Accelerating Motion Planning via Optimal Transport

An T. Le[1], Georgia Chalvatzaki[1,3], Armin Biess[5] and Jan Peters[1,2,3,4]

Abstract— **Motion planning is still an open problem in robotics. A class of methods striving to provide smooth solutions is gradient-based trajectory optimization. However, those methods might suffer from bad local minima, while for many settings, they may be inapplicable due to the absence of access to objectives-gradients. In response to these issues, we introduce Motion Planning via Optimal Transport (MPOT) - a *gradient-free* method that optimizes a batch of smooth trajectories over highly nonlinear costs, even for high-dimensional tasks, while imposing smoothness through a Gaussian Process trajectory prior that serves as cost. To facilitate batch trajectory optimization, we introduce an original zero-order and highly-parallelizable update rule – the Sinkhorn Step, which uses the regular polytope family for its search directions; each regular polytope, centered on trajectory waypoints, serves as a local neighborhood, effectively acting as a trust region, where the Sinkhorn Step "transports" local waypoints toward low-cost regions. With these properties, MPOT solves batch planning tasks even with narrow passages in *less than a second*, finding locally optimal solutions. We show the efficiency of MPOT in a range of problems from low-dimensional point-mass navigation to high-dimensional whole-body robot motion planning, evincing its superiority compared with popular motion planners and paving the way for new applications of optimal transport in motion planning.**

I. Introduction

Motion planning is a fundamental problem in robotics [1], aiming to find feasible, smooth, and collision-free paths from start-to-goal configurations. Motion planning has been studied both as sampling-based search [2], [3], [4] and as an optimization problem [5], [6], [7], [8]. Nevertheless, as with every optimization pipeline, trajectory optimization depends on initialization and can get trapped in local minima due to the non-convexity of complex objectives. Moreover, in some problem settings, objective gradients are unavailable or expensive to compute. Indeed, trajectory optimization is difficult to tune and is often avoided in favor of sampling-based methods with probabilistic completeness.

To address these issues of trajectory optimization, we propose a zero-order, fast, and highly parallelizable update rule – the Sinkhorn Step. We apply this novel update rule in trajectory optimization, resulting in *MPOT* – a gradient-free trajectory optimization method optimizing a batch of smooth trajectories. MPOT optimizes trajectories by solving a sequence of entropic-regularized Optimal Transport (OT), where each OT instance is solved efficiently with the celebrated Sinkhorn-Knopp algorithm [9]. In particular, MPOT discretizes the trajectories into waypoints and structurally probes a local neighborhood around each of them, which effectively exhibits a *trust region*, where it "transports" local waypoints towards low-cost areas given the local cost approximated by the probing mechanism. Our method is simple, and neither requires gradients of cost functions nor sampling from proposal distributions. Crucially, the planning-as-inference perspective [10], [8] allows us to impose constraints related to transition dynamics as planning costs, additionally imposing smoothness through a GP prior. Delegating complex constraints to the planning objective allows us to locally resolve trajectory update as an OT problem at each iteration, updating the trajectory waypoints towards the local optima, thus effectively optimizing for complex cost functions formulated in joint and task space.

Our **contribution** is twofold. *(i)* We propose the Sinkhorn Step - an efficient zero-order update rule for optimizing a batch of parameters, formulated as an entropic-regularized OT problem. *(ii)* We, then, apply the Sinkhorn Step to motion planning, resulting in a novel trajectory optimization method that optimizes a batch of trajectories by efficiently solving a sequence of linear programs. It treats every waypoint across trajectories equally, enabling fast batch updates of multiple trajectories-waypoints over multiple goals within a single OT instance while retaining smoothness due to integrating the GP prior as cost function.

II. Preliminaries

Entropic-regularized optimal transport. We briefly introduce discrete OT. For a thorough introduction, we refer to [11], [12], [13].

Notation. Throughout the paper, we consider the optimization on a d-dimensional Euclidean space $\mathbb{R}^d$, representing the parameter space (e.g., a system state space). $\mathbf{1}_d$ is the vector of ones in $\mathbb{R}^d$. The scalar product for vectors and matrices is $x, y \in \mathbb{R}^d$, $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$; and $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{d \times d}$, $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{i,j=1}^{d} \boldsymbol{A}_{ij} \boldsymbol{B}_{ij}$, respectively. $\|\cdot\|$ is the l_2-norm, and $\|\cdot\|_{\boldsymbol{M}}$ denotes the Mahalanobis norm w.r.t. some positive definite matrix $\boldsymbol{M} \succ 0$. For two histograms $\boldsymbol{n} \in \Sigma_n$ and $\boldsymbol{m} \in \Sigma_m$ in the simplex $\Sigma_d := \{\boldsymbol{x} \in \mathbb{R}_+^d : \boldsymbol{x}^\intercal \mathbf{1}_d = 1\}$, we define the set of $n \times m$ matrices $U(\boldsymbol{n}, \boldsymbol{m}) := \{\boldsymbol{W} \in \mathbb{R}_+^{n \times m} \mid \boldsymbol{W}\mathbf{1}_m = \boldsymbol{n}, \boldsymbol{W}^\intercal \mathbf{1}_n = \boldsymbol{m}\}$ containing doubly stochastic $n \times m$ matrices with row and column sums $\boldsymbol{n}$ and $\boldsymbol{m}$ respectively. Correspondingly, the entropy for $\boldsymbol{A} \in U(\boldsymbol{n}, \boldsymbol{m})$ is defined as $H(\boldsymbol{A}) = -\sum_{i,j=1}^{n,m} a_{ij} \log a_{ij}$.

Let $\boldsymbol{C} \in \mathbb{R}_+^{n \times m}$ be the positive cost matrix, the OT between $\boldsymbol{n}$ and $\boldsymbol{m}$ given cost $\boldsymbol{C}$ is $\mathrm{OT}(\boldsymbol{n}, \boldsymbol{m}) := \min_{\boldsymbol{W} \in U(\boldsymbol{n}, \boldsymbol{m})} \langle \boldsymbol{W}, \boldsymbol{C} \rangle$. Traditionally, OT does not scale well with high dimensions. To address this, [14] proposes to regularize its objective with an entropy

This work was funded by the German Research Foundation project METRIC4IMITATION (PE 2315/11-1). [1]Department of Computer Science, Technische Universitat Darmstadt, Germany; [2]German Research Center for AI (DFKI); [3]Hessian.AI; [4]Centre for Cognitive Science; [5]armin.biess@gmail.com. Corresponding author: An T. Le, an@robot-learning.de

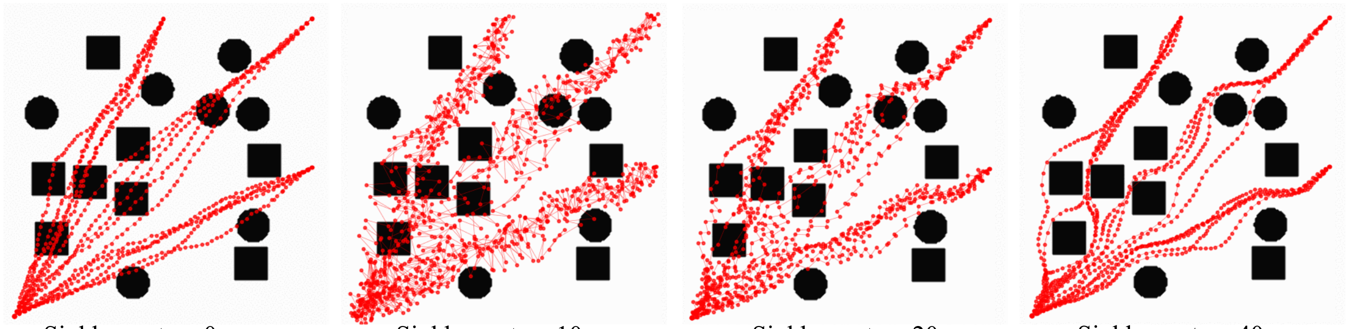

Fig. 1: Example of Motion Planning via Optimal Transport (MPOT) in planar navigation scenario with three goals. For each goal, we sample five initial trajectories from a Gaussian Process (GP) prior. We illustrate four snapshots of our proposed Sinkhorn Step that updates a batch of waypoints from multiple trajectories over multiple goals. For this example, the **total planning time was 0.12s**. More demos can be found on `https://sites.google.com/view/sinkhorn-step/`

term, resulting in the entropic-regularized OT

$$\mathrm{OT}_\lambda(\boldsymbol{n}, \boldsymbol{m}) := \min_{\boldsymbol{W} \in U(\boldsymbol{n}, \boldsymbol{m})} \langle \boldsymbol{W}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{W}). \quad (1)$$

Solving (1) with Sinkhorn-Knopp [14] has a complexity of $\tilde{O}(n^2/\epsilon^3)$ [15], where ϵ is the approximation error w.r.t. the original OT. Higher λ enables a faster but "blurry" solution, and vice versa.

Trajectory optimization. Given a parameterized trajectory by a discrete set of support states and control inputs $\boldsymbol{\tau} = [\mathrm{x}_0, \boldsymbol{u}_0, ..., \mathrm{x}_{T-1}, \boldsymbol{u}_{T-1}, \mathrm{x}_T]^\intercal$, trajectory optimization aims to find the optimal trajectory $\boldsymbol{\tau}^*$, which minimizes an objective function $c(\boldsymbol{\tau})$, with x_0 being the start state. Standard motion planning costs, such as goal cost c_g defined as the distance to a desired goal-state $\boldsymbol{x}_g$, obstacle avoidance cost c_{obs}, and smoothness cost c_{sm} can be included in the objective. Hence, trajectory optimization can be expressed as the sum of those costs while obeying the dynamics constraint

$$\begin{aligned} \boldsymbol{\tau}^* &= \arg\min_{\boldsymbol{\tau}} \left[c_{obs}(\boldsymbol{\tau}) + c_g(\boldsymbol{\tau}, \mathrm{x}_g) + c_{sm}(\boldsymbol{\tau}) \right] \quad (2) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\mathrm{x}, \boldsymbol{u}) \text{ and } \boldsymbol{\tau}(0) = \boldsymbol{x}_0. \end{aligned}$$

For many manipulation tasks with high-degrees of freedom (DoF) robots, this optimization problem is typically highly nonlinear due to many complex objectives and constraints.

III. Sinkhorn Step

To address the problem of batch-optimizing multiple trajectories in a gradient-free setting, we propose *Sinkhorn Step* - a parallelizable zero-order update rule for a batch of optimization variables. Our method draws inspiration from the free-support barycenter problem [16], where the mean support of a set of empirical measures is optimized w.r.t. the OT cost.

We introduce the *Sinkhorn Step*, consisting of two components: a polytope structure defining the unbiased search-direction bases, and a weighting distribution for evaluating the search directions. Particularly, the weighting distribution has row-column unit constraints and must be efficient to compute. Following the motivation of [16], the entropic-regularized OT fits nicely into the second component, providing a solution for the weighting distribution as an OT plan, which is solved extremely fast, and its solution is unique [14].

Sinkhorn Step relates to *directional-direct search* methods [17], [18], that typically evaluate the objective function over a (typically fixed) search-direction-set D ensuring descent with a sufficiently small stepsize. The search-direction-set is typically a vector-set requiring to be a *positive spanning set* [19], i.e., its conic hull is $\mathbb{R}^d = \{\sum_i w_i \boldsymbol{d}_i,\, \boldsymbol{d}_i \in D,\, w_i \geq 0\}$, ensuring that every point (including the extrema) in $\mathbb{R}^d$ is reachable by a sequence of positive steps from any initial point.

Regular Polytope Search-Directions. Consider a $(d-1)$-unit hypersphere $S^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : \ \|\boldsymbol{x}\| = 1\}$ with the center at zero. Let us denote the regular polytope family $\mathcal{P} = \{\text{simplex, orthoplex, hypercube}\}$. Consider a d-dimensional polytope $P \in \mathcal{P}$ with m vertices, the search-direction set D^P is constructed from the vertex set of the regular polytope P inscribing S^{d-1}

$$D^P = \{\boldsymbol{d}_i \mid \|\boldsymbol{d}_i\| = 1\}_{i=1}^m.$$

The d-dimensional regular polytope family $\mathcal{P}$ has all of its dihedral angles equal and, hence, is an unbiased sparse approximation of the circumscribed $(d-1)$-sphere, i.e., $\sum_i \boldsymbol{d}_i = 0, \|\boldsymbol{d}_i\| = 1 \, \forall i$. There also exist other regular polytope families. However, the regular polytope types in $\mathcal{P}$ exist in every dimension (cf. [20]).

Batch Update Rule. At an iteration k, given the current optimizing points X_k and their matrix form $\boldsymbol{X}_k \in \mathbb{R}^{n \times d}$, we first construct the direction set from a chosen polytope P, and denote the direction set $\boldsymbol{D}^P \in \mathbb{R}^{m \times d}$ in matrix form. Following [16], let us define the prior histograms reflecting the importance of optimizing points $\boldsymbol{n} \in \Sigma_n$ and the search directions $\boldsymbol{m} \in \Sigma_m$, then the constraint space $U(\boldsymbol{n}, \boldsymbol{m})$ of OT is defined. With these settings, we define Sinkhorn Step. The batch update rule is the barycentric projection [12] that optimizes the free-support barycenter of the optimizing points and the batch polytope vertices

$$\begin{aligned} \boldsymbol{X}_{k+1} &= \boldsymbol{X}_k + \boldsymbol{S}_k,\ \boldsymbol{S}_k = \alpha_k \mathrm{diag}(\boldsymbol{n})^{-1} \boldsymbol{W}_\lambda^* \boldsymbol{D}^P \\ \text{s.t. } \boldsymbol{W}_\lambda^* &= \operatorname*{argmin}_{\boldsymbol{W} \in U(\boldsymbol{n}, \boldsymbol{m})} \langle \boldsymbol{W}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{W}) \end{aligned} \quad (3)$$

with $\alpha_k > 0$ as the stepsize, $\boldsymbol{C} \in \mathbb{R}^{n \times m}$, $\boldsymbol{C}_{i,j} = f(\boldsymbol{x}_i + \alpha_k \boldsymbol{d}_j)$, $\boldsymbol{x}_i \in X_k, \boldsymbol{d}_j \in D^P$ is the local objective matrix evaluated at the linear-translated polytope vertices. Observe that the matrix $\mathrm{diag}(\boldsymbol{n})^{-1} \boldsymbol{W}_\lambda^*$ has n row vectors in the simplex Σ_m. The suggested update *transports* $\boldsymbol{X}$ as a barycenter shaping by the polytopes with weights defined by the optimal solution $\boldsymbol{W}_\lambda^*$.

IV. Motion Planning via Optimal Transport

Here, we introduce MPOT - a method that applies *Sinkhorn Step* to solve the batch trajectory optimization problem, where we realize each waypoint in a set of trajectories as an optimizing point. Due to Sinkhorn Step's properties, MPOT does not require gradients propagated from cost functions over long kinematics chains. It optimizes trajectories by solving a sequence of strictly convex linear programs with a maximum entropy objective (cf. Eq. (3)), effectively transporting the waypoints according to the local polytope structure. To promote smoothness and dynamically feasible trajectories, we incorporate the GP prior as a cost by formulating the motion planning problem with a KL divergence between a proposal and a target posterior distribution.[1]

A. Trajectory optimization objective

We motivate the trajectory optimization problem from the planning-as-inference perspective [10], [8]. This perspective allows us to naturally incorporate the system dynamics as a GP prior to the planning objective. Assuming a first-order trajectory optimization, the control sequence can be defined as a time-derivative of the states $U = [\dot{\mathbf{x}}_0, ..., \dot{\mathbf{x}}_T]$. With the first-order optimizing trajectory $\boldsymbol{\tau} = (X, U) = \{\boldsymbol{x}_t \in \mathbb{R}^d : \boldsymbol{x}_t = [\mathbf{x}_t, \dot{\mathbf{x}}_t]\}_{t=1}^T$, through a series of mathematical derivations, the trajectory optimization can be formulated as

$$\boldsymbol{\tau}^* = \operatorname*{argmin}_{\boldsymbol{\tau}} \sum_{t=0}^{T-1} \underbrace{\eta C(\boldsymbol{x}_t)}_{\text{state cost}} + \underbrace{\frac{1}{2} \|\boldsymbol{\Phi}_{t,t+1}\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|^2_{\boldsymbol{Q}^{-1}_{t,t+1}}}_{\text{transition model cost}}, \tag{4}$$

with $\boldsymbol{\Phi}_{t,t+1}$ the state transition matrix, and $\boldsymbol{Q}_{t,t+1}$ the covariance between time steps t and $t+1$ originated from the GP prior, which are specific to a considered system. This objective is a batch optimization problem by realizing the trajectory as a batch of optimizing points $\boldsymbol{\tau} \in \mathbb{R}^{T \times d}$. This realization also extends naturally to a batch of trajectories described in the next section.

B. Practical considerations for applying Sinkhorn Step

For the practical implementation, we make the following modifications to the Sinkhorn Step formulation for optimizing a trajectory $\boldsymbol{\tau}$.

First, we define a set of probe points (including the polytope vertices) for denser function evaluations. We populate equidistantly *probe* points along the directions in D^P outwards till reaching a *probe radius* $\beta_k \geq \alpha_k$, resulting in the *probe set* H^P with its matrix form $\boldsymbol{H}^P \in \mathbb{R}^{m \times h \times d}$ with h probe points for each direction. Second, due to the deterministic behavior of Sinkhorn Step, sometimes the optimization process empirically converges to undesirable local minima. We add stochasticity in the search directions by applying a random d-dimensional rotation $\boldsymbol{R} \in SO(d)$ to the polytopes to promote local exploration. Third, to further decouple the correlations between the waypoints updates, we sample the rotation matrices in batch and then construct the direction sets from the rotated polytopes, resulting in the tensor $\boldsymbol{D}^P \in \mathbb{R}^{T \times m \times d}$. Similarly, the *probe set* is also constructed in batch for every waypoint $\boldsymbol{H}^P \in \mathbb{R}^{T \times m \times h \times d}$. The Sinkhorn Step is computed with the *einsum* operation along the second dimension of $\boldsymbol{D}^P$.

[1]Full derivations can be found at https://www.ias.informatik.tu-darmstadt.de/uploads/Team/AnThaiLe/mpot_preprint.pdf

Algorithm 1: Motion Planning via Optimal Transport

```
$\mathcal{T}^0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{K}_0)$ and $\boldsymbol{n} = \mathbf{1}_N/N$, $\boldsymbol{m} = \mathbf{1}_m/m$
while termination criteria not met do
    // Epsilon Annealing for Sinkhorn Step
    (Optional) $\alpha \leftarrow (1-\epsilon)\alpha$, $\beta \leftarrow (1-\epsilon)\beta$.
    Construct randomly rotated $D^P, H^P$.
    Compute the cost matrix $\boldsymbol{C}$ as in Eq. (5).
    Perform Sinkhorn Step $\mathcal{T} \leftarrow \mathcal{T} + \mathbf{S}$.
end
```

With these considerations, the element of the t^{th}-waypoint and i^{th}-search directions in the OT cost matrix $\boldsymbol{C} \in \mathbb{R}^{T \times m}$ is the mean of probe point evaluation along a search direction (i.e., cost-to-go)

$$\begin{aligned} \boldsymbol{C}_{t,i} =& \frac{1}{h} \sum_{j=1}^{h} \eta C(\boldsymbol{x}_t + \boldsymbol{y}_{t,i,j}) \\ &+ \frac{1}{2} \|\boldsymbol{\Phi}_{t,t+1}\boldsymbol{x}_t - (\boldsymbol{x}_{t+1} + \boldsymbol{y}_{t+1,i,j})\|^2_{\boldsymbol{Q}^{-1}_{t,t+1}} \end{aligned} \tag{5}$$

with the probe point $\boldsymbol{y}_{t,i,j} \in H^P$. Then, we ensure the cost matrix positiveness for numerical stability by subtracting its minimum value. With uniform prior histograms $\boldsymbol{n} = \mathbf{1}_T/T$, $\boldsymbol{m} = \mathbf{1}_m/m$, the problem $\boldsymbol{W}^* = \operatorname{argmin} \mathrm{OT}_\lambda(\boldsymbol{n}, \boldsymbol{m})$ is instantiated and solved with the log-domain stabilization version [21], [22] of the Sinkhorn algorithm.

Most trajectory optimization methods arrive at approximately locally optimal solutions. Hence, the need for multiple parallelized trajectory optimizations is evident. We leverage our Sinkhorn Step to optimize multiple trajectories in parallel, efficiently providing many feasible solutions for multi-modal planning problems. Specifically, we implement MPOT using PyTorch [23] for vectorization across different motion plans, randomly rotated polytope constructions, and *probe set* cost evaluations. For a problem instance, we consider N_p trajectories of horizon T, and thus, the trajectory set $\mathcal{T} = \{\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_{N_p}\}$ is the parameter to be optimized. We can flatten the trajectories into the set of $N = N_p \times T$ waypoints. Now, the tensors of search directions and *probe set* $\boldsymbol{D}^P \in \mathbb{R}^{N \times m \times d}$, $\boldsymbol{H}^P \in \mathbb{R}^{N \times m \times h \times d}$ can be efficiently constructed and evaluated by the state cost function $C(\cdot)$, provided that the cost function is implemented with batch-wise processing (e.g., neural network models in PyTorch). Similarly, the model cost term in Eq. (4) can also be evaluated in batch by vectorizing the computation of the second term in Eq. (5).

To initialize the trajectories, we randomly sample from the discretized GP prior $\mathcal{T}^0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{K}_0)$, where $\boldsymbol{\mu}_0$ is a constant-velocity, straight-line trajectory from start-to-goal state, and $\boldsymbol{K}_0 \in \mathbb{R}^{(T \times d) \times (T \times d)}$ is a large GP covariance matrix for exploratory initialization [24], [25]. For execution, we select the lowest cost trajectory $\boldsymbol{\tau}^* \in \mathcal{T}^*$. For collecting a trajectory dataset, all trajectories $\mathcal{T}^*$ are stored along with contextual data, such as the occupancy map, goal state, etc.

TABLE I: Trajectory generation benchmarks in densely cluttered environments. RRT* and I-RRT* success and collision-free rates depict the maximum achievable values for all planners. For the *point-mass* environment, we populate 15 square and circle obstacles randomly, with each obstacle having a radius or width of 2 (cf. Fig. 1). We generate 100 environment-seeds, and for each environment-seed, we randomly sample 10 collision-free pairs of start and goal states, resulting in 1000 planning tasks. We plan each task in parallel 100 trajectories of horizon 64. For the *Panda* environment, we also generate 100 environment-seeds. Each environment-seed contains randomly sampled 15 obstacle-spheres having a radius of 10cm. Then, we sample 5 random collision-free target configurations (including self-collision free), resulting in 500 planning tasks, and plan in parallel 10 trajectories containing 64 timesteps. The metrics are T$[s]$ - *planning time* until convergence; SUC[%] - *success rate* over tasks in an environment-seed, where success means there is at least one successful trajectory found each task; GOOD[%] - success percentage of total parallelized plans in each environment-seed, reflecting the *parallelization quality*; S - *smoothness* measured by the norm of the finite difference of trajectory velocities, averaged over all trajectories and horizons; PL - *path length*.

	Point-mass Experiment					Panda Experiment				
	T$[s]$	SUC[%]	GOOD[%]	S	PL	T$[s]$	SUC[%]	GOOD[%]	S	PL
RRT*	43.2 ± 15.2	$100 \pm 0.$	$100 \pm 0.$	0.43 ± 0.12	23.8 ± 4.6	186.9 ± 184.2	$100 \pm 0.$	73.8 ± 26.7	0.17 ± 0.05	7.8 ± 2.9
I-RRT*	43.6 ± 13.8	$100 \pm 0.$	$100 \pm 0.$	0.43 ± 0.11	23.9 ± 4.8	184.2 ± 166.0	$100 \pm 0.$	74.6 ± 29.0	0.17 ± 0.05	7.6 ± 3.2
STOMP	2.2 ± 0.1	31.4 ± 13.9	10.5 ± 25.7	0.01 ± 0.01	17.0 ± 1.4	4.3 ± 0.1	50.8 ± 28.3	35.3 ± 42.0	0.01 ± 0.0	4.5 ± 0.8
SGPMP	6.5 ± 0.9	98.6 ± 4.5	74.9 ± 28.9	0.03 ± 0.01	18.3 ± 2.0	5.0 ± 0.2	67.8 ± 23.5	58.1 ± 45.8	0.01 ± 0.0	4.5 ± 0.9
CHOMP	0.5 ± 0.1	70.9 ± 16.7	38.6 ± 40.7	0.03 ± 0.0	17.7 ± 1.7	3.1 ± 0.3	63.0 ± 25.5	51.6 ± 46.2	0.02 ± 0.0	4.6 ± 0.8
GPMP2	2.8 ± 0.1	98.3 ± 4.9	74.9 ± 32.1	0.07 ± 0.05	20.3 ± 3.1	3.3 ± 0.2	66.0 ± 25.2	53.2 ± 42.3	0.01 ± 0.0	4.9 ± 0.8
MPOT	$\mathbf{0.4} \pm 0.0$	$\mathbf{99.2} \pm 3.1$	73.6 ± 26.7	0.06 ± 0.03	19.3 ± 2.3	$\mathbf{0.8} \pm 0.1$	$\mathbf{71.6} \pm 23.2$	60.2 ± 44.4	0.01 ± 0.01	4.6 ± 0.9

V. Experiments

First, we benchmark our method against strong motion planning baselines in a densely cluttered 2D-point-mass and a 7-DoF robot arm (Panda) environment. Then, we demonstrate the efficacy of our method on high-dimensional mobile manipulation tasks with TIAGo++.

Experimental setup. In all experiments, all planners optimize first-order trajectories with positions and velocities in configuration space. The motion planning costs are the $SE(3)$ goal, obstacle, self-collision, and joint-limit costs. The state dimension (configuration position and velocity) is $d = 4$ for the point-mass experiment, $d = 14$ for the Panda experiment, and $d = 36$ (3 dimensions for the base, 1 for the torso, and 14 for the two arms) for the mobile manipulation experiment. As for polytope settings, we choose a 4-cube for the point-mass case, a 14-othorplex for Panda, and a 36-othorplex for TIAGo++.

Baselines. We compare MPOT to popular trajectory planners, which are also straightforward to implement and vectorize in PyTorch for a fair comparison (even if the vectorization is not mentioned in their original papers). The chosen baselines are gradient-based planners: CHOMP [5] and GPMP2 (no interpolation) [6]; sampling-based planners: RRT* [3], [4] and its informed version I-RRT* [26], STOMP [7], and the recent work SGPMP [8]. We implemented all baselines in PyTorch except for RRT* and I-RRT*, which we plan with a loop.

Results. In point-mass and Panda environments, MPOT achieves better planning time, success rate, and parallelization quality, some with large margins, especially for the Panda experiments, while retaining smoothness due to the GP cost. We observe that MPOT performs particularly well in narrow passages, since each waypoint across all trajectories is updated independently due to Sinkhorn Step's property.

The TIAGo++ task requires designing many non-convex costs, e.g., SDFs for gradient-based planners. Moreover, the task space is larger while the $SE(3)$ goal is locally small (i.e., a grasp pose and the whole-body IK solution is not always good); hence, it typically requires long-horizon configuration trajectories and a small update step size. These factors add to the worse performance of the baselines in planning time (Table II). In contrast, MPOT achieves much better planning times by avoiding the propagation of gradients in a long computational chain (i.e., gradient propagation over whole-body kinematics) and by having an efficient update rule.

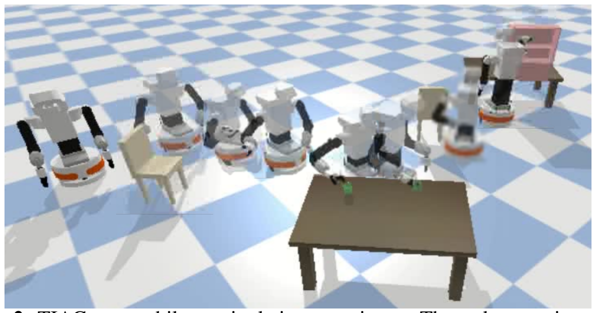
Fig. 2: TIAGo++ mobile manipulation experiment. The task comprises two parts: the *fetch* part and *place* part; thus, it requires solving two planning problems. Each plan contains 128 timesteps, and we plan a single trajectory for each planner due to the high-computational and memory demands. We generate 20 seeds by randomly spawning the robot in the room, resulting in 20 tasks.

TABLE II: Mobile fetch & place experiment. TF$[s]$ depicts the planning time for achieving first successful solution. Due to the sparsity of the 36-othorplex ($m = 72$) defining the search direction bases in this high-dimensional case, it becomes hard to balance success rate and smoothness when tuning MPOT, resulting in lower smoothness than the baselines.

	TF$[s]$	SUC[%]	S	PL
RRT*	1000 ± 0.00	0	-	-
I-RRT*	1000 ± 0.00	0	-	-
STOMP	-	0	-	-
SGPMP	27.75 ± 0.29	25	0.010 ± 0.001	6.69 ± 0.38
CHOMP	16.74 ± 0.21	40	0.015 ± 0.001	8.60 ± 0.73
GPMP2	40.11 ± 0.08	40	0.012 ± 0.015	8.63 ± 0.53
MPOT	$\mathbf{1.49} \pm 0.02$	**55**	0.022 ± 0.003	10.53 ± 0.62

VI. Conclusions

We presented MPOT - a gradient-free and highly-parallelizable motion planner that optimizes multiple high-dimensional trajectories over non-convex objectives. In particular, we proposed the Sinkhorn Step - a zero-order batch update rule parameterized by a local optimal transport plan with a nice property of cost-agnostic step bound, effectively updating waypoints across trajectories independently. We demonstrated that in practice, our method converges, scales very well to high-dimensional tasks, and provides practically smooth plans.

References

[1] J.-P. Laumond *et al.*, *Robot motion planning and control*. Springer, 1998, vol. 229.

[2] 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*, 1996.

[3] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE ICRA*, 2000.

[4] 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.

[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE ICRA*, 2009.

[6] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *IJRR*, 2018.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE ICRA*, 2011.

[8] J. Urain, A. T. Le, A. Lambert, G. Chalvatzaki, B. Boots, and J. Peters, "Learning implicit priors for motion optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[9] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[10] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1049–1056.

[11] C. Villani, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[12] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[13] A. Figalli and F. Glaudo, *An invitation to optimal transport, Wasserstein distances, and gradient flows*, 2021.

[14] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[15] J. Altschuler, J. Niles-Weed, and P. Rigollet, "Near-linear time approximation algorithms for optimal transport via sinkhorn iteration," *Advances in neural information processing systems*, vol. 30, 2017.

[16] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," in *International conference on machine learning*. PMLR, 2014, pp. 685–693.

[17] R. Hooke and T. A. Jeeves, ""direct search"solution of numerical and statistical problems," *Journal of the ACM (JACM)*, vol. 8, no. 2, pp. 212–229, 1961.

[18] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019.

[19] R. G. Regis, "On the properties of positive spanning sets and positive bases," *Optimization and Engineering*, vol. 17, no. 1, pp. 229–262, 2016.

[20] H. S. M. Coxeter, *Regular polytopes*. Courier Corporation, 1973.

[21] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, "Scaling algorithms for unbalanced optimal transport problems," *Mathematics of Computation*, vol. 87, no. 314, pp. 2563–2609, 2018.

[22] B. Schmitzer, "Stabilized sparse scaling algorithms for entropy regularized transport problems," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, pp. A1443–A1481, 2019.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[24] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[25] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[26] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.