IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

SDPRLayers: Certifiable Backpropagation Through Polynomial Optimization Problems in Robotics

Connor Holmes, *Graduate Student Member, IEEE*, Frederike Dümbgen, *Member, IEEE*, and Timothy D. Barfoot, *Fellow, IEEE*

***Abstract*—A recent set of techniques in the robotics community, known as *certifiably correct methods*, frames robotics problems as *polynomial optimization problems* and applies convex, semidefinite programming (SDP) relaxations to either find or certify their global optima. In parallel, *differentiable optimization* allows optimization problems to be embedded into end-to-end learning frameworks and has received considerable attention in the robotics community. In this article, we consider the ill effect of convergence to spurious local minima in the context of learning frameworks that use differentiable optimization. We present SDPRLayers, an approach that seeks to address this issue by combining convex relaxations with implicit differentiation techniques to provide *certifiably correct solutions and gradients* throughout the training process. We provide theoretical results that outline conditions for the correctness of these gradients and provide efficient means for their computation. Our approach is first applied to two simple-but-demonstrative simulated examples, which expose the potential pitfalls of reliance on local optimization in existing, state-of-the-art, differentiable optimization methods. We then apply our method in a real-world application: we train a deep neural network to detect image keypoints for robot localization in challenging lighting conditions. We provide our open-source, PyTorch implementation of SDPRLayers and our differentiable localization pipeline.**

***Index Terms*—Certifiable methods, differentiable optimization, visual learning.**

I. Introduction

THE versatility of learning models has made them ubiquitous in robotics, permeating almost all layers of the modern software stack [71]. On the other hand, model-based optimization—the mainstay of traditional robotics—provides a level of robustness, accuracy, and generalization that has proven difficult to match by learning-based methods [59]. A recent paradigm, leveraging advances in so-called *differentiable optimizers*, now enables roboticists to combine these two approaches into a single *end-to-end learning* framework.

In this approach, optimization problems are embedded as "layers" in deep-learning networks, with optimization parameter data as the input and the optimal solution as the output. Similar to other layers in machine learning, the forward pass of the layer involves solving the optimization, while the backward pass computes the gradients.

The benefits of this approach are several, allowing practitioners to capitalize on the respective advantages of model-based and learning approaches while also mitigating their disadvantages. For example, this integration means that domain knowledge can be injected directly into the robotics pipeline, while still allowing machine learning parameters to be trained on the final goal of a robotics module. This also obviates costly integration and fine-tuning stages that are necessary when learning and optimization modules are developed in parallel [71].

Despite its recent successes, there is a fundamental issue with differentiable optimization that has gone unaddressed in the literature; the majority of optimization problems in robotics are nonconvex and, hence, prone to convergence to local optima, rather than a global optimum. In particular, if the optimization layer converges to spurious local minima, the gradients that are backpropagated through the network will not correspond to the true global optimum, and can, therefore, corrupt the training/optimization process.[1] We will see that this can have serious consequences, leading not only to longer training times, but also outright failure to meet objectives of the training process.

In this article, we address this issue by taking inspiration from a recent body of work in the robotics and vision communities that deals with so-called *certifiably correct methods* [61, Sec. 2.2] (also called *certifiable algorithms* [76]). These methods use convex semidefinite relaxations of nonconvex, polynomial optimization problems (POPs) to either directly find a global optimum or provide a certificate of global optimality for a given solution. Certifiably correct methods originate in the robotics

Received 24 December 2024; revised 18 April 2025; accepted 2 June 2025. Date of publication 9 June 2025; date of current version 9 July 2025. The work of Frederike Dümbgen was supported in part by the Swiss National Science Foundation, Postdoc Mobility under Grant 206954, and in part by the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie under Grant 101207106. This work was supported in part by the National Sciences and Engineering Research Council of Canada (NSERC). This article was recommended for publication by Associate Editor M. Posa and Editor M. Schwager upon evaluation of the reviewers' comments. *(Corresponding author: Connor Holmes.)*

Connor Holmes and Timothy D. Barfoot are with the University of Toronto Robotics Institute,, ON M6P2T4, Canada, and also with the University of Toronto, Toronto, ON M6P2T4, Canada (e-mail: connor.holmes@mail.utoronto.ca).

Frederike Dümbgen is with the Inria, École Normale Supérieure, PSL University, 94270 Paris, France.

Code for SDPRLayer available online at: https://github.com/utiasASRL/sdprlayer Code for differentiable localization pipeline available online at: https://github.com/utiasASRL/deep_learned_visual_features/tree/mat-weight-sdp-version.

Digital Object Identifier 10.1109/TRO.2025.3578228

[1] We acknowledge that, in some cases, finding *any* stationary point may be sufficient for the objectives of the practitioner and the associated gradients may in fact be desirable. In this work, we take the stance that the goal of an optimization problem is to find the *global* optimum and that convergence to local minima is undesirable.

community, but are closely related to the Moment-sums-of-squares (SOS) hierarchy, which seeks to solve POPs using an *automated hierarchy* of SDP relaxations [36]. However, since the goal certifiably correct methods is application in robotics, they tend to focus on bespoke relaxations that are tailored toward *efficiently* solving or certifying solutions.

Similar to the Moment-SOS hierarchy, our approach is to directly solve the semidefinite relaxations of robotics problems. When the solution to the relaxation can be directly used to obtain a globally optimal solution for the original POP, the relaxation is said to be *tight*.[2] Even when the relaxation solution is not tight, it can still provide useful information about the original problem (see Section V-C).

A. Contributions

We focus on providing *differentiable, globally optimal solutions* to POPs. Although there are existing differentiable solvers that could potentially find and differentiate first-order critical points for this class of problems, the resulting derivatives may not correspond to the solution of the optimization problem (i.e., the global optimum). In general, such solvers are only guaranteed to find local optima.

As one of our key contributions, we provide our approach as an optimization layer, which we call the *SDPRLayer*[3] that conveniently encapsulates the parameterized POP. The layer is guaranteed to yield *certifiably correct, differentiable* solutions whenever the solution can be retrieved from the convex relaxation (i.e., the relaxation is tight) (see Fig.1 for a graphical overview of the method). We do this *efficiently* by leveraging the optimality conditions of the original, nonconvex problem. For nontight relaxations, we also provide the solution to the SDP, which we differentiate via existing, implicit-differentiation techniques for convex problems.

At the core of our approach is a theoretical result that provides conditions under which solution gradients can both be certified and efficiently computed. We view this result as a second key technical contribution.

Our layer can be easily embedded in end-to-end learning pipelines and bilevel optimization for robotics problems. As a third contribution, we use simulated and real-world examples to demonstrate the potential pitfalls of relying on local solvers in this context. Crucially, we demonstrate that if local minima are used for backpropagation, the resulting gradients can be quite different from those of the global minimum. We also show that the SDPRLayer can be used for large-scale training of a robotics perception pipeline.

[2]The astute reader may argue that in cases where strong duality holds at the solution, the solution can be certified without directly solving the SDP (e.g., scalar-weighted pose graph optimization [37], [60]). In our experience, the majority of robotics problems require the addition of so-called *redundant constraints* to "tighten" the relaxation (see below in Section III-B2 and Appendix C). However, when these constraints are used, certifying a solution amounts to solving an equivalent (dual form) SDP [25].

[3]The naming is based on CVXPYLayers [1] with SDPR standing for semidefinite programming relaxation.

B. Outline

In the following section (see Section II), we review works that are closely related to this article. In Section III, we review notation and relevant background material on implicit differentiation and semidefinite programming that may not be common knowledge to all readers, but that can be safely skipped by domain experts. Section IV provides the theoretical result that underpins our methodology, while Section V describes the implementation of our layer. In Section VI, we provide two examples that demonstrate the advantages of our approach and a third that demonstrates its application to a real-world robotics problem. Finally, Section VII concludes this article.

II. Related Work

A. Certifiably Correct Optimization in Robotics

As mentioned, we rely on advances in certifiably correct optimization methods, which provide guarantees on the global optimality of solutions. In robotics, this approach has been applied to robust state estimation [76], [77], sensor calibration [32], [74], inverse kinematics [32], image segmentation [40], rotation synchronization [19], [27], pose-graph optimization [7], [60], multiple-point-set registration [15], [41], range-only localization [24], [34], and range-aided simultaneous localization and mapping (SLAM) [55], among others. It is also worth noting that a similar approach using SOS polynomials has been applied in the nonlinear control community for automatic synthesis or verification of Lyapunov functions [46], [50]. We note that any problem to which certifiably correct methods can be applied (including all those stated above) can necessarily be formulated as a POP.

B. Differentiable Optimization in Robotics

New tools geared toward differentiable optimization in a robotics setting have been recently presented. Theseus, developed by Pineda et al. [57], presented a differentiable optimization layer for unconstrained, nonlinear least-squares problems that typically appear in robotics. Crucially, this tool built off work by Teed and Deng [68] to solve problems with lie group constraints. For trajectory optimization, CALIPSO provided differentiable solutions using an interior-point solver and was able to handle conic and complementarity constraints, which often occur when dealing with friction and contact in robotic motion [39]. Many of these methods *implicitly differentiate* the conditions of optimality, a technique which, to date, has proven to be the most accurate and efficient way to generate gradient information of an optimum [8].

Many roboticists have adopted differentiable optimization layers into their pipelines. Building off their seminal work in differentiating quadratic programs [4], Amos et al. [5] presented a differentiable model predictive control (MPC) framework that enabled learning of dynamics and objective functions. This work was subsequently extended to integrate reinforcement learning (RL) and safe learning into MPC [26], [59], [79]. In robot planning, several works have developed differentiable approaches

to dynamic programming [22], [42], [45]. Differentiable optimization has been used in physics-based simulators to enable efficient training of RL control strategies [18] or robotic hardware design [75]. In state estimation, DROID-SLAM, which introduced a differentiable, end-to-end, neural architecture for visual SLAM [67], is most notable, though there have been several works on differentiable SLAM [31], [43]. Differentiable optimization has also been used for factor-graph-based estimators [58] and smoothers [78], tactile sensing [65], and shape estimation [28].

The most closely related work to ours is SATNet, introduced by Wang et al. [72], which used semidefinite relaxations to generate approximate, differentiable solutions to the maximum satisfiability problem in combinatorial optimization. Amos et al. [5] also took a similar approach, applying a convex *quadratic* approximation to a nonconvex MPC problem. Our approach is a more general framework directed at providing *exact*, differentiable solutions to robotics problems. Talak et al. [66] also made use of certifiably correct methods to verify correctness of solutions, but did not directly differentiate the optimization problem.

C. Implicit Differentiation

A key element of modern differentiable optimization is so-called *implicit differentiation*, which applies the implicit function theorem (IFT) to the set of conditions—known as the Karush–Kuhn–Tucker (KKT) conditions—that characterize a local optimum [23]. The main idea is that the zero-level set of the KKT conditions defines an implicit function that uniquely relates the parameters to the primal solution. Under certain conditions, this relationship is differentiable, providing a means to back-propagate gradient information through optimization problems. Prior differentiation methods, such as unrolling, are either less accurate [8], slower, or more memory intensive than implicit differentiation [57].

Implicit differentiation can be applied to *any* local optimum satisfying the (second-order) KKT conditions [33]. When the problem is convex, implicit differentiation becomes much more powerful since the KKT conditions are sufficient and necessary for global optimality [10].[4] As such, it is often applied to convex problems or convex approximations of nonconvex problems [4], [18], [72]. One notable example is CVXPYLayers [1], a general differentiable solver for convex problems, which interfaces the popular CVXPY parser [21] with a differentiable conic optimization solver [2] and has become the de-facto standard for differentiating convex programs.

III. Background

A. Notation

We denote matrices with bold-faced, capitalized letters, $\boldsymbol{A}$, column vectors with bold-faced, lower-case letters, $\boldsymbol{a}$, and scalar quantities with normal-faced font, a. Let $\mathbb{S}^n$ (resp. $\mathbb{S}^n_+$) denote the space of n-dimensional symmetric (resp. positive, semidefinite) matrices, and $\mathbb{S}^n_1 \subset \mathbb{S}^n_+$ denote the set of positive, semidefinite, rank-1 matrices. We also use the conic notation $\boldsymbol{X} \succeq \boldsymbol{0}$ in place of $\boldsymbol{X} \in \mathbb{S}^n_+$. Let $\boldsymbol{I}$ denote the identity matrix, whose dimension will be clear from the context or otherwise specified. Let $\boldsymbol{0}$ denote the matrix with all-zero entries, whose dimension will be evident from the context. Let $\boldsymbol{A}^\dagger$ denote the Moore–Penrose pseudoinverse of a given matrix $\boldsymbol{A}$ and let $\ker(\boldsymbol{A})$ and $\mathrm{im}(\boldsymbol{A})$ be the kernel space and image space of $\boldsymbol{A}$, respectively. Let $\mathrm{vec}(\boldsymbol{A})$ denote the vectorization (reshape) of a given matrix $\boldsymbol{A}$. Let $\otimes$ denote the Kronecker product. Let $[N] = \{1, \ldots, N\} \subset \mathbb{N}$ be the set of indexing integers. Let $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B})$ denote the Frobenius inner product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Let $\|\cdot\|_F$ denote the Frobenius norm. Let $\mathrm{SO}(d)$ denote the d-dimensional special orthogonal group.

B. Semidefinite Relaxations of POPs

Many key problems in robotics can be expressed as POPs, including all of the problems mentioned in Section II-A. In this section, we review the well-known procedure for deriving convex, SDP relaxations of a standard form of POP. This procedure was pioneered by Shor [63] and has become the cornerstone of *certifiably correct methods* in robotics and computer vision.

We consider POPs that are parameterized on a variable, $\boldsymbol{\theta} \in \mathbb{R}^d$, and are formulated in the standard *homogenized, quadratically constrained quadratic problem* (QCQP) form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^\top \boldsymbol{Q}_{\boldsymbol{\theta}} \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x}^\top \boldsymbol{A}_{\boldsymbol{\theta} i} \boldsymbol{x} = 0 \quad \forall i \in [m] \\ & \boldsymbol{x}^\top \boldsymbol{A}_0 \boldsymbol{x} = 1 \end{aligned} \tag{P1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{Q}_{\boldsymbol{\theta}} \in \mathbb{S}^n$ and $\boldsymbol{A}_{\boldsymbol{\theta} i} \in \mathbb{S}^n$ denote the (parameterized) cost and constraint matrices, respectively. The *homogenizing constraint*, $\boldsymbol{x}^\top \boldsymbol{A}_0 \boldsymbol{x} = 1$, is used to ensure that the optimization variable is *homogeneous* (i.e., $x_i = 1$, for some $i \in 1, \ldots, n$) and $\boldsymbol{x} \in \mathbb{R}^n$. The exact form of the matrices, $\boldsymbol{Q}_{\boldsymbol{\theta}}$ and $\boldsymbol{A}_{\boldsymbol{\theta} i}$, and an explanation of this transformation can be found in Cifuentes et al. [17]. We note that it is always possible to formulate a POP in the standard form of Problem (P1). The *parameterized feasible set* of this problem is given by

$$\boldsymbol{\Omega}_{\boldsymbol{\theta}} = \{\boldsymbol{x} \in \mathbb{R}^n | \, \boldsymbol{x}^\top \boldsymbol{A}_{\boldsymbol{\theta} i} \boldsymbol{x} = 0 \quad \forall i \in [m] \, , \, \boldsymbol{x}^\top \boldsymbol{A}_0 \boldsymbol{x} = 1\}. \tag{1}$$

Problem (P1) can be re-expressed in terms of semidefinite matrices by using the properties of the trace operator[5] as follows:

$$\begin{aligned} \min_{\boldsymbol{X}} \quad & \langle \boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{X} \rangle \\ \text{s.t.} \quad & \langle \boldsymbol{A}_{\boldsymbol{\theta} i}, \boldsymbol{X} \rangle = 0 \quad \forall i \in [m] \\ & \langle \boldsymbol{A}_0, \boldsymbol{X} \rangle = 1 \\ & \boldsymbol{X} \succeq \boldsymbol{0} \\ & \mathrm{rank}(\boldsymbol{X}) = 1 \end{aligned} \tag{P2}$$

[4] Subject to an appropriate regularity condition such as Slater's condition.

[5] The trace is implicit in the inner product over matrices (i.e., Frobenius inner product).

where the last two constraints implicitly enforce the fact that $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^\top$. All the nonconvexity of Problem (P2) is contained in the single nonconvex rank constraint. It follows that we can find a convex relaxation of Problem (P2) by removing the rank constraint

$$\begin{aligned} \min_{\boldsymbol{X}} \ & \langle \boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{X} \rangle \\ \text{s.t.} \ & \langle \boldsymbol{A}_{\boldsymbol{\theta}i}, \boldsymbol{X} \rangle = 0 \quad \forall i \in [m] \\ & \langle \boldsymbol{A}_0, \boldsymbol{X} \rangle = 1 \\ & \boldsymbol{X} \succeq \boldsymbol{0}. \end{aligned} \tag{P3}$$

This relaxation—known as Shor's relaxation—has been well studied by the optimization community.

It can be shown that the Lagrangian of Problem (P1) is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \boldsymbol{x}^\top \left(\boldsymbol{Q}_{\boldsymbol{\theta}} + \boldsymbol{A}_0 \lambda_0 + \sum_{i=1}^{m} \boldsymbol{A}_{\boldsymbol{\theta}i} \lambda_i \right) \boldsymbol{x} - \lambda_0 \tag{2}$$

where $\boldsymbol{\lambda}^\top = [\lambda_0 \ \lambda_1 \cdots \lambda_m]$ are the Lagrange multipliers associated with the constraints (λ_0 being associated with the homogenizing constraint).[6] It is well known that Problem (P1) and Problem (P3) share a common Lagrangian dual problem

$$\begin{aligned} \min_{\boldsymbol{H}, \boldsymbol{\lambda}} \quad & \lambda_0 \\ \text{s.t.} \quad & \boldsymbol{H} = \boldsymbol{Q}_{\boldsymbol{\theta}} + \boldsymbol{A}_0 \lambda_0 + \sum_{i=1}^{m} \boldsymbol{A}_{\boldsymbol{\theta}i} \lambda_i \\ & \boldsymbol{H} \succeq \boldsymbol{0} \end{aligned} \tag{3}$$

where $\boldsymbol{H}$ is the dual matrix associated with the positive semidefinite constraint on $\boldsymbol{X}$. In what follows, we will refer to $\boldsymbol{H}$ as the *certificate matrix* associated with the solution, since constructing such a matrix can be used to certify global optimality of solutions [14].

1) Recovering a Global Solution: When the optimal solution of Problem (P3), $\boldsymbol{X}^*$, satisfies $\operatorname{rank}(\boldsymbol{X}^*) = 1$ then the convex relaxation is said to be *tight* to the original QCQP and the SDP solution can be factorized as $\boldsymbol{X}^* = \boldsymbol{x}^* \boldsymbol{x}^{*\top}$ to obtain the *globally optimal* solution, $\boldsymbol{x}^*$, of Problem (P1). This result has been proved rigorously by Cifuentes et al. [17], among others, and depends on the inherent properties of the QCQP formulation and the fact that strong duality holds *generically* for SDPs.[7] The stability of the *tightness* of semidefinite relaxations under perturbations to the objectives and constraints of the original QCQP was also studied extensively by Cifuentes et al. [17]. This property is important in our context, since it suggests that tuning the parameters of a problem with a tight relaxation will not cause tightness to be lost.

2) Redundant Constraints and Solving SDPs: For some problems in robotics and vision, the SDP relaxation is not immediately tight, but can be made so by adding so-called *redundant constraints* to the original QCQP (e.g., [11], [77]). Although redundant in the original QCQP, these constraints are *not redundant* for the SDP relaxation. In our recent works, we have shown how these redundant constraints can be efficiently and automatically generated for use in different state estimation problems that do not initially have tight relaxations [25], [38].

More generally, it has been shown that, subject to a mild technical condition [52], any POP (including those problems that we have cited in Section II) can be tightened via the *Moment-SOS Hierarchy* [36].[8] Put simply, the hierarchy involves the addition of redundant variables and constraints to the problem that tighten the SDP relaxation. Yang and Carlone [76] provide a thorough-yet-accessible introduction to these ideas for robotics practitioners.

In general, SDPs can be solved in *polynomial time* using interior-point methods [70]. These methods are fast for small problems and even tractable for problems of up to thousands of variables, but become prohibitive when real-time deployment is desired. Recent advances leverage low-rank SDP techniques such as that of *Burer and Monteiro* [9], [12] and the *Riemannian Staircase* [60] to solve larger problem instances in real time. However, our focus is on problems that require redundant constraints, which, to date, require solving the SDP directly.

C. Implicit Differentiation of Equality-Constrained Optimization Problems

In this section, we briefly review the mechanics of implicit differentiation as they apply to equality-constrained optimization problems. Our development is largely based on the tutorial by Giorgi and Zuccotti [33].

Consider the following optimization problem that is parameterized on $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \quad & f(\boldsymbol{x}, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{0} \end{aligned} \tag{P4}$$

where both f and $\boldsymbol{g}$ are smooth, differentiable functions and $\boldsymbol{g} : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^m$ is a vector-valued function representing m constraints. Letting $\boldsymbol{\lambda} \in \mathbb{R}^m$ be the Lagrange multiplier vector associated with the constraints, the Lagrangian of the optimization is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = f(\boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{4}$$

The *first-order KKT conditions* for Problem (P4) can be succinctly described in terms of a zero-level set of a function

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) \end{bmatrix} = \boldsymbol{0}. \tag{5}$$

For a given parameter value, $\bar{\boldsymbol{\theta}}$, any *primal-dual pair*, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}})$, corresponding to a *local optimum* of Problem (P4) will necessarily satisfy these conditions. These conditions are also sufficient for

[6] See Boyd and Vandenberghe [10] for an overview on Lagrangian duality theory.

[7] Slater's condition, which holds quite generally for SDPs (and in all of our problems), guarantees that strong duality holds.

[8] We refer here to the fact, introduced by Nie [52] that Lasserre's hierarchy has finite convergence for *generic* POPs, as long as the Archimedean condition holds (i.e., compactness of the feasible set)

strict local optimality if, in addition, the pair also satisfies the so-called *second-order sufficiency conditions* (SOSC),

$$\boldsymbol{v}^T \nabla_{\boldsymbol{x}}^2 L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \boldsymbol{v} > 0 \quad \forall \boldsymbol{v} \in \ker \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{6}$$

In this article, we are mainly concerned with the *set-valued solution mapping*, $\mathcal{S} : \mathbb{R}^d \rightrightarrows \mathbb{R}^{n+m}$,[9] from parameters to the *set* of primal-dual solutions

$$\mathcal{S} : \boldsymbol{\theta} \mapsto \{\boldsymbol{z} \in \mathbb{R}^{n+m} | \, \boldsymbol{k}(\boldsymbol{z}, \boldsymbol{\theta}) = \boldsymbol{0}\} \tag{7}$$

where for convenience, we have concatenated the primal-dual pair into a single vector, $\boldsymbol{z}^\top = \begin{bmatrix} \boldsymbol{x}^\top & \boldsymbol{\lambda}^\top \end{bmatrix}$. More importantly, we wish to understand the differential relationship between the input parameters and the solution. Adopting the notion of *differentials* from Magnus and Neudecker [48], we consider the following differential relationship:

$$\mathrm{d}\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{\theta}) = \boldsymbol{M} \mathrm{d}\boldsymbol{z} + \boldsymbol{N} \mathrm{d}\boldsymbol{\theta} = \boldsymbol{0} \tag{8}$$

where

$$\boldsymbol{M} = \frac{\mathrm{d}\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{z}} = \begin{bmatrix} \nabla_{\boldsymbol{x}}^2 L(\boldsymbol{z}, \boldsymbol{\theta}) & \nabla_{\boldsymbol{x}} g(\boldsymbol{x}, \boldsymbol{\theta}) \\ \nabla_{\boldsymbol{x}} g(\boldsymbol{x}, \boldsymbol{\theta})^\top & \boldsymbol{0} \end{bmatrix} \tag{9}$$

and $\boldsymbol{N} = \frac{\mathrm{d}\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}$. Whenever the matrix $\boldsymbol{M}$ is nonsingular, we can apply the classical version of the IFT to $\boldsymbol{k}$.[10] Broadly speaking, the IFT guarantees that, at least in a neighborhood containing $\bar{\boldsymbol{\theta}}$, the solution mapping in (7) is *single-valued*, and its exact Jacobian is given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{S}(\bar{\boldsymbol{\theta}}) = -\boldsymbol{M}^{-1} \boldsymbol{N}. \tag{10}$$

In the context of both bilevel optimization frameworks and neural network training, this Jacobian is used to find optimal search directions to minimize an overall loss function, $\ell(\boldsymbol{\theta})$. In general, parameters, $\boldsymbol{\theta}$, are provided to an optimization "layer," which then solves the inner optimization in a forward pass. In the backward pass, the Jacobian of the solution is used to backpropagate gradient information to the parameters.

In theory, Problem (P1) can be solved using any nonlinear, nonconvex optimization solver and its solution map can be differentiated by applying the IFT to the KKT conditions, as shown above [8], [29]. However, in practice, there is a serious issue with this approach; there is no guarantee that the nonlinear solver will converge to the global optimum of Problem (P1). Indeed, if the solver converges to a local optimum, then subsequent differentiation will occur with respect to the local solution instead of the true solution.

On the other hand, if global optimality of the solution can be certified, then gradients associated with that solution are also certified. Our approach is to directly solve the SDP relaxation in Problem (P3) in order to obtain a globally optimal solution, which often requires the use of redundant constraints to make the relaxation tight. Efficient differentiation of this solution is the subject of the following section.

[9] We use two arrows, $\rightrightarrows$, to denote a set-valued mapping. That is, a mapping that acts on a set and produces another set. In our case, the input set (the parameter set) is always singleton.

[10] For a formal definition of the IFT, see [23, Th. 1B.1].

IV. Differentiation of Certified Solutions

Given a tight solution to Problem (P3), there are several ways that we could implicitly differentiate the solution. One approach would be to implicitly differentiate the SDP solution itself using the CVXPYLayers framework. This approach has the advantage of efficiently reusing the Lagrange multipliers and certificate matrix that are computed as a byproduct of solving the SDP. Unfortunately, tight SDP relaxations often violate the solution uniqueness assumptions of CVXPYLayers (and its underlying solver [2]), meaning that the gradients they return cannot necessarily be trusted.[11]

Another approach would be to apply implicit differentiation to the original QCQP at a globally optimal solution obtained from the relaxation. The KKT system of the original QCQP is much smaller than that of its SDP relaxation and is, therefore, more computationally efficient to solve. However, the existing theory requires the constraint gradients to be linearly independent, which is violated when redundant constraints are present.[12] As a result, this approach would require us to first remove the redundant constraints and use the primal solution to recompute the Lagrange multipliers and certificate matrix before implicitly differentiating.

The goal of this section is to use an alternate version of the IFT that allows the use of redundant constraints and obviates the need for recomputation of the multipliers and certificate. We first show how the classic IFT can be applied to Problem (P1) when no redundant constraints are present and highlight the reason that it fails when redundant constraints are required. We then introduce an alternate version of the IFT that can be applied even when redundant constraints are used and provide our main result: under certain conditions, we can compute the Jacobian in (10) using the pseudoinverse of a matrix related to $\boldsymbol{M}$.

In Section VI-B3, we empirically show that the Jacobians of all the approaches discussed in this section are, to high precision, equal.

A. Differentiation Via Classic IFT

We first consider the situation in which no redundant constraints are needed to tight the relaxation and Problem (P1) satisfies the linear independence constraint qualification (LICQ). That is, the gradients of the constraints are linearly independent.

The KKT function from (5) associated with Problem (P1) is given by

$$\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{\theta}) = \begin{bmatrix} 2\boldsymbol{H}(\boldsymbol{\lambda}, \boldsymbol{\theta}) \boldsymbol{x} \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) \end{bmatrix}, \; \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{x}^\top \boldsymbol{A}_{\boldsymbol{\theta}1} \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^\top \boldsymbol{A}_{\boldsymbol{\theta}m} \boldsymbol{x} \\ \boldsymbol{x}^\top \boldsymbol{A}_0 \boldsymbol{x} - 1 \end{bmatrix} \tag{11}$$

[11] CVXPYLayers requires uniqueness of both the primal and the dual solution. It is known that low-rank SDPs are often primal degenerate, meaning that their dual solutions are not unique [76]. When the SDP solution has a rank of one, the results in [3] show that the dual is nonunique whenever $m + 1 > n$. This is quite often the case when we add redundant constraints to tighten the relaxation.

[12] See [33] for an extensive survey of the existing theory of perturbation of optimization problems.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

where $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{\lambda})$ is the primal-dual solution, $\nabla^2_{\boldsymbol{x}} L(\boldsymbol{z}, \boldsymbol{\theta}) = 2\boldsymbol{H}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ is exactly the certificate matrix described in Section II-I-B. We hereafter refer to $\boldsymbol{H}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ as $\boldsymbol{H}$ for the sake of brevity. The solution map corresponding to these KKT conditions is

$$\mathcal{S} : \boldsymbol{\theta} \mapsto \{\boldsymbol{z} \in \mathbb{R}^{n+m+1} \,|\, 2\boldsymbol{H}\boldsymbol{x} = \boldsymbol{0},\, \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{0}\}. \tag{12}$$

Considering the differential relationship in (8), we have

$$\boldsymbol{M} = 2\begin{bmatrix} \boldsymbol{H} & \boldsymbol{G}^\top \\ \boldsymbol{G} & \boldsymbol{0} \end{bmatrix} \tag{13}$$

where the rows of $\boldsymbol{G} \in \mathbb{R}^{(m+1)\times n}$ are the constraint gradients

$$\boldsymbol{G} = \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{\theta}1}\boldsymbol{x} & \cdots & \boldsymbol{A}_{\boldsymbol{\theta}m}\boldsymbol{x} & \boldsymbol{A}_0\boldsymbol{x} \end{bmatrix}^\top. \tag{14}$$

An explicit definition of $\boldsymbol{N}$ is deferred to Appendix A since it is not required for our main discussion at this point.

It can be shown that when the primal-dual solution satisfies the SOSC, $\boldsymbol{M}$ is invertible and the Jacobian of the solution map (12) is exactly given by (10) (see [30]).

B. Efficient Reuse of the Certificate Matrix

Practically speaking, the matrices, $\boldsymbol{H}$ and $\boldsymbol{G}$, which need to be computed to certify optimality, can be reused to efficiently construct the KKT matrix $\boldsymbol{M}$ for differentiation. However, when redundant constraints are used to tighten the relaxation, the resulting KKT matrix is singular, since $\boldsymbol{G}$ necessarily has linearly dependent rows. Although we cannot apply the *classical* IFT in this case, an alternative version of the IFT from [23] can still be applied under certain conditions. Making use of this alternate IFT, our main result shows that the solution Jacobian can be computed from the certificate matrix, even when redundant constraints are used. This result will require the following constraint qualification, which is weaker than the typical qualifications used in the optimization literature[13] and allows us to apply redundant constraints [17].

Definition 1 (Abadie Constraint Qualification (ACQ)): Given a set of constraints $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$, let $\boldsymbol{\Omega} := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}\}$ be the feasible set. The ACQ holds at $\boldsymbol{x} \in \boldsymbol{\Omega}$, if $\boldsymbol{\Omega}$ is a smooth manifold nearby $\boldsymbol{x}$, and

$$\operatorname{rank}(\nabla \boldsymbol{g}(x)) = \operatorname{codim}_{\boldsymbol{x}}(\boldsymbol{\Omega}) \tag{15}$$

where $\operatorname{codim}_{\boldsymbol{x}}(\boldsymbol{\Omega}) = n - \dim_{\boldsymbol{x}} \boldsymbol{\Omega}$ denotes the local codimension of $\boldsymbol{\Omega}$ at $\boldsymbol{x}$, $\dim_{\boldsymbol{x}} \boldsymbol{\Omega}$ denotes the local dimension of $\boldsymbol{\Omega}$ at $\boldsymbol{x}$, and $\nabla \boldsymbol{g}$ denotes the Jacobian matrix.

The ACQ holds in many robotics problems, especially in state estimation, where optimization variables such as rotations or poses are confined to a smooth manifold. With this qualification in hand, we provide our main theorem.

Theorem 2 (QCQP Jacobian): For some parameter, $\bar{\boldsymbol{\theta}}$, let $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}})$ be a primal-dual solution satisfying the (first-order) KKT conditions for Problem (P1) and let $\bar{\boldsymbol{H}}$ be the associated certificate matrix. Assume that the following conditions hold.

1) The set of parameterized matrices, $\{\boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{A}_{\boldsymbol{\theta}i}\}$, are continuously differentiable with respect to $\boldsymbol{\theta}$.
2) The ACQ and SOSC hold at $\bar{\boldsymbol{x}}$.
3) The mapping from the parameters, $\boldsymbol{\theta}$, to the feasible set, $\boldsymbol{\Omega}_{\boldsymbol{\theta}}$, is smooth near $\bar{\boldsymbol{\theta}}$.
4) The certificate matrix is positive semidefinite, $\bar{\boldsymbol{H}} \succeq \boldsymbol{0}$.

Then, $\bar{\boldsymbol{x}}$ is the globally optimal solution for Problem (P1) at $\bar{\boldsymbol{\theta}}$ and its unique Jacobian with respect to $\boldsymbol{\theta}$ is given by

$$\boldsymbol{J} = -\boldsymbol{P}\boldsymbol{M}_r^\dagger \boldsymbol{N} \tag{16}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{M}_r = 2\begin{bmatrix} \bar{\boldsymbol{H}} & \boldsymbol{G}^\top \\ \boldsymbol{G}_r & \boldsymbol{0} \end{bmatrix}. \tag{17}$$

$\boldsymbol{M}_r^\dagger = \boldsymbol{M}_r^\top(\boldsymbol{M}_r\boldsymbol{M}_r^\top)^{-1}$ denotes the (right) Moore–Penrose pseudoinverse, and $\boldsymbol{G}_r$ is obtained from $\boldsymbol{G}$ by removing any linearly dependent rows.

For brevity, the proof of the main result is provided in Appendix B.

Remark 1 (Smooth Feasible Set Assumption): The assumption of a smooth mapping from the parameters to the feasible set is more formally defined in [17, Definition 4.1], but is akin to assuming that the set, $\mathcal{W} = \{(\boldsymbol{\theta}, \boldsymbol{x}) \,|\, \boldsymbol{x} \in \boldsymbol{\Omega}_{\boldsymbol{\theta}}\}$, is (locally) a smooth manifold near $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{x}})$. In many robotics contexts, only the cost of the optimization is a function of the parameters and this assumption always holds. This is the case in all our experiments.

In our context, $\bar{\boldsymbol{x}}$ in Theorem 2 is obtained from a rank-one solution to the SDP relaxation, Problem (P3). As mentioned above, this theorem provides a means by which we can reuse the certificate matrix and Lagrange multipliers (obtained as a byproduct when solving the SDP), to compute the Jacobian of the solution with respect to the parameters. Crucially, the theorem applies regardless of whether redundant constraints have been used to tighten the problem.

C. Corank of the Certificate Matrix

The corank of the certificate matrix is important in our context for two key reasons. First, it has been shown that, when the corank of the certificate is one, the tightness of SDP relaxations is stable under small perturbations of the cost and constraint matrices [17]. This provides us with some guarantee that, as long as we start with a tight relaxation, tuning the input parameters will not cause the tightness of the SDP to be lost.

Second, the following Lemma (adapted from [17]) shows that the same condition on the corank implies that the SOSC holds for our problem.

Lemma 3: Let $\bar{\boldsymbol{H}}$ be the certificate matrix associated with a solution $\bar{\boldsymbol{x}}$ to Problem (P1) and suppose that $\bar{\boldsymbol{H}} \succeq \boldsymbol{0}$ and $\operatorname{corank} \bar{\boldsymbol{H}} = 1$. Then, $\boldsymbol{v}^\top \bar{\boldsymbol{H}} \boldsymbol{v} > 0$ for all nonzero $\boldsymbol{v}$ such that $\boldsymbol{v} \in \ker \boldsymbol{G}$.

Proof: Let $\boldsymbol{v} \in \ker \boldsymbol{G}$. The assumption $\boldsymbol{v}^\top \bar{\boldsymbol{H}} \boldsymbol{v} \geq 0$ holds with equality *only if* $\boldsymbol{v} = \mu \bar{\boldsymbol{x}}$ ($\mu \geq 0$), since $\operatorname{corank} \bar{\boldsymbol{H}} = 1$ and $\bar{\boldsymbol{H}}\bar{\boldsymbol{x}} = \boldsymbol{0}$ imply that $\ker \bar{\boldsymbol{H}} = \operatorname{im} \bar{\boldsymbol{x}}$. Since $\boldsymbol{v} \in \ker \boldsymbol{G}$ and $\bar{\boldsymbol{x}}^\top \boldsymbol{A}_0$ is a row of $\boldsymbol{G}$, we have that $\bar{\boldsymbol{x}}^\top \boldsymbol{A}_0 \boldsymbol{v} = \boldsymbol{v}_h = 0$, where $\boldsymbol{v}_h$ is the element of $\boldsymbol{v}$ that corresponds to the homegenization index.

[13] LICQ or the Mangasarian–Fromovitz Constraint Qualification are commonly used in sensitivity analysis of optimization problems [33]. Both of these qualifications imply the (ACQ), though the converse is not generally true.

Since $\boldsymbol{x}_h = 1$, $\boldsymbol{v}^\top \bar{\boldsymbol{H}} \boldsymbol{v} = 0$ only if $\mu = 0$. Thus, $\boldsymbol{v}^\top \bar{\boldsymbol{H}} \boldsymbol{v} > 0$ whenever $\boldsymbol{v} \neq \boldsymbol{0}$.

When *strict complementarity* of the relaxation holds, the certificate corank condition above is equivalent to having a rank-one solution to the relaxation. Strict complementarity has been shown to be *generic* for SDPs [3] and, in practice, we have observed that many relaxations of robotics problems have solutions that enjoy this property (c.f. [38]). Indeed, this property holds for all the problems investigated in Section VI.

V. Implementation

In this section, we introduce our implementation, the SDPRLayer, which computes the differentiable, globally optimal solution to a POP and can be embedded in any PyTorch autodifferentiation graph. As one of our contributions, we provide an encapsulation of our method in a PyTorch neural network module.

In a nutshell, this module takes as input a set of PyTorch tensors representing the parameterized cost and constraint matrices, $\{\boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{A}_{\boldsymbol{\theta}i}\}$, and returns the solution to the optimization as (differentiable) PyTorch tensor, which can then be used in succeeding layers of the PyTorch compute graph. If redundant constraints are used to tighten the problem, we assume that the user provides a list of these constraints.

As with many differentiable frameworks, the key components of our layer can be described in terms of a "forward" and "backward" function. The forward function computes the solution to the optimization problem and caches the primal-dual solution, while the backward function uses the cached solution to compute gradients of input parameters (tensors) from a known gradient of the solution. We provide details on the specific implementation of these functions in the remainder of this section. All operations are designed to be compatible with batched inputs and outputs.

A. SDPRLayer Forward Function

As mentioned, the forward function is used to find the primal-dual solution based on the parameterized input cost and constraint matrices.[14] By default, the primal-dual solution is found by formulating the SDP relaxation as a disciplined convex program (DCP) and passing it to (a modified version of) CVXPYLayers [1]. Alternatively, the user can specify to solve the problem using Mosek or provide their own primal-dual solution, computed by a custom solver. The primal-dual solution is then cached for use by the backward function.

The forward function always returns the matrix solution of the SDP relaxation [Problem (P3)] and, *when the relaxation is tight*, also returns the globally optimal solution vector to the nonconvex problem [Problem (P1)]. Both the matrix and vector solutions are returned as differentiable PyTorch tensors.

B. SDPRLayer Backward Function

The backward function has been implemented for both the globally optimal vector solution to Problem (P1) (when available) and the matrix solution to its relaxation, Problem (P3). In practice, the input parameters of the SDPRLayer are the (vectorized) input matrices

$$\boldsymbol{\nu}^\top = \begin{bmatrix} \mathrm{vec}(\boldsymbol{Q}_{\boldsymbol{\theta}})^\top & \mathrm{vec}(\boldsymbol{A}_{\boldsymbol{\theta},1})^\top & \cdots & \mathrm{vec}(\boldsymbol{A}_{\boldsymbol{\theta},m})^\top \end{bmatrix}. \quad (18)$$

The interpretation is that $\boldsymbol{\theta}$ is a set of parameters in the layers preceding the SDPRLayer, on which the cost and constraint matrices are parameterized.

The PyTorch framework that we have adopted uses reverse-mode autodifferentiation, in which the adjoint of the Jacobian is used to backpropagate gradient information. Given the gradient of the loss function with respect to the solution, $\nabla_{\boldsymbol{x}} \ell^\top$ (or $\nabla_{\boldsymbol{X}} \ell^\top$ if the matrix solution is used), the backward function computes the gradient of the loss with respect to $\boldsymbol{\nu}$

$$\nabla_{\boldsymbol{\nu}} \ell^\top = \boldsymbol{J}^\top \nabla_{\boldsymbol{x}} \ell^\top \quad (19)$$

where $\boldsymbol{J}$ is the solution map Jacobian that was discussed throughout Section IV. The rest of this section discusses the computation of this gradient in different cases.

1) Backpropagation via Implicit Selections (SDPR-IS): When the solution is tight (i.e., the rank of the SDP is one), we can backpropagate gradients using the Jacobian in Theorem 2, regardless of whether or not redundant constraints are used (referred to as SDPR-IS hereafter). We have

$$\nabla_{\boldsymbol{\nu}} \ell^\top = -\boldsymbol{N}^\top \boldsymbol{M}_r^{\top\dagger} \boldsymbol{P}^\top \nabla_{\boldsymbol{x}} \ell^\top. \quad (20)$$

For efficiency, we avoid explicit computation of the Jacobian (or its pseudoinverse) when computing the gradients. To do so, we first define an intermediate gradient, $\nabla_{\boldsymbol{y}} \ell^\top = \boldsymbol{M}_r^{\top\dagger} \boldsymbol{P}^\top \nabla_{\boldsymbol{x}} \ell^\top$, representing the gradient with respect to the KKT conditions. Noting that $\boldsymbol{M}_r^\top$ has full column rank, by the properties of the pseudoinverse we have

$$\nabla_{\boldsymbol{y}} \ell^\top = \arg\min_{\boldsymbol{y}} \|\boldsymbol{M}_r^\top \boldsymbol{y} - \boldsymbol{P}^\top \nabla_{\boldsymbol{x}} \ell^\top\|_2^2. \quad (21)$$

Similar to [2], we solve this optimization using the LSQR algorithm, which is specialized for sparse, unsymmetric linear systems [54]. This algorithm effectively leverages the sparsity of $\boldsymbol{M}_r$ by only requiring matrix-vector products and is also robust to ill-conditioned problems, which is advantageous in situations where constraint gradients are *nearly* dependent.[15] Gradients can then be computed via $\nabla_{\boldsymbol{\nu}} \ell^\top = -\boldsymbol{N}^\top \nabla_{\boldsymbol{y}} \ell^\top$, where $\boldsymbol{N}$ is as shown in Appendix A.

Under the conditions of Theorem 2, the approach shown above is guaranteed to provide the gradients of the globally optimal solution with respect to the input parameters.

2) Backpropagation via Classic IFT (SDPR-CIFT): The method in the previous section represents the default behavior of SDPRLayers, but we also provide an option to use the classic IFT to differentiate the original QCQP (referred to as SDPR-CIFT

[14] Since the homogenizing constraint is *always* required in our framework, we add it automatically. It does not need to be specified by the user.

[15] In practice, we have observed that the LSQR algorithm even provides meaningful gradients when some of the constraints are exactly dependent (i.e., $\boldsymbol{G}_r$ has linearly dependent rows).

hereafter). In this case, the primal solution, $\boldsymbol{x}$, is used to compute the (unique) Lagrange multipliers of a nonredundant subset of the constraints

$$\boldsymbol{\lambda} = -\boldsymbol{G}_r^{\top\dagger}\boldsymbol{Q}_{\boldsymbol{\theta}}\boldsymbol{x} \tag{22}$$

where the rows of $\boldsymbol{G}_r$ are the gradients of the nonredundant constraints, as in (38). Using the multipliers, we construct the matrices

$$\boldsymbol{M} = 2\begin{bmatrix}\boldsymbol{H}_r & \boldsymbol{G}_r^\top \\ \boldsymbol{G}_r & \boldsymbol{0}\end{bmatrix}, \quad \boldsymbol{H}_r = \boldsymbol{Q}_{\boldsymbol{\theta}} + \boldsymbol{A}_0\lambda_0 + \sum_{i=1}^{r}\boldsymbol{A}_{\boldsymbol{\theta}i}\lambda_i. \tag{23}$$

Finally, backpropagation is performed by solving the linear system

$$\boldsymbol{M}\boldsymbol{y} = \boldsymbol{P}^\top\nabla_{\boldsymbol{x}}\ell^\top \tag{24}$$

and computing the gradients via $\nabla_{\boldsymbol{\nu}}\ell^\top = -\boldsymbol{N}^\top\boldsymbol{y}$. In practice, we also use the LSQR algorithm to solve (24).

3) Backpropagation via SDP Solution (SDPR-SDP): As mentioned, the (matrix) solution to the SDP relaxation is provided to the user, regardless of whether the solution is tight. We use CVXPYLayers, which implicitly differentiates the KKT conditions of the SDP relaxation rather than the QCQP, to compute the gradients [1] (referred to as SDPR-SDP hereafter).

As mentioned in Section IV, the requisite assumption of a unique primal-dual solution in CVXPYLayers is often violated in our context.[16] Despite this technical issue, we have empirically observed that backpropagation through the SDP relaxation yields the same gradients as the backpropagation through the QCQP, at least when the relaxation is tight (see Section VI-B3).[17] However, backpropagation through the SDP relaxation typically involves solving a much larger linear system,[18] which is slower than differentiating through the nonconvex problem when the problem is large.

C. Recourse For Nontight Relaxations

The SDP matrix solution, $\boldsymbol{X}$, can still be used to obtain a near-optimal solution, $\hat{\boldsymbol{x}}$, by *rounding* to the nearest feasible point for the original problem (see, for example, [60]). This rounding procedure is problem dependent, but typically involves differentiable operations such as singular value decomposition (SVD) and projection.

The *suboptimality gap* of the near-optimal solution can be computed as follows:

$$\mu(\hat{\boldsymbol{x}}, \boldsymbol{X}) = \frac{\left\langle \boldsymbol{Q}_{\boldsymbol{\theta}}, \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^\top - \boldsymbol{X}\right\rangle}{\langle \boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{X}\rangle}. \tag{25}$$

The solution may be acceptable if this *suboptimality gap* is low. If the suboptimality gap is large, then the feasible point can still serve as an initial guess for a local solver, similar to the methodology used in [76]. There are then two possibilities for differentiation of the solution. If the local solver is also differentiable, then gradient information can be backpropagated through the local solver, rounding procedure, and CVXPYLayers implicit differentiation. Alternatively, the final solution can be passed back to the SDPRLayer and gradients can be computed via SDPR-CIFT (see Section V-B2).

In this case, the theory does not guarantee correctness of the gradients of the solution. An interesting avenue of future work could include an investigation of the relationship between the suboptimality gap and the *level* of correctness of the gradients.

We have added tools to our implementation for assessing and improving tightness of a given relaxation. These tools, along with a general approach to tightening SDP relaxations, are described in Appendix C.

VI. Experiments

We now present a series of examples that demonstrate the utility of our method in comparison to (nonglobal) alternatives. The first two experiments are simulated and highlight the issues with naive application of local optimization methods in the context of differentiable optimization. We use the second experiment to provide a detailed comparison of the Jacobians produced by the methods discussed above. The final experiment shows that our approach can be used to train a deep neural network in a real-world robotics pipeline. In all experiments, the optimal solutions satisfy all the assumptions of Theorem 2 and the corank of the certificate matrix was equal to one. We, therefore, use the SDPR-IS method in all cases, except when comparing Jacobians.

Note that our theory and implementation allow both cost and constraints to be functions of parameters, but our examples focus on cases where the constraints are fixed.

A. Polynomial Experiment

In this section, we consider a toy example in bilevel polynomial optimization that clearly illustrates the potential issues that can arise when local optimization is used and is assumed to converge to the global minimum. The objective of this bilevel optimization problem is to find a sixth-order polynomial that has a global minimum at a prespecified point, $(\bar{x}, \bar{y})$. The polynomial function is parameterized by its coefficients $\boldsymbol{\theta}$

$$y(x, \boldsymbol{\theta}) = \sum_{i=0}^{6}\theta_i x^i. \tag{26}$$

The task is split into an inner optimization, which attempts to find the global minimum of the current polynomial, and an outer optimization, which tunes the polynomial coefficients to shift the minimum to the specified point. The bilevel optimization can be written as follows:

$$\min_{\boldsymbol{\theta}} \quad (x^*(\boldsymbol{\theta}) - \bar{x})^2 + (y(x^*(\boldsymbol{\theta}), \boldsymbol{\theta}) - \bar{y})^2$$

[16] In particular, it is violated for the localization experiments in Sections VI-B and VI-C, though not in polynomial example in Section VI-A

[17] Internally, CVXPYLayers uses a least-squares formulation to solve for the gradients during backpropagation, which provides a solution even when the KKT Jacobian (equivalent to $\boldsymbol{M}$ above) is not invertible. We suspect that the reason that the gradients are correct is that they correspond to a local selection, similar to our development above. However, further exploration of this idea is left as future work.

[18] The number of primal and slack variables in the relaxation scales as $O(n^2)$.

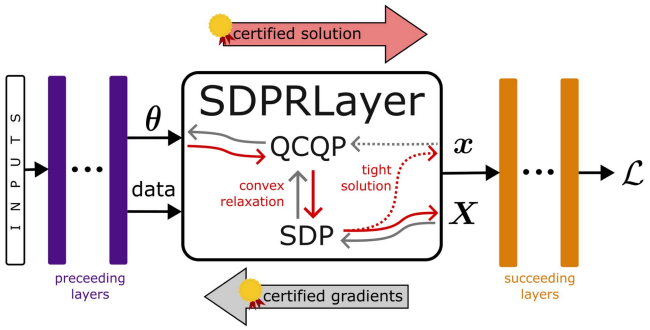

Fig. 1. Our SDPRLayer embedded in a PyTorch autodifferentiation graph. For nonconvex problems with *tight* semidefinite relaxations, the SDPRLayer finds the certified, globally optimal solution and makes it differentiable via implicit differentiation of the nonconvex QCQP. Current differentiable solvers for nonconvex problems can return gradients of spurious local minima rather than the gradients of the global solution, corrupting the learning process. In contrast, the gradients produced by the SDPRLayer correspond to the global solution as long as the relaxation is tight, leading to better training. Even when the relaxation is not tight, the relaxed, SDP solution is still provided, and can be used to find good initializations for local methods. The relaxed solution is also made differentiable by leveraging existing differentiable convex optimization layers.

TABLE I
INITIAL POLYNOMIAL COEFFICIENTS

θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
10.0	2.6334	4.3443	0.0	0.8055	0.1334	0.0389

$$\text{s.t.} \quad x^*(\boldsymbol{\theta}) = \arg\min_x y(x, \boldsymbol{\theta}). \tag{P5}$$

Although this example does not have direct application in robotics, it is a simplified analogue of robotics problems in RL, where an overparameterized value function must be learned to achieve a specific task.

1) Problem Parameters: In this example, we set the target global minimum to $(\bar{x}, \bar{y}) = (1.7, 7.3)$ and initialize the polynomial coefficients as shown in Table I.

2) Inner Optimization: We consider two different methods for solving the inner optimization. The first method is a standard nonlinear gradient descent (GD) method applied directly to the (unconstrained) polynomial function. On the first outer iteration, this method requires an initialization point, x_0, but subsequent (inner) optimizations are initialized using the previous minimum. Since we are dealing with a polynomial, the Jacobian can be computed analytically using the IFT.

The second method is a tight SDP relaxation of the inner optimization, which is implemented using our SDPRLayer. We parameterize the cost matrix as

$$\boldsymbol{Q}_{\boldsymbol{\theta}} = \begin{bmatrix} \theta_0 & \frac{1}{2}\theta_1 & \frac{1}{3}\theta_2 & \frac{1}{4}\theta_3 \\ \frac{1}{2}\theta_1 & \frac{1}{3}\theta_2 & \frac{1}{4}\theta_3 & \frac{1}{3}\theta_4 \\ \frac{1}{3}\theta_2 & \frac{1}{4}\theta_3 & \frac{1}{3}\theta_4 & \frac{1}{2}\theta_5 \\ \frac{1}{4}\theta_3 & \frac{1}{3}\theta_4 & \frac{1}{2}\theta_5 & \theta_6 \end{bmatrix}.$$

The following constraints lead to a tight semidefinite relaxation:

$$\boldsymbol{A}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_1 = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ 0 & -1 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{A}_2 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}.$$

Note that at least one of the constraints is redundant for the original QCQP.[19]

3) Outer Optimization: Since the outer optimization is unconstrained, we find the minimum using GD, with the gradients being computed based on the solution map of the inner optimization. The optimization is terminated when the loss function has a value less than 1×10^{-4}.

4) Results: The progression of the bilevel optimization is shown in Fig. 2. The key observation is that when GD is used for the inner optimization, the convergence of the overall optimization to a valid solution depends heavily on the initialization point.

When GD is initialized at $x_0 = 2$ (left plot), the solution converges to and remains at a local minimum of the polynomial. The gradients that are computed here correspond to the local minimum solution, thus, the outer optimization then shifts the *local minimum* of the polynomial to the target point, while the global minimum remains largely unchanged.

When GD is initialized at $x_0 = -2$ (center plot), the solution initially converges to the global minimum and provides valid gradients. As the outer iterations proceed, the inner optimization temporarily gets stuck in a local minimum. At this point, the outer optimization pushes the new global minimum in the wrong direction until the inner optimization eventually converges to the new global minimum. Subsequently, GD is able to reach a valid solution, though it required an increased number of iterations. This case shows that initializing the inner optimization well does not guarantee that it will not get stuck in a local minimum at some point during the optimization, hence providing incorrect gradient information.

In stark contrast to GD, the SPDRLayer *always* converges to the global solution of the problem and, therefore, always provides the correct gradients to the outer optimization (right plot). As a consequence, the outer optimization is able to consistently push the global minimum toward the target, and is even able to switch discontinuously to a different minimum as necessary (see Fig. 2).

B. Stereo Localization Example

In this example, we investigate the performance of SDPRLayers on a stereo-vision localization problem, which commonly

[19]This can be seen by noting that the original QCQP variable is given by $\boldsymbol{x}^\top = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix}$, which can be enforced by two constraints and one homogenizing constraint.

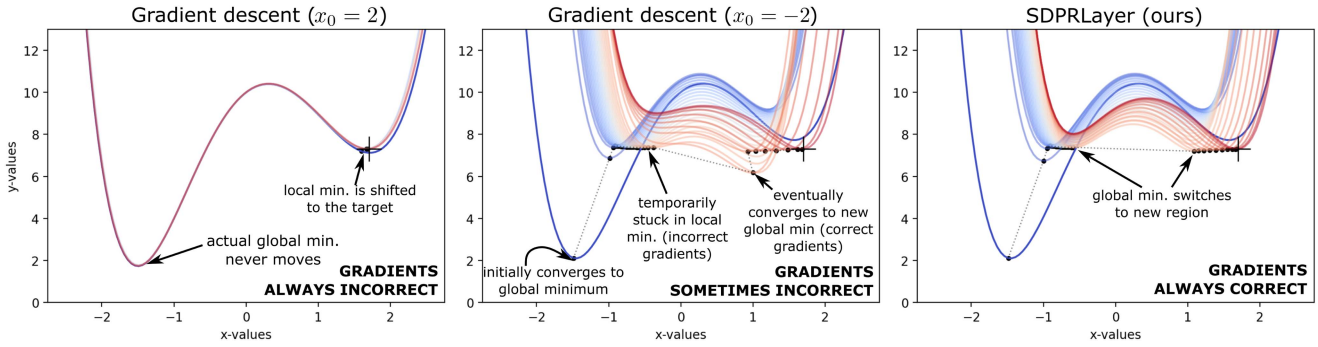

Fig. 2. Evolution of the polynomial function throughout the bilevel optimization. Columns present different methods for solving the inner optimization problem: (*left*) GD initialized at $x = 2$, (*center*) GD initialized at $x = -2$, and (*right*) our approach. In all three cases, the colour of the function indicates the progress of outer loop iterations (blue at the beginning, red at the end), the black dots indicate the minima found by the inner optimization, and the plus sign indicates the target. All three cases converge, but GD only converges to a valid solution when it is initialized well ($x_0 = -2$) and, even then, is temporarily trapped in a local minimum. SPDRLayer converges to a valid solution, does not require initialization, and is even able to discontinuously "switch" the minimum to a new region.

needs to be solved when estimating the state of a robot.[20] We assume that we have stereo camera measurements of a known set of N_m features, with known data association and no outliers. Pixel measurements are converted into Euclidean measurements using the following differentiable inverse-stereo-camera model (see Gridseth and Barfoot [35, Sec. IIIC] for details)

$$\boldsymbol{m}_k = \frac{b}{d_k}\begin{bmatrix} u_k - c_u \\ \frac{f_u}{f_v}(v_k - c_v) \\ f_u \end{bmatrix} \tag{27}$$

where $\boldsymbol{m}_k \in \mathbb{R}^3$ is the Euclidean measurement of the kth feature, b is the camera baseline, f_u and f_v are the horizontal and vertical focal lengths for the cameras, respectively, and c_u and c_v are the horizontal and vertical centers for the cameras, respectively. The left-camera horizontal, vertical, and disparity measurements (in pixels) of the kth feature are given by u_k, v_k, and d_k, respectively.

The (pixel-space) measurements are assumed to be perturbed by Gaussian noise with standard deviations of σ_u and σ_v in the horizontal and vertical directions, respectively. In turn, the Euclidean measurements are also perturbed by noise

$$\tilde{\boldsymbol{m}}_k = \boldsymbol{m}_k + \boldsymbol{\epsilon}_k \tag{28}$$

where $\boldsymbol{\epsilon}_k$ is a approximated by a zero-mean, Gaussian noise term with anisotropic covariance matrix $\boldsymbol{\Sigma}_k$. This matrix is a function of the camera parameters and can be computed using the inverse measurement model, as shown in [38]. The maximum-likelihood camera pose can be found by solving the matrix-weighted localization problem [38]

$$\begin{aligned} \min_{\boldsymbol{C},\boldsymbol{t}} \quad & \sum_{k=1}^{N_m} \boldsymbol{e}_k^\top \boldsymbol{W}_k(\boldsymbol{\theta})\boldsymbol{e}_k \\ \text{s.t.} \quad & \boldsymbol{C} \in \mathrm{SO}(3) \\ & \boldsymbol{e}_k = \tilde{\boldsymbol{m}}_k(\boldsymbol{\theta}) - \boldsymbol{C}(\boldsymbol{m}_k + \boldsymbol{t}) \end{aligned} \tag{P6}$$

where $\boldsymbol{\theta}$ represents the camera baseline, $\boldsymbol{C}$ and $\boldsymbol{t}$ are the camera rotation and translation, $\boldsymbol{m}_k$ is a feature point with known location, $\tilde{\boldsymbol{m}}_k(\boldsymbol{\theta})$ is the Euclidean measurement of the point from the camera frame and $\boldsymbol{W}_k(\boldsymbol{\theta})$ is the inverse of the covariance matrix, $\boldsymbol{\Sigma}_k$.

[20]Localization is also sometimes referred to as pointcloud regression.

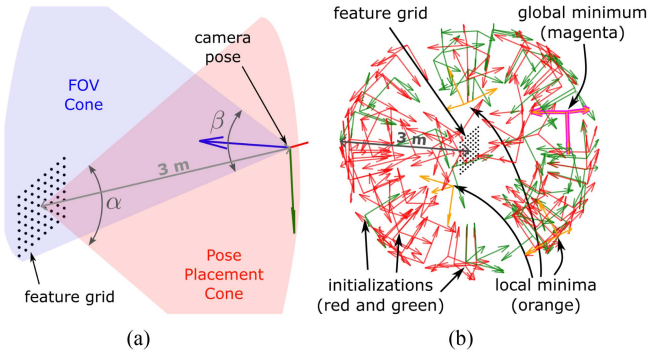

Fig. 3. (a) Setup for one of the ground-truth poses used in the stereo tuning example. The position of the ground-truth poses are placed 3 m from the center of the grid in a cone of angle $\beta = 90$ deg (red cone). The poses are oriented such that the center of the grid is within the field of view (FOV) of the camera (z) axis, $\alpha = 90$ deg. (b) 100 random initialization samples for Theseus' Gauss–Newton solver. The initializations are colour coded based on whether they converged to the local minimum (red) or global minimum (green). The local and global solutions are orange and magenta frames, respectively.

TABLE II
GROUND-TRUTH CAMERA PARAMETERS

Parameter	b	f_u	f_v	c_u	c_v	σ_u	σ_v
Units	m	$\frac{\text{pix}}{\text{m}}$	$\frac{\text{pix}}{\text{m}}$	pix	pix	pix	pix
Values	0.24	484.5	484.5	0.0	0.0	0.5	0.5

1) Problem Parameters: The experiments in this section use the setup shown in Fig. 3. We ran simulated experiments using the stereo camera parameters given in Table II. The features, $\boldsymbol{m}_k$, were arranged in an equally spaced, 8-by-8 grid occupying a 1.0 × 1.0 m rectangle located at the origin (similar to a checkerboard calibration pattern). In each experiment, ground-truth camera poses were placed randomly at a radius of 3 m from the center of the grid, within a 90° cone. Orientations of the camera poses were also randomized, but it was ensured that

TABLE III
JACOBIAN COMPARISON FOR SOLUTION $(\boldsymbol{C}, \boldsymbol{t})$ W.R.T. FEATURE LOCATIONS $(\boldsymbol{m}_k)$ FOR SCALAR-WEIGHTED LOCALIZATION (RELATIVE TO SVD SOLUTION)

Method	Jac. Diff. (mean)	Jac. Diff. (std)	RMSE Trans.	RMSE Rot.	Backprop Time (s)
SDPR-SDP	1.88E-05	2.44E-05	6.09E-07	1.08E-08	**1.36E-01**
SDPR-CIFT	**1.29E-06**	**3.06E-06**	6.09E-07	1.08E-08	3.89E-01
SDPR-IS (Default)	3.10E-06	7.82E-06	6.09E-07	1.08E-08	2.08E-01
Theseus-GT	1.02E-02	6.62E-03	1.61E-15	9.80E-10	1.86E-01
Theseus-GT-UR	8.28E-06	1.19E-05	4.81E-14	4.42E-08	7.74E-01
Theseus-RND	1.02E-02	6.62E-03	**1.57E-15**	**1.05E-09**	1.80E-01

TABLE IV
JACOBIAN COMPARISON FOR SOLUTION $(\boldsymbol{C}, \boldsymbol{t})$ W.R.T. FEATURE LOCATIONS $(\boldsymbol{m}_k)$ FOR MATRIX-WEIGHTED LOCALIZATION (RELATIVE TO THESEUS-GT-UR SOLUTION)

Method	Jac. Diff. (mean)	Jac. Diff. (std)	RMSE Trans.	RMSE Rot.	Backprop Time (s)
SDPR-SDP	1.49E-04	3.78E-04	8.38E-06	9.53E-06	**1.33E-01**
SDPR-CIFT	**3.11E-05**	**7.01E-05**	8.38E-06	9.53E-06	4.27E-01
SDPR-IS (Default)	3.95E-05	8.31E-05	8.38E-06	9.53E-06	2.31E-01
Theseus-GT	2.07E-02	2.88E-02	**6.10E-10**	**7.45E-09**	1.42E-01
Theseus-RND	1.95E-01	2.60E-01	9.02E-01	6.08E-01	1.79E-01

(LEGEND) **SDPR-SDP**: SDP solution diff. through SDP (CVXPYLayers); **SDPR-CIFT**: SDP solution diff. through QCQP using the classic IFT (Section V-B2); **SDPR-IS**: SDP solution diff. through QCQP using implicit selections (Section V-B1); **Theseus-GT**: Theseus solution diff. implicitly with ground-truth init.; **Theseus-RND**: Theseus solution diff. implicitly with random init.; **Theseus-GT-UR**: Theseus solution diff. via unrolling with ground-truth init.

TABLE V
BASELINE TUNING RESULTS ACROSS RUNS

Method	Final Baseline Error (avg)	Final Baseline Error (std)	Number of Outer Iterations (avg)	Outer Loss (avg)
Theseus-RND	1.132e-01	3.191e-02	8.034e+01	1.384e+02
Theseus-GT	**3.463e-04**	**4.901e-04**	**3.540e+01**	**9.854e-03**
SDPR-IS	**3.454e-04**	**4.907e-04**	**3.422e+01**	**9.857e-03**

TABLE VI
SUMMARY OF AVERAGE INLIERS AND LOCALIZATION ERROR ACROSS TEST DATA FROM [35]

Pipeline	Avg. Inliers	Long. Pos. Err. (m)	Lat. Pos. Err. (m)	Head. Err. (deg)	Avg. Inf. Time (s)
Baseline	**547.6**	0.209	0.134	0.414	**0.052**
Baseline (w. VGG16)	541.6	0.125	0.049	0.301	0.133
Ours	535.6	**0.035**	**0.021**	**0.288**	0.243

TABLE VII
SUMMARY OF AVERAGE INLIERS AND LOCALIZATION ERROR ACROSS TEST RUNS FROM CHALLENGING DATASET (TRANS. ≥ 0.5 M OR ROT $\geq 4°$)

Pipeline	Avg. Inliers	Long. Pos. Err. (m)	Lat. Pos. Err. (m)	Head. Err. (deg)	Avg. Inf. Time (s)
Baseline	**446.3**	0.432	0.144	0.377	**0.089**
Baseline (w. VGG16)	431.3	0.213	0.072	0.318	0.182
Ours	423.4	**0.053**	**0.027**	**0.316**	0.327

the center of the grid points were within a 90° field of view of the camera.

2) Solving the Optimization: Since Problem (P6) is a nonlinear least-squares problem, with $\mathrm{SO}(3)$ Lie group constraints, it can be solved using the Theseus optimization framework [57]. Moreover, as shown by Holmes et al. [38], Problem (P6) also has a tight semidefinite relaxation, which is particularly robust to noise when redundant constraints are used.

We implemented both Theseus and the SDPRLayer to solve the optimization. Since Theseus is a local solver, it requires an initial estimate of the pose to solve Problem (P6). Similar to the polynomial example, after the first iteration, the inner optimization is warm-started with the previous solution. We investigated situations in which Theseus is initialized with both the ground-truth and randomized initializations in order to demonstrate the effect of convergence to local minima. The random initializations used in this context are exemplified in Fig. 3(b); translations are randomly selected on the surface of a 3 m ball around the feature grid (at the origin), with orientation selected such that the z-axis points at the center of the grid and the other two axes are random. Fig. 3(b) shows that roughly 60% the initializations converge to poor local minima.

In contrast, the SDPRLayer does not require any initialization and always finds the globally optimal solution.

3) Jacobian Comparisons: To demonstrate the validity of our approach we first compare the Jacobians of the solution to Problem (P6) with respect to the parameters for different differentiation approaches across 50 trials. To make the experiments in this section as accurate as possible, we used Mosek [6] to solve the SDP with tolerances set to 1×10^{-12} and set the Theseus tolerances to 1×10^{-12}.

We performed an initial experiment with scalar weighting $(\boldsymbol{W}_k = \boldsymbol{I})$, since this allows the problem to be solved using the

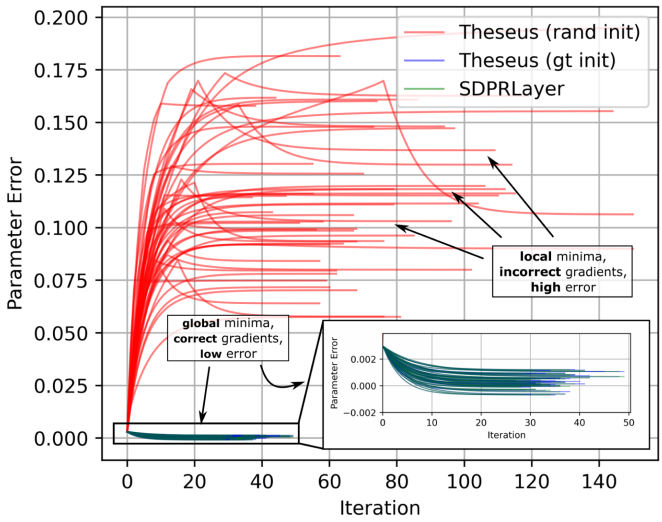

Fig. 4. Baseline parameter error trajectories across outer optimization iterations for 50 trials with different inner loop optimization approaches. Theseus with ground-truth initialization and SDPRLayer always converge to global minima, hence provide *correct* gradients to the outer opt. Theseus with random initialization sometimes converges to local minima and therefore provides *incorrect* gradients to the outer optimization, resulting in large parameter error.

(differentiable) SVD. The SVD solution is both closed-form and differentiable, serving as an accurate baseline for comparison of our approaches (see Umeyama [69] for the details). We compute the relative difference between Jacobians using the infinity norm

$$\Delta \boldsymbol{J}_{\text{est}} = \frac{\|\boldsymbol{J}_{\text{est}} - \boldsymbol{J}_{\text{true}}\|_\infty}{\|\boldsymbol{J}_{\text{true}}\|_\infty} \tag{29}$$

where $\boldsymbol{J}_{\text{true}}$ is the SVD Jacobian and $\boldsymbol{J}_{\text{est}}$ is the Jacobian of the alternate method. In Table III, we show the mean [Jac. Diff. (mean)] and standard deviation [Jac. Diff. (std)] of $\Delta \boldsymbol{J}_{\text{est}}$ across trials for each approach. We also provide the root mean squared error (RMSE) of the relative translation and rotation between the solutions to demonstrate that the all solutions converge to (approximately) the same point. [21]

The three SDPR approaches have (approximately) the same deviation in the Jacobian, indicating that they are equivalent in terms of accuracy. Despite having better accuracy in terms of the actual solution, the Jacobians of the local methods were less accurate except when using the "unrolling" method (UR).[22]

When the matrix weights were introduced, the SVD solution is no longer applicable, and we instead perform our comparisons with respect to the unrolled Theseus solution initialized at the ground truth (Theseus-GT-UR), since it was the most accurate. The results for this comparison are shown in Table IV. We call attention to the fact that, when Theseus is initialized randomly (Theseus-RND) in the matrix-weighted case, it can converge to local minima [as shown in Fig. 3(b)] and, as a result, the Jacobian does not match well with the ideal case. On the other hand, the SDPR solutions always converge to global minima and have accurate Jacobians.

Tables III and IV also include the mean backpropagation time for each case. As discussed in Section IV, reusing the multipliers and certificate matrix when backpropagating (SDPR-IS) leads to faster compute times than recomputing them (SDPR-CIFT). Curiously, backpropagation through the SDP KKT conditions (SDPR-SDP) was consistently faster than differentiating through the QCQP conditions. We posit that this is due to the small size of the SDP in this example and the fact that the CVXPYLayers backend is implemented using optimized C++ libraries while our QCQP differentiation is implemented using standard Python libraries.

4) Baseline Calibration: We now provide a further example of how gradient information obtained from an uncertified local solution can contaminate the processes that rely on gradient information. We again consider a bilevel optimization that uses Problem (P6) to calibrate the stereo baseline, b, of a camera rig. Problem (P6) constitutes the inner optimization, while the loss minimized by the outer optimization is the squared error between estimated camera pose and ground-truth camera pose

$$\min_{\boldsymbol{\theta}} \quad \|\boldsymbol{t}^*(\boldsymbol{\theta}) - \boldsymbol{t}_{gt}\|_2^2 + \|\boldsymbol{C}^*(\boldsymbol{\theta})^\top \boldsymbol{C}_{gt} - \boldsymbol{I}\|_F^2 \tag{P7}$$

where $\{\boldsymbol{C}^*(\boldsymbol{\theta}), \boldsymbol{t}^*(\boldsymbol{\theta})\}$ is the estimated pose from the inner optimization and $\{\boldsymbol{C}_{gt}, \boldsymbol{t}_{gt}\}$ represents the ground-truth pose. In practice, the loss could be unsupervised (i.e., not include ground-truth data), but we use this simplified loss to make comparison between approaches more straightforward. Since the outer optimization is unconstrained it can be solved iteratively using stochastic GD.

We ran 50 stereo calibration experiments with 20 different ground-truth poses and compared the results between Theseus and our approach. In the Theseus implementation, we use a Gauss–Newton solver with a stepsize of 0.2, with termination tolerances set to 1×10^{-8}, and implicit differentiation for backpropagation. For this experiment, the backend of the SDPRLayer implementation used the SCS to solve the SDP with tolerance set to 1×10^{-9} [53].

The baseline parameter was initialized with a 0.003 m error from the true value and the outer optimization was solved using stochastic GD with a learning rate of 1×10^{-4}. Each experiment terminated when the outer optimization gradient magnitude was less than 1×10^{-3} or 150 iterations had been reached. We stress that the *only difference* in the trials using Theseus in this section is the initialization used.

The parameter error trajectories for the different approaches are shown in Fig. 4 across outer optimization iterations and aggregate results are provided in Table V. From Fig. 4, it is clear that initializing randomly causes convergence to camera baseline values that are completely incorrect. By contrast, both our approach and Theseus with ground-truth initialization converge to a low level of error in a shorter number of iterations. Note that the individual trajectories of these two approaches are very close to each other. This is confirmed by the investigation of the gradients and inner optimization error. This is to be expected, since both approaches converge to the global minimum and thus

[21] Relative translation and rotation was computed in the Lie algebra vector space.

[22] Unrolling refers to backpropagation of gradients through all iterations of an optimization.

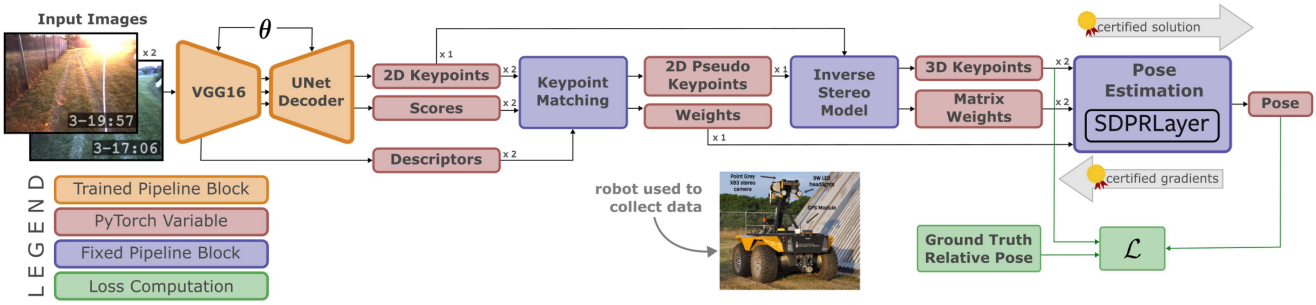

Fig. 5. Diagram of our Pytorch pipeline (based on the pipeline in [35]) that we use to estimate the relative pose between two images (i.e., localize a target image to a (stored) source image). Orange blocks denote the deep neural networks whose weights are tuned during training (VGG16 network and UNet Decoder network). Blue blocks denote blocks that are part of the pipeline, but do not have trainable parameters. Red blocks indicate key PyTorch tensor variables. Green blocks and arrows indicate training loss function computation.

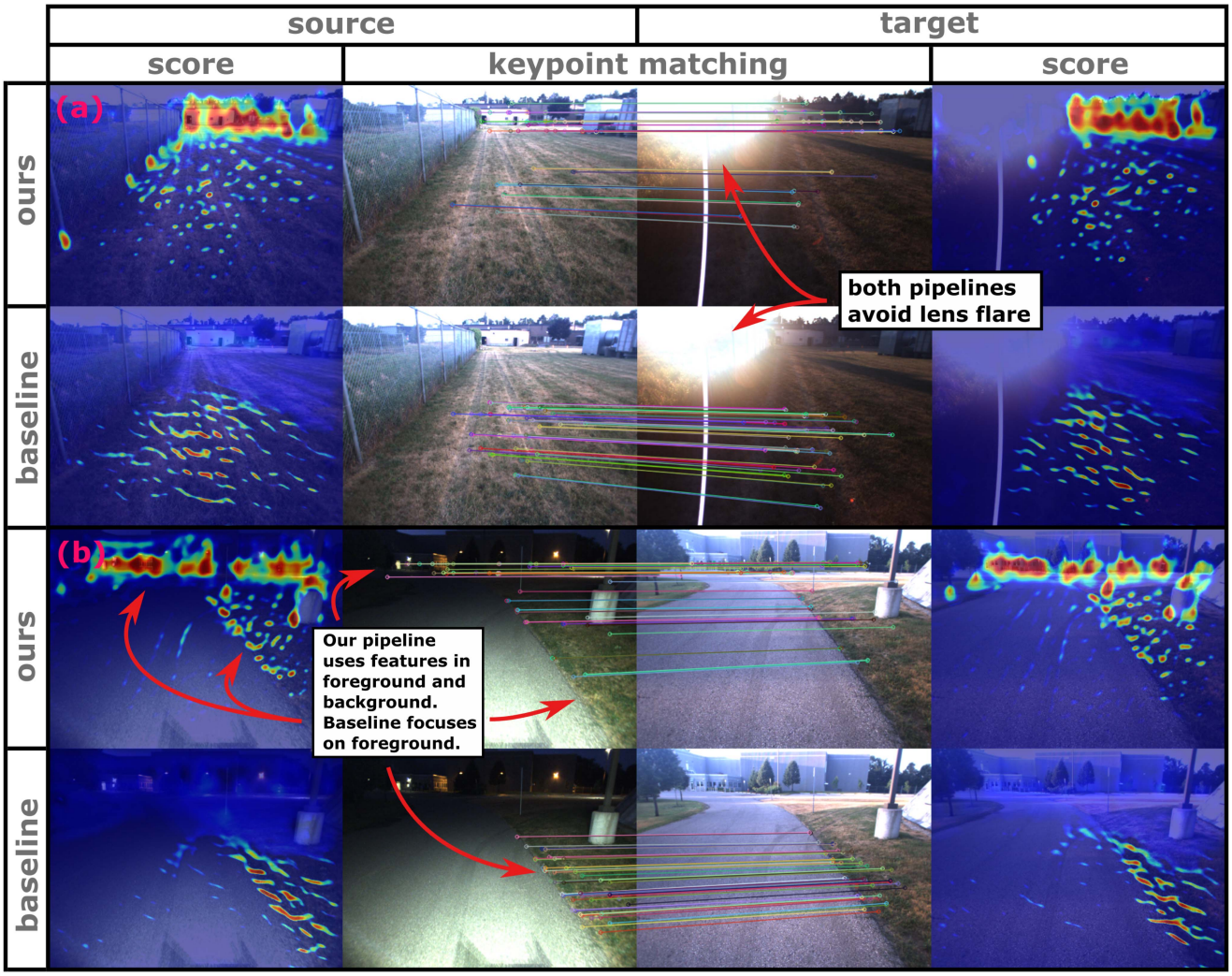

Fig. 6. Scores and keypoint matches for two selected stereo-image pairs from the In-The-Dark dataset after RANSAC filtering. Image pair (a) corresponds to source (Run 27, Frame 2182) and target (Run 8, Frame 1886) and demonstrates "lens flare." Image pair (b) corresponds to source (Run 21, Frame 2057) and target (Run 27, Frame 1830) and demonstrates a night-day match. The keypoint-match images (center two columns) display the source and target images with the 50 highest-weighted keypoint matches overlaid. The score columns show the score map from the neural network corresponding to their adjacent images.

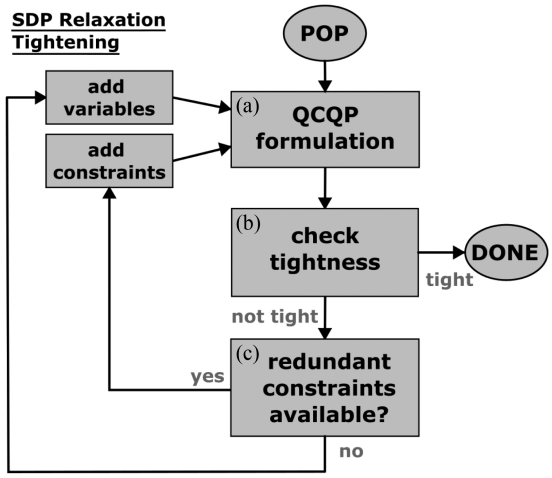

Fig. 7. General algorithm for tightening an SDP relaxation of a POP. Given a POP. (a) QCQP can be formulated using the methods discussed in [76] or [13]. (b) Tightness of the relaxation can be assessed using the SDPRLayer presented herein (see Section A3). (c) If the relaxation is not tight, SDPRLayer provides a method to find all possible constraints to improve tightness based on [25]. If the relaxation is still not tight, then variables can be added (i.e., move to a higher level of Lasserre's hierarchy).

return almost exactly the same gradients (subject to numerical precision of solvers).

Our investigations suggest that the divergence of the randomly initialized Theseus approach is exactly because the inner optimization converges to local minima for some poses. In turn, this leads to gradients that push the baseline in the wrong direction. Initial experimentation suggests that the "kinks" in the red trajectories of Fig. 4 occur because some poses of the batch are able to escape local minima and converge to better solutions as the baseline parameter changes.

The aggregate results in Table V corroborate our findings in Fig. 4; random initialization leads to poor tuning overall, increasing the number of required iterations, the average error and variation of tuned parameter. Again, our approach and Theseus with ground-truth initialization match very closely.

C. Deep Learned Visual Features for Localization

Our goal in this section is to demonstrate that the SDPRLayer can be used to train deep neural networks for large, real-world robotics problems. We consider the task of supervised learning of visual features for robot localization when environment lighting conditions can change drastically. In particular, we adopt and modify the learning pipeline introduced by Gridseth and Barfoot [35] for stereo-vision-based robot localization and show that the SDPRLayer enables *matrix-weighted* localization, which leads to increased accuracy.

Our adaptation of this fully differentiable pipeline is shown in Fig. 5. A convolutional neural network (VGG16 and UNet Decoder) is used to extract keypoints along with descriptors and scores for source and target RGB, stereo-images (i.e., four images total). Then, 2-D keypoints are detected in the left source and target images and matched. The keypoints in each stereo-image pair are then converted to 3-D coordinates using a stereo camera model and disparity between left and right images. Finally, the 3-D matched coordinates are used to compute a relative pose between source and target with a differentiable pose estimator. Relative (scalar) weighting of the keypoint pairs in the estimation are determined based on the scores and descriptor alignment (see [35, eq. (5)]).

The pose-estimation stage of the original pipeline used an SVD since, as mentioned above, it provides a closed-form and differentiable solution. However, since this method only supports scalar weights in the cost function, it cannot fully incorporate the (directional) uncertainty of keypoints that are derived from stereo images [51].

Properly accounting for depth uncertainty has been shown to be important for accurate state estimation in robotics [49], [51], [73]. This can be accomplished by replacing the pose-estimation block of the baseline pipeline with the (nonconvex) matrix-weighted pose estimation given in Problem (P6). As shown in Section VI-B, solving this optimization with local solvers can subject the network training to erroneous gradient information. To avoid this issue, we solve the optimization using the SDPRLayer with Mosek as the internal solver. As before, the matrix weights are computed based on the inverse covariance of each 3-D keypoint, which is known based on camera model and assumed pixel-space covariance.[23] The matrix weights are also scaled by the scalar weights provided by the matching block in the pipeline.

Finally, similar to Chen and Barfoot [16], we replace the encoder segment of the neural network with a VGG16 network [64] (truncated at `conv_5_3` layer) that has been pretrained on ImageNet [20] to facilitate faster training. We also retrained the baseline (SVD) pose estimation with the VGG16 encoder network to ensure a fair comparison.

1) Training: We kept as many parameters unchanged as possible between the baseline training setup and our setup. The feature detector network was trained for 30 epochs (10 000 samples in each epoch) on an NVIDIA Tesla V100 DGXS GPU using the same keypoint and pose-estimation training loss (with the same relative weights) as in [35] [see (7) and (8) therein]. The pose-estimation loss was not used for a "warm-up" period of 10 epochs (i.e., only keypoint loss in this period).

For training, we used 100 000 training samples and 20 000 validation samples from the "In-The-Dark" dataset,[24] which was also used for training by Gridseth and Barfoot [35] and has many instances of severe lighting changes. Using the Adam optimizer with a learning rate of 1×10^{-4}, the decoder network was trained from scratch and the VGG16 network was fine-tuned from its pretrained weights.

2) Testing: It is assumed that ground-truth data is unavailable during inference. As such, a random sample consensus (RANSAC) was used in the matching block to find the correct set of inliers based on reprojection error. In all cases, the pose estimation within the RANSAC algorithm was performed

[23] We assumed an isotropic, pixel-space, noise distribution with a standard deviation of 0.5 pixels.

[24] Dataset publically available at: http://asrl.utias.utoronto.ca/datasets/2020-vtr-dataset/.

using SVD to minimize runtime, with the final pose refinement performed with either SVD (scalar weighted) or SDPR (matrix weighted).

We compare our pipeline (Ours) against the baseline pipeline (Baseline) proposed in [35] and the baseline with VGG16 (Baseline (w. VGG16)) on a set of test runs that were held out of the training and validation data. In particular, we tested the localization pipelines on all source-target stereo-image pairs[25] from all combinations of runs 2, 11, 16, 17, 23, 28, and 35 from the In-The-Dark dataset and assessed the error based on comparison with ground truth. Test runs for the baseline were performed using the available code and network weight parameters associated with [35].[26]

The original test set involved only small relative-pose changes between source and target viewpoints since the localization pose graph was quite dense. As such, the original ground-truth relative-pose changes were 6.36 cm and 0.56° on average in translation and rotation, respectively. However, we also generated a more challenging test set by forcing the relative-pose change between map and live frames to be strictly greater than either 0.5 m translationally or 4° rotationally. For this challenging dataset, the average relative-pose change was 55.6 cm and 1.39° in translation and rotation, respectively.

3) Results: Tables VI and VII show number of inliers (from RANSAC), longitudinal position error, lateral position error, heading (yaw) error,[27] and inference time per pose, averaged across all runs. In both test sets, it is clear that our pipeline performs best in terms of longitudinal, lateral, and heading error, although the difference in heading error is negligible. We posit that the improved longitudinal and lateral error are due to appropriate accounting of the stereo-based depth uncertainty when matrix weights are used.

The average inference time differs between the two tables because the challenging dataset typically requires more RANSAC iterations to converge to an acceptable solution. The baseline pipeline with VGG16 took substantially longer than with the original encoder from [35], likely due to larger network size of VGG16. Interestingly, we note that, since the SDP used to globally solve Problem (P6) is relatively small, the average inference time with the SDPRLayer was only about two times longer than inference with the closed-form, SVD-based pose estimation.

The qualitative performance of the pipelines (with VGG16 encoder) is assessed in Fig. 6, which provides the scores and most-highly weighted keypoint matches on two samples from the In-the-Dark dataset. Both pipelines successfully find a good set of keypoint matches and manage to avoid problems induced by drastic lighting changes (i.e., "lens flare" and night-day matching).

We note that the baseline consistently focuses on keypoints in the foreground. We speculate that this is because, during training, the network cannot separate the high depth uncertainty from the low lateral uncertainty in faraway measurements and, therefore, reduces the weighting on the background keypoints.

On the other hand, our pipeline characterizes this directional uncertainty using the matrix weights. It is, therefore, able to weight the background keypoints more heavily than the baseline. We note in passing that this can be quite advantageous in localization because keypoints that are farther away typically provide better information about orientation and, as long as the problem is matrix weighted, will not corrupt the translation estimate. Both pipelines seem to use the background keypoints to achieve good heading estimates, but our implementation does so without adversely affecting the translation estimates.

The experiments shown in this section clearly demonstrate that our SDPRLayer can be used to train neural networks in real-world robotics pipelines. The bottleneck of the SDPRLayer approach is computational cost of solving the SDP. However, this localization problem can be solved quickly due to its small size and is useful in a practical context since it can provide globally optimal image registration with a large of numbers of features.

VII. Conclusion

In this article, we have presented the SDPRLayer, a differentiable optimization layer for POPs with tight semidefinite relaxations. We have demonstrated that differentiable optimization approaches with *local* solvers can provide gradient information that does not correspond to the global solution (due to convergence to spurious local minima). By extension, the training/optimization process may be lengthened or even fail entirely to achieve its objective. On the other hand, we provided theoretical and experimental results showing that the SDPRLayer efficiently computes *certified* gradients, in the sense that they correspond to the certified, global minimum of the optimization problem.

The first two examples shown in this article have demonstrated the potential pitfalls of naive application of differentiable optimization. The final example demonstrates that the SDPRLayer can be used for real-world robotics applications that combine deep-learned and model-based components. This method could be readily extended to train more recent feature-detection-and-matching pipelines (e.g., SuperGlue [62] or its variants [44], [47]) with pose registration error.

Currently, our theory is limited to SDP relaxations that are exactly tight (i.e., solution has a rank of one). For problems that do not have such relaxations, we have suggested an alternative approach using backpropagation through the KKT conditions of the relaxation (see Section V-C), but note that this approach is subject to theoretical limitations. An interesting direction of future work includes alleviating these limitations as well as further experimentation with problems that are not exactly tight.

Another current limitation of our approach is that the forward (optimization) and backward steps currently take place on the CPU, leading to costly memory transfers when combined with training with a GPU. A parallel GPU implementation of our approach necessary for the application of SDPRLayers—and more generally, certifiable methods—to larger robotics problems.

[25] The image data association between different runs was also performed using the code from [35].

[26] Code publically available at https://github.com/utiasASRL/deep_learned_visual_features

[27] All errors are average RMSE values relative to ground-truth vehicle pose frame from [56].

Finally, we believe that SDPRLayers could be applied to other areas of robotics apart from the perception methods demonstrated in this article. In particular, differentiable MPC is an exciting problem that has been recently studied in the literature [59] and may benefit from certifiable gradients in practice.

APPENDIX

A. Jacobians of KKT Conditions

In this section, we derive the expressions for the Jacobian of the KKT conditions with respect to the (vectorized) input parameters $\{\boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{A}_{\boldsymbol{\theta}i}\}$. Throughout this section, we adopt the notion of *differentials* and notation from Magnus and Neudecker [48]. We will make use of the following properties.

Proposition 4: Let $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ and let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ be fixed vectors. Then, we have the following differential relationships:

$$\mathrm{d}(\boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x}) = \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\top)^\top \mathrm{vec}(\mathrm{d}\boldsymbol{B}) \tag{30}$$

$$\mathrm{d}(\boldsymbol{B}\boldsymbol{x}) = (\boldsymbol{I} \otimes \boldsymbol{x}^\top)\mathrm{vec}(\mathrm{d}\boldsymbol{B}). \tag{31}$$

Proof: The first property can be verified easily using the trace operator and its relation to vectorized matrices, $\mathrm{tr}\boldsymbol{A}^\top\boldsymbol{B} = \mathrm{vec}(\boldsymbol{A})^\top\mathrm{vec}(\boldsymbol{B})$. We have

$$\begin{aligned}\mathrm{d}(\boldsymbol{x}^\top\boldsymbol{B}\boldsymbol{x}) &= \mathrm{tr}\boldsymbol{x}^\top\mathrm{d}\boldsymbol{B}\boldsymbol{x} = \mathrm{tr}\boldsymbol{x}\boldsymbol{x}^\top\mathrm{d}\boldsymbol{B}\\ &= \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\top)^\top\mathrm{vec}(\mathrm{d}\boldsymbol{B}).\end{aligned}$$

We consider the second property element-wise

$$(\mathrm{d}(\boldsymbol{B}\boldsymbol{x}))_i = \mathrm{d}(\boldsymbol{e}_i\boldsymbol{B}\boldsymbol{x}) = \mathrm{vec}(\boldsymbol{e}_i\boldsymbol{x}^\top)\mathrm{vec}(\mathrm{d}\boldsymbol{B})$$

where $\boldsymbol{e}_i$ is a vector of zeros with a one at index i. Collecting the differentials into a vector, we have

$$\mathrm{d}(\boldsymbol{B}\boldsymbol{x}) = \begin{bmatrix}\mathrm{vec}(\boldsymbol{e}_1\boldsymbol{x}^\top)\\ \vdots \\ \mathrm{vec}(\boldsymbol{e}_n\boldsymbol{x}^\top)\end{bmatrix}\mathrm{vec}(\mathrm{d}\boldsymbol{B}) = (\boldsymbol{I}\otimes\boldsymbol{x}^\top)\mathrm{vec}(\mathrm{d}\boldsymbol{B}).$$

We note that the differential of the certificate matrix is given by

$$\mathrm{d}\boldsymbol{H} = \mathrm{d}\boldsymbol{Q}_{\boldsymbol{\theta}} + \sum_{i=1}^{m}\mathrm{d}\boldsymbol{A}_{\boldsymbol{\theta}i}\lambda_i. \tag{32}$$

Applying Proposition 4, the differential of the KKT conditions in 11 are given by

$$\mathrm{d}(2\boldsymbol{H}\boldsymbol{x}) = 2(\boldsymbol{I}\otimes\boldsymbol{x}^\top)\left(\mathrm{vec}(\mathrm{d}\boldsymbol{Q}_{\boldsymbol{\theta}}) + \sum_{i=1}^{m}\lambda_i\mathrm{vec}(\mathrm{d}\boldsymbol{A}_{\boldsymbol{\theta}i})\right)$$

$$\mathrm{d}(\boldsymbol{x}^\top\boldsymbol{A}_{\boldsymbol{\theta}i}\boldsymbol{x}) = \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\top)\mathrm{vec}(\mathrm{d}\boldsymbol{A}_{\boldsymbol{\theta}i}).$$

Letting

$$\mathrm{d}\boldsymbol{\nu}^\top = \begin{bmatrix}\mathrm{vec}(\mathrm{d}\boldsymbol{Q}_{\boldsymbol{\theta}})^\top & \mathrm{vec}(\mathrm{d}\boldsymbol{A}_{\boldsymbol{\theta}1})^\top & \cdots & \mathrm{vec}(\mathrm{d}\boldsymbol{A}_{\boldsymbol{\theta}m})^\top\end{bmatrix}. \tag{33}$$

The differential of the KKT conditions is given by

$$\mathrm{d}\boldsymbol{k}(\boldsymbol{z},\boldsymbol{\theta}) = \begin{bmatrix}\mathrm{d}(2\boldsymbol{H}\boldsymbol{x})\\ \mathrm{d}(\boldsymbol{x}^\top\boldsymbol{A}_{\boldsymbol{\theta}1}\boldsymbol{x})\\ \vdots \\ \mathrm{d}(\boldsymbol{x}^\top\boldsymbol{A}_{\boldsymbol{\theta}m}\boldsymbol{x})\\ 0\end{bmatrix}. \tag{34}$$

We have the following differential relationship (when $\boldsymbol{z}$ is constant):

$$\mathrm{d}\boldsymbol{k}(\boldsymbol{z},\boldsymbol{\theta}) = \boldsymbol{N}\mathrm{d}\boldsymbol{\nu}$$

where applying (30), (30), and (32), the Jacobian is given by

$$\boldsymbol{N} = \begin{bmatrix}2(\boldsymbol{I}\otimes\boldsymbol{x}^\top) & \boldsymbol{\lambda}'^\top\otimes(2\boldsymbol{I}\otimes\boldsymbol{x}^\top)\\ \boldsymbol{0} & \boldsymbol{I}\otimes\mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\top)\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} \tag{35}$$

where $\boldsymbol{\lambda}'$ is the vector of Lagrange multipliers with λ_0 removed.

B. Proof of Theorem 2

In our context, the presence of redundant constraints causes the primal-dual solution, $\mathcal{S}(\boldsymbol{\theta})$, to be *necessarily set-valued* (due to the nonuniqueness of the Lagrange multipliers). This fact complicates the application of the classical IFT. However, when we find a globally optimal solution, we implicitly *select* a solution, $\bar{\boldsymbol{z}} \in \mathcal{S}(\boldsymbol{\theta})$, from this set. This *local selection* is formally defined in [23] as follows.

Definition 5 (Local Selection): Given a set-valued mapping $\mathcal{S}: \mathbb{R}^d \rightrightarrows \mathbb{R}^{n+m+1}$ and a pair, $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$, such that $\bar{\boldsymbol{z}} \in \mathcal{S}(\bar{\boldsymbol{\theta}})$, a function $\boldsymbol{w}: \mathbb{R}^d \to \mathbb{R}^{n+m+1}$ is said to be a *local selection* of $\mathcal{S}$ around $\bar{\boldsymbol{\theta}}$ for $\bar{\boldsymbol{z}}$ if $\boldsymbol{w}(\bar{\boldsymbol{\theta}}) = \bar{\boldsymbol{z}}$ and, for a neighborhood $\mathcal{V} \subseteq \mathbb{R}^d$ containing $\bar{\boldsymbol{\theta}}$, we have that $\boldsymbol{w}(\boldsymbol{\theta}) \in \mathcal{S}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \mathcal{V}$.

It turns out that the local selection function has a well-defined Jacobian, as shown by the next theorem, drawn from [23, Exercise 1F.9].

Theorem 6 (Implicit Selections): Consider a function $\boldsymbol{k}: \mathbb{R}^l \times \mathbb{R}^d \to \mathbb{R}^p$, where $p \leq l$, along with the associated solution mapping

$$\mathcal{S}: \boldsymbol{\theta} \mapsto \left\{\boldsymbol{z} \in \mathbb{R}^l \mid \boldsymbol{k}(\boldsymbol{z},\boldsymbol{\theta}) = \boldsymbol{0}\right\} \quad \text{for } \boldsymbol{\theta} \in \mathbb{R}^d. \tag{36}$$

Let $\boldsymbol{k}(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}}) = \boldsymbol{0}$, so that $\bar{\boldsymbol{z}} \in \mathcal{S}(\bar{\boldsymbol{\theta}})$. Assume that $\boldsymbol{k}$ is strictly differentiable at $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ and suppose further that the partial Jacobian $\nabla_{\boldsymbol{z}}\boldsymbol{k}(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ is of rank p. Then, the mapping $\mathcal{S}$ has a local selection $\boldsymbol{w}$ around $\bar{\boldsymbol{\theta}}$ for $\bar{\boldsymbol{z}}$ that is strictly differentiable at $\bar{\boldsymbol{\theta}}$ with Jacobian

$$\nabla\boldsymbol{w}(\bar{\boldsymbol{\theta}}) = \boldsymbol{M}^\top\left(\boldsymbol{M}\boldsymbol{M}^\top\right)^{-1}\boldsymbol{N} \tag{37}$$

where $\boldsymbol{M} = \nabla_{\boldsymbol{z}}\boldsymbol{k}(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ and $\boldsymbol{N} = \nabla_{\boldsymbol{\theta}}\boldsymbol{k}(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$.

Here, the equation $\boldsymbol{k}(\bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}}) = \boldsymbol{0}$ corresponds to the parameterized KKT conditions, where $\boldsymbol{z}$ is the primal-dual solution $((\boldsymbol{x}, \boldsymbol{\lambda}))$ and $\boldsymbol{\theta}$ are the parameters.

Remark 2 (Strict Differentiability): The notion of *strict differentiability* at a given point is rigorously defined in Dontchev and Rockafellar [23] and is equivalent to continuous differentiability at every point in an open set containing the point (c.f. [23,

Exercise 1D.8]). For our purposes, this distinction is tautalogical, since we will consider continuously differentiable functions.

Our approach to proving Theorem 2 is inspired by [17, Appendix A] and involves applying Theorem 6 to the KKT conditions of Problem (P1) at a globally optimal, primal-dual solution.

Proof: To apply the theorem we must first modify the KKT conditions so that $\boldsymbol{M}$ has full row rank. Let $\boldsymbol{g}_r(\boldsymbol{x},\boldsymbol{\theta})$ be a maximal subset of the constraint equations in $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\theta})$ such that the rows of the corresponding Jacobian, $\boldsymbol{G}_r \in \mathbb{R}^{r\times n}$, are linearly independent, where[28]

$$\boldsymbol{G}_r = \nabla_{\boldsymbol{x}} \boldsymbol{g}_r(\boldsymbol{x},\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{\theta}1}\boldsymbol{x} & \cdots & \boldsymbol{A}_{\boldsymbol{\theta}r}\boldsymbol{x} \end{bmatrix}^\top. \tag{38}$$

Under assumptions 2 and 3 in the theorem statement, [17, Lemma A.8] guarantees that the feasible set induced by $\boldsymbol{g}_r(\boldsymbol{x},\boldsymbol{\theta})$ is locally equivalent to the feasible set of Problem (P1), $\boldsymbol{\Omega}_{\boldsymbol{\theta}}$. Our solution mapping can, therefore, be re-expressed as follows:

$$\mathcal{S}:\boldsymbol{\theta} \mapsto \{\boldsymbol{z}\in\mathbb{R}^{n+r} \,|\, \bar{\boldsymbol{H}}\boldsymbol{x}=\boldsymbol{0},\, \boldsymbol{g}_r(\boldsymbol{x},\boldsymbol{\theta})=\boldsymbol{0}\}. \tag{39}$$

The Jacobian of this mapping with respect to $\boldsymbol{z}$ is exactly $\boldsymbol{M}_r$, as given in (17).

We now show that $\boldsymbol{M}_r$ has linearly independent rows. Let $\boldsymbol{t}^\top = \begin{bmatrix}\boldsymbol{v}^\top & \boldsymbol{u}^\top\end{bmatrix}$ be such that $\boldsymbol{M}_r^\top \boldsymbol{t} = \boldsymbol{0}$. We have

$$\begin{aligned} \bar{\boldsymbol{H}}\boldsymbol{v} + \boldsymbol{G}_r^\top \boldsymbol{u} &= \boldsymbol{0} \\ \boldsymbol{G}\boldsymbol{v} &= \boldsymbol{0}. \end{aligned}$$

The second line implies that $\boldsymbol{v}\in\ker\boldsymbol{G}=\ker\boldsymbol{G}_r$. Multiplying the first equation by $\boldsymbol{v}^\top$, we have

$$\boldsymbol{v}^\top\bar{\boldsymbol{H}}\boldsymbol{v} + \boldsymbol{v}^\top\boldsymbol{G}^\top\boldsymbol{u} = \boldsymbol{v}^\top\bar{\boldsymbol{H}}\boldsymbol{v} = \boldsymbol{0}$$

since $\boldsymbol{v}\in\ker\boldsymbol{G}$. The SOSC assumption implies that $\boldsymbol{v}^\top\bar{\boldsymbol{H}}\boldsymbol{v} > 0$ for all nonzero $\boldsymbol{v}\in\ker\boldsymbol{G}$. Combined with the equation above, we must have that $\boldsymbol{v}=\boldsymbol{0}$.

It follows that $\boldsymbol{u}=\boldsymbol{0}$ since $\boldsymbol{G}_r^\top\boldsymbol{u}=\boldsymbol{0}$ and $\boldsymbol{G}_r^\top$ has linearly independent columns by construction. Thus, the only vector $\boldsymbol{t}\in\ker\boldsymbol{M}_r^\top$ is the zero vector, so $\boldsymbol{M}_r$ has linearly independent rows.

Continuous differentiability of the KKT conditions with respect to $\boldsymbol{z}$ and $\boldsymbol{\theta}$ holds by the quadratic nature of the parameterized input matrices and by assumption, respectively.

All the conditions of Theorem 6 are satisfied (since $\boldsymbol{M}_r$ has full row rank) and, applying the theorem, there exists a local selection $\boldsymbol{w}_1$ of $\mathcal{S}$ around $\bar{\boldsymbol{\theta}}$ for $\bar{\boldsymbol{z}}$. Defining $\begin{bmatrix}\boldsymbol{x}_1^\top & \boldsymbol{\lambda}_1^\top\end{bmatrix} = \boldsymbol{w}_1^T(\boldsymbol{\theta})$, we have that the Jacobian of $\boldsymbol{x}_1$ with respect to $\boldsymbol{\theta}$ is given exactly by (16).

It remains to show that this Jacobian is, in fact, unique and not specific to the local selection $\boldsymbol{w}_1$. Let $\boldsymbol{w}_2$ be any other local selection of $\mathcal{S}$ around $\bar{\boldsymbol{\theta}}$ for $\bar{\boldsymbol{z}}$ and let $\begin{bmatrix}\boldsymbol{x}_2^\top & \boldsymbol{\lambda}_2^\top\end{bmatrix} = \boldsymbol{w}_2^T(\boldsymbol{\theta})$. Since $\boldsymbol{w}_1(\boldsymbol{\theta}),\boldsymbol{w}_2(\boldsymbol{\theta})\in\mathcal{S}$ for all $\boldsymbol{\theta}$ in a neighborhood about $\bar{\boldsymbol{\theta}}$, both selections satisfy the differential relationship given in (8).

[28]Note that here we have reordered the constraints such that the first r are linearly independent.

Subtracting, it follows that

$$\boldsymbol{M}_r \begin{bmatrix} d\boldsymbol{x}_1 - d\boldsymbol{x}_2 \\ d\boldsymbol{\lambda}_1 - d\boldsymbol{\lambda}_2 \end{bmatrix} = \boldsymbol{0}. \tag{40}$$

Let $\boldsymbol{v} = d\boldsymbol{x}_1 - d\boldsymbol{x}_2$ and $\boldsymbol{u} = d\boldsymbol{\lambda}_1 - d\boldsymbol{\lambda}_2$. Then, $\boldsymbol{v}\in\ker\boldsymbol{G}$ and $\bar{\boldsymbol{H}}\boldsymbol{v} + \boldsymbol{G}^\top\boldsymbol{u} = \boldsymbol{0}$. Multiplying by $\boldsymbol{v}^\top$, we have that $\boldsymbol{v}^\top\bar{\boldsymbol{H}}\boldsymbol{v}=0$, which implies that $\boldsymbol{v}=\boldsymbol{0}$ (since the SOSC holds). Therefore, it must be that $d\boldsymbol{x}_1 = d\boldsymbol{x}_2$. It follows that the differential $d\boldsymbol{x}_1$ is unique and has a unique Jacobian with respect to $\boldsymbol{\theta}$ given by (16).

Finally, it is well known that $\bar{\boldsymbol{H}}\succeq\boldsymbol{0}$ is sufficient to guarantee global optimality of $\bar{\boldsymbol{x}}$ [17]. By extension, (16) is the Jacobian of a globally optimal minimum.

C. Addressing Tightness

For the convenience of the user, we have added two functions to the SDPRLayer module to address the tightness of the SDP solution. The first function (`check_tightness`) computes the ratio of the maximum two eigenvalues of the SDP solution

$$r = \frac{\lambda_1(\boldsymbol{X}^*(\boldsymbol{\theta}))}{\lambda_2(\boldsymbol{X}^*(\boldsymbol{\theta}))}$$

where for $\boldsymbol{A}\in\mathbb{R}^n$, $\lambda_i(\boldsymbol{A})$ denotes the ith eigenvalue of $\boldsymbol{A}$, where $\lambda_1\geq\cdots\geq\lambda_m$. If the ratio exceeds a given threshold then the solution can be considered to be rank-1, and the relaxation is tight. Empirically, we have found a ratio of $r=1\times10^5$ to be a good indicator of relaxation tightness.

If the specified SDP relaxation is not initially tight, the procedure proposed by Dümbgen et al. [25] can be used to find a set of redundant constraints that may tighten the problem. Fig. 7 provides a visual representation of this procedure. Note that the approach outlined here is similar to the Moment-SOS hierarchy. However, the hierarchy does not necessarily find all possible constraints at each a given level before proceeding to the next level and adding new variables.

As mentioned, there are some guarantees on when this procedure—or more generally, the Moment-SOS hierarchy—results in a tight relaxation, but it can also lead to intractably large SDPs. Finding efficient, tight relaxations for POPs remains an active area of research.

D. Modifications to CVXPYLayers

CVXPYLayers parses user-provided DCPs into a canonical form and solves the DCP using a convex program solver (default solver is Splitting Conic Solver or SCS [53]). This canonicalization can often lead to inefficiencies when converting a DCP to a cone program. We have observed this empirically for the SDPs studied in robotics, depending on the solver to be used. This issue can be avoided by formulating the problem in dual form, but the implementation of CVXPYLayers did not previously support differentiation of slack or dual variables. As such, we have modified the interface of CVXPYLayers (and its underlying dependencies) to expose the (differentiable) Lagrange multipliers and slack variables as outputs, which are already computed by the underlying solvers.

We also extended the interface to allow the requisite primal, dual, and slack solution variables to be provided directly by the

user. In this case, the forward pass simply injects the solution variables from the external solver, bypassing the optimization. On the backward pass, the stored variables are used by the existing implicit differentiation machinery. This allows users to opt to use *external solvers* rather than the solvers in the CVXPYLayers codebase. We have used this approach to solve and differentiate SDPs with Mosek [6], which is often faster and more robust than the default solver.

REFERENCES

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Zico Kolter, "Differentiable convex optimization layers," in *Proc. Adv. Neural Inf. Process Syst.*, 2019, vol. 32.

[2] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. Moursi, "Differentiating through a cone program," *J. Appl. Numer. Optim*, vol. 10, no. 2, pp. 107–115, 2019.

[3] F. Alizadeh, A.J.-P. Haeberly, and M. L. Overton, "Complementarity and nondegeneracy in semidefinite programming," *Math. Program.*, vol. 770, no. 1, pp. 111–128, Apr. 1997.

[4] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," Dec. 2021, *arXiv:1703.00443*.

[5] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8299–8310.

[6] MOSEK ApS, *MOSEK Optimizer API for Python*, Mar. 2024.

[7] T. D. Barfoot, C. Holmes, and F. Dümbgen, "Certifiably optimal rotation and pose estimation based on the Cayley Map," *Int. J. Robot. Res.*, Sep. 2024, Art. no. 02783649241269337.

[8] M. Blondel et al., "Efficient and modular implicit differentiation," *Adv. Neural Inform. Process. Syst.*, vol. 35, pp. 5230–5242, Dec. 2022.

[9] N. Boumal, V. Voroninski, and A. S. Bandeira, "The non-convex Burer–Monteiro approach works on smooth semidefinite programs," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2016, pp. 2765–2773.

[10] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[11] L. Brynte, V. Larsson, J. P. Iglesias, C. Olsson, and F. Kahl, "On the tightness of semidefinite relaxations for rotation estimation," *J Math Imag. Vis*, vol. 640, no. 1, pp. 57–67, Jan. 2022.

[12] S. Burer and R. D. C. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Program.*, vol. 950, no. 2, pp. 329–357, Feb. 2003.

[13] L. Carlone, "Estimation contracts for outlier-robust geometric perception," *Found. Trends Robot.*, vol. 110, no. 2/3, pp. 90–224, Jun. 2023.

[14] L. Carlone, D. M. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert, "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Hamburg, Germany, Sep. 2015, pp. 125–132.

[15] K. N. Chaudhury, Y. Khoo, and A. Singer, "Global registration of multiple point clouds using semidefinite programming," *SIAM J. Optim.*, vol. 250, no. 1, pp. 468–501, Jan. 2015.

[16] Y. Chen and T. D. Barfoot, "Self-Supervised feature learning for long-term metric visual localization," *IEEE Robot. Autom. Lett.*, vol. 80, no. 2, pp. 472–479, Feb. 2023.

[17] D. Cifuentes, S. Agarwal, P. A. Parrilo, and R. R. Thomas, "On the local stability of semidefinite relaxations," *Math. Program.*, vol. 1930, no. 2, pp. 629–663, Jun. 2022.

[18] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.

[19] F. Dellaert, D. M. Rosen, J. Wu, R. Mahony, and L. Carlone, "Shonan rotation averaging: Global optimality by surfing $SO(p)^n$," in *Proc. Comput. Vis.–Eur. Vis*, 2020, pp. 292–308.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[21] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

[22] T. Dinev, C. Mastalli, V. Ivan, S. Tonneau, and S. Vijayakumar, "Differentiable optimal control via differential dynamic programming," Sep. 2022, *arXiv:2209.01117*.

[23] A. L. Dontchev and R. T. Rockafellar, "Implicit functions and solution mappings: A view from variational analysis," in *Operations Research and Financial Engineering*. New York, NY, USA: Springer, 2014.

[24] F. Dümbgen, C. Holmes, and T. D. Barfoot, "Safe and smooth: Certified continuous-time range-only localization," *IEEE Robot. Autom. Lett.*, vol. 80, no. 2, pp. 1117–1124, Feb. 2023.

[25] F. Dümbgen, C. Holmes, B. Agro, and T. Barfoot, "Toward globally optimal state estimation using automatically tightened semidefinite relaxations," *IEEE Trans. Robot.*, vol. 40, pp. 4338–4358, 2024.

[26] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," in *Proc. Int. Conf. Learn. Representations*, 2020.

[27] A. Eriksson, C. Olsson, F. Kahl, and T.-J. Chin, "Rotation averaging and strong duality," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 127–135.

[28] T. Fan, K. V. Alwala, D. Xiang, W. Xu, T. Murphey, and M. Mukadam, "Revitalizing optimization for 3D human pose and shape estimation: A sparse constrained formulation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Montreal, QC, Canada, Oct. 2021, pp. 11437–11446.

[29] A. V. Fiacco and Y. Ishizuka, "Sensitivity and stability analysis for nonlinear programming," *Ann. Oper. Res.*, vol. 270, no. 1, pp. 215–235, Dec. 1990.

[30] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Number 4 in Classics in Applied Mathematics. Philadelphia, PA, USA: SIAM, 1990.

[31] T. Fu, S. Su, and C. Wang, "iSLAM: Imperative SLAM," Jul. 2023, *arXiv:2306.07894*.

[32] M. Giamou et al., "Convex iteration for distance-geometric inverse kinematics," *IEEE Robot. Autom. Lett.*, vol. 70, no. 2, pp. 1952–1959, Apr. 2022.

[33] G. Giorgi and C. Zuccotti, "A tutorial on sensitivity and stability in nonlinear programming and variational inequalities under differentiability assumptions."

[34] A. Goudar, F. Dümbgen, T. D. Barfoot, and A. P. Schoellig, "Optimal initialization strategies for range-only trajectory estimation," *IEEE Robot. Autom. Lett.*, vol. 90, no. 3, pp. 2160–2167, Mar. 2024.

[35] M. Gridseth and T. D. Barfoot, "Keeping an eye on things: Deep learned features for long-term visual localization," *IEEE Robot. Autom. Lett.*, vol. 70, no. 2, pp. 1016–1023, Apr. 2022.

[36] D. Henrion, M. Korda, and J.-B. Lasserre, *The Moment-SOS Hierarchy: Lectures in Probability Statistics, Computational Geometry, Control Nonlinear PDEs* (Optimization and Its Applications 4). Hackensack, NJ, USA: World Scientific, 2021.

[37] C. Holmes and T. D. Barfoot, "An efficient global optimality certificate for landmark-based SLAM," *IEEE Robot. Autom. Lett.*, vol. 80, no. 3, pp. 1539–1546, Mar. 2023.

[38] C. Holmes, F. Dümbgen, and T. Barfoot, "On semidefinite relaxations for matrix-weighted state-estimation problems in robotics," *IEEE Trans. Robot.*, vol. 40, pp. 4805–4824, 2024.

[39] T. A. Howell, K. Tracy, S. L. Cleac'h, and Z. Manchester, "CALIPSO: A differentiable solver for trajectory optimization with conic and complementarity constraints," in *Robotics Research*, A. Billard, T. Asfour, and O. Khatib, Eds. Cham, Switzerland: Springer Nature, 2023, pp. 504–521.

[40] S. Hu and L. Carlone, "Accelerated inference in Markov random fields via smooth Riemannian optimization," *IEEE Robot. Autom. Lett.*, vol. 40, no. 2, pp. 1295–1302, Apr. 2019.

[41] J. P. Iglesias, C. Olsson, and F. Kahl, "Global optimality for point set registration using semidefinite programming," in *Proc. 2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Seattle, WA, USA, Jun. 2020, pp. 8284–8292.

[42] W. Jallet, N. Mansard, and J. Carpentier, "Implicit differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, USA, May 2022, pp. 1455–1461.

[43] K. M. Jatavallabhula, G. Iyer, and L. Paull, "∇SLAM: Dense SLAM meets automatic differentiation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Paris, France, May 2020, pp. 2130–2137.

[44] H. Jiang, A. Karpur, B. Cao, Q. Huang, and A. Araujo, "OmniGlue: Generalizable feature matching with foundation model guidance," May 2024, *arXiv:2405.12979*.

[45] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Pontryagin differentiable programming: An end-to-end learning and control framework," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 7979–7992.

[46] S. Kang, Y. Chen, H. Yang, and M. Pavone, "Verification and synthesis of robust control barrier functions: Multilevel polynomial optimization and semidefinite relaxation," Mar. 2023, *arXiv:2303.10081*.

[47] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "LightGlue: Local feature matching at light speed," Jun. 2023, *arXiv:2306.13643*.

[48] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus With Applications in Statistics and Econometrics* (Probability and Statistics), 3rd ed. Hoboken, NJ, USA: Wiley, 2019.

[49] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars exploration rovers," *J. Field Robot.*, vol. 240, no. 3, pp. 169–186, Mar. 2007.

[50] A. Majumdar, G. Hall, and A. A. Ahmadi, "recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. annual review of control," *Robot., Auton. Syst.*, vol. 30, no. 1, pp. 331–360, 2020.

[51] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Autom.*, vol. 3, no. 3, pp. 239–248, Jun. 1987.

[52] J. Nie, "Optimality conditions and finite convergence of Lasserre's Hierarchy," *Math. Program.*, vol. 1460, no. 1, pp. 97–121, Aug. 2014.

[53] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *J. Optim. Theory Appl.*, vol. 1690, no. 3, pp. 1042–1068, Jun. 2016.

[54] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM Trans. Math. Softw.*, vol. 80, no. 1, pp. 43–71, Mar. 1982.

[55] A. Papalia, A. Fishberg, B. W. O'Neill, J. P. How, D. M. Rosen, and J. J. Leonard, "Certifiably correct range-aided SLAM," Feb. 2023, *arXiv:2302.11614*.

[56] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, South Korea, Oct. 2016, pp. 1918–1925.

[57] L. Pineda et al., "Theseus: A library for differentiable nonlinear optimization," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 3801–3818, 2022.

[58] M. Qadri and M. Kaess, "Learning observation models with incremental non-differentiable graph optimizers in the loop for robotics state estimation," Sep. 2023, *arXiv:2309.02525*.

[59] A. Romero, Y. Song, and D. Scaramuzza, "Actor-Critic model predictive control," Sep. 2023, *arXiv:2306.09852*.

[60] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," *Int. J. Robot. Res.*, vol. 38, no. 2/3, pp. 95–125, Mar. 2019.

[61] D. M. Rosen, K. J. Doherty, A. T. Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 40, no. 1, pp. 215–242, 2021.

[62] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Seattle, WA, USA, Jun. 2020, pp. 4937–4946.

[63] N. Z. Shor, "Quadratic optimization problems," *Sov. J. Comput. Syst. Sci.*, vol. 25, pp. 1–11, 1987.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Apr. 2015, *arXiv:1409.1556*.

[65] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning tactile models for factor graph-based estimation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Xi'an, China, May 2021, pp. 13686–13692.

[66] R. Talak, L. R. Peng, and L. Carlone, "Certifiable object pose estimation: Foundations, learning models, and self-training," *IEEE Trans. Robot.*, vol. 390, no. 4, pp. 2805–2824, Aug. 2023.

[67] Z. Teed and J. Deng, "DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 16558–16569, 2021.

[68] Z. Teed and J. Deng, "Tangent space backpropagation for 3D transformation groups," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Nashville, TN, USA, Jun. 2021, pp. 10333–10342.

[69] S. Umeyama, "Least-Squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 130, no. 4, pp. 376–380, Apr. 1991.

[70] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 380, no. 1, pp. 49–95, Mar. 1996.

[71] C. Wang et al., "PyPose: A library for robot learning with physics-based optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Vancouver, BC, Canada, Jun. 2023, pp. 22024–22034.

[72] P.-W. Wang, P. L. Donti, B. Wilder, and Z. Kolter, "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver," May 2019, *arXiv:1905.12149*.

[73] J. Weng, P. Cohen, and N. Rebibo, "Motion and structure estimation from stereo image sequences," *IEEE Trans. Robot. Autom.*, vol. 80, no. 3, pp. 362–382, Jun. 1992.

[74] E. Wise, M. Giamou, S. Khoubyarian, A. Grover, and J. Kelly, "Certifiably optimal monocular hand-eye calibration," in *Proc. IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, Karlsruhe, Germany, Sep. 2020, pp. 271–278.

[75] J. Xu et al., "An end-to-end differentiable framework for contact-aware robot design," in *Proc. Robot.: Sci. Syst. XVII. Robot.: Sci. Syst. Found.*, Jul. 2021.

[76] H. Yang and L. Carlone, "Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 450, no. 3, pp. 2816–2834, Mar. 2023.

[77] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 370, no. 2, pp. 314–333, Apr. 2021.

[78] B. Yi, M. A. Lee, A. Kloss, R. Martín-Martín, and J. Bohg, "Differentiable factor graph optimization for learning smoothers," Aug. 2021, *arXiv:2105.08257*.

[79] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Trans. Autom. Control*, vol. 660, no. 8, pp. 3638–3652, Aug. 2021.

Connor Holmes (Graduate Student Member, IEEE) received the B.A.Sc. degree in engineering science and the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2014 and 2016, respectively. Since 2021, he has been working toward the Ph.D. degree in certifiable methods for state estimation in robotics with the University of Toronto Robotics Institute, Toronto, ON, Canada.

From 2016 to 2021, he was a Guidance Navigation and Controls Engineer with MDA Space, Toronto, ON, Canada. His research interests include the application of efficient optimization techniques in robotics, particularly for state-estimation algorithms.

Frederike Dümbgen (Member, IEEE) received the B.Sc. and M.Sc. degrees in mechanical engineering, with minor in computational science and engineering, and the Ph.D. degree in computer and communication sciences from École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2013, 2016, and 2021, respectively.

She is currently conducting her master's thesis at ETH Zürich, Zürich, Switzerland. She was a Postdoctoral Researcher with the Robotics Institute of University of Toronto, Toronto, ON, Canada, from 2022 to 2024. Since May 2024, she has been a Researcher in the WILLOW team, affiliated with Inria and ENS, PSL University, Paris, France. Her research interests lie in the areas of estimation, control and optimization for robotics.

Timothy D. Barfoot (Fellow, IEEE) received the B.A.Sc. degree in engineering science and the Ph.D. degree in aerospace science and engineering from the University of Toronto, Toronto, ON, Canada, in 1997 and 2002, respectively.

He is currently a Professor with the University of Toronto Robotics Institute, Toronto, ON, Canada. He works in the areas of guidance, navigation, and control of autonomous systems for a variety of applications. He is interested in developing methods to allow robotic systems to operate over long periods of time in large-scale, unstructured, three-dimensional environments, using rich onboard sensing (e.g., cameras and laser rangefinders) and computation.