion pairs, LM, at iteration t and $\widetilde{VAM}_t$ be the set of variances of the estimated rating for all label-motion pairs at iteration t.*

The ratings of the individual label-motion pairs are modeled as the state variables in the Kalman filter, and the observation is the observed rating for the sequence.

At each iteration, we determine the arm to pull using Thompson Sampling, a multi-armed bandit algorithm, which in turn uses the Kalman's estimated state.

We repeat the process of choosing the arm to pull and using the Kalman filter to update our estimates of the individual ratings of the label-motion pairs till we reach a stopping condition. We term our approach "MAK" - **M**ulti-**A**rmed bandit and **K**alman filter.

Definition 5.3.7. *Let $\widetilde{am}_t^i$ be the estimate of the rating of the label-motion pair lm_i in our model at iteration t, where $\widetilde{am}_t^i \in \widetilde{AM}_t$. Let $\lambda_i = |\widetilde{am}_t^i - \widetilde{am}_{t-1}^i|$ be the absolute difference between the estimate of the rating of the label-motion pair lm_i at iteration t and $t-1$. Let $\hat{\lambda}_i$ be the maximum absolute difference.*

Definition 5.3.8. *Let $\widetilde{vam}_t^i$ be the variance of the label-motion pair lm_i in our model at iteration t, where $\widetilde{vam}_t^i \in \widetilde{VAM}_t$. Let $\lambda_i^v = |\widetilde{vam}_t^i - \widetilde{vam}_{t-1}^i|$ be the absolute difference between the estimate of the rating of the label-motion pair lm_i at iteration t and $t-1$. Let $\hat{\lambda}_i^v$ be the maximum absolute difference.*

MAK will stop when either of the following two conditions is met: (1) the maximum iterations MI has occurred, or (2) the maximum absolute change in the current estimated rating of the label-motion pairs and the previous estimated rating of the label-motion pairs is less than or

equals to ϵ, i.e., $\max_i(\lambda_i, \lambda_i^v) \leq \epsilon$, where $i \in \{1, 2, \ldots, \| \widetilde{AM} \|\}$.

MAK Algorithm

We present the algorithm for MAK in Algorithm 8. We will discuss the initialization of $\widetilde{AM}, \widetilde{VAM}$ in the Experiments section. Algorithm 8 uses both Algorithm 9 and Algorithm 10.

Algorithm 8 Determine the best sequence u with the highest audience preference.

MAK($U^s, W^U, \widetilde{AM}, \widetilde{VAM}$)
  $t \leftarrow 0$
  $\Delta \leftarrow$ Infinity
  while ($t \leq$ MI) **and** ($\Delta > \epsilon$) **do**
    $P_t \leftarrow$ diag($\widetilde{VAM}$) // P_t is a diagonal matrix where the diagonal values are the respective variances of the individual label-motion pairs
    $u_c^s \leftarrow$ MAB($U^s, W^U, \widetilde{AM}, \widetilde{VAM}$) // Algo. 9
    $as^c \sim \mathbb{N}(\mathbb{A}(u_c^s), R_k)$ // as^c is the noisy observation of the rating of sequence u_c^s
    $[\widetilde{AM}_t, \widetilde{VAM}_t] \leftarrow$ Kalman($W^U, \widetilde{AM}, \widetilde{VAM}, u_c^s, as^c, P_t$) // Algo. 10
    $\hat{\lambda_i} \leftarrow \max_{i \in \{1,\ldots,\|\widetilde{AM}_t\|\}} |\widetilde{am}_t^i - \widetilde{am}^i|$
    $\hat{\lambda_i^v} \leftarrow \max_{i \in \{1,\ldots,\|\widetilde{VAM}\|\}} |\widetilde{vam}_t^i - \widetilde{vam}^i|$
    $\Delta \leftarrow \max_i(\hat{\lambda_i}, \hat{\lambda_i^v})$
    $\widetilde{AM} \leftarrow \widetilde{AM}_t$
    $\widetilde{VAM} \leftarrow \widetilde{VAM}_t$
    $t \leftarrow t + 1$
  end while

Algorithm 9 is a multi-armed bandit algorithm that determines the sequence to query based on the means and variances of the label-motion pairs, $\widetilde{AM}, \widetilde{VAM}$. In Algorithm 9, we use Thompson Sampling as an example. Other multi-armed bandit algorithms, e.g., Upper Confidence Bound, are also applicable.

Algorithm 10 uses the Kalman filter to estimate the individual audience preference values for each label-motion pair based on the noisy observation of the rating of the label-motion pairs in a sequence u_c^s.

We illustrate how Algorithm 8 works with an input signal s consisting of the following labels (l_1, l_2). There are four possible sequences – $u_1^s, u_2^s, u_3^s, u_4^s$ with a motion library of four label-motion pairs – $lm_1 = (l_1, m_1), lm_2 = (l_1, m_2), lm_3 = (l_2, m_3), lm_4 = (l_2, m_4)$. The sequences

Algorithm 9 Determine the next sequence to query based on the means and variances of the label-motion pairs using a multi-armed bandit algorithm – Thompson Sampling.

MAB($U^s, W^U, \widetilde{AM}, \widetilde{VAM}$)
  $v_{\text{max}} \leftarrow 0$
  for $i = 1$ **to** $\mid U^s \mid$ **do**
    $v_i \leftarrow 0$
    for $j = 1$ **to** $\mid u_i^s \mid$ **do**
      $v_i \leftarrow v_i + w_j^i \times$ Random($\widetilde{am}^j, \widetilde{vam}^j$) // the function Random randomly samples from a distribution with a mean, $\widetilde{am}^j$ and variance, $\widetilde{vam}^j$
    end for
    if $v$max $< v_i$ **then**
      $v_i = v$max
      $u_{\text{max}}^s \leftarrow u_i^s$
    end if
  end for
  return u_{max}^s

have the following label-motion pairs:

- $u_1^s = (lm_1, lm_3)$;
- $u_2^s = (lm_1, lm_4)$;
- $u_3^s = (lm_2, lm_3)$;
- $u_4^s = (lm_2, lm_4)$.

The estimated ratings of the label-motion pairs in our model are $\widetilde{AM} = (\widetilde{am}^1, \widetilde{am}^2, \widetilde{am}^3, \widetilde{am}^4)$ and we initialize the estimated ratings to some value in the beginning. We also initialize $\widetilde{VAM}$ with a large number since we are not confident about the initial estimated rating $\widetilde{AM}$.

Using Algorithm 9 where Thompson Sampling is used as an example, we randomly sample values using the function Random that uses the model of the estimated rating of the label-motion pairs – $\widetilde{AM}$ and $\widetilde{VAM}$. We compute the weighted sum of these values for each sequence and return the sequence with the highest sampled value. Next, we observe a noisy audience preference, as^c, of the sequence u_c^s using the function $\mathbb{A}$.

We use the observation as^c in Algorithm 10, where the Kalman filter uses the observation to update the estimates of the estimated ratings of the label-motion pairs in our model.

Different sequences are made up of different label-motion pairs.The function IndicateLMUsed

Algorithm 10 Kalman filter where the the states are the estimates of the ratings of the individual label-motion pairs, $\widetilde{AM}$.

Kalman($W^U, \widetilde{AM}, \widetilde{VAM}, u_c^s, as^c, P_t$)

$F_t \leftarrow$ getStateTransition(IndicateLMUsed(u_c^s), DF) // F_t is a diagonal matrix, where the diagonals consist of 1 for the label-motion pairs not used in u_c^s and DF for the label-motion pairs used in u_c^s

$\hat{x}_{t|t-1} \leftarrow F_t\widetilde{AM}$ // Predicted state estimates, there is no B_tu_t term as there is no known control input and no process noise w_t

$H_t \leftarrow$ getObservationModel(IndicateLMUsed(u_c^s), W^U) // H_t is a vector that indicates the label-motion pairs in the sequence u_c^s and their respective weights

$P_{t-1|t-1} \leftarrow$ diag($\widetilde{VAM}$)

$P_{t|t-1} \leftarrow F_tP_{t-1|t-1}F_t^T$ // Predicted covariance estimates

$\tilde{y}_t \leftarrow as^c - H_t\hat{x}_{t|t-1}$ // Innovation

$s_t \leftarrow H_tP_{t|t-1}H_t^T + R_t$ // Innovation covariance

$K_t \leftarrow P_{t|t-1}H_t^TS_t^{-1}$ // Optimal Kalman gain

$\hat{x}_{t|t} \leftarrow \hat{x}_{t|t-1} + K_t\tilde{y}_t$ // Updated state estimate

$P_{t|t} \leftarrow (I - K_tH_t)P_{t|t-1}$ // Updated estimate covariance and I is an identity matrix

$\widetilde{AM}_t \leftarrow \hat{x}_{t|t}$

$\widetilde{VAM} \leftarrow$ extractVariance($P_{t|t}$)

return $[\widetilde{AM}, \widetilde{VAM}]$

in Algorithm 10 takes in a sequence u_c^s and returns a vector whose values indicate if the corresponding unique label-motion pair $lm_i \in LM$ is used in the sequence u_c^s. If the i^{th} value in the vector is 1, lm_i is used in the sequence u_c^s, otherwise the value is 0. For example, the function IndicateLMUsed(u_1^s) returns a vector with values $\begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix}$.

The state transition matrix F_t depends on the label-motion pairs used in the sequence and the degradation factor DF. The function getStateTransition returns a matrix F_t. If u_1^s is being observed and DF $= 1$, F_t is $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$. If u_2^s is being observed and DF $= 0.999$, F_t is

$$\begin{bmatrix} 0.999 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.999 \end{bmatrix}.$$

H_t is the observation model which maps the true state space into the observed space. Therefore, H_t depends on the label-motion pairs used in the sequence and the weights assigned. If the weights for the observed sequence u_3^s are equal, the function getObservationModel returns $H_t = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$. If the weights are not equal and the weights for u_4^s are w_1 for lm_2 and w_2 for lm_4, the function getObservationModel returns $H_t = \begin{bmatrix} 0 & w_1 & 0 & w_2 \end{bmatrix}$.

We form a covariance matrix using diag where the covariance matrix is a diagonal matrix, and the diagonals are the respective $\widetilde{vam}^i \in \widetilde{VAM}$.

Algorithm 10 updates the estimates of the mean $\widetilde{AM}$ and variance $\widetilde{VAM}$ for the unique label-motion pairs. We use the function extractVariance to extract the diagonals of the matrix $P_{t|t}$ to determine $\widetilde{VAM}$.

We use the function $\max_i(\lambda_i, \lambda_i^v)$ to determine Δ and update our model of $\widetilde{AM}$ and $\widetilde{VAM}$. We repeat these steps till we reach the maximum number of iterations MI or $\Delta \leq \epsilon$.

5.3.3 Comparison – Least Squares Regression

Given that we know the label-motion pairs used in each sequence and the multiple noisy observations that we make for each sequence, we consider least squares regression as the baseline comparison to estimate the rating of the individual label-motion.

Least squares regression uses the equation $Ax = B$. x is a $|LM| \times 1$ vector containing the list of ratings for each label-motion pair. A is a $n \times |LM|$ matrix that indicates the label-motion pairs used in n observed sequences and the weights assigned to the label-motion pairs used. B is a $n \times 1$ vector containing the noisy observations for n sequences.

Similarly, we illustrate least squares regression with the same example for MAK, where the four possible sequences are $u_1^s, u_2^s, u_3^s, u_4^s$ and we have a motion library of four label-motion pairs – lm_1, lm_2, lm_3, lm_4.

For this example, $x = \begin{bmatrix} \widetilde{am}^1 \\ \widetilde{am}^2 \\ \widetilde{am}^3 \\ \widetilde{am}^4 \end{bmatrix}$.

When DF $= 1$ and we observe the four possible sequences in order, $A = \begin{bmatrix} w_1^1 & 0 & w_3^1 & 0 \\ w_1^2 & 0 & 0 & w_4^2 \\ 0 & w_2^3 & w_3^3 & 0 \\ 0 & w_2^4 & 0 & w_4^4 \end{bmatrix}$.

If DF < 1, we keep track of the number of times the unique label-motion pair is used. We define the number of times the unique label-motion pair, lm_i is viewed as v^i. Each row of A will contain the values IndicateLMUsed$(u_c^s) \cdot w_i^c \cdot \text{DF}^{\max(0,v^i-1)}$, where i is the index of the label-motion pair in the sequence u_c^s.

Least squares regression estimates the rating of the individual label-motion pair in the event that there is no observation noise of the audience preference of a sequence. Given p unique motion labels, we need at least p sequences that include all unique label-motion pairs to determine the individual audience preference values. However, since the observation is noisy, we will need at least $30p$ sequences using the Central Limit Theorem in order to get a good estimate of the ratings of the individual label-motion pairs.

Therefore, for least squares regression, we randomly pick p sequences at the start of each trial. These p sequences include all unique label-motion pairs so that the equations formed are not under-constrained. Following that, to determine the next sequence to query, we select the sequence u_{max}^s with the highest cumulative linear sum using the least squares estimates of the ratings of the label-motion pairs. We obtain a noisy observation of the rating as^{max} using the function $\mathbb{A}$.

We continue adding rows to the matrix A and the vector B till one of the two stopping conditions is met. The first stopping condition is that the maximum iterations MI has occurred and MI $> \| \ LM \ \|$, where $\| \ LM \ \|$ is the number of unique label-motion pairs used in all possible sequences U^s. The second stopping condition is that the maximum absolute change in the current estimated rating of the label-motion pairs and the previous estimated rating of the label-motion pairs is less than or equals to ϵ. The second condition is checked using the equation $\max_i(\lambda_i) \ \leq \ \epsilon$, where $i \in \{1, 2, \ldots, \| \ \widetilde{AM} \ \|\}$.

5.3.4 Experiments

We consider two models of the audience – Constant and Degradation. Constant is the model where the rating of a label-motion pair remains constant regardless of the number of times the label-motion pair is viewed, so DF $= 1$. The model, Degradation, is the model where the rating for a label-motion pair degrades by a constant known factor each time it is seen and we set DF $= 0.999$.

To evaluate the performance of our approach, MAK, versus the baseline comparison of least squares regression, we created four labels with ten unique label-motions per label, resulting in a total of forty unique label-motion pairs. We also generated a black box where the ratings for the unique label-motion pairs are uniformly randomly generated from 0 to 100 and are hidden from our model of the ratings of the label-motion pairs. The number of possible sequences is based on the number of labels in the signal. For example, if there are n labels, there are 10^n possible sequences.

We query the black box for the audience rating using the function $\mathbb{A}$ and $\mathbb{A}$ returns a noisy value that is computed based on an equally weighted sum of the ratings of the label-motion pairs in the sequence. The noise added to the observation is $R_k = 100$. We used the stopping conditions of MI $= 500$ and $\epsilon = 0.1$.

We compared MAK against Least Squares regression for each experiment and ran 30 trials for each experiment since there are randomness in the sequences selected for queries. We varied the following variables:

- Initialization: We initialized our model of the rating for each unique label-motion pair in $\widetilde{AM}$, with one of three different values: minimum value of 0, mean value of 50 or maximum value of 100. We initialized each variance in $\widetilde{VAM}$ to be 100^2 for each label-motion pair.
- Number of label-motion pairs in a sequence: We varied the input signal by changing the number of label-motion pairs in a sequence. We considered input signals with 2 label-motion pairs, 3 label-motion pairs and 4 label-motion pairs.
- Audience model: We conducted experiments with the two models – Constant where DF $=$ 1 and Degradation where DF $= 0.999$.

As there are too many combinations of the different variables, we choose to conduct the

following experiments:

- Comparison of three initializations with 2 label-motion pairs for 2 audience models: We initialized the model with three different initializations: Minimum [0] / Mean [50] / Maximum [100]. We also compared two audience models – Constant versus Degradation.
- Comparison of two / three / four label-motion pairs: We varied the number of labels in the input signal; where there are either two, three, four label-motion pairs. We also compared two audience models – Constant versus Degradation.

For numerical computations involving matrices, we used GNU Octave. For least squares regression, we used Octave's `lsqnonneg` function and used the inputs A and B and the initialized model of the ratings for the unique label-motion pairs as the initial guess.

In our experiments, we assume that the weights for the ratings of the individual label-motion pairs are equal. These weights should come from empirical evidence of how the label-motion pairs are weighted for the audience's evaluation of a sequence. Since we do not have a good model for the weights, we use an equally weighted sum for our experiments. We believe that changing the weights will have little effect on our results.

All the results shown are averaged across thirty trials. We show the results for the experiment – Comparison of three initializations with two label-motion pairs with a Constant audience model in three figures:

1. Figure 5.7: Our model's rating for the individual label-motion pair is initialized to the minimum value of 0.
2. Figure 5.8: Our model's rating for the individual label-motion pair is initialized to the mean value of 50.
3. Figure 5.9: Our model's rating for the individual label-motion pair is initialized to the maximum value of 100.

In these three figures, the highest rating for the best sequence in the black box is plotted in black and labeled as "Best". We define two terms for the results. "ModelBest" refers to the rating of the best sequence based on our model of the ratings of the individual label-motion pairs. "FindBestAndGetFromBlackBox" refers to finding the best sequence based on our model of the ratings of the individual label-motion pairs and querying the noise-free rating of this sequence from the blackbox.

With "ModelBest", we show how well our model of the ratings of individual label-motion

pairs fare in terms of estimating the rating of the best sequence.

With "FindBestAndGetFromBlackBox", we show whether our model of the ratings of individual label-motion pairs is accurate by finding the best sequence and getting a noise-free observation from the blackbox. We highlight that "FindBestAndGetFromBlackBox" is not available to our model in MAK or least squares regression. "FindBestAndGetFromBlackBox" is used to analyze the results so as to show that the approach is indeed able to determine the best sequence given that the rating of the best sequence correspondings to "Best".

The dashed lines in each figure represent the data from Least Squares regression whereas the straight lines represent the data from MAK. The blue lines represent data from "ModelBest" for the respective approaches whereas the orange lines represent data from "FindBestAndGetFromBlackBox" for the respective approaches.

Figures 5.7-5.9 show that our approach - MAK performs better than Least Squares in terms of finding the best sequence as "MAKFindBestAndGetFromBlackBox" converges to "Best", whereas there is a gap between "LeastSquaresFindBestAndGetFromBlackBox" and "Best". We also show that MAK is able to model the rating for the best sequence accurately since "MAKModelBest" converges to "MAKFindBestAndGetFromBlackBox" and "Best". We plot only 100 iterations per figure so as to make a fair comparison between MAK and Least Squares regression and to show convergence in values.

We compare Figures 5.7-5.9 to determine if different initializations make a difference in the number of iterations for "ModelBest" to converge to "FindBestAndGetFromBlackBox" and "Best". Since the ratings in the model are real numbers, we define convergence when $|\text{ModelBest} - \text{FindBestAndGetFromBlackBox}| < 1$ and $|\text{Best} - \text{FindBestAndGetFromBlackBox}| < 1$. We plot two arrows:

- Green arrow: Number of iterations "MAKModelBest" converges to "MAKFindBestAndGetFromBlackBox" and "Best".
- Green dashed arrow: Number of iterations "LeastSquaresModelBest" converges to "LeastSquaresFindBestAndGetFromBlackBox". We highlight that LeastSquares **does not** converge to "Best".

MAK converges at 46 iterations using the initialization of the mean value of 50 and 46 is the least number of iterations when compared to the other two initializations – minimum and maximum. There are no significant differences in the number of iterations for the convergence in values for

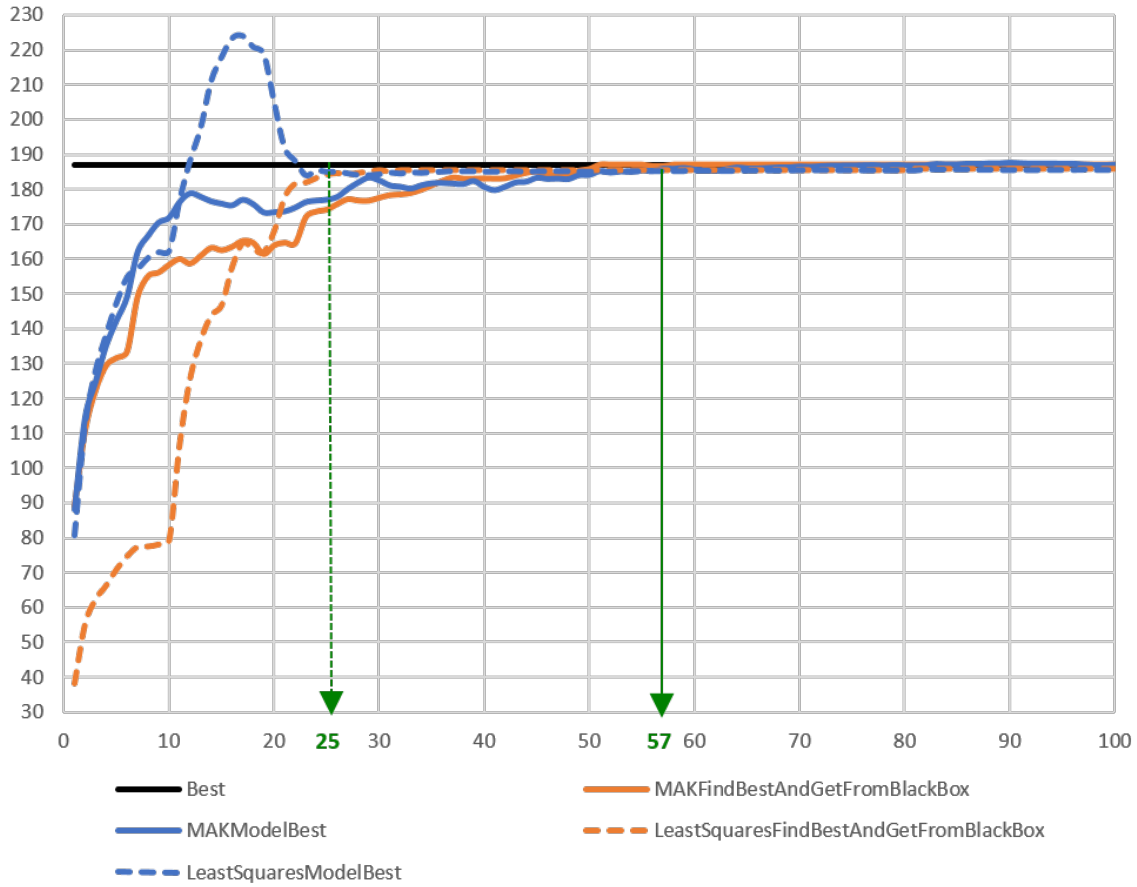


Figure 5.7: Comparison of MAK versus Least Squares For Constant audience model with minimum value initialization.

Least Squares.

Next, we show the results for the experiment – Comparison of three initializations with two label-motion pairs with the Degradation audience model in three figures:

1. Figure 5.10: Our model's rating for the individual label-motion pair is initialized to the minimum value of 0.
2. Figure 5.11: Our model's rating for the individual label-motion pair is initialized to the mean value of 50.
3. Figure 5.12: Our model's rating for the individual label-motion pair is initialized to the maximum value of 100.

In these three figures, the highest audience rating for the best sequence in the black box

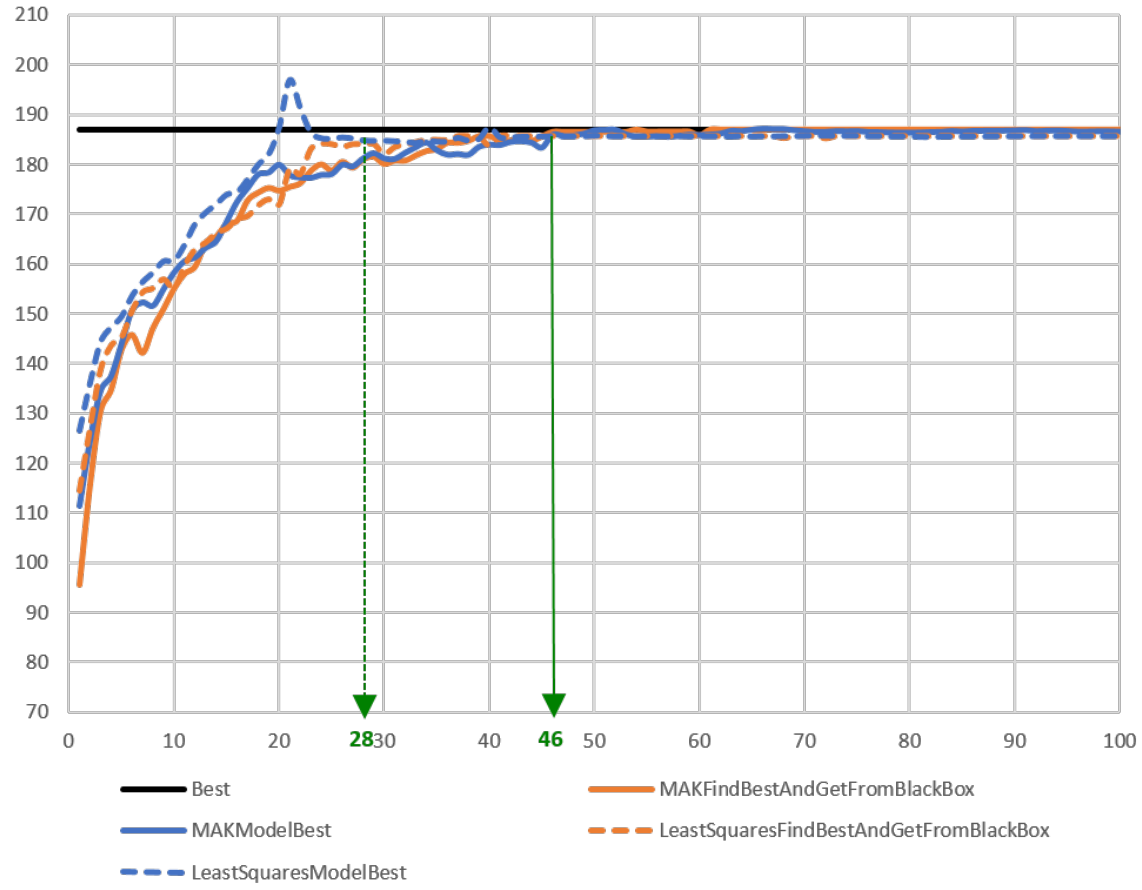


Figure 5.8: Comparison of MAK versus Least Squares For Constant audience model with mean value initialization.

using the approach MAK is labeled as "MAKBest" and the highest audience rating for the best sequence in the black box using the least squares approach is labeled as "LeastSquaresBest". The "Best" value is shown separately for MAK and Least Squares as the average number of times the label-motion pairs in the best sequence is queried is different, hence, in the figures MAK degrades less than Least Squares though they use the same degradation factor of 0.999.

For a degradation audience model, convergence in the approach MAK occurs with the least number of 50 iterations with the maximum value initialization compared to the other two values for initialization. We note that the difference in the number of 55 iterations with respect to the convergence in values with the mean value initialization may not be significant. Convergence in the approach Least Squares occurs with the least number of iterations with minimum value

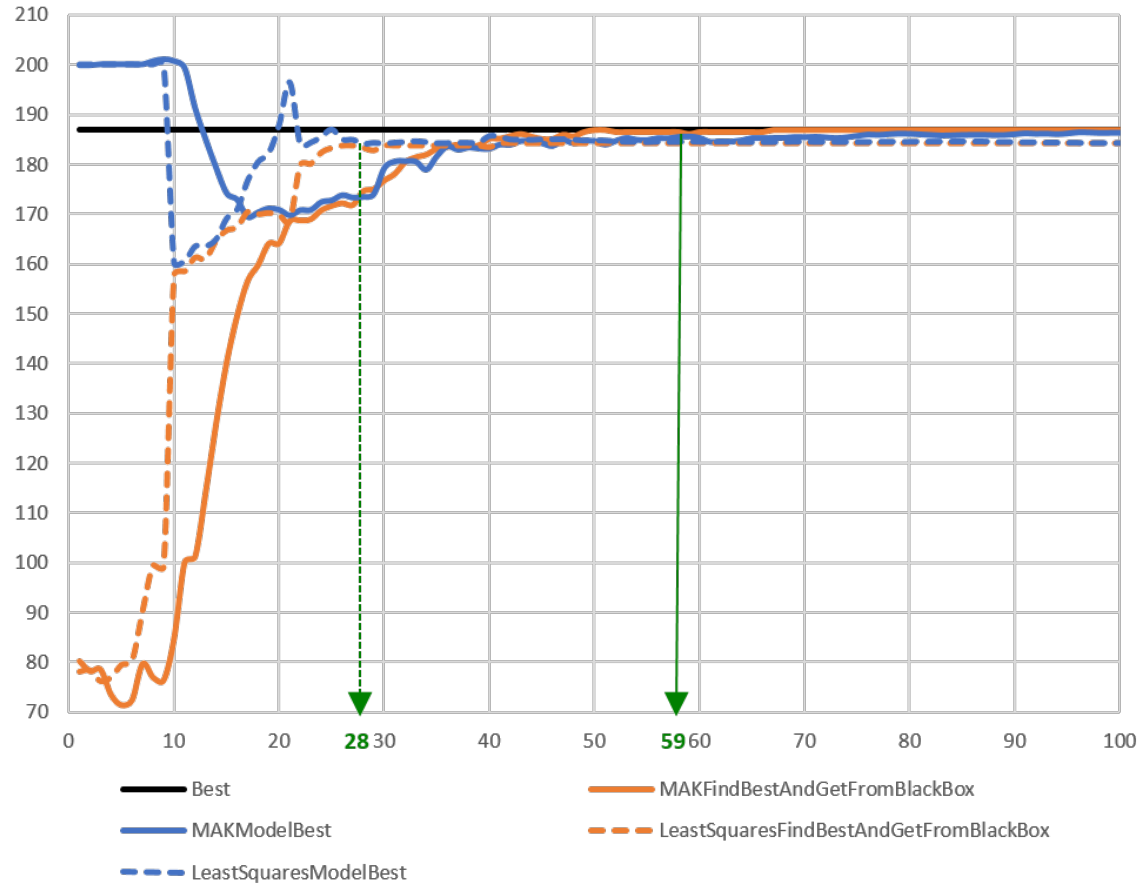


Figure 5.9: Comparison of MAK versus Least Squares For Constant audience model with maximum value initialization.

initialization, but we note that the difference in the least number of iterations with mean value initialization may not be significant.

It is difficult to visually show the difference in performance of MAK versus Least Squares as the difference in the "ModelBest" and "FindBestAndGetFromBlackBox", and the difference between "Best' and "FindBestAndGetFromBlackBox" are indiscernible on the plots. Therefore, we also present numerical results in Tables 5.5-5.7 for the experiment "Comparison of two / three / four label-motion pairs".

We define these two differences below.

Definition 5.3.9. *Let the absolute difference between ModelBest and FindBestAndGetFromBlackBox be* Υ *and the absolute difference between Best and FindBestAndGetFromBlackBox be* ρ.

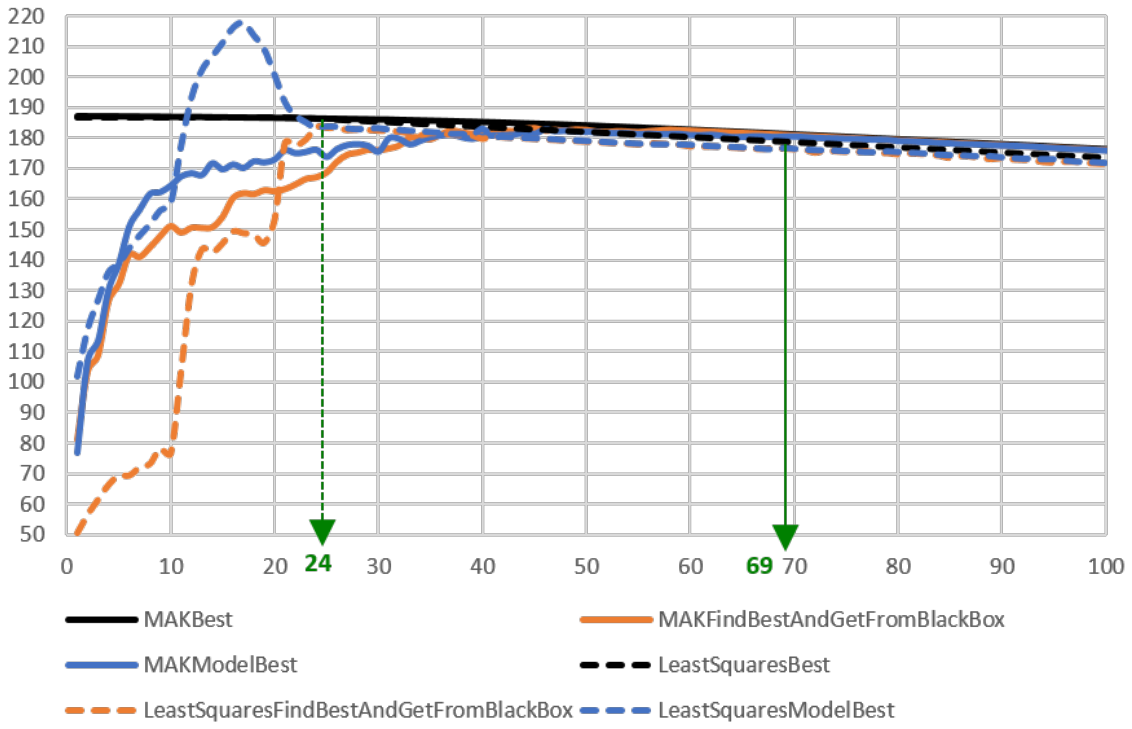


Figure 5.10: Comparison of MAK versus Least Squares For Degradation audience model with minimum value initialization.

For the experiment "Comparison of two / three / four label-motion pairs", we show in Tables 5.5-5.7 that regardless of the number of labels, MAK always finds the best sequence since ρ is 0 for the constant audience model and $\rho \approx 0$ for the degradation model, but Least Squares is unable to find the best sequence given that $\rho > 0$. The model of the ratings of the individual label-motion pairs takes longer to converge for MAK compared to Least Squares, but Least Squares is unable to find the best sequence.

Since the number of iterations also refers to the number of times sequences are queried and the number is much less than the possible number of sequences for two/three/four labels, we show that we do not have to query all sequences for either MAK or least squares regression.

We show that MAK selects the best sequence without querying all possible sequences. MAK performs better than least squares regression in terms of selecting the best sequence and is capable of using noisy observations of the ratings for different sequences.

MAK appears to take more iterations than Least Squares to converge, but we note that we are

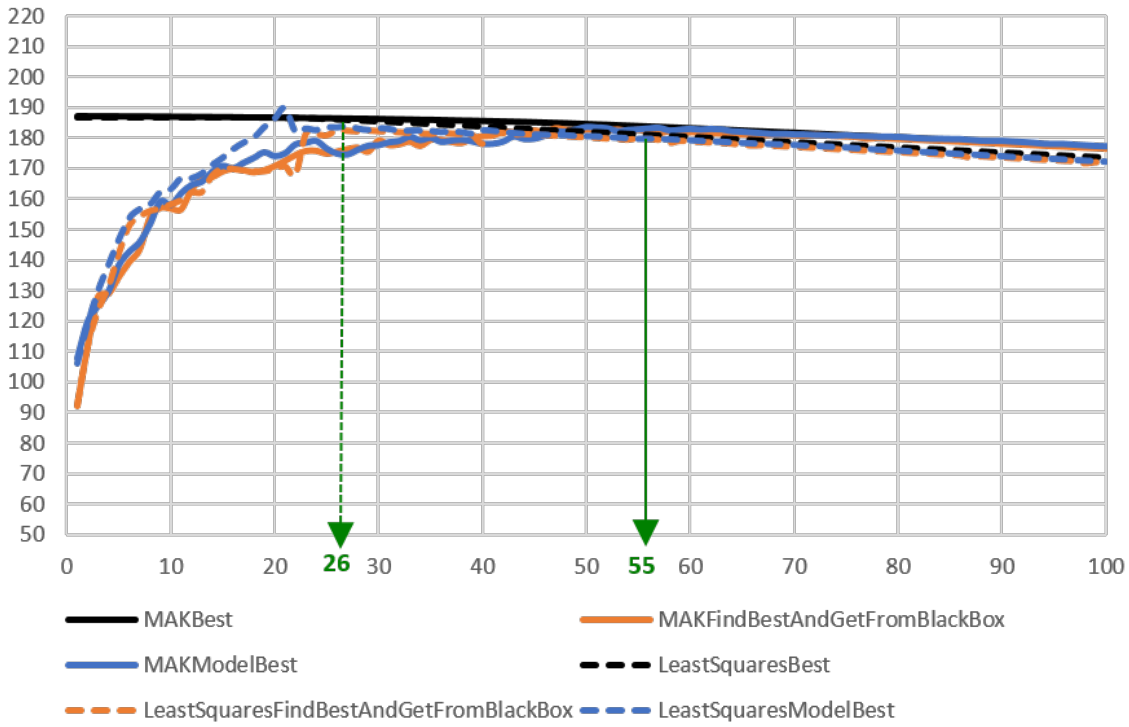


Figure 5.11: Comparison of MAK versus Least Squares For Degradation audience model with mean value initialization.

unable to find the best sequence using Least Squares. MAK stops when the stopping condition is met, given that the first condition is that the maximum iterations MI $= 500$, MAK did not stop because of the first condition, but due to the second condition that $\epsilon = 0.1$. Table 5.5 shows that MAK stops approximately after 70-76 iterations for the Constant and Degradation audience models, whereas Least Squares stops approximately after 33-39 iterations. We show in Figures 5.7-5.12 that MAK has already converged approximately after 46-69 iterations and Least Squares converged approximately after 24-39 iterations. The stopping conditions can be varied so that MAK can stop earlier given that MAK has already converged approximately after 33-39 iterations, whereas with the current stopping conditions, MAK stops approximately after 70-76 iterations. We also observe that by changing the initialization of the model, convergence occurs at different iterations. We cannot conclude that a particular initialization is the best way since convergence occurs at different rates for different initializations in different audience models.

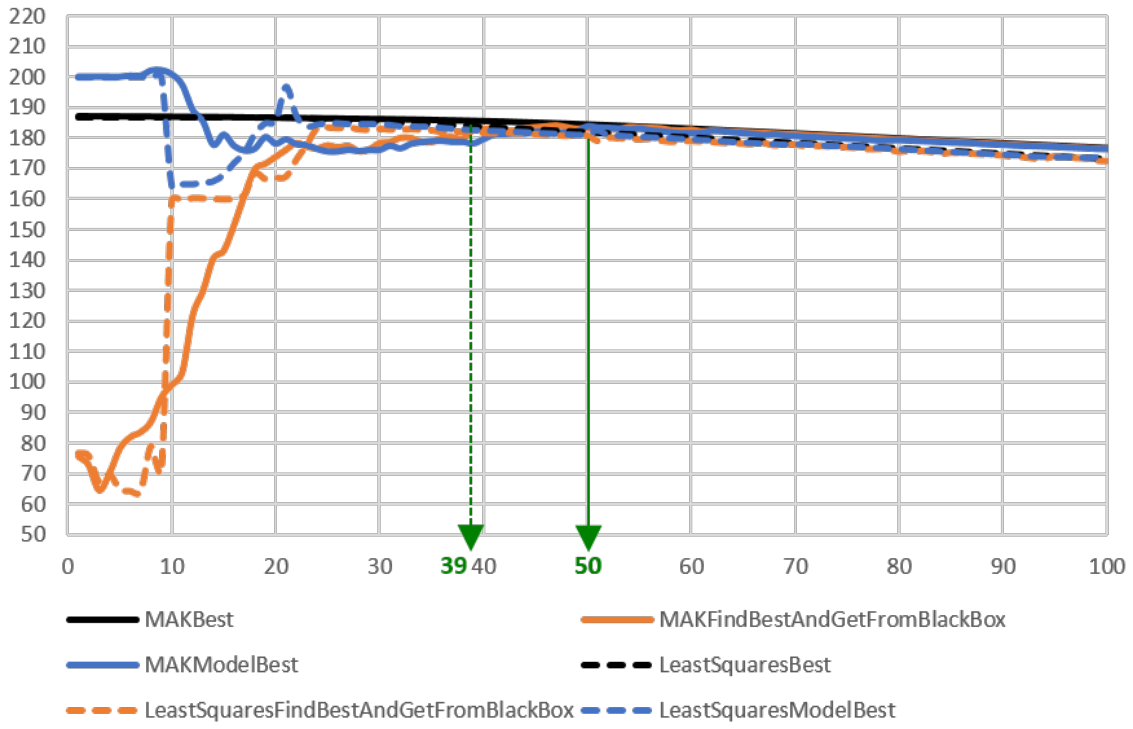


Figure 5.12: Comparison of MAK versus Least Squares For Degradation audience model with maximum value initialization.

5.4 Chapter Summary

This chapter presents our approach to probabilistically select relevant motions based on the similarity between the emotional labels assigned to the motion and the emotional label of the music (input signal). We also explain how we synchronize the motions to the beats of the music.

We also show how we select relevant motions based on the similarity between semantic labels assigned to motions and the labels in the text-to-speech (a sentence from a story). We also describe how we synchronize the motions to the starting times of the labels in the text-to-speech. We illustrate how we discard invalid sequences and rank the sequences with an example.

Lastly, we show how we determine the sequences to get feedback from, learn the audience preferences of the individual motions and determine the best sequence of motions with the highest audience rating. We also demonstrate how we consider the effect of 'boredom' when the audience views the same motion repeatedly.

Table 5.5: Performance of MAK versus Least Squares for two labels in the input signal.

Audience Model	**Initialization**	**Approach**	**Iterations**	Υ	ρ
Constant	Minimum value - 0	MAK	70.3 ± 6.7	2.0 ± 1.4	0 ± 0
		Least Squares	33.2 ± 6.5	1.9 ± 1.2	3.8 ± 5.9
	Mean value - 50	MAK	74.7 ± 8.6	2.4 ± 1.6	0 ± 0
		Least Squares	33.0 ± 6.5	3.4 ± 2.6	2.6 ± 5.3
	Maximum value - 100	MAK	71.6 ± 7.3	1.8 ± 1.3	0 ± 0
		Least Squares	34.1 ± 6.0	3.1 ± 2.4	1.3 ± 5.5
Degradation	Minimum value - 0	MAK	70.0 ± 7.7	2.8 ± 1.9	0.2 ± 0.0
		Least Squares	38.4 ± 10.7	2.2 ± 2.0	2.8 ± 7.0
	Mean value - 50	MAK	76.0 ± 6.3	2.0 ± 1.6	0.2 ± 0.0
		Least Squares	38.5 ± 9.1	2.8 ± 1.8	1.6 ± 4.5
	Maximum value - 100	MAK	74.7 ± 7.4	1.7 ± 1.9	0.2 ± 0.0
		Least Squares	38.3 ± 8.8	2.2 ± 3.3	2.8 ± 6.1

Table 5.6: Performance of MAK versus Least Squares for three labels in the input signal.

Audience Model	**Initialization**	**Approach**	**Iterations**	Υ	ρ
Constant	Minimum value - 0	MAK	107.3 ± 9.2	2.0 ± 1.8	0 ± 0
		Least Squares	44.9 ± 6.4	2.8 ± 2.4	6.8 ± 10.8
	Mean value - 50	MAK	111.0 ± 6.9	2.7 ± 2.1	0 ± 0
		Least Squares	47.1 ± 9.7	2.8 ± 2.2	3.9 ± 7.2
	Maximum value - 100	MAK	109.4 ± 7.8	2.3 ± 1.7	0 ± 0
		Least Squares	44.6 ± 7.8	2.9 ± 2.6	4.9 ± 8.4
Degradation	Minimum value - 0	MAK	115.0 ± 11.7	1.8 ± 1.9	0.6 ± 2.1
		Least Squares	68.3 ± 32.1	1.6 ± 1.4	2.0 ± 5.4
	Mean value - 50	MAK	115.7 ± 9.0	1.9 ± 1.2	0.3 ± 0.0
		Least Squares	65.9 ± 22.9	3.1 ± 4.4	6.0 ± 9.3
	Maximum value - 100	MAK	115.9 ± 6.6	1.7 ± 1.5	0.3 ± 0.0
		Least Squares	70.7 ± 26.7	1.6 ± 2.3	3.7 ± 7.4

Table 5.7: Performance of MAK versus Least Squares for four labels in the input signal.

Audience Model	**Initialization**	**Approach**	**Iterations**	Υ	ρ
Constant	Minimum value - 0	MAK	162.3 ± 12.3	2.0 ± 1.6	0 ± 0
		Least Squares	56.5 ± 7.1	2.3 ± 2.0	3.9 ± 6.1
	Mean value - 50	MAK	164.4 ± 10.1	2.1 ± 1.5	0 ± 0
		Least Squares	54.8 ± 7.9	2.0 ± 1.8	4.2 ± 8.3
	Maximum value - 100	MAK	165.1 ± 10.4	2.2 ± 1.5	0 ± 0
		Least Squares	57.6 ± 6.8	2.1 ± 1.7	4.0 ± 8.6
Degradation	Minimum value - 0	MAK	172.4 ± 16.4	2.3 ± 1.8	1.1 ± 0.8
		Least Squares	81.1 ± 32.2	1.6 ± 0.9	4.0 ± 4.8
	Mean value - 50	MAK	175.1 ± 16.6	2.4 ± 3.1	1.4 ± 2.3
		Least Squares	76.9 ± 33.1	2.4 ± 2.6	7.8 ± 10.5
	Maximum value - 100	MAK	169.6 ± 13.8	2.2 ± 2.2	1.0 ± 0.8
		Least Squares	82.6 ± 30.7	1.9 ± 1.3	4.9 ± 7.0

Chapter 6

Stability

Stability of the humanoid robot is vital for the robot to animate an input signal. Moreover, if a humanoid robot falls, the humanoid robot may get damaged due to the fall, or even break. Therefore, we are interested to determine the stability of a robot given the sequence of motion primitives the robot is to execute. Although each motion in the motion library is stable, a sequence of motions may not be. We assume that there is no model of the dynamics of the robot, so that our approach does not depend on the accuracy of the robot's model. Also, there is no prior execution of the sequence of motions and we do not execute the sequence of motions to determine if the robot will fall. We discuss our approach – ProFeaSM – in Section 6.1.

Next, we aim to determine the most stable sequence from a list of possible motion sequences for an input signal. By executing the most stable sequenc, we increase the probability that the robot continues to remain stable and increase the tolerance for errors in the prediction of the stability of sequences. Thus, we investigate the problem of determining the relative stability of sequences of motion primitives. We also do not require a model of the robot, but we possess data of prior executions of different sequences of motions. We describe our approach – RS-MDP – in Section 6.2.

6.1 Predicting the Stability of a Motion Sequence with No Prior Execution

In this section, we present our approach to predict the stability of a sequence of motion primitives. First, we list the assumptions. Next, we explain the data we collect and the algorithms we use to predict the stability of a sequence of motion primitives. Lastly, we describe the experiments and explain the results.

Assumptions

We assume the following:

- The model of the robot is not available.
- There is no prior execution of any sequence of motions primitives.
- Each motion in the motion library is stable.
- Data are collected using the executions of the single motions and interpolations between pairs of motions on the humanoid robot that is used to animate the motion sequence. The data collected are the body angles X (roll) and body angles Y (pitch) sensor readings via the inertial measurement unit and these sensor readings are available.
- We predict the stability of a motion sequence that is executed on the same humanoid robot used to collect the data.
- There is no wear and tear on the humanoid robot.
- Every motion in the motion library is stable.
- The humanoid robot starts each motion sequence with the same keyframe.

Description of Data Collected

We record the body angles of the robot via the inertial measurement unit of the NAO during the execution of each motion primitive in the motion library and the interpolations between pairs of motion primitives. We record the body angles at a regular frequency, `f`.

Definition 6.1.1. *Let Ψ be the number of body angles collected when a motion primitive m is*

executed. The Ψ body angles comprise $(\psi_0, ba_0), \ldots, (\psi_\Psi, ba_\Psi)$, where ψ_i is the timestamp, ba_i are the body angles at time ψ_i. ba_0 are the body angles of the robot before executing m. ba_i is made up of the body angle readings, X and Y, per time step.

We determine Ψ using the duration of the execution, dt, i.e., $\Psi = (\text{dt} \times \texttt{f}) + 1$. We add one more time step as we also collect the body angles of the robot's initial pose for one time step before m is executed.

We collect the body angles of three groups of motion executions and term each group as:

1. single: We execute each motion primitive from the motion library individually. The robot always begins with the same initial pose shown in Figure 6.1 at the start of each execution. For the body angles collected for the motion primitive m_i in single, we denote the body angles as single_{m_i}.
2. startSingle: We begin the execution of each motion primitive with the first keyframe of the motion primitive as the robot's pose, and not the initial pose shown in Figure 6.1. For the body angles collected for the motion primitive m_i in startSingle, we denote the body angles as startSingle_{m_i}.
3. interpolation: First, we determine all possible pairs of motion primitives, m_i and m_j in the motion library. Then, we execute the interpolation between each pair and collect the body angles of the interpolation. The interpolation between two motion primitives, m_i and m_j, is executed from the last keyframe k_n of the first motion primitive m_i to the first keyframe k_1 of the second motion primitive m_j. For the body angles collected during the interpolation between the two motion primitives, m_i and m_j in interpolation, we denote the body angles as $\text{interpolation}_{m_i,m_j}$.

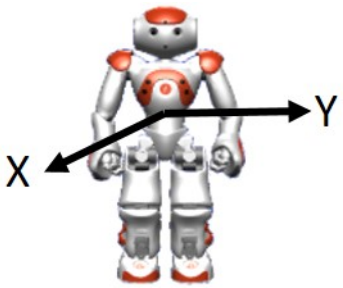


Figure 6.1: NAO's initial pose and coordinate frame of the inertial measurement unit.

For each group of executions, we collect ς iterations. In the next section, we explain how our approach, ProFeaSM, uses these three groups of body angles to predict if a particular sequence of motion primitives, u, will cause the robot to fall.

6.1.1 Approach – ProFeaSM

Algorithm 11 – ProFeaSM – is made up of four algorithms, namely Process, Feasibility, Stitch and Multiplier (Algorithms 12-15). Lines 5-7 of Algorithm 11 use Process (Algorithm 12) to process the body angles of the three groups of executions when we collect more than one iteration of the three groups of executions. Algorithm 12 determines the median of the body angle trajectories collected. Algorithm 12 processes the ς iterations and determines the median for body angle X trajectory, $\overline{\text{bax}}$, and body angle Y trajectory, $\overline{\text{bay}}$. Algorithm 12 returns the median of body angles trajectories, given ba, a list of D body angle trajectories. Therefore, $\overline{\text{bax}} = \text{Process}(\text{bax})$, where $\overline{\text{bax}}$ is the median of body angle X trajectories and bax contains the ς body angle X trajectories.

Algorithm 11 ProFeaSM: Process-Feasibility-Stitch-Multiplier [Tay et al., 2016].

ProFeaSM(u, inertialMultiplier)

```
1:  (ς, Ψ) ← size(single)
2:  if ς == 1 then
3:      hasFallen ← 𝔽(u, inertialMultiplier)
4:  else
5:      single ← Process(single)
6:      startSingle ← Process(startSingle)
7:      interpolation ← Process(interpolation)
8:      hasFallen ← 𝔽(u, inertialMultiplier)
9:  end if
10: return hasFallen
```

Line 8 of Algorithm 11 uses Algorithm 13, $\mathbb{F}$, which uses the median body angle trajectories to predict if a sequence of motion primitives, u, will fall. $\mathbb{F}$ has two parameters, u and inertialMultiplier.

Definition 6.1.2. *The algorithm $\mathbb{F} : U \times \mathbb{R} \rightarrow \{0, 1\}$ computes the **feasibility** of a sequence of motion primitives, where a sequence of motion primitives is **feasible** if and only if the robot*

Algorithm 12 Process ς iterations of Ψ time steps [Tay et al., 2016].

Process(ba)

1: // ba is a $\varsigma \times \Psi$ matrix, containing ς iterations with Ψ time steps
2: **for** $i = 1$ **to** Ψ **do**
3: medianAtEachStep(i) = median$^{\varsigma}_{j=1}$(ba(j, i)) // finds median at time step i
4: **end for**
5: medianBA = argmin$^{\varsigma}_{j=1}(\sum_{i=1}^{\Psi}$ |ba(j, i) − medianAtEachStep(i)|)
6: **return** medianBA

*executes the keyframes and continues to remain **stable**. Hence,* $\mathbb{F}(u,$ *inertialMultiplier*$) = 1$ *when* u *is feasible.*

Algorithm 13 uses the fact that when the acceleration of the body angles increases, the velocity increases and vice versa. The velocity of the body angles reaches a constant when the acceleration of the body angles approaches zero. The body angle is the angle of the robot's torso with respect to the ground. Hence, the higher the body angle, the higher the probability that the robot is going to fall. We calculate the velocity, vel by determining the change in body angles at each time step. We calculate the acceleration, acc by determining the change in vel at each time step. We model the velocity as an exponential curve in Algorithm 13 since the velocity does not increase linearly due to the effects of gravity, inertia and momentum. The x-value of the exponential curve is termed as stepMultiplier and is affected by the acceleration, acc.

Algorithm 14 is called in Line 2 of Algorithm 13 and stitches up the body angle values collected. Algorithm 14 begins with the original body angle trajectory of the first motion primitive in the sequence, single$_{m_1}$ since the body angle trajectory is the same. Next, Algorithm 14 determines the change in body angles at each time step and adds each change to the last known body angle. Algorithm 14 continues adding the changes in body angles for the rest of the sequence by using the body angles collected for interpolation and startSingle.

Lines 3 and 4 in Algorithm 13 determine the velocity of the body angles, vel, and the acceleration, acc. We predict the body angle trajectory, predictTraj so as to determine the stability of a motion sequence. We start with the body angles collected from single$_{m_1}$ since the body angles should be similar to executing the motion primitive single$_{m_1}$ that starts from the initial pose. We determine the stepMultiplier using Algorithm 15 since the inertial and momentum change in Lines 7-9 of Algorithm 13. As we construct the predicted body angle trajectory, predictTraj, we determine the velocity using the exponential velocity curve and stepMultiplier so as to determine

Algorithm 13 Predict whether a sequence of motion primitives is feasible [Tay et al., 2016].

$\mathbb{F}(u$, inertialMultiplier)

// Indices start from 1
data $\leftarrow$ Stitch(u)
vel $\leftarrow (0, \text{data}(2) - \text{data}(1), \text{data}(3) - \text{data}(2), \ldots)$
acc $\leftarrow (0, \text{vel}(2) - \text{vel}(1), \text{vel}(3) - \text{vel}(2), \ldots)$
predictTraj $\leftarrow$ single$_{m_1}$
stepMultiplier $\leftarrow 0$ // initialized as 0 as $e^0 = 1$
for $i = 1$ **to** |single$_{m_1}$| **do**
    stepMultiplier $\leftarrow$ Multiplier(acc(i), stepMultiplier, inertialMultiplier)
end for
hasFallen $\leftarrow$ false
for $i =$ |single$_{m_1}$| $+ 1$ **to** |data| **do**
    predictAngle $=$ vel$(i) \times \exp($stepMultiplier$)$ + predictTraj$(i-1)$
    stepMultiplier $\leftarrow$ Multiplier(acc(i), stepMultiplier, inertialMultiplier)
    predictTraj $\leftarrow$ append(predictTraj, predictAngle)
    if |predictAngle| $>$ fallenThresh **then**
        hasFallen $\leftarrow$ true
    end if
end for
return hasFallen

Algorithm 14 Stitch collected data into a trajectory [Tay et al., 2016].

Stitch(u)

data $\leftarrow$ single$_{m_1}$
lastAngle $\leftarrow$ single$_{m_1}$(|single$_{m_1}$|)
for $l = 2$ **to** L **do**
    for $i = 2$ **to** |interpolation$_{m_{l-1},m_l}$| **do**
        lastAngle $\leftarrow$ lastAngle + (interpolation$_{m_{l-1},m_l}(i)$ $-$ interpolation$_{m_{l-1},m_l}(i-1)$)
        data $\leftarrow$ append(data, lastAngle)
    end for
    for $i = 2$ **to** |startSingle$_{m_l}$| **do**
        lastAngle $\leftarrow$ lastAngle + (startSingle$_{m_l}(i)$ $-$ startSingle$_{m_l}(i-1)$)
        data $\leftarrow$ append(data, lastAngle)
    end for
end for
return data

the change to the previous body angle in Line 12 of Algorithm 13. Next, stepMultiplier changes in Line 13 of Algorithm 13 and appends the predicted body angle, predictAngle, to predictTraj. If predictAngle exceeds the threshold, fallenThresh, the robot is deemed to have fallen in Lines 15-17 of Algorithm 13.

Algorithm 15 determines how stepMultiplier varies along the exponential velocity curve. As the acceleration acc per time step is small, inertialMultiplier is used as a multiplier to acc, and varies how stepMultiplier changes in Line 4 of Algorithm 15. accThres is used as a threshold to determine if the acceleration approaches zero and if so, stepMultiplierDec is used to decrease stepMultiplier in Lines 1-2 of Algorithm 15.

Algorithm 15 Determine the step multiplier based on the acceleration [Tay et al., 2016].

Multiplier(acc, stepMultiplier, inertialMultiplier)

1: **if** $|\text{acc}| < \text{accThres}$ **then**
2: $\quad$ stepMultiplier $\leftarrow$ stepMultiplier $-$ stepMultiplierDec
3: **else**
4: $\quad$ stepMultiplier $\leftarrow$ stepMultiplier $+ (\text{acc} \times \text{inertialMultiplier})$
5: **end if**
6: **if** stepMultiplier < 0 **then**
7: $\quad$ stepMultiplier $\leftarrow 0$ **//** stepMultiplier will not go below 0
8: **end if**
9: **return** stepMultiplier

To summarize, Algorithm 11 – ProFeaSM – is made up of Algorithms 12–15. With ς iterations of body angles recorded, we use Algorithm 12 to determine the median of the body angle trajectories collected. Next, we use Algorithm 13 to predict the stability of a sequence of motion primitives. Algorithm 13 uses Algorithm 14 to stitch up the body angles collected from the three groups of executions using their respective velocities of the body angles of the motions and interpolations. Algorithm 13 also uses Algorithm 15 to determine the stepMultiplier for the exponential velocity curve and is used as a multiplier to the velocity.

6.1.2 Experiments

We conduct experiments in simulation using Webots 7 [Webots, 2014] and on a real NAO humanoid robot. Webots 7 [Webots, 2014] is a real-time simulator that simulates the dynamics of the NAO humanoid robot whilst executing a sequence of motion primitives.

We use a motivating example of an autonomous humanoid robot playing a game of charades to guess different emotions. With ς iterations of each group of execution, we collect a total of $\varsigma \times (|M| + |M| + |M|(|M| - 1)) = \varsigma \times |M|(|M| + 1)$ executions, where $|M|$ is the number of motion primitives in the motion primitive library. Since the game of charades is to guess different emotions, pairs of motion primitives will not contain the same motion primitive, i.e., the pair of motion primitives m_i and m_j, where $i \neq j$.

Experiments in Simulation

We simulate a NAO V4.0 H25 humanoid robot and collect body angle values for three groups of executions for $\varsigma = 10$ iterations: $\text{single}, \text{startSingle}, \text{interpolation}$. To check if our prediction of the stability of the sequence of motion primitives is correct, we simulate the robot executing the sequence of motion primitives and determine if the robot remains stable using its body angles.

We assume that every motion primitive in the library is stable and check that the assumption is true by running 10 iterations of the NAO robot executing the same motion primitive in Webots and that the NAO robot remains stable. Webots is restarted each time an iteration is ran to ensure that the NAO robot starts with the same initial pose and position in the environment.

Experiments on the Real NAO

We ran our experiments on a real NAO V3.3 H21 humanoid robot, with a V4.0 head. By using a different model from the NAO V4.0 H25 robot in the simulation, we test if ProFeaSM is applicable to different models. The NAO H25 has 25 degrees of freedom and the NAO H21 has 21 degrees of freedom. The mass of the NAO H25 and the mass of the NAO H21 are also different.

We ran $\varsigma = 1$ iteration to collect the three groups of executions as it is impractical to collect many iterations in reality. At the same time, we evaluate if ProFeaSM works well when $\varsigma = 1$. We also execute each motion primitive from the motion library on the real NAO and check that the robot remains stable after executing a motion primitive. Though each motion primitive in the motion library is stable, the interpolations between the pairs of motion primitives may be unstable.

We also test the stability of every sequence on the real NAO so as to compare our predictions to the actual results. During our experiments, we observe that a safety feature as part of the fall

manager software [Aldebaran Robotics, 2013] of the NAO is often triggered prematurely even though the motion primitive is stable. This safety feature detects a potential fall when we execute whole body motions on the robot and triggers the NAO to put its arms in front of its face before falling forward onto the ground. This safety feature is introduced to brace the NAO's fall and reduces impact to other parts of the body, e.g., the head of the NAO, where the central processing unit is located. Hence, we disable the fall manager so that the robot only executes the intended motions, without disrupting the data collection of body angles.

To avoid damaging the NAO robot when we execute unstable motions, we tie a string around the robot's torso to allow the robot to fall gently so as to prevent the robot from hitting the ground too hard. There are instances when the real NAO falls and we have to stop the execution of the rest of the sequence of motions so that the NAO's joints do not actuate when the NAO is lying flat on the ground. We do not stop the execution of motions during the collection of body angles for the three groups of motion executions – single, startSingle and interpolation. The motions in single and startSingle are stable and only some motions in interpolation fall but the motions are short as compared to the entire sequence of motions.

Experimental Setup

We devise a scenario where the NAO humanoid robot is to play a game of charades to guess emotions. There are three different emotions: angry, sad and surprised. There is no restriction on the order of the emotions being acted out by the NAO huamnoid robot. For every emotion, there are two motion primitives from the motion library that are labeled with the particular emotion. We ensure that the robot is able to stably execute each individual motion primitive.

The number of possible sequences of motions for three different motions (angry, sad, surprised in any order) is $2 \times\ 2\ \times\ 2\ \times\ 3! =\ 48$. For the three groups of executions, we collect a total of $|\text{single}| +\ |\text{startSingle}| + |\text{interpolation}| = 6 + 6 + (6 \times 4) = 36$ body angle trajectories. Since we do not use the two motion primitives labeled with the same emotion consecutively, we do not collect all the $6 \times 5 = 30$ body angle trajectories for interpolation.

We use the interpolation time computation function $\mathcal{T}$ to compute the interpolation time between keyframes. $\mathcal{T}$ uses the maximum joint angular velocity. To ensure that the robot remains stable after executing each motion primitive in the library, the maximum joint angular velocity in the simulation is limited to 70 percent of the real maximum joint angular velocity in simulation

and 40 percent of the real maximum joint angular velocity on the real NAO.

fallenThresh in Algorithm 13 is set as 1.0 based on the empirical data collected when the robot falls and is lying on the ground. accThres and stepMultiplierDec in Algo. 15 is set to 0.005 and 0.001 respectively as the 0.005 is close to 0, and a value of 0.001 only changes the multiplier slightly.

To predict the fall of the sequences on the real NAO, we skip Algorithm 12 since we only collect 1 iteration of body angles for the three groups of executions and use these body angles as the median body angle trajectory. We vary different values of inertialAccMultiplier in Algorithm 15 from 10 to 100.

Each sequence of the motion primitives starts with the same initial pose in Figure 6.1. The body angles are recorded at a frequency of 100 Hz (every 10 milliseconds) using a function provided by the NAO's software [Aldebaran Robotics, 2013] and computed using the accelorometer and gyrometer sensors readings from the inertial measurement unit (IMU) [Aldebaran Robotics, 2013]. The body angles recorded are body angle X (roll) and Y (pitch) as shown in Figure 6.1.

Experimental Results

Table 6.1 shows two sequences of motion primitives: (a) Sad2, Angry2, Surprised1 and (b) Surprised1, Sad2, Angry2. The first row shows the intended sequence of the motion primitives (shown in bold) and the interpolations between motion primitives. "Start-" indicates the interpolation from the initial pose of the robot to the first motion primitive.

Using Table 6.1, we show that even though each motion primitive in the sequence of Surprised1, Sad2 and Angry2 is stable, the sequence of individually stable motion primitives does not guarantee the robot's stability since the sequence results in a fall. We also demonstrate that although the sequence of Surprised1, Sad2 and Angry2 is unstable, a different ordering of the motion primitives, Sad2, Angry2 and Surprised1 is stable. In the sequence of Surprised1, Sad2 and Angry2, we may deduce that the instability of the sequence is attributed to the sub-sequence of Sad2 and Angry2, but the sub-sequence from Sad2 to Angry2 in the sequence of Sad2, Angry2 and Surprised1 is stable. Hence, we do not predict the fall of the robot based solely on part of the sequence, but we have to consider the entire sequence.

From the results of the experiments, we observed that body angle Y values is sufficient for predicting the stability of the robot, since the robot only falls forward or backward and never

Table 6.1: Intended and actual execution showing two motion sequences [Tay et al., 2016].

Intended	Start-Sad2	**Sad2**	Sad2-Angry2	**Angry2**	Angry2-Surprised1	**Surprised1**
Actual	Start-Sad2	**Sad2**	Sad2-Angry2	**Angry2**	Angry2-Surprised1	**Surprised1**
Intended	Start-Surprised1	**Surprised1**	Surprised1-Sad2	**Sad2**	Sad2-Angry2	**Angry2**
Actual	Start-Surprised1	**Surprised1**	Surprised1-Sad2	***Fallen***	***Fallen***	***Fallen***

sideways. Hence, we present results regarding body angle Y values since we only use body angle Y values to predict if a sequence will fall.

Figure 6.2 shows the body angle Y values of the execution of the sequence, Sad2, Angry2 and Surprised1 over time, and Figure 6.3 shows the body angle Y values of the execution of the sequence, Surprised1, Sad2 and Angry2 over time. Figure 6.2 shows that the sequence, Sad2, Angry2 and Surprised1 is stable whereas Figure 6.3 shows that the sequence, Surprised1, Sad2 and Angry2 is unstable. Both sequences are executed in simulation and the prediction of the body angle trajectories are made from the body angles collected in simulation.

Both figures in Figure 6.2 and Figure 6.3 show the plots of three body angle Y trajectories. We use a value of 90 for inertialAccMultiplier. First, we plot the body angle Y trajectory in black with a line style of $-\cdot-$ and term this plot **Actual**. The body angle Y trajectory, **Actual**, was collected during the actual execution of the sequence. Next, we plot a stitched body angle Y trajectory in blue with a line style of $--$ using only Algorithm 14 and term it **Stitched**. Lastly, we plot the body angle Y trajectory that we predicted using Algorithm 12-15 in red with a line style of **—** and term it **Predicted**.

In Figure 6.2, the actual, stitched, and predicted body angle Y trajectories are similar. However in Figure 6.3, we show that the predicted body angle Y trajectory is similar to the actual body angle Y trajectory, while the stitched body angle Y trajectory is not. Thus, we do not simply stitch up body angles collected. We demonstrate that the algorithm works well in predicting the body angle trajectory given that the curvature of the predicted body angle trajectory is similar to the actual body angle trajectory.

We refer to the use of precision and recall for classification tasks to determine the accuracy of the prediction on the stability of a sequence [Russell and Norvig, 2003]. Similar to the classification tasks, we want to classify sequences that are unstable as falls. Precision is the number of true positives (sequences that we label as falls and will actually fall during the execution) divided by the sum of true positives and false positives (sequences that we label as falls but will not fall

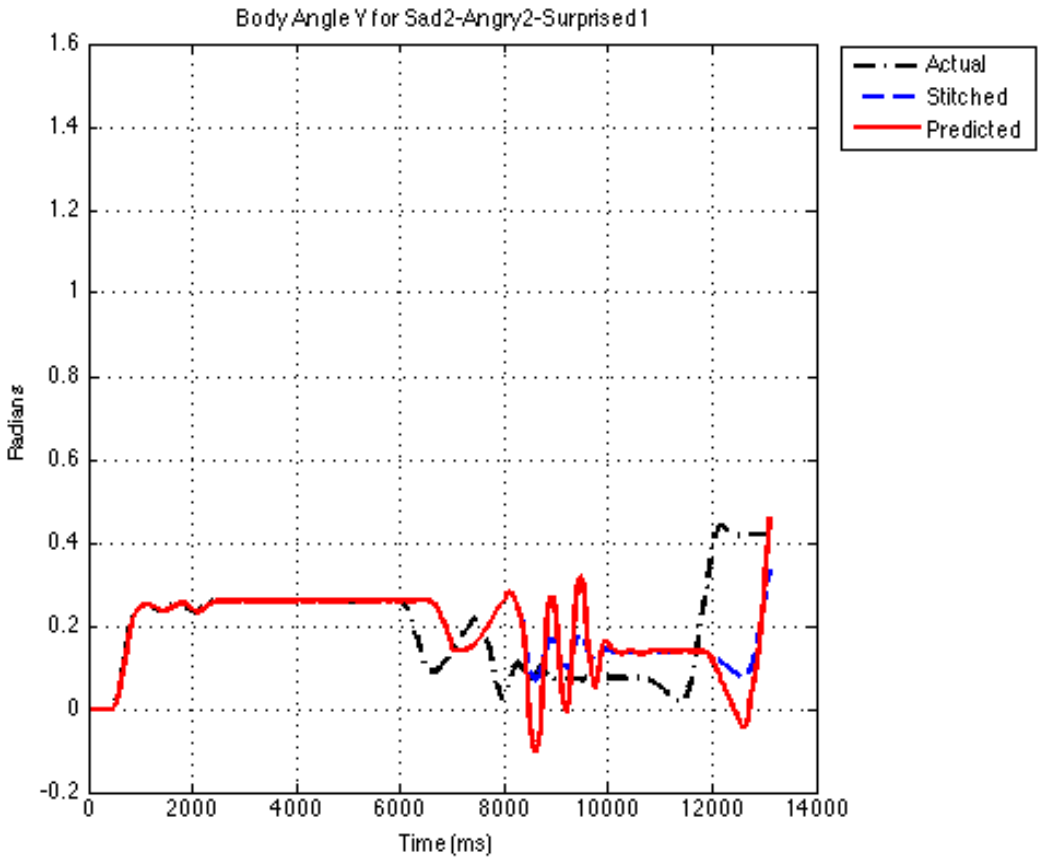


Figure 6.2: Body angle Y values for Surprised1-Sad2-Angry2 [Tay et al., 2016].

during the execution). A perfect precision score of 1.0 means that every sequence that the algorithm labeled as a fall actually did fall during the execution of the sequence. Recall is the number of true positives divided by the total number of sequences that actually fall during the execution. A perfect recall score of 1.0 means that every sequence that actually fell during the execution is labeled as a fall by the algorithm, but does not consider sequences that are wrongly labeled as falls. Precision and recall have an inverse relationship whereby increasing one decreases the other. We aim to have as high a precision and recall as possible, but it is very difficult to achieve both precision and recall at a perfect score.

Figure 6.4 shows two curves, one for simulation and one for the real robot. We vary the parameter, inertialAccMultiplier, from 60 to 100, to determine if the accuracy represented by

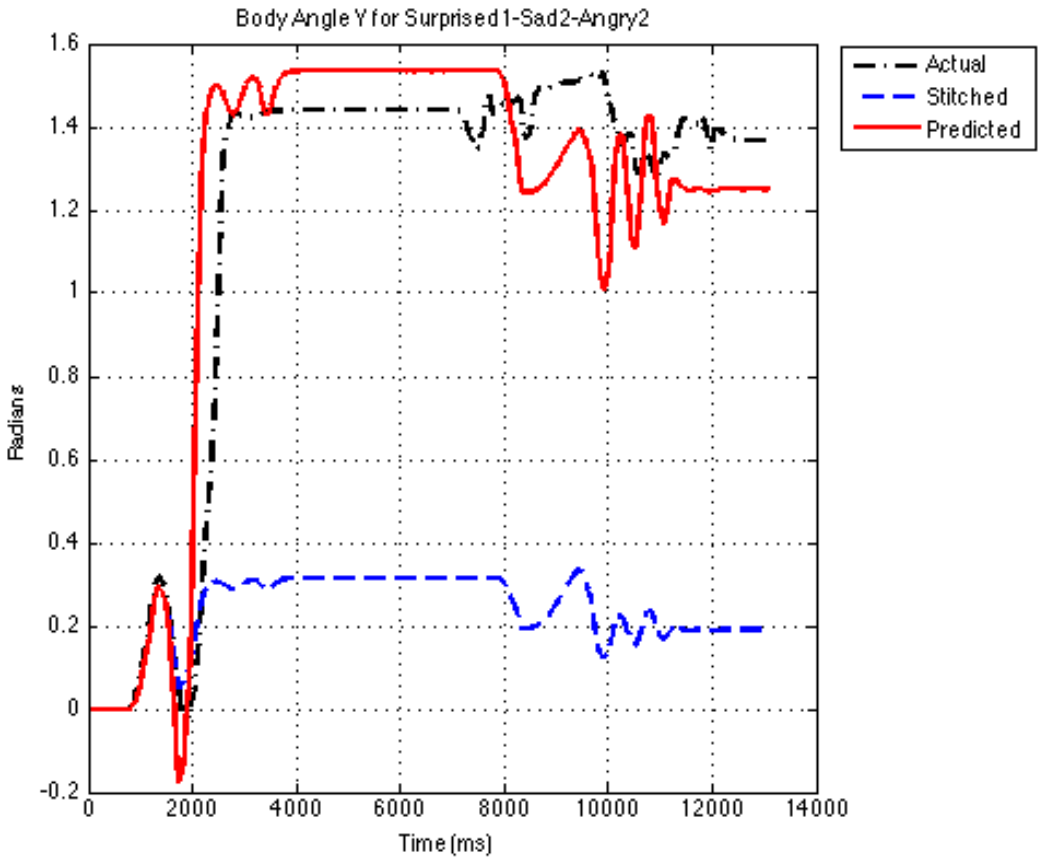


Figure 6.3: Body angle Y values for Sad2-Angry2-Surprised1 [Tay et al., 2016].

precision and recall is improved. The two curves use two sets of body angle Y values – the simulated data and the actual robot data. We do not plot values of 10 to 50 as there are no sequences that are predicted as falls. Each curve is marked with the value for inertialAccMultiplier. The blue line for Simulation shows the precision and recall rate of ProFeaSM which uses the body angle Y values collected in simulation and by varying inertialAccMultiplier. The red line for Robot shows the precision and recall rate of ProFeaSM which uses the body angle Y values collected on the real robot and by varying inertialAccMultiplier.

From the Simulation results, 90 is a value to be used for the inertialAccMultiplier if we want to ensure that all sequences that will fall will be predicted as falls (a perfect recall value of 1.0), but we have a low precision of 0.72, which means that we have predicted some false positives

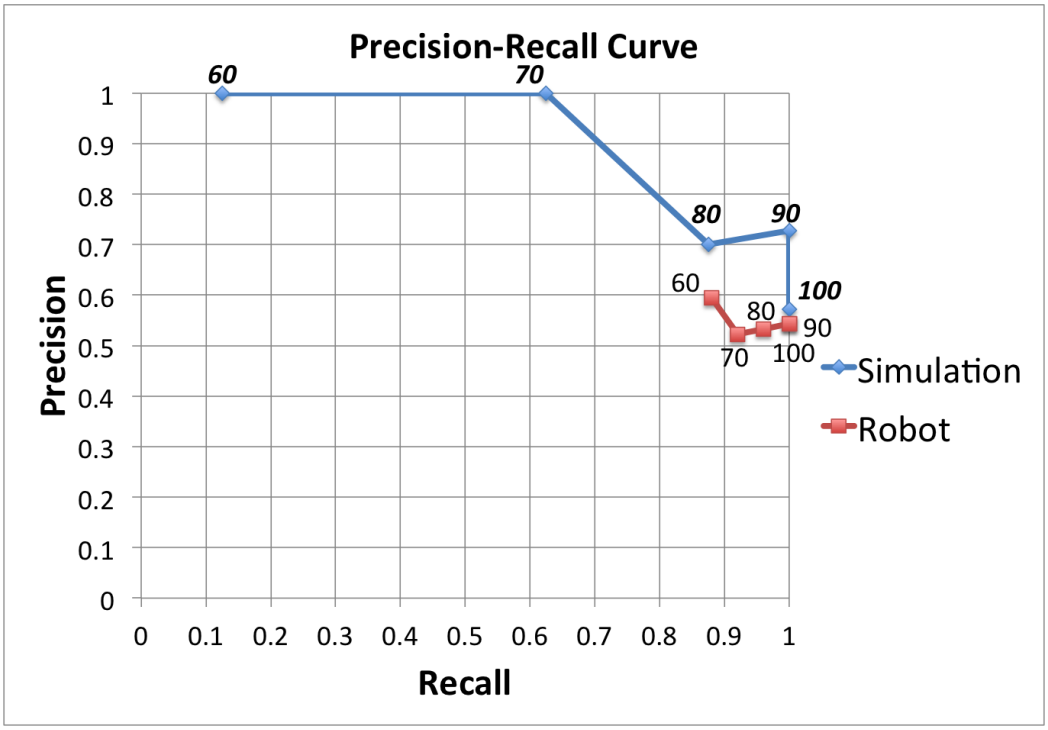


Figure 6.4: Precision-Recall curve [Tay et al., 2016].

(sequences that we predict as falls did not fall). If we use the same value of 90 for the actual robot prediction, we will also achieve a perfect recall value of 1.0, but the precision value is lower at 0.54. This means that we have quite a high number of false positives, which may not be desirable since we have less choices of sequences to execute. Hence, there is a trade-off between precision and recall depending on the requirements. When we require a high precision, the false positives are minimized and we have more sequences to choose from. When we require a high recall instead of precision, the true positives are maximized and we avoid sequences that fall and we have less sequences to choose from.

Discussion

ProFeaSM scales quadratically with the number of motion primitives in the motion primitive library. We reduce the number of times the body angle Y values is recorded for ProFeaSM if the interpolations between pairs of motion primitives are not unique, e.g., the last keyframe of the motion primitive m_1 and the first keyframe of the next motion primitive m_2, are the same two

keyframes for the last keyframe of the motion primitive m_3 and the first keyframe of the next motion primitive m_4.

We contribute an algorithm, ProFeaSM, to predict the stability of the robot executing a sequence of motions. We use only the body angle Y values collected from the executions of single motions and the interpolations between pairs of single motions as the robot only falls in the pitch direction (forward or backwards). Body angles are computed using the IMU comprising accelerometer and gyroscope sensors. These sensors are commonly found in humanoid robots. Compared to traditional fall prediction methods, we do not require training instances of body angle Y values collected from sequences of motions to make predictions of the sequences. Moreover, traditional fall prediction methods only predict possible falls whilst monitoring the execution of the sequence. We make predictions before any sequence of motions is executed on the robot. We also require no model to determine the dynamics of the robot and the environment to make a prediction. We collect data in the real environment and use those data to predict the stability of a sequence of motions.

ProFeaSM includes the parameter, inertialAccMultiplier, that is varied to achieve different precision and recall values. ProFeaSM uses the body angles collected from the executions of all the single motion primitives in the motion primitive library of 6 motion primitives and the 24 interpolations between pairs of the motion primitives in the experiments. We conduct experiments in simulation and test the efficacy of ProFeaSM. We show that ProFeaSM achieves a perfect recall value of 1 and a precision value of 0.72 at inertialAccMultiplier $= 90$ in simulation. We also conduct experiments on the real robot. We show that by using the same value of inertialAccMultiplier $= 90$, ProFeaSM achieves the same perfect recall score and predicts all the sequences that fall, albeit at a poorer precision value of 0.54. By varying different values of inertialAccMultiplier, we achieve different precision and recall values. We explain the trade-off of having a higher recall value versus a higher precision value.

We show ProFeaSM working in simulation and on the real robot. The robot in simulation is a NAO V4.0 H25 humanoid robot with 25 degrees of freedom, and for the real robot, we use a NAO V3.3 H21 humanoid robot with a V4.0 head that has only 21 degrees of freedom. Despite the differences between the simulated robot and the real robot in the number of degrees of freedom, the weight of the robot and the interpolation time between keyframes, ProFeaSM still achieves the same recall value and predicts all the unstable sequences without executing the sequences.

6.2 Predicting Relative Stability of Motion Sequences using Prior Executions

Motions executed by robots are used to fulfill a particular task, for example, a sequence of motions is used by humanoid robots as gestures to communicate during the human-robot interaction such as for storytelling [Ng et al., 2010, Tay and Veloso, 2012] or to dance [Xia et al., 2012]. The stability of the robot is vital for the successful completion of a task. Existing approaches generally determine if a sequence is stable or unstable, and filter out the unstable motions, before determining the best sequence from the remaining motions.

The best sequence of motions is selected from many possible sequences when a robot plans the sequence to execute for a task. The metric to evaluate the best sequence is often related to the completion of the task, for example in the case of a storytelling robot, the robot selects a sequence based on how well the sequence of motions conveys the meanings of the story, or if the robot completes the sequence of motions according to the time constraints of the task. Multiple criteria are used to select the best sequence. In this section, we investigate a single criterion of stability – how to predict the relative stability of sequences so as to select the most stable sequence out of the possible sequences to execute, including new sequences that have not been executed by the robot.

We commonly eliminate sequences that are unstable for a robot and choose any sequence that is stable without determining the most stable sequence. However, it is important to determine the relative stability of motion sequences for the following reasons:

- The more stable the robot is after executing a sequence, the higher the probability that the robot remains stable after executing multiple sequences simultaneously.
- An algorithm that determines the stability of a robot with 100% accuracy has yet to exist given that it is difficult to model environment variables such as ground friction accurately. By selecting the most stable sequence, out of sequences that are deemed to be stable by existing algorithms, increases the chance that the sequence selected is stable. In other words, the margin of error in the algorithms to predict stability is increased.

Therefore, instead of simply determining whether sequences are stable (a binary yes/no decision) and choosing a sequence out of these stable sequences based on other metrics, we investigate how to compute the relative stability of sequences so as to determine the most stable

sequence.

To our knowledge, we are the first to define relative stability of a motion sequence and contribute an approach that evaluates the relative stability of a sequence among a set of possible motion sequences. We will explain how we define relative stability and how our approach predicts the relative stability of a sequence that has not been executed before.

6.2.1 Problem Description

In this section, we describe the motivating scenarios and present the formal problem definition and assumptions.

Motivating Scenario

A humanoid robot is tasked with animating an input signal using its labeled motion library. Examples of input signals are a piece of music or a story. The input signal is labeled, such that there are multiple motions that are applicable for each label in the signal. With multiple applicable motions per label and multiple labels in the input signal, the humanoid robot has to select a sequence from many possible sequences of motions that are synchronized to the input signal. The goal is to select the most stable sequence to execute by comparing the stability of each sequence to the other possible sequences, since we aim to keep the robot as stable as possible. Therefore, relative stability of a sequence is important when we want to determine the best sequence to animate a given input signal and that the robot remains as stable as possible.

Formal Problem Definition

Motions in the motion library are labeled. Each label-motion pair is unique.

Definition 6.2.1. *Let lm be a label-motion pair, and LM be the set of all label-motion pairs in the motion library.*

There exists different sequences of motions for the robot to animate the signal s, where the labels of the signal match the corresponding labels in the motions and the motions are synchronized to the starting times of the labels in the signal.

Definition 6.2.2. *Let $u^s = (lm_1, \ldots, lm_D)$ be an ordered set, i.e., a sequence, of D label-motion pairs for a pre-processed input signal s, where $D \geq 2$ and these label-motion pairs are synchronized to s. Let U^s be the set of all possible sequences of motion for s.*

The goal is to determine the relative stability of the motion sequences in U^s, i.e., a function $\mathbb{U} : U^s \rightarrow \mathbb{R}$ such that $\mathbb{U}(u_i^s) > \mathbb{U}(u_j^s) \Rightarrow u_i^s$ is more stable than u_j^s, for some notion of stability.

Assumptions

We assume the following:

- Sensor data such as accelorometer and gyrometer sensors readings from the inertial measurement unit (IMU) that outputs the body angles of the robot are available.
- The robot starts each sequence with a known initial pose, such as the pose shown in Figure 6.1.
- Data from past executions of all the label-motion pairs in the motion library exist. However, we **do not** assume that there exists data on all the sequences to evaluate. For example, suppose that we are to evaluate the relative stability of three possible sequences for the input signal s – u_1^s, u_2^s, u_3^s. We have the execution data of label-motion pairs in the sequences $u_1^s = (lm_1, lm_2, lm_3)$ and $u_2^s = (lm_2, lm_3, lm_1)$, but have no data on the sequence $u_3^s = (lm_3, lm_1, lm_2)$. However, every label-motion pair (i.e., lm_1, lm_2, lm_3) exists in some sequence in the available execution data.
- We are using the same humanoid robot to collect data and predict the relative stability of a motion sequence executed on the humanoid robot.
- There is no wear and tear on the humanoid robot.

6.2.2 Approach – RS-MDP

To our knowledge, no one has defined relative stability of one sequence to another in a set of sequences of motions. To define relative stability, we first consider the stability of a sequence, which is easier to determine if we observe the robot's state at the end of the sequence, for example whether the robot has fallen after executing the sequence. However, since we are interested in relative stability, the state of the robot should not be only a binary value of whether it is upright or fallen, but expressed as the body angle of the robot so as to determine how stable it is at the

end of a sequence. Figure 6.1 shows the coordinate frame of the IMU that outputs body angle X (roll) and body angle Y (pitch).

Relative stability of a sequence to other sequences is not solely dependent on the final state of the robot at the end of a sequence. Instead, we should also consider the state of the robot throughout the sequence. For example, consider two sequences, u_1^s and u_2^s. If the state of the robot, e.g., the body angle Y (pitch) of the robot is 0, meaning that its body is perfectly upright at the end of the two sequences u_1^s and u_2^s, it does not necessarily imply that both sequences are equally stable. Instead, if we consider that the body angle Y of the robot in the middle of sequence u_1^s is 20° (the robot is leaning forward), whereas the body angle Y of the robot in the middle of sequence u_2^s is 30° (the robot leans even more), we surmise that u_2^s is less stable than u_1^s given that it is more likely to fall in u_2^s than u_1^s during execution.

With this example, one may think that one simply considers the maximum body angle Y of each sequence. This approach is not feasible for the following reasons:

- To determine the maximum body angle Y throughout a sequence, we would need to collect the body angles at every time step to determine the maximum body angle.
- Extracting only the maximum body angle ignores the aspect of time, where a longer sequence may have a higher or the same maximum body angle Y, but is treated as less or equally stable.

We present an approach that takes into account the body angles (stability) at the end of each motion in the sequence, so that we do not have to store sensor data at every time step, and using the body angles at the end of each motion is a good approximation to the relative stability of the sequence.

Using this approach, we also compare the relative stability for two sequences that are unstable at the end of the sequence. For example, consider two sequences u_3^s and u_4^s, where u_3^s has 3 motions and u_4^s has 2 motions. The body angles Y at the end of each motion in u_3^s are $20^\circ, 15^\circ, 90^\circ$ respectively, and $20^\circ, 90^\circ$ for u_4^s. u_3^s is more stable than u_4^s as u_3^s remains stable for a longer period of time.

We build upon the existing approach of modeling the stability of a humanoid robot as an inverted pendulum and research done on using a MDP and reinforcement learning to keep the inverted pendulum upright. We make use of the Markov property, where the next state of the robot depends only upon its present state and not on the sequence of motions that precede it.

With the Markov property, we predict the next state given that the current state of the robot and the next motion to execute.

We form a Markov decision process (MDP) based on the previous executions of the sequences of motions.

Definition 6.2.3. *Let the set of previous sequences of motions executed be U^H and the i^{th} sequence in U^H be u_i^H.*

U^H is not necessarily a subset of U^s as we construct our MDP using sequences for multiple signals. We define a MDP and explain how we form the MDP in Algorithm 16.

Definition 6.2.4. *A MDP consists of the following:*

- *A finite set of states, BAS, where the state is represented by the body angles of the robot.*
- *A finite set of actions, A, where an action is a motion. We will use motions and actions interchangeably from here on.*
- *Transition probabilities, i.e., $TP(bas, a, bas')$ is the probability that state bas will lead to the next state bas$'$ after taking action a, where $a \in A$.*
- *Reward function, RF(bas, a, bas') is the reward received by transitioning from state bas after action a to bas$'$. The reward function is based on the state of the robot, in this case, the body angles of the robot.*

Algorithm 16 Form the MDP using past sequences that were executed.

MDP(U^H)
  $\forall_{\text{bas}\in\text{BAS}}\forall_{a\in\text{A}}\forall_{\text{bas}'\in\text{BAS}} TC(\text{bas}, a, \text{bas}') = 0$ // Initializes all transition counts to 0
  for $i = 1$ **to** $\mid U^H \mid$ **do**
    $\overrightarrow{V_i} \leftarrow \Omega(u_i^H)$ // retrieve state vectors for the sequence
    bas $\leftarrow \mathbb{D}(\overrightarrow{v_0})$ // discretize state
    for $p = 1$ **to** $\mid \overrightarrow{V_i} \mid$ **do**
      $a \leftarrow$ getAction(u_i^H, p) // get the p^{th} motion in u_i^H
      bas$' \leftarrow \mathbb{D}(\overrightarrow{v_p})$
      $TC(\text{bas}, a, \text{bas}') \leftarrow TC(\text{bas}, a, \text{bas}') + 1$
      bas $\leftarrow$ bas$'$
    end for
  end for
  $\widetilde{TP} \leftarrow$ determineTransitionProbabilities$(\widetilde{TC})$
  return [BAS, A, $\widetilde{TP}$]

The state of the robot at any time is characterized by two parameters, body angle X, bax, and body angle Y, bay. We consider the absolute value of the body angles as we do not differentiate between the robot leaning forward or backward, or the robot leaning towards the left versus the right.

Definition 6.2.5. *Let the state vector* $\overrightarrow{v_t} = (|bax_t|, |bay_t|)$ *be the absolute body angles of the robot after executing a label-motion pair* lm_t *in the sequence* u^s.

Definition 6.2.6. *Let* $\Omega(u^s) = (\overrightarrow{v_1}, \ldots, \overrightarrow{v_{|u^s|}})$ *be the function that maps sequences into the respective state vectors after each motion in the sequence, e.g., if* $\Omega(u^s) = \overrightarrow{V_i}$*, then* $\overrightarrow{V}$ *contains the list of state vectors for sequence* u^s_i.

The initial state of the robot is known and is defined as $\overrightarrow{v_0}$. As the body angles are continuous from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, we approximate the state space using a discretization function that maps the state vector $\overrightarrow{v}$ into a number from 1 to the number of states in the MDP.

Definition 6.2.7. *Let the function* $\mathbb{D}(\overrightarrow{v}) \rightarrow \mathbb{Z}_1^{NS}$ *be the mapping from the state vector into a discrete state, where NS is the number of discrete states.*

We introduce two additional actions - Hold and Observe. Hold is an action when the robot does not execute any motions but holds its current pose till the next motion. Observe is an action always taken at the end of the motion sequence for a fixed amount of time. Observe is added as the robot may not immediately fall at the end of the sequence, but may take some time before the robot falls. We use the function getAction(u, p) to determine the p^{th} action in sequence u.

We determine the transition probabilities in the MDP by counting the number of times the robot's state starts from bas after taking action a to another state, bas$'$. The sum of transition probabilities from state bas is $\sum_{\text{bas}' \in \text{BAS}} \sum_{a \in \text{A}} TP(\text{bas}, a, \text{bas}') = 1$.

Definition 6.2.8. *Let* $TC(bas, a, bas')$ *be the transition count – the number of times the robot's state transitions from bas to bas*$'$ *after taking action* a*. Let* $\widetilde{TC}$ *be the set of all the transition counts. Let* $TP(bas, a, bas')$ *be the transition probability that the robot's state transitions from bas to bas*$'$ *after taking action* a *where:*

$$TP(bas, a, bas') = \frac{TC(bas, a, bas')}{\sum_{bas'' \in BAS} TC(bas, a, bas'')}$$

Let $\widetilde{TP}$ *be the set of all transition probabilities.*

We use the function determineTransitionProbabilities that uses the set of the transition counts $\widetilde{TC}$ to calculate the transition probabilities $\widetilde{TP}$ using the equation for $TP(\text{bas}, a, \text{bas}')$.

With the formation of the MDP, we predict the relative stability of a sequence using the reward function given that we know that the robot always starts a sequence from the same initial state (determined using the body angles of the initial pose) and traverses the MDP using the known sequence of motions. Our approach is useful in the event that we do not have data of sequences that have not been executed, but possess data of previous executions of the motions in the sequences, e.g., the order of the motions are different in the predicted sequences.

To predict the relative stability of each sequence in U^s, we use Algorithm 17. We calculate the expected reward of the sequence since we know that the robot always starts from the initial state; we are given the reward function RF and the sequence of motions that the robot will execute, and we have computed the transition probabilities in the MDP.

Definition 6.2.9. *Let the expected reward of the sequence u_i^s be er_i. Let the set of expected rewards for the sequences U^s be ER.*

The expected reward is summed up across possible states using the transition probabilities in the MDP and the reward function. We determine the list of possible states BAS_{j+1} after executing action a from the current state bas_j using the function getPossibleNextStates. The expected reward for a sequence u_i is calculated using $\text{er}_i = \sum_{j=1}^{|u|} TP(\text{bas}_{j-1}, a_j, \text{bas}_j) * \text{RF}(\text{bas}_{j-1}, a_j, \text{bas}_j)$.

A longer sequence will accrue a higher expected reward than a shorter sequence. It is also possible that different sequences have different number of motions due to the synchronization of the motions to the input signal. Hence, to ensure a fair comparison, we average the expected reward by dividing the expected reward by the number of actions in the sequence.

Definition 6.2.10. *Let the average expected reward of the sequence u_i^s be aer_i, where $aer_i = \frac{er_i}{|u_i^s|}$ and is the predicted relative stability for the sequence u_i^s. Let the set of predicted relative stability for all the sequences, U^s, be AER for the input signal s.*

Lastly, we normalize the predicted relative stability. Normalizing the expected reward is useful when multiple criteria are used to select the best sequence. Normalizing the expected reward (predicted relative stability) enables us to scale these values to a common scale with other criteria. A value of 0 means that the sequence is the least stable whereas a value of 1 means that the sequence is the most stable. In this case, for our experiments, normalizing is not

Algorithm 17 Evaluates the predicted relative stability in U^s.

$\mathbb{U}(U^s)$
  AER $\leftarrow \emptyset$
  for $i = 1$ **to** $\mid U^s \mid$ **do**
    $\text{er}_i \leftarrow 0$ **//** Init expected reward to 0
    BASP $\leftarrow \{(\mathbb{D}(\overrightarrow{v_0}), 1)\}$
    for $j = 1$ **to** $\mid u_i^s \mid$ **do**
      for $(\text{bas}, p) \in$ BASP **do**
        $a \leftarrow getAction(u_i^s, j)$
        BAS$'$ $\leftarrow$ getPossibleNextStates$(\text{bas}, a, \widetilde{TP})$
        BASP$'$ $\leftarrow \emptyset$
        for bas$'$ $\in$ BAS$'$ **do**
          $\text{er}_i \leftarrow \text{er}_i + p \cdot TP(\text{bas}, a, \text{bas}')\text{RF}(\text{bas}, a, \text{bas}')$
          if stable(bas$'$) **then**
            BASP$'$ $\leftarrow$ BASP$'$ $\cup \{\text{bas}', p \cdot TP(\text{bas}, a, \text{bas}')\}$
          end if
        end for
      end for
      BASP $\leftarrow$ BASP$'$
    end for
    // Reward for last state(s)
    for $(\text{bas}, p) \in$ BASP **do**
      $\text{er}_i \leftarrow \text{er}_i + p \cdot \text{RF}(\text{bas}, \text{NULL}, \text{NULL})$
    end for
    $\text{aer}_i \leftarrow \frac{\text{er}_i}{|u_i^s|}$
    AER $\leftarrow$ AER $\cup \{\text{aer}_i\}$
  end for
  return normalize(AER)

required as we do not compare the difference in value between the actual relative stability and the predicted relative stability, but we determine if the sequence has a higher relative stability than the other sequences using the rankings of the relative stability values.

We have thus presented our RS-MDP approach using Algorithms 16-17. Next, we explain the two benchmarks we created to compare to the performance of RS-MDP.

6.2.3 Comparisons

We create two benchmarks where we assume that the probabilities of an action causing the robot to fall is independent across motions:

1. RightAfter: Using all sequences, we count the number of times the robot falls right after an action a and term this $\text{RA}_a^{\text{fall}}$. We also count the number of times the robot is stable right after this action a and term this $\text{RA}_a^{\text{stable}}$. To calculate the probability of the action being stable for RightAfter, we use the equation $\text{RAP}_a = \frac{\text{RA}_a^{\text{stable}}}{\text{RA}_a^{\text{stable}}+\text{RA}_a^{\text{fall}}}$. We highlight that $(\text{RA}_a^{\text{stable}} + \text{RA}_a^{\text{fall}})$ is not equivalent to the number of times the action appears in all the sequences. This is because we count the number of times the robot falls right after this action a and we ignore the actions after this action a in the unstable sequence. This benchmark assumes that the instability of the robot is credited to the action that was executed just prior to the robot's fall.
2. Anytime: We count the number of times the action a is found in an stable sequence and term this $\text{AT}_a^{\text{stable}}$. We count the number of times an action is used in sequences and term this AT_a. Similar to RightAfter, we do not count the actions after the action that causes the robot to fall in AT_a. For example, if we have a sequence of actions (a_1, a_2, a_3, a_4) and the robot falls after a_2, we do not include the counts of a_3 and a_4 in AT_a. To calculate the probability of being stable for Anytime, we use the equation $\text{ATP}_a = \frac{\text{AT}_a^{\text{stable}}}{\text{AT}_a}$. This benchmark assumes that the stability of the robot is equally credited to actions from the start of the sequence to the end, similarly the instability of the robot is equally credited to actions from the start of the sequence to the action that causes the robot to fall for an unstable sequence.

To determine the probability that a sequence of actions, u_i^s, is stable, we use the equation $\text{RAPS}_i = \prod_{a \in u_i^s} \text{RAP}_a$ for the comparison – RightAfter and $\text{ATPS}_i = \prod_{a \in u_i^s} \text{ATP}_a$ for the comparison – Anytime. Since we compute the average the predicted relative stability predicted by RS-MDP by the number of actions, we also do the same for the two benchmarks by averaging the probabilities with the number of actions in the sequence. We use these probabilities calculated for each sequence to be the predicted relative stability.

Definition 6.2.11. *Let $\widetilde{RAP}_i$ be the average probability for the sequence u_i^s for the comparison – RightAfter. Let $\widetilde{ATP}_i$ be the average probability for the sequence u_i^s for the comparison – Anytime.*

We illustrate the two benchmarks with this example of two sequences to show how the two benchmarks differ. The first sequence $u_1^s = (a_1, a_4, a_4)$ and the status of the robot after each action is (stable, stable, stable). The second sequence $u_2^s = (a_2, a_1, a_4, a_3, a_1)$ and the status of the robot after each action is (stable, stable, stable, unstable, unstable). We list the probabilities for each action in Table 6.2 and show how we derive them.

Table 6.2: Probabilities for each action using RightAfter and Anytime.

Action	**Comparisons**	**Probability**
a_1	RightAfter	2 / 2 = 1
	Anytime	1 / 2 = 0.5
a_2	RightAfter	1 / 1 = 1
	Anytime	0 / 1 = 0
a_3	RightAfter	0 / 1 = 0
	Anytime	0 / 1 = 0
a_4	RightAfter	3 / 3 = 1
	Anytime	2 / 3 = 0.67

We calculate the probabilities for u_1^s and u_2^s for the two comparisons – RightAfter and Anytime in Table 6.3. Using these probabilities, we multiply the probabilities for each action in these two sequences and average the probabilities by the number of actions in each sequence. If we compare the rankings, the two sequences' ranks are the same for these two comparisons.

Table 6.3: Probabilities for two sequences u_1^s and u_2^s.

Sequence	**Comparisons**	
	RightAfter ($\widetilde{RAP}_i$)	Anytime ($\widetilde{ATP}_i$)
u_1^s	(1*1*1) / 3 = 0.67	(0.5*0.67*0.67) / 3 = 0.07
u_2^s	(1*1*1*0*1) / 5 = 0	(0*0.5*0.67*0*0.5) / 5 = 0

6.2.4 Experiments

Our approach – **R**elative **S**tability using a **M**arkov **D**ecision **P**rocess (RS-MDP) – models past executions of sequences of motions with a Markov Decision Process (MDP) and predicts the relative stability of sequences using the constructed MDP. RS-MDP does not require a model

of the robot, compared to existing algorithms that determine the stability of the robot. Since no one has yet explored the concept of relative stability, we compare RS-MDP to two baseline comparison methods that use the probabilities of motions in unstable sequences.

We generate many sequences of motions and simulate the execution of these sequences by a NAO humanoid robot in a real-time simulator, Webots 7 [Webots, 2014]. We use two established metrics, Kendall's Tau [Kendall, 1948] and Spearman's rank correlation coefficient [Spearman, 1904] to compare the rankings of the predicted relative stability values versus the actual relative stability values since comparing differences between relative values (such as the sum of root-mean-squared error) does not reflect whether the sequences are ranked correctly and that the most stable sequence is selected. Hence, showing that the ranking of the predicted relative stability matches the ranking of the actual relative stability is more important than ensuring that the absolute values are accurate.

We use Algorithm 18 to calculate the actual relative stability of each sequence in the set of sequences. We use the states of the robot after executing each action in the sequence and the reward function to determine the total reward.

Definition 6.2.12. *Let rs_i be the reward of sequence u_i^s in U^s. Let RS be the set of rewards of all the possible sequences U^s.*

Similarly, a longer sequence will accrue a higher total reward than a shorter sequence, by virtue of taking more actions. Different sequences may have different number of motions due to the synchronization of the motions to the input signal. Hence, to ensure a fair comparison, we determine the actual relative stability by dividing the total reward by the number of actions in the sequence.

Definition 6.2.13. *Let $ars_i = \frac{rs_i}{|u_i^s|}$ be the actual relative stability of sequence u_i^s in U^s where $|u_i^s|$ is the number of actions in the sequence. Let ARS be the set of the actual relative stability of all the possible sequences U^s.*

For our experiments, we generated 101 input signals that were pre-processed. These 101 input signals are the 101 sentences from the 20 stories listed in Appendix C. For each input signal, there is a list of possible sequences of motions. For these 101 input signals, there are 2445 possible sequences of motions that match the labels and are synchronized to the labels of the sentence.

We simulate the 2445 sequences that are executed on the NAO humanoid robot using Webots

Algorithm 18 Evaluates the actual relative stability in U^s.

TRS(U^s)
  ARS $\leftarrow \emptyset$
  for $i = 1$ **to** $\mid U^s \mid$ **do**
    $\text{rs}_i \leftarrow 0$
    $\overrightarrow{V_i} \leftarrow \Omega(u_i^s)$
    bas $\leftarrow \mathbb{D}(\overrightarrow{v_0})$
    for $p = 1$ **to** $\mid \overrightarrow{V_i} \mid$ **do**
      $a \leftarrow getAction(u_i^s, p)$
      bas$' \leftarrow \mathbb{D}(\overrightarrow{v_p})$
      $\text{rs}_i \leftarrow \text{rs}_i + \text{RF}(\text{bas}, a, \text{bas}')$
      bas $\leftarrow$ bas$'$
    end for
    $\text{ars}_i \leftarrow \frac{\text{rs}_i + \text{RF(bas,NULL,NULL)}}{|\overrightarrow{V_i}|}$ // Reward for last state
    ARS $\leftarrow$ ARS $\cup \{\text{ars}_i\}$
  end for
  return normalize(ARS)

7 [Webots, 2014], a real-time simulator that simulates the dynamics of the robot. There are 157 unique actions in total in the motion library, including Hold and Observe. We record the body angles of the robot at each instant a motion in the sequence is executed.

For the function $\mathbb{D}(\overrightarrow{v})$, we map the state vector of body angles into discrete states, where we discretize the continuous body angles into bins and the index of the bin for the body angle ba is determined by the equation bin $= \frac{|\text{ba}|}{\varpi} + 1$, where $\varpi \in \mathbb{R}^+$. For example, if a robot's body angle X is 5° and $\varpi = 5^\circ$, the index of the X bin is 2; if the body angle Y is 24°, the index of the Y bin is 5. Hence, the state is represented by the vector $(\text{bin}^X, \text{bin}^Y)$, where bin^X is the index of the bin for body angle X and bin^Y is the index of the bin for body angle Y.

Since the state consists of discretized bins for body angles, to calculate the reward based on the state, we use the mean of the body angle in the bin.

Definition 6.2.14. *The function* $\mathbb{Y} : \mathbb{Z}_1^+ \rightarrow \mathbb{R}^+$ *converts the index of the bin for the body angle into the mean of the body angle in the bin:* $\mathbb{Y}(bin) = (bin * \varpi) - \frac{\varpi}{2}$

We define two reward functions that use the cosine function. We use cosine because the body angle is 0 when the robot is completely upright, and hence the reward is at a maximum of 1 whereas when the body angle of the robot increases towards $\frac{\pi}{2}$, the reward decreases to a

minimum of 0. The first reward function uses both body angles X and body angles Y whereas the second reward function uses only body angle Y. If we use the first reward function, we look at how far the robot's body deviates in terms of pitch and roll, whereas for the second reward function, we only care about the pitch of the robot's body.

Definition 6.2.15. *The first reward function is* $RF_1(bas, a, bas') = \cos(\mathbb{Y}(bin^X_{bas}))+\cos(\mathbb{Y}(bin^Y_{bas}))$, *where* bin^X_{bas} *is the index of the bin of body angle X in bas and* bin^Y_{bas} *is the index of the bin of body angle Y in bas. Let the second reward function be* $RF_2(bas, a, bas') = \cos(\mathbb{Y}(bin^Y_{bas}))$.

Next, we describe the training data used to form the MDP and the test data to test our approach RS-MDP against the two comparisons, RightAfter and Anytime. We create two experiments:

- LearnAll: We use all 2445 sequences to form the MDP. The tests consist of all the 2445 sequences for each sentence. Hence, there are 101 tests with the 101 input signals, where we determine the relative stability of the sequences for each sentence.
- LeaveOneOut: We conduct a leave-one-out cross-validation, where we remove all the possible sequences for a particular input signal out of the training data to form the MDP. These sequences for the particular signal are used for testing.

Definition 6.2.16. *Let the sequences we use for the training data to form the MDP be* U^{Train}. *Let the sequences we use for testing be* U^{Test}.

For LeaveOneOut, we iterate through sequences in the test data and discard any sequence from the training data that is the same sequence of motions. Hence, $\forall_{u \in U^{\text{Test}}} u \notin U^{\text{Train}}$. For example, if U^{Train} contains $u = (lm_1, lm_2, lm_3)$, then any sequences $u' = (lm_1, lm_2, lm_3)$ are not included in the training set, even if u and u' are applicable to different signals. After forming the MDP from the remaining training sequences, we predict the relative stability of the sequences for each input signal.

If we compare the difference between the predicted relative stability aer_i and the actual relative stability ars_i for the sequence u^s_i, it is difficult to evaluate how well our approach does with a summed difference. For example, for our experiment using all the data to form the MDP, we determine the absolute difference between the expected rewards and the actual rewards to be 0.068 ± 0.125. However, this absolute difference does not illustrate how well our approach performs. Hence, we compare the ranking of the average actual rewards to the average expected rewards.

To evaluate how well our approach performs in terms of ranking, we use two metrics that are commonly used to compare rankings. The two metrics are Kendall's Tau [Apache Commons Math, 2015a] and Spearman's Rank correlation [Apache Commons Math, 2015b] and we use these two functions available in Apache Commons Math package.

We compare the rankings of the rewards, RS, versus the rankings of the expected rewards ER computed from RS-MDP. We also compare the rankings of the actual relative stability, ARS, versus the rankings of the predicted relative stability AER from RS-MDP. We want to show that averaging the actual reward and the expected reward by the number of actions in the sequence to calculate the actual relative stability and the predicted relative stability results in better performance. As we explained earlier, the longer the sequence is, the higher the reward attained, we therefore average the reward by the number of actions to determine the relative stability of a motion sequence.

To illustrate how Kendall's Tau and Spearman's Rank correlation work, we are comparing two sets of values, where $G = (g_1, \ldots, g_n)$ and $H = (h_1, \ldots, h_n)$, where in our case G refers to the true relative stability for the various sequences (regardless whether it is averaged or not) and H refers to the predicted relative stability for the corresponding sequences in G (regardless whether it is averaged or not). In our experiments, g_i would be the true relative stability computed for the i^{th} sequence and h_i would be the predicted relative stability computed for the i^{th} sequence. Both G and H need not be sorted in any order, but each i^{th} value in G and H has to be the respective value for the i^{th} sequence.

Kendall's Tau groups the rewards into pairs, i.e., $(g_1, h_1), (g_2, h_2), \ldots, (g_n, h_n)$. It computes a τ value using this formula, $\tau = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$.

n_c is the number of concordant pairs. Concordant pairs refer to pairs (g_i, h_i) and (g_j, h_j) that fulfill the following conditions:

- $i \neq j$;
- $g_i > g_j$ and $h_i > h_j$ or $g_i < g_j$ and $h_i < h_j$.

n_d is the number of discordant pairs. Discordant pairs refers to pairs (g_i, h_i) and (g_j, h_j) that fulfill the following conditions:

- $i \neq j$;
- $g_i > g_j$ and $h_i < h_j$ or $g_i < g_j$ and $h_i > h_j$.

n_1 and n_2 refer to tied pairs for g and h respectively. n_1 refers to pairs (g_i, h_i) and (g_j, h_j)

Table 6.4: Comparisons with the reward function RF_1.

Approach	Ranking (Actual vs Predicted)	LearnAll		LeaveOneOut	
		Kendall's tau	Spearman's rank	Kendall's tau	Spearman's rank
RS-MDP	RS vs ER	0.74 ± 0.18	0.85 ± 0.17	0.66 ± 0.30	0.76 ± 0.34
	ARS vs AER	0.73 ± 0.16	0.85 ± 0.12	0.69 ± 0.18	0.81 ± 0.16
RightAfter	RS vs $RAPS_i$	-0.55 ± 0.28	-0.65 ± 0.32	-0.54 ± 0.28	-0.63 ± 0.32
	ARS vs $\widetilde{RAP}_i$	0.57 ± 0.25	0.68 ± 0.28	0.56 ± 0.27	0.64 ± 0.31
Anytime	RS vs $ATPS_i$	-0.46 ± 0.33	-0.55 ± 0.38	-0.45 ± 0.32	-0.54 ± 0.37
	ARS vs $\widetilde{ATP}_i$	0.54 ± 0.29	0.66 ± 0.32	0.49 ± 0.31	0.60 ± 0.36

where $g_i = g_j$ and n_2 refers to pairs (g_i, h_i) and (g_j, h_j) where $h_i = h_j$.

n_0 is computed with the equation $n_0 = \frac{n(n-1)}{2}$, where n is the number of values in each set. Kendall's Tau returns a value between -1 to 1, where -1 means that there is perfect disagreement between two pairs and 1 means there is perfect agreement between two pairs. If either G or H contains a list of equal values, Kendall's Tau is 0.

For Spearman's rank correlation, we use the equation, $\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$. The values in G are ranked and the average of the ranks is assigned to tied values and we term the ranked values G^r. For example, if $G = (0.8, 2.3, 1.2, 1.2, 15)$, $G^r = (1, 4, 2.5, 2.5, 5)$. Tied values share the same ranking, where the ranking is the sum of applicable rankings averaged by the number of values with the same rankings. Likewise, we term the ranked values for H, H^r. d_i refers to the absolute difference between g_i^r and h_i^r. n refers to the number of values in G, where $n = |G| = |H|$. Spearman's rank correlation also returns a value between -1 to 1, where -1 means that there is perfect disagreement between the two sets of rankings and 1 means there is perfect agreement between two sets of rankings. If either G or H contains a list of equal values, Spearman's rank correlation is 0.

Table 6.4 shows the results for LearnAll and LeaveOneOut using the reward function RF_1. Our approach, RS-MDP, outperforms the baseline comparisons given that there is a higher level of agreement for expected reward (with or without averaging) when compared to RightAfter and Anytime. We reach the same conclusion whether we use all the sequences to learn for LearnAll

or for LeaveOneOut, where we leave out all the sequences for a particular input signal.

For LearnAll, when we use RS-MDP, there is little difference in the level of agreement for the rankings whether we average the relative stability value by the number of actions when we compare RS vs ER to ARS vs AER. However, for LeaveOneOut, ARS vs AER has a higher level of agreement than RS vs ER. Therefore, the performance of RS-MDP improves by averaging the relative stability value by the number of actions to determine the actual or predicted relative stability.

Using Kendall's Tau and Spearman's rank correlation, the results show that without averaging for RightAfter and Anytime, the rankings are inversely correlated. After averaging, Kendall's tau and Spearman's rank correlation show that the rankings are more correlated with a higher level of agreement.

When we compare the Kendall's Tau and Spearman's rank correlation for the same comparisons in LearnAll versus LeaveOneOut, it is expected that the rankings have a higher level of agreement in LearnAll than LeaveOneOut since we train the MDP with all the data in LearnAll.

For the two comparisons using RightAfter and Anytime, the order of the actions is ignored since the probabilities remain the same regardless of the order of the actions. Moreover, once an action has a stable probability of 0 and appears in a sequence, the sequence will always have a probability of 0. For some input signals, RightAfter and Anytime produce probabilities that are exactly the same, therefore Kendall's Tau and Spearman's rank correlation return a 0. However, RightAfter and Anytime act as a baseline for comparison to RS-MDP, albeit that the comparisons are naive and suffer from the problems we describe earlier.

We present the comparisons for the reward function RF_2 in Table 6.5. We observe that the level of agreement in rankings are similar. We believe that this is due to the robot only falling forward or backwards and seldom falling to the right or left during the experiments, an observation that we made in Section 6.1.2. Hence, the performance is similar when using body angles X (roll) and body angles Y (pitch) in RF_1 to using only body angles Y (pitch) in RF_2.

Discussion

Relative stability is a new concept and we present our approach – RS-MDP – to predict the relative stability of a sequence compared to other possible sequences generated for an input signal. RS-MDP forms a MDP using past executions of sequences and predicts the relative

Table 6.5: Comparisons with the reward function RF_2.

Approach	Ranking (Actual vs Predicted)	LearnAll		LeaveOneOut	
		Kendall's tau	Spearman's rank	Kendall's tau	Spearman's rank
RS-MDP	RS vs ER	0.74 ± 0.20	0.85 ± 0.18	0.66 ± 0.31	0.76 ± 0.35
	ARS vs AER	0.73 ± 0.17	0.85 ± 0.13	0.70 ± 0.19	0.81 ± 0.17
RightAfter	RS vs $RAPS_i$	-0.56 ± 0.29	-0.65 ± 0.33	-0.55 ± 0.29	-0.63 ± 0.32
	ARS vs $\widetilde{RAP}_i$	0.58 ± 0.25	0.68 ± 0.28	0.56 ± 0.26	0.65 ± 0.30
Anytime	RS vs $ATPS_i$	-0.46 ± 0.34	-0.55 ± 0.39	-0.44 ± 0.32	-0.54 ± 0.37
	ARS vs $\widetilde{ATP}_i$	0.55 ± 0.27	0.67 ± 0.31	0.51 ± 0.28	0.61 ± 0.33

stability of a sequence using the expected rewards accrued given the initial state and the sequence of actions. We show that RS-MDP outperforms the two benchmarks – RightAfter and Anytime.

Using RS-MDP, there will be instances where a particular action in the sequence of motion primitives has not been performed at a particular state. In these instances, we take a "pessimistic" view that the robot will fall and give it a minimum reward. For future work, we adopt an "optimistic" view that the robot is stable and give it the maximum reward.

For sequences with new actions that do not exist in the sequences used to form the MDP, we are unable to predict the relative stability. Instead, we use the execution of such sequences for future predictions where we learn from the execution of these sequences by adding the data in the MDP.

We only use the body angles for states to showcase our approach. Researchers have used angles, angle velocities etc. in the inverted pendulum problem. There is no restriction on the definition of the state in our approach and the definition of states can be varied for future work.

6.3 Chapter Summary

Motions are used to convey meanings of an input signal. Given that there are multiple motions per label in the motion library and multiple labels in an input signal, different sequences of relevant motions are generated. Each motion in the motion library is assumed to be stable, but a

sequence may cause the robot to fall. We are interested in predicting if a motion sequence will result in a fall, without executing the sequence on the robot. We contribute ProFeaSM that uses only body angles collected during the execution of single motions and interpolations between pairs of motions, to predict if a sequence will cause the robot to fall. We demonstrate the efficacy of ProFeaSM and explore the trade-off between precision and recall on a real NAO V3.3 H21 humanoid robot and a simulated NAO V4.0 H25 in Webots.

A humanoid robot executes a sequence of motions to fulfill a particular task, such as telling a story or for human-robot interaction. The robot selects a sequence of motions from a list of possible sequences. Choosing the most stable sequence will ensure that the robot executes future sequences without interruption, e.g., without having to recover from a fall. Relative stability refers to the stability of a sequence as compared to other sequences. To our knowledge, we are the first to contribute an approach – RS-MDP – to determine the relative stability of motion sequences. RS-MDP does not require a model of the robot to determine the dynamics of the robot's stability, and uses past executions of sequences of motions to predict the relative stability of a motion sequence. Moreover, RS-MDP predicts the relative stability of sequences that have not been executed, as long as there exists data on the motions in the sequence. Since relative stability is new and there are no existing methods to compute relative stability, we came up with two benchmarks, and show that RS-MDP outperforms these benchmarks.

Chapter 7

Related Work

This chapter presents a review of related work. We discuss the relevant past research about the autonomous animation of humanoid robots for music and speech and highlight the differences of this thesis.

We categorize the review of past research based on the five core challenges of the thesis – R-M-S^3, namely:

- **R**epresentation: We investigate how robot motions are defined and look at how the input signals for different domains are represented in the current literature.
- **M**appings: We explore how researchers assign meanings, e.g., mapping labels to motions so as to select relevant motions. We investigate how researchers compare different motion trajectories so as to propose mappings for similar motion trajectories.
- **S**election and **S**ynchronization: We consider existing work on selecting relevant motions and synchronizing motions to the task using the two task domains: synchronization with music and synchronization with text-to-speech. We evaluate the existing literature on how feedback for motions is used to improve the selection of motions.
- **S**tability: We explore the current literature on how the fall of a sequence of motions is predicted. We also review work done on comparing the stability of motion sequences.

7.1 Representation

We explore the existing literature on how researchers define robot motions for the two tasks that we investigate for this thesis and organize a review of the current literature into these two subsections:

- Motions for music;
- Motions for speech.

Motions for music

Humanoid robot dances are generally preprogrammed by choreographers for a particular piece of music so that the motions are smoothly executed, synchronized to the music and that the robots remain stable [Ellenberg et al., 2008, Shanie, 2006], e.g., four QRIOs danced in a music video where the dance was manually choreographed [Montgomery, 2005] and 540 humanoids performed a synchronized dance that was also manually choreographed [Reich, 2016]. Some other robots with multiple degrees of freedom, e.g., Keepon, a creature-like robot with four degrees of freedom, was programmed to dance to the beats of the music with five parameters for each of the four degrees of freedom randomly selected and changed at random intervals [Michalowski et al., 2007]. Other robot dances were created by imitating the dance movements of humans using motion capture data, e.g., [Nakaoka et al., 2010]. Others also randomized motions using a few discrete options or paths for the robot to move to, e.g., [de Sousa Junior and Campos, 2011].

To automate dance motions for music, we select motions based on the beats and emotions of the music. Comparatively, the emotions of the music are usually not analyzed and reflected in robot dances done by other researchers. Also, robots have been programmed to use their facial features to express emotions for human-robot interactions [Breazeal, 2003, Kirby et al., 2006], but not body postures or movements.

Computer animation for dancing virtual characters is often done by synthesizing motion clips selected from a database of motion capture data and time-warping the clips to synchronize the motion clips with the music beats [Kim et al., 2003b, Shiratori et al., 2006, Kim et al., 2007]. However, the number of combinations is limited to the number of motion clips. Computer animation researchers have also explored the use of emotions in music, but they only vary the speed

of the body movements [Shiratori et al., 2006] and do not reflect the positive-negative aspect of emotions (e.g., happy and sad). In our work on automating the dance motions for any piece of music, we selected motions based on the emotion of the music and adjusted the motion to synchronize with the beats of the music [Xia et al., 2012]. We have also generated a large number of combinations of motions using a small motion library. For example, to automate dance motions, we defined a small library of 52 motions and combined them to form 16,848 possible motions combinations [Xia et al., 2012].

Motions for speech

Gestures for speech have generally been organized into several categories, though some researchers use different names for similar categories, e.g., *beat* gestures are also termed as *batonic* gestures. [Bennewitz et al., 2007, Nieuwenhuisen and Behnke, 2013]. We summarize each category with a brief description from various sources [McNeill, 1996, Beattie, 2004, Ng et al., 2010]. Iconics illustrate the characteristics of physical concrete entities and/or actions with the motion of the hands. For example, showing how big an object is using the arms and hands. Metaphorics are gestures whose pictorial content describe abstract concepts rather than concrete objects. For example, referring to both sides of an argument using both hands. Deictics are pointing gestures that point to items that are being described. Beats are small, short movements moving along with the rhythm of the speech to convey emphasis, emotion and personality and do not convey any semantic information. Emblems are commonly understood without speech and are self-explanatory, but are culturally-specific. For example, waving the hand to say goodbye or nodding in agreement with someone. Regulators are turn-taking gestures. For example, a person wanting to speak raises an arm. Affect displays are gestures that show emotions. In our motion library, we have iconics, metaphorics, emblems, and deictics which are defined as spatially targeted motions in this thesis. We do not use beat gestures as they convey no meanings, nor regulators since we use a single humanoid robot and there is no turn-taking involved.

Though these are categories of gestures proposed by various researchers, McNeill claims that "it is more appropriate to think in terms of combinable dimensions rather than categories and there is no need of a hierarchy" [McNeill, 2005]. The breakdown of the types of gestures is generally about 40% iconics, 40% beats, and the remaining 20% are divided between deictic and metaphoric gestures [Cassell et al., 1994, McNeill, 1996]. The breakdown is useful in

determining the type of gestures selected to be used to accompany speech. We built upon their work and proposed a different category of gestures – body part categories. By defining body part categories, we generated many whole body motions for a humanoid robot using a small motion library [Tay and Veloso, 2012].

Researchers create guidelines for generating motions for human robot interaction and we compare three trajectory design methods: keyframing methods, physical modeling techniques and motion capture methods [Saerbeck and van Breemen, 2007]. We summarized the advantages and disadvantages of these three trajectory design methods in Table 7.1. We formalized motions as keyframes with parameters to generate motions that are adjusted to synchronize to the input signal [Xia et al., 2012, Tay and Veloso, 2012].

Table 7.1: Comparison of three trajectory design methods.

Method	**Advantage**	**Disadvantage**
Keyframes	It is a good representation for creating expressive motions for all kinds of embodiments.	It is difficult to model physically correct motions.
Physical Modeling	Since physical modeling techniques model a movement as a force affecting a physical system, natural motion trajectories are created.	It is difficult to create specific expressions.
Motion Capture	Motion capture covers both advantages of the *Keyframes* and *Physical Modeling* methods.	Motion capture is restricted to embodiments that have similar dynamics to the human body.

Salem et al. built upon the Articulated Communicator Engine (ACE) [Salem et al., 2009, Salem et al., 2010, Salem et al., 2012] that was implemented on a virtual agent named Max, and implemented it on the Honda humanoid robot, ASIMO [Salem et al., 2012]. ACE does not store any definitions of motions but generates motion trajectories given the end-effector targets defined in the task space for different motor planning modules, such as the arms, the wrists and the hands [Salem et al., 2009]. To determine the joint angles for the whole body motion, "inverse kinematics (IK) is solved on the velocity level using the ASIMO whole body motion (WBM) controller framework" [Salem et al., 2009]. However, such an approach suffers from

the drawback that the IK solution is not feasible and does not consider the robot's phsyical constraints, such as collision avoidance. In our approach, we assume that the gestures generated for the robot do not have self-collision and each gesture is stable. Our formalization of the robot motions is also applicable to humanoid robots given that the definitions of the motions are specifically designed for the humanoid robot.

Ng et al. define gestures as a "set of key points for each parameter" and trajectories are generated using Kochanek-Bartels tension-continuity-bias (TCB) cubic splines [Kochanek and Bartels, 1984] where the parameters determine how closely the trajectories follow the key points [Ng et al., 2010]. Though defining motions in trajectories with parameters to adjust the shape of the trajectories allow for highly varied and expressive motions, the final motion trajectory generated may not express the original intended meaning well. The choice of the values for the parameters is derived either from empirical data or defined.

Xing and Chen implemented a set of predefined gestures on a robotic puppet made up of nine moving parts with wires, consisting 30 degrees of freedom and eight motors for non-verbal gesture generation accompanying speech [Xing and Chen, 2002]. Xing and Chen created a set of predefined gestures using primitive templates that describe the desired trajectories of movements and the dynamic properties of the motor actions controlling the wires, [Xing and Chen, 2002]. Luo et al. continued on the work using a hand glove to animate the actions of the puppet [Luo et al., 2011]. The work on the robotic puppet was based on controlling the wires and did not provide details on how the motions were modified to synchronize with the accompanying speech.

Other researchers have explored how humanoid robots imitate human motions but they do not define motions that can be combined with output modalities such as speech. We review the literature on imitating human motions using robots in Section 7.2 to determine how motion trajectories are compared. The literature is useful for mapping associations between meanings (labels) and motions since motion trajectories are clustered to avoid duplicate motions.

Many researchers have highlighted the importance of proxemics in the non-verbal communication expression for an autonomous humanoid robot and provided guidance on the distance between humans and robots [Brooks and Arkin, 2007, Mumm and Mutlu, 2011, Walters et al., 2011]. We introduced the Spatially Targeted Motion primitive (STM), whereby an acceptable range of distance to a target is defined [Tay and Veloso, 2012]. Sisbot et al. presented an integrated motion synthesis framework that plans and generates robot motions from the human's perspective by taking into account the human's safety, the human's vision field and perspective

etc [Sisbot et al., 2010]. However, the STM focuses on the robot's perspective, where the distance from the robot and the orientation to face any target is defined in a range that is bounded by two values [Tay and Veloso, 2012]. The STM allows the robot to direct the motion at the target without moving constantly to compensate for the motions of the human since it is determined from the robot's perspective and the position is bounded within a range of two values.

Previous research was done on automating the generation of speech-based motions of virtual agents on screens [Kopp and Wachsmuth, 2000, Cassell et al., 2001]. Sergey uses the prosody of speech to train a hidden Markov model using motion capture data and prosody cues in the speech [Sergey, 2009]. No context of the speech is used to select the speech since only the tones of the utterances and smooth transitions between gestures drive the selection of the gestures from the motion capture sequences.

The aspect of physical embodiment is ignored by generating gestures for virtual agents given that real physical humanoid robots share time and space with people. The perspective of the user is not easily shifted from one scene to the next in the real world. Physics-based effects such as the robot falling is not inherent in virtual reality and actuated robots also interact with objects in the environment. We demonstrate our work using a NAO humanoid robot, use its physical constraints such as joint velocities to compute the execution time of motions and parameterize the motions to change the duration of the motion so that each motion is synchronized to the input signal.

7.2 Mappings

Kim et al. recorded human gestures and modeled these gestures on a real robot and in simulation by categorizing the gestures with 13 types of sentences and 3 emotions [Kim et al., 2010]. The robot gestures are selected based on the type of the sentence and the emotion and require the sentence to be categorized. Researchers have also described methods to learn new robot's behaviors and motions based on nautral language descriptions [Kress-Gazit et al., 2007, Rybski et al., 2008, Cantrell et al., 2011], but to automatically map the new behaviors and motions to the descriptions, the natural language descriptions have to follow a specific structure or satisfy certain requirements so that they are parsed properly. Other researchers annotate existing gestures with semantic tags [Cassell et al., 2001, Neff et al., 2008, Sergey et al., 2010]. To our knowledge, we are the first to map emotional labels to motions based on the static poses autonomously.

Researchers found that people were able to identify most of the emotions expressed by key poses of a NAO humanoid robot [Beck et al., 2010, Haring et al., 2011b, Beck et al., 2013]. We built upon their work by using the features of the static poses to determine the activation-valence label [Xia et al., 2012]. To analyze the similarity of the motions with the labeled emotional static postures collected, we considered the points of interest of the robot's body. These points of interest are similar to the points used in motion capture systems and point-light animation [Li and Chignell, 2011].

Some researchers considered modifying motions based on the features of the motions. Masuda and Kato introduced a motion rendering system that modifies motions based on the features of the Laban movement analysis (LMA) [Masuda and Kato, 2010]. The features are: space, time, weight, inclination, height and area. Space "represents the bias of whole-body movement" and is related to the movement direction of the extremities and the direction of the face. Time "represents the quickness of whole-body movement" and is related to the joint angle velocities. Weight "represents the powerfulness of whole-body movement" and is related to the joint angle accelerations. Inclination represents the forward inclination of the body posture and is related to the center gravity of the body by modifying the joint angle of the waist. Height "represents the straightness of posture". Area represents the range of the body and is related to the quadrilateral area made up of the four extremity points of the limbs on the horizontal plane.

We explore the use of emotional static postures to label motions using Thayer's Activation-Valence model [Thayer, 1989] instead of using the features of the LMA since the motions are synchronized to the emotions and beats of the music. The time and weight aspects of LMA are incorporated into the motions due to the beats of the music in this thesis. We also consider the inclination and height of LMA since the heights and inclinations of the emotional static postures are varied based on the emotions conveyed by the static postures.

Next, we investigate how mappings between motions and labels are autonomously proposed when a new motion or label is added to the existing motion library. Researchers have proposed how the trajectories of motions are compared and also researched on the similarities of the meanings of labels, particularly semantic labels. We separate the comparisons into two groups: *Comparisons of Motion Trajectories* and *Comparisons of Labels*.

Comparisons of Motion Trajectories

Determining the similarity of motions is challenging due to the imprecision in spatio-temporal data, particularly when one has to define when two trajectories are sufficiently similar. For example, trajectories may be similar in the entire trajectory or in part, such as the start and end points, but be internally different. Researchers use metrics such as Euclidean distance, Hausdorff metric or dynamic time warping and tuning of multiple parameters to cluster trajectories [Meratnia and de By, 2002, Erdogan and Veloso, 2011b, Sung et al., 2012].

Erdogan and Veloso analyzed the similarities between pairs of two-dimensional motion trajectories using the Euclidean distance between points and the Hausdorff metric [Erdogan and Veloso, 2011a]. Using the similarities derived, Erdogan and Veloso used a variant of agglomerative hierarchical clustering to determine groups of similar robot trajectories [Erdogan and Veloso, 2011a]. However, Erdogan and Veloso's method is used to find a cluster of trajectories that is assigned to only one group. Their method cannot be applied in a many-to-many relationship where a cluster of trajectories is assigned to one group and a subset of the trajectories is assigned to another. For our task of assigning labels to a new motion, some of the labels are assigned to other motions as well. Thus, a motion is mapped to different labels and a label is mapped to many motions, resulting in a many to many relationship.

Researchers have also considered how to determine similarity between motions of humanoid robots and human motions from motion capture data so as to determine if the imitated human motions are similar to the robot's motions. Imitated human motions are associated with existing robot motions whereby the robot motions are assigned to certain tasks or labels. To avoid duplications in the robot motions stored in the library, researchers used variations of Hidden Markov Models (HMM) to differentiate between similar and different motions [Kulić et al., 2008, Okuzawa et al., 2009, Calinon et al., 2010]. Others used methods like Principal Component Analysis [Motomura et al., 2009, Tran et al., 2010] or expectation-maximization clustering algorithms [Sung et al., 2012]. However, these methods also do not allow motions to be clustered into different groups at the same time.

Huang et al. used a similarity function that compares the joint angles and velocities with a parameter that is adjusted to weigh the similarity between the spatial effect and temporal effect [Huang et al., 2010]. We consider both spatial and temporal effects by varying the joint angles and velocities separately and both joint angles and velocities in our experiments.

Comparisons of Labels

Labels refer to different things in different input signals. For example, using a piece of emotional music as the input signal, emotional labels are used to identify different emotions and are represented using Thayer's 2-dimensional Activation-Valence (AV) model [Thayer, 1989]. Using a 2-dimensional AV model, we compare the differences in labels by using the Euclidean distances between two 2-dimensional points [Xia et al., 2012]. Besides using a 2-dimensional AV model, we did not find other models that compare emotional meanings. Another example is a story whree text labels are used to represent semantic meanings. Similarity of semantic labels is determined by using open-source large lexical database such as WordNet (Princeton University) [Princeton University, 2010] and MindNet (Microsoft) [Microsoft Research, 2005]. We leverage on the research on the similarity of the semantic meanings of words and use existing large lexical databases to determine the similarity of the semantic labels. We use word2vec [Mikolov et al., 2013] that determines the similarity of words using numerical values, so that it is easy to compare and determine the differences in meanings.

7.3 Selection and Synchronization

We considered how motions are selected, generated to motion sequences, and synchronized to dance music and text-to-speech. We will present work done for virtual agents and robots. We note that the difference between synchronization of motions to the input signal for virtual agents and robots differs in the aspect of physical embodiment. With physical embodiment, physical constraints such as joint velocity constraints are taken into account. Nonetheless, synchronization of motions to the input signal for virtual agents and robots share many similarities. We also review how the audience evaluates the motion sequences and researchers acquire the feedback from audience to improve the selection of motions. We organize the review into these subsections:

- Selection and Synchronization for Autonomous Dances;
- Selection and Synchronization for Autonomous Co-Verbal Gestures;
- Selection of Motions using Audience Preferences of Motion Sequences.

Selection and Synchronization for Autonomous Dances – Virtual Agents

Creating dancing characters is commonly found in computer animation research [Kim et al., 2007, Oliveira et al., 2010]. Researchers extract features of the motions based on the features of the music, e.g., rhythms or beats of the music, and classify these motions into groups [Kim et al., 2003a, Shiratori et al., 2006, Kim et al., 2007, Oliveira et al., 2010]. They select the motions based on the features of a new piece of music and synchronize the motions to the beats of the music. However, their approach requires the music to have a similar feature of the music, e.g., beats, to select relevant motions in the motion library, so this approach is not general for all kinds of music. We did not find work done on examining the emotions of the music to generate emotional dances for virtual agents. Though the virtual agents' motions are synchronized to the music, physical constraints are not enforced or taken into consideration. The trajectories of the motions for the dancing characters are often fixed.

Selection and Synchronization for Autonomous Dances – Robots

Robot dances are generally choreographed by humans or generated from motion capture data of humans [Nakaoka et al., 2003, Kudoh et al., 2008]. As we previously mentioned in Section 7.1, robots such as Keepon were programmed to dance to the beats of the music by randomly selecting parameters to change the motions at random intervals [Michalowski et al., 2007]. Others also randomized the selection of motions from the motion library using a few discrete options or paths for the robot's dance [Grunberg et al., 2009, de Sousa Junior and Campos, 2011], whereas Seo et al. generated repetitive rhythmic motions like head nodding or hand shaking for the DARwIn-OP humanoid robot to dance to the rhythm of the music [Seo et al., 2013]. Although some of the robot dances are autonomously generated to dance to the beats of the music [Grunberg et al., 2009, Seo et al., 2013], they select random motions with no consideration of the relationship between the motion and music. We select relevant motions based on the motion's AV label and the AV label of the music. Thus, with our approach, the emotions of the music are reflected in the robot dances.

Selection and Synchronization for Autonomous Co-Verbal Gestures – Virtual Agents

Sergey et al. generate gestures based on the online processing of live speech and select gestures using a Markov decision process and value iteration [Sergey et al., 2010] and there is no synchronization as the gestures are selected based on the prosody features for an utterance. Several virtual agents select gestures based on the rules defined and specify timing constraints at the phoneme level for co-verbal gestures [Cassell et al., 2001, Kopp and Wachsmuth, 2004, Salem et al., 2012]. The stroke phase of the gesture is set to precede the corresponding text by a given offset (e.g., 0.3 seconds) or to start exactly at the start of the text [Kopp and Wachsmuth, 2004]. Sometimes, the exact gesture is specified to be executed at a certain time [Salem et al., 2012]. Some researchers timed the use of different types of gestures: for example, for the iconic or metaphoric gesture, the preparation of the gesture is set to begin at or before the beginning of the text and to finish at or before the next gesture, or the intonational stress of the phrase, whichever comes first [Cassell et al., 1994, Kipp et al., 2007b, Nieuwenhuisen and Behnke, 2013]. The use of the gestures is dependent on the linguistic and contextual information extracted from the speech [Cassell et al., 1994, Breitfuss et al., 2007, Breitfuss et al., 2009]. Similarly, gestures for virtual agents rarely consider the physical constraints, such as the dynamics of the motions, since virtual agents do not fall. Also, physical humanoid robots have less degrees of freedom than a virtual animated character, thus the expressiveness of the gestures is decreased.

Selection and Synchronization for Autonomous Co-Verbal Gestures – Robots

Many researchers focus on generating a single type of gesture for humanoid robots, either deictic [Striegnitz et al., 2005, Okuno et al., 2009, Shiwa et al., 2009] or emblematic gestures [Erden, 2013]. Salem et al. use the ACE framework where the speech utterance is described using the Multimodal Utterance Representation Markup Language (MURML) [Salem et al., 2012]. The ACE engine synchronizes the motions and speech by adapting the motion to the structure and the timing of the speech by obtaining absolute motion time information at the phoneme level [Salem et al., 2012]. There are only a few systems where different types of gestures, e.g., combination of iconics, metaphorics and beat gestures, are generated for a robot [Bennewitz et al., 2007, Ng et al., 2010]. For example, Ng et al. probabilistically selects the gesture type based on

the possible candidates for each word and the desired expressivity [Ng et al., 2010]. Next, the timing of the word and style parameters are used to determine the shape of the motion trajectory [Ng et al., 2010]. We do not select gestures based on gesture type, but our approach allows the gestures to be labeled with the gesture type and the input signal is processed to produce such labels. We present two ways to select motions, either probabilistically or based on a weighted criteria.

Research has shown different guidelines on the synchronization of the gestures to speech. Shiwa et al. studied the preferred length of response time for communication robots to determine design guidelines and suggested that a maximum of one to two seconds delay in a robot's response is acceptable [Shiwa et al., 2009]. Kanda et al. found that the robot's body movements should be delayed for 0.89 seconds to look natural for route guidance interaction [Kanda et al., 2007]. Yamamoto and Watanabe found that the robot's body movements should be delayed after 0.3 seconds and utterance should be delayed about 0.6 seconds in a greeting interaction [Yamamoto and Watanabe, 2006]. We define the function $\mathbb{H}$ to synchronize the motions for speech to the start of each label as an example. The function $\mathbb{H}$ can be defined using these guidelines or according to the users' needs.

Gestures have also been broken down into various phases defined by Kendon [Kendon, 1980] and researchers have analyzed gestures based on these phases [Kita et al., 1998, Kipp et al., 2007a], namely preparation, stroke, hold and retraction also known as recovery. Preparation is the phase whereby the gesture moves to the stroke's starting position. Stroke is the most energetic part of the gesture and is a phase that always exists in a gesture. Multiple strokes can occur in a gesture [Kita et al., 1998]. With the breakdown of these phases, the stroke "occurs either with or just before the phonologically most prominent syllable of the accompanying speech [Cassell et al., 1994, Kopp and Wachsmuth, 2004]. Hold is an optional still phase that can occur before and/or after the stroke. Retraction or recovery is the phase that returns to a rest pose (e.g. arms hanging down, resting in lap, or arms folded). Motion primitives in the library are deemed as strokes and we perform preparation automatically by interpolating to the first keyframe of each motion primitive. We perform holds when there is enough time between labels to hold the static pose before moving on to other gestures. We also perform recovery to the initial pose defined whenever the robot has time to return to the initial pose, and the robot performs gestures smoothly and does not stop at awkward poses. At the end of each sequence, the robot returns to the initial pose.

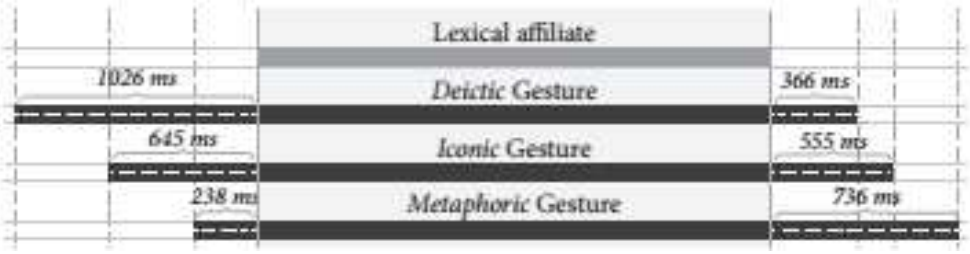


Figure 7.1: Temporal alignment between different types of gestures and lexical affiliates. [Huang and Mutlu, 2013]

Huang and Mutlu proposed a model of how people coordinate their gestures with their speech [Huang and Mutlu, 2013]. Huang and Mutlu identified lexical affiliates which are "words and phrases that co-express meaning with representative gestures, including deictic, iconic and metaphoric gestures" [Huang and Mutlu, 2013]. To determine the timings when a gesture starts and ends, Huang and Mutlu empirically obtained the timings from observations of participants acting as instructors to teach how to fold paper [Huang and Mutlu, 2013]. Figure 7.1 shows the temporal parameters and Figure 7.2 shows the algorithm for synchronizing robot behaviors. Similarly, though we synchronize the start of each gesture to the word that matches the semantic label, we note that our formalization of the motions allows variations in the synchronization function where one chooses to synchronize the start of the gesture earlier or at the start of the label since our approach allows the user to define the synchronization function $\mathbb{H}$.

```
Require: Speech utterances, timestamp[onset,end]-ID pairs for lexical affiliates and beat points
for each utterance do
    new nonverbalBehavior
    for each timestamp[onset,end]-ID pair do
        gesture.selectGestureFromLibrary(ID)
        gesture.setGestureDuration(duration(timestamp[onset,end]))
        gaze.setGazeTarget(sampleFromGazeDistribution(gesture.getType()))
        gaze.setGazeDuration(gesture.getDuration())
        nonverbalBehavior.append(gesture,gaze)
    end for
    nonverbalBehavior.executeBehavior()
end for
```

Figure 7.2: Algorithm for synchronizing robot behaviors. [Huang and Mutlu, 2013]

Selection of Motions using Audience Preferences of Motion Sequences

Hartmann et al. proposed to capture gesture expressivity with a set of six attributes to evaluate co-speech gestures [Hartmann et al., 2005, Hartmann et al., 2006], namely overall activation, spatial extent, temporal extent, fluidity, power and repetition. Overall activation refers to the quantity of movement during a conversational turn (e.g., passive/static or animated/engaged). Spatial Extent is the amplitude of movements (e.g., amount of space taken up by body). Temporal Extent is the duration of movements (e.g., quick versus sustained actions). Fluidity is smoothness and continuity of overall movement (e.g., smooth/graceful versus sudden/jerky). Power is the dynamic properties of the movement (e.g., weak/relaxed versus strong/tense). Repetition is the tendency to rhythmic repeats of specific movements. In this thesis, we explore overall activation and repetition using the definition of the parameter N, the number of times a motion primitive is repeated, and also look at temporal extent, where the motion is dependent on the timings of the labels in the input signal.

Kamide et al. developed a humanoid-oriented psychological scaled called PERception to humaNOiD (PERNOD) that comprises five dimensions: Familiarity, Utility, Motion, Controllability and Toughness [Kamide et al., 2014]. Bartneck et al. came up with the GODSPEED measure that measures five aspects of the robot: Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety [Bartneck et al., 2009]. Joosse et al. also developed a data collection instrument, BEHAVE-II, to assess user responses towards a robot's behavior using both attitudinal and behavioral responses [Joosse et al., 2013].

Pelachaud conducted human studies to evaluate their virtual agent's gestures with speech [Pelachaud, 2005]. Pelachaud also created a "neutral" move of generic action in which the setting of each parameter in their algorithm was set to neutral [Pelachaud, 2005]. Ng et al. conducted evaluations of their co-verbal gestures for ASIMO, the Honda humanoid robot, using videos of the ASIMO in action [Ng et al., 2010]. Salem et al. had evaluations of their co-verbal gestures for ASIMO evaluated with the real robot since they argued that "it is necessary to evaluate non-verbal behavior in actual interaction scenarios" [Salem et al., 2012].

Huang and Mutlu also came up with measures that we summarized in Table 7.2 to evaluate robot's gestures with speech [Huang and Mutlu, 2013] in a scenario where the robot instructs the participants to fold paper. Several researchers also proposed guidelines for motion design for humanoid robots [Breemen, 2004, Ribeiro and Paiva, 2012, Jung et al., 2013, Kamide et al.,

2014] by using animation principles and studies done to evaluate speech-based motions.

We note that some of the evaluation approaches are dependent on the task involved and may be different in other scenarios. Also, subjective measures of human evaluation have to be carefully considered since measures such as naturalness of robot's behavior are difficult to quantify unless compared to a baseline. These metrics can be incorporated in creating the audience ratings for motions.

Table 7.2: Measures used to evaluate robot's gestures and speech.

Type of Measure	**Category of Measure**	**Measure**
Objective	Task Performance	Participants' recall of the information presented by the robot
Subjective	Perceived Performance	Naturalness of robot's behavior
	Perceived Performance	Competence of robot
	Perceived Performance	Effective use of gestures
	Social and Affective Evaluation	Engagement with robot
	Social and Affective Evaluation	Rapport with robot
Behavioral	Narration Behavior of the participants' ability to retell the robot's story	Narration duration
	Narration Behavior of the participants' ability to retell the robot's story	Gesture Use

Feedback from the audience ranges from using visual cues such as the audience holding colored markers such as paddles [Knight et al., 2011] to indicate their preference or audio feedback such as the applause or cheers from the audience or surveys at the end of the interaction [Addo and Ahamed, 2014]. These feedback can be converted into a noisy numerical value to model the distribution of the observed audience preference.

Knight et al. models the audience using the features of jokes and selects the best joke for a robot to tell using the audience feedback and the features of the joke [Knight et al., 2011]. We do not need to model the features of motions and use the feedback on sequences to model the ratings of individual motions. The approach of Knight et al. assumes that the audience preference on features do not vary over time, whereas we model the boredom of the audience and account for different weights assigned to the motions in different sequences. Addo and Ahamed also use a robot to tell jokes, but use reinforcement learning [Addo and Ahamed, 2014]. However, to learn a good policy, they have to explore all jokes in all the states, whereas we do not have to query all sequences to pick the best sequence.

Abbeel and Ng introduce the concept of inverse reinforcement learning, where the reward function is unknown and that it is difficult to specify a reward function, but the "unknown reward function is expressed as a linear combination of known features" [Abbeel and Ng, 2004]. Our approach is similar in the sense that the audience rating of a sequence is expressed as a linear sum of the unknown ratings of single motions. However, we do not require a Markov decision process made up of states and actions to model the audience preferences and determine the best policy for each state. Instead, we only model the ratings of single motions.

Akrour et al. explore the use of a Markov decision process and rank sequences of state-action pairs based on the preferences instead of assigning values to the sequences [Akrour et al., 2012]. We do not order preferences through ranking as the magnitude of how much a sequence is preferred over another sequence is lost in ranking of sequences.

Our approach, MAK, uses the multi-armed bandit algorithm and Kalman filter. The multi-armed bandit problem is a well-known problem, where the goal is to select which arms to pull to maximize the sum of expected rewards, and Thompson sampling [Thompson, 1933] is one of the common algorithms used to optimize the arm-pulling. The multi-armed bandit problem was applied to allocating training instances to learning agents, so as to estimate their learning rates and maximize the team performance [Liemhetcharat and Veloso, 2014b, Liemhetcharat and Veloso, 2014a], and Kalman filters were used to estimate the agents' learning rates. In this paper, we use a single Kalman filter to estimate the audience preferences for the motion-label pairs, and we modify the multi-armed bandit problem to select multiple motions to match a sequence to receive feedback from the audience.

7.4 Stability

Researchers have tackled the problem of generating co-speech gestures in the area of computer animation for virtual conversational agents and in the area of robotics for non-verbal behaviors along with speech in humanoid robots [Salem et al., 2012]. Salem et al. highlighted the difference between the level of complexity between these two fields by stating that "character animation has less restrictive motion than even the most state-of-the-art humanoid robots as animation of virtual agents reduces or even eliminates the problems of handling joint and velocity limits", whereas a robot has real physical restrictions on its motions [Salem et al., 2012]. Stability is vital for robots since a fall for a robot may cause the robot to be damaged or broken

whereas a virtual agent can easily reset to its original pose without any issues.

Falls should be prevented for a humanoid robot as it takes time to get up and may cause the robot to damage some of its joints or cause serious wear and tear. To predict a fall, researchers determine the dynamic stability using a model of the robot and its environment. They verify their predictions by executing the sequence of motions in simulation or on the real robot and check if they are stable. However, it is difficult to obtain an accurate model to predict stability reliably as it is hard to model real world variables such as friction, wear and tear and slippage.

Motion planners for stability generally require an accurate model of the robot and its environment. It is also difficult to generate dynamically balanced motion for humanoid robots due to the high number of degrees of freedom and "the size of the space to explore is augmented with the robot velocity and footprint positions" [Dalibard et al., 2013]. Some researchers plan geometric paths by approximating a dynamic trajectory [Dalibard et al., 2013]. However, the drawback of this method is that "some feasible dynamic motions are inherently impossible to compute with this approach" [Dalibard et al., 2013]. There are planners that compute dynamically stable motion trajectories offline [Kuffner et al., 2003, Kanehiro et al., 2008], but these planners require an accurate robot model and change the desired motions in terms of the timings and the trajectories. Adjusting the timings of the desired motion causes the motion to be no longer synchronized to the input signal and changing the motion trajectory may change the meaning expressed.

Falls predictions are generally made through online monitoring methods. These online monitoring methods predict falls by thresholding relevant physical attributes such as angular momenta [Kajita et al., 2003]) or determine stability by tracking the position of the zero moment point (ZMP) [Czarnetzki et al., 2010] or center of pressure (CoP) such that the CoP stays within the support polygon. Others analytically model the robot's dynamics to determine if a fall will occur [Kuffner et al., 2003, Borovac et al., 2011]. However, these methods do not scale well to humanoid robots with complex geometries and high degrees of freedom. Others adopt a data-driven approach where sensor data of stable and unstable trajectories are classified to determine if a fall will occur since it is difficult to model real world variables such as wear and tear [Höhn and Gerth, 2009, Kalyanakrishnan and Goswami, 2011]. The prediction is done during the execution before a fall occurs whereas we do not execute any motion sequence to make a prediction. Searock et al. monitored the sensor readings of a dynamically balancing Segway RMP to detect the onset of a robot's failure by using supervised learning techniques to create a classifier [Searock et al., 2005]. Our approach is to use sensor values from previous executions of single

motions and interpolations to predict if a motion sequence is stable offline. We do not execute or monitor the execution of the motion sequence on the robot. We also do not require a model of the robot to analyze its dynamics or model its environment.

Some researchers adopt fall avoidance methods where falls are avoided using reflex motions to stop the fall [Höhn et al., 2006, Renner and Behnke, 2006, Petri et al., 2013], or to execute a controlled falling motion [Höhn and Gerth, 2009] to reduce the impact of falls. Others determine if there is a need to insert additional motions before a fall is apparent [Noritake et al., 2006]. However, executing reflex motions or inserting additional motions that override the intended motions [Höhn et al., 2006, Renner and Behnke, 2006] will change the motions that convey the meanings of the input signal and are synchronized to the input signal. We want to execute sequences of motions on the robot without the robot falling or appears to be unstable. If we use reflex motions or perform controlled falling, the animation is interrupted. Changing the intended motions may also alter the meanings expressed by the motions.

Fall avoidance methods reduce the damage done to the robot by predicting when falls will occur and perform motions to reduce the impact of falls. Fall avoidance are triggered in the midst of the execution of a motion sequence and only slightly before the fall occurs. Fall avoidance methods cannot predict falls before execution and require training data of instances of the robot falling. Fall avoidance methods can be falsely triggered if the training data are insufficient. Even if the falls are predicted accurately, the controlled falling motions may not be executed in time to prevent bad consequences. Therefore, our approach of using offline predictions of falls without execution will be better than trying to reduce the impact of a fall. Fall avoidance should only be used as a last resort to avoid significant damage to the robot when a fall occurs.

Relative Stability

The term – relative stability is a concept used in control systems to determine the range of parameters for a control system to remain stable so that there is room for margin of error [Ogata, 2001]. In the case of relative stability of sequences, we want to select the most stable sequence for the same reason. Research are concentrated on ensuring the stability of a humanoid robot with online monitoring methods to predict falls by thresholding relevant physical quantities (e.g., angular momenta [Kajita et al., 2003]) or determine stability by tracking the position of the zero moment point (ZMP) [Czarnetzki et al., 2010]. Others analytically model the robot's dynamics

to determine if a fall will occur [Kuffner et al., 2003, Borovac et al., 2011]. We build upon our work where we predict falls without a model of the robot and no previous history of the executions of sequences of motions [Tay et al., 2016]. To our knowledge, we have yet to come across literature that evaluates relative stability of sequences, where sequences of motions are compared to determine the most stable sequence.

A Markov Decision Process (MDP) is generally used to model decision making at different states where the outcomes are stochastic and dependent on the current state and the action taken [Bellman, 1957]. The inverted pendulum problem is solved using a MDP with reinforcement learning whereby the policy determines the best action to execute based on the current state [Sutton and Barto, 1998]. Researchers have also used the inverted pendulum model where equations for the dynamics of the humanoid robots are determined for gait planning [Kajita et al., 2001] and push recovery [Stephens and Atkeson, 2010] in order for the robot to remain stable. Instead of using the inverted pendulum model for equations on the dynamics of the robot, we apply the approach of modeling the humanoid robot as an inverted pendulum and using a MDP where the states of the robot are discretized and the actions are the motions executed.

Chapter 8

Conclusion

This chapter presents the contributions of this thesis and discusses the potential applications and several directions for future work that build on this thesis.

8.1 Contributions

In this thesis, we set out to address the following question:

> In order to autonomously animate a humanoid robot given an input signal, how do we ***r****epresent* motions, automate ***m****appings* between motions and meanings, ***s****elect* the relevant motions and consider the audience's preferences, ***s****ynchronize* motions to the input signal and to determine a ***s****table* sequence of motions?

This thesis addresses the question with the following contributions:

- **Keyframes**
  We formally define two types of keyframes – fixed and variable keyframes for parameterized motion primitives. A fixed keyframe is represented with fully defined joint angles, whereas a variable keyframe is represented with joint angles that are varied based on the parameter, α.

- **Motion Primitives**

  Robot motions are parameterized motion primitives that are instantiated. We explain how parameters in motion primitives are instantiated to form a humanoid robot's whole body motion. We define two types of motion primitives – general motion primitive and spatially targeted motion primitive. A general motion primitive is made up of fixed keyframes with two parameters to adjust the duration of the motion primitive and the number of times the motion primitive is repeated. A spatially targeted motion primitive (STM) is made up of variable keyframes and is a parameterized motion primitive, where one of the parameters defines the target the STM is directed at. A STM is directed at a defined point or direction or repeated by changing the parameters in a STM. We also contribute an algorithm, DeterminePoseForSTM, to instantiate a STM. The formalization of a STM reduces the number of motion primitives defined in the motion library as we do not have to define different motion primitives for different targets or define motions with varying changes in joint angles, e.g., the robot nods by varying its head pitch angle from 5° to -5° to 5° compared to another nod where the pitch angle is varied from 10° to -10° to 10°.

- **Motion Primitives Categories**

  We propose using the robot's body parts to categorize motion primitives so that variations of the whole body motion can be generated. We show that we generate many whole body motions for a dancing robot using a small motion library. The approach of using body part categories also reduces the number of motion primitives in the motion library.

- **Pre-processed Input Signal**

  We formally define a pre-processed input signal. We describe the relationship between the labels of the input signal and the labels of the motion primitives. We explain how the labels of the input signal are used to select relevant motion primitives.

- **Autonomous Mappings between Motions and Labels**

  We describe how manual mappings between motions and labels become tedious when the motion library grows. We contribute an approach, LeDAV, to identify features in the motions using emotional static poses collected to automatically map motions to labels. Specifically, we use the emotional static poses collected to identify the similarity between the keyframes of the motions and the emotional static poses and use a weighted similarity to assign emotional labels to the motion.

- **Metric to determine similarity between motions**
  In situations where the features of the new motions are unavailable and cannot be autonomously labeled, we identify similar motions in the motion library to assign existing labels to the new motions. We evaluate multiple variations of motion trajectory metrics to determine the similarity between motions, and compare the different metrics and present the efficacy of each distance metric using precision and recall. We determine the best metric to be EuclideanJoint in terms of precision, recall and computational complexity.
- **Probabilistic Selection and Synchronization**
  We contribute an approach – CEN – to probabilistically select relevant motion primitives for each label of the input signal using three factors – continuity, emotion and normalization. We also demonstrate how to synchronize the sequence of motion primitives to the beats of the music by varying the parameter, β. For probabilistic selection and synchronization, we use an autonomous dancing robot as an example.
- **Selection and Synchronization using Weighted Criteria**
  We select relevant motions based on an existing similarity function defined for text labels. We synchronize each motion primitive to the start of the corresponding label in the input signal. We use a weighted criteria to rank the sequences and select the best sequence. For selection and synchronization using weighted criteria, we use an autonomous storytelling robot as an example.
- **Selection using Audience Preferences**
  We aim to select the best motion sequence given the list of possible motion sequences for an input signal, and noisy observations of the audience preference using the audience model we created. We investigate how to determine the audience preference of the individual motions from the audience preference of motion sequences. We also consider the effects of audience 'boredom' with a degradation model. We contribute an approach – MAK – that selects the sequence to query the audience for the rating, updates the estimates of the preference of the individual motions, and repeats the process until convergence. We show that MAK outperforms the least-squares benchmark with and without the effects of audience 'boredom'.
- **Predicting the Stability of a Motion Sequence with No Prior Execution**
  We contribute ProFeaSM, an algorithm that predicts the stability of a motion sequence with

no data on prior executions of the sequence. We collect sensor data on the execution of the individual motions and the interpolations between pairs of motions. Using these sensor data, we predict if the sequence of motions is stable using ProFeaSM. We demonstrate the efficacy of ProFeaSM in simulation and on the real robot.

- **Predicting Relative Stability of Motion Sequences Using Prior Executions**
  We introduce relative stability, that refers to the comparison of how stable a sequence is compared to other sequences. Relative stability is useful when we want to determine the most stable sequence so as to increase the probability that the humanoid robot continues to animate without interruptions such as a fall. We contribute an approach, RS-MDP, that determines the relative stability of motion sequences using a MDP generated from previous executions of motion sequences. RS-MDP then predicts the relative stability of new motion sequences. RS-MDP does not require a model of the robot, compared to existing algorithms to determine if a robot falls or is stable. We demonstrate that RS-MDP outperforms two baseline comparison methods, RightAfter and Anytime.
- **Complete Algorithm to Autonomous Animation for Humanoid Robots**
  We contribute a complete algorithm, AAMPS, which captures the meaning of the signal by selecting relevant motions and determines the best sequence based on a weighted criteria comprising the stability of sequences and the audience preferences. We explain how AAMPS is made up of the solutions proposed for each of the five core challenges: **R**epresentation, **M**appings, **S**election, **S**ynchronization and **S**tability (R-M-S^3).
- **Autonomous Animation for Two Types of Input Signals**
  Throughout the thesis, we demonstrate our work using two types of input signals – music and text-to-speech – and show that we autonomously animate a dancing NAO humanoid robot and a storytelling NAO humanoid robot.

8.2 Potential Applications

This thesis is motivated by the laborious work put in to manually animate a humanoid robot stably given an input signal. We break down the problem into five core challenges and present our solutions to these five challenges and a complete algorithm that automates the process of animating a humanoid robot stably given an input signal. We also take into account the audience

preferences of motions used in the animation.

We summarize the requirements to autonomously animate a humanoid robot R using our approach and algorithms:

- The physical constraints of R are known, e.g., joint angular limits and joint velocity limits.
- A labeled motion library is available and the motion primitives in the motion library are defined for R using our representations. The interpolation times for each motion primitive are defined.
- If some motions in the library are not labeled, there are examples of labeled motions that are used by our autonomous mapping algorithms to map existing labels to the new motions.
- The input signal is pre-processed to extract the labels that are animated and the timings of these labels.
- The similarity function $\mathbb{S}$ for determining the similarity of the labels is defined.
- The synchronization function $\mathbb{H}$ is defined.
- The sensor data required for predicting the stability of R are collected on R and R is used to animate all motion sequences.
- The audience preferences of motion sequences are assigned using our audience model.

Given the contributions of this thesis, manual generation of a sequence of motions for an input signal is replaced by an autonomous approach of selecting a motion sequence to animate the humanoid robot.

We hope that our work will be useful to robot choreographers and facilitate their work to animate the humanoid robot, instead of starting from scratch for different input signals. Thus, robot choreographers can use our approach as a guide to determine the motion sequence using our criteria or customize the criteria based on their needs. They can also use our work as a starting point before they refine the final motion sequence for the humanoid robot to animate. Robot choreographers who have previously manually animated humanoid robots can build their motion library using their previous manual animations of the humanoid robot.

Robot choreographers who have a large library of motion primitives can use our autonomous mapping algorithms to determine if there are more mappings between motion primitives and labels without having to manually go through all the motions and labels. Choreographers can also choose between a probabilistic approach or a set of criteria to select the motions. Choreographers

can use our audience model to determine the most preferred motion sequence assuming that the audience assigns preferences according to our model. Robot choreographers can also minimize the execution of unstable sequences using our algorithm to predict the stability of a motion sequence and to determine the most stable sequence using the prediction of relative stability.

8.3 Future Work

This thesis presents new algorithms and approaches to autonomously animate humanoid robots in general. We enumerate a few directions for future work.

- We defined a keyframe-based representation for motions. Motions such as stable walking require modeling the dynamics of the robot and cannot be represented with fixed keyframes. We can investigate how stable walking is incorporated so that the humanoid robot walks around and gestures at the same time.
- We can relax the constraint that a motion is only synchronized to the input signal when the start of a motion corresponds to the start of the label the motion is expressing. For example, a motion is still synchronized to a sequence when the start of the motion is within a certain time duration of the start of the corresponding label in the input signal.
- We can investigate the similarity of labels so as to map new labels to existing motions. For example, the label "happy" is a synonym to the label "cheerful". Hence, motions with the label "happy" can also be mapped to the label "cheerful".
- Labels such as "wave" can convey different meanings in different text-to-speech input signals and require different motions to convey the meaning, e.g., "wave goodbye" versus "wave a wand". We currently model the audience preference using a rating for each label-motion pair. With our approach, the rating for the same label-motion pair will vary across different text inputs. In the future, the context of the sentence can be considered.
- We evaluated eight distance metrics that are varied along three dimensions – Euclidean versus Hausdorff distances, joint angles versus POIs, Original versus Mirrored motions. We can consider other distance metrics such as dynamic time warping, longest common subsequence, etc. These distance metrics can also be evaluated to compare their performance in precision, recall and computational complexity.

- The original motion can be labeled with "waving with his left hand" and the mirrored motion can use the same label by replacing the word "left" to "right" and vice versa. We can keep a dictionary of such words and motion features so as to replace the corresponding words in the labels.
- We consider using only body angles to predict the stability and relative stability of motion sequences. We can explore using other sensor data such as foot pressure sensor readings, which are commonly available in humanoid robots.
- We use RS-MDP to predict the relative stability of sequences. For instances where a particular motion in the motion sequence has not been performed at a particular state, we take a "pessimistic" view that the robot will fall and give it the minimum reward. For future work, we can adopt an "optimistic" view that the robot is stable and give it the maximum reward so as to try out different motion sequences to update the MDP.
- This thesis contributes to the autonomous animation of a single humanoid robot. We can build upon this thesis to generate multiple motion sequences for multiple robots.
- We can explore changing the input signal to allow time for a highly preferred motion to be used when the duration of the motion is too long and cannot be synchronized to the input signal. For example, we can insert pauses to the text-to-speech input signal.

8.4 Concluding Remarks

This thesis addressed the five core challenges for the thesis question to autonomously animate humanoid robots. These five core challenges are **R**epresentation, **M**appings, **S**election, **S**ynchronization and **S**tability (R-M-S^3). We introduced representations for motions, input signals and labels. We contributed algorithms to autonomously map motions to labels and evaluated metrics to determine similar motions. We selected relevant motions using the similarity between labels of the motions and the labels of the input signal and synchronized the motion sequence to the input signal. We demonstrated different approaches to consider selection of relevant motions, i.e., probabilistic selection and selection via weighted criteria. We described how we consider audience preferences and stability of motion sequences. We demonstrated our approach and algorithms using a NAO humanoid robot and a simulated NAO humanoid robot in Webots 7 [Webots, 2014] for two types of input signals – music and text-to-speech.

Appendix A

List of Symbols

The list of symbols here is used as a reference to the symbols used throughout this thesis.

Symbol	**Description**	**Chapter(s)**
m	Motion primitive	2, 3, 4, 5, 6
M	Set of motion primitives consisting general and spatially targeted motion primitives	2, 3, 4, 5, 6
l	A label of a motion primitive m	2, 3, 4
L	Set of all labels	2, 3, 4
M^L	Set of labeled parameterized motion primitives in the motion library	2, 3
s	A pre-processed input signal	2, 3, 5
S	Set of all pre-processed input signals	2, 3, 5
$\mathbb{X}$	Function to determine if a mapping between a motion primitive m and label l^m exist	2, 3, 4
u	A sequence of motions	2, 5, 6
U	A set of sequences of motions	2, 5, 6
u^s	Sequence of motion primitives for the signal s	2, 5, 6

U	Set of all sequences	2, 5, 6
U^s	Set of sequences for signal s	2, 5, 6
$\mathbb{S}$	Function to determine similarity of meanings between labels	2, 4
lm	Label-motion pair	2, 5
$\mathbb{A}$	Function to observe the audience rating of a sequence	2, 5
$\mathbb{H}$	Function to determine if a sequence of motion primitives is synchronized to the input signal	2, 5
μ	Motion primitives are selected with a similarity value larger or equal to this variable	2, 5
$\mathbb{F}$	Function to determine feasibility (stability) for a sequence of motion primitives	2, 6
$\mathbb{U}$	Function to determine relative stability of a sequence given the set of possible sequences	2, 6
$\mathbb{SS}$	Function to determine relevant motions, generate possible sequences, synchronize motions and discard unsynchronized sequences	2
u^{ss}	Stable synchronized sequences of instantiated motion primitives for signal	2
U^{ss}	Set of stable synchronized sequences of instantiated motion primitives for signal	2
$\hat{A}$	Audience rating	2, 5
$\hat{A}^{\max}$	Maximum audience rating	2, 5
$\hat{A}^{\min}$	Minimum audience rating	2, 5

$\mathbb{P}$	Function to return the normalized audience rating	2, 5
γ	Weight to the criterion – relative stability	2
R	Robot	3
D	Number of actuated joints or degrees of freedom	3
J_d	Robot's joint with index d	3
θ_d	Robot's joint angle with index d	3
θ_d^{min}	Minimum angle of the joint d	3
θ_d^{max}	Maximum angle of the joint d	3
$\dot{\theta}_d^{max}$	Maximum velocity of the joint d	3
ζ	D-dimensional configuration space of R	3
k	Keyframe	3
k^f	Keyframe with fixed joint angles	3
K^f	Set of fixed keyframes	3
α	Factor to the amplitude of the relative changes in a variable keyframe	3
k^v	Variable keyframe	3
$\tilde{\theta}$	Relative joint angle change	3
$\tilde{\theta}^{min}$	Minimum relative joint angle change	3
$\tilde{\theta}^{max}$	Maximum relative joint angle change	3
K^v	Set of variable keyframes	3
K	Set of all keyframes consisting fixed and variable keyframes	3

m^g	General motion primitive	3
M^g	Set of general motion primitive	3
m^{st}	Spatially targeted motion primitive	3
M^{st}	Set of spatially targeted motion primitive	3
$\mathtt{G}$	Number of primitives in a general motion primitive	3
$\mathcal{M}^g$	Primitive in a general motion primitive	3
β	Factor to interpolation time	3
N	Number of times a motion primitive is repeated	3
$t_{n-1,n}$	Time to interpolate between k_{n-1} and k_n	3
$\mathbb{T}$	Time computation function to determine minimum duration required to interpolate from one keyframe to another keyframe	3
$\mathtt{S}$	Number of primitives in a spatially targeted motion primitive	3
$\mathcal{V}$	Vector that defines the direction m^{st}'s first keyframe is directed at	3
P^s, P^e	P^s and P^e are two ego-centric coordinates used to define the vector $\mathcal{V}$	3
$\mathcal{M}^{st}$	Primitive in a spatially targeted motion primitive	3
D^{min}, D^{max}	Minimum and maximum distance the STM can be at, so that if the STM's distance to the target is within the defined range, the STM is executed	3
$\mathcal{P}_o^R$	Robot's original global position	3

O_o^R	Robot's original global orientation	3
$\mathcal{T}$	STM's target for the robot to face, can be a point or vector defined in global coordinates	3
$\mathtt{t}^s$	STM's target point or the starting point of $\mathcal{T}$	3
$\mathtt{t}^e$	STM's target point or the ending point of $\mathcal{T}$	3
ω_a	Angular tolerance between current m^{st}'s orientation and desired orientation	3
O^{st}	Current m^{st}'s orientation	3
$O^{\mathcal{T}}$	Desired m^{st}'s orientation	3
P	A point P	3
$\mathcal{P}_f^R$	Robot's final global position	3
O_f^R	Robot's final global orientation	3
D^{mean}	The mean distance between minimum and maximum distance the STM can be at	3
P^{gs}	$\mathtt{t}^s$ in global coordinates	3
P^{ge}	$\mathtt{t}^e$ in global coordinates	3
f	Motion feature	3
F	Set of motion features	3
c	A motion primitive category	3
C	Set of all motion primitive categories	3
$\mathbb{C}$	Function determine if a feature f is assigned to the category c	3
b	Body part	3

c^b	Body part category	3
J^{c^b}	Name or index of joint in category c^b	3
l^s	A label assigned to the pre-processed input signal s	3
L^s	A set of labels assigned to the pre-processed input signal s	3
l^m	A label assigned to the motion primitive m	3
L^m	A set of labels assigned to the motion primitive m	3
$\mathcal{I}$	Number of primitives in a pre-processed signal s	3
$\mathcal{S}$	Signal Primitive	3
t^{ss}	Start time of label in signal	3
t^{se}	End time of label in signal	3
$d^{l_i^s}$	Duration of label l_i in signal	3
ε	Increment to the interpolation time between keyframes	3
t^{ms}	Starting time of the motion primitive	3
em	Emotion	4
EM	Set of emotions	4
SP	Emotional static pose	4
a	Activation value in the activation-valence label	4
v	Valence value in the activation-valence label	4
AV	Function to determine the activation-valence label of a motion primitive	4
$\theta_i^{\text{original}}$	Joint angle for i^{th} joint in the original motion	4

θ_i^{mirror}	Joint angle for i^{th} joint in the mirrored motion	4
dt	Duration of motion	4, 6
ED	Euclidean distance between joints	4
EP	Euclidean distance between POIs	4
DM	Metric for motion similarity	4
TP	True positive	4
FP	False positive	4
FN	False negative	4
$\mathcal{C}$	Continuity factor	5
$\mathcal{E}$	Emotion factor	5
N	Normalization factor	5
DE	Cartesian distance between two activation-valence labels	5
λ	Maximum time multiplier	5
η	Time multiplier for body part category	5
$\mathcal{R}$	Ranking	5
χ	Number of criteria	5
w	Weights to criteria	5
LM	Set of label-motion pairs	5
am^i	Audience rating for i^{th} label-motion pair	5
as^i	Audience rating for the i^{th} sequence of motions	5
$\widetilde{am}^i$	Audience rating for i^{th} label-motion pair in the model or mean audience rating in the model	5

$\widetilde{vam}^i$	Variance audience rating for i^{th} label-motion pair	5
$\widetilde{AM}$	Set of mean audience ratings in the model	5
$\widetilde{VAM}$	Set of variance audience ratings in the model	5
DF	Degradation factor	5
R_k	Variance of observation noise	5
w_i^j	Weight to the audience rating of the i^{th} label-motion pair in j^{th} sequence	5
W^U	Set of weights to the audience ratings of the label-motion pairs	5
MI	Maximum iterations for the stopping condition of MAK	5
ϵ	Maximum absolute difference between current and previous model of audience ratings for the stopping condition of MAK	5
maxAbsoluteDiff	Function to determine ϵ	5
λ	Absolute difference between current and previous model of mean audience ratings	5
λ^v	Absolute difference between current and previous model of variance audience ratings	5
v^i	Number of times the i^{th} label-motion pair is viewed	5
Υ	Absolute difference between ModelBest and FindBestAndGetFromBlackBox	5
ρ	Absolute difference between BestFromBlackBox and FindBestAndGetFromBlackBox	5
Ψ	Time steps	6

f	Frequency	6
ψ	Time stamp	6
ς	Number of iterations	6
ba	Body angles	6
bax	Body angle X	6
bay	Body angle Y	6
$\overline{\text{bax}}$	Median of body angle X	6
$\overline{\text{bay}}$	Median of body angle Y	6
vel	Body angle velocities	6
acc	Body angle accelerations	6
BA^x	Set of body angle X	6
BA^y	Set of body angle Y	6
$\mathbb{D}$	Variable to discretize body angles	6
NS	Number of states in the RS-MDP	6
u^H	Past sequence	6
U^H	Set of past sequences	6
U^{Train}	Set of training sequences	6
U^{Test}	Set of test sequences	6
$\overrightarrow{v}$	Vector of states for RS-MDP	6
$\overrightarrow{V}$	List of vector of states for RS-MDP	6
Ω	Function that maps sequences into the respective state vectors	6

bas	Current state	6
bas$'$	Next state	6
BAS	Set of states	6
BASP	Set of states and probabilities	6
a	Current action	6
A	Set of actions	6
RF	Reward function	6
$TC(\text{bas}, a, \text{bas}')$	Number of times the robot's state transitions from bas to bas$'$ after taking action a	6
$\widetilde{TC}$	Set of all transition counts	6
$TP(\text{bas}, a, \text{bas}')$	Probability of the robot's state transitions from bas to bas$'$ after taking action a	6
$\widetilde{TP}$	Set of transition probabilities	6
er	Expected reward for sequence	6
ER	Set of expected rewards for set of sequences	6
aer	Average expected reward for sequence	6
AER	Set of average expected rewards for set of sequences	6
$\text{RA}_i^{\text{fall}}$	Number of times the robot falls right after i^{th} action	6
$\text{RA}_i^{\text{stable}}$	Number of times the robot is stable right after i^{th} action	6
RAP_i	Probability the i^{th} action is stable for the metric RightAfter	6
$\widetilde{RAP}_i$	Average probability the i^{th} action is stable for the metric RightAfter	6

RAPS_i	Probability the i^{th} sequence is stable for the metric RightAfter	6
$\text{AT}_i^{\text{stable}}$	Number of times the i^{th} action is found in a stable sequence	6
AT_i	Number of times the i^{th} action is found in an unstable sequence	6
ATP_i	Probability the robot is stable after the i^{th} action for the metric Anytime	6
$\widetilde{ATP}_i$	Average probability the robot is stable after the i^{th} action for the metric Anytime	6
ATPS_i	Probability the i^{th} sequence is stable for the metric RightAfter	6
rs_i	Reward for i^{th} sequence	6
RS	Set of rewards for i^{th} sequence	6
ars_i	Actual relative stability for i^{th} sequence	6
ARS	Set of actual relative stability for sequences	6
TRS	Function to evaluate the actual relative stability	6
bin	Index of the state for body angle	6
ϖ	Divisor to determine state for body angles	6
$\mathbb{Y}$	Function to convert body angles into states	6
bin^X	Index of bin for the body angle X	6
bin^Y	Index of bin for the body angle Y	6
g	Actual relative stability value	6

h	Predicted relative stability value	6
G	Set of actual relative stability values	6
H	Set of predicted relative stability values	6

Table A.1: List of Symbols.

Appendix B

Fifty Two Words and Corresponding Number of Motions

Word	**Number of motions**	**Word**	**Number of motions**
Wolf	3	Yes	6
Look	3	Point	6
Small	3	Cat	3
No	14	Nod	2
Raise	4	Happy	8
Chicken	2	Left	2
Shake	4	Think	3
Surprised	6	Bird	4
Right	2	Bow	2
Wipe	4	Fear	5
Baby	3	Goodbye	5
Sneeze	2	Clean	4
Sad	8	Eyes	3
Hello	4	Kick	4

Sing	3	Angry	6
Ear	3	Wave	6
Dance	8	Throw	2
Disgust	6	Mouth	3
Eat	7	Clap	3
Big	3	Proudly	5
Car	3	Run	4
Laugh	4	Drive	2
Up	3	Lion	2
Airplane	2	Down	3
Push	3	Pull	3
Stop	3	Open	1

Table B.1: List of Fifty Two Words and Number of Motions Per Word.

Appendix C

Twenty Stories

These are the twenty stories created using fifty two words (labels) selected. There are at least two labels per sentence and five sentences per story. Each label is highlighted in bold.

Story 1

A **lion**, **wolf**, **cat** and **chicken** were stranded on an island after a ship to the London Zoo capsized. The **cat ate** the **chicken**. The **lion ate** the **cat**. The **wolf** pleaded to the **lion** in **fear** to spare his life. The **lion** was **disgusted** at his timidness.

Story 2

An **airplane** heading towards New York from Paris experienced some flight problems after going **up** in the air. After much struggle, the pilots were able to maintain the **airplane** in flight and calmed everyone **down**. The **airplane** finally touched **down** in Paris. Ambulance has arrived to treat passengers who experienced trauma and shock, especially those **shaking** with **fear**. After the ordeal was over, the pilots **shook** hands to recognize each other's efforts in saving five hundred lives onboard the **plane**.

Story 3 At the party, Cinderella laid her **eyes** on a young prince and began **dancing** with him. After the dance, she **happily** took a drink from the waiter and **nodded** to thank him. She **sneezed** and accidentally spilled the prince's **eyes** with wine. The poor prince had to **clean** and

wipe himself dry. Cinderella was **disgusted** at her carelessness and **left** the party in **fear** of embarrassing herself further.

Story 4

Dan and Jill went hunting **happily**, hoping to find something to **eat**. Dan **pointed** to a **wolf** in the distance. The **wolf** saw them and **ran** towards their direction. Jill **raised** and aimed her gun at the **wolf**. The **wolf** was shot dead, and the scene **disgusted** Dan.

Story 5

A couple had a **baby** together and were very **happy**. He had the **eyes** of the father and **mouth** of his mother. One day, a **wolf** went into the house and took the **baby** away. The couple were deeply **saddened** and could hardly **eat** for days. The **wolf** raised the **baby** like its own child.

Story 6

Mary greeted Barry at the circus with a welcoming **hello happily**. They went to watch the famous clown show and saw the clown's dog **run** from **left** to **right**. They could not **stop laughing** and **clapped happily**. They were **surprised** when the clown suddenly **pushed** the dog **down** into a pool of water. The dog disappeared before their **eyes** and the clown **proudly** showed that the dog was actually inside a box.

Story 7

The kindergarten children were having fun and **running** around **happily**. The teacher taught them a **song** and **dance**. It goes If you are **happy** and you know it **clap** your hands. If you are **happy** and you know it **nod** your head. If you are **happy** and you know it and you really want to show it, then you can go on to **clap** your hands.

Story 8

I was **driving down** to the countryside when the radio played the song ”Love Me” by Colin Raye. The lyrics of the **song** reminded me of the **happy** times spent with my grandmother. Grandma had already **left** my world and I **thought** about my times with her. As I reminisce my childhood in **happiness**, **sadness** of nostalgia still struck me. I **wiped** my tears off as they dripped **down** my face.

Story 9

As students of Dunman High school, **bowing** to say **hello** to our teachers has long become a custom in our lives. However, when the teacher we **bowed** to dismisses us with little respect, we will be very **angry**. Teachers who say **hello** back makes us **happy**. We **thought** that **waving hello** will be the same. We should **raise** this **point up** the next time we meet the teachers.

Story 10

Joe was **surprised** to see Lee greeting him with a **hello**. He had been **sad** and **angry** these few days. What thing could have happened to make him so **happy** to **sing** today? Joe went into deep **thought** and **looked** around. Just then, he heard someone **laugh** and **clap** in the distance.

Story 11

One day, Tommy saw a **small bird** on the tree. He **threw** a stone at it and smiled **proudly** as it fell **down** the tree. Tommy walked over to where the injured bird has fallen and **kicked** it cruelly with his **right** foot. A **car pulled** over and off alighted a tall lady screaming “**NO**” at the top of her lungs. Tommy’s **mouth opened** in horror when he **looked up** and saw the familiar face of the discipline master.

Story 12

Dad **drove** Tom to school in his **car**. He **pulled** over the car and said **goodbye** to Tom. Tom **looked up**, thanked his father and got out of the **car** with his **small** bag. He smiled at his son **proudly**, and **drove** away. **Pushing** his **right** foot on the accelerator, he made a **right** turn and

slowly accelerated away.

Story 13

Tim's dad **drove** to Tim's school with a wide smile on his face that stretched from **ear** to **ear**. He beamed **proudly** as he **pulled open** the door of the principal's office. Earlier, he was informed that his son was the only student who had stayed behind and voluntarily **cleaned** the **big** mess made after assembly. It was a **small** deed but it had a **big** impact on the other students. As he bid **goodbye** to the principal, the chirping of the **birds** seemed to be extremely melodious and he found himself humming along the tune.

Story 14

Jim was bored one day as he strolled along the streets so he **kicked** the **baby birds** hidden in the bush. He **opened** his **small** arms wide and cheered **proudly** as he saw one of them shivering in pain. Suddenly he felt a **pull** on his **ear**. He **pushed** away the hand and **looked up** immediately, feeling **angry**. It was his turn to shiver when he saw a **big** sized lady glaring at him, making him have **no** other **thought** but regret for his action.

Story 15

Tim exclaimed, "Your pet **bird** is really **big** and pretty!". Jov replied, "**Yes**, it used to be so much **smaller** when I first bought it though". The **bird kicked** and **danced** a little in her cage **proudly** as she heard Tim compliment her. Tim: "**Look**, that's my father's **car** over there. **Goodbye** Jov, my father is **driving** her to the vet now to get a checkup for her loss of appetite recently".

Story 16

Tim and Tom got into a fight in school and they have been **kicking** and **pushing** each other for ten minutes since the fight started. Tom got hit by a **big** blow and his **mouth** started bleeding, leading to a **big** crowd gathering around them. Someone **pointed** to the **right** and they both

looked to find their form teacher **running** towards them. The teacher **pushed** past the crowd and brought Tom to her office to have his wound **cleaned**, before reprimanding them harshly. She then **drove** Tom to the hospital to have his wound on his **mouth** properly attended to.

Story 17

Cleaning one's **mouth** is actually an important thing to do. To **clean** it the **right** way, one must brush their teeth after meals and we can say **goodbye** to decaying tooth. This way, they can **proudly** show off their teeth and **look** great. Brushing one's teeth often helps remove any **big** or **small** pieces of food **left** in the **mouth** cavity. **Cleaning** one's **mouth** should not be overlooked or else problems will start **kicking** in one after another.

Story 18

Mary **pulled** out a piece of tissue from the tissue box as she **sneezed** for the hundredth time that day. She let out a sigh as she **looked** at the mess on the sofa, having **no** other choice but to **clean** it before the tissue paper balls roll onto the floor. Her **mouth** felt exceptionally dry and she felt this tingling sensation in her **ears**. She **threw** her thermometer on the sofa in frustration as she saw her temperature going yet another half a degree **up**. She decided that she could not take it anymore and **drove** to the **small** clinic nearby to get some treatment for her terrible cold.

Story 19

He **looked up**, caught the ball and **threw** it into the net. Cheers of **yes** could be heard all around in the indoor stadium as their score went **up** yet another **point**. The national tournaments were going on, a **big** event that **no** one would ever want to miss. **Right** at this moment, a whistle ringed in our **ears**. This signifies yet another win for our school team, and we smiled **proudly**, knowing that this **big** confidence boost will aid in our last round of competitions, the finals.

Story 20

Hundreds of **cars drove** into the shopping mall today. Fans queued for hours and **pushed** past the crowd just to have a quick **look** at their favorite writer. The writer beamed **proudly** as he saw his fans carrying **big** piles of his bestsellers, queuing orderly on his **right** to wait for his autograph. The queue stretched all the way **up** to the second floor of the bookshop, yet the place was kept quite **clean**. There was also **no** presence of crying and **kicking small** kids as there have usually been for his previous book signing events.

Appendix D

Twenty Four Static Poses for Paul Ekman's Six Basic Emotions

Happy

	Happy1	Happy2	Happy3	Happy4
Front view				
Side view				

Figure D.1: 4 Happy Static Poses.

Table D.1: Heights and Tilts for 4 **Happy** Static Poses.

Poses	Height	Tilt
Happy1	3	2
Happy2	5	2
Happy3	5	3
Happy4	5	3

Sad

	Sad1	Sad2	Sad3	Sad4
Front view				
Side view				

Figure D.2: 4 Sad Static Poses.

Table D.2: Heights and Tilts for 4 **Sad** Static Poses.

Poses	Height	Tilt
Sad1	1	5
Sad2	1	5
Sad3	3	3
Sad4	2	4

Anger

	Anger1	Anger2	Anger3	Anger4
Front view				
Side view				

Figure D.3: 4 Anger Static Poses.

Table D.3: Heights and Tilts for 4 **Anger** Static Poses.

Poses	Height	Tilt
Anger1	4	4
Anger2	5	3
Anger3	5	5
Anger4	4	4

Surprise

	Surprise1	Surprise2	Surprise3	Surprise4
Front view				
Side view				

Figure D.4: 4 Surprise Static Poses.

Table D.4: Heights and Tilts for 4 **Surprise** Static Poses.

Poses	Height	Tilt
Surprise1	5	2
Surprise2	4	1
Surprise3	5	2
Surprise4	4	2

Fear

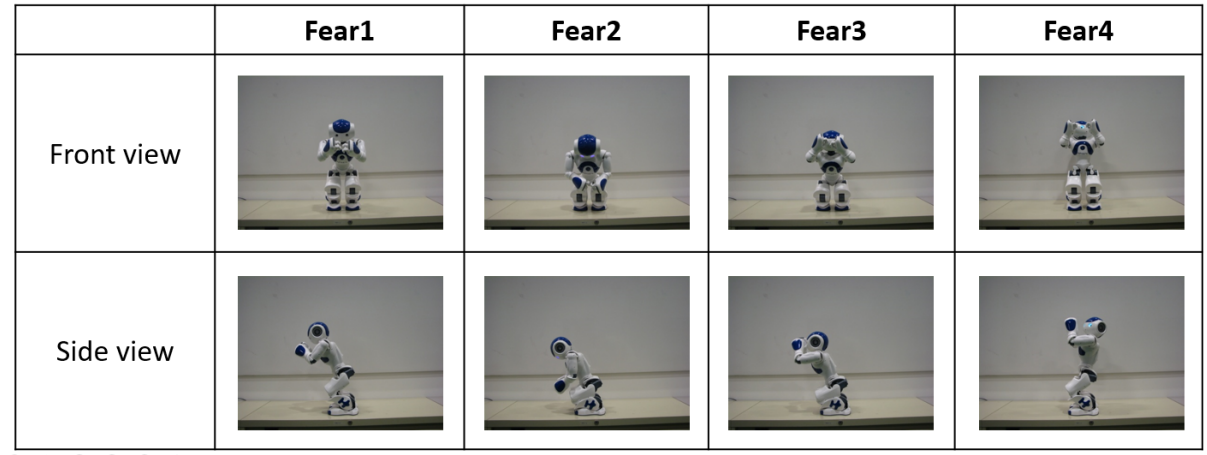


Figure D.5: 4 Fear Static Poses.

Table D.5: Heights and Tilts for 4 **Fear** Static Poses.

Poses	Height	Tilt
Fear1	3	5
Fear2	1	5
Fear3	1	5
Fear4	3	2

Disgust

	Disgust1	Disgust2	Disgust3	Disgust4
Front view				
Side view				

Figure D.6: 4 Disgust Static Poses.

Table D.6: Heights and Tilts for 4 **Disgust** Static Poses.

Poses	Height	Tilt
Disgust1	2	4
Disgust2	5	2
Disgust3	4	4
Disgust4	1	2

Bibliography

[Abbeel and Ng, 2004] Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press. 7.3

[Addo and Ahamed, 2014] Addo, I. D. and Ahamed, S. I. (2014). Applying affective feedback to reinforcement learning in zoei, a comic humanoid robot. In *RO-MAN*, pages 423–428. IEEE. 7.3

[Akrour et al., 2012] Akrour, R., Schoenauer, M., and Sebag, M. (2012). APRIL: Active preference learning-based reinforcement learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 7524, pages 116–131. Springer. 7.3

[Aldebaran Robotics, 2013] Aldebaran Robotics (2013). https://community.aldebaran-robotics.com/doc/1-14/. NAO Software 1.14.5 documentation. 6.1.2

[Aldebaran Robotics, 2014a] Aldebaran Robotics (2014a). Effector and chain definitions. 4.2.1

[Aldebaran Robotics, 2014b] Aldebaran Robotics (2014b). NAO actuator and sensor list. (document), 4.5

[Aldebaran Robotics, 2014c] Aldebaran Robotics (2014c). NAO software. 3.3

[Apache Commons Math, 2015a] Apache Commons Math (2015a). KendallsCorrelation. 6.2.4

[Apache Commons Math, 2015b] Apache Commons Math (2015b). SpearmansCorrelation. 6.2.4

[Bartneck et al., 2009] Bartneck, C., Kuli, D., Croft, E., and Zoghbi, S. (2009). Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International Journal of Social Robotics*, 1(1):71–81. 7.3

[Beattie, 2004] Beattie, G. (2004). *Visible thought: The new psychology of body language*. Routledge, New York. 7.1

[Beck et al., 2010] Beck, A., Caamero, L., and Bard, K. (2010). Towards an affect space for robots to display emotional body language. In *Robot and Human Interactive Communication*, pages 464–469. 7.2

[Beck et al., 2013] Beck, A., Caamero, L., Hiolle, A., Damiano, L., Cosi, P., Tesser, F., and Sommavilla, G. (2013). Interpretation of emotional body language displayed by a humanoid robot: A case study with children. *International Journal of Social Robotics*, 5(3):325–334. 7.2

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684. 7.4

[Bennewitz et al., 2007] Bennewitz, M., Faber, F., Joho, D., and Behnke, S. (2007). Fritz – a humanoid communication robot. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*. 7.1, 7.3

[Borovac et al., 2011] Borovac, B., Nikolić, M., and Raković, M. (2011). How to compensate for the disturbances that jeopardize dynamic balance of a humanoid robot? *International Journal of Humanoid Robotics*, 08(03):533–578. 7.4

[Breazeal, 2003] Breazeal, C. (2003). Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59:119–155. 7.1

[Breemen, 2004] Breemen, V. (2004). Bringing robots to life: applying principles of animation to robots. In *Proceedings of the CHI 2004 Workshop on Shaping Human-Robot Interaction*. 7.3

[Breitfuss et al., 2007] Breitfuss, W., Prendinger, H., and Ishizuka, M. (2007). Automated generation of non-verbal behavior for virtual embodied characters. In *Proceedings of the 9th International Conference on Multimodal Interfaces*, ICMI '07, pages 319–322, New York, NY, USA. ACM. 7.3

[Breitfuss et al., 2009] Breitfuss, W., Prendinger, H., and Ishizuka, M. (2009). Automatic generation of non-verbal behavior for agents in virtual worlds: A system for supporting multimodal conversations of bots and avatars. In Ozok, A. and Zaphiris, P., editors, *Online Communities and Social Computing*, volume 5621 of *Lecture Notes in Computer Science*, pages 153–161.

Springer Berlin Heidelberg. 7.3

[Brooks and Arkin, 2007] Brooks, A. and Arkin, R. (2007). Behavioral overlays for non-verbal communication expression on a humanoid robot. *Autonomous Robots*, 22(1):55–74. 7.1

[Calinon et al., 2010] Calinon, S., Sauser, E., Billard, A., and Caldwell, D. (2010). Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2671–2676. 7.2

[Cantrell et al., 2011] Cantrell, R., Schermerhorn, P., and Scheutz, M. (2011). Learning actions from human-robot dialogues. In *Robot and Human Interactive Communication*, pages 125–130. 7.2

[Cassell et al., 1994] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., douville, B., Prevost, S., and Stone, M. (1994). Animated conversation: rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *SIGGRAPH Proceedings of the annual conference on Computer graphics and interactive techniques*, pages 413–420. 7.1, 7.3

[Cassell et al., 2001] Cassell, J., Vilhjálmsson, H. H., and Bickmore, T. (2001). BEAT: The Behavior Expression Animation Toolkit. In *SIGGRAPH Proceedings of the annual conference on Computer graphics and interactive techniques*, pages 477–486. 7.1, 7.2, 7.3

[Czarnetzki et al., 2010] Czarnetzki, S., Kerner, S., and Klagges, D. (2010). Combining key frame based motion design with controlled movement execution. In Baltes, J., Lagoudakis, M., Naruse, T., and Ghidary, S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 58–68. Springer Berlin Heidelberg. 7.4

[Dalibard et al., 2013] Dalibard, S., El Khoury, A., Lamiraux, F., Nakhaei, A., Tax, M., and Laumond, J.-P. (2013). Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. *Int. Journal of Robotics Research*, 32(9-10):1089–1103. 7.4

[de Sousa Junior and Campos, 2011] de Sousa Junior, S. F. and Campos, M. F. M. (2011). Shall we dance? a music-driven approach for mobile robots choreography. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1974–1979. 7.1, 7.3

[Ekman, 1992] Ekman, P. (1992). Are there basic emotions? *Psychological Review*, 99(3):550–553. 4.1.1

[Ellenberg et al., 2008] Ellenberg, R., Grunberg, D., Kim, Y., and Oh, P. (2008). Exploring creativity through humanoids and dance. In *Ubiquitous Robot and Ambient Intelligence*. 7.1

[Ellis, 2006] Ellis, D. (2006). Beat tracking with dynamic programming. *MIREX Audio Beat Tracking Contest sys. desc.* 3.3, 5.1.2

[Erden, 2013] Erden, M. (2013). Emotional postures for the humanoid-robot nao. *International Journal of Social Robotics*, 5(4):441–456. 7.3

[Erdogan and Veloso, 2011a] Erdogan, C. and Veloso, M. (2011a). Action selection via learning behavior patterns in multi-robot domains. In *Int. Joint Conf. on Artificial Intelligence. (IJCAI)*, pages 192–197. 4.2.2, 7.2

[Erdogan and Veloso, 2011b] Erdogan, C. and Veloso, M. (2011b). Action selection via learning behavior patterns in multi-robot domains. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 192–197. AAAI Press. 7.2

[Geppert, 2004] Geppert, L. (2004). QRIO: The Robot That Could. `http://spectrum.ieee.org/robotics/robotics-software/qrio-the-robot-that-could`. [Online; accessed 22-March-2016]. 1

[Grunberg et al., 2009] Grunberg, D., Ellenberg, R., Kim, Y., and Oh, P. (2009). Creating an autonomous dancing robot. In *Proceedings of the 2009 International Conference on Hybrid Information Technology*, ICHIT '09, pages 221–227, New York, NY, USA. ACM. 7.3

[Han et al., 2009] Han, B., Rho, S., Dannenberg, R., and Hwang, E. (2009). SMERS: Music emotion recognition using support vector regression. In *International Society for Music Information Retrieval 2009*, pages 651–656. 3.3, 4.1, 5.1.2

[Haring et al., 2011a] Haring, M., Bee, N., and Andre, E. (2011a). Creation and evaluation of emotion expression with body movement, sound and eye color for humanoid robots. In *Int. Symp. on Robots and Human Interact. Comm. (RO-MAN)*, pages 204–209. 3.3

[Haring et al., 2011b] Haring, M., Bee, N., and Andre, E. (2011b). Creation and evaluation of emotion expression with body movement, sound and eye color for humanoid robots. In *Robot and Human Interactive Communication*, pages 204–209. 7.2

[Hartmann et al., 2005] Hartmann, B., Mancini, M., Buisine, S., and Pelachaud, C. (2005). De-

sign and evaluation of expressive gesture synthesis for embodied conversational agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, pages 1095–1096, New York, NY, USA. ACM. 7.3

[Hartmann et al., 2006] Hartmann, B., Mancini, M., and Pelachaud, C. (2006). Implementing expressive gesture synthesis for embodied conversational agents. In Gibet, S., Courty, N., and Kamp, J.-F., editors, *Gesture in Human-Computer Interaction and Simulation*, volume 3881 of *Lecture Notes in Computer Science*, pages 188–199. Springer Berlin Heidelberg. 7.3

[Höhn et al., 2006] Höhn, O., Ganik, J., and Gerth, W. (2006). Detection and classification of posture instabilities of bipedal robots. In Tokhi, M., Virk, G., and Hossain, M., editors, *Climbing and Walking Robots*, pages 409–416. Springer Berlin Heidelberg. 7.4

[Höhn and Gerth, 2009] Höhn, O. and Gerth, W. (2009). Probabilistic balance monitoring for bipedal robots. *International Journal of Robotics Research*, 28(2):245–256. 7.4

[Honda, 2005] Honda (2005). ASIMO Has New "Home" Inside Disneyland. `http://asimo.honda.com/news/asimo-has-new-home-inside-disneyland/newsarticle_0034/`. [Online; accessed 22-March-2016]. 1

[Huang and Mutlu, 2013] Huang, C.-M. and Mutlu, B. (2013). Modeling and evaluating narrative gestures for humanlike robots. In *Proceedings of Robotics: Science and Systems Conference (RSS)*. (document), 7.1, 7.3, 7.2, 7.3

[Huang et al., 2010] Huang, Q., Yu, Z., Zhang, W., Xu, W., and Chen, X. (2010). Design and similarity evaluation on humanoid motion based on human motion capture. *Robotica*, 28(5):737–745. 7.2

[Joosse et al., 2013] Joosse, M., Sardar, A., Lohse, M., and Evers, V. (2013). Behave-ii: The revised set of measures to assess users attitudinal and behavioral responses to a social robot. *International Journal of Social Robotics*, 5(3):379–388. 7.3

[Jung et al., 2013] Jung, J., Kanda, T., and Kim, M. (2013). Guidelines for Contextual Motion Design of a Humanoid Robot. *International Journal of Social Robotics*, 5(2):153–169. 7.3

[Kajita et al., 2003] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *Intelligent Robots and Systems, 2003. (IROS 2003).*

Proceedings. 2003 IEEE/RSJ International Conference on, volume 2, pages 1644–1650 vol.2. 7.4

[Kajita et al., 2001] Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001.*, volume 1, pages 239–246 vol.1. 7.4

[Kalyanakrishnan and Goswami, 2011] Kalyanakrishnan, S. and Goswami, A. (2011). Learning to predict humanoid fall. *International Journal of Humanoid Robotics*, 08(02):245–273. 7.4

[Kamide et al., 2014] Kamide, H., Takubo, T., Ohara, K., Mae, Y., and Arai, T. (2014). Impressions of humanoids: The development of a measure for evaluating a humanoid. *International Journal of Social Robotics*, 6(1):33–44. 7.3

[Kanda et al., 2007] Kanda, T., Kamasima, M., Imai, M., Ono, T., Sakamoto, D., Ishiguro, H., and Anzai, Y. (2007). A humanoid robot that pretends to listen to route guidance from a human. *Autonomous Robots*, 22(1):87–100. 7.3

[Kanehiro et al., 2008] Kanehiro, F., Suleiman, W., Lamiraux, F., Yoshida, E., and Laumond, J.-P. (2008). Integrating dynamics into motion planning for humanoid robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 660–667. 7.4

[Kendall, 1948] Kendall, M. (1948). *Rank correlation methods*. Griffin, London. 6.2.4

[Kendon, 1980] Kendon, A. (1980). Gesticulation and speech: two aspects of the process of utterance. In *The relationship of verbal and non-verbal communication*, pages 207–227. Walter de Gruyter. 7.3

[Kim et al., 2007] Kim, G., Wang, Y., and Seo, H. (2007). Motion control of a dancing character with music. In *IEEE/ACIS International Conference on Computer and Information Science*, pages 930–936. 7.1, 7.3

[Kim et al., 2003a] Kim, T.-h., Park, S., and Y., S. S. (2003a). Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*. 7.3

[Kim et al., 2003b] Kim, T.-h., Park, S. I., and Shin, S. Y. (2003b). Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph.*, 22(3):392–401. 7.1

[Kim et al., 2010] Kim, W. H., Park, J. W., Lee, W. H., and Chung, M.-J. (2010). Robot's

emotional expression generation based on context information and combination of behavior database. In *Robot and Human Interactive Communication*, pages 316–323. 7.2

[Kipp et al., 2007a] Kipp, M., Neff, M., and Albrecht, I. (2007a). An annotation scheme for conversational gestures: how to economically capture timing and form. *Language Resources and Evaluation*, 41(3-4):325–339. 7.3

[Kipp et al., 2007b] Kipp, M., Neff, M., Kipp, K. H., and Albrecht, I. (2007b). Towards natural gesture synthesis: Evaluating gesture units in a data-driven approach to gesture synthesis. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, IVA '07, pages 15–28, Berlin, Heidelberg. Springer-Verlag. 7.3

[Kirby et al., 2006] Kirby, R., Simmons, R., and Forlizzi, J. (2006). Modeling affect in socially interactive robots. In *IEEE International Symposium Robot Human Interactive Communication*, pages 558–563. 7.1

[Kita et al., 1998] Kita, S., van Gijn, I., and van der Hulst, H. (1998). Movement phases in signs and co-speech gestures, and their transcription by human coders. In Wachsmuth, I. and Frhlich, M., editors, *Gesture and Sign Language in Human-Computer Interaction*, volume 1371 of *Lecture Notes in Computer Science*, pages 23–35. Springer Berlin Heidelberg. 7.3

[Knight et al., 2011] Knight, H., Divvala, S., Satkin, S., and Ramakrishna, V. (2011). A savvy robot standup comic: Online learning through audience tracking. In *Fifth International Conference on Tangible, Embedded and Embodied Interaction*. 5.3.1, 7.3

[Kochanek and Bartels, 1984] Kochanek, D. H. U. and Bartels, R. H. (1984). Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Computer Graphics*, 18(3):33–41. 7.1

[Kopp and Wachsmuth, 2000] Kopp, S. and Wachsmuth, I. (2000). A knowledge-based approach for lifelike gesture animation. In Horn, W., editor, *European Conference on Artificial Intelligence*, pages 663–667. IOS Press. 7.1

[Kopp and Wachsmuth, 2004] Kopp, S. and Wachsmuth, I. (2004). Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):39–52. 7.3

[Kress-Gazit et al., 2007] Kress-Gazit, H., Fainekos, G. E., and Pappas, G. J. (2007). From structured english to robot motion. In *Intelligent Robots and Systems, 2007. IROS 2007.*

IEEE/RSJ International Conference on, pages 2717–2722. 7.2

[Kudoh et al., 2008] Kudoh, S., Shiratori, T., Nakaoka, S., Nakazawa, A., Kanehiro, F., and Ikeuchi, K. (2008). Entertainment robot: Learning from observation paradigm for humanoid robot dancing. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Workshop: Art Robots*. 7.3

[Kuffner et al., 2003] Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2003). Motion planning for humanoid robots. In *Proceedings of the 11th International Symposium on Robotics Research (ISRR 2003)*. 7.4

[Kulić et al., 2008] Kulić, D., Takano, W., and Nakamura, Y. (2008). Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *International Journal of Robotics Research*, 27(7):761–784. 7.2

[Laura Mize, 2008] Laura Mize (2008). Dolch word list. 3.3

[Li and Chignell, 2011] Li, J. and Chignell, M. (2011). Communication of emotion in social robots through simple head and arm movements. *International Journal of Social Robotics*, 3(2):125–142. 7.2

[Liemhetcharat and Veloso, 2014a] Liemhetcharat, S. and Veloso, M. (2014a). Allocating training instances to learning agents that improve coordination for team formation. In *AAMAS workshop on Autonomous Robots and Multirobot Systems (ARMS)*, AAMAS '14, pages 1531–1532, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. 7.3

[Liemhetcharat and Veloso, 2014b] Liemhetcharat, S. and Veloso, M. (2014b). Team formation with learning agents that improve coordination. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 1531–1532, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. 7.3

[Luo et al., 2011] Luo, Z., Lin, C.-C., Chen, I.-M., Yeo, S., and Li, T.-Y. (2011). Puppet playing: An interactive character animation system with hand motion control. In Gavrilova, M., Tan, C., Sourin, A., and Sourina, O., editors, *Transactions on Computational Science XII*, volume 6670 of *Lecture Notes in Computer Science*, pages 19–35. Springer Berlin Heidelberg. 7.1

[Masuda and Kato, 2010] Masuda, M. and Kato, S. (2010). Motion rendering system for emotion expression of human form robots based on laban movement analysis. In *Robot and Human Interactive Communication*, pages 324–329. 7.2

[McNeill, 1996] McNeill, D. (1996). *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, Chicago. 7.1

[McNeill, 2005] McNeill, D. (2005). *Gesture and thought*. University of Chicago Press, Chicago. 7.1

[Meratnia and de By, 2002] Meratnia, N. and de By, R. A. (2002). Aggregation and comparison of trajectories. In *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, GIS '02, pages 49–54, New York, NY, USA. ACM. 7.2

[Michalowski et al., 2007] Michalowski, M. P., Sabanovic, S., and Kozima, H. (2007). A dancing robot for rhythmic social interaction. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*, ACM/IEEE International Conference on Human-Robot Interaction, pages 89–96, New York, NY, USA. ACM. 7.1, 7.3

[Microsoft Research, 2005] Microsoft Research (2005). MindNet. `http://research.microsoft.com/en-us/projects/mindnet/`. [Online; accessed 30-June-2014]. 7.2

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, L., and Zweig, G. (2013). word2vec. `https://code.google.com/p/word2vec/`. [Online; accessed 7-December-2015]. 5.2.1, 7.2

[Montgomery, 2005] Montgomery, J. (2005). Beck Gets World's Only 'Dream Robots' Dancing to 'Hell Yes'. `http://www.mtv.com/news/1513517/beck-gets-worlds-only-dream-robots-dancing-to-hell-yes/`. [Online; accessed 22-March-2016]. 7.1

[Motomura et al., 2009] Motomura, S., Kato, S., and Itoh, H. (2009). Generating association-based motion through human-robot interaction. In Motomura, S., Kato, S., and Itoh, H., editors, *Principles of Practice in Multi-Agent Systems*, volume 5925 of *Lecture Notes in Computer Science*, pages 389–402. Springer Berlin Heidelberg. 7.2

[Mumm and Mutlu, 2011] Mumm, J. and Mutlu, B. (2011). Human-robot proxemics: Physical and psychological distancing in human-robot interaction. In *Proceedings of the 6th International Conference on Human-robot Interaction*, ACM/IEEE International Conference on Human-Robot Interaction, pages 331–338, New York, NY, USA. ACM. 7.1

[Nakaoka et al., 2010] Nakaoka, S., Kajita, S., and Yokoi, K. (2010). Intuitive and flexible user

interface for creating whole body motions of biped humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1675–1682. 7.1

[Nakaoka et al., 2005] Nakaoka, S., Nakazawa, A., Kanehiro, F., Kaneko, K., Morisawa, M., and Ikeuchi, K. (2005). Task model of lower body motion for a biped humanoid robot to imitate human dances. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3157–3162. 1

[Nakaoka et al., 2003] Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., and Ikeuchi, K. (2003). Generating whole body motions for a biped humanoid robot from captured human dances. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3905–3910 vol.3. 7.3

[Neff et al., 2008] Neff, M., Kipp, M., Albrecht, I., and Seidel, H.-P. (2008). Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Transactions on Graphics (TOG)*, 27(1):5. 7.2

[Ng et al., 2010] Ng, T.-H., Luo, P., and Sandra, O. (2010). Synchronized gesture and speech production for humanoid robots. In *IEEE Int. Conf. Intelligent Robots and Systems*, pages 4617–4624. 1, 6.2, 7.1, 7.1, 7.3, 7.3

[Nieuwenhuisen and Behnke, 2013] Nieuwenhuisen, M. and Behnke, S. (2013). Human-like interaction skills for the mobile communication robot robotinho. *International Journal of Social Robotics*, 5(4):549–561. 7.1, 7.3

[Noritake et al., 2006] Noritake, K., Kato, S., and Itoh, H. (2006). A interpolation-based approach to motion generation for humanoid robots. In Braz, J., Araújo, H., Vieira, A., and Encarnação, B., editors, *Informatics in Control, Automation and Robotics I*, pages 179–185. Springer Netherlands. 7.4

[Ogata, 2001] Ogata, K. (2001). *Modern Control Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition. 7.4

[Okuno et al., 2009] Okuno, Y., Kanda, T., Imai, M., Ishiguro, H., and Hagita, N. (2009). Providing route directions: Design of robot's utterance, gesture, and timing. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, ACM/IEEE International Conference on Human-Robot Interaction, pages 53–60, New York, NY, USA. ACM. 7.3

[Okuzawa et al., 2009] Okuzawa, Y., Kato, S., Kanoh, M., and Ito, H. (2009). Imitative motion generation for humanoid robots based on the motion knowledge learning and reuse. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4031–4036. 7.2

[Oliveira et al., 2010] Oliveira, J., Naveda, L., Gouyon, F., Leman, M., and Reis, L. (2010). Synthesis of variable dancing styles based on a compact spatiotemporal representation of dance. In *IEEE Int. Conf. Intelligent Robots and Systems*. 7.3

[Paul Ekman and Wallace V Friesen, 1975] Paul Ekman and Wallace V Friesen (1975). Unmasking the face: a guide to recognizing emotions from facial clues. 4.1.1

[Pelachaud, 2005] Pelachaud, C. (2005). Multimodal expressive embodied conversational agents. In *Proceedings of the annual ACM international conference on Multimedia*, pages 683–689. 7.3

[Petri et al., 2013] Petri, T., Gams, A., Babi, J., and lajpah, L. (2013). Reflexive stability control framework for humanoid robots. *Autonomous Robots*, 34(4):347–361. 7.4

[Princeton University, 2010] Princeton University (2010). About WordNet. `http://wordnet.princeton.edu`. [Online; accessed 30-June-2014]. 7.2

[Reich, 2016] Reich, J. (2016). Watch 540 Robots Dance In Honor Of The Chinese New Year. `http://www.techtimes.com/articles/131643/20160208/watch-540-dancing-robots-honor-the-chinese-new-year.htm`. [Online; accessed 22-March-2016]. 1, 7.1

[Renner and Behnke, 2006] Renner, R. and Behnke, S. (2006). Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2967–2973. 7.4

[Ribeiro and Paiva, 2012] Ribeiro, T. and Paiva, A. (2012). The Illusion of Robotic Life. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 383–290. 7.3

[Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). *AI: A Modern Approach*. Prentice Hall. 6.1.2

[Rybski et al., 2008] Rybski, P., Stolarz, J., Yoon, K., and Veloso , M. (2008). Using dialog and human observations to dictate tasks to a learning robot assistant. *Journal of Intelligent Ser-*

vice Robots, Special Issue on Multidisciplinary Collaboration for Socially Assistive Robotics, 1(2):159–167. 7.2

[Saerbeck and van Breemen, 2007] Saerbeck, M. and van Breemen, A. J. (2007). Design guidelines and tools for creating believable motion for personal robots. In *Robot and Human Interactive Communication*, pages 386–391. 7.1

[Salem et al., 2009] Salem, M., Kopp, S., Wachsmuth, I., and Joublin, F. (2009). Towards meaningful robot gesture. In Ritter, H., Sagerer, G., Dillmann, R., and Buss, M., editors, *Human Centered Robot Systems*, volume 6 of *Cognitive Systems Monographs*, pages 173–182. Springer Berlin Heidelberg. 7.1

[Salem et al., 2010] Salem, M., Kopp, S., Wachsmuth, I., and Joublin, F. (2010). Towards an integrated model of speech and gesture production for multi-modal robot behavior. In *Robot and Human Interactive Communication*, pages 614–619. 7.1

[Salem et al., 2012] Salem, M., Kopp, S., Wachsmuth, I., Rohlfing, K., and Joublin, F. (2012). Generation and evaluation of communicative robot gesture. *International Journal of Social Robotics*, 4(2):201–217. 7.1, 7.3, 7.3, 7.4

[Searock et al., 2005] Searock, J., Browning, B., and Veloso, M. (2005). Learning to prevent failure states for a dynamically balancing robot. Technical Report CMU-CS-TR-05-126, School of Computer Science, Carnegie Mellon University. 7.4

[Seo et al., 2013] Seo, J. H., Yang, J. Y., Kim, J., and Kwon, D. S. (2013). Autonomous humanoid robot dance generation system based on real-time music input. In *RO-MAN, 2013 IEEE*, pages 204–209. 7.3

[Sergey, 2009] Sergey, L. (2009). *Body Language Animation Synthesis From Prosody*. Undergraduate honors thesis, Stanford University. 7.1

[Sergey et al., 2010] Sergey, L., Philipp, K., Sebastian, T., and Vladlen, K. (2010). Gesture controllers. *ACM Trans. Graph.*, 29(4):124:1–124:11. 7.2, 7.3

[Shanie, 2006] Shanie (2006). Sony Dream Robot QRIO. `http://www.sonyaibo.net/aboutqrio.htm`. [Online; accessed 27-January-2013]. 7.1

[Shiratori et al., 2006] Shiratori, T., Nakazawa, A., and Ikeuchi, K. (2006). Dancing-to-music character animation. *Computer Graphics Forum*, 25:449–458. 7.1, 7.3

[Shiwa et al., 2009] Shiwa, T., Kanda, T., Imai, M., Ishiguro, H., and Hagita, N. (2009). How quickly should a communication robot respond? delaying strategies and habituation effects. *International Journal of Social Robotics*, 1(2):141–155. 7.3

[Sisbot et al., 2010] Sisbot, E., Marin-Urias, L., Broqure, X., Sidobre, D., and Alami, R. (2010). Synthesizing robot motions adapted to human presence. *International Journal of Social Robotics*, 2(3):329–343. 7.1

[Spearman, 1904] Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101. 6.2.4

[Statt, 2016] Statt, N. (2016). Hilton and IBM built a Watson-powered concierge robot. `http://www.theverge.com/2016/3/9/11180418/hilton-ibm-connie-robot-watson-hotel-concierge`. [Online; accessed 22-March-2016]. 1

[Stephens and Atkeson, 2010] Stephens, B. J. and Atkeson, C. G. (2010). Push recovery by stepping for humanoid robots with force controlled joints. In *Humanoids*, pages 52–59. IEEE. 7.4

[Striegnitz et al., 2005] Striegnitz, K., Lovett, A., and Cassell, J. (2005). Knowledge representation for generating locating gestures in route directions. In *Proceedings of Workshop on Spatial Language and Dialogue (5th Workshop on Language and Space*. 7.3

[Sung et al., 2012] Sung, C., Feldman, D., and Rus, D. (2012). Trajectory clustering for motion prediction. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1547–1552. 7.2

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition. 7.4

[Tay et al., 2016] Tay, J., Chen, I.-M., and Veloso, M. (2016). Fall prediction for new sequences of motions. In Hsieh, A. M., Khatib, O., and Kumar, V., editors, *Experimental Robotics: The 14th International Symposium on Experimental Robotics*, pages 849–864. Springer International Publishing, Cham. (document), 11, 12, 13, 14, 15, 6.1, 6.2, 6.3, 6.4, 7.4

[Tay and Veloso, 2012] Tay, J. and Veloso, M. (2012). Modeling and composing gestures for human-robot interaction. In *Robot and Human Interactive Communication*, pages 107–112. (document), 3.1, 3.2, 2, 3.3, 3.4, 3.5, 5.3, 5.4, 5.6, 5.3, 6.2, 7.1, 7.1

[Thayer, 1989] Thayer, R. E. (1989). *The Biopsychology of Mood and Arousal*. Oxford University Press, New York. (document), 3.3, 4, 4.1, 4.1, 7.2

[The Centre for Speech Technology Research, The University of Edinburgh, 2010] The Centre for Speech Technology Research, The University of Edinburgh (2010). Festival. `http://www.cstr.ed.ac.uk/projects/festival/`. [Online; accessed 30-June-2014]. 3.3, 5.2.2

[Thompson, 1933] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294. 7.3

[Tran et al., 2010] Tran, M. T., Soueres, P., Taix, M., Sreenivasa, M., and Halgand, C. (2010). Humanoid human-like reaching control based on movement primitives. In *Robot and Human Interactive Communication*, pages 546–551. 7.2

[Walters et al., 2011] Walters, M., Oskoei, M., Syrdal, D., and Dautenhahn, K. (2011). A long-term human-robot proxemic study. In *Robot and Human Interactive Communication*, pages 137–142. 7.1

[Webots, 2014] Webots (2014). http://www.cyberbotics.com. Commercial Robot Simulation Software. 1.1, 3.3, 6.1.2, 6.2.4, 6.2.4, 8.4

[Wikipedia, 2015] Wikipedia (2015). First 100 words-advancing your toddler's vocabulary with words and signs. 3.3

[Xia et al., 2012] Xia, G., Tay, J., Dannenberg, R., and Veloso, M. (2012). Autonomous robot dancing driven by beats and emotions of music. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 205–212. (document), 4.2, 4.3, 4, 5, 6, 5.2, 5.1, 5.3, 6.2, 7.1, 7.2

[Xing and Chen, 2002] Xing, S. and Chen, I.-M. (2002). Design expressive behaviors for robotic puppet. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 1, pages 378–383 vol.1. 7.1

[Yamamoto and Watanabe, 2006] Yamamoto, M. and Watanabe, T. (2006). Time lag effects of utterance to communicative actions on cg character-human greeting interaction. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 629–634. 7.3

[Zheng and Meng, 2012] Zheng, M. and Meng, M. Q. H. (2012). Designing gestures with semantic meanings for humanoid robot. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 287–292. 1