Integrating Trajectory Optimization in Quality-Diversity for Kinodynamic Motion Planning

Konstantinos A. Asimakopoulos
Laboratory of Automation & Robotics (LAR)
Department of Electrical & Computer Engineering
and
Computational Intelligence Laboratory (CILab)
Department of Mathematics,
University of Patras, GR-26504 Patras, Greece
k_asimakopoulos@ac.upatras.gr

Konstantinos I. Chatzilygeroudis
Laboratory of Automation & Robotics (LAR)
Department of Electrical & Computer Engineering
and
Computational Intelligence Laboratory (CILab)
Department of Mathematics,
University of Patras, GR-26504 Patras, Greece
costashatz@upatras.gr

***Abstract*—Kinodynamic planning is a critical component of robotics, enabling robots to generate dynamically feasible trajectories while adhering to geometric constraints. Existing methods for kinodynamic planning primarily fall into two categories: (a) stochastic planners, such as Rapidly-Exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), which excel in state-space exploration due to their directed randomness, and (b) optimization-based approaches, which are well-suited for structured environments, producing smooth and optimal solutions. In this work, we present Hybrid Trajectory Exploration for Kinodynamic Planning (HyTraX), a hybrid framework that integrates the strengths of these two paradigms. Specifically, HyTraX enhances the effectiveness of the Go-Explore algorithm by incorporating trajectory optimization with random target selection as its core exploration strategy. We evaluate HyTraX on two challenging kinodynamic motion planning tasks: a car-like agent and a planar quadrotor navigating through maze environments. Our results demonstrate significant performance improvements with minimal task-specific tuning, highlighting the robustness and versatility of the proposed approach.**

***Index Terms*—Kinodynamic Motion Planning, Trajectory Optimization, Go-Explore, Quality-Diversity**

I. Introduction & Related Work

Kinodynamic motion planning is a fundamental problem in the field of robotics [10]. It involves finding a feasible path for an agent to move from a starting point to a desired goal while considering not only the kinematics (geometric constraints) but also the dynamics (forces, velocities and accelerations) of the robot. It ensures that the discovered trajectories are both collision-free and dynamically feasible, given the robot's physical capabilities. Whether it be a robotic arm assembling parts in a factory, a self-driving car navigating through traffic, or a drone flying through an obstacle course, motion planning is crucial for enabling agents to navigate autonomously in real-world scenarios. A wide range of diverse methods have been developed for this task. We could group these methods into three general categories:

This work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "3rd Call for H.F.R.I. Research Projects to support Post-Doctoral Researchers" (Project Acronym: NOSALRO, Project Number: 7541).

- **Sampling-based methods:** [13] Sampling-based methods explore the robot's configuration space by randomly sampling possible configurations and connecting them to build a roadmap or a tree. These methods are particularly useful for high-dimensional spaces where explicit representation of obstacles is challenging.
- **Optimization-based methods:** [9] Optimization-based methods formulate motion planning as an optimization problem, where the goal is to find a trajectory that minimizes (or maximizes) a given cost (or objective) function while satisfying constraints related to the robot's dynamics and the environment.
- **Learning-based methods:** [3], [11] Learning-based methods use machine learning techniques to improve motion planning by learning from data. These methods can learn to predict feasible and/or optimized paths, or even directly generate motion plans from sensory inputs.

Sampling-based methods, like RRT* or PRM*, can potentially discover an initial solution quickly and will even converge asymptotically over time to an optimal solution [8]. But as the complexity of the system increases the computational time required makes them inefficient [5]. This means that the discovered trajectories will not be optimal and further post processing and smoothing might be required [14], [15]. In addition, RRT* operates under the assumption that any pair of states can optimally be connected with a straight line, which, for kinodynamic systems, is typically not a valid trajectory due to the existence of differential constraints in the problems [21].

Optimization-based methods [7], [17], [18] on the other hand, can handle complex dynamics and constraints effectively and provide "high-quality" smooth trajectories. The existence of a cost function can also lead to several performance improvements, such as minimizing time, energy efficiency etc. A disadvantage of such methods is that they usually require a good initial condition as a warm start or some form of manual tuning (e.g. waypoints) to find a solution, especially in long-horizon problems [23].

Hybrid approaches in motion planning combine the strengths of different planning techniques to address the lim-

Algorithm 1 Go-Explore Framework

procedure GO-EXPLORE($\boldsymbol{x}_0$, n_{batch}, n_{iter})
  $\mathcal{A} \leftarrow$ new_cell($\boldsymbol{x}_0, \emptyset$) ▷ Initialize archive, $\mathcal{A}$, with initial state $\boldsymbol{x}_0$
  for $n = 1 \rightarrow n_{iter}$ **do**
    cells $\leftarrow$ SELECTCELLS($\mathcal{A}$, n_{batch}) ▷ Select n_{batch} cells from archive to explore from
    new_cells $\leftarrow$ EXPLORE(**cells**) ▷ Run stochastic exploration starting
    ADDTOARCHIVE($\mathcal{A}$, **new_cells**) ▷ Add explored cells to the archive
    UPDATESCORES($\mathcal{A}$) ▷ Update the scores of cells
  end for
  return $\mathcal{A}$ ▷ The outcome of the algorithm is the archive
end procedure

itations inherent in using a single method. Such techniques usually combine a sampling-based method with optimization or learning methods. In [17], [18], a random planner (like RRT*) is used to get a warm start for trajectory optimization. In [15], the authors use a big database of pre-computed motion primitives, attempt to connect them via an RRT-like method and "repair" the constraints violations via trajectory optimization. Another approach is to use reinforcement learning (RL) to train a policy that is then used as a local planner for RRT [3].

In this work, we aim at bypassing the above-mentioned drawbacks by taking advantage of ideas first proposed in [1], and provide a novel hybrid kinodynamic motion planner that has the following characteristics:

1) Does not require pre-computation of motion primitives
2) Converges to a valid solution faster than pure sampling-based planners
3) Effectively explores the state space and finds solutions in complex environments, where trajectory optimization requires extensive task-specific tuning
4) At convergence, we end up with multiple solutions instead of a single one

Overall, our proposed method effectively combines the specific characteristics of trajectory optimization and quality diversity algorithms.

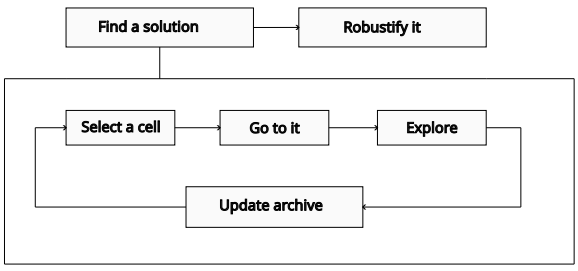

Fig. 1: Go-Explore high-level diagram

II. Proposed Approach

We propose to use trajectory optimization as the exploration strategy of Go-Explore instead of taking random actions. In this manner, we leverage the robust solutions provided by trajectory optimization on a smaller scale, all the while retaining the exploration benefits inherent in Go-Explore.

We begin by detailing the Go-Explore framework (Sec. II-A), we continue by explaining our trajectory optimization implementation (Sec. II-B) and how to deal with a bidirectional search scheme (Sec. II-C), and finally we end by describing the final proposed algorithm (Sec. II-D).

A. Go-Explore

Go-Explore [4] is a Quality-Diversity algorithm [2] and an exploration framework that aims to foster continuous exploration by (1) maintaining a global archive of all interestingly different states that were visited, as opposed to working on an archive of behaviors, and (2) exploiting deterministic simulators/models and dividing exploration into exactly returning to a state, without any intentionally added stochasticity, and then performing stochastic exploration from that state (see Fig. 1).

As the non-stochastic return to the states may lead to brittle solutions, the algorithm divides itself in two phases: (1) explore until solving the task, and (2) robustify the solution, *e.g.* by running imitation learning on a set of good trajectories found during exploration.

The first phase of Go-Explore is predicated into building a set of interesting states, paired with the sequence of actions, called *history*, that was taken to arrive to that state from the starting condition, and with a *score* indicating how interesting exploration from that state would be. A state/history/score group is called a *cell*. The set of cells is called the *archive*. The archive, $\mathcal{A}$, starts out containing only the initial state (*initial_state*). At every iteration, the algorithm performs the following steps (see also Alg. 1):

1) Choose n_{batch} cells from the archive, based on a random selection weighted on each cell's score;
2) For each selected cell, return to its state by performing the associated sequence of actions after resetting the experiment;
3) Explore from that state stochastically, by performing a sequence of short commands, generated based on an arbitrary *exploration strategy*, and recording the state of the system after every command;
4) Update the archive with the results from that exploration, by inserting the explored states and updating every cell selection scores.

Updating the cell selection scores depends on the specific score used. Insertion of a cell in the archive follows these instructions:

1) Compare the cell's state (*new_state*) to all other cells' states in the archive, according to a binary *similarity metric* (generally, comparing their distance to a predetermined delta),

2) If the state is different to all states in the archive, its cell inserted in the archive. If it is similar to one or more states, and if the new state scores higher than them in an arbitrary *insertion metric*, it replaces the corresponding cells in the archive.

In this work, we focus on the first part of Go-Explore, that is the exploration part. In particular, we explore the case of treating Go-Explore as a kinodynamic planner, and we propose an alternate exploration strategy that allows Go-Explore to effectively account for kinodynamic constraints even in high-dimensional state spaces. Although unexplored in the literature Go-Explore and RRT share many common ideas.

B. Go-Explore with Trajectory Optimization

The general optimal control problem is defined as follows [22]:

$$\underset{\boldsymbol{x}(t),\boldsymbol{u}(t)}{\operatorname{argmin}} \mathcal{J}(\boldsymbol{x}(t),\boldsymbol{u}(t)) = \int_{t_0}^{t_f} \ell(\boldsymbol{x}(t),\boldsymbol{u}(t))dt + \ell_F(\boldsymbol{x}(t_f)) \quad \text{subject to} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t)) \tag{1}$$

where[1]

- $\boldsymbol{x}(t) \in \mathbb{R}^{N_s}$ and $\boldsymbol{u}(t) \in \mathbb{R}^{N_u}$ represent the state and control trajectories,
- $\mathcal{J}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ is the cost function,
- $\ell(\boldsymbol{x}(t),\boldsymbol{u}(t))$ represents the stage cost,
- $\ell_F(\boldsymbol{x}(t_f))$ is the terminal cost,
- $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t))$ defines the dynamics constraints,
- additional constraints can be incorporated as needed.

This leads to an infinite-dimensional problem, which is generally intractable on a computer. Various techniques are used to convert it into a finite-dimensional problem, with discretization and direct collocation being the most common methods [9], [22]. Upon discretizing the problem, we arrive at the following formulation:

$$\underset{\boldsymbol{x}_{1:K},\boldsymbol{u}_{1:K-1}}{\operatorname{argmin}} \mathcal{J}(\boldsymbol{x}_{1:K},\boldsymbol{u}_{1:K-1}) = \sum_{k=1}^{K-1} \ell(\boldsymbol{x}_k,\boldsymbol{u}_k) + \ell_F(\boldsymbol{x}_K) \quad \text{subject to} \quad \boldsymbol{x}_{k+1} = f_{\text{discrete}}(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{2}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{N_s}$ and $\boldsymbol{u}_k \in \mathbb{R}^{N_u}$. Direct collocation optimizes over knot points and connects them using splines [9]. Both methods are flexible enough to produce robust solutions. For discretization, the problem is typically solved using Differential Dynamic Programming [6], [12], [22], while direct collocation utilizes general non-linear optimization solvers [9]. In this work, we adopt a direct transcription method and solve the problem using the Ipopt solver [20].

1) Trajectory Optimization as an Exploration Strategy: In the original implementation of Go-Explore the exploration strategy used is essentially a sequence of random actions. We propose to leverage trajectory optimization instead of random exploration. At the start of every exploration phase, n_{batch} cells are chosen from the archive to explore from. For every chosen *cell* we sample a random target point within a small distance. We then use trajectory optimization to find the path between the cells and the respective random target points. *In this manner, we leverage the robust and physically plausible solutions provided by optimization on a smaller scale, all the while retaining the exploration benefits inherent in the Go-Explore algorithm.*

[1]Time derivatives are indicated by an upper dot: *e.g.* $\dot{x}$.

C. Bidirectional Go-Explore

To accelerate and enhance exploration, we introduce a bidirectional Go-Explore scheme employing two agents — one forward and one backward — each initialized at different points in the state space. These agents operate with slightly distinct versions of our trajectory optimization-enhanced Go-Explore, maintaining separate archives. The forward agent follows our proposed model, while the backward agent tackles the inverse optimization problem. For the backward agent, exploration begins from a randomly sampled point, aiming to return to a designated cell. The bidirectional process involves selecting a batch of cells, restoring their states, sampling nearby random points, applying trajectory optimization in both forward and backward directions, and updating each agent's archive with the resulting optimized trajectories.

Algorithm 2 HyTraX Algorithm

procedure HyTraX($\boldsymbol{x}_0$, n_{batch}, n_{iter})
  $\mathcal{A} \leftarrow$ new_cell($\boldsymbol{x}_0, \emptyset$) ▷ Initialize archive, $\mathcal{A}$, with initial state $\boldsymbol{x}_0$
  for $n = 1 \rightarrow n_{iter}$ **do**
    cells $\leftarrow$ SelectCells($\mathcal{A}$, n_{batch}) ▷ Select n_{batch} cells from archive to explore from
    for cell in **cells do**
      $\boldsymbol{x}_{\text{target}} \leftarrow$ RandomSample($cell$) ▷ Randomly sample a target state near the cell
      $\boldsymbol{\tau} \leftarrow$ TrajectoryOptimize($\boldsymbol{x}_{\text{current}}, \boldsymbol{x}_{\text{target}}$) ▷ Optimize trajectory from current state to the target
      AddTrajectoryToArchive($\mathcal{A}, \boldsymbol{\tau}$) ▷ Add the new trajectory to the archive
    end for
    UpdateScores($\mathcal{A}$) ▷ Update the scores of cells based on new trajectories
  end for
  return $\mathcal{A}$ ▷ The outcome of the algorithm is the archive
end procedure

It is important to note that for a kinodynamic motion planning application, vanilla Go-Explore is not able to run in a bidirectional mode. This is because an optimization needs to be performed to ensure that the randomized behavior will surely reach the end point.

D. HyTraX Method

The algorithm (Alg. 2) begins by initializing an archive of explored states, $\mathcal{A}$, with an initial state, $\boldsymbol{x}_0$. In each iteration, a batch of previously explored cells (states) is selected from the archive for further exploration. For each selected cell, we

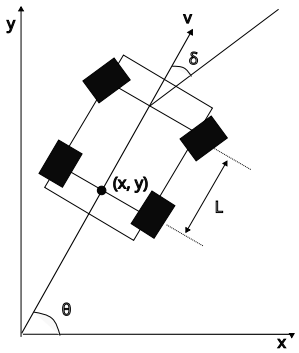

Fig. 2: Simple Car system [16].

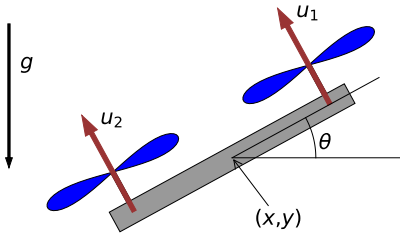

Fig. 3: Planar Quadrotor System [19].

randomly sample a nearby target state in the state space. The goal then is to compute an optimal trajectory between the current state and the sampled target state using a trajectory optimization method. Each optimized trajectory is subsequently added to the archive, expanding the known regions of the state space. The scores of cells in the archive are updated so that we can compute new selection probabilities that balance quality and diversity. The process continues for a fixed number of iterations or until the desired exploration coverage is achieved.

When the bidirectional mode is enabled, we also sample starting points for the backward archive, and use trajectory optimization to find paths between the sampled points and the selected cell (i.e. doing the opposite of the forward archive). For clarity, we did not include the second archive and the backward process in the pseudocode.

III. Experimental Setup & Results

A. Setup

We evaluated the effectiveness of HyTraX through two experimental environments: a simple car (Fig. 2) navigating a complex maze and a planar quadrotor (Fig. 3) tasked with reaching a target while traversing a maze. These environments were selected to test the method's ability to handle kinodynamic constraints and complex state spaces.

In the first experiment, we employ a skidding-free simple car model (see Eq. 4) [16] constrained to traverse a two-dimensional maze. The frequently branching and narrow paths of the maze pose significant challenges both for efficient exploration and path optimization. The car is required to find a path from an initial position to a goal state on the opposite side of the maze. The primary objective is to demonstrate that our method can efficiently explore the maze and converge to a solution path faster than vanilla Go-Explore.

The simple car model has a three-dimensional state space, representing its position (x, y) in meters and orientation θ in radians. The control space is two-dimensional, with the linear velocity v in meters per second and δ the steering angle of the front wheels in radians. So, the simple car system can be represented as:

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \mathbf{u}_k = \begin{bmatrix} v \\ \delta \end{bmatrix} \tag{3}$$

and the time derivatives of the state can be expressed as:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\delta \end{aligned} \tag{4}$$

where L is the car length in meters.

The second experiment involves a planar quadrotor (Fig. 3), a more complex kinodynamic system (see Eq. 6), which needs to reach a target state while navigating a two-dimensional maze. This system is significantly more challenging to control: it even requires a control signal just to stay idle and not fall on the ground. The dynamics of the quadrotor introduce an additional layer of complexity, as it has to account for both position and velocity in its movement through the environment. The maze includes varying heights along the branching paths, forcing the quadrotor to balance between speed and control to reach the target. The goal of this experiment is to demonstrate that HyTraX can effectively handle both the quadrotor's dynamics and the constraints imposed by the environment.

The quadrotor model has a six-dimensional state space, representing its position (x, y) in meters, orientation θ in radians and its velocities $\dot{x}, \dot{y}, \dot{\theta}$ in meters per second and radians per second respectively. The control space is two-dimensional, representing the forces from each motor (u_1, u_2) in Newtons. So, the quadrotor can be represented as:

$$\mathbf{x}_k = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T, \mathbf{u}_k = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{5}$$

and the time derivatives of the state can be expressed as:

$$\begin{aligned} \ddot{x} &= -\frac{(u_1+u_2)\sin\theta}{m} \\ \ddot{y} &= \frac{(u_1+u_2)\cos\theta}{m} - g \\ \ddot{\theta} &= \frac{u_2-u_1}{0.4ml} \end{aligned} \tag{6}$$

where m is the mass of the quadrotor in kg, l its length in meters and g the gravitational constant.

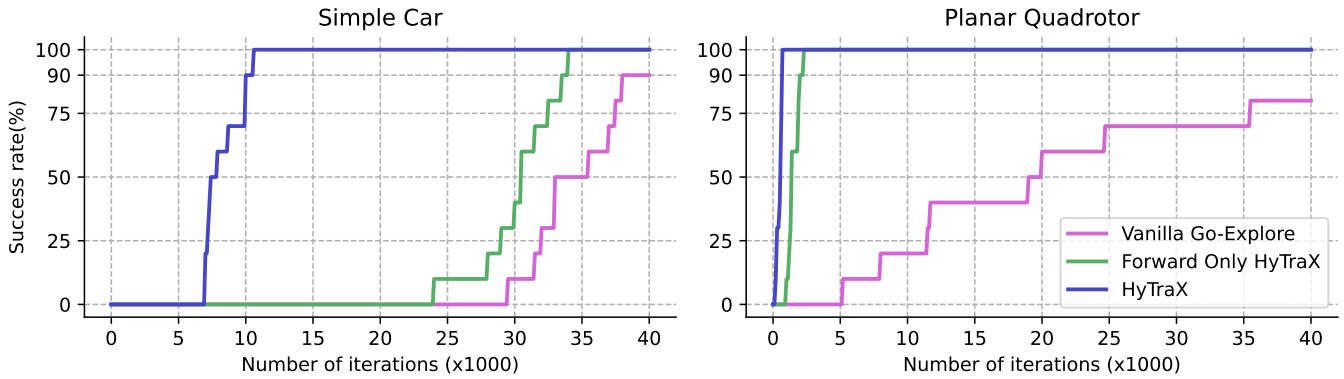

Fig. 4: Success rate for both experiments after 10 runs of the experiments with the various methods.

B. Results

We run both experiments 10 times for a maximum of 40000 iterations. We compare the results of three different methods: ours (HyTraX), our method without the bidirectional functionality (Forward-Only HyTraX) and the Vanilla Go-Explore algorithm. The results are shown in Fig. 4, 5, 6.

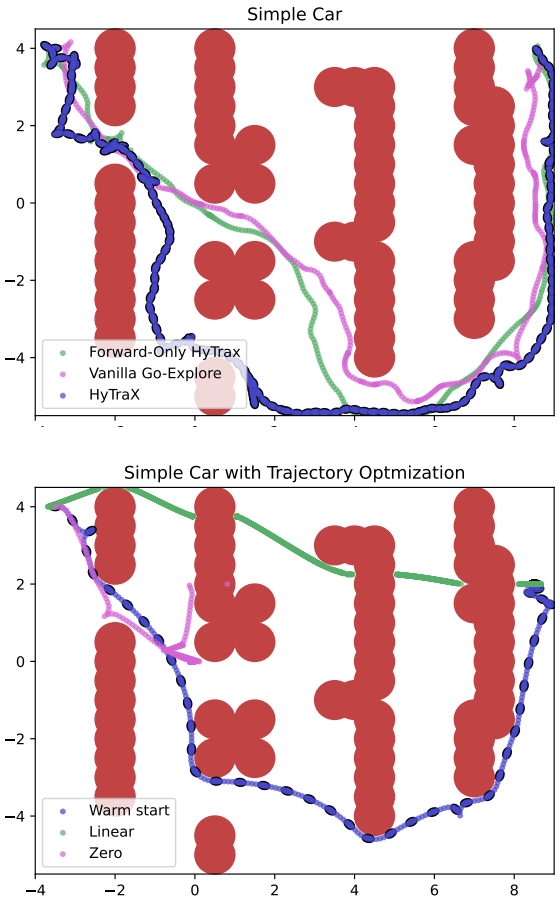

Fig. 5: **Top**: Shortest paths to the target discovered by the different methods in the simple car experiment. **Bottom**: Paths found using only trajectory optimization. Without a good warm start, trajectory optimization fails to solve the problem.

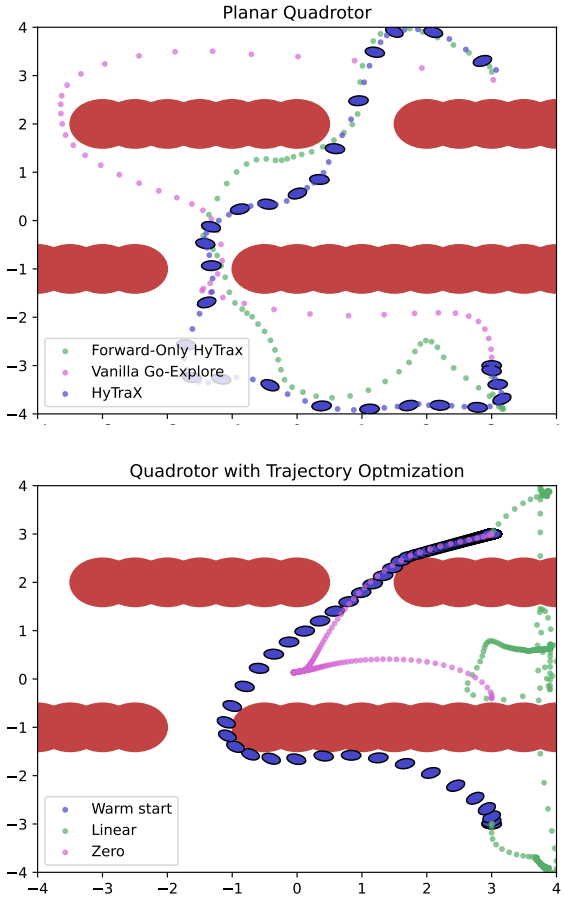

Fig. 6: **Top**: Shortest paths to the target discovered by the different methods in the planar quadrotor experiment. **Bottom**: Paths found using only trajectory optimization. Without a good warm start, trajectory optimization fails to solve the problem.

In the simple car experiment (Fig. 5) which is a highly constrained environment, HyTraX converges significantly faster than both the other methods, as shown in Fig. 4 and achieves a perfect 100% success rate over all runs. The advantage of using a bidirectional scheme for exploration is clearly showcased in the results; in a highly cluttered environment

such as a narrow maze, the number of iterations needed to achieve a perfect score are significantly fewer.

In the quadrotor experiment (Fig. 6) our method (both in bidirectional and forward-only mode) achieves 100% success rate in less than 3000 iterations on average, while the Vanilla Go-Explore needs almost 40000 for an 80% success rate. This shows the benefit of using trajectory optimization as the exploration strategy. Because of the higher dimensionality of the system, taking random actions makes it very difficult to effectively control the quadrotor and find a feasible path to the goal point. Trajectory optimization easily overcomes this problem and therefore quickly discovers kinodynamically feasible paths for the quadrotor.

One of the main advantages of HyTraX is that it requires no manual tuning to solve the problem. To show this we attempted to solve the same problems using only classic trajectory optimization. We tried three different modes: a) using some good manually selected waypoints as a warm start for the optimizer, b) using a linear interpolation between start and finish as an initial guess and c) setting the initial guess to zero. The optimizer failed to find dynamically feasible paths without a good initial condition as a warm start (Fig. 5, and 6).

IV. Conclusion

In this work, we presented HyTraX, a hybrid kinodynamic planning method that integrates trajectory optimization with the exploration capabilities of the Go-Explore algorithm. By leveraging a bidirectional approach with forward and backward agents, our method effectively improves exploration and planning performance, demonstrated on two challenging kinodynamic motion planning problems. The combination of randomized exploration and trajectory optimization leads to significant performance gains with minimal task-specific tuning and no precomputed motion primitives.

However, our approach has limitations. The current method relies on the quality of the randomly sampled points, which can result in inefficiencies for highly complex state spaces. Additionally, while trajectory optimization enhances exploration, it introduces computational overhead that may limit scalability for complicated systems (e.g. walking robots).

Future work will explore ways to mitigate these limitations. This includes incorporating more intelligent sampling strategies to reduce dependence on random points, as well as optimizing the trajectory update process to improve computational efficiency. We also plan to investigate the use of shared archives or more dynamic inter-agent communication to further boost exploration capabilities. Moreover, extending the approach to higher-dimensional systems and real-world robots will be a key area of focus. Finally, we aim at theoretically analyzing our approach such that we can provide interesting guarantees (e.g. probabilistic completeness).

References

[1] Konstantinos A Asimakopoulos, Aristeidis A Androutsopoulos, Michael N Vrahatis, and Konstantinos I Chatzilygeroudis. Effective kinodynamic planning and exploration through quality diversity and trajectory optimization. In *The 18th Learning and Intelligent OptimizatioN Conference (LION)*, 2024.

[2] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, pages 109–135. Springer, 2021.

[3] Hao-Tien Lewis Chiang, Jasmine Hsu, Marek Fiser, Lydia Tapia, and Aleksandra Faust. Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies. *IEEE Robotics and Automation Letters*, 4(4):4298–4305, 2019.

[4] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[5] Dibyendu Ghosh, Ganeshram Nandakumar, Karthik Narayanan, Vinayak Honkote, and Sidharth Sharma. Kinematic constraints based bi-directional RRT (KB-RRT) with parameterized trajectories for robot path planning in cluttered environment. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8627–8633. IEEE, 2019.

[6] Wilson Jallet, Antoine Bambade, Etienne Arlaud, Sarah El-Kazdadi, Nicolas Mansard, and Justin Carpentier. Proxddp: Proximal constrained trajectory optimization. *Preprint*, 2023.

[7] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[8] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[9] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[10] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[11] Abu Layek, Ngo Anh Vien, TaeChoong Chung, et al. Deep reinforcement learning algorithms for steering an underactuated ship. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 602–607. IEEE, 2017.

[12] Carlos Mastalli, Wolfgang Merkt, Josep Marti-Saumell, Henrique Ferrolho, Joan Solà, Nicolas Mansard, and Sethu Vijayakumar. A feasibility-driven approach to control-limited ddp. *Autonomous Robots*, 46(8):985–1005, 2022.

[13] Andreas Orthey, Constantinos Chamzas, and Lydia E Kavraki. Sampling-based motion planning: A comparative review. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[14] Joaquim Ortiz-Haro, Wolfgang Hoenig, Valentin N Hartmann, and Marc Toussaint. idb-a*: Iterative search and optimization for optimal kinodynamic motion planning. *arXiv preprint arXiv:2311.03553*, 2023.

[15] Joaquim Ortiz-Haro, Wolfgang Hönig, Valentin N Hartmann, Marc Toussaint, and Ludovic Righetti. idb-rrt: Sampling-based kinodynamic motion planning with motion primitives and trajectory optimization. *arXiv preprint arXiv:2403.10745*, 2024.

[16] Romain Pepy, Alain Lambert, and Hugues Mounier. Path planning using a dynamic vehicle model. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 781–786. IEEE, 2006.

[17] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation*, pages 489–494. IEEE, 2009.

[18] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Berlin, Germany, 2013.

[19] Russ Tedrake. *Underactuated Robotics*. Course Notes for MIT 6.832, 2023.

[20] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[21] Dustin J Webb and Jur van den Berg. Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints. *arXiv preprint arXiv:1205.5088*, 2012.

[22] Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 2023.

[23] Zhigen Zhao, Shuo Chen, Yan Ding, Ziyi Zhou, Shiqi Zhang, Danfei Xu, and Ye Zhao. A survey of optimization-based task and motion planning: From classical to learning approaches. *arXiv preprint arXiv:2404.02817*, 2024.