Control Engineering Practice 126 (2022) 105255

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Fast trajectory planning and control of a lab-scale 3D gantry crane for a moving target in an environment with obstacles

M.N. Vu [a,*], A. Lobe [b], F. Beck [a], T. Weingartshofer [a], C. Hartl-Nesic [a], A. Kugi [a,b]

[a] Automation & Control Institute, TU Wien, Vienna, Austria
[b] Center for Vision, Automation & Control, Austrian Institute of Technology GmbH, Vienna, Austria

ARTICLE INFO

Keywords:
Trajectory optimization
Motion planning
Obstacle avoidance
Model predictive control

ABSTRACT

In this work, the real-time optimal trajectory planning, together with a cascaded tracking controller, is presented for a three-dimensional (3D) gantry crane in an environment with static obstacles and a dynamically moving target considering dynamic constraints and control input limits. State-of-the-art trajectory optimization-based approaches require long computation times and cannot quickly respond to changes in the target state. The focus of this paper lies on a novel trajectory planning algorithm, which consists of two steps. First, an offline trajectory planner is implemented to compute a time-optimal, collision-free, and dynamically feasible trajectory database that connects all possible initial states of the gantry crane from a predefined starting subspace to the target states in a target subspace. Second, based on linear constrained quadratic programming, the online trajectory replanner makes use of this trajectory database to generate an optimal trajectory in real time that accounts for all changes in the target state. Additionally, a trajectory tracking controller is developed to take into account the dynamic constraints of the gantry crane and to compensate for possible model inaccuracies, disturbances, and other non-modeled effects. Both simulation and experimental results are presented to demonstrate the performance of the proposed trajectory (re)planning algorithm and the control concept.

1. Introduction

Gantry cranes are important robotic systems for automated transportation and manufacturing processes in a broad range of fields such as the steel industry and the construction sector. Gantry cranes are mostly used to move payloads from a starting point to a destination in factories, warehouses, and ports. Typically, gantry crane systems are controlled by an experienced operator. Due to challenging requirements, such as time-optimal operation, high positioning and tracking accuracy of the payload, and safety aspects, path planning strategies and control technologies for this type of systems have been thoroughly studied in the literature. A gantry crane constitutes a nonlinear and underactuated mechanical system. To achieve the goals of accurate positioning and minimal payload oscillations simultaneously, many concepts are based on a combined strategy in which an optimal trajectory is carefully planned first offline, see, e.g., Chen et al. (2016) and Kolar et al. (2017), and then the tracking controller is designed to follow this optimal trajectory, see, e.g., Lobe et al. (2018) and Lu et al. (2021). Although these combined strategies have been successfully applied to real-world scenarios, the trajectory of the gantry crane is typically computed offline in an obstacle-free environment and the control input limits are neglected.

In the literature, the differential flatness property, see, e.g., Fliess et al. (1995) for the concept of flatness, of 2D and 3D gantry crane models is often exploited, where all the system states and inputs can be parameterized by the flat output and its time derivatives. By replacing the flat output with its desired trajectory, the parameterization of the control input directly yields the feedforward control law, see, e.g., Blajer and Kołodziejczyk (2007), Chen et al. (2019) and Kim et al. (2021). In this context, the constraints of the system state and the control input are often neglected due to the complex nonlinear mapping function between the flat output and the system variables. Unlike flatness-based trajectory planning, optimization-based methods can be used to find a locally optimal trajectory given the dynamic constraints of the system, see, e.g., Betts (1998) and Rao (2009). By discretizing the state trajectory of the gantry crane, the full state of the system can be limited to admissible ranges in the optimization problem, including the sway angles and the angular velocities of the payload. Furthermore, an obstacle avoidance strategy can be incorporated into the optimization-based trajectory planning by an additional term in the cost function or in form of additional constraints, see, e.g., Schulman et al. (2014) and Zucker et al. (2013). Such concepts are successfully utilized in

* Corresponding author.
E-mail addresses: vu@acin.tuwien.ac.at (M.N. Vu), amadeus.lobe@ait.ac.at (A. Lobe), beck@acin.tuwien.ac.at (F. Beck), weingartshofer@acin.tuwien.ac.at (T. Weingartshofer), hartl@acin.tuwien.ac.at (C. Hartl-Nesic), kugi@acin.tuwien.ac.at (A. Kugi).

https://doi.org/10.1016/j.conengprac.2022.105255
Received 21 February 2022; Received in revised form 24 May 2022; Accepted 18 June 2022
Available online 30 June 2022

many applications (Iftikhar et al., 2019; Zhang et al., 2020). Here, the advantage of optimization-based trajectory planning over flatness-based methods becomes noticeable in terms of obstacle avoidance, sway suppression, and compliance with state constraints and control input limits. However, the computational time of optimization-based trajectory planning is still too long for a real-time implementation on a standard electronic control unit, see, e.g., Iftikhar et al. (2019) and Zhang et al. (2020).

In scenarios with repetitive tasks, the optimization-based trajectory planning can be solved in a computationally more efficient way. For example, if the starting and/or target states are only changed slightly, only the deviations from the previous trajectory need to be computed instead of running the full optimization. For this reason, trajectory replanning algorithms are developed using dynamic motion primitives (DMPs) or Gaussian Mixture Models (GMMs), see, e.g., Ijspeert et al. (2002) and Khansari-Zadeh and Billard (2011). However, the computation of GMMs and DMPs becomes inefficient in a high-dimensional state space. For the considered lab-scale 3D gantry crane, the dimension of the state space is 10. Moreover, GMM- and DMP-based approaches typically neglect the state constraints and control input limitations. Therefore, these concepts are not suitable for the considered application.

Together with the trajectory planning, the trajectory tracking controller also plays an important role in a gantry crane system. In the literature, the mathematical model of a gantry crane is often decomposed into the slow pendulum subsystem and the fast subsystem which contains the dynamics of the trolley and the hoist drum, see, e.g., Kolar et al. (2017) and Lobe et al. (2018). Thus, the trajectory tracking controller typically relies on a cascaded structure, in which the outer loop of the cascaded trajectory tracking controller keeps the unactuated angles of the hoist cable around the desired trajectory. The inner control loop provides tracking of the desired payload trajectory, see, e.g., Abdullahi et al. (2020), Kolar et al. (2017) and Lobe et al. (2018). In these works, the system state and control input constraints are not directly considered. Thus, there is room for improving the cascaded controller design by using a control scheme that systematically accounts for these constraints. This feature of the controller is particularly important for the system to navigate around obstacles and for suppressing sway of the payload.

In our previous work (Vu et al., 2020), a two-stage fast motion planning algorithm for a lab-scale 3D gantry crane was proposed for the application of moving goods or materials from a predefined starting position to a predefined target position in a static environment with known obstacles. Note that the concept proposed in Vu et al. (2020) was only validated in simulations. Moreover, the online replanning in the second stage is only applicable to a static scenario where the target position does not change during the operation of the gantry crane. This paper significantly extends this previous work in three main contributions:

- First, the fast trajectory planning algorithm not only considers a stationary scenario, but is also able to systematically generate trajectories when the target state changes during the movement of the 3D gantry crane, i.e. the proposed concept is suitable for moving targets. To achieve this goal, a new method for building the offline trajectory database is proposed along with a fast search algorithm. This allows a novel online replanner to quickly look up the relevant offline trajectory in the database.
- Second, the online replanner of Vu et al. (2020) is reformulated and a computationally very efficient sparse quadratic programming solver is used. In this way, the online replanner is able to generate a new trajectory, with an average computation time of 2.5 ms on a PC and 15 ms on the dSPACE MicroLabBox, respectively. The validation of the proposed algorithm is performed using both simulations and an experimental setup. A video of the experimental results can be found at https://www.acin.tuwien.ac.at/en/65ce/

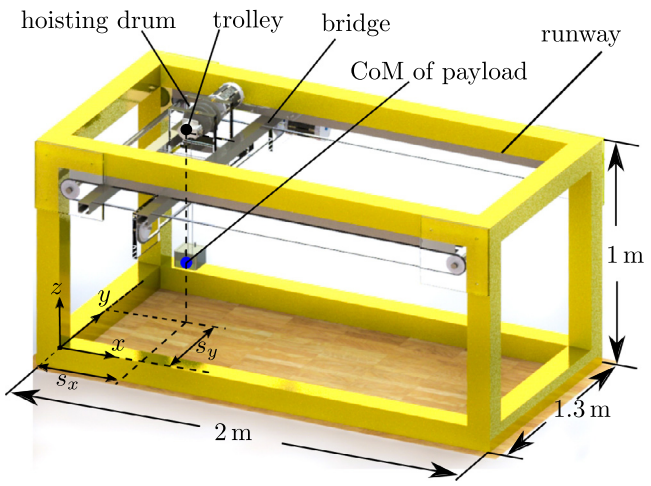

Fig. 1. Schematic of the lab-scale 3D gantry crane for $\alpha = \beta = 0$.

- Third, a model predictive controller (MPC) is introduced in the outer loop of the cascaded trajectory tracking controller, which is able to accurately perform the tracking task while adhering to the state and input constraints and suppressing sway in the payload.

In recent literature for automated cranes, e.g., Wang et al. (2019) and Zhang et al. (2021), the trajectories for a 3D gantry crane are calculated offline and tested in simulations. In contrast, the proposed algorithm in this work generates collision-free trajectories online, and is validated by experiments. Moreover, in comparison with the most successful practical studies, e.g., Böck and Kugi (2013) and Sawodny et al. (2002), the proposed combined method has the ability to generate near time-optimal dynamically feasible trajectories online for scenarios with obstacles and a moving target. To the best of the authors' knowledge, the real-time control and experimental validation of a 3D gantry crane in an environment with obstacles and a moving target have not been demonstrated so far. In addition, both the fast trajectory planning and the trajectory tracking controller systematically take into account the system constraints and control input limits, which is also experimentally validated.

Although the two-stage trajectory planning algorithm is widely used in robotics, see, e.g., Chai et al. (2018) and Lembono et al. (2020), the novel trajectory planning algorithm proposed in this paper enables online trajectory replanning in a dynamically changing environment with static obstacles and a moving target. This work is a proof of concept in the form of a laboratory experiment for real-time trajectory planning, where it is assumed that the required modules for scanning the environment including obstacle detection and the tracking system of the moving truck are available. One of the limitations of the proposed trajectory (re)planning is that it works only within the predefined starting and target subspaces. However, arbitrary subspaces can be chosen according to application requirements. For example, in this work, the starting subspace covers the entire workspace, while the target subspace corresponds to the workspace of a moving truck in 2D.

The paper is organized as follows: In Section 2, the mathematical model of the lab-scale 3D gantry crane of Lobe et al. (2018) is summarized. In Section 3, the novel two-stage trajectory planning algorithm is presented. Section 4 introduces the details of the cascaded control concept including a model predictive controller (MPC). Simulations and experiments are presented in Section 5. Finally, the last section concludes this work.

2. Mathematical model

The CAD model of the lab-scale 3D gantry crane is illustrated in Fig. 1. The gantry crane system consists of five degrees of freedom

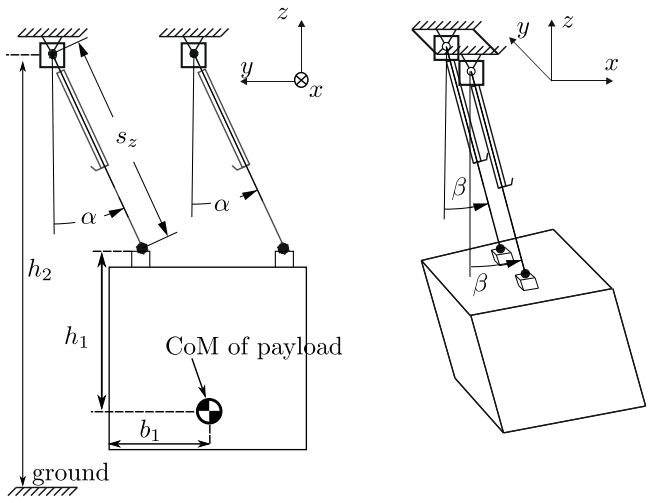

Fig. 2. The payload with the corresponding hosting cable angles α and β.

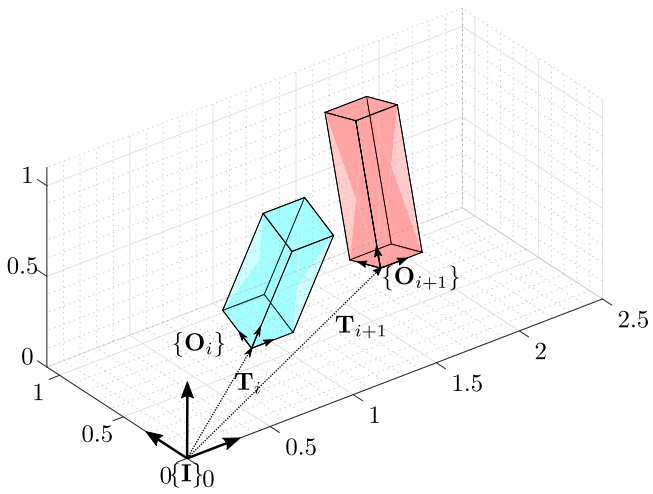

Fig. 3. Illustration of the obstacles in the working space and their coordinate frames.

$\mathbf{q}^{\mathrm{T}} = [s_x, s_y, s_z, \alpha, \beta]$, where s_x, s_y denote the position of the trolley on the bridge in x-, y-direction and s_z is the current hosting cable length. The variables α and β refer to the angles of the hoisting cables in the zy- and zx-plane, respectively, see Fig. 2. Note that the system state $\mathbf{q}$ is measured by five incremental encoders located at the three actuators for s_x, s_y, and s_z, and at the lifting drum for the two sway angles α and β. By assuming that the two hosting cables suspending the payload are identical and always under tension, the lab-scale 3D gantry crane can be modeled as a rigid-body system. Here, only the two sway angles α and β are considered as degrees of freedom and the twisting motion of the payload is neglected. For more details on the mechanical design parameters and the integrated equipment, the reader is referred to Lobe et al. (2018).

Using the five generalized coordinates $\mathbf{q}$, the state–space model of the 3D gantry crane reads as

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}^{-1}(\mathbf{q})\left(\begin{bmatrix}\mathbf{u}\\ \mathbf{0}\end{bmatrix} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q})\right) \end{bmatrix}, \tag{1}$$

with $\mathbf{z}^{\mathrm{T}} = [\mathbf{q}^{\mathrm{T}}, \dot{\mathbf{q}}^{\mathrm{T}}]$. The matrix $\mathbf{M}(\mathbf{q})$ denotes the symmetric and positive definite mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ includes Coriolis and centrifugal terms, $\mathbf{g}(\mathbf{q})$ are the forces associated with the potential energy, and $\mathbf{u}^{\mathrm{T}} = [u_1, u_2, u_3] \in \mathbb{R}^3$ are the driving forces in the x-, y-, and z-axis, respectively. A detailed derivation of the equations of motion (1) for the 3D gantry crane is given in the Appendix A of this paper.

3. Two-step trajectory planning

In this section, the fast trajectory planning algorithm is presented, which consists of an offline trajectory optimization to build up a trajectory database, a fast search algorithm to search within this trajectory database, and an online trajectory replanner.

3.1. Offline trajectory optimization

Note that only a brief introduction to the offline trajectory planner is presented here. For more details, the reader is referred to our earlier work (Vu et al., 2020). The general task of a gantry crane is to transport the payload from a starting (initial) state $\mathbf{z}_S$ to a target state $\mathbf{z}_T$ in a minimum time t_F, while respecting the constraints on the state variables and control inputs and avoiding collisions with obstacles. To achieve these objectives, the trajectory of the 3D gantry crane is discretized in time with $N+1$ grid points, the so called collocation points. By using trapezoidal direct collocation, see, e.g., Betts (2010) and Kelly (2017), the system dynamics (1) are transcribed into the nonlinear constraints (2b) and the nonlinear optimization problem for offline trajectory planning is written as

$$\min_{\boldsymbol{\xi}} \; t_F + \frac{1}{2}h\sum_{k=1}^{N-1}\sum_{i=1}^{m}\varphi_i(\mathbf{q}_k) \tag{2a}$$

$$\text{s.t. } \mathbf{z}_{k+1} - \mathbf{z}_k = \frac{1}{2}h(\mathbf{f}_k + \mathbf{f}_{k+1}) \tag{2b}$$

$$\mathbf{z}_0 = \mathbf{z}_S, \;\; \mathbf{z}_N = \mathbf{z}_T \tag{2c}$$

$$\underline{\mathbf{z}} \le \mathbf{z}_k \le \overline{\mathbf{z}}, \; k = 0, \dots, N \tag{2d}$$

$$\underline{\mathbf{u}} \le \mathbf{u}_k \le \overline{\mathbf{u}}, \; k = 0, \dots, N, \tag{2e}$$

where (2a) is the objective function with the traversal time t_F and the time step $h = t_F/N$. Moreover, (2c) refers to the desired starting and target state, and (2d), (2e) reflect the state and input constraints. In this context, $\underline{\mathbf{z}}$, $\overline{\mathbf{z}}$, $\underline{\mathbf{u}}$, and $\overline{\mathbf{z}}$ denote the lower and upper bounds of the state variables $\mathbf{z}_k^{\mathrm{T}} = [\mathbf{q}_k^{\mathrm{T}}, \dot{\mathbf{q}}_k^{\mathrm{T}}]$ and control input $\mathbf{u}_k$, $k = 0, \dots, N$. Note that the sway of the payload can be kept small by adjusting the admissible range for α_k and β_k in (2d). The index k in (2) refers to the discrete-time step at $t = kh$. Henceforth $\boldsymbol{\xi}$ denotes the optimization variables

$$\boldsymbol{\xi}^{\mathrm{T}} = [t_F, \mathbf{z}_0^{\mathrm{T}}, \dots, \mathbf{z}_N^{\mathrm{T}}, \mathbf{u}_0^{\mathrm{T}}, \dots, \mathbf{u}_N^{\mathrm{T}}] \in \mathbb{R}^{1+13(N+1)}. \tag{3}$$

The expression $\varphi_i(\mathbf{q}_k)$ in (2a) is an artificial potential function, which is evaluated at the collocation points $\mathbf{q}_k$, $k = 0, \dots, N-1$, for avoiding collisions with obstacles $i = 1, \dots, m$, in the operating range of the gantry crane.

To solve the optimization problem (2), the interior point method (IPM) solver from the open source package Interior Point OPTimize (IPOPT) (Wächter & Biegler, 2006) is used. To speed up the convergence rate of the solver, the artificial potential functions φ_i in (2a) are formulated in a convex and smooth way, see Section 3.2 for more details. To further improve the computational speed, the automatic differentiation (AD) method is applied to compute the analytical gradient and the Hessian functions of the objective function (2a) and the constraint function (2b).

3.2. Collision avoidance

A brief introduction of the artificial potential functions $\varphi_i, i = 1, \dots, m$ is given in the following, see also Vu et al. (2020). As illustrated in Fig. 3, obstacles are considered to be bounded by boxes with the parameter vector ${}_{\mathbf{O}_i}\mathbf{p}_i = [w_i, h_i, d_i]^{\mathrm{T}}$ containing the width w_i, the height h_i, and the depth d_i of the box along the x-, y-, and z-axis in the box

frame $\{\mathbf{O}_i\}$ of the ith box, $i = 1, \dots, m$. Furthermore, the location of the obstacles is known with the translation vector $\mathbf{T}_i$ and the rotation matrix $\mathbf{R}_i$ at the box frame $\{\mathbf{O}_i\}$ w.r.t. the inertial frame $\{\mathbf{I}\}$. The position of the center of mass (CoM) of the payload described in the inertial frame is denoted by $\mathbf{r}$ and is associated with a point $\mathbf{q}^{\mathrm{T}} = [s_x, s_y, s_z, \alpha, \beta]$ on the trajectory in the form

$$\mathbf{r}(\mathbf{q}) = \begin{bmatrix} s_x + \sin(\beta)\cos(\alpha)s_z - \sin(\beta)h_1 \\ s_y - \sin(\alpha)s_z - b_1 \\ \cos(\beta)\cos(\alpha)s_z - \cos(\beta)h_1 \end{bmatrix}, \tag{4}$$

with the parameters h_1 and b_1 depicted in Fig. 2. This CoM position can be expressed in the ith box frame $\{\mathbf{O}_i\}$ as

$${}_{\mathbf{O}_i}\mathbf{r} = \mathbf{R}_i^{\mathrm{T}}\Big(\mathbf{r} - \mathbf{T}_i\Big). \tag{5}$$

A point $\mathbf{q}$ on the trajectory is considered as obstacle-free if and only if the condition

$$\bar{S}_i(\mathbf{q}) = \min(\Delta p_{i,j})_{j=1,2,3} < 0 \tag{6}$$

is satisfied, where $\Delta p_{i,j}(\mathbf{q})$ is the jth component of the vector

$$\mathbf{\Delta p}_i = \Big({}_{\mathbf{O}_i}\mathbf{r}\Big) \circ \Big({}_{\mathbf{O}_i}\mathbf{p}_i - {}_{\mathbf{O}_i}\mathbf{r}\Big). \tag{7}$$

The operator $\circ$ in (7) refers to the element-wise product. The artificial potential function helps to pull the trajectory out of the obstacles and has nearly no effect on the searching direction in the free space. Hence, the artificial potential function is defined as:

$$\bar{\varphi}_i(\mathbf{q}) = \max(\gamma_i \bar{S}_i(\mathbf{q}), 0), \tag{8}$$

where $\gamma_i > 0$, $i = 1, \dots, m$, is a user-defined scaling parameter. In order to render the potential function (8) with (6) sufficiently smooth, the LogSumExp function is employed, see, e.g., An et al. (2016) and Nielsen and Sun (2016), resulting in

$$S_i(\mathbf{q}) = \frac{1}{\eta_1}\log\left(\sum_{j=1}^{3} \mathrm{e}^{\eta_1 \Delta p_{i,j}}\right) \tag{9a}$$

$$\varphi_i(\mathbf{q}) = \frac{1}{\eta_2}\log\Big(1 + \mathrm{e}^{\eta_2 \gamma_i S_i(\mathbf{q})}\Big), \tag{9b}$$

with the so-called softness coefficients $\eta_1 < 0$ and $\eta_2 > 0$. In order to create a safety margin around the obstacles, the margin $\boldsymbol{\delta} = [\delta_x, \delta_y, \delta_z]^{\mathrm{T}}$ is added in the form $\mathbf{p}_i = [w_i + \delta_x/2, h_i + \delta_y/2, d_i + \delta_z/2]^{\mathrm{T}}$ and $\mathbf{T}_i$ is replaced by $\mathbf{T}_i - \boldsymbol{\delta}/2$. Finally, the artificial potential functions φ_i of all obstacles $i = 1, \dots, m$ are combined by simply forming the sum $\sum_{i=1}^{m} \varphi_i$. Note that the proposed obstacle avoidance does not directly consider the collision with the ropes. However, in the optimization problem (2), the two sway angles α and β are restricted to a small range of ± 0.05 rad ($\approx \pm 3°$) and a margin is introduced to enlarge the real obstacles. These measures reduce the risk of rope collisions with the obstacles. For further details on this formulation, the reader is referred to our previous work (Vu et al., 2020).

3.3. Trajectory database

Although the offline trajectory planner in the previous subsection is able to compute the optimal trajectory very quickly, with an average time of 50 ms for one trajectory with the desired convergence tolerance of 10^{-8}, this computation time is still not sufficient for the considered real-time application. A gantry crane often performs repetitive tasks, which means that many similar trajectories have to be tracked during a work shift. Therefore, it suggests itself to reuse the previous trajectories in the form of an offline trajectory database. Since (2) only leads to locally optimal solutions and the obstacle potential function is represented as a soft constraint in the objective function (2a), the solution of (2) might be trapped in a local minimum and may violate the obstacle constraint. In this case, (2a) is recomputed with a different random initial guess. In this way, we succeed that all trajectories in the offline database are collision-free and dynamically feasible. Note that the average computation time of 50 ms for one trajectory in the offline database consisting of 10^4 optimal trajectories includes the violation checking and recomputation. Without loss of generality, the starting subspace $\mathcal{X}_S$ is assumed to cover the entire workspace, while the target subspace $\mathcal{X}_T$ covers the workspace of a moving target, in our case a moving truck on the ground which is represented by a plane parallel to the bottom of $\mathcal{X}_S$, as shown in Fig. 4. For each subspace, a grid with equal spacing is chosen. Then, the offline trajectory planner is used to plan offline trajectories from each of the n_S grid points of the starting subspace $\mathcal{X}_S$ (starting state $\mathbf{z}_S^{\mathrm{T}} = [\mathbf{q}_S^{\mathrm{T}}, \mathbf{0}]$ in (2c)) to each of the n_T grid points in the target subspace $\mathcal{X}_T$ (target state $\mathbf{z}_T^{\mathrm{T}} = [\mathbf{q}_T^{\mathrm{T}}, \mathbf{0}]$ in (2c)), in total $n_S n_T$ trajectories. The forward kinematics (4) computes the position of the center of mass $\mathbf{r}(\mathbf{q})$ of the payload based on the five degrees of freedom $\mathbf{q}$. In addition, each trajectory in the database is represented by two labels containing the position $\mathbf{r}_S = \mathbf{r}(\mathbf{q}_S)$ in the Cartesian space of the starting state $\mathbf{z}_S$ and the position $\mathbf{r}_T = \mathbf{r}(\mathbf{q}_T)$ in the Cartesian space of the target state $\mathbf{z}_T$. In order to efficiently search for the nearest trajectory, an offline database is constructed using the two labels for each trajectory according to the k-d tree algorithm (Bentley, 1975), which is a well-known space partitioning data structure for partitioning and organizing points. The search complexity of this algorithm for N labels in the database is $\mathcal{O}(\log N)$ compared to $\mathcal{O}(N)$ for an unprocessed database, see, e.g., Pinkham et al. (2020).

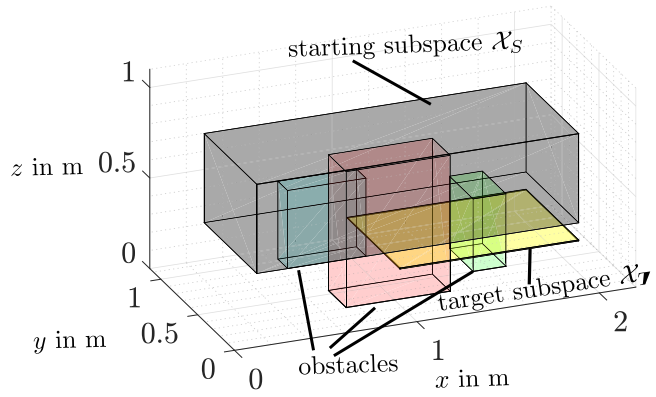

Fig. 4. Illustration of the two subspaces $\mathcal{X}_S$ and $\mathcal{X}_T$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The offline database structure is shown in Fig. 5, where the labels in $\mathcal{X}_T$ are denoted by

$$\mathcal{X}_T = \{\mathbf{r}_T^1, \mathbf{r}_T^2, \dots, \mathbf{r}_T^{n_T}\}$$

and the labels in $\mathcal{X}_S$ by

$$\mathcal{X}_S = \{\mathbf{r}_S^1, \mathbf{r}_S^2, \dots, \mathbf{r}_S^{n_S}\}.$$

The third layer of the offline database contains the time-optimal offline trajectories $\Xi_i = [\xi_{ij}^*] \in \mathbb{R}^{[1+13(N+1)]\times n_S}$ with $i = 1, \dots, n_T$ and $j = 1, \dots, n_S$ connecting the Cartesian positions $\mathbf{r}_S^i$ and $\mathbf{r}_T^j$. The details of the search algorithm are presented in the next subsection.

3.4. Online trajectory replanner

The online trajectory replanner computes the optimal trajectory according to the following procedure. After receiving the command to start the motion from the starting state $\bar{\mathbf{z}}_S$ to the target state $\bar{\mathbf{z}}_T$, the online trajectory replanner first searches the trajectory database to find the closest trajectory. Later in this subsection, we will thoroughly explain what we exactly mean with the term closest. For this, we will specify the metrics used for measuring distances. Then, linear constrained quadratic programming is applied to minimize the deviation

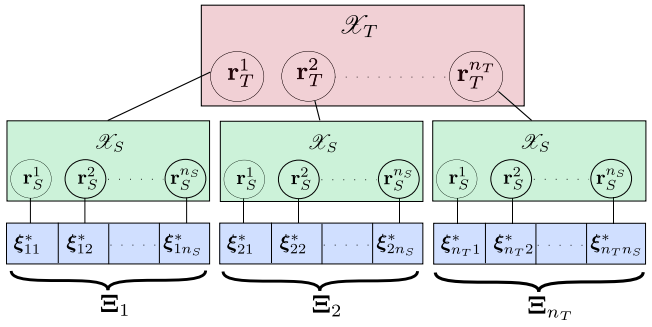

Fig. 5. Offline database structure.

Algorithm 1 Retrieving offline trajectory in database

1: **function** FastRetrievingOfflineTrajectory
2: $\hat{\mathbf{r}}_T, \hat{\mathbf{z}}_T \leftarrow$ TargetStatePrediction($\mathbf{r}_T^{\text{curr}}, \mathbf{v}_T^{\text{curr}}, T_{s,2}$)
3: $\hat{\mathbf{r}}_S, \hat{\mathbf{z}}_S \leftarrow$ StartingStatePrediction($\mathbf{z}_S^{\text{curr}}, T_{s,2}$)
4: $i \leftarrow$ knnsearch($\mathcal{X}_T, \hat{\mathbf{r}}_T$)
5: **if** IsTargetStationary($\hat{\mathbf{r}}_T, \hat{\mathbf{r}}_T^{\text{prev}}$) **then**
6: **if** IsTheFirstSearch **then**
7: $j \leftarrow$ knnsearch($\mathcal{X}_S, \hat{\mathbf{r}}_S$)
8: $\boldsymbol{\xi}^* = \boldsymbol{\xi}_{ij}^*$
9: **end if**
10: **else**
11: $(\boldsymbol{\xi}_{ij}^*, l) =$ knnsearch($\boldsymbol{\Xi}_i, \hat{\mathbf{z}}_S$)
12: $\boldsymbol{\xi}^* =$ TrajectoryRefinement($\boldsymbol{\xi}_{ij}^*, l$)
13: **end if**
14: **end function**

from the offline trajectory while satisfying the dynamic constraints of the 3D gantry crane. To achieve fast trajectory replanning in real time, a computationally efficient algorithm for the search task is summarized in Algorithm 1 and presented next.

In the first step of Algorithm 1, the target position and the target state are predicted for one sampling time $T_{s,2}$ of the trajectory replanner denoted by $\hat{\mathbf{r}}_T$ and $\hat{\mathbf{z}}_T$, based on the current target position $\mathbf{r}_T^{\text{curr}}$ and velocity $\mathbf{v}_T^{\text{curr}}$. For this prediction, it is assumed that the target velocity $\mathbf{v}_T^{\text{curr}}$ remains constant for the sampling time $T_{s,2}$. Similarly, in line 3 of Algorithm 1, the predicted starting position $\hat{\mathbf{r}}_S$ and the predicted state $\hat{\mathbf{z}}_S$ are estimated using the current starting state $\mathbf{z}_S^{\text{curr}}$ measured by encoders. Once the predicted target state $\hat{\mathbf{r}}_T$ is obtained, the search algorithm k nearest neighbor (k-nn) (Soleymani & Morgera, 1987) is employed to find the index i of the offline target state $\mathbf{r}_T^i, i = 1, \dots, n_T$, in $\mathcal{X}_T$ that is closest in the sense of the smallest Euclidean distance $\|\hat{\mathbf{r}}_T - \mathbf{r}_T^i\|_2$ to the predicted target position $\hat{\mathbf{r}}_T$, see line 4 of Algorithm 1. By comparing the predicted state $\hat{\mathbf{r}}_T$ with the previous predicted state $\hat{\mathbf{r}}_T^{\text{prev}}$, the search algorithm can detect whether the target is moving or stationary. At this point, the search algorithm distinguishes between two cases:

- If the target state is stationary and the trajectory replanning algorithm was not executed before, the k-nn search is applied to find the index j of the closest starting position to $\hat{\mathbf{r}}_S$, see line 7 in Algorithm 1 and Fig. 5. Using the two indexes i, j of the label in $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively, the closest trajectory $\boldsymbol{\xi}_{ij}^*$ is directly taken from the database.
- If the target moves, the k-nn search must be executed in the trajectory set $\boldsymbol{\Xi}_i$ (line 11 of Algorithm 1). In this case, the predicted starting point is normally not a stationary point, but $\hat{\mathbf{z}}_S^{\mathrm{T}} = [\hat{\mathbf{q}}_S^{\mathrm{T}}, \hat{\dot{\mathbf{q}}}_S^{\mathrm{T}}]$, with $\hat{\dot{\mathbf{q}}}_S \neq \mathbf{0}$. Thus, the closest starting state in the trajectory database is obtained from the trajectory set $\boldsymbol{\Xi}_i$ by choosing those trajectory $\boldsymbol{\xi}_{ij}^*, j = 1, \dots, n_S$, which has a collocation point $(\mathbf{z}_{ij,l}^*)^{\mathrm{T}} = [(\mathbf{q}_{ij,l}^*)^{\mathrm{T}}, (\dot{\mathbf{q}}_{ij,l}^*)^{\mathrm{T}}]$, see (3), that is closest to $\hat{\mathbf{z}}_S$ in the weighted Euclidean distance metric

$$\|\mathbf{q}_{ij,l}^* - \hat{\mathbf{q}}_S\|_2 + \|\mathrm{diag}(\rho_n, n = 1, \dots, 5)(\dot{\mathbf{q}}_{ij,l}^* - \hat{\dot{\mathbf{q}}}_S)\|_2,$$

with the user-defined weighting parameter $\rho_n > 0, n = 1, \dots, 5$. Since the range of the system state $\underline{\mathbf{q}} \le \mathbf{q} \le \overline{\mathbf{q}}$ and of the system state velocity $\underline{\dot{\mathbf{q}}} \le \mathbf{q} \le \overline{\dot{\mathbf{q}}}$ are different, ρ_n is chosen to normalize the velocity error with respect to the state error in the form $\rho_n = \dfrac{\overline{q_n} - \underline{q_n}}{\overline{\dot{q}_n} - \underline{\dot{q}_n}}, n = 1, \dots, 5$. Note that the sway angles α, β and the angular velocities $\dot{\alpha}$, $\dot{\beta}$ are also considered in this case, since the algorithm considers the whole state $\hat{\mathbf{z}}_S$. There are two return variables j and l of the k-nn search in line 11 of Algorithm 1, where j is the index of the retrieved trajectory $\boldsymbol{\xi}_{ij}^*$ in $\boldsymbol{\Xi}_i$ and l is the index of the closest state $\mathbf{z}_l^*$ of this offline trajectory $\boldsymbol{\xi}_{ij}^*$. At this point, the closest trajectory from the offline database reads as

$$(\boldsymbol{\xi}^*)^{\mathrm{T}} = [\frac{l}{N} t_F^*, (\mathbf{z}_l^*)^{\mathrm{T}}, \dots, (\mathbf{z}_N^*)^{\mathrm{T}}, (\mathbf{u}_l^*)^{\mathrm{T}}, \dots, (\mathbf{u}_N^*)^{\mathrm{T}}]. \tag{10}$$

Note that only a segment of the whole offline trajectory $\boldsymbol{\xi}_{ij}^*$ is used in (10) and therefore the number of grid points of $(\boldsymbol{\xi}^*)$ is reduced to $N - l + 1$. Since the number of $(N + 1)$ grid points is fixed during the computation of the online trajectory replanner, an interpolation scheme has to be employed in line 12 of Algorithm 1. Between two adjacent collocation points k and $k + 1$, with $k = l, \dots, N-1$, the input $\mathbf{u}^*(t)$ and the state $\mathbf{z}^*(t)$ for $t = [kh^*, (k+1)h^*]$, are interpolated as linear and quadratic splines, respectively, i.e.,

$$\mathbf{z}^*(t) \approx \mathbf{z}_k^* + (t - kh^*)\mathbf{f}_k^* + \frac{(t - kh^*)^2}{2h^*}(\mathbf{f}_{k+1}^* - \mathbf{f}_k^*), \tag{11a}$$

$$\mathbf{u}^*(t) \approx \mathbf{u}_k^* + \frac{t - kh^*}{h^*}(\mathbf{u}_{k+1}^* - \mathbf{u}_k^*), \tag{11b}$$

for $k = l, \dots, N - 1$, with $h^* = t_F^*/N$.

For brevity, the same notation as in (3) is used after the refinement of the closest trajectory (10). To this end, the trajectory is expressed as

$$(\boldsymbol{\xi}^*)^{\mathrm{T}} = [t_F^*, (\mathbf{z}_0^*)^{\mathrm{T}}, \dots, (\mathbf{z}_N^*)^{\mathrm{T}}, (\mathbf{u}_0^*)^{\mathrm{T}}, \dots, (\mathbf{u}_N^*)^{\mathrm{T}}]. \tag{12}$$

Recalling the system dynamics condition (2b) for the closest offline trajectory, the relation

$$\mathbf{z}_{k+1}^* = \mathbf{z}_k^* + \frac{t_F^*}{2N}(\mathbf{f}_k^* + \mathbf{f}_{k+1}^*) \tag{13}$$

holds.

In the following, the online trajectory replanner is explained in detail. Assuming that the number of grid points in the starting and target subspace is sufficiently dense, only small deviations

$$\delta\boldsymbol{\xi}^{\mathrm{T}} = [\delta t_F, (\delta\mathbf{z}_0)^{\mathrm{T}}, ..., (\delta\mathbf{z}_N)^{\mathrm{T}}, (\delta\mathbf{u}_0)^{\mathrm{T}}, \dots, (\delta\mathbf{u}_N)^{\mathrm{T}}]$$

need to be taken into account to compute the new trajectory connecting the predicted starting state $\hat{\mathbf{z}}_S$ with the predicted target state $\hat{\mathbf{z}}_T$. The first-order linearization of the discrete-time system dynamics (2b) w.r.t. the closest database trajectory $\boldsymbol{\xi}^*$ reads as

$$\begin{aligned}\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{t_F^* + \delta t_F}{2N} \Big(& \mathbf{f}_k^* + \boldsymbol{\Gamma}_k^{\mathbf{z}}\delta\mathbf{z}_k + \boldsymbol{\Gamma}_k^{\mathbf{u}}\delta\mathbf{u}_k \\ & + \mathbf{f}_{k+1}^* + \boldsymbol{\Gamma}_{k+1}^{\mathbf{z}}\delta\mathbf{z}_{k+1} + \boldsymbol{\Gamma}_{k+1}^{\mathbf{u}}\delta\mathbf{u}_{k+1} \Big),\end{aligned} \tag{14}$$

with $\delta\mathbf{z}_k = \mathbf{z}_k - \mathbf{z}_k^*$, $\delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^*$, $\delta t_F = t_F - t_F^*$, and

$$\boldsymbol{\Gamma}_k^{\mathbf{z}} = \frac{\partial \mathbf{f}}{\partial \mathbf{z}}\Big|_{\mathbf{z}_k^*, \mathbf{u}_k^*}, \quad \boldsymbol{\Gamma}_k^{\mathbf{u}} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Big|_{\mathbf{z}_k^*, \mathbf{u}_k^*}$$

for $k = 0, \dots, N - 1$. Note that the state deviations at $k = 0$ and $k = N$ are fixed by

$$\delta\mathbf{z}_0 = \hat{\mathbf{z}}_S - \mathbf{z}_0^* \tag{15a}$$

$$\delta\mathbf{z}_N = \hat{\mathbf{z}}_T - \mathbf{z}_N^* \tag{15b}$$

due to the predicted starting and target state $\hat{\mathbf{z}}_S$ and $\hat{\mathbf{z}}_T$. Subtracting (13) from (14) and neglecting the terms containing a product of deviation variables, the constraint function reduces to

$$\delta\mathbf{z}_{k+1} = \delta\mathbf{z}_k + \frac{t_F^*}{2N}\left(\boldsymbol{\Gamma}_k^{\mathbf{z}}\delta\mathbf{z}_k + \boldsymbol{\Gamma}_k^{\mathbf{u}}\delta\mathbf{u}_k + \boldsymbol{\Gamma}_{k+1}^{\mathbf{z}}\delta\mathbf{z}_{k+1} + \boldsymbol{\Gamma}_{k+1}^{\mathbf{u}}\delta\mathbf{u}_{k+1}\right) + \frac{\delta t_F}{t_F^*}(\mathbf{z}_{k+1}^* - \mathbf{z}_k^*). \tag{16}$$

In a more compact form, (16) is rewritten as

$$\mathbf{C}_{k+1}\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k, \tag{17}$$

where

$$\mathbf{C}_{k+1} = \begin{bmatrix} \mathbf{I} - \frac{h^*}{2}\boldsymbol{\Gamma}_{k+1}^{\mathbf{z}} & -\frac{h^*}{2}\boldsymbol{\Gamma}_{k+1}^{\mathbf{u}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix},$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{I} + \frac{h^*}{2}\boldsymbol{\Gamma}_k^{\mathbf{z}} & \frac{h^*}{2}\boldsymbol{\Gamma}_k^{\mathbf{u}} & \frac{\mathbf{z}_{k+1}^* - \mathbf{z}_k^*}{t_F^*} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix},$$

$$\mathbf{x}_k = \begin{bmatrix} \delta\mathbf{z}_k \\ \delta\mathbf{u}_k \\ \delta t_{F,k} \end{bmatrix},$$

and $h^* = \frac{t_F^*}{N}$. For simplicity, only one variable for the final time δt_F in (14) was introduced instead of $\delta t_{F,k}, k = 0, \dots, N-1$. Thus, $\delta t_{F,k+1} = \delta t_{F,k}$ was used in (17). The deviation vector $\boldsymbol{\xi}_k$ is obtained as the solution of a linear constrained quadratic program (LCQP) of the form

$$\min_{\mathbf{x}_k} \ \frac{1}{2}\sum_{k=1}^{N-1} \mathbf{x}_k^{\mathrm{T}}\mathbf{Q}_k\mathbf{x}_k \tag{18a}$$

$$\text{s.t. } \mathbf{C}_{k+1}\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k, \ k = 0, \dots, N-1 \tag{18b}$$

$$\underline{\mathbf{x}_k} \le \mathbf{x}_k \le \overline{\mathbf{x}_k}, \qquad k = 0, \dots, N \tag{18c}$$

with (15) and the positive definite weighting matrix

$$\mathbf{Q}_k = \mathrm{diag}(\mathbf{Q}_{\mathbf{z}_k}, \mathbf{Q}_{\mathbf{u}_k}, Q_{t_F}). \tag{19}$$

With the choice of $Q_{t_F} > 0$ and the submatrices $\mathbf{Q}_{\mathbf{z}_k}$ and $\mathbf{Q}_{\mathbf{u}_k}$, the deviation of the online trajectory from the selected database trajectory (12) can be specifically weighted in the objective function (18a) w.r.t. the traversal time t_F, the state $\mathbf{z}_k$, and the control input $\mathbf{u}_k$, respectively. The inequality condition (18c) corresponds to (2d) and (2e), where

$$\underline{\mathbf{x}_k}^{\mathrm{T}} = [\underline{\mathbf{z}_k}^{\mathrm{T}} - (\mathbf{z}_k^*)^{\mathrm{T}}, \underline{\mathbf{u}_k}^{\mathrm{T}} - (\mathbf{u}_k^*)^{\mathrm{T}}, \underline{\delta t_F}],$$
$$\overline{\mathbf{x}_k}^{\mathrm{T}} = [\overline{\mathbf{z}_k}^{\mathrm{T}} - (\mathbf{z}_k^*)^{\mathrm{T}}, \overline{\mathbf{u}_k}^{\mathrm{T}} - (\mathbf{u}_k^*)^{\mathrm{T}}, \overline{\delta t_F}],$$

for $k = 1, \dots, N-1$, and $\overline{\delta t_F}$ and $\underline{\delta t_F}$ is a sufficiently large upper and lower bound for δt_F, respectively. Note that the equality constraints in (15) must be taken into account by

$$\underline{\mathbf{x}_0}^{\mathrm{T}} = [\delta\mathbf{z}_0^{\mathrm{T}}, \delta\mathbf{u}_0^{\mathrm{T}}, \underline{\delta t_F}], \ \overline{\mathbf{x}_0}^{\mathrm{T}} = [\delta\mathbf{z}_0^{\mathrm{T}}, \delta\mathbf{u}_0^{\mathrm{T}}, \overline{\delta t_F}],$$
$$\underline{\mathbf{x}_N}^{\mathrm{T}} = [\delta\mathbf{z}_N^{\mathrm{T}}, \delta\mathbf{u}_N^{\mathrm{T}}, \underline{\delta t_F}], \ \overline{\mathbf{x}_N}^{\mathrm{T}} = [\delta\mathbf{z}_N^{\mathrm{T}}, \delta\mathbf{u}_N^{\mathrm{T}}, \overline{\delta t_F}].$$

It is important to keep $\delta\mathbf{q}_k$ of $\delta\mathbf{z}_k^{\mathrm{T}} = [\delta\mathbf{q}_k, \delta\dot{\mathbf{q}}_k]$ close to zero if the corresponding collocation points $\mathbf{q}_k^*$ on the selected database trajectory are already close to one of the obstacles. Therefore, the submatrix $\mathbf{Q}_{\mathbf{q}_k}$ of the weighting matrix $\mathbf{Q}_{\mathbf{z}_k} = \mathrm{diag}(\mathbf{Q}_{\mathbf{q}_k}, \mathbf{Q}_{\dot{\mathbf{q}}_k})$ in (19) is adjusted depending on the distance of the corresponding payload position $\mathbf{r}(\mathbf{q}_k^*)$ to an obstacle. Note that instead of calculating the exact distance, the artificial potential functions $\varphi_i(\mathbf{q})$, $i = 1, \dots, m$, according to (9), see also (2), serve as a basis to indirectly consider the obstacles in (18a). In particular, the Hessian of the artificial potential function is used to adjust the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$ in the form

$$\mathbf{Q}_{\mathbf{q}_k} = \mathbf{Q}_{\mathbf{q}} + \lambda \left.\frac{\partial^2\left(\sum_{i=1}^m \varphi_{i,k}(\mathbf{q}_k)\right)}{\partial\mathbf{q}_k^2}\right|_{\mathbf{q}_k^*}, \tag{20}$$

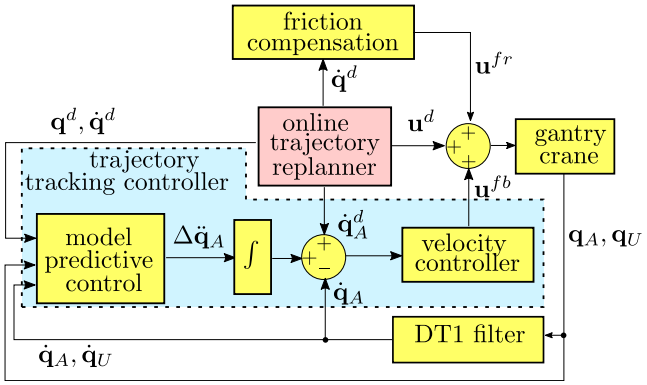

Fig. 6. Block diagram of the control structure. The yellow blocks are processed with the fast sampling time $T_{s,1}$ while the red block is computed with the slower sampling time $T_{s,2}$, see also Fig. 11. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Monte Carlo simulations with 10^4 test cases for two scenarios.

	Stationary target	Moving target
Number of simulation fails	0	6
Number of failed collision checks	230	251
Success rate	97.70%	97.43%
Average computing time	2 ms	2.5 ms

with the constant matrix $\mathbf{Q}_{\mathbf{q}} > 0$ and the tuning parameter $\lambda > 0$. Clearly, the closer a collocation point $\mathbf{q}_k^*$ of the database trajectory is to an obstacle, the larger is the corresponding entry in the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$. This in turn makes the deviation of the online trajectory, which is a solution of (18), from the database trajectory in terms of $\delta\mathbf{q}_k$ at the considered point $\mathbf{q}_k^*$ small. This adaption of the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$, see (20), is introduced to ensure that also the online replanned trajectory is collision-free. Finally, the optimal trajectory of the online trajectory replanner $\boldsymbol{\xi}^{*,onl}$ reads as

$$\boldsymbol{\xi}^{*,onl} = \boldsymbol{\xi}^* + \delta\boldsymbol{\xi}^*, \tag{21}$$

where $\delta\boldsymbol{\xi}^*$ results from the solution of (18) in the form

$$(\delta\boldsymbol{\xi}^*)^{\mathrm{T}} = [\delta t_F^*, (\delta\mathbf{z}_0^*)^{\mathrm{T}}, \dots, (\delta\mathbf{z}_N^*)^{\mathrm{T}}, (\delta\mathbf{u}_0^*)^{\mathrm{T}}, \dots, (\delta\mathbf{u}_N^*)^{\mathrm{T}}].$$

Since the LCQP (18) only leads to locally optimal solutions, it may get stuck in a local minimum that violates the obstacle constraint. Thus, if the solution of (18) gets stuck in a local minimum, (18) is reprocessed with a different nearest offline trajectory resulting from the solution of the k-neighbor search in Algorithm 1. This heuristic modification is used in the experiment as a safety measure to protect the laboratory equipment. In the Monte Carlo simulation results, see Table 1 in Section 5, this is not included to show the performance of the proposed algorithm without any heuristic modification.

4. Control design

The control structure of the 3D gantry crane consists of the online trajectory replanner, the trajectory tracking controller, and a friction compensation. The overall control structure is depicted in Fig. 6. To stabilize the payload around a given trajectory, a combination of a feedforward and a feedback controller, also denoted as trajectory tracking controller, is introduced. The control output of the online trajectory replanner is used as feedforward part, while the feedback controller has a cascaded structure consisting of the outer model predictive control (MPC) and the inner velocity controller. The details of the friction compensation are discussed in Lobe et al. (2018) and the velocity controller is a decoupled standard PI controller. In this section, we focus on the model predictive controller.

As a first step, in order to reduce the complexity of the system dynamics and further increase the computational speed, the state $\mathbf{q}$ is divided into the state of the actuated subsystem $\mathbf{q}_A = [s_x, s_y, s_z]^{\mathrm{T}}$ and the unactuated subsystem $\mathbf{q}_U = [\alpha, \beta]^{\mathrm{T}}$. Therefore, the system dynamics (1) is expressed as

$$\begin{bmatrix} \mathbf{M}_A & \mathbf{M}_{AU} \\ \mathbf{M}_{AU}^{\mathrm{T}} & \mathbf{M}_U \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{C}_A & \mathbf{C}_{AU} \\ \mathbf{C}_{UA} & \mathbf{C}_U \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{g}_A \\ \mathbf{g}_U \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix}. \tag{22}$$

The acceleration of the unactuated angles $\ddot{\mathbf{q}}_U$ and the control input $\mathbf{u}$ are calculated from (22), which yields

$$\ddot{\mathbf{q}}_U = \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\ddot{\mathbf{q}}_A - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U), \tag{23a}$$

$$\mathbf{u} = \mathbf{M}_A\ddot{\mathbf{q}}_A + \mathbf{M}_{AU}\ddot{\mathbf{q}}_U + \mathbf{C}_A\dot{\mathbf{q}}_A + \mathbf{C}_{AU}\dot{\mathbf{q}}_U + \mathbf{g}_A. \tag{23b}$$

It is worth noting that the acceleration of the actuated state $\ddot{\mathbf{q}}_A$ acts directly on the unactuated subsystem. Thus, considering $\mathbf{u}_A = \ddot{\mathbf{q}}_A$ as a new control input to the system, the state–space representation of (1) reads as

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_U \\ \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \\ \mathbf{u}_A \\ \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\mathbf{u}_A - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U) \end{bmatrix}. \tag{24}$$

In a more compact form, (24) is rewritten as

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{z} = \tilde{\mathbf{f}}(\mathbf{z}, \mathbf{u}_A). \tag{25}$$

The system (25) is linearized around the desired trajectory $(\mathbf{z}^d, \mathbf{u}_A^d) = (\mathbf{z}^* + \delta\mathbf{z}, \ddot{\mathbf{q}}_A^d)$, with

$$(\mathbf{z}^d)^{\mathrm{T}} = [(\mathbf{q}_A^d)^{\mathrm{T}}, (\mathbf{q}_U^d)^{\mathrm{T}}, (\dot{\mathbf{q}}_A^d)^{\mathrm{T}}, (\dot{\mathbf{q}}_U^d)^{\mathrm{T}}]$$

and $\delta\mathbf{z}$ is the solution of the LCQP (18). By approximating the system dynamics (24) around $(\mathbf{z}^d, \mathbf{u}_A^d)$, the discrete time-varying system dynamics is expressed as

$$\Delta\mathbf{z}_{k+1} = \boldsymbol{\Phi}_k\Delta\mathbf{z}_k + \boldsymbol{\Omega}_k\Delta\mathbf{u}_{A,k}, \tag{26}$$

with

$$\begin{aligned} \Delta\mathbf{z}_k &= \mathbf{z}_k - \mathbf{z}_k^d & \Delta\mathbf{u}_{Ak} &= \mathbf{u}_{A,k} - \mathbf{u}_{A,k}^d \\ \boldsymbol{\Phi}_k &= \mathbf{I}_{10} + T_{s,1}\mathbf{A}_k & \boldsymbol{\Omega}_k &= T_{s,1}\mathbf{B}_k \\ \mathbf{A}_k &= \frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{z}}\Big|_{\mathbf{z}_k^d, \mathbf{u}_{A,k}^d} & \mathbf{B}_k &= \frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{u}_A}\Big|_{\mathbf{z}_k^d, \mathbf{u}_{A,k}^d}, \end{aligned} \tag{27}$$

and the sampling time $T_{s,1}$. The model predictive control is applied at the time instant t_k to control the error system (26) by solving the following optimization problem w.r.t. $\Delta\mathbf{u}_A^{\mathrm{T}} = [\Delta\mathbf{u}_{A,k}^{\mathrm{T}}, \dots, \Delta\mathbf{u}_{A,k+N_m}^{\mathrm{T}}]$ over a finite horizon of $N_m + 1$ steps

$$\min_{\Delta\mathbf{u}_A} \sum_{j=0}^{N_m} (\Delta\mathbf{z}_{k+j}^{\mathrm{T}}\mathbf{Q}_m\Delta\mathbf{z}_{k+j} + \Delta\mathbf{u}_{A,k+j}^{\mathrm{T}}\mathbf{R}_m\Delta\mathbf{u}_{A,k+j}) + \Delta\mathbf{z}_{k+N_m+1}^{\mathrm{T}}\mathbf{Q}_f\Delta\mathbf{z}_{k+N_m+1} \tag{28a}$$

$$\text{s.t.} \quad \Delta\mathbf{z}_{k+j+1} = \boldsymbol{\Phi}_k\Delta\mathbf{z}_{k+j} + \boldsymbol{\Omega}_k\Delta\mathbf{u}_{A,k+j}, \tag{28b}$$

$$\underline{\Delta\mathbf{z}} \le \Delta\mathbf{z}_{k+j} \le \overline{\Delta\mathbf{z}}, \; j = 0, \dots, N_m, \tag{28c}$$

where $\mathbf{Q}_m$, $\mathbf{R}_m$, and $\mathbf{Q}_f$ are positive definite weighting matrices and $\underline{\Delta\mathbf{z}} = \underline{\mathbf{z}} - \mathbf{z}^d$ and $\overline{\Delta\mathbf{z}} = \overline{\mathbf{z}} - \mathbf{z}^d$ is the lower and upper bound of the state variables, respectively. The first element of the solution $\Delta\mathbf{u}_A$ from (28), i.e. $\Delta\mathbf{u}_{A,k}$ is then used as control input for the inner velocity controller. Since the model predictive control is able to specifically constrain the full state $\mathbf{z}_k$, including the sway angles α_k and β_k, the use of model predictive control is advantageous for the performance of the closed-loop system.

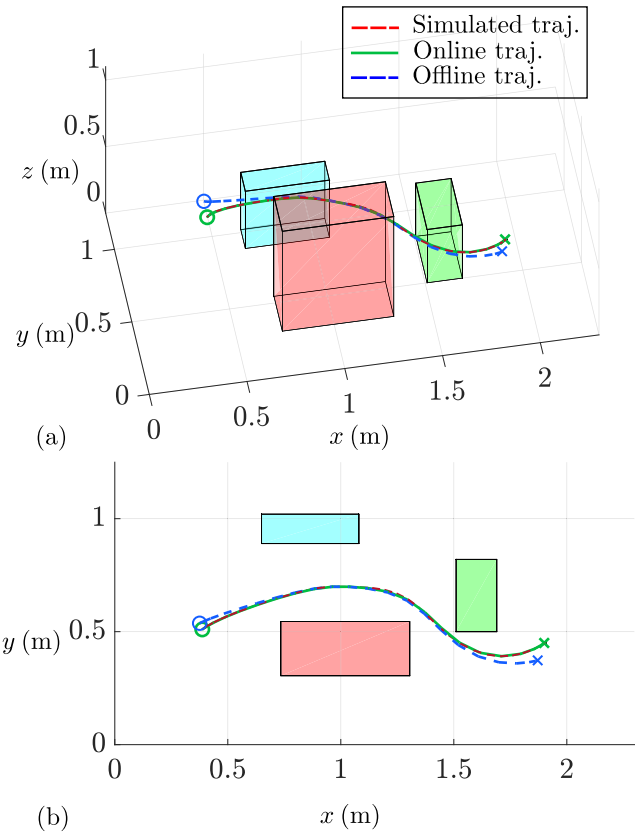

Fig. 7. Collision-free trajectories in the stationary target scenario: closest trajectory from the offline database, online replanned trajectory, and simulated trajectory using the trajectory tracking controller. The circle and the cross symbol constitute the starting and the target point, respectively. (a) Trajectories in 3D space, (b) Trajectories in the xy-plane. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Simulation and experimental results

In this section, simulation and experimental results are presented for two different scenarios. In the first scenario, the 3D gantry crane follows the trajectory from the online replanner connecting two points, with the states $\mathbf{z}_S$ and $\mathbf{z}_T$, in $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively, and the target is not changing during the motion of the crane. The second scenario concerns the case when the target can move freely within $\mathcal{X}_T$. Therefore, the online replanner must actively update the trajectory according to the new position of the moving target and its current speed. To prove the efficiency of the proposed combined method of fast trajectory optimization and trajectory tracking control, this paper assumes that the map containing all obstacles is known, i.e. the obstacles are static. This assumption is justified for many practical scenarios.

5.1. Simulation results

The offline trajectory optimization and the offline database are both developed in MATLAB/SIMULINK 2020b on a computer with 3.8 GHz Intel Core i7 and 32 GB RAM. The open-source package Interior Point OPTimize (IPOPT) was employed to solve the nonlinear optimization problem (2), see, e.g., Wächter and Biegler (2006). In IPOPT, the multifrontal linear solver (MA57) was used to increase the computational speed. The analytical gradients of the cost function and the constraint functions are computed using CasADi, see, e.g., Andersson et al. (2019). In addition, the numerical Hessian is evaluated using the BFGS approximation method, see, e.g., Liu and Nocedal (1989).

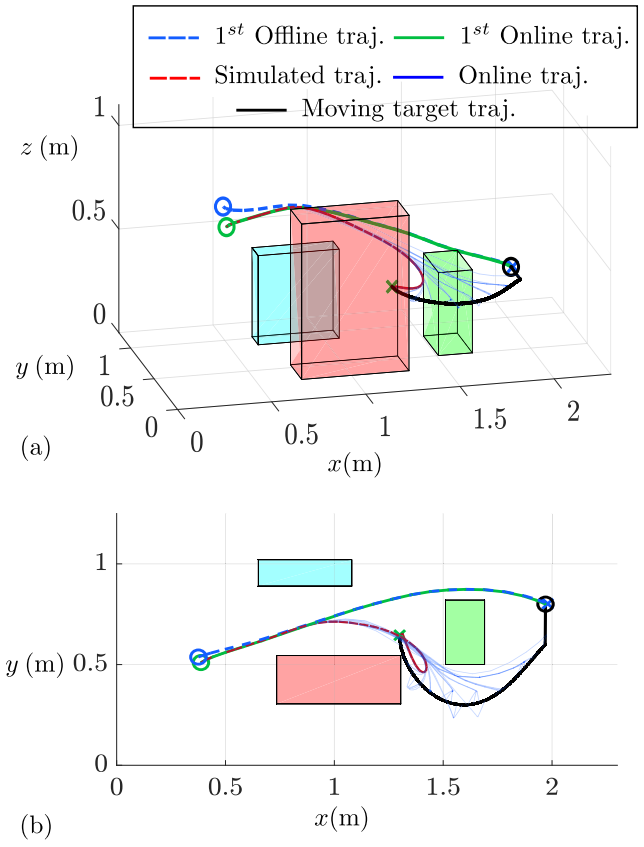

Fig. 8. The collision-free offline path and the online paths resulting from the online replanner for a moving target. (a) Trajectories in 3D space, (b) Trajectories in the xy-plane. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

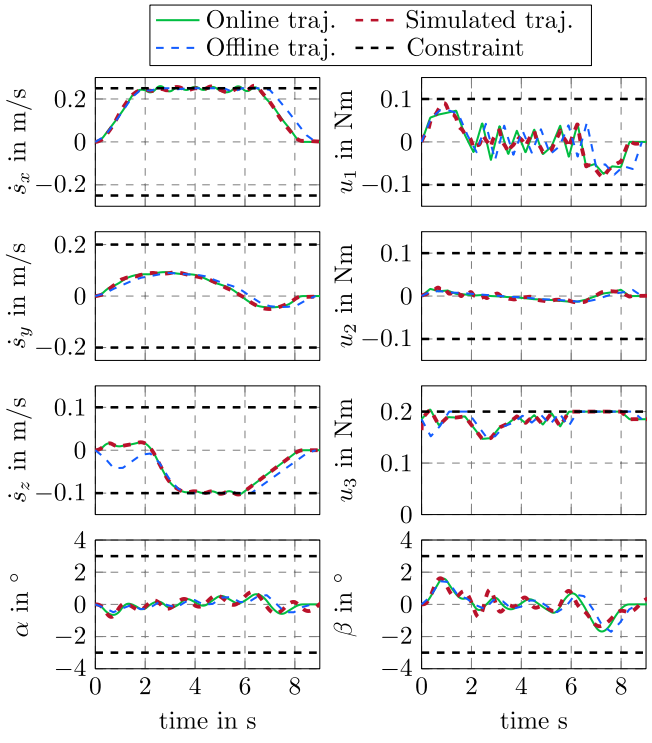

Fig. 9. Time evolution of the states and control inputs for the stationary target scenario in Fig. 7. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Each time-optimal obstacle-free trajectory in the offline database is discretized with 26 grid points, resulting in 339 optimization variables. Since the solution of the direct collocation optimization is interpolated between two neighboring grid points, the accuracy of the resulting trajectory depends on the number of grid points. However, there is a trade-off between the number of grid points and the computational speed. Because the workspace is approximately a cuboid of size 2 m × 1 m × 0.55 m, the average traversal time from the initial location to the most distant target location is approximately 9 s. Thus, the solution accuracy of direct collocation optimization with 26 grid points turns out to be sufficient for this application. Furthermore, considering the memory requirements for the offline database in the dSPACE MicroLabBox of the real experiment, up to 12.000 trajectories can be stored in flash memory with 339 variables of type double per trajectory. For a large scene, such as a factory or a port, the number of required grid points, but also the required accuracy, scales accordingly. However, the computational power can be increased depending on the requirements. In this paper, we present a proof of concept from a theoretical and an experimental point of view on a lab-scale 3D gantry crane.

In addition, the sparsity of the matrices is exploited to reduce memory consumption. The starting and target points lie in the subspaces $\mathcal{X}_S$ (cuboid of size 1.8 m × 0.8 m × 0.55 m) and $\mathcal{X}_T$ (plane of size 1 m × 0.8 m), shown as gray box and yellow plane in Fig. 4, respectively. Based on the offline trajectory planning algorithm, a database of collision-free trajectories is calculated connecting each grid point in $\mathcal{X}_S$ with each grid point in $\mathcal{X}_T$. Even for coarse grids in the offline database, the online trajectory replanner shows a high success rate, i.e. provides feasible trajectories, in the Monte Carlo simulation, see Vu et al. (2020). In this work, a fixed number of grid points $n_S = 12 \times 8 \times 3$ and $n_T = 10 \times 6$ was chosen for $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively. This results in a total of 11.520 near time-optimal collision-free offline trajectories in the database after removing the invalid trajectories having its starting or target state inside an obstacle. Note that the average computation time for a single trajectory in the database is approximately 50 ms. The user-defined weighting parameters $\mathrm{diag}(\rho_n, n = 1, \dots, 5) = \mathrm{diag}(4, 2.5, 3.5, 2, 2)$ are used in the k-nearest neighbor search in Algorithm 1.

To illustrate the overall concept consisting of the offline trajectory database, the online trajectory replanner and the underlying trajectory tracking controller, two example cases are shown in Figs. 7 and 8:

- Fig. 7 shows the results of the stationary target scenario where the offline trajectory (dashed blue line) is deformed to the online trajectory (dashed green line) according to a given pair of starting and target positions (green circle and cross symbols). The dashed red path illustrates the simulated trajectory of the trajectory tracking controller, which is perfectly tracked with respect to the trajectory generated by the online trajectory replanner. The time evolution of the corresponding states $\dot{s}_x$, $\dot{s}_y$, $\dot{s}_z$, α and β as well as the three control inputs u_1, u_2 and u_3 are depicted in Fig. 9. The green lines refer to the online trajectories, which deviate from the offline trajectory (blue dashed line). The simulated trajectory (red dashed line) shows a good tracking performance of the system. Also, the state and input constraints according to (18c), depicted as black dashed lines, are well respected.
- Fig. 8 shows the performance of the online replanner in the case of a moving target. The first solution of the online trajectory replanner (i.e. the green path) leads the gantry crane to the left side of the green obstacle. Later, the online replanner generates completely different trajectories moving around the right side of the green obstacle (i.e. multiple blue lines). Analogous to Fig. 9, the simulation results of the moving target scenario are depicted in Fig. 10. The four colored square symbols and the corresponding colored dashed lines represent the offline trajectories that are taken from the offline database. Note that the blue dashed line is the first closest trajectory, while the other three colored dashed lines illustrate offline trajectories in the database at selected time instances, which are chosen by the online replanner during the movement of the 3D gantry crane.

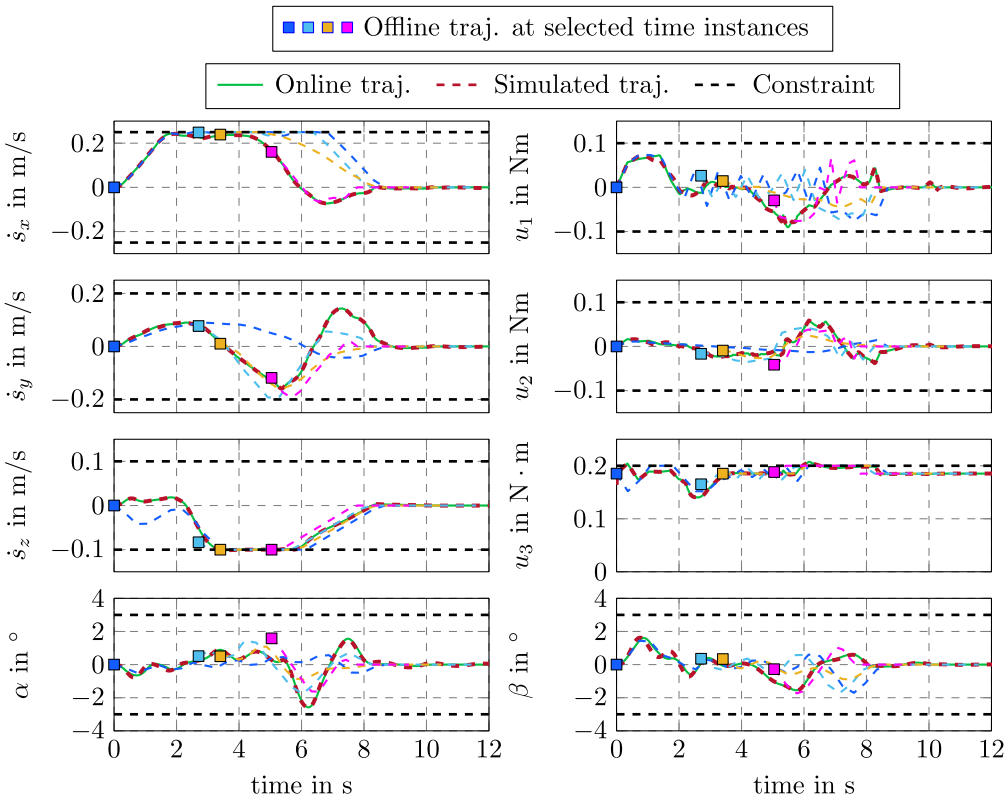

Fig. 10. Simulations of the states and control inputs for the moving target scenario in Fig. 8. The four colored squares and the corresponding dashed lines illustrate the offline trajectories which are utilized by the online replanner at selected time instances. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

It should be noted that small violations of the constraints are possible between two adjacent collocation points. The simulation results clearly show that the proposed concept is able to significantly replan the offline trajectory while adhering to the constraints and achieving near time-optimal execution. It can be nicely seen that the 3D gantry crane is driven in one of the limits for most of the time, which shows that the admissible range of the system is fully exploited.

Monte Carlo simulations were performed for both scenarios to investigate the versatility and robustness of the proposed approach. In the Monte Carlo simulation of the stationary target scenario, 10^4 trajectories were planned and simulated using pairs of randomly uniformly distributed starting and target states from the two subspaces $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively. In the moving target scenario, the following procedure was repeated 10^4 times. First, a collision-free trajectory for the moving target was generated, where the target moves from a random location A to a random location B within the target subspace $\mathcal{X}_T$. Second, a random starting state was chosen in the starting subspace $\mathcal{X}_S$. Third, the scenario including the online trajectory replanner and the trajectory tracking controller was simulated. All simulations were run in Simulink rapid accelerator mode. After each simulation, two result flags, i.e. the simulation fail flag and the collision check flag, were collected. The simulation fail flag corresponds to an unexpected numerical error during the simulation, i.e., an infeasible trajectory. The collision check flag indicates that the online trajectory collides with obstacles. The statistics of the flags are shown in Table 1. The average computation time of the online replanner is 2 ms for the stationary and 2.5 ms for the moving target scenario, respectively. Since the online replanner must actively find the closest trajectory in the third layer of the database for moving targets, a longer computation time was expected. It should be noted that not all trajectories are collision-free, since the obstacles for the online trajectory replanner are only approximated by the Hessian of the artificial potential functions. This is reflected in the success rate in Table 1. The simulation only fails in the moving target scenario when the online replanner is unable to generate a feasible trajectory corresponding to the moving target.

5.2. Experimental setup

The experimental setup shown in Fig. 11 consists of the 3D lab-scale gantry crane equipped with five incremental encoders and a dSPACE MicroLabBox. The online trajectory replanner and controller are implemented in Matlab/Simulink and are compiled and deployed on the dual-core real-time processor of the dSPACE MicroLabBox. The online trajectory replanner runs on core 2 with the sampling time $T_{s,2} = 15$ ms, while all other tasks such as the state measurement, friction compensation, and the cascaded controller are sampled with the faster sampling time $T_{s,1} = 1$ ms and run on core 1. In addition, a six-camera OptiTrack system is connected to dSPACE via Ethernet and is used to estimate the position of a remote-controlled moving truck (target) in the workspace.

5.3. Experimental results

In the experiment of the moving target scenario, a truck following the colored path at a random speed is used as the moving target. Without loss of generality, the position of the truck is measured using the OptiTrack system. For a real application in a larger or more complex environment, there are many ways to determine the position of a moving truck, e.g., using GPS or vision-based object tracking. The proposed online trajectory replanner and trajectory tracking controller are implemented on a dSPACE MicroLabBox real-time system with the sampling times of $T_{s,1} = 1$ ms and $T_{s,2} = 15$ ms on the cores 1 and 2, respectively. To solve the LCQP (18) in the online trajectory replanner with box constraints, the CVXgen package, see, e.g., Mattingley and Boyd (2012), is used to generate optimized C code. Note that the time step $h = (t_F^* + \delta t_F)/N$ of the online trajectory is much larger than the

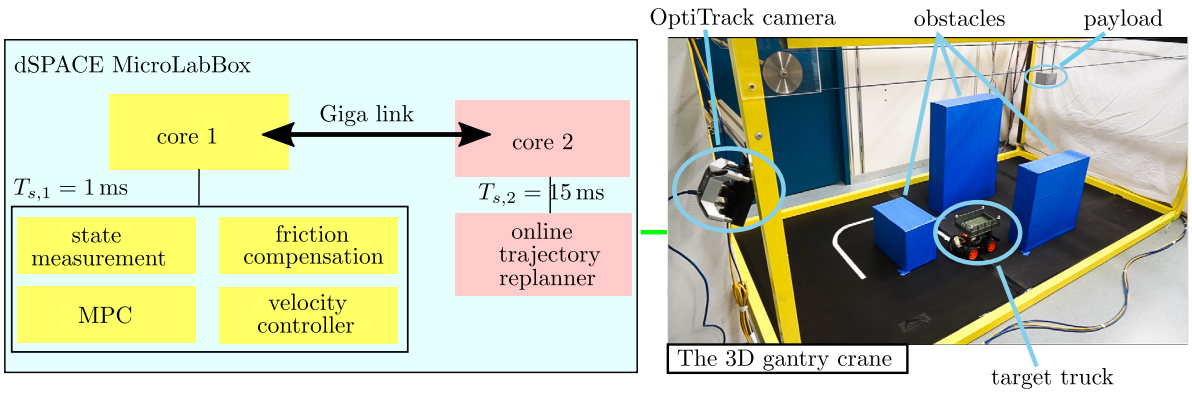

Fig. 11. System overview of the 3D laboratory gantry crane.

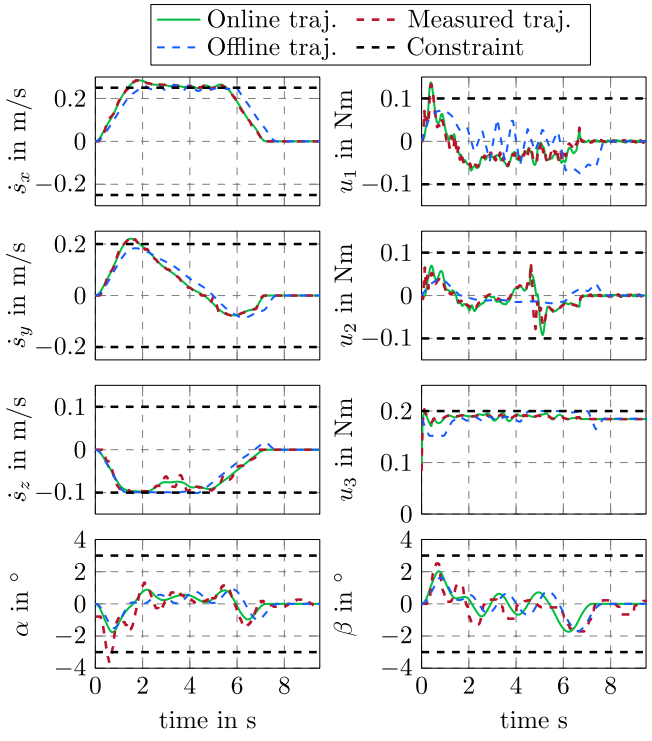

Fig. 12. Measurements of the system state and control inputs in the stationary target scenario. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sampling time $T_{s,1}$ of the controller. Therefore, the desired trajectory is interpolated before it is fed to the controller.

Similar to the previous subsection, experimental results are demonstrated for a stationary and a moving target scenario shown in Figs. 12 and 13, respectively.

- Fig. 12 depicts the experimental results of the stationary target scenario containing the measurements of the system states $\dot{s}_x$, $\dot{s}_y$, $\dot{s}_z$, α and β and the control inputs u_1, u_2 and u_3. The measured signals and the desired trajectories are depicted as dashed red lines and solid green lines, respectively. All measured signals satisfy the state and input constraints (black dashed lines) given by (18c). In addition, the online trajectories (solid green lines) of the system states slightly deviate from the closest offline trajectories (dashed blue lines). Note that the traversal time of the online trajectory differs from the offline trajectory due to δt_F in the LCQP (18).
- In a similar way as in Fig. 12, the experimental results of the moving target scenario are shown in Fig. 13. The four colored square symbols and the corresponding dashed lines represent the offline trajectories obtained by the search algorithm from the offline database. Note that online trajectory generation is indeed sufficient in satisfying the system state and input constraints at the collocation points. However, since an interpolation scheme is applied between two adjacent collocation points of the desired trajectory, small violations of the system state constraints may occur between the collocation points. Snapshots of the gantry crane and target truck motions are presented in Fig. 14. At the turning point (green circle in Fig. 14), the gantry crane has not come to a stop, but follows the motion of the moving truck smoothly. This turnaround point is approximately at the time $t = 7\,\text{s}$ where the system state velocities $\dot{s}_x, \dot{s}_y$ and $\dot{s}_z$ cross the zero line, see Fig. 13.

Note that in Figs. 12 and 13, although the gantry crane is traveling up to its full speed, i.e., the velocities $\dot{s}_x$, $\dot{s}_y$, and $\dot{s}_z$ reach the respective limits, the sway angles α and β remain within a small range of $[-0.05, 0.05]$ rad ($\approx \pm 3°$).

Overall, the trajectory tracking controller exhibits a good tracking performance, while the deviations of the pendulum angles α and β from the desired trajectory are clearly visible in both scenarios. These deviations can be attributed to model uncertainties and backlash caused by the measurement mechanism of the lab-scale 3D gantry crane. Moreover, a video of the discussed scenarios shown in Figs. 12 and 13 is available at https://www.acin.tuwien.ac.at/en/65ce/

6. Conclusions

In this paper, we considered the lab-scale experiment of a 3D gantry crane with the task to pick up a payload at an arbitrary position and then deposit it on a moving truck in a workspace with static obstacles. The time from picking up the payload to its deposition should be kept as small as possible and at the same time collisions with obstacles must be avoided and the system state and control input constraints must be respected. Although there are many studies on trajectory planning and control of 3D gantry cranes in the literature, the considered scenario cannot be solved with state-of-the-art concepts.

Therefore, a novel two-step trajectory planning algorithm, consisting of an offline trajectory optimization and an online trajectory replanner, in combination with an MPC (model predictive control)-based trajectory tracking controller is proposed in this paper. The offline trajectory optimization is used to generate a collision-free and dynamically feasible trajectory database. The computation time for an offline trajectory in the database is about 50 ms on average on a standard PC, which is too long for real-time planning in the moving-truck scenario. The idea of the online trajectory planner is to minimize the deviation from a suitable reference trajectory, which is selected according to a specifically designed strategy from the database, by solving a linear constrained quadratic program. This is computationally

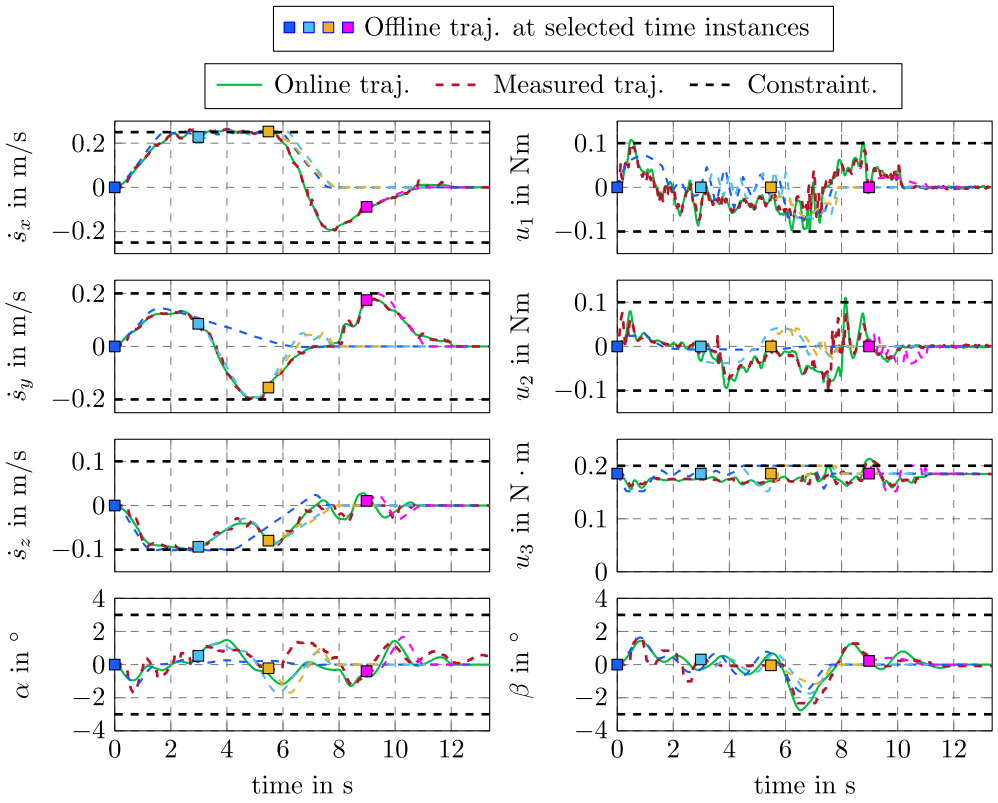

Fig. 13. Measurements of states and control inputs over traversal time for the moving target scenario in Fig. 14. The four colored squares and the corresponding dashed lines illustrate the offline trajectories taken by the online replanner at selected time instances. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

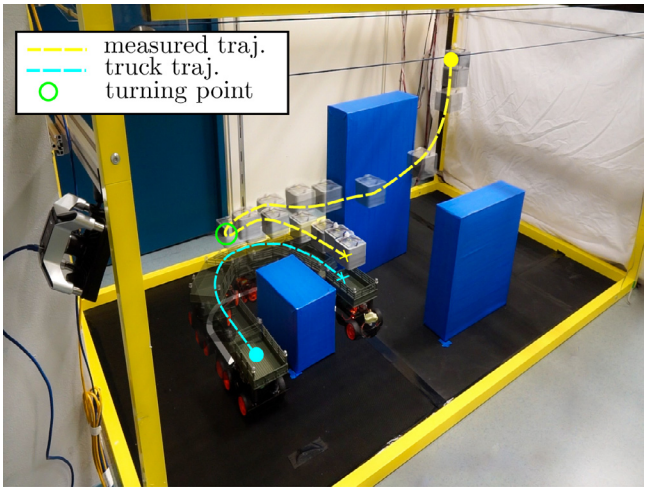

Fig. 14. Illustration of the experimental execution of the online trajectory replanning for a moving target. The collision-free online path is shown as dashed yellow line. The cyan dashed line is the path of the moving truck. The dot and cross symbols represent the starting and target positions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

very efficient because it yields a feasible trajectory within an average computation time of 2.5 ms on a standard PC and is thus 25 times faster than the offline trajectory optimization. The online trajectory then serves as a reference input for an MPC-based tracking controller. Here an MPC was employed to be able to systematically account for state and control input constraints. Simulation studies and experimental results demonstrate the feasibility of the proposed approach.

This work was intended to give a proof of concept in form of a laboratory experiment for real-time trajectory planning in a dynamically changing environment (static obstacles, moving truck) for the autonomous loading of trucks. Clearly, in order to reduce the complexity of the lab-scale experiment, we resorted to an OptiTrack system for detecting the truck position. For real applications in a larger or more complex environment, we have to use other methods, as for instance GPS and/or vision-based obstacle detection and tracking methods. Currently, we are extending the algorithm for dynamically changing obstacles and in a mid-term perspective we plan to combine the real-time trajectory planning with environment detection and vision-based object tracking.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

Appendix A

In this section, a detailed derivation of the equations of motion (1) for the 3D gantry crane is presented. The potential energy of the gantry crane in terms of the generalized coordinates $\mathbf{q}$ reads as

$$V = m_z g(h_2 + \cos(\alpha)\cos(\beta)s_z - \cos(\beta)h_1), \tag{29}$$

with the mass of the payload m_z, see Figs. 1 and 2. The kinetic energy of the system is given by

$$T = \frac{1}{2} m_z \dot{\mathbf{r}}^{\mathrm{T}} \dot{\mathbf{r}} + \frac{1}{2}(m_x + m_y)\dot{s}_x^2 + \frac{1}{2} m_y \dot{s}_y^2 + \sum_i \frac{J_i \dot{s}_i^2}{R_i^2}, \tag{30}$$

where $\mathbf{r}(\mathbf{q})$ is the position of the center of mass (CoM) of the payload in the inertial frame from (4), m_x is the mass of the bridge, m_y denotes the mass of the trolley and hoisting drum, J_i is the moment of inertia of the axis $i \in \{x, y, z\}$, and R_i is the radius of the associated sprocket wheels. The equations of motion of the 3D gantry crane are derived by using the Euler–Lagrange equations with the Lagrangian $L = T - V$, leading to (1). With the expressions

$$m_X = m_x + m_y + m_z + \frac{J_x}{R_x^2}$$
$$m_Y = m_y + m_z + \frac{J_y}{R_y^2}$$
$$m_Z = m_z + \frac{J_z}{R_z^2},$$

the entries of the system matrices

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} M_{11} & \cdots & M_{15} \\ \vdots & \ddots & \vdots \\ M_{51} & \cdots & M_{55} \end{bmatrix}, \quad \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} C_{11} & \cdots & C_{15} \\ \vdots & \ddots & \vdots \\ C_{51} & \cdots & C_{55} \end{bmatrix},$$
$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} g_1 \\ \vdots \\ g_5 \end{bmatrix}$$

follow as

$$M_{11} = m_X$$
$$M_{13} = \cos(\alpha)\sin(\beta) m_z$$
$$M_{14} = -\sin(\alpha)\sin(\beta) s_z$$
$$M_{15} = m_z \cos(\beta)(s_z \cos(\alpha) - h_1)$$
$$M_{22} = m_Y$$
$$M_{23} = -m_z \sin(\alpha)$$
$$M_{24} = -m_z s_z \cos(\alpha)$$
$$M_{31} = m_z \cos(\alpha)\sin(\beta)$$
$$M_{32} = -m_z \sin(\alpha)$$
$$M_{33} = m_Z$$
$$M_{41} = -m_z s_z \sin(\alpha)\sin(\beta)$$
$$M_{42} = -m_z s_z \cos(\alpha)$$
$$M_{44} = m_z s_z^2$$
$$M_{51} = m_z \cos(\beta)(s_z \cos(\alpha) - h_1)$$
$$M_{55} = -m_z(-s_z \cos(\alpha) + h_1)^2$$
$$M_{12} = M_{21} = M_{25} = M_{34} = M_{35} = 0$$
$$M_{43} = M_{45} = M_{52} = M_{53} = M_{54} = 0,$$

$$C_{13} = -m_z(\cos(\alpha)\cos(\beta)\dot{\beta} - \sin(\alpha)\sin(\beta)\dot{\alpha})$$
$$C_{14} = -m_z(\sin(\alpha)(\sin(\beta)v_z + \dot{\beta}\cos(\beta)s_z) + \sin(\beta)\cos(\alpha)\dot{\alpha}s_z)$$
$$C_{15} = m_z((-\cos(\alpha)\sin(\beta)\dot{\beta} - \sin(\alpha)\cos(\beta)\dot{\alpha})s_z + \cos(\alpha)\cos(\beta)v_z + \dot{\beta}\sin(\beta)h_1)$$
$$C_{23} = -\cos(\alpha) m_z \dot{\alpha}$$
$$C_{24} = m_z(-\cos(\alpha)v_z + \sin(\alpha)\dot{\alpha}s_z)$$
$$C_{34} = -m_z \dot{\alpha} s_z$$
$$C_{35} = \dot{\beta}\cos(\alpha) m_z(-\cos(\alpha)s_z + h_1)$$
$$C_{43} = m_z s_z \dot{\alpha}$$
$$C_{44} = m_z s_z v_z$$
$$C_{45} = m_z \dot{\beta}\sin(\alpha)(\cos(\alpha)s_z^2 - h_1 s_z)$$
$$C_{53} = m_z \dot{\beta}\cos(\alpha)(\cos(\alpha)s_z - h_1)$$
$$C_{54} = -m_z \dot{\beta}\sin(\alpha)(\cos(\alpha)s_z^2 - h_1 s_z)$$
$$C_{55} = m_z(-\cos(\alpha)\sin(\alpha)\dot{\alpha}s_z^2 + (-\cos(\alpha)^2 v_z + \sin(\alpha)\dot{\alpha}h_1)s_z - \cos(\alpha)h_1 v_z)$$
$$C_{11} = C_{12} = C_{21} = C_{22} = C_{25} = 0$$
$$C_{31} = C_{32} = C_{33} = C_{41} = C_{42} = C_{51} = C_{52} = 0,$$

$$g_1 = g_2 = 0$$
$$g_3 = m_z \cos(\alpha)\cos(\beta) g$$
$$g_4 = -m_z g \sin(\alpha)\cos(\beta) s_z$$
$$g_5 = m_z g(-\cos(\alpha)\sin(\beta)s_z + \sin(\beta)h_1).$$

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.conengprac.2022.105255.

References

Abdullahi, A., Mohamed, Z., Selamat, H., Pota, H., Abidin, M. Z., & Fasih, S. (2020). Efficient control of a 3D overhead crane with simultaneous payload hoisting and wind disturbance: design, simulation and experiment. *Mechanical Systems and Signal Processing*, *145*, Article 106893.

An, N. T., Giles, D., Nam, N. M., & Rector, R. B. (2016). The log-exponential smoothing technique and nesterov's accelerated gradient method for generalized sylvester problems. *Journal of Optimization Theory and Applications*, *168*(2), 559–583.

Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, *21*(2), 193–207.

Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming*. Philadelphia, USA: Siam.

Blajer, W., & Kołodziejczyk, K. (2007). Motion planning and control of gantry cranes in cluttered work environment. *IET Control Theory & Applications*, *1*(5), 1370–1379.

Böck, M., & Kugi, A. (2013). Real-time nonlinear model predictive path-following control of a laboratory tower crane. *IEEE Transactions on Control Systems Technology*, *22*(4), 1461–1473.

Chai, R., Tsourdos, A., Savvaris, A., Chai, S., & Xia, Y. (2018). Two-stage trajectory optimization for autonomous ground vehicles parking maneuver. *IEEE Transactions on Industrial Informatics*, *15*(7), 3899–3909.

Chen, H., Fang, Y., & Sun, N. (2016). Optimal trajectory planning and tracking control method for overhead cranes. *IET Control Theory & Applications*, *10*(6), 692–699.

Chen, H., Yang, P., & Geng, Y. (2019). A time optimal trajectory planning method for overhead cranes with obstacle avoidance. In *Proceedings of IEEE/ASME international conference on advanced intelligent mechatronics (AIM)* (pp. 697–701).

Fliess, M., Lévine, J., Martin, P., & Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, *61*(6), 1327–1361.

Iftikhar, S., Faqir, O. J., & Kemgan, E. C. (2019). Nonlinear model predictive control of an overhead laboratory-scale gantry crane with obstacle avoidance. In *Proceedings of the conference on control technology and applications (CCTA)* (pp. 382–387).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement limitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE conference on robotics and automation (ICRA)* (pp. 1398–1403).

Kelly, M. (2017). An introduction to trajectory optimization: how to do your own direct collocation. *SIAM Review*, *59*(4), 849–904.

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, *27*(5), 943–957.

Kim, J., Kiss, B., Kim, D., & Lee, D. (2021). Tracking control of overhead crane using output feedback with adaptive unscented Kalman filter and condition-based selective scaling. *IEEE Access*, *9*, 108628–108639.

Kolar, B., Rams, H., & Schlacher, K. (2017). Time-optimal flatness based control of a gantry crane. *Control Engineering Practice*, *60*, 18–27.

Lembono, T. S., Paolillo, A., Pignat, E., & Calinon, S. (2020). Memory of motion for warm-starting trajectory optimization. *IEEE Robotics and Automation Letters*, *5*(2), 2594–2601.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1–3), 503–528.

Lobe, A., Ettl, A., Steinboeck, A., & Kugi, A. (2018). Flatness-based nonlinear control of a three-dimensional gantry crane. *IFAC-PapersOnLine*, *51*(22), 331–336.

Lu, B., Cao, H., Hao, Y., Lin, J., & Fang, Y. (2021). Online antiswing trajectory planning for a practical rubber tire container gantry crane. *IEEE Transactions on Industrial Electronics*, 1.

Mattingley, J., & Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, *13*(1), 1–27.

Nielsen, F., & Sun, K. (2016). Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities. *Entropy*, *18*(12), 442.

Pinkham, R., Zeng, S., & Zhang, Z. (2020). Quicknn: Memory and performance optimization of kd tree based nearest neighbor search for 3d point clouds. In *Proceedings of the IEEE international symposium on high performance computer architecture (HPCA)* (pp. 180–192).

Rao, A. V. (2009). A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, *135*(1), 497–528.

Sawodny, O., Aschemann, H., & Lahres, S. (2002). An automated gantry crane as a large workspace robot. *Control Engineering Practice*, *10*(12), 1323–1338.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., & Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, *33*(9), 1251–1270.

Soleymani, M., & Morgera, S. (1987). An efficient nearest neighbor search method. *IEEE Transactions on Communications*, *35*(6), 677–679.

Vu, M., Zips, P., Lobe, A., Beck, F., Kemmetmüller, W., & Kugi, A. (2020). Fast motion planning for a laboratory 3D gantry crane in the presence of obstacles. *IFAC-PapersOnLine*, *54*(1), 7–12.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57.

Wang, X., Liu, J., Zhang, Y., Shi, B., Jiang, D., & Peng, H. (2019). A unified symplectic pseudospectral method for motion planning and tracking control of 3D underactuated overhead cranes. *International Journal of Robust and Nonlinear Control*, *29*(7), 2236–2253.

Zhang, W., Chen, H., Chen, H., & Liu, W. (2021). A time optimal trajectory planning method for double-pendulum crane systems with obstacle avoidance. *IEEE Access*, *9*, 13022–13030.

Zhang, X., Liniger, A., & Borrelli, F. (2020). Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, *29*(3), 972–983.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *International Journal of Robotics Research*, *32*(9–10), 1164–1193.