Verifying Controllers with Vision-based Perception Using Safe Approximate Abstractions

Chiao Hsieh, *Graduate Student Member, IEEE*, Yangge Li, Dawei Sun, *Graduate Student Member, IEEE*, Keyur Joshi,
Sasa Misailovic, Sayan Mitra, *Senior Member, IEEE*

***Abstract*—Fully formal verification of perception models is likely to remain challenging in the foreseeable future, and yet these models are being integrated into safety-critical control systems. We present a practical method for reasoning about the safety of such systems. Our method is based on systematically constructing approximations of perception models from system-level safety requirements, data, and program analysis of the modules that are downstream from perception. These approximations have some desirable properties like being low-dimensional, intelligible, and tractable. The closed-loop system, with the approximation substituting the actual perception model, is verified to be safe. Establishing formal relationship between the actual and the approximate perception models remains well beyond available verification techniques. However, we do provide a useful empirical measure of their closeness called *precision*. Overall, our method can trade off the size of the approximation against precision. We apply the method to two significant case studies (a) a vision-based lane tracking controller for an autonomous vehicle and (b) a controller for an agricultural robot. We show how the generated approximations for each system can be composed with the downstream modules and be verified using program analysis tools like CBMC. Detailed evaluations of the impacts of size, and the environmental parameters (e.g., lighting, road surface, plant type) on the precision of the generated approximations suggest that the approach can be useful for realistic control systems.**

***Index Terms*—Abstraction, autonomous systems, formal verification, vision-based control.**

I. Introduction

COMPUTER vision algorithms, and in particular deep neural network (DNN) models, are now indispensable in autonomous systems, but research on rigorous safety analysis of vision-based control systems is sparse. Motivated by safety criticality of autonomous systems, the problem of verifying neural networks has received keen attention (see [1]–[6], and the references therein[1]). In the meantime, techniques and limitations of *system-level safety* remain poorly understood. Regulatory agencies from many industries – aerospace [7], automotive [8], robotic surgery, and manufacturing – are creating processes and guidelines, and a small number of recent research papers are starting to tackle this problem [9]–[14] (see Section II).

Manuscript received April 07, 2022; revised June 11, 2022; accepted July 05, 2022. This article was presented at the International Conference on Embedded Software (EMSOFT) 2022 and appeared as part of the ESWEEK-TCAD special issue. This work was supported in part by NSF under Grant NSF-SHF-2008883, USDA/NIFA under Grant 2021-67021-33449, and the Boeing Company.

The authors are with the University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (email: chsieh16@illinois.edu)

[1]The neural network verification competition [6] had twelve research teams.

Investigating verification of vision-based control systems is interesting from two perspectives: (1) Even though writing formal requirements for CNN perception models may be ill-posed [15], safety requirements for autonomous systems using CNN models are usually fairly obvious. A "lane" may be difficult to specify in terms of pixel intensity thresholds, but the safety requirements of a lane keeping control system are less mysterious. (2) It is well-known that CNNs have fragile decision boundaries and are susceptible to adversarial inputs [16]. Since the existing NN verification tools verify properties around a small neighborhood of the input space, the presence of adversarial inputs makes the NN verification results conservative. On the other hand, system-level safety analysis deals with the temporal evolution of the whole system (including the NNs), and therefore, could benefit from the smoothness of natural signals, and be more robust to occasional misclassifications. Such robustness has indeed been empirically observed [17].

In this paper, we propose a safety assurance technique of perception-based control systems. Our approach is inspired by notions of abstraction and compositional reasoning. An *exact abstraction* $\overline{P}$ (or over-approximation) of a sensing and perception subsystem P is obviously useful: if the abstract system obtained by substituting actual perception P with the abstract $\overline{P}$ can be verified, then we can infer safety of the original concrete system. If a model of a perception system (abstraction or otherwise) can substitute P and the resulting system can be verified, then we say that the model is *verifiable*.

However, the problem of proving that $\overline{P}$ is an exact abstraction of actual perception P, is at least as hard as the neural network verification problem. So, in this paper, we propose a pragmatic path forward: a model M for P is constructed, which still enjoys verifiability, and the abstraction relationship between M and P will have some error that can be estimated arbitrarily precisely with high probability.

Another benefit of the above approach is that we can choose an *intelligible* structure for the approximation M. That is, M not only proves safety but helps explicate *why* the overall system is safe and where it deviates from the actual perception P. The importance of such explanations has been argued in [18], [19]. While firmly claiming intelligibility requires user studies beyond the scope of the current paper, our construction does enable succinct English description of M and visualization of its error with respect to P.

Our main claim is that this is one of the first[2] approaches to provide safety assurances for realistic vision-based control

[2]The only other closely related works are [10] and [11].

systems with abstractions, approximate or otherwise. We use a piece-wise affine template for M: Suppose the ground truth perception input to the control system is $m^*(\mathbf{x})$, at a given state $\mathbf{x}$. In the actual vision pipeline, P estimates $m^*(\mathbf{x})$ using images, which depend on the state $\mathbf{x}$ and also environmental parameters $\mathbf{e}$ such as lighting and weather conditions. These environmental parameters $\mathbf{e}$ can add bias and variance in the estimation. Thus, $M(\mathbf{x})$ will be a *set-valued function* to account for such variations, where the center (mean or bias) of the set is a piece-wise affine function $\mathbf{A}_i(m^*(\mathbf{x})) + \mathbf{b}_i$ of the ground truth $m^*(\mathbf{x})$. We can infer the linear model using regression on the output from the vision pipeline P on images and their ground truth labels.

While the center (mean) of the set $M(\mathbf{x})$ is defined by training data, the size and shape of the set (variance) are inferred from the safety property. Assume that the control system with perfect perception is safe with respect to a given unsafe set U. Using program analysis tools like CBMC [20] and IKOS [21] on the controller code, we infer the set of *unsafe* perception outputs for any $\mathbf{x}$. Then, the set-valued output from $M(\mathbf{x})$ is determined to be the *largest set*, centered at $\mathbf{A}_i(m^*(\mathbf{x})) + \mathbf{b}_i$, that keeps the system safe. The computation of this largest set is an optimization problem. We call the resulting model M an *approximate abstraction of perception (AAP)*.

The constructed AAP M is a piece-wise affine set-valued function of the actual variable that the perception system P is trying to estimate. By construction, M is verified safe relative to a given property U. We also double-check this using CBMC by plugging in M into the downstream modules of the control system. Finally, we empirically evaluate the *precision* of the constructed AAPs, i.e., the error between M and P, across large environmental variations such as roads with varying numbers of lanes, lighting conditions, different types of crops.

We apply the method to two realistic end-to-end autonomous systems using the Gazebo simulator for rendering images and detailed vehicle control models: a vision-based lane tracking controller for an electric vehicle and a vision-based corn row scouting robot. For both case studies, our method can compute the AAP in few seconds for each part of state space in a simple grid partition. On the positive side, for parts of the state space, with a probability of at least 0.9, we observe the precision of M approximating P is over 90%. This analysis tells us that the end-to-end system is safe for these parts of state space, because the verifiable-by-construction M is likely to be an exact abstraction of P in these parts. Somewhat counter-intuitively, we observe that error between M and P can be high in some very safe states (e.g., vehicle at the center of the lane and aligned with the lane). While these states happen to be the ones where safety assurance is less important in practice, we discuss why this makes sense and how to address it with different verification techniques or multi-resolution approximation. In addition, we discuss how to use sublevel set partitioning to address the exponential growth of the number of parts using grid partitioning.

In summary, our contributions are: (1) Formalization of verifiable approximate models for vision-based perception used in autonomous systems. (2) An approach to compute piece-wise affine set-valued approximations. (3) Demonstration that the constructed approximations can be composed with the downstream modules for end-to-end verification using existing techniques. (4) Careful empirical evaluation of precision of the verified approximations on two significant case studies. (5) An analysis and simulation framework for two realistic vision-based control systems to be released on open source platforms. Safety, intelligibility, and precision appear to be useful dimensions for thinking about AAPs. The constructed approximate abstractions are useful for verification, identifying where perception fails, which can, in turn, help design better perception and help define system-level *operating design domains (ODDs)* [22].

II. Related Works

Simulation-based testing is the most common technique for the system level analysis. It evaluates on the whole system, provides useful debugging information, but cannot provide a proof of safety. In contrast, our analysis exploits the decomposition of an autonomous system's pipeline and gives a formal proof of the system safety *under a strong assumption* that the AAP over-approximates the perception component. We further discuss other related techniques.

A. Analysis of Closed-loop Systems With NNs

There is a closely related line of work on the analysis of closed-loop systems with ML-based perception. VerifAI [9] uses techniques like fuzz testing and simulation to falsify the system specifications. Our work provides a safe approximation and therefore complements the falsification approaches of VerifAI.

Our work is similar in spirit to the idea of using abstraction/contracts for perception components in the white paper by Păsăreanu et al. [23]. Following this idea, Katz et al. [11] in particular trains generative adversarial networks (GANs) to produce a simpler network. This simpler network transforms states and environment parameters to estimates similar to our AAP. In comparison, our work provides an intelligible set-valued function instead.

In addition, Dean et al. [12] considers synthesizing robust perception based controller. Ghosh et al. [10] uses VerifAI [9] for counter-example guided controller synthesis with perception models. Plenty of recent works focus on verification [3], [24]–[26], reachability analysis [5], [27]–[30], statistical model checking [31], and synthesis [32] on *neural feedback systems* with neural network controllers. Such controller NNs are typically very small compared to perception CNNs. To our knowledge, the case study with the largest image input dimension is NNLander-VeriF [13], which encodes formal camera models for black-white images with 16×16 pixels and verifies them along with NN controllers using existing neural network verification tools. Our work is the **first** to provide intelligible AAPs for highly cited CNN-based perception pipeline, LaneNet [33], and guarantees safety for the approximated closed-loop system.

B. Isolated Neural Network Verification

Recently, there are many works on verifying an isolated neural network such as ReLuplex [1], NNV [3], Verisig [5],

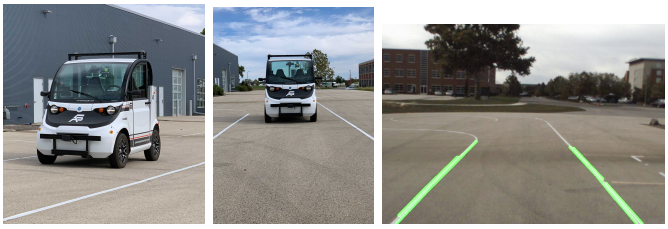

Fig. 1. Vision-based lane tracking control system (LTC) on AV platform.

etc. We refer readers to a summary of the VNN competition [6] for a complete list. It is, however, challenging for these isolated NN verification tools to analyze ML-based Perception. It is pointed out in [34] that the perception NN used for a small UAS is significantly larger than any NN verified in [6], and so are the NNs in our study. GAS [14] analyzes the impact of NN perception uncertainty on vehicles using an approximated model and Generalized Polynomial Chaos. Unlike our approach, the perception model from GAS does not incorporate system-level specifications.

C. Interpretability

Research in explainable AI (XAI) and interpretable machine learning (IML) has grown explosively (400+ publications on interpretability in 2020 [35]). The survey articles [18], [35], [36] provide overviews of XAI for text, image, and tabular models. Prominent techniques use feature importance [37], Shapley values [38], and counter-factual explanations [39]. Our piece-wise affine models are interpretable models of CNNs *in the context of control systems*—something we have not seen in the literature. In the XAI parlance, our method provides *model-agnostic*, *global* interpretations of image-based AI models. Our approximate models help with *transparency*, or equivalently *intelligibility*, in that they can help a human to understand the functioning of the perception system through visualizations and succinct English descriptions of the model's relationship to ground truth, as discussed in Section V-B.

III. System-level Safety Assurance

The problem of assuring safety of a control system can be stated as follows: Given a control system or a program Sys on a state space $\mathcal{X}$, we would like to check that it satisfies an invariant $\mathcal{I} \subseteq \mathcal{X}$. For example, for a lane tracking control system (LTC) for a vehicle in Fig. 1, the invariant requirement is that the vehicle always remains within the lanes. This textbook statement of the problem is complicated by two factors in an actual autonomous system.

First, Sys uses vision for perception – converting pixels to *percepts* such as deviation from lane-center, and such perception systems are not amenable to formal specification and verification. Secondly, the output of the perception pipeline depends on environmental factors E such as lighting, texture, and pavement moisture. These dependencies are neither well-understood nor controllable.

A. System Description

We model the complete control system as a discrete time transition system Sys with four components transforming

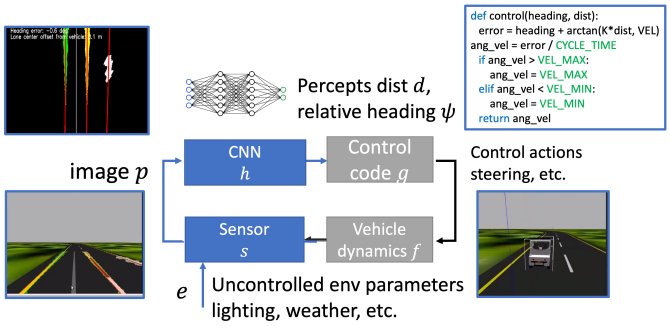

Fig. 2. Closed-loop model of LTC Sys with camera and CNN-based perception.

different types of data (Fig. 2). The vision-based perception pipeline takes an image (or a high-dimensional vector) $\mathbf{p}$ as an input and produces a percept or a low-dimensional *estimate* vector $\mathbf{z} = h(\mathbf{p})$ as the output. In a lane tracking control system (LTC), $\mathbf{z}$ is the position of the camera relative to the lanes seen in the image. That is, we model the perception pipeline as a function $h : \mathcal{P} \to \mathcal{Z}$ mapping the space of images $\mathcal{P}$ to the space of percepts $\mathcal{Z}$.

The *control* module takes a percept $\mathbf{z}$ as an input and produces a control action $\mathbf{u} = g(\mathbf{z})$ as the output. In LTC, the control action $\mathbf{u}$ is a vector of throttle, steering, and brake signals. The implementation of the controller *control* may involve a number of modules, including navigation, planning, and optimization. Abstractly, *control* is a function $g : \mathcal{Z} \to \mathcal{U}$ mapping the space of precepts to the space of control actions.

Then *dynamics* defines the evolution of the system state $\mathbf{x}$ as a function of the previous state and the output from the *control*. We model the *dynamics* as a function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. In our example, the state $\mathbf{x}$ of the vehicle includes its position, orientation, velocity, etc., and the dynamics function defines how the state changes with a given control action $\mathbf{u} \in \mathcal{U}$. In this paper, we consider discrete time models, and write the state at time $t+1$ as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t),$$

where $\mathbf{x}_t$ and $\mathbf{u}_t$ are the state and the control actions at time t. This state transition function could be generalized to a relation to accommodate uncertainty, without significantly affecting our framework or the results.

The final component closing the loop is the *sensor* which defines the image $\mathbf{p}$ as a function of the current state $\mathbf{x}$ and a set of non-time varying, *environmental parameters* $\mathbf{e}$. For LTC, these parameters include, for example, lighting conditions, nature of the road surface, types of markings defining lanes, etc. We model the *sensor* as a function $s : \mathcal{X} \times E \to \mathcal{P}$, where E is the space of environmental parameter values. In a real system, we may not know all the environmental parameters, they may not be time-invariant, and their precise functional influence on the image will also be unknown. Therefore, it does not make sense to prove anything mathematically about s. For the purpose of generating AAPs of $h \circ s$, we reasonably assume that we can sample inputs of s according to some distribution over E and $\mathcal{X}$. Based on the samples, an empirical precision of the AAP can be computed. In Proposition 4, we also give a lower bound for the actual precision of the AAP

using the empirical precision. In our experiments, we generate synthetic data using a simulator, and the same could also be done with the actual vehicle platform at a higher cost.

B. Assurances for the Closed-loop System

The behaviors of the overall system are modeled as sequences of states called *executions*. Given an initial state $\mathbf{x}_0 \in \mathcal{X}$ and an environmental parameter value $\mathbf{e} \in E$, an execution of the overall system $\alpha(\mathbf{x}_0, \mathbf{e})$ is a sequence of states $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ such that for each index t in the sequence:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, g(h(s(\mathbf{x}_t, \mathbf{e})))). \tag{1}$$

We would like to have methods that can assure that, given a range of environmental parameter values $E_0 \subseteq E$, an unsafe set $U \subseteq \mathcal{X}$, and a set of initial conditions $\mathcal{X}_0 \subseteq \mathcal{X}$, none of the resulting executions of the system from $\mathcal{X}_0$ can reach U under any choice of E_0. Such a method will be a useful tool in checking safety of autonomous systems. Secondly, it can help search for E_0 for which the system can (and cannot) be assured to be safe and, therefore, can be used as a scientific basis for specifying the *operating design domain (ODD)* [8] for the control system (and direct expensive field tests, respectively). Since, the functions s and h are partially unknown with unknown dependence on $\mathbf{e}$ and $\mathbf{x}$, it is unreasonable to look for the above type of methods. Instead, in this paper, we develop a method for the following weaker problem:

Problem*:* Given an unsafe set $U \subseteq \mathcal{X}$ and a range for the parameters $E_0 \subseteq E$, find an approximation M of the perception system $h \circ s$, such that it is:

(a) *Safe*, i.e., M used in the closed-loop system substituting $h \circ s$ makes the resulting system provably safe against U.
(b) *Intelligible*, i.e., human designers can understand the behavior of M.
(c) *Precise*, that is, M and $h \circ s$ are close.

M may and indeed will depend on the unsafe set U. For the substitution in (a) to make sense, we make M a set valued function to accommodate variations in $h \circ s$ from different environments. Since the actual perception system $h \circ s$ and its dependence on the environment E is incompletely understood, assertions about the closeness to precision (c) have to be statistical. We will see later that indeed fine-grained measurement of closeness is possible.

C. An Example: Vision-based Lane Keeping

The details of the lane tracking control system (LTC) model (of Fig. 1 and 2) are as follows.

a) Dynamics and control: The vehicle state $\mathbf{x} \in \mathcal{X}$ consists of the 2D position (x, y) of the center of the front axle in a global coordinate system, and the heading angle θ w.r.t the x-axis. The input $\mathbf{u} \in \mathcal{U}$ is the steering angle δ. The discrete time model is the well-known kinematic bicycle function [40] $f(\mathbf{x}, \mathbf{u})$:

$$\begin{aligned} x_{t+1} &= x_t + v_f \cdot \cos(\theta_t + \delta) \cdot \Delta T \\ y_{t+1} &= y_t + v_f \cdot \sin(\theta_t + \delta) \cdot \Delta T \\ \theta_{t+1} &= \theta_t + v_f \cdot \frac{\sin(\delta)}{L} \cdot \Delta T \end{aligned}$$

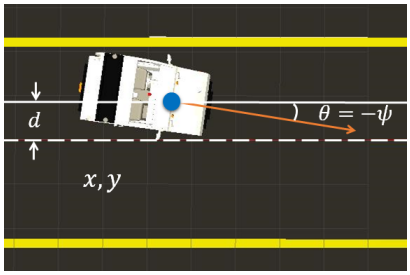

Fig. 3. State (x, y, θ) and perception variables (d, ψ) for lane keeping.

where v_f is the forward velocity, L is the wheel base, and ΔT is a time discretization parameter.

The input to f comes from the decision and control program. Here we use the standard Stanley controller [41] for lateral control of vehicles. This controller uses the percept $\mathbf{z} \in \mathcal{Z}$, which consists of the heading difference ψ and cross-track distance d from the center of the lane to the ego-vehicle. In Fig. 3, the heading θ coincides with the negation of heading difference $-\psi$, but this happens only in the special case where the lane is aligned with the x-axis. The controller function $g(\mathbf{z})$ is defined as:

$$\delta = g(d, \psi) = \begin{cases} \psi + \arctan\left(\frac{K \cdot d}{v_f}\right), & \text{if } \left|\psi + \arctan\left(\frac{K \cdot d}{v_f}\right)\right| < \delta_{max} \\ \delta_{max}, & \text{if } \psi + \arctan\left(\frac{K \cdot d}{v_f}\right) \geq \delta_{max} \\ -\delta_{max}, & \text{if } \psi + \arctan\left(\frac{K \cdot d}{v_f}\right) \leq -\delta_{max} \end{cases}$$

where δ_{max} is the steering angle limit and K is a controller gain parameter.

b) Perception: The complicated perception pipeline estimates heading difference ψ and cross track distance d using several computer vision functions. First, the sensor function s uses cameras to capture an image and processes the image through cropping, undistortion, resizing, etc., to prepare the image $\mathbf{p}$ for the neural network. The particular neural network used here is LaneNet [33] which uses 512×256 RGB images to detect lane pixels. Internally, LaneNet contains two 18-layer sub-nets for the identification and instance segmentation of lane marking pixels; then curve fitting is applied on identified pixels to represent each detected lane as a polynomial function. Further, the perspective warping is applied to map the lanes to the bird's eye view, which gives the percept $\mathbf{z} = (d, \psi)$ as shown in Fig. 2.

c) System safety requirement: A common specification for lane keeping control is to avoid going out of the lane boundaries. We assume the vehicle is driving on a straight road with lane width W. For the purpose of simplifying exposition, we assume that the center line is aligned with the x-axis of the global coordinate system. Thus, the unsafe set can be specified as $U = \{(x, y, \theta) \mid |y| > 0.5W\}$.

IV. Safe Approximated Abstract Perception

In this section we will discuss our method for constructing the approximation M for the perception system $h \circ s$. Section IV-A sets the stage. It shows that plugging in any set-valued approximation M of $h \circ s$ naturally defines an approximation $\widehat{Sys}(M)$ of the original system Sys. Section IV-B presents the main algorithm for constructing a particular structure of M. It learns, from perception data, the center (mean) of the output set $M(\mathbf{x})$. Section IV-C defines the next

step in the construction of M. This step analyzes the control program $f \circ g$ to optimize the shape and the size of the output set around the mean to assure the safety of $\widehat{Sys}(M)$ with respect to the unsafe set U. Section IV-D establishes the safety of the constructed M, not only at the theoretical model level, but it also shows how M can be plugged into the rest of the Sys code and verified using program analysis tools, namely CBMC [20] in our work. Finally, Section IV-E discusses our methods for empirically evaluating the precision of M.

A. Approximate Abstract Perception in Closed-loop

We will construct a set-valued perception function M that approximates the complex perception system $h \circ s$. For the safety requirement, our constructed function $M : \mathcal{X} \to 2^{\mathcal{Z}}$ should be such that when it is "substituted" in the closed-loop system of Equation (1), the resulting system is safe with respect to the requirement U. Formally, substituting $h \circ s(\mathbf{x}, \mathbf{e})$ with $M(\mathbf{x})$, the result is the non-deterministic system $\widehat{Sys}(M)$ given by:

$$\mathbf{x}_{t+1} \in \{f(\mathbf{x}_t, g(\mathbf{z})) \mid \exists \mathbf{z} \in M(\mathbf{x}_t)\}.$$

That is, when the actual system state is $\mathbf{x}_t$ (and the environmental parameters $\mathbf{e}$), then the output from the abstract perception function M can be *anything* in the set $M(\mathbf{x}_t)$. This set-valued approach is a standard way of modeling noisy sensors. Notice that M is independent of environments.

Definition 1. *A function $M : \mathcal{X} \to 2^{\mathcal{Z}}$ is an* abstraction *of $h \circ s$ if:*

$$\forall \mathbf{e} \in E, \forall \mathbf{x} \in \mathcal{X}. h \circ s(\mathbf{x}, \mathbf{e}) \in M(\mathbf{x}).$$

This definition requires $M(\mathbf{x})$ covers all possible percepts from $h \circ s(\mathbf{x}, \mathbf{e})$ for all states and environments, and that is why it is an abstraction.[3] If a function M is an abstraction of $h \circ s$, then it follows that $\widehat{Sys}(M)$ is an abstraction of Sys, that is, the set of executions of $\widehat{Sys}(M)$ contains the executions of Sys. Therefore, any state invariant $\mathcal{I} \subseteq \mathcal{X}$ for $\widehat{Sys}(M)$ carries over as an invariant of Sys.

Proposition 1. *If M is an abstraction of $h \circ s$ then $\widehat{Sys}(M)$ is an abstraction of Sys.*

Fixing an arbitrary initial state $\mathbf{x}_0$ and an environment $\mathbf{e}$, Proposition 1 follows immediately from Definition 1 by:

$$f(\mathbf{x}, g(h \circ s(\mathbf{x}, \mathbf{e}))) \in \{f(\mathbf{x}, g(\mathbf{z})) \mid \exists \mathbf{z} \in M(\mathbf{x})\}.$$

Definition 1 is too general to be useful for constructing safe, intelligible, and precise abstractions. At one extreme, it allows the definition $M(\mathbf{x}) := \{h \circ s(\mathbf{x}, \mathbf{e}) \mid \exists \mathbf{e} \in E\}$ which is a symbolic abstraction but does not help with intelligibility nor with safety. At the other end, we can make $M(\mathbf{x})$ to be the entire space of percepts $\mathcal{Z}$, which may be intelligible but not useful for safety.

Our approach is to *utilize available information about safety of the control system without perception*. Informally, consider a version of the closed-loop control system that uses the ground truth values of ψ, d instead of relying on the vision pipeline to estimate these values. In order to prove safety of this *ideal system* with respect to U, we can use standard invariant assertions [42]–[46] and derive inductive invariants $\mathcal{I} \cap U = \emptyset$. We will construct M for Sys that can utilize the knowledge of such invariants.

Definition 2. *Given a set $\mathcal{I} \subseteq \mathcal{X}$ and a function $M : \mathcal{X} \to 2^{\mathcal{Z}}$, M is* preserving *$\mathcal{I}$ if*

$$\forall \mathbf{x} \in \mathcal{I}, \forall \mathbf{z} \in M(\mathbf{x}), f(\mathbf{x}, g(\mathbf{z})) \in \mathcal{I}.$$

Finding an invariant preserving function satisfying Definition 2 will guide us towards creating more practical approximations of the perception system.

B. Learning Piece-wise Approximations from Data

For an invariant preserving abstract perception function $M : \mathcal{X} \to 2^{\mathcal{Z}}$ to be intelligible, for any $\mathbf{x} \in \mathcal{X}$, the output $M(\mathbf{x})$ should somehow be related to the ground truth value $\mathbf{z}^* \in \mathcal{Z}$ that the perception system is supposed to estimate. For example, for a given state $\mathbf{x} = (x, y, \theta)$ of the vehicle in the lane tracking system, the ground truth $\mathbf{z}^* = (d^*, \psi^*)$—consisting of the relative position to lane center (d^*) and the angle with the lane orientation (ψ^*)—is uniquely determined by the geometry of the vehicle, the camera, and the lanes. The perception system $h \circ s$ is designed to capture this functional relationship between $\mathbf{x}$ and $\mathbf{z}$ (and it is affected by the environment $\mathbf{e}$). For the sake of this discussion, let $m^*(\mathbf{x}) = \mathbf{z}^*$ be the idealized function that gives the ground truth percept $\mathbf{z}^*$ for any state $\mathbf{x}$. A well-trained and well-designed perception system $h \circ s$ should minimize the error[4] $||m^*(\mathbf{x}) - h \circ s(\mathbf{x}, \mathbf{e})||$ over relevant states and environmental conditions. As M is an AAP of $h \circ s$, to achieve precision, M should also minimize error with respect to $m^*(\mathbf{x})$.

In this paper, we consider a piece-wise affine structure of M. This is an expressive class of functions with conceptual and representational simplicity, and hence human readable and comprehensible. First, given a partition with N parts $\{\mathcal{X}_i\}_{i=1\ldots N}$ of the target invariant domain, i.e., $\mathcal{I} = \bigcup_{i=1}^{N} \mathcal{X}_i$, we define M as:

$$M(\mathbf{x}) = \begin{cases} \mathcal{R}_1(m^*(\mathbf{x})), & \text{iff } \mathbf{x} \in \mathcal{X}_1 \\ \quad \vdots & \\ \mathcal{R}_N(m^*(\mathbf{x})), & \text{iff } \mathbf{x} \in \mathcal{X}_N \end{cases}$$

where we search for $\mathcal{R}_i : \mathcal{Z} \to 2^{\mathcal{Z}}$ that returns a neighborhood around $m^*(\mathbf{x})$.

In what follows, we will show how $\mathcal{R}_i$'s can be derived as a linear function of $m^*(\mathbf{x})$ that is both safe with respect to the target invariant $\mathcal{I}$ and minimizes error over training data available from the perception system. `ComputeAAP` gives our algorithm to compute the approximation for each $\mathcal{X}_i$.

To find a candidate $\mathcal{R}_i : \mathcal{Z} \to 2^{\mathcal{Z}}$ for a given subset $\mathcal{X}_i \subseteq \mathcal{X}$, we consider that, when given $\mathbf{z}^*$ as input, $\mathcal{R}_i$ returns a parameterized ball defined as below:

$$\mathcal{R}_i(\mathbf{z}^*) = \{\mathbf{z} \mid \|\mathbf{z} - (\mathbf{A}_i \times \mathbf{z}^* + \mathbf{b}_i)\| \leq r_i\}$$

where the parameters $\mathbf{A}_i$ and $\mathbf{b}_i$ define an affine transformation from $\mathbf{z}^*$ to the ball's center, and r_i defines the radius. Here we are using a ball defined by the L^2 norm on $\mathcal{Z}$.

[3] We will see later that the M that we will construct cannot be guaranteed to satisfy this requirement, but it is motivated by this idea.

[4] The precise choice of the error function is a design parameter and we will discuss this further in later sections.

Input: Subspace $\mathcal{X}_i$; Invariant $\mathcal{I}$; Dynamics f; Control g; Ideal estimator m^*
Data: Training set of ground truth vs percepts $L = \{(\mathbf{z}^*_1, \mathbf{z}_1), \ldots, (\mathbf{z}^*_{|L|}, \mathbf{z}_{|L|})\}$
Output: Linear transform matrix $\mathbf{A}_i$; Translation vector $\mathbf{b}_i$; Safe Radius r_i

1 **Function** `ComputeAAP`
2 $\mathbf{A}_i$, $\mathbf{b}_i \leftarrow$ `LinearRegression(`L`);`
3 $r_i \leftarrow$ `MinDist(`$\mathbf{A}_i$,$\mathbf{b}_i$,$\mathcal{X}_i$,$\mathcal{I}$,f,g,m^*`);`
4 **return** $\mathbf{A}_i$, $\mathbf{b}_i$, r_i;

Algorithm 1: Construction of the AAP M for the part $\mathcal{X}_i$. The output set is represented by a center defined by transformation matrix $\mathbf{A}_i$ and a vector $\mathbf{b}_i$, and a ball around the center defined by r_i.

Our approach generalizes to other norms and linear coordinate transformations with examples in Section VII.

We start with the input to `ComputeAAP` in Algorithm 1. Besides the subset $\mathcal{X}_i \subseteq \mathcal{X}$, the invariant $\mathcal{I}$, aforementioned modules f, g, and m^*, `ComputeAAP` also requires a *training set* of pairs $(\mathbf{z}^*, \mathbf{z})$ where the $\mathbf{z}^* = m^*(\mathbf{x})$ is the ground truth, and $\mathbf{z} = h \circ s(\mathbf{x}, \mathbf{e})$ is the percepts obtained with the perception pipeline. These pairs can be obtained from existing labeled data for testing the vision pipeline or training CNNs. A labeled data point for h is already an image $\mathbf{p} = s(\mathbf{x}, \mathbf{e})$ sampled from $\mathcal{X}$ and E and its labeled ground truth $\mathbf{z}^* = m^*(\mathbf{x})$. In practice, the state $\mathbf{x} = (x, y, \theta)$ can be obtained from other accurate sensors such as GPS to label the images. We use the state $\mathbf{x} \in \mathcal{X}_i$ and obtain the ground truth $\mathbf{z}^* = m^*(\mathbf{x})$. We then simply collect the perceived $\mathbf{z} = h(\mathbf{p})$ by applying the vision pipeline on image $\mathbf{p}$.

`ComputeAAP` first uses the training set of pairs of $(\mathbf{z}^*, \mathbf{z})$ to learn $\mathbf{A}_i$ and $\mathbf{b}_i$ using multivariate linear regression. The next section describes how it infers a safe radius r_i around the center $\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i$ via constrained optimization.

C. Constructing Safe Approximations of Perception

At Line 2 of `ComputeAAP`, multivariate linear regression minimizes the distance from the center line $\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i$ to the training data in $\mathcal{X}_i$ and computes $\mathbf{A}_i$ and $\mathbf{b}_i$. Next, we would like to infer a safe radius r_i around the center line $\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i$. There is a tension between safety and precision in the choice of r_i. On the one hand, we want a larger radius r_i to cover more samples, making M a more conservative approximation of $h \circ s$. On the other hand, the neighborhood should not include any *unsafe perception value* that can cause a violation of $\mathcal{I}$.

Formally, the set of unsafe percepts is $\{\mathbf{z} \mid \exists \mathbf{x} \in \mathcal{X}_i . f(\mathbf{x}, g(\mathbf{z})) \notin \mathcal{I}\}$ and should be disjoint with the safe neighborhood. Fig. 4 illustrates such a safe neighborhood for one particular state $\mathbf{x}$. Note that $\mathcal{R}_i$ has to extend to all states $\mathbf{x} \in \mathcal{X}_i$, and hence we need to find a minimum r_i for any $\mathbf{x} \in \mathcal{X}_i$. At the same time, we would also like r_i as large as possible to cover more perceived values. Further, Fig. 5 shows we have to infer for all $\mathcal{X}_i$ in the partition.

Our solution is to find an r_i just below the minimum distance r^* from the center $\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i$ to the set of unsafe

Input: Subspace $\mathcal{X}_i$; Invariant $\mathcal{I}$; Linear transform matrix $\mathbf{A}_i$; Translation vector $\mathbf{b}_i$; Dynamics f; Control g; Ideal estimator m^*;
Local: Solver status *status*; Estimated minimum $\hat{r}$; Bound on estimated to true minimum bnd
Output: Safe radius $r_i \in \mathbb{R}_{\geq 0}$

1 **Function** `MinDist`
2 solver.addVar($\mathbf{x}, \mathbf{z}, \mathbf{x}'$)
3 solver.addConstraints($\mathbf{x} \in \mathcal{X}_i, \mathbf{x}' = f(\mathbf{x}, g(\mathbf{z})), \mathbf{x}' \notin \mathcal{I}$)
4 solver.setObjective($\|\mathbf{z} - (\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i)\|$)
5 *status*, $\hat{r}$, bnd = solver.minimize()
6 **if** *status is OPTIMAL or SUBOPTIMAL* **then** $r_i \leftarrow \hat{r} - bnd$
7 **else** $r_i \leftarrow +\infty$ // status is INFEASIBLE
8 **return** r_i

Algorithm 2: Minimum distance to unsafe percepts.

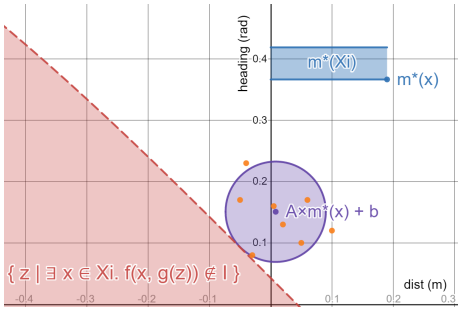

Fig. 4. Example safe neighbor function $\mathcal{R}_i$ inferred from linear regression and constrained optimization.

percepts. This is formalized as the constrained optimization problem below:

$$r_i < r^* = \min_{\mathbf{x} \in \mathcal{X}_i, \mathbf{z} \in \mathcal{Z}, \mathbf{x}' \in \mathcal{X}} \|\mathbf{z} - (\mathbf{A}_i \times m^*(\mathbf{x}) + \mathbf{b}_i)\| \quad \text{s.t. } \mathbf{x}' = f(\mathbf{x}, g(\mathbf{z})), \mathbf{x}' \notin \mathcal{I}$$

Observe that $\mathbf{x} \in \mathcal{X}_i$ is a set of simple bounds on each state variables by designing the partition. $\mathbf{x}' \notin \mathcal{I}$ is simply the invariant predicate over state variables. However, the third constraint $\mathbf{x}' = f(\mathbf{x}, g(\mathbf{z}))$ encodes the controller g and dynamics f in optimization constraints. Encoding the dynamic model f as optimization constraints is a common technique in Model Predictive Control. Encoding the controller g can be achieved with a program analysis tool to convert each if-branch of control laws into equality constraints between $\mathbf{z}$ and controller output $\mathbf{u} = g(\mathbf{z})$. An example template for Gurobi solver [47] is shown in `MinDist`.

We argue `ComputeAAP` computes a function $\mathcal{R}_i$ to return a safe neighborhood for any ground truth percept $m^*(\mathbf{x})$.

Proposition 2. *Every* $\mathbf{x} \in \mathcal{X}_i$, $\mathcal{R}_i(m^*(\mathbf{x}))$ *computed by* `ComputeAAP` *is disjoint from the unsafe percepts, i.e,*

$$\forall \mathbf{x} \in \mathcal{X}_i . \mathcal{R}_i(m^*(\mathbf{x})) \cap \{\mathbf{z} \mid \exists \mathbf{x} \in \mathcal{X}_i . f(\mathbf{x}, g(\mathbf{z})) \notin \mathcal{I}\} = \emptyset$$

Proof. The proof analyzes the possible outcomes of the optimization problem, and propagates each outcome through rest of the system functions. At Line 5, the solver may return the following status:

a) The status is OPTIMAL or SUBOPTIMAL: The solver returns a distance $\hat{r}$ and a bound bnd such that the true

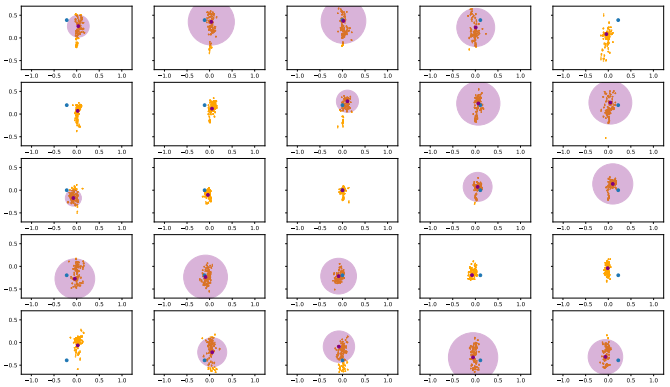

Fig. 5. Ground truth (blue dot), perceived values (orange points), and inferred safe neighborhood (purple circle). Notice the biases in different subspace: the mean of the perceived values do not align with the ground truth.

minimum r^* is within the bound, i.e., $\hat{r} \geq r^*$ and $\hat{r} - r^* < bnd$. Modern solvers all provide the bound to address numerical error or sub-optimal solutions. Consequently, $r_i = \hat{r} - bnd$ at Line 6 ensures $r_i < r^*$; thus the ball defined by r_i is disjoint with the unsafe set.

b) The status is INFEASIBLE: The constraints are unsatisfiable, i.e., the unsafe set $\{\mathbf{z} \mid \exists \mathbf{x} \in \mathcal{X}_i . f(\mathbf{x}, g(\mathbf{z})) \notin \mathcal{I}\}$ is proven to be $\emptyset$. We let $r_i = +\infty$ and thus $\mathcal{R}_i(\mathbf{x})$ is equivalent to the whole space of percepts $\mathcal{Z}$. □

D. Verifying with AAPs: Theory and Code

In this subsection, we summarize the claim that M computed by `ComputeAAP` indeed assures the safety of the approximated system $\widehat{Sys}(M)$ and show how it can be used for code-level verification. At a mathematical level, the safety of M follows essentially from the construction in `ComputeAAP`. Using Proposition 2, we can show that M preserves $\mathcal{I}$.

Proposition 3. *If every function $\mathcal{R}_i : \mathcal{Z} \to 2^{\mathcal{Z}}$ returns the safe neighborhood of $\mathcal{X}_i$ for all i, then the AAP M preserves the invariant $\mathcal{I}$.*

Proof. Let us fix $\mathbf{x} \in \mathcal{X}_i$ and the corresponding ground truth percept $m^*(\mathbf{x})$, and $\mathcal{R}_i(m^*(\mathbf{x}))$ represents all percepts allowed by $\mathcal{R}_i$. Using the $\mathcal{R}_i$ computed by `ComputeAAP`, we have shown in Proposition 2 that $\mathcal{R}_i(m^*(\mathbf{x}))$ does not intersect with any percept that can cause the next state $f(\mathbf{x}, g(\mathbf{z}))$ to leave $\mathcal{I}$. We then rewrite it as, for each $\mathbf{x} \in \mathcal{X}_i$, any percept $\mathbf{z} \in \mathcal{R}_i(m^*(\mathbf{x}))$ preserves $\mathcal{I}$, i.e,

$$\forall \mathbf{x} \in \mathcal{X}_i . \forall \mathbf{z} \in \mathcal{R}_i(m^*(\mathbf{x})) . f(\mathbf{x}, g(\mathbf{z})) \in \mathcal{I}. \tag{2}$$

Therefore, the invariant $\mathcal{I}$ is preserved for each subset $\mathcal{X}_i$, The proof of Proposition 3 is then to expand Definition 2 with the body of M and extend the guarantee from Equation (2) to all $\mathbf{x} \in \mathcal{I}$ simply because $\{\mathcal{X}_i\}_{i=1\ldots N}$ covers $\mathcal{I}$. □

More importantly, the constructed AAP M can be plugged into the models of the system Sys, with different levels of detail, and verified using any number of powerful formal verification tools that have been developed over the past decades. For example, the abstract perception system could be plugged into the controller g and dynamics f functions represented by complex, explicit models, code, and differential equations, and we can verify the resulting system rigorously.

To illustrate this point, in this paper, we showcase how to use M with C code implementations of g and f and verify the resulting system with CBMC [20] to gain a high-level of assurance for the control system. Recall our piece-wise affine AAP defined in Section IV-B, it can be directly translated into program *contracts*, that is, preconditions and postconditions, supported by numerous existing program analysis tools [20], [48]–[50]. For instance, we are able to implement M shown as C code in the following template with CBMC's APIs.

```
Z M(X x){
  __CPROVER_requires(⋁_{i=1}^{N} x ∈ X_i);
  Z z = nondet_z(); __CPROVER_ensures(
      ⋀_{i=1}^{N} x ∈ X_i → z ∈ R_i(m*(x))
  );
  return z;}
```

We then are able to verify the whole system integrating the controller and the dynamics shown in the example code in Appendix A with CBMC.

E. Measuring the Precision of AAPs

How close is the computed AAP M to the actual perception system $s \circ h$? As we discussed earlier, it is difficult, if not impossible, to rigorously answer this question because the perception system (and therefore the learning stage of M) depends on the $\mathbf{e}$ in complex and unknown ways. There are many options for measuring closeness that can factor in information about the environmental parameters.

We propose a simple and fine-grained empirical measure of precision. We fix a range of environmental parameter values E. For each part $\mathcal{X}_i$, we collect a *testing set* of pairs of $(\mathbf{z}^*, \mathbf{z})$ by sampling across $\mathcal{X}_i \times E$ using *some distribution* $\mathcal{D}$, where as before $\mathbf{z}^* = m^*(\mathbf{x})$ is the ground truth and $\mathbf{z} = h \circ s(\mathbf{x}, \mathbf{e})$ is the actual perception output. We denote a pair $(\mathbf{z}^*, \mathbf{z})$ that satisfies $\mathbf{z} \in \mathcal{R}_i(\mathbf{z}^*)$ as a positive pair. Then, the fraction of positive pairs gives us the empirical probability of $M(\mathbf{x})$ covering the vision-based perception system output $h \circ s(\mathbf{x}, \mathbf{e})$ in $\mathcal{X}_i$. Formally, the empirical probability with respect to $\mathcal{D}$ is defined as $\hat{p}_i = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}\left(\mathbf{z}_j \in \mathcal{R}_i(\mathbf{z}_j^*)\right)$, where $(\mathbf{z}_1^*, \mathbf{z}_1), (\mathbf{z}_2^*, \mathbf{z}_2), \cdots, (\mathbf{z}_N^*, \mathbf{z}_N)$ are i.i.d. samples from distribution $\mathcal{D}$, and $\mathbb{I}$ is the indicator function. In contrast, the actual probability, which is more important, is defined as $p_i = \mathbb{E}_{\mathcal{D}}\left[\mathbb{I}\left(\mathbf{z} \in \mathcal{R}_i(\mathbf{z}^*)\right)\right]$. Going forward we call p_i and $\hat{p}_i$ the *precision* and the *empirical precision* of M over $\mathcal{X}_i$.

The following proposition immediately follows from Hoeffding's inequality, which bound the difference between the actual and the empirical probabilities.

Proposition 4. *For any $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have that*

$$p_i \geq \hat{p}_i - \sqrt{-\frac{\ln \delta}{2N}}.$$

It may be tempting to interpret this probability as a probability of system-level safety, but without additional information how $\mathcal{D}$ is related to the actual distributions over $\mathcal{X}_i$ and E, we cannot make such conclusions.

In the following sections, we use a uniform distribution $\mathcal{D}$ over simulated states and environments. Each of the heatmaps shown in Fig. 6 illustrate the empirical precision $\hat{p}_i$ of different $\mathcal{X}_i$ of M. A darker green $\mathcal{X}_i$ means that a higher fraction of

outputs from the perception matches the provably safe AAP M. We ensure that at least $N = 300$ images are collected for each $\mathcal{X}_i$. By Proposition 4, with probability at least 0.9 (i.e., $\delta = 0.1$), we have that $p_i \geq \hat{p}_i - 0.062$.

V. Case study 1: Vision-based Lane Tracking with LaneNet

We study the Polaris GEM e2 electric vehicle and its high-fidelity Gazebo simulation [51] (Motivating example of Section III-C). The perception module uses LaneNet [33] for lane detection.[5] We discuss the construction of the approximation M in Section V-A, the interpretation of the precision *heatmaps* in Section V-B, finally, in Section V-C we study behavior of the closed-loop system where LaneNet ($h \circ s$) is replaced by the approximation (M). We aim to study the impact of partitions $\{\mathcal{X}_i\}_{i \leq N}$ and the environment parameter distributions $\mathcal{D}$.

A. Implementation Details in the Construction of AAPs

$\mathcal{X}_0 = \left\{(x, y, \theta) \mid x = 0 \wedge |y| \leq 1.2 \wedge |\theta| \leq \frac{\pi}{12}\right\}$ is the initial set of states, and we recall the unsafe set is $U = \{(x, y, \theta) \mid |y| > 2.0\}$. Next, we discuss about the invariant $\mathcal{I}$ we will use to prove U. A standard induction-based proof for control systems is to define an error function (Lyapunov function) over the perceived values, and then prove that the error is non-increasing by induction. Formally, a tracking error function is $V : \mathcal{Z} \mapsto \mathbb{R}_{\geq 0}$, with $V(\mathbf{0_z}) = 0$ and $V(\mathbf{z}) > 0$ when $\mathbf{x} \neq \mathbf{0_z}$. where $\mathbf{0_z} \in \mathcal{Z}$ is the equilibrium. In this section, we use the vector norm, i.e., $V(d^*, \psi^*) = \|(d^*, \psi^*)\|$. Choices of different tracking error functions are discussed in Appendix A.

The ground truth perceptual output of a state (x, y, θ) is $(d^*, \psi^*) = m^*(x, y, \theta)$, and the tracking error is $V(d^*, \psi^*)$. The next state according to Equation (1) is $(x', y', \theta') = f((x, y, \theta), g(m^*(x, y, \theta)))$. The next percept is then obtained by $(d^{*\prime}, \psi^{*\prime}) = m^*(x', y', \theta')$. We then define the invariant of non-increasing error $\mathcal{I} \subseteq \mathcal{X}$ as: $\mathcal{I} = \{\mathbf{x} \mid V(d^{*\prime}, \psi^{*\prime}) \leq V(d^*, \psi^*)\}$. Detailed descriptions and values of parameters in f and g and the definition of m^* are in Appendix A.

To infer the AAP M, we consider that the partition of the invariant is $\{\mathcal{X}_i\}_{i \leq N}$ with y within $\pm 0.3W = \pm 1.2$m to ensure safety and heading angle θ within $\pm 15^\circ$, i.e., $\bigcup_{i=1}^{N} \mathcal{X}_i = \left\{(x, y, \theta) \mid |y| \leq 1.2 \wedge |\theta| \leq \frac{\pi}{12}\right\}$. Further, we consider three different partitions[6] $N \in \{8 \times 5, 8 \times 10, 8 \times 20\}$; larger numbers partition more finely and refines coarser AAPs.

To prepare the training data for learning $\mathbf{A}_i$ and $\mathbf{b}_i$ to construct $\mathcal{R}_i$, we use the Gazebo model in [51] and generate camera images $\mathbf{p}$ labeled with their ground truth percepts $\mathbf{z}^*$. Each image is sampled from an uniform distribution $\mathcal{D}$ over $\mathcal{X}_i \times E$, where E is defined by: (i) three types of roads with two, four, and six lanes, (ii) two lighting conditions, day and dawn. The ground truth percept $\mathbf{z}^* = m^*(\mathbf{x})$ is calculated using information from simulator.

For each part $\mathcal{X}_i$, given $\mathbf{A}_i$ and $\mathbf{b}_i$ learned from multivariate linear regression using the data. `MinDist`, implemented in Gurobi [47], solves the following nonlinear optimization problem to find r_i: We discuss the optimization problem for a particular part $\mathcal{X}_i$ that covers y from 0.9 to 1.2 meters and θ from 12° to 15° as an example, i.e, $\mathcal{X}_i = \left\{(x, y, \theta) \mid y \in [0.9, 1.2] \wedge \theta \in \left[\frac{\pi}{15}, \frac{\pi}{12}\right]\right\}$.

$$
\begin{aligned}
\min_{(x,y,\theta)\in\mathcal{X}_i,(d,\psi)\in\mathcal{Z},(x',y'\theta')\in\mathcal{X}} \quad & \|(d, \psi) - (\mathbf{A}_i \times m^*(x, y, \theta) + \mathbf{b}_i)\| \\
\text{subject to} \quad & x' = x + v_f \cos(\theta + g(d, \psi))\Delta T, \\
& y' = y + v_f \sin(\theta + g(d, \psi))\Delta T, \\
& \theta' = \theta + v_f \frac{\sin(g(d, \psi))}{L}\Delta T, \\
& V(m^*(x', y', \theta')) > V(m^*(x, y, \theta))
\end{aligned}
$$

We composed the computed AAPs with the code for the controller g and the dynamics f and successfully verified the corresponding invariant with CBMC. In addition to being an extra check, this CBMC verification closes the gap between the mathematical functions used in constructing the verified AAP, and the corresponding C functions in code (E.g., $\arctan\left(\frac{K \cdot d}{v_f}\right)$ has to be implemented with `atan2` in C library to avoid division by zero).

TABLE I
Time usage for computing each of the six AAPs in Fig. 6.

	8×5		8×10		8×20	
	Total	Avg.	Total	Avg.	Total	Avg.
Three road types	271s	6.78s	617s	7.71s	1097s	6.85s
One road type	364s	9.10s	765s	9.57s	1351s	8.44s

B. Interpretation of the Precision of AAPs

We computed six AAPs resulting from three increasingly finer partitions and two sets of testing environments. Our method is able to compute AAPs efficiently even for finest partition $N = 8 \times 20$ as shown in Table I. Fig. 6 shows the precision heatmaps for the six AAPs. A darker green cell implies a higher empirical precision $\hat{p}_i$ (and therefore, a higher lower-bound of actual precision p_i by Proposition 4), i.e., the safe AAP approximates the perception system with higher probability in $\mathcal{X}_i$. First, we discuss the broad trends and then delve into the details.

a) At equilibrium, AAP breaks but it does not matter: All six heatmaps demonstrate a common trend where there is a lump of white (low score) cells around the origin. There are areas where either (1) the safe radius r_i of $\mathcal{R}_i$ is too small for M to include the outputs from $h \circ s$ *or* (2) the center of $\mathcal{R}_i$ is unsafe. This phenomenon can be understood as follows: First, the center (equilibrium) of the plot corresponds to near zero error in deviation d and heading ψ. Consider when a vehicle's state has nearly 0 tracking error; the percept must also approach the ground truth $m^*(\mathbf{x})$ so that the next state can maintain the 0 error. Recall that our AAP consists of the mean $\mathbf{A}_i m^*(\mathbf{x}) + \mathbf{b}_i$ and the safe radius r_i. If the mean $\mathbf{A}_i m^*(\mathbf{x}) + \mathbf{b}_i$ already deviates from ground truth, it can lead to control actions to always increase tracking error in the next state. In this case, we cannot infer a safe region around the bias, and M returns an empty set. The precision is 0 by definition. In the other case, the mean is close to the ground truth. The safe region to maintain nonincreasing error is still small, and hence r_i is almost 0. The precision will be very low because the percept from the vision pipeline is unlikely to be

[5] We use https://github.com/MaybeShewill-CV/lanenet-lane-detection, one of the most popular open source implementation of LaneNet on GitHub.

[6] Here we do not partition along x because lanes are aligned with the x-axis, and partitioning x-axis does not produce interesting results.

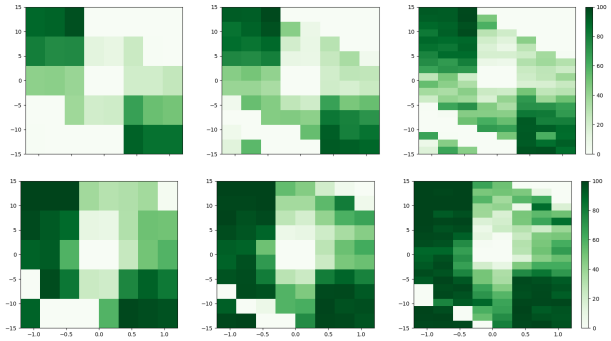

Fig. 6. Precision heatmaps of AAPs inferred for LaneNet with Stanley controller. The partitions with $N = 8\times5$, 8×10, and 8×20. The environment parameter space with three road types (*Top row*) and only two-lane road (*Bottom row*).

exactly equal to the ground truth. Alternatively, we can view the overall system $\widehat{Sys}(M)$ as a fixed-resolution quantized control system. It is well-known that such a system cannot achieve perfect asymptotic stability [52]. The feedback does not have enough resolution to drive the state to the equilibrium, and the error function V cannot be non-increasing around the origin. We note that not proving safety around the origin is less of a problem because the vehicle is safe—centered and aligned with the lane.

b) Finer partitions improve precision: We observe that finer partitions generate more precise approximations. With the finest partition, several cells are over 90 percent. This could be made higher with finer partitions. The reasons are twofold. 1) With a finer partition, regression can better fit a smaller interval of the original perception function. 2) The radius r_i is minimized for all $\mathbf{x} \in \mathcal{X}_i$. If a subset $\mathcal{X}_j \subset \mathcal{X}_i$ excludes the worst state, the radius r_j for $\mathcal{X}_j$ can be larger than r_i.

c) Fewer environmental variations improve precision: We generated two testing sets under different distributions over the environment space including 1) the same uniform distribution for the training set, and 2) an uniform distribution over the subspace with only the two-lane road. Fig. 6 shows that the colors become darker for the same cell locations. The variance in the perceived values by the vision pipeline reduces because of the fewer environmental variations. The same radius can cover more samples in the testing set.

C. Closed-loop System with Approximate Perception Model

We test the performance of the worst AAP (with the $N = 8\times5$ partition) by simulating it in the closed-loop lane-keeping system in Gazebo (blue in Fig. 7). At each time step, the AAP generates a set of possible perception values, and we randomly pick a point from this set and feed that into the controller g to close the loop. For comparison, we also run the original system with LaneNet (orange) and with perfect ground-truth (green) perception, starting from the exact same initial condition.

We run 50 simulation starting randomly from $0.6 \leq d \leq 0.9$ and $|\psi| \leq \frac{\pi}{60}$ each with time horizon 3s. For all these runs, we plot the mean and standard deviation of the perceived tracking error (left) and the actual tracking error (right) in Fig. 7. First, we observe that, as expected, the distribution of tracking error using AAP and LaneNet are both biased compared to the ground truth. Second, the perceived tracking error from

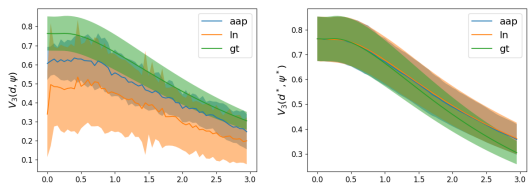

Fig. 7. $V(d,\psi)$ (Left) computed using perception output, and $V(d^*,\psi^*)$ (Right) computed using ground truth.

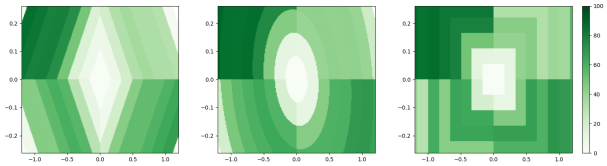

Fig. 8. Precision heatmaps of AAPs inferred for LaneNet with Stanley controller using partitions of eight sublevel sets with L^1(*Left*), L^2(*Mid*), L^∞(*Right*).

AAP is close to the tracking error of the actual system, and the real tracking error between the two is even closer. These experiments provide empirical evidence that the closed-loop system with the AAP, namely $\widehat{Sys}(M)$, closely approximates the actual system Sys on average.

D. Alternative Partitions with Sublevel Sets

The aforementioned grid based partitioning strategy has an obvious scalability issue. That is, the number of grids can grow exponentially against the dimension k of the percept domain $\mathcal{Z} \subseteq \mathbb{R}^k$. We further consider a partitioning strategy combining large grids with sublevel sets of the error tracking function $V(d,\psi) = \|(d,\psi)\|$ so that we can control the number of levels independent of the dimension k. We use more general *norm* functions $\|C\mathbf{z}\|_p$ defined by the standard L^p norm and an injective linear transformation matrix C for scaling and rotation. Fig. 8 shows the precision heatmaps using the partitions by four quadrants and eight sublevel sets, and each heatmap from left to right is generated from L^1, L^2, and L^∞ norms with proper scaling matrices. We can observe the exact same trend of low precision scores around the equilibrium as well as the upper right and lower left corners.

E. Safety with respect to Barrier Certificates

As discussed in Section V-B, the low precision score around the equilibrium is because a quantized system cannot achieve perfect asymptotic stability and maintain non-increasing tracking error. We therefore examine a relaxed invariant specification using *barrier certificates* [42]. We use the invariant barrier $\mathcal{I} = \{\mathbf{x} \mid V(m^*(\mathbf{x})) \leq \rho\}$ and manually derive the constant $\rho = 1.27$ such that the ideal system are proven to stay within the invariant barrier. Fig. 9 shows the precision heatmaps for the AAP with respect to the invariant barrier. We can observe the white cells around the equilibrium disappeared, but the white upper right and lower left corners, which the truly unsafe states are, remains.

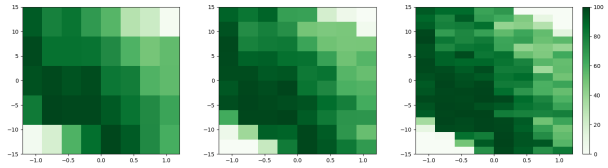

Fig. 9. Precision heatmaps of AAPs inferred from the invariant barrier $\mathcal{I} = \{\mathbf{x} \mid V(m^*(\mathbf{x})) \leq 1.27\}$ using grid based partitions with $N = 8\times5$, 8×10, and 8×20.

Fig. 10. Real and simulated camera images for corn row following for agricultural robots.

VI. Case Study 2: Corn Row Following AgBot

Our second case study is the visual navigation system, named CropFollow, for under-canopy agricultural robots (AgBot) developed in [53]. The system is responsible for avoiding collisions to the row boundaries when the vehicle traverses the space between two rows of crops. The system follows the same architecture in Fig. 2 with very similar interfacing variables: The vehicle state $\mathbf{x}$ consists of the 2D position x and y and the heading θ. The sensor captures the image $\mathbf{p}$ in front of the vehicle with a camera (Fig. 10). The percept $\mathbf{z} \in \mathcal{Z}$ composed of the heading difference ψ and cross track distance d to an imaginary center line of two rows.

However, all components in CropFollow are very different from the LTC system in GEM. The vehicle dynamics is a kinematic differential model of a skid-steering mobile robot in contrast to the bicycle model. The modified Stanley controller takes the percept and steers the robot to reduce the lateral deviation through the angular velocity ω instead of the steering angle. The perception component perceives the relative positions of the rows to the vehicle under largely varying crop field environments, such as different seasons of the crops, different plant types, etc. This is achieved by applying a ResNet-18 CNN on the camera image detailed in [53]. As a result, the error function for the system also differs.

For the farm robots, we wish to avoid two undesirable outcomes: 1) if $|y| > 0.5W = 0.38$ meters, the vehicle will hit the corn, and 2) if $|\theta| > 30°$, the neural network output becomes highly inaccurate and recovery may be impossible. That is, $U = \{(x, y, \theta) \mid |y| > 0.38 \wedge |\theta| > \frac{\pi}{6}\}$. We use the error function $V(d, \psi) = |\psi + \arctan(\frac{K \cdot d}{v_f})|$ from [41] to specify the invariant $\mathcal{I}$. The definition of the dynamics and controller, the partitions of states, environments, and constants for the dynamics, are provided in Appendix B.

To cover the invariant and disjoint from the unsafe set U, we choose the whole space $\bigcup_{i=1}^{N} \mathcal{X}_i$ covers $\pm0.3W = \pm0.228$ meters in y and $\pm30°$ in θ. We consider three different partitions $N \in \{5 \times 5, 10 \times 10, 20 \times 20\}$. We follow the same procedure to sample images and derive the safe neighbor function $\mathcal{R}_i$ for $\mathcal{X}_i$. For this case study, the environment parameter space E is defined by five different plant fields, including three stages of corns (baby, small, and adult) and two stages of tobaccos (early and late). We use the uniform distribution over the state space $\mathcal{X}_i$ and the five environments for both the training testing set.

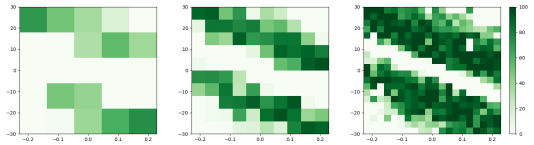

Fig. 11. Precision heatmap of AAPs inferred for CropFollow using $N \in \{5 \times 5, 10 \times 10, 20 \times 20\}$.

In Fig. 11, we observe almost identical broad trends including the white cells around equilibrium (the band from upper left to lower right), the white spots in the upper right and lower left corners close to the violation of invariant, and higher precision score with finer partitions. This case study reaffirms the validity of our interpretation of the precision heatmap in Section V. It also showcases that our analysis can be applied to different vision-based control systems.

VII. Discussion and Future Directions

Safety assurance of autonomous systems that use machine learning models for perception is an important challenge. We presented an approach creating approximate abstractions for perception (AAP) that are safe by construction. The approach learns piece-wise affine set-valued AAPs of the perception system from data. Viewing AAPs along the triple axes of safety, intelligibility, and precision may give a productive perspective for tackling the problems of safety assurance of autonomous systems.

Within the space of intelligible AAPs, we have explored one corner with piece-wise affine models. Our piece-wise affine AAPs use uniform rectangular partitions, and the size of the partitions have significant impact on improving precision. The results suggest that non-uniform or adaptive partitioning (e.g., finer partitions nearer to the equilibrium) would yield more precise approximations. Exploration of other structures such as decision trees, polynomial models, and space partitions, would be fruitful from the point of achieving precision without making the partition size too big.

As expected, the safety requirement and its verification method (e.g., invariants and Lyapunov functions) significantly impact the precision of the constructed approximation model. The precision maps shed light on parts of the state space and environment where the actual vision-based perception system is most fragile and is likely to violate requirements. Such quantitative insights can inform design decisions for the perception system, the control system, and the definition of the system-level *operating design domains (ODDs)*.

Finally, we chose to use discrete time models and used CBMC for verifying the closed system with the AAP. Extending the approach to continuous and hybrid would be interesting and require nontrivial extensions of existing verification tools.

VIII. Acknowledgments

The authors would like to acknowledge the constructive feedback from the anonymous reviewers. We thank the

help from Hang Cui, Arun Narenthiran, Prof. Katie Driggs-Campbell and Prof. Girish Chowdhary of the University of Illinois at Urbana-Champaign for the simulator of the GEM vehicle and AgBot. The work benefited from many insightful discussions with Aaron Mayne, Michael R. Abraham, Italo Romani De Oliveira, and Huafeng Yu of the Boeing Company. The researchers were supported by research grants from the National Science Foundation of the United States (Award numbers 2008883), the USDA National Institute of Food and Agriculture (USDA/NIFA) through the National Robotics Initiative (NIFA#2021-67021-33449), and the Boeing Company.

REFERENCES

[1] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Proc. 29th Int. Conf. CAV*, 2017, pp. 97–117.

[2] L. Pulina and A. Tacchella, "Never: A tool for artificial neural networks verification," *Annals of Mathematics and Artificial Intelligence*, vol. 62, no. 3–4, pp. 403—425, Jul. 2011.

[3] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *Proc. 32nd Int. Conf. CAV*, 2020, pp. 3–17.

[4] S. Dutta, X. Chen, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Sherlock - a tool for verification of neural network feedback systems: Demo abstract," in *Proc. 22nd ACM Int. Conf. HSCC*, 2019, pp. 262—263.

[5] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: Verifying safety properties of hybrid systems with neural network controllers," in *Proc. 22nd ACM Int. Conf. HSCC*, 2019, pp. 169—178.

[6] S. Bak, C. Liu, and T. Johnson, "The second international verification of neural networks competition (vnn-comp 2021): Summary and results," 2021. [Online]. Available: https://arxiv.org/abs/2109.00498

[7] E. A. S. Agency, "Easa concept paper: First usable guidance for level 1 machine learning applications," 2021. [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/easa_concept_paper_first_usable_guidance_for_level_1_machine_learning_applications_-_proposed_issue_01_1.pdf

[8] P. Koopman, U. Ferrell, F. Fratrik, and M. Wagner, "A safety standard approach for fully autonomous vehicles," in *Proc. 38th Int. Conf. SAFECOMP*, 2019, pp. 326–332.

[9] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Proc. 31st Int. Conf. CAV*, 2019, pp. 432–442.

[10] S. Ghosh, Y. V. Pant, H. Ravanbakhsh, and S. A. Seshia, "Counterexample-guided synthesis of perception models and control," in *2021 American Control Conf. (ACC)*. IEEE, 2021, pp. 3447–3454.

[11] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," in *Proc. 40th IEEE/AIAA DASC*, 2021, pp. 1–10.

[12] S. Dean, N. Matni, B. Recht, and V. Ye, "Robust guarantees for perception-based control," in *Proc. 2nd Conf. L4DC*, vol. 120, 2020, pp. 350–360.

[13] U. Santa Cruz and Y. Shoukry, "NNLander-VeriF: A Neural Network Formal Verification Framework for Vision-Based Autonomous Aircraft Landing," in *NASA Formal Methods*, J. V. Deshmukh, K. Havelund, and I. Perez, Eds. Cham: Springer International Publishing, 2022, pp. 213–230.

[14] K. Joshi, C. Hsieh, S. Mitra, and S. Misailovic, "Estimating Uncertainty of Autonomous Vehicle Systems with Generalized Polynomial Chaos," 2022. [Online]. Available: https://arxiv.org/abs/2208.02232

[15] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue, "Formal specification for deep neural networks," in *Proc. 16th Int. Symp. ATVA*, 2018, pp. 20–34.

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[17] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "No need to worry about adversarial examples in object detection in autonomous vehicles," https://arxiv.org/abs/1707.03501, 2017.

[18] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[19] C. Baier, C. Dubslaff, F. Funke, S. Jantsch, R. Majumdar, J. Piribauer, and R. Ziemek, "From verification to causality-based explications," 2021. [Online]. Available: https://arxiv.org/abs/2105.09533

[20] E. Clarke, D. Kroening, and F. Lerda, "A tool for checking ANSI-C programs," in *Proc. 10th Int. Conf. TACAS*, vol. 2988, 2004, pp. 168–176.

[21] G. Brat, J. A. Navas, N. Shi, and A. Venet, "Ikos: A framework for static analysis based on abstract interpretation," in *Proc. 12th Int. Conf. SEFM*, 2014, pp. 271–277.

[22] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?" in *SafeAI@ AAAI*, 2019.

[23] C. S. Păsăreanu, D. Gopinath, and H. Yu, *Compositional Verification for Autonomous Systems with Deep Learning Components*. Cham: Springer International Publishing, 2019, pp. 187–197.

[24] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verifying the safety of autonomous systems with neural network controllers," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 1, Dec. 2020.

[25] K. D. Julian and M. J. Kochenderfer, "Guaranteeing safety for neural network-based aircraft collision avoidance systems," *Proc. 38th IEEE/AIAA DASC*, pp. 1–10, 2019.

[26] ——, "Reachability analysis for neural network aircraft collision avoidance systems," *J. Guidance, Control, and Dynamics*, vol. 44, no. 6, pp. 1132–1142, 2021.

[27] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proc. 22nd ACM Int. Conf. HSCC*, 2019, pp. 157—168.

[28] J. Fan, C. Huang, X. Chen, W. Li, and Q. Zhu, "Reachnn*: A tool for reachability analysis of neural-network controlled systems," in *Automated Technology for Verification and Analysis*, 2020, pp. 537–542.

[29] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," in *59th IEEE Conf. Decision and Control,*, 2020, pp. 5929–5934.

[30] M. Everett, G. Habibi, C. Sun, and J. P. How, "Reachability Analysis of Neural Feedback Loops," *IEEE Access*, vol. 9, pp. 163 938–163 953, 2021.

[31] T. P. Gros, H. Hermanns, J. Hoffmann, M. Klauck, and M. Steinmetz, "Deep statistical model checking," in *Formal Techniques for Distributed Objects, Components, and Systems*, 2020.

[32] R. Ivanov, K. Jothimurugan, S. Hsu, S. Vaidya, R. Alur, and O. Bastani, "Compositional learning and verification of neural network controllers," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, Sep. 2021.

[33] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intell. Veh. Symp.*, 2018, pp. 286–291.

[34] K. Muvva, J. M. Bradley, M. Wolf, and T. Johnson, "Assuring learning-enabled components in small unmanned aircraft systems," in *AIAA Scitech 2021 Forum*, 2021.

[35] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable machine learning – a brief history, state-of-the-art and challenges," in *ECML PKDD 2020 Workshops*, 2020, pp. 417–431.

[36] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo, "Benchmarking and survey of explanation methods for black box models," 2021.

[37] D. W. Apley and J. Zhu, "Visualizing the effects of predictor variables in black box supervised learning models," 2019.

[38] D. Janzing, L. Minorics, and P. Blöbaum, "Feature relevance quantification in explainable AI: A causal problem," in *Proc. 23rd Int. Conf. Artificial Intelligence and Statistics*, vol. 108, 2020, pp. 2907–2916.

[39] T. Laugel, M. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "The dangers of post-hoc interpretability: Unjustified counterfactual explanations," in *Proc. 28th Int. Joint Conf. Artificial Intelligence, IJCAI 2019*, 2019, pp. 2801–2807.

[40] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.

[41] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *2007 American Control Conference*, 2007, pp. 2296–2301.

[42] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. 7th Int. Workshop HSCC*, 2004, pp. 477–492.

[43] S. Sankaranarayanan, H. B. Sipma, and Z. Manna, "Constructing invariants for hybrid systems," in *Proc. 7th Int. Workshop HSCC*, 2004, pp. 539–554.

[44] A. Platzer and E. M. Clarke, "Computing differential invariants of hybrid systems as fixedpoints," in *Proc. 20th Int. Conf. CAV*, 2008, p. 176–189.
[45] A. Platzer, *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*, 1st ed. Springer Publishing Company, Incorporated, 2010.
[46] S. Mitra, *Verifying Cyber-Physical Systems: A Path to Safe Autonomy*. Cambridge, MA, USA: MIT Press, 2021.
[47] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com
[48] M. Fähndrich, "Static verification for code contracts," in *Static Analysis*, 2010, pp. 2–5.
[49] G. T. Leavens, A. L. Baker, and C. Ruby, "Preliminary design of jml: A behavioral interface specification language for java," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 3, pp. 1–38, May 2006.
[50] M. Barnett, K. R. M. Leino, and W. Schulte, "The spec# programming system: An overview," in *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices*, 2005, pp. 49–69.
[51] P. Du, Z. Huang, T. Liu, T. Ji, K. Xu, Q. Gao, H. Sibai, K. Driggs-Campbell, and S. Mitra, "Online monitoring for safe pedestrian-vehicle interactions," in *2020 IEEE 23rd Int. Conf. Intell. Transportation Systems (ITSC)*, 2020, pp. 1–8.
[52] R. W. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *IEEE Trans. Autom. Control*, vol. 45, no. 7, pp. 1279–1289, 2000.
[53] A. N. Sivakumar, S. Modi, M. V. Gasparino, C. Ellis, A. E. Baquero Velasquez, G. Chowdhary, and S. Gupta, "Learned Visual Navigation for Under-Canopy Agricultural Robots," in *Proc. Robotics: Science and Systems*, Virtual, July 2021.

APPENDIX A
STANLEY CONTROLLER FOR GEM CAR

In this Appendix, we provide the details of the vehicle dynamics and the proof of non-increasing cross track distance for the GEM car discussed earlier in Section V.

TABLE II
CONSTANTS FOR POLARIS GEM E2 ELECTRIC CART CASE STUDY

Symbol	Value	Description
W	4.0	Width of the lane (m)
v_f	2.8	Constant forward velocity (m/s)
L	1.75	Wheel base (m)
ΔT	0.1	Time discretization (s)
δ_{max}	0.61	Steering angle limit (rad)
K	0.45	Stanley controller gain

a) Non-increasing cross track distance: Following the proof in [41], when $|\psi + \arctan(\frac{K\cdot d}{v_f})| < \delta_{max}$,

$$\dot{d} = -v_f \cdot \sin(\arctan(\frac{K\cdot d}{v_f})) = \frac{-K\cdot d}{\sqrt{1+(\frac{K\cdot d}{v_f})^2}}$$

Note that $\|d\|$ converges to zero because $-\frac{K\cdot d}{\sqrt{1+(\frac{K\cdot d}{v_f})^2}}$ is always the opposite sign of d. We can find the Lyapunov function for nominal region $V_2(d,\psi) = \|d\|$. This is however not entirely true in discrete dynamics because the value $\|d\|$ can cross zero and become larger in a discrete transition.

b) Non-increasing vector norm value: When in the nominal region $|\psi + \arctan(\frac{K\cdot d}{v_f})| < \delta_{max}$,

$$\dot{d} = -\frac{K\cdot d}{\sqrt{1+(\frac{K\cdot d}{v_f})^2}}; \quad \dot{\psi} = -\frac{v_f \cdot \sin(\psi + \arctan(\frac{K\cdot d}{v_f}))}{L}$$

The sign of $\dot{\psi}$ is opposite of $(\psi + \arctan(\frac{K\cdot d}{v_f}))$, so ψ approaches $\arctan(\frac{K\cdot d}{v_f})$. Further, $\arctan(\frac{K\cdot d}{v_f})$ converges to zero because d converges to zero as proven above. Therefore, the origin is the equilibrium, and L^2 norm is non-increasing.

TABLE III
DEFINITIONS OF TRACKING ERROR FUNCTIONS.

Tracking error function	Description
$V_1(d,\psi) = \|\psi + \arctan(\frac{K\cdot d}{v_f})\|$	Combined error in [41]
$V_2(d,\psi) = \|d\|$	Distance error only

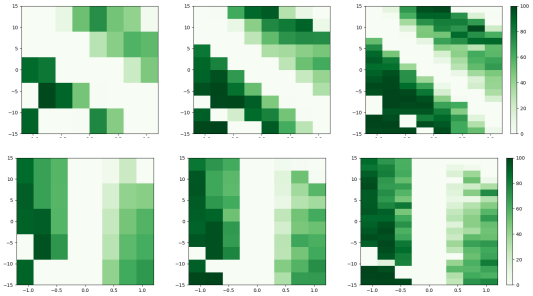

Fig. 12. Precision heatmaps for LaneNet with Stanley controller with only two-lane road for error functions V_1 (*Top*) vs V_2 (*Bottom*).

A. C Code Encoding for CBMC

The controllers in the paper are represented by a mathematical function g. This function is used in the algorithms for computing the safe approximate abstractions. However, the actual implementation of g is in code, and the two may have subtle discrepancies. To bridge this gap, we verify the controller *code* composed with the computed approximate perception model M and the dynamics using CBMC.

```
U g(Z z) { // Stanley controller example code
  U δ = z.ψ + atan2(K*z.d, v_f);
  if(δ >= δ_max)
    δ = δ_max;
  else if(δ <= -δ_max)
    δ = -δ_max;
  return δ; }
X f(X x, U δ) { // Bicycle model example code
  X new_x;
  new_x.x = x.x + v_f*cos(x.θ+δ)*ΔT;
  new_x.y = x.y + v_f*sin(x.θ+δ)*ΔT;
  new_x.θ = x.θ + v_f*sin(δ)/L*ΔT;
  return new_x; }
```

B. Variations with Different Safety Requirements

We consider other invariants which uses different tracking error functions listed in Table III. V_1 is the original function in [41] to combine the heading and lane deviation as a single tracking error. V_2 considers only lane deviation error (d). Both can be used to prove the same unsafe set U. Three heatmaps for each tracking error function are in Fig. 12 for the same three partitions and with the testing set with two-lane road.

a) Weak invariants can break safe AAP: Along the diagonal line (through the origin) we have states where the vehicle's deviation from the lane center d and the heading ψ are in opposing directions. By observing V_1 from Table III, we know ψ and d are of opposite signs at the equilibrium points $V_1(d,\psi)$. Hence, the band of white cells goes from the second to the fourth quadrant; the tracking error of these states cannot be non-increasing in one step as required by $\mathcal{I}$. Similarly, we see a white band surrounding the line $d = 0$ for V_2. This validates our explanation that the AAP breaks due to the stringent requirement of non-increasing error.

APPENDIX B
MODIFIED STANLEY CONTROLLER WITH AGBOT

TABLE IV
CONSTANTS USED IN THE AGRICULTURAL ROBOT CASE STUDY.

Symbol	Value	Description
W	0.76	Width of the corn row (m)
v_f	1.0	Constant forward velocity (m/s)
ΔT	0.05	Time discretization (s)
ω_{max}	0.5	Angular velocity limit (rad/s)
K	0.1	Stanley controller gain

The dynamics $f(\mathbf{x}, \mathbf{u})$ is:

$$\begin{aligned} x_{t+1} &= x_t + v_f \cos(\theta_t)\Delta T \\ y_{t+1} &= y_t + v_f \sin(\theta_t)\Delta T \\ \theta_{t+1} &= \theta_t + \omega \Delta T \end{aligned}$$

The controller g is given as:

$$g(d,\psi) = \begin{cases} \frac{\psi + \arctan\left(\frac{K \cdot d}{v_f}\right)}{\Delta T}, & \text{if } \left|\psi + \arctan\left(\frac{K \cdot d}{v_f}\right)\right| < \omega_{max} \cdot \Delta T \\ \omega_{max}, & \text{if } \psi + \arctan\left(\frac{K \cdot d}{v_f}\right) \geq \omega_{max} \cdot \Delta T \\ -\omega_{max}, & \text{if } \psi + \arctan\left(\frac{K \cdot d}{v_f}\right) \leq -\omega_{max} \cdot \Delta T \end{cases}$$