Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **70**, *2*, pp. 241–246, Bucarest, 2025

A VELOCITY SELF-LEARNING ALGORITHM FOR TIME-OPTIMAL TRAJECTORY PLANNING ALONG FULLY SPECIFIED PATH – Part II

WENHU NAN[1], HAOJUN QIN[1]

Key words: Industrial handling robot; Velocity self-learning; Hermite interpolation; Correction trajectory; Actual joint torque.

In response to the problem of uncertainty in the system dynamics model during time-optimal trajectory planning for industrial handling robots, a novel online, self-learning, model-free time-optimal trajectory planning method is proposed. First, offline kinematic constraints and the Hermite interpolation algorithm are used to obtain the optimal spline velocity curve under kinematic constraints. Then, online trajectory data of the robot's operation is collected, and the trajectory generation method using a self-learning strategy is employed to iteratively refine the trajectory iteratively, resulting in a time-optimal trajectory under actual dynamic constraints. Finally, taking the UR5e cooperative robot and the ABB IRB9670-235 industrial robot as the experimental platform, experiments verify the effectiveness and efficiency of the proposed method.

1. EXPERIMENT STUDY

1.1 THE ONLINE INTERACTIVE SELF-LEARNING EXPERIMENT

Figure 8 shows this research takes the ur5e 6-DOF cooperative robot as the experimental platform. The D-H parameters of the robot are shown in Table 2. An absolute encoder is installed inside each rotating joint of the robot to detect the joint position. The robot carries a grinding head with a mass of 2.3 kg.

Fig. 8 – UR5E 6-DOF cooperative robot.

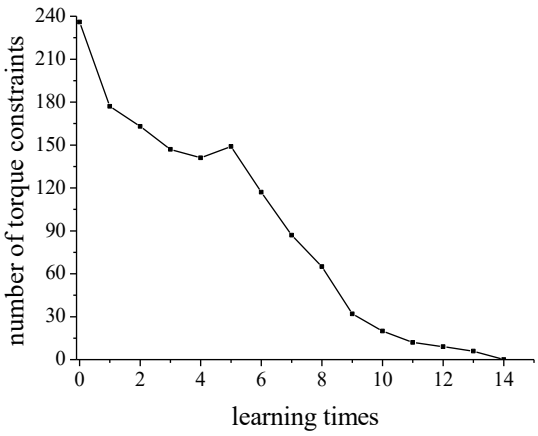


Fig. 9 – Schematic diagram of learning process.

Like most industrial robots, UR5E robot has no torque sensor installed inside the joint, so the joint torque cannot be obtained in real time, but it can be calculated according to the motor current. The motor torque is directly proportional to the current, and the current value can be obtained at low cost through the servo bus, so as to provide data basis for model free dynamics online self-learning planning. At the hardware level, the robot is connected with the computer through Ethernet. On software level, the robot and computer exchange real-time data using the Real-Time Data Exchange (RTDE) protocol of UR robot, and the data acquisition frequency is 125 Hz. The data acquisition and control programs are developed based on the UR robot software package of ROS.

Figure 10 shows the strategy trajectory generated in the self-learning process. The black solid line represents the speed track at learning times z = 0, which denotes the initial speed track during self-learning. The red, green, blue, and green lines represent the speed tracks during the learning process. Finally, the magenta line represents the speed curve after successful learning.

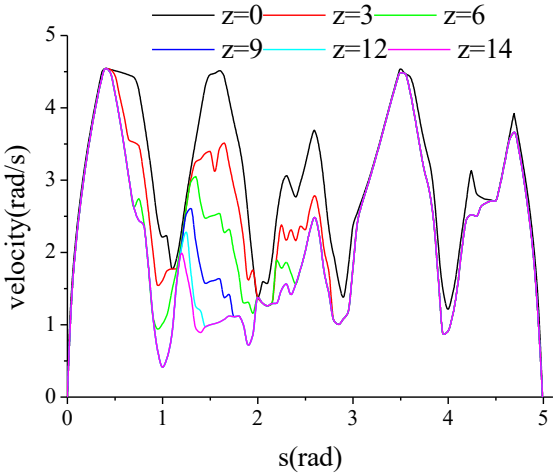


Fig. 10 – The joint velocity curve in learning process.

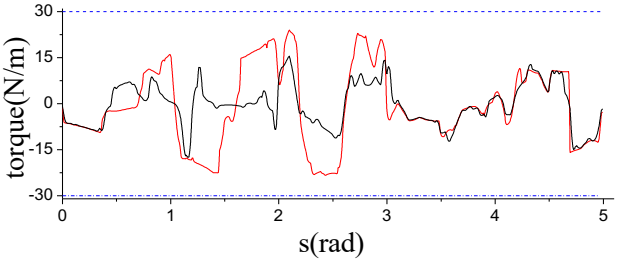
Figure 11 shows the actual measured joint torque curve before and after learning. The extreme value of six joint torque constraints is 30N/m. The red line and black line are the measured torque before and after learning respectively. It can be seen that after 14 times of learning, all joint torques are within the limited constraint value.

As shown in Fig. 11(b) and Fig. 11(c), the torque of joint 2 and joint 3 is greater than that of other joints, and the actual measured torque of joint 2 and joint 3 exceeds the extreme value of joint torque before learning. After 14 times of learning, the actual torque curves of joint 2 and joint 3 are within the extreme value of joint torque. This is because the off-line kinematic planning algorithm before learning does not consider the actual dynamic

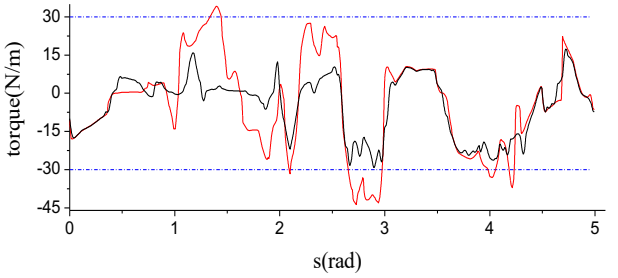
[1] Key Laboratory of heavy-duty flexible robot in mechanical industry, Lanzhou University of Technology, Lanzhou 730050, China.
E-mail: nanwenhu@163.com; 20160028@lut.edu.cn

DOI: 10.59277/RRST-EE.2025.2.15

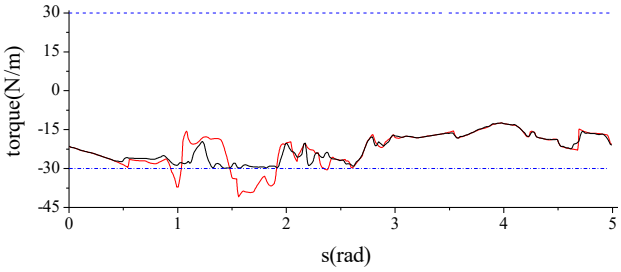
factors. As a result, when the UR robot follows the speed track before learning, the torque of some joint path points exceeds the limit. After interactive learning with the real environment, the dynamic constraints are introduced, and the joint velocity trajectory is gradually adjusted locally. Finally, the measured torque of the trajectory meets the joint torque constraints. This proves the effectiveness of the self-learning method proposed in this paper.



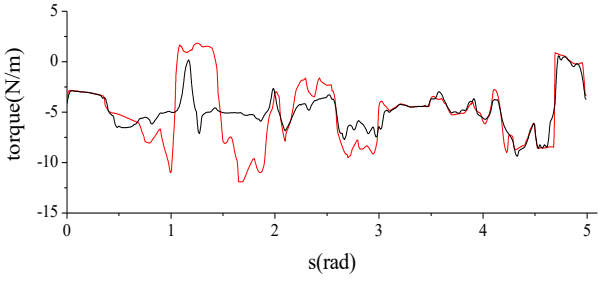
(a) The torque of joint 1.



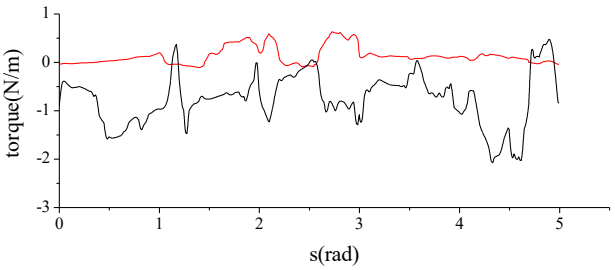
(b) The torque of joint 2.



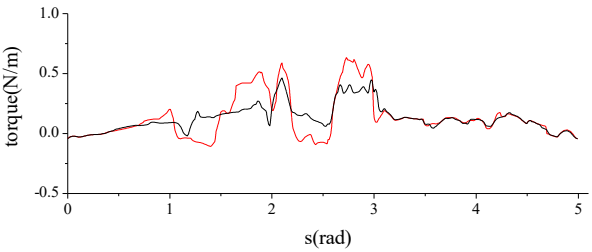
(c) The torque of joint 3.



(d) The torque of joint 4.



(e)The torque of joint 5.



(f) The torque of joint 6.

Fig. 11 – Measured torque before and after learning.

1.2 THE ONLINE INTERACTIVE SELF-LEARNING EXPERIMENT FOR ABB IRB9670-235

We validate the proposed approach in this paper with an industrial use case applied in the field of automobile hood stamping testing. The working environment is an automatic forging workstation. During the working process, the robot in the workstation seizes the sheet blank from the conveyor line and transports the sheet to the forging machine. The schematic diagram of our proposed test system in ABB RobotStudio6.08 is shown in Fig. 12. used to carry out a dynamic simulation with MATLAB 2021a software and conducts simulation research on time optimal trajectory planning. When the mass of the forged plate and fixture is 100 kg, a semi elliptical path is selected for handing path. First, we conduct off-line programming with Robotstdio6.08 and obtain the robotic trajectory in different handling speeds. And then input the trajectory information into the robot dynamics simulation system to verify the number of

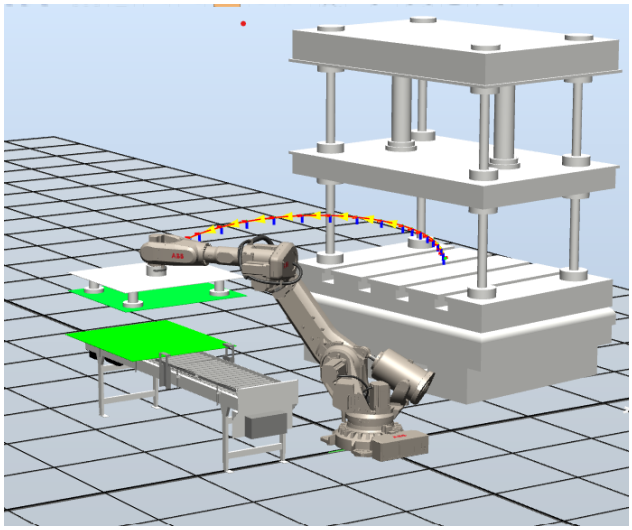


Fig. 12 – The schematic diagram of our proposed test system.

torque constraints and the extent of violating the maximum torque when the robot works with the trajectory planned by offline software. Finally, the related trajectory data of the offline planning is counted. The simulation results are shown in Table 4. With the increase of the handling speed, the number of torque constraints and torque overshoots gradually increase, but the torque overshoot changes are smaller by comparison with the number of torque constraints. The main reason is that the gravity of the robot and the handling load are large and the dynamic factors are small. According to the analysis of the track running time, the efficiency of the robot to complete the automatic handling is gradually improving with the improvement of the handling speed. However, for such a heavy load, the number of paths

where the robot motor exceeds the maximum torque is increasing. Long term operation in torque overshoot state will cause damage to the robot.

Table 4
Statistical analysis of trajectories at different velocity

Num	Speed (m/s)	Torque constraint points	Trajectory time(s)	Torque overshoot (%)
1	0.08	0	54.744	0
2	0.1	1	32.976	81.33
3	0.2	3	16.416	82.92
4	0.3	5	10.992	83.29
5	0.4	8	8.304	83.48
6	0.5	10	6.696	84.73
7	0.6	11	5.664	86.03
8	0.8	12	4. 334	98.13
9	1	16	3.552	84.15
10	1.2	21	2.616	87.72

Using the method proposed in this paper, input the semi elliptical transport path, discretize the motion path, input the algorithm, and finally obtain the optimal velocity of kinematics, input the robot driver, drive the robot motion, initialize the trajectory with the optimal velocity of motion after spline interpolation, conduct the first iterative interaction experiment with the virtual dynamics parameters, and subsequently take the learning descent coefficient d = 0.15, the interactive learning experiment was conducted with the strategy in subsection 2.3. The results are shown in Fig. 13. It can be seen that before learning, there are 16 torque constraint points at first, and the number of joint torque constraint points gradually decreases with the interactive self-learning. As shown in Fig. 14, after 7 self-learning sessions, there are no torque constraint points.

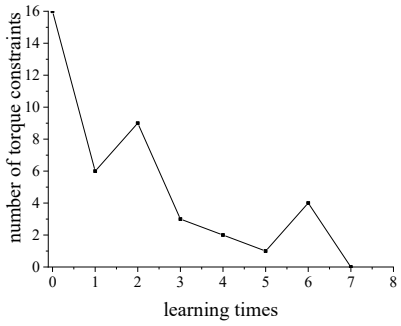


Fig. 13 – Schematic diagram of learning process

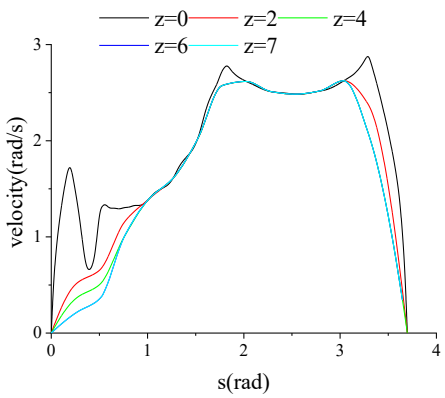
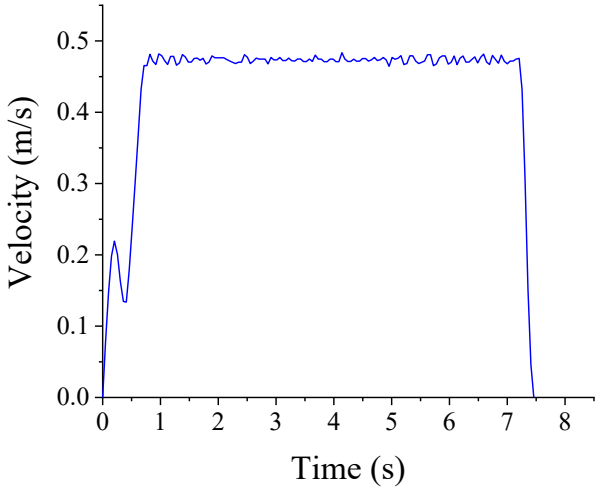
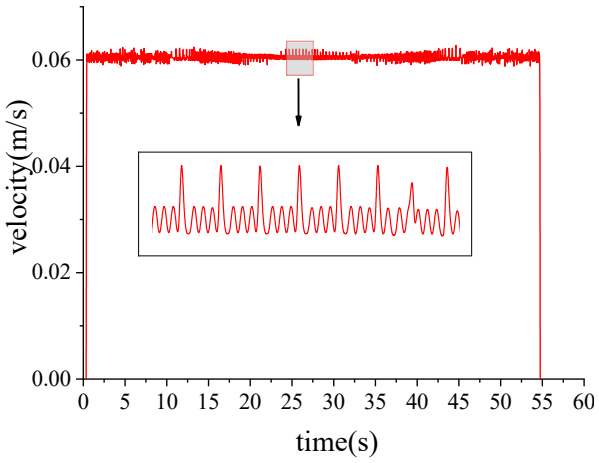


Fig. 14 – The joint velocity curve in learning process.

To show the benefit of the proposed approach, select the workspace velocity of 0.06m/s in Table 4 as an example for comparative study on the Robotstdio method and the proposed method. The length of the selected handing path is 3.2895 m. The workspace velocity of two methods is shown in Fig. 15(a) and (b), which can be observed that the two-velocity curve is all like the trapezoidal velocity curve. However, the velocity fluctuation with Robotstdio method is larger than that of the proposed method. The speed curve planned by the algorithm in this paper is approximate to trapezoid, but the speed fluctuation is small, except in the initial stage as shown in the Fig. 15(a). We conduct a comprehensive analysis of the velocity distribution and find that the velocity planned in the proposed method is more stable and efficient than that with Robotstdio.



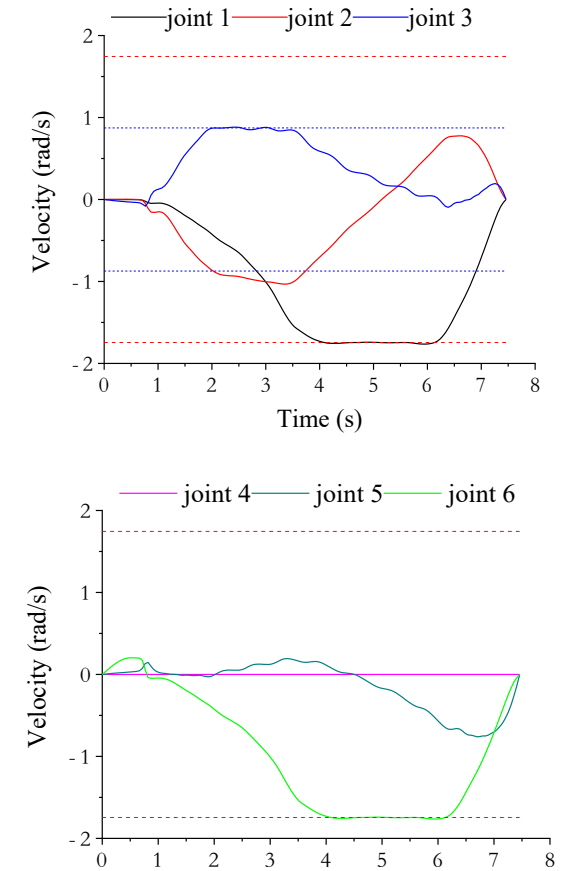
(a)



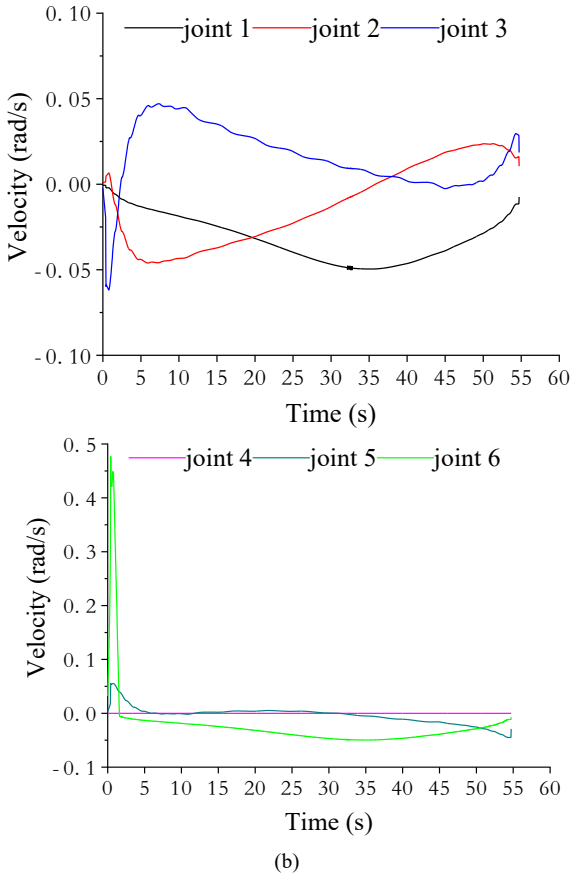
(b)

Fig. 15 – Comparison of (a) the workspace velocity planned by the proposed algorithm, (b) the angular velocity by the Robotstdio.

With the proposed self-learning algorithm and the Robotstdio method, the planned result of the joint velocity along the path is shown in Fig. 16(a). It can be observed that the planned trajectory with the proposed method is continuous, the joint velocity changes smoothly, which satisfies the limit of the angular velocity (1.7453,1.7453,0.8727,1.7453,1.7453,1.7453) (rad/s) as shown in Fig. 16(a) with dot line. Compared with the proposed method in Fig. 16(a), we can observe that the joint velocity planned by Robotstdio changes unevenly at the starting stage in Fig. 16(b), which indicates that the acceleration at the starting stage is high.



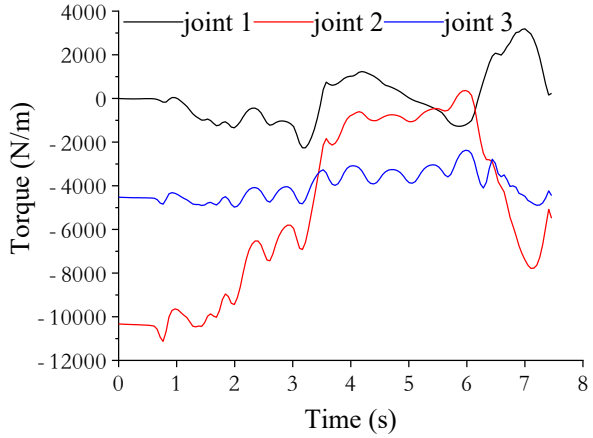
(a)

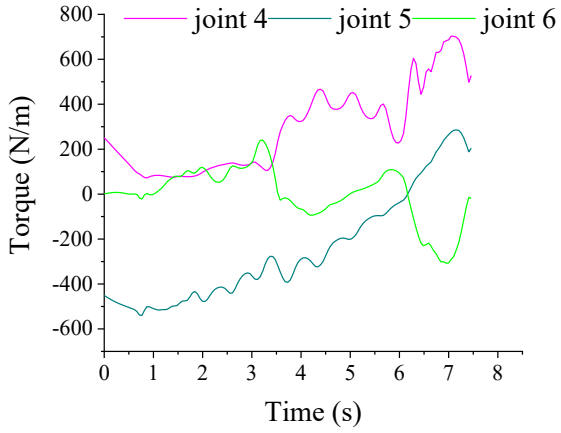


(b)

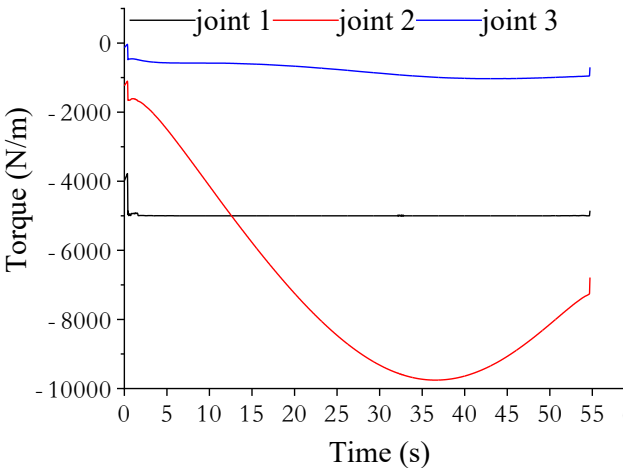
Fig. 16 – Comparison of (a) the angular velocity planned by the proposed algorithm, (b) the angular velocity by the Robotstdio.

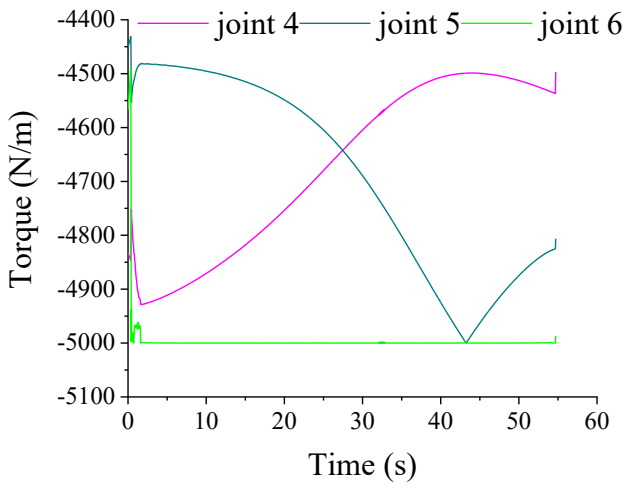
As shown in Fig. 17(a), the joint torque with proposed algorithm in this paper changes steadily and the amplitude of fluctuation is small. The torque of all the joints along the handing path is continuous, which also satisfies the limit of the joint torque (5000, 12000, 5000, 5000, 5000, 5000) (N/m). Compared with the trajectory planned by Robotstdio, the joint torque fluctuates greatly at the initial stage as shown in Fig. 17(b). In particular, the torque fluctuation value of joint 6 is larger than that of others, which indicates that the trajectory planned by Robotstdio does not consider dynamic factors, resulting in the occurrence of dynamic oscillations. The torque of the self-learning trajectory is within the actual torque constraint range, and the torque changes steadily. The results show that the algorithm proposed in this paper is effective for conventional transportation path learning.





(a)





(b)

Fig. 17 – Comparison of torque (a) planned by the proposed algorithm, (b) planned by the Robotstdio algorithm.

The action time of the proposed algorithm is 7.4601 seconds, and the action time of the Robotstdio method at 0.06m/s is 54.744 seconds. The work efficiency of the proposed algorithm is 7.34 times that of the Robotstdio method. It demonstrates that the self-learning algorithm can enhance the robot's working efficiency. Therefore, the proposed self-learning algorithm can be applied to high-efficiency handling operations, generating a continuous trajectory that satisfies both kinematic and dynamical constraints.

2. CONCLUSION

This paper presents a velocity self-learning method, a finite approach for time-optimal trajectory planning along specified paths with actual dynamic constraints. The kinematic time-optimal velocity is obtained with a dynamic programming technique, and the Hermite spline interpolation amplitude limiting algorithm is designed to smooth the initial velocity of the self-learning. To determine the time-optimal trajectory that satisfies actual dynamic constraints, an online velocity self-learning trajectory decision algorithm is proposed without requiring a dynamic model. The interactive process of online self-learning is described in detail.

Finally, two experiments are conducted to demonstrate the effectiveness of the proposed method. First, we utilize the UR5E cooperative robot as the experimental platform for the star transportation path. When the load is 2.3 kg, through 14 interactive learning, the time-optimal trajectory under the constraints of kinematics and the actual joint torque is obtained, which proves the effectiveness of the proposed method.

Second, a robotic trajectory planning for the hand operation in automobile hood stamping is done by a self-learning method and Robotstdio method. The experimental results demonstrate that the proposed algorithm's work efficiency is 7.34 times that of the Robotstdio6.08 method, indicating that the self-learning algorithm can enhance the robot's working efficiency.

Future work should include investigating the issue of dynamic singularities using the velocity self-learning method to enhance the algorithm’s adaptability. The velocity self-learning method would also benefit from dynamic singularities. The application of the self-learning method in industrial robots, particularly those involved in handling tasks, as well as welding robots with a time-optimal goal, could be advantageous.

ACKNOWLEDGEMENTS

The support of the Gansu Provincial Science and Technology Commissioner Special Project under Grant 23CXGA0024 is gratefully acknowledged. The authors would like to thank the editor for their valuable contribution to this work and the anonymous reviewers for their insightful comments.

Received on 25 December 2023

REFERENCES

1. H. Pham, Q.C. Pham, *A new approach to time–optimal path parameterization based on reachability analysis*, IEEE Transactions on Robotics, **34**, *3*, pp. 645–659 (2018).
2. E. Barnett, C. Gosselin, *A bisection algorithm for time-optimal trajectory planning along fully specified paths*, IEEE Transactions on Robotics, **37**, *1*, pp. 131–145 (2020).
3. K. Shin, N. McKay, *Minimum-time control of robotic manipulators with geometric path constraints*, IEEE Transactions on Automatic Control, **30**, *6*, pp. 531–541 (1985).
4. J.A. Rojas-Quintero, F. Dubois, H.C. Ramírez-De-Ávila, *Riemannian formulation of Pontryagin’s maximum principle for the optimal control of robotic manipulators*, Mathematics, **10**, *7*, pp. 1117–1128 (2022).
5. K. Shin, N. McKay, *A dynamic programming approach to trajectory planning of robotic manipulators*, IEEE Transactions on Automatic Control, **31**, *6*, pp. 491–500 (1986).
6. S. Singh, M. Leu, *Optimal trajectory generation for robotic manipulators using dynamic programming*, J. Dyn. Syst. Meas. Control, **109**, *2*, pp. 88–96 (1987).
7. Q.C. Pham, *A general, fast, and robust implementation of the time-optimal path parameterization algorithm*, IEEE Transactions on Robotics, **30**, *6*, pp. 1533–1540 (2014).
8. D. Verscheure, B. Demeulenaere, J. Swevers, et al, *Time-optimal path tracking for robots: A convex optimization approach*, IEEE Transactions on Automatic Control, **54**, *10*, pp. 2318–2327 (2009).
9. Z. Kingston, M. Moll, L.E. Kavraki, *Sampling-based methods for motion planning with constraints*, Annual review of control, robotics, and autonomous systems, **1**, *1*, pp. 159–185 (2018).
10. F. Debrouwere, W.V. Loock, G. Pipeleers, et al, *Time-optimal path following for robots with convex–concave constraints using sequential convex programming*, IEEE Transactions on Robotics, **29**, *6*, pp. 1485–1495 (2013).
11. A. Tharwat, M. Elhoseny, A.E. Hassanien, et al, *Intelligent Bézier curve-based path planning model using chaotic particle swarm optimization algorithm*, Cluster Computing, **22**, *4*, pp. 1–22 (2019).
12. L. Zhang, et al, *Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm*, Journal of Mechanical Science and Technology, **35**, *7*, pp. 3171–3181 (2021).
13. A. Steinhauser, J. Swevers, *An efficient iterative learning approach to time-optimal path tracking for industrial robots*, IEEE Transactions on Industrial Informatics, **14**, *11*, pp. 5200–5207 (2018).
14. L. Yu, X. Shao, Y. Wei, K. Zhou, *Intelligent land-vehicle model transfer trajectory planning method based on deep reinforcement learning*, Sensors, **18**, *9*, 2905 (2018).
15. X. Lei, Z. Zhang, P. Dong, *Dynamic path planning of unknown environment based on deep reinforcement learning*, Journal of Robotics (2018).
16. S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, *Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection*, The International Journal of Robotics Research, **37**, *4-5*, pp. 421–436 (2018).

17. K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, J.B. Mouret, *Black-box data-efficient policy search for robotics*, In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 51–58 (2017).

18. S.A. Khader, H. Yin, P. Falco, et al, *Data-efficient model learning and prediction for contact-rich manipulation tasks*, IEEE Robotics and Automation Letters, **5**, *3*, pp. 4321–4328 (2020).