Safety Verification of Reactive Controllers for UAV Flight in Cluttered Environments using Barrier Certificates

Andrew J. Barry, Anirudha Majumdar, and Russ Tedrake

***Abstract*— Unmanned aerial vehicles (UAVs) have a so-far untapped potential to operate at high speeds through cluttered environments. Many of these systems are limited by their ad-hoc reactive controllers using simple visual cues like optical flow. Here we consider the problem of formally verifying an output-feedback controller for an aircraft operating in an unknown environment. Using recent advances in sums-of-squares programming that allow for efficient computation of *barrier functions*, we search for global certificates of safety for the closed-loop system in a given environment. In contrast to previous work, we use rational functions to globally approximate non-smooth dynamics and use multiple barrier functions to guard against more than one obstacle. We expect that these formal verification techniques will allow for the comparison, and ultimately optimization, of reactive controllers for robustness to varying initial conditions and environments.**

I. Introduction

Imagine a UAV flying through a forest at high speeds. Individual tree trunks fly past in fractions of a second, requiring quick thinking and even faster reaction. With current limitations in onboard instrumentation and computation, fast, reactive controllers and fast actuators are likely essential for staying aloft.

Designing theses types of control systems is hard. Testing and comparing different controllers can be costly, especially so when trees and airplanes meet unexpectedly. Here we discuss a verification technique for reactive controllers that enables the application of some of the rigorous tools from optimization and robust control. We believe that efficient verification of these controllers is the first step towards explicitly optimizing the performance and robustness of control systems that can fly through dense, cluttered environments at high speeds.

In particular, we are interested in verifying vision-based control strategies, since onboard cameras have distinct advantages in terms of range, update rate, and power requirements. However, verifying systems with vision in the loop is potentially difficult because these sensors are fundamentally discrete (pixel by pixel), and often result in control systems which are not smooth. All controllers flying through a forest must at some point choose "left" or "right" around an obstacle, with no options in between. Furthermore, these feedback controllers need not have specific goals in state space, and may only respond to the sensor readings as necessary to avoid a crash – this makes it difficult to apply tools for checking stability to a nominal trajectory as in [20].

The authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology, Cambridge, MA, USA {abarry,anirudha,russt}@csail.mit.edu

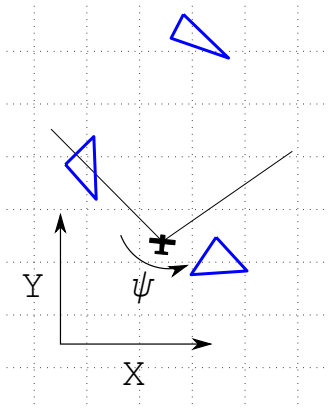

Fig. 1: Example visualization of the simulator showing the coordinate system. Blue polygons represent obstacles and the whiskers attached to the plane represent the arc of the vision system. ψ is the angle between the Y-axis and the aircraft heading (the aircraft shown here has a slightly negative ψ value.)

Thus, instead of computing regions of invariance around trajectories of the system, we verify using a technique more akin to the controllers' goals: we ensure that the systems will not enter any "unsafe" regions in state space.

II. Related Work

Our approach builds directly off the work on safety verification with barrier certificates in [13]. In that work, the authors used sums-of-squares optimization [12] to analyze polynomial dynamical systems. By constructing a Lyapunov-like function, which they called a "barrier certificate" which was positive inside every obstacle and negative in the initial conditions, and verifying that the vector field could not cross into the positive region, they could guarantee the safety of the system. The work that we present here makes two primary modifications to that basic algorithm. First, instead of searching for smooth global polynomial barrier certificates which avoid all of the obstacles, we use a "multiple Lyapunov function"-like approach to verify each obstacle individually. This allows us to verify complex environments by combining many low-degree polynomial certificates, which has major advantages in the scaling and numerical performance of the algorithm. Second, we extend the analysis of polynomial dynamics to rational polynomial dynamics, as rational polynomials are much better suited to represent vector fields which navigate around complex

obstacles. As a result, in this paper we are able to verify substantially more complex systems.

The work is also motivated by previous work on UAV path planning in known environments. [15] uses mixed-integer linear programs to efficiently plan paths for UAVs through polygonal obstacles, and [3] adds the ability to enforce chance constraints for linear Gaussian systems. Here we consider a nonlinear aircraft model in the same polygonal environments, and focus on reactive feedback control instead of path planning.

The Lyapunov-like barrier certificates which we compute here are very analogous to the constructs used in work on motion planning for piecewise affine systems [2]. For the piecewise affine systems, these certificates can even guarantee the execution of linear temporal logic (LTL) specifications [8]. They are also very analogous to the Lyapunov "funnels" computed in [19]. However, the simple reactive controllers that we consider here are difficult to analyze efficiently with piecewise affine analysis or with funnels around trajectories. Instead, we compute barrier certificates for the smooth nonlinear system around the obstacles.

III. Problem Formulation

A. Aircraft Model

We consider the dynamics of a simple, Dubin's-style [5], aircraft model, which is constrained to maintain a constant forward speed. We assume that the yaw dynamics are fast enough that we can control the yaw rate, leading to the following dynamics:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \qquad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -v\cos\psi \\ v\sin\psi \\ u \end{bmatrix}, \qquad (1)$$

where ψ is aircraft yaw, v is the (constant) aircraft forward speed, and u is the control input for $\dot{\psi}$. The model's coordinate system is show in Figure 1.

In order to maintain a constant forward speed, the aircraft must be constantly regulating the pitch and altitude. By assigning a mass m and approximating the lift force as $L = c\alpha v^2$, where c is a constant[1] and α is the angle of attack, and assuming that the plane does not experience side-slip, we can always back out the pitch and angle of attack. The model is not meant to be a complete description of a full 3D aircraft, but does capture some of the essential dynamics and provides a low dimensional state space which can be easily visualized.

For the purposes of the analysis, we will write Equation 1 in the general form:

$$\dot{x} = f(x, u),$$

where $x \in \mathbb{R}^n$ is the n dimensional time-varying state and $u \in \mathbb{R}^m$ is the m dimensional control input. For the aircraft model $n = 3$ and $m = 1$.

[1]Using $L = \frac{1}{2}\rho v^2 A C_L$, with ρ the density of the air, A the surface area of the wing, and C_L the lift coefficient, for which we take the standard small α approximation, $C_L = 2\pi\alpha$, we have $c = \rho A \pi$.

Our goal is to analyze the performance of a vision-based reactive controller on this vehicle while it is navigating in an obstacle field. As in previous work [3], [15], we will use convex polygons to represent the obstacles.

B. Reactive Controller

We assume that the control system has no prior knowledge (e.g. a map) of the obstacle field and that it must rely only on local vision-based sensing. Here we will model the vision system as a series of 101 range measurements arranged in a +/- 50 degree arc in front of the vehicle.

The following is a simple, reactive controller which evaluates the 101 range measurements and determines a yaw rate command for the system:

Input:
- r the array of ranges given by the vision system (indexed by i)
- $\hat{r}$ the array of unit vectors pointing in the direction of the area viewed by each pixel (indexed by i)
- $\hat{x}$ the unit vector pointing in the direction of the plane's velocity (can be computed directly from ψ)
- $\hat{G}$ the unit vector pointing towards a nominal goal far in front of the craft
- c_1, c_2, c_3, positive constants

Output:
- u, the control action

Algorithm:
- for each range value, 1 through N,
  - if an obstacle was detected at pixel i,
    - Let the "force" F_i applied on the plane from this pixel be: $F_i = -c_1 \hat{x} \cdot \hat{r}_i \frac{(1-(r/c_3))^2}{(1+(r/c_3)^3)}$
- Let F_G be the "force" towards the faraway point: $F_G = -c_2 \hat{G} \cdot \hat{x}$
- The output of the controller is given by: $u = F_G + \sum_{i=1}^{N} F_i$

Listing 1: Control algorithm. Note that the controller is reactive, and does not require the full state (complete descriptions of obstacles and complete global position).

This simple controller performs fairly well in simulations of the obstacle field based only on hand-tuned gains, although it will occasionally crash into obstacles. It is important to note that the controller makes decisions based on a fundamentally discrete set of sensors, which change discontinuously with the state of the vehicle (e.g. when an individual range sensor makes or breaks contact with an obstacle). Therefore, the resulting closed-loop dynamics of this system are not smooth.

C. Safety Verification

Given the simple aircraft model and the reactive controller, the goal of this paper is to efficiently evaluate the performance of the controller, which we measure as the volume of initial conditions in state space for which we can prove that the system will not collide with an obstacle. We will accomplish this by constructing a random obstacle field and analyzing the resulting closed-loop dynamics of the controller in state space, even though the controller does not have any explicit notion of state. This approach allows us to compare different reactive controllers by generating statistics based on each controllers' performance throughout a range of initial conditions and environments. In this way, we hope

to ultimately formulate an optimization over the combination of sensors and reactive controllers to maximize robustness in a variety of environments.

IV. Approach

A. Global Smooth Approximation of Dynamics

The discontinuities in sensor readings, and resulting discontinuities of the reactive control system pose a potentially fundamental, and realistic, challenge for safety verification. Previous work on mobile robots operating around polygonal obstacles often made use of cell decompositions and piecewise models of the dynamics, which can lead to very powerful and efficient planning and control algorithms [1], [2], [8]. However, the reactive controller does not fit nicely into a parsimonious piecewise smooth approximation; in fact the sensor readings threaten to tessellate the space into millions of pieces because the discontinuities can happen at each sensor, spaced 1 degree apart in orientation.

In order to take advantage of powerful tools for verification of smooth, polynomial dynamical systems, we will instead derive our best smooth approximation of the resulting vector field of the closed-loop system. In fact, the discontinuities in the sensors prevent approximation of the vector field through local analysis (e.g. Taylor approximation), but sampling the vector field reveals that the closed-loop dynamics are relatively smooth; the most challenging effects are in locations where the vehicle splits between going left or right around an obstacle. In fact, the vector field is effectively singular at every obstacle, and on non-trivial manifolds approaching the obstacle. Polynomial models of reasonable degree have a very difficult time describing this behavior; rational polynomial models, however, are better suited to representing functions of this form.

For this reason, we approximate the dynamics of the closed-loop system as $\hat{\dot{x}} = \frac{p(x)}{q(x)}$ where $p(x)$ and $q(x)$ are polynomials with optimized coefficients. We let $p(x)$ be a $n \times 1$ vector of polynomials and $q(x)$ be a single scaling polynomial. We find the set of coefficients by minimizing the quantity $(q(x_i)\dot{x}_i - p(x_i))^2$ where x_i and $\dot{x}_i$ are the sampled state and the resulting closed-loop dynamics respectively. Finally, we enforce that $q(x) > 0, \forall x$ using a sums-of-squares constraint. This class of functions was chosen so that it supports the verification framework presented below. This approximation is able to remain true to the system despite the complexities of multiple homotopy classes of paths and the dynamics' non-smooth transitions between them.

B. Safety Verification with Barrier Certificates

Verification of nonlinear control systems has been a fertile area of research in recent years, spurred on by advances in computational approaches to the problem [12]. In particular, Sums-of-Squares (SOS) optimization has emerged as a computationally tractable approach to the verification and synthesis of nonlinear controllers [19]. SOS techniques have been used to compute and verify Lyapunov functions for systems governed by polynomial dynamics, guaranteeing that the control system is stable (in the Lyapunov sense) either to a fixed point of the system, or to a time-varying trajectory. The technique is based on the fact that one can check positive-definiteness of a multivariate polynomial by expressing it as a "sum of squares". This can be used to check the Lyapunov conditions ($V(x)$ is positive definite and $\dot{V}(x) < 0$), by casting the sums-of-squares problem as a convex optimization problem, making it accessible to efficient solvers (such as interior point methods).

Although checking Lyapunov stability is a natural approach to follow in several settings, the method is not well suited to controllers that are purely reactive in nature, such as the one presented in our work. This is because reactive controllers, which attempt to avoid obstacles while nominally moving in some direction, do not have an explicit notion of a goal state/trajectory that the system is being driven to. Thus, computing Lyapunov functions for specific "goal" states/trajectories will not yield meaningful results. A far more natural approach is to compute *Barrier functions*, which guarantee that the closed loop control system does not collide with any obstacles. Barrier functions have been used previously in order to verify safety for both continuous and hybrid systems [14].

Formally, given a nonlinear system whose closed-loop dynamics are described by:

$$\dot{x} = f(x)$$

with state $x \in X$, initial condition set $X_0 \subset X$, and obstacle set $X_{u,i} \subset X$ (i indexes the separate obstacles), a Barrier function, $B : X \mapsto \mathbb{R}$, must satisfy the following conditions:

$$\begin{aligned} &B(x) \leq 0, \forall x \in X_0 \\ &B(x) > 0, \forall x \in X_{u,i}, \forall i \\ &\dot{B}(x) \leq 0, \forall x \in X \ s.t. \ B(x) = 0 \end{aligned} \tag{2}$$

The search for Barrier functions can be cast as a Sums-of-Squares optimization problem for systems with polynomial closed loop dynamics, where one searches over a suitable family of candidate functions (such as all polynomials in x of degree less than or equal to d). However, in order to do so, the conditions in (2) must be modified slightly in order to ensure that the resulting SOS optimization problem is convex. This can be ensured by imposing a stronger condition on the derivative of the Barrier function: $\dot{B}(x) \leq 0, \forall x \in X$.

Assume that the sets X_0, X_u and X are semi-algebraic sets, described as follows:

$$\begin{aligned} X_0 &= \{x | 1 - X_{0,ineq} \geq 0\} \\ X_{u,i} &= \{x | 1 - X_{u,i,ineq} \geq 0\} \\ X &= \{x | 1 - X_{ineq} \geq 0\} \end{aligned}$$

where $X_{0,ineq}$, $X_{u,i,ineq}$, and X_{ineq} are polynomial expressions (note that although these define algebraic sets, semi-algebraic sets can be defined in a similar manner by simply imposing more than one inequality constraint in the descriptions of the sets).

In order to check the conditions in (2) using SOS optimization, it is sufficient to find a polynomial Barrier

function $B(x)$ and sums-of-squares Lagrange multipliers $L_0(x), L_{u,i}(x), L(x)$ such that the following expressions are sums-of-squares:

$$\begin{aligned} &-B(x) - L_0(x)X_{0,ineq} \\ &B(x) - L_{u,i}(x)X_{u,ineq}, \forall i \\ &-\dot{B}(x) - L(x)X_{ineq} \end{aligned}$$

Note that, as mentioned in [14], the non-convex version of the problem (2) can also be tackled by alternatively searching over Barrier functions and Lagrange multipliers.

An important observation makes it possible to search for Barrier functions over a less conservative class of barriers. Relative to [14], we compute separate Barrier functions for each obstacle. Thus, a separate Sums-of-Squares optimization problem is solved for each obstacle, yielding a set of Barrier functions, $B_i(x)$. Then,

$$B(x) = \max_i \quad B_i(x)$$

is a valid Barrier function that guarantees that the closed loop system never collides with any obstacle. $B(x)$ trivially satisfies the first two conditions in (2). Further, even though $B(x)$ may not be differentiable, the condition that $B_i(x)$ $(\forall i)$ is decreasing along trajectories of the system ensures that $B(x)$ is also decreasing for all trajectories of the system. This allows us to impose fewer conditions on each $B_i(x)$, hence making our final Barrier certificate less conservative.

Finally, although the approximation of our dynamics is done through functions of the form $\dot{x} = \frac{p(x)}{q(x)}$, where $p(x)$ is a vector of polynomials and $q(x)$ is a single sums-of-squares polynomial (as mentioned in IV-A), it is still possible to check the $\dot{B}(x) \leq 0$ condition. Since $\dot{B}(x) = \frac{\partial B(x)}{\partial x}\dot{x} = \frac{\partial B(x)}{\partial x}\frac{p(x)}{q(x)}$, we can simply multiply both sides of the inequality $\dot{B}(x) \leq 0$ by $q(x)$. Since $q(x)$ is sums-of-squares, this does not change the inequality. Hence, the condition that needs to be checked becomes $\frac{\partial B(x)}{\partial x}p(x) \leq 0$. This important observation allows us to broaden our class of functions considerably (relative to polynomial dynamics), allowing for better approximation of the system dynamics.

C. Barrier Functions for Verification

This powerful framework for computing Barrier functions using SOS optimization can be applied to the verification of output feedback controllers. Although the controller is not aware of the full state of the system, in order to provide guarantees for our controller, we make the full state (including positions of the obstacles) available to the verification procedure in order to evaluate the performance of the controller. In this context, we find verification useful for comparing controllers and as guidance for their design, so as noted in Section VI-B, the requirement that the verifier must know the full system state is not particularly taxing.

While it might seem that barrier functions would ideally be local around each obstacle, we find that this is almost never the case. Consider, for example, a smooth vector field that passes both to the left and to the right of an obstacle, such as the one show in Figure 2. By the intermediate value theorem, the field that points to the left and to the right must somewhere point directly at the obstacle. Rightly so, the barrier function will not verify that region as safe. Unfortunately, as we move further from the obstacle, the same requirement remains, and we are forced to conclude that the barrier function cannot be both completely local and verify two paths around an obstacle.

We note, however, that a quartic parameterization of the barrier function will allow the 0-level set to extend to infinity in some directions, so while it will not be local, a quartic function can still provide a region of safety near the obstacle. Thus, in practice with multiple barrier functions, we expect to find "channels" of safety between obstacles.

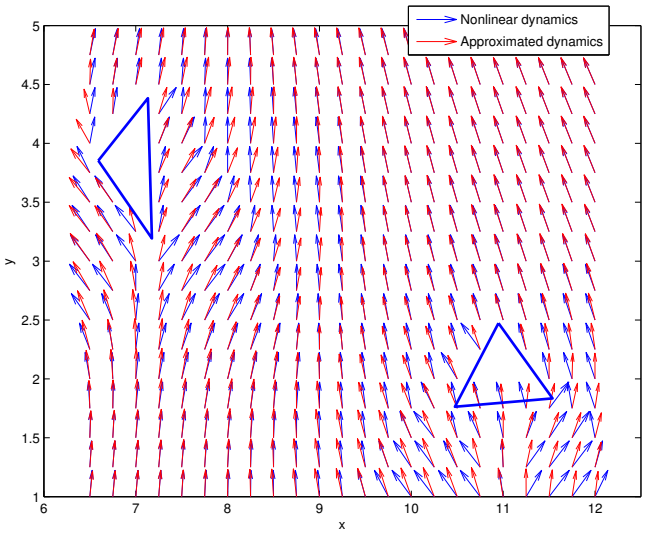

Fig. 2: Comparison of approximated (red) and true dynamics (blue) around obstacles. To create this plot, we take a slice of the state space at $\psi = 0$ and compute the system dynamics at each sample point. We then recompute $\dot{x}$ by simulating the dynamics forward for a small time with $\dot{\psi}$ held constant to its value at the sample point. Note that if we did not simulate forward, $\dot{x}$ would always be 0 since $\psi = 0$ for all points in our slice.

V. Results

A. Approximate Dynamics

We use YALMIP [10] and SeDuMi [18] to compute approximate dynamics using 5th and 4th order polynomials for $p(x)$ and $q(x)$ respectively in the approximate dynamics $\dot{x} = \frac{p(x)}{q(x)}$, resulting in a vector field that captures the nuances of the closed-loop dynamics. Importantly, the approximation captures the the system's ability to travel around the obstacles using multiple different paths (Figure 2).

B. Controller Verification

The performance of the controller described in III-B was verified using the Barrier function approach described in IV-B for two different scenarios. The first scenario, depicted in Figure 3 contains a forest of three polygonal obstacles. For the sake of illustration, we evaluate the Barrier function at

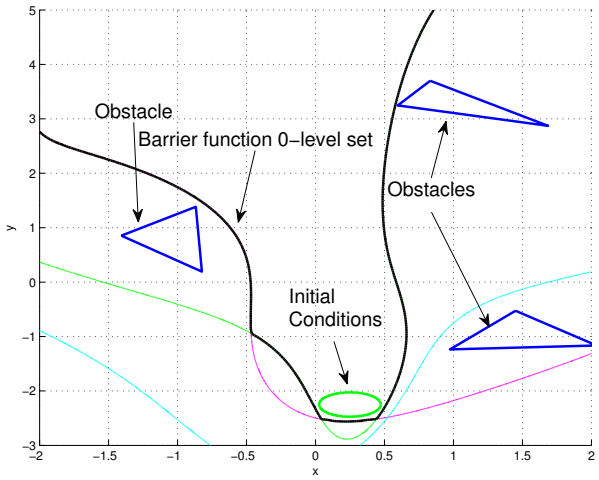

Fig. 3: Barrier function for the controller operating in a "forest" with three obstacles. The figure shows the 0-level set in the $\psi = 0$ "slice" of the Barrier function (black), which successfully separates the set of initial conditions from the unsafe regions. Also shown in cyan, green, and purple are the individual barrier functions for each obstacle. The obstacles are shown in blue.

$\psi = 0$, allowing us to plot the zero-level set of the "slice" of the Barrier function (corresponding to $\psi = 0$). As the figure demonstrates, the Barrier function provides us with a certificate of non-collision with the obstacles, for the given set of initial conditions. Note that the zero-level set of the Barrier function is not differentiable. This is because our Barrier function is evaluated by computing the maximum of three functions, as described in Section IV-B.

Figure 4 shows similar results for a scenario where the plane is navigating an "Infinite Corridor". Again, we plot the 0-level set in the $\psi = 0$ "slice" of the Barrier function. As before, the 0-level set guarantees the safety of the closed-loop dynamics for the given set of initial conditions. As one would expect, the Barrier function is symmetric about the y-axis for the $\psi = 0$ slice.

Figure 5 provides another perspective of the Barrier function in the "Infinite Corridor" scenario by plotting the zero-level set along a slice of the function at a given value of y. The vector field along the zero level set is plotted, demonstrating that it points inwards everywhere, thus guaranteeing that one never leaves the region. An interesting feature that the plot also illustrates is that the zero-level set is asymmetric about the $x = 0$ axis. This corresponds with intuition, as one would expect it to be safer when the plane is close to the right edge of the corridor and pointed left than it is if the plane were pointed right. Hence, when x is positive, the zero-level set extends closer to the obstacle when ψ is positive (recall from Figure 1 that $\psi > 0$ when the plane is turned towards the left).

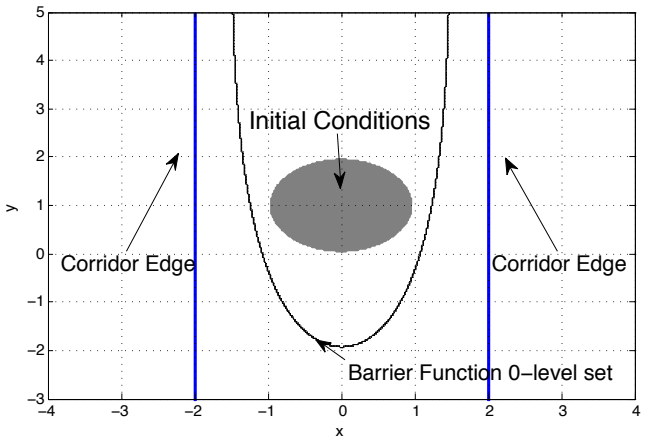

Fig. 4: Barrier function for the controller operating in an "infinite corridor". The figure shows the 0-level set in the $\psi = 0$ "slice" of the Barrier function. As the figure shows, the Barrier function successfully guarantees the safety of the controller for the given set of initial conditions.

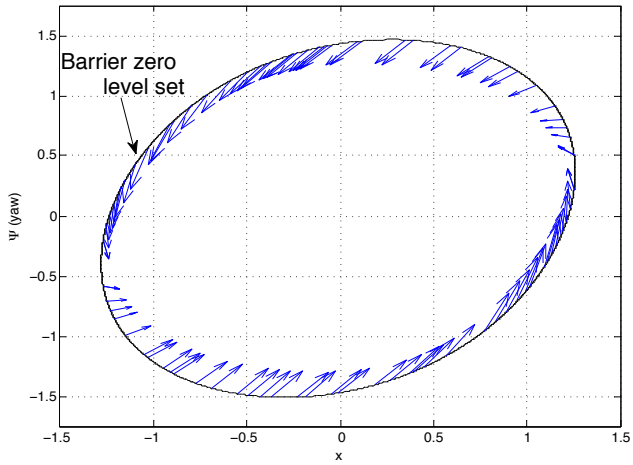

Fig. 5: Validation of the Barrier function for the "infinite corridor" case. The plot shows the vector field in the $\psi - x$ plane, sampled along the 0-level set of the barrier. As the figure demonstrates, the vector field is pointing inwards everywhere along the level set.

VI. Discussion

A. Limitations of Approximating Dynamics Using Rational Functions

As mentioned in Section IV-A, our model of the closed loop dynamics is of the form $\dot{x} = p(x)/q(x)$, where $p(x)$ is a 3×1 vector of polynomials and $q(x)$ is a single sums-of-squares polynomial. Although this approximation works reasonably well for our system, other approaches to this problem may yield better approximations. The main difficulty in approximating the dynamics is that the vector field changes sharply close to an obstacle. Although our model captures most of this sharp variation through $q(x)$ when there are a small number of obstacles, it requires increasingly high

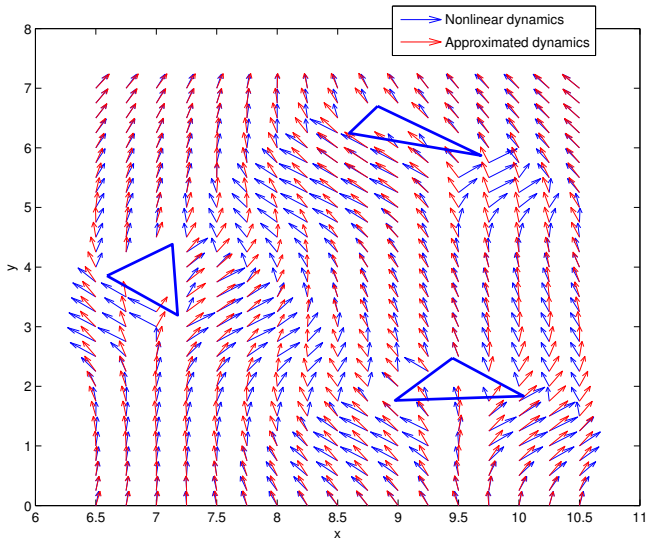

Fig. 6: Global rational approximation with three obstacles. Note that while the approximation is good in most of the space, it degrades near the obstacles. Fitting with a higher order rational function would help mitigate this issue, but would also cause the search for barrier functions to become numerically more difficult. We again show a slice at $\psi = 0$ as described in Figure 2.

order rational functions as more obstacles are added, as can been seen in Figure 6. It is conceivable that a better approach would be to have piecewise polynomial approximations of the dynamics. However, the difficulty with this method is that the approximation would have to automatically break up the space into pieces where the polynomial approximations are good. Further, it would be difficult to ensure that the resulting (global) vector field is differentiable, which is necessary for us to check that $\dot{B}(x) \leq 0$.

B. Reactive Control

Reactive controllers have enjoyed a lot of success in robotic applications and continue to be among the most popular control techniques for navigation tasks [4] [6] [11]. The main advantage to reactive control is that the agent (UAV in our case) needs to extract only a limited amount of information from the environment in order to carry out the task. This reactive framework seems to be nature's preferred method for insect navigation and has inspired several biomimetic control strategies [11]. Further, an explicit plant model is not required in order to carry out this output feedback control. However, it is often the case that different reactive controllers are difficult to compare (even when applied to the same task scenario). The framework for the verification of reactive controllers via Barrier functions presented in this paper allows one with a good method for comparing reactive controllers. In particular, one can run the controllers in a given obstacle field and compare the resulting barriers. For each controller (and for a given obstacle field), one can try to maximize the size of the initial condition set such that the controller does not fail (i.e. does not result in a collision with the obstacle). The controller that is most robust to variations in initial conditions is more desirable than others. The computationally efficient SOS verification allows one to run the controllers for different obstacle distributions and thus compile statistics for the performance of the different controllers.

The success and intuitive appeal of reactive controllers notwithstanding, reactive controllers often fail because of their short-sighted nature. A different approach is to create longer terms *motion plans* for the robot. Motion planning has been the subject of significant research in the robotics literature and has given rise to a set of very powerful tools that have enjoyed success in varied application domains [17] [16]. Algorithms like the RRT [9] and RRT* [7] can handle large state space dimensions and can handle differential constraints. More recently, the LQR-Tree algorithm [19] has combined these randomized motion planning algorithms with tools from control theory and sums-of-squares programming in order to generate motion plans that exploit the natural dynamics of the system and augment them with feedback controllers.

While reactive control and motion planning seem at first glance to be conceptually at odds with each other, we believe that a combination of tools from the two areas is necessary in the domain of UAV flight through cluttered environments. The long-term planning algorithms can be used to guide the general trajectory of the path that the robot follows while a lower level reactive control layer can be used to avoid obstacles. The tools presented in this paper could be used to allow for such a combination of planning and reactive control. One could envision having a *library* of reactive controllers that one pre-computes off-line. Each controller in the library can be tuned to handle a particular local distribution of obstacles. For each controller (and the corresponding obstacle field), one could compute barrier functions that provide guaranteed regions of safety. Then, at runtime, a higher level planner can go through sequences of possible reactive control switching strategies and find a sequence that is safe by looking up the corresponding barrier functions. In this manner, one avoids being too short-sighted by planning ahead, while still retaining the benefits of having reactive controllers.

C. Dealing with Bounded Uncertainty

The controller verification method using Barrier certificates can be easily extended to deal with uncertainties in both the perception system and dynamics model. In particular, suppose that due to perceptual uncertainty, we only have an imprecise estimate of the position of each obstacle, but that we can bound the uncertainty in the position. Then, one can leave the positions of obstacles as free variables that are constrained to lie within the limits imposed by the uncertainty bound. These constraints can be added to the SOS program presented in IV-B. The resulting barrier functions will verify the controller for all possible positions of the obstacles. A similar procedure can be used to handle model uncertainties and is presented in [14].

D. Barrier Function Optimization

Finally, we propose to optimize the size of the barrier functions. Our current work finds valid barriers, but does not attempt to maximize the volume of those solutions. One natural way to do this would be to fix the value of the barrier function to be zero at some point (x_0) in the initial condition set and greater than β inside the obstacles. Then, we can ask for the derivative of the barrier function to be negative on the β level set (instead of the zero level set) and maximize β. Following the notation of Section IV-B, the sums-of-squares program is:

$$\begin{aligned} \underset{\beta,B,L_0,L,L_{u,i}}{\text{maximize}} \quad & \beta \\ \text{subject to} \quad & B(x_0) = 0 \\ & B(x) - \beta - L_{u,i}(x)X_{ineq} \geq 0 \\ & -\dot{B}(x) - L(x)(\beta - B(x)) \geq 0 \\ & L_{u,i} \geq 0 \end{aligned} \tag{3}$$

A few things are worthy of note here. First, the barrier function needs to be normalized somehow in order to prevent β from being artificially maximized by increasing the coefficients of $B(x)$. One way to do this would be to fix the value of $B(x)$ at two points inside the initial condition set instead of just one. Second, this optimization program is bi-linear in the decision variables and cannot be solved using a single sums-of-squares program. One has to alternate between searching over $L(x)$ and the rest of the decision variables. This is similar to the case where we want to ensure that the derivative of the barrier function is negative only on the zero-level set (as described in [14]). Finally, we should note that increasing β is only a surrogate for increasing the "size" of the zero-sublevel set of the barrier function. Other measures may provide better results and should be adopted as necessary.

VII. Conclusion

In this paper, we have presented a method for applying formal tools from Sums-of-Squares optimization in order to verify reactive controllers for UAV flight through cluttered environments. In particular, given a nonlinear control system with a perceptual model, we search for polynomial Barrier functions that verify that the closed-loop dynamics do not result in collisions with obstacles. The efficiency of the Sums-of-Squares optimization methods allow us to compare different reactive controllers for different distributions of obstacles. These methods can be extended in order to deal with uncertainties in both the plant model and the perception system. We believe that the introduction of these tools to UAV control will help tap the as yet unexploited potential for UAVs to operate at high speeds through cluttered environments.

VIII. Acknowledgements

This work was supported by ONR MURI grant N00014-09-1-1051. Andrew Barry is partially supported by the National Science Foundation Graduate Research Fellowship. Anirudha Majumdar is supported by the MIT Intelligence Initiative.

References

[1] H. Ayanian and V. Kumar. Abstractions and controllers for groups of robots in environments with obstacles. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3537–3542. IEEE, 2010.

[2] C. Belta, V. Isler, and G. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Automatic Control*, 21(5):864–874, 2005.

[3] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006*, page 7 pp., june 2006.

[4] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1984.

[5] L. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[6] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek. Autonomous legged hill and stairwell ascent, November 2011.

[7] S. Karaman and E. Frazzoli. Incremental sampling-based optimal motion planning. *submitted to Robotics: Science and Systems*, 2010.

[8] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1):287–297, 2008.

[9] J. Kuffner and S. Lavalle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.

[10] J. Lofberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions On Automatic Control*, 54(5):1007–, May 2009.

[11] T. Neumann and H. Bulthoff. Behavior-oriented vision for biomimetic flight control. In *Proceedings of the EPSRC/BBSRC international workshop on biologically inspired robotics*, pages 196–203, 2002.

[12] P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, May 18 2000.

[13] S. Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117 – 126, 2006.

[14] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB Users guide*, 2.00 edition, June 1 2004.

[15] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1936–1941. IEEE, 2002.

[16] P. Sermanet, M. Scoffier, C. Crudele, U. Muller, and Y. LeCun. Learning maneuver dictionaries for ground robot planning. In *Proc. 39th International Symposium on Robotics (ISR08)*, 2008.

[17] A. Shkolnik. *Sample-Based Motion Planning in High-Dimensional and Differentially-Constrained Systems*. PhD thesis, MIT, February 2010.

[18] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625 – 653, 1999.

[19] R. Tedrake, I. R. Manchester, M. M. Tobenkin, and J. W. Roberts. LQR-Trees: Feedback motion planning via sums of squares verification. *International Journal of Robotics Research*, 29:1038–1052, July 2010.

[20] M. M. Tobenkin, I. R. Manchester, and R. Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *Proceedings of the 18th IFAC World Congress, extended version available online: arXiv:1010.3013 [math.DS]*, 2011.