NeuroEvolution of Augmenting Topologies based Musculor-Skeletal Arm Neurocontroller

Ruoshi Wen, Zixi Guo, Tong Zhao，Xiang Ma, Qiang Wang, *Member IEEE*，Zhaojun Wu
Dept. of Control Science and Engineering, Harbin Institute of Technology
Harbin, China
Email: ruoshiwen_hit@sina.com, gzx81355@163.com, 16S004038@stu.hit.edu.cn，maxianghit@163.com, wangqiang@hit.edu.cn，wzjnb1989@163.com

Abstract—**Human-like musculor-skeletal arm model with its unique smooth and natural movements as well as low energy cost has attracted many researchers' interests and developed rapidly nowadays. In this paper, we focus on a human-like musculor-skeletal arm model with three links driven by nine muscles and propose a control model with two neurocontrollers trained by the NEAT algorithm. The NEAT (Neuroevolution of Augmenting Topologies) is a novel neuroevolution algorithm using a modified genetic algorithm to train neural networks by changing the topology structure as well as connection weights simultaneously. To evolve the network, we generate the training sets by using forward kinematics, geometry relationships, and muscle mechanic equations. By using two neurocontrollers, Position-Angle and Angle-Activation, the control model can deal with the redundancy and non-linear problems in separate neurocontroller at the same time.**

Keywords—musculor-skeletal arm; neurocontroller; genetic algorithm; NEAT

I. INTRODUCTION

Traditional robots' arms composed of joints and links are actuated by motors [1]. Different from human's smooth and dexterous movements based on the skeleton-muscle-tendon structure, these motor-actuated robots move very rigidly and unnaturally. In 1938, A.V. Hill first proposed the muscle model which is widely used in muscle movement studies [2]. Physiologically, when a muscle is activated by a certain level of stimulus, its contraction drives the joints thus move the arm to a corresponding position. Nowadays, the development in pneumatic soft materials and 3D printing makes it possible to emulate human's muscles' contraction motion [3], [4]. Robots based on musculor-skeletal model are human-like and low-power. They also have advantages such as compatibility and flexibility. What's more, modeling of human neuromuscular physiology plays an important role in furthering the state-of-the-art technology in rehabilitation [5], [6]. Each year, thousands of million people get injured because of car accidents, natural disasters and so on. The researches of musculor-skeletal arm control can not only promote the robotic industry, but also release the pain of people who are disabled by various reasons.

Inverse kinematics has been used to develop controllers that can move the robot arms to accomplish tasks such as arm positioning and target tracking [7]. One drawback using inverse kinematics control methods is that they are designed for specific environments and cannot be used to operate a robot arm in an environment with obstacles. Human's movements have both redundancy and non-linearity problems which make it rather difficult to solve inverse dynamics equations. In 1994, Karl Sims' evolved virtual creatures showed the potential of evolutionary algorithms to produce natural, complex morphologies and behaviors [8]. In this paper, we use an algorithm called NEAT [9] to train the musculor-skeletal neurocontroller. Compared to the fixed-topology artificial neural network, the NEAT uses genetic algorithm to evolve increasingly complex neural networks by evolving both topology and connection weights simultaneously. This algorithm has been used to evolve controllers effectively in robot control tasks such as pole balancing, playing Atari games and vehicle control [10].

In this paper, we focus on a three-link planar arm model driven by nine muscles in order to simulate the human arm. Firstly, we introduce the musculor-skeleton arm model and the non-linear muscle characteristics corresponding to the findings of A.V. Hill. Then, we introduce the NEAT algorithm and its unique features compared to the original neural network. Afterwards, by using forward kinematics, geometry relationships, and muscle mechanic equations, we generate the training sets which will be fed to the NEAT by activating the muscles and recording the corresponding end-point positions. Given the training sets, we will train two networks using the NEAT algorithm, one of which could output the joint rotations required to the target position, the other could output the muscle activations required to certain joint rotations. Despite the extra time taken than only training one network, two neurocontrollers consider both redundancy and the non-linear characteristics. In this way, the control system of the musculor-skeletal arm is more precise and easier to analyze when system failure happens. Moreover, each neurocontroller has its unique function and can be encapsulated into a module which will strengthen its generality, i.e. the neurocontroller can be applied to situations other than the musculor-skeletal arm.

II. Neurocontrollers for Musculor-Skeletal Arm

A. Musculor-skeletal arm model

We use a planar musculor-skeletal arm model consisted of three serial links, six mono-articular muscles, and three bi-articular muscles to imitate human's arm moving in a horizontal plane. This arm model includes three joints separately imitating the shoulder joint, the elbow joint and the wrist joint. The model [11] is shown in Fig. 1.

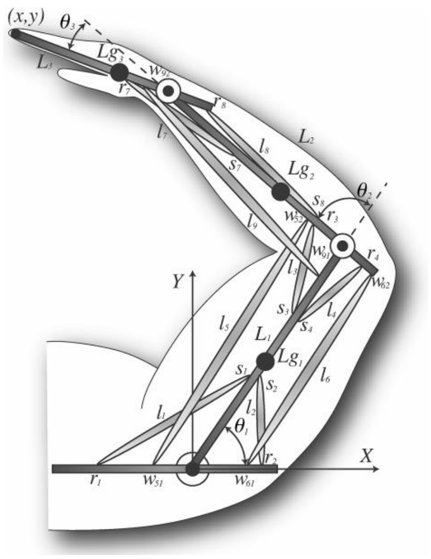

Fig. 1. Musculor-skeletal planar arm model with six mono-articular muscles and three bi-anticular muscles.

The muscle lengths of the musculor-skeletal arm model can be formulated as the distance between the insertion points of the muscles. They can be expressed as follows:

$$\boldsymbol{l}(\boldsymbol{\theta}) = (l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9)^T$$

$$= \begin{pmatrix} (\gamma_1{}^2+s_1{}^2+2\gamma_1 s_1\cos\theta_1)^{\frac{1}{2}} \\ (\gamma_2{}^2+s_2{}^2-2\gamma_2 s_2\cos\theta_1)^{\frac{1}{2}} \\ (\gamma_3{}^2+s_3{}^2+2\gamma_3 s_3\cos\theta_2)^{\frac{1}{2}} \\ (\gamma_4{}^2+s_4{}^2-2\gamma_4 s_4\cos\theta_2)^{\frac{1}{2}} \\ (\omega_{51}{}^2+\omega_{52}{}^2+L_1{}^2+2\omega_{51}L_1\cos\theta_1 \\ +2\omega_{52}L_1\cos\theta_2+2\omega_{51}\omega_{52}\cos(\theta_1+\theta_2))^{\frac{1}{2}} \\ (\omega_{61}{}^2+\omega_{62}{}^2+L_1{}^2-2\omega_{61}L_1\cos\theta_1 \\ -2\omega_{62}L_1\cos\theta_2+2\omega_{61}\omega_{62}\cos(\theta_1+\theta_2))^{\frac{1}{2}} \\ (\gamma_7{}^2+s_7{}^2+2\gamma_7 s_7\cos\theta_3)^{\frac{1}{2}} \\ (\gamma_8{}^2+s_8{}^2-2\gamma_8 s_8\cos\theta_3)^{\frac{1}{2}} \\ (\omega_{91}{}^2+\omega_{92}{}^2+L_2{}^2+2\omega_{91}L_2\cos\theta_2 \\ +2\omega_{92}L_2\cos\theta_3+2\omega_{91}\omega_{92}\cos(\theta_2+\theta_3))^{\frac{1}{2}} \end{pmatrix}, \quad (1)$$

where l_i (i = 1-9) are the lengths of the muscles, and θ_i (i = 1-3) are the joint angles. Also, γ_i (i = 1-4, 7-8) and s_i (i = 1-4, 7-8) are the distances between the center of each joint and the insertion point of each mono-articular muscle, and ω_i (i = 51, 52, 61, 62, 91, 92) are those of bi-articular muscles.

B. Non-linear muscle model

The muscle produces contraction force through muscle fibers' contracting activity [12]. The contraction force is transmitted to the muscle-tendon which will apply the force to the joints. Each muscle is based on Hill's three-component model. The model is composed of three elements: the contractile element (CE, muscle fibers), the parallel elastic element (PEE, connective tissue around the fibers and fiber bundles), and the series elastic element (SEE, muscle tendon).

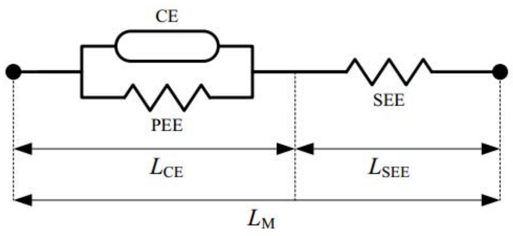

Fig. 2. The three-component model of Hill type muscle.

The force produced by contractile element, F_{CE}, depends on the constant maximum isometric force of the muscle, F_{max}, the muscle activation a, the fiber length L_{CE}, and the contraction velocity V_{CE}, shown in (1):

$$F_{CE} = a\, F_{max}\, f_L(L_{CE})\, f_V(V_{CE}) \quad (2)$$

where f_L is a function of the contraction force and muscle length, while f_V is a function of the force and current contraction velocity. Normalized force-length and force-velocity relations of the contractile element are shown in Fig. 3.

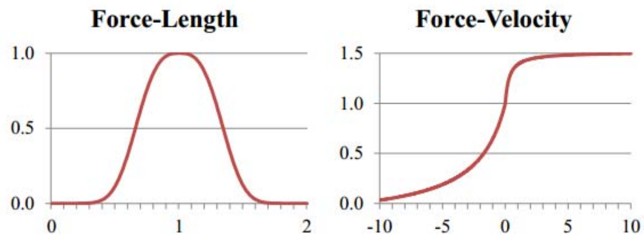

Fig. 3. Normalized force-length and force-velocity relations.

Here comes a conclusion that if the muscle length and contraction velocity are known, the contraction force can be determined by the muscle activation. The force applied on the joints produces torque which can change the joints' angles, thus changing the posture of the arm. In turn, the change of the arm posture can bring changes to the muscle length which can be calculated using (1). Also, the contraction velocity is the differential muscle length to time.

C. The NEAT genetic algorithm

The NEAT is a powerful method for artificial evolving neural networks, which can evolve the network topology along with the connection weights simultaneously. It has been shown effectively in many applications such as pole balancing, robot control, vehicle control, board games and videogames.

The NEAT improves the GA (genetic algorithm) and artificial neural network through the following four aspects:

(1) Use genetic encodings to describe the nodes and the connected genes.

(2) Every connection in a single gene is labeled an innovation number used as historical marking for us to track genes.

(3) Protect innovation through speciation.

(4) Minimize dimensionality through incremental growth from minimal structure.

D. Control Model with Position-Angle and Angle-Activation Neurocontrollers for Musculor-skeletal Arm

Using the above musculor-skeletal arm model and neurocontrollers trained by the NEAT algorithm, we design a musculor-skeletal control model shown in Fig. 4.

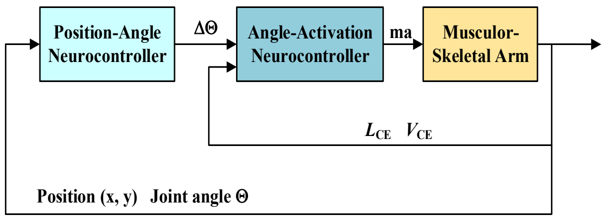
Position-Angle Neurocontroller
ΔΘ
Angle-Activation Neurocontroller
ma
Musculor-Skeletal Arm
L_{CE} V_{CE}
Position (x, y) Joint angle Θ

Fig. 4. Control model with Position-Angle and Angle-Activation Neurocontroller for musculor-skeletal arm

The robot arm is configured with sensors along its three joints to measure joint angles. The arm also has a range sensor configured at the endpoint which provides the target position (x, y) relative to the endpoint. The Position-Angle Neurocontroller, provided with the current joint angles Θ and the target position (x, y), generates a set of changes of joint angles ΔΘ.

In the meantime, the Angle-Activation Neurocontroller provided with the current muscle length L_{CE}, and velocity V_{CE} as well as ΔΘ outputs a set of muscle activations a and applies them to the musculor-skeletal arm model. Execute the above steps several times until the position of the endpoint is close to position (x, y). Then the two neurocontrollers have done their jobs and moved the arm to the target position.

Algorithm 1 MSAP (Musculor-Skeletal Arm Position)

```
Input:
   the current position of target (x, y), the current joint angle
   vector Θ, the current muscle length vector L_CE, the current
   contraction velocity vector V_CE.
Output:
1: Calculate the distance d = || (x, y) || between the end-point
   and the target.
2: if d < ε then
3:     Output the joint rotation vector ΔΘ = 0
4: else
5:     Based on the current target position (x, y) and the
       current joint angle vector Θ, the Position-Angle
       Neurocontroller generates a set of the changes of
       joint angles.
6: end if
7: The Position-Angle neurocontroller outputs the joint
   rotation vector ΔΘ.
8: Given the current muscle length vector L_CE, the current
   contraction velocity vector V_CE, and the joint rotation
   vector ΔΘ, the Angle-Activation controller generates a set
   of activations of each muscle
9: Output the muscle activation vector ma.
```

The control model with the Position-Angle Neurocontroller and the Angle-Activation Neurocontroller for musculor-skeletal arm can solve the redundancy and non-linearity problems at the same time. We will introduce how to generate the training sets through forward kinematics and how to use the given training sets to train the neurocontrollers through the NEAT algorithm in the next section.

III. Experiement

To obtain the neurocontrollers, a series of experiments are needed. To start with, we use the above arm model to generate two training sets, one of which is the relationship of position (x, y) and the joint rotations, the other is the relationship of the joint rotations and muscle activation level. Then, we use the two training sets to train the Position-Angle Neurocontroller and the Angle-Activation Neurocontroller.

The training set must be carefully chosen so that the neurocontroller learns general behavior that allows it to effectively control the arm in situations that are not present in the training set. Because our task is moving the arm to the target without obstacles, we can use a relatively easy way to generate the training sets. We randomly give the joints a small rotation and record the current joint angles and the position of the endpoint.

A. Generate the position-angle training set

To avoid oscillation in the process of moving the arm to the target, a small joint rotation at every timestep is recommended. This way, the arm can gradually approach the target more precisely without oscillation step by step.

To generate the position-angle training set, we place several target points uniformly in the arm's reach space. The range sensor can output the target position (x, y) and the angle sensors can output the current joint angles. We randomly give joint rotations between [-5, +5] degrees at every timestep and record the target position and the current joint angles, in which case we get a set of data at every timestep.

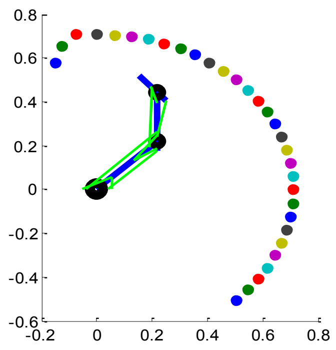

Fig. 5. Target points are placed uniformly in the arm's reach space.

As is shown in Fig. 5, the arm can move in the region bounded by the colorful round points. On the track from the initial place to a certain target, we take 200 samples. There are 31 targets which means we can get 200*31=6200 datasets.

B. Generate the angle-activation training set

According to (2), F_{CE} depends on the muscle length L_{CE}, the contraction velocity V_{CE}, f_{L}, f_{V} and muscle activation. Given L_{CE} and V_{CE}, the relationship between F_{CE} and muscle activation is determined. F_{CE} is applied to the joint and produces torque which brings the joint rotations.

Because joint rotations in every time step is limited between [-5, +5] degrees, it's necessary to determine what level of activation can produces such joint rotations. We firstly conduct an experiment to get the maximum activation level of each muscle by separately activating each muscle until the joint rotation is equal to or close to -5 degree or +5 degree. In this way, we can estimate a range of muscle activation level approximately.

Since different muscles have different functions, and one muscle mainly influences a single joint, i.e. brings more angle changes than the other joints. Given a certain activation, one muscle can either increase or decrease the joint angle dominated by the muscle. We divide the range of each muscle into two and get three activation levels of 0, 0.5*maximum, and maximum. Thus, we get a training set whose size is 3^9=72171.

To generate the angle-activation training set, the muscle lengths and contraction velocities are needed. For a given arm model, γ_i (i = 1-4, 7-8), s_i (i = 1-4, 7-8) and ω_i (i = 51, 52, 61, 62, 91, 92) are constants. θ_i (i = 1-3) of the joints can be measured by sensors, given the above parameters, the muscle lengths can be calculated by (1). In a minor timestep, the muscle contraction velocity can be regarded as the change of the muscle length. Therefore, a complete training set including joint rotations, muscle lengths and muscle contraction velocities can be obtained.

C. Use the NEAT to train Position-Angle Neurocontroller

The network has three outputs that determine how much each joint changes at every timestep. The joint angles as analyzed above have a threshold between [-5, +5] degrees, which forces the neurocontroller to make a series of minor rotations towards the target position. Fig. 6 shows the configuration of the networks evolved to control the arm to the target.

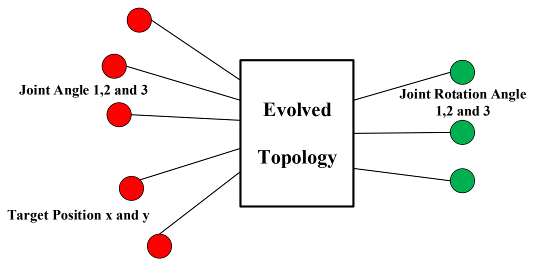

Fig. 6. Configuration of the networks evolved to train Position-Angle Neurocontroller.

As is shown in Fig. 6, the inputs to the neurocontroller is the current joint state and the target position relative to the arm's endpoint. The outputs are the joint rotations. The initial network is composed of inputs directly connected to the outputs with random weights. As the network evolves, more nodes and connections are added and the weights are modified in the meantime so that the network can work effectively, i.e. given a set of reasonable inputs, the outputs of joint rotations can move the arm to a certain position close to the target position (x, y).

D. Use the NEAT to train Angle-Activation Neurocontroller

Joint rotations are produced by activating the muscles. The nine muscles cooperate to produce forces and apply them to each joint, thus rotate the joints. So the inputs to the network include the joint rotations. However, the contraction force which brings changes to the joint angles is not determined by the only parameter, muscle activation. The inputs to the network also include two other components: nine muscle lengths, and nine muscle contraction velocities, as shown in Fig. 7.

The network has nine outputs that determine to what extent the muscle activations are needed.

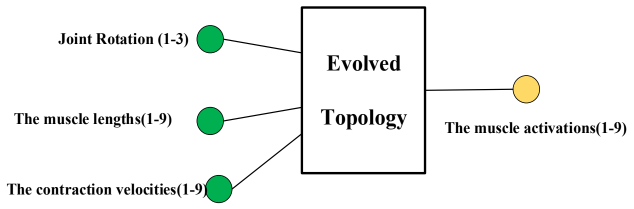

Fig. 7. Configuration of the networks evolved to train Angle-Activation Neurocontroller.

Similar to the evolving process of the Position-Angle Neurocontroller, the network starts evolving from a minimal topology structure with random weights. As the network evolves, an exact relationship between inputs and outputs can be described by this network and the Angle-Activation Neurocontroller outputs the muscle activations to actuate the arm.

E. Results and Analysis

We conduct an experiment to roughly evaluate the two neurocontrollers as well as analyze the musculor-skeletal arm's movements. We first initialize the arm by setting the activation level of each muscle to zero. The initial arm posture is shown in Fig 8 and the red point is the target. As we can see from Fig. 8, the target position is (-0.05, -0.15) relative to the endpoint of the arm. After a few timesteps, as shown in Fig. 9, the shoulder and wrist angle decrease while the elbow angle increases. The blue points forming a track line of the endpoint moving from the initial place towards the target position. The target position is (-0.05, -0.08) which indicates the arm's endpoint is getting closer to the target from the initial position.

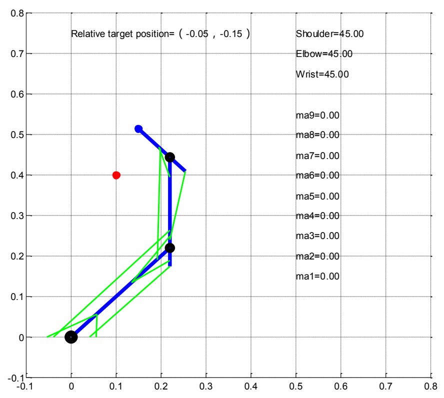

Fig. 8. Initial musculor-skeletal arm state with no muscle activation.

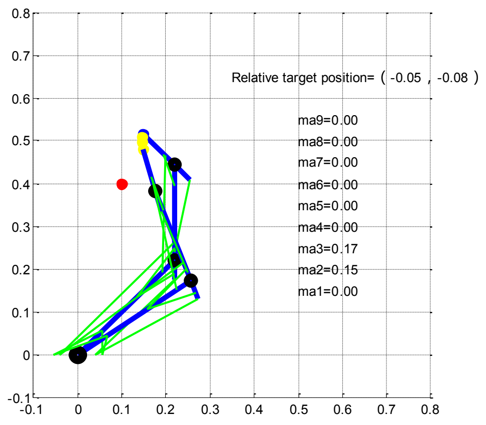

Fig. 9. Musculor-skeletal arm state after a few timesteps.

As time goes, the musculor-skeletal arm moves closer and closer towards the target. As shown in Fig. 10, the relative target position is (0, -0.002), which indicates that the endpoint of the arm reaches the target. This experiment demonstrates that the control model of evolving neurocontrollers through the NEAT can move the musculor-skeletal arm to a specific position and converge the distance between the endpoint and the target to the minimum.

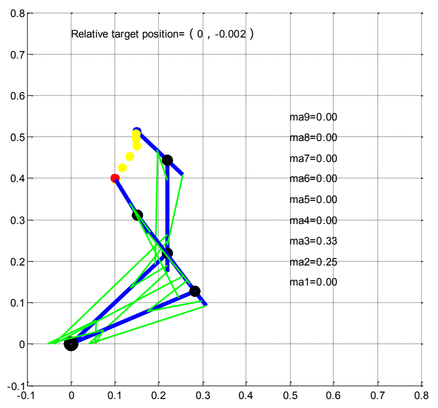

Fig. 10. The track of the arm endpoint moving from the initial place to the target.

We also run several experiments of different population size to compare the maximum fitness, generation in which the first peak fitness value appears and the number of hidden nodes within 150 generations to find the most suitable population size. To simplify this process, we choose10 datasets uniformly in the reachable space to train the network. Inputs are the target position (x, y) and outputs are the activations of nine muscles. Less data can also evaluate the performances of the network training by the NEAT algorithm. As we all know, neural networks with fixed topology structure behave badly when there is not enough data, however, the neural networks trained by the NEAT algorithm by updating the topology structure and connection weights simultaneously perform pretty well despite consuming more time.

TABLE I

THE PERFORMANCE COMPARISON OF DIFFERENT POPULATION SIZE

Population size	generation	max fitness	No. of average hidden nodes
50	23	0.838	4.22
100	47	0.860	5.27
150	69	0.838	4.78
200	30	0.846	5.66
300	92	0.867	6.71
500	117	0.870	10.27

As is shown in Tab. I, overall, along with the population size increases, the maximum fitness gets larger while it takes more generations to reach the peak fitness, and the number of hidden nodes increases. The increasing hidden nodes indicates that this NEAT algorithm does evolve the topology of the neural network. However, the network with population size of 100 behaves better than that of 150 and 200, which illustrates that blindly increasing the population size can't assure that the performance will get better.

IV. Conculsion

In this paper, we propose a control model with two neurocontrollers trained by the NEAT algorithm that could move the end-point of the musculor-skeletal robot arm close to the target positions. First, we generate a training set using forward kinematics and geometry relationships between joint rotations and the target position. By activating the muscles and recording the joint rotations, the second training set is generated. Second, the Position-Angle Neurocontroller and the Angle-Activation Neurocontroller, are trained by the NEAT algorithm using the above two training sets. As we can see from the training processes of the two networks, both the topology and the connection weights are updated simultaneously, which is more effective compared to the fixed-topology neural network updated by genetic algorithms or backpropogation. The experiment conducted by us indicates that the two neurocontrollers can move the musculor-skeletal arm to the target gradually. In addition, the comparison experiments of evolving neural works with different population size show that increasing the population size can improve the performance to a certain extent. In conclusion, the advantages of using musculor-skeletal model to build robot arms are smooth, natural and human-like movements. Also, the two neurocontrollers in this control model can deal with the redundancy problem and the non-linear problem in separate neurocontroller at the same time.

ACKNOWLEDGMENT

This work was supported and funded by NSF. The authors also want to express the gratitude to Dr. Travis of University of Waterloo for his help with the muscle model.

REFERENCES

[1] Richard. M. Murray, Zexiang Li, S. Shankar Sastry, and S. Shankara Sastry, "A Mathematical Introduction to Robotic Manipulation," Boca Raton, Florida: CRC Press, pp. 211-214, 1994.

[2] A.V.Hill, "The Heat of Shortening and the Dynamic Constants of Muscle," Proceedings of the Royal Society B, 1938.

[3] Bobak Mosadegh, Panagiotis Polvgerinos, Christoph Keplinger, Sophia Wenstedt, Robert F. Shepherd, Unmukt Gupta, Jongmin Shim, Katia Bertoldi, Conor J. Walsh, and George M. Whitesides, "Pneumatic Networks for Soft Robotics that Actuate Rapidly," Advanced Functional Materials 24(15), April, 2014.

[4] Caroline Cvetkovic, Ritu Raman, Vincent Chan, Brian J. Williams, Madeline Tolishe, Piyush Bajaj, Mahmut Selman Sakar, H. Harry Asadad, M. Taher A. Saif, and Rashid Bashir, "Threedimensionally printed biological machines powered by skeletal muscle," PNS, vol.111, no.28, Jul, 2014.

[5] Manish Sreenivasa, Ko Ayusawa, and Yoshihiko Nakamura, "Modeling and Identification of a Realistic Spiking Neural Network and Musculoskeletal Model of the Human Arm, and an Application to the Stretch Reflex," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.24, Issue.5, May 2016.

[6] B. Ghannadi, N. Mehrabi, J. McPhee, "Development of a human-robot dynamic model to support model-based control design of an upper limb rehabilitation robot," ECCOMAS Thematic Conference on Multibody Dynamics, July 2015.

[7] Taku Komura, Yoshihisa Shinagawa, and Tosiyasu L. Kunii, "An Inverse Kinematics Method Based on Muscle Dynamics," In proceedings of Computer Graphics International, pp.15-22, 2001.

[8] K. Sims, "Evolving virtual creatures," In proceedings of the 21st Annual Conf. on Computer Graphics and Interactive Techniques, ACM, pp. 15-24, 1994.

[9] K.O. Stanley and R. Miikkulainen, "Evolving neural networks through augumenting topologies," Evolutionary Computation, 10(2), pp. 99-127, 2002.

[10] E. Haasdijk, A.Rusu, and A.Eiben, "HyperNEAT for Locomotion Control in Modular Robots", Evolvable Systems: From Biology to Hardware, pp. 169-180, 2010.

[11] Kenji Tahara, Suguru Arimoto, Masahiro Sekimo, and Zhi-Wei Luo, "On control of reaching movements for musculo-skeletal redundant arm model," Applied Bionic and Biomechanics, vol. 6, no.1, pp. 57-72, March 2009.

[12] Thomas Geijtenbeek, Michiel van de Panne, and A.Frank van der Stappen, "Flexible Muscle-Based Locomotion for Bipedal Creatures," ACM Trans. Graph. (SIGGERAPH Asia 2013), vol. 32, no.6, November 2013.