Optical Flow Estimation pre-training with simulated stage II retinal waves

Christos Kyriazis, Bruno Cessac, Hui-Yin Wu, Pierre Kornprobst

Optical Flow Estimation pre-training with simulated stage II retinal waves

Christos Kyriazis, Bruno Cessac, Hui-Yin Wu, Pierre Kornprobst

RESEARCH REPORT
N° 9562
November 2024
Project-Team Biovision

ISSN 0249-6399 ISRN INRIA/RR--9562--FR+ENG

Optical Flow Estimation pre-training with simulated stage II retinal waves

Christos Kyriazis*, Bruno Cessac†, Hui-Yin Wu‡, Pierre Kornprobst§

Project-Team Biovision

Research Report n° 9562 — November 2024 — 34 pages

Abstract: This report presents the work of a 6-month internship part of the *MSc in Modeling for Neuronal and Cognitive systems*. This work explores possible approaches in the use of simulated Retinal Waves (RWs) as pre-training for machine learning (ML) computer vision models in Optical Flow Estimation (OFE). Retinal Waves are one of the early processes of visual system development in mammals, structuring the retinal connectivity and thus preparing for vision, including motion detection. We select a recent, non-Transformer ML architecture, RAFT [1], and adopt a Transfer Learning strategy to leverage RWs in enhancing OFE performance through a related task. Additionally, we explore an alternative approach that estimates an approximated optical flow of RWs, allowing for its direct application within OFE. The idea of using these biological stimuli to generate more accessible training data to improve the generalization capabilities of OFE models shows a limited effectiveness with both approaches.

Key-words: Optical Flow Estimation, Deep Learning, Retinal Waves

* Christos Kyriazis has worked as an intern for the Biovision Team for 13 months in INRIA, Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France, between March 2023 and August 2024.

† Bruno Cessac is an Inria Research Director and team leader of the Biovision team in INRIA, Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France.

‡ Hui-Yin Wu is an Inria Research Scientist in the Biovision team in INRIA, Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France.

§ Pierre Kornprobst is an Inria Research Director in the Biovision team in INRIA, Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France.

RESEARCH CENTRE
Centre Inria d'Université Côte d'Azur

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Pré-entraînement à l'estimation du flux optique avec des ondes rétiniennes simulées de stade II

Résumé : Ce rapport présente le travail réalisé dans le cadre d'un stage de 6 mois pour le *MSc en Modélisation des systèmes neuronaux et cognitifs*. Le projet s'intéresse aux possibilités d'utiliser des ondes rétiniennes (RW) simulées de stade II pour pré-entraîner des modèles de *computer vision* en apprentissage automatique (ML) dédiés à l'estimation du flux optique (OFE). Les ondes rétiniennes sont un processus clé dans le développement initial du système visuel des mammifères, structurant les connexions rétiniennes et préparant la détection du mouvement. Pour évaluer cette idée, nous avons sélectionné une architecture récente d'OFE, RAFT [1], et mis en place une stratégie de Transfer Learning visant à exploiter les RW à travers un objectif associée pour améliorer les performances en OFE. Une seconde approche, consistant à approximer le flux optique des RW afin de l'appliquer directement à l'OFE, a également été explorée. L'idée d'utiliser ces stimuli biologiques pour générer des données de pré-entraînement plus accessibles, dans le but d'améliorer les capacités de généralisation des modèles d'OFE, montre une efficacité limitée avec ces deux approches.

Mots-clés : Estimation du flux optique, Deep Learning, Ondes rétiniennes

Contents

1 Introduction

Mammals, including humans, undergo a crucial phase of learning vision, part of if before birth, i.e. before light reaches the retina. This process is orchestrated by spontaneous neural activity within the retina, termed **Retinal Waves** (RWs). These pre-visual neural activities play a fundamental role in shaping the neural circuitry necessary for processing motion once the eyes are functional. In contrast, machine learning (ML) computer vision models learn by being trained on vast amounts of data, allowing them to fine-tune their artificial neural networks (ANNs) to accurately process and interpret visual inputs for a given task, such as to identify and classify actions in videos. Despite fundamental differences between biological learning and ML, such as one depending on genetically programmed neural development while the other relies on optimization and statistical algorithms, both processes can involve a form of training to refine their respective systems. For ML, this involves training on different datasets to obtain better performance, whereas RWs are a dynamical process, arising in a short period of time, and triggering plasticity processes in the visual cortex (see [2] and references therein). RWs capture intricate spatiotemporal dynamics which could serve as accessible and efficient training data. The central objective of this work, is to investigate whether simulated stage II RWs can contribute in ML training for a specific computer vision task, namely **Optical Flow Estimation** (OFE). In the next section of this introduction, the role of RWs is presented, followed by the related work about the employed mathematical model for stage II RWs, OFE theory and a literature review.

1.1 Retinal Waves and their role in the visual development of mammals

Retinal Waves have been observed across various vertebrate species and are spontaneous bursts of action potentials in the immature retina, forming activity waves in abstract patterns. They occur during late embryonic development and disappear a few weeks after birth [3], depending on the species. Experiments as [4][5][6], show that retinal waves play a crucial role in development of early activity-dependent formation of retinal circuits as well as their projections to the visual pathway; i.e. they have an *instructive* role in eye-specific segregation and retinotopic refinement. Figure 1 below shows an example of a time-lapse of retinal waves propagating across the axonal arbors of retinal ganglion cells (RGCs) that terminate in the superior colliculus (midbrain) in P10 mice [6].

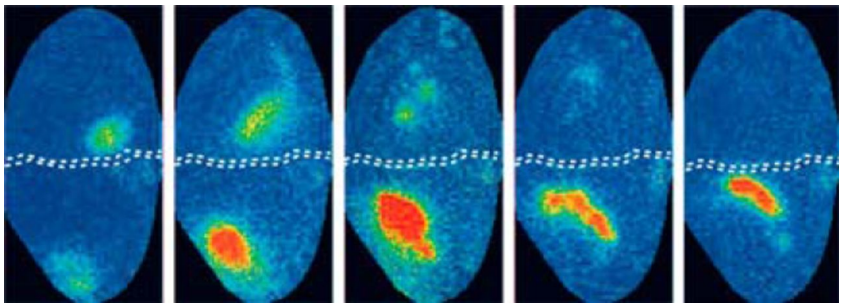


Figure 1: Time-lapse images of retinal waves propagating across the axonal arbors of retinal ganglion cells that terminate in the right (above dashed lines) and left (below dashed lines) superior colliculus, from [6].

The development of the retina is divided in several steps explained below, taken from [7]. The **first** step involves establishing the appropriate distribution of the 7 different cell types that

compose the retina, namely rod, cone, bipolar, horizontal, amacrine, ganglion and müller cells. In the **second** step, the cells undergo migration to accurately position themselves within the retina. In the **third** step, neurons establish synaptic connections with other retinal neurons and with their projections to the brain. In this step, three different **stages** of retinal waves are observed, varying in period of occurrence depending on the species. The first stage waves, *embryonic waves* (before birth), propagate via electric synapses (gap junctions). The second stage waves, *cholinergic waves*, are produced by a transient network of cholinergic (acetylcholine) connections between amacrine cells, and start around birth and end after the first postnatal week. The third and final stage of retinal waves, *glutamatergic waves*, occur 2-3 days before eye opening and end right after; they are driven by glutamatergic signaling. These retinal wave stages are in line with a concept of *checkpoint model* of neuronal development [8], suggesting a sequential maturation of the retinal circuitry, relying on checkpoints to transition between stages. It is believed, that stage II retinal waves, *cholinergic waves*, are one of the main contributors for retinotopic map refinement, eye-specific segregation of retinal projections, and maturation, connectivity of both inhibitory and excitatory neurons in the visual cortex [9].

Experiments have shown that Starbust Amacrine Cells (SACs) are the cell type that gives rise to stage II retinal waves [10] by synchronizing locally through cholinergic coupling (acetylcholine and nicotinic receptors). These waves start in small clusters of synchronized neurons and then propagate over the entire retina, with the wave-induced refractory zones as boundaries. Retinal waves can be characterized by a size, duration, speed and a refractory period [11], and their global spatial patterns are determined by the local history of retinal activity [12]. Note that, after the stage II, SACs evolve to show directionally selective behaviour in the mature retina [13][14], playing a significant role in motion detection in visual processing. The work presented in this report, employs simulations of cholinergic waves (stage II), using a SAC-model [15][16].

2 Related Work

2.1 Modeling stage II retinal waves

To simulate stage II retinal waves, a biophysical model by B. Cessac [16] and D. Matzakou-Karvouniari [17][15] is employed. This model allows us to simulate a lattice of SACs in the conditions where stage II retinal waves occur. The following section briefly discusses this biophysical model. It should be acknowledged that the dynamical model by D. Matzakou-Karvouniari et al. is highly complex and consists of a substantial number of equations, for this reason the full model is included in the Appendix.

The model is based on the Morris-Lecar neuron model [18] and involves 6 variables, controlled by non-linear differential equations. The variables include $V(t)$, the local membrane potential; $N(t)$, the gating variable for fast voltage-gated K^+ channels; $C(t)$, the intracellular Calcium (Ca) concentration; $A(t)$ the concentration of Actylcholine (Ach) released by SACs ; $R(t)$ and $S(t)$, the gating variables for slow Ca^{2+}-gated K^+ channels. The membrane voltage $V(t)$, contains *Leak, Calcium, Potassium* currents, as well as a total current. Depending on the choice of parameters, and in agreement with experimental observations, SACs exhibit spontaneous bursting which propagates to the SACs network via Acetylcholine release exciting the connected cells. The bursting and therefore the waves, are initiated by a white noise ξ_t with mean-square deviation σ.

This model simulates the behavior of SACs and aims to demonstrate the underlying mechanisms of retinal wave generation. Retinal waves are triggered if the cholinergic conductance (g_A) is large enough to start a chain reaction. A neuron starts bursting, generating a current which causes other neighbours to burst, with a feedback prolonging the bursting duration of

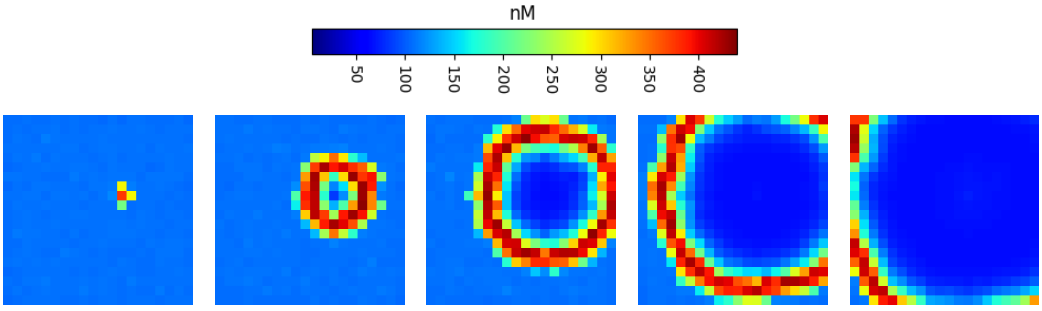
the initiatory SAC [17]. The cholinergic coupling between SACs facilitates local synchrony and coordinated bursting activity, generating propagating waves of neural activity across the retinal circuitry.

When simulating this model, a transient period where the network is relaxing to a rest state is necessary. During this transient state, due to their initial state and the noise, the SACs can fire in irregular patterns in a desynchronized manner. The transient state allows the system to stabilize, reaching an equilibrium point.

Simulating Stage II retinal waves in 2D

In order to simulate the previously introduced cholinergic wave model in a two-dimensional setting, a python package called Brian2 [19] is used. Brian2 is a spiking neural network simulator, which uses mathematical neuron models to accurately simulate neural behaviour and interaction. Brian2 allows for complex and memory-efficient clock-driven simulations, and is employed for its flexibility and relatively simple usage.

The Python code for simulating the SAC model using Brian2 is based on the work by D. Matzakou-Karvouniari and E. Kartsaki [20]. The simulation parameters are modified to be consistent with the work of B. Cessac and D. Matzakou-Karvouniari [15][16]. The script was substantially modified to accommodate a higher degree of configurability, as well as CUDA parallelization [21] for large-scale simulations. Further, it should be mentioned that during a previous internship, a pipeline utilizing an improved algorithm was developed to analyze the statistical properties of simulated stage II RWs. This pipeline enabled the tuning of model parameters for the large-scale 2D simulations employed in this work. The SAC model is set to be simulated with noise-driven bursting, within a square lattice of horizontally and vertically connected neurons. The intracellular Ca concentrations for each neuron across the lattice, can be visualized as seen in Figure 2. In the remainder of this report, to avoid confusion with optical flow vector fields, the SACs Ca concentrations are visualized with a black (low concentration) and white (high concentration) color-map.



(a) 4.5s (b) 4.8s (c) 5.1s (d) 5.4s (e) 5.7s

Figure 2: Intracellular Ca concentrations of a simulated retinal wave in noise-driven bursting (400 Neurons). The 5 frames with 0.3 sec intervals are taken from a 80 sec long simulation with V=-72.0V, $\sigma = 6pA\,ms^{-\frac{1}{2}}$, $g_{sAHP} = 2\,nS$, $g_A = 0.25\,nS$ (See Appendix for the meaning of these parameters). Starting from the left (4.5-4.8s), a cluster of bursting neurons is triggering a retinal wave sequence. (5.1-5.8s) The retinal wave is expanding and a refractory zone appears in darker blue at the previously excited cells. At 5.8s we see the wave disappearing into the edges of the lattice and a global refractory period is starting.

As seen from the example, the retinal waves are visualized by color-mapping the intracellular Ca concentrations, C, the medium-timescale variable. C has a much slower timescale compared to V (see Figure 12 in the Appendix), thus monitoring Ca dynamics provides a more comprehensive insight into the global patterns of neuronal activity. During the global refractory period, neurons do not spike, and by consequence, C is low. Once noise excites one or more neurons enough to start bursting, the Ach production of the cell increases, thereby increasing the depolarisation of neighbours, eventually triggering a bursting chain reaction. The increase of polarity of the neurons, generates an influx of Ca, forming the wave patterns observed in 2D. Consequently, Ca-gated Potassium channels generate a slow after hyperpolarization current (sAHP), which stops the neurons from bursting after a certain time. This forces the neurons to rest, returning C to a baseline level and hyperpolarizing the neurons (refractory period). For this reason, in a scenario with multiple interacting waves, we cannot have waves crossing these trails of refractory zones left by traveling waves.

2.2 Optical Flow Estimation

2.2.1 Theory

Motion analysis has always been one of the fundamental challenges in the field of computer vision. Motion analysis involves understanding the apparent motion that is present in two or more consecutive images, where the 3D motion of the physical world is represented in a 2D medium. A critical component of this analysis is Optical Flow Estimation (OFE), which aims to estimate the 2D motion of 3D objects within a scene, based on the changes in the 2D pixel intensities over time.

The term Optical Flow was introduced by J.J. Gibson [22] and is described as the apparent flow of movement of objects in the visual field, relative to the observer. Mathematically, this corresponds to a dense field of displacement vectors, representing the pixel motion of adjacent frames. The problem of estimating Optical Flow is defined as **determining the dense field of displacement vectors that describe the motion of pixels between two consecutive images of a video**.

Definitions:

- The data (two consecutive images) is a function of space (x, y) and time t.
- A pixel at location (x, y, t) has an *intensity* (brightness) $I(x, y, t)$

We assume that the time interval (δt) between the two consecutive images is constant and that the image intensity between these images is also constant (*brightness constancy assumption*[1]). This means that:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{1}$$

For simplicity, we also assume that the origin (i.e. the camera, the observer) of the images is immobile; no additional motion is added between the images. By taking the Taylor series approximation of the RHS, and dividing by δt we obtain the Optical Flow equation:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{2}$$

[1]The brightness constancy assumption posits that the intensity of a pixel remains constant as it moves between frames. While this is an idealized scenario rarely met in real-world conditions due to changing lighting, reflections, or camera settings, it serves as a fundamental simplification that enables the derivation of the optical flow equation and forms the basis for many optical flow estimation algorithms.

where $u = \delta x/\delta t$ and $v = \delta y/\delta t$ are the x and y components of the velocity (displacement vectors) of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in their respective directions. Rewriting this using the gradient notation we have:

$$\nabla I \cdot \vec{V} + I_t = 0 \tag{3}$$

where ∇I is the gradient of the image, $\vec{V}$ is the velocity vector with components (u, v), and I_t is the temporal derivative of the image intensity.

Equation (3) has two unknowns (V_u, V_v) and relates to the *aperture problem*[23] in computer vision: the ambiguity in estimating motion when an object is viewed through a limited window of observation. To solve this equation, most methods introduce additional conditions and assumptions for estimating the optical flow $\vec{V}$. One of the classic *knowledge-driven* approaches to addressing the aperture problem is the *Lucas-Kanade* method [24]. Lucas-Kanade introduces the assumption that *the motion is locally constant within a small neighborhood around each pixel.* Equation (3) is then over-constrained and allows to estimate the optical flow $\vec{V}$ using a least squares approach. This approach works well on texture-rich regions and corners in an image, making it suitable for sparse OFE but less accurate in dense OFE. Another influential method for OFE is *Gunnar Farnebäck's* algorithm [25], which is designed to handle dense OFE. Unlike Lucas-Kanade method, Gunnar Farnebäck's algorithm approximates the local neighborhood of each pixel with a quadratic polynomial, allowing it to capture more detailed motion information. The algorithm estimates the displacement fields at multiple scales, which enables it to efficiently capture both fine and large-scale motions. Both of these knowledge driven approaches are tested in this project, and results of the latter one are discussed in Section 3.2.

To evaluate the performance of OFE approaches, datasets and benchmarks containing dense ground-truth displacements are created. One of the first of such benchmarks is the synthetic *Yosemite sequence* [26], generated by taking an aerial image of Yosemite valley and texture mapping it onto a depth map of the valley. Another important early benchmark is the *Middleburry benchmark*[27], which provides a comprehensive set of synthetic and real-world sequences with accurate ground-truth optical flow data. These benchmarks target mostly knowledge-driven approaches as they provide a very small number of images and are of relatively low resolution. It should be highlighted that generating accurate optical flow ground truth in real life is challenging due to factors such as camera calibration, motion capture precision and the inherent complexity of natural scenes. Real-world environments often feature varying lighting conditions and occlusions, making it difficult and very time consuming to obtain precise and consistent motion data.

Finally, one of the common metrics of error in OFE is the *Endpoint Error* (EPE) defined as the Euclidean distance between the estimated optical flow $\vec{V}_{est}$ and ground truth optical flow $\vec{V}_{gt}$ as $\|\vec{V}_{est} - \vec{V}_{gt}\|_2$. To provide a comprehensive measure of overall performance in OFE benchmarks, EPE is often averaged across all the N pixels of the image, yielding the average endpoint error (AEE).

2.2.2 Data-driven methods for Optical Flow Estimation

In recent years, deep learning methods have increasingly lead OFE benchmarks firstly adopting *convolutional neural networks* (CNNs) [FlowNet, 2015 [28]] and *transformers* [FlowFormer,2022 [29]]. As expected, supervised approaches are significantly ahead in minimizing AEE, often reducing the error by more than half than leading unsupervised or semi-supervised approaches.

FlowNet [28] marked a significant milestone in OFE by being the first to leverage CNNs for this task, offering a new approach to estimating optical flow from raw images, a field which had previously been dominated by traditional methods. FlowNet demonstrated the feasibility of

using a generic *U-Net* [30] architecture[2] for OFE, showcasing not only its effectiveness but also its ability to achieve real-time performance, a significant advancement over traditional methods at the time [31]. Further works based on this architecture followed, until FlowNet2.0 [32] was introduced in 2017, greatly improving the accuracy of OFE at the cost of increased model size and inference time. Around the same year, SpyNet [33] proposed a different strategy, utilizing a spatial pyramid approach to break down the estimation process using a traditional OFE method, namely *coarse-to-fine* refinement. This method estimates large motions by progressively refining flow estimates through pyramid levels at different resolutions. This greatly improved the model size and inference time, at the cost of accuracy.

In 2021, a highly influential model in the field of OFE was introduced, *Recurrent All-Pairs Field Transforms* commonly known as RAFT [1]. RAFT is explained in greater detail in section 3.1.1. In short, RAFT outperformed the state-of-the-art at the time (FlowNet2.0) by ∼30% in most benchmarks, by introducing a 4D correlation volume to capture all-pair similarity and a recurrent update operator for the iterative refinemenet of the optical flow estimate. This work inspired many OFE models that followed the next two years, GMA [34], GMFlow [35] and FlowFormer [29] to name a few. GMA introduced a global motion aggregation module using transformer-like attention [36] mechanisms to enhance RAFT's architecture. GMFlow further developed this concept by proposing a global matching approach with transformer-based feature enhancement. Finally, FlowFormer presented a full transformer architecture for OFE, leveraging self-attention and cross-attention mechanisms to process 4D cost volumes more efficiently.

2.2.3 Optical Flow Estimation datasets and benchmarks

At the same time as the method development described above, more OFE datasets and benchmarks became established, increasing the available training and testing data to develop and compare data-driven ML models. The following table summarizes the major datasets and benchmarks.

Name	Year	Real World	Dense GT	Resolution (H× W)	Benchmark	Total images
Middlebury [27]	2011	yes	yes	$380 \times 420 - 480 \times 640$	yes	24
MPI-Sintel [37]	2012	no	yes	436×1024	yes	1628
KITTI 2012 [38]	2012	yes	no	376×1240	yes	778
KITTI 2015 [39]	2015	yes	no	375×1242	yes	800
FlyingChairs [28]	2015	no	yes	384×512	no	45744
Spring [40]	2023	no	yes	1080×1920	yes	6000

Table 1: Comparison of OFE datasets. Both images from an image-pair are added in totals. Note that while the input images of spring are in FHD resolution, the ground-truth is in UHD resolution.

We can observe that most datasets and benchmarks are made of synthetically generated data and except for the *Middleburry* benchmark [27], real-world datasets do not contain a densely labeled ground-truth (GT). Additionally, each dataset often specializes in a particular type of

[2]U-Net is a CNN architecture originally designed for biomedical image segmentation. It features an encoder-decoder structure where the encoder captures context through downsampling, and the decoder enables precise localization through upsampling, making it particularly effective for tasks requiring both high-level and low-level information.

(f) Middleburry (g) MPI-Sintel (h) KITTI 2015 (i) FlyingChairs (j) Spring

Figure 3: Samples from the datasets of Table 1. **Top** row shows the first image of an optical flow estimation sample. **Bottom** row shows the ground truth optical flow of the sample. All optical flow samples are visualized using the color-wheel in Figure 14a in the appendix.

motion and realism level. While the *Middleburry* benchmark contains real-world images, it does not entirely represent realistic motions, as it is mostly created with props and controlled environments, simulating very specific scenarios of small displacements. On the other hand, the *MPI-Sintel* benchmark [37] contains 23 realistic computer-generated scenes, showcasing diverse displacements across a variety of environments. KITTI 2012 [38] and 2015 [39] are datasets containing real-world driving scenarios captured from a car-mounted camera, focusing on tasks such as OFE, stereo vision and visual odometry. While these datasets provide optical flow ground truth, the labeling is not dense, covering only about 50% of the pixels. One of the more popular pre-training datasets for OFE, FlyingChairs [28], is synthetically generated by applying affine transformations to images collected from *Flickr* and a set of 3D chair models. FlyingChairs offers a very large variety of motion patterns and displacements, created by overlaying multiple chair images onto cropped background scenes. While it lacks the realism of the other datasets, its extensive size, containing the most samples among OFE datasets, makes it particularly useful for pre-training ANNs. Finally, Spring [40], is a realistic, high-resolution, synthetically generated dataset designed for OFE, featuring diverse and complex motion scenarios with densely labeled ground-truth at a higher resolution than its image pairs. It is important to note that data-driven approaches typically aim to benchmark and train on several of the above-mentioned datasets, often demonstrating their performance on various subsets of these datasets to highlight their generalization capabilities.

2.2.4 Simulated Retinal Waves for Optical Flow Estimation

As mentioned in Section 1.1, stage II RWs contribute to the shaping of the retina and the visual system by forming activity waves in the retina during development, before the retina is capable to perceive light. Now, our visual system essentially handles motion. Indeed, the real word is made of movement. Static images do not exist as our eyes, our body, are always moving. The hypothesis that we make here is that RWs are a way for the visual system to efficiently sample the huge space of spatio-temporal entries, somewhat constituting a natural basis (in the sense of functional spaces). Indeed, due to their dynamics and interactions RWs shows a wide variety of size, duration and curvature [16][17]. Especially, they may display a large training basis for optical flows sampling. In contrast to classical data basis, here, the samples are dynamically generated, at minimal computational cost. As introduced in the previous section, state-of-the-art data-driven approaches for OFE train ANNs using vast amounts of labeled synthetic data. This

section briefly introduces in which ways these two approaches differ to biological vision as well as how simulated RWs may be used as ANN-training data.

The main difference between biological visual development and ML for computer vision lies in the underlying mechanisms of learning. Biological visual development is a gradual process driven by genetic programming, spontaneous neural activity, and environmental interactions, which collectively refine the brain's ability to process visual information [2]. This is notably different from how ANNs learn, which typically depend on predefined architectures, extensive training data and machine learning paradigms, where data directly linked to a specific training objective is provided. While biological systems evolve and adapt over time, classical ML models do not inherently adapt without explicit re-training. Furthermore, the lack of natural image datasets for OFE poses the question of potential bias in model performance, as most OFE datasets are synthetic, and may not fully capture the complexity and variability of real-world motion.

2.3 Retinal Waves in Machine Learning Literature Review

This idea of pre-training with retinal waves is particularly new, as there exists only two publicly available papers investigating this [41][42] dating from December and November 2023. Both works are summarized in the following sections, briefly reviewing and contrasting them with our general objective. Table 2 below provides an overview of both publications.

	A. Ligeralde et al.[41]	B. Cappell et al.[42]
Dataset(s) used	P8-P11 mice retina recordings, reaction diffusion model [43] retinal wave simulations, CIFAR-10, CIFAR-100	stage II retinal wave model [44] simulations, CIFAR-100, ImageNet1k
Retinal wave pre-processing	spatial- and/or temporal-shuffling, inactivity and noise removal	conversion to binary, discarding of images based on similarity and activity
Pre-training	to maintain temporal order of the sequence	to predict the parameter set used to simulate the input retinal wave image
Task-training	standard image classification + 2 variants	standard image classification
Architectures	ResNet-18 (frozen during task-training) + task adapted output layer	ResNet-50 (not frozen during task-training) + task adapted output layer

Table 2: Comparison of experimental setups

2.3.1 Unsupervised learning on spontaneous retinal activity leads to efficient neural representation geometry - A. Ligeralde et al.

In [41], the authors investigate the effects of retinal wave pre-training of Artificial Neural Networks (ANNs) by comparing a classification pipeline between different variants of recorded and simulated retinal waves. The performance of the models were evaluated by classification tasks on the CIFAR dataset. A reaction diffusion model [43] is used for the retinal wave generation, which has been one of the inspirations for the model by B. Cessac and D. Matzakou-Karvouniari [15]. Note however that, in contrast to cessac-karvouniari the model [43] does not allow to modify the probability distribution of RWs. This probability distribution is suspected to play a prominent role in the visual system shaping [16].

The general pipeline can be summarized as follows:

- Pre-processing of retinal wave data by removal of inactivity periods and random noise, as well as shuffling variants of recorded and simulated retinal wave sequences. (variants: temporally-, spatially-, spatiotemporally-shuffled, unshuffled)
- Pre-training of ResNet-18, a well-established ANN architecture for images [45], using SimCLR as learning framework (self-supervised) [46], here used for temporal similarity.
- Supervised task training of the last (classifier) layer, while freezing the pre-trained hidden layers.

The authors justify the choice of pre-training objective (temporal similarity) by evidence that temporally close activity bursts convey the most spatial information about RGC position. Finally, they evaluate the effects of the pre-trained model variants on 3 *supervised* classification tasks: CIFAR-10 classification, spatial translation and color change. The first task involves standard classification of the 10 classes of the CIFAR-10 dataset. The other two tasks are also classification tasks, but modified; **one** randomly chosen image for **10** randomly chosen classes of CIFAR-100, is randomly *translated* or *color changed*. The effects of this pre-training are compared with a random initialization training on an identical network architecture.

Figure 4 shows the accuracy of the three pre-trained classification tasks, compared to the random initialization. The color change task is purely selected for comparison purposes; as retinal waves do not have any role in visual color perception. An important point to understand the significant overall increase in accuracy (random initialization is above 75%) in the spatial translation classification task, is that this task is rather simple. 10 randomly selected image classes of the CIFAR-100 dataset are used to generate 60000 randomly translated versions of **one** randomly selected image for each class for training and testing. However, the authors dive deeper with a manifold analysis to understand the differences between shuffled retinal waves.

The manifold analysis addresses quantitative characteristics of manifolds, high-dimensional geometric structures. Here, the idea is that ANN training corresponds to a motion on a high dimensional, curved, manifold. In summary, manifolds in machine learning define how data is represented in the internal feature space of the network. In the current context of classification, a successful training implies that manifolds are more separable, and more efficiently compressed in the feature space. To better understand the various results of the manifold analysis and due to the limited space of this report, readers are encouraged to refer to the original pre-print [41]. Ligeralde et al. show that in the spatial translation task, manifolds throughout the pre-trained layers are increasingly more separable primarily for networks of simulated or recorded unshuffled retinal waves. They also show that the same networks for the same task compresses the corresponding manifolds best, obtaining a decreasing manifold radius and dimension, across layers. This effect is absent or even reversed in the shuffled retinal wave variants and in the color change task. Concluding, both spatial and temporal characteristics are necessary in pre-training using retinal waves, and with the translation task especially, it is demonstrated that spatial invariance can be learned, without training on large labeled datasets.

2.3.2 ReWaRD: Retinal Waves for Pre-Training Artificial Neural Networks Mimicking Real Prenatal Development - B. Cappell et al.

In [42], the authors use a cholinergic retinal wave model [44] to simulate retinal waves, binarized based on an activation threshold. Their pre-training involves classifying retinal wave images using two versions of their proposed dataset, one with 1024 classes, altering 5 model parameters, and one with 4096 classses, altering 6 model parameters. During this pre-training,

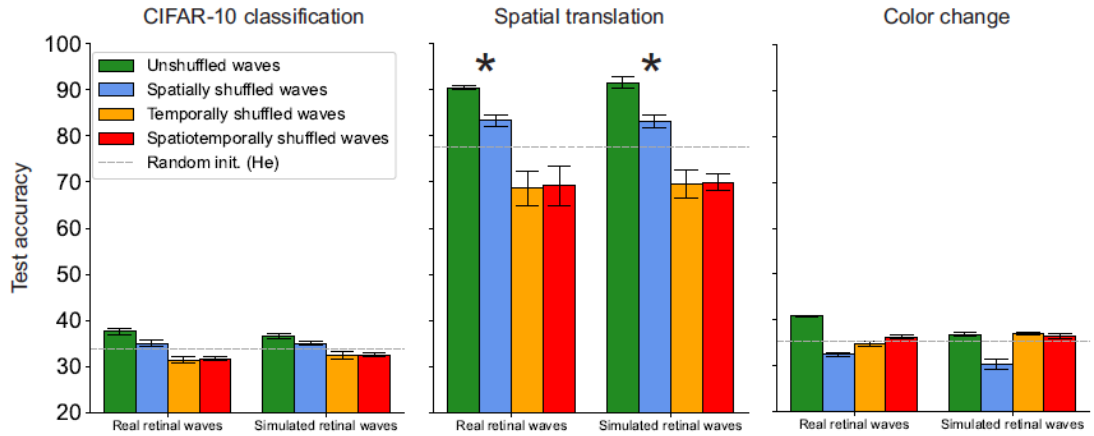


Figure 4: Test accuracy for pre-trained networks in three classification tasks, from [41]. Dashed lines represent accuracy obtained with the same architecture, without retinal wave pre-training. Only the last layer is trained for these tasks, explaining the mostly low accuracy. Asterisks indicate the highest performance increase, with unshuffled retinal waves at the spatial translation task.

images that are temporally *too close* or that contain *too little activity* are skipped, and from the stated pipeline it can be assumed that the images are shuffled (temporally). Their approach reuses the pre-training pipeline of FractalDB (fractal images) on Resnet-50 [47], using CIFAR-100. Interestingly, their resulting test accuracy of the retinal wave pre-trained network is higher but very close to that of FractalDB, for both CIFAR-100 and ImageNet1k datasets.

The general pipeline can be summarized as follows:

- Pre-processing of simulated retinal waves using [44]: binary conversion, sample removal based on temporal similarity and cell activity.
- Classification pre-training on retinal wave images labeled with their parameter configurations (effectively changing wave size, shape and propagation speed) that were used to generate them.
- Classification fine-tuning on image datasets (CIFAR-100 and ImageNet1k)

Contrary to the previously presented publication, during fine tuning, the weights of pre-trained layers are also updated, something that is justified with the fact that our visual system is still adapting after eye opening. Figure 5, shows the validation accuracy during the training on CIFAR-100 for each of the networks trained. We observe that retinal wave pre-trained networks converge a bit earlier than fractal pre-training. However, the end accuracy is reasonably close, perhaps due to the similarity of the pre-trained networks: trained on similar data (black and white fractals vs binary retinal wave images) using the same pipeline. Finally, a Brain-Score is also computed, measuring how well the trained networks match physiological measurements of different regions of the ventral stream of the primate brain. Using this metric, the retinal wave pre-trained networks finetuned on CIFAR100, show increased brain-scores, compared to other models. This is not the case for models fine-tuned on ImageNet1k, likely due to the higher learning rate requirement.

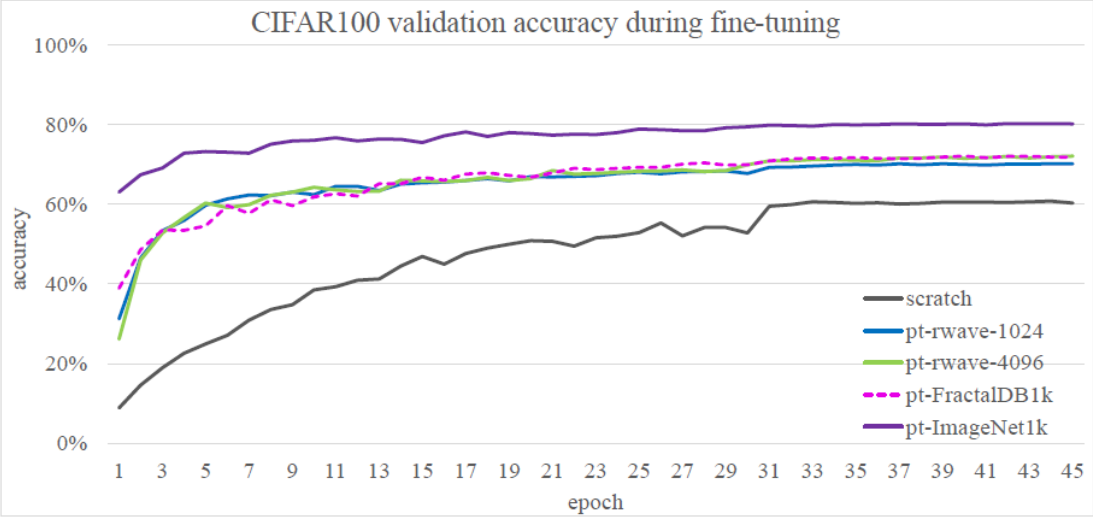


Figure 5: Validation accuracy during fine-tuning of 4 pre-trained networks from [42]; that is, the accuracy after each epoch during training on CIFAR-100, after pre-training with various datasets (denoted pt-). The *scratch* network is trained without pre-training, and we can notice a slower, and lower generalization during training. All models use the same architecture and learning parameters.

2.3.3 Discussion

The two publications use retinal waves in different ways, to pre-train ANNs for image classification. Firstly, both consider natural image classification scenarios, which can be considered less relevant than temporal prediction tasks, when recalling the role of retinal waves in visual development as well as their temporal nature. As mentioned in the introduction, retinal waves are responsible for the establishment of retinotopy, and more importantly for the establishment of directional circuits and motion detection in the retina [14] [48]. This strongly suggests the potential benefits in efficient learning of temporal prediction tasks in machine learning, something which both papers briefly mention. Secondly, Cappell et al.[42] employ binarized stage II retinal waves, a choice that comes with the drawback of sacrificing fine spatial details for a simplified representation of the underlying neural activity. They pre-train the ANNs in a supervised manner, using the simulation parameter combinations as labels, and do not consider the spatiotemporal nature of retinal waves, which is what defines them as *waves*, a spatio-temporally ordered process. Despite this, they prove that even for classification, a biologically inspired pre-training with static retinal waves can bring significant improvements to training from scratch.

On the other hand, Ligeralde et al. use stage II retinal waves to showcase specifically the effects of retinal wave pre-training in three simplified classification tasks through a detailed manifold analysis. The downside here is that they use only one-trainable layer for the classification networks, preceded with a pre-trained ResNet18 (fixed weights) which yields poor results in combination with the chosen tasks. The manifold analysis supports the expected representational efficiency improvements only for the spatial translation task. This can be explained from the usage of SimCLR, pre-training for *temporal closeness* of the retinal waves, which is very closely related to random image translations. Here one can almost expect the networks pre-trained on temporally shuffled waves and spatiotemporally shuffled waves to underperform.

Having said the above, both publications do not consider temporal prediction tasks, but classification tasks in supervised and unsupervised pre-training contexts. Both retinal wave pre-training approaches present interesting results, that showcase the improvements mostly in efficiency of finetuning (generalization), of a given task. These two pre-training approaches show the interest and relevancy in further exploring biologically plausible means of machine learning for computer vision tasks.

3 Retinal Wave pre-training for Optical Flow Estimation

A straightforward idea to use simulated RWs for OFE is to employ them in a *pre-training* context, meaning that a ML model would be trained on RWs and then fine-tuned for OFE. One challenge that arises when using simulated RWs as OFE training data, is the scarcity of the ground truth RW optical flow. Obtaining a precise optical flow ground truth from simulated RW activity is challenging, as the system of SACs is heavily influenced by random noise. In this project, a self-supervised task is adopted using the ML concept of *Transfer Learning*[3] (TL). Additionally, a second approach is tested, which involves approximating the optical flow of RWs.

The first approach is used to train a first ML model for Sequence-to-Sequence (Seq2Seq) prediction, to predict the progression of RW sequences. Applying TL here could allow a subsequent ML model to be trained for OFE, utilizing a portion of the *pre-trained* weights from the first Seq2Seq model. The second approach attempts to pre-train a ML model directly for OFE, using an approximated displacement ground-truth. Recalling the role of RWs in visual development, one could expect that utilizing RWs in both approaches could improve the generalization efficiency of an ML model for OFE, enabling it to achieve performance comparable to its standard OFE training with less data or fewer model parameters.

3.1 Next Frame Prediction pre-training

A first approach to leverage the spatio-temporal dynamics of RWs for OFE is to employ them in a self-supervised task that does not require ground truth optical flow. We select *next frame prediction* (NFP), as it involves predicting the subsequent frame in a video sequence given two or more of the previous frames, which inherently captures motion patterns without needing explicit optical flow ground truth. On RWs, this task enables the model to learn how the wave fronts evolve. Other than the random noise needed to start a burst causing RWs to emerge, their propagation is purely *deterministic*, making it a suitable context for machine learning models to train. A biologically inspired approach as such, could improve the accessibility of the training data needed for OFE models and potentially their generalization capabilities. For this project, we choose RAFT to test this approach, due to its relatively lightweight architecture, recency (2020), and still relevant performance[49]. In the next sections, the architecture of RAFT and how it is used in a TL context are explained, followed by the simulated RW dataset, the methodology and the results.

3.1.1 Overview of RAFT

This section serves as a comprehensive overview of the RAFT model for OFE, based on both the original publication [1] and its official implementation. RAFT is composed of three main

[3]Transfer Learning in machine learning is a set of techniques where a model trained on one task or dataset is adapted to perform on a different task or dataset. This approach allows the model to leverage knowledge from the original task or dataset, improving performance on the new task or dataset, often with less data and/or training time.

components: a feature and a context encoder, a correlation volume, and an update operator. The feature encoder extracts per-pixel features from both input images using CNNs and outputs feature maps at 1/8 resolution, requiring cropping or padding of the input images to be divisible by 8. These features are used to compute a 4D correlation volume, which is then average-pooled at 1, 1/2, 1/4, 1/8 resolution. It is important to note that the correlation volume is not a *trainable* layer, but a dot product computation (to score similarity), between all pairs of the two feature maps of the feature encoder, corresponding to the two input images. The computed volume stores similarity scores between all pixels, and only needs to be computed once for every input image-pair. It allows information to be maintained about both small displacements (at high resolutions) and big displacements (at low resolutions). This information is utilized by a lookup operator in the update block, to iteratively refine a flow estimation. Next, the role of the context encoder is to provide additional context for the flow estimation, extracting features only from the first frame. This can be understood as, by definition, the optical flow displacement vector field represents how pixels move from the first image to their corresponding positions in the second image. The update block refines the optical flow output in iterations, each time retrieving and processing correlation features from the correlation volume to generate flow features, before applying the output of the context encoder. All of the above components can be seen in the RAFT architecture diagram in Figure 6. Finally, it should be noted that RAFT uses a weighted Manhattan distance[4] for the loss function instead of EPE.

The authors of RAFT propose two models, RAFT (∼5.3M parameters) and RAFTsmall (∼1.0M parameters). Their architectures are mostly identical, scaling down the hidden dimensions and removing some layers of the ANNs of all components, and can be schematically visualized in Figure 6. The two variants are trained and fine-tuned on various OFE datasets and start the training sequence by pre-training on the *FlyingChairs* [28] and *FlyingThings* [50] datasets. Performance-wise, both variants of RAFT are parameter efficient, achieving lower EPE faster, with fewer parameters and with a shorter inference time, than other OFE models at the time. This is also one of the reasons why RAFT is chosen; it is interesting to see whether RWs can enhance it in any way. Figure 6 color-codes the reusability of components for next frame prediction (NFP). This is an important consideration for TL, as it is most effective when entire parts of the model can be reused for a different task and/or dataset. Noting how specialized RAFT's update block is, we decide to reuse its feature encoder and correlation volume as the encoder of a NFP model that is trained separately. Then, we transfer those weights to RAFT and train for OFE.

In the context of NFP, the relevance of the context encoder diminishes, as it extracts features solely from the first image, an approach beneficial for OFE but of limited use in NFP. The motion encoder is a key module in RAFT, but is difficult to reuse it as a whole, as it separately processes correlation features and optical flow. If the motion encoder would be used for TL, only two convolutional layers would be involved, making it a less ideal TL strategy. The ConvGRU and FlowHead networks are both specifically designed to handle optical flow features, rendering them unsuitable for NFP.

[4]The weighted Manhattan distance loss function is defined as: $\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-i} \|f_{gt} - f_i\|_1$ where $f_1, ..., f_N$ is the sequence of predictions from N decoder iterations and γ is the weighting factor.

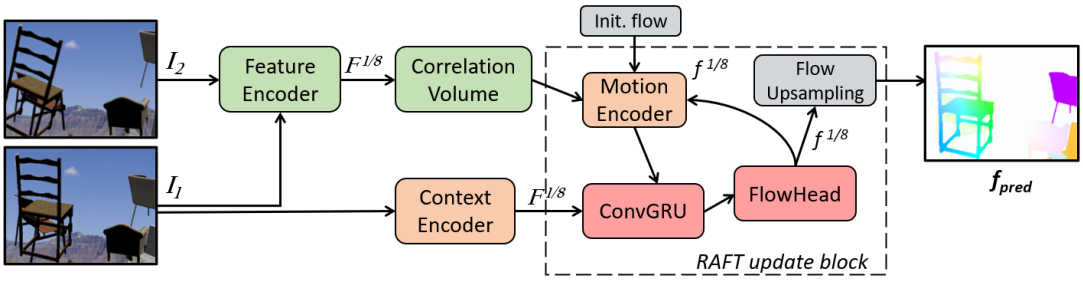


Figure 6: Schematic diagram of the RAFT architecture. I_1, I_2 are the two consecutive input images. f_{pred} is the optical flow prediction. $F^{1/8}$ indicate feature map scale; $f^{1/8}$ indicate flow prediction scale. Colors indicate reusability of components for NFP; Red denotes components where weight transfer is impractical, Orange indicates that some modification is required, Green suggests compatibility with minimal modification. The two input images are fed into the *feature encoder* and the *correlation volume* is computed from their feature maps. Feature maps from the first frame only are also obtained using the *context encoder*. The recurrent *update block* operates by initialising a flow prediction that is processed by the *motion encoder* using selected features from the correlation volume, before passing its output to a *ConvGRU*[5] that takes additional context from the first image. The hidden state of the *ConvGRU* is passed to a 2-layer CNN (*FlowHead*). Upon reaching the specified number of iterations, the predicted optical flow is upsampled and outputted. I_1, I_2 and f_{pred} are taken from the *FlyingChairs* dataset.

3.1.2 Simulated Retinal Wave dataset

The SAC model described in section 2.1 is simulated in 2D using Brian2 [19] (see section 2.1). To match the resolution of the RW dataset to the resolution used when pre-training RAFT (see previous section), the RWs are simulated on a square lattice of 432×432 (186624 cells total). The cells are connected in a grid structure, where each cell has four neighbors: one above, one below, one left, one right. The simulation parameters are set according to B. Cessac and D. Matzakou-Karvouniari [16][17]. To generate enough data, 9-minute long simulations (6 simulations total) are performed with 6 levels of *acetylcholine conductance* $[0.2 - 0.25nS]$; a parameter experimentally [51] shown to decrease with time during visual system development, altering various visual properties of RW [16]. To simulate such long large-scale simulations, brian2CUDA [21] is employed to parallelize the code on a single GPU. This substantially increases the efficiency of the simulation, reducing execution time from several days on CPU, to just a few hours on GPU. Additionally, to make these simulations storage efficient also, only the C variable (intracellular Ca concentration) is recorded in a binary format, at every $t = 10ms$ of the simulation. The model operates in a noise-driven bursting mode with noise parameter $\sigma = 6\ pA\ ms^{-0.5}$.

Preprocessing To obtain an exhaustive dataset for ML, the raw simulation files are preprocessed. Each simulation file is split by timestep and normalized between $[0, 1]$ using *Min-Max*

[5] A Convolutional Gated Recurrent Unit (ConvGRU) is a variant of GRU that replaces the linear operations with convolutional operations. GRU is a recurrent type of ANN, designed to capture long-term dependencies in sequential data.

normalization[6]; all the relevant metadata is maintained in the filename of each timestep. Additionally, the global quiescent phases[7] during each simulation are filtered, by thresholding a number of cells to cross a certain level of activity. This decision is made as for NFP, having the entirety of the cell lattice inactive, could hinder the learning process of the model. The dataset employed for NFP, contains the *active* samples of each simulation, i.e. once a single cell reaches an activity threshold ($4C_0$ where C_0 is the baseline C concentration) determining that a RW is starting. Samples are made up of three (NFP) or two (OFE) consecutive frames of the simulation with a fixed timestep. To augment the samples and have the option to train with more data, at the runtime of the ML pipeline, all possible combinations of a set of spatial transformations can be performed: 90° clockwise rotation, vertical flip, horizontal flip. Finally, the order which the samples are fed to the ML model can be shuffled.

3.1.3 Methodology

NFP Model Next frame prediction is a *sequence-to-sequence* task requiring an encoder-decoder architecture. To reuse the original feature encoder of RAFT and with OFE as context, we define the NFP task as predicting the next frame, when given two consecutive frames from a video. In the context of RWs, the frames are composed by the C values of each cell in the square lattice. Equivalently to RAFT, the NFP model is implemented in PyTorch [52]. The feature encoder taking input two frames, is used exactly as in RAFT, allowing to compute a correlation volume from its outputted feature maps. The correlation features are indexed only once, utilizing the similarity scores at all scales. The indexing is not repeated as done in RAFT for OFE; the recurrent decoder predicts and upsamples the output frame using correlation features once. For simplicity, the official implementation is reused and modified, to accommodate the required changes for TL in this context. For that reason, the NFP decoder is designed to be comptatible with the feature encoder of both RAFT and RAFTsmall without changes in architecture, totalling in $\sim$ 2.8M and $\sim$ 0.9M parameters. The decoder reuses RAFT's implementation of ConvGRU and three transposed convolution layers, each doubling the spatial resolution (*to go from 1/8 to full resolution*) with a kernel size of 4.

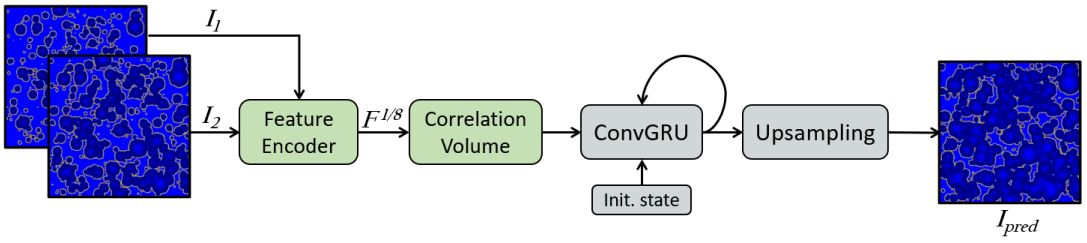


Figure 7: The NFP model is composed of an encoder using RAFT components: *feature encoder*, correlation volume; and a decoder: *ConvGRU*, *ConvTranspose*[8] upsampling layers. The ConvGRU operates at 1/8 of the resolution. The images in this example are simulated RWs from the previously described dataset.

[6] Min-max normalization is a feature scaling technique that transforms numerical data to a fixed range ([0,1]), preserving the relationships among the original data values while bounding the range. $X_n = \frac{X - min(X)}{max(X) - min(X)}$ where X_n is the normalized value and X the original value. Here we normalize each cell of the lattice according to the global min. and max. of each simulation.

[7] A quiescent phase in a neuronal dynamical system, refers to a period of inactivity, representing a state where the neuron is temporarily unresponsive to stimuli, often following a period of intense activity.

3.1.4 Results

The previously described NFP model is trained with the *AdamW* [53] optimizer with the same weight decay as RAFT on FlyingChairs. The loss during training is computed using *Mean-Squared-Error*[9] (MSE) loss. Before training for NFP and performing TL generalization for OFE, the *hyperparameters*[10] of the NFP model need to be tuned. The learning rate is fixed at 5e-4, batch size at 16, for 20 epochs. The performance of the model with different intervals between consecutive frames is also tested; to balance these datasets, we set an upper bound to the augmented samples at ∼12k training samples and ∼1k testing samples. Lastly, the NFP decoder iterations are also compared, and for this we compare 1,4 and 8 iterations of the ConvGRU decoder.

Dataset/hyperparameter	Small → default	iterations 1 → 4	iterations 4 → 8	remove aug.
Relative MSE train	-38.92% (±2.76%)	-12.41% (±3.06%)	-0.50% (±2.45%)	16.85% (±6.52%)
Relative MSE val.	-38.39% (±2.62%)	-12.82% (±3.02%)	0.06% (±3.30%)	32.63% (±1.51%)

Table 3: Summary of performance gains on the recorded MSE after training in percent (lower is better). The rows describe the gains over the training set and validation set. Columns from left to right: model size, decoder iterations, data augmentation removal. Performance gains are computed between all equivalent models and then averaged. For this table, comparable RW datasets of intervals 100ms, 150ms, 200ms are taken into account. All columns except the last, are performed on datasets of ∼12k samples of all frame intervals with a batch size of 16; removing augmentations with 100ms frame intervals decreases the dataset size to ∼3k samples and the batch size is scaled accordingly to 4.

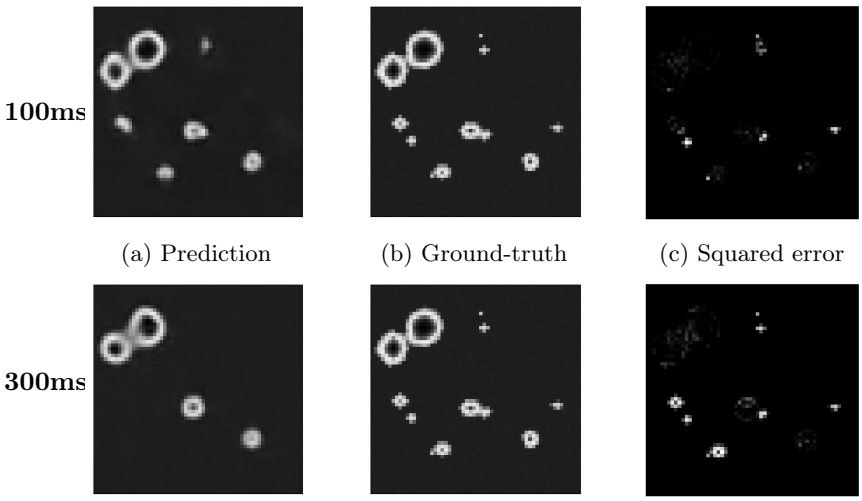
NFP training Table 3 compares the performance gains (relative MSE change) over the model size, decoder iteration and dataset augmentation on the training and test splits. We can observe that the larger feature encoder from RAFT (default) obtains improved accuracy over the small feature encoder. This is in line with the published OFE results as the training EPE decreases by -35.29% on *MPI-Sintel* clean, and -32.89% on KITTI-15 training datasets (*RAFT and RAFTsmall pre-trained on FlyingChairs and FlyingThings datasets*). Comparing NFP performance to OFE performance we can get an indication of how the overall model size impacts task performance. For the NFP ConvGRU decoder performance gains at different iteration counts, we can observe that 4 iterations are sufficient, as at 8 iterations we begin to see minor performance gains. As expected in this context, keeping the order between samples increases the MSE error by 29.45% (±9.93%). The temporal ordering within samples (3 images) is always maintained, but not shuffling between samples increases temporal bias during training and makes the weight updates more prone to overfitting to sequential patterns, reducing the model's ability to generalize. Finally, it should be noted that the models were trained on 4 RW datasets with different intervals between consecutive

[8] ConvTranspose layers are transposed convolutions which are commonly used for upsampling. They perform the reverse operation of standard convolution with trainable filters and increase the dimensions of the input.

[9] MSE is a common loss function in ML, measuring the mean squared difference between predicted and ground-truth values. It is defined as: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{gt} - y_{pred})^2$ where N is the number of samples, y_{gt} are the ground truth values, and y_{pred} are the predicted values.

[10] Hyperparameters in ML are fixed configuration variables that are set before the learning process begins, and control various aspects of model training and architecture. Common examples include learning rate, epoch count, batch size, hidden layer count.

frames (100ms, 150ms, 200ms, 300ms). A larger interval between consecutive frames in a NFP sample means that there is more time for a wave to start and develop between the second input frame and the next (ground-truth) frame. This can be visualized in Figure 8 below. The squared error visualizes the missing waves from the model prediction. It is evident that the models are not able to predict the waves that started between last input frame and ground-truth as the model has no information about them. This is the main contributor to the error during training, with higher frame intervals yielding an increased MSE.



(a) Prediction (b) Ground-truth (c) Squared error

Figure 8: RW crops visualizing the NFP model error on 60x60 crops from a 432x432 RW simulation. From left to right, the crops show the prediction, ground-truth and squared error between the two. The top set is taken from training on dataset samples with a 100ms frame interval, while the bottom set is from training with a 300ms interval. Black to white color-mapped; lower-higher values of C, squared error.

OFE training Here we discuss the results after training the NFP model and transferring the encoder to RAFT for OFE training. The EPE over the training step can be found in Figure 16 in the Appendix. The top four graphs compare training and validation EPE when training on the FlyingChairs dataset for both RAFT and RAFTsmall, as well as their NFP pre-trained variants (RWRAFT, RWRAFTsmall). The pre-trained variants train only the OFE decoder, with the weights of the encoder completely frozen. While this is not a completely fair comparison, it shows how the RAFT model performs with a pre-trained encoder, and whether the RWs improve its training efficiency. From all four graphs, it is clear that the trainable non-pretrained encoder, extracts and adapts to features from the data much better than the pre-trained variants. To understand these results better, a generalization training is performed on the *MPI-Sintel* dataset and is shown in the bottom four graphs of Figure 16. For a fairer comparison of RW pre-training, we pre-train both variants of RAFT on FlyingChairs and fine-tune pre-trained models (both pre-trained on FlyingChairs for OFE, and RW for NFP) with frozen encoders on MPI-Sintel. The recorded EPE after 50k training steps is 75.37% ($\pm$2.29%) higher during training and 100.73% ($\pm$14.49%) higher during validation for the models pre-trained on RWs for NFP. Example predictions of these last two models can be found in Figure 15(b)(c). We can visually compare the above-described difference in performance by noticing the edges of the example such as the tip of the spear and the hair of the left character.

Since the transferring of the weights is only done on the feature encoder between NFP on RWs and OFE, the decreased performance are most likely due the differences in the RW data vs OFE data. The displacement present in RWs is constant and depends on the simulation parameters. This means that during a RW simulation, the speed at which the waves travel is fixed as it depends mostly on the *acetylcoline conductance* of the cells [17]. For this reason we incorporate the different intervals between frames of the dataset, and different values of the relevant parameter, g_A. Nevertheless, there is not a large variety of displacement in a single sample, contrary to most OFE datasets, which often contain various types of motions. Additionally, it should be noted that OFE utilizes certain properties of images, such as *texture*, *edges* and *corners*, all of which are not present in the RW data. For these reasons, in the following experiment, we attempt to describe the motion of simulated RWs to generate a somewhat accurate estimation of their optical flow.

3.2 Optical Flow Estimation pre-training

By the nature of RWs and their biophysical model described in section 2.1, there is no notion of motion or displacement in the system. The simulated SAC cells are not moving, and the only thing that is traveling is the activity of the cells which is best visualized with their *Ca* concentration, due to the slower timescale with respect to their electrical activity. As with Optical Flow ground-truth, we cannot have a true correspondence of pixels between recorded timesteps. In other words, we cannot know that a specific part of a retinal wave front corresponds to cell i at timestep t and cell j at timestep $t+\delta t$. Employing traditional methods such as *Gunnar Farnebäck's algorithm* yields a rather blurry estimation; using this for ML-based OFE with RWs would be less ideal, as the model would learn an inaccurate optical flow. Figure 9 shows such flow estimation, depicting rather blurry fronts without the sharp detail that optical flow ground-truth preserves. Increasing the window size increases the blurriness, and since retinal waves mostly expand, they challenge the *local motion* assumption that most traditional approximations are based on. This means that certain pixels in the second frame do not have a clear source, causing an inconsistent flow field.

3.2.1 Approximation of Retinal Wave Optical Flow

Knowing that RWs have a constant velocity [17], we can approximate the direction and orientation of travel by the gradient; obtaining information about change in activity, at the vertical and horizontal directions at each cell. For this, we can employ *structure tensors*. A structure tensor is a 2×2 matrix used in image processing to capture gradient information in a local neighborhood. It is formed by computing the outer product of the smoothed image gradient and then smoothing it again, usually with Gaussian filters.

$$\text{Smoothed image: } I_s(x,y) = I(x,y) * G_{\sigma_1}$$

$$\text{Structure Tensor: } \mathbf{T} = (\nabla I_s \otimes \nabla I_s) * G_{\sigma_2} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} * G_{\sigma_2} \tag{4}$$

Where I_x and I_y are the image gradients (partial derivatives) of the smoothed image I_s along axis x and axis y.

The outer product of the gradient vector with itself is a mapping that allows local averaging without loss of information. It allows to average gradients with opposite direction, without having them cancel out, maintaining their average orientation. This is achieved with structure tensors by mapping 2D vectors to a 3D space (since it is a symmetrical matrix).

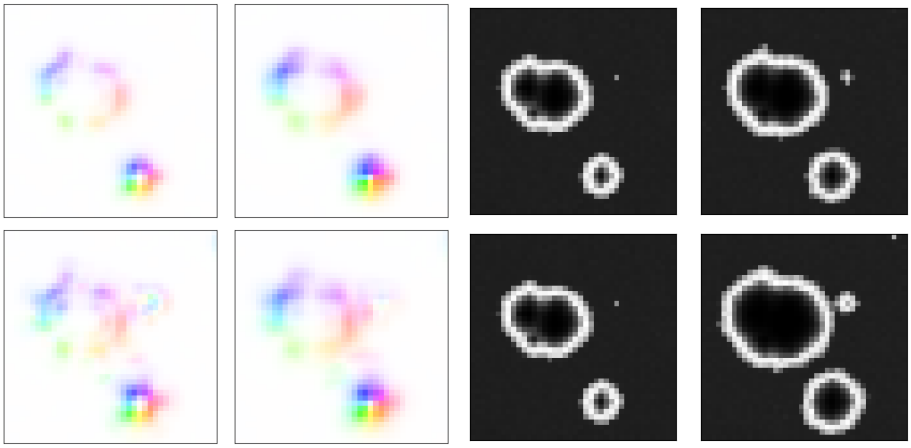


Figure 9: Crop from Farnebäck optical flow estimations of retinal wave simulation using the OpenCV library. [54]. From **left** to **right**: estimation using window size of 2, estimation using window size of 4, reference frame 1, reference frame 2. **Top** row: 100ms interval between reference frames; **bottom row**: 200ms interval between reference frames. All images are crops of 40x40 pixels, corresponding to simulated SAC cells. Flows are visualized with the color-wheel from Appendix 14a.

This information about the orientation of the gradients can be retrieved from the structure tensor by **eigenvalue analysis**. Obtaining the eigenvalues and eigenvectors of this matrix can give various measures of local image structure such as orientation, coherence, anisotropy, and curvature.

In the context of retinal waves for Optical Flow Estimation, to obtain a vector field $\vec{w}$ that provides the overall direction of displacement and growth of the retinal waves, we use structure tensor information in combination with the time derivatives as follows:

The eigenvectors $\vec{e_1}$ (normalized) corresponding to the larger eigenvalue λ_1 give us the information about the **dominant orientation**. To obtain the direction we need to apply a sign α to the vector field $\vec{e_1}$:

$$\vec{w}(t,x,y) = \alpha(t,x,y)\vec{e_1} \tag{5}$$

We want $\vec{w}$ to be resembling the gradient vector field $\vec{z}$, a vector field of the gradients aligned to the direction of the time derivative. This is necessary as the Ca concentrations of retinal waves form fronts with an increasing and then decreasing gradient along the direction of propagation through time. The outer gradient points towards the origin and the inner gradient towards the opposite direction. This happens because prolonged bursting of a cell increases its Ca concentration, while exciting its neighbors, until the bursting stops where the Ca concentration starts to decrease. This can be better understood by the visualization in Figure 10. Since the concentrations decrease along time in the inner gradient, and increase in the outer, an extra minus $(-)$ is required to align the vector field to the propagation of the waves. We use the opposite sign of the time derivative to align the gradients to its direction as:

$$\vec{z}(t,x,y) = -Sign(\frac{\partial C}{\partial t})\nabla C(t,x,y) \tag{6}$$

It is not ideal to use solely $\vec{z}$ for the orientation and direction of retinal waves, as Figure 10

a) and **b)** show that a white, ring-like structure appears in the gradients. This is the peak of the gradient at the wave front, where the *Ca* concentrations are very high and almost equal between neighbors. This is the main reason we use structure tensors, as both the image and the tensor are smoothed (we apply a Gaussian filter with $\sigma = 1.0$).

We obtain α from Equation 5:

$$\alpha(t,x,y) = -Sign(\frac{\partial C}{\partial t})Sign(\nabla C(t,x,y) \cdot \vec{e_1}) \tag{7}$$

The Calcium concentration fluctuates on the peak of the gradient, often causing a small number of cells obtaining the opposite direction by $\alpha(t,x,y)$ (see Figure 10). This is corrected by applying a 3×3 sliding window that checks if the signs of the neighborhood ($\mathcal{N}$) are majority positive or negative.

$$\text{Let } \mathcal{N}_{x,y} = \{(i,j) \mid x-1 \leq i \leq x+1,\ y-1 \leq j \leq y+1\}$$

Majority sign function is defined as:

$$f(\alpha_{x,y}) = \begin{cases} 1 & \text{if } \sum\limits_{(i,j)\in N(x,y)} \text{Sign}(\alpha_{ij}) > 0, \\ -1 & \text{otherwise} \end{cases} \tag{8}$$

f is applied on $\vec{w}(t,x,y)$ by reflecting border values at the edges of the lattice:

$$\vec{w}_f = f(\alpha) \cdot \vec{e}_1 \tag{9}$$

Finally, we scale back $\vec{w}(t,x,y)$ by multipliying it with the original calcium concentrations C, which are min-max normalized per simulation between $[0,1]$. This allows us to retrieve the precise shape of the fronts and diminish the gradient changes of the background. It is important to note that $\vec{w}(t,x,y)$ does not convey information about displacements on its own, but only about the exact shapes present, along with their directions, and orientations. Note that by default, the inactive cells have small fluctuations on their *Ca* concentrations, hence why the background is not completely white despite appearing so. The normalized vector field of eigenvalues $\vec{e_1}$ makes those background fluctuations more visible, which can be impractical for ML. It is decided to keep those as their magnitudes are significantly lower than at the fronts. The completely white parts in the center of the retinal wave (origin) appear white, as the cells are in absolute refractory periods, thus unable to burst; the *Ca* concentrations are at the lowest. Figure 10 below, visualizes the above procedure.

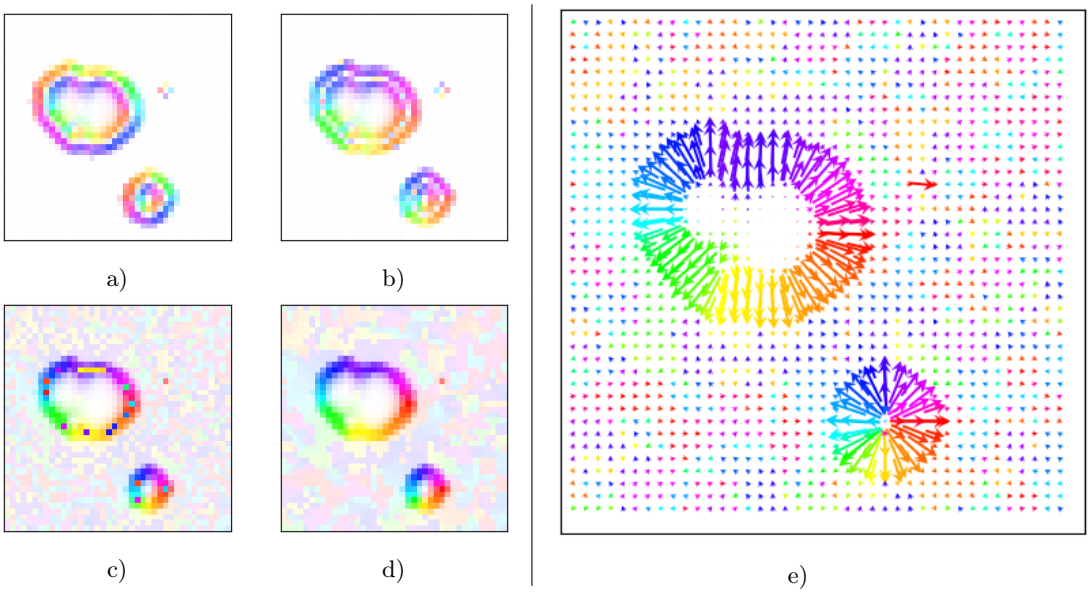


Figure 10: Visualization of Optical Flow approximation of simulated retinal waves using gradient information, of reference frame 1 from Figure 9. **a)** shows the shape of the gradient, computed using central differences. **b)** shows the gradient vector field using the corrected sign ($\vec{z}$). **c)** shows the vector field $\vec{w}$. **d)** shows the vector field $\vec{w}_f$. **e)** shows the quiver plot of the vector field $\vec{w}_f$. All figures are color-mapped using the colorwheel of Figure 14a.

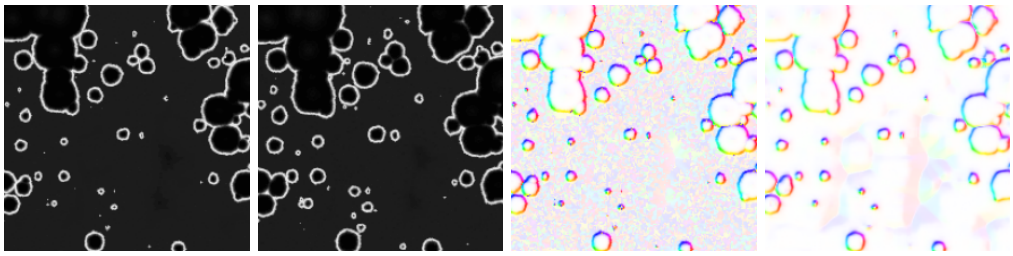
Comparing Figure 10 **d)** to the Farnebäck approximation and corresponding *reference frame 1* from Figure 9, it is clear that the gradient approach catered to the RW data appears more consistent and precise. Now the missing element to make vector field $\vec{w}_f$ resemble more to a displacement vector field, is the scaling of the vector components by a scalar v, such that their magnitudes correspond to the motion, displacement present in the frames (see [55] for a related problem). Due to time constraints of the project we roughly estimate this by sampling isolated waves for each simulation (for each g_A value) and compute the change in radius length. We do this by cropping isolated waves with no interference from other waves throughout T timesteps between t_1 and t_2, and apply a mask to identify the circular wave front at t_1 and t_2. Then, the contours are detected, and the contour area A is extracted using the *Shoelace formula*[11] to compute the radius of the wave as: $r_{t_1} = \sqrt{A/\pi}$. With r_{t_1} and r_{t_2} we can compute the change in the radius over the timesteps, giving us an estimated velocity in pixels (px) $\mathbf{v}$ px/T. For simulations of each g_A value present in the RW dataset, an estimated $\mathbf{v}$ displacement is computed as in Figure 13 in the Appendix. It is important to highlight that estimating $\mathbf{v}$ using observed samples could be inaccurate and is solely performed to *test* this second approach of employing RWs for OFE. A more accurate approach to calculate the speed of waves would involve the entire 6-variable dynamical system as in [17].

[11]The shoelace formula is a mathematical method for calculating the area of a simple polygon given the coordinates of its vertices. $A = \frac{1}{2}\| \sum_{i=1}^{n} (x_{i+1} + x_i)(y_{i+1} - y_i) \|$ where x_i, y_i are the coordinates of the vertices in clockwise order.

3.2.2 Results

We train RAFT for 50000 steps using the approximated RW optical flow dataset, achieving stable convergence. The training and validation EPE graph can be seen in the Appendix on Figure 14b. Due to the relatively small displacements of the waves at the selected 200 ms frame interval (<3 px), all EPE values remain sub-pixel (<1 px). Weight decay helps prevent overfitting, as the default RAFT configuration (5.3M parameters) is likely oversized for the RW data. It should be reminded that EPE in this case is an indicative metric and that the model uses a weighed manhattan distance as a training loss. We set γ to 0.6 (from the default 0.8), to reduce emphasis on the early predictions of the decoder and yield improved end-results.

A sample frame pair with its ground-truth and model prediction is shown on Figure 11. The model accurately captures the optical flow of present wave fronts, though it treats background concentration changes —which are more prominent than ideal (see second-last paragraph of section 3.2.1)— as noise, effectively ignoring them. This model is compared against an equivalent instance of the RAFT model trained for 50000 steps on *FlyingChairs*, by evaluating on the *MPI-Sintel* dataset (Figure 15(e)(f)). We notice the prediction (e), from the model solely trained on the approximated RW optical flow, completely fails to estimate any of the displacements present on the *MPI-Sintel* frame accurately. This is partly expected due to the model not converging on this dataset at all, and due to the many differences in the nature of the two datasets; the motion type and feature variety present in *MPI-Sintel* is absent in the RW dataset. On the other hand, (f), the prediction of the model trained solely on the *FlyingChairs* dataset is comparable to the prediction (c), RAFT fine-tuned on *FlyingChairs*. This is due to the intended statistical similarity of the two datasets (supp. material of [28]). The range of displacements as well as the type of motion, textures and edges present in the *FlyingChairs* dataset are much closer to those found in *MPI-Sintel* than the motion and shapes present in RWs.



(a) Frame 1 (b) Frame 2 (c) Ground-truth (d) Prediction

Figure 11: 200×200 crops from RW optical flow predictions and their corresponding ground-truths (validation set). (a) and (b) show the first pair, and (c) and (d) show the second pair from the approximated RW optical flow dataset described in Section 3.2.1. The model trained on RWs with a 200ms frame interval for 50000 iterations with a batch size of 16 and a learning rate of 5×10^{-4}.

4 Discussion

The work presented in this report undertakes a preliminary investigation into the potential use of retinal waves (RWs) for optical flow estimation (OFE). However, the findings suggest that the tasks selected for transfer learning, such as next frame prediction (NFP), or the specialized pipeline used for OFE (RAFT), may not be ideally suited for this objective, given the methodology and results presented. The structured yet unsupervised nature of RWs was effectively utilized, offering a novel perspective on how this biological activity could inform ML approaches. However, the use of a specialized architecture like RAFT, without incorporating a biologically inspired loss function or constraint, may have hindered both NFP and OFE models from fully capitalizing on the potential of RWs.

In hindsight, a more straightforward proof-of-concept, utilizing a *vanilla* architecture, should have been the initial step of this investigation. Starting with a simpler, baseline model would have provided a clearer understanding of the feasibility and potential of employing RWs for OFE without the added complexity of a state-of-the-art architecture like RAFT.

Due to time constraints, the second approach —approximating RWs for optical flow estimation— was not fully developed or explored. An unintended side effect of this approximation, discussed in Section 3.2.1, was the accentuated noise-like background that may have hindered the model's generalization. Additionally, the larger RAFT architecture's complexity may have led to overfitting, which could have been mitigated by using a smaller RAFT variant. Next, for the RW dataset, not enough RW data was generated to include a broader displacement variance similar to OFE datasets. Finally, the model in Section 3.2.2 was not fine-tuned on *MPI-Sintel*; incorporating the above-discussed adjustments might allow RAFTsmall to generalize more effectively on *MPI-Sintel* using RW data.

While the role of RWs in developing directional selectivity and motion processing in biological systems is well-established, there is growing evidence that their influence on orientation selectivity may be more limited than previously thought [56][13][14][48]. Directly applying the concept of RWs to machine learning tasks like Optical Flow Estimation (OFE) presents challenges, as the learning mechanisms in biological and ML systems fundamentally differ. Biological visual systems undergo complex developmental stages with substantial plasticity[57], allowing adaptive refinement of neural circuits; a trait that is not easily replicated in current ML architectures [58]. In contrast, ML algorithms typically learn in specialized, task-specific ways, often limited by the artificial nature or size of synthetic OFE datasets [59]. Recent works, such as [60], suggest that RWs can be useful in foundational stages of visual learning, particularly for tasks involving basic spatiotemporal structures, rather than high-level visual processes like OFE. Thus, while RWs offer valuable insights into biological visual development, directly applying them to ML-based OFE would likely require rethinking ML-paradigms to bridge the gap between early-stage neural adaptation and advanced visual tasks.

5 Conclusion

This work presents a novel approach to leveraging RWs for next frame prediction, with the broader objective of enhancing optical flow estimation. Through this investigation, significant differences have been highlighted in how specialized ML models engage with RW stimuli, presenting the limitations of these models in effectively leveraging such biologically inspired data for optical flow estimation. It becomes evident that biological visual systems mature through a far more intricate and convoluted process compared to current ML approaches. This disparity calls for different training stages in ML models, particularly when drawing inspiration from biological systems. Furthermore, this project sets up a basis for using retinal waves to pre-train ML models in motion prediction tasks, on which future work can build to evolve and design models that better replicate the characteristics of biological systems, such as those related to adaptability and plasticity. Progress was also made toward generating RW-based optical flow data, which could enhance the generalization capabilities of existing computer vision models. Although still in an early stage, further refinement in that approach and the presented model adaptation remains for future work to fully harness the potential of RWs for computer vision tasks. Even though the models in the first experiment converged, they did not outperform the RAFT model, and several aspects were not fully explored due to time constraints. In conclusion, this work highlights the potential of biologically inspired approaches while acknowledging the complexity of biological visual systems, suggesting that replicating such processes in ML models may require additional exploration.

References

[1] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," ECCV, 2020. DOI: 10.1007/978-3-030-58536-5_24.

[2] N. Gale, "*Theoretical investigations into principles of topographic map formation and applications*," Ph.D. dissertation, 2022. DOI: 10.17863/CAM.90424.

[3] R. Wong, "*Retinal waves and visual system development.*," Annual Review of Neuroscience, 1999. DOI: 10.1146/annurev.neuro.22.1.29.

[4] T. McLaughlin, C. Torborg, M. Feller, and D. O'Leary, "*Retinotopic Map Refinement Requires Spontaneous Retinal Waves during a Brief Critical Period of Development*," Neuron, 2003. DOI: 10.1016/S0896-6273(03)00790-6.

[5] H. Xu, T. Burbridge, M. Chen, X. Ge, Y. Zhang, Z. Zhou, and M. Crair, "*Spatial pattern of spontaneous retinal waves instructs retinotopic map refinement more than activity frequency*," Developmental Neurobiology, 2015. DOI: 10.1002/dneu.22288.

[6] H. Xu, T. Burbridge, M. Ye, M. Chen, X. Ge, Z. Zhou, and M. Crair, "*Retinal Wave Patterns Are Governed by Mutual Excitation among Starburst Amacrine Cells and Drive the Refinement and Maintenance of Visual Circuits*," Neuroscience, 2016. DOI: 10.1523/JNEUROSCI.3549-15.2016.

[7] H. Kolb, R. Nelson, E. Fernandez, and B. Jones. "*WEBVISION - The organization of the Retina and Visual System, PART VI: Retinal Neurogenesis: Early stages in the development of neurons and pathways.*" (2012), [Online]. Available: https://www.webvision.med.utah.edu/.

[8] Y. Ben-Ari and N. C. Spitzer, "*Phenotypic checkpoints regulate neuronal development.*," Trends Neurosci., 2010. DOI: 10.1016/j.tins.2010.08.005.

[9] T. J. Burbridge, J. M. Ratliff, D. Dwivedi, U. Vrudhula, F. Alvarado-Huerta, L. Sjulson, L. A. Ibrahim, L. Cheadle, G. Fishell, and R. Batista-Brito, "*Disruption of Cholinergic Retinal Waves Alters Visual Cortex Development and Function*," bioRxiv, 2024. DOI: 10.1101/2024.04.05.588143.

[10] K. Ford, A. Félix, and M. Feller, "*Cellular Mechanisms Underlying Spatiotemporal Features of Cholinergic Retinal Waves*," Neuroscience, 2012. DOI: 10.1523/JNEUROSCI.5309-12.2012.

[11] S. Firth, C. Wang, and M. Feller, "*Retinal waves: mechanisms and function in visual system development*," Cell Calcium, 2005. DOI: 10.1016/j.ceca.2005.01.010.

[12] M. Feller, D. Butts, H. Aaron, D. Rokhsar, and C. Shatz, "*Dynamic Processes Shape Spatiotemporal Properties of Retinal Waves*," Neuron, 1997. DOI: 10.1016/S0896-6273(00)80940-X.

[13] S. Fried, T. Münch, and F. Werblin, "*Mechanisms and circuitry underlying directional selectivity in the retina*," Nature, 2002. DOI: 10.1038/nature01179.

[14] J. Elsstrott and M. Feller, "*Vision and the establishment of direction-selectivity: a tale of two circuits*," Curr Opin Neurobiol., 2009. DOI: 10.1016/j.conb.2009.03.004.

[15] D. Matzakos-Karvouniari, L. Gil, E. Orendorff, O. Marre, S. Picaud, and B. Cessac, "*A biophysical model explains the spontaneous bursting behavior in the developing retina*," Scientific Reports, 2019. DOI: 10.1038/s41598-018-38299-4.

[16] B. Cessac and D. Matzakos-Karvouniari, "*The non linear dynamics of retinal waves*," Physica D: Nonlinear Phenomena, 2022. DOI: 10.48550/arXiv.2308.12579.

[17] D. Karvouniari, "*Retinal waves : theory, numerics, experiments*," HAL code: tel-01818522, Ph.D. dissertation, 2018. [Online]. Available: https://theses.hal.science/tel-01818522.

[18] C. Morris and H. Lecar, "*Voltage oscillations in the barnacle giant muscle fiber*," Biophysical Journal, 1981. DOI: 10.1016/S0006-3495(81)84782-0.

[19] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," eLife, 2019, ISSN: 2050-084X. DOI: 10.7554/eLife.47314.

[20] E. Kartsaki, *Numerical analysis of retinal waves*, Projet de Fin d'Etudes, 2017.

[21] D. Alevi, M. Stimberg, H. Sprekeler, K. Obermayer, and M. Augustin, "Brian2cuda: Flexible and efficient simulation of spiking neural network models on gpus," Frontiers in Neuroinformatics, vol. 16, 2022. DOI: 10.3389/fninf.2022.883700.

[22] J. Gibson, "*The Perception of the Visual World*," Science, 1950. DOI: 10.1126/science.113.2940.535.a.

[23] H. Li and J. Wang, "*Chapter 2 - A Neural Network Model for Optical Flow Computation*," in Neural Networks and Pattern Recognition, San Diego: Academic Press, 1998, pp. 57–76, ISBN: 978-0-12-526420-4. DOI: 10.1016/B978-012526420-4/50003-3.

[24] B. D. Lucas and T. Kanade, "*An iterative image registration technique with an application to stereo vision*," in IJCAI, 1981, pp. 674–679.

[25] G. Farnebäck, "*Two-Frame Motion Estimation Based on Polynomial Expansion*," In: Image analysis, 2003. DOI: 10.1007/3-540-45103-X_50.

[26] D. Heeger, "*Model for the extraction of image flow.*," In: Journal of the Optical Society of America, 1987. DOI: 10.1364/josaa.4.001455.

[27] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "*A Database and Evaluation Methodology for Optical Flow*," International Journal of Computer Vision, 2007. DOI: 10.1007/s11263-010-0390-2.

[28] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "*FlowNet: Learning Optical Flow with Convolutional Networks*," ICCV, 2015. DOI: 10.48550/arXiv.1504.06852.

[29] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "*FlowFormer: A Transformer Architecture for Optical Flow*," ECCV, 2022. DOI: 10.1007/978-3-031-19790-1_40.

[30] O. Ronneberger, P. Fischer, and T. Brox, "*U-Net: Convolutional Networks for Biomedical Image Segmentation*," in MICCAI, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[31] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "*Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation*," PAMI, 2020. DOI: 10.1109/TPAMI.2019.2894353.

[32] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "*FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*," CVPR, 2017. DOI: 10.1109/CVPR.2017.179.

[33] A. Ranjan and M. J. Black, "*Optical Flow Estimation using a Spatial Pyramid Network*," CVPR, 2017.

[34] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "*Learning to Estimate Hidden Motions with Global Motion Aggregation*," ICCV, 2021. DOI: 10.1109/ICCV48922.2021.00963.

[35] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "*GMFlow: Learning Optical Flow via Global Matching*," CVPR, 2022. DOI: 10.1109/CVPR52688.2022.00795.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "*Attention is all you need*," NIPS, 2017. DOI: 10.48550/arXiv.1706.03762.

[37] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "*A naturalistic open source movie for optical flow evaluation*," ECCV, 2012. DOI: 10.1007/978-3-642-33783-3_44.

[38] A. Geiger, P. Lenz, and R. Urtasun, "*Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*," CVPR, 2012.

[39] M. Menze, C. Heipke, and A. Geiger, "*Joint 3D Estimation of Vehicles and Scene Flow*," ISPRS Workshop on ISA, 2015. DOI: 10.5194/isprsannals-II-3-W5-427-2015.

[40] L. Mehl, J. Schmalfuss, A. Jahedi, Y. Nalivayko, and A. Bruhn, "*Spring: A High-Resolution High-Detail Dataset and Benchmark for Scene Flow, Optical Flow and Stereo*," CVPR, 2023. DOI: 10.18419/darus-3376.

[41] A. Ligeralde, Y. Kuang, T. Yerxa, M. Pitcher, M. Feller, and S. Chung, "*Unsupervised learning on spontaneous retinal activity leads to efficient neural representation geometry*," 2023. arXiv: arXiv:2312.02791.

[42] B. Cappell, A. Stoll, W. Umah, and B. Egger, "*ReWaRD: Retinal Waves for Pre-Training Artificial Neural Networks Mimicking Real Prenatal Development*," 2023. arXiv: arXiv:2311.17232.

[43] B. Lansdell, K. Ford, and J. Kutz, "*A Reaction-Diffusion Model of Cholinergic Retinal Waves*," PLoS Comput Biol., 2014. DOI: 10.1371/journal.pcbi.1003953.

[44] K. Godfrey and N. Swindale, "*Retinal Wave Behavior through Activity-Dependent Refractory Periods*," PLoS Comput Biol., 2007. DOI: 10.1371/journal.pcbi.0030245.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "*Deep Residual Learning for Image Recognition*," CVPR, 2016. DOI: 10.1109/CVPR.2016.90.

[46] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "*A Simple Framework for Contrastive Learning of Visual Representations*," ICML, 2020. DOI: 10.48550/arXiv.2002.05709.

[47] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh, "*Pre-training without Natural Images*," ACCV, 2020. DOI: 10.48550/arXiv.2101.08515.

[48] X. Ge, K. Zhang, A. Gribizis, A. Hamodi, A. Sabino, and M. Crair, "*Retinal waves prime visual motion detection by simulating future optic flow*," Science, 2021. DOI: 10.1126/science.abd0830.

[49] A. Alfarano, L. Maiano, L. Papa, and I. Amerini, "*Estimating optical flow: A comprehensive review of the state of the art*," CVIU, 2024. DOI: 10.1016/j.cviu.2024.104160.

[50] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "*A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation*," CVPR, 2016. DOI: 10.1109/CVPR.2016.438.

[51] J. Zheng, S. Lee, and Z. Zhou, "*A Developmental Switch in the Excitability and Function of the Starburst Network in the Mammalian Retina*," Neuron, 2004. DOI: 10.1016/j.neuron.2004.11.015.

[52] A. Paszke, S. Gross, F. Massa, et al., "*PyTorch: An Imperative Style, High-Performance Deep Learning Library*," NeurIPS, 2019. DOI: 10.48550/arXiv.1912.01703. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[53] I. Loshchilov and F. Hutter, "*Decoupled Weight Decay Regularization*," ICLR, 2019. DOI: 10.48550/arXiv.1711.05101.

[54] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[55] G. Aubert and P. Kornprobst, "*Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*," Applied Mathematical Sciences, vol. 147, pp. 340–341, 2006. DOI: 10.1007/978-0-387-44588-5.

[56] D. J. Vita, F. S. Orsi, N. G. Smith, and E. M. Callaway, "*Development and organization of the retinal orientation selectivity map*," Nature Communications, vol. 15, 2024. DOI: 10.1038/s41467-024-49206-z.

[57] V. J. Li, Z. Chorghay, and E. S. Ruthazer, "*A Guide for the Multiplexed: The Development of Visual Feature Maps in the Brain*," Neuroscience, 2023. DOI: 10.1016/j.neuroscience.2022.07.026.

[58] N. Shervani-Tabar, M. Mirhoseini, and R. Rosenbaum, "*Oja's plasticity rule overcomes several challenges of training neural networks under biological constraints*," 2024. arXiv: arXiv:2408.08408.

[59] L. Pandey, S. Wood, and J. Wood, "*Are Vision Transformers More Data Hungry Than Newborn Visual Systems?*" NIPS, 2023. DOI: 10.48550/arXiv.2312.02843.

[60] T. A. Keller and M. Welling, "*Neural Wave Machines: Learning Spatiotemporally Structured Representations with Locally Coupled Oscillatory Recurrent Neural Networks*," in ICML, vol. 202, PMLR, 2023, pp. 16 168–16 189.

[61] *NEF computing platform*. [Online]. Available: https://wiki.inria.fr/ClustersSophia/Clusters_Home.

A Appendix

Here we present the extended retinal waves biophysical model [17] used in the simulations with in Brian2. It consists of $6N$ coupled non linear differential equations, where N is the number of SAC neurons.

$$\begin{cases} C_m \dfrac{dV_i}{dt} = I_{L_i} + I_{C_i} + I_{K_i} + I_{sAHP_i} + I_{A_i} + \sigma\xi_t \\ \tau_N \dfrac{dN_i}{dt} = \Lambda(V_i)(N_\infty(V_i) - N_i) \\ \tau_C \dfrac{dC_i}{dt} = -\dfrac{\alpha_C}{H_X} C_i + C_0 - \delta_C g_C(V_i)(V_i - V_C) \\ \tau_S \dfrac{dS_i}{dt} = \alpha_S(1 - S_i)C_i^4 - S_i \\ \tau_R \dfrac{dR_i}{dt} = \alpha_R S_i(1 - R_i) - R_i \\ \dfrac{dA_j}{dt} = -\mu_A A_j + \beta_A T_A(V_j) \end{cases} \tag{10}$$

where V_i, the membrane potential, is influenced by 5 current terms, and C_m is the membrane capacitance. There is a leak current $I_{L_i} = -g_L(V_i - V_L)$, a calcium current $I_{C_i} = -g_C M_\infty(V_i)(V_i - V_C)$, a potassium current $I_{K_i} = -g_K N_i(V_i - V_K)$, a slow after hyperpolarization (sAHP) current $I_{sAHP_i} = -g_{sAHP}(R_i^4)(V_i - V_K)$, and finally the *Ach* current, $I_{A_i} = -g_A(V_i - V_A)\sum_{j\in\mathcal{B}_i} \frac{A_j^2}{\gamma_A + A_j^2}$. Note that I_{A_i} is a synaptic term, modeling the cholinergic coupling of the SAC cells, where $\mathcal{B}_i$ is the neighbourhood of the neuron i, and i is the neuron index in the 2D lattice. This coupling relies on nichotinic acetylcholine receptors, which respond to *Ach*, and are responsible for the mutual excitation between SACs and stage II wave propagation.

There are two gating variables: N (voltage-gated K^+ channel), R (Ca^{2+}-gated K^+ channel). I_K depends on the N_i gating variable which is a fast variable, along with V_i. I_{sAHP} is a calcium-gated potassium current, depends on the R gating variable (slow variable), with a fourth power to model the four bound terminals needed to open the corresponding channel. Four calcium ions bind to one saturated calmodulin complex (CaM), four CaMs are needed to activate the Ca^{2+}-gated K^+ channel. I_A depends on the extracellular acetylcholine concentration A_i, emitted by neuron i. For noise induced bursting, a parameter σ controls the whitenoise ξ on the voltage term.

C_i models the intracellular calcium concentration, with C_0 the baseline concentration and tuning parameters α_C, H_X, δ_C. S_i controls the fraction of saturated calmodulin concentration, with tuning parameter α_S. $\tau_N, \tau_C, \tau_S, \tau_R$ are time-scale separation parameters: V and N are fast, C is medium, and S and R are slow.

Below, we see the auxiliary functions of the SAC model. M_∞ represents the steady-state activation variable for the *Ca*-dependent K^+ current. $\Lambda(V)$ describes the dynamics of the *Ca*-dependent current, and its effect on the membrane potential. $N_\infty(V)$ represents the fraction of ion channels that are in the open or activated state at a given membrane potential V.

$$M_\infty = \frac{1}{2}\left[1 + \tanh\left(\frac{V_i - V_1}{V_2}\right)\right] \tag{11}$$

$$\Lambda(V) = \cosh\left(\frac{V - V_3}{2V_4}\right) \tag{12}$$

$$N_\infty(V) = \frac{1}{2}\left[1 + \tanh\left(\frac{V - V_3}{V_4}\right)\right] \tag{13}$$

The *Ach* concentration A_j depends on the pre-synaptic cell's (j) voltage and is squared on the synaptic term in V_i as two *Ach* molecules have to bind to the receptor to open the corresponding ion channel. $T_A(V_j) = 1/1 + e^{-\kappa_A(V_j - V_A)}$ and $\mu_A, \kappa_A, \beta_A, V_A$ are parameters fitted from experiments [16].

The model parameters used for the retinal wave simulations are taken from [17][Table 3.1][16][Table 2 and 3] and $\kappa_A = 60V^{-1}$. These parameters are based on biophysical principles, derived from existing literature or optimized through fitting experimental data.

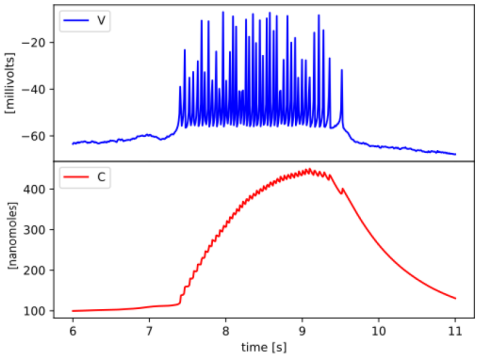


Figure 12: Voltage V and Calcium C of a simulated cell using the SAC model [15]. The bursting is noise driven with $\sigma = 6\ pA\ ms^{-0.5}$.

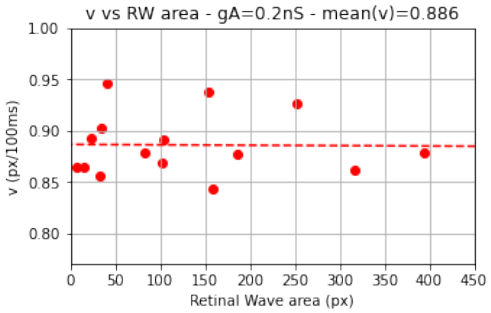
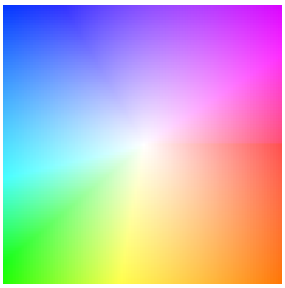
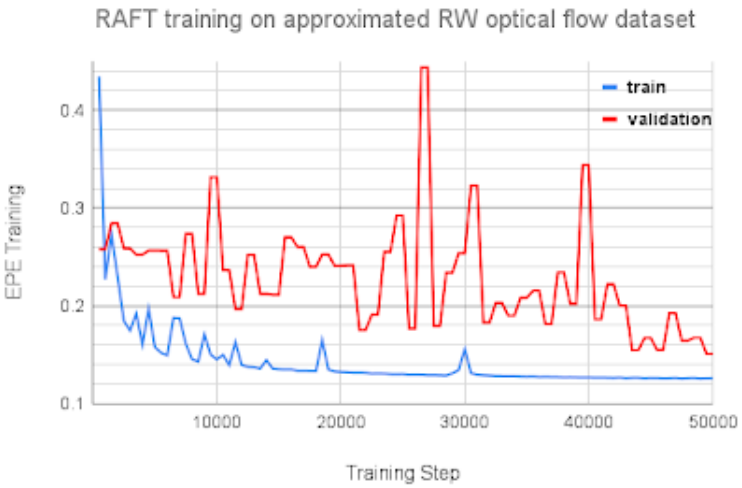


Figure 13: Estimated displacement velocity of RW with $g_A = 0.2nS$. The velocity is estimated based on the radius increase over time. The 15 samples present correspond to different RWs sampled from different timesteps. A permutation test of independence is performed: Pearson $\mathbf{r}: -0.014$, $p - value = 0.95$. The retinal wave velocity is independent of its size.



(a) Colorwheel used to visualize (colormap) most optical flow vector fields present in this report. Origin (no motion) is at the center. Motion velocities in an optical flow vector field are mapped on this colorwheel based on their direction, orientation and magnitude.



(b) Training and validation graph over 50000 steps on the approximated RW optical flow. The chosen frame interval is 200ms with a batch size of 16, learning rate of 5×10^{-4} and a weighting factor of the loss function $\gamma = 0.6$. The remainder of hyperparameters are identical to the standard training of RAFT.

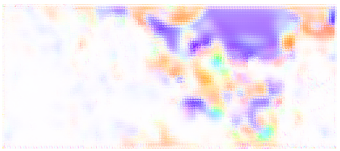
Figure 14: Colorwheel for optical flow visualizations (a) and training and validation graph of RAFT on the approximated RW optical flow (b).

(a) Sintel ground-truth

(b) RW-NFP-pre-trained Sintel training prediction

(c) RAFT Flyingchairs-pre-trained Sintel training prediction

(d) Sintel reference frame



(e) RW-OFE-pre-trained Sintel evaluation

(f) FlyingChairs-pre-trained Sintel evaluation

Figure 15: Optical flow predictions from the *Mountain 1* scene from the *MPI-Sintel* dataset. The first row contains the reference ground-truth (a), the prediction after pre-training for NFP on RWs and fine-tuning only decoder on Sintel (b), the prediction after pre-training for OFE on *FlyingChairs* and fine-tuning only decoder on Sintel (c). The second row contains the first reference frame (d), the equivalent evaluation predictions after training on the approx. RW optical flow (e) and *FlyingChairs* (f), both without fine-tuning on MPI-Sintel. Evaluation EPE on the *clean* version: 3.45(b), 1.95(c), 13.7(e), 2.3(f).

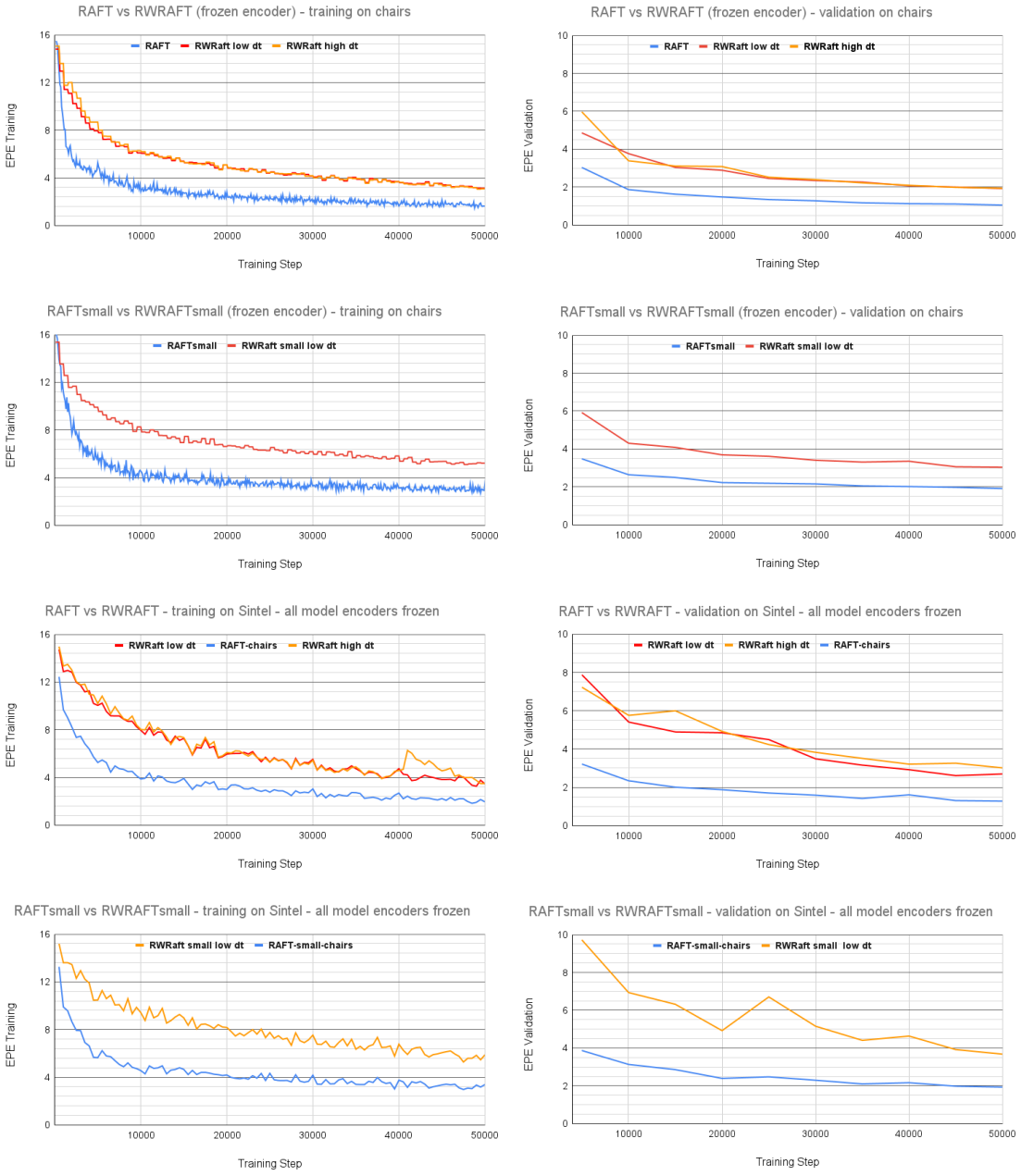


Figure 16: Training results on NFP TL for OFE, using RAFT and RAFT on RWs (RWRaft). **Top four graphs:** training and validation on the FlyingChairs dataset, RWRaft with frozen encoder, pre-trained on RWs for NFP. **Bottom four graphs:** training and validation on the *MPI-Sintel* dataset, RAFT pre-trained on FlyingChairs for OFE, RWRaft pre-trained on RWs for NFP, both encoders are frozen. *high dt* indicates the NFP pre-training was performed on the RW dataset with 300ms frame intervals, while *low dt* corresponds to NFP pre-training on the 100ms RW dataset. The anomaly on the bottom left plot is due to an interrupted training. All models are trained on the NEF cluster[61].

RESEARCH CENTRE
Centre Inria d'Université Côte d'Azur

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399