Proceedings of the 10th RSI International Conference on Robotics and Mechatronics (ICRoM 2022), Nov. 15-18, 2022, Tehran, Iran

Application of Artificial Neural Networks in Automatic Optimum Trajectory Selection for the Hitting Task of a Ping Pong Robot

2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM) | 978-1-6654-5452-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRoM57054.2022.10025145

Saleh Farsi
Mechanical Engineering Department, Center of Advance Rehabilitation and Robotic Research (FUM-CARE) Ferdowsi University of Mashhad
Mashhad, Iran
salehfarsi@mail.um.ac.ir

Saleh Emami
Mechanical Engineering Department, Center of Advance Rehabilitation and Robotic Research (FUM-CARE) Ferdowsi University of Mashhad
Mashhad, Iran
saleh.emamipour@mail.um.ac.ir

Erfan Koochakzadeh
Mechanical Engineering Department, Center of Advance Rehabilitation and Robotic Research (FUM-CARE) Ferdowsi University of Mashhad
Mashhad, Iran
erfan.koochakzadehdandansaz@mail.um.ac.ir

Iman Kardan
Mechanical Engineering Department, Center of Advance Rehabilitation and Robotic Research (FUM-CARE) Ferdowsi University of Mashhad
Mashhad, Iran
iman.kardan@um.ac.ir

Amirhossein Nayebiastaneh
ERAM 3D Scanning Technologies,
Mashhad, Iran
am.nayebi@gmail.com

Alireza Akbarzadeh
Mechanical Engineering Department, Center of Advance Rehabilitation and Robotic Research (FUM-CARE) Ferdowsi University of Mashhad
Mashhad, Iran
ali_akbarzadeh@um.ac.ir

***Abstract*— As the great setups for the implementation of human-competitive artificial intelligence algorithms, ping pong robots have found an increasing attention in the literature. One of the multiple factors that affects the performance of these robots, is the paddle trajectory, travelled to hit the ball. A good hitting task trajectory should be accurate and fast enough to ensure that the ping pong paddle will hit the ball at the correct position/orientation, in the correct time, and with the correct speed. Moreover, considering the limitations in the speed and torque of the robot's actuators, the generated trajectory should be implementable by the robot. In this paper, three of most popular path generation curves, namely Spline, Bézier, and Dubins curves, are used to generate the hitting task trajectory for different hitting conditions. The generated trajectories are then compared based on a multi-objective cost function that includes the maximum values of the jerk, acceleration, and torque imposed to the robot's actuators. Finally, an artificial neural network is used to learn the best trajectory for each hitting condition. The neural network takes the hitting condition (three-dimensional hitting position and velocity) as the input and selects the best trajectory at the output. The results show that the trained neural network selects the true trajectory with an 85% success rate for the test data. The proposed method enables the ping pong robots to instantly select the optimum trajectory for each hitting condition, with no need to time-consuming optimization procedures.**

Keywords—Hitting trajectory, neural network, ping-pong robot, trajectory optimization, optimal trajectory

I. INTRODUCTION

In recent years, artificial intelligence (AI) has found increasing applications in diverse areas. One of the most common applications is to develop algorithms and devices that act like a human and have the ability to compete with human opponents. AI has shown promising results in developing powerful algorithms to defeat human rivals in computer games like chess [1] and Go [2]. However, physical competition with human opponents is a much more complicated task. As a good case study for this case, ping pong player robots have found growing attention by the researchers [3-16]. Design and implementation of Ping-pong robots has started in the 1980s. As described in [1], one of the first ping-pong-playing robots was built using a PUMA 260 robot.

For a robot to play ping pong there are different components to be implemented such as control system, ball trajectory prediction, hitting policy, hitting trajectory and etc. A visual control system for a five DOF table tennis robot is developed in [4]. In [5-7], some algorithms are developed to return the incoming ball to a certain landing position. The hitting policy is studied in [8,9]. In [8] particle swarm optimization is used to decide when and how a robot should strike the incoming ball. [9] proposes a combined learning framework consisting of the maps for ball trajectory prediction and robot trajectory generation. Algorithms for predicting the ball trajectory are studied in [10], and [11].

Each trajectory requires a specific amount of torque, acceleration and jerk to be provided by the actuators to move the robot. Because of limitations in the structure of the robots and the output speed/torque of the actuators, it is important to select the trajectories that require the optimum value of these parameters. Due to the nature of the ping pong game, high-speed trajectories are required for the robot. These trajectories require a critical amount of the mentioned kinematic and dynamic parameters to hit the ball in the right position/direction at the right time. Consequently, the trajectory of the paddle is a significant factor that affects the performance of a ping pong robot.

There are some previously works focusing on trajectory generation for a robot to play ping pong. In these works, multiple approaches are used to generate a trajectory in the best way regarding different parameters. For example, in [12] the lower-level control system uses fifth-order polynomials to move the paddle's center to the proper position and hit the ball.

The authors in [14] and [15] introduce a new framework for generating robot trajectories without a fixed hitting plane.

Due to importance of paddle trajectory generation, in this paper we have put our focus on generating the optimum hitting task trajectory. In this work, we consider three common curves with low computational cost in real-time applications, namely Spline, Bézier, and Dubins curves [16-18]. Unique time trajectories are optimized and implemented on each curve. Then, an artificial neural network is used for automatically selection of the best hitting trajectory type, among these trio of curves.

The rest of this paper is organized as follows: In section II, the conditions of the ping pong game and the setup and processes are explained. In section III, three types of trajectories are introduced, which leads to description of optimal trajectory selection strategy in section IV. Results are shown and discussed in section V. Finally, section VI concludes the paper.

II. The Overall System and Processes

Ping-Pong game includes a standard 2.7*1.525 [m^2] table with a height of 0.76 [m]. A net of 0.15 [m] height is located in the middle of the table as shown in Fig. 1. The game also includes two ping-pong paddles (one for each player) and a ball with a radius of 0.02 [m] a mass of 2.7 grams. The average velocity of the ping-pong ball has been determined to be 25 mph which is 8 [m/s] [19]. If we consider the average distance between the opponent and the robot to be equal to the table length (2.7 [m]), then the total time for the robot response is 0.3375 seconds. Considering some dead time for the vision processing and other calculations, there will be only 0.2 [s] for the robot to move. Therefore, very highspeed trajectories are required for the ping pong robots.

In this work, a 5-DOF robot based on delta parallel robot is considered for the ping pong game. The moving platform of the robot moves in $X, Y,$ and Z directions, while the attached serial links provide the yaw and roll rotations for the paddle. Two stereo full HD cameras with a frame rate of 120 [fps] record two-megapixel images for the calculation of the position/velocity of the ball. There is also a high-performance computer that implements the algorithms for ball trajectory prediction, trajectory planning, and the robot control.

The overall process of the robotic Ping-Pong game is shown in Fig. 2 and explained below.

1) Vision System: The two stereo cameras are used in the vision system to determine the 3D position and velocity of the ball after hitting by the opponent. This block sends the results to the ball trajectory prediction block.

2) Ball trajectory prediction: Receiving the initial position and velocity of the ball, the implemented algorithm for ball trajectory prediction starts its calculations. This block predicts the velocity and trajectory of the ball for its entire motion, i.e. the flying motion in the air and the inelastic collision with the table.

3) Hitting Policy: This block receives the predicted ball trajectory and defines the appropriate position that the paddle should hit the ball. Moreover, the orientation of the paddle and its linear/angular velocity at the hiiting point, are defined in the section. Hitting the ball to make a powerful, accurate, and hard-to-answer shot, is a very complicated task. Therefore, this section may include some artificial intelligence methods, like reinforcement learning.

4) Paddle Trajectory Generation: This block communicates with the hitting policy unit to receive the hitting information, including the hitting position/orientation, hitting velocity, and hitting time. This information is used to generate an optimized hitting trajectory for the robot, with the lowest possible dynamic requirements. This part is in the main focus of this paper.

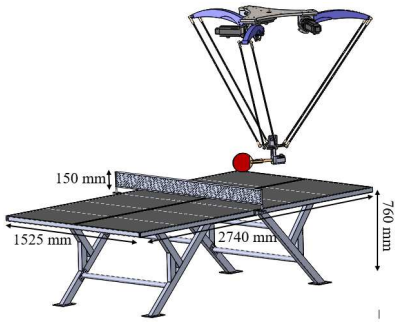

Fig. 1. Configuration of the robot with ping-pong table

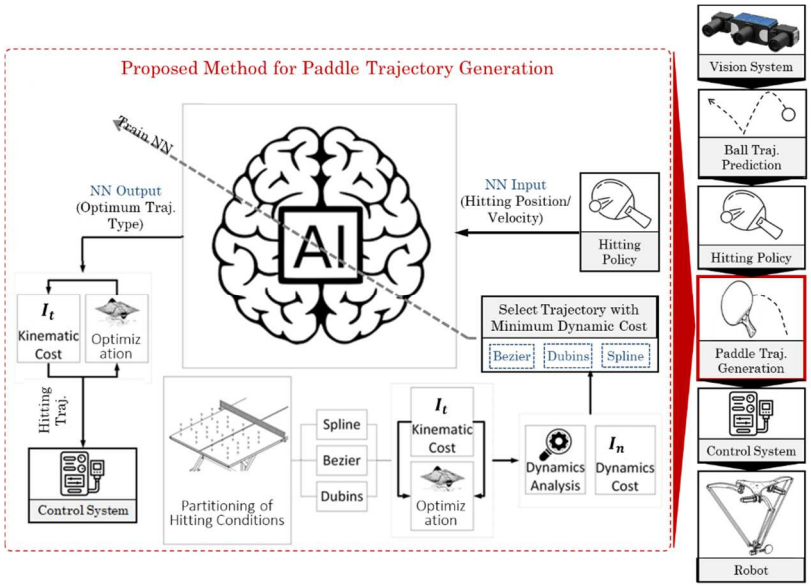

Fig. 2. The overal playing algorithm for ping pong robot

5) Control System: Finally, the generated hitting trajectory is sent to the control system to produce appropriate control commands and send them to the servo drivers to move the robot, accordingly.

III. Moving Platform Trajectory Generation

Although the paddle trajectory has 5 DOF, including 3 Cartesian and 2 rotational DOF, in this work, only the 3 Cartesian DOF of the moving platform are considered. The reason of why more attention must be paid to the Cartesian DOF, is that the rotational displacements are generally smaller than linear displacements of the paddle. Furthermore, the paddle rotational and translational DOF may be studied almost independently. For each hitting position/orientation, there will be a unique position for the moving platform. The first three motors are responsible for putting the moving platform in the calculated position/speed, while the two last motors should position/orient the paddle. The motion of the two last motors may be considered separately as single DOF motions.

As the hitting task needs high-speed movements, the selected curves should have low computational costs [20]. Therefore, amongst different curves that may be used for the hitting trajectory, this paper focuses on three specific curves, namely Bezier, Spline and Dubins curves [20].

A. Cubic Spline

A spline [21], [22] is introduced locally as a very simple, yet globally flexible and smooth polynomial function. It is often preferred to use spline interpolation [20] rather than polynomial interpolation in interpolating problems because it gives similar results, even for low-degree polynomials. As the result, cubic spline is used in this study.

As shown in Fig. 3, let us consider $n+1$ nodes between a and b as $a = x0 < x1 < \ldots < xn = b$ and also the corresponding evaluations of a given function $f(x)$ as $f_i, i = 0, \ldots, n$. The goal is to find a cubic spline curve that approximate the function $f(x)$, in an accurate and smooth way. Since the spline is of 3rd degree, the derivative of its second order are also continuous. By introducing the following notation $f_i = s_3(x_i), m_i = s_3'(x_i), M_i = s_3''(x_i), i = 0, \ldots, n$, the cubic spline interpolation is given by,

$$s_{3,i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + \frac{M_i(x - x_{i-1})^3}{6h_i} + C_{i-1}(x - x_{i-1}) + \tilde{C}_{i-1} \quad (1)$$

where, $h_i = x_i - x_{i-1}, i = 0, \ldots, n$ and,

$$C_{i-1} = f_{i-1} - M_{i-1}\frac{h_i^2}{6}, C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1}) \quad (2)$$

M_i can be calculated from the M-continuity system,

$$\mu_i * M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, i = 0, \ldots, n \quad (3)$$

In this work, the hitting position and velocity are assured by adjusting the two final nodes in an appropriate way. The other variables and nodes are assigned during optimization.

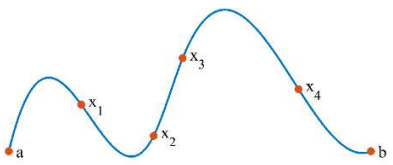

Fig. 3. Spline curve through node points $x_1, x_2, \ldots, x_6$

B. Bézier

The shape of these parametric curves is defined by control points. They are based on Bernstein polynomial functions at their core [23]. The corresponding Bézier curve (or Bernstein-Bézier curve) for $n+1$ control points $P_0, P_1, \ldots, P_n$ is defined as [24,25],

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad (4)$$

where, $B_{i,n}(t)$ is a Bernstein polynomial [17] and $t \in [0,1]$. A Bernstein polynomial of degree n is defined by.

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (5)$$

where,

$$\binom{n}{i} = \frac{n!}{i!(n-1)!} \quad (6)$$

The trajectory of this curve is a polynomial of degree n. The hitting position and velocity are assigned as the boundary conditions for the n^{th} order polynomial. The other variables are assigned during optimization.

C. Dubins Curve

In 1957, Dubins [26] used circular arcs and straight-line segments to find the shortest path of bounded curvature joining two points in a plane with a specific direction of motion [27]. The trajectory of Dubins curve is a polynomial of 6^{th} degree. The hitting position and velocity are assigned as the boundary conditions for the polynomial. The remaining variable is assigned during optimization.

The Spline, Bezier and Dubins trajectories are defined in this section. Regarding dynamic cost function mentioned (Section IV-C), each trajectory may perform better in a specific hitting condition. The final hitting trajectory generation process and Training part of the artificial neural network are described in the following sections.

IV. Optimal Trajectory Selection Strategy

As shown in Fig. 2, a novel approach is proposed to find the optimal trajectory with a fixed total time (hitting time), that puts the moving platform in the desired position with the desired velocity. First, to consider every possible hitting position, the workspace of the robot is discretized into a certain number of hitting points.

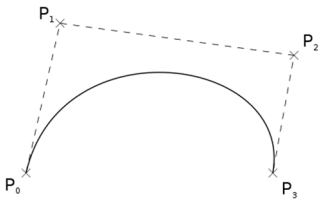

Fig. 4. Bézier curve through control points $P_0, P_1, \ldots, P_3$

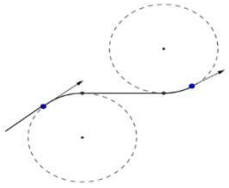

Fig. 5. Dubins Curve

For each hitting point, different possible vectors of the hitting velocity are considered (see Section IV-A). For all these hitting conditions (position/velocity), the three aforementioned trajectories are optimized to produce an optimal trajectory. The optimization process is done to minimize a kinematic cost function that includes the maximum velocity, acceleration, and jerk along the trajectory. Details of the cost function are presented in Section IV-B and equation (7). Second, the dynamics model of the robot is simulated and the maximum motor torques are calculated for each optimized trajectory. Then, a dynamic cost function is calculated that considers the maximum jerk, acceleration, and torques along the trajectories. Details of the dynamic cost function are provided in Section IV-B and equation (10). The trajectory type that corresponds to the minimum value of the dynamic cost function (10), is considered as the optimal hitting task trajectory. Then, a dataset is created that includes the hitting conditions (hitting position/velocity) and the corresponding optimal trajectory types. Third, a neural network is designed to learn the complicated relationship between the hitting conditions and the corresponding optimal type of the hitting task trajectory. This network is trained with the dataset, obtained in the previous step.

In the practical implementation of the proposed method, the trained neural network will receive the hitting condition from the hitting policy unit. The trained network will then define the optimal trajectory type for the received hitting condition. Finally, a kinematic optimization process will be performed to produce the entire hitting trajectory, which will be sent to the robot control unit.

A. Workspace Partitioning

To create a suitable training dataset for the neural network, the hitting conditions are discretized into small intervals. A three-column matrix is defined with its first three columns containing the three $X, Y,$ and Z components of the hitting positions, and the second three columns including the three components of hitting velocity V_x, V_y, V_z. Each row of this matrix provides a training set for the neural network. The range of the hitting velocity magnitude is considered between 0 to 10 [m/s] [19]. The direction of the hitting velocity is defined by two angles in the spherical system which range from -25 to 25 degrees for θ and from -45 to 45 degrees for ϕ. Considering three trajectory types for each hitting condition, a total number of 81900 trajectories are created by the discretization method. The schematic of the hitting point and spherical hitting angle are shown in Fig. 6.

B. Optimization of The Trajectories

In order to find the best solution for desired kinematics parameters, a kinematic optimization process is applied. As the computing time of the trajectory generation is very important, a local minimum is preferred rather than finding the global minimum, with more calculation time.

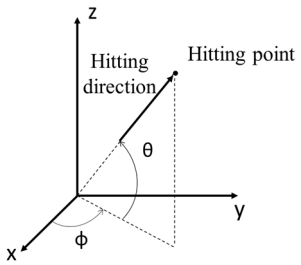

Fig. 6. The schematic of the hitting point and the spherical hitting angles.

Therefore, a classic optimization method is used. The cost function of this multi-objective optimization problem is defined in (7), which consists of the absolute value of the maximum jerk, acceleration, and velocity along each trajectory.

$$\boldsymbol{I_t} = w_1 * j_{max} + w_2 * a_{max} + w_3 * v_{max} \qquad (7)$$

In (7), w_1, w_2, and w_3 are the weights of the multi objective optimization and v_{max}, a_{max}, and j_{max} are the maximum values of velocity, acceleration, and jerk of moving platform along the path, respectively.

C. Cost function

To choose the best trajectory type for each hitting condition, the trajectories are compared with a certain criterion. This criterion consists of kinematic and dynamic parameters of the robot.

1) Kinematics: The structure of Delta robot and its actuators have limitted jerk and acceleration capacities. The actuators may not be able to afford high-acceleration motions. Moreover, the structure of the robot, such as the moving platform, links, and plates, may not be robust enough to tolerate the huge stresses resulted from high jerks and accelerations. The acceleration and jerk of the joint motions are obtained by successive differentiation of their velocity, which is calculated as,

$$\dot{\theta} = J^* \dot{X} \qquad (8)$$

where $\dot{\theta}$ is joint velocity vector, $\dot{X}$ is the cartesian velocity of the moving platform, and J^* is the jacobian matrix [28].

2) Dynamics: The torque capabilities of the actuators are the main limitations for the trajectory generation of Delta robots. In order to find the joint torques, the equation of motion of the delta robot (9) is used [29].

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + J^T(q)\,\lambda = Q \qquad (9)$$

In (9), M, C and G are the mass matrix, the centrifugal/Coriolis matrix, and the gravitational vector, respectively. q is generalized coordinate vector, λ is a vector consisting of the Lagrangian multipliers and Q is the generalized force vector. The constraint Jacobian matrix J is a matrix, which consists of partial derivatives of the constraint equations concerning the generalized coordinates. Details of the dynamic and kinematic model of Delta robot may be found in [28] and [29].

In the kinematics and dynamics section, the importance of each parameter is described. To achieve the optimized kinematics and dynamics parameters, each of the parameters should be weighted. As the structure of the delta robot is not robust enough to resist large inertial forces, the acceleration is the first parameter to check. Moreover, the actuators cannot apply large torques and the required torque is another parameter to evaluate. By considering jerk, trajectories can be executed more rapidly and accurately [30]. Therefore, jerk is another parameter as important as acceleration and torque. All of these parameters are included in a so-called dynamic cost function,

$$I_n = \mathrm{w}_1' * j_{a,max} + \mathrm{w}_2' * a_{a,max} + \mathrm{w}_3 * \tau_{a,max} \qquad (10)$$

where, w_1', w_2', and w_3' are the weights of multi objective optimization and $\tau_{a,max}$, $a_{a,max}$, and $j_{a,max}$ are the maximum values of the actuator torque and the maximum acceleration and jerk in the motion of the actuators, respectively.

D. Neural Network

The results of the comparison between the three types of trajectories are not inferable enough to decide which curve is the most suitable for a ping-pong robot. Therefore, a neural network is used to decide which trajectory type is best choice for each hitting condition. The network is trained using the dataset, mentioned in the previous sections. For this neural network, the 6 inputs are the hitting position (three components) and the hitting velocity (three components). The output of the network is an integer number between 1 to 3, each representing one of the three types of the trajectories. The output type of trajectory is the one that has the lowest value of cost function (10).Cascade-forward neural network is the chosen type of the neural network that uses the Variable Learning Rate Gradient Descent as the training function. In this neural network, 100 neurons are used in the hidden layer, as shown in Fig. 7. The training dataset is 70 percent of the total dataset, 15 percent is used as the validation dataset and the remaining 15 percent is utilized to test the performance of the network.

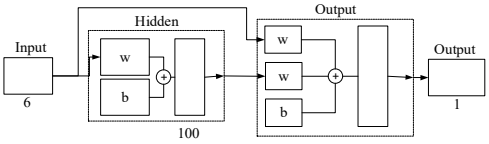

Fig. 7. Structure of the implemented neural network

V. Results And Discussion

The reason for using a neural network is the complexity of the results. To prove that, in the first step, we try to cluster the data and the results of the three trajectory types in terms of the hitting position in the XY plane for a fixed hitting velocity and direction. The maximum jerk, acceleration, and torque are investigated in Fig. 8. The intensity of the color represents the number of cases that each trajectory is selected as the optimal trajectory, with minimum dynamic cost. Fig. 8 reports the results in nine different zones on the left side of the table. Because of the symmetry, the Right side of the table is not depicted. The results show that each trajectory has its advantages and disadvantages in different hitting conditions. For example, by considering the maximum jerk of trajectories, the Bézier trajectory seems to do better in most of the conditions, while the spline trajectory is not preferred in any region. By considering maximum acceleration and maximum torque, the results are different. The spline trajectory seems to do better, and the Bézier trajectory does not seem to be a good choice. Although, respecting the selected criterion, the Dubins trajectory has not been the best trajectory in any zones, but it has a better performance than the spline trajectory regarding the jerk parameter, and has a comparable performance with the Bézier trajectory regarding the torque and acceleration.

Fig. 8 compares the trajectories by only one criterion at each part of the figure. However, to provide a better comparison of the trajectories, it is necessary to consider a combination of these parameters with suitable weights. This multi-objective comparison is performed by the use of the cost function defined in equation (10) and the results are provided in Fig. 9. In Fig. 9, the type of each trajectory is defined by a shape. Cube, sphere, and pyramid are representations for the Bézier, spline, and Dubins trajectories, respectively. Considering 0 m/s hitting velocity (Fig. 9 (a)), spline has the best performance in most positions. Dubins has never been successful relative to the others. By considering $\vec{V} = [5\ 2\ 0]^T$ [m/s] for the hitting velocity (Fig. 9 (b)), Bézier has the best performance in most of the positions. Particularly spline has the best performance in the nearest positions to the net, and also, Dubins seems suitable partly in side positions. Considering $\vec{V} = [8\ 0\ 3]^T$ [m/s] for the hitting velocity (Fig. 9 (c)), Bézier performs best in the most positions. Spline has never been suitable for any positions and Dubins seems particularly better in front positions. The results in Fig. 8 and Fig. 9 verify that the selection of the best trajectory is a complicated task. This observation justifies the application of the neural network. The hardware used to run the codes was Intel(R) Core(TM) i7-8550U CPU @ 1.806 [Hz] Memory Speed 2400 [MHz]. The calculation time for every input of the nueral network has estimated to be 7.6 [ms]. In case of not using the neural network to select the best trajectory, we are required to generate all the three types of trajectories. In this case, according to Table I, the total time is equal to 1224 [ms].

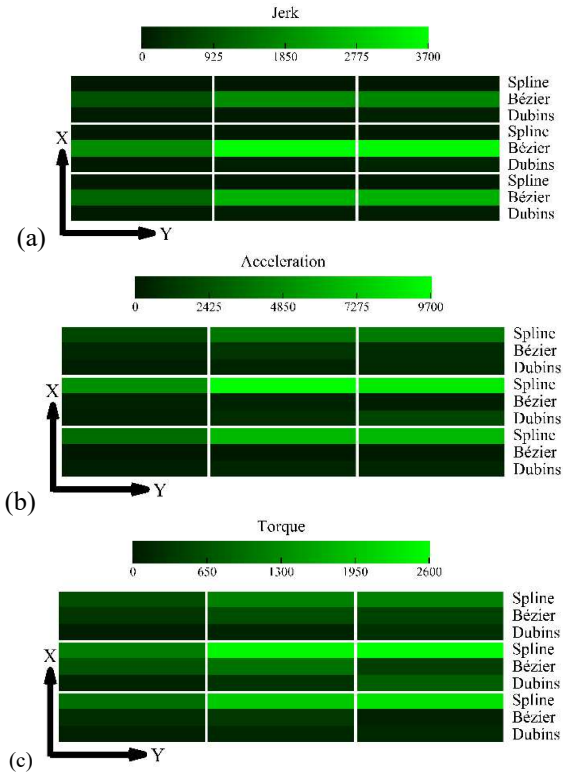

Fig. 8. Comparison of the kinematic and dynamic parameters of the actuators' motion for different trajectories. (a) Jerk comparison, (b) acceleration comparison, (c) Torque comparison.

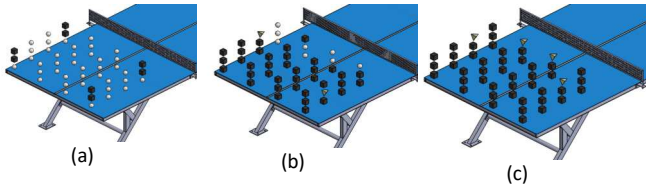

Fig. 9. The optimum trajectories at different hitting positions for the different hitting velocities. Cube, sphere, and pyramid are representations of the Bézier, spline, and Dubins trajectories. (a) Hitting velocity = [0 0 0] [m/s], (b) Hitting velocity = [5 2 0] [m/s], (c) Hitting velocity = [8 0 3] [m/s].

TABLE I. CALCULATION TIME

	Spline	Bézier	Dubins
Time of trajectory generation + NN (ms)	424.9	69.3	651.0
Saved time (ms)	697.5	1053.1	471.4
Percent reduction of total time (%)	62.1	93.8	41.9

Using the neural network, only one of the trajectories is generated and the execution time of the network is also added to it. As shown in Table I, the neural network has reduced the total time of trajectory generation and selection by an average of 65.93%.

VI. CONCLUSION

This paper addresses the optimal trajectory generation for a Delta ping pong robot. The main idea is to use the neural networks to automatically select the best trajectory for each hitting condition. Three of the most common path generation curves are used to generate optimized trajectories. The inverse kinematics and inverse dynamics of the Delta robot are solved and the maximum jerk, acceleration, and torque of the actuators are calculated for each trajectory. The best trajectory is then defined using a multi-objective cost function. Finally, a neural network is designed to learn the best trajectory for each hitting condition. Implementation of the trained neural network enables the ping pong robot to automatically select the best trajectory for every hitting position/orientation and velocity. The results indicate that the trained neural network successfully selects the best curve with an accuracy of 85%.

REFERENCES

[1] Ensmenger, Nathan. "Is chess the drosophila of artificial intelligence? A social history of an algorithm." Social studies of science 42.1 (2012): 5-30.

[2] Silver, David, et al. "Mastering the game of go without human knowledge." nature 550.7676 (2017): 354-359.

[3] Andersson, Russell L. "Dynamic sensing in a ping-pong playing robot." IEEE Transactions on Robotics and Automation 5, no. 6 (1989): 728-739.

[4] Yang, Ping, De Xu, Huawei Wang, and Zhengtao Zhang. "Control system design for a 5-DOF table tennis robot." In 2010 11th International Conference on Control Automation Robotics & Vision, pp. 1731-1735. IEEE, 2010.

[5] Serra, Diana, Aykut C. Satici, Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. "An optimal trajectory planner for a robotic batting task: the table tennis example." In International Conference on Informatics in Control, Automation and Robotics, vol. 3, pp. 90-101. SCITEPRESS, 2016.

[6] Huang, Yanlong, De Xu, Min Tan, and Hu Su. "Adding active learning to LWR for ping-pong playing robot." *IEEE Transactions on Control Systems Technology* 21, no. 4 (2012): 1489-1494.

[7] Matsushima, Michiya, Takaaki Hashimoto, Masahiro Takeuchi, and Fumio Miyazaki. "A learning approach to robotic table tennis." *IEEE Transactions on robotics* 21, no. 4 (2005): 767-771.

[8] Jahandideh, Hossein, Mohammad Nooranidoost, Behnam Enghiad, and Armin Hajimirzakhani. "Ball striking algorithm for a 3 DOF ping-pong playing robot based on particle swarm optimization." In 2012 16th International Conference on System Theory, Control and Computing (ICSTCC), pp. 1-6. IEEE, 2012.

[9] Huang, Yanlong, Dieter Büchler, Okan Koç, Bernhard Schölkopf, and Jan Peters. "Jointly learning trajectory generation and hitting point prediction in robot table tennis." In 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp. 650-655. IEEE, 2016.

[10] Matsushima, Michiya, Takaaki Hashimoto, Masahiro Takeuchi, and Fumio Miyazaki. "A learning approach to robotic table tennis." IEEE Transactions on robotics 21, no. 4 (2005): 767-771.

[11] Lin, Hsien-I., Zhangguo Yu, and Yi-Chen Huang. "Ball tracking and trajectory prediction for table-tennis robots." Sensors 20, no. 2 (2020): 333.

[12] Acosta, Leopoldo, J. J. Rodrigo, Juan A. Mendez, G. Nicolás Marichal, and Marta Sigut. "Ping-pong player prototype." IEEE robotics & automation magazine 10, no. 4 (2003): 44-52.

[13] Andersson, Russell L. "Aggressive trajectory generator for a robot ping-pong player." IEEE Control Systems Magazine 9, no. 2 (1989): 15-21.

[14] Koç, Okan, Guilherme Maeda, and Jan Peters. "Online optimal trajectory generation for robot table tennis." Robotics and Autonomous Systems 105 (2018): 121-137.

[15] Koç, Okan, Guilherme Maeda, and Jan Peters. "A new trajectory generation framework in robotic table tennis." In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3750-3756. IEEE, 2016.

[16] Mülling, Katharina, Jens Kober, and Jan Peters. "A biomimetic approach to robot table tennis." Adaptive Behavior 19, no. 5 (2011): 359-376.

[17] Hilario, Lucia, Nicolás Montés, Marta C. Mora, and Antonio Falcó. "Real-time Trajectory Modification based on Bézier Shape Deformation." In IJCCI (ICEC), pp. 243-248. 2010.

[18] Yomchinda, Thanan, Joseph F. Horn, and Jack W. Langelaan. "Modified Dubins parameterization for aircraft emergency trajectory planning." Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 231, no. 2 (2017): 374-393.

[19] Senoo, Taku, Akio Namiki, and Masatoshi Ishikawa. "High-speed batting using a multi-jointed manipulator." In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, vol. 2, pp. 1191-1196. IEEE, 2004.

[20] Ravankar, Abhijeet, Ankit A. Ravankar, Yukinori Kobayashi, Yohei Hoshino, and Chao-Chung Peng. "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges." Sensors 18, no. 9 (2018): 3170.

[21] Weisstein, Eric W. "Spline." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/Spline.html (accessed Aug. 28, 2022).

[22] Quarteroni, Alfio, Riccardo Sacco, and Fausto Saleri. "Numerical Mathematics (Texts in Applied Mathematics)." (2006).

[23] Weisstein, Eric W. "Bernstein Polynomial." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/BernsteinPolynomial.html(accessed Aug. 28, 2022).

[24] Choi, Ji-wung, Renwick Curry, and Gabriel Elkaim. "Path planning based on bézier curve for autonomous ground vehicles." In Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, pp. 158-166. IEEE, 2008.

[25] Weisstein, Eric W. "Bézier Curve." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/BezierCurve.html(accessed Aug. 28, 2022).

[26] Dubins, Lester E. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents." American Journal of mathematics 79, no. 3 (1957): 497-516.

[27] Reeds, James, and Lawrence Shepp. "Optimal paths for a car that goes both forwards and backwards." Pacific journal of mathematics 145, no. 2 (1990): 367-393.

[28] Hadfield, Hugo, Lai Wei, and Joan Lasenby. "The forward and inverse kinematics of a delta robot." In Computer Graphics International Conference, pp. 447-458. Springer, Cham, 2020.

[29] Lee, Fu-Shin, and Chen-I. Lin. "Controller Design for a Delta Robot Using Lagrangian Multipliers." (2021)

[30] Zanotto, Vanni, Alessandro Gasparetto, Albano Lanzutti, Paolo Boscariol, and Renato Vidoni. "Experimental validation of minimum time-jerk algorithms for industrial robots." *Journal of Intelligent & Robotic Systems* 64, no. 2 (2011): 197-219.