IEEE International Conference on Humanoid Robots (HUMANOIDS 2018).
Beijing, China, November 6-9.

Humanoid Robot Grasping with a Soft Gripper Through a Learned Inverse Model of a Central Pattern Generator and Tactile Servoing

Conference Paper · November 2018

DOI: 10.1109/HUMANOIDS.2018.8625035

Humanoid robot grasping with a soft gripper through a learned inverse model of a central pattern generator and tactile servoing

Yuxiang Pan , Fred Hamker , and John Nassour

Artificial Intelligence, Computer Science, Chemnitz University of Technology, 09111, Chemnitz, Germany

***Abstract*— Grasping and manipulation are essential skills that humanoid robots need in order to operate in the human environment. Model-based methods require a precise calibration and suffer from high order non-linearity. While, neural-based representations does not require a dedicated calibration process to solve these tasks. However, some suffer from high generalization error that reduces the accuracy or require large-scale data collection. The role of sensory feedback is therefore important to adapt the action. We present a control framework to learn grasping with a soft gripper attached to a humanoid robot arm. The inverse kinematic model of the arm is acquired through motor babbling of a central pattern generator and encoded by a feed-forward neural network. To overcome the generalization error we provide the gripper with a tactile sensors array at each finger. The tactile servoing is used to correct the action before grasping. The proposed model has been tested in simulation, and on the real robot where a soft sensory gripper was used to interact with a human subject (Tactile Servoing). Successful grasping was achieved thanks to the integration of a learned inverse model with the sensory feedback.**

I. Introduction

Traditional methods based on inverse kinematics and hand-eye coordination to solve robot grasping require calibration to build a precise representation of the end effector in the camera space. This calibration suffers from time consumption due to the non-linear optimization [1], [2]. A feedforward neural network has been used to build an inverse kinematics model for robotic arm and to perform the hand-eye calibration [3]. A convolutional neural network with a large scale data collection was proposed recently to address the coordination problem for robotic grasping in 2D task space [4]. Deep leaning methods require a large-scale data collection (e.g., 800,000 grasp in [4]), which is not always possible to do on a humanoid robot. A common problem appears when a neural network is used to represent the inverse kinematics model, it consists in the generalization error. This error leads to inaccurate grasp. Pastor et al. has addressed the role of sensory feedback in the adaptation of motor program. Based on the expectation of tactile sensory activation, tactile sensors implemented on a rigid robotic hand has been used to adapt the motor patterns, named dynamic movement primitives DMPs, during grasping in 2D task space [5]. In this paper, we propose a framework to learn robot grasping of an object with ArUco markers in 3D space for hand-eye coordination, see Fig. 1. We use a feedforward neural network as an inverse kinematics representation and a central pattern generator for motor patterns. Grasping is performed with a soft robotic hand equipped with soft tactile sensors. Grasping behavior is composed of two parts, the first is generated by the neural network that encodes the hand-eye coordination, while the second is a servoing behavior that is

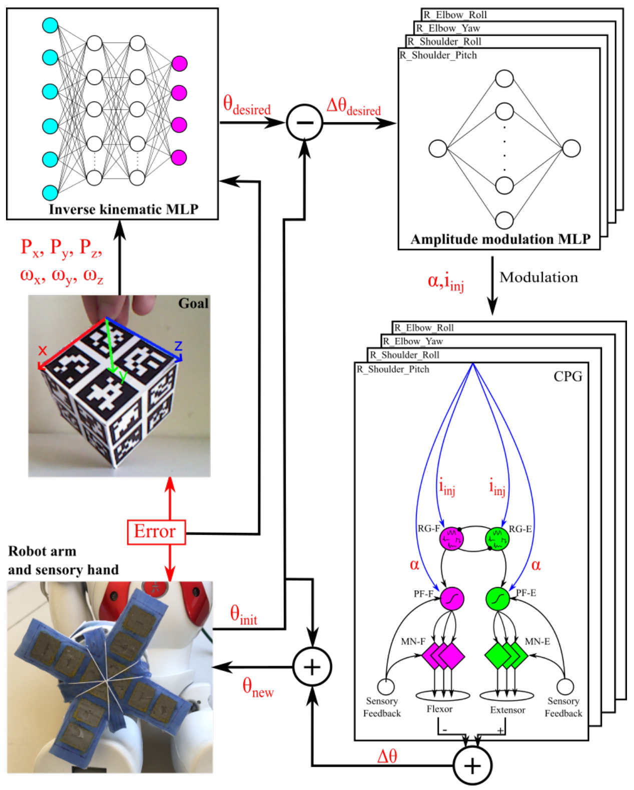

Fig. 1: The control framework for grasping in 3D space. The camera captures the position and the orientation of the ArUco markers cube. Camera frame is used to represent the desired position and orientation. A multilayer perceptron neural network calculates the desired joints' position $\theta_{desired}$ which will be subtracted from the current configuration θ_{init}. Joints variations $\Delta\theta$ are then introduced to the central pattern generator modulation networks to obtain appropriate CPG parameters that leads to the desired end-effector position. An external camera has been used to cover larger space in front of the robot that can not be viewed by the robot on-board cameras. `https://youtu.be/O6AxZUecGg8`

yuxiang.pan@hotmail.com
fred.hamker@informatik.tu-chemnitz.de
john.nassour@informatik.tu-chemnitz.de

generated using tactile sensors. This paper is organized as follows. Section II represents the central pattern generator model used to control robot joints, then it represents the neural modulation of these patterns in the joint space. Section III describes the motor babbling and the learning of inverse kinematic model. Soft gripper and tactile sensory feedback are presented in Sec. IV. Tactile servoing, reaching, and grasping experiments are illustrated in Sec. V. Finally, we present the conclusion in Sec. VI.

II. Pattern generation

A. Central pattern generator model

The Multi-layered multi-pattern central pattern generator (MLMP-CPG) model is a computational neural model which is used to generate rhythmic and non-rhythmic behavior on the robots' legs and arms [6], [7]. Each CPG model controls one joint, and it is divided into three layers: Rhythm Generator (RG), Pattern Formation (PF) and Motor Neuron (MN), see Fig. 1.

Rhythm generation neurons (RG) are based on two cells with self-rhythmic generation ability [8]. The neural model for each neuron is represented by:

$$\tau_m \frac{dV}{dt} = -(\text{fast}(V, \sigma_f) + q - i_{\text{inj}}), \tag{1}$$

$$\tau_s \frac{dq}{dt} = -q + q_\infty(V), \tau_m < \tau_s, \tag{2}$$

$$\text{fast}(V, \sigma_f) = V - A_f \tanh((\sigma_f / A_f)V), \tag{3}$$

$$q_\infty(V) = \sigma_s(V - E_s), \tag{4}$$

where V is the membrane potential, and represents the RG neuron output. q and q_∞ are the slow current and its steady state value, respectively. A_f determines the width of the N shaped current-voltage curve. τ_m and τ_s are time constants. i_{inj} is the injected current. σ_s and σ_f represent the conductance for potassium and calcium currents, respectively. E_s is the reversal potential. Different patterns such as quiescence, almost an oscillator, oscillations, and plateau, can be generated by tuning the aforementioned cell parameters. The current paper is built on the generation of plateau patterns in the grasping task. Pattern formation neurons (PF_E, PF_F) receive input from RG neurons. The activation function is expressed by:

$$PF = \frac{1}{1 + e^{\alpha_0 \cdot \alpha(\psi_0 - w_{\text{RG}\to\text{PF}} \cdot V)}}, \tag{5}$$

where $\alpha_0 = 1$ is the slope of the sigmoid, $\psi_0 = 0$ defines the center point. α represents the descending control from the high-level controller that modulates the activation of the pattern formation neuron PF, see Fig. 1. $w_{\text{RG}\to PF}$ is the weight of the synaptic connection between RG and PF neurons. Each PF neuron projects to the corresponding motor neuron (MN). The activation function of each extensor and flexor motor neuron is expressed by:

$$MN = \frac{1}{1 + e^{\xi(\beta - (w_{\text{PF}\to\text{MN}} \cdot PF + w \cdot S)/2)}}, \tag{6}$$

where, $\xi = 5$ and $\beta = 0.5$ are the slope and threshold of the sigmoid activation function, respectively, whose values were set empirically. $w_{\text{PF}\to\text{MN}}$ is the weight of the synaptic connection between PF and MN neurons. S is the proprioceptive sensory feedback, and w is its corresponding weight. In the current study w is set to 0, because no sensory feedback is considered at the motorneuron level. CPG Patterns are generated by RG neurons where i_{inj} is responsible for temporal coordination, while the spatial coordination is done by varying α in PF neurons.

B. Pattern modulation

To modulate the amplitude of the generated pattern for a desired joint motion, we used multilayer perceptron as an approximation function. Since the rhythm generation neurons select only the nature of a pattern, the pattern's amplitude modulation occurs in pattern formation neurons. For example, to move one joint by $\Delta\theta$ from its current configuration θ_{init}, α in pattern formation layer has to be tuned to perform that motion. Since the range of motion for each joint may differ from other joints, each joint will have its own MLP network to approximate the relationship between α and $\Delta\theta$, see Fig. 1. The desired joint angle is the sum of the initial joint angle and the output of the CPG ($\theta = \theta_{init} + \Delta\theta$). At the end of the motion, the new joint angle becomes the initial one for the successive CPG pattern. The training data for each joint's neural network is collected

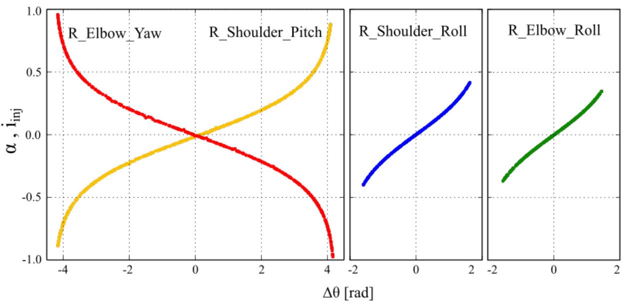

Fig. 2: The relationship between joints variation $\Delta\theta$ and the pattern formation parameter α. A separated MLP neural network was used for each joint to approximate the relationship.

by motor babbling. For a random α value in the CPG, the joint perform a random motion $\Delta\theta$. If a joint reaches one of its mechanical limits, the training data producing this motion will be eliminated from the training set. The neural network for each joint has one input ($\Delta\theta$), one output (α), and one hidden layer with 25 neurons. As activation function, "Tanh" was used for the output layer neurons, while "ReLu" was used for the hidden layer. The loss is calculated using Root Mean Squared Error (RMSE), which is the the average of the square root of the errors between the predicted value and the target value. Figure 2 shows the input-output relationship after learning the pattern modulation for each joint. The average of obtained error is around $0.0003[rad]$.

III. Neural network for inverse kinematics

To solve the inverse kinematics problem, a multilayer perceptron neural network is introduced to find the relationship between the 3D space and joint space, see Fig. 3. In this paper, the right arm of NAO humanoid robot is used to perform grasping. The right arm of NAO robot has 5 joints “ShoulderPitch”, “ShoulderRoll”, “ElbowYaw”, “ElbowRoll” and “WristYaw”. To reduce the number of training data that need to be collected, we have used only the first four joints.

A. Data collection

To collect the data set which is required to train the network, we have rigidly attached an ArUco marker cube to the end-effector of the right arm. A camera has been fixed in front of the robot (distance of 80[cm]) to detect the positions and orientations of the cube during motion. The robot was in standing position, while the right arm moves with a random selection of four joint angles. The right arm’s joint variables, the position, and the orientation of the marker have been collected. We have obtained 10,000 sets that will be used for the learning process.

B. Learning

The MLP network dedicated for inverse kinematics has six input neurons, which are the orientation parameters (ω_{x}, ω_{y}, ω_{z}) and the position parameters (P_{x}, P_{y}, P_{z}) of the end-effector in the 3D Cartesian space, see Fig. 3(top). The network has four output neurons representing the joint angles ($\theta_{\mathrm{S.P.}}$, $\theta_{\mathrm{S.R.}}$, $\theta_{\mathrm{E.Y.}}$, $\theta_{\mathrm{E.R.}}$), and 2 hidden layers with 25 neurons with “sigmoid” activation function. Root mean square error “RMSE” was selected as loss function. It calculates the difference between the actual output and the desired one, (7):

$$RMES = \sqrt{\frac{\sum_{i=1}^{n} (d_i - o_i)^2}{n}} \tag{7}$$

where d_i is the desired value (ground truth labels), o_i is the network output at time i, and n is the number of data set. During the training process, Cross-Validation is applied to the MLP network. The training sample were split into a training set and a testing set. 80% of the samples were randomly selected for training, and 20% for testing. Because this neural network has multiple inputs and the ranges of these inputs are different, hence the inputs are normalized into the range of 0 to 1 as follow:

$$P = \frac{P_{\mathrm{desired}} - P_{\min}}{P_{\max} - P_{\min}}, \omega = \frac{\omega_{\mathrm{desired}} - \omega_{\min}}{\omega_{\max} - \omega_{\min}}, \tag{8}$$

where P_{desired} and $\omega_{\mathrm{desired}}$ are vectors that describe the desired end-effector’s position and orientation in the Cartesian space, respectively. $P_{\min}$, $P_{\max}$, $\omega_{\min}$, $\omega_{\max}$ are vectors that describe the “min” and “max” values for the end-effector position and orientation, they are obtained from the training set. Figure 3.(bottom) shows the histogram of the error at each joint after learning the inverse model.

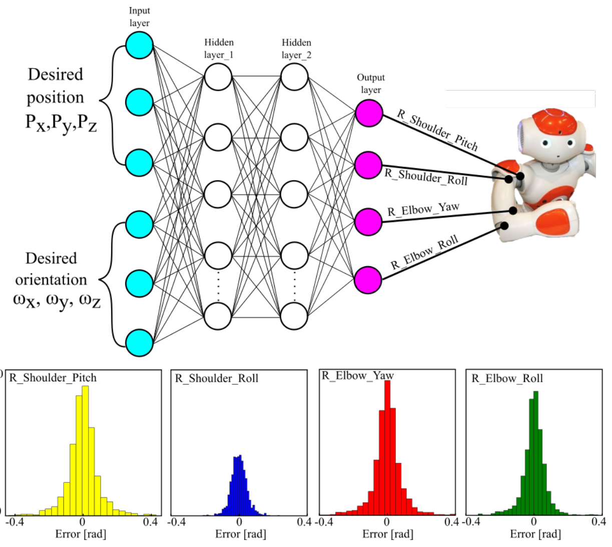

Fig. 3: Inverse model neural network, two hidden layers with 25 neurons each layer, one input layer with 6 input neurons and one output layer with 4 output neurons (top). Histograms of the error of inverse kinematics based on the neural network representation (bottom).

IV. Sensory soft gripper

To compensate the end-effector error caused by the generalization error of the neural network, we divide the reaching task into two subtasks: the first task uses the inverse model encoded by the neural network, while the second task relies on the tactile feedback to correct the network error.

A. fabrication and sensory set-up

Sensory feedback are essential to interact with environment and to adapt the robot actions. In our previous work, a sensory soft hand has been developed [9]. The hand can be easily worn by the Nao humanoid robot. In the current work, we include 12 capacitive sensors to provide tactile information. Sensors are made of conductive fabric sewed on cloth fabric, where we used conductive threads to connect the conductive fabrics to the capacitive touch sensor breakout “MPR121” which is connected to a Raspberry Pi board. The board is connected to the robot by an Ethernet cable.

B. Tactile feedback

There are 12 capacitive sensors on the soft gripper, three sensors per finger, see Fig. 1. When a conductive object approaches the gripper, the capacitive sensors will be activated based on the object position and the distance from each sensor. We determine the center position of the object (y_c, z_c) with respect to the center of the gripper using:

$$f_y = \sum_{i=1}^{N_y} f_i \quad , \quad y_c = f_y^{-1} \sum_{i=1}^{N_y} f_i \cdot y_i, \tag{9}$$

where f_i is the normalized i^{th} sensor reading. y_i is the position of i^{th} sensor on $y - axis$. N_y is the number of sensors positioned on $y - axis$. Figure 4 illustrates the

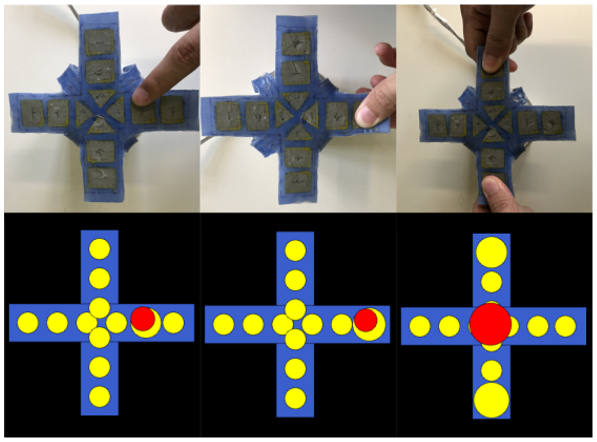
Fig. 4: The sensory gripper was subject to different stimulations from the sensors side (top). Capacitive sensors activations in yellow and the center of stimulation in red (bottom).

capacitive sensors activation (in yellow), and the center position of the stimulation (in red) during the interaction with a human hand. We show three different situations: a small force applied on a single sensor, more important force applied on one sensor, and two forces applied on two different sensors at the same time.

V. Experimental Results

A. Reaching a point

We first tested the MLP neural network that modulate the inverse kinematics in the V-REP simulator. This has been done by creating two virtual cubes in the simulator, one in white color and one in red.

Each of whose sides has four different "ArUco" markers which create a distinctive pattern. Then we placed a fixed camera 80 [cm] away from the NAO robot and 40 [cm] above the ground. The red cube was rigidly attached to the robot's hand, and it was set to be invisible to the camera as we used it only to measure the error during the experiment. For a given position and orientation for the white cube in the 3D space (identified by the camera), the MLP neural network is presented in Sec. III provides the desired joint angles for right arm to reach the cube. The initial joint angle θ_{init} will be substituted from desired joint angle $\theta_{\mathrm{desired}}$ to work out the desired motion $\Delta\theta_{\mathrm{desired}}$, which is required for each joint to reach the goal, see Fig. 1. For each joint variation, the CPG modulation parameters $\alpha \times i_{\mathrm{inj}}$ will be calculated as in Fig. 2. Then, a CPG plateau pattern will be generated at each joint to move the robot hand into the desired configuration. The error of the neural network for encoding the inverse kinematics are presented by the difference in the translations and orientations between both cubes. Ideally, the white cube is overlapped by the red cube. But due to the mechanical limitation of the robot (non-redundant), it is not possible to reach a point inside the workspace with any desired orientation. Figure 5 shows several reaching experiments in V-REP simulation and also on the real robot with the sensors soft gripper. The distance between the two cubes' center points is used to measure the magnitude of the error. We made 30 reaching trials in simulation, the maximum error was 2.13 [cm] and average error was 0.43 [cm].

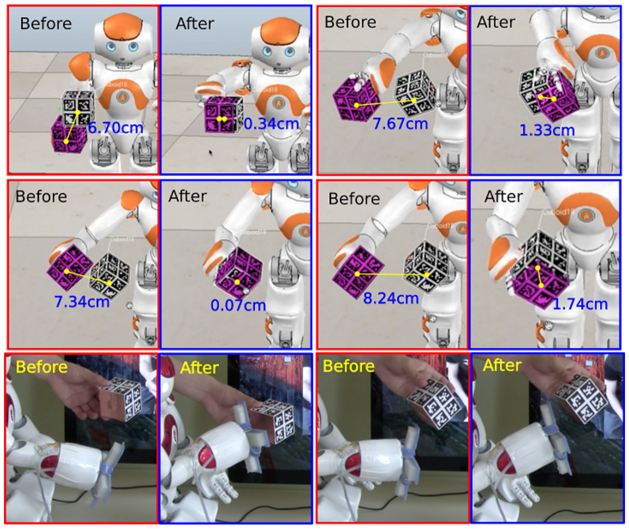

Fig. 5: Reaching test in simulation and real robot. Error is measured in simulation by the distance between the two cubes' center points after the action.

B. Following a trajectory

This section introduces experiments for a predetermined hand trajectories. In the first experiment, the robot hand has to move along a straight line (15 [cm] length) with a fixed orientation, see Fig. 6. The motor commands are generated by the CPG model, all joints use "Plateau" pattern that start and end simultaneously. The desired line trajectory is represented by a sequence of points in the Cartesian space (1 [cm] distance between each two successive points). One CPG pattern was employed to move the hand from one point to another. The hand trajectory and the joints angles are presented in "red". We repeated the same experiment while updating the desired hand orientation after reaching each point to match the current hand orientation. The resulting trajectory was significantly improved, hand trajectory and joints angles are represented in "green", see Fig. 6. The end-point errors for the red trajectory are 2 [cm] on Y-axis, 2.5 [cm] on X-axis, 5.5 [cm] on Z-axis. These errors are reduced with the green trajectory being 0.3 [cm] on Y-axis, 0.1 [cm] on X-axis, 1.5 [cm] on Z-axis.

In the third experiment, the robot hand moves with a fixed orientation in a three-dimensional space along a square-shape trajectory composed of 16 points, see Fig. 7. In every step, the end-effector moves 1 [cm] on the given direction in Cartesian space and in every 4 steps it changes the direction. Due to the limited orientation of the hand in the workspace, the performed trajectory is not perfectly matching the desired one.

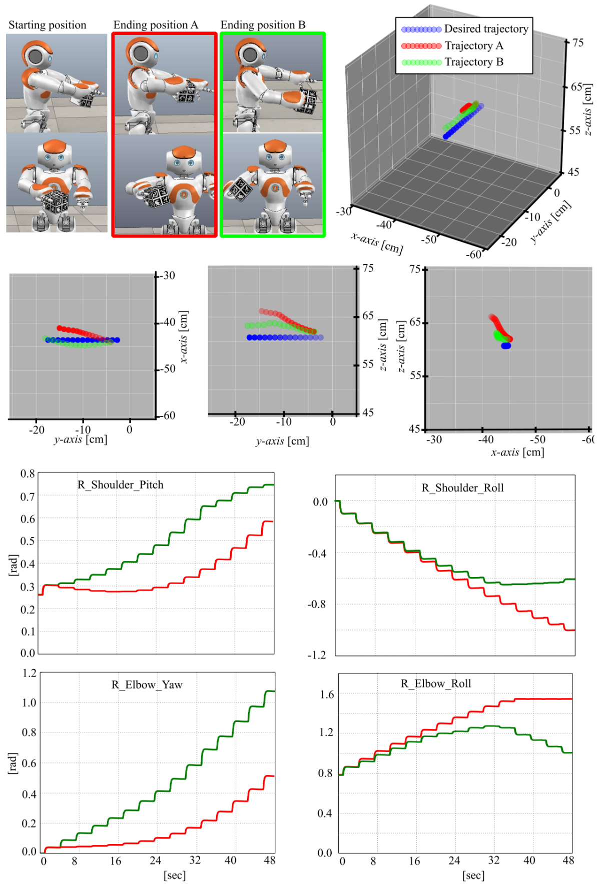

Fig. 6: Two simulation experiments show the end-effector following a desired trajectory "in blue". The arm has to move by 15 [cm] along the negative Y direction. In the first experiment (trajectory A presented in red), the desired end-effector configurations ω_x, ω_y, ω_z, P_x, P_z are constant along the trajectory (we used starting point configurations), only the desired P_y is varying. This results in a larger error compared to the second experiment (trajectory B presented in green), where the desired orientation parameters ω_x, ω_y, ω_z are updated after each action to match the new end-effector orientation. The starting joints' configurations of right arm are as follow: ShoulderPitch 15°, ShoulderRoll 0°, ElbowYaw 0°, ElbowRoll 45°, WristYaw 0°.

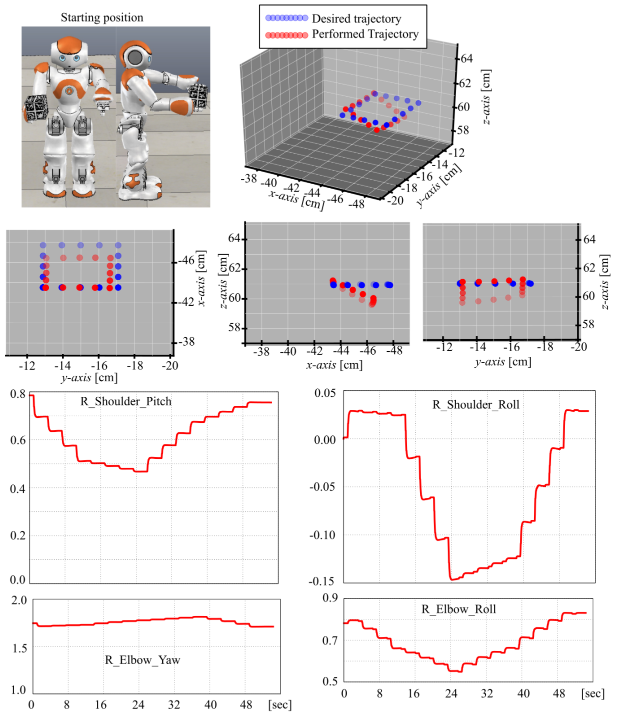

Fig. 7: The end-effector moves along a 4 [cm] × 4 [cm] square shape in X-Y plane (Left → Forward → Right → Backward) with fixed orientation in Cartesian space. The starting configurations of the right arm are ShoulderPitch 45°, ShoulderRoll 0°, ElbowYaw 100°, ElbowRoll 45°, WristYaw 0°.

C. Tactile servoing

The presented reaching experiments show that the neural network model provide the joint angles that move the robot hand to the target position with a deviation error. The usage of a sensory feedback is therefore essential to correct the reaching error in the three-dimensional space. Figure 8 demonstrates the tactile servoing with the MLP model for inverse kinematics. When the hand is placed on the soft gripper, the sensors on the soft gripper will be activated. The center of stimulation is then calculated as in (9). The coordination of this center will be translated into a translation vector in the end-effector space. The translation vector is provided to the inverse kinematics MLP network to find out the corresponding motion pattern that will be generated by the CPG to move the hand in the same direction of the tactile stimulation. Figure 8 demonstrates tactile servoing movement in three directions: up, right and left. It is worth mentioning that if the object moves along the servoing and it is no longer stimulating the hand sensors, another reaching action will be executed based on the new object position with respect to the camera view. In addition, a trajectory planning algorithm cam be added on gthe top of the current grasping model.

D. Grasping a cube

We demonstrate a grasping task of a cube in the 3D space.The cube is within the robot's workspace and has conductive layers covered by ArUco markers at each side. First, the position and the orientation of the cube in the space were detected by the camera. These information are then provided to the neural network which works out the CPG patterns to reach the cube. Once the arm reaches its final position, the tactile feedback is used to reduce the reaching error with a servoing action that follows the center of stimulation. Once, the cube is near the center of the soft gripper (in a range of 1 [cm] from each side), the air pumps will start to inflate into the soft gripper and grasp the object. Figure 9 demonstrates the three phases (reaching, tactile servoing, and grasping) for two experiments.

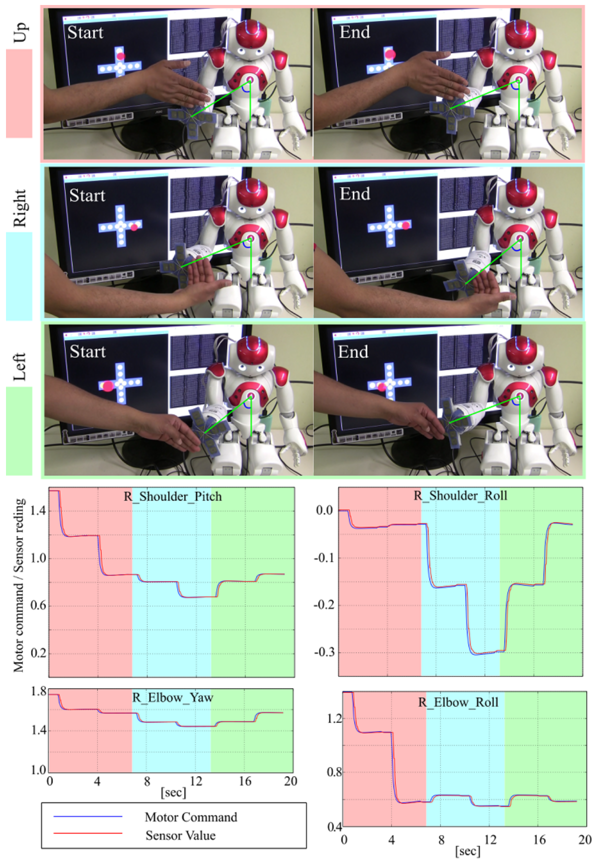

Fig. 8: Three different stimulations for tactile servoing. Stimulations from upside, right side, and left side. Two CPG actions are shown in each servoing.

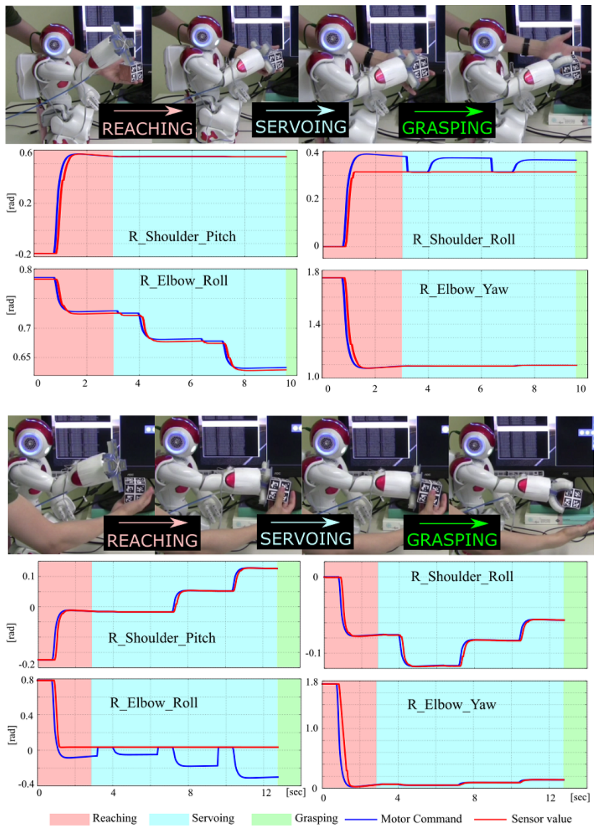

Fig. 9: Two robotic experiments show three phases: reaching, tactile servoing, and grasping. In reaching phase, a single CPG pattern is required at each joint. In servoing phase, a sequence of three patterns per joint is presented.

VI. Conclusion

We presented a framework for grasping in the three-dimensional work space based on the combination of a learned inverse kinematic model and tactile feedback. Discrete CPG patterns are generated and modulated at the joint level. The framework does not require a manual calibration and it works without the robot kinematics configuration. Tactile feedback from a sensory soft hand is used to align the object with the hand before grasping. Several simulations and experiments were carried out on the NAO humanoid robot. The results showed successful reaching and grasping attempts with an acceptable error range. Compared with the traditional inverse kinematics model obtained by mathematical analysis of the robot structure, this model approximated by the neural network has advantages, such as fast computation, independency from the robot dimension, and no requirement for a manual camera calibration. The inverse kinematic model of the arm was learned by a feed-forward neural network through motor babbling of CPG patterns. The sensory soft hand has the advantage of soft interaction, it also compensates the generalization error of the neural network through a tactile servoing phase before grasping.

References

[1] G. Yang and L. Zhao, "Optimal hand-eye calibration of imu and camera," *Chinese Automation Congress (CAC)*, pp. pp. 1023–1028, 2017.

[2] H. L. Qianli Ma and G. S. Chirikjian, "New probabilistic approaches to the ax = xb hand-eye calibration without correspondence," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. pp. 4365–4371, 2016, stockholm.

[3] H. Wu, W. Tizzano, T. Timm, A. Nils, A. Andersen, and O. Ravn, "hand-eye calibration and inverse kinematics of robot arm using neural network," *International Conference on Robot Intelligence Technology and Applications*, pp. pp. 581–591, 2014.

[4] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. pp. 421 – 436, 2017.

[5] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," *IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco*, pp. pp. 365–371, 2011.

[6] J. Nassour, P. Hénaff, F. Benouezdou, and G. Cheng, "Multi-layered multi-pattern cpg for adaptive locomotion of humanoid robots," *Biological cybernetics*, vol. 108, no. 3, pp. 291–303, 2014.

[7] S. Debnath, J. Nassour, and G. Cheng, "Learning diverse motor patterns with a single multi-layered multi-pattern CPG for a humanoid robot," pp. 1016–1021, 2014.

[8] P. Rowat and A. Selverston, "Oscillatory mechanisms in pairs of neurons connected with fast inhibitory synapses," *Journal of computational neuroscience*, vol. 4, no. 2, pp. 103–127, 1997.

[9] J. Nassour, V. Ghadiya, V. Hugel, and F. H. Hamker, "Design of new sensory soft hand: Combining air-pump actuation with superimposed curvature and pressure sensors," pp. 164–169, April 2018.