Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

PolySafe: A Formally Verified Algorithm for Conflict Detection on a Polynomial Airspace

Brendon K. Colbert * **J. Tanner Slagel** * **Luis G. Crespo** *
Swee Balachandran ** **César Muñoz** *

* *NASA Langley Research Center, Hampton, VA, USA*
** *National Institute of Aerospace, Hampton, VA, USA*

Abstract: This paper presents a strategy for verifying that an aircraft following a polynomial path complies with a given set of safety criteria in continuous time. Such criteria ensure that a minimal separation between the aircraft and a set of obstacles, which can be either static or dynamic, is maintained. Dynamic obstacles are also assumed to follow a known polynomial path. Dynamic obstacles may, for example, correspond to a separation volume around another flying aircraft. In the most general case, the separation criteria vary in time depending upon the position and relative velocity between the aircraft and the obstacle. The efficiency and scalability of the proposed algorithm, to be called PolySafe, make it suitable for real-time conflict detection and path re-planning of aircraft flying in a complex and crowded airspace. PolySafe has been formally verified to guarantee the detection of conflicts within a finite time horizon.

Keywords: trajectory and path planning, safety, autonomous systems

Advances in Unmanned Aircraft Systems (UAS) promise to carry out missions such as search and rescue, surveillance, data gathering and package delivery operations autonomously. However, autonomous UAS will be required to meet existing safety requirements before they are allowed to share the airspace with existing aircraft. The concept *sense and avoid* is defined in the Federal Aviation Administration (FAA) Sponsored Sense and Avoid Workshop (2009) as "the capability of a UAS to remain well clear from and avoid collisions with other airborne traffic." The notion of UAS being *well clear* is cast in terms of a well-clear volume in Cook et al. (2015). Specifically, a well-clear violation occurs when an aircraft trajectory enters the well-clear volume of another aircraft.

Changing flying conditions such as weather or cross-wind may force an aircraft to change its nominal trajectory. Such changes might lead to unanticipated well-clear volume violations. Thus, there is a need for determining, both reliably and efficiently, if the current flight path is in well-clear violation so a new guidance law can be computed as early as possible. Such an algorithm must be fast so that a re-planning algorithm is given sufficient time to find a new safer path, but it should also provide the guarantee that no safety violations go undetected. Furthermore, the algorithm should avoid diagnosing "false positives" that will trigger unnecessary replanning.

Current algorithms for detecting conflicts with geofences include Stevens and Atkins (2016) and Dill et al. (2016). The former paper proposes a method of shared control between a pilot and autopilot, by which control commands resulting in a fence breach are modified to keep the UAS within a given geofence. The latter paper proposes a method by which geofence conflicts are detected, and violation of geofence constraints are prevented using vehicle contingency maneuvers and flight termination.

Regarding air traffic conflicts, the Traffic Alerting and Collision Avoidance System (TCAS) has been adopted by the commercial aviation industry as the standard collision avoidance system, see Williamson and Spencer (1989). The system provides alerts to pilots if a collision threat is detected. In recent years, the FAA has formed a group led by Lincoln Laboratory to define a new collision avoidance system. The system, called ACAS X, relies upon probabilistic and computer-based optimization techniques, see Kochenderfer et al. (2012). In the case of UAS, DAIDALUS (Detect and Avoid Alerting Logic for Unmanned Systems) is the reference implementation of detect and avoid minimum operational requirements for large UAS, see Narkawicz et al. (2018). DAIDALUS provides alerting and maneuver guidance logic for a well-clear volume defined by distance and time thresholds. These air traffic conflict systems implement tactical approaches. In the case of TCAS and DAIDALUS, aircraft states are projected using relatively simple kinematic models. These approaches can be integrated into path planning algorithms using discrete search techniques such as A* and Rapidly Exploring Random Trees (RRT) as in Balachandran et al. (2017).

This paper proposes a formally verified algorithm called *PolySafe* for determining if a polynomial flight path exhibits enough separation from a possibly moving obstacle. Obstacles are characterized as semi-algebraic sets, i.e., a region of the airspace satisfying a set of polynomial constraints that might vary in a continuous time. This characterization allows determining if a given polynomial path intersects static obstacles such as buildings or mountains, or dynamic obstacles such as the well-clear volume

of another aircraft. Methods in Colbert et al. (2019), for instance, return a semi-algebraic set containing a given cloud of points from sensor measurements of an obstacle. Hereafter, it is assumed that all the aircraft and obstacles follow known polynomial paths.

Polynomial paths will be denoted as

$$r(t) = [x(t), y(t), z(t)]^\top, \tag{1}$$

where $x(t), y(t), z(t) \in \mathbb{R}[t]$ are univariate polynomial functions with respect to time $t > t_0 \in \mathbb{R}$. This path is prescribed by a guidance law relative to a reference frame on the ground. The trajectory $r(t)$ is assumed to account for the effects of cross-winds, and for the presence of nearby obstacles. It is also assumed that the desired polynomial path can be tracked closely by means of a flight control system and a set of pilot commands.

The rest of this paper is organized as follows. Section 1 describes the problem statement and goals of the proposed algorithm while Section 2 describes our solution approach. Section 3 characterizes static and dynamic obstacles including the well-clear volume. Section 4 presents the PolySafe algorithm and discusses efficient numerical tools required for its implementation. Section 5 discusses the formal verification of safety properties of PolySafe. Section 6 evaluates the proposed algorithm through simulation. Finally, Section 7 provides concluding remarks and describes future work.

1. PROBLEM STATEMENT

In this work, the airspace is represented as the 3-dimensional Euclidean space $\mathbb{R}^3$. Each of n obstacles is denoted $O_i(t) \subset \mathbb{R}^3$, for $i = 1, \dots n$, with

$$O_i(t) \triangleq \big\{(x, y, z) : p_{i,j}(x, y, z, t) \leq 0,\ \forall j = 1, ...m_i\big\}, \tag{2}$$

where $p_{i,j}(x, y, z, t)$ is a polynomial function. The function $O_i(t)$ can represent a large class of obstacles including the intersection of parallelepipeds, convex polygons, spheres, and pyramids. In addition, it can also describe the time-dependent well-clear volume defined in Munoz et al. (2014).

The polynomial path $r(t)$ will be called conflict-free if $r(t) \cap (\cup_{i=1}^n O_i(t)) = \emptyset$, for all $t \in [t_0, t_f]$. In this setting, the problem statement can be stated as follows: given a polynomial path $r(t)$ and the set of obstacles $O_i(t)$, for $i = 1, \dots n$, first determine if such a path is conflict-free for all times in the interval $[t_0, t_f]$. Second, if the path is not conflict free, determine

$$t^* = \arg \min_{t \in [t_0, t_f]} \big\{t : r(t) \in \cup_{i=1}^n O_i(t)\big\}. \tag{3}$$

In Formula (3), t^* is the time instance when conflict first occurs. Based on t^*, $r(t^*)$, and the particular obstacle the nominal path $r(t)$ is in conflict with, a suitable new path $\hat{r}(t)$ can be computed by a conflict resolution logic of the underlying guidance law. This paper focuses on conflict detection only. A (re)planning algorithm, to work in unison with PolySafe, will be presented elsewhere. Polysafe should be run continuously in time over a receding time horizon $[t_0, t_f]$ while the polynomial paths of all aircrafts are updated according to measurements of their position and velocity over the past time interval $[t_0 - \Delta, t_0]$ for $\Delta > 0$.

2. SOLUTION APPROACH

A violation occurs at a time t if all polynomial inequalities defining an obstacle are non-positive at that time. In particular, the aircraft with polynomial path $r(t)$ is in conflict with obstacle $k \in [1, \dots n]$ during the time interval $[t_0, t_f]$ if and only if there exists a time $\tilde{t} \in [t_0, t_f]$ such that

$$p_{k,j}\Big(x(\tilde{t}), y(\tilde{t}), z(\tilde{t}), \tilde{t}\Big) \leq 0,\ \forall\ j = 1, \dots m_k. \tag{4}$$

Denote as $\tilde{t}_k$ the minimum time for which (4) holds. If (4) does not hold make $\tilde{t}_k = \infty$. Therefore,

$$t^* = \min_{k=1,\dots n} \tilde{t}_k. \tag{5}$$

Determining the times when a violation occurs is difficult because they may occur at any point within a set of infinitely many points. However, the detection approach used in this paper only considers the first occurrence of such an event. This time instance must be at a real root of one of the m_k polynomial inequalities in (4). Therefore, it is necessary to find the real roots of all the m_k polynomials $p_{k,j}$, sort them in increasing order, and then evaluate $p_{k,j}$ at all of them. The smallest root at which all $p_{k,j} \leq 0$ defines $\tilde{t}_k$. If no such point is found, there is no conflict between the aircraft and the k-th obstacle, so $\tilde{t}_k = \infty$. The next section describes how to characterize obstacles.

3. MODELING OBSTACLES AS SEMIALGEBRAIC SETS

This section focuses on modeling static and dynamic obstacles as possibly time-varying semi-algebraic sets. The first assumption is that obstacles can be defined, or closely bounded by polynomial inequalities in euclidean space.

3.1 Static Obstacles

Any convex polygon can be described as the intersection of half-planes, and the half-planes themselves can be used to define a semi-algebraic set. For instance, a cube whose bottom left corner occurs at the point $(1, 2, 5)$ with side lengths of 5 can be defined by the intersection of the following six half-planes,

$$\begin{aligned} &1 - x \leq 0, \quad x - 6 \leq 0, \quad 2 - y \leq 0, \\ &y - 7 \leq 0, \quad 5 - z \leq 0, \quad z - 10 \leq 0. \end{aligned} \tag{6}$$

Let O be a semi-algebraic set representing a static obstacle with m half-planes, then O can be defined as

$$O = \{(x, y, z) : p_j(x, y, z) \leq 0\ \forall j = 1, ..., m\},$$

where p_j are polynomials that represent the desired half-spaces. In the example above, $p_1(x, y, z) = 1 - x$, ..., and $p_6(x, y, z) = z - 10$. Additional polynomial inequalities can be considered, e.g., a sphere of radius r centered at (5,2,-5) can be defined by $(x - 5)^2 + (y - 2)^2 + (z + 5)^2 - r^2 < 0$. Such an obstacle would be defined by the semi-algebraic set

$$O = \{(x, y, z) : (x - 5)^2 + (y - 2)^2 + (z + 5)^2 - r^2 \leq 0\}.$$

The intersection of any of the above sets is, as expected, a representable obstacle geometry. The above framework enables describing a wide range of shapes with varying levels of refinement. As long as O fully contains the true geometry of the obstacle, false negatives (i.e., missing the

detection of a conflict) will not occur. The greater the offset between the true obstacle and its outer bounding set, the more false positives (i.e., declaring an inexistent conflict) might occur. As such, tight outer approximations of the obstacle geometry are preferable.

3.2 Dynamic Obstacles

Time-varying obstacles will be represented as time-varying semi-algebraic sets:

$$O_d(t) = \{(x, y, z) : p_j(x, y, z, t) \leq 0,\ \forall j = 1, ..., m\},$$

where $p_j(x, y, z, t)$ is a polynomial function. For instance, a growing spherical obstacle centered at the origin whose radius $r > 0$ increases at $\dot{r}$ meters per second, while moving in the positive x direction at a rate of $\dot{x}$ meters per second is given by

$$O_d(t) = \{(x, y, z) : (x - \dot{x}t)^2 + y^2 + z^2 - (r + \dot{r}t)^2 \leq 0\}.$$

This framework enables the characterization of more complex obstacles such as the well-clear volume defined in Munoz et al. (2014). Such a volume, which depends on the relative position and velocity of two aircraft, is characterized next.

Let $x_r(t), y_r(t)$, and $z_r(t)$ be the relative position between the two aircraft in the x, y, and z coordinates and let $\dot{x}_r(t), \dot{y}_r(t)$, and $\dot{z}_r(t)$ be the relative velocity. If the positions of both aircraft are polynomial functions of time, then the relative position and velocity functions are also polynomials. The well clear-volume is centered at the location of a moving aircraft (i.e., an obstacle from the perspective of the aircraft with trajectory $r(t)$). For simplicity, the following presentation assumes that such a center is the origin. A simple coordinate transformation can be used to translate the well clear-volume from the origin to any other location in the airspace.

Multiple examples of the well-clear volume for different dynamic obstacles are shown in Figure 1. The well-clear volume is defined as the union of three time-varying sets. Hence, if any of these sets yields a conflict, then the well-clear volume has been violated. An exact representation of the clear volume developed in, Munoz et al. (2014), is obtained by using the semi-algebraic representation presented next. The first of the three sets is defined by the cylinder

$$\begin{aligned}(x_r(t)^2 + y_r(t)^2) - D^2 < 0,\\ z_r(t)^2 - Z^2 < 0,\end{aligned}$$

where $D > 0$ and $Z > 0$ are fixed parameters. The second set is the intersection of the following polynomial sets

$$\begin{aligned}x_r(t)\dot{x}_r(t) + y_r(t)\dot{y}_r(t) < 0,\\ (\dot{y}_r(t)^2 x_r(t) - \dot{y}_r(t)\dot{x}_r(t)y_r(t))^2 \ldots\\ +(\dot{x}_r(t)^2 y_r(t) - \dot{y}_r(t)\dot{x}_r(t)x_r(t))^2 \ldots\\ -(\dot{x}_r(t)^2 + \dot{y}_r(t)^2)D^2 < 0,\\ (x_r(t)^2 + y_r(t)^2) + \tau(x_r(t)\dot{x}_r(t) + y_r(t)\dot{y}_r(t)) - D^2 < 0,\\ z_r(t)^2 - Z^2 < 0,\end{aligned}$$

where τ and Z are parameters. The last set is defined by the following five polynomial inequalities:

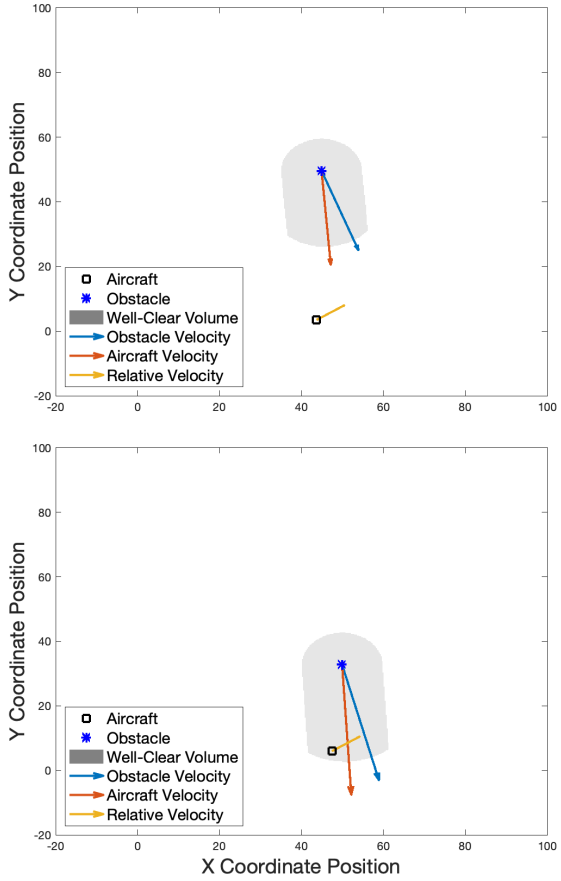

Fig. 1. The well-clear volume between an aircraft (square) and a moving obstacle (asterisk) is shaded in grey for two different conditions. The volume changes shape based on the relative velocity between the aircraft and the obstacle.

$$\begin{aligned}x_r(t)\dot{x}_r(t) + y_r(t)\dot{y}_r(t) < 0,\\ (\dot{y}_r(t)^2 x_r(t) - \dot{y}_r(t)\dot{x}_r(t)y_r(t))^2 \ldots\\ +(\dot{x}_r(t)^2 y_r(t) - \dot{y}_r(t)\dot{x}_r(t)x_r(t))^2 \ldots\\ -(\dot{x}_r(t)^2 + \dot{y}_r(t)^2)D^2 < 0,\\ (x_r(t)^2 + y_r(t)^2) + \tau(x_r(t)\dot{x}_r(t) + y_r(t)\dot{y}_r(t)) - D^2 < 0,\\ z_r(t)\dot{z}_r(t) < 0,\\ -(T\dot{z}_r(t)^2 + z_r(t)\dot{z}_r(t) < 0,\end{aligned}$$

where T is another constant defined a priori.

If d_x, d_y and d_z are the degrees of the relative position in the corresponding coordinates, the degree of the polynomial inequalities above is bounded by $\max(4(d_y - 1)d_x, 4(d_x - 1)d_y, 2d_z)$. Hence, if the polynomial paths are cubic, a commonly made assumption, this ensures that the degree of the univariate polynomial inequalities is less than or equal to 24. The next section presents means to efficiently and accurately calculate the roots of polynomials needed to estimate t^* in Formula (5).

4. CONFLICT DETECTION

This section addresses the problem of finding the first time instance in $[t_0, t_f]$ (if any) when $r(t)$ intersects $\cup_{i=1}^{n} O_i(t)$, given the polynomial path $r(t)$ in (1) and the obstacles in (2). To do so, the formula $p_{i,j}(x(t), y(t), z(t), t) \leq 0$ is

rewritten as a univariate polynomial with respect to time. If any of the following sets,

$$\begin{aligned} T_i := \{t : t_0 - t \leq 0,& \\ t - t_f \leq 0,& \\ p_{i,j}(x(t), y(t), z(t), t) \leq 0,\ \forall j = 1, ..., m_i\},& \end{aligned}$$

are non-empty then there exists a time $t \in [t_0, t_f]$ such that $r(t) \in \cup_{i=1}^{n} O_i(t)$, and a violation occurs.

It is clear that a violation will only occur if the set T_i is non-empty. The following theorem states that, over a finite time interval $t \in [t_0, t_f]$, if T_i is non-empty (a conflict occurs) then t^* will occur at either t_0 or at a root of $p_{i,j}(x(t), y(t), z(t), t)$ falling within $[t_0, t_f]$.

Theorem 1. *Let $p_j(t),\ \forall j = 1, \dots m$ be polynomials of degree at most d. If the semi-algebraic set*

$$T := \{t :\ t_0 - t \leq 0,\ t - t_f \leq 0,\ \ p_j(t) \leq 0, \forall j = 1, \dots m\}$$

is non-empty then the first instance of conflict $t^* = \min T$ corresponds to a root of the polynomial, $p_j(t^*) = 0$, for some $j \in [1, ..., m]$ or to $t^* = t_0$.

Proof. Define

$$t^* = \inf T.$$

Since t^* is the infimum of the set T, there is a sequence of points $\{t_k\}$ in T such that $t_k \to t^*$. Since $\{t_k\}$ is in T, $p_j(t_k) \leq 0, \forall j = 1, \dots m$, for each k. The fact that each p_j is a continuous function implies that $p_j(t^*) \leq 0, \forall j = 1, \dots m$. This means that $t^* \in T$ and therefore $t^* = \min T$.

If $t^* = t_0$ the result is shown. Suppose $t^* \neq t_0$ and that $p_j(t^*) < 0, \forall j = 1, \dots m$. Since each p_j is continuous and $t^* \neq t_0$, there exists an $\epsilon > 0$ such that for all $t \in (t^* - \epsilon, t^* + \epsilon)$, $p_j(t) < 0, \forall j = 1, \dots m$ and $t_0 < t$. This is a contradiction since, $t^* - \frac{\epsilon}{2} \in T$ and $t^* - \frac{\epsilon}{2} < t^* = \min T$. Therefore, there must exist a $j = 1, \dots m$ such that $p_j(t^*) = 0$. □

The first time at which a violation occurs must be at a root of one of the polynomials or at $t^* = t_0$. Therefore, to determine if a violation occurs, it suffices to check for violations at t_0 and at the roots of each polynomial function, $p_{i,j}(x(t), y(t), z(t), t)$.

In practice, the algorithm can be sped up by minimizing the number of times the polynomial functions $p_{i,j}(x(t), y(t), z(t), t)$ are evaluated. Because the algorithm only checks whether or not all of the polynomial functions, $p_{i,j}(x(t), y(t), z(t), t)$, are non-positive at some time instance, it is not necessary to know the actual value of the polynomial at any given time, but just if it is positive, negative, or zero. In addition, since the sign of a polynomial will only change at the roots of the polynomial, only sign information around the roots of a set of polynomials is needed to detect if they are ever negative at a point.

More precisely, given a set of n univariate non-zero polynomials p_i, with $1 \leq i \leq n$, and a sequence $t_0 < \ldots < t_l$ containing roots of all the polynomials p_i, define the sequence $v(t_0), \ldots, v(t_l)$ of n-dimensional vectors with entries 1, -1, or .5 as follows:

- If t_0 is not a root of p_i, the i-th component of $v(t_0)$ is the sign (1 or -1) of $p_i(t_0)$. If t_0 is a root of p_i, the i-th component of $v(t_0)$ is -1 if p_i is negative exactly after t_0, and .5 if p_i is positive exactly after t_0.
- For $1 \leq k \leq l$,
  1. If t_k is a root with odd multiplicity of p_i and the i-th component of $v(t_{k-1})$ is 1, then $v(t_k) = -1$
  2. If t_k is a root with odd multiplicity of p_i and the i-th component of $v(t_{k-1})$ is -1, then $v(t_k) = .5$
  3. If t_k is a root with even multiplicity of p_i and the i-th component of $v(t_{k-1})$ is 1, then $v(t_k) = .5$
  4. If t_k is not a root of p_i then the i-th component of $v(t_k)$ is the sign of the i-th component of $v(t_{k-1})$.

For $v(t_k)$ defined above, a violation happens when all entries of $v(t_k)$ are less than 1. A violation that lasts for more than a single instance, happens when the entries of $v(t_k)$ are all less than 0.

For instance, let the functions $p_i(x(t), y(t), z(t), t)$, for $i = 1, 2$, have the following values at $t = t_0 = 0$.

$$p_1(t_0) = -5 \text{ and } p_2(t_0) = 1.$$

Therefore, the vector $v(t_0)$ is defined as $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. Furthermore, assume the function $p_1(t)$ has real roots with odd multiplicity at $t = 1$ and $t = 5$, while $p_2(t)$ has a real root with even multiplicity at $t = 1$ and odd multiplicity at $t = 2$.

Since $v(t_0)$ has a positive component at t_0, no violation occurs at that time. Furthermore the next time at which a violation could occur is at the next root of one of the polynomial inequalities because that is the first time at which the sign of the polynomials, and thus $v(t)$, will change. In the given example,

$$v(1) = \begin{bmatrix} .5 \\ .5 \end{bmatrix},\ v(2) = \begin{bmatrix} 1 \\ -1 \end{bmatrix},\ \text{ and } \ v(5) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

The first root at which every element of v is less than 1 is $t = 1$, so this is where the first violation occurs. This violation happens for an instant, then is immediately resolved. At $t = 5$ all entries of $v(5)$ are less than 0, which indicates a violation that lasts for more than an instant.

This process is not defined for zero polynomials since they do not have a finite number of roots. For conflict detection, zero polynomials p_i may be discarded since the inequality $p_i(t) \leq 0$ is always satisfied, so a violation is contingent only on the rest of the polynomial constraints in the set T

As soon as a root is found such that every entry of v is less than 1, it can be determined that a violation will occur at that time. The entries of v encode the necessary sign information to determine if a violation occurs and whether the violation occurs for a single moment in time or for a longer period of time.

Algorithm 1 provides pseudocode for PolySafe. The algorithm must be run for each aircraft-obstacle combination, and many of the operations can be performed independently making it suitable for parallel programming.

Key to the implementation of Algorithm 1 is the ability to efficiently find roots of univariate polynomials. The roots of univariate polynomials of degree five or less can be calculated analytically. However, in cases such as the well-clear zone for cubic polynomial paths, the resulting polynomials have degree 24. As such, numerical methods to find the roots of high-order polynomials have to be used. Fortunately, there are a number of efficient and accurate numerical methods for determining the roots of

Algorithm 1 PolySafe Algorithm for Conflict Detection.

Input: $r(t)$, $O := \{(x,y,z) : p_j(x,y,z,t) \leq 0, \forall j = 1,...,m\}$, t_0, and t_f
Output: t^* (time at violation)
set $T := \{p_i(x(t),y(t),z(t),t) : \forall i = 1,...,n\}$,
discard zero polynomials, update T
for all polynomials in T **do**
  calculate roots of the i'th polynomial
  discard imaginary roots
  discard roots not within $[t_0, t_f]$
end for
sort roots in ascending order
calculate $v(t_0)$
for each root **do**
  update v at current root
  if all entries of v are less than 1 **then**
    return current root
  end if
end for
return ∞

univariate polynomials. For instance, there is the Jenkins-Traub method described in Ford (1977) and other matrix-based methods. Such methods are widely used in many engineering and science applications.

An approach to calculate the roots using the matrix-based approach is presented below. This approach, which requires finding the eigenvalues of the companion matrix of the polynomial, is suitable for real-time applications. Given the degree n polynomial $p(t) = c_0 + c_1 t + c_2 t^2 + ... + c_q^q$, where $c_q \neq 0$, the corresponding companion matrix $M \in \mathbb{R}^{(q+1)\times(q+1)}$ is given by

$$M \triangleq \begin{bmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_q \end{bmatrix}.$$

The determinant of $M - tI$ is the polynomial function $p(t)$, and thus the eigenvalues of M are the roots of the polynomial as shown in Edelman and Murakami (1995). The eigenvalues of mid-size matrices can be estimated accurately and efficiently using well-established numerical methods. For the purposes of this paper, a matrix larger than 25 by 25 is rarely required. Furthermore, for obstacles defined by the intersection of half planes and a cubic polynomial path, the size of the companion matrix is only 4 by 4.

5. FORMAL VERIFICATION OF POLYSAFE

A formal analysis of PolySafe was carried out in the Prototype Verification System (PVS). PVS, first introduced by Owre et al. (1992), is a formal verification tool that tightly couples a specification language with an interactive theorem prover. The specification language is a mathematical language for writing algorithms and specifying their properties in the form of lemmas and theorems. These lemmas and theorems are to be discharged by the PVS user issuing proof commands that corresponds to deductive rules of a higher-order logic. This approach is known as *formal verification* and stands in contrast to standard methods for validating algorithms that are purely based on testing and code review. Formal verification provides a high level of assurance that an algorithm is logically correct with respect to a well-defined set of assumptions. In the case of PolySafe, the formal analysis shows that under real arithmetic, assuming all roots of the polynomial constraints are precisely known, the PolySafe algorithm is guaranteed to detect the earliest violation. The following theorem states the correctness of the PolySafe algorithm and was formally verified in PVS.

Theorem 2. *For a polynomial path $r(t)$ defined in Formula (1), an obstacle*

$$O = \{(x,y,z) : p_j(x,y,z,t) \leq 0, \forall j = 1,...,m\},$$

where each $p_j(x,y,z,t)$ is a polynomial function, and $t_0, t_f \in \mathbb{R}$ such that $t_f \geq t_0$, the PolySafe algorithm in Algorithm 1 returns $t^* \in \mathbb{R}$ if and only if the earliest violation between $r(t)$ and O in $[t_0, t_f]$ occurs at time t^*.

Formal verification of PolySafe helped improve earlier versions of the algorithm. In the original algorithm, only the signs of the polynomials before and after each root were calculated, instead of the slightly more complicated procedure described in Section 4. This method could detect violations that lasted for any positive length of time, but not violations that occurred at a single instance. Whether this sort of violation is considered an actual threat is problem specific.

For example, when an obstacle is a separation volume around an aircraft, a point violation might not be an issue, as the volume is a buffer around the aircraft. On the other hand, for an exact geofence that has no buffer, a point violation would be a collision with that obstacle. The PolySafe algorithm can be changed to detect only violations that occur for more than an instance, by considering violations where each entry of $v(t_k)$ is less than 0 rather than 1.

Another part of the PolySafe algorithm that was developed during the formal verification process was the check that discarded the polynomials that were exactly zero, i.e. p_i= 0. There are many non-zero polynomials $p(x,y,z,t)$ and non-zero paths $r(t)$ in (1) where $p(x(t),y(t),z(t),t) = 0$ for all t. For example, choosing

$$r(t) = \left[\frac{1}{2}t,\ t-1,\ 2t+2\right]^\top, \tag{7}$$

$$p(x,y,z,t) = -32x + -4y^2 + z^2, \tag{8}$$

results in $p(x(t),y(t),z(t),t) = 0$ for all t, although this is not obvious by looking at $r(t)$ and $p(x,y,z,t)$.

A full PVS specification of PolySafe will be available as part of the NASA PVS Library. Future formal verification of PolySafe will account for violation detection using floating point arithmetic instead of real arithmetic, and when only approximations of the roots of the polynomial constraints are known.

6. SIMULATION RESULTS

The efficiency of the algorithm will impact the type and number of obstacles that can be modeled, the speed at which conflicts can be detected, and the speed at which conflicts can be later resolved.

Table 1 shows the average time taken to verify the safety of a trajectory of a polynomial path of degree 3 for a single obstacle. Obstacles with different numbers of inequalities and inequalities of different degrees are compared. These calculations were performed on a standard desktop computer with 2.8GHz quad-core Intel Core i5 processor and 8 GB of 1867MHz LPDDR3 onboard memory. The times given account for the time needed to replace the x, y, and z values in the inequalities with the $x(t)$, $y(t)$, and $z(t)$ functions of a trajectory, and for the execution of the PolySafe algorithm. The code was tested in Matlab with a randomly generated polynomial path, and an obstacle at the origin 1000 different times. To represent the worst-case scenario, a large time horizon is considered so every root of the polynomial is within the time interval, and must be checked. Therefore, the values in the table are conservative estimates of the time required by the algorithm on a realistic application.

The low computational cost allows for checking a large number of obstacles. For instance, given one tenth of a second to check the safety of a trajectory, over 1,300 spherical obstacles can be verified and over 350 cubic obstacles. For checking well-clear violations of moving aircraft, the algorithm could check an average of 120 aircraft every one tenth of a second.

It is also important to note that PolySafe can be trivially parallelized. The checking of the safety properties of a trajectory $r(t)$ relative to any given obstacle $O_i(t)$ is completely independent from all other obstacles. Therefore, the capability exists for obstacles to be split into n groups and verified on separate cores.

Obstacle	Max Degree	N	Time (ms)	STD (ms)
Pyramid	3	5	0.215	0.047
Cube	3	6	0.281	0.092
Sphere	6	1	0.073	0.012
Cylinder	6	3	0.159	0.029
Well-Clear	24	16	0.829	0.127

Table 1. Average time for PolySafe with a randomly selected degree 3 polynomial path and a single obstacle where N is the number of inequalities that define the obstacle. The average time and standard deviation (STD) for 1000 tests are reported.

7. CONCLUSION

This paper proposes a formally verified algorithm for checking safety criteria of a polynomial flight path on a dynamic airspace. By representing static and dynamic obstacles as semi-algebraic sets, not only can a wide range of obstacles be accurately modeled, but potential flight conflicts with such obstacles can be efficiently detected. PolySafe can be used to generate violation warnings in real time, as well as to provide key information needed to replan aircraft paths that violate acceptable levels of separation. Additionally, PolySafe has been formally verified to guarantee accurate reporting of these violation warnings.

PolySafe is shown to efficiently handle hundreds to thousands of obstacles in just a tenth of a second, thereby making it suitable for complex and crowded airspaces. Because each obstacle can be processed independently from the rest, PolySafe can be implemented in parallel thereby further increasing the speed of computation. Strategies for the integration of PolySafe to guidance algorithms that yield new conflict-free paths will be considered in the future.

REFERENCES

Balachandran, S., Narkawicz, A., Muñoz, C., and Consiglio, M. (2017). A path planning algorithm to enable well-clear low altitude UAS operation beyond visual line of sight. In *Proceedings of the 12th USA/Europe Air Traffic Management R&D Seminar, ATM 2017*, 16. Seattle, Washington.

Colbert, B.K., Crespo, L.G., and Peet, M.M. (2019). A sum of squares optimization approach to uncertainty quantification. In *2019 American Control Conference (ACC)*. IEEE.

Cook, S.P., Brooks, D., Cole, R., Hackenberg, D., and Raska, V. (2015). Defining well clear for unmanned aircraft systems. In *Proceedings of the 2015 AIAA Infotech @ Aerospace Conference*, AIAA-2015-0481. Kissimmee, Florida.

Dill, E., Young, S., and Hayhurst, K. (2016). Safeguard: An assured safety net technology for UAS. *2016 IEEE/AIAA 35th digital avionics systems conference (DASC)*.

Edelman, A. and Murakami, H. (1995). Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210).

Federal Aviation Administration (FAA) Sponsored Sense and Avoid Workshop (2009). Sense and avoid (SAA) for unmanned aircraft systems (UAS).

Ford, J. (1977). A generalization of the jenkins-traub method. *Mathematics of Computation*, 31(137).

Kochenderfer, M.J., Holland, J.E., and Chryssanthacopoulos, J.P. (2012). Next-generation airborne collision avoidance system. *Lincoln Laboratory Journal*, 19(1).

Munoz, C., Narkawicz, A., Chamberlain, J., Consiglio, M.C., and Upchurch, J.M. (2014). A family of well-clear boundary models for the integration of UAS in the NAS. In *14th AIAA Aviation Technology, Integration, and Operations Conference*.

Narkawicz, A., Muñoz, C., and Dutle, A. (2018). Sensor uncertainty mitigation and dynamic well clear volumes in DAIDALUS. In *Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018)*. London, England, UK.

Owre, S., Rushby, J.M., and Shankar, N. (1992). PVS: A prototype verification system. In *International Conference on Automated Deduction*, 748–752. Springer.

Stevens, M. and Atkins, E. (2016). Multi-mode guidance for an independent multicopter geofencing system. In *16th AIAA Aviation Technology, Integration, and Operations Conference*.

Williamson, T. and Spencer, N. (1989). Development and operation of the traffic alert and collision avoidance system (TCAS). *Proceedings of the IEEE*, 77(11).