Second-Order Convergent Collision-Constrained Optimization-Based Planner

Chen Liang[1], Xifeng Gao[2], Kui Wu[2], Zherong Pan[2†]

Abstract—**Finding robot poses and trajectories represents a foundational aspect of robot motion planning. Despite decades of research, efficiently and robustly addressing these challenges is still difficult. Existing approaches are often plagued by various limitations, such as intricate geometric approximations, violations of collision constraints, or slow first-order convergence. In this paper, we introduce two novel optimization formulations that offer provable robustness, achieving second-order convergence while requiring only a convex approximation of the robot's links and obstacles. Our first method, known as the Explicit Collision Barrier (ECB) method, employs a barrier function to guarantee separation between convex objects. ECB uses an efficient matrix factorization technique, enabling a second-order Newton's method with an iterative complexity linear in the number of separating planes. Our second method, referred to as the Implicit Collision Barrier (ICB) method, further transforms the separating planes into implicit functions of robot poses. We show such an implicit objective function is twice-differentiable, with derivatives evaluated at a linear complexity. To assess the effectiveness of our approaches, we conduct a comparative study with a first-order baseline algorithm across six testing scenarios. Our results unequivocally justify that our method exhibits significantly faster convergence rates compared to the baseline algorithm.**

Index Terms—**Trajectory and Pose Optimization, Collision Constraint, Convex Analysis**

I. INTRODUCTION

The issue of robot safety stands as a paramount concern, imposing stringent constraints on motion planning algorithms. Among these constraints, collision avoidance requirements demand that the robot maintains a minimum distance from static or moving obstacles at all times. This paper addresses the challenge of generating optimal robot poses and trajectories while adhering to collision constraints and optimizing a user-defined cost function. Such a problem finds practical applications in (multi-)UAV trajectory planning [1], inverse kinematics [2], object settling [3], and dynamic simulations [4]. Ideally, an effective planning algorithm should possess several key attributes to cater to the diverse needs of these applications: (Scalability) handling a significant number of robot links and obstacles; (Efficacy) rapid convergence and low iterative and overall computational costs; (Robustness) guarantee to satisfy all collision constraints; (Generality) applying to arbitrary articulated robots with complex geometries. However, despite decades of dedicated research in pursuit of such algorithms, it remains a challenge to find a single solution that fully satisfies all these diverse requirements.

Existing trajectory generation algorithms are engineered to suit a set of specific applications. Take, for instance, (online) UAV trajectory generation, where computational efficiency is

paramount. Existing algorithm [5] employs penalty methods to handle constraints, which can compromise robustness. In contrast, other algorithms such as [1, 6] offer guaranteed robustness and global optimality. Unfortunately, their applicability is limited to point robots and convex-decomposable freespace. When dealing with general articulated robots, the formulation of collision constraints becomes notably challenging. Many optimization-based algorithms, as seen in the works [7, 8], once again resort to less robust penalty-based techniques. Furthermore, these algorithms often exhibit limited efficacy, having only first-order convergence. Recent advancements [9, 10] have introduced interior point methods to address collision constraints with guaranteed robustness. However, Li *et al.* [9] necessitates the decomposition of robot geometry into a large number of triangles, which scales poorly with the complexity of the robot geometry. On the other hand, Zhang *et al.* [10] relies on the strict convexity of robot links, demanding intricate geometric modifications and suffering from first-order convergence limitations.

Method	Scalability	Efficacy	Robustness	Generality
[5]	point-penalty	second-order	no	point
[1, 6]	convex	mixed-integer	yes	point
[9]	triangle	second-order	yes	linear motion
[10]	convex	first-order	yes	articulated
[3]	point	second-order	no	articulated
[7, 8]	point-penalty	first-order	no	articulated
[11]	convex	first-order	yes	articulated
Ours	convex	second-order	yes	articulated

TABLE I: Comparison of planning methods across four dimensions: (Scalability) granularity (convex>triangle>point) of collision constraints, finer granularity can result in a larger number of constraints with limited scalability (certain algorithms use point-wise soft penalty and do not involve hard constraints, denoted as point-penalty); (Efficacy) type/convergence-speed of underlying numerical algorithm; (Robustness) whether collision constraints are guaranteed to be satisfied; (Generality) allowed type of robot motions.

Main Results: We present two innovative optimization solvers designed for collision-constrained robot pose and trajectory planning. Our approach highlights a structural analysis of the barrier function governing collision constraints between pairs of convex polytopes. We demonstrate that the Hessian matrix of this barrier function possesses a unique structure that enables efficient factorization using a Schur-complement solver, leading to our Explicit Collision Barrier (ECB) method. Furthermore, by employing a generalized barrier function, we eliminate the need for a separating plane between convex polytope pairs by representing it as an implicit function of robot poses, leading to our Implicit Collision Barrier (ICB) method. Both of these advancements empower us to apply second-order convergent Newton's methods with an iterative cost linear in the number of convex polytope pairs.

Our method exhibits several advantageous properties that have not been simultaneously achieved by prior algorithms. First, we decompose the robot's geometry into convex polytopes rather than points [3] or triangles [9]. As a result, our method can scale to large, complex robot and environmental

geometries using a moderate number of geometric primitives, leading to enhanced scalability. Second, in contrast to previous first-order convergent methods [12, 10], our approach enjoys faster second-order convergence. Finally, our method adopts the interior point algorithm that inherits the robustness established by [9, 10], i.e., our method ensures the collision-free properties of the optimize robot poses/trajectories. We have conducted evaluations of our method across six challenging robot planning scenarios. Our results unequivocally demonstrate that our approach achieves significantly faster convergence compared to the first-order baseline method.

II. RELATED WORK

We review related work on collision-constrained planning, collision constraint formulations, numerical optimization techniques, and semi-infinite collision constraints.

Collision-Constrained Planning: The algorithms summarized in Table I predominantly fall into the category of local optimization methods. These approaches leverage (high-order) derivatives of the objective/constraint functions to efficiently identify locally optimal plans. In contrast, optimal sampling-based methods [14, 15] pursue globally optimal plans by exploring the entire configuration space. However, a notable drawback of these methods is their limited scalability concerning the dimensionality of the configuration space [16]. An additional category of techniques includes the control barrier function [17], designed for dynamic systems to adhere to specific constraints. However, these algorithms primarily focus on constraint satisfaction and do not directly address the planning of robot motions that fulfill users' requirements. We also acknowledge recent advancements [18, 19] that involve precomputing feasible sub-domains within the configuration space. Subsequently, mixed-integer second-order numerical solvers are employed to derive globally optimal motion plans restricted to these sub-domains. This approach represents a promising alternative to our method. Their method uses expensive precomputations to find a sequence of convex subsets in the freespace and then solve a mixed-integer programming to find the globally optimal solution restricted to the subsets. Instead, our method does not need the sub-domain precomputation or restriction, but our solution can only achieve local optimality.

Collision-Constraint Formulations: The local optimization techniques outlined in Table I can be distinguished by the granularity at which they formulate collision constraints. Several methods [5, 3] opt to replace collision constraints with point constraints, due to their neat closed-form derivatives. However, this approach necessitates a considerable number of point constraints to ensure the safety of a robot with complex geometries. Recent advancements, as exemplified in [9], extend point constraints to collision constraints between pairs of triangles, benefiting from closed-form second-order derivatives. Nevertheless, even this approach requires a substantial number of triangles to represent intricate robot geometries, resulting in a high volume of collision constraints. As proposed in [13, 11], the number of geometric primitives can be significantly reduced if we can formulate collision

constraints between general convex hulls. While Escande *et al.* [12] demonstrated that the distance function between convex hulls possesses first-order derivatives, a comprehensive exploration of second-order differentiability and the efficient evaluation of convex constraints remains an area that requires further investigation.

Numerical Optimization Techniques: Methods from Table I harness the power of numerical optimization solvers to search for optimal plans. The majority of existing approaches [5, 1, 6] cast the problem in the form of a conventional (mixed-integer) Nonlinear Program (NLP) and subsequently employ off-the-shelf optimization software packages [20, 21]. These algorithms typically exhibit second-order convergence when an accurate Hessian matrix can be evaluated. However, they regress to first-order convergence when solely gradient information is accessible, as in [12]. Furthermore, customized optimization solvers have been devised to enhance robustness or efficiency. For instance, Li *et al.* [9] seamlessly integrated continuous collision checks into the line-search procedure to circumvent the challenge of tunneling, where the robot has no collisions at sampled time instances, but collisions can occur in between. Meanwhile, methods like [3, 10] strategically sample collision constraints along the trajectory to approximate the solution of a Semi-Infinite Program (SIP). Ni *et al.* [11] introduced a first-order ADMM approach that interleaves the optimization of robot trajectories and the separating planes between convex hulls to facilitate parallel computation.

Semi-Infinite Collision Constraints: As clarified in Section I, the necessity to satisfy collision constraints at infinitely many time instances inherently gives rise to SIP problem instances, a concept recently articulated by [3, 10]. While this paper primarily addresses a finite set of collision constraints, we emphasize that our methodology can be adapted to tackle SIP problems. Notably, existing SIP solvers propose to transform SIP into a standard NLP by either sampling [3] or discretizing [10] the constraints. In this context, we can readily substitute the underlying solver in [3, 10] with our approach to achieve second-order convergence when dealing with convex constraints. However, it is noteworthy that, while this enables second-order convergence in the NLP subproblem, the convergence speed of the original SIP problem remains an open question and warrants further exploration, which we leave as a subject for future work.

III. PROBLEM STATEMENT

We address the challenge of collision-constrained optimization, focusing on the task of optimizing the pose/trajectory of a robot within the context of a twice-differentiable objective function $\mathcal{O}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the robot's configuration parameter. The robot is subject to a set of collision constraints taking the following general form:

$$C_i(\boldsymbol{\theta}) \cap C_j(\boldsymbol{\theta}) = \varnothing \quad \forall ij \in \mathcal{C},$$

Here, C_i and C_j represent pairs of potentially contacting objects drawn from the set of contact pairs denoted as $\mathcal{C}$. $C_\bullet$ can correspond to a link within an articulated robot or a static
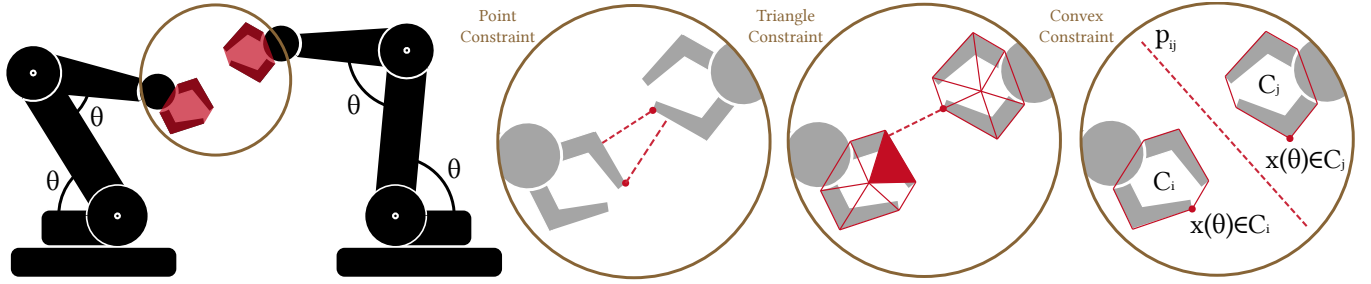
Fig. 1: We exemplify the optimization process involving two articulated robot poses, wherein the two end-effectors (red) must maintain a certain separation distance. Previous approaches [5, 3] substitute such collision constraints with point constraints, while [9] suggests discretizing the end-effector using a triangle mesh and employing triangle constraints. In contrast, our method builds upon the convex constraints formulation introduced by [11, 13], where a separating plane denoted as p_{ij} is jointly optimized along with the robot poses. While [11, 13] are first-order convergent methods, our approach stands out with its local second-order convergence.

obstacle. For the sake of generality, we assume that $\mathcal{O}(\boldsymbol{\theta})$ is a function of the robot's configuration. Additionally, we presume that $C_\bullet$ possesses a (not necessarily strictly) convex shape, which is a widely adopted geometric representation for robots [12, 22, 18]. Using convex shapes can effectively reduce the number of constraints as compared with point-pair or triangle-pair constraints. Non-convex shapes can be approximately decomposed into convex primitives [23]. Although such approximations can shrink the space of feasible solutions, the error can be made arbitrarily small at the price of increasing the number of convex constraints. Consequently, the primary optimization model addressed in this paper can be summarized as:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; \mathcal{O}(\boldsymbol{\theta}) \quad \text{s.t. } C_i(\boldsymbol{\theta}) \cap C_j(\boldsymbol{\theta}) = \varnothing \quad \forall ij \in \mathcal{C}. \quad (1)$$

Equation 1 embraces a generalized formulation that encompasses both the optimization of a static robot pose and a dynamic robot trajectory. In the case of optimizing a static robot pose, $\boldsymbol{\theta}$ corresponds to a single configuration, and $\mathcal{C}$ is a finite set that enumerates all potential convex object pairs susceptible to collision. In contrast, when optimizing a robot trajectory, $\boldsymbol{\theta}$ represents the parameterization of the robot's trajectory, such as the control points of a spline curve [1, 24]. In this scenario, $\mathcal{C}$ expands to an infinite set that encompasses all possible convex object pairs at infinitely many time instances across the trajectory. This extension leads to the formulation of SIP problem instances. In the latter case, $\mathcal{C}$ can be further approached by sampling at finite time instances [3] or discretizing [10], thus unified with the former case.

IV. METHOD

The effective resolution of Equation 1 poses practical challenges, particularly when dealing with convex collision constraints known for their non-differentiability [12, 3]. As illustrated in Figure 1, when complex environmental geometries are considered, the number of collision pairs can rapidly escalate, leading to sizable problem dimensions. In this study, we introduce two efficient optimization techniques, both featuring local second-order convergence and an iterative cost linear in the number of collision pairs. Our approach draws inspiration from recent research [13, 11], which introduced separating planes between pairs of convex objects, represented by the normal direction $\boldsymbol{n}_{ij}$ and offset d_{ij}. However, prior works [13,

11] employed an Alternating Optimization (AO) approach that interleaved the updates of $\boldsymbol{n}_{ij}, d_{ij}$ and $\boldsymbol{\theta}$. Unfortunately, this approach is limited to at most first-order convergence speed [25]. Our primary contribution is the development of two quasi-Newton algorithms that jointly optimize $\boldsymbol{n}_{ij}, d_{ij}$, and $\boldsymbol{\theta}$ without significantly increasing the computational overhead. To achieve this, we augment our objective function with a barrier energy designed to prevent the intersection of C_i and C_j. In our ECB method, we derive the Hessian matrix of the augmented objective function concerning both $\boldsymbol{n}_{ij}, d_{ij}$, and $\boldsymbol{\theta}$. Subsequently, we efficiently invert this large Hessian matrix using the Schur-complement lemma [26]. In our ICB method, we eliminate $\boldsymbol{n}_{ij}$ and d_{ij} by expressing them as implicit functions of $\boldsymbol{\theta}$ and derive its Hessian using the implicit function theorem [27].

A. ECB Method

A convex object can be represented in two ways: the V-representation characterizes $C_\bullet$ using a set of vertices, while the H-representation characterizes it using a set of separating planes. In this work, we consistently utilize the V-representation. Without any ambiguity, we denote $C_\bullet$ as the set of vertices. We assume each vertex $\boldsymbol{x}(\boldsymbol{\theta}) \in C_\bullet$ is a twice-differentiable function of $\boldsymbol{\theta}$, which holds generally for articulated robot with $\boldsymbol{x}(\boldsymbol{\theta})$ being the forward-kinematic function. To define a valid separating plane $\boldsymbol{p}_{ij} \triangleq (\boldsymbol{n}_{ij}, d_{ij})$, it should ensure that the vertices of C_i and C_j are on opposite sides of it, i.e.:

$$\boldsymbol{n}_{ij}^T \boldsymbol{x} + d_{ij} < 0 \; \forall x \in C_i \wedge \boldsymbol{n}_{ij}^T \boldsymbol{x} + d_{ij} > 0 \; \forall x \in C_j.$$

We enforce each of these linear constraints by introducing a barrier energy function $P(\bullet)$ and augmenting the objective function as follows:

$$\bar{\mathcal{O}}(\boldsymbol{\theta}, \cdots, \boldsymbol{p}_{ij}, \cdots) = \mathcal{O}(\boldsymbol{\theta}) + \sum_{ij \in \mathcal{C}} P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij})$$

$$P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij}) \triangleq \sum_{x \in C_i} P(-\boldsymbol{n}_{ij}^T \boldsymbol{x}(\boldsymbol{\theta}) - d_{ij}) + \sum_{x \in C_j} P(\boldsymbol{n}_{ij}^T \boldsymbol{x}(\boldsymbol{\theta}) + d_{ij}),$$

where we assume the barrier function $P(\bullet)$ is strictly convex, defined in the interval $(0, \infty)$, and satisfies the condition $\lim_{\bullet \to 0^+} P(\bullet) = \infty$, where the last property ensures the feasibility of our problem, since any finite function value corresponds to a feasible solution. Put together, our primary

optimization problem Equation 1 is approximately by the following barrier-augmented optimization:

$$\underset{\boldsymbol{\theta},\boldsymbol{p}_\bullet}{\text{argmin}} \ \bar{\mathcal{O}}(\boldsymbol{\theta},\cdots,\boldsymbol{p}_{ij},\cdots) \quad \text{s.t.} \ \|\boldsymbol{n}_{ij}\| = 1 \quad \forall ij \in \mathcal{C}.$$

Note that we still need an additional constraint to ensure unit normal vectors for the separating planes. This method was previously proposed in [11]. However, they optimized $\boldsymbol{p}_{ij}$ and $\boldsymbol{\theta}$ in separate sub-problems. In contrast, we introduce a joint optimization formulation for both $\boldsymbol{p}_{ij}$ and $\boldsymbol{\theta}$, achieving faster convergence. We refer to this method as the ECB method since all variables are explicitly treated as independent decision variables.

Our approach to optimizing the explicit formulation employs a standard Newton's method. We calculate the joint Hessian matrix $\nabla^2\bar{\mathcal{O}}$ and then determine the joint update direction $\boldsymbol{\delta} \triangleq \left(\delta\boldsymbol{\theta}^T,\cdots,\delta\boldsymbol{p}_{ij}^T,\cdots\right)^T$ by solving the following KKT system:

$$\boldsymbol{0} = \begin{pmatrix} \nabla^2\bar{\mathcal{O}} & \boldsymbol{J} \\ \boldsymbol{J}^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\delta} \\ \boldsymbol{\lambda} \end{pmatrix} + \begin{pmatrix} \nabla\bar{\mathcal{O}} \\ \boldsymbol{0} \end{pmatrix}, \tag{2}$$

where $\boldsymbol{J}$ represents the Jacobian of the unit normal constraint $\|\boldsymbol{n}_{ij}\| = 1$, with each column of $\boldsymbol{J}$ corresponding to the normal vector of a separating plane. However, this straightforward approach can encounter two main issues. Firstly, when dealing with complex environmental geometries, the number of separating planes can reach tens of thousands [10], resulting in large Hessian matrices. If we use standard direct factorization methods to invert such matrices, the computational overhead can compromise the advantages of second-order convergence. Additionally, the Hessian matrix can exhibit exceedingly large condition numbers, leading to numerical instability. Fortunately, we can leverage the unique structure of the Hessian matrix to develop an efficient and stable matrix factorization scheme with linear complexity in the number of separating planes.

We first notice that a separating plane only appears in a single term P_{ij}. Therefore, the Hessian matrix takes the following form:

$$\nabla^2\bar{\mathcal{O}} = \begin{pmatrix} \nabla_{\boldsymbol{\theta\theta}}\bar{\mathcal{O}} & \cdots & \nabla_{\boldsymbol{\theta p}}P_{ij} & \cdots \\ \vdots & \ddots & & \\ \nabla_{\boldsymbol{p\theta}}P_{ij} & & \nabla_{\boldsymbol{pp}}P_{ij} & \\ \vdots & & & \ddots \end{pmatrix}.$$

We propose a Gauss elimination approach to solve the KKT system in Equation 2 and show that this procedure is well-defined. We first notice that $\boldsymbol{\lambda}$ can also be decomposed into λ_{ij}, each related to one separating plane. The joint linear sub-system of $\boldsymbol{p}_{ij}, \lambda_{ij}$ takes the following form:

$$\begin{pmatrix} \nabla_{\boldsymbol{p\theta}}P_{ij} \\ 0 \end{pmatrix}\delta\boldsymbol{\theta} + \begin{pmatrix} \nabla_{\boldsymbol{pp}}P_{ij} & \boldsymbol{n}_{ij} \\ \boldsymbol{n}_{ij}^T & 0 \end{pmatrix}\begin{pmatrix} \delta\boldsymbol{p}_{ij} \\ \lambda_{ij} \end{pmatrix} = \begin{pmatrix} \nabla_{\boldsymbol{p}}P_{ij} \\ 0 \end{pmatrix},$$

where we abuse notation and add a pending zero to $\boldsymbol{n}_{ij}$ when necessary. Suppose the coefficient matrix is invertible, we can solve for $\delta\boldsymbol{p}_{ij}$ as:

$$\delta\boldsymbol{p}_{ij} = \boldsymbol{H}_{ij}\left(\nabla_{\boldsymbol{p}}P_{ij} - \nabla_{\boldsymbol{p\theta}}P_{ij}\delta\boldsymbol{\theta}\right)$$
$$\boldsymbol{H}_{ij} \triangleq \nabla_{\boldsymbol{pp}}P_{ij}^{-1}\left[\boldsymbol{I} - \boldsymbol{n}_{ij}\boldsymbol{n}_{ij}^T\frac{\nabla_{\boldsymbol{pp}}P_{ij}^{-1}}{\boldsymbol{n}_{ij}^T\nabla_{\boldsymbol{pp}}P_{ij}^{-1}\boldsymbol{n}_{ij}}\right], \tag{3}$$

where we have used Gauss elimination to factor out λ_{ij} assuming $\nabla_{\boldsymbol{pp}}P_{ij}$ is invertible. Next, we plug all the $\delta\boldsymbol{p}_{ij}$ into the first equation to yield the following Schur-complement system of $\boldsymbol{\theta}$ alone:

$$\boldsymbol{H}_{\boldsymbol{\theta}}\delta\boldsymbol{\theta} = \left[\nabla_{\boldsymbol{\theta}}\bar{\mathcal{O}} - \sum_{ij\in\mathcal{C}}\nabla_{\boldsymbol{\theta p}}P_{ij}\boldsymbol{H}_{ij}\nabla_{\boldsymbol{p}}P_{ij}\right] \tag{4}$$
$$\boldsymbol{H}_{\boldsymbol{\theta}} \triangleq \left[\nabla_{\boldsymbol{\theta\theta}}\bar{\mathcal{O}} - \sum_{ij\in\mathcal{C}}\nabla_{\boldsymbol{\theta p}}P_{ij}\boldsymbol{H}_{ij}\nabla_{\boldsymbol{p\theta}}P_{ij}\right],$$

which is then solved for $\boldsymbol{\theta}$. Notably, the above procedure has a complexity linear in the number of hyper planes, which is much faster than using an off-the-shelf solver to factorize the large KKT matrix in Equation 2. However, the well-definedness of the above system relies on the inversion of the matrix $\nabla_{\boldsymbol{pp}}P_{ij}$ and $\boldsymbol{H}_{\boldsymbol{\theta}}$. Using the Schur-complement lemma [26], we have the following corollary (proved in appendix):

Corollary IV.1. *If $\nabla^2\bar{\mathcal{O}} \succeq \epsilon\boldsymbol{I}$, then the Schur-complementary solver is well-defined and $\boldsymbol{H}_{\boldsymbol{\theta}} \succeq \epsilon\boldsymbol{I}$.*

In practice, however, the strict requirement of $\nabla^2\bar{\mathcal{O}} \succeq \epsilon\boldsymbol{I}$ may not be met, and we introduce a diagonal perturbation to ensure well-definedness. Specifically, we employ an adjustment function $\mathcal{A}(\boldsymbol{H},\epsilon)$ that ensures all eigenvalues of $\boldsymbol{H}$ to be greater than a specified ϵ. This adjustment is applied to both $\nabla_{\boldsymbol{pp}}P_{ij}$ and $\boldsymbol{H}_{\boldsymbol{\theta}}$. The primary challenge in this adjustment process is the eigen decomposition. Thankfully, this operation is computationally efficient due to the relatively small matrix sizes involved. Following the computation of $\boldsymbol{\delta}$, we utilize a line-search strategy to find a step size that strictly decreases the objective function. Finally, we re-normalize $\boldsymbol{n}_{ij}$ after the update. The algorithm pipeline for the ECB method is outlined in Algorithm 1.

Algorithm 1 ECB Method

Input: Initial $\boldsymbol{\theta}, \boldsymbol{p}_{ij}, \mathcal{C}, \epsilon_1, \epsilon_2, \epsilon$
Output: Locally optimal $\boldsymbol{\theta}$
1: **for** $\|\nabla\bar{\mathcal{O}}\|_\infty > \epsilon$ **do**
2:      Compute derivatives in Equation 2
3:      **for** $ij \in \mathcal{C}$ **do**
4:          $\nabla_{\boldsymbol{pp}}P_{ij} \leftarrow \mathcal{A}(\nabla_{\boldsymbol{pp}}P_{ij},\epsilon_1)$
5:          Form complement system for $\boldsymbol{p}_{ij}$ (Equation 3)
6:      Form complement system for $\boldsymbol{\theta}$ (Equation 4)
7:      $\boldsymbol{H}_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(\boldsymbol{H}_{\boldsymbol{\theta}},\epsilon_2)$ then solve for $\delta\boldsymbol{\theta}$
8:      **for** $ij \in \mathcal{C}$ **do**
9:          Plug $\delta\boldsymbol{\theta}$ in Equation 3 for $\delta\boldsymbol{p}_{ij}$
10:     Use line-search procedure to find step size α
11:     Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{\theta}$
12:     **for** $ij \in \mathcal{C}$ **do**
13:        Update $\boldsymbol{p}_{ij} \leftarrow \boldsymbol{p}_{ij} + \alpha\delta\boldsymbol{p}_{ij}$, $\boldsymbol{n}_{ij} \leftarrow \boldsymbol{n}_{ij}/\|\boldsymbol{n}_{ij}\|$

B. ICB Method

The primary computational challenge encountered in the ECB method is the inversion of the large Hessian matrix.

However, many variables correspond to separating planes, which are intermediary variables unrelated to the robot's motion itself. Hence, it is advantageous to eliminate these variables, leaving only $\boldsymbol{\theta}$ as the independent variable. In this section, we demonstrate that it is indeed possible to express $\boldsymbol{p}_{ij}$ as a well-defined, sufficiently smooth function of $\boldsymbol{\theta}$. For this reason, we refer to this method as the ICB method. To initiate this discussion, we observe that each plane $\boldsymbol{p}_{ij}$ only appears in the collision penalty terms. Therefore, the optimal $\boldsymbol{p}_{ij}$ should satisfy the following optimization problem:

$$\boldsymbol{p}_{ij} \in \underset{\boldsymbol{n}_{ij}, d_{ij}}{\operatorname{argmin}} P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij}) \quad \text{s.t.} \ \|\boldsymbol{n}_{ij}\| = 1.$$

However, the function defined above does not guarantee a well-defined $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ because the solution may not be unique due to the additional non-convex unit-norm constraint. To obtain a well-defined function $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$, we turn to an equivalent formulation based on the Support Vector Machine (SVM) [28]. This formulation replaces the non-convex constraint $\|\boldsymbol{n}_{ij}\| = 1$ with a relaxed convex constraint $\|\boldsymbol{n}_{ij}\| \leq 1$, giving the following alternative definition:

$$\boldsymbol{p}_{ij} \in \underset{\boldsymbol{n}_{ij}, d_{ij}}{\operatorname{argmin}} P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij}) \quad \text{s.t.} \ \|\boldsymbol{n}_{ij}\| \leq 1.$$

It is evident that the above optimization is convex. However, convexity alone is insufficient to express $\boldsymbol{p}_{ij}$ as a function of $\boldsymbol{\theta}$ because there may not be a unique minimizer. Even if a unique solution exists, the resulting implicit function $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ may not be differentiable. As pointed out in [27], general convex optimization problems only have sub-gradients. Therefore, we introduce the following smoothed optimization amenable to differentiability, employing our log-barrier function $P(\bullet)$ to handle the additional convex constraint:

$$\boldsymbol{p}_{ij}(\boldsymbol{\theta}) \triangleq \underset{\boldsymbol{n}_{ij}, d_{ij}}{\operatorname{argmin}} P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij}) + P(1 - \|\boldsymbol{n}_{ij}\|). \tag{5}$$

We have the several essential properties for Equation 5 as summarized below (proved in appendix):

Corollary IV.2. *Assuming C_i and C_j are closed set, C_i or C_j has non-zero volume, i.e. $|C_i| \neq 0$ or $|C_j| \neq 0$, and $P(\bullet)$ is third-order differentiable, we have the following properties: 1) Equation 5 has a solution iff $C_i(\boldsymbol{\theta}) \cap C_j(\boldsymbol{\theta}) = \varnothing$; 2) $\boldsymbol{p}_{ij}$ is a well-defined function of $\boldsymbol{\theta}$; 3) $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ is twice-differentiable; 4) $\lim_{dist(C_i, C_j) \to 0^+} P_{ij}(\boldsymbol{\theta}, \boldsymbol{p}_{ij}(\boldsymbol{\theta})) = \infty$, with $dist(\bullet, \bullet)$ being the distance between convex polytopes.*

Corollary IV.2 ensures the differentiable structure of $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ as long as one of the convex sets is non-degenerate. This limitation is not significant for robotic applications because robot links practically always have non-zero volume. With this structure, we can define an implicit optimization approach and solve the following optimization problem:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \tilde{\mathcal{O}}(\boldsymbol{\theta}) \triangleq \bar{\mathcal{O}}(\boldsymbol{\theta}, \cdots, \boldsymbol{p}_{ij}(\boldsymbol{\theta}), \cdots), \tag{6}$$

replacing the explicit variable $\boldsymbol{p}_{ij}$ with the implicit function $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$. Corollary IV.2 3) ensures that Equation 6 has the twice-differentiable objective function and Corollary IV.2 4) ensures that the feasibility of Equation 6 implies the collision-free property. To evaluate such objective function at given $\boldsymbol{\theta}$,

---

Algorithm 2 ICB Method

Input: Initial $\boldsymbol{\theta}, \boldsymbol{p}_{ij}, \mathcal{C}, \epsilon_1, \epsilon$
Output: Locally optimal $\boldsymbol{\theta}$
1: **for** $\|\nabla\tilde{\mathcal{O}}\|_\infty > \epsilon$ **do**
2:      **for** $ij \in \mathcal{C}$ **do**
3:          $\boldsymbol{p}_{ij}(\boldsymbol{\theta}) \leftarrow$ solution to Equation 5
4:          Compute derivatives of $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ via Equation 7
5:      Form $\nabla\tilde{\mathcal{O}}$ and $\nabla^2\tilde{\mathcal{O}}$
6:      Adjust $\nabla^2\tilde{\mathcal{O}} \leftarrow \mathcal{A}(\nabla^2\tilde{\mathcal{O}}, \epsilon_1)$
7:      Form search direction $\delta\boldsymbol{\theta} \leftarrow -\nabla^{-2}\tilde{\mathcal{O}}\nabla\tilde{\mathcal{O}}$
8:      Use line-search procedure to fine step size α
9:      Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{\theta}$

---

we need to solve Equation 5 for every pair of convex sets in close proximity. After the solution, we can evaluate the first- and second-order derivative of $\boldsymbol{p}_{ij}(\boldsymbol{\theta})$ and $\tilde{\mathcal{O}}$ (appendix). The computational complexity of these equations is linear with respect to the number of separating planes, similar to our explicit algorithm. However, solving Equation 5 for each pair of convex sets may initially appear computationally burdensome. Fortunately, we can expedite the process by employing a warm-start strategy, where we retain the solution from the previous iteration. If this warm-start approach fails (for instance, if the solution from the last iteration is infeasible), we can establish a feasible initial estimate by computing the closest pair of points on C_i and C_j. We then use this information to determine the initial separating plane, setting it as the middle sectioning plane. In practice, we can solve the subproblems in Equation 5 with just a few Newton iterations, and all of these subproblems can be solved in parallel. The algorithmic workflow of the ICB method is summarized in Algorithm 2. Since the Hessian matrix of $\tilde{\mathcal{O}}$ remains relatively small, we can calculate its dense eigen-decomposition, which is essential for matrix inversion and performing the positive-definite adjustment. This adjustment ensures that the minimal eigenvalue of the Hessian matrix is at least ϵ_1.

	$|\boldsymbol{\theta}|$	$|\mathcal{C}|$		
		ECB	ICB	AO
(a)	54	50	49	50
(b)	108	716	570	714
(c)	24	1757	1024	1266
(d)	36	3099	3585	2444
(e)	54	6424	11122	6892
(f)	33	973	837	899

TABLE II: The number of degrees of freedom $|\boldsymbol{\theta}|$ and separating planes $|\mathcal{C}|$ for each benchmarking scenario.

V. Evaluation

We conducted our experiments using C++ on a single desktop machine equipped with an 8-core AMD EPYC 7K62 CPU. All available cores were utilized to facilitate parallel computations related to separating planes, such as the computation of per-plane Schur-complements in the ECB method and the evaluation of implicit functions and their derivatives in the ICB method. We also incorporated the widely used
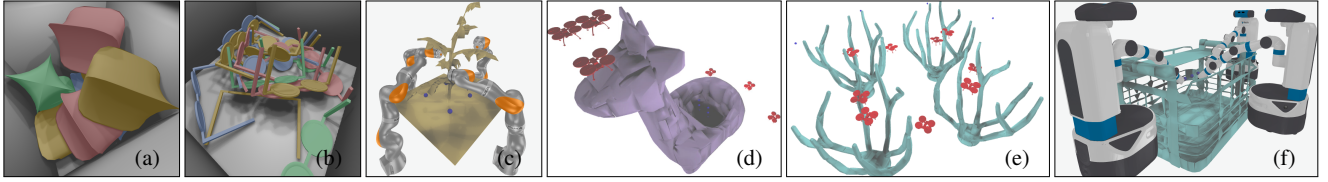
Fig. 2: The six scenarios used for evaluating the three methods: bulky object settling (a), thin object settling (b), quadruple-arm reaching (c), UAV trajectory generation (d), long-distance UAV trajectory generation (e), basket-reaching (f).

	ECB		ICB		AO	
Scenario	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$
(a)	697	26906	337	21610	24182	N.A.
(b)	6898	10329	1470	13245	42000	79300
(c)	424	2356	132	525	371	N.A.
(d)	1068	1987	406	623	1983	5010
(e)	1802	2936	1115	1396	2305	6773
(f)	219	1066	159	400	277	N.A.

TABLE III: The number of iterations for the three algorithms to reach difference level of gradient norm ϵ (N.A. denotes the algorithm cannot finish computation within the time limit).

	ECB		ICB		AO	
Scenario	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$
(a)	0.04	1.70	0.02	1.44	1.14	N.A.
(b)	3.58	5.40	0.74	7.54	20.63	38.56
(c)	2.58	38.05	0.76	5.51	1.78	N.A.
(d)	13.44	30.80	4.99	9.56	27.28	81.06
(e)	107.88	222.79	76.74	109.99	161.08	679.28
(f)	2.57	26.73	1.72	7.71	4.47	N.A.

TABLE IV: The computational time (min) for the three algorithms to reach difference level of gradient norm ϵ.

acceleration technique of broadphase collision checks, similar to the one employed in prior works [9, 11]. This technique excludes distant pairs of convex hulls from consideration, introducing barrier terms only when the convex hulls are in close proximity. It is important to note that once inserted, these barrier terms are not removed, even if the pair of convex hulls later move apart. Throughout our analysis, we require our penalty function to be positive and strictly convex. One function satisfying our needs is the layered penalty function [29], which is used in our implementation. Finally, we use hyper-parameters $\epsilon_1 = 0.001$ and $\epsilon_2 = 0.001$ throughout our experiements.

Additionally, we integrated our solver with an implementation of the SIP solver from [10]. This allows us to solve SIP problem instances, ensuring collision-free trajectories at every continuous time interval, by leveraging their constraint discretization method. For comparison, both the ECB and ICB methods are compared with the baseline Alternating Optimization (AO) technique [11], which optimizes $\boldsymbol{\theta}$ and $\boldsymbol{p}_{ij}$ in separate subproblems, where we also optimize $\boldsymbol{p}_{ij}$ subproblems in parallel. This is state-of-the-art optimization technique, allowing the optimization of arbitrary articulated robot poses under convex constraints, with the only difference being the convergence speed.

We assess the performance of our method using a series of six challenging benchmarks, as outlined in Figure 2. The first two benchmarks involve object settling problems, with bulky objects (a) and thin objects (b) dropped into a box.

These problems require us to compute the static equilibrium poses of the objects. To this end, we minimize the gravitational potential energy as our objective function $\mathcal{O}$. Since we optimize a single pose, these scenarios only involve a finite set of collision constraints. The remaining four scenarios focus on trajectory optimization problems, where we solve SIP problem instances by integrating both our methods and AO [11] into the SIP solver framework [10]. For all four examples, our objective function $\mathcal{O}$ involves a distance-to-target cost term and a Laplacian trajectory smoothness cost. In Figure 2 (c), four robot arms reach various end-effector positions on a table, while avoiding obstacles. In Figure 2 (d), we optimize the trajectories of six UAVs with a giraffe-shaped obstacle in the middle. Figure 2 (e) extends this challenge with longer trajectories for the nine UAVs and multiple obstacles along their paths. Finally, Figure 2 (f) presents three fetch robots reaching 3 positions in a basket. In these scenarios, the trajectories for each robot's configuration space are parameterized using high-order spline curves, with the number of degrees of freedom and separating planes summarized in Table II, which shows that our method can scale to complex high-DOF robots and environmental setups. We measure the number of iterations and computational costs required to reach different gradient norm values, and the results are summarized in Table III and Table IV, respectively. We further plot the evolution of energy values overtime for two of our results in Figure 3. Note that as local optimization methods, different techniques can converge to different solutions and energy values.
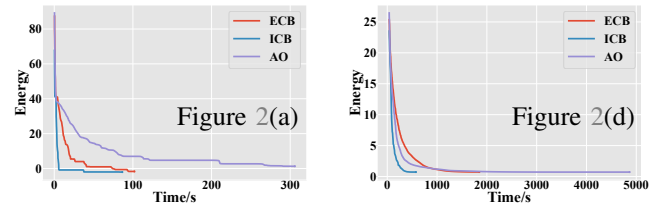

Fig. 3: We plot the energy value evolution for two of the examples in Figure 2.

Our results clearly demonstrate the superior convergence speed of our method compared to AO, especially when aiming for highly optimal motion plans with a small threshold value (ϵ). The key factor behind this superior performance is the significantly lower number of iterations required by our method, while maintaining a comparable per-iteration cost. Furthermore, it is worth noting that ICB exhibits faster convergence compared to ECB, despite both methods being

locally second-order. This can be attributed to ICB's approach of computing the optimal separating planes during each iteration. Although this leads to a slightly higher iterative cost, the overall reduction in the number of iterations more than compensates for this additional computational overhead. In summary, our method consistently achieves a remarkable reduction in computational time, with a speedup ranging from 5.12 to 206.34 for pose optimization (Figure 2a-b) and 2.63 to 151.23 for trajectory optimization (Figure 2c-f).

VI. Conclusion & Limitations

We present innovative second-order optimization-based planners for both robot poses and trajectories, building upon the convex constraint formulation introduced by [13, 11]. A key distinguishing feature of our approach is the efficient application of Newton's method to simultaneously optimize the separating planes and robot poses. Our work has paved the way for several promising avenues in future research. ICB is more efficient than ECB, requiring even fewer iterations to converge. However, it demands a more complex implementation with an intricate inner loop of optimization and a careful treatment of second-order derivatives. The reliable evaluation of these derivatives under finite-precision floating-point formats requires further investigation. Moreover, due to the inherent complexity of the underlying problem, our method may still be time-consuming. In addition, our method can be extended to incorporate robot dynamics by utilizing differential flatness such as [30], i.e. formulating the flat variables as decision variables. Finally, our method is a form of local optimization, which suffers from sub-optimal or even useless solutions, i.e., our method cannot explore multiple homotopy classes to find the optimal solution. This issue can be alleviated by combining our method with search-based methods or multi-start optimization.

References

[1] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 42–49.

[2] Y. Chua, K. P. Tee, and R. Yan, "Robust optimal inverse kinematics with self-collision avoidance for a humanoid robot," in *2013 IEEE RO-MAN*, 2013, pp. 496–502.

[3] K. Hauser, "Semi-infinite programming for trajectory optimization with non-convex obstacles," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1106–1122, 2021.

[4] C. Khazoom, D. Gonzalez-Diaz, Y. Ding, and S. Kim, "Humanoid self-collision avoidance using whole-body control with control barrier functions," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, IEEE, 2022, pp. 558–565.

[5] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 3681–3688.

[6] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 4569–4574.

[8] M. Zucker *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[9] M. Li *et al.*, "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics.," *ACM Trans. Graph.*, vol. 39, no. 4, p. 49, 2020.

[10] D. Zhang, X. Gao, K. Wu, and Z. Pan, "Provably robust semi-infinite program under collision constraints via subdivision," *arXiv preprint arXiv:2302.01135*, 2023.

[11] R. Ni, Z. Pan, and X. Gao, "Robust multi-robot trajectory optimization using alternating direction method of multiplier," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5950–5957, 2022.

[12] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A strictly convex hull for computing proximity distances with continuous gradients," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 666–678, 2014.

[13] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[15] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[16] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[17] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[18] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2022, pp. 328–348.

[19] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science Robotics*, vol. 8, no. 84, eadf7843, 2023.

[20] 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.

[21] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[22] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2015, pp. 109–124.

[23] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[24] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[25] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, vol. 72, no. 2, pp. 700–734, 2017.

[26] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[27] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[28] P.-H. Chen, C.-J. Lin, and B. Schölkopf, "A tutorial on ν-support vector machines," *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.

[29] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous contact mechanics," in *ACM SIGGRAPH 2009 papers*, 2009, pp. 1–12.

[30] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 2520–2525.

[31] D. Carlson, "What are schur complements, anyway?" *Linear Algebra and its Applications*, vol. 74, pp. 257–275, 1986.

APPENDIX: ADDITIONAL PROOFS

Proof of Corollary IV.1. $\nabla^2 \bar{\mathcal{O}}$ is positive definite and so is all its principal submatrices. As a result, $\nabla_{pp} P_{ij}$ is invertible and $n_{ij}^T \nabla_{pp} P_{ij}^{-1} n_{ij} \neq 0$ so H_{ij} is well-defined. We further rewrite:

$$H_{ij} = \nabla_{pp} P_{ij}^{-\frac{1}{2}} \left[I - \bar{n}_{ij} \bar{n}_{ij}^T \right] \nabla_{pp} P_{ij}^{-\frac{1}{2}}$$

$$\bar{n}_{ij} \triangleq \nabla_{pp} P_{ij}^{-\frac{1}{2}} n_{ij} / \| \nabla_{pp} P_{ij}^{-\frac{1}{2}} n_{ij} \|,$$

from which we immediately have $\nabla_{pp} P_{ij}^{-1} \succeq H_{ij} \succeq 0$. By Guttman rank additivity [31], we have the following matrix is positive definite:

$$\bar{H}_{\theta} \triangleq \left[\nabla_{\theta\theta} \bar{\mathcal{O}} - \sum_{ij \in \mathcal{C}} \nabla_{p\theta} P_{ij} \nabla_{pp} P_{ij}^{-1} \nabla_{p\theta} P_{ij} \right].$$

Combined with the fact that $\nabla_{pp} P_{ij}^{-1} \succeq H_{ij}$, we have: $H_{\theta} \succeq \bar{H}_{\theta} > 0$. Note the above argument only requires $\nabla^2 \bar{\mathcal{O}} \succeq 0$. If we further have: $\nabla^2 \bar{\mathcal{O}} \succeq \epsilon I$, then we can apply the above argument to the perturbed matrix:

$$\begin{pmatrix} \nabla^2 \bar{\mathcal{O}} & J \\ J^T & 0 \end{pmatrix} - \begin{pmatrix} \epsilon I & \\ & 0 \end{pmatrix},$$

which immediately yields: $H_{\theta} \succeq \bar{H}_{\theta} \succeq \epsilon I$. $\qquad \square$

Proof of Corollary IV.2. 1) If Equation 5 is feasible, then $n_{ij} \neq 0$ because otherwise we have $P(-d_{ij}) + P(d_{ij}) < \infty$. But this implies $d_{ij} \in (0, \infty)$ and $-d_{ij} \in (0, \infty)$ which is impossible. Therefore, we can verify that $n_{ij} / \| n_{ij} \|, d_{ij} / \| n_{ij} \|$ is a plane separating C_i and C_j. Conversely, if $C_i \cap C_j = \varnothing$, then dist$(C_i, C_j) > 0$, i.e., we have a separating plane n_{ij} such that $-n_{ij}^T x(\theta) - d_{ij} > 0$ for all $x(\theta) \in C_i$ and $n_{ij}^T x(\theta) + d_{ij} > 0$ for all $x(\theta) \in C_j$. We can then choose $\alpha \in (0, 1)$ sufficiently close to 1 such that: $-\alpha n_{ij}^T x(\theta) - d_{ij} > 0$ for $x(\theta) \in C_i$ and $\alpha n_{ij}^T x(\theta) + d_{ij} > 0$ for $x(\theta) \in C_j$. And we can immediately verify that $p_{ij} = \left(\alpha n_{ij}^T, d_{ij}^T \right)^T$ is a feasible solution of Equation 5.

2) This property essentially requires the uniqueness of minimizer. To establish uniqueness, we can write the Hessian of the objective function of Equation 5 as:

$$\tilde{H}_{ij}(\theta, p_{ij}) \triangleq \nabla^2_{p_{ij}} \left[P_{ij}(\theta, p_{ij}) + P(1 - \| n_{ij} \|) \right]$$

$$= \sum_{x \in C_i} P''(-n_{ij}^T x(\theta) - d_{ij}) \begin{pmatrix} x(\theta) \\ 1 \end{pmatrix} \left(x(\theta)^T, 1 \right) +$$

$$\sum_{x \in C_j} P''(n_{ij}^T x(\theta) + d_{ij}) \begin{pmatrix} x(\theta) \\ 1 \end{pmatrix} \left(x(\theta)^T, 1 \right) + \nabla^2_{p_{ij}} \left[P(1 - \| n_{ij} \|) \right].$$

We show that $\tilde{H}_{ij}$ must be a matrix of full-rank. Note that $P''(-n_{ij}^T x(\theta) - d_{ij}) > 0$ and $P''(n_{ij}^T x(\theta) + d_{ij}) > 0$ due to the strong convexity of P. Next, note that $P(1 - \| n_{ij} \|)$ is a convex function because it is a composite of a convex function $-1 + \| n_{ij} \|$ and a non-decreasing function $P(-\bullet)$ [26]. Further, we have the top-left entry of Hessian taking the following form:

$$\frac{\partial^2 P(1 - \| n_{ij} \|)}{\partial n_{ij,x}^2} = \frac{n_{ij,y}^2 + n_{ij,z}^2}{\| n_{ij} \|^3 (1 - \| n_{ij} \|)} + \frac{n_{ij,x}^2}{\| n_{ij}^2 \| (1 - \| n_{ij} \|)} > 0.$$

This implies the Hessian of $P(1 - \| n_{ij} \|)$ is a positive semi-definite, non-zero matrix, so it has the following diagonaliza-

tion:

$$\frac{\partial^2 P(1 - \| n_{ij} \|)}{\partial n_{ij}^2} = V \Sigma V^T,$$

where $0 \neq \Sigma \geq 0$ and V is a orthogonal matrix. Putting all these facts together, we have:

$$\tilde{H}_{ij} = U U^T$$

$$U \triangleq \left(U_1, U_2, U_3 \right)$$

$$U_1 \triangleq \left(\cdots, \sqrt{P''(-n_{ij}^T x(\theta) - d_{ij})} \begin{pmatrix} x(\theta) \in C_i \\ 1 \end{pmatrix}, \cdots \right)$$

$$U_2 \triangleq \left(\cdots, \sqrt{P''(n_{ij}^T x(\theta) + d_{ij})} \begin{pmatrix} x(\theta) \in C_j \\ 1 \end{pmatrix}, \cdots \right)$$

$$U_3 \triangleq \begin{pmatrix} V_1 & V_2 & V_3 \\ 0 & 0 & 0 \end{pmatrix} \sqrt{\Sigma}.$$

Without a loss of generality, we can assume $|C_i| \neq 0$ and $\Sigma_{11} > 0$. Then $x(\theta) \in C_i$ spans $\mathbb{R}^3$ and hence the following vectors span $\mathbb{R}^4$:

$$\left(\cdots, \sqrt{P''(-n_{ij}^T x(\theta) - d_{ij})} \begin{pmatrix} x(\theta) \in C_i \\ 1 \end{pmatrix}, \cdots, \begin{pmatrix} V_1 \sqrt{\Sigma_{11}} \\ 0 \end{pmatrix} \right).$$

Put together, we have U spans $\mathbb{R}^4$ and $\tilde{H}_{ij} = U U^T > 0$. Due to the arbitary of p_{ij}, we have Equation 5 is a strictly convex optimization and p_{ij} is a unique minimizer.

3) Due to the strict convexity, $p_{ij}(\theta)$ is equivalently defined as the solution of:

$$\tilde{G}_{ij}(\theta, p_{ij}) \triangleq \nabla_{p_{ij}} \left[P_{ij}(\theta, p_{ij}) + P(1 - \| n_{ij} \|) \right] = 0.$$

Since $\tilde{H}_{ij}(\theta, p_{ij})$ has full-rank, we can then invoke the high-order implicit function theorem to see that $p_{ij}(\theta)$ is a twice-differentiable function of θ if the barrier $P(\bullet)$ is third-order differentiable.

4) Denote $d \triangleq $ dist$(C_i, C_j) > 0$, then for every separating plane p_{ij} with $\| n_{ij} \| = 1$, we have $-n_{ij}^T x(\theta) - d_{ij} < d$ for some $x(\theta) \in C_i$ and $n_{ij}^T x(\theta) + d_{ij} < d$ for some $x(\theta) \in C_j$ because otherwise dist$(C_i, C_j) > d$. Now the solution to Equation 5 can be denoted as a generalized separating plane with $\| n_{ij} \| < 1$, so we have $-n_{ij}^T x(\theta) - d_{ij} < d \| n_{ij} \| < d$ for some $x(\theta) \in C_i$ and $n_{ij}^T x(\theta) + d_{ij} < d \| n_{ij} \| < d$ for some $x(\theta) \in C_j$. As a result we have: $\lim_{d \to 0^+} P_{ij}(\theta, p_{ij}(\theta)) \geq \lim_{d \to 0^+} P(d) = \infty$. $\qquad \square$

APPENDIX: DERIVATIVES

We use Einstein's notation with scripts α, β, γ:

$$\frac{\partial p_{ij}(\theta)}{\partial \theta^\alpha} = -\tilde{H}_{ij}^{-1} \frac{\partial \tilde{G}_{ij}}{\partial \theta^\alpha}$$

$$\frac{\partial^2 p_{ij}(\theta)}{\partial \theta^\alpha \partial \theta^\beta} = -\tilde{H}_{ij}^{-1} \Big[\frac{\partial^2 \tilde{G}_{ij}}{\partial \theta^\alpha \partial \theta^\beta} + \frac{\partial \tilde{H}_{ij}}{\partial \theta^\alpha} \frac{\partial p_{ij}(\theta)}{\partial \theta^\beta} +$$

$$\frac{\partial \tilde{H}_{ij}}{\partial \theta^\beta} \frac{\partial p_{ij}(\theta)}{\partial \theta^\alpha} + \frac{\partial \tilde{H}_{ij}}{\partial p_{ij}^\gamma} \frac{\partial p_{ij}(\theta)}{\partial \theta^\alpha} \frac{\partial p_{ij}^\gamma(\theta)}{\partial \theta^\beta} \Big]$$

$$\frac{d\tilde{\mathcal{O}}}{d\theta} = \frac{\partial \tilde{\mathcal{O}}}{\partial \theta} + \sum_{ij \in \mathcal{C}} \frac{\partial \tilde{\mathcal{O}}}{\partial p_{ij}} \frac{\partial p_{ij}(\theta)}{\partial \theta} \qquad \frac{d^2 \tilde{\mathcal{O}}}{d\theta^2} = \frac{\partial^2 \tilde{\mathcal{O}}}{\partial \theta^2} + \qquad (7)$$

$$\sum_{ij \in \mathcal{C}} \Big[\frac{\partial^2 \tilde{\mathcal{O}}}{\partial \theta \partial p_{ij}} \frac{\partial p_{ij}(\theta)}{\partial \theta} + \frac{\partial p_{ij}(\theta)}{\partial \theta}^T \frac{\partial^2 \tilde{\mathcal{O}}}{\partial p_{ij} \partial \theta} +$$

$$\frac{\partial p_{ij}(\theta)}{\partial \theta}^T \frac{\partial^2 \tilde{\mathcal{O}}}{\partial p_{ij}^2} \frac{\partial p_{ij}(\theta)}{\partial \theta} + \frac{\partial \tilde{\mathcal{O}}}{\partial p_{ij}^\alpha} \frac{\partial^2 p_{ij}^\alpha(\theta)}{\partial \theta^2} \Big].$$