2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Motor Program Learning for Humanoid Robot Drawing

Deepanshu Makkar, Payam Atoofi, Fred Hamker, and John Nassour

***Abstract*— How do robots generalize the acquired motor representation in the workspace? In this paper, we present a framework that generates motor patterns for drawing arcs in the Cartesian workspace. The basic combinations of patterns resulting in a desired arc has been shown, where the patterns are generated by a Central Pattern Generator (CPG) model. The optimization of those combinations using Genetic Algorithm (GA) and then applying Inverse Distance Weighting (IDW) for generalization in the workspace are further discussed. However, due to the limitations of the aforementioned algorithms in the generalization of motor program, we proposed an approximation function using multilayer perceptron (MLP) to map the features of the trajectory of an arc into a corresponding motor parameters. After learning, we present scenarios, in which a humanoid robot, NAO, draws sketches in a 2D space. Unlike classical methods that use inverse kinematics to draw arcs through connecting several intermediate points in the Cartesian space, our proposed model generalizes the motor features of the pattern generator in the workspace.**

I. Introduction

Robots move by executing a temporal sequence of motor commands that is sent to each joint involved in the action. There are two ways to generate that sequence, the first uses an inverse model to calculate the exact joint positions and velocities for every time step. The second uses a forward representation to build a model-free mapping. The ensemble of joints' variations over time is called motor program. Keele has described motor program as a sequence of muscle commands that generate a desired motion without sensory feedback [1]. However, the motor program needs to be scalable, transferable, and generalizable. Therefore, a simple time-dependent variation will not be sufficient to represent an adaptive motor program. Yet the less parameters the motor program has the easier to be generalized. The Central Pattern Generator (CPG) is one of the examples of motor program [2]. CPG is referred to a set of interneurons located in the spinal cord of vertebrates, these neurons are able to produce patterns without sensory feedback [3]. Different mathematical CPG models have been proposed to control robots' legs and arms [4], [5], [6]. Motor program was also described using so-called Dynamic Movement Primitives (DMPs) [7], [8]. Pastor et al. generalize the grasping motor skill through demonstration [9], a velocity-based inverse kinematic model was used to generate the DMPs. In addition to each joint's motion patterns, motor program includes the coordination of body parts to perform that motion. Spatio-temporal coordination in the motor program determines how patterns at different joints are scaled in the joint space and synchronized in time while keeping other features of the pattern unchanged [10]. The generalization of motor program through its coordination parameters is a challenging

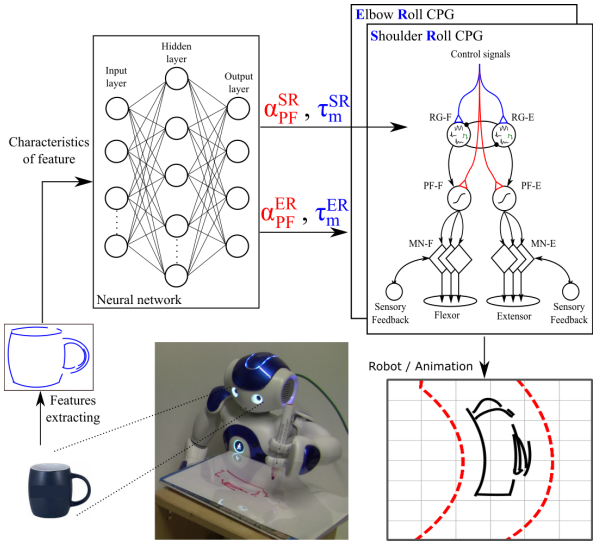

Fig. 1: Block Diagram for Sketch Drawing. The schematic illustration encapsulates the framework, where the image would undergo an image processing routine and its features would be extracted using line and arc segment detector, from which the elliptical- and circular arcs would be selected and transformed from image space to workspace. The features of the arc would then be sent as an input to a trained neural network (multilayer perceptron), whose output provides the parameters of a motor program. The motor primitives are generated by a CPG model (ML-MP CPG), and the performed action would ideally result in a trajectory similar to the desired arc. The output of the neural network would be considered as the control signal that manipulates the time constant in RG layer (in blue) and the slope of the sigmoid in PF layer (in red). `https://youtu.be/fC7GGPRaXO4`

problem in robotics. In this paper, we address the generalization of motor program in humanoid robot drawing, see Fig. 1. The concept of motor program shows an advantage in generalization as opposed to traditional methods used to solve robot drawing task, e.g., model-based algorithms. Calinon et al. proposed a framework for humanoid robot capable of sketching [11]. In their approach, the trajectory planning was based on inverse kinematics. Singh et al. has also implemented an inverse kinematics approach for sketch drawing [12]. Bertram et al. has applied Rapidly-exploring Random Trees (RRTs), where the robot's joint configuration needs to be fed to RRTs [13]. The output is the inverse kinematics configuration. This method would not find the best solution rather it finds different paths to reach the

end effector, hence it can have multiple solutions which can be computed in a fixed time frame. Although model-based approaches are more accurate than model-free ones, the executed motor movement cannot be used in a new context. Yet in the model-free approaches the data needs to be gathered by robot experiments or demonstrations, e.g. imitation learning. Tan et al. have applied imitation learning to make the robot learn how to write numbers (0 to 9), whose motor primitives were generated based on Dynamic Movement Primitives (DMPs) [14]. Probabilistic methods such as Gaussian Mixture Regression and dimensionality reduction technique can also be together used to generalize the robot's task trajectory acquired by demonstration [15]. Atoofi et al. have used Genetic Algorithm to optimize the action of drawing straight lines, generated by a CPG model [24]. Furthermore, applying Inverse Distance Weighting (IDW) has improved the estimation of the motor program parameters provided by a self-organizing map (SOM).

To address the generalization of motor program in robot drawing, we first present in Sec. II a framework for pattern generation which was used previously to control robot's legs and arms for rhythmic and discrete motions [6], [16]. Section III addresses the generalization of motor program using different algorithms. A multilayer perceptron (MLP) has been used to create a neural network representation of motor program for drawing arcs in a 2D space. In Section IV, it is shown how the trained neural network (MLP) has been used on a simulation and a real robot in order for the robot to draw a sketch of the observed image comprising of arcs. The image processing aspects of the project is also discussed. Conclusion is provided in Section V.

II. Motor Program

As discussed in Sec. I, motor program is formed before an action takes place, where its sequence can be stored in memory, and later can be executed without sensory feedback. This sequence of movement is also called motor primitives, suggested by Tamar [17], which are the building blocks in the motor hierarchy. By employing appropriate transformations and operations on the motor primitives, a variety of different movements can be derived, and stored in the motor repertoire; and further be used to make more complex patterns. Repetition and practice will correct the motor program so that an action becomes more accurate, which ultimately creates a more reliable motor repertoire consisting of several motor programs [9]. The motor repertoire with its stored movements would be used later to perform skills that might require a combination of movements. Since CPGs are considered to follow the definition of motor program, in this study the motor primitives are generated with a computational model of CPG based on [18]: Multi-Layered Multi-Pattern CPG (ML-MP CPG). The CPG has been used in two joints of an arm in a humanoid robot, NAO, where the joints are selected in a way that the motion resembles a two link planar arm. The CPG model is able to generate both rhythmic and non-rhythmic patterns, which creates the possibility to have a wide range of motor primitives. A brief insight to the model is given hereinafter.

A. Motor Primitives Generation and Pattern Generation Model

The architecture of the CPG model is based on the separation of timing and activation of motor cycles, influenced by [19] for two level CPG. The applied CPG model consists of three layers of neurons: Rhythm Generation Layer (RG), Pattern Formation Layer (PF), and Motor Neurons (MN).

Rhythm Generation Layer: RG neurons are responsible for the generation of various motion patterns, e.g. Quiescent, Oscillatory, Plateau, etc. [21], see Fig. 1. The RG neuron model is represented by two differential equations (1) and (2):

$$\tau_m . \frac{dU}{dt} = -(U - A_f tanh((\sigma_f / A_f)U) + q - i_{inj}), \quad (1)$$

$$\tau_s . \frac{dq}{dt} = -q + \sigma_s (U - E_s), \quad (2)$$

where U is the membrane potential, q is the lumped slow current, τ_m is the membrane time constant, τ_s is the slow current's time constant ($\tau_m < \tau_s$). σ_s and σ_f are the potassium and the calcium conductances, respectively. A_f is the width of the N-shape of the current-voltage curve. A pattern is triggered by an injected current i_{inj} to the RG neuron.

Pattern Formation Layer: The neurons in the PF layer shape the patterns generated in RG layer for both flexion and extension [22]. PF neurons are modeled as sigmoid activation functions:

$$U_{\mathrm{PF}} = \frac{1}{1 + e^{\alpha_{\mathrm{PF}}(\theta_{\mathrm{PF}} - (w_{\mathrm{RG}\to\mathrm{PF}} \cdot U_{\mathrm{RG}}))}}, \quad (3)$$

where U_{PF} is the output of the PF neuron, and U_{RG} is the output of the RG layer given by (1) and (2), which is influenced by the strength of the connection between the two consecutive layers with the weight $w_{\mathrm{RG}\to\mathrm{PF}}$. α_{PF} and θ_{PF} are the slope and threshold of the sigmoid activation function, respectively.

Motor Neurons Layer: The Motor Neurons (MN) perform the flexion and the extension movements of the joint, where it receives signals from PF layer and also sensor neurons. The neurons in this layer are also modeled similar to neurons in PF layer as sigmoid activation functions:

$$U_{\mathrm{MN}} = \frac{1}{1 + e^{\alpha_{\mathrm{MN}}(\theta_{\mathrm{MN}} - (w_{\mathrm{PF}\to\mathrm{MN}} \cdot U_{\mathrm{PF}} + w_{\mathrm{SN}\to\mathrm{MN}} \cdot S))}}, \quad (4)$$

where U_{MN} is the output of the MN neuron, and U_{PF} is the output of the PF layer given by (3), and the weight $w_{\mathrm{PF}\to\mathrm{MN}}$ represents the strength in connection between these two layers. Similarly, α_{MN} and θ_{MN} are the slope and threshold of the sigmoid activation function, respectively. S is the value of the sensory feedback provided from sensory neurons, which is 0 in our case, and the strength between the current layer neurons and sensory neuron is shown by the weight $w_{\mathrm{SN}\to\mathrm{PF}}$. Therefore, the outputs of each layer only alter

with the signals from the previous layer plus the high-level controller that modulates their parameters, e.g. slope and threshold. Particularly the α_{MN}, θ_{MN} and θ_{PF} were fixed and remained unaltered throughout the whole study, whereas the α_{PF} was assumed to be modulated from the high-level controller, hence the ability to change the amplitude of the signal generated by RG layer.

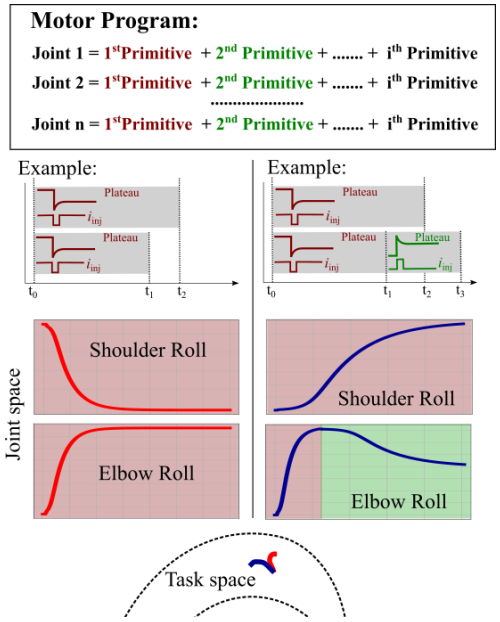

Fig. 2: Each motor program consists of a variety of motor primitives of the involved joints. Arcs are drawn using only plateau patterns. Although the patterns remain the same in their nature, but RG parameters could change the speed of the non-rhythmic patterns, as well as the phase and direction of each pattern, providing the opportunities to have signals sent to each joint at different time with an arbitrary delay. Furthermore, in PF for instance, the amplitude of each pattern can be adjusted. These blocks of patterns, with different timing and activation would provide us with the possibility to have a vast range of actions. Here an example of single primitive per joint (left) is provided where the different speed in the patterns in each joint has led to a trajectory of an arc, whereas a slightly more complex scenario (right) of having two successive primitives at the Elbow could provide a different arc.

B. Selection of Motor Primitive

The designed task as mentioned earlier is for a humanoid robot to be able to draw sketches consisting of arcs. Two joints in the left arm (Shoulder Roll and Elbow Roll) of a humanoid robot, NAO, would guarantee a two link planar motion. Each joint with its own CPG is capable of generating variety of patterns each with different amplitude, speed in non-rhythmic and frequency in rhythmic patterns, phase and direction. Any combination of these patterns could lead to a new action. The task of finding which combination of patterns could result in the desired action can be a rather complex task even in a manipulator with two degrees of freedom. However, for drawing arcs only plateau patterns were chosen to be in the motor program. Drawing desired arcs is not necessarily possible with only one pattern for each joint as their motor primitives, see Fig. 2 (left). Depending on the starting point of the arc and its other features a more complicated motor program is needed, e.g. one pattern in Shoulder Roll and two patterns in Elbow Roll as their motor primitives, see Fig. 2 (right). The exploration of the whole parameter space to obtain the best motor program for a desired action, whether it is just the RG parameters and their combinations or RG parameters and PF parameters together, is not feasible due to the immensity of the dimension of the available parameters, especially if more than one pattern is generated at each joint. The best approach is to limit the number of modifiable parameters in each layer and also limit the possible combinations of generated patterns at each joint. Hence, in this work only the slope of sigmoid in PF layer, α_{PF}, the direction of the injected current, $i_{\mathrm{inj}} = \{-1, +1\}$, and time constant, τ_m, in RG layer have been subjected to changes; however in case of having more than one pattern for a joint, the delay between the first generated pattern and the second forthcoming pattern needs to be taken into account.

III. Motor Program Generalization

The task regarding drawing arcs in the reachable workspace of the left arm of NAO, is not only a multi-objective task, in that the features of an arc are represented as a vector, but also could consist of combination of motor primitives for each joint, hence more parameters available for control. The first step toward evaluation of an action, to measure its success or failure, is to define an error function. The error function allows us to evaluate whether the result of an action is close to the desired action. In a sketch drawing task, humans would not consider the mathematical definition of an "elliptical" arc, rather only the resemblance between the resulted trajectory and an arc. Yet to define an error function as close as possible to our understanding of how a human evaluates an arc in a sketch drawing task, two different approaches have been applied. The simplest method to measure the difference between a trajectory resulted from an action and the desired trajectory (in this case a desired arc) is to have the distance of each point on the desired arc to its corresponding point on the drawn trajectory. However, such approach undermines many traits of an arc, ergo a vector of features of an arc is introduced to be measured and compared with that of a desired arc, i.e. the sum of the distances between the points on two trajectories (desired and obtained) cannot properly reflect the nature of an arc. Hence the vector of features regarding a trajectory could be represented as following:

$$v_{\mathrm{arc}} = \left[x_0, y_0, x_{\mathrm{e}}, x_{\mathrm{e}}, l, d_{\mathrm{p}}, \delta_{\mathrm{p}}, \beta\right]^T, \tag{5}$$

where x_0, y_0, x_{e} and y_{e} are the $x-y$ Cartesian coordinates of the starting point and end point of a trajectory, respectively. The length of the trajectory is given by l. The value of d_{p} provides the distance between the peak of a trajectory

and the line connecting the starting and end point of said trajectory, whose sign would determine whether the peak is below the connecting line or above it, and it was obtained by the assumption of trajectories rotating counter-clockwise, i.e. a trajectory rotated around its starting point would always have the same d_{p}. The counterpart of d_{p}, δ_{p}, is another value, whose absolute is the distance from the midpoint of the line connecting start and end of a trajectory to the projection of the peak on the same line, and its sign would determine if the projection is closer to the start point or to the end point of the trajectory. β is the angle of the line connecting starting and end point of a trajectory. Regardless of the measurement technique to evaluate the performance of an action, to improve an action new set of parameters of a motor program needs to be acquired that reduces the error calculated between the desired- and obtained trajectory. [**?**]

A. Motor Program Optimization

One way to find the correct motor program and its corresponding parameters is through optimization, by trying to reduce the error generated from the trajectories to reach the action that would result in the desired trajectory (desired arc). For simplicity it was assumed that the error is calculated based on the distances of each pair of points between the two trajectories (desired and obtained). The first stage of optimization was done on motor programs with single primitives per joint. The parameters involved are the slope of the sigmoid in Pattern Formation layer, the time constant and the direction of injected current in Rhythm Generation layer, in both Shoulder Roll and Elbow Roll, i.e. $\alpha_{\mathrm{PF}}^{\mathrm{SR}}$, τ_m^{SR}, $i_{\mathrm{inj}}^{\mathrm{SR}}$, $\alpha_{\mathrm{PF}}^{\mathrm{ER}}$, τ_m^{ER} and $i_{\mathrm{inj}}^{\mathrm{ER}}$. Since the direction of i_{inj} can be combined with the amplitude of the signal to create one general parameter that encapsulates both sign and amplitude, the α_{PF} and i_{inj} will be represented as one parameter that can be both positive and negative, ergo the number of parameters in optimization is four, two for each joint: α_{PF} and τ_m. In addition to the selection of parameters in optimization and the error function, the optimization method should also be suitable for the assigned task. Knowing that the derivative of evaluation function is not available, the Genetic Algorithm (GA) has been chosen as it is a derivative-free optimization technique. Although the time for injected current for both motor primitives (one for each joint) are the same, due to the difference in values of τ_m^{SR} and τ_m^{ER}, both joints start at the same time but may finish at different times, i.e. different speeds. The results of the GA optimization for single primitive per joint are shown in Fig. 3 (left). The fitness function as explained previously is the sum of the Euclidean distances between each point on the desired trajectory and its corresponding point on the obtained trajectory, see (6). All the points on each trajectory are interpolated, using B-spline [23], to guarantee that the points in the vectors representing the trajectories are equally distributed along their trajectory. In (6), $\mathbf{p}_i^{\mathrm{desired}}$ and $\mathbf{p}_i^{\mathrm{obtained}}$ are the points on the desired- and obtained trajectory, respectively. The first population of GA (initialization) is selected via random values for α_{PF} and τ_m. The selection is done by tournament selection, and polynomial mutation is chosen for mutation process.

$$\sum_{i=0}^{n} \|\mathbf{p}_i^{\mathrm{desired}} - \mathbf{p}_i^{\mathrm{obtained}}\| \tag{6}$$

As it can be seen for certain arcs, the GA cannot provide desired results, e.g. the arcs that are on or close to the borders of the regions where the direction of the i_{inj} is changing, see Fig. 3 (left). Therefore, a more complicated motor program is introduced. Elbow roll joint will have a sequence of two CPG patterns, while we keep using a single pattern for shoulder roll joint, an example is shown in Fig. 2 (right). In such case, the delay between two consecutive signals also plays a role, however for simplicity we assume that the second motor primitive starts when the first motor primitive ends, since all motor primitives generate plateau patterns. Hence the number of parameters in optimization would increase by only two: α_{PF} and τ_m for the new motor primitive in Elbow Roll. The results of optimization for those particular cases of having multiple-motor primitives are shown in Fig. 3 (right).

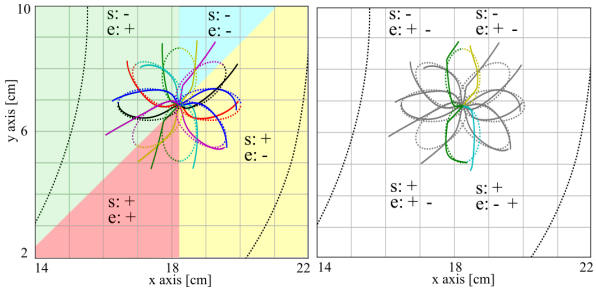

Fig. 3: Optimization of 16 arcs with genetic algorithm using a single primitive per joint (left), and using two primitives for the elbow "e" and one for the shoulder "s" (right).

B. Neural Network Representation of Motor Program

To generalize the obtained motor programs over the workspace (e.g. drawing the same arc at a different starting position) or to use them to develop a motor program for a new arc at the same starting position (e.g. arcs rotated around their starting positions), a linear mapping is not sufficient, as the quality/nature of the motor primitives' parameters are different. The use of Inverse Distance Weighting (influenced by [24]) also has not led to a successful result, because the IDW could not be applied on a multi-objective task, as well as the simplicity of Euclidean distance as the sole error function would mask the other features of the drawn trajectory as an arc. To generalize motor programs and have a representation of workspace, an MLP (multilayer perceptron) neural network has been employed, with 8 input neurons, 4 output neurons, and 3 hidden layers, each layer with 100 neurons. The input vector is given in (5), and the output vector is a vector of motor primitives for both joints, see (7), as it can be seen in Fig. 4.

$$v_{\mathrm{output}} = \left[\alpha_{\mathrm{PF}}^{\mathrm{SR}}, \tau_m^{\mathrm{SR}}, \alpha_{\mathrm{PF}}^{\mathrm{ER}}, \tau_m^{\mathrm{ER}}\right]^T \tag{7}$$

The training data is collected by randomly selecting starting positions on the reachable workspace for 300 different points and for each point to assign 300 random sets of motor primitives which resulted in random trajectories, provided the random motor primitives are all essentially plateau patterns, i.e. random selection of $\alpha_{\mathrm{PF}} \in [-0.06, 0.06]$ and $\tau_m \in [0.01, 1.0]$ for each joint (SR and ER). The significant features of each of these random trajectories would be calculated and stored as a vector, (5). "Sigmoid" was used as an activation function and "Adam" optimiser is used for optimising the loss with α value to 0.06. After training with a data set of size 90,000, the small difference between the training loss by 0.2865 and the validation loss by 0.2987 ensured that the model is not overfitting.

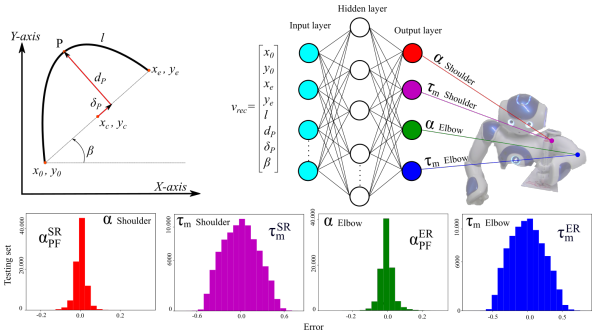

Fig. 4: MLP neural network with the illustration of the input vector. The network has 4 outputs, 2 for shoulder roll joint, and 2 for elbow roll joint (top). Histograms of the error for the output neurons are presented (bottom).

IV. Experimental Results

The MLP network has been tested on both simulation and real robot. In both cases an image processing routine has been applied in order to extract necessary information and also to transform the data in image space into Cartesian space.

A. Image Processing

The collected image, for instance from NAO's camera, is sent to a feature extraction routine, where the lines and elliptical arc segments are found. The method of use is proposed by [25], which is based on the *a contrario* approach and can be seen as an extension to Line Segment Detector [26]. Due to the design of the task, only the elliptical- and circular arcs are of importance. The next step is to transform the data such as starting and end point of an arc on the image space to the Cartesian space, where a simple projective transformation has been applied to map the frame of the image (rectangle) to a rectangle inside the reachable workspace. Having chosen the four corners of the rectangle in the workspace, any point can be then transformed from the image to lie inside the rectangular area in the workspace. Finally, the vector of arc features will be calculated and would be given to the MLP, which would then provide the motor program parameters for each segment detected on the image, see Fig. 4.

B. Results on Simulation and Real Robot

Figure 5 shows the results of the network in simulation, where the first column (from left) shows the images, the second depicts the results of feature extraction from the given image, third column shows the transformed segments extracted from images into Cartesian space, where the four corners of the rectangular area of workspace are $(161, 48), (161, 166), (206, 166), (206, 48)$. The last column shows the results of the actions of the motor programs, whose parameters are provided by the MLP. On the real robot, the same steps from feature extraction to mapping the features on the Cartesian space has been applied, and the results of the actions of the provided motor program from MLP to generate arcs with given features are shown in Fig. 6.

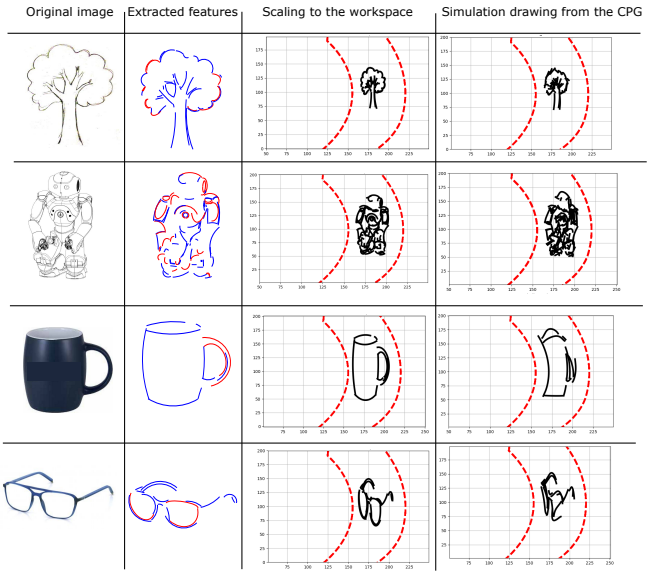

Fig. 5: Drawing in Simulation

Fig. 6: Drawing with Real Robot

In Fig. 5 four different images are provided: a tree, NAO robot, a mug, and a pair of glasses. The feature extraction routine has detected 45 arcs for the tree, 98 arcs for NAO robot, 11 arcs for the mug, and 18 for the glasses. The feature vector for each of the arcs in each of the images has been created and passed to the MLP. After that the MLP has provided a set of parameters for each arc, the result of the performed action is then evaluated by a new vector of features for the obtained trajectory. To measure the accuracy of the results of the action, the difference between the desired vector of features and the obtained vector is calculated by:

$$Error = \frac{\|\hat{v}_{\mathrm{arc}}^{\mathrm{Obtained}} - \hat{v}_{\mathrm{arc}}^{\mathrm{Desired}}\|}{\|\hat{v}_{\mathrm{arc}}^{\mathrm{Desired}}\|}, \tag{8}$$

where $\hat{v}_{\mathrm{arc}}$ is a vector of normalized feature elements, each of which is normalized by the maximum and minimum

values of the collected data used for training the MLP. Hence, in the performance to draw the tree, the average error of 0.186, equivalently the accuracy of 81.4%, has been obtained, with the minimum error of 0.0107 (accuracy 98.9%) and maximum error of 0.9905 (accuracy 0.9%). As for the robot picture, the average error of 0.132 (accuracy 86.8%), with the minimum error of 0.0048 (accuracy 99.5%) and maximum error of 0.9795 (accuracy 2.05%), has been obtained. The performance to draw the mug and the glasses produced the average error of 0.150 (accuracy 85%), with the minimum error of 0.015 (accuracy 98.5%) and maximum error of 0.724 (accuracy 27.65%), and 0.109 (accuracy 89.1%), with the minimum error of 0.0117 (accuracy 98.8%) and maximum error of 0.9765 (accuracy 2.07%), respectively.

V. Conclusion

The combinations of motor primitives are essential to perform complex actions. This paper discusses the difficulty of finding the proper parameters for these primitives and the generalization of the motor program. A CPG model is used to generate the motions and each CPG pattern generated in a joint is considered as a single primitive. An MLP neural network has been implemented, whose inputs are the features of a trajectory as an arc, and the coordination parameters as its output. The idea of generalization has always been a challenge in robotics. One of the problem arose from the large number of parameters is the curse of dimensionality, which always demands methods for dimensionality reduction to reduce the size of the search space. There are empirical experiences and innovative ways, which would help to further limit the number of primitive parameters that are involved and contribute to the varying patterns as variables. For instance, the spatio-temporal coordination parameter that combines both the direction of injected current and the slope of the sigmoid in pattern formation layer, represented as one variable. This could be achieved by using a multi-layered CPG model, another advantageous property that DMPs do not possess. As a result one could easily control the spatio-temporal coordination without affecting the nature of the patterns. This work is a stepping stone toward a more sophisticated framework where a robot can draw a sketch comprising of simple and complicated shapes, e.g. lines, arcs, etc. Our future work will involve more complex patterns to perform a better matching of the desired end-effector trajectory.

References

[1] S. W. Keele, "Movement control in skilled motor performance." *Psychological bulletin*, vol. 70, no. 6p1, p. 387, 1968.

[2] M. E. Morris, J. J. Summers, T. A. Matyas, and R. Iansek, "Current status of the motor program," *Physical therapy*, vol. 74, no. 8, pp. 738–748, 1994.

[3] T. G. Brown, "On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system," *The Journal of physiology*, vol. 48, no. 1, pp. 18–46, 1914.

[4] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[5] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert, "Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives," *Autonomous Robots*, vol. 31, no. 2-3, pp. 155–181, 2011.

[6] S. Debnath, J. Nassour, and G. Cheng, "Learning diverse motor patterns with a single multi-layered multi-pattern cpg for a humanoid robot," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 1016–1021.

[7] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robots," in *The International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, Feb 2013.

[9] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 763–768.

[10] A. P. Georgopoulos, "Cognitive motor control: spatial and temporal aspects," *Current Opinion in Neurobiology*, vol. 12, no. 6, pp. 678 – 683, 2002.

[11] S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, Dec 2005, pp. 161–166.

[12] A. K. Singh, P. Chakraborty, and G. Nandi, "Sketch drawing by nao humanoid robot," in *TENCON 2015-2015 IEEE Region 10 Conference*. IEEE, 2015, pp. 1–6.

[13] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1874–1879.

[14] Q. D. Huan Tan and N. Wu, "Robots learn writing," *Journal of Robotics*, vol. vol. 2012, p. 15 pages, 2012.

[15] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[16] J. Nassour, P. Hénaff, F. B. Ouezdou, and G. Cheng, "A study of adaptive locomotive behaviors of a biped robot: patterns generation and classification," in *Proceedings of the 11th international conference on Simulation of adaptive behavior: from animals to animats*, ser. SAB'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 313–324.

[17] Z. Pan, Y. LishanKang, G. Liu *et al.*, "Parameter estimation by genetic algorithms for nonlinear regression," *High Technology*, vol. 946, p. 953, 1995.

[18] 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.

[19] I. A. Rybak, N. A. Shevtsova, M. Lafreniere-Roula, and D. A. McCrea, "Modelling spinal circuitry involved in locomotor pattern generation: insights from deletions during fictive locomotion." *The Journal of Physiology*, vol. 577, pp. 617–639, 2006.

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

[21] P. Rowat and A. Selverston, "Learning algorithms for oscillatory networks with gap junctions and membrane currents," *Network: Computation in Neural Systems*, vol. 2, no. 1, pp. 17–41, 1991.

[22] D. A. McCrea and I. A. Rybak, "Organization of mammalian locomotor rhythm and pattern generation," *Brain research reviews*, vol. 57, no. 1, pp. 134–46, January 2008.

[23] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[24] P. Atoofi, F. H. Hamker, and J. Nassour, "Learning of central pattern generator coordination in robot drawing," *Frontiers in Neurorobotics*, vol. 12, p. 44, 2018.

[25] V. Pătrăucean, P. Gurdjos, and R. G. Von Gioi, "A parameterless line segment and elliptical arc detector with enhanced ellipse fitting," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 572–585.

[26] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: a line segment detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.