2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Certified Vision-based State Estimation for Autonomous Landing Systems using Reachability Analysis

Ulices Santa Cruz[1] and Yasser Shoukry[1]

***Abstract*— This paper studies the problem of designing a certified vision-based state estimator for autonomous landing systems. In such a system, a neural network (NN) processes images from a camera to estimate the aircraft's relative position with respect to the runway. We propose an algorithm to design such NNs with certified properties in terms of their ability to detect runways and provide accurate state estimation. At the heart of our approach is the use of geometric models of perspective cameras to obtain a mathematical model that captures the relation between the aircraft states and the inputs. We show that such geometric models enjoy mixed monotonicity properties that can be used to design state estimators with certifiable error bounds. We show the effectiveness of the proposed approach using an experimental testbed on data collected from event-based cameras.**

I. INTRODUCTION

Machine learning models, like deep neural networks, are increasingly used to control dynamical systems in safety-critical applications. These black-box models trained using data are used heavily to process high-dimensional imaging data like LiDAR scanners and cameras to produce state estimates to low-level, model-based controllers. While these deep Neural Networks (NNs) provide empirically accepted results, they lack certified guarantees in terms of their ability to process complex scenes and provide estimates of the location of different objects within the scene. It is then unsurprising the increasing number of reported failures of these deep NNs in building reliable autonomous systems.

In this paper, we will consider the safety of deep neural networks that control aircraft while approaching runways to perform an autonomous landing. Such a problem enjoys geometric nature that can be exploited to develop a geometrical/physical model of the perception system. Yet, it represents a significant real-world problem of interest to the designers of the autonomous system. In particular, we present a novel neural network-based filter that can process complex scenes along with estimates of the state of the aircraft—computed by unverified complex deep neural networks—and output a state estimate of the aircraft with a certified error bound. That is, akin to the "control shields" in the reinforcement learning literature [1], [2], the proposed filter can be thought of as a "shield" that can filter out incorrect estimates of the aircraft and replaces them with ones with certified error bounds. In contrast, the correct estimates pass this filter (or shield) unaltered.

A central challenge to designing such a filter is the need to explicitly model the imaging process, i.e., the relation between the system state and the images created by the camera [3]. An early result on the application of formal verification for vision-based dynamical systems controlled with neural networks [4] focused only on the usage of LiDARs. The first steps in formally modeling the imaging process for camera-based systems have been recently studied in [5], [6], [7]. In particular, the work in [5], [6] proposes the use of abstractions of the perception system as a formal model of perception. Unfortunately, these abstractions are only tested on a set of samples and lack guarantees in their ability to model the perception system formally. The work in [7] extends the notion of imaging-adapted partitions, originally defined for LiDAR images [4], to the notion of image-invariant regions, which are regions within which the captured images are identical. Unfortunately, the work in [7] focuses only on simple scenes that can be modeled as a collection of triangles that represent the triangulated faces of objects in the environment. The work in [8] considers the problem of estimating the pose of different objects in the scene. Given a partial point cloud of an object, the goal is to estimate the object's pose and provide a certificate of correctness for the resulting estimate. While capable of handling complex objects, the framework in [8] is sound but not complete, meaning that if it can identify the object's pose, it will generate a certificate. Still, not all poses of the object will be identified, even if the object of interest exists in the scene. Other techniques include classification that uses targeted inputs with the aim of finding counterexamples that violate safety [9]. However, such techniques do not provide formal guarantees regarding the ability to find all counterexamples.

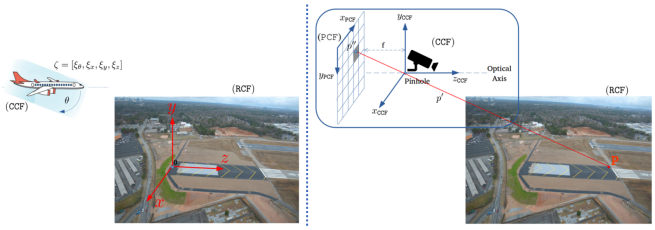

Fig. 1: Coordinate frames: Runway (RCF), Camera (CCF) and Pixel (PCF).

In this paper, we build on our recent results [3] that exploit the geometry of the autonomous landing problem to construct a formal model for the image formation process (a map between the aircraft states and the image produced by the camera). This physics-based formal model is designed such that it can be encoded as a neural network (with manually chosen weights) that we refer to as the Runway Generative Model neural network. Thanks to the recent development in computing the reachable sets of neural networks (the set of all possible outputs of the network) [10], [11], [12], we can characterize the set of all possible images for the runway. We use such reachability analysis to design novel filters that can remove all the other objects in the scene

This work was partially sponsored by the NSF awards #CNS-2002405, #CNS-2013824, and #CNS-2313104.

[1]Ulices Santa Cruz, and Yasser Shoukry are with the Department of Electrical Engineering and Computer Science, University of California Irvine, Email: {usantacr,yshoukry}@uci.edu

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10384107

by matching the spatial and geometrical properties of the runway to those in the computed reachable set. Moreover, as a by-product of this design, the proposed filter identifies the set of possible state estimates of the aircraft. This set of possible state estimates can then be used to cross-check the ones computed by unverified neural network estimators and provide certifiable error bounds on the final state estimate.

II. Preliminaries

A. Notation

We denote by $\mathbb{N}$, $\mathbb{B}$, $\mathbb{R}$ and $\mathbb{R}^+$ the set of natural, Boolean, real, and non-negative real numbers, respectively. We use $||x||_\infty$ to denote the infinity norm of a vector $x \in \mathbb{R}^n$. We denote by $\mathcal{B}_r(c)$ the infinity norm centered at c with radius r, i.e., $\mathcal{B}_r(c) = \{x \in \mathbb{R}^n | ||c - x||_\infty \leq r\}$. We use the notation $A_{[i,j]}$ to denote the element in the i^{th} row and j^{th} column of A. Analogously, the notation $A_{[i,:]}$ denotes the i^{th} row of A, and $A_{[:,j]}$ denotes the j^{th} column of A; when A is a vector instead, both notations return a scalar. Let $\mathbf{0}_{n,m}$ be an $(n \times m)$ matrix of zeros, and $\mathbf{1}_{n,m}$ be the $(n \times m)$ matrix of ones. Finally, the symbols $\oplus$ and $\otimes$ denote element-wise addition and multiplication of matrices.

B. Aircraft State Space

In this paper, we consider an aircraft landing on a runway. We assume the states of the aircraft to be measured with respect to the origin of the Runway Coordinate Frame (shown in Figure 1 (left)), where positions are: ξ_x is the axis across runway; ξ_y is the altitude and ξ_z is the axis along the runway. We consider only one angle ξ_θ, representing the pitch rotation around the x axis of the aircraft. The state vector of the aircraft at time $t \in \mathbb{N}$ is denoted by $\xi^{(t)} \in \mathbb{R}^4 = [\xi_\theta^{(t)}, \xi_x^{(t)}, \xi_y^{(t)}, \xi_z^{(t)}]^T$.

C. Runway Parameters

We consider a runway that consists of two border line segments, L and R. Each line segment can be characterized by its start and end point (also measured in the Runway Coordinate Frame) i.e., $L = [(L_x, 0, L_z), (L_x + r_w, 0, L_z + r_l)]$ and $R = [(R_x, 0, R_z), (R_x + r_w, 0, R_z + r_l)]$ where r_w and r_l refers to the runway width and length (e.g. standard international runways are designed with $r_w = 40$ meters wide and $r_l = 3000$ meters).

D. Camera Model

We assume the aircraft is equipped with a monochrome camera $\mathcal{C}$ that produces images of $a \times b$ pixels. Since the camera is assumed to be monochromatic, each pixel in the image I takes a value of 0 or 1. The image produced by the camera depends on the relative location of the aircraft with respect to the runway and the other objects in the scene. In other words, we can model the camera $\mathcal{C}$ as a function that maps aircraft states into images, i.e., $\mathcal{C} : \mathbb{R}^4 \to \mathbb{B}^{a \times b}$. Although the images created by the camera depend on the runway parameters and the other objects in the scene, we drop this dependence from the notation $\mathcal{C}$ for ease of notation. In this paper, we utilize an ideal pinhole camera model [13] and leverage our prior work [3] to capture the image formation process of this camera.

Fig. 2: Monochromatic images generated using state-of-the-art event-based cameras. The full image I to the left can be decomposed into one that contains only the runway image I_r (center) and the remaining objects/noise I_n (right), i.e., $I = I_r + I_n$.

Since the scene contains both a runway and other unknown objects (see Figure 2), we define the final image $I \in \mathbb{B}^{a \times b}$ captured by the camera as:

$$I(\xi) = I_r(\xi) + I_n(\xi) \tag{1}$$

where $I_r \in \mathbb{B}^{a \times b}$ is the image corresponding to the existence of the runway in the scene and $I_n \in \mathbb{B}^{a \times b}$ is the image corresponding to the existence of other objects/noise.

E. Neural Network Estimator

We are interested in designing a Neural Network (NN)-based estimator that can process an image $I(\xi) = I_r(\xi) + I_n(\xi)$ to produce an estimate of the aircraft state ξ. An F-layer NN is specified by composing F layer functions (or just layers). A layer ω with $\mathtt{i}_\omega$ inputs and $\mathtt{o}_\omega$ outputs is specified by a weight matrix $W^\omega \in \mathbb{R}^{\mathtt{o}_\omega \times \mathtt{i}_\omega}$ and a bias vector $b^\omega \in \mathbb{R}^{\mathtt{o}_\omega}$ as follows:

$$L_{\theta^\omega} : z \mapsto \phi(W^\omega z + b^\omega), \tag{2}$$

where ϕ is a nonlinear function, and $\theta^\omega \triangleq (W^\omega, b^\omega)$ for brevity. Thus, an F-layer NN is specified by F layer functions $\{L_{\theta^\omega} : \omega = 1, \ldots, F\}$ whose input and output dimensions are composable: that is, they satisfy $\mathtt{i}_\omega = \mathtt{o}_{\omega-1}$, $\omega = 2, \ldots, F$. Specifically:

$$\mathcal{NN}(I) = (L_{\theta^F} \circ L_{\theta^{F-1}} \circ \cdots \circ L_{\theta^1})(I). \tag{3}$$

As a common practice, we allow the output layer L_{θ^F} to omit the nonlinear function ϕ.

F. Problem Formulation

Problem 1. *Given an image $I(\xi) = I_r(\xi) + I_n(\xi)$ that contains the projection of a runway and other unknown objects and an estimation error $\epsilon > 0$, design a neural network estimator $\mathcal{NN}$ such that $||\mathcal{NN}(I_r + I_n) - \xi|| < \epsilon$.*

III. Framework

Classical machine learning approaches to solve Problem 1 entail training neural networks on large labeled data sets that contain different possibilities of runway positions and surrounding objects. Since ensuring the correctness of the resulting NN is challenging, we propose a framework in which we manually design a NN filter $\mathcal{NN}_\mathcal{F}$ that is guaranteed to "filter out" the noise I_n, i.e., $\mathcal{NN}_\mathcal{F}(I_r + I_n) = I_r$. Moreover, such a filter $\mathcal{NN}_\mathcal{F}$ also computes a certified bound on the possible states of the aircraft $\hat{\Xi}$. The size of this possible set of states $\hat{\Xi}$ is chosen to guarantee the ϵ bound in Problem 1. The resulting filtered-out image $\mathcal{NN}_\mathcal{F}(I_r + I_n)$ is then passed into a neural network estimator $\mathcal{NN}_e$ that is trained using existing techniques in machine learning.

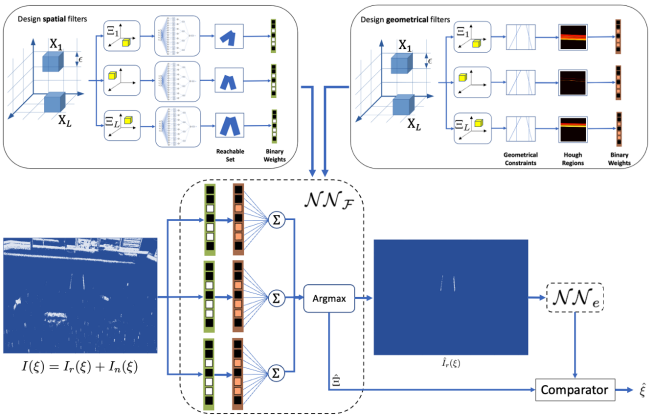

Fig. 3: Overall main framework elements: Spatial filter, Geometrical filter, $\mathcal{NN}_{\mathcal{F}}$ and $\mathcal{NN}_e$

Finally, the outcome of $\mathcal{NN}_e$ is checked against the certified bounds $\hat{\Xi}$ to provide the final estimate as:

$$\hat{\xi} = \begin{cases} \mathcal{NN}_e\left(\mathcal{NN}_{\mathcal{F}}(I)\right) & \text{if } \mathcal{NN}_e\left(\mathcal{NN}_{\mathcal{F}}(I)\right) \in \hat{\Xi} \\ \text{center}(\hat{\Xi}) & \text{otherwise} \end{cases} \tag{4}$$

where center($\hat{\Xi}$) is well defined whenever the set $\hat{\Xi}$ is a hypercube. In other words, the certified bounds $\hat{\Xi}$ are used to *replace* the incorrect state estimates with ones with guaranteed error bound from within the set $\hat{\Xi}$. This process is depicted in Figure 3. Steps to manually design the NN filter $\mathcal{NN}_{\mathcal{F}}$ and its theoretical guarantees are given in the subsequent subsections.

A. Physics-based Generative Model for Runway Images:

Our prior work in [3] developed a physics-based generative model that can generate all possible images containing runways $I_r(\xi)$ based on the physical parameters of the camera $f, \rho_h, \rho_w, v_0, u_0$ (discussed in Section 2). Crucially, this physics-based generative model was shown to be mathematically equal to a change of coordinates $h : \mathbb{R}^4 \to \mathbb{R}^4$ and a neural network $\mathcal{NN}_r(h(\xi))$ with carefully selected weights and parameters, i.e.,

$$I_r(\xi) = \mathcal{NN}_r(h(\xi)).$$

The change of coordinates h maps the state of the aircraft into the projections of the endpoints of the lines L and R on the Pixel Coordinate Frame (PCF). For the sake of brevity, we omit the details of h and $\mathcal{NN}_r$ and we refer the reader to [3] for the detailed analysis of the correctness of this generative model.

B. Design of spatial filters using output reachability analysis

Given a partitioning parameter δ, we partition the state space $\Xi \subset \mathbb{R}^4$ into L regions $\Xi_1, \ldots, \Xi_L$ such that each Ξ_i is an infinity-norm ball with radius δ. For each of these partitions, we aim to design a spatial filter that matches the spatial properties of the runway images that can be produced by states within such a partition. To that end, consider the following filter $\mathcal{S}^{\Xi_i} \in \mathbb{B}^{a \times b}$ defined as:

$$\mathcal{S}^{\Xi_i} = \bigotimes_{h(\xi) \in \Xi_i} I_r(\xi) = \bigotimes_{h(\xi) \in \Xi_i} \mathcal{NN}_r(h(\xi)). \tag{5}$$

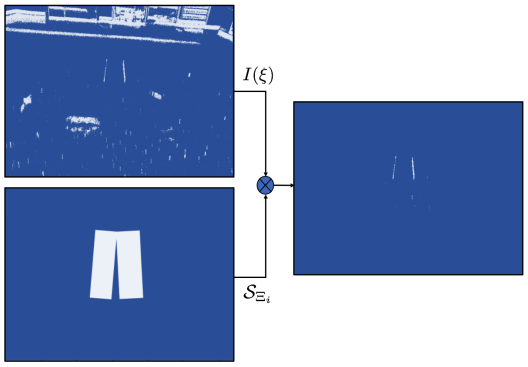

Fig. 4: Spatial filtering focuses attention on different regions.

Recall that all images $I_r(\xi)$ are monochromatic (i.e., each pixel can take only a value of 0 or 1), then the following result follows directly from the definition above.

Proposition 1. *Consider the filter $\mathcal{S}^{\Xi_i}$ defined in (5). The following holds:*

$$(i)\xi \in \Xi_i,\ \forall \xi \in \Xi_i.[I_n(\xi) \otimes \mathcal{NN}_r(h(\xi)) = \mathbf{0}_{a,b}] \implies [I_r(\xi) + I_n(\xi)] \otimes \mathcal{S}^{\Xi_i} = I_r(\xi) \tag{6}$$

$$(ii)\xi \notin \Xi_i,\ I_n(\xi) \notin \mathcal{I}_r^{\Xi_i} \implies [I_r(\xi) + I_n(\xi)] \otimes \mathcal{S}^{\Xi_i} \neq I_r(\xi) \tag{7}$$

where $\mathcal{I}_r^{\Xi_i} = \{I_r(\xi) \in \mathbb{B}^{a \times b} | h(\xi) \in \Xi_i\}$.

Note that the condition $\forall \xi \in \Xi_i.[I_n(\xi) \otimes \mathcal{NN}_r(h(\xi)) = \mathbf{0}_{a,b}]$ is equivalent to $I_n(\xi) \otimes \mathcal{S}^{\Xi_i} = \mathbf{0}_{a,b}$. That is, the filter $\mathcal{S}^{\Xi_i}$ is capable of removing all noise in the image as long as the noise image $I_n(\xi)$ does not affect pixels that are δ/ρ_w away from the runway image $I_r(\xi)$. Figure 4 shows an example of such a filter. Specifically, equations (6)-(7) imply that the filter will accurately process the filtered image, provided that the noise does not resemble the pattern of runways. Additionally, the filter must be applied to the specific region corresponding to the state responsible for generating such a runway. Furthermore, it is reasonable to assume that as we increase the geometric complexity of the runway, the likelihood of noise resembling runway patterns diminishes. In other words, the more intricate the entity we are examining, the safer it is to rely on our assumptions.

What is remaining is to provide an algorithm that can compute the filter $\mathcal{S}^{\Xi_i}$ for each partition Ξ_i. Thanks to the fact that the physics-based generative model $\mathcal{NN}_r(\xi)$ is captured as a neural network, one can use output reachability algorithms to compute an overapproximation of the reach set (set of all possible images) for the runway image $I_r(\xi)$. To that end, we leverage Mixed-monotonicity reachability analysis of neural networks [14] as follows:

Proposition 2. *(from [14]) Given a neural network $\mathcal{NN} : \mathbb{R}^{\mathfrak{i}} \to \mathbb{R}^{\mathfrak{o}}$ and an interval $[\underline{J}, \overline{J}] \subseteq \mathbb{R}^{\mathfrak{o} \times \mathfrak{i}}$ bounding the derivative of $\mathcal{NN}$ for all input $\zeta \in [\underline{\zeta}, \overline{\zeta}]$. Let us denote the center of the interval as J^* and for each output dimension $i \in \{1, ..., \mathfrak{o}\}$, define input vectors $\underline{\zeta}_{[i,:]}, \overline{\zeta}_{[i,:]} \in \mathbb{R}^{\mathfrak{i}}$ and a row vector $\alpha^i \in \mathbb{R}^{1 \times \mathfrak{i}}$ such that for all $j \in \{1, ..., \mathfrak{i}\}$ the*

following holds:

$$(\underline{\psi}_{[i,j]}, \overline{\psi}_{[i,j]}, \alpha_{[i,j]}) = \begin{cases} (\underline{\zeta}_{[:,j]}, \overline{\zeta}_{[:,j]}, min(0, \underline{J}_{[i,j]})) & \text{if } J^*_{[i,j]} \geq 0 \\ (\overline{\zeta}_{[:,j]}, \underline{\zeta}_{[:,j]}, max(0, \overline{J}_{[i,j]})) & \text{if } J^*_{[i,j]} \leq 0 \end{cases} \quad (8)$$

Then for all neural network input $\zeta \in [\underline{\zeta}, \overline{\zeta}]$ *and* $i \in \{1, ..., \mathfrak{o}\}$*, we have:*

$$\mathcal{NN}(\zeta)_{[i,:]} \in [\mathcal{NN}(\underline{\psi}_{[i,:]} - \alpha_{[i,:]}(\underline{\psi}_{[i,:]} - \overline{\psi}_{[i,:]})), \mathcal{NN}(\overline{\psi}_{[i,:]} + \alpha_{[i,:]}(\underline{\psi}_{[i,:]} - \overline{\psi}_{[i,:]}))] \quad (9)$$

To implement the method in Proposition 2, we define the input vectors as $\overline{\zeta} = \text{center}(\Xi_i) + \frac{\delta}{2}$ and $\underline{\zeta} = \text{center}(\Xi_i) - \frac{\delta}{2}$. Additionally, we compute the bounds on the Jacobian matrix of the neural network $\mathcal{NN}_r$ to find the bounds $[\underline{J}, \overline{J}]$. Details about obtaining such bounds are given in Appendix in [15]. These bounds on the output of $\mathcal{NN}_r$ identifies which pixels are equal to zero for all the images generated by the states in each Ξ_i, which can be used to compute the filters in (5).

C. Design of Geometric Filters using Hough Transform

The spatial filters $\mathcal{S}^{\Xi_i}$ can focus attention on different regions of the state space. Although these filters provide a guarantee of the filter output that satisfies $\xi \in \Xi_i$ it does not provide any guarantee on the output of the filters for which $\xi \notin \Xi_i$. Therefore, it is necessary to augment the spatial filters with another filter that aims to detect whether the output follows the geometrical structure of the runway images. To achieve this, consider the following filters:

$$\mathcal{H}^{\Xi_i}(I) = \begin{cases} 1 & \text{if } \exists \xi \in \Xi_i \text{ such that } I = \mathcal{NN}_r(h(\xi)) \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Such filter can be efficiently computed using the classical Hough-space transformation [16]. In this transformation, a straight line is represented by a normal line that passes through the origin and is orthogonal to that straight line. The equation of the normal line is given by $\rho = \zeta_1 \cos(\theta) + \zeta_2 \sin\theta$, where ρ is the length of the normal line and θ is the angle between the normal line and the x-axis of the Pixel Coordinate Frame. By using the projections of the endpoints of the runway lines edges obtained from $h(\xi) = [\zeta_1, \zeta_2, \zeta_3, \zeta_4]$ as $P_1 = (\zeta_1, \zeta_2)$ and $P_2 = (\zeta_3, \zeta_4)$, we can solve for θ and ρ for the generated image as:

$$\theta = \tan^{-1}\left(\frac{\zeta_1 - \zeta_3}{\zeta_4 - \zeta_2}\right) \qquad \rho = \zeta_1 \cos(\theta) + \zeta_2 \sin(\theta) \quad (11)$$

Given a partition Ξ_i, we can obtain the range of ρ, θ for all runway images as follows. First, recall that each partition Ξ_i is an infinity ball with a radius equal to δ around a center point $\text{center}(\Xi_i) \in \mathbb{R}^4$. The two points $P_1 = (\text{center}(\Xi_i)_{[1]}, \text{center}(\Xi_i)_{[2]})$ and $P_2 = (\text{center}(\Xi_i)_{[3]}, \text{center}(\Xi_i)_{[4]})$ represent 2-dimensional points in the Pixel Coordinate Frame that corresponds to the center of Ξ_i (see Figure 5 for illustration). Following the 2-dimensional geometry of the Pixel Coordinate Frame, it is direct to show that:

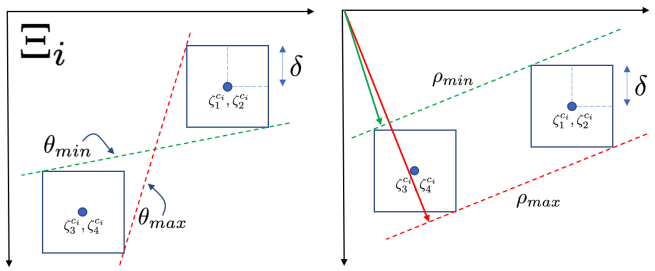

Fig. 5: Feasible range of angles and distances in Hough Space.

$$(\zeta_1^{c_i}, \zeta_2^{c_i}, \zeta_3^{c_i}, \zeta_4^{c_i}) = \text{center}(\Xi_i) \quad (12)$$

$$\theta^{\Xi_i}_{max} = \begin{cases} \tan^{-1}\left(\frac{\zeta_1^{c_i} - \zeta_3^{c_i} + 2\delta}{\zeta_4^{c_i} - \zeta_2^{c_i} + 2\delta}\right), & \text{if } \frac{\zeta_4^{c_i} - \zeta_2^{c_i}}{\zeta_3^{c_i} - \zeta_1^{c_i}} > 0 \\ \tan^{-1}\left(\frac{\zeta_1^{c_i} - \zeta_3^{c_i} + 2\delta}{\zeta_4^{c_i} - \zeta_2^{c_i} - 2\delta}\right), & \text{otherwise} \end{cases} \quad (13)$$

$$\theta^{\Xi_i}_{min} = \begin{cases} \tan^{-1}\left(\frac{\zeta_1^{c_i} - \zeta_3^{c_i} - 2\delta}{\zeta_4^{c_i} - \zeta_2^{c_i} - 2\delta}\right), & \text{if } \frac{\zeta_4^{c_i} - \zeta_2}{\zeta_3^{c_i} - \zeta_1^{c_i}} > 0 \\ \tan^{-1}\left(\frac{\zeta_1^{c_i} - \zeta_3^{c_i} - 2\delta}{\zeta_4^{c_i} - \zeta_2^{c_i} + 2\delta}\right), & \text{otherwise} \end{cases} \quad (14)$$

$$\rho^{\Xi_i}_{min} = \underline{b}_\delta \frac{\sqrt{1+m^2}}{m + \frac{1}{m}}, \quad \rho^{\Xi_i}_{max} = \overline{b}_\delta \frac{\sqrt{1+m^2}}{m + \frac{1}{m}} \quad (15)$$

where $m, \overline{b}_\delta, \underline{b}_\delta$ are defined in Appendix in [15].

Equations (12)-(15) define the reachable set of the runway images within the Hough space (the $\rho - \theta$ space). Moreover, the discretization introduced in the Pixel Coordinate Frame (the flooring operation in the camera model described on [3]) introduces a discretization over the range of ρ and θ computed by equations (12)-(15) which existing implementations of Hough transformation algorithms take into account. We denote by $\mathcal{L}^{\Xi_i} = \{(\rho^{\Xi_i}_{max}, \theta^{\Xi_i}_{\max}), \ldots, (\rho^{\Xi_i}_{min}, \theta^{\Xi_i}_{min})\}$ the discrete set of the allowable values of ρ and θ within the partition Ξ_i. For each possible (ρ_j, θ_j) in $\mathcal{L}^{\Xi_i}$, we define the filter $\mathcal{R}^{\Xi_i}(\rho_j, \theta_j) \in \mathbb{B}^{a \times b}$ as:

$$\mathcal{R}^{\Xi_i}(\rho_j, \theta_j)_{[k,l]} = \begin{cases} 1 & \text{if } l - 1 < -\frac{\cos\theta_j}{\sin\theta_j} k + \frac{\rho_j}{\sin\theta_j} < l \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

For each filter $\mathcal{R}^{\Xi_i}(\rho_j, \theta_j)$ we can define a mismatching score that computes how far the input image I is from the expected output of this filter as:

$$\mathcal{M}(I, \mathcal{R}^{\Xi_i}(\rho_j, \theta_j)) = \left\| -I \oplus \mathcal{R}^{\Xi_i}(\rho_j, \theta_j) \right\|_1 \quad (17)$$

That is, $\mathcal{M}$ is equal to zero whenever the input image I matches exactly the line represented by $\mathcal{R}^{\Xi_i}(\rho_j, \theta_j)$ and non-zero otherwise. Finally, we can implement the filter $\mathcal{H}^{\Xi_i}$ in (10) as:

$$\mathcal{H}^{\Xi_i}(I) = \begin{cases} 1 & \text{if } \arg\min\{\mathcal{M}(I, \mathcal{R}^{\Xi_i}(\rho^{\Xi_i}_{max}, \theta^{\Xi_i}_{max})), \ldots, \\ & \qquad \mathcal{M}(I, \mathcal{R}^{\Xi_i}(\rho^{\Xi_i}_{min}, \theta^{\Xi_i}_{min}))\} = 0 \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

In other words, the filter $\mathcal{H}^{\Xi_i}$ produces 1 whenever any of the filters $\mathcal{R}^{\Xi_i}(\rho^{\Xi_i}_{max}, \theta^{\Xi_i}_{max}), \ldots, \mathcal{R}^{\Xi_i}(\rho^{\Xi_i}_{min}, \theta^{\Xi_i}_{min})$ were able to match its input image. The following proposition follows directly from the definition of $\mathcal{H}^{\Xi_i}(I)$ above.

Proposition 3. *Consider the filter $\mathcal{H}^{\Xi_i}$ defined in (18). The following holds:*

$$\mathcal{H}^{\Xi_i}(I) = 1 \iff \exists \xi \in \Xi_i \text{ such that } I = \mathcal{NN}_r(h(\xi)) \quad (19)$$

D. Design of the NN filter $\mathcal{NN}_{\mathcal{F}}$

The final filter $\mathcal{NN}_F$ consists of processing the images I using all the spatial filters $\mathcal{S}^{\Xi_1}, \ldots, \mathcal{S}^{\Xi_l}$ followed by the geometric filters $\mathcal{H}^{\Xi_1}, \ldots, \mathcal{H}^{\Xi_l}$. Finally, the filter $\mathcal{NN}_F$ identifies the partition $\hat{\Xi}$ for which the geometric filter returns 1 to produce its final outputs as follows:

$$\hat{\Xi} = \arg\max\{\mathcal{H}^{\Xi_1}(I \otimes \mathcal{S}^{\Xi_1}), \ldots, \mathcal{H}^{\Xi_l}(I \otimes \mathcal{S}^{\Xi_l})\} \quad (20)$$

$$\hat{I}_r = I \otimes \mathcal{S}^{\hat{\Xi}} \quad (21)$$

The following result captures the correctness of the $\mathcal{NN}_F$.

Theorem 1. *Consider a noisy image $I(\xi) = I_r(\xi) + I_n(\xi)$, a partitioning of the state space Ξ into infinity balls of radius δ namely $\Xi_1, \ldots, \Xi_l$. Denote by Ξ^* the partition for which the aircraft state ξ belongs, i.e., $h(\xi) \in \Xi^*$. Under the following assumptions:*

$$(i) I_n(\xi) \notin \{\mathcal{NN}_r(h(\xi)) \mid h(\xi) \in \Xi\} \quad (22)$$

$$(ii) \forall \xi \in \Xi^*. [I_n(\xi) \otimes \mathcal{NN}_r(h(\xi)) = \mathbf{0}_{a,b}] \quad (23)$$

then the following holds:

$$\hat{\Xi} = \Xi^* \quad (24)$$

$$\hat{I}_r = I_r(\xi) \quad (25)$$

$$\|\xi - \hat{\xi}\| \leq 4L_h\delta \qquad \forall \hat{\xi} \in \hat{\Xi} \quad (26)$$

where $(\hat{\Xi}, \hat{I}_r) = \mathcal{NN}_F(I(\xi))$ and L_h is the Lipschitz constant of h^{-1}.

Proof. We start by proving (24) as follows. For the sake of contradiction, we assume that there exists a partition $\Xi^\dagger \neq \hat{\Xi}$ such that which the aircraft state ξ satisfies $h(\xi) \in \Xi^\dagger$. It follows from Proposition 1 and assumptions (22) and (23) that:

$$I \otimes \mathcal{S}^{\hat{\Xi}} \neq I_r(\xi), \qquad I \otimes \mathcal{S}^{\Xi^\dagger} = I_r(\xi)$$

and hence Proposition 3 entails that:

$$\mathcal{H}^{\hat{\Xi}}(I \otimes \mathcal{S}^{\hat{\Xi}}) = 0, \qquad \mathcal{H}^{\Xi^\dagger}(I \otimes \mathcal{S}^{\Xi^\dagger}) = 1$$

Nevertheless, this contradicts the fact that:

$$\hat{\Xi} = \arg\max\{\ldots, \mathcal{H}^{\hat{\Xi}}(I \otimes \mathcal{S}^{\hat{\Xi}}), \ldots, \mathcal{H}^{\Xi^\dagger}(I \otimes \mathcal{S}^{\Xi^\dagger}), \ldots\}$$

which proves that $h(\xi) \in \hat{\Xi}$.

Equation (25) follows directly from (24) and Proposition 1. Similarly, equation (26) follows from the fact that the partition $\hat{\Xi}$ is an infinity ball of radius δ and hence for any $\hat{\xi} \in \hat{\Xi}$:

$$\begin{aligned}\|h(\xi) - h(\hat{\xi})\|_\infty &= \|h(\xi) + \text{center}(\hat{\Xi}) - \text{center}(\hat{\Xi}) - h(\hat{\xi})\|_\infty \\ &\leq \|h(\xi) - \text{center}(\hat{\Xi})\|_\infty + \|\text{center}(\hat{\Xi}) - h(\hat{\xi})\|_\infty \\ &\leq 2\delta\end{aligned}$$

Hence from the relation between the 2-norm and the infinity norm, we conclude that:

$$\|h(\xi) - h(\hat{\xi})\| \leq \sqrt{4}\|h(\xi) - h(\hat{\xi})\|_\infty \leq 4\delta$$

from which we conclude that $\|\xi - \hat{\xi}\| \leq 4L_h\delta$ which concludes the proof. □

Before we conclude this section, it is essential to interpret the assumptions (22) and (23) in Theorem 1. In particular, the assumption in (22) entails that the noise I_n can not be generated using the runway generative model $\mathcal{NN}_r$. In other words, this assumption ensures that the noise does not look like a runway and hence only one image of the runway exists in the scene. The assumption in (23) asks that the pixels that are δ close to the runway are not affected by the noise. It is crucial to note that assumption (23) is required to be satisfied in Ξ^* only and does not affect other partitions.

IV. Experimental Evaluation

We present the results of a vision-based aircraft landing system that uses a target runway. We consider two runway segments, $L = [(L_x, 0, L_z), (L_x, 0, L_z + r_l)]$ and $R = [(R_x, 0, R_z), (R_x, 0, R_z + r_l)]$ where $R_x = 0.1$, $L_x = -0.1$, $R_z = 0$, $L_z = 0$, $r_l = 0.3$ (in meters).

To generate monochromatic images, we utilized the SilkyEvCam event-based camera with a resolution of 640×480 pixels, a focal length of 8 mm, and a pixel size of $15\ \mu m \times 15\ \mu m$. We measured the ground-truth states of the vehicle using Vicon motion capture cameras to track optical markers attached to the camera envelope, and the centroid of the camera was defined as the camera coordinate frame (CCF) origin. Similarly, we defined the runway target as the runway coordinate frame (RCF) from which all measurements were made.

We partitioned the state space $\Xi \subset \mathbb{R}^4$ into 27 regions $\Xi_1, \ldots, \Xi_{27}$ using a partitioning parameter $\delta = 0.1$. These regions correspond to the range of states $[\xi_y \times \xi_z \times \xi_\theta] = [0.8, 1] \times [1.6, 1.8] \times [0.5, 0.7]$ (we fix $\xi_x = 0$ in our experiments). We then implemented the runway generative model neural network $\mathcal{NN}_r$ for a resolution of 640×480 pixel images, the filter $\mathcal{NN}_f$, and the corresponding application of the spatial $\mathcal{S}^{\Xi_i}$ and geometric filters $\mathcal{H}^{\Xi_i}$ on all partitions to create the binary weights needed using PyTorch libraries. This process took approximately 20 minutes per partition, resulting in a total of approximately 9 hours to generate the neural network weights for all 27 partitions using an Apple M1 Pro processor with 32 GB of RAM.

Next, the filter $\mathcal{NN}_F$ was used to process images collected from the SilkyEvCam event-based camera. We operated the camera for several minutes resulting in a total of 1320 images using 25 frames per second. Figure 6 and Figure 7 show two instances of the images collected and processed during our experiments. As seen from the two figures, the scene contains one runway and several objects, and noisy pixels. The neural network $\mathcal{NN}_F$ is used to filter these images and remove all objects except for the runway. Figure 6 (right) and Figure 7 (right) show the outputs of the 4 different spatial filters $\mathcal{S}^{\Xi_i}$. As can be observed in the two figures, the result of these filters focuses the attention on specific segments of the scene. Some of these filtered images contain the runway (or segments of it) while others contain only parts of the noise image I_n. Next, we execute the geometric filters $\mathcal{H}^{\Xi_i}$ to identify the images that match the geometric structure of the runways. We highlight the partition with the smallest mismatch score $\mathcal{M}$ with a green box in Figure 6 and Figure 7. In particular, in Figure 6, the output corresponding to partition 1 contains leads to the smallest mismatch score while partition 24 corresponds to the one with the smallest mismatch score in Figure 7.

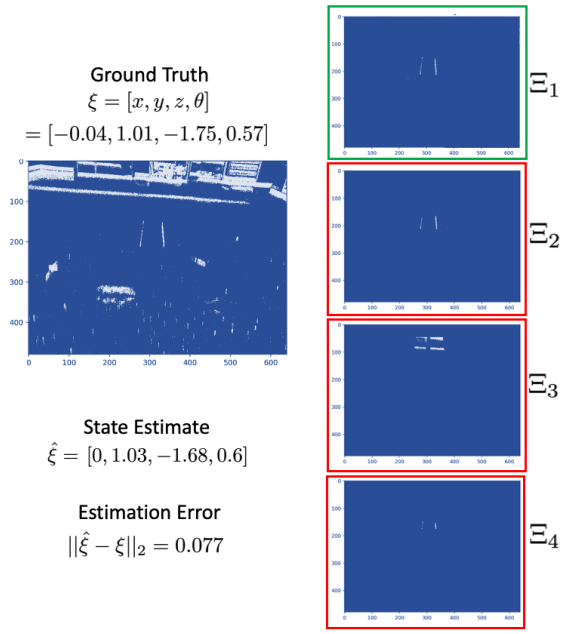

Fig. 6: Test 1: Framework application on image #1 delivers correct filtered runway (in Green) found on Partition #1.

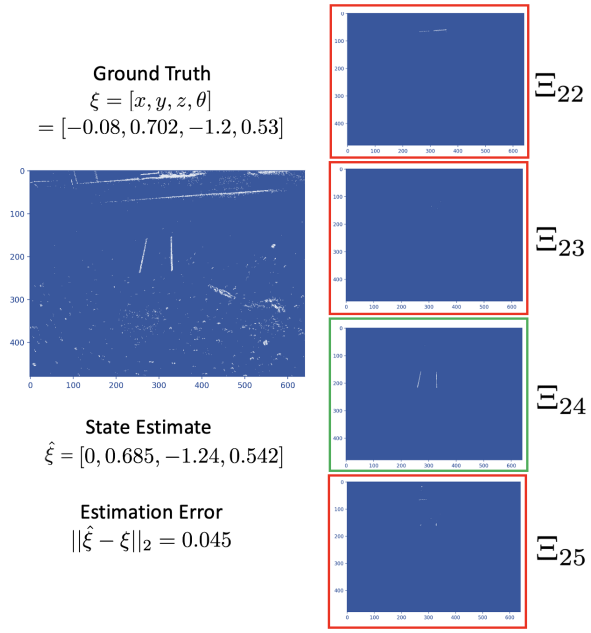

Fig. 7: Test 2: Framework application on image #2 delivers correct filtered runway (in Green) found on Partition #24.

Finally, we used off-the-shelf algorithms to process the filtered image and produce the final state estimate. For the test reported in Figure 6, the resulting state error is 0.0777 while for the test reported in Figure 7 the resulting error is 0.045, both are below the threshold of $4\delta L_h$ and hence no further processing is required. Additionally, for comparison purposes, we applied an off-the-shelf standard Hough transformation-based filter that can discover line segments in the scene with the aim of identifying the runway without our proposed filter. Figure 8 shows the output of the standard Hough transformation-based filter when operated on the same input image used in Figure 6. The dashed lines in Figure 8 correspond to the line segments that were detected

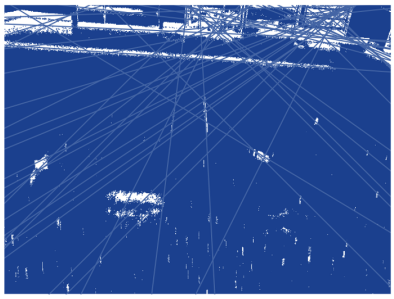

Fig. 8: Filtering using only Hough filter without geometrical constraints.

by the standard filter. As can be appreciated from Figure 8, the standard filter leads to several false line detections that do not match the runway due to the noise and the other objects in the scene. Fortunately, our proposed filter does not suffer from such an issue and comes with provable guarantees.

REFERENCES

[1] J. Ferlez, M. Elnaggar, Y. Shoukry, and C. Fleming, "Shieldnn: A provably safe nn filter for unsafe nn controllers," *arXiv preprint arXiv:2006.09564*, 2020.

[2] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[3] U. Santa Cruz and Y. Shoukry, "Nnlander-verif: A neural network formal verification framework for vision-based autonomous aircraft landing." NASA Formal Methods. NFM 2022. Lecture Notes in Computer Science, vol 13260. Springer, pp 213–230, 2022.

[4] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems." HSCC 19: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp 147–156, 2019.

[5] C. Hsieh, Y. Li, D. Sun, K. Joshi, S. Misailovic, and S. Mitra, "Verifying controllers with vision-based perception using safe approximate abstractions." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, 2022.

[6] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *Journal of Aerospace Information Systems*, vol. 19, no. 9, pp. 574–584, 2022.

[7] P. Habeeb, N. Deka, D. D'Souza, K. Lodaya, and P. Prabhakar, "Verification of camera-based autonomous systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[8] R. Talak, L. Peng, and L. Carlone, "Certifiable 3d object pose estimation: Foundations, learning models, and self-training," 2023.

[9] S. Shoouri, S. Jalili, J. Xu, I. Gallagher, Y. Zhang, J. Wilhelm, J.-B. Jeannin, and N. Ozay, "Falsification of a vision-based automatic landing system," in *AIAA Scitech 2021 Forum*, p. 0998, 2021.

[10] 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 *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I*, pp. 3–17, Springer, 2020.

[11] J. Ferlez and Y. Shoukry, "Polynomial-time reachability for lti systems with two-level lattice neural network controllers," *IEEE Control Systems Letters*, 2022.

[12] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings 3*, pp. 670–686, Springer, 2019.

[13] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*, vol. 26. Springer Science & Business Media, 2012.

[14] P.-J. Meyer, "Reachability analysis of neural networks using mixed monotonicity." IEEE Control Systems Letters,vol. 6, pp. 3068-3073, 2022.

[15] U. Santa Cruz and Y. Shoukry, "Certified vision-based state estimation for autonomous landing systems using reachability analysis." arXiv preprint arXiv: 2309.05167, 2023.

[16] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science, Springer, 2022.