

Graphs of Trajectory Bundles for Nonlinear Motion Planning

Joris Verhagen¹, Jon Arrizabalaga², Jana Tumova¹, and Zachary Manchester²

Abstract—We present the Graphs of Trajectory Bundles (GTB) method for motion planning of systems with nonlinear and black-box dynamics, combining sampling-based motion planning with trajectory optimization and graph search. Based on a graph-of-convex-sets decomposition of the workspace, GTB iteratively constructs a graph of trajectory bundles that captures the local dynamics of the system at any vertex. By solving a shortest-path problem on this graph, we find maximally dynamically feasible iterates that satisfy the system’s dynamics in the limit and any constraints such as collision avoidance or additional safety or specification constraints by construction of the underlying graph. As a result, GTB can handle highly nonlinear, unstable, and even black-box dynamics with complex collision and specification constraints, making it a powerful tool for motion planning of complex systems. **code**
Keywords: graphs of convex sets, derivative-free optimization

I. INTRODUCTION

We are interested in solving the general motion planning problem of the form

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^N J(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\ & x_k \in \mathcal{W}_{\text{free}} \\ & u_k \in \mathcal{U}, \quad x \in \mathcal{C}, \end{aligned} \quad (1)$$

where $\mathcal{W}_{\text{free}}$ denotes a (non-convex) collision-free set, $\mathcal{U}$ is a convex actuation set, and $\mathcal{C}$ is an additional constraint set that, e.g., encodes final state, safety constraints, or logic specifications. The system’s dynamics, $f(x_k, u_k) : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$, are assumed continuous but otherwise permitted to be nonlinear or *black-box*: required to only be available as a forward map (a numerical integrator, a physics engine, or a learned model) as Jacobians may be unavailable, unreliable, or non-smooth.

The nonconvexities from 1) collision avoidance constraints and 2) nonlinear or black-box dynamics make this problem especially difficult to solve. The first challenge can be reasoned over efficiently by considering collision-avoidance as traversal over a graph of convex, collision-free sets, which has been shown to yield globally optimal solutions for differentially flat systems [1]. While this method is not applicable for systems that are differentially flat with actuation constraints, general nonlinear, or black-box since the dynamic constraints are not convex, its decision-making

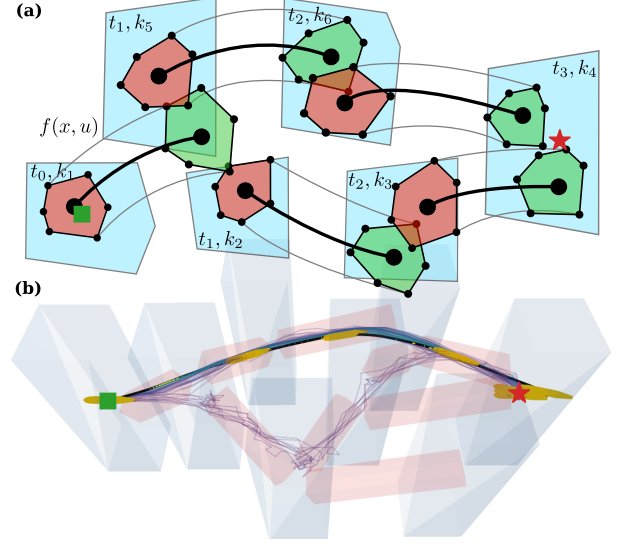


Fig. 1: The Graph of Trajectory Bundle (GTB) method poses the motion planning problem as a shortest path problem in dynamics space. On each vertex of a graph of convex sets (shaded blue), states are sampled (shaded red) and transported through the dynamics $f(x, u)$ to form a bundle (shaded green), themselves being convex sets. At each iteration, GTB finds the interpolators of the sequence of bundles. (b) a high-dimensional underwater robot through a 3D maze with iterations of GTB taking different paths.

over sequences of sets via convex optimization is still extremely powerful. We build upon this idea to reason about the second challenge by collecting sampled trajectories into *bundles* [2] and constructing a *graph of trajectory bundles* over which we can reason about the dynamics in a convex manner via interpolation over the sampled trajectories. This sampling- and optimization-based approach iteratively refines the trajectory by resampling around the current solution, yielding solutions that satisfy collision-avoidance and specification constraints by construction, and dynamic feasibility in the limit of full algorithm convergence.

Concretely, we formulate Eq. (1) as a *graph of trajectory bundles* through which we iteratively seek the path of minimal accumulated dynamic infeasibility, i.e., the shortest path in dynamics space from start to goal, while iteratively resampling and refining the dynamics approximation.

A. Contributions

This work makes the following specific contributions:

- **Graphs of Trajectory Bundles:** We propose a novel formulation of the problem in Eq. (1) as a graph-planning problem over sampled trajectory bundles.
- **Sequential Convex Solution Approach:** We solve the Graphs-of-Trajectory-Bundles problem with a sequential convex programming approach on a graph of sampled trajectories.

¹Authors are with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden {jorisv, tumova}@kth.se

²Authors are with the Department of Aeronautics and Astronautics, MIT, Cambridge, MA, USA {jonarri, zacm}@mit.edu

B. Related Work

The problem in Eq. (1) is incredibly general and has been studied extensively in many shapes and forms. Arguably its most fundamental challenge relates to nonconvexities, making it difficult to find optimal or even feasible solutions.

Popular approaches that maintain a global perspective over the solution space include graph-based planning methods such as Probabilistic Roadmaps (PRM) [3] or A*. These methods search for a path over a graph of feasible states. By constructing vertices of feasible states and edges of feasible transitions, these methods can find globally optimal solutions, limited by the resolution of the graph. Recently, the Graph of Convex Sets (GCS) method [1] has been proposed to solve Eq. (1) for differentially flat dynamics. GCS generalizes discrete graph search to problems where vertices and edges can be convex sets and convex costs, respectively. Extensions have been proposed to enable non-convex constraints of Bézier trajectory optimization [4] and piecewise affine approximations of nonlinear edge costs [5], hybrid contact modes for manipulation [6], and forward search for contact-implicit decision-making [7]. Yet none tackle explicit nonlinear or black-box dynamics.

Trajectory optimization methods, such as nonlinear programming [8] and sequential convex programming [9], [10], have been widely used to find locally optimal solutions to Eq. (1). These methods rely on gradient-based optimization of a differentiable model; they are sensitive to initial conditions, may converge to local minima, and do not apply when f is non-smooth or available only through simulation.

Derivative-free optimization [11], [12] and sampling-based trajectory optimization can optimize trajectories without requiring gradient information or explicit dynamics. The most popular example in robotics is likely Model Predictive Path Integral control (MPPI) [13]. These methods naturally deal with nonlinear dynamics and, with advances in simulators and parallelization, can scale to high-dimensional problems [14] and multi-modal reasoning [15]. Single-shooting methods are, however, unsuitable for long-horizon planning and cannot deal with explicit state constraints [13]. Additionally, for planning in complex geometry, the *tail-wagging-the-dog* problem can lead to infeasible solutions [2]. The Trajectory Bundle Method (TBM) [2] generalizes MPPI by using a sequence of sample trajectories, referred to as trajectory bundles. This allows constraint satisfaction at every knot point, making it suitable for long-horizon problems with more complex constraints. However, it is still a local method, requiring a good initial guess to converge to local minima.

II. PRELIMINARIES

We present Graphs of Trajectory Bundles (GTB), which combines the global reasoning of GCS for obstacle avoidance with the local derivative-free reasoning of the Trajectory Bundle Method (TBM) for dynamic feasibility.

A. Graphs of Convex Sets

Graph of Convex Sets (GCS) [16] generalizes the shortest-path problem by constraining vertices to convex sets and

defining convex costs between them. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph in which every vertex $v \in \mathcal{V}$ is equipped with a compact convex set $\mathcal{X}_v$ and a variable $x_v \in \mathcal{X}_v$, and every edge $e = (v, v') \in \mathcal{E}$ a convex cost $\ell_e(x_v, x_{v'})$. The shortest path problem seeks both a path p from a source x_s to a target x_t and the points along it,

$$\begin{aligned} \min_{p, \{x_v\}} \quad & \sum_{e=(v,v') \in \mathcal{E}_p} \ell_e(x_v, x_{v'}) \\ \text{s.t.} \quad & p \in \mathcal{P}, \\ & x_v \in \mathcal{X}_v, \quad \forall v \in \mathcal{V}, \end{aligned} \quad (2)$$

where $\mathcal{P}$ is the set of source-to-target paths and $\mathcal{E}_p$ the edges traversed by p . Fixing x_v makes (2) a standard shortest path problem with a known Linear Programming (LP) formulation, while fixing the path p makes it a convex program in x_v . Equation (3) below is a concrete formulation of (2) in which the path variable p is replaced by a binary flow $\phi_e \in \{0, 1\}$ on every edge,

$$\begin{aligned} \min_{\phi, x} \quad & \sum_{e \in \mathcal{E}} \phi_e \ell_e(x_v, x_{v'}) \\ \text{s.t.} \quad & \sum_{e \in \mathcal{E}_s^{\text{out}}} \phi_e = \sum_{e \in \mathcal{E}_t^{\text{in}}} \phi_e = 1, \\ & \sum_{e \in \mathcal{E}_v^{\text{in}}} \phi_e = \sum_{e \in \mathcal{E}_v^{\text{out}}} \phi_e \leq 1, \quad \forall v \in \mathcal{V} \setminus \{s, t\}, \\ & \phi_e \in \{0, 1\}, \quad x_v \in \mathcal{X}_v, \quad \forall v \in \mathcal{V}_p, \end{aligned} \quad (3)$$

where the flow conservation and degree constraints confine ϕ to the s - t path polytope. The bilinear product $\phi_e \ell_e(x_u, x_v)$ can be written in a relaxed convex form via a binary relaxation, $\phi_e \in [0, 1]$, and a perspective transformation. We refer to [1] for details. The GCS formulation is, empirically, a tight relaxation meaning that (3) can be solved efficiently via a single second-order cone program (SOCP).

GCS is applied to motion planning by letting $\mathcal{G}$ be a coverage of the collision-free space with convex regions $\mathcal{Q}_k$, one per vertex, and letting $\mathcal{X}_v$ hold the control points of a Bézier curve constrained to $\mathcal{Q}_k$ [16]. This requires the trajectory to be an affine function of the vertex variables: the curve is planned in a differentially flat output and dynamic feasibility is assumed to follow from flatness. This assumption means that nonlinear or black-box dynamics of (1) are excluded; moreover, even for a flat model, actuation limits and state-dependent couplings can leave the planned flat-output path infeasible for the real system.

B. Function approximation by interpolation

To enable (sequential) convex motion planning on the aforementioned nonlinear and black-box systems requires a convex approximation of the dynamics and constraint functions. An affine approximation can be obtained by Taylor expansion, but requires Jacobians, which may be expensive, unreliable, or unavailable. An alternative is derivative-free approximation by interpolating sampled function values [17]. Following the notation in [2], a weight vector in the simplex $\alpha \in \Delta^{M_s} = \{\alpha \in \mathbb{R}_{\geq 0}^{M_s}, \mathbf{1}^\top \alpha = 1\}$ can interpolate over M_s samples of a function $p : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_p}$. For any p , we can

form an affine approximation around $\bar{y}$ by drawing samples around $\bar{y}$ and placing it in a data matrix

$$W_y = [y^{(1)} \dots y^{(M_s)}], \quad W_p = [p(y^{(1)}) \dots p(y^{(M_s)})],$$

where the affine approximation follows

$$y = W_y \alpha, \quad p(W_y \alpha) \approx W_p \alpha. \quad (4)$$

This derivative-free approximation is exact at every sample and an affine approximation in between.

C. The Trajectory Bundle Method

The Trajectory Bundle Method (TBM) [2] applies this construction to trajectory optimization. The trajectory is discretized at N knots, $i = 0, \dots, N-1$. Suppose we draw M_s state-control samples $(x_i^{(m)}, u_i^{(m)})$ at every knot i . We roll out the dynamics, $f_i^{(m)} = f(x_i^{(m)}, u_i^{(m)})$, and evaluate a residual (e.g. actuation cost and tracking error), $r_i^{(m)} = r(x_i^{(m)}, u_i^{(m)})$, and constraint (e.g. collision avoidance), $c_i^{(m)} = c(x_i^{(m)}, u_i^{(m)})$, collecting the data matrices

$$\begin{aligned} W_x^{(i)} &= [x_i^{(1)} \dots x_i^{(M_s)}], & W_u^{(i)} &= [u_i^{(1)} \dots u_i^{(M_s)}], \\ W_f^{(i)} &= [f_i^{(1)} \dots f_i^{(M_s)}], & W_r^{(i)} &= [r_i^{(1)} \dots r_i^{(M_s)}], \\ W_c^{(i)} &= [c_i^{(1)} \dots c_i^{(M_s)}]. \end{aligned}$$

The pair $(W_x^{(i)}, W_f^{(i)})$ is a *bundle* at knot i : a sampled, local approximation of the dynamics (the green-red pair in Fig. 1; formed per region and knot in Fig. 2c). If the weight vector α_i is shared by all data matrices of knot i , trajectory optimization can be approximated by the convex program

$$\min_{\alpha, s, w} \sum_{i=0}^{N-1} \|W_r^{(i)} \alpha_i\|_2^2 + \mu \sum_{i=0}^{N-2} \|s_i\|_1 + \mu \sum_{i=0}^{N-1} \|w_i\|_1 \quad (5)$$

$$\begin{aligned} \text{s.t. } & W_x^{(i+1)} \alpha_{i+1} = W_f^{(i)} \alpha_i + s_i, \\ & W_c^{(i)} \alpha_i + w_i \geq 0, \\ & \alpha_i \in \Delta^{M_s}, \quad \forall i, \end{aligned}$$

where $\mu \in \mathbb{R}_{>0}$ is a penalty weight. This is a convex problem and an affine approximation of the original problem and returns a trajectory that minimizes the dynamics residual, $\|s_i\|_1$, and constraint violation, $\|w_i\|_1$, while minimizing the cost. Solving Eq. (5), resampling around the new trajectory and re-solving gives a derivative-free SCP method that converges to a local minimum of the original problem [2].

III. THE GRAPHS OF TRAJECTORY BUNDLES ALGORITHM

We solve (1) with a sequential, sampling-based scheme that we call the *Graphs of Trajectory Bundles* (GTB) method, summarized in Algorithm 1 and depicted in Fig. 2. We start with a *workspace* graph of convex sets, $\mathcal{G}$, whose vertices are feasible convex regions at each time knot. On this *workspace* graph, we construct, iteratively, a graph of sampled dynamics which, for each vertex in $\mathcal{G}$, approximate the dynamics with an implicit trust region. In other words, each vertex in $\mathcal{G}$ is replaced by a *projected bundle*: the convex hull of

Algorithm 1 Graphs of Trajectory Bundles (GTB)

Require: workspace graph $\mathcal{G}$, source x_s , target x_t

- 1: Initialize sampling scale σ
- 2: Set initial vertex solutions $(\bar{x}_{i,k}, \bar{u}_{i,k})$
- 3: **repeat**
- 4: Build projected bundles on each vertex, $\mathcal{W}_{i,k}$
- 5: Construct sampled graph $\hat{\mathcal{G}}_\sigma$, solve relaxation
- 6: Obtain vertex probabilities from $(\phi_e, \alpha_e), \forall e \in \mathcal{E}$
- 7: Overwrite vertex solutions $(\bar{x}_{i,k}, \bar{u}_{i,k})$ and σ
- 8: **until** dynamic infeasibility $\leq \text{tol}$
- 9: **return** path p and trajectory (x, u) rounded from ϕ

samples projected onto that region and propagated through the dynamics (Section IV). GTB therefore solves a sampled, inner approximation of the workspace GCS that considers the dynamics and is refined over the iterations of the algorithm via a contracting process. Each iteration i executes four steps:

- 1) **Sample & project** — perturb the current per-vertex solution at every knot and every region and project the samples onto that region, Fig. 2b.
- 2) **Build bundles** — propagate every sample through the dynamics to obtain forward states, durations and cost residuals, Fig. 2c, Line 4.
- 3) **GTB region selection & trajectory** — solve a Graph of Convex Sets problem whose vertices are projected bundles, finding a shortest path with respect to *dynamic infeasibility* by interpolating between the samples in each bundle, Fig. 2d, Line 5.
- 4) **Fractional flow updates** — update the vertex solutions for subsequent iterations and compute edge- and vertex probabilities for graph cutting, Line 7.

The sample scaling is annealed across iterations, so that the bundles contract around the persistent vertex solutions; this annealing improves the local estimates of the dynamics and drives the relaxation towards tightness.

IV. TRAJECTORY BUNDLES AS SETS IN A GRAPH

We now describe the *workspace* graph and the sampled inner approximation used in GTB.

A. Directed Acyclic Graph of Convex Sets

Let $\{\mathcal{Q}_k\}_{k=1}^K$ be connected convex regions that sufficiently covers $\mathcal{W}_{\text{free}}$. We build a layered Directed Acyclic Graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with

$$\mathcal{V} = \{(i, k) : i \in \{0, \dots, N-1\}, k \in \{1, \dots, K\}\} \cup \{s, t\},$$

i.e. one *layer* per knot i and one vertex per region k within each layer, plus a source s and a target t . Edges only connect consecutive layers,

$$\begin{aligned} \mathcal{E}_{\text{int}} &= \{(i, j) \rightarrow (i+1, k) : j \cap k \neq \emptyset\}, \\ \mathcal{E}_{\text{src}} &= \{s \rightarrow (0, k) : k \in \mathcal{S}_s\}, \\ \mathcal{E}_{\text{tgt}} &= \{(N-1, k) \rightarrow t : k \in \mathcal{S}_t\}, \\ \mathcal{E} &= \mathcal{E}_{\text{int}} \cup \mathcal{E}_{\text{src}} \cup \mathcal{E}_{\text{tgt}}, \end{aligned}$$

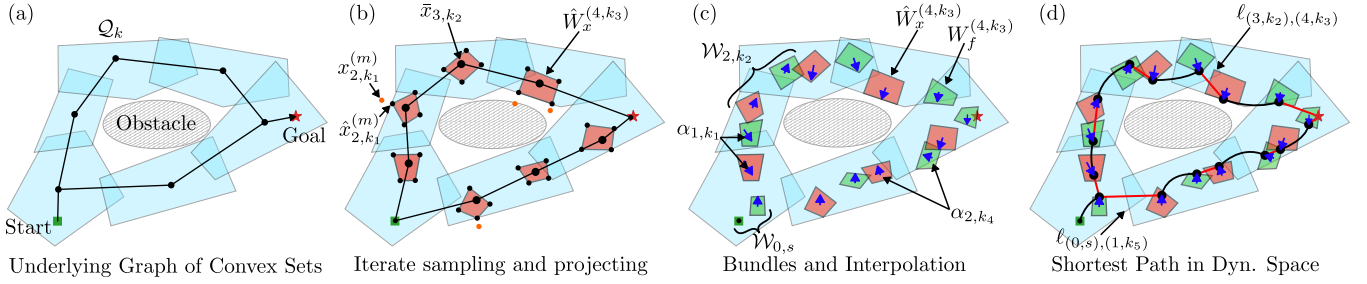


Fig. 2: The workspace graph of convex sets (a) and one iteration of the Graphs of Trajectory Bundles (GTB) algorithm (b,c,d). In (b) we perturb the current vertex solution $\bar{x}_{i,k}$ (red samples) and project onto the region Q_k (blue) to form the *projected bundle* $\hat{W}_x^{(i,k)}$. In (c) each projected sample is rolled through the dynamics to form the *forward bundle* $\hat{W}_f^{(i,k)}$ (green); the two together are the bundle $\mathcal{W}_{i,k}$ at vertex (i,k) of $\hat{\mathcal{G}}_\sigma$. The vertex variables are the weights $\alpha_{i,k}$, chosen in (d) to minimize dynamic infeasibility (red lines) along a path (black curves) from Source to Target (11).

where $j \cap k \neq \emptyset$ denotes intersection of region j and k , permitting self-transitions ($j = k$). The source connects to the source regions $\mathcal{S}_s = \{k : x_s \in Q_k\}$ at layer 0 and the target regions $\mathcal{S}_t = \{k : x_t \in Q_k\}$ at layer $N-1$ connect to the target, so that any $s \rightarrow t$ trace is a valid region sequence $(k_0, \dots, k_{N-1})$ from source to target. In GTB, the graph topology is fixed but the convex sets themselves can change; each iteration creates new bundles, providing a new (improved) approximation of the dynamics.

B. Vertex solutions

We let every core vertex (i,k) in $\mathcal{G}$ carry a persistent *vertex solution* $(\bar{x}_{i,k}, \bar{u}_{i,k})$ (Fig. 2b): the state and control the planner currently expects at knot i if the trajectory passes through region k . At the first iteration there is no trajectory information, so they are initialized to the center of Q_k and $\mathcal{U}$. Every later iteration overwrites $(\bar{x}_{i,k}, \bar{u}_{i,k})$ with the flow-weighted bundle point $(\hat{W}_x^{(i,k)} \alpha_{i,k}, \hat{W}_u^{(i,k)} \alpha_{i,k})$ (Section V). It is around these vertex solutions that the sampling procedure creates a sampled inner approximation of $\mathcal{G}$, creating a new graph $\hat{\mathcal{G}}_\sigma$ whose vertices are the projected bundles.

C. Bundle Construction

At each vertex (i,k) we draw M_s perturbation samples around the vertex solution,

$$x_{i,k}^{(m)} = \bar{x}_{i,k} + \varepsilon_{i,k}^{(m)}, \quad \varepsilon_{i,k}^{(m)} \sim \mathcal{S}(\sigma), \quad m = 1, \dots, M_s, \quad (6)$$

and analogously sample controls $u_{i,k}^{(m)}$ around $\bar{u}_{i,k}$. Here, $\mathcal{S}(\sigma)$ denotes a perturbation scheme with scale σ . This includes, e.g., isotropic Gaussian sampling $\mathcal{S}(\sigma) = \mathcal{N}(0, \sigma^2 I)$, uniform sampling on the hypercube $\mathcal{S}(\sigma) = \mathcal{U}([- \sigma, \sigma]^{n_x})$, or deterministic (pointwise) perturbations on a grid of radius σ . These samples are then projected onto the corresponding convex set, creating projected bundles (Fig. 2b);

$$\begin{aligned} \hat{x}_{i,k}^{(m)} &= \Pi_{Q_k}(x_{i,k}^{(m)}), \quad \hat{u}_{i,k}^{(m)} = \Pi_{\mathcal{U}}(u_{i,k}^{(m)}), \\ \hat{W}_x^{(i,k)} &= [\hat{x}_{i,k}^{(1)} \dots \hat{x}_{i,k}^{(M_s)}] \in \mathbb{R}^{n_x \times M_s}, \\ \hat{W}_u^{(i,k)} &= [\hat{u}_{i,k}^{(1)} \dots \hat{u}_{i,k}^{(M_s)}] \in \mathbb{R}^{n_u \times M_s}. \end{aligned}$$

Because each $\hat{x}_{i,k}^{(m)} \in Q_k$ by construction, the bundle requires *no* additional region-membership constraints downstream (Section IV).

Each projected state/control sample $(\hat{x}_{i,k}^{(m)}, \hat{u}_{i,k}^{(m)})$ is propagated through the dynamics with duration $t_{i,k}^{(m)}$ (via e.g. integrating known dynamics or inferring a simulator or black-box dynamics model). This leads to a forward state $f_{i,k}^{(m)} = f(\hat{x}_{i,k}^{(m)}, \hat{u}_{i,k}^{(m)}, t_{i,k}^{(m)})$, all of which are collected into a forward bundle and a duration bundle .

$$\begin{aligned} W_f^{(i,k)} &= [f_{i,k}^{(1)} \dots f_{i,k}^{(M_s)}] \in \mathbb{R}^{n_x \times M_s} \\ W_t^{(i,k)} &= [t_{i,k}^{(1)} \dots t_{i,k}^{(M_s)}] \in \mathbb{R}_{\geq 0}^{1 \times M_s} \end{aligned}$$

together with residual bundles [2]. Bundles now live at vertices (i,k) of the graph rather than at knots as in [2]. Collision avoidance at knot $i+1$ is guaranteed by the projected bundle $\hat{W}_x^{(i+1,k)}$, not by $W_f^{(i,k)}$. The vertex bundles represent the local, sampled approximation of $f(x, u)$ at (i,k) around which the convex subproblem is built.

D. A Graph of Trajectory Bundles

The workspace graph $\mathcal{G}$ of Section IV-A is a graph of convex sets in the classical sense: each vertex carries a convex region Q_k . Solving a GCS over those regions directly would require a representation of the nonlinear dynamics valid over all of Q_k , which we do not have. Instead we solve a GCS over the *projected bundles*. First, we stack the projected state, control, duration, cost, and forward state of every sample into a *lifted generator* $g_{i,k}^{(m)} \in \mathbb{R}^{2n_x + n_u + 2}$,

$$g_{i,k}^{(m)} = [\hat{x}_{i,k}^{(m)\top} \hat{u}_{i,k}^{(m)\top} t_{i,k}^{(m)} r_{i,k}^{(m)} f_{i,k}^{(m)\top}]^\top. \quad (7)$$

We then define a V-polytope from these lifted generators with vertices $V_{i,k} = [g_{i,k}^{(1)} \dots g_{i,k}^{(M_s)}]$ yielding $\mathcal{W}_{i,k}$ which is the convex hull of *lifted generators* in which any interior point is implicitly a convex combination of the generators, i.e. a weight vector $\alpha_{i,k} \in \Delta^{M_s-1}$ exists implicitly,

$$\mathcal{W}_{i,k} = \text{conv}\{g_{i,k}^{(m)}\}_{m=1}^{M_s} = \{V_{i,k} \alpha_{i,k} : \alpha_{i,k} \in \Delta^{M_s-1}\}. \quad (8)$$

Although f is nonlinear, each generator $g_{i,k}^{(m)}$ is fixed data so $V_{i,k} \alpha_{i,k}$ is affine in $\alpha_{i,k}$ and $\mathcal{W}_{i,k}$ is a polytope. The nonlinearity is absorbed at sampling time, leaving the vertex set, and every edge cost built from it, convex. This convex set represents an inner, sampled, approximation of the dynamics in Q_k at step i . A weight vector $\alpha_{i,k}$ fixes a state x , control input u , propagation f , duration t , and cost r through the

bundle matrices, (4). To construct the iteration's graph, we replace each region $\mathcal{Q}_k$ at knot i by its projected bundle $\mathcal{W}_{i,k}$, keeping the edge set $\mathcal{E}$ of Section IV-A unchanged: $\hat{\mathcal{G}}_\sigma = (\mathcal{W}_{(i,k)}, \mathcal{E})$. Recall that $\mathcal{E}$ contains only forward, single-step edges between consecutive time layers, and only between regions that overlap ($j \cap k \neq \emptyset$); an edge is thus a candidate one-step transition from region j at knot i to a spatially adjacent region k at knot $i+1$.

V. THE GTB RELAXATION

We now describe how GTB formulates iterative shortest path problems in dynamics space via $\hat{\mathcal{G}}_\sigma$ and how, analogously to [1], it can be relaxed to a convex program. We then show that the iterative nature of GTB, which contracts the bundles around the vertex solutions, drives the relaxation towards tightness.

A. The dynamic-infeasibility edge cost

For an interior edge $e = ((i, j) \rightarrow (i+1, k)) \in \mathcal{E}_{\text{int}}$ we assign the convex *consensus* (dynamic-infeasibility) cost between the forward block of the source vertex and the state block of the destination vertex,

$$\ell_e(\alpha_{i,j}, \alpha_{i+1,k}) = \|\hat{W}_f^{(i,j)} \alpha_{i,j} - \hat{W}_x^{(i+1,k)} \alpha_{i+1,k}\|, \quad (9)$$

The boundary edges carry the matching endpoint costs

$$\begin{aligned} \ell_{s \rightarrow (0,k)} &= \|\hat{W}_x^{(0,k)} \alpha_{0,k} - x_s\|, \\ \ell_{(N-1,k) \rightarrow t} &= \|\hat{W}_x^{(N-1,k)} \alpha_{N-1,k} - x_t\|. \end{aligned} \quad (10)$$

A valid trace of the graph is then a sequence of regions from x_s to x_t whose accumulated cost (9)–(10) is small: the final state reaches the target and, at every layer, the one-step image of the chosen point in $\mathcal{W}_{i,j}$ coincides with the chosen point in the next vertex $\mathcal{W}_{i+1,k}$ (minimal dynamic infeasibility). This is a shortest-path problem in *dynamics space* as opposed to a shortest path in, e.g., Euclidean space. The “length” of a path is therefore not how far the system traveled (as in GCS) but how badly a sequence of interpolators fail to create a dynamically feasible trajectory.

B. A GTB shortest-path iteration

Collecting the convex vertex sets (8) and the dynamic-infeasibility edge costs (9)–(10) into a single shortest path problem over $\hat{\mathcal{G}}_\sigma$,

$$\min_{\alpha, \phi} \sum_{e \in \mathcal{E}} \phi_e \left[\ell_e(\alpha_{i,j}, \alpha_{i+1,k}) + \mu \|h_e(\alpha_{i,j})\| + \nu \|\alpha_{i,j}\| \right] \quad (11)$$

s.t. ϕ on the s – t path polytope of (3),

$$\phi_e \in \{0, 1\}, \quad \alpha_{i,k} \in \Delta^{M_s-1}, \quad \forall e, \forall i, \forall k$$

Rather than an abstract path variable $p \in \mathcal{P}$, the region sequence is now a *flow encoding*: a binary edge flow ϕ_e held on the s – t path polytope by the flow-conservation and degree constraints of (3). Here $h_e(\alpha_{i,j})$ is an additional running cost, e.g. $W_r^{(i,j)} \alpha_{i,j}$, and $\|\alpha_{i,j}\|$ is a regularization term as $M_s > n_x$; $(\mu, \nu) \geq 0, (\mu, \nu) \ll 1$ are small weights. A solution to (11) selects both the flow (a region sequence)

and the interpolators $\alpha_{i,k}$ along it, shown schematically in Fig. 2c. We solve its perspective-form relaxation, relaxing $\phi_e \in \{0, 1\}$ to $\phi_e \in [0, 1]$ and rewriting each bilinear $\phi_e \ell_e$ through the perspective transform [1].

a) *Norms*: Different norms can be used in (11). The ℓ_2 and ℓ_2^2 norms are originally presented in GCS [16], [1]. The ℓ_1 norm, used in TBM [2], is also a valid choice and promotes sparsity. Additionally its perspective relaxation is a (larger) LP rather than an SOCP. We note in Section VI that the ℓ_2 norm is most robust for iterative GTB solves, while the ℓ_1 norm is more sensitive to the initial bundle sampling and can lead to suboptimal paths.

C. Fractional Flow Solutions

The relaxed solution provides, for every vertex carrying flow, the points $\bar{x}_{i,k} = \hat{W}_x^{(i,k)} \alpha_{i,k}$ and $\bar{u}_{i,k} = \hat{W}_u^{(i,k)} \alpha_{i,k}$ (the new vertex solution of Section IV-B), shown in Fig. 2d.

Convex relaxations of GCS shortest-path problems are often *tight*: the optimal flow ϕ is integral and traces a single $s \rightarrow t$ path. The relaxation of (11) is not, particularly in early iterations, where bundles are wide, sets overlap, and no single region sequence dominates. This is not a failure mode that should be rounded away: the fractional flow is a convex combination of integral flows so (11) returns a *weighted* set of candidate paths with a vertex solution for each region along those paths. We exploit this in two ways:

a) *Resampling live vertices*: Any vertex solution $(\bar{x}_{i,k}, \bar{u}_{i,k})$ on an edge with $\phi_e > 0$ is a candidate for the next iterate, and we therefore resample around this solution to create the next bundle $\mathcal{W}_{i,k}$. This is the powerful aspect of GTB where, rather than committing to a single path, the fractional flow and GCS construction keeps track of a graph of possible paths. Fig. 3 shows how early iterations explore different paths while later iterations focus on refining the selected path.

b) *Marginals over regions*: The magnitudes of the fractional flow admit a probabilistic interpretation. For every edge flow $\phi_e \in [0, 1]$, satisfying flow conservation (3), the edge flow is a probability mass over transitions,

$$q_{i,k \rightarrow k'} = \frac{\phi_{i,k \rightarrow k'}}{\sum_{k''} \phi_{i,k \rightarrow k''}}. \quad (12)$$

The marginal probability that a path then visits region k at layer i is obtained from the incoming flow

$$p_{i,k} = \sum_{k': (i-1, k' \rightarrow k) \in \text{in}(k)} q_{i-1, k' \rightarrow k} \cdot p_{i-1, k'}, \quad (13)$$

$$p_{0,k} = \phi_{s \rightarrow k},$$

and by flow conservation, $\sum_k p_{i,k} = 1$ at every layer i . We accumulate these marginals over iterations in order to disregard vertices that are consistently unlikely, lowering the computational complexity of subsequent iterations.

D. Tightness, Guarantees, and Limitations

A critical point for a convex relaxation, which is solved as in Eq. (11) for every GTB iteration, is its tightness or integrality gap Δ , measuring the degree to which the optimal

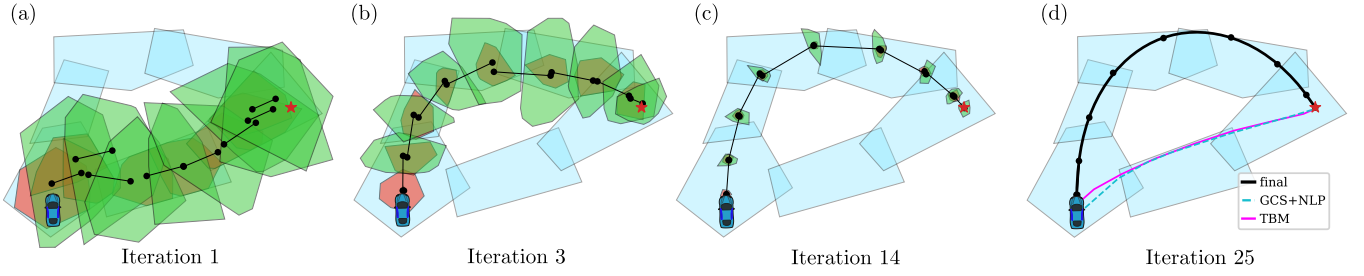


Fig. 3: Iterations of GTB for a path planning problem for a car model navigating around an obstacle, showing the projected bundles (shaded red) and forward bundles (shaded green) of the highest-probability path. Early iterations explore modes via the coarse approximation of the dynamics while later iterations focus on refining the trajectory (d) shows the final trajectory and a comparison to (dynamically infeasible) results of GCS initializing an NLP solver, which gets stuck on the shortest-path local minima and the Trajectory Bundle Method which gets stuck on the local minima of the initial guess.

value of the GCS relaxation, p_{CP} , matches the original (non-convex) problem of the binary GCS formulation, p_{IP} , for a specific GTB iteration. While we cannot prove convergence of GTB, we can show that the relaxation is tight in the limit: $\Delta(\sigma) = p_{IP}(\sigma) - p_{CP}(\sigma) = 0$ as $\sigma \rightarrow 0$.

Lemma 1 (Tightness of the GCS relaxation). *Assume $f(x, u)$ is continuous, $\mathcal{U}$ is compact and convex, and each region $\mathcal{Q}_k$ is a nonempty closed convex set, making the generators $\{\bar{g}_{i,k}\}$ well-defined. Then at $\sigma = 0$ the integrality gap vanishes, $\Delta(0) = 0$.*

Proof sketch: At $\sigma = 0$, the bundles are a single point, so R_σ degenerates to a min-cost $s \rightarrow t$ flow with fixed edge weights, and flow polytopes of layered DAGs are integral. ■

Proposition 1 (Optimality gap shrinking). *Adopt Lemma 1, with edge cost $\ell_e = \|\cdot\|_2$ as in (9)–(10), and assume f is locally Lipschitz and the sampling/projection scheme (6) is nonexpansive with scale σ . Then the integrality gap $\Delta(\sigma)$ is controlled by the bundles' diameter, which itself shrinks linearly with σ : $\Delta(\sigma) \rightarrow 0$ continuously as $\sigma \rightarrow 0$. Annealing the bundles down (Section III) therefore drives the relaxation smoothly toward the exact, tight limit of Lemma 1.*

Remark 1. *The converse of Proposition 1 explains the behavior and the power of GTB solving sequential sampled GCS subproblems. In early iterations, the interpolators show a convex combination of paths as a suitable candidate of the overall solution. Rather than treating this as a failure, GTB interprets it probabilistically, keeping track of promising solution candidates and updating vertex solutions accordingly.*

Remark 2 (Guarantees and limitations). *GTB inherits the guarantees of sequentially convex methods. Each iteration solves an inner, sampled approximation of (1), so the method reaches at best a local minimum; we cannot certify global optimality, tightness of the relaxation at a given $\sigma > 0$, or that the dynamic-infeasibility cost vanishes as $\sigma \rightarrow 0$. What GTB adds over standard SCP and TBM is a global perspective (the fractional flow keeps several region sequences alive rather than committing to one linearization point) and support for black-box f . GTB also removes sensitivity to the initial trajectory guess, but a weaker dependence on the convex decomposition $\{\mathcal{Q}_k\}$ remains.*

E. Solver Details

a) Graph cutting: The marginals (12)–(13) are used to accumulate evidence: every edge e carries a persistent probability P_e which is updated after each solve by a log-odds increment of that iteration's marginal. A region transition has to be *persistently* unused before its probability falls below a threshold δ and it is dropped from the next relaxation. Since the number of variables and cones in (11) scales with $|\mathcal{E}_{\text{int}}|$, this progressively shrinks the conic program as the run converges, without altering the formulation on the edges that remain.

b) Annealing the bundle scale: The bundle scale (and hence the trust region) is annealed via an *exploration* and *exploitation* phase. First, the bundles are wide enough to explore several region sequences in parallel, and the fractional flow is used to refine the vertex solutions of all live vertices. When the rounded path settles, the bundles are re-bloated to a trust region wide enough to allow the continuous refinement to still move the trajectory, and then contracted again to polish the solution.

VI. RESULTS

We demonstrate GTB on motion planning problems for systems with nonlinear or black-box dynamics, satisfying complex collision- and specification constraints. For all scenarios we sample with a Gaussian distribution and report average results over 20 random seeds.

We compare GTB against the Trajectory Bundle Method (TBM) and using GCS as an initial guess for a nonlinear trajectory problem (GCS+NLP). In the TBM benchmark, we use the same σ -annealing and sampling scheme and run the same number of iterations. Collision avoidance is enforced via a penalty term in the cost function which is satisfied if any bundle is in at least one convex region. For GCS+NLP, the GCS solution is used as an initial guess for a nonlinear solver which enforces collision avoidance for the set that GCS allocated to the segment. The car and drone models are analytic so that GCS+NLP can be compared against; for the MuJoCo cartpole, which GTB and TBM treat as black-box, GCS+NLP is handed an analytic model.

As both GTB and TBM are iterative sampling-based methods, neither are guaranteed to find a feasible solution and there are always seeds and scenarios where optimal

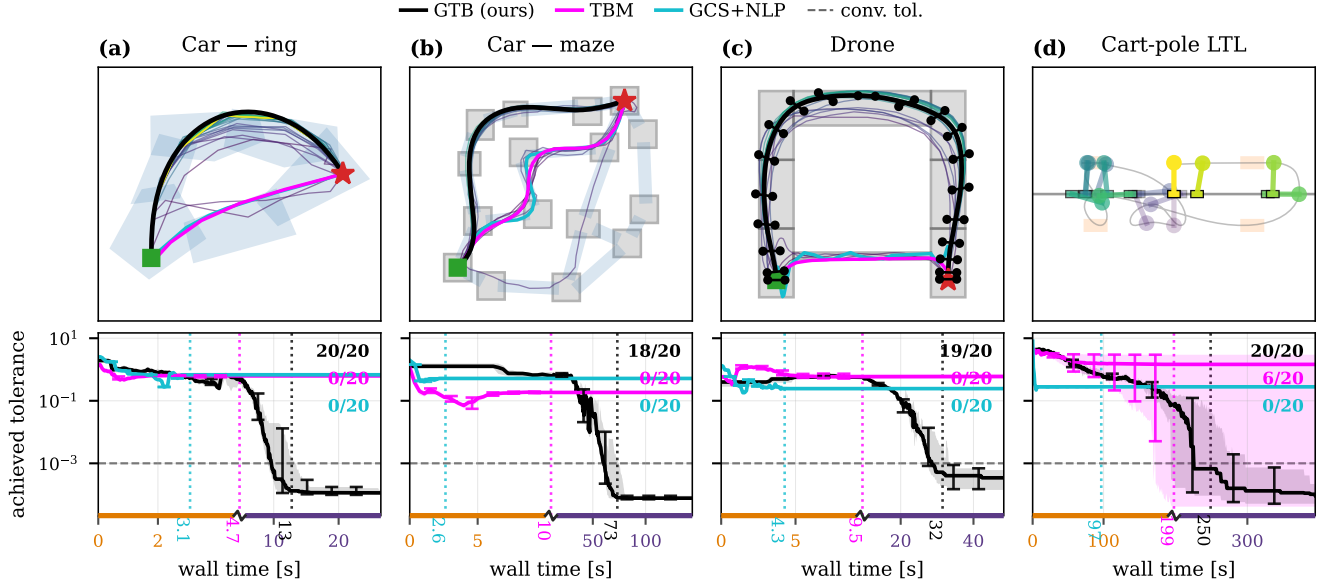


Fig. 4: Comparisons of GTB against the Trajectory Bundle Method and the GCS solve with an NLP refinement. On top the physical trajectories and early iterations of the GTB algorithm highlighting the different paths that are considered. On the bottom the tolerance as a function of the wall clock time. Dotted vertical lines denote the average computation time. The orange and purple x-axis denote the different scales to highlight that GTB converges approximately equally rapid to infeasible solutions, but converges to lower tolerances afterwards. While GTB almost always finds feasible solutions, both TBM and GCS+NLP often fail and are stuck in local minima.

solutions can be found. The results highlight that GTB is able to, much more reliably, find feasible solutions on a variety of graph-based motion planning problems, while TBM and GCS+NLP often get stuck in local minima.

A. Car through a maze

We consider the bicycle model with Ackermann steering, which is a common model for car-like vehicles where heading change is coupled to the forward velocity with $x = [x, y, v, \theta, \delta]^T \in \mathcal{W}_{free}$ and $u = [u_a, u_\delta]^T \in \mathcal{U}$. The dynamics are given by

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v \cos(\theta) \\ v \sin(\theta) \\ u_a \\ \frac{v}{L} \tan(\delta) \\ u_\delta \end{bmatrix}$$

The bicycle model is differentially flat in the position (x, y) , so a GCS planner can trace a geometric path through the maze. That path, however, ignores input bounds $\mathcal{U}$ on (u_a, u_δ) . The GTB method in the corridor scenario is shown in Fig. 3 and Fig. 4a showing the evolution of the trajectory bundle over iterations and the tolerance over wall-clock time. Fig. 4b shows the car through a more complicated maze, substantiating the capability of GTB yet highlighting the additional computation time required to iteratively solve the GCS subproblems. GTB finds feasible solutions, satisfying the dynamics, input, and the collision constraints in 20 and 18 out of 20 seeds respectively. For both scenarios, TBM and GCS+NLP fail to find dynamically feasible solutions. The initial guess (straight-line interpolation) or the shortest path formulation respectively bring the system to an infeasible local minimum.

B. Planar drone through a thin gap

We consider a planar drone with $x = [x, y, \dot{x}, \dot{y}, \theta, \omega]^T \in \mathcal{W}_{free}$ and $u = [u_1, u_2]^T \in \mathcal{U}$ being the thrusts of the two propellers. The dynamics are given by

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ -\frac{1}{m}(u_1 + u_2) \sin(\theta) \\ -g + \frac{1}{m}(u_1 + u_2) \cos(\theta) \\ \omega \\ -\frac{l}{I}(u_1 - u_2) \end{bmatrix}$$

The drone needs a non-zero orientation, θ , to travel horizontally. The considered scenario has two gaps: a narrow one where collisions can only be avoided if the orientation is severely limited, and a thick one where the drone can orient itself more freely. GTB reasons over all possible sequences of regions to find one which permits a dynamically feasible trajectory, finding a path through a sequence of convex sets that permits the required orientation in 19 out of 20 seeds. The results are shown in Fig. 4c where GTB finds a feasible solution, satisfying the dynamics and the collision constraints, while both TBM and GCS+NLP fail to find feasible solutions and are again stuck in local (shortest-path) minima.

C. Simulated Cartpole with Temporal Logic

While GTB permits state-space decomposition to solve maze-like scenarios, we can also use the DAG structure to encode temporal logic specifications, as in [18]. As the GTB method permits explicit constraints via set-containment, it goes beyond single-shooting methods such as MPPI, encoding temporal logic specifications by construction. This makes

it possible for unstable nonlinear systems to satisfy temporal logic specifications, which is not possible with TBM or GCS+NLP. We consider a cartpole defined in Mujoco [19] that, from a downward position, has to satisfy the Linear Temporal Logic (LTL) specification ϕ , encoding that we either visit *top left* or *top right* after which we visit both *down left* and *down right*;

$$\phi = F((\text{top left} \vee \text{top right}) \wedge F(\text{down left} \wedge \text{down right}))$$

and finally end upright in the middle. Fig. 4d shows the GTB method finds a feasible solution, satisfying the dynamics and the temporal logic specification explicitly via the product graph of this unstable black-box system. In contrast, TBM and GCS+NLP fail to consistently find feasible solutions. While the GCS+NLP method finds a satisfying solution for the specification, but the shortest path formulation limits segments to infeasible transitions when swing-up needs to be considered. The average computation time is 250 seconds.

D. Autonomous Underwater Vehicle

We finally show how GTB can also be applied to more complex and learned dynamics models. The SAM Autonomous Underwater Vehicle (AUV) [20] is a torpedo-like robot often modelled via Fossen's model [21]. A challenge for this model is an accurate representation of the damping matrix, which models complex hydrodynamic effects. We learn this damping matrix via a physics-informed neural network (PINN), inspired by [22], modelling the damping matrix D as $L^T L$ where L is a 2 layer, 64 neuron model. The state of the AUV is the position, orientation, and linear/angular velocities, plus two internal states for the pitch and depth actuation; $N_x = 15$. The control inputs are the propellers, propeller pitch and yaw, as well as pitch and depth actuation; $N_u = 6$. Details can be found in [20]. Fig. 1 shows a collision-free trajectory through a maze-like environment while satisfying the dynamics and actuation constraints, even on a complex, high-dimensional system as SAM. The total computation time is 544 seconds with an overall success rate of 19/20, dominated by several early full-maze solves to discover the feasible sequence of regions.

VII. CONCLUSION AND FUTURE WORK

We presented the Graph of Trajectory Bundles (GTB) method for graph-based motion planning for nonlinear or black-box systems. GTB relies on constructing a sampled inner approximation of a workspace GCS that considers the dynamics via convex hulls of sampled trajectories. Minimizing the dynamics residual can then be formulated as a shortest path problem over the graph which is solved iteratively, converging to feasible trajectories in the limit.

Future work is on improving the solution times of GTB. The sequential/iterative nature of the algorithm, where large but sparse graph-structured second-order cone programs need to be solved, could lend itself to custom solvers that utilize e.g. distributed consensus, warm-starting, or tailored factorization methods. Tailored solvers can then open the door to the line-graph construction of $\hat{\mathcal{G}}$, richer specifications, more complex environments, and hardware implementation.

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, pp. 507–532, 2024.
- [2] K. Tracy, J. Z. Zhang, J. Arrizabalaga, S. Schaal, Y. Tassa, T. Erez, and Z. Manchester, "The trajectory bundle method: Unifying sequential-convex programming and sampling-based trajectory optimization," *arXiv preprint arXiv:2509.26575*, 2025.
- [3] L. E. Kavraki, *Random networks in configuration space for fast path planning*. stanford university, 1995.
- [4] D. von Wrangel and R. Tedrake, "Using graphs of convex sets to guide nonconvex trajectory optimization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 9863–9870.
- [5] C. L. Clark and B. Xie, "Plan optimal collision-free trajectories with non-convex cost functions using graphs of convex sets," *IEEE Transactions on Robotics*, 2025.
- [6] B. P. Graesdal, S. Y. C. Chia, T. Marcucci, S. Morozov, A. Amice, P. A. Parrilo, and R. Tedrake, "Towards tight convex relaxations for contact-rich manipulation," *arXiv preprint arXiv:2402.10312*, 2024.
- [7] 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.
- [8] I. Mir, F. Gul, S. Mir, M. A. Khan, N. Saeed, L. Abualigah, B. Abuhaija, and A. H. Gandomi, "A survey of trajectory planning techniques for autonomous systems," *Electronics*, vol. 11, no. 18, p. 2801, 2022.
- [9] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.
- [10] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [11] M. J. Powell, "A view of algorithms for optimization without derivatives," *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, vol. 43, no. 5, pp. 170–174, 2007.
- [12] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019.
- [13] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [14] H. Homburger, K. Baumgärtner, M. Diehl, and J. Reuter, "Gauss-newton accelerated mppi control," *arXiv preprint arXiv:2512.04579*, 2025.
- [15] Y. Zhang, C. Pezzato, E. Trevisan, C. Salmi, C. H. Corbato, and J. Alonso-Mora, "Multi-modal mppi and active inference for reactive task and motion planning," *IEEE Robotics and Automation Letters*, vol. 9, no. 9, pp. 7461–7468, 2024.
- [16] 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.
- [17] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67.
- [18] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3791–3804, 2023.
- [19] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.
- [20] S. Bhat, I. Torroba, Ö. Özkahraman, N. Bore, C. I. Sprague, Y. Xie, I. Stenius, J. Severholt, C. Ljung, J. Folkesson *et al.*, "A cyber-physical system for hydrobat aivs: System integration and field demonstration," in *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. IEEE, 2020, pp. 1–8.
- [21] T. I. Fossen and O.-E. Fjellstad, "Nonlinear modelling of marine vehicles in 6 degrees of freedom," *Mathematical Modelling of Systems*, vol. 1, no. 1, pp. 17–27, 1995.
- [22] L. Backne-Genborg, "Physics-informed machine learning for system identification of an autonomous underwater vehicle," 2025.