Underactuated Waypoint Trajectory Optimization for Light Painting Photography

Christian Eilers*, Jonas Eschmann*, Robin Menzenbach*,
Boris Belousov, Fabio Muratore, and Jan Peters

***Abstract*— Despite their abundance in robotics and nature, underactuated systems remain a challenge for control engineering. Trajectory optimization provides a generally applicable solution, however its efficiency strongly depends on the skill of the engineer to frame the problem in an optimizer-friendly way. This paper proposes a procedure that automates such problem reformulation for a class of tasks in which the desired trajectory is specified by a sequence of waypoints. The approach is based on introducing auxiliary optimization variables that represent waypoint activations. To validate the proposed method, a letter drawing task is set up where shapes traced by the tip of a rotary inverted pendulum are visualized using long exposure photography.**

I. Introduction

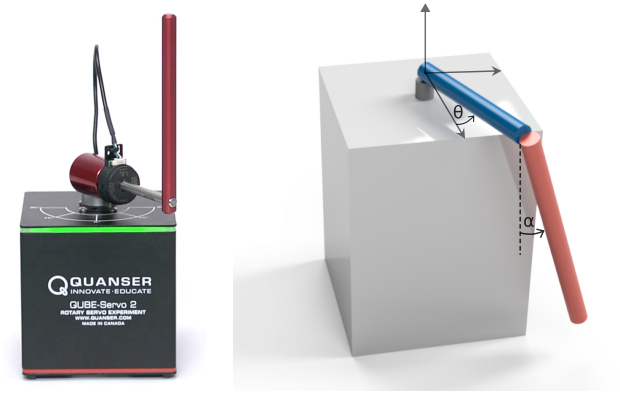
Controlling underactuated systems is of special interest in robotics and engineering because many common systems such as automobiles, hovercrafts, aircrafts, ships, legged and wheeled robots, as well as underwater vehicles are underactuated [1]. Nevertheless, designing efficient controllers for such systems requires significantly more effort than for fully actuated ones [2]. In particular, even if a feasible trajectory is obtained in simulation, trajectory tracking on a real system is non-trivial because not all deviations from the desired trajectory can be compensated due to the underactuation [3].

Although techniques such as partial feedback linearization [4], which aim to cancel the system dynamics, can be effective at reducing the plant to a partially linear form, they do not exploit the passive system dynamics [3]. For classical control systems, such as the cart-pole, convey-crane, pendubot, etc., a number of controllers have been hand-designed [5] that do exploit the system dynamics. The task in those examples is typically to drive the system to an equilibrium state. In this paper, on the other hand, we are interested in generating and tracking a dynamic trajectory rather than reaching a static target state.

Trajectory generation for both actuated and underactuated systems is commonly performed using numerical optimization [3]. For fully actuated systems, waypoints are relatively straightforward to incorporate into the trajectory generation process because kinematic path planners can be used [6]. For underactuated systems, however, a kinematic plan may be dynamically infeasible [7]. Therefore, a dynamics-based trajectory optimization method is needed that can handle trajectory specification in the form of a sequence of waypoints.

*Authors contributed equally name.surname@stud.tu-darmstadt.de
All authors are with the Computer Science Department, Technische Universität Darmstadt, Germany name.surname@tu-darmstadt.de

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640554.



(a) Quanser Qube. (b) Coordinate systems.

Fig. 1: Evaluation platform: Furuta pendulum.

The main contribution of this paper is the design of the objective function for trajectory optimization described in Section III. This objective function explicitly takes into account the waypoints and thus enables the generation of recognizable letter contours in long exposure photography. However, open-loop execution of the optimal trajectory is not sufficient on its own because even small deviations from the planned trajectory yield unrecognizable letters. Section IV details the implementation of the stabilizing feedback controller that enables efficient trajectory tracking. Finally, the resulting trajectories and letters are presented in Section V.

II. Light Painting Setup

The hardware platform used for experiments is the Quanser Qube shown in Fig. 1a. It implements the rotary inverted pendulum system introduced by Furuta et al. [8], which consists of a freely rotating pendulum attached to a motor-driven arm. A schematic is shown in Fig. 1b. While the arm can be rotated in the horizontal plane, the pendulum swings in the vertical plane orthogonal to the arm. The state of the nonlinear system is described by the two angles and the corresponding angular velocities

$$\mathbf{x} = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T .$$

The Furuta pendulum is a classical platform for evaluating control algorithms, appreciated for its rich passive dynamics and underactuation. Its equations of motion are provided in the Appendix, with derivations starting from the Euler-Lagrange equations available in [8] and [9].

The light painting task is set up as follows. A piece of reflective tape is attached to the tip of the Furuta pendulum.

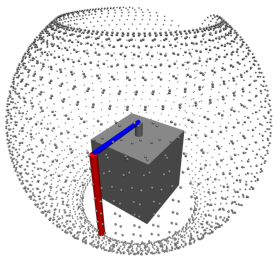


Fig. 2: Reachable space visualized as a point cloud.

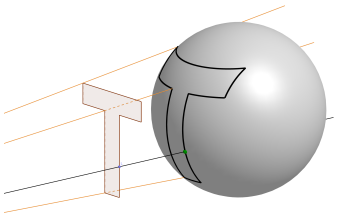


Fig. 3: Projection of a letter onto the reachable space.

While the pendulum is moving, a long exposure photograph is taken. The goal is to draw recognizable letters with the tip of the pendulum. Since the reachable space of the Furuta pendulum covers a part of a sphere, as shown in Fig. 2, all letters first need to be projected onto the reachable space before drawing, as depicted in Fig. 3.

Ideally, we would like to have a controller that receives a letter as input and is able to trace its contour with the tip of the pendulum. If the system was fully actuated, trajectory tracking would be straightforward. However, due to underactuation, not every trajectory can be executed but only dynamically feasible ones. Therefore, the crucial task is to find a trajectory which most closely follows the shape of the desired letter. As a proxy for this task, we discretize the letter into a sequence of waypoints and subsequently search for a trajectory that passes through these waypoints.

The diagram in Fig. 4 shows the full pipeline of our approach to underactuated light painting. On a high level, it can be split into three parts, from top to bottom: *waypoint generation* (first row), *trajectory optimiztion* (rows 2–3), and *execution* (bottom 3 rows). Waypoint generation comprises letter discretization and projection discussed above. Trajectory optimization takes the generated waypoints as input and finds a sequence of control commands that drives the system through these waypoints using the knowledge of the system kinematics and dynamics. Finally, an LQR feedback controller is added for tracking of the optimized trajectory at the execution stage. Additionally, a synchronized set of LEDs is activated when the pendulum passes through the trajectory segments belonging to the letter to increase illumination.

III. Trajectory Optimization

Given a set of waypoints obtained via letter discretization and subsequent projection onto the reachable space, we aim to devise an objective function that will yield a trajectory passing through the waypoints upon optimization. To this end, we first describe the trajectory optimization method which we employ in Section III-A. After that, we present the main idea of our approach of introducing 'attention' into the optimization objective and explain it on the task of reaching a single desired waypoint in Section III-B. Finally, in Section III-C, we demonstrate how the idea of introducing 'attention' can be extended to multiple waypoints and how to enforce a desired ordering among them.

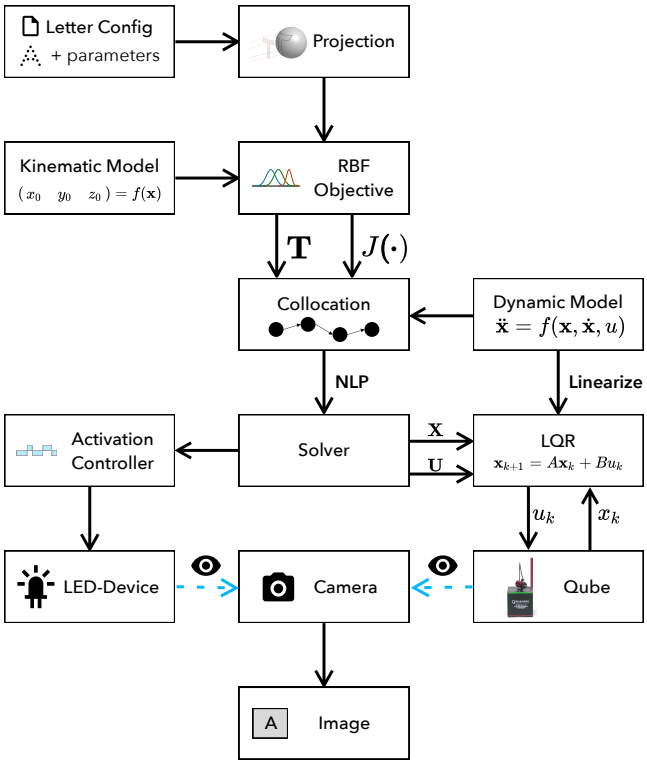


Fig. 4: Full pipeline from planning to tracking for drawing letters via light painting photography using Furuta pendulum.

A. Direct Collocation

Trajectory optimization is concerned with finding a feasible trajectory that minimizes a given objective function. Numerical optimization methods such as multiple shooting and direct collocation work by transforming a continuous-time optimal control problem into a big Nonlinear Program (NLP) [10]. Methods differ in how exactly the discretization is done and what variables are treated as optimization variables. We use direct collocation with cubic splines [11], widely spread in robotics [3], and implement our optimization problem in CasADi [12].

Direct collocation treats both states $\mathbf{x}_t$ and control commands $\mathbf{u}_t$ as optimization variables,

$$\mathbf{X} = \begin{bmatrix}\mathbf{x}_0 & \dots & \mathbf{x}_N\end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix}\mathbf{u}_0 & \dots & \mathbf{u}_{N-1}\end{bmatrix},$$

whereas the system dynamics are imposed as constraints. The objective function typically has the form of a sum over the time steps

$$J(\mathbf{X}, \mathbf{U}) = \underbrace{\sum_{t=0}^{N} \alpha_t d(\hat{\mathbf{x}}, \mathbf{x}_t)}_{J_\alpha(\mathbf{X})} + \underbrace{\beta \sum_{t=0}^{N-1} \mathbf{u}_t^2}_{J_\beta(\mathbf{U})} \tag{1}$$

where $d(\hat{\mathbf{x}}, \mathbf{x}_t)$ is a distance-based metric that encodes the state-dependent part of the running cost. Weights $\alpha_t \in [0, 1]$ determine the importance of each time step and are usually set to $\alpha_t = 1$. Parameter β is chosen such that the cost of the squared control commands is orders of magnitude smaller than the other cost terms. Moreover, we introduce $\hat{\mathbf{x}}$ as a parameter, which will later play the role of a waypoint.

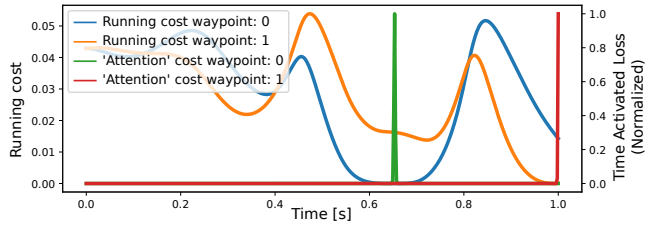


Fig. 5: Effect of 'attention' on the loss function. Components of the loss function are drawn over time for a task with two waypoints. Note that the loss is minimal and close to zero when 'attention' is one.

Our key idea is to parameterize the weights α_t in a specific way that draws the 'attention' of the optimizer to the important moments in time when the waypoints need to be reached. Crucially, which moments exactly are important is determined by the optimizer itself. In the following, we detail how this is done, first on a single-waypoint example and then on the full sequential problem.

B. Attention Mechanism for Reaching a Single Waypoint

If there is only one point $\hat{\mathbf{x}}$ that needs to be reached, the coefficients α_t in Eq. (1) can be set as

$$\alpha_t = \begin{cases} 1, & t = N, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

which puts all the weight on the last time step and yields a trajectory that ends up at the target state. At first sight, one could imagine solving a set of such one-waypoint problems and then chaining the solutions together to obtain a complete trajectory. However, this approach will not work, because it does not account for the fact that the final state of one segment becomes the initial condition for the subsequent one. Since the dynamics are nonlinear and the system is underactuated, the optimizer may decide to e.g. do an additional swing between going from one waypoint to another, despite the points being next to each other, just because the velocity with which the first waypoint was reached was not sufficiently high. Moreover, switching controllers between segments is non-trivial and leads to jerky transitions. Therefore, we aim for developing a method that allows to pass through multiple waypoints smoothly instead.

An approach based on Eq. (2), where the activation time is trivially set to the last time step, is hardly scalable to multiple waypoints, as the activation time for each point would have to be known in advance. Setting the activation times for multiple waypoints by hand is prohibitive and in general leads to suboptimal solutions. This can be attributed to the underactuated and oscillating nature of the Furuta pendulum, which makes it hard to anticipate how much swinging is needed to accumulate sufficient energy for reaching certain states.

For long exposure photography, it does not matter at what exact time the system passes through each waypoint. This renders hard-coded activations such as in Eq. (2) unnecessary and motivates a more flexible approach. Namely, instead of pre-specifying the activations α_t, we treat them as optimization variables. More concretely, we parameterize the coefficients α_t by Radial Basis Functions (RBFs) of the form

$$\alpha_t = \exp\left(-\frac{(\hat{t} - t)^2}{\sigma^2}\right) \tag{3}$$

where $\hat{t}$ is the center of the RBF and σ is the bandwidth. The center $\hat{t}$ determines the activation time and is introduced as a new optimization variable in the NLP. Thus, the optimizer is able to shift its 'attention' and can account for the time needed to accumulate sufficient energy to reach a desired state. Inserting Eq. (3) into Eq. (1), we obtain the objective function that incorporates 'attention' for a single waypoint.

$$J_\alpha(\mathbf{X}, \hat{t}) = \sum_{t=0}^{N} \exp\left(-\frac{(\hat{t} - t)^2}{\sigma}\right) d(\hat{\mathbf{x}}, \mathbf{x}_t). \tag{4}$$

To exclude trivial solutions achieved by shifting the attention out of the scope of the finite trajectory, $\hat{t}$ needs to be constrained to the interval $[0, N]$.

This formulation also allows one to minimize the time of arrival at the waypoint by simply adding a punishment term $\gamma\hat{t}$ to the objective function in Eq. (4) with some positive weight γ. The main advantage of this approach is its independence on pre-specified activation times, which also makes it scalable to multiple waypoints.

C. Attention for Reaching Multiple Waypoints in Sequence

Extending Eq. (4) with an activation time $\hat{t}_i$ for each waypoint $\hat{\mathbf{x}}_i$ and summing over the waypoints, we obtain the objective function for multiple waypoints

$$J_\alpha(\mathbf{X}, \mathbf{T}) = \sum_{i=0}^{M-1} \sum_{t=0}^{N} \exp\left(-\frac{(\hat{t}_i - t)^2}{\sigma}\right) d(\hat{\mathbf{x}}_i, \mathbf{x}_t) \tag{5}$$

where M is the number of waypoints and $\mathbf{T}$ is the set of their associated activation times $\hat{t}_i$.

The order in which the waypoints are traversed matters: if the waypoints are traversed in an arbitrary order, the drawn letters are hardly recognizable. However, the ordering is not enforced by the objective function in Eq. (5). To impose order, we augment the optimization problem with constraints of the form $\hat{t}_i \leq \hat{t}_{i+1}$, $i = 0, \ldots, M-2$. Furthermore, it is beneficial to split up the set of the waypoints into segments. All segments are then treated within one NLP, but the ordering constraints are only enforced within each segment. To further improve the smoothness of the trajectory, we add a punishment term

$$J_\mu(\mathbf{T}) = \mu \sum_{j=0}^{S-1} \left(\hat{t}_{l_j} - \hat{t}_{f_j}\right) \tag{6}$$

to the objective function that favors short segments $\hat{t}_{l_j} - \hat{t}_{f_j}$. Here, f_j and l_j denote the first and last waypoints in segment j, respectively. Each letter is split into S segments and μ determines the strength of the segment duration punishment. The resulting objective function for multiple segments is given by

$$J(\mathbf{X}, \mathbf{U}, \mathbf{T}) = J_\alpha(\mathbf{X}, \mathbf{T}) + J_\beta(\mathbf{U}) + J_\mu(\mathbf{T}). \tag{7}$$

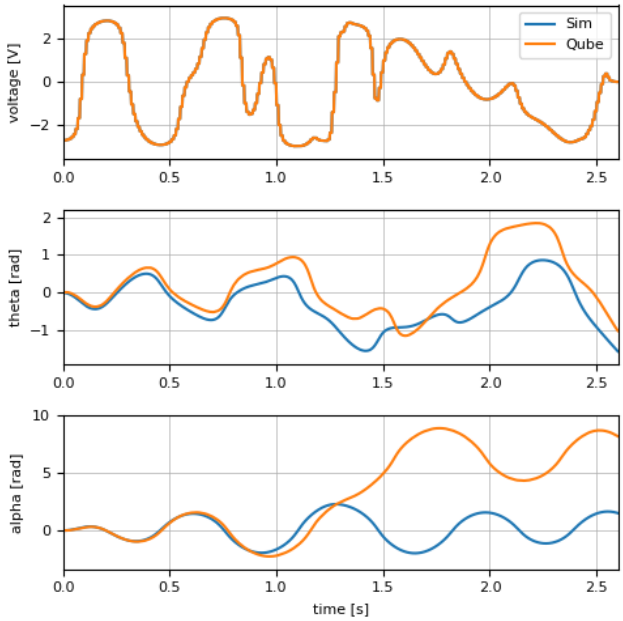


Fig. 6: Open-loop control on the Quanser Qube: (top) motor input voltage, (middle) horizontal joint angle θ, (bottom) pendulum joint angle α, plotted over time. Trajectory in blue (Sim) was optimized in simulation to trace letter 'S' as shown in Fig. 8. Trajectory in orange (Qube) was obtained on the real system, and it rather quickly diverges from simulation.

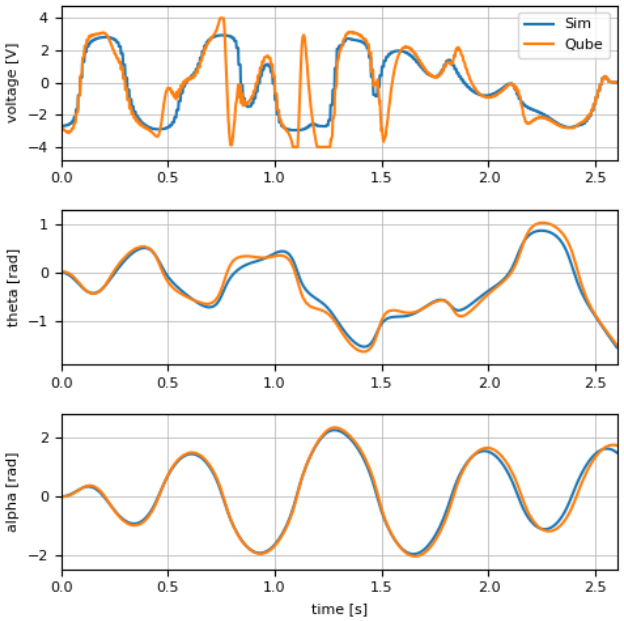


Fig. 7: Closed-loop LQR tracking controller successfully tracks the same desired trajectory as in Fig. 6, and it is able to correct the deviations from the planned trajectory despite underactuation. Undesirable overshoots in the input voltage, owing to exploits of the linearized system dynamics, are clipped to prevent excessively large control signals.

The NLP is then solved by minimizing the objective Eq. (7) subject to the collocation constraints on the system dynamics, path and boundary constraints, and the proposed activations ordering constraints.

Fig. 5 illustrates the effect of 'attention' on the loss function. The results were obtained by optimizing the objective given in Eq. (5) with two waypoints. The curves correspond to the individual terms $d(\hat{\mathbf{x}}_i, \mathbf{x}_t)$ and α_t^i for each of the two waypoints $i = 0$ and $i = 1$. Thus, the value of the full loss is given by the sum of the terms $\alpha_t^i d(\hat{\mathbf{x}}_i, \mathbf{x}_t)$ over t and i. Notably, when 'attention' rises to one, the corresponding distance-based loss goes to zero, signalling that the waypoint is reached. Summing the losses up without time activations would yield a high value for the total cost, despite both waypoints being reached (indicated by the loss going to zero once for each waypoint). Therefore, a formulation with a flat weighting $\alpha_t = 1$ for all time steps, as it is used in most of the literature, would yield a high loss value despite the desired states being reached. In contrast, the RBF-based objective function in Eq. (5), which only accumulates the distance-based losses close to the waypoints, results in a much lower loss value.

IV. Linear-Quadratic Optimal Tracking

Executing an open-loop sequence of control commands on the real system results in a trajectory rather quickly diverging from the desired path due to disturbances, modeling errors, and uncertainties in the initial conditions. An example is shown in Fig. 6. To prevent such divergence and to keep the system on the desired trajectory, we employ an LQR tracking controller described in the following.

The first step is to linearize the system dynamics along a desired trajectory. If $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, then the linearization around a given point $(\mathbf{x}^d, \mathbf{u}^d)$ can be written as

$$\dot{\mathbf{x}} \approx \dot{\mathbf{x}}^d + \frac{\partial \mathbf{f}(\mathbf{x}^d, \mathbf{u}^d)}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x}^d) + \frac{\partial \mathbf{f}(\mathbf{x}^d, \mathbf{u}^d)}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}^d). \quad (8)$$

Performing such linearization at every time step, we can obtain a linearization around the desired trajectory. It is convenient to introduce auxiliary variables representing the deviations from the desired trajectory

$$\tilde{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}_t^d, \qquad \tilde{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}_t^d. \quad (9)$$

Inserting Eq. (9) into Eq. (8), we obtain

$$\dot{\tilde{\mathbf{x}}}_t = \frac{\partial \mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)}{\partial \mathbf{x}} \tilde{\mathbf{x}}_t + \frac{\partial \mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)}{\partial \mathbf{u}} \tilde{\mathbf{u}}_t. \quad (10)$$

Discretizing the continuous-time linear dynamical system given in Eq. (10) using the Euler integration scheme

$$\tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t + \Delta t \dot{\tilde{\mathbf{x}}}_t, \quad (11)$$

we arrive at the discrete-time time-varying dynamics

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1} &= \left(\mathbf{I} + \Delta t \frac{\partial \mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)}{\partial \mathbf{x}} \right) \tilde{\mathbf{x}}_t + \Delta t \frac{\partial \mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)}{\partial \mathbf{u}} \tilde{\mathbf{u}}_t \\ &= \mathbf{A}_t \tilde{\mathbf{x}}_t + \mathbf{B}_t \tilde{\mathbf{u}}_t. \end{aligned} \quad (12)$$

These dynamics provide the basis for designing a time-varying tracking feedback controller.

Given the linearized model along the trajectory in Eq. (12), we can formulate the trajectory stabilization problem as the minimization of the cost

$$J = \sum_{t=0}^{N-1} \left(\tilde{\mathbf{x}}_t^T \mathbf{Q} \tilde{\mathbf{x}}_t + \tilde{\mathbf{u}}_t^T \mathbf{R} \tilde{\mathbf{u}}_t \right). \tag{13}$$

The system is quadratically penalized for being away from the desired trajectory using weighting matrices $\mathbf{Q}$ and $\mathbf{R}$. The optimal feedback controller that minimizes the cost given in Eq. (13) subject to the dynamics provided in Eq. (12) is an affine control law of the form

$$\mathbf{u}_t = \mathbf{u}_t^d - \mathbf{K}_t \tilde{\mathbf{x}}_t \tag{14}$$

where the feedback gain matrix $\mathbf{K}_t$ is found by solving the discrete-time Riccati equation backwards in time [13].

The result of applying the stabilizing LQR controller derived in Eq. (14) to the same trajectory on which the open-loop execution failed is shown in Fig. 7. As it can be seen from the plots, the system is able to follow the desired trajectory, canceling all disturbances and deviations, in spite of being underactuated.

Notwithstanding its impressive performance, the LQR as a tracking controller for the Furuta pendulum has some limitations. First, the controller can only stabilize the system when it is sufficiently close to the desired trajectory. Due to underactuation, the envelope of correctable deviations is quite small. Second, due to high nonlinearity of the dynamics, linearizations can be rather bad in some states, leading to overshooting and instability. As the LQR has no natural way of incorporating control constraints, the applied control voltages were clipped.

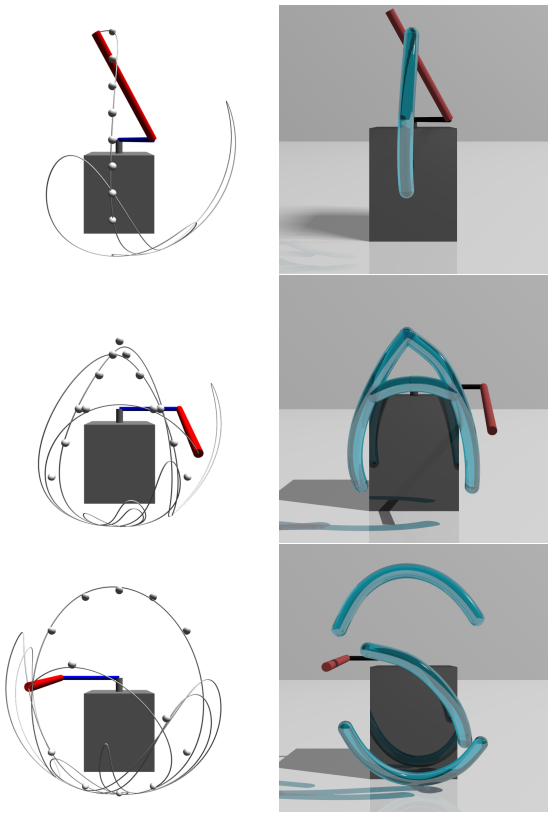
Another general problem of the LQR is the choice of the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$, which are typically found using prior knowledge or trial-and-error. We were able to find good parameters for the presented examples, but as generated trajectories for different letters show significant variability, a tailored set of parameters is required for each letter. A similar problem is stated in [14]. Finding a good set of parameters without many trials is still an open research area and could be the subject for future work, potentially solved by learning or optimization algorithms such as [15].

V. Results

In the previous sections, individual blocks from the pipeline in Fig. 4 have been introduced. In this section, the complete approach is evaluated and the resulting light painted trajectories are presented.

The Quanser Qube implementation of the Furuta pendulum imposes a hard limit on the range of values that the horizontal rotary angle θ can take, reflected in the reachable space shown in Fig. 2. In addition, a software limit is imposed on the input voltage signal u to avoid damaging the motor. To account for the joint and control limits, the following inequality constraints

$$-u_{\max} \leq u_t \leq u_{\max}, \qquad -\theta_{\max} \leq \theta_t \leq \theta_{\max}$$



(a) Trajectory trace. (b) Light painted shape.

Fig. 8: Pendulum trajectories and corresponding light painted letter shapes. Each highlighted segment on the right consists of a set of waypoints traversed in quick succession. Note that the complete trajectories may be quite long, as seen in the traces on the left, and it is virtually impossible to design such trajectories by hand or using kinematic path planning.

are added to our direct collocation formulation of the trajectory optimization problem described in Section III.

For all of our experiments, the initial state $\mathbf{x}_0$ is assumed to be zero, which corresponds to the system being still, with the pole centered in the front and hanging down.

Following the pipeline from Fig. 4, the trajectories shown in Fig. 8 were obtained in simulation. The traces on the left show that a significant amount of time is spent in preparation of each maneuver, while the pendulum is accumulating the required energy and momentum to pass through the waypoints in the specified order and in quick succession. The visualizations on the right show the expected results from the light painting photography, where the letter segments are highlighted based on the activation times $\hat{t}_i$ obtained through the 'attention'-augmented trajectory optimization described in Section III-C. Note that while letter 'I' consists of a single segment, letters 'A' and 'S' are comprised of three segments each. The letter 'S' is specially challenging because of the kinodynamic structure of the Furuta pendulum.

Long exposure photographs of the light painted letters are presented in Fig. 9. The pictures have been taken in a dark room with an LED device synchronized with the trajectory execution and activated based on the optimized

Fig. 9: Images of letters 'I', 'A', and 'S' created by light painting photography following the pipeline from Fig. 4.

segment beginning/end times $\hat{t}_i$ described in Section III-C. Comparing the real images in Fig. 9 with the simulated renderings in Fig. 8, we observe a sufficiently good match allowing the letters to be well recognizable. However, the trajectories slightly deviate towards the end, as it can be seen on the middle strokes in the letters 'A' and 'S' that are drawn last. These segments are slightly tilted compared to their desired location. For a better view, see the accompanying video, where real and simulated trajectories are drawn side by side.

VI. Discussion and Conclusion

A method for objective function design in the context of trajectory optimization with waypoints has been presented (see Section III). The proposed objective function (see Eq. (5)) features an RBF-smoothed 'attention' over time that activates the distance-based loss when the corresponding waypoint is near. Crucially, the RBF-activations are not hand-designed but jointly optimized together with the states and control commands. For the tasks in which the order of the waypoints matters, the objective function has been extended to enforce the desired ordering (see Eq. (7)).

The proposed method has been evaluated on a task of drawing letters with the Furuta pendulum, a highly dynamic underactuated system (see Section V). The letters were discretized into a set of waypoints, and a trajectory passing through them was optimized using the proposed objective function. This procedure yielded activation times at which the waypoints were reached as a byproduct (see Fig. 5). An LQR-based tracking controller has been applied to execute the planned trajectories (see Section IV). To visualize the trajectory traces, long exposure photography has been employed, with an LED ring illuminating the scene at the activation times obtained through optimization (see Fig. 9).

Although the desired performance has been achieved, several improvements are possible. First, the letter segmentation and discretization process should be automated. Second, the complexity of waypoint optimization needs to be evaluated in more depth; we used between 5 and 15 waypoints, but larger numbers may be required in other tasks. Finally, parameters such as time horizon, waypoints order, segment duration penalty, as well as the tracking LQR cost matrices are currently set by hand for each letter. Automating this procedure would be of great practical interest even beyond the light painting task.

Appendix

Equations of motion of the Quanser Qube are given by

$$\left(m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos^2\alpha + J_r\right)\ddot{\theta} + \left(\frac{1}{2} m_p L_p L_r \cos\alpha\right)\ddot{\alpha} + \left(\frac{1}{2} m_p L_p^2 \sin\alpha\cos\alpha\right)\dot{\theta}\dot{\alpha} - \left(\frac{1}{2} m_p L_p L_r \sin\alpha\right)\dot{\alpha}^2 + D_r\dot{\theta} = \frac{k_m(u - k_m\dot{\theta})}{R_m},$$

$$\left(\frac{1}{2} m_p L_p L_r \cos\alpha\right)\ddot{\theta} + \left(J_p + \frac{1}{4} m_p L_p^2\right)\ddot{\alpha} + D_p\dot{\alpha} - \left(\frac{1}{4} m_p L_p^2 \cos\alpha\sin\alpha\right)\dot{\theta}^2 + \frac{1}{2} m_p L_p g \sin\alpha = 0.$$

The control command u (see upper Eq.) is the motor voltage. The dynamics parameters can be found in [16].

References

[1] M. W. Spong, "Underactuated mechanical systems," in *Control problems in robotics and automation*. Springer, 1998, pp. 135–150.

[2] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[3] R. Tedrake. Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for MIT 6.832). [Online]. Available: http://underactuated.mit.edu/

[4] M. W. Spong, "Partial feedback linearization of underactuated mechanical systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 1994, pp. 314–321.

[5] I. Fantoni and R. Lozano, *Non-linear control for underactuated mechanical systems*. Springer Science & Business Media, 2001.

[6] J. Lin, N. Somani, B. Hu, M. Rickert, and A. Knoll, "An efficient and time-optimal trajectory generation approach for waypoints under kinematic constraints and error bounds," in *International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 5869–5876.

[7] J. Schultz and T. Murphey, "Trajectory generation for underactuated control of a suspended mass," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 123–129.

[8] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[9] B. S. Cazzolato and Z. Prime, "On the dynamics of the furuta pendulum," *Journal of Control Science and Engineering*, vol. 2011, p. 3, 2011.

[10] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[11] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[12] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2019.

[13] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[14] A. W. Divelbiss and J. T. Wen, "Trajectory tracking control of a car-trailer system," *IEEE Transactions on Control systems technology*, vol. 5, no. 3, pp. 269–278, 1997.

[15] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 270–277.

[16] Quanser Inc., "Qube-servo 2 workbook - student version," 2013.