Learning Optical Flow Field via Neural Ordinary Differential Equation

Leyla Mirvakhabova Hong Cai Jisoo Jeong Hanno Ackermann Farhad Zanjani Fatih Porikli
Qualcomm AI Research†
{lmirvakh, hongcai, jisojeon, hackerma, fzanjani, fporikli}@qti.qualcomm.com

Abstract

Recent works on optical flow estimation use neural networks to predict the flow field that maps positions of one image to positions of the other. These networks consist of a feature extractor, a correlation volume, and finally several refinement steps. These refinement steps mimic the iterative refinements performed by classical optimization algorithms and are usually implemented by neural layers (e.g., GRU) which are recurrently executed for a fixed and pre-determined number of steps. However, relying on a fixed number of steps may result in suboptimal performance because it is not tailored to the input data.

In this paper, we introduce a novel approach for predicting the derivative of the flow using a continuous model, namely neural ordinary differential equations (ODE). One key advantage of this approach is its capacity to model an equilibrium process, dynamically adjusting the number of compute steps based on the data at hand. By following a particular neural architecture, ODE solver, and associated hyperparameters, our proposed model can replicate the exact same updates as recurrent cells used in existing works, offering greater generality.

Through extensive experimental analysis on optical flow benchmarks, we demonstrate that our approach achieves an impressive improvement over baseline and existing models, all while requiring only a single refinement step.

1. Introduction

Optical flow (**OF**) is a fundamental challenge in computer vision aiming to estimate a dense displacement field between consecutive video frames. By assuming appearance consistency, where pixels at corresponding locations in these frames share similar appearances, traditional approaches tackle this by solving a differential equation or optimization problem to derive the displacement field.

In recent years, there has been a notable progress in this field, particularly with the emergence of deep learning based models, such as recurrent neural networks (RNNs). Among these, RAFT [40] stands out for its significant accuracy improvement and strong generalizability as compared to previous optimization- and learning-based models. It utilizes a gated recurrent unit (GRU) cell for iterative flow estimation updates, resembling an optimization process. This is a critical component of RAFT and this approach has been widely embraced by the latest state-of-the-art optical flow models, including FlowFormer [14].

While this recurrent approach with GRU cells yields improved performance, it comes with several limitations. Firstly, such a design mandates a discrete setting for the refinement process, which constrains learning precision, especially in scenarios with complex underlying dynamics. Moreover, from a practical standpoint, employing a recurrent neural module requires a prescribed number of iterations. In existing optical flow literature, these iteration counts are defined manually or through grid search. However, such counts may vary between training and testing datasets or due to discrepancies in image-to-image motion. Consequently, there is a lack of a principled way to determine the optimal number of iterations, particularly when deploying the model on new, unseen data.

Another known drawback of RNNs relates to backpropagation-through-time (BPTT) that imposes the difficulties in converging to local optima [2, 42]. In this setting, the optimization is done through unfolded RNN iterations, complicating the modeling of long-term dependencies.

In this study, we take a step back and revisit deep learning based OF estimation, questioning the efficacy of current algorithms in adequately capturing OF dynamics. Starting with the Taylor-expansion of the color or brightness constancy assumption, conventional OF estimation algorithms substitute the temporal derivatives by the pixel-to-pixel differences. Using recurrent layers, on the other hand, necessitates fixing the number of iterations, thereby introducing aforementioned limitations. In contrast, we advocate for approximating these dynamics with neural ordinary differential equations, which offer superior capabilities in modeling time-varying processes.

Unlike previous approaches that use either a GRU cell or a transformer architecture to iteratively compute the flow

† Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

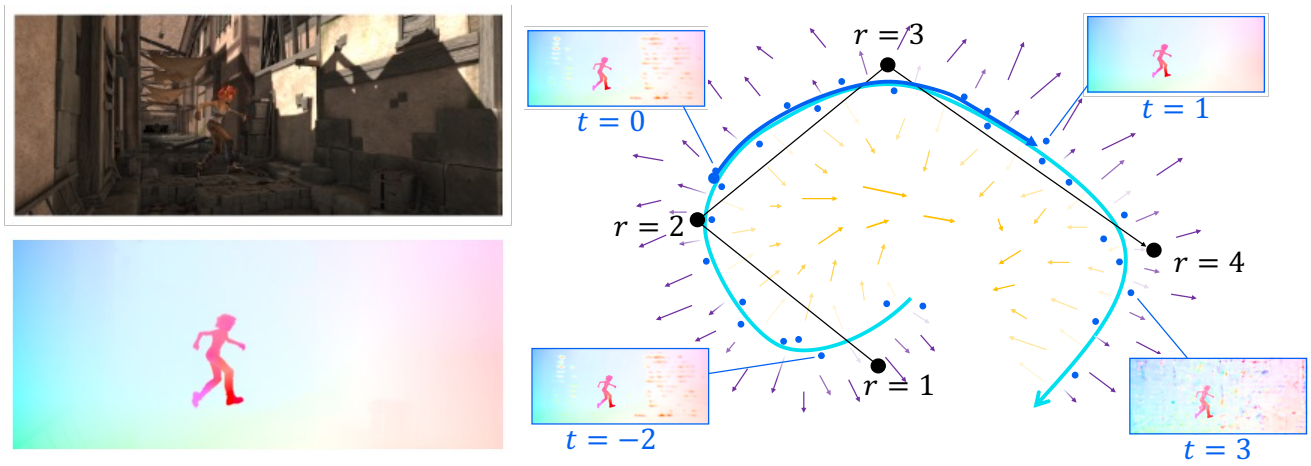


Figure 1. Upper left: source image. Lower left: corresponding GT flow. Right: ODEs can represent a vector field, thus the time scale parameter of the neural ODE solver generates points along the path (both in blue) along the GT curve (cyan). Compared to the fixed number of steps of an RNN (black lines and points), the neural ODE does not overshoot because its termination criterion is data-driven.

updates, we propose to predict the derivative of a latent flow, which is subsequently decoded to update the flow estimate. This is achieved by integrating a neural ordinary differential equation layer into our proposed architecture. This neural ODE layer represents a specific instance of an implicit layer tailored to effectively model temporal dynamics. As opposed to GRUs, our methods involves solving a differential equation on each step until convergence, with the flexibility to adjust the number of integration steps within the solver as needed.

In the process of solving neural ordinary differential equations (ODEs), the solver operates iteratively, but this process is fundamentally different from the standard iterative optical flow estimation. Typically, an iterative optical flow model refines its input, adding extra computational load. In contrast, the proposed method avoids recalculating the input and works solely with the initial approximation. Similarly, the backpropagation process differs: in baseline optical flow, the gradient is computed at each iteration that may potentially lead to BPTT, whereas a black box differential solver in Neural ODEs requires only constant memory and does not suffer from BPTT-related issues.

This allows for finer-grained learning of the optical flow field, resulting in significantly improved accuracy of the estimated optical flow, as demonstrated by our experimental results. With our proposed approach, there is no need to prescribe a fixed iteration number. Instead, we define a stopping criterion that allows the number of optimization steps to be adaptive based on the current data. Furthermore, any potential excessive memory consumption due to a large number of steps can be mitigated by backward-mode integration during backpropagation.

Additionally, we offer an interpretation of Neural ODEs as a broader extension of GRU-based updates. As detailed in [7], a GRU-ODE is introduced as a continuous-time variant of the GRU architecture, suggesting that Neural ODEs can represent the same dynamics learned by GRUs as a special case.[1] Thus, opting for a Neural ODE as refinement block presents a more versatile framework for optical flow estimation, capable of capturing more complex dynamics.

In this paper, we select GMFlow [43] as a baseline model for its reported high accuracy across various optical flow estimation benchmarks. We demonstrate significant performance enhancements by substituting its GRU-based update module by implicit updates derived from the Neural ODE layer.

The main contributions of this paper are summarized as follows:

- We introduce modeling of image-to-image temporal dynamics by ordinary differential equations.
- In the proposed framework, the number of internal steps is guided by the prediction accuracy. On each step of the solver, we are approximating a solution of the differential equation, thus the total number of refinement steps in the optical flow model itself can be fixed to one.
- The experimental results show better accuracy than baseline models that typically require multiple refinement stages.

[1] One can easily define such a model by choosing a solver with a fixed step update and maintaining the same number of refinement iterations as in the baseline GRU model.

2. Related Work

Optical Flow Estimation

Optical flow is based on a brightness or color constancy assumption whose first-order Taylor-expansion yields the famous optical flow. Early, classical approaches solve for the OF either globally or locally, which are later combined [3, 9, 12, 25]. More recently, neural approaches like FlowNet [8] propose task-specific correlation volumes, i.e., correlation maps between the two images on multiple resolutions, which have been used by many following works [15, 16, 30, 38, 39, 45].

To mimic the iterative optimization performed by classical algorithms, recurrent layers are proposed in RAFT [40]. This algorithm uses GRU cells to refine the estimate in the last stage [6, 23, 34]. This idea has since been used in many other algorithms [14, 20, 43, 46]. In GMFlow [43], the correlation volume is replaced by a differentiable matching on a single resolution. Although GMFlow originally uses a single step in the refinement module, the follow-up work of GMFlow+ [44] demonstrates improved performance when using multiple refinement steps. Works like GMFlow, PerceiverIO [19] and FlowFormer [14] are based on transformer architectures [41]. FlowFormer also uses multiple refinement steps. Recently, a solution based on diffusion has been proposed in [33].

Implicit Models

Models using implicit layers have been proposed for OF estimation [1]. Implicit layers locate some fixed point via root finding [13]. They have been applied to 3D surface fitting [28], flow estimation [36], and solving boundary value problems [35]. Due to the variable number of iterations, implicit methods can be more accurate than methods that rely on a pre-determined number of refinements.

Neural Ordinary Differential Equations (NODE)

This method also relies on an implicit layer [5]. Differently from other implicit methods, an ordinary differential equation (**ODE**) is modeled, thus they parameterize the derivative of some quantity. Solvers, for instance based on Runge-Kutta methods, are used to determine the solution of the ODE by evaluating its integral. In the paper [5], the right-hand side of the ODE in Eq. (3) is parameterized by a neural network. NODEs have all the advantages of other implicit methods while being more accurate and robust [11] when used to model time-varying differential processes [31]. To reduce errors during reverse mode integration, gradient checkpointing has been proposed by [48].

3. Method

Many recent optical flow models adopt roughly the same pipeline for motion prediction. In this section, we provide a brief overview of a general setup and later describe our approach in more detail.

3.1. Preliminaries

Optical flow methods predict the motion displacement on a pixel level. Specifically, an optical flow $\mathbf{f}$ maps every position $\mathbf{x} = (x, y, t)$ at time t to a position $\mathbf{x}' = \mathbf{x} + \mathbf{f}(\mathbf{x})$ at time $t + \tau$ by assuming brightness constancy

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x} + \Delta\mathbf{x}). \tag{1}$$

Recently proposed methods usually tackle this problem in the following manner. Given a pair of images $\mathbf{I_1}, \mathbf{I_2} \in \mathbb{R}^{H\times W\times 3}$ that are two consecutive video frames, the model first extracts low-dimensional representations $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{R}^{H/n\times W/n\times D}$, where D denotes the hidden feature dimension and n usually equals 8 (e.g., in RAFT [40], FlowFormer [14], GMFlow [43]). Sometimes an additional context network extracting the features only from the first image $\mathbf{I_1}$ is introduced [14, 40, 43]. Effectively, the context network is identical to the feature encoder network.

Since it is expected for correspondences in the consecutive frames to have a high similarity, the next step is usually a construction of a correlation volume. In practice, it is realized as a four-dimensional tensor $\mathcal{C} \in \mathbb{R}^{H\times W\times H\times W}$ with entries corresponding to the dot product between the corresponding feature vectors. By pooling across two last dimensions, we obtain a correlation pyramid $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ with $\mathcal{C}_i \in \mathbb{R}^{H\times W\times H/2^k\times W/2^k}$, where k usually runs from 0 to 3 [40].

One of the main stages in existing optical flow models is the refinement step. Integrating this additional stage into models helps to achieve a higher accuracy. Usually, the refinement is done by iteratively applying GRU-based updates. In this setting, we enforce a *discrete* nature of the refinement. Instead, we propose to generalize it to the *continuous* case by replacing GRU with a Neural ODE block. Moreover, GRU-based update may be rewritten in a Neural ODE setting making it a more general setting.

The first flow approximation is initialized as zero $\mathbf{f}_0 = \mathbf{0}$. Then, the current flow estimation, a correlation and a context embedding $\mathbf{q}$ are fed into the iterative update module. Typically, the update module is realized as a separable convolutional GRU network and its output $\Delta\mathbf{f}$ is fed again into the update module several times in order to mimic the optimization process: $\mathbf{f}_{k+1} = \mathbf{f}_k + \Delta\mathbf{f}$. In this paper, we suggest to go beyond beyond ConvGRUs as the primary building block for optical flow refinement. Instead, we introduce a Neural ODE-based update module.

3.2. Neural Ordinary Differential Equations

Neural ODEs reformulate recurrent updates in the continuous manner. Suppose we have an update equation given

by:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + g(\mathbf{h}_t, \theta_t), \tag{2}$$

where $t \in \{0, \dots, T\}$ is a time step, $\mathbf{h}_t \in \mathbb{R}^n$ is a hidden state, θ is for model parameters and $g(\cdot)$ is a dynamics generating function. The continuous version of this dynamics can be written as:

$$\frac{d\mathbf{h}(t)}{dt} = g(\mathbf{h}(t), t, \theta_t). \tag{3}$$

As proposed in [5], we parameterize the displacement field by a neural network f_θ with learnable parameters θ.

In practice, the right-hand side of the equation 3 is parametrized by a neural network. In essence, instead of explicitly parametrizing the underlying function itself, we parametrize the *change* of the function. During Neural ODE training, the output is calculated using black-box differential equation solvers. By choosing the adjoint sensitivity method, we can use memory-efficient $O(1)$ backpropagation for updating the model parameters θ [29].

3.3. Parametrizing the right-hand side of the Neural ODE

In optical flow models the flow estimation $\mathbf{f}$ is usually iteratively updated as $\mathbf{f}_{k+1} = \mathbf{f}_k + \Delta\mathbf{f}$. In this paper, we propose to model $\Delta\mathbf{f}$ as an output of the neural ODE block.

Transformer right-hand side

Since we do not limit ourselves to the specific type of the architecture for the right-hand side of the ODE, an alternative option is to use a transformer-based right-hand side. This allows to model more global dependencies due to increased receptive field. In our experiments, we considered a simple transformer block. Since the dynamics modeled by ODE has to have the same output dimension as the input, we apply an additional module that projects the concatenated flow and correlation. In our experiments, we use a simple convolutional network consisting either of only one convolutional layer or of two convolutions with ReLU nonlinearity in between. More details on the architecture can be found in the next sections.

Additionally, since we model the *gradient* of the flow update, the actual solution is of a higher order, thus having a higher representation power. This property allows us to keep the right-hand side of the ODE very simplistic: we use only one Transformer block with one head and hidden MLP dimension the same as the input dimension. Besides that, we have a mixing network projecting the contatenated input to a hidden dimension represented by only one convolutional layer. However, this basic setup allows to model the optical flow with a higher accuracy compared to convolutional GRUs.

3.4. Neural ODE solvers

Neural ODEs are being solved using black-box differential solvers. We have a flexibility when setting a solver: one can choose an adaptive step solver or a solver allowing for setting the step size and a maximal number of iterations allowed for a solver. Apart from that, one can choose between higher-order solvers (as some of the Runge-Kutta methods [22, 32]) or first or second order solvers (e.g., Euler and midpoint methods, respectively). In our experiments, we chose a simple midpoint method that is a refinement of the Euler method. We observed a reasonable tradeoff between the convergence speed and accuracy but one can always adjust the solver choice and tolerance parameters. In some experiments, we have used Fehlberg method, which is an adaptive fourth-order method from Runge-Kutta family. We did not notice a significant difference in accuracy while varying a solver, thus, in later experiments we just chose a faster option.

3.5. Training Loss

We adopt the same training losses as in the baseline model:

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-i} \|\mathbf{f}_{gt} - \mathbf{f}_i\|_1, \tag{4}$$

where $\mathbf{f}_{gt}$ is a ground truth flow, $\mathbf{f}_i$ is a predicted flow on the i-th timestamp, N is a number of predictions and $\gamma = 0.9$ is a weighting coefficient. In our experiments, we use only 1 refinement step.

3.6. Architecture details

For the Neural ODE right-hand side block we use a Transformer block with input and output dim $d_{inp} = d_{out} = 128$, the hidden dimension in the MLP layer equals $d_{hid} = 128$, number of heads equals to one. The mixing network is a mapping

$$\mathcal{M}\big([\mathbf{q}, \mathbf{f}, \mathcal{C}]\big) \colon \mathbb{R}^{d_{inp}} \times \mathbb{R}^{d_{inp}} \times \mathbb{R}^{d_{hid}} \to \mathbb{R}^{d_{hid}},$$

where $\mathbf{q}$ is a context embedding, $\mathbf{f}$ is a flow estimation and $\mathcal{C}$ is a correlation tensor. We use two versions: one convolutional layer or a small convolutional network consisting of two layers.

4. Experiments

In this section, we present a comprehensive evaluation of NODE-Flow on realistic or real-world data, including Sintel [4] and KITTI [10, 27].

Datasets

We follow commonly used training and evaluation protocols [14, 21, 40, 44, 46]. First, we train our model on FlyingChairs (Chairs) [8] and FlyingThings3D (Things) [26]

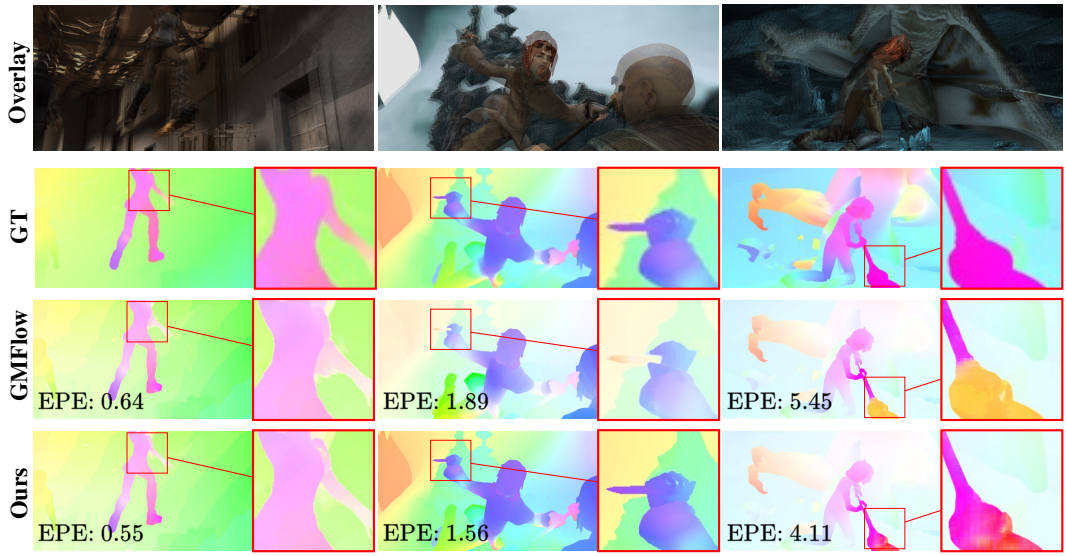


Figure 2. Qualitative results on Sintel (train) using GMFlow and NODE-GMFlow. The top row depicts the two frames overlay, the second row is a ground truth optical flow. We additionally zoom into regions highlighting the difference between these two models.

datasets. Using the model checkpoint after the first training stage, we evaluate the zero-shot generalization on Sintel (train) [4] and KITTI (train) [10, 27] datasets. After that, we finetune the model on the mix of the datasets consisting of Sintel, KITTI and HD1K datasets. Following the training protocol of GMFlow, we additionally finetune the C + T checkpoint on Virtual KITTI dataset.

Evaluation Metrics

We follow the widely used evaluation metrics for optical flow estimation. End-Point Error is an Euclidean distance between ground truth and flow prediction and Fl is the percentage of optical flow outlier.

Networks and Training

We use GMFlow [2] architecture as the baseline. The refinement module based on Convolutional GRU was replaced with the Neural ODE block. In our implementation, we used the codebase proposed in the original Neural ODE paper [5] [3]. The mixing network $\mathcal{M}$ consists of two convolutional layers with ReLU non-linearity mapping from $\mathbb{R}^{d_{inp}+d_{inp}+d_{hid}} \rightarrow \mathbb{R}^{d_{hid}+d_{hid}}$, with kernel size 5 and padding 3 and from $\mathbb{R}^{d_{hid}+d_{hid}}$ to $\mathbb{R}^{d_{hid}}$ with the same kernel size and padding. The tolerance parameter in the neural ODE block is set to $1e-3$.

Implementation details

Due to computational restrictions, we could not run the experiments with the hyperparameters from original GMFlow paper and had to scale the batch size and learning rates accordingly. We trained our model on FlyingChairs dataset for $50K$ iterations with a batch size of 32 and learning rate of $4e-4$. The input images were resized to $(384, 512)$. Later, we finetuned the model on Things dataset for $800K$ iterations with a batch size equal to 8 and learning rate of $1e-4$. Here, the input image size was set to be equal to $(384, 768)$. For later stages of training, we finetuned the model on Sintel dataset for $200K$ iterations with batch size of 4 and learning rate of $1e-4$. For finetuning on Virtual KITTI, we initialized the model with the Things checkpoint and trained on Virtual KITTI dataset for $80K$ iterations with batch size of 8 and learning rate of $5e-5$. Finally, finetuning on KITTI was done by training the model initialized with the Virtual KITTI phase checkpoint for $30K$ steps with batch size of 8 and learning rate of $1e-4$. All models were trained using AdamW optimizer [24] and with OneCycleLR scheduler [37].

4.1. Evaluations on Sintel (train) and KITTI (train)

Table 1 shows the performance of previous state-of-the-art optical flow models and ours on Sintel (train) and KITTI (train). Our models trained on FlyingChairs and FlyingThings3D (C+T) shows significant improvements compared to GMFlow and comparable performance with state-of-the-art architectures on Sintel dataset. On KITTI dataset, although GMFlow and Ours show slightly worse performance than SOTA methods, our NODE-GMFlow models improve the performance over the baseline GMFlow model. Same as GMFlow, we additionally train Virtual KITTI 2 dataset, and our NODE-GMFlow shows the state-of-the-art performance

[2] https://github.com/autonomousvision/unimatch
[3] https://github.com/rtqichen/torchdiffeq

Table 1. Optical flow estimation results on Sintel (train), and KITTI (train) datasets. We train the models on FlyingChairs (C) and FlyingThings (T). Similarly to the training protocol of GMFlow, we additionally train our model on Virtual KITTI dataset. Number in brackets (columns of KITTI in lower three rows of table) indicate results of models additionally trained on VK. We provide two options of our models: version (I) was trained for 400K iterations on FlyingThings, while (II) was trained for 800K iterations.

Method	Training Dataset	#Parameters	Sintel (train)		KITTI (train)	
			Clean	Final	EPE	Fl-all
LiteFlowNet [15]	C+T	5.4M	2.48	4.04	10.39	28.5
FlowNet2 [18]		162.5M	2.02	3.54	10.08	30.0
PWC-Net [38]		8.8M	2.55	3.93	10.35	33.7
RAFT [40]		5.3M	1.43	2.71	5.04	17.4
GMA [21]		5.9M	1.30	2.74	4.69	17.1
FlowFormer [14]		18.2M	**1.01**	2.40	4.09	14.7
GMFlow [47]	C+T / (+VK)	4.7M	1.08	2.48	7.77 / (2.85)	23.4 / (10.8)
NODE-GMFlow I (Ours)		9.2M	1.04	**2.39**	7.21 / (2.44)	22.2 / (8.8)
NODE-GMFlow II (Ours)		9.2M	**1.01**	2.59	6.53 / (**2.38**)	20.5 / (**8.4**)

Table 2. Optical flow estimation results on Sintel (test) and KITTI (test) datasets. For evaluation on test dataset, we used NODE-GMFlow II version. We observe a significant improvement over the baseline model GMFlow.

Method	Sintel (test)		KITTI (test)
	Clean	Final	(Fl-all)
LiteFlowNet2 [17]	3.48	4.69	7.74
PWC-Net+ [39]	3.45	4.60	7.72
RAFT [40]	1.94	3.18	5.10
GMA [21]	1.40	2.88	5.15
FlowFormer [14]	1.16	2.09	4.68
GMFlow+ [44]	1.03	2.37	4.49
GMFlow [47]	1.74	2.90	9.32
NODE-GMFlow (Ours)	1.26	2.55	5.41

on KITTI dataset. While, in general, version (I) has better performance compared to GMFlow, training for more iterations seems to be more beneficial except for less accurate predictions on Sintel (final). More discussion on that can be found in Ablation section 4.4.

4.2. Evaluations on Sintel (test) and KITTI (test).

Table 2 summarizes the performance of the existing methods and the accuracy of the proposed approach. Our model shows significant improvement compared to the baseline GMFlow model on both Sintel and KITTI datasets.

4.3. Qualitative Results

Figure 2 shows qualitative results on the Sintel (train, clean) using the original GMFlow and NODE-GMFlow which are trained on FlyingChairs and FlyingThings3D. Our NODE-GMFlow shows more accurate and robust flow estimation compared to GMFlow. Figure 3 shows qualitative results on the KITTI (test) using GMFlow methods and NODE-GMFlow. Although our NODE-GMFlow fl-all score is slightly worse than GMFlow+, our NODE-GMFlow shows high quality prediction (as highlighted by red circle).

4.4. Ablation Study

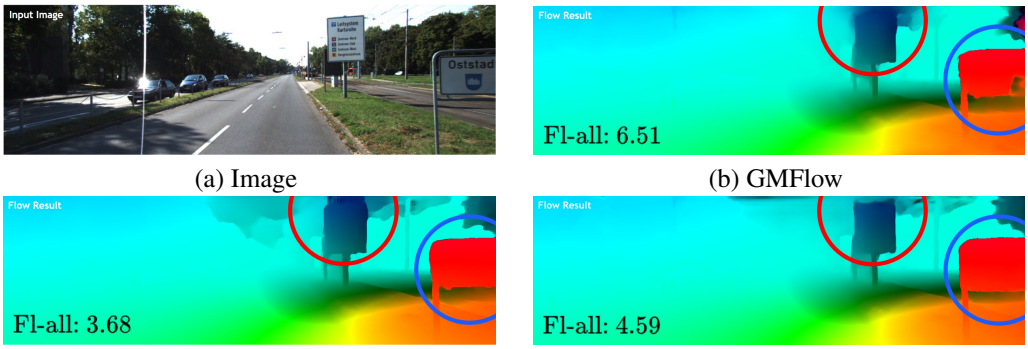
In this part, we discuss the experimental results after varying parameters in the proposed model. We check the importance of the solver choice, number of training iterations and the structure of the mixing network.

Varying time-scale t

The model was trained using boundary values of $t = 0$ and $t = 1$. In Figure 4 we demonstrate the effect of decoding the latent flow with time scales $t < 1$. As can be seen, the quality improves significantly between $t = 0$ and $t = 0.5$, yet only marginally afterwards.

Mixing network

Before refining the predicted flow using the Neural ODE module, we project the input into the lower-dimensional space. Usually, this is done within the Convolutional GRU module mapping from $d_{inp} + d_{inp} + d_{hid}$ to d_{hid}. Instead, we first project the input to $\mathbb{R}^{d_{inp}}$ and then run Neural ODE module in the latent space. For the mixing network $\mathcal{M}$, we used two different networks: the first consisting of a single convolution and the second consisting of two convolutional layers with ReLU in between. The experimental results are summarized in Table 5. We observe a better performance with the mixing network consisting of two convolutions. For



(a) Image (b) GMFlow (c) GMFlow+ (d) NODE-GMFlow

Figure 3. Qualitative results on KITTI (test) using GMFlow, GMFlow+, and NODE-GMFlow. Our model shows a significant improvement over the baseline GMFlow model with only one refinement step, while GMFlow+ uses 6 refinement steps on every iteration.

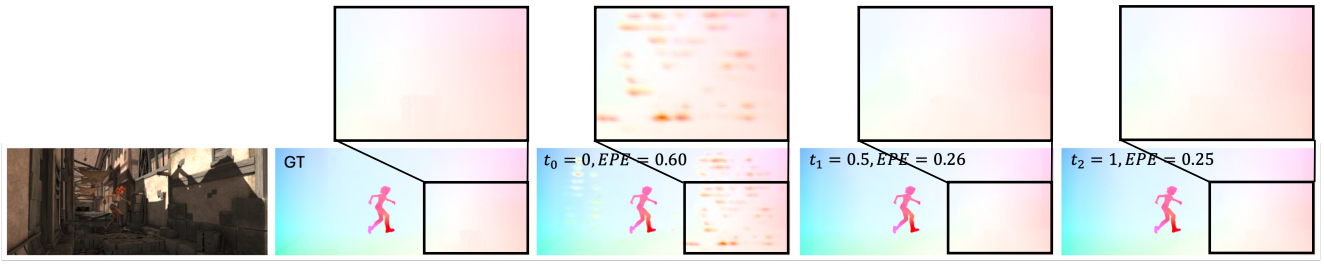


Figure 4. Instead of approximating the optical flow using iterative GRUs, we propose to use a Neural ODE that implicitly solves a differential equation. Neural ODEs model the flow in a continuous manner: the implicit integration is done from 0 to 1. In this figure, we visualize the convergence of the flow obtained from the intermediate steps during integration inside the Neural ODE block. As can be seen, the quality of the flow initially improves significantly.

our experiments, we choose this option.

Table 3. Number of iterations. Generally, an increased number of iterations leads to higher accuracy.

Model	Iters	Things	Sintel (train)		KITTI (train)	
		val	clean	final	EPE	Fl-all
NODE-GMFlow (I)	400K	2.63	1.04	**2.39**	7.21	22.16
NODE-GMFlow (II)	800K	**2.47**	**1.01**	2.59	**6.52**	**20.46**

Number of iterations

We trained two models with different number of iterations, the experimental results are summarized in Table 3. In general, the experiments suggest that training the model for more iterations benefits the prediction accuracy and generalization on unseen datasets. The original GMFlow model is trained for 800K iterations, however we can see the improvement over the baseline model even after training on 400K iterations; the comparison with the baseline can be found in the Table 1. For qualitative examples and final evaluations on KITTI dataset, we use the model trained for 800K iterations.

Table 4. Solver choice. All models were trained for 200K iterations on the Things dataset.

Solver	Things	Sintel (train)		KITTI (train)	
	val	clean	final	EPE	Fl-all
Fehlberg	2.88	**1.14**	**2.76**	**7.20**	23.17
Midpoint	**2.86**	1.15	**2.76**	7.66	**22.33**

Solver

We also experiment with different solvers; our findings are summarized in Table 4. We can observe an almost identical accuracy on Things and Sintel, while on KITTI dataset the model trained with Fehlberg solver behaves slightly better. In later experiments, we resort to Midpoint solver due to better speed-accuracy tradeoff.

Table 5. Mixing network. Models were trained for 800K iterations. Choosing a deeper mixing network leads to a significantly increased accuracy.

Mixing network	Things	Sintel (train)		KITTI (train)	
	val	clean	final	EPE	Fl-all
One conv layer	2.55	1.06	**2.59**	6.71	21.50
Two conv layers	**2.47**	**1.01**	**2.59**	**6.52**	**20.46**

5. Limitations

The limitations of the proposed method are the following. First, we observe a worse accuracy on Sintel final dataset thus our method is not improving the baseline model on all the benchmarks. Second, Neural ODEs are known to be prone to instabilities during the training [48]. While we did not observe it during our experiments, there is the possibility of such problems if trained on other data. Finally, the proposed approach does not take into account novel advances in Neural ODE field, for instance [48].

We believe that our work does not lead to negative societal impacts. The proposed approach is a foundational research and most of the training data is synthetic with the only real-life dataset used during the training being the KITTI dataset [10, 27]. On the contrary, we anticipate that our work impacts fields like autonomous driving which relies on OF estimation at the beginning of the information processing pipeline. As such, we hope that even small improvements in the accuracy of OF estimates may lead to fewer accidents involving self-driving vehicles.

6. Conclusion

In this paper, we proposed a novel approach that leverages Neural Ordinary Differential Equations to learn the optical flow field, based on given correlation and context features. Prior works use recurrent neural modules for refinement which are motivated from solvers in classical optimization but which require a fixed and pre-determined number of steps. The proposed method is more general than existing methods since it can generate the exact same updates if restrictive choices are made for its architecture, solver and hyperparameters. In other words, the proposed model can learn more general dynamics than works using recurrent neural modules. Extensive experiment results demonstrate the efficacy of our solution.

References

[1] Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 620–630, 2022. 3

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 1

[3] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision (IJCV)*, 61:211–231, 2005. 3

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 611–625. Springer, 2012. 4, 5

[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3, 4, 5

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 3

[7] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2758–2766, 2015. 3, 4

[9] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*, pages 363–370. Springer, 2003. 3

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 4, 5, 8

[11] YAN Hanshu, DU Jiawei, TAN Vincent, and FENG Jiashi. On robustness of neural ordinary differential equations. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[12] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, pages 319–331. International Society for Optics and Photonics, 1981. 3

[13] Zhichun Huang, Shaojie Bai, and J Zico Kolter. $(\text{Implicit})^2$: Implicit layers for implicit representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9639–9650, 2021. 3

[14] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *Proceedings of the European Conference on Computer Vision*, 2022. 1, 3, 4, 6

[15] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8981–8989, 2018. 3, 6

[16] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn-revisiting data fidelity and regularization. *arXiv preprint arXiv:1903.07414*, 2019. 3

[17] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn—revisiting data fidelity and regularization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2555–2569, 2020. 6

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 6

[19] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[20] Jisoo Jeong, Jamie Menjay Lin, Fatih Porikli, and Nojun Kwak. Imposing consistency for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3181–3191, 2022. 3

[21] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9772–9781, 2021. 4, 6

[22] Wilhelm Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. Teubner, 1901. 4

[23] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3367–3375, 2015. 3

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[25] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981. 3

[26] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 4

[27] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015. 4, 5, 8

[28] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *International Conference on Computer Vision (ICCV)*, pages 4743–4752, 2019. 3

[29] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018. 4

[30] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4161–4170, 2017. 3

[31] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019. 3

[32] Carl Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895. 4

[33] Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[34] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015. 3

[35] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:7462–7473, 2020. 3

[36] Kevin D Smith, Francesco Seccamonte, Ananthram Swami, and Francesco Bullo. Physics-informed implicit representations of equilibrium network flows. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:7211–7221, 2022. 3

[37] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 5

[38] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 3, 6

[39] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019. 3, 6

[40] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*, pages 402–419. Springer, 2020. 1, 3, 4, 6

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[42] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 1

[43] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8121–8130, 2022. 2, 3

[44] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 3, 4, 6

[45] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Advances in neural information processing systems*, pages 794–805, 2019. 3

[46] Feihu Zhang, Oliver J Woodford, Victor Adrian Prisacariu, and Philip HS Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10807–10817, 2021. 3, 4

[47] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17592–17601, 2022. 6

[48] Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International Conference on Machine Learning*, pages 11639–11649. PMLR, 2020. 3, 8