Evaluation of Neural Network Verification Methods for Air to Air Collision Avoidance *

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, 45433

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. An example of such systems is the unmanned Airbone Collision Avoidance System (ACAS Xu), which is a very popular benchmark for open-loop neural network control system verification tools. This paper proposes a new closed loop extension of this benchmark, which consists of a set of ten closed loop properties selected to evaluate the safety of an ownship aircraft in the presence of a co-altitude intruder aircraft. These closed loop safety properties are used to evaluate 5 of the 45 neural networks that comprise the ACAS Xu benchmark (corresponding to co-altitude cases) as well as the switching logic between the 5 neural networks. The combination of nonlinear dynamics and switching between five neural networks is a challenging verification task accomplished with star set reachability methods in two verification tools. The safety of the ownship aircraft under initial position uncertainty is guaranteed in every scenario proposed.

Nomenclature

ρ = distance between ownship and intruder, ft

θ = angle to intruder w.r.t ownship heading, rad

*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.

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

Θ = tuple

$[\![\Theta]\!]$ = set of states

W = mapping matrix

b = offset vector

α = vector

κ = center vector

V = set of basis vectors

μ = basis vector

P = predicate

G = polyhedron

H = constraint matrix

d = constraint vector

Subscripts

cg = center of gravity

int = intruder

own = ownship

G = polyhedron

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 (ACAS X) to 5 megabytes of neural network weights [7]. However, verification of machine learning in safety-critical system domains has historically been limited to frameworks that treat neural networks like black boxes. Neural networks (NN) are not black boxes, they compute deterministic mathematical functions that are increasingly amenable to formal reasoning methods [8, 9]. While many of these methods focus on analyzing the networks in isolation, several recent methods attempt to verify neural network control systems (NNCS) [10–22].

Despite the recent progress in NNCS verification, developing scalable and non-conservative methods remains a key issue. Due to this growing trend and remaining challenges, several friendly competitions have been recently organized to compare the performance of these verification tools, including [23] for NN verification and [24] for NNCS verification. In addition to these competitions, studies by Ivanov et al. [25], Tran et al. [18] and D. Manzanas Lopez et al. [20], have investigated the limits and capabilities of existing verification methods and tools [10, 19], and the trade-offs between scalability and conservativeness. Similar to the proposed NNCS in this work, Julian and Kochenderfer [26] and Akintunde et al. [16, 17] have formally verified another collision avoidance system for aircraft. Specifically, a closed-loop variant of the simplified version of the vertical avoidance control system in aircraft, the Vertical Collision Avoidance System (VerticalCAS) [27], is verified using reachability analysis and Mixed Integer Linear Programming (MILP) methods respectively. In addition to the VerticalCAS, Julian and Kochenderfer [28, 29] presented the same formal approach to verify the safety of Horizontal Collision Avoidance System (HorizontalCAS), which is another advisory system inspired by early prototypes of the ACAS Xu and presents the same input-output space of the ACAS Xu system, but with smaller size neural networks. Irfan et al. [21] formally verified the advisory system for small unmanned aircraft [30] (ACAS sXu) via symbolic interval reachability analysis with ReluVal [31]. The research studied a different state space and was comprised of smaller neural network controllers than the ACAS Xu NNCS presented here. Similarly, Claviere et al. [22] use ReluVal to formally analyze one test case of the ACAS Xu NNCS in combination with

a validated simulation approach to approximate the reachable sets of the plant. This manuscript aims to extend and improve the verification results of these studies to a comprehensive set of benchmarks in which two aircraft are travelling at the same altitude. These benchmarks are evaluated using star set reachability methods via the NNV [19] and nnenum [32] NNCS verification tools, which improve the scalability and over-approximation tightness of the symbolic interval analysis approaches. To the best of our knowledge, without any additional modifications or enhancements to existing NNCS reachability tools, these are the only NNCS tools that are able to verify this benchmark due to the switching behavior of the classification-based controllers. These improvements are specially notable in 7 of the 10 test cases where the intruder is approaching the ownship from either side. The symbolic interval analysis methods in [22] are only able to verify 75% of the area considered in these cases, while increasing the verification time between 1 and 2 orders of magnitude with respect to other areas. On the other hand, the star set methods utilized in this work are able to verify the safety of the entire region considered for each of these test cases whilst maintaining a similar computation time as the other experiments.

The main contributions of this paper are:

- A description of the ACAS Xu neural network compression closed loop benchmark and definition of ten closed loop verification properties.
- Improved scalability and over-approximation tightness in star-set reachability methods by developing a verification approach for neural network closed loop systems with switching classification-based controllers using star sets.
- Successful verification of 5 of 45 ACAS Xu Neural Networks corresponding to cases with no vertical separation, and the switching behavior against all ten safety properties using two reachability analysis tools: NNV and nnenum.

The rest of the paper is organized as follows. Section II describes the ACAS Xu NN compression system, and Section III describe the plant model used as well as the NNCS benchmark. Section IV describes the closed loop test case generation of safety properties while in Section V the evaluation approaches used are defined. Finally, section VI introduces the results and section VII provides the concluding remarks and future work plans.

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 [33]. 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 [33].

There are five variants of ACAS X [33], and this manuscript provides a collection of 10 closed loop properties to analyze the 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 [34].

- ACAS Xu (unmanned): Optimized for unmanned aircraft systems (UAS), ACAS Xu issues turn rate advisories to remote pilots [35].
- ACAS Xo (operation): special alerts during operations such as parallel runway approaches [33].
- ACAS Xp (passive): tracks aircraft for potential collisions, but doesn't produce advisories [33].
- ACAS sXu (small unmanned): Designed for small UAS (sUAS) to avoid collision with manned aircraft, UAS, and other sUAS [30].

Both ACAS Xa and ACAS Xu have a goal to avoid near midair collisions (NMAC) [36], defined as separation less than 100 ft vertically and 500 ft horizontally [37], as depicted in Fig. 1.

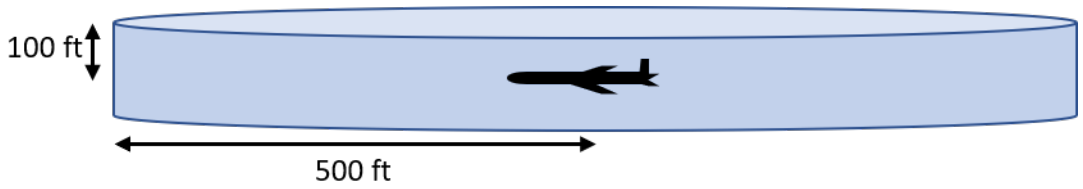


Fig. 1 Near midair collision cylinder, defined as separation less than 100 ft vertically and 500 ft horizontally.

ACAS X functionality centers on the use of a set of look up tables generated offline with dynamic programming and Markov decision processes (MDPs) [38]. The input to these lookup tables is a probabilistic state distribution of the relative position and velocity of nearby aircraft approximated from sensor data. The output of the lookup tables is a cost used by ACAS X to decide the optimal advisory to provide to the pilot: no alert, traffic advisory, or a resolution advisory for pilots to maintain or increase safe separation from other aircraft [33].

This manuscript focuses on analysis of the ACAS Xu variant of ACAS X, described briefly in Section II.A, and specifically, a neural network compression of this ACAS Xu algorithm, described in Section II.B. The ACAS Xu neural networks receive actual states of the system, rather than probabilistic state distributions from sensor data, and output the most optimal action.

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 [39] 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 [39]. To deal with sensor uncertainty in the system state, ACAS Xu uses an unscented Kalman filter [40].

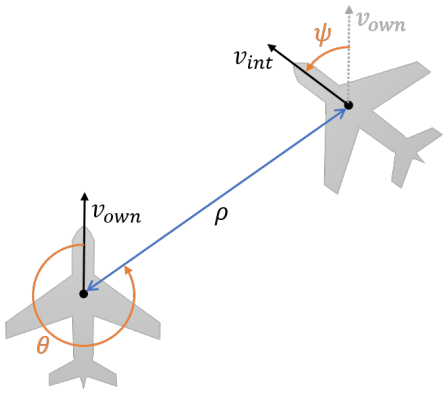


Fig. 2 Diagram of the ACAS Xu physical variables.

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

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 the 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 [41]. However, ACAS Xu's 2 gigabyte size may still be too large for the storage constraints of certified avionics hardware on UAS[28].

Developed in [7] and evaluated in [35], 45 separate neural networks were used to compress the lookup table. Each network is denoted $N_{\gamma,\beta}$, where γ corresponds to the index (1 to 5) of a specific value of previous advisory $a_{prev} \in$

$\{COC, WL, WR, SL, SR\}$ and β 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 (ρ, θ, ψ, v_{own}, and v_{int}) 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.

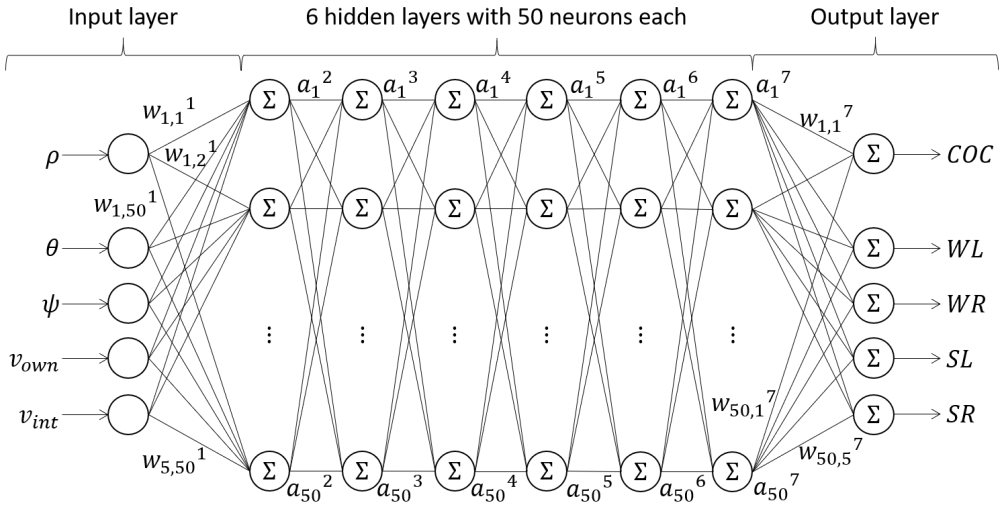


Fig. 3 Depiction of the ACAS Xu neural network.

C. Open-Loop vs. Closed-Loop Verification

Previous research has verified the neural networks using a set of open-loop test points [27, 28, 35, 39]. 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 [35] became a popular benchmark for open loop verification, and are summarized here in the Appendix. 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. The development and description of a set of closed-loop safety properties is one of the main contributions of this paper.

D. Set-based Verification Compared to Monte Carlo and Design of Experiments

Set-based verification is a possible alternative to Monte Carlo simulation, which uses thousands or millions of random samples to evaluate performance across the system state space [42]. An alternative approach to Monte Carlo,

especially when evaluating individual points is computationally expensive, is to apply Design of Experiments to sample the state space in a structured way rather than randomly [43]. Latin Hypercube can be applied to Monte Carlo or Design of Experiments approach to use a reduced sample spread evenly across the state space [44]. However, Monte Carlo and Design of Experiments are not as comprehensive as set-based neural network verification, which is the focus of this manuscript.

III. Model

The verification effort in this paper focuses on verifying 5 of 45 total neural networks corresponding to cases where the loss of vertical separation τ is 0 (N_{11}, N_{12}, N_{13}, N_{14}, and N_{15}). This reduces the problem to a two-dimensional co-altitude case. Nonetheless, verification of these neural networks and the switching between them present a significant challenge. Focusing on the co-altitude cases, the nonlinear plant model of the intruder and ownship aircraft are both represented using Dubins aircraft model described in Eq. 1.

$$\begin{aligned} \dot{x}_1 &= \dot{x} = v cos(\psi), \\ \dot{x}_2 &= \dot{y} = v sin(\psi), \\ \dot{x}_3 &= \dot{\psi} = u, \end{aligned} \tag{1}$$

This is a simplified yet reasonable model of the aircraft dynamics at a similar level of abstraction as the inputs to the ACAS Xu NNCS, which represents the aircraft in terms of velocities, headings, and distances. However, the ACAS Xu NN model requires inputs in terms of distance ρ, angle to intruder from ownship θ, and heading of intruder with respect to ownship ψ. Conversion between the states of the Dubins model of each aircraft and the variables used as inputs to the NN is achieved via a set of complex nonlinear functions defined in Eq. 2.

$$\begin{aligned} \rho &= \sqrt{(x_{int} - x_{own})^2 + (y_{int} - y_{own})^2} \\ \theta &= 2tan^{-1}(\frac{y_{int} - y_{own}}{x_{int} - x_{own} + \rho}) - \psi_{own} \\ \psi &= \psi_{int} - \psi_{own} \end{aligned} \tag{2}$$

Verification tools are often not able to encode these equations. Thus, these functions are converted to ordinary differential equations (ODEs) by computing their time derivatives. Finally, these and the Dubins models are combined into a single nonlinear model with 9 state variables as described in Eq. 3.

$$
\begin{aligned}
\dot{x}_1 &= \dot{x}_{own} = v cos(\psi_{own}),\\
\dot{x}_2 &= \dot{y}_{own} = v sin(\psi_{own}),\\
\dot{x}_3 &= \dot{\psi}_{own} = u,\\
\dot{x}_4 &= \dot{x}_{int} = v cos(\psi_{int}),\\
\dot{x}_5 &= \dot{y}_{int} = v sin(\psi_{int}),\\
\dot{x}_6 &= \dot{\psi}_{int} = c,\\
\dot{x}_7 &= \dot{\rho} = \frac{(y_{int} - y_{own})(\dot{y}_{int} - \dot{y}_{own}) + (x_{int} - x_{own})(\dot{x}_{int} - \dot{x}_{own})}{\sqrt{(x_{int} - x_{own})^2 + (y_{int} - y_{own})^2}},\\
\dot{x}_8 &= \dot{\theta} = \frac{2(\dot{y}_{int} - \dot{y}_{own})(x_{int} - x_{own} + \rho) - 2(y_{int} - y_{own})(\dot{x}_{int} - \dot{x}_{own} + \dot{\rho})}{(y_{int} - y_{own})^2 (x_{own} - x_{int} + \rho)^2} - \dot{\psi}_{own},\\
\dot{x}_9 &= \dot{\psi} = \dot{\psi}_{int} - \dot{\psi}_{own}
\end{aligned}
\tag{3}
$$

ACAS Xu NN input ranges require that θ and ψ must be in the $[-\pi, \pi]$ interval. Thus, it is necessary to ensure that the angles are always defined in the $[-\pi, \pi]$ range for each control step.

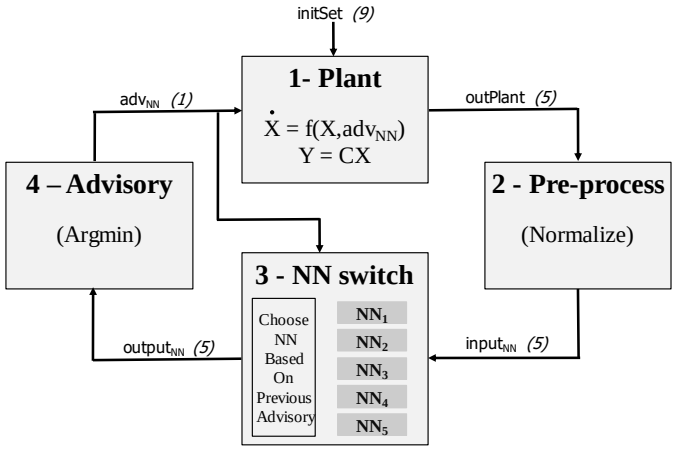


Fig. 4 Diagram of the ACAS Xu neural network closed loop system.

A. NNCS benchmark

Combining the ACAS Xu neural network system and the nonlinear dynamical model, the ACAS Xu NNCS benchmark is presented in a diagram in Fig. 4. There are four main components or operations highlighted:

- 1: plantReach(): The plant dynamics are simulated based on a specified initial state (initSet) and advisory (adv_{NN}) for two seconds (sampling time).
- 2: normalizeNNinputs(): Then, the outputs of the plant (outPlant) are scaled and normalized.

- 3 and 4: NNreach():
  - 3:Reach(): The scaled outputs (input_{NN}) are processed by the corresponding neural network controller, which is selected based on the previous advisory and the loss of vertical separation, as described in Section II.B.
  - 4: argminNN(): Based on the five outputs of the neural network (output_{NN}), the turn rate advisory (adv_{NN}) is computed (argmin) and commanded to the plant.

The name of the variables being processed by the next component are defined on top of the arrows and their respective dimensions are specified in parenthesis, i.e. outPlant (5) is a five dimensional variable output in 1) and normalized in 2). It is assumed that the velocities and the flight altitude of both aircraft is constant. Thus, the ACAS Xu neural network system is able to choose between the five neural networks corresponding to no time to loss of vertical separation ($\tau = 0$). The sampling time of 2 seconds is selected based on the dynamics of the airplane, which doing it too often and changing the direction of the trajectory may be chattering, while doing it too infrequently may lead to crashes.

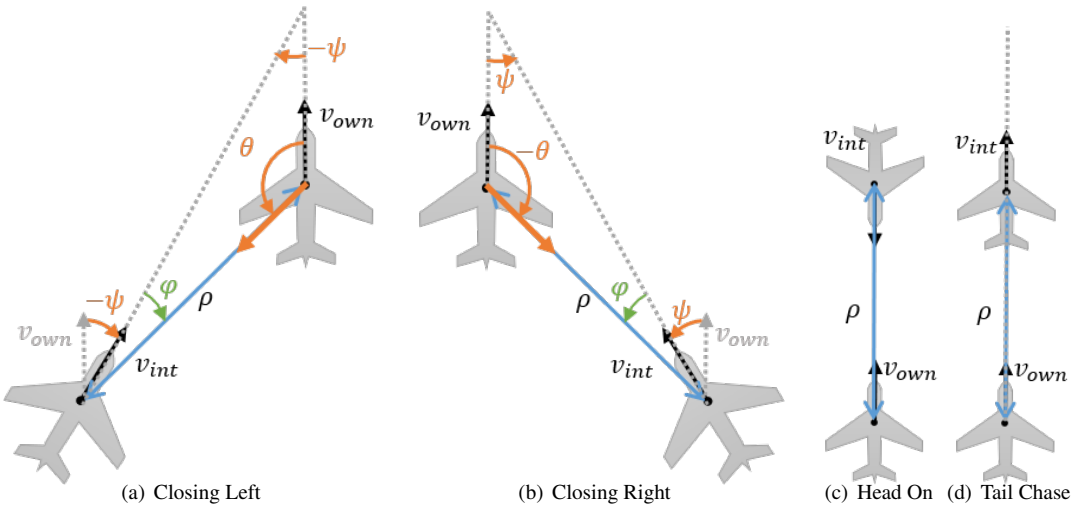
IV. Closed Loop Verification Test Case Generation

The objective of the closed loop verification tests is to demonstrate that the ownship and intruder aircraft remain safely separated given a set of initial conditions with uncertainty. The benchmarks presented here are not intended to provide complete coverage of the state space, but rather provide a challenge for NNCS verification approaches with a collection of 10 closed loop properties or test cases defined in Table 4 and depicted in Fig. 6. The closed loop properties presented in this paper are a complement to the open loop properties, both of which sample portions of the state space. The closed loop cases specifically sample from a set of cases on a collision course, while the open loop cases do not specifically look at collision cases. A larger sampling outside of the 10 individual test cases is created by adding uncertainty about these points (for example ±5000 ft in x and ± 200 ft in y). Then the set of states reachable from the initial set of states is found using reachability tools such as NNV or nnenum. These benchmarks are not intended to provide complete coverage of the state space, but rather demonstrate the capabilities of star set-based verification approach, a more comprehensive approach with formal guarantees, as an alternative to Monte Carlo or Design of Experiments-based simulation analysis.

For the closed loop benchmarks, 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). These limits, including different velocity ranges for the ownship and intruder, are inspired by previous work in defining open loop verification properties [35]. 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. 4, and four examples (closing left, closing right, head on and tail chase) are depicted in Fig. 5. 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.

$$\frac{\rho}{\sin\psi} = \frac{v_{own}T}{\sin(\theta - \pi - \psi)} = \frac{v_{int}T}{sin(2\pi - \theta)} \tag{4}$$



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

Fig. 5 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: $\theta \in \{-3\pi/4, -\pi/2, -3\pi/8, -\pi/4, 0, \pi/4, \pi/3, 3\pi/4, \pi\}$ 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_{int} 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. 5-6 are used. With the exception of the head on collision test case 10, represented by Fig. 5(c), each of the test cases are selected so that both aircraft are traveling in roughly the same direction, as depicted in Fig. 6. This is because aircraft are generally separated by at least 1000 feet in altitude when traveling in opposite directions, as commanded by 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 for pilots operating under visual flight rules) when traveling East on a heading between 0-179 degrees, or an even altitude (plus 500 feet for pilots operating under visual flight rules) when traveling West on a heading between 180-359 degrees [45]. This work did not consider non-straight trajectories of the ownship and intruder, which would need to be considered in a more comprehensive safety assessment.

$$x_{int} = -\rho \sin(\theta) \tag{5}$$

$$y_{int} = \rho \sin(\pi/2 - \theta) \tag{6}$$

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

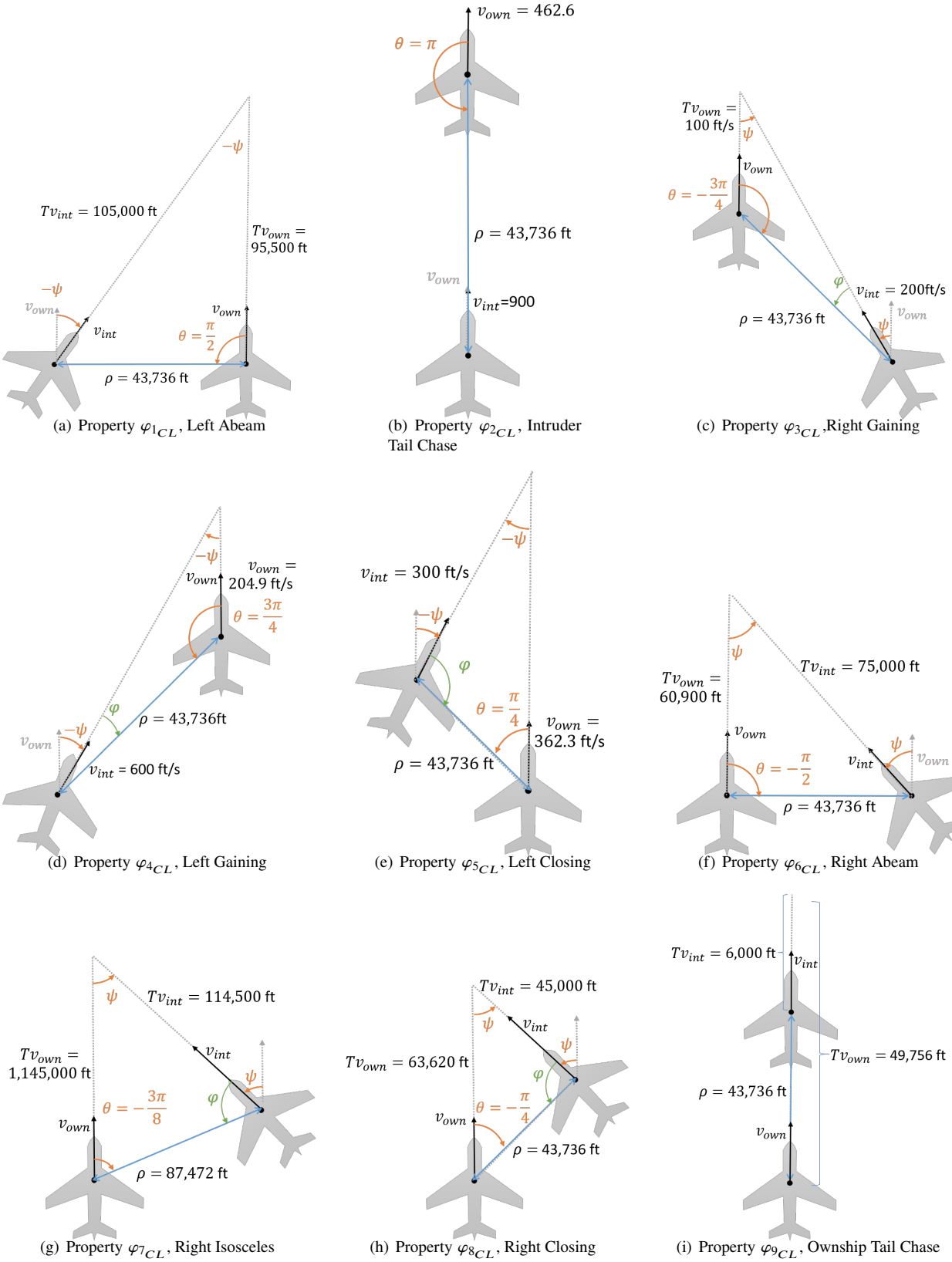
This section describes the fundamental theory behind star sets used to approximate reachable sets, as well as the closed loop neural network verification approaches. In terms of complexity, the reachability analysis of nonlinear ODEs is undecidable [46], NN reachability problem is NP-complete [35, 47], thus the reachability analysis of the NNCS with nonlinear ODE plant is undecidable.

A. Star Set Theory

A star set [48, 49] (or simply star) Θ is a tuple $\langle \kappa, V, P \rangle$ where $\kappa \in \mathbb{R}^n$ is the center vector, $V = \{\mu_1, \mu_2, \cdots, \mu_m\}$ is a set of m basis vectors in $\mathbb{R}^n$, and $P : \mathbb{R}^m \rightarrow \{\top, \bot\}$ is a predicate. The set of states represented by the star is given as:

$$[\![\Theta]\!] = \{\omega \mid \omega = \kappa + \Sigma_{i=1}^{m}(\alpha_i \mu_i) \text{ such that } P(\alpha_1, \cdots, \alpha_m) = \top\}. \tag{7}$$

Sometimes both the tuple Θ and the set of states $[\![\Theta]\!]$ are referred to as Θ. In this work, the predicates are restricted to be a conjunction of linear constraints, $P(\alpha) \triangleq H\alpha \leq d$ where, for p linear constraints, $H \in \mathbb{R}^{p \times m}$, α is the vector of



(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. 6 Depiction of the aircraft encounter geometry for the closed loop ACAS Xu verification properties

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 polyhedron can be represented as star set. An affine mapping of a star set $\Theta = \langle \kappa, V, P\rangle$ defined by a mapping matrix W and an offset vector b is another star set $\Theta' = \langle W\kappa + b, WV, P\rangle$. The intersection of a star set $\Theta = \langle \kappa, V, P\rangle$ with a polyhedron $G \triangleq H_G\alpha \leq d_G$ is another star set $\Theta' = \langle \kappa, V, P \wedge P_G\rangle$, where $P_G \triangleq H_G \times V\alpha \leq d_G - H_G \times \kappa$.

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 the ten closed-loop benchmarks where the safety of the ownship is analyzed with respect to one intruder aircraft. The analysis consists of evaluating, under multiple scenarios and initial state uncertainty, the safety of the ownship aircraft using Algorithm 1, which follows the general computation flow to the one described in section III.A. For the nonlinear plant reachability analysis NNV makes use of a CORA [50] integration.

Algorithm 1: ACAS Xu NNCS reachability algorithm in NNV

Result:
Rs: Plant reachable states
Initialize
$initSet$ // Initial state set
a_{prev} // Previous advisory
tF // Number of control steps
$safe$ = True // Safety variable, change to false if NMAC violated
v_{int} // Intruder velocity
v_{own} // Ownship velocity
while *safe* **do**
  for *(*$i = 0;\ i < tF;\ i = i + 1$*)* **do**
    [state_set,outPlant] = plantReach(initSet , a_{prev}) // reachability analysis of the plant
    ρ = extractDist(state_set) // Extract distance set
    if $\rho < 500$ **then**
      safe = False
    end
    $input_{NN}$ = normalizeNNinputs(outPlant) // normalize inputs to the neural network
    adv_{NN} = NNreach(a_{prev}, $input_{NN}$) // reachability analysis of the Nueral Network (NN) controller, see Algorithm 2.
    initSet = state_set
    a_{prev} = adv_{NN}
    Rs[i] = state_set
  end
end
return: Rs

The sets are represented using star-sets and the reachability functions compute over-approximations of the reachable sets. Due to the initial uncertainty added to the initial states of the aircraft, it is possible that multiple advisories are issued at each time step, increasing the number of possible trajectories of the ownship. In this case, the reach sets are split and paired with the corresponding advisory to decrease the number of operations and trajectories computed in

*https://github.com/verivital/nnv

order to reduce the conservativeness and memory consumption. The workflow of this algorithm implemented in NNV using star-set methods is presented in Algorithm 2. The importance of this algorithm is illustrated with the following example with one initial advisory and one initial state set, where the number of operations is exponentially reduced. For the purpose of this example, it is assumed that at each time step, there are 2 possible advisories (output_{NN}) issued for each plant output (output_P) set computed.

EXAMPLE 1.

Step 1) There is **1** output_P set computed based on the initial state and previous advisory. For this output_P set, there are 2 possible advisories (output_{NN}) computed by a NN. *Step 2)* For each advisory, one output_P is computed based on the initial state, obtaining **2** sets. For each possible combination of advisories and output_P sets (4), there are 2 advisories issued, increasing the number of advisories to 8. However, since there is a limit of five maximum unique advisories (*COC,WL,WR,SL,SR*), these 5 advisories are utilized for the remainder of the example. *Step 3)* There are 5 advisories and 2 state sets, increasing the number of plant output_P sets to **10** in step 3. By computing all possible combinations between the 10 output_P sets and 5 advisories, there are a total of 50 NN output_{NN} sets to compute, although the number of advisories computed is limited to 5 as previously described. *Step 4)* There are now 10 initial sets and 5 advisories, out of which **50** plant output_P sets are computed. Following this procedure, it is observed that the number of output_P sets computed grows by a multiple of 5 at each future control step, i.e. in *Step 5)* there are **250** plant output_P sets, in *Step 6)* there are **1250** sets and so on.

EXAMPLE 2.

The number of plant output_P sets can be significantly reduced by using Algorithm 2. *Step 1)* There is **1** output_P set, for which there are 2 possible advisories computed by a NN. *Step 2)* Based on the initial state set, for each of the 2 advisories the output_P set of the plant is computed, obtaining **2** sets. By tracking which set corresponds to which advisory, the number of operations are reduced from 4 to 2. For each combination, there are 2 advisories issued, increasing the number of advisories to 4. *Step 3)* There are 4 advisories and 2 output_P sets. For each advisory, the corresponding plant output_P set is computed for a total of **4** plant output_P sets. Then, for each output_P set, there are 2 advisories issued for a total of 8 advisories. *Step 4).* The relationship between the advisories and the plant output_P sets computed is one to one, therefore **8** sets are obtained. Following this pattern, it is observed that the number of plant output_P sets increases by 2 for each control step, i.e. in *Step 5)* there are **16** plant output_P sets, in *Step 6)* there are **32** sets and so on. If both examples are run for 10 control steps, the total number of output_P sets are reduced from **781250** to **512**.

C. Neural Network Enumeration (nnenum) Tool

The second verification tool used for analysis is nnenum[†], which uses star sets to reason over possible neural network outputs given sets of input states. Nnenum combines many engineering improvements to the base star set method [51],

[†] https://github.com/stanleybak/nnenum

Algorithm 2: Reachability algorithm of switching classification based controllers — *NNreach*

Result:
adv_{NN}: Advisories paired with each plant reach set
Inputs
a_{prev} // Previous advisory
$input_{NN}$ // inputs to NNs
N // number of reach sets & previous advisory pairs
k = 0
for *(*$i = 0;\ i < N;\ i = i + 1$*)* **do**
    output_{NN}[i] = Reach(input_{NN}[a_{prev}[i,1]] , a_{prev}[i,0]) // Compute reach set of neural network
    advise[i] = argminNN(output_{NN}[i]) // Compute advisory based on minimum value of output reach set output_{NN}
    for *adv in advise[i]* **do**
        adv_{NN} [k] = [adv, i] // Track advisories
        k++
    end
end
return: adv_{NN}

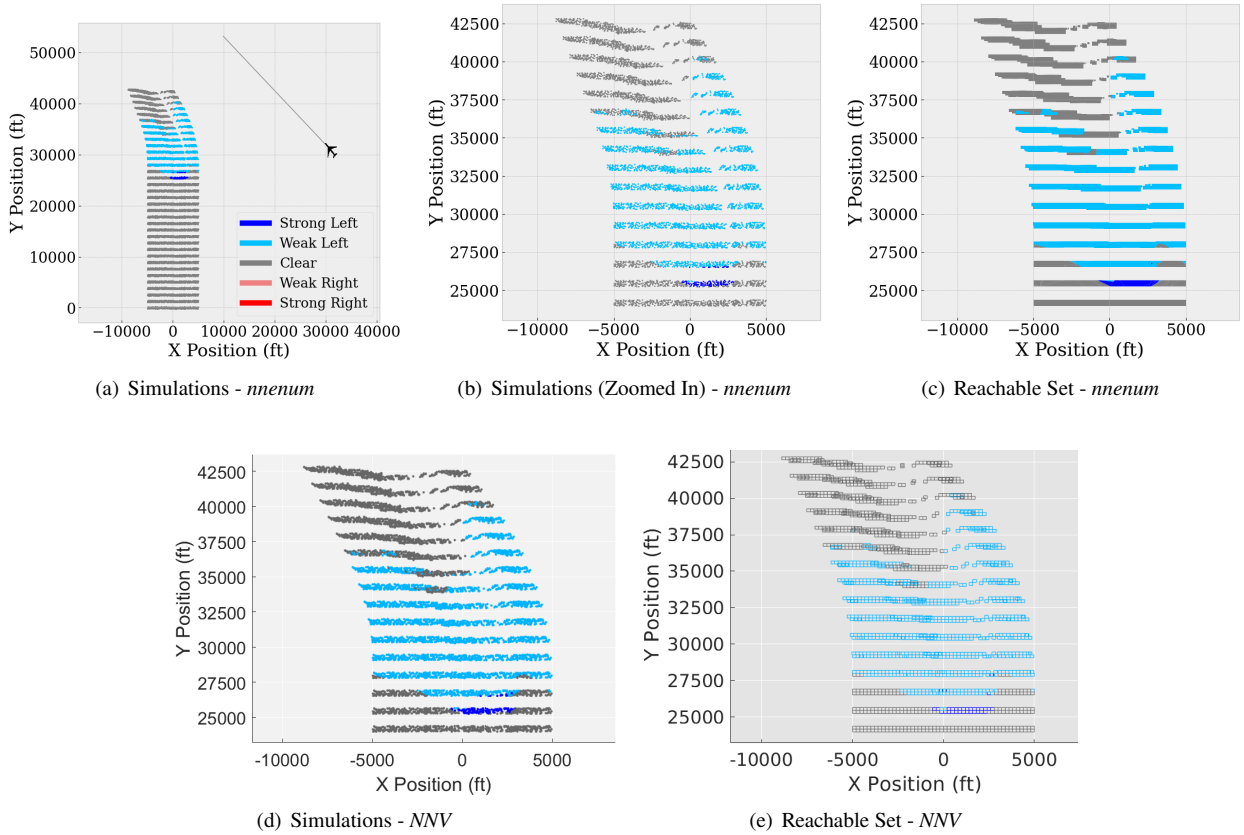
such as using quick bounds estimates using zonotopes with domain contraction to improve accuracy. This is further combined with an abstraction-refinement approach [32], where star sets first analyze the system with overapproximation, and then refine by splitting sets on individual neurons only if it is needed to prove the safety property.

In this work, first it is analyzed a set of states using the quicker overapproximation mode of nnenum to check whenever multiple advisories are possible. Otherwise, exact analysis is used, which splits the set of states into many smaller states, each with a unique classification. For the plant dynamics, rather than using CORA and the nonlinear differential equations in Equation 3, nnenum instead performs numerical simulation within each set using the original Dubin's car dynamics in Equation 1 and observations from Equation 2, in order to produce a local numerical linearization of the observed state variables. Since reachability analysis through linear differential equations is more efficient than reachability with nonlinear equations, the method has better expected scalability. This result is demonstrated in our evaluation section next, where larger uncertainties in the initial state can be checked, with accuracy that visually resembles simulations of the closed-loop system.

VI. Results

The 10 closed loop test cases depicted on Fig. 6 are evaluated using NNV and nnenum under specified initial uncertainty. For each scenario, an initial ownship uncertainty of ±5000 in the x direction and ±200 in the y direction is used when evaluated. Although there are some small differences in 2 out of the 10 cases evaluated between NNV and nnennum ($\varphi_{2_{CL}}$ and $\varphi_{10_{CL}}$), both tools are able to succesfully verify the safety of the ownship aircraft with respect to the intruder in all 10 cases. These results are depicted in ten figures, which consist of 5 subplots, organized as follows: subplots a, b and c correspond to nnenum results and subfigures d and e to NNV. For each tool there are zoomed in (b and d) simulations results as well as the reachable sets (c and e), while a depicts the complete simulation trajectories of

the ownship in nnenum.



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

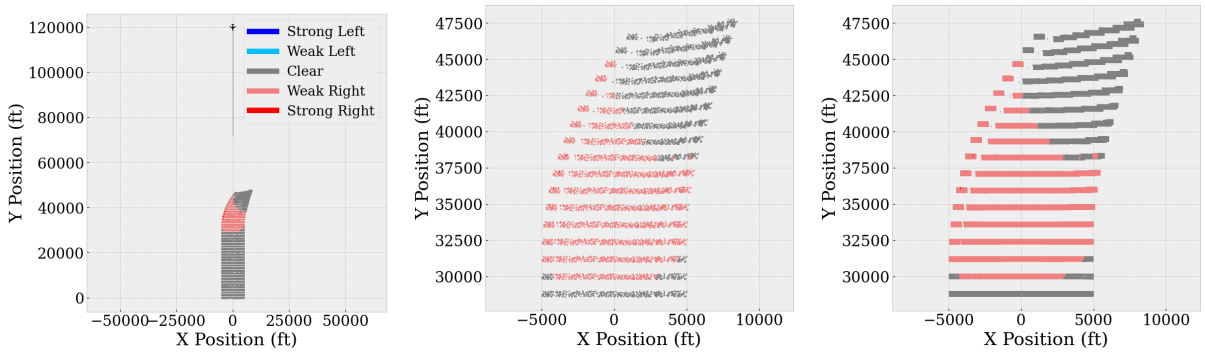
Fig. 7 Right Closing $\varphi_{8_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

To illustrate and explain the results from this investigation, 2 out of the 10 test cases are visualized in this section: Right Closing $\varphi_{8_{CL}}$ (Fig. 7) and Head On Collision $\varphi_{10_{CL}}$ (Fig. 8). The remaining results are found in Figs. 9 to 16 in the Appendix. This set of results can be summarized as the ownship is always safe, maintaining a much greater distance than the safety distance specified in the NMAC 1. This is accomplished by turning to the opposite side from which the intruder is approaching as depicted in Figs. 7 and 9, and Figs. 11 to 15. In the other 3 scenarios, the ownship is biased to deviate towards the right in order to avoid a collision with the intruder when both are flying in the same path but opposite direction or same direction and different speed. This bias is very evident in Fig. 8, and it is also observed in Fig. 16, in which the ownship reachable sets split into two main paths, one towards the rights, and one towards the left.

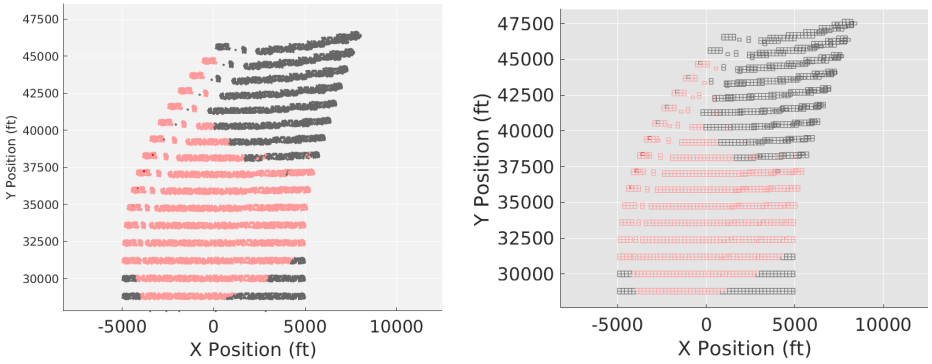
If the ownship is initially located between approx. -2000 ft and +5000 in the x-axis, the ownship will deviate to the right. If initially located to the left of -2000 ft in the x-axis, then the ownship will avoid the collision by turning to the left. The final scenario, Intruder Tail Chase $\varphi_{2_{CL}}$ does not present this bias and behaves more as initially predicted. This result is demonstrated in Fig. 10 by both NNV and nnenum, although the tools present slightly different reachable set trajectories. The initial state set is split into two halves at x = 0 ft in both simulations and reachable sets. If the

ownship is initially located in the positive x-axis, it will turn to the right and vice versa.

The first test case, Right Closing $\varphi_{8_{CL}}$, is illustrated in Fig. 7. It is observed in the simulation plots that when the ownship approaches the intruder's trajectory at approximately y = 25000 ft, the controllers start issuing weak and strong left commands to the ownship to maintain a safe distance with the intruder. These decisions are specially evident in the zoomed in and reach set plots, where the trajectories turn towards the left for a few steps, followed by clear of conflict commands as the ownship safety is maintained. On the other scenario in Fig. 8, the Head On Collision $\varphi_{10_{CL}}$ test case demonstrates the ownship's safety when the ownship faces another aircraft in the same path but in the opposite direction. Both tools show very similar results as the ownship deviates to the right when flying at approximately y = 30000 ft, issuing only weak right commands to maintain safety with respect to the intruder.



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*



(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 8 Head On Collision $\varphi_{10_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

The similarity between the simulation and verification results observed in these plots is an indication of the reduced over-approximation errors computed by both tools using star-set methods. Despite both tools successfully verifying the safety of the ownship with respect to the intruder aircraft in all 10 cases, there are some main differences between nnenum and NNV, discussed in the next section.

VII. Discussion

The results in Section VI and in Appendix I demonstrate the capabilities of nnenum and NNV for the verification of real-world unmanned CPSs, where it can also be observed some of the main differences between these tools. NNV needs to compute an initial partition on the initial set and then compute the reachable sets for each partition, while nnenum is able to compute these partitions as needed at each control step, improving its efficiency with respect to NNV. Another main difference is the nonlinear ODE reachability analysis, where NNV is also capable of computing and displaying the complete continuous-time reachable sets of NNCS, but for consistency purposes, only the reachable sets of the ownship at each control step are shown for ease of comparison against nnenum. The results also show some of the complexities of deploying and verifying these complex systems. In 8 out of 10 cases, NNV and nnenum obtain the same results, but there are two cases in which there are differences in both the simulations and the reachability analysis. There is a specific step in each scenario in which NNV and nnenum do not issue the same advisory commands. In test case $\varphi_{10_{CL}}$ (Fig. 8), the main difference is in the first reachable set shown, in which nnenum only selects COC commands while NNV also has some weak right commands towards the center of the reachable set. The other test case, $\varphi_{2_{CL}}$ (Fig. 10) seems similar to the previous test case in the sense that there is one main step in the reachable set computation that differs between NNV and nnenum (step 5) in both sides of the path, which leads to different final paths. To complement the reachability analysis presented and try to better understand the differences between NNV and nnenum, the simulation differences between NNV and nnenum have been analyzed across 1000 randomly initialized simulations and the results are introduced in Table 5.

It is observed that even in the test cases in which NNV and nnenum return the same results, there are still some minor differences between the two. Some possible sources of error are the different solvers used for the computation of the states of the plant, or a slightly different implementation on the plant model, among others. NNV computes the reachability analysis of the plant model using all 9 state variables described in Eq. 3, but nnenum makes use of only the first 6 variables, and then uses those to compute the remaining 3 (7 to 9) a posteriori. These differences in computation may lead to small computation differences between the two, which accumulate through time and eventually lead to slightly different final paths, although both tools successfully verify the safety of the ownship in every case. Another main difference between NNV and nnenum is the different reachability schemes utilized, which lead to significant differences in the computation time. In average, nnenum is able to compute the reachable set of each closed loop property in 172 seconds, while NNV completes each test case in 140819 seconds, approximately 39 hours. The main reasons for the difference in computation time are the different partitioning routines and the plant reachability techniques. When compared to other validation techniques such as Monte Carlo simulations, these reachability techniques can prove and guarantee the safety of the ownship under the whole initial uncertainty of the ownship aircraft, while Monte Carlo simulations can only provide probable guarantees, determined by the number of simulations run. In terms of computation time, nnenum is equivalent to running 0.13 simulations per sqft of the initial set of the ownship, while NNV

Table 5 Simulation differences between NNV and nnenum across all 10 closed-loop properties. For each experiment, 1000 random simulations were sampled. x_{own} and y_{own} values are in ft and ψ_{own} in rad, and they correspond to the average values of the absolute difference between the recorded variables in NNV and nnenum.

		Closed-Loop Properties									
		$\varphi 1_{CL}$	$\varphi 2_{CL}$	$\varphi 3_{CL}$	$\varphi 4_{CL}$	$\varphi 5_{CL}$	$\varphi 6_{CL}$	$\varphi 7_{CL}$	$\varphi 8_{CL}$	$\varphi 9_{CL}$	$\varphi 10_{CL}$
$\mathbf{x}_{own}$	*mean*	0.24	243.29	9.9*e*-11	7.4*e*-10	5.6*e*-11	1.4*e*-09	2.3*e*-9	2.1*e*-9	9.5*e*-10	71.86
	std	0.039	355.89	1.5*e*-10	5.6*e*-10	2.0*e*-10	1.5*e*-9	3.3*e*-9	2.8*e*-9	1.0*e*-9	105.41
	max	0.95	1.1*e*+3	3.7*e*-10	1.3*e*-9	8.8*e*-10	3.7*e*-9	8.9*e*-9	6.2*e*-9	2.3*e*-9	270.77
$\mathbf{y}_{own}$	*mean*	0.0562	51.58	4.8*e*-11	1.9*e*-10	1.3*e*-9	2.9*e*-10	7.3*e*-10	1.1*e*-10	1.2*e*-10	14.54
	std	0.098	75.94	7.7*e*-11	1.7*e*-10	1.5*e*-9	3.8*e*-10	1.3*e*-9	1.5*e*-10	1.3*e*-10	24.42
	max	0.26	241.61	2.7*e*-10	4.2*e*-10	4.7*e*-9	1.0*e*-9	3.9*e*-9	4.6*e*-10	3.4*e*-10	68.43
ψ_{own}	*mean*	1.2*e*-5	0.039	8.3*e*-16	6.3*e*-15	3.2*e*-16	2.6*e*-15	1.6*e*-15	2.0*e*-15	2.3*e*-15	0.0069
	std	3.2*e*-5	0.048	1.6*e*-15	4.9*e*-15	1.1*e*-15	2.8*e*-15	2.6*e*-15	3.0*e*-15	2.6*e*-15	0.011
	max	1.0*e*-4	0.11	5.7*e*-15	1.2*e*-14	4.9*e*-15	8.6*e*-15	8.5*e*-15	7.7*e*-15	5.7*e*-15	0.025

is equivalent to running 114 simulations per sqft. In this aspect, nnenum is preferable over Monte Carlo simulations and NNV due to its efficient computation of the reachable sets of this ACAS Xu closed-loop benchmark.

VIII. Conclusion

This paper introduced a set of 10 new closed loop verification properties and developed and applied a closed loop verification approach to verify both the output of NNCS and the switching behavior between neural network controllers. The approach was demonstrated using the NNV and nnenum tools on 5 of 45 ACAS Xu neural networks with switching between all five corresponding to co-altitude collision cases. Both tools show reachable states of an ownship aircraft with modified Dubins dynamics on a collision course with an intruder aircraft. The reachability computation considers initial state uncertainty and ownship control inputs provided by 5 switched NNs. The verification objective was to show that the switch NNCS resulted in collision free courses, even a large initial uncertainty. This study showed that it is possible to do a comprehensive safety verification analysis of a complex air collision advisory system for aircraft travelling in straight, co-altitude paths, but further research is needed to determine if climbing or descending flights can also be proven safe using these tools. In the tool demonstration, nnenum showed a tighter and faster computation of the over-approximation of its reachability methods, as observed by the similarity between the simulation and reachable plots, while NNV is able to present the complete continuous-time reachable sets of the ownship trajectories. A summary of the main outcomes of our methods can be summarized as follows:

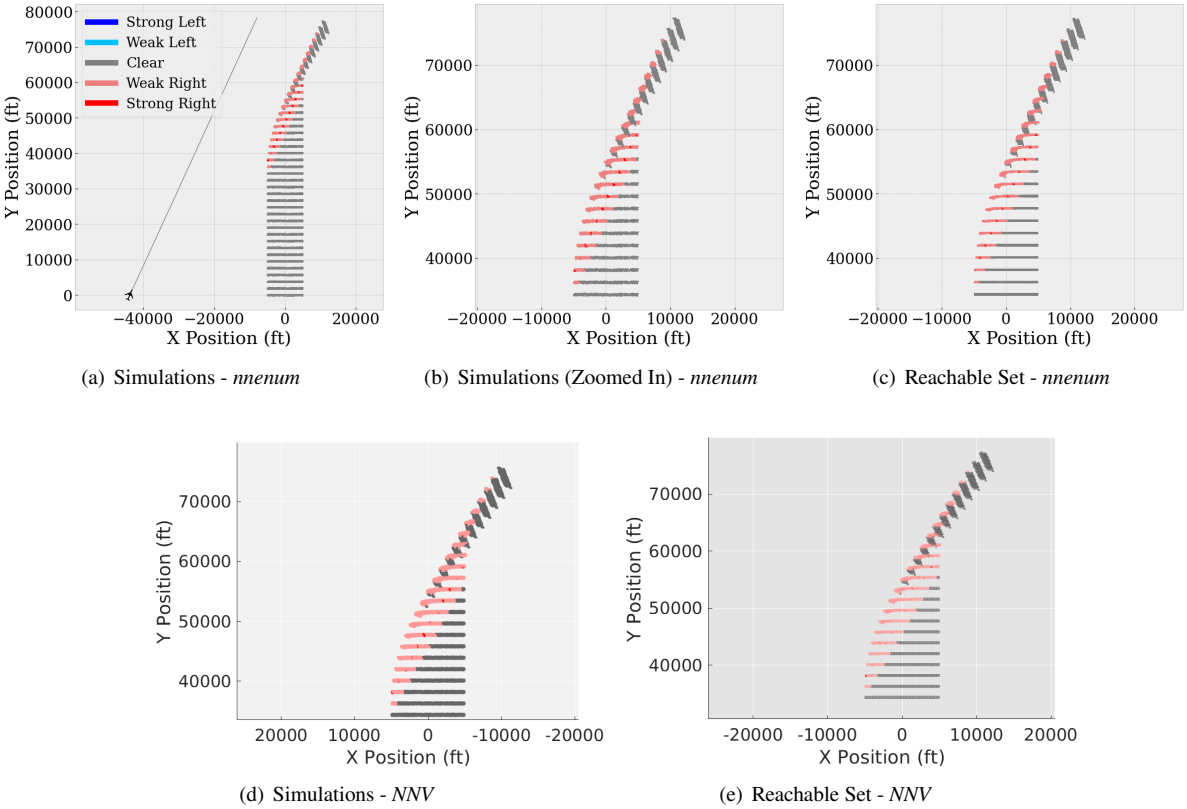
- For an initial assessment, a small amount of Monte Carlo simulations may be sufficient, in combination of other testing routines, to check the proper behavior of the CPS and its implementation.
- In safety-critical applications, Monte Carlo simulations cannot provide formal guarantees, which is vital for safety-critical systems such as autonomous aircraft. In this case, formal verification tools such as nnenum and

NNV would be needed.

- NNV and nnenum can both provide formal guarantees on the safety of the aircraft using reachability analysis for neural networks as well as nonlinear ODEs.
- nnenum has proven to compute the reachable sets of the NNCS much faster than NNV, about 4 orders of magnitude faster, largely due to ts efficient partitioning scheme and the nonlinear ODE reachability method.

Appendix I: Closed Loop Verification Results

The ACAS Xu NN compression closed loop verification benchmark presents a total of 10 properties described in Section IV, specified in Table 4 and illustrated in Fig. 6. In section VI, a summary of the verification results for all ten properties was presented, but only properties $\varphi_{8_{CL}}$ and $\varphi_{10_{CL}}$ were depicted. In this Appendix, the remainder of the verification results are illustrated in Figs. 9 to 16.



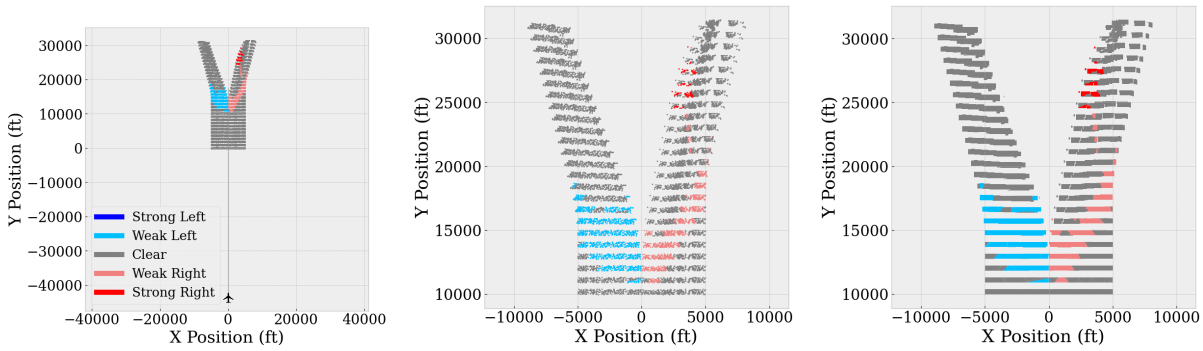
(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

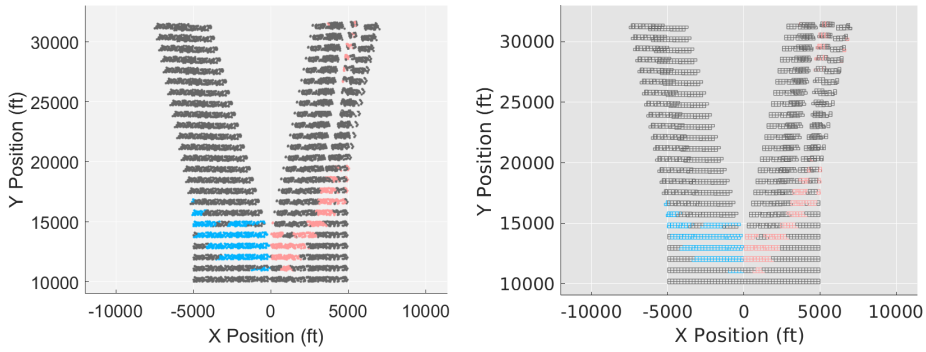
Fig. 9 Left Abeam $\varphi_{1_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

Appendix II: 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



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*



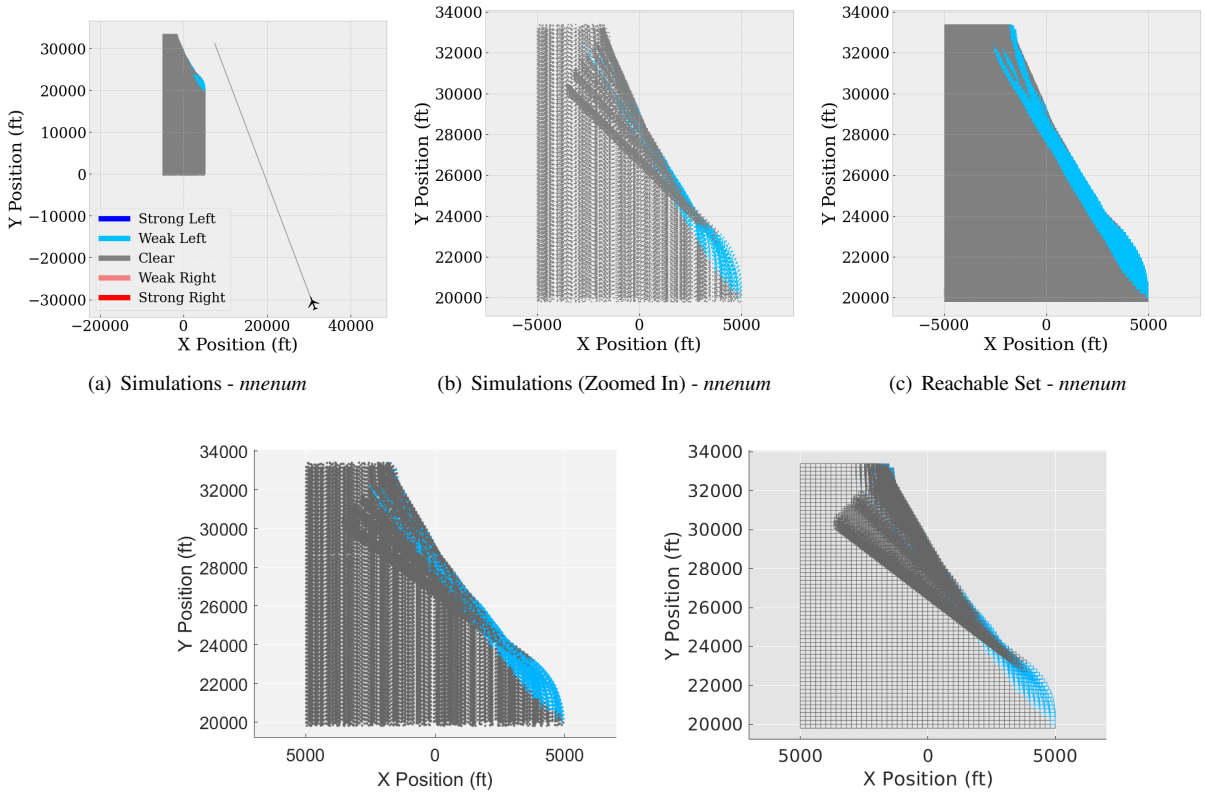
(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 10 Intruder Tail Chase $\varphi_{2_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

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 6 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



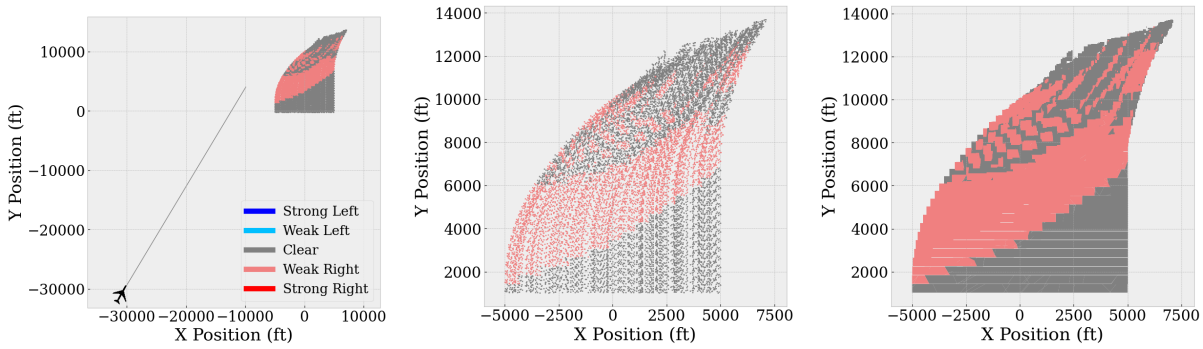
(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

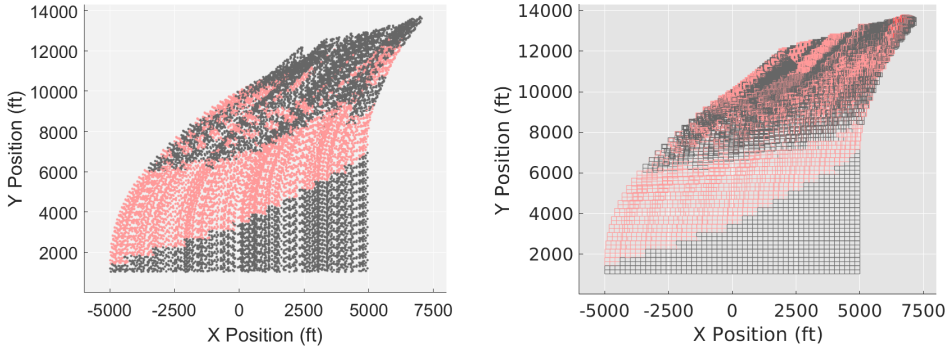
Fig. 11 Right Gaining $\varphi_{3_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

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 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



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*



(d) Reachable Set - *NNV* (e) Reachable Set - *NNV*

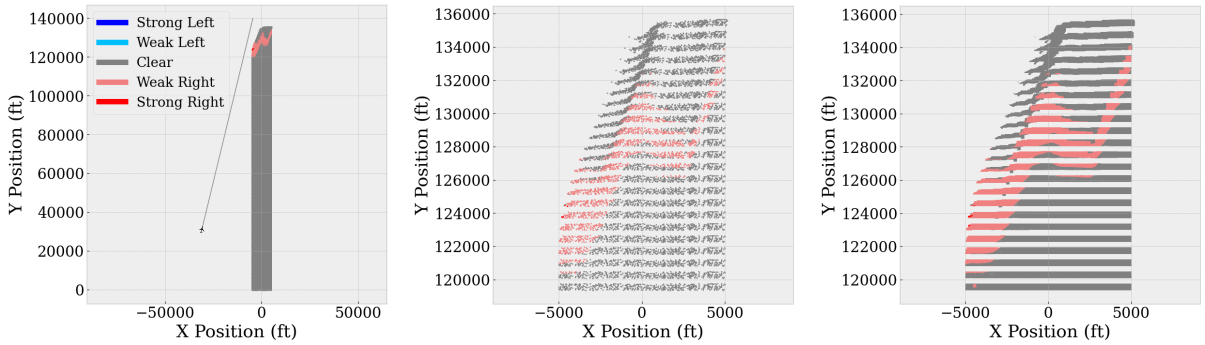
Fig. 12 Left Gaining $\varphi_{4_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

than 9 seconds.

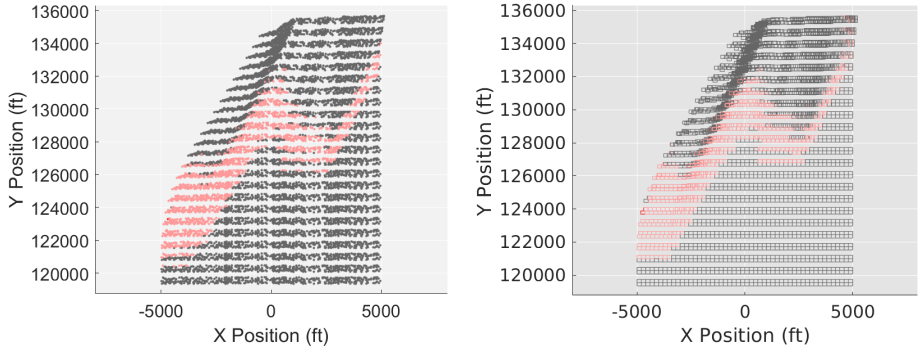
The remaining 6 properties check a wide range of conditions. Property φ_5 checks that the neural network favors the strong right advisory if the intruder is passing very close to the ownship's left. Property φ_6 checks that the neural network outputs clear of conflict if the intruder is sufficiently 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 (τ = 100s) 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 (AFOSR) under award numbers FA9550-19-1-0288, FA9550-18-1-0122, FA9550-22-1-0019, FA9550-21-1-0121, and FA9550-22-1-0450, the National Science Foundation (NSF) EPSCoR First Award and under grant numbers FMitF 1918450, 1910017and



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*



(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 13 Left Closing $\varphi_{5_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

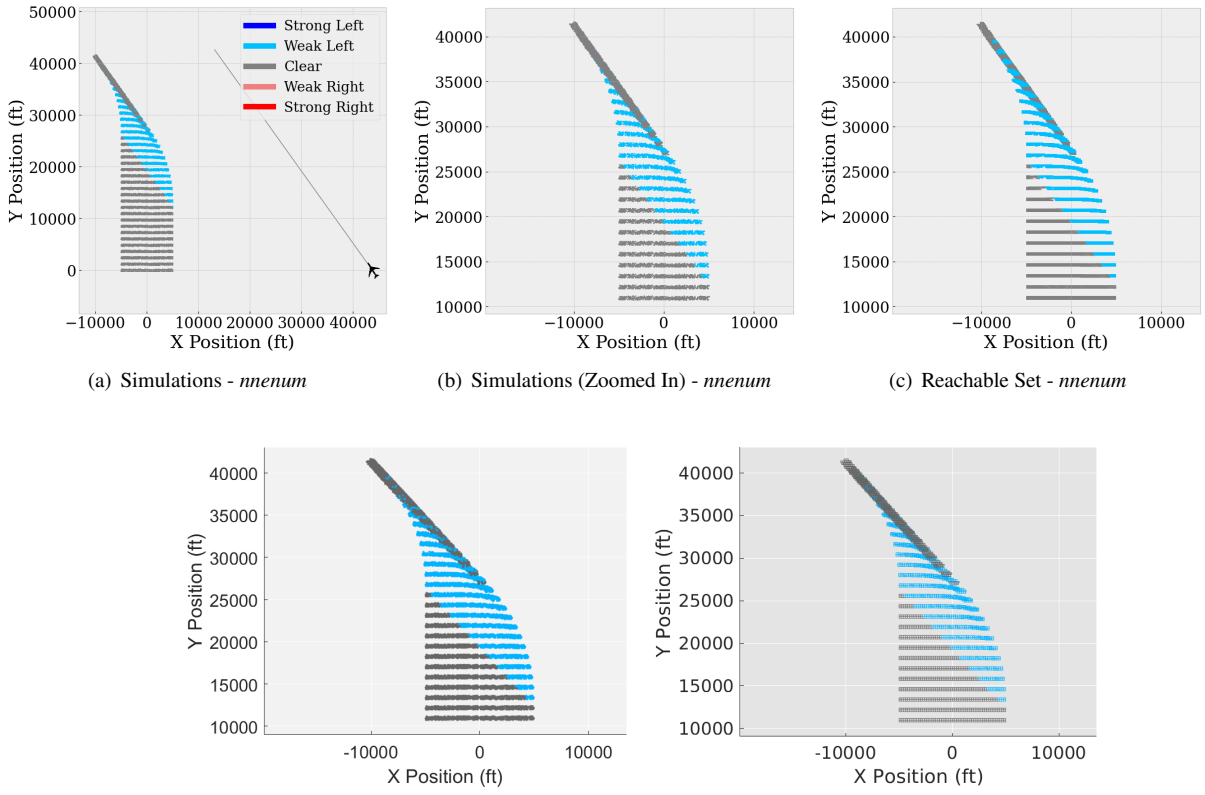
2028001, the Defense Advanced Research Projects Agency (DARPA) Assured Autonomy program through contract number FA8750-18-C-0089, and the Office of Naval Research (ONR) under award number N00014-22-1-2156. Any opinions, finding, and conclusions or recommendations expressed in this material are those 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, NSF or ONR.

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," *Institute of Electrical and Electronics Engineers (IEEE) Journal of Selected Topics in Signal Processing*, Vol. 5, No. 3, 2011, pp. 606–617. doi:10.1109/JSTSP.2011.2139193.

[3] Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D., "Advanced deep-learning techniques for salient and category-specific object



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 14 Right Abeam $\varphi_{6_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

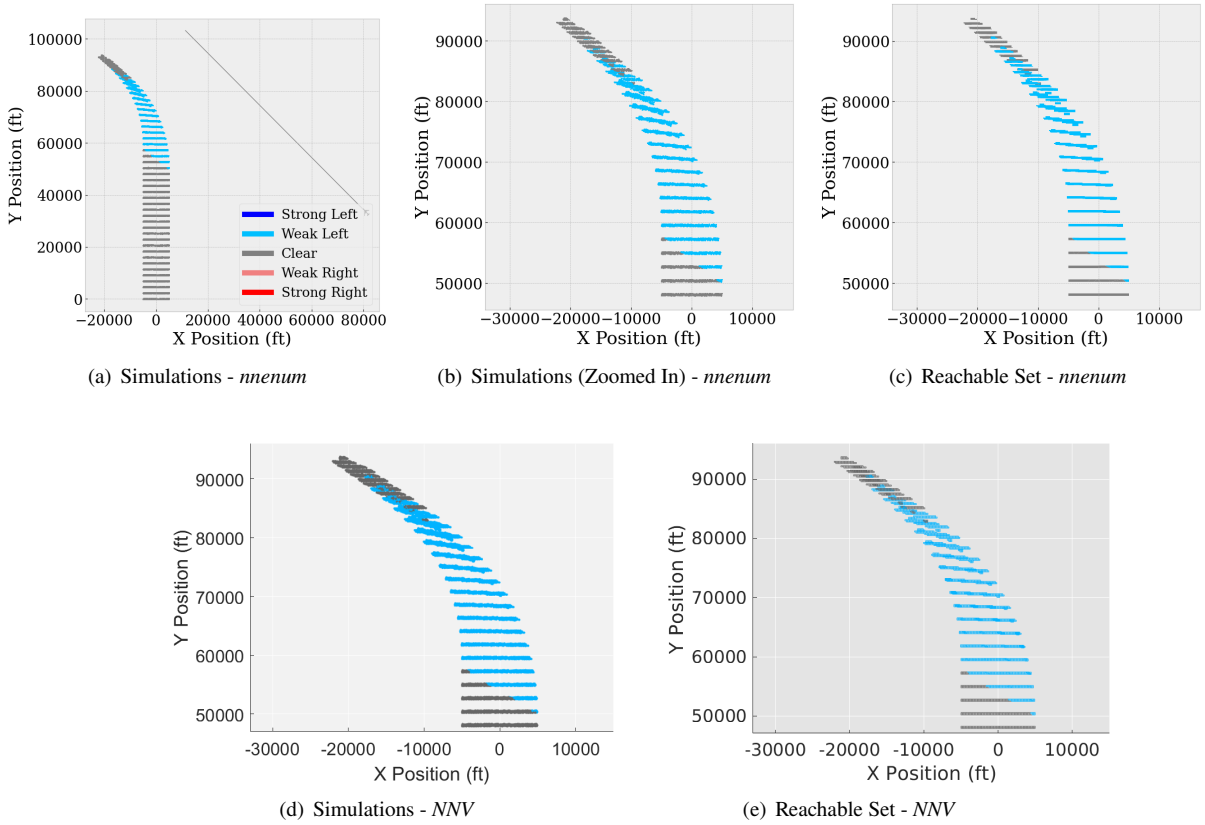
detection: a survey," *Institute of Electrical and Electronics Engineers (IEEE) Signal Processing Magazine*, Vol. 35, No. 1, 2018, pp. 84–100. doi:10.1109/MSP.2017.2749125.

[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. doi:10.1038/nature24270.

[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. doi:10.1038/s41586-019-1724-z.

[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)*, Institute of Electrical and Electronics Engineers (IEEE), 2016, pp. 1–10. doi:10.1109/DASC.2016.7778091.



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 15 Right Isosceles $\varphi_{7_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

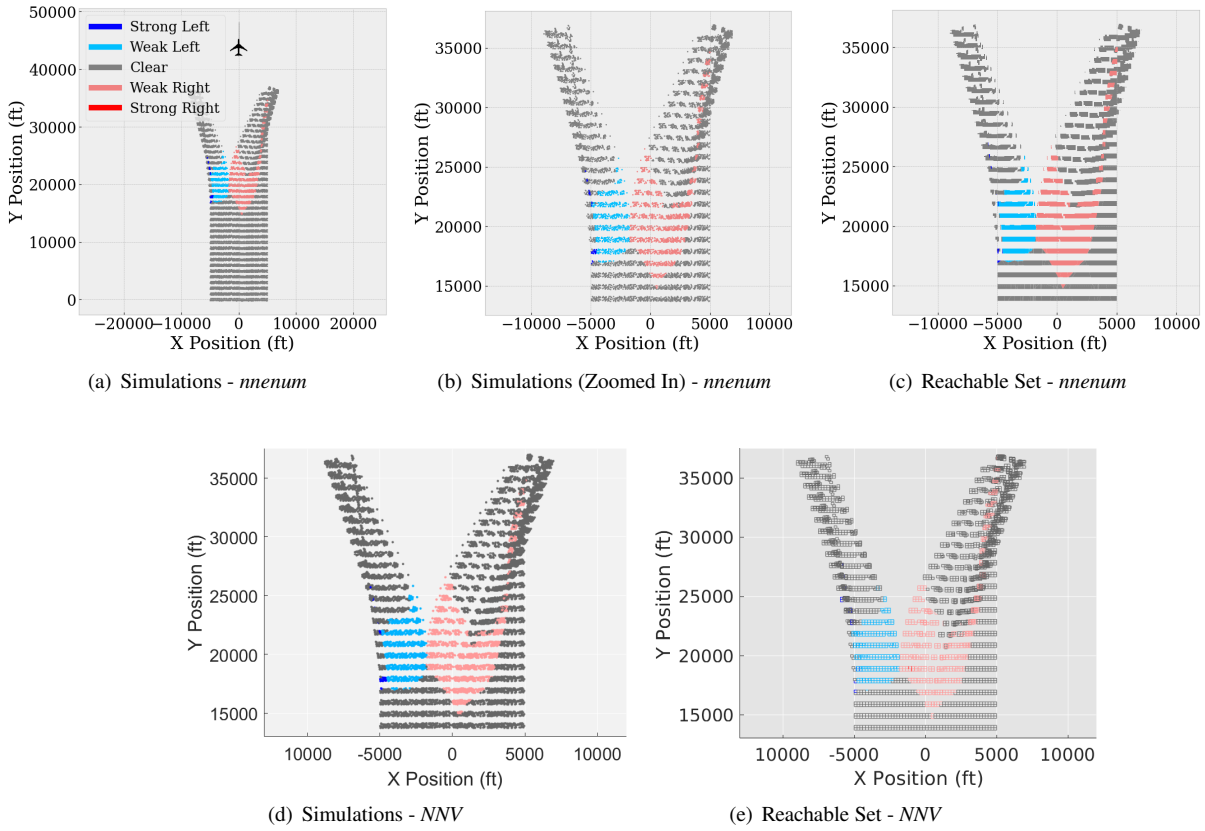
[8] Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., and Kochenderfer, M. J., "Algorithms for verifying deep neural networks," *Foundations and Trends in Optimization*, Vol. 4, No. 3–4, 2021, pp. 244–404. doi:10.1561/2400000035.

[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.

[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 Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control*, Association for Computing Machinery (ACM), New York, NY, USA, 2019, pp. 169–178. doi: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 Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control*, Association for Computing Machinery (ACM), New York, NY, USA, 2019, pp. 157–168. doi:10.1145/3302504.3311807.

[12] Sun, X., Khedr, H., and Shoukry, Y., "Formal Verification of Neural Network Controlled Autonomous Systems," *Proceedings*



(a) Simulations - *nnenum* (b) Simulations (Zoomed In) - *nnenum* (c) Reachable Set - *nnenum*

(d) Simulations - *NNV* (e) Reachable Set - *NNV*

Fig. 16 Ownship Tail Chase $\varphi_{9_{CL}}$ with initial ownship uncertainty x_0, y_0 = [±5000, ±200].

of the 22nd Association for Computing Machinery (ACM) International Conference on Hybrid Systems: Computation and Control, Association for Computing Machinery, New York, NY, USA, 2019, p. 147–156. doi:10.1145/3302504.3311802.

[13] Xiang, W., Manzanas Lopez, D., Musau, P., and Johnson, T. T., "Reachable Set Estimation and Verification for Neural Network Models of Nonlinear Dynamic Systems," *Safe, Autonomous and Intelligent Vehicles*, edited by H. Yu, X. Li, R. M. Murray, S. Ramesh, and C. J. Tomlin, Springer International Publishing, 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*, Institute of Electrical and Electronics Engineers (IEEE), 2018. doi:10.23919/ACC.2018.8431048.

[15] Fan, J., Huang, C., Chen, X., Li, W., and Zhu, Q., "ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems," *Automated Technology for Verification and Analysis*, edited by D. V. Hung and O. Sokolsky, Springer International Publishing, Cham, 2020, pp. 537–542. doi:10.1007/978-3-030-59152-6_30.

[16] 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)*, Association for Computing Machinery (ACM), Auckland, New Zealand, 2020. doi:10.1007/s10458-021-09529-3.

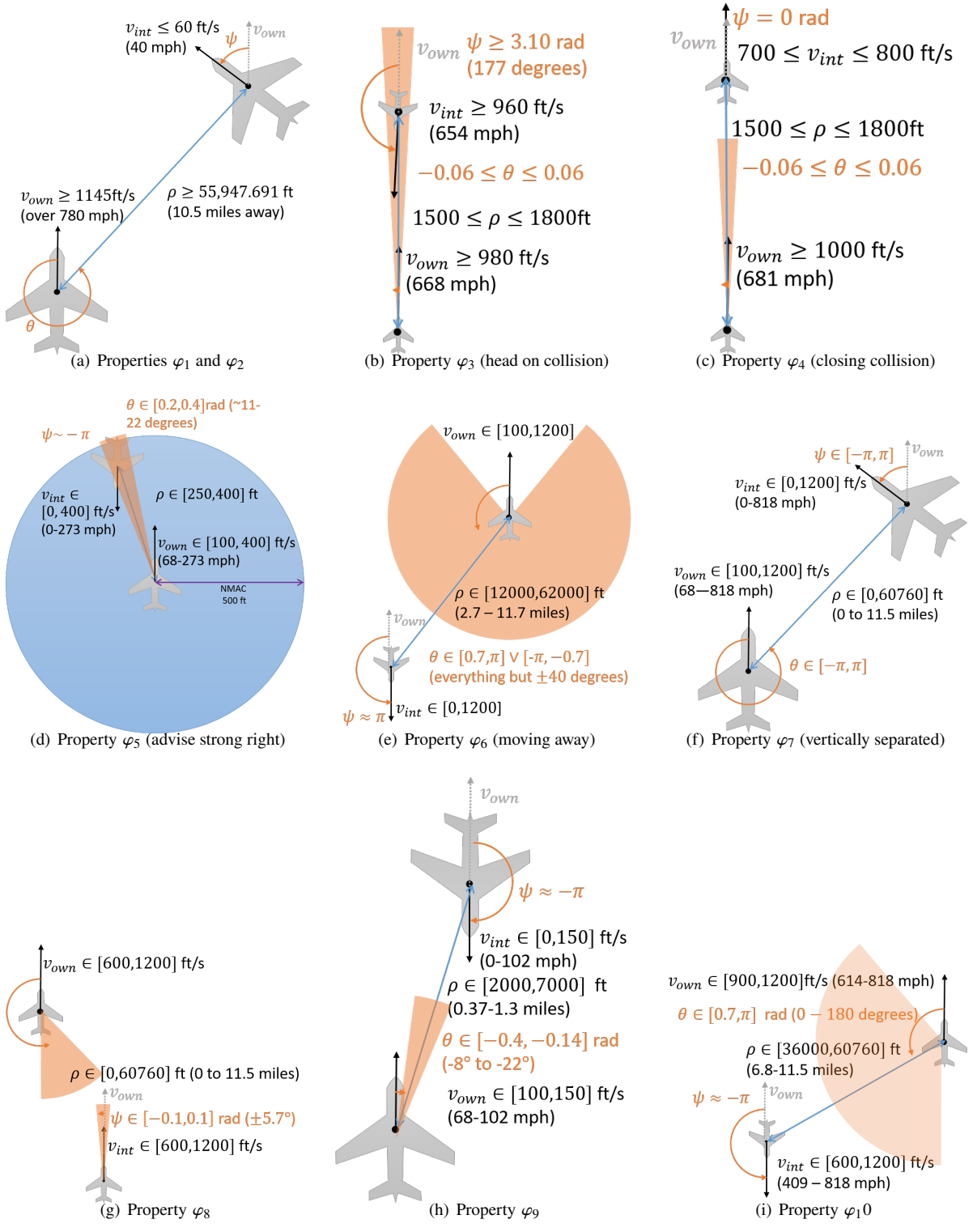


Fig. 17 Depiction of the aircraft encounter geometry for the open loop ACAS Xu verification properties

[17] Akintunde, M. E., Botoeva, E., Kouvaros, P., and Lomuscio, A., "Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems," *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, 2020, pp. 22–32. doi:10.24963/kr.2020/3.

[18] Tran, H.-D., Cai, F., Diego, M. L., Musau, P., Johnson, T. T., and Koutsoukos, X., "Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control," *Association for Computing Machinery (ACM) Trans. Embed. Comput. Syst.*, Vol. 18, Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3358230.

[19] Tran, H.-D., Yang, X., Manzanas Lopez, D., 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," *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I*, Springer-Verlag, Berlin, Heidelberg, 2020, p. 3–17. doi:10.1007/978-3-030-53288-8_1.

[20] Manzanas Lopez, D., Musau, P., Hamilton, N., Tran, H.-D., and Jonhson, T. T., "Case Study: Safety Verification of an Unmanned Underwater Vehicle," *2020 Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy Workshops (SPW)*, 2020, pp. 189–195. doi:10.1109/SPW50608.2020.00047.

[21] Irfan, A., Julian, K. D., Wu, H., Barrett, C., Kochenderfer, M. J., Meng, B., and Lopez, J., "Towards Verification of Neural Networks for Small Unmanned Aircraft Collision Avoidance," *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 2020, pp. 1–10. doi:10.1109/DASC50938.2020.9256616.

[22] Clavière, A., Asselin, E., Garion, C., and Pagetti, C., "Safety Verification of Neural Network Controlled Systems," *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2021, pp. 47–54. doi:10.1109/DSN-W52860.2021.00019.

[23] Bak, S., Liu, C., and Johnson, T. T., "The Second International Verification of Neural Networks Competition (VNN-COMP 2021): Summary and Results," *CoRR*, Vol. abs/2109.00498, 2021.

[24] Johnson, T. T., Lopez, D. M., Benet, L., Forets, M., Guadalupe, S., Schilling, C., Ivanov, R., Carpenter, T. J., Weimer, J., and Lee, I., "ARCH-COMP21 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants," *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, EPiC Series in Computing, Vol. 80, edited by G. Frehse and M. Althoff, EasyChair, 2021, pp. 90–119. doi:10.29007/kfk9.

[25] 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," *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, Association for Computing Machinery, New York, NY, USA, 2020. doi:10.1145/3365365.3382216.

[26] 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.

[27] Julian, K. D., Sharma, S., Jeannin, J.-B., and Kochenderfer, M. J., "Verifying aircraft collision avoidance neural networks through linear approximations of safe regions," *AIAA Spring Symposium*, 2019.

[28] Julian, K. D., and Kochenderfer, M. J., "Guaranteeing safety for neural network-based aircraft collision avoidance systems," *IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019. doi:10.1109/DASC43569.2019.9081748.

[29] Julian, K. D., and Kochenderfer, M. J., "Reachability Analysis for Neural Network Aircraft Collision Avoidance Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 6, 2021, pp. 1132–1142. doi:10.2514/1.G005233.

[30] Alvarez, L. E., Jessen, I., Owen, M. P., Silbermann, J., and Wood, P., "ACAS sXu: Robust decentralized detect and avoid for small unmanned aircraft systems," *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, Institute of Electrical and Electronics Engineers (IEEE), 2019, pp. 1–9. doi:10.1109/DASC43569.2019.9081631.

[31] Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S., "Formal Security Analysis of Neural Networks Using Symbolic Intervals," *Proceedings of the 27th USENIX Conference on Security Symposium*, USENIX Association, USA, 2018, p. 1599–1614.

[32] Bak, S., "nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement," *NASA Formal Methods Symposium*, Springer, 2021, pp. 19–36.

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

[34] Kochenderfer, M. J., Amato, C., Chowdhary, G., How, J. P., Reynolds, H. J. D., Thornton, J. R., Torres-Carrasquillo, P. A., Ure, N. K., and Vian, J., *Optimized airborne collision avoidance*, MIT press, 2015, Chap. Decision Making Under Uncertainty: Theory and Application, pp. 249–276.

[35] 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. doi:10.1007/978-3-319-63387-9_5.

[36] Marston, M., and Baca, G., "ACAS-Xu initial self-separation flight tests," Tech. rep., NASA, March 2015.

[37] 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. doi:10.2514/atcq.21.3.275.

[38] 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.

[39] 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. doi:10.2514/1.G003724.

[40] Julier, S., and Uhlmann, J., "Unscented filtering and nonlinear estimation," *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)*, Vol. 92, No. 3, 2004, pp. 401–422. doi:10.1109/JPROC.2003.823141.

[41] Kochenderfer, M. J., and Monath, N., "Compression of Optimal Value Functions for Markov Decision Processes," *2013 Data Compression Conference*, 2013, pp. 501–501. doi:10.1109/DCC.2013.81.

[42] Raychaudhuri, S., "Introduction to Monte Carlo Simulation," *2008 Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), 2008, pp. 91–100. doi:10.1109/WSC.2008.4736059.

[43] Antony, J., *Design of Experiments for Engineers and Scientists*, Elsevier, 2014.

[44] Viana, F. A., "A Tutorial on Latin Hypercube Design of Experiments," *Quality and Reliability Engineering International*, Vol. 32, No. 5, 2016, pp. 1975–1985. doi:10.1002/qre.1924.

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

[46] Hainry, E., "Reachability in Linear Dynamical Systems," *Logic and Theory of Algorithms*, edited by A. Beckmann, C. Dimitracopoulos, and B. Löwe, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 241–250. doi:10.1007/978-3-540-69407-6_28.

[47] Ruan, W., Huang, X., and Kwiatkowska, M., "Reachability analysis of deep neural networks with provable guarantees," *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, edited by J. Lang, International Joint Conferences on Artificial Intelligence, 2018, pp. 2651–2659. doi:10.24963/ijcai.2018/368.

[48] Bak, S., and Duggirala, P. S., "Simulation-Equivalent Reachability of Large Linear Systems with Inputs," *Computer Aided Verification*, edited by R. Majumdar and V. Kunčak, Springer International Publishing, Cham, 2017, pp. 401–420. doi:10.1007/978-3-319-63387-9_20.

[49] Tran, H.-D., Manzanas Lopez, D., Musau, P., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., "Star-Based Reachability Analysis of Deep Neural Networks," *Formal Methods – The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings*, Springer-Verlag, Berlin, Heidelberg, 2019, p. 670–686. doi:10.1007/978-3-030-30942-8_39.

[50] Althoff, M., "An Introduction to CORA 2015," *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, EPiC Series in Computing, Vol. 34, edited by G. Frehse and M. Althoff, EasyChair, 2015, pp. 120–151. doi:10.29007/zbkv.

[51] 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.