Journal Title
XX(X):1–20
©The Author(s) 0000
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Global Inverse Kinematics via Mixed-Integer Convex Optimization

Hongkai Dai[1], Gregory Izatt[2] and Russ Tedrake[1, 2]

Abstract

In this paper we present a novel formulation of the inverse kinematics (IK) problem with generic constraints as a mixed-integer convex optimization program (MIP). The proposed approach can solve the IK problem globally with generic task space constraints - a major improvement over existing approaches, which either solve the problem in only a local neighborhood of the user initial guess through nonlinear non-convex optimization, or address only a limited set of kinematics constraints. Specifically, we propose a mixed-integer convex relaxation of non-convex $SO(3)$ rotation constraints, and apply this relaxation on the inverse kinematics problem. Our formulation can detect if an instance of the IK problem is *globally* infeasible, or produce an approximate solution when it is feasible. We show results on a 7-joint arm grasping objects in a cluttered environment, an 18 DoF quadruped standing on stepping stones, and a parallel Stewart platform. Moreover, we show that our approach can find a collision free path for a gripper in a cluttered environment, or certify such a path doesn't exist. We also compare our approach against the analytical approach for a 6-joint manipulator. The code is open-sourced at `drake.mit.edu`.

1 Introduction

The inverse kinematics (IK) problem is one of the most fundamental problems in robotics. It aims to find robot postures, so as to satisfy certain kinematic constraints. The kinematic constraints can be the end effector reaching a certain location with a given orientation, or more complex constraints in the task space, such as grasping a mug on the table while keeping the robot free from collision (Fig 1a).

Solving the IK problem is quite challenging, as it requires solving a set of nonlinear equations involving products of trigonometric functions (sin/cos). There have been tremendous efforts to solve the inverse kinematics problem Craig (2005); Mason (2001); Murray et al. (1994); Berenson et al. (2011); Singh et al. (2018), and they can be coarsely categorized as analytical or numerical. The analytical approach solves the kinematics equations as polynomials of sine and cosine and can produce closed-form solutions. It is widely known that many manipulators with 6 Degree of Freedoms (DoFs) allow analytical solution for the end effector to reach a specified position with a given orientation Peiper (1968); Raghavan and Roth (1990); Manocha and Canny (1992); Husty et al. (2007); Qiao et al. (2010). More generally, Diankov (2010) introduced *IKfast*, which can find inverse kinematics solutions for more complicated robots by numerically discretizing the DoFs above 6, and solving the remaining 6 DoFs in the closed form. But the set of kinematics constraints it can handle is still limited. For example, the collision avoidance constraint is ignored in the analytical IK solvers, and postures in collision are rejected only in a post-processing step.

Some researchers treated solving inverse kinematics problems as numerically computing the roots of polynomial equations. The polynomial equations can be solved by solvers like Bertini (Bates et al. (2013)), or through homotopy continuation method (Li (2003); Varedi et al. (2009)). This approach does not permit inequality constraints on the link poses, which occur frequently when planning postures for certain tasks. This shortcoming makes it hard to handle general task space constraints involving inequalities, such as “putting the hand between the two boxes on the table.” It is also challenging to impose collision avoidance constraint with this approach.

On the other hand, optimization based numerical approach can solve inverse kinematics problems for complicated robots with generic constraints. This approach formulates an inverse kinematics problem as a general non-convex nonlinear optimization problem, and calls gradient-based nonlinear solvers to handle these non-convex constraints. For example, Beeson and Ames (2015) parameterized the link poses using double quaternions, and solve the nonlinear IK problem through their software Trac-IK. Similarly other researchers solve the IK problem through nonlinear optimization techniques, such as sequential quadratic programming Dai et al. (2014); Fallon et al. (2015) and Jacobian pseudo-inverse Buss (2004). The drawback of this approach is that the solution heavily relies on the user-supplied initial seed. Because the nonlinear optimization typically solves the problem locally around the initial seed, it can get trapped in a locally infeasible region and never reach the distant feasible solutions. As a result, when these non-convex nonlinear solvers report the problem being infeasible, they only assert local infeasibility, providing no guarantee on

[1] Toyota Research Institute, USA
[2] Massachusetts Institute of Technology, USA

Corresponding author:
Hongkai Dai, Toyota Research Institute, Los Altos, CA, USA
Email: hongkai.dai@tri.global

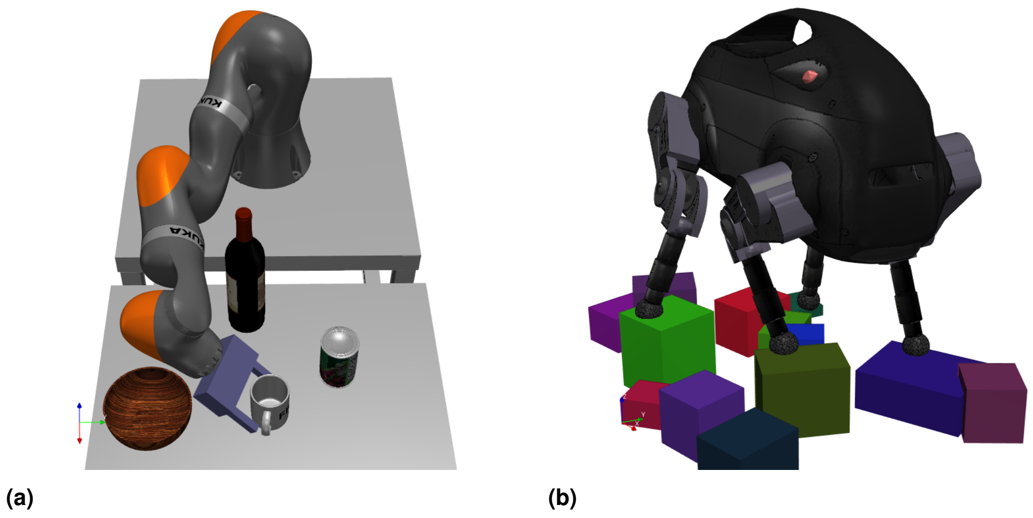


Figure 1. Our IK solver finds a collision free grasping posture for the KUKA IIWA arm (left), and a posture for Little Dog standing on stepping stones (right).

the absence of a global solution Bertsekas (1999). Typically the user would call the nonlinear optimization solver with different initial guesses. If none of the initial guess leads to a feasible solution, the user *thinks* the IK problem doesn't have a solution, but there is no guarantee on how many initial guesses should be attempted before claiming the global infeasibility. We will show that our approach can significantly improve the robustness and success rate of the IK solver, over the gradient-based nonlinear optimization approach. More importantly, our approach can also certify *global* infeasibility of the IK problem, when our approach cannot find a solution. This global certificate is in sharp contrast to the result from previous nonlinear optimization techniques, which only provide local certificates without any claim to the size of the local infeasible region.

We shall use an optimization-based numerical approach to the IK problem with generic constraints to obtain the *global* solution. Instead of tackling the problem through non-convex nonlinear optimization, we instead consider a mixed-integer convex optimization formulation. Solvers for this class of optimization do not require an initial seed, and can provide a global solution Boyd and Vandenberghe (2004); Bertsimas and Weismantel (2005), such as a certificate of global infeasibility. The non-convexity of the IK problem originates from the non-convexity of $SO(3)$ constraints on rotation. If we choose to represent rotation with a matrix $R = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3] \in \mathbb{R}^{3\times3}$, this rotation matrix should satisfy the following constraints

$$\mathbf{u}_i^{\mathrm{T}}\ \mathbf{u}_i = 1 \text{ (Unit length)} \tag{1a}$$
$$\mathbf{u}_i^{\mathrm{T}}\ \mathbf{u}_j = 0 \text{ if } i \neq j \text{ (Orthogonality)} \tag{1b}$$
$$\mathbf{u}_i \times \mathbf{u}_j = \mathbf{u}_k, \tag{1c}$$

where $(i,j,k) = (1,2,3), (2,3,1)$ or $(3,1,2)$ in (1c). We choose the rotation matrix as the orientation representation, since the position of a point attached to a body is a *linear* expression of the body rotation matrix. If we used unit quaternions or angle-axis representations, this expression would be nonlinear, and would complicate the expression of pose-dependent constraints. Using the rotation matrix representation "concentrates" the non-convexity into only constraints on R, leaving the rest of the IK formulation clean and convex. It's easy to see that orientation constraint $SO(3)$ (1a)-(1c) is non-convex. Take constraint (1a) as an example: geometrically, the nontrivial convex combination of two distinct points on the surface of a unit sphere, lies strictly in the interior of the unit sphere. Algebraically, an equality constraint is non-convex, if it includes quadratic terms such as $\mathbf{u}_i(1)^2$ in (1a), or bilinear terms like $\mathbf{u}_i(1)\mathbf{u}_j(1)$ in (1b).

Various convex relaxations for $SO(3)$ constraints have been proposed. Saunderson et al. (2015) proved that the convex hull of the set of rotation matrices can be described by a *positive semidefinite constraint* (PSD), a special type of convex constraint. In Dai et al. (2015) this convex relaxation is exploited to solve the IK problem through a sequence of convex optimizations. The drawback of this approach is that the convex hull of the rotation matrix is a rather loose relaxation of $SO(3)$. For example, the matrix $[\mathbf{u}_1, \mathbf{0}, \mathbf{0}] = 0.5\,[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] + 0.5\,[\mathbf{u}_1, -\mathbf{u}_2, -\mathbf{u}_3]$ is in the convex hull for any rotation matrix $[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]$, but it is far from satisfying the $SO(3)$ constraint. Hence quite often the relaxed convex program is too loose to detect global infeasibility of the original IK problem. To overcome this, we seek a tighter mixed-integer convex relaxation of the $SO(3)$ constraint.

Our intuition is that instead of considering the convex hull of *all* rotation matrices as in Saunderson et al. (2015), we divide the range of the rotation matrices into smaller intervals, and compute the convex hull of each small interval to obtain a tighter approximation. We can then constrain the approximated rotation matrix to be within *one of* the convex hulls. This approach is inspired by Deits and Tedrake (2014), in which the orientation constraint $SO(2)$ in 2D is approximated by mixed-integer linear constraints. In a similar fashion, we replace (1a)-(1c) with mixed-integer convex (quadratic) constraints, as a relaxation of the original non-convex $SO(3)$ constraints. This relaxation allows the inverse kinematics problem to be formulated as a mixed-integer convex optimization program, which can be solved efficiently by modern solvers, such as Gurobi Optimization (2016) and Mosek (2010). Like convex optimization, mixed-integer convex optimization also does not rely on the initial seed, and warrants global solution (Schrijver (1998); Bertsimas and Weismantel (2005)). With the rapid advancement in numerical solvers, mixed integer convex optimization has been becoming increasingly popular in robotics. For example, it has been used in footstep planning Deits and Tedrake (2014), aggressive legged locomotion planning Valenzuela (2016); Ding et al. (2018), quadrotor motion planning Mellinger et al. (2012); Deits and Tedrake (2015b); Landry et al. (2016), grasp simulation Pang and Tedrake (2018), pose estimation Izatt et al. (2017) and grasp force optimization Hauser et al. (2017). In this paper we show that our mixed-integer convex optimization approach can either produce an approximate solution to the inverse kinematics problem, or prove that the problem is globally infeasible for generic constraints.

Porta et al. (2009) used a similar approach to solve the IK problem. They relaxed the non-convex constraints by first cutting the variable intervals into small boxes, and computing the lower and upper bounds of the constraint within each small box. They then used a *branch-and-prune* procedure (Porta et al. (2005)) to discard the boxes that cannot possibly have a solution, and continue to divide the boxes likely to have a solution. Although our work shares similarities with Porta's approach in the mathematical formulation to relax $SO(3)$ constraint, the key distinction is that we formulate and solve the IK problem through mixed integer programming (MIP). There are some important advantages of our MIP approach: 1) Owing to the flexibility of MIP, our approach is capable of handling a rich set of kinematic constraints, including the collision avoidance constraint. We impose collision avoidance as a mixed-integer constraint, with integer variables denoting the assignment from robot links to collision free regions. On the other hand, Porta's approach cannot incorporate the collision avoidance constraint, as all the variables in their approach have continuous values. 2) Many robotics problems have been solved through mixed integer programming already, as we mentioned before. But all the aforementioned work ignores the accurate modelling on the kinematic constraints, as mixed-integer formulation of these constraints has not yet been available. For example, Aceituno-Cabezas et al. (2018) computed the gait pattern for a quadruped robot through MIP, to determine which foot to move in the next step. The reachability of each foot is only coarsely approximated as some spherical regions. We believe our mixed-integer convex formulation can readily incorporate the joint-level kinematic constraints into these planning problem, to improve the completeness and accuracy of the constraints. Another example of applying our formulation is in Izatt et al. (2017), where we estimated the object pose from the point cloud data, with binary variables to assign each scene point to a model feature, and we relied on the mixed-integer formulation from this paper to handle the kinematic constraint arising from the estimated object pose. 3) There has been abundant research on mixed integer programming Achterberg and Wunderling (2013), and there exist both commercial and open source solvers, such as Gurobi Optimization (2016), Mosek (2010), CPLEX (2016) and coin-or/CBC Forrest et al. (2018). As a result the users won't need to write their own algorithm (like *branch-and-prune*), making our approach easy to adopt. Compared to Porta's approach, the shortcoming of our approach is that the intervals have been cut before the solver starts, instead of adaptively being adjusted online through the *branch-and-prune* procedure. Hence we can't precisely control the accuracy of the approximated solution as Porta's approach does, and our solver is slower. Nevertheless, we believe that compared to Porta's approach, our MIP formulation grants the flexibility to handle more generic constraints (including collision avoidance), and can be incorporated into other robotics problems with integer variables.

Our contribution in this paper includes 1) A mixed-integer convex relaxation on the non-convex $SO(3)$ constraint. 2) Formulating the IK problem as a mixed-integer convex problem using the $SO(3)$ relaxation. As a result, we can reliably solve the IK problems with complicated constraints, and more importantly, we can also certify global infeasibility of the problem.

The paper is organized as follows: in Section 2 we introduce background knowledge on relaxing non-convex constraints as mixed-integer convex constraints, and the techniques to solve mixed-integer convex problems. In Section 3 we introduce our approach on formulating the IK problem as a mixed-integer optimization problem. In Section 4 we show our results of solving IK problems on robot arms, a quadruped robot and a parallel Stewart platform. Finally we conclude the paper in Section 5.

A conference version of this paper is published in Dai et al. (2017). In this journal version, we add the following improvements: 1) a section to give a brief background on mixed-integer programming. 2) An improved formulation to tackle collision avoidance constraints. 3) A different set of mixed-integer convex constraints to relax the $SO(3)$ constraint in the Appendix. 4) More results, including finding a collision free path of a gripper, and solving the forward kinematics problem of a parallel robot.

2 Background

2.1 Relaxing non-convex quadratic constraint

In this section we briefly review how to relax a non-convex quadratic constraint as mixed-integer convex constraints, which will be used to relax the non-convex quadratic $SO(3)$ constraint on the rotation matrix (1a)-(1c).

Consider a generic non-convex quadratic constraint

$$\{\mathbf{z}|a \le \mathbf{z}^{\mathrm{T}} Q\mathbf{z} + \mathbf{c}^{\mathrm{T}}\mathbf{z} \le b\}, \tag{2}$$

where the matrix Q is not necessarily positive semidefinite. This quadratic constraint contains bilinear product terms like xy (as the cross terms in $\mathbf{z}^{\mathrm{T}} Q\mathbf{z}$), visualized in Fig 2a -2b. It is NP-hard to obtain an exact global optimal solution to optimization problems with this non-convex constraints, but there has been a lot of research on obtaining an *approximated* one through mixed-integer-convex optimization McCormick (1976); Misener and Floudas (2012); Huchette and Vielma (2016). The key idea is to partition the range of each variable into smaller intervals, and replace the bilinear term xy with a linear term which approximates xy inside each interval. To see this geometrically, we draw the surface of $w = xy$ in

Fig 2 within $x \in [0,1], y \in [0,1]$. We then partition the range $[0,1]$ into smaller intervals, with interval i as $[\phi_i, \phi_{i+1}]$. The convex hull of $w = xy$ in interval $x \in [\phi_i, \phi_{i+1}], y \in [\phi_j, \phi_{j+1}]$ is a tetrahedron. This tetrahedron is also called the *McCormick envelope* of the surface $w = xy$. We can then relax the constraint $w = xy$, with the constraint that the point (x, y, w) lies within one of the tetrahedrons. We will use some binary variables to determine within which tetrahedron the point lies.

To enforce that (x, y, w) falls into one of the convex hulls, we rely on the *special ordered set of type 2* (sos2) constraint Beale and Tomlin (1970), defined as follows

Definition 2.1. Special ordered set of type 2: $\lambda = (\lambda_0, \ldots, \lambda_n) \in \mathbb{R}^{n+1}$ is in sos2, if

$$\sum_i \lambda_i = 1, \lambda_i \geq 0 \tag{3a}$$

$$\exists j \in 0, ..., n, \text{ s.t } \lambda_i = 0 \; \forall i \neq j, i \neq j+1. \tag{3b}$$

Namely at most two entries in λ can be strictly positive, and these two entries must be consecutive in their ordering.

The sos2 constraint is often used to formulate a piecewise linear approximation to a nonlinear function. For example in Fig 3, we approximate $f(x)$ with a new variable w satisfying

$$\begin{bmatrix} x \\ w \end{bmatrix} = \sum_i \lambda_i \begin{bmatrix} \phi_i \\ f(\phi_i) \end{bmatrix}, \quad \lambda \text{ is in sos2} \tag{4}$$

(4) forces the point (x, w) to be on the black lines in Fig 3.

We can formulate the sos2 (3) as mixed-integer linear constraints by introducing auxiliary binary variables. In this paper we adopt the formulation proposed in Vielma and Nemhauser (2011), which introduces $\log_2 n$ binary variables. We refer the readers to Vielma and Nemhauser (2011) for more details on sos2 constraint.

To impose the constraint that (x, y, w) is in one of the tetrahedron in Fig 2c-2d so as to approximate $w \approx xy$, we introduce auxiliary continuous variables $\gamma \in \mathbb{R}^{(n+1)\times(n+1)}, \alpha \in \mathbb{R}^{n+1}, \beta \in \mathbb{R}^{n+1}$ with the additional constraints

$$x = \sum_i \alpha_i \phi_i, \; y = \sum_j \beta_j \phi_j \tag{5a}$$

$$w = \sum_i \sum_j \gamma_{i,j} \phi_i \phi_j \tag{5b}$$

$$\sum_j \gamma_{i,j} = \alpha_i, \; \sum_i \gamma_{i,j} = \beta_j, \; \gamma_{i,j} \geq 0, \; \sum_{i,j} \gamma_{i,j} = 1 \tag{5c}$$

$$\alpha, \beta \text{ are in sos2 }. \tag{5d}$$

Constraint (5d) is formulated as mixed-integer linear constraints with auxiliary binary variables.

Notice that (x, y, w) satisfying $w = xy$ also satisfies constraint (5), namely the mixed-integer constraint is a *relaxation* of the non-convex constraint $w = xy$. Thus by replacing the bilinear term xy in the non-convex quadratic constraint (2) with (x, y, w) satisfying the mixed-integer convex constraint (5), if the mixed-integer program is infeasible, it guarantees that the original non-convex quadratic constraint (2) is globally infeasible.

To summarize, for a non-convex quadratic constraint (2), we can replace the bilinear term xy with w satisfying (5), and obtain mixed-integer convex constraints as a relaxation of the original non-convex constraint. An optimization problem with mixed-integer convex constraint can be solved through mixed-integer programming techniques, introduced in the next subsection 2.2.

2.2 Mixed-integer programming

A mixed-integer (convex) program (MIP) (without a cost) takes the following form

$$\text{find } \mathbf{x}, \mathbf{b} \tag{6a}$$

$$\text{s.t} \begin{bmatrix} \mathbf{x} \\ \mathbf{b} \end{bmatrix} \in \mathcal{C} \tag{6b}$$

$$\mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{b} \in \{0,1\}^{n_b} \tag{6c}$$

, where $\mathcal{C}$ is a convex set. $\mathbf{x}$ are the continuous variables, while $\mathbf{b}$ are the binary variables. Generally a mixed-integer program also minimizes a convex objective function. For simplicity of presentation, we will ignore the objective function in the optimization.

For mixed-integer convex program (6), we can either find its solution globally, meaning that it is guaranteed to find a solution if one exists, or we can prove that (6) is globally infeasible. A common algorithm for solving (6) is the *branch-and-cut* algorithm, which consists of *branch-and-bound* and *cutting planes* techniques. Here we give a brief introduction to these two techniques, in order to help the readers to understand how the solvers can detect global infeasibility, or obtain global solution in MIP.

The *branch-and-bound* algorithm (Bertsimas and Tsitsiklis (1997); Lawler and Wood (1966)), as illustrated in Fig. 4, builds a binary search tree. At each node of the tree a convex program is solved in order to either prove the node is infeasible, or find a solution of the node. In the root node, (6) is relaxed by replacing the integral constraint $\mathbf{b} \in \{0,1\}^{n_b}$ with linear constraints $\mathbf{0} \leq \mathbf{b} \leq \mathbf{1}$. If the relaxed convex program in the root node is infeasible, it certifies the original mixed-integer problem (6) is infeasible. Since the constraint set to the relaxed convex problem is a super set of the constraint in (6); if there doesn't exist a solution in the super set, then there doesn't exist a solution to the original constraints in (6). If this relaxed convex program is feasible, then two child nodes are created, by fixing one binary variable $\mathbf{b}_i$ to either 0 (left node) or 1 (right node). The solutions to the mixed-integer program (6) must satisfy the constraint in one of the child nodes. If both child nodes are infeasible, then (6) is proved to be globally infeasible, otherwise we continue to branch the feasible nodes by fixing the binary variables, until either a solution satisfying integral constraint is obtained, or the program in all leaf nodes are infeasible, which proves global infeasibility of (6). Although it seems that in the worst case the algorithm needs to traverse all 2^{n_b} nodes, empirically it usually needs only a small fraction of nodes to solve the problem. The reason is that many nodes are infeasible, thus do not need to be branched. As mentioned in a previous study (page 208 of Williams (2013)), sometimes branch-and-bound solves a problem with 100 binary variables by exploring just a few hundred nodes, about 10^{-26} percent of the entire tree (with $2^{100} \approx 10^{30}$

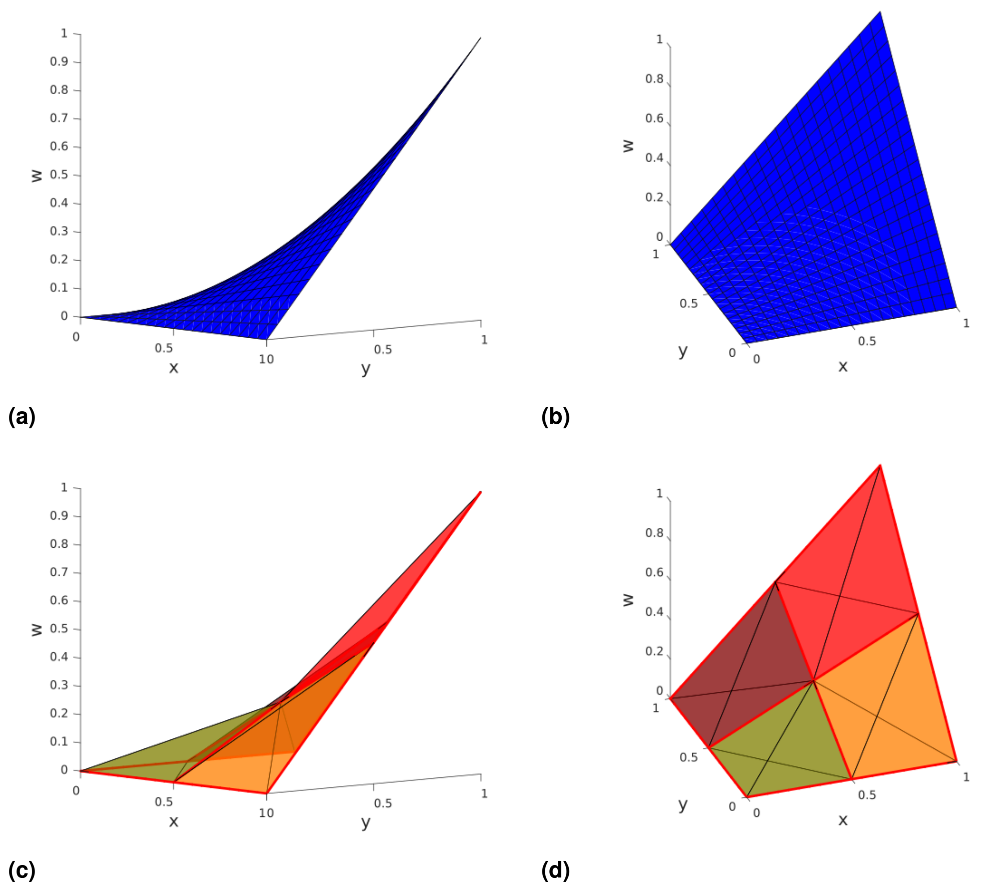


Figure 2. 2a,2b: $w = xy$ surface in the range $x \in [0,1], y \in [0,1]$ from two perspectives. In 2c,2d we divide the range $[0,1]$ into two intervals, with $(\phi_0, \phi_1, \phi_2) = (0, 0.5, 1)$, and compute the convex hull of the surface in each interval $x \in [\phi_i, \phi_{i+1}], y \in [\phi_j, \phi_{j+1}]$. Each convex hull is a tetrahedron drawn in distinct colors. A similar figure is drawn in Valenzuela (2016).

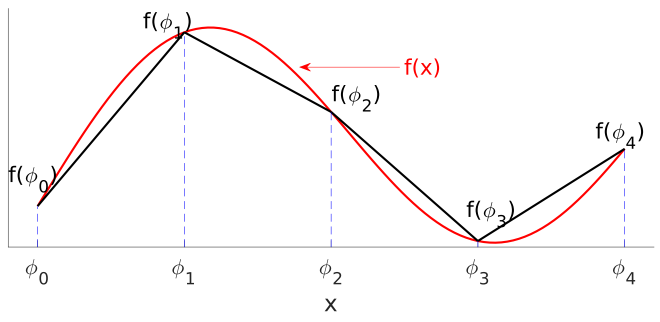
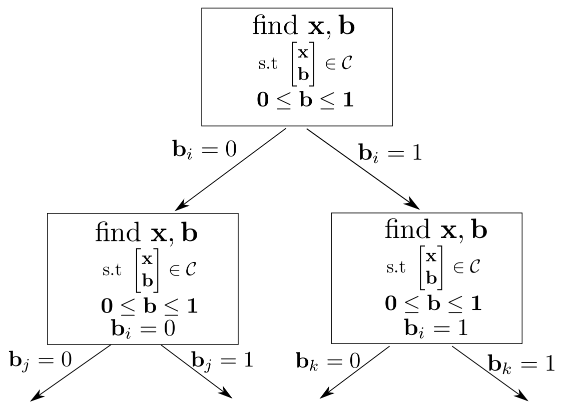


Figure 3. A piecewise linear function (black lines) approximating the nonlinear function $f(x)$ (red curve).



find $\mathbf{x}, \mathbf{b}$
s.t $\begin{bmatrix}\mathbf{x}\\ \mathbf{b}\end{bmatrix} \in \mathcal{C}$
$\mathbf{0} \le \mathbf{b} \le \mathbf{1}$
$\mathbf{b}_i = 0$
$\mathbf{b}_i = 1$
find $\mathbf{x}, \mathbf{b}$
s.t $\begin{bmatrix}\mathbf{x}\\ \mathbf{b}\end{bmatrix} \in \mathcal{C}$
$\mathbf{0} \le \mathbf{b} \le \mathbf{1}$
$\mathbf{b}_i = 0$
find $\mathbf{x}, \mathbf{b}$
s.t $\begin{bmatrix}\mathbf{x}\\ \mathbf{b}\end{bmatrix} \in \mathcal{C}$
$\mathbf{0} \le \mathbf{b} \le \mathbf{1}$
$\mathbf{b}_i = 1$
$\mathbf{b}_j = 0$
$\mathbf{b}_j = 1$
$\mathbf{b}_k = 0$
$\mathbf{b}_k = 1$

Figure 4. A binary search tree used in branch-and-bound process to solve the mixed-integer convex program (6).

nodes). It is worth noting that global infeasibility can usually be detected at the early stage of the branching process. In the result section, we will demonstrate that by formulating IK problem as a mixed-integer convex problem, we can usually detect global infeasibility in a few hundred milliseconds for problems with ≈ 100 binary variables.

The *cutting plane* method can quickly cut out some search regions, by exploiting the property of variables being integers. For example, for the constraint

$$5b_1 + 5b_2 + 4b_3 \le 8, \quad b_i \in \{0,1\} \tag{7}$$

we can generate a cut (inequality)

$$b_1 + b_2 + b_3 \le 1 \tag{8}$$

which is equivalent to (7) when all b_i are binary variables, but stronger than (7) in the continuous relaxation ($0 \le b_i \le 1$). Thus with the cutting planes, the continuous relaxation can search for a smaller region, and more likely to cut out globally infeasible regions. MIP solvers typically utilize the cutting plane method alongside the branch and bound algorithm during the solution process. For a more detailed introduction to cutting plane method, the readers could refer to Marchand et al. (2002).

3 Approach

As explained in Section 1, the non-convexity of the inverse kinematics problem originates from the non-convexity of the $SO(3)$ constraint. In order to obtain a global solution to this non-convex problem, in Section 3.1, we first relax the $SO(3)$ constraint (1a)-(1c), to a set of mixed-integer convex constraints, using the technique described in Section 2.1. Then in Section 3.2, we formulate the inverse kinematics problem by searching over the link poses (position/orientation) satisfying both the kinematic constraints, and the relaxed $SO(3)$ constraint introduced in Section 3.1. Finally, in Section 3.3 we project the approximated rotation matrices to the $SO(3)$ manifold to obtain the angle of each joint within the joint limits.

3.1 Rotation constraint relaxation

We aim to find a matrix $\bar{R} \in \mathbb{R}^{3\times 3}$ satisfying a relaxation of the $SO(3)$ constraint (1a)-(1c). To this end, we cut the range $[-1, 1]$ into n small intervals. The quadratic constraints (1b)-(1c) can be relaxed by replacing each bilinear term with a new variable approximating the bilinear term within each interval, described by (5). For example, to relax the orthogonality constraint $\mathbf{u}_1^T \mathbf{u}_2 = 0$, we introduce a slack variable $\mathbf{v} \in \mathbb{R}^3$, such that $\mathbf{v}(i)$ approximates $\mathbf{u}_1(i)\mathbf{u}_2(i)$, by requiring that $(\mathbf{u}_1(i), \mathbf{u}_2(i), \mathbf{v}(i))$ to be in one of the tetrahedrons in Fig. 2 (namely they satisfy the constraint (5)). And we instead impose the linear constraint $\mathbf{v}(1) + \mathbf{v}(2) + \mathbf{v}(3) = 0$ as a relaxation of the orthogonality constraint $\mathbf{u}_1^T \mathbf{u}_2 = 0$.

To relax the unit length constraint $\mathbf{u}_i^T \mathbf{u}_i = 1$ (1a), we first introduce auxiliary variables $\mathbf{w}_i \in \mathbb{R}^3, \lambda^{i,j} \in \mathbb{R}^{n+1}$ satisfying

$$\begin{bmatrix}\mathbf{u}_i(j)\\ \mathbf{w}_i(j)\end{bmatrix} = \sum_{k=0}^{n} \lambda_k^{i,j} \begin{bmatrix}\phi_k\\ \phi_k^2\end{bmatrix}, \quad \lambda^{i,j} \text{ is in sos2.} \tag{9}$$

Constraint (9) forces the point $(\mathbf{u}_i(j), \mathbf{w}_i(j))$ to be on the black lines in Fig 5, for $n = 4$ case. Namely $\mathbf{w}_i(j)$ is an upper-bound of $\mathbf{u}_i(j)^2$. We can thus relax the unit length constraint (1a) as

$$\mathbf{u}_i^T \mathbf{u}_i \le 1 \tag{10a}$$
$$\mathbf{w}_i(1) + \mathbf{w}_i(2) + \mathbf{w}_i(3) \ge 1. \tag{10b}$$

Geometrically, constraint (10a) forces the rows/columns of the rotation matrix to be within the unit sphere, while constraint (10b) pushes the row/column vector from inside the unit sphere, such that the Euclidean length of the row/column vector is bounded from below.

While the generic technique introduced in Section 2.1 can already approximate the orthogonality constraint, it is desirable to introduce an additional relaxation, because enforcing the solution to live in the intersection of different relaxations can only improve its approximation quality. We thus impose the following convex (quadratic) constraints as a relaxation to the orthogonal constraint (1b)

$$|\mathbf{u}_i \pm \mathbf{u}_j|_2^2 \le 2, \tag{11}$$

where $\mathbf{u}_i, \mathbf{u}_j$ are two distinctive rows/columns of the matrix $\bar{R}$, $|\bullet|_2$ is the l_2 norm of a vector. Constraint (11) is

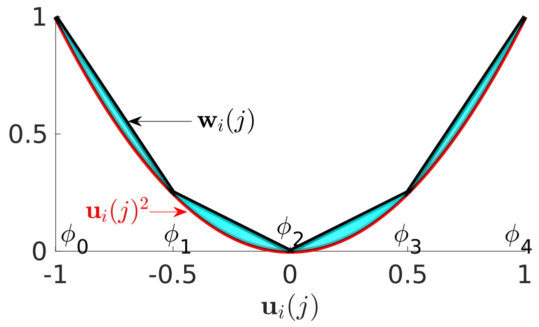


Figure 5. The range of $\mathbf{u}_i(j)$ is cut into 4 intervals, with $(\phi_0, \phi_1, \phi_2, \phi_3, \phi_4) = (-1, -0.5, 0, 0.5, 1)$. The adjacent two points (ϕ_i, ϕ_i^2) and $(\phi_{i+1}, \phi_{i+1}^2)$ are connected with the black straight line. We introduce an auxiliary variable $\mathbf{w}_i(j)$ to be on the black lines, as an upper bound of the red curve $\mathbf{u}_i(j)^2$. The unit length constraint $\sum_j \mathbf{u}_i(j)^2 = 1$ is relaxed as $\sum_j z_i(j) = 1$, where $z_i(j)$ lies within the shaded cyan region, as a relaxation of the red curve $\mathbf{u}_i(j)^2$.

equivalent to $|\mathbf{u}_i^T \mathbf{u}_j| \le \frac{1}{2}(2 - \mathbf{u}_i^T \mathbf{u}_i - \mathbf{u}_j^T \mathbf{u}_j)$. When the vectors $\mathbf{u}_i, \mathbf{u}_j$ are "close to" the unit sphere (due to the constraint (10a)-(10b)), constraint (11) approximates the orthogonal constraint $\mathbf{u}_i^T \mathbf{u}_j = 0$. Similarly we can impose the convex constraint $|\mathbf{u}_1 \pm \mathbf{u}_2 \pm \mathbf{u}_3|_2^2 \le 3$ as an relaxation of the orthogonal constraint, where $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ are all three rows/columns of $\bar{R}$.

To conclude this section, the non-convex $SO(3)$ constraint is thus relaxed to a set of mixed-integer convex constraints. These mixed-integer constraints are readily handled by modern numerical solvers such as Gurobi Optimization (2016). We also note that this mixed-integer convex relaxation of $SO(3)$ can be used beyond inverse kinematics problems. For example, it has been applied to estimating the object pose from point cloud data (Izatt et al. (2017)), and the code is open-sourced for general use in Drake (Tedrake and the Drake Development Team (2016)).

3.2 Inverse kinematics formulation

We aim to solve the inverse kinematics problem by finding the pose of each link. The pose of link i can be represented by rigidly attaching a frame to the link, with the position ${}^W\mathbf{p}_i \in \mathbb{R}^3$ and the rotation matrix ${}^W R_i$, where the right subscript i denotes it is link i's frame we are interested in, the left superscript ^{W} indicates the quantity is expressed in the *world* frame. In Fig 6, we illustrate the kinematics relationship between two links connected by a revolute joint. We will express this kinematics relation as constraints on the position ${}^W\mathbf{p}_i$ and orientation ${}^W R_i$ of link i expressed in the world frame, and the position/orientation of its parent link $i-1$. A joint frame J is rigidly attached to the parent link $i-1$, and coincides with the child link frame, when the joint angle $q_i = 0$. We denote the fixed translation of joint frame to the parent link frame as ${}^{i-1}\mathbf{p}_J$. The joint axis is expressed as ${}^{i-1}\hat{\mathbf{z}}_i$ in the parent link $(i-1)$'s frame, and ${}^i\hat{\mathbf{z}}_i$ in the child link i's frame. Since the position and direction of joint axis is invariant under rotation about this axis, we can impose the following constraints on the poses of the parent and child

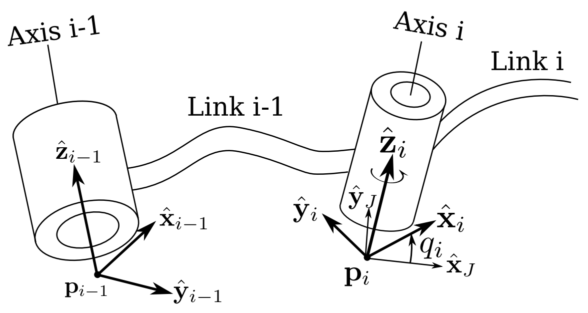


Figure 6. The kinematics relation between adjacent link $i-1$ and i Craig (2005). The frame with axes $\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{z}}_i$ is rigidly attached to the child link i, the position of the frame origin is $\mathbf{p}_i$. Without loss of generality we can assume the rotation axis i is along the z axis $\hat{\mathbf{z}}_i$. Similarly for the parent link $i-1$. The joint frame J with axes $\hat{\mathbf{x}}_J, \hat{\mathbf{y}}_J, \hat{\mathbf{z}}_J$ is rigidly attached to the parent link $i-1$; this joint frame coincides with the child link frame when the joint angle $q_i = 0$.

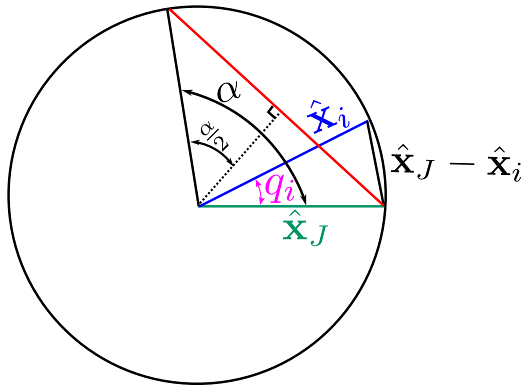


Figure 7. The visual illustration of the joint limit constraint (13). The red chord is the difference between unit vector $\hat{\mathbf{x}}_J$ and another vector, obtained by rotating $\hat{\mathbf{x}}_J$ about an axis by angle α. The chord's length is $2\sin\alpha/2$. $\hat{\mathbf{x}}_J - \hat{\mathbf{x}}_i$ should have shorter length than the red chord, since the angle q_i between these two vectors is smaller than α.

links

$$
\begin{aligned}
{}^W\mathbf{p}_i &= {}^W\mathbf{p}_{i-1} + {}^W R_{i-1}\,{}^{i-1}\mathbf{p}_J && \text{(12a)}\\
{}^W R_{i-1}\,{}^{i-1}\hat{\mathbf{z}}_i &= {}^W R_i\,{}^i\hat{\mathbf{z}}_i && \text{(12b)}
\end{aligned}
$$

(12) are linear constraints on the decision variables ${}^W R_i$, ${}^W R_{i-1}$, ${}^W\mathbf{p}_i$, ${}^W\mathbf{p}_{i-1}$.

3.2.1 Joint limits We also handle the angle limits on the revolute joint. Without loss of generality, we assume that angle limits for joint i are $-\alpha \le q_i \le \alpha$. In order to enforce this joint limit constraint with only link poses as decision variables, we consider a unit length vector $\hat{\mathbf{x}}_J$, perpendicular to the joint axis $\hat{\mathbf{z}}_i$, and how it relates to itself before and after joint rotation. For example in Fig 6, rotating the i'th joint by q_i transforms the unit length vector $\hat{\mathbf{x}}_J$ to $\hat{\mathbf{x}}_i$. The joint limit constraint $|q_i| \le \alpha$ can be imposed as

$$
\begin{aligned}
&|\angle({}^W\hat{\mathbf{x}}_J, {}^W\hat{\mathbf{x}}_i)| \le \alpha\\
\Leftrightarrow\; &{}^W\hat{\mathbf{x}}_J^{\mathrm{T}}\;{}^W\hat{\mathbf{x}}_i \ge \cos\alpha\\
\Leftrightarrow\; &\left|{}^W\hat{\mathbf{x}}_J - {}^W\hat{\mathbf{x}}_i\right|_2 \le 2\sin\frac{\alpha}{2}\\
\Leftrightarrow\; &\left|{}^W R_{i-1}\,{}^{i-1}\hat{\mathbf{x}}_J - {}^W R_i\,{}^i\hat{\mathbf{x}}_i\right|_2 \le 2\sin\frac{\alpha}{2} \qquad (13)
\end{aligned}
$$

where $\angle(\cdot,\cdot)$ is the angle between two vectors. ${}^{i-1}\hat{\mathbf{x}}_J$ is the given unit length vector fixed and expressed in the parent $i-1$ frame, and ${}^i\hat{\mathbf{x}}_i$ is the given unit length vector fixed and expressed in the child i frame. Equation (13) is a convex (quadratic) constraint on the decision variables (parent and child link orientations ${}^W R_i$, ${}^W R_{i-1}$). The joint limit constraint (13) would be tight if ${}^W R_i$, ${}^W R_{i-1}$ satisfied $SO(3)$ constraints exactly.

3.2.2 Kinematic constraints on link position and orientation We can also show that the kinematic constraints on link position and orientation can be readily formulated as convex constraints on the link poses. For example, the position of a point Q rigidly attached to a link i is a linear function of the decision variables, as ${}^W\mathbf{p}_Q = {}^W\mathbf{p}_i + {}^W R_i\,{}^i\mathbf{p}_Q$, where ${}^i\mathbf{p}_Q$ is the position of point Q fixed in the link i's body frame. Hence constraining the point Q to be within a convex region $\mathcal{S}$ is a convex constraint on the link pose ${}^W\mathbf{p}_i$, ${}^W R_i$.

$$
{}^W\mathbf{p}_i + {}^W R_i\,{}^i\mathbf{p}_Q \in \mathcal{S} \qquad (14)
$$

To constrain that the angle difference between a link orientation ${}^W R_i$ and a desired orientation R^* is less than a tolerance γ, we utilize Rodriguez Formula (15), that the rotation matrix corresponding to rotation axis $\mathbf{a} \in \mathbb{R}^3$ and rotation angle θ is

$$
R(\mathbf{a},\theta) = I_{3\times 3} + \sin\theta\lfloor\mathbf{a}\rfloor_\times + (1-\cos\theta)\lfloor\mathbf{a}\rfloor_\times^2 \qquad (15)
$$

where $\lfloor\mathbf{a}\rfloor_\times \in \mathbb{R}^{3\times 3}$ is the skew-symmetric matrix representing the cross product with $\mathbf{a}$. From (15), it can be readily shown that $\text{trace}(R(\mathbf{a},\theta)) = 1 + 2\cos\theta$. Hence we can impose the following convex (quadratic) constraint to enforce the angle between link orientation ${}^W R_i$ and the desired orientation R^* is less than γ

$$
\begin{aligned}
|{}^W R_i - R^*|_F^2 &= \text{trace}(\underbrace{{}^W R_i^T\,{}^W R_i}_{=I_{3\times3}} + \underbrace{(R^*)^T R^*}_{=I_{3\times3}} - 2\,{}^W R_i^T R^*)\\
&= 6 - 2\,\text{trace}({}^W R_i^T R^*)\\
&= 6 - 2\,\text{trace}(R(a,\theta))\\
&= 4 - 4\cos\theta\\
&\le 4 - 4\cos\gamma \qquad (16)
\end{aligned}
$$

where $|\cdot|_F$ is the Frobenius norm, $|X|_F^2 = \text{trace}(X^{\mathrm{T}}X) = \sum_i\sum_j X(i,j)^2$.

Notice that the constraints on the link position ((14)) and orientation ((16)) can be used in finding a continuous path of the robot. The "continuity" requirement can be imposed by constraining that between the adjacent knots, the angle difference between link orientations R_1, R_2 to be less than a threshold γ as $|R_1 - R_2|_F^2 \le 4 - 4\cos\gamma$, and the position difference between link position $\mathbf{p}_1, \mathbf{p}_2$ to be less than distance d as $|\mathbf{p}_1 - \mathbf{p}_2| \le d$. We will demonstrate finding a continuous path in the result section 4.2.4.

3.2.3 Collision avoidance To impose collision avoidance constraints, we use a similar formulation as Blackmore et al. (2010); Deits and Tedrake (2015b). As shown in Fig.8c, we

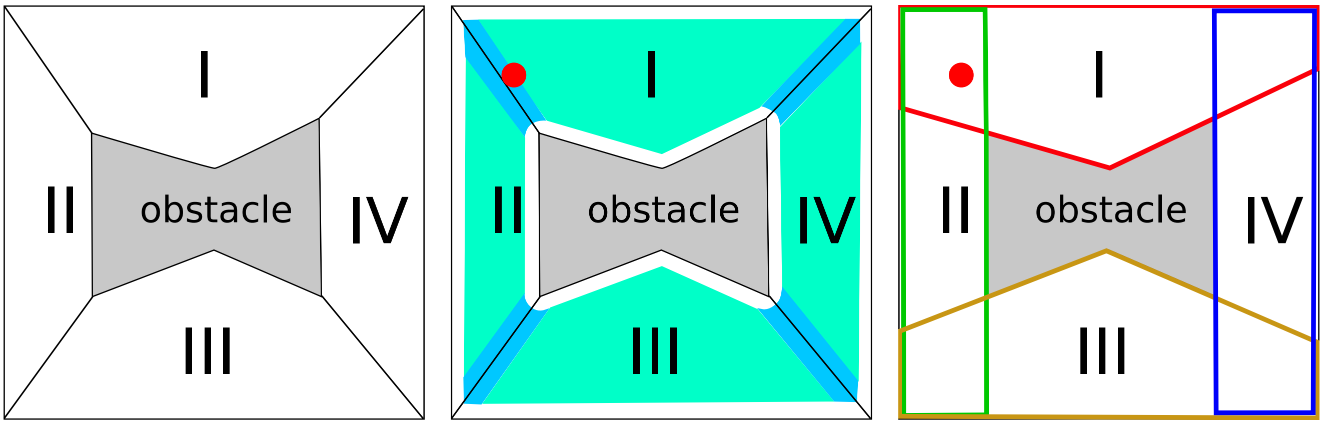


Figure 8. The collision free region is represented as unions of polytopes, and the entire sphere (or point) has to be in one of the polytopes. In 8a, the collision free region is partitioned into non-overlapping polytopes. This works for a point object, but not when the object is a sphere (red) as in 8b: the center of the sphere can only be in the cyan region (obtained by shrinking each polytope boundary by the radius of the sphere). The problem is that the sphere center can't be in the blue region, as part of the sphere is in one polytope, while the other part is in another polytope, although the spheres would be collision free. The solution is shown in 8c, that the collision free polytopes should overlap with each other.

represent the collision free space as unions of N bounded polytopes, that intentionally overlap each other. We represent the collision geometry of the robot as unions of spheres and points, and constrain that each of the spheres and points should be in one of the collision free polytopes. We denote the i'th polytope parameterized as $\mathcal{P}_i = \{\mathbf{x}|A_i\mathbf{x} \le \mathbf{b}_i\}$, and assume that the sphere is centered at $\mathbf{p}$ with radius r (a point is a sphere with 0 radius). The sphere being in the i'th polytope is equivalent to $A_i\mathbf{p} \le \mathbf{b}_i - \mathbf{a}_i r$, where $\mathbf{a}_i(j) = |A_i(j,:)|$, namely $\mathbf{a}_i(j)$ is the norm of the j'th row of A_i. We introduce binary variables $\mathbf{z} \in \{0,1\}^N$, such that $\mathbf{z}_i = 1$ assigns the sphere to the polytopes $\mathcal{P}_i$. We also introduce continuous variables $\mathbf{y}_i$ that would be equal to $\mathbf{p}$ if the sphere is assigned in polytope $\mathcal{P}_i$, and equal to 0 otherwise. The collision free constraint is imposed as the following mixed-integer linear constraint

$$\mathbf{p} = \mathbf{y}_1 + \ldots + \mathbf{y}_N \tag{17a}$$

$$A_i\mathbf{y}_i \le (\mathbf{b}_i - \mathbf{a}_i r)\mathbf{z}_i, \quad i = 1,\ldots,N \tag{17b}$$

$$\mathbf{z}_1 + \ldots + \mathbf{z}_N = 1, \quad \mathbf{z}_i \in \{0,1\} \tag{17c}$$

Note that when $\mathbf{z}_i = 0$, $A_i\mathbf{y}_i \le 0$ ((17b)) is equivalent to $\mathbf{y}_i = 0$ due the boundedness of the polytope (Proof by contradiction, suppose $\mathbf{y}_i \ne 0, A_i\mathbf{y}_i \le 0$, then for a point $\hat{\mathbf{y}}$ in the polytope $\mathcal{P}_i$ satisfying $A_i\hat{\mathbf{y}} \le \mathbf{b}_i$, the ray $\hat{\mathbf{y}} + t\mathbf{y}_i, t \ge 0$ is always in the polytope $\mathcal{P}_i$, contradicting to the boundedness assumption.) (17c) means that only one binary variable in $\mathbf{z}$ will be 1, hence all $\mathbf{y}_i$ except one will be non-zero, and $\mathbf{p}$ (equal to this non-zero $\mathbf{y}_i$) is in the collision free polytope $\mathcal{P}_i$.

It is worth noting that $\mathbf{z}_i = 1$ implies that the sphere is in polytope $\mathcal{P}_i$, but the converse is not true, that the sphere lying in $\mathcal{P}_i$ doesn't necessarily imply that $\mathbf{z}_i = 1$. For example, in Fig.8c, when the sphere is in the overlapping region of polytope $\mathcal{P}_1$ and $\mathcal{P}_2$, the solution to constraint (17a)-(17c) could be either $\mathbf{z} = [1,0,0,0]$ or $\mathbf{z} = [0,1,0,0]$, both are equally valid solutions. In either case, the sphere is assigned to one of the polytopes containing the overlapping region.

We assume that the collision free polytopes $\mathcal{P}_i$ are already given, through some pre-processing steps. There are numerous approaches to represent the free space as unions of polytopes. For example, Deits and Tedrake (2015a) computed the large collision free regions through convex optimization. We can also use simple bounding boxes to segment the free space, as shown in the Result section 4.2.

With all the kinematic constraints formulated as mixed-integer convex constraints in this subsection, we can solve the IK problem through solvers like Gurobi Optimization (2016) or Mosek (2010). If the solver reports the MIP being infeasible, then we get a certificate that the IK problem is globally infeasible.

3.3 Reconstruct joint angle by projecting back to $SO(3)$

After obtaining the approximated solution to the body orientations through the mixed-integer convex optimization, we need to recover the value of each joint angle. Since the solution to the body orientation does not satisfy the $SO(3)$ constraint exactly, we project the approximated solution to the rotation constraint, starting from the root link. If the link is floating, then the optimal projected solution is obtained as UV^T, where $\bar{R} = U\Sigma V^\text{T}$ is the SVD of $\bar{R}$, as proved in Haralick et al. (1989). On the other hand, if the link is connected to its parent link through a revolute joint, we project ${}^WR_{i-1}^\text{T}\,{}^W\bar{R}_i$ onto $SO(3)$ with the given rotation axis and angle limits, where ${}^WR_{i-1}$ is the rotation matrix of the parent $i-1$ link, computed by doing forward kinematics using the recovered posture from the root to the $i-1$ link. To project ${}^WR_{i-1}^\text{T}\,{}^W\bar{R}_i$, we find the joint angle q_i such that the joint rotation matrix $R({}^{i-1}\hat{\mathbf{z}}_i, q_i)$ has the minimal error to ${}^WR_{i-1}^\text{T}\,{}^W\bar{R}_i$ under the joint limits (Fig 6). $R({}^{i-1}\hat{\mathbf{z}}_i, q_i) \in SO(3)$ means rotating by angle q_i about the axis ${}^{i-1}\hat{\mathbf{z}}_i$. Algebraically, the optimal q_i is the solution to the following program

$$\min_{q_i} \left| R({}^{i-1}\hat{\mathbf{z}}_i, q_i) - {}^W R_{i-1}^{\mathrm{T}}\, {}^W \bar{R}_i \right|_F^2 \tag{18a}$$

$$\text{s.t } -\alpha \le q_i \le \alpha. \tag{18b}$$

In the objective function $|\cdot|_F$ is the matrix Frobenius norm.

As in (15), we again use Rodriguez Formula to compute the rotation matrix $R({}^{i-1}\hat{\mathbf{z}}_i, q_i)$ from its rotation angle and rotation axis

$$R({}^{i-1}\hat{\mathbf{z}}_i, q_i) = I_{3\times3} + \sin q_i \lfloor {}^{i-1}\hat{\mathbf{z}}_i \rfloor_\times + (1-\cos q_i)\lfloor {}^{i-1}\hat{\mathbf{z}}_i \rfloor_\times^2 \tag{19}$$

Substituting equation (19) into the optimization objective (18a), and taking the derivative of the cost function w.r.t angle q_i, we know the optimality occurs when either the gradient is zero with positive Hessian, or at the boundary points. Thus we can obtain the optimal q_i in the closed-form as

$$\begin{cases} \text{if } \exists k \in \mathbb{Z}, -\alpha \le \beta + 2k\pi \le \alpha, & q_i = \beta + 2k\pi \\ \text{else:} & \\ \quad \text{if } -\cos(-\alpha-\beta) \le -\cos(\alpha-\beta), & q_i = -\alpha \\ \quad \text{if } -\cos(-\alpha-\beta) > -\cos(\alpha-\beta), & q_i = \alpha. \end{cases} \tag{20}$$

where $\beta = \text{atan2}\left(\text{trace}(MA), -\text{trace}(MA^2)\right), A = \lfloor {}^{i-1}\hat{\mathbf{z}}_i \rfloor_\times, M = {}^W R_i^{\mathrm{T}}\, {}^W \bar{R}_{i-1}$. Note that at value $\beta + 2k\pi$ the derivative of the objective (18a) is zero, i.e., it is the optimal solution to (18a) without the joint limits (18b). The optimal solution is truncated to the joint bounds if $\beta + 2k\pi$ is outside of the angle limits. q_i in (20) is the optimal joint angle within the joint limits, that minimizes the projection error from the mixed-integer convex optimization solution, to $SO(3)$ with a given rotation axis and angle bounds. The joint limit truncation will cause large constraint violation error in the reconstructed postures.

With (20) to compute a single joint angle, we sweep the robot kinematic tree from the root link to the leaf links, to compute the posture including every joint. Since the projection error is accumulated along the kinematic path from the root link, we then sweep back from the leaf links to the root link, to compute the matrix ${}^W \bar{R}_i$ through forward kinematics, using the leaf link pose and the reconstructed joint angles from link i to the leaf links. The complete procedure to reconstruct the joint angles is summarized in Algorithm 1.

Algorithm 1 Reconstruct a robot posture from mixed-integer optimization result.

```
 1: $iter = 1$.
 2: while $iter <$ MaxIteration do
 3:     $i = 1, {}^W R_0 = I_{3\times3}$.
 4:     ${}^W \bar{R}_i$ =GetBodyApproximatdPose($iter$)
 5:     while Not all joint angles are computed do
 6:         if Joint $i$ is a floating joint then
 7:             Compute ${}^W R_i = UV^{\mathrm{T}}$ where ${}^W \bar{R}_i = U\Sigma V^{\mathrm{T}}$.
 8:         else
 9:             Obtain the joint angle $q_i$ from (20).
10:             Do forward kinematics to compute the orientation of link $i$ as ${}^W R_i = {}^W R_{i-1} R({}^{i-1}\hat{\mathbf{z}}_i, q_i)$.
11:         end if
12:         $i \leftarrow i + 1$.                    ▷ Go to the child link.
13:     end while
14:     $iter \leftarrow iter + 1$
15: end while
```

Algorithm 2 GetBodyApproximatedPose($iter$)

```
1: if $iter == 1$ then
2:     ${}^W \bar{R}_i$ is the solution to mixed integer program.
3: else
4:     ${}^W \bar{R}_i$ is obtained by doing forward kinematics from the leaf link to link $i$ using the reconstructed posture $q$ from the previous iteration.
5: end if
```

4 Results

We present the results on the $SO(3)$ relaxation and the inverse kinematics algorithm. These results were collected on 64 bit Intel Xeon E3-1505M 2.8GHz CPUs with Gurobi 7.5 Gurobi Optimization (2016) as the solver. The computation time reported in this section is the Gurobi solver time.

4.1 Tightness of rotation matrix relaxation

We first show the tightness of our mixed-integer convex relaxation of the $SO(3)$ constraint on the rotation matrix. If $\bar{R} \in \mathbb{R}^{3\times3}$ satisfies the relaxed constraints, then we know that $|\bar{R}\mathbf{v}|_2 \approx |\mathbf{v}|_2\ \forall \mathbf{v} \in \mathbb{R}^3$, and $\angle(\bar{R}\mathbf{v}_1, \bar{R}\mathbf{v}_2) \approx \angle(\mathbf{v}_1, \mathbf{v}_2)$. Geometrically the relaxation allows the transformation $\mathbf{v} \to \bar{R}\mathbf{v}$ to change the length of a vector, and to perturb the angle between two vectors. In this section we show how much shorter the transformed vector can be, and also derive a bound on the angle error.

We find the global optimum to two mixed-integer convex optimization programs, to determine the tightness of the relaxation. First we compute the global minimum of $\min_{\bar{R},i} |\bar{R}\mathbf{e}_i|_2$, where $\mathbf{e}_i$ is the unit length vector with $\mathbf{e}_i(i) = 1$. To determine the angle error, we compute the global minimum $\min_{\bar{R} \text{ in relaxation}, i\neq j} |\bar{R}\mathbf{e}_i + \bar{R}\mathbf{e}_j|_2$. If we denote this global minimum as d, we obtain loose bounds on the angle after transformation as $180° - \arccos(d^2/2 - 1) \le \angle(\bar{R}\mathbf{e}_i, \bar{R}\mathbf{e}_j) \le \arccos(d^2/2-1)$. We compute the error bounds on the relaxation, by partitioning the range $[-1, 1]$ into $n = 2, 4$ or 8 intervals, with $\phi_i = -1 + 2i/n$. The results are in Table 1. The last column with infinite number of binary variables is the ideal $\bar{R} \in SO(3)$ case without any relaxation. The bounds on $|\bar{R}e_i|_2$ and $|\bar{R}e_i + \bar{R}e_j|_2$ are tight, meaning that there are $\bar{R}$ satisfying our mixed-integer relaxation that gives the values in Table 1, but the bound on the angle in the last row of the table is not tight.

As the number of intervals increases, the convex hull of xy in each interval is closer to the bilinear product. As a result in Table 1, the minimal length of $\bar{R}e_i$ increases, and the bound on the angle error shrinks, leading to a tighter relaxation of $SO(3)$. In the inverse kinematics problem, we choose to cut

# of intervals	2	4	8	∞
$\min_{\bar{R}} \left\|\bar{R}\mathbf{e}_i\right\|_2$	0.57735	0.90453	0.97641	1
$\min_{\bar{R}} \left\|\bar{R}\mathbf{e}_i + \bar{R}\mathbf{e}_j\right\|_2$	0.60229	1.22474	1.36288	1.414
$\angle(\bar{R}e_i, \bar{R}e_j) - 90°$	$[-54.9°, 54.9°]$	$[-14.5°, 14.5°]$	$[-4.1°, 4.1°]$	$[0°, 0°]$

Table 1. Tightness of the mixed-integer convex relaxation on $SO(3)$.

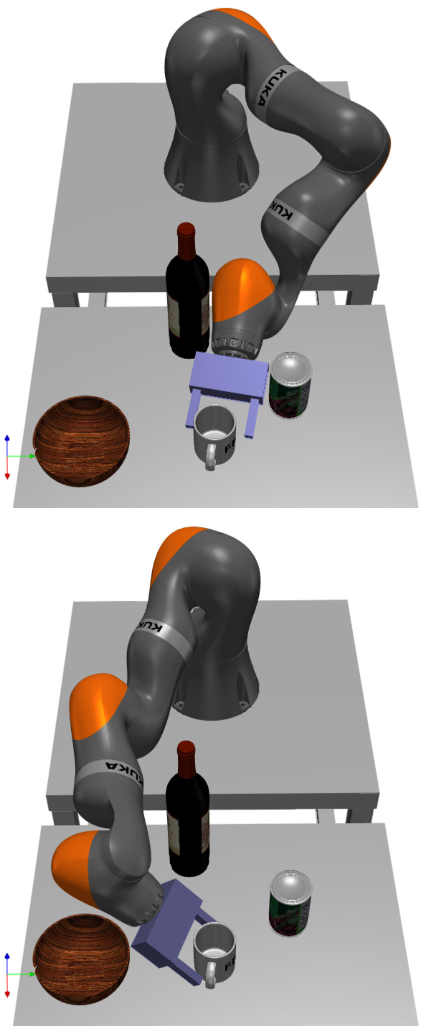


Figure 9. KUKA IIWA arm grasping a mug on the table with two different postures. The mesh files of the objects are obtained from Shapenet Chang et al. (2015).

$[-1, 1]$ into 4 intervals by default, as a compromise between computation speed and relaxation accuracy.

4.2 Inverse Kinematics

4.2.1 KUKA IIWA arm We first show that our inverse kinematics approach can find postures for complicated kinematics constraints including inequalities, such as those arising from the task "grasping a mug on the table in a cluttered environment" in Fig 9. We use a KUKA IIWA arm and a Schunk gripper with 7 total joints. We constrain the gripper to achieve an antipodal grasp on the mug. Rather than finding a single solution, the branch-and-bound algorithm for mixed-integer problem can find multiple solutions, corresponding to different sets of active binary variables. In Fig 9, we show two distinct postures found by solving the mixed-integer convex program. These two postures belong to different homotopy classes, grasping the mug from either left or right side of the wine bottle. The collision geometry of the robot is represented by a union of points and spheres in Fig 10a, and the free space as unions of 6 overlapping boxes in Fig 10b. The computation took 7 seconds in the Gurobi solver. The antipodal grasp constraint is violated by about 0.5 cm in the reconstructed posture, due to the relaxation of the $SO(3)$ constraint.

As a comparison, we also solve the same IK problem with the gradient-based nonlinear optimization based IK solver (Dai et al. (2014)) in the software package Drake (Tedrake and the Drake Development Team (2016)). This IK solver can also incorporate collision avoidance as a nonlinear constraint. With the initial guess of all joint angles being 0 (the posture shown in the bottom plot of Fig 11), the nonlinear optimization IK fails, as the initial guess is singular with 0 gradient. We also attempt 10 randomly chosen initial guesses, and 6 of them find the collision free postures through nonlinear optimization. The computation time is usually about 0.3 s with the nonlinear optimization based IK, and the constraint violation is less than 0.1 mm. This comparison shows that the MIP IK is slower and less accurate than the nonlinear optimization based IK. The advantage of MIP IK is being insensitive to the initial guess.

We also show that our approach can prove the global infeasibility of some kinematic constraints. In Fig 11, we impose the task constraint that the hand should grasp the wine bottle, by keeping its center on the green line segment, and its orientation aligned with the bottle's longitudinal axis. In the left plot the mixed-integer program is solved successfully, while in the right plot with the fridge on the table, the collision avoidance constraint causes the mixed-integer convex program to be globally infeasible. The solver takes 0.75 seconds to detect the global infeasibility.

4.2.2 Little Dog For more complicated robots like LittleDog (Fig 12, 13), we show that our approach can find postures of the robot with each leg standing on one of the stepping stones. This constraint is imposed similar to the collision avoidance constraint (17a)-(17c), but replacing each polytope with the top face of each stepping stone, and use binary variables to assign the tip of each foot to one of the stepping stones. In Fig 12, our mixed-integer solver finds different postures of the robot, with its feet on distinct sets of stepping stones. In Fig 13 the mixed-integer convex IK spends 0.25s to determine that there does not exist a posture for the robot to put each of its toes on one of the violet stepping stones, while avoiding the red obstacles.

4.2.3 ABB IRB140 6R robot arm To get some statistics on our mixed-integer convex IK approach, we test its performance on an ABB IRB140 arm with 6 DoF, for which the IK problem can be solved analytically Craig (2005). In Fig 14, we show the results of running both analytical and mixed-integer convex IK on the robot. We solve the

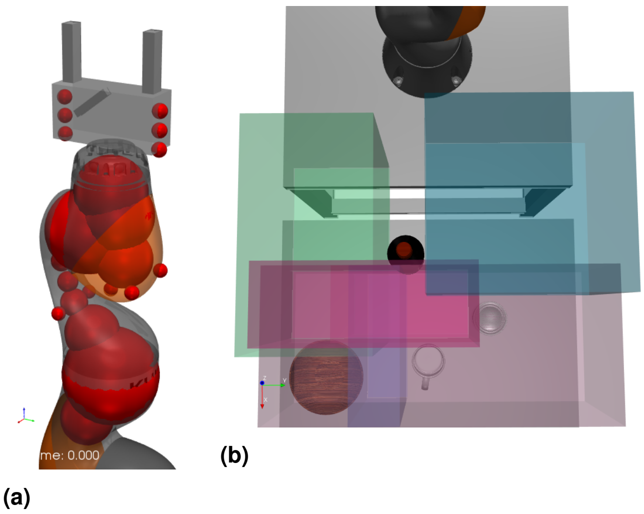


Figure 10. The collision geometry of the robot is represented by a union of spheres and points in 10a. Note that due to the antipodal grasp constraint (the gripper is open, the midpoint between the two grippers coincides with the object center, with appropriate constraint on the orientation of the gripper), we don't need to add collision spheres on the grippers. Satisfying the antipodal grasp automatically guarantees that the gripper fingers don't collide with the object. The free space of Fig 9 is represented as the union of axis-aligned bounding boxes in Fig 10b (seen from the top). These axis-aligned bounding boxes are created manually, by expanding a point along each axis until visually touching the obstacles.

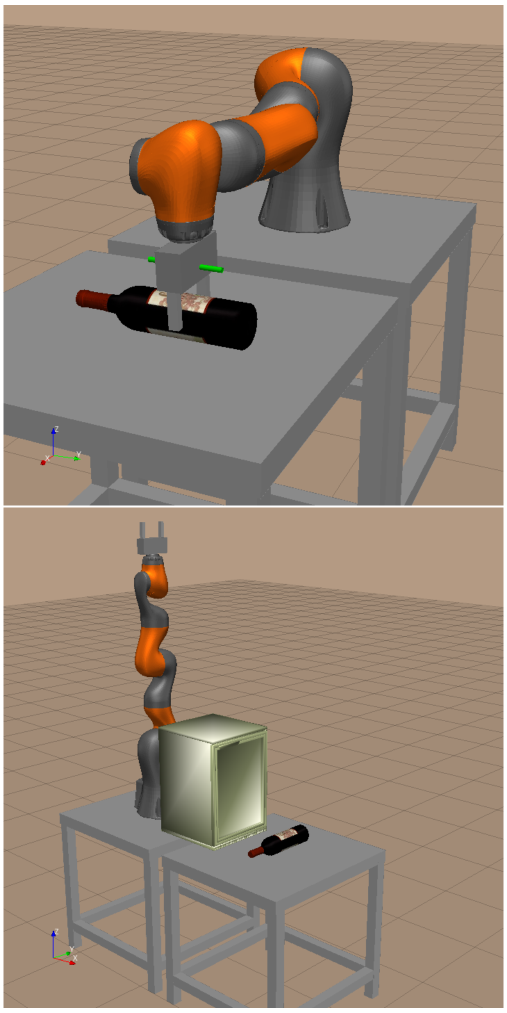


Figure 11. KUKA IIWA arm grasping the wine bottle. On the left, the IK problem is solved successfully. The center of the hand is constrained to be on the green line segment. On the right, with the fridge on the table, the mixed-integer convex program proves there is no posture that can satisfy the same grasping constraint, while keeping the robot from colliding with the fridge.

IK problem to reach each sampled position with the given orientation. There are three categories of the solutions

- Green dot. Both analytical and mixed-integer convex IK find the solution.
- Blue dot. Both analytical and mixed-integer convex IK prove global infeasibility.
- Red dot. Analytical IK proves the problem is infeasible, but mixed-integer convex IK thinks the problem is feasible, due to the constraint relaxation.

We test the MIP IK solver with different number of intervals $[\phi_i, \phi_{i+1}]$ per entry in rotation matrix R. As explained in section 2.1, the range $[-1, 1]$ of each entry in R is divided into n intervals, each with length $2/n$. So each entry in R requires $\log_2 n$ binary variables. We show the status of MIP solvers for each sample point in Fig 14, and the average performance in Table. 2. For 2 or 4 intervals cases, we solve the MIP with nonlinear convex constraints (10a) (13). For 8 intervals case, to speed up the computation, we outer-approximate these nonlinear convex constraints with linear constraints.

We first analyze the tightness of the relaxation. The red dots, which imply loose relaxation on the $SO(3)$, form only a thin layer between the unreachable blue dots, and the reachable green dots, implying the relaxation is pretty tight. All mixed integer convex IK solvers can detect more than 90% of global infeasibility (last row in Table 2). As a comparison, we also solve the same IK problem with the PSD relaxation on $SO(3)$ proposed in Saunderson et al. (2015); Dai et al. (2015), which is the tightest convex relaxation of $SO(3)$. Merely 42.23% of global infeasibility is detected with the PSD relaxation. This huge difference demonstrates that our mixed-integer convex relaxation of $SO(3)$ constraint is a lot tighter than the best convex relaxation.

We also examine the computation time of the mixed-integer convex IK, together with the quality of the approximated solution. In Fig 15a we show the histogram of the mixed-integer convex IK computation time for the green dots (both IK approaches can find the solution). In Fig 15c and 15d we demonstrate the accuracy of the approximated solution. After reconstructing the posture from the mixed-integer convex IK, we compute the end effector pose error from the reconstructed posture to the sampled pose, to demonstrate how much relaxation error is brought into the approximated IK solution. The orientation matches almost exactly, with less than $0.001°$ error. The end effector position is a lot less accurate. With 4 intervals per $R(i, j)$, the position error can be centimeters; and with 8 intervals, the average position error is reduced to half a centimeter.

Although it takes several seconds to obtain a solution to the mixed-integer program, the mixed-integer solver

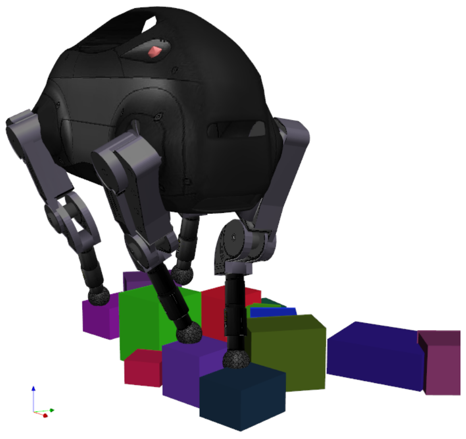
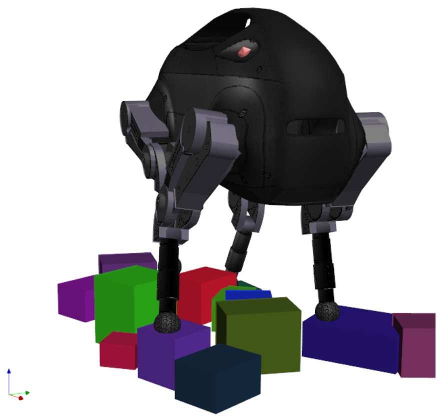



Figure 12. Postures of LittleDog standing on different sets of stepping stones. Computation time is about 15s for this 18 Dof quadruped.

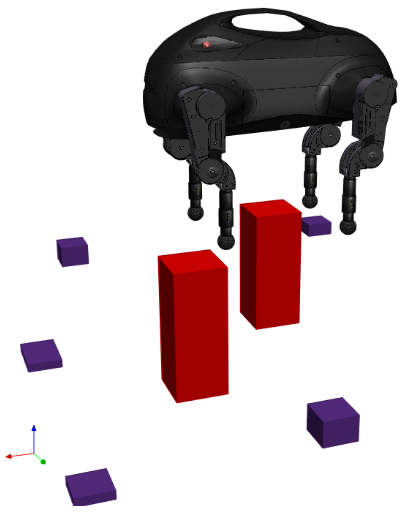


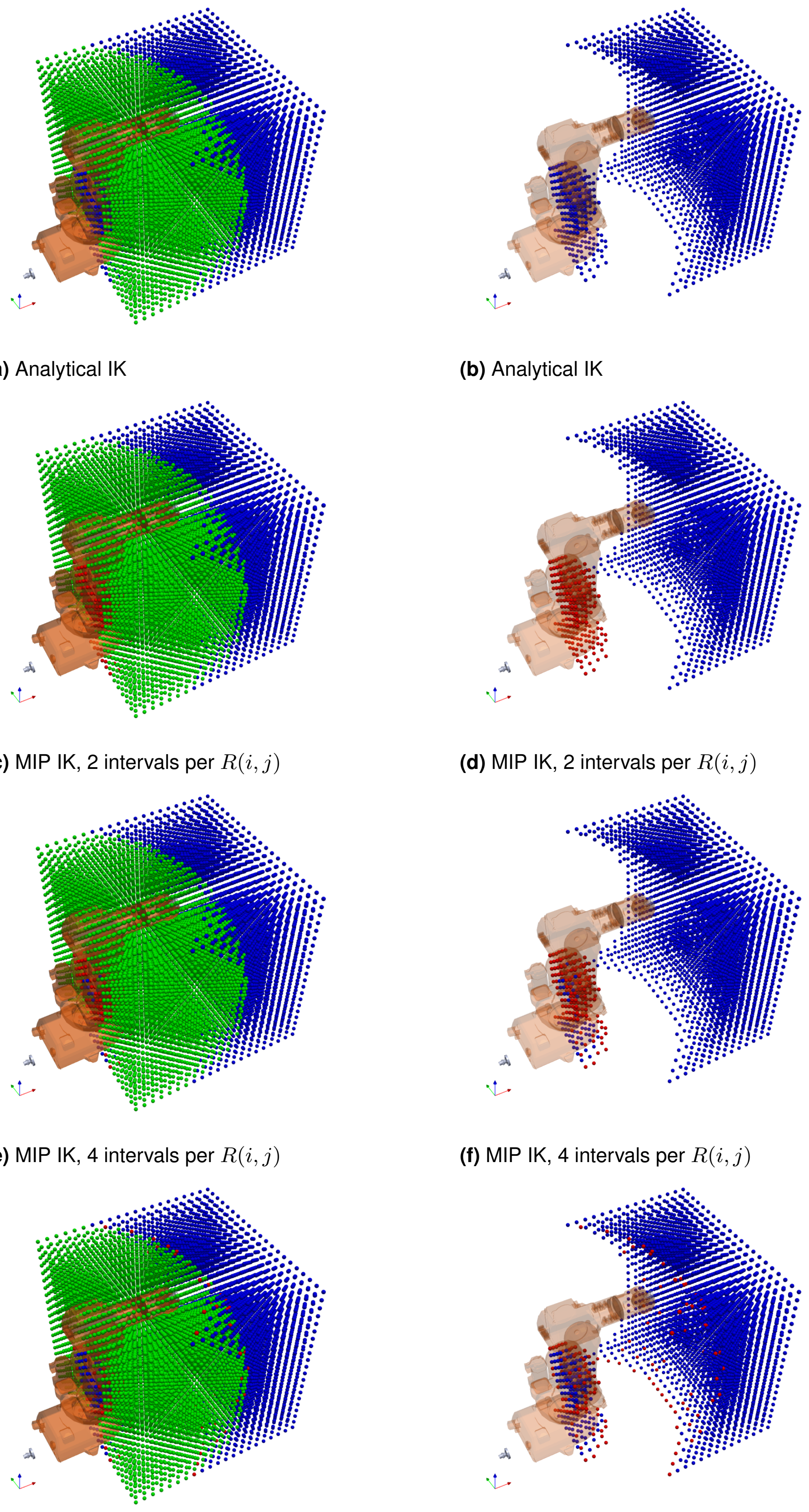
Figure 13. Mixed-Integer Convex IK proves it infeasible to put each foot on one of the stepping stone (violet), while avoiding the red obstacles.

is usually fast to prove that the IK problem is globally infeasible. In Fig 15b we draw the histograms of the computation time, when the mixed-integer convex IK proves global infeasibility. In most cases the computation takes less than 0.1s (row 4 in Table 2). This demonstrates that when the IK problem is infeasible, our mixed-integer approach can quickly detect the infeasibility and terminates early in the branch-and-bound process, as explained in Sec 2.2. We believe that the rapid detection of global infeasibility is a major advantage of our mixed-integer program IK. By adopting our method, users of high-DoF robots can avoid needing to exhaustively try out different initial guesses with the nonlinear optimization based approaches, or to fix some DoFs to certain sampled values with analytical approaches.

Moreover, we solve the inverse kinematics program for the samples in Fig 14 through a nonlinear optimization based IK solver Dai et al. (2014); Fallon et al. (2015). We first set the initial seed as the postures in Fig 14, and then solve the nonlinear optimization again, but replace the initial seed with the reconstructed posture from the mixed-integer convex IK. The success rate of the nonlinear IK solver increases from 85.67% to 98.35%, indicating the solution of the MIP IK usually lives within the region of attraction of the nonlinear optimization solver. Our experiments indicate that the approximated solution from mixed-integer programming can frequently serve as a good initial guess for the nonlinear optimization.

4.2.4 Schunk Gripper As we have seen in the previous results, when searching for a feasible solution, the MIP IK is significantly slower than its alternatives, like the nonlinear optimization based IK (Dai et al. (2014); Fallon et al. (2015); Beeson and Ames (2015)) or the analytical approach Diankov (2010). On the other hand, MIP IK is good at detecting global infeasibility, and also providing good initial guesses for the nonlinear optimization IK. Hence we think MIP IK can be best used as a complement to other IK approaches. Here we use a parallel gripper example to demonstrate how MIP IK can help other approaches.

If the goal is to grasp an object with a Schunk parallel gripper mounted on the robot arm (like the KUKA robot shown in Fig 9), then we can first test if there exists a collision free grasping posture or an approaching path with the gripper only, without considering the robot arm. For example, in Fig 16, the MIP IK detects it is infeasible for the gripper to achieve the antipodal grasp on the mug in the small cabinet, so we can save the effort to call the other IK approaches to solve the more complicated IK problem with the robot arm. On the other hand, if we enlarge the cabinet size a little bit, then the MIP IK can find an antipodal grasp, but if we ask to find an approaching path of the gripper starting from outside the cabinet to the mug, then in 2.3 seconds the MIP IK certifies non-existence of the path, as the small space in the cabinet doesn't allow the gripper to yaw its orientation without hitting the obstacles. Finding an approaching path can be formulated as a mixed-integer optimization problem. We first discretize the path to a few way points, and impose the kinematic constraints for each way points (no collision for all way points, initial way point start outside the cabinet, and the final way point grasp the mug). We also impose the continuity constraint on the path, by requiring that the change of pose is smaller than a certain threshold between two consecutive way points, formulated as convex constraints on the link poses (explained in the end of Section 3.2.2). We solve this mixed-integer problem to search for the postures at all way points simultaneously.



(a) Analytical IK

(b) Analytical IK

(c) MIP IK, 2 intervals per $R(i, j)$

(d) MIP IK, 2 intervals per $R(i, j)$

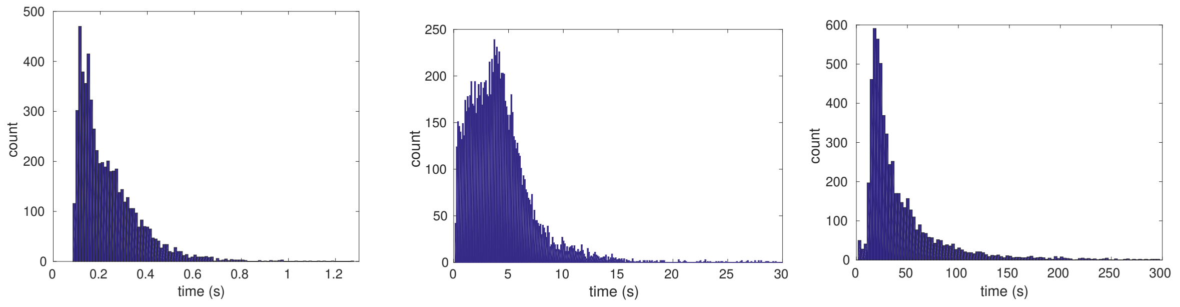
(e) MIP IK, 4 intervals per $R(i, j)$

(f) MIP IK, 4 intervals per $R(i, j)$

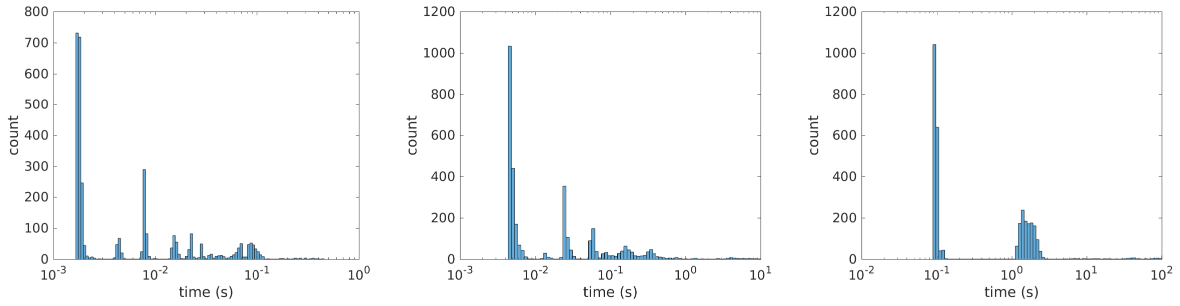
(g) MIP IK, 8 intervals per $R(i, j)$

(h) MIP IK, 8 intervals per $R(i, j)$

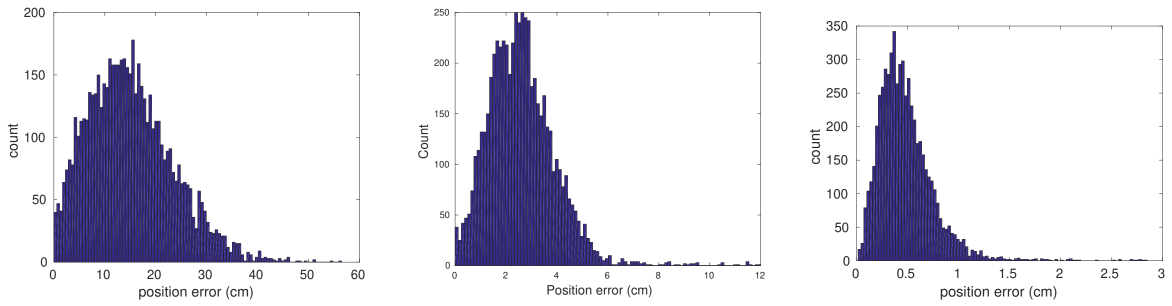
Figure 14. The results of running both analytical and mixed-integer program IK on an ABB IRB140 arm. We take $21^3 = 9321$ sample points in a $1\ m^3$ cube. The robot end effector is required to reach each sampled location with the given orientation, shown in silver color next to the coordinate axes in the bottom left. The green dots correspond to positions for which IK solvers obtain the solution. The blue dots correspond to positions for which the IK solvers detect global infeasibility. The red dots correspond to the gap for which the analytical IK proves that the problem is infeasible, while the mixed-integer convex IK thinks the problem is feasible under relaxation. We show MIP IK with different number of intervals per $R(i, j)$. For the last row with 8 intervals per $R(i, j)$, we outer-approximate all nonlinear convex constraints with linear constraints.



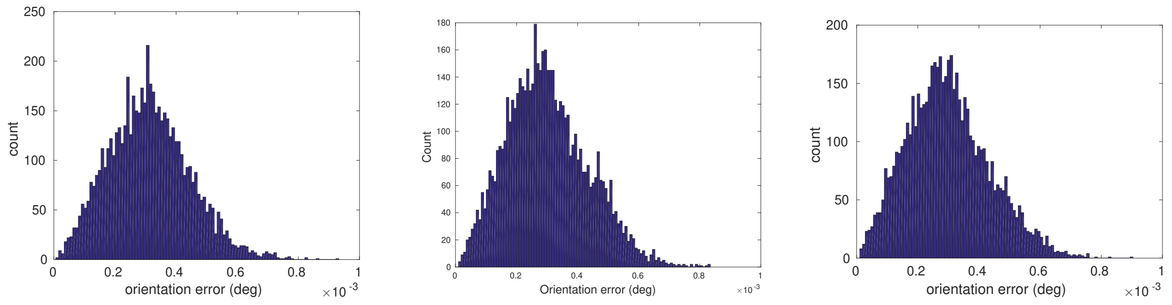
(a) Histogram on the computation time for MIP IK to find a solution on the ABB IRB140 robot (green dots in Fig 14). The number of intervals per $R(i,j)$ is 2 (left), 4 (middle) and 8 (right).



(b) Histogram on the computation time for MIP IK to detect global infeasibility on the ABB IRB140 robot (red dots in Fig 14). The number of intervals per $R(i,j)$ is 2 (left), 4 (middle) and 8 (right).



(c) Histogram on the end effector position error for the reconstructed postures from MIP IK. The number of intervals per $R(i,j)$ is 2 (left), 4 (middle) and 8 (right).



(d) Histogram on the end effector orientation error for the reconstructed postures from MIP IK. The number of intervals per $R(i,j)$ is 2 (left), 4 (middle) and 8 (right).

Figure 15. Performance of the MIP IK solver on ABB IRB140 arm for the samples in Fig 14.

# of intervals per $R(i,j)$	2	4	8
total # of binary variables	54	108	162
average time to find a solution (s)	0.2338	4.2	43.4
average time to detect global infeasibility (s)	0.0173	0.1151	1.126
average end effector orientation error (deg)	3.16×10^{-4}	3.01×10^{-4}	2.98×10^{-4}
average end effector position error (cm)	15.08	2.6	0.48
percentage of detected global infeasibility	93.19%	94.87%	95.43%

Table 2. The average performance of the mixed integer IK solver on ABB IRB140 robot, for different number of intervals per $R(i,j)$. This table summarizes the histograms in Fig 15.

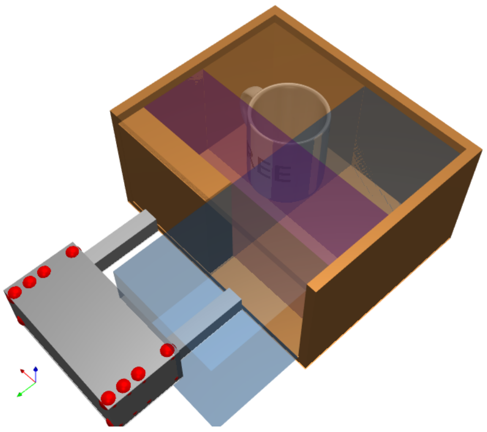


Figure 16. MIP IK detects global infeasibility for the gripper to achieve antipodal grasp on the mug within the small cabinet. The collision geometry of the gripper is represented by the red spheres, and the free space is represented by the polytopes in shaded colors. The cabinet top is covered by glass, so the gripper cannot directly grasp the mug from the top.

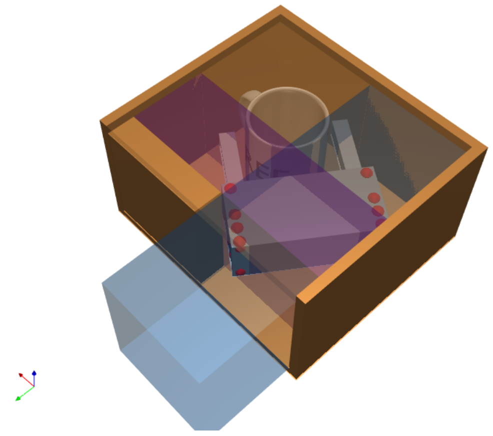


Figure 17. Enlarge the cabinet size in Fig 16 by 2 cm along the green axis direction, the MIP IK can find an antipodal grasp on the mug. On the other hand, the MIP IK detects that it is infeasible to find a path such that the mug can move from outside the cabinet, to the grasping posture.

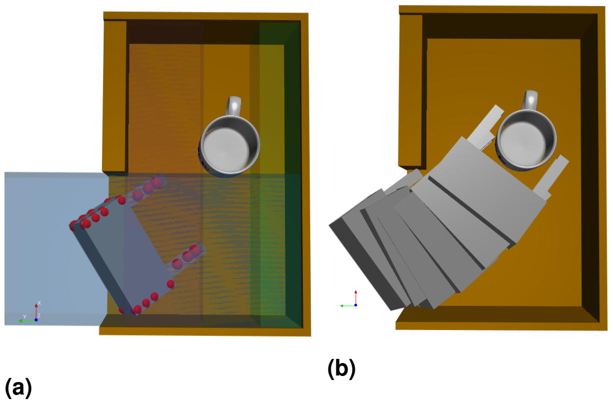


Figure 18. The approaching sequence of the gripper to the mug. The cabinet size is further enlarged from that in Fig 17. The gripper starts outside the cabinet, and ends up with an antipodal grasp on the mug. The collision geometry of the gripper (red spheres on the palm and fingers) and 3 collision free polytopes are shown in Fig 18a. All of the 6 way points are visualized in Fig 18b.

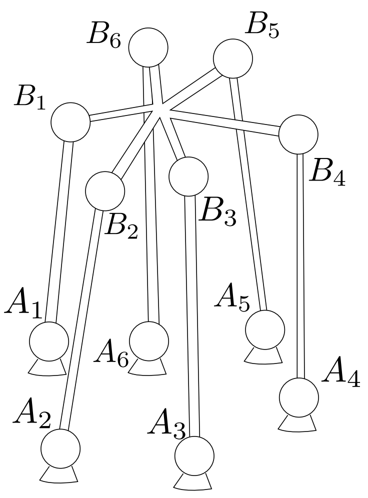


Figure 19. A general Stewart platform Porta et al. (2009).

If this mixed-integer problem is infeasible, it certifies there doesn't exist an approaching path. Finally in Fig 18 we further enlarge the cabinet size, and the MIP IK took 9 seconds to find an approaching path. Hence we show that we can use MIP IK to check the feasibility of the posture/path with the gripper only, before forwarding the problem to a nonlinear optimization based IK solver.

4.2.5 Parallel Stewart platform Finally, we show that our approach can find the solution to the forward-kinematics problem on parallel robots like the Stewart platform Stewart (1965), with constraints that the leg length are fixed. The schematic of this Stewart platform is shown in Fig 19. Here we adopt the settings in Dietmaier (1998), that the leg length $l_i = |A_iB_i|$ is fixed, and position ${}^B\mathbf{p}_{A_i}$ of A_i in base B and position ${}^P\mathbf{p}_{B_i}$ of point B_i in the platform P are given in Table 3. It is known that this platform has 40 possible poses.

Since the constraint

$$|A_iB_i|^2 = |{}^B\mathbf{p}_P + {}^BR_P\,{}^P\mathbf{p}_{B_i} - {}^B\mathbf{p}_{A_i}|^2 = l_i^2 \qquad (21)$$

is a bilinear function of the platform position ${}^B\mathbf{p}_P$ and orientation ${}^BR_P$, we can again approximate constraint (21) as mixed-integer convex constraint, through McCormick envelope technique introduced in Section 2.1, and find all the solutions using MIP. In Fig 20, we show our approximated solution together with the analytical solution found in Dietmaier (1998). Our MIP solver can find all approximated solutions, as each of the 40 analytical solutions is accompanied by an approximated solution found through MIP. The converse is not true, that the approximated solution is not always accompanied by an analytical solution, there are some approximated solutions (blue dots in Fig 20) that have no analytical solutions (red dots) in their neighbourhood, since the approximated solution only satisfies the relaxed constraints (for example, $x = 0$ approximately satisfies the constraint $x^2 + 10^{-6} = 0$, but there is no actual solution in the neighbourhood). We also draw the histogram on the constraint violation $|l_i - |A_iB_i||$ for the approximated solutions in Fig 21. All the violation is

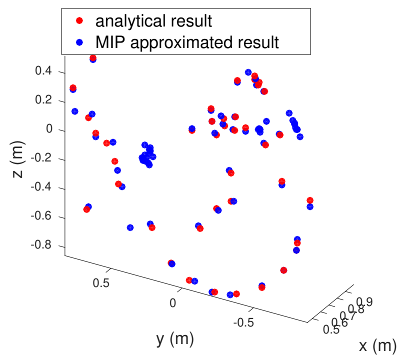


Figure 20. The solution to the Stewart platform for the platform origin position ${}^{B}\mathbf{p}_{P}$.

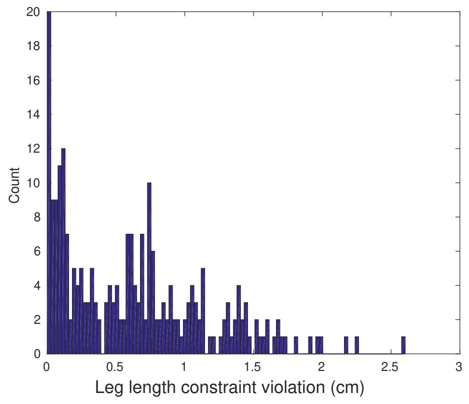


Figure 21. Histogram on leg length constraint violation for the Stewart platform.

less than 3 cm. The computation time for finding all solutions is 340s.

5 Conclusion and Discussion

In this paper we propose a mixed-integer convex relaxation of the non-convex $SO(3)$ constraint, and formulate a mixed-integer convex optimization program to solve the inverse kinematics problem globally with generic constraints. We show that this relaxation is relatively tight, and our IK approach can either produce an approximate solution, or prove that the solution does not exist globally. We demonstrate results when applying our approach on manipulators, a quadruped robot and a parallel Stewart platform.

In the future, we would like to apply our mixed-integer optimization approach to trajectory optimization problems, to combine the kinematics constraints in this paper, together with dynamic constraints. Some trajectory optimization problems, such as dexterous manipulation or walking gait optimization, involves frequent change of contact. The contact state (active/inactive contact) can naturally be represented though binary variables, and thus can be incorporated into a mixed-integer optimization formulation.

6 Acknowledgement

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374. We would also like to thank our colleagues at Toyota Research Institute and MIT Robot Locomotion Group for the valuable discussions.

References

Aceituno-Cabezas B, Mastalli C, Dai H, Focchi M, Radulescu A, Caldwell DG, Cappelletto J, Grieco JC, Fernández-López G and Semini C (2018) Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters* 3(3): 2531–2538.

Achterberg T and Wunderling R (2013) Mixed integer programming: Analyzing 12 years of progress. In: *Facets of combinatorial optimization*. Springer, pp. 449–481.

Bates DJ, Hauenstein JD, Sommese AJ and Wampler CW (2013) *Numerically solving polynomial systems with Bertini*, volume 25. SIAM.

Beale EML and Tomlin JA (1970) Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR* 69(447-454): 99.

Beeson P and Ames B (2015) Trac-ik: An open-source library for improved solving of generic inverse kinematics. In: *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, pp. 928–935.

Berenson D, Srinivasa S and Kuffner J (2011) Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research* 30(12): 1435–1460.

Bertsekas DP (1999) *Nonlinear programming*. Athena scientific Belmont.

Bertsimas D and Tsitsiklis JN (1997) *Introduction to linear optimization*.

Bertsimas D and Weismantel R (2005) *Optimization over integers*, volume 13. Dynamic Ideas Belmont.

Blackmore L, Ono M, Bektassov A and Williams BC (2010) A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics* 26(3): 502–517.

Boyd S and Vandenberghe L (2004) *Convex optimization*. Cambridge university press.

Buss SR (2004) Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation* 17(1-19): 16.

Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, Xiao J and Yu F (2015) Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* .

CPLEX I (2016) 12.7. 0 users manual.

Craig JJ (2005) *Introduction to robotics: mechanics and control*, volume 3.

Dai H, Izatt G and Tedrake R (2017) Global inverse kinematics via mixed-integer convex optimization. In: *International Symposium on Robotics Research*.

Dai H, Majumdar A and Tedrake R (2015) Synthesis and optimization of force closure grasps via sequential semidefinite programming. In: *International Symposium on Robotics Research*.

Dai H, Valenzuela A and Tedrake R (2014) Whole-body motion planning with centroidal dynamics and full kinematics.

i	${}^{B}\mathbf{p}_{A_i}$	${}^{P}\mathbf{p}_{B_i}$	l_i
1	$(0, 0, 0)$	$(0, 0, 0)$	$l_1 = 1$
2	$(1.107915, 0, 0)$	$(0.542805, 0, 0)$	$l_2 = 0.645275$
3	$(0.549094, 0.756063, 0)$	$(0.956919, 0.528915, 0)$	$l_3 = 1.086284$
4	$(0.735077, 0.223935, 0.525991)$	$(0.665885, 0.353482, 1.402538)$	$l_4 = 1.503439$
5	$(0.514188, 0.526063, 0.368418)$	$(0.478359, 1.158742, 0.107672)$	$l_5 = 1.281933$
6	$(0.590473, 0.094733, 0.205018)$	$(0.137087, 0.235121, 0.353913)$	$l_6 = 0.771071$

Table 3. Geometric parameters of the Stewart platform Dietmaier (1998)

In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, pp. 295–302.

Deits R and Tedrake R (2014) Footstep planning on uneven terrain with mixed-integer convex optimization. In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE.

Deits R and Tedrake R (2015a) Computing large convex regions of obstacle-free space through semidefinite programming. In: *Algorithmic foundations of robotics XI*. Springer, pp. 109–124.

Deits R and Tedrake R (2015b) Efficient mixed-integer planning for uavs in cluttered environments. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE.

Diankov R (2010) *Automated construction of robotic manipulation programs*. PhD Thesis, Carnegie Mellon University.

Dietmaier P (1998) The stewart-gough platform of general geometry can have 40 real postures. In: *Advances in Robot Kinematics: Analysis and Control*. Springer, pp. 7–16.

Ding Y, Li C and Park HW (2018) Single leg dynamic motion planning with mixed-integer convex optimization. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–6.

Fallon M, Kuindersma S, Karumanchi S, Antone M, Schneider T, Dai H, D'Arpino CP, Deits R, DiCicco M, Fourie D, Koolen T, Marion P, Posa M, Valenzuela A, Yu KT, Shah J, Iagnemma K, Tedrake R and Teller S (2015) An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics* 32(2): 229–254.

Forrest J, Ralphs T, Vigerske S, LouHafer, Kristjansson B, jpfasano, EdwinStraver, Lubin M, Santos HG, rlougee and Saltzman M (2018) coin-or/cbc: Version 2.9.9. DOI:10.5281/zenodo.1317566. URL `https://doi.org/10.5281/zenodo.1317566`.

Gurobi Optimization I (2016) Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Haralick RM, Joo H, Lee CN, Zhuang X, Vaidya VG and Kim MB (1989) Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics* 19(6): 1426–1446.

Hauser K, Wang S and Cutkosky M (2017) Efficient equilibrium testing under adhesion and anisotropy using empirical contact force models. In: *In proceedings of Robotics: Science and Systems (RSS)*.

Huchette J and Vielma JP (2016) Small independent branching formulations for unions of v-polyhedra. *arXiv preprint arXiv:1607.04803* .

Husty ML, Pfurner M and Schröcker HP (2007) A new and efficient algorithm for the inverse kinematics of a general serial 6r manipulator. *Mechanism and machine theory* 42(1): 66–81.

Izatt G, Dai H and Tedrake R (2017) Globally optimal object pose estimation in point clouds with mixed-integer programming. In: *International Symposium on Robotics Research*.

Landry B, Deits R, Florence PR and Tedrake R (2016) Aggressive quadrotor flight through cluttered environments using mixed integer programming. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 1469–1475.

Lawler EL and Wood DE (1966) Branch-and-bound methods: A survey. *Operations research* 14(4): 699–719.

Li TY (2003) Solving polynomial systems by the homotopy continuation method. *Handbook of numerical analysis* 11: 209–304.

Manocha D and Canny JF (1992) Real time inverse kinematics for general 6r manipulators. In: *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, pp. 383–389.

Marchand H, Martin A, Weismantel R and Wolsey L (2002) Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics* 123(1-3): 397–446.

Mason MT (2001) *Mechanics of robotic manipulation*. MIT press.

McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. *Mathematical programming* 10(1): 147–175.

Mellinger D, Kushleyev A and Kumar V (2012) Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, pp. 477–483.

Misener R and Floudas CA (2012) Global optimization of mixed-integer quadratically-constrained quadratic programs (miqcqp) through piecewise-linear and edge-concave relaxations. *Mathematical Programming* : 1–28.

Mosek A (2010) The mosek optimization software. *Online at http://www. mosek. com* 54(2-1): 5.

Murray RM, Li Z, Sastry SS and Sastry SS (1994) *A mathematical introduction to robotic manipulation*. CRC press.

Pang T and Tedrake R (2018) A robust time-stepping scheme for quasistatic rigid multibody systems .

Peiper DL (1968) The kinematics of manipulators under computer control. Technical report.

Porta JM, Ros L and Thomas F (2009) A linear relaxation technique for the position analysis of multiloop linkages. *IEEE Transactions on Robotics* 25(2): 225–239.

Porta JM, Ros L, Thomas F and Torras C (2005) A branch-and-prune solver for distance constraints. *IEEE Transactions on Robotics* 21(2): 176–187.

Qiao S, Liao Q, Wei S and Su HJ (2010) Inverse kinematic analysis of the general 6r serial manipulators based on double quaternions. *Mechanism and Machine Theory* 45(2): 193–199.

Raghavan M and Roth B (1990) Kinematic analysis of the 6r manipulator of general geometry. In: *International symposium on robotics research*. pp. 314–320.

Saunderson J, Parrilo PA and Willsky AS (2015) Semidefinite descriptions of the convex hull of rotation matrices. *SIAM Journal on Optimization* 25(3): 1314–1343.

Schrijver A (1998) *Theory of linear and integer programming*. John Wiley & Sons.

Singh A, Ghabcheloo R, Muller A and Pandya H (2018) Combining method of alternating projections and augmented lagrangian for task constrained trajectory optimization.

Stewart D (1965) A platform with six degrees of freedom. *Proceedings of the institution of mechanical engineers* 180(1): 371–386.

Tedrake R and the Drake Development Team (2016) Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems. URL http://drake.mit.edu.

Valenzuela AK (2016) *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain*. PhD Thesis, Massachusetts Institute of Technology.

Varedi S, Daniali H and Ganji D (2009) Kinematics of an offset 3-upu translational parallel manipulator by the homotopy continuation method. *Nonlinear Analysis: Real World Applications* 10(3): 1767–1774.

Vielma JP and Nemhauser GL (2011) Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming* 128(1): 49–72.

Williams HP (2013) *Model building in mathematical programming*. John Wiley & Sons.

A Appendix

A.1 Tighter relaxation on $SO(3)$ constraint

In Section 3.1, we relax the $SO(3)$ constraint, by cutting the range $[-1, 1]$ into smaller intervals, and approximate the $SO(3)$ constraint (1) by replacing the bilinear products and quadratic terms with slack variables in their convex hulls within each interval. In this subsection, we introduce an alternative approach to relax the $SO(3)$ constraint as mixed-integer convex constraints. This approach is more geometric, and it has been used in Izatt et al. (2017). The constraints formulated in this section can be imposed in addition to the constraints in Section 3.1, to give a tighter approximation on $SO(3)$ constraint.

Geometrically, for each 3D column vector $\mathbf{u}_i \in \mathbb{R}^3$ in the rotation matrix R, the unit length constraint $\mathbf{u}_i^T\,\mathbf{u}_i = 1$ ((1a)) means $\mathbf{u}_i$ is on the surface of a 3D unit sphere. By cutting the range of each entry $R(i, j)$, namely $[-1, 1]$, into smaller intervals $[\phi_k, \phi_{k+1}]$ as we did in 3.1, the 3D space that $\mathbf{u}_i$ lives in is segmented to small boxes, by planes perpendicular to either x, y or z axes. In Fig. 22a, we show the sphere together with the boxes in the first orthant ($x, y, z \geq 0$). The intersection region between each box and the surface of the sphere is depicted in Fig. 22b.

Although each intersection region is non-convex, we can readily formulate its convex hull as $\{\mathbf{u}_i | \mathbf{u}_i^T\,\mathbf{u}_i \leq 1, A\mathbf{u}_i \leq \mathbf{b}\}$. Namely the boundary of the convex hull includes the sphere surface, and some planes inside the sphere. As an example, the convex hull of the shaded region in Fig. 22b can be written as

$$\mathbf{u}_i^T\,\mathbf{u}_i \leq 1 \tag{22a}$$
$$\mathbf{n}_1^T\,\mathbf{u}_i \geq \mathbf{n}_1(2) \tag{22b}$$
$$\mathbf{n}_2^T\,\mathbf{u}_i \geq \mathbf{n}_2(2) \tag{22c}$$
$$0 \leq \mathbf{u}_i(1) \leq 0.5 \tag{22d}$$
$$0 \leq \mathbf{u}_i(3) \leq 0.5 \tag{22e}$$

where $\mathbf{n}_1, \mathbf{n}_2$ are the normal vectors of the planes, shown in Fig. 22c. Since both planes pass the point $[0, 1, 0]^T$, the right-hand side of (22b)(22c) are the inner-product $[0, 1, 0]\mathbf{n}_1$, $[0, 1, 0]\mathbf{n}_2$ respectively.

For each column vector $\mathbf{u}_i$, to approximate the unit length constraint $\mathbf{u}_i^T\,\mathbf{u}_i = 1$, we will always impose constraint (22a) to bound the vector to be within the unit sphere. In order to push the vector to the boundary of the sphere, depending on which box the vector is in, we will activate the linear constraints, such as (22b)-(22e) using big-M tricks in mixed-integer optimization Bertsimas and Weismantel (2005). The binary variables introduced in the sos2 constraint (3) can be used to determine the activated box. The big-M trick is a common technique to activate constraint based on binary variables. For example, to active linear constraint $\mathbf{a}^T\,\mathbf{x} \leq b$ when binary variable $z = 1$, we can impose the constraint $\mathbf{a}^T\,\mathbf{x} \leq b + M(1 - z)$ where M is a big positive number.

In order to relax the orthogonality constraint $\mathbf{u}_i^T\,\mathbf{u}_j = 0$, we consider to introduce a given vector $\mathbf{v}$ that approximates $\mathbf{u}_i$ within one of the intersection region, and then impose the orthogonality constraint as $\mathbf{v}^T\,\mathbf{u}_j \approx 0$ instead. The geometric intuition is illustrated in Fig 23 in a 2D plot. When the vector $\mathbf{u}_i$ is within on of the intersection region in Fig.22b, we can find the "center" of that intersection region as $\mathbf{v}$, also with the bound that $\mathbf{v}^T\,\mathbf{u}_i \geq \cos\theta$, namely $\angle(\mathbf{v}, \mathbf{u}_i) \leq \theta$ for all $\mathbf{u}_i$ within the intersection region. (The procedure to find such $\mathbf{v}$ and θ will be explained in detail in Section A.2.) The orthogonality constraint $\angle(\mathbf{u}_i, \mathbf{u}_j) = 90°$ implies $|\angle(\mathbf{v}, \mathbf{u}_j) - 90°| \leq \theta$, namely $\mathbf{v}^T\,\mathbf{u}_j \leq \sin\theta$. Algebraically, when the binary variables indicate that $\mathbf{u}_i$ is in an intersection region, we can impose the linear constraint

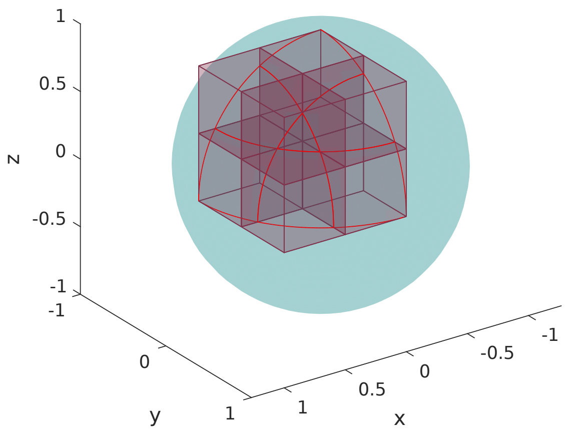
$$\mathbf{v}^T\,\mathbf{u}_j \leq \sin\theta \tag{23}$$

as a relaxation of the orthogonality constraint $\mathbf{u}_i^T\,\mathbf{u}_j = 0$.

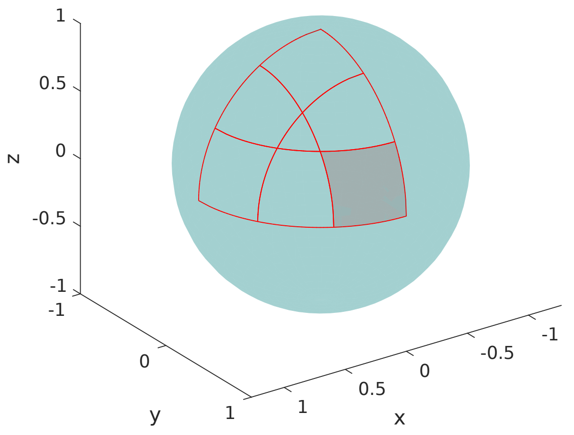
In order to relax the constraint $\mathbf{u}_k - \mathbf{u}_i \times \mathbf{u}_j = \mathbf{0}$, we again consider to replace $\mathbf{u}_i$ by the vector $\mathbf{v}$ in the "center" of the intersection region in which $\mathbf{u}_i$ is, we thus obtain the following constraint

$$|\mathbf{u}_k - \mathbf{v} \times \mathbf{u}_j|_2^2 \tag{24a}$$
$$= |\mathbf{u}_k|_2^2 + |\mathbf{v} \times \mathbf{u}_j|_2^2 - 2\mathbf{u}_k^T\,(\mathbf{v} \times \mathbf{u}_j) \tag{24b}$$
$$= 1 + |\mathbf{v} \times \mathbf{u}_j|_2^2 - 2\mathbf{v}^T\,(\mathbf{u}_j \times \mathbf{u}_k) \tag{24c}$$
$$= 1 + |\mathbf{v} \times \mathbf{u}_j|_2^2 - 2\mathbf{v}^T\,\mathbf{u}_i \tag{24d}$$
$$\leq 1 + |\mathbf{v} \times \mathbf{u}_j|_2^2 - 2\cos\theta \tag{24e}$$
$$\leq 1 + 1 - 2\cos\theta \tag{24f}$$

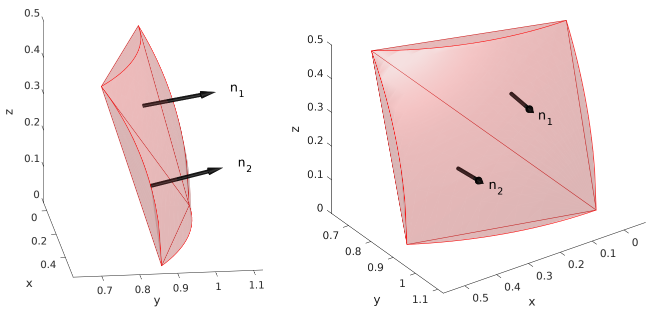
(24c) holds because $\mathbf{u}_k$ should be a unit length vector, and the property of cross product $\mathbf{a}^T\,(\mathbf{b} \times \mathbf{c}) = \mathbf{b}^T\,(\mathbf{c} \times \mathbf{a}) \forall \mathbf{a}, \mathbf{b}, \mathbf{c}$. (24d) holds because $\mathbf{u}_j \times \mathbf{u}_k = \mathbf{u}_i$. The inequality (24e) is obtained from the assumption $\mathbf{v}^T\,\mathbf{u}_i \geq \cos\theta$ when we approximate $\mathbf{u}_i$ with the "center" vector $\mathbf{v}$.



(a) The unit-length sphere, together with the boxes obtained by cutting 3D space at $0, 0.5, 1$ on each axis. Each plane intersects with the surface of the sphere along the red curves. The first orthant is cut into 8 boxes.



(b) The intersection region between each box and the surface of the sphere in the first orthant. The boundaries of the regions are the red curves in Fig. 22a. Note that although there are 8 boxes in Fig.22a, the box $0 \leq x, y, z \leq 0.5$ does not intersect with the surface of the sphere, thus there are only 7 regions on the sphere surface. We highlight the bottom right region in shaded color as an example for analysis. This region is the intersection of the sphere surface with the box $0 \leq x \leq 0.5, 0.5 \leq y \leq 1, 0 \leq z \leq 0.5$.



(c) Fig.22c are the convex hull of the shaded region in Fig.22b from two perspectives. $\mathbf{n}_1, \mathbf{n}_2$ are the normal vectors of the planes, as the boundary of the convex hull region.

Lastly (24f) is valid because $|\mathbf{v} \times \mathbf{u}_j|_2 \leq 1$ as both $\mathbf{v}, \mathbf{u}_j$ are unit length vectors. As a result, we can impose (24) as a convex quadratic constraint on the decision variables $\mathbf{u}_k, \mathbf{u}_j$ to approximate the constraint $\mathbf{u}_k = \mathbf{u}_i \times \mathbf{u}_j$ in $SO(3)$ constraint.

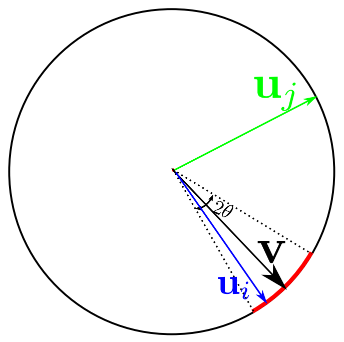


Figure 23. If the unit length vector $\mathbf{u}_i$ is within a region (the red arc) on the sphere surface, we can find the "center" of that region as $\mathbf{v}$ with $\angle(\mathbf{u}_i, \mathbf{v}) \leq \theta$. The orthogonal constraint that $\angle(\mathbf{u}_i, \mathbf{u}_j) = 90°$ implies that $|\angle(\mathbf{v}, \mathbf{u}_j) - 90°| \leq \theta$.

To conclude, we can divide the range $[-1, 1]$ into small intervals, and then consider the approximation of $SO(3)$ constraint within each interval. To do so, we segment the unit sphere surface into small regions, depicted in Fig. 22b, and then impose constraint (22)(23)(24) as a relaxation of $SO(3)$ constraint. By imposing these constraints in addition to the relaxation in Section 3.1, we obtain a tighter relaxation on $SO(3)$ constraint, as can be seen in Table 4. $\min_{\bar{R}} |\bar{R}\mathbf{e}_i + \bar{R}\mathbf{e}_j|_2$ is slightly larger with the additional constraints formulated in this section.

A.2 Computing "center" vector on the intersection region

We aim to compute the "center vector" $\mathbf{v}$ in the middle of each region. By "center vector", we mean $\mathbf{v}$ minimizes the largest angle $\angle(\mathbf{v}, \mathbf{u}_i)$ for all vector $\mathbf{u}_i$ in the intersection region. To this end, we denote the intersection region as $\mathcal{S}$, and solves the following optimization problem to find the center vector $\mathbf{v}$.

$$\min_{\mathbf{v}\in\mathcal{S}} \max_{\mathbf{u}_i\in\mathcal{S}} \angle(\mathbf{v}, \mathbf{u}_i) \tag{25a}$$

$$= \min_{\mathbf{v}\in\mathcal{S}} \max_{\mathbf{u}_i\in\mathcal{S}} \arccos(\mathbf{v}^T\mathbf{u}_i) \tag{25b}$$

From (25a) to (25b), we use the fact that since both $\mathbf{v}, \mathbf{u}_i$ have unit length (as they are both on $\mathcal{S}$), thus $\angle(\mathbf{v}, \mathbf{u}_i) = \arccos(\mathbf{v}^T\mathbf{u}_i)$. The optimal objective function in (25) is the angle θ, satisfying $\angle(\mathbf{v}, \mathbf{u}_i) \leq \theta\ \forall \mathbf{u}_i \in \mathcal{S}$, namely θ is the maximal angle between the "center vector" $\mathbf{v}$ and any vector $\mathbf{u}_i \in \mathcal{S}$.

Since arccos is monotonically decreasing, (25b) is equivalent to the following a max-min problem without arccos

$$\max_{\mathbf{v}\in\mathcal{S}} \min_{\mathbf{u}_i\in\mathcal{S}} \mathbf{v}^T\mathbf{u}_i \tag{26}$$

(26) and (25) has the same optimal $\mathbf{v}, \mathbf{u}_i$, and the optimal objective of (26) is $\cos\theta$.

In order to solve this max-min problem (26), we aim to first convert (26) to a maximization problem, by obtaining the optimal solution to the inner minimization problem $\min_{\mathbf{u}_i\in\mathcal{S}} \mathbf{v}^T\mathbf{u}_i$ for a fixed vector $\mathbf{v}$. After the inner minimization problem is solved for a fixed $\mathbf{v}$, we can then

# of intervals	2	4	6	∞
With constraints in Section 3.1	0.60229	1.22474	1.32667	1.414
With constraints in both Section 3.1 and A.1	0.60302	1.25649	1.33283	1.414

Table 4. $\min_{\bar{R}} \left|\bar{R}\mathbf{e}_i + \bar{R}\mathbf{e}_j\right|_2$ as the tightness measure of the mixed-integer convex relaxation on $SO(3)$.

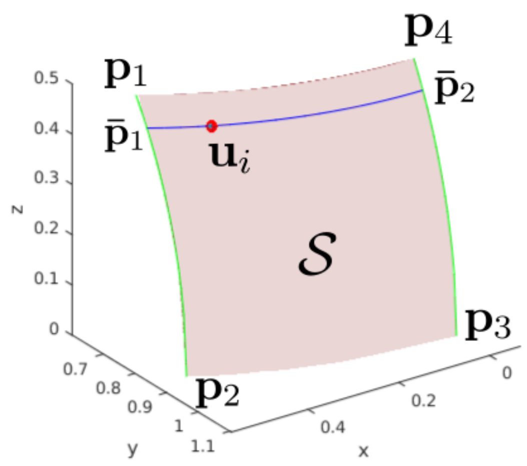


Figure 24

maximize over $\mathbf{v} \in \mathcal{S}$. In the following discussion we will show that when $\mathbf{v}$ is fixed, the optimal $\mathbf{u}_i$ to the inner minimization problem $\min_{\mathbf{u}_i} \mathbf{v}^T \mathbf{u}_i$ is always obtained at one of the corners of $\mathcal{S}$.

Lemma The minimal of

$$\min_{\mathbf{u}_i \in \mathcal{S}} \mathbf{v}^T \mathbf{u}_i \tag{27}$$

is obtained at one of the corners $\mathbf{p}_j$ of the intersection region $\mathcal{S}$, for a given vector $\mathbf{v} \in \mathcal{S}$.

Proof. Without loss of generality, we prove the lemma for the highlighted intersection region in Fig 22b, namely $\{\mathbf{u}_i | \mathbf{u}_i^T \mathbf{u}_i = 1, 0 \le \mathbf{u}_i(1) \le 0.5, 0.5 \le \mathbf{u}_i(2) \le 1, 0 \le \mathbf{u}_i(3) \le 0.5\}$, and show that $\mathbf{v}^T \mathbf{u}_i \ge \min_j \mathbf{v}^T \mathbf{p}_j$ where $\mathbf{v}$ is a given point on the surface patch, $\mathbf{p}_j$ is the corner of the intersection region, shown in Fig 24.

To do so, we first draw an arc (the blue arc) crossing $\mathbf{u}_i$, as the intersection between the plane perpendicular to z axis, and the surface region. We will show that the minimal of $\mathbf{v}^T \mathbf{w}$ over $\mathbf{w}$ along the arc occurs at one of the two ends $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2$. $\min_{\mathbf{w} \text{ on the blue arc}} \mathbf{v}^T \mathbf{w}$ is equivalent to $\min_\gamma \mathbf{v}(1)\cos\gamma + \mathbf{v}(2)\sin\gamma$, where $\gamma = \text{atan2}(\mathbf{w}(2), \mathbf{w}(1))$. The range of γ is within $[0°, 90°]$. The minimal of this concave function occurs at its boundary, namely either $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2$. Likewise, we can consider the two green arcs in Fig 24, and the minimal of $\min_{\mathbf{w}} \mathbf{v}^T \mathbf{w}$ on the green arc also occurs at one of the two ends. Thus we proved that $\mathbf{v}^T \mathbf{u}_i \ge \min_j \mathbf{v}^T \bar{\mathbf{p}}_j \ge \min_j \mathbf{v}^T \mathbf{p}_j$.

On the other hand, $\mathbf{p}_j \in \mathcal{S}$, so by taking $\mathbf{u}_i = \mathbf{p}_j$ we obtain the minimal $\mathbf{v}^T \mathbf{u}_i$. □

Using the lemma above, we can replace the inner minimization $\min_{\mathbf{u}_i \in \mathcal{S}} \mathbf{v}^T \mathbf{u}_i$ in (26) with $\min_j \mathbf{v}^T \mathbf{p}_j$, and (26) is equivalent to $\max_{\mathbf{v} \in \mathcal{S}} \min_j \mathbf{v}^T \mathbf{p}_j$, which could be reformulated as the following maximization problem with the slack variable t, which represents $\min_j \mathbf{v}^T \mathbf{p}_j$.

$$\max_{\mathbf{v},t} t \tag{28a}$$
$$\text{s.t } \mathbf{v}^T \mathbf{p}_j \ge t \ \forall j \tag{28b}$$
$$\mathbf{v} \in \mathcal{S} \tag{28c}$$

The optimization problem (28) is equivalent to (26). But now instead of solving a max-min problem as in (26), we only need to solve a maximization problem (28), which is a lot easier. (28) is almost a convex optimization problem, except constraint (28c), as $\mathcal{S}$ is not a convex set. On the other hand, $\mathcal{S}$ is on the boundary of the convex sphere $\mathbf{v}^T \mathbf{v} \le 1$, and it is easy to see that if we replace the constraint $\mathbf{v} \in \mathcal{S}$ with $\mathbf{v}$ on the surface of the sphere $\mathbf{v}^T \mathbf{v} = 1$, the optimal solution is unchanged, as the "center vector" $\mathbf{v}$ would still be on $\mathcal{S}$, even if we relax the range of $\mathbf{v}$ to the whole sphere surface. Moreover, we know that the optimal value of (28) occurs when $\mathbf{v}$ has the longest length, so we can replace the non-convex constraint $\mathbf{v}^T \mathbf{v} = 1$ with the convex constraint $\mathbf{v}^T \mathbf{v} \le 1$, and expect the same optimal solution. Namely we solve the following convex (second order conic) problem which is equivalent to (28)

$$\max_{\mathbf{v},t} t \tag{29a}$$
$$\text{s.t } \mathbf{v}^T \mathbf{p}_j \ge t \ \forall j \tag{29b}$$
$$\mathbf{v}^T \mathbf{v} \le 1 \tag{29c}$$

The optimal $\mathbf{v}$ to (29) is the "center vector", and the optimal objective to (29) is $\cos\theta$.

As a result, we can find the "center" vector $\mathbf{v}$ in the intersection region $\mathcal{S}$, together with the maximal angle difference θ between $\mathbf{v}$ and $\mathbf{u}_i$ in the same region, by solving the convex optimization problem (29) above. This optimization problem (29) is solved before solving the mixed-integer problem in Section A.1, to set up constraints (23) (24) in the mixed-integer problem.