IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Robust Filtering based on Complex Cell Networks from the Visual Cortex

Mehdi Kermanshah[1*], Nguyen Nguyen[2*], Calin Belta[1] and Roberto Tron[1]

***Abstract*— We propose a novel approach for filtering that is inspired by Complex Cell Networks (CCN) in the primary visual cortex of mammals; our aim is to emulate the robustness of the biological system, showing graceful degradation in face of gross deterioration of the input. Instead of relying on energy minimization as in frequency-based filter design, or on Bayes' theorem as in statistical filtering, our formulation is founded on three principles that have been observed in real neural responses: *1) winner-take-all*, where perceptual ambiguity is solved by focusing on the strongest signal; *2) persistence*, where information is fused across time to lessen the impact of noise, outliers, and temporary cancellations in the input data; and *3) boundedness*, where the responses in the filter are bounded to be non-negative and below a maximum value. In neuroscience, the typical goal is to find models that match and explain measurements from a biological system. In this paper, we take an engineering approach, where we encode the three properties above as mathematical constraints, and find filter parameters that guarantee convergence of the filter (for constant, bounded inputs), optimize bounds on the convergence rate, and improve sparsity of the filter kernel; overall, the filter is obtained from the solution to a Linear Program (LP). As a proof-of-concept, we integrated the proposed filter architecture with a neural network to estimate the vehicle speed solely based on camera images in extremely noisy environments.**

I. INTRODUCTION

Fitering, in broad terms, aims to improve the reliability and accuracy of signals recovered from noisy measurements. It is ubiquitous in controls [13], localization [25], robot vision [7], signal processing [5], and finance [11] (to name just a few). A typical example is to estimate information about the state of a vehicle (e.g., the speed of a self-driving car) based on an input signal obtained from a sensor (e.g., a video from a front-mounted camera). In real-world situations, the input is corrupted, sometimes severely, by noise and hard-to-model exogenous disturbances (e.g., bad weather or dust in front of the camera). The goal of filtering is to extract an output signal that is as close as possible to the ground truth.

Filtering algorithms can be roughly divided into *stochastic Bayesian-based* and *deterministic frequency-based* methods. Stochastic Bayesian-based methods [8], [23] are the most common in robotics. In this approach, each signal is represented as a Markov random process, and estimation is performed by predicting its distribution and updating it with measurements at each time step. For linear-Gaussian systems, the optimal solution is given by the Kalman Filter (KF) [16]. For nonlinear systems, popular approaches are the Extended Kalman Filter (EKF) and the Unscented Kalman Filter [15], which, however, give only approximate solutions to the underlying complete Bayesian problem. Kalman Filtering methods and their extensions have been widely applied, e.g., to pose tracking, visual odometry, and Simultaneous Localization And Mapping (SLAM) [10], [21]. Their fundamental limitation is the assumption that the noise has always a Gaussian distribution, which makes them ill-suited to deal with outliers and other gross disturbances of the input signal. Particle filters [14] represent arbitrary distributions of estimated variables as mixtures of Gaussian random variables (particles). Compared to the KF and its extensions, particle filters can give approximate solutions for nonlinear non-Gaussian systems, but have significantly larger computational costs that limit their scalability.

Frequency-based methods assume that the noise can be separated from the main signal and removed based on differences in the frequency representations. In this field, the *1€* filter [6] is a low-pass filter that reduces lag and jitter by varying the cutoff frequency based on the rate of change of the input signal. The authors of [17] proposed a double exponential smoothing-based method (DESP) for tracking signals, which proposes a linear filter based on the approximation of the system as a second-order integrator. In this particular line of work, the focus is on minimizing the computation overhead (with respect to the KF, which is already considered fast for stochastic approaches). However, similarly to the KF, these approaches are sensitive to outliers.

While the primary focus of the paper is on a novel filter formulation, we do include a proof-of-concept test on a velocity estimation task from monocular images. We estimate the vehicle speed using a convolutional neural network and then apply our filter to improve reliability. For this problem, Visual Odometry (VO) and Visual Inertial Odometry (VIO) algorithms are standard solutions. These methods provide accurate results in general. However, they can be brittle and completely fail when the stream of images is corrupted.

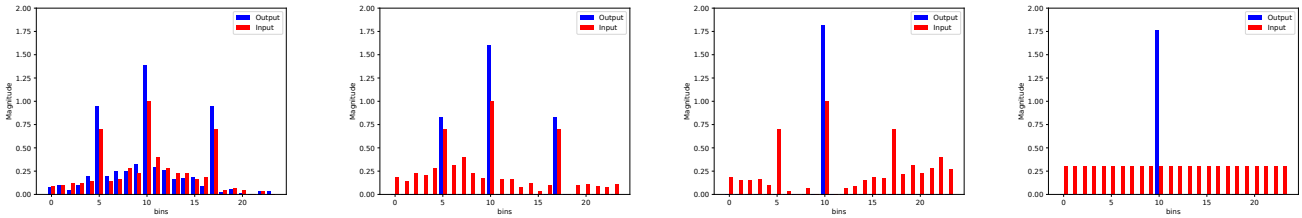
More broadly, our work is part of the area of bioinspired algorithms, which have been used in several domains, including SLAM [20], controls [18] and path planning [19]. The most relevant work to ours is [24], in which the dynamic model of neurons with manually tuned gains is exploited as a low-pass filter in order to smooth the jumps of a backstepping tracking controller. Compared to this category of works, our

This work was supported by ONR MURI N00014-19-1-2571 "Neuro-Autonomy: Neuroscience-Inspired Perception, Navigation, and Spatial Awareness"

[1]Mehdi Kermanshah, Calin Belta and Roberto Tron are with the Department of Mechanical Engineering, Boston University, 730 Commonwealth Ave, MA 02215, United States {mker,cbelta,tron}@bu.edu

[2]Nguyen Nguyen is with the Department of System Engineering, Boston University, 730 Commonwealth Ave, MA 02215, United States nguyenpn@bu.edu

[*]These authors contributed equally to this work



(a) (b) (c) (d)

Fig. 1: Desired properties and progression of the filter for a simple constant input signal. The input and output signals(histograms) are plotted as red and blue bars, respectively. 1a-1c show the progression of the filter. 1a is the output of the filter initially , 1b is an example in the transition phase and 1c is the output of the filter after convergence. 1c demonstrates (P1). Although the input contains multiple modes, only the global maximum is amplified. 1c and 1d are two sequential frames, illustrating (P2) behavior . While the input signal is flattened immediately, the filter still holds the position of the last peak due to its short-time memory.

paper is the only one focusing on filtering with principled guarantees on the behavior of the filter.

Regarding to work in neuroscience, this paper builds on the observed behavior of complex cell layers in the primary visual cortex of mammals. A standard model for this behavior was first given in [4], which proposed a histogram-based filter with two layers: a first layer with a bank of tuned-response neurons that respond to a specific narrow range of the input signal (called stimulus in neuroscience), and the second layer with recursive neurons that implement a linear convolution (with a $\cos(\cdot)$-shaped kernel) followed by a rectifying nonlinearity (ReLU function). This model was proposed to capture three important properties of real biological systems:

(P1) *Winner takes all:* If the input signal has multiple local maxima, the model amplifies the part of the signal in correspondence to the global one (see Figure 1c for an illustration); this selectivity can be achieved only with some type of nonlinearity in the model.

(P2) *Sustained activity:* The model include a form of short-term memory that allows it to compensate for significant but short-lived input disturbances (e.g., temporary occlusions of the stimulus, see Figure 1d for an example); this is implemented through the recursive connections of the neurons in the second layer.

(P3) *Boundedness:* The neuron responses are always positive and bounded; this implies a form of stability, since this property prevents unbounded responses.

The model given in [4], while being able to qualitatively capture behaviors (P1)-(P3), has several shortcomings (see also Section III-B) First, it uses a cosine kernel, which is a hand-crafted choice, and does not appear to be optimal under any particular criterion. Second, the kernel and additional parameters of the filter need to be manually tuned in order to obtain stability (for which [4] does not give formal results); in fact, this model does not enforce an upper limit on the output signal. Third, the proposed kernel has support over the entire domain of the input histogram, which implies direct connections between every pair of neurons in the second layer; this requires more computations, and is not biologically realistic.

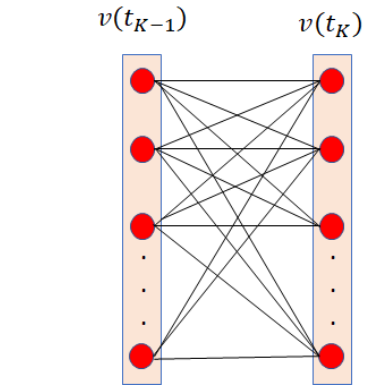
Finally, the LP formulation of our problem is reminiscent of the work in [3]; however, in that case it was used for synthesizing controllers for navigating in an environment, while here we consider different sets of constraints derived from the filtering problem.

Paper contributions. We propose a novel filtering technique that generalizes the filter architecture of [4], but formally encodes properties (P1)-(P3) into a Linear Program (LP), whose solution provides all the filter coefficients. In addition to providing some theoretical guarantees on the behavior of the filter, we show through our proof-of-concept that, by mimicking the properties of biological systems, our filter shows promising robustness and computational properties where traditional solutions from robotics can be fragile.

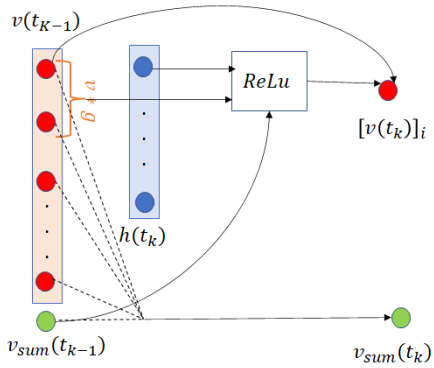
Our main contributions can be summarized as:

- We introduce a neuro-inspired filter that reduces the effect of outliers and noise by following the global maximum of a signal in real time with minimal processing.
- We provide a more systematic method to model a complex cell network in primary visual cortex, and find the model parameters through a linear optimization problem with linear and Control Barrier Function (CBF) constraints to guarantee the desired properties.
- In a proof-of-concept application, we use our filter to track a vehicle's neural network-estimated velocity. This neural network estimates vehicle speed from images streamed from a mounted camera. Our filter improves the overall performance in the presence of outliers that are in general challenging for standard neural network approaches; this is true, in particular, when some frames are intermittently and heavily corrupted, thanks to the short memory built in the filter. As a result, our filter allows to estimate the speed of a vehicle more robustly than existing visual odometry (VO) approaches.

Paper outline. The remainder of the paper is organized as follows. Section II, contains our notation and preliminaries. We formulate the desired properties of the filter as linear constraints in section III. Section IV introduces our filter



(a)



(b)

Fig. 2: Comparing our proposal architecture (2b)of complex layer with the prior one(2a). By introducing V_{sum} neuron the number of connection is greatly reduced since neurons are connected more locally.

architecture and our design approach based on a Linear Program. Section V analyzes the non-linearity of the proposed filter and presents results on boundedness and stability of the non-linear model. Finally, Section VI demonstrates the application of our filter to a VO task as a proof of concept evaluation.

II. Preliminaries and Problem Formulation

A. Definitions and notation

We use $[v]_i$ to indicate the i-th element (bin) of a vector v and $\mathcal{J}$ for a set of indices with cardinality n. We use e_j to denote a unit vector with all the entries equal to zero except for $[e_j]_j = 1$. A Rectified Linear Unit function [1] is denoted by ReLU. We use $\mathbf{1}$ to denote the vector of all ones. We consider a 1-D continuous discrete-time random process $x \in \mathcal{I}$ and a partition of the domain, $\mathcal{I} = \bigcup_{i=1}^{n} \mathcal{I}_i$, where each $\mathcal{I}_i$ is a disjoint interval.

Definition 1: A histogram hist of x is defined as a non-normalized probability mass function over the elements of $\mathcal{I}_i$ such that $[\mathrm{hist}(x)]_i$ is proportional to the probability of x falling in the i-th element $\mathcal{I}_i$.

Example 1: In our proof of concept case study, we cast the problem of estimating the speed (i.e., magnitude of the velocity) x of a camera mounted on a vehicle as a classification problem. The elements of the partition $\mathcal{I}_i$ represent different bins for the value of x. Our goal is to find the weights of a neural network that outputs a vector $h \in \mathbb{R}^n$ as a prediction for $\mathrm{hist}(x)$.

The following definitions are used to simplify the discussion of the constraints for our Linear Programming solution.

Definition 2: The *mode index* of a histogram v is defined as $i_{\mathfrak{m}} = \mathrm{argmax}_i [v]_i$; the *mode* $\mathfrak{m}(h)$ of a histogram is defined as the center of the interval $\mathcal{I}_{i_{\mathfrak{m}}}$. The *margin* of the mode is given by $[h]_{i_{\mathfrak{m}}} - \max_{i \neq i_{\mathfrak{m}}} [h]_i$.

Definition 3: Given two vectors $g \in \mathbb{R}^{d_1}$, $v \in \mathbb{R}^{d_2}$, with $d_2 > d_1$, the *same-size* convolution $g * v$ is defined as

$$[v * g]_k = \sum_{1 \leq j \leq d_2} [g]_j [v]_{k-j+1}, \; k \in \{1, \ldots, d_2\}. \quad (1)$$

Fact 1: There exists a matrix $A_k \in \mathbb{R}^{d_1, d_2}$ such that

$$[v * g]_k = v^{\mathrm{T}} A_k g \quad (2)$$

Proof: The matrix A_k has 1's in correspondence of the terms in the summation (1), and zeros elsewhere. ■

B. Zeroing Control Barrier Functions for Linear Constraints

In this paper we will derive dynamic constraints based on Zeroing Control Barrier Functions (ZCBFs) [2]. We review the theory below, but restricted to the specific case of linear constraints (although ZCBFs can generally be applied also to nonlinear differentiable constraints). For the sake of brevity, we define subsets of $\mathbb{R}$ based on simple linear inequalities of the form $\mathcal{C}_{\leq M} = \{x : x \leq M\}$, and $\mathcal{C}_{\geq M} = \{x : x \geq M\}$, where $x \in \mathbb{R}, M \in \mathbb{R}$. The following proposition provides a condition on a state $x \in \mathbb{R}$ that makes these sets forward invariant.

Proposition 1: Given any constant $c > 0$ and $x(0) = x_0 \in \mathcal{C}_{<M}$ (respectively, $x_0 \in \mathcal{C}_{>M}$), if $\dot{x}$ satisfies

$$\begin{gathered} \dot{x} + c(x - M) \leq 0 \\ \text{(respectively, } \dot{x} + c(x - M) \geq 0\text{)}, \end{gathered} \quad (3)$$

for all $t > 0$, then $x(t) \in \mathcal{C}_{<M}$ (respectively, $x(t) \in \mathcal{C}_{>M}$) for all $t > 0$.

Proof: The claim is a corollary of [2, Proposition 3]. ■

III. Problem Formulation

For our problem we consider a continuous time random process $x(t) \in \mathbb{R}$, for which we assume that we observe an histogram $h = \mathrm{hist}(x) + \varepsilon \in \mathbb{R}^n$, where ε is a stochastic process that can have a multimodal distribution, contain outliers, or cancel x completely for brief periods of time.

Example 1 (continued): Images from the camera can occasionally be blurry, or have obstructed regions due to weather conditions or large objects moving within the field of view of the camera. This might cause the large deviations ε in the observed histogram h.

Our goal is to produce a sequence of histograms $v(t)$ which is a filtered version of h such that $\mathfrak{m}(v)$ is close to x.

A. From Properties to Linear Constraints

We now revisit properties (P1)-(P3) to formally define them as linear constraints on v and $\dot{v}$.

(P1) implies that, over time, $[v]_{i_m}$ must grow faster than other bins, i.e.,

$$[\dot{v}]_{i_m} - [\dot{v}]_j \in \mathcal{C}_{\geq 0},\ \forall j \in \mathcal{J}, j \neq i_{\mathfrak{m}}(h). \tag{4}$$

(P2) requires that v is produced by some process that has memory of past values, i.e., by a filter

$$\dot{v} = f(h, v; g), \tag{5}$$

where g is a vector of parameters. The specific form of the function f is very important to achieve stability and other properties and is discussed in more detail below.

(P3) means that the values in v must be bounded from below by zero, and from above by a (user-defined) constant M_v, i.e.,

$$[v(t)]_i \in \mathcal{C}_{\geq 0} \cap \mathcal{C}_{\leq M_v}\ \forall i \in \mathcal{J}. \tag{6}$$

Additionally, we will focus on designing a filter that satisfies all the properties (P1)-(P3) for all inputs h that have the following properties: First, they are bounded as :

$$[h]_i \in \mathcal{C}_{\geq 0} \cap \mathcal{C}_{\leq M_h} \quad \forall i \in \mathcal{J}, j \neq i_{\mathfrak{m}}(h). \tag{7}$$

Second, $[h]_{i_m}$ is larger than other bins by at least δ, which can be expressed as:

$$[h]_{i_m} - [h]_i \in \mathcal{C}_{\geq \delta} \quad \forall i \in \mathcal{J}, j \neq i_{\mathfrak{m}}(h), \tag{8}$$

where $\delta > 0$ is an arbitrary small number. The only purpose of constraint (8) is to exclude cases where we have two or more modes that are exactly the same.

B. Existing model from neuroscience

Numerous studies have been conducted on the primary visual cortex to mathematically model its responses [9]. According to these studies, in order to determine the direction of movement, the visual information is first processed by layers of simple cells (see Figure 3a for a schematic representation of the overall process). In these simple layers, each neuron is stimulated by (i.e., *tuned* to) the motion in a specific direction. The output of this layer is often quite noisy. For instance, in our example, multiple apparent motions can appear in different directions can exist in different parts of the image. The layer of simple cells is connected to a layer of complex cells, which can robustly determine the direction of movements by selectively amplifying the input with the strongest response. Figure 1c shows an example. The following dynamic model was proposed in [9] to capture the behaviors of complex cells:

$$[\dot{v}]_i = -v_i + \mathrm{ReLU}([h]_i + v^{\mathrm{T}} \tilde{A}_i g_0), \tag{9}$$

where the kernel g_0 is defined as $[g_0]_i = -\lambda_0 + \lambda_1 cos(2i)$, with λ_0, λ_1 denoting tuning parameters. Note that in [9], the domain $\mathcal{I}$ of the histogram represents a view-angle θ in the range $[-\pi/2, \pi/2]$, and the matrices $\tilde{A}_i$ (which are analogous to the matrices A_i defined in Fact 1) implement a cyclic convolution. While this model explicitly captures (P2) and (P1), it does not completely enforce (P3). More specifically, it does not enforce an upper bound (6), whereas the response of a real neurons is always bounded.

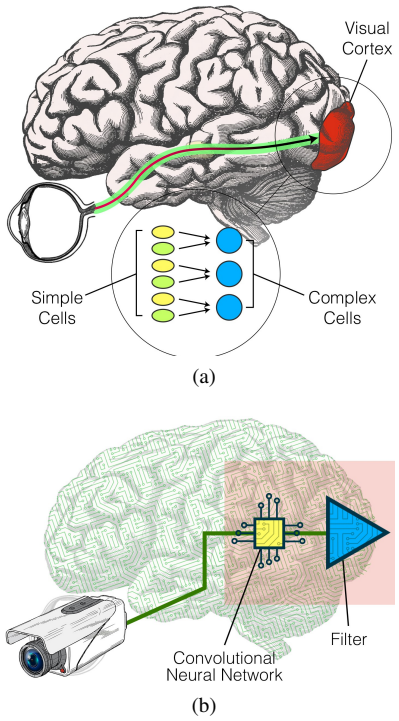


Fig. 3: High-level system comparison between the biological system (3a) and our version (3b . In the biological system, the visual information is initially processed by simple cells, whose output can have multi-excitation at each instance. These cells are connected to complex cells, and they eliminate the ambiguity of the input signal due to recurrent connections, resulting in a short-time memory. In our case study, we replicate this system. First, a convolutional neural network (CNN) estimates the speed of the camera based on images. Since it does not have any recurrent connections, it is prone to noise. We add our filter to this pipeline to enhance the robustness of the estimation by fusing the output of the CNN through time.

IV. Filter Design

In order to achieve (P1)-(P3), we start from the architecture of the previous model (9), but with two important differences. First, we introduce an additional state in the filter that approximately tracks the sum of all the bins; this allow the kernel to be sparse. Second, instead of manually specifying the coefficients g, we obtain them from the solution of a LP whose constraints explicitly enforce (4)-(6).

A. Improving the sparsity of the filter

Intuitively, to enforce (P1), the update at one bin needs to depend on the values of all other bins; a straightforward implementation (as in the model (9)) leads to all-to-all

connections between neurons (see Figure 2a for a pictorial representation). As mentioned in the introduction, this not only implies computations in the order of n^2, but also is not biologically realistic. Instead, we propose to split the kernel g that appears in (9) into two parts: a kernel g_0 that has a small support, plus a term that depends on the average $\frac{1}{n}\mathbf{1}^{\mathrm{T}}v$. For the latter, to avoid all-to-all connections, we introduce a new state (neuron) v_{sum} that approximately tracks the sum $\mathbf{1}^{\mathrm{T}}v$. Consequently, the update for one neuron requires only the values of a small number of nearby neurons, and the value of the common neuron v_{sum}*, significantly reducing connections (see Figure 2b for a pictorial representation). This leads to the following model for our proposed filter:

$$[\dot{v}]_i = -[v]_i + \mathrm{ReLU}([h]_i + v^{\mathrm{T}}A_i g_0 - g_1 v_{sum}), \quad (10a)$$

$$\dot{v}_{sum} = g_2\left(\left(\sum_{i\in I}[v]_i\right) - v_{sum}\right), \quad (10b)$$

where g_0 is the local kernel, and g_1 and g_2 are scalars.

In model (10), it is necessary to specify the filter kernel g_0 and the gains g_1, g_2 such that properties (4)-(8) and (11) are satisfied. Furthermore, based on its definition, v_{sum} should closely track the sum over v with a reasonable error b. We express this constraint using the following:

$$v_{sum}(t) - \mathbf{1}^{\mathrm{T}}v(t) \in \mathcal{C}_{\geq -b} \cap \mathcal{C}_{\leq b} \quad \forall t. \quad (11)$$

B. Filter design

In this section we show how the filter coefficients g_0, g_1, g_2 can be obtained from the solution of a Linear Program. For convenience, we define the following shorthand notation:

$$g = \begin{bmatrix} g_0 & g_1 & g_2 \end{bmatrix}^{\mathrm{T}} \quad z = \begin{bmatrix} h & v & v_{sum} \end{bmatrix}^{\mathrm{T}} \quad (12)$$

In order to proceed, we ignore the nonlinearity in the dynamics (10), and substitute the ReLU with its argument. As a result, the solution to the LP will guarantee that all constraints and properties are satisfied on a subset of the joint state-input space where all the ReLU's are not active (i.e., their arguments are positive). We discuss the extension of these guarantees to the entire state space in Section V.

As discussed in detail below, constraints (4), (6) and (11) can be applied to our proposed model (10); in all cases, we obtain constraints that are bilinear in the coefficients g and the state x. While this is straightforward for (4), constraints (6) and (11) are not as trivial because they apply to the states v and v_{sum}; therefore, we will use the ZCBF framework reviewed in section II-B to obtain constraints where the coefficients g appear explicitly.

1) Winner-takes-all constraint: Substituting the dynamics (10) (ignoring the ReLU) into constraint (4), we have:

$$-([v]_{i_{\mathrm{m}}} - [v]_i) + ([h]_{i_{\mathrm{m}}} - [h]_i) + v^{\mathrm{T}}(A_{i_{\mathrm{m}}} - A_i)g_0 \geq 0; \quad (13)$$

*The update of the single neuron v_{sum} still needs access to the values of all other neurons. We believe that our approach could be extended to the case where v_{sum} is computed by a sub-network of neurons using linear consensus [22] at the expense of more computations and slower convergence.

for all $i \in \mathcal{J}$; expanding, and using the standard basis vectors $\mathbf{e}_i$ to pick individual entries in the state z we rewrite (13) as

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ A_i - A_{i_{\mathrm{m}}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} g + \begin{bmatrix} \mathbf{e}_i - \mathbf{e}_{i_{\mathrm{m}}} \\ \mathbf{e}_{i_{\mathrm{m}}} - \mathbf{e}_i \\ 0 \end{bmatrix}\right)^{\mathrm{T}} z \leq 0. \quad (14)$$

2) Constraint on $v(t)$: Based on Proposition 1, we construct the following linear ZCBF constraint to enforce the upper bound $v(t) \in \mathcal{C}_{\leq M_v}$ from (6):

$$[\dot{v}]_i + c_1([v]_i - M_v) \leq 0, \quad (15)$$

with $i \in \mathcal{J}$. Substituting (10) (ignoring the ReLU) we have

$$-[v]_i + [h]_i + (v^{\mathrm{T}}A_i g_0 - g_1 v_{sum}) + c_1[v]_i < c_1 M_v, \quad (16)$$

which, in vector form becomes

$$v^{\mathrm{T}}A_i g_0 - g_1 v_{sum} + (c_1 - 1)[v]_i + [h]_i < c_1 M_v. \quad (17)$$

Finally, using the standard basis vectors $\mathbf{e}_i$ to consider the entire state, we have

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ A_j & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} g + \begin{bmatrix} \mathbf{e}_j \\ (c_1 - 1)\mathbf{e}_j \\ 0 \end{bmatrix}\right)^{\mathrm{T}} z \leq c_1 M_v. \quad (18)$$

The lower bound $v(t) \in \mathcal{C}_{\geq 0}$ is automatically enforced by our model, as detailed by the following.

Lemma 1: Assuming $v(0) \geq 0$, the dynamics (10a) ensures that $v(t) \geq 0$ for all $t > 0$.

Proof: The dynamics (10) produces continuous trajectories, since its right-hand side is continuous. Together with the assumption that $v(0) \geq 0$, this implies that the only way to have $[v]_i < 0$ is to have $[v]_i(t_0) = 0$ and $[\dot{v}]_i(t_0) < 0$ for some $t_0 \geq 0$. By way of contradiction, assume that such t_0 exists. Then, the right-hand side of (10a) must be negative, i.e., $-[v]_i + \mathrm{ReLU}([h]_i + v^{\mathrm{T}}A_i g_0 - g_1 v_{sum}) < 0$ which implies $[v]_i < 0$. Since ReLU's are always non-negative this creates a contradiction. ■

3) Tracking constraint on $v_{sum}(t)$: As formalized by (11), we desire v_{sum} to track $\mathbf{1}^{\mathrm{T}}v$ with an error tolerance b. Applying Proposition 1 to the first part of the constraint $v_{sum} - \mathbf{1}^{\mathrm{T}}v \in \mathcal{C}_{\leq b}$, we have

$$\dot{v}_{sum} - \mathbf{1}^{\mathrm{T}}\dot{v} - c_3(b - v_{sum} + \mathbf{1}^{\mathrm{T}}v) \leq 0; \quad (19)$$

substituting the dynamics (10) (ignoring the ReLU) we obtain

$$g_2(\mathbf{1}^{\mathrm{T}}v - v_{sum}) + \mathbf{1}^{\mathrm{T}}v - \mathbf{1}^{\mathrm{T}}h - g_0^{\mathrm{T}}\mathbf{1}\mathbf{1}^{\mathrm{T}}v + ng_1 v_{sum} - c_3(b - v_{sum} + \mathbf{1}^{\mathrm{T}}v) \leq 0 \quad (20)$$

which, in matrix form with the full state x, becomes

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ -\mathbf{1}\mathbf{1}^{\mathrm{T}} & 0 & \mathbf{1} \\ 0 & n & -1 \end{bmatrix} g + \begin{bmatrix} -\mathbf{1} \\ (1 - c_3)\mathbf{1} \\ c_3 \end{bmatrix}\right)^{\mathrm{T}} z \leq c_3 b. \quad (21)$$

Following similar steps for the lower bound $v_{sum} - \mathbf{1}^{\mathrm{T}}v \in \mathcal{C}_{\geq -b}$ we obtain:

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ \mathbf{1}\mathbf{1}^{\mathrm{T}} & 0 & -\mathbf{1} \\ 0 & -n & 1 \end{bmatrix} g + \begin{bmatrix} \mathbf{1} \\ (c_4 - 1)\mathbf{1} \\ -c_4 \end{bmatrix}\right)^{\mathrm{T}} z \leq c_4 b. \quad (22)$$

Constraints (21) and (22) are the only ones relating the filter coefficient g_2 to the scalar coefficients g_0, g_1.

Remark 1: The ZCBF coefficients $c_1, c_2, c_3, c_4 > 0$ are user-defined parameters. While any positive value will provide the guarantees of Proposition 1, higher or lower values imply that the trajectories of our system can approach the boundaries of the constraints slower or faster, respectively. In practice, higher values might risk to make infeasible the design problem described in the section below.

C. Filter design via LP Formulation

In this section we use all the robust constraints (14)-(22) and linear constraints (6), (11) into a feasibility LP.

First, we rewrite constraints (14)-(22) into a common form:

$$(M_j g + q_j)^{\mathrm{T}} z \leq r_j \ \forall j \in \mathcal{R}, \tag{23}$$

where M_j are matrices, q_j are vectors, r_j are scalars, and $\mathcal{R}$ is a set of indices mapping to the aforementioned constraints.

Next, we convert the (8) and (7) into the matrix form as follows:

$$Cz \leq d \tag{24}$$

Which all the constraints (23) must hold Our filter design problem is then to find coefficients g such that constraints (23) are satisfied for all z such that (24); this can be formalized into a linear min-max feasibility problem:

$$\begin{aligned} &\min_{\{\delta_j\}} \sum_j \omega_j \delta_j \\ &\text{subject to} \quad \begin{bmatrix} \max\limits_{z^j} (M_j g + q_j)^{\mathrm{T}} z^j \\ \text{subject to } C z^j \leq d \end{bmatrix} \leq r_j + \delta_j \ . \end{aligned} \tag{25}$$

Where $\delta_j \leq 0$ denotes a slack variable representing safety margins j-th constraint. The objective function is to maximize a linear combination of safety margin with user-specified weights denoted by $\omega_j \geq 0$. Intuitively, the inner-level $\max$ problem looks for the worst-case state z for the j-th constraint; since this worst-case can be different for different constraints, we need to have a separate variable z^j for each constraint $j \in \mathcal{R}$.

Problem (25) is not an LP due to the presence of the inner $\max$ and the bilinear nature of its objective; both of these issues can be addressed by replacing the inner problem with its dual:

$$\begin{aligned} &\min \sum_j \omega_j \delta_j \\ &\text{subject to} \quad \begin{bmatrix} \min\limits_{p^j} \ -d^{\mathrm{T}} p^j \\ \text{subject to } C^{\mathrm{T}} p^j = -(M_j g + q_j) \\ p^i \leq 0 \end{bmatrix} \leq r_j + \delta_j \ \forall j, \end{aligned}$$

which reduces to:

$$\begin{aligned} \min_{g, p^j, \delta_j} \ & \sum_j \omega_j \delta_j \\ \text{subject to} \ & -d^{\mathrm{T}} p^j \leq r_j + \delta_j \ \forall j \\ & C^{\mathrm{T}} p^j + M_j g = -q_j \ \forall j \\ & p^j \leq 0 \ \forall j. \end{aligned} \tag{26}$$

Equation (26) is finally an LP, which can be solved using standard linear optimization software packages.

V. Considering the non-linearity of the model

The solution to (26) ensures that (P1)-(P3) are satisfied for all pairs of input and state such that $[h]_i + v^{\mathrm{T}} A_i g_0 - g_1 v_{sum} \geq 0$ for all i, i.e., the ReLU in (10) is not active. Property (P2) (sustained activity) is captured by the dynamic nature of the filter, and part of (P3) (non-negativity bound $v \in \mathcal{C}_{\geq 0}$) also holds for the full nonlinear dynamics (10). In addition to these, in this section, we show that property (P1) and the full property (P3) also hold on a significant part of the state space, for constant inputs, and if the kernel g_0 is a scalar, i.e., $g_0 \in \mathbb{R}$.

A. Property (P1): Winner takes all

Proposition 2: If $\mathrm{ReLU}(\mathfrak{m}(h) + v^{\mathrm{T}} A_{i_{\mathfrak{m}}} g_0 - g_1 v_{sum}) > 0$, then (4) holds, i.e., all the bins decrease faster than the mode.

Proof: When the LP problem (26) is feasible, it implies that the inequality (14) holds. For a given element $i \neq i_{\mathfrak{m}}$, if its ReLU argument is not active, i.e., $\mathrm{ReLU}([h]_i + v^{\mathrm{T}} A_i g_0 - g_1 v_{sum})$ is strictly greater than zero, then $[\dot{v}]_{i_{\mathfrak{m}}} > [\dot{v}]_i$ as desired. Based on (14), and since the output of ReLU is always positive:

$$-[v]_{i_{\mathfrak{m}}} + \mathrm{ReLU}(\mathfrak{m}(h) + v^{\mathrm{T}} A_{i_{\mathfrak{m}}} g_0 - g_1 v_{sum}) \geq -[v]_i$$

Which implies that $\dot{v}_{i_m} > [\dot{v}]_i$ also when ReLU is equal to zero for $[v]_i$ ∎

B. Property (P3): Boundedness and convergence

The full nonlinear dynamics (10) can be seen as a hybrid system where each mode is defined based on whether the ReLU function is zero or not for $[v]_i$. We denote each mode with $\mathcal{P}_{\mathcal{Q}}$, where $\mathcal{Q} \subset \{1, \cdots, n\}$ indicates which ReLU terms are active (non-zero) in each v_i. Note that the dynamics is linear in each mode.

We first focus on mode $\mathcal{P}_{i_{\mathfrak{m}}}$, and show that it contains a stable equilibrium. In the following, we show that v^* is stable equilibrium point.

Proposition 3: Let v^* be a state where $[v]_i = 0$, except for $i = i_{\mathfrak{m}}$. The following is an equilibrium for the dynamics (10) restricted to mode $\mathcal{P}_{i_{\mathfrak{m}}}$:

$$\begin{aligned} [v^*]_j &= 0 \quad \forall j \neq i_{\mathfrak{m}}, \\ [v^*]_{i_{\mathfrak{m}}} &= \frac{[h]_{i_{\mathfrak{m}}}}{1 + g_1 - e_{i_{\mathfrak{m}}}^{\mathrm{T}} A_i g_0}, \\ v^*_{sum} &= [v^*]_{i_{\mathfrak{m}}}. \end{aligned} \tag{27}$$

If, in addition, $g_2 - g_0 + 1 \geq 0, g_1 \leq g_2$ and $g_2 > 0$, then the equilibrium is stable.

Proof: By substituting $[\dot{v}]_{im}$ for $[\dot{v}]_i$ and assuming $[\dot{v}]_i$, $i \neq i_m$, one can prove that $\dot{v} = 0$ at v^*. Let $\dot{x} = Ax + Bh$ be a shorthand for the linear dynamics $\dot{x}_v = Ax_v + Bh$ where $x_v = [v, v_{sum}]$ and $A \in \mathbb{R}^{n+1, n+1}$ obtained by restricting

(a) (b) (c)

Fig. 4: An example of disrupted images in our scenario. 4a is an image in KITTI dataset, 4b and 4c are the same image with blur noise and obstruction noise added, respectively.

(10) to mode $\mathcal{P}_{i_\mathrm{m}}$. Each row of A (A_i) is equal to:

$$\begin{aligned} A_i &= [0 \quad 0 \cdots \underbrace{-1}_{i} \cdots 0] \quad \forall i \neq i_{m(v)}, i \neq n+1 \\ A_{i_\mathrm{m}} &= [0 \quad 0 \cdots \underbrace{-1+g_0}_{i_m} \quad 0 \cdots g_1] \\ A_{n+1} &= [g_2 \quad g_2 \cdots -1] \end{aligned} \tag{28}$$

Since the filter dynamics (10) is invariant to i_m, so characteristic polynomial of A is independent of i_m. Hence, without loss of generality we can calculate it for the special case that $i_m = n$

$$(-\lambda - 1)^{(n-1)}(\lambda^2 + \lambda(g_2 - g_0 + 1) + g_1 g_0 - g_0 g_2) \tag{29}$$

If $g_2 - g_0 + 1 \geq 0, g_1 \leq g_2$ and $g_2 > 0$ all eigenvalues are in the left-half plane. ■

A similar procedure can be applied for the case where the dimension of g_0 is greater than one.

Remark 2: The constraint $g_2 - g_0 + 1 \geq 0, g_1 \leq g_2$ can be added to the LP (26) to ensure stability, although in all our tests this was not necessary to obtain a stable filter (i.e., the solution (26) naturally satisfied this additional constraint).

Conjecture 1: For any arbitrary h that satisfies (7), (8), system (10) traverses modes $\mathcal{P}_\mathcal{Q}$ until it reaches $\mathcal{P}_{i_\mathrm{m}}$, where it converges to v^*.

Propositions 2 and 3 provide the intuition that, $[v]_{i_\mathrm{m}}$ grows faster and other bins decrease until their ReLU become inactive (i.e., ReLU argument is strictly negative) which is equivalent of mode $\mathcal{P}_{i_\mathrm{m}}$ where it converges to v^*. We have seen this in practice, but we are not able to prove it rigorously.

VI. Implementation and Case Study

We solve (26) for $n = 16$ bins and a scalar kernel $g_0 \in \mathbb{R}$, obtaining the following: $g_0 = 0.998$, $g_1 = 0.621$, $g_2 = 80$ for $b = 1.6, \delta = 0.1, c_3 = c_4 = 800, c_1 = c_2 = 100$. To demonstrate the performance of this filter, we trained a convolutional neural network to estimate the velocity of a vehicle given the video stream of a camera mounted on top of it. The input of the neural network is two sequential frames with size $[128, 512]$ and the output is the discrete distribution of speed probability with 16 bins. The speed range in our dataset is between 0-$16m/s$ which is distributed uniformly among the output bins.

The CNN consists of three convolution blocks and four fully connected layers with Relu activation functions with $[4096, 1024, 256, 15]$ neurons, respectively. A softmax activation function is applied as the last layer to classify to get a probabilistic class across 16 possible probabilistic classes. Each convolution block includes convolutional layers with 64 channels and a 3×3 filter, followed by a batch normalization layer and a ReLU activation function.

(a) (b)

Fig. 5: Example in which the CNN cannot estimate the speed correctly due to the presence of an outlier (the truck crossing the street)

We use a mean square loss function with Adam optimizer for 25 epochs with a constant learning rate 0.0001. The raw grayscale images of the KITTI dataset [12] are used for both training and validation.

The proposed architecture was able to estimate the vehicle speed with 95 % and 91 % accuracy for training and validation, respectively. However, in some more challenging scenarios, such as when an pedestrian or a car crosses, the network could return an incorrect estimation. One example is shown in Figure 5.

Since the output is essentially a histogram, we applied our filter in order to make the estimation more robust and evaluated its performance in a very noisy scenario as well. In order to simulate disrupted frames, we blur or add black boxes, as demonstrated in Figure 4. We add these disrupted frames with different frequencies and evaluate the accuracy of our system with and without the filter. The result is plotted in Figure 6. The horizontal axis represents the ratio of disrupted frames, indicating one frame out of n_f frames is disrupted, where n_f is a natural number in the range of $[2, 10]$, and the vertical axis is the accuracy of speed estimation. The neural network is not trained for these scenarios; hence, its accuracy increases linearly as the number of disrupted frames decreases. Whereas the accuracy of filtered estimation does not change significantly after a while as the number of disrupted frames decreases, indicating the effectiveness of the filter short-time memory.

The robustness of our filter introduces lag to the estimation. The filter lag can be adjusted based on the change rate of the signal and noise by two hyperparameters; b in (11) and the simulation time-step Δt. Note that, our filter is continuous

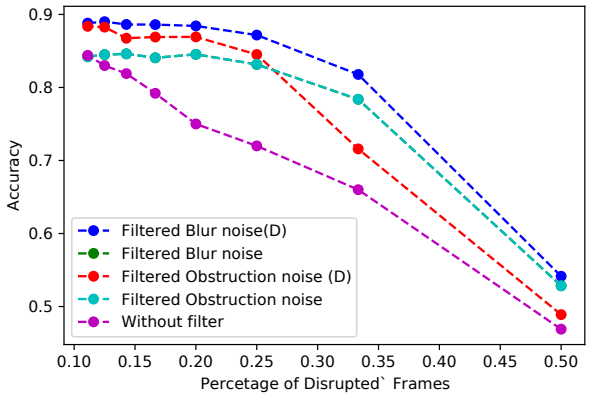


Fig. 6: The effect of the filter on estimating the velocity of a vehicle based on a disrupted stream of images. The horizontal axis shows the ratio of number of disrupted frames. In results denoted by (D), frames are fed to the pipeline twice in order to decrease the lag.

in time, but in order to implement it we have to make it discontinuous with the time-step Δt. To further counteract this, the filter can be run multiple times for each image; this effectively increases the convergence rate and reduces lag. However, if repeated too many times, the performance can actually decrease because corrupted frames become more frequent as well.

VII. Conclusion

In this paper, we introduced a novel filtering method inspired by the behavior of complex cell networks. The filter parameters are derived from an optimization problem. This optimization problem returns the optimal filter coefficients such that the filter selectively amplifies the mode of the input histogram while having (P1)-(P3) properties. Similar to biological systems, as demonstrated by our case study, our filter exhibits promising robustness with the minimum required computation. Due to its short-term memory, this filter is able to efficiently reduce the effect of outliers, which is difficult to achieve with conventional methods.

As a case study, we utilized this filter to improve the robustness of the estimation of the speed of a vehicle based on a neural network. Other approaches incorporate other measurements, such as IMU to overcome visual measurement problems; however, we showed our filter provides robust estimation solely relying on a stream of images.

Future work:. One of the consequences of the robustness of this filter is the lag, which becomes more significant as the rate of change of the input signal increases. By taking the system model into account, we intend to reduce lag and improve accuracy. In addition, there are some missing parts in IV. In order to complete it, first we need to prove conjunction (1). Second, Propositions3,2 assume ReLU term of $[v]_{i_{\mathrm{m}}}$ is active, and we must demonstrate that it becomes active sooner than other bins regardless of the initial condition.

References

[1] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[3] M. Bahreinian, E. Aasi, and R. Tron. Robust path planning and control for polygonal environments via linear programming. In *2021 American Control Conference (ACC)*, pages 5035–5042. IEEE, 2021.

[4] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848, 1995.

[5] B. Boashash. *Time-frequency signal analysis and processing: a comprehensive reference*. Academic press, 2015.

[6] G. Casiez, N. Roussel, and D. Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530, 2012.

[7] S. Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on industrial electronics*, 59(11):4409–4420, 2011.

[8] Z. Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[9] P. Dayan, L. F. Abbott, et al. Theoretical neuroscience: computational and mathematical modeling of neural systems. *Journal of Cognitive Neuroscience*, 15(1):154–155, 2003.

[10] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[11] E. S. Gardner Jr and D. G. Dannenbring. Forecasting with exponential smoothing: Some guidelines for model selection. *Decision sciences*, 11(2):370–383, 1980.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[13] G. C. Goodwin and K. S. Sin. *Adaptive filtering prediction and control*. Courier Corporation, 2014.

[14] N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[15] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[16] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[17] J. J. LaViola. Double exponential smoothing: an alternative to Kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments*, pages 199–206, 2003.

[18] J. Li, Z. Xu, D. Zhu, K. Dong, T. Yan, Z. Zeng, and S. X. Yang. Bio-inspired intelligence with applications to robotics: a survey. *arXiv preprint arXiv:2206.08544*, 2022.

[19] J. Li, S. X. Yang, and Z. Xu. A survey on robot path planning using bio-inspired algorithms. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2111–2116. IEEE, 2019.

[20] M. J. Milford, G. F. Wyeth, and D. Prasser. Ratslam: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 403–408, 2004.

[21] A. I. Mourikis, S. I. Roumeliotis, et al. A multi-state constraint kalman filter for vision-aided inertial navigation. In *ICRA*, volume 2, page 6, 2007.

[22] R. Olfati-Saber. Ultrafast consensus in small-world networks. In *American Control Conference*, volume 4, pages 2371–2378, 8-10 June 2005.

[23] S. Särkkä. *Bayesian filtering and smoothing*. Cambridge university press, 2013.

[24] B. Sun, D. Zhu, and S. X. Yang. A bioinspired filtered backstepping tracking control of 7000-m manned submarine vehicle. *IEEE Transactions on Industrial Electronics*, 61(7):3682–3693, 2013.

[25] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–93, 2000.