

Utilizing Reinforcement Learning to Continuously Improve a Primitive-Based Motion Planner

Zachary C. Goddard*, Kenneth Wardlaw[†], Rohith Krishnan[‡], and Panagiotis Tsiotras[§]
Georgia Institute of Technology, Atlanta, GA, 30332, USA

Michael R. Smith[¶], Mary Rose Sena^{||}, and Julie Parish**
Sandia National Laboratories, Albuquerque, NM, 87123, USA

Anirban Mazumdar^{††}
Georgia Institute of Technology, Atlanta, GA, 30332, USA

This paper describes how the performance of motion primitive based planning algorithms can be improved using reinforcement learning. Specifically, we describe and evaluate a framework for policy improvement via the discovery of new motion primitives. Our approach combines the predictable behavior of deterministic planning methods with the exploration capability of reinforcement learning. The framework consists of three phases: evaluation, exploration, and extraction. This framework can be iterated continuously to provide successive improvement. The evaluation step scores the performance of a motion primitive library using value iteration to create a cost map. A local difference metric is then used to identify regions that need improvement. The exploration step utilizes reinforcement learning to examine new trajectories in the regions of greatest need. The extraction step encodes the agent’s experiences into new primitives. The framework is tested on a point-to-point navigation task using a 6-DOF nonlinear F-16 model. One iteration of the framework discovered 17 new primitives and provided a maximum planning time reduction of 96.91%. After 3 full iterations, 123 primitives were added with a maximum time reduction of 97.39%. The proposed framework is easily extensible to a range of vehicles, environments, and cost functions.

I. INTRODUCTION

PRIMITIVE-BASED planning offers fast and powerful means of generating feasible trajectories for complex dynamic systems. These methods can enable rapid kinodynamic motion plans that leverage the full performance envelope of the physical system [1, 2]. Motion primitives are repeatable subtrajectories that can be concatenated to form full motion plans. Primitives can include a range of behaviors including aggressive maneuvers.

A key challenge with primitive-based motion planning is the creation of the primitive library. Current methods include emulating expert pilots [3], optimal control [1], and utilizing an existing control system [4]. However, these methods often require human expert involvement to either a) generate the trajectory, b) formulate the optimal control problem, or c) design the control system. These methods can make the primitive generation process time consuming. In addition, the involvement of human experts creates potential variability and biases. Similarly, access to expert knowledge may be limited or prior knowledge may not exist for a novel platform. Advancements in machine learning can improve the process of creating primitive libraries. Reinforcement learning has shown the potential to develop novel

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND2020-13089 C

*Graduate Student, Woodruff School of Mechanical Engineering, AIAA Student Member.

[†]Undergraduate Student, Woodruff School of Mechanical Engineering.

[‡]Undergraduate Student, School of Computer Science.

[§]Professor, Guggenheim School of Aerospace Engineering, AIAA Fellow.

[¶]Technical Staff, Mission Analytics, AIAA Member.

^{||}Technical Staff, Mission Analytics, AIAA Member.

**Pathfinder Technologies, Mission Analytics, AIAA Senior Member.

^{††}Assistant Professor, Woodruff School of Mechanical Engineering, AIAA Member.

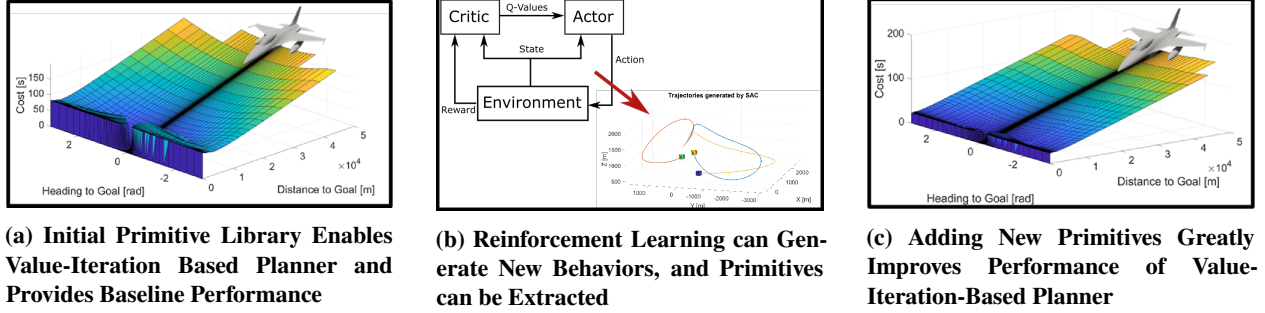


Fig. 1 Overview of the framework and illustration of improvement made by adding new primitives.

strategies and methods without human involvement [5]. For example, machine learning was used to generate a large library of aggressive maneuvers for a quadrotor [6]. However, this method focused only on maneuvers, used a relatively restrictive definition of maneuvers, and did not use feedback to provide tailored improvements.

This paper provides a new way to utilize reinforcement learning for improving the performance of motion primitive-based planners. The core contribution of this work is the development and quantitative evaluation of an algorithmic framework that uses motion primitives and combines the predictable behavior of deterministic planning algorithms with the exploration of reinforcement learning. Specifically, we show how a soft actor-critic algorithm can generate new trim states and maneuvers simply by attempting to minimize a reward function. These new primitives can dramatically increase the performance of a value-iteration based planner. Key aspects of this work include the description of the overall framework, identification of a metric for estimating library deficiency, the development of shaping rewards, creation of a primitive extraction algorithm, and quantitative analysis of overall performance. The framework is illustrated in Figure 1. The proposed framework can be extended to a range of deterministic planners and a variety of reinforcement learning methods.

This paper starts with an overview of past work in motion planning and motivates the value of combining deterministic and learning-based techniques. We then describe the requirements for our planning framework. We describe our planning framework in detail and illustrate its performance with a simulated nonlinear 6-DOF F-16 model. We conclude with a discussion of the results which illustrate the overall performance.

II. PRIOR WORK

The Maneuver Automaton (MA) is a particular architecture that defines motion primitives for mobile vehicles [7]. A MA classifies primitives as trim states, which are equilibrium points in the vehicle dynamics, and maneuvers that connect any two trim states. The MA also describes a symmetry group, $\mathcal{H}$, which is defined as components of the state that do not influence the vehicle dynamics. For example, lateral translation or heading rotation may not affect forces and moments experienced by certain vehicles, such as air and watercraft. Motion plans are constructed by choosing coasting times in each trim state and the maneuver to execute afterwards. Several methods for obstacle-free path planning have been built on top of the MA, including precomputing a cost table through value iteration then solving with a greedy policy [7] and directly solving for an explicit solution [1]. Both methods find solutions that are optimal on the available primitives, but not necessarily optimal in the continuous state space.

Later works further built on the MA by adding robustness to uncertainty and employing primitives to steer sampling based planning methods. Other work added pre-computed funnels to characterize uncertainty in a motion plan constructed from primitives [8]. The work also demonstrates the use of primitives for rapid and safe planning of nonlinear systems. The previously mentioned obstacle-free planning methods can be extended with sampling based planning to navigate around static and dynamic obstacles [9]. Together these extensions improve the usability of primitives in real-time aggressive motion planning. Other work has demonstrated how a few carefully designed primitives may be used to great effect for navigating dense fields of obstacles [2]. Frazzoli used carefully designed primitives for fast turn-arounds to improve performance of a helicopter library [1]. All previous works depend on expert knowledge to design motion primitives for a given task. If expert knowledge about the vehicle or the environment is lacking, a library of primitives may fail to characterize useful trajectories.

Expert knowledge may be supplemented by modern reinforcement learning techniques to fill a MA with useful motion primitives. Recent algorithms, such as Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG

(TD3), and Soft Actor Critic (SAC), have shown success in continuous systems, allowing their applications in controls problems [10–12]. DDPG uses a neural network based actor and critic to learn a policy and value function, respectively. The value function is learned from experience and used to optimize the policy. TD3 and SAC both extend DDPG with improvements for learning stability and exploration. Each of these methods have shown success in continuous state and action spaces, making them candidates for learning a vehicle control policy.

DDPG has already been applied to motion planning problems to learn short point-to-point navigation policies in PRM-RL[13]. PRM-RL successfully learned control policies for a ground robot and a quadcopter with a payload, demonstrating the power of deep learning methods when paired with other planning methods.

One common issue with deep learning methods is the lack of transparency and guarantees in the policies they produce. The final policy may not be stable for all condition, so it is undesirable to be used directly in tasks where robustness and safety are of high importance. An alternative approach may be to encode useful regions of the control policy into an analytic solution.

Reinforcement learning has been leveraged to learn quadcopter primitives in [6], which demonstrated learned improvement over state-of-the-art planning methods. This approach is limited in its definition of primitives, which are represented as Bézier curves in the control space. Unlike the MA, the formulation in [6] does not store information on the reference trajectories through the state space. Additionally, Bézier curves may not be applicable to other classes of vehicles, such as fixed-wing aircraft which may be unable to track an arbitrary curve.

We build on the past works in motion primitives and reinforcement learning by creating a new framework. Our new approach lends more flexibility to the reinforcement learning agent in terms of exploring the state space and creating new trims and more free-form maneuvers. Additionally our approach reduces expert involvement with a fully autonomous process for primitive generation and planning improvement.

III. LEARNING FRAMEWORK REQUIREMENTS

We seek a learning framework that is capable of improving performance autonomously without human intervention. This framework should be able to store its experience transparently and integrate with planners to create motion plans rapidly and predictably. In addition, the framework should be generalizable to a range of dynamic systems and environments. We summarize our desired requirements below:

- 1) **Full Autonomy:** The framework should be able to learn new primitives without human input. Human expertise may be unavailable for a task that has not been thoroughly explored already. Avoiding human input prevents bias and allows the framework to learn primitives that are not necessarily obvious to a human expert. These new behaviors may perform better than those generated by expert knowledge.
- 2) **Transparency of Learned Policy:** Encoded data must be clear to human observers. The behaviors learned should be visualizable to provide intuition on why they improve the policy. They should also be dynamically characterized so control may be designed in post process to establish guarantees on stability.
- 3) **Learning Stability:** Some learning algorithms are subject to instability and may get progressively worse or forget important old behaviors as they gain new experience. This framework must store experience in a way that preserves performance and prevents old behaviors from being overwritten by new behaviors.
- 4) **Vehicle and Environment Modularity:** Vehicles may vary significantly in their dynamics and operation. This framework should be configurable to multiple classes of vehicles (e.g., cars, rotorcraft, fixed-wing aircraft, etc.) without changes to the foundational architecture. The learning agent should be able to handle new dynamics without explicit human input. This should also hold true when the vehicle remains the same, but is given a new task or placed in a different environment.
- 5) **Interchangeable Cost Function:** Different scenarios may require that an agent prioritize different objective functions. Each step of our process needs to be able to reconfigure to different cost functions without changing the overall architecture. Further, learning a new cost function must be done without losing information on a previous cost function.
- 6) **Improvement with Minimal Additions:** New behaviors should only be accepted if they provide some benefit. Excessive additions may slow down planning algorithms that use the behaviors found and could make it difficult to find identify important strategies discovered.

Our review of existing solutions did not find a learning architecture for mobile vehicle tasks that accomplishes all of the above. Therefore, we propose a new framework that consists of three phases: evaluation, exploration, and extraction.

This framework can provide autonomous improvement of the MA by iterating through the three steps several times. The evaluation step scores the performance of a motion primitive library using value iteration to create a cost map. A

local difference metric is then used to identify regions that need improvement. The exploration step utilizes modern reinforcement learning to examine new trajectories in the regions of the greatest need. The extraction step encodes the machine learning agent’s experiences into new primitives.

IV. FRAMEWORK OVERVIEW

This section presents the framework for the guided discovery of new motion primitives. The framework consists of three phases: evaluation of the current MA, exploration of high cost states, and extraction of new primitives. We have made new contributions to each of these three parts, which enables our methodology to function as a single framework. These are 1) identification of a metric for estimating library deficiency, 2) outlining new shaping reward functions to help maintain completeness for the primitive graph, and 3) development of a primitive extraction algorithm for general vehicle trajectories.

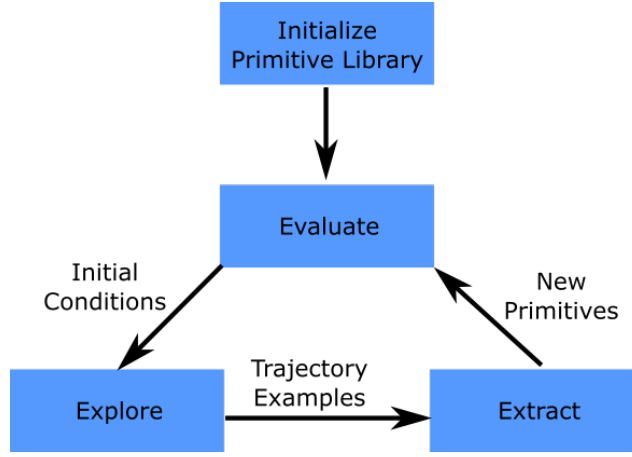


Fig. 2 Diagram of full framework process.

A. Evaluation

The initial evaluation phase begins with a preliminary primitive library. This can be very sparse and include only a few simple behaviors. If the initial trim primitives are chosen such that they form a complete basis in the symmetry group, the MA will be controllable [1]. Successive iterations will use the primitive library that was augmented at the end of the previous iteration. This work uses value iteration as described in [7], because it provides a comprehensive cost function across the symmetry space. The output of value iteration is a cost map, $J : Q_T \times \mathcal{H} \rightarrow \mathbb{R}$ across the continuous symmetry space, $\mathcal{H}$, and the MA trim states, Q_T , as shown in Figure 3. The symmetry space used in this example is the distance and relative heading to the goal. The map gives the cost to go to the goal when starting from any point in the symmetry space and any trim state in the MA. Value iteration may be generalized to different objective functions allowing it to evaluate a MA on a variety of criteria. In this work we focus on minimum time trajectories, so the cost, J , is the time to reach the goal. The initial cost map, J_0 , for obstacle-free point-to-point navigation is shown in Figure 3. Note how the cost is high with increasing distance and high near the goal.

The cost map can be used to improve the performance of the planning algorithm by highlighting cases of poor performance. Regions of the cost map can be used to create the initial conditions for the exploration step. Specifically, performance can be improved by adding new maneuvers or trims to the MA library. While it may seem intuitive to examine areas of highest cost, this is not necessarily appropriate. Specifically, high costs may stem from challenging initial conditions rather than library deficiencies. Therefore, a core contributions of this work is the generation of a methodology for identifying regions where poor performance is caused by "library deficiency."

We estimate library deficiency by comparing the cost of a state relative to surrounding points in the symmetry space, since a highly maneuverable vehicle should be able to reach a nearby position with relatively low cost. If the available primitives do not capture the vehicle’s maneuverability, there may be a significant difference between two close points, since the vehicle may not be able to easily reach one from the other. This work proposes using the maximum difference between each point and uniformly sampled surrounding points to determine the best candidates for

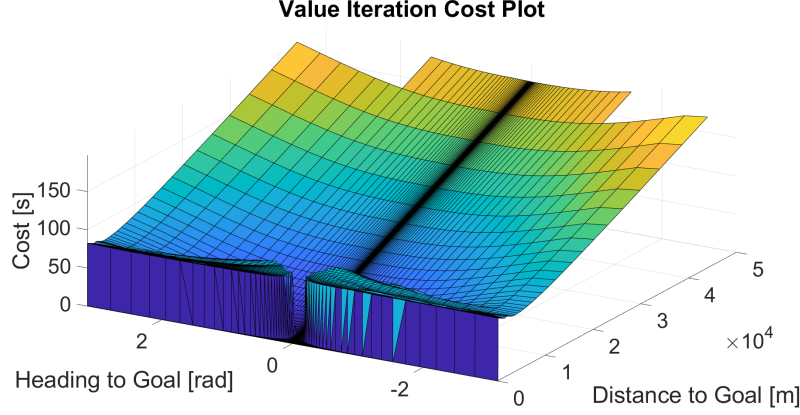


Fig. 3 Map of time-to-go for distance and direction of goal relative to vehicle shown for a single trim state. The sharp ridges on the map illustrate regions where the current policy does poorly.

improvement. Weights for sampling initial conditions are calculated as follows:

$$W(\mathbf{h}) = \max_{i,j} J(\mathbf{h}) - J\left(\mathbf{h} + \begin{bmatrix} \Delta h_{1i} & \Delta h_{2j} \end{bmatrix}^T\right) \quad (1)$$

$$\forall i, j = \{-n, 1-n, \dots, 0, \dots, n\}$$

Where J is the cost map produced by value iteration, and Δh_1 and Δh_2 are given step sizes for each dimension of the symmetry space. The weights produced from the cost map in Figure 3 can be seen in Figure 4. These weights are normalized into a probability distribution which is used to sample initial conditions during the exploration phase.

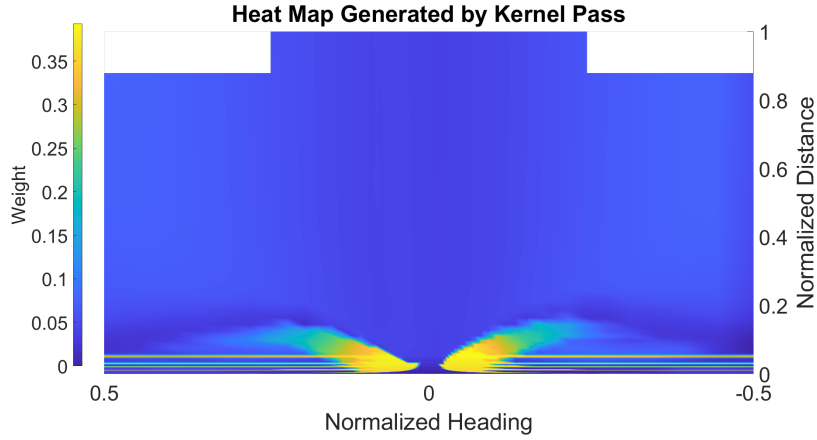


Fig. 4 Map of weights calculated from Equation 1. Distance and heading were normalized by their respective ranges before calculating.

B. Exploration

The objective of the exploration phase is to learn a new policy that outperforms the MA in the regions identified in the evaluation phase as initial conditions. Reinforcement learning is an ideal tool for this exploration process because it provides a means to autonomously generate and improve a policy for a wide range of tasks. This work investigates the use of reinforcement learning agents for this step. The agent is initialized in a trim state and at a point in the symmetry space chosen probabilistically based on the weights found during evaluation. This encourages learning to focus on regions where the MA performs poorly due to a deficient selection of primitives. We choose to use SAC because it can

handle continuous state and action spaces, and it has shown promise for high degree of freedom control tasks, such as quadrupedal locomotion [14]. DDPG and TD3 are also potential alternatives. All three of these use an actor-critic model [15], as illustrated in Figure 5.

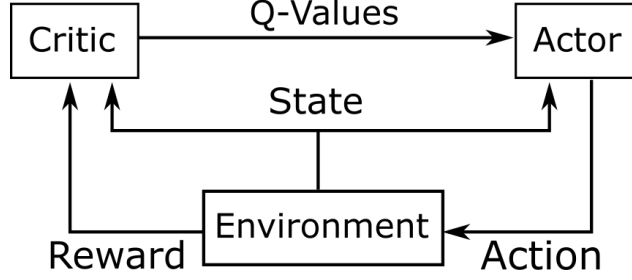


Fig. 5 Generic structure of an actor-critic model. The critic learns Q-values from the environment, which the actor uses to improve its policy.

The learning agent also requires a reward function to define its objective. The simplest reward function would be some positive value at the goal and zero everywhere else. Since reinforcement learning agents often struggle with sparse returns, some shaping rewards are added to help guide the agent as shown in Equations (2)-(5). These shaping rewards are tailored to our unique framework and represent a key contribution of this work.

$$R_{\text{obs}}(x') = \begin{cases} -k_{\text{obs}} & , \text{ if } x' \in O \\ 0 & , \text{ otherwise} \end{cases} \quad (2)$$

$$R_{\text{dist}}(x, x') = k_{\text{dist}} (d(x, x_g) - d(x', x_g)) \quad (3)$$

$$\tilde{R}_{\text{trim}}(x) = \max_q \frac{1}{1 + d(x, x_q)}, \forall q \in Q_T \quad (4)$$

$$R_{\text{trim}}(x, x') = k_{\text{trim}} (\tilde{R}_{\text{trim}}(x) - \tilde{R}_{\text{trim}}(x')) \quad (5)$$

Where k_{obs} , k_{dist} , and k_{trim} are constants to weight the importance of each reward, O is the obstacle or out-of-bounds space, x_g is the goal state and $d(., .)$ is a distance function between two states. For the distance function, this work uses square Euclidean distance.

The first reward penalizes the agent for going out of bounds or contacting an obstacle if there are any. The second is a positional reward that gives some reward or penalty at every time step if the agent gets closer or further from the goal, respectively. The positional reward helps eliminate problems with sparse rewards by gently guiding the agent to the primary reward at the goal state.

The last function is a trim reward which penalizes the agent for moving away from trim states currently in the MA. The trim reward helps encourage the agent to return to known trim states, so that the extraction phase produces a connected graph. Without this reward the agent may discover new primitives, but not connections between them and the existing ones. One drawback is that this reward may encourage the agent to prioritize staying in a trim state over taking an action to get to the goal. The reward is formulated such that the maximum cumulative penalty is k_{trim} . To ensure the agent prioritizes going to the goal over staying in a trim state, k_{trim} is set to be much smaller than the reward for reaching the goal.

Further shaping rewards can be easily added to prioritize other cost functions. Additional rewards should be analogous to the cost function used during value iteration. For example, if the cost function used for value iteration was change in total energy, a corresponding reward function may add a negative value proportional to the energy loss at every time step. This allows for easy interchangeability of different objective functions for the same vehicle and environment.

During and after training, some episodes are recorded as a sample trajectory of the vehicle state and control inputs. While recording a sample trajectory, SAC switches from a stochastic policy to a deterministic policy which always outputs the best control values. The recorded trajectories are passed to the extraction step which searches for subtrajectories that meet the definition of a motion primitive for the MA. Episodes are recorded throughout training since a trajectory that does not reach the goal may still contain a primitive that adds value to the MA. Examples of the trajectories generated by the SAC are shown in Figure 6.

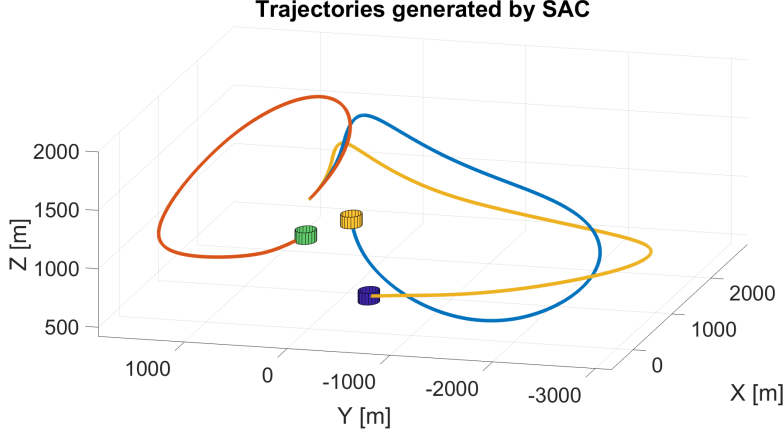


Fig. 6 Trajectories generated by the SAC. The cylinders indicate the goals for each episode.

C. Extraction

The extraction phase searches for subtrajectories that are either a trim state or bounded by two trim states. We propose an algorithm for breaking down trajectories into primitives that fit the MA definition. The algorithm consists of two passes: one adds new trim states and a second adds new maneuvers.

Algorithm 1 ADD NEW TRIMS

Inputs: A state trajectory, $X = \{x_1, \dots, x_T\}$, a state time derivative trajectory, $\dot{X} = \{\dot{x}_1, \dots, \dot{x}_T\}$, threshold values, r_{eq}, r_{trim} .

for $0 < i \leq T$ **do**

if $|\dot{x}_i| < r_{eq}$ AND $d(x_i, x_q) < r_{trim} \forall q \in Q_T$ **then**

$q_{new} \leftarrow NEW-TRIM(x_i)$

$Q_T = Q_T \cup q_{new}$

end if

end for

The first pass looks at the derivative of the state at every point in the trajectory. Points where $\dot{x}$ fall within a threshold centered on zero are identified and compared to existing trim states. If the point falls within a given margin of an existing trim state, it is ignored to prevent dense trim states that add little value. Otherwise, the point is added into the library as a new trim state. Additionally the control input at that point is recorded as equilibrium input, so a controller may be designed around the point in post process.

The second pass uses the known trim states to identify points that are within the maneuver start set, C_q , defined in [16]. Points which fall within this set for any trim state are labeled by the respective primitive:

$$label(x) = \begin{cases} \arg \min_{q \in Q_x} d(x, x_q), & \text{if } Q_x \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where,

$$Q_x = \{q \mid q \in Q_T \text{ and } x \in C_q\} \quad (7)$$

The trajectory is then broken up into segments with contiguous labels. Segments that did not match any known trim state, but start and end in a known trim state are identified as new maneuvers as shown in Figure 7. Algorithm 2 describes the process of finding segments that match maneuvers for the MA.

Some maneuvers are pruned if they do not contribute anything to the MA. Each maneuver is temporarily added to the library of primitives and evaluated with a simplified version of value iteration. Specifically, new costs are only calculated for the trim state from which the new maneuver initializes. If no improvement is made after one iteration, the maneuver will never contribute any improvement to the MA with the given cost function and can be pruned. Algorithm

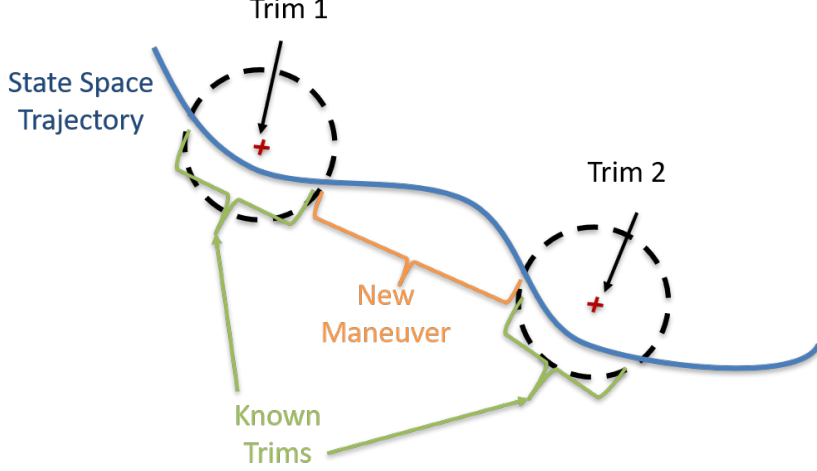


Fig. 7 Illustration of primitive extraction. Segments that start and end within the boundaries around a known trim state are extracted as reference trajectories for a new maneuver.

3 describes the pruning process. Pruning may not be desirable where multiple mission types (cost functions) are envisioned. However, the size of the library can increase run times.

V. EXPERIMENT DESIGN



Fig. 8 Visual output of the FlightGear simulator used for experimental evaluation.

The described framework was tested in a simulation environment using a model of an F-16 aircraft [17]. The F-16 model is nonlinear, 6-DOF, validated in a wind tunnel, and widely used in publication. We have developed software tools that combine Python, the F-16 model, and FlightGear. This provides seamless combination with machine learning packages and valuable visualization of aircraft trajectories. The task for this environment was point-to-point navigation with the only obstacle space being the ground.

An initial library of conservative primitives was generated manually and evaluated using value iteration. The library consists of the following trim states: straight and level flight, ascending and descending flight, and a left and right steady turn. The primitives also include transition maneuvers connecting the straight and level trim state to and from all other trim states. This set of primitives allows for controllability in the symmetry space but has limited agility.

Evaluation of the primitive libraries was done using value iteration with the symmetry space being the relative distance and heading to the goal. The cost function was the time to reach the goal.

Exploration was done using a Soft Actor-Critic model with the following input vector:

$$s = [\rho_g, \psi_g, z_g, h, M, \alpha, \beta, p, q, r, \phi, \theta, \dot{\phi}, \dot{\theta}, \dot{\psi}]$$

Algorithm 2 FIND MANEUVERS

Inputs: A state trajectory, $X = \{x_1, \dots, x_T\}$, where $label(x_1)$ and $label(x_T)$ are not 0, and a control signal, $U = \{u_1, \dots, u_T\}$

$i \leftarrow 0$
 $M \leftarrow \{\}$
while $i \leq T$ **do**
 while $label(x_i) \neq 0$ **and** $i \leq T$ **do**
 $i \leftarrow i + 1$
 end while
 $m \leftarrow NEW-MANEUVER()$
 $m.q_{init} \leftarrow label(x_{i-1})$
 $m.x \leftarrow []$
 $m.u \leftarrow []$
 while $label(x_i) = 0$ **and** $i \leq T$ **do**
 $m.x.Append(x_i)$
 $m.u.Append(u_i)$
 end while
 $m.q_{final} \leftarrow label(x_i)$
 $M \leftarrow M \cup m$
end while

Algorithm 3 PRUNE MANEUVERS

Inputs: A set of candidate maneuvers M and an initial cost map J_0

for $m \in M$ **do**
 $J \leftarrow J_0$
 $q_0 \leftarrow m.q_{init}$
 for $h \in H$ **do**
 $J_{new} \leftarrow \min_{\tau} \gamma_{q_0} \tau + \Gamma_m + J(\Phi_{\tau,m}(q_0, h))$
 $J(q_0, h) \leftarrow \min(J_{new}, J(q_0, h))$
 end for
 if $\sum_h J(q_0, h) - J_0(q_0, h) > 0$ **then**
 $Q_M \leftarrow Q_M \cup m$
 end if
end for

Where ρ_g , ψ_g , and z_g are the relative lateral distance, heading, and altitude to the goal, respectively, h is altitude, M is Mach number, α is angle of attack (AoA), β is side slip, p , q , and r are body angular rates, ϕ and θ are attitude angles, and $\dot{p}$, $\dot{\theta}$, and $\dot{\psi}$ are attitude rates. Mach number, AoA, and side slip are used to keep the inputs on a similar order of magnitude, and for similar reasons, ρ_g , z_g , and h are input in units of kilometers. We acknowledge that attitude rates are redundant, given body rates and attitude angles; however, they are included to simplify the information the neural networks need to encode. Outputs for the actor are angular rate commands. The throttle command to the F-16 is fixed.

The neural networks for both the SAC actor and critics use ReLU activations for hidden units. The critics are two layer networks with 400 and 300 hidden units, respectively. The actor uses two shared layers and two layers for the mean and standard deviation outputs. The shared layers have 400 and 300 units and the separate layers have 64 and 32 units. Output activations for the actor are tanh functions. When generating trajectories for primitive extraction the actor only uses the deterministic output. The actor also uses automatic temperature adjustment.

New primitives are added to the library for several iterations of evaluation, exploration, and extraction, until the evaluation phase stops seeing significant improvement. Data is collected in the form of all primitives discovered and the MA scores from the evaluation phase at every iteration.

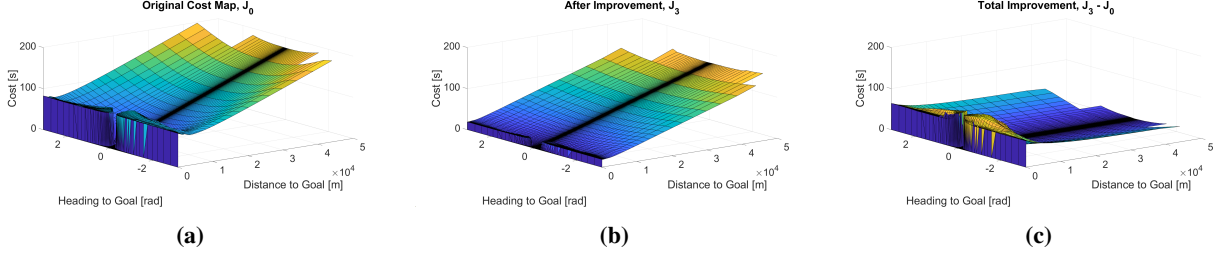


Fig. 9 (a) Original cost map. (b) Cost map after a full iteration of exploration, extraction, and re-evaluation. (c) Difference between the original and improved maps.

VI. DATA ANALYSIS

This section reports the overall results on the MA improvement and quantities of new primitives discovered. Ideally, this method should not just add new primitives to the MA, but specifically add primitives that result in significant improvement. Maneuvers that are eclipsed in functionality by an existing maneuver will not only see no improvement, but may slow down planning algorithms that have to search through all maneuvers in the library.

We begin by describing the first full iteration of our framework. Primitives were extracted from 100 trajectories generated by the exploration agent which ranged from 10 to 100 seconds depending on how long the agent took to get to the goal given the starting position. From this sample 68 subtrajectories were found to fit the MA's definition of a primitive in the first iteration. These were further pruned to 17 primitives which added unique improvements to the existing library. The process was repeated for two more iterations and added 123 primitives overall.

The result of re-evaluating the MA with these primitives through value iteration is shown in Figure 9. Regions where the heading was close to 0 see little to no improvement, since they were likely restricted only by the vehicle's speed. Some visible improvement occurred where heading was larger, indicating the new primitives included faster turning maneuvers. Significant improvement is also seen close to the goal where the vehicle is not facing the goal. This also indicates that new primitives are taking advantage of more of the vehicle's maneuverability.

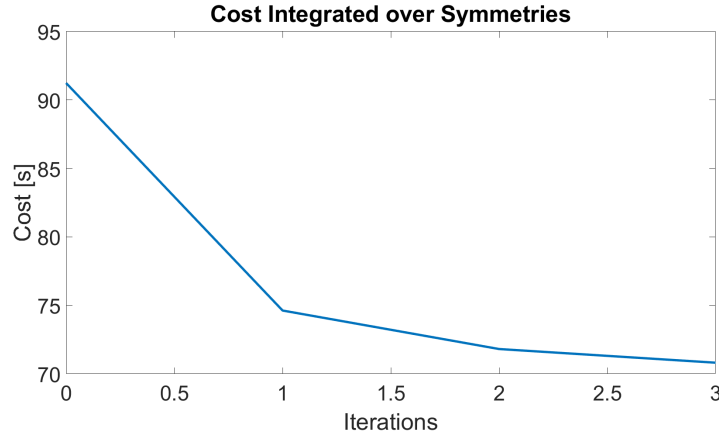


Fig. 10 Value of the integrated cost map for each iteration.

The improvement was calculated as the difference between the original and the new cost map. The maximum improvement, or greatest difference between a point on the final cost map and the same point on initial cost map, was 97.1 sec. The highest relative improvement was 96.91% of the original value. This shows the process has the capacity to significantly improve a MA's ability to complete a task from some initial conditions.

Once the full iteration is complete, we quantify a single cost associated with the planner. The improvement is integrated over the symmetry space to measure the overall improvement between each iteration, as shown in Figure 10. Before integrating, the symmetries are normalized by the range used in the cost map. The results show a significant improvement after the first iteration and smaller improvements for the following evaluations. The framework can be halted once the overall cost no longer decreases substantially. In this case, iteration 3 was within 2% of iteration 2.

Therefore, the framework was halted after iteration 3.

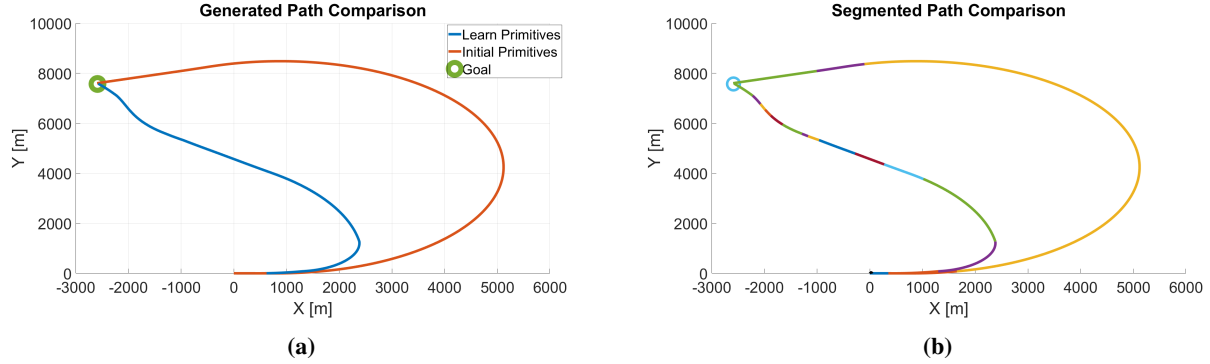


Fig. 11 (a) Comparison of full trajectory generated by original primitives vs. the learned primitives given the same goal position. (b) Comparison of trajectories with individual primitives segmented by color.

The improvement in planner performance can also be seen by comparing two trajectories with the same endpoints, but composed of the original primitives and the new primitives, respectively. Figure 11 shows trajectories generated before and after new primitives were added to the MA. The learned primitives clearly allow a more direct path to the goal. Figure 11b indicates the different primitives used by color. The second trajectory uses learned primitives with a smaller turning radius to align with the goal more quickly than the steady turn from the original library.

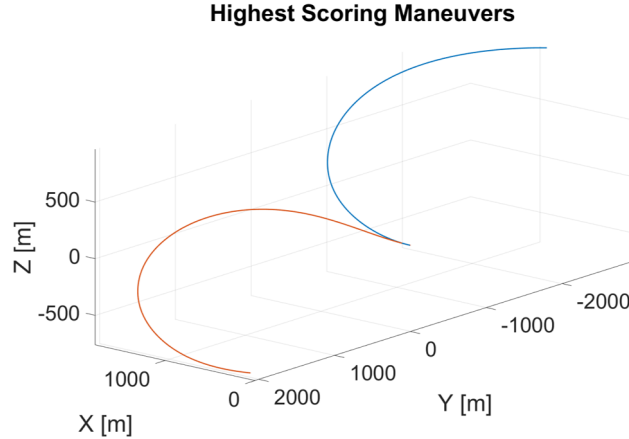


Fig. 12 Two maneuvers which produced the highest individual improvements in the cost map. Both are aggressive turns that change heading by approximately 180 degrees.

Qualitative results were also observed to see what behaviors provided the greatest improvement to the MA. Figure 12 shows two of the maneuvers that added the greatest improvements. Both seem to be quick turn-arounds that change the heading by approximately 180 degrees. The intuition behind these maneuvers is clear since the original library only provided heading changes in the form of steady turns. If the aircraft was close to the goal but facing away, the MA would need to plan a path that flies away and makes a slow turn to face the goal. These new maneuvers allow the MA to plan more aggressive trajectories that succeed faster. Additionally, Figure 11a shows two full paths generated by the original primitives and the learned primitives. Both paths go to the same goal.

VII. Conclusion

The framework presented in this work autonomously improves the performance of a MA on a point-to-point navigation task by adding a small number of new primitives. This improvement is made entirely through exploration by a reinforcement learning agent with no human-in-the-loop input. This provides a stepping stone to employ this method in more complicated environments where expert knowledge may be lacking. The approach can easily be transferred to

new mobile systems or different task spaces. Additionally, cost functions can be substituted to prioritize other objectives while completing a task.

Future directions for this work include investigation of alternate exploration agents and application to tasks in dynamic environments. Trajectory optimization techniques may provide an alternative to reinforcement learning agents, though they may require modification to the model structure. There may be cases where trajectory optimization techniques generate better maneuvers. However, we anticipate that such techniques may provide localized benefits. Specifically, optimization techniques may provide a benefit when the initial conditions match those used to generate the trajectory. In addition, such trajectories may generalize more poorly if a different cost function is used. We view the broad exploration of the SAC as a key advantage for creating a broadly effective library. Dynamic environments can also be explored using this method. It may be more challenging for experts to design primitives for such environments. Adversarial environments are of particular interest since recent works have shown that reinforcement learning has the ability to develop novel strategies that out perform human experts [5].

References

- [1] Frazzoli, E., Dahleh, M. A., and Feron, E., “Maneuver-based motion planning for nonlinear systems with symmetries,” *IEEE Transactions on Robotics*, Vol. 21, No. 6, 2005, pp. 1077–1091. <https://doi.org/10.1109/TRO.2005.852260>.
- [2] Paranjape, A. A., Meier, K. C., Shi, X., Chung, S.-J., and Hutchinson, S., “Motion primitives and 3D path planning for fast flight through a forest,” *The International Journal of Robotics Research*, Vol. 34, No. 3, 2015, pp. 357–377. <https://doi.org/10.1177/0278364914558017>, URL <https://doi.org/10.1177/0278364914558017>.
- [3] Abbeel, P., Coates, A., and Ng, A. Y., “Autonomous Helicopter Aerobatics through Apprenticeship Learning,” *The International Journal of Robotics Research*, Vol. 29, No. 13, 2010, pp. 1608–1639. <https://doi.org/10.1177/0278364910371999>, URL <https://doi.org/10.1177/0278364910371999>, publisher: SAGE Publications Ltd STM.
- [4] Frazzoli, E., Dahleh, M. A., and Feron, E., “Robust hybrid control for autonomous vehicle motion planning,” *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, Vol. 1, IEEE, 2000, pp. 821–826.
- [5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., “Mastering the game of go without human knowledge,” *Nature*, Vol. 550, No. 7676, 2017, pp. 354–359.
- [6] Camci, E., and Kayacan, E., “Planning Swift Maneuvers of Quadcopter Using Motion Primitives Explored by Reinforcement Learning,” *2019 American Control Conference (ACC)*, IEEE, Philadelphia, PA, USA, 2019, pp. 279–285. <https://doi.org/10.23919/ACC.2019.8815352>, URL <https://ieeexplore.ieee.org/document/8815352/>.
- [7] Frazzoli, E., Dahleh, M. A., and Feron, E., “A hybrid control architecture for aggressive maneuvering of autonomous helicopters,” *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, Vol. 3, 1999, pp. 2471–2476 vol.3. <https://doi.org/10.1109/CDC.1999.831296>.
- [8] Majumdar, A., and Tedrake, R., “Funnel libraries for real-time robust feedback motion planning,” *The International Journal of Robotics Research*, Vol. 36, No. 8, 2017, pp. 947–982. <https://doi.org/10.1177/0278364917712421>, URL <https://doi.org/10.1177/0278364917712421>.
- [9] Frazzoli, E., Dahleh, M. A., and Feron, E., “Real-Time Motion Planning for Agile Autonomous Vehicles,” *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129. <https://doi.org/10.2514/2.4856>, URL <https://doi.org/10.2514/2.4856>.
- [10] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [11] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,” *arXiv:1801.01290 [cs, stat]*, 2018. URL <http://arxiv.org/abs/1801.01290>, arXiv: 1801.01290.
- [12] Fujimoto, S., Van Hoof, H., and Meger, D., “Addressing function approximation error in actor-critic methods,” *arXiv preprint arXiv:1802.09477*, 2018.
- [13] Faust, A., Oslund, K., Ramirez, O., Francis, A., Tapia, L., Fiser, M., and Davidson, J., “PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Brisbane, QLD, 2018, pp. 5113–5120. <https://doi.org/10.1109/ICRA.2018.8461096>, URL <https://ieeexplore.ieee.org/document/8461096/>.

- [14] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.
- [15] Konda, V. R., and Tsitsiklis, J. N., “Actor-critic algorithms,” *Advances in neural information processing systems*, 2000, pp. 1008–1014.
- [16] Frazzoli, E., “Robust hybrid control for autonomous vehicle motion planning,” Ph.D. thesis, Massachusetts Institute of Technology, 2001.
- [17] Nguyen, L., Ogburn, M., Gilbert, W., Kibler, K., Brown, P., and Deal, P., “Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability,” , Dec. 1979. URL <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19800005879.pdf>.