Neural Network Verification Methods for Closed-Loop ACAS Xu Properties [*]

Diego Manzanas Lopez[†], and Taylor T. Johnson[‡]
Vanderbilt University, Nashville, TN, 37235

Stanley Bak[§]
Stony Brook University, Stony Brook, NY, 11794, USA

Hoang-Dung Tran[¶]
University of Nebraska-Lincoln, Lincoln, NE, 68588, USA

Kerianne L. Hobbs [‖]
Air Force Research Laboratory, Wright-Patterson AFB, OH, 45431

Neural network approximations have become attractive to compress data for automation and autonomy algorithms for use on storage-limited and processing-limited aerospace hardware. However, unless these neural network approximations can be exhaustively verified to be safe, they cannot be certified for use on aircraft. This manuscript evaluates the safety of a neural network approximation of the unmanned Airborne Collision Avoidance System (ACAS Xu). First, a set of ACAS Xu closed-loop benchmarks is introduced, based on a well-known open-loop benchmark, that are challenging to analyze for current verification tools due to the complexity and high-dimensional plant dynamics. Additionally, the system of switching and classification-based nature of the ACAS Xu neural network system adds another challenge to existing analysis methods. Experimental evaluation shows selected scenarios where the safety of the ownship aircraft's neural network action selection is assessed with respect to an intruder aircraft over time in a closed loop control evaluation. Set-based analysis of the closed-loop benchmarks is performed using the Star Set representation using both the NNV tool and the nnenum tool, demonstrating that set-based analysis is becoming increasingly feasible for the verification of this class of systems.

Nomenclature

ρ	=	distance between ownship and intruder, ft

[*]Approved for Public Release. Case Number 88ABW-2020-3203.
[†]Graduate Student, Electrical Engineering and Computer Science, 1025 16th Ave South, Suite 102
[‡]Assistant Professor, Electrical Engineering and Computer Science, 1025 16th Ave South, Suite 102
[§]Assistant Professor, Computer Science, 100 Nicolls Rd
[¶]Assistant Professor, Computer Science and Engineering,
[‖]Aerospace Engineer, Autonomy Capability Team (ACT3), 2210 8th Street, AIAA Member.

θ	=	angle to intruder w.r.t ownship heading, rad
ψ	=	heading of intruder w.r.t. ownship heading, rad
v	=	velocity, ft/s
τ	=	time until loss of vertical separation, s
a_{prev}	=	previous advisory
SL	=	strong left
WL	=	weak left
COC	=	clear of conflict
WR	=	weak right
SR	=	strong right
x	=	position of the aircraft in cartesian x-direction, ft
y	=	position of the aircraft in the cartesian y-direction, ft
$\dot{x}$	=	aircraft velocity in the x-direction, ft/s
$\dot{y}$	=	aircraft velocity in the y-direction, ft/s
$\dot{\psi}$	=	time rate of change in heading, rad/s
c	=	user defined constant control input to ownship, ft/s
φ	=	third angle planar relative in aircraft geometry, rad
T	=	time to collision, s

Subscripts

cg	=	center of gravity
int	=	intruder
own	=	ownship

I. Introduction

The use of machine learning in information technology domains has drastically improved automated capabilities like natural language processing [1], image classification [2], and object detection [3]. In the last several years machine learning has also tackled complex game playing with high dimensional state spaces, beating world experts in Go [4, 5] and StarCraft II [6]. In addition to potentially optimizing performance of complex systems, neural networks are often much smaller and faster to compute than other optimization algorithms, making them attractive for use on Cyber-Physical Systems (CPS) like robots, cars, drones, and satellites. For example, neural networks have been used to compress the 2 gigabytes of lookup tables used by the Airborne Collision Avoidance System X to 5 megabytes of neural network weights [7].

However, use of machine learning in safety-critical system domains has historically been limited by verification frameworks that treat neural networks like black boxes. Neural networks are not black boxes, however, they are compute deterministic mathematical functions that are increasingly amenable to formal reasoning methods [8, 9]. While many of these methods analyze the networks in isolation, several recent methods attempt to verify neural network control systems (NNCS) [10–15]. NNCS can be generated through a variety of approaches, including reinforcement learning (RL), model predictive control (MPC), and learning by demonstration. In the realm of NNCS safety verification, the problem is often postulated as a reachability analysis question. By analyzing the reachable sets of the plant and the controller the possible states of the plant are computed and evaluated for safety. Typically the controller is a feed-forward neural network (FNN) and the plant may be described using ordinary differential equations (ODEs) or hybrid automata (HA), in either discrete time or continuous time.

Although there has been significant progress in NNCS verification, developing scalable and non-conservative methods remains a key issue. Recently, studies by Ivanov et al. [16], Tran et al. [17] and D.M. Lopez et al. [18], have investigated the limits and capabilities of existing verification methods and tools [10, 19], and the trade-offs between scalability and conservativeness. Julian et al. [20] and Akintunde et al. [21] have formally verified two closed-loop variants of the simplified version of the vertical advisory control system in aircraft, the VerticalCAS [22], using reachability analysis and MILP methods respectively. Building on these studies, this manuscript seeks to examine a more complex NNCS model—the ACAS Xu NNCS benchmark formally described in section III—by using star-set verification methods [23] in both the NNV tool [19] and the nnenum tool [24]. The largest novelty and challenge of the ACAS Xu NN approximation is the switching nature of the NN control system; the system contains multiple neural networks and the neural network executed at each step depends on the previous step's output advisory. This NNCS is evaluated using set-based analysis methods with the goal of verifying the correct and safe behavior of an aircraft controlled with ACAS Xu in a set of initial states for each of the proposed closed-loop scenarios.

II. ACAS X

The Airborne Collision Avoidance System X (ACAS X) is under development to one day replace the Traffic Collision Advisory System II (TCAS II) as a mid-air collision prevention system [25]. ACAS X is designed to be compatible with the FAA's Next-Generation Air Transportation System (NextGen), which uses new sensing and navigation technologies coupled with new procedures to better optimize air traffic [25].

There are four variants of ACAS X [25], and this manuscript analyzes safety of a neural network approximation of the ACAS Xu variant:

- ACAS Xa (active): Designed to provide protection from all tracked aircraft using onboard sensors on large manned aircraft, ACAS Xa issues alerts and vertical advisories to the pilot [26].
- ACAS Xu (unmanned): Optimized for unmanned aircraft systems, ACAS Xu issues turn rate advisories to remote

pilots [27].

- ACAS Xo (operation): special alerts during operations such as parallel runway approaches [25].
- ACAS Xp (passive): tracks aircraft for potential collisions, but doesn't produce advisories [25].

Both ACAS Xa and ACAS Xu have a goal to avoid near midair collisions (NMAC) [28], defined as separation less than 100 ft vertically and 500 ft horizontally [29], as depicted in Fig. 1.

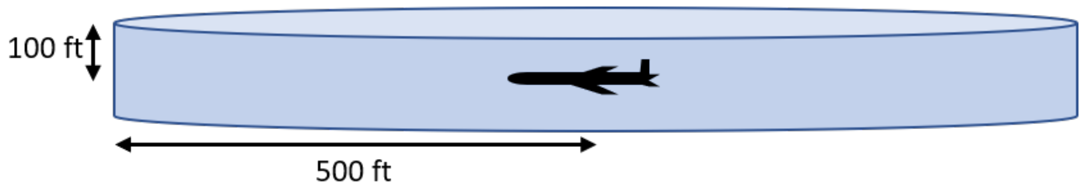


Fig. 1 Near midair collision cylinder, defined as separation less than 100 ft vertically and 500 ft horizontally.

ACAS X functionality centers on the use of a set of look up tables generated offline with dynamic programming and Markov decision processes (MDPs) [30]. The input to these lookup tables is a probabilistic state distribution of the relative position and velocity of nearby aircraft approximated by sensor data. The output of the lookup tables is a cost for each of three possible actions: no alert, traffic advisory, or a resolution advisory for pilots to maintain or increase safe separation from other aircraft [25].

A. ACAS Xu

ACAS Xu, the unmanned aircraft version, assigns values to a set of output actions based on a set of input variables as described in Table 1. The first five variables describe 2D considerations, the sixth variable brings the scenario into 3D, and the seventh variable promotes advisory selection consistency. In the initial lookup tables, the state variables are discretized in a seven dimensional grid with 120 million points [31] that assign a value to each of 5 different output action options, resulting in 600 million floating point numbers. When the state of the system falls between discrete points, the nearest neighbor point is used [31]. To deal with sensor uncertainty in the system state, ACAS Xu uses an unscented Kalman filter.

Table 1 Input state variables used in ACAS Xu.

Variable	Units	Description	Tables	NN
ρ	ft	distance between ownship and intruder	Y	Y
θ	rad	angle to intruder w.r.t ownship heading	Y	Y
ψ	rad	heading of intruder w.r.t. ownship heading	Y	Y
v_{own}	ft/s	velocity of ownship	Y	Y
v_{int}	ft/s	velocity of intruder	Y	Y
τ	s	time until loss of vertical separation	Y	N
a_{prev}	deg/s	previous advisory	Y	N

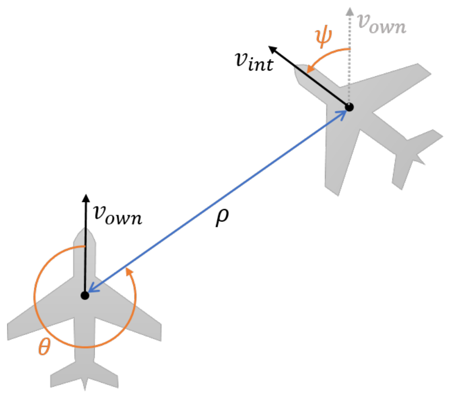


Fig. 2 Diagram of the ACAS Xu physical variables.

Table 2 ACAS Xu Actions (Horizontal Collision Avoidance).

Action	Description
SL	strong left at 3.0 deg/s
WL	weak left at 1.5 deg/s
COC	clear of conflict (do nothing)
WR	weak right turn at 1.5 deg/s
SR	strong right turn at 3.0 deg/s

B. Neural Network Compression of ACAS Xu

A major challenge to implementation of ACAS Xu is that the initial look up tables require hundreds of gigabytes of floating point storage [7]. Using a technique called downsampling, values may be removed from the table in areas where variation between the values is very small and has been shown to reduce ACAS Xu table size to approximately 2 gigabytes [7]. For ACAS Xa, which limits advisories to vertical maneuvers and has smaller lookup tables to begin with, downsampling was sufficient [32]. However, ACAS Xu's 2 gigabyte size may still be too large for the storage constraints of certified avionics hardware [33].

Developed in [7] and evaluated in [27], 45 separate neural networks were used to compress the lookup table. Each network is denoted $N_{x,y}$, where x corresponds to the index (1 to 5) of a specific value of previous advisory $a_{prev} \in \{COC, WL, WR, SL, SR\}$ and y corresponds to the index (1 to 9) of a specific value of time to loss of vertical separation $\tau \in \{0, 1, 5, 10, 20, 40, 60, 80, 100\}$ seconds. For example, $N_{2,3}$ corresponds to a neural network in which $a_{prev} = WL$ and $\tau = 5$. Then each of these networks receives inputs for the remaining five state variables, and outputs a value associated with each of the five output variables ($\{COC, WL, WR, SL, SR\}$). Several architectures and optimizers were considered and analyzed for the training of all the 45 neural networks. AdaMax, a variant of Adam, was chosen as it proved to learn the fastest without getting stuck in local optima. As for the layer architecture, six hidden layers of 50 neurons each proved to yield the best results while maintaining an efficient computation time [7]. Hence, all the neural networks have five inputs, five outputs and six hidden layers of 50 neurons each, as depicted in Fig. 3.

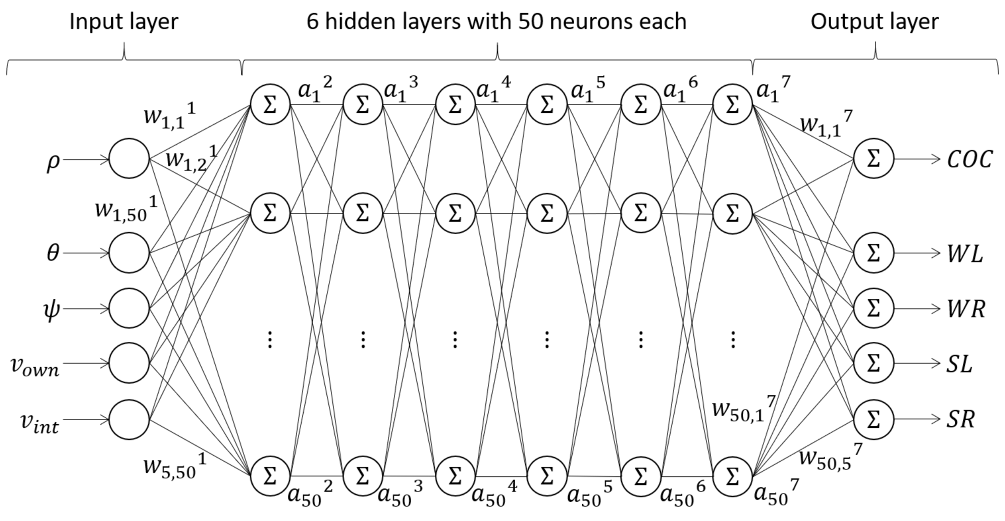


Fig. 3 Depiction of the ACAS Xu neural network.

C. Open-Loop vs. Closed-Loop Verification

Previous research has verified the neural networks using a set of open-loop test points [22, 27, 31, 33]. In other words, for a single instant in time, the neural network is verified to output appropriate scores for a particular set of inputs. The open-loop properties described in [27] became a popular benchmark for open loop verification, and are summarized here in the Appendix of this paper. However, a much more important property that needs to be verified for these neural networks is that, when used in *closed-loop* control, the neural network approximation of the lookup tables will not result in control actions that violate the NMAC safety constraint depicted earlier in Fig. 1. One of the contributions of this manuscript is the description of a set of closed-loop verification scenarios.

III. Model

Considering the intruder and ownship as Dubins aircraft models, a nonlinear relationship is obtained between the aircraft states and the NN inputs. The index of the output with the minimum value is computed and the corresponding advisory command is sent to the ownship aircraft plant. These relationships are not trivial to encode in most reachability analysis tools, so transformations were necessary.

This paper introduces and uses a closed-loop ACAS Xu benchmark, where state variables 1 to 3 correspond to the ownship states, variables 4 to 6 correspond to the intruder state. Two approaches are analyzed, one using the NNV tool and a second using the nnenum tool.

For NNV, additional variables 7 to 9 are introduced represent the nonlinear relationship between aircraft states that correspond to inputs 1 to 3 of the NNs. The original functions for these 3 inputs are described in Eq. 1. Defining these as differentiable functions, their time derivatives are computed and added as state variables in the system resulting in a 9-dimensional ODE plant model in Eq. 2, where c is a user-defined constant.

$$
\begin{aligned}
\rho &= \sqrt{(x_i - x_o)^2 + (y_i - y_o)^2} \\
\theta &= 2tan^{-1}(\frac{y_i - y_o}{x_i - x_o + \rho}) - \psi_o \\
\psi &= \psi_i - \psi_o
\end{aligned}
\tag{1}
$$

In the second nnenum approach, only the first six variables and ODEs are used from Eq. 2, with an external observation function that computes the nonlinear variables needed for the neural network inputs as described in Eq. 1.

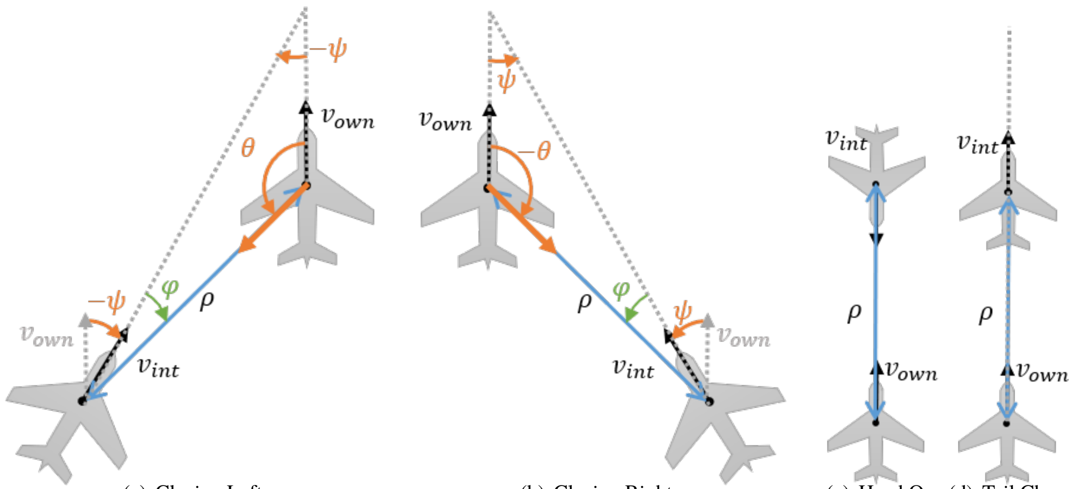
$$
\begin{aligned}
\dot{x}_1 &= \dot{x}_o = vcos(\psi_o), \\
\dot{x}_2 &= \dot{y}_o = vsin(\psi_o), \\
\dot{x}_3 &= \dot{\psi}_o = u, \\
\dot{x}_4 &= \dot{x}_i = vcos(\psi_i), \\
\dot{x}_5 &= \dot{y}_i = vsin(\psi_i), \\
\dot{x}_6 &= \dot{\psi}_i = c, \\
\dot{x}_7 &= \dot{\rho} = \frac{(y_i - y_o)(\dot{y}_i - \dot{y}_o) + (x_i - x_o)(\dot{x}_i - \dot{x}_o)}{\sqrt{(x_i - x_o)^2 + (y_i - y_o)^2}}, \\
\dot{x}_8 &= \dot{\theta} = \frac{2(\dot{y}_i - \dot{y}_o)(x_i - x_o + \rho) - 2(y_i - y_o)(\dot{x}_i - \dot{x}_o + \dot{\rho})}{(y_i - y_o)^2(x_o - x_i + \rho)^2} - \dot{\psi}_o, \\
\dot{x}_9 &= \dot{\psi} = \dot{\psi}_i - \dot{\psi}_o
\end{aligned}
\tag{2}
$$

For input into the NN, it is required that that θ and ψ must be in the $[-\pi, \pi]$ range. Thus, these dynamics can be defined as hybrid automata with one mode of operation, where preset self-transitions to keep these states within range, or can be transformed into their correct range after computing the plant outputs, separate from the plant dynamics computation.

IV. Closed Loop Verification Test Case Generation

The objective of the closed loop verification tests is to demonstrate that the ownship and intruder aircraft remain safely separated at all times. The unsafe set is defined by the NMAC cylinder when there is a separation of less than 100 ft vertically or 500 ft horizontally. To scope the testing, the following ranges of variables are used: distances from 0 to the maximum turn diameter ($\rho \in [0, 87472]$ ft), the full range of angles to the intruder and heading of the intruder with respect to the ownship ($\theta \in [-pi, pi]$,$\psi \in [-pi, pi]$), and full range of reasonable velocities from a slow takeoff velocity to over Mach 1 ($v_{int} \in [60, 1145]$ ft/s, $v_{own} \in [100, 1145]$ ft/s). By law of sines, the variables used to describe the relative aircraft geometry are related with the time to collision T as described in Eq. 3, and depicted in Fig. 4. Any collision test case can be generated by fixing the time to collision T, θ, v_{int}, and ρ, and computing the remaining variables as described in Table 3.

$$\frac{\rho}{\sin\psi} = \frac{v_{own}T}{\sin(\theta - \pi - \psi)} = \frac{v_{int}T}{sin(2\pi - \theta)} \tag{3}$$



(a) Closing Left (b) Closing Right (c) Head On (d) Tail Chase

Fig. 4 Closed Loop Test Case Geometry Examples

Table 3 Computing Ownship Velocity and relative heading from randomly selected angle to intruder, intruder velocity, and distance to the intruder

	Closing Left	Closing Right	Head On	Tail Chase
θ	$0 < \theta < \pi$	$-\pi < \theta < 0$	$\theta = \pm\pi$	$\theta = 0$
ψ	$-\sin^{-1}\left(\frac{\rho}{T\, v_{int}} sin(\lvert\theta\rvert)\right)$	$\sin^{-1}\left(\frac{\rho}{T\, v_{int}} sin(\lvert\theta\rvert)\right)$	0	0 or π
φ (from sum of angles)	$\pi - \lvert\theta\rvert - \lvert\psi\rvert$	$\pi - \lvert\theta\rvert - \lvert\psi\rvert$	-	-
v_{own}	$\frac{v_{int}\sin(\varphi)}{sin(\lvert\theta\rvert)}$	$\frac{v_{int}\sin(\varphi)}{sin(\lvert\theta\rvert)}$	$\frac{T\, v_{int} - \rho}{T}$	$\frac{T\, v_{int} + \rho}{T}$

To generate a standard set of closed loop test cases, θ, v_{int}, and ρ were selected from the following discretized sets of values and used to calculate appropriate values of v_{own} and ψ: $\theta \in \{-3\pi/4, -pi/2, -3\pi/8pi/4, 0, \pi/4, \pi/3, 3\pi/4, \pi\}$ rad, $v_{int} \in \{60, 150, 300, 450, 600, 750, 900, 1050, 1145\}$ ft/s, and $\rho \in \{10000, 43736, 87472, 120000\}$ ft. Using a Latin hypercube sampling of $v_i nt$ and θ for baseline coverage, the test cases in Table 4 may be used to evaluate a range of properties across the state space. To compute the Cartesian coordinates of the intruder, Eqs. 4-5. With the exception of the head on test case 10, each of the test cases are selected so that both aircraft are traveling in roughly the same direction. This is because aircraft are generally separated by at least 1000 feet in altitude Air Traffic Control or federal regulations. For example, the Code of Federal Regulations, Title 14, Section 91.159 instructs pilots to fly at an odd altitude plus 500 feet when traveling East on a heading between 0-179 degrees, or an even altitude plus 500 feet when traveling West on a heading between 180-359 degrees [34].

$$x_{int} = -\rho \sin(\theta) \tag{4}$$

$$y_{int} = \rho \sin(\pi/2 - \theta) \tag{5}$$

Table 4 ACAS Xu Benchmark Closed Loop Test Cases

Test Point	Name	v_{int} (ft/s)	θ (rad)	ρ (ft)	v_{own} (ft/s)	ψ (rad)	x_{int} (ft)	y_{int} (ft)
$\varphi_{1_{CL}}$	Left Abeam	1050	$\pi/2$	43,736	954.6	-0.4296	-43,736	0
$\varphi_{2_{CL}}$	Intruder Tail Chase	900	π	43,736	462.6	0	0	-43,736
$\varphi_{3_{CL}}$	Right Gaining	200	$-3\pi/4$	43,736	100	0.3617	30,926	-30,926
$\varphi_{4_{CL}}$	Left Gaining	600	$3\pi/4$	43,736	204.9	-0.5415	-30,926	-30,926
$\varphi_{5_{CL}}$	Left Closing	300	$\pi/4$	43,736	362.3	-0.2379	-30933	30933
$\varphi_{6_{CL}}$	Right Abeam	750	$-\pi/2$	43,736	609.3	0.6226	43,736	0
$\varphi_{7_{CL}}$	Right Isosceles	1145	$-3\pi/8$	87,472	1145.9	0.7835	80,814	33,474
$\varphi_{8_{CL}}$	Right Closing	450	$-\pi/4$	43,736	636.2	0.7577	30,926	30,926
$\varphi_{9_{CL}}$	Ownship Tail Chase	60	0	43,736	497.4	0	0	43,736
$\varphi_{10_{CL}}$	Head On Collision	600	0	120,000	600	π	0	120,000

V. Analysis Approaches

This section describes the fundamental theory behind star sets used to approximate reachable sets, as well as the closed loop neural network verification approaches.

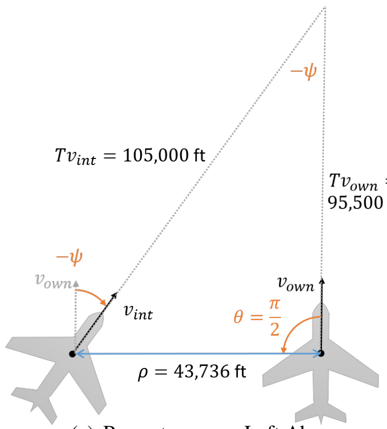
A. Star Set Theory

A star set [23, 35] (or simply star) Θ is a tuple $\langle c, V, P\rangle$ where $c \in \mathbb{R}^n$ is the center vector, $V = \{v_1, v_2, \cdots, v_m\}$ is a set of m basis vectors in $\mathbb{R}^n$, and $P : \mathbb{R}^m \to \{\top, \bot\}$ is a predicate. The set of states represented by the star is given as:

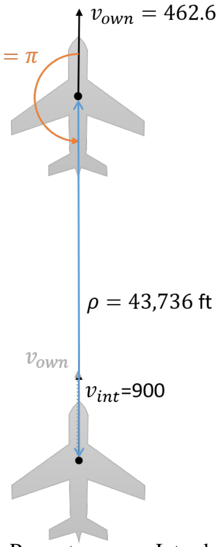
$$[\![\Theta]\!] = \{x \mid x = c + \Sigma_{i=1}^{m}(\alpha_i v_i) \text{ such that } P(\alpha_1, \cdots, \alpha_m) = \top\}. \tag{6}$$

Sometimes we will refer to both the tuple Θ and the set of states $[\![\Theta]\!]$ as Θ. In this work, we restrict the predicates to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ where, for p linear constraints, $C \in \mathbb{R}^{p\times m}$, α is the vector of m-variables, i.e., $\alpha = [\alpha_1, \cdots, \alpha_m]^T$, and $d \in \mathbb{R}^{p\times 1}$. A star is an empty set if and only if $P(\alpha)$ is empty.

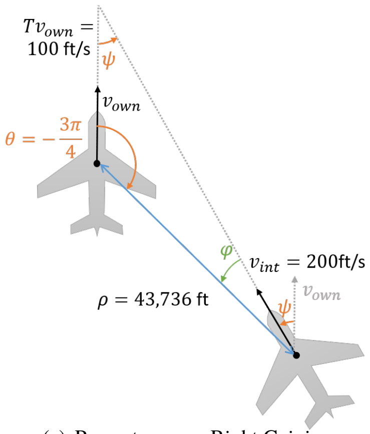
Any convex polyhedra can be represented as star sets. An affine mapping of a star set $\Theta = \langle c, V, P\rangle$ defined by



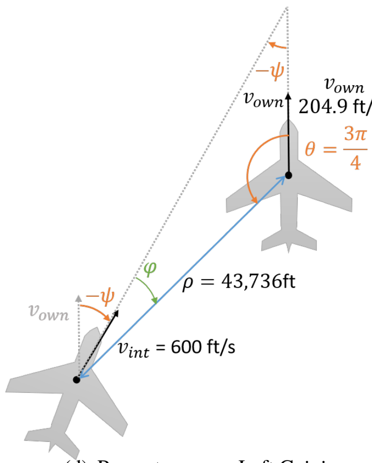
(a) Property $\varphi_{1_{CL}}$, Left Abeam



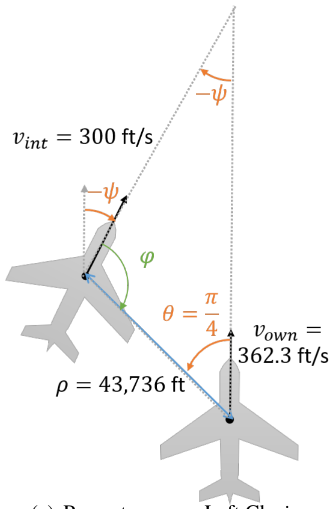
(b) Property $\varphi_{2_{CL}}$, Intruder Tail Chase



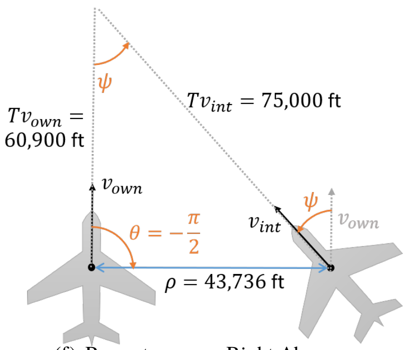
(c) Property $\varphi_{3_{CL}}$,Right Gaining



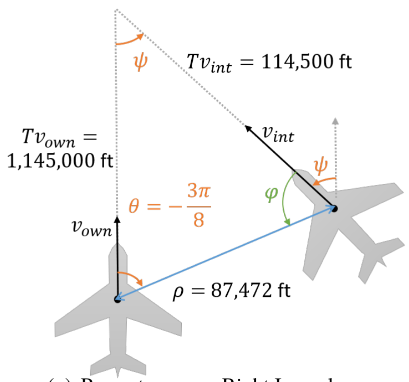
(d) Property $\varphi_{4_{CL}}$, Left Gaining



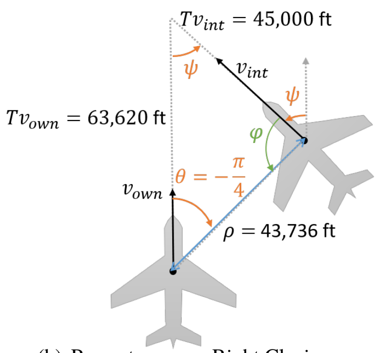
(e) Property $\varphi_{5_{CL}}$, Left Closing



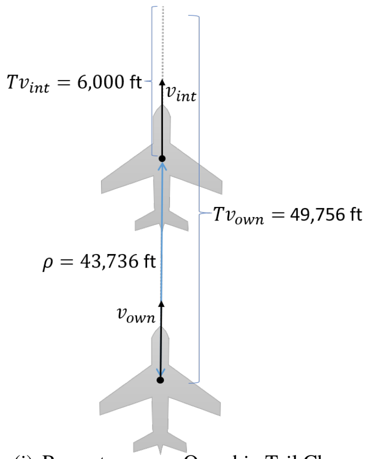
(f) Property $\varphi_{6_{CL}}$, Right Abeam



(g) Property $\varphi_{7_{CL}}$, Right Isosceles



(h) Property $\varphi_{8_{CL}}$, Right Closing



(i) Property $\varphi_{9_{CL}}$, Ownship Tail Chase

Fig. 5 Depiction of the aircraft encounter geometry for the closed loop ACAS Xu verification properties

a mapping matrix W and an offset vector b is another star set $\Theta' = \langle Wc + b, WV, P\rangle$. The intersection of a star set $\Theta = \langle c, V, P\rangle$ with a polyhedron $G \triangleq H\alpha \leq l$ is another star set $\Theta' = \langle c, V, P \wedge P_G\rangle$, where $P_G \triangleq H \times V\alpha \leq l - H \times c$.

B. Neural Network Verification (NNV) Tool

The NNV tool is one of the two verification tools used to analyze the safety and functionality of the ACAS Xu NN system. This tool is evaluated on 10 closed-loop benchmarks where the safety of the ownship is analyzed with respect to one intruder aircraft. The initial analysis consists of evaluating the safety of the ownship aircraft using algorithm 1 under multiple scenarios, where the plant is described as the set of non-linear ODEs in Eq. 2. All the sets are represented using star-sets, and all the reachability functions return over-approximations of the reachable sets. As part of the functions used in the reachability algorithm, multiple trajectories are allowed to be computed at each time step. In the case that multiple indexes are returned, the reach sets are split and paired with the corresponding advisory, keeping a record of the parents of each set to avoid an explosion of computed sets and reduce the conservativeness and memory consumption in NNV methods. This is a new feature added to the existing NNV star-set methods.

Algorithm 1: General ACAS Xu NNCS reachability algorithm in NNV

```
Result:
$Rs$: Plant reachable states
Initialize
$init\_set$: Initial state set;
$a_{prev}$: Previous advisory;
$tF$: Number of control steps;
$safe$: True;
$out\_mat$: Plant output matrix;
while safe do
    for ($i = 0$; $i < tF$; $i = i + 1$) do
        state_set = plantReach(init_set , a$_{prev}$);
        if $\rho < 500$ then
            safe = False
        end
        output_set = affineMap(state_set, out_mat);
        input$_{NN}$ = normalizeNNinputs(output_set);
        output$_{NN}$ = NNreach(a$_{prev}$, input$_{NN}$);
        advisory$_{NN}$ = argminNN(output$_{NN}$);
        init_set = state_set;
        a$_{prev}$ = advisory$_{NN}$;
        Rs[i] = state_set;
    end
end
```

C. Neural Network Enumeration (nnenum) Tool

The nnenum [24] tool is also used for analysis, which uses star sets to reason over possible neural network outputs given sets of input states. For nnenum's analysis, sets are split whenever multiple advisories are possible, as well as when needed to maintain the input ranges for the neural networks (for example, if an angle can be less than or greater than π). The plant reachability is done using a local numerical linearization of the observed state variables.

VI. Results

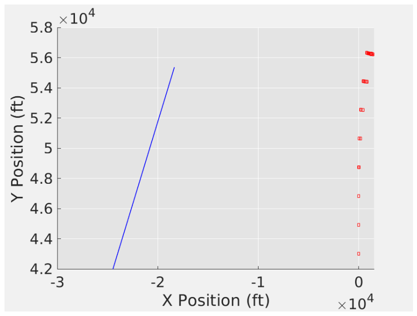
The correct and safe behavior of the ACAS Xu NN approximation is evaluated with two different tools NNV and nnenum on the 10 closed loop test cases depicted on Fig. 5. For all the following scenarios, a control period of 2 seconds is used.

A. Results with the NNV Tool

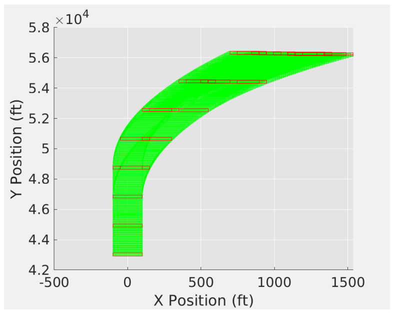
Results for all 10 benchmarks are introduced in the order presented in Table 4, with a reach step size of 0.05 seconds. For each case, it is verified that the ownship's advisory system guides the aircraft to safety in the presence of the intruder. Based on the simulations of each experiment, the computation process is lightened by fast-forwarding the initial state to a future point in the trajectory while still maintaining the COC advisory for at least the following 2 control steps. At this point, the uncertainty to the x-y position of the ownship is included.

All these results can be found in Figs. 6 to 15. For each scenario, three different plots are presented to illustrate our results. Plot *a* corresponds to the control step reach sets of ownship and intruder trajectory, while subfigure *b* corresponds to the ownship reachable sets: the control step reach sets in red and all the interval reach sets along the ownship's trajectory in green. Subfigure *c* corresponds to a rotation to the original test case, to demonstrate that initial orientation of aircraft is not relevant to the ACAS Xu NN advisory system. However, NNV methods may be affected by them under certain specific conditions, such as the intruder tail chase $\varphi_{2_{CL}}$. Due to the nature of the ODE model and the over-approximation methods, when uncertainty is added both to the x and y positions, the plant reach sets split infinitely many times and thus causing out of memory errors. Therefore, uncertainty is added to only the x position as it provides a set of more relevant and meaningful results.

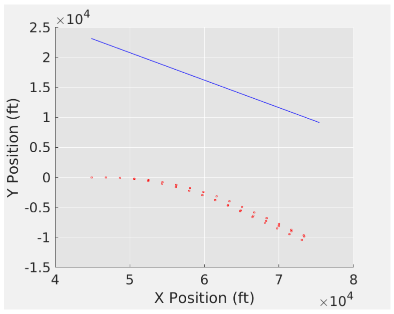
Depicted in the ten experiments, when the ownship is approached by the intruder from one side, the ownship deviates its trajectory towards the opposite side. This can be observed in Figs. 6, 8-13. In the remaining cases, the ownship navigates away to its right in order to avoid a head-on collision in Fig. 15 and to avoid crashing into the intruder's tail, as depicted in Fig. 14. Lastly, when the ownship is the one being tail chased by the intruder, depending on the exact location of the aircraft along the x-axis, the advisory system may send strong left or weak right turn commands, as observed in Fig. 7.



(a) Ownship (red) vs. intruder (blue)

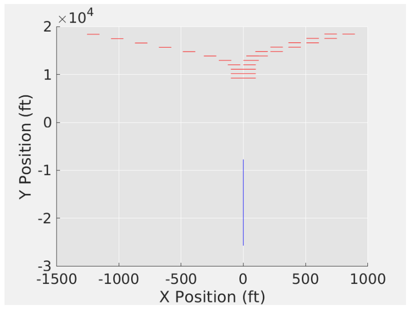


(b) Ownship step sets (red) and all sets (green). (Zoomed in)

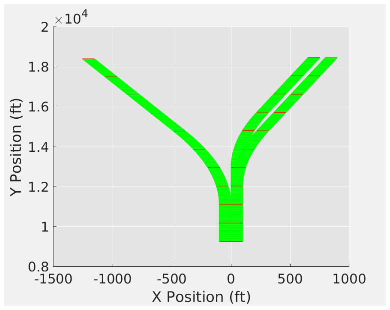


(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

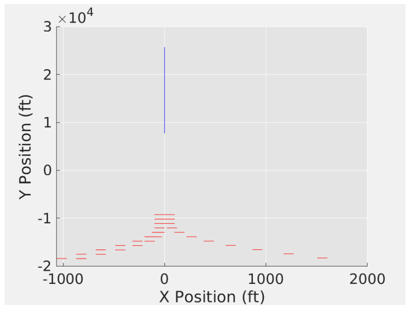
Fig. 6 ***Reachable Sets:*** **Left Abeam $\varphi_{1_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

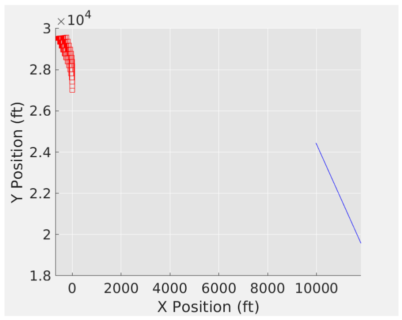


(b) Ownship step sets (red) and all sets (green). (Zoomed in)

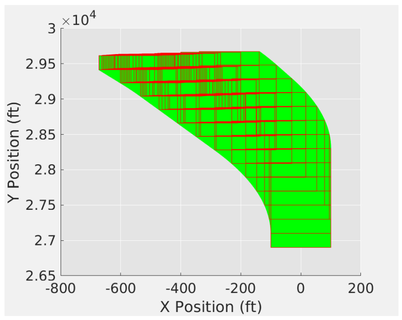


(c) Ownship (red) vs. intruder (blue). (Rotation 180°)

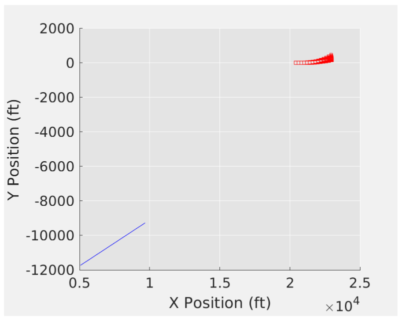
Fig. 7 ***Reachable Sets:*** **Intruder Tail Chase $\varphi_{2_{CL}}$ with initial ownship uncertainty $x_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

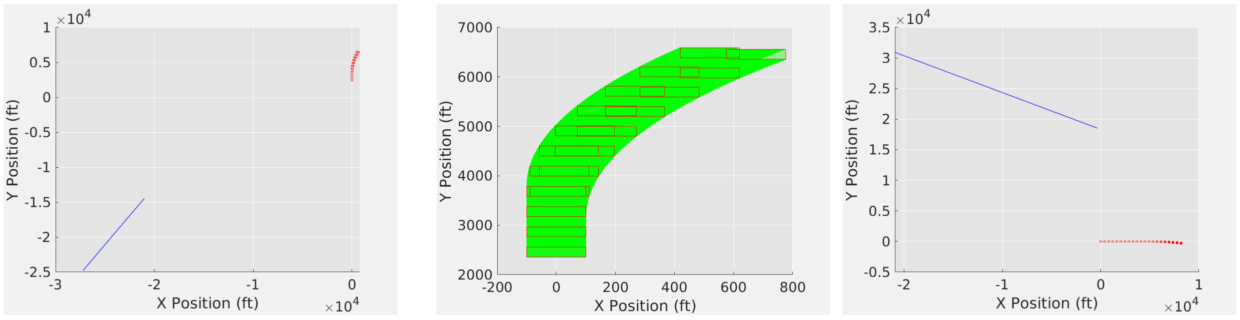


(b) Ownship step sets (red) and all sets (green). (Zoomed in)



(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

Fig. 8 ***Reachable Sets:*** **Right Gaining $\varphi_{3_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**

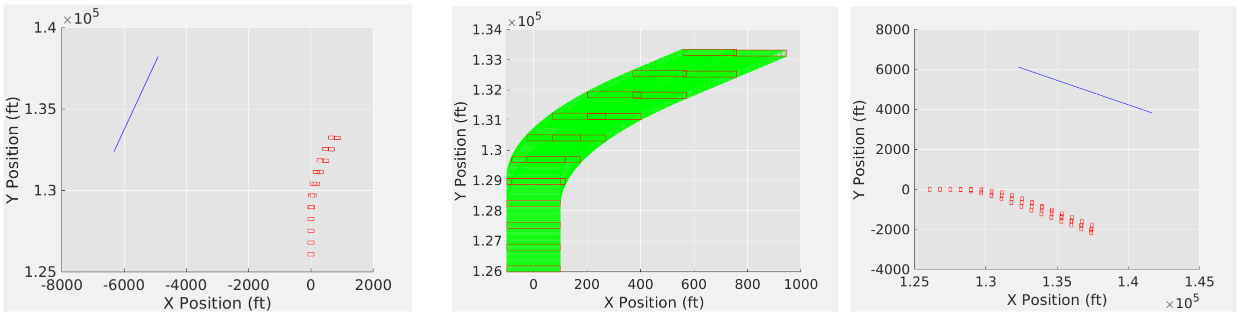


(a) Ownship (red) vs. intruder (blue)

(b) Ownship step sets (red) and all sets (green). (Zoomed in)

(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

Fig. 9 ***Reachable Sets:*** **Left Gaining $\varphi_{4_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**

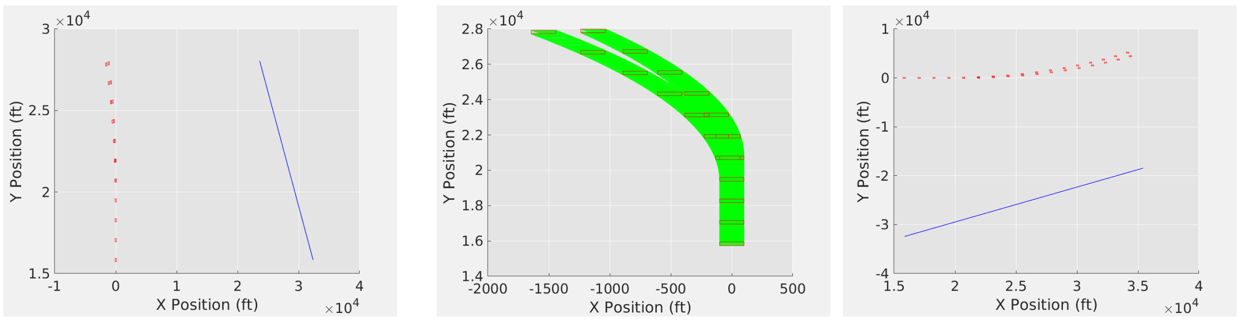


(a) Ownship (red) vs. intruder (blue)

(b) Ownship step sets (red) and all sets (green). (Zoomed in)

(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

Fig. 10 ***Reachable Sets:*** **Left Closing $\varphi_{5_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**

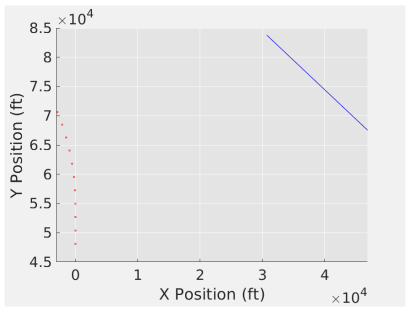


(a) Ownship (red) vs. intruder (blue)

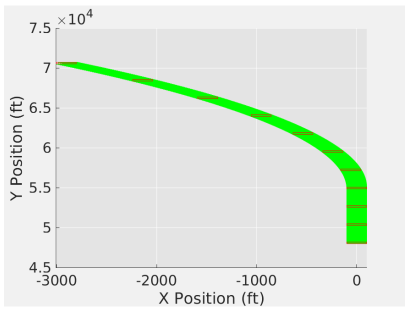
(b) Ownship step sets (red) and all sets (green). (Zoomed in)

(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

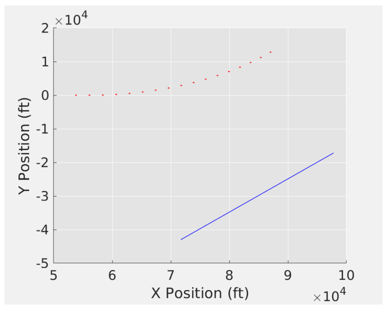
Fig. 11 ***Reachable Sets:*** **Right Abeam $\varphi_{6_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

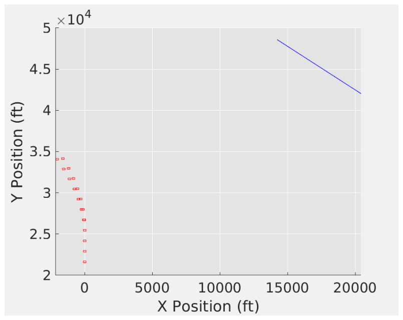


(b) Ownship step sets (red) and all sets (green). (Zoomed in)

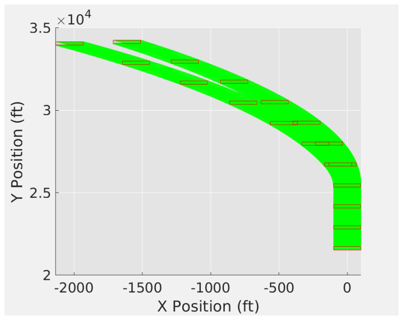


(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

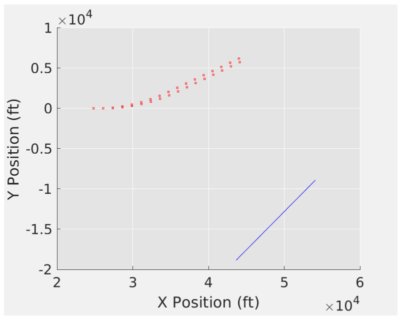
Fig. 12 ***Reachable Sets:*** **Right Isosceles $\varphi_{7_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

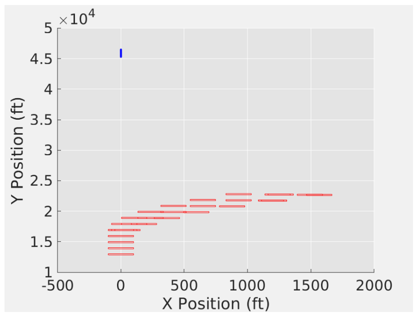


(b) Ownship step sets (red) and all sets (green). (Zoomed in)

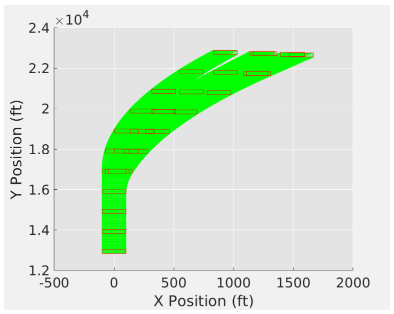


(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

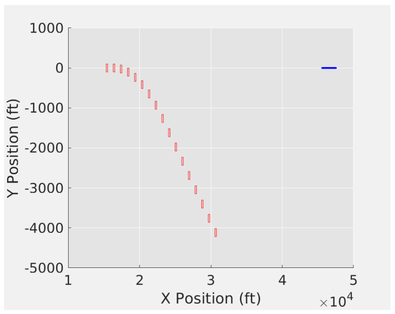
Fig. 13 ***Reachable Sets:*** **Right Closing $\varphi_{8_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

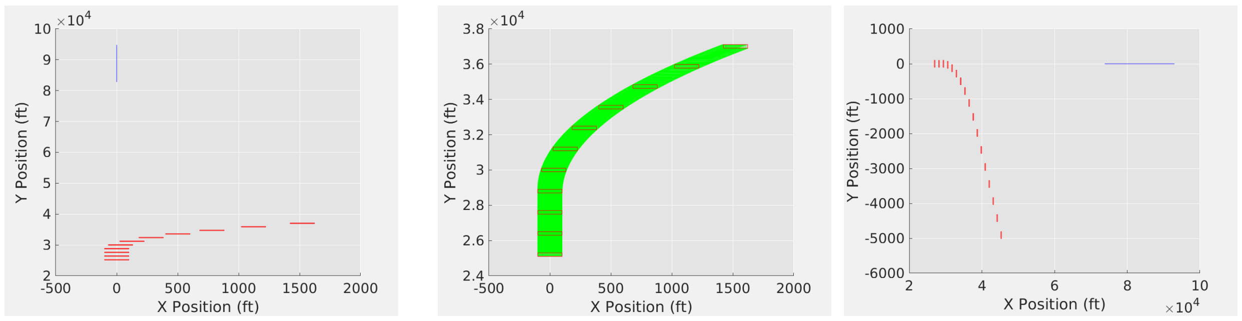


(b) Ownship step sets (red) and all sets (green). (Zoomed in)



(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

Fig. 14 ***Reachable Sets:*** **Ownship Tail Chase $\varphi_{9_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**



(a) Ownship (red) vs. intruder (blue)

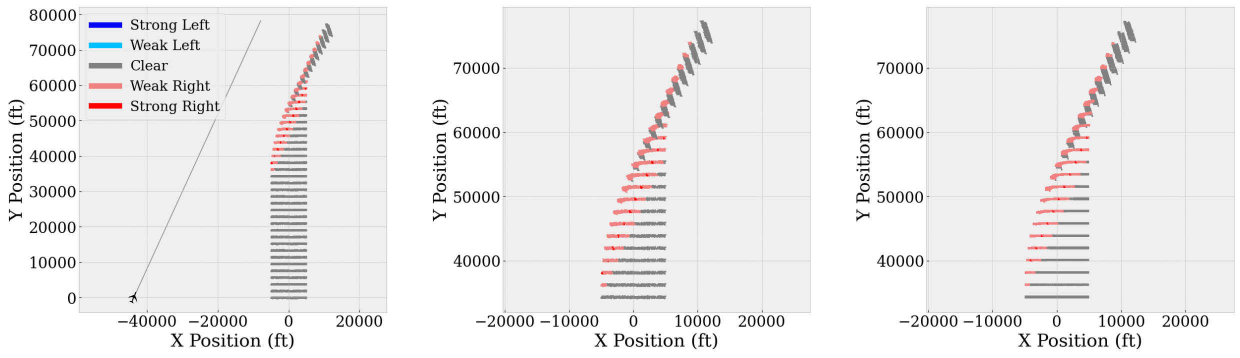
(b) Ownship step sets (red) and all sets (green). (Zoomed in)

(c) Ownship (red) vs. intruder (blue). (Rotation 90°)

Fig. 15 ***Reachable Sets:*** **Head On Collision $\varphi_{10_{CL}}$ with initial ownship uncertainty $x_o \pm 100$ and $y_o \pm 100$.**

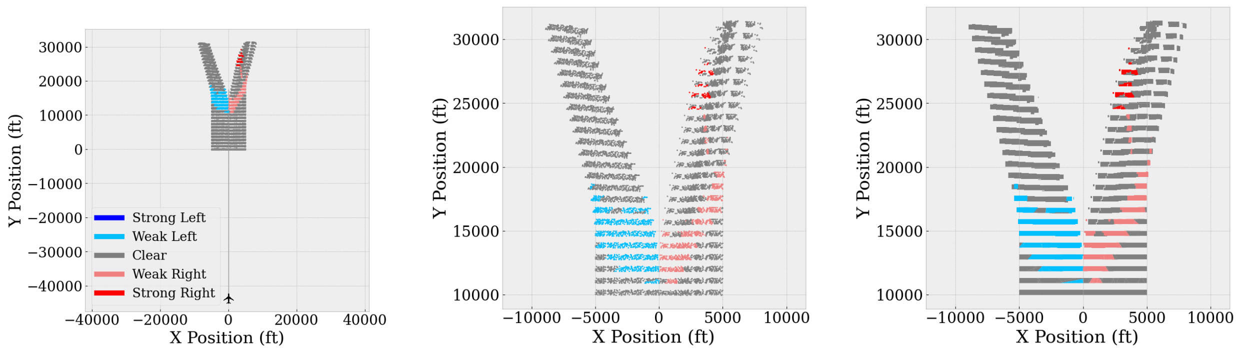
B. Results with the nnenum Tool

Simulations of the ten closed-loop scenarios and reach sets computed with nnenum are shown in Figs. 16-25. The plotted points are the locations where a decision was made by the ACAS Xu system, with the colors indicating the advisory issued at that location. For each scenario, an initial ownship uncertainty of ±5000 in the x direction and ±200 in the y direction is used. Although this meant splitting of star sets was needed due to multiple possible commands in a large set, it was still feasible to enumerate all possibilities in reasonable analysis time. This is because splitting causes the sets to become smaller, so that further splitting is less likely at future states. Further, since the regions are split during neural network analysis, the individual star sets become small, so that local linearization of the plant dynamics is accurate. This can be observed by the similarity of the simulations and computed reachable sets in the figures.



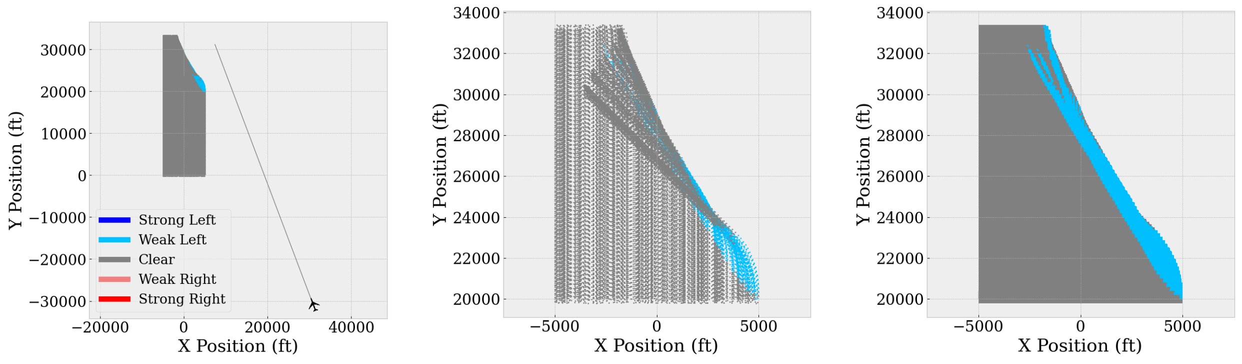
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 16 Left Abeam Scenario $\varphi_{1_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



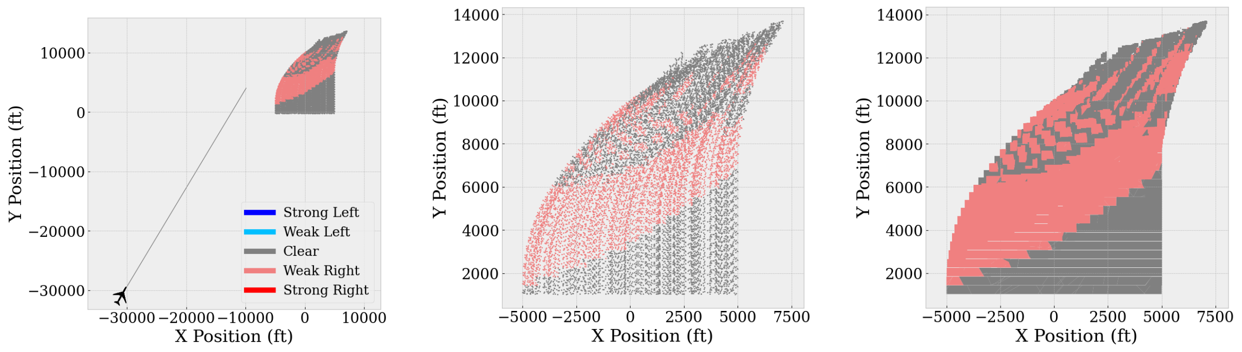
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 17 Intruder Tail Chase Scenario $\varphi_{2_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



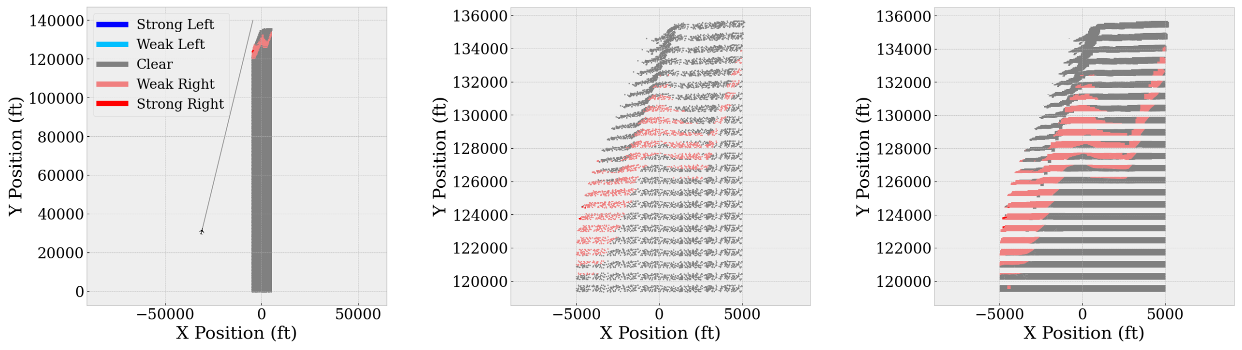
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 18 Right Gaining Scenario $\varphi_{3_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



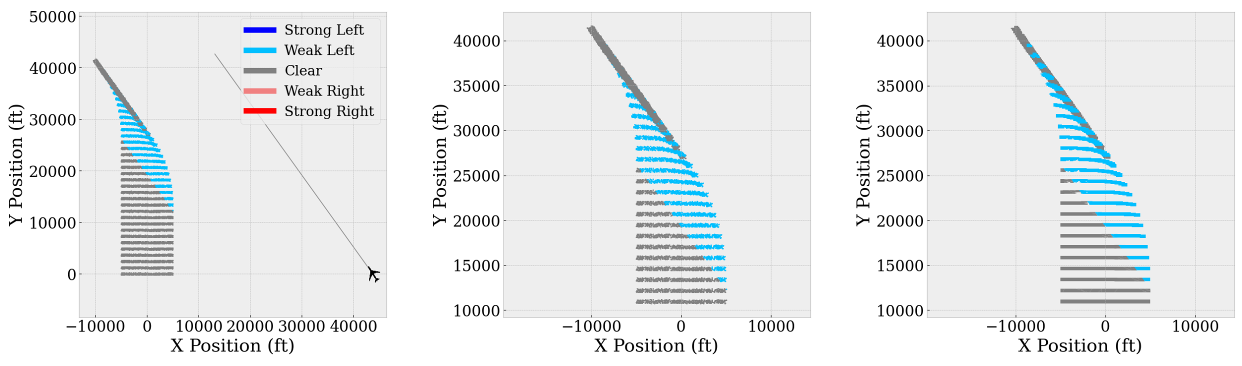
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 19 Left Gaining Scenario $\varphi_{4_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



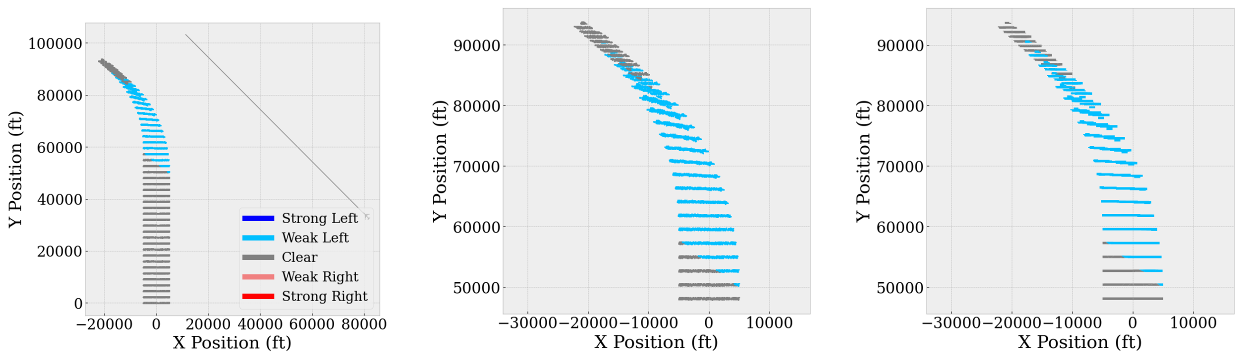
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 20 Left Closing Scenario $\varphi_{5_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



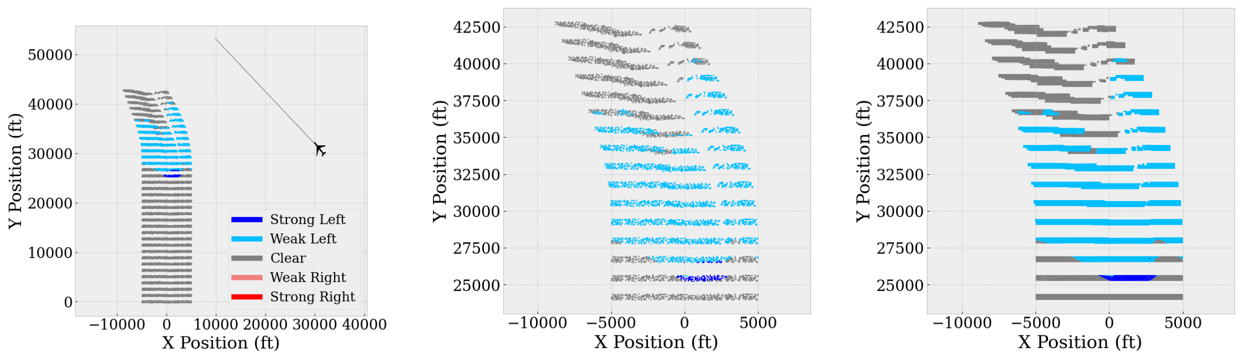
(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 21 Right Abeam Scenario $\varphi_{6_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 22 Right Isosceles Scenario $\varphi_{7_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.



(a) Simulations (b) Simulations (Zoomed In) (c) Reachable Set

Fig. 23 Right Closing Scenario $\varphi_{8_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.

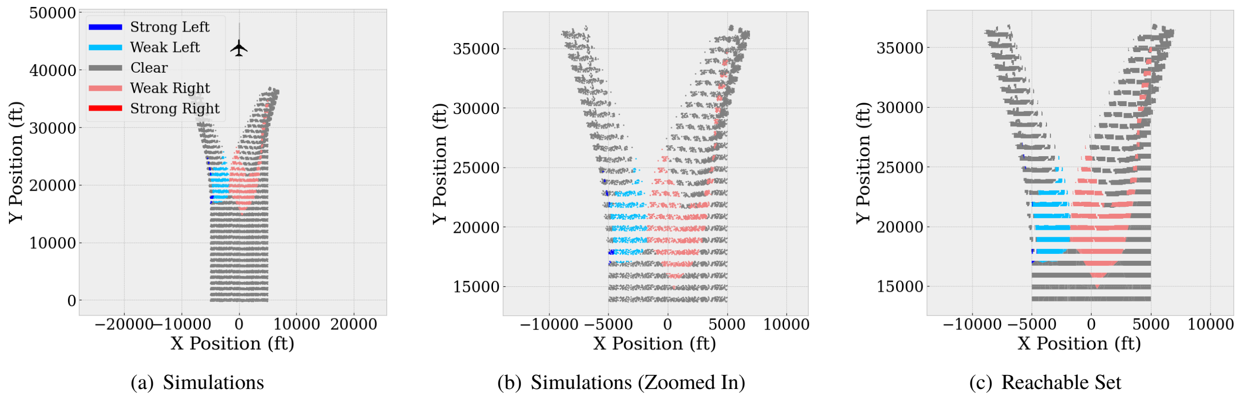


Fig. 24 Ownship Tail Chase Scenario $\varphi_{9_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.

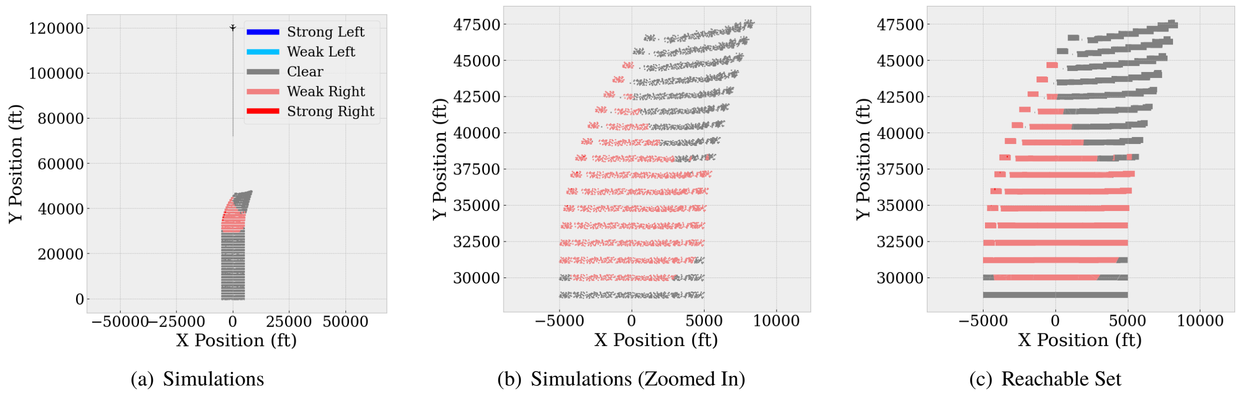


Fig. 25 Head On Collision Scenario $\varphi_{10_{CL}}$ with initial ownship uncertainty $x_0 \pm 5000$ and $y_0 \pm 200$.

VII. Conclusion

This paper proposes ten closed-loop test scenarios for the ACAS Xu Neural Network Control System, which are analyzed using two reachability analysis tools. The reachability tools are used to show that the neural network approximation of ACAS Xu maintains safety by avoiding intersection with the NMAC of an intruder aircraft, defined as maintaining a distance of 500 ft laterally and 100 feet vertically. Both NNV and nnenum reachability tools show the safety of the aircraft for all ten ACAS Xu benchmark closed loop test cases when uncertainty is added to the initial aircraft position.

This research was limited to 5 of the 45 ACAS XU neural networks that corresponded to co-altitude flight. Further research is needed to determine if climbing or descending flights can also be proven safe using these tools. For next steps, the team plans to add closed-loop benchmarks and to add uncertainty to other state variables and to analyze a more complex aircraft model based on the F-16 Fighting Falcon [36].

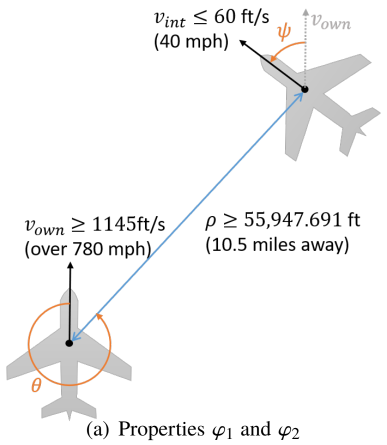
Appendix: Open Loop Verification Properties

An ACAS Xu verification benchmark proposed 10 open loop properties of the neural network approximation. These properties are summarized in this appendix with visual depictions of the geometry to give provide the reader with more intuition into how these neural networks have been verified in the past and how this compares to the closed loop test cases. It is worth noting that these neural networks assign a value to each of the five possible outputs with the convention that the *lowest* output is the best choice (in other neural network designs, the highest input may be the best choice).

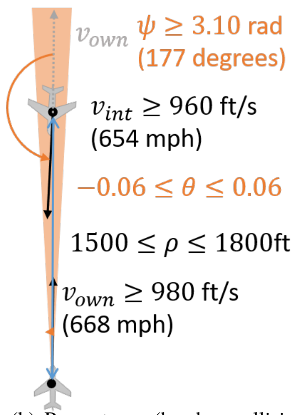
Table 5 ACAS Xu Benchmark Open Loop Properties.

	ρ (ft)	θ (rad)	ψ (rad)	v_{own} (ft/s)	v_{int} (ft/s)	Networks	Output
φ_1	$\geq$ 55,948	$\times$	$\times$	$\geq$1145	$\leq$ 60	All	COC $\leq$ 1500
φ_2	$\geq$ 55,948	$\times$	$\times$	$\geq$1145	$\leq$ 60	$N_{x\geq 2,y}$	COC not max
φ_3	$\in$ [1500,1800]	$\in [-0.06,0.06]$	$\geq$3.10	$\geq$980	$\geq$960	$\neq N_{1,y\geq 7}$	COC not min
φ_4	$\in$ [1500,1800]	$\in [-0.06,0.06]$	0	$\geq$1000	$\in [700, 800]$	$\neq N_{1,y\geq 7}$	COC not min
φ_5	$\in$ [250,500]	$\in [0.2, 0.4]$	$\approx -\pi$	$\in [100, 400]$	$\in [0, 400]$	$N_{1,1}$	SR min
φ_6	$\in$ [12000, 62000]	$\in [0.7, \pi] \vee \in [-pi, -0.7]$	$\approx \pi$	$\in [100, 1200]$	$\in [0, 1200]$	$N_{1,1}$	COC min
φ_7	$\in [0, 60760]$	$\in [-\pi, \pi]$	$\in [-\pi, \pi]$	$\in [100, 1200]$	$\in [0, 1200]$	$N_{1,9}$	SL,SR min
φ_8	$\in [0, 60760]$	$\in [-\pi, -0.75\pi]$	$\in [-0.1, 0.1]$	$\in [600, 1200]$	$\in [600, 1200]$	$N_{2,9}$	WL $\vee$ COC
φ_9	$\in [2000, 7000]$	$\in [-0.4, -0.14]$	$\approx -\pi$	$\in [100, 150]$	$\in [0, 140]$	$N_{3,3}$	SR min
φ_{10}	$\in [36000, 60760]$	$\in [0.7, \pi]$	$\approx -\pi$	$\in [900, 1200]$	$\in [600, 1200]$	$N_{4,5}$	COC min

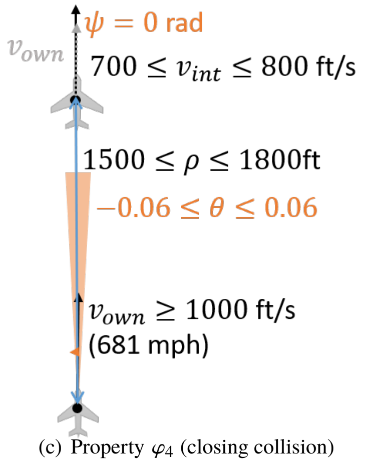
Property φ_1 may select a value of ρ (approximately 10.5 miles away) that is larger than the maximum turning radius of an aircraft going at 1145 ft/s (43,736). This maximum velocity of 1145 ft/s is approximately 780 miles per hour (mph), which as a conservative upper bound is much larger than what a commercial aircraft would typically fly (Mach



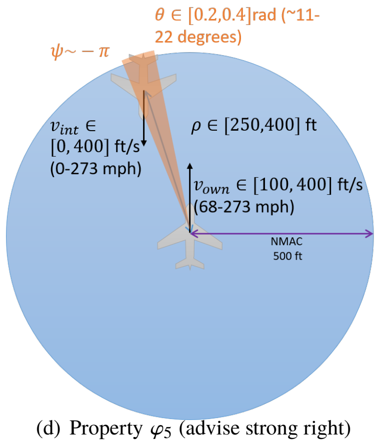
(a) Properties φ_1 and φ_2



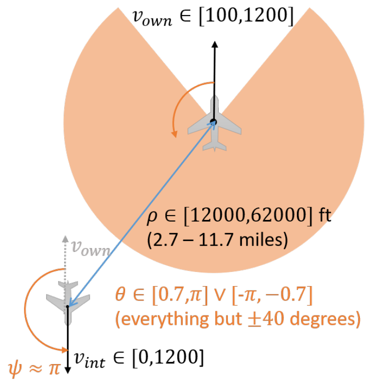
(b) Property φ_3 (head on collision)


$\psi = 0$ rad
v_{own}
$700 \leq v_{int} \leq 800$ ft/s
$1500 \leq \rho \leq 1800$ft
$-0.06 \leq \theta \leq 0.06$
$v_{own} \geq 1000$ ft/s
(681 mph)

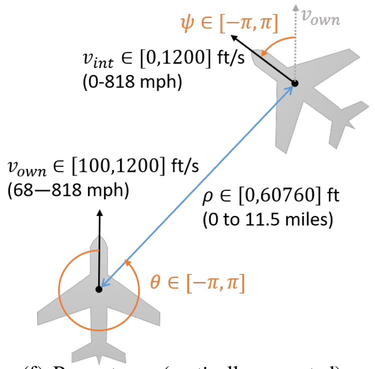
(c) Property φ_4 (closing collision)


$\theta \in [0.2{,}0.4]$rad (~11-22 degrees)
$\psi \sim -\pi$
$v_{int} \in$
$[0, 400]$ ft/s
(0-273 mph)
$\rho \in [250{,}400]$ ft
$v_{own} \in [100, 400]$ ft/s
(68-273 mph)
NMAC
500 ft

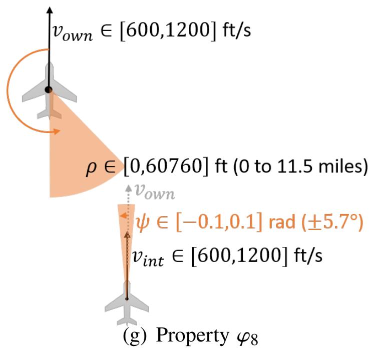
(d) Property φ_5 (advise strong right)


$v_{own} \in [100{,}1200]$
$\rho \in [12000{,}62000]$ ft
(2.7 – 11.7 miles)
v_{own}
$\theta \in [0.7,\pi] \vee [\text{-}\pi, -0.7]$
(everything but ±40 degrees)
$\psi \approx \pi$
$v_{int} \in [0{,}1200]$

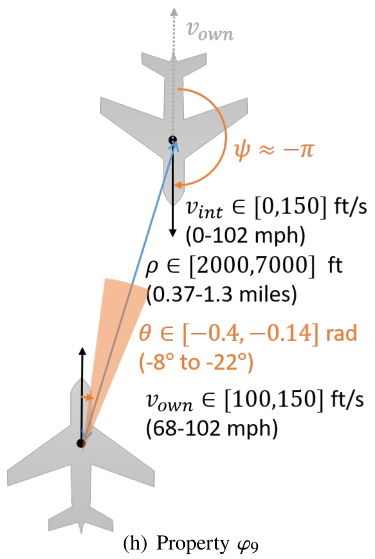
(e) Property φ_6 (moving away)



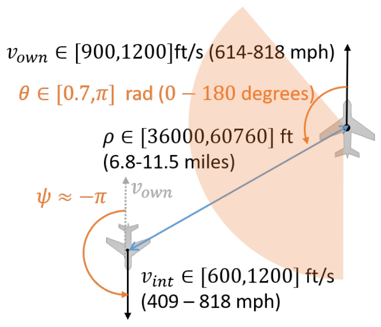
(f) Property φ_7 (vertically separated)


$v_{own} \in [600{,}1200]$ ft/s
$\rho \in [0{,}60760]$ ft (0 to 11.5 miles)
v_{own}
$\psi \in [-0.1{,}0.1]$ rad (±5.7°)
$v_{int} \in [600{,}1200]$ ft/s

(g) Property φ_8


v_{own}
$\psi \approx -\pi$
$v_{int} \in [0{,}150]$ ft/s
(0-102 mph)
$\rho \in [2000{,}7000]$ ft
(0.37-1.3 miles)
$\theta \in [-0.4, -0.14]$ rad
(-8° to -22°)
$v_{own} \in [100{,}150]$ ft/s
(68-102 mph)

(h) Property φ_9


$v_{own} \in [900{,}1200]$ft/s (614-818 mph)
$\theta \in [0.7,\pi]$ rad (0 – 180 degrees)
$\rho \in [36000{,}60760]$ ft
(6.8-11.5 miles)
$\psi \approx -\pi$
v_{own}
$v_{int} \in [600{,}1200]$ ft/s
(409 – 818 mph)

(i) Property φ_10

Fig. 26 Depiction of the aircraft encounter geometry for the open loop ACAS Xu verification properties

1 is 761 mph at sea level, decreasing to 678 mph at a 30,000 ft typical commercial aircraft cruise altitude). The 60 ft/s (40 mph) velocity of the intruder in this property is a conservative lowest velocity for a fixed wing aircraft. While the selection of variables seems reasonable, this first property isn't necessarily a good property because it specifies a value of the output which is an artefact of the design choices for the neural network rather than an ordinal ranking of the desired outcome for the aircraft encounter. In other words, a better property might specify that the selected action produced by the neural network should be clear of conflict. Property φ_2 is a slightly better variation of φ_1 because it test that clear of conflict is not the last choice of the neural network for a subset of the conditions of φ_1.

Properties φ_3 and φ_4 checks that the neural network does not output clear of conflict in cases where the intruder is ahead of the ownship and a collision is imminent. Property φ_3 checks when the two aircraft are approaching for a head on collision within less than a second (unless there is sufficient vertical separation). Property φ_4 checks when the ownship is directly behind and closing on the intruder at a rate of 200-300 ft/s, which will result in a collision in less than 9 seconds.

The remaining 6 properties check a wide range of conditions. Property φ_5 checks that the neural network favors the strong right advisory if the intruder is passing very close to the ownship's left. Property φ_6 checks that the neural network outputs clear of conflict if the intruder is sufficiently far away behind and moving away from the ownship. Property φ_7 check that the neural network does not output a strong left or right advisory when vertical separation is large ($\tau = 100$s) and the previous advisory was clear of conflict. Property φ_8 checks that the neural network outputs weak left or clear of conflict as the minimal score when the previous advisory was weak left, there is 100 seconds until a loss of vertical separation, and the intruder is behind and to the right of the ownship. Property φ_9 checks that the neural network outputs a strong left signal in a potential head on collision where the intruder is passing to the right. Property φ_{10} checks that the neural network outputs clear of conflict if the intruder aircraft is far away and moving away.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-0288, the National Science Foundation (NSF) under grant number FMitF 1918450, the Air Force Office of Scientific Research (AFOSR) through contract number FA9550-18-1-0122, and the Defense Advanced Research Projects Agency (DARPA) Assured Autonomy program through contract number FA8750-18-C-0089. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or NSF.

References

[1] Indurkhya, N., and Damerau, F. J., *Handbook of natural language processing*, Vol. 2, CRC Press, 2010.

[2] Tuia, D., Volpi, M., Copa, L., Kanevski, M., and Munoz-Mari, J., "A survey of active learning algorithms for supervised remote sensing image classification," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 5, No. 3, 2011, pp. 606–617.

[3] Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D., "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, Vol. 35, No. 1, 2018, pp. 84–100.

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., "Mastering the game of Go with deep neural networks and tree search," *nature*, Vol. 529, No. 7587, 2016, pp. 484–489.

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., "Mastering the game of go without human knowledge," *Nature*, Vol. 550, No. 7676, 2017, pp. 354–359.

[6] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, Vol. 575, No. 7782, 2019, pp. 350–354.

[7] Julian, K. D., Lopez, J., Brush, J. S., Owen, M. P., and Kochenderfer, M. J., "Policy compression for aircraft collision avoidance systems," *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–10.

[8] Liu, C., Arnon, T., Lazarus, C., Barrett, C. W., and Kochenderfer, M. J., "Algorithms for Verifying Deep Neural Networks," *CoRR*, Vol. abs/1903.06758, 2019. URL `http://arxiv.org/abs/1903.06758`.

[9] Xiang, W., Musau, P., Wild, A. A., Lopez, D. M., Hamilton, N., Yang, X., Rosenfeld, J. A., and Johnson, T. T., "Verification for Machine Learning, Autonomy, and Neural Networks Survey," *CoRR*, Vol. abs/1810.01989, 2018. URL `http://arxiv.org/abs/1810.01989`.

[10] Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I., "Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers," *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, ACM, New York, NY, USA, 2019, pp. 169–178. doi:10.1145/3302504.3311806, URL `http://doi.acm.org/10.1145/3302504.3311806`.

[11] Dutta, S., Chen, X., and Sankaranarayanan, S., "Reachability Analysis for Neural Feedback Systems Using Regressive Polynomial Rule Inference," *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, ACM, New York, NY, USA, 2019, pp. 157–168. doi:10.1145/3302504.3311807, URL `http://doi.acm.org/10.1145/3302504.3311807`.

[12] Sun, X., Khedr, H., and Shoukry, Y., "Formal Verification of Neural Network Controlled Autonomous Systems," *CoRR*, Vol. abs/1810.13072, 2018. URL `http://arxiv.org/abs/1810.13072`.

[13] Xiang, W., Lopez, D. M., Musau, P., and Johnson, T. T., “Reachable Set Estimation and Verification for Neural Network Models of Nonlinear Dynamic Systems,” 2019, pp. 123–144. doi:10.1007/978-3-319-97301-2_7.

[14] Xiang, W., Tran, H.-D., Rosenfeld, J., and Johnson, T. T., “Reachable Set Estimation and Verification for a Class of Piecewise Linear Systems with Neural Network Controllers,” *American Control Conference (ACC 2018), Special Session on Formal Methods in Controller Synthesis I*, IEEE, 2018.

[15] Huang, C., Fan, J., Li, W., Chen, X., and Zhu, Q., “ReachNN: Reachability Analysis of Neural-Network Controlled Systems,” *ACM Trans. Embed. Comput. Syst.*, Vol. 18, No. 5s, 2019. doi:10.1145/3358228, URL `https://doi.org/10.1145/3358228`.

[16] Ivanov, R., Carpenter, T. J., Weimer, J., Alur, R., Pappas, G. J., and Lee, I., “Case Study: Verifying the Safety of an Autonomous Racing Car with a Neural Network Controller,” *arXiv e-prints*, 2019, arXiv:1910.11309.

[17] Tran, H.-D., Cei, F., Lopez, D. M., Johnson, T. T., and Koutsoukos, X., “Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control,” *ACM SIGBED International Conference on Embedded Software (EMSOFT’19)*, ACM, 2019.

[18] Lopez, D. M., Musau, P., Hamilton, N., Tran, H.-D., and Johnson, T. T., “Case Study: Safety Verification of an Unmanned Underwater Vehicle,” *Workshop on Assured Autonomous Systems (WAAS)*, IEEE, 2020.

[19] Tran, H.-D., Yang, X., Lopez, D. M., Musau, P., Nguyen, L. V., Xiang, W., Bak, S., and Johnson, T. T., “NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems,” *32nd International Conference on Computer-Aided Verification (CAV)*, 2020.

[20] Julian, K. D., and Kochenderfer, M. J., “A Reachability Method for Verifying Dynamical Systems with Deep Neural Network Controllers,” *CoRR*, Vol. abs/1903.00520, 2019. URL `http://arxiv.org/abs/1903.00520`.

[21] Akintunde, M. E., Botoeva, E., Kouvaros, P., and Lomuscio, A., “Formal Verification of Neural Agents in Non-deterministic Environments,” *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, ACM, Auckland, New Zealand, 2020.

[22] Julian, K. D., Sharma, S., Jeannin, J.-B., and Kochenderfer, M. J., “Verifying aircraft collision avoidance neural networks through linear approximations of safe regions,” *arXiv preprint arXiv:1903.00762*, 2019.

[23] Tran, H.-D., Musau, P., Lopez, D. M., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., “Star-Based Reachability Analysis for Deep Neural Networks,” *23rd International Symposium on Formal Methods (FM’19)*, Springer International Publishing, 2019.

[24] Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T., “Improved Geometric Path Enumeration for Verifying ReLU Neural Networks,” *32nd International Conference on Computer-Aided Verification (CAV)*, 2020.

[25] Olson, W. A., “Airborne collision avoidance system x,” Tech. rep., Massachusetts Institute of Technology, Lincoln Laboratory, 2015.

[26] Kochenderfer, M. J., "Optimized airborne collision avoidance," *Decision making under uncertainty: theory and application*, 2015.

[27] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J., "Reluplex: An efficient SMT solver for verifying deep neural networks," *International Conference on Computer Aided Verification*, Springer, 2017, pp. 97–117.

[28] Marston, M., and Baca, G., "ACAS-Xu initial self-separation flight tests," 2015.

[29] Holland, J. E., Kochenderfer, M. J., and Olson, W. A., "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," *Air Traffic Control Quarterly*, Vol. 21, No. 3, 2013, pp. 275–297.

[30] Kochenderfer, M. J., and Chryssanthacopoulos, J., "Robust airborne collision avoidance through dynamic programming," *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, Vol. 130, 2011.

[31] Julian, K. D., Kochenderfer, M. J., and Owen, M. P., "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2019, pp. 598–608.

[32] Kochenderfer, M. J., and Monath, N., "Compression of optimal value functions for Markov decision processes," *2013 Data Compression Conference*, IEEE, 2013, pp. 501–501.

[33] Julian, K. D., and Kochenderfer, M. J., "Guaranteeing Safety for Neural Network-Based Aircraft Collision Avoidance Systems," *arXiv preprint arXiv:1912.07084*, 2019.

[34] *VFR cruising altitude or flight level*, 2003. 14 CFR §91.159.

[35] Bak, S., and Duggirala, P. S., "Simulation-equivalent reachability of large linear systems with inputs," *International Conference on Computer Aided Verification*, Springer, 2017, pp. 401–420.

[36] Heidlauf, P., Collins, A., Bolender, M., and Bak, S., "Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers," *5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, EasyChair, 2018.