Semi-Infinite Programming with Complementarity Constraints for Pose Optimization with Pervasive Contact

Mengchao Zhang[1] and Kris Hauser[2]

***Abstract*— This paper presents a novel computational model to address the problem that contact is an infinite phenomena involving continuous regions of interaction. The problem is cast as a semi-infinite program with complementarity constraints (SIPCC). Rather than pre-discretize contacting surfaces into a finite number of contact points, we use semi-infinite programming (SIP) techniques that operate on the underlying continuous geometry, but dynamically determine a finite number of constraints that are most relevant to solving the problem. Then we solve the series of problems whose solutions converge toward one that contains a true optimum of the original SIPCC. We apply the model to a grasping pose optimization problem for a gripper and a humanoid robot, and our model enables the robots to find a feasible pose to hold (non-)convex objects while ensuring force and torque balance.**

I. Introduction

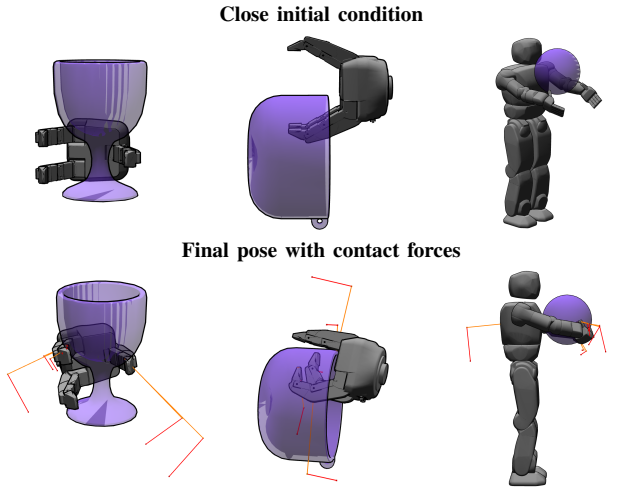
Contact is pervasive in nature. Humans and other organisms treat contact as a fact of life and utilize contact with their environment to enable dexterous manipulation of objects or agile locomotion. In contrast, the majority of current robots avoid touching things as much as possible, and when they do, they tend to avoid complex manipulations like pushing, sliding, and hugging objects and have difficulties sitting, crawling, and leaning on supports [7], [8]. Optimization is a central tool in robot planning, but a major limitation with existing tools is the representation of geometric contact in optimization, which is not easily captured as a numeric-valued constraint except for simple geometries like polyhedra and geometric primitives [21], [22]. In this paper, we propose a novel method for handling complex, irregular geometries to optimize poses of objects in contact while respecting force and torque balance conditions.

Our method is based on the semi-infinite programming (SIP) paradigm, which represents a collision constraint as an infinite set of simpler constraints and dynamically instantiates a finite subset of these constraints during optimization. We present a novel formulation *semi-infinite programming with complementarity constraints* (SIPCC) that optimizes a force distribution over points of contact, and encodes constraints and/or objectives on the force distribution, e.g., force balance and Coulomb friction. The complementarity condition, which states that the force at the point may only be nonzero if contact is made at that point, makes this tractable, since the solver only needs to reason about the force distribution at points of active contact.

* This work was supported by NSF Grant IIS #-1911087.

[1]Mengchao Zhang is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign. mz17@illinois.edu

[2]Kris Hauser is with the Department of Computer Science, University of Illinois at Urbana-Champaign. kkhauser@illinois.edu



(a) Gripper, Wine glass (b) Gripper, Bowl (c) Humanoid, Sphere

Fig. 1: Our algorithm optimizes for stability considering multiple contacts anywhere on the robot. The first row shows the initial configurations for three examples, and the bottom row shows final poses and contact forces. Normal forces are drawn in orange, and friction forces are drawn in red, translated to the end of the normal forces. [Best viewed in color.]

We present a local optimization algorithm for SIPCC that uses the exchange method [12], which iteratively chooses a finite subset of contact points and forces to consider, and then formulates a finite-dimensional nonlinear constrained optimization problem to solve for a step direction. Using a judicious constraint selection procedure, called an oracle, the series of problems converges toward one that contains a true optimum of the original problem. We discuss the impact of several implementation design choices, such as the oracle, merit function, inner-loop optimization step size, and outer-loop line-search step-shrinking coefficient.

The model is implemented on a gripper and a humanoid robot to optimize their static pose to hold an object and ensure that the object is in both force and torque balance. The algorithm converges quickly to local optima even with highly complex geometries, and poses can be found in tens of seconds.

II. Related work

In this section, we review related works in the robotics field using SIP and works on pose optimization with contacts.

A. Semi-infinite programming

Historically, SIP appeared as a smooth reformulation of the non-smooth Chebyshev approximation problem [27]. Further applications of SIP arose in different fields such as engineering design, optimal control, disjunctive optimization and etc.

In the field of robotics, SIP has been applied to robot trajectory optimization problems under state and control constraints, in which the 1D time variable of the spline is the index parameter [26]. It has also been applied to robust optimization problems in which the disturbance is the index parameter [29]. In our prior work [11], we used SIP with an efficient deepest-penetration oracle to handle collision constraints with complex, non-convex 3D geometries. In this paper, we extend our previous work to explicitly reason about contact forces by encoding frictional contact force constraints, constraints on stability under gravity, and objective functions involving robot joint torques.

A related formulation is the semi-infinite mathematical programming problem with equilibrium constraints (SIMPEC) which is studied in [19]. In SIMPEC the equilibrium constraints are established only on the primary variables, whereas in SIPCC equilibrium constraints are semi-infinite. Moreover, Ref. [19] only studies the optimality conditions and duality of the problem; it does not propose an algorithm to solve SIMPEC.

B. Grasping Pose Optimization

Grasping pose optimization finds a high-quality pose of a robot to grasp an object, where the quality of a pose should consider robot-self collision, robot-object collision, and static equilibrium of the object at the grasping pose.

Model-based grasp planners search explicitly for such a pose using knowledge of the object and robot's geometry. Both sampling-based [17] and continuous optimization methods [6] have been used. Sampling-based grasp planners can use almost any grasp quality metric as the objective function and incorporate collision detection naturally. However, they tend to converge slowly toward an optimum, particularly when the dimensionality of the robot's pose grows high. Continuous optimization, on the other hand, converges more quickly to optima, but requires differentiable geometry representations to encode collision constraints and grasp points as a function of the robot pose. Recent work has formulated a grasp planner that optimizes grasp points and gripper poses in a two-stage fashion [13]. The outer stage searches globally over grasp points to optimize some metric, while the gripper pose is computed by inverse kinematics and collision constraints. However, this approach requires convex geometries where each link can only make contact at a single point with the object. Our method is a local optimization method that handles complex non-convex geometries and can allow multi-point contact on a single link. However, our method does not search for a force closed grasp.

Learning-based grasp planners can predict grasp points and/or gripper poses given noisy observations of the environment. However, they always perform extensive data generation. For example, in [15], Lu et al performed 1507 grasp trials in simulation, and only 159 were successful, because the data are collected by initializing the gripper at some pre-grasping pose and then closing the fingers by some fixed manner. Our method, which takes the geometry information of different objects into account, and could also be easily combined with any differentiable grasp metric, could be a good replacement for the simulator to generate data for the training of learning-based planners.

III. Semi-Infinite Programming with Complementarity Constraints

A. Adding complementarity into semi-infinite programs

A semi-infinite programming problem is an optimization problem in finitely many variables $x \in \mathbb{R}^n$ on a feasible set described by infinitely many constraints:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ s.t. \quad &g(x,y) \geq 0 \quad \forall y \in Y, \end{aligned} \tag{1}$$

where $g(x,y) \in \mathbb{R}^m$ is the constraint function, y denotes the index parameter, and $Y \in \mathbb{R}^p$ is its domain. In the case of pose optimization with collision constraints, x describes the pose of objects in the scene, y is a contact point on the surface of one object, Y denotes that surface, and g is the distance from the point to the other object. Typically, optimal solutions will be supported by contact at some points, i.e., $g(x^\star, y) = 0$ will be met for the optimal solution $x^\star$ at some set of points y.

An additional challenge is posed when forces need to be considered as part of the solution to ensure force and torque balance, since force variables need to be introduced to the optimization problem at each point of contact. We do this by defining a continuous field $z : Y \to \mathbb{R}^r$ in which is an optimization variable. To ensure that forces are only felt at points where objects are in contact, the field is required to satisfy a complementarity condition $z(y)g(x,y) = 0 \quad \forall y \in Y$, which ensures that the force is nonzero only if the distance between the geometries is zero. Meanwhile, there may be some other inequality constraints $h(x,y,z(y)) \leq 0$ that need to be satisfied pointwise, such as friction constraints. Finally, we may require some constraints on the integral of the field over the domain, such as force and torque balance. In this way, we define an SIPCC as a problem in the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n, z\in Y\to\mathbb{R}^r} f(x,z) = f_x(x) + \int_{y\in Y} f_z(x,y,z(y))dy \\ s.t. \quad &g(x,y) \geq 0 \quad \forall y \in Y \\ &z(y)g(x,y) = 0 \quad \forall y \in Y \\ &z(y) \geq 0 \quad \forall y \in Y \\ &h(x,y,z(y)) \geq 0 \quad \forall y \in Y \\ &s(x,z) = s_x(x) + \int_{y\in Y} s_z(x,y,z(y))dy = 0, \end{aligned} \tag{2}$$

The objective $f(x,z)$ and integral constraint $s(x,z)$ are actually functionals, because they accept a function z as input.

SIPCC problems not only have infinitely many constraints like SIP problems, but also introduce a continuous infinity of variables in z. To solve the SIPCC problems using numerical methods, we hope that z only is non-zero at a finite number of points. Indeed, if an optimal solution $x^\star$ is supported by a finite subset of index points $\tilde{Y} = (y^1, y^2, \ldots, y^N) \in Y$, then it suffices to solve for the values of z at these supporting

points, since z should elsewhere be zero. We borrow this concept, which is used in the exchange method used to solve SIP problems [14], to solve the SIPCC problem.

B. Exchange method and oracle

A discretization of an SIPCC problem creates constraints and variables corresponding to a finite number N of instantiated index points $\tilde{Y} = (y^1, \ldots, y^N)$. Force variables $z = (z^1, \ldots, z^N)$ are instantiated for each index point (slightly abusing notation). To replace integrals with sums, the true distribution $z(y)$ is represented by a set of Dirac impulses: $z(y) = \sum_i \delta(y - y^i) z^i$. In this way, the SIPCC problem is converted into a standard mathematical program with complementarity constraints (MPCC) over $(n + Nr)$ variables:

$$\begin{aligned} \min_{x\in\mathbb{R}^n, z\in\mathbb{R}^{Nr}} \; & f_x(x) + \sum_{i=1}^{N} f_z(x, y^i, z^i) \\ s.t. \quad & g(x, y^i) \geq 0 \quad \forall i = 1, ..., N \\ & z^i g(x, y^i) = 0 \quad \forall i = 1, ..., N \\ & z^i \geq 0 \quad \forall i = 1, ..., N \\ & h(x, y^i, z^i) \geq 0 \quad \forall i = 1, ..., N \\ & s_x(x) + \sum_{i=1}^{N} s_z(x, y^i, z^i) = 0, \end{aligned} \tag{3}$$

where r is the number of force variables for each index point. In a slight abuse of notation, when discussing discretized SIPCC problems we will write the discretized objective as $(x, \tilde{Y}, z)$, and stacked vectors of inequality constraints as $g(x, \tilde{Y})$ and $h(x, \tilde{Y}, z)$, and the force and torque balance constraint as $s(x, \tilde{Y}, z)$.

For a solution of the MPCC to correspond to a feasible impulse solution of the original SIPCC, we note that the constraint function h must be conic in z, i.e., $h(x, y, z(y)) \geq 0 \implies h(x, y, c \cdot z(y)) \geq 0$ for any scaling $c > 0$.

To apply the exchange method to SIPCC, we progressively instantiate index sets $\tilde{Y}_1, \tilde{Y}_2, \ldots$ and their finite-dimensional MPCCs whose solutions converge toward the true optimum [20]. Specifically, define P as the original SIPCC, Q_k as the MPCC instantiation corresponding to $\tilde{Y}_k$, and let x_k^* be the solution to Q_k. If the index sets are chosen wisely, we expect that the iterates $x_1^\star, x_2^\star, \ldots$ will eventually approach an optimum of P. A naive approach would sample points incrementally from the domain Y (e.g., randomly or on a grid), and hopefully, with a sufficiently dense set of points, the MPCC solutions will approach an optimal solution. But this approach is inefficient as most samples will not affect the iterated solutions, and also whether the iterated solutions approach an optimal solution of the SIPCC using this strategy is an open question.

The key question here is which new constraint should be selected, and oracle is a subroutine that performs this selection process. In Ref. [11], a maximum-violation oracle that selected closest / deepest penetrating points was used to avoid collisions between the robot and its environment. We argue that adding such points is necessary to solve our problem, but maximum-violation alone may not be sufficient because closest points may not encourage the instantiated MPCC toward finding a solution to the force balance constraints. We show that an oracle that balances the residual of the collision constraint against that of the force balance constraint, as described in Section IV, yields improved solve rates at the expense of additional computation time.

Also, it is possible to delete constraints from the constraint set when they are not deemed necessary (the "exchange"), which saves time in later MPCC solve steps. In our implementation, we delete the index points whose distances to the robot are bigger than a threshold and the forces assigned on that index points are smaller than some other threshold.

C. Optimization of instantiated MPCCs

MPCCs are generally difficult to solve since they are highly degenerate problems and they do not satisfy the majority of Constraint Qualifications established for standard nonlinear optimization [16]. But recently, they have attracted significant attention of operation researchers. The standard form of an MPCC problem is given in Problem 4, where $0 \leq g(x, y) \perp z \geq 0$ means the two vectors are positive and orthogonal, $c_{\mathcal{I}}$ is the inequality constraint and $c_{\mathcal{E}}$ is the equality constraint.

$$\begin{aligned} \min_{x\in\mathbb{R}^n, z\in\mathbb{R}^p} \; & f(x, z) \\ \text{s.t. } & c_{\mathcal{I}}(x, z, g(x, y)) \leq 0 \\ & c_{\mathcal{E}}(x, z, g(x, y)) = 0 \\ & 0 \leq g(x, y) \perp z \geq 0. \end{aligned} \tag{4}$$

We convert Problem 3 into an MPCC in the form of Problem 4 to get the search direction $p = (d_x, d_z)$.

Anitescu shows that SQP with elastic mode converges globally for MPCCs [3]. Moreover, Fletcher et al. [9] presented a large collection of MPCC test problems and compared the performance of standard NLP solvers (SNOPT [10], Knitro [5] and loqo [25]) on those problems. SQP methods are proved to be the most robust at solving MPCCs, and SNOPT had the best performance among all the tested solvers. We use SNOPT in our implementation, and also take the most common smoothing approach for the complimentarity gap, which replaces $g(x, y) \cdot z = 0$ by $g(x, y) \cdot z \leq \tau$ and gradually driving τ to 0.

D. SIPCC Outer Iteration

Once we get a step toward a desired point x_d, z_d, the SIPCC outer loop moves from the current iterate (x_k, z_k) toward x_d, z_d. However, due to nonlinearity, the full step may lead to worse constraint violation. To avoid this problem, we perform a line search over the following merit function that balances reducing the objective and reducing the constraint error:

$$\phi(x, \tilde{Y}, z; \mu) = f(x, z) + \mu \| v(x, \tilde{Y}, z) \|_1, \tag{5}$$

where v denotes the vector of constraint violations of Problem 3, which includes the negative components of each $g^-(x, y^i)$ and $h^-(x, y^i, z^i)$ term, each complementarity

term $z^i g(x, y^i)$, and the force/torque balance term $s_x(x) + \sum_{i=1}^{N} s_z(x, y^i, z^i)$. We denote the negative component of a term as $\cdot^- \equiv \min(\cdot, 0)$. Also, in SIP for collision geometries, a serious problem is that using existing instantiated index parameters, a step may go too far into areas where the minimum of the inequality $g^*(x) \equiv \min_{y \in Y} g(x, y)$ violates the inequality, and the optimization loses reliability. So we add the max-violation $g_k^{-*}(x)$ to v. Moreover, since the scaling of the complementarity constraint is in different units from either g, h, or s, we allow a user-defined weight on each entry of v. Note that, with a fixed index set, our problem becomes a standard MPCC, which is solved using SQP. Therefore, the search direction of the inner problem must be a descendent direction of the above l_1-merit function, as analyzed in Ref. [4].

Ref. [18], Eq. (18.33) gives a reasonable method for choosing μ to ensure that the chosen descent direction p_k computed by SNOPT descends the merit function. It is sufficient to choose any μ_k satisfying

$$\mu_k > \frac{\nabla_{xz} f(x_k, \tilde{Y}_k, z_k)^T p_k}{\|v(x_k, \tilde{Y}_k, z_k)\|}, \tag{6}$$

and hence we simply double the right hand side.

One notable challenge is the complementarity constraint. Because (x_d, z_d) satisfies the constraint and (x_k, z_k) often is quite close to satisfying it, the midpoint between these points is likely to have a large constraint error. Hence, we perform a line search that backtracks only by a small amount on each iteration (we use a 20% reduction).

For an accepted point of the line search that decreases the merit function, we also enforce the requirement that the penetration depth between two geometries is not too deep, because the quality of penetration depth and normal estimation degrades with depth. When the penetration depth exceeds a threshold, that is $g^\star(x) \le g_{min}$, we add the detected penetration point as an index point to $\tilde{Y}_{k+2}$ in the next iteration.

E. Performance tuning

Coherence between problems Q_{k-1} and Q_k may suggest the use of warm-starting to speed up solve. We also keep the forces assigned for each kept index point from Q_{k-1} and use it as the initial value for Q_k. By default, we set the guessed forces for new index points to 0 as listed in Alg. 1 Line 7.

The SNOPT iteration count should be set lower than the default value to avoid spending too much time on problems whose constraint sets are not sufficiently populated. Also, the step size of SNOPT should be set lower than the default value to avoid a deep penetration step.

IV. Stable Grasping Formulation

We apply SIPCC to solve for a gripper/humanoid robot to hold an object, while making sure that the object is in both force and torque balance. We assume that A) the object geometry is known; B) the mass and center of mass (CoM) of the object are known; C) the friction coefficient between the robot and object is known; D) the base joint of the gripper can provide arbitrary large force and torque.

Algorithm 1 SIPCC Solver

Input: N_{outer}^{max}, N_{inner}^{max}, step size tolerance ϵ_x, complementarity gap tolearance ϵ_{gap}, balance tolerance ϵ_s, penetration tolerance ϵ_p, index deletion thresholds z_{min} and g_{max}, initial guess $x_{init} \in \mathbb{R}^n$

1: $x_0 \leftarrow x_{init}$
2: $\tilde{Y}_0 = [\,]$ ▷ Initialize empty constraint set
3: $z_0 \leftarrow \varnothing$ ▷ Initialize empty force vector
4: **for** $k = 0, \ldots, N_{outer}^{max} - 1$ **do**
5: ▷ Update constraint set and guessed forces $\tilde{z}_k$
6: For any $y^i \in \tilde{Y}_k$ with corresponding z^i where $z^i \ge z_{min}$ and $g(x_k, y^i) \le g_{max}$, add y^i to $\tilde{Y}_{k+1}$ and z^i to $\tilde{z}_k$
7: Run the oracle to add one or more new points to $\tilde{Y}_{k+1}$. Initialize their force(s) z^i to 0
8: ▷ Solve for step direction
9: $\tau_0 \leftarrow \tilde{z}_k^T g(x_k, \tilde{Y}_{k+1})$
10: Run SNOPT to yield the desired endpoint x_d, z_d
11: ▷ Line Search
12: $\alpha \leftarrow 1$
13: $\Delta x \leftarrow x_d - x_k$, $\Delta z \leftarrow z_d - \tilde{z}_k$
14: Calculate μ_k from (6)
15: $score_0 = \psi(x_k, \tilde{Y}_{k+1}, \tilde{z}_k; \mu_k)$
16: converged $\leftarrow$ false
17: **while** ¬converged **and** $n_{inner} < N_{inner}^{max}$ **do**
18: $x \leftarrow x_k + \alpha \Delta x$, $z \leftarrow \tilde{z}_k + \alpha \Delta z$
19: $score = \psi(x, \tilde{Y}_{k+1}, z; \mu_k)$
20: **if** $score \le score_0$ **then**
21: converged $\leftarrow$ true
22: **if** $\min_{y \in Y} g(x, y) \le g_{min}$ **then**
23: Add $y^* = \arg\min_{y \in Y} g(x, y)$ to $\tilde{Y}_{k+2}$
24: **else**
25: $\alpha \leftarrow \alpha \cdot 0.8$
26: $n_{inner} \leftarrow n_{inner} + 1$
27: ▷ Update state and test for convergence
28: $x_{k+1} \leftarrow x_k + \alpha \Delta x$
29: $z_{k+1} \leftarrow \tilde{z}_k + \alpha \Delta z$
30: **if** $\alpha \|\Delta x\| \le \epsilon_x$ and $z_{k+1}^T g(x_{k+1}, \tilde{Y}_{k+1}) \le \epsilon_{gap}$ and $\sum g_{k+1}^{-*}(x) < \epsilon_p$ and $\|s_0\| < \epsilon_s$ **then return** x_{k+1},z_{k+1}

A. Geometry modeling

To handle collision avoidance, we establish a semi-infinite constraint $g(x, y)$ between the robot and the object. We perform some pre-computation to accelerate SIPCC solve. The object is represented as a point cloud with resolution 1 mm, and the robot links are represented as a signed distance field (SDF) with resolution 0.8 mm, which supports $O(1)$ depth lookup and $O(1)$ gradient estimation at a point.

B. Friction force modeling

We model the contacts between the robot and each index point on the object as point contacts with dry friction. The i'th force variable $z^i = (f_i^N, f_{i,1}^F, \ldots, f_{i.j}^F)$ represents the force applied at the i'th contact y^i, which is divided into the normal component f_i^N and j frictional components $f_{i,j}^F$ along the edges of a polyhedral approximation of the friction cone [24]. Given the normal vector $\mathbf{n}_i$ along the outward surface normal, and d tangential directions $\mathbf{t}_{i,j}$, the overall force applied at at y^i is $\mathbf{f}_i = f_i^N \mathbf{n}_i + \sum_j f_{i,j}^F \mathbf{t}_{i,j}$. Each of the components of z^i is required to be non-negative ($z^i \ge 0$), and given the friction coefficient μ_i, the friction cone constraints are given by $\sum_j f_{i,j}^F \le \mu_i f_i^N$.

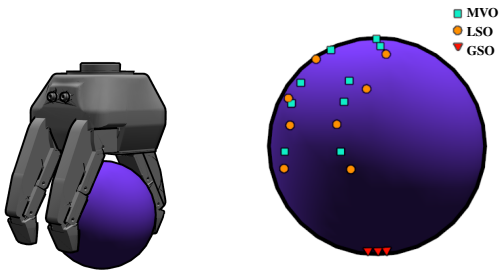


Fig. 2: Comparison of different oracles at the given pose. Squares, circles and triangles indicate the index points instantiated by MVO, LSO and GSO, respectively. [Best viewed in color.]

C. Force and torque balance

We establish an integral equality constraint on the object force and torque balance. Force balance requires that the force exerted by the object on the robot matches the gravity force of the object, $\sum_i \mathbf{f}_i = m\mathbf{g}$. Torque balance requires that the torque applied to the object to be zero, $\sum_i \mathbf{r}_i \times (-\mathbf{f}_i) = 0$, where $\mathbf{r}_i = y^i - \text{CoM}$ is the vector from the CoM of the object to the index point y^i.

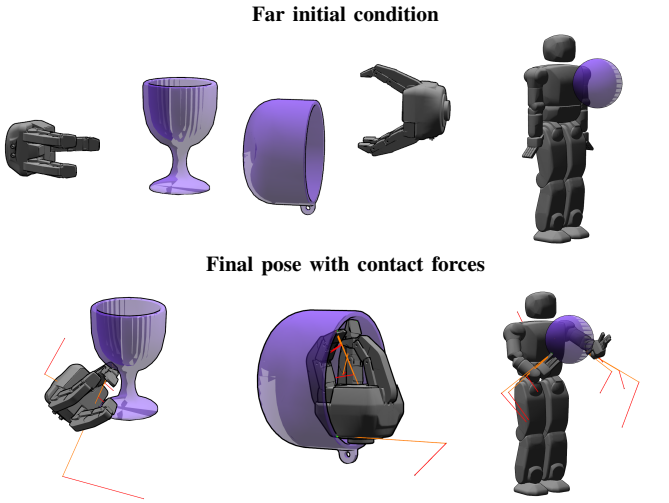
We adopt a heuristic to reduce the number of constraints in the complementarity condition and accelerate solve times. By only applying complementarity to the normal component, $g(x, y^i) \cdot f_i^N = 0$, we reduce the number of complementarity constraints from Nr to N, and the problem is unchanged because $f_i^N = 0 \Rightarrow z^i = 0$ due to the friction cone constraint.

D. Custom oracle

As discussed in Section III.B, an oracle should be designed to instantiate a sequence of contact points y that help the SIPCC find a feasible solution. The ***Maximum violation oracle*** (MVO) chooses the closest point y for each link $\arg\min_{y \in Y} g(x, y)$, but fails to consider the balance and complementarity constraints.

In Algorithm 1, any (deepest) penetrating point is automatically added to the constraint set, so we turn our attention to non-penetrating index points. The complementarity constraint alone is not sufficient to select y, since for any y for which $g(x, y) > 0$, we can let $z^i = 0$ to satisfy complementarity. Instead, we must consider the interaction of force-torque balance with the other constraints. The balance residual at a new point y, if we were to find a force z, is $s_0(x) + s_z(x, y, z)$, where $s_0(x) = s_x(x) + \sum_{i=1}^N s_z(x, y^i, z^i)$ is the residual at the current index points. Then, an ideal new index point y, would enable solving for a new pose x' and force z to simultaneously satisfy $g(x', y) \geq 0$, $h(x', y, z) \geq 0$, and $s_0(x') + s_z(x', y, z) = 0$.

Solving these constraints directly would require finding a constraint-minimizing (x', z) for every $y \in Y$, which would be expensive to solve. So we propose an approximate score that is expected to take large values for high quality y. We define $-s_0(x) \cdot s_z(x, y, 1)$ as a score for the balance constraint, and $\exp(-g(x, y))$ as a score for the contact constraint, since a point with a small $g(x, y)$ has a good chance to satisfy the complementarity constraint if a force would be assigned at that point.



(a) Gripper, Wine glass (b) Gripper, Bowl (c) Humanoid, Sphere

Fig. 3: The same test cases as in Figure 1 but with farther initial conditions. SIPCC is still able to find a pose to hold the object, and sometimes determines unusual strategies. [Best viewed in color.]

We define an overall score $w_1 \cdot (s_0 \cdot s_z(x, y, 1)) + w_2 \cdot exp(-g(x, y))$ that should be maximized at a high-quality point, where w_1 and w_2 are custom weights. This is too expensive to minimize over the entire domain Y, so we propose an alternate approach, called the ***Local score oracle*** (LSO), that only minimizes over a neighborhood of the MVO point. Overall, we choose $y^* = \arg\max_{y \in B_r(y_M^*)} [w_1 \cdot (s_0 \cdot s_z(x, y, 1)) + w_2 \cdot exp(-g(x, y))]$, where $B_r(y_M^*)$ is an neighborhood of the MVO point y_M^* with radius r. Here, r is a user-defined parameter. When $r = 0$, LSO is equivalent to MVO, and when $r = \infty$, the optimization occurs over the entire object, which we call the ***Global Score Oracle*** (GSO). We perform this computation for each robot link.

The index points instantiated by the three oracles at the same configuration are shown in Figure 2. For LSO, r is chosen to include $2,000$ points around the MVO point. From the results we can see that the GSO finds points that would immediately cause balance constraints to be met, but does not respect the initial pose. LSO strikes a balance between proximity and points that have better normals to apply upward forces.

V. Experiments

The SIPCC algorithm is implemented with front end in the Python programming language, with the SNOPT solver [10], and custom C++ collision detection software. The communication between Python and SNOPT is through the pyOptSparse [28]. The Klamp't library is used for robot kinematics, collision queries, and visualization [1]. The objects are taken from Princeton Shape Benchmark [23] and Ycb benchmark object and model set [2]. All experiments were run on a single core of a 3.6 GHz AMD Ryzen 7 processor.

To show the generality of our algorithm, we first solve for a Robotiq Adaptive 3-finger gripper holding a wine glass and a bowl, and a HUBO2 humanoid robot (which can be viewed as a giant gripper) holding a large sphere. In this and subsequent experiments, we set the minimum signed distance

Test Case	# in PC	DoF	Initial	Iter	Time (s)	Comp Gap (N · m)	Bal Res	Pen (m)	# Contact	Ave Index	Ave Active Index
Gripper, Wine Glass	436,804	18	Close	22	33.6	1.1e-3	9.8e-14	9.3e-4	10	86.5	15.9
			Far	14	53.1	1.1e-4	2.4e-13	6.1e-4	6	74.6	5.9
Gripper, Bowl	674,594	18	Close	12	17.8	2.6e-15	2.6e-15	4.6e-4	10	63.5	12.1
			Far	8	9.9	9.1e-14	9.1e-14	5.6e-4	7	57.6	4.9
Humanoid, Sphere	28,362	63	Close	23	199.3	1.6e-5	4.9e-15	8.6e-4	8	215.4	5.3
			Far	24	246.3	6.8e-3	2.3e-13	7.2e-4	8	204.3	3.7

TABLE I: Test results, listing # points in the point cloud (# in PC), robot degrees of freedom (DoF), outer iterations (Iter), computation time (Time), complementarity gap at final pose (Comp Gap), balance residual at final pose (Bal Res), sum of penetration depth of all the robot links (Pen), # contact points at final pose (# Contact), average # of index points instantiated in each iteration (Ave Index) and average # of index points kept after "exchange" (Ave Active Index).

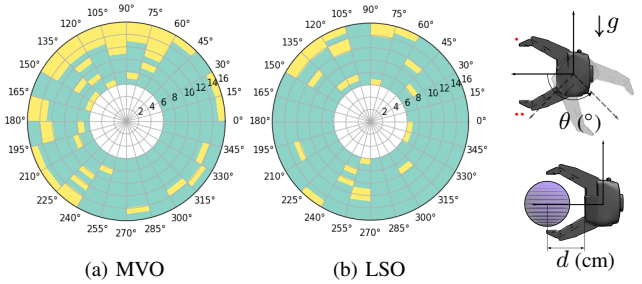
Oracle		Time (s)	Pen (m)	Comp Gap (N · m)	Bal Res	Success (%)
LSO	Mean	27.71	5.2e-4	1.2e-3	2.3e-3	
	Std	15.52	2.8e-4	1.7e-3	1.2e-3	89.17
	Median	23.03	5.3e-4	5.1e-4	9.3e-16	
MVO	Mean	13.99	4.9e-4	1.6e-3	1.4e-3	
	Std	12.53	2.8e-4	2.1e-3	8.1e-4	78.75
	Median	11.28	5.1e-4	8.1e-4	7.1e-16	

TABLE II: Convergence test results include mean, standard deviation (Std) and median of the computation time (Time), sum of penetration depth of all the robot links (Pen), complimentarity gap at the final pose (Comp Gap), balance residual at the final pose (Bal Res), and success rate.

to $g_{min} = -10^{-3}$ m. The solver terminates with success if the L_1 norm of penetrations of all links is smaller than 1 mm, the complementary gap is smaller than 10^{-2} N·m, and the balance residual is smaller than 0.01.

We start the robot first from a close initial position (Figure 1) and then a faraway initial position (Figure 3). The results show that our method can generate contact with "unusual" parts of the geometry. The gripper supports the object using the side of the finger, the end of the finger, the palm, and even the back of the finger. The humanoid cradles the sphere between its torso and hands. Moreover, multiple contacts can happen on the same robot link, which distinguishes our method from prior works on grasp planning, e.g. [13], where precision grasps are planned with designated contact points. The test results are summarized in Table I.

Next, we examine the convergence of SIPCC under different initial conditions and oracle choices (Figure 4). The gripper is required to grasp a sphere. We initialize the gripper from different positions (0.06–0.12 m, 0.02 m/step) and orientations ([0,2π), $\frac{\pi}{12}$/step) around a circle. MVO and LSO oracles are compared, and for LSO r is chosen so that $B_r(y_M^*)$ contains 2,000 points. We observe that MVO's performance decreases rapidly when the initial distance between the gripper and the object increases, while LSO is much more robust. Moreover, the success rate of the left half circle is lower than the right half circle, which is caused by the fact that the single-finger side of the gripper is underneath the object when $\theta \in (90°, 270°)$. Besides, the success rate of the upper half circle is lower than the lower half circle, and this is because when the initial index points allow the inner optimization to find a feasible solution, the algorithm is likely to find a solution. Otherwise, more burden is put on the oracle to choose index points to guide the gripper to a feasible configuration. Detailed test results are summarized in Table II.



(a) MVO (b) LSO

Fig. 4: Convergence results of MVO and LSO oracles, for sphere grasping. Green is success and yellow is fail. The angular axis represents the CCW rotation angle of the gripper in the vertical plane, with gravity pointing downwards. The radial axis represents the distance in cm between the CoM of the sphere and the palm plane of the gripper. Red dots indicate the number of fingers on each side. [Best viewed in color.]

VI. Conclusion and Future Work

In this paper, we present a novel computational model to address the problem that contact is an infinite phenomena involving continuous regions of interaction. We cast the problem a SIPCC, and solve it through dynamically determining a finite number of constraints that are most relevant to solving the problem and solving a series of problems whose solutions converge toward one that contains a true optimum of the original SIPCC.

Future work should study the convergence of SIPCC algorithms to establish theoretical conditions by which a merit function and an oracle enable convergence toward critical points of the original continuous SIPCC. Our current investigations suggest that an oracle that chooses an index point that leads to steepest descent of a merit function is a sound choice, but evaluating this condition for each point is computationally expensive. Approximation techniques might lead to oracles that are both theoretically sound and generalizable to other problem settings. It would also be interesting to study methods to handle uncertainty in the problem parameters, such as friction or external forces. For friction, the required friction coefficient could be incorporated to the objective function so that the grasp is likely to obey the true friction. For uncertain external disturbances, the worst-case disturbance could be included as a separate index variable. Furthermore, we would like to improve running times, e.g., by using faster solvers rather than off-the-shelf NLP solvers. Accelerating the method is necessary for it to be used in online behavior, and scale toward larger problems like trajectory optimization or multi-object manipulation.

REFERENCES

[1] Klampt – Intelligent Motion Laboratory at UIUC. [Online]. Available: http://motion.cs.illinois.edu/klampt/

[2] Ycb benchmarks object and model set. [Online]. Available: http://ycbbenchmarks.org

[3] M. Anitescu, "On solving mathematical programs with complementarity constraints as nonlinear programs," *Preprint ANL/MCS-P864-1200, Argonne National Laboratory, Argonne, IL*, vol. 3, 2000.

[4] P. T. Boggs, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, Jan. 1995.

[5] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[6] I.-M. Chen and J. W. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 507–512, 1993.

[7] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," 2012.

[8] C. Eppner and O. Brock, "Planning grasp strategies that exploit environmental constraints," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4947–4952.

[9] R. Fletcher* and S. Leyffer, "Solving mathematical programs with complementarity constraints as nonlinear programs," *Optimization Methods and Software*, vol. 19, no. 1, pp. 15–40, 2004.

[10] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[11] K. Hauser, "Semi-infinite programming for trajectory optimization with nonconvex obstacles," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 565–580.

[12] R. Hettich and K. O. Kortanek, "Semi-infinite programming: theory, methods, and applications," *SIAM review*, vol. 35, no. 3, pp. 380–429, 1993.

[13] M. Liu, Z. Pan, K. Xu, and D. Manocha, "New formulation of mixed-integer conic programming for globally optimal grasp planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4663–4670, 2020.

[14] M. López and G. Still, "Semi-infinite programming," *European Journal of Operational Research*, vol. 180, no. 2, pp. 491–518, 2007.

[15] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Robotics Research*. Springer, 2020, pp. 455–472.

[16] Z.-Q. Luo, J.-S. Pang, and D. Ralph, *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[17] A. T. Miller and P. K. Allen, "Graspit!: A versatile simulator for grasp analysis," in *in Proc. of the ASME Dynamic Systems and Control Division*. Citeseer, 2000.

[18] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[19] Y. Pandey and S. Mishra, "Optimality conditions and duality for semi-infinite mathematical programming problems with equilibrium constraints, using convexificators," *Annals of Operations Research*, vol. 269, no. 1-2, pp. 549–564, 2018.

[20] R. Reemtsen and S. Görner, "Numerical methods for semi-infinite programming: a survey," in *Semi-Infinite Programming*. Springer, 1998, pp. 195–275.

[21] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[22] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[23] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.* IEEE, 2004, pp. 167–178.

[24] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[25] R. J. Vanderbei and D. F. Shanno, "An interior-point algorithm for nonconvex nonlinear programming," *Computational Optimization and Applications*, vol. 13, no. 1-3, pp. 231–252, 1999.

[26] A. I. F. Vaz, E. M. Fernandes, and M. P. S. Gomes, "Robot trajectory planning with semi-infinite programming," *European Journal of Operational Research*, vol. 153, no. 3, pp. 607–617, 2004.

[27] F. G. Vázquez, J.-J. Rückmann, O. Stein, and G. Still, "Generalized semi-infinite programming: a tutorial," *Journal of computational and applied mathematics*, vol. 217, no. 2, pp. 394–419, 2008.

[28] N. Wu, G. Kenway, C. A. Mader, J. Jasa, and J. R. R. A. Martins, "pyoptsparse: A python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, vol. 5, no. 54, p. 2564, 2020. [Online]. Available: https://doi.org/10.21105/joss.02564

[29] B. Zeng and L. Zhao, "Solving two-stage robust optimization problems using a column-and-constraint generation method," *Operations Research Letters*, vol. 41, no. 5, pp. 457–461, 2013.