Master Thesis in
Computer Science
Artificial Intelligence Technologies

The Influence of Out-Of-Frame Moving Objects on Optical Flow Accuracy

Thomas Sebastiaan Streefkerk

October 2024

To obtain the degree of Master of Science at the Delft University of Technology

Thomas Sebastiaan Streefkerk:
The Influence of Out-Of-Frame Moving Objects on Optical Flow Accuracy

Supervisors:	Dr. J.C. van Gemert
	M.Sc. A.S. Gielisse
Graduation Committee:	Dr. J.C. van Gemert
	Dr. P. Kellnhofer

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. J.C. van Gemert and M.Sc. A.S. Gielisse, for their invaluable support, guidance, and expertise throughout this project. Their advice and knowledge have been instrumental in shaping my research and helping me overcome numerous challenges. I would also like to thank Dr. P. Kellnhofer, for agreeing to be my third committee member, and for his valuable feedback, which has further strengthened this work.

Finally, I am deeply grateful to my parents for their unwavering support and encouragement, which has been essential to me throughout my studies.

Thomas Streefkerk,
October 2024, Delft.

Contents

1 Introduction

Optical flow is the process of estimating the motion of objects in a visual scene between consecutive frames. It has been a fundamental problem in computer vision since it was first introduced by Horn and Schunck in the early 1980s. Initially, optical flow was calculated through classical methods, focusing on energy-based approaches and variational frameworks to estimate pixel movements. However, as computational power and data availability increased, deep learning techniques revolutionized optical flow estimation, enabling more accurate and efficient predictions by leveraging neural networks to model complex motion patterns.

Despite its progress, optical flow estimation remains challenging, especially in scenarios involving ambiguous data. Ambiguities can arise from various factors, such as occlusions, lighting changes, or out-of-frame movements—where objects move out of the visible scene between frames. Out-of-frame movements are particularly problematic because, in many cases, the model simply cannot predict these motions accurately, as it lacks any visual information about the object once it leaves the frame. This lack of information leads to spikes in the loss function during training, as these out-of-frame regions are handled the same way as regions that remain in-frame. These spikes introduce instability, often making the training process less stable and leading to overall reduced model accuracy.

While most optical flow models overlook these challenges, some recent approaches have started to acknowledge the impact of ambiguous data and introduce new architectures and loss functions aimed at addressing these issues. However, many of these methods stop short of directly verifying the influence of ambiguous cases, such as out-of-frame movements, leaving open questions about how these ambiguities affect overall model accuracy.

In the scientific article, we empirically verify the impact of out-of-frame movements on optical flow estimation using a custom toy dataset, specifically designed to distinguish between in-frame and out-of-frame object movements. Through this controlled setting, we develop a method to mitigate the influence of out-of-frame movement on optical flow estimation, demonstrating improved accuracy over existing methods on our custom dataset. Furthermore, we extend this method to existing synthetic datasets, where it continues to yield promising results, highlighting its potential for enhancing robustness in optical flow models.

Section 2 presents the scientific article, aimed at an audience of experts in computer vision and related theories. For the more general reader, supplementary material is included in Section 3 to provide additional context on the underlying theories. Finally, Appendix A provides additional insights into topics we initially explored but ultimately set aside, either due to reaching a dead-end or determining that the challenges involved were not feasible to address within the limited time available. Specifically, Appendix A.1 discusses optical flow with diffusion models, while Appendix A.2 examines optical flow as a class imbalance problem, a topic to which we dedicated considerable time and effort.

2 Scientific Article

The scientific article starts on the next page.

The Influence of Out-of-Frame Moving Objects on Optical Flow Accuracy

T.S. Streefkerk
Computer Vision Lab, Delft University of Technology, The Netherlands

Abstract

Training data ambiguity, such as the presence of out-of-frame moving objects, introduces significant challenges in deep learning-based optical flow models by causing large loss spikes and training instability. Most models overlook this ambiguity, treating it as a limitation of existing datasets. SEA-RAFT [1] attempts to address these ambiguous areas with an additional network and a modified loss function, yet does so without explicitly verifying the specific drawbacks. In this paper, we investigate the influence of out-of-frame movements on model accuracy by generating the FlyingIcons dataset, which includes in-frame and out-of-frame masks for precise analysis. Using the latter masks, we introduce a weighted masked training scheme that selectively penalizes errors in out-of-frame areas, significantly increasing model accuracy over both standard GMA training and SEA-RAFT. Building on this concept, we propose a weighted partially masked training method, which uses partial out-of-frame masks generated through a simple process that adds the ground truth flow to pixel locations and checks if they fall outside the frame. While this method only yields improvements in error reduction on FlyingThings3D, our findings suggest that incorporating similar masks into other synthetic datasets could improve model stability and accuracy with minimal additional overhead. This highlights a promising direction for further research, particularly in developing more complex mask generation strategies and creating synthetic datasets with out-of-frame masks to enhance generalizability across datasets.

1 Introduction

Optical flow is the task of predicting a per-pixel vector representing the apparent motion between two frames. It is a fundamental problem in computer vision that suffers from numerous challenges, including occlusion, out-of-frame movements, many-to-one motions, one-to-many motions, large motions and (motion) blur [2].

In this paper, we investigate the influence of out-of-frame moving objects on the overall accuracy of deep learning optical flow models, and we bring to light a downside of existing synthetic datasets which are commonly used to pretrain such models. We perform this investigation by training GMA [3], a deep learning optical flow model designed to track complex movements across frames, on a custom generated synthetic dataset we call FlyingIcons. Besides the typical first frame, second frame and ground truth flow, we generate two additional mask matrices. The first mask, called the out-of-frame mask, identifies objects that move out of the frame, while the second mask, called the in-frame mask, identifies objects that stay within the frame. These masks are used as a tool to analyse the accuracy of the model on different areas of the data.

Additionally, the out-of-frame mask is used in experiments during training to down-weight the relative contribution to the loss of out-of-frame moving objects, as these areas carry high uncertainty that disrupts the stability of training. By reducing their influence, we achieve better accuracy on FlyingIcons than standard training allows. This demonstrates that the absence of out-of-frame masks in current synthetic datasets results in reduced accuracy. Weighted masked training was selected over other methods for its simplicity and efficiency, as it directly adjusts the loss contributions without modifying the model architecture or requiring additional parameters [4]. We also compare this method with SEA-RAFT [1], which tackles the challenge of ambiguous cases (such as out-of-frame movements) reducing model accuracy by introducing an extra network and a new loss function designed to model uncertainty and down-weigh these areas dynamically. Our results show that weighted masked training achieves better accuracy than SEA-RAFT on FlyingIcons, highlighting its effectiveness as a lightweight, adaptable approach to improving model robustness.

Finally, we extend our findings to the widely-used FlyingThings3D dataset. In our custom FlyingIcons dataset, we had precise knowledge of out-of-frame masks since they were generated alongside the data. However, for existing datasets such as FlyingThings3D, obtaining these masks is challenging, especially when aiming to capture all types of ambiguity, including occlusions and complex motions such as many-to-one and one-to-many object movements. Ideally, these complete ambiguity masks would be created during the generation of synthetic datasets, as this is when object-level infor-

mation is readily available. This approach allows for accurate identification of ambiguous cases, such as distinguishing between partial and fully out-of-frame objects. In an ideal scenario, only fully out-of-frame objects would be masked out. However, generating these complete ambiguity masks accurately for existing datasets is difficult and would essentially require optical flow predictions themselves. Instead, we create partial out-of-frame masks by assigning pixels to the mask when the sum of their position and flow falls outside the frame. This simpler approach, which adjusts the loss function based on partial masks, yields modest accuracy improvements on FlyingThings3D. While the improvements are dataset-specific, this method highlights a promising direction for future research into enhancing model accuracy through targeted weighting.

Based on these findings, we highlight the following key contributions of our work:

- We quantitatively demonstrate the impact of out-of-frame objects on optical flow model accuracy, revealing a trade-off between in-frame and out-of-frame accuracy that depends on their relative masking weights.
- By reweighing loss contributions with out-of-frame masks, we enhance GMA's accuracy on FlyingIcons, surpassing standard training and SEA-RAFT [1].
- We propose a weighted partially masked training method using naively generated partial out-of-frame masks, achieving accuracy improvements specifically on FlyingThings3D, with potential for refinement on more complex datasets.

2 Related Work

Traditionally, optical flow is framed as an optimization problem with a trade-off between a data term, which encourages alignment of visually similar areas, and a regularization term encouraging smoothness in the predicted flow. This approach has provided a foundation for early optical flow methods [5–7], but these classical techniques often struggled in complex real-world scenarios.

2.1 Training Data Ambiguity

Recently, with the rapid advancements in deep learning and the availability of synthetic datasets, optical flow models have achieved substantial improvements. Many state-of-the-art models [3, 8–18] rely on pretraining on FlyingChairs [9], followed by FlyingThings3D [19], to mitigate the scarcity of real-world optical flow data. However, these models are affected by ambiguities present in synthetic datasets, such as out-of-frame movements and occlusions. These ambiguities contribute to spikes in the training loss and increase instability, as they represent some of the most challenging conditions for models [5]. In certain cases, these situations even make it impossible for the model to predict the correct flow. Our findings demonstrate the impact of these effects, and we propose a preliminary solution to address this instability.

2.2 Key Optical Flow Models

RAFT [8] steps away from the traditional coarse-to-fine approach by computing and refining a single flow field iteratively at full resolution from the beginning. It starts with a zero-initialized flow field and refines it through multiple iterations using a recurrent gated update operator. A key innovation is its dense all-pairs correlation volume, which captures pixel-wise similarities across frames. This volume enables RAFT to efficiently model fine details and large displacements by pooling information from multiple correlation resolutions. As a result, RAFT avoids the limitations of the coarse-to-fine approach, which can struggle with recovering from errors in early, low-resolution stages, handling small fast-moving objects, and requires extensive iterative training

GMA [3] builds on RAFT by introducing a global motion aggregation (GMA) module. This attention-based module, inspired by the success of transformers [20], aggregates global motion information, improving the model's accuracy on occluded regions and capturing long-range dependencies. By enhancing RAFT's ability to model global interactions, GMA addresses some of the challenges posed by occlusions and long-range motion. We adopt GMA as a baseline in most of our experiments due to its widely recognized success and its strong reputation in the field.

SEA-RAFT [1] builds on the RAFT framework by addressing key challenges in optical flow estimation, particularly in handling ambiguous areas such as occlusions and regions with unpredictable motion. These ambiguous areas often lead to greater uncertainty or errors in the flow predictions, as the correct motion cannot be easily inferred. SEA-RAFT assumes, though never empirically verifies, that these ambiguous areas negatively affect accuracy. To address this, SEA-RAFT introduces an additional network to predict which pixels should be estimated by the model with high confidence. This pixel-wise uncertainty estimation allows the model to focus its predictions on regions where it can be more accurate and reliable. To complement this, SEA-RAFT utilizes a Mixture of Laplace loss, which is designed to handle the uncertainty in regions where the flow is difficult to predict. We investigate the assumption that ambiguous areas negatively impact accuracy, and we compare GMA's accuracy when trained with our weighted training method with SEA-RAFT's accuracy to evaluate the efficacy of their added network and adapted loss.

2.3 Synthetic Datasets

Synthetic datasets play a crucial role in optical flow research, as obtaining large-scale annotated data in real-world settings is challenging. Some datasets, such as FlyingChairs [9] and FlyingThings3D [19], are generated through simulations, while others, such as Sintel [21], Middlebury [22] and Spring [23], utilize rendered animations or stereo camera setups, both of which allow for precise ground truth calculation. This is essential for training deep learning models, as synthetic data can cover a wide range of challenging scenarios such as occlusions and large displacements. However, despite their advantages, synthetic datasets face a domain gap problem, where the textures, lighting, and object appearances differ from real-world conditions [24]. As a result, models pre-trained on synthetic data often require fine-tuning on real-world datasets to improve their generalization and accuracy in practical applications. We use FlyingThings3D, Sin-

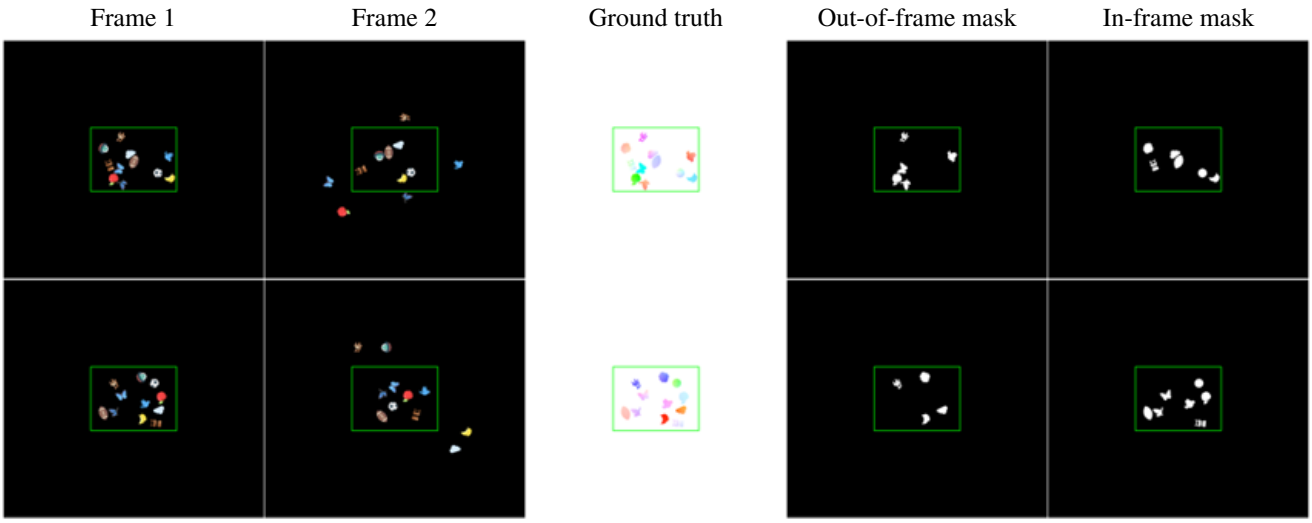


Figure 1: Two samples from the FlyingIcons dataset are shown. From left to right, the columns represent the first frame, the second frame, the ground truth flow, the out-of-frame mask, and the in-frame mask. The green box highlights the specific area the model trains on, while the remaining area provides a visualization of out-of-frame movements. These examples provide a visual overview of the dataset and demonstrate the tools available for both model analysis and weighted training on out-of-frame regions.

tel, and KITTI-15 [25], a real-world dataset captured with stereo cameras and 3D laser scanners in driving scenes, for evaluation. Additionally, we draw inspiration from existing simulated synthetic datasets to generate our own, incorporating out-of-frame and in-frame mask matrices for use in model training and accuracy analysis. Ideally, such masks would have been included in synthetic datasets to further enhance their utility.

3 Method

To investigate the influence of out-of-frame moving icons in the training data on model accuracy, we generated our own synthetic dataset called FlyingIcons and used it to train using weighted masked training. The following sections describe the dataset generation and the method of weighted (partially) masked training.

3.1 FlyingIcons dataset generation

Figure 1 shows three samples from the FlyingIcons dataset. The green box indicates the region on which the model is trained, while the remaining area visualizes out-of-frame movements.

Frame 1 contains eleven in-frame icons [26] that are positioned and rotated randomly. In Frame 2, these icons are again randomly rotated and then translated according to a Normal distribution with a standard deviation σ. This translation approach is similar to the Normal distribution used in synthetic datasets such as FlyingChairs [9] and FlyingThings3D [19], and allows the model to learn meaningful patterns from out-of-frame movements. For example, an icon starting near the middle-right edge of the frame that moves out-of-frame will likely have moved beyond the right frame border, allowing the model's prediction to capture this behaviour. In contrast, using purely random translations in Frame 2 would not enable the model to learn these out-of-frame patterns effectively.

The ground truth flow is generated using affine transformations [27], and the two mask matrices are created by checking whether each icon stays in-frame or moves out-of-frame. These masks are relative to Frame 1, as we only consider forward-looking cases, with Frame 1 always preceding Frame 2 during training and testing. Additionally, we ensure that icons do not overlap with each other or the frame's border, as such overlaps would introduce further ambiguities that could affect the accuracy of our experiments. Finally, the icons were selected to provide a diverse range of shapes and colours, while still sharing some overlapping features to ensure varied yet comparable data points.

3.2 Weighted masked training

Training the model using weighted masked training consists of applying a custom L1 loss. This loss is described by equation 1:

$$\mathcal{L}_{L1}^{\text{masked}} = \frac{1}{\sum_{i=1}^{N} \mathbf{M}_i^w} \sum_{i=1}^{N} \mathbf{M}_i^w \circ \left| \mathbf{F}_i - \hat{\mathbf{F}}_i \right|. \tag{1}$$

Here, $\mathbf{M}^w$ represents the out-of-frame object mask matrix with w indicating the mask weight for out-of-frame moving pixels. $\mathbf{F}$ is the ground truth flow, $\hat{\mathbf{F}}$ is the predicted flow and N is the amount of pixels in the sample.

The mask values $\mathbf{M}^w$ control pixel-wise contributions to the loss, where a value of zero ignores the pixel, and one counts it fully. In-frame pixels are assigned a value of one,

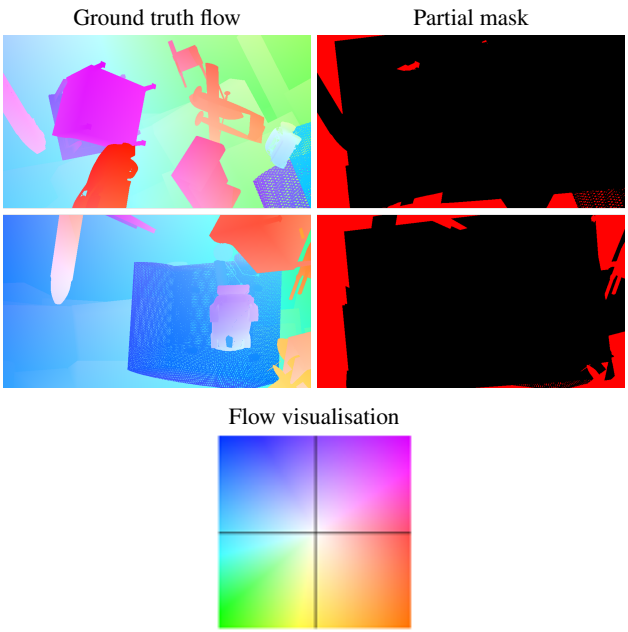


Figure 2: Two partial masks for the FlyingThings3D dataset are shown. On the left, we have the ground truth flows, and on the right, the corresponding partial masks. Red indicates out-of-frame movement, while black indicates in-frame movement. Additionally, in the bottom middle, a flow visualization is provided, where each colour corresponds to a specific direction of motion relative to the centre pixel, with higher colour intensity representing larger motion magnitudes. This figure illustrates both the ground truth flow and the partial masks, with the flow visualization included to help clarify how the partial masks are constructed. The partial masks highlight the in-frame and out-of-frame regions used in analysis and training.

while out-of-frame pixels are weighted by $w \in [0, 1]$, allowing control over their relative loss contribution. The division by $\sum_{i=1}^{N} \mathbf{M}_i^w$ normalizes by the total active weight, ensuring proper scaling even when parts of the image are excluded or weighted less in the mask.

3.3 Weighted partially masked training

To extend the weighted masked training method from Section 3.2 beyond our custom FlyingIcons dataset to widely used synthetic datasets, such as FlyingThings3D, we introduce a method called weighted partially masked training. In this approach, we first generate partial masks by adding the ground truth flow to pixel locations and determining whether they move out-of-frame. Pixels that move out-of-frame are included in the partial mask. Figure 2 shows two examples of these partial masks, where red represents out-of-frame pixels and black represents in-frame pixels. It also includes the ground truth flow and a flow visualization to help illustrate how the partial masks were created.

Although this method is computationally simple, it lacks object-level knowledge. As a result, it often includes only part of an object in the mask, leaving the rest in-frame. Given that it's not intended to fully represent entire objects moving out-of-frame, this approach provides an efficient approximation.

With the partial masks generated, we can then apply the weighted masked training strategy outlined in Section 3.2. Here, the mask matrix $\mathbf{M}^w$ in Equation 1 is replaced by the partial mask with some value for w depending on the experiment, allowing the training process to adjust the penalty for errors in out-of-frame regions according to the partial mask weight.

4 Experiments

4.1 Experimental setup

For each experiment, we adhere to the training setup outlined in the GMA paper [3]. Specifically, when training GMA on FlyingIcons, we replicate the training configuration used for FlyingChairs as described in the paper. To ensure a fair comparison, we also follow the same training procedure for FlyingChairs when training SEA-RAFT on FlyingIcons.

Similarly, for experiments on FlyingThings3D, we initialize GMA with the FlyingChairs checkpoint, as both described and provided by the authors [3].

During the generation of the FlyingIcons dataset, we use a standard deviation (σ) of 92, equivalent to a quarter of the frame height. This value was chosen after experimentation, as it resulted in an approximate average of four out-of-frame moving icons per sample, which seemed to be a reasonable quantity based on preliminary observations.

4.2 Mask weight variation

In this experiment, we aim to determine how different weights on out-of-frame masks influence the average endpoint error (AEPE) on both in-frame and out-of-frame regions. We do this by applying the weighted masked training method described in Section 3.2, varying the mask weight w from Equation 1 when training GMA on FlyingIcons. We chose mask weights $w \in [0, 1]$ to capture a wide range of data points while avoiding instability from overemphasizing out-of-frame regions. The results are presented in Table 1.

As expected, the lowest AEPE for GMA on in-frame icons occurs at a mask weight of 0.00. This is because, during training, GMA only focused on in-frame icons and the background, with no out-of-frame icons present to introduce instability or degrade accuracy on in-frame regions. Conversely, the highest AEPE is observed on out-of-frame icons, which is also anticipated since GMA did not train on these regions and thus has not learned how to handle them effectively. Training on them could have been beneficial, as the positions of the icons are determined according to a Normal distribution (see Section 3.1).

Interestingly, we might have expected the lowest AEPE for out-of-frame icons at a mask weight of 1.00, as this case places the greatest emphasis on out-of-frame regions during training. However, large standard deviations at mask weights of 0.60, 0.80, and 1.00 suggest that training was unstable in these cases, negatively affecting accuracy. As a result, the lowest AEPE for out-of-frame icons is achieved at a mask

Model	Mask weight	Overall	In-frame icons	Out-of-frame icons
GMA	0.00	8.00 (0.00)	**1.88 (0.02)**	213.06 (0.07)
	0.01	6.19 (0.00)	1.91 (0.02)	163.23 (0.23)
	0.02	6.07 (0.02)	1.95 (0.01)	159.50 (0.56)
	0.05	5.93 (0.01)	1.97 (0.04)	154.34 (0.21)
	0.10	5.87 (0.01)	2.00 (0.02)	152.56 (0.17)
	0.15	**5.83 (0.01)**	2.05 (0.01)	151.32 (0.24)
	0.20	5.84 (0.00)	2.19 (0.02)	151.08 (0.21)
	0.40	5.84 (0.02)	2.42 (0.07)	**150.15 (0.17)**
	0.60	6.89 (0.60)	12.39 (6.35)	155.88 (3.46)
	0.80	7.43 (0.99)	17.75 (9.54)	158.48 (5.37)
	1.00	6.85 (1.02)	12.02 (9.88)	155.13 (5.72)
SEA-RAFT	1.00	6.71 (0.51)	12.12 (4.91)	153.51 (2.15)

Table 1: Validation average end-point error (AEPE) values with standard deviations (in brackets) on different image regions for GMA and SEA-RAFT trained on FlyingIcons with the weighted masked training scheme (Section 3.2) and various mask weights. The mask weight represents the training weight on out-of-frame moving pixels, while in-frame pixels are consistently weighted at 1.0. Statistics are averaged over three runs, except for GMA with mask weights of 0.80 and 1.00, which are based on four runs. Bold values highlight the lowest AEPEs per column. The table illustrates a trade-off between accuracy on in-frame and out-of-frame icons, with an optimal mask weight at 0.15, yielding the lowest AEPE of 5.83, outperforming both standard GMA training and SEA-RAFT. Additionally, large standard deviations at mask weights 0.60, 0.80, and 1.00 suggest that training was unstable at these values, negatively impacting accuracy.

weight of 0.40, which is the largest stable mask weight. Similarly, for the runs considered stable (those with a mask weight of 0.40 and lower), the highest AEPE on in-frame icons also occurs at a mask weight of 0.40, which is expected because in-frame icons receive a lower relative weight when the mask weight increases. Additionally, training on out-of-frame regions introduces larger errors, potentially contributing to instability and the higher AEPE on in-frame icons at higher mask weights.

Overall, between the stable mask weight endpoints of 0.00 and 0.40, we observe a trade-off between in-frame and out-of-frame accuracy as the mask weight varies. This trade-off yields a sweet spot for GMA's overall accuracy at a mask weight of 0.15.

When comparing GMA and SEA-RAFT, we observe that SEA-RAFT performs marginally better than GMA under normal training conditions, with an overall AEPE of 6.71 compared to GMA's AEPE of 6.85 when trained with a mask weight of 1.00. This mask weight represents GMA's standard training setup, making SEA-RAFT the better model in this case. However, when we optimize GMA's training with a mask weight of 0.15, it achieves a significantly lower overall AEPE of 5.83, outperforming SEA-RAFT by a notable margin. Furthermore, while SEA-RAFT also exhibits relatively large standard deviations, they are still lower than those of GMA when trained with a mask weight of 1.00, indicating that SEA-RAFT's training is more stable in that scenario.

Lastly, the mask weights presented in Table 1 do not fully span the possible range from 0 to 1. As a result, it is possible that more optimal weight values exist which could achieve lower AEPEs.

4.3 Partial mask weight variation

In this experiment, we aim to evaluate how different partial mask weights impact GMA's accuracy on different datasets, specifically FlyingThings3D (test) [19], Sintel (train) [21] and KITTI-15 (train) [25]. We do this by applying the weighted partially masked training scheme described in Section 3.3.

We begin by generating partial masks, which are then used to train GMA on FlyingThings3D with varying partial mask weights w as defined in Equation 1. We use mask weights $w \in [0, 1]$ allowing us to explore diverse data points without risking instability by over-weighting out-of-frame regions.

Results are presented in Table 2, showing average end-point errors (AEPE) and standard deviations (in brackets) for different image regions, with the model initialized using the FlyingChairs checkpoint from GMA [3]. The final row (partial mask weight "1.00*") shows statistics for the FlyingThings3D checkpoint from [3] and is used as the baseline.

The table presents a pattern similar to what we observed in our previous experiment (Section 4.2), revealing a trade-off between accuracy on in-frame (IF) and out-of-frame (OF) regions, with KITTI-15 being the exception.

For FlyingThings3D (clean), we observe the lowest overall (All) AEPE of 2.95 at a partial mask weight of 0.15, outperforming the FlyingThings3D checkpoint from [3] with an AEPE of 3.14. Notably, this weight is consistent with the optimal value identified in our previous experiment (Section 4.2). We also observe a slight exception to the usual trade-off between IF and OF accuracy, as the lowest IF AEPE occurs at a mask weight of 0.10 with a value of 2.26. However, this difference is minor, as it is only 0.01 lower than the AEPE of 2.27 at mask weights of 0.00 and 0.05, likely due to standard deviations.

Partial mask weight	FlyingThings3D (test)						Sintel (train)						KITTI-15 (train)			
	Clean			Final			Clean			Final						
	All	IF	OF	All	IF	OF	All	IF	OF	All	IF	OF	All	IF	OF	F1 (%)
0.00	4.62 (0.02)	2.27 (0.02)	21.65 (0.03)	4.46 (0.02)	**2.12 (0.01)**	21.32 (0.12)	1.99 (0.00)	**0.95 (0.00)**	13.41 (0.04)	3.38 (0.02)	**2.30 (0.02)**	15.34 (0.04)	11.82 (0.14)	2.41 (0.02)	56.25 (0.77)	20.7 (0.06)
0.05	3.10 (0.03)	2.27 (0.04)	9.95 (0.04)	2.86 (0.04)	2.14 (0.03)	8.83 (0.09)	1.38 (0.00)	0.97 (0.00)	6.50 (0.00)	2.77 (0.01)	2.35 (0.02)	8.11 (0.02)	5.86 (0.00)	2.39 (0.00)	21.26 (0.02)	18.4 (0.09)
0.10	2.99 (0.01)	**2.26 (0.01)**	9.11 (0.00)	2.81 (0.01)	2.15 (0.01)	8.25 (0.02)	1.34 (0.01)	0.96 (0.00)	6.10 (0.06)	**2.72 (0.02)**	2.34 (0.03)	7.58 (0.03)	5.44 (0.03)	2.40 (0.00)	18.98 (0.19)	18.1 (0.14)
0.15	**2.95 (0.01)**	2.27 (0.01)	8.65 (0.04)	2.77 (0.00)	2.16 (0.01)	7.80 (0.06)	1.33 (0.01)	0.97 (0.00)	5.91 (0.02)	2.73 (0.02)	2.38 (0.02)	7.34 (0.05)	5.20 (0.08)	2.38 (0.02)	17.78 (0.31)	18.0 (0.07)
0.20	2.98 (0.02)	2.31 (0.02)	8.51 (0.01)	2.75 (0.02)	2.16 (0.02)	7.57 (0.01)	1.33 (0.00)	0.97 (0.00)	5.76 (0.06)	2.75 (0.02)	2.40 (0.02)	7.24 (0.06)	5.08 (0.03)	2.37 (0.00)	17.05 (0.18)	17.7 (0.13)
0.40	2.99 (0.02)	2.37 (0.01)	8.12 (0.01)	2.76 (0.02)	2.20 (0.02)	7.40 (0.03)	1.33 (0.00)	0.99 (0.01)	5.38 (0.05)	2.73 (0.00)	2.40 (0.01)	6.93 (0.04)	4.97 (0.00)	2.38 (0.01)	16.34 (0.04)	17.6 (0.00)
0.60	3.02 (0.02)	2.44 (0.02)	7.73 (0.00)	**2.75 (0.00)**	2.24 (0.00)	7.00 (0.00)	1.32 (0.00)	0.99 (0.00)	5.31 (0.02)	2.75 (0.01)	2.43 (0.01)	6.77 (0.08)	4.80 (0.01)	**2.34 (0.01)**	15.60 (0.01)	17.3 (0.04)
0.80	3.09 (0.02)	2.52 (0.01)	7.71 (0.08)	2.79 (0.00)	2.27 (0.01)	6.99 (0.08)	1.35 (0.03)	1.03 (0.02)	5.18 (0.19)	2.75 (0.00)	2.43 (0.00)	6.69 (0.09)	4.79 (0.05)	2.36 (0.03)	15.50 (0.11)	17.3 (0.17)
1.00	3.08 (0.03)	2.52 (0.03)	**7.58 (0.01)**	2.80 (0.01)	2.29 (0.01)	7.00 (0.01)	1.34 (0.01)	1.02 (0.01)	5.07 (0.01)	2.74 (0.02)	2.43 (0.02)	6.57 (0.03)	4.78 (0.04)	2.35 (0.01)	15.39 (0.16)	17.4 (0.09)
1.00*	3.14	2.60	7.65	2.80	2.32	**6.91**	**1.30**	1.02	**4.89**	2.74	2.44	**6.55**	**4.69**	2.36	**15.02**	**17.1**

Table 2: Average end-point error (AEPE) values and standard deviation (in brackets) for different image regions and datasets, obtained using GMA trained with the weighted partial mask scheme (Section 3.3) on FlyingThings3D. Models were initialized with the FlyingChairs checkpoint from GMA [3]. "All" indicates the overall AEPE, "IF" indicates AEPE on in-frame regions, and "OF" on out-of-frame regions. Additionally, we report the F1-all score on KITTI-15, which measures the percentage of pixels with an endpoint error greater than 3 pixels or 5% of the magnitude of the ground truth flow. Partial mask weight refers to the training weight for out-of-frame pixels, with in-frame pixels weighted at 1.0. Results are averaged over two runs per weight and rounded. Bold values indicate the lowest AEPEs in each column; in the case of ties, the value with the lowest standard deviation is bolded. The final row (partial mask weight "1.00*") shows statistics for the FlyingThings3D checkpoint from [3] and is used as the baseline. The table shows a trade-off between IF and OF accuracy for all datasets except KITTI-15, who's IF accuracy remains relatively constant. Notably, the All AEPE on FlyingThings3D (clean) improves from 3.14 to 2.95 at a partial mask weight of 0.15, and from 2.80 to 2.75 on FlyingThings3D (final) at a partial mask weight of 0.60, while Sintel and KITTI-15 show no significant All AEPE improvements.

On FlyingThings3D (final), we also observe a reduction in overall AEPE compared to our baseline model from [3]. At a partial mask weight of 0.60, the AEPE decreases from 2.80 to 2.75. Notably, a similar reduction occurs at a partial mask weight of 0.20, though with a higher standard deviation.

In contrast, on Sintel (clean), no partial mask weight surpasses the baseline All accuracy achieved by the FlyingThings3D checkpoint from [3]. This outcome appears to be driven by the superior OF accuracy on the Sintel dataset, indicating that its focus on out-of-frame regions contributed positively to the overall AEPE.

On Sintel (final), we observe slight improvements in overall AEPE compared to the baseline, with the largest reduction occurring at a partial mask weight of 0.10, where the AEPE decreases from 2.74 to 2.72. However, this difference is minimal, with a standard deviation of 0.02, suggesting that the variation may be attributed to standard deviation rather than indicating a significant accuracy improvement.

KITTI-15 does not exhibit the same trade-off pattern observed in other datasets. Here, IF accuracy remains relatively stable across different partial mask weights, ranging from 2.34 to 2.41, with the lowest value of 2.34 at a weight of 0.60. Additionally, the FlyingChairs3D GMA checkpoint consistently outperforms all other configurations quite significantly, including the one with a partial mask weight of 1.00, which would be expected to perform similarly. This suggests that standard deviation may have a greater impact on the results for KITTI-15, as larger standard deviations are observed across its runs.

Lastly, the partial mask weights used in Table 2 represent only a subset of possible values between 0 and 1. As a result, there may be more optimal weights that could further reduce AEPEs on specific datasets.

5 Discussion

Our findings reveal a trade-off in GMA's accuracy between in-frame and out-of-frame icons on the FlyingIcons dataset, dependent on the relative weights assigned to these icons. We identified an optimal overall average end-point error (AEPE) at an out-of-frame mask weight of 0.15, achieving a value of 5.83, which is significantly lower than standard GMA train-

ing at 6.85 and SEA-RAFT at 6.71. This outcome highlights a limitation of existing synthetic datasets, which lack not only the out-of-frame masks we used in the experiment but also complete ambiguity masks. Arguably, either could have been incorporated during dataset generation with minimal additional effort.

Rather than attempting to generate complete ambiguity masks after the dataset had already been created, we opted to produce partial out-of-frame masks for the FlyingThings3D dataset and applied the same weighted training method. This approach revealed a similar trade-off between in-frame and out-of-frame accuracy, consistent with our findings on the FlyingIcons dataset, with KITTI-15 standing as an exception. In particular, the weighted partially masked training method improved the accuracy on FlyingThings3D compared to the baseline GMA checkpoint. At a partial mask weight of 0.15 for the clean version, the AEPE decreased from 3.14 to 2.95. Similarly, at a partial mask weight of 0.60 for the final version, the AEPE decreased from 2.80 to 2.75. However, when we tested this method on Sintel and KITTI-15, we observed no significant overall accuracy improvements, which suggests either that the partial masks are too simplistic for these complex datasets or that even complete ambiguity masks would not suffice to improve accuracy.

These results demonstrate that this trade-off is present across multiple datasets, including FlyingIcons, FlyingThings3D, and Sintel, with the optimal out-of-frame mask weight for overall accuracy varying for each. This finding suggests an opportunity for models to focus on in-frame regions when needed, allowing them to specialize in scenarios where accurate in-frame predictions are more critical.

Although FlyingIcons is useful for exploring out-of-frame movement, it lacks some of the complexities present in real-world scenarios, such as scaling effects and three-dimensional motion, found in more realistic datasets such as FlyingThings3D. Additionally, FlyingIcons is limited to out-of-frame movements and does not address other types of ambiguities often encountered in optical flow, such as occlusions and complex motions.

Despite these limitations, the out-of-frame masks enabled effective weighted masked training on FlyingIcons, suggesting that integrating similar masks into existing datasets could be beneficial. However, as these findings are specific to the FlyingIcons dataset, future work could investigate the generalizability of this approach across more complex and realistic datasets to validate its effectiveness under a broader range of conditions.

Furthermore, while the weighted partially masked training approach only resulted in immediate overall accuracy improvements on FlyingThings3D, we believe that the concept of weighted masked training holds considerable potential. This initial test used relatively naive partial masks by design, as we aimed to determine whether even a straightforward approach could yield accuracy improvements. Future work could enhance this method by developing advanced mask generation strategies that capture additional ambiguities beyond out-of-frame movements. Additionally, creating synthetic datasets specifically designed with out-of-frame or complete ambiguity masks could further support research on improving model accuracy in these areas.

Lastly, since we tested the effects of weighted (partially) masked training exclusively on GMA, it remains unclear whether these findings generalize to other optical flow models. Future work could explore whether similar accuracy patterns emerge in different models.

References

[1] Y. Wang, L. Lipson, and J. Deng, "SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow," in *Computer Vision – ECCV 2024* (A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, eds.), vol. 15065, (Cham), pp. 36–54, Springer Nature Switzerland, 2025. Series Title: Lecture Notes in Computer Science.

[2] S. T. H. Shah and X. Xuezhi, "Traditional and modern strategies for optical flow: an investigation," *SN Applied Sciences*, vol. 3, p. 289, Feb. 2021.

[3] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9772–9781, October 2021.

[4] D. Rengasamy, M. Jafari, B. Rothwell, X. Chen, and G. P. Figueredo, "Deep Learning with Dynamically Weighted Loss Function for Sensor-Based Prognostics and Health Management," *Sensors*, vol. 20, p. 723, Jan. 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[5] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, Aug. 1981.

[6] Q. Chen and V. Koltun, "Full Flow: Optical Flow Estimation By Global Optimization over Regular Grids," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 4706–4714, IEEE, June 2016.

[7] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," vol. 4713, pp. 214–223, 09 2007.

[8] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), vol. 12347, pp. 402–419, Cham: Springer International Publishing, 2020. Series Title: Lecture Notes in Computer Science.

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766, 2015.

[10] A. Jahedi, L. Mehl, M. Rivinius, and A. Bruhn, "Multi-scale RAFT: Combining hierarchical concepts for learning-based optical FLow estimation." Issue: arXiv:2207.12163.

[11] A. Jahedi, M. Luz, L. Mehl, M. Rivinius, and A. Bruhn, "High resolution multi-scale RAFT (robust vision challenge 2022)."

[12] S. Saxena, C. Herrmann, J. Hur, A. Kar, M. Norouzi, D. Sun, and D. J. Fleet, "The surprising effectiveness of diffusion models for optical flow and monocular depth estimation." Issue: arXiv:2306.01923 Issue: arXiv:2306.01923.

[13] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "FlowFormer: A transformer architecture for optical flow."

[14] X. Shi, Z. Huang, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, and H. Li, "Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1599–1610, 2023.

[15] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," June 2018. arXiv:1709.02371 [cs].

[16] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8121–8130, June 2022.

[17] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. Goh, and H. Zhu, "Craft: Cross-attentional flow transformer for robust optical flow," 2022.

[18] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr, "Separable flow: Learning motion cost volumes for optical flow estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10807–10817, October 2021.

[19] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, June 2016. arXiv:1512.02134 [cs, stat].

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[21] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision (ECCV)*, pp. 611–625, 2012.

[22] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[23] L. Mehl, A. Jahedi, J. Schmalfuss, and A. Bruhn, "Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[24] Y. Han, K. Luo, A. Luo, J. Liu, H. Fan, G. Luo, and S. Liu, "Realflow: Em-based realistic optical flow dataset generation from videos," in *Computer Vision – ECCV 2022* (S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds.), (Cham), pp. 288–305, Springer Nature Switzerland, 2022.

[25] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[26] Flaticon, "Free vector icons." https://www.flaticon.com, 2024. Accessed: 2024-10-08.

[27] E. W. Weisstein, "Affine transformation." https://mathworld.wolfram.com/AffineTransformation.html, 2024. Accessed: 2024-10-22.

3 Supplementary Material

3.1 Deep Learning

3.1.1 What is deep learning?

Deep learning is a subset of machine learning and artificial intelligence that mimics the functioning of the human brain in processing data and creating patterns for use in decision-making. It uses multi-layered artificial neural networks to identify patterns and make predictions. It has significantly enhanced our ability to process large amounts of structured and unstructured data, enabling breakthroughs across various domains. For instance, in image recognition, deep learning models excel in tasks such as object detection, facial recognition, and medical image analysis. This has made them invaluable in fields ranging from healthcare—where they assist in diagnosing diseases and personalizing treatment plans—to autonomous vehicles, where deep learning aids in real-time decision-making, navigation, and object recognition in complex environments. Moreover, in natural language processing, deep learning models power applications such as language translation, sentiment analysis, and text summarization, improving communication across languages and enhancing customer experience in industries such as finance and customer service.

3.1.2 Non-linearity

At the heart of deep learning lie artificial neural networks, which are inspired by the structure and functioning of the human brain. These networks are composed of layers of interconnected nodes, or neurons, that process and transform inputs to produce meaningful outputs. While a single perceptron, the basic unit of these networks, performs linear classification, stacking multiple layers of perceptrons enables the network to tackle more complex problems.

The key to handling non-linear relationships lies in the activation functions within each layer. Activation functions, such as sigmoid, tanh, and ReLU, introduce the necessary non-linearity by transforming the output of each neuron in a way that allows the network to model intricate patterns in data. ReLU, often favoured for its computational efficiency and simplicity, helps mitigate issues such as vanishing gradients in deeper networks, allowing them to perform well on large-scale tasks.

3.1.3 Training networks

Training a deep learning model involves iteratively optimizing its parameters to minimize the error between predicted and actual outputs. This process begins with forward propagation, where inputs are passed through the network to generate predictions. The predictions are then compared to the true values using a loss function, such as Mean Squared Error (MSE) for regression tasks or Cross-Entropy for classification tasks. Optimization algorithms, such as Stochastic Gradient Descent (SGD) and Adam, guide the backward propagation phase, where the model's parameters are adjusted in a direction that reduces the error.

Evaluating a model's performance relies on metrics such as accuracy, precision, and recall, which provide insights into how well the model performs on the data. Training often includes hyperparameter tuning, where parameters such as learning rate, batch size, and network depth are adjusted to optimize performance. To prevent overfitting, regularization techniques such as dropout and weight decay are commonly applied. Dropout, for instance, reduces the likelihood of overfitting by randomly "dropping" neurons during training, thereby promoting a more generalized model. Other techniques such as batch normalization help stabilize training by normalizing the input data within mini-batches. Additionally, in scenarios with class imbalance, approaches such as class weighting or data augmentation can be employed to ensure balanced learning.

3.1.4 Masking

In deep learning, masking is a technique used to selectively emphasize certain parts of the input data or ignore irrelevant information during training and inference. A mask acts as a filter on the input, output, or intermediate layers of a neural network, specifying which elements should be processed and which should be disregarded. Typically, masks are binary or weighted matrices

that align with the data dimensions, where values of 1 include elements, and values of 0 exclude them. Masking can also be applied in loss calculations, allowing models to prioritize specific data points, such as emphasizing certain regions in images or focusing on particular time frames in video data.

3.1.5 Vanishing gradient

The vanishing gradient problem occurs in deep neural networks, particularly those with many layers, where gradients used to update model weights during backpropagation become extremely small as they propagate backward through the network. This issue is most common in networks with sigmoid or tanh activation functions, which can squash input values into a narrow range, causing gradients to shrink as they pass through each layer. As a result, the weights in earlier layers receive minimal updates, hindering the network's ability to learn and capture complex patterns. This makes training deep networks challenging, as they become slow to converge or may even stop learning entirely. Solutions such as the ReLU activation function, which avoids squashing by passing through values above zero unchanged, and techniques such as batch normalization, help alleviate the vanishing gradient problem by maintaining larger gradient values across layers.

3.1.6 Architectures

Fully Connected Neural Networks Fully Connected Neural Networks, often referred to as feedforward networks, consist of an input layer, multiple hidden layers, and an output layer, where each neuron in one layer is fully connected to every neuron in the next. While these networks are foundational, they struggle with high-dimensional structured data, making them less suited for tasks like image and sequence analysis.

Convolutional Neural Networks Convolutional Neural Networks (CNNs), in contrast, are explicitly designed for handling structured grid-like data, such as images. By using shared-weight filters that capture local patterns, CNNs achieve parameter efficiency and exhibit translation equivariance, allowing them to detect patterns regardless of their location in the input. Pooling layers further refine the model by downsampling feature maps, reducing sensitivity to spatial variations. Despite their effectiveness, training deep CNNs can be challenging due to issues such as vanishing gradients.

Recurrent Neural Networks Recurrent Neural Networks (RNNs) specialize in sequence modelling, making them well-suited for tasks that involve time series or sequential data, such as speech recognition and language translation. The structure of an RNN allows it to remember previous inputs, though basic RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-term dependencies. Advanced variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, address these limitations by incorporating memory cells, enabling them to handle longer sequences effectively.

(Variational) Autoencoders (Variational) Autoencoders (VAEs) represent another class of networks, primarily used for representation learning and generative tasks. Autoencoders learn to encode input data into a lower-dimensional representation before reconstructing it, capturing key features of the data. VAEs extend this by introducing a probabilistic approach, learning a distribution over the latent space, which makes them suitable for generative tasks where producing new, realistic samples is desired. They find applications in areas such as image synthesis, anomaly detection, and data denoising.

Generative Adversarial Networks Generative Adversarial Networks (GANs) consist of two networks: a generator that produces synthetic data and a discriminator that differentiates between real and generated data. The generator learns to create increasingly realistic samples by trying to "fool" the discriminator, which, in turn, becomes better at identifying fake samples. This dynamic has made GANs highly effective in fields such as image generation, style transfer, and

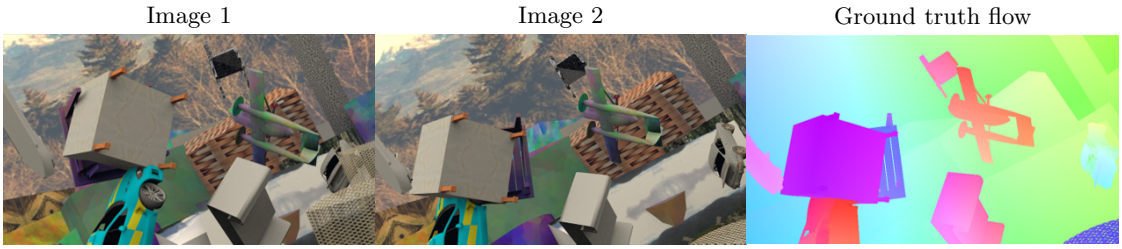


Figure 1: Example of optical flow between two consecutive frames. From left to right: the first frame (Image 1), the second frame (Image 2), and the ground truth optical flow. The optical flow (right) represents the pixel-wise displacement between Image 1 and Image 2, with colours indicating the direction and magnitude of movement according to the colour wheel. This ground truth flow is used to evaluate the accuracy of optical flow algorithms on this image sequence.

Figure 2: Optical flow colour wheel visualization. This colour wheel represents the direction and magnitude of optical flow vectors relative to the centre pixel, with each colour corresponding to a specific direction of motion. The hue indicates the flow direction, while the brightness represents the magnitude of movement, where brighter colours indicate faster motion. For example, rightward motion is shown in red, upward in purple, leftward in cyan, and downward in yellow. The centre of the wheel corresponds to zero motion. This visualization is used to interpret the direction and intensity of motion in optical flow fields across the image.

domain adaptation. However, GANs are notoriously difficult to train due to their sensitivity to training dynamics and convergence issues.

Attention Mechanisms and Transformers Attention Mechanisms and Transformers have revolutionized sequence modelling by enabling parallel processing of sequential data. Attention mechanisms allow the network to focus on the most relevant parts of the input sequence, making it especially effective for tasks involving language and context. Transformers, such as BERT [1] and GPT [2], leverage self-attention to learn complex relationships between sequence elements, leading to significant advances in NLP. These models have also been adapted for computer vision tasks, as seen with Vision Transformers (ViTs) [3], and for multi-modal applications that span text, image, and other data types.

3.2 Optical Flow

Optical flow is the task of predicting per-pixel vectors corresponding to the apparent motion between two frames of, for example, a video. See Figure 1 for an example of an optical flow sample and Figure 2 for a visualization of what the ground truth flow colours mean. Optical flow is used in combination with a number of other computer vision tasks such as: video coding, segmentation, tracking and multi view-reconstruction. Optical flow has also been proven useful

in other fields, such as: fluid mechanics, solar physics, autonomous driving, biomedical images, breast tumours, bladder cancer surveillance, traffic monitoring, virtual reality, face recognition and tracking, and action recognition videos. [4]

3.2.1 Traditional Optical Flow Methods

Traditionally, optical flow has been tackled using handcrafted feature-based approaches designed to estimate the apparent motion between consecutive frames in a video. Among these, variational methods have been particularly successful, as they minimize an energy cost function derived from two main assumptions: brightness constancy and spatial smoothness. The brightness constancy assumption posits that the intensity of a pixel remains constant across frames, while the smoothness assumption enforces a certain level of coherence in the flow field, promoting similar motion vectors in neighbouring regions. Based on how they approach this optimization, these methods can be categorized into global and local methods.

Global methods Global methods aim to optimize the flow field across the entire image by minimizing the energy cost function holistically. A classical example of a global method is the Horn-Schunck method [5], which formulates optical flow as a global optimization problem. This approach utilizes both brightness and smoothness constraints to iteratively estimate the optical flow for each pixel. While effective in providing a dense, smooth flow field, global methods can struggle under conditions of varying illumination. Furthermore, global methods are computationally intensive due to the iterative nature of solving for the entire image at once, and they often produce overly smooth results that may blur motion boundaries in complex scenes.

Local methods Local methods, on the other hand, focus on estimating optical flow within small neighbourhoods or individual pixels. These methods operate on the local consistency assumption, which holds that neighbouring pixels exhibit similar motion patterns. This local focus enables the estimation of optical flow without relying on information from the entire image, thus reducing computational complexity. Local methods typically use techniques such as block matching, which involves comparing pixel blocks between consecutive frames, or gradient-based approaches, such as the Lucas-Kanade method [6], which assumes a locally constant flow within each neighbourhood. While these methods are efficient and effective for small, subtle motions, they typically struggle with larger displacements.

3.2.2 Deep Learning Optical Flow

The emergence of deep learning has brought significant advancements to optical flow estimation by replacing traditional, rule-based techniques with data-driven models that learn complex motion patterns directly from large datasets. Classical methods rely on assumptions such as brightness constancy and spatial smoothness and often struggle with diverse motion types, such as large displacements and non-rigid movement. In contrast, deep learning-based approaches bypass these handcrafted features, allowing neural networks to learn representations that generalize better to complex scenes.

The use of deep learning also allows for end-to-end training, where the model learns an entire pipeline of feature extraction, motion estimation, and refinement from pairs of images. This holistic approach contrasts with traditional methods, which often treat these steps as separate, handcrafted components. By training the network to optimize for accurate flow estimation directly, deep learning models can learn a broader range of motion patterns, which may include large displacements, occlusions, and changes in viewpoint.

Training deep learning models for optical flow typically requires large datasets with ground truth flow fields, which are often generated synthetically due to the difficulty of obtaining accurate motion data in real-world scenes. By training on synthetic datasets, such as FlyingChairs [7] or FlyingThings3D [8], models can learn the basic structures of motion patterns. For real-world applications, fine-tuning on datasets such as MPI Sintel [9] and KITTI [10] helps improve generalization to more diverse environments. More about datasets in Section 3.2.3.

In terms of supervision, deep learning models for optical flow typically use loss functions that measure the difference between predicted and ground truth flow fields. A primary loss function is the $\mathcal{L}_1$ loss, which calculates the absolute difference between each predicted flow vector and the corresponding ground truth vector. This loss is widely used in optical flow because it is less sensitive to outliers than Euclidean (or $\mathcal{L}_2$) loss, providing more balanced learning. However, model accuracy is generally evaluated using the $\mathcal{L}_2$ loss, commonly referred to as the Average End-Point Error (AEPE).

While deep learning-based optical flow has significantly advanced the field, it still faces challenges, particularly in handling occlusions and non-visible regions between frames. When parts of the scene are hidden or missing from view, models struggle to estimate motion accurately due to the lack of consistent information. Although occlusion-aware architectures are being developed to address this, achieving reliable predictions in these areas remains an active area of research.

3.2.3 Datasets

Optical flow research has been supported by several key datasets, each offering unique features and challenges to help evaluate and benchmark models. While some datasets are synthetic and intended primarily for pretraining, others provide real-world data for testing and validating model accuracy.

FlyingChairs FlyingChairs, introduced by [7], contains 22,872 synthetic image pairs, each with ground truth optical flow. The images depict chairs moving across varied background scenes, typically skylines or cityscapes, under various translations, rotations, and scales. Due to its synthetic nature, FlyingChairs has precisely labelled ground truth, making it ideal for pretraining optical flow models before fine-tuning on more complex datasets. However, it is not meant for final testing, as its primary purpose is to provide a foundational training set that helps models learn basic motion patterns.

FlyingThings3D Following the concept of FlyingChairs, the FlyingThings3D dataset, proposed by [8], expands on this idea with a richer set of 3D synthetic data. It consists of 25,000 stereo frames of objects flying across diverse backgrounds, incorporating depth and realistic object interactions. The dataset includes additional modalities such as disparity and segmentation masks, making it useful for multitask training. Although it is another synthetic dataset, FlyingThings3D presents a more complex pretraining environment than FlyingChairs due to its 3D nature and variety of objects. Like FlyingChairs, it is intended for pretraining, not for final evaluation.

FlyingThings3D has two versions:

- Clean: Contains simplified scenes with fewer artefacts and clearer object boundaries, ideal for training on basic 3D motion.
- Final: Includes added noise, motion blur, and other realistic effects, simulating more complex real-world conditions.

These versions offer flexibility in training, from controlled conditions to scenarios that closely mimic real-world challenges.

MPI Sintel The MPI Sintel dataset, developed by [9] and derived from the open-source animated film "Sintel" by the Blender Foundation, is a well-established benchmark for evaluating optical flow models. It provides high-resolution images with ground truth optical flow, depth, and motion segmentation, offering a realistic yet challenging environment for model testing. Sintel also has two main versions:

- Clean: This version provides sharp, detailed frames without additional effects, focusing on high-quality imagery for motion analysis.
- Final: The Final version adds complex rendering effects such as motion blur, atmospheric effects, and lighting changes, creating a more challenging environment for models.

Testing on Sintel's test data requires researchers to submit results to an online evaluation server, ensuring a fair and consistent benchmarking process by restricting access to the ground truth for test sequences. This constraint helps avoid overfitting and allows the research community to compare results reliably.

KITTI The KITTI dataset, developed by [10], is a cornerstone for evaluating computer vision models in autonomous driving. It contains real-world data collected from a vehicle equipped with cameras and LIDAR sensors and provides high-resolution images across various tasks, including optical flow, visual odometry, object detection, and 3D scene reconstruction. KITTI's optical flow data includes ground truth for some but not all pixels, due to the inherent challenges of obtaining precise flow measurements in real-world driving scenarios. As with Sintel, KITTI uses an online submission system for test set evaluations to promote fair comparison and prevent overfitting. The dataset's diversity, covering different lighting conditions, road types, and dynamic scenes, makes it a valuable benchmark for evaluating the robustness of optical flow models.

Middlebury The Middlebury dataset, introduced by [11], is one of the earliest optical flow benchmarks and remains a widely used resource for evaluating flow algorithms. It includes a range of synthetic and real-world scenes, covering both indoor and outdoor environments with a variety of motions, textures, and lighting conditions. Middlebury provides high-quality ground truth obtained through stereo camera setups and manual alignment, which ensures a precise benchmark for fine-tuning and testing. Despite its limited size compared to more recent datasets, Middlebury's diverse content and accuracy continue to make it relevant in assessing the precision of optical flow algorithms.

Spring The Spring dataset, introduced by [12], represents a recent addition to optical flow and scene flow benchmarking. While it is a synthetic dataset, Spring is notable for its high-resolution, realistic scenes, which feature complex and dynamic motions, including non-rigid deformations and occlusions. The dataset includes a wide range of motion types, from small details such as grass and hair movements to larger dynamic actions such as flight and chasing sequences. This makes it particularly valuable for testing models on challenging, highly detailed scenarios that reflect realistic visual effects

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] S. T. H. Shah and X. Xuezhi, "Traditional and modern strategies for optical flow: an investigation," vol. 3, no. 3, p. 289, 2021.

[5] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, Aug. 1981.

[6] D. Patel and S. Upadhyay, "Optical Flow Measurement using Lucas Kanade Method," *International Journal of Computer Applications*, vol. 61, pp. 6–10, Jan. 2013.

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766, 2015.

[8] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, June 2016. arXiv:1512.02134 [cs, stat].

[9] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision (ECCV)*, pp. 611–625, 2012.

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[12] L. Mehl, A. Jahedi, J. Schmalfuss, and A. Bruhn, "Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[13] S. Saxena, C. Herrmann, J. Hur, A. Kar, M. Norouzi, D. Sun, and D. J. Fleet, "The surprising effectiveness of diffusion models for optical flow and monocular depth estimation." Issue: arXiv:2306.01923 Issue: arXiv:2306.01923.

[14] Q. Dong, B. Zhao, and Y. Fu, "Open-DDVM: A reproduction and extension of diffusion model for optical flow estimation." Issue: arXiv:2312.01746 Issue: arXiv:2312.01746.

[15] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), vol. 12347, pp. 402–419, Cham: Springer International Publishing, 2020. Series Title: Lecture Notes in Computer Science.

[16] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9772–9781, October 2021.

A Appendix

A.1 High Resolution Diffusion Optical Flow

In this first topic, we explored optical flow estimation using diffusion models. Google introduced the DDVM model [13], which applies a general-purpose denoising diffusion model to optical flow without using any specific architectural adjustments or optimization techniques traditionally found in optical flow models. However, training DDVM requires a significant amount of computational resources, making it impractical for us to reproduce the results on a large scale due to resource constraints.

To address these challenges, Open-DDVM [14] was developed as an adaptation of DDVM with reduced computational demands. This version incorporates a correlation volume, similar to that used in RAFT [15], which significantly decreases the resources needed, albeit with a slight accuracy reduction. However, the correlation volume only samples every eighth pixel, both horizontally and vertically, leading to potential gaps in information. This design choice suggested that Open-DDVM might miss important details between sampled pixels, which could impact accuracy by effectively reducing the resolution.

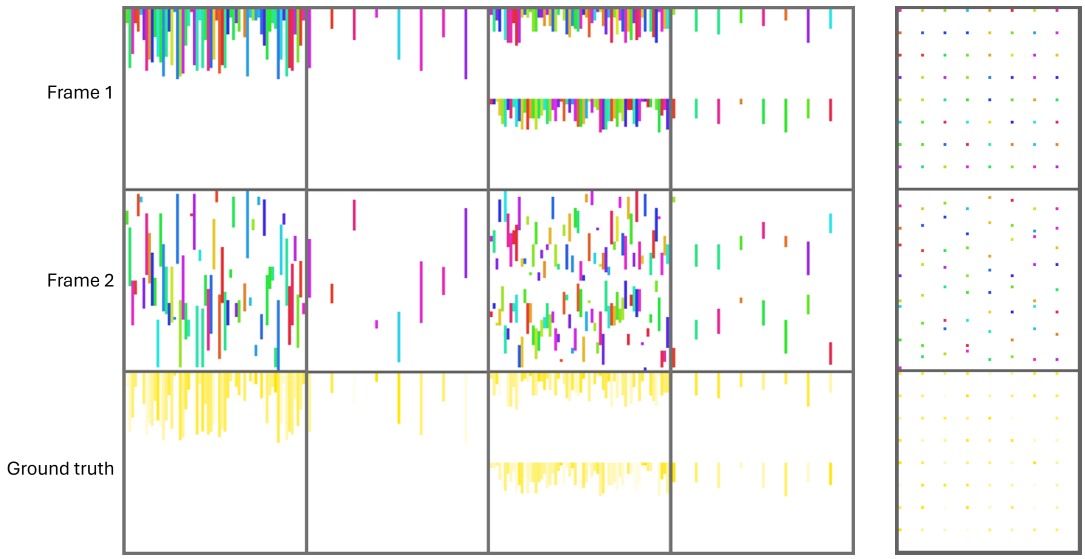


Figure 3: Samples from different versions of the MovingGrass dataset, where "grass" blades of different colours move vertically. Each column represents a sample, with the top row showing the first frame, the middle row showing the second frame, and the bottom row showing the ground truth flow. The dataset includes various configurations: two versions with a single row of moving blades, one where every column has a moving blade and one where every eighth column does; two versions with two rows of moving blades, again one with every column moving and another with every eighth column; and finally, a version where only every eighth row and column contains a moving blade of grass.

The plan was to verify this hypothesis and, once confirmed, develop a solution to recapture the lost information and improve the resolution. We tested this hypothesis by creating a synthetic dataset called MovingGrass which we used to train Open-DDVM. The samples contain coloured blades of "grass" moving vertically, with variations in spacing—either every column or every eighth column. Figure 3 displays some samples from different dataset variations.

Given the setup of the correlation volume, we anticipated that the model would perform better on the dataset where only every eighth column had a moving blade of grass, as this configuration would allow the model to "see" a larger percentage of the moving blades compared to when every column had a blade. We initially tested with single rows of blades, but after observing no accuracy difference, we increased the complexity by adding multiple rows and by significantly increasing the width of the samples. However, these changes did not yield the expected gap in accuracy.

We hypothesized that the simplicity of the dataset might be limiting the utility of the correlation volume. To test this, we conducted a second experiment by training a standard DDVM model without the correlation volume. Although DDVM typically has large computational requirements, the small 64 by 64 pixel sample size made this doable. The results revealed no accuracy improvement, suggesting that the correlation volume had minimal impact on these toy datasets.

After additional attempts to increase dataset difficulty and observing no significant differences in accuracy or training stability, we concluded that further progress would either require a more complex dataset, which would most likely mean larger computational requirements, or a shift to a different research topic. Ultimately, we chose the latter path.

A.2 Optical Flow As A Class Imbalance Problem

The second topic we explored consisted of framing optical flow as a class imbalance problem.

A.2.1 Introduction

We identified two key issues affecting optical flow. The first issue relates to the evaluation metric, specifically the average end-point error (AEPE) or L2 loss. This metric calculates the average per-pixel difference between the predicted flow and the ground truth flow. However, when this error is divided by the product of the image's width and height, the overall value can become quite small, especially at high resolutions. As a result, errors in small, detailed regions tend to be lost in the average and are largely ignored during evaluation, reducing the model's sensitivity to these areas.

The second issue involves an imbalance in the dataset itself. Optical flow datasets typically contain large regions with minimal or uniform movement between frames, while regions with detailed, complex motion in various directions are far less common. Consequently, models trained on such data tend to perform poorly on areas with underrepresented movement details, as shown in Figure 4. This imbalance suggests that optical flow models are impacted by a class imbalance problem, where the "classes" correspond to the detail levels of movement across different regions of the flow.

Our objective was to make three key contributions:

- An optical flow evaluation metric that accurately reflects model accuracy on high-detail areas.
- A demonstration that current optical flow methods are affected by a class imbalance problem.
- An improved training method that mitigates the effects of class imbalance in optical flow models.

While we partially succeeded with the first objective, we effectively demonstrated the second, and achieved the third on the FlyingChairs dataset, though it did not generalize to other datasets.

A.2.2 Method

Labelling the data To view optical flow as a class imbalance problem, we first need to assign labels. These labels are generated by dividing the ground truth flow into patches of $n \times n$ pixels and calculating a detail score for each patch, which serves as its label.

The detail score for a single flow direction is calculated using the following formula:

$$\boldsymbol{D(f)} = \nabla(\boldsymbol{f}) * \boldsymbol{k_\mu}. \tag{1}$$

In this equation, ∇ denotes a binary Canny edge detector, $\boldsymbol{f}$ is the ground truth flow matrix, and $\boldsymbol{k_\mu}$ represents a mean kernel of size $n \times n$. The edge detector output is convolved with this kernel to produce a detail score, indicating the percentage of edge pixels within each patch. This results in a score matrix with dimensions of $W/n \times H/n$, which we then linearly interpolate back to the original $W \times H$ dimensions for easier usage. To ensure each patch receives a single detail score, we apply the convolution with a stride of n.

To combine the detail scores from both flow directions, we calculate the per-pixel average of the detail scores for each direction:

$$\boldsymbol{D_\mu} = \frac{\boldsymbol{D(f_1)} + \boldsymbol{D(f_2)}}{2}. \tag{2}$$

In this equation, $\boldsymbol{f_1}$ represents the ground truth flows in the horizontal direction, while $\boldsymbol{f_2}$ represents the ground truth flows in the vertical direction. The resulting matrix, $\boldsymbol{D_\mu}$, provides a single detail score for each pixel.

Scaling the loss With the per-pixel detail scores calculated, we can assign weights to each pixel based on its corresponding detail score. These weights are determined by first calculating the detail score distribution for the training dataset, which is then used to create scaling strategies. Each scaling strategy consists of a list of values, where each index corresponds to a detail score bin. These strategies are denoted as DSx, where x indicates the strategy number. In these strategies, δ represents the dataset distribution, and we construct the arrays as we would in Python. Below, we present two of the more successful scaling strategies, DS9 and DS10, starting with DS9:

$$DS6 = [1, 2 + \delta[1:]_{normalized}], \tag{3}$$

$$DS7 = DS6^2, \tag{4}$$

$$DS9 = 1 + DS7 - \frac{\sum(\delta \circ DS7)}{\sum \delta}. \tag{5}$$

In this equation, DS9 is constructed from two intermediate scaling strategies that proved ineffective by themselves, DS6 and DS7. For DS6, the scaling value for the first bin is set to 1. From the second bin onward, we calculate each value by adding 2 to the normalized distribution values starting from the second element. DS7 is then obtained by squaring DS6. Finally, to create DS9, we subtract the weighted average of DS7 (normalized by the distribution) and add 1, ensuring that the learning rate is not implicitly lowered or raised.

Next, we define DS10:

$$\alpha = \frac{1}{\sqrt{\delta}}, \tag{6}$$

$$\beta = \frac{\sum(\delta \circ \alpha)}{\sum \delta}, \tag{7}$$

$$DS10 = \frac{\alpha}{\beta}, \tag{8}$$

which takes a more conventional approach to addressing class imbalances by dividing by the square root of the distribution, δ. We then further normalize by dividing by the weighted average, again ensuring that the learning rate is not implicitly lowered or raised.

Evaluation metric Typical optical flow evaluation metrics fail to properly reflect accuracy on high-detail areas. Specifically, the average end-point error (AEPE) has the drawback of averaging out errors in small, high-detail regions.

To address this, we propose a novel evaluation metric called the Bin Weighted-Average Error Ratio (BWAER). As the name suggests, this metric is calculated by taking a weighted average of the bin AEPEs, where each bin is weighted by its index. First, we calculate the Bin-Weighted Average Error (BWAE) as follows:

$$BWAE = \frac{\sum_i AEPE_i \cdot i}{\sum_i i}. \tag{9}$$

In this formula, i represents the bin index, starting at 1. Bins with zero occupancy are ignored in the calculation. We then obtain the BWAER by normalizing with the overall AEPE:

$$BWAER = \frac{BWAE}{AEPE}. \tag{10}$$

By assigning higher weights to bins with higher detail levels, this metric emphasizes high-detail areas. The BWAER approaches 1.0 when the model performs uniformly across all detail levels, and it increases when accuracy varies significantly across detail levels. However, because BWAER is normalized by AEPE, it is essential to consider both metrics together to avoid misinterpreting the results.

A.2.3 Experiments

Class imbalance Figure 4 demonstrates the class imbalance in current optical flow models. For the first thirty bins, we plotted the AEPE per bin for various models, along with each bin's contribution to the overall AEPE. As shown, there is a negative correlation between the average occupancy of a bin in the training data and its AEPE during evaluation. In other words, bins with higher average occupancy tend to have lower AEPE, while bins with lower occupancy generally exhibit higher AEPE.

Furthermore, we see each bin's contribution to the overall AEPE as a percentage. Across all dataset/model combinations, the bins with the highest AEPE contributions typically fall between bins 8 and 14, which account for only a small percentage of the data.

Detail scaling Figure 5 presents the results of applying the detail scaling strategies DS9 from Equation 3 and DS10 from Equation 6. On the FlyingChairs dataset, both scaling strategies improve the AEPE, reducing the error to 0.767 for DS9 and 0.791 for DS10, compared to the baseline of 0.823.

Figure 6 further illustrates individual cases where the DS10 scaling strategy (referred to as "Ours") enhanced the final flow predictions. The green-boxed areas highlight improved detail levels relative to the GMA checkpoint from [16] (referred to as "Theirs").

However, these improvements did not generalize to other datasets, such as FlyingThings3D, indicating a limited transferability of the scaling strategies.

Additionally, we present the BWAER values for these configurations, where higher BWAER values indicate a greater imbalance in accuracy across different bin detail levels. Both scaling strategies show improvement, with DS10 achieving the most significant reduction, decreasing from 10.415 to 9.484.

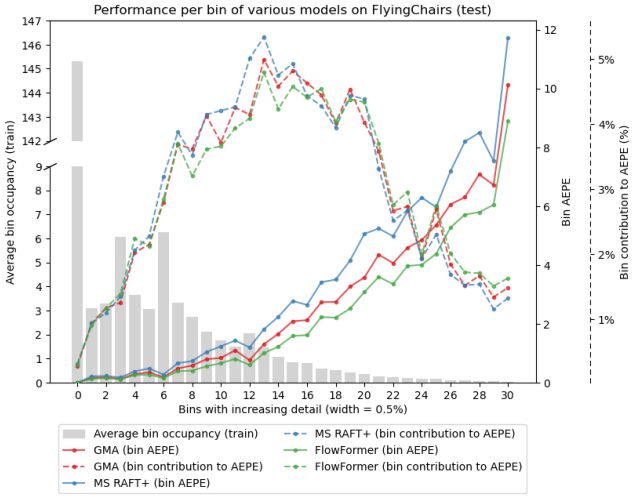
A.2.4 Discussion

We demonstrated that optical flow models face a class imbalance issue based on the detail levels in the flow fields of training data, where high-detail regions with complex motion are underrepresented in standard datasets. This imbalance affects model accuracy, as these regions contribute disproportionately to the AEPE relative to their frequency in the data.

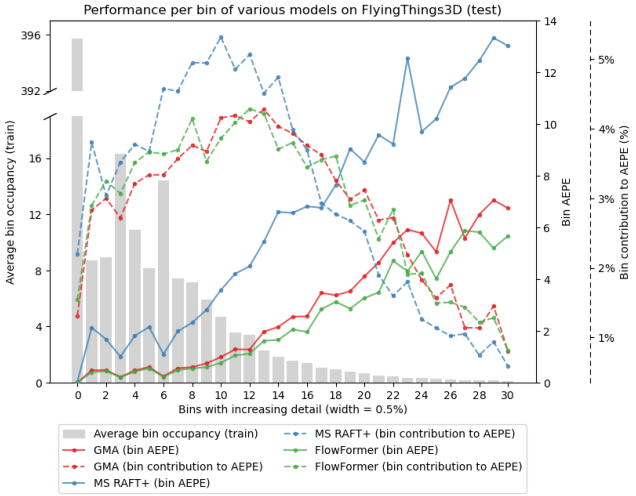
To address this, we applied detail scaling strategies (DS9 and DS10) based on training data distributions, achieving improved accuracy on the FlyingChairs dataset. By weighting higher-detail patches more heavily, these strategies reduced AEPE values, indicating that the approach can improve model accuracy on simpler datasets. However, this improvement did not generalize to more complex datasets such as FlyingThings3D and Sintel, where additional motion complexities likely diminish the effectiveness of detail scaling alone.

We also introduced a custom metric, the Bin Weighted-Average Error Ratio (BWAER), to assess accuracy on high-detail areas. Although BWAER provided a more nuanced view by weighting errors according to detail, it proved challenging to interpret and remained dependent on AEPE, limiting its practical utility.

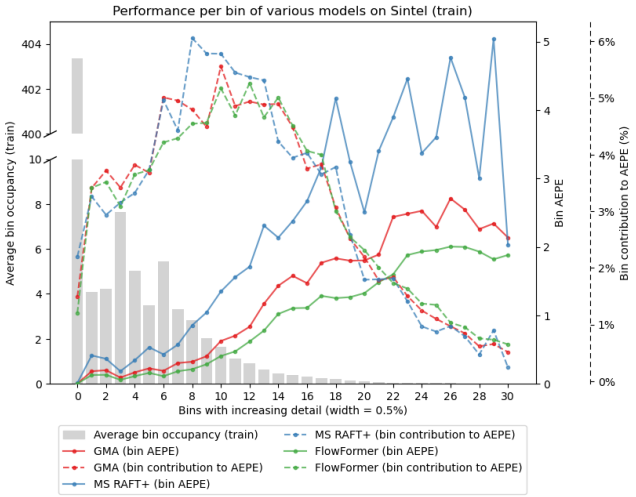
In conclusion, while detail scaling shows promise on simpler datasets, it requires further refinement to be effective on more complex data. BWAER offers some insight into accuracy across detail levels, but highlights the need for more intuitive, robust evaluation metrics in optical flow. As a result, we decided to pursue a different topic within the field of optical flow that showed more promise, which can be found in Section 2.



(a) The bin AEPE and bin contribution to the AEPE as a percentage for several models on the FlyingChairs dataset.



(b) The bin AEPE and bin contribution to the AEPE as a percentage for several models on the FlyingThings3D dataset.



(c) The bin AEPE and bin contribution to the AEPE as a percentage for several models on the Sintel dataset.

Figure 4: The bin AEPE and bin contribution to the AEPE as a percentage for several models on the several datasets. Additionally, the average bin occupancy is shown for each dataset.

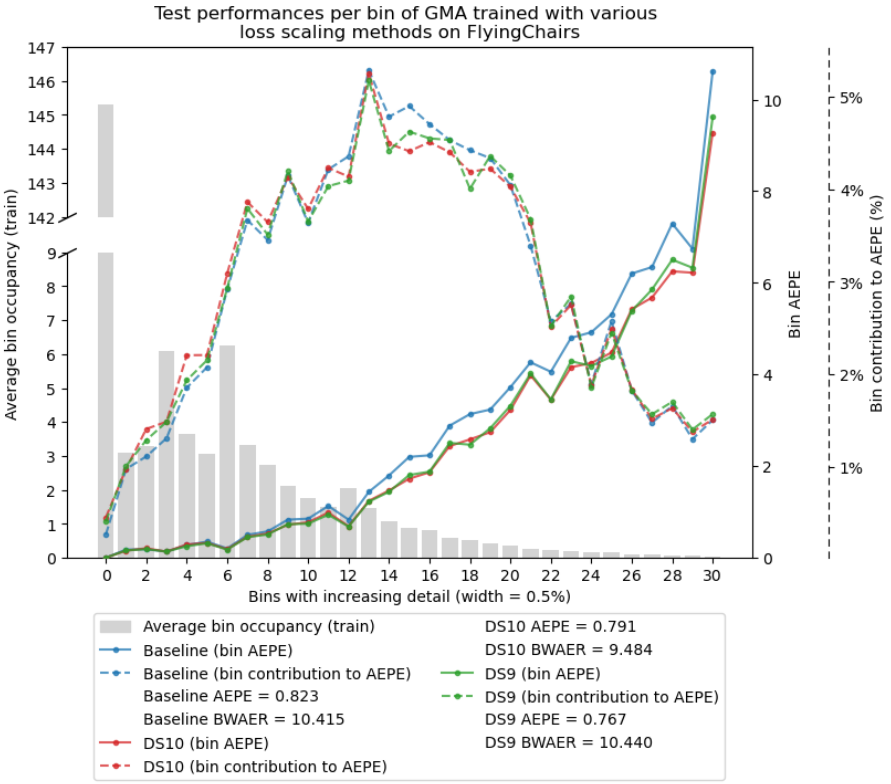


Figure 5: The bin AEPE and bin contribution to the AEPE as a percentage for several scaling strategies on the FlyingChairs dataset. Additionally, the average bin occupancy is shown for each dataset. We see improvements compared to the baseline for both DS9 and DS10 in terms of both overall AEPE and BWAER.

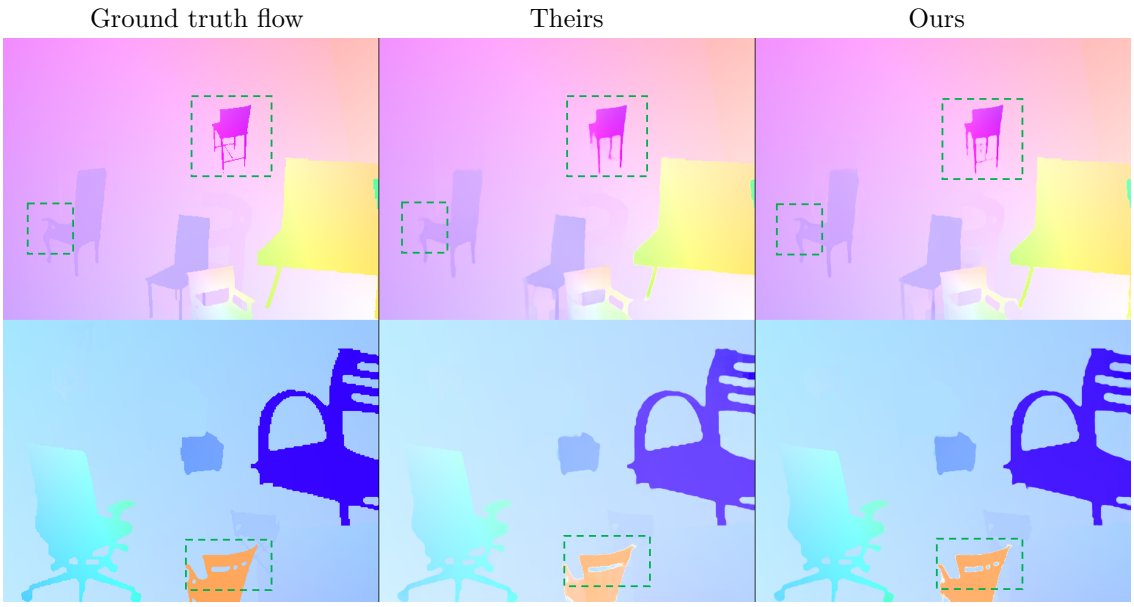


Figure 6: Two samples illustrating the effect of detail scaling on high-detail flow areas. From left to right: the ground truth flow, the flow predicted by the FlyingChairs checkpoint from GMA [16] ("Theirs"), and the flow predicted by our checkpoint using the DS10 detail scaling strategy ("Ours") from Equation 6. While DS10 demonstrates enhanced accuracy on these high-detail areas, these improvements did not generalize to other datasets, such as FlyingThings3D.