IMITATION OF HUMAN ARM MOVEMENTS BY A HUMANOID ROBOT USING MONOCULAR VISION

by

Barış Kurt

B.S., Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2009

ACKNOWLEDGEMENTS

First I thank sincerely to my advisor Prof. Dr. H. Levent Akın, for his invaluable guidance, boundless patience and motivation. This thesis would not be possible without his contributions.

My appreciations go to my jury members, Prof. Dr. Ethem Alpaydın and Assist. Prof. Olcay Taner Yıldız.

I would like to thank all members of the Artificial Intelligence Laboratory (AILab). I thank Ergin Özkucur for his comments and support in programming, and Barış Gökçe for his support in kinematic modeling. I thank Çetin Meriçli, Tekin Meriçli, Serhan Daniş and Abuzer Yakaryılmaz for their helpful criticism and motivating comments.

I would like to thank my family, my parents Aynur Kurt and Şadi Kurt for their self-sacrifice, patience, trust and support. My special thanks goes to my sister Pelin Kurt for her support and motivation.

ABSTRACT

IMITATION OF HUMAN ARM MOVEMENTS BY A HUMANOID ROBOT USING MONOCULAR VISION

A sociable robot must have the capability to imitate agents around it. In a human society, people generally teach new skills to other people by demonstration. Hence, our artificial partners should be able to learn from us by watching what we do.

In this thesis, we programmed a humanoid robot to imitate human arm movements. The main problems we dealt with is the perception of the human arm movement and finding a corresponding motor sequence that will make the robot's movement as much the same as human's. We placed colored markers on the arm joints of the human demonstrator for tracking the arm. The physical difference between the human and the robot's arms made it difficult to directly extract the necessary joint angles for the robot. Instead, we employed a shared representation of movement which is neither the joint values of the human nor the robot. We made our comparisons on that common model.

We have conducted real world experiments where the robot watched a human demonstrator drawing horizontal and vertical lines and circle on the board. The robot successfully drew the lines and the circle.

ÖZET

İNSAN KOL HAREKETLERİNİN BİR İNSANSI ROBOT TARAFINDAN TEK KAMERA KULLANILARAK TAKLİT EDİLMESİ

İnsan toplumunda becerileri bireyden bireye aktarmanın en önemli yollarından biri taklittir. Robotların sosyalleşip insan toplumuna katılabilmesi için, etraflarındaki diğer insan ve robotları taklit etme yeteneğine sahip olmaları gerekmektedir.

Bu çalışmada, bir insansı robotu insan kol hareketlerini taklit etmek üzere programladık. Çözmemiz gereken başlıca problemler insan kol hareketlerinin tek kamera aracılığıyla algılanması ve robotun bu hareketlere en benzer hareketleri yapabilmesi icin gerekli olan kol eklem açılarını bulabilmekti. İnsan kol hareketini algılamak için kol eklemlerine renkli belirleyiciler yerleştirdik ve bunları izledik. İnsan ve robot kolundaki fiziksel farklar nedeniyle insan eklem açılarını birebir olarak robota aktarmak mümkün olmadığından, her iki kol icin ortak bir temsil yöntemi bulmamız gerekti. Daha sonra kollar arasındaki benzerliği bu model üzerinden karşılaştırabildik.

Yaptığımız üç gerçek dünya deneylerinde robot tahtaya dikey ve yatay çizgiler ve de bir daire çizen bir göstericiyi izledi. Daha sonra robot çizgileri ve daireyi başarıyla çizdi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

t	Time step at the execution of a discrete time system
α	The demonstrator agent
β	The imitator agent
S_t^α	State vector of agent α at time t
Z_t^α	Observation of agent α at time t
X_t^α	Action of agent α at time t

LIST OF ABBREVIATIONS

PET	Positron Emission Tomography
fMRI	Functional Magnetic Resonance Imaging
MSI	Mental State Inference
MNS	Mirror Neuron System
DOF	Degree of Freedom
ALICE	Action Learning for Imitation via Correspondence Between Embodiments

1. INTRODUCTION

Imitation is an important way of skill transfer in biological agents. Many animals imitate their parents in order to learn how to survive. It is also a way of social interaction. Imitative capabilities of the biological agents increase with the complexity of the agent. It starts from simple mimicry to intention and goal-based imitation. The most complicated form of imitation is observed in humans. This shows that imitation requires higher mental capabilities.

A sociable robot must have the capability to imitate the agents around it. In a human society, people generally teach new skills to other people by demonstration. We do not learn to dance by programming, instead we see other dancers and try to imitate them. Hence, our artificial partners should be able to learn from us by watching what we do.

In this thesis, we programmed a humanoid robot to imitate human arm movements. The main problems we dealt with is the perception of the human arm movement and finding a corresponding motor sequence that will make robots movement as much the same as human's. The physical difference between the human and robot arms makes the representation of the movement difficult. We employed a common representation and made comparisons of the movements of two different arms on that representation.

The rest of the thesis is organized as follows: The denitions of the imitation problem and solution methods discussed in the literature are given in Chapter 2. The specications and limitations of our problem denition and the underlying hardware and software platforms are detailed in Chapter 3. In Chapter 4, our methods and implementation details for the complete system are explained. In Chapter 5, experimental results are given. In Chapter 6, the results are discussed and some possible future works are pointed.

2. BACKGROUND WORK

Imitation is a commonly observed behavior among the animals, especially the ones with high cognitive skills such as dolphins [1] and great-apes, including human beings. The reason behind biological imitation is that it is a powerful tool for the transfer of knowledge between individuals.

Imitation is the ability of an agent (biological or artificial) to observe another agent which we call *demonstrator* or *model*, and act like it. This is an open-ended definition since "acting like" can be defined in many ways. If we consider the high level behaviors, trying to achieve the same goal can be defined as acting like, no matter how the goal is achieved. But if we consider lower level behaviors, *acting* like can mean executing motor commands in a way as similar as possible, allowed by the difference between the embodiments of agents.

Building a mixed robot-human society needs the robots to be able to adapt themselves to the society. Imitation is the predominant mechanism of doing this. A robot should observe the way humans and other robots act in a given context and act in the same way.

Another benefit of imitation is that it dramatically reduces the search space of the motor commands of an agent making a goal directed movement [2]. For an agent with 30 DOF, if a single joint command has only 3 possible values (increase, decrease, stand still), there would be $3^{30} > 10^{14}$ possible motor commands. Clearly it is impossible to make a search among the members of such a large command set. However, if the agent observes the action sequence of a demonstrator for doing the desired movement, and finds its corresponding action sequence that looks like the demonstrator's sequence, it can fine tune the sequence according to its own body dynamics. Hence, the search space will be reduced dramatically.

An example of search space reduction has been shown in an experiment where

a 7 DOF robotic arm learns to balance an inverse pendulum by observing a human demonstrator [2, 3]. The task is divided into two sub tasks, i.e, swinging the pendulum up and balancing it on the upright position. In order to achieve these tasks, the robot first learns its own model, the relation between its motor commands and the resulting hand and pendulum position, while it tries to balance the pendulum by itself. Later on, a metric is defined as the difference between the hand and the pendulum trajectories of the robot and the human demonstrator. The robot tries to minimize this metric, as it tries to make the pendulum. In five trials, the robot succeeds to bring the pendulum up, and balance it on upright position.

2.1. Robot Imitation Problem

Imitation is a complex problem, and can be divided into smaller problems. Dautenhahn and Nehaniv [4] identify five subproblems: who, when, what and how to imitate, and how to evaluate the success of imitation.

Biological agents employ imitation in order to acquire new skills, hence the imitator should select the best demonstrator. In order to make the selection, the imitator should examine the possible demonstrators and evaluate them with its own criteria. Once the imitator finds a suitable demonstrator, it should decide on when to imitate according to the time and place, in other words, the context. The imitator must also decide on which behavior of the demonstrator is going to be imitated. Will the imitator imitate only the goal of the demonstrator, or imitate in lower levels (subgoals, action sequences etc.). After selecting who, when and what to imitate, the imitator should use appropriate mechanisms to perform the imitation. This step is called the "correspondence problem" in which the behavior of the demonstrator and the imitator should look like to each other as much as they can, allowed by the differences in the embodiments and the affordances of the agents. In order to evaluate the success of the imitation, appropriate metrics must be defined to find the difference between the desired and performed actions and states. This evaluation can be done by either the imitator, the demonstrator or an external observer.

2.2. The Correspondence Problem

The success of the imitation is directly determined by how good it solves the correspondence problem. Nehaniv and Dauthenhahn [5] defines the correspondence problem as:

> Given an observed behavior of the model, which from a given starting state leads the model through a sequence (or hierarchy) of subgoals in states, action and/or effects, one must find and execute a sequence of actions using ones own (possibly dissimilar) embodiment, which from a corresponding starting state, leads through corresponding subgoals - in corresponding states, actions and/or effects, while possibly responding to corresponding events.

The imitator has to find a relation between the states and actions of the demonstrator and itself. Once this relation is found, the imitator can come to a corresponding state with the demonstrator, by doing the corresponding actions always reach to corresponding states. This relation can be found in different granularities. If the imitator finds a direct mapping of joint movements, it can mimic the demonstrators movement. However, if it finds a relation between high level actions, it can make an emulation of the demonstrator.

A good example of making correspondence using different granularities is given in a chess world experiment [6]. In the experiment, there are three agents with dissimilar embodiments and affordances, namely the Queen, Knight, and Bishop. The Queen is chosen as the demonstrator, since it has the better movement capability, and the Knight and Bishop are chosen as imitators. The Queen makes two consecutive movements, go left and up by three squares for each. The imitators try to imitate Queen's movement with two different granularities; the end-point granularity, in which the imitators try to go to the final destination of the Queen, ignoring the path the Queen had traveled, and the path-level granularity in which the imitators try to follow the exact path the Queen travels. When using end-point granularity, the bishop goes to the final destination in a single move: going up-left diagonally by three squares. On the other hand, when it uses path-level granularity, it makes zig-zags to make it's path as much the same as that of the Queen's.

2.2.1. Metrics for evaluating the success of correspondence

In order to evaluate the success of correspondence between the agents, three different and complementary metrics have been proposed, namely state, action and effect metrics [7]. The *state metric* evaluates the similarity between the body states of the agents, where the action state evaluates the similarity between the changes of states. If the agents have similar embodiment, these two metrics can be defined as:

$$\mu_{state} = \sum_{i=1}^{n} |s_i^{\alpha} - s_i^{\beta}| \tag{2.1}$$

$$\mu_{action} = \sum_{i=1}^{n} |a_i^{\alpha} - a_i^{\beta}| \tag{2.2}$$

where s_i^{α} and a_i^{α} are the state and action of the i^{th} joint of agent α respectively and n is the number of joints. Similarly s_i^{β} and a_i^{β} are the state and action of the i^{th} joint of the agent β. μ_{state} and μ_{action} are the similarity values of the states and actions of agents α and β.

Whenever the agents have dissimilar embodiments, a correspondence matrix is used to make a correspondence mapping. If the demonstrator has n DOFs and the imitator has m DOFs, a $n \times m$ correspondence matrix can transform the state vector of the demonstrator into a corresponding state vector which has the same size as the state vector of the imitator. After the transformation, the action and state metrics can be used as in equations 2.1 and 2.2. By defining the correspondence matrix, many different mappings can be defined: identity, mirror, one-to-many etc. Clearly this matrix can be obtained by various learning algorithms such as reinforcement learning, and learning this matrix can be an interesting research problem.

The last metric proposed by the authors is the *effect metric*, which measures the similarity of the results of the agents' actions. If they are trying to manipulate an object, the object's position and orientation can be used as the effect matrix. Likewise, if the agents are moving, their own positions can be used as effect metric.

Depending on the granularity, a combination of these metrics can be used. For example, if a robot dog is trying to imitate a humanoid robot playing soccer, then there is no need to make a mapping between the bodies of the two robots, instead, a mapping between their behaviors and goals can be made. In this case, the effect metric gets the highest priority, whereas the state and action metrics have no importance. On the other hand, a robot trying to imitate a human manipulating an object by his hand should use all of the metrics.

2.2.2. Solutions to the Correspondence Problem

A generic imitation framework ALICE (Action Learning for Imitation via Correspondence between Embodiments) [6, 8] has been proposed for the solution of the correspondence problem. The framework creates a *correspondence library* that relates the actions, states and effects of the demonstrator to the actions (or action sequences) that the imitator is capable of. The library stores key-action sequence pairs where the key is composed of perceptive and proprioceptive data. Whenever a perception is received, the key is formed and its corresponding action sequences is searched in the library. Since the perceptions are continuos values, it is impossible to find a perfect match between any two keys. Instead, if the keys are close to each other with some similarity, it is assumed that they match. This way, a kind of generalization is achieved.

There is a need for a generating mechanism that creates action sequences for a given key, since the library initially does not contain any keys, and may need to be updated when the context is changed. This generating mechanism is independent of the framework, and can be anything like an inverse kinematics engine, or a random generator, etc. Once the action sequence is generated by the generating mechanism, it is compared with the sequence proposed by the library. If the one proposed by the library is found to be better, the agent choses that action. Otherwise, the new generated action is chosen and the library is updated.

The comparison of the proposed actions are done with a metric. The choice of the metric also determines the level of imitation. If the metric compares the effects of

actions, the imitation can be characterized as goal emulation, if it compares the actions and states, the imitation looks more likely to mimicry. Although explicitly mentioned, the metric should use a forward model which makes an internal simulation of the proposed actions, since the actions are evaluated before they are executed. Forward models are explained in section 2.4.1.

This framework has been tested in the chess world problem [8] and imitation between simulated robotic arm manipulators [6]. In the robotic arm experiment, imitators with different embodiments have imitated a demonstrator. Although they all have different DOFs and different arm lengths, it has been observed that the imitator imitated the behavior of the demonstrator very accurately. Also the online change in the embodiment of the imitators (which simulates the growth of a person) is handled by the framework, and the system adapted itself to the changes. The authors claim that this can be classified as a cultural transmission of behaviors between the individuals of a society. But this social dimension of imitation is out of the scope of this thesis.

Although the ALICE framework targets the correspondence problem directly, other proposed methods implicitly target the same problem. The computational models proposed so far targets how the imitation is performed and how it is evaluated. The social aspects of imitation such as choosing a good demonstrator, choosing when to imitate, and what to imitate has been kept out of scope. The reader should keep this in mind, while reading the other computational approaches.

2.3. Imitation from a Neuroscience Perspective: Mirror Neurons

Computer scientists should understand how biological agents imitate other agents, before trying to make their own computational models for imitation. Since imitation requires visual perception, mapping of visual representation to motor representation and executing the final motor commands, the circuit starting from the visual cortex to the primary motor cortex of the brain should be studied in detail. Neuroscientists are trying to discover the properties of these cortices in the brains of humans and macaque monkeys, which are the closest biological relative of humans. The investigations in

monkeys are done in more detailed way since invasive methods can be used. Single neuron activities in the monkey brain can be monitored by inserting electrodes in it. On the other hand, experiments with humans are done with non-invasive methods such as positron emission tomography (PET) and functional magnetic resonance imaging (fMRI) techniques. The results of these experiments gives ideas only about the functioning of a large group of neurons in the specific parts of the brain. Here, we will try to explain the brain activity in the monkey brain and after that the hypothesis about the human brain areas.

In order to understand the imitation in animals, one of the most important parts of the monkey brain is the rostral part of the inferior premotor area 6, which is also called area F5. This area is active during the planning and execution of hand and mouth movements [9]. The premotor cortex is responsible for the high level descriptions of motor acts, and control of proximal muscles. The execution of the selected motor act is done in the primary motor cortex. The importance of area F5 comes from the fact that it is directly connected to the visual area AIP, which extracts the affordances of objects that are visually percepted. F5 is assumed to translate these affordances coming from AIP in visual terms into appropriate motor terms [9, 10]. F5 neurons show two very important characteristics. First they discharge selectively for specific types of actions. Some neurons discharge during precision grip where only the thumb and the index finger is used to grip a small object, while some others discharge during a power grasp, where whole fingers and the palm of the hand is used to grasp a bigger object. Secondly, there is a temporal relation between F5 neuron firings. Some of them discharge during the whole grasping action, some are active during the opening of the fingers, some are active during the contact of the fingers with the object etc. These two properties may indicate that F5 neurons form a motor vocabulary, in which the words of the vocabulary are populations of neurons [11, 10].

Murata et. al. [12] discovered that some of the F5 neurons also respond to visual stimuli. These neurons, which are active during grasping an object are also active when the object is only visually presented but not manipulated. These neurons are called *canonical neurons*. The most important property of these neurons is that there

is a clear congruence between the motor acts and the visual properties they respond to. The neurons which fire during a precision grip are also active when a small object which can be manipulated with precision grip is observed, but they do not fire for observing other objects. It can be proposed that these neurons are defining the objects in motor terms.

Another type of neurons, called *mirror neurons* [11], was found in the premotor cortex that respond to both visual and motor stimuli. Unlike canonical neurons, they do not discharge when an agent observes an object. They discharge when an action towards an object is perceived. Additionally, the action without object (mimicry) does not make these neurons discharge. These neurons are also selective and show congruence between the motor acts and the visual properties they respond to. There are two types of neurons according to their congruency levels. Strictly congruent mirror neurons respond to visual stimuli unless it is exactly the same as the motor stimuli they respond to. For example if they fire when the monkey is performing precision grip, they fire only if a precision grip of another agent is observed. On the other hand, broadly congruent neurons do not need such a strict relationship between the motor and visual stimuli. A strictly congruent neuron which fires when the monkey is performing precision grip, may fire when any type of grip is observed. Finally, in addition to visio-motor mirror neurons, audio-motor mirror neurons are also present [13]. Besides responding to the observation of an action, they respond when the sound of the action (for example ripping a paper) is heard.

Since we will investigate some computational models for imitation based on the findings in neuroscience, we have to understand the hypothetical circuits on the monkey brain. The mirror neurons in the monkey are primarily found in area F5. The input to F5 mirror neurons are thought to be coming from Superior Temporal Sulcus (STS). The importance of STS is that, the neurons in this area show the visual properties of mirror neurons. They fire selectively when an action is perceived, but they do not fire when the action is performed. It has been proposed that STS makes the first identification of movements by getting input from the visual cortex [10].

After the discovery of mirror neurons in macaque monkeys, experiments for detecting these type of behavior in humans were conducted [14, 15]. It has been found that there are mirror-like activities in the areas Brodmann 40 and 44 in the human brain. The area Brodmann 44 is considered to be the human homologue of the monkey's area F5. Additionally, this area covers some part of the Broca's area, which is responsible for speech generation.

There are some studies that show that the mirror neurons also help to understand the intentions of other agents. Intention can be described as the high level goal of an agent. For example, if an agent is reaching for a cup, its immediate goal is to grasp the cup, but its high level goal may be to drink the tea in the cup or to clean the cup etc. The clue for extracting the intention of the agent is the context in which the action takes place. If the cup is full of hot tea, the intention is probably to drink tea. On the other hand, if the cup is empty and dirty, the intention to grasp the cup is probably cleaning it. Iacoboni et. al. [16] did experiments by showing grasping videos to the subjects and recording their brain activity using fMRI. Three types of videos were shown: context only, action only and action in context. The differences between the signals collected in these three scenarios showed that there was a significant increase in the signals emitted from the inferior frontal cortex, where the mirror neurons is assumed to exist. The authors concluded that mirror neurons may help to extract the intentions of the observed agent.

It is also proposed that people understand other people by simulating their emotional states internally [17]. An experiment on emotional simulation was done by Wicker et. al. [18]. The subjects observing facial expressions of other people showed a mirror-like neural activity in their cortical regions which are responsible for emotion. Furthermore, there are studies that show that there are less mirror-like neural activities in the premotor areas of autistic children compared to the normal children's [17, 19]. Autistic patients lacks the ability to understand other's emotional state.

2.4. Computational Approaches

2.4.1. Mental State Inference (MSI)

Inspired from the dual role of premotor cortex, a mental state inference method has been proposed. The method uses the control mechanisms developed for the manual manipulation to internally simulate the action of the demonstrator. This simulation leads to the inference of the mental state of the demonstrator [20]. It has been tested in a simulation environment, where a robotic arm tries to reach certain points on a board, and the observer watching it in order to infer it's mental state (the point which the arm is aimed to reach).

The mechanism for the control of the agent is inspired from the brain and divided as visual, parietal, premotor and primary motor cortices. The visual cortex process the sensory input and extract visual features, and the parietal cortex uses these features and current goal to calculate a control variable X (equation 2.6). If the goal is to reach for a point, than the control variable becomes the distance between the point and the tip of the manipulator. Then, the reaching goal can be defined to minimize this distance. After the control variable is extracted, premotor cortex calculates the difference between the desired and the current states of the control variable and tells the primary motor cortex the desired change in the position of the hand (equation 2.4). A copy of the command is given to a forward model FM and it calculates the effect of the command for control, since the visual feedback of the command will be delayed (equation 2.3). The primary motor cortex calculates the necessary motor commands, and moves the arm (equation 2.5).

$$X^n_{j,pred} = FM_j(\Delta\Theta^{n-1}_j, X^{n-1}_{j,pred}) \quad (2.3)$$

$$\Delta\Theta^n_j = MP_j(X^n_{j,pred}, X^{n-delay}_j) \quad (2.4)$$

$$\Theta^{n+1}_j = FD(DC_j(\Delta\Theta^n_j + \Theta^{n-1}_j, \Theta^{n-1}_j)) \quad (2.5)$$

$$X^{n+1}_j = CV_j(\Theta^{n+1}_j) \quad (2.6)$$

In the mental state inference mode, visual and parietal cortices calculates the control variable (which translated according to the observer), from the visual perception of the demonstrators action. The control variable is given to the premotor cortex, and according to the estimated mental state of the demonstrator, a motor command is produced. However, this command is not executed but instead, used by the forward model to calculate the possible next state of the demonstrator. When the next visual feedback about the demonstrator arrives, it's compared with the estimation of the forward model, and the estimated mental state of the demonstrator is updated accordingly. This work has shown that the internal simulation of the demonstrator by using the observer's own control mechanisms can be used to extract the mental state (intention) of the demonstrator.

$$X^n_{i,observed} = CV_i(Actor) \quad (2.7)$$

Mental simulation (m=1,...,n)

$$X^m_{i,pred} = FM_i(\Delta\Theta^{m-1}_i, X^{m-1}_{i,pred}) \quad (2.8)$$

$$\Delta\Theta^m_i = MP_i(X^m_{i,pred}) \quad (2.9)$$

In order to infer the mental state of the actor, the observer has to make a search among the possible mental states. If the search space is discrete, an exhaustive method can be used. The observer can simulate all possible mental states internally and com-

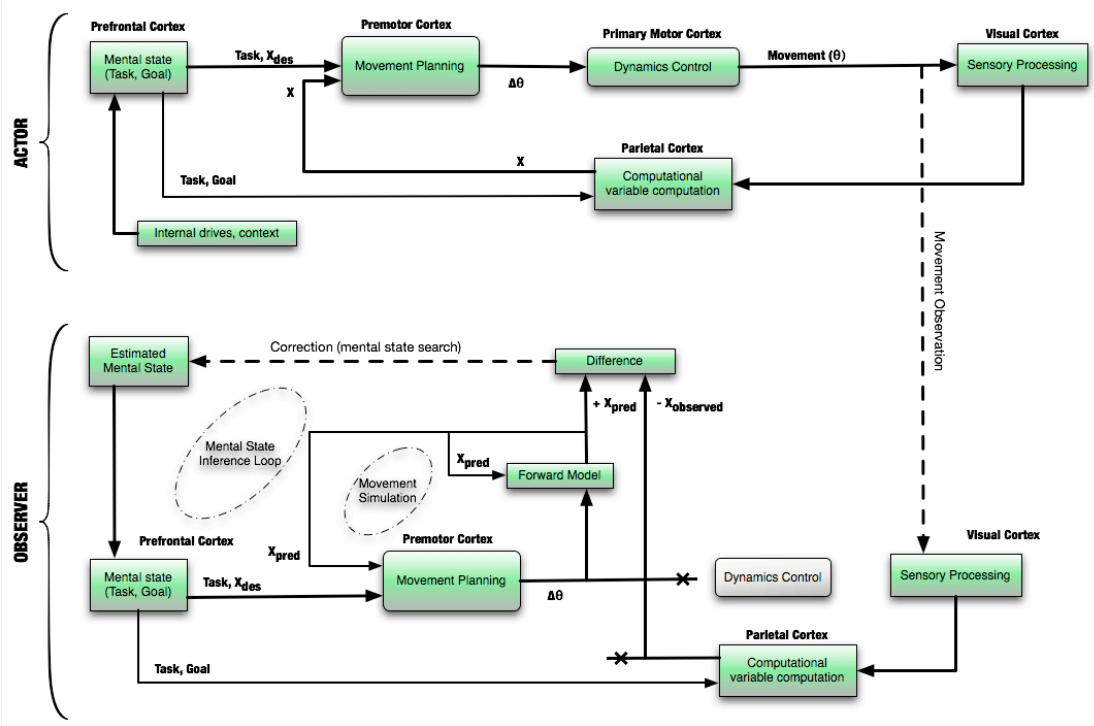


Figure 2.1. MSI Model [20]

pare their control variables with the observed control variable. The mental state whose control variable show the maximum similarity to the observed control variable is said to be the mental state of the actor.

If the search space is continuous, which is the case in most of the real world situations, an exhaustive search can not be applied. But a stochastic gradient descent method can be applied. Starting from a random mental state (for example a random point on a 2D surface for the manipulator to reach) the observer can make random movements and try to converge to the actual mental state.

This method is applied in a two dimensional simulator where a robotic hand tries to infer the mental state of the actor (the point which the actor tries to reach).

Algorithm *ExhaustiveMentalStateSearch*

1: $T_k = S_k = []$ {T_k and S_k are sequences observed and simulated control variables for mental state k}

2: **repeat**

3: Pick next possible mental state j

4: x_j^i {Calculate control variable for hypothesized mental state j}

5: $T_j = [T_j, x_j^i]$

6: Mentally simulate movement with mental state j

7: Compute X_j

8: $S_j = [x_j^0, x_j^1, ..., x_j^N]$ {N is the control variables collected during movement observation}

9: $D_N = \frac{(1-\gamma)}{1-\gamma^{N+1}} \sum_{i=0}^{N} (x_{sim}^i - x^i)^T W (x_{sim}^i - x^i) \gamma^{N-i}$ {$x_{sim}^i \in S_j$ and $x^i \in T_j$}

10: **until** Movement is finished

11: Return j_{min}

Figure 2.2. The exhaustive mental state search algorithm.

Algorithm *StochasticGradientDescentMentalStateSearch*

1: Set A to a random mental state
2: $MaxIter = 20$
3: $D = 1e20$
4: **repeat**
5: Add actor's current kinematics in sequence K
6: **for** $i = 1$ to $MaxIter$ **do**
7: Generate random perturbation ΔA
8: $A = A + \Delta A$
9: $T_A = [x^0, x^1, ..., x^N]$
10: $D_N = \frac{(1-\gamma)}{1-\gamma^{N+1}} \sum_{i=0}^{N} (x^i_{sim} - x^i)^T W (x^i_{sim} - x^i) \gamma^{N-i}$
11: **if** $D_{last} < D_N$ **then**
12: $A = A - 1.2\Delta A$ {Apply 20% penalty}
13: Go to step 7
14: **end if**
15: **if** $rand() < 0.1$ **then**
16: Go to step 7 {Make a random move}
17: **end if**
18: Go to step 8 {If perturbation is good, use it once more}
19: **end for**
20: **until** Movement is Finished
21: Return A

Figure 2.3. The gradient descent mental state search algorithm.

3. PROPOSED APPROACH

The goal of this thesis is the imitation of human arm movements by a humanoid robot. This goal is composed of two sub-goals: perceiving the human motion sequence and executing the necessary motion commands to replicate the perceived action. Considering the five major problems of imitation [4], we are dealing only with the "how to imitate" problem. The selection of demonstrator, the time and reason of imitation is out of our scope. After a successful imitation, the robot is not supposed to understand the intention of the action. Our scope is limited with programming a robot, which moves its arm in a way as much similar as the human demonstrator's.

Imitation starts with the perception of the demonstrator. Many types of sensors can be used for collecting information about the demonstrator's body posture. Cameras and motion capture suits are the most common ones. In this work, we used the monocular vision system of our humanoid robot. Using a single camera for the action detection introduced another problem. The estimation of the depth of the objects in the image with a single camera is difficult.

The correspondence problem is the main problem of this task. The arm of a human differs from the arm of the robot that we use. The biggest difference is that the humans have ball-and-socket type of joints in their shoulders while robots usually have two rotational shoulder joints. We have seen that it is hypothesized that the mirror neuron systems are used for expressing the movement of others in the agents own motor vocabulary [10, 11]. A similar approach should be used by the robot to represent the arm movements of the human demonstrator.

In almost all of the robotic imitation experiments, the imitator robots are assumed to extract the exact joint positions of the demonstrator. But it is less probable that the biological agents extract each others exact joint positions. Instead, the usage of qualitative representations are more likely. We have also done experiments on using qualitative representations for joint positions in a simulator environment.

3.1. The Stick Figure

In order to solve the correspondence problem, the arm is modeled as a stick figure as shown in Figure 3.1. It can be the arm of a robot or a human regardless of the number and types of joints, provided that the arm is composed of two parts. In the experiments, both the human and the robot arms have two parts: upper and lower arm. But the shoulder joint of the human arm is of ball and socket type. On the other hand, the robot's shoulder has two rotational joints, allowing the arm turn in two perpendicular axes. The stick figure is represented by the lengths of the upper and lower arms and 2 angles for the orientations of each part. These orientation angles are neither the joint angles of the robot nor the human arm. This allows us to compare the position of the two arms in a common representation different than the internal representations of the human and robot. This common representation solves the correspondence problem between the human and robot arms.

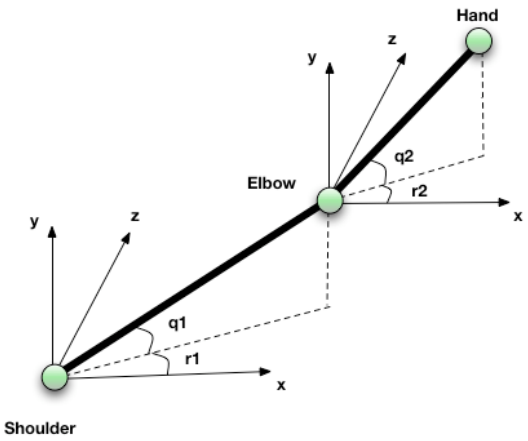


Figure 3.1. The stick figure

The stick figure of the robot itself is calculated by the forward model. The forward model takes the joint angles of the robot as input and calculates the stick model. Let $s = [r_1, q_1, r_2, q_2]$ be the vector of the stick figure and $j = [j_1, j_2, j_3, j_4]$ be the vector of joint angles. Then the forward model makes the transform:

$$s = FM(j) \tag{3.1}$$

Where FM is the forward model. First, it calculates the positions of the elbow (according to the shoulder) and hand (according to the elbow) by using forward kinematics. Let $p_e = (x_e, y_e, z_e)$ and $p_h = (x_w, y_w, z_w)$ be the elbow and wrist points in cartesian coordinates respectively. Then,

$$p_e = T^e j \tag{3.2}$$

$$p_h = T^w j \tag{3.3}$$

where T^e and T^w are the transformation matrices for elbow and hand. These matrices are calculated by using the Denavit-Hartenberg (DH) convention [21]. The stick figure is formed by transferring these points into spherical coordinates.

$$r_1 = \arctan\left(z_e / \sqrt{x_e^2 + y_e^2}\right) \tag{3.4}$$

$$q_1 = \pi/2 + \arctan\left(y_e / x_e\right) \tag{3.5}$$

$$r_2 = \arctan\left(z_w / \sqrt{x_w^2 + y_w^2}\right) \tag{3.6}$$

$$q_2 = \pi/2 + \arctan\left(y_w / x_w\right) \tag{3.7}$$

The stick figure of the demonstrator can be constructed after the shoulder, elbow and wrist points are located on the 3D space. Locating these body parts can be done by various methods, for example by vision processing, placing sensors on the body parts. Once the imitator constructs the stick figure of the demonstrator, it can replicate the same stick figure by setting its arm joints to necessary values.

The replication of the stick figure differs according to the arm configuration of the robot. The programmer either should find a direct method for the robot to replicate the stick figures, or should employ a training procedure for the robot to learn how to replicate the sticks.

4. EXPERIMENTS

We did our experiments in an indoor environment under stable lightning conditions. We placed colored markers on the joints of the demonstrator for simplifying the vision processing. This type of color based perception required the stable lightning conditions as our training method for color detection does not tolerate variances in light intensity. The hardware and software platforms will be described in the following section.

4.1. Hardware Platforms

4.1.1. Aldebaran Nao Humanoid Robot

The main robot for our experiments is the Nao humanoid robot, produced by Aldebaran Robotics company [22]. The specifications of the robot are as follows:

- 21 degrees of freedom.
- 2 CMOS cameras with 640x480 maximum resolution at 30 frames for second.
- 4 ultrasound distance sensors on the chest.
- Loudspeakers and microphones on the head.
- Pressure sensors and bumpers at the feet.
- x86 AMD Geode 500 Mhz CPU.

In our experiments we controlled 10 of the joints (2 joints for the head pan and tilt, and 4 joints on each arm). The remaining 11 joints on the hips and legs were just released. Additionally we used only one of the cameras. Although Nao has 2 cameras, they can not be used as a stereo vision system. First, they are aligned vertically on the head and their field of views do not overlap; furthermore only one camera can be active at a time, according to robot's hardware limitations. The images were taken from the upper camera in a 320x240 resolution at 15 frames per second and the color space is YUV442. Additionally, we used the speech synthesis API provided by Aldebaran to

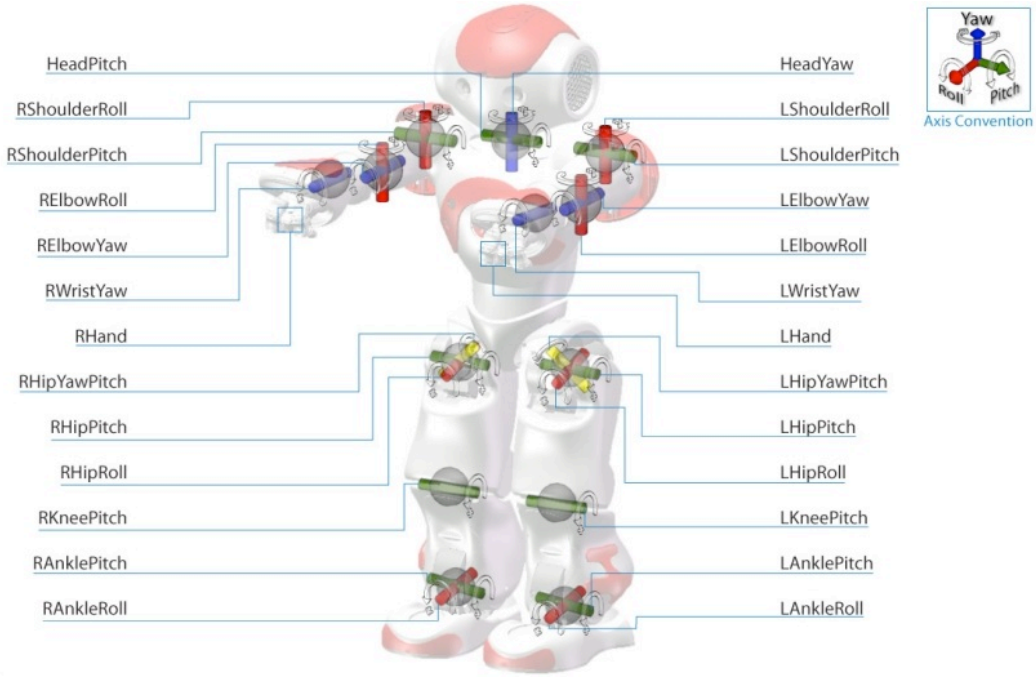


Figure 4.1. Overview of the Aldebaran Nao Robot [22]

interact with the robot in human-robot experiments.

4.1.2. Sony Aibo 4-Legged Robots

Aibo is the general name of 4-legged robot dogs, produced by Sony [23]. We used two models of Aibo's: ERS-210 and ERS-7. These two models have similar embodiments. They have 18 degrees of freedom: 3 joints for each leg, 3 joint for the neck, 1 joint for the mouth and 2 joints for the tail. Their differences come from their cameras and processing powers. ERS-210 has a CMOS camera with 176x144 resolution and 57.6° (horizontal) and 47.8° (vertical) field of view. The camera provides 25 frames per second. ERS-7 has a CMOS camera with 208x160 resolution and 56.9° (horizontal) and 45.2° (vertical) field of view. It provides 30 frames per second. They both have 64 bit RISC processors but ERS-210's processor runs at 384 MHz and ERS-7's runs at 576. The robots are running on *Aperios* real time operating system designed by Sony and programmed via *OPEN-R* software development kit. We used Aibo's in the simulator environment together with Nao's for correspondence experiments.

4.2. Software Platforms

We used the C++ library developed for the robot soccer team Cerberus called *BOUNLib* [24]. This library provides the necessary tools such as data structures, wireless communication, image processing, neural networks, fuzzy inference engines. Additionally we used the monitoring tool *Cerberus Station*. The station listens to the messages from the robot or simulator and show their content in appropriate monitors. The raw or processed images can be viewed instantly with this station. Furthermore it helps the development of some vision algorithms and it provides a mechanism for color calibration. The station is extensible by introducing new plugins. We have written an imitation controller plugin for monitoring the imitation activity in the robot.

In addition to the *BOUNLib*, we have developed a *RobotLib* which includes controllers for Aibo and Nao robots. All of the algorithms used in the experiments were included in this library. The controllers for the real and simulated robots are just wrappers which include the robot controller inside the *RobotLib*. This way we used our controllers both in real world and simulator experiments, by saving a lot of development time. The general structure of *BOUNLib*, *RobotLib* and *CerberusStation* is shown in Figure 4.2.

4.2.1. Webots Simulation Environment

Webots is a development environment used to model, program and simulate mobile robots [25]. It is a 3D simulator where the users can model their own robots, as well as use pre-modeled ones. In our experiments we used Aibo ERS-210, Aibo ERS-7 and Aldebaran Nao models. Webots includes these robot models ready to use 4.3. The user has to write his own controller by using Webots C++ or Python API. We modified the robot models in order to add colored markers in the arm to help image processing. This simulation platform was used for prototyping and testing of algorithms. Additionally we conducted several experiments using Aibo and Nao robots to solve correspondence problem.

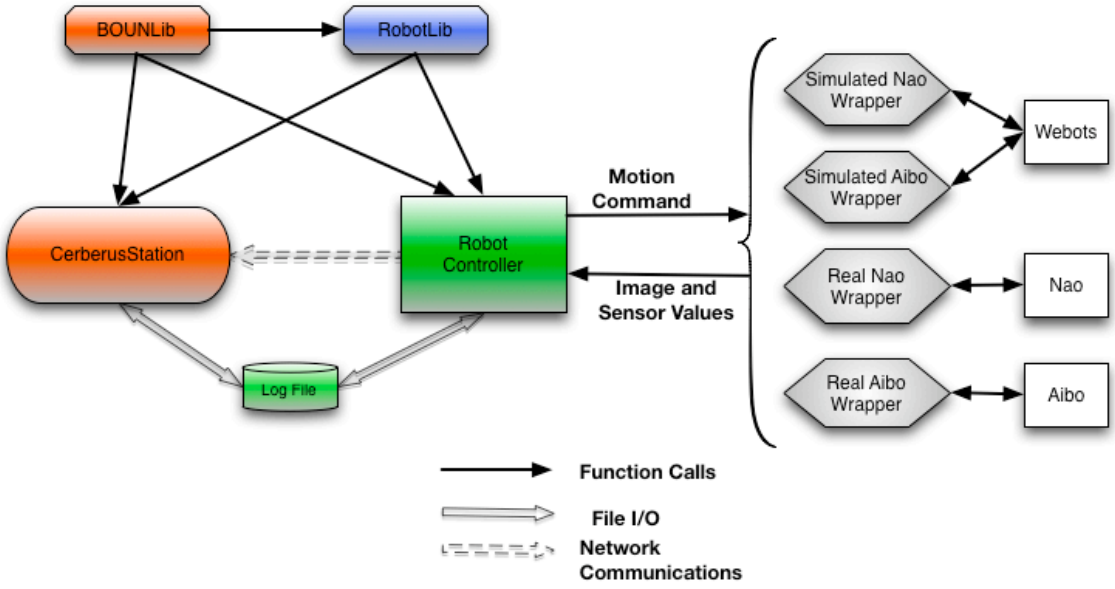


Figure 4.2. General Software Architecture

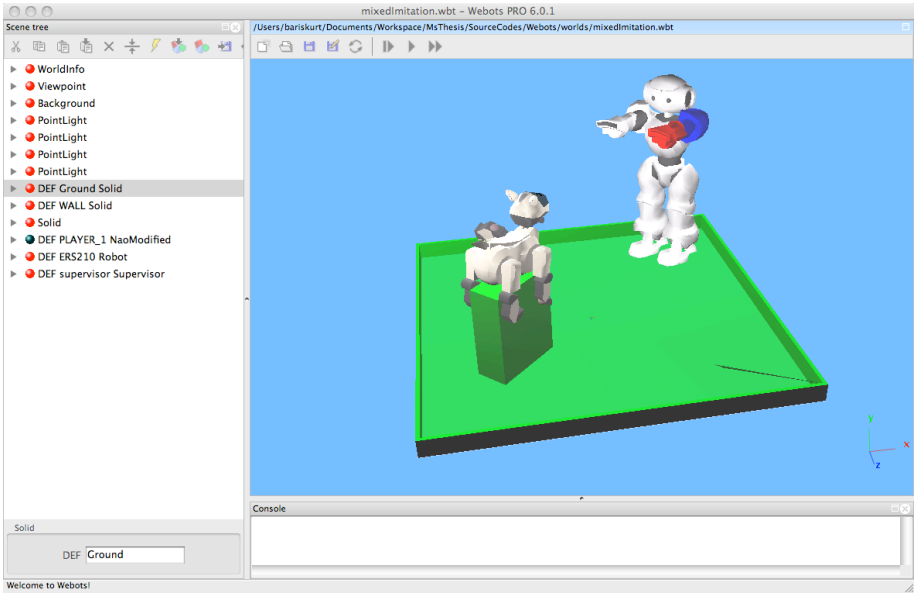
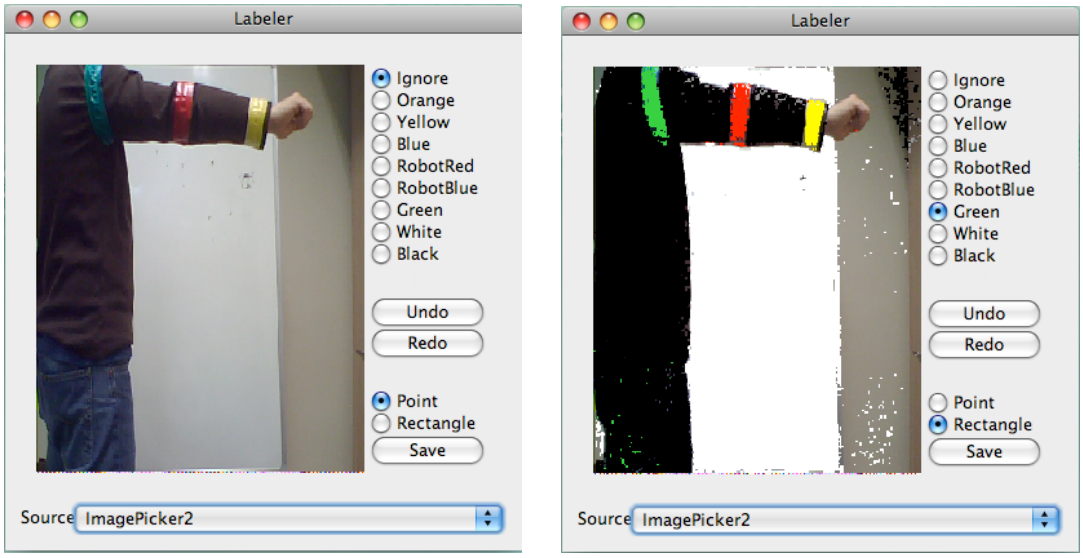


Figure 4.3. Webots Simulation Environment.

4.3. Vision System

We have used a monocular vision system for tracking the arm movements. For the ease of the image processing, colored marks are placed on the joints of the human and robot arms that are to be imitated. Therefore, the first step in processing the image is to segment the pixels in the image into discrete color groups (green, red, yellow, blue, white, black). We trained a *Generalized Regression Neural Network* [26] that takes the YUV values of a pixel and returns its probabilities of being a member of each color class. The pixel is classified as the color with the highest probability value. In order to train the network, a training set must be prepared. This is done by taking several images from the robot's camera and labeling the colors in the image (Figure 4.4). The network generalizes this data set and creates a color table. The color table stores all the possible YUV values and their corresponding color classes, hence no complex calculation is done online for the color classification. A lookup operation from the color table is performed for each pixel.

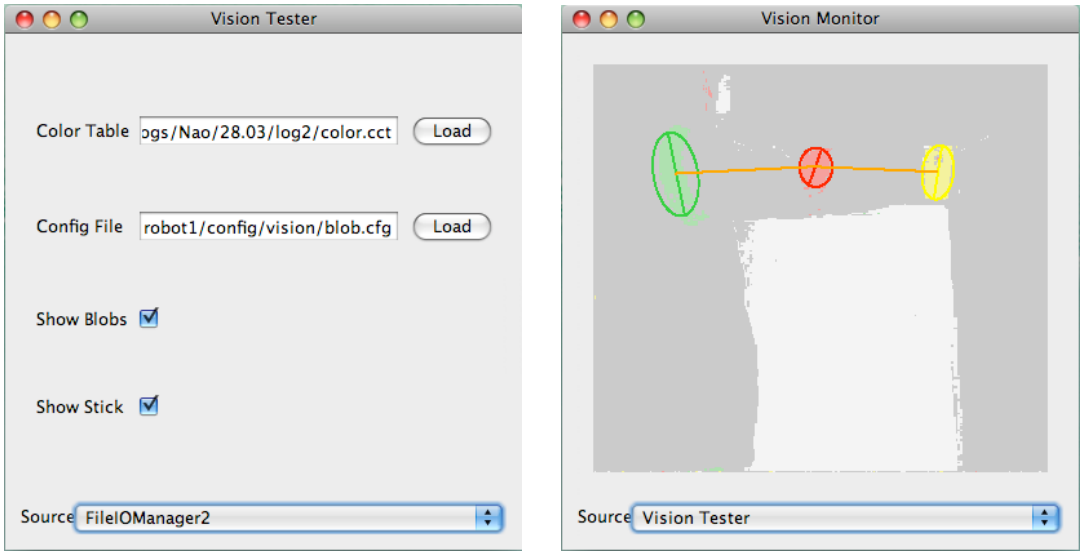


(a) Raw Image (b) Color Labeled Image

Figure 4.4. Raw and labeled images shown in the labeler plugin of the *Cerberus Station*.

After the training, the color table is tested with several images in the *CerberusStation*. Figure 4.5 shows the vision tester plugin and the color classified image with the

color blobs. A color table has to be trained for each robot and environment pair, since both the robots and environments have different characteristics.



(a) Vision Tester (b) Vision Monitor

Figure 4.5. An example of a classified image shown in the vision tester plugin of the *CerberusStation*.

Once the image is segmented into discrete colors, the next step is to find blobs in the image. A blob is a group of pixels of the same class that are connected. After the grouping of the pixels, the region is approximated as an ellipse which has the following properties as shown in Figure 4.6. Arm positions are extracted by using these properties of these blobs.

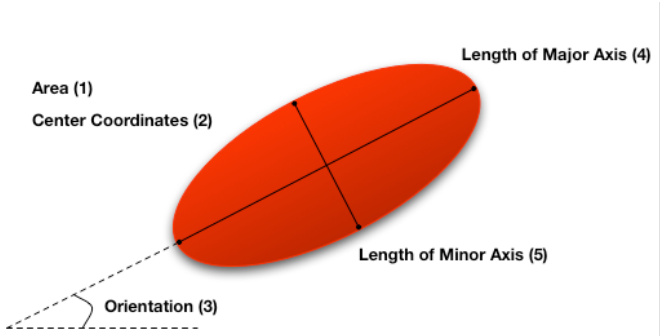


Figure 4.6. An example of a red blob and its properties.

4.4. Implementation of Mental State Inference Method

We developed a toy simulation environment to replicate the experiments of mental state inference (MSI) model (Figure 4.7) proposed by Oztop et. al. [20] . In the simulator, there are two robots, a demonstrator and an imitator. The demonstrator (bottom robot) moves its two-joint arm for reaching the blue points on the board. The other robot watches and tries to estimate the point which the arm is trying to reach before the arm actually reaches the point. The biological inspiration in this experiment is that both robots use the same modules for creating and perceiving the movement. As the demonstrator moves its arm, the imitator simulates the demonstrators movements internally by using its own motion modules, the authors state that this can be a model of mirror neuron system.

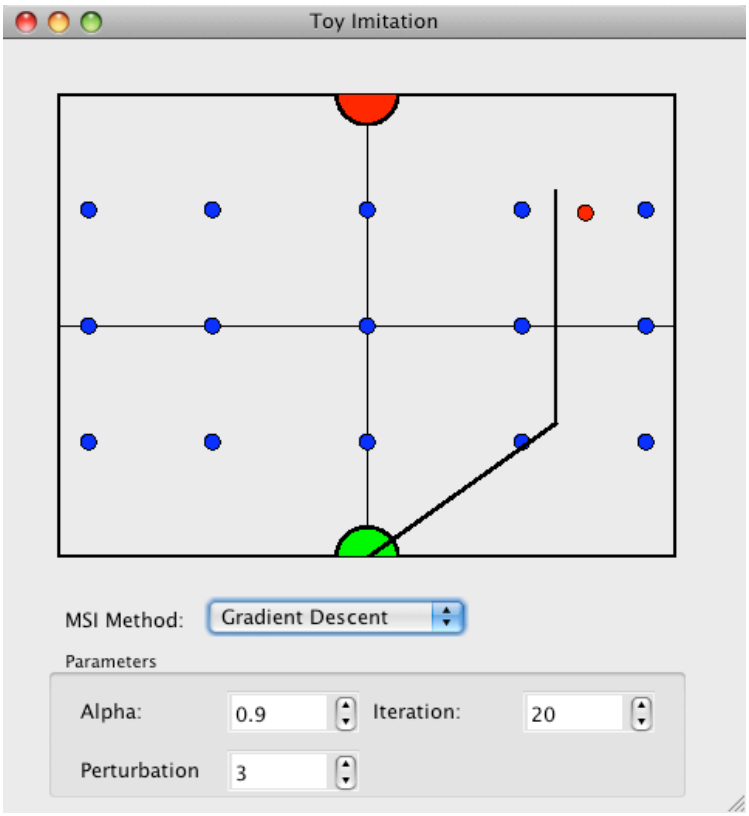


Figure 4.7. 2D Toy Simulator Environment. Green robot is the actor, red robot is the watcher.

In order to employ such a high level imitation model for imitating the human arm movements, we should solve the problems of low level imitation. The robot must be able to understand the position of the human arm and find a proper representation for it. We postponed this research as a possible future work, and continued our study with the low level imitation.

4.5. A Fuzzy Imitation Approach

Biological agents use qualitative terms to express an event. In the case movement understanding, we can say that the arm of a person can be down, up, half-down , half-up etc. But we do not say that the arm makes an angles of 73 degrees with the body for instance. Inspired from this phenomenon, we proposed to express the joint values qualitatively. Figure 4.8 shows an example fuzzification of the Aibo front arm joints by using five fuzzy sets with triangular membership functions.

In our fuzzy imitation approach, the imitator perceives the demonstrator's arm joints as fuzzy memberships. Afterwards, these fuzzy membership values are used to produce the joint values of the imitator itself. This method requires one-to-one mapping of the joints. We conducted our experiments on Webots simulation environment with three different robots: Aibo Ers-210, Aibo Ers-7 and Nao (Figures 4.9 and 4.10). The arm configurations of the Aibo robots were almost identical except their joint limits. However, the arm configuration of the Nao robot differs from the Aibo arm configurations. First of all, the Nao arm has four degrees of freedom, whereas the Aibo arm has three degrees of freedom. We fixed the elbow yaw joint of the Nao at zero degrees so that we were able to make a one-to-one mapping between Aibo and Nao arm joints. The limitations of the arm joints are shown in Tables 4.1, 4.2 and 4.3.

4.5.1. Learning Fuzzy Memberships

The imitator learned how to extract the fuzzy membership values of the joints of the demonstrator from the visual data and generated motor commands by defuzzifying the membership values. We trained neural networks (one for each membership func-

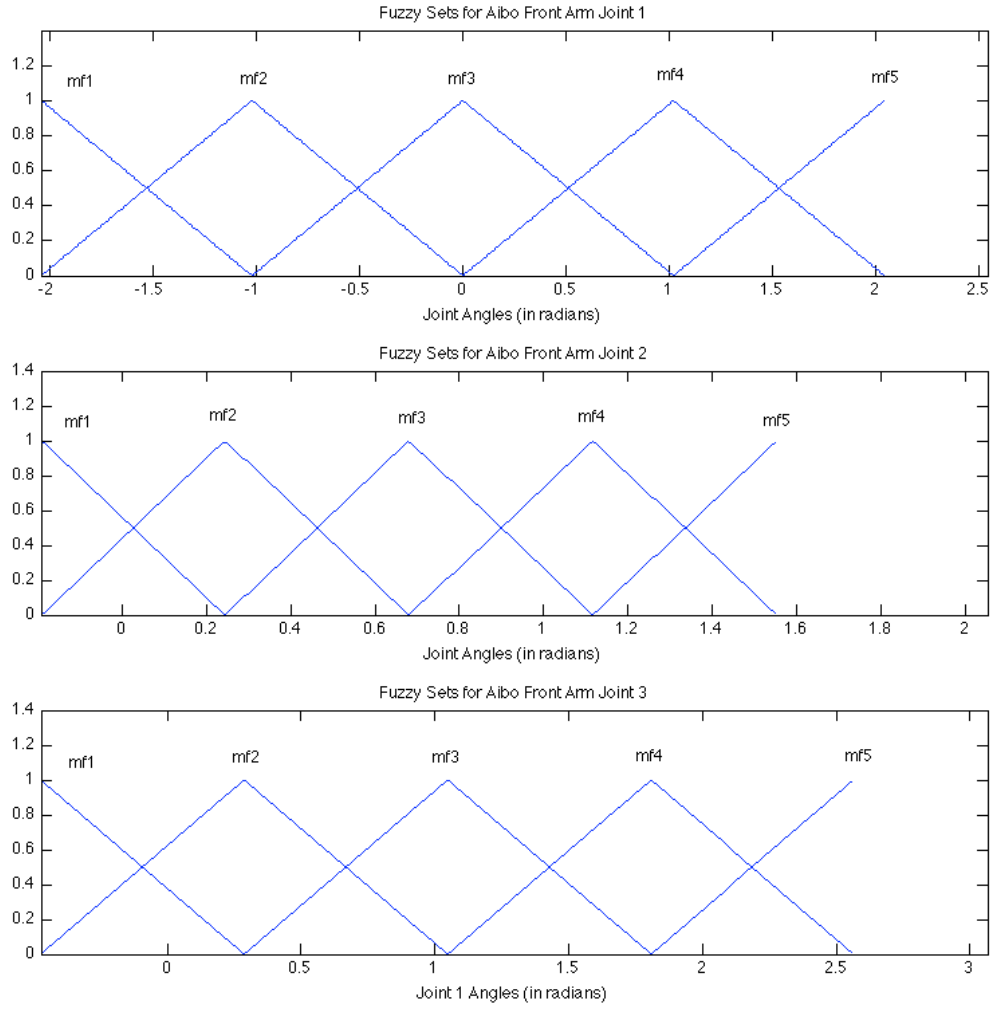


Figure 4.8. Fuzzy Sets for Aibo Front Arm Joints

Table 4.1. The joint limits of the left arm of the Aibo ERS-210 Robot.

ERS-210 Joints	Minimum	Maximum
J1	-117	117
J2	-11	89
J3	-27	147

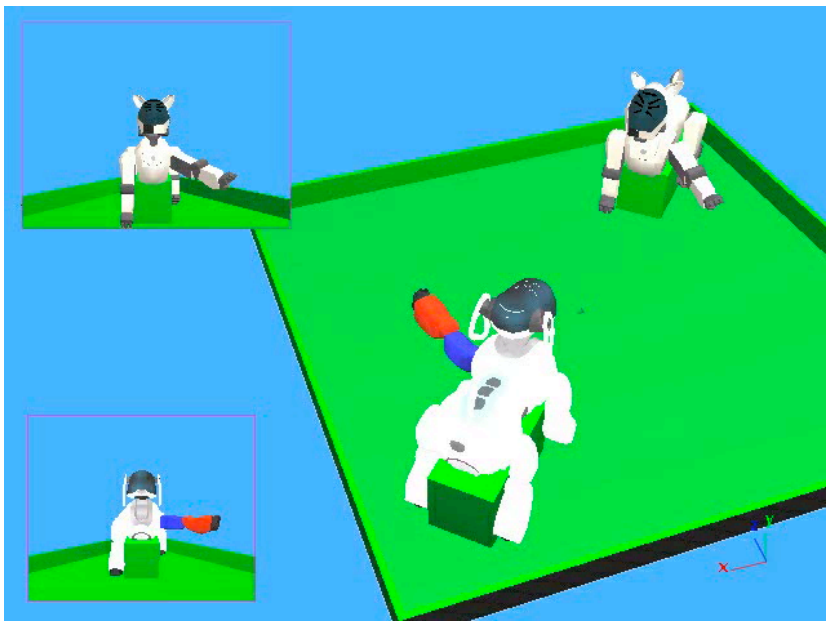


Figure 4.9. The Aibo robot imitating another Aibo robot on Webots Simulator

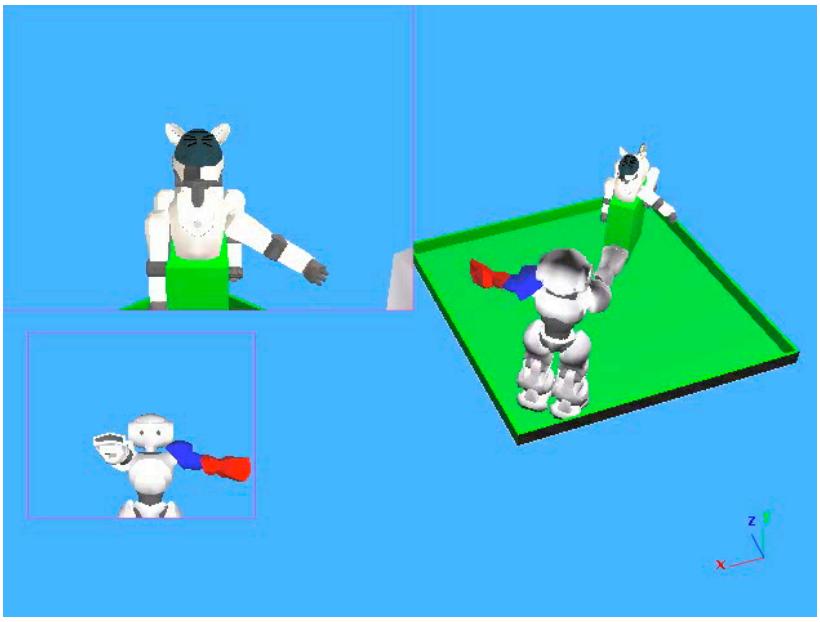


Figure 4.10. The Aibo robot imitating the Nao robot on Webots Simulator

Table 4.2. The joint limits of the left arm of the Aibo ERS-7 Robot's arm.

ERS-7 Joints	Minimum	Maximum
J1	-115	130
J2	-10	88
J3	-25	122

Table 4.3. The joint limits of the left arm of the Nao Robot.

Nao Joints	Minimum	Maximum
Shoulder Pitch	-120	120
Shoulder Roll	-95	0
Elbow Roll	0	90

tion) for learning the membership values. We created the data set by storing the joint values of the actor and perceptions of the imitator. The joint values of the actor are converted into fuzzy memberships. The perceptions of the imitator are the following blob properties: center, area and orientation. We collected data for 400 time steps.

We trained one network for one membership function, which means, if we have five memberships for each joint value, it requires 15 networks since we are trying to learn three joints. Figure 4.11 shows the final result of the system. The blue lines shows the actual arm movements while the red lines show the joint values of the imitator.

4.5.2. Intermediate Results

Although representing the joint angles as fuzzy memberships seems more natural, this method does not help solving the correspondence problem between the human arm and a robot arm. The method, in its current form, is only applicable whenever the arm configurations of the agents are similar. By similar, we mean that both agents have the same number of joints with similar movement ranges. However, the method may be applicable in a later step, after solving the correspondence problem. This issue will be saved as a future work for our thesis.

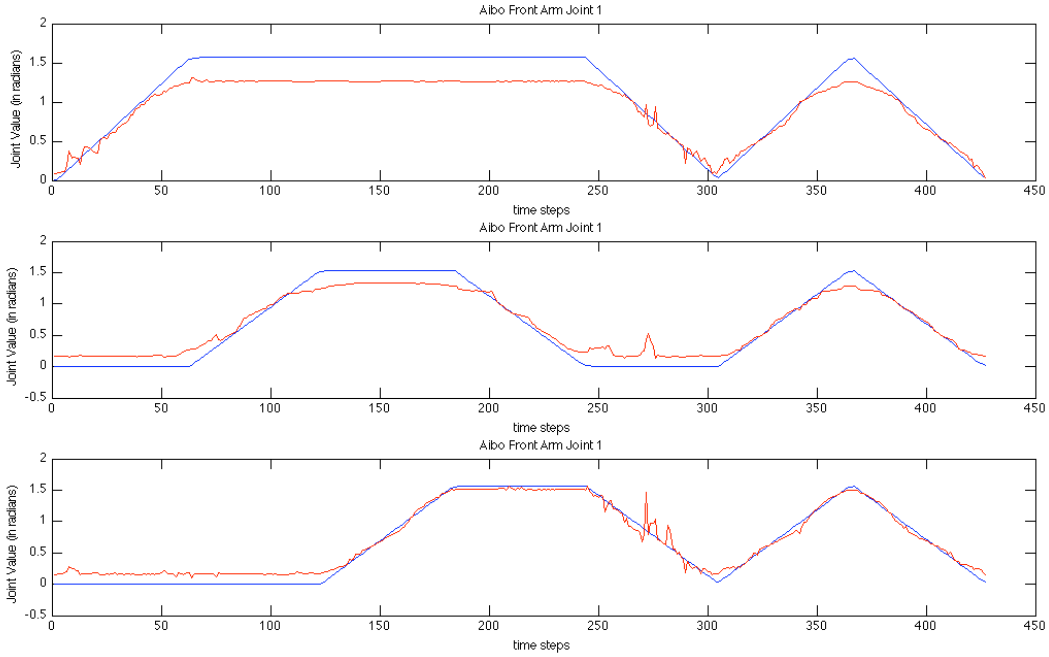


Figure 4.11. Results of The Training

4.6. Stick Figure Experiments

We tested our stick figure imitation model in a real world experiment. In the experiment, the demonstrator was wearing colored markers on his right arm joints while drawing lines and circles on the board, as shown in Figure 4.12. The robot watched and stored the movements executed by the demonstrator. Since we are not dealing with "when to imitate" problem, we cropped the necessary parts of the perception log offline. Afterwards we ran the networks to find the joint sequences on the log.

4.6.1. Perception of the Demonstrator's Stick Figure

The stick figure of the human (the demonstrator) is estimated by vision processing. First, the image of the human demonstrator is segmented into discrete colors. After the color segmentation 3 blobs are found: green, red and yellow. The center of the green blob shows the position of the shoulder. The centers of red and yellow show the elbow and wrist points respectively. The estimation process starts with the

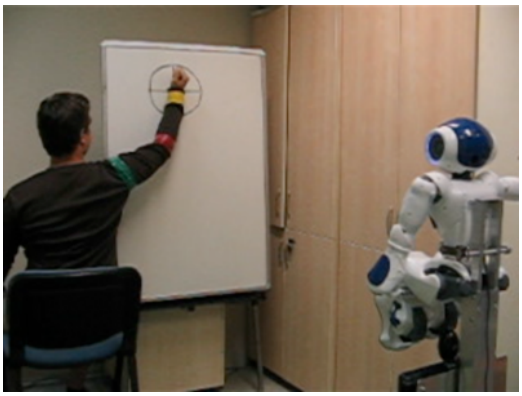


Figure 4.12. The experiment setup.

registration of the stick. In the registration, the demonstrator stands still in front of the robot, and stretches his arm. The view of the arm at this stage is shown in Figure 4.14a. The robot registers the lengths of the upper and lower arms, namely L_1^{max} and L_2^{max}, in terms of numbers of pixels. These values will be used for comparison in the later steps.

Algorithm *InitializeStickFigure*

1: $L_1^{max} = L_2^{max} = 0$
2: **while** $t < 10$ **do**
3: Compute L_1 and L_2
4: $L_1^{max} = L_1 + L_1^t/10$
5: $L_2^{max} = L_2 + L_2^t/10$
6: **end while**

Figure 4.13. Stick Figure Initialization

Let $\alpha = (x_\alpha, y_\alpha)$, $\beta = (x_\beta, y_\beta)$, $\gamma = (x_\gamma, y_\gamma)$ be the 2D cartesian coordinates of the centers of the green, red and yellow blobs on the image. The first step in estimating the stick figure is to calculate the distances between shoulder-elbow and elbow-hand on the image.

$$L_1 = \sqrt{(x_\alpha - x_\beta)^2 + (x_\alpha - y_\beta)^2} \quad (4.1)$$

$$L_2 = \sqrt{(x_\beta - x_\gamma)^2 + (y_\beta - y_\gamma)^2} \quad (4.2)$$

The orientations r_1 and r_2 of the stick figure are estimated as the slope of the lower and upper arms parts on the image (Figure 4.14b).

$$r_1 = \arctan\left((y_\alpha - y_\beta)/(x_\alpha - x_\beta)\right) \quad (4.3)$$

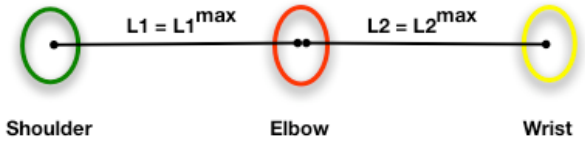
$$r_2 = \arctan\left((y_\beta - y_\gamma)/(x_\beta - x_\gamma)\right) \quad (4.4)$$

The orientations q_1 and q_2 of the stick figure are estimated by using the difference of the arm lengths with the lengths registered in the registration phase. The idea is that as the arm moves into the image (getting further from the observer), its projection onto the image will be smaller (Figure 4.14c). By assuming that this relationship is linear, the angles are estimated as:

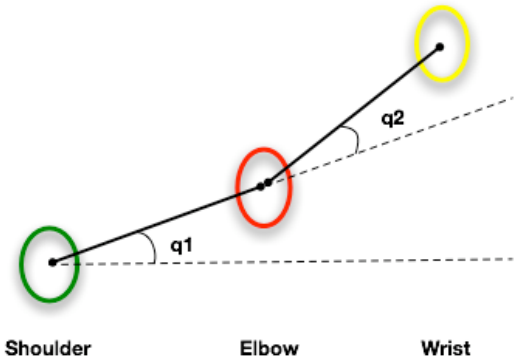
$$q_1 = \arccos\left(L_1^{max}/L_1\right) \quad (4.5)$$

$$q_2 = \arccos\left(L_2^{max}/L_2\right) \quad (4.6)$$

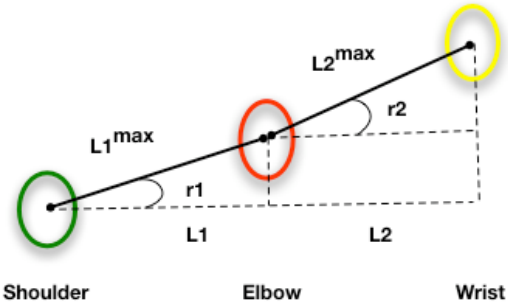
4.6.1.1. Depth Estimation Problem. The depth estimation technique is based on some simplifying assumptions. First of all, the projection of the arm onto the image plane cannot be done linearly, instead a camera matrix should be calculated to make a better projection. Secondly, for a given image, there are multiple candidate position for the



(a) Registration View



(b) Front View



(c) Top View

Figure 4.14. Estimating the stick figure from the image. (a) The robot sees the demonstrator in registration. (b) Front view for estimating angles r_1 and r_2. (c) Top view for estimating angles q_1 and q_2. .

arm as shown in Figure 4.15. The figure shows the top view of the arm. It is clear that the arm can be in 4 different positions for a given image. We assumed that the arm always moves further to the observer.

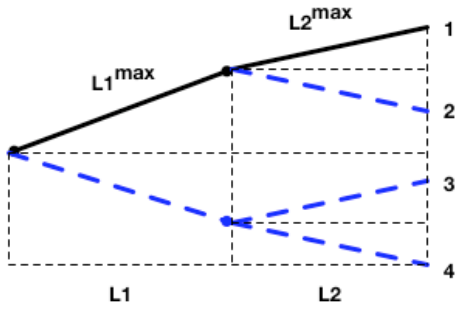


Figure 4.15. Stick Figure Depth Estimation Problem. The top view of the arm.

4.6.2. Replicating the Stick Figure

The next step after the extraction of the stick figure of the human arm is the replication of the stick figure by the imitator. The robot should find necessary joint angles in order to replicate the figure. We trained a hierarchy of feed-forward neural networks for converting the perceived stick figure to joint angle values. In our model, the first neural net extracts the proximal angles which are shoulder pitch and shoulder roll. Taking these values together with the stick figure as input, two neural networks determine the elbow yaw and elbow pitch angles.

4.6.2.1. Inverse Model Training. We have trained a hierarchy of feed forward neural network which generates the shoulder and elbow joints angles as shown in Figure 4.17. The first network learns the shoulder angles by taking the first two angles (q_1 and r_1) of the stick figure, and other two networks learn the elbow yaw and roll angles by taking the output of the first network and the second two angles (q_2 and r_2) of the stick figure. Before training the networks, we made a preprocessing for removing the singular cases in both data set generation and blob perception.

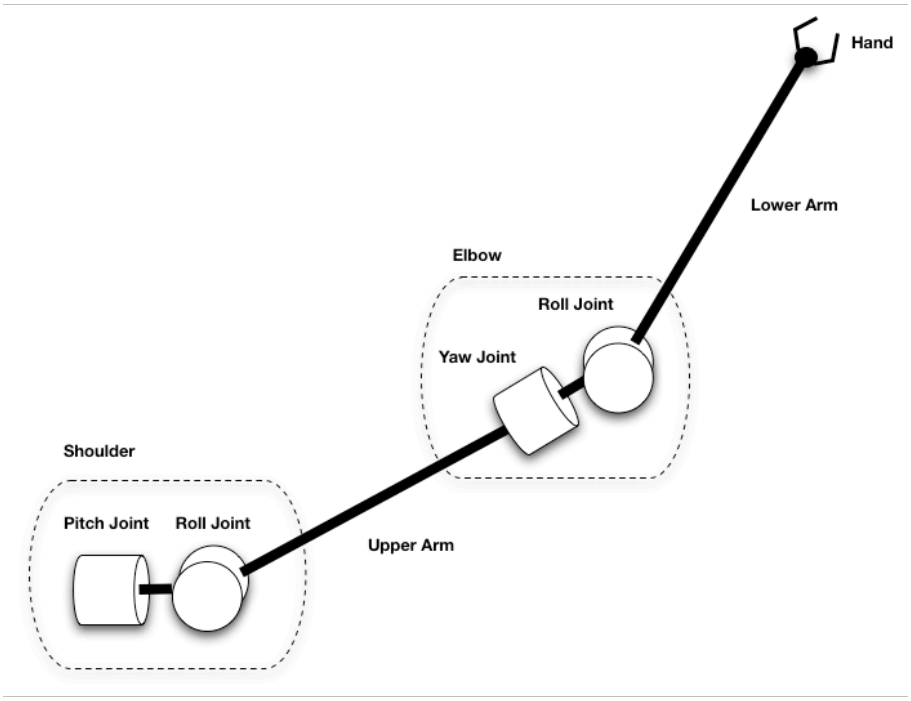


Figure 4.16. Nao Arm Joints

4.6.2.2. Eliminating Singular Configurations. In order to train a feed-forward network for solving the mapping problem, the mapping should be a one-to-one relationship. However, there exists singular configurations, such that, different arm configurations result in a single stick figure. These configurations can not be handled by a feed-forward network, instead a recurrent neural network which takes the target stick figure and the current arm configuration as input can be employed. In order to train such a network, we should create a larger data set, such that all possible movements from all arm configurations should be represented. This increases the search space exponentially.

Another solution for singular configurations is to make a preprocessing to eliminate them before using a neural network. As shown in Figure 4.18, a singular configuration occurs whenever the roll joint is in rest position (shown in dashed lines). In such cases the positions of the pitch or yaw joints do not affect the stick figure that results from the arm. Hence, we can say that whenever any of the shoulder roll or elbow yaw joints are in zero position, the arm is in a singular configuration.

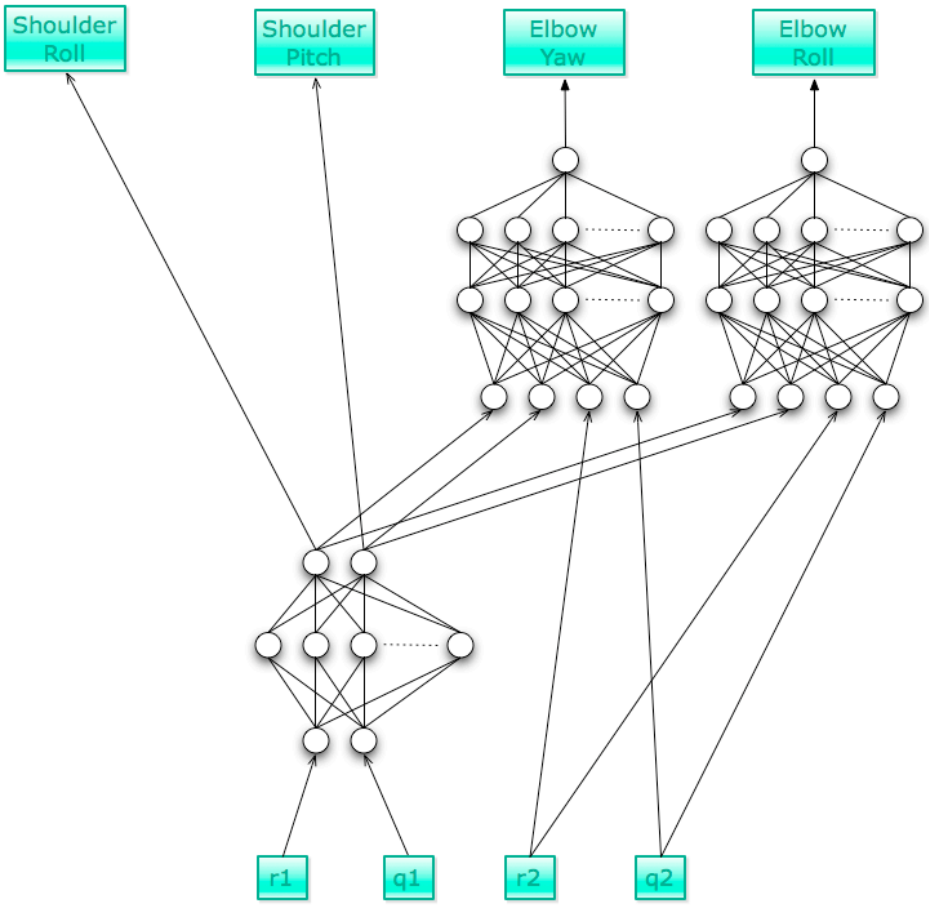


Figure 4.17. Artificial Neural Networks used as Inverse Model.

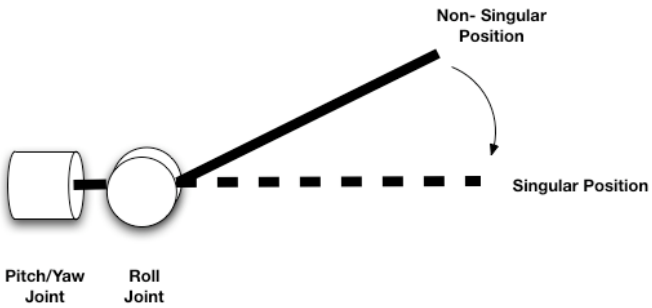


Figure 4.18. Singular Configurations of the Arm

In order to eliminate these configurations, first we should exclude the joint values that makes results in a singular configuration from the training data set. We have observed that if these values are not excluded, they result in unacceptable errors in shoulder pitch and elbow yaw angles. Hence, we have excluded the cases where shoulder pitch is zero and elbow yaw is -90 degrees.

Second, whenever a stick figure is observed, it should be checked that whether the figure is in a singular position or not. This situation occurs in two conditions. First, whenever $q_1 = 0$, meaning that the upper arm is parallel to the ground and second whenever $q_1 = q_2$ and $r_1 = r_2$ at the same time, meaning that the lower arm is parallel to the upper arm. In these situations, the necessary joint values are perturbed by one degree to their previous level in order to prevent a singular configuration.

4.6.3. Learning Shoulder Angles

In order to train the shoulder network, we have created a data set by selecting a subset of arm joints, and creating the stick figures for the joint combinations. For the shoulder pitch joint, we used angles starting from -120 to -60 by steps of 3 degrees. Similarly for the shoulder roll joint, we used angles starting from -88 to -2 by steps of 3 degrees. Hence, 21 angles for shoulder pitch and 29 angles for shoulder roll is used, which makes 609 combinations for upper arm position. For these combinations, we calculated the corresponding stick figures by using forward kinematics. We have trained a feed-forward neural network with one hidden layer with 40 neurons by using Levenberg-Marquardt algorithm. The first two angles q_1 and r_1 of the stick figure is given as input, and the shoulder joint values are given as target output. The learning curve is shown in Figure 4.19.

4.6.4. Learning Elbow Angles

We trained two networks for elbow joints: one for the elbow yaw joint and the other for the elbow roll joint. We used the same procedure as we used in training shoulder values. This time we used angles for all four joints combinations to create the

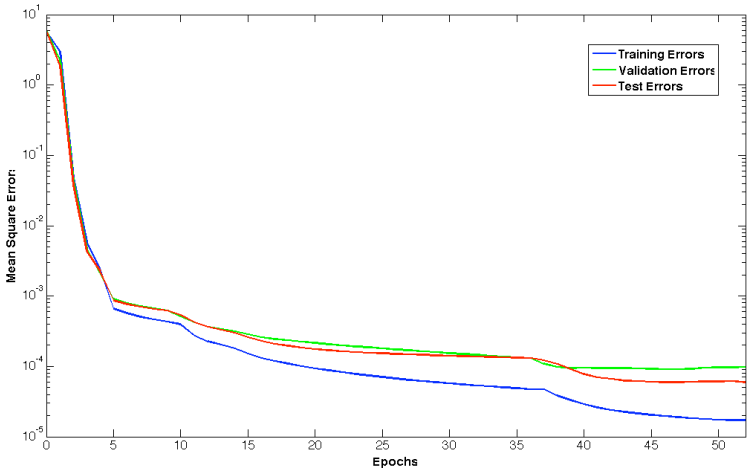


Figure 4.19. Training and Validation Errors for Shoulder Network

data set. For each joint combination, the stick figure is calculated by using forward model. Both networks take the upper stick figure angles q_2 and r_2 and shoulder joint angles as input. The values of elbow yaw and elbow roll joints are the target outputs respectively. The learning curves of the networks are shown in Figures 4.20 and 4.21.

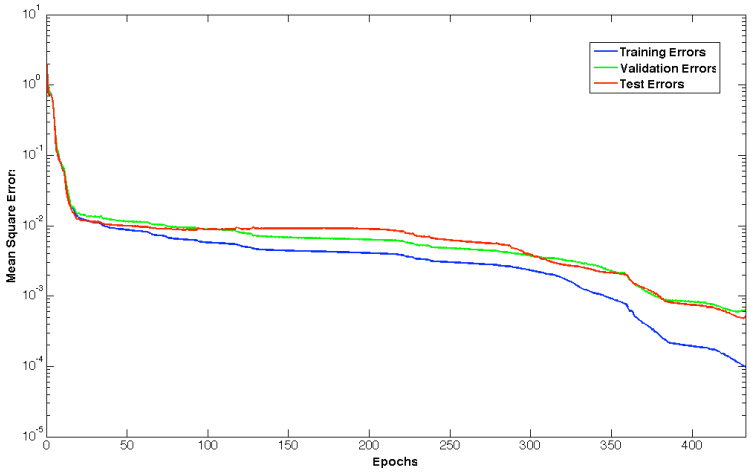


Figure 4.20. Training and Validation Errors for Elbow Yaw Network

After running the network to find the joint angles (Figure 4.22(a)), we smoothed it with the moving average method with a window size of five (Figure 4.22(b)). Finally,

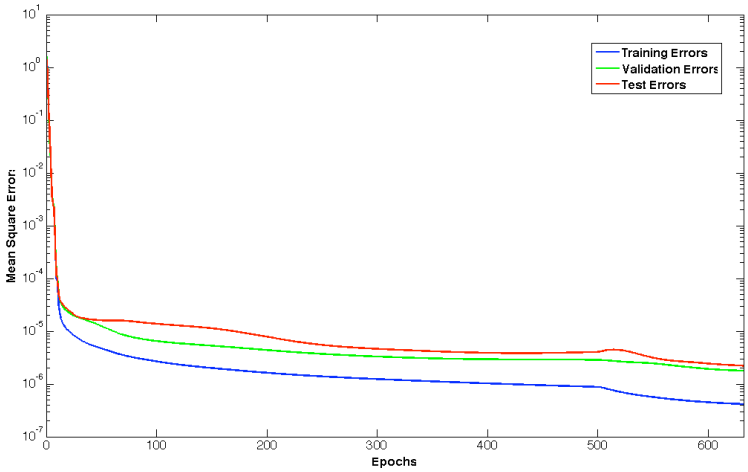
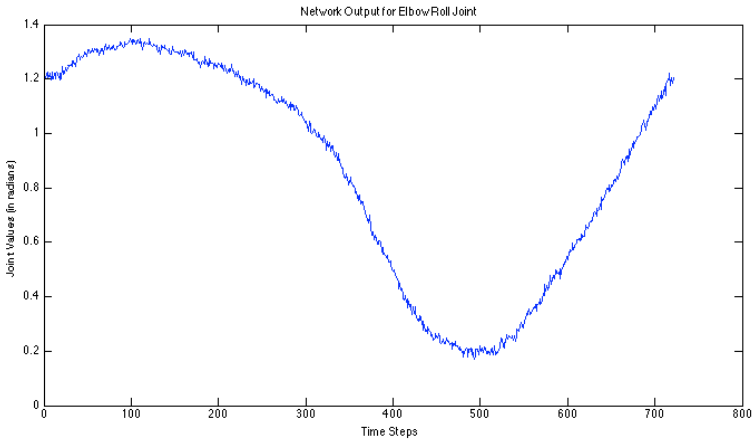


Figure 4.21. Training and Validation Errors for Elbow Roll Network

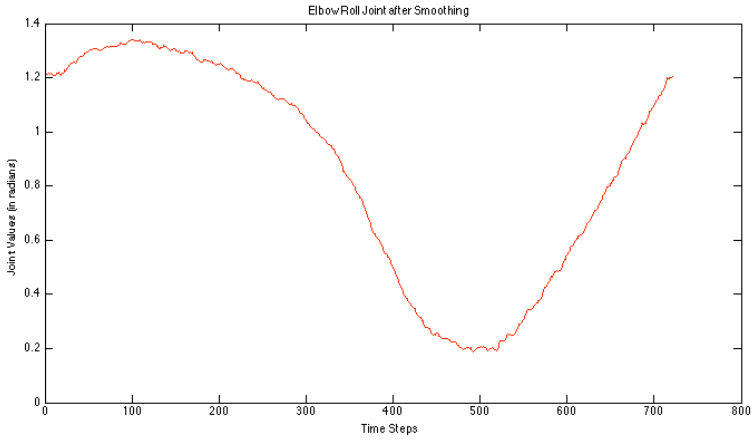
we chose target points with equal intervals from the data (Figure 4.22(c)). In each step, robot tried to reach the target points, and after reaching all of the targets, it finished its execution.

4.7. Results

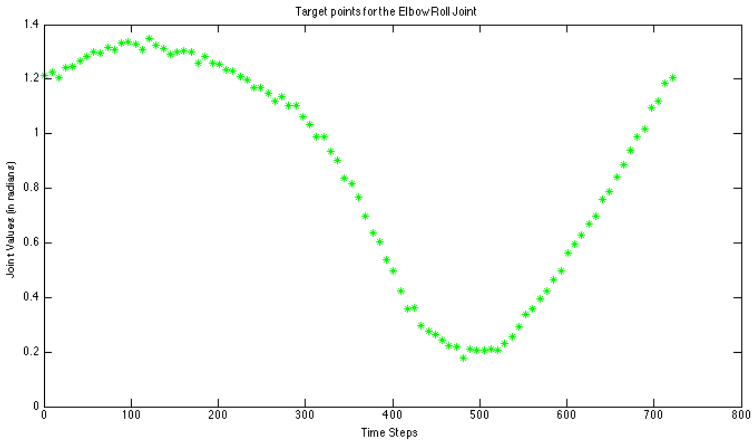
Figures 4.23, 4.24, 4.25 show how the robot draws the lines and circle. Although figures drawn by the robot are not perfect, they can be categorized as horizontal and vertical lines and circle by a human observer. The errors in the drawings come from the perception errors, simplifying assumptions in the vision system and inverse model errors. The biggest effect on the drawings are perception errors. Additionally, there is another significant difference between the human (as the demonstrator), and the robot (as the imitator). The robot stands still and does not move its shoulder's base point. However the human demonstrator unwillingly moves his shoulder base point as he draws lines. In the circle experiment, it should be noted that even if the figure is not a perfect shoulder, the robot has successfully finished the figure at the point where it started, hence it succeeded to make a closed loop.



(a) Network Output for Elbow Roll Joint



(b) Elbow Roll Joint data after smoothing



(c) Target points for Elbow Roll Joint

Figure 4.22. The post procesing steps for target angles

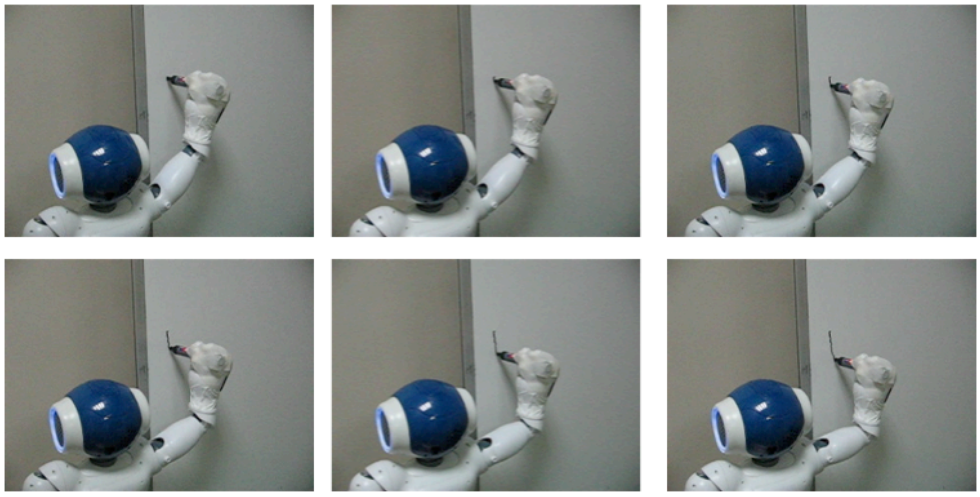


Figure 4.23. Robot drawing a vertical line.

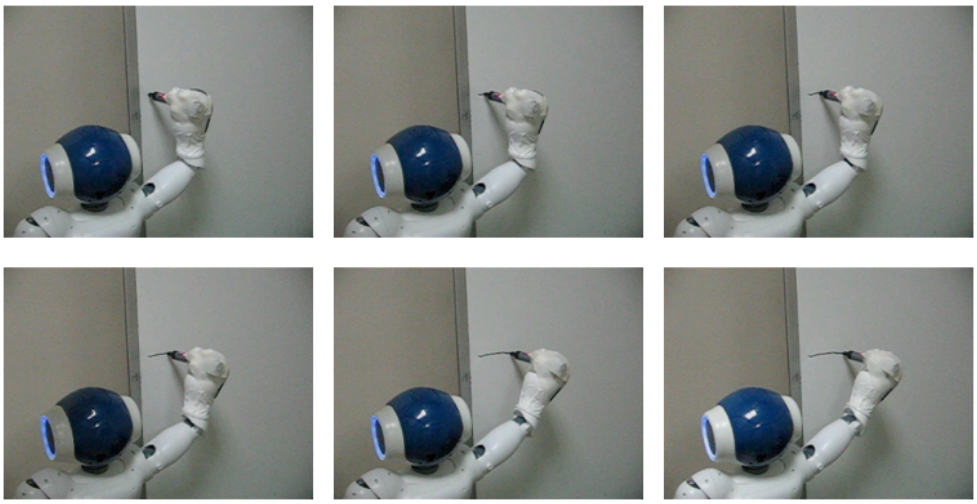


Figure 4.24. Robot drawing a horizontal line

Figure 4.25. Robot drawing a circle.

Figures 4.26, 4.27, 4.28 show different lines and circles drawn by the robot. In each of the drawings, the actions are generated from the same perception log. The difference in the figures are mostly the intensity of the ink.

Figure 4.26. Robot drawing circles.

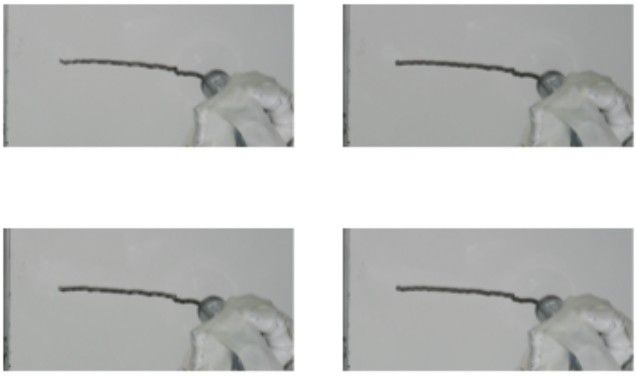


Figure 4.27. Robot drawing horizontal lines.

Figure 4.28. Robot drawing vertical lines.

5. CONCLUSIONS

In this thesis, we trained a humanoid robot to imitate the movements of a human demonstrator's arm. We tried to perceive the movement of the demonstrator with the single camera of the robot. We use a stick figure as an agent-independent representation of the arm movements, and train the robot to generate motor commands that will make its arm look as much the same as the demonstrator's.

We have seen that the differences in the bodies of the agents forces us to create a common representation system. The imitator robot can not store the movement information in terms of its own joint angles, because the joint types of the demonstrator and the robot itself are different. We have seen that it is hypothesized that the biological agents use a motor vocabulary by the help of their mirror neurons. Our stick figure can be considered as the motor vocabulary of the imitator.

We have also seen that using single camera for movement detection gives poor performance, since it is very difficult to extract depth information. We employed a registration based approach. After registering the image of the stretched arm, we looked at the differences with the registered image and the images taken at any time. We used the differences to estimate the position of the arm. Depth was not successfully estimated as the projection of the arm onto the image place is not linear as we assumed.

5.1. Future Work

The imitation process consists of two stages. Perception and motion generation. There can be many future work on improving these parts. Additionally, our imitation is limited with simple mimicry, but the higher levels of imitation such as goal directed imitation can be achieved.

First of all, vision processing can be done without color markers so that the robot does not need strict lighting conditions. Furthermore, more complex algorithms such

as Hidden Markov Model or particle filter based tracking can be employed for better depth estimation. A significant improvement on movement tracking can be achieved by using a stereo camera based tracking.

We proposed a fuzzy representation of joint values for understanding the actions of the imitator. This method can be used if the joints of the actor and the imitator can be mapped one-to-one. But it does not offer a solution for the correspondence problem. However a fuzzy approach may be employed during the solution of correspondence problem or in the stick figure perception phase.

APPENDIX A: Forward Kinematics

Table A.1. The Denavit-Hartenberg parameters for the right arm of Aldebaran NAO Humanoid Robot.

i	α_i	a_i	Θ_i	d_i
1	$-\pi/2$	0	j_1	0
2	$\pi/2$	0	$\pi/2 + j_2$	0
3	$\pi/2$	0	$\pi + j_3$	L_1
4	$\pi/2$	0	$\pi/2 + j_4$	0
5	$-\pi/2$	L_2	0	0

The transformation matrix used for calculating the position of the elbow according to the shoulder is as follows:

$$T^e = \begin{bmatrix} T^e_{11} & T^e_{12} & T^e_{13} & T^e_{14} \\ T^e_{21} & T^e_{22} & T^e_{23} & T^e_{24} \\ T^e_{31} & T^e_{32} & T^e_{33} & T^e_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T^e_{11} = \cos(j_1) * \sin(j_2) * \cos(j_3) - \sin(j_1) * \sin(j_3)$$

$$T^e_{12} = -\cos(j_1) * \sin(j_2) * \sin(j_3) - \sin(j_1) * \cos(j_3)$$

$$T^e_{13} = \cos(j_1) * \cos(j_2)$$

$$T^e_{14} = \cos(j_1) * \cos(j_2) * L1$$

$$T^e_{21} = -\cos(j_2) * \cos(j_3)$$

$$T_{22}^{e} = \cos(j_2) * \sin(j_3)$$

$$T_{23}^{e} = \sin(j_2)$$

$$T_{24}^{e} = \sin(j_2) * L1$$

$$T_{31}^{e} = -\sin(j_1) * \sin(j_2) * \cos(j_3) - \cos(j_1) * \sin(j_3)$$

$$T_{32}^{e} = \sin(j_1) * \sin(j_2) * \sin(j_3) - \cos(j_1) * \cos(j_3)$$

$$T_{33}^{e} = -\sin(j_1) * \cos(j_3)$$

$$T_{34}^{e} = -\sin(j_2) * \cos(j_2) * L1$$

The transformation matrix used for calculating the position of the wrist according to the shoulder is as follows:

$$T^{w} = \begin{bmatrix} T_{11}^{w} & T_{12}^{w} & T_{13}^{w} & T_{14}^{w} \\ T_{21}^{w} & T_{22}^{w} & T_{23}^{w} & T_{24}^{w} \\ T_{31}^{w} & T_{32}^{w} & T_{33}^{w} & T_{34}^{w} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{11}^{w} = -(\cos(j_1)*\sin(j_3)*\cos(j_3)-\sin(j_1)*\sin(j_3))*\sin(j_4)+\cos(j_2)*\cos(j_2)*\cos(j_4)$$

$$T_{12}^{w} = -\cos(j_1) * \sin(j_2) * \sin(j_3) - \sin(j_1) * \cos(j_3)$$

$$T_{13}^{w} = -(\cos(j_1)*\sin(j_2)*\cos(j_3)-\sin(j_1)*\sin(j_3))*\cos(j_4)-\cos(j_1)*\cos(j_2)*\sin(j_4)$$

$$T_{14}^{w} = (-(\cos(j_1) * \sin(j_2) * \cos(j_3) - \sin(j_1) * \sin(j_3)) * \cos(j_4) - \cos(j_1) * \cos(j_2) *$$

$\sin(j_4)) * L2 + \cos(j_1) * \cos(j_2) * L1$

$$T_{21}^{w} = \cos(j_2) * \cos(j_3) * \sin(j_4) + \sin(j_2) * \cos(j_4)$$

$$T_{22}^{w} = \cos(j_2) * \sin(j_3)$$

$$T_{23}^{w} = \cos(j_2) * \cos(j_3) * \cos(j_4) - \sin(j_2) * \sin(j_4)$$

$$T_{24}^{w} = (\cos(j_2) * \cos(j_3) * \cos(j_4) - \sin(j_2) * \sin(j_4)) * L2 + \sin(j_2) * L1$$

$T_{31}^{w} = -(-\sin(j_1) * \sin(j_2) * \cos(j_3) - \cos(j_1) * \sin(j_3)) * \sin(j_4) - \sin(j_1) * \cos(j_2) * \cos(j_4)$

$$T_{32}^{w} = \sin(j_1) * \sin(j_2) * \sin(j_3) - \cos(j_1) * \cos(j_3)$$

$T_{33}^{w} = -(-\sin(j_1) * \sin(j_2) * \cos(j_3) - \cos(j_1) * \sin(j_3)) * \cos(j_4) + \sin(j_1) * \cos(j_2) * \sin(j_4)$

$T_{34}^{w} = (-(-\sin(j_1) * \sin(j_2) * \cos(j_3) - \cos(j_1) * \sin(j_4)) * \cos(j_4) + \sin(j_1) * \cos(j_2) * \sin(j_4)) * L2 - \sin(j_1) * \cos(j_2) * L1$

REFERENCES

1. Herman, L. M., "Vocal, social and self-imitaion by bottlenosed dolphins", pp. 63–108, 2002.

2. Atkeson, C. G. and S. Schaal, "Learning tasks from a single demonstration", *IEEE International Conference on Robotics and Automation (ICRA97)*, 1997.

3. Atkeson, C. G. and S. Schaal, "Robot learning from demonstration", *International Conference for Machine Learning*, pp. 12–20, 1997.

4. Dautenhahn, K. and C. L. Nehaniv, "An agent-based perspective on imitation", pp. 1–40, 2002.

5. Nehaniv, C. L. and K. Dautenhahn, "The correspondence problem", pp. 41–61, 2002.

6. Alissandrakis, A., C. L. Nehaniv, and K. Dautenhahn, "Solving the correspondence problem between dissimilarly embodied robotic arms using the ALICE imitation mechanism", *In Proceedings of the second international symposium on imitation in animals and artifacts*, 2003.

7. Alissandrakis, A., C. L. Nehaniv, and K. Dautenhahn, "Action, State and Effect Metrics for Robot Imitation", *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pp. 232–237, 2006.

8. Alissandrakis, A., C. L. Nehaniv, and K. Dautenhahn, "Imitation with ALICE: learning to imitate corresponding actions across dissimilar embodiments", *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, Vol. 32, No. 4, pp. 482–496, 2002.

9. Rizzolatti, G., L. Fogassi, M. Gentilucci, G. Luppino, and M. Matelli, "Functional

organization of area 6 in the macaque monkey. II. Area F5 and the control of distal movements.", *Experimental Brain Research*, Vol. 71, pp. 491–507, 1988.

10. Rizzolatti, G. and C. Sinigaglia, *Mirrors in the brain*, Oxford University Press, 2006.

11. Gallese, V., L. Fadiga, L. Fogassi, and G. Rizzolatti, "Action Recognition in the premotor cortex", *Brain*, Vol. 119, pp. 593–609, 1996.

12. Murata, A., L. Fadiga, L. Fogassi, V. Gallese, V. Raos, and G. Rizzolatti, "Object representation in the ventral premotor cortex (area F5) of the monkey", *Journal of Neurophysiology*, Vol. 78, No. 4, pp. 2226–2230, 1997.

13. Kohler, E., C. Keysers, M. A. Umilta, L. Fogassi, V. Gallese, and G. Rizzolatti, "Hearing sounds, understanding actions: action representation in mirror neurons", *Science*, Vol. 297, No. 5582, pp. 846–848, 2002.

14. Buccino, G., F. Binkofski, and L. Riggio, "The mirror neuron system and action recognition", *Brain and Language*, Vol. 89, No. 2, pp. 370–376, 2004.

15. Grezes, J., J. L. Armony, J. Rowe, and R. E. Passingham, "Activations related to "mirror" and "canonical" neurones in the human brain: an fMRI study", *NeuroImage*, Vol. 18, No. 4, pp. 928–937, 2003.

16. Iacoboni, M., I. Molnar-Szakacs, V. Gallese, G. Buccino, J. C. Mazziotta, and G. Rizzolatti, "Grasping the intentions of others with one's own mirror neuron system.", *Plos Biology*, Vol. 3, No. 3, 2005.

17. Gallese, V., "Intentional attunement: A neurophysiological perspective on social cognition and its disruption in autism", *Brain Research*, Vol. 1079, No. 1, pp. 15–24, 2006.

18. Wicker, B., C. Keysers, J. Plailly, J.-P. Royet, V. Gallese, and G. Rizzolatti, "Both of us disgusted in my insula: the common neural basis of seeing and feeling disgust",

Neuron, Vol. 40, pp. 655–664, 2003.

19. Iacoboni, M. and M. Dapretto, "The Mirror Neuron System and The Consequences of its Dysfunction", *Nature Reviews Neuroscience*, Vol. 7, No. 12, pp. 942–951, 2006.

20. Oztop, E., D. Wolpert, and M. Kawato, "Mental state inference using visual control parameters", *Cognitive Brain Research*, 2005.

21. Hartenberg, R. S. and J. Denavit, *Kinematic Synthesis of Linkages*, McGraw-Hill, 1964.

22. 2009, `http://www.aldebaran-robotics.com/eng/index.php`.

23. 2009, `http://support.sony-europe.com/aibo/`.

24. "Robotics Group of Bogazici University", 2008, `http://robot.cmpe.boun.edu.tr`.

25. 2009, `http://www.cyberbotics.com/`.

26. Alpaydın, E., *Machine Learning*, MIT Press, 2004.

REFERENCES NOT CITED

1. Demiris, Y. and G. Simmons, "Perceiving the unusual: Temporal properties of hierarchical motor representations for action perception.", *Neural Networks*, Vol. 19, No. 3, pp. 272–284, 2006.

2. Dearden, A. and Y. Demiris, "Learning Forward Models for Robots", *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 1440–1445, 2005.

3. Tidemann, A. and P. Ozturk, "Self-organizing Multiple Models for Imitation: Teaching a Robot to dance the YMCA", *IEA / AIE*, Vol. 4570 of *Lecture Notes in Computer Science*, pp. 291–302, Springer, 2007.

4. Cole, J. B., D. B. Grimes, and R. P. N. Rao, "Learning Full-Body Motions from Monocular Vision: Dynamic Imitation in a Humanoid Robot", *International Conference on Intelligent Robots and Systems (IROS)*, pp. 240–246, 2007.

5. Ito, M. and J. Tani, "Generalization in Learning Multiple Temporal Patterns Using RNNPB", *International Conference on Neuroral Information Processing*, pp. 592–598, 2004.

6. Arbib, M. A., A. Billard, M. Iacoboni, and E. Oztop, "Synthetic brain imaging: grasping, mirror neurons and imitation", *Neural Networks*, Vol. 13, No. 8-9, pp. 975–997, 2000.

7. Johnson, M. and Y. Demiris, "Abstraction in recognition to solve the correspondence problem for robot imitation", *Towards Autonomous Robotic Systems, TAROS*, pp. 63–70, 2004.

8. Brass, M. and C. Heyes, "Imitation: is cognitive neuroscience solving the correspondence problem?", *Trends in Cognitive Sciences*, Vol. 9, No. 10, pp. 489–495,

2005.

9. Schaal, S., A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation", *Philosophical transaction: biological sciences*, , No. 1431, pp. 537–547, 2003.

10. Schaal, S., "Is imitation learning the route to humanoid robots", *Trends in Cognitive Sciences*, Vol. 3, No. 6, pp. 233–242, 1999.

11. Tidemann, A. and P. Ozturk, "Learning Dance Movements by Imitation: A Multiple Model Approach", *31st Annual German Conference on AI*, Vol. 5243, pp. 380–388, 2008.

12. Demiris, Y. and B. Khadhouri, "Hierarchical Attentive Multiple Models for Execution and Recognition of actions", *Robotics and Autonomous Systems*, Vol. 54, pp. 361–369, 2006.

13. Amit, R. and M. Mataric, "Learning Movement Sequences from Demonstration", *Proceedings of the 2nd International Conference on Development and Learning*, 2002.

14. Billard, A. and M. Mataric, "Learning human arm movements by imitation: Evaluation of biologicallyinspired connectionist architecture", *Robotics and Autonomous Systems*, 2001.

15. Wolpert, D. M. and M. M. Kawato, "Multiple paired forward and inverse models for motor control", *Neural Networks*, Vol. 11, No. 7-8, pp. 1317–1329, 1998.

16. Tani, J. and M. Ito, "Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 2003.

17. JunTani, "Learning to generate articulated behavior through the bottom-up and the top-down interaction processes", *Neural Networks*, 2003.

18. Tani, J., M. Ito, and Y. Sugita, "Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB", *Neural Networks*, Vol. 17, pp. 1273–1289, 2004.

19. Oztop, E. and M. A. Arbib, "Schema design and implementation of the grasp-related mirror neuron system", *Biological Cybernatics*, 2002.

20. Arbib, M. A., J. Bonaiuto, and E. Rosta, "The mirror system hypothesis: From a macaque-like mirror system to imitation", *Proceedings of the 6th International Conference on the Evolution of Language*, 2006.

21. Wolpert, D. M., K. Doya, and M. M. Kawato, "A unifying computational framework for motor control and social interaction", *Philosophical Transactions of The Royal Society B Biological Sciences*, 2003.

22. Haruno, M., D. M. Wolpert, and M. M. Kawato, "MOSAIC Model for Sensorimotor Learning and Control", *Neural Computation*, 2001.

23. Metta, G., G. Sandini, L. Natale, L. Craighero, and L. Fadiga, "Understanding mirror neurons: a bio-robotic approach.", *Interaction Studies*, Vol. 7, No. 2, pp. 197–232, 2006.

24. Oztop, E., M. Kawato, and M. Arbib, "Mirror neurons and imitation: a computationally guided review", *Neural Networks*, Vol. 19, No. 3, pp. 254–271, 2006.

25. Iacoboni, M., L. M. Koski, M. Brass, H. Bekkering, R. P. Woods, M.-C. Dubeau, J. C. Mazziotta, and G. Rizzolatti, "Reafferent Copies of Imitated Actions in the Right Superior Temporal Cortex", *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 98, pp. 13995–13999, 2001.

26. Iacoboni, M., "Neural Mechanisms of Imitation", *Current Opinion in Neurobiology*, Vol. 15, 2005.

27. Kozima, H., C. Nakagawa, and H. Yano, "Emergence of imitation mediated by

objects", *Proceedings of the 2nd International Workshop on Epigenetic Robotics*, 2002.

28. Metta, G., G. Sandini, L. Natale, and F. Panerai, "Development and Robotics", *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2001.

29. Grupen, R. and M. Huber, "A Framework for the Development of Robot Behavior", *2005 AAAI Spring Symposium Series: Developmental Robotics*, 2005.

30. Şimşek, Ö. and A. G. Barto, "An intrinsic reward mechanism for efficient exploration", *Proceedings of the 23rd international conference on Machine Learning*, 2006.

31. Barto, A. G., S. Singh, and N. Chentanez, "Intrinsically Motivated Learning of Hierarchical Collections of Skills", *Proceedings of International Conference on Developmental Learning*, MIT Press, 2004.

32. Andry, P., P. Gaussier, and J. Nadel, "From Visio-Motor Development of Low-level Imitation", *Proceedings of the 2nd International Workshop on Epigenetic Robotics*, 2002.

33. Dautenhahn, K. and A. Billard, "Studying Robot Social Cognition Within A Developmental Psychology Framework", *Proceedings of the 3rd International Workshop on Advanced Mobile Robots.*, 1999.

34. Oudeyer, P.-Y., F. Kaplan, and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development", *IEEE Transactions on Evalutionary Computation*, 2007.

35. Kaplan, F. and P.-Y. Oudeyer, "Motivational principles for visual know-how development", *Proceedings of the 3rd international workshop on Epigenetic Robotics*, pp. 73–80, 2003.

36. Huang, X. and J. Weng, "Novelty and reinforcement learning in the value system of developmental Novelty and reinforcement learning in the value system of developmental robots", *Proceedings of the 2nd International Workshop on Epigenetic Robotics*, 2002.

37. Lee, M. H., Q. Meng, and F. Chao, "Developmental learning for autonomous robots", *Robotics and Autonomous Systems*, Vol. 55, No. 9, pp. 750–759, 2007.

38. Chella, A., H. Dindo, and I. Infantino, "A cognitive framework for imitation learning", *Robotics and Autonomous Systems*, Vol. 54, No. 5, pp. 403–408, 2006.

39. Weng, J., "Developmental robotics: theory and experiments", *International Journal of Humanoid Robotics*, 2003.

40. Fong, T., I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots", *Robotics and Autonomous Systems*, pp. 143–166, 2003.

41. Lungarella, M., G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey", *Connection Science*, Vol. 15, No. 4, pp. 151–190, 2003.

42. Arbib, M. A., "The mirror system, imitation, and the evolution of language", pp. 229–280, 2002.