AUTONOMOUS VEHICLES

Monte Carlo tree search with spectral expansion for planning with dynamical systems

Benjamin Rivière*†, John Lathrop*†, Soon-Jo Chung*

The ability of a robot to plan complex behaviors with real-time computation, rather than adhering to predesigned or offline-learned routines, alleviates the need for specialized algorithms or training for each problem instance. Monte Carlo tree search is a powerful planning algorithm that strategically explores simulated future possibilities, but it requires a discrete problem representation that is irreconcilable with the continuous dynamics of the physical world. We present Spectral Expansion Tree Search (SETS), a real-time, tree-based planner that uses the spectrum of the locally linearized system to construct a low-complexity and approximately equivalent discrete representation of the continuous world. We prove that SETS converges to a bound of the globally optimal solution for continuous, deterministic, and differentiable Markov decision processes, a broad class of problems that includes underactuated nonlinear dynamics, nonconvex reward functions, and unstructured environments. We experimentally validated SETS on drone, spacecraft, and ground vehicle robots and one numerical experiment, each of which is not directly solvable with existing methods. We successfully show that SETS automatically discovers a diverse set of optimal behaviors and motion trajectories in real time.

INTRODUCTION

Endowing robots with high-performing and reliable autonomous decision-making is the ultimate goal of robotics research and will enable applications such as sea, air, and space autonomous exploration; self-driving cars; and urban air mobility. These autonomous robots need to make decisions encompassing low-level physical movements (i.e., motion planning) and high-level strategy, such as selecting goals, sequencing actions, and optimizing other decision variables.

This vision of robotic autonomy remains elusive because solving the decision-making problem exactly in high-dimensional continuous-space systems has "the curse of dimensionality" (*1*). In light of this complexity, many autonomous robots in deployment avoid a general problem formulation and instead exploit a particular problem structure for computational benefits. For example, motion planning can be solved with sampling-based methods (*2*–*4*), trajectory optimization can be solved with convex optimization (*5*, *6*), and high-level discrete decision-making can be solved with value iteration (*7*). These methods can also be combined hierarchically for complex behavior, such as autonomous multi-spacecraft on-orbit inspection (*8*), robotic manipulation (*9*, *10*), and self-driving urban vehicles (*11*, *12*). Problem-specific solutions are well understood, can be made computationally efficient, and have proven to be effective. However, these solutions are limited because they cannot be easily adapted to new problems, adding an expanding burden to the designer for multitask autonomy. In addition, there exist problems that cannot easily be decomposed into computationally tractable subproblems.

Reinforcement learning is an alternative approach that uses trajectory data to train optimal policies. Unlike the aforementioned methods, this approach is a generalized procedure that can be applied directly to a broader class of problems. This property enables important new capabilities such as drone racing (*13*), helicopter flight (*14*), grasping (*15*), and bipedal locomotion (*16*). However, these methods usually require an offline training phase, and therefore their operation is fundamentally limited in new or changing environments. In addition, these black-box approaches are unexplainable and provide limited guarantees on optimality, stability, or robustness.

Some planning and model-based reinforcement learning approaches use tree-based algorithms (*17*–*19*) that strategically explore simulated future trajectories from the current state using a family of methods collectively known as Monte Carlo tree search (MCTS) (*20*). In contrast with offline learning methods, MCTS generates high-quality solutions using real-time computation: Given a system's dynamics and a reward function, the goal is to return the best possible plan with the computational budget available. However, whereas the tree's nodes and edges are naturally defined for discrete spaces, the continuous space of robots presents new challenges. Uniform spatial and temporal discretization of continuous spaces leads to very large trees and slow convergence rates: a poor discrete representation of the underlying continuous problem.

In this work, we present Spectral Expansion Tree Search (SETS), a real-time and continuous space planning algorithm that converges to globally optimal solutions. The new capability of SETS is enabled by efficiently representing continuous dynamical systems with the system's natural motions, a concept formalized through the spectrum of the locally linearized controllability Gramian, a well-established concept in linear systems theory. Without further assumptions on the dynamics or reward, the transformed low-complexity form is solved with MCTS. The SETS tree is visualized in Fig. 1A. Compared with a direct discretization, our method reduces the branching factor and the number of decisions in the time horizon, leading to an exponential reduction in tree size. Our theoretical contribution is twofold: We show that our algorithm discretizes the continuous space problem while preserving its optimal solution, and we provide a new finite-time convergence analysis of MCTS in our setting. This analysis proves fast convergence to a bound of the globally optimal solution for planning problems with deterministic and differentiable

Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA.
*Corresponding author. Email: briviere@caltech.edu (B.R.); jlathrop@caltech.edu (J.L.); sjchung@caltech.edu (S.-J.C.)
†These authors contributed equally to this work.

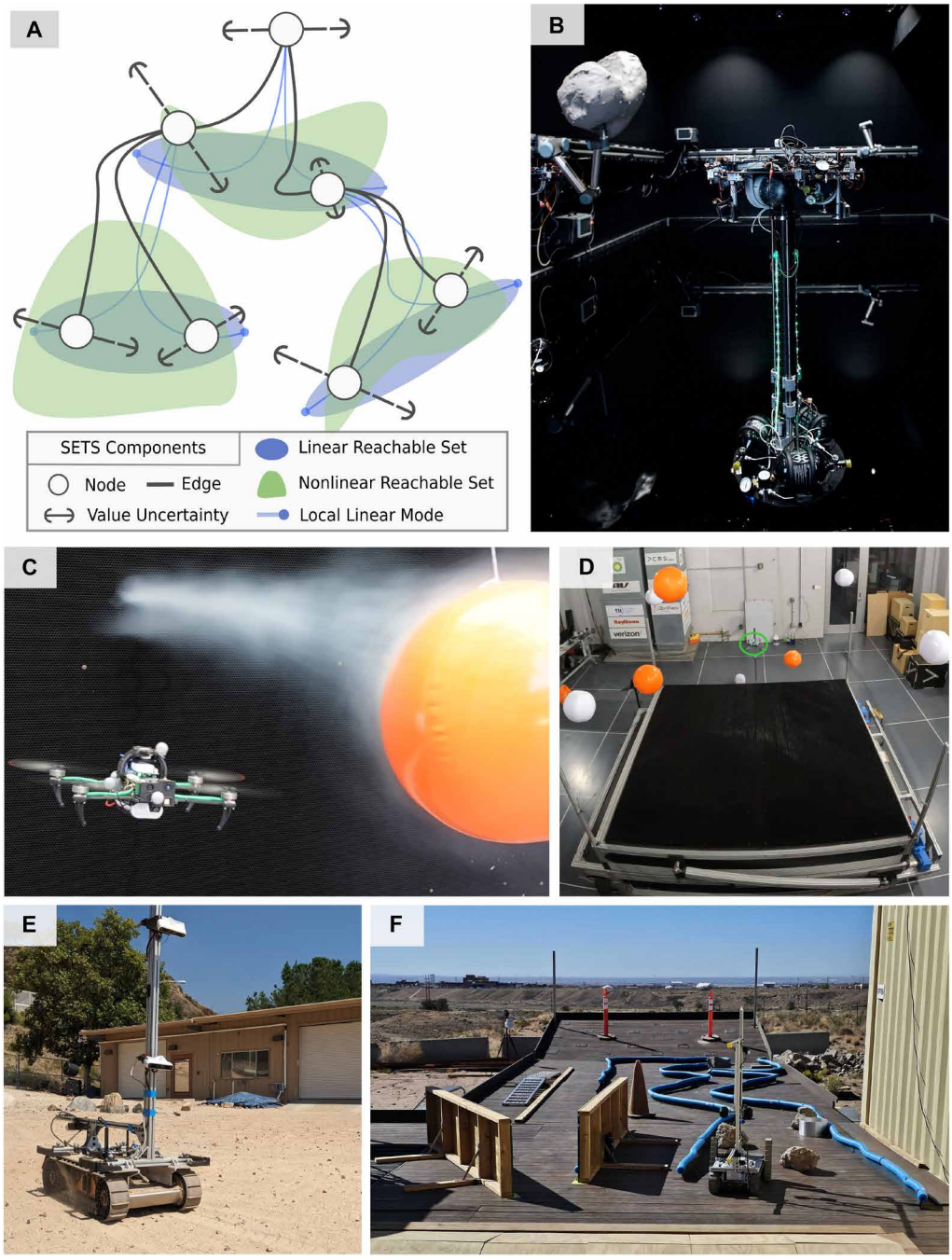


Fig. 1. Overview of method and experiments. (**A**) Our method, SETS, is a new tree-based planning algorithm for dynamical systems. The tree's edges (shown in gray) are constructed by tracking the spectral modes of the local linearization (shown in blue) with nonlinear feedback control. (**B** to **F**) We demonstrate that SETS is widely applicable in robotic domains, spanning ground, aerial, and space domains.

dynamics, state-dependent and Lipschitz rewards, and continuous state-action spaces.

Our hardware and simulation experiments showcase a diverse set of "discovered-not-designed" behaviors generated in real time for various robot dynamics and objectives: First, a quadrotor solved a dynamically constrained traveling salesman problem to quickly monitor multiple targets in a windy arena, selectively traversing wind gusts and avoiding floating ball obstacles; second, a tracked vehicle shared control with a driver through a concourse of ramps, chicane, and sawtooth tracks subject to adversarial degradation; third, a team of spacecraft used a net to capture and redirect an uncooperative target in a frictionless environment; and fourth, in simulation, a glider experiencing aerodynamic drag detoured into a thermal to extract energy from the environment and survived long

enough to achieve its directed task. We used this last experiment as a case study to empirically validate the theoretical result, compare our method with state-of-the-art baselines, and tune parameters in a systematic and informed procedure.

RESULTS

We present results validating SETS in four experiments, three on different robotic platforms and one in simulation. The breadth of our experiments highlights the general applicability of SETS to many dynamical systems. We compare against state-of-the-art baselines in simulation and analyze the empirical results of our theoretical guarantees.

Quadrotor navigates a dangerous wind field

Our first experiment used SETS to plan trajectories for a quadrotor to monitor multiple targets in a cluttered three-dimensional (3D) environment of dangerous wind drafts and moving obstacles and is shown in Fig. 2. The experimental arena, shown in Fig. 2A, is a cube with 3-m side lengths where the Real Weather Wind Tunnel in Caltech's Center for Autonomous Systems and Technologies generates controlled columns of air to suspend and move spherical obstacles and observation targets. In addition to the physical obstacles, there exist dangerous and benign regions of flow of varying speed that affect the planning problem. A video of the experiment is presented in Movie 1.

This experiment was designed to test the ability of SETS to quickly plan in high-dimensional space subject to the quadrotor dynamics and external forces from aerodynamic interactions. In particular, the dynamic model used by SETS is a standard quadrotor model augmented with a deep neural network (DNN) to model the learned residual wind force. In addition, the algorithm must run in real time to correct model error and to react to new information. This problem is challenging for existing solutions because the complex interaction among varying wind strengths, dynamics, and obstacles determines path feasibility. Therefore, the problem is not decomposable into position path planning then tracking control. Furthermore, an indicator reward function, a dense obstacle configuration, and multiple goal regions make it challenging to accurately model with conventional motion planning or optimization-based frameworks. Instead, the algorithm must find a global solution by searching through complex dynamics.

SETS is the planning module of the autonomy stack, generating trajectories of 10-s duration every 5 s and running in a model predictive control manner. SETS runs in real time by fixing the computation time, planning from the predicted future state, and closing the loop with feedback control. In Fig. 2A, the search tree is visualized by projecting the 12D state trajectories onto the 2D surface of the fan array, and the branches are colored by when they were expanded, where purple indicates the first trajectories in the tree and yellow indicates the last trajectories in the tree. We can see that the trajectories concentrate at the origin and at each of the target locations.

The SETS tree was constructed with the spectrum of the locally linearized controllability Gramian, shown in Fig. 2B. Each column of the spectrum plot is a natural motion ("mode") of the locally linearized system, and each mode was used to create a branch in the tree search algorithm. The modes in still air are interpretable: The first mode of the spectrum corresponds to accelerating the vertical velocity v_z, and the next two modes correspond to a pitch and roll maneuver, φ and θ, respectively. Although the dynamics in the thermal are more complex, the modes capture important dynamical

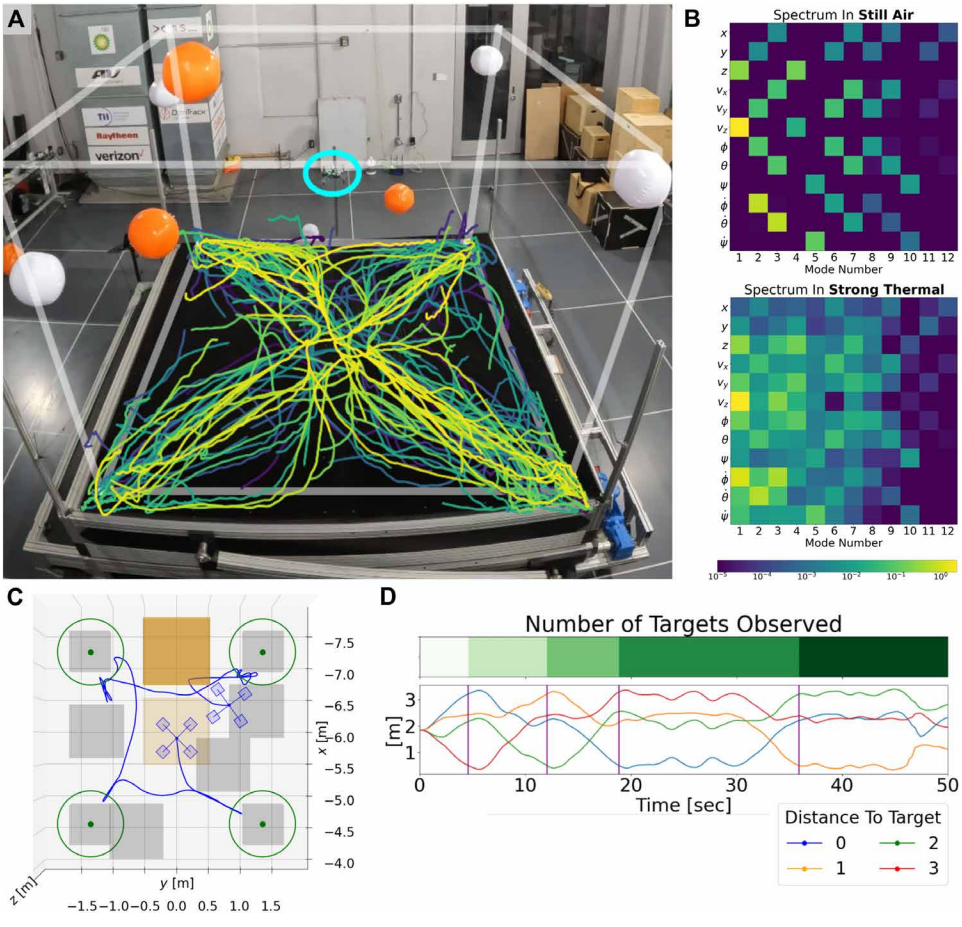
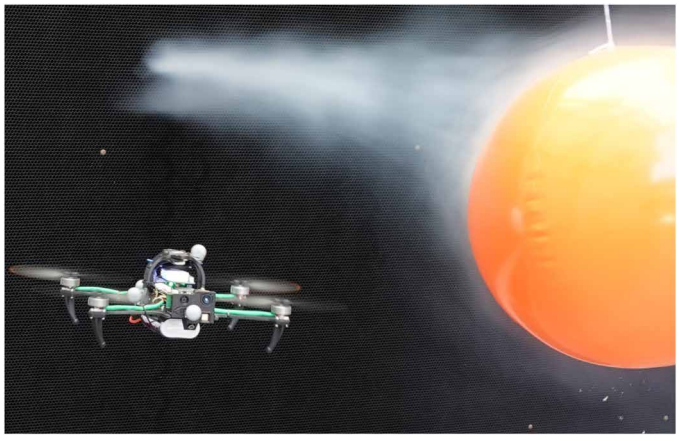


Fig. 2. Quadrotor experiment. (**A**) SETS enables a drone, circled in blue, to plan trajectories to multiple targets (white) over a fan array and obstacles (orange) in real time. The 12D search tree is projected onto the 2D fan surface. The branches are colored by the order of expansion, with yellow indicating later trajectories. (**B**) The spectrum of the controllability Gramian is shown for flying in still air and flying through a thermal. Each column corresponds to a natural motion of the system, each row corresponds to a dimension of the state, and each cell is colored by the magnitude of its controllability. (**C**) A top-down view of the final trajectory, where the targets are shown in green, the thermals in orange, and the obstacles in gray. The thermals are shaded by their relative strength. (**D**) The distance to each target over time. The mission objective is to visit all four targets and is completed after 37 s.



Movie 1. Overview of SETS for planning with dynamical systems.

information. For example, it becomes much harder for the quadrotor to excite the yaw, ψ, degree of freedom independently from the other degrees of freedom.

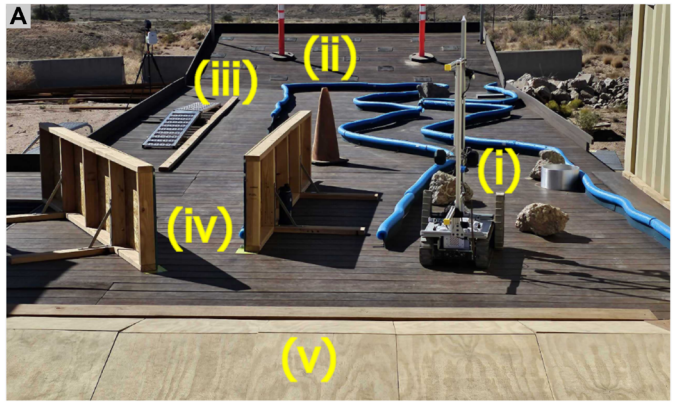
The resulting trajectory from this experiment is shown in Fig. 2C, where we see that the quadrotor visits all the targets while avoiding obstacles and dangerous winds. To observe the target, the quadrotor has to be within a sensing radius of 50 cm of the center of the target (visualized in Fig. 2C). The physical size of the target creates an obstacle with a 30-cm radius, leaving a feasible viewing volume of 0.3 m^3 for each target. SETS's ability to quickly find trajectories to these small regions of the position state space while accounting for fast attitude and aggressive wind dynamics demonstrates a sophisticated balance of precision, exploration, and stability.

The overall mission progress can be described by the distance to each target over time. This metric and the cumulative number of targets observed are shown in Fig. 2D. The quadrotor visits all of the targets in 37 s, with the final transition through the narrow corridor of thermals taking the longest time to find a solution. The solution's nature is reminiscent of the classical traveling salesman problem (*21*); here, SETS discovers the dynamic-dependent cost of traveling between targets and the optimal path to visit them all. Discovering this solution automatically, rather than prescribing a sequence of waypoints, has two advantages: First, the burden on the designer to enumerate and integrate many behaviors is relieved, extending the operational envelope of the robot, and second, SETS may solve problems beyond the intuition of the designer. For example, it is not clear at what fan strength and exposure duration a wind gust becomes too dangerous to cross safely.

Tracked vehicle with human in the loop

Our second experiment, shown in Fig. 3, used SETS for driver assistance of a tracked vehicle subject to antagonistic terrain effects, tipping-over constraints, obstacle avoidance, and actuator degradation. This human-in-the-loop configuration is also called "parallel autonomy" in the self-driving car literature (*12*). These experiments were developed during the Defense Advanced Research Projects Agency (DARPA) Learning Introspective Control (LINC) research program, with various versions of the algorithm tested at the Sandia National Laboratory Robotic Vehicle Range. A video of this experiment is presented in Movie 1 and movies S1 and S2.

In the experiment, the tracked vehicle was tasked with assisting a driver to traverse a 20 m–by–10 m test circuit with slippery slopes, a variety of obstacles, and a thin chicane track, shown in Fig. 3 (A and C). In Fig. 3D, we zoom in on one section of the chicane track, in



B

Track Section	Chicane Track	Inclined Plane	Single-Ramp Incline	Narrowing Corridor	Sawtooth Track	Total
Safety Violation Count (driver alone)	2	0	1	3	0	6
Safety Violation Count (with SETS)	**0**	0	**0**	**0***	0	**0**

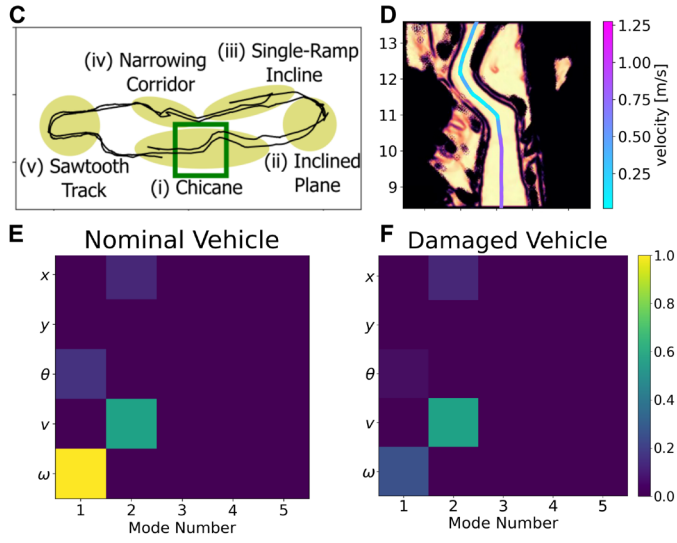


Fig. 3. Tracked vehicle experiment. (**A**) The five sections of the test circuit are indicated. (**B**) We counted safety violations of a human driver versus a human driver sharing autonomy with SETS for a selected run through the test course, where a safety violation is a collision with obstacles or a vehicle tip-over. *Because of human error, the autonomy stack was disabled for a few seconds, at which point a collision occurred. (**C**) The trajectory of our vehicle as it completes two loops of the track. Each section of the track is highlighted in yellow. We boxed in green a segment of the chicane track, shown in (D), where SETS slows the vehicle to prevent a collision with the wall. (**D**) We overlay the path of the vehicle onto a visualization of the hazard map. With no driver input, the intelligent reasoning of SETS slows and turns the vehicle as the course narrows to maintain safety. (**E**) For the tracked vehicle facing in the positive *x* direction, we show the spectrum of the local controllability Gramian, where columns correspond to natural motions, rows are states, and each element is colored by its controllability. Under degradation to its actuators (**F**), the vehicle is limited in its turning ability. The spectrum of the controllability Gramian enables SETS to efficiently interpret actuator degradation, allowing it to make updated plans according to the capabilities of the vehicle.

which our algorithm automatically adjusts the driver's command to maintain safety against tipping over and collisions. Passing through a particularly narrow portion of the track (less than 5 cm of clearance), without the need for an updated command from the driver, the robot slowed and turned to avoid hitting the walls from a nominal speed of 1 to 0.25 m/s. SETS predicted a collision if no diversionary maneuver was taken and then created and executed an updated plan.

This problem is challenging because of the complex human-robot interaction. In our experiments, human pilots prefer an interface that tracks commanded speed and direction rather than commanded waypoints, meaning that a position-space goal region for motion planning does not capture the human-robot interaction well. In addition, local optimization-based approaches are inapplicable because exploration is needed to consider nearby candidate trajectories to assist the driver. For example, the human driver may command a forward velocity, not understanding that there exists an impending collision or risk of tipping over, and instead of coming to a halt, our method explores alternative plans, including slowing down, backing up, and turning, that ultimately provide the commanded velocity tracking and match the operator intention. SETS adjusts the pilot's command to maintain safety while maintaining forward progress, displaying intricate maneuvers such as navigating closely around obstacles, decelerating when traversing ridges to avoid tip-over, and executing reverse turning maneuvers in tight corners, all while experiencing adversarial track degradation and time-varying dynamics.

While moving through the track, the vehicle was subject to actuator degradation that scaled the control limits of each drive motor by 25% in an alternating sequence of separate degradation and mixed degradation. This attack signal was randomly toggled on and off with a period of approximately 10 s. SETS was able to efficiently interpret this attack signal through the local spectrum of the tracked vehicle, shown in Fig. 3 (E and F). Here, the locally linearized dynamics have a 2D controllable subspace, one dimension associated with moving forward/backward and one dimension associated with turning left/right. First, we note the effect of degrading the actuator on the spectrum: The controllability of turning mode was decreased, providing the tree search an expressive interpretation of the current physical situation. Second, we note that the mode numbers 3, 4, and 5 had zero magnitude, implying that the tracked vehicle is not locally controllable (*22*). This illustrates an informal notion of nonlinear global controllability for systems that are not locally controllable: By stitching together modes from multiple locally linearized systems, SETS creates motions that are not available to a single linearized system. For example, a sideways translation can be achieved through the sequential combination of a forward snaking motion followed by a backward snaking motion, reminiscent of motion planning strategies in literature (*23*).

In program demonstrations, the same professional driver drove through the track with and without autonomous driver assistance. Throughout the experiment, the expert driver was asked to command the vehicle in the same adversarial manner with intentional commands to try to force a safety error. We show the number of safety violations in a selected run of the test circuit where adversarial and changing disturbances were present in Fig. 3B, and we see that SETS outperformed the driver alone and completely prevented safety violations. In the chicane section, at 2:36 in Movie 1, SETS caused the robot to turn and avoid a collision with the track, despite a driver command that would have caused an impact. In the chicane section, at 2:45 in Movie 1, SETS caused the robot to act more conservatively, slowing down as it traversed the narrow section. In addition, during program demonstrations, using SETS, even an inexperienced driver was able to safely navigate the course.

SETS acts as a planning module that interacts with custom perception, control, and safety algorithms. Running in model predictive control fashion, SETS generates trajectories of 1.6 s into the future every 0.1 s. SETS's ability to solve problems with general rewards, dynamics, and constraints facilitates its integration with the other autonomy components. For example, SETS interacts with a hazard map built with foundational vision models for segmentation to predict traversability. SETS uses the traversability map to generate a safe plan that takes into account complicated geometric and dynamic information about the robot's environment. SETS also interacts with an adaptive controller, which compensates for terrain changes and actuator degradation using parameter adaptation to rapidly update the corresponding dynamic function (*24*). SETS uses the system identification of the adaptive controller to update the dynamic function. The real-time nature of SETS and its ability to plan over a wide class of dynamics allow it to incorporate nonlinear dynamic updates and automatically benefit from this interaction.

Two collaborative spacecraft redirect debris

In the third and final hardware experiment, we deployed SETS on a team of two spacecraft to capture and redirect a piece of space debris with a tether, reminiscent of a NASA mission concept to capture and redirect a near-Earth asteroid (*25*). This experiment tested the ability of SETS to coordinate multiple robots and plan through high-dimensional nonlinear dynamics induced from contact forces and tether dynamics.

The experiment, depicted in Fig. 4A, took place in an arena with a smooth and ultraflat epoxy floor, where spacecraft robots hovered on air bearings and were actuated with onboard thrusters to simulate a frictionless space environment. SETS controlled the two cooperative spacecraft tethered together. The third spacecraft was not controlled by SETS and modeled an uncooperative target. The cooperative spacecraft were tasked with stopping the motion of the target and shepherding it to exit the arena in a desired direction. SETS predicted the motion of the three bodies and the tether, which was modeled with a spring-mass-damper finite element model, while only controlling the thruster inputs of the two cooperative spacecraft. This high-dimensional and underactuated problem tested the ability of SETS to plan for complex dynamics over a horizon. In particular, the two controlled spacecraft could only affect the dynamics of the target through contact with the center of the net, and the center of the net could only be influenced by the thrusters after propagating through the lattice elements of the tether. This chain of dynamics required deliberate maneuvering of the controlled spacecraft to redirect the target.

Despite these challenges, SETS automatically generated correct behavior because of its efficient representation of the dynamics. In Fig. 4B, the controllable modes of the system capture this networked structure of the finite element tether model. The most controllable states are those associated with the outermost nodes in the network, which are exactly the controlled spacecraft. The controllability of the nodes in the net decreases toward the center of the structure. The spectrum reveals that actuating the target is impossible before contact.

SETS interpreted this discrete representation of the dynamics to efficiently find near-optimal solutions. Four trials were tested by varying

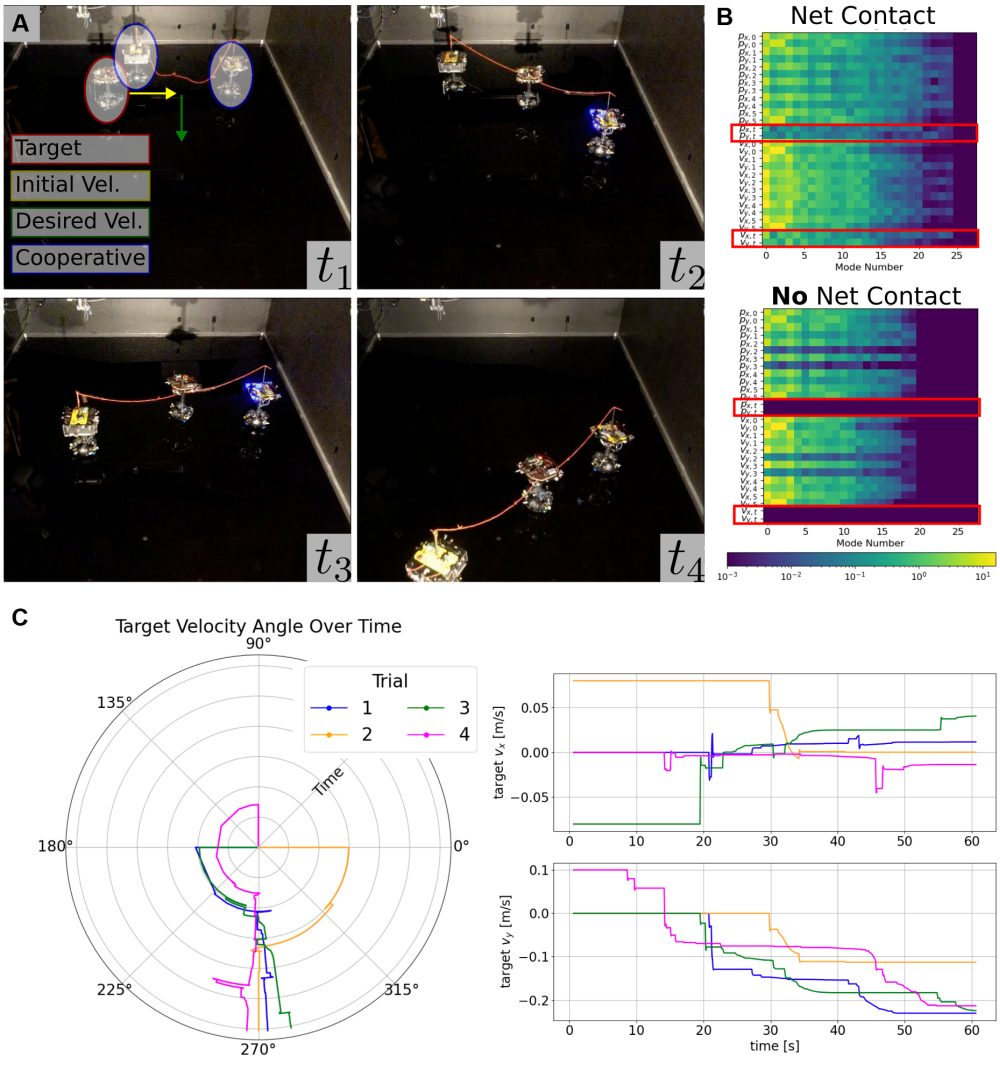


Fig. 4. Spacecraft experiment. (**A**) Still shots of the spacecraft experiment. The two tethered spacecraft arrest the motion of the target spacecraft. Planning through the tether-contact model, the controlled spacecraft shepherd the target spacecraft in the desired direction, toward the camera. These frames have been modified to reduce glare off the floor. (**B**) The spectrum of the controllability Gramian captures the high-dimensional dynamics of this problem. At the moment of contact, the states associated with the target (outlined in red) become controllable. (**C**) The direction of the target spacecraft velocity over the course of each experiment. In each case, the motion of the target spacecraft is correctly redirected.

the target's initial position and velocity. In the first, the target spacecraft was stationary, and SETS found a trajectory that sequentially deployed, captured, and redirected. In the second and third configurations, the target was initialized moving parallel to the initial cooperative spacecraft configuration, and quick motion was required of the tethered spacecraft to move into its way for capture. In the fourth, the target was initialized moving toward the initial cooperative spacecraft configuration, and they executed a "trampoline-like" maneuver, pulling the tether taut to springboard the target back in the desired direction.

The trajectory data for each case are shown in Fig. 4C. The right plots show the absolute velocities over time for each trial. The polar plot, which is coordinate-aligned with the snapshots in Fig. 4A, shows the relative angle between the x and y components of the target's velocity converge to 270°, which is the desired mission behavior. Successfully controlling this high-dimensional and underactuated system in real time was only possible with SETS and can ultimately enable new mission concepts.

Aerodynamic glider performs persistent observation

In this numerical experiment, we used SETS on a six–degree-of-freedom glider, which was tasked to observe a target in the presence of a thermal updraft, shown in Fig. 5. We analyzed tree data to make observations about the policy and value convergence. We also used this experiment to compare against external baselines and make ablation studies.

The environment is a 2 km–by–2 km–by–1 km volume arena and takes place over 10 min in simulation time, shown in Fig. 5A. The dynamics, parameters, and aerodynamic coefficients are from an autonomous glider in the literature (*26*). The interaction of aerodynamics, the observation objective, and the environmental thermal makes this a challenging planning problem: To generate enough lift to counteract gravity, the glider needs to average approximately 30 m/s of forward velocity. At the same time, drag drains the system's kinetic energy and, without exploiting the thermal, would cause the glider to crash into the ground after about 4 min. However, the glider can save itself by flying into the thermal, extracting energy from the environment, and resuming its observation task. SETS discovered this solution in real time, running in model predictive control fashion and generating 100 s of trajectory every 45 s. The trade-off between kinetic energy and potential energy in this periodic solution is visualized in Fig. 5B. This periodic solution is challenging to generate with existing motion planning or optimization frameworks because modeling the problem with a single static goal region would either cause the glider to crash or never observe the target. Furthermore, detouring to the thermal once at the target requires aggressive exploration and contradicts the goal-seeking behavior of local optimization.

The data for Fig. 5 (C and D) were generated by running SETS from a given state and analyzing the search tree. In Fig. 5C, we plotted SETS's root node value estimate versus the number of simulations. As predicted by our analysis in the "Theoretical results" section, our algorithm's branch length hyperparameter, H, controls the trade-off between the convergence rate and error: Short branches have slow convergence to small steady-state error, and long branches have fast convergence to large steady-state

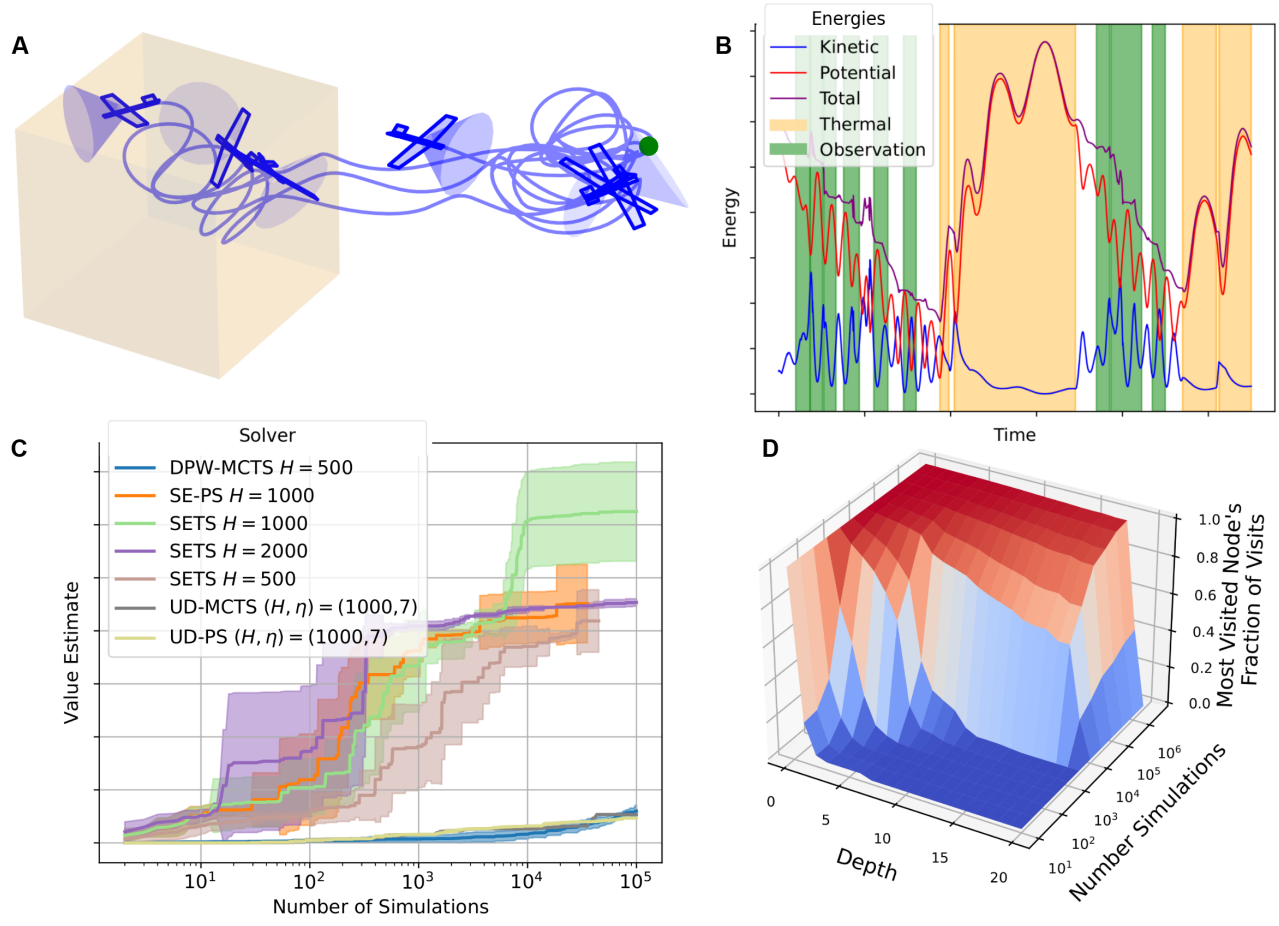


Fig. 5. Glider experiment. (**A**) The glider (blue) discovers an oscillation strategy between observing the target (green) to maximize its objective and passing through the thermal (orange) to gain energy and maintain altitude. (**B**) Kinetic energy and potential energy, in joules, of the glider over time. In nominal air conditions, the glider loses energy to drag and would eventually crash, but flying through the thermal (orange) provides upward force and allows the glider to accumulate potential energy. (**C**) Value convergence. As the number of simulations grows, the value estimate increases. As predicted by theory, the branch length parameter H controls the trade-off between convergence and error. The shaded region shows one standard deviation. We also plot the value estimate of each baseline. (**D**) Policy convergence. As the number of simulations grows, the visits concentrate deeper in the tree, and the plan becomes more refined.

error. In particular, we selected $H = 1000$ as the optimal branch length because the $H = 500$ case creates a high-complexity problem with a solution that cannot be well estimated within the operating regime of 10^5 simulations (10 min of wall-clock time), and the $H = 2000$ case creates a low-complexity problem with a solution that is well estimated but does not accurately represent the original problem.

We used the same value estimate data from the tree in the baseline study. This study was designed to isolate the effects of representation and exploration strategy. For choices of representation (low-level construction of nodes and edges), we implemented spectral expansion (SE), uniform discretization (UD), and double progressive widening (DPW) (*27*). The SE method is our approach, the UD approach uniformly discretizes the action space with η discrete points per dimension, and the DPW method adaptively samples from the action space, with more samples given to promising nodes. For choices of exploration strategy (high-level search on a set of existing nodes and edges), we implemented MCTS (Algorithm 1) and predictive sampling (PS) (*28*), where PS is a uniform sampling approach that returns the best trajectory found. The baselines and our method in Fig. 5C are a permutation of these representation and exploration strategies. The baseline data in Fig. 5C were generated by running each solver over 10 random seeds for branch lengths $H = 100$, 500, 1000, or 2000 and discretization levels $\eta = 3$, 5, 7, or 11, where only the best variant of each baseline is shown. Other than our approach, the only algorithm that performs well is spectral expansion with predictive sampling (SE-PS), where SE-PS with $H = 1000$ outperforms SETS with $H = 500$, suggesting that representation, not exploration strategy, is the most important component of optimal planning for dynamical systems.

In Fig. 5D, we defined a measure of the tree's confidence at a particular depth as the most visited node's fraction of the total visit count and plotted it versus the depth and size of the tree. The relative visitation frequency is a metric for the tree's confidence in its action selection because of the upper confidence bound rule of MCTS (*18*): Actions are selected by their optimistic value estimate, so when the visitations are concentrated to a single action, the optimistic value estimate of all the other actions is less than that of the highly visited (and well-estimated) selected action. As the number of simulations increases, concentration occurs deeper in the tree,

indicating that the plan is being refined further into the future, and the tree has reached sufficient confidence in the plan until that point. When running in receding horizon fashion, the time between replanning steps should be large enough to allow a highly confident plan to develop.

DISCUSSION

We present a qualitative comparison with existing approaches and discuss how our work can affect the field of robotics.

Optimization and gradient-based planning

Some planning problems can be solved with convex optimization techniques (*5*, *6*). Taking advantage of fast numerical solvers [e.g., (*29*)], this family of methods has been shown to solve, in real time, a variety of challenging dynamical problems, including quadrotor drone racing (*30*), in-orbit assembly (*31*), bipedal locomotion (*32*), swarm coordination (*33*), and swarm coverage (*34*). A variety of techniques, including sequential convex programming (*5*, *33*), collocation (*32*), and single and multiple shooting methods (*35*–*37*), enable this method to be applied to nonconvex problems, although theoretical analysis of these methods has shown that convergence is limited to local minima (*38*, *39*). The general planning problem's state, input, dynamics, and reward specifications can each create local minima traps under which a pure exploitation strategy will fail to find the global optimal solution. The classical example is the bug trap problem (*40*), where the obstacle configuration will cause methods that do not explore to converge to a highly suboptimal point. Recent work (*41*) has proposed avoiding local minima in motion planning by partitioning the environment into convex sets and solving a relaxed mixed-integer program.

Compared with these approaches, SETS provides globally optimal solutions by performing a discrete search on a carefully constructed representation of natural motions. For example, the quadrotor avoided local minima traps from obstacles and wind gusts, and the glider temporarily ignored the high reward of visiting the target and instead navigated to the thermal to maintain altitude and energy constraints.

Sampling-based motion planning

The standard robotics approach to overcome the nonconvexities inherent in problem data, such as traps from obstacle configurations, is sampling-based search (*4*). Foundational methods, such as probabilistic roadmaps (*3*) and rapidly exploring random trees (RRT) (*2*) and their variations [e.g., RRT* (*42*)], sample configurations and use a local planner to build trees and graphs on which to perform global optimization. Although RRT was originally designed for kinodynamic planning, these planners are most commonly used as geometric path planners.

Applying sampling-based planning for dynamical systems has two challenges: First, the higher-order states increase the dimensionality of the search space, resulting in slow convergence and poor performance, and second, these methods rely on the existence of a local planner capable of solving arbitrary two-point boundary value problems. Addressing these challenges is an active area of research. For example, stable-sparse-RRT (*43*) relaxes the knowledge of a local planner by sampling control inputs to generate dynamically feasible paths and then prunes redundant edges to maintain a sparse tree. Another work (*44*) considers quasi–Monte Carlo sampling to speed convergence of tree search for control-affine and driftless control-affine systems. Discontinuity-bounded A* (*45*) is similar to our work in that they plan for high-dimensional dynamical systems over a discrete structure of motion primitives and is different because their motion primitives are generated offline by sampling input and goal states. The generation and integration of motion primitives into discrete planning have a rich history (*46*–*48*).

As in our work, the differential fast marching tree (DFMT) method (*49*) uses the controllability Gramian for planning with dynamical constraints. Whereas DFMT uses the Gramian to verify reachability of a given two-point boundary value problem, with their terminal point uniformly sampled from the state space, our work explicitly uses the spectrum of the Gramian to generate the terminal point in a provably efficient exploration strategy. This distinction is important because uniform sampling poorly covers high-dimensional spaces. In addition, whereas DFMT considers linear systems and a single global Gramian, we consider nonlinear systems, construct a local Gramian at each node, and bound the linearization error with an additional tracking control procedure. Other inspired works that blend control and planning are LQR-trees (*50*) and funnel library planning (*51*), both of which rely on sum-of-square programming to compute regions of attraction upon which the discrete planner searches.

Most continuous space planning algorithms scale poorly to high-dimensional spaces because they generate nodes and edges by sampling from the state or action space. Unlike existing methods, SETS uses the spectrum of the locally linearized system to generate nodes and edges, resulting in a branching factor that is linear in the state dimension. This conceptual difference manifests itself in new search capabilities: To the best of our knowledge, kinodynamic motion planners that search in real time through high-dimensional nonlinear dynamics, such as a 12D quadrotor with DNN-modeled wind effects, do not exist.

In addition to the differences in search strategies, our work differentiates itself in the generality of the problem assumptions. Whereas motion planning problems from the robotics community focus on static, position-space goal regions (*52*), the planning problem defined by the computer science community (*53*) relaxes the problem setting. This is useful in instances where a goal region does not capture the essence of the problem or when a goal region is difficult or impossible to define. For example, in chess, a notion of a goal region could be the set of all positions in which the opponent's king is checkmated, but this set is challenging to enumerate or path to with a motion planning method. With this in mind, the experiments in this paper were selected because they are easily formulated as decision-making problems, but they cannot be cast as motion planning. For example, in the tracked vehicle experiments, human drivers preferred interfacing with an algorithm that tracks their commanded velocity rather than navigating to a waypoint, implying that a conventional motion planning framework with a position goal region does not model the human-robot interaction well. In the glider experiment, a static goal region does not exist because the optimal behavior is to oscillate between observing the target and the thermal. Besides, immediately growing the set of feasible robot behaviors, the more general problem framework is a more natural

foundation to extend to stochastic dynamics (*54*), partially observable (*55*–*57*), and game-theoretic (*58*) settings.

There is a recent body of work on sampling-based model predictive control methods, including cross-entropy motion planning (*59*) and model predictive path integral control (*60*). By sampling many trajectories and then performing a weighted average, these methods have shown impressive performance in scenarios including autonomous driving (*60*) and manipulation (*61*). These methods have also shown straightforward integration with high-fidelity simulators (*62*) and learned dynamic models (*60*). These strategies traditionally perturb inputs with Gaussian noise to promote exploration. However, without careful tuning of noise distributions and averaging temperature, this can lead to slow exploration or the generated policies getting stuck in local minima. Similar to our work, it is possible to consider sampling over abstractions, such as splines in state space (*61*), and these are active areas of research. Whereas these methods explore by perturbing trajectories with Gaussian noise, our approach uses a theoretically rigorous balance of exploration and exploitation, and we demonstrate that this results in convergence to globally optimal solutions.

Sampling-based search is also investigated in the partially observable case. Algorithms for handling continuous state and action spaces have focused on particle filtering methods, with specializations to overcome the challenges inherent in uncertain observations, such as sharing observation data in the tree (*63*), progressive widening (*56*), or linear filtering (*57*). We bring our attention to the fully observed case to focus on the difficulty of decision-making, rather than combined decision-making and information gathering. There is recent work that transforms a stochastic planning problem into a deterministic planning problem (*64*), moving the difficulty of dealing with stochastic dynamics and observations into the same decision-making framework we consider.

Reinforcement learning

Our problem setting is closest to planning in model-based reinforcement learning: Once the dynamics and reward models are learned, the planning component finds the optimal value and policy. The classical planning methods using dynamics and reward models are value and policy iteration (*1*, *7*). The fundamental issue with these methods, which persists in modern approaches, is that representing a high-dimensional continuous space has a high complexity (*53*, *65*). The direct approach is to discretize the state and action space and run value iteration, but this has a storage complexity that is exponential in the state dimension, making these methods computationally intractable for high-dimensional robotics applications. Research has developed in multigrid (*66*) and adaptive grid (*67*) representations, but the practical gains in scaling to high-dimensional systems are marginal. Recent work (*68*) has proposed a tensor-based method that exploits a low-rank decomposition of value functions, enabling efficient discretization and improved policy generation on systems of comparable dimensionality to our experiments.

An alternate reinforcement learning approach is that of model-free methods that directly learn correlations between observations and optimal actions without explicitly using a dynamic or reward function (*69*, *70*). Policy gradient methods have offered a powerful technique to handle continuous state and action spaces, with methods such as proximal policy optimization (*71*) becoming a standard tool in the community. In special cases, such as linear quadratic regulation (*72*), global convergence results exist. However, the general convergence of these algorithms in continuous space is limited to local stationary points (*73*). Additional work has gone on to classify these stationary points between problems, drawing conclusions about structural similarities that help policy gradient methods converge globally (*74*). These methods, although powerful and general, require large datasets and an offline training phase. These methods suffer fundamentally from domain shift, the difference between the offline training environments and the environment of the deployed system. In contrast with data-driven reinforcement learning methods, our algorithm is able to run on a never-before-seen problem with guaranteed global convergence to an optimal solution, thus avoiding the danger of domain shift.

A natural direction to develop real-time intelligence is via anytime algorithms that can be stopped at an arbitrary point, returning satisfactory solutions that increase in quality as more time is given. In the field of decision-making, the prototypical example is MCTS (*20*), an algorithm that simulates random trajectories while biasing toward actions of high reward. Exhaustive (*75*) and uniformly random searches (*28*) have been shown to be effective in some scenarios, but the improved strategic exploration of MCTS enables convergence in problems where simpler techniques fail to efficiently search the combinatorially large space. Although MCTS can perform very well in traditional artificial intelligence settings such as games, the complexity of the high-dimensional and continuous world presents a fundamental challenge. Random search using simulators (*28*) has been investigated, but the investigated search strategies scale poorly. Directly sampling the action space (*54*), even with sophisticated strategies (*76*, *77*), induces trees with large width and depth: Sampling high-dimensional continuous action spaces creates a high branching factor, and discretizing time creates a large number of decisions over the horizon.

Temporal abstraction, or options (*78*), is a principled framework to make decisions over sequences of actions and reduce the complexity of decision-making. Options are analogous to motion primitives in motion planning. There has been work in option construction in planning (*79*) and using MCTS (*80*, *81*), where options are handcrafted. There has also been work in option construction (*82*) in the partially observable setting. An interpretation of our method is that it automatically generates options in a dynamically informed and provably correct framework.

Data-driven methods (*83*) and combinations with gradient-based techniques (*84*–*86*) have also been deployed for decision-making. However, their reliance on large amounts of demonstration data during an offline training phase limits their applicability to systems and scenarios where the complete problem data are known ahead of time. In contrast, our SETS method can be deployed on a never-before-seen problem and, for any allowed computational budget, produce a plan of approximately optimal decisions that increases in quality with more time.

Importance

We expect SETS to positively affect the field of autonomous robots and research in autonomous decision-making. From a design perspective, SETS provides many important advantages: First, SETS solves a broad class of Markov decision processes (MDPs) and therefore interfaces naturally with other autonomy components and can be used for new and diverse tasks and systems. This relieves the burden on the designer and extends the operational

envelope of autonomous behavior. Second, because the search tree of SETS can be visualized and analyzed, it has a high degree of explainability and can be tuned and verified by the user. Third, because SETS is efficient enough to run in real time, it can react to new information on the fly with real-time computation. For these reasons, we believe that SETS can be a default choice for planners in a wide range of autonomy applications.

From a research perspective, SETS builds an important connection between dynamical systems and machine learning. To develop this connection, we interpret the tree policy as an online learning process of a categorical distribution on each node's children. Our first theoretical result (Theorem 1), which applies to a broad class of dynamical systems, suggests that the spectrum of the local controllability Gramian can be used as provably correct and widely applicable learning features. These features enable real-time learning for complex dynamical systems by simplifying the decision-making problem to selecting among a set of natural motions of the system. Although beyond the scope of this work, we believe that these features can be used for offline policy learning, kinodynamic motion planning, and other sampling-based planning, providing reduction in computational complexity and improved convergence rates.

From the algorithmic complexity perspective, SETS reduces complexity in two separate mechanisms as compared with the uniform-discretization approach: First, because SETS plans over trajectories instead of individual actions, the total tree depth is decreased by a factor of H, where H is the branch length or duration of a trajectory generated by the spectral nodal expansion. Second, the width of the tree (branching factor) is decreased from an exponential dependency on control dimension to a linear dependency on state dimension. This is enabled because each child node uniquely tracks one of the basis vectors of the controllable subspace, and, from linear theory, the number of basis vectors is upper bounded by the state dimension. In contrast, for a UD, the number of elements in the m-cube has an exponential dependence on m. The number of combinations in the tree (i.e., width to the power of depth) is important because it appears in the convergence rate analysis of MCTS. This fact is revealed by our second theoretical result, Theorem 2, which presents a complete MCTS value convergence result for deterministic systems that does not require knowledge of privileged problem information (the "gap" parameter).

MATERIALS AND METHODS

In this section, we present the problem formulation, algorithmic overview, and theoretical analysis of convergence to global optimality.

Problem formulation

We consider an MDP (*87*), an abstract problem description written as a tuple of components: $\langle X, U, F, R, D, \Omega, K, \gamma \rangle$. Here $X \subseteq \mathbb{R}^n$ is a compact state space, $U \subseteq \mathbb{R}^m$ is a compact action space, $F: X \times U \to X$ is the discrete-time dynamics, $R: X \times U \to [0, 1]$ is the stage reward, $D: X \to \mathbb{R}_{\geq 0}$ is the terminal reward, $\Omega \subset X$ is a set of unsafe states, $K \in N$ is the time horizon, and $\gamma \in [0, 1]$ is the discount factor. At an initial state $\boldsymbol{x}_0$, the planning problem is to select a sequence of actions that maximizes the sum of the stage reward plus the terminal reward, subject to the dynamics and state/action constraints:

$$V^*, \boldsymbol{x}^*_{[K]}, \boldsymbol{u}^*_{[K]} = \underset{\boldsymbol{x}_{[K]},\ \boldsymbol{u}_{[K]}}{\operatorname{argmax}} \sum_{k=1}^{K} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{u}_{k+1}) + \gamma^K D(\boldsymbol{x}_K)$$
$$s.t.\ \boldsymbol{x}_k = F(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k),\ \boldsymbol{x}_k \in X \backslash \Omega,\ \boldsymbol{u}_k \in U,\ \forall k \in [1, K] \quad (1)$$

where k is the time step index and a bracket subscript indicates a sequence, e.g., $\boldsymbol{x}_{[K]} = \left[\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_K^T\right]^T \in \mathbb{R}^{nK}$.

Similar frameworks of decision-making over a horizon are considered by the reinforcement learning community and in optimal control. The space of problems we consider is focused on smooth, deterministic, and nonlinear dynamics with continuous state and action spaces. Because the systems we consider operate in continuous time, we discretize time uniformly so each discrete time step k index has a duration of Δt seconds.

Monte Carlo tree search

The pseudocode for SETS is shown in Algorithm 1. SETS performs a MCTS with a specialized nodal expansion operator, defined as SE in the pseudocode. We briefly summarize the procedure of MCTS, with more in-depth descriptions available in the literature (*20*): Although the robot has remaining computational budget, simulate future state trajectories from the current world state forward to the horizon of the MDP. At each node, the tree policy selects the best child by balancing exploration of visit counts and exploitation of observed reward. If a node is not fully expanded, generate a new child by taking an action and stepping forward in time. When a simulated trajectory terminates, the accumulated reward and visit count information are backpropagated up the tree to update the total value and the number of visits at each node in the path, and the process iterates. We adopt for following notation for the pseudocode: p is the "path" and the list of nodes in one iteration, i is a single node, $r(i)$ is the reward to node i, $C(i)$ is the set of node i's children, $\tilde{V}(i, \ell)$ is the total value of node i after ℓ iterations, $T(i, \ell)$ is the number of visits of node i after ℓ iterations, and $V(i, \ell)$ is the average value of node i after ℓ iterations.

Whereas the standard MCTS variant (*18*) uses a logarithmic exploration term, SETS uses a polynomial exploration term (see line 6 of Algorithm 1), with constants $c_1 = 1$, $c_2 = 0.5$, and $c_3 = 1$. This technique is supported by our theoretical analysis in Theorem 2, as well as other MCTS variants, both in theory (*54*, *88*) and practice (*89*). Although the dynamics, reward, and SE operator are deterministic, SETS is a stochastic algorithm because, when multiple children of a node have not been visited, the algorithm randomly selects one. This stochasticity, which is the same as the original MCTS algorithm, is noted in line 6 of Algorithm 1.

Spectral expansion operator

The SE operator, specified in Algorithm 1, lines 16 to 29, computes a trajectory of length H. The first step is to compute the linearization and eigendecomposition of the local controllability Gramian. We consider both time-invariant and time-varying linearizations, where the linearized system data are either computed once at the current state and a single nominal control input or over a time-varying trajectory initialized at the current state, subject to a nominal control sequence. In practice, we found that time-varying linearization produces more stable trajectories, especially for highly agile platforms such as a quadrotor, and we select the unforced dynamics as the default nominal control input ($\overline{\boldsymbol{u}} = 0$).

Algorithm 1. Spectral Expansion Tree Search (SETS)

def SpectralExpansionTreeSearch($\mathbf{x}_0$, MDP, H):
  /* set root */
  $i_0 =$ Node($\mathbf{x}_0$);
  for $\ell = 1, \dots,$ **do**
    /* initialize path at root then rollout */
    $p = [i_0]$;
    for $d = 1, \dots, [K/H]$ **do**
      /* best child (if not fully expanded, choose randomly) */
      $i^* = \arg\max_{i \in C(p[-1])} V(i, \ell) + c_1 \frac{T(p[-1],\ell)^{c_3}}{T(i,\ell)^{c_2}}$;
      if i^* is not expanded **then**
        SpectralExpansion(i^*, $\mathbf{x}_0$, $\bar{\mathbf{u}}$, MDP, H);
      if $i^* \in \Omega$ **then** break;
      p.append(i^*);
    /* backup */
    for $d = 1, \dots, |p|$ **do**
      $\tilde{V}(p[d], \ell) += \sum_{t=d}^{\lceil K/H \rceil} \gamma^{tH} r(p[t])$;
      $T(p[d], \ell) += 1$;
    yield $V(\mathbf{x}_0, \ell)$

def SpectralExpansion(i, $\mathbf{x}_0$, $\bar{\mathbf{u}}$, MDP, H):
  /* compute local linearization */
  $A_k, B_k, c_k = \text{Linearization}(F, \mathbf{x}_0, \bar{\mathbf{u}}_{[H]}) \quad \forall k \in [0, H-1]$;
  $\mathbf{z}_{k+1} = L_k(\mathbf{z}_k, \mathbf{u}_{k+1}) = A_k \mathbf{z}_k + B_k \mathbf{u}_{k+1} + c_k \quad \forall k \in [0, H-1]$;
  /* compute spectrum of normalized controllability Gramian */
  $S = \text{diag}\left(\left\{ \frac{2}{\bar{u}_j - u_j} | \forall j \in [1, m] \right\}\right)$;
  $C = \left[\left(\prod_{k=1}^{H-1} A_k \right) B_0 S, \left(\prod_{k=2}^{H-1} A_k \right) B_1 S, \dots, A_{H-1} B_{H-2} S, B_{H-1} S \right]$;
  $[\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n], [\lambda_1, \lambda_2, \dots, \lambda_n] = \text{eig}(CC^T)$;
  /* compute linear reference trajectory for the ith branch */
  $\mathbf{z}_H = (-1)^{i\%2} \sqrt{\lambda}_{i//2} \mathbf{v}_{i//2} + \left(\left(\prod_{k=0}^{H-1} A_k \right) \mathbf{z}_0 + \sum_{k=0}^{H-1} \left(\prod_{j=k}^{H-1} A_j \right) c_k \right)$;
  $\mathbf{u}_{[H]}^{\text{ref}} = \text{clip}(C^\dagger \mathbf{z}_H, U)$;
  $\mathbf{z}_{[H]}^{\text{ref}} = L^H(\mathbf{z}_0, \mathbf{u}_{[H]}^{\text{ref}})$;
  /* track reference trajectory with nonlinear system */
  $M_k = \text{DARE}(A_k, B_k, \Gamma_x, \Gamma_u)$;
  $\mathcal{K}_k = (\Gamma_u + B_k^\top M_k B_k)^{-1} B_k^\top M_k A_k$;
  $\mathbf{u}_{[H]} = \left\{ \text{clip}(\mathbf{u}_k^{\text{ref}} - \mathcal{K}_{k-1}(\mathbf{x}_{k-1} - \mathbf{z}_{k-1}^{\text{ref}}), U) \middle| \forall k \in [1, H] \right\}$;
  $\mathbf{x}_{[H]} = F^H(\mathbf{x}_0, \mathbf{u}_{[H]})$;
  return $(\mathbf{x}_{[H]}, \mathbf{u}_{[H]}, R^H(\mathbf{x}_0, \mathbf{u}_{[H]}))$;

From the linearized system, we construct the controllability matrix C, which is a linear mapping from sequences of control actions to the resulting terminal state. This matrix and its associated Gramian are well known in linear control theory (*90*), with two important properties: First, for linear systems, the set of states reachable with one unit of control energy is an ellipse parameterized by the spectrum of the Gramian, and second, the pseudoinverse of the controllability matrix applied to a feasible desired terminal state computes the minimum energy control input that drives the system to that state.

In lines 19 and 20 of Algorithm 1, the inputs are scaled by their control limits before computing the Gramian. This procedure scales the notion of bounded energy of the reachable set to the control limits rather than inputs that potentially differ by orders of magnitude. An alternate preconditioning matrix $\mathbf{S}$ could be selected here to create a different trade-off in control energy. In addition to informing the input rescaling, the elliptical reachable sets property reveals a simple exploration strategy for linear systems: The vertices of the ellipse, which are computed via the spectrum of the Gramian, form a bounded covering of the reachable set.

In lines 22 to 24, we compute the linearized reference trajectory. We iterate through the modes using the integer floor division and modulus operators, // and %, respectively. Visiting each mode in the plus and minus direction implies a total branching factor of $2n$, where n is the state dimension. The pseudoinverse of the controllability matrix mapped onto the desired state yields a trajectory for the linearized system that, by the pseudoinverse mapping property, is minimum energy and reaches the target. However, the notion of minimum and bounded energy is over the entire trajectory, and to verify that each individual control action is valid, we impose the bounded input constraint with a clip operation: Given a vector and an interval, the values outside the interval are clipped to the interval edges.

Whereas the reference trajectory is dynamically feasible for the locally linearized system, the actual branch trajectory must satisfy nonlinear dynamics. In lines 25 to 28, we compute a feedback controller from the discrete algebraic Riccati equation (*91*) using an existing software implementation (*92*) and roll out a trajectory of the nonlinear system that tracks the linear reference trajectory to the desired mode. In lines 28 and 29, we use the multistep notation for the dynamics and reward, e.g., $F_H(\boldsymbol{x}_0, \boldsymbol{u}_{[H]}) = F(\dots F(F(\boldsymbol{x}_0, \boldsymbol{u}_1), \boldsymbol{u}_2), \dots, \boldsymbol{u}_H)$. We include a proof of the controller stability in lemma 14 in the Supplementary Materials.

Heuristics and deep learning

At the cost of an increased branching factor, the user can add manually designed nodal expansion heuristics to guide the search. In the quadrotor experiments, we used a heuristic to guide the system to the nearest target by including the projection of the nearest target onto the linear set as a branching option. This is similar to goal-biased sampling used in the sampling-based motion planning community (*40*). For the remaining experiments, we did not use heuristics and relied only on the natural exploration of SETS. In a manner similar to AlphaZero (*89*) and our own work on neural tree expansion (*93*), it is also possible to incorporate DNN-based heuristics to predict the value of the next child state.

The user can also manually decrease the branching factor by prioritizing certain degrees of freedom. In the quadrotor experiment, we found that only searching among the velocity and angular velocity modes led to higher performance. Physically, the system maintains its ability to translate and make attitude adjustments, because the eigenvectors that maximally excite the velocity and angular velocity coordinates also create changes in the position and attitude, respectively. We expect this reasoning to be applicable in many second-order systems, and we apply it in all our experiments. The final implementation detail is that we return the maximum valued trajectory (rather than the child of highest average value). This practice maintains theoretical guarantees because the maximum valued trajectory always has a higher value than the average value because of our deterministic rewards setting.

Theoretical results

Here, we present our two main theoretical results, which are combined to show that SETS quickly converges to a bounded-error

solution of the planning problem (*1*). The proofs and supporting lemmas are in the Supplementary Materials.

We make the following assumptions:

Assumption 1

The dynamics F are twice differentiable, the reward R is Lipschitz and depends only on the state, and the input set U is bounded in a product of intervals:

$$F \in C_2(X \times U; X) R \in Lip_1(X; \mathbb{R}) U \subseteq \left\{ \boldsymbol{u} \in \mathbb{R}^m \mid \underline{u}_j \leq u_j \leq \overline{u}_j \right\}$$

where $Lip_1(X;\mathbb{R})$ is the space of Lipschitz functions from X to $\mathbb{R}$. Because X is compact and R is Lipschitz, R is therefore bounded. Without loss of generality, we suppose $R(\boldsymbol{x}) \in [0, 1]$.

Our first result, Theorem 1, states that SE creates a bounded equivalent discrete representation of continuous MDPs.

Theorem 1

Consider an MDP $\langle X, U, F, R, D, \Omega, K, \gamma \rangle$ that satisfies Assumption 1. For the initial state $\boldsymbol{x}_0$, SE with branch length H creates a discrete representation with a bounded equivalent optimal value function

$$\left| V^*(\boldsymbol{x}_0) - V^*_{\mathrm{SETS}}(\boldsymbol{x}_0) \right| \leq \frac{\kappa_0 + \gamma^H}{1 - \gamma^H} \left(\kappa_1 \left(1 + \kappa_2 \Delta t \right)^H + \kappa_3 \right) \quad (2)$$

where $\kappa_{0,1,2,3}$ are problem-specific positive constants and Δt is the temporal discretization of continuous-time dynamics.

To derive this theorem, we first show that the distance between optimal value functions of two discounted MDPs is upper-bounded by a constant times the distance between their reachable sets. Then, we show that the finite number of trajectories generated by spectral expansion efficiently covers the reachable set of the continuous nonlinear system. We achieve this by introducing two intermediate sets: the reachable set of the linear system subject to energy constraints and the collection of states constructed from the spectrum of the controllability Gramian. This procedure is visualized in Fig. 6.

Our second result, Theorem 2, states that the value estimate of MCTS converges to the optimal value function for discrete MDPs.

Theorem 2

Consider an MDP $\langle X, U, F, R, D, \Omega, K, \gamma \rangle$ with a finite action set. Consider a node i at depth d in the tree, where the tree is grown using Algorithm 1. Let $T(i, \ell)$ be the number of visits to node i after ℓ iterations and let $V(i, \ell)$ be the average value after ℓ iterations. Then, there exist nonnegative constants such that the expected value converges to the optimal value at node i

$$\left| V^*(i) - \mathbb{E}\left[V(i, \ell) \mid T(i, \ell) = \tau \right] \right| \leq \frac{\kappa_4}{\tau^{\kappa_5}} \quad (3)$$

and the distribution of values sampled in the tree concentrates to the expected value. $\forall z > 0, \forall \tau \geq 1$

$$\mathbb{P}\left[\left| \mathbb{E}\left[V(i, \ell) \right] - V(i, \ell) \right| \geq \frac{z}{\tau^{\kappa_6}} \mid T(i, \ell) = \tau \right] \leq \frac{\kappa_7}{z^{\kappa_8}} \quad (4)$$

where $\kappa_5 = \kappa_6 = 0.5$ under our choice of exploration law.

Our Theorem 2 analysis draws from the recent convergence analysis in the literature (*88*). We were particularly inspired by their proof strategy of backward induction and nonstationary bandits. However, we studied a fundamentally different problem setting with deterministic rewards. Deterministic rewards, the standard setting for robotics, isolates the source of uncertainty in the tree search to the uncertainty in the future policy rather than combined uncertainty from both stage rewards and future policy. This difference leads to a departure in the low-level analysis, as well as some tangible improvements: First, our exploration law requires less information about the problem. Whereas the existing work's exploration law (*88*) requires knowledge of the gap from bandit literature (which is typically unknown), we do not require this knowledge. Second, our exploration law is simpler. Whereas the existing work's exploration law (*88*) is a complex function of depth of the tree, our constants are

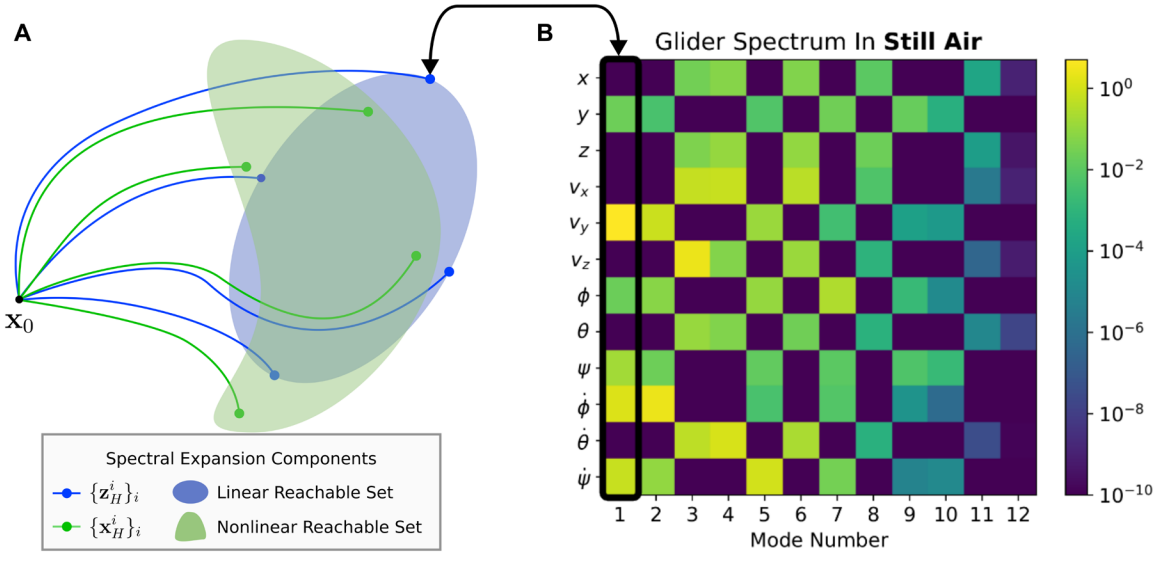


Fig. 6. Spectral expansion. (**A**) The SE operator. The nonlinear reachable set is shaded in green, and the locally linearized reachable set is shaded in blue. The modes of the controllability spectrum are the axes of the ellipse, and these are used to compute SETS trajectories with contraction-based feedback control. (**B**) We show the glider's spectrum, where each column corresponds to a natural motion, each row corresponds to a state dimension, and each element is colored according to its controllability. We include this figure to connect the theory to the data of the three experiments: Each column of the image in (B) corresponds to a controllable mode z_H^i in (A).

depth independent. Both our analysis and that in (*88*) share the same final convergence rate of order $\ell^{-1/2}$.

Application of these two results with the triangle inequality [using $V^*_{\mathrm{SETS}}(\boldsymbol{x}_0)$ as a cross term], and evaluating Theorem 2 at the root shows that SETS quickly converges to the optimal solution of the continuous-space planning problem:

Theorem 3

Consider an MDP $\langle X, U, F, R, D, \Omega, K, \gamma \rangle$ that satisfies Assumption 1. For initial state $\boldsymbol{x}_0$, SETS (Algorithm 1) with branch length H yields a value estimate as a function of number of iterations ℓ satisfying

$$\left|V^*(x_0) - \mathbb{E}\left[V(x_0, \ell)\right]\right| \leq \frac{\kappa_4}{\sqrt{\ell}} + \frac{\kappa_0 + \gamma^H}{1-\gamma^H}\left(\kappa_1\left(1+\kappa_2\Delta t\right)^H + \kappa_3\right) \tag{5}$$

where $\kappa_{0,1,2,3} > 0$ are as in Theorem 1 and κ_4 is as in Theorem 2.

The first term in Theorem 3 goes to zero as the number of iterations ℓ increases. In addition, its leading constant κ_4 scales with a polynomial power of the total number of action sequences. The tree formed by SETS has a branching factor upper bounded by $2n$ (the number of controllable modes) and the number of decisions equal to K/H. Therefore, an increase of the hyperparameter H improves the convergence speed of the value estimate: A longer branch length results in fewer overall decisions and therefore a faster convergence.

Although providing faster convergence, the trade-off of a longer branch length is a larger asymptotic error, shown in the second term and labeled steady-state error. We empirically validated this trade-off with an illustrative example in Fig. 7. This error term can also be controlled with other parameters. For example, the term with the worst growth behavior, $(1 + \kappa_2\Delta t)$, can be made arbitrarily small by decreasing the integration time of the simulator. However, making this change incurs a higher computational cost per trajectory, limiting the total number of iterations ℓ finished by the end of the planning budget. Similarly, the entire error term can also be controlled by decreasing the discount factor γ, at the cost of using less information about the future.

The fast optimality convergence of SETS also assists the robot to find a feasible solution. Under a well-selected terminal reward D, the trajectory corresponding to the optimal value function is guaranteed to be feasible over the full horizon. One possible choice is $D(\boldsymbol{x}) = K \max_{\boldsymbol{x}} R(\boldsymbol{x})$ where the maximum reward is known from Assumption 1. From this terminal reward, the highest-valued infeasible trajectory incurs less reward than the least-valued feasible trajectory, and therefore the optimal trajectory must be feasible if a feasible solution exists. A proof of this reasoning was made in recent work (*94*). Although the optimal trajectory of the transformed problem is guaranteed to be feasible, the trajectory returned by SETS may not be if the number of iterations ℓ is insufficient to converge to a feasible trajectory. In practice, the algorithm converged to a sufficiently optimal value that the solution was also feasible.

Our theoretical analysis validates the algorithm and provides an explainable interpretation of the planning process. In addition, the discussion of the effect of branch length H and, to a lesser extent, Δt and γ makes it clear that our analysis also enables systematic parameter design of various decision-making agents. For example, if a robot operates in a dynamic environment and has to replan frequently to react to new information, then the designer can tune parameters to sacrifice some combination of asymptotic error (larger H), dynamic fidelity (larger Δt), or long-term planning (smaller K). Alternatively, when a robot operates in a relatively static environment but has complex long-term behavior to discover, the designer should allocate more budget to each plan. The former example corresponds to our quadrotor experiment in the "Quadrotor navigates a dangerous wind field" section, and the latter example corresponds to the glider experiment in the "Aerodynamic glider" section.

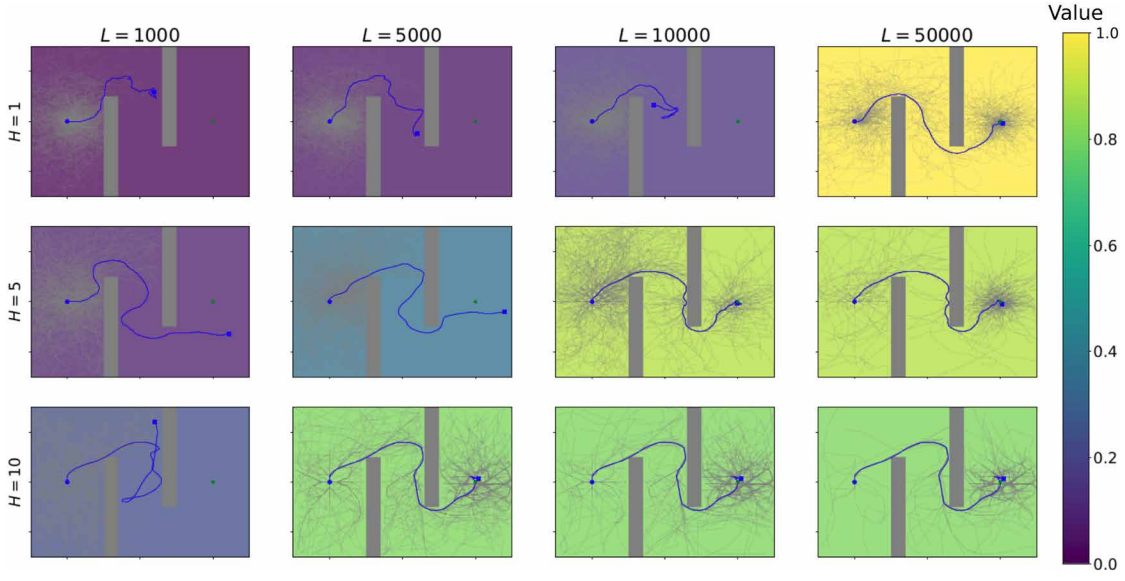


Fig. 7. Convergence rate versus asymptotic error tradeoff. We evaluate SETS on a motion planning problem, where a 2D double integrator starts at the blue dot on the left and is tasked to reach the green dot on the right. We vary the branch length parameter H and the total number of simulations L. Each plot is shaded by the value of its trajectory. The empirical trend of the value is predicted by theory: Trees with large H converge quickly to suboptimal solutions, and trees with small H converge slowly to highly optimal solutions.

Last, although we combined our main theoretical results into the SETS algorithm, they might be of independent interest as individual components. For example, a PS or model predictive path integral approach might benefit from an SE representation, and a traditional game-playing artificial intelligence agent might benefit from our improved MCTS analysis.

Supplementary Materials

The PDF file includes:
Supplementary Text
Table S1
Legends for movies S1 and S2

Other Supplementary Material for this manuscript includes the following:
Movies S1 and S2

REFERENCES AND NOTES

1. R. Bellman, *Dynamic Programming* (Princeton Univ. Press, 1957).
2. S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Research Report 9811 (1998).
3. L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.* **12**, 566–580 (1996).
4. A. Orthey, C. Chamzas, L. E. Kavraki, Sampling-based motion planning: A comparative review. *Annu. Rev. Control Robot. Auton. Syst.* **7**, 285–310 (2023).
5. D. Morgan, S.-J. Chung, F. Y. Hadaegh, Model predictive control of swarms of spacecraft using sequential convex programming. *J. Guidance Control Dyn.* **37**, 1725–1740 (2014).
6. D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, B. Acikmese, Convex optimization for trajectory generation. arXiv:2106.09125 [math.OC] (2021).
7. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).
8. Y. K. Nakka, W. Honig, C. Choi, A. Harvard, A. Rahmani, S.-J. Chung, Information-based guidance and control architecture for multi-spacecraft on-orbit inspection. *J. Guidance Control Dyn.* **45**, 1184–1201 (2022).
9. L. P. Kaelbling, T. Lozano-Perez, Hierarchical task and motion planning in the now, in *IEEE International Conference on Robotics and Automation* (IEEE, 2011), pp. 1470–1477.
10. C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, T. Lozano-Perez, Integrated task and motion planning. *Annu. Rev. Control Robot. Auton. Syst.* **4**, 265–293 (2021).
11. B. Paden, M. Čáp, S. Z. Yong, D. Yershov, E. Frazzoli, A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intel. Vehicles* **1**, 33–55 (2016).
12. W. Schwarting, J. Alonso-Mora, D. Rus, Planning and decision-making for autonomous vehicles. *Annu. Rev. Control Robot. Auton. Syst.* **1**, 187–210 (2018).
13. Y. Song, A. Romero, M. Muller, V. Koltun, D. Scaramuzza, Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Sci. Robot.* **8**, eadg1462 (2023).
14. P. Abbeel, A. Coates, M. Quigley, A. Ng, "An application of reinforcement learning to aerobatic helicopter flight" in *Advances in Neural Information Processing Systems 19*, B. Scholkopf, J. Platt, T. Hoffman, Eds. (MIT Press, 2007), vol. 19.
15. I. Lenz, H. Lee, A. Saxena, Deep learning for detecting robotic grasps, in *Proceedings of Robotics: Science and Systems* (RSS Foundation, 2013).
16. G. A. Castillo, B. Weng, W. Zhang, A. Hereid, Robust feedback motion policy design using reinforcement learning on a 3D digit bipedal robot, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2021), pp. 5136–5143.
17. M. Kearns, Y. Mansour, A. Y. Ng, A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Mach. Learn.* **49**, 193–208 (2002).
18. L. Kocsis, C. Szepesvari, Bandit based Monte-Carlo planning, in *Machine Learning: ECML 2006 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, J. Fürnkranz, T. Scheffer, M. Spiliopoulou, Eds., vol. 4212 of *Lecture Notes in Computer Science* (Springer, 2006), pp. 282–293.
19. R. Munos, From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Found. Trends Mach. Learn.* **7**, 1–129 (2014).
20. C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intel. AI Games* **4**, 1–43 (2012).
21. M. M. Flood, The traveling-salesman problem. *Oper. Res.* **4**, 61–75 (1956).
22. K. J. Astrom, R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers* (Princeton Univ. Press, 2008).
23. R. M. Murray, *Robotic Control and Nonholonomic Motion Planning* (University of California, 1991).
24. E. S. Lupu, F. Xie, J. A. Preiss, J. Alindogan, M. Anderson, S.-J. Chung, MAGIC^{VFM} -meta-learning adaptation for ground interaction control with visual foundation models. *IEEE Trans. Robot.*, 10.1109/TRO.2024.3475212 (2024).
25. NASA, "Asteroid redirect mission reference concept" (2015); https://nasa.gov/wp-content/uploads/2015/04/asteroid_redirect_mission_reference_concept_description_tagged.pdf.
26. R. W. Beard, T. W. McLain, *Small Unmanned Aircraft: Theory and Practice* (Princeton Univ. Press, 2012).
27. A. Couetoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, N. Bonnard, Continuous upper confidence trees, *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*, C. A. Coelho Coelho, Ed. (Springer, 2011), pp. 433–445.
28. T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, Y. Tassa, Predictive sampling: Real-time behaviour synthesis with MuJoCo. arXiv:2212.00541 [cs.RO] (2022).
29. M. ApS, MOSEK Fusion for C++ 10.1.21 (2019).
30. P. Foehn, A. Romero, D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight. *Sci. Robot.* **6**, eabh1221 (2021).
31. R. C. Foust, E. S. Lupu, Y. K. Nakka, S.-J. Chung, F. Y. Hadaegh, Autonomous in-orbit satellite assembly from a modular heterogeneous swarm. *Acta Astron.* **169**, 191–205 (2020).
32. A. Hereid, E. A. Cousineau, C. M. Hubicki, A. D. Ames, 3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics, *in 2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2016), pp. 1447–1454.
33. D. Morgan, G. P. Subramanian, S.-J. Chung, F. Y. Hadaegh, Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *Int. J. Robot. Res.* **35**, 1261–1285 (2016).
34. M. Schwager, D. Rus, J.-J. Slotine, Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *Int. J. Robot. Res.* **30**, 371–383 (2011).
35. D. Mayne, A second-order gradient method for determining optimal trajectories of nonlinear discrete-time systems. *Int. J. Control* **3**, 85–95 (1966).
36. H. Bock, M. Diehl, D. Leineweber, J. Schloder, "A direct multiple shooting method for real-time optimization of nonlinear DAE processes" in *Nonlinear Model Predictive Control*, F. Allgöwer, A. Zheng, Eds. (Springer, 2000), pp. 245–267.
37. W. Li, E. Todorov, Iterative linear quadratic regulator design for nonlinear biological movement systems, in *First International Conference on Informatics in Control, Automation and Robotics* (SciTePress, 2004), vol. 1, pp. 222–229.
38. Q. T. Dinh, M. Diehl, "Local convergence of sequential convex programming for nonconvex optimization" in *Recent Advances in Optimization and Its Applications in Engineering*, M. Diehl, F. Glineur, E. Jarlebring, W. Michiels, Eds. (Springer, 2010), pp. 93–102.
39. R. Bonalli, A. Cauligi, A. Bylard, M. Pavone, Gusto: Guaranteed sequential trajectory optimization via sequential convex programming, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 6741–6747.
40. I. A. Sucan, M. Moll, L. E. Kavraki, The open motion planning library. *IEEE Robot. Autom. Mag.* **19**, 72–82 (2012).
41. T. Marcucci, M. Petersen, D. von Wrangel, R. Tedrake, Motion planning around obstacles with convex optimization. *Sci. Robot.* **8**, eadf7843 (2023).
42. S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **30**, 846–894 (2011).
43. Y. Li, Z. Littlefield, K. E. Bekris, Asymptotically optimal sampling-based kinodynamic planning. *Int. J. Robot. Res.* **35**, 528–564 (2016).
44. E. Poccia, "Deterministic sampling-based algorithms for motion planning under differential constraints," thesis, Pisa Univ., Pisa, Italy (2017).
45. W. Honig, J. O. de Haro, M. Toussaint, db-a*: Discontinuity-bounded search for kinodynamic mobile robot motion planning, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2022), pp. 13540–13547.
46. E. Frazzoli, M. A. Dahleh, E. Feron, Real-time motion planning for agile autonomous vehicles. *J. Guidance Control Dyn.* **25**, 116–129 (2002).
47. E. Frazzoli, M. A. Dahleh, E. Feron, Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. Robotics* **21**, 1077–1091 (2005).
48. M. Saveriano, F. J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey. *Int. J. Robot. Res.* **42**, 1133–1184 (2021).
49. E. Schmerling, L. Janson, M. Pavone, Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics, in *2015 54th IEEE Conference on Decision and Control (CDC)* (IEEE, 2015), pp. 2574–2581.
50. R. Tedrake, "LQR-Trees: Feedback motion planning on sparse randomized trees" in *Robotics: Science and Systems* (RSS Foundation, 2009).
51. A. Majumdar, R. Tedrake, Funnel libraries for real-time robust feedback motion planning. *Int. J. Robot. Res.* **36**, 947–982 (2017).
52. S. Karaman, E. Frazzoli, Optimal kinodynamic motion planning using incremental sampling-based methods, in *49th IEEE Conference on Decision and Control (CDC)* (IEEE, 2010), pp. 7681–7687.
53. T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, *Model-based Reinforcement Learning: A Survey* (Foundations and Trends in Machine Learning, 2023), vol. 1, 16.

54. D. Auger, A. Couetoux, O. Teytaud, Continuous upper confidence trees with polynomial exploration–consistency, in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, H. Blockeel, K. Kersting, S. Nijssen, F. Železný, Eds. (Springer, 2013), pp. 194–209.
55. D. Silver, J. Veness, Monte-Carlo planning in large POMDPS, in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta, Eds. (Curran Associates, 2010), vol. 23.
56. Z. Sunberg, M. Kochenderfer, Online algorithms for POMDPS with continuous state, action, and observation spaces, in *Proceedings of the International Conference on Automated Planning and Scheduling* (AAAI, 2018), vol. 28, pp. 259–263.
57. J. Ragan, B. Riviere, S.-J. Chung, Bayesian active sensing for fault estimation with belief space tree search, paper presented at AIAA SCITECH 2023 Forum, National Harbor, MD, 23 to 27 January 2023.
58. V. Lisy, V. Kovarik, M. Lanctot, B. Bosansky, Convergence of Monte Carlo tree search in simultaneous move games, in *Advances in Neural Information Processing Systems 26,* C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Q. Weinberger, Eds. (Curran Associates, 2013), vol. 26.
59. M. Kobilarov, Cross-entropy motion planning. *Int. J. Robot. Res.* **31**, 855–871 (2012).
60. G. Williams, P. Drews, B. Goldfain, J. M. Rehg, E. A. Theodorou, Aggressive driving with model predictive path integral control, in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2016), pp. 1433–1440.
61. M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, B. Boots, Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation, in *Conference on Robot Learning* (MLResearchPress, 2022), pp. 750–759.
62. C. Pezzato, C. Salmi, M. Spahn, E. Trevisan, J. Alonso-Mora, C. H. Corbato, Sampling-based model predictive control leveraging parallelizable physics simulations. arXiv:2307.09105 [cs.RO] (2023).
63. N. P. Garg, D. Hsu, W. S. Lee, "Despot-alpha: Online pomdp planning with large state and observation spaces" in *Robotics: Science and Systems* (RSS Foundation, 2019), vol. 3, pp. 3–2.
64. Y. K. Nakka, S.-J. Chung, Trajectory optimization of chance-constrained nonlinear stochastic systems for motion planning under uncertainty. *IEEE Trans. Robot.* **39**, 203–222 (2022).
65. R. S. Sutton, Planning by incremental dynamic programming, in *Machine Learning Proceedings* (Elsevier, 1991), pp. 353–357.
66. C.-S. Chow, J. N. Tsitsiklis, An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Trans. Automat. Contr.* **36**, 898–914 (1991).
67. R. Munos, A. Moore, Variable resolution discretization in optimal control. *Mach. Learn.* **49**, 291–323 (2002).
68. A. Gorodetsky, S. Karaman, Y. Marzouk, High-dimensional stochastic optimal control using continuous tensor decompositions. *Int. J. Robot. Res.* **37**, 340–377 (2018).
69. A. S. Polydoros, L. Nalpantidis, Survey of model-based reinforcement learning: Applications on robotics. *J. Intel. Robot. Syst.* **86**, 153–173 (2017).
70. K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **34**, 26–38 (2017).
71. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. arXiv:1707.06347 [cs.LG] (2017).
72. M. Fazel, R. Ge, S. Kakade, M. Mesbahi, Global convergence of policy gradient methods for the linear quadratic regulator, in *International Conference on Machine Learning* (MLResearchPress, 2018), pp. 1467–1476.
73. K. Zhang, A. Koppel, H. Zhu, T. Basar, Global convergence of policy gradient methods to (almost) locally optimal policies. *SIAM J. Control Optimization* **58**, 3586–3612 (2020).
74. J. Bhandari, D. Russo, Global optimality guarantees for policy gradient methods. *Oper. Res.* **72**, 1906–1927 (2024).
75. E. Schmerling, K. Leung, W. Vollprecht, M. Pavone, Multimodal probabilistic model-based planning for human-robot interaction, in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 3399–3406.
76. B. Kim, K. Lee, S. Lim, L. Kaelbling, T. Lozano-Perez, Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2020), vol. 34, pp. 9916–9924.
77. G. Williams, A. Aldrich, E. A. Theodorou, Model predictive path integral control: From theory to parallel computation. *J. Guidance Control Dyn.* **40**, 344–357 (2017).
78. R. S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **112**, 181–211 (1999).
79. Y. Lee, P. Cai, D. Hsu, Magic: Learning macro-actions for online POMDP planning, in *Proceedings of Robotics: Science and Systems* (RSS Foundation, 2021).
80. A. Bai, S. Srivastava, S. Russell, Markovian state and action abstractions for MDPS via hierarchical MCTS, in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, S. Kambhampati, Ed. (IJCAI/AAAI Press, 2016), pp. 3029–3039.
81. M. De Waard, D. M. Roijers, S. C. Bakkes, Monte Carlo tree search with options for general video game playing, in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2016), pp. 1–8.
82. A. Jamgochian, H. Buurmeijer, K. H. Wray, A. Corso, M. J. Kochenderfer, Constrained hierarchical Monte Carlo belief-state planning. arXiv:2310.20054 [cs.AI] (2023).
83. R. S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen, K. Müller, Eds. (MIT Press, 1999).
84. M. Deisenroth, C. E. Rasmussen, Pilco: A model-based and data-efficient approach to policy search, in *Proceedings of the 28th International Conference on Machine Learning (ICML11)* (Association for Computing Machinery, 2011), pp. 465–472.
85. S. Levine, V. Koltun, Guided policy search, in *International Conference on Machine Learning* (MLResearchPress, 2013), pp. 1–9.
86. B. Riviere, W. Honig, Y. Yue, S.-J. Chung, Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. *IEEE Robot. Autom. Lett.* **5**, 4249–4256 (2020).
87. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, 2014).
88. D. Shah, Q. Xie, Z. Xu, Non-asymptotic analysis of Monte Carlo tree search, in *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems* (Association for Computing Machinery, 2020), pp. 31–32.
89. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
90. S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory* (SIAM, 1994).
91. K. Zhou, J. C. Doyle, *Essentials of Robust Control* (Prentice Hall, 1998), vol. 104.
92. R. Tedrake, Drake Development Team, Drake: Model-based design and verification for robotics (2019); https://drake.mit.edu.
93. B. Riviere, W. Honig, M. Anderson, S.-J. Chung, Neural tree expansion for multi-robot planning in non-cooperative environments. *IEEE Robot. Autom. Lett.* **6**, 6868–6875 (2021).
94. J. Ragan, B. Riviere, F. Hadaegh, S.-J. Chung, Online tree-based planning for active spacecraft fault estimation and collision avoidance. *Sci. Robot.* **9**, eadn4722 (2024).

Acknowledgments: We thank the DARPA LINC team at Caltech and JPL, who contributed to the full autonomy stack of the results shown in Fig. 3. We thank J. Preiss (planning), S. Lupu (control), and M. Anderson (ROS/software) for support and collaboration on the tracked vehicle experiment. Other LINC team members include J. Burdick, Y. Yue, A. Rahmani, L. Gan (localization), F. Xie (control), J. Becker (segmentation and mapping), T. Touma, and J. Alindogan (experiment). We thank J. Cho for support and collaboration on the spacecraft experiment, H. Tsukamoto for discussions on discrete contraction, F. Hadaegh for discussions on space autonomy, and the Sandia National Labs team (T. Blada, E. Lu, and D. Wood) for LINC testing support (Fig. 3). The drone experiment was conducted at Caltech's Center for Autonomous Systems and Technologies. **Funding:** We acknowledge the funding support of DARPA LINC (J. F. Mergen), the Aerospace Corporation (J. Brader and B. Bycroft), and Supernal (R. Stefanescu and H. Park). This material is based on work supported by the NSF Graduate Research Fellowship Program under grant no. 2139433. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. **Author contributions:** The algorithm conceptualization, theory, and software implementation were equally developed by B.R. and J.L. under the guidance and critical reviews of S.-J.C. The quadrotor and spacecraft experiments were equally designed by B.R. and J.L. Both B.R. and J.L. contributed to the tracked vehicle experiment and the glider experiment, with J.L. leading the tracked vehicle experiment and B.R. leading the glider experiment. The manuscript and figures were equally developed by B.R. and J.L. with feedback and multiple iterations with S.-J.C. All authors reviewed the manuscript. S.-J.C. supervised the research. **Competing interests:** California Institute of Technology filed a US nonprovisional patent application on this work on 5 August 2024. **Data and materials availability:** All data and code used to produce the plots presented here and in the Supplementary Materials are available on the online repository Dryad: doi:10.5061/dryad.s7h44j1h5. Our code is also available at https://github.com/aerorobotics/sets.

Submitted 18 January 2024
Accepted 6 November 2024
Published 4 December 2024
10.1126/scirobotics.ado1010

Science Robotics

Monte Carlo tree search with spectral expansion for planning with dynamical systems

Benjamin Rivière, John Lathrop, and Soon-Jo Chung

Sci. Robot. **9** (97), eado1010. DOI: 10.1126/scirobotics.ado1010

Editor's summary

Autonomous robots require effective decision-making processes to adapt to complex new environments. Monte Carlo tree search (MCTS) is a planning algorithm that uses real-time computation to strategically explore future decisions, but it cannot be directly applied to generate physical motions for robots. Rivière *et al*. have now developed Spectral Expansion Tree Search that enables real-time MCTS-based planning by computing an efficient discrete representation of the physical world. The framework was deployed on various robots that were shown to be capable of autonomously discovering optimal trajectories to avoid dynamic obstacles, traverse windy gusts, support a human driver in shared control tasks, and capture and redirect an uncontrolled agent. —Amos Matsiko