2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9340761

# Energy-Efficient Motion Planning for Multi-Modal Hybrid Locomotion

H.J. Terry Suh[1], Xiaobin Xiong[2], Andrew Singletary[2], Aaron D. Ames[2], Joel W. Burdick[2]

***Abstract*— Hybrid locomotion, which combines multiple modalities of locomotion within a single robot, enables robots to carry out complex tasks in diverse environments. This paper presents a novel method for planning multi-modal locomotion trajectories using approximate dynamic programming. We formulate this problem as a shortest-path search through a state-space graph, where the edge cost is assigned as optimal transport cost along each segment. This cost is approximated from batches of offline trajectory optimizations, which allows the complex effects of vehicle under-actuation and dynamic constraints to be approximately captured in a tractable way. Our method is illustrated on a hybrid double-integrator, an amphibious robot, and a flying-driving drone, showing the practicality of the approach.**

## I. Introduction

A hybrid locomotor combines multiple movement modalities into a single platform. Examples of hybrid locomotion include amphibious vehicles with the ability to swim and drive, or flying cars with the ability to drive and fly. Hybrid locomotion can allow robots to tackle more complex tasks in complicated environments, while achieving greater performance, such as improved energy efficiency. For instance, a flying-car can readily fly over obstacles or uneven terrain via aerial mobility, while driving when possible to save energy (see Fig. 1 for examples).

Prior works on hybrid locomotion have investigated the design and feasibility of hybrid locomotion strategies ([1], [2], [3], [4], [5], [6]); however, realizing the full potential of these robots not only depends on clever design, but also on autonomous planning of their complex motion strategies. The continuous inputs, combined with discrete mode switches, produce entirely different energy costs, travel times, and robustness, which ultimately dictate performance. The complexity of combinatorial optimization of the switching sequences, as well as trajectory optimization within each modality, makes this problem particularly challenging. Directly transcribing this problem into a mixed-integer program [7], [8] may not scale well enough to handle switching sequences and coordinates of high-dimensional problems.

This paper presents a novel motion-planning method for hybrid locomotion using approximate dynamic programming. Our solution relies on the following key insight: jointly optimizing for the continuous and discrete decision variables in the space of policies is difficult, but we can use the approximate optimal cost of a continuous trajectory segment as a proxy to the effect of optimal continuous policies within each modality. This allows us to decouple the original mixed-integer problem into discrete and continuous problems by first performing a graph search with approximated energy costs, then doing a final multi-phase continuous optimization with the obtained sequence and switch coordinates.

Fig. 1. Top Row: A "Drivocopter" Drone (developed by the authors) which can fly and drive. Video of operation can be accessed at https://www.youtube.com/watch?v=QZyuvXfifvQ. Bottom Row: "Ambot" Amphibious Robot [1] capable of ground and marine locomotion.

Prior work on multi-modal planning often only consider geometric graph-based planning ([9],[10],[11],[12]). By ignoring the continuous dynamics of the robot, these planners often ignore dynamic feasibility. In addition, although energy expenditure is often the most relevant cost in hybrid locomotion, most works often assume that energy is linearly proportional to geometrical distance traveled ([9],[11],[13]), which ignore how dynamic constraints affect the optimal cost. As our cost is approximated from full-dimensional trajectory optimization with rigid body and motor dynamics, it accurately captures the robot's dynamic characteristics, including effects of underactuation that might occur in some locomotion modalities.

Other relevant work can be found in the concept of "Combined Task and Motion Planning" (TAMP) [14], [15], [16], in which a discrete sequence of tasks must be found simultaneously with the geometric motions that satisfy these sequences. Most of these works are in the domain of manipulation, where there are little-to-no continuous dynamic effects, which are often critical for locomotion. In addition, whereas each discrete task changes the geometric preconditions that affect the next task in a manipulation sequence, we assume a fixed geometric map in our framework.

Within the field of robotic locomotion, there has been considerable effort to solve the "Hybrid Activity and Trajectory Planning" (HATP) problem [9], [17], [18], [13],

* This work is supported by NSF Award No. 1932091

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, hjsuh@mit.edu

[2]Deptartment of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125, USA, {xxiong,asinglet, ames}@caltech.edu, jwb@robotics.caltech.edu

where individual tasks of multiple robots, as well as their trajectories, must simultaneously be planned. Some of these planners also take into account the dynamic behavior of the robot. However, due to the difficulty of simultaneously doing trajectory generation with task planning, these approaches often consider simplified forms of dynamics, such as constant rate [17], or first-order behavior [18], that admit fast and convex formulations of trajectory planning. By alleviating the need to compute exact control inputs in the discrete-planning process, and capturing their effects with offline-approximated costs, our approach allows tractable utilization of realistic robot dynamics.

Our framework is most related to [13], which uses hierarchical planning for multi-agent systems: a graph-search first creates a global plan for multiple agents, and a local controller is used for the agents to track the global plan. The cost function in their graph utilizes the value function of the closed-loop policy that is computed offline, which is similar to our offline cost approximation. However, while [13] utilizes distance-dependent energy costs with a double-integrator model for their robot dynamics, we directly optimize for electrical energy expenditure, thus capturing a more detailed and dynamically accurate behavior of the robot. This further emphasizes the full power of this framework.

The proposed method is primarily implemented in simulation: the hybrid double-integrator with viscous friction is shown as a low-dimensional case (Sec.IV). Then, example trajectories for more realistic systems are given by considering amphibious (Sec.V) and flying-driving locomotion (Sec.VI). As most hybrid planners ([17], [18]), our planner does not guarantee probabilistic optimality due to the underlying heuristics required to solve the problem, but we show that our method performs quantitatively well in practice.

## II. Problem Formulation

### A. The Hybrid Locomotion System

We define a hybrid locomotion system as a type of hybrid control system [19], [20] with additional constraints. The hybrid locomotion system, $\mathscr{HL}$, is defined as a tuple

$$\mathscr{HL} = (FG, \mathcal{D}, \mathcal{U}, \mathcal{S}, \Delta). \tag{1}$$

In the following descriptions of each system element, $i$ indexes the locomotion mode (i.e. flying or driving):

- $FG = \{(f_i, g_i)\}$ describes the dynamics associated with each locomotion mode, which are assumed to take a control-affine form: $\dot{x} = f_i(x) + g_i(x)u$.
- $\mathcal{D} = \{\mathcal{D}_i\}$ is the set of domains, or state-spaces, associated with the continuous dynamics of each mode.
- $\mathcal{U} = \{\mathcal{U}_i\}$ is the set of admissible control inputs associated with each mode.
- $\mathcal{S} = \{S_{i,j}\}$ is the set of guard surfaces that describes the boundaries between domains of mode $i$ and $j$.
- $\Delta = \{\Delta_{i\to j}\}$ is the set of reset maps that describe discrete transformations on the guard surface $S_{i,j}$

We additionally assume that each state $x \in \bigcup \mathcal{D}_i$ belongs to a single mode $i$. i.e., the domains disjointly partition the reachable state-space.

### B. Optimal Trajectories in the Hybrid Locomotion System

To define an optimal hybrid trajectory, we formulate a cost for each mode's control-affine system in Bolza form:

$$J_i = \Phi_i(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}_i(x(t), u(t), t)dt. \tag{2}$$

There also exists a constant switching cost $J(\Delta_{i\to j})$ to transition from one modality $i$ to $j$. We formulate the problem of finding the optimal trajectory for a hybrid locomotion system as the following two-point boundary value problem:

$$\begin{aligned} \min_{u} \quad & \sum J_i + J(\Delta_{i\to j}) \\ \text{s.t.} \quad & \dot{x} = f_i(x) + g_i(x)u \quad \forall x \in \mathcal{D}_i \quad u \in \mathcal{U}_i \quad \forall i, \\ & x(t_0) = x_0, \quad x(t_f) = x_f. \end{aligned} \tag{3}$$

We want to find a trajectory that is dynamically feasible within each modality, while optimizing the cost functional throughout the entire trajectory, which would also require optimizing the order of discrete modes to visit. This problem can be transformed into an instance of a mixed-integer nonlinear program [8].

## III. Planning Methodology

Sec. III-A summarizes our planning method that combines sampling-based planning with local trajectory optimization. To extend this approach to problems with high dimensions, Sec. III-B introduces virtual constraints and cost approximation to improve real-time performance.

### A. Dynamic Programming with Continuous Optimization

*1) Graph Structure:* First, we discretize the problem by sampling coordinates in the free state-space of each domain $\mathcal{D}_i$. The vertices, $V$, of a digraph, $G(V, E)$, are constructed from these samples, similar to the framework of Probabilistic Roadmaps (PRM) [21]. The edges represent locally optimal paths between the vertices. Each edge is weighted with the optimal transport cost. To avoid the situation where the two vertices of an edge lie in different modalities, we additionally impose the following constraints:

1) $e = (x_i \to x_j) \in E$, $x_i, x_j \in \mathcal{D}_k$ for some mode $k$. i.e., edges only connect states in the same mode.
2) We explicitly sample the guard surface and only allow paths to cross a guard through a guard sample point.

Fig.2. A and B illustrate these conditions. The shortest-path search is tackled by Djikstra's algorithm [22] once the locally optimal trajectory costs are known.

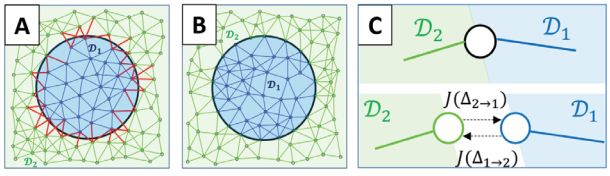


Fig. 2. A) red edges cross a guard surface between domains $\mathcal{D}_1$ and $\mathcal{D}_2$, violating the constraint on edges. B) By sampling on the guard surface and allowing no edges between $\mathcal{D}_1$ to $\mathcal{D}_2$, all graph edges are constrained to a single mode. C) Node augmentation to handle modality switching costs.

If there exists a switching cost from one modality to another, we augment the sample on the guard surface $x$ with two connected nodes $x_i$ and $x_j$ that shares the same state-space coordinates, and assign switching cost to the edge cost between the two samples, as illustrated in Fig. 2.C.

*2) Continuous Optimization of Trajectory Segments:* As each edge connects states in a single mode, we assign the edge weight by solving the optimization problem:

$$\begin{aligned} w(x_1 \to x_2) = \min_{u} \quad & J_i \\ \text{s.t.} \quad & \dot{x} = f_i(x) + g_i(x)u,\ x \in \mathcal{D}_i,\ u \in \mathcal{U}_i, \\ & x(t_0) = x_1, \quad x(t_f) = x_2, \end{aligned} \tag{4}$$

where $x_1, x_2 \in \mathcal{D}_i$. The result of this problem is used as the running cost in our dynamic programming framework. This standard trajectory optimization problem can be solved using existing methods, such as direct collocation [23].

*3) Final Path Smoothing:* The path(s) returned from graph-search are smoothed via trajectory optimization, knowing the switching sequence and the guard surface points. Given a path of samples $P = (x_1, x_2, \cdots, x_k)$ resulting from graph search, we partition the samples using their modalities:

$$\begin{aligned} \bigcup P_i = P_1 = \{x_i | 0 \le i \le k_1, \forall x_i \in \mathcal{D}_{j_1}\} \ \cup \cdots \cup \\ P_n = \{x_i | k_{n-1} \le i \le k_n, \forall x_i \in \mathcal{D}_{j_n}\}, \end{aligned} \tag{5}$$

where $j_i$ denotes the mode of each partition, and $x_{k_i}$ denotes the sample on the guard surface where the trajectory switches modes. The optimal trajectories between boundary points are then found by re-solving the partition-wise trajectory optimization problem. The total trajectory is reconstructed by concatenating the partition-wise optimal trajectories. Existing works have shown that this type of final smoothing, which delays the final choice the control inputs until the end, often leads to much enhanced performance [16], [18].

In addition, although we assume no significant presence obstacles in this work, we note that the result of the path of samples can be used as a nominal collision-free trajectory that can be used to bound the final smoothing process [24], [25], which is a direction we further plan to investigate.

### B. Extension to High Dimensions

Although dynamic programming with running cost of continuous trajectory optimization shows good promise, it is computationally expensive, requiring $O(|V|^2)$ instances of trajectory optimization. In high dimensions, the number of samples increases exponentially if the resolution is maintained, and trajectory optimization methods scale poorly. The two methods introduced in this section aim to make this method tractable for high-dimensional systems.

*1) Virtual Constraints for Search-Space Reduction:* We can reduce the dimensionality of the sample space by introducing heuristic virtual constraints that fix some coordinates as functions of the sampled-coordinates. The state-space is divided into sampled coordinates ($x^s$) and auxiliary coordinates ($x^a$), which are functions of the sampled coordinates:

$$x = (x^s, x^a)^T = (x^s, v(x^s))^T. \tag{6}$$

The state partioning into $x^s$ and $x^a$ is problem-dependent, but can be understood in the context of model-order reduction: if the original system and the virtually-constrained system show bounded difference in their evolution, it indicates a good choice of coordinates and constraints. Point-mass coordinates of position and velocity [20], or differentially flat coordinates [26] can be good choices. Eliminating the sampling of the subspace $x^a$ can significantly reduce computation, making the method tractable.

*2) Approximate Dynamic Programming:* To find the weight between two sampled coordinates $x_1^s$ and $x_2^s$ in the graph, let us first define a function $J : \mathbb{R}^{\dim(x^s)} \times \mathbb{R}^{\dim(x^s)} \to \mathbb{R}$, which is described by the following optimization problem:

$$\begin{aligned} J(x_1^s, x_2^s) = \min_{u} \quad & J_i \\ \text{s.t.} \quad & \dot{x} = f_i(x) + g_i(x)u,\ \ x \in \mathcal{D}_i,\ \ u \in \mathcal{U}_i, \\ & x_0 = [x_1^s, v(x_1^s)]^T, x_f = [x_2^s, v(x_2^s)]^T \end{aligned} \tag{7}$$

where $(x_1^s, v(x_1^s))^T, (x_2^s, v(x_2^s))^T \in \mathcal{D}_i$. Since this optimization problem has to be solved $O(|V|^2)$ times, we choose to learn a function approximator offline.

Using $(x_1^s, x_2^s)$ as feature vectors, and $J(x_1^s, x_2^s)$ as label, we first produce a batch $\left((x_1^s, x_2^s), J(x_1^s, x_2^s)\right)$ from multiple trajectory optimization runs. Then, function approximators from supervised learning algorithms such as Support Vector Regression (SVR) [27] or Neural Nets are used to approximate $J(x_1^s, x_2^s)$. Denoting the approximated function as $\tilde{J}(x_1^s, x_2^s)$, the weights on the graph are assigned by $w(x_1^s \to x_2^s) = \tilde{J}(x_1^s, x_2^s)$.

Since $\tilde{J}$ is learned offline, its evaluation does not require a full instance of nonlinear programming, greatly reducing online computation. Yet, as $\tilde{J}$ is learned from trajectory optimization, all costs in Bolza form can be utilized, and dynamic or temporal constraints can be incorporated.

## IV. Case Study: Hybrid Double Integrator

This section first verifies our low-dimensional method for 1D problem of a thrust-vectored mass on a linear rail, with viscous drag appearing at $p \ge 0$. This can be formulated as a hybrid locomotion system with the dynamics of:

$$\ddot{p} = u \text{ if } p < 0, \quad \ddot{p} = u - \dot{p} \text{ if } p \ge 0. \tag{8}$$

In addition, consider that we have the input constraint $|u| \le 1$ for both domains. Converting this to a first-order system $x = \left(p, v\right)^T$, the system can be described as:

$$\mathscr{HL} = \begin{cases} FG & = \{(f_-, g_-), (f_+, g_+)\}, \\ \mathcal{D} & = \{\{x | p < 0\}, \{x | p \ge 0\}\} \\ \mathcal{U} & = \{\{u | |u| \le 1\}, \{u | |u| \le 1\}\}, \\ \mathcal{S} & = \{S_{+,-} = \{x | p = 0\}\}, \\ \Delta & = \{\Delta_{+,-} = x^+ \to x^-\} \end{cases} \tag{9}$$

where the dynamics are described by,

$$f_- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ f_+ = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix},\ g_+ = g_- = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{10}$$

Then, let us find a trajectory from $x_i$ to $x_f$ while minimizing the input,

$$J_- = J_+ = \int_{t_0}^{t_f} u^2 dt. \tag{11}$$

Using our framework, we first place a graph structure on the state-space using knowledge of the domains $\mathcal{D}_i$, then optimize each continuous trajectory using GPOPS-II [23] with IPOPT [28] solver. The trajectory obtained using graph search, and the final smoothened trajectory using the knowledge of the switching sequence and the boundary points on the guard surface is displayed in Fig.3.

Finally, since the switching sequence is trivial to guess for this example, we utilize multi-phase optimization in GPOPS-II with IPOPT, which puts an equality constraint from the end of the first phase in $\mathcal{D}_+$ and the beginning of the second phase in $\mathcal{D}_-$ and compare the results. The trajectory using multi-phase optimization is displayed in Fig.3.

To empirically study the effect of having increased number of samples, we run the algorithm 10 times with different inter-sample distances (controlled by Poisson disc sampling [29] on the state-space), and show convergence in Fig. 4. Fig. 4 shows that our method results in a lower cost compared to multi-phase optimization, with inter-sample distance as large as 0.3. Although the dynamics are linear and cost is quadratic, the problem is no longer convex in the switching coordinate. Thus, our PRM framework, which searches more globally over the domain, performs better than the local optimum provided by IPOPT [28].

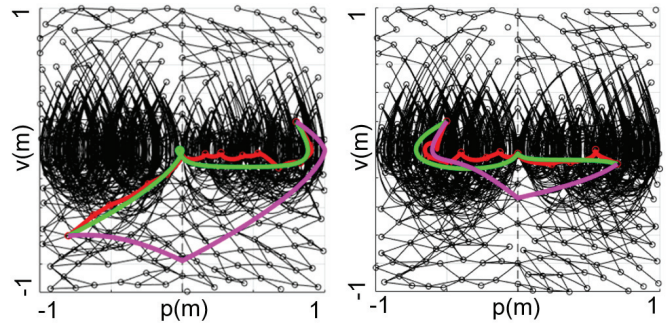


Fig. 3. Optimal Trajectories from $x_i = [0.8, 0.2]^T$ to $x_f = [-0.8, -0.]^T$ (left), and from $x_i = [0.7, -0, 1]^T$ to $x_f = [-0.5, 0.2]^T$ (right). Red trajectories are obtained using graph search, green trajectories are results of final smoothened path, and the pink trajectory is result of Multi-phase optimization in GPOPS-II [23]. The edges represent optimal trajectories between each sample.

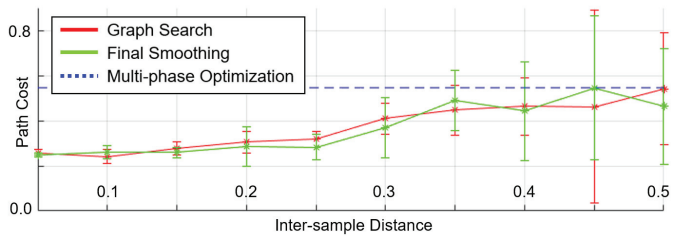


Fig. 4. Empirical convergence of cost with decreasing intersample distance.

## V. 2D Case Study: Amphibious Tank (AmBot)

This section describes optimal trajectories for the amphibious vehicle introduced in [1], which uses tank treads for ground locomotion (skid-steer), and marine locomotion (paddles). After describing the vehicle dynamics in both modes, we obtain optimal trajectories for an example environment.

### A. Dynamics

*1) Ground and Marine Dynamics:* We derive the Newtonian mechanics for planar motion, and incorporate first-order armature motor dynamics. The ground states, $x_g \in \mathbb{R}^6$, and marine states, $x_m \in \mathbb{R}^8$, are defined as

$$\begin{cases} x_g &= (p_b^w, v^b, \theta_b^w, \omega^b)^T \\ x_m &= (p_b^w, v^b, \theta_b^w, \omega^b, \phi_L, \phi_R)^T, \end{cases} \tag{12}$$

where $p_b^w \in \mathbb{R}^2$ is the body position with respect to (wrt) a world frame, $v^b \in \mathbb{R}^2$ is the velocity in the body frame, and $\theta_b^w, \omega^b \in \mathbb{R}$ denote the orientation and angular velocity wrt a world frame. Finally, $\phi_L, \phi_R \in \mathbb{R}$ denote the left and right motor speeds. In both locomotion modes, the control action $u_g = u_m = (u_L, u_R)^T \in [-1, 1]^2$ correspond to commanded motor speeds via fraction of applied motor voltage.

We model a no-slip constraint for ground operation. A $1^{st}$-order motor model relates motor torque (which generates tractive forces on the vehicle) to command inputs. A drag force proportional to the square of vehicle speed and a similar $1^{st}$-order motor model are used in the aquatic domain.

*2) Hybrid Dynamics:* The governing dynamical systems for each mode are represented by the hybrid dynamics

$$FG = \begin{cases} \dot{x}_g = f_g(x_g) + g_g(x_g)u_g & x \in \mathcal{D}_g \\ \dot{x}_m = f_m(x_m) + g_m(x_m)u_m & x \in \mathcal{D}_m, \end{cases} \tag{13}$$

where $g, m$ denotes ground and marine modes. The domains and guard surfaces $\mathcal{D}_g, \mathcal{D}_m, \mathcal{S}_{m,g}$ are obtained from terrain, and $\mathcal{U}_g = \mathcal{U}_m = [-1, 1]^2$ for both inputs. We apply the identity map to $\Delta_{m\to g}$ and $\Delta_{g\to m}$.

*3) Cost Function:* We minimize the robot's total energy expenditure, modeled as:

$$J_g = J_m = \int_{t_i}^{t_f} \left( \sum_{i=L,R} V_{cc} u_i \cdot \frac{k_t}{R}(V_{cc} u_i - k_t \phi_i) + P_d \right) dt, \tag{14}$$

where $V_{cc}$ is the battery voltage, $k_t$ is the motor torque-constant, $R$ the internal resistance, and $P_d$ the constant power drain. The first term models actuator power dissipation, and the latter term models constant power drainage from on-board electronics. We assume no switching costs associated with the discrete reset map, $J(\Delta_{m\to g}) = J(\Delta_{g\to m}) = 0$

### B. Learning the Cost Approximator

For ground operation, we divide the state-space $x_g \in \mathbb{R}^6$ into sampled and auxiliary coordinates

$$x_g^s = (p_x^w, p_y^w, v_x^b, \theta)^T,\ x_g^a = (v_y^b, \omega)^T = (0, 0)^T. \tag{15}$$

This heuristic division of coordinates recognizes that side-slip is constrained for skid-steer vehicles, and angular velocity is small. For marine operation, $x_m$ is divided into

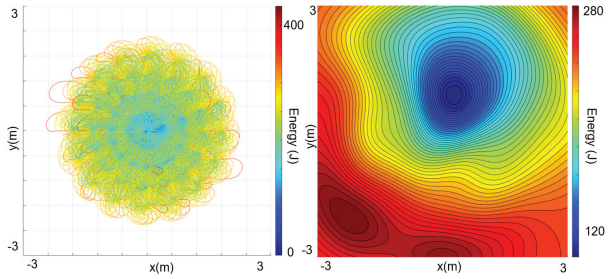


Fig. 5. Left: 11520 ground trajectories colored by their cost. Right: $xy$-energy contour for $v_x^b = 1.0m/s$, $\theta = \pi/2$. The heatmap corresponds to the energy cost to go from $x_i^s = [0,0,0,0]^T$ to $x_f^s = [x, y, 1.0, \pi/2]^T$. The approximated cost captures complex effects of underactuation.

$$\begin{cases} x_m^s = (p_x^w, p_y^w, v_x^b, \theta)^T \\ x_m^a = (v_y^b, \omega, \phi_L, \phi_R)^T = (0, 0, \phi_n, \phi_n)^T, \end{cases} \tag{16}$$

where track forces equal water drag at equilibrium speed $\phi_n$.

The cost $J(x_i^s, x_f^s)$ in (7) is approximated from multiple optimal trajectories evaluated offline. Using 11520 samples, the function $J(x_i^s, x_f^s)$ is evaluated using GPOPS-II [23], and SVR with Gaussian kernel trains the function $\tilde{J}$ with Sequential Minimal Optimization [30]. The process is repeated for ground and marine locomotion. Fig.5 shows the resulting trajectories and the learned function's contour.

### C. Results

We sample position using the method of Sec.III.A, and grid the states $v_x, \theta$ to create $x^s$. The edge weights are estimated from the learned function $\tilde{J}$. Finally, the shortest path is found by Djikstra's algorithm [22]. Fig. 6 illustrates this process. The final smoothed trajectory is shown in Fig.7.

The final trajectories differ noticeably from those produced by a shortest-path planner due to the differences in Costs of Transport. Since the robot expends more energy in water, it drives further on the ground until it switches to swimming. This example shows that our method exhibits reasonably correct qualitative behavior.

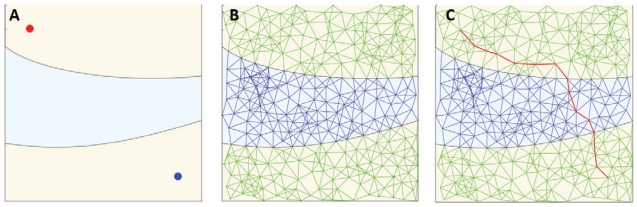


Fig. 6. A: Model Environment. B: Graph Generation. C. Result of shortest path search.

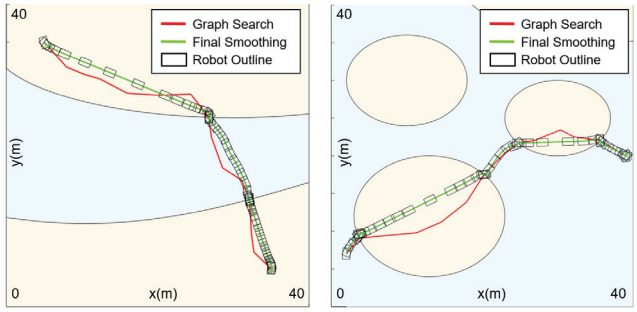


Fig. 7. Final trajectories for example of river crossing (left), and island crossing (right). The Robot outline is displayed at equal time differences.

## VI. 3D Case Study: Drivocopter

This section models the *Drivocopter* flying-driving drone of Fig. 1. It uses skid-steer driving and quadrotor flight.

### A. Dynamics

We use the ground model of Sec.V with different parameters, while the flight dynamics are based on [31] and [32].

*1) Flight Dynamics:* Standard rigid-body dynamics [33] describe flight motions driven by four rotor forces, which use a speed-squared-dependent lift term and $1^{st}$-order armature motor dynamics. The state vector $x_f \in \mathbb{R}^{16}$ is

$$x_f = (p_b^w, v^b, \Theta_b^w, \omega^b, \phi_i)^T, \tag{17}$$

where $p_b^w \in \mathbb{R}^3$ is the vehicle position wrt a world frame, $v^b \in \mathbb{R}^3$ is the 3D velocity in the body frame, $\Theta_b^w \in \mathbb{R}^3$ denotes vehicle orientation w.r.t the world frame, parametrized by ZYX Euler angles, $\omega^b \in \mathbb{R}^3$ is the body angular velocity, and $\phi_i = (\phi_1, \phi_2, \phi_3, \phi_4) \in \mathbb{R}^4$ are the motor rotational speeds.

*2) Hybrid Dynamics:* Again, the two modalities of ground and flight are represented by a hybrid dynamical system

$$FG = \begin{cases} \dot{x}_f = f_f(x_f) + g_f(x_f)u_f & x_f \in \mathcal{D}_f \\ \dot{x}_g = f_g(x_g) + g_g(x_g)u_g & x_g \in \mathcal{D}_g, \end{cases} \tag{18}$$

where $f, g$ denotes flight and ground modes, the domains and guard surfaces $\mathcal{D}_f, \mathcal{D}_g, \mathcal{S}_{f,g}$ are obtained from the ground surface geometry. The motor inputs are $\mathcal{U}_f = (u_1, u_2, u_3, u_4) = [0,1]^4$ with $\mathcal{U}_g = (u_L, u_R) = [-1,1]^2$. Finally, $\Delta_{f\to g}$ (landing) and $\Delta_{g\to f}$ (takeoff) are discrete transitions:

$$\begin{cases} \Delta_{f\to g} = (p_x^w, p_y^w, p_z^w, 0^9, \phi_n) \to (p_x^w, p_y^w, 0^4) \\ \Delta_{g\to f} = (p_x^w, p_y^w, 0^2, \theta_b^w, 0) \to (p_x^w, p_y^w, p_z^w, 0^9, \phi_n), \end{cases} \tag{19}$$

where $\phi_n$ is the motor speed needed to provide hovering lift. During takeoff, we set $p_z^w$ to be a meter higher than the ground surface of the ground sample.

*3) Cost Function:* We use the same ground energy cost as (14), and formulate the same energy for flight with different motor parameters. The costs for reset maps $J(\Delta_{f\to g})$ and $J(\Delta_{g\to f})$ are constant takeoff and landing energy costs obtained via trajectory optimization.

### B. Learning the Cost Approximator

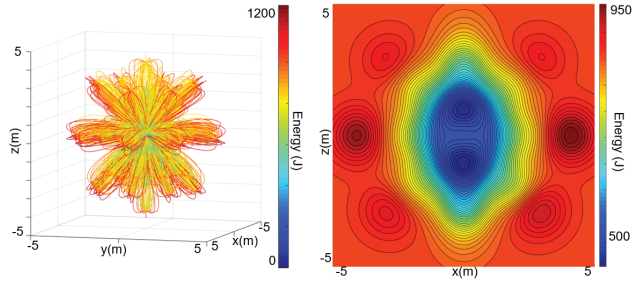


Fig. 9. Left: 17016 trajectories produced to learn the flight energy function. Right: xz-projection of the learned function $\tilde{J}(0, x)$.

The ground states are divided into sampled / auxiliary coordinates via (15). Flight states are divided by:

$$\begin{cases} x_f^s = (p_b^w, v^b)^T \\ x_f^a = (\Theta_b^w, \omega^b, \phi_i) = (0^{1\times3}, 0^{1\times3}, \phi_n \cdot 1^{1\times4})^T, \end{cases} \tag{20}$$

where the $\phi_n$ is the rotor rate at which the lift provided by the propellers allows the drone to hover in stable equilibrium. The cost $J(x_1^s, x_2^s)$ is learned as in Sec.V.B from 17016 paths. Fig. 9 shows the trajectories and energy map. The ground energy cost is found with Drivocopter parameters.

### C. Results

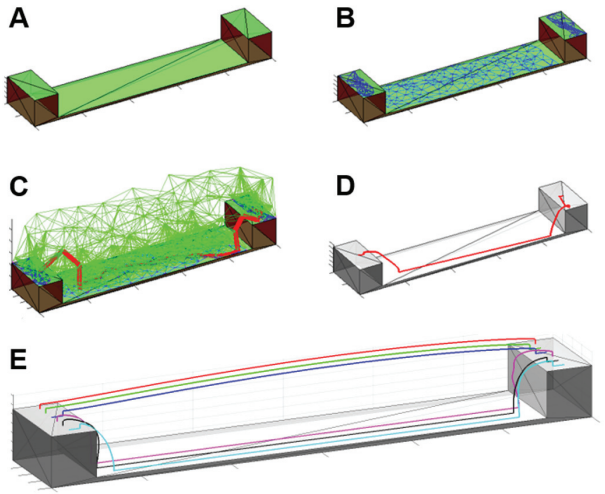


Fig. 10. A. Model Terrain classified into drivable and undrivable terrains. B. Poisson sampling on ground mesh. C. Poisson sampling on air and shortest path search. D. Smoothened final path. E. Heuristic trajectories for comparison in Tab.I. From back to front: F (red), DF (green), DFD (blue), FDF (pink), DFDF (black), DFDFD (cyan)

A CAD environment model, consisting of two raised platforms separated by a flat-bottom chasm, is meshed into drivable and undrivable regions (Fig.10.A), and the ground and free-space meshes are Poisson sampled (Fig.10.B). The result of a shortest-path (Fig.10.C) is smoothened (Fig.10.D). This process is depicted in Fig. 10. We hypothesized that when the landing platforms are nearby, the drone should not drive in the chasm, since gravitational losses exceed energy gains by driving. As the landing platform becomes more distant, the drone saves energy by driving in the chasm. We tested this idea on 5 different terrains parametrized by the distance between platforms (see Fig.8). Our planning results show correct qualitative behavior. Illustration of the trajectories is available in the video [34].

To illustrate how our results perform quantitatively, we also generate a few heuristic trajectories with different sequences (illustrated in Fig.10.E) and compare the final cost of these heuristic trajectories with our method in Table.I. Our method produces switching sequences that mostly agree with lowest-cost producing sequences among heuristic trajectories, and costs are quantitatively comparable to the heuristically optimal trajectories. However, our comparison is limited by the fact that the true optimal solution to the original mixed-integer problem is not tractable to obtain.

## VII. Conclusion

We presented a novel scheme to plan energy-efficient hybrid locomotion trajectories using approximate dynamic programming. Through capturing optimal policies within individual modalities with the optimal cost function, we showed that our approach is successful in decoupling the continuous and discrete optimization problems. We have also demonstrated that our approximated cost is successful in capturing complex dynamic characteristics of the robot through examples of practical hybrid locomotion: the hybrid double-integrator, the Ambot, and the Drivocopter.

Improvements are possible by upgrading elements of this framework. Better computational speed could be realized by adaptive sampling [35]. An (A*) [36] graph search would be enabled by transport energy heuristics, while other function approximators, such as Neural Nets, might improve the cost function learning module. More efficient implementations [37] can be used for trajectory optimization.

We also note some major limitations of the planner. Our

| | Cost from Fixed Sequence Heuristic Trajectories (Joules) | | | | | | Our Method | |
|---|---|---|---|---|---|---|---|---|
| Dist. (m) | F | DF | DFD | FDF | DFDF | DFDFD | Sequence | Cost (Joules) |
| 110 | 12129.72 | 11991.18 | 11780.26 | **6612.62** | 6620.16 | 6642.41 | DFDFD | 6655.44 |
| 90 | 10167.32 | 9902.18 | 9867.10 | **6345.64** | 6558.94 | 6578.74 | DFDFD | 6600.45 |
| 70 | 8208.72 | 7941.78 | 7726.73 | **6470.98** | 6492.85 | 6511.88 | DFD | 8156.85 |
| 50 | 6248.72 | 6112.78 | **5894.58** | 6477.60 | 6449.91 | 6452.32 | DF | 6456.37 |
| 30 | 4244.2 | 4148.18 | **3815.68** | 6343.64 | 6365.65 | 6387.47 | DFD | 4033.92 |

TABLE I

Comparison of costs from heuristic trajectories with fixed sequences. D: Driving, F: Flying

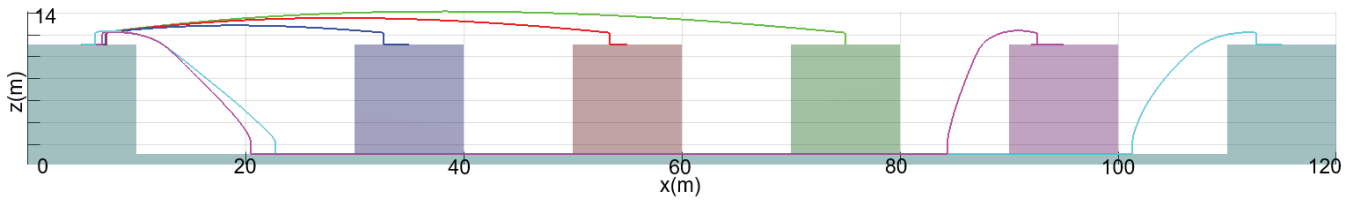


Fig. 8. Depiction of trajectory differences as the target platforms become more distant. At less than $75m$ separation, the robot always flies. After 95 meters separation, driving in the chasm saves energy.

framework relies on approximating solutions to Boundary Value Problems (BVP), but it is difficult to guarantee how well the function approximator captures the cost landscape, especially due to reachability constraints; even if the resulting BVP is infeasible, the cost approximator will still return a finite cost. An RRT [38] approach to planning through hybrid dynamical systems can address this issue [39]; while all nodes are reachable in this approach, detecting where the tree has crossed the guard surface is much more difficult.

For future works, we wish to understand how this planner can extend to settings with many obstacles, by using our sampled points as waypoints that can bound the result of final path smoothing. In addition, we are interested in studying cases where the map is not known in advance [40]. Finally, efforts are underway to demonstrate our results experimentally on the Drivocopter of Fig. 1.

## REFERENCES


[1] L. Cui, P. Cheong, R. Adams, and T. Johnson, "AmBot: A Bio-Inspired Amphibious Robot for Monitoring the Swan-Canning Estuary System," vol. 136, no. 11, 2014, pp. 115 001–115 001.

[2] G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades, "AQUA: An Amphibious Autonomous Robot," *Computer*, vol. 40, no. 1, pp. 46–53, Jan 2007.

[3] S. Mintchev and D. Floreano, "A multi-modal hovering and terrestrial robot with adaptive morphology," *Proc. 2nd Int. Symp. Aerial Robotics*, 2018.

[4] K. Peterson, "Hybrid Aerial and Terrestrial Locomotion, and Implications for Avian Flight Evolution," Ph.D. dissertation, University of California, Berkeley, 2013.

[5] Y. Chen, H. Wang, F. Helbling, N. T. Jafferis, R. Zufferey, A. Ong, K. Ma, N. Gravish, P. Chirarattananon, M. Kovac, and R. J. Wood, "A biologically inspired, flapping-wing, hybrid aerial-aquatic microrobot," *Science Robotics*, vol. 2, no. 11, 2017.

[6] N. Meiri and D. Zarrouk, "Flying STAR, a Hybrid Crawling and Flying Sprawl Tuned Robot," in *IEEE Int. Conf. Robotics & Auto.*, May 2019.

[7] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2016, pp. 1469–1475.

[8] T. Marcucci and R. Tedrake, "Mixed-integer Formulations for Optimal Control of Piecewise-affine Systems," in *Proc. 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control*, ser. HSCC '19. New York, NY, USA: ACM, 2019, pp. 230–239.

[9] B. Araki, J. Strang, S. Pohorecky, C. Qiu, T. Naegeli, and D. Rus, "Multi-robot path planning for a swarm of robots that can both fly and drive," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5575–5582.

[10] B. Flom, "Autonomous Path Planning for an Amphibious Vehicle," Naval Surface Warfare Center, Caderock Division, Tech. Rep., August 2009.

[11] A.Sharif, S.Choi, and H.Roth, "A New Algorithm for Autonomous Outdoor Navigation of Robots That Can Fly and Drive," in *Proc. 5th Int. Conf. Mechatronics and Robotics Engineering*, ser. ICMRE'19. New York, NY, USA: ACM, 2019, pp. 141–145.

[12] K. Hauser and J.-C. Latombe, *Multi-modal Motion Planning in Non-expansive Spaces*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 615–630.

[13] S. Choudhury and M. J. Kochenderfer, "Dynamic Real-time Multimodal Routing with Hierarchical Hybrid Planning," *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2397–2404, 2019.

[14] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE Int. Conf. Robotics and Automation*, May 2014, pp. 639–646.

[15] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE Int. Conf. on Robotics and Automation*, May 2011, pp. 1470–1477.

[16] M. Toussaint, "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning," in *Proc. 24th Int. Conf. on Artificial Intelligence*, 2015, p. 1930–1936.

[17] A. J. Coles, A. Coles, M. Fox, and D. Long, "COLIN: Planning with Continuous Linear Numeric Change," *CoRR*, vol. abs/1401.5857, 2014.

[18] E. Fernández-González, B. Williams, and E. Karpas, "ScottyActivity: Mixed Discrete-Continuous Planning with Convex Optimization," *J. Artificial Intelligence Research*, vol. 62, no. 1, p. 579–664, May 2018.

[19] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic Humanoid Locomotion: A Scalable Formulation for HZD Gait Optimization," *IEEE Trans. Robotics*, vol. 34, no. 2, pp. 370–387, April 2018.

[20] X. Xiong and A. D. Ames, "Orbit Characterization, Stabilization and Composition on 3D Underactuated Bipedal Walking via Hybrid Passive Linear Inverted Pendulum Model," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[21] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," vol. 1994. IEEE Transactions of Robotics, 1994.

[22] E. Dijkstra., "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec 1959.

[23] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Trans. Math. Softw.*, vol. 41, no. 1, Oct. 2014.

[24] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IEEE/RSJ Int. Conf. Intel. Robots Syst.s*, Sep. 2017, pp. 2872–2879.

[25] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Towards Search-based Motion Planning for Micro Aerial Vehicles," *ArXiv*, 2018.

[26] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *IEEE Int. Conf. Robotics Automation*, May 2011.

[27] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support Vector Regression Machines," *Advances Neural Info. Processing Syst.*, vol. 9, pp. 155–161, July 1997.

[28] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.

[29] R. Bridson, "Fast Poisson Disk Sampling in Arbitrary Dimensions," in *SIGGRAPH*, August 2007.

[30] R. Fan, P. Chen, and C. Lin, "Working Set Selection Using Second Order Information for Training Support Vector Machines," *J. Mach. Learn. Res.*, vol. 6, pp. 1889–1918, Dec. 2005.

[31] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *IEEE Int. Conf. on Robotics and Automation*, May 2016, pp. 3261–3266.

[32] F. Morbidi, R. Cano, and D. Lara, "Minimum-Energy Path Generation for a Quadrotor UAV," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017.

[33] H. T. Suh, "End-to-End Full-State Dynamics of Generic Rotorcrafts," http://hjrobotics.net/wp-content/uploads/2018/10/End-to-End-Full-State-Dynamics-of-Generic-Rotorcrafts.pdf, 2018.

[34] "https://www.youtube.com/watch?v=HMNLue8lduw."

[35] D. Hsu, "Randomized Single-Query Motion Planning in Expansive Spaces," Ph.D. dissertation, Stanford University, May 2000.

[36] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, no. 2, July 1968.

[37] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *Int. J. Control*, pp. 1–17, 2017.

[38] S. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.

[39] A. Wu, S. Sadraddini, and R. Tedrake, "R3T: Rapidly-exploring Random Reachable Set Tree for Optimal Kinodynamic Planning of Nonlinear Hybrid Systems," *International Conference on Robotics and Automation (To Appear)*, 2020.

[40] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative Temporal Motion Planning for Hybrid Systems in Partially Unknown Environments," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 353–362.