Optimal 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, can enable robots to carry out complex tasks in diverse environments. This paper presents a novel method of combining graph search and trajectory optimization for planning multi-modal locomotion trajectories. We also introduce methods that allow the method to work tractably in higher dimensional state spaces. Through the examples of a hybrid double-integrator, amphibious robot, and the flying-driving drone, we show that our planner tractably gives full-state trajectories that are probabilistically optimal and dynamically feasible.**

I. Introduction

A hybrid locomotor combines multiple movement modalities into a single platform. Examples of hybrid locomotion (HL) 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 improve energy-efficiency. Prior works on hybrid locomotion [1], [2], [3], [4], [5], [6], [7] have often investigated the design and feasibility of hybrid locomotion strategies, and examples of such robots are given in Fig. 1.

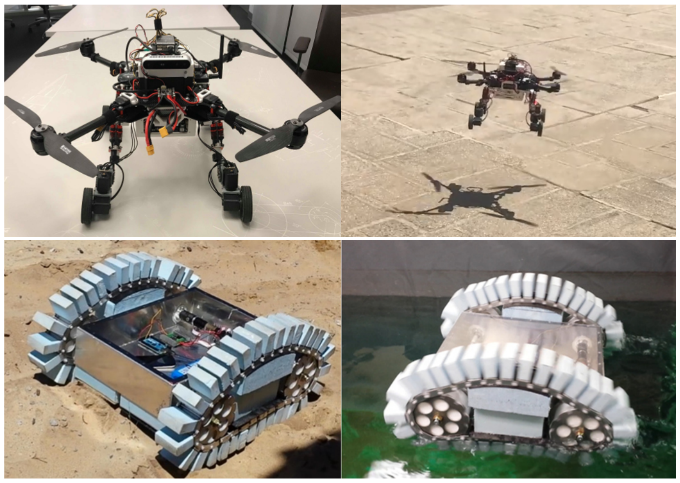

Fig. 1. Top Row: A "Drivocopter" Drone (developed by the authors) which can fly and drive. Video of operation can be accessed at https://www.youtube.com/watch?v=QZyuvXfifvQ. Bottom Row: "Ambot" Amphibious Robot [1] capable of ground and marine locomotion.

[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

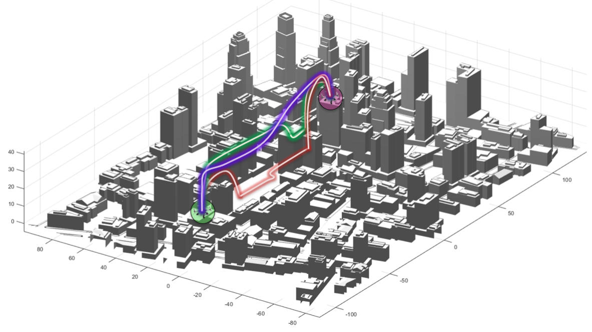

Fig. 2. Different paths that a flying-driving robot could take between building rooftops. The vehicle flies along the purple trajectory, drives as much as possible along the red trajectory, and alternates between flying and driving on the green trajectory.

However, realizing the full potential of these robots not only depends on clever design, but also on autonomous planning of their complex motion strategies. For instance, Fig. 2 illustrates different paths that a flying-driving robot (such as the Drivocopter of Fig. 1) could use to traverse between two building rooftops. The different paths have entirely different energy-costs, travel times, and robustness, which ultimately dictate robot performance in this task.

Unlike conventional motion planning problems, the path cost depends not only on the optimality of the trajectory segments in each modality, but also on the switching sequence, time, and state coordinates. The problem of achieving combinatorial optimization of the switching sequences, as well as optimization of trajectories within each modality, makes this problem particularly challenging, as illustrated by the different paths in Fig. 2.

Many existing motion planning strategies cannot directly address the difficulty of multi-modal planning. Graph-based motion planning approaches (PRM [8], RRT [9], RRT* [10]) excel at discrete optimization in sampled coordinates, but suffer from the curse of dimensionality. They are unable to produce full-state trajectories for high-dimensional systems and do not readily incorporate costs that are not functions of sampled coordinates.

Motion planning algorithms based on trajectory optimization (e.g., via direct collocation [11], [12]) either assume smooth dynamics, or use multi-phase optimization with pre-specified domain sequences [13]. This is due to the fact that standard nonlinear programming frameworks cannot handle the combinatorial optimization of discrete locomotion mode switches. Mixed-Integer Programming methods [14], [15] transcribe these problems well, but do not scale well enough to handle switching sequences and coordinates of realistic high-dimensional problems.

Existing works in multi-modal planning often bypass this problem by only considering discrete graph-based planning ([16],[17],[18]). By ignoring the continuous dynamics of the robot, these planners often ignore dynamic feasibility in a single modality, and how the dynamic constraints affect the cost. In addition, although energy expenditure is often the most important cost in hybrid locomotion, direct calculations of this cost from change in position is not possible without making vast simplifications, such as using Cost of Transport (COT) to linearly map distance to energy ([16],[18]).

To address this problem, we present a novel motion-planning method for hybrid locomotion that considers the problem as a graph-search with local trajectory optimization on the continuous dynamics between connected nodes. In low-dimensions, we propose a graph construction strategy on the modality-partitioned state-space such that no edge crosses the guard between partitions. The cost of traveling between sample points is found using optimal trajectory generation for the corresponding modality, and a graph-search determines the nearly optimal state-space path. To make the method tractable in higher dimensional state spaces, we also propose constraining some subspace of the state-space to be a function of sampled states via virtual constraints, and learning the cost function from offline trajectory optimization batches.

To the best of our knowledge, we present the first motion planner for multi-modal locomotion that considers direct representation of the energy cost as well as dynamic constraints, while retaining probabilistic optimality. 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).

II. Problem Formulation

A. The Hybrid Locomotion System

We define a hybrid locomotion system as a type of hybrid control system [13] with additional constraints. We define the hybrid locomotion system $\mathscr{HL}$ as a tuple

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

In the below description of each element, i denotes the index of the locomotion mode (i.e. flying or driving):

- $FG = \{(f_i, g_i)\}$ describes the dynamics associated with each locomotion mode. The dynamics 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:

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

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.

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

In words, 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.

III. Planning Methodology

Section III-A reviews our basic planning concept that combines sampling-based planning with local trajectory optimization. Because this approach may not scale well to practical situations, Section III-B introduces virtual constraints and off-line learning to improve real-time performance.

A. Dynamic Programming with Continuous Optimization

1) Graph Structure: First, we discretize the problem by sampling coordinates in each domain $\mathcal{D}_i$. The vertices, V, of a digraph, $G(V, E)$, are constructed from these samples. 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 locomotion modalities, we additionally impose the following constraints on the graph:

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.3. A and B illustrate these conditions. The shortest-path search is tackled by Djikstra's algorithm [19] once the locally optimal trajectory costs are known.

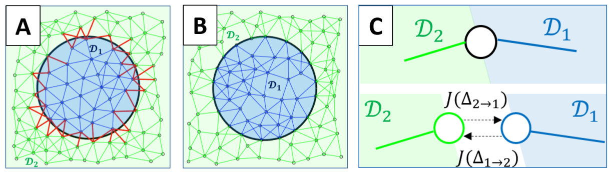

Fig. 3. A) red edges cross a guard surface between domains $\mathcal{D}_1$ and $\mathcal{D}_2$, violating the constraint on edges. B) By sampling on the guard surface and allowing no edges between $\mathcal{D}_1$ to $\mathcal{D}_2$, all graph edges are constrained to a single mode. C) Node augmentation to handle modality switching costs.

If there exists a switching cost to go from one modality to another, we augment the sample on the guard surface x with two connected nodes x_1 and x_2 that shares the same state-space coordinates, and assign switching cost to the edge cost between the two samples, as illustrated in Fig. 3.C.

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

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

where $x_1, x_2 \in \mathcal{D}_i$. This standard trajectory optimization problem can be tackled using existing methods, such as direct collocation [12].

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 state space samples using their modalities, such that

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

where j_i denotes the mode of each partition, and x_{k_i} denotes the sample on the guard surface where the trajectory crosses during a mode switch. The optimal trajectories between boundary points is then found by solving the following optimization problem for each partition:

$$\begin{aligned} & \min_{u} \quad J_{j_i} \\ & \text{s.t.} \quad \dot{x} = f_i(x) + g_i(x)u,\ \ x \in \mathcal{D}_{j_i},\ \ u \in \mathcal{U}_{j_i}, \\ & \qquad\ \ x(t_{k_i}) = x_{k_i}, \quad x(t_{k_{i+1}}) = x_{k_{i+1}} \end{aligned}$$

The total trajectory is reconstructed by concatenating the trajectories $x^* = (x_1^*, x_2^*, \cdots, x_n)^*$ induced by the optimal control inputs u_i^* for each partition.

B. Extension to High Dimensions

Although dynamic programming with segment-wise trajectory optimization shows good promise for hybrid locomotion, 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) Reduced-Order Coordinates via Virtual Constraints: We can reduce the dimensionality of the sample space by introducing virtual constraints that fix some coordinates as a function of the sampled-coordinates. The state-space is divided into sampled coordinates (x^s) and auxiliary coordinates (x^a). The full state is recovered from samples in the reduced space, x^s, by appending auxiliary coordinates:

$$x = (x^s, x^a)^T = (x^s, v(x^s))^T \tag{1}$$

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

2) Learning the cost function from offline optimization: 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:

$$\begin{aligned} J(x_1^s, x_2^s) = \ & \min_{u} \quad J_i \\ & \text{s.t.} \quad \dot{x} = f_i(x) + g_i(x)u,\ \ x \in \mathcal{D}_i,\ \ u \in \mathcal{U}_i, \\ & \qquad\ \ x_0 = \begin{pmatrix} x_1^s \\ v(x_1^s) \end{pmatrix}, x(t_f) = \begin{pmatrix} x_2^s \\ v(x_2^s) \end{pmatrix} \end{aligned} \tag{2}$$

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 this function off-line for faster evaluation online.

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 $((x_1^s, x_2^s), J(x_1^s, x_2^s))$ from multiple trajectory optimization runs. Then, function approximators from supervised learning algorithms such as Support Vector Regression (SVR) [21] 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 from data, its evaluation does not require a full instance of nonlinear programming, greatly reducing on-line computation. Yet, as $\tilde{J}$ is learned from trajectory optimization, all costs in Bolza form can be utilized, and dynamic 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. While the rail is lubricated and frictionless at $p < 0$, viscous drag appears $p \ge 0$. This can be formulated as a simple hybrid locomotion system with the following dynamics:

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

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

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

where the dynamics are described by

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

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

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

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 [12] with IPOPT [22] 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.4.

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.4.

In addition, to show probabilistic convergence, we run the algorithm 10 times with different inter-sample distances (controlled by Poisson sampling [23] on state-space), and show convergence in Fig. 5. Fig. 5 shows that our method probabilistically converges, with a lower cost compared to multi-phase optimization using GPOPS-II with IPOPT. IPOPT is only local optimal, while a PRM framework searches more globally over the domain. Our final cost shows that we can produce probabilistically optimal and dynamically feasible trajectories with inter-sample distance as large as 0.3.

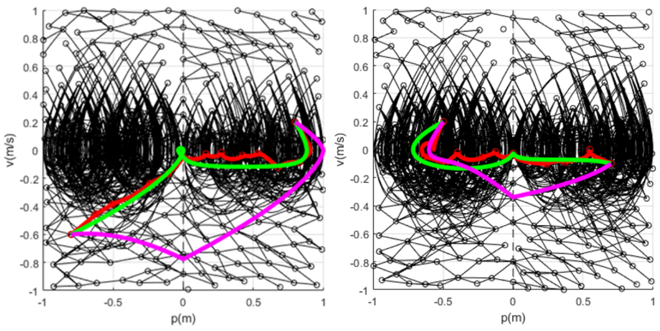

Fig. 4. Optimal Trajectories from $x_i = (0.8, 0.2)^T$ to $x_f = (-0.8, -0.6)^T$ (left), and from $x_i = (0.7, -0, 1)^T$ to $x_f = (-0.5, 0.2)^T$ (right). Red trajectories are obtained using graph search, green trajectories are results of final smoothened path, and the pink trajectory is result of Multi-phase optimization in GPOPS-II. The edges represent optimal trajectories between each sample.

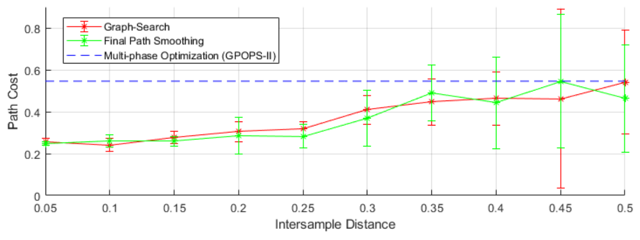

Fig. 5. Probabilistic convergence with decreasing intersample distance.

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 a model environment.

A. Dynamics

1) Ground and Marine Dynamics: We derive the Newtonian mechanics for planar operation, 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

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

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

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

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:

$$J_g = J_m = \int_{t_i}^{t_f} \left[\sum_{i=L,R} V_{cc} u_i \cdot \frac{k_t}{R}(V_{cc}u_i - k_t\phi_i) + P_d \right] dt \tag{3}$$

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 motor power dissipation, and the latter term models constant power drainage. We assume no switching costs associated with the discrete reset map, $J(\Delta_{m\to g}) = J(\Delta_{g\to m}) = 0$

B. Cost Learning

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

$$x_g^s = (p_x^w, p_y^w, v_x^b, \theta)^T \quad , \quad x_g^a = (v_y^b, \omega)^T = (0, 0)^T \tag{4}$$

This division of coordinates recognizes that side-slip is constrained for skid-steer vehicles, and angular velocity is

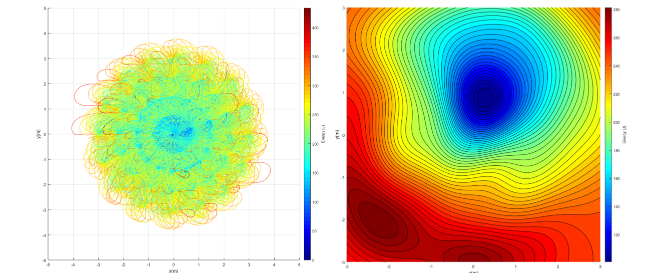

Fig. 6. Left: 11520 ground trajectories colored by their cost. Right: xy-energy contour for $v_x^b = 1.0m/s$, $\theta = \pi/2$. The heatmap corresponds to the energy cost to go from $x_i^s = [0,0,0,0]^T$ to $x_f^s = [x, y, 1.0, \pi/2]^T$

small. For marine operation, $x_m \in \mathbb{R}^8$ is divided into

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

where track forces equal water drag at equilibrium speed ϕ_n.

The cost function $J(x_i^s, x_f^s)$ in Eq. (2) is learned from multiple offline optimizations. Using 11520 samples, the function $J(x_i^s, x_f^s)$ is evaluated using GPOPS-II [12], and SVR with Gaussian kernel trains the function $\tilde{J}$ with Sequential Minimal Optimization [24]. The process is repeated for both ground and marine locomotion. Fig.6 shows the generated trajectories and the contour of the learned function.

C. Results

We sample position using the method of Sec.III.A, and grid the states v_x, θ 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 [19]. Fig. 7 illustrates this process. The final smoothed trajectory is shown in Fig.8.

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 can autonomously decide switching sequences and switching points.

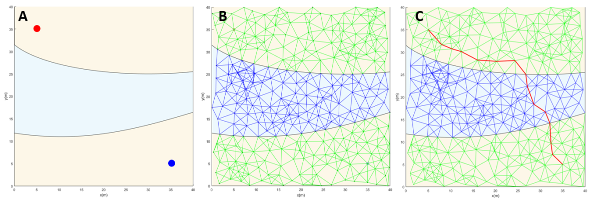

Fig. 7. A: Model Environment. B: Graph Generation. C. Result of shortest path search.

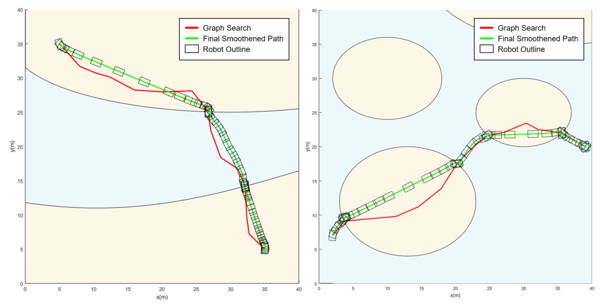

Fig. 8. Final trajectories for example of river crossing (left), and island crossing (right). The Robot outline is displayed at equal time differences.

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 [25] and [26].

1) Flight Dynamics: Standard rigid-body dynamics [27] describe flight motions driven by four rotor forces, which use a speed-squared-dependent lift term and 1st-order armature motor dynamics. The state vector $x_f \in \mathbb{R}^{16}$ is

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

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 wrt 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

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

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 knowledge of the ground surface. 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:

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

where ϕ_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 Eq.(3), 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. Cost Learning

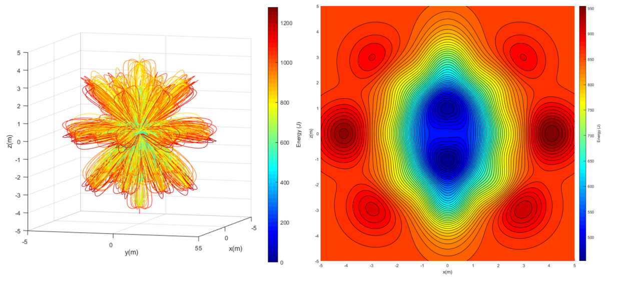

Fig. 10. Left: 17016 trajectories produced to learn the flight energy function. Right: xz-projection of the learned function $\tilde{J}(0, x)$.

The ground states are divided into sampled / auxiliary coordinates via Eq. (4). Flight states are divided by:

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

where the ϕ_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. 10 shows the trajectories and energy map. The ground energy cost is found with Drivocopter parameters.

C. Results

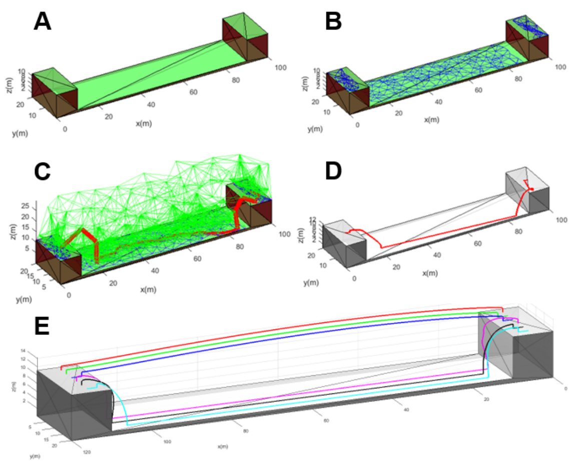

Fig. 11. A. Model Terrain classified into drivable and undrivable terrains. B. Poisson sampling on ground mesh. C. Poisson sampling on air and shortest path search. D. Smoothened final path. E. Heuristic trajectories for comparison in Tab.I. From back to front: F (red), DF (green), DFD (blue), FDF (pink), 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.11.A), and the ground and free-space meshes are Poisson sampled (Fig.11.B). The result of a shortest-path (Fig.11.C) is smoothened (Fig.11.D). This process is depicted in Fig. 11. We hypothesized that when the platforms are nearby, the drone should not drive in the chasm, since gravitational losses exceed energy gains from by driving. As the platforms separate further, 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.9). Our planning results show correct qualitative behavior.

To show quantitative competence, we also generate few heuristic trajectories per given fixed sequence (illustrated in Fig.11.E) and tabulate the final costs in Table.I. Our method produces a switching sequence that mostly agrees with lowest-cost producing sequences among heuristic trajectories, and costs are quantitatively comparable to the heuristically optimal trajectories.

VII. Conclusion

We presented a novel scheme to plan near-optimal hybrid locomotion trajectories. A double-integrator example showed that our method can generate probabilistically optimal and dynamically feasible trajectories in low-dimensional state-spaces. The Ambot and Drivocopter examples showed that virtual constraints and cost function learning renders our method practical in high-dimensional problems.

Improvements are possible by upgrading components of our framework. Better computational speed could be realized by adaptive sampling [28] and the use of RRT [9] search to achieve faster single-query tractability. An (A*) [29] graph search would be enabled by transport energy heuristics, while other function approximations, such as Neural Nets, might improve the cost function learning module. Differential Dynamic Programming [30] is another promising scheme for planning trajectory segments.

While we have demonstrated probabilistic optimality in a low-dimensional example, we acknowledge the lack of provable optimality in high dimensions. A better understanding and justification of virtual constraints should result from studying the bisimilarity between the full-state and reduced-order systems. Finally, efforts are underway to demonstrate our results on the Drivocopter of Fig. 1.

	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 our costs from hueristic trajectories with fixed sequences. D: Driving, F: Flying

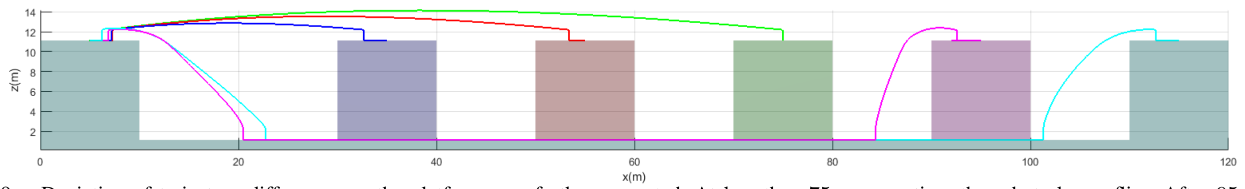

Fig. 9. Depiction of trajectory differences as the platforms are further separated. At less than $75m$ separation, the robot always flies. After 95 meters separation, driving in the chasm saves energy.

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] Y. Mulgaonkar, B. Araki, J. Koh, L. Guerrero-Bonilla, D. M. Aukes, A. Makineni, M. T. Tolley, D. Rus, R. J. Wood, and V. Kumar, "The picobug: A mesoscale robot that can run, fly, and grasp," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[4] S. Mintchev and D. Floreano, "A multi-modal hovering and terrestrial robot with adaptive morphology," *Proceedings of the 2nd International Symposium on Aerial Robotics*, 2018.

[5] K. Peterson, "Hybrid aerial and terrestrial locomotion, and implications for avian flight evolution," Ph.D. dissertation, University of California, Berkeley, 2013.

[6] 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.

[7] N. Meiri and D. Zarrouk, "Flying star, a hybrid crawling and flying sprawl tuned robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[8] 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.

[9] S. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[10] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," 2006.

[11] S. L. Campbell, "Practical methods for optimal control using nonlinear programming, john t. betts, siam, philadelphia, pa, 2001, isbn 0-89871-488-5," *International Journal of Robust and Nonlinear Control*, vol. 14, no. 11, pp. 1019–1021, 2004.

[12] 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, pp. 1:1–1:37, Oct. 2014.

[13] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, April 2018.

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

[15] T. Marcucci and R. Tedrake, "Mixed-integer formulations for optimal control of piecewise-affine systems," in *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '19. New York, NY, USA: ACM, 2019, pp. 230–239.

[16] 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.

[17] B. Flom, "Autonomous path planning for an amphibious vehicle," Naval Surface Warfare Center, Caderock Division, Tech. Rep., August 2009.

[18] S. A.Sharif and H.Roth, "A new algorithm for autonomous outdoor navigation of robots that can fly and drive," in *Proceedings of the 5th International Conference on Mechatronics and Robotics Engineering*, ser. ICMRE'19. New York, NY, USA: ACM, 2019, pp. 141–145.

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

[20] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, May 2011.

[21] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advanced in Neural Information Processing Systems*, vol. 9, pp. 155–161, July 1997.

[22] 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.

[23] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," in *SIGGRAPH*, August 2007.

[24] 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.

[25] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3261–3266.

[26] F. Morbidi, R. Cano, and D. Lara, "Minimum-energy path generation for a quadrotor uav," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[27] H. T. Suh, "End-to-end full-state dynamics of generic rotorcrafts," 2018.

[28] D. Hsu, "Randomized single-query motion planning in expansive spaces," Ph.D. dissertation, Stanford University, May 2000.

[29] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[30] D. Jacobson and D. Mayne, *Differential Dynamic Programming*, ser. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1970.