CONTINUAL UNSUPERVISED LEARNING FOR OPTICAL FLOW ESTIMATION WITH DEEP NETWORKS

Simone Marullo
DINFO, University of Florence
Florence, Italy
simone.marullo@unifi.it

Matteo Tiezzi
DIISM, University of Siena
Siena, Italy
mtiezzi@diism.unisi.it

Alessandro Betti
Inria, CNRS, Lab I3S, Maasai Team
Université Côte d'Azur, Nice, France
alessandro.betti2@unisi.it

Lapo Faggi
DINFO, University of Florence
Florence, Italy
lapo.faggi@unifi.it

Enrico Meloni
DINFO, University of Florence
Florence, Italy
enrico.meloni@unifi.it

Stefano Melacci
DIISM, University of Siena
Siena, Italy
mela@diism.unisi.it

ABSTRACT

In the last few years there has been a growing interest in approaches that allow neural networks to learn how to predict optical flow, both in a supervised and, more recently, unsupervised manner. While this clearly opens up the possibility of learning to estimate optical flow in a truly lifelong setting, by processing a potentially endless video stream, existing techniques assume to have access to large datasets and they perform stochastic mini-batch-based gradient optimization, paired with further ad-hoc components. We present an extensive study on how neural networks can learn to estimate optical flow in a continual manner while observing a long video stream and reacting online to the streamed information without any further data buffering. To this end, we rely on photo-realistic video streams that we specifically created using 3D virtual environments, as well as on a real-world movie. Our analysis considers important model selection issues that might be easily overlooked at a first glance, comparing different neural architectures and also state-of-the-art models pretrained in an offline manner. Our results not only show the feasibility of continual unsupervised learning in optical flow estimation, but also indicate that the learned models, in several situations, are comparable to state-of-the-art offline-pretrained networks. Moreover, we show how common issues in continual learning, such as catastrophic forgetting, do not affect the proposed models in a disruptive manner, given the task at hand.

1 INTRODUCTION

The outstanding results of machine learning-based solutions in the last decade are usually the outcome of well-established practices in which a data collection is carefully prepared and, afterwards, processed by the learning algorithm. Stochastic gradient descent is commonly exploited, packing randomly selected mini-batches of data samples and generally iterating for multiple epochs, in which the same data are exploited over and over. When data are streamed over time, or whenever we want to build a machine learning system that progressively learns from newly available data without forgetting acquired skills, learning becomes more challenging, but the overall setting sounds more natural and interesting (Betti et al., 2021b). Researches in the field of Lifelong/Continual Learning (CL) recently proposed a variety of approaches to try to deal with such a setting (Delange et al., 2021). However, while a lot of emphasis has been devoted to supervised classification problems, little has been done in the context of Continual Unsupervised Learning (CUL), that is even more intriguing and realistic, due to the intrinsic cost in providing supervisions (Madaan et al., 2022). Apart from a few ad-hoc designed benchmarks (Lomonaco & Maltoni, 2017), evaluating CL algorithms is generally done by adapting well-known classic datasets for machine learning (Delange et al., 2021).

Vision offers a natural playground for investigating learning models that continuously evolve over time, especially when considering data streamed from a single visual source. Dynamical environments (camera movements, objects appearing in the scene, changes in lightning, etc.) require algorithms that adapt to the newly available information without losing the capability of making good predictions on the already processed one. Amongst a large set of interesting vision tasks, in the last few years there has been a growing interest in learning how to predict the optical flow in a visual scene, i.e., the apparent motion of individual pixels on the image plane (Horn & Schunck, 1981). A reliable estimation of the optical flow enables the exploitation of semantic correspondences between frames, and it can be

used as precious information for several computer vision-related applications (Betti et al., 2021a). Besides a variety of classic techniques to estimate optical flow (Horn & Schunck, 1981; Lucas & Kanade, 1981; Brox et al., 2004), it has been shown that neural networks can be effectively trained to this end with outstanding results (Dosovitskiy et al., 2015). More recently, the machine learning community moved from supervised approaches (Dosovitskiy et al., 2015; Teed & Deng, 2020) to unsupervised training (Stone et al., 2021). However, to the best of our knowledge, the optical flow problem has not been investigated in the CL perspective, while common training pipelines leverage large datasets processed in an offline manner via randomly shuffling samples in mini-batches.

In this paper we propose to frame the problem of learning to estimate optical flow with neural networks in the context of CUL, considering temporally-correlated data continuously streamed from a single source and performing single-pass online learning. We study and evaluate how neural networks behave in this online context, without buffering the video (with the exception of the previous frame only, that is an input of any optical flow estimation method) and without using any ad-hoc CL-based architectures, somewhat representing the most challenging setting of CUL. We rely on modern 3D virtual environments to generate streams in controlled conditions. In detail, the contributions of this paper are the following. (i) We face a novel problem in the context of CUL, evaluating whether neural networks can learn to predict motion while "observing" a long video stream. We consider different streams in which we inject specific biases to evaluate the behaviour of the networks in critical conditions, proposing proper tricks of the trade to overcome evident issues. (ii) We define an experimental procedure based on 3D virtual environments to generate continuous streams in controlled conditions (different levels of complexity) and we share the data for further research. We include practical considerations on the validation of the model hyper-parameters, that is a frequently overlooked issue. (iii) We also consider the case of a full-length real-world movie and the case of an artificial stream obtained by concatenating the other ones described so far, thus analyzing how the networks adapt to significantly different data and if they forget about the properties of the older streams. (iv) Even if there are no attempts to set a new state-of-the art in the field, we compare with existing state-of-the-art models trained in an offline manner, using large collections of data samples.

Our findings show that learning to predict motion in a CUL setting is indeed possible and effective, with a quality of the predictions that is not far from state-of-the art approaches pre-trained on offline data collections (including supervised approaches). In some cases, our models overcome such pre-trained baselines. An interesting result of our experience is that the neural models usually do not "forget" in an evident manner in this task, and they constantly benefit from handling more and more data over time. Of course, the notion of forgetting must be carefully defined in this task, and when it comes to data with critical biases the network might still experience learning issues, that are solved when applying some simple filtering criteria. This paper is organized as follows. Section 2 describes related work. Section 3 introduces the CUL problem of learning to predict optical flow using a neural network. Section 4 reports the details of our experiments and of our results. Section 5 concludes the paper, with suggestions for future work.

2 Related Work

Our work is framed in the context of online CUL, and it is inspired by existing experiences of learning to estimate optical flow with neural nets (out of CL). To our best knowledge, we are the first to evaluate the feasibility of learning to predict motion when learning in an online CUL setting and, in the following, we describe the existing peculiar activities in CL/CUL, optical flow estimation (without and with neural nets), and also 3D environments, that we used to create the streams we share with this paper.

Continual Learning. When trying to update a trained model in order to conquer skills on a new task or on new domains, neural networks experience a loss of performance in already acquired capabilities (Goodfellow et al., 2014), which is generally termed *catastrophic forgetting*. This leads to the quest for a much better stability-plasticity trade-off (Parisi et al., 2019), with more and more effort in devising systems capable of continuous adaptations. Three main families of approaches have emerged in the context of CL: (*i*) replay methods, (*ii*) regularization methods, (*iii*) parameter-isolation methods–see (Delange et al., 2021) and references therein. In all these cases, a significant portion of the existing research in CL focuses on a specific setting in the realm of computer vision, considering supervised problems (Rebuffi et al., 2017; Gallardo et al., 2021) that sequentially introduce new tasks (e.g., new categories), while here we study a single unsupervised task on a video stream that continuously evolves over time.

Continual Unsupervised Learning. In the CL scientific community, there is still little research on continual exploitation of new data in the lack of further supervision, and on unsupervised tasks. We mention the very recent work of Madaan et al. (2022) for unsupervised representation learning, that is limited to standard classification datasets. A recent trend in neural networks emphasizes the importance of time when developing agents without supervision or under unobtrusive supervision. Inspired by the principle of least action in physics, Betti et al. (2020b) proposed a time-embedded learning theory, that was then evaluated in the maximization of the mutual information on a video

stream paired with a human-like attention model (Tiezzi et al., 2020). Such framework naturally deals in a principled way with all the cases in which data become available over time (Betti et al., 2020a), including CL. The problem studied in this paper is an instance of CUL, and it could be easily framed in such a theory, since it is approached with an online learning strategy.

Optical Flow. Horn & Schunck (1981) were the first to formalize the problem of optical flow estimation as the minimization of a functional, considering brightness invariance and regularization. However, such a method has a number of weaknesses, including those that are due to changes in illumination, occlusions and large displacements. Another popular approach is the one of Lucas & Kanade (1981), who proposed a least-square technique to estimate a locally velocity field, with neighboring pixels that feature the same velocity within small patches. The output is sparse but more robust to outliers. Several authors have extended the Horn & Schunck (1981) algorithm in the attempt of dealing with its weaknesses, such as Brox et al. (2004), who devised a variational algorithm which features robustness to additive illumination changes and better handling of large displacements thanks to a coarse-to-fine strategy.

Optical Flow Estimation with Neural Networks. Dosovitskiy et al. (2015) showed that neural networks (FlowNet) can be effectively trained to predict motion in a supervised manner. Besides being more accurate than classical methods, learned models can be faster at inference because all optimization occurs during training and popular hardware can be efficiently exploited (e.g., GPUs). Supervision typically comes from synthetic data (real-world labeled data are scarce). Teed & Deng (2020) proposed RAFT, which includes recurrent computations on the so-called cost volume, taking into account all the pairs of pixels of the frame. They emulate the steps of a first-order optimization procedure to compute a single high-resolution field, which differs from the prevailing coarse-to-fine design. Recently, unsupervised learning was applied to train neural networks to predict optical flow (Jonschkowski et al., 2020). Unsupervised methods are very attractive since they can leverage abundant collections of videos. Most methods work with the warping paradigm, where the estimated flow field is used to warp back a frame into the previous one, and the network is optimized to reduce the photometric distance loss between the real frame and the reconstructed one. A smoothness penalty, computed on the predicted flow, is typically proposed as regularization. Stone et al. (2021) recently adapted the current best performing supervised model (RAFT) to the unsupervised setting. They proposed SMURF, embracing the self-supervision student-teacher paradigm, trained with extensive data augmentation and occlusion handling. Our work differs from these approaches since, to the best of our knowledge, we are the first to study the optical flow problem in a CUL perspective.

3D Virtual Environments. Simulators based on 3D environments have been proposed in the last years to help machine learning research—see Meloni et al. (2021b) and references therein. Amongst them, SAILenv (Meloni et al., 2021b) was also evaluated in the context of continual learning (Meloni et al., 2021a), even if it lacks the more advanced scene modeling features of ThreeDWorld (Gan et al., 2021), that we used in this work.

3 Continual Learning for Optical Flow Estimation

Amongst a variety of specific formulations, the problem of estimating optical flow is based on the so-called brightness constancy assumption paired with a (usually spatial) regularizer to enforce smoothness in the solution (Brox et al., 2004). In detail, the flow fields $U^\star = \{u^\star_{xy},\ \forall x, \forall y\}$ (horizontal direction) and $V^\star = \{v^\star_{xy}, \forall x, \forall y\}$ (vertical direction) between the image at time $t-\delta$ and the one at time t, for a small δ, are given by

$$\arg\min_{U,V} L(U,V,t) = \arg\min_{U,V} \iint \rho\left(I(x+u_{xy}, y+v_{xy}, t) - I(x,y,t-\delta)\right) + \lambda(\|\nabla u_{xy}\|^2 + \|\nabla v_{xy}\|^2)\mathrm{d}x\mathrm{d}y, \quad (1)$$

being x, y spatial coordinates and $I(x,y,t)$ the intensity of a pixel, while ρ is a penalty function, such as $\rho(a) = a^2$. The scalar $\lambda > 0$ weighs a spatial regularizer, while ∇ is the spatial gradient operator. Computing $I(x+u_{xy}, y+v_{xy}, t)$ for all (x,y)'s basically consists in *warping* the image at time t due to displacements collected in U and V. The formulation of Horn & Schunck (1981) is based on the minimization of a functional that is analogous to the one of Eq. 1 that, however, involves the linearized version of the constancy assumption (Brox et al., 2004).

Suppose we are given a video stream defined on the time interval $[0, T)$, where T could be potentially infinite. The previous functional naturally extends to

$$\arg\min_{\overline{U},\overline{V}} \int_0^T L(U_t, V_t, t)\mathrm{d}t, \quad (2)$$

where $\overline{U} = \{U_t,\ \forall t\}$ and $\overline{V} = \{V_t,\ \forall t\}$, $\delta = \mathrm{d}t$, and where we assumed $t-\delta$ to be equal to 0 if negative. We consider the case in which U_t and V_t are predicted by a neural network f with parameters in θ_t. Recent literature (Dosovitskiy et al., 2015; Teed & Deng, 2020; Stone et al., 2021) explored the possibility of predicting the displacement field by observing a pair of consecutive images, thus $U_t = f_U(I_t, I_{t-\delta}, \theta_t)$, $V_t = f_V(I_t, I_{t-\delta}, \theta_t)$, where we used the subscripts

U and V to distinguish among the two portions of the output of network f and where we used the shorthand I_t to indicate a c-channel image yielded by the stream at time t, slightly overloading the previously introduced notation I, for simplicity. The problem of Eq. 2 can be rewritten as a minimization over $\overline{\theta} = \{\theta_t, \forall t\}$, thus replacing $L(U_t, V_t, t)$ with $\mathcal{L}(\theta_t, t)$ defined as $\mathcal{L}(\theta_t, t) = L(f_U(I_t, I_{t-\delta}, \theta_t), f_V(I_t, I_{t-\delta}, \theta_t), t)$.

Let us suppose that we cast the problem in the discrete case, considering a video stream with frame rate ν and $\delta = \nu^{-1}$. At each discrete time instant t (multiple of ν^{-1}) the stream yields frame I_t, and we assume to have access to the previous frame $I_{t-\delta}$ as well. In this paper we explore the classic causal solution in which the network parameters $\theta_{t+\delta}$ depend on the ones at t, and are obtained by updating the current estimate θ_t exploiting the pair of frames available at time t. This ends up in framing the problem in an online setting, where the learning step is

$$\theta_{t+\delta} = \theta_t - \gamma \frac{\partial \mathcal{L}(\theta, t)}{\partial \theta}\Big|_{\theta=\theta_t}, \tag{3}$$

being $\gamma > 0$ the selected step size. The learning problem is fully unsupervised. We take inspiration from related work, selecting ρ to be the generalized Charbonnier photometric distance (Sun et al., 2010), which proved to be suitable for optical flow learning since it downplays the importance of small deviations, i.e., $\rho(a) = (a^2 + \epsilon)^\alpha$. While many other components could be inherited from related literature to customize Eq. 1 (e.g., edge-aware smoothness, occlusion-oriented terms, etc., see Jonschkowski et al. (2020)), we specifically aim at evaluating a minimalist implementation that can be eventually enriched in many ways. We remark that we are proposing an always-learning approach, where we perform inference and a learning step at each time instant, nicely adapting to time-variant dynamical domains. As a consequence, it is relevant to evaluate the quality of the model while learning is still taking place.

3.1 Update Policies

When dealing with potentially lifelong horizons, it sounds pretty natural to consider the possibility of not updating the model parameters at each t. This not only reduces the computational burden of the backward stage, but also avoids the network to be affected from redundant information that is likely to be present in frames that are close in time. Moreover, applying the vanilla update rule of Eq. 3 at each t is likely to bias the network capabilities towards the information contained in the very recent frames, "forgetting" the oldest ones (Parisi et al., 2019). The notion of "forgetting" might be about the lack of capability of estimating the movements of an object that has not been seen for a long time, or about incorporating lower-level biases that introduce difficulties in predicting motion, even if not related to specific object instances or categories (for example, long periods in which there is almost no motion or motion always in the same direction). We propose to decrease the correlation between consecutive updates with four simple policies, as alternatives to updating the weights on every new frame (Always policy):

- Decimation (Dec): update the model every n frames, $n > 1$. This vanilla rule is basically a way to skip frames in a somewhat uninformed manner, given the unknown properties of the input stream.
- FlowDivergence (Div): after a warmup stage of w frames, to let the model develop early prediction skills, update the model only when the predicted motion is significantly changing. This avoids the model develop direction biases, discarding redundant information coming from objects that move over the same direction for a long time. If $\tilde{m}_t = [\tilde{u}_t, \tilde{v}_t]$ is the vector that collects the average of U_t and the one of V_t, respectively, we update the model only if $\frac{\|\tilde{m}_t - \tilde{m}_z\|_2}{\|\tilde{m}_z\|_2} > l$, with $l > 0$, being z the time at which the last model update was performed.
- Diff (Diff): update the model only if the scene is not static, i.e., if the average L_2 distance over the pixel intensities of the frames I_t and I_{t-1} is greater than q, with $q > 0$.
- FlowMagnitude (Mag): after a warmup stage of w frames, update the model only if a significant portion of the frame is predicted to be moving, filtering out cases of small spurious motion predictions. At least a fraction r of the frame pixels must be predicted to have a displacement vector larger than h to trigger a model update.

Smart update policies can be crucial in general when considering real-world video stream sources. Of course, while Dec and Div are pretty generic, Diff and Mag are appropriate to filter out effects of almost static segments.

3.2 Evaluation Measures and Ground Truth

Existing work on optical flow estimation typically rely on supervised benchmarks in order to evaluate the trained models (Dosovitskiy et al., 2015; Teed & Deng, 2020), that is not practical in the case of real-world unsupervised data. In fact, the most easily accessible measures are the ones based on the photometric similarity, in which the previous frame is compared with the reconstructed frame (i.e., the current one, warped by the predicted U and V). We define

our notion of RECONSTRUCTION ACCURACY as

$$r_{acc}(U, V, t) = \frac{1}{HW} \sum_{x,y} \left[\| I(x + u_{xy}, y + v_{xy}, t) - I(x, y, t - \delta) \|_\infty < \tau \right], \tag{4}$$

where we considered the resolution $W \times H$, and where $[\cdot]$ is 1 if the condition in brackets is true, otherwise it is 0. The ∞ norm has been added to take into account images with $c > 1$ channels (it vanishes in case of $c = 1$). Such measure can be affected by occlusions, but we deem that negligible if the spatio-temporal sampling rates are sufficiently high.

Whenever modern 3D environments for machine learning (see Section 2) are used to generate video streams, motion-related facilities might be available (Meloni et al., 2021b; Gan et al., 2021). The motion field returned by 3D engines is built from full physical knowledge of the environment. Such information might not be recoverable at all just by observing projected 2D views, thus excessively relying on the motion field may be tricky when learning and evaluating optical flow. Moreover, the way pixel velocities are encoded might depend on internal timings of the 3D engine, making it hard to recover displacements for precise comparisons and frame reconstruction. However, in simple streams with fixed camera and a few-known moving objects, we found the motion ground truth returned by SAILenv (Meloni et al., 2021b) or ThreeDWorld (Gan et al., 2021) to be still very precious to evaluate whether the network-predicted flow is about pixels that are marked as moving by the 3D engine or not (thresholding the motion vectors).

Given the rather slippery nature of the optical flow problem (e.g., brightness constancy assumption, occlusions, etc.), we propose an evaluation that focuses on consistency, measured by the (i) RECONSTRUCTION ACCURACY of Eq. 4, and motion detection, where the latter is evaluated by computing what we refer to as (ii) MOTION-F1, that is the F1 score in the 2-class problem of predicting whether a pixel is moving or not, starting from flow predictions. The last ingredient we consider for model selection purposes is the (iii) SPATIAL REGULARITY given by the second term of Eq. 1 (the lower the better). In fact, the optical flow problem typically admits spurious solutions that are very irregular and sparse, but with possibly high reconstruction accuracy. Locally-smooth solutions are much more likely to be useful for downstream applications, and they are captured by low values of the SPATIAL REGULARITY term.

3.3 MODEL SELECTION IN LIFELONG STREAMS

In offline learning tasks, it is common to hold out a portion of the available data as validation set for comparing the quality of different trained models, assuming such a set to reasonably represent the properties of the considered task. When we embrace the lifelong learning perspective, the model selection process and the hyper-parameter validation procedure should change accordingly. For example, the data the agent will be exposed to during its life is huge and variable, and it is almost impossible to determine the right size and content of a hypothetical validation set.

For this reason, we propose to face the validation problem in a greedy manner, with a procedure that, although sub-optimal, avoids to pre-buffer data from the stream. Let us assume that we have enough computational resources to afford spawning a pool P of visual agents running in parallel over the considered stream, each one embodying a different neural model. Such agents are trained with unsupervised learning from time $t = 0$ up to time $t = T_{val} - 1$, with $T_{val} \ll T$, as usual. Then, starting from $t = T_{val}$, we focus on two non-overlapping time windows for measuring performance, each of them of length β. In the first window the agents are still *learning*, while in the second one we keep the network parameters *frozen*. The idea is to look for a model that maximizes the performance in both the windows, thus able to learn by adapting to the dynamical features of the stream (first window) and that embeds meaningful information in the network weights independently on the online update steps (second window). The procedure is very versatile, and it can be eventually repeated spawning another pool of new agents that includes the best model found so far (re-initializing it to make it comparable with the new ones), until the outcome is satisfactory.

As performance measure we considered the RECONSTRUCTION ACCURACY, that can be computed in every stream without requiring any ground truth, paired with the SPATIAL REGULARITY of the predictions. Both the measurements are averaged over the temporal range covered by the two windows (2β), and we refer to them as $\overline{r}_{acc}(z)$ and $\overline{s}_{reg}(z)$ for the z-th agent of the pool P. In particular, we identify the best model as the one with the largest $\overline{s}_{reg}$ whose $\overline{r}_{acc}$ is greater than $c \cdot \max_{j \in P} \overline{r}_{acc}(j)$, being $c \in (0, 1)$, i.e., preferring those models that are a bit less accurate but more regular. Fig. 1 shows a qualitative example of the predictions of a model selected with the proposed criterion, comparing it with a model selection not considering $\overline{s}_{reg}$ at all. This procedure might end up in selecting models that have just learnt to predict smooth uniform motion, almost ignoring the input. This might happen if the considered validation windows cover an almost static video segment. For this reason, the selection policy must automatically exclude models that feature very low standard deviation in the displacement prediction.

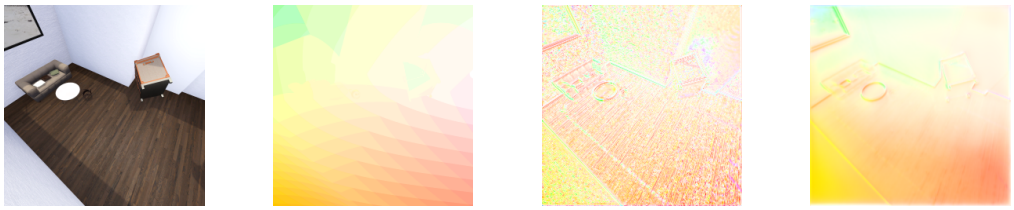

Figure 1: From left to right: frame, ground truth, predictions of a model selected with a reconstruction-accuracy-only selection, predictions of a model selected with the proposed criterion. Motion is given by camera rotation, so that accurate solutions should be smooth. See Sec. C.6 for details on the visualization.

4 EXPERIMENTS

Our experiments have been performed on five different visual streams, referred to as A, B, C, M, and ABCM. The first three ones have been created by ourselves, the 4th one is a real-world movie, while the last one is the concatenation of all the streams. These streams are longer than the ones in popular optical flow benchmarks, and they are designed with increasing level of complexity. We recall that we are not looking for shuffled collections of frame pairs, but for data that naturally evolves over time. We created A, B, C using ThreeDWorld (TDW) (v1.9.1) (Gan et al., 2021), a platform for physical simulation in virtual worlds based on Unity (popular game engine). Manipulating the objects dynamics, we created three photo-realistic living-room-like scenes in which objects move and bounce, and that can provide potentially-endless streams of visual data with dynamics that are not determined by an artificial loop. In detail:

- STREAM A: the simplest one, it features translation and rotation of a single object that never leaves the view, similar to FlyingChairs (Dosovitskiy et al., 2015) but with pseudo-realistic rendering.
- STREAM B: it features four different objects randomly moving in a realistic scenario.
- STREAM C: similar to B, even if the camera is now moving. This stream is much more complex, since it features both very slow movements of objects far from the camera and fast motion patterns from close objects/surfaces.
- STREAM M: "1917", a 2019 British war film directed and produced by Sam Mendes. The film has a duration of 119 minutes (approx. 103 without credits) and it appears as a single long continuous take, without artificial cuts.
- STREAM ABCM: the concatenation of all the above streams.

In the case of A, B, and C,[1] we rendered frames at the resolution of 256×256 for 1 hour (at 25fps, 90k frames), while M consists of $\approx$ 150k frames, downsampled at resolution 320×128. See Fig. 2 for a showcase of the streams.

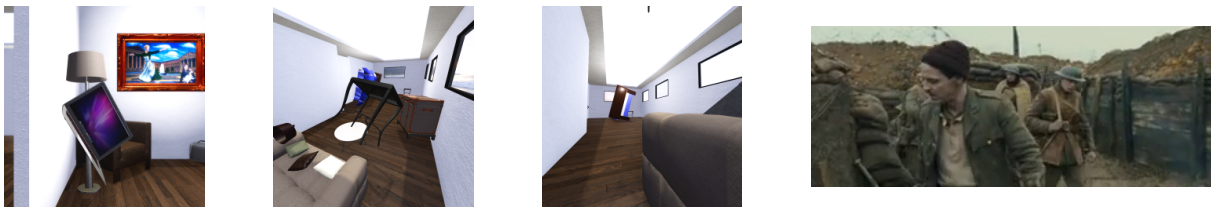

Figure 2: Stream A, B, C, M considered in this experience. We created from scratch and shared the first three ones, where A and B have fixed camera. Stream M is ©Universal Pictures, DreamWorks Pictures.

The aim of our experience is to show that reasonable optical flow estimation can be obtained with common neural models through online learning. For this reason, we have evaluated standard convolutional networks, sometimes inheriting their structure from related work (please refer to the shared code for all the internal details):

- RESUNET: inspired by U-Net (Ronneberger et al., 2015), it encodes context information into progressively lower-resolution feature maps, and then upsamples and transforms the data until it matches the original input size. It exploits skip connections, and ResNet18 for the contractive path–see (Zhang et al., 2018).
- FLOWNETS: the optical flow estimation neural architecture proposed by Dosovitskiy et al. (2015).

[1] Pre-rendered streams are available at the link provided in the main page of the code repository: https://github.com/sailab-code/continual-of

- NDCONV: a standard 8-layer convolutional neural network, without any downsampling operators (i.e., the input resolution is kept fixed throughout all the layers).
- DNDCONV: similar to NDCONV, but with dilated kernels on the last 4 layers.

All the models take as input the concatenation of the frames (I_{t-1}, I_t) along the channel dimension. We adopted typical parameters for the Charbonnier distance ($\alpha = 0.5, \epsilon = 0.001$), as in (Stone et al., 2021). Regarding the model selection procedure (Section 3.3, with $T_{val} = 14$ minutes, 21k frames, and $\beta = 1$ minute), we picked $c = 0.95$, and we tuned λ (Eq. 1) and the optimization parameters, considering learning rate γ and also adding a weight decay term controlled by ξ. Finally, we considered adaptive step sizes given by Adam/AdamW – PyTorch. We selected the best model among: $\lambda \in \{0.01, 0.05, 0.1, 0.5, 0.5, 1.0, 5.0\}$, $\gamma \in \{5 \cdot 10^{-6}, 10^{-5}, 5 \cdot 10^{-5}, 10^{-4}\}$, $\xi \in \{0, 0.01\}$.[2] Concerning update policies, we set $w = 15$ minutes $= 22.5$k frames and we compare DIFF with $q = 0.001$, MAG with $r = 0.02$ and $h = 0.2$, DIV with $l = 0.05$.

Networks are randomly initialized and they learn over the whole stream, accumulating running averages of the metrics of interest. This is what we refer to as *"learning"* setting, different from the one in which we expose the networks to another repetition of the stream keeping *"frozen"* the model parameters obtained at the end of the *"learning"* stage. Results on RECONSTRUCTION ACCURACY are based on $\tau = 0.025$ (i.e., 2.5% of the $[0, 1]$ range), while for MOTION - F1 a pixel is marked as static if its flow has L_∞ norm < 0.5 (see Section 3.2). We avoid reporting standard deviations over multiple runs, since they are very small (we report them in the Appendix). Concerning computation time, a typical run would require about 2 hours (when including *"learning"* and *"frozen"* settings) and overall, we can estimate a total compute budget of 250 hours.

4.1 RESULTS AND DISCUSSION

Table 1: RECONSTRUCTION ACCURACY (%) on the considered streams/settings. We report the best result between the model that always updates its parameters (ALWAYS) and the one that uses DIV update policy (adding a * when results from DIV is reported). Best results among the models considered in the proposed experience are in bold.

	Model	LEARNING					FROZEN				
		A	B	C	M	ABCM	A	B	C	M	ABCM
	RESUNET	86.7	**76.8**	92.3	85.6	87.4	84.0	**78.6** *	90.0 *	86.1	**88.4**
	NDCONV	**87.4**	76.3	92.9	**87.3**	**87.5**	86.0	77.5	91.2	**87.9**	87.7
	DNDCONV	86.0	76.3 *	93.0	86.4	86.9	**86.2**	77.3 *	91.1	86.5 *	87.3
	FLOWNETS	83.7	73.4 *	**93.1**	82.7	84.3	81.7 *	69.3 *	**91.8**	83.3	84.3
REFERENCE	NULL	65.2	65.9	90.5	74.6	74.1	65.2	65.9	90.5	74.6	74.1
REFERENCE	FLOWNETS	76.2	69.3	91.4	79.5	79.2	76.2	69.3	91.4	79.5	79.2
REFERENCE	RAFT-SMALL	83.9	73.9	94.6	86.3	84.9	83.9	73.9	94.6	86.3	84.9
REFERENCE	SMURF	87.6	79.8	95.8	90.1	88.6	87.6	79.8	95.8	90.1	88.6
REFERENCE	RAFT	85.9	77.2	95.4	88.0	86.9	85.9	77.2	95.4	88.0	86.9

In Tab. 1 we report results from both our online-trained models (top rows) and, as a bare reference, from popular state-of-the-art offline-pretrained models (bottom rows) for optical flow estimation. For comparison, we also show a dummy predictor (NULL) which always predicts zero flow. It is evident that online-trained models are able to learn to predict optical flow in a satisfactory way, well overcoming the reference NULL predictor and, surprisingly, also some state-of-the-art offline-pretrained models. Pretrained FLOWNETS (Dosovitskiy et al., 2015) works significantly worse than our models, since it is designed and trained mostly on large-displacement flows, while we considered pretty smooth streams. On the other hand, we acknowledge that other offline-pretrained models such as RAFT (Teed & Deng, 2020) and SMURF (Stone et al., 2021) often achieve higher performance. Of course, our models (top rows) start learning from scratch, thus the outcome of the *"learning"* setting is negatively biased by the inevitably low performance in the early stages of life.[3] Moreover, the reference state-of-the-art exploits several tricks to improve their quality (edge awareness, occlusion detection, etc.), while our models are working in a plain, vanilla setting. Nonetheless, it is interesting to notice that RAFT-SMALL (Teed & Deng, 2020) does not perform largely better. More importantly, Tab. 1 shows that the distance between our models and the offline-pretrained ones is further reduced, and sometimes overturned, if we employ a longer stream (ABCM), confirming the capability of improving over longer horizons. When considering the

[2] Due to the artificial nature of the streams, we simulated the case in which P is large enough to include all the desired agents, by training them on the same data.

[3] We remark that the networks benefit from even longer exposure to the data, further reducing the gap with offline-trained state-of-the-art (see Appendix, Tab. 10, where we repeated each stream twice).

"frozen" setting, we can clearly see that the performance of our models are even slightly better than while learning, indicating that the networks are not simply adapting to the last seen data. Overall, NDCONV behaved pretty well in all the considered streams. All the policies of Section 3.1 were evaluated, with only DIV that allowed to improve the always-update case. For this reason, in Tab. 1 we marked those cases in which we found such an improvement. Fig. 3 reports a qualitative visualization of the motion predicted on a frame from stream M.

(a) Frame (b) SMURF (c) RESUNET

(d) FLOWNETS (e) NDCONV (f) DNDCONV

Figure 3: Qualitative comparison of different models on a scene from STREAM M, where two soldiers are talking while slightly moving their bodies. SMURF (offline pre-trained) output is pretty clear but it suppresses small movements (near the cannon two soldiers are moving, one towards the right, the other one is lowering on his knees, only partially detected) and sometimes oversimplifying. The online-trained models are more sensitive to small changes. The output of NDCONV and DNDCONV is quite similar to RESUNET, while FLOWNETS overemphasizes the slight motion of the man in the foreground and results to be very blurry due to its architecture. See Sec. C.6 for details on the visualization.

The results of Tab. 1, *"frozen"* setting, seem to suggest that no evident effects due to catastrophic forgetting are taking place. In fact, differently from other vision tasks, there is a big chance that new frames can be effectively handled leveraging previously acquired skills, so that interference is reduced. However, we are left with the open question on whether the reported averaged measurement is actually the outcome of very unstable predictions over time. In Fig. 4, we report the evolution of the RECONSTRUCTION ACCURACY, measured on time windows of 1-minute length, RESUNET (similar behaviours were observed for the other networks). Predictions are quite stable in A, B, C, and more variable in M, as expected, since the movie includes significantly larger variability with respect to the other streams. Overall, we do not observe performance drops in the earliest time windows–see also Tabs. 6, 7 in the Appendix.

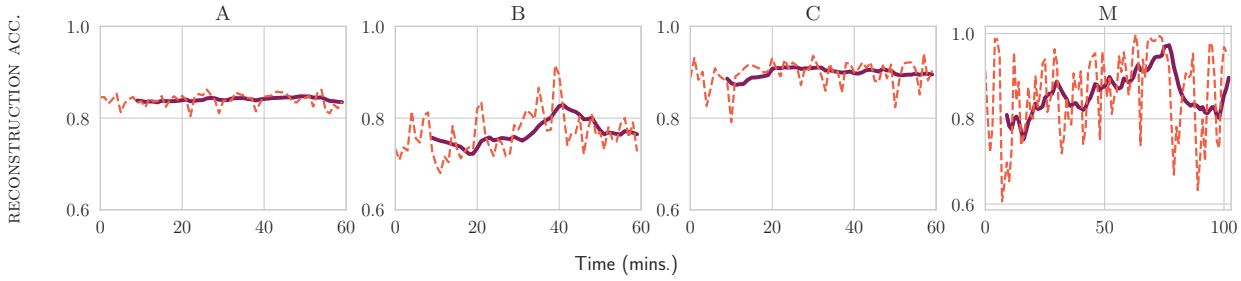

Figure 4: RECONSTRUCTION ACCURACY over the whole stream of model RESUNET, computed in small windows of 1-minute length (dashed). The continuous line is a moving average (10-minute interval) of the window-oriented results. Model weights are in the *"frozen"* setting.

The positive impact of DIV criterion is more evident when measuring MOTION-F1 in those streams where the camera is fixed (in the other streams, almost everything is moving), as shown in Tab. 2. DIV helps in increasing the discriminative power of motion detection, especially in the *"frozen"* setting. The different neural models seem to have similar performance in the given task, apart from FLOWNETS. Stream B is more challenging, with multiple and smaller moving objects, so there might be larger prediction errors. We further inspected this result in Fig. 5, where we compared the MOTION-F1 (frozen) computed on the last minute of each stream (on which we expect less forgetting, since it is

near to the moment we froze the weights, i.e. the end of learning) and on the first minute of the stream (oldest data, prone to forgetting). The DIV strategy can reduce the gap between metrics measured at different time instants (this suggests that the learnt solution is a bit more general). We also show that a simple DEC policy decreases the model performance, at least on medium-length streams as the ones we tested. Overall, some slight forms of performance reduction/forgetting are observed (whenever below the dashed diagonal), but they cannot be classified as catastrophic. From a visual inspection, the first and last minute appear to be quite similar in terms of difficulty.

Table 2: MOTION-F1 in fixed-camera streams. The * symbol has the same meaning as in Tab. 1.

Model	LEARNING		FROZEN	
	A	B	A	B
RESUNET	0.829	0.661 *	**0.834** *	0.712 *
NDCONV	**0.840**	0.667	0.818 *	**0.738** *
DNDCONV	0.836	**0.682** *	0.827	0.701 *
FLOWNETS	0.771 *	0.623 *	0.784 *	0.425 *

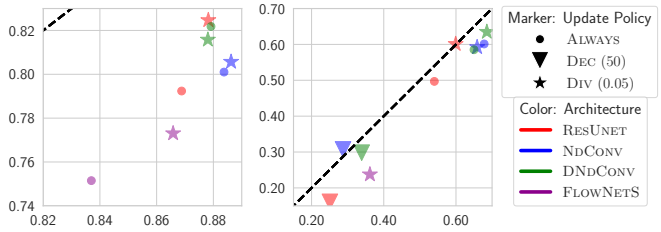

Figure 5: MOTION-F1 on recent (x) and older data (y) on stream A (left) and B (right). DIV policy (with $l = 0.05$) yields improvements in most of the cases. Some low-performance configurations are out of the scope of the plot.

We conducted a further experience in order to evaluate the sensitivity of the networks in presence of motion-related data biases. We simulated the case in which the scene stands still for a significant amount of time before letting objects move again. This is actually a very likely setting of several real-world scenarios (e.g., surveillance systems, public cameras, etc.). We tested the criteria presented in Section 3.1 in all our streams, adding 5 minutes of pause after every 5 minutes of playback. Of course, we focused on DIFF and MAG that are specifically designed to handle static shots, and we kept also DIV due to its promising behaviour in the previous experiments, together with the vanilla case with no special policies (ALWAYS). Fig. 6 reports the results in stream A with two networks (similar results are obtained with the other architectures and streams, see the Appendix, Tab. 8, Tab. 9). While in the *"learning"* case networks are able to learn and predict in an appropriate way, moving to the *"frozen"* setting we get a huge performance drop in ALWAYS and, partially, in DIV, suggesting that the networks are embedding little global information into their weights, and rather mostly latching the unsupervised signal. Differently, the DIFF and MAG criteria significantly help. This experience, when paired with the previous ones, remarks the importance of both DIFF (or MAG) and DIV criteria, to cope with static shots and to filter out other redundant portions of the stream.

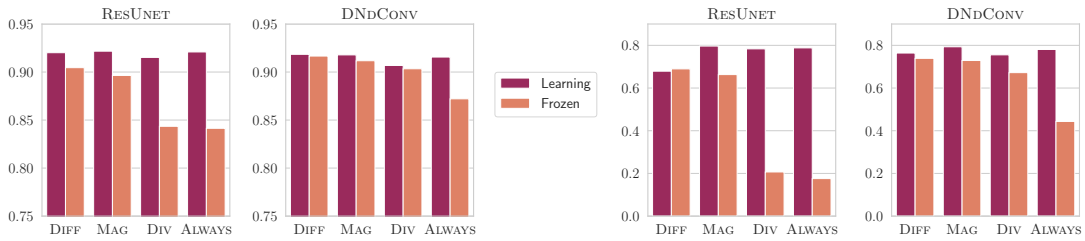

Figure 6: RECONSTRUCTION ACCURACY (left) and MOTION-F1 (right) with two sample architectures (stream A, with injected static shots). Sequence-filtering criteria significantly improve the quality of the solutions (frozen weights).

In the spirit of evaluating even longer continual learning settings, we deepen our analysis in stream ABCM (see Tab. 11, Tab. 12, Appendix, for more results). The plots of Fig. 7 start in *"learning"* mode, while the rightmost part, after the vertical dotted line, is about the *"frozen"* mode. Together with the already discussed DIV criterion, we also considered the not-yet promising DEC, as well as a reference criterion that RESETS the network weights at the beginning of each sub-stream and learns with the ALWAYS policy. The informed DIV strategy ($l = 0.05$) confirms its versatility, and DEC ($n = 150$) also reduces the gap with the other criteria on the very long run, even if the former is still preferable. There is an evident margin between RESET and the best competitors, showing that starting to process a new sub-stream from an already settled configuration (although obtained on different streams) is beneficial–e.g., see sub-stream C. Again, we observe no evident forgetting due to tuning on data from other sub-streams, and the networks benefit from learning on a larger variety of data.

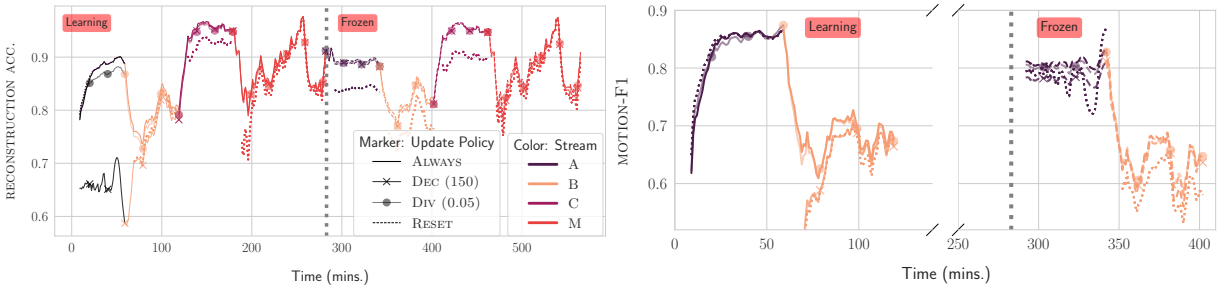

Figure 7: Left: RECONSTRUCTION ACCURACY in the concatenated stream ABCM. Right: MOTION-F1 in ABCM, focussing on the time range of sub-streams A and B (where the said metric is well-defined); moving average (10-minute interval). Being exposed to other streams improves the performance, both in *"learning"* (leftmost part of each plot, before the vertical dotted line) and *"frozen"* settings (rightmost part).

In order to better grasp the benefits of using a neural network to predict optical flow compared to a classic iterative algorithm such as the one of Horn & Shunck (HS), we evaluated our GPU-based implementations of HS (30 or 200 iterations, and whose smoothness coefficient λ was validated with the same procedure previously described) with the neural nets, reporting the MOTION-F1 results in Tab. 3 (see the Appendix for more results), after having learnt from long streams. Neural networks increase the quality of predicted flow with respect to the classic HS algorithm. This consideration also holds when a warm start is used in HS (i.e., initializing the flow estimation with flow prediction from the previous time instant). Concerning computational demands, Fig. 8 shows that, for example, the RESUNET model is much faster than HS (200 iter.) when predicting the optical flow, while it is of comparable speed when accounting for the learning (backward) phase too. The already discussed benefits of the update policies, such as DIV, indicate that the backward step is not always needed, thus saving a significant amount of time and making the neural network-based solutions more attractive.

Table 3: MOTION-F1, comparing Horn & Shunck (HS) flow, with (*w.s.*) and without *w*arm *s*tart, and our neural models, streams A and B. To emphasize the data efficiency of the nets we consider the *"learning"* setting when the stream is shown a 2nd time (LEARNING+) and the *"frozen"* setting after having learnt from ABCM (FROZEN+).

Model	LEARNING+		FROZEN+	
	A	B	A	B
HS 30	0.574	0.369	0.574	0.369
HS 30 (w.s.)	0.649	0.489	0.649	0.489
HS 200	0.462	0.322	0.462	0.322
HS 200 (w.s.)	0.683	0.462	0.683	0.462
RESUNET	0.858	0.707	0.813	0.664
NDCONV	0.872	0.693	0.848	0.633
DNDCONV	0.869	0.715	0.845	0.645
FLOWNETS	0.800	0.638	0.774	0.617

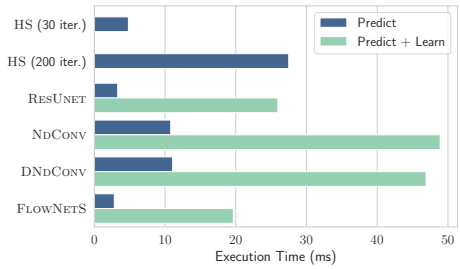

Figure 8: Run-times of our implementations (prediction time and prediction+learning/backward time).

5 CONCLUSION AND FUTURE WORK

We described a novel experience in the context of Continual Unsupervised Learning, focusing on the problem of learning to predict optical flow by letting a neural network "watch" a potentially life-long video stream. Results on streams we created from scratch using 3D environments are not far from the ones yielded by advanced state-of-the-art models pre-trained offline. We proposed simple criteria to filter the input data, and we focused on a model selection procedure that copes well with the optical flow prediction problem. Future work includes the joint estimation of motion and pixel-level visual representations, together with novel procedures to evaluate the resulting models.

ACKNOWLEDGMENTS

This work was partly supported by the PRIN 2017 project RexLearn, funded by the Italian Ministry of Education, University and Research (grant no. 2017TWNMH2) and by the French government, through the 3IA Côte d'Azur,

Investment in the Future, project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

REFERENCES

Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. A database and evaluation methodology for optical flow. In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007. doi: 10.1109/ICCV.2007.4408903.

Alessandro Betti, Marco Gori, Simone Marullo, and Stefano Melacci. Developing constrained neural units over time. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020a. doi: 10.1109/IJCNN48605.2020.9207028.

Alessandro Betti, Marco Gori, and Stefano Melacci. Cognitive action laws: The case of visual features. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):938–949, 2020b. doi: 10.1109/TNNLS.2019.2911174.

Alessandro Betti, Giuseppe Boccignone, Lapo Faggi, Marco Gori, and Stefano Melacci. Visual features and their own optical flow. *Frontiers in Artificial Intelligence*, 4, 2021a.

Alessandro Betti, Marco Gori, Stefano Melacci, Marcello Pelillo, and Fabio Roli. Can machines learn to see without visual databases? *35th Conference on Neural Information Processing Systems (NeurIPS 2021), Data-Centric AI Workshop*, abs/2110.05973, 2021b. URL https://arxiv.org/abs/2110.05973.

Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pp. 25–36. Springer, 2004.

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.) (ed.), *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pp. 611–625. Springer-Verlag, October 2012.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. doi: 10.1109/TPAMI.2021.3057446.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766, 2015. doi: 10.1109/ICCV.2015.316.

Jhair Gallardo, Tyler L. Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. *CoRR*, abs/2103.14010, 2021. URL https://arxiv.org/abs/2103.14010.

Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K. Yamins. Threedworld: A platform for interactive multi-modal physical simulation, 2021.

Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6211.

Berthold Horn and Brian Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 08 1981. doi: 10.1016/0004-3702(81)90024-2.

Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. In *European Conference on Computer Vision*, pp. 557–572. Springer, 2020.

Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 17–26. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/lomonaco17a.html.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pp. 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Hrka5PA7LW.

Enrico Meloni, Alessandro Betti, Lapo Faggi, Simone Marullo, Matteo Tiezzi, and Stefano Melacci. Evaluating continual learning algorithms by generating 3d virtual environments. *arXiv preprint arXiv:2109.07855*, 2021a.

Enrico Meloni, Luca Pasqualini, Matteo Tiezzi, Marco Gori, and Stefano Melacci. Sailenv: Learning in virtual visual environments made simple. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 8906–8913. IEEE, 2021b.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pp. 234–241, 10 2015. ISBN 978-3-319-24573-7.

Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3887–3896, June 2021.

Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2432–2439, 2010. doi: 10.1109/CVPR.2010.5539939.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

Matteo Tiezzi, Stefano Melacci, Alessandro Betti, Marco Maggini, and Marco Gori. Focus of attention improves information transfer in visual features. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22194–22204. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/fc2dc7d20994a777cfd5e6de734fe254-Paper.pdf.

Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

A Neural Architectures

The experimental campaign was carried out evaluating the performance of the proposed approach using different types of deep convolutional architectures.

The ResUNet network[4] is a U-Net-like architecture (Ronneberger et al., 2015). The contractive backbone is composed by the popular ResNet18 (notice that we removed the Batch Normalization layers). Skip connections are leveraged, repeatedly concatenating current feature maps with lower-level feature maps and applying upsampling combined with convolution.

The FlowNetS architecture was proposed by Dosovitskiy et al. (2015), who showed that standard convolutional networks are capable of solving the optical flow estimation problem as a supervised learning task, relying on large randomly generated synthetic datasets of 3D moving objects. The network consists of two parts: the prediction layers and the refinement layers. Two different approaches were presented. In the first one (FlowNetS), they stack the two frames as network input. Basically, the prediction part is a convolutional contractive path (where contraction is given

[4]Implementation available at https://github.com/usuyama/pytorch-unet/.

by stride), while the refinement module is responsible of upscaling the flow. We stick with this architecture since the other one (FLOWNETC) is more involved and was shown to provide marginal benefits.

NDCONV is a standard convolutional network composed by 8 layers (each one equipped with 5x5 filters) that maintains the same image-resolution throughout the whole architecture, without any downsamplings/poolings. The layers are composed by `32,64,64,128,128,64,64` filters banks, respectively, and ReLU activation functions.

DILNDCONV is very similar to the latter model, apart from a dilation factor of 2 on the last four layers. See the provided code repository for further details.

In the paper text, we compared the proposed solution with results obtained from offline pretrained baselines; we used 'MPI-Sintel' (Butler et al., 2012) training weights when available, since that dataset is quite visually similar to streams A, B, C. We used 'Flying Chairs' (Dosovitskiy et al., 2015) weights in the case of FLOWNETS.

B CODE & VISUAL STREAMS

The reader can download our implementation (and data) from the following code repository: https://github.com/sailab-code/continual-of. It includes the code (PyTorch–Modified BSD license) and the instructions to reproduce our results (file `reproduce_runs.txt` contains command lines already prepared with the best-selected hyperparameters, outcome of the model-selection procedure illustrated in Sec. 3.3). See the `README.md` file for further details on the contents of the repository and on how to execute the code (a paper-code mapping of the parameters name is included). In the same files, there are some pointers for the download and installation of baselines.

Moreover, we provide in the same repository (files `tdw_stream_a.py`, `tdw_stream_b.py`, `tdw_stream_c.py`) the code needed to reproduce and, eventually, customize, the TDW streams[5]. In addition, the same streams were pre-rendered and exploited in the experimental section, saving frames and motion information to disk. Such data (MIT license) can be downloaded from the link indicated in the code repository.

Computing infrastructure. The experimental campaign was mainly carried out on two identical machines equipped with Ubuntu 18.04.3 LTS, two NVIDIA GeForce RTX 3090 (24GB) GPUs, Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz, 128 GB of RAM.

We used Numpy (v1.19.5) for saving numerical arrays and we exploited the WandB platform to organize and track our experiments.

B.1 VIDEO STREAMS FOLDERS

We rendered the three scenes and extracted the corresponding visual streams described in Section 4. The `data` folder should contain one subfolder for each one of such visual streams (available in the shared archive with our data). The directory tree of each one of such folders (named after the streams name `stream_a_9424`, `stream_b_8714`, `stream_c_1235`) is composed as follows.

`frames` directory It contains the full-resolution frames, stored in PNG format in subfolders of 100 frames.

`motion` directory It contains the motion arrays (pixel-wise XY displacements, in $(h, w, 2)$-shaped tensors), stored in NumPy format (gzip compressed, .bin extension) in subfolders of 100 frames.

```
from stream_utils import PairsOfFramesDataset

# create PairsOfFramesDataset object
d = PairsOfFramesDataset(root_dir="data/stream_a_9424",
                         force_gray="no", device="cuda")

# getting pair of frames and motion ground truth (if available)
for (old_frame, frame, motion) in d:
    ...
```

Figure 9: Loading the pre-rendered stream A with the `PairsOfFramesDataset` class (PyTorch).

[5]TDW software (v1.9.1) is needed. Available at https://www.threedworld.org/

Loading the pre-rendered scenes. The `PairsOfFramesDataset` class utility in `stream_utils.py` can be used to load prerendered streams. Figure 9 contains a simple code example to load one of the streams and get some data from it.

C Detailed numerical results

C.1 Standard deviations

Throughout the main paper we always reported average numerical results over three runs with different seeds. For the sake of completeness, in Tab. 4, 5 we report the complete results shown in Tab. 1, 2 enriched with the standard deviations over the three runs. We chose to neglect this information in the main paper, since it is always quite small and not very informative.

Table 4: Reconstruction accuracy (%) on the considered streams/settings. We report the best model between the one that always updates its parameters (Always) and Div update policy (adding a * when Div is reported). Best results among the models considered in the proposed experience are in bold.

Model	Learning				Frozen			
	A	B	C	M	A	B	C	M
ResUNet	86.7 ± 0.4	**76.8** ± 0.2	92.3 ± 0.1	85.6 ± 0.2	84.0 ± 0.4	**78.6** ± 0.9 *	90.0 ± 0.8 *	86.1 ± 0.0
NdConv	**87.4** ± 0.3	76.3 ± 0.2	92.9 ± 0.1	**87.3** ± 0.0	86.0 ± 0.2	77.5 ± 0.1	91.2 ± 0.4	**87.9** ± 0.1
DNdConv	86.0 ± 0.2	76.3 ± 0.6*	93.0 ± 0.1	86.4 ± 0.0	**86.2** ± 0.2	77.3 ± 0.2 *	91.1 ± 0.1	86.5 ± 0.1 *
FlowNetS	83.7 ± 0.1	73.4 ± 0.3*	**93.1** ± 0.0	82.7 ± 0.1	81.7 ± 0.1 *	69.3 ± 3.0 *	**91.8** ± 0.2	83.3 ± 0.1

Table 5: Motion-F1 in fixed-camera streams. The * has the same meaning as in Tab. 4.

Model	Learning		Frozen	
	A	B	A	B
ResUNet	0.829 ± 0.008	0.661 ± 0.002 *	**0.834** ± 0.012 *	0.712 ± 0.031 *
NdConv	**0.840** ± 0.006	0.667 ± 0.007	0.818 ± 0.003 *	**0.738** ± 0.015 *
DNdConv	0.836 ± 0.003	**0.682** ± 0.020 *	0.827 ± 0.006	0.701 ± 0.004 *
FlowNetS	0.771 ± 0.002 *	0.623 ± 0.009*	0.784 ± 0.001 *	0.425 ± 0.253 *

C.2 Forgetting

In this section we deepen our analysis on the *forgetting* issue, previously discussed in the paper text (see Fig. 4, 5), where we investigate the impact of forgetting earlier experience by freezing the weights. We show the performance obtained leveraging the different proposed update policies (see Section 3.1) and the alternative of updating the weights on every new frame (Always policy). We show the performance via the Motion-F1 metric (in Tab. 6) and the Reconstruction accuracy (Tab. 7) metric. We report the results obtained on all the streams and in both the *"learning"* and *"frozen"* settings. Additionally, we provide measurements restricted to the first (long-term – Lt) and the last (short-term – St) 1-minute windows (notice that St and Lt have been visualized along the axes of the scatter plots in Fig. 5). The positive impact of the update policies is mostly visible in the *"frozen"* setting. By looking at the relatively small difference between columns Lt and St, we notice that *catastrophic* forgetting does not seem to be an actual issue for the specific task of Optical Flow estimation.

Table 6: MOTION-F1 in fixed-camera streams (streams A and B) in different settings: learning (L), frozen all-stream (F), frozen short-term trial (ST), frozen long-term trial (LT). We compare DEC with $n = 50$, DIV with $l = 0.05$ and ALWAYS policy.

	Policy	STREAM A				STREAM B			
		L	F	ST	LT	L	F	ST	LT
RESUNET	ALWAYS	0.829	0.803	0.869	0.792	0.659	0.618	0.540	0.497
	DEC (50)	0.013	0.024	0.022	0.024	0.114	0.324	0.251	0.161
	DIV (0.05)	0.821	0.834	0.878	0.825	0.661	0.712	0.599	0.601
NDCONV	ALWAYS	0.840	0.807	0.884	0.801	0.667	0.718	0.678	0.601
	DEC (50)	0.479	0.699	0.816	0.704	0.235	0.332	0.287	0.307
	DIV (0.05)	0.833	0.818	0.886	0.806	0.663	0.738	0.659	0.592
DNDCONV	ALWAYS	0.836	0.827	0.879	0.822	0.643	0.715	0.699	0.650
	DEC (50)	0.030	0.237	0.250	0.232	0.196	0.700	0.355	0.339
	DIV (0.05)	0.822	0.823	0.878	0.816	0.682	0.717	0.701	0.686
FLOWNETS	ALWAYS	0.768	0.764	0.837	0.751	0.618	0.059	0.076	0.027
	DEC (50)	0.013	0.001	0.000	0.002	0.027	0.198	0.175	0.086
	DIV (0.05)	0.771	0.784	0.866	0.773	0.623	0.425	0.362	0.237

Table 7: RECONSTRUCTION ACCURACY. Subcolumns as defined in Tab. 6.

	Policy	STREAM A				STREAM B				STREAM C				STREAM M			
		L	F	ST	LT	L	F	ST	LT	L	F	ST	LT	L	F	ST	LT
RESUNET	ALWAYS	86.7	84.0	83.6	84.5	76.8	76.7	72.8	73.3	92.3	89.8	90.4	88.8	85.6	86.1	95.0	91.2
	DEC (50)	65.4	65.7	55.0	66.1	67.3	69.5	66.2	67.8	90.4	90.5	90.2	89.3	75.1	76.0	89.1	84.7
	DIV (0.05)	84.7	83.6	83.0	84.0	76.8	78.6	74.6	75.1	90.9	90.0	90.1	89.0	85.1	85.8	94.8	91.0
NDCONV	ALWAYS	87.4	86.0	85.3	86.2	76.3	77.5	72.0	73.3	92.9	91.2	91.3	90.4	87.3	87.9	95.8	92.6
	DEC (50)	74.1	81.5	79.7	81.8	68.0	70.5	66.3	68.3	90.5	90.5	90.3	89.4	74.5	74.5	87.5	83.4
	DIV (0.05)	86.0	85.7	84.3	86.3	76.1	77.0	72.1	73.1	92.2	89.9	90.6	89.1	86.9	87.7	95.5	92.3
DNDCONV	ALWAYS	86.0	86.2	85.1	86.5	75.1	76.7	71.3	72.5	93.0	91.1	91.5	90.4	86.4	86.3	94.8	91.5
	DEC (50)	65.6	69.1	58.0	69.9	67.2	68.2	65.0	66.2	90.5	90.5	90.2	89.3	74.4	74.5	87.6	83.4
	DIV (0.05)	84.7	85.9	84.9	86.2	76.3	77.3	71.8	72.9	92.2	90.4	90.8	89.7	86.2	86.5	94.9	91.5
FLOWNETS	ALWAYS	83.7	81.2	81.2	81.3	73.2	66.4	64.1	65.5	93.1	91.8	92.3	90.8	82.7	83.3	92.7	88.6
	DEC (50)	64.7	65.0	54.1	65.5	65.8	67.7	64.1	66.2	90.0	90.2	90.1	88.9	74.2	74.4	87.3	83.4
	DIV (0.05)	83.1	81.7	81.3	81.5	73.4	69.3	64.8	67.1	92.4	91.4	92.0	90.3	82.4	82.9	92.6	88.4

C.3 DATA BIASES

In the paper text we have also presented experiments designed to investigate the effect of specific biases in the input stream. In particular, in Fig. 6 (stream A) we show what happens when playback and pause are interleaved in order to simulate static segments of real-world streams. In Tab. 8, 9 we report results in *"learning"* and *"frozen"* settings on all the streams and the architectures.

C.4 LONGER STREAMS

One could be interested in quantitatively assessing the impact of long-stream exposure. For this reason, we provide in Tab. 11, 12 numerical results (running averages) for the stream concatenation experiment (see paper text, denoted with ABCM), considering the ALWAYS update policy and the proposed alternatives (DIFF, MAG, DIV). Fig. 7 in the main paper includes visualizations of the evolution of the metrics when considering the RESUNET model. Such information can be found, aggregated over time, in top rows of Tab. 11, 12.

A different experiment (Tab. 10) concerns updating the weights on a second exposure to the streams, that significantly improves the metrics (overcoming the baseline on two streams out of three).

Table 8: MOTION-F1 when injecting static shots in fixed-camera streams, varying the update policy. Learning (L), Frozen (F). We compare DIFF with $q = 0.001$, MAG with $r = 0.02$ and $h = 0.2$, DIV with $l = 0.05$ and ALWAYS policy.

	Policy	STREAM A		STREAM B	
		L	F	L	F
RESUNET	ALWAYS	0.788	0.176	0.596	0.447
	DIFF	0.679	0.690	0.588	0.651
	MAG	0.797	0.663	0.618	0.624
	DIV	0.784	0.207	0.634	0.486
NDCONV	ALWAYS	0.820	0.541	0.646	0.521
	DIFF	0.788	0.714	0.629	0.640
	MAG	0.795	0.745	0.640	0.644
	DIV	0.769	0.462	0.650	0.483
DNDCONV	ALWAYS	0.781	0.443	0.655	0.588
	DIFF	0.764	0.739	0.627	0.658
	MAG	0.793	0.729	0.659	0.666
	DIV	0.755	0.672	0.655	0.617
FLOWNETS	ALWAYS	0.730	0.751	0.612	0.496
	DIFF	0.730	0.788	0.569	0.609
	MAG	0.730	0.792	0.616	0.641
	DIV	0.729	0.711	0.612	0.187

Table 9: RECONSTRUCTION ACCURACY (%) when injecting static shots in the streams, varying the update policy. Learning (L), Frozen (F). See Tab. 8 for details.

	Policy	STREAM A		STREAM B		STREAM C		STREAM M	
		L	F	L	F	L	F	L	F
RESUNET	ALWAYS	92.1	84.1	85.8	84.0	95.9	95.4	90.9	91.9
	DIFF	92.0	90.5	86.2	86.3	95.1	94.4	88.9	92.1
	MAG	92.2	89.6	86.1	86.7	95.5	95.3	91.0	91.8
	DIV	91.5	84.3	86.4	85.0	94.8	95.1	88.5	91.5
NDCONV	ALWAYS	92.8	88.6	86.1	85.6	96.2	95.3	92.1	92.4
	DIFF	92.9	91.8	86.2	87.2	95.6	92.3	91.1	92.3
	MAG	92.5	91.5	86.0	87.3	96.0	95.5	92.1	92.2
	DIV	91.5	87.3	86.3	84.9	95.5	95.5	92.1	92.1
DNDCONV	ALWAYS	91.6	87.2	86.1	85.9	96.0	95.3	91.8	91.8
	DIFF	91.8	91.7	85.7	86.6	95.5	93.9	90.0	92.0
	MAG	91.8	91.2	86.1	87.3	95.5	95.4	91.8	91.7
	DIV	90.7	90.3	86.2	86.8	95.3	95.4	91.6	91.7
FLOWNETS	ALWAYS	90.8	89.4	85.4	83.9	96.3	95.4	89.3	89.6
	DIFF	90.8	90.9	85.1	85.0	95.4	94.1	87.7	89.1
	MAG	90.8	91.0	85.4	85.5	96.0	95.5	89.2	88.9
	DIV	90.5	89.0	85.3	82.7	95.4	95.1	88.5	88.7

C.5 HS COMPARISON

In the main paper we presented a comparison between the proposed neural-based approach and classic iterative algorithms, e.g. Horn & Schunck (1981). In Tab. 3 we show that the MOTION-F1 obtained in the case of the HS solutions achieves much lower values compared with our proposal, to the point that it is unlikely to be useful for downstream applications. For the sake of completeness, in Tab. 13 we show the RECONSTRUCTION ACCURACY. This appearingly good result highlights that achieving good performance in both the metrics is crucial for a satisfactory solution.

Table 10: RECONSTRUCTION ACCURACY (left) and MOTION-F1 (right) on the shorter streams, *"learning"* setting, presenting the stream a second time.

	Model	A	B	C
	RESUNET	0.902	**0.805**	0.934
	NDCONV	**0.907**	0.790	0.944
	DNDCONV	0.895	0.788	**0.947**
	FLOWNETS	0.863	0.745	0.939
REFERENCE	NULL	0.652	0.659	0.905
REFERENCE	FLOWNETS	0.762	0.693	0.914
REFERENCE	RAFT-SMALL	0.839	0.739	0.946
REFERENCE	SMURF	0.876	0.798	0.958
REFERENCE	RAFT	0.859	0.772	0.954

Model	A	B
RESUNET	0.858	0.707
NDCONV	**0.872**	0.693
DNDCONV	0.869	**0.715**
FLOWNETS	0.800	0.638

Table 11: MOTION-F1 when streams are sequentially shown to the learning agent (A, B, C, M). MOTION-F1 is also measured at frozen weights. We compare DEC with $n = 150$, DIV with $l = 0.05$ and ALWAYS policy.

	Policy	LEARNING		FROZEN	
		A	B	A	B
RESUNET	ALWAYS	0.824	0.709	0.813	0.664
RESUNET	DEC (150)	0.000	0.667	0.787	0.664
RESUNET	DIV (0.05)	0.823	0.702	0.807	0.672
NDCONV	ALWAYS	0.848	0.689	0.848	0.633
NDCONV	DEC (150)	0.069	0.682	0.851	0.630
NDCONV	DIV (0.05)	0.817	0.685	0.851	0.623
DNDCONV	ALWAYS	0.839	0.705	0.845	0.645
DNDCONV	DEC (150)	0.000	0.650	0.845	0.641
DNDCONV	DIV (0.05)	0.824	0.698	0.848	0.636
FLOWNETS	ALWAYS	0.768	0.668	0.774	0.617
FLOWNETS	DEC (150)	0.031	0.604	0.760	0.559
FLOWNETS	DIV (0.05)	0.771	0.663	0.768	0.612

Table 13: RECONSTRUCTION ACCURACY comparing neural models and HS algorithm (with different iteration limits and warm start mode). Here the difference is less evident, but it is important to remark that HS reconstruction accuracy is paired with a poor MOTION-F1, see Tab. 3.

	Model	A	B	C	M
	HS 30 it (w.s.)	0.830	0.787	0.932	0.884
	HS 30 it	0.879	0.822	0.945	0.867
	HS 200 it (w.s.)	0.827	0.782	0.936	0.846
	HS 200 it	0.816	0.790	0.935	0.886
LEARNING+	RESUNET	0.902	0.805	0.934	0.880
LEARNING+	NDCONV	0.907	0.790	0.944	0.894
LEARNING+	DNDCONV	0.895	0.788	0.947	0.878
LEARNING+	FLOWNETS	0.863	0.745	0.939	0.850
FROZEN+	RESUNET	0.892	0.817	0.952	0.877
FROZEN+	NDCONV	0.889	0.805	0.952	0.869
FROZEN+	DNDCONV	0.879	0.790	0.949	0.873
FROZEN+	FLOWNETS	0.847	0.748	0.933	0.845

C.6 OPTICAL FLOW VISUALIZATION

One of the most popular visualization methods for the optical flow predictions involves color coding. In particular, motion vectors from the flow fields U and V are transformed into polar coordinates. Then, each vector direction is associated with a color. Higher brightness in the color space corresponds to larger movements. On the contrary, the

Table 12: RECONSTRUCTION ACCURACY when streams are sequentially shown to the learning agent (A, B, C, M). RECONSTRUCTION ACCURACY is also measured at frozen weights. We compare DEC with $n = 150$, DIV with $l = 0.05$ and ALWAYS policy.

	Policy	LEARNING				FROZEN			
		A	B	C	M	A	B	C	M
RESUNET	ALWAYS	86.4	79.1	95.5	88.0	89.2	81.7	95.2	87.7
	DEC (150)	65.1	76.0	95.2	87.7	88.8	81.0	95.0	87.1
	DIV (0.05)	85.2	78.2	94.9	87.7	88.8	81.0	95.0	87.3
NDCONV	ALWAYS	87.7	79.0	95.4	87.8	88.9	80.5	95.2	86.9
	DEC (150)	66.0	76.9	95.2	87.5	88.5	79.7	95.0	86.6
	DIV (0.05)	85.3	78.1	95.1	87.2	88.1	79.2	94.8	86.2
DNDCONV	ALWAYS	86.1	78.0	95.3	87.7	87.9	79.0	94.9	87.3
	DEC (150)	65.2	74.7	94.9	87.3	87.6	78.2	94.8	87.0
	DIV (0.05)	84.9	77.2	94.8	87.2	87.7	78.2	94.8	87.1
FLOWNETS	ALWAYS	83.7	74.1	94.0	84.9	84.7	74.8	93.3	84.5
	DEC (150)	64.5	72.3	93.8	84.7	84.7	74.6	93.1	84.3
	DIV (0.05)	83.1	73.7	93.6	84.6	84.4	74.7	93.2	84.4

presence of very small or null motion is associated with colors moving towards the white (or the black). One of the most popular color coding is the one presented by Baker et al. (2007), which we report in Fig. 10.

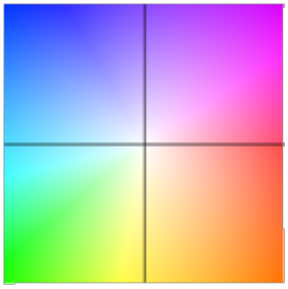

Figure 10: Color coding of optical flow vectors. According to this code, downward movements will be depicted in yellow and leftward movements in cyan. Static regions will be white.