# ANIMATING HUMAN FINGER MOVEMENTS: A FEEDFORWARD NEURAL NETWORK APPLIED TO ARTIFICIAL TENDONS

**O.A. Kuchar and J.N. Scrimger**

Faculty of Computer Science, DalTech - Dalhousie University
P.O. Box 1000, Halifax, N.S., Canada, B3J 2X4
{okuchar | scrimger}@cs.dal.ca http://www.cs.dal.ca/~okuchar

**Abstract**

In our research, we are focusing on the preliminary developments of creating an artificially intelligent virtual human (AIVH) who is capable of moving and interacting in its environment with little intervention from the animator. Before an AIVH can move completely, an AIVH needs to learn how to move its body. Thus, the body needs to be divided into separate areas and the AIVH needs to learn about movements in these areas. In this paper, we report the use of a feedforward artificial neural network (ANN) to animate a computerized human finger. The underlying model of the computerized human finger is based on a biological model involving tendons. The ANN triggers any tendons that need to be manipulated to achieve an animated goal. The prescribed animated goals involve different flexion and extension movements. There are several drawbacks associated with using a multilayer feedforward network. Firstly, since the architecture is designed by trial-and-error, it is very difficult to create an effective a priori architecture. Secondly, ANNs are not trained on an entire input space, but there is a supposition that the ANN will work correctly when presented with any possible input within this problem space. In this ongoing project, the ability for an AIVH to move its finger using an ANN has been developed and tested, producing positive results.

**Keywords:** biological-based animation, artificial neural networks, backpropagation training algorithm, motion control, human figure movements

**List of Medical Abbreviations:**

**DIP (Distal Interphalangeal Joint):** The joint located between the distal and proximal phalanges in the fingers.

**PIP (Proximal Interphalangeal Joint):** The joint located between the middle and proximal phalanges in the fingers.

**MCP (Metacarpophalangeal Joint):** The joint between the proximal phalanx and the metacarpal bone in the fingers.

**FDP (Flexor Digitorum Profundus):** Primarily a muscle that produces flexion of the DIP joint in the fingers.

**FDS (Flexor Digitorum Sublimis):** Primarily a muscle that produces flexion of the PIP joint in the fingers.

**DOR (Dorsal Interosseous):** One of the functions of the dorsal interosseous is to help in flexing the MCP joint; the other function of the dorsal interosseous is abduction of the finger.

**EDC (Extensor Digitorum Communis):** The muscle that allows a finger to extend.

**INTR (intrinsic muscles):** A group of muscles that act together in extending the PIP and DIP joints.

## 1. INTRODUCTION

Films such as *Rendezvous à Montréal* [Thalm87a] were created from the desire to use human beings as synthetic actors in 3D computer animation. The difference between film and virtual reality is the dynamic behaviour required for the synthetic or virtual human. Cavazza et al. indicated some current research directions in virtual humans and their control [Cavaz98a]. Some of these directions include: generating and controlling virtual humans in real time; creating characters with individuality and personality who react to and interact with other real or virtual people; and enabling virtual humans to perform complex tasks. Controlling body motion involves animating a *skeleton* that consists of connected segments corresponding to limbs and joints. Animators control the skeleton locally and define it in terms of coordinates, angles, velocities, and accelerations. Motion control techniques from computer animation can be applied to virtual humans. Magnenat-Thalmann et al. [Magne96a] propose a classification of computer animation scenes involving synthetic actors based on the motion control method: geometric; physical; and behavioural. Geometric methods correspond to methods driven by geometric data, and some of these techniques are performance animation, keyframing, and inverse kinematics. Physical methods control the object's motion by describing the dynamic equations of motion relating the forces, torques, constraints, and the mass distribution of objects. The computer solves the dynamic equations and obtains the motion. A problem with this approach is the computational time in solving a large number of dynamic equations. Behavioural methods drive motion based on relationships between each object and other objects in an environment. A problem with this approach is defining the behavioural model completely for all types of interaction between objects. Reynolds [Reyno87a] uses a distributed behavioural model simulating interaction within flocks of birds, herds of land animals, and fish schools.

Several of the techniques in these methods are time consuming and not acceptable when applying them to artificially intelligent virtual humans (AIVHs). A method is needed to create motion in real-time such that a virtual human can react to its environment *dynamically*. In this paper we propose the application of artificial neural networks to motion control of an animated human finger. Discussion in this paper includes: an overview of artificial neural networks and the error backpropagation training algorithm; the underlying finger/hand model that is used in our research; experimentation and results; and conclusions and future work of this project.

## 2. AN OVERVIEW OF FEEDFORWARD ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) provide an alternative approach to many problems, such as: pattern recognition; optimization; and control. This section provides an overview of the common type of artificial neural network and describes a modified error backpropagation training algorithm that is used in our research.

### 2.1 Artificial Neural Networks

The main artificial neuron model consists of a processing element (node) with input connections and a single output. The signal flow of neuron inputs, $z_i$, is considered to be unidirectional, as is a neuron's output signal flow, $\mathbf{o}$. A general neuron symbol is shown in Figure 1. The neuron output signal is given by the following relationship:

$$\mathbf{o} = f(\mathbf{wz}) = f(\Sigma w_i z_i)$$

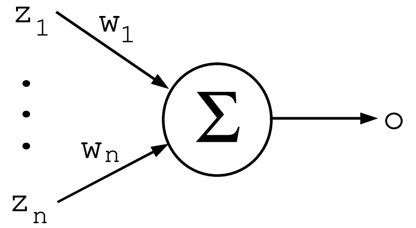
where $\mathbf{w}$ is a weight vector defined as:

$$\mathbf{w} = [w_1 \; w_2 \; ... \; w_n]$$

and $\mathbf{z}$ is the input vector defined as:

$$\mathbf{z} = [z_1 \; z_2 \; ... \; z_n]^T$$

The function $f(\mathbf{wz})$ is referred to as an activation function. The mathematical models associated with the activation function vary for different ANN applications.




A general neuron symbol.

Figure 1.

An ANN is a set of interconnected neurons, such that neuron outputs are connected by weights to other neurons and themselves. There exist many different types of ANNs and can be grouped into two categories based on their connection pattern: feedforward neural networks; and recurrent or feedback neural networks. A feedforward network is a graph with no loops, and the most common of these is the multilayer perceptron network. A recurrent neural network is a graph with loops, and some examples of these are Competitive and Hopfield networks. We use a multilayer feedforward network in our research, and this network is described in detail in section 4.

### 2.2 Error Backpropagation Training Algorithm

One of the major advantages of ANNs over expert systems is the ability to learn. The operation of an ANN involves two processes: learning; and recall. Learning is the process of adapting the connection weights in response to the input values. The ANN "learns" based on a learning rule that governs how the connection weights (**w**) are modified because of the input values (**z**). Recall is the process of taking the input and producing a response based on what the ANN learned.

The learning algorithm applied in this research is a general delta-rule learning algorithm, also known as the error backpropagation training algorithm [Rumel86a]. Training involves multiple cycles through the training data during which the weights are modified to reduce the error between the ANN output and the desired output. The difference between the desired output and actual output is calculated using a least-squares process. The standard error backpropagation training algorithm can be summarized as follows:

1. One pattern is presented from the training data to the ANN and the ANN computes the output.

2. The output from the ANN is compared to the desired output.

3. An error signal is calculated using a negative gradient descent technique.

4. The error signal is propagated to both the input and hidden layers, and connection weights are modified.

In this research, the authors modified the standard error backpropagation algorithm, by dividing the last part into two steps:

4. The error signal is propagated to the input layer and connection weights are modified.

5. Parts 1-3 are repeated for the same pattern, and the error signal is propagated to the hidden weight layer and connection weights are modified.

As noted in [Samad88a] and tested on a set of ANN architectures in this research, this method speeds convergence greatly.

Once all cases have been presented to the ANN, a cycle has been completed and the summed error generated by this cycle is compared to the user's desired maximum error. If the summed error is greater than the desired error, the ANN continues to train on the training data.

## 3. THE FINGER MODEL

Several muscles for the fingers and thumb were modelled according to their original biological counterparts, and a full description of these tendons and their functions is described in [Kucha96a]. Since we are training an ANN to move one finger, only a subset of tendons was used in the ANN. These tendons are depicted in Table 1. The finger model and the mathematical model applied to the muscle effect on the fingers are described in [Kucha97b].

| Abbreviation | Name and Function |
|---|---|
| FDP | Flexor Digitorum Profundus - flexes the fingers |
| FDS | Flexor Digitorum Superficialis - flexes the fingers |
| DOR | Dorsal Interroseus - flexes the fingers |
| EDC | Extensor Digitorum - extends the fingers |
| INTR | Intrinsic Muscles - extends the fingers |

Summary of modelled finger tendons and their functions.

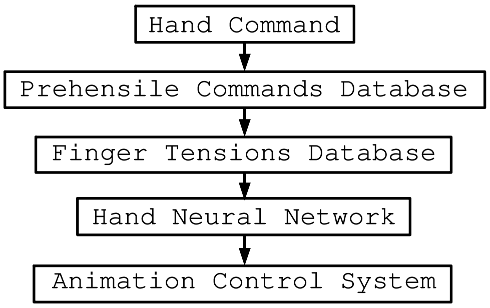
Table 1.

## 4. FEEDFORWARD NEURAL NETWORK APPLIED TO FINGER MOVEMENTS

In this section we describe the creation, training, and results of our ANN applied to animating finger movements.

### 4.1 Overview

An animator can manipulate the movement of the fingers and thumb by activating certain tendons. A problem with this approach is the reusability of code. For example, if the animator animates a hand to grasp a box, the animator has to redo all steps to animate a hand grasping a baseball bat. To decrease the time in creating an animation, it would be helpful for an animation system to have an ANN attached to the hand. In this respect, once an ANN has learned from the animator how to grasp certain object shapes, it could then apply that knowledge to grasp any similar objects such as cans, phones, and bottles. Thus, in a virtual environment, a virtual human can modify its actions dynamically.




The hierarchical finger movement control scheme.

Figure 2.

Before an animated hand can grasp any object, the animated hand has to move and synchronize its fingers. The animated hand has to learn how to move each finger and how the fingers should work together to accomplish a task. A hierarchical control scheme, depicted in Figure 2, would be a first approach in animating human hand movements. The first stage would require the user to input a command, such as *FIST*. This command is used to search a database of prehensile commands, depicted at the second stage. This database contains finger commands to attain specified actions for the hand. For example, a command *FIST* would require *FULL_FLEX_INDEX*, *FULL_FLEX_MIDDLE*, *FULL_FLEX_RING*, *FULL_FLEX_PINKY*, and *FULL_FLEX_THUMB*. An excerpt from the Prehensile Commands Database is depicted in Figure 3a. The third stage includes a second database for finger tensions related to the specified finger action. For example, a command *FULL_FLEX_INDEX* would require the following tensions: FDP tension at 4.9%; FDS tension at 0%; DOR tension at 0%; EDC tension at 4.9%; and INTR tension at 4.9%. An excerpt from the Finger Tension Database is depicted in Figure 3b. The fourth stage is the ANN whose input contains both the current fingers' tensions and the desired fingers' tensions from the third stage. The output from the ANN is the amount of tension change that is required for each finger to attain the desired tension. This output is then directed to the final stage. The final stage is the animation control system that manipulates the tendons to the desired state.

| HAND COMMAND | FINGER COMMANDS |
|---|---|
| FIST | FULL_FLEX_INDEX, FULL_FLEX_MIDDLE, FULL_FLEX_RING, FULL_FLEX_PINKY, FULL_FLEX_THUMB |
| POINT | EXT_INDEX, FULL_FLEX_MIDDLE, FULL_FLEX_RING, FULL_FLEX_PINKY, FULL_FLEX_THUMB |
| PEACE | FULL_FLEX_RING, FULL_FLEX_PINKY, FULL_FLEX_THUMB, EXT_INDEX, EXT_MIDDLE, ABD_INDEX, ABD_MIDDLE |

a.) Excerpt from Prehensile Commands Database

| FINGER COMMAND | FINGER TENSIONS |
|---|---|
| FULL_FLEX_INDEX | 0.0490 0.0 0.0 0.0490 0.0490 |
| EXT_INDEX | 1.0 1.0 1.0 1.0 1.0 |

b.) Excerpt from Finger Tension Database.

Figure 3.

### 4.2 Finger Flexion and Extension

The previous section introduced the concept of using ANNs to manipulate hand movements. Before an animated hand can be manipulated by an ANN, the ANN must learn how to move individual animated fingers. As a first step in developing the animated hand, an ANN must be developed for animating finger movements. This section describes in detail the artificial finger neural network (AFNN) including current results.

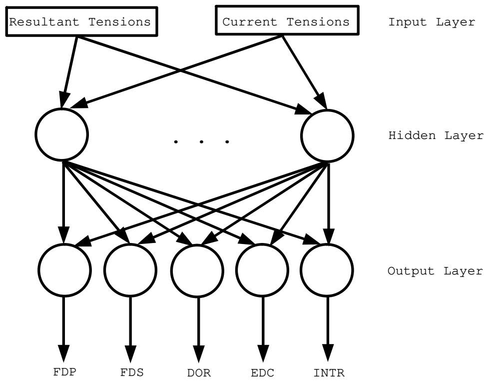
#### 4.2.1 AFNN

To create the specified movement or animated goal, the output of the AFNN calculates: which tendon(s) must be activated; and the amount of tension change needed by the flexors and extensors in the finger model. Currently, the AFNN only flexes and extends an animated human index finger.

The AFNN, depicted in Figure 4, is a multilayer feedforward, fully connected network consisting of a set of neurons that are arranged into three layers. The three layers consist of: ten neurons in the input layer corresponding to two sets of input data; nine neurons in the hidden layer; and five neurons in the output layer. The first set of the input data corresponds to the resultant tensions of the FDP, FDS, DOR, EDC, and INTR once the movement has been completed (Figure 3b). The second set of the input data corresponds to the current tensions of the FDP, FDS, DOR, EDC, and INTR. The hidden layer, consisting of nine neurons that do not directly correspond to any tendons, is required for the AFNN to make the necessary correlations between the input and desired output of the AFNN. More hidden nodes may be required if the AFNN fails to converge to a solution. Fewer hidden nodes may suffice if the AFNN does converge. The final size of the hidden layer will be based on the overall system performance. Several experiments were conducted to evaluate the size of the hidden layer, and it was discovered that a hidden layer of nine nodes performed efficiently compared to other AFNN architectures.. The output layer consists of five neurons and corresponds to the tendon activation layer. The activation function used was the unipolar continuous function:

$$F(net) = 1 / (1 + e^{-net})$$

where net = $\Sigma w_i z_i$. A sigmoidal activation function was chosen to ensure a desirable normalization.




Artificial Finger Neural Network.

Figure 4.

#### 4.2.2 Flexion/Extension Cases

Each tendon in the finger has specific effects on the finger when triggered [Kucha96a]. To train the AFNN, it needs to learn the cause/affect relationship of each tendon. Using anatomic location analysis and electromyographic studies detailed in [Lands65a], we incorporated the phases of exercises analyzed by [Long64a] into the first 14 cases shown in Table 2. The remainder of the cases were created after an in-depth study of finger movements from [Tubia81a], [Kapan70a], and [Seire89a]. The table incorporates only five different starting and finishing positions, specifically:

- CLOSE ( ): fully flexed finger;
- OPEN ( ): fully extended finger;
- FF-DIP-EXT ( ): **F**ully **F**lexed finger with the DIP joint remaining **EXT**ended;
- HOOK ( ): fully flexed finger with the MCP joint fully extended;
- V-SHAPE ( ): fully flexed finger with the DIP and MCP joints fully extended;
- POINT-DOWN ( ): fully extended finger with the MCP joint fully flexed.

There should be 30 cases in total, but five cases have been eliminated:

- From FF-DIP-EXT to CLOSE: when the MCP and PIP joints are flexed to 90°, the FDP cannot flex the DIP joint because it has become too slack for any useful contraction [Kapan70a];
- From CLOSE to FF-DIP-EXT;
- From CLOSE to V-SHAPE;
- From HOOK to FF-DIP-EXT;
- From HOOK to V-SHAPE.

The reasoning for the last four eliminated cases is that no tendon exists in the finger that can extend the DIP joint before the PIP joint. The fourth column in Table 2 indicates which tendons are required to be triggered in the starting position to create the desired finishing position.

#### 4.2.3 Input-Output Data

The input-output sets for the AFNN only include cases 1-7, 9, 12-13, and 15-22 because these are the cases where only one specific tendon is required for a specified movement. The other cases were not included in the training or test set since these cases can be derived from a combination of other cases. The gathering of empirical data for cases where more than one tendon is required for a specified movement was problematic. Since one tendon affects the tensions of the other tendons, it was difficult to measure the amount of tension change in one tendon within the movement.

The AFNN's inputs and outputs have been normalized (range from 0.0 to 1.0) to allow for the manipulation of different finger lengths. The input-output relationship was obtained from previous work described in [Kucha96a]. The author created a computer interface that allows the user to manually manipulate the tendon tensions of an animated human hand. The user can gather the information needed for the AFNN from the script provided in the interface. This information can be correlated to the coding system used for specifying finger positions [Jacob76a]. Currently, the data set contains 826 patterns of input data.

### 4.3 Training the AFNN

All data may be used to train the AFNN; however, a small subset is often all that is needed to train a network successfully. The remaining data can be used to test the verification of the AFNN on input data that it has not processed during training. The training data should cover the entire expected input space. The training data used for the AFNN contains 317 patterns (in random order) for flexion and extension of an animated finger. Weights were initialized to random values between -0.5 and +0.5.

### 4.4 AFNN Results

There are three variables in training the AFNN: the number of nodes in the hidden layer (H); the learning rate parameter (N); and the maximum error desired by the trainer ($E_{MAX}$). First, the number of nodes in the hidden layer is a variable in constructing the AFNN. As noted previously (Section 4.2.1), a hidden layer of nine nodes performed efficiently.

Secondly, selection of a value for the learning rate parameter affects the network performance. For the AFNN, N should be a number between 0.01 and 0.2 to ensure that the network will converge to a solution. A smaller value of N means that the network will have to make a larger number of iterations, and any larger N value may prevent the AFNN from converging. Several experiments were conducted to evaluate N. The results were that smaller values of N converged the network, noting that N=0.1 seemed efficient.

Thirdly, the maximum error desired by the trainer was varied to allow observation of the number of training cycles required for the AFNN to converge to a solution. An error of 0.2% in the extension or flexion of a finger cannot be visually detected during animation, and thus $E_{MAX}$ was set to 0.002.

There are several drawbacks associated with using an ANN. First, given the specifications of a problem, it is very difficult to specify an effective architecture. Secondly, after an ANN has been trained, intuitive understanding of how it works may be difficult. Thirdly, there is a supposition that the ANN will work correctly when presented with any possible input within a problem space. Techniques for strict mathematical verification of an ANN's performance are still being developed. An ANN that has been well trained and verified with a reasonable test set will perform well in practice. Performance quality is simply difficult to prove at this time.

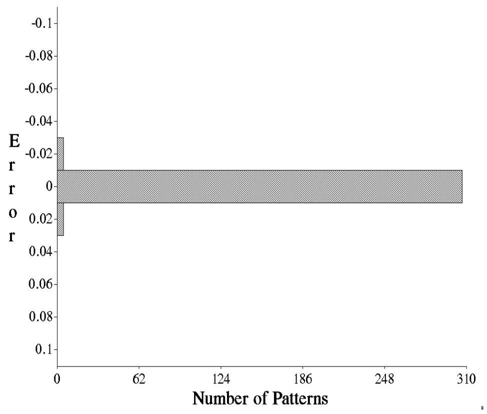
### 4.5 Test Data Performance

After testing the AFNN, results were compared between the output of the AFNN and the desired output. The error between the desired and actual output was computed for all patterns in the test data, and the errors were plotted in a histogram depicted in Figure 5. Figure 5a represents the overall performance error on the training data. The AFNN computed the proper tendon tension change for 307 patterns (total of 317 patterns) within an error range of ±1%. All patterns were inside the ±3% error range. Figure 5b represents the overall performance error on the test data. The AFNN computed the proper tendon tension change for 527 patterns (total of 826 patterns) within an error range of ±1%. Only 4 patterns were outside the ±3% error range.

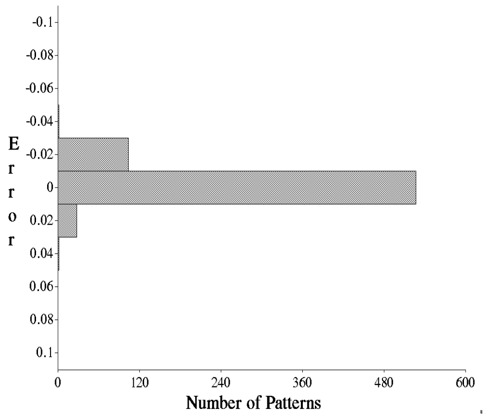
### 4.6 Discussion and Future Direction

From Figure 5, we were able to attain information on how well the AFNN's performance was on both training and test data. The test set allowed us to notice certain input patterns with which the network had difficulties and to address those problems individually. We wanted to be able to recognize what the AFNN still could not "understand". Usually, this required adding the problematic finger movements into the training set.

A test set should cover the entire input space. Since even our test set contained a subset of possible starting positions for the AFNN, we incorporated the AFNN into an animation program to view the end-position results. This allowed us to test more situations and visually detect problems with the AFNN. When problems were detected, the input patterns were incorporated into both training and test data.




a.) Overall performance error on the training set.




b.) Overall performance error on the test set.

Figure 5.

The first fourteen cases described in Table 2 can be divided into four classes of finger control [Lands65a]: clawing motions (cases 1, 4, 10); opening motions (cases 2, 6, 8); reciprocal motions (5, 9, 11); and closing motions (cases 3, 7, 12). Thus, entire hand motion can be achieved by the cases described in Table 2. So once the AFNN can learn the cases in Table 2, we can then build an ANN for each finger (if need be) on a hand. These finger networks can then be incorporated and a prehensile commands database can be created that coordinates the fingers for specified animated goals.

## 5. SUMMARY

In this paper, an approach for creating an ANN to manipulate an animated human finger based on tendon tension control is described and tested. An ANN was developed and tested to flex and extend an animated finger. The results of our ANN produced correct tendon tension control and this could be viewed in our animation program. Using the cases described in Table 2, entire hand motion can be achieved and this is an important step in AIVH development. This work is an ongoing research endeavour.

## 6. ACKNOWLEDGMENTS

One author (OAK) was supported in part by the National Science and Engineering Research Council under a PGS-B scholarship.

## 7. REFERENCES


[Cavaz98a] Cavazza, M., Earnshaw, R., Magnenat-Thalmann, N., and Thalmann, D.: *Motion Control of Virtual Humans*, IEEE Computer Graphics and Applications, Volume 18, Number 5, pp. 24-31, 1998.

[Jacob76a] Jacobson, C., and Sperling, L.: *Classification of the hand-grip: a preliminary study*, Journal of Occupational Medicine, Volume 18, pp. 295-298, 1976.

[Kapan70a] Kapandiji, I.A.: *The Physiology of the Joints. Annotated diagrams of the mechanics of the human joints.*, Volume 1, Longman Group Ltd., London, G.B., 1970.

[Kucha96a] Kuchar, O.A.: *Finger Movements on a Keyboard: An Animated Simulation of the Biomechanics of the Hand*, Technical University of Nova Scotia, thesis, 1996.

[Kucha97b] Kuchar, O.A. and Scrimger, J.N.: *Biomechanical Modeling and Animating Human Hand Movement*, The Fifth International Conference in Central Europe on Computer Graphics and Visualization, Volume 2, pp. 242-251

[Lands65a] Landsmeer, J.M.F., and Long, C.: *The Mechanism of Finger Control Based on Electromyograms and Location Analysis*, Acta anat., Volume 60, pp. 330-347, 1965.

[Long64a] Long, C., and Brown, M.E.: *Electromyographic kinesiology of the hand: Pt. III. Lumbricalis and flexor digitorum profundus to the long finger.*, Arch. Phys. Med., Volume 43, pp. 450-460, 1964.

[Magne96a] Magnenat-Thalmann, N., and Thalmann, D.: *Computer Animation*, Handbook of Computer Science, CRC Press, pp. 1300-1318, 1996.

[Reyno87a] Reynolds, C.: *Flocks, Herds, and Schools: A Distributed Behavioral Model*, Proc. SIGGRAPH 1987, Computer Graphics, Volume 21, Number 4, pp. 25-34, 1987.

[Rumel86a] Rumelhart, D.E., Hinton, G.E., and Williams, R.J.: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1, pp. 318-362, 1986.

[Samad88a] Samad, T.: *Backpropagation is Significantly Faster if the Expected Value of the Source Unit is Used for Update*, Conference of the International Neural Network Society, 1988.

[Seire89a] Seireg, A., and Arvikar, R.: *Biomechanical Analysis of the Musculoskeletal Structure for Medicine and Sports*, Hemisphere Publishing Corp., USA, 1989.

[Thalm87a] Thalmann-Magnenat, M., and Thalmann, D.: *The Directions of Synthetic Actors in the Film Rendezvous a Montreal*, IEEE Computer Graphics and Applications, December, pp. 9-19, 1987.

[Tubia81a] Tubiana, R.: *The Hand*, Volume 1, W.B. Saunders Company, Toronto, Canada, 1981.

| CASE NUMBER | POSITION CHANGE FROM | POSITION CHANGE TO | ACTIVE TENDON |
|---|---|---|---|
| 1 | | | EDC |
| 2 | | | INTR |
| 3 | | | FDP |
| 4 | | | FDP |
| 5 | | | INTR |
| 6 | | | EDC |
| 7 | | | FDP |
| 8 | | | EDC+INTR |
| 9 | | | DOR |
| 10 | | | EDC+FDP |
| 11 | | | DOR+INTR |
| 12 | | | FDP |
| 13 | | | FDS |
| 14 | | | EDC+INTR |
| 15 | | | INTR |
| 16 | | | FDS |
| 17 | | | EDC |
| 18 | | | FDS |
| 19 | | | FDP |
| 20 | | | INTR |
| 21 | | | FDS |
| 22 | | | FDP |
| 23 | | | EDC+FDS |
| 24 | | | EDC+FDP |
| 25 | | | DOR+INTR |

25 cases of finger flexion and extension.

Table 2.