Nonlinear model predictive control of an overhead laboratory-scale gantry crane with obstacle avoidance

Saad Iftikhar, Omar J. Faqir, Eric C. Kerrigan

Abstract—Gantry cranes are complex nonlinear electromechanical systems representing a challenging control problem. We propose an optimization-based controller for guiding the crane through arbitrary obstacles. Solving path planning problems with obstacles typically requires a two-stage approach. First, a path is found that is feasible w.r.t. system dynamics and obstacles. The path is then interpreted as a series of set points by a lower-level controller that guides the system. We instead generate a path, and the associated control input to move along that path, from a single optimization problem using a nonlinear model predictive control framework. In doing so, we generate a trajectory that is locally optimal and feasible w.r.t. system dynamics and obstacles. Multiple obstacle avoidance constraint formulations are proposed as smooth, differentiable functions. Objects are approximated either as the union of a set of smooth shapes or as smooth indicator functions. The formulations presented in this work are applicable to (non-)convex problems in 2-D or 3-D spaces. Numerical methods are used to solve the proposed problems for both 2-D (fixed string length) and 3-D (varying string length) models of the gantry crane, resulting in consistently lower costs than nodal or sampling based algorithms.

I. INTRODUCTION

Autonomous navigation through obstructed environments is of high importance, with applications found in multiple fields, including autonomous vehicles, industrial warehouse robots, UAVs and search and rescue robots. Determining precise and collision-free trajectories in real-time is both theoretically and practically challenging. Early works in collision free navigation include [1], [2], where edge detection and dynamic window approaches were used. Recent works (e.g. [3], [4]) tackle this problem using potential field algorithms, random sampling or dynamic programming. Comprehensive surveys of motion planning with collision avoidance, including examples of the aforementioned applications, can be found in [5]–[7]. A taxonomy of algorithms is shown in Figure 1.

One approach is to divide the task into two separate layers. At the higher level, a path planner computes a feasible path offline. Online, this path is typically parameterized as a time series of set-points, which are tracked by a low level regulating controller e.g. LQR [8], [9].

The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1) is gratefully acknowledged.

Eric C. Kerrigan is with the Department of Electrical and Electronic Engineering and Department of Aeronautics, Imperial College London, London SW7 2AZ, U.K. e.kerrigan@imperial.ac.uk

Saad Iftikhar and O. J. Faqir are with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ, U.K. si34178@ic.ac.uk ojf128@ic.ac.uk

Path Planning Algorithms

<table>
<thead>
<tr>
<th>Sampling</th>
<th>Nodal</th>
<th>Model-based</th>
<th>Bio-inspired</th>
<th>Multi-fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRT</td>
<td>A*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1: Path planning algorithm taxonomy. Italics indicate algorithms considered for comparison in this work.

One possibility for offline path planning is to discretize the space and use node-based planning methods. A*, for example, is an optimal node-based algorithm in the sense that it returns the shortest distance between two points in the presence of an obstacle [10]. This algorithm, being an extension of Dijkstra’s algorithm, relies on the concept of dynamic programming. However, A* does not take into account system dynamics, and because of this there is no necessary equivalence between shortest distance and actual metrics of interest (e.g. minimum-time or energy). Rapidly exploring random trees (RRT), another node-based algorithm, uses random sampling to expand a tree in the configuration space until a feasible trajectory is found from start to goal node. This algorithm theoretically guarantees probabilistic completeness, but with no guarantee of optimality. The work [11] presents the extended algorithm RRT*, which is asymptotically optimal. RRT* requires a domain-specific heuristic to enable exploration of the state space and can be efficiently used in control applications if system dynamics are enforced through this heuristic. Another approach to offline path generation is to formulate the task of computing collision-free trajectories as an optimal control problem (OCP) [12]–[14]. Solving this OCP may be more computationally expensive, but explicitly accounts for system dynamics in the solution method. Furthermore, in these methods, obstacle avoidance conditions are imposed directly as constraints on the system state.

In this paper we perform online computation of an optimal collision-free trajectory using model predictive control (MPC), and implement the computed control action at each sampling instant [13]–[15]. Efficient formulation of obstacle avoidance constraints in the OCP is a key difficulty, as these constraints are inherently non-convex and may even be non-smooth [14]. Several ways of formulating these constraints have been proposed and are discussed in Section II-A.

MPC is an extensively used optimization-based control strategy [16], [17]. A control sequence is computed at each sample time as the solution of a constrained optimization
problem, wherein a system model is used to predict state evolution. To account for uncertainty, only the first input is applied to the system before the procedure is repeated at the next sampling instant. Advantageously, MPC explicitly incorporates constraints and system dynamics in the problem formulation. In nonlinear MPC (NMPC) [18] a nonlinear OCP is solved online. Nonlinearities typically arise in the system dynamics, but may also occur in constraints e.g. obstacle avoidance constraints. While linear MPC can be solved efficiently online for even moderate/large sized problems, the NMPC problem is in general non-convex and numerical optimization methods may struggle [19]. As such it is imperative to consider proper problem formulation, transcription methods, and NLP solvers for such a problem, the choice of each being application/problem dependent.

In this work we use NMPC to generate collision-free trajectories for a laboratory-scale gantry crane with five degrees of freedom (DOF) and three control inputs. This nonlinear electro-mechanical system poses an interesting control problem, particularly due to the under-actuated nature of the system and associated high load swings. We refer the interested reader to [20] for a recent overview of existing crane control methodologies, including MPC. In particular, we consider a minimum-time formulation, where the objective is to minimize the time it takes to move the payload from one position to another in space. We present two different ways of formulating the problematic non-convex (and non-differentiable) obstacle constraints for our system; approximating obstacles using either polynomials or hyperbolic indicator functions.

The NMPC problem is formulated in continuous-time using the Imperial College London Optimal Control Software (ICLOCS2) [21], a MATLAB based optimal control toolbox. ICLOCS2 allows users to define and solve an OCP with general path and boundary constraints. ICLOCS2 performs transcription of the OCP to a nonlinear program (NLP), then solved using IPOPT [22]. We solve the crane trajectory planning problem and compare the results for various obstacle formulations, as well as with nodal planners A* and RRT*.

The paper is organized as follows. In Section II we describe potential obstacle constraint formulations, and determine the resulting decreasing horizon NMPC problem. In Section III we detail the NMPC and ICLOCS2 simulation parameters, following with a discussion of numerical results. Section IV presents comparison of the results obtained by using our proposed methodology with two of the most commonly used path planner i.e. A* and RRT*.

II. Obstacle Avoidance NMPC

NMPC allows us to generate a feasible stabilizing control law, while optimizing a particular system metric. In this work we formulate a minimum-time control problem, with obstacle avoidance imperatives imposed as path constraints.

A. Constraint Formulation

Apart from nonlinearities in the system dynamics, a key source of complexity is the inclusion of obstacle avoidance constraints. These are in general non-convex and may even be non-smooth. Care must be taken when formulating these constraints and several variations are available in literature. A separating hyperplane is introduced in [23], [24]. This is only applicable to a convex motion system coupled with convex obstacles. A mixed-integer problem was formulated for polyhedral obstacles in [25], [26]. Strong duality of the distance metric is used in [14] to compose smooth nonlinear avoidance constraints. In this work, similar to [15], we consider an obstacle $O$ in $\mathbb{R}^n$ space to be the intersection of an arbitrary number of smooth functions $h_i(\cdot)$,

$$O \triangleq \{ \rho \in \mathbb{R}^n : h_i(\rho) > 0, i = 1, \ldots, k \}. \quad (1)$$

We then approximate $O$ using either ellipsoids or hyperbolic indicator functions, and compare performance with the formulation in [14], as well as nodal planning algorithms.

1) Union of Intersecting Ellipsoids: Consider approximating obstacles as a union of smooth, differentiable shapes. For example, an irregular polygon can be broken down as a combination of rectangles and triangles. Each of these lower order polygons may then be approximated with a smooth $C^2$ function. Using higher-degree polynomials result in better approximation of these shapes (the Hausdorff distance is used as a metric to measure similarity). However this also results in higher numerical error during the discretization process and may result in ill-conditioning within the NLP. Ellipsoids appear a good trade-off between desirable accuracy and beneficial numerical properties. Even so, the conditioning of the NLP still depends on the ellipsoids’ eccentricity. Through numerical experiments we found a 60% increase in solution time as the eccentricity of an ellipse (specifically placed not to effect the optimal trajectory) is varied in the range $[0, 1]$.

If $\rho \in \mathbb{R}^2$ is the position of our payload in the the $xy$-plane, we can approximate a single obstacle as

$$O \approx O_E \triangleq \{ \rho \in \mathbb{R}^2 : \bigcup_{i=1}^L S_i(\rho) \leq 0, \ i = 1, \ldots, L \}, \quad (2)$$

where $S_i(\cdot)$ denotes the ellipsoid defined by

$$S_i(\rho) = \frac{(\rho_{x} - O_{x,c}) \cos \theta_i - (\rho_{y} - O_{y,c}) \sin \theta_i}{a_i^2} + \frac{(\rho_{x} - O_{x,c}) \sin \theta_i + (\rho_{y} - O_{y,c}) \cos \theta_i}{b_i^2} - 1. \quad (3)$$

Here $O_{x,c}$ and $O_{y,c}$ represent the center of the obstacle w.r.t. the $x$ and $y$ axis, $a$ and $b$ are the major and the minor axis and $\theta$ is the angle of our ellipsoid’s major axis w.r.t. the $x$ axis. $L$ is the number of ellipsoids used to estimate each obstacle. The resulting obstacle avoidance constraint is $\rho \notin O_E$. For simplicity of exposition we have only presented this formulation for the 2-D case. Higher dimensional ellipsoids may be synonymously used to approximate 3-D obstacles.

2) Sum of smooth nonlinear indicator functions: Considering the obstacle definition (1), which can be described as the intersecting set of $k$ (non)linear equations. In this formulation $h_i(\cdot)$ is chosen to be a separating hyperplane.
Obstacle avoidance constraints are again written as $\rho \notin \mathcal{O}_I$. For the system to avoid obstacles, at least one of the inequalities bounding the obstacle shape in (4) must be violated. This condition is implemented in [15] as a product constraint on $\max(0, h_i(\tau))$, which is then superimposed on the objective function as a penalty term. In addition to being discontinuous, these added penalties may make tuning difficult, ultimately effecting the minimum-time objective.

To avoid this we propose writing the obstacle avoidance constraint as a sum of smooth indicator functions,

$$\begin{equation}
H(\rho) \triangleq \sum_{i=1}^{m} (c + d \tanh(\epsilon(h_i(\rho) + f))) < \epsilon, \tag{3}
\end{equation}$$

where parameters $c, d, \epsilon \in \mathbb{R}$ and tolerance $\epsilon \in \mathbb{R}^+$. The hyperbolic tangent function acts as an indicator to differentiate between points inside and outside of the obstacle. We use the hyperbolic tangent as a smooth approximation of the step indicator function to implement the obstacle avoidance constraint. Large negative values are assigned to coordinates outside the obstacle, and small positive values to points intersecting the obstacle, resulting in the approximation,

$$\mathcal{O} \approx \mathcal{O}_I \triangleq \{ \rho \in \mathbb{R}^2 : H(\rho) < \epsilon \}. \tag{4}$$

The quality of the approximation/OCPr solution and efficiency of the NLP solver will depend on choice of parameters. Values of $d$ and $\epsilon$ adjusts the slope of the indicator function, while $c$ and $f$ are respectively the vertical and horizontal offset. Higher values for the slope lead to a closer approximation of a step function, but introduces a discontinuity at the origin which ma cause some gradient based solvers to struggle. Conversely, lower slope parameters lead to over-approximation of the obstacles and may result in more conservative trajectories. The tolerance $\epsilon$ determines allowable constraint violations. Any point where (3) is satisfied is said to be outside the obstacle boundaries (for the specified tolerance). Setting $\epsilon = 0$ leads to over-approximation of the shape.

B. System Dynamics

The dynamical system under study is an overhead gantry crane. We consider both the 2-D system, where payload length is fixed, and the 3-D system, where the payload may be raised/lowered. The system is depicted in Figure 2. We only present equations of motion for the 3-D case, and refer interested readers to [27] for full derivations:

$$\begin{align}
(M + M_R + m)\ddot{x} &+ x \cos \theta_y \dot{\theta}_x - ml \sin \theta_x \sin \theta_y \ddot{y} + m \sin \theta_x \cos \theta_y \ddot{y} \nonumber \\
+ m \sin \theta_x \cos \theta_y \ddot{y} &+ 2m \cos \theta_x \cos \theta_y \ddot{y} - ml \sin \theta_x \sin \theta_y \ddot{y} + ml \cos \theta_x \cos \theta_y \ddot{y} - 2ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y - ml \sin \theta_x \dot{\theta}_y \nonumber \\
- 2ml \cos \theta_x \dot{\theta}_y &- ml \sin \theta_x \cos \theta_y \dot{\theta}_y - ml \sin \theta_x \cos \theta_y \dot{\theta}_y - 2ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y = f_x, \tag{5a}
\end{align}$$

$$\begin{align}
(M + m)\ddot{y} &+ ml \cos \theta_y \ddot{x} + m \sin \theta_y \dot{\theta}_y \ddot{y} + m \sin \theta_y \dot{\theta}_y \ddot{y} + T_y \ddot{y} + 2m \cos \theta_y \dot{\theta}_y \ddot{y} = f_y, \tag{5b}
\end{align}$$

$$\begin{align}
ml^2 \cos \theta_y \ddot{\theta}_y + ml \cos \theta_x \ddot{x} + ml \sin \theta_x \dot{\theta}_x \ddot{y} + ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y - 2ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y - 2ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y &+ 2ml \cos \theta_x \dot{\theta}_x \dot{\theta}_y = 0, \tag{5c}
\end{align}$$

Fig. 2: 3-D Gantry Crane model [28].

$$\begin{align}
ml^2 \ddot{\theta}_y + ml \cos \theta_y \ddot{y} &- ml \sin \theta_x \ddot{x} + 2ml \dot{\theta}_y + ml \cos \theta_y \sin \theta_x \ddot{y} + mgl \sin \theta_x \dot{\theta}_y = 0, \tag{5d}
\end{align}$$

$$\begin{align}
(M_l + m)\ddot{y} &+ m \sin \theta_x \cos \theta_x \ddot{y} + m \sin \theta_y \ddot{y} + T_l \ddot{y} - ml \cos \theta_x \ddot{y}^2 - ml \ddot{\theta}_y^2 - mgl \cos \theta_x \cos \theta_y = f_l. \tag{5e}
\end{align}$$

String length is denoted as $l$, cart weight as $M_l$, and we assume the crane load to be a point mass of weight $m$. $f_1$, $f_x$ and $f_y$ are the forces in the $z$,$x$ and $y$ directions. Similarly, $M$ is the mass of cart, $M_R$ the rail mass, and $T_1, T_x, T_y$ are the friction coefficients in each dimension. These system equations are converted into state space form for use in the NMPC problem.

C. Minimum-time NMPC Formulation

We seek a control sequence to transfer the system from given initial state $\xi(t_0) = \xi_0$ to a target set of states $\bar{S}(\chi(t), t)$ in the shortest time [29]. To this end, we propose solving the following OCP in a decreasing horizon fashion,

$$\begin{align}
\min_{\xi, u_{t_f}} &\int_0^{t_f} ||u(t) - u_{ref}||^2_{R_k} \, dt \tag{6a}
\end{align}$$

s.t. $\forall t \in [0, t_f], \quad \dot{\xi}(t) = f(\xi(t), u(t)) \tag{6b}$

$$\begin{align}
g(\xi(t), u(t), t) &\geq 0 \tag{6c}
\end{align}$$

$$\begin{align}
g(\xi(t_0), \xi(t_f)) &\geq 0 \tag{6d}
\end{align}$$

$$\begin{align}
u(t) &\in U, \quad \xi(t) \in X. \tag{6e}
\end{align}$$

The final trajectory time $t_f$ is our main objective. We include the integration of a small cost $||u(t) - u_{ref}||^2_{R_k} := (u(t) - u_{ref})^T R_k (u(t) - u_{ref})$ on the input as a quadratic regularization term to improve convergence properties of the NLP solver. This regularization term results in $400 - 500\%$ decrease in computation time. The scaling matrix $R_k$ is kept as small as possible without affecting the solution’s convergence properties, so that our original objective i.e. the final time is not dominated by the path cost. The system state is denoted with $\xi(t) \in \mathbb{R}^{10}$,

$$\begin{align}
\xi := [x, \dot{x}, y, \dot{y}, \theta_x, \dot{\theta}_x, \theta_y, \dot{\theta}_y, l, \dot{l}]^T, \tag{7}
\end{align}$$
and dynamics with $\dot{\xi}(t) = f(\xi(t), u(t))$, where $u(t) = [f_x(t), f_y(t), f_z(t)]$. Constraints (6c) and (6d) are, respectively, smooth path and boundary constraints. States and inputs are subject to box constraints $\mathcal{X}, \mathcal{U}$.

In many we may wish to enforce additional state constraints on the crane states. Swing suppression is often of particular importance for the purpose of stability, safety, integrity of the physical system and – in the particular case of model based control – maintaining fidelity of models based on small angle approximations. Swing suppression can be implemented, for example, as input constraints [30] or as an additional cost in the objective [31]. Although additional constraints may well be added to (6), swing suppression leads to a drastic decrease in final time, and so are not included in simulation results.

III. SIMULATIONS

Closed-loop simulations using ICLOCS2 were performed in MATLAB 2017b. All simulations were performed on an Intel(R) Core(TM) i7-6700 CPU 3.4GHz processor with 16GB of RAM memory. System dynamics from (5a)–(5d) are transformed to state space form, and implemented in the NMPC problem (6). Obstacle avoidance constraints are implemented as path constraints (6c). See [32] for details of the INTECO 3-D crane. The continuous-time problem is transcribed using Hermite Simpson collocation on a fixed grid. In doing so, we take advantage of the numerical derivative and automatic scaling features of [21]. Specifically, we use a limited memory BFGS Hessian update which is free of matrix operations and requires limited inner products. For closed-loop simulations, problem (6) is implemented in a decreasing horizon fashion. At each sample period the system is measured and the prediction horizon is reduced by a single fixed step.

A. Model and Path formulation

Simulations were performed for both 2-D and 3-D gantry crane systems. Dynamics for the 2-D gantry crane model can be taken from [33]. The 2-D model is a 4-DOF under-actuated system with two control inputs, in contrast to the 3-D crane model discussed earlier which is a 5-DOF system, with three actuator inputs. Obstacles are formulated using the formulations presented in (2) and (4).

B. Initial Guess

Problem (6) is non-convex and may be computationally difficult to solve. Most nonlinear numerical optimization packages, including IPOPT, generate local solutions. The initial guess for the NLP plays a crucial role in determining convergence rate and solution quality. Having said this, determining a suitable initial guess a priori is non-trivial. One possibility is to use a solution from path planning algorithms (e.g. A*) which may not account for system dynamics. The quality of the path planning based initial guesses depends heavily on the application (see [14] for examples). Furthermore, these algorithms do not give the associated control inputs.

TABLE I: Comparison of NMPC solution with obstacle avoidance formulations (2), (4) with 2-D and 3-D crane models.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Formulation</th>
<th>$t_f$ (s)</th>
<th>Cost (6a)</th>
<th>Comp Time (s)</th>
<th>NLP Iters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 3a</td>
<td>(2)-2D</td>
<td>1.6495</td>
<td>1.8470</td>
<td>1.4</td>
<td>55</td>
</tr>
<tr>
<td>Fig. 3d</td>
<td>(2)-2D</td>
<td>1.6417</td>
<td>1.8448</td>
<td>1.2715</td>
<td>41</td>
</tr>
<tr>
<td>Fig. 3b</td>
<td>(4)-2D</td>
<td>1.6629</td>
<td>1.8635</td>
<td>0.9890</td>
<td>31</td>
</tr>
<tr>
<td>Fig. 3e</td>
<td>(4)-2D</td>
<td>1.6408</td>
<td>1.8327</td>
<td>1.314</td>
<td>44</td>
</tr>
<tr>
<td>Fig. 3c</td>
<td>(4)-3D</td>
<td>1.5891</td>
<td>2.0651</td>
<td>2.0716</td>
<td>56</td>
</tr>
<tr>
<td>Fig. 3f</td>
<td>(4)-3D</td>
<td>1.4393</td>
<td>1.7134</td>
<td>1.99</td>
<td>44</td>
</tr>
</tbody>
</table>

Instead we first solve the unconstrained problem (subject to system dynamic but no obstacle avoidance constraints). The problem is then resolved adding one obstacle at a time, using the previous solution as an initial guess. One may try all possible combination of introducing obstacles one by one in the path to reach the best possible solution. Generally, it took us an average of 6.3 s to complete this whole procedure when four obstacles were present in our path, as in Figures 3a–3c.

C. Simulation Analysis

Robustness of the proposed formulations has been tested for multiple obstacle configurations. For this system, the NMPC formulation using either (2) or (4) does converge to a locally optimal solution, assuming a random but feasible initial guess. In absence of a feasible initial guess, (2)-constrained NMPC converged 30% of the time, while (4)-constrained NMPC converged less than 10% of the times tested. Convergence was further seen to be highly dependent on the complexity of the environment. Figure 3 shows generated trajectories using both formulations, with the associated computation time and metrics displayed in Table I.

As expected, the 3-D controller is never slower than the case when the length of the pendulum is fixed. The final time $t_f$ is shortened by reducing/lengthening the pendulum.
— thereby changing the system stiffness — to control the load swing. The trajectories in Figure 3 appear similar in the $xy$-plane for the second configuration, but markedly different for the first configuration, when considering the 2-D and 3-D case. The additional degree of freedom allows the load to pass through smaller gaps between obstacles. The increased performance in switching from 2-D to 3-D is associated with an almost doubling in computation time.

Considering just 2-D operation, the trajectories and $t_f$ appear similar for formulations (2) and (4). However, there is a 50% increase in the associated computation time when the ellipsoid approximation of obstacles is used in place of the indicator formulation. This has been experimentally verified as being due to the number of ellipses required to properly approximate the simulated environments (in this case three ellipses). In obstacle avoidance constraint formulation (3), associated with obstacle formulation (4), the value of allowable tolerance parameter $\epsilon$ is to be set between 0–0.0175. Any $\epsilon$ above this limit leads to an under-approximation of the path obstacles in our tested configurations. Figure 4, and the associated Table II, compares the two proposed approximate formulations with the exact formulation detailed in [14]. The method proposed by [14] is marginally faster (0.9–0.95% quicker) than formulation (2) and (4). However, in simulation we found computing the solution of the exact formulation was 5 times slower.

### IV. Comparison with Other Motion Planners

As discussed in Section I, collision free paths may be generated using node search or sampling-based algorithms. We compare our NMPC approach to A* and RRT*. A* does not take into account system dynamics, while our implementation of RRT* uses an exploration heuristic based on the linearized crane dynamics. These path planners generate a time ordered sequence of set points, which are used for references for a lower-level regulation-based NMPC controller which steers the crane. Importantly, minimum-time trajectories accounting for full nonlinear dynamics can be remarkably different to trajectories generated by planning algorithms which attempt to minimize distance covered.

A comparison of the resulting trajectories is shown in Figure 5. A* finds the shortest path in terms of distance, but the close vicinity of obstacles restricts the lower level controller from adopting large swing angles. The closed-loop velocities are therefore correspondingly slow because the path generated by A* did not account for these dynamics. By comparison, the RRT*-LQR [34] algorithm is only asymptotically optimal. Due to the randomness of the generated tree, the resulting path was not time-optimal. In fact, it was found to be the slowest of the three formulations. Finally, the trajectory generated by the decreasing horizon NMPC formulation is also shown in Figure 5. Whereas the paths resulting from A* and RRT* are nominally very similar, the minimum-time path generated through NMPC is noticeably different. The crane is moved through an obstacle-free region of the maze, allowing for large swings, and a final time that was 27% faster than either of the node-based algorithms.

### V. Conclusions and Future Work

We have investigated and compared the performance of optimization-based path planning techniques for a 3D gantry crane. Non-convex, possibly non-smooth obstacles have been approximated through smooth nonlinear constraint formulations. Experimentally, the solution quality is comparable for formulations using either smooth indicator functions or ellipsoid approximations. However, the indicator function formulation seemingly scales better due to the number of ellipsoids needed to approximate complex obstacles. Fur-
thermore, the resulting trajectories consistently displayed a significant improvement in final time over node and sample based path planners. This was particularly apparent in the case where non-linearities in the crane dynamics meant that the shortest path was not the fastest, resulting in a 27% decrease in $t_f$.

A drawback of the investigated approaches is the complexity of solving these NMPC problems in real-time. Even a best case sampling time of approximately 1 s (Table I) is orders of magnitude too long for real-time implementation, which may be a subject of future work. One possibility is sub-optimal MPC [35] where the NLP is only solved to a reduced tolerance. This may be achieved by terminating the NLP algorithm after a fixed number of iterations. Alternatively, we may calculate the optimal trajectory offline and use it as an initial guess for a feasibility problem to be solved in a decreasing horizon fashion. Stability may be enforced through contraction constraints of the cost function at each sample time [35]. Alternatively, we may consider sensitivity based techniques such as the real-time iteration scheme proposed in [36] or the advanced-step MPC proposed in [37].

**References**


