Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Fast motion planning for a laboratory 3D gantry crane in the presence of obstacles

M.N. Vu * **P. Zips** ** **A. Lobe** ** **F. Beck** * **W. Kemmetmüller** *
A. Kugi *,**

* *Automation and Control Institute (ACIN), TU Wien, Vienna, Austria (e-mail: {vu,beck,kemmetmueller,kugi}@acin.tuwien.ac.at).*
** *Center for Vision, Automation & Control, AIT Austrian Institute of Technology Vienna, Austria (e-mail: {amadeus.lobe,patrik.zips}@ait.ac.at)*

Abstract: In this paper, a concept is presented for the fast motion planning of a 3D laboratory crane in a static environment with obstacles taking into account the dynamic constraints on the state variables and control inputs. The focus is set on the possibility of a fast (re)planning if the starting and/or target state is changing. The proposed concept consists of two parts: an offline trajectory planner to set up a database of collision-free, time optimal trajectories from the starting to the target space, with an average computing time of 0.17 s for one trajectory, and an online planner based on a constrained quadratic program, with an average computing time of 7 ms for one trajectory. Simulation results for a validated mathematical model demonstrate the feasibility of the proposed approach.

Keywords: fast optimal trajectory planning, online re-planning, direct collocation method, constrained quadratic optimization, collision avoidance, 3D gantry crane.

1. INTRODUCTION

Motion planning is an important task in robotics. There are a number of different concepts in the literature, which can be essentially divided into two groups. In the first group, a feasible path is searched for and then the redundant and jerky motion are removed by trajectory optimization. For this, the continuous state space is discretized into grids for graph-based search methods or randomly sampled for sampling-based search strategies, see, e.g., Kavraki et al. (1996), Karaman and Frazzoli (2011). Grid-based methods such as A* and D*, see, e.g., Hart et al. (1968); Ferguson and Stentz (2005), respectively, are called resolution complete path planners because they split the configuration space into implicit grids. While these approaches are successfully used in many applications such as manipulation planning and kino-dynamics planning, see, e.g., Kondo (1991), and Cherif (1999), their computational effort significantly increases if a higher quality solution is requested. On the other hand, sampling-based methods such as RRT*, see, e.g., Karaman et al. (2011), and Informed RRT*, see, e.g., Gammell et al. (2014), provide the guarantee for the asymptotic optimality of the solution. However, sampling-based methods do not take into account the dynamical constraints, which are nonlinear in general. Therefore, and in order to smooth the trajectory, further post processing is required.

In the second group, optimization is directly applied to find a locally optimal, collision-free trajectory which accounts for all dynamical constraints of the system, see, e.g., Betts (1998), Rao (2009). Covariant Hamiltonian Optimization for Motion Planning (CHOMP), developed by Zucker et al. (2013), turns out to be one of the most successful algorithms. The following key features are taken into account by CHOMP. Firstly, the trajectory costs are formulated to be invariant with respect to the time parametrization. Secondly, the precalculated signed distance field (SDF), which relies on the Euclidean distance transform (EDT), see, e.g., Maurer et al. (2003), provides the global distance and its gradient to obstacle surfaces. The numerical optimization solver uses a preconditioned gradient to find the locally optimal trajectory. CHOMP has proven its effectiveness in several applications including the Little-Dog quadruped, PR2 robot, and the HERB mobile manipulation platform, see, e.g., Zucker et al. (2013). More recently, Schulman et al. (2014) introduced the TrajOpt optimization package which is motivated by CHOMP. There are differences between CHOMP and TrajOpt such as the approach for collision detection and the numerical optimization scheme. Instead of formulating the costs to be invariant with respect to time, TrajOpt uses the time parametrization for the cost functions. Sequential Quadratic Programming (SQP) is used as the numerical optimization scheme implemented in TrajOpt. Moreover, the obstacle avoidance relies on the convex-convex collision checking approach, which takes two shapes (e.g. the robot's link and the obstacle) and computes the minimal translation between them by the Gilbert-Johnson-Keerthi (GJK) algorithm, see, Gilbert et al. (1988). The TrajOpt package can be straightforwardly used for both kinematic and dynamic constraints. While these algorithms are quite powerful and already exhibit decent computational efficiency, they do not exploit the knowledge of the previously planned trajectories if only the starting and/or the target state are slightly changed. Thus, several approaches solve the replanning task by creating an offline library

for the trajectories and for instance generate the new trajectories based on dynamic movement primitives (DMP) or Gaussian Mixture Models (GMM), see, e.g., Ijspeert et al. (2002), Khansari-Zadeh and Billard (2011). However, this strategy becomes computationally inefficient when the state space grows and these algorithms normally do not take into account the dynamical feasibility of the generated motion. Pekarovskiy et al. (2017) used the Laplacian trajectory editing method to preserve the local trajectory properties through the least-squares method by fixing a set of positional constraints. The local trajectory properties can be velocity deviation, acceleration deviation or jerk deviation.

In this work, we consider the following scenario. A 3D gantry crane has to move goods or material from one dedicated place (starting space $\mathcal{X}_S$) to another place (target space $\mathcal{X}_T$) in a static environment with known obstacles. The main objective of this paper is the fast planning of a time-optimal trajectory from a given starting point to a given target point which systematically accounts for both the obstacles and the dynamic constraints on the state variables and control inputs. The offline planner provides a database of time-optimal and collision-free trajectories which connect the grid points of the starting space with those of the target space and respect the dynamic constraints. The online planner is able to calculate a new trajectory in a computationally very efficient way if the starting and/or target point do not conform to a grid point in the database. The main contribution of this paper is the design of this fast planning framework and the investigation of how the number of grid points in the starting and target space influence the computing speed and success rate of the online planner.

The paper is organized as follows: Section 2 shortly introduces the 3D laboratory gantry crane under consideration and provides the equations of motion. An offline planner based on the direct transcription method is presented in Section 3. The static obstacles are taken into account by potential functions based on the LogSumExp function and dynamic constraints enter the optimization problem as box constraints. Section 4 is concerned with the design of an online planner which selects the closest trajectory in the database and calculates the required deviation based on the solution of a constrained quadratic program. Simulation results for a validated mathematical model are presented in Section 5. The last section, Section 6, finally gives some conclusions.

2. EQUATIONS OF MOTION

The setup of the considered laboratory gantry crane is illustrated in Fig. 1. It is used to simulate the handling of coils in the steel industry. The hook is mounted on two ropes which are assumed to be identical, see Fig. 2. Thus, the gantry crane is able to operate in the three dimensional (3D) space. The position in x-direction s_x is controlled by the bridge belt motor, while movements in y-direction s_y are generated by the trolley motor. For lifting and lowering the hook, the hoisting drum performs the movement s_z in z-direction. According to Fig. 2, the orientation of the hook in 3D space is described by the two angles α and β in the (zy) and the (zx) plane, respectively.

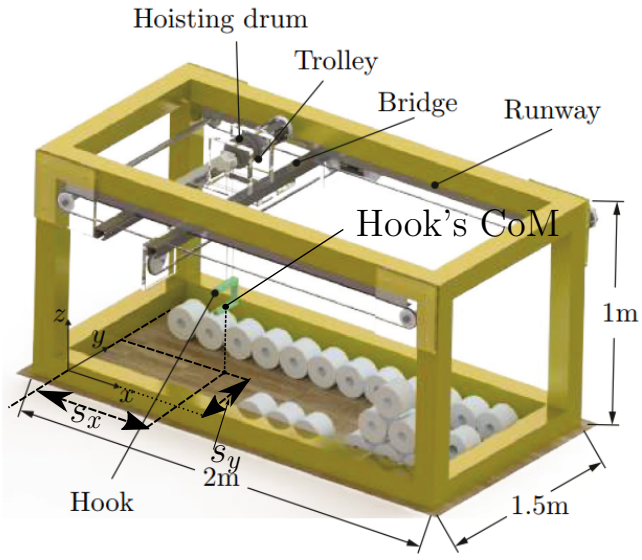

Fig. 1. Schematic of the considered gantry crane.

These angles are assumed to be identical for both ropes. Thus, the 3D gantry crane exhibits five degrees of freedom $\mathbf{q} = [s_x, s_y, s_z, \alpha, \beta]^T$.

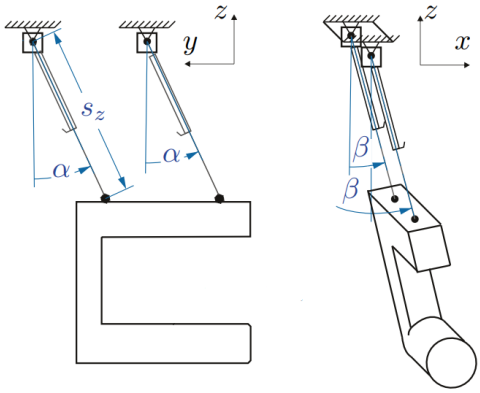

Fig. 2. The hook with the corresponding rope angles.

We assume that the massless ropes are always under tension. Hence, the gantry crane can be treated as a rigid-body system. The equations of motion of the gantry crane are derived by using the Euler-Lagrange equations, for more detail, see Lobe et al. (2018)

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{B}\mathbf{u}, \tag{1}$$

where $\mathbf{D}(\mathbf{q})$ denotes the symmetric and positive definite mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ comprises Coriolis and centrifugal terms, $\mathbf{g}(\mathbf{q})$ are the forces related to the potential energy, and $\mathbf{u} = [u_1, u_2, u_3]^T$ is the vector of driving torques at the x-, y-, and z-axis, respectively. Since the mass matrix $\mathbf{D}(\mathbf{q})$ is invertible, (1) can be rewritten in a more compact state-space form

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mathbf{u}), \tag{2}$$

with $\mathbf{z}^T = [\mathbf{q}^T, \dot{\mathbf{q}}^T]$.

3. OFFLINE TRAJECTORY PLANNING

In this section, a time-optimal trajectory is planned from a starting state $\mathbf{z}_S$ to a target state $\mathbf{z}_T$ taking into

account the constraints on the system dynamics as well as obstacles in the 3D space. Essentially, the approaches know from the literature can be classified into direct and indirect methods. In the following, the direct transcription method, see, e.g., Betts (2010), Kelly (2017), is applied by discretizing the trajectory into $N+1$ grid points, also named collocation points, and solving the discrete optimization problem

$$\min_{\boldsymbol{\xi}} J = t_F + \frac{1}{2}h\sum_{k=0}^{N-1}\sum_{i=1}^{m}\varphi_{i,k} \tag{3a}$$

$$\text{s.t. } \mathbf{z}_{k+1} - \mathbf{z}_k = \frac{1}{2}h(\mathbf{f}_k + \mathbf{f}_{k+1}) \tag{3b}$$

$$\mathbf{z}_0 = \mathbf{z}_S,\ \ \mathbf{z}_N = \mathbf{z}_T \tag{3c}$$

$$\underline{\mathbf{z}} \leq \mathbf{z}_k \leq \overline{\mathbf{z}},\ k = 0, ..., N \tag{3d}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}},\ k = 0, ..., N, \tag{3e}$$

with $\mathbf{z}_k^T = [\mathbf{q}_k^T, \dot{\mathbf{q}}_k^T]$, the optimization variables

$$\boldsymbol{\xi}^T = [t_F, \mathbf{z}_0^T, ..., \mathbf{z}_N^T, \mathbf{u}_0^T, ..., \mathbf{u}_N^T], \tag{4}$$

and the time step $h = t_F/N$. In (3a), $\varphi_{i,k}$ refers to the potential function $\varphi_i(\mathbf{q}_k)$ of the i^{th} obstacle, $i = 1, ..., m$, evaluated at the collocation points $\mathbf{q}_k$, $k = 0, ..., N-1$ and t_F is the time it takes to bring the system from the starting state $\mathbf{z}_S$ to the target state $\mathbf{z}_T$. Moreover, the system dynamics (2) is approximated using the trapezoidal rule with $\mathbf{f}_k = \mathbf{f}(\mathbf{z}_k, \mathbf{u}_k)$, and $\underline{\mathbf{z}}$, $\overline{\mathbf{z}}$, $\underline{\mathbf{u}}$, and $\overline{\mathbf{u}}$ denote the lower and upper bounds of the state and control input, respectively.

To solve the optimization problem (3a), we use the Interior Point OPTimize (IPOPT), an open source package based on the interior point method (IPM) for large scale nonlinear programming, see, e.g., Wächter and Biegler (2006). It is worth noting that the gradient of (3a) can be computed analytically in a straightforward way. After the optimal values have been found at the collocation points by solving (2), these values are interpolated in the interval $t \in [kh, (k+1)h]$

$$\begin{aligned}\mathbf{z}(t) &\approx \mathbf{z}_k + (t-kh)\mathbf{f}_k + \frac{(t-kh)^2}{2h}(\mathbf{f}_{k+1} - \mathbf{f}_k),\\ \mathbf{u}(t) &\approx \mathbf{u}_k + \frac{t-kh}{h}(\mathbf{u}_{k+1} - \mathbf{u}_k),\end{aligned} \tag{5}$$

for $k = 0, ..., N-1$.

Next, we will derive the potential function φ_i, $i = 1, ..., m$ of the obstacles in the working space. For this, let us consider that the obstacles can be embedded by boxes with the parameter vector ${}^{\mathbf{O}_i}\mathbf{p}_i = [w_i, h_i, d_i]^T$ 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 i^{th} box, $i = 1, ..., m$, see Fig. 3 for $m = 2$. Moreover, we assume that the location of the obstacles is known with ${}^{\mathbf{O}_i}_{\mathbf{I}}\mathbf{R}_i$ and ${}^{\mathbf{O}_i}_{\mathbf{I}}\mathbf{T}_i$ denoting the rotation matrix and the translation vector from the box frame $\{\mathbf{O}_i\}$ to the inertial frame $\{\mathbf{I}\}$. Thus, the position in the 3D workspace ${}^{\mathbf{I}}\mathbf{r} = [{}^{\mathbf{I}}r_x, {}^{\mathbf{I}}r_y, {}^{\mathbf{I}}r_z]^T$ associated to a point $\mathbf{q}$ on the trajectory can be expressed in the i^{th} box frame $\{\mathbf{O}_i\}$ in the form

$${}^{\mathbf{O}_i}\mathbf{r} = \left({}^{\mathbf{O}_i}_{\mathbf{I}}\mathbf{R}_i\right){}^{\mathbf{I}}\mathbf{r} + {}^{\mathbf{O}_i}_{\mathbf{I}}\mathbf{T}_i. \tag{6}$$

The point is considered as obstacle-free if and only if the following condition

$$\bar{S}_i(\mathbf{q}) = \min(\Delta p_{i,j})_{j=1,2,3} < 0 \tag{7}$$

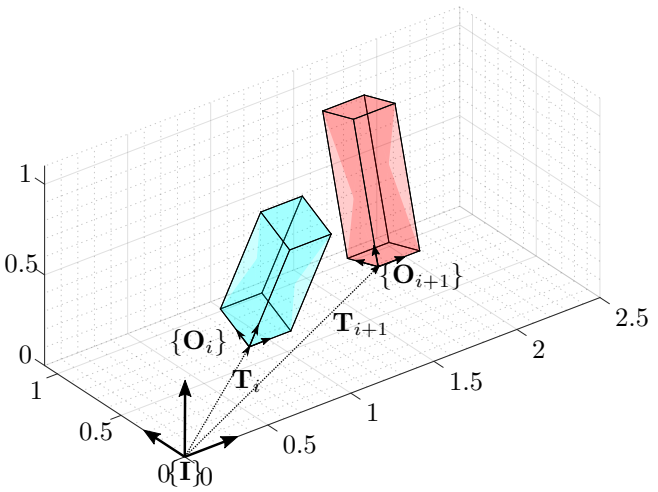

Fig. 3. Illustration of the obstacles in the working space and their coordinate frames.

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

$$\boldsymbol{\Delta}\mathbf{p}_i = \left({}^{\mathbf{O}_i}\mathbf{r}\right) \circ \left({}^{\mathbf{O}_i}\mathbf{p}_i - {}^{\mathbf{O}_i}\mathbf{r}\right). \tag{8}$$

The operator $\circ$ refers to the element-wise product. Based on (7), the potential function φ_i for the i^{th} obstacle can be defined as

$$\bar{\varphi}_i(\mathbf{q}) = \max(\gamma_i \bar{S}_i(\mathbf{q}), 0), \tag{9}$$

where $\gamma_i > 0$, $i = 1, ..., m$, is a user-defined scaling parameter. In order to render the potential function (9) and (7) sufficiently smooth, the LogSumExp function is employed, see, e.g., An et al. (2016), to obtain

$$\begin{aligned}S_i(\mathbf{q}) &= \frac{1}{\eta_1}\log\left(\sum_{j=1}^{3} \mathrm{e}^{\eta_1 \Delta p_{i,j}}\right)\\ \varphi_i(\mathbf{q}) &= \frac{1}{\eta_2}\log\left(1 + \mathrm{e}^{\eta_2 \gamma_i S_i(\mathbf{q})}\right),\end{aligned} \tag{10}$$

with the so-called softness coefficients $\eta_1 < 0$ and $\eta_2 > 0$. This allows to calculate the analytic gradient

$$\frac{\partial \varphi_i(\mathbf{q})}{\partial \mathbf{q}} = \gamma_i \frac{\mathrm{e}^{\gamma_i \eta_2 S_i(\mathbf{q})}}{1 + \mathrm{e}^{\gamma_i \eta_2 S_i(\mathbf{q})}} \frac{\partial S_i(\mathbf{q})}{\partial \mathbf{q}} \tag{11}$$

with

$$\frac{\partial S_i(\mathbf{q})}{\partial \mathbf{q}} = \sum_{j=1}^{3} \frac{\mathrm{e}^{\eta_1 \Delta p_{i,j}(\mathbf{q})}}{\sum_{j=1}^{3} \mathrm{e}^{\eta_1 \Delta p_{i,j}(\mathbf{q})}} \frac{\partial \Delta p_{i,j}(\mathbf{q})}{\partial \mathbf{q}}. \tag{12}$$

In order to create a safety margin around the obstacles, the margin $\boldsymbol{\delta} = [\delta_x, \delta_y, \delta_z]^T$ is added in the form ${}^{\mathbf{O}_i}\mathbf{p}_i = [w_i + \delta_x/2, h_i + \delta_y/2, d_i + \delta_z/2]^T$ and $\mathbf{T}_i$ is replaced by $\mathbf{T}_i - \boldsymbol{\delta}/2$.

4. ONLINE TRAJECTORY PLANNING

As already discussed in the introduction, we consider that the gantry crane operates in a static environment with m obstacles and the starting and target state, $\mathbf{z}_S$ and $\mathbf{z}_T$, lie within predefined subspaces of the 3D working space. For each subspace, an equally spaced grid is defined. Then, from every grid point of the starting subspace $\mathcal{X}_S$ to every grid point in the target subspace $\mathcal{X}_T$, a time-optimal trajectory is planned offline with the direct transcription method presented in Section 3 and stored in a database. This section deals with the design of an online trajectory

planning algorithm if the starting state $\bar{\mathbf{z}}_S$ and/or the target state $\bar{\mathbf{z}}_T$ do not conform to a grid point of the corresponding subspaces in the database.

In a first step, a computationally efficient algorithm was implemented to determine the closest grid points to $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$. This algorithm starts with identifying those hypercubes which contain $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$, respectively, and transform them to a unit hypercube with one corner at $[0,0,0]^T$ and the diagonally opposite one at $[1,1,1]^T$. Then a sorting algorithm is employed to discover the tetrahedron according to Fig. 4 in which the points $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$ lie, see, e.g., Moore (1992). By transforming $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$ to barycentric coordinates with respect to the coordinates of the four corner points of the tetrahedron, see, e.g., Davies (1996), it can be directly seen which of the four corners is the closest point. This procedure is computationally more efficient than computing the distances to the eight corners of the corresponding hypercubes.

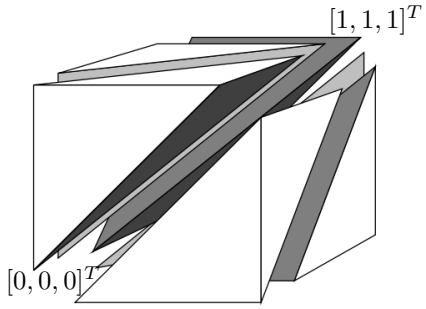

Fig. 4. Kuhn triangulation of a cube by Davies (1996).

Once selected the closest grid points with respect to $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$, the time-optimal trajectory connecting these points in the database is selected with

$$(\boldsymbol{\xi}^*)^T = [(t_F^*)^T, (\mathbf{z}_0^*)^T, ..., (\mathbf{z}_N^*)^T, (\mathbf{u}_0^*)^T, ..., (\mathbf{u}_N^*)^T]. \quad (13)$$

In a second step, we utilize $\boldsymbol{\xi}^*$ to calculate the trajectory which connects $\bar{\mathbf{z}}_S$ with $\bar{\mathbf{z}}_T$. If the grid in the starting and target subspace is sufficiently dense, it can be expected that only small deviations $\delta\boldsymbol{\xi} = [\delta t_F, \delta\mathbf{z}_0, ..., \delta\mathbf{z}_N, \delta\mathbf{u}_0, ..., \delta\mathbf{u}_N]^T$ are required to calculate the new trajectory. Therefore, the discrete-time system dynamics (3b) can be linearized around $\boldsymbol{\xi}^*$ in the form

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

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

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

for $k = 0, ..., N-1$. Note that

$$\delta\mathbf{z}_0 = \bar{\mathbf{z}}_S - \mathbf{z}_0^* \text{ and } \delta\mathbf{z}_N = \bar{\mathbf{z}}_T - \mathbf{z}_N^* \quad (15)$$

are fixed due to the given starting and target state $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$. The deviation $\delta\boldsymbol{\xi}$ is obtained as the solution of a constrained quadratic program (QP) of the form

$$\min_{\delta\boldsymbol{\xi}} J_{\boldsymbol{\xi}} = \frac{1}{2}\delta\boldsymbol{\xi}^T\mathbf{Q}\delta\boldsymbol{\xi} \quad (16a)$$

$$\text{s.t. } \mathbf{A}\delta\boldsymbol{\xi} = \mathbf{b} \quad (16b)$$

$$\underline{\delta\boldsymbol{\xi}} \le \delta\boldsymbol{\xi} \le \overline{\delta\boldsymbol{\xi}}, \quad (16c)$$

with the positive definite weighting matrix

$$\mathbf{Q} = \text{diag}(Q_{t_F}, \mathbf{Q}_{\mathbf{z}_0}, ..., \mathbf{Q}_{\mathbf{z}_N}, \mathbf{Q}_{\mathbf{u}_0}, ..., \mathbf{Q}_{\mathbf{u}_N}). \quad (17)$$

The linearized system dynamics (14) together with (15) yields the equality constraints (16b) and the inequality constraints (16c) correspond to (3d), (3e) where $\underline{\delta\boldsymbol{\xi}}^T = [\underline{0}, \underline{\delta\mathbf{z}}^T, \underline{\delta\mathbf{u}}^T]$, $\overline{\delta\boldsymbol{\xi}}^T = [\overline{\delta t}_F, \overline{\delta\mathbf{z}}^T, \overline{\delta\mathbf{u}}^T]$, with $\underline{\delta\mathbf{z}} = \underline{\mathbf{z}} - \mathbf{z}^*$, $\overline{\delta\mathbf{z}} = \overline{\mathbf{z}} - \mathbf{z}^*$, $\underline{\delta\mathbf{u}} = \underline{\mathbf{u}} - \mathbf{u}^*$, $\overline{\delta\mathbf{u}} = \overline{\mathbf{u}} - \mathbf{u}^*$, and $\overline{\delta t_F}$ a sufficiently large upper bound. However, it makes sense to force the deviations $\delta\mathbf{q}_k$ of the $\delta\mathbf{z}_k^T = [\delta\mathbf{q}_k, \delta\dot{\mathbf{q}}_k]$ to be small if the corresponding collocation points $\mathbf{q}_k^*$ are already close to one of the obstacles. Therefore, the submatrix $\mathbf{Q}_{\mathbf{q}_k}$ of the weighting matrix $\mathbf{Q}_{\mathbf{z}_k} = \text{diag}(\mathbf{Q}_{\mathbf{q}_k}, \mathbf{Q}_{\dot{\mathbf{q}}_k})$ is increased if the distance of $\mathbf{q}_k^*$ to any obstacle is small. This is achieved by utilizing the Hessian of the potential function $\varphi_{i,k}$, $i = 1, ..., m$ in the form

$$\mathbf{Q}_{\mathbf{q}_k} = \mathbf{Q}_{\mathbf{q}} + \lambda\frac{\partial^2\Big(\sum_{i=1}^m \varphi_{i,k}\Big)}{\partial\mathbf{q}_k^2}\Big|_{\mathbf{q}_k^*}, \quad (18)$$

with the constant matrix $\mathbf{Q}_{\mathbf{q}} > 0$ and the tuning parameter $\lambda > 0$. Note that the concept for fast online re-planning of trajectories proposed in this paper has some relation to sensitivity-based methods, see, e.g., Büskens and Maurer (2001).

To further enhance the calculating speed for the online trajectory planner, the constrained QP of (16) is not solved for all N grid points at once but for s sub-horizons consisting of N/s grid points. Thus, for each subhorizon the equality constraint (16b) has to be adjusted accordingly so that the end point of the preceding subhorizon corresponds to the starting point of the next subhorizon.

5. SIMULATION RESULT

The simulation was obtained using MATLAB(R2018b) with 1.8 GHz Intel Core i7 and 16GB of RAM. The Interior Point OPTimize (IPOPT) open source package, see, e.g.,Wächter and Biegler (2006), was used to solve the nonlinear optimization problem (3). In IPOPT, the Multifrontal Massively Parallel sparse direct Solver (MUMPS 5.2) is used as the linear solver to further enhance the computing speed. The analytic gradient of the cost function is calculated with CasADi, see, e.g., Andersson et al. (2019). Moreover, the numerical Hessian is computed by the BFGS approximation method, see, e.g., Liu and Nocedal (1989). The trajectory is discretized into 51 grid points yielding 664 optimization variables. The sparsity matrix structure is exploited to reduce the memory usage. To solve the constrained QP (16) with box constraints, the CVXgen package, see, e.g., Mattingley and Boyd (2012), is employed to generate a fast C-code. For simplification, we consider that the obstacles are on the ground plane and aligned with the x-, y-, and z-axis of the frame. The scenario consists of three obstacles with $\mathbf{T}_1 = [0.735, 0.305, 0]^T$, $\mathbf{T}_2 = [0.65, 0.89, 0]^T$, and $\mathbf{T}_3 = [1.51, 0.5, 0]^T$ as illustrated in Fig. 5. The state and the input of the dynamic system are $\mathbf{z}^T = [\mathbf{q}^T, \dot{\mathbf{q}}^T]$, $\mathbf{q} = [s_x, s_y, s_z, \alpha, \beta]^T$, and $\mathbf{u} = [u_1, u_2, u_3]^T$ according to (2).

Fig. 5 depicts three representative collision-free paths of the offline-planning from a starting state $\mathbf{z}_S$ (circle symbol) to a target state $\mathbf{z}_T$ (cross symbol). The computing

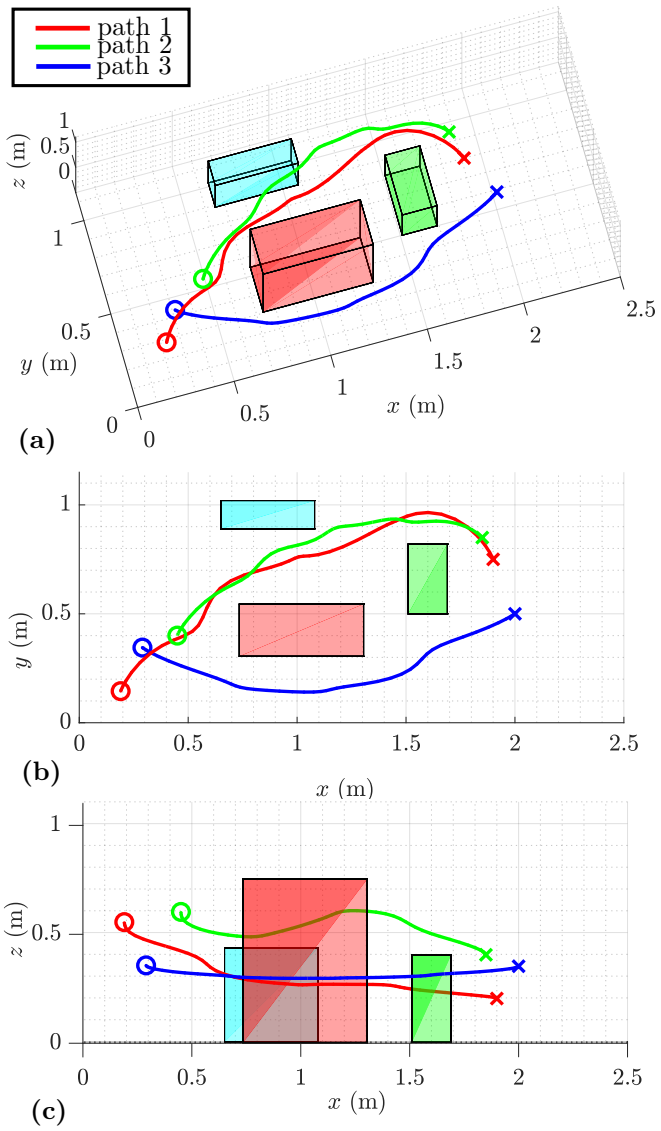

Fig. 5. Collision-free paths for three different scenarios from a starting state $\mathbf{z}_S$ (circles) to a target state $\mathbf{z}_T$ (crosses) resulting as a solution of the offline optimization (3). (a) Paths in the 3D space. (b) Paths in the xy-plane. (c) Paths in the xz-plane.

time are 0.17 s, 0.17 s, and 0.21 s for the red, green, and blue trajectory, respectively. The time evolution of the corresponding states $\dot{s}_x, \dot{s}_y, \dot{s}_z, \alpha$, and β as well as the three control inputs u_1, u_2, and u_3 are shown in Fig. 6. Moreover, the state and input constraints according to (3d) and (3e) are illustrated as black dashed lines. Note that a small violation of the constraints is possible for points of the trajectory different from collocation points.

The starting and target points of interest lie in the subspaces $\mathcal{X}_S$ (size 0.4 m × 0.8 m × 0.55 m) and $\mathcal{X}_T$ (size 0.3 m × 0.8 m × 0.4 m), illustrated as grey boxes in Fig. 7. For both subspaces, an equally spaced grid is defined. Based on the offline trajectory planning algorithm, a database of collision-free trajectories is determined which connect each grid point of $\mathcal{X}_S$ to each grid point in $\mathcal{X}_T$. For this purpose, different number of grid points N_{ST} are investigated according to Table 1. Then a Monte Carlo simulation is

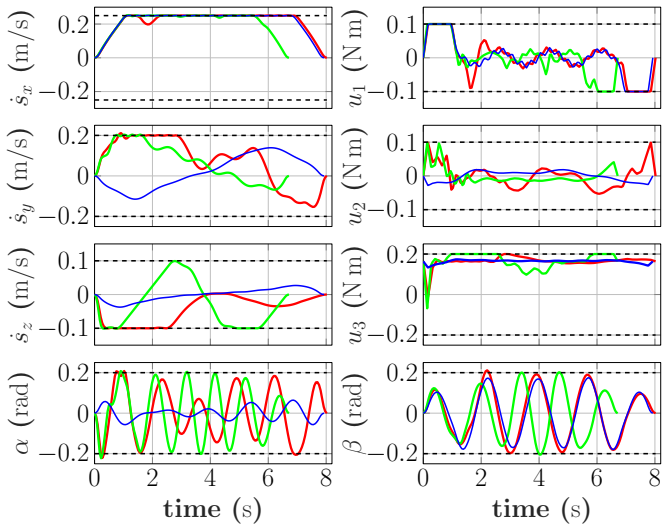

Fig. 6. Time evolution of the states and control inputs for the three different scenarios depicted in Fig. 5. The black dashed lines illustrate the constraints on the states and the control inputs.

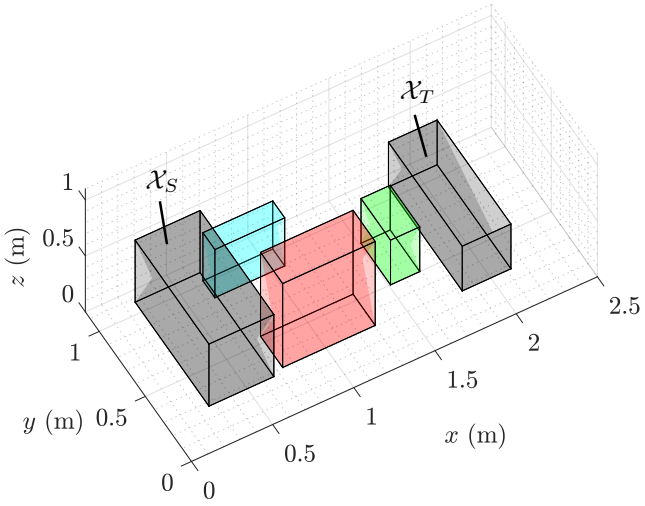

Fig. 7. Illustration of the starting and the target subspaces, $\mathcal{X}_S$ and $\mathcal{X}_T$, in grey color.

performed by randomly selecting 10^5 uniformly distributed random pairs of starting and target states $\bar{\mathbf{z}}_S$ and $\bar{\mathbf{z}}_T$ from the two subspaces and planning a new trajectory by the constrained QP (16). The statistics is shown in Table 1. While the average computing time to calculate one trajectory of the database takes approximately 0.17 s, the computing time for the constrained QP is about 7 ms. Clearly, if the grid for setting up the database becomes coarser, the distance of an arbitrarily picked starting and target state to the nearest grid points also increases on average. Therefore, the computing time also slightly grows with a smaller number of grid points N_{ST} in $\mathcal{X}_S$ and $\mathcal{X}_T$. Since the obstacles for the online planning are only taken into account in an approximate way with the Hessian of the potential functions, not all trajectories are collision free. This is reflected in the success rate in Table 1 which, as a matter of fact, decreases when the number of grid points is reduced.

Fig. 8 depicts two representative cases of an online trajectory and the corresponding nearest trajectory in the database. It can be clearly seen that if a sufficiently fine grid is chosen $N_{ST} = 100$ $(5 \times 5 \times 4)$ for each subspace $\mathcal{X}_S$ and $\mathcal{X}_T$, there are only small deviations required to realize the new trajectory. In contrast, for the coarser grid $N_{ST} = 27$ $(3 \times 3 \times 3)$, largers deviations are necessary. It can also be inferred from Fig. 8 (a) that the deviation of the online trajectory from the trajectory in the database becomes smaller for collocation points that are already closer to obstacles, which is a consequence of the chosen weighting matrices (18). Fig. 9 shows the corresponding time evolutions of the states and control inputs. In summary, the online planner is quite fast (approximately 7 ms on average) and exhibits a high success rate (greater than 98, 7%) even for coarse grids in the offline database.

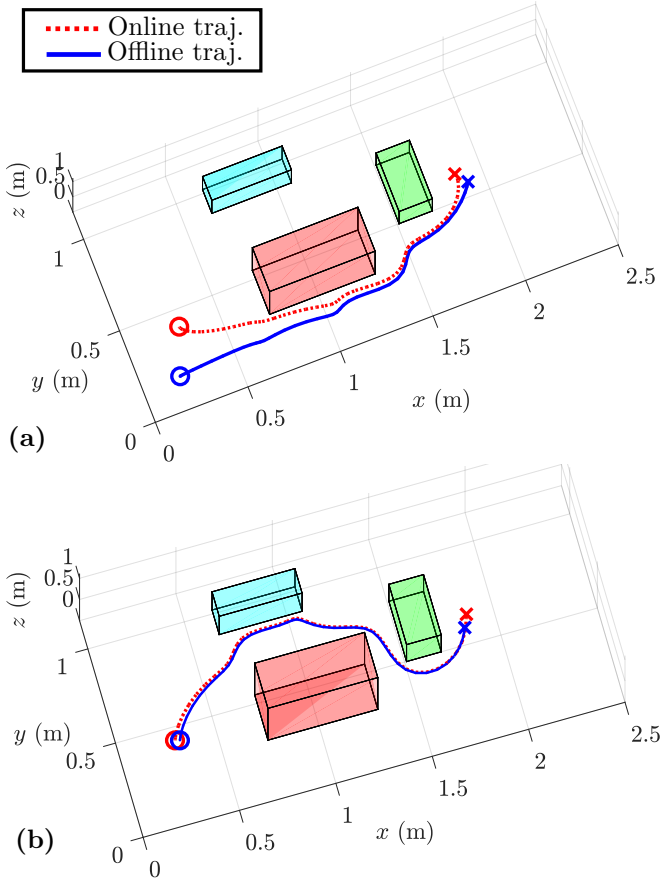

Fig. 8. Collision-free paths for two different scenarios resulting from the online trajectory planning from a starting state $\bar{\mathbf{z}}_S$ (circles) to a target state $\bar{\mathbf{z}}_T$ (crosses). The offline and online trajectories are illustrated in red and blue, respectively. (a) Online trajectory planning with a coarse grid ($N_{ST} = 27$). (b) Online trajectory planning with a fine grid ($N_{ST} = 100$).

6. CONCLUSIONS

In this work, a fast combined offline and online motion planning strategy for a 3D laboratory gantry is proposed. The offline planner is used to set up a database of time-optimal trajectories which connect the grid points of a

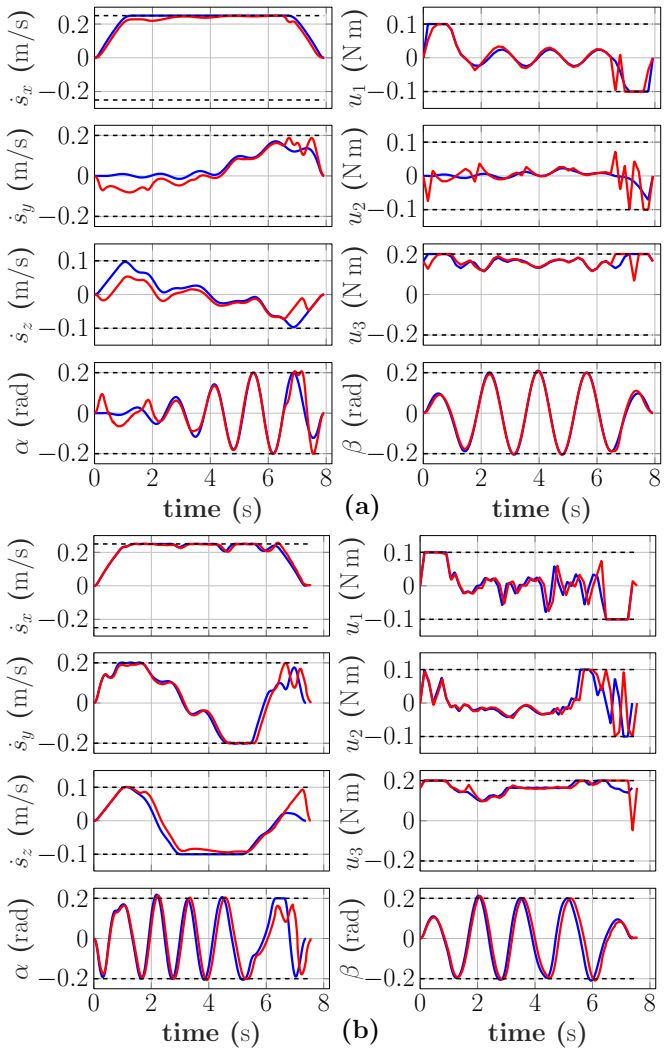

Fig. 9. Time evolution of the states and control inputs for the two different scenarios depicted in Fig. 8. The black dashed lines illustrate the constraints on the states and the control inputs. The online trajectory and the offline trajectory are illustrated as red and blue lines, respectively. (a) Online trajectory planning with a coarse grid ($N_{ST} = 27$). (b) Online trajectory planning with a fine grid ($N_{ST} = 100$).

Table 1. Monte Carlo simulations with 10^5 test cases for different numbers N_{ST} of grid points in the starting and target subspaces.

No. of grid points N_{ST}	27	80	100
Average computing time in s (offline)	0.17	0.17	0.171
Average computing time in ms (online)	7.2	7.1	6.9
Success rate (%) (online)	98.67	98.99	99.1

starting subspace with those of a target space in a collision-free way and with respect to the dynamic constraints of the 3D gantry crane. The obstacles are taken into account in form of potential functions which are based on the Log-SumExp function. The associated nonlinear optimization problem is solved by the open source package IPOPT in

combination with the parallel solver MUMPS. With this, it is possible to calculate the optimal trajectories within a computing time of 0.17 s on average. The online planner has the task to determine a new trajectory if the starting and/or target point do not conform to a grid point in the database. In a first step, the closest trajectory in the database is selected and then, in a second step, the deviation from this trajectory is minimized by a constrained quadratic optimization. It is demonstrated that this can be achieved within approximately 7 ms, even if the grid in the starting and target subspace is quite coarse. This is also the main contribution of this paper. The high computing speed of the online planner allows a fast (re)planning of the trajectories and thus it is also suitable for moving targets, which is the current research.

REFERENCES

An, N.T., Giles, D., Nam, N.M., and 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., and Diehl, M. (2019). Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

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.

Büskens, C. and Maurer, H. (2001). Sensitivity analysis and real-time optimization of parametric nonlinear programming problems. In *Online Optimization of Large Scale Systems.* Grötschel, M., Krumke, S.O., and Rambau, J. (Eds.), Berlin-Heidelberg: Springer, 3-16.

Cherif, M. (1999). Kinodynamic motion planning for all-terrain wheeled vehicles. In *Proc. of the IEEE Conf. on Robotics and Automation*, 317–322.

Davies, S. (1996). Multidimensional triangulation and interpolation for reinforcement learning. In *Proc. of the 9th Conf. on Neural Information Processing Systems*, 1005–1011.

Ferguson, D. and Stentz, A. (2005). The delayed D* algorithm for efficient path replanning. In *Proc. of the IEEE Conf. on Robotics and Automation*, 2045–2050.

Gammell, J.D., Srinivasa, S.S., and Barfoot, T.D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proc. of the IEEE Conf. on Intelligent Robots and Systems*, 2997–3004.

Gilbert, E.G., Johnson, D.W., and Keerthi, S.S. (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2), 193–203.

Hart, P.E., Nilsson, N.J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Ijspeert, A.J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. of the IEEE Conf. on Robotics and Automation*, 1398–1403.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894.

Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the RRT. In *Proc. of the IEEE Conf. on Robotics and Automation*, 1478–1483.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580.

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. and Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5), 943–957.

Kondo, K. (1991). Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. *IEEE Transactions on Robotics and Automation*, (3), 267–277.

Liu, D.C. and 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., and Kugi, A. (2018). Flatness-based nonlinear control of a three-dimensional gantry crane. *IFAC-PapersOnLine*, 51(22), 331–336.

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

Maurer, C.R., Qi, R., and Raghavan, V. (2003). A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2), 265–270.

Moore, D.W. (1992). Simplical mesh generation with applications. Technical report, Cornell University.

Pekarovskiy, A., Nierhoff, T., Hirche, S., and Buss, M. (2017). Dynamically consistent online adaptation of fast motions for robotic manipulators. *IEEE Transactions on Robotics*, 34(1), 166–182.

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

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

Wächter, A. and 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.

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