Planning Shorter Paths in Graphs of Convex Sets by Undistorting Parametrized Configuration Spaces

Shruti Garg[1], Thomas Cohn[1], and Russ Tedrake[1]

***Abstract*— Optimization based motion planning provides a useful modeling framework through various costs and constraints. Using Graph of Convex Sets (GCS) for trajectory optimization gives guarantees of feasibility and optimality by representing configuration space as the finite union of convex sets. Nonlinear parametrizations can be used to extend this technique to handle cases such as kinematic loops, but this distorts distances, such that solving with convex objectives will yield paths that are suboptimal in the original space. We present a method to extend GCS to nonconvex objectives, allowing us to "undistort" the optimization landscape while maintaining feasibility guarantees. We demonstrate our method's efficacy on three different robotic planning domains: a bimanual robot moving an object with both arms, the set of 3D rotations using Euler angles, and a rational parametrization of kinematics that enables certifying regions as collision free. Across the board, our method significantly improves path length and trajectory duration with only a minimal increase in runtime.**

I. Introduction

Reliable motion planning is essential to developing and deploying robotic manipulation systems. Such systems need to produce efficient paths while obeying various constraints. Optimization-based motion planning, which tries to minimize an objective function while satisfying constraints, offers a powerful paradigm to solve this problem. The decision variables describe the robot's trajectory, the objective allows for choosing desired qualities in the solution, and constraints on these decision variables define obstacle avoidance, dynamic limits, and other interesting task-specific constraints such as coordinating arms in a bimanual system. However, the power of these techniques is tempered by the need to carefully formulate the optimization problems for reliability.

A manipulator's configuration space is often inherently nonconvex, and nonconvex trajectory optimization usually cannot guarantee optimality (due to local minima), or even feasibility. These guarantees are key to efficient and robust systems that can be used for repetitive motions in safety critical settings. Graphs of Convex Sets (GCS) [1], [2] encapsulates the nonconvexities from obstacle avoidance as discrete decisions, casting it as a convex optimization problem. More concretely, it takes the inner approximation of planning or configuration space represented as a series of intersecting collision free convex subsets. Then, it can optimize over a graph (where vertices are a convex c-space set and edges represent intersections between these sets) for discrete paths through these sets and simultaneously optimize for the shortest or fastest continuous path within each set. Many relevant properties of the trajectory and its derivatives can be transcribed into convex costs and constraints [2].

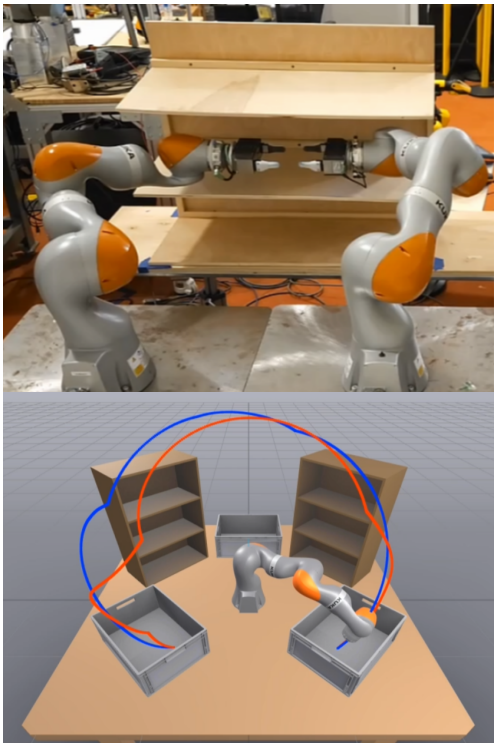

Fig. 1: Experiments include constrained bimanual motion planning between shelves (top) and certifiable 7DoF KUKA iiwa trjaectories between bins (below). The red path is the original result, and the blue path is our improved result.

When the configuration space does not admit finite Euclidean inner approximations, recent work has reformulated the problem through parametrization to use GCS. For example, Cohn et al. [3] tackle planning with an equality constraint between end effectors of two manipulators using inverse kinematics (IK). This parametrization satisfies the aforementioned equality constraint by construction [3]. Another useful application of parametrizations is the Euler angles representation of 3D rotations, which enables planning over this space with GCS [4, §2.7.5]. Certified C-IRIS [5] uses a rational parametrization of robot kinematics

This work was supported by Amazon.com, PO No. 2D-06310236, Lincoln Labs, and the National Science Foundation Graduate Research Fellowship Program under Grant No. 2141064. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1]The authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA, 02139 [sgrg,tcohn,russt]@mit.edu

to formulate an optimization problem to provide collision free certificates in presence of task space obstacles.

Unfortunately, these parametrizations are non-isometric; they distort space, so the shortest path in the parametrized space may not be the shortest path in the original configuration space. This leads to suboptimal results when the convex objective used (path length costs) in the parametrized space is a weak approximation for the true objective. Allowing GCS to handle nonconvex objectives, such as the true objective in the parametrized space, gives more modeling freedom, widening the breadth of problems we can tackle while maintaining guarantees.

This work considers GCS problems with nonconvex objectives, while maintaining convex constraints. Specifically, we use Projected Gradient Descent (PGD) to optimize such nonconvex objectives in a GCS approach. Using a gradient based solver guarantees local optimality around the initial guess if the objective is Lipschitz-continuous. Since we keep constraints convex, a small convex program can always project infeasible solutions back to feasibility. Thus by exploiting structured nonconvexity, our solver improves optimality of solutions while maintaining feasibility guarantees.

For each of the three parametrizations mentioned earlier (bimanual IK, Euler angles, rational kinematics), we formulate and test a nonconvex objective against a convex surrogate in the GCS optimization formulation. This nonconvex optimization is treated as a postprocessing step, improving upon the best solution from GCS. Our method offers significant quantitative and qualitative improvements to motion plans across multiple experiments: path lengths and trajectory times shorten and visual artifacts of planning in the distorted parametrized space are undone. Expanding beyond the regime of undoing distortions, we also optimize over a general nonconvex cost, spatial curvature, to further improve the results from the bimanual experiment.

In the rest of this paper, we first review related work on nonconvex trajectory optimization and give necessary background on GCS and existing efforts to add nonconvexity. Then, we describe our methodology, relevant implementation details, and experimental set-ups. Finally, we present our experimental results, and conclude with a brief discussion of the limitations of our work and potential directions for future research.

II. Background and Related Work

Sampling-based planners such as Probabilistic Roadmaps [6] and Rapidly Exploring Random-Trees [7] work very well in practice for kinematic planning problems. By growing finer approximations of the configuration space through sampling, they will eventually find a solution if one exists. Some methods, such as RRT*[8], can even achieve asymptotic optimality. However, these planners on their own struggle to handle more complex objectives.

Using optimization to solve for the entire trajectory enables more modeling freedom. Objectives can be used to prioritize choice qualities (such as distance or speed) and general constraints are essential for handling dynamics. Roboticists implement optimization based motion planning through a variety of different formulations, including direct collocation [9], Augmented Lagrangian [10], and pseudo-spectral methods [11]. These transcriptions can then be solved using general purpose solvers such as SNOPT [12] or gradient-based methods. For example, KOMO uses gradient descent on an Augmented Lagrangian transcription [13], and CHOMP uses covariant gradient descent [14]. Nonconvexity will often be handled with clever initialization or stochasticity, such as in STOMP [15]. cuRobo [16] moves all constraints into the objective and leverages parallelization to simultaneously consider many initial guesses. Across all of these approaches, the formulation remains nonconvex, lacking feasibility or optimality guarantees.

A. Graphs of Convex Sets

Graph of Convex Sets (GCS) presents a new strategy for solving shortest path problems with continuous and discrete decisions. Formally, a GCS is a graph, where each vertex v has an associated continuous variable x_v within a convex set X_v, and each edge (u, v) is a convex function of x_u and x_v. Finding the shortest path $\mathcal{P}$ through this graph can then be formulated as a Mixed Integer Convex Problem (MICP) with $\mathcal{P}$ and x_v as decision variables [1, §5].

In motion planning, the intuitive application of GCS is to plan through overlapping convex sets and then optimize trajectories within the sets. These sets constitute the vertices of $\mathcal{P}$, and their union is an inner approximation of our planning space. x_v or the continuous variable within each set then is the continuous trajectory through it. Like the paper introducing GCS for Trajectory Optimization paper (GCSTrajOpt) [2] we define x_v as the control points of a Bézier curve to parametrize the continuous trajectory. This choice admits convex path continuity and differentiability constraints, and collision-avoidance of the whole trajectory is guaranteed [2, p.9]. To optimize for the shortest path, GcsTrajOpt minimizes the distance between adjacent control points as a proxy for minimizing the length of the curve [1]. We use the same objective for the convex relaxation and any convex optimization we do.

The above discussion focuses only on the path, but in GcsTrajOpt, x_v has an additional parameter: a time-scaling variable, h. We do not use this additional parameter to avoid nonconvex acceleration constraints that use this time parametrization variable to ensure our trajectories obey physical limits. Instead we use TOPP (Time Optimal Path Parametrization) [17] to generate timed trajectories from our planned spatial paths. As any differentiable path can be navigated under acceleration constraints by slowing down, using TOPP helps maintain feasibility guarantees by keeping nonconvexity out of our constraints. So, for a collision free convex set Q_i, our vertices are of dimension Q_i^{d+1} given Bézier curves of degree d with $d+1$ control points.

B. Parametrizing Configuration Space

In some cases the configuration space benefits from being parametrized to enable building convex sets for GCS or

generating collision free certificates. In this sub-section we review three such parametrizations and associated related works that apply GCS to manipulation motion planning problems. These parametrizations form the three main cases we will tackle with our method in the rest of the paper.

1) IK and Constrained Bimanual Planning: Constrained bimanual manipulation refers to two robot arms that move with a fixed transform between their end effectors. This nonlinear equality constraint in task space prevents using GCS to plan in the full $\mathbb{R}^{14}$ configuration space. Cohn et al. [3] use inverse kinematics to determine the joint angles of a subordinate robot given the end effector position of the leading arm. This parametrization collapses the $\mathbb{R}^{14}$ full joint space over both arms into an $\mathbb{R}^8$ space formed by the leading arm's joints and a redundancy factor (required to generate consistent joint angles for the subordinate arm). Thus, no nonlinear equality constraints are required. However, trying to minimize path length in the $\mathbb{R}^8$ effectively minimizes the path length for the leading arm while ignoring the subordinate arm. As a result, paths are visibly imbalanced and favor the leading arm.

2) Euler Angles and GCS on $\mathrm{SO}(3)$*:* Planning over $\mathrm{SO}(3)$, the set of 3D rotations, is an important motion planning domain in robotics. As any roboticist knows, there are many different ways to represent rotations. Rotation matrices perfectly represent $\mathrm{SO}(3)$, but require bilinear constraints when included in optimization problems (constraints to ensure the validity of the Rotation matrix). Cohn et al. [4] explores different parametrizations to plan over rotations with GCS: Euler angles, axis-angle, and quaternions. The axis-angle and quaternion representations require piecewise-linear approximations, and also require solving two planning problems due to double cover of $\mathrm{SO}(3)$. Though Euler angles are quicker to plan over, the original work observes planning with Euler angles gives longer paths than the quaternion and axis-angle approximations. This discrepancy is due to the distortion of the underlying geometry of $\mathrm{SO}(3)$: distances get arbitrarily large when approaching gimbal lock.

3) Rational Kinematics to Certify Collision Free:

The forward kinematic mapping (needed for checking if a configuration is collision-free) is a trigonometric polynomial. Amice et al. [5] transform this nonconvex relationship into a multi-linear polynomial, using the tangent half-angle substitution $s = \tan\frac{\theta}{2}$, further implying

$$\sin(\theta) = (1-s^2)/(1+s^2),\ \cos(\theta) = (2s^2)/(1+s^2),$$

for $\theta \in (-\pi, \pi)$. They then use this change of coordinates to formulate Semi-Definite Programs (SDPs) that can certify non-collision with task space obstacles for regions in the robot's rational configuration space. This certification can be done for individual convex sets [5], or even an entire trajectory [18].

The *rational* parametrization of kinematics, similar to the Stereographic Projection, is non-isometric. This is most obvious when θ approaches $\pm\pi$, as $\tan\frac{\theta}{2}$ asymptotically approaches to $\pm\infty$. This means that in the parametrized space, as joint values approach limits at $\pm\pi$, distances grow arbitrarily. More generally, near zero, equidistant points in the original space will be closer together in parametrized space than the same points further away from zero. Therefore, when running GCS in this space, a convex formulation of distance in the parametrized variables will be inaccurate. Specifically, we expect that if any joint is moving near $\pm\pi$ away from the point of stereographic projection, any paths planned will be sub-optimal due to the distorted objective.

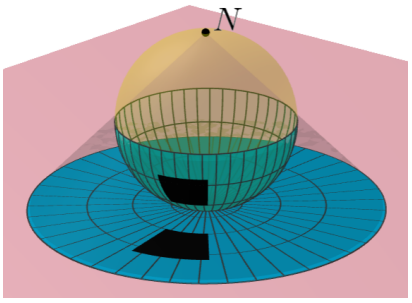

Fig. 2: A stereographic projection about N will project the bottom of the black rectangle as being smaller than the top. A planner operating in the post-projection (parametrized) space and optimizing for short distances would favor the bottom even though both the sides are equal in the original setting. Image generated using code from [19].

For all of these cases, convex objectives being minimized in GCS are in the parametrized spaces and therefore subject to the discussed pathologies. The planner clearly would benefit from the use of nonconvex objectives that represent the true objectives in the original space. This work bridges the gap between using nonconvex objectives and maintaining guarantees of GCS due to convexity.

C. Nonconvexity and GCS

There is precedent for working with nonconvexity in GCS or similar optimization frameworks. The original GcsTrajOpt preprint [20] suggested using convex approximations to incorporate nonconvex objectives and constraints. In line with this suggestion, existing GCS works using the parametrizations from Subsection II-B use a convex surrogate objective to approximate the optimal solution. While this approach preserves convexity, the approximations are inherently heuristic and must be hand-designed. Moreover, optimizing for an approximation means that the optimality of the solution is bounded by the quality of the approximation. The nuances of the approximations can also lead to systematic pathologies, such as the imbalance between arms in the bimanual planning domain.

Another approach is to work with local convex approximations of the nonconvexity. Clark and Xie [21] suggest approximating the nonconvex costs using piecewise-linear approximations and creating smaller sets within which the objective is convex. This approach maintains convexity, but may scale poorly when dealing with complex objectives and finer approximations. Using a mix of biconvex alternation and local convex approximation, Fast Path Planning [22]

handles a bilinear in a similar set-up as GCS. The nonconvexity in this problem is contained to the constraints and is handled using alternation. The nonconvexity being a bilinear nonconvexity is key to enabling this method. Our work aims to enable a broader class of nonconvexity in the objective functions.

Enabling GCS to handle nonconvexity without approximation expands the method's applicability and improves solution quality. We restrict ourselves to nonconvexity in only the objective to improve motion planning results while maintaining guarantees. This restriction does prevent us from handling acceleration constraints, due to their nonconvexity in the GcsTrajOpt formulation. Von Wrangel [23] presents specific strategies for handling certain common nonconvexities in GCS, including acceleration constraints. But the empirical success comes without strong guarantees.

III. Methodology

A. Nonlinear Changes of Coordinates

Each of the parametrizations discussed in Subsection II-B distort the robot's configuration space by introducing a nonlinear change of coordinates. More formally, each domain has a smooth (nonlinear) transformation $\alpha : Q \to C$ that maps x from the more useful *parametrized space* Q to a point $\alpha(x)$ in the original configuration space C. Each of the works used GCS to solve for a trajectory in Q, which then is remapped to C using α to get an actual robot trajectory. However, since α is a nonlinear transformation, the path with minimum length in Q is not guaranteed to be the path of minimum length in C.

The key limitation is that the convex path length cost in Q can be arbitrarily far from the true objective: minimizing distance in C. Using α enables changing coordinates back to the original space C and defining this true (now nonconvex) objective, but the sets and constraints stay convex in the parametrized space Q.

For the constrained bimanual case, $\alpha : \mathbb{R}^8 \to \mathbb{R}^{14}$ is defined as the nonlinear analytic IK function with an original configuration space of both arms' joints ($\mathbb{R}^{14}$) and a parametrized space of one arm's joints and the self-motion of the other arm ($\mathbb{R}^8$). For planning over $\mathrm{SO}(3)$ with Euler angles, $\alpha : \mathbb{R}^3 \to \mathbb{R}^4$ is the standard conversion from Euler angles to quaternions. For planning in the rational parametrization of kinematics, α is defined as $\theta = 2\tan^{-1} s$.

B. Formulating the Optimization

The nonconvex objective using α still needs to be expressed in terms of our decision variables x_v, the control points of the Bézier curve in Q. We cannot directly apply α on x_v to define a distance objective as the remapped control points from Q do not define a same Bézier curve in C. However, any points along the Bézier curve in Q will still be along the same path in C, and any point along the Bézier curve is a convex combination of its control points. Therefore, a piecewise-linear approximation of the curve in Q can be mapped to a piecewise-linear approximation in C using α.

The representative cost can then be the length of this piecewise-linear approximation of the curve. For the bimanual and rational configuration space, this length is the sum of the Euclidean distance between each adjacent pair of points in the full configuration space. For $\mathrm{SO}(3)$, we use the length of the SLERP (Spherical Linear Interpolation) path since the underlying geometry is a sphere. For better results, we square the length of each segment. While this is not the same as the distance between adjacent points, the squared objective is much better numerically for the optimizer and in the limit of an infinitely-fine discretization, these will both produce the same answer [24, p.189]. A higher-resolution approximation of the cost will yield a more accurate approximation of the arc length cost, but require more computational effort. For our experiments, we find that using 10 samples strikes a good balance of accuracy and speed.

The GcsTrajOpt [2] transcription with the original convex objective in the changed coordinates can be written as:

$$\begin{aligned} \min_{\mathbf{x_v}} \quad & \sum_{i=0}^{v}\sum_{j=0}^{d} ||x_{ij} - x_{ij-1}|| \\ \text{s.t.} \quad & x_i \in Q_i, \\ & Q_i \in \mathcal{P} \end{aligned}$$

where x_{ij} is the j^{th} control point of the Bézier curve in set Q_i. Our proposed optimization is:

$$\begin{aligned} \min_{\mathbf{x_v}} \quad & \sum_{i=0}^{v}\sum_{k=0}^{10} f(\alpha(x_{ik}), \alpha(x_{ik-1}))^2 \\ \text{s.t.} \quad & x_i \in Q_i, \\ & Q_i \in \mathcal{P} \end{aligned}$$

where x_{ik} is the k^{th} sampled point in set Q_i and $f(a, b)$ gives the distance between two points a and b in C. Note that both optimizations live in Q with collision free sets $Q_i \subseteq Q$ but have different objectives. This demonstrates how our optimization problem is specifically structured to isolate the nonconvexity in the objective function via the parametrization α.

C. Projected Gradient Descent

To exploit the aforementioned structure, we use *Projected Gradient Descent* (PGD) to maintain guarantees of feasibility and optimality. PGD is an iterative first-order or gradient-based solver with two parts: the gradient step and the projection back into feasibility. PGD steps in the direction of steepest decrease of the objective until a minimum is achieved. If any step yields an infeasible configuration, the solver projects the updated point back into feasible space. Because the constraints remain convex, the projection, a quadratic program finding the closest point in the set, solved with Mosek [25], always returns a solution. Moreover, the convergence of PGD is well-understood if the multiplication factor of the negative gradient is less than or equal to the Lipschitz constant of our objective function [26]. Our objective landscapes do not admit Lipschitz constants, but those that do can leverage this useful guarantee.

D. Solver Performance

Beyond theoretical guarantees, certain implementation details further improve the performance of our solver.

1) Initialization: As PGD finds local minimizers, the solution highly depends on the initialization. Within the GCS workflow, this initialization can come from two distinct candidates: after or during the rounding procedure (the step which projects the convex relaxation result to the near optimal discrete solution). Post processing the solution after rounding is a great way to quickly improve a solution to a fixed discrete path in the parametrized space. We focus on this method, but one could also use this nonconvex optimization for each sampled path as an integral component of the rounding stage.

2) Optimal Step Sizes: Our objectives are too complex to easily identify the Lipschitz constant and theoretically determine a good step size. To avoid experimental testing and manual tuning of step size in classical PGD, we use a variant that searches for and uses an optimal step size. In particular, we use backtracking line search PGD [27], which starts with an upper bound on the step size, and repeatedly halves it until the Armijo condition of sufficient decrease is met. This keeps the solver from overshooting minima while maintaining speed in convergence.

3) Gradient Precompilation: Initially, gradient computations were the majority of the runtime. Precompiling gradients with JAX [28] moves this time cost offline, speeding up the PGD iterations quick. By precompiling gradients for each vertex individually, we ensure we can use the same computations for any start and goal.

4) Affine Projections: With the gradients pre-compiled and checking for feasibility being fast because our feasible space can be expressed as a halfspace intersection, the majority of the time cost comes from the QP projection step. To reduce the number of QP solves and optimize for speed, we initially project onto the affine hull of the feasibility polyhedron. This projection satisfies any equality constraints such as path continuity and differentiability. This projection is much cheaper than the QP projection, since we can efficiently compute the affine hull. (All equality constraints are known explicitly, and the convex sets making up the GCS are positive volume, since they are produced by the IRIS-NP algorithm [29].) In some cases (especially with smaller step sizes), this projection will suffice to push the solution back into feasibility, saving time for the solver. If the point is still infeasible, the solver runs the full QP.

5) Convergence Criteria: The solver tracks the moving average of the cost over the last 5 iterations, and terminates when the average changes by less than 0.5%. The moving average prevents us from terminating early; the cost occasionally jumps for a single iteration before continuing on a significant downward trend. For cases that do not converge, the solver terminates after a maximum of 70 iterations. We hypothesize this occurs when the projection step increases the cost too much, indicating a high Lipschitz constant. In practice for these experiments, optimizations that converged, typically converged well before 70 iterations.

E. More general nonconvex objectives: Curvature

So far the methodology has focused primarily on the special case of eliminating the distortion caused by non-isometric parametrizations. However, we can also optimize for any smooth nonconvex objective, expanding our modeling power. Some examples of useful nonconvex objectives would be minimizing curvature (or other higher-order path derivatives) or penalizing proximity to obstacles.

Penalizing the *curvature* of the path

$$\kappa = \left(||x'||^{-3}\right)\sqrt{||x'||^2\,||x''||^2 - (x'\cdot x'')^2}$$

should help TOPP produce better trajectories, as high curvature paths contain tight turns, that require a slower traversal to stay within acceleration limits. Although such paths might be longer than those produced by a pure shortest-path trajectory, they can be traversed more quickly.

Similar to the cases focusing on undistorting paths, we define this objective over sampled points along the path defined by x_v. Given sampled points, we calculate the curvature of each point and then apply the RealSoftMax (a smooth maximum function) to approximate the maximum curvature of our paths. We expect paths under this optimization will have higher path length but lower duratrion when time-parametrized by TOPP.

IV. Experiments

In this section, we detail the results collected on the three motion planning domains of interest: constrained bimanual, $\mathrm{SO}(3)$ with Euler angles, and rational kinematics. For all of our experiments, we solve the GCS problem with the original convex objective first and then run the projected gradient descent to improve the solution. Interactive recordings of all trajectories and other results are available online at https://shrutigarg914.github.io/pgd-gcs-results/

A. Constrained Bimanual Motion Planning

In this experiment two iiwas navigate a shelf while keeping the transform between end effectors constant, as if they were jointly carrying an object. This setup, shown in Figure 1, matches the setup in [3]. We evaluate the PGD solver on the key start and goal pairs from [3]. The comparison of these benchmark paths before and after optimizing for the nonconvex objective is presented in Table I. The "GCS" column indicates using the convex $\mathbb{R}^8$ objective, the "Distance" column indicates using the nonconvex $\mathbb{R}^{14}$ objective, and the "Curvature + Distance" column indicates a linear combination of the nonconvex $\mathbb{R}^{14}$ objective and the nonconvex curvature cost with a ratio of 8 to 0.01 respectively. While the $\mathbb{R}^{14}$ objective results in the shortest paths, regularizing for lower curvature lengthens paths, but shortens traversal times after TOPP's re-timing. Visually, minimizing this joint objective leads to more rounded paths, as shown in Figure 4.

To quantify the difference in distance traveled between the arms, we define the *imbalance* of a trajectory as $(d_s - d_c)/(d_s + d_c)$, where d_c is the distance traveled by the controlled arm and d_s is the distance traveled by the subordinate arm. When both arms travel comparable distances, the

imbalance distribution centers around 0. When one arm travels much longer distributions than the other, the imbalance metric approaches ± 1 in magnitude. As indicated by Table I this imbalance metric approaches 0 after post-processing under the $\mathbb{R}^{14}$ objective. This shift is also apparent in Figure 3 as the path becomes more centered. The imbalance for the joint curvature and distance optimization is higher than the $\mathbb{R}^{14}$ distance optimization indicating that smoother paths are more imbalanced. This asymmetry likely comes from the same-handedness of the iiwas.

TABLE I: Optimizing over the nonconvex cost improves metrics for the three genchmark trajectories.

Top to Middle			
	GCS	Distance	Distance + Curvature
Trajectory Time	4.889	3.469	**3.243**
R14 Path Length	4.241	**3.766**	3.884
Imbalance	1.046	**0.317**	0.448
Middle to Bottom			
	GCS	Distance	Distance + Curvature
Trajectory Time	5.326	3.08	**2.99**
R14 Path Length	3.325	**3.175**	3.247
Imbalance	0.386	**0.219**	0.248
Top to Bottom			
	GCS	Distance	Distance + Curvature
Trajectory Time	7.48	4.263	**3.99**
R14 Path Length	5.622	**5.048**	5.13
Imbalance	0.493	**0.199**	0.217

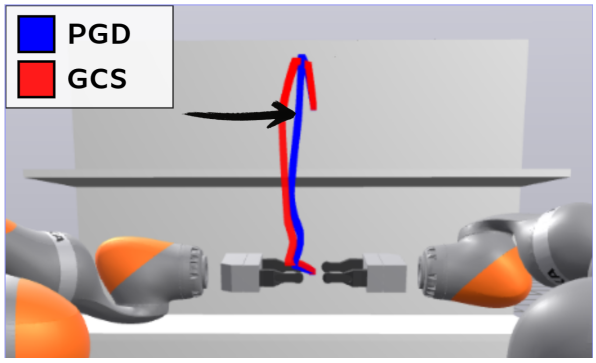

Fig. 3: Paths become more centered because the nonconvex objective used by the PGD considers the distance traveled by both arms, rather than just the controlled arm in the original convex GCS objective.

For a more comprehensive analysis, we randomly sample 100 start and end points from the valid and reachable configuration space. Paths generated are on average 20.60% shorter in the $\mathbb{R}^{14}$ configuration space after applying our postprocessing step. The total time taken to navigate these paths is on average 31.02% shorter. The imbalance shifts towards 0, indicating that the paths for the subordinate arm are more comparable to the leading arm after the nonconvex optimization. These improvements took an average of 0.0554 seconds of compute (approximately 13.7 iterations that took varying time based on length and complexity of the path) in addition to the 2.133 seconds that the surrogate convex optimization takes.

80% of the runtime is solving QP projections. The affine projection is only useful when step size is fixed. When using backtracking to determine step size, the projection onto the affine hull is almost never sufficient. This observation indicates to us that at our boundary the gradients are consistently pointing outward. This is not unexpected, since the collision-avoidance constraints are active at the boundary, and moving closer to obstacles generally allows a shorter path length.

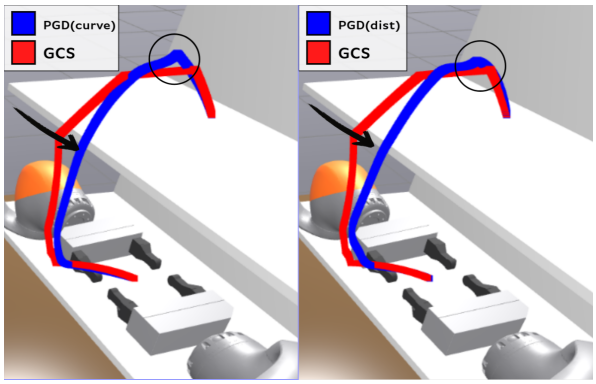

Fig. 4: Paths optimized jointly for curvature and distance result in quicker trajectories, even though they are longer (as seen near the top). Note that the curvature-regularized path does not pass as close to the edge of the shelf.

B. Planning over $\mathrm{SO}(3)$

To plan shortest paths over $\mathrm{SO}(3)$, we set up the same charts and convex regions as in [4] for the Euler angles, quaternions, and axis-angle parametrizations. The latter two leverage piecewise-linear approximations of the original $\mathrm{SO}(3)$ space and are there for reference. We run PGD on the Euler angles setting only. For Euler angles, the GCS graph is fully connected and is optimizing Euclidean distance within each set, so the shortest path between any two points will be a linear path, regardless of the order of our Bézier curves. For time-efficiency, we generate order one GCS solutions and initialize the PGD solver with the control points evenly spaced along each line segment.

Given there are no obstacles, SLERP gives the shortest path between any two orientations in this setting. We use it for a closed-form ground truth distance for each start and goal pair. Figure 5 shows the distribution of relative error in the path length when using the three representations of $\mathrm{SO}(3)$ to plan across 125 random start and goal pairs, along with the PGD post-processing on the Euler angles paths. The distribution of error for Euler angles shifts significantly closer to 0 after running PGD on it. The relative error decreases by 42.5% on average. This improvement has a bimodal distribution: for some paths the PGD greatly improves the solution, whereas for others, there is little improvement to be made. The latter might be local minima, where a different discrete path admits a better trajectory.

These improvements require on average an additional 4.08 seconds on top of the 17.28 seconds to generate the original solutions for the Euler angles. Of this time, the

solver only was running for 0.62 seconds. The remaining 3.46 seconds were set-up time to re-use the vertices and Bézier curves from the original solution; this time could be further optimized. Comparatively, the time to generate GCS paths when planning with axis-angles is 41.10 seconds. At a lower resolution quaternions take 16.73 seconds, but for higher resolutions, their solve time is on the order of minutes. Our method offers a good payoff between finding shorter paths with Euler angles while still being faster than the more accurate axis-angle and quaternion representations. Euler angles are also currently the only parametrization compatible with the IRIS-NP algorithm [29] for growing collision-free regions in the presence of obstacles.

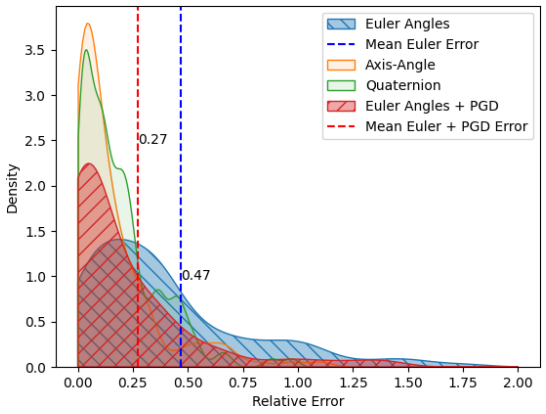

Fig. 5: Comparing the distributions of relative error of paths with respect to the SLERP distance between start and goal orientations. The PGD significantly improves the results of the Euler angles parametrization.

C. Rational Parametrizations of Robot Kinematics

We have two experimental settings that use the rational kinematics parametrization. One is a 3 degree-of-freedom iiwa (four of the joints are locked) that moves within a vertical 2D plane. The other is a 7 degree-of-freedom iiwa mounted on a table, as shown in Figure 1. The nominal position (i.e. point of projection) for both iiwas is when the arms stand straight up with all joint angles at 0. All the regions in the 3DoF case are certified to be completely collision-free using the Certified IRIS algorithm [5]. All the trajectories in the 7DoF setting can be certified using [18].

For the 3DoF planar iiwa, visually the paths become less biased towards the point of projection. For example, in Figure 6, the original paths in red spike towards the nominal position, but after the PGD refinement, the end effector has less extraneous motion. Quantitatively, when measured across 100 randomly chosen start and goal points navigating the shelves, most paths do not have a lot of improvement to be made. On average, the paths get about 0.2% shorter and most terminate within 7 iterations and 0.22 seconds. For the specific path in Figure 6, the numerical improvement is a 1.2% shorter path length. Weaker numerical results for this case is expected as the distortion of configuration space is most intense near the boundaries of the configuration space (i.e. near the joint limits), so the average path does not have a lot of room for improvement.

For the 7dof iiwa, the projected gradient descent on random paths in configuration space between the bins results in 3.89% shorter in path lengths and a 4.74% shorter trajectory times. When one or more joints travel near their limits, these improvements are higher. For example, Figure 6 shows a trajectory that gets 10.8% shorter and 17.6% faster.

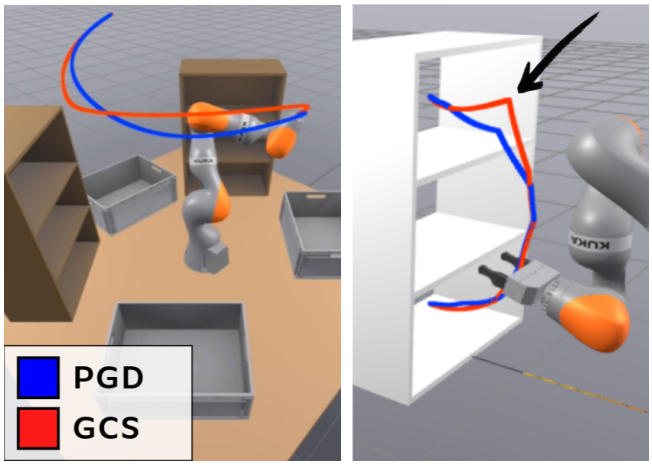

Fig. 6: The 3DoF iiwa (right) shows a decrease in skew towards the nominal position as compared to the original GCS solution (in red). The 7DoF iiwa (left) shows an improvement in path length before and after the post processing for a random selection of start and goal points.

V. Discussion

We have presented a method to solve GCS problems with nonconvex objectives, granting greater modeling freedom and yielding better motion plans. By keeping the constraints convex, we maintain the feasibility guarantees of GCS and avoid the inconsistency typical of nonconvex optimizations.

Our method is particularly effective when accounting for the distortion from nonlinear parametrizations of planning spaces. In constrained bimanual motion planning, our post-processing step produces paths that are more balanced between the arms, 20% shorter on average, and 31.02% faster after being time-parametrized. For Euler angles, the paths are 40% shorter on average. In addition to undistorting paths from parametrized spaces, the approach enables the optimization of a path based on general nonconvex objectives. Optimizing over curvature, we find paths with greater radii of curvature, facilitating quicker traversal.

For rational kinematic parametrizations, the lack of significant improvement in path length in the average case suggests that the distortion from the stereographic projection is not usually a major problem. Thus, the benefits of planning in this parametrization of configuration space (enabling rigorous certification) plausibly outweigh the minor costs. In the worst case, our method can produce strong improvements, and when there is not much space for improvement, the solver terminates quickly

An obvious limitation of the proposed methods is the added computation time. We find the time taken by the solver quite satisfactory given that we used a Python based custom PGD. The use of commercial solvers, code improvements, and a compiled language could make a mature implementation much faster. In cases where a solution requires strong guarantees and high quality trajectories (such as in path planning and control of surgery robots), a post-processing step will certainly be worth the additional runtime.

Future work can include larger scale parallelization, especially during the rounding stage. Just as cuRobo has shown incredible results by solving many nonconvex trajectory optimization problems in parallel, our post-processing stage could be integrated with a parallel rounding scheme to get better solutions more quickly. This step could also be used in an Anytime Motion Planning framework [30], where the later parts of a trajectory are refined as the earlier parts are traversed. Another possibility is using the nonconvex objectives with incremental search methods such as GCS* [31].

Another line of work might focus on designing better convex surrogates. These convex surrogates still play an important role during the convex relaxation and initialization stages for the nonconvex optimization. When there are clear deficiencies in the convex surrogate (such as in the original constrained bimanual case), one can try to hand-design a better surrogate. When it is unclear how to make a better surrogate, one could potentially generate them automatically using learning-based approaches.

REFERENCES

[1] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, p. 507–532, Feb. 2024. [Online]. Available: http://dx.doi.org/10.1137/22M1523790

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

[3] T. Cohn, S. Shaw, M. Simchowitz, and R. Tedrake, "Constrained bimanual planning with analytic inverse kinematics," in *Proceedings of International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[4] T. B. Cohn, "Motion planning along manifolds with geodesic convexity and analytic inverse kinematics," Master's thesis, Massachusetts Institute of Technology, 2024.

[5] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," 2022. [Online]. Available: https://arxiv.org/abs/2205.03690

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[7] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[10] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[11] D. Garg, M. Patterson, C. Francolin, C. Darby, G. Huntington, W. Hager, and A. Rao, "Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a radau pseudospectral method," *Computational Optimization and Applications*, vol. 49, no. 3, pp. 335–358, 2011.

[12] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002. [Online]. Available: https://doi.org/10.1137/S1052623499350013

[13] M. Toussaint, "A tutorial on newton methods for constrained trajectory optimization and relations to slam, gaussian process smoothing, optimal control, and probabilistic inference," in *Geometric and Numerical Foundations of Movements*. Springer, 2017, pp. 361–392.

[14] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2009, pp. 489–494.

[15] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 4569–4574.

[16] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8112–8119.

[17] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[18] A. Amice, P. Werner, and R. Tedrake, "Certifying bimanual rrt motion plans in a second," *arXiv preprint arXiv:2309.08770*, 2023.

[19] M. Zamboj, "Interactive 4-d visualization of stereographic images from the double orthogonal projection," in *Proceedings of the 19th International Conference on Geometry and Graphics 2020*, ser. Advances in Intelligent Systems and Computing, L. Y. Cheng, Ed. Cham, Switzerland: Springer, 2021, vol. 1296. [Online]. Available: https://doi.org/10.1007/978-3-030-63403-2_11

[20] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *arXiv preprint arXiv:2205.04422*, 2022.

[21] C. L. Clark and B. Xie, "Planning optimal collision-free trajectories with non-convex cost functions using graphs of convex sets," in *IROS 2023 Workshop on Task and Motion Planning: from Theory to Practice*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023.

[22] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," in *IEEE Transactions on Robotics (TRO)*, 05 2023.

[23] D. von Wrangel, "Guiding nonconvex trajectory optimization with hierarchical graphs of convex sets," Master's thesis, Massachusets Institute of Technology, May 2024.

[24] M. P. Do Carmo and F. J. Flaherty, *Riemannian Geometry*. Springer, 1992, vol. 2.

[25] M. ApS, *The MOSEK optimization toolbox for Drake manual. Version 10.1.*, 2024. [Online]. Available: http://docs.mosek.com/latest/toolbox/index.html

[26] P. Hansen and B. Jaumard, "Lipschitz optimization," in *Handbook of Global Optimization. Nonconvex Optimization and Its Applications*, R. Horst and P. Pardalos, Eds. Boston, MA: Springer, 1995, vol. 2. [Online]. Available: https://doi.org/10.1007/978-1-4615-2025-2_9

[27] J. Nocedal and S. J. Wright, "Chapter 3: Line search methods," in *Numerical Optimization*, 2nd ed. Springer, 2006, pp. 37–60.

[28] J. Contributors, "Jax: Composable transformations of python+numpy programs," https://github.com/google/jax, 2018.

[29] M. Petersen and R. Tedrake, "Growing convex collision-free regions in configuration space using nonlinear programming," *arXiv preprint arXiv:2303.14737*, 2023.

[30] I. Mishani, H. Feddock, and M. Likhachev, "Constant-time motion planning with anytime refinement for manipulation," *arXiv preprint arXiv:2311.00837*, 2023.

[31] S. Y. C. Chia, R. H. Jiang, B. P. Graesdal, L. P. Kaelbling, and R. Tedrake, "Gcs*: Forward heuristic search on implicit graphs of convex sets," *arXiv preprint arXiv:2407.08848*, 2024.