Learning Time-Optimal Control of Gantry Cranes

Zhong, Junmin; Nikovski, Daniel N.; Yerazunis, William S.; Ando, Taishi

TR2024-181 December 19, 2024

Abstract

The paper presents an experimental study on the application of deep reinforcement learning (DRL) methods to the problem of optimally transporting cargo loads by an overhead gantry crane in minimal time. Experiments in simulation using a physics engine on two versions of the problem, with two and four degrees of freedom and employing reward functions that reflect the objective of load stabilization in minimal time, demonstrate that policies trained with the Stochastic Actor Critic (SAC) DRL method achieve up to 20% shorter transport time in comparison with controllers designed by means of more traditional methods from the field of control engineering.

International Conference on Machine Learning and Applications (ICMLA) 2024

Learning Time-Optimal Control of Gantry Cranes

J. Zhong, D. Nikovski, W. S. Yerazunis
Mitsubishi Electric Research Labs
Cambridge, Massachusetts, 02139
email: {jzhong,nikovski,yerazunis}@merl.com

T. Ando
Mitsubishi Electric Automation, Inc.
Cambridge, Massachusetts, 02141
email: Taishi.Ando@meau.com

***Abstract*—The paper presents an experimental study on the application of deep reinforcement learning (DRL) methods to the problem of optimally transporting cargo loads by an overhead gantry crane in minimal time. Experiments in simulation using a physics engine on two versions of the problem, with two and four degrees of freedom and employing reward functions that reflect the objective of load stabilization in minimal time, demonstrate that policies trained with the Stochastic Actor Critic (SAC) DRL method achieve up to 20% shorter transport time in comparison with controllers designed by means of more traditional methods from the field of control engineering.**

***Index Terms*—Learning control, reinforcement learning, trajectory optimization**

I. Introduction

Cranes are an indispensable part of construction, material handling, warehousing, and supply chain operations, and their safe and fast operation has high economic significance. In order to optimize their use, short transport and settling times are usually desired. These two objectives are typically at odds with each other, because the faster the crane moves, the more its load tends to swing, delaying the settling and the productive use of the load. It is desired then to find a method to control cranes in a way that moves the load quickly while also counteracting sways of the load.

What makes this problem difficult is that practically all types of cranes are underactuated, that is, they have fewer actuators than degrees of freedom (DoF). A common type of crane is the overhead (gantry) crane shown in Fig. 1. It has two linear stages oriented perpendicularly to each other and actuated by motors, and another motor for hoisting the load vertically on a cable. In contrast to the three actuators available, the load has six degrees of freedom, allowing it to assume any position and orientation within the workspace of the crane. What makes possible the transportation of a load to a desired position by such a crane is the fact that the dynamics of the crane are stable, and in the absence of control effort, the load will eventually settle into a position directly below the trolley cart it is suspended from. Thus, a simple control method consists of bringing the trolley to the desired (x, y) coordinates of the load and letting gravity stabilize the load. This method is commonly used in practice but is far from optimal, for the reason mentioned above – the faster the crane moves, the wider the load swings and hence the longer it takes to settle on its own.

A wide variety of optimal control methods have been proposed to optimize the transportation and settling time of crane loads [1]. All of them have to deal with the severe under-actuation of the crane, implying limits on which DoFs can be controlled at a given time. For this reason, control methods usually first compute a desired trajectory for the trolley and the load of the crane (a planning stage), followed by stabilizing of the system along this trajectory (an execution stage). Early work on trajectory planning used the tools of optimal control, such as Pontryagin's maximum principle, to derive optimal paths given a dynamical model of the system in analytical form [2]. Prior knowledge and/or insight about the shape of the optimal trajectory can be used in conjunction with numerical optimization techniques to parameterize the optimal trajectory as a spatial curve and find an optimal solution numerically [3]. Other classical control techniques have been used for stabilizing the crane and load along a computed trajectory, such as sliding mode control [4], linear quadratic regulator (LQR) control [5], [6], direct inverse control [7], etc.

A major limitation of methods based on classical control is that they rely on the availability of a dynamical model of the crane in analytical form (a set of ordinary differential equations), usually derived on the basis of Lagrangian dynamics. Deriving such a model, and then using it in controller design, is very laborious and error prone. Moreover, the difficulty associated with this method for controller design is further exacerbated by the existence of multiple types of cranes (at least three major varieties exist: gantry, rotary, and boom cranes, with multiple variations), each with different mechanical design, possibly complicated by factors such as stretching cable and flexible components. This means that no universal controller exists, and consequently a laborious and costly manual design process must be followed for each type of mechanism.

This difficulty creates a strong motivation for applying methods for controller design based on alternative methodologies, including machine learning. An early application of neural networks to the problem of trajectory tracking for overhead cranes was proposed in [8]. Recently, a combination of a classical controller to bring the trolley of the crane to the desired position with a load anti-sway controller based on reinforcement learning was proposed in [9].

The above two methods apply ML to one component of the overall controller (trajectory stabilization or anti-sway control, respectively). In contrast to them, in this paper we explore the possibility of applying RL to the problem of learning an entire control policy for a gantry crane that encapsulates both the trajectory planning and the trajectory stabilization components of the controller. This approach is inspired by recent successful applications of learning control to difficult optimal control problems such as in-hand manipulation of objects by robots that require both computation of long and complicated trajectories of motion, as well as stabilizing the system around them by means of feedback control [10], [11].

Section II describes the version of the problem that we are solving, and Section III describes the modeling and control methods used in our study, including the RL and classical controller design methodologies that we have applied to the problem. Section IV presents empirical results on two versions of the problem, and Section V proposes directions for future work and concludes the paper.

II. Problem Statement

In our experimental study, we chose to investigate control methods for overhead gantry cranes, as they are among the largest cranes in use and their usual method of deployment (stationary in locations such as ports, warehouses, etc.) and the highly repetitive nature of their operation stand to benefit the most from optimized high-speed movement.

The crane is an underactuated six-DoF system, only five of whose DoFs are controllable in practice. In our study, we chose to keep the length of the cable constant and eliminate the rotation of the load about the cable's axis, restricting the DoFs of the studied system to four. (This corresponds to suspending the load on a thin rod moving about the suspension point on two hinge joints.) The state of the resulting mechanical system is eight-dimensional and can be described by the vector $\mathbf{x} \doteq [c_x, c_y, \theta_x, \theta_y, v_x, v_y, \dot{\theta}_x, \dot{\theta}_y]$, where c_x and c_y are the cart's coordinates in its plane of motion, θ_x and θ_y are the cable's angles with respect to its vertical position, v_x and v_y are the cart's linear velocities, and $\dot{\theta}_x$ and $\dot{\theta}_y$ are the load's angular velocities. The cart is actuated by two linear stages, applying forces τ_x and τ_y in the x and y directions. The control signal is thus $u \doteq [\tau_x, \tau_y]$.

The equations of motion of the crane are of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ and we assume that we have a way to simulate them accurately at discrete moments in time $t_k = k\Delta t$, $k = 1, 2, \ldots$, where Δt is the control step. One way to obtain a simulation model is to derive the equations of motion in symbolic form using Lagrangian dynamics. An alternative method is to build a simulation model directly in a physics engine using rigid bodies and joints connecting them. We also assume that all components of the crane's state are fully observable.

When learning a minimum-time controller for the crane using reinforcement learning, a suitable reward function needs to be provided to guide the learning process. One option is to formulate the reward directly in terms of the state variables, defining success as entering a goal region for the crane's cart position and load and staying inside it for a specified number of control steps, thus ensuring that the load's oscillations have subsided. However, in a practical deployment, it is the load's own position that matters the most, as the stabilization of this position is what enables the start of the next operation with the load. For this reason, we defined the success criterion in terms of the cart's and load's position, as opposed to using the load angle. Computing the load's position for a given state of the crane is straightforward, if the length of the cable is known, too. The concrete reward functions used in our experiments are described in the next section.

III. Methods and Tools

A. Simulation Model

Simulations were conducted using a model of an overhead crane implemented in the MuJoCo physics engine [12], a tool extensively utilized in studies of robotics and reinforcement learning. The detailed simulation parameters of the crane model are shown in Table I. All controllers operated at a control rate of 25 Hz ($\Delta t = 40$ ms) on the same OpenAI Gym environment simulated in MuJoCo.

TABLE I: MuJoCo Crane Model Specifications

Parameter	Dimensions	Values
Pillar	L, W, H	0.05 m, 0.05 m, 0.6 m
Rail	R, L	0.01 m, 2.0 m
Bridge	R, L	0.01 m, 0.65 m
Cart	L, W, H, M	0.05 m, 0.05 m, 0.05 m, 2 kg
Cable	R, L, M	0.02 m, 0.3 m, 0.1 kg
Load	L, W, H, M	0.05 m, 0.05 m, 0.05 m, 1 kg
Damping	Damping	0.01 Ns/m
Force limit for x axis	Range	[-150, 150] Nm
Force limit for y axis	Range	[-10, 10] Nm

B. Simulation Environment

We used the OpenAI Gym platform [13] to develop a simulation environment that allows controllers to interact with a crane plant. In this environment, the task requires the controller to manage both trajectory planning and stabilization when moving from a given initial location to a random goal location.

2-DoF Crane Environment. The initial phase of our experiment focused on a 2-DoF crane problem, where the cart is restricted to movement along the x axis and the cable can only rotate around the y axis, about its suspension point on the crane's cart. For the purposes of RL training, the observation vector $\mathbf{o}_{2DoF} \doteq [c_x, \theta_x, l_x, v_x, \dot{\theta}_x, g_x]$ is defined

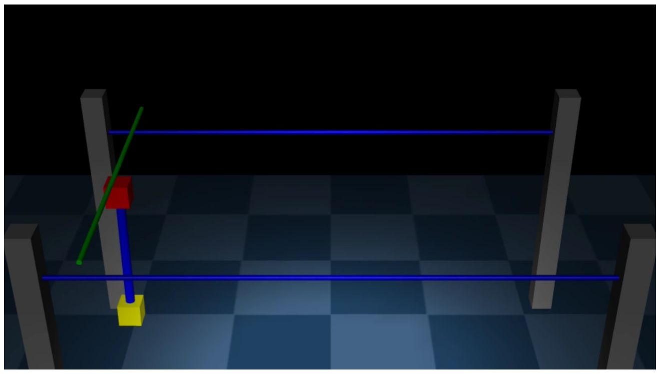

Fig. 1: The MuJoCo model of an overhead crane used in the empirical study.

based on the cart location c_x, load location l_x, cable rotation angle θ_x, cart speed v_x, cable angular velocity $\dot{\theta}_x$, and goal location g_x. (This follows the common practice in RL to include the goal state in the observation vector if the goal is variable, as it will be with most real cranes.) The reduced control consists only of the linear force applied in the x direction, $\mathbf{u}_{2DoF} \doteq \tau_x$.

The controller's objective is to move a load from a random initial position within the range of $[-1, -0.8]$ m to a random goal position within the range of $[-0.6, 0.8]$ m. Given our goal to develop a user-friendly RL-based crane control application, we employ a sparse reward system, which is considered a straightforward design yet presents significant exploration challenges for RL algorithms [14], [15]. Our reward r and termination criteria are based only on the load, cart, and goal locations:

$$\text{r, Term} = \begin{cases} \text{0, False} & \text{if } |c_x - g_x| > 0.1m \text{ or } |l_x - g_x| > 0.1m \\ \text{1, False} & \text{if } |c_x - g_x| \le 0.1m \text{ and } |l_x - g_x| \le 0.1m \\ \text{1, True} & \text{if } |c_x - g_x| \le 0.1m \text{ and } |l_x - g_x| \le 0.1m \text{ for 1 s} \end{cases} \tag{1}$$

In this reward structure, if both the cart and load locations are outside of the goal region, defined as locations less than 0.1 m away from the goal position, the termination condition is False, and the reward is 0. If both are within the goal region, the reward is 1. Furthermore, if the cart and load maintain their position within the goal region for at least 1 second, the episode terminates with a True condition.

4-DoF Crane Environment. We also investigated a more challenging task which allows for both x- and y-axis movement as well as rotation around the respective axes. The action now is fully two-dimensional ($\mathbf{u}_{4DoF} = [\tau_x, \tau_y]$) and the observation now extends to 12 dimensions, as follows:

$$\mathbf{o}_{4DoF} \doteq [c_x, c_y, \theta_x, \theta_y, l_x, l_y v_x, v_y, \dot{\theta}_x, \dot{\theta}_y, g_x, g_y] \tag{2}$$

As with the 2-DoF version, when designing classical controllers, we use the actual full state $\mathbf{x}_{4DoF} \doteq [c_x, c_y, \theta_x, \theta_y, v_x, v_y, \dot{\theta}_x, \dot{\theta}_y]$.

For the 4-DoF problem, the objective for the x axis remains the same as in the 2-DoF problem. The goal along the y axis is to move the load from a random initial position along the y axis within the range of $[-0.3, 0.3]$ m to a random goal position within the range of $[-0.3, 0.3]$ m. Similar to the 2-DoF problem, the sparse reward and termination criterion are only based on load, cart, and goal locations, as follows:

$$\text{r, Term} = \begin{cases} \text{0, False} & \text{if } |c_x - g_x| > 0.1m \text{ or } |l_x - g_x| > 0.1m \\ & \text{ or } |c_y - g_y| > 0.1m \text{ or } |l_y - g_y| > 0.1m \\ \text{1, False} & \text{if } |c_x - g_x| \le 0.1m \text{ and } |l_x - g_x| \le 0.1m \\ & \text{and } |c_y - g_y| \le 0.1m \text{ and } |l_y - g_y| \le 0.1m \\ \text{1, True} & \text{if } |c_x - g_x| \le 0.1m \text{ and } |l_x - g_x| \le 0.1m \\ & \text{and } |c_y - g_y| \le 0.1m \text{ and } |l_y - g_y| \le 0.1m \text{ for 1 s} \end{cases} \tag{3}$$

C. Control Methods

To optimize the transportation and settling times of crane loads, many control methods have been proposed. For this study, we have selected several well-established control methods, including Pole Placement (PP), Linear Quadratic Regulator (LQR), and iterative LQR (iLQR), to serve as baseline methods for comparison with learning controllers based on RL.

1) Control Engineering Methods: Two of the baseline methods, PP and LQR, work with a linearization of the original nonlinear system around a chosen point. An important question is which point to linearize the nonlinear dynamics around. In the crane control problem, a stable equilibrium where the cable is vertical ($\theta_x = \theta_y = 0$) is a suitable choice, because the linearized system remains the same for any cart position and velocity. Furthermore, if the load transportation problem starts from resting state, as is usual in practice, this linearization is valid for both the initial and the goal states. (It will also be valid during very slow motion of the cart, if the cable remains largely vertical, but this motion will likely not be time-optimal.)

Pole Placement (PP) is a traditional control technique for designing a full-state feedback (FSF) controller of the form $\mathbf{u} = -\mathbf{K}\mathbf{x}$, where $\mathbf{K}$ is a matrix of feedback gains. The method consists of strategic placement of the poles of the transfer function of the closed-loop system to achieve a desired response [16]. **LQR** is another popular method for designing an FSF controller for a linear time invariant system that aims to minimize a quadratic cost function balancing feedback error with control cost [16].

A big advantage of the PP and LQR controllers is that they precompute a constant vector of feedback control gains and use them during real-time control with only a very minimal amount of computation. However, their notable disadvantage is that they assume that the controlled system is linear, whereas the crane has nonlinear dynamics. The discrepancy between the linearized and true, nonlinear dynamics is especially pronounced when the load swings far from its vertical position.

To address this problem, trajectory optimization methods based on differential dynamic programming, such as the iterative LQR (**iLQR**) method, compute nominal state and control trajectories for a specified initial state, along with variable feedback control gains whose purpose is to regulate the system's path along the nominal trajectory. The iLQR method solves this trajectory optimization problem very efficiently when the dynamics and stage costs are differentiable [17], [18]. However, the precomputation of the nominal trajectory for a specific initial state is an iterative process that requires a nontrivial amount of computation. Because this computation must be performed only after the initial state has become known, it slows down the operation of the crane, potentially offsetting savings in transportation time resulting from more optimal control.

2) Control Policies Based on Deep Reinforcement Learning: Given the limitations of the classical control design methods described above, it is desirable to design a controller tailored to the true nonlinear system, as opposed to the linearized approximation used by the PP and LQR methods, but one that requires a minimal time to compute the control signal (like these methods and unlike iLQR), so that it can be deployed for real-time control. RL, and in particular recent algorithms for deep RL that employ deep neural networks to represent value functions and control policies, are a strong candidate for such controller design methodology, particularly after recent advances enabling the use of continuous actions.

RL methods often use a state-action value function $Q(\mathbf{x}, \mathbf{u})$ to compute the current policy π as $\mathbf{u}_k = \pi(\mathbf{x}_k)$. It is often the case that instead of the true system state $\mathbf{x}_k$, a vector of observations $\mathbf{o}_k$ is used both in the value function Q as well as in the policy function π. This is possible when the observation $\mathbf{o}_k$ uniquely identifies the system state $\mathbf{x}_k$, and is convenient when the reward function is formulated in terms of the elements of the observation vector.

In this paper, we used the Soft Actor-Critic (SAC) algorithm which has emerged as a competitive DRL algorithm with state-of-the-art performance over continuous control problems [19]. For our experiments, we used the implementation provided in the Stable Baselines3 library [20].

IV. Experimental Results

In this section, our simulation and experimental results are presented. We performed a comprehensive benchmark comparison across all four control methods (PP, LQR, iLQR, RL), on the two versions of the problem (2-DoF and 4-DoF). For the 2-DoF problem, the goal positions were chosen in the range from -0.6 m to 0.8 m at 0.2 m increments, and the initial cart locations were in the set $\{-1, -0.9, -0.8\}$ m (a total of 24 test cases). For the 4-DoF problem, the initial and goal positions along the x axis were the same, whereas the initial y positions were chosen from the set $\{-0.3, -0.1, 0, 0.1, 0.3\}$ m and the goal y positions were in the same set, $\{-0.3, -0.1, 0, 0.1, 0.3\}$ m (for a total of 600 test cases). All simulations were conducted using a Intel i9-12900k CPU and RTX 4090 GPU.

A. Experimental Results in Simulation of a 2-DoF Crane

For the 2-DoF Crane simulation, we used the following hyper-parameters for the control methods:
1. **PP**: Poles are placed at $s = 0.9$.
2. **LQR**: Q matrix: $\text{diag}[48.1, 86.7, 0, 0]$, R matrix: $[0.00001]$.
3. **iLQR**: Q matrix: $\text{diag}[0, 48.1, 86.7, 0, 0]$, Q_T matrix: $\text{diag}[0, 17.34, 17.34, 17.34, 17.34]$, R matrix: $[0.00001]$
Additionally, the hyper-parameters for **SAC** are shown in Table II.

Hyperparameter SAC	Value
Max timesteps	4e5 steps
Batch size	256
Buffer size	1e6
γ	0.99
τ	0.005
Adam Learning rate	3e-4
Number MLP layer	3
Number of hidden neuron	256

TABLE II: Hyper-parameters used by the SAC algorithm

First, we compare the computation time or training time of all the control methods. As shown in Table III, the classical control methods such as PP and LQR compute the controller in a fraction of a second. Since the iLQR is an iterative method, it takes longer, with a computation time of 15.63 seconds. Additionally, the SAC method spends approximately 40 minutes to find the converged controller.

Methods	PP	LQR	iLQR	SAC
Time (s)	0.01	0.02	15.63	2435.3

TABLE III: Computation/training time comparison, 2 DoF

Fig. 2 presents the evaluation of transport time across the 24 benchmark cases. Of the four control methods, RL achieves the fastest average transport time. Although the iLQR method has a slightly slower mean transport time than RL, its standard deviation is smaller.

Fig. 3 displays average transport times vs. the travel distance (i.e., the distance between initial and goal states) and provides further insights into the relative performance of the control algorithms as a function of travel distance. While Pole Placement, LQR, and iLQR always complete transport successfully, their relative performance is highly dependent on the travel distance to be traversed. This is likely due to the fact that their control gains are tuned according to hyperparameters (pole locations for PP and cost matrices for LQR and iLQR) and thus the gains end up being better for some travel distances and worse for others. For instance, the LQR method, with its current hyperparameters, outperforms other methods for short travel distances (less than 0.8 m). Conversely, the iLQR method excels at longer travel distances (greater than 1.0 m). These methods struggle to generalize due to fixed hyperparameters that restrict their adaptability across different scenarios.

In contrast, RL utilizes deep neural networks with thousands of parameters and, through extensive training, can learn a generalized solution that adapts effectively across all travel distances. The adaptability of RL results from its ability to approximate complex state-action relationships and adjust strategies based on learned patterns. This allows RL agents to handle varying travel distances consistently, as they optimize through trial-and-error learning, ultimately identifying a robust policy that accommodates a wide range of scenarios.

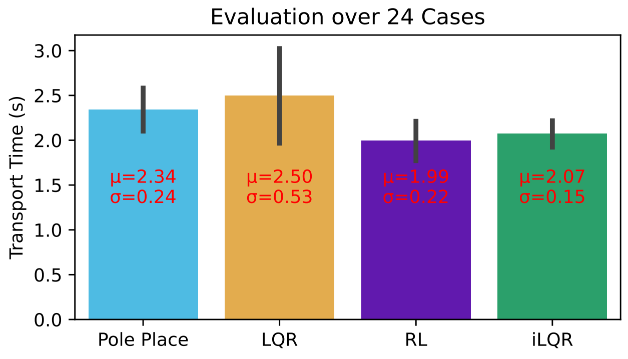

Fig. 2: Transport time over 24 test cases, 2-DoF problem.

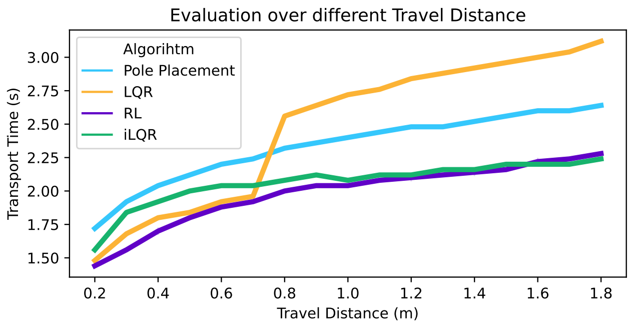

Fig. 3: Transport time over different travel distances, 2-DoF problem.

B. Experimental Results in Simulation of a 4-DoF Crane

For the 4-DoF Crane simulation, we used the following hyper-parameters for the control methods:
1. **PP**: Poles are placed at $s = 0.9$.
2. **LQR**: Q matrix: diag$[164.7, 164.7, 164.7, 164.7, 0, 0, 0, 0]$, R matrix: $[0.00001, 0.00001]$.
3. **iLQR**: Q matrix: diag$[0, 0, 70, 70, 100, 100, 0, 0, 0, 0, 0]$, Q_T matrix: diag$[0, 0, 200, 200, 200, 200, 100, 100, 200, 200, 0]$, R matrix: $[0.001, 0.1]$
The hyper-parameters for **SAC** are shown in Table II.

Methods	PP	LQR	iLQR	SAC
Time (s)	0.01	0.02	17.69	2815.2

TABLE IV: Computation/training time comparison, 4 DoF

First, we compare the computation time or training time of all the control methods. As shown in Table IV, the classical control methods such as PP and LQR compute the controller in less than a second. Since the iLQR is an iterative method, it takes longer, with a computation time of 17.69 seconds. Additionally, the SAC method spends approximately 42 minutes to find the converged controller.

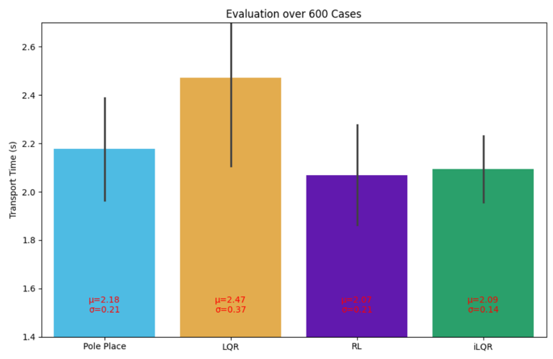

Fig. 4: Transport time over 600 cases, 4-DoF problem.

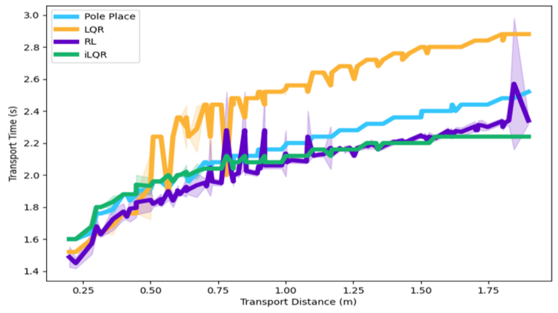

Fig. 5: Transport time over different travel distances, 4-DoF problem. The shaded area shows variations in transport time for the same distance.

Fig. 4 presents the evaluation of transport time across 600 test cases. Similarly to the 2-DoF experiment, out of the four control methods, RL achieves the fastest average transport time. However, compared with the 2-DoF experiment, all methods have larger standard deviation, likely resulting from the more complicated coupled dynamics of the load's movement.

Figure 5 provides further insights into the transport time over varying travel distances. Similar to the 2-DoF problem, the LQR method, with its current hyperparameters, outperforms other methods for short travel distances (less than 0.5 m). Conversely, the iLQR method excels for longer travel distances (greater than 1.3 m). Classical methods like LQR struggle to generalize due to their fixed hyperparameters. Additionally, we observe performance variations in transport time across different combinations of x- and y-axis movements. Among all four methods, RL generally performs well but exhibits four significant spikes, likely due to insufficient exploration from certain initial positions. Pole Placement and iLQR show the smallest variance, while LQR demonstrates significant variance.

These performance variances highlight the challenges faced by linear control methods such as Pole Placement and LQR when dealing with more complex dynamics. While iLQR can perform comparably to RL, it requires recomputation whenever the initial conditions change. Despite the complexity of the dynamics and the associated exploration challenges, the RL controller still performs well overall. This underscores the adaptability of RL, which, through extensive training, can handle diverse initial conditions and travel distances, optimizing performance across a wide range of scenarios.

V. Conclusion and Future Work

In this paper, we presented an experimental study on the application of DRL methods to the problem of time-optimal control of overhead gantry cranes. Experiments in simulation using a physics engine on two versions of the problem, with two and four degrees of freedom, demonstrated that policies trained with the SAC DRL algorithm can achieve up to 20% shorter transport time in comparison with controllers designed by means of more traditional methods from the field of control engineering. Although DRL methods do require an offline control policy training stage, the crane controllers could be obtained within several hours of computation on standard computing hardware. This means that such policies can be easily retrained when the operating conditions change.

The presented study assumed that the length of the crane cable remained the same throughout transportation. In practical deployments, this can be ensured by first hoisting the load to a safe height, and then commencing the transport using the controllers discussed in this paper. However, it might be advantageous to perform hoisting and transportation simultaneously, or even vary the cable length purposefully during transport in order to reduce load oscillations. This more advanced version of the control problem adds effectively a fifth DoF (the length of the cable), and in future work, we plan to investigate if DRL methods could be equally effective for this problem setting, too.

References

[1] E. M. Abdel-Rahman, A. H. Nayfeh, and Z. N. Masoud, "Dynamics and control of cranes: A review," *Journal of Vibration and Control*, vol. 9, no. 7, pp. 863–908, 2003.

[2] J. W. Auernig and H. Troger, "Time optimal control of overhead cranes with hoisting of the load," *Automatica*, vol. 23, no. 4, pp. 437–447, 7 1987.

[3] K. Kudara, H. Takahashi, S. Sasai, H. Sakurai, M. Okubo, H. Nakayama, T. Maedo, and N. Uchiyama, "Time-Optimal Motion Generation and Load-Sway Suppression for Rotary Cranes with a Two-Stage S-Curve Trajectory Based on Skilled Operation Analysis," in *European Control Conference*, Stockholm, 6 2024.

[4] W. Yang, J. Chen, D. Xu, and X. Yan, "Hierarchical global fast terminal sliding-mode control for a bridge travelling crane system," *IET Control Theory & Applications*, vol. 15, no. 6, pp. 814–828, 2021.

[5] J. Abdullah, R. Ruslee, and J. Jalani, "Performance Comparison between LQR and FLC for Automatic 3 DOF Crane Systems," *International Journal of Control and Automation*, vol. 4, no. 4, pp. 163–178, 2011.

[6] M. Faisal, M. Jamil, Q. Awais, U. Rashid, M. S. S. O. Gilani, Y. Ayaz, and M. N. Khan, "Iterative Linear Quadratic Regulator (ILQR) controller for trolley position control of quanser 3DOF Crane," *Indian Journal of Science and Technology*, 2015.

[7] A. Piazzi and A. Visioli, "Optimal dynamic-inversion-based control of an overhead crane," *IEE Proceedings-Control Theory and Applications*, vol. 149, no. 5, pp. 405–411, 2002.

[8] J. A. Mendez, L. Acosta, L. Moreno, A. Hamilton, and G. N. Marichal, "Design of a neural network based self-tuning controller for an overhead crane," in *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No. 98CH36104)*, vol. 1. IEEE, 1998, pp. 168–171.

[9] G. Eaglin, T. Poche, and J. Vaughan, "Controlling a Double-Pendulum Crane by Combining Reinforcement Learning and Conventional Control," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 788–793.

[10] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 378–383.

[11] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, and R. Ribas, "Solving Rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[12] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[14] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International conference on machine learning*. PMLR, 2018, pp. 4344–4353.

[15] J. Zhong, R. Wu, and J. Si, "A long n-step surrogate stage reward for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 36. Curran Associates, Inc., 2023, pp. 12 733–12 745.

[16] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Prentice hall Upper Saddle River, 2015.

[17] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *First International Conference on Informatics in Control, Automation and Robotics*, vol. 2. SciTePress, 2004, pp. 222–229.

[18] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[19] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.