Expressive Motion Synthesis for Interactive Humanoid Robots

I. Introduction – Motion for Human-Robot Interactions
    1. Human-Robot Interactions
    2. Some Anthropomorphic Robots
    3. Motion in Human-Robot Interaction
    4. Mentality in Motion
II. Related Work
    1. Sociable robots
    2. Animation in computer graphics
III. Perception System
    1. Vision
        1. Face Detection
        2. Face Recognition
    2. Dialogue
        1. Text
        2. Speech recognition
IV. Cognition System
    1. Communication norms
    2. Emotion-base
    3. Personality-base
    4. Emotional-Personality Model
V. Response System
    1. Speech
    2. Motion
VI. Robot Animation
        1. Denavit-Harternberg notation
        2. Forward Kinematics
        3. Inverse Kinematics
VII. Expressive animation
    1. Disney Animation Principles
    2. Laban Movement Analysis
VIII. Creating Expressive Motions
    1. Applying Animation Theories
    2. Model
        1. Representation as signal
        2. Interpolation
        3. LMA-DAP Model
        4. Motion Signal Processing
IX. Experiment Results
X. Conclusion and Future Works

CHAPTER 1 – INTRODUCTION

Our interest in this thesis is on how to control the motions of an interactive humanoid robot, such that the motion enhances the interaction between the robot and its human user. As such, there are several points we need to expand on:

1. What is an interactive humanoid robot? What is its application?
2. What is the interaction between an interactive humanoid robot and a human user? In other words, what needs to happen when a human tries to interact with a humanoid robot?
3. How does the motion of the robot contribute to the human-robot interaction? What information can be conveyed through the motion of the robot? How to encode information into the motion of the robot, such that the information can be easily 'decoded' by the human user/oberserver?

1.1. Human-Robot Interactions

Robots are fascinating. They come in all different forms and sizes; from virtual robots (pure software) such as web-crawlers used to collect data, to the ASIMO humanoid robot from Honda. From the tiny microbot which is no bigger than a segment of the index finger, to the gargantuan industrial robots used in manufacturing/assembly plants. Sometimes, a robot is not contained in a single object, but is a network of devices scattered in some area, working together to achieve some tasks, for example:

'intelligent' homes.

The earliest robot application is in automation of some manufacturing process, such as in car assembly facilities. Most of their applications are to do structured tasks that cannot be done easily by human workers, such as lifting and assembling heavy components. In addition, because of their speed and precision, robots are also used to do precision tasks that are very repetitive, such as welding. These industrial robots are designed to be purely functional and efficient, and usually have very specific function – a welding robot arm cannot be used to lift a door panel. These applications demand precision, and automation. The control system usually consists of the mathematical model of the robot, including its geometrical description (described by the *Denavit-Hartenberg Notation*), forward or inverse kinematics, and stability controls such as PID. The robots are then controlled either manually through some console or peripherals, or if the task is repetitive, the motions and actions are programmed in the controller of the robot and let run automatically.

Recently, there is a growing market of personal/home robotics products. For example: the Roomba robots from iRobot, RoboSapiens and other anthropomorphic robot toys from WowWee robotics, AIBO and Qrio robots from Sony, and so forth. These robots work closely with humans, and often their main purpose is to *interact* with humans. Their users are often not technically-savvy like the engineers who operate the industrial robots. Thus, instead of entering technical instructions or using clunky

peripherals, these users want to be able to communicate with the robot as a pet or another human being. What this means is that users want – and worse, *expect* – to interact with robots through human communication modalities: spoken words (speech), touch, and gestures. For example, if the user wants the robot to leave the room, he/she does not want to tell the robot: go forward 5 feet, turn left 63 degrees, go forward 2 feet, turn left 12 degrees, and go forward 10 feet through the doorway. Instead, the user wants to be able to just say to the robot "leave the room," and point in the direction of the doorway, and the robot needs to be able to navigate the room and find the doorway itself. Moreover, these robots are expected to understand the human communication norms, such as turn-taking, gestures, emotions, and other conversation etiquettes. These kinds of robots and their human-robot interactions are the interest of this thesis.

1.2. Some Anthropomorphic Robots

The term 'anthropomorphic robot' is synonymous to the term 'humanoid robot,' which is a robot that was made with human characteristics, physically and/or behaviorally. Our concern with interactive humanoid robots are then with robots of this kind.

a *b* *c*

Figure 1.1 Anthropomorphic robots, (a) Kismet, (b) WE-4R II, (c) MIT NEXI

Cynthea Breazeal from MIT [Bre02] developed an emotional model which was implemented in the robot Kismet (Figure 1.1a). This is one of, if not *the* pioneer of state-of-the-art model of mentally anthropomorphic robot. The system is capable of interacting with users by understanding communication cues such as the users' presence, facial expressions, conversational patterns, and gestures. Kismet is then able to respond through its facial expression, some simple spoken responses, and head gestures (Kismet only consists of a neck and a head). It is used for research in human-robot social interaction, for example Kismet is able to sense when there are no people around to interact with, which will make it sad by showing this emotion through its facial expression, and start to 'look around' for people to interact with. She also introduced the concept of socially interactive robots (SIRs) where robots (humanoid or otherwise) are able to socialize with humans and other robots, and have the drives/desire/need to engage in social interactions.

Takanishi robot laboratory at the Waseda University in Japan developed a humanoid robot WE-4R II [MIM+] (Figure 1.1b). The robot also has an emotional model which part of it is based on "A.H. Maslow's Hierarchy of Needs". The WE-4R II has more human-like features than Kismet which is modeled more like a cartoon creature. WE-4R II has a head/face with 26 degrees of freedom (DOF) including its neck, a pair of arms with 18 DOF, and 2 DOF at the waist. It capabilities for facial expression, reaching and grabbing objects, focusing to objects in its view, and reacting to physical stimulation (e.g. slap or shout to the side of its head and it will turn its head in the direction of the stimuli).

Recently, the Personal Robotics Group at MIT developed NEXI (NEXI) (Figure 1.1c), a humanoid robot with highly artistic face and body design. This robot is a wirelessly-controlled mobile robot using a two-wheel platform (i.e. like Segway). As with Kismet and WE-4R II, NEXI is capable of a wide range of facial expressions thanks to its multi-articulated eyes and jaw. Its main design purpose is as a platform for research in human-robot interaction, and specifically, social interaction such as teamwork. In addition, all of those robots are able to dynamically learn new things such as how to respond to a new situation.

1.3. Expressive Motion in Human-Robot Interaction

One important aspect of social interaction is *gesture*; that is, how information can be

conveyed through motion. If the goal is to develop a truly intuitive human-robot social interaction, this is one of the sought after ways to interact with the robot since gesture is one of the main human communication modalities. The robot would be interacting with people who do not want to interact with the robot through awkward peripherals, or complicated syntaxes. The users need to be able to tell the robot to look at an object just by pointing at the object, to say that he/she is leaving by waving his/her hand, to show that he/she is happy by smiling; simply by gestures. And so does the robot, who should be able to express the same things and to say the same type of things as the user did, not by cryptic messages.

In addition to be able to know what action (motion) to execute, there is also *how* to do the motion. In this 'how' sense, the goal is to have the motion to be as natural-looking, as how it would be done by a human. Or at least, *expressive* enough. When humans move, there is information embedded in the motion; whether it is the person's intentions, mood, emotion, energy, habit, or personality. There are some theories on human motion that investigate how information is embedded into motion, or how motions carry information about the person's internal state at the moment. Most of these theories were developed to be used in the field of performing arts such as theater and animation. The theories are used to help actors, dancers, or animators understand how to create the performance for their audience, whether it is to convey a message consciously (e.g. describing the shape of an object through gestures), or expressing the internal state of the human body and mind (e.g. emotions: sad, angry, etc., physical

condition: tired, energetic, etc. or personality, anxiety, etc.).

The two prominent theories of motion that are being used for the basis of this thesis are the Laban Movement Analysis (LMA) and the Disney Animation Principles (DAP). LMA originated from theatrical and dance performances by Rudolf Laban, while DAP originated from hand-drawn cartoon animations by Walt Disney. Obviously, there are many other motion theories out there, but many of them are specific to a certain context (e.g. ballet, opera, etc.), and less applicable to general/everyday human-robot interactions. Meanwhile, LMA and DAP theories are more general and widely used in the performing arts and entertainment industry today, often complementing one another.

LMA has been implemented as a computational model which is used to augment a computer animation (three-dimensional animations, in particular) [CCZ+00]. The model allows users to modify a stored animation (e.g. from motion capture) using the LMA terminologies; that is, by modifying *LMA parameters* such as Effort and Shape, which will be explained in later chapters. DAP method on the other hand, has never been formally modeled as a computational model, but its terminologies kept being mentioned to refer to the characteristics of a motion [iCat, [CCZ+00], [BW95], [Las87], [UAT95]. Our initial guess is that DAP can only be described through human intuition, while LMA is more detailed and technical, seen by the parameters it provides.

We found that the elegant approach to obtain the delicate expressions in motions is through representing the motion as a set of signals, a concept introduced by Bruderlin and Williams [BW95]. They showed that by applying simple signal processing techniques, many motion elements described by LMA and DAP can be achieved. This opens a way to achieve several things in humanoid robot animation:

- A way to achieve more fluid or natural looking motion.
- The ability to quickly create motions showing different characteristics or personality (fast prototyping).
- The ability to compose motions as a combination of multiple motions.
- Visualization of the motion as signals allows a more intuitive representation for editing and analysis.
- Develop a framework for autonomous motion generation and/or modifications.
- Compact motion data, where the motion can be represented as a function (e.g. Fourier series).
- Computationally-elegant influence of input/stimuli to motion (e.g. speech/image inputs received as signals, affecting motion that is also represented as a signal).

The task is now to identify, what kind of transformations or processing techniques will invoke which kind of motion effects as described by LMA and DAP. We aim to relate and describe the effects and elements of the resulting motion using LMA and DAP language/terminologies because they are widely used in the performing arts industry,

yet the vocabulary is simple enough to be understood by most people. For example, according to both LMA and DAP, a motion usually consists of three stages: preparation/anticipation, execution/main action, and follow-through or recovery (completion of the release of energy, and going back to neutral pose/position), or overlap (transition to the next motion/action) [Bis07].

Another component that contributes to the believableness of a motion is its dynamics. The so-called naturalness of human motion comes from the ability to effectively and efficiently use energy in each motion. This is described by the motion dynamics, which involves physical characteristics such as mass, friction, tension, elasticity, energy, velocity, inertia, and so on. In other words, motion dynamics is the physics considering how the movement of the body is being affected or reacts to the physical forces acting on it from the environment, and is closely related to kinesiology. For example, when throwing a ball, the arm does not start abruptly swinging and stop abruptly right after the ball leaves the hand. Instead, during preparation/anticipation, the arm would be retracted back to allow more space to gain momentum, decelerate and pause for a moment. Then as the throwing motion begins, the arms starts to accelerate. When the swing is at the highest kinetic energy point (maximum velocity), the hand releases the ball (assuming the goal is to achieve maximum speed/distance). After the ball is released, the arm relaxes and the swing starts to decelerate, until it comes to a full stop (recovery stage). The deceleration will depend on the mass of the arm, and the velocity of the swing. A high-energy motion sometimes involves some

overshoots in the motion. Of course, this is just one variant of throwing a ball, but all the dynamics exist in every step.

Dynamics simulation can be used to achieve these effects, and adds naturalness to the motion. It has been used recently in modern video games, to create realistic animations for characters or objects in the game [naturalmotion]. Moreover, simulation makes the animation dynamic, as opposed to the traditional approach of canned (pre-programmed) animations, where the character executes the same animation sequences each time. With dynamic simulation, the animation will be different each time yet not random. For example, a character being pushed over and fall down will have different animations of falling each time. The video game player will have the experience that the characters actually responds and interact with the environment elements in the game (e.g. gravity, wind, other objects, and so on). Motion dynamics are the passive side of a motion – it is about the physical *reaction* of the body to the forces acting upon it. We still need to enable the robot to *initiate* a motion with specific emotional effects to convey a message. Both LMA and DAP have concepts that discusses the dynamics in the movement.

1.4. Mentality in Motion

[Bis07], [Las87] indicate that motion and interaction without some kind of internal drive of the agent is dead. The interaction must be as a result of a stimuli either from external sources (people, objects, music), or internal sources (emotion, drive,

initiative, desires). In other words, the interaction must be in *context* for it to make sense. This means that the robot must know how to interpret the external stimuli, and choosing the relevant responses. Psychology theories often call this as the *personality*.

Bishko in [Bis07] points out that motion can be done functionally, or intentionally. Functional motion is just to achieve a certain task, such as reaching for a cup on the table. With intentional motion, there is a goal to achieve (intention), such as reaching for a cup on the table to drink from it because of thirst. She suggested the use of LMA to create this intentionality because the LMA methodology provides the building blocks for this motion.

We demonstrated expressive humanoid robot animations which are dependent on the interaction between the human user and the humanoid robot. The motions of the robot are in reaction to the keywords from the user's inputs, and also the historical records of the interaction?. A set of keywords from the user's inputs are mapped to a set of basic actions or motions. Another set of keywords of adjectives modifies the execution of the selected actions or motions. To ensure that the modifications to the motions are realistic and natural (i.e. they are familiar and makes sense to the human observers), theories of motion from the field of performing arts such as the DAP and LMA are applied to construct the motion modifiers. The robot thus appears to represent/express its internal states in context with the interaction through the resulting motions.

To summarize, the goal of this thesis is to investigate a system that would allow a humanoid robot to show emotions through its motions being executed, by modifying the motion on-the-fly, as opposed to have a complete set of pre-programmed motions. Ultimately, we aspire that the system would be improved to enable the robot to create expressive motions autonomously, such that the expressions of the robot enhance the intuitiveness of human-robot interaction. We found that treating the motion data as signals enables us to quickly, and effectively modify our set of basic motions into motions with some emotional expressions. The motion data (i.e. motion signal) is modified using signal processing methods of interpolation, resampling, and multiresolution filtering. These three methods are used to modify the following motion parameters: acceleration/deceleration in the motion, the duration of the motion, and the range of motion of the motion. The type of emotional expressions that will appear in the motion is determined by the parameter values of the three signal processing methods we used in our system. To determine what kind of changes to the motion parameter values will create which emotional expressions, we applied a subset of theories of motion from animation, in particular from Disney Animation Principles and Laban Movement Analysis. The relationship between the motion parameter values and the kinds of expression perceived in the motion was investigated empirically. And finally, in the spirit of improving the intuitiveness of human-robot interaction, we created a user-interface which the user can use to interact with the robot through text input using English sentences using the ALICE chatbot engine, provides the user with visual/text feedback of the state of the system, and the robot can

identify the user through face recognition. The change of expressions in the motion of the robot is then not triggered by explicit syntaxes/commands, but rather based on the context and keywords derived from the user's text inputs and the text responses from ALICE, and the personality/emotional model of the robot.

In Chapter 2, we discuss the current, and related researches on humanoid robots and animations on humanoid robots. We present our goals, hypothesis, methodology, and overview of the system we use/implemented for this thesis in Chapter 3. In Chapter 4, 5, and 6 we discuss our perception, cognition, and response systems, respectively. As the thesis is focused on robot motion, we added a discussion on kinematics in Chapter 7, which we hope will serve as future reference for a more sophisticated motion synthesis system. We discuss the animation theories and concepts from Disney Animation Principles and Laban Movement Analysis in Chapter 8. The details of our motion processing system, the signal processing methods we used, our design decisions, and application of the animation theories through the signal processing methods are explained in Chapter 9. In Chapter 10 we present the results from our experiments with the system. And lastly, in Chapter 11, we give our conclusions, and future work.

CHAPTER 2 - RELATED WORKS ON MOTION OF INTERACTIVE HUMANOID ROBOTS

2.1. Interactive/Sociable Robots

In her book *Designing Sociable Robots* [Bre02], Cynthia Breazeal indicated that interactive/sociable robots require a combination of systems, from the physical (mechanical system), vision, auditory, to its psychological system such as motivation and behavior, and the system to express its emotion and personality.

Breazeal used an anthropomorphic robot head called Kismet (Figure 1.1a) as a platform to test the idea of a sociable robot. Physically, Kismet is a head/face robot, with 15 DOF in its face and neck. It has 2 DOF for each of its eyebrow, 1 DOF for each of its eyelids, 2 DOF for its eyes, 2 DOF on each of its ears, 1 DOF on its jaw, 4 DOF for its lips, and 3 DOF on its neck (which actually totals to 18 DOF). For its perception system, Kismet has four cameras: two on the eyes are high-resolution cameras used to analyze the face of the user, to extract the user's facial expressions. Two other cameras located in the middle of the face, one between the eyes, and another one below it (nose). These center cameras are used to analyze the environment, and measure the proximity of the person in front of it. In addition, its auditory system is used to recognize speech input, and extract its emotional cues from the intensity and pitch. Its auditory system is also used for Kismet to give speech responses. The psychological system of Kismet was designed to be infant-like, and

was based on child developmental psychology. It could be said that the psychological system models a simple motivation and behavioral system. For example, Kismet will show a sad expression if it is left without anyone interacting with it for some time. It was also used to model a regulated interaction pattern such as in its learning mode, where the user and Kismet are trying to establish a 'comfortable' interaction by following some rules. These rules are based on basic interaction norms such as personal space, frequency of stimuli over time, and so on. Kismet does have a set of facial expressions, and there are certain motions as a result of interaction, such as moving the head back if the person is too close (exhibiting personal space concept). But in general, they are poses, meaning that there are some mappings between input stimuli and output actions. There is not much expression in the motion itself.

The WE-4R II (Waseda Eyes No. 4 Refined II) robot (Figure 1.1b) has a more human-like anatomy than Kismet, and consists of a head/face, a pair of arms and hands, and a torso. It has a total of 59 DOF throughout: 6 DOF for each hand, 9 DOF for each arm, 2 on the waist, 4 on the neck, 3 for the eyes, 3 for each eyebrow, 4 for the lips, 1 for the jaw, and 1 on its 'lungs' . It is also equipped with many sensors, such as 16 tactile (touch) sensors on each hand, and 3D force sensors on its index finger and thumb. In addition, there are more touch sensors on the head, along with temperature sensor, gas sensor, two cameras in the eyes, and microphone. The mental system consists of several components. It has a *need model* that took inspiration from *Maslow's hierarchy of needs*, by having a basic 'need of appetite', then 'need for security', and to

a higher level 'need for exploration'. In addition, it has a mood space and emotion space which models the changing of mood and emotion, respectively, using vectors, as a function in 3D space. The expressions in its motion is controlled by the 'strength' of the emotion, which then affects the speed of the arms.

2.2. Animation of Humanoid Characters

Unfortunately, much of the technical research around automated humanoid robot animation is seriously disconnected with the artistic aspects of animation. As seen in some of the most advanced sociable robots above, often the expressions are just in postures, and less in the motions itself. This often results in robot movements that still feel awkward and eerie to the human observers, even with complex physical and mathematical models for the movement. This is why many approaches to humanoid robot animation tend to rely on canned (pre-programmed) animations which are carefully crafted by skilled animators. When certain states or conditions occur, a corresponding animation is then invoked to simulate the robot's response to the stimuli. Only until recently that researchers take motion theory from art into consideration. Some of the related works mentioned here involve 3D computer animation because there is a direct correlation between robot animation and 3D models animation. That is, both are working with the animation of rigid bodies.

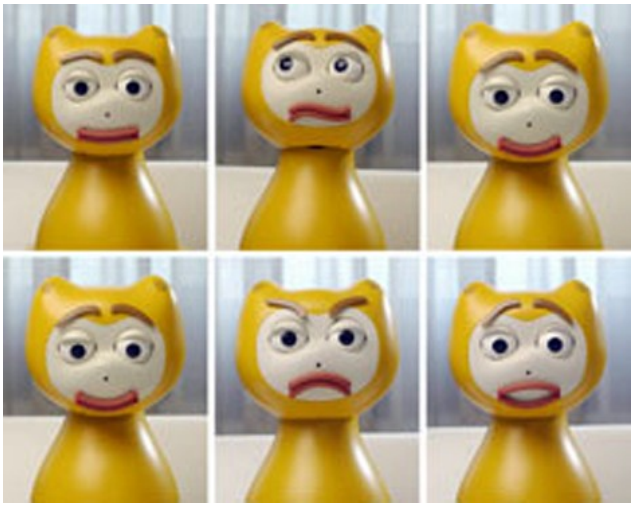


Figure 2.1 iCat Robot.(Image source: [BYM05])

Van Breemen's iCat [BYM05] is a robotic head (and neck) with a face that looks like a cartoon cat. Every possible movement of each part of the face (eye brows, eye lids, uppoer and lower lips, eyeballs) is stored as an individual animation. For example, rolling the eyeball up from looking forward is one animation. Closing of eyelids is another animation. By combining these simple, individual animations, complex animations can be obtained such as expressing emotions, or some gestures. The system employed also a scheduling scheme to avoid conflicting animations (animations that are to be executed at the same time, and using the same facial parts), and filtering techniques to enable smooth transitions between animations.

Many researches toward 'fluid', 'natural-looking', or 'human-like' animations are focused on legged locomotion. For example, [GM85], [BC89], [RH91], [Hye96] focused on bipedal walking by analyzing the details of the gait cycle and rigid-body

dynamics (physics model) of the biped. The goal was to enable walking animations that can be easily controlled in real-time. The authors showed that by providing a comprehensive dynamics and kinematics models, animators can easily generate different kinds of walking by entering few parameters at a fairly high level, such as: figure height, leg length, weights, locomotion speed, and similar parameters. This gives a hint to how natural-looking animations can be automated, given that the model provided is accurate enough. However, there are several drawbacks to this approach. First, the animator must give up some of his/her ability to control the animation because part of it is being controlled by the dynamics simulator. Second, the dynamics simulator takes specific physics inputs such as amount of force being applied to the body; meaning the animators must specify these values, which can be un-intuitive to them. While the animator can do some trial-and-error to get the right values, he/she has no control over the resulting animation if the animation does not look right. Lastly, the results also show that the complexity of the models and the available computation power at the time cannot support the walking animation generation in real-time. Furthermore, the model is not generalizable – it is specifically modeled for walking animations. Other kinds of animation such as running, require modifications to the model – which may involve manual tweaks and adjustments to the parameters of the model, or a completely different model.

Because the joints on the human body have limits for their range of motion, defining constraints somewhat contributes to the naturalness of the motion. For example, the

head cannot turn more than about 180 degrees from left to right and in the opposite direction. Being able to turn the head more than that is extraordinary, yet unnatural for a human. [Wel] , [ZB89] applied constraints to computer animation by considering them in solving the *Inverse Kinematics* (IK) of the motion. The non-linearity of the IK problem often yields multiple solutions, and applying the constraints limits the feasible solution space at the expense of an additional constraint solver.

[Las87] presented the application of the traditional Disney animation principles (DAP) to computer animation. The author argued that even though the medium is very technical, animation is still an art form. In his paper, he described how a rigid body in 3D, when animated properly can give the *illusion of life* – that it will look believably alive. And he showed that by using the DAP as the animation guidelines, this effect can be achieved. Still, the quality of the motion is still heavily dependent on how well the model is built (in terms of articulation) and the skills of the animators.

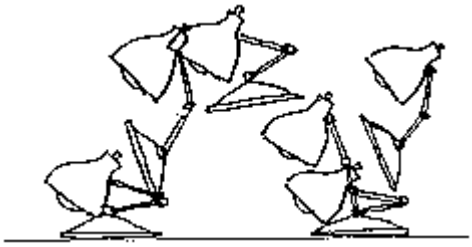


Figure 2.2. [Las87] demonstrated how traditional animation principles can be applied to animate jumping in computer-generated rigid body models better.

Norman Badler and his research group approached the issue from a dance notation

perspective [CCZ+00], [CB] . Taking inspiration from the already mentioned Laban Movement Analysis (LMA), the group presented a model of high-level representation and parameters for computer animation using their EMOTE (Expressive MOTion Engine) system. The LMA theory was originally developed for dance choreography, and provided some explanation of the relationship between the quality of movement and the mental drive behind that quality [Bis07]. The EMOTE system models that relationship, so that by specifying the quantities of the LMA parameters at certain keyframes, animators can quickly modify a ready-made animation to get the corresponding motion qualities. Most of the model was derived empirically with the help of a certified LMA practitioner.

The general perception in both computer character and robot animation is that the more (i.e. redundant) degrees of freedom (DOF) the articulation has, the more fluid or life-like the animation *can* be. Notice the emphasis on the word "can." The only problem is, as the number of DOF increases, the complexity of the computation for the animation also increases. That is why some researches are geared towards high-level representations or language of both the model and the animation. The motion parameters such as speed, joint angles, acceleration are obscured from the users and replaced with behavioral parameters such as "tired", "happy", "old person", and so on. This leads to the model of scripted animation such as PAR (Parameterized Action Representation) [AB02], AER (Aesthetic Exploration and Refinement) [NF05], BEAT (Behavior Expression Animation Toolkit) [CVB01], Improv [PG96], and many more.

These models provide some levels of abstraction to describe the character, and the virtual world/environment the character is in. For example, in PAR, everything in the world – objects and characters – are of the class 'Object.' Regular object like a chair is a subclass of Object, and may have specific chair-properties such as number of legs, height of backrest, and so on. A character is also a subclass of Object, and has some other specific set of properties that are different than a chair, for example: number of DOF, age, gender, and so on. Then there are Action objects, which contain the set of actions available to an Object. For example, a car can move forward, backward, turn right and left. A character may have actions such as jumping, running, waving, and other kinds of action. Each of these actions, analogous to *methods* in the Java programming language (the Action object may also be analogous to *interface class* in Java), may have input parameters that define the behaviours of the action. For example, for running action, there are speed, direction, and so on. Notice that this abstraction does not require the animator to specify individual joint angles. Most of these script models are focused on computer animation, and only a few are applied to robotics such as CRL (Common Robot Language) [Luk]. This kind of framework also provides a way to incorporate motion-specific parameters such as introduced in DAP and EMOTE, and the possibility for automation, i.e. by generating/modifying the scripts in real-time as is supported in CRL.

The abstraction frameworks presented above allow animators to create animations using high-level language, by specifying specific input values to certain animation

properties. While the front-end is somewhat intuitive language-wise (i.e. using natural language), the back-end of the system (i.e. computational model) is not very elegant. They often involve empirically-derived models (developed from observations and trial-and-error). Parts of the model may have been developed specifically to address specific elements of the animation, for example, the motion preparation stage (or Anticipation in DAP). In other words, there are often many 'local' functions made to cover all, if not most aspects of a motion. This makes the framework fragmented with these 'local' functions, adding the number of parameters that the animators should consider, thus increasing the complexity of the system. More importantly, the effects of the parameters are not immediately visible to the animator until the animation is completely generated.

2.3 Motions as Signals

[BW95] introduced a powerful metaphor for animation: looking at s motion as a set of signals. The motion for each degree of freedom is represented as a series of angles/positions in time, which can be plotted as a signal. The authors showed that by applying some simple and common signal processing techniques, they were able to easily generate different motion qualities, and modify certain aspects of the motion elegantly. These qualities were achieved only excruciatingly using other methods. This method can apply both global and local modifications to a motion. For example, by applying multiresolution filtering to a walk animation and increasing the middle frequency gain, the animation becomes exaggerated; the steps taken are “bigger” - the

knees are raised higher, and the foot lands farther. Conversely, when the middle or low frequency gains are reduced, the steps become “smaller”. These properties show that there is a correlation between using common signal processing approach and the artistic side of animation (Exaggeration is one of the principles in the DAP framework).

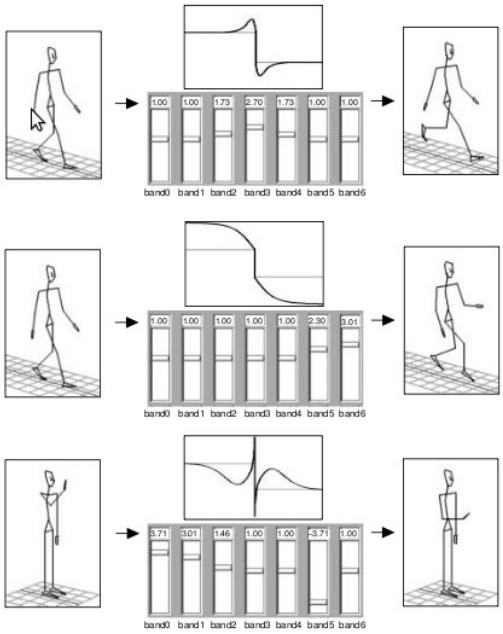


Figure 2.3 Modifying gains of different frequency bands from multiresolution filtering can exaggerate or dampen the range of motion. (Image source: [BW95])

By applying a waveshaping function, the animation may have certain characteristic that is visible in the function (because it is represented as a waveform), such as having undulations in the beginning, middle, or towards the end of the animation sequence. This approach also allows multitarget motion interpolation (combining multiple animations into one), and motion displacement mapping (graphically editing the

motion signal while preserving its continuity). Currently, there are very few studies done to find the mapping between signal processing techniques and their effects on motion qualities. So, there is more work to be done before these ideas can be applied to automate motion modifications.

A similar signal/filtering approach was implemented in iCat [BYM05], in the way it handles animation transitions. Note that these methods are similar to EMOTE in terms that both systems are used to modify a ready-made animation, and not for generation. Although, further studies may reveal the possiblity to generate new robot animations using these approaches.

[UAT95] identified that when a periodic motion such as walking or running is represented as a Fourier series, the Fourier coefficients can be used to interpolate (morph) between two similar periodic motions (e.g. normal walking and tired walking), and extract motion characteristics (e.g. extracting 'tiredness' from a tired walking animation by taking its difference from a normal walking animation). But the morphing process is slow, and not feasible for real-time applications (e.g. modifying animation as it runs). The authors noted also that although the Fourier series were normalized, and the Fourier coefficients are continuous in the range [0,1], the transition from one effect to the other is not invertible. They gave an example in which the transition from walking to running resembles how if it was done by a human, but the transition from running to walking is 'unnatural'. The authors also

discovered the exaggeration property using their approach; this is achieved by increasing the coefficients for lower-frequency components.

Research on 'natural-looking' motions are often associated with 'expressive motions', that is, for a motion to be expressive, it must look natural. Yet, a natural-looking motion is related to some expression of the performer. As such, many researches on expressive motions include some kind of emotional model, personality model, or some ways to represent the internal state of the performer (mood, emotion, energy level, etc.) [DD06]. The authors used fuzzy logic to model the relationship between the emotion/personality/physical states and some intermediate parameters for motion quality. The intermediate parameters are then fed into a multilayer perceptron artificial neural network that translates them into physical motion properties such as joint angles, angular velocity, and acceleration. The authors show that they were able to simulate some emotional motions. For example, an angry state results in the motion having many jerky motions with large intensity (motion range) variations with some noise in the motion signal, while happy or calm state results in the motion having smooth trajectories, and relatively small intensity variations. [GO03] implemented the Big Five personality model using fuzzy logic. The model maps some emotional state to behavioral expressions, but the expressions were not delivered as motions, but only as descriptions of the response behaviors in text output.

CHAPTER 3 – PROPOSED THESIS

3.1 Goals

Based on our knowledge from the previous and related works on interaction of interactive humanoid robots, in this thesis we are focusing on the issue of creating emotional expressions in the movements of a humanoid robots. Specifically, our main goal is to gain understanding on how emotions are expressed in motions, and to synthesize movements for a humanoid robot that expresses different emotions. Ultimately, the movements of the humanoid robot are not random, nor repetitive, and just 'for show,' but also have meaningful contributions and become an integral part in the human-robot interactions, as the gestures are in human-human interactions.

Our secondary goal of this thesis is to be able to achieve our main goal in the most effective, and intuitive way such that if successful, this should be as accessible to anyone with or without technical knowledge.

3.2 Hypothesis

Thus, as we recognize the challenges and possible solutions to achieve our Goals, we developed our hypothesis. For our hypothesis, we consider the following:

1. As the robot is humanoid, that is in human form, users and observers already set their expectations that the robot can, and will behave like a human.

Specifically, users and observers would want to be able to interact with the humanoid robot as if with a human.

2. For the human-robot interaction to be successful and meaningful, these expectations must be met; the human-robot interaction should be through human communication modalities such as vision, and natural human language (in this case, English).
3. To facilitate the expectations, at least partially, we assumed a minimal set of interaction modalities that would be acceptable to meet the expectations. We implement the following modalities: face detection and recognition through vision, natural language processing through text.
4. The emotional expressions in the motion response of the robot must come from the context of the interaction, rather than direct commands.
5. Because the robot is incapable of having facial expressions, the expressions of emotion, personality, and physical states can only be done through the motions of the robot.

Thus, our hypothesis:

Human-like emotion, personality, and physical states can be expressed by a humanoid robot through its motions, encoded as a set of signal transformations.

3.3 Approach

To test our hypothesis, we do the following:

1. We create a scheme of human-robot interaction:
    1. We implemented human-like interaction modalities, such as: face detection and recognition through vision, conversation, and motion.
        1. Vision input is obtained through a normal webcam and is used to detect and recognize the users' faces.
        2. Conversation is in text format for both input and output, with additional speech output.
        3. Motion is the gestures or actions performed by the robot as a response to the context of the interaction.
    2. The interaction flow follows some basic human communication norms such as greetings, turn-taking, and interaction history (i.e. how the interaction progresses over time). This is realized using the ALICE conversational engine.
    3. The interaction will determine the emotional state of the robot/system either explicitly (through certain keywords) or progressively (as the interaction goes).
    4. The context of the interaction is extracted and presented to the user from the information obtained through the perceptions of the robot/system.
    5. During testing/experiment, users will be presented with feedback of the current states and perceptions of the robot/system. However, these information may be hidden during public performances.
    6. Users will also have access to some reserved commands to directly control

the state and actions of the robot/system at all times.

2. We represent the motion as signals because of the following properties:
    1. When the motion is modified using signal processing techniques, the integrity of the motion is preserved.
    2. The visualization of motion as a set of signals is relatively intuitive; the visuals immediately give a sense of the speed and range of the motion.
    3. There are some indications from literature that expressions of emotions can be extracted from and encoded into motions using signal processing techniques.
3. We investigate the effects of a set of different signal transformations on a basic motion, and how it translates to an expression of an emotion, if any.
    1. To understand how emotions are expressed in motion, we looked into the Disney Animation Principles and Laban Movement Analysis.
4. Once item 3 above is understood, a set of rules is developed to relate the context of the interaction to the states of the robot/system to the applications of the signal transformations on a selected motion.

3.4 The System

The block diagram of the overall system is shown in figure 3.1.

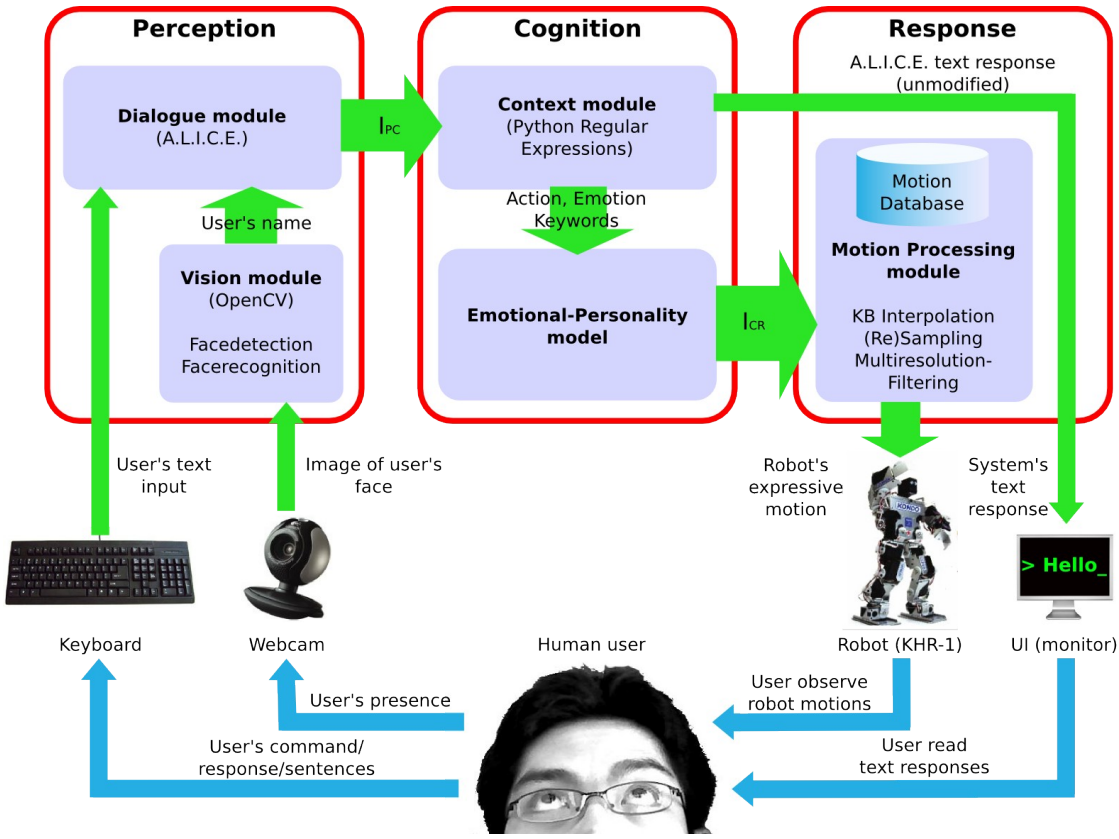


Figure 3.1. Overview of our system.

There are three main components to interactions: perception, cognition, and response. As such, we divide our system into those three aforementioned components. Perception refers to the input stimuli during the interaction. Cognition is the proces of understanding the input stimuli, and deciding the response to it. Finally, response – which can be in the form of a language/sentences, and motion (e.g. gestures, actions), is the reaction or answer to the input stimuli. The workflow of the system goes as follows:

1. The Perception block receives inputs from a webcam, and the text inputs entered by the user. The inputs are in the form of an image of the user's face

and text inputs in the form of commands, words, or English sentences.

2. The Perception block extracts information from inputs above, and outputs I_{PC} (Information from Perception to Cognition). I_{PC} consists of information such as: number of faces detected, name of person identified from the face, and output sentence from the ALICE engine.
3. The Context module in the Cognition block receives the information in I_{PC} as its input.
4. The Context module in the Cognition block then extracts action, and emotion keywords, if any, from the user's input sentences and ALICE's responses using Regular Expression.
5. The keywords extracted by the Context module are passed to the Emotional-Personality module.
6. In the Emotional-Personality model, the emotion keywords are used to produce the parameter values for the Motion Processing Methods, and the 'willingness' parameter, according to the 'weights' of the keywords. The action keywords are mapped to call specific basic motions in the Motion Database.
7. The output of the Emotional-Personality model is I_{CR} (Information from Cognition to Response). I_{CR} consists of information such as: the name of the selected motion or motions, the set of Motion Processing parameter values, and the 'willingness' value. The response sentence from ALICE is passed through the Response block without any changes.
8. The Response block receives the information in I_{CR} as its input. Specifically,

I_{CR} is the input for the Motion Processing module in the Response block. For the sake of completeness, the response sentence from ALICE is also received by the Response block, but no additional processing is done to the sentence, and will immediately be displayed on screen.

9. The information in I_{CR} dictates the Motion Processing module to do the following:
    - Load one or more basic motions from the Motion Database,
    - Set the parameter values for the Motion Processing methods (KB Interpolation, (Re)Sampling, and Multiresolution filtering),
    - Apply the Motion Processing methods to the basic motion and produce the expressive motion.
10. The outputs of the Response block are in texts, and motion of the robot.
11. The user observes the motion response of the robot, and the text response (from ALICE) to understand the robot's response.
12. The User decides how to proceed with the interaction and respond back to the system (Step 1).

3.4.1 Software

Our software consisted of several components:

- The main program is written in Python, C, and C++ programming languages, running on Ubuntu Linux.

- We used Python to program the user interface (UI) and motion processing algorithms.
- We also used the Python Numerical library (NumPy/SciPy) which provides many MATLAB-like functions such as matrix operations, and signal processing functions which helped us greatly in doing the motion processing.
- In addition to NumPy/SciPy, we also used PyQt and PyQwt graphical libraries to create the UI and plot the motion signals, respectively.
- We used the C/C++ OpenCV (Open Computer Vision) library for our face detection and face recognition features,
- We used Fusion, a robotics demonstration application that provides the Python API (Application Programming Interface) to interface with several types of robot platforms such as Kephera I, Kephera II, and KHR-1. We used the Fusion API to send commands to KHR-1, which also handles the serial communication.
- We integrated the Python implementation of the ALICE chatbot program to handle the dialogue part of the human-robot interface.
- We also used Speech Dispatcher, a text-to-speech or speech synthesizer program to provide speech for the responses of the robot.

3.4.2 Hardware

We use the follwing hardwares:

- Our system is running on a normal PC laptop with 1.6 GHz dual processors, and 2 GB of RAM.
- A Logitech QuickCam Communicate MP webcam for face detection and recognition.
- A Kondo KHR-1 humanoid robot, which is described in more detail in the following section.

3.4.3 Kondo KHR-1 Humanoid Robot

Our robot is a Kondo KHR-1 humanoid robot. The robot has a total of 17 degrees of freedom (DOF, i.e. servos): two DOFs on each left and right shoulder, one DOF on each elbow, one DOF for the 'head/neck', two DOFs on each hip, one DOF on each knee, and two DOFs one each ankle. The servos for the DOFs are controlled by two RCB-1 servo controller boards that are coupled together.

The first RCB-1 board controls the upper half of the robot, which includes both arms and head/neck (7 DOFs total). The second RCB-1 controls the lower half of the robot, which consists of both legs (10 DOFs total). Each RCB-1 can drive up to 12 servos. Each RCB-1 uses a PIC16F873A microcontroller, and has 128kbit on-board memory (EEPROM). The RCB-1 board communicates with a PC through a RS232 serial port, using special commands. The commands can be seen in the RCB-1 Command

Reference documentation which can be found online, or on the RoboSavvy.com website. The memory is used to store 'motions' and 'scenarios' which can be called by their index numbers.

Figure 3.2 Kondo KHR-1 humanoid robot. (Image source: http://1.bp.blogspot.com/_9wWgM4K5ikE/RZ2V1-42qII/AAAAAAAAACQ/6Jqz0KTfgO8/s320/Kondo-KHR-1HV-Robot.jpg)

In Kondo's terminologies, a 'motion' is a series of rows of servo positions. A row in a motion has the information of the position of all the servos (e.g. in this case, the position of up to 24 servos) at a point in time. In animation terminologies, a row can be considered as a 'keyframe' in animation, and thus a 'motion' consists of a series of keyframes. Each 'motion' can have up to 100 keyframes. A 'scenario' is a series of 'motions,' which can consists of up to 200 'motions' in one 'scenario.'

Our KHR-1 has only one one-axis gyroscope sensor KRG-2, which is manufactured by Kondo for their KHR-series robot. The KRG-2 is connected between the RCB-1

and two servos for lateral motion of the left and right ankles. When a tilt to the side is detected, the KRG-2 will compensate the given joint angle of the ankle if the tilt is beyond a certain threshold. The threshold value of the KRG-2 can be adjusted using a potentiometer located on the KRG-2. At this point, we are not using any information from the KRG-2 sensor in our system.

3.4.3.1 Motions for KHR-1

Motions are the key ingredient to our system. Ideally, we would like that through its perception-cognition system, the robot would be able to generate its own motion response and paths. Currently, for our experiments, we rely on a set pre-programmed motions such as 'right arm wave', 'push ups', 'flapping arms like wings', and other simple motions. We refer to this pre-programmed motions as 'basic motions.' Our system then modifies this basic – emotionless – motions, to motions with emotional expressive qualities while maintaining the general form of the motion. For example: the basic motion is 'wave right arm', after processing the motion becomes 'happy right arm wave,' or 'tired right arm wave.' These basic motions were created using the Heart-to-Heart (H2H) software that comes with the KHR-1 kit.

3.4.3.2 Executing Motions on KHR-1

In general, the executions of a motion or scenario from the RCB-1 on-board memory are complete and consistent. That is, each line of servo positions in the motions and/or scenarios will be executed in order. However, we realize that since we want to be able

to perform many changes to our motions, and these changes can be different each time, this means we will have many different motions for the robot to execute. Despite being the best way to run a motion, we consider that storing the motion in RCB-1 memory each time would quickly exhaust the writability life of the on-board memory. Therefore, we decided that each line of the motion would have to be fed from the PC to the RCB-1 sequentially.

Each line of motion is sent to the RCB-1 as a string of byte. The string to manually send servo positions to the RCB-1 has the format shown in Figure 3.3 (the full list of commands for RCB-1 can be found in Appendix 3):

Bytes:	1	2	3	4	5	...	15	16
	CMD	ID	SPD	CH1	CH2	...	CH12	CHKSUM

Figure 3.3 RCB-1 string format to send servo positions manually.

Where:

- CMD = Command byte (FDh for manually sending servo positions)
- ID = ID number for the RCB-1 boards (ID=0 for upper body, ID=1 for lower body/legs)
- SPD = Servo speed [0, 7], where: 0 = fastest, 7 = slowest.
- CH1 – CH12 = The twelve channels a RCB-1 can drive
- CHKSUM = (CMD + ID + SPD + CH1 + ... + CH12) & 7Fh

In our initial testings of our method of manipulating the motion data for speed, we found that the SPD parameter in the byte string of the RCB-1 command presented some challenges. If the SPD is set too low, each line of motion is executed fast, and with stops in between, resulting in sudden movements. If the SPD is set too high, the motion becomes too slow, and the variations of speed from our method becomes less apparent. We found that the best SPD value for our approach is around 5 or 6. Thus, we always set the SPD value constant at 5 for the execution of each line of motion.

3.5 Experiment Method and Result Analysis

After the set of rules is developed and implemented, we let users interact with the robot, and let them guess the emotional states of the robot at different instances of the interaction. In the following, we present the overview of our experiments, while the experiments and results are discussed in-depth in Chapter 10. We divided the experiments into three Scenarios. The first scenario is the simplest of the three, and basically tests the functionality of our Motion Processing system. In the first scenario we bypassed our Emotional-Personality model in order to be able to observe the motions. In the second scenario we invited an observer to provide feedback on their perception on the motion response of the robot. And the third scenario, we interact with our system including the Emotional-Personality model, and evaluate how the emotions are perceived in the motion responses of the robot.

During the experiment, the state of the robot is hidden from the user (although it can

be made visible on command). The user can guess the emotion of the robot based on the motion of the robot and the text responses. Additionally, the users can evaluate his/her own inputs to the robot through the use of insult words, praise words, repetition of inputs, meaningful or gibberish inputs, and so on. For example: if the user attempts to make the robot angry, will the robot exhibit behaviors through its motions that the user can identify as “angry,” or will the user identify the behavior as something else?

The success rate of the experiment is measured by the number of times the user correctly guessed the emotional state of the robot over the number of trials. If the user can correctly guess the state of the robot most of the time (let's say > 75%), then the hypothesis is considered true, otherwise it fails.

CHAPTER 4 - PERCEPTION

In this chapter, first we will discuss the components for perception which we implement in our system: the vision system for face detection and recognition, and the ALICE conversational engine for processing text input sentences (Figure 4.1). Additionally, we will also discuss other perception modalities which we do not implement in our system. We believe these modalities would improve the intuitiveness for the human-robot social interactions, such as: gesture recognition, stereo vision, and speech recognition. ~~Obviously, there are many other physical communication modalities that are being used in human-to-human social interactions such as: touch, smell,~~

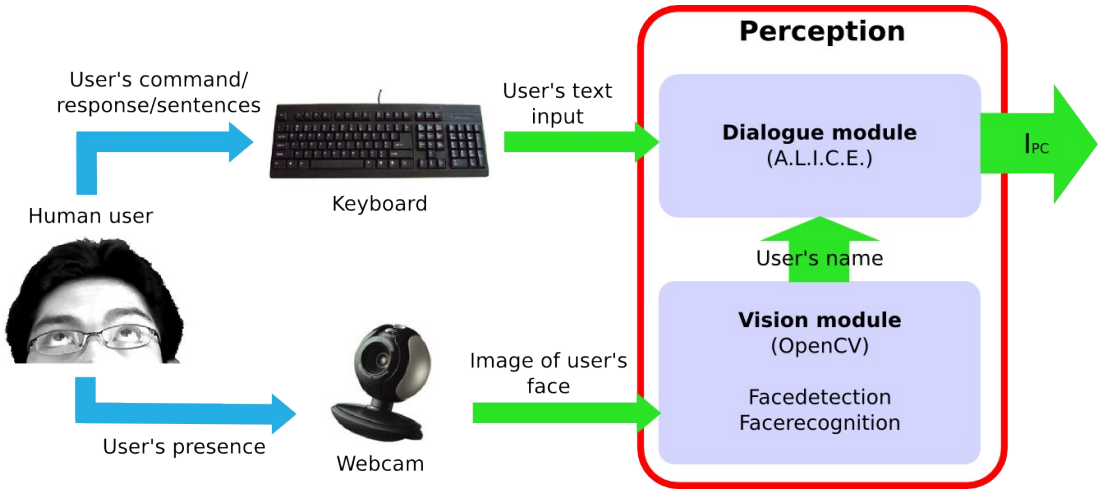


Figure 4.1 The implemented perception system (excerpt from Figure 3.1). I_{PC} is the output of the Perception system, consisting of the original user's input sentence, and the response sentence generated by the ALICE engine.

4.1. Vision

Vision is the natural form of perception for humans, but is very computationally

expensive when executed by computers. This is because the amount of information contained in an image (*"A picture is worth a thousand words!"*), and so it requires many processes to extract all that information. The information that can be obtained through vision includes: color, shape, edges, pose, facial expression, face detection, face recognition, position, orientation, distance, and others. In addition, there are information in vision that can be obtained *in time* such as: movement direction, and gestures.

For our interest in socially interactive robot s (SIRs), we found vision is needed in making the human-robot interaction as intuitive as human-human interaction. In terms of interaction with human users, robotic vision is commonly used in the following applications:

- To identify the user (i.e. face detection and recognition) so the robot can communicate with the user on a personal level.
- To identify an object of interest in the interaction.
- Gauge the distance between the robot and the object of interest (i.e. through global vision[1], or some depth perception algorithm such as Stereo Vision) such that the robot can do the calculation to reach the object of interest (e.g. the ball in RoboSoccer).
- Identify user's facial and/or body gestures (i.e. gesture recognition) as direct commands to the robot, or as an indicator of the state of the user (e.g. emotions).

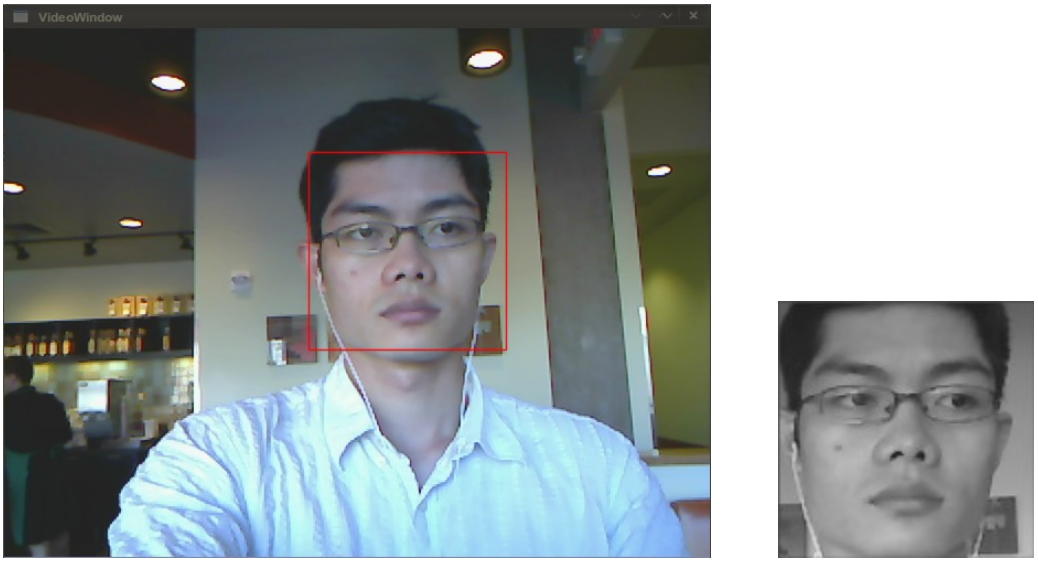
1 One or more cameras may be placed in the robot's environment (global view) and not necessarily on the robot itself (local view).

- On mobile robots, to create an internal model of the world (using edge detection, wall detection) for navigation.

For our application in socially interactive robots, we are particularly interested in computer vision for face detection and recognition, object detection and recognition, and gesture recognition. Additionally, we might also benefit from not just general face detection, but also facial gestures or expressions recognition to enhance the interaction experience.

In our system, we only implemented vision for face detection and recognition. Because of the inherent problem with basic, pattern-matching face recognition approach, we coupled the face recognition feature with a face detection system. The main problem of the basic face recognition approach is because it tries to find a match from the whole image captured from the camera. This 'whole image' often contains background objects, noise, and other artifacts that does not contribute to the facial features. In other words, the background 'noise' are not making it any easier to recognize the face, in fact, the recognition becomes even more difficult as the background are often not consistent in real-life applications. This is why most experiments of face recognition are often done in a controlled environment with minimal background objects (plain wall, or screens), or kept at the same location, and with the same lighting conditions. If the images are an exact match (as if matching the same photograph with itself), the face recognition works perfectly.

Instead of keeping the environment under a certain condition (which would not be feasible for real-time, real-world applications), our approach is to only focus on the face in the image, which is captured through face detection. When the face detection system identifies a face in an image, the system marks the face by drawing a rectangle around the face. We only store this part of the image which contains the face, so we only have images of the face to match, and significantly reduce the background noise (Figure 4.2).



a *b*

Figure 4.2 (a) Camera view, (b) Captured face for recognition. Note: the captured face image (b) is slightly different from (a) because of delays when the first and second image are saved.

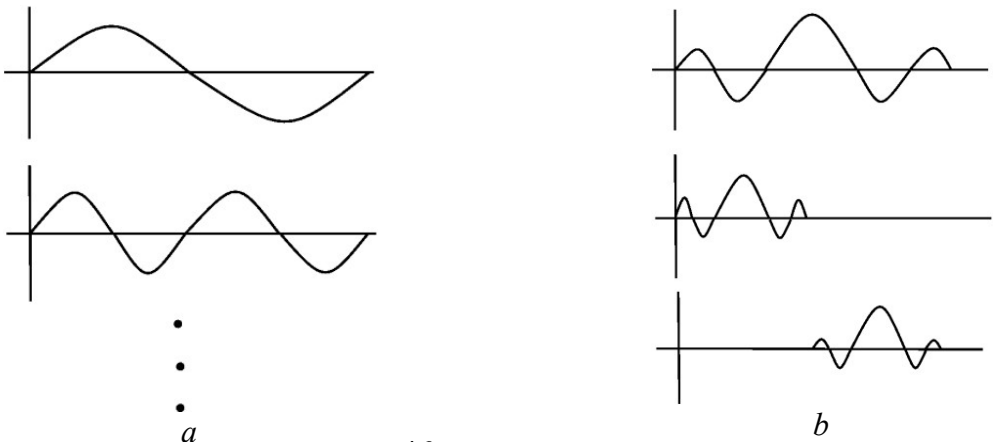
4.1.1 Face Detection

Face detection refers to the ability to distinguish a human face from the other objects

in the image. Our face detection system is based on a face detection algorithm developed by Paul Viola and Michael Jones (aptly known as the Viola-Jones method) [VJ04]. This method uses a set of primitive features (i.e. simple patterns) that referred to as *Haar-like features*. These patterns were selected from their observations to adequately represent facial features. This method has been proven to be generalizable to detect other objects as well by using different sets of features [SK04].

4.1.1.1 Features

These features are called Haar-like because they are inspired by the Haar wavelets, and as such, they have similar properties, and are used in a way similar to Haar wavelets. The Haar wavelets are a representation of a complex function by means of simple functions, called *basis functions.* In other words, arbitrary complex functions can be constructed through the combination of these basis functions. This is closely related to the Fourier series. However there is a significant difference between the two. In Fourier series, the basis functions last for the entire interval, while in Haar wavelets, the functions are localized in time [haarwaveletbasis].



a *b*

Figure 4.3 (a) Fourier basis functions, (b) Wavelet[2] basis functions (Image source: http://cnx.org/content/m10764/latest/)

Thus, the wavelet approach has an interesting property: the basis function can be manipulated by *shifting* and *scaling*. As seen in Figure 4.3b, the top wavelet is the original basis function. The middle wavelet is the scaled (only) basis function, while the bottom one is the scaled and shifted basis function.

This concept is then used for object detection in an image with the following analogy: The object of interest (for example, a human face) is modeled as an arbitrary function. To find this 'arbitrary function' we need a set of basis functions. This set of basis functions is the set of features that identifies an image as a face. Therefore, if we have a set of features (basis functions), that represents features in a face (an arbitrary function), we can detect the presence of a face (identify the arbitrary function) if we found some amount of the features (basis functions) localized in an image. Localized, meaning that statistically the features are found relatively close to each other in an image.

The feature set is overcomplete, that is, some of the features are redundant: representing the same regions (parts) of the face, although the features themselves are different. Therefore, in detecting some regions (or parts) of the face, there may be

2 For any kind of wavelet basis functions, not just Haar.

some features that overlap. Figure 4.4 illustrates this property. In Figure 4.4 there are two different features. The first feature is a rectangle divided into top (shaded) and bottom (unshaded) sub-regions. The second feature is a rectangle divided into three sub-regions (left and right shaded, middle unshaded). Notice that while the shape, size, and number of sub-regions (shaded or unshaded area) of the features may be different, the feature is always divided evenly into sub-regions of the same size.

For example: the eyes are separated by the nose bridge, thus the feature is represented by a rectangle divided into three equally-sized columns (i.e. 3 cells arranged horizontally), with the center column unshaded, and the end columns shaded, based on the observation that the eye areas would normally be darker than the nose bridge area. However, the same part of the face can also be represented as two rows with the darker areas on top (across the eyes region), and lighter area for the area below the eyes. The value of this feature is then calculated as the sum of the shaded areas minus the unshaded area in the middle.

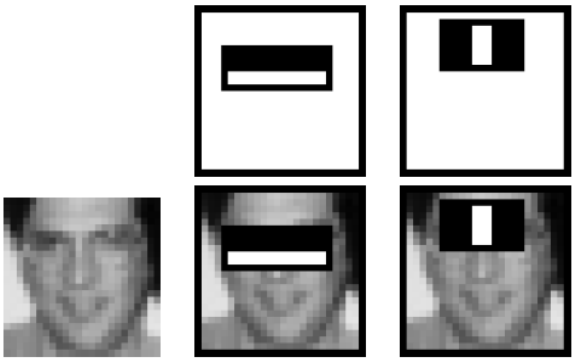


Figure 4.4 Features example (Image source: [VJ04])

In Viola-Jones method, much inspired by the work of Papageorgiou, Oren and Poggio [POP98], the most basic features evaluates the horizontal (Figure 4.5a), vertical (Figure 4.5b), and diagonal (Figure 4.5d) features. To evaluate more complex features of the face, more complex rectangle features are used (Figure 4.5c).

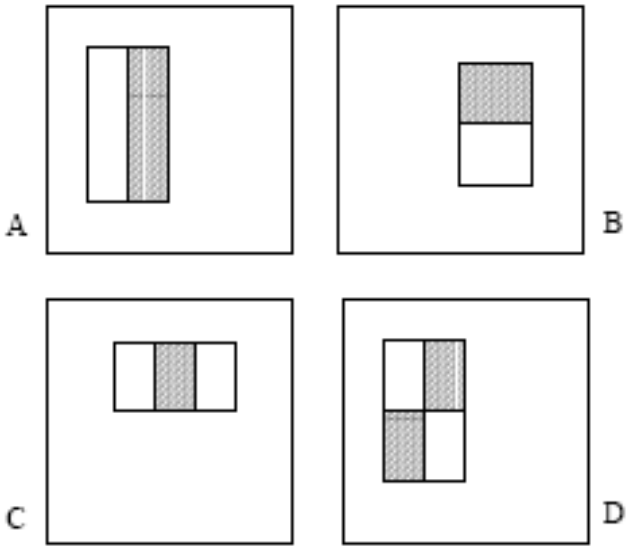


Figure 4.5 Example of rectangle features (Image source: [VJ04])

These features are calculated as follows. For a two-rectangle feature, the value of the feature is calculated as the difference between the sum of all the pixels in each rectangle. For a three-rectangle feature, the value is calculated as the sum of the pixels in the two outer rectangles, minus the sum of the values of the pixels in the middle rectangle. For the four-rectangle (diagonal) feature, the value of the feature is the difference between the sum of the pixels of the diagonal pairs of rectangles. Let's see some example on their calculation by assuming that we are evaluating a 4 x 4 pixels area of a grayscale image, where the value of each pixel ranges from 0 to 255. Lower value denotes darker area (towards black), and higher value denotes lighter area (towards white), and anything in between are varying levels of gray. The shaded/unshaded area in the following example represents the kind of filter being applied to the area.

120	123	40	42
115	123	47	53
102	142	58	52
80	52	25	51

Figure 4.6 Example of two-rectangle feature value

For example, for the two-rectangle feature (Figure 4.6), the value of the feature is the difference between the sum of the pixel values in the unshaded area (written here from top left to right, then down):

$$120 + 123 + 115 + 123 + 102 + 142 + 80 + 52 = 857$$

And the sum of the pixel values in the shaded area:

$$40 + 42 + 47 + 53 + 58 + 52 + 25 + 51 = 368$$

Which makes the difference to be:

$$857 – 368 = 489$$

Unfortunately, it was not clear if the difference is an absolute value, or it is possible for the value to be negative.

In case of the three-rectangle feature (Figure 4.7), assume that the rectangles are the leftmost column, two center columns, and the rightmost column.

120	123	40	42
115	123	47	53
102	142	58	52
80	52	25	51

Figure 4.7 Example of three-rectangle feature value

The value of the feature is calculated as the difference between the sum of the pixel values in the outer rectangles:

$$120 + 42 + 115 + 53 + 102 + 52 + 80 + 51 = 615$$

And the values in the two center columns:

$$123 + 40 + 123 + 47 + 142 + 58 + 52 + 25 = 610$$

Therefore, the value of this three-rectangle feature is:

$$615 - 610 = 5$$

Let's now evaluate a four-rectangle (diagonal) feature example.

120	123	40	42

115	123	47	53
102	142	58	52
80	52	25	51

Figure 4.8 Example of four-rectangle feature value

The value of this feature on this particular area is the difference between the diagonal pairs. The sum of the shaded diagonal pair is:

$$120 + 123 + 115 + 123 + 58 + 52 + 25 + 51 = 667$$

The sum of the unshaded diagonal pair is:

$$40 + 42 + 47 + 53 + 102 + 142 + 80 + 52 = 558$$

So, the difference between the shaded and unshaded diagonal pair is:

$$667 – 558 = 109$$

This final value from the difference between the regions denotes if the feature is present in that area of the image. The lower the value, the feature becomes less apparent, or even non-existent. Conversely, the higher the value, the feature is more apparent. Just to convince the unconvinced reader, let's try to examine another four-rectangle feature. This time, the pixel values are different and the diagonal feature is

apparent (Figure 4.9).

120	123	40	42
115	123	98	53
42	100	150	135
80	52	133	102

Figure 4.9 Example of four-rectangle feature value with a diagonal feature in the image

The sum of the shaded area is:

$$120 + 123 + 115 + 123 + 150 + 135 + 133 + 102 = 1001$$

The sum of the unshaded area is:

$$40 + 42 + 98 + 53 + 42 + 100 + 80 + 52 = 507$$

The difference between them:

$$1001 - 507 = 494$$

The maximum value of the feature when it exists in the scanned area, with our assumptions that the pixel value ranges between 0 – 255, is when the value of the pixels in the shaded area is maximum (255), while the value of the pixels in the unshaded area is minimum (0). In our four-rectangle example, this maximum value

then would be:

$$(8 * 255) – (8 * 0) = 2040$$

Note that when the feature is completely non-existent, or the area of the image is of uniform shade, the feature value gets closer to 0 (zero) with the feature value being zero when all the pixels in the area have the same value.

Notice how in our previous examples, the value when the feature is present is about less than 500, which is roughly 25% of the maximum value. While in the other example where the feature is not present, the value ranges between 5 – 109 (5% and below the maximum value). We think that for the detection of the feature, there must be some threshold value to distinguish when a feature is or is not present.

As we mentioned earlier, we are not sure whether or not the Viola-Jones algorithm takes count of negative feature values. A negative feature value would mean that the feature exist but with different polarity. To illustrate, imagine seeing a black line over a white background, then imagine seeing the same line on the same background but with the opposite polarity like in a film negative.

In their paper, the Viola and Jones only specified that the value of the feature to be only as "*the difference between the sum of the areas"* but the paper did not specify

how it is evaluated (sum of shaded area minus sum of unshaded area, or vice versa). One simple solution is to take the absolute of the feature value to detect the feature in any polarity condition.

The set of features used to detect the face was selected from a large set of features using a learning algorithm called *Adaptive Boost (AdaBoost)*. Roughly, the algorithm goes as follows:

- From a set of features, for each feature *i*:
  - Initialize a normalized weight w_i (i.e. for *m* features, the inital weight of each feature is 1/m)
  - Run a set of negative (no face in the image) and positive (with face in the image) image samples/training set, and try to detect the presence of a face using only this feature.
  - Calculate the error of the feature, as the sum of false detection times weight of the feature (or conversely, reduce the error with every correct detection).
- Pick the set of features with errors less than a certain threshold.

Once a face is detected, naturally the next step is to ask: *who* does that face belong to? Face detection, when coupled with face recognition capabilities enhances the degree of human-robot interaction, and so, next we shall discuss face recognition.

4.1.2. Face Recognition

To perform face recognition on the image of the user's face, we used a face recognition system based on the Principal Component Analysis (PCA) method according to the implementation in [facerecog].

4.1.2.1 Principal Component Analysis

One of the practical approaches which is fast enough for real-time application is through the Principal Component Analysis (PCA) method. The main idea of PCA is to analyze a set of data to obtain the main characteristics, features, or pattern in the data – that is, the 'principal component' of the data. Thus, the rest of the computations for pattern-matching or recognition are in terms of the principal components, and not directly between the data themselves. An analogous illustration of this method is on *linear regression* where we try to model a function from a set of data. The resulting model is represented as a line through a set of data where the distance of each data point to the line would be minimal. Thus, when the model receives an input that was not in the original data set, the model can still yield a reasonably good output approximation. For example: when we have a set of random data as plotted in figure 4.10a, we can see that the data are scatterred around some kind of a line. Let's assume that we obtained the line after we did a regression analysis on the data. If we plot this line over the data plot (Figure 4.10b), the line becomes the characteristic of this set of data. In PCA terms, this line is the principal component of the data, and is going to be

used to match this set of data with another set of data.

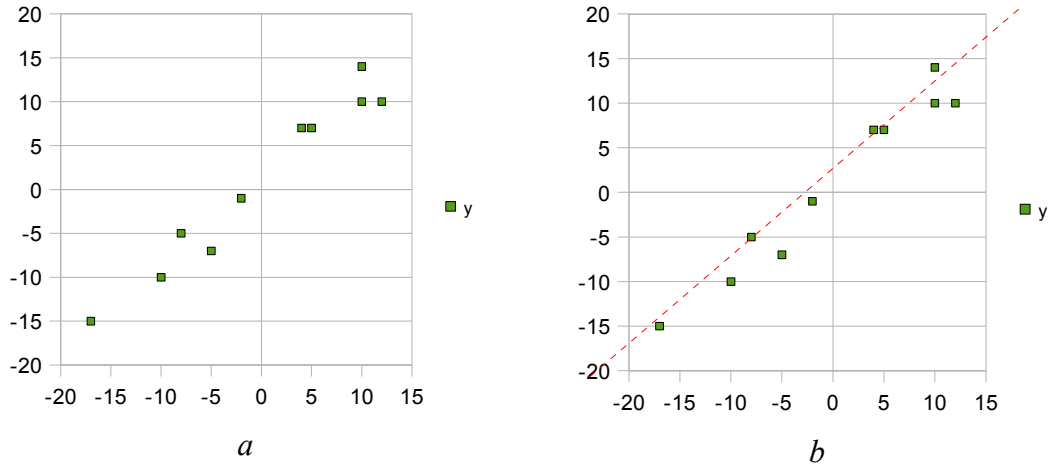


Figure 4.10 Regression analysis on a data set

Before we describe the PCA algorithm for face recognition, let's look into the three main components in PCA: *covariance matrix, eigenvalue, and eigenvector*.

4.1.2.2 Covariance

The covariance matrix is the matrix representation of the covariance between two sets of data. The covariance between two sets of data describes the relationship between the data in the first set with the data in the second set. For example, let's say we have two data sets. The first set is set D, where each element is the distance you travelled each day with your car. The second set is set G, where each element is the amount of gas the car spent each day. Let's represent the covariance of D and G as *covariance(D,G)*. The value of the covariance itself is not as important as the *signs* of

the covariance itself to describe the relationship between the sets. When *covariance(D,G)* is negative, it means that the relationship between the distance travelled by your car and the amount of gas it spends is in adverse: when one goes up, the other goes down, and vice versa. If the covariance is positive, it means the relationship is linear, that if one goes up, the other also goes up, such as: the more distance your car travels, the more gas the car consumes. However, when the covariance is *zero* (0), it means that the two data sets are completely independent (i.e. orthogonal) of each other. As a side note, when there is only one set of data, i.e. one-dimensional data, it is called *variance* instead of covariance. Also, when there is *n* data sets, the covariance matrix is of size *n-by-n*. The covariance between two data sets is calculated as follows:

$$covariance(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

Where:

- covariance(X, Y) = covariance between data set X and data set Y.
- n = length/number of elements in the data set (normally, each of the compared data sets have the same number of elements).
- X_i, Y_i = the *i*th element of data set X and data set Y, respectively.
- $\bar{X}, \bar{Y}$ = the mean values of data set X and Y, respectively.

In actuality, covariance or variance is equivalent to *standard deviation squared*.

Based on the above, for *n* data sets, the covariance matrix is a square matrix with size *n* x *n*.

$$for\ 2\ data\ sets\ x\ ,\ y: \begin{bmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{bmatrix}$$

$$for\ 3\ data\ sets\ x\ ,\ y\ ,\ z: \begin{bmatrix} C_{xx} & C_{xy} & C_{xz} \\ C_{yx} & C_{yy} & C_{yz} \\ C_{zx} & C_{zy} & C_{zz} \end{bmatrix}$$

Where:

- C_{xx}, C_{yy}, C_{zz} = *Variance* of data set x, y, z, respectively.
- $C_{xy} = C_{yx}$ = Covariance between data set x and y. The order of the subscripts does not matter. This is because of the comutativity of the multiplication of the mean-subtracted values in the covariance formula. The same also goes for $C_{yz} = C_{zy}$, and $C_{xz} = C_{zx}$. Because of this, the covariance matrix is always a symmetrical matrix along its diagonal.

4.1.2.3 Eigenvector and Eigenvalues

The eigenvector and eigenvalues are emerging elements in a special case of matrix multiplication. Essentially, for a given square matrix of size *m* x *m*, there exists a set of *m* column matrices of size *m-by*-1 (*m* rows and 1 column), i.e. a *vector*, such that when the square matrix is multiplied with any of these vectors, the result equals the vector itself multiplied by some *constant*. In other words, the square matrix can be

considered as a transformation matrix for the vector, and in this case the transformation is *scaling*. The column matrix is called the *eigenvector*, and the constant is called the *eigenvalue.* For example:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix}$$

Where the vector $\begin{bmatrix} x & y \end{bmatrix}^T$ (here, the column vector is written as a row matrix transposed), is the eigenvector, and λ is the eigenvalue.

4.1.2.4 The Face Recognition Algorithm

The PCA algorithm for face recognition is the following:

- From a set of training face images, find the covariance matrix.
- Calculate the eigenvectors (in this context referred as 'eigenface') of each image in the set with respect to the covariance matrix.
- From this, the eigenvalues of each image are also found.
- Keep the set of M number of faces that has the highest eigenvalues.
  - There must be some criteria to select M.
- The set of M eigenvalues then defines the 'face space.'
- Create classes of the individuals in the training set by projecting them to this face space.
  - Each class is described by a vector of weights $\Omega = [w_1, w_2, ... w_M]$ where each weight corresponds to the projection of the face image to each eigenvalue that

defines the face space.

To recognize a newly captured face image, the following algorithm is executed:

- Project the face image to the face space
- Calculate the distance between the projected face image to the face space and the known faces (classes) using common distance measures such as Euclidean or Mahalonobis:
  - If the image is far from the face space, and all of the classes (i.e. beyond certain threshold), then the image is not a face (far from face space), and is unknown.
  - If the image is far from the face space, and near one of the classes, then the image is not a face, but may be recognized as the nearest class (person).
  - If the image is near the face space, but far from all of the classes, then the image is a face, but the person is unknown (new person or unrecognizable).
  - If the image is near the face space, and near one of the classes, then the image is a face, and is recognizable (and will return the person's name).

Each of the M eigenvalues that define the face space corresponds to the largest variances among the image set. These variances can be considered as the principal components (or features) that can be used to identify a face. Hence, the name "Principal Components Analysis" (PCA). However, these features are often not intuitive for humans, but they make sense for computation. Consciously, human

identifies a face by features such as shapes of eyes, nose, jaw, etc., hair color, eye color, and so on. But for real-time computation purposes, identifying and classifying these features are very expensive. PCA allows for a compact way to store the class information, and also the retrieval (i.e. the recognition of class memberships).

4.1.3. Gesture Recognition

Currently, we do not implement any gesture recognition in our system, and more information on the recognition system can be found in the work by Scott and Quay (Scott, Quay 2007). Nevertheless, we feel that this feature is important to discuss in the context of this thesis, and also since this thesis is a part of our bigger project of Robot Theater.

Gesture recognition refers to the ability to recognize a certain pose, or action (i.e. a series of poses) performed by a human. This allows the robot to interpret a particular pose or action as a command (input) to execute an action or change its internal states, mimic the pose or action, and learn to reproduce them by storing them in memory. Gesture recognition – if executed well – can elevate the intuitiveness of the interactions because of the fact that humans use gestures in their interactions to communicate. The HandVu hand gesture recognition program created by Mathias Kölsch, Matthew Turk, Tobias Höllerer, Stephen DiVerdi [KM04], [KT04] is one candidate gesture recognition program that we intended to implement in our system.

Currently, HandVu can recognize six specific hand gestures (for the right hand only): closed palm (or back of palm), “two” or “victory” sign formed using the index and middle finger shaped like a “v”, an “L” shape formed by the index finger and the thumb both from the back of the hand or from the palm side, spreaded fingers, index finger pointing up with the other fingers curled and with the thumb side facing the camera. The hand gesture detection is best when the skin color of the hand does not suffer from over exposure, uniform background with color different from the skin, and the gesture must not be rotated more than 15 degrees (all gestures are preferably performed upright). Also, HandVu was implemented using OpenCV, and using similar Viola-Jones detection method to detect hand features as our face detection system [KM04]. Thus, HandVu can be used to enhance the interaction experience and control by recognizing the user's communicative hand gestures such as waving hand, greeting, offering a hand shake, and other hand gestures, as the user interact with our system/robot.

To enable gesture recognition, first, the robot needs to identify where is the *part of the person* that is performing the gesture (e.g. hand, face, whole upper body). Secondly, once the robot finds the person, it must understand the shape that is created by the person, or how the shape is created. For example: an “O” shape can be created by using both arms, or simply by the thumb and index finger or middle finger, using one hand or both hands. Third, if the gesture is an action, the robot must memorize the

sequence of poses, and somehow know that the system is supposed to identify the entire action, and not just some particular poses. Moreover, the robot must be able to distinguish between actual gestures and random movements.

A common algorithm to recognize gestures is by recognizing the different body parts, such as shoulder, upper arm, elbow, lower arm, hands, torso, head, and so on, and then calculate their relative position to a reference (usually the torso). To do this, a technique called image segmentation can be used to segment the image into the different parts, such as implemented by the Edge Detection and Image SegmentatiON (EDISON) system, developed by Chris M. Christoudias and Bogdan Georgescu from Rutgers University [GC02].

4.1.4. Stereo Vision

As with gesture recognition, we currently do not implement stereo vision in our system. We discuss stereo vision for future reference when we improve our system. Stereo vision would enable the system to calculate distances of objects in front of the camera, such as a person, a person's hand, a can, and other objects. The distance information can then be used to create actions, such as reaching the person's hand to offer a handshake. Robot stereo vision is directly inspired by the human Stereo Vision system which enables humans to gauge distances through *depth perception*. The idea is by taking two or more images of the same object at the same time from different

angles, by finding the corresponding points of the object in the images, and using the difference of the position of the points because of the different angles, the distance of the point – and thus, the object – can be measured.

The premise of Stereo Vision is ultimately to allow completely autonomous navigation and/or interaction for robots without restricting them to controlled environments where global vision is provided. For example, envision a mobile robot that navigates through hallways to pick up empty soda cans using an arm. First of all, the robot needs to be able to avoid obstacles in the hallway (e.g. people, trashcans, boxes, cabinets, tables, etc.), so it will need to be able to determine the location of those obstacles in its path, and calculate some deviations in its path to effectively avoid them. Naturally, simpler sensors such as 'feelers'[3] can be used to sense the obstacles once the feelers touch them; inanimate objects probably would not mind being "feeled" by the robot, but humans will most likely feel otherwise. Second, the robot will need to be able to reach for the soda cans when it finds them. The robot needs to know how far it needs to approach the can such that it will be within the range of its arm, and how far it needs to extend its arm to reach and grab the can. Without its own object recognition (to identify the cans and obstacles), and distance measuring capabilities, the robot must be aided with environment alterations, such as cameras placed at strategic locations such that it covers all possible areas the robot will go and need to "see", to be able to calculate the distances (or find the positions) of the cans

3 Tactile sensor, analogous to (and inspired by) antennas on insects.

and obstacles. Needless to say, this robot will not work properly without the aids perfectly set up, or if suddenly the environment changes.

If the robot is equipped with Stereo Vision capabilities, the robot can (theoretically) do all of the above through its vision system. For example, the robot can find the relative distance of those objects to itself. While using the same captured images, the robot can also identify cans and the other obstacles through detecting the shape and size, or the pattern printed on the objects through pattern matching – all through vision (without the awkwardness of the feelers).

However, Stereo Vision is easier in theory than in practice; most of the difficulty lies on extracting the actual intrinsic parameters of the cameras through calibration[4], and keeping the capture devices (i.e. camera) on a relatively static configuration physically (i.e. position). The intrinsic parameters consist of the *epipolar geometry,* and the camera focal points. The epipolar geometry describes the physical configuration of the system such as distance between cameras, the image plane of the cameras, the relationship between (i.e. mapping of) the point in three-dimensional space (from the actual object) and the image planes. All this information is described in a geometrical description, and will be explained in the following subsection. A slight change in the camera position could affect the values of the intrinsic parameters significantly, and most likely to cause the detection to fail. If there are changes to the configuration, the

4 The intrinsic parameters of a camera includes: the focal point of the camera, and the epipolar geometries.

system must be re-calibrated.

4.1.4.1 Epipolar Geometry

The epipolar geometry is used to obtain depth information using *triangulation*. As the term indicates, the distance (or depth) of a point is measured using a triangle, which is created by the relationship of the point itself to the two cameras. This triangle is called the *epipolar plane*.

4.2 Dialogue

Dialogue refers to inputs that are in the form of a language sentences. It can be perceived either by speech or sound, or by text input. More importantly, dialogue is where the explicit exchange of information happens (versus implicit information where the information must be extracted from meaning in body/facial gestures). In speech or text dialogue, most of the information that are useful for inputs can be easily recognized from *keywords* or *keyphrases* in the sentence. However, in speech, there may also exist implicit information such as emotions, which can be recognized by the tone, intonation, or volume of the speech.

We implement dialogue in order to create the social human-robot interaction. In other

words, so that the human user and the robot can communicate with each other in a social interaction. In our system, dialogue is implemented by using two components: a *conversational agent* (i.e. chatbot), and a *speech synthesis* system. The conversational agent uses the ALICE engine [ALICE] to engage user in a text-based conversation. The agent is initiated after a person is recognized by the face recognition system. The person's name and information are then used as the context of the conversation. Once the system established/determined that the condition is ready for a conversation, all the text responses from the conversational agent are displayed on the screen, while at the same time also synthesized as speech signals.

4.2.1 Text

Text input in the context of social human-robot interaction is to allow the user to communicate with the robot using natural language, and not technical syntaxes. There has been many research efforts and competition on how to create a system that could pass the *Turing test*. Basically, the Turing test is a test for intelligent machines. The successful machine is the one which is able to fool the human user that he/she thinks that he/she is interacting with an actualperson. The system is usually in the form of a *chatbot*, or automated programs that take text input and give text responses to the user. The way user interact with this system is similar to internet-relay chat (IRC) or instant messaging (IM) programs.

One appealing factor of including text input is that it is *at most* as complex as a speech recognition (SR) system. Words entered as text inputs are rarely mistaken as different words as in SR system, and there are no insertion or deletion errors except when caused by the user's entry errors. However, in SR the system will always only try to find words that it understands (was trained to), which are actual words. In text inputs, it is possible for users to enter gibberish information (words that are not actual words in a known language). But handling such cases are relatively easy in text input, because the system can just throw an error saying it does not understand the word, or try to find a closest match.

4.2.1.1 ALICE

We used the ALICE engine to handle the conversation between the user and the robot. Specifically, we utilize ALICE's sentence processing capability and its large database of input-output patterns to process the user's text input and give a text response based on the user's input. ALICE is developed by the A.L.I.C.E. Artificial Intelligence Foundation, and won the 2004 Loebner Prize, which is the first formal artificial intelligence competition based on the Turing Test. For our system, we used the Python language implementation of ALICE[5] for better integration with our whole system. ALICE is not able to compose its own sentences for response, but rather relies on a

5 ALICE is also implemented in Java, C/C++, .NET, Ruby, PHP, Lisp, and Perl. For more information on the implementations, see the ALICE website: http://www.alicebot.org/downloads/programs.html

database of input-output patterns. The database is made of a set of files of an Extensible Markup Language (XML) implementation called Artificial Intelligence Markup Languange (AIML). Each file in this 'database' is a topic, or context of conversation. At run time, the ALICE engine can match the user's input sentences with any of the AIML file that has been loaded when the program was started. ALICE can perform symbolic reduction which recursively reduce the user's input sentence to a simple pattern.

Table 4.1 Symbolic reduction in ALICE using AIML

Input pattern	AIML pattern	After reduction

I am very tired today	`<category>`
	`<pattern>I am very * today</pattern>`
	`<template><srai>I am <star /></srai></template>`
	`</category>`
		I am tired

Do you know who Dr. Seuss is?	`<category>`
	`<pattern>Do you know who * is</pattern>`
	`<template><srai>who is <star /></template>`
	`</category>`
		Who is Dr. Seuss

Have you watched the new Star Trek movie?	`<category>`
	`<pattern>Have you watched the new * movie</pattern>`
	`<template><srai>Did you watch <star /></srai></template>`
	`</category>`
		Did you watch Star Trek

Notice that the 'reduction' does not necessarily directly reduce from the original input sentence, but can be a summary or the main idea of the input sentence. Naturally,

ALICE's ability to comprehend these kinds of inputs depends on whether or not such ideas have been programmed in the AIML database.

The <category> tag indicates a set of input-output pattern. The <pattern> tag indicates the input pattern, while the <template> tag indicates the response pattern. An example of basic input-output pattern in AIML would look something like this:

```
<category>
<pattern>How are you</pattern>
<template>I am fine thanks</template>
</category>
```

The pattern can capture the input “How are you?” and ALICE will respond with “I am fine thanks.” In the symbolic reduction example, there are two other tags that are used: <star /> and <srai>. The <star /> tag references whatever pattern is being symbolized by the wildcard character asterisk (*) in the input pattern. The <srai> tag indicates that symbolic reduction must be performed for that input pattern. Namely, the pattern surrounded by the <srai> tag is a reference to another category in the AIML database which has the <srai> pattern as its input pattern. The <star /> tag, then passes the symbolized pattern to this 'reduced' category. If this reduced category also has <srai> as its output pattern, then the symbolic reduction is repeated. This feature is useful to handle input patterns that essentially mean the same thing. For example: “I'm leaving”, “I have to go now”, “I'm going home”, “See you later”, all mean “goodbye”, thus <srai> can be used to redirect these inputs to the “goodbye”

category. For more information on ALICE and AIML, please refer to the AIML documentation titled "Artificial Intelligence Markup Language (AIML) Version 1.0.1" available at: http://www.alicebot.org/documentation/.

Thus, symbolic reduction somewhat gives the illusion that ALICE understands the message of the user. For our purposes, this is useful to set up the context of the conversation when the user's input is reduced to main idea of the message. ALICE provides a way to create and set values for a variable during the conversation. For example, we can create the variable 'mood' or 'emotion', and set their values according to certain responses. We can then extract these values to be used as the inputs to our main program for the mood or emotion of the robot, that may be used to modify the motion of the robot.

However, currently the only way to extract the values of these variables is through creating another <category> pattern which output pattern is the value of the variable. The problem with this solution is that by giving ALICE a new input pattern, some information ALICE already collected from the duration of the conversation is often becomes obsolete. In other words, unwittingly we already changed the context of the conversation because of this variable-call pattern. To overcome this issue, we do not use ALICE's internal variable capability exclusively so ALICE can maintain its context. Instead, we use some simple *regular expressions* patterns to capture certain

keywords from the user's inputs and ALICE's responses. We try to capture phrases which give some indications of performing an action (or verb, such as: run, fly, jump, dance, and so on), and emotions or mood (e.g.: happy, sad, tired, afraid, angry, and so on) from the conversation. We use these keywords to establish context as triggers to modify the motion response.

4.2.1.2 Context

Understanding the context of a given dialogue input remains one of the most challenging issue in human-robot interaction, and human-computer interaction in general. Context in a dialogue depends on several things.

First, a context may develop from a keyword or key-phrase (a set of words) in the dialogue that either prompts interest, offense, or triggers some emotion. In the subsequent discussions after the keyword is recognized, the whole conversation often said to “revolve” around the keyword (i.e. topic). This means that the next sentences, or words in the conversation that follows are focused, or related (or should be somehow related) toward that keyword. For example, if person A (named “Jack”) mention his “pet cat”, a person B (named “Jill”) may ask more about Jack's pet cat, talk about Jill's own cat, or whatever pet Jill has. Otherwise, if the topic of “pet cat” is not interesting to the Jill, Jill may start talking about something else (invoking other keywords/phrases).

Second, context may arise from some meta-concepts, or abstract ideas. For example, concepts about the progression of the dialogue itself, or the trend in the interaction. If Jack brings up a keyword, and Jill does not react to the keyword, or brings up a different keyword not related to Jack's keyword, Jack may recognize that Jill is not interested in the topic or does not understand it well. Jill can be said to "keep changing the subject." Jack then may invoke some keywords or keyphrases about the interaction itself. If this trend persists in the interaction, this may evoke Jack's emotion, and may lead Jack to leave the conversation. Another example of concepts-to-context is related to gestures, or body/facial expressions which can convey emotions.

Third, context may come from something or someone in view/sight. For example, if Jack is showing an object that is new (unknown) to Jill and it invokes Jill's interest or emotion. The context may also be affected by who the person Jack is talking to. For example, Jack would talk about romance with Jill, while he will talk about sports with Joe, because Jack knows Jill and Joe's topic preferences.

In human-computer interaction practices, context can be seen in menus or options of a computer application or website. Basically, the application or website will only give a certain set of options to the users depending on what state the application or website is

in. For example, in an application such as Microsoft Word, clicking the right mouse button (right-click) will bring up a list of options on screen, right by the position of the cursor. The list of options will be different when the cursor is positioned over a text, than when the cursor is positioned over the toolbar. This is an example of context-based menu.

In human-robot interaction, context is derived from the perception system of the robot. First, the robot may get context by identifying who is the person that the robot is interacting with (through face recognition). From the person's identity, the robot can retrieve all historical information from previous interaction with that person. The information may include, but is not limited to: which keywords were recorded, how the interaction went (did it end well, or badly? e.g. When the interaction ended, is the emotion of the robot happy or angry or sad?), was there another person involved in the previous interaction, how long was the interaction, and so forth. This information can be stored in a database, and retrieved based on the identity of the person. This will allow, for example, to set the robot in an unhappy mood immediately after it identifies a person whose last interaction went sour.

In general, context is initiated through the invocation of keywords, or keyphrases. The challenge is to develop a model that can carry on a conversation while keeping it within the context, and knows how to keep it interesting (keep the user engaged).

Conversational software or *chatbot,* such as ALICE uses a database of input-output patterns, categorized by topics [Wal]. The input-output patterns and their mappings are manually programmed (by human), and are easily modified to mimic some personality, and expanded to cover other topics. However, ALICE is a very static program; that is, it cannot generate or combine words on its own to create a new sentence and simply relies on whatever patterns it already has in its database. To overcome always having the same answers for an input pattern (for example: "Bye" always replied with "Goodbye."), ALICE supports symbolic reduction to map several different input patterns (or sentences) into a category. . For example, "Goodbye", "Bye", "See you later", "Talk to you later", "Until next time" may be categorized as a "GOODBYE" pattern. Under the "GOODBYE" pattern, there may be a list of possible responses such as: "Goodbye", "Good bye", "Bye", "It was nice talking to you.", "Take care.", "It was a pleasure to meet you." and so forth, that could be selected randomly. Randomizing the responses allows a different answer most of the time, even if the input pattern is always the same.

4.2.1.2 Regular Expressions

To extract context from texts, the most common approach is to use Regular Expressions (Regex). Regex is a representation of patterns in a set of strings and is explained using formal language theory. The representation consists of a set of special characters or symbols that would help the Regex *parser* (i.e. translator) to dissect the

Regex patterns to match it with the input string. On top of pattern-matching, Regex can be used extract a set of characters, or words by *grouping* to be used for further processing when necessary. For example: a Regex patterns "`behavior|behaviour`" or "`behavio?r`"would catch the words "behavior" or "behaviour" in the input string. The symbol "|" in the first pattern represents boolean OR, and the symbol "?" in the second pattern represents *one wildcard character*, meaning it would accept matches with any one or no character following the character before it (in this case, the letter "o") and before the letter "r." And so, using the latter pattern would also match non-english words like "behavioxr", "behavioar", and so on. Other Regex symbols (i.e. operators) include but not limited to:

- *: wildcard for arbitrary character, and any number of characters
- (): used to group certain parts of the pattern, and can also be used to return those certain parts of pattern.
- ^: (a Python programming language Regex implementation) used to complement a character or a set of characters in the pattern. In other words, the Regex parser will only find any other characters except the complemented ones.
- And so on.

We use Regex in Python to extract context-related keywords from both the input and output strings, such as "Topic", "It" (current subject), "Username", "Usermood",

"Personality", "Robotmood", and "Action." The keywords, along with the other information gathered from vision, are then processed to extract the context of the interaction by Cognition, which is explained in Chapter 5.

4.2.2. Speech (Recognition)

In our system we did not implement a speech recognition component. However, we include speech recognition as part of our discussion on robot perceptions since we believe speech recognition is highly desirable for an intuitive human-robot interface. Speech recognition will be utilized in our collective work on Robot Theater. Further reference on speech recognition can be found in the work by Dimitry Labunsky [Lab08] and the PSU Nautilus team project [Nau08].

Speech Recognition (SR) refers to the capability to translate spoken words into machine language (e.g. binary codes, string of characters, ASCII codes, etc.). It has been applied in applications such as customer service, word processor dictation softwares, telephone voice dial, security systems, military aircraft controls, and so on [Wiki:SR]. For social behavioral robotics application, it is important that the machine can also understand the *context* of the words or sentences, so it can respond accordingly. However, understanding the context of the sentence is still difficult to achieve computationally. At the end of this chapter, we will discuss why it is so, and what can be done to achieve it to some extent.

4.2.2.1. Perception

Obviously, an SR system takes sound as its input. The sensing (input) device is usually a transducer that converts sound into electrical signals such as a microphone. The electrical signals are then preprocessed by being converted into a signal representation (i.e. model) before being used next for the recognition processing. The preprocessing usually models the signals as *functions* (e.g. sine wave). Depending on the SR system and the recognition approach it uses, certain parameters of the functions are extracted, and stored to be used in the actual recognition process. The parameters can be the phase, amplitude, *(or other parameters),* obtained through spectral analysis[6]. Thus, this representation is also called the *parametric representation* (White 1976).

(White 1976) indicated that *all* the information about speech signals can be encoded in *formants*. A formant is a peak in the frequency spectrum of a sound created by acoustic resonance [Wiki:For], which in humans is the voice from the vocal tract. Examples of the formants for the vowels i, u, and a are seen in figure 4.11. Formant can be used to distinguish the voices between vowels, consonants, and thus between the two (vowels vs. consonants). Therefore, formants can be used to encode phonemes. A phoneme can consist of multiple formants; the formant at the lowest

6 Spectral analysis usually involves transforming the time-based representation of the signal (e.g. sine wave) into the frequency domain (i.e. spectral domain) often done using Fourier transform.

frequency range is denoted as F1[7], at the next range is F2, and the range above that is F3, and so on. While a phoneme can consist of multiple formants, often it is enough to distinguish them by just the first two formants (F1 and F2). This property is helpful because the phoneme then can be stored with the least amount of information (data compression). Formant analysis is also called *pitch analysis* (Filipsson).

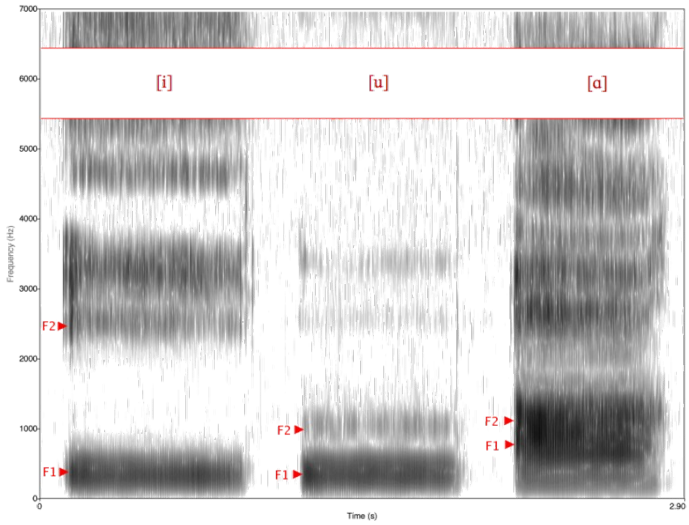
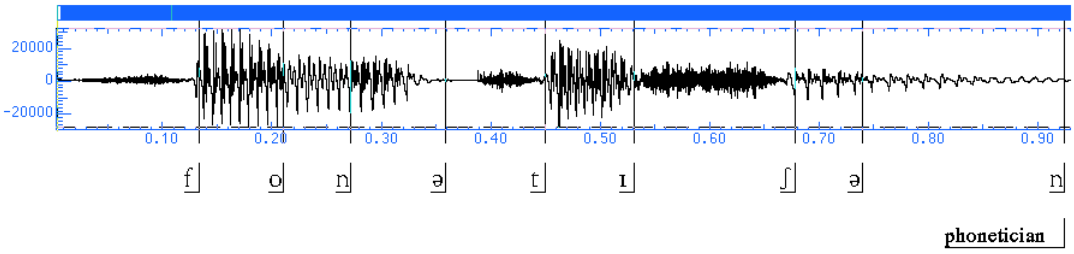


Figure 4.11 Spectrogram of English vowel i, u, and a (image taken from [Formantwiki])

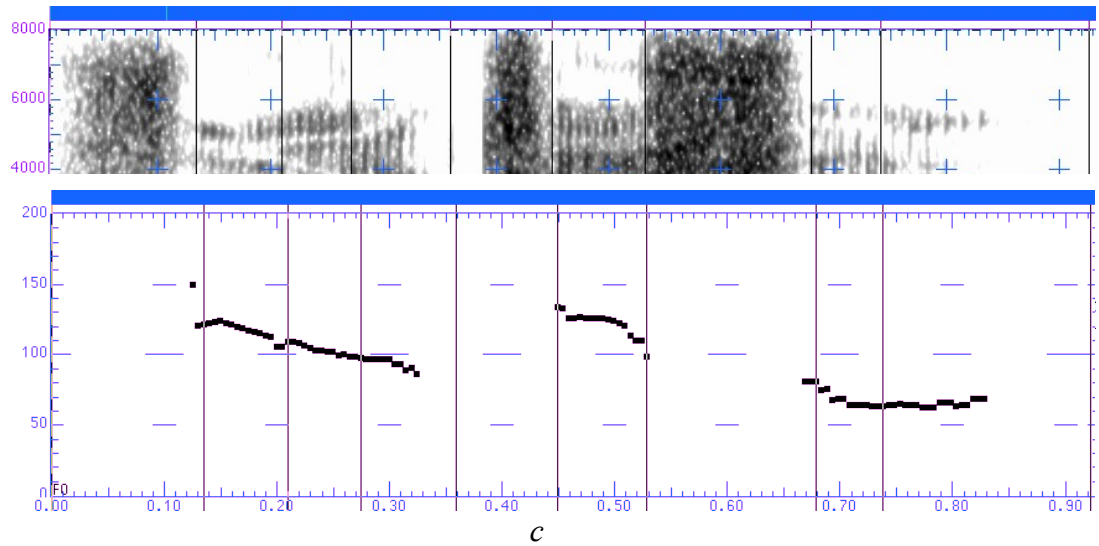
Aside from the formant model, there are other models that can be used to analyze speech. The most familiar form is the waveform model (figure 4.12a), which models the pronunciation of the word "phonetician" by a male voice, in English language[8]. The waveform model represents the pressure changes in the medium where the voice travels such as air. In Figure 4.12a, the X-axis represents time, and the Y-axis represents the pressure (in Pascal (Pa)). The other models are fundamental frequency

7 Some notations may start at zero (F0).
8 The nine segments are not part of the waveform, but shown to separate the phonemes in the pronunciation.

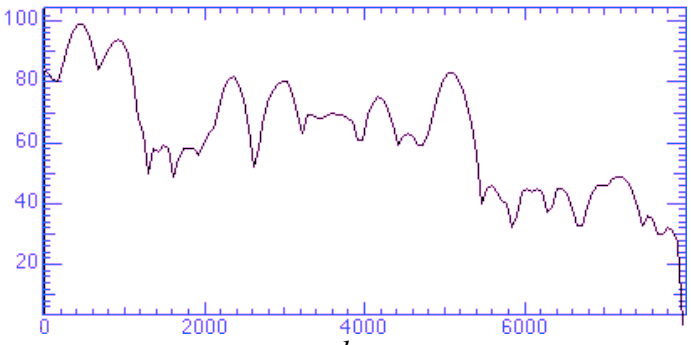
(Figure 4.12b[9]), spectrum (Figure 4.12c), spectrogram (Figure 4.12d), and waterfall spectrogram (Figure 4.12e). The fundamental frequency (Figure 4.12b) refers to base formant (F0), where the X-axis represents time, and the Y-axis represents frequency. The spectrogram representation (Figure 4.12c) shows the whole frequency range (and thus, all formants) of the voice, where the X-axis represents time, and the Y-axis represents frequency. The spectrum representation (Figure 4.12d) shows the frequency (X-axis) and amplitude (Y-axis) of a voice sample (Figure 4.12d shows the sample at 0.15 seconds into the utterance of "phonetician"). The waterfall spectrogram (Figure 4.12e) is a representation of the spectrum representation of Figure 4.12d over time. As shown, it has three dimensions: the frequency range (0-8000), the amplitude range (0-106), and time (0.004 – 0.952). To read more on these other forms, refer to [spatutorial].



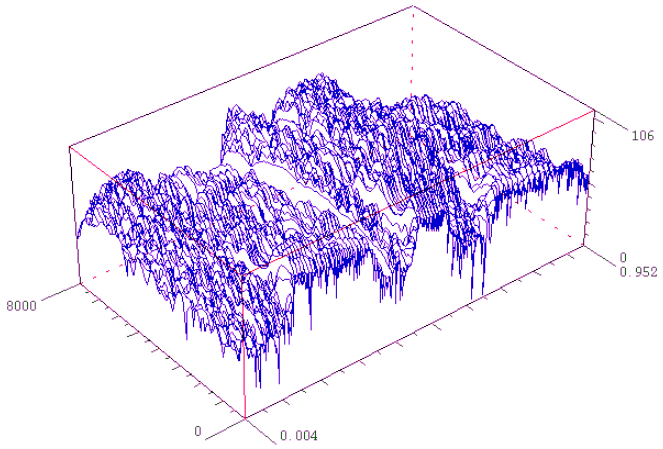
a



c



d



e

Figure 4.12 Different signal representations. (a) Pressure/vibration, (b) Fundamental frequency, (c) Spectrogram, (d) Spectrum, (e) Waterfall spectrogram. Graphs are referenced from [spatutorial].

The models discussed above are deterministic models; they use the physical properties of the voice (signal) such as frequency, amplitude, phase, and pressure as the parameters of the voice. Another type of model is the *statistical model* of the signal.

In the statistical model, a speech is modeled as a set of probabilities, usually obtained through a 'training' process. An example of the probability is: for a detected voice (e.g. phoneme), what is the probability of a particular word being uttered.

4.2.2.2 Classification

A common approach to the SR problem is by statistical methods, such as Hidden Markov Models (HMM). Usually, the spoken language is recognized by first extracting the acoustic input of its signal characteristics, stored as vectors of real values. The real values in the vectors are "*ceptral coefficients,which are obtained by taking a Fourier transform of a short time window of speech and decorrelating the spectrum using a cosine transform, then taking the first (most significant) coefficients."* [Wiki:For] The vectors are extracted at 30 to 100 Hz (every 10 to 30 milliseconds). HMM then evaluates the statistical properties of the signal characteristics to extract the probabilities that a certain word is uttered given that a certain symbol (i.e. letter, or set of letters in the form of a signal characteristic) appears in the speech.

The characteristics used are usually of acoustic models of phonemes (i.e. basic linguistic units), extracted using signal processing techniques. However, this issue becomes problematic when different languages are observed, where there exist different accents and pronunciations for the same alphabet symbols. In such case, usually different libraries of phonemes are created to accommodate the different

languages.

To recognize words from audible inputs, assume that there exists a library of phonemes of symbols $\bar{A}$ and a vocabulary of words $\bar{W}$. Let:

$$\boldsymbol{A} = a_1, a_2, a_3, \ldots, a_n \quad a_i \in \bar{A}$$

where $\boldsymbol{A}$ is a sequence of symbols, where each symbol a_i is a member of the set of alphabet $\bar{A}$. And let

$$\boldsymbol{W} = w_1, w_2, w_2, \ldots, w_m \quad w_j \in \bar{W}$$

where $\boldsymbol{W}$ is a string of m number of words, where each word w_j is a member of the set of vocabulary $\bar{W}$.

The probability that the word w_j is spoken, given that the symbol a_i is observed in the speech is:

$$P(\boldsymbol{W}|\boldsymbol{A})$$

Thus, to recognize that certain sequence of words (i.e. sentence) was spoken, all the system needs to do is to find the one with the highest probabilities:

$$\hat{W} = \underset{w}{argmax}\, P(W|A) \quad \text{or} \quad \hat{W} = \underset{w}{argmax}\, P(W)P(A|W)$$

where $\hat{\mathbf{W}}$ is the recognized word, $P(\boldsymbol{W})$ is the probability that the word $\boldsymbol{W}$ will occur (i.e. spoken), and $P(\boldsymbol{A}|\boldsymbol{W})$ is the probability that the symbol $\boldsymbol{A}$ will be observed, when the word $\boldsymbol{W}$ occurs. *Argmax* stands for *argument of the maximum*. For example, for a given function *f(x)*, *argmax f(x)* will return *x* that yields the maximum value for *f(x)*.

In our case, $\underset{w}{argmax}\, P(W|A)$ will return the *w* that yields the maximum value for $P(W|A)$ (or $P(W)P(A|W)$).

As mentioned in (Jelinek 1997), the recognition only identifies the words, but does not recognize *keywords for context*, that is, although the words are understood/recognized, the context may be missed.

The PSU Nautilus Capstone project team [Nau08] explored the use of SR for a speech-command control for a treadmill. The project gave an excellent insight to the issues of setting up a SR system.

In its applications, SR has three main implementations: real-time/continuous recognition, discrete speech, and word-spotting. Real-time/continuous SR systems are commonly used for dictation programs. The real-time system requires significant

computing resources as the system has to continuously monitor the speech, parse the sentence, and recognize each word uttered. The system also requires a large vocabulary database equivalent to a dictionary, and all the matching needs to be done in real-time. The discrete speech technique is cheaper computationally, but requires the speaker to speak with pauses between each word. Each pause allows the system to process the audio input stream and recognize the words, but the scheme makes the speech unnatural.

The word-spotting technique provides a good compromise between the previous two techniques by being relatively computationally cheap, while allowing the speaker to speak naturally. This is possible because the word-spotting technique only recognizes a few keywords rather than the whole language vocabulary (e.g. English). So for example, when the speaker says, “Robot, I want you to go left”. The system may have the words “Robot”, “Go”, and “Left” in its vocabulary, so the spoken sentence may be recognized by the system only as “Robot .... go left.” We can apply meanings to the words in the system's vocabulary, for example, the keyword “Robot” means that the speaker is giving order to the robot. The keyword “Go” may mean an order to travel, while the keyword “Left” means a direction for the robot.

If we expand this idea, we can apply a bit of context (or rule) to some of the vocabulary. For example, if the word “Left” is *preceded* by the word “Go”, the system

may understand it as an order for the robot to turn "Left" [10] and continue traveling forward in that direction. If the word "Left" is preceded by the word "Look", then the system only tells the robot to turn in place 90 degrees to the "Left" direction. The PSU Nautilus team indicated that they used the word-spotting technique because it fits the requirements of their application in terms of cost and performance[11].

In particular, the Nautilus team successfully increased the recognition accuracy from 82% with an untrained vocabulary, to over 97% after training. They were able to achieve higher accuracy rates by analyzing the hit/miss ratio of each word in the vocabulary using a Confusion Matrix. Using the matrix, several things can be observed:

- The hit rate of each word – from *n* times the word is spoken, how many times the system correctly recognized the word.
- The confusion rate of each word – from the missed/wrong recognitions, what words does the system confuse the spoken word with, and how many times it occurs.

These useful insights led the Team to analyze the vocabulary, and they decided that words that are often confused with other words in the vocabulary are to be replaced with their synonyms. There are words that are required to stay in the vocabulary, such

10 Let's say a "Left" or a "Right" is defined as a direction of 90 degrees to the left or right, respectively, from the robot's current heading.

11 Performance in terms of allowing the speaker to speak naturally (as opposed to discrete speech), and near-real-time responses.

as numbers from zero (0) to nine (9), thus they cannot be replaced. Their insight indicates that we can achieve good recognition performance if we can customize the vocabulary to the application, rather than being very general and hoping to cover every possible inputs. Moreover, if the system is trained using a particular voice, its recognition rate for that voice will be higher than other voices.

SR performance is often measured in Real-Time Factor (RTF), Word Error Rate (WER)or Single Word Error Rate (SWER), and Command Success Rate (CSR) [Wiki:SR]. RTF is measured as the ratio of the time it takes to process the input (P) over the duration of the input (I):

$$RTF = \frac{P}{I}$$

When P is equal to I, so the RTF = 1, then the performance is said to be *real-time*.

The WER is measured by the total number of words in the actual sentence (N) under the sum of the errors: number of words in the actual sentence that are confused with other words in the recognition (Substitution - S), the number of words that appear in the recognition but do not exist in the actual sentence (Insertion – I), and the number of words missing in the recognition (Deletion – D):

$$WER = \frac{S + I + D}{N}$$

The opposite measurement of WER can be done as Word Recognition Rate (WRR), which is defined as:

$$WRR = 1 - WER = \frac{N-(S+D)-I}{N} = \frac{H-I}{N}$$

Where $H = N-(S+D)$ is the number of correctly recognized words.

The downside of this measurement is that it is difficult to determine the Substitution error [mcowan]. For example, if an actual sentence is "I like fluffy cats", and the recognition returns "I leg flaw fee cats." We can easily tell that there is one Insertion error because there are four words in the actual sentence, but there are five words recognized, so I=1. The same can be said for Deletion errors (for example, if the recognition returns "I leg cats")[12]. Intuitively, we can also see that there is one Substitution error of replacing the word "like" with "leg", but it is difficult to determine the Substitution error caused by mis-recognizing "fluffy" with "flaw fee."

There is another variant of WER where the Deletion and Insertion errors are weighted:

12 It seems that there can only be either Insertion error or Deletion error in a recognition, but cannot be both. This can get confusing if what actually happens is one Deletion and one Insertion, but this may be considered as Substitution. Thus, this is one other reason why sometimes this measurement is questionable; which error affects the recognition?

$$WER = \frac{S + 0.5\mathrm{D} + 0.5\mathrm{I}}{N}$$

This form was proposed by [Hunt] to take count of errors that are more disruptive than others and those that are not as disruptive as others. According to [SRwiki], this form is still questionable for assessing a particular system, as it was better-suited to compare performances of different systems.

CSR or Command Success Rate measures the success of a speech recognition by the rate of successful/correct commands executed from a number of trials, despite that not all the words in the sentence may have been recognized correctly. In other words, even if the system cannot always accurately recognize the words, what matters is that the response (or context) is correct.

CHAPTER 5 – COGNITION

Cognition is the proces of understanding the input stimuli, and deciding the response to it. It involves making sense out of the input stimuli (i.e deriving context), and selecting the actions as a response to the stimuli, either to proceed with the interaction or to end it.

In the context of this thesis, cognition can be thought of as a control system. The complexity of cognition for a robot can be as simple as a hard-wired circuit in a *Braitenberg vehicle* (Braitenberg 1984), or as complex as 'intelligent' systems such as a PID controller, an artificial neural network (ANN), and so forth. In particular, our Cognition system is focused towards determining a set of motions or gestures, and the emotional expression that needs to be embedded in the motion. The goal is that the motion being executed by the robot becomes the visual feedback for the user about the emotional state of the robot.

For our application, the role of cognition is to take the information from the user's text inputs and use the information to select motions, and the parameter values for the motion modifiers. The workings of our Cognition system is illustrated in Figure 5.1, which is an excerpt from Figure 3.1 (whole system), and succeeds our Perception system, which was discussed in Chapter 4. The succession from the Perception system is shown by the input of the Cognition system, $I_{PC,}$ which is the output of the

Perception system. I_{PC} currently consists of the user's original input text, and the complete text sentence generated by ALICE as a response to the user's input sentence. The context module will extract keywords from both the user's input and ALICE's response sentences. Particularly, the context module will try to extract *action keywords*, and *emotive keywords*. Action keywords are keywords in the user's or ALICE's sentences that are mapped to a set of known actions such as: waving arm, push up, flying, dancing, and other actions. In other words, the action keywords determine what behavior (i.e. motion/action) should the system load from the motion database. Emotive keywords are a set of keywords that we defined which may invoke certain type of emotion *to some degree*. Our set of emotive keywords consists of words such as: stupid, death, kind, nice, dummy, compassion, angry, mad, and other similar adjectives. The emotive keywords are categorized by the type of emotion they may invoke, and with what intensity. The invoked emotion and the *intensity* of the emotion will determine the values for the motion processing parameters. Both the selected behavior and the values for the motion processing parameters becomes the output of the Cognition system, I_{CR}, and is passed to the Response system – specifically, to the Motion Processing module. The contents of I_{PC} is passed through without change, to be displayed on the monitor. A more detailed explanation of the emotive keywords are discussed further in Section 5.5 on our Emotional-Personality model.

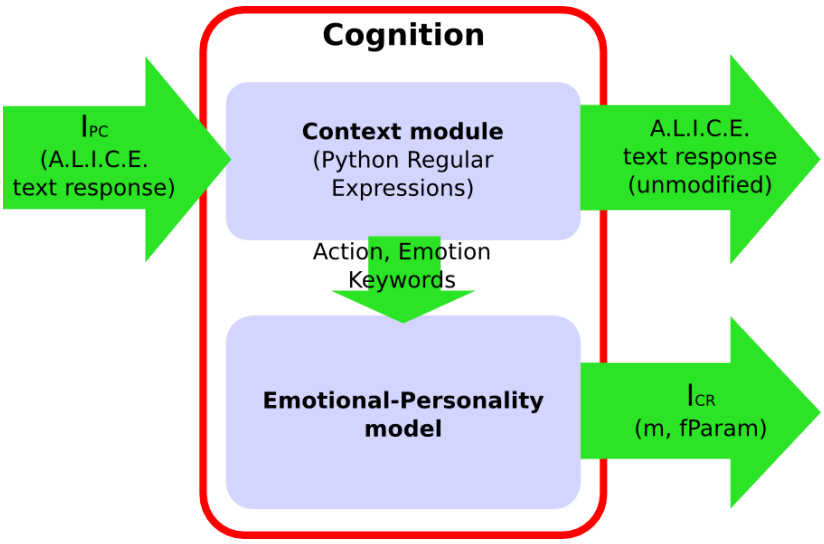


Figure 5.1 The implemented cognition system (excerpt from Figure 3.1). The input to the system, I_{PC}, is obtained from the output of the Perception system as discussed in Chapter 4.

For a more comprehensive cognition system, a recognized command may be deviated to a different command depending on the context of the interaction. The deviation may be caused by the emotional state of the robot, the 'personality' or preferences of the robot, other keywords recognized along with the command within the same sentence, other input stimuli (from vision, for example). For example, upon recognizing a command of "waving right arm", if the vision system detects that the person is showing an angry face expression, the robot may decide not to wave its right arm. Cynthia Breazeal developed a motivational system which allows such sophisticated behavior in the KISMET robot [Bre98].

Also, we can also implement or test cognition theories and/or concepts on interaction. Some of the concepts that are often used as the model of cognition for interactive

humanoid robots are (human) *communication norms*, emotions, and personality models.

5.1 Communication Norms

For Kismet, Breazeal applied basic communication/interaction concept such as *personal space*, *turn-taking,* and *irritability* [Bre02]. Personal space refers to the degree of comfortable distance between the two interacting parties. When the two parties are not in a close relationship, usually there is a certain distance that must be maintained for the interaction to be comfortable for both of them (although it may differ by culture). For example, when the robot senses that the person in front of it is too close, it will 'retreat' by moving slightly backward to establish a comfortable distance. In return, the person also retreat for a bit, because he/she senses that the robot is trying to establish the personal space. Turn-taking refers to the idea of waiting for one of the parties to finish their sentence or turn, before responding. This often can be determined by some period of pause in the persons speech, or intonation at the end of his/her sentence (e.g. question, statement, order, and so on) . Irritability refers to the speed of which the person is moving around or the object of interest is being moved around. For example, if the person keeps moving back and forth, closing in and away from Kismet, it will express irritation.

Other concepts in communication norms could be the entities such as touch, and

gestures of both parties. Touch could mean different things, depending on how established is the relationship between the interacting parties. For example, a punch is considered 'touch', but the amount of energy in the touch makes it often associated with hostility. However, when the relationship is established well, depending on where the punch lands and how much pain is inflicted, it could also mean a friendly gesture, even establishing a stronger bond. Conversely, a gentle touch is often associated with a kind or friendly gesture. However, when the relationship is not yet established, a gentle touch such as dabbing could affect the interaction negatively because such touch gestures actually only acceptable when the relationship is established. Other gestures could give some indication when a party is finished with his/her turn, for example when he/she actually stopped gesturing, or giving an open palm (or both palms), i.e. “please” gesture.

By detecting these situations, the robot cognition system can determine when to start the process of determining a response for the input stimuli.

5.2 Emotion-base

We refer to emotion as mental states such as happy, angry, sad, and fear. Emotion-based cognition refers to cognition that is influenced by the emotional state of the robot. For example, when a person wave his/her hand to greet the robot, if the emotion state of the robot is neutral or happy, the robot most likely will wave its hand to return

the gesture. However, when the robot is in angry or sad state, the robot may not return the wave. One generalization of this is if the robot is in neutral to happy state, the robot would be more responsive or engaging in interaction. However, in angry or sad state, the robot may be less likely (or interested) to engage in an interaction, and may tend to end the interaction as soon as possible, or try to avoid it altogether. We gauge this behavior as the *willingness*, which is affected by the intensity of any invoked emotions.

Plutchik's emotional model in figure 5.1 shows that each emotion has a range of intensity. Each spoke of the graph represent a type of emotion. The placement of the spokes corresponds to the similarities between neighboring emotion types. The emotions between spokes are a mixture of two neighboring emotions. Each spoke has four regions corresponding to their degrees of intensity. The outer region of the spoke is of the least intensity, and it increases as the region gets closer to the center of the graph. For example: the 'Joy' emotion spoke tells that the least intense form of 'Joy' is 'Serenity', and the most intense form is 'Ecstasy.' If we look at the neighbor spoke 'Trust', then we can say that 'Joy' and 'Trust' creates the emotion 'Love.' Conversely, the opposing emotion of 'Joy' is 'Grief', and so forth. Some might argue on Plutchik's choice of the types of emotion, but that is a topic for a whole different discussion (or thesis).

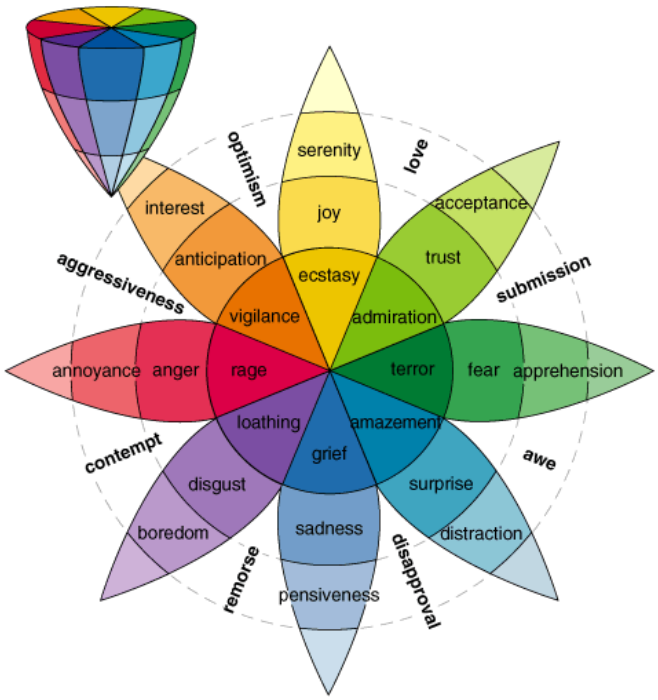


Figure 5.1 Plutchik Emotional Model

(Amaya et. al.) indicates from motion capture data that different kinds of emotion affect motion speed and motion range. In the authors' examples, compared to neutral emotion, the movement with angry emotion is faster and has bigger motion range, while with sad emotion, the same movement becomes slower and smaller. However, we have also seen instances of people exhibiting extreme sad or grief with movements that are fast and have big motion range. For example: a person with extreme grief may express their emotion by crying profusely, hitting their chests, running away, and other 'active' or 'energetic' behaviors. We often refer the state of extreme emotions as *hysterical*, or more precisely, when someone is expressing their emotion with very

'active'/'energetic' behaviors, we say that they are hysterical.

5.3 Personality-base

Personality is often used interchangibly with the term '*character*'. It refers to a set of behavioral characteristics or preferences that is inherent to a person [Net07]. It often determines how a person respond or behave when dealing with an event. For example: when a person is concentrating on reading a book, and an acquaintance called the person. The person can respond by getting annoyed because his/her concentration is disturbed, or can respond politely. The acquaintance may characterize the former response as the person's personality as being closed or unfriendly, while the latter being friendly.

The most well-known personality model is called *The Big Five* or *Five Factor Model* (FMM), which consists of five traits: *Openness, Conscientiousness, Extraversion, Agreeableness,* and *Neuroticism* [Wiki:BF]. Openness refers to how well a person accepts or learn new experiences. Conscientiousness is how careful a person acts, or driven by his/her conscience. A person who finds fulfillment or excitement by being social and with other people is called an *extravert* (or *'extrovert'*) and the trait is extraversion. When a person has low extraversion level, he/she is said to be an *intravert* (or *'introvert'*). Agreeableness refers how a person tends to create harmony in social interactions. Finally, neuroticism refers to a person's tendency to experience

negative emotions such as anxiety, anger, guilt, and so on (high level: more likely).

Cynthia Breazeal [Bre98] introduced a motivational system that controls the robot's preference to certain behaviors and also affects the emotive system of the robot.

5.4 Hypothesis

If we based our emotional model using Plutchik's model, we present our second hypothesis:

> *The type of emotion may dictate the kind of motion to execute, but does not determine the values of the parameters of the motion (speed, motion range) which instead are determined by the intensity of the emotion.*

So then, the implications of the above hypothesis are:

1. We cannot distinguish which emotion is being expressed solely through the speed and range of the motion.
2. We can only determine the *intensity of the emotion* through the speed and range of the motion.

If we assume the hypothesis is true, and we have the two implications of the hypothesis, how then we can have a robot express its emotion through its motion? In other words, this contradicts our previous argument that we can understand or identify emotions expressed through motion. We believe the answer is through *context*. This is

why we emphasize that it is first necessary to have interaction between the robot and human users to establish context. This follows a principle in the Disney Animation Principles guideline called *staging*.

The common approach to emotionally-expressive motions in animation or robots is to generalize that there are two polars of emotions: positive and negative. Positive emotions are associated with 'energetic emotions', or emotions which exhibit high energy such as “happy” and “angry.” Negative emotions are the opposite, such as “sad”, and “fear.” When they are translated to motions, the positive emotions are depicted as fast and have large motion ranges, while negative emotions are exhibited with slow and small motions. Our approach dissects this generalization into two parts:

- Type of emotion determines the type of action or motion to execute.
- The intensity of the emotion being experienced determines the speed and motion range of the motion.

As an illustration, imagine we are observing an actor acting within a sound-proof room, and the actor is wearing a mask such that we cannot see the his facial expressions. Facial expressions are a direct giveaway to emotions. When the actor's movements is fast and large, such as jumping around the room, waving his arms wildly, without knowing what he is thinking or feeling, think about the first question that comes to mind. Our first question is, “What is he doing?” The next question is,

“Is he angry, happy, or simply mad?”

5.5 Emotional-Personality Model

For our system, we implement an Emotional-Personality model. We liberally use the concepts in Plutchik's emotional model (Figure 5.1) and combine them with some of our own concepts and ideas about the relationship between emotion, personality, and their impact on a motion response. At this point, the model has not been tested rigorously for completeness, reachability, nor whether or not it is a realistic model. However, we find that the model sufficiently provide us with a reasonable simulation on the effect of a set of keywords to the degrees of emotional states. The relationship between the keywords, emotional states, and their effcts to motion responses can be considered as the *personality* aspect of the model. Here, we empirically determine the personality of the robot using our intuition about expressive motions from animation theories, which we discuss in greater detail in Chapter 9. Additionally, we add a 'willingness to cooperate' concept to our Emotional-Personality model.

5.5.1 Taking Ideas from Plutchik's Emotional Model

As already observed in section 5.2, Plutchik's emotional model has several interesting concepts. In our Emotional-Personality model, we used three concepts:

- The 'spokes' in the wheel of emotions are located by the affinities of the

emotions.

- Emotions on the opposing sides of the wheel indicate some opposing types of emotions. For example: Joy vs. Sadness.
- Each spoke has three layers.

To keep our model simple, we only use four extremes of emotions from Plutchik's wheel of emotion (Figure 5.1): Joy, Fear, Sadness, and Anger. Because of the concept of 'affinities' between these emotion, we preserve the relative 'positions' of these emotions according to Plutchik's model (Figure 5.2). By preserving the 'positions' of the emotion extremes, we also preserve the oppositions between the emotions: Joy vs. Sadness, and Fear vs. Anger.

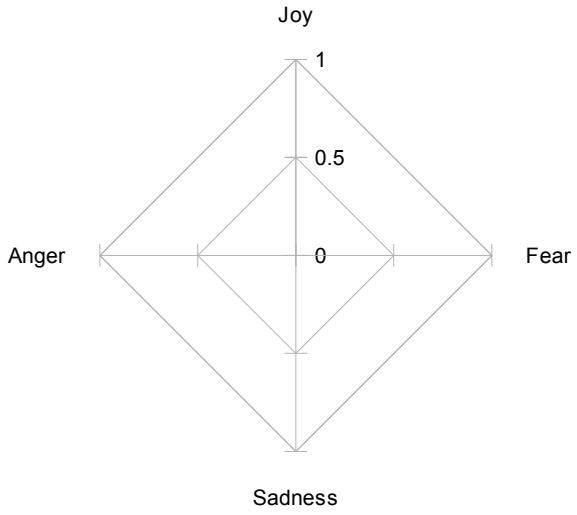
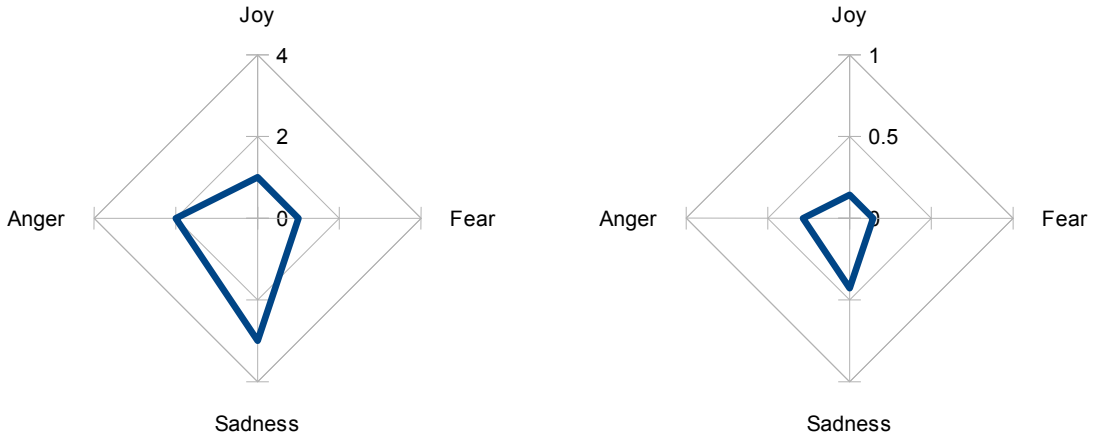


Figure 5.2 Modified and simplified Plutchik emotional model (Our model)

The concept of having three layers on each spoke (i.e. type of emotion) in Plutchik's model indicates that there are degrees of intensities of each type of emotion. We use

this concept in the following way: we can categorize *emotion-triggering keywords* into three degrees of severity: light severity, medium severity, and high severity. In our model, severities acts as *weights* or *scores*. Light, medium, and high severities are assigned with values: 1, 2, and 3, respectively. The keywords are extracted from the robot's text response generated by ALICE using Regular Expressions.



a *b*

Figure 5.3 Emotion graph example with: Joy=1, Fear=1, Sadness=3, Anger=2. a) Unnormalized graph, b) normalized graph.

The overall intensity of the emotion is determined not by the severities directly, but instead by the *centroid* of the area of the graph created by the severity values of the emotions. An example is shown in Figure 5.3 where the following emotional severites are perceived: Joy=1, Fear=1, Sadness=3, Anger=2. Before the centroid is calculated, the severity levels of the emotions are normalized. Once normalized, these emotion levels are referred to as the *emotion intensity levels*. The center of the graph is considered as the origin of a 2D coordinate system, the Joy-Sadness axis is the y-axis,

and the Fear-Anger axis is the x-axis. The position of the centroid of a N-cornered polygon can be calculated as [COM]:

$$x = \frac{1}{N}\sum_{i=1}^{N} x_i$$
$$y = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Since our emotion graph has four points, and the points are always on the axes, thus:

$$x_{emo} = \frac{1}{4} * (Fear - Anger)$$
$$y_{emo} = \frac{1}{4} * (Joy - Sadness)$$

Then, the intensity of emotion is calculated as:

$$Intensity_{emo} = MAX(x_{emo}, y_{emo})$$

From our example shown in Figure 5.3, we obtain the emotion intensity as:

$$Joy = 1,\ \ Fear = 1,\ \ Sadness = 3,\ \ Anger = 2$$
$$Normalized:\ \ Joy = 0.14,\ \ Fear = 0.29,\ \ Sadness = 0.43,\ \ Anger = 0.14$$
$$x_{emo} = \frac{1}{4} * (0.29 - 0.14) = 0.0125$$
$$y_{emo} = \frac{1}{4} * (0.14 - 0.43) = -0.0725$$

$$Intensity_{emo} = MAX(0.0125, -0.0725) = 0.0125$$

Initially, at the beginning of an interaction, the emotion intensity is reset to 'neutral' where Joy = Fear = Sadness = Anger = 0.25.

5.5.2 Keyword Severity

We created four sets of keywords, categorized by the type of emotions the keywords can trigger. Each set of keywords further categorize the keywords into three groups, based on the level of severity. Some of the keywords may overlap with multiple emotion category, but none of them overlaps with different severity levels within one emotion category. The full list of our set of emotion-triggering keywords can be seen in Table 5.1.

Table 5.1 Set of Keywords

Emotion	Severity		
	Low = 1	Medium = 2	High = 3
Joy	Peace, serene, play, nice, smart, friend, partner	Happy, joy, please, funny, cheer, beautiful, cute, great	Ecstatic, love, compassion
Fear	Danger, fear, punch, monster	Hit, slap, kick, ghost, devil, scary/scared	Terror/terrified/terrifying, blood, death, murder, kill, kidnap, bee
Sadness	Ugly, hate, lie, leave/leaving	Gone, miss, died, sad	Sorrow, grief, death, ignore, disappoint
Anger	Kill, kick, war, enemy, shoot	Stupid, angry, hate, mad, punch, hit	Rage, furious, cheat, lie, ugly, dummy, idiot

As shown in our example at the end of section 5.5.1, everytime one or more keywords are detected in the conversation (text inputs/responses), the severity levels of each keywords are added to the emotion intensity levels, and then normalized to produce the updated emotion intensity levels.

The categorization of the keywords into emotion categories and severity categories are not done based on any particular guidelines or researches found so far. Now let us assume for a moment that categorization of keywords into different severity levels in several emotion categories can be considered as a valid model to simulate the relationship between dialogue and emotion. And let us further assume that each person may perceive certain keywords differently. Then, using our model, the categorization process can be considered as creating *personalities* for the robot. Future works may involve investigating how to evolve the categorization process, thus the personality of the robot itself.

5.5.3 Relationship between Emotions and Motion Processing Parameters

Our motion processing components and their parameters are discussed in greater details in Chapter 9. However, to explain our model of the relationship between emotions and the motion processing parameters, we need to elude to the motion processing components briefly here. At this moment we will only mention that there are seven parameters that we use: *continuity, tension, bias, rate, highgain, mediumgain,* and *lowgain*. These seven parameters are parameters for three signal processng methods we used in our system to control the acceleration/deceleration, speed, and range of motion of a motion:

- interpolation - has parameters: continuity, tension, and bias

- resampling - has parameter: rate
- multiresolution filtering - has parameters: highgain, mediumgain, and lowgain

The value of each parameter is a function of the normalized intensity of each emotion: Joy, Fear, Sadness, and Anger. We determine the functions for each parameter using our knowledge of animation theories, which are explained in Chapter 8. From the animation theories, we gain some intuition on how emotions are conveyed in motions, which is also discussed in detail in Chapter 9. Using this intuition, we empirically determine the functions for the motion processing parameters, using the normalized emotion intensities as the parameters for the functions. The functions for the motion processing parameters are as follows:

$$
\begin{aligned}
continuity &= Joy - Fear + Sadness - Anger \\
tension &= -Joy + 0.5 * Fear + 0.5 * Sadness - Anger \\
bias &= -Joy + Fear + 0.5 * Sadness + Anger \\
rate &= Joy - Fear - Sadness + Anger \\
highgain &= -Joy + Fear + Sadness - Anger \\
mediumgain &= Joy - Fear - Sadness + Anger \\
lowgain &= Joy - Fear - Sadness + Anger
\end{aligned}
$$

We can represent this relationship in matrix format:

$$
M = \begin{bmatrix}
1 & -1 & 1 & -1 \\
-1 & 0.5 & 0.5 & -1 \\
-1 & 1 & 0.5 & 1 \\
1 & -1 & -1 & 1 \\
-1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1
\end{bmatrix}
\qquad
E = \begin{bmatrix}
Joy \\ Fear \\ Sadness \\ Anger
\end{bmatrix}
$$

$$M * E = \begin{bmatrix} continuity \\ tension \\ bias \\ rate \\ highgain \\ mediumgain \\ lowgain \end{bmatrix}$$

Where all the values in *E* are normalized. The motions to be executed are extracted also using Regular Expressions at the same time as the emotion keywords above. The motion (or motions) to which the seven motion processing parameters will be applied is determined by a simple mapping of verb keywords to motion data to be read from file. It is obvious that we can easily modify *M*, which effectively modifies the behavior of the system towards the motion processing parameters. Thus, this also means that the specification of *M* can be considered as creating the *personality* of the robot. Furthermore, *M* can be evolved using some evolvable computation such as Genetic Algorithm or Neural Network, assuming that an appropriate fitness function can be defined.

5.5.4 Willingness to Cooperate

The last component of our Emotional-Personality model is the concept of *willingness to cooperate*. 'Willingness to cooperate' or 'willingness' is defined as the probability of the robot to actually execute the action that is selected. We define this 'willingness' as *P* = [0,1] where:

$$P = Joy + 0.5 * Fear - 0.5 * Sadness - Anger$$

We determined the constants in the 'willingness' function using the following reasoning:

- When emotion has some amount of 'Joy', robot is most likely be willing to do any action asked/ordered by the user.
- When emotion has some amount of 'Fear', robot has some willingness (out of fear) to do the action asked/ordered by the user.
- When emotion has some amount of 'Sadness', robot is less willing to do any action asked/ordered by the user.
- When emotion has some amount of 'Anger', robot is unlikely be willing to do any action asked/orderd by the user.

To decide whether or not to execute the requested action, *P* is compared to a generated random number *R*, where R = [0,1]. If *R* is less than *P*, then the action is executed. Otherwise, the action is ignored. Note that we used the normalized emotion intensity values when calculating *P*.

5.5.5 Preservation of Emotional Levels

In our implementation of the Emotional-Personality model, the intensity levels of the emotions are preserved throughout the interaction. When a new keyword is detected which triggers an emotion with a degree of severity, then the severity is added to the

current intentisy level of the corresponding emotion category. Afterwards, the intensity levels of all the categories are normalized again. For example: from our previous example above, we have a normalized set of emotional intensity: [Joy, Fear, Sadness, Anger] = [0.14, 0.29, 0.43, 0.14]. And let's say the new keyword is under the Fear category with the low severity level (= 1). Then, the new set of emotional intensity is:

$$[Joy, Fear, Sadness, Anger] = [0.14, 1.29, 0.43, 0.14]$$
$$T = \sum (Joy, Fear, Sadness, Anger) = 2$$
$$[Joy_N, Fear_N, Sadness_N, Anger_N] = \frac{1}{T} * [Joy, Fear, Sadness, Anger]$$
$$[Joy_N, Fear_N, Sadness_N, Anger_N] = \frac{1}{2} * [0.14, 1.29, 0.43, 0.14]$$
$$= [0.07, 0.645, 0.215, 0.07]$$

This new normalized set of emotional intensity is then used to determine the values of the motion processing parameters and the 'willingness' level as described in section 5.5.3 and 5.5.4, respectively. Note that at any given time, an input sentence may contain one or more keywords that may contribute severity levels to multiple emotion categories at simultaneously.

Currently, however, the levels are not saved based on the user's profile. For future works, we will implement a saving feature to save these 'interaction states' which can be carried on or loaded when the same person is recognzied in the next interaction.

5.6 Output

Thus, there are nine outputs from the Cognition system which will serve as inputs to the Response system:

- Motion (or motions) to be executed
- Motion processing parameters:
    - For interpolation: continuity, tension, and bias values
    - For resampling: rate of sampling
    - For multiresolution filtering: high, medium, and low frequency band gains
- Texts from user's input, and responses from ALICE (just passed through from Perception system)

CHAPTER 6 – RESPONSE SYSTEM

The Response System is mainly responsible for delivering the chosen response by the Cognitive System in a manner that is understandable to the user. The Cognitive System decides *what* response to do, and *how* to deliver that response to the user. The Response System essentially just executes the responses dictated by the Cognitive System. Mainly, there are two forms of responses in our system: a sentence, and an action or movement. The sentence response is handled by the ALICE program, and serves as the verbal response from a user's inquiry. For motion, the system applies some signal transformation functions that modifies the chosen action to express the emotion in the action.

For the response system, we are focused on the delivery of the motions of the robot as the response to the given keyword, or the context of the interaction with the user. The decision of what kind of motion to execute, and howthe execution of the motion has already been done by the Cognition System. Thus, the Cognition System outputs are: the basic motion data, and the parameter values for the signal transformation functions, which become the inputs to the Response System. The Response System then applies all these transformation functions, with their corresponding parameter values, to the given motion data. The final output is then a motion response that is derived from the chosen motion data, but is now 'embedded' with additional

information of emotion, mood, or physical state of the robot. The verbal response is considered to be separate from our own Response System as it is being handled independently by ALICE.

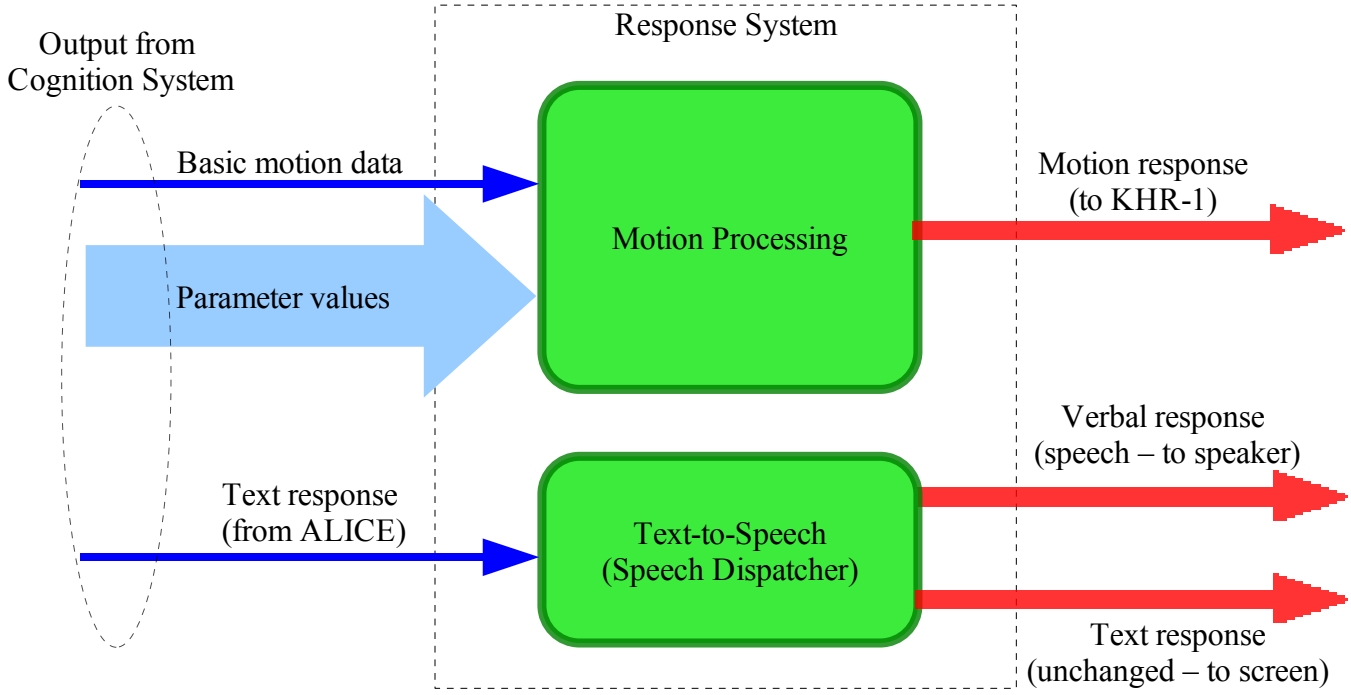


Figure 6.1 Input and output of the Response System

As seen in Figure 6.1, the Response System consists of two main components: Motion Processing, and Text-to-Speech. Motion Processing is responsible for applying the signal transformation functions to the basic motion data, based on the given parameter values. The Text-to-Speech component converts the text response from ALICE into speech, using a speech synthesis program called Speech Dispatcher. The signal

transformation functions used in the Motion Processing component include: Kochanek-Bartels spline interpolation, (re)sampling, and multiresolution filtering. Since the heart of this thesis is in this Motion Processing component, a deeper discussion of the subject is provided in section 9.2 on Motion Processing. For now, in this chapter we only provide a brief overview of this component.

The basic motion data only provides the key positions throughout the motion (in animation terms, these points are called *keyframes*). When the motion is executed, the execution is always linear, with constant zero acceleration or deceleration. This lack of acceleration or deceleration is often the main cause of the 'robotic' look in the movement. The spline interpolation method can eliviate the 'robotic' feel of the movement of the robot by adding position data between the keyframes that change non-linearly, thus giving the illusion of acceleration and deceleration. For the Kochanek-Bartels interpolation, there are three parameters involved: tension, bias, and continuity. These three parameters control the tangent of the spline, and may create different artifacts on the resulting motion.

Resampling or sampling refers to the method of taking only a few data points at certain interval from a large set of data points. Essentially, resampling is the opposite of interpolation – interpolation adds data points, sampling reduces data points. Resampling is needed to allows changing the *speed* of the motion.

Multiresolution filtering decomposes the motion data (which now is represented as a signal) into its lowpass and bandpass frequency components. The motion data can be reconstructed by simply summing up the bandpass components with the DC value. By adjusting the gains of the bandpass components, many effects can be applied to the motion, such as: removing noise or abrupt changes, removing or accentuate undulations in the motion, reducing or increasing the range of motion of the overall motion, to name a few.

CHAPTER 7 – KINEMATICS

7.1 Robot Motion Control Basics

In this chapter we will touch on the basic concepts of robot motion control such as the Denavit-Hartenberg (D-H) Notation and Parameters, Forward Kinematics, and Inverse Kinematics. Although these concepts are not implemented in the final product of this thesis, we feel that these methods are important to discuss, since the topic of this thesis *is* about the synthesis of robot motions. In particular, these concepts will be necessary when in the future, the response of the robot will involve interaction with objects the robot needs to reach, such as to shake the user's hand, picking up objects, emotional expressions related to the shape of the robot's body, and so forth.

We will start the discussions by introducing the terminologies used in these concepts,. Then, we will discuss the D-H Notation and Parameters, Forward Kinematics, Inverse Kinematics, and explain their use with some examples.

7.1.1 Terminologies

7.1.1.1 Degree of Freedom (DOF)

A DOF refers to a joint on a robot. The two terms can be used interchangeably (with one sounding more sophisticated than the other). A joint can be *translational* (also

called *prismatic*) or rotational (also called *revolute*). Translational joint allows the component (e.g. limb, arm) to extend or retract. In other words, sliding and change its length, similar to a piston. Rotational joint allows the component to be rotated, and is more common in anthropomorphic robots. The elbow, for example, has one DOF, and is a rotational joint. On a special note, a *spherical* joint is a joint that has three DOF, such as the human hip joint. In a 3D coordinate space (x, y, and z axes), the spherical joint can be rotated on all three axes. Thus, spherical joint can be considered a special case of a rotational joint, and normally represented as three rotational joints in one *location* (explained further in D-H Notation section).

7.1.1.2 End Effector

The term 'end effector' usually refers to the end point of a limb. For example, the end effector of an arm is the hand. That being said, an arm may have an end effector that is as complex as a human-like hand which has more than 20 DOFs, a gripper with 2 DOFs, or none at all (e.g. a welding tool, soldering point). In discussing arm movements, usually the end-effector is ignored and not included in the configuration calculation. For example, in controlling an arm to reach and grab an object, usually first the arm is controlled with the goal to position the end effector to be close enough to the object. Once the object is within reach, the arm movement calculation is done, then the grabbing movement of the end effector (e.g. hand) is calculated/executed.

7.1.1.3 Robot Work Space, Reach Space, or Reachable Space

The 'work space' or 'reach space' or 'reachable' space of a robot refers to the area that can be reached by the end effector of the moving component of the robot. It also implies the limits of the movement of the component. For example, the maximum reachable distance for an arm is when the arm is fully extended. Then, the maximum reachable space of the same arm is all the points that can be reached by the arm while it is fully extended (i.e. through all possible positions of the shoulder). Conversely, the minimum reachable point/space refers to the closest point that can be reached by the end effector to its origin (usually the base, or if it is an arm, the shoulder).

7.1.1.5 Motion Range

The term 'motion range' in our context is used as the amount of movement or how much change occured in the joint angles of the robot during the movement. Asssume the motion range of the robot is scaled from zero (0) to one (1). For this explanation, let's also assume that the axis of the servo of the robot can turn to a maximum of 180 degrees. If the starting position of the servo is at 0 degree, and the movement requires the servo to turn 180 degrees, then the movement uses the maximum motion range of the servo. Thus, when this happens, we say that the movement has a big motion range. Conversely, if the movement only require the servo to turn 10 degrees, we say that the movement has a small motion range. However, the concept of motion range can become highly subjective, that is, how much change is considered big, and how much is small? In our discussion, since all the servos in the robot have a range of only 180

degrees, we consider 0 to 60 degrees as “small”, 61 to 120 degrees as “medium”, and 120 degrees and beyond as “big,” essentially dividing the motion range in three equal ranges.

7.1.1.6 Local and Global Coordinate Frame/System

A coordinate frame/system in robotics is usually in 3D space, which has 3 axes: x, y, and z. Local coordinate system is the coordinate system of a joint, meaning the axis of the joint coincides with at least one of the axes of the frame. The convention used is that the DOF is always on the z axis. For prismatic joint, the translation is on the z axis. For revolute joint, the rotation is along the z axis. This will affect the values of the D-H Parameters (explained below).

The origin of the local coordinate system (local origin) is located at the center of the joint. A global coordinate system refers to the coordinate system which origin is manually defined; it does not have to coincide with any axis of any part of the robot. The origin can be one of the corners of the room the robot is in, but in case of an industrial robot arm, the global origin is often the local origin of the base joint.

7.1.2 Denavit-Hartenberg Notation (D-H Notation)

Denavit and Hartenberg developed the way to represent the geometrical positions of links and joints of a robot. Links are the parts between two joints, while joints are

DOFs. To represent this, D-H Notation uses several parameters called – naturally – *D-H Parameters*. In D-H Notation, each joint has its own coordinate frame (i.e. local coordinate frame), and is set up as follows:

- The axis of a joint (either prismatic or revolute) is on the z axis
- The x axis of the local coordinate frame is used as the common normal with the next joint. In other words, the x axis of joint *i* points to the direction of joint *i+1* (i.e. the next joint), or in-line with the link. Except, when the link to joint *i+1* coincides with the z axis of joint *i*.
- The y axis is determined using the right-hand rule based on the directions of the x and z axis.

The quick list of the D-H Parameters are (refer to figure 7.1):

- Link length: a_i
  - the perpendicular distance between the z axes of joint *i-1* (z_{i-1}) and joint *i* (z_i).
  - The length is measured along x_i.
  - There are three cases that determines where a_i intersects z_i:
    - z_{i-1} and z_1 are not coplanar: there exist exactly one a_i that intersects z_i and it would be at the minimal distance.
    - z_{i-1} and z_i are coplanar: there exsist infinite number of possible a_i that intersects z_i – the designer has the freedom to choose one, preferably one that will simplify the computation.
    - z_{i-1} and z_i instersect each other: then a_i equals 0.

- Link twist: α_i
  - the angle between the axes z_{i-1} and z_i, measured on a plane normal to x_i.
- Link offset: d_i
  - the offset of the origin of coordinate frame i (O_i) relative to the origin of coordinate frame$i$-1 (O_{i-1}) measured along the z axis of joint i-1 (z_{i-1}).
- Link rotation or joint angle: θ_i
  - the angle between the axes x_{i-1} and x_i, measured on a plane normal to z_{i-1}.

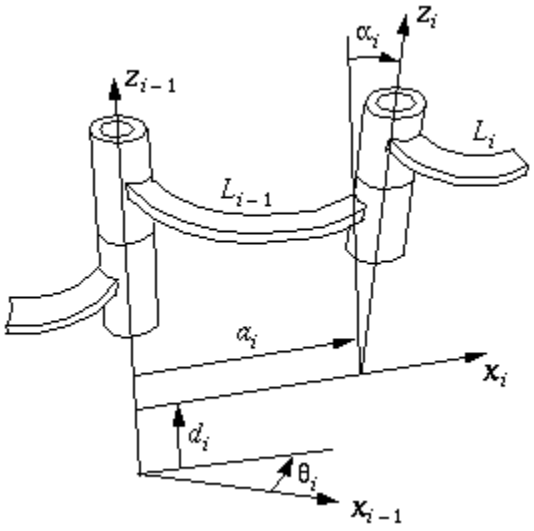


Figure 7.1 D-H Notation (image taken from: http://www.cs.cmu.edu/~rapidproto/mechanisms/chpt4.html)

The parameters can be thought as the transformations of the coordinate frame of joint i with respect to joint i-1. The transformations are:

- a_i, and d_i: translation
- α_i and θ_i: rotation

Thus, each transformation is represented by a transformation matrix, with:

- $T_{a_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (Translation of a_i along x_i)

- $Rot_{x_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (Rotation of α_i on x axis)

- $T_{d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (Translation of d_i along z_{i-1})

- $Rot_{z_i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (Rotation of θ_i on z axis)

And the relationship between joint *i* and joint *i-1* is then represented as:

$$^{i-1}T_i = Rot_{z_i} T_{z_i} T_{x_i} Rot_{x_i}$$, or

$$^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

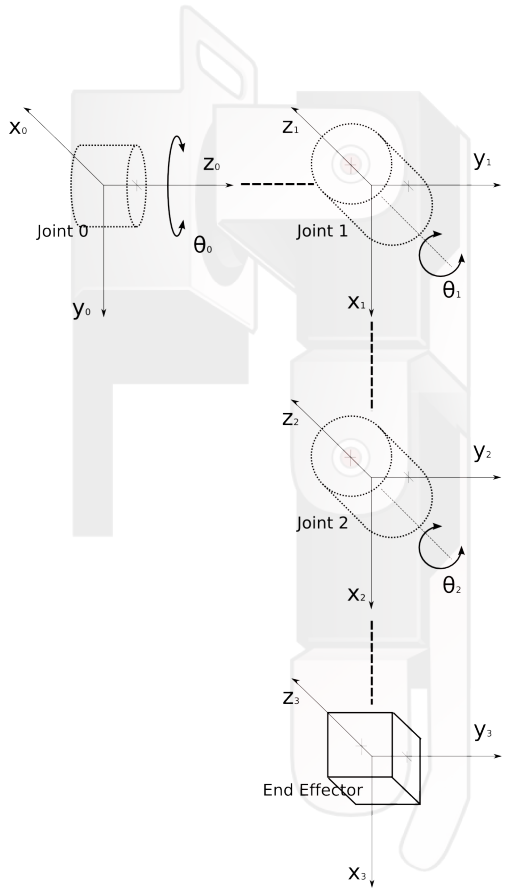


Figure 7.2 D-H Notation example using KHR-1 arm kinematics (not to scale).

For example, let us use the KHR-1 robot arm kinematics with three DOFs, which all are rotational joints. Let's examine the given configuration:

- The KHR-1 arm has two degrees of freedom at the shoulder: sagittal – which swings the arm forward up or down, and lateral – which swings the arm sideways up or down.

- For the sake of clarity when we define the D-H Parameters, let's enumerate each joint of this robot arm where the joints and End Effectors are:
    - The sagittal shoulder joint is Joint 0,
    - The lateral shoulder joint is Joint 1
    - The elbow joint is Joint 2,
    - and the End Effector is Joint 3.

      Such that the total transformation matrix will be denoted as $^{0}T_{3}$, which is read as *"The transformation matrix for point 3 (tip of the hand) relative to point 0 (shoulder joint)."*
- We define the sagittal shoulder joint as the base joint, denoted as Joint 0, which in our example in figure 6.# is the joint at the lower left corner. Thus, the coordinate frame of this joint becomes the Global Origin of the robot, and later our calculation for the position of the End Effector will be relative to this Origin.
- Since now we have defined our global origin at Joint 0, then the next immediate joint linked to Joint 0 is the lateral shoulder joint, denoted as Joint 1, and the next immediate joint linked to Joint 1 is the elbow joint, denoted as Joint 2.
- At the other end of Joint 2 we immediately see the End Effector or the tip of the KHR-1 hand frame.
- Therefore, we can define the D-H Parameters in this example as the following:
    - Link length a_1: the length of the link connecting Joint 0 to Joint 1. (Note:

the length of a_0 will be 0, since there is no link connecting Joint 0 with Joint -1, since Joint -1 does not exist).

- To simplify this example, let's assume the length for all links a_1, a_2, a_3 is 1 for some unit of length.
- Link twist α_1: the angle between the z axis of Joint 0 and the z axis of Joint 1. In this example, since z_1 is perpendicular to z_0, thus the value for α_1 is 90°.
- Link twist $\alpha 2 = \alpha 3 = 0°$ because z_2 is parallel with z_1. Since the End Effector does not have a DOF or joint, thus its orientation relative to Joint 2 will never change, it does not matter how the End Effector is oriented. Thus, to simplify the transformation matrix, we select the simplest orientation where z_3 is parallel to z_2, i.e. the same orientation as Joint 2.

- Notice that the rotation axis of Joint 1 (the z axis – according to our convention), is perpendicular to the axis of Joint 0. This is the one exception from the convention where it says that the x axis of joint i coincides with the link that is connected to joint $i+1$. We will see later that this peculiar exception is captured in the D-H parameter α_i (i.e. the *twist* parameter).
- Also notice that the local coordinate frames of Joint 1 and Joint 2 follow the normal convention: the x axis of Joint 1 coincides with the link which connects to Joint 2. The rotational axes of both Joint 1 and Joint 2 are on their z axes.

Thus, with the above configurations, we have the following assignments of D-H

Parameters for this robot: $a_i\ \alpha_i\ d_i\ \theta_i$

Table 7.1 D-H Parameter value assignments for example in Figure 7.1

D-H Parameter	Description	Value
a_1	Perpendicular distance (azimuth) between joint 0 and joint 1	0
a_2	Perpendicular distance between joint 1 and joint 2	1
a_3	Perpendicular distance between joint 2 and joint 3 (End Effector)	1
α_1	Angle between the z axis of joint 0 and the z axis of joint 1	90°
α_2	Angle between the z axis of joint 1 and the z axis of joint 2	0°
α_3	Angle between the z axis of joint 2 and the z axis of joint 3	0°
d_1	Displacement of joint 1 relative to joint 0 (displacement of origins of coordinate frame 1 relative to origin of coordinate frame 0, with respect to their x axes)	1
d_2	Displacement of joint 2 relative to joint 1	0
d_3	Displacement of joint 3 relative to joint 2	0
θ_1	Rotation of axis x_1 relative to axis x_0	90°
θ_2	Rotation of axis x_2 relative to axis x_1	0°
θ_3	Rotation of axis x_3 relative to axis x_2	0°

So, the Transformation matrices of ${}^{0}T_{1}$ is:

$$ {}^{0}T_{1}=\begin{bmatrix} \cos\theta_1 & -\sin\theta_1\cos\alpha_1 & \sin\theta_1\sin\alpha_1 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1\cos\alpha_1 & -\cos\theta_1\sin\alpha_1 & a_1\sin\theta_1 \\ 0 & \sin\alpha_1 & \cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{1}=\begin{bmatrix} \cos(90) & -\sin(90)\cos(90) & \sin(90)\sin(90) & 0.\cos(90) \\ \sin(90) & \cos(90)\cos(90) & -\cos(90)\sin(90) & 0.\sin(90) \\ 0 & \sin(90) & \cos(90) & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{1}=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Similarly, we can obtain ${}^{1}T_{2}$ and ${}^{2}T_{3}$:

$$ {}^{1}T_{2}=\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}T_{3}=\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

And the Total Transformation matrix is ${}^{0}T_{3}$ is:

$$ {}^{0}T_{3}={}^{0}T_{1}.{}^{1}T_{2}.{}^{2}T_{3} $$

$$ {}^{0}T_{3}=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The Total Transformation matrix ${}^{0}T_{3}$ describes the position and orientation of a point in the coordinate frame 3, as it is seen from the coordinate frame 0. Thus, a point $\mathbf{P_3}$ = [x=1, y=2, z=3] in coordinate frame 3, if seen from coordinate frame 0 is:

$$\boldsymbol{P} = {}^{0}T_{3} \cdot \boldsymbol{P}_{3}$$

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P} = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1 \end{bmatrix}$$

The transformation can be verified with the following illustrations:

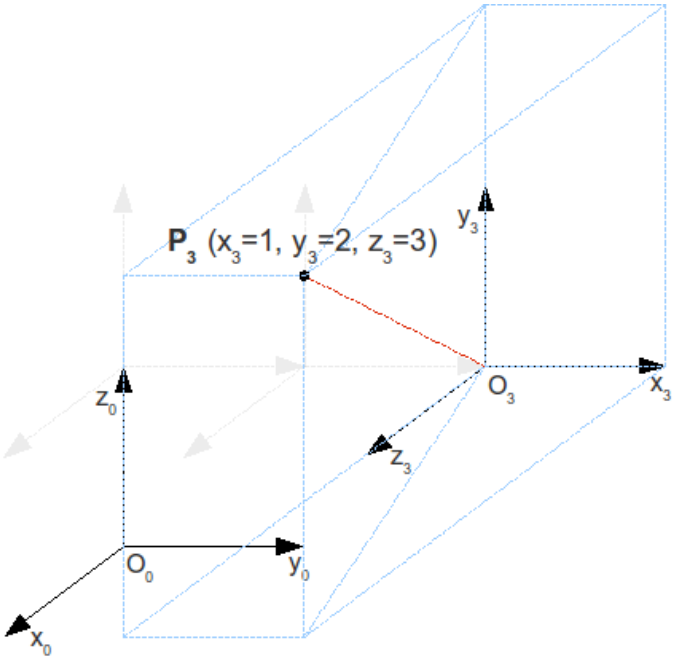


Figure 7.3 Point $\boldsymbol{P}_3$ with respect to coordinate frame 3 (x_3, y_3, z_3)

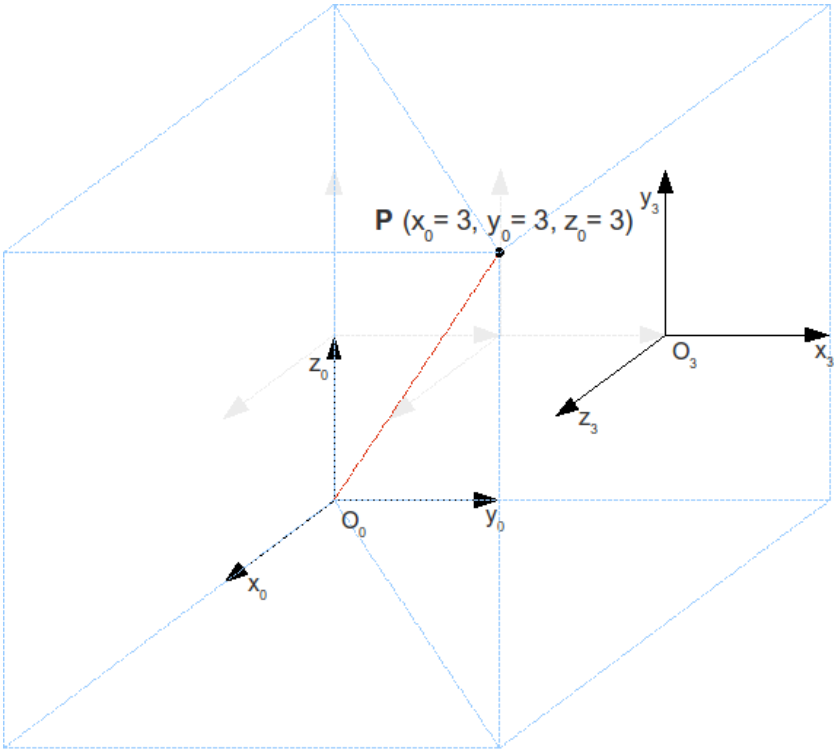


Figure 7.4 Point $\boldsymbol{P_3}$ *(here* ***P****) with respect to coordinate frame 0* (x_0, y_0, z_0)

With this example we have shown that by using D-H Notation to describe the kinematics of the robot, the position of the End Effector can be re-calculated at any given time. This is the working principle of *forward kinematics*.

7.2 Forward Kinematics

Simply put, forward kinematics (FK) is the computation of the position and orientation of the end effector as a function of all of its joint angles. It means, for a known kinematic chain of a robot, by knowing the positions of all the joint angles, the position and orientation of the end effector at the end of the kinematic chain can be

calculated with respect to the base coordinate frame. Or:

$$\boldsymbol{P} = f(\boldsymbol{\theta})$$

Where P is the vector (in the form of a column vector) of the final position of the end effector, $\boldsymbol{\theta}$ is a vector or set of joint angles, and $f(\boldsymbol{\theta})$ is the FK function, which consists of a total transformation function, multiplied by the start position of the end effector. The total transformations by the joint angles is described mathematically as:

$${}^{0}T_{n} = ({}^{0}T_{1}(\theta_1))({}^{1}T_{2}(\theta_2))({}^{2}T_{3}(\theta_3)) \ldots ({}^{n-1}T_{n}(\theta_n)) \quad \text{, or}$$

$${}^{0}T_{n} = \prod_{i=1}^{n} {}^{i-1}T_{i}(\theta_i)$$

Where:

$${}^{i-1}T_{i} = Rot_{z_i} T_{z_i} T_{x_i} Rot_{x_i}$$

which is obtained from the D-H Notation, describing the geometry of each joint with respect to the joint before it. The θ_i in ${}^{i-1}T_{i}(\theta_i)$ is the amount of rotation (angle) for joint i on the z axis (remember the convention above that the axis of the joint is aligned with the z axis). So, when the starting position of the end effector is known, represented as the column vector:

$$P_{Start} = [x \quad y \quad z \quad 1]^T$$

By pre-multiplying the total transformations with the start position of the end effector, we get the final position of the end effector:

$$P_{end} = {}^0T_n P_{Start}$$

7. Inverse Kinematics

As the name implies, inverse kinematics (IK) is the opposite process of FK. Given the desired final position of the end effector, the joint angle of each joint must be calculated. Which can be described as:

$$\theta = f^{-1}(\boldsymbol{P})$$

IK is a desirable feature in most humanoid robots as in most cases, what is known is the position where the End Effector needs to be. In computer animations and video games, IK is used by animators or the game engine to create believable physical interactions between the animated character with its environment. Figure 7.5 shows an example of IK using Maya 3-D animation software. On the model's left leg, there is a line connecting the left hip joint to the left ankle joint. This indicates that an IK link was established between the left hip joint, the left ankle joint, and any joints in between the two (in this case, the left knee joint). Also, in this IK link, the left ankle

joint is established as the End Effector. The three orthogonal arrows on the left ankle joint indicates that the joint is being selected, and let's refer to it as a 'handle'. Figure 7.5 right shows that, because of the established IK, the animator can just move the handle, and the IK solves for the position of the knee joint, and the rest of the left leg.

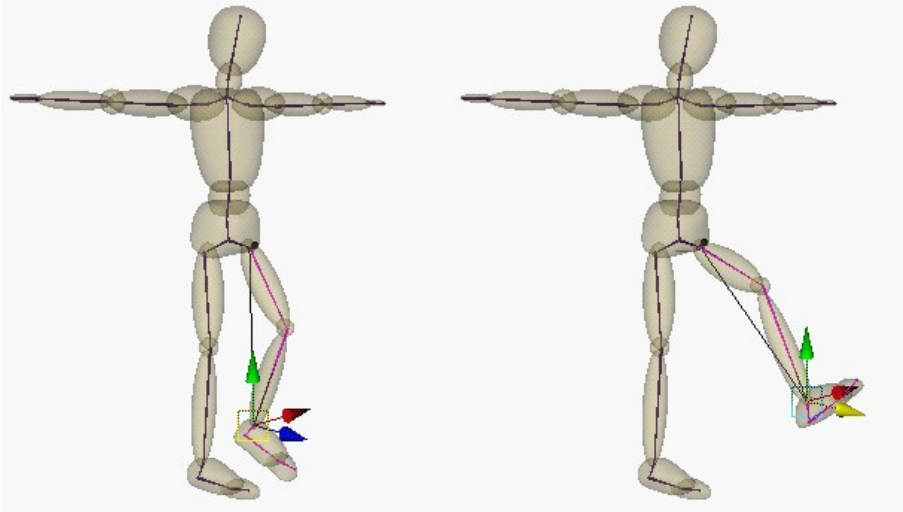
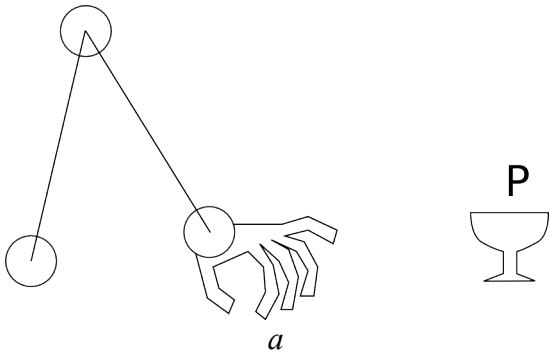
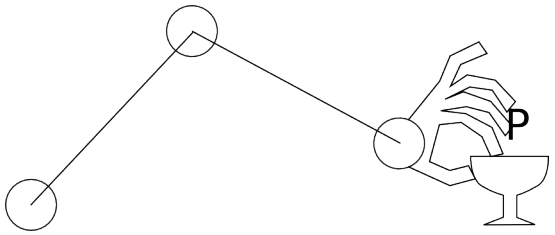


Figure 7.5 IK example in Maya 3-D animation software (image from: http://caad.arch.ethz.ch/info/maya/manual/UserGuide/CharSetup/images/SkeletonPose.fm.anc3.gif)

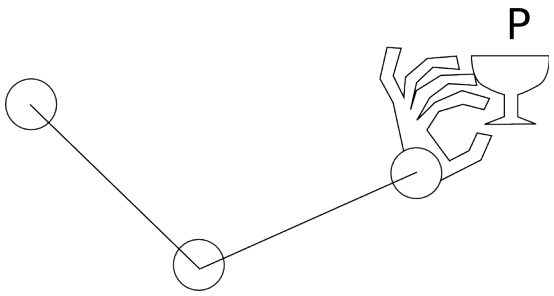
Solving an IK problem is dificult as there often exist many solutions for a given situation. For example, in the problem in figure 7.6a, both configuration A (figure 7.6b) and configuration B (figure 7.6c) allows the End Effector to reach point P. Both A and B are feasible and equally correct solutions.



a



b



c

Figure 7.5 (a) IK problem, (b) and (c) possible solutions

As the number of joints increases in a kinematic chain like in figure 7.6 (e.g. 6 joints instead of 3), the number of possible solutions increases. Needless to say, there are some advantages and disadvantages of having multiple solutions. One main advantage is that some solutions may have better characteristic than the others, perhaps in terms of the amount of energy used, the shape of the arm, the time to reach the target, and so forth. The disadvantage is that solving for IK becomes more complex. Often, *constraints* are used to limit the solution space; such as: limited

range of motion/joint angle, obstacle detection, desired path, and so forth.

Figure 7.6 shows an example of solving an IK problem using straightforward mathematical analysis, adapted from [http://www.learnaboutrobots.com/inverseKinematics.htm]. Given is the length of links l_1 and l_2, and the desired position of the End Effector (X_{EE}, Y_{EE}, θ_{EE}), the task is to find θ_1, and θ_2. X_{EE} and Y_{EE} is the position of the End Effector on the X axis and Y axis, respectively. θ_{EE} is the angle of orientation of the End Effector with respect to the X axis. L is the length of an imaginary line connecting the base joint with the End Effector, φ_1 is gthe angle between the imaginary line and the X axis, and φ_2 is the inner angle between link 1 (which length is l_1) and the imaginary line.

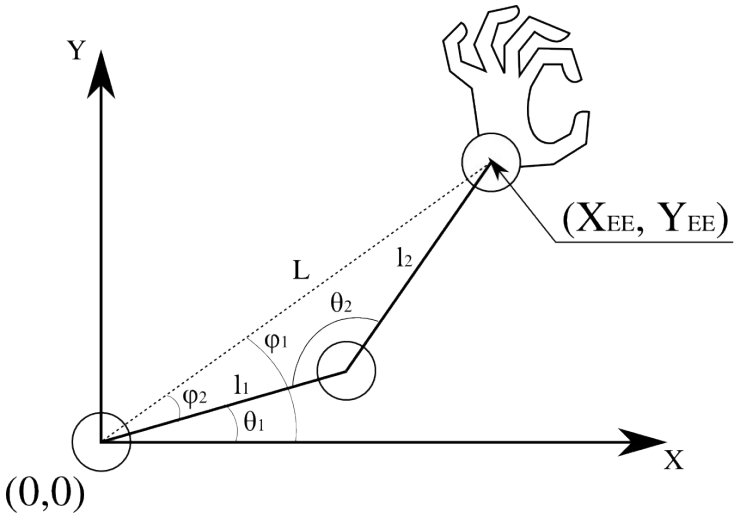


Figure 7.6 IK Example

Finding θ_1, and θ_2 can be done by solving for:

$$L^2 = {X_{EE}}^2 + {Y_{EE}}^2$$

$$\psi_1 = \tan^{-1}\left(\frac{Y_{EE}}{X_{EE}}\right)$$

The law of Cosine is illustrated in Figure 7.7:

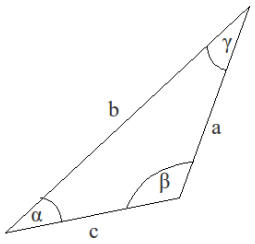


Figure 7.7 a general triangle

$$a^2 = b^2 + c^2 - 2\,b\,c\,Cos(\alpha)$$

Using this law, we can find φ_2 and θ_2:

$$\psi_2 = Cos^{-1}\left(\frac{L^2 + {l_1}^2 - {l_2}^2}{2\,L\,l_1}\right)$$

$$\theta_2 = Cos^{-1}\left(\frac{{l_1}^2 + {l_2}^2 - L^2}{2\,l_1\,l_2}\right)$$

Thus, we can find θ_1:

$$\theta_1 = \psi_1 - \psi_2$$

Assuming the values, (X_{EE}, Y_{EE}) = (10, 15), and l_1 = 10 and l_2 = 10, then:

$$L^2 = X_{EE}^2 + Y_{EE}^2$$
$$L = \sqrt{10^2 + 15^2} = 18.028$$

$$\psi_1 = \tan^{-1}(15/10) = 56.31^\circ$$

$$\psi_2 = \cos^{-1}\left(\frac{L^2 + l_1^2 - l_2^2}{2\,L\,l_1}\right)$$
$$\psi_2 = \cos^{-1}\left(\frac{325 + 100 - 100}{2 \times 18.028 \times 10}\right) = \cos^{-1}(0.901) = 25.66^\circ$$

$$\theta_2 = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - L^2}{2\,l_1\,l_2}\right)$$
$$\theta_2 = \cos^{-1}\left(\frac{100 + 100 - 325}{2 \times 10 \times 10}\right) = \cos^{-1}(-0.625) = 128.682^\circ$$

$$\theta_1 = \psi_1 - \psi_2$$
$$\theta_1 = 56.31^\circ - 25.66^\circ = 30.65^\circ$$

Consequently, it can be proven that with $\theta_1 = \psi_1 + \psi_2 = 81.97^\circ$, a feasible solution can be also found (the arm configuration will be mirrored along L).

CHAPTER 8 – EXPRESSIVE ANIMATION

In Chapter 7, we introduced the classical paradigm of creating movements on a robot in general, including the humanoid robots. Next, as our interest is in humanoid robots, we want to make the movements of the robot to be able to express human emotions. For this purpose, we take inspirations from the performing arts. In this chapter, we discuss two prominent guidelines that are popularly used by artists to create emotionally-expressive movements: the Laban Movement Analysis and the Disney Animation Principles.

8.1. Laban Movement Analysis

Laban Movement Analysis (LMA) is a system of language to understand, observe, describe, and analyze human movements (actions, motions, and gestures). LMA was first developed by Rudolf Laban in the early 1900's. Laban was a pioneer of modern dance in Europe and originated the movement study. LMA was originally used to direct dance choreography. Today, LMA is widely used in many fields that require human movement analysis such as dance choreography, sports [Ham95], and physical therapies. Laban's work was then extended by Irmgard Bartenieff by introducing the so-called 'Labanotation'.

Recently, there has been an increasing number of researches who apply LMA in

human-robot interaction (HRI) for socially interactive robots. LMA is being used to recognize human gestures to infer the meaning behind the gesture (i.e.: context) for the robot [Zha01]. LMA was also used to create parameters that generate gestures [CCZ+00]. However, so far the use of LMA in generating gestures have been applied to virtual agents (i.e.: computer generated), and have not been applied to physical agents (e.g.: robot) yet. We see a direct correlation between applying the technology in virtual agents and in physical agents. And while we acknowledged the different constraints and challenges in applying the technology in these two different media, we believe that this approach is still feasible.

The concepts in LMA are used to represent movement parameters We believe thus that if one can develop the computational models for them, it would be possible to generate movements for them. Hence, gestures will be created by specifying those parameters, instead of animating or moving each individual joints.

The EMOTE system focuses on the Shape and Effort categories as the motion generation parameters, by applying them to arm and torso movements, which are pre-programmed, keyframe animations (Effort was only implemented on arm movements).

Rett and Dias [RD] used LMA for *recognizing* gestures, instead of motion generation. Not only they used the Shape and Effort categories in their model, but also used the Space category to represent directions of the recognized gestures, and relative

positions or reach of the gestures to the center of the body as well. This was represented by the Kinesphere concept from LMA, which essentially describes the reachable space of the actor, including its directions, positions, and orientations – much like the 3D coordinate space in classical 3D animation. EMOTE also used the Space/ Kinesphere concept, but only limited to the arm movement, and not on the whole body [CCZ+00].

In [Zha01], LMA is used in terms of gesture acquisition, by extracting the Effort and Shape qualities of the movement. Zhao emphasized that their work is focused on extracting the movement qualities, and is different than gesture recognition – where a gesture is observed to be matched or classified with a set of known gestures. The acquisition is applied to a virtual agent (i.e. 3D computer-generated model). And because his work uses the EMOTE system as its basis, the acquisition is focused more towards the Effort and Shape qualities.

LMA provides a set of concepts (and thus, vocabulary) that describes the qualities of movement that are observable. These concepts are grouped into categories, and each category may have several sub-categories. At the lowest level of the hierarchy, the concepts are parameterized into two extremes and can be directly applied to a computational model. This is the main reason why we consider LMA as a good representations of motion parameters for robots. However, LMA has some 'grammar' where several concepts exist at the same time in a movement, but not all combinations

of concepts exist in the 'grammar'. We will explore this 'grammar' later in this chapter, but first let us get acquainted with the LMA concepts.

8.1.1 LMA Categories

First and foremost, LMA consists of four categories: Space, Body, Shape, and Effort. The following explanations are based on [Wiki:LMA].

- *Space* refers to the relationship of the gesture to the environment, such as the shape, patterns, or flow created by the movement of the gestures, and how they interact with the environment. In this category, the movement is observed through its use of space represented by:
  - Kinesphere: the reachable space around the body, represented as an icosahedron (20-sided polygon). The icosahedron is constructed from the three planes of movement direction in 3D space: horizontal, vertical, and sagittal planes. The horizontal plane is a plane which normal (or face) is perpendicular to the ground. The normal of the vertical plane is parallel to the ground, pointing the forward (or back) of the object. The normal of the sagittal plane is also parallel to the ground, but pointing to either sides (either left or right) of the object.
  - Spatial Intention: the direction or points in space that the actor is acting upon.
  - Geometrical observations of where the movement is being done such as: the position in space, the directions of the movements, the planes where the movement takes place, and so on.

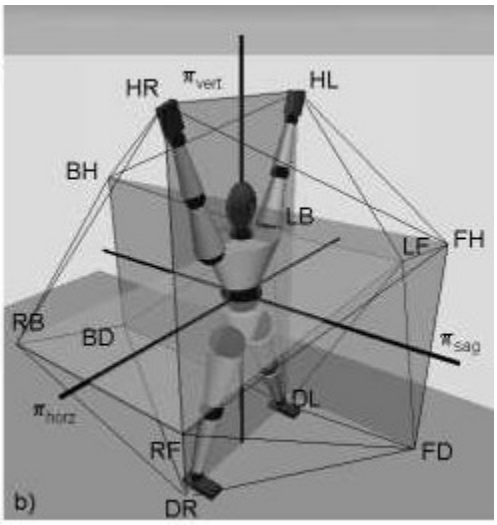


Figure 8.1. Kinesphere. The icosahedron is constructed by the Horizontal plane (RB-RF-LF-LB), vertical plane (HR-HL-DL-DR), and sagittal plane (FH-FD-BD-BH).

- *Body* refers to the structure, organization of the moving parts of the human body to create the movements, and the structure (hierarchy), connection between the body parts (i.e. limbs) that are involved in the movements. Irmgard Bartenieff expanded this category into several subcategories:
  - Initiation of movement from a specific body part(s) – which body part initiate the movment
  - Relationship of one body part with other body parts – e.g.: the hand is connected to the lower arm through the wrist, the lower arm is connected to the upper arm through the elbow, the upper arm – and thus, the whole arm – is connected to the torso through the shoulder, and so on.
  - Sequencing of movements between the body parts – e.g.: for a reaching motion, does the movement started by moving the wrist, elbow, or shoulder

first, what body part moves next and how?

- Patterns of body organization and connectivity - ... *don't know yet...*

- *Shape* refers to how the overall shape of the human body during the movement, e.g.: 'expanding' (imagine stretching the arms and legs out), 'shirinking' (imagine cowering), and so on, and the meanings behind them. Shape has several subcategories:
  - Shape Forms: describes the shape created by the body, e.g.: ball-like, tree-like, and so on.
  - Modes of Shape Change: describes the relationship of the body with itself, or with the environment in the movement. There are three modes:
    - Shape flow: represent the interaction between the body with itself (no direct interaction with the environment). Such as: shrugging, scratching head, and so on.
    - Directional: represent some interaction of the body with the environment or an object in the environment, where the interaction does not cause a change of shape to the objects. Such as: pointing at an object in the environment, wiping a window, and so on.
    - Carving: represent interaction of the body with an object in the environment where the interaction causes the object to change shape. Such as: kneading bread dough, squeezing an orange, and so on.
- *Effort* refers to the dynamics or control of the movement, such as speed,

acceleration, weight (or the illusion of it), freedom of movement, and so on. Effort has four subcategories for those dynamics: Space, Weight, Time, and Flow.

- Space: Direct vs. Indirect
    - Direct: the movement is directed to a certain target in an efficient path, e.g.: pointing at someone.
    - Indirect (also referred to as Flexible [Bis92]: the movement is directed to a certain target but requires some extra movement, e.g.: swatting a fly – requires a swinging action.
- Weight: Strong vs. Light
    - Strong: the movement (appears to) requires a lot of energy, seen either by speed, acceleration, or weight of an object, e.g.: punching, lifting a heavy object.
    - Light: the movement (appears to) be floating, or soft, e.g.: carressing, patting.
- Time: Sudden vs. Sustained
    - Sudden: fast, surprising movements, e.g.: slapping, punching.
    - Sustained: careful, precise movements, e.g.: reaching, golf(?).
- Flow: Bound vs. Free
    - Bound: the movement follows a path that is limited, or constrained – sometimes discrete, or rigid, e.g.: walking between bookshelves in the library (i.e. a maze).
    - Free: the movement follows a continuous path, often in wide arc, reaching

the full extension of the body parts (arms, legs, torso, etc.) e.g.: waving a hand to stop a taxi.

In addition, there is a fifth category called *Phrasing*. Phrasing refers to the sequencing of the movements over time. [Bis07] indicates that phrasing of movements is analogous to constructing a verbal sentence using words, or a song using musical phrases that ultimately convey a meaningful message. Phrasing consists of three stages: *Preparation*, *Action*, and *Recuperation*.

- Preparation refers to the movement before Action. For example: before jumping over a puddle, a person would gauge to make sure his/her jump would be far enough to avoid the puddle by crouching slightly to gain momentum during the jump.
- Action refers to the main (intended) movement. For example: the action of 'jumping-over-the-puddle'. It goes from when the person's feet leave the ground, over the puddle, and when they touch the ground again.
- Recuperation refers to the movement after the Action to position the body back to its 'normal' pose. For example: after the jump in the previous example, after the person's feet touch the ground, the person will slightly lean forward, and bend his/her knees and ankles to absorb the momentum from the impact. Once the momentum is absorbed enough to not cause imbalance, then the person will start straighten his/her body up to its 'normal' pose.

We will see later on that the Phrasing category in LMA has direct correlation with the Anticipation, Squash and Stretch, and Follow Through and Overlapping in the Disney Animation Principles discussed in the next chapter.

8.1.2 Intentionalities in LMA Categories

According to Bishko, sometimes the *intentions* are missing from the movements of an animated character – in our case, a robot – that it makes the character becomes unnatural, or lacks 'life'. This is an issue also seen in poorly animated video game or animated movie characters. They have been programmed *functionally*, and not *expressively* [Bis07]. If the movements are such that it expresses the robot's 'intentions', it shows that the robot is 'alive' and even may show some kind of personality. Perhaps because we see humans by this view, the humans always move intentionally [Zha01], as opposed to simply following orders (i.e. merely functional).

LMA's Effort subcategories (qualities) has a relation with humans' *ego functions*: Sensing, Thinking, Intuiting, and Feeling [Bis07]. The following relationships of the ego functions with the Effort qualities are described in [Bis92].

- Space relates to Thinking and Attention, also referred to as the *Where.*

  Space describes the attention of the actor's orientation to his/her surrounding space. When the actor goes through many points in his/her space, the actor is said to have a broad awareness to his/her space (Flexible/Indirect). Otherwise, when the movement is described as having a focused attention (Direct).

- Weight relates to Sensing and Intention, also referred to as the *What.*

  Weight can be considered as how the actor senses and reacts to gravity. Despite of the gravity pull, if the actor can move 'lightly' as if without effort (Light), the movement is said to show 'easy intention'. In adverse, if the actor moves 'forcefully' or with strength (Strong), the movement shows 'strong intention' or 'determination'.
- Time relates to Intuition and Decision, also referred to as the *When.*

  Time relates to the actor's intuitive decision on . A continuous or lingering movement (Sustained) shows indulgence, while an abrupt movement (Sudden) shows surprise, or something unexpected.
- Flow relates to Feeling and Progression, also referred to as the *How.*

  Flow is used to describe the release of energy seen through the progression of the movements. A free flow (Free) is described as external release of energy, while a constrained/resisted flow (Bound) describes the energy as directed inwards.

When these qualities are combined they form a mental ground for the movement. A combination of two Effort qualities forms *incomplete efforts* or *inner attitudes*. There are six combinations:

- Space and Weight (stable, where, what steady, balanced)
- Space and Time (: awake where, when alert [Bis92])
- Space and Flow (remote where, how small pensive, visual, projecting outwards)
- Weight and Time (near what, what earthy, rhythmic)

- Weight and Flow (dreamlike how, what bodily feelings, hazy fantasies)
- Time and Flow (mobile when, how getting on, progressing)

When three of the Effort qualities are combined, they create *externalized drives*:

- Space, Weight, and Time create *Action Drive*
- Space, Weight, and Flow create *Spell Drive*
- Space, Time, and Flow create *Vision Drive*
- Weight, Time, and Flow create *Passion Drive*

Within the Action Drive, there are eight actions called Basic Effort Actions, as shown in Figure 8.2. These actions are created through the combinations of the extremes of the Effort qualities Space, Weight, and Time that constitutes the drive. The action pairs are:

- Thrust (Direct, Strong, Sudden) – Float (Indirect, Light, Sustained)
- Press (Direct, Strong, Sustained) – Flick (Indirect, Light, Sudden)
- Glide (Direct, Light, Sustained) – Slash (Indirect, Strong, Sudden)
- Dab (Direct, Light, Sudden) – Wring (Indirect, Strong, Sustained)

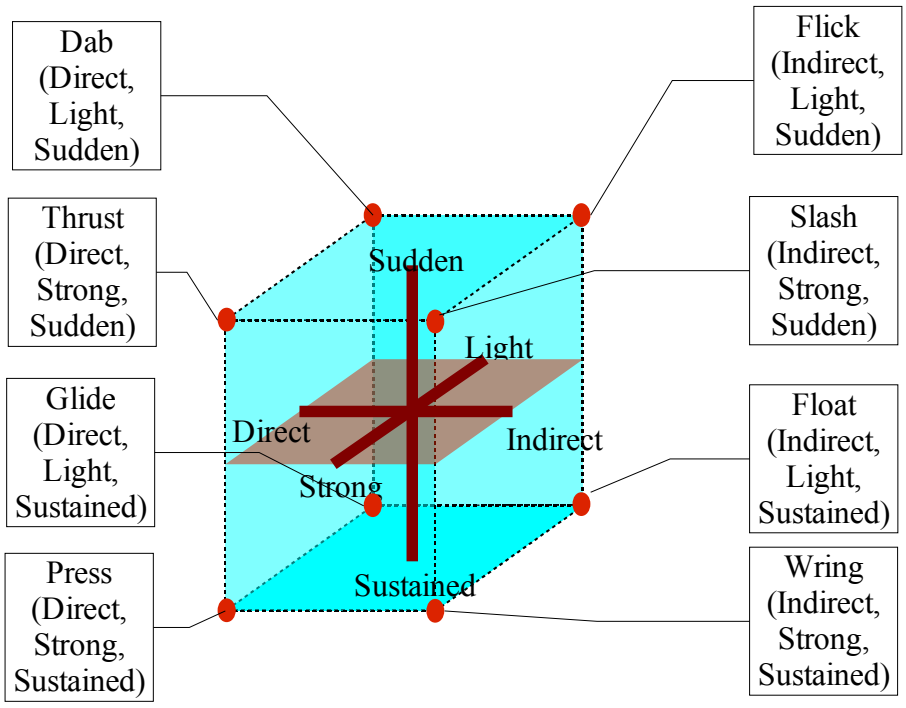


Figure 8.2. Basic Effort Actions

Because the Eight Basic Effort Actions are defined by the extremities of the Effort qualities, we can only assume that it will create confusion when an action is right in the middle of one or more Effort qualities. Even worse, if the action is neither Sudden or Sustained, Strong or Light, Direct or Indirect, it may be almost impossible to classify the action.

There is no equivalent to the Basic Actions on the other Externalized Drives. However, [Bis92] that the other Drives have the following interpretations:

- Passion Drive: enables expressive and emotional actions through changes in body form.

- Spell Drive: creates trance-like movements that are somewhat random and without rhythm.
- Vision Drive: described as *"movement lacking bodily awareness and tactile control"*, which we can consider as movements where the actor has no regard to the environment. In other words, analogous to the movement of most robots that do not have sensors, thus the movement is not a direct reaction to perception.

LMA practitioners use an Effort Graph to represent the elements of Effort [Bis92], seen in Figure 8.3. They can summarize the desired movement quality by drawing parts of the graph that represent the Effort qualities that they want.

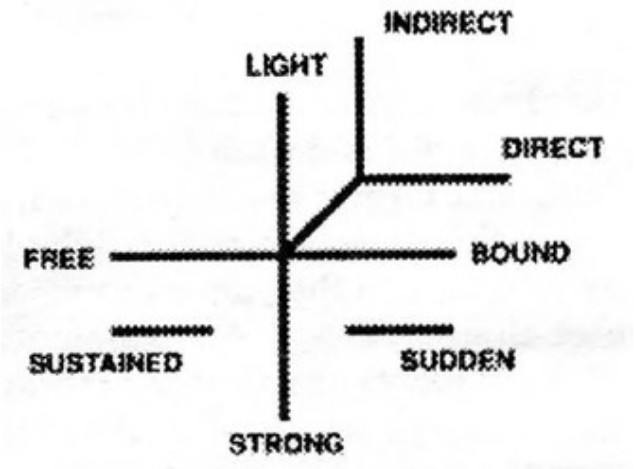


Figure 8.3. Effort Graph

The table in Figure 8.4 shows how the Effort Graph is being used to symbolize the desired movement qualities for both Effort and Shape. As we can see, the Shape

qualities are differentiated by drawing the diagonal part with two lines. In practice, the Effort graph is used in shorthand form shown in Figure 8.4.

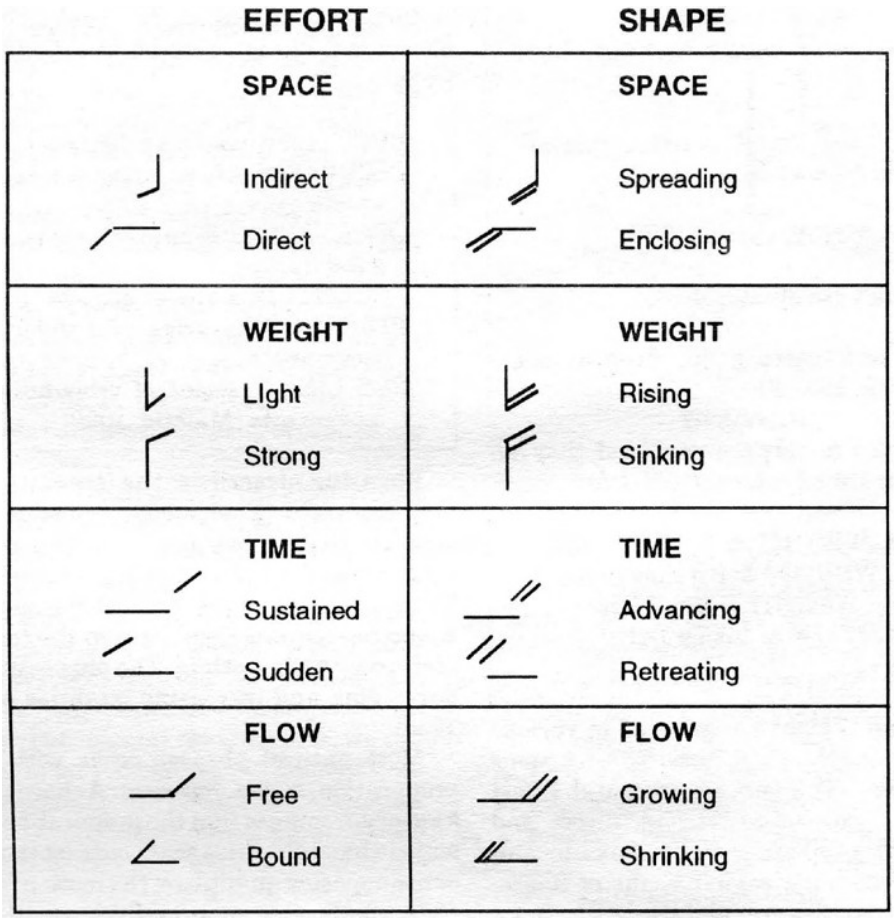


Figure 8.4. Shorthand of the Effort graph.

8.1.3 Phrasing

Phrasing in LMA refers to the pattern or rhythm of the movements. In contrast to our previous discussions which only focused on a *single* movement, Phrasing concerns

with a ***set*** of movements and how they create messages.

According to [Bis92], there are six classes of Phrasing:

- Even: the movements are continuous from one to another, without changes in intensity
- Increasing: the movements increase in intensity as they progress, usually signified by acceleration
- Decreasing: the opposite of Increasing – the movements decrease in intensity in the process, signified by deceleration
- Accented: "series of accents"
- Vibratory: a series of repetitive movements
- Resilient: a series of movements that rebounds from one to the next

8.1.4 Summary of LMA for Humanoid Robot Animation

The LMA theory was deducted from extensive observations on human movements, by other humans. Arguably, there will be elements of the theory that will be difficult to realize in an artificial kinematic system such as a humanoid robot, for example: body or muscle tension, which is one of the main features of interest of the LMA Effort parameters. Physically, tension on the muscles of a human body can be observed by the naked eye. As robots do not use muscles as their actuators, muscle tension is very much absent on robot movements, which makes it difficult to say whether or not the Effort quality exists in the robot's motion. This claim may be argued for robots with

McKibben muscles [Tondu, Lopez], or where the actuators look and work like biological muscles, but that is a whole different discussion which will not be addressed further in this thesis.

Besides the physiology of the actor, LMA rightfully includes concepts of how the actor interact with his/her environment (e.g. objects, forces such as gravity, and so on) and can thus add meaning to the movement. In other words, when the actor's actions are synergistic with his/her environment, the actor is seen as actively sensing his/her environment, thus creates a sense that the actor is aware, awake, or alive. In a robotic system, this sensing implies the use of sensors, or some kind of inputs that somehow affect the robot's movement. The sensing may involve: distance, object recognition, force, touch, sound/voice (including speech), and so on.

8.2 Disney Animation Principles

The Disney Animation Principles (we will refer it as Animation Principles from here on) are the basic animation knowledge that every animator and animation student refers to when creating or learning how to animate. The Animation Principles are different from LMA in the way that while LMA is an analysis of movements, the animation principles are guidelines to create believable, natural, and interesting animations because they are originally developed for visual entertainment (e.g. cartoons). We can think of LMA as suitable for the analysis of movements, while the Disney animation principles as the base for the synthesis of movements.

There are twelve basic concepts in this Animation Principles: *Staging, Anticipation, Stretch & Squash,Follow Through and Overlap, Timing & Motion, Slow-In and Out, Exaggeration, Secondary Action, Arcs, Straight Ahead and Pose-to-pose Action, Appeal,* and *Solid Drawing*. As the Animation Principles were developed for Disney's 2D animations, the Solid Drawing principle refers to the consistency of the drawing throughout the animations. Thus, as indicated by [Las87], the Solid Drawing principle does not apply to mediums where drawing skills are not required to create the motions, such as 3D animations, and so it does not apply to our robot 'medium' as well. Therefore, we will consider only the remaining eleven principles in helping us synthesize expressive motions for our humanoid robot KHR-1.

To clarify the discussion on the Animation Principles, we will use a jumping-in-place sequence of movements in Figure 8.5 which will serve as a reference to our humanoid robot KHR-1.

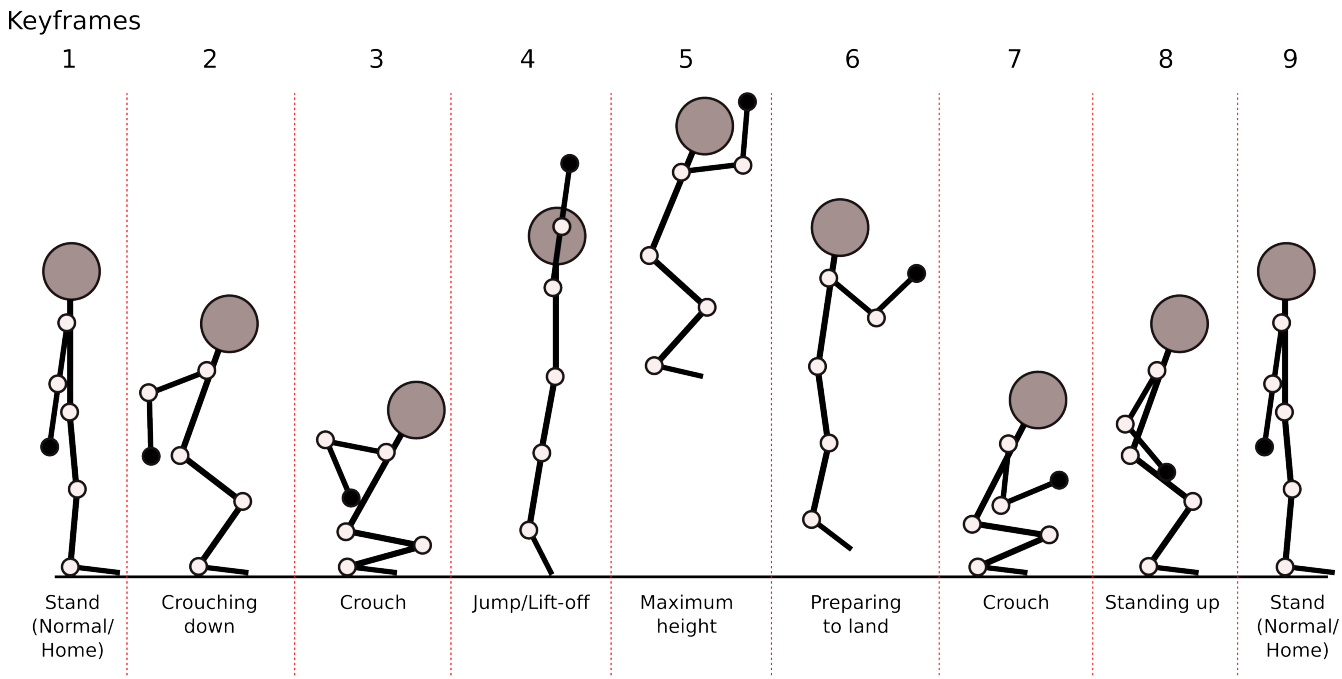


Figure 8.5 Jumping action in nine sequences (keyframes).

8.2.1 Staging

Staging refers to the presentation of an idea that is to be conveyed by the animation, and is a very theatrical concept. The presentation needs to be as clear as possible, such that it does not confuse the audience of what is being communicated (e.g. having a double meaning, or does not make sense). Note that it still depends on the intentions of the idea; in art, it is still legitimate to present an idea with the purpose of confusing the audience (i.e. stimulate), but must be executed perfectly and with great cautions. Some examples of staging are: depiction of the mood of the character, positioning the character on the stage so the audience can anticipate what the character's next position will be, a character standing and another character kneeling in front of the first character

shows that the first character is overpowering the second, or the second is weaker than the first, and so forth. In our jumping example, Staging is being established in the first sequence/keyframe. In general, Staging is the context or situation of the actor.

8.2.2 Anticipation

Anticipation refers to the act of preparation ('prep') before an action. We gave an example of anticipation as the effect of staging for the audience, and this shows the close relationship between the two. In terms of motion, the Anticipation relates more to the need of the agent itself; the 'prep' of an action is referred to as a motion in the opposite direction of the main action. For example: to jump up, an agent (e.g. humanoid) will bend its knees slightly, bowing the body at the waist slightly, and a short pause before leaping upwards. To achieve a higher jump, the agent may bend its knees, and lower its body more, reaching a crouch position, and take a longer pause before leaping up. The range of this 'prep' motion is usually smaller than the main action. This range and speed of this 'prep' motion is the crucial part of anticipation; it prepares the audience for the qualities of the main action (e.g. heavy, light, fast, slow, exciting, lethargic, etc.). It also directly represents the contribution of the energy level, mood, and state of the agent in the action. Moreover, it gives the illusion of 'elasticity' to the agent; this issue will be discussed more in the principle of Stretch and Squash – there is also a strong

relation between Anticipation and Stretch- and-Squash.

In general, the more powerful the main action is (i.e. requires strength, large range of motion, heavy), the larger the 'prep' action is. However, speed is an element that does not always directly relate to the range of the 'prep' action. For example: a quick straight jab punch is fast, but does not have a large range of 'prep' action, in fact, that is the purpose of a jab: is to catch the opponent off guard – the opponent is not expected to anticipate it, but usually the effect of the jab is relatively lighter than the other punches. An uppercut is a powerful punch, thus it has a larger 'prep' than the jab; the 'prep' provides extra space between the fist and the target to accelerate to reach higher speed, and thus delivering higher energy on impact.

In our jumping example, shown in Figure 8.6, Anticipation is in the first three sequences/keyframes. Anticipation is directly related to the 'Preparation' of Phrasing in LMA.

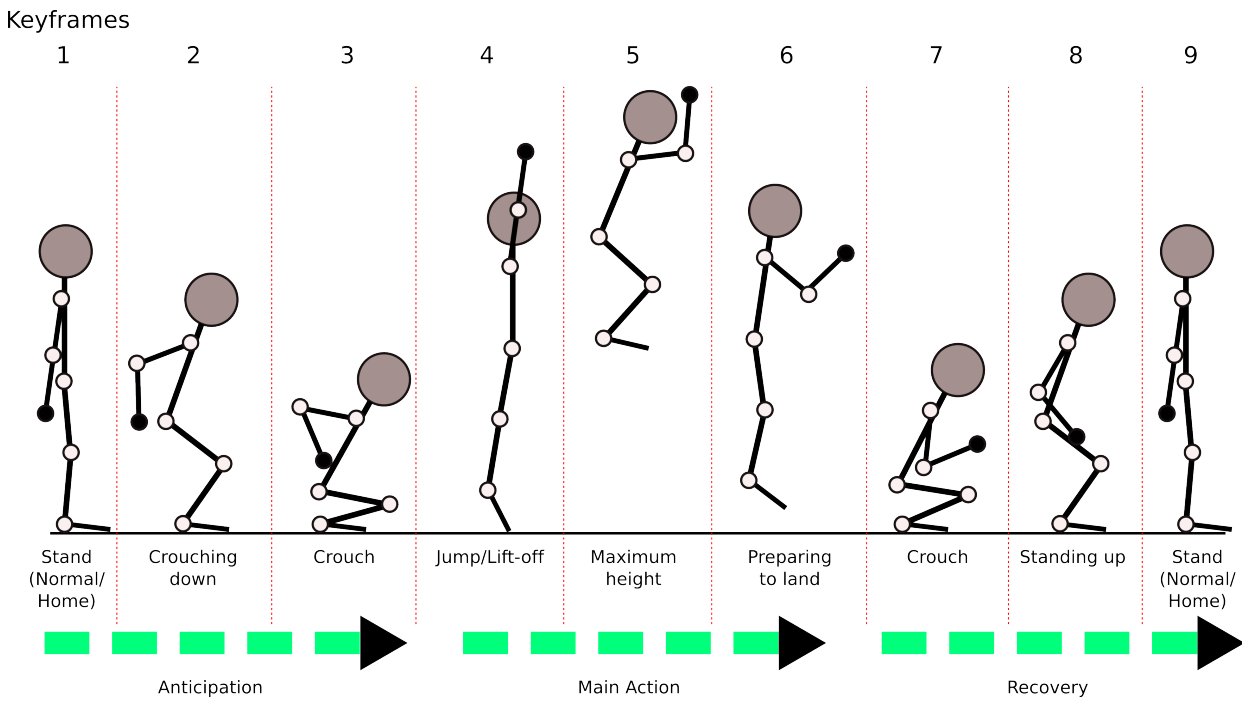


Figure 8.6 Anticipation

8.2.3 Stretch and Squash

Stretch and Squash principle refers to the elastic qualities of organic creatures. This principle is tightly related to the Anticipation, and Follow-through-and-Overlapping principles. Simply put, most organic creatures are 'squishy' in nature, either their physical bodies or their movements. Sure, some creatures such as mollusks and crustaceans are covered in rigid shells and exoskeletons, but their movements always display some kind of preparation action, and/or elasticity – these features are important to create the appeal and the illusion of life in animations (especially cartoons).

Stretch and Squash are effects that usually follow and/or preceed one another: a stretch followed/preceeded by a squash, or a squash followed/preceeded by a stretch, therefore they are always discussed together. The simplest (and most commonly used) example in teaching about Stretch and Squash is the use of a bouncing ball.

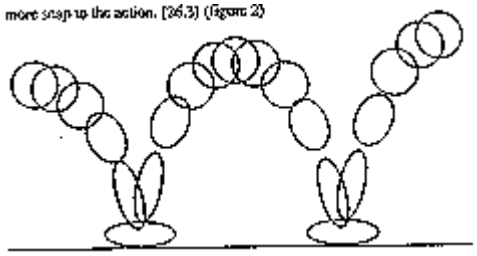


Figure 8.7 Stretch and squash in bouncing ball

The ball deforms as it reaches maximum velocity (Stretch), on impact (Squash/ Anticipation), and on lift-off (Stretch). The Squash on impact gives the effect that the object is made of soft, yet elastic material such as rubber. This is easily observable by watching a basketball being dribbled in slow motion. If the ball does not Squash, it gives the impression the ball is made of a solid material, such as a bowling ball. However, even an object that is solid to the touch, such as a golf ball, still does Squash on impact, only slightly.

Even when the object is rigid, Stretch and Squash qualities still apply, and should be applied to bring life to the object. In our jumping example, seen in

Figure 8.8, the Squashes are seen at sequence/keyframes 3, 5, and 7. Stretches are seen at keyframes 4 and 6.

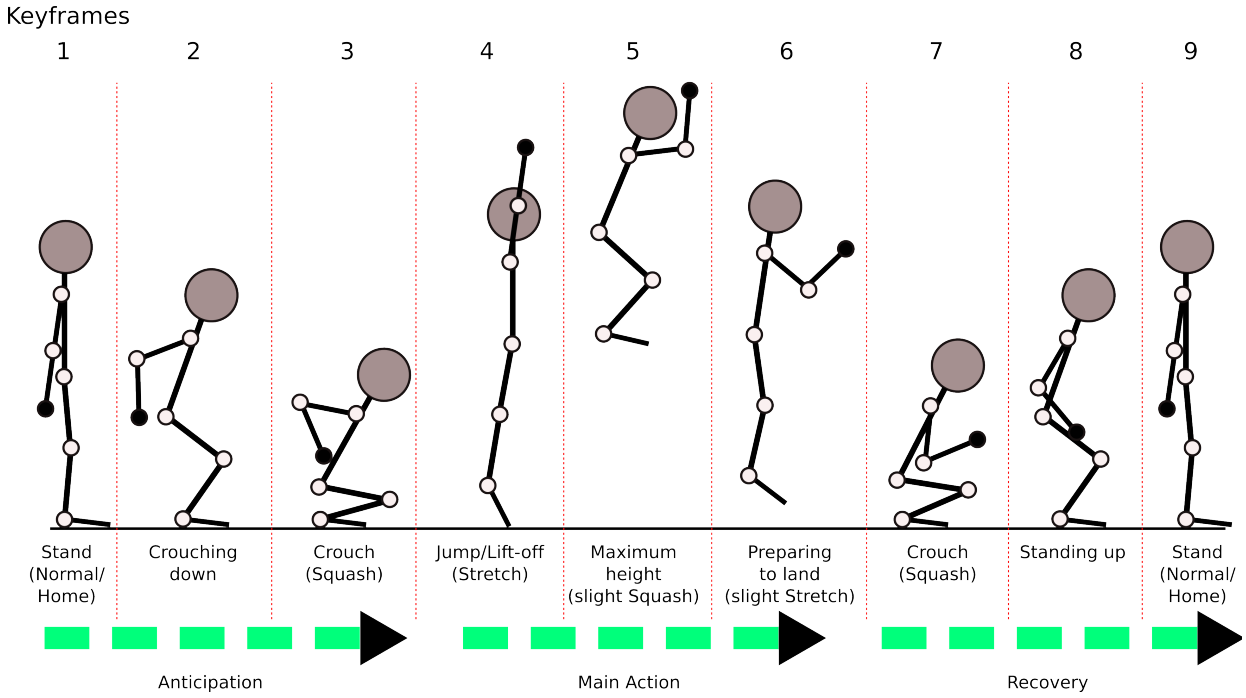


Figure 8.8 Stretch and Squash in jumping example.

8.2.4 Follow Through and Overlap

The Follow Through and Overlap principle refers to the additional animations at the end of an action, which can be considered as the opposite of Anticipation. Follow Through is the "returning to normal, or home position" action. For example, when throwing a baseball, the pitcher's arm does not stop after the ball leaves the hand, instead it continues to travel (because of momentum), until it reaches the resting position. So, follow through can be considered as creating the illusion of the momentum.

Overlap, on the other hand, refers to the transition between actions (or animations), usually the continuation of the Follow Through animation. For example, after the ball leaves the pitcher's hand, and the pitcher's arm and hand are still swinging because of the momentum (i.e. Follow Through), the pitcher would want to retract his arm, and fold it close to his body (torso). The action continues seamlessly from the swinging action to the retracting action, instead of suddenly stopping the swinging action, then retract the arm. While it is possible (a legitimate animation; depends on what the desired effect is), in general, this last animation is not as 'natural' nor appealing as the one with the seamless transition (see Motion Blending).

In our jumping example, seen in Figure 8.9, some follow-throughs are seen on the arm from keyframe 5 through 9. In keyframe 5 and 6, the arm is trying to move to a relaxed position after being stretched in keyframe 4. And in keyframe 7 through 9, the arm moves back to its default position as the figure returns to its initial standing position. In addition, the crouch at keyframe 7 is a Follow Through because the body is absorbing the impact with the ground. Overlap is not shown in this example.

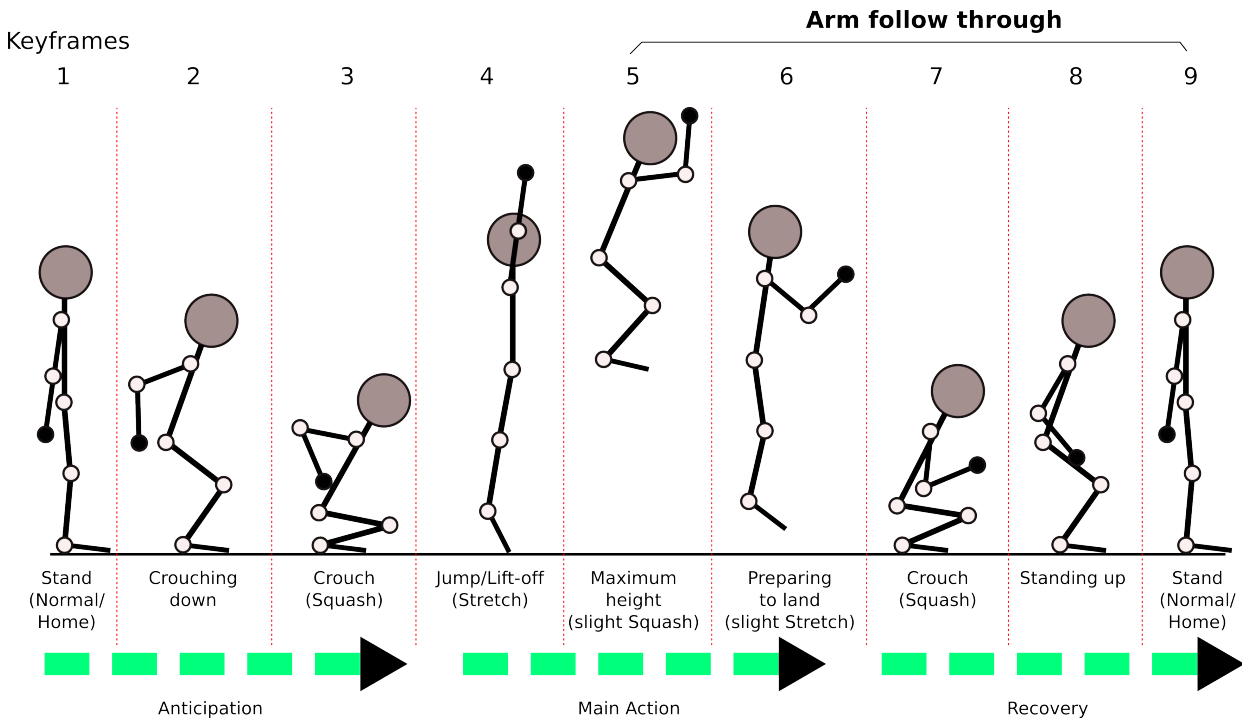


Figure 8.9 Follow through in jumping example.

8.2.5 Timing

Timing refers to the speed of the action; whether it is the anticipation, followthrough, or the main action. Lasseter [Las87] suggests that Timing is crucial, as the same motion done in different speeds will give a completely different message. This is because timing gives the illusion of weight and size of an object, amount of energy exerted or absorbed, and even can convey emotions. In terms of frame-based animations (either 2D or 3D), this is indicated by the number of frames from one keyframe (or pose) to the next keyframe. Since usually animations have a set frame rate (commonly 24 or 30

frames per second), the more frames between keyframes (the frames are aptly called *in-between frames*), the slower the movement seems, and vice versa.

In our jumping example, in terms of robot motion, the speed, acceleration and deceleration of the motion need to be controlled in between the keyframes to create a believable jumping action. We see in figure 8.10 of our jumping example, the movement between keyframe 1 and 2 may be of acceleration, becomes deceleration towards keyframe 3, and comes to a pause for a short period of time before getting into keyframe 4. The Timing between keyframe 3 to keyframe 4 involves some high acceleration for lift-off. From keyframe 4 to keyframe 5 there will be some deceleration as the lift-off energy completely turns into the potential energy at keyframe 5. As the body starts to fall back down, from keyframe 5 to keyframe 6 there is acceleration because of gavity. And so forth.

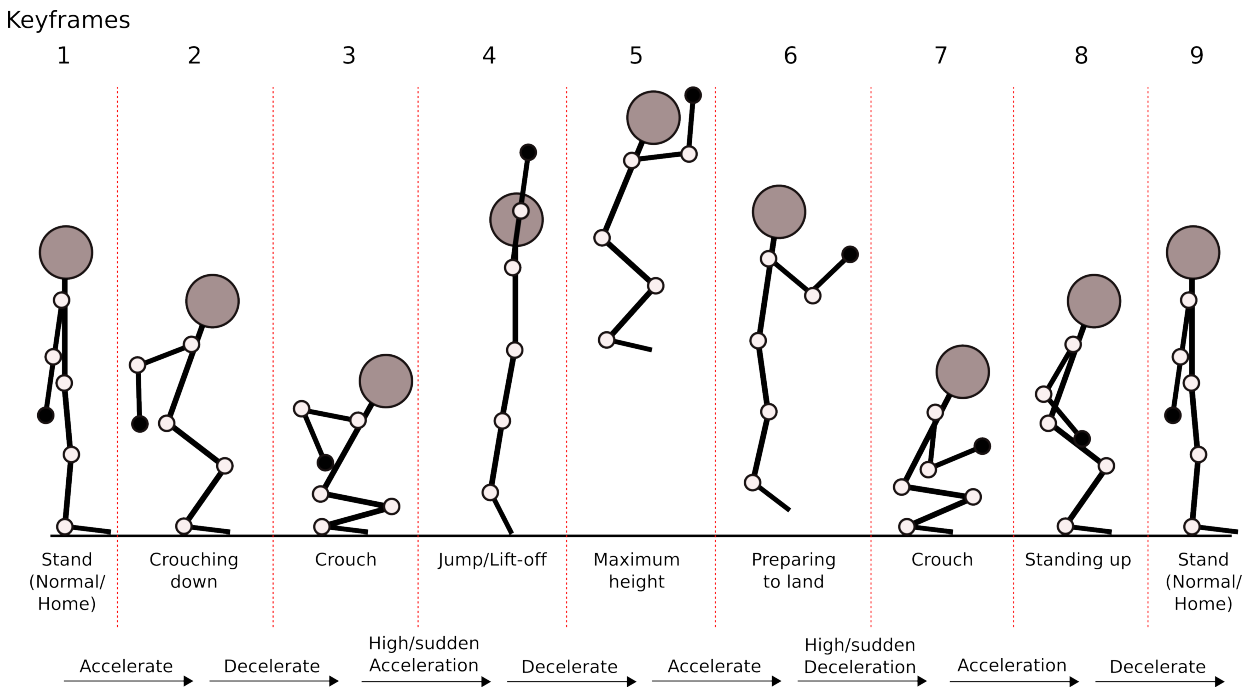


Figure 8.10 Timing of acceleration and deceleration

8.2.6 Slow-in/Slow-out

Slow-in and Slow-Out refers to the deceleration and acceleration of the movement, respectively, and is closely related to the Timing principle. More specifically, the deceleration and acceleration when ending one movement or pose, and entering a new movement or pose, respectively. For example, the animation of a bouncing ball involves the ball decelerates as it reaches the top of the bounce (as the vertical velocity approaches zero), and after the ball reaches the top (vertical velocity = 0), the ball starts to accelerate as it goes

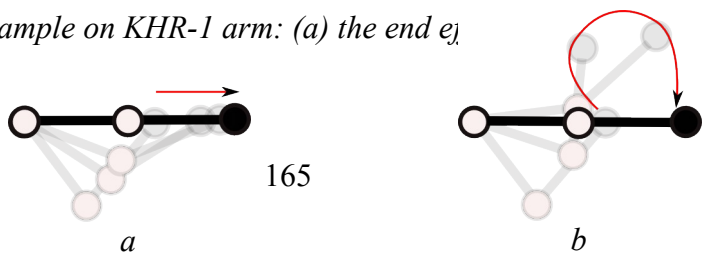
down. This creates a more natural action for the ball as it travels in an arc. We already touch upon Slow-in/Slow-out in our jumping example in the discussion about Timing in section 8.2.5.

8.2.7 Arcs

Arcs refers to the curving paths of the movement, which makes the movement look 'biological' rather than 'mechanical.' There may be cases where the movement is intended to be linear, in a straight line, or when the transition between movements is unavoidably (or intentionally) jerky, but in general it is preferred to have all the movements follow some arc, however subtle it is.

As an example of arcs on KHR-1, we will depart from the jumping example, and show two examples of arm movement. The first example (figure 8.11a) is the movement of the end effector (hand) in a straight line. The second example (figure 8.11 b) shows that the arm arrives at the same final position as the first example, but the hand moves in an arc. The former gives the impression of a stabbing or mechanical movement, while the latter is like a reaching or waving movement as in a dance.

Figure 8.11 Arcs example on KHR-1 arm: (a) the end ej *is moving*



a *b*

8.2.8 Exaggeration

The Exaggeration principle refers to creating an emphasis to, or accentuate the actions, or personality of a character. Exaggeration essentially adds a "more" aspect to the action or personality traits, for example: scared to terrified, sad to sobbing, surprised to shocked, strong punch to powerful punch, and other intensified actions or emotions. In 2D or 3D animation, Exaggeration does not imply to simply deform the object to the extremes (with Stretch and Squash) that would make the object appear "cartoon-like", or comical, instead it is used carefully to make the whole scene seem natural, yet interesting and understandable to the audience.

In humanoid robots, as we are dealing with rigid bodies, we can only exhibit Exaggeration through changing the range of motion. For example, in our jumping example, in keyframe 3 we can exaggerate the pose by bending the upper body forward further, and pulling and folding the arm further, as seen in Figure 8.12. The exaggerated pose also gives the indication that the jump is going to be more powerful.

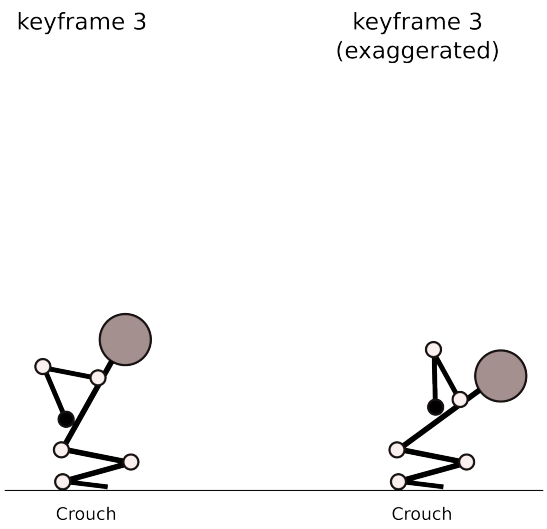


Figure 8.12 Exaggeration on keyframe 3 (crouching)

8.2.9 Secondary Action

Secondary Action refers to the actions that support the naturalness, or realism of the main action. For example, when walking, the clothes and hair of the actor moves along with every step he/she takes. Secondary Action can also be thought of a passive action, because their movements depend on nature, or are side effects of the main action. While Secondary Action is important for animators in 2D and 3D animation, in the animation of physical figure like KHR-1, this principle effectively is not of concern for the motion synthesis system.

8.2.10 Appeal

Appeal refers to the interestingness of the action, the whole scene, or the

characters. Appeal emphasizes on the importance of *artistic visual design* in animation. As in graphic design, all aspects of a product must contribute to the design (whether it is a poster, object, or web sites), and thus within context. The same goes in animation (naturally, since it is a form of art); a character's visual/physical design shows the audience the first clues of what the personality of the character is. A big, muscular, and bulky character gives the impression to the audience that this character is strong, and maybe prefers to solve his problems with brute force. It makes the character even more interesting when he actually has traits that are opposite to his physical appearance such a very gentle personality, and scared to things significantly smaller than him like mice. All these creates an Appeal-ing character.

Appeal extends not to just the visual design of the physical appearance of a character, but also to his/her actions. The proper use of Anticipation, Slow-in/ Slow-out, Exaggeration, Secondary Action, Stretch and Squash, Arcs, and Timing in a movement to show the character's emotions, intentions, and effort, creates an interesting action, and thus makes it Appeal-ing. Even posing of a character when he/she is standing and not doing anything else, require certain considerations to make he/she Appeal-ing. The robot character Wall-e in the movie Wall-e, was shown in many promotional advertisements as just 'standing' there, but with his head tilted to the side, it expresses Wall-e's personality as being curious, child-like, and innocent, and thus intrigues the

audience: *a curious, child-like, and innocent robot?? This I got to see.*

The issue of Appeal is addresed a little bit by the Uncanny Valley theory from Mori [Reference to Mori]. The Uncanny Valley theory tries to identify the relationship between the human-likeness of an object and how humans perceive such objects. Mori postulated that humans would accept objects with human or animal likeness to some degree. But, as the features of the objects become too human, but not exactly human, humans would perceive the object as eerie, creepy, and uncanny, as if zombie-like. However, if the object has enough features to very, very closely resembles a living being or human, humans would easily 'accept' the object, and overcome the uncanny feeling towards it. Currently, there is no agreement what set of features is required to pass this uncanny valley, and we cannot imagine what kind of 'acceptance' it will be; will the object be treated as a sentient being, thus raising ethical and moral issues about attititudes towards the object, or the object will be accepted as just another common object in everyday human life? Again, this discussion is beyond the scope of this thesis and will not be addressed further.

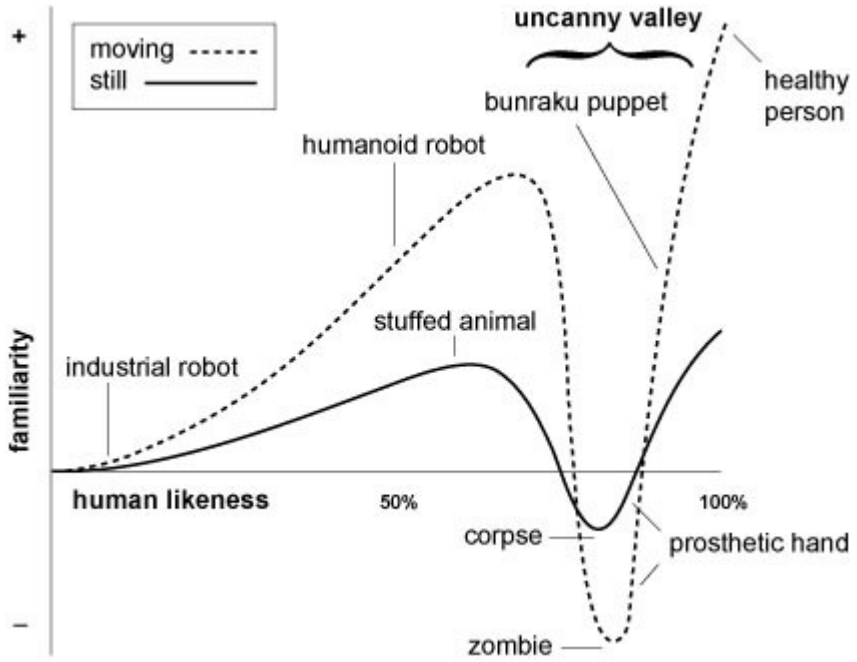
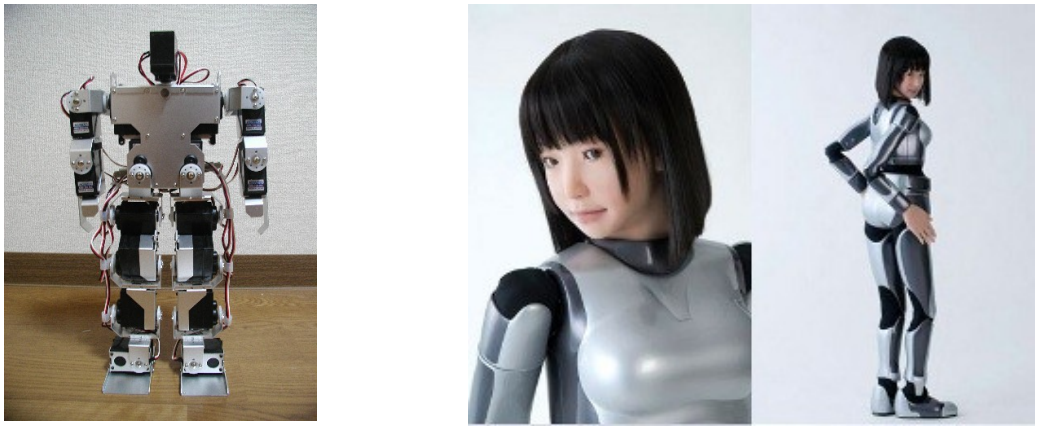


Figure 8.13 Mori's Uncanny Valley. (Image source: [Wiki:UV])

Nevertheless, this issue highlights the Appeal aspect of our robot. We found that people are more accepting when the robot looks and feel mechanical, as opposed to human-like such as android. When we show the KHR-1 to people, their acceptance is very good. We define the 'acceptance' as the relative attractiveness of the robot as perceived by the audience, and the eagerness of the audience to interact and play with the robot. Both adults and children seem to be very interested in the robot, and often excited to see the robot do very simple actions such as push-ups. Some of the audience even commented that the KHR-1 looks "cute."



a *b*

Figure 8.14 Appeals of (a) KHR-1, (b) HRP-4C

We also observed the state-of-the-art androids and the reactions of the people when seeing the android in action. In comparison to the KHR-1, the HRP-4C is a full human-sized android with human-like face, and limbs. In particular, the HRP-4C has a complete woman face, made of skin-like material, capable of some facial expressions, and even lip-sync for speech. Supposedly, the face was modeled as an "average japanese woman." In addition, HRP-4C is capable of unassisted bipedal walking, and is controlled wirelessly. The android has made headlines in recent news as being the first android to be used as a model in a fashion show. Scientists touted the HRP-4C as a significant advancement in humanoid robotics, following Honda's Asimo, but the public's opinion are mostly negative. We observed the comments from the audience in public websites that show the HRP-4C in action, such as YouTube.com and blogs such as Engadget.com. Most of the comments of general audience after

seeing the HRP-4C in action are either “creepy,” “I don't want that *thing*,” “scary,” and in general, the opposite to the kind of attitudes towards the KHR-1.

8.2.11 Straight Ahead and Pose-to-pose

Straight Ahead and Pose-to-pose refer to the two main approaches of creating an animation. Straight Ahead refers to creating animations by progressively making one frame after another, and is the common approach in creating clay animations. Pose-to-pose refers to creating animation by making *keyframes*, that is, certain poses in an action that in a way, minimally describes the action. For example, a bouncing ball animation has its main poses of when the ball hits the ground, and when the ball is at the top of the bounce. After these 'main poses' are created, then an animator creates the frames between those poses (*in-between frames*), according to the timing and effects of the action that the animator desires. In 2D animations, this involves stacking two keyframes on top of each other over on a lightbox, and then add another piece of paper on top of the stack, and draw the pose between the two keyframes. In 3D animations, this is done by means of interpolations.

In a sense, the animation of a humanoid robot can follow both Straight Ahead and Pose-to-pose principle. The equivalent of Straight Ahead animation would be Forward Kinematics, where the joint angles are fed in a certain sequence,

while Pose-to-pose may be associated with the Inverse Kinematics approach to an extent. Our approach in this thesis is considered more towards the Pose-to-pose approach. This is because we are working with pre-programmed motion data that consists of joint angles that are treated as keyframes. The final motion executed by the robot is the result of interpolating this pre-programmed motion data, which is analogous to creating in-between frames.

8.3 LMA and Disney Animation Principles for Robot Motion Synthesis

We are using LMA as the high-level *language* to command movements and gestures to the robot, while the Animation Principles are the *knowledge* of how to move for the robot. For example, if we tell the robot to touch, let's say a person's hand, gently, according to LMA, “gentle” can be described as Direct, Light, and Sustained in terms of Effort. Using the Animation Principles, Direct can determine the direction of the Arc, Light and Sustained determine the Timing, amount of Anticipation, perhaps some Exaggeration, and the Slow-in/Slow-out of the movement.

We choose LMA for the language because it is the most developed, and the standard commonly used system in movement analysis in fields such as arts, dance, physiotherapy, sports, and so on. We also choose the Disney Animation Principles as it is the dominant guidelines used in the animation industry for animators to synthesize motions. For the sake of having a natural Human-Robot Interaction, we need to provide robots with a way to synthesize motions as a reaction to human interaction

with the robots, or expressing themselves without requiring a human to specify the motions down to the details such as joint angles.

CHAPTER 9 – CREATING EXPRESSIVE MOTIONS

In this chapter we will describe the methods we used in our software to process a motion of the KHR-1, and how the animation theories discussed in Chapter 8 are being applied.

9.1 Motion Processing

We already introduced an overall view of our whole system in Chapter 3 (Figure 3.1), and some details about the Perception, Cognition, and Response blocks in Chapter 4, 5, and 6, respectively. Figure 9.1. shows the summary of the type of processing that happens in each of the Perception, Cognition, and Response block, and how-and-what information flows in the system.

Our approach to process the motions of our KHR-1 robot is done by applying two main concepts: the paradigm of motions as a set of signals, and the motion theories of DAP and LMA. Effectively, we must simultaneously select a set of signal processing methods to realize the ideas presented in DAP and LMA, and represent the concepts in DAP and LMA using those signal processing methods. Therefore, in the following discussion we will introduce the signal processing methods we use, and afterwards, we explain how we are realizing the DAP and LMA concepts using those methods to an extent. Lastly, we explain how the information from preception and cognition come

into play with the motion processing.

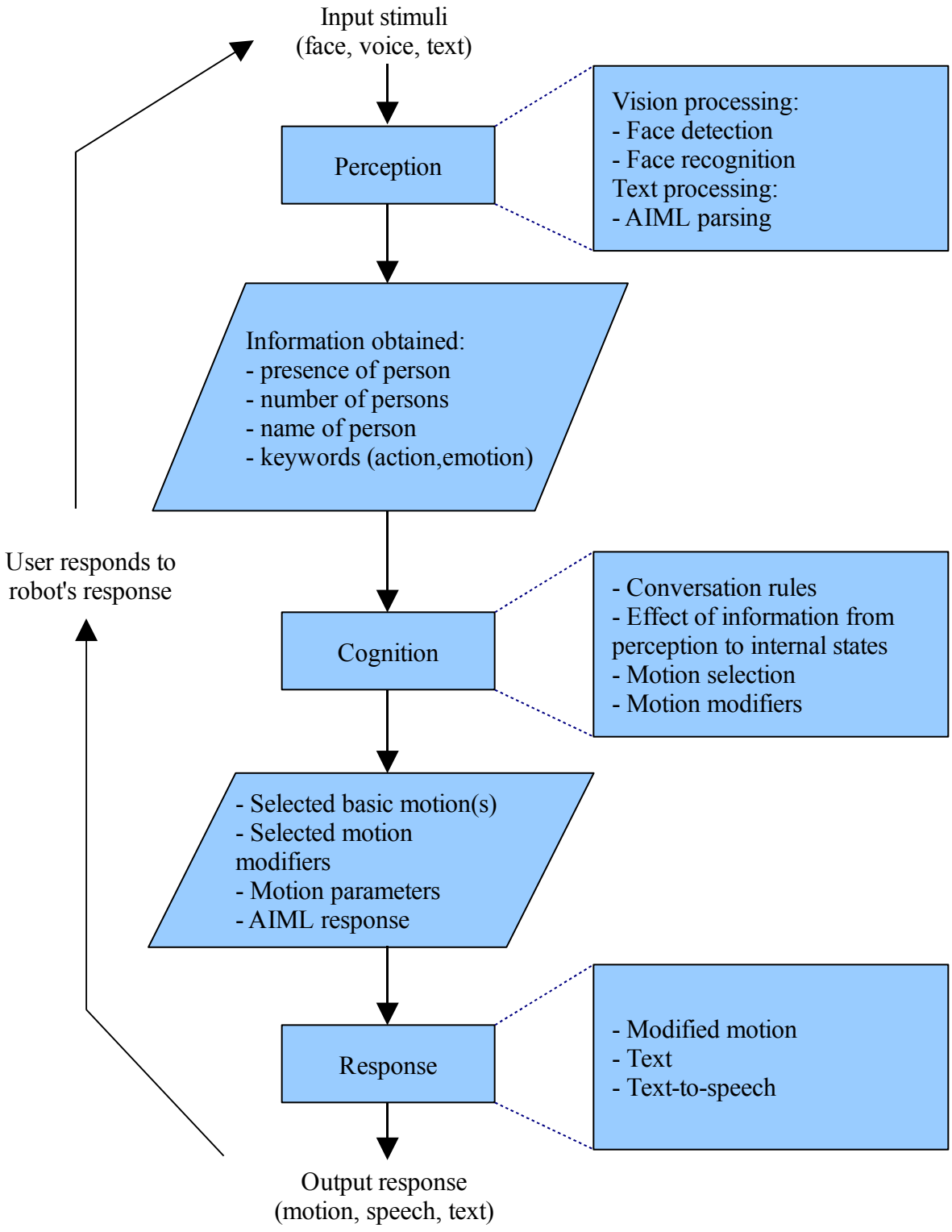


Figure 9.1. Information flow and processes in the system.

9.1.1 Signal Processing Methods

There are four properties in a motion: the range of motion, the path of the motion, the speed of the motion, and the acceleration/deceleration in the motion. Those properties are to be controlled in order to create emotional expressions in the motion outlined in DAP and LMA.

9.1.1.1 Kochanek-Bartels Interpolation

The Kochanek-Bartels interpolation method is based on the cubic Hermite spline, and provides three parameters to control the tangents of the spline: *tension, bias,* and *continutiy.* The effects of these parameters on a spline can be seen on figure 9.2.

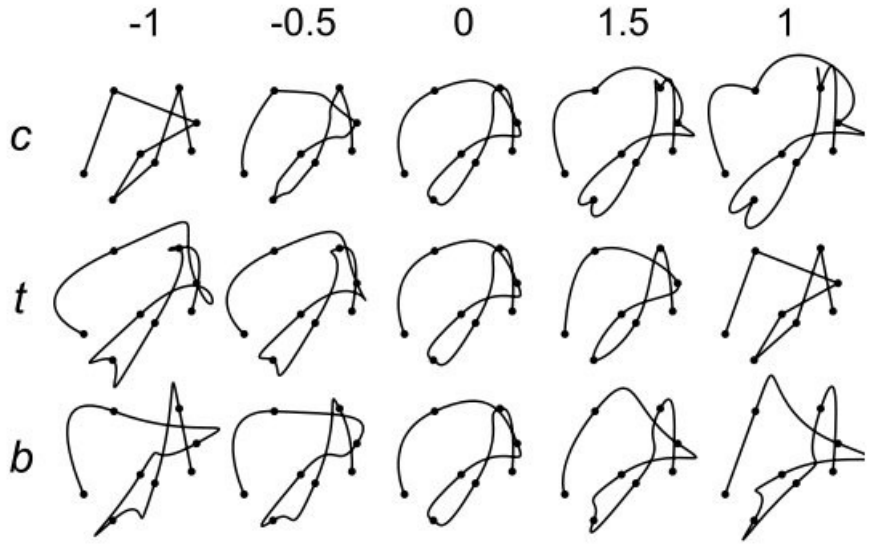


Figure 9.2 Effects of continuity (c), tension (t), and bias (b) parameters on Kochanek-Bartels splines (Image source: [Wiki:KBS]).

The tangents T1 (incoming tangent) and T2 (outgoing tangent) can be calculated as follows [Wiki:KBS]:

$$T1 = \frac{(1-t)*(1-c)*(1+b)}{2}*(P_{start}-P_{start-1})+\frac{(1-t)*(1+c)*(1-b)}{2}*(P_{end}-P_{start})$$

$$T2 = \frac{(1-t)*(1+c)*(1+b)}{2}*(P_{end}-P_{start})+\frac{(1-t)*(1-c)*(1-b)}{2}*(P_{end+1}-P_{end})$$

Where:

- t = tension
- c = continuity
- b = bias
- P_{start}, P_{end} = starting and end points of the segment which to be interpolated
- $P_{start-1}$ = the point before P_{start}
- P_{end+1} = the point after P_{end}

To calculate the interpolated point, P_{start}, P_{end}, $T1$, and $T2$ are multiplied with the four Hermite basis functions and then added together. The Hermite basis functions are:

$$\begin{aligned} h1(s) &= 2s^3-3s^2+1 \\ h2(s) &= -2s^3+3s^2 \\ h3(s) &= s^3-2s^2+s \\ h4(s) &= s^3-s^2 \end{aligned} \tag{9.14}$$

Where s is the value [0,1] specifying the interpolation point between P_{start} and P_{end}. For example, $s = 0.5$ would be the middle point between P_{start} and P_{end}. Then, the

interpolated point P is calculated as follows:

$$P = h1 * P_{start} + h2 * P_{end} + h3 * T1 + h4 * T2$$

Or in matrix form:

$$P = s * k * h$$

where:

$$s = \begin{bmatrix} s^3 \\ s^2 \\ s \\ 1 \end{bmatrix} \quad k = \begin{bmatrix} y_1 \\ y_2 \\ T1 \\ T2 \end{bmatrix} \text{ and } h = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Using the Kochanek-Bartels interpolation, we have some freedom to control the tangents of the curve. This freedom is useful for us to control the acceleration and deceleration of the motion, as they are related to the curvature of the signal. Consider figure 9.3 and figure 9.4. The intuition presented in figure 9.3 shows that a linear signal indicates constant speed (zero acceleration/deceleration). In figure 9.4, at right before time 4, we see that the position values 'eases-in' to 5, and then slowly 'eases-out' from position 5 to the next position value. This 'ease-in'/'ease-out' indicates some acceleration and deceleration. This intuition is true for our system as we are working with time-position data sets.

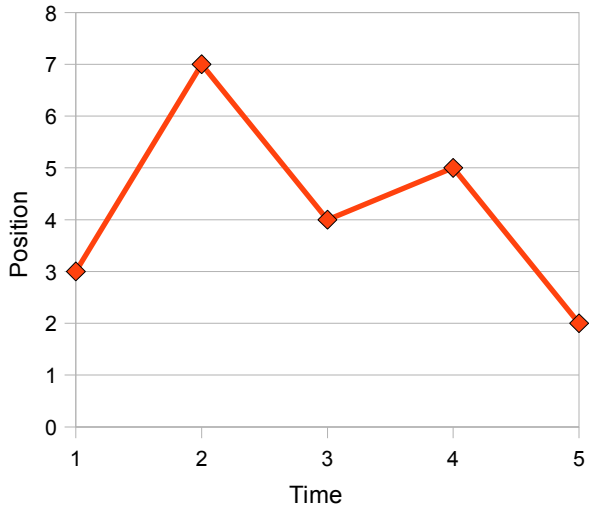


Figure 9.3 Linear motion signal.

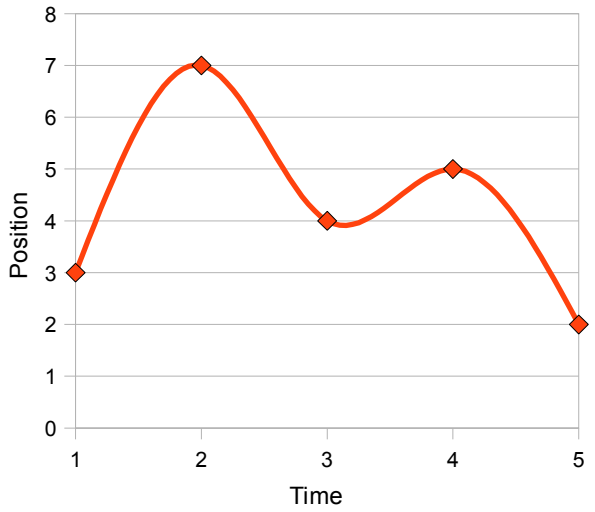


Figure 9.4 Non-linear motion signal.

Currently, our system has a significant side effect from interpolating a motion signal. Namely, interpolating the motion signal *elongates the duration* of the motion, as we are inserting several new data points in between two points from the original motion data set, while each element of the data set always correspond to one time step. Figure 9.5 shows the data set from figure 9.3 interpolated by inserting seven points in

between each two data points; expanding the length of the data set from 5 to 33. Thus, to control the duration of the motion, we use a simple (Re)Sampling method.

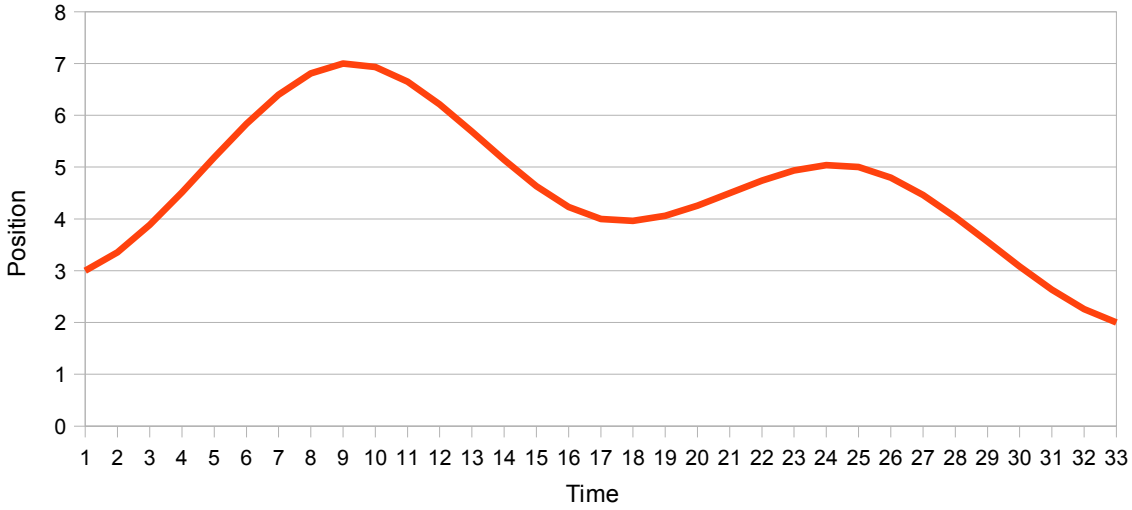


Figure 9.5 Interpolated (elongated) data set (continuity=0, tension=0, bias=0).

9.1.1.2 (Re)Sampling

As mentioned above, the (Re)Sampling method is used to compensate for the side effect of the interpolation. The (Re)Sampling method is basically similar to normal analog-to-digital sampling method, where a continuous-time signal is sampled at a certain frequency to produce discrete-time signal. In our case, we sample the interpolated signal with a rate of *r*, where *r* indicates taking one data point for every *r* points in the interpolated signal (data set). Thus, *r* is not necessarily related to time per se, but to the number of elements in the data set. If $r = 1$, then no sampling occurs, as it means taking *every element* in the data set. Figure 9.6 shows the interpolated data set from figure 9.5 resampled with a $r = 3$; the number of elements in the data set is reduced from 33 to 11, effectively shortening the duration of the motion by one-third

(assuming this is a motion data set).

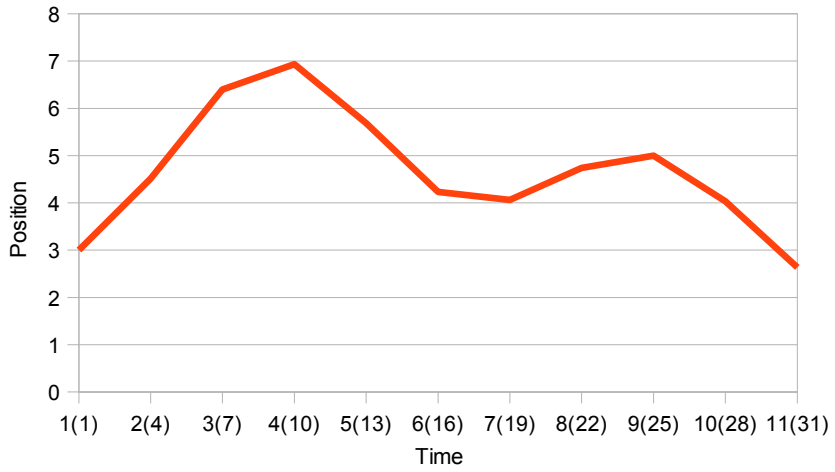


Figure 9.6 Resampled data set with r=*3. Numbers in parantheses indicate the index numbers of the sampled elements from the interpolated data set.*

9.1.1.3 Multiresolution Filtering

Multiresolution filtering is a signal processing method that filters a signal at different resolution seccessively, resulting with a pyramid of bandpass and lowpass frequency components ([BW95], [HBT99]. The original signal can be reconstructed by summing the bandpass components and the DC component (the last lowpass component). Each bandpass components can be multiplied with a gain value before the signal is reconstructed. Adjusting the gains of the high frequency component can reduce or increase the details, and noise in the motion, while the gains of the low frequency component will alter the overall shape of the signal.

For our application to edit motions, multiresolution filtering allows an effective way to

manipulate the range of motion on different levels of details (high frequency vs. low frequency) in the motion, while preserving the motion itself, up to a point. For example, reducing the gain of the high frequency component can 'smooth' out the motion, reducing noise in the motion signal. Increasing the gain of the middle frequency components can exaggerate the motion by increasing the range of motion. Conversely, reducing the gains of the middle or low frequency components reduces the range of motion, creating an illusion of 'heavy' or 'tired' motion. However, if the gains are reduced too much, the effect may instead 'inverts' the shape of the motion signal, depending on which frequncy component was adjusted.

In our system, we implemented multiresolution filtering based on the implementation by [BW95]. The signal is filtered with a normalized 5-points kernel $w = [c\ b\ a\ b\ c]$, where: $a + 2b + 2c = 1$. The kernel values are selected to be as follows: $a=3/8$, $b=1/4$, and $c=1/16$. The signal is convolved with this kernel to give the lowpass data. A lowpass pyramid, consisting of layers of lowpass data, is created by reducing the resolution of the signal by a factor of 2 in each iteration, until the DC value is found. The number of layers of lowpass data fb for a signal with length m is:

$$2^{fb} \leqslant m \leqslant 2^{fb+1}$$

The multiresolution algorithm can be summarized as follows:

1. Convolve the original signal G_0 with the kernel w_1 to generate the array of lowpass data G_1, or by:

$$G_{i+1} = G_i \times w_{i+1}$$

2. Expand the kernel w_1 to generate w_2 by inserting zeros in between the kernel values, such that:

   w_1 = [*c b a b c*],

   w_2 = [c 0 b 0 a 0 b 0 c],

   w_3 = [c 0 0 0 b 0 0 0 a 0 0 0 b 0 0 0 c], and so forth.

   This is equivalent to reducing the resolution of the signal by a factor of 2.

3. Repeat to convolve the signal with the expanded kernels to generate the lowpass pyramid $G_1 - G_{fb}$, where G_{fb} is the DC value.

4. Create the bandpass pyramid, consisted of frequency bands $L_0 - L_k$ by:

   $L_k = G_k - G_{k+1}$, where: $0 \leqslant k < fb$

5. Adjust the gains of the frequency bands as necessary,

6. Reconstruct the motion signal, by:

$$G_0 = G_{fb} + \sum_{k=0}^{fb-1} L_k$$

, or $G_0 = G_{fb} + \sum_{k=0}^{fb-1} (a_k L_k)$, where; a_k = gain of frequency band k

9.2 Applying Concepts of Expressive Motion from Disney Animation Principles (DAP) and Laban Movement Analysis (LMA)

There are four properties in a motion: the range of motion, the path of the motion, the speed of the motion, and the acceleration/deceleration in the motion. Those properties are to be controlled in order to create emotional expressions in a functional/basic motion. Thus, we take cues from DAP and LMA for the relationship between those motion properties, and emotional expressions in a motion. Specifically, we focus on a set of important motion qualities consisted of: motion dynamics, and motion transitions. To realize these motion qualities, we describe how we use our motion processing methods using the animation terminologies: Effort (from LMA), Shape (LMA), Phrasing (LMA), Anticipation (from DAP), Stretch and Squash (DAP), Follow through and Overlap (DAP), and Slow-in/Slow-out (DAP). Our implementation of the parameter values in relation to the relization of the motion qualities will be explained in section 9.1.2.

9.2.1 Functional Motion (Basic Motion)

Functional motion refers to motions which are meant to achieve a particular task. For example: reaching for a pen on the table. In our system, we consider our set of pre-programmed motions as functional motions, or basic motions. Thus, if the motions are executed with out any processing through interpolation, (re)sampling, or multiresolution filtering, the motions will look stiff, dead, or robotic.

9.2..2 Motion Dynamics

The DAP and LMA theories taught us that human motions are often simultaneously by initiative of the human (active motion), and as a response to forces acting on the body (passive motion), such as gravity. Motions that are showing exertion or absorbtion of force (or motion dynamics) are often characterized by the acceleration/deceleration, and the speed in the motion. For example: when a person is carrying a very heavy object, the person is fighting the resistance from the down force of the object because of gravity, and the inertia of the object. The movement of the person will be slow, and has very low acceleration. Thus, we can simulate 'exertion of force because of weight' in a motion by slowing the motion down, and with very slow acceleration. But when the exertion of force is not hampered by weight, the force is shown by a fast, and high acceleration movement, such as punching. We will show that we can control the speed and acceleration of a motion by using sampling and interpolation methods, respectively. In case of simulating heaviness such that it limits the range of motion, we need to use multiresolution filtering, with which we can manipulate the amplitudes of the motion signal.

The behavior of force exertion in the motion can be verified using *dynamics simulation*. In dynamics simulation, a model of the musculoskeletal system of a human body is used. Using this model, a movement can be simulated by either giving excitation to the relevant muscles (*forward dynamics*), or by applying force to a particular limb (*inverse dynamics*), such as the hand. In general, the amount of

excitation to the muscle indicates the amount of force exerted through the limb. Thus, ideally the robot needs to be able to sense how much resistance it has to fight in order to complete the motion. This force exertion behavior is related to the Weight Effort in LMA, and Anticipation in DAP. Dynamics simulation is not yet implemented in our current system, but definitely be the next logical component to add to the system to enhance the motion dynamics.

We can also show the Time Effort of LMA. According to LMA, the Time element of Effort is related to the sense 'urgency' vs 'indulgence' of time in the movement. The urgent movement can be characterized by its acceleration/deceleration, rather than by its speed. To exhibit the Time Effort, we control the acceleration of the movement by the *tension* parameter in the Kochanek-Bartels interpolation method. As shown in figure 9.2, when the tension parameter value is low (-1), the curve appears smooth, without sharp corners. Conversely, when the tension parameter is high (1), the edges of the curve is linear, and the curve has many sharp corners. These edges and corners in a motion signal give the intuition of the acceleration and deceleration in the motion:

- Signals that are interpolated with low tension value has gradual acceleration or deceleration throughout the motion – *time-indulgent* quality.
- While signals that are interpolated with high tension value has almost no acceleration or deceleration throughout the motion, except for a very brief period at the beginning, end, or changes of direction in the motion – *time-*

urgent quality.

In addition, *time-urgent* quality will often be coupled with (re)sampling method to shorten the duration of the motion (hasten), thus emphasizing the illusion of 'urgency' the motion. The Time Effort of LMA is related to the Time principle of DAP. However, in DAP, Time is explained as also related to motion dynamics, for example: when a person is being hit by a strong force on the head, the head will rebound faster after impact than when the head is only being pushed with a slight force.

9.2..3 Motion Transitions

Motion transition, or transition between one motion the next, can be explained using the Flow Effort component from the LMA framework. The Flow Effort of LMA is understood as the progression of one movement to the next. Therefore, we exhibit a 'bound' Flow by abrupt transition between movements, and 'free' Flow by smooth transition between movements. In DAP, Flow is explained as Overlap in Follow-through and Overlap principle, In our system, we show a 'free' Flow or good Overlap by first concatenating two (or more) motion signals to be executed, and then performing interpolation on the concatenated motion signal. Thus, there is a continuous transition between one movement to the next. Conversely, for 'bound' Flow, we let the system execute the subsequent motions only after the previous motion is done, thus paying no regards to the continuity of the transition between movements.

9.3 Parameters and Parameter Values for Expressive Motion

We used three methods to modify a motion signal to realize expressive motion: the Kochanek-Bartels interpolation, (Re)Sampling, and Multiresolution Filtering. Modification of a motion signal is done by determining the parameter values for these three methods:

- For Kochanek-Bartels interpolation:
  - Parameters: *Continuity*, *Tension, Bias*
  - Parameter values: [-5, +5] for all three parameters
- For (Re)Sampling:
  - Parameter: *Rate of sampling*
  - Parameter values: [1, 10] (Rate = 1, means no sampling)
- For Multiresolution Filtering:
  - Parameters: *n* number of frequency band gains
  - Parameter values: [-5, +5] for all gains

9.3.1 Parameters for Kochanek-Bartels interpolation

We already discussed a little bit about the mechanisms for the Kochanek-Bartels interpolation in section 9.1.1.1. Here, we would like to explain how we use the *Continuity*, *Tension*, and *Bias* parameters as part of the motion processing.

We use the *Continuity* parameter when we want to modify the motion transitions within a motion, or between motions, such as described in Section 9.1.1.3. The value

for *Continuity* is reduced when the motion transitions are desired to be less continuous and more linear. The *Continuity* value should be increased, otherwise. The *Tension* parameter is used similarly as the Continuity parameter

9.3.2 Parameters for (Re)Sampling

(Re)Sampling only has one parameter: the rate of sampling. The range of the rate of sample is $r = [1, 10]$. When $r = 1$, no resampling is done, since it means taking *every* data point of the sampled signal. When $r = 8$, if the resampled signal is the interpolated signal, then the signal is returned to its basic, unprocessed form. This is because the interpolation method inserts seven additional points for each two points in the basic motion signal. By using $r = 8$, the signal is sampled by taking one data point for every eight data points in the signal, which means, taking the original data points of the basic motion signal. We set a maximum rate of 10 for experimental purposes. Intuitively, when the motion signal is resampled with a rate greater than the number of the interpolation points plus one (in our case, 7+1 = 8), and assuming the number of steps in the motion is greater than ten, the resulting motion becomes unpredictable. This is because the data points sampled from the signal now does not necessarily consists of the positions that define the original motion. The first point would be the same point as the original or interpolated motion data. The second resampled point would be the tenth data point in the interpolated motion data, which is two data points away after the second data point of the original motion data. The third resampled point would be the thirtieth data point, which is fourteen points away from the third data point of the original motion data. We immediately see that the resampling causes

the sampled data to drift further and further away from the data points that define the characteristics of the original motion. Although we do not endorse using resampling rate greater than 8 in order to rertain the characteristic of the original motion, we do believe using $r > 8$ would create an interesting response for the robot under extraordinary situations, such as extreme anger towards the user.

9.3.3 Parameters for Multiresolution Filtering

The parameters for multiresolution filtering is slightly trickier to set automatically. This is because the number of frequency bands, and thus their gains, varies depending on the length of the motion. Recall from Section 9.1.1.3 that the number of frequency bands, *fb*, that can be extracted from a signal depends on the length of the signal *m*, such that: $2^{fb} \leqslant m < 2^{fb+1}$. Therefore, we only modify the gains of the frequency bands by using three categories: high, medium, and low. We do this categorization of *fb* number of frequency bands by doing a modulo 3 on the number of indices of the generated frequency bands, to determine the first index, which is for the low gain, and number of steps to the next indices for the medium, and high gains. We do this by the following:

```
def steps(fb):
    remainder = fb modulo 3
    f = integer of fb/3
    if s > 1:
        return f-1, f+1 #starting index, steps, respectively
    else:
        return f-1, f  #starting index, steps, respectively
```

For example: if there are four frequency bands (*fb*=4), the function `steps(4)` would return (0, 1), which means the starting index is 0, and the number of steps to the next indices for the medium and high gains is 1. Therefore, the gain the low gain would be the gain of the first frequency band (starting index=0), the medium gain would be the gain with index=1, and the high gain will be the gain with index=2, leaving the fourth gain (index=3) unused. As another example: steps(7) returns (1, 2), which means the index for the low gain is 1, the index for medium gain is 3, and the index for high gain is 5. We feel that by defining three categories for arbitrary number of frequency bands is a fair compromise between giving us enough control over the gains of the low, medium, and high frequency bands, while sacrificing our control over the other gains of the frequency bands.

CHAPTER 10 - EXPERIMENTS

In this chapter, we present our experiments with our system to evaluate our hypothesis:

> *Human-like emotion, personality, and physical states can be expressed by a humanoid robot through its motions, encoded as a set of signal transformations.*

We test our system with the following approach:

- For all scenarios we assume the following:
  - The system is started: the program is loaded, and the KHR-1 is turned on.
  - The speed parameter for RCB-1 is always set to 6.
  - User is in the view of the camera (Webcam) (optional).
- The test is done in two Scenarios:
  - Scenario 1 – Show Emotions (Joy, Fear, Sadness, Anger) on a Waving-Arm Motion using Reserved Commands.
  - Scenario 2 – Public Evaluation.
  - Scenario 3 – 'Driving' the Emotion of the Robot with the Emotional-Personality model.
- The evaluation is done empirically:

- Relative qualities such as 'smoother/rougher transitions', 'faster/slower acceleration/deceleration', 'faster/slower duration', 'bigger/smaller range of motion' are compared to the *baseline motion.* We define the *baseline motion* as*:* the basic motion that is interpolated with default interpolation parameter values (*continuity=0, tension=0, bias=0*), without additional interpolation, sampling, or gain adjustments.
- Whether or not the observer able to identify any resemblence between the modified motion and the original basic motion. For example: will the observer still be able to identify the motion as a Pushup motion after the modifications?

10.1 Scenario 1: Show Emotions (Joy, Fear, Sadness, Anger) on an Waving-Arm Motion using Reserved Commands

10.1.1 Scenario 1 – Goals

The goals of the experiment in Scenario 1 are as follows:

1) To find out if it is *possible* to use the signal processing methods outlined in Chapter 9 in such a way that a simple motion can be modified using those methods into a motion with different emotional expressions. In other words, to investigate if we can represent emotions using a set of values for the parameters of the signal processing methods,
2) To observe if a motion (e.g. Waving arm) will be modified when the robot is in

an emotional state, and

3) To see if the modifications are appropriate to convey the type of emotion we specified (e.g. is the motion slow enough? Does the slowed motion can be perceived as being sad?).

10.1.2 Scenario 1 – Description

Here, we are not concerned with the context-based response of the robot, the identity of the person, nor the intensity levels of the emotions, and only focused on our goals above. For Scenario 1, we bypassed our Emotional-Personality system using *reserved commands* (Table 10.1). To command the robot to do the Pushup motion, we assign "Wave" to the <action> clause. Using the reserved commands, the Pushup motion will be modified using a set of fixed values for the motion processing parameters for each type of emotion (Table 10.2).

We use the reserved commands in this experiment for the following reason. When the Emotional-Personality system interferes, often the system will not execute the motion because the 'willingness' is not satisfied (see Section 5.5.4) . Specifically, when keywords that trigger Sadness and Anger emotions are found in the input text, the 'willingness' level quickly drops to zero – the robot will not (i.e. 'refuse' to) do the actions we ask the robot to do until the emotion is elevated to somewhat Joy or Fear. Thus, we cannot observe the modified motions if the robot 'refuses' to do the motion

we requested. The behavior of the robot's response with interference from the Emotional-Personality system is observed in Scenario 3.

Table 10.1 Reserved Commands for Testing

Reserved Command	Description
"Say (a) happy <action>"	Do <action> with emotion = Joy
"Say (a) scared <action>"	Do <action> with emotion = Fear
"Say (a) sad <action>"	Do <action> with emotion = Sadness
"Say (an) angry <action>"	Do <action> with emotion = Anger

Table 10.2 Emotions vs. Fixed Motion Processing Parameter Values

Motion Processing Parameter	Emotions			
	Joy	Fear	Sadness	Anger
Continuity (c) (default = 0)	3	-1	2	-2
Tension (t) (default = 0)	-2	4	-2	4
Bias (b) (default = 0)	-1	0	1	2
Resampling rate (r) (default = 1)	3	1	1	5
High gain (h) (default = 1)	1	2	1	1
Medium gain (m) (default = 1)	3	-0.5	1	1
Low gain (l) (default = 1)	1	1	1	1.5

10.1.3 Scenario 1 – Set Up

The values for the motion processing parameters in Table 10.2 are determined based on the following intuitions (explored in Chapter 9):

1) The intuition gained from the animation theories on the continuity of transitions between joint positions, acceleration/deceleration, speed, and range of motion of an action/movement to convey Joy, Fear, Sadness, and Anger. In this case, we select the following motion characteristics for each emotion:

    1. Joy:

        - smooth/continuous transition from one joint angle position to the next,
        - observable acceleration/deceleration,
        - medium speed of motion (slower than the basic motion, but faster than the 7-points-interpolated version),
        - normal or increased range of motion.

    2. Fear:

        - rough/discontinuous transitions between joint angle positions,
        - very little acceleration/deceleration,
        - slow speed of motion,
        - normal or reduced range of motion.

3. Sadness:
    - smooth/continuous transitions between joint angle positions,
    - very little acceleration/deceleration,
    - slow speed of motion,
    - normal or reduced range of motion.
4. Anger:
    - rough/discontinuous transitions between joint angle positions,
    - very high acceleration/deceleration,
    - medium to high speed of motion,
    - increased range of motion.

2) The intuition on the effects of different values of the parameters for the motion processing methods (Kochanek-Bartels Interpolation, (Re)Sampling, and Multiresolution Filtering) on a motion signal data:

1. Kochanek-Bartels Interpolation:
    - *Continuity*: controls the direction (tangent) for the transition from one joint angle position to the next. *Continuity* < 0: more direct, immediate change of direction to the next position (linear, like a military march). *Continuity* >0: smooth, exaggerated motion with sudden change of direction to the next position. *Continuity* = 0: gradual change of

direction from one position to the next.

- *Tension*: controls the smoothness of the transitions between joint angle positions (the sharpness of the curves/corners), and also acceleration and deceleration. *Tension* < 0: rough transitions, more linear interpolation, *Tension* > 0: smooth transitions, more curved interpolation. Tension = 0: neutral/normal interpolation.
- *Bias:* controls the positions of the *overshoot*. *Bias* < 0: pre-shoot, *Bias* > 0: post-shoot, *Bias* = 0: no overshoots.

2. (Re)Sampling:
   - *Rate*: controls the duration of the motion. *Rate* = [1,8]. *Rate* = 1: no resampling, slowest motion, *Rate* = 8: restore basic motion (undoing Interpolation), fastest motion where the original motion is preserved.
3. Multiresolution Filtering:
   - *High gain*: controls the gain of the high frequency components (e.g.: noise) of the motion signal. *High gain* < 1: smoothes the signal, removing noise, *High gain* > 1: accentuate noise, adds to discontinuities in the motion signal. *High gain* = 1: no effect.
   - *Medium gain*: controls the gain of the middle frequency components of the motion signal, affecting the range of motion (amplitude) throughout the motion signal. *Medium gain* < 1: decrease range of motion,

Medium gain > 1: increase the range of motion, *Medium gain* = 1: no effect.

- *Low gain*: controls the gain of the low frequency components of the motion signal, affecting the range of motion around *the middle* of the motion signal, and does not affect the range of motion at the beginning and end of the motion signal. *Low gain* < 1: decrease the range of motion, *Low gain* > 1: increase the range of motion, *Low gain* = 1: no effect.

10.1.4 Scenario 1 – Experiment

The experiment of Scenario 1 was done as follows:

1. Through the user-interface, enter the text phrase:

   ```
   >> Say a happy wave
   ```

2. The ALICE engine will respond with:

   ```
   >>  "a happy wave"
   ```

3. The Perception system, using Regular Expression will extract the emotion keyword: "happy," and the verb keyword: "wave".

4. Evaluate if the executed Waving-arm motion has the characteristics defined for the emotion "happy" with respect to smoothness of transitions, existence of acceleration/deceleration, speed/duration of motion, and the range of motion.

5. Repeat step 1 and 2 by iterating through other emotions: “scared”, “sad”, and “angry.”

10.1.5 Scenario 1 – Results

The results of the experiments using Scenario 1 are as follows:

- The system were able to vary the acceleration, decelration, duration, and range of motion of the Waving-arm motion, but not very well on the continuity of the motion.
- We have confidence that some of our intuition on the relationships between the motion processing method parameters and the motion parameters are valid:
    - Increasing the value of the tension parameter adds acceleration/deceleration, while reducing the value reduces acceleration/deceleration – making the motion more linear/robotic. This property is seen when the movement of the arm gradually speeds up as the arm starts to move (acceleration) and slows down before reaching a stop (deceleration) at different points in the Waving-arm motion.
    - Increasing the rate parameter shorten the duration of the motion – making the motion seems hasted. Decreasing the rate parameter does otherwise. This property is apparent between the emotions: Joy vs. Fear, Joy vs. Sadness, Anger vs. Fear, Anger vs. Sadness, and slightly between Joy vs.

Anger (with Anger tends to be faster because of the higher *rate* value than Joy). There is no difference in speed between Fear vs. Sadness as the *rate* values are the same for both emotions.

- Increasing the medium and high frequency band gains increases the range of motion, and vice versa. The difference between the result from modifying the medium gain and modifying the low gain is seen on where in the motion the range of motion is increased. For the Joy emotion we set the medium gain = 3, while for the Anger emotion, we set the low gain = 1.5. With the Joy emotion, the range of motion was increased when the arm spreads out to the side after fully extended upwards. With the Anger emotion, the arm immediately spreads up and to the side from the rest position, without ever fully extended upwards. Thus the increase of range of motion by the medium gain is in the middle of the motion, while the increase of range of motion by the low gain is throughout the whole motion. This property is useful to realize Exaggeration.

- There are also some unexpected behaviors:
  - The execution of the motion is not entirely consistent. At several instances, the Waving-arm motion with the same emotion type was executed differently. For example, when the emotion is Joy, only 50% of the time the arm was spreaded far to the side (as the effect of the increased medium gain), while in the other times the arm was only extended upwards and

immediately moved back down. It appears that the stream of position data was executed without waiting for the current one to be completed. For example: the current shoulder position is at 0. The next position is 60, and the position after that is -10. As the servo rotates the shoulder to position 60, the second position data (-10) is already received, and the servo immediately changes direction according to this second data before reaching position 60.

10.1.5 Scenario 1 – Conclusions

We achieved Goal #1 and Goal #2 for this Scenario 1, but Goal #3 was only partially satisfied.

Goal #1 was achieved since we have shown that with one basic motion, we can generate four different movements. The four new movements varies with respect to the duration, acceleration/deceleration, and range of motion. The four variations was created only by using different parameter values for our set of motion processing methods (Kochanek-Bartels interpolation, (Re)Sampling, and Multiresolution Filtering).

Goal #2 was achieved by having shown that the Waving-arm motion was executed

according to the mapping of the set of parameter values to the four emotion types (Joy, Fear, Sadness, and Anger) as shown in Table 10.2. In addition, this experiment shows that using the simple motion processing methods, we can quickly modify a motion data without having to painstakingly adjust each servo positions anymore. Instead, we can use simple commands, mapped to a small set of parameter values.

Goal #3 was only partially satisfied as the emotional characteristics of the motion was satisfied based on *our own specifications* on what a Joy, Fear, Sad, or Anger movements *should* look like – based on our developed intuition from animation theories. We are fully aware that the observer's perception of emotional expressions are often highly subjective. An untrained observer would have difficulties distinguishing between Joy or Anger, and Fear or Sadness with our specifications. We even doubt they would be able to tell that we 'infused' emotions into the motion. The observer's perception/opinion is investigated further in Scenario 2 and 3.

10.2 Scenario 2: An Observer's Evaluation

10.2.1 Scenario 2 – Goals

The goals of the experiment in Scenario 2 are:

1) to investigate the perception of several observers (excluding the author) to see whether or not the different variations of the movements are perceived by the

observers as expressing some kind of emotion.

2) To gain feedback from the observers on what they perceive/expect as the characteristics of Joy, Fear, Sadness, and Anger emotions in a movement.

10.2.2 Scenario 2 – Description

We fully acknowledge the subjective nature of our thesis, and the proving of our thesis is a challenge to achieve objectively. Therefore, we invited an observer to provide feedback and the perception on the variations of the motion of the robot. In this experiment, we would like to investigate the following:

- Does/can the observer recognize the executed modified motion as the same action as the original motion? (For example: does the angry Waving-arm motion still recognized as a Waving-arm action?)
- Do our motion-emotion specifications agree with the observer's motion-emotion specifications?

10.2.3 Scenario 2 - Set Up

We use the reserved commands to execute the different Waving-arm behaviors as described in Scenario 1. The observer was told of the goals of this thesis, but was not told of the goals of Scenario 2. The observer was only asked to watch the Waving-arm behaviors of the robot and provide any comments or feedback on whatever the observer perceived from watching the behaviors of the robot at any time during the

experiment.

10.2.4 Scenario 2 – Experiment

The experiments in Scenario 2 was conducted as follows:

1. We enter the reserved commands (see Scenario 1) in random order, with each commands executed at least twice. The observer was not told of which commands we executed.
2. After step 2 is done, the observer was asked the following questions:
    1. Do they notice any difference between each command execution? If yes, how would they describe the differences?
    2. Do they perceive any emotion in the execution? If yes, which motion characteristics did they perceive as showing what emotion? If not, why cannot they perceive any emotion?
3. Lastly, the observer is shown the reserved commands and the motion executions.

10.2.5 Scenario 2 – Result

The result from Scenario 2 is as follows:

- The observer reported distinguishable differences between the Waving-arm motion executed under the Happy, Sad, and Angry commands, but reported no perceived difference between the Scared and Sad motions.
- The observer reported that emotions were not immediately perceived through

the different motions.

- The observer reported that instead of emotions, *attitudes* were perceived from the robot when the Happy and Angry Waving-arm motions were ended with a pose. The Happy Waving-arm motion was ended with a 'flying' pose where the robot spreads its arms to the side, bringing the elbows above the shoulder, and bending the elbows so the lower arms were pointing down at an angle. The Angry Waving-arm motion was ended with a 'hands-on-the-hip' pose where both elbows were pointed out to the side and the elbows were bent so the hands are pointing to the hips.
- Initially, the observer reported of perceiving a “challenging” or “in-your-face” attitude when the robot struck the 'hands-on-the-hip' pose at the end of the Angry Waving-arm motion. With the 'flying' pose at the end of the Happy Waving-arm motion, the observer perceived a rather friendly attitude.
- After being told that the 'flying' pose was at the end of the Happy command, and that the 'hands-on-the-hip' pose was at the end of the Angry command, in subsequent executions of the Happy and Angry commands, the observer immediately associated the poses with emotions.

10.2.6 Scenario 2 - Conclusions

As expected, the observer was not able to report emotional expressions immediately

by just observing the different motions without any interaction with the robot. In other words, without understanding the context of the interaction. We also have some partial confirmations on our second hypothesis on emotional expressions which is presented in Section 5.4. To recall, we said that one of the implication of our second hypothesis is as follows:

> We cannot distinguish which emotion is being expressed solely through the speed and range of motion of the motion.

Although the observer reported noticable differences of speed, acceleration/ deceleration and range of motion between some of the supposedly emotional motions, the observer reported no perceived emotions, at least initially. Instead, perception of attititudes emerges with the observer. However, when some of the motion properties are exaggerated, the observer reported to be able to perceive the attitudes better. For example, we lower the SPD value from 5 to 4[13] which in effect increases the speed of the servos during motion executions. When the Angry Waving-arm motion was executed, the observer reported to perceive a more aggressive behavior in the Waving-arm gesture itself, and not only by the pose at the end.

The results from Scenario 2 gave some indications that without context, emotions are difficult to perceive. Instead, *attitudes* seem to give clearer indications to the user about the emotional state of the robot when context is absent. Exaggeration in the

13 SPD = speed parameter in the RCB-1 string of bytes to manually set the servo positions (see Section 3.4.3.2)

attitudes may give an even clearer indication on the type of behavior (e.g. friendly, aggressive). Attitudes can be thought of as certain gestures that are culture and personality-dependent, that may expresses the internal states or personality of an individual.

In the next scenario, Scenario 3, we observe the perceptions of the observer when context is involved by allowing the observer to directly interact with the system.

10.3 Scenario 3: 'Driving' the Emotion of the Robot with the Emotional-Personality model

10.3.1 Scenario 3 – Goals

The goals of the experiment in Scenario 3 are:

1) to examine if context-based motion responses enable observers to more easily identify the emotion expressed by the robot through its motion, and
2) to evaluate the user experience with the human-robot interaction, with the addition of emotionally-expressive motion responses.

10.3.2 Scenario 3 – Description

As was seen in the results and conclusions of Scenario 2 that perception of emotions in a motion is difficult in the absence of context. Thus, to enable context, we let the observer interact with the robot/system using regular English sentences such as in a

conversation through the user-interface. This time, the response of the robot is regulated by the Emotional-Personality model (discussed in Section 5.5).

10.3.3 Scenario 3 – Set Up

CHAPTER 11 – CONCLUSIONS AND FUTURE WORK

Conclusions

BIBLIOGRAPHY

[AB02] Allbeck, J. and N. Badler. Toward representing agent behaviors modified by personality and emotion. Proceedings of the first annual joint conference on autonomous agents and multiagent systems, pages 202-207. 2002.

[BC89] Bruderlin, A. and T. W. Calvert. Goal-directed, dynamic animation of human walking. Proceedings of the 16th annual conference on computer graphics and interactive techniques, pages 233-242. 1989. ACM New York, NY, USA.

[BRE] Breemen, A. J. N. van, P. Res and N. Eindhoven. Animation engine for believable interactive user-interface robots. Intelligent robots and systems, 2004.(iros 2004). proceedings. 2004 ieee/rsj international conference on.

[BS99] Breazeal, C. and B. Scassellati. How to build robots that make friends and influence people. Intelligent robots and systems, 1999. iros'99. proceedings. 1999 ieee/rsj international conference on. 1999.

[BW95] Bruderlin, A. and L. Williams. Motion signal processing. Computer Graphics, 29(4):97-104, 1995.

[BYM05] Breemen, A. van, X. Yan and B. Meerbeek. Icat: an animated user-interface robot with personality. Proceedings of the fourth international joint conference on autonomous agents and multiagent systems, pages 143-144. 2005. ACM New York, NY, USA.

[Bis07] Bishko, L. The uses and abuses of cartoon style in animation. Animation, 2, 2007.

[Bis92] Bishko, L. Relationships between laban movement analysis and computer animation. Dance and Technology I: Moving Toward The Future:1-9, 1992.

[Bre02] Breazeal, C. L. Designing sociable robots, 2002. Bradford Book, 2002.

[Bre98] Breazeal, C. A motivational system for regulating human-robot interaction, Proceedings of the national conference on artificial intelligence, pp. 54-61, 1998. John Wiley and Sons Ltd.

[CB] Chi, D. M. and N. I. Badler. A motion control scheme for animating expressive arm movements.

[CCZ+00] Chi, D., M. Costa, L. Zhao and N. Badler. The emote model for effort and shape. Proceedings of the 27th annual conference on computer graphics and interactive techniques, pages 173-182. 2000. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[COM] Center of Mass, Website,URL: http://library.thinkquest.org/C006300/noflash/data/five1.htm

[CGM02] Christoudias, C., B. Georgescu and P. Meer. Synergism in low level vision. International conference on pattern recognition, pages 150-155. 2002.

[CM02] Comaniciu, D. and P. Meer. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence:603-619, 2002.

[CMN07] Cassinis, R., L. M. Morelli and E. Nissan. Emulation of human feelings and behaviors in an animated artwork. International Journal on Artificial Intelligence Tools, 16(2):291, 2007.

[CVB01] Cassell, J., H. H. Vilhjálmsson and T. Bickmore. Beat: the behavior expression animation toolkit. Proceedings of the 28th annual conference on computer graphics and interactive techniques, pages 477-486. 2001. ACM New York, NY, USA.

[DAA+07] Delp, S. L., F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman and D. G. Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING BME, 54(11):1940, 2007.

[DD06] Davoudi, M. and M. Davoudi. Intelligent robot motion using fuzzy logic-based ctp and artificial neural networks. Cognitive informatics, 2006. icci 2006. 5th ieee international conference on. 2006.

[FK] Fischer, C. A. and G. V. Kondraske. A new approach to human motion quality measurement. Engineering in medicine and biology society, 1997. Proceedings of the 19th Annual International Conference of the IEEE.

[FND03] Fong, T., I. Nourbakhsh and K. Dautenhahn. A survey of socially interactive robots. Roboties and Autonomous Systems, 42(3-4):143-166, 2003.

[GC02] Georgescu, B. and CM Christoudias. The Edge Detection and Image segmentatiON (EDISON) system. Robust Image Understanding Laboratory, Rutgers University. code available at http://www. caip. rutgers.

edu/riul/research/code. html, 2002.

[GM85] Girard, M. and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. Proceedings of the 12th annual conference on computer graphics and interactive techniques, pages 263-270. 1985. ACM Press New York, NY, USA.

[GO03] Ghasem-Aghaee, N. and T. I. Oren. Towards fuzzy agents with dynamic personality for human behavior simulation. Summer computer simulation conference, pages 3-10. 2003. Society for Computer Simulation International; 1998.

[GRA+02] Gratch, J., J. Rickel, E. Andre, J. Cassell, E. Petajan and N. Badler. Creating interactive virtual humans: some assembly required. Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications], 17(4):54-63, 2002.

[HBT99] Huang, Z., R. Boulic and D. Thalmann. Motion editing using multiresolution filtering. Proceedings of MMM99, pp323-334, 1999.

[Ham95] Hamburg, J. Coaching athletes using laban movement analysis. JOPERD--The Journal of Physical Education, Recreation & Dance, 66(2), 1995.

[Hye96] Hyeongseok, N. I. Animating human locomotion with inverse dynamics. IEEE Computer Graphics and Applications:50-59, 1996.

[KM04] Kölsch, M. Vision based hand gesture interfaces for wearable computing and virtual environments, Ph.D Dissertation, UCSB, 2004.

[KT04] Kölsch, M., Turk, M., Robust hand detection, Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, 2004.

[Lab08] Labunsky, Dimitry. ECE 574 class project on Speech Recognition, 2008.

[Las87] Lasseter, J. Principles of traditional animation applied to 3d computer animation. Proceedings of the 14th annual conference on computer graphics and interactive techniques, pages 35-44. 1987. ACM New York, NY, USA.

[Luk06] Lukac, M. Common Robot Language (CRL), Unpublished paper, 2006.

[MG01] Meer, P. and B. Georgescu. Edge detection with embedded confidence. IEEE Transactions on Pattern Analysis and Machine Intelligence:1351-1365,

2001.

[MIM+] Miwa, H., K. Itoh, M. Matsumoto, M. Zecca, H. Takanobu, S. Roccella, M. C. Carrozza, P. Dario and A. Takanishi. Effective emotional expressions with emotion expression humanoid robot we-4rii. Iros, 2004.

[MSK07] Michalowski, M. P., S. Sabanovic and H. Kozima. A dancing robot for rhythmic social interaction. ACM SIGCHI/SIGART Human-Robot Interaction:89-96, 2007.

[Nau08] Nautilus team, Speech Recognition Senior Capstone Project, Department of Electrical and Computer Engineering, Portland State University, 2008.

[NF05] Neff, M. and E. Fiume. Aer: aesthetic exploration and refinement for expressive character animation. Proceedings of the 2005 acm siggraph/eurographics symposium on computer animation, pages 161-170. 2005. ACM New York, NY, USA.

[Net07] Nettle, D. Personality: what makes you the way you are, 2007. Oxford University Press, USA, 2007.

[PG96] Perlin, K. and A. Goldberg. Improv: a system for scripting interactive actors in virtual worlds. Proceedings of the 23rd annual conference on computer graphics and interactive techniques, pages 205-216. 1996. ACM New York, NY, USA.

[POP98] Papageorgiou, C. P., M. Oren and T. Poggio. A general framework for object detection. Computer vision, 1998. sixth international conference on, pages 555-562. 1998.

[RD] Rett, J. and J. Dias. Bayesian models for laban movement analysis used in human machine interaction.

[RH91] Raibert, M. H. and J. K. Hodgins. Animation of dynamic legged locomotion. ACM SIGGRAPH Computer Graphics, 25(4):349-358, 1991.

[SK04] Schneiderman, H. and T. Kanade. Object detection using the statistics of parts. International Journal of Computer Vision, 56(3):151-177, 2004.

[Smi95] Smith, A. R. A pixel is not a little square. i> Microsoft Technical Memo,</i, 1995.

[TLB88] Touzi, R., A. Lopes and P. Bousquet. A statistical and geometrical edge

detector for sar images. Geoscience and Remote Sensing, IEEE Transactions on, 26(6):764-773, 1988.

[TP91] Turk, M. and A. Pentland. Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1):71-86, 1991.

[UAT95] Unuma, M., K. Anjyo and R. Takeuchi. Fourier principles for emotion-based human figure animation. Proceedings of the 22nd annual conference on computer graphics and interactive techniques, pages 91-96. 1995. ACM New York, NY, USA.

[VJ04] Viola, P. and M. J. Jones. Robust real-time face detection. International Journal of Computer Vision, 57(2):137-154, 2004.

[Wal] Wallace, R. S. The anatomy of A.L.I.C.E. URL: http://www.alicebot.org/anatomy.html

[Wel] Welman, C. Inverse kinematics and geometric constraints for articulated figure manipulation.

[Whi76] White, G.M. *Speech recognition: a tutorial overview*, Computer, Vol. 9, No. 5, pp 40-53, May 1976.

[Wiki:BF] Wikipedia: Big Five, URL: http://en.wikipedia.org/wiki/Big_five

[Wiki:For] Wikipedia: Formant, URL: http://en.wikipedia.org/wiki/Formant

[Wiki:KBS] Wikipedia: Kochanek-Bartels Spline, URL: http://en.wikipedia.org/wiki/Kochane-Bartels_spline

[Wiki:LMA] Wikipedia: Laban Movement Analysis, URL: http://en.wikipedia.org/wiki/Laban_Movement_Analysis

[Wiki:SR] Wikipedia: Speech Recognition, URL: http://en.wikipedia.org/wiki/Speech_recognition

[Wiki:UV] Wikipedia: Uncanny Valley, URL: http://en.wikipedia.org/wiki/Uncanny_valley

[WM2000] Wiens, S., Mezzacappa, E. S., Katkin, E. S., *Heartbeat detection and the experience of emotions*, Cognition and Emotion, Vol 14, No. 3, pp. 417-427, 2000.

[ZB89] Zhao, J. and N. I. Badler. Real time inverse kinematics with joint limits and spatial constraints, 1989. University of Pennsylvania, School of

Engineering and Applied Science, Dept. of Computer and Information Science, 1989.

[Zha01] Zhao, L. Synthesis and acquisition of laban movement analysis qualitative parameters for communicative gestures. Unpublished PhD thesis, Computer and Information Science, Univ.of Pennsylvania, Philadelphia, PA, 2001.

APPENDIX

Distance Measures

1. Euclidean Distance:

Euclidean distance is also often referred to as the "ordinary" distance measure (wikipedia), i.e. distance between two points that can be measured using Pythagorean theorem. To calculate the Euclidean distance between two points *p* and *q,* D(p,q) in *n*-dimensional space:

$$D(p,q)=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

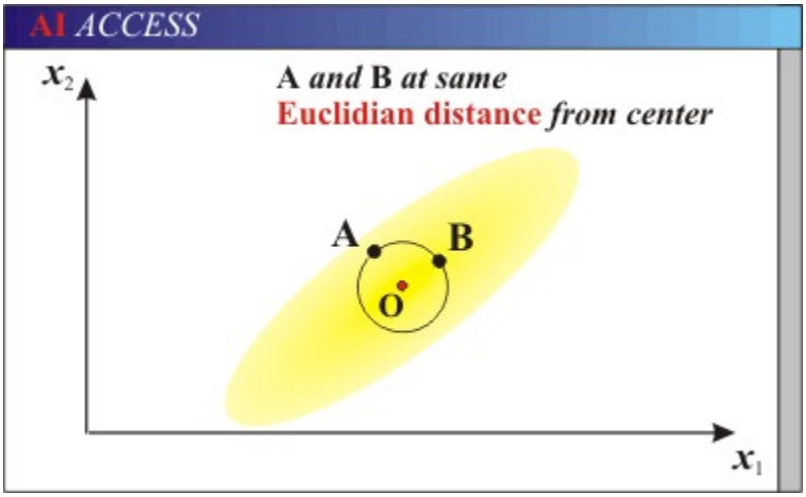
2. Mahalonobis Distance:

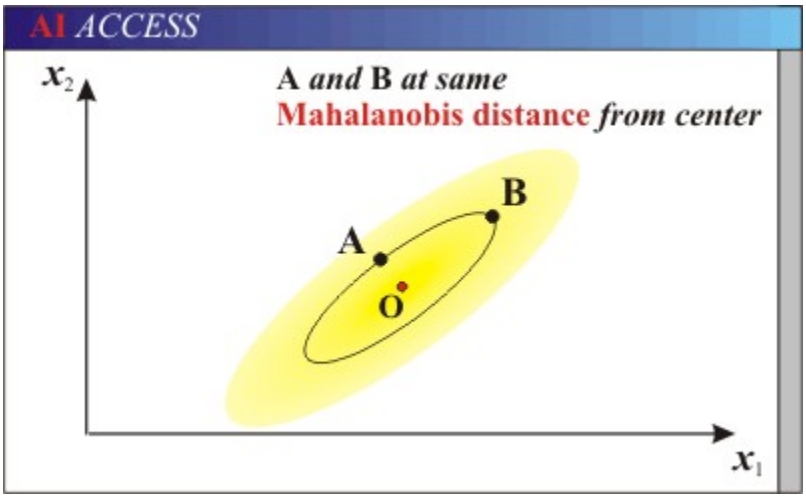
Maholonobis distance is a more general notion of distance than Euclidean distance. Here, the distance measurement of points in data set *p* is calculated against the correlation (i.e. covariance matrix) of data set *q*. The data set *p* is then considered as the *reference data set*. When the data sets are represented as *m*-by-*n* matrices, *m* is the number of rows of the matrix and correspond to the number of observations (or samples) in the data set. *n* is then the dimension of the data set (e.g. the matrix for a data set in 2-dimensions, say, x and y, with ten data samples will be of size 10-by-2). Both data set *p* and *q* must have the same number of dimensions, but the reference data set (in this case, *p*) must have a larger number of observations than the compared data. Thus, the Mahalonobis distance can be calculated as:

$$D_M(\vec{p},\vec{q})=\sqrt{(\vec{p}-\vec{q})^T S^{-1}(\vec{p}-\vec{q})}$$

Where:

- $D_M(\vec{p},\vec{q})$ = Mahalonobis distance of data *q* with respect to *p*.
- S^{-1} = the covariance matrix for *p* and *q*, inverted.
- $\vec{p},\vec{q}$ = data in *p* and *q*, respectively. Represented as 1-by-n vectors, where *n* is the dimension of the data





http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_mahalanobis.htm

APPENDIX

Understanding the Rotation/Transformation Matrix

The intuition for reading the rotation/transformation matrix can be summarized as follows:

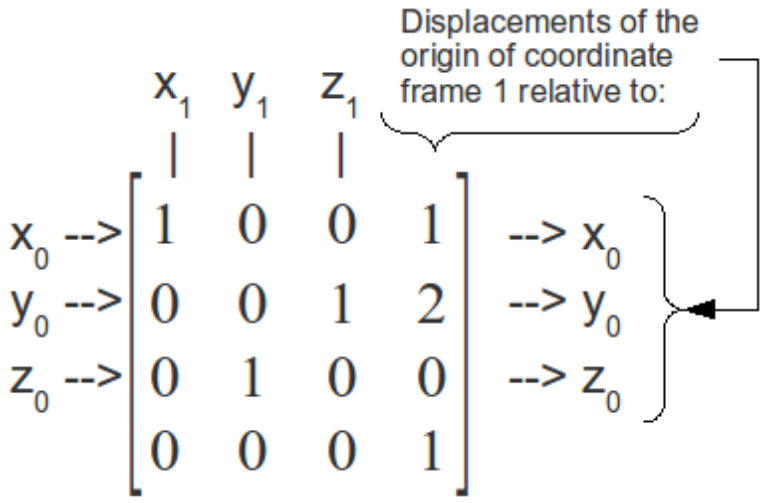
1. Think of the first three rows of the matrix as the axes of the first coordinate frame,
2. The first three columns as the axes of the second coordinate frame,
3. The values in the first 3 x 3 part of the matrix describes the *orientation* the axes of the second coordinate frame relative to the first coordinate frame, i.e.: how much each axis of the second coordinate frame is rotated with respect to the axis of the first coordinate frame.
    1. The orientations can be calculated by plugging the values of the D-H Parameters into the matrix:

$$ {}^{0}T_{1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1\cos\alpha_1 & \sin\theta_1\sin\alpha_1 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1\cos\alpha_1 & -\cos\theta_1\sin\alpha_1 & a_1\sin\theta_1 \\ 0 & \sin\alpha_1 & \cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

    2. To simplify the calculation of this transformation matrix, often the joints are oriented perpendicular or parallel to each other.
    3. When the orientations of the joints are perpendicular, the values in the matrix will be either 1 or 0. Otherwise, the values are $0 < x < 1$.
4. And the three rows of the fourth column as the translation of the second coordinate frame relative to the first coordinate frame.

For example:

$$ {}^{0}T_{1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



The transformation matrix ${}^{0}T_{1}$ describes that:

- the x axis of frame 1 (x_1) is aligned and parallel with the x axis of frame 0 (x_0)
- the y axis of frame 1 (y_1) is aligned and parallel with the z axis of frame 0 (z_0)
- the z axis of frame 1 (z_1) is aligned and parallel with the y axis of frame 0 (y_0)
- the origin of frame 1 (x_1=0, y_1=0, z_1=0) is displaced (translated) by 1 unit length along x_0, and 2 unit lengths along y_0, and 0 unit lengths along z_0.

Therefore, we can immediately tell when two coordinate frames have the same orientations (regardless of their displacements from each other): the first 3 x 3 part of the matrix is an identity matrix, or the diagonal is all 1's. For example:

$$ {}^{0}T_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

In this case, coordinate frame 1 has the same orientation as coordinate frame 0 for all its axes x_1, y_1, and z_1, and only displaced 1 unit lengths along y_0.

REFERENCES

Allbeck, J. & Badler, N. 2002, "Toward Representing Agent Behaviors Modified by Personality and Emotion", *Proceedings of the First Annual Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 202.

Bishko, L. 2007, "The Uses and Abuses of Cartoon Style in Animation", *Animation,* vol. 2.

Bishko, L. 1992, "Relationships between Laban Movement Analysis and computer animation", *Dance and Technology I: Moving Toward The Future,* , pp. 1-9.

Braitenberg, V. 1984, Vehicles: Experiments in synthetic psychology. Cambridge, MA: MIT Press.

Breazeal, C.L. 2002, *Designing Sociable Robots,* Bradford Book.

Bruderlin, A. & Calvert, T.W. 1989, "Goal-directed, dynamic animation of human walking", *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*ACM New York, NY, USA, , pp. 233.

Bruderlin, A. & Williams, L. 1995, "Motion signal processing", *Computer Graphics,* vol. 29, no. 4, pp. 97-104.

Cassell, J., Vilhjálmsson, H.H. & Bickmore, T. 2001, "BEAT: the Behavior Expression Animation Toolkit", *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*ACM New York, NY, USA, , pp. 477.

Chi, D., Costa, M., Zhao, L. & Badler, N. 2000, "The EMOTE model for effort and shape", *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, , pp. 173.

Chi, D.M. & Badler, N.I. "A motion control scheme for animating expressive arm movements",

.

Davoudi, M. & Davoudi, M. 2006, "Intelligent Robot Motion using Fuzzy logic-Based CTP and Artificial Neural Networks", *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on*.

Filipsson, Marcus, *Speech analysis tutorial*, URL: http://http://www.ling.lu.se/research/speechtutorial/tutorial.html (Last accessed: August 3, 2009)

Ghasem-Aghaee, N. & Oren, T.I. 2003, "Towards Fuzzy Agents with Dynamic Personality for Human Behavior Simulation", *SUMMER COMPUTER SIMULATION CONFERENCE*Society for Computer Simulation International; 1998, , pp. 3.

Girard, M. & Maciejewski, A.A. 1985, "Computational modeling for the computer animation of legged figures", *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*ACM Press New York, NY, USA, , pp. 263.

Hamburg, J. 1995, "Coaching Athletes Using Laban Movement Analysis.", *JOPERD--The Journal of Physical Education, Recreation & Dance,* vol. 66, no. 2.

Hyeongseok, N.I. 1996, "Animating Human Locomotion with Inverse Dynamics", *IEEE Computer Graphics and Applications,* , pp. 50-59.

Jelinek, F. 1997, Statistical Methods for Speech Recognition, Chapter 1, pp. 4-5, The MIT Press

Lang, A., Measuring Psychological Responses to Media Messages, Lawrence Erlbaum Associates, 1994.

Lasseter, J. 1987, "Principles of traditional animation applied to 3D computer animation", *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*ACM New York, NY, USA, , pp. 35.

Miwa, H., Itoh, K., Matsumoto, M., Zecca, M., Takanobu, H., Roccella, S., Carrozza, M.C., Dario, P. & Takanishi, A. "Effective Emotional Expressions with Emotion Expression Humanoid Robot WE-4RII", *IROS, 2004*.

Neff, M. & Fiume, E. 2005, "AER: aesthetic exploration and refinement for expressive character animation", *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*ACM New York, NY, USA, , pp. 161.

Nettle, D. 2007, *Personality: What Makes You the Way You Are,* Oxford University Press, USA.

NEXI, MIT Personal Robots Group, URL: http://robotic.media.mit.edu/projects/robots/mds/overview/overview.html

Perlin, K. & Goldberg, A. 1996, "Improv: a system for scripting interactive actors in virtual worlds", *Proceedings of the 23rd annual conference on Computer*

*graphics and interactive techniques*ACM New York, NY, USA, , pp. 205.

Raibert, M.H. & Hodgins, J.K. 1991, "Animation of dynamic legged locomotion", *ACM SIGGRAPH Computer Graphics,* vol. 25, no. 4, pp. 349-358.

Rett, J. & Dias, J. "Bayesian models for Laban Movement Analysis used in Human Machine Interaction", .

Schneiderman, H. & Kanade, T. 2004, "Object Detection Using the Statistics of Parts", *International Journal of Computer Vision,* vol. 56, no. 3, pp. 151-177.

Unuma, M., Anjyo, K. & Takeuchi, R. 1995, "Fourier principles for emotion-based human figure animation", *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*ACM New York, NY, USA, , pp. 91.

van Breemen, A., Yan, X. & Meerbeek, B. 2005, "iCat: an animated user-interface robot with personality", *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*ACM New York, NY, USA, , pp. 143.

Viola, P. & Jones, M.J. 2004, "Robust Real-Time Face Detection", *International Journal of Computer Vision,* vol. 57, no. 2, pp. 137-154.

Wallace, R. S. The anatomy of A.L.I.C.E. URL: http://www.alicebot.org/anatomy.html

Welman, C. "Inverse kinematics and geometric constraints for articulated figure manipulation", .

White, G.M. *Speech recognition: a tutorial overview*, Computer, Vol. 9, No. 5, pp 40-53, May 1976.

Wikipedia: Formant, URL: http://en.wikipedia.org/wiki/Formant

Wikipedia: Kochanek-Bartels Spline, URL: http://en.wikipedia.org/wiki/Kochane-Bartels_spline

Wikipedia: Laban Movement Analysis, URL: http://en.wikipedia.org/wiki/Laban_Movement_Analysis

Wikipedia: Speech Recognition, URL: http://en.wikipedia.org/wiki/Speech_recognition

Wiens, S., Mezzacappa, E. S., Katkin, E. S., *Heartbeat detection and the experience of emotions*, Cognition and Emotion, Vol 14, No. 3, pp. 417-427, 2000.

Zhao, J. & Badler, N.I. 1989, *Real Time Inverse Kinematics with Joint Limits and Spatial Constraints,* University of Pennsylvania, School of Engineering and Applied Science, Dept. of Computer and Information Science.

Zhao, L. 2001, "Synthesis and acquisition of Laban Movement Analysis qualitative parameters for communicative gestures", *Unpublished PhD thesis, Computer and Information Science, Univ.of Pennsylvania, Philadelphia, PA,* .