Myopic Control of Systems with Unknown Dynamics

Melkior Ornik[1], Steven Carr[2], Arie Israel[3], and Ufuk Topcu[1,2]

***Abstract*— This paper introduces a strategy for satisfying basic control objectives for systems whose dynamics are almost entirely unknown. This setting is motivated by a scenario where a system undergoes a critical failure, thus significantly changing its dynamics. In such a case, retaining the ability to satisfy basic control objectives such as reach-avoid is imperative. To deal with significant restrictions on our knowledge of system dynamics, we develop a theory of myopic control. The primary goal of myopic control is to, at any given time, optimize the current direction of the system trajectory, given solely the limited information obtained about the system until that time. Building upon this notion, we propose a control algorithm which simultaneously uses small perturbations in the control effort to learn local system dynamics while moving in the direction which seems to be optimal based on previously obtained knowledge. We show that the algorithm results in a trajectory that is nearly optimal in the myopic sense, i.e., it is moving in a direction that seems to be nearly the best at the given time, and provide formal bounds for suboptimality. We demonstrate the usefulness of the proposed algorithm on a high-fidelity simulation of a damaged Boeing 747 seeking to remain in level flight.**

I. Introduction

In an event of a rapid and unexpected change in the dynamics of a system, the ability to retain a degree of useful control over a system is crucial. A notable example of such a catastrophic event is that of an aircraft losing its wing after collision with another airplane [2]. In the particular event, the pilot managed to retain enough control authority over the aircraft to be able to safely land at a nearby airbase. The pilot's strategy depended on his intuition and prior experience. This paper seeks to develop a methodical approach to control of an unknown system in real time. The strategy we propose is based on an intuitive approach one might use when trying to drive an unknown vehicle: performing small "wiggles" in controls to see how the system behaves, before deciding on a longer-term control action.

In the described setting of control of an unknown system, the only data available to extract information on the system dynamics can be obtained during the system run. Data-driven learning of the dynamics of unknown systems has been a subject of significant recent research [6], [15], [33]. However, the developed methods often make significant assumptions on possible system dynamics or require a large amount of data; hence, they are not suitable for the above motivating example. A recent work [1] based on a similar scenario uses differential inclusions to assess the safety of a system governed by unknown dynamics. However, [1] discusses uncontrolled dynamics, and does not prescribe a control law which would ensure safety of a trajectory. A number of works do deal with control design for unknown dynamics [18], [26], [36] and damaged aircraft [23], [28]. The results of these papers again significantly differ from our objective, as they generally do not include formal guarantees on short-time, non-asymptotic system performance. Additionally, the control specifications discussed in those works are markedly different than the ones that naturally arise in the scenario we are exploring, as they are all based on reference tracking. In particular, [18], [23], [36] employ methods based on transfer functions to solve a tracking problem in an unknown system, while [26] and [28] use neural networks in optimal tracking control. Furthermore, [23] and [28] require significantly more assumptions on the dynamics than the approach proposed in this paper.

In this paper, we are primarily concerned with solving reach-avoid-type problems. This consideration is motivated by practical reasons: in the event of a system failure, satisfying advanced control specifications likely becomes impossible, and the system should concentrate on its survival. In physical systems, it is reasonable to pose the problem of system survival as a question of reaching a certain target set as soon as possible while avoiding obstacles and staying within a predefined safety envelope. In the motivating example of a damaged aircraft, which we will use as a running example throughout this paper, the target set is the airbase runway, and the aircraft should reach it as soon as possible without touching the ground beforehand.

A standard approach to reach-avoid problems in systems with known dynamics [38] is based on constrained optimal control, where the constraints are given by the geometry of the safety envelope and obstacles, and the time to reach a target set is minimized. The solutions to this problem clearly depend on the full dynamics of the system. In our setting we have very little information on the system dynamics. Hence, it is unlikely that we can find the exact optimal control law to solve a reach-avoid problem.

We instead propose the notion of *myopic control*, and use it to solve problems of the reach-avoid type. In this notion, the system aims to use a control law that seems to be *myopically optimal*, i.e., work best at the given time, without any knowledge on whether that control law will lead to good results in the future. This framework is motivated by the inherent lack of firm knowledge of future system dynamics.

To counterbalance the lack of confidence about the future, a new myopically-optimal control input is calculated and applied in the system every so often, after a short time

[1] Institute for Computational Engineering and Sciences, University of Texas at Austin. e-mails: mornik@ices.utexas.edu, utopcu@utexas.edu

[2] Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin. e-mail: stevencarr@utexas.edu

[3] Department of Mathematics, University of Texas at Austin. e-mail: arie@math.utexas.edu

interval. In order to determine the next myopically-optimal control input, the system continually modifies the previous input by adding a series of *wiggles*: small, short-time controls that do not significantly change the system trajectory, but act as a learning mechanism to determine the local control dynamics of the system. In order for this mechanism to function, we make a technical assumption that the underlying system is affine in control.

The proposed algorithm results in an approximate solution to the myopic optimal control problem, with a degree of suboptimality dependent on the length of the control law update interval and the size of learning controls, i.e., wiggles. Additionally, if there are known bounds on regularity of system dynamics, the parameters of the algorithm can be set to ensure a desired bound on the degree of suboptimality.

The organization of this paper is as follows: Section II introduces the class of problems we are interested in solving, and formalizes the limitations to control design imposed by the lack of knowledge of the system. Section III then defines the myopic optimal control problem. Section IV provides an overview of the proposed algorithm to approximately solve this problem, and Section V provides performance bounds for the proposed algorithm. Finally, Section VI provides simulation results for the example of a wing-damaged aircraft, showing that the algorithm indeed performs as expected.

Notation. Set of all continuous functions from set $\mathcal{A} \subseteq \mathbb{R}^n$ to set $\mathcal{B} \subseteq \mathbb{R}^m$ is denoted by $C(\mathcal{A}, \mathcal{B})$. If $I \subseteq \mathbb{R}$ is an interval and $t \in I$, the right one-sided derivative of function $f : I \to \mathbb{R}^n$ at time t, if it exists, is denoted by $df(t+)/dt$. For a vector $v \in \mathbb{R}^n$, $\|v\|$ denotes its 2-norm, and $\|v\|_\infty$ its max-norm. Finally, $e_i \in \mathbb{R}^m$ denotes the i-th coordinate vector, i.e., a vector consisting of all zeros, except for 1 at the i-th position.

II. Problem Definition

We investigate the control system

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x) u_i \tag{1}$$

which evolves on $\mathcal{X} \subseteq \mathbb{R}^n$, where $f, g_i \in C(\mathcal{X}, \mathcal{X})$ are a priori unknown functions, and $u = (u_1, \ldots, u_m)$ is the m-dimensional control input with values in a bounded set $\mathcal{U} \subseteq \mathbb{R}^m$. While it is trivial to modify our proposed algorithm for the case where $\mathcal{U}$ is any manifold with corners in the sense of [17], for simplicity we take $\mathcal{U} = \{u \in \mathbb{R}^m \mid \|u\|_\infty \leq 1\}$. In the remainder of the text, a solution to (1) with control input u and initial value x^0 is denoted by $\phi_u(\cdot, x^0)$.

We are assuming that, at any time instance, we are able to observe the full state x and the entire control u; a modification of our approach to allow for noise in the state observations and partial observations is a subject of current work not covered by this paper. Additionally, in a practical sense of control of a real-world system, we are usually only interested in the behavior of a system on a compact state space $\mathcal{X}$. It thus makes sense to assume that there exist $M_0 \geq 0$ and $M_1 \geq 0$ such that

- $\|f(x)\| \leq M_0$, $\|g_i(x)\| \leq M_0$, for all $x \in \mathcal{X}$ and all $i \in \{1, \ldots, m\}$, and
- $\|f(x) - f(y)\| \leq M_1 \|x - y\|$, $\|g_i(x) - g_i(y)\| \leq M_1 \|x - y\|$, for all $x, y \in \mathcal{X}$ and all $i \in \{1, \ldots, m\}$.

The class of control-affine systems (1) covers a wide array of physical control systems, including standard linear aircraft models (e.g., [7], [34]) and a variety of nonlinear models of mechanical systems [21], e.g., a unicycle with rider [27], quadrotor helicopter [24], and the F-8 Crusader aircraft [14]. Control and optimal control of general control-affine systems have been substantially covered by literature (see, e.g., [8], [16], [21]), including investigations of reach-avoid problems [12]. However, we emphasize that, motivated by the scenario of an unexpected failure in a physical system, our setting introduces two additional requirements on control design:

R1 Functions $f, g_1, \ldots, g_m$ in (1) are unknown at the beginning of the system run. We are allowed to use trajectory data *during* the system run to determine information on them, and we may have some prior limited information on the dynamics (given, e.g., by our knowledge of physical laws; we give a specific example in Section VI). However, when the run starts, we do not know the actual values of functions $f, g_1, \ldots, g_m$, but are only aware that the system is of the form (1), and potentially know a bound on the functions' norms M_0 and Lipschitz constants M_1.

R2 There is only one system run, starting from a single predetermined initial state x^0. All control design needs to be performed during that run, without any repetitions.

We note that the bulk of previous work on systems with disturbances or noise, as discussed in, e.g., [9], [20], does not help to deal with requirement R1. The standard setup in the theory of systems with disturbances requires the controller to know the "general" system dynamics, which are then modified by some unknown, but bounded, disturbance or noise. Requirement R1 is significantly more limiting for control design. It stipulates that the system dynamics are essentially unknown at the beginning of the system run.

The goal of this paper is to develop a method for solving reach-avoid-type control problems [22] in the control-affine setting (1) under the above requirements R1 and R2. In the interest of space, we forgo a formal definition of a *reach-avoid problem*, with the understanding that it is a problem of reaching a certain target area $\mathcal{T}$ of the state space, while avoiding certain undesirable areas $\mathcal{B}$ throughout the system run. Its simplification is an *avoid problem* — if there is no particular target area, the reach-avoid problem naturally transforms into the problem of remaining outside the undesirable set $\mathcal{B}$ for as long as possible. This problem is again one natural abstraction of our running example: instead of having to deal with the technicalities of aircraft landing, we may simply want to pose the question of requiring the damaged aircraft to stay above ground for as long as possible.

This paper would ideally end by providing a control design to solve reach-avoid-type problems which respects requirements R1 and R2. This outcome is likely impossible. The

requirements that we have imposed imply that we have no way of acquiring significant knowledge about the dynamics in a certain area of $\mathcal{X}$ until the system trajectory reaches that area. In fact, even when the system trajectory reaches a point $x \in \mathcal{X}$, we are still unable to exactly determine the values of $f(x)$ and $g_i(x)$, because a single controlled trajectory provides only limited information on full control dynamics around a point. Thus, it is not reasonable to expect that a reach-avoid-type problem can be exactly solved under the requirements of our setting. We now present an intuitive variation of these problems that can be approached using strategies which respect requirements R1 and R2.

III. Myopic Control

Suppose that we want to solve the reach-avoid problem under requirements R1 and R2. Requirement R2 implies that, in order to determine the optimal control law u^* to be used starting at some time t, one may only use the information obtained from the system trajectory until time t. Thus, based on our knowledge of the system dynamics in the interval $[0, t]$, one would have to assess the "quality" of a candidate future control law on the entire interval $(t, +\infty)$. As mentioned, we can only do so with very limited confidence, since the dynamics far away from points already visited at some time $0 \leq t' \leq t$ are unknown.

The above discussion behooves us to replace the original reach-avoid-type problem by a *myopic optimal control problem*, where we want to design a control law such that, at every time instance, the trajectory *appears* to behave as well as possible. For instance, if our ultimate desire is to solve the avoid problem, it makes sense to require that the trajectory at any given instance of time is moving away from the undesirable set $\mathcal{B}$ as fast as possible. Clearly, this reasoning may not be ultimately optimal, as moving away from $\mathcal{B}$ as fast as possible at some point could bring the trajectory into a position where the dynamics are such that it is simply forced to enter $\mathcal{B}$ (see Fig. 1 for an example). Thus, this notion of "appearing to behave as well as possible" is simply our best guess, based on intuition and any prior or side information about the dynamics available during the system run.

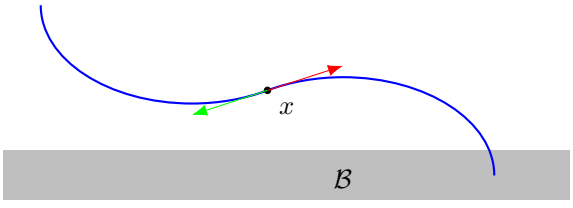

Fig. 1. An example illustrating the built-in imperfection of myopic control. Assume that the system dynamics are such that, unbeknownst to the control designer, the system is restricted to moving along the blue curve. The control objective is to avoid the undesirable set $\mathcal{B}$, denoted in gray. When the system is at point x, the available direction vectors are denoted by the green and red arrows. From the designer's myopic perspective, it may appear better to control the system in the red direction, because this direction forces the system to move further away from $\mathcal{B}$. That action will ultimately result in the system moving into the undesirable set.

We propose to formalize the above notion of "appearing to behave as well as possible" using a *goodness function*

$$(\phi, v) \mapsto G(\phi, v), \quad \phi \in \mathcal{F}, v \in \mathbb{R}^n, \tag{2}$$

where $\mathcal{F} = \cup_{T \geq 0} C([0, T], \mathcal{X})$. Function G is intended to quantify how well the trajectory ϕ and its current velocity v appear to be doing at satisfying the desired specifications.

Myopic policies and goodness or utility functions have long been used in decision theory, in a wide variety of control-theoretical settings, including resource allocation [31], computer vision [32], and Bayesian learning [37]. However, all these settings differ significantly from the setting of this paper; to the authors' knowledge, myopic decision-making has never been utilized in dealing with control of unknown systems.

Example 1: Going back to our running example of a damaged aircraft, assume that we want to have the aircraft stay away from the ground for as long as possible. In that case, if x_1 and x_2 denote the aircraft's horizontal and vertical position, respectively, undesirable set $\mathcal{B}$ is given by $\mathcal{B} = \{(x_1, x_2) \mid x_2 \leq 0\}$. One — but not the only — natural option is for G to correspond to the slope on which the trajectory is moving towards the boundary of $\mathcal{B}$, i.e., $x_2 = 0$. Without additional information about the system, the more negative this slope is, the worse the aircraft naturally appears to be doing. Hence, $G(\phi, v) = v_2/v_1$, where we disregard the case of $v_1 \leq 0$, is a reasonable goodness function. Fig. 2 provides an illustration.

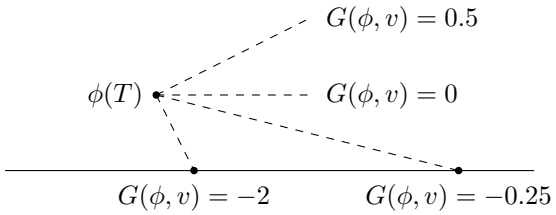

Fig. 2. An illustration of the goodness function G from Example 1. Dashed lines represent the possible tangents to the trajectory of system (1) at $\phi(T)$ and thus our local predictions of future movement of the system state. These tangents are evaluated by G depending on their slope towards the undesirable set $\mathcal{B}$, whose boundary is represented by a solid line.

In general, determining an appropriate goodness function depends on our information on the system and the control specifications. While this is a subject of future research, the topic is discussed in more detail in an extended version [30] of this paper. In this paper, we will provide a possible design for our running example of a damaged aircraft in Section VI.

Having elaborated on how a goodness function provides a measure of apparent quality of a trajectory at a given point (with respect to problem specifications), we now formally introduce the myopic optimal control problem. As a slight deviation from standard notation, in order to emphasize that $\phi \in \mathcal{F}$ has $[0, T]$ as its domain, we will occasionally write such functions as $\phi|_{[0,T]}$.

Myopic Optimal Control Problem (MOCP): Let $x^0 \in \mathcal{X}$. Find a control law $u^* : [0, +\infty) \to \mathcal{U}$ such that, for all $t \geq 0$,

if $x = \phi_{u^*}(t, x^0)$, then

$$G\left(\phi_{u^*}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_{u^*}(0+, x)}{dt}\right) = \max_{u \in \mathcal{U}} G\left(\phi_{u^*}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_u(0+, x)}{dt}\right). \tag{3}$$

Equation (3) slightly abuses notation: if $u \in \mathcal{U}$, then u is not formally a control input as a function of time. By $d\phi_u(0+, x)/dt$ we denote the right-hand side of (1) when u is plugged in. Additionally, while the notion of a solution to (1) is questionable in the case of general measurable functions u^*, in the remainder of the paper we will be working with piecewise-constant control laws, and in that sense we will not encounter any issues.

In the following section we propose a strategy to find an approximate solution to the MOCP, while respecting requirements R1 and R2. This strategy is the central theoretical contribution of this paper. The obtained solution is approximate in the sense that, for any $T > 0$ and $\mu > 0$, Algorithm 3 can generate a piecewise-constant control law u^+ that satisfies

$$\left| G\left(\phi_{u^+}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_{u^+}(0+, x)}{dt}\right) - \max_{u \in \mathcal{U}} G\left(\phi_{u^+}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_u(0+, x)}{dt}\right)\right| < \mu \tag{4}$$

for all $t \geq T$. This law will be generated in real-time, during the system run. Thus, our construction will approximately solve the MOCP for a single initial state x^0, while satisfying requirements R1 and R2.

Existence of a control law that exactly solves the MOCP is, to the authors' knowledge, a novel problem. However, even if an exact solution to the MOCP exists, it can be easily shown that it cannot be found while respecting requirements R1 and R2. We omit a discussion for lack of space, but once again point the reader to the extended version [30] of this paper, where the question is investigated.

IV. Learn-Control Algorithm

We intend to approximately solve the MOCP by the use of short-time piecewise-constant controls. The proposed strategy relies on repeatedly learning the local dynamics at points on the observed system trajectory, and using those dynamics to simultaneously apply a myopically-optimal control law.

In order to learn the local dynamics around a single point on the trajectory, the controller can apply any $m+1$ affinely independent constant controls for a short period of time. Since system (1) is control-affine, we can use the observed changes in states to approximate the function $u \mapsto v_x(u)$ defined by

$$v_x(u) = f(x) + \sum_{i=1}^{m} g_i(x) u_i,$$

where x is a point on the trajectory around the time of this learning process.

After doing so, if ϕ denotes the trajectory until the end of the learning process, by determining $\operatorname{argmax}_u G(\phi, v_x(u))$, a constant control u^* which appears to best satisfy the control specifications at point x can be found. This control is then used during the next iteration of the learning process. In other words, the next learning process uses an affinely independent set of controls $u^* + \Delta u^0, \ldots, u^* + \Delta u^m$, where Δu^i are small enough to ensure that their application still results in near-optimal goodness.

The described method of using short-time controls to achieve real-time (nearly) optimal behavior bears some resemblance to the work in [3] and to extremum seeking [4]. However, both the problem statements and the settings of these approaches are significantly different from ours, not aiming to deal with a reach-avoid problem in a system with entirely unknown dynamics.

The above description of the control strategy is formally given as Algorithm 3.

Algorithm 3

Input ε, δ.
$u^* := 0$
$t_0 := 0$
repeat
  Let x^0 be the current state of the system.
  Take $\Delta u^0 = 0$ and choose
  $\Delta u^1 = \pm\delta e_1, \ldots, \Delta u^m = \pm\delta e_m$
  such that $u^* + \Delta u^1, \ldots, u^* + \Delta u^m \in \mathcal{U}$.
  for $j = 0, \ldots, m$
    Apply control $u^* + \Delta u^j$ in the interval of time
    $[t_0 + j\varepsilon, t_0 + (j+1)\varepsilon)$.
    Let x^{j+1} be the system state at the end of that period.
  end for
  $x := x^{m+1}$
  Define function $\tilde{v}_x : \mathcal{U} \to \mathbb{R}^n$ as follows:
    If $\lambda_0, \ldots, \lambda_m$ are unique coefficients
    with $\sum \lambda_j = 1$ such that $u = \sum \lambda_j (u^* + \Delta u^j)$,
    let $\tilde{v}_x(u) = \sum \lambda_j (x^{j+1} - x^j)/\varepsilon$.
  Take $u^* \in \operatorname{argmax} G(\phi|_{[0,t_0+(m+1)\varepsilon]}, \tilde{v}_x(u))$
  $t_0 := t_0 + (m+1)\varepsilon$
until the end of system run

While Algorithm 3 prescribes a particular form for Δ^i, the local dynamics of the system can be learned by using any Δ^i such that $u^* + \Delta u^1, \ldots, u^* + \Delta u^m \in \mathcal{U}$ is an affinely independent set. The particular form for Δ^i is included in the algorithm in order to more easily establish a bound on the degree of suboptimality of the resulting control law. This bound, which is dependent on parameters $\delta \in (0, 1)$ and $\varepsilon > 0$, is described in Theorem 8.

Remark 4: Algorithm 3 does not separate the learning and control phases of the algorithm: the algorithm learns the local dynamics as a result of performing slight perturbations of the previously established optimal control law. However, we note that these two phases can be decoupled by a minor modification of the above algorithm. After learning the local dynamics through consecutively applying *any* $m+1$ affinely

independent controls in a short time period $\varepsilon' < (m+1)\varepsilon$, the system can then apply the optimal control law derived from these dynamics for the remaining $(m+1)\varepsilon - \varepsilon'$ time in the cycle, after which it begins a new cycle by learning the new local dynamics. While this modification results in an ε' period of time inside every iteration of the learn-control cycle without any guarantees on the degree of myopic suboptimality, it offers computational benefits in the case where the optimal control law does not rapidly change over time. Because of this reason, the algorithm used in the simulation work of Section VI contains the modification described above. The performance bounds presented in the subsequent section apply to Algorithm 3 as originally presented.

V. Performance Bounds

To save space, the theoretical results are accompanied merely by intuitive proof sketches. We refer the reader to the extended version [30] of this paper for full proofs.

A. Learning

We claim that the procedure in lines 5–11 of Algorithm 3 produces a good approximation $\tilde{v}_x : \mathcal{U} \to \mathbb{R}^n$ of the map $v_x : \mathcal{U} \to \mathbb{R}^n$, with $x = x^{m+1}$ defined as in the algorithm. For every point $u^* \in \mathcal{U} = [-1,1]^m$, if $u_i^* \geq 0$, then $-1 \leq u_i^* - \delta \leq 1$, and if $u_i^* < 0$, then $-1 \leq u_i^* + \delta \leq 1$, for all $0 < \delta < 1$. Hence, for all $u^* \in \mathcal{U}$ we can choose $\Delta u^i = \delta e_i$ or $\Delta u^i = -\delta e_i$ such that $u^* + \Delta u^i \in \mathcal{U}$, as stipulated by line 5 of Algorithm 3. We also note that $\{u^* + \Delta u^i \mid 0 \leq i \leq m\}$ is trivially an affinely independent set.

In our description of the remainder of the procedure, we slightly abuse notation and denote $u^* + \Delta u^i$ by u^i.

For each $j \in \{0, 1, \ldots, m\}$, vector

$$\frac{x^{j+1} - x^j}{\varepsilon} = \frac{\phi_{u^j}(\varepsilon, x^j) - \phi_{u^j}(0, x^j)}{\varepsilon} \tag{5}$$

approximates the value

$$\frac{d\phi_{u^j}(\varepsilon, x^j)}{dt} = f(x^{j+1}) + \sum_{i=1}^{m} g_i(x^{j+1}) u_i^j = v_{x^{j+1}}(u^j). \tag{6}$$

Additionally, since x^{j+1} and $x = x^{m+1}$ are not far apart (because x^{m+1} is the state of the trajectory just $(m-j)\varepsilon$ later than x^{j+1}), $v_{x^{j+1}}(u^j) \approx v_x(u^j)$.

Finally, since $u^0, \ldots, u^m$ are affinely independent, for every $u \in \mathcal{U}$ there exist unique $\lambda_0, \ldots, \lambda_m \in \mathbb{R}$ such that $u = \lambda_0 u^0 + \ldots + \lambda_m u^m$ and $\lambda_0 + \ldots + \lambda_m = 1$. Thus,

$$v_x(u) = \sum_{j=0}^{m} \lambda_j \left(f(x) + \sum_{i=1}^{m} g_i(x) u_i^j \right) = \sum_{j=0}^{m} \lambda_j v_x(u^j).$$

Since we already devised an approximation for $v_x(u^j)$ based on (5)-(6), we can thus approximate $v_x(u)$ for any $u \in \mathcal{U}$. A bound on the error of this approximation is given by the following theorem.

Theorem 5: Let $x^0, \ldots, x^{m+1} = x$ be as in Algorithm 3, and let $u \in \mathcal{U}$. Let $\lambda_0, \ldots, \lambda_m \in \mathbb{R}$ be such that $u = \lambda_0 u^0 + \ldots + \lambda_m u^m$ and $\lambda_0 + \ldots + \lambda_m = 1$. Then,

$$\left\| v_x(u) - \sum_{j=0}^{m} \lambda_j \frac{x^{j+1} - x^j}{\varepsilon} \right\| \leq 2 M_0 M_1 (m+1)^3 T_\delta \varepsilon,$$

with $T_\delta = 1 + 4m^{3/2}/\delta$.

B. Control

From Theorem 5 it follows that Algorithm 3 approximates $v_{x^{m+1}}(u) = f(x^{m+1}) + \sum g_i(x^{m+1}) u_i$ for any value $u \in \mathcal{U}$ with arbitrary precision. Thus, we are able to accurately calculate G at time $t_0 + (m+1)\varepsilon$ for any control u, with bounds on precision depending on the regularity of G. Additionally, for a fixed x and ϕ, $u \mapsto G(\phi, v_x(u))$ is a real function of a bounded variable $u \in \mathcal{U}$. Depending on the properties of G, $\max$ and argmax of this function can be found by an analytic or numerical optimization method; discussion of available methods is not within the scope of this paper. Thus, we can find an (approximately) optimal control u^* to be applied when the previous system trajectory is given by ϕ, and the trajectory is at x.

As shown in Algorithm 3, our plan is to use the found optimal control u^* for a short time $(m+1)\varepsilon$ after the system is at point x^{m+1}. During that time, the system will learn new local dynamics around the new x^{m+1}, and the whole procedure will be repeated again. We note that this process does not necessarily generate an optimal control law, as the best control u^* for a certain point x^{m+1} may no longer be the best immediately after the system leaves the said point. However, if the function G is "tame enough", u^* will still be a good approximation of the optimal control. This fact remains true even if we take into account that the policy u^* was calculated based on a slightly incorrect learned dynamics at x^{m+1}. To prove this claim, we first introduce a measure of tameness of G.

Definition 6: Function $G : \mathcal{F} \times \mathbb{R}^n \to \mathbb{R}$ has *Lipschitz constant* L if for all $\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]} \in \mathcal{F}$ and $v_1, v_2 \in \mathbb{R}^n$, the following holds:

$$\begin{aligned} &\left| G(\phi_1|_{[0,T_1]}, v_1) - G(\phi_2|_{[0,T_2]}, v_2) \right| \leq \\ &\quad L \left(d\left(\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]}\right) + \|v_1 - v_2\| \right), \end{aligned} \tag{7}$$

where

$$\begin{aligned} &d\left(\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]}\right) = \\ &\quad |T_1 - T_2| + \max_{t \in [0, \min(T_1, T_2)]} \|\phi_1(t) - \phi_2(t)\|. \end{aligned} \tag{8}$$

Definition 6 explicitly makes note of the usually standard notion of a Lipschitz constant because d as defined in (8) is not a proper metric on $\mathcal{F}$. Thus, the existence of a Lipschitz constant as in Definition 6 does not imply continuity of G in the standard sense.

The claims of the beginning of this section are now formalized by the following result.

Theorem 7: Let $\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]} \in \mathcal{F}$, and let $\nu > 0$. Define $x = \phi_1(T_1)$ and $y = \phi_2(T_2)$. Assume that G has Lipschitz constant L in the sense of Definition 6, and let u^* be the optimal control at x under *approximately* learned local dynamics with the previous trajectory $\phi_1|_{[0,T_1]}$, i.e., $G(\phi_1|_{[0,T_1]}, \tilde{f} + \sum \tilde{g}_i u_i^*) = \max_{u \in \mathcal{U}} G(\phi_1|_{[0,T_1]}, \tilde{f} + \sum \tilde{g}_i u_i)$, where $\|\tilde{f} + \sum_{i=1}^m \tilde{g}_i u_i - (f(x) + \sum_{i=1}^m g_i(x) u_i)\| \leq$

ν for all $u \in \mathcal{U}$. Then,

$$\left|\max_u G\left(\phi_2|_{[0,T_2]}, v_y(u)\right) - G\left(\phi_2|_{[0,T_2]}, v_y(u^*)\right)\right| / L \leq 2d\left(\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]}\right) + 2\left((m+1)M_1\|x-y\| + \nu\right).$$

In the context of Algorithm 3, the result of Theorem 7 can be interpreted in the following way: let us take T_1 to be the time at the beginning of one **repeat** loop in the algorithm, $\phi_1|_{[0,T_1]}$ to be the trajectory of the system until time T_1, T_2 to be any time in the interval $[T_1, T_1+(m+1)\varepsilon)$, and $\phi_2|_{[0,T_2]}$ to be the trajectory of the system until time T_2. Then, Theorem 7 essentially proves that Algorithm 3 results in a nearly optimal policy with respect to (3). There is only one major point that remains to be discussed. In Algorithm 3, we do not apply *just* u^* as the control. In order to facilitate learning, we modify this u^* by some small Δu^i. We need to show that such a small control perturbation will not significantly impact the degree of suboptimality of Algorithm 3. Theorem 8 deals with that issue. It also translates the bound in Theorem 7, which depends on $d(\phi_1|_{[0,T_1]}, \phi_2|_{[0,T_2]})$, $\|x-y\|$, and ν, into a bound in terms of algorithm parameters ε and δ. Solely for notational purposes, we assume that the run of system (1) is of infinite horizon.

Theorem 8: Let $x^0 \in \mathcal{X}$. Let $u^+ : [0, +\infty) \to \mathcal{U}$ be the control law used in Algorithm 3. Assume that G has Lipschitz constant L. Then, for all $t \geq (m+1)\varepsilon$,

$$\begin{aligned} &\left| G\left(\phi_{u^+}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_{u^+}(0+, x)}{dt}\right)\right. \\ &\left. - \max_{u\in\mathcal{U}} G\left(\phi_{u^+}(\cdot, x^0)|_{[0,t]}, \frac{d\phi_u(0+, x)}{dt}\right)\right| \leq \\ &6L(M_0+1)(M_1+1)(m+1)^3\left(1+\frac{4m\sqrt{m}}{\delta}\right)\varepsilon \\ &+ LM_0(m+1)\delta, \end{aligned} \tag{9}$$

where $x = \phi_{u^+}(t, x^0)$.

Theorem 8 is the central result of the theoretical discussions of this paper. It shows that, for any $\mu > 0$, if ε and δ are chosen in such a way that $6L(M_0+1)(M_1+1)(m+1)^3(1+4m\sqrt{m}/\delta)\varepsilon + LM_0(m+1)\delta \leq \mu$, Algorithm 3 will result in a control law that approximately satisfies (3) *at every time* $t \geq (m+1)\varepsilon$, with an error no larger than μ.

As can be seen from Theorem 8, in order to achieve arbitrarily good performance bounds on Algorithm 3, the control actuators must allow that an arbitrarily small control input can be applied for an arbitrarily short amount of time. While actual minimal sizes of control inputs and control application time will depend on the actuator type, improvement on both of these resolutions is a topic of recent practical research [10], [19].

VI. Simulation Results

The simulation work presented in this section is motivated by our running example: a damaged aircraft that is attempting to retain a safe altitude. In particular, we consider the dynamics of a Boeing 747-200 that lost 33% of its right wing. The dynamics of such an aircraft were developed in [5], and we use the nonlinear model contained therein to simulate aircraft behavior. We emphasize, however, that we do not use these dynamics at any point to decide on an appropriate control law: the controller is ignorant of the true system dynamics and bases its decisions on learned local dynamics as described in Algorithm 3.

The state variables $x = [v,\ w,\ q,\ \theta,\ \beta,\ p,\ r,\ \phi,\ z]^T$ that we consider are forward velocity, vertical velocity, pitch rate, pitch angle, sideslip angle, roll rate, yaw rate, roll angle, and altitude, respectively. The control inputs $u = [\delta_e,\ \delta_a,\ \delta_r]$ are the elevator, aileron, and rudder deflections from the wings-level trim condition for an undamaged aircraft, respectively. For the sake of readability, in the remainder of the paper, we denote the appropriate coordinates of x by $x_1, \ldots, x_9$, and analogously for u. The limits (in degrees) on allowed control inputs $u \in \mathcal{U}$ are set to $u_1 = \delta_e \in [-15, 15]$, $u_2 = \delta_a \in [-25, 25]$ and $u_3 = \delta_r \in [-10, 10]$, roughly informed by the descriptions in [13].

A. Controller specifications

The initial flight conditions are taken from the trim conditions of a Boeing 747-200 at 283000 kg, 500 knots and a nominal altitude of 500 m [29]. The setting investigated in this example is that the aircraft suffered damage while in flight, during a banked turn, at an angle of 0.5 radians, i.e., 28.6 degrees. Thus, the initial system state is $x(0) = [257.22, -0.7818, 0, 2.5, 0, 0, 0, 28.6, 500]^T$ (all values are in metres, seconds and degrees). The aircraft's main objective is to remain in the air:

(i) $x_9(t) > 0$ for all $t \geq 0$.

While (i) is a primary objective, we also want the aircraft to reach and stay within safe altitude bounds, and remain nearly horizontal (i.e., within 5 degrees):

(ii) $x_9(t) \in [1900, 2100]$ for all $t \geq T$,
(iii) $x_8(t) \in [-5, 5]$ for all $t \geq T$,

with $T > 0$ as small as possible. Additionally, we require adherence to the following safety specifications, based upon physical constraints of an aircraft:

(iv.a) $x_4(t) \in [-10, 10]$ for all $t \geq 0$ (high pitch angles can cause stall; the conservative bound is informed by [25]), and
(iv.b) $x_2(t) \in [-50, 50]$ for all $t \geq 0$ (maximum sink and climb rates; the bound is informed by [25]).

Since it may be impossible to progress towards all objectives at all times, we impose the following importance ranking of specifications, ranked in descending order: (i), (iv.a), (iv.b), (iii) and (ii).

We now design a goodness function corresponding to the above specifications. As previously mentioned, there is no formally correct way for developing a goodness function. Its design depends primarily on the geometric intuition about the problem. We develop the goodness function G from the following conditional functions:

if $z = x_9 < 100$:	x_9 should quickly increase,
else if $\lvert x_4\rvert > 10$	$\lvert x_2\rvert$, $\lvert x_4\rvert$ or both (as needed)

or $\lvert x_2\rvert > 50$:	should quickly decrease,
else if $\lvert x_8\rvert > 5$:	x_8 should quickly approach 0,
else if $x_9 \notin [1900, 2100]$:	x_9 should quickly approach 2000,
else:	$\lvert x_4\rvert$ and $\lvert x_2\rvert$ should remain small

Defining the above functions would be simple if we had full knowledge of the system dynamics. However, the only a priori knowledge about system dynamics comes from physical laws and basic understanding of aircraft control inputs. In particular, we know the following:

(1) $u_1 = \delta_e$ will have the most effect on longitudinal parameters $x_2 = w$ and $x_3 = q$ (by the design of the elevator);
(2) u_2 and u_3 will have the most effect on lateral parameters $x_5 = \beta$, $x_6 = p$ and $x_7 = r$ (by the design of the ailerons and rudder);
(3) $\dot{x}_4 = x_3$ and $\dot{x_8} = x_6$ (by definition);
(4) $\dot{x_9}$ depends directly on x_2 and indirectly on x_4: an increase in either of x_2 or x_4 will increase $\dot{x}_9$ (by longitudinal force definitions). $\dot{x_8}$ depends on mostly on x_6 and x_7 at high pitch angles: a decrease in x_6 leads to an increase in $\dot{x_8}$ (by lateral force definitions);
(5) u_1 does not directly influence x_4 and x_9, but instead acts on them through x_2 and x_3 (by Newton's second law on the longitudinal forces); and
(6) u_2 and u_3 do not directly influence x_8, but instead act on bank angle through x_6 (by Newton's second law on the lateral forces).

We now design goodness functions that result in the required motions, using solely facts (1)–(6). We first want to design a function that results in an increase of x_9. From (5), we know that we cannot directly increase x_9 through control inputs. Thus, from (4), in order to increase x_9 (and $\dot{x}_9$) we know that we want to increase both x_2 and x_4. Since by (3) and (5) we cannot directly influence x_4 through our inputs, but only x_3, we want to find a control input that results in an increase of x_2 and x_3. We do not know exactly from the dynamics which of these two methods we should prioritize, so we will find an input that maximizes the value $\dot{x}_2 + \dot{x}_3$. Analogously, in order to decrease $\dot{x}_9$ and x_9, we want to minimize $\dot{x}_2 + \dot{x}_3$.

What remains is to find methods for decreasing $|x_4|$, $|x_8|$ and keeping $|x_2|$, $|x_4|$ and $|x_6|$ around 0. In order to decrease $|x_4|$, we maximize the increase of $x_3 = \dot{x}_4$ in the direction opposite from x_4, thus providing negative acceleration which will ultimately result in x_4 reducing to 0. In order for $|x_4|$ to remain around 0, we simply use the above method to decrease $|x_4|$ whenever it grows beyond some small number (e.g., $|x_4| > 0.5$). We omit the discussion of designs for the remaining motions, as they are similar to the two above designs.

Using the above methods, our previous rough idea of a goodness function G is now formalized as follows:

if $x_9 < 100$:	$G(x\vert_{[0,t]}, v) = v_3 + v_2$,
else if $\lvert x_4\rvert > 10$ or if $\lvert x_2\rvert > 50$ or if $\lvert x_8\rvert > 5$ or if $x_9 \in [1900, 2100]$:	$G(x\vert_{[0,t]}, v) = m_1 + m_2 + m_3$, where $m_1 = -v_2 \cdot \text{sign}(x_2)$ if $\lvert x_2\rvert > 1$ and $m_1 = 0$ otherwise, and $m_2 = -v_3 \cdot \text{sign}(x_4)$ if $\lvert x_4\rvert > 0.5$ and $m_2 = 0$ otherwise, and $m_3 = -v_6 \cdot \text{sign}(x_8)$ if $\lvert x_8\rvert > 0.5$ and $m_3 = 0$ otherwise.
else:	$G(x\vert_{[0,t]}, v) = (v_3 + v_2) \cdot \lvert v_8\rvert \cdot \text{sign}\,(2000 - x_9(t))$.

We emphasize that this goodness function is not the only possible one. The proposed function has the benefit of being simple to design, but does not have a Lipschitz constant in the sense of Definition 6. Thus, we are unable to directly use the results of Section V to determine good choices of parameters ε and δ. The choice of ε and δ is discussed in Section VI-B. We employ Algorithm 3 with a minor modification as in Remark 4.

B. Sample-rate and wiggle size selection

In selecting parameters ε and δ, we seek to account for scenario-specific issues such as the non-minimum phase of response of x_2 to control u_1. The choice of learning period has to be of sufficient length that we can observe the correct response of an input: most aircraft functionally behave as natural low-pass filters [35], which means that control resolution below a certain period has minimal impact. On the other hand, the learning period must not be too long, or the wiggle size δ too large, that they impact the performance of the control. In this simulation, we decoupled the control of the longitudinal parameters from the lateral [11], and chose the length of the learn-control cycle to be $\varepsilon = 1$ and $\varepsilon = 0.1$ with the learning period, as defined in Remark 4, equal to $\varepsilon' = 0.1$ and $\varepsilon' = 10^{-2}$ for the longitudinal and lateral cases, respectively. We chose $\delta = 5$.

C. Results

The simulation results are presented in Fig. 3. Myopic control strategy performs exactly as desired, even though the true dynamics are not control-affine. In the first few seconds, the aircraft attempts to approach a wings-level flight, primarily controlling for roll angle. After that, it begins its climb until about 50 seconds, where the controller focuses back to the roll angle.

While the peaks and troughs that result in the trajectory leaving the desired bounds could be avoided by a more careful design of the goodness function, we did not make such changes in order to emphasize that the control design performs exactly as intended: as soon as the climb rate or roll angle leave the desired bounds, the controller takes immediate action to return within them.

The aircraft's oscillatory behavior after reaching the altitude bounds is due to the phugoid and Dutch roll aircraft

dynamic modes. Designing a goodness function to control for this behavior is outside of the scope of this paper. We emphasize that myopic control is intended to be used to satisfy the basic control objectives in an emergency; the presented simulation is focused on demonstrating that it successfully performs this task.

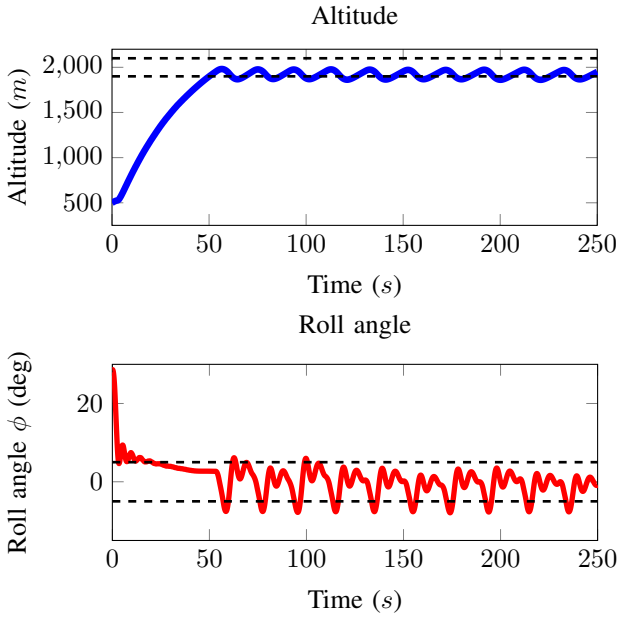

Fig. 3. The simulated aircraft trajectory from the setting of Section VI. The desired system state bounds are denoted by dashed lines. A shortened video corresponding to this simulation is available at `https://goo.gl/a4qYAU`.

REFERENCES

[1] M. Ahmadi, A. Israel, and U. Topcu, "Safety assessment based on physically-viable data-driven models," in *56th IEEE Conference on Decision and Control*, 2017, pp. 6409–6414.

[2] S. Aloni, *Israeli F-15 Eagle Units in Combat*. Osprey Publishing, 2006.

[3] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.

[4] K. B. Ariyur and M. Krstić, *Real-Time Optimization by Extremum-Seeking Control*. Wiley, 2003.

[5] M. Arruda, "Dynamic inverse resiliant control for damaged asymmetric aircraft: Modeling and simulation," Master's thesis, Wichita State University, 2009.

[6] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 15, pp. 3932–3937, 2016.

[7] A. E. Bryson, *Control of Spacecraft and Aircraft*. Princeton University Press, 1994.

[8] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*. Springer, 2004.

[9] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Prentice Hall, 2011.

[10] H.-A. Eckel, S. Scharring, S. Karg, C. Illg, and J. Peter, "Overview of laser ablation micropropulsion research activities at DLR Stuttgart," in *International Symposium on High Power Laser Ablation and Beamed Energy Propulsion*, 2014.

[11] B. Etkin and L. D. Reid, *Dynamics of Flight: Stability and Control*. Wiley, 1995.

[12] F. Fadaie and M. E. Broucke, "A viability problem for control affine systems with application to collision avoidance," in *45th IEEE Conference on Decision and Control*, 2006, pp. 5998–6003.

[13] S. Ganguli, A. Marcos, and G. Balas, "Reconfigurable LPV control design for Boeing 747-100/200 longitudinal axis," in *American Control Conference*, 2002, pp. 3612–3617.

[14] W. L. Garrard and J. M. Jordan, "Design of nonlinear automatic flight control systems," *Automatica*, vol. 13, no. 5, pp. 497–505, 1977.

[15] D. J. A. Hills, A. M. Grütter, and J. J. Hudson, "An algorithm for discovering Lagrangians automatically from data," *PeerJ Computer Science*, vol. 1, 2015.

[16] A. Isidori, *Nonlinear Control Systems*. Springer, 1995.

[17] D. Joyce, "On manifolds with corners," in *Advances in Geometric Analysis*, S. Janeczko, J. Li, and D. H. Phong, Eds. International Press, 2012, pp. 225–258.

[18] S. Khadraoui, H. N. Nounou, M. N. Nounou, A. Datta, and S. P. Bhattacharyya, "Adaptive controller design for unknown systems using measured data," *Asian Journal of Control*, vol. 18, no. 4, pp. 1453–1466, 2016.

[19] D. Krejci, F. Mier-Hicks, R. Thomas, T. Haag, and P. Lozano, "Emission characteristics of passively fed electrospray microthrusters with propellant reservoirs," *Journal of Spacecraft and Rockets*, vol. 54, no. 2, pp. 447–458, 2017.

[20] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*. Wiley, 1972.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[22] W. S. Levine, *The Control Systems Handbook, Second Edition: Control System Advanced Methods*. CRC Press, 2011.

[23] Y. Liu, G. Tao, and S. M. Joshi, "Modeling and model reference adaptive control of aircraft with asymmetric damage," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1500–1517, 2010.

[24] A. Mokhtari, A. Benallegue, and Y. Orlov, "Exact linearization and sliding mode observer for a quadrotor unmanned aerial vehicle," *International Journal of Robotics and Automation*, vol. 21, no. 1, pp. 39–49, 2006.

[25] S. S. Mulgund and R. F. Stengel, "Target pitch angle for the microburst escape maneuver," *Journal of Aircraft*, vol. 30, no. 6, pp. 826–832, 1993.

[26] J. Na and G. Herrmann, "Online adaptive approximate optimal tracking control with simplified dual approximation structure for continuous-time unknown nonlinear systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 4, pp. 412–422, 2014.

[27] Y. Naveh, P. Z. Bar-Yoseph, and Y. Halevi, "Nonlinear modeling and control of a unicycle," *Dynamics and Control*, vol. 9, no. 4, pp. 279–296, 1999.

[28] N. T. Nguyen, K. S. Krishnakumar, J. T. Kaneshige, and P. P. Nespeca, "Flight dynamics and hybrid adaptive control of damaged aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 751–764, 2008.

[29] T. T. Ogunwa and E. J. Abdullah, "Flight dynamics and control modelling of damaged asymmetric aircraft," *IOP Conference Series: Materials Science and Engineering*, vol. 152, no. 1, 2016.

[30] M. Ornik, A. Israel, and U. Topcu, "Control-oriented learning on the fly," arXiv:1709.04889 [math.OC], 2017.

[31] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2011.

[32] R. D. Rimey and C. M. Brown, "Control of selective perception using Bayes nets and decision theory," *International Journal of Computer Vision*, vol. 12, no. 2, pp. 173–207, 1994.

[33] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[34] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Wiley, 2016.

[35] T. R. Yechout, *Introduction to aircraft flight mechanics*. AIAA, 2003.

[36] H.-H. Yeh, S. S. Banda, and P. J. Lynch, "Control of unknown systems via deconvolution," *Dynamics and Control*, vol. 13, no. 3, pp. 416–423, 1990.

[37] S. Zhang and A. J. Yu, "Forgetful Bayes and myopic planning: Human learning and decision-making in a bandit setting," in *Neural Information Processing Systems Conference*, 2013, pp. 2607–2615.

[38] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin, "A general, open-loop formulation for reach-avoid games," in *51st IEEE Conference on Decision and Control*, 2012, pp. 6501–6506.