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. Speech recognition
        2. Text
IV. Cognition System
    1. Communication norms
    2. Emotion-base
    3. Personality-base
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

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.

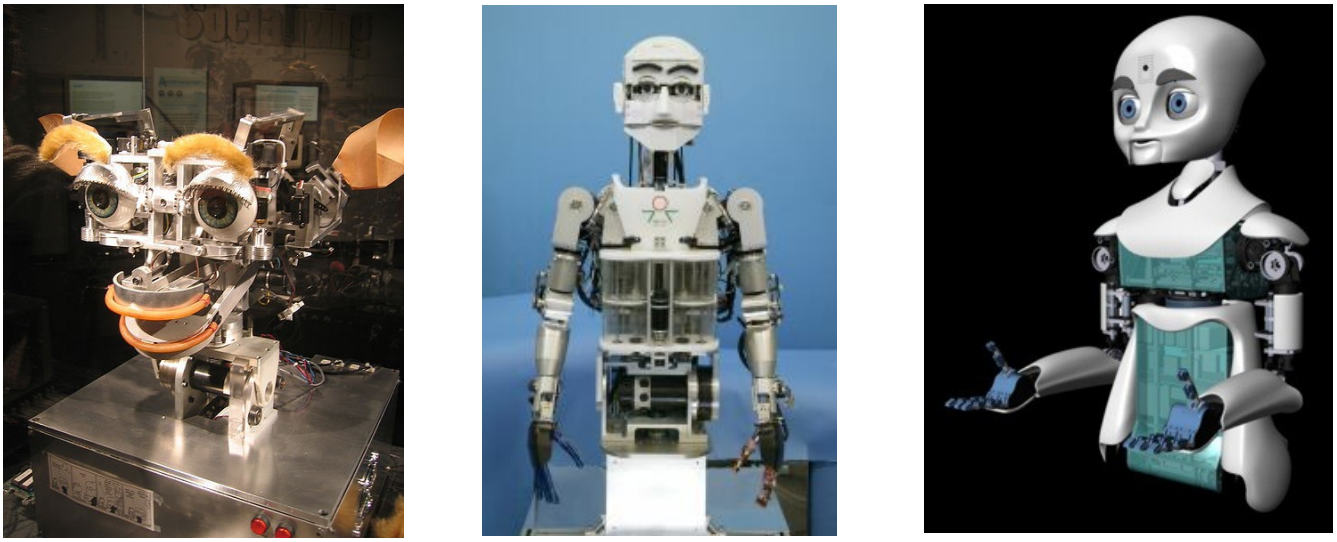
The earliest robot application is in automation of some manufacturing process, such as in car assembly facilities. Most of their applications are to do structured tasks that cannot be done easily by human workers, such as lifting and assembling heavy components. In addition, because of their speed and

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 (Breazeal 2002) 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 (Miwa et al.) (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 it 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 express the same things and to say the same things as the user did, not by cryptic messages.

In addition to be able to know what action (motion) to do, 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 will 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) [reference to EMOTE]. 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 on the other hand, have never been formally modeled as a computational model, but its terminologies kept being mentioned to refer to the characteristics of a motion [iCat, EMOTE, Bruderlin, Lasseter, Unuma]. 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 (Bruderlin, Williams 1995). 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) (Bishko 2007).

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 on 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 to 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

(Bishko 2007, Lasseter 1987) 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 (Bishko 2007) 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 it 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 adverbs (similar to *hedges* in fuzzy logic) modifies the execution of the selected actions or motions. To ensure that the modifications to the motions are realistic and natural (i.e. 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 represents/expresses its internal states in context with the interaction through the resulting motions.

CHAPTER 2 - RELATED WORKS ON MOTION OF INTERACTIVE HUMANOID ROBOTS

2.1. Interactive/Sociable Robots

In her book *Designing Sociable Robots* (Breazeal 2002), 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 for 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

Much of the technical research around automated humanoid robot animation has some serious disconnect with the artistic aspect of animation. As seen in some of the most advanced sociable robots

above, often the expressions are just in postures, and less in the motion itself. This often results in robot movements that still feels awkward and eerie to the human observers, even with complex physical and mathematical model 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.

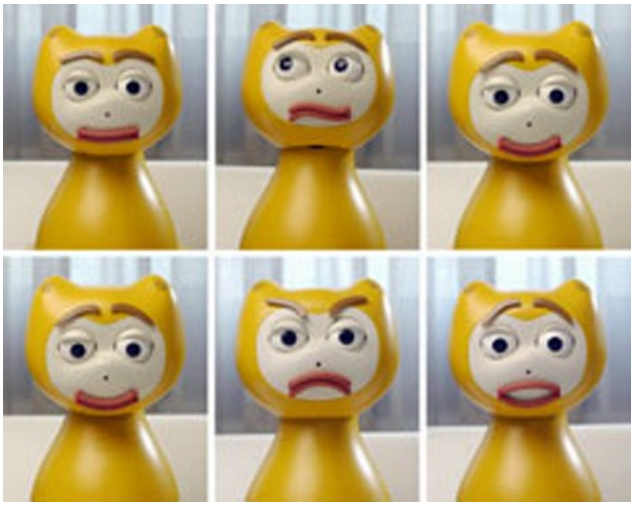


Figure 2.1 iCat Robot.

Van Breemen's iCat (van Breemen, Yan & Meerbeek 2005) 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 technique to enable smooth transitions between animations.

Many researches toward 'fluid', 'natural-looking', or 'human-like' animations are focused on legged locomotion. For example, (Girard, Maciejewski 1985, Bruderlin, Calvert 1989, Raibert, Hodgins 1991, Hyeongseok 1996) 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 model, animators can easily generate different kinds of walking by entering a few parameters at a fairly high level, such as: figure height, leg length, weights, locomotion speed, and so on. 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 for 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. (Welman , Zhao, Badler 1989) 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.

(Lasseter 1987) 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.

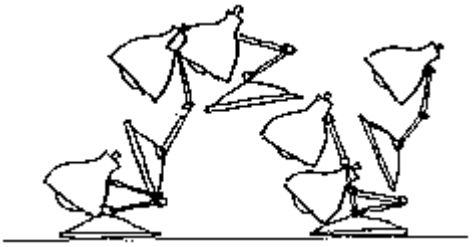


Figure 2.2. (Lasseter 1987) demonstrated how traditional animation principles can be applied to animate jumping in computer-generated rigid body models better.

Badler and his research group approached the issue from a dance notation perspective (Chi et al. 2000, Chi, Badler). Taking inspiration from a theory called 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. LMA 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 (Bishko). 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 part 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) (Allbeck, Badler 2002), AER (Aesthetic Exploration and Refinement) (Neff, Fiume 2005), BEAT (Behavior Expression Animation Toolkit) (Cassell, Vilhjálmsson & Bickmore 2001), Improv (Perlin, Goldberg 1996), 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 have 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 contains 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 so on. 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) *[reference?]*. 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 animators must 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

(Bruderlin, Williams 1995) introduced a powerful metaphor for animation: looking at 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, modify certain aspects of the motion elegantly, which was done 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 were raised higher, and the foot landed farther. Conversely, when the middle or low frequency gains are reduced, the steps become "smaller". These show that there is a correlation between using this approach and the artistic side of animation (Exaggeration is one of the DAP).

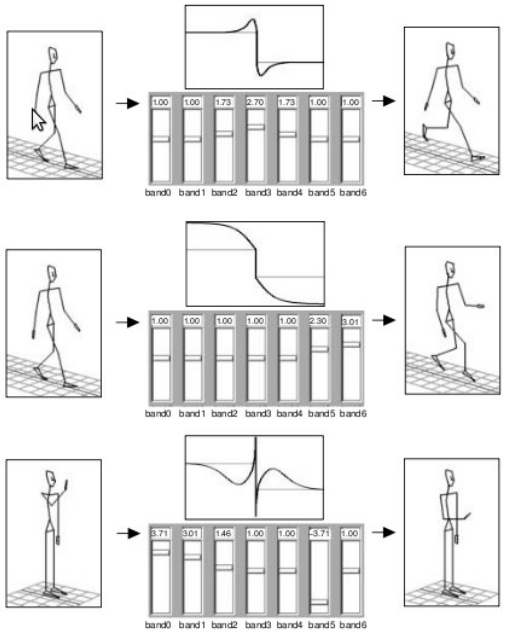


Figure 2.3 Modifying gains of different frequency bands from multiresolution filtering can exaggerate or dampen the range of motion

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 is 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 this can be applied to automate motion modifications.

A similar signal/filtering approach was implemented in iCat, 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 animations using these approaches.

(Unuma, Anjyo & Takeuchi 1995) 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 normal walking animation). But the process is slow, and not feasible for real-time applications (e.g. modifying animation as it runs). They also noted 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 that 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, 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.). (Davoudi, Davoudi 2006) 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. (Ghasem-Aghaee, Oren 2003) 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 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 they are in human-human interactions.

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

3.2 Hypothesis

Thus, as we recognize the challenges and possible solutions to achieve our Goal, we developed several hypotheses. Throughout our hypotheses, we assume 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 robot must be able to perceive users' inputs from visual cues, and natural human

language (in this case, English). In addition, the robot must respond in similar human-like mannerisms through motion and natural language through speech and/or texts. We assume a minimal set of interaction modalities that is acceptable to meet the expectations, through face detection and recognition, simple speech synthesis, and natural language processing.

3. To have human-like behaviors, the behaviors of the robot must express some kind of emotion, personality, or physical states.
4. 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. To support our first assumption above, we implemented human-like interaction modalities, such as vision, conversation (text or otherwise), and motion.
        1. Vision input is obtained through a normal webcam and is used to detect and recognize faces.
        2. Conversation is both as input and output, in the form of text input/output with additional speech output, controlled by a chatbot program.
        3. Motion is the gestures or actions performed by the robot as a response to the user's

inputs.

2. The interaction flow must follow 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. Users will be presented with feedback of the current states and perceptions of the robot/system. But during testing/experiment, these information will be hidden.
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; it immediately gives 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 rule 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.

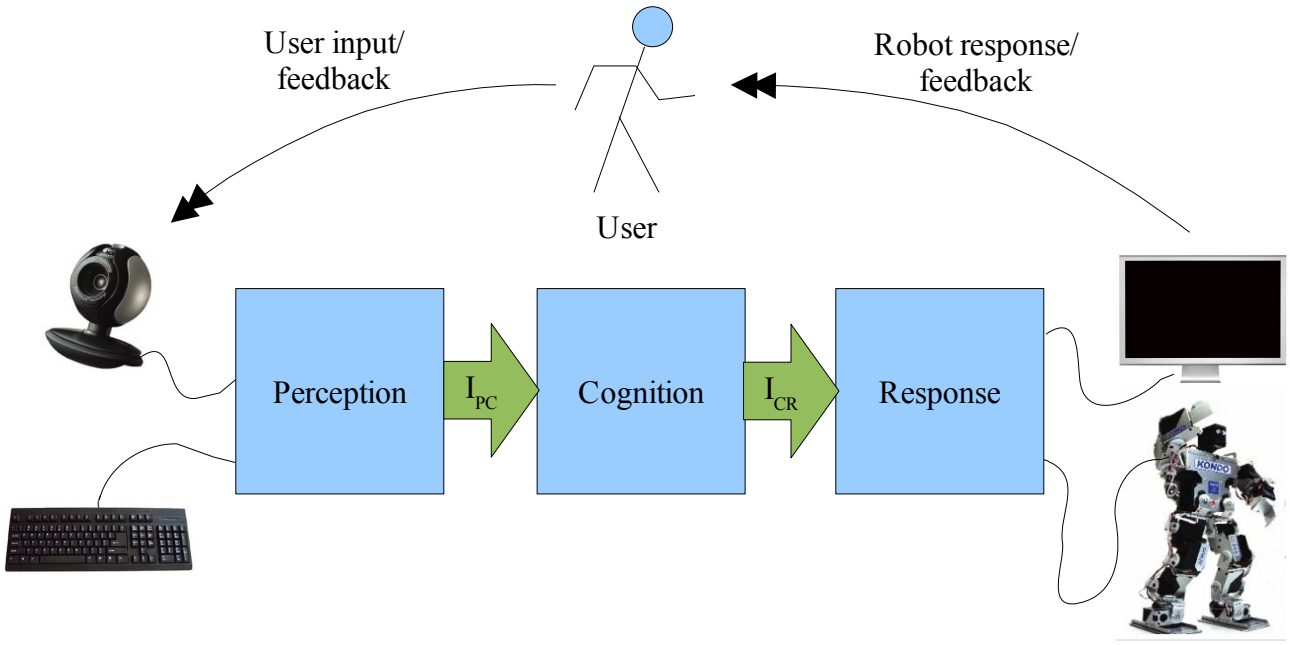


Figure 3.1. The system.

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 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 keywords.

3. The Cognition block receives the information in I_{PC} as its input.
4. The Cognition block then uses a combination of regular expression, and a set of interaction rules to process the information in I_{PC} to generate I_{CR} (Information from Cognition to Response) as its output. I_{CR} consists of information such as: text response, selected motion, selected motion modifiers, and values of motion modifier parameters.
5. The Response block receives the information in I_{CR} as its input.
6. The Response block is responsible to direct the information from I_{CR} to their corresponding output channels. The text response is to be displayed to the monitor, and the motion response is to be directed to the robot. Before the motion response is directed to the robot, the motion modifiers are applied to the selected motion with the specified parameter values, according to the instructions from the Cognition block.
7. The outputs of the Response block are in texts, and motion of the robot.

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 1.6GHz dual processors, and 2GB of RAM.
- A Logitech QuickCam Communicate MP webcam for face detection and recognition.
- A Kondo KHR-1 humanoid robot, which is described further 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 board that are coupled together.

The first RCB-1 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 9.1 Kondo KHR-1 humanoid robot

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.

3.4.2.1 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 so forth. 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.

3.4.2.2 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.

3.5 Experiment Method and Result Analysis

After the set of rule 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. 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. 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

There are three main components to interactions: perception, cognition, and response. 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 speech and motion (i.e. gestures, actions), is the reaction or answer to the input stimuli.

In this chapter, we will discuss the components for perception. In robotics, perception generally implies sensors. Because perception is about input stimuli, the robot requires sensors to gather these stimuli. For social interactions, or human-to-human-like interactions, the common stimuli are from: vision, speech or voice, and touch.

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

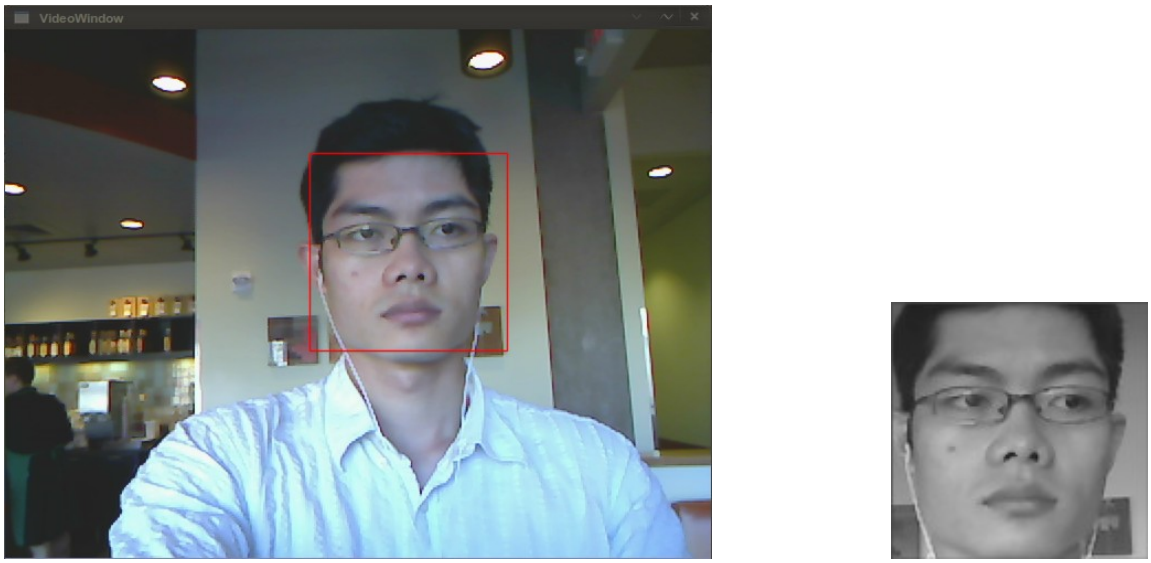
For our application in socially interactive robots, we are particularly interested in computer vision for face detection and recognition, object detection and recognition, and gesture recognition. Additionally, we might also benefit from not just general face detection, but also facial gestures or expressions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

We 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

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

objects, noise, and other artifacts that does not contribute to the facial features. In other words, they are not making it any easier to recognize the face. 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 through face detection. When the face detection system identifies a face in an image, it 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 eliminate the background noise.



a *b*

Figure 4.1 (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) (Viola, Jones 2004). 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 (Schneiderman, Kanade 2004).

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].

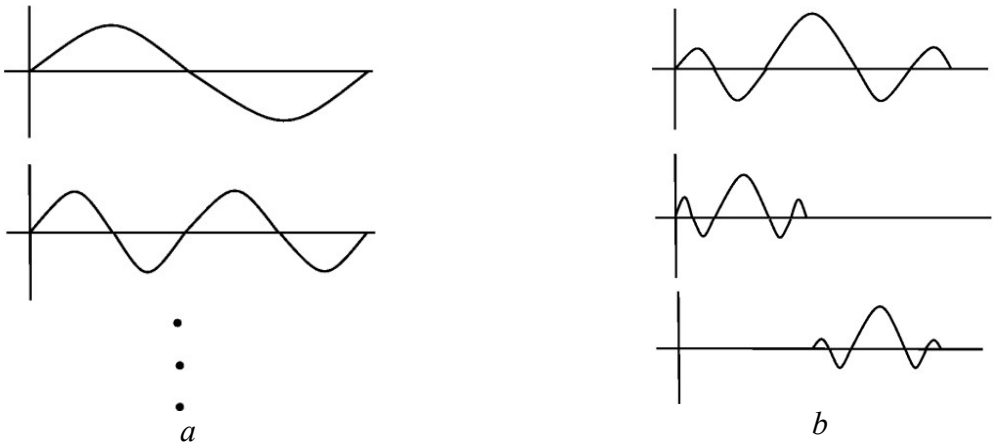


Figure 4.2 (a) Fourier basis functions, (b) Wavelet[2] basis functions (image taken from [haarwaveletbasis])

Thus, the wavelet approach has an interesting property: the basis function can be manipulated by *shifting* and *scaling*. As seen in Figure 4.2b, 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 the 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 more than one features that overlap. Figure 4.3 illustrates this property. In Figure 4.3 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

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

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.

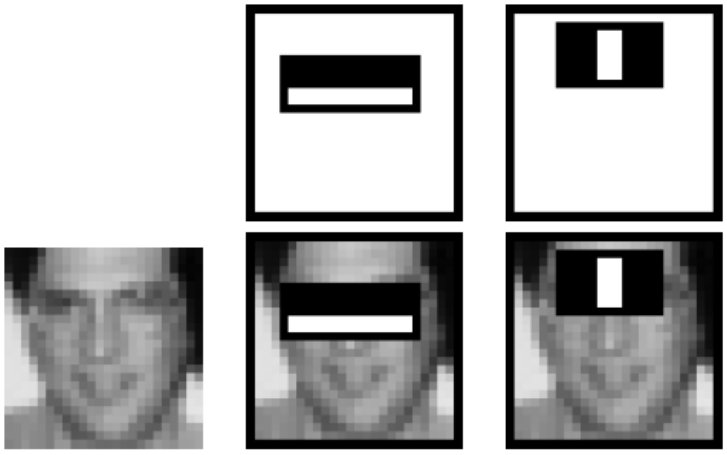


Figure 4.3b Features example (image taken from (Viola, Jones 2004))

In Viola-Jones method, much inspired by the work of Papageorgiou [generalframeworkobjdetect], the most basic features evaluates the horizontal (figure 4.4a), vertical (figure 4.4b), and diagonal (figure 4.4d) features. To evaluate more complex features of the face, more complex rectangle features are used (figure 4.4c).

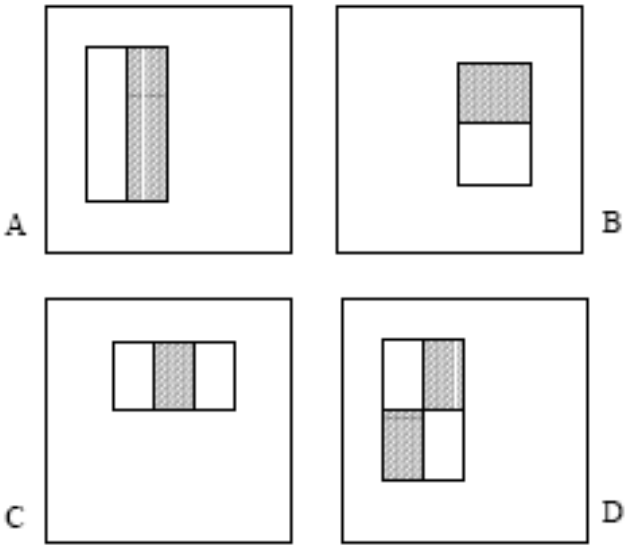


Figure 4.4 Example of rectangle features (image taken from (Viola, Jones 2004))

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 4.5 Example of two-rectangle feature value

For example, for the two-rectangle feature (figure 4.5), 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 4.6), 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 4.6 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 4.7 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 4.8).

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

Figure 4.8 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 especially 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 output approximation. For example: when we have a set of data as plotted in figure 4.9a, 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, 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.

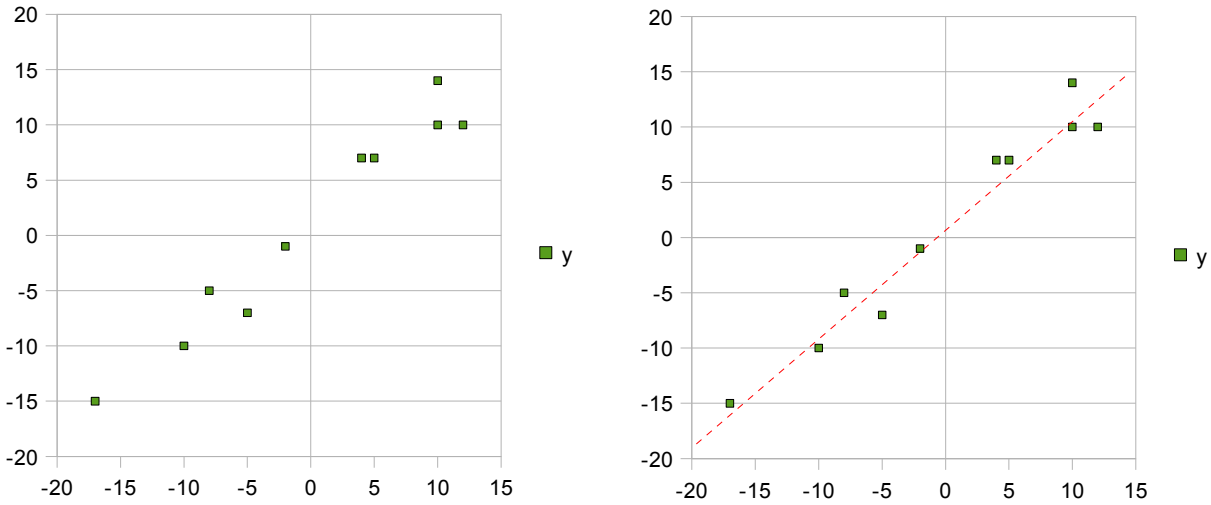


Figure 4.9 Regression analysis on a data set

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, which each element is the distance you travelled each day with your car. The second set is set G, which 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 X 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 of data set X and Y, respectively.

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

Such that, 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* x 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 retrieval (i.e. recognition).

4.1.3. Gesture Recognition

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, and so on. 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.

To enable gesture recognition, first, the robot needs to identify where is the person that is performing the gesture (i.e., by shape, face). 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 thr system is supposed to identify the 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 bythe Edge Detection and Image SegmentatiON (EDISON) system, developed by Chris M. Christoudias and Bogdan Georgescu from Rutgers University.

4.1.4. Stereo Vision

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 rach 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 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 consists 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

3 Tactile sensor, analogous to (and inspired by) antennas on insects.
4 The intrinsic parameters of a camera includes: the focal point of the camera, and the epipolar geometries.

the image planes. All this information is described in a geometrical description, and are 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 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 meanings 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 exists 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 establsihed/determined that the condition is ready for a conversation, all the text responses from the conversational agent is displayed on screen, while at the same time also synthesized to speech.

4.2.1 ALICE

We used the ALICE engine to handle the conversation between the user and the robot. 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 for the Turing Test. For our system, we used the Python 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 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>	I am tired

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

	<pattern>I am very * today</pattern> <template><srai>I am <star /></srai></template> </category>	
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 a direct reduction 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 are 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 on: 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 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 that involves 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.

4.2.2. Speech (Recognition)

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 [SRwiki]. 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 used for 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 (Wikipedia: Formant), 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 4.9. 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 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).

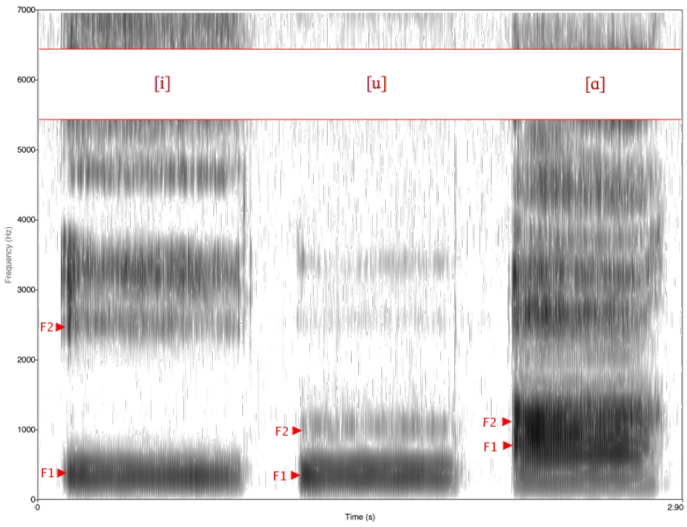
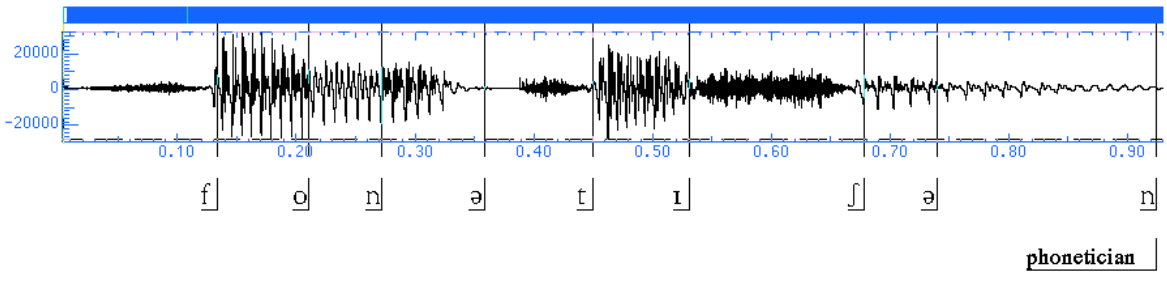


Figure 4.10 Spectrogram of English vowel i, u, and a (image taken from [Formantwiki])

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.

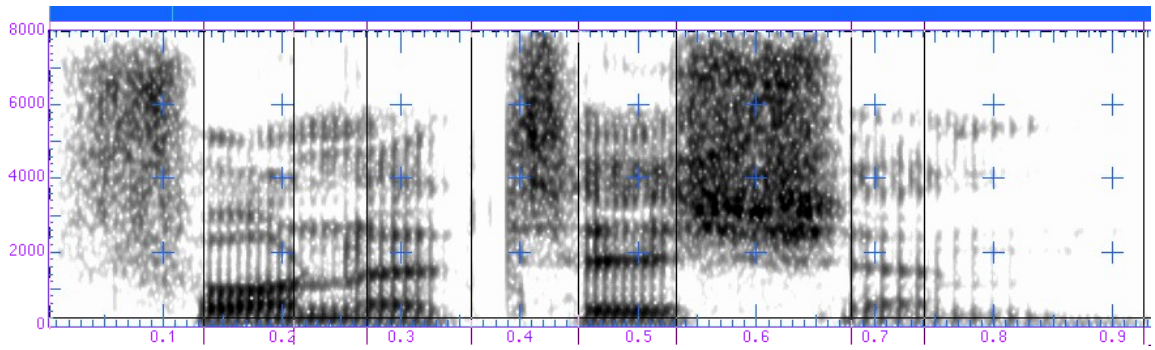
7 Some notations may start at zero (F0).

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 4.10a), 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 4.10a, the X-axis represents time, and the Y-axis represents the pressure (in Pascal (Pa)). The other models are fundamental frequency (figure 4.10b[9]), spectrum (figure 4.10c), spectrogram (figure 4.10d), and waterfall spectrogram (figure 4.10e). The fundamental frequency (figure 4.10b) refers to base formant (F0), where the X-axis represents time, and the Y-axis represents frequency. The spectrogram representation (figure 4.10c) 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 4.10d) shows the frequency (X-axis) and amplitude (Y-axis) of a voice sample (figure 4.10d shows the sample at 0.15 seconds into the utterance of "phonetician"). The waterfall spectrogram (figure 4.10e) is a representation of the spectrum representation of figure 4.10d 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].

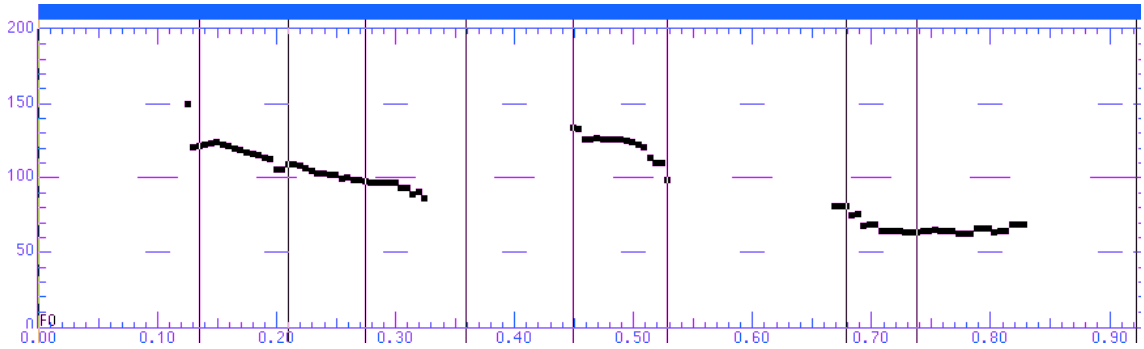


a

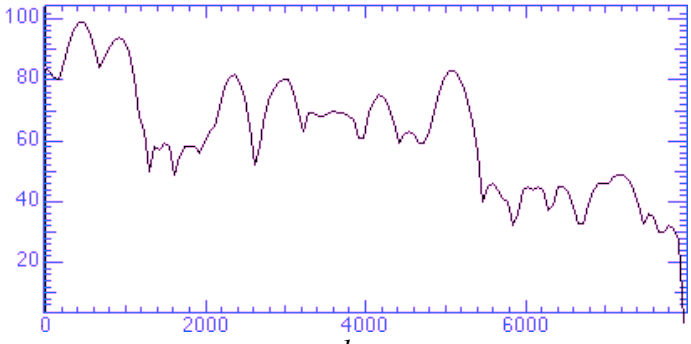
8 The nine segments are not part of the waveform, but shown to separate the phonemes in the pronunciation.
9 Figure 3.2b only shows the peaks of the lowest formant.


8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

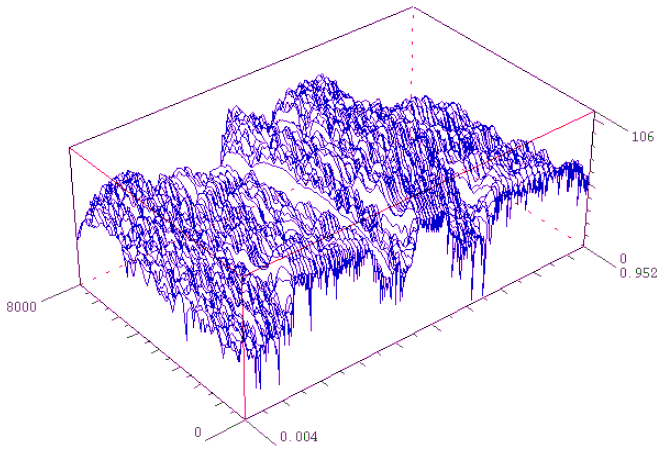
b


200
150
100
50
0
F0
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90

c


100
80
60
40
20
0
2000
4000
6000

d



e

Figure 4.11 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.

4.2.1.2 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."* (Wikipedia: Speech Recognition) 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 [PSUNautilus] 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 require 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

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.

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 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) r(Wikipedia: Speech Recognition). RTF is measured as the ratio of the time it takes to process the input (P) over the duration of the input (I):

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

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

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:

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

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.

4.2.3 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

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?

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.

4.2.3.1 Context

Understanding the context of a given dialogue input (whether it is from SR or text inputs) 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 on 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 it 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 (Wallace). 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.

4.2.3.2 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 the next chapter.

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 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 as 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 inputs and use the information to select motions, the motion modifiers, and the parameter values for the motion modifiers. In our system, cognition is still very barebone – that is, the decisions that will be made by the system (robot) are hard-coded, and the system is not modeled after any particular models of cognition. Instead of deriving implicit 'meanings' from the input values, our cognition system will take the input as a direct command, and any values associated with it will directly affect the execution of the command. So for example, if a sentence entered by the user (either text or speech) contains a keyword that is a known command for the robot to execute the movement, say, “waving right arm”, the robot will

immediately execute the movement “waving right arm” without trying to derive the context of the sentence.

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.

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*. 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 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.

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).

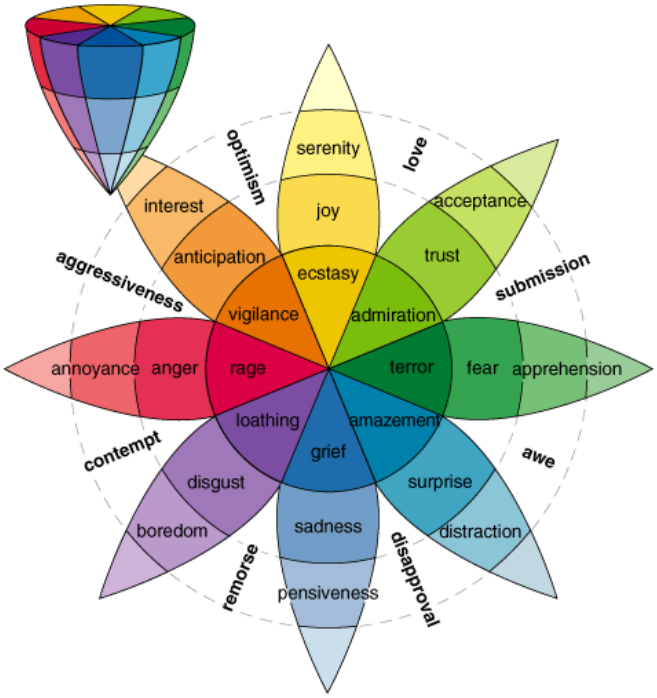


Figure 5.1 Plutchik Emotional Model

(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 (Nettle 2007). 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* (Wikipedia, Big Five). 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).

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?”

CHAPTER 6 – RESPONSE SYSTEM

The Response System is mainly responsible (pun not intended) 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 e the response 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 how to execute the motion has already been done by the Cognition System. Thus, the Cognition System outputs: 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.

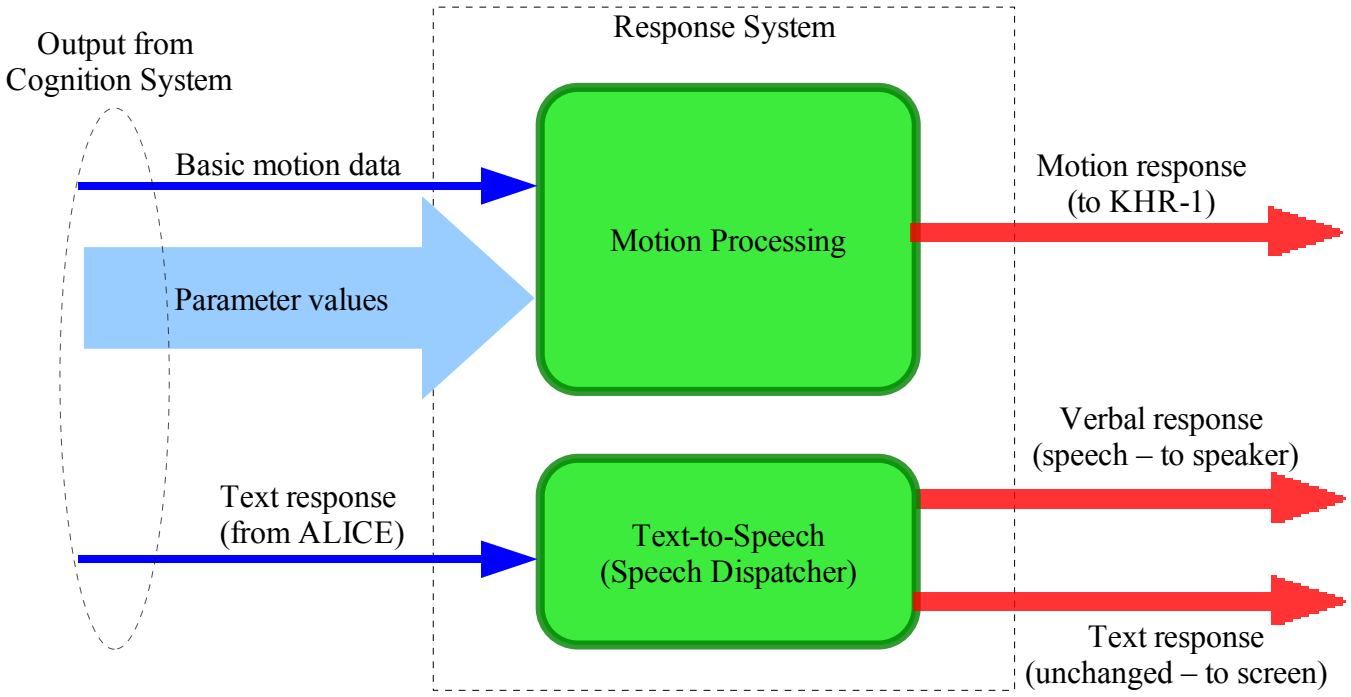


Figure 6.1 Input and output of the Response System

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 changes 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 controls 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 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 has only a range of 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 a industrial robot arm, the global origin often is 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}.

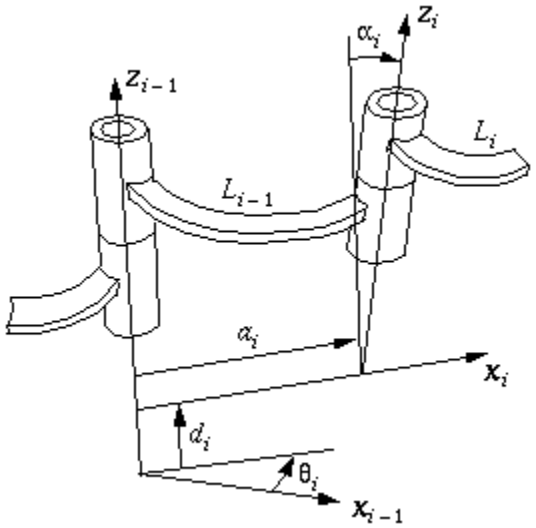


Figure 7.1 D-H Notation (image taken from: http://www.cs.cmu.edu/~rapidproto/mechanisms/chpt4.html)

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:

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

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

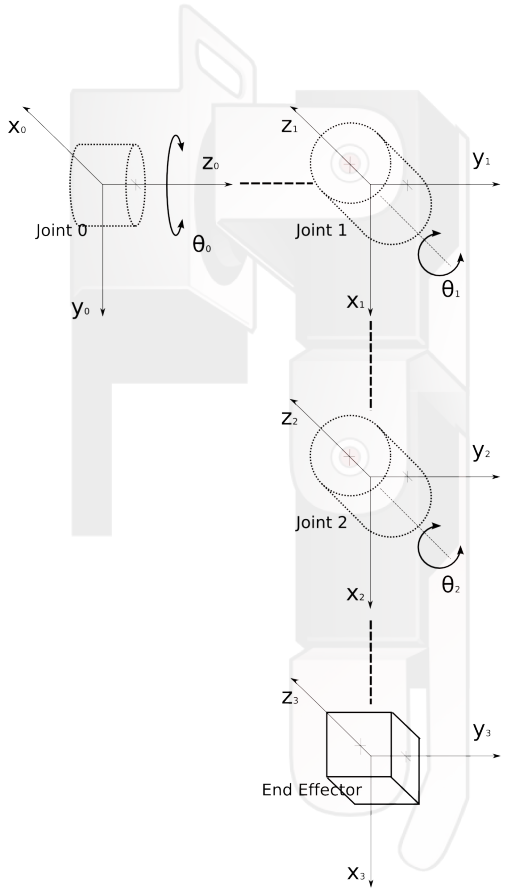


Figure 7.2 D-H Notation example using KHR-1 arm kinematics (not to scale).

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 α2 = α3 = 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 ${}^0T_1$ is:

$$ {}^0T_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1\cos\alpha_1 & \sin\theta_1\sin\alpha_1 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1\cos\alpha_1 & -\cos\theta_1\sin\alpha_1 & a_1\sin\theta_1 \\ 0 & \sin\alpha_1 & \cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0T_1 = \begin{bmatrix} \cos(90) & -\sin(90)\cos(90) & \sin(90)\sin(90) & 0.\cos(90) \\ \sin(90) & \cos(90)\cos(90) & -\cos(90)\sin(90) & 0.\sin(90) \\ 0 & \sin(90) & \cos(90) & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0T_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Similarly, we can obtain ${}^1T_2$ and ${}^2T_3$:

$$ {}^1T_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2T_3 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

And the Total Transformation matrix is ${}^0T_3$ is:

$$ {}^0T_3 = {}^0T_1 . {}^1T_2 . {}^2T_3 $$

$$ {}^0T_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The Total Transformation matrix ${}^0T_3$ 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:

$$ \boldsymbol{P} = {}^0T_3 . \boldsymbol{P_3} $$

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

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

The transformation can be verified with the following illustrations:

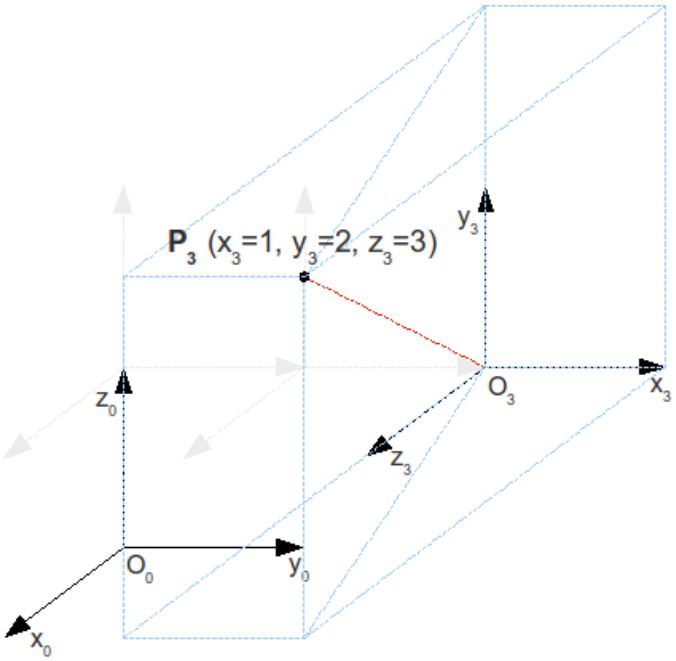


Figure 7.3 Point $\boldsymbol{P_3}$ *with respect to coordinate frame 3* (x_3, y_3, z_3)

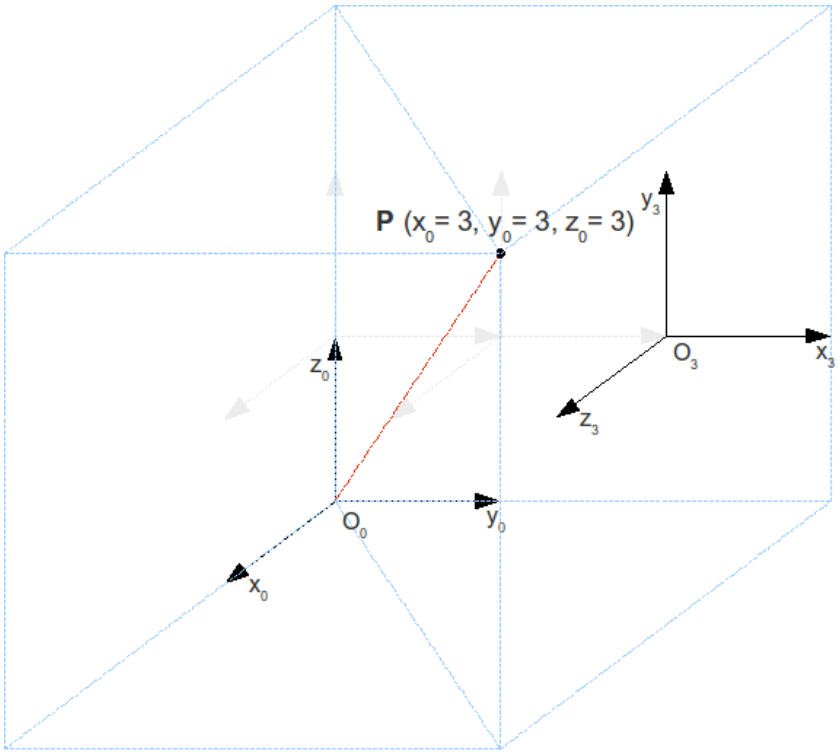


Figure 7.4 Point $\boldsymbol{P_3}$ *(here* $\boldsymbol{P}$*) with respect to coordinate frame 0* (x_0, y_0, z_0)

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 with the start position of the end effector. The total transformations by the joint angles is described mathematically by:

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

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

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:

$$P_{end} = {}^{0}T_{n} P_{Start}$$

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.

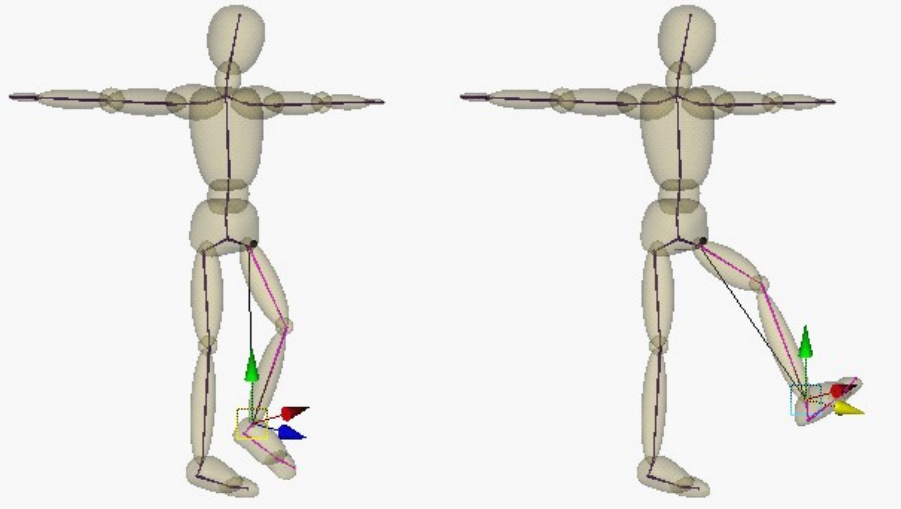
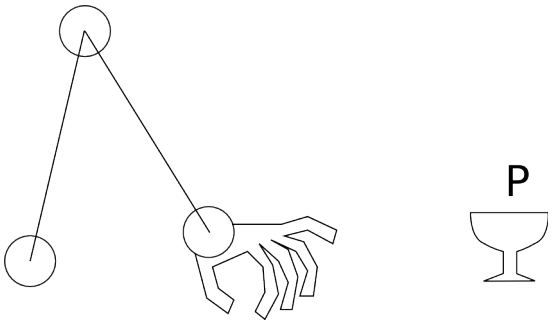


Figure 7.5 IK example in Maya 3-D animation software (image from: http://caad.arch.ethz.ch/info/maya/manual/UserGuide/CharSetup/images/SkeletonPose.fm.anc3.gif)

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

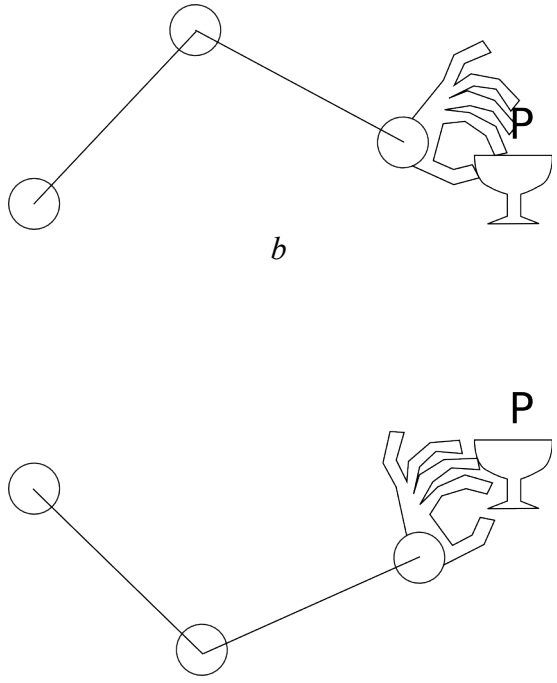


Figure 7.5 (a) IK problem, (b) and (c) possible solutions

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.

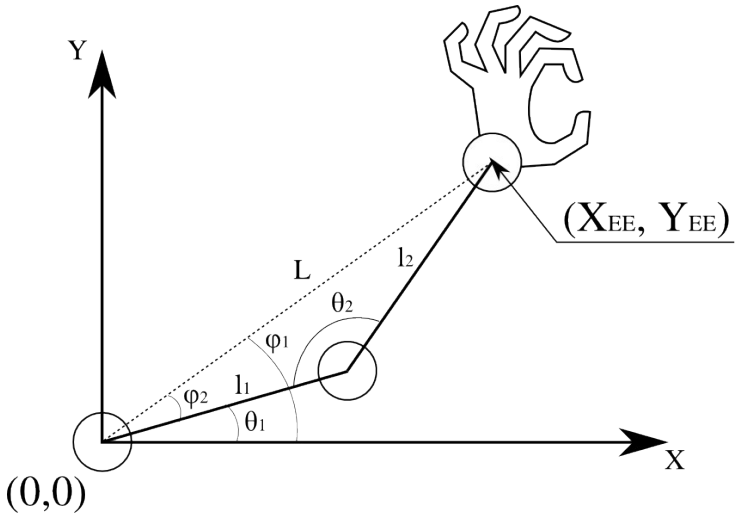


Figure 7.6 IK Example

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

$$L^2 = X_{EE}{}^2 + Y_{EE}{}^2$$
$$\psi_1 = \tan^{-1}\left(\frac{Y_{EE}}{X_{EE}}\right)$$

By virtue of the law of Cosine, where:

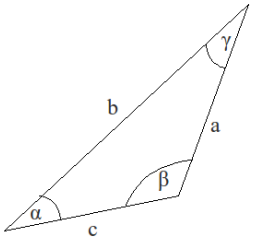


Figure 7.7 a general triangle

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

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:

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

And so, if (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 $\theta_1 = \psi_1 + \psi_2 = 81.97^\circ$ is also a feasible solution (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 humanoid robots. Next, as our interest is on humanoid robots, we want to be able to make the movements able to express human emotions. For this purpose, we take inspirations from performing art. In this chapter, we discuss two prominent guidelines that are popularly used by artists to create emotionally-expressive movements: Laban Movement Analysis and 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 who is a pioneer of modern dance in Europe and in movement study. It 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 (Hamburg 1995), and physical therapies. Laban's work was then extended by Irmgard Bartenieff who introduced Labanotation.

Recently, there has been an increasing number of researches in applying LMA in HRI for socially interactive robots. It is being used to recognize human gestures to infer the meaning behind the gesture (i.e.: context) for the robot (Zhao 2001), and also as parameters to generate gestures (Chi et al. 2000). 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, and we believe if we can develop the computational models for them, we would be able to generate movements, and hence, gestures 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, Dias) used LMA for *recognizing* gestures, instead of motion generation. They also 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. 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.

In (Zhao 2001), 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 (Wikipedia: Laban Movement Analysis).

- *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.

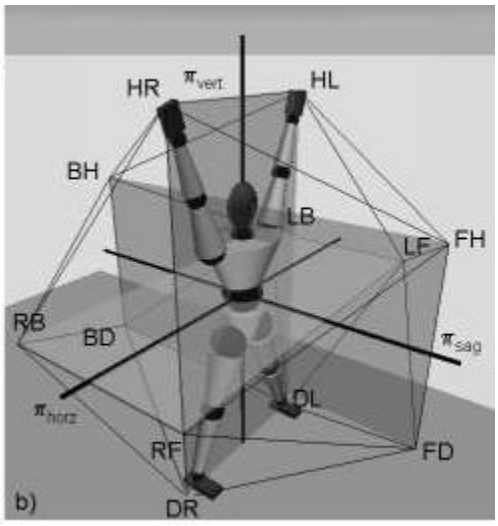


Figure 8.1. Kinesphere. The icosahedron is constructed by the Horizontal plane (RB-RF-LF-LB), vertical plane (HR-HL-DL-DR), and sagittal plane (FH-FD-BD-BH).

- *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 (Bishko 1992): the movement is directed to a certain target but requires some extra movement, e.g.: swatting a fly – require 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. (Bishko 2007) 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* (Bishko 2007). 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. This perhaps because we see humans through this view, that humans always move intentionally (Zhao 2001), 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 (Bishko 2007). The following relationships of the ego functions with the Effort qualities are described in (Bishko 1992).

- 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 sense and react 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 (Bishko 1992))
- 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. 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)

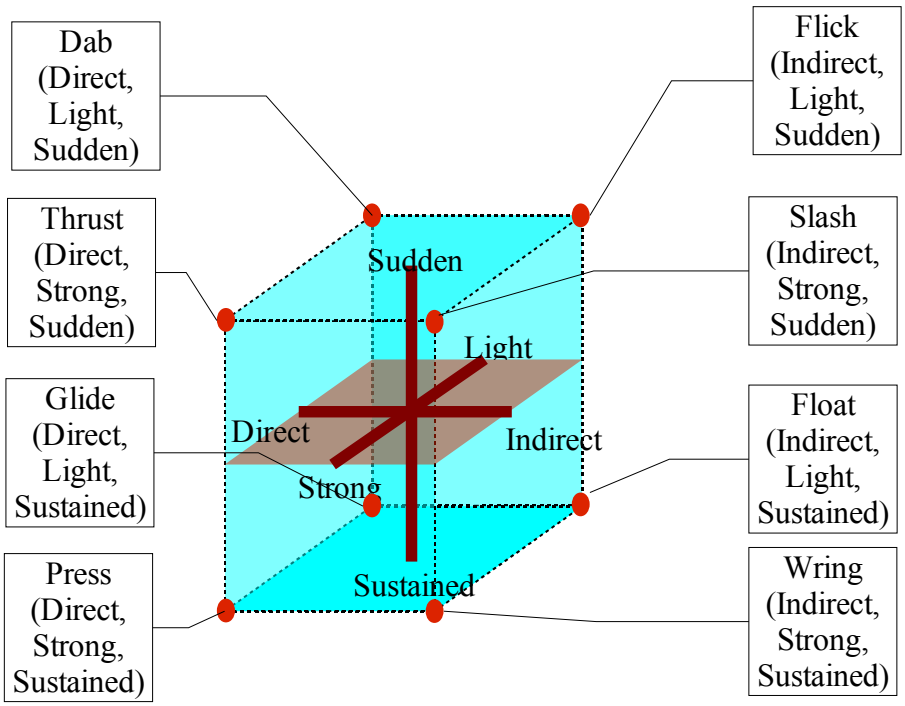


Figure 8.2. Basic Effort Actions

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, (Bishko 1992) 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 (Bishko 1992). They can summarize the desired movement quality by drawing parts of the graph that represent the Effort qualities that they want.

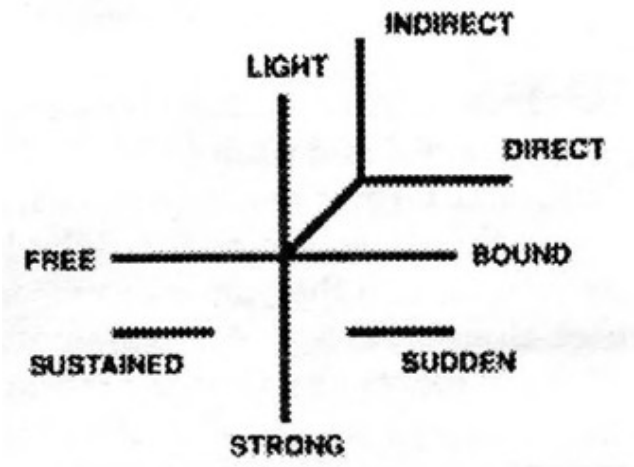


Figure 8.3. Effort Graph

The following table 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.

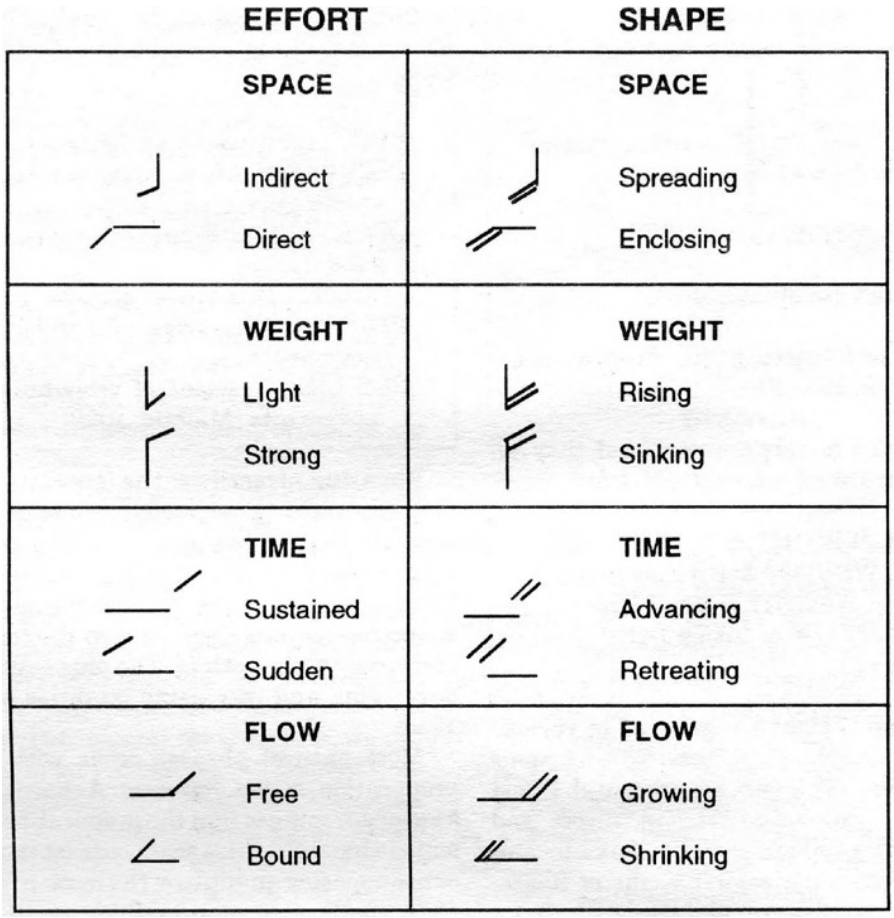


Figure 8.4. Shorthand of the Effort graph.

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 (Bishko 1992), 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. On our humanoid robot, the KHR-1, Effort qualities become much more complicated to realize where the control for the servo motor is very limited.

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) can 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 input 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 students refer 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 analysis of movements, while the Disney animation principles as 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 [Lasseter], 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. So, we will only consider 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. which will serve as a reference to our humanoid robot KHR-1.

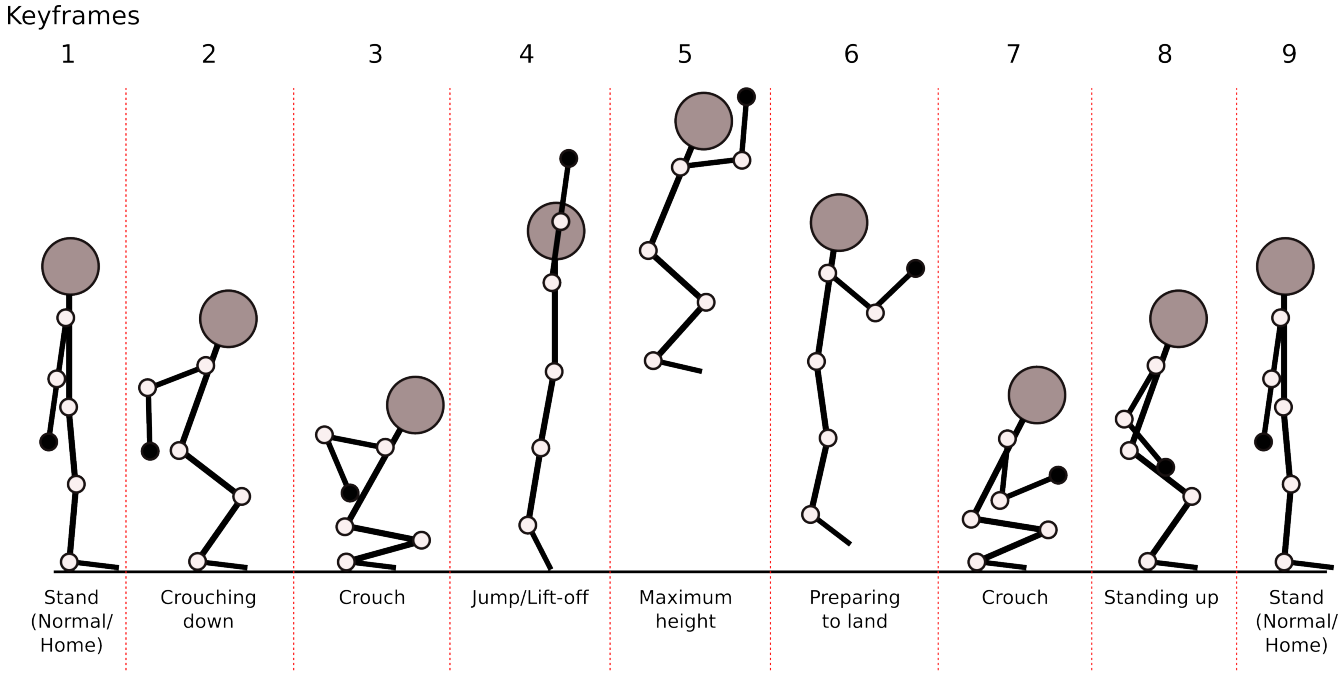


Figure 8.5 Jumping action in nine sequences (keyframes).

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 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, Anticipation is in the first three sequences/keyframes. Anticipation is directly related to the 'Preparation' of Phrasing in LMA.

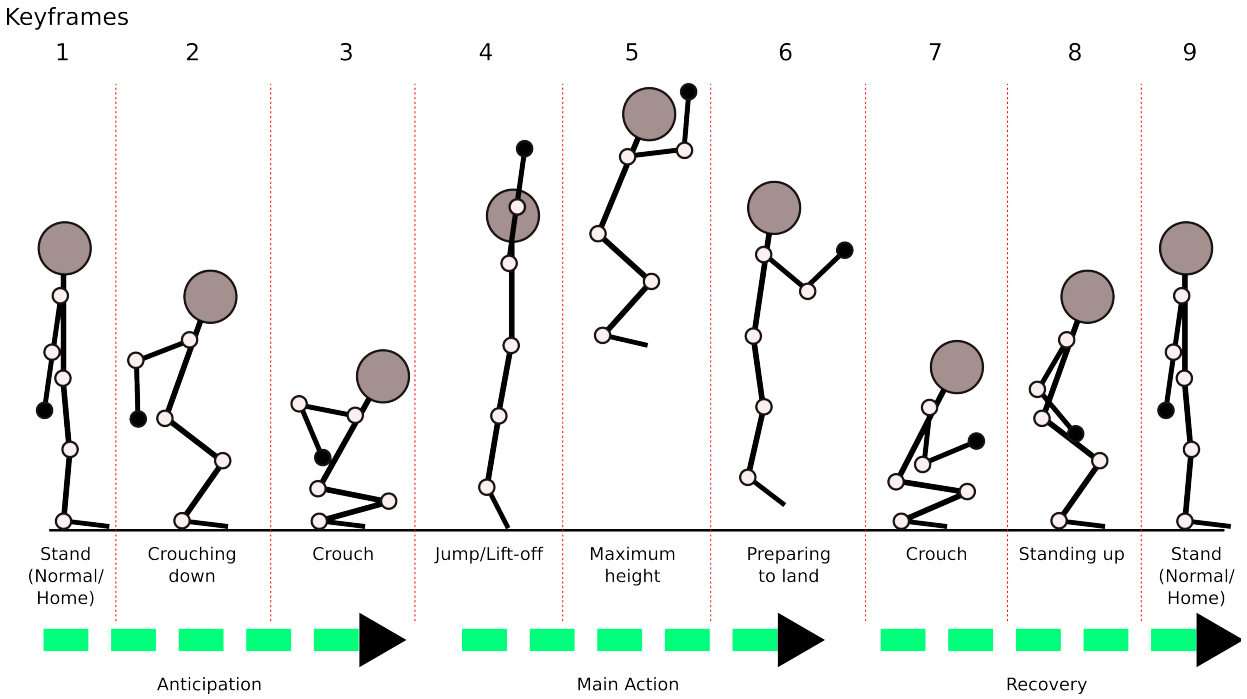


Figure 8.6 Anticipation

8.2.3 Stretch and Squash

Stretch and Squash refers to the elastic qualities of organic creatures. This principle is tightly related to 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 displays 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.

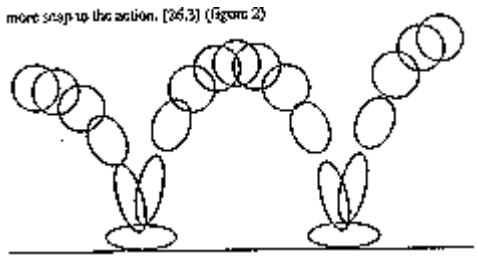


Figure 8.7 Stretch and squash in bouncing ball

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 is important to bring

life to the object. In our jumping example, the Squashes are seen at sequence/keyframes 3, 5, and 7. Stretches are seen at keyframes 4 and 6.

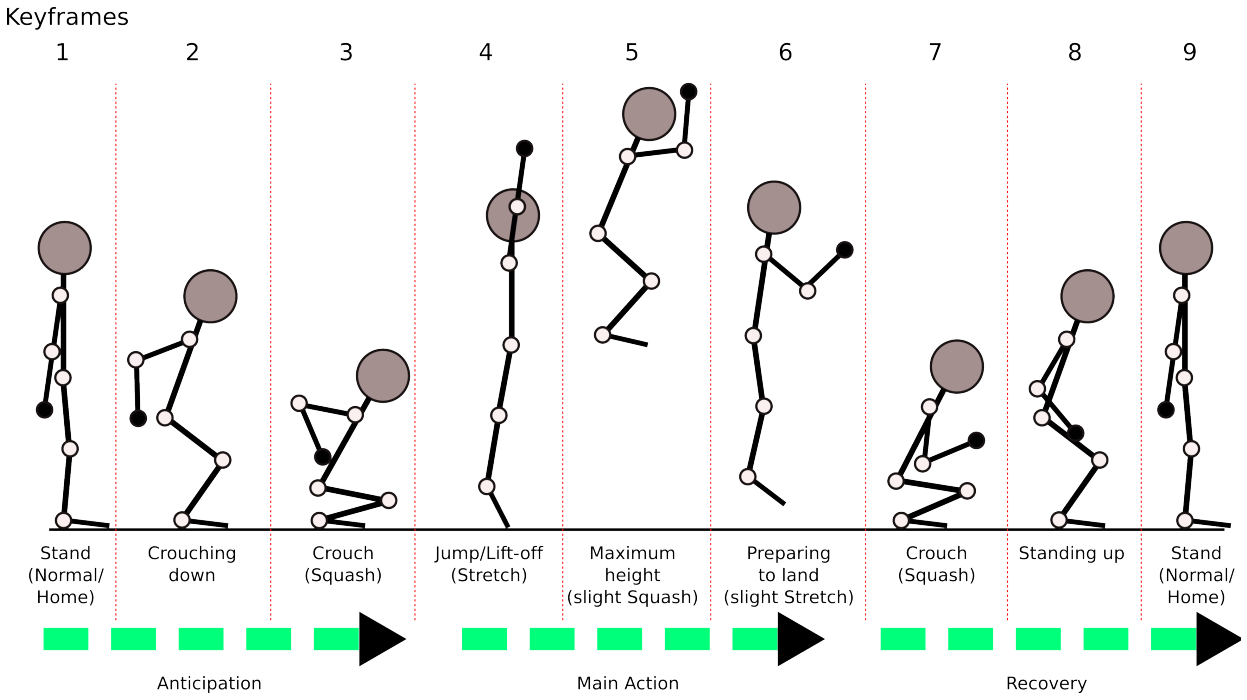


Figure 8.8 Stretch and Squash in jumping example.

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), in general, this last animation is not as 'natural' nor appealing as the one with seamless transition. (see Motion Blending).

In our jumping example, 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.

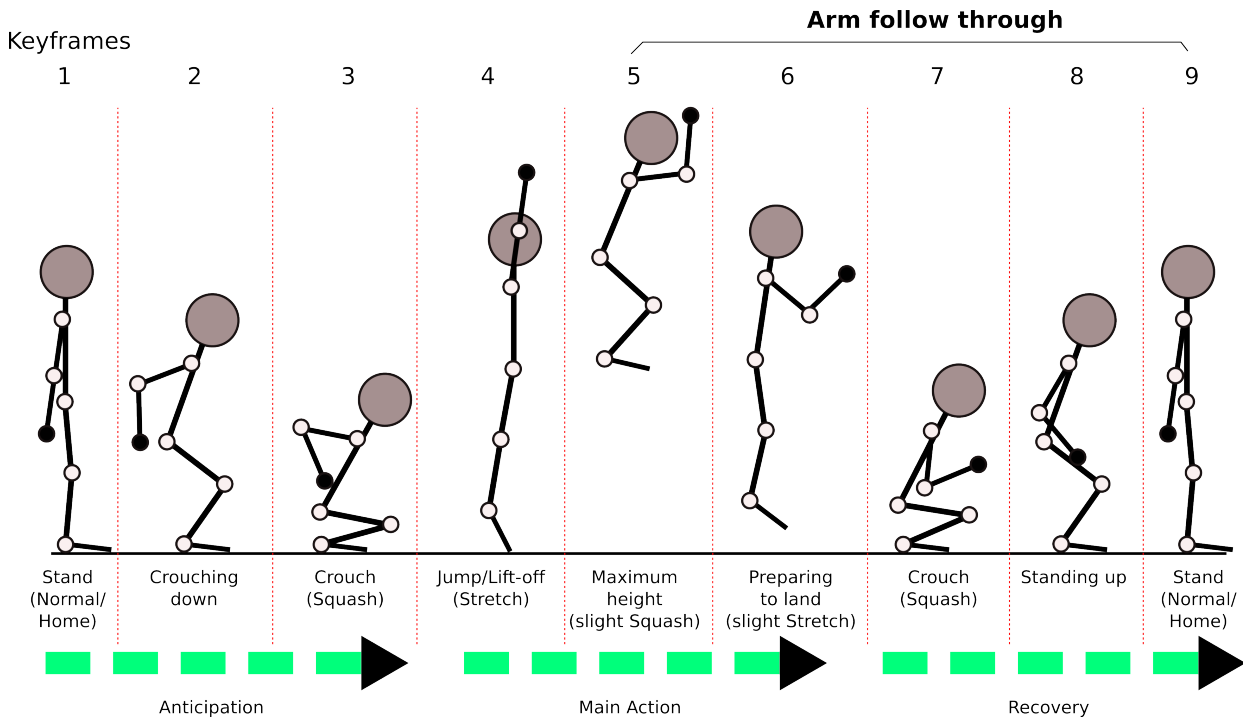
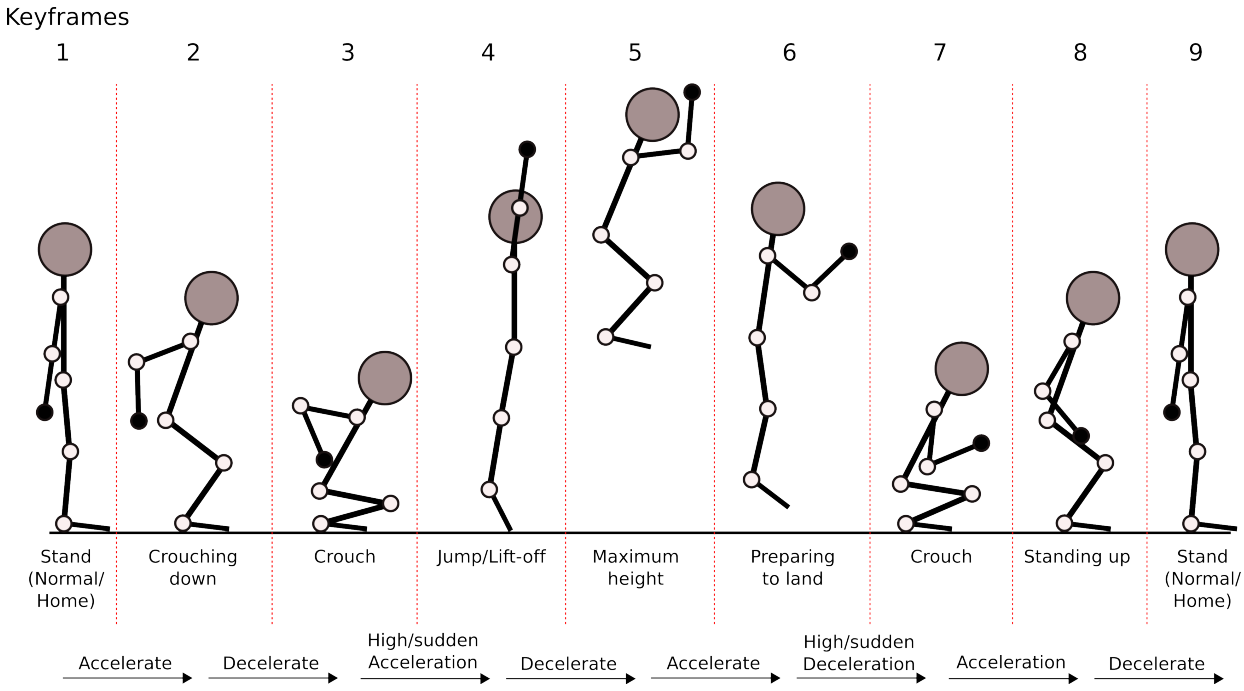


Figure 8.9 Follow through in jumping example.

8.2.5 Timing

Timing refers to the speed of the action; whether it is the anticipation, followthrough, or the main action. [Lasseter] 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. 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 potential energy at keyframe 5. As the body starts to fall back down, from keyframe 5 to keyframe 6 there is acceleration because of gravity. And so forth.



8.10 Timing of acceleration and deceleration

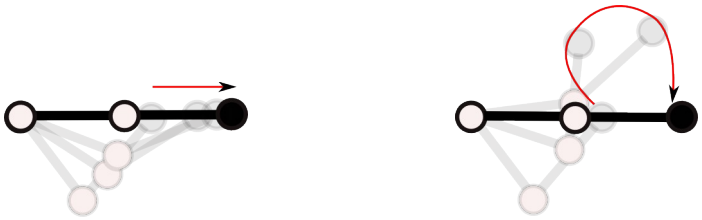
8.2.6 Slow-in/Slow-out

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 our discussion about Timing above.

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.#a) is the movement of the end effector (hand) in a straight line. The second example (figure 8.# 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 like in a dance.



a *b*

Figure 8.11 Arcs example on KHR-1 arm: (a) the end effector (filled node) is moving in a linear path, (b) the end effector is moving in an arc.

8.2.8 Exaggeration

Exaggeration refers to creating an emphasis to, or accentuate the actions, or personality of a character. It essentially adds a "more" aspect to the action or personality traits; scared to terrified, sad to sobbing, surprised to shocked, strong punch to powerful punch, and so on. This is related to presenting an idea as clear as possible, as in Staging. In 2D or 3D animation, Exaggeration does not imply to simply deform the object to the extremes (with Stretch and Squash) that makes the object appear "cartoon-like", or comical, instead it is used carefully to make the whole scene seems 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. The exaggerated pose also gives the indication that the jump is going to be more powerful.

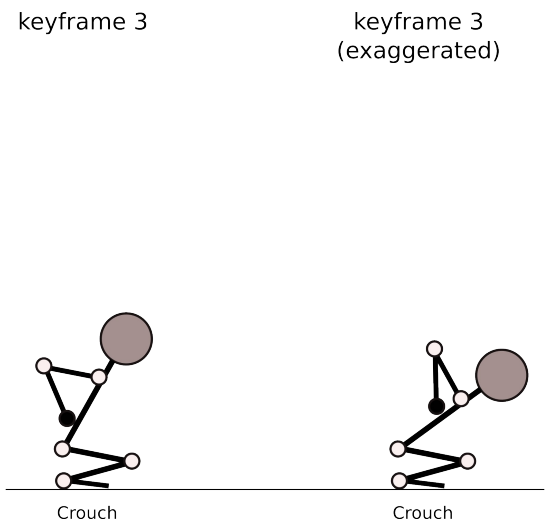


Figure 8.12 Exaggeration on keyframe 3 (crouching)

8.2.9 Secondary Action

Secondary Action refers to the actions that supports 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 depends on nature, or are side effects of a 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 do 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 becomes 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.

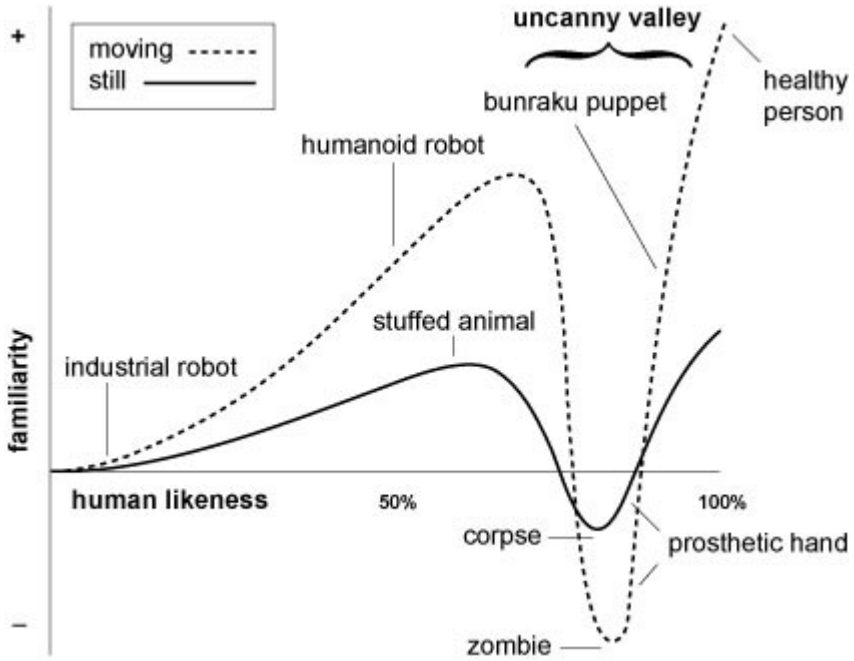
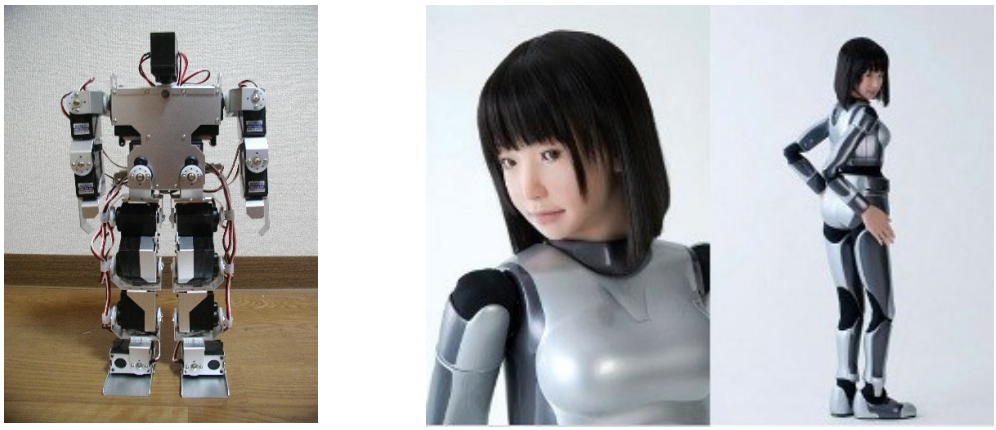


Figure 8.13 Mori's Uncanny Valley

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. ~~In figure 9.1 we show~~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.

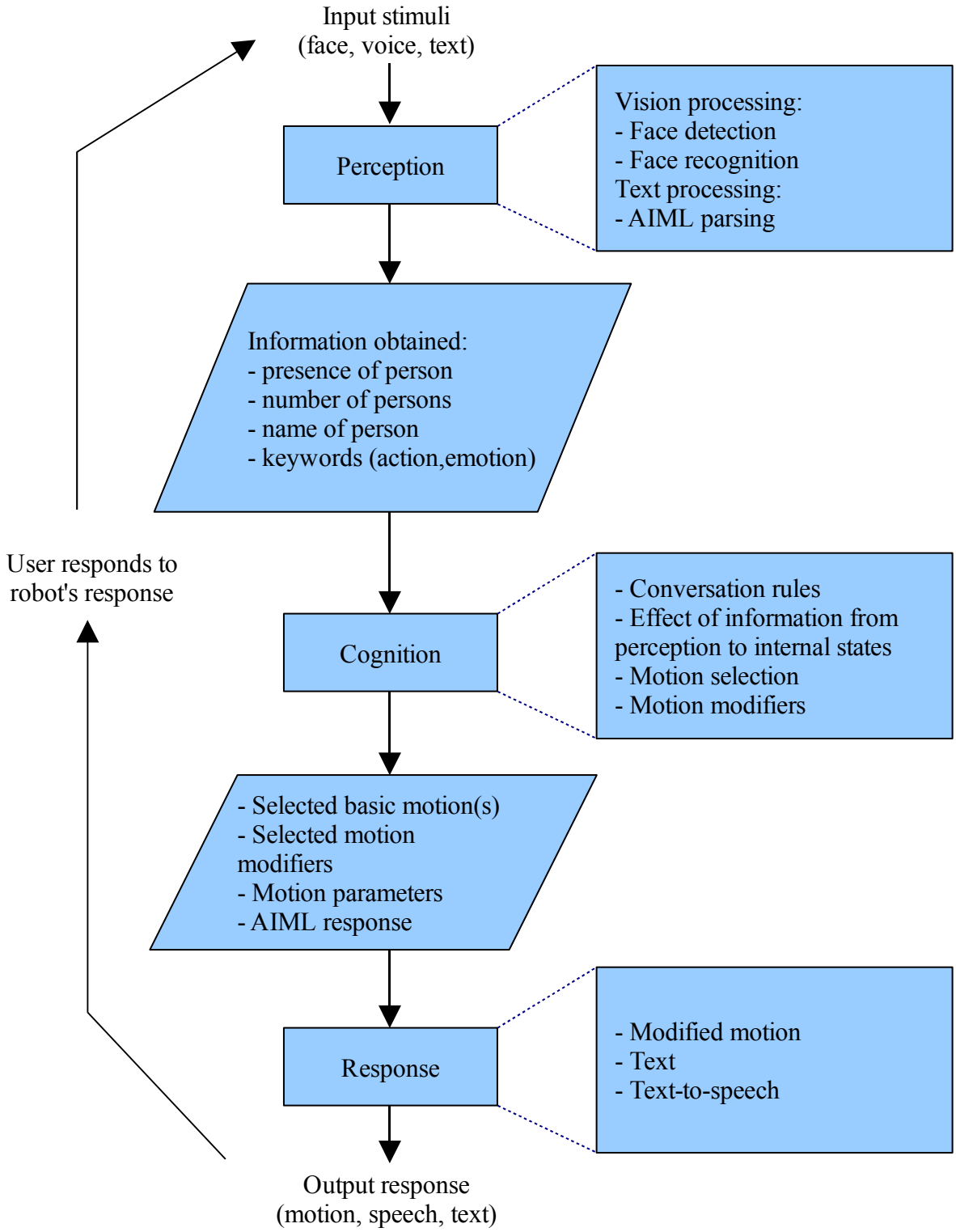


Figure 9.1. Information flow and processes in the system.

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.

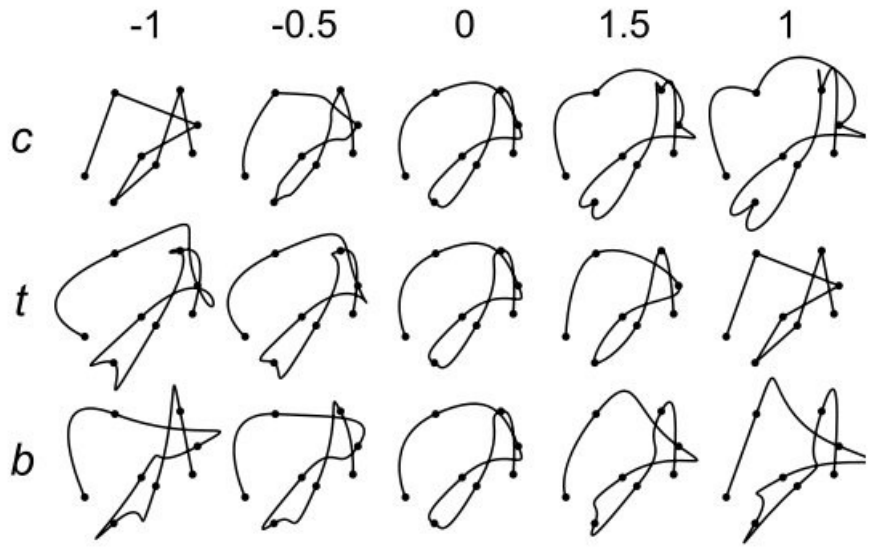


Figure 9.2 Effects of continuity (c), tension (t), and bias (b) parameters on Kochanek-Bartels splines (source: http://en.wikipedia.org/wiki/Kochanek-Bartels_spline).

The tangents T1 (incoming tangent) and T2 (outgoing tangent) can be calculated as follows (Wikipedia: Kochanek-Bartels Spline):

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

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

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:

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

Where s is the value [0,1] specifying the interpolation 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:

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

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.

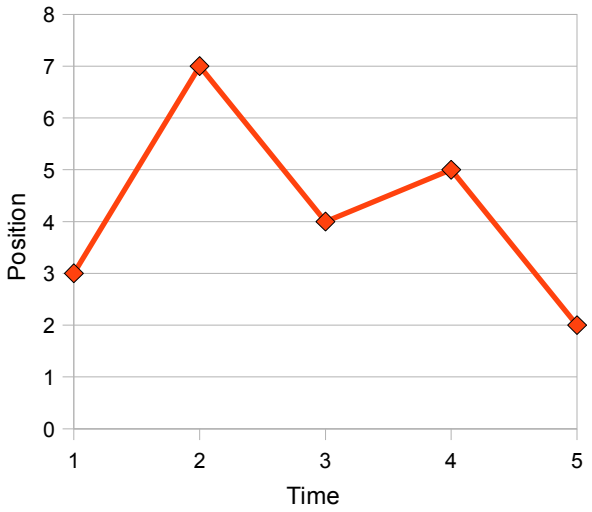


Figure 9.3 Linear motion signal.

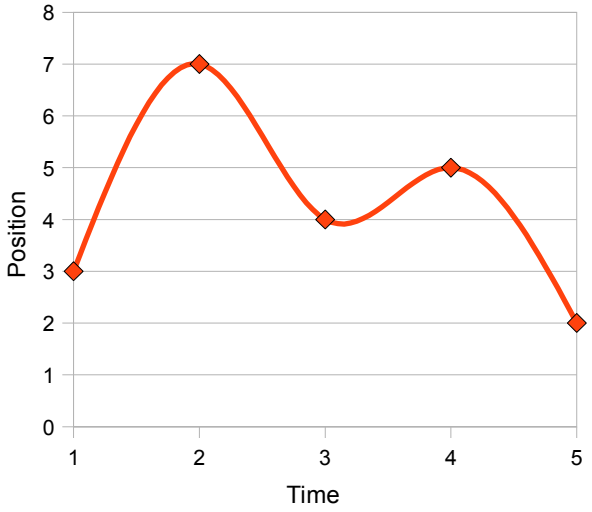


Figure 9.4 Non-linear motion signal.

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.

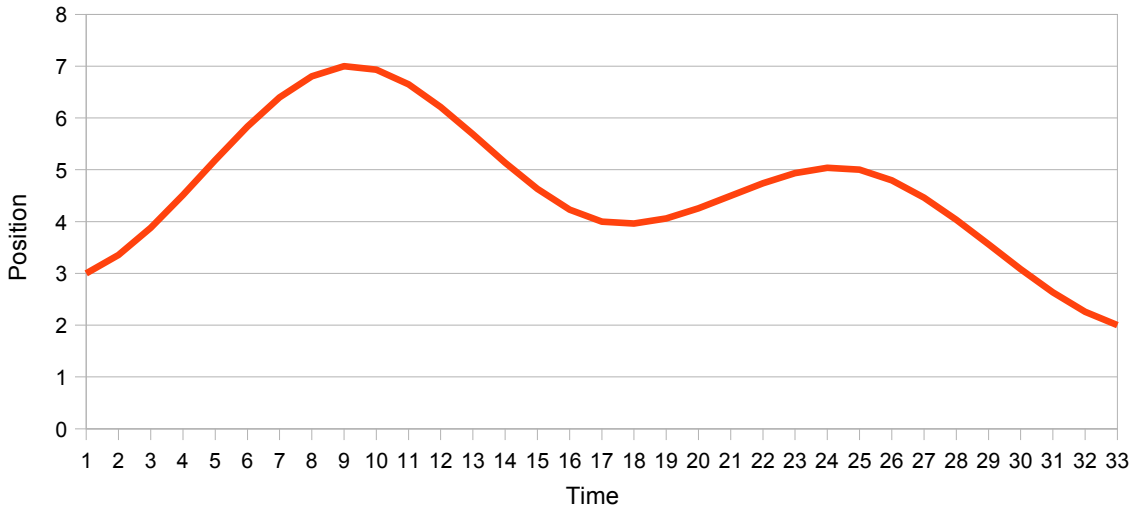


Figure 9.5 Interpolated (elongated) data set (continuity=0, tension=0, bias=0).

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).

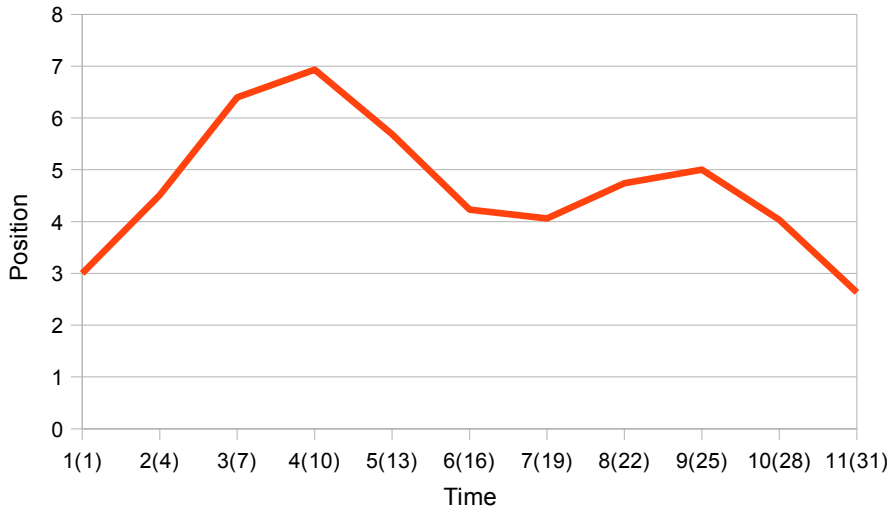


Figure 9.6 Resampled data set with r=3. *Numbers in parantheses indicate the index numbers of the sampled elements from the interpolated data set.*

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 (Bruderlin, Williams 1995, Huang, Boulic and Thalmann 1999). 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 (Bruderlin, Williams 1995). 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

~~*Figure 9.1. Information flow and processes in the system.*~~

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

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. Thus, we take cues from DAP and LMA for the relationship between those motion properties, and emotional expressions in a motion. Specifically, we

want to focus on the most prominent motion qualities such as: motion dynamics, motion transitions, Effort (from LMA), Shape (LMA), Phrasing (LMA), Anticipation (from DAP), Stretch and Squash (DAP), Follow through and Overlap (DAP), Slow-in/Slow-out (DAP), and Arcs (DAP).

9.1.1.1 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.

9.1.1.2~~1~~ Motion Dynamics

DAP and LMA 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~~is~~ 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, s~~—~~uch 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.

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, 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. 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, which we already discussed above.

~~***9.1.1.2 Active Motions***~~

~~**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, 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. 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, which we already discussed above.**~~

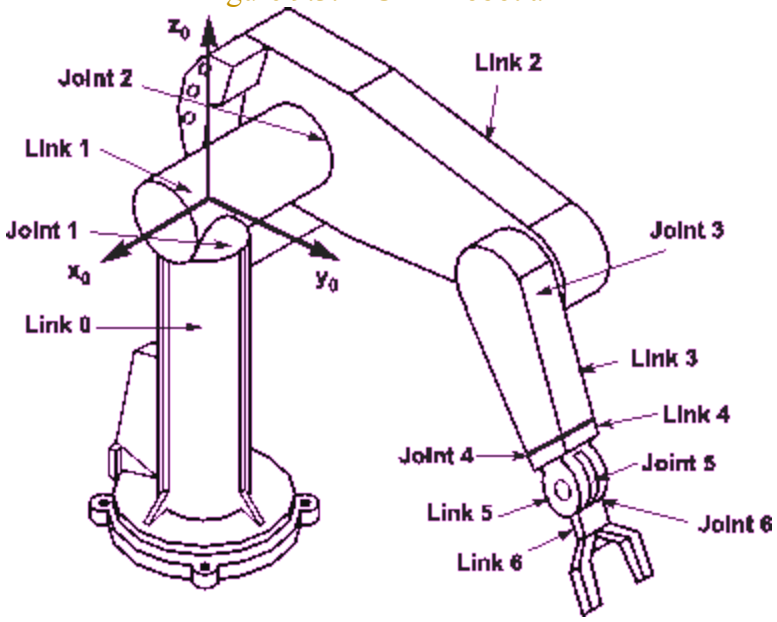
The Flow Effort of LMA is understood as the progression of one movement to the next. Therefore, we exhibit a 'bound' Flow by abrupt transition between movements, and 'free' Flow by smooth transition

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) movements to be executed, and then performing interpolation on the concatenated movement. 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.

~~*9.2.1 The Classical Robot Motion*~~

~~Traditionally, the motions of an anthropomorphic robot is created through *kinematics*. There are two approaches: *forward kinematics (FK)*, and *inverse kinematics (IK)*. Creating a motion through FK is rather straightforward. For example, given the PUMA 562 robotic arm (see Figure 9.3), with two degrees of freedom (DOF) at the base (Joint 1 and 2), one DOF at the "elbow" (Joint 3), and three DOF at the "wrist" (end-effector – not counting the "gripping action", Joint 4, 5, and 6). Using FK, given the angles of each joint, we can find out the position of the end-effector (usually with respect to the location of the base). Conversely, using IK, we give a *target position* of the end-effector, and we (or the system) must find out what are the angles of each joint. IK is considered a highly complex problem, as with more DOF, the more possible solutions there are (counter-intuitively, this is actually more difficult to do than it sounds).~~

Figure 9.3. PUMA robot arm

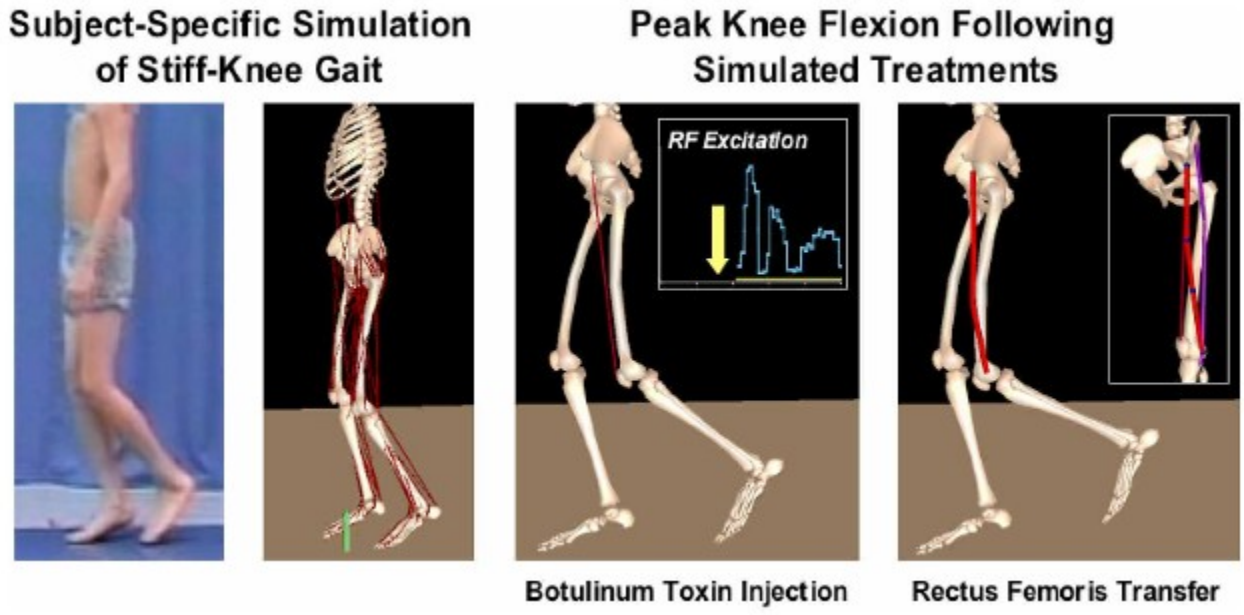


The kinematics approach is not obsolete, of course, it is just at a very *low level*. That is, because it involves individually specifying each joint angle. Moreover, the definition of "kinematics" itself refer to the study of motion of objects without consideration of the dynamics of the motion [6-1]. Pure kinematics for creating motion works well for industrial robots, but not enough for humanoid robots. This is because humanoid/anthropomorphic robots are designed and *expected* to move like, well, *humans*. Humans and other organic creatures have the unique ability to move very efficiently, taking advantage and managing the dynamics in their movements. This ability is what differentiates a 'natural' motion from a 'robotic' one.

9.2.2 Natural Motion

There are numerous researches in computer animation field which try to incorporate dynamics into the animation. This is done by modeling the dynamics of the objects through some formulas/equations,

and simulating the forces in the environment. Given that the dynamics models (we will refer to it as *model* from here on) are accurate, the objects now can react to the forces in a natural-looking way (i.e. motion). These approaches have been proven to be very useful in performing pre-surgery analysis in orthopedics[6-2], and most recently, video games[6-3].



a.

b.

Figure 9.4. a) OpenSim motion analysis and simulation, b) NaturalMotion dynamic motion synthesis

Note that both approaches rely on the accuracy on the model; an inaccurate model may give some weird – but interesting – results. In OpenSim (Figure 9.4 a), the model is usually acquired using

motion capture data. An additional software package (e.g. SIMM [6-4]) can extract the model from motion capture data. This model then can be used by OpenSim to execute the simulations.

Interestingly, these sophisticated approach only address the *reactive* issues of the motion. Or, more specifically, how the body reacts to the forces that affects it. Initiating a motion is a different issue.

9.2.3 Expressive Motion

While kinematics provide a way to mathematically describe the position and orientation of the body at particular points in time for the duration of the motion, and dynamics simulation provides 'naturalness' of the body (and reflected in the reactive motion), the remaining issue is the *initiation* of the motion. Bishko (Bishko 1992) indicates that humans initiate motions as an expression of their mental drive, and it gives "life" to the motion. The same goes for animation; for it to be believable, a motion needs to have some mental impulses behind it. We apply this concept to the animation of our humanoid robot.

(Bishko 1992) suggested the use of Laban Movement Analysis (LMA) theory as a base to create expressive motions. LMA encoded the complex association between the mental component of a motion and the expression of that component in the motion itself. Basically, how a mental state is expressed in a motion in general. For example, LMA associates a 'weight' of a motion, which could be 'strong' or 'light', to the intention of the actor (in this case, our robot). A 'light' motion indicates that the actor is sensitive or delicate, while 'strong' motion gives the sensation of 'forceful' or the determination of the actor.

Speaking of animation, it is almost impossible not to mention the animation principles developed by the Walt Disney animation studios. The Disney animation principles (DAP) are guidelines for

animators to pay attention to the elements of motion that must exist in the animation to create the 'illusion of life' (Lasseter 1987). For example, 'anticipation' refers to the preparation action before the main action, such as crouching before a jump. 'Stretch and squash' refers to the absorbtion and release of energy: 'squash' is building potential energy, 'stretch' is the release of energy or transformation from potential to kinetic energy. Often, the 'stretch and squash' quality also gives an 'organic' sensation to the object, as organic objects tends to be 'squishy' (ew). And so on.

9.2.4 Motion as Signals

Bruderlin and Williams (Bruderlin, Williams 1995) introduced the concept of representing motions as signals. It was actually obvious that motion data can easily be represented as signals (time/frames on x-axis, angles on y-axis). However, they showed that using common signal processing methods, a motion can be quickly modified in many different ways, or give interesting effects. For example, by giving negative value to the low frequency gains and increasing the high frequency gains, a knocking motion becomes exaggerated on anticipation and follow-through (Figure 9.5 bottom).

Moreover, treating motions as signals provides some elegant ways to manipulate motions. Two motions can be combined into one using *dynamic timewarping*, with the resulting motion still have identifiable characteristics of the original motions. Applying a shaping function to the motion using *waveshaping* gives the ability to modify the motion globally, for example: adding undulations at the beginning, middle, end, or anywhere in between the motion. *Motion displacement mapping* allows local modifications to a motion, while keeping the continuity of the whole motion. And so on.

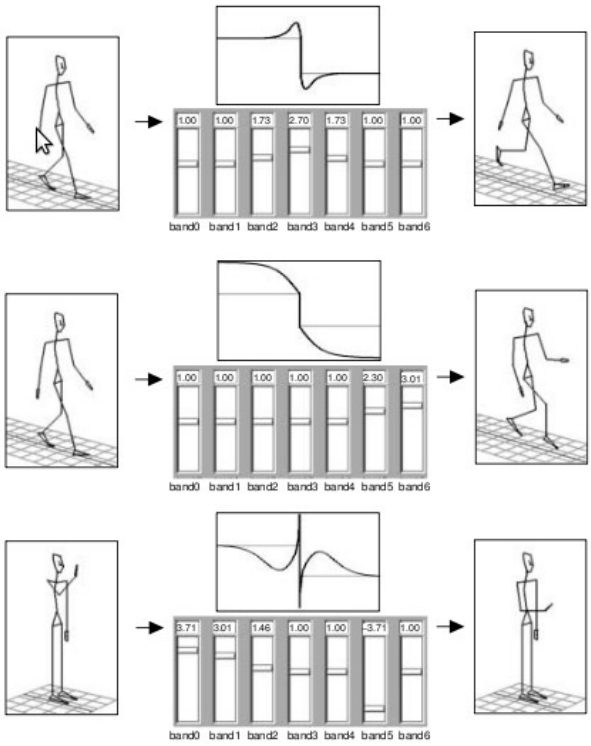


Figure 9.5. Gain adjustments affects motions. (Comment: "give more explanation")

This method provides another layer of abstraction between the very high-level representations of DAP and LMA, and the low-level representation of joint angles and kinematics. Thus, it is possible to specify what kind of DAP or LMA component translate to what kind of signal parameters – treating them as controls for the motion signal.

9.3 Naturally Expressive or Expressively Natural Motions?

We believe that DAP actually encodes the dynamics of motion and expression of the mental component in a motion. Essentially, we are saying that DAP is a general concept that covers the dynamics of motion and the concepts covered by LMA, albeit at a very high level. This is because everything is still up to the intuition of the animator on how to execute (i.e. animate) the motion. In contrast, dynamics simulation models *exactly* (to some degree) of what would happen if a force is acting on the

~~object; animators can only imagine and position (or draw) the objects according to his/her imagination. LMA is also a general concept, while it is a bit more precise than DAP since LMA provides certain parameters to describe a motion.~~

~~Chi et. al. (Chi et al. 2000) created a computational model using LMA parameters as inputs called *EMOTE* to modify ready-made animations. The EMOTE model applies the *Effort* and *Shape* components of LMA to the torso and arms of a computer-generated 3D character. It is quite an elaborate empirical model, built with the help of a certified LMA practitioner. Many of the coefficients in the model are obtained through trial-and-error and observations. So far we have not found a model for DAP.~~

~~To be fair, DAP was developed originally for hand-drawn animations. Some of its principles such as 'solid drawing' cannot be directly applied to other types of animation, such as 3D animation (Lasseter 1987). But, most of them are still useful, and have been referenced many times in non-DAP techniques [6-5, 6-6, 6-7, 6-8].~~

~~---------~~

~~Our intention for this system is so it can be automated. That is, the system will take inputs from its sensors, perceiving human communication modalities such as speech, gestures into certain values, and interpret them as inputs for the animation generator parameters, and the generator creates animation based on those values. Thus, a context-specific behavior of the robot.~~

EMOTE – Effort Parameters

With the help of a certified LMA practitioner [2], the EMOTE system models the instructions for the dynamics of a motion by explicitly implement the LMA Effort parameters. The parameters are: *Weight, Space, Time,* and *Flow*[13]. What is nice about these Effort parameters is that they consist of two extremes (1-dimensional data), such that they can be easily used for computation as ranges. In EMOTE each of them are in the range of [-1, +1] (similar to the Shape parameters). However, their implementation in EMOTE involves several *intermediate variables*. Unlike in Shape, where the parameters directly affect the angles as in equation (2.3), Effort parameters do not directly affect rotations or positions (low-level motion elements).

Instead, the Effort parameters affect things such as *timing* or *speed* of the motion, and some interpolation parameters. This is natural, because that exactly what the Effort parameters are: high-level description of a motion. So everything is good.

The Effort parameters extremes are shown in Table 9.1. The assignment of which extreme to which value (-1 or +1) was determined arbitrarily. However, if we use some convention like *intensity* of the dynamics of the motion, we might assign the values like we did in Table 9.1. There, my assignment was based on that the words: Strong, Direct, Sudden, Bound motions describe motions that *feel* more intense than motions described by the words: Light, Indirect, Sustained, and Free. Of course, this is

13 The parameters will be referred using capital letters to differentiate them from normal words.

highly subjective, and other assignments might work as well (but the model may need to be updated accordingly).

Table 2.3. Effort parameters extremes

Parameter\Extremes	+1	-1
Weight	Strong	Light
Space	Direct	Indirect
Time	Sudden	Sustained
Flow	Bound	Free

Effort Model

In the EMOTE model, these parameters are intermingled and work together to affect the high-level elements of the motion. Before we get into a bit more depth on where the parameters play their part in the motion editing, let's see the general flow how Effort affect the motions.

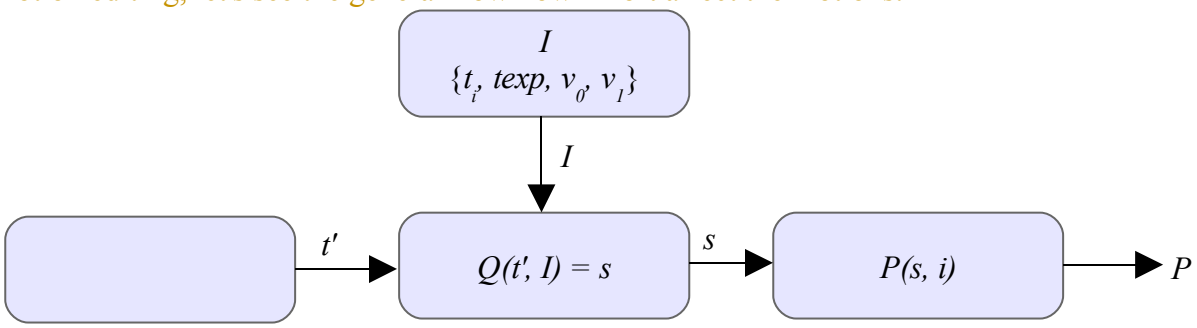


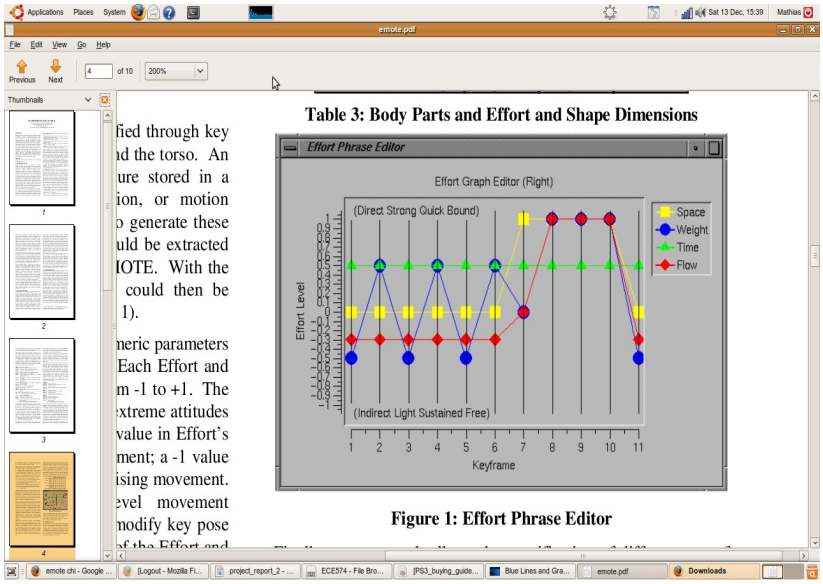
Figure 9.6. Effort model

As seen in Figure 9.6, *P* is the final output of the model, which is the new/modified position of a particular position of a motion. A motion is basically a series of positions. This implicitly say that when I refer to as a motion in these texts, I am talking about one degree of freedom (DOF, i.e. a rotational joint like elbow or knee which only moves in one axis[14]) – I will not refer a 'motion' as the combined movements of multiple DOFs, unless it is explicitly said so.

So a motion is a series of positions (in animation terms, also referred to as *frames*). When the Effort parameters are set, that is to say, I want to modify the motion, each position of the motion will be modified in some way. So the above model, is supposedly applied to each position of the motion and the system generates a series of new positions based on the original motion.

To make it easier, the EMOTE system allows applying Effort parameters only to *keypoints* (as in Shape above). Keypoints (or also referred to as *keyframes*[15]) are usually the points when there are some changes (direction, speed, etc.) in the motion. This is not always the case, though. If the motion is linear, one can always put keypoints anywhere in the motion. But any directional change in the motion (except motion in ares) will always be a keypoint. Note the distinction.

14 A freely rotating joint like the shoulder are often (and better) represented as having 3 DOFs as it can rotate in three axes.
15 The terms *frames* and *keyframes* will be used interchangibly with *positions/points,* and *keypoints,* respectively.



~~Figure 9.7 EMOTE Effort Phrase Editor~~

~~(Keyframe = Keypoints, Quick = Sudden)~~

~~A Note on Keypoints~~

~~Understanding the difference between keyframes and normal frames is important because between keyframes there are *arbitrary number of frames*! Frames are always separated by equal distance (in time), but between keyframes, there could be different number of frames. Additionally, in animation there is a term for resolution (also referred to as *speed*) of the animation, and it is *frames per second* (fps). Exactly as the term implies, the resolution (and thus, the smoothness) of an animation is determined by the fps value – the higher it is, the smoother it is[16]. For example, a motion from point A to point B is made of 2 keyframes; the first keyframe is at point A, the second is at point B. Let's say that the our first motion duration is 3 seconds, and the resolution is 12 fps. So for the first motion, there are 3s x 12fps = 36 frames total including the keyframes. If we have a second motion with, say, 72 frames for the same movement from point A to point B, then the motion lasts for 72f/12fps = 6~~

16 Not just smoothness. A high-speed video camera recorder used by professionals to capture action scenes in sport or nature can have a resolution up to 10,000 frames per second! That way, they can analyze every bit of the actions they capture.

seconds – twice as long as the first motion for covering the same distance. Both motions have the exact same keypoints, but different number of frames between their keypoints, and thus give the effect that the second motion is (twice) slower than the first motion.

Figure 9.8. Various number of frames of the same two keyframes creates two different motions

So by only allowing users to apply the Effort parameters to the keyframes (or keypoints), it saves the users from being overwhelmed from modifying each frame. Also, this helps keep the integrity of the motion by keeping the modifications on the higher level.

Multiresolution Filtering

Bruderlin and Williams started by applying multiresolution filtering to the motion signals using bandpass filters. The number of frequency bands depends on the number of frames of the motion, where each in sequence the signal is scaled down by a factor of 2. If m is the number of frames, and fb is the number of frequency bands, let:

 (9.7)

Then: $fb = n$

The signal is filtered using a kernel w of size 5. Instead of reducing the signal by a factor of 2 for each sequence, the kernel is expanded by a factor of 2 by inserting zeroes between the kernel values.

The kernel values are: $a = 3/8$, $b = 1/4$, and $c = 1/16$. So in w_1 the kernel values have $2^0 = 1$ distance (distance 0 is impossible as it means they are in the same spot), in w_2 the distance is $2^1 = 2$, in w_3 the distance is $2^2 = 4$, and so on. Notice that in the kernel, the values add up to 1.

The filtering algorithm is as follows (the following explanation is taken directly from (Bruderlin, Williams 1995):

1. Calculate lowpass sequence of all fb signals ($0 \leq k < fb$) by successively convolving the signal with the expanded kernels, where G_0 is the original motion signal and G_{fb} is the overal l average intensity of the signal.

    ( : convolution sign) (9.8)

   This can be calculated efficiently by keeping the kernel constant and skipping signal data points (i ranges over all data points of a signal):

 (9.9)

2. Obtain the bandpass filter bands ($0 \leq k < fb$):

 (9.10)

3. Adjust gains for each band and multiply Lk by their current gain values

4. Blend bands of different motions (optional)

5. Reconstruct motion signal:

 (9.11)

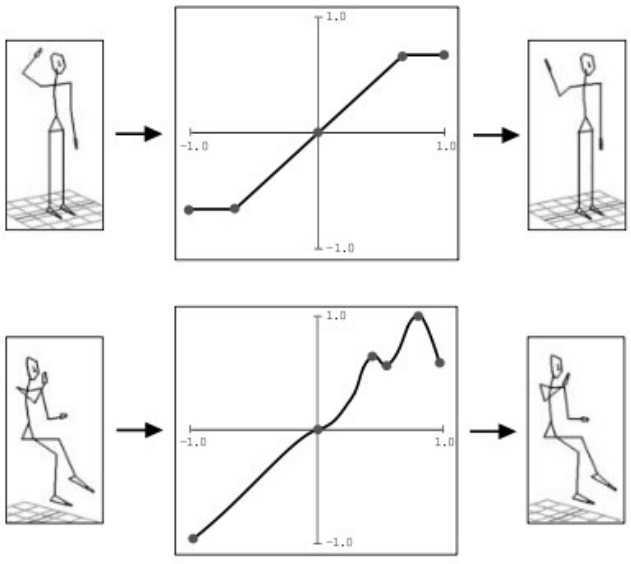
Figure 9.5 illustrates how adjusting the gains for each band affects motions. The graph in the middle is just showing the effect of the gain adjustments to a step function. The top image shows the effect of increasing middle frequencies, result in smoothed but exaggerated walk [1]. The middle image shows the effect of increasing low frequencies of band 5 and 6, which creates an attenuated walk with reduced joint movements. The bottom image uses negative gain value on the lower frequency (band 5) while increasing the gain on high frequencies (bands 0, 1), and the knocking motion has an exaggerated anticipation and follow-throughs. The same gains are applied to all degrees of freedom.

2.2.2 Motion Waveshaping

Motion waveshaping refers to the approach to superpose a motion signal input with a *shaping function*. The motion signal is normalized to the range of [-1, +1]. If the shaping function is an identity function *f(x) = x*, then there are no changes to the input motion signal.

In the Figure 9.11 top, the shaping function is used to cap/limit the joint angles of the motion. If a joint angle exceeds the limits (the horizontal part of the function), then the angle will remain at that value. The shaping function can be applied to the whole duration of the motion, and thus provide a 'global' view and changes to the motion. For example, if the shaping function is something like a gaussian/bell curve, where the center is has positive (> 0) value, and the extremes are low, then the middle part of the motion will be exaggerated, while the start and end motions will be reduced. In Figure 9.11 bottom, undulations (wavy effect) are applied towards the end of a motion.

Figure 9.118. Motion waveshaping (image taken from [1])



(Top: hard-capping (limiting) joint angle; Bottom: adding undulations to a motion)

Motion waveshaping is useful to quickly add characteristics to a motion, and thus enable animators to quickly create a library of different motion characters.

~~*2.2.3 Motion Displacement Mapping*~~

~~Motion displacement mapping allows users to modify a motion by specifying the amount of displacements at any points of the motion. The resulting motion will have the displacements at the points specified by the user while preserving the continuity of the motion. So, if a user specifies that point X of a motion is to be increased by some value (For example: the joint angle at point X is 100, and the user wants to add 20 to it = 120), instead of just point X being changed which may cause an abrupt motion, the points around X are also modified, creating a smoothly continuous motion.~~

~~This is done by creating a spline fitted through the displacement points , where the start and end points of the spline always start at zero (0). The motion signal is then superposed (added) with this spline to create the final modified motion. The motion displacement mapping is only done per joint/DOF.~~

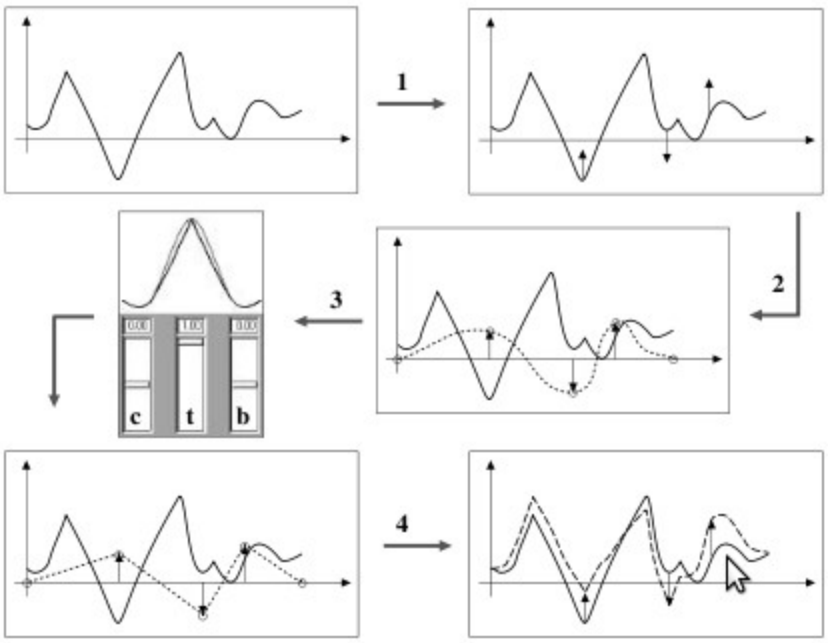


Figure 9.12. Motion displacement mapping process (image taken from [1])
(1: User specifies displacement points and amount of displacements, 2: a spline is fitted through the displacement points, 3: user can modify the spline parameters, 4: the spline is added with the original motion (solid line) creating a modified motion (dashed line))

There are several other methods explained by Bruderlin and Williams such as multitarget interpolation and timewarp/dynamic timewarping. Those methods were shown to be useful to combine two (possibly more) motions into one. Currently, they are not of interest in this project, and is not explained in this report. Please refer to [1] on more information on those methods.

Hermite Interpolation

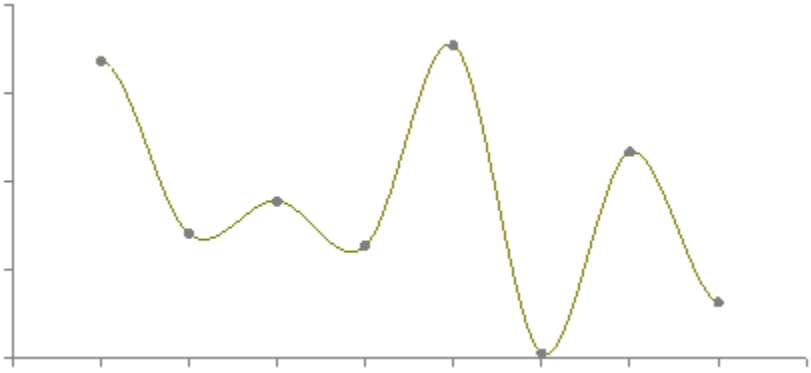
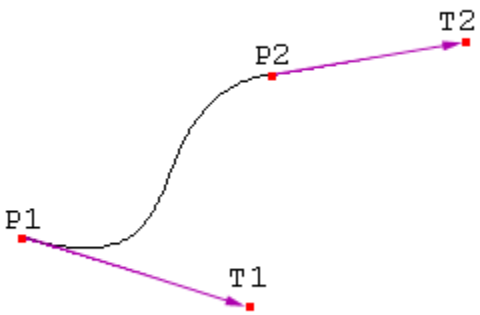


Figure 9.13. Hermite interpolation (image taken from [4])

Interpolation is essentially the problem of finding a point between two points. The problem arise when we do not know *how* the two points are connected. That is, the two points (let's say point A, and point B) may be connected by a straight line (linear), or by some curve (splines). The linear or spline connections between the two point may be described by a function *f(x)*. However, in discrete data points, such as animation frames, the function is generally unknown. If A and B are discrete data points, and we want to find a point in between, then we first must *estimate* the function which is fitted (passes) all the known data points including A and B.

In general, the minimum input to an interpolation method is a line segment described by two points like A and B (start and end point), and a value in [0,1] that specifies the interpolation point. A value of 0 equals the start point, 1 equals the end point of the segement, and anything in between is a point between the start and end points. Interpolation using Hermite splines takes 2 additional inputs: *tension*, and *bias* (although in [3] there is a third input: *continutity*). As implemented by [4], *tension* has a value of [-1, +1] where -1 = low tension, 0 = no tension, +1 = high tension. *Bias* may have negative or positive values (not limited to some value), where negative value will make the interpolated point biased (closer) towards the endpoint, positive value biased towards the start point, and 0 is even (no bias).

Figure 9.14. Tangents of Hermite spline



(P1: start point, P2: end point, T1: tangent of P1, T2: tangent of P2)

The *tension* and *bias* parameters are affecting the *tangent* of the start and endpoints. The tangents are essentially derivatives of the spline function at those points. This means that they specify the rate of change of the bends of the curve (tension), and the direction of the curve as it passes the start/endpoints (bias). They can also specify the rate of change for the rate the bends and directions change (the second derivative – referred to as continuity). These parameters are introduced by D. Kochanek and R. Bartels to give more control.

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

 (9.12)

 (9.13)

Where:

- t = tension
- c = continuity
- b = bias
- P_{start} = start point
- $P_{start-1}$ = a point before the start point
- P_{end} = end point
- P_{end+1} = a point after the end point

I used Bourke's implementation which set normal continuity preference (c = 0).

As such, to calculate the tangents, additional points are required to do Hermite interpolation. Namely, a point *before* the start point ($P_{start-1}$), and a point *after* the end point (P_{end-1}). This causes a bit of a problem when the P_{start} is at time = 0 and P_{end} at time = end where there are no points before the start point and after the end point. To overcome this, in my implementation I assumed that $P_{start-1}$ has the same value as P_{start}, and P_{end+1} has the same value as 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:

 (9.14)

Where *s* is the value [0,1] specifying the interpolation point between P_{start} and P_{end}. Then, the interpolated point is calculated as follows:

 (9.15)

CHAPTER 10 – EXPERIMENT

Our hypotheses:

1. *Synthesizing humanoid robot motions is more effective and intuitive when the motions are represented and treated as signals. Thus, …*
2. *By treating motions as signals, emotional and physical expressions can be easily embedded in the motions of humanoid robots. Because, …*
3. *Emotional and physical expressions can be represented as a set of signal transformation functions. However, …*
4. *Establishing context, preceeded by interaction, is crucial to understand the expressions of the emotional and physical states present in a motion, especially in the absence of facial expressions.*

10.1 KHR-1 Humanoid Robot

The KHR-1 is a humanoid robot, meaning it has the appearance in terms of body proportions and limbs that are similar to a human. The KHR-1 consists of a head, a torso, a pair of arms, and a pair of legs. In total, the KHR-1 has seventeen degrees of freedom (DOF): one on the neck, three on each arm, and six on each leg. There is no DOF on its torso. Each DOF corresponds to one servo.

The seventeen servos are controlled by two RCB-1 servo controller boards which are connected in tandem. Each RCB-1 is able to control up to twelve servos. The first controller board controls the seven servos controlling the head and arms (e.g. upper body), and the second board controls the remaining twelve servos of the legs (e.g. lower body).

~~To control the servos, commands in the form of strings of bytes must be sent to the controller boards with certain format. A full description of the commands and their formats is presented in Appendix X.~~

~~Figure 10.1 KHR-1 Robot~~

10.2 Input-Output

Because part of the goals of this research is to have socially-interactive robots, the user must be able to interact with the robot through human social interaction methods. The first interaction method is a face detection-recognition vision system. This enables the system to recognize a person's face and retrieve his or her name from the database. The second interaction method is the chatbot program based on the ALICE system, that is able to carry out *seemingly intelligent* conversations. The conversation is based on pattern matching on the sentences given by the user. Depending on how well its conversational database is constructed, the program can exhibit context-specific conversations. The third interaction method is a speech synthesis system that will convert the conversation text from the chatbot program into speech. This system is based on an implementation of the Mbrola speech synthesis engine. These three systems provide then one 'closed-loop' conversational interaction system with the user (i.e. 'closed loop' because it creates a complete feedback loop from user's inputs to a response from the robot, which then initiate another user input).

But that is not all. These inputs, from vision and conversation, are then being used by the system to determine the response of the robot through motion, and to modify the motion (e.g. emotional expressions in the motion). Specifically, the combination of information received from vision (such as: presence of a person/face, number of people) and conversation (through certain keywords extracted from the dialogue) are used to select and determine the motion editing function and its parameters. For

example: through vision, the system detected the presence of a person. Through the conversation with the person, the system determined that the person is in a 'happy mood', and the system is also in a 'happy mood'. When there is a keyword in one of the sentences that would invoke an action, let's say, 'wave right hand', the system would then retrieve the corresponding motion file from its database. The motion list for motion 'wave right hand' is then internally represented as a signal, and the system would apply a signal processing function that modifies the motion to represent the mood 'happy', such that the person observing the motion could understand that the mood of the robot is 'happy'. If this is thought as a game, this would be a guessing game such as 'charade', where a person can only gesture to the other players to describe a certain topic/idea/keyword and those other players have to guess what topic/ idea/keyword that person is conveying. In this case, that gesturing person is the robot, and the idea that it is trying to convey is the internal state (i.e. mood, personality, and so on) 'embedded' into its motions.

Our task is then to provide the robot (system) with the set of signal processing functions that appropriately express the internal states of the robot through its motion, based on the ideas on how those expressions are presented by humans in normal, daily interactions.

10.2.1 The Vision System

The vision system serves manly as input to the interaction system. The system also provides feedback to the user on what is being seen and done by the vision system. To obtain vision input, a standard USB web camera is used. The stream of images from the camera are queried and processed using the OpenCV image processing library. Because the core of the system was written in the Python programming language, and the OpenCV library was written in C/C++, a Python – C/C++ wrapper/binder was used to enable access to the OpenCV library through Python.

Currently, the vision system consists of a face detection system. The data that can be obtained from the face detection system are: the presence of one or more person, and the number of faces detected.

10.2.2 The Dialogue System

The dialogue system serves as both input and output for the whole interaction system. The dialogue system can be used to initiate an interaction, by either the person/user or the robot/system itself. At its current state, the dialogue system takes only text inputs, which are then processed using the ALICE conversational agent program (popularly called as 'chatbot' program). The agent then gives a response also in text, based on its knowledge of input-output sentence patterns. The conversation patterns in the database are represented using Artificial Intelligence Markup Language (AIML).

The other part of the dialogue system is the speech synthesizer, which converts text from character strings into speech/spoken language.

10.2.3 The Motion System

The motion system controls the execution of the motions for the robot. The motion system has four main tasks: to load a specified motion, select a transform function for the motion from a set of transforms, apply the transformation to the loaded motion, and execute the motion on the robot. The first three tasks will depend on the processing of the inputs from the vision and dialogue systems.

For the first task, as illustrated above, a motion is loaded depending on the keyword(s) recognized in the dialogue, or a perceived information from vision (e.g. seeing a person's face after a period of no presence will prompt a greeting motion). As for the second task, the selection of the transform function for the motion also depends on certain context from the conversation. The dialogue system is set up so

it could pick up information such as user's mood, user's personality, and certain user-input keywords that would affect the robot's mood. This information is used to select the transform functions. Each of these functions has also different adjustable parameters that would affect the motion extremely or subtly; this is the third task. Finally, the execution of the motion just consists of either: feeding a stream of servo positions serially to the RCB-1, or invoking a built-in RCB-1-specific commands (such as calling a 'scenario' from the memory of RCB-1, in which case no transformation can be applied because the 'scenario' is loaded directly from the on-board memory). In both cases, the motion execution only concerns with the communication using the serial port, formatting the output byte string, and is not affected by the perceived inputs (from vision and dialogue).

Currently, the system is unable to generate its own motion from scratch. The system relies on pre-programmed motions which are stored as motion list files in the comma-separated values (.csv) text file format. The pre-programmed motions were created using the Heart-to-Heart software that was provided with the KHR-1 kit. The Heart-to-Heart software allows the users to create some basic movements, but it is difficult to use the software create subtle effects or expressions in the movements. Therefore, the Heart-to-Heart software was used only to create these basic motions.

10.2.4 The Interaction Rule System

The Interaction Rule System (IRS) is the system that 'glue' the vision, dialogue, and motion systems together. Essentially, this system is the cognition system consisting of a set of rules which is responsible of figuring out the context of the interaction that is happening, determine the appropriate motion responses, and the emotional and physical expressions that should be present in the motion response. This may be considered the true 'brain' of the whole system, while the other systems are sensory, and motoric systems.

IRS takes all the information extracted from the inputs (e.g. Number of people present, dialog topic, user's mood, and other keywords), and based on the interaction model (i.e. *rule*) we implement, it determines which motion to execute as a response, and the parameter values of the motion transformations.

In the IRS, we can implement different interaction models found in literatures, such as “personal space” model (physical-distance-based), “teacher-student” model (turn-based), “teamwork” model (cooperation-based), and so on (Breazeal 2002). For our system, we implement a simple, direct conversational model. That is, because we are interested in seeing different emotional and physical expressions in the motion as a response of the interaction between the user and the robot, our interaction model will only focus on *capturing the contexts of the conversations* and translate it directly into motion.

We capture contexts by recognizing certain keywords and saving them in some variables (in parantheses), such as: the current topic of discussion (Topic), mood/emotion/attitude of the robot (Robot's Mood), the user's attitude towards the robot (Personality), the user's name (Person Name), action/user commands (Action), and the context/object of the previous sentence (It). We can add these variables in the ALICE engine using AIML. The Action variable contains the keywords that are directly mapped to an action. For example, if the Action is 'Dance', then it is the command to the robot to execute its dance motion. If Action is 'Greet', then the robot will raise its left arm (greeting motion).

a

To set the values of these variables, we rely on the Dialogue module which uses the ALICE engine to retrieve the keywords from the conversations. In addition, we also try to capture the *trends* or the

progress over time of these contexts. For example, the user may say an insult, that would set the behavior of the robot (Robot's Mood variable) to be 'unhappy.' If for the next several conversations (i.e. Exchange of words) the user says another insult that also set the robot behavior to 'unhappy' and the behavior of the robot is still 'unhappy' (no change), this is recorded as a second offense, and elevates the intensity of the behavior of the robot from 'unhappy' to 'angry.' If there was a change in the behavior of the robot, the intensity level is reset. Similarly to other emotions and also physical states.

10.3 Emotional States, Physical States, and Their Expressions in a Motion

According to Amaya et. al., two motion parameters play the biggest role of creating emotional motions: *speed* (how fast the motion execution) and *spatial range* (or range of motion). In terms of signals, speed refers to the wavelength, or width of the motion signal. Range of motion refers to the amplitude, or height of the motion signal. Sad motions tend to have a wider wavelength/width (slower), and lower amplitudes (reduced motion range). Angry motions are the opposite of sad motions (faster and bigger motion range).

Even in an anger, when the intensity is low ('Annoyance'), we do not always see a significant difference in speed and range of motion to, say, low-intensity terror ('Apprehension'). For example, the movement speed and range of motion of a person being annoyed by being stuck in traffic (e.g. slapping the steering wheel in disbelief), may not be any different from a person realizing he just lost his wallet (e.g. slapping his forehead in disbelief). The reactions may be different. Therefore, we adapt the idea from Plutchik where each emotion type has a common measure of intensity, which in turn determines the speed and range of motion in the expression of the emotion. For this, we introduce the concept of *heartbeat*.

In (*Lang94, Wiens00*), there are some indications that as a person is experiencing intense emotions, it drives the person's heartbeat to go faster, whether it is of sorrow, anger, or happiness. For example, for the emotion happy: when a person is only slightly amused, his/her heartbeat remains normal, or at most, becomes slightly faster. When a person is ecstatic (intensely happy), his/her heartbeat could go extremely fast. It even applies to sadness (the more intense is the sadness, the faster the heart beats). Conversely, as the intensity of the emotions goes down, the heartbeat becomes slower – we often refer this as being *calm* or neutral, while the intense emotion is often referred as being *hysterical*

Therefore, we apply a 'heartbeat' function as an intensity indicator. The 'heartbeat' function is a pulsing signal, with certain interval called *heart rate*.

In terms on how this heartbeat concept affect motion, we can think of it as the main indicator for the speed and motion range mentioned above. As the heartbeat goes faster, the faster and bigger the motion becomes, and vice versa. Next, we have to figure out how or what transformation to use to change the speed and spatial range of the motion signal. This is discussed in the next section.

The question is then: *if only intensity of the emotion – regardless of the type of emotion – that drives the speed of the heartbeat, thus, determines the speed and range of the motion, how would we be able to distinguish which emotion is being expressed?* Through *context;* that is, what is the state of the interaction, and what motion is being given as the response. For example: when a child is extremely unhappy (maybe her parents did not get her the toy she wants), she will throw a tantrum, running and jumping all over the place (and maybe crying as well). Similarly, when the child is excited and happy

(maybe she got the toy she really wants), she might also show her happiness by running around the room or jumping around. This is a slight shift in the generic paradigm on creating expressions on a humanoid robot. The generic paradigm is that happy emotions are expressed in bigger and faster movements, while sad emotions are expressed in smaller and slower movements. Now, we have to ask: *how intense is the emotion?* Making the type of emotion itself irrelevant in determining the 'size' and speed of the motion.

Thus, we emphasize the importance of interaction to establish the context. In the Disney Animation Principles, this is called *Staging*. For example, when the interaction between the user and the robot goes bad (e.g. the user keep disagreeing with the robot and say to the robot that it is stupid), as it gets worse, it may drive the 'heartbeat' up. When the movement of the robot becomes erratic because of the high heartbeat (causing the movement to be fast and big), the observer understands that movement as the robot being angry, uneasy, or frustrated. However, when the interaction goes well and gets better, it too drives the 'heartbeat' up, but as a different emotion. The faster and bigger movement of the robot is now understood as being excited, ecstatic, or very happy. As a counter-example: when the context of our interaction with someone is about a sad topic (for example: a relative just passed away). If the other person suddenly making waving arm gestures that are fast and big, we would assume the person is so sad that he becomes hysterical, but instead he declared that he is very happy. This disrupts our interaction context; now we are focused (i.e. the context becomes) on how this person is not behaving within context and not on the sad news, and we might label this person as a mad man. This example gives us another hint that we must establish and understand the context of the interaction to understand what emotion is being expressed. This phenomenon may be from a human social interaction norm or rule. When it is violated – where the expression is not relevant to the context – our interaction becomes awkward.

Note that we have not even mentioned about the facial expression. Facial expressions would explicitly indicate what kind of emotion is being expressed. However, since our robot does not have a 'face' per se, we have to attempt to show these emotions solely through the actions of the robot. Our analogy is on the puppets in Sesame Street or the Muppet Show. Those puppets cannot show different facial expressions as their emotional expressions (with some exceptions on certain puppets), yet in the show we were able to clearly identify what emotion they are expressing through their actions, which are controlled by human hands. The Oscar character in figure 10.3a, for example. Oscar is a grouchy character which is always angry at everything, sarcastic, and see things negatively. Yet, as we see in the picture that Oscar's facial expression shows him being happy, and perhaps friendly character. We can only see Oscar's true angry character during his interactions in the show (his facial expression does not change throughout the show). Another good example is the R2-D2 robot from the movie Star Wars. The physical form of the robot is essentially a trash can on wheels with no humanoid 'face', yet we can see its emotions (so to speak) through its movements. The next example is the image of Bugs Bunny holding what appears to be a dynamite stick (figure 10.3.b, 10.3c). In the figure 10.3b, Bugs Bunny has an angry facial expression. We can assume that what he is doing is in an angry gesture. In the figure 10.3c, – edited using an image editor (GIMP) – by having the same body pose and action, with a different facial expression, the body gesture cannot be said to be an angry gesture anymore. We might think he is happy (despite being at an awkward pose). These examples, especially the Bugs Bunny example, show that in some cases, the same body gestures/pose/action can mean different things depending on the context. And the context, if it is on emotion, can quickly be established from facial expressions.

Figure 10.3 a) Oscar character from Sesame Street, b) Bugs Bunny character with angry face, c) Bugs Bunny character with happy face.

Along with emotion, a person's physical state can also be expressed in the motion. The common understanding is that the physical state 'tired' is a state where the person has very little energy, thus his movements will tend to be slow and small (have reduced motion range). Conversely, when the person is energetic (have high energy), his movements will be faster and bigger (have increased motion range). However, since expressing the physical states solely through speed and 'size' of the motion would make it indistinguishable from the expressions of the emotional states, we added *posture* as an additional expression of the physical state. A 'tired' physical state will have the robot at a slouching/slumping posture (knees slightly bent, torso arching forward, arms hanging down). An 'energetic' physical state will have the normal straight posture. These are very broad generalizations of course, and may not be true all the time in the real human-human interaction. But we will use this generalization as a basis to express the physical states, at least for our experiment.

When the robot is in one of these physical states, the executed motion will be superposed onto the posture signal, so that the posture is apparent even in the motion, thus expressing the physical state

along with the emotional state. Since posture is static (i.e. The joint angles remain the same all the time), it can be represented as a list of constants of the servo positions. Therefore to superpose posture with a motion signal, the constant (posture value) just only needs to be added to all the data points of the motion signal. This is the same principle as applying a waveshaping function.

So, our 'rule' is as follows:

1. Initally, the robot (system) starts with a 'neutral' physical state, and low intensity 'happy' emotional state.
2. When a user initiate interaction by typing in a sentence, or when the vision system detects a face where before it did not detect any, the intensity of the 'happy' emotional state is increased. This gives the robot a behavior that it is happy to see a person, and assumes the person has a 'nice' or 'polite' personality.
3. An action (motion) may be triggered by certain keywords. For example:
    1. The input keyword “Dance” will command the robot to load the dance motion and play a music file.
    2. The input keywords “Hello”, “Hi”, “Bye”, or “Goodbye” are associated with the action 'greet' and triggers the motion of raising the left arm.
    3. For testing and demonstration, direct commands are available to immediately set the values of the context variables. For example: the user can enter “SET MOOD TIRED,” which will set the value of the Robot's Mood variable to “Tired,” and makes the robot load the “Tired” posture.
4. Regular conversations (sentence patterns that do not change or set the Robot's Mood, Personality, and Action variables) may initiate some arm gestures. The gesture will be selected from a limited set of pre-programmed arm gestures, which will be modified according to the

context, emotional and physical state of the robot.

5. Interactions that repeatedly set the Robot's Mood to the same value increase the intensity of the emotion of the robot.
6. Conversely, repeated interactions that do not trigger the Robot's Mood (i.e. the interaction do not stimulate the emotion of the robot) will reduce the intensity of the current emotional state until it reaches 0 or the lowest intensity. This means the interaction gets boring.
7. If the interaction triggers an emotion that is the direct opposite of the current emotion (according to Plutchik's model), then the current emotion remains, but the intensity is reduced twice as much as in rule #6. If this repeats, the intensity level of the current emotion may drop below 0, which then switches the emotion to the opposite emotion. Thus, the intensity of the new emotion equals the negative of the value of the intensity of the previous emotion. Or:

$$emointensity_{\bar{A}} = 1 - emointensity_{A}$$

where: $emointensity_{A}$ is the intensity of emotion A, and $emointensity_{\bar{A}}$ is the intensity of the opposite of emotion A. For example: the intensity of emotion 'Joy' is $emointensity_{Joy}$, thus, $emointensity_{J\bar{o}y} = emointensity_{Sadness} = 1 - emointensity_{Joy}$. The value of the emotion intensity is [0,1]. For our experiment, we will use discrete increments/decrements of 0.25 each step (i.e. when the emotion is triggered), according to the four regions of emotional intensity in each spoke in Figure ... (Plutchik's model).
8. The intensity of the emotion is represented to the user in the form of virtual heartbeats. The frequency of the heartbeat (in beats per minute, or *bpm*) is proportional to the intensity of the emotion; the higher the intensity value, the higher the frequency of the heartbeat. As the

baseline, we use 70 bpm in a calm state or at the lowest emotional intensity level, which is the average heart rate of a healthy adult human male (*Reference*).

9. The intensity of the emotion or the heart rate of the robot determines the parameters of the signal transformation functions. The parameters will determine the speed, and range of motion of the selected motion. This means, when the emotional intensity level is high, we must adjust the parameters of the signal transformation functions, such that they will increase the amplitudes of the motion signal (making the motion 'bigger'), and shortening the motion signal (making the motion faster). And vice versa when the emotional intensity level is low.

10.4 Set of Motion Transformations

Since we took inspirations from the work of Bruderlin and Williams (Bruderlin, Williams 1995), and Unuma (Unuma, Anjyo & Takeuchi 1995), by treating the motions as signals, what we have at our disposal for our motion transformation is virtually every kind of signal processing techniques ever conceived! From Fourier transforms, multiresolution filtering, low/high frequency filtering, matched filtering, waveshaping, interpolation techniques, through wavelets techniques can be used to modify the motions. The question we are trying to answer is then, *from all those available techniques, which technique can be used to express what emotion, and if it can, how to achieve it?* Once we answer this question, then we have our set of motion transformations that can be invoked to modify our 'basic motions' into 'expressive motions'.

(Bruderlin, Williams 1995) showed that signal processing techniques can be applied as motion transformations. (Unuma, Anjyo & Takeuchi 1995)used Fourier analysis to blend two periodic behaviors (e.g. normal walk + tired walk). Unuma et. al. showed that by representing both behaviors as Fourier series, superposing them, and adding a parameter s = [0,1] the authors were able to create

smooth transitions between the two behaviors. Setting $0 \leq s \leq 1$ gives an interpolated or 'hybrid' behavior while still retaining/exhibiting the characteristics of the original behaviors. In short, Fourier analysis/synthesis effectively interpolates (merges, morphs) the two behaviors.

However, Unuma's approach is still a bit restricted to *periodic behaviors*, and provides little insights to other kinds of behaviors which are not periodic. A walking motion is periodic, while jumping back from a surprise is not. Using other approaches may be easier to achieve some of the 'expressive' effects in addition to using Fourier- based transformations. Moreover, not all of the motions will always be periodic, and neither will be the expressive behaviors (e.g. emotion). Thus, using some localized transformations in time (such as wavelets), and other kinds of transformations will allow to create a richer 'expressive vocabulary' for the system.

While our approach is still in the same spirit as the approach of Unuma et. al., that is, it is mostly based on superpositioning a behavior with another behavior, our objective is slightly different. Our goal is to find a set of general behavioral patterns that could be applied to any kind of motion such that the desired expressive behaviors would be apparent in the motion. So, instead of creating a 'slightly tired walk' by combining 'normal walk' with 'tired walk', we are combining 'walk' with 'tired' to get 'tired walk'. Thus, we might be able to combine 'wave hand' with the same 'tired' to get 'tired wave hand', instead of having all different variations of each motion – we want to encode the 'adverbs' (e.g. 'tired') so we can add them to the 'verbs' (e.g. walk), so to speak. The variations, or impacts of these 'adverbs' on the basis motions – that is, the parameters of the transformations – and also the 'adverb' itself, will be determined from the information received from the inputs mentioned above.

10.5 Methodology of Finding the Set of Motion Transformations

Therefore, to look for our set of motion transformations, several transformation techniques were explored. Throughout our investigations of these techniques, we follow a few steps to develop our set. First, we select one kind of transformation. Second, we observe what is the behavior or characteristics of the transformations. For example, what kind of transformations are happening to the motion signal when the parameters of the transformation are adjusted and re-adjusted? Third, we try to adjust the values of one or a combination of the parameters of the transformation to try to get the effects or expressions that we want. The set of 'target' effects or expressions is based on the literature research on motion expressions. We took hints and cues from the performing arts and animation theories such as the Laban Movement Analysis (LMA) and Disney Animation Principles (DAP). We iterate this process of adjusting and re-adjusting the transformation parameters until we have exhausted our options to create those 'target' expressions. If we are not able to synthesize all those 'target' expressions using that particular transformation, we try another transformation, and again iterate the three steps. If we are able to synthesize all the 'target' expressions using one kind of transformation, we could still try different transformations to investigate if other transformations can also achieve the same expressions/effects. Then, if other transformations can also achieve the same expressions/effects, we can compare their performances (e.g. in terms of speed of computation/synthesis, resource usage, and so on). In this heuristic, we evaluate the 'expressiveness' of the transformation using .... (*see the metrics paper*)?

Looking at this heuristic, it would be possible in the future to develop some evolutionary computation algorithm to look for this set, given that the objective or cost function can be defined quantitatively.

10.6 Set of Emotional and Physical States

We define a set of emotional and physical states for the robot. Our set of emotional states is based on

the model by Plutchik, which consists of eight basic states: anger, fear, sadness, disgust, surprise, curiosity, acceptance, and joy (*.Reference to Plutchik's model*). Each of the basic emotional states has a degree of intensity. For example, the intensity of the emotion anger ranges from the least intense, annoyance, to the most intense, rage. Our set of physical states consists of: tired, energetic, and neutral (or normal).

We believe that these emotional and physical states can be expressed in (or embedded into) any kind of motion using signal transformation functions, such that the observer can identify the emotional and physical states in the motion. Amaya et.al. showed that by comparing motion capture data of a 'neutral' motion and an 'emotional' version of the same motion, one can extract the 'emotional transformation' function from the difference between the motion signals. The authors tried to verify their findings by appliyng the acquired emotional transformation function to a neutral motion of a different kind. Their results showed that the synthesized emotional motion was close enough to the emotional version of the motion-capture data of this other motion.

The emotional states will mostly affect the motion parameters (such as speed, motion range, acceleration/deceleration), while the physical states will affect the posture of the robot, and some of the motion parameters. In addition, we are looking for 'naturalness' in the motion, as explained in the DAP. From the eleven principles in DAP, we pick three (or is it four?) principles that we belive have the most contribution to 'naturalness': anticipation, follow-through, and stretch and squash. A quick review; anticipation is the brief starting motion which is in the opposite direction of the main action (e.g. The anticipation of jumping up is crouching down). Follow-through is the continuous recovery motion from the main action, back to a resting position (as opposed to an immediate/abrupt stop). Stretch and squash relate to both anticipation and follow-through, and it is the rubber band-like quality in the

motion of an organic matter (e.g. Crouching before jumping can be considered squashing, extending the body when jumping can be considered as stretching, and landing from the jump by bending the legs to absorb the impact, can be considered another squashing).

The set of emotional and physical states, and the 'naturalness' quality of the motion are our objective functions, so to speak. These are the things we are trying to achieve by only treating the motion as a set of signals.

10.7 First Step: Selecting a Transformation

We selected a set of well-known signal processing techniques that we use to try to 'emulate' the expressive behaviors. In our experiments, we selected the following waveform-transforming techniques: Interpolation, Multiresolution Filtering, and Waveshaping.

10.7.1 First Technique: Interpolation

Basically, interpolation is the method of adding more data points between two known data points. There are different kinds of interpolation: linear, b-spline, cubic spline, Hermite spline are among the most commonly used ones in animation.

We are using 'basic motions' or simple behaviors, which are created using Heart-to-Heart. The motion signal of these behaviors when plotted as curves, looks very linear. What this means is that the motions tend to have a constant acceleration every time, which is the main reason why the movement looks 'robotic'. It is not impossible to create motions with good acceleration/deceleration effects, but it is difficult to achieve using Heart-to-Heart and it requires a good sense and knowledge of animation. So, our intention of applying interpolation is to create this accelerating/decelerating effects when the

motion changes its direction. Therefore, we looked into spline interpolations.

We found no problems using b-spline, cubic spline, or Hermite spline interpolation techniques. The motion signals became smoother sinusoidal signals when using any of these methods. Interestingly, in addition to having the acceleration and deceleration effects on the motion signal, the resulting interpolated motion signal also exhibits some overshoots. In animation, this would be equivalent to the 'follow-through' effect, which says that a natural-looking motion does not stop abruptly. Instead, the motion continues and finishes after absorbing completely the momentum of the current motion before the motion changes direction or to a different motion. This effect is also related to the 'anticipation' effect, or a 'preparation' of building the momentum before a motion.

However, we decided to choose the Hermite spline interpolation because of the additional degrees of freedom it provides for the interpolation. The Hermite spline offers two additional interpolation parameters: *Tension* and *Bias* (b-spline and cubic spline interpolations do not have these additional parameters). The Tension parameter allows the adjustment of the parabola of the curve. The Tension value ranges from less than 0 (< 0) or bigger than 0 (> 0), with 0 being neutral. The higher the tension (> 0), the curve becomes sharper on the inflection points (points where the curvature changes direction). Conversely, the less the Tension (< 0), the curvature at its inflection points becomes less sharp.

The bias parameter allows adjusting the curve at the inflection points to 'bias' towards the first point or the second point. At 0, the highest (or lowest) inflection points of the curve are at these points themselves. If the bias is less than 0 (< 0), then the inflection points are *leading*, meaning they happened earlier (assuming the x-axis is time). And vice versa, when the bias is greater than 0 (> 0),

the inflection points are *lagging*, i.e. they happened later. In other words, bias seems to affect the *phase* of the signal. In reality, it is not a true phase-shifting behavior. This is because the original data points remain at their original positions. In Table ... if we compare the signal where tension = 0, bias = -5, with the signal where tension = 0, bias = 5, we see that at the positions of the original data points y = [10, 20, 15, 25] remains at x = [1, 5, 9, 13]. In phase shift, this should change. Instead, this appears more like rotating the signal 180 degrees, or mirroring it twice: once vertically, and once horizontally. In addition, the amplitude of the signal is alo amplified.

Hermite interpolation requires four points to do the interpolation. Let's say the four points are y0, y1, y2, and y3. Points y1 and y2 are the two points we want to interpolate; we want to add additional data points between these two points. Points y0 and y1 are the points before y1, and after y2, respectively. Hermite interpolation requires these two additional data points (y0, and y3) to ensure the continuity of the curve. This is done by calculating the incoming tangent to point y1, and outgoing tangent from point y2. To calculate the incoming tangent at y1, then we need to know where the previous point (y0) is relative to y1. And to calculate the outgoing tangent at y2, we need to know where the next point is (y3).

The Hermite interpolation uses four basis functions:

$$h_1(s) = 2s^3 - 3s^2 + 1$$
$$h_2(s) = -2s^3 + 3s^2$$
$$h_3(s) = s^3 - 2s^2 + s$$
$$h_4(s) = s^3 - s^2$$

where *s* is the interpolation point. For example: s = 0.5 means the point exactly in the middle between

y1 and y2, s = 0.25 means the point a quarter of the way between y1 and y2, and so on.

These basis functions are multiplied with the four values: the points to interpolate (in this case, y1, and y2), and the incoming tangent at y1 (T1) and the outgouing tangent from y2 (T2). The tangents are obtained from:

$$T1=\frac{(1-t)(1-c)(1+b)}{2}\left(P_i-P_{i-1}\right)+\frac{(1-t)(1+c)(1-b)}{2}\left(P_{i+1}-P_i\right)$$
$$T2=\frac{(1-t)(1+c)(1+b)}{2}\left(P_i-P_{i-1}\right)+\frac{(1-t)(1-c)(1-b)}{2}\left(P_{i+1}-P_i\right)$$

where *t, c,* and *b* are Tension, Continuity, and Bias parameters, respectively. *i* is the point (i.e. Frame) number.

Thus, the interoplated point P can be obtained by multiplying the Hermite basis functions with these four values, and adding them together. Or in matrix form:

$$P=s*c*h$$

where:

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

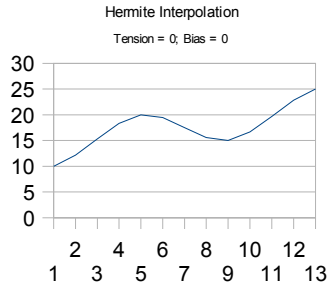
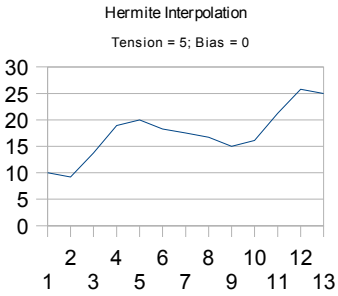
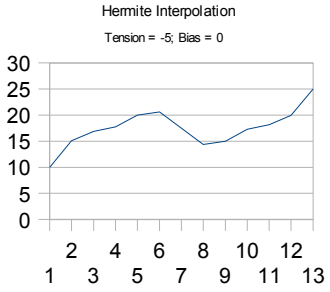
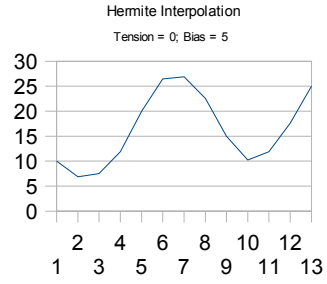
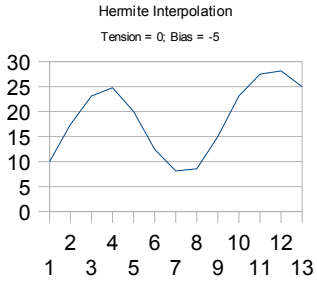
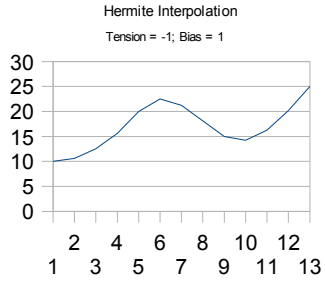
In terms of how these parameters affect the motion, the tension can be thought of affecting the *stiffness* of the motion, while the bias affects the *leading or lagging* effects of the motion. In LMA terms,

tension seems to partially influence the *Flow* Effort parameter. There does not seem to be any single equivalent term for the bias parameter, unfortunately. This is because the bias parameter causes multiple effects: the inflection points to lead or lag, and it also increases the amplitude of the motion signal (for both bias > 0, and bias < 0). However, the phase-shifting behavior of the bias parameter makes it look as if it would create the *anticipation* at the beginning of the motion and the *follow-through* at the end of the motion.

The side effect of interpolation is that it significantly slows down the motion because of the additional data points. Since we notice that the main characteristic of interpolation is *speed manipulation*, we need to also provide a way to reverse the process. That is, we need a way to *hasten* (make faster) the motion. We do this through a *sampling* process.

Sampling is usually used in digitization of analog data; where the length of the analog data is theoretically infinite, sampling only records information from the analog data at a certain rate or discrete points in time. Similarly, we only take the motion data (i.e. joint angle) from the motion signal only at a discrete rate. A rate of two means we are only taking the first point of every two data points in the motion data, essentially shortening the motion signal in half, thus speeding up the motion. At the sampling rate of three, we are taking the first point from every three data points, and so forth. Depending on the length of the whole motion, we may want to limit our sampling rate. A sampling rate of five on a motion signal with length ten, would result in a motion signal with length two. It may very well destroy the original motion, but it depends on the intention of the user. Eventually, the system itself will determine the sampling rate based on its deduction from the user interaction with the system.

Table 10.1 Hermite Interpolation results

Original Data Points	tension = 0, bias = 0	tension = 5, bias = 0	tension = -5, bias = 0
y = [10, 20, 15, 25]			
	tension = 0, bias = 5	tension = 0, bias = -5	tension = -1, bias = 1
			

10.7.2 Second Technique: Multiresolution Filtering

We proceeded with the multiresolution filtering approach. Multiresolution filtering is a filtering technique which successively passes a signal through a cascade of lowpass filters. Thus, the result of the multiresolution filtering is a set of lowpass components and bandpass components of the signal. These components can be used to modify the signal by adjusting and multiplying the gains of the bandpass components, or by blending the bandpass components with the bandpass components of a different motion

By adjusting the gains of each of these signal components, we can eliminate or amplify the effects of each component. Thus, when the components are added together, instead of obtaining the original signal, we would get a different signal but in general the resulting signal would be similar to the original signal. What this means in terms of motion signals, is that we can modify a motion, while still being able to observe the original motion in the new motion. In other words, the observer would still be able to identify the original motion. For example, let's take a 'waving arm' motion. And let's assume that the 'waving arm' motion is 'shaky', as if the actor is freezing or scared. When represented as signal, the motion appears to have a lot of noise. Or in other words, the motion signal has very pronounced high frequency components. By lowering the gains of the high frequency components (removing noise), the reconstructed motion signal becomes a lot smoother. In terms of motion, the motion becomes a 'relaxed', 'normal' motion instead of a 'freezing' or 'scared' motion. Of course, the opposite can be achieved conversely.

10.7.2.1 Effects Achieved using Multiresolution Filtering

We noticed several characteristics on the resulting motion signal from applying different gain values on each component:

1. Increasing high frequency components would accentuate any sharp changes in the motion signal. This results in motion with some sudden or abrupt movements. The converse (reducing the gains of the high frequency components), would reduce the noise in the motion signal; generally making the motion signal 'smoother', thus having a more continuous flow.

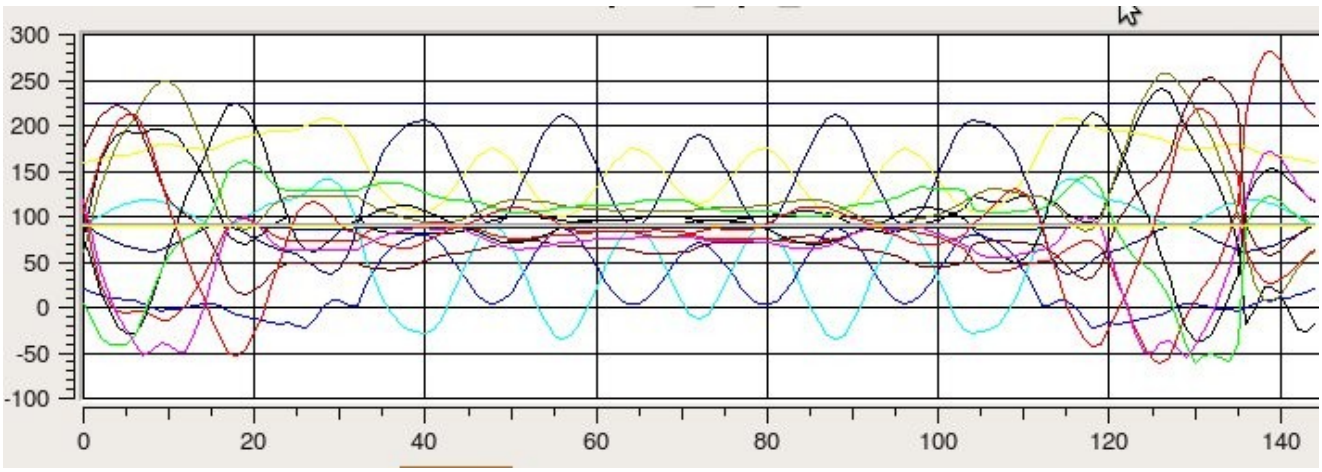
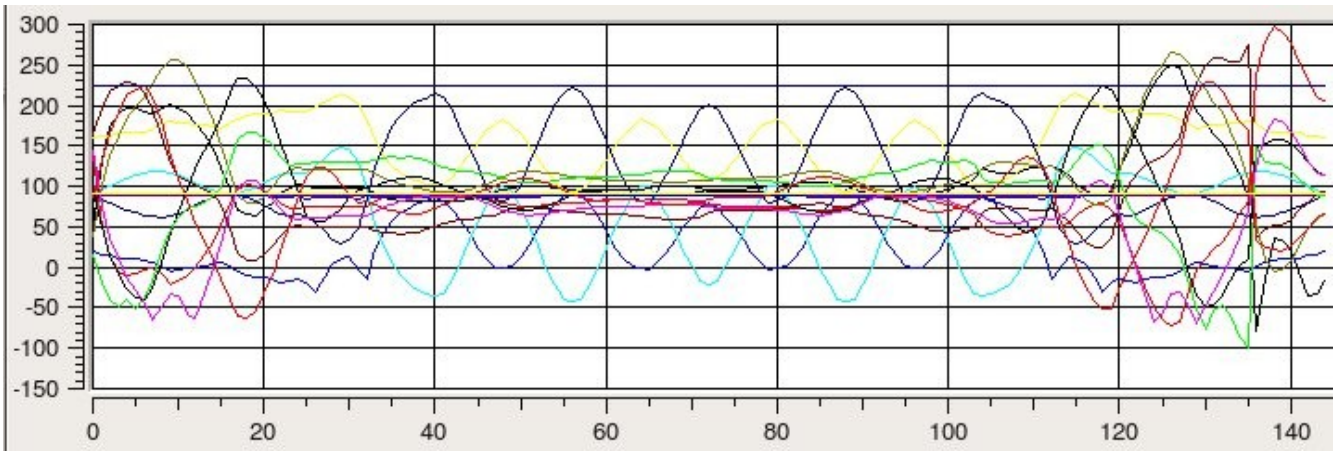


Figure 10.4 Original motion signal

a.



b.

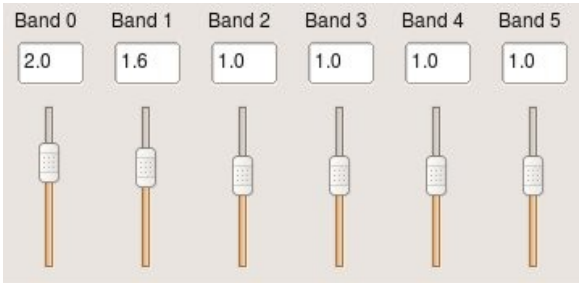
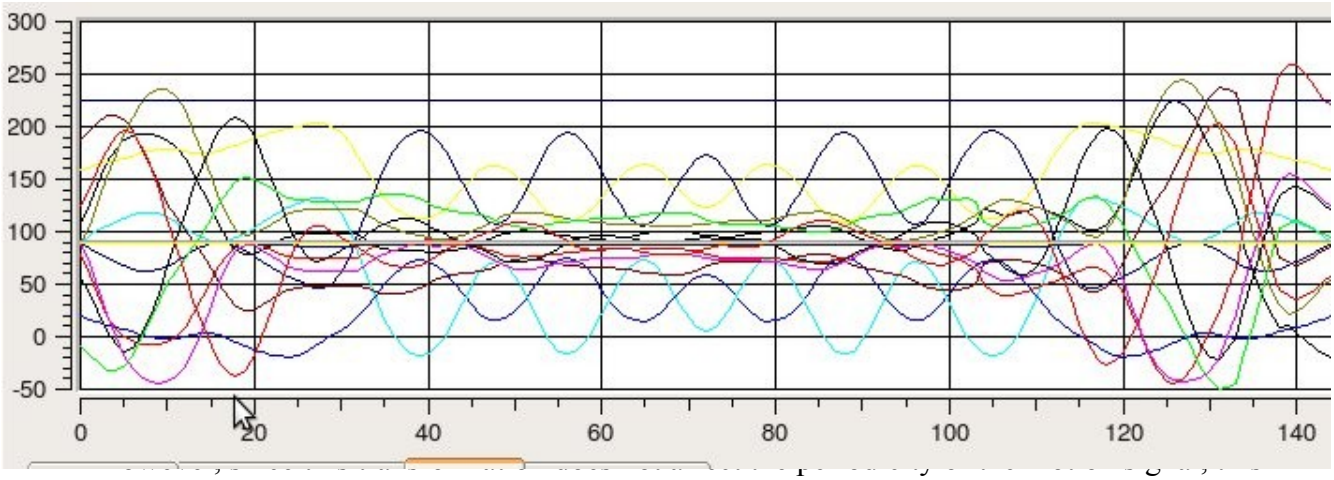


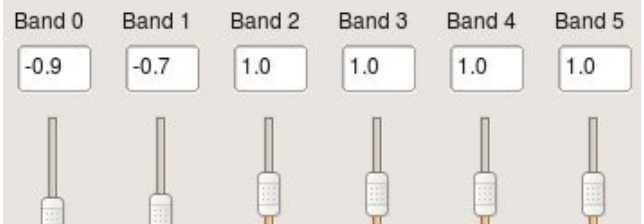
Figure 10.5 a) Increased high frequency components gains, b) Gains settings

a.

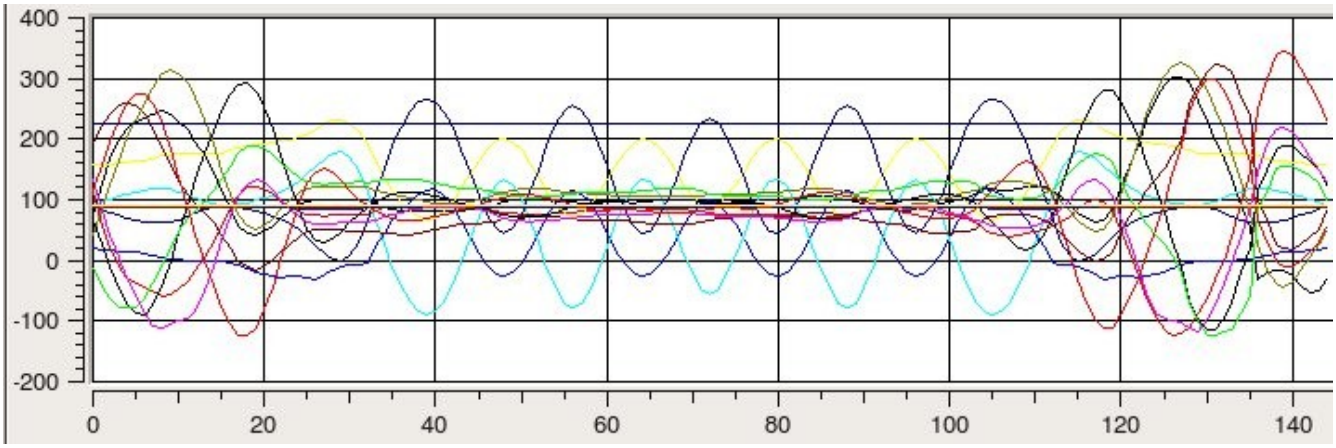
b.



action makes some parts of the movement 'bigger' and *seems* faster. This is because the bigger range of motion is being achieved in the same amount of time as the original range. If not



executed carefully, this would often lead to having the motion range out of the range of the servo.



a.

b

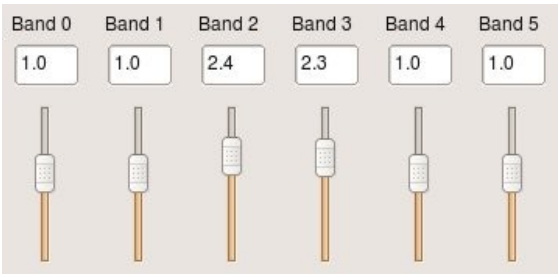
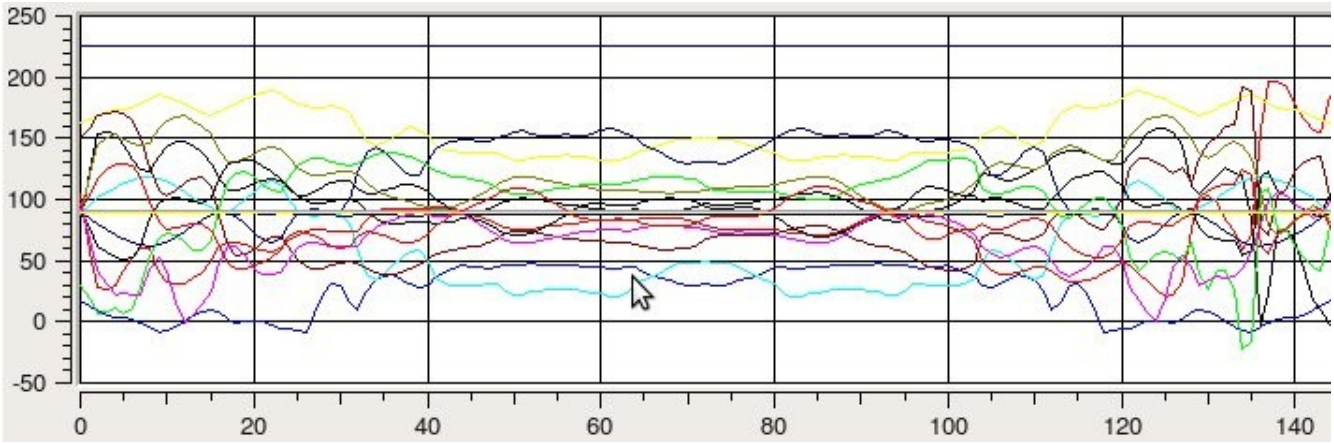


Figure 10.7 a) Increased middle frequency components gains, b) Gains settings

3. Reducing the middle frequency components can potentially 'kill' the motion. That is, when the motion signal has many 'hills' and 'valleys', reducing the gain on the middle frequency component would turn most of these hills and valleys into 'plateaus' or flat signals, especially on the middle of the signal. Thus, effectively eliminating the central part of the motion.



a.

b.

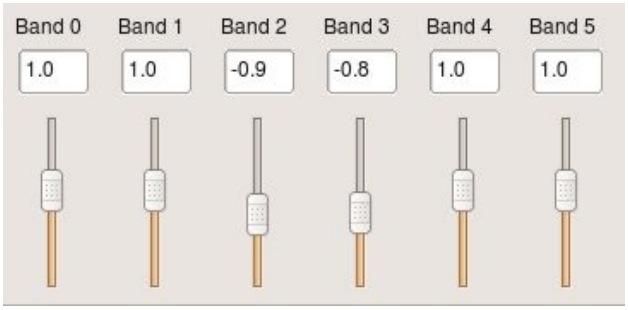


Figure 10.8 a) Reduced middle frequency gains, b) Gains settings

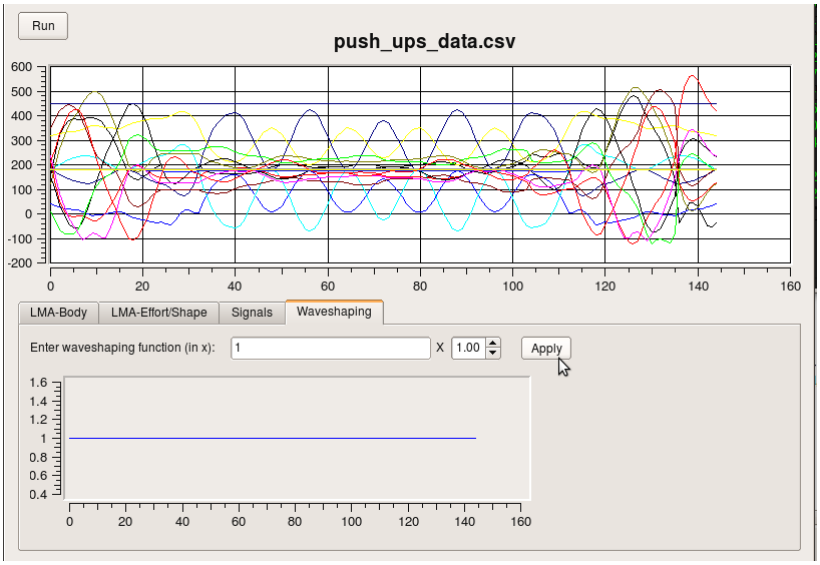
4. Increasing the lowest frequency component gain has the effect of increasing the amplitude (range of motion) of the beginning, middle, and end of the motion signal. In terms of motion, this has the effect of exaggerating anticipation, main action, and follow-through parts of the motion. Reducing the low frequency component gains tends to attenuate the middle part of the signal. While attenuating the middle part of the signal can also be achieved by reducing the middle frequency gains, reducing the low frequency gains attenuates the signal with less noise. Interestingly, adjusting the second-lowest frequency component gain has the opposite effects

from the effects of the lowest frequency component. That is, decreasing this gain amplifies most of the signals, but more obvious in the middle part of the signal. Increasing this gain actually attenuates the undulations on the middle part of the signal instead.

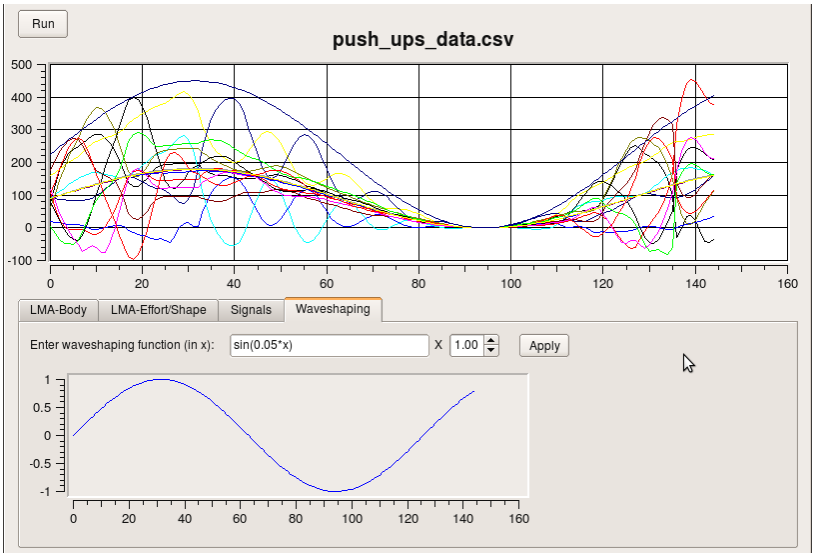
5. At its original state, the gains of all the components are one (1). If the gain of any of the components is adjusted to a negative value, after a certain point, the adjustment has the effect of *inverting* the signal instead of just reducing the effect of that frequency component of the motion signal. For example, we said above that reducing the middle or low frequency gain will create plateaus from the hills and valleys at the middle of the signal. If we keep reducing the gain (more negative value), the hills and valleys will start to appear again, but in the opposite direction of the ones in the original motion signal.
6. It was immediately obvious that there is no way to change the *duration* of the motion, or any part of the motion using strictly multiresolution filtering. This is because the transformation (i.e. from the gains) only provides scaling in the spatial domain, and not in the time domain.

10.7.3 Third Technique: Waveshaping

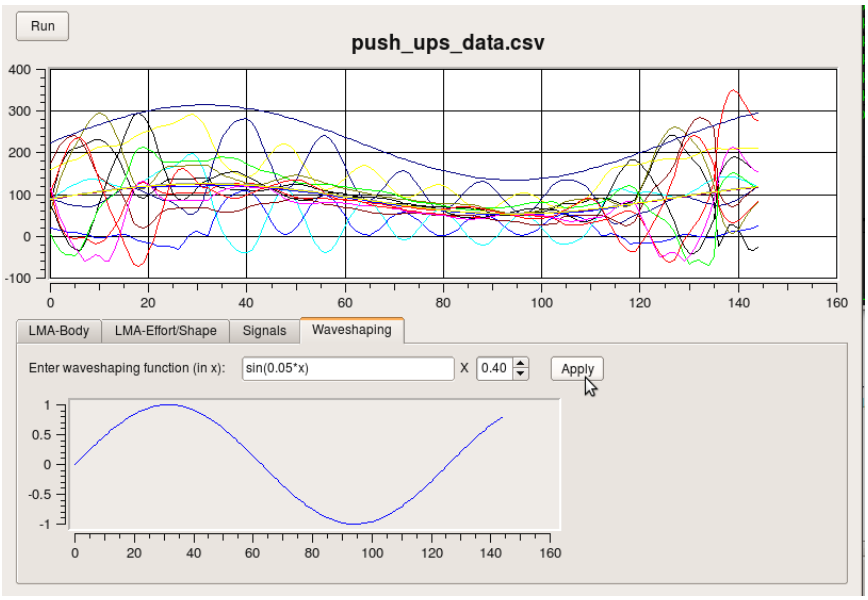
Waveshaping is essentially a method to modify a signal by superposing it with another signal or function. It gives the user a lot of freedom to modify the motion signal in the spatial domain, and *somewhat* in the time domain.


Run
push_ups_data.csv
LMA-Body
LMA-Effort/Shape
Signals
Waveshaping
Enter waveshaping function (in x):
1
X
1.00
Apply

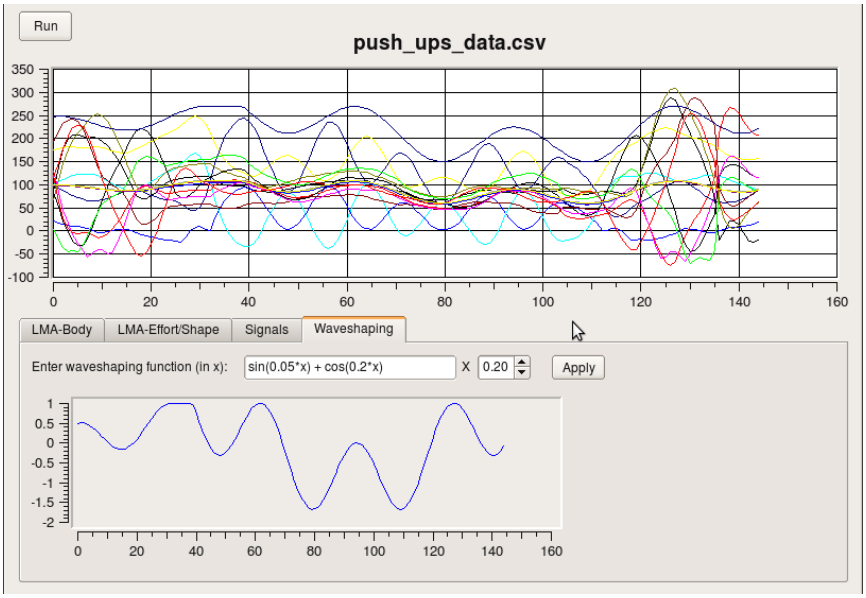
a.


Run
push_ups_data.csv
LMA-Body
LMA-Effort/Shape
Signals
Waveshaping
Enter waveshaping function (in x):
sin(0.05*x)
X
1.00
Apply

b.



c.



d.

Figure 10.9 a) Original motion signal, b) Applying waveshaping function with multiplier = 1.0, c) Applying waveshaping function with multiplier = 0.4, d) Applying a more complex waveshaping function.

The waveshaping function is mainly a superposition of a function to the motion signal; its main effect is creating an 'envelope' of the function on the motion signal. In the spatial domain, the users can easily modify the amplitudes of the motion signals by using a simple constant, linear functions, or low frequency periodic functions (or signals). In the time domain, the users can add undulations to the motion signal by adding a high frequency signal. However, we consider this not a 'true' time domain modification to the motion signal. A true time signal modification of the motion signal would be where we can make the signal shorter, longer, or moving certain parts of the signal to a different position (time). We see this capability more in the Hermite interpolation. Here, adding a high frequency signal to the motion signal does not change the length of the signal. The addition does create new components (i.e. movement) in the motion signal, which can be observed as creating 'faster' movements. But in reality, we are not accelerating the original motion, but instead *adding* faster motions to the original motion. For example, in the push up motion signal, by adding the high frequency signal, we create an undulating push up motion.

We could also create localized modifications to the motion signal using waveshaping. For example, we can create a waveshaping function that has peaks only at the beginning and the end of the function, while the middle remains flat. This can create or exaggerate the beginning of the motion (perhaps the anticipation/preparation phase), and the end of the motion (follow through phase). We can also use waveshaping to create a motion from scratch. Compared to the other methods (interpolation, multiresolution filtering), we consider waveshaping a rather 'brute force' approach to modify a motion.

Because waveshaping could be very destructive to the original motion signal, we applied a multiplier value to give the option to control the effect of the waveshaping function and preserve the original motion signal. The multiplier ranges from 0 to 1 where 0 means no effects at all, and 1 applies

(superpose) the waveshaping function completely. When the multiplier value is $0 < m < 1$, only m percent of the waveshaping function is being applied to the motion signal.

10.8 Some Conclusions

We conclude our investigation of the behaviours of the three signal transformation techniques by relating how they are best used to create the emotional and physical expressions we desire.

The Hermite interpolation is used to elongate a motion signal, smooth a linear motion signal, create overshoots, and create leading or lagging inflection points. In terms of motion, interpolation is used to make the motion slower, adding acceleration and deceleration, create anticipation and follow-through, and accentuate or deaccentuate a motion, respectively. Out of the three methods above, only interpolation has the ability to change the speed (i.e. duration) of the motion. To do the inverse, that is, to make the motion faster, the motion signal needs to be shrunk. We do this through *sampling*. The motion signal can be resampled with user-defined sampling rate between two to five, with sampling rate of one equals the original signal (no sampling).

Multiresolution filtering can be used to increase or decrease the range of motion. By adjusting the gains of the low frequency band through the high frequency band, we can change the motion range of the whole motion or only certain parts of the motion, or remove abrupt changes in the motion, thus making the motion smoother. For example, by increasing the gains on the middle to low frequency bands of the motion signal, we can make the motion range bigger.

Waveshaping is the most “intrusive” method – so to speak – between the three. Using waveshaping,

users can 'envelope' the original motion signal, and quickly distort the motion in different ways. Like multresolution filtering, waveshaping mostly affects the amplitudes (motion range) of the motion and has no direct effect on the speed of the motion. Because waveshaping is so destructive, we recommend only applying a fraction (0 – 30%) the waveshaping function to the original motion signal to preserve the original motion. However, if the waveshaping function is being applied to *create*a motion (i.e. there is no original motion to modify), then it is up to the user to determine how much effect should the waveshaping function has.

As we go back to our question: "*From all those available techniques, which technique can be used to express what emotion, and if it can, how to achieve it?* " we found the answer of the first part of the question when we discussed how emotions and physical states are being expressed in motion. It does not matter what kind of emotion is being represented by a signal transformation technique. What matters is that the intensity of the emotion that is driving the motion to be bigger or smaller, slower or faster. This also answers the second part of the question: any signal transformations that can manipulate the amplitude, length, and interpolation of the motion signal will be able to emulate any kind of emotion. This is because the type emotion is determined by the context of the interaction, and not how big or fast the movement is.

The more appropriate observation on how a motion expresses emotions should be on *gestures*. However, gestures are difficult to generalize because they are very culturally specific, and new gestures may be developed as time progresses. But, we recognize the importance of gestures as a way to explain or express an idea, concept, and emotions. Therefore, we selected a set of pre-programmed gestures that are general enough to be understood by most human culture.

CHAPTER 10 CONCLUSIONS

1. *Analysis and synthesis of humanoid robot motions are more effective and intuitive when the motions are represented and treated as signals. Thus, ...*
2. *By treating motions as signals, emotional and physical expressions can be easily embedded in the motions of humanoid robots. Because, ...*
3. *Emotional and physical expressions can be represented as a set of signal transformation functions. However, ...*
4. *Establishing context, preceeded by interaction, is crucial to understand the expressions of the emotional and physical states present in a motion, especially in the absence of facial expressions.*

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,* .

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:

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

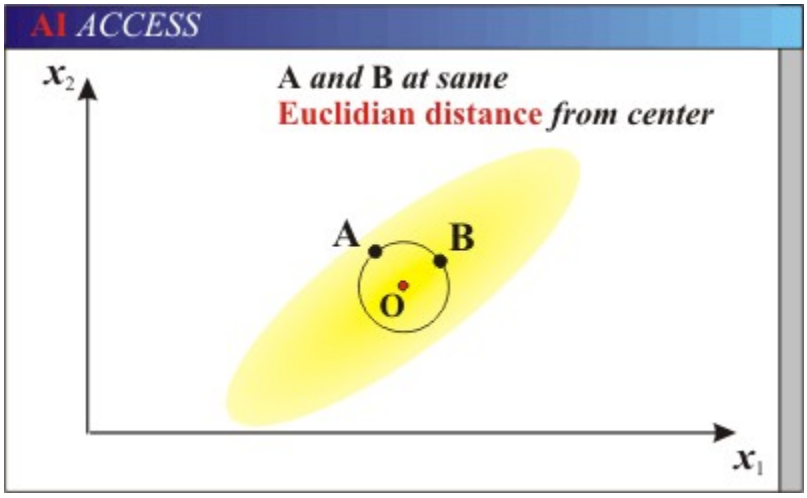
2. Mahalonobis Distance:

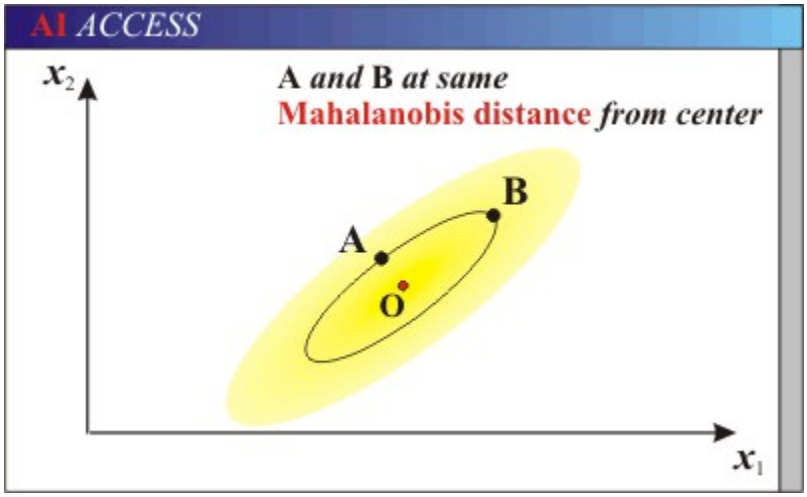
Maholonobis distance is a more general notion of distance than Euclidean distance. Here, the distance measurement of points in data set *p* is calculated against the correlation (i.e. covariance matrix) of data set *q*. The data set *p* is then considered as the *reference data set*. When the data sets 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:

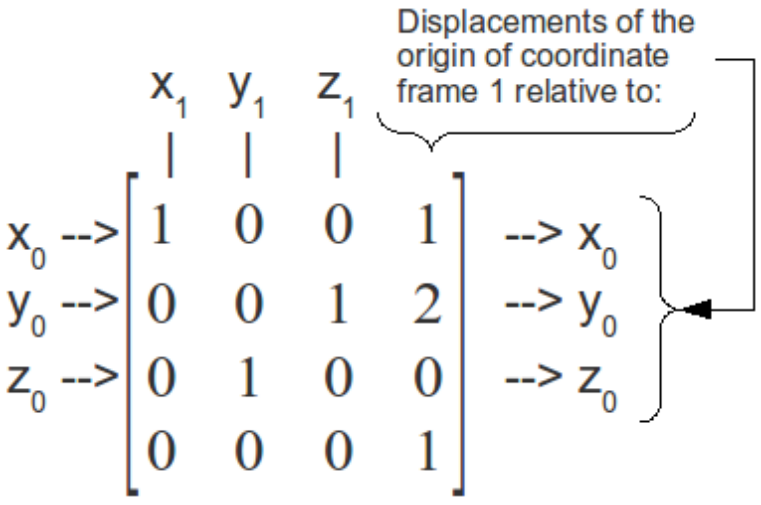
1. Think of the first three rows of the matrix as the axes of the first coordinate frame,
2. The first three columns as the axes of the second coordinate frame,
3. The values in the first 3 x 3 part of the matrix describes the *orientation* the axes of the second coordinate frame relative to the first 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.