ScaleFlow++: Robust and Accurate Estimation of 3D Motion from Video

Han Ling, Yinghui Sun, Quansen Sun and Yuhui Zheng, *Member, IEEE,*

***Abstract*—Perceiving and understanding 3D motion is a core technology in fields such as autonomous driving, robots, and motion prediction. This paper proposes a 3D motion perception method called ScaleFlow++ that is easy to generalize. With just a pair of RGB images, ScaleFlow++ can robustly estimate optical flow and motion-in-depth (MID). Most existing methods directly regress MID from two RGB frames or optical flow, resulting in inaccurate and unstable results. Our key insight is cross-scale matching, which extracts deep motion clues by matching objects in pairs of images at different scales. Unlike previous methods, ScaleFlow++ integrates optical flow and MID estimation into a unified architecture, estimating optical flow and MID end-to-end based on feature matching. Moreover, we also proposed modules such as global initialization network, global iterative optimizer, and hybrid training pipeline to integrate global motion information, reduce the number of iterations, and prevent overfitting during training. On KITTI, ScaleFlow++ achieved the best monocular scene flow estimation performance, reducing SF-all from 6.21 to 5.79. The evaluation of MID even surpasses RGBD-based methods. In addition, ScaleFlow++ has achieved stunning zero-shot generalization performance in both rigid and nonrigid scenes. Code is available at https://github.com/HanLingsgjk/CSCV.**

***Index Terms*—Motion-in-depth, Cross scale, Correlation volume, Scene flow, Optical flow, Monocular 3d, Time-to-collision.**

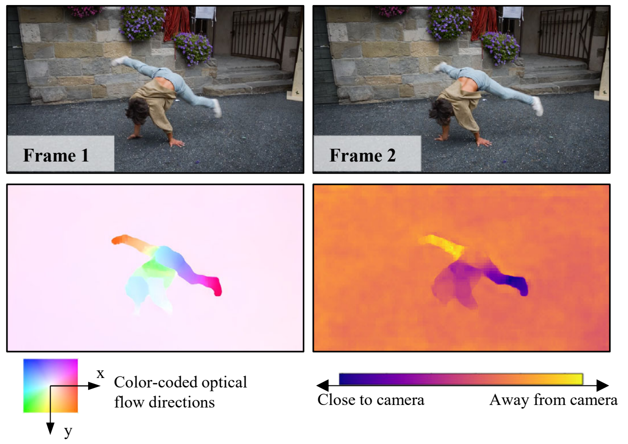

Fig. 1. **Zero-shot Generalization in Real Scenes.** Only needing to input two consecutive frames of images, ScaleFlow++ can robustly estimate dense 3D motion fields. The lower left corner of the image is the color-coded optical flow map, and the lower right corner is the motion-in-depth (MID). MID provides additional depth cues in the Z-axis direction, such as the dancer's left leg moving away from the camera and the right leg moving closer to the camera.

I. Introduction

THREE-DIMENSIONAL motion estimation aims to estimate the motion trend of pixels in a pair of images in 3D space. It has essential applications in video understanding [1], [2], autonomous driving [3]–[9], and autonomous robot navigation [10]. For example, in autonomous driving scenarios [11], [12], the car platform needs to obtain the 3D motion state of typical targets to make path decisions. Most existing platforms use Lidar to obtain 3D point clouds of scenes and construct corresponding relationships between two frames of 3D point clouds to estimate 3D motion. However, due to limited range and sparse results, this active depth perception scheme cannot effectively handle objects at a distance and abnormally reflective surfaces. In addition, the expensive price and maintenance costs further limit the broader application of Lidar. In this paper, we consider using a monocular camera to estimate 3D motion in dynamic scenes (commonly called normalized scene flow), which is a more stable and easy-to-maintain sensor that is not limited by limited range and sparse results.

Challenge: Normalized scene flow (NSF) consists of optical flow and motion-in-depth. Monocular cameras can robustly estimate 2D optical flow and normalized camera motion trajectories by constructing pixel-by-pixel feature associations between two frames [13], [14]. However, monocular cameras cannot provide real scene depth [15]. In dynamic scenes, 2D optical flow is a function of scene speed, camera motion trajectory, and scene depth. Given only the optical flow and camera trajectory, solving these three parts is an under-constrained problem.

Previous work: In early time-to-collision (TTC) work [16]–[18], people found that there is a close correlation between the scale change of the object and the depth motion. They estimated the scale change of a specific image block through algorithms such as SIFT [19], and converted the scale change into sparse depth motion through the transformer formula. Recent works [20], [21] regress dense scale changes from optical flow fields and RGB images through deep networks. However, their over-reliance on regression leads to lower accuracy and unstable results. Moreover, they separate the calculation of optical flow and motion-in-depth, requiring the network to be trained in stages rather than end-to-end training.

Han Ling and Quansen Sun are with the School of Nanjing University of Science and Technology, Nanjing, Jiangsu; Yinghui Sun is with the School of Southeast University; Yuhui Zheng is with the Nanjing University of Information Science and Technology. (Corresponding authors: Quansen Sun) E-mail: 321106010190@njust.edu.cn, sunyh@seu.edu.cn, sunquansen@njust.edu.cn, zhengyh@vip.126.com

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

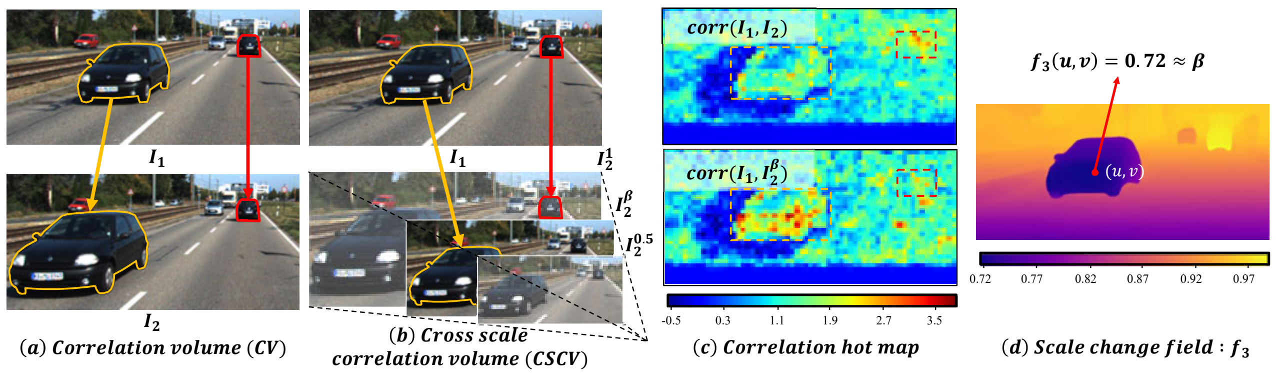

Fig. 2. Cross-scale matching idea. We match cars between two consecutive frames I_1 and I_2, where the yellow car is close to the camera and the red car is relatively static to the camera. (a) The CV module in the optical flow baseline [22] matches cars at the same scale, while the yellow car cannot match well because of the scale change. (b) CSCV matches objects in the 3D scale space, so that each object can achieve the perfect matching of position and scale simultaneously. (c) We visualize the correlation hot map sampled from the CV module. $corr(I_1, I_2^\beta)$ means that CV is built based on I_1 and I_2, where the meaning of each pixel point (x, y) is the correlation between the poin $I_1(x, y)$ and its corresponding point $I_2^\beta(\beta(x + \boldsymbol{f}(x, y)), \beta(y + \boldsymbol{f}(x, y)))$, $\boldsymbol{f}$ is the ground truth optical flow. **The correlation of the scale-changed yellow car in $corr(I_1, I_2)$ is smaller than that of the scale-invariant red car, while the correlation of the yellow car in $corr(I_1, I_2^\beta)$ is higher, which proves that the scale change has an important impact on the stability of optical flow matching**. (d) Dense scale change field f_3 estimated from our CSCV, the value of each pixel represents the scale change ratio.

In addition to the regression dependency dilemma faced by motion-in-depth estimation, existing optical flow pipelines are also plagued by scale issue. Most optical flow methods [22]–[26] rely on feature-matching techniques to associate objects in two frames by comparing the correlations between features. However, previous studies have pointed out that the commonly used CNN feature extractors only satisfy translation invariance and do not satisfy scale invariance [27]. When an object moves along the Z-axis (depth direction) of the camera, its scale change often leads to feature-matching failure. We demonstrate the significant impact of scale variation on feature matching in Fig. 2(c).

Our method: Fortunately, previous work has already explored many issues related to scale. As early as 1998, Lin et al. [28]. proposed a scheme based on the Laplace operator to calculate the maximum response position in the scale pyramid to solve the problem of scale change in feature recognition. Even if the scale of the input image changes, the output response features are stable. Furthermore, in 2005, Mikolajczyk [29] successfully estimated the scale changes of the bell tower captured at different focal lengths using a similar approach. Therefore, inspired by the success of previous work, we propose a cross-scale correlation volume (CSCV) module. As shown in Fig. 2, CSCV extends the original optical flow matching from the 2D-pixel plane to the scale pyramid, matching the correct features at the correct position and scale, improving the previous dilemma of estimating MID overly dependent on regression.

We propose an enhanced monocular 3D flow estimation network ScaleFlow++ based on the successful optical flow framework RAFT [22]. Its core feature is to replace the original 4D correlation volume with CSCV, and estimate both optical flow and MID based on feature matching in a unified architecture. To enhance the information aggregation capability of the network, we propose a lightweight global iterative optimizer (GIR). In previous studies, most RAFT-style methods used the GRU optimizer. However, this simple single-scale optimizer can only provide a local view, which makes the optimizer overly focused on the correlation features obtained from local sampling and unable to coordinate global motion information over a larger scale range. On the contrary, GIR referred to Unet's [30] successful experience and constructed an hourglass-shaped multi-scale residual convolutional network. In the encoding stage, it used the ConvNextV2 [31] module with a super large convolution kernel and employed pyramid pooling [32] at the minimum scale to coordinate global information. The results indicate that using the GIR optimizer can significantly improve background performance and slightly enhance foreground performance while keeping the total time almost constant.

In addition to GIR, we also introduced a global initialization module, which is mainly due to the task alienation observed in RAFT-style methods at different iteration periods. In the early stages of iteration, the optimizer starts optimizing from zero, which requires the optimizer to make aggressive estimates. At the end of the iteration, the optimizer focuses more on fine-tuning. The difficulty and focus of tasks vary significantly at different stages of iteration. Therefore, in order to alleviate the training burden of task alienation on the optimizer, we used an additional optimizer to quickly initialize the entire motion field at a scale of 1/16, to alleviate the alienation problem in subsequent iterations and reduce the total number of iterations.

During the training phase, we improved the commonly used truth supervision mode and proposed an enhanced training pipeline that combines self-supervised mixed truth. Specifically, we generate a flying foreground with random textures and shapes, iteratively overlay it on the original image pairs, and calculate the 3D stream truth of the flying foreground as a self-supervised training label. Previous works [22] often used solid color blocks to cover the source image in order to actively create occlusion. However, this behavior encourages the network to overly rely on regressing to estimate optical flow, resulting in poor generalization ability. Our hybrid training pipeline actively creates occlusions while forcing the network to learn 3D flow of randomly moving foreground through texture matching, ultimately balancing occlusion prediction and texture matching learning.

The significant performance improvement validates our motivation. In the KITTI [33] scene flow test set, ScaleFlow++ surpassed all NSF methods, reducing the core metric SF-all from 6.21% to 5.79%. In MID estimation, the minimum MID error was reduced from 42.84 to 38.44. In the Sintel [34] optical flow test set, ScaleFlow++ outperforms most similar methods with highly competitive performance. Moreover, as shown in Fig. 1, ScaleFlow++ also has strong generalization ability and robustness. In Sec. IV-G, we demonstrated the outstanding performance of ScaleFlow++ in unseen scenarios.

Differences from the preliminary version: This work is a substantial extension of our previous paper, Scale-flow [35], presented at the ACM MM 2022 conference. The preliminary work proposes a monocular 3D flow estimation method based on cross-scale matching, which alleviates the previous dilemma of relying solely on regression to calculate MID. This work comprehensively refactored the previous method while retaining the core module CSCV and made substantial improvements in multiple aspects, such as sports field initialization, iterative optimizer, and training methods, reducing the core indicator MID error from 48.9 to 38.44. The improved version significantly improved performance on various test benchmarks. The new contributions can be summarized as follows: (1) We introduce a self-supervised and truth mixed training pipeline that helps the network better balance the learning between occlusion fitting and texture matching, alleviate potential overfitting problems during training, and better predict motion prospects. (2) In order to solve the problem of standard GRU iteration modules having difficulty perceiving global motion, we propose an iterative optimizer GIR with a global perspective, which significantly improves the background performance of the network and slightly enhances the foreground performance while keeping the total time unchanged. (3) To alleviate the task alienation problem of the optimizer at different iteration stages, we constructed an initialization module to initialize the 3D flow field quickly. Reducing the total number of iterations while improving the optimizer's optimization capability. (4) We further demonstrated the superiority of our method in the field of optical flow on the Sintel dataset. (5) Based on the self-supervised generalization method ADF, we provide a real-world generalization model for readers to test. Our code and test model can be obtained on https://github.com/HanLingsgjk/CSCV.

II. Related Works

In this section, we first introduce the optical flow task and scene flow task closely related to 3D flow, and then present the motion-in-depth estimation and cross-scale matching technique, which is the primary source of inspiration for the CSCV module.

A. Optical Flow

Early methods [36], [37] viewed optical flow as an optimization problem for minimizing energy. Based on the regularization term and the constant brightness assumption, the optical flow results are optimized iteratively. Some recent excellent works have inherited their ideas [22], [38], [39], gradually updating the optical flow field through correlation matching and warping techniques. Compared with the previous methods [23], [40]–[44] of directly using convolutional networks to regress optical flow, the method based on correlation matching has a clearer theoretical background and more robust results. In this work, inspired by the successful experience of correlation matching, we extended this matching pattern to the scale dimension based on CSCV, improving the scale robustness of the optical flow baseline model.

It is worth mentioning that although some previous optical flow works [22], [45], [46] also used the idea of scale, our method differs fundamentally from theirs. Previous works are often based on multi-scale technology to increase the network receptive field, improve the inference speed, and strengthen the loss function. However, it is still image-to-image matching in essence. CSCV introduces a pattern for optical flow matching in scale space, a brand-new attempt.

B. Scene Flow

Scene flow aims to estimate the motion of pixels in 3D space, which is the main component of the 3D flow problem.

Methods Based on Rigid Assumption. Early methods [47], [48] decompose the image into rigid blocks, and iteratively optimize the 3D motion of the rigid blocks based on the rigid assumption and regularization terms. In recent scene flow methods [12], [39], [49] based on convolutional networks, a semantic network is often introduced to judge whether pixels belong to the same rigid object, and the 3D flow is optimized again according to the semantic results. Although it performs well on datasets, such methods are difficult to promote and apply in field scenarios due to the need for depth information and additional semantic labels. In addition, sub-modules such as 3D flow and semantic estimation are independent. Any defects in sub-module will affect overall performance, as the entire pipeline depends on its results.

Methods Based on Point Cloud. Point cloud based methods [50]–[53] capture 3D point clouds of the surrounding environment through an active laser or stereo camera, and establish the corresponding relationship between two frame point clouds to estimate 3D motion. However, this kind of method is limited by the sparse results and limited range, which is challenging to apply in more common video scenes. There is also a point cloud method based on the fusion framework, Camliflow [12]. By highly fusing the point cloud and the optical flow pipeline, the two pipelines can obtain complementary results at the same time. Camliflow achieves end-to-end scene flow estimation, but it still has the problem of depth information dependence.

C. Motion-in-depth (MID)

MID describes the motion in the depth direction through the scale change of depth. It has many practical application scenarios, such as time-to-collision (TTC) estimation, LiDAR scene flow, scene flow, Etc. MID is also the core indicator of our ScaleFlow++.

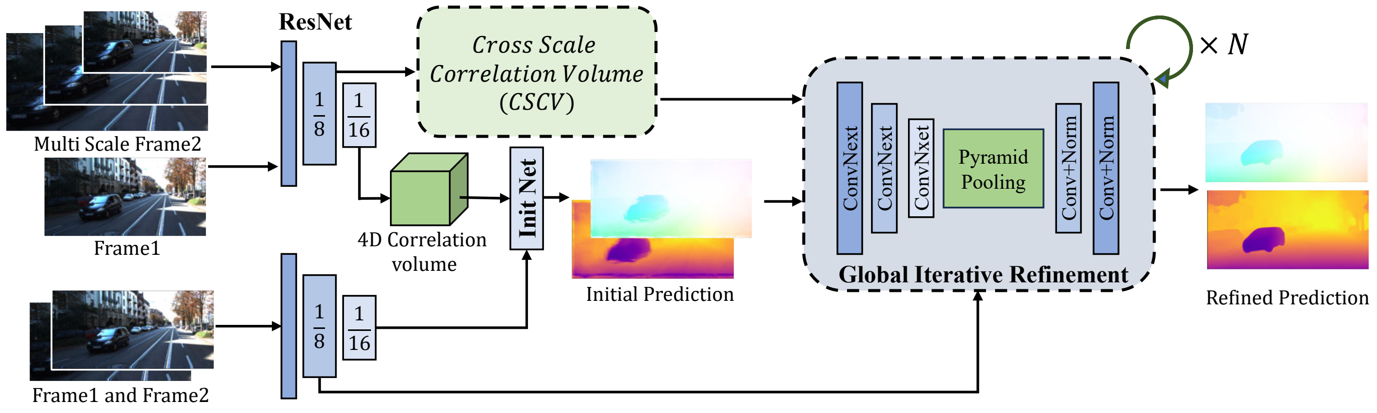

Fig. 3. Overview of ScaleFlow++. The network is mainly divided into two stages: initialization and iterative optimization. In the initialization stage, we construct a 4D correlation volume based on the 1/16 features extracted from ResNet and sample it using an all-zero optical flow field. Based on this, we regress the initialized 3D motion field. In the iterative optimization phase, we sample cross-scale correlation features from CSCV based on the current motion field, encode them, and send them to the GIR module for optimization. Output the final optimization result after N iterations.

Early TTC works [16]–[18] estimated the MID by tracking the motion trajectory of the interest points and building a motion model. Because the distribution of the interest points is random, this kind of method often makes it hard to pay attention to the primary target, yielding only sparse and low-precision results. Binary TTC [21] is a successful TTC scheme proposed recently, which can regress a dense MID field segmentally. Binary TTC sends the scaled image to the encoder to directly return the MID result of a certain range, and the specific range depends on the scaling factor. Therefore, a complete MID estimation requires multiple regressions to cover the complete MID interval. Unlike them, ScaleFlow++ only needs one calculation to get complete and dense MID results.

Yang & Ramanan [20] proposed an other monocular normalized scene flow method. They obtained dense MID results based on optical flow step-wise regression optical expansion and MID. However, the same as previous TTC work, they separated optical flow and MID estimation, resulting in the depth motion estimation result dependent on the optical flow result. Unlike them, ScaleFlow++ integrates optical flow and depth motion estimation into a harmonious task based on CSCV, making information blend and help each other. In addition, the cross-scale matching mechanism can also help the optical flow module more accurately estimate the objects with scale changes.

D. Cross-scale Matching

Cross-scale matching aims to establish the correspondence between objects in different frames, and the scales of objects in these frames often have large changes, which is a difficult problem faced by current optical flow methods. In the early works, SIFT [19] obtains scale-invariant features in the scale space through hand-designed feature extraction operators, and then matches objects of different scales based on these stable features. In addition to being used for matching, cross-scale matching technology can also be used to estimate the scale changes of objects between frames. As early as 2005, Mikolajczyk [29] used Lindeberg's scale selection method [28] to successfully estimate the optical expansion of clock towers captured at different focal lengths. Inspired by the above works, CSCV performs optical flow matching in scale space and estimates a dense scale change field based on the cross-scale matching results.

III. Method

The framework of ScaleFlow++ is shown in Fig. 3, similar to RAFT, which constructs a Cross-Scale Correlation Volume (CSCV) on all pixel pairs and iteratively optimizes the 3D motion field through an GIR optimizer. We also constructed a small 4D correlation volume based on features of 1/16 size for rapid initialization of optical flow and scale flow, thereby reducing the optimization difficulty of the GIR. In the inference process, GIR **directly outputs the optical flow** (f_1, f_2) **and scale change** f_3 **fields for CSCV sampling**. Combining previous work [20], we convert f_3 into motion-in-depth, which is convenient for training and can be more widely applied to downstream tasks.

Below, we will provide a detailed introduction to the construction and training process of ScaleFlow++, as well as how to convert network estimation results into a 3D motion field.

A. Feature Extraction

We replaced the residual network in RAFT with ResNet18 [54], which has the advantage of being able to use pre-trained weights from MAE [55] or ImageNet [56]. The results indicate that this can improve training efficiency and introduce additional prior knowledge to improve generalization performance. Specifically, we constructed a feature encoder and a context encoder, respectively:

Feature Encoder. For the feature encoder g_θ, we take its truncated outputs at 1/8 and 1/16 scales, where $g^1_\theta : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{(H/8)\times(W/8)\times D}$ and $g^2_\theta : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{(H/16)\times(W/16)\times D} (D = 256)$.

Context Encoder. Following RAFT, we build a context encoder h_θ, whose input is the superposition of frames I_1 and I_2, and the rest of the structure is the same as g_θ. The output

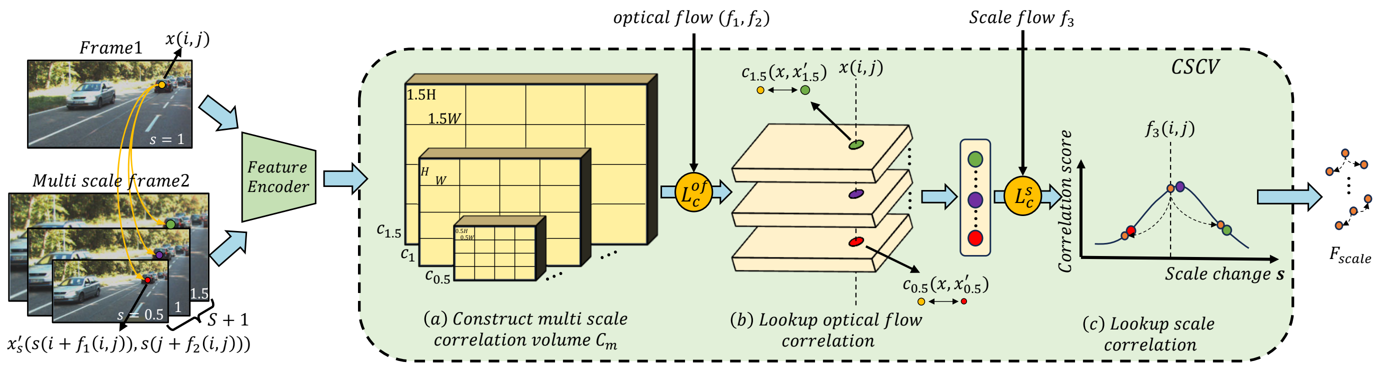

Fig. 4. The complete structure of cross-scale correlation volume (CSCV). The CSCV module is composed of three parts: multi scale correlation volume C_m, optical flow lookup operator L_c^{of}, and scale lookup operator L_c^s. After the optical flow field (f_1, f_2) and scale change field f_3 are input, the optical flow feature F_{of} and multi-scale optical flow feature F_{multi} are first sampled from C_m by the L_c^{of} operator, while the scale feature F_{scale} is obtained from F_{multi} by the L_c^s operator.

of the 1/16 scale $F_{init}(D = 384)$ is directly used to initialize, while the output of the 1/8 scale is divided into two parts: one is the context feature $F_{context}(D = 192)$ and the other is the implicit feature $h_0(D = 192)$ used for GIR iteration.

B. Initialization of Optical Flow and Scale Flow

RAFT-type methods usually initialize the flow field to 0 and then refine it iteratively through a GRU module. However, we found that this leads to two problems. Firstly, GRU iterative networks must simultaneously complete tasks such as large field optimization (early iteration) and fine-tuning (late iteration), significantly increasing the difficulty of training the iterative network. Secondly, due to limitations in memory and cost, it is not easy to increase the sampling range of the correlation volume during iteration in order to obtain a larger receptive field.

In ScaleFlow++, we drew on the successful experiences of IGEV [57] and SEA-RAFT [58] and used an independent lightweight network to quickly initialize the 3D flow, reducing the number of iterations of the optimizer and alleviating task differences at different iteration stages. Specifically, the input for initializing the network is the contextual feature F_{init} and the correlation feature F_{of}^{init}. We first use the GIR module proposed in Sec. III-D to encode the above features:

$$h_{init} = \mathbf{GIR}(F_{of}^{init}, F_{init}) \tag{1}$$

where $F_{of}^{init} \in \mathbb{R}^{H\times W\times(2r+1)\times(2r+1)}$ is sampled from a 1/16 scale 4D correlation volume, the sampling optical flow field is all 0, and the sampling radius r is 6.

Next, we upsample the hi to 1/8 scale and use a simple two-layer convolution head to predict the initialized optical flow field (f_1^0, f_2^0) and scale change field f_3^0:

$$\begin{aligned}(f_1^0, f_2^0) &= \text{Conv}(\text{ReLU}(\text{Conv}(\text{Upsample}(h_{init})))) \\ f_3^0 &= \text{Conv}(\text{ReLU}(\text{Conv}(\text{Upsample}(h_{init}))))\end{aligned} \tag{2}$$

C. Cross-scale Correlation Volume (CSCV)

In this part, we introduce the construction of the cross-scale correlation volume (CSCV) module. As shown in Fig.4 , firstly, the correlation volume C_m is constructed from the dot multiplication of input multi-scale features. Then the multi-scale optical flow correlation features F_{multi} and scale correlation features F_{scale} are extracted from C_m successively using the optical flow lookup operator L_c^{of} and scale lookup operator L_c^s.

1) The Input of CSCV: The main function of CSCV is to sample the correlation features of the current input 3D flow field. Specifically, the input 3D flow field is composed of an optical flow field (f_1, f_2) and a scale change field f_3. The optical flow field describes the displacement of pixels between two frames, and the scale change field is the reciprocal of optical expansion (OE), which describes the pixel scaling factor of the second frame for perfect matching the first frame.

When constructing C_m, multi-scale image features are extracted by a residual convolutional network g_θ, where the input of the network is I_1 and P_{I_2}:

$$P_{I_2} = \{I_2^s \mid s = 0.5, 0.5 + 1/S, ..., 1, ..., 1.5\} \tag{3}$$

where $I_2^s \in \mathbb{R}^{(8\times H\times s)\times(8\times W\times s)\times 3}$, S specifies the degree of refinement of the input scale, which is generally set to 4 in our experiment. Because subsequent iterations were conducted on the 1/8 scale, the width and height were multiplied by a coefficient of 8 to facilitate the expression of following functions.

2) Multi Scale Correlation Volume: The multi scale correlation volume C_m is a collection of multiple 4D correlation matrices. The construction of the 4D matrix follows RAFT, which can be sampled by the coordinates of a pair of matching points. The sample result is the correlation of these two points. Specifically, for the given image features $g_\theta^1(I_2^s) \in \mathbb{R}^{(H\times s)\times(W\times s)\times D}$ and $g_\theta^1(I_1) \in \mathbb{R}^{H\times W\times D}$, dot them to obtain a single 4D correlation matrix c_s:

$$\begin{gathered}c_s\left(g_\theta^1\left(I_2^s\right), g_\theta^1\left(I_1\right)\right) \in \mathbb{R}^{H\times W\times(H\times s)\times(W\times s)} \\ (c_s)_{ijkl} = \sum_h g_\theta^1\left(I_2^s\right)_{ijh} \cdot g_\theta^1\left(I_1\right)_{klh}\end{gathered} \tag{4}$$

where (i, j), (k, l) are the coordinates of two pairs of points on I_2^s and I_1 respectively, h is the dimension index of the feature, and c_s is the 4D correlation volume between I_1 and I_2 with scale s.

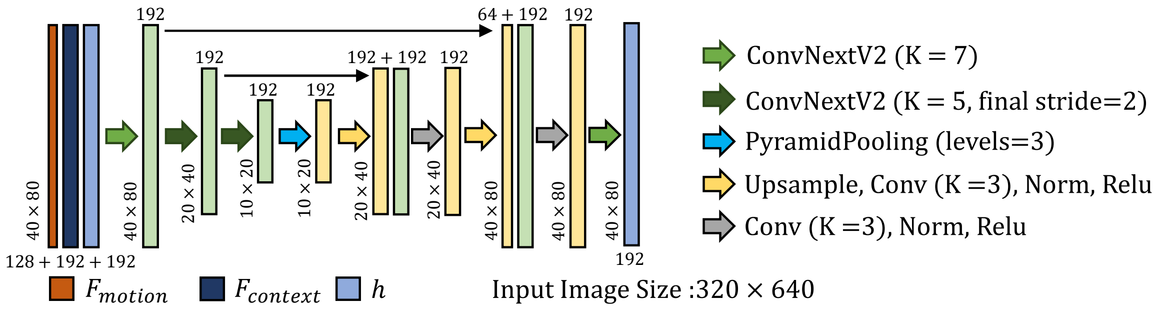
ConvNextV2 (K = 7)
ConvNextV2 (K = 5, final stride=2)
PyramidPooling (levels=3)
Upsample, Conv (K =3), Norm, Relu
Conv (K =3), Norm, Relu
F_{motion}
$F_{context}$
h
Input Image Size :320 × 640
128 + 192 + 192
64 + 192
192 + 192

Fig. 5. Global Iterative Refinement Module (GIR). Similar to Unet, GIR is mainly composed of funnel-shaped encoders and decoders. Specifically, GIR uses ConNextV2 with large kernel convolution characteristics at multiple scales, where a single convolution kernel can cover almost the entire frame. Combined with the pyramid pooling module at the smallest scale, it greatly enhances global perception capability.

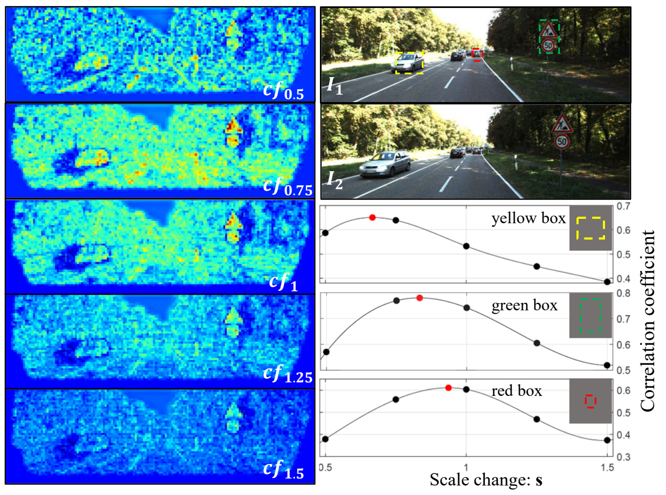

Fig. 6. Visualization of the scale matching by CSCV. On the left is the multi-scale optical flow feature F_{multi} when $S = 4$ and $r_f = 0$. The redder the color of the point, the higher the correlation. On the right is the correlation curve, where the value of each black point is the mean value of the points in the box on the left, and the red point is the extreme value obtained by interpolation. The extreme points of the cars in the yellow box and the signs in the green box tend to be 0.5 and 0.75, which means they are moving toward the camera rapidly. The extreme point of the red box tends to 1, which means that the object is relatively stationary with the camera.

The final C_m is a set about c_s:

$$C_m = \{c_s \mid s = 0.5, 0.5 + 1/S, ..., 1, ..., 1.5\} \quad (5)$$

3) Optical Flow Lookup Operator: We define the optical flow lookup operator L_c^{of} to sample the correlation features corresponding to the current optical flow field (f_1, f_2) from C_m.

Given an optical flow field (f_1, f_2), it maps each pixel $x = (k, l)$ in I_1 to $x' = s(k + f_1(k, l), l + f_2(k, l)) = (i, j)$ in I_2^s, L_c^{of} samples in the neighborhood $N_{x'}^f$ of x':

$$N_{x'}^f = \{ \ (i + d_u, j + d_v) \mid d_u, d_v \in \{ \ -r_f, ..., r_f\} \ \} \quad (6)$$

where r_f is an integer, it specifies the size of the sampled neighborhood, and neighborhood sampling provides the optimal direction for the network.

Based on the sampling coordinates provided by $N_{x'}^f$, we conducted bilinear interpolation sampling for each c_s in C_m to get the collection of multi-scale optical flow correlation feature F_{of}:

$$F_{of} = \{cf_s \mid s = 0.5, 0.5 + 1/S, ..., 1, ..., 1.5\} \quad (7)$$

where $cf_s \in \mathbb{R}^{H \times W \times (2r_f+1) \times (2r_f+1)}$. In Fig. 6, we demonstrate the working principle of F_{of} through a simplified cross-scale matching example.

Finally, we spliced all the elements in F_{of} to form a 5D correlation matrix $F_{multi} \in \mathbb{R}^{H \times W \times (2r_f+1) \times (2r_f+1) \times (S+1)}$, The newly added fifth dimension is the scale dimension, which can be used to sample the correlation between the I_2 features of different scales and the I_1 features of the original scale.

4) Scale Lookup Operator.: We define the scale lookup operator L_c^s to sample the correlation features corresponding to the current scale change field f_3 from F_{multi}.

Given a scale change field f_3, it describes the scale change of a pixel block between two frames. The scale change field f_3 maps each pixel $x = (k, l)$ in I_1 to $x' = f_3(k, l) \times (k + f_1(k, l), l + f_2(k, l)) = (i, j)$ in $I_2^{f_3(k,l)}$, L_c^s sample the neighborhood $N_{x'}^s$ in the scale direction of point x':

$$N_{x'}^s = \{ \ f_3(k, l) + d_s \mid d_s \in \{ \ -r_s, ..., r_s\} \ \} \quad (8)$$

where r_s is a real number, we set $r_s = 1/S$ in our experiments, $N_{x'}^s$ provides optimal awareness in the scale direction.

Based on the sampling coordinates provided by $N_{x'}^s$, we sample in F_{multi} based on bilinear interpolation, and splice the results into scale correlation feature $F_{scale} \in \mathbb{R}^{H \times W \times (2r_f+1) \times (2r_f+1) \times (2r_s+1)}$. Moreover, we also follow the RAFT strategy to extract the optical flow correlation feature $F_{of} \in \mathbb{R}^{H \times W \times (2r_f+1) \times (2r_f+1) \times 4}$, which is sampled from c_1 (2×2 pooling for three times) by L_c^{of}.

D. Global Iterative Refinement

1) Optimizer: In most RAFT-type methods, the optimizer's field of view is often limited to the size of the search radius in the correlation volume, and can not obtain a global field of view. To overcome this problem, we propose the Global Iterative Refinement Module (GIR), which has a detailed structure as shown in Fig.5. Similar to Unet, GIR consists of funnel-shaped encoders and decoders, and the encoding stage mainly uses ConvNextV2 with a large field of view. Unlike the commonly used GRU iteration module, multi-scale large kernel convolution and pyramid pooling module can ensure that GIR obtains sufficient global perception capability.

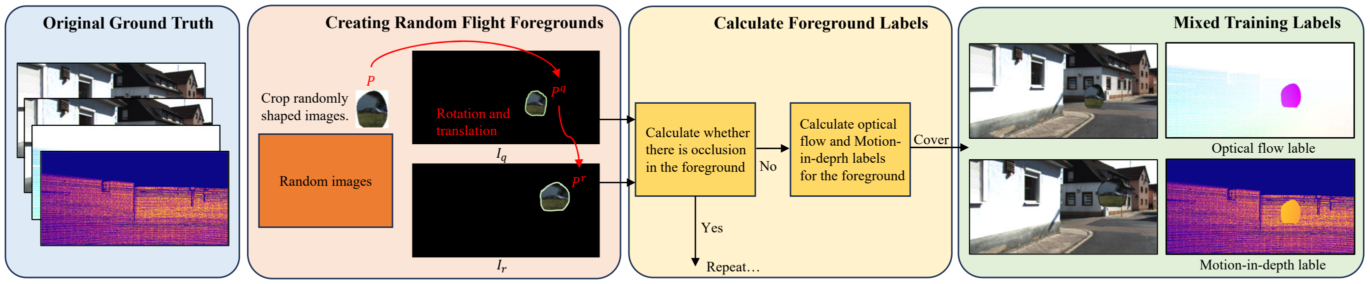
Original Ground Truth
Creating Random Flight Foregrounds
Crop randomly shaped images.
P
P^q
Rotation and translation
Random images
I_q
P^r
I_r
Calculate Foreground Labels
Calculate whether there is occlusion in the foreground
No
Calculate optical flow and Motion-in-deprh labels for the foreground
Yes
Repeat…
Cover
Mixed Training Labels
Optical flow lable
Motion-in-depth lable

Fig. 7. Hybrid training data generation pipeline. Our pipeline is divided into three parts. Firstly, we crop randomly shaped foreground image blocks from random images and perform spatial rotation and translation transformations between two frames. Afterwards, it is determined whether the image block has been occluded (flying out of the image range). If no occlusion has occurred, the corresponding optical flow truth value and depth change truth value are calculated. Finally, iteratively add the generated flight foreground to the original image and update the related truth labels at the same time.

2) Initial State: The initial states of the optical flow field (f_1^0, f_2^0) and scale change field f_3^0 are given by the initialization network in Sec. III-B, and the initial hidden layer features h_0 is given by the context encoder.

3) Updating: We iteratively optimize the 3D flow field based on residual update $f^{k+1} = f^k + \Delta f$. Before starting the update, we first need to encode the correlation features F_{scale}, F_{of}, as well as the optical flow field (f_1, f_2) and scale change field f_3, into motion features F_{motion}:

$$F_{motion} = \textbf{MotionEncoder}(F_{scale}, F_{of}, f_1, f_2, f_3) \tag{9}$$

where **MotionEncoder** is an encoder composed of a simple two-layer convolution, $F_{motion} \in \mathbb{R}^{H\times W\times 128}$.

Next, as shown in Fig. 5, use the GIR module to iteratively update the hidden features:

$$h_t = \textbf{GIR}(F_{motion}, F_{context}, h_{t-1}) \tag{10}$$

Finally, the residual $\Delta f = (\Delta f_1, \Delta f_2, \Delta f_3)$ for updating is estimated from h_t by a simple two-level convolution head:

$$\begin{aligned} (\Delta f_1, \Delta f_2) &= \text{Conv}(\text{ReLU}(\text{Conv}(h_t))) \\ \Delta f_3 &= \tanh(\text{Conv}(\text{ReLU}(\text{Conv}(h_t)))) \end{aligned} \tag{11}$$

Considering that the scale flow requires more detailed estimation, we performed an additional update on f_3 individually using the GIR network at the end.

E. Hybrid Training Pipeline

We use a combination of self-supervised and ground truth methods to train our model. In this section, we focus on introducing the self-supervised data generation pipeline. As shown in Fig. 7, details are as follows.

Creating Random Flight Foregrounds. Firstly, based on the randomly generated closed curve (generated by the Bessel curve), crop the corresponding image block p on a random image, and then project it to the P in the camera coordinate system:

$$p(i) = (x, y, 1) \tag{12}$$

$$P(i) = Z\boldsymbol{K}^{-1}p(i) = (X, Y, Z) \tag{13}$$

where i is the point contained in p, $\boldsymbol{K}$ is the camera intrinsic matrix, and the depth Z of all pixels in P is 1.

After obtaining P, we first perform the initial spatial transformation to obtain P^q, and then project it onto the pixel coordinate system to get the p^q on image I_q:

$$P^q(i) = \boldsymbol{R}_r P(i) + \boldsymbol{T}_r \tag{14}$$

$$p_q(i) = \boldsymbol{K}P^q(i) \tag{15}$$

among them, $\boldsymbol{R}_r$ and $\boldsymbol{T}_r$ are the rotation matrix and translation matrix of random values. Similarly, we perform the spatial transformation on P^q to obtain P^r and project it onto the pixel plane to get p^r on image I_r.

Note that the captured image blocks here are random textures, as we found that using foreground flying objects with clear semantic information (such as complete car, human, and animal contours) does not perform as well as random textures. Specifically, when occlusion occurs, the network becomes overly confident in inferring the motion of the occluded part of the object as planar motion. Random textures can encourage networks to calculate the motion of these flying foreground based on texture matching, and balance the learning weight with occlusion regression.

Calculate Foreground Labels. Before calculating the label for the flight foreground, it is necessary first to determine whether there is an occlusion in the foreground. Because the flight foreground is primarily a random texture, the network cannot make reasonable inferences about the occluded parts. We propose a difference rate Df indicator to measure whether the foreground is retained:

$$Df = \frac{|N_r - N_q|}{|N_r + N_q|} \tag{16}$$

where N_q and N_r represent the number of foreground flying object pixels in the first and second frames, respectively. In the experiment, we only used flight prospects with Df less than 0.5 to ensure no significant obstructions or great scale changes during the flight.

After filtering, calculate the accurate value of the flight foreground optical flow and the true value of the depth change rate as follows:

$$\boldsymbol{f_{fg}} = p_r - p_q \tag{17}$$

$$\boldsymbol{\tau_{fg}} = Z^r / Z^q \tag{18}$$

where $\boldsymbol{f_{fg}}$ is the optical flow of the flight foreground, and $\boldsymbol{\tau_{fg}}$ is the corresponding motion-in-depth.

TABLE I
TRAINING DETAILS. *Pre* REPRESENTS THE PRE-TRAINING STAGE, AND *Ft* REPRESENTS THE FINE-TUNING STAGE. THE HYBRID TRAINING PIPELINE IS USED BY DEFAULT DURING TRAINING, AND THE NUMBER OF FLYING FOREGROUND ADDITIONS IS 1. HYBRID REPRESENTS THE NUMBER OF FOREGROUND IN TRAINING

Section	Dataset	Stage	Batch	Refine Iteration	Train Iteration	Size	LR	Hybrid
Ablation (IV-B)	D+ S-18		4	6	10k	320x720	0.000125	1
Motion-in-depth (IV-C)	D+T+S+M+K-160	*Pre*	4	6	30k	320x720	0.00025	1
	K-160	*Ft*	6	6	5k	320x896	0.00005	2
Sintel-test (IV-D)	D+T+S+M+K15	*Pre*	4	6	30k	320x720	0.00025	1
	T+S+M	*Ft*	6	6	30k	320x896	0.00005	2
Kitti-test (IV-E)	D+T+S+M+K15	*Pre*	4	6	30k	320x720	0.00025	1
	K15	*Ft*	6	6	5k	320x896	0.00005	2
Generalization (IV-G)	ADFactory [59]		4	6	30k	320x768	0.00025	1

Optical Flow Loss. We supervise the optical flow results with L_1 distance between prediction and ground truth, the optical flow loss $\mathcal{L}_f$ is defined as:

$$\mathcal{L}_f = \sum_{k=1}^{N} \gamma^{N-k} (\left\|f_1^k - f_{1gt}\right\|_1 + \left\|f_2^k - f_{2gt}\right\|_1) \tag{19}$$

we set $\gamma = 0.8$, $N = 6$ in our experiments, the $\boldsymbol{f_{gt}} = (f_{1gt}, f_{2gt})$ is composed of a hybrid of truth labels from the dataset and the generated $\boldsymbol{f_{fg}}$ labels.

Scale Change Field Loss. According to Eq (22), the Scale change field f_3 equal to MID $\boldsymbol{\tau}$ under the slight rotation assumption, which allows us to use the scene flow datasets to train the f_3. The loss is defined as follows:

$$\mathcal{L}_s = \sum_{k=1}^{M} \gamma^{M-k} \left\|f_3^k - \boldsymbol{\tau}_{gt}\right\|_1 \tag{20}$$

where $\boldsymbol{\tau}_{gt}$ is also a hybrid of true values $\boldsymbol{\tau}$ and generated labels $\boldsymbol{\tau}_{fg}$, $\boldsymbol{\tau} = Z'_{gt}/Z_{gt}$, Z'_{gt} and Z_{gt} are the ground truth depths of matching pixels in the second frame and the first frame, we set $\gamma = 0.8$, $N = 7$ in our experiments.

The overall loss function is:

$$\mathcal{L} = \mathcal{L}_f + \mathcal{L}_s \tag{21}$$

F. Convert ScaleFlow++ output to 3D motion

Referring to previous work, we converted the 3D motion (f_1, f_2, f_3) into motion-in-depth and scene flow. Firstly, based on yang's derivation [20], we can obtain the following approximate equation:

$$f_3 \approx \frac{Z'}{Z} = \boldsymbol{\tau} \tag{22}$$

where, Z and Z' is the depth of the object in two frames, and $\boldsymbol{\tau}$ is the motion-in-depth.

Furthermore, based on Eq.22, we can derive the relationship between (f_1, f_2, f_3) and scene flow $\boldsymbol{t}$:

$$\boldsymbol{t} = Z\boldsymbol{K^{-1}}[(f_3 - 1)\boldsymbol{Cd} + f_3(f_1, f_2)] = Z\bar{\boldsymbol{t}} \tag{23}$$

where $\boldsymbol{Cd}$ is the image coordinate index matrix, normalize scene flow $\bar{\boldsymbol{t}} = \boldsymbol{K^{-1}}[(f_3(\boldsymbol{p}) - 1)\boldsymbol{p} + f_3(\boldsymbol{p})\boldsymbol{f}]$. We have presented a detailed derivation process in the **Supplementary Material**.

IV. EXPERIMENTS

In this section, we introduce the key details of the dataset and experiment, and then further discuss the experimental results. Specifically, the experiment includes the following aspects. Firstly, a comprehensive ablation was conducted on the proposed module. Secondly, test the performance of ScaleFlow++ on mainstream monocular 3D task MID. Thirdly, submit the results of ScaleFlow++ to public testing platform and compared with the most advanced scene flow and optical flow methods currently available. Finally, we qualitatively evaluate the generalization performance of ScaleFlow++ in unseen scenarios.

A. Datasets and Training Setting

We mainly compared the methods quantitatively on the KITTI [33] and Sintel [34] baselines, as they have publicly available online evaluation websites and have been widely applied in previous works.

Datasets Used: 200 image pairs from KITTI 2015 (K15) [33]. Commonly used pre-training datasets Flyingthings3D (T) [60], Driving (D) [60], Monkey (M) [60] and Sintel (S) [34] from movie.

Dataset Split: For the convenience of ablation experiments, we also divided the Sintel dataset into two subsets. One is the training set (S-18) containing 824 image pairs from 18 scenes, and the other is the testing set (S-5) containing 240 image pairs from 5 scenes. In addition, for the MID task, we use the same strategy as yang [20] to split the dataset, selecting one out of every five images in KITTI for evaluation (K-40) and the remaining 160 images for fine-tuning training (K-160).

Training Setting: Tab. I shows the specific training process, during which we also performed enhancement operations such as scaling, translation, and color domain transformation on the image pairs. Please refer to RAFT [22] for details.

B. Ablation

In this section, we discuss the effectiveness of the various components of the ScaleFlow++ method proposed in this paper. The complete ablation results are shown in Tab. II; we gradually stack the proposed modules based on the Base model.

TABLE II
ABLATION EXPERIMENT. RESNET* REPRESENTS THE USE OF IMAGENET WEIGHTS FOR INITIALIZATION. Mid IS THE MOTION-IN-DEPTH ERROR, AND THE CALCULATION METHOD IS DETAILED IN EQ. 24. EPE IS THE ENDPOINT ERROR OF OPTICAL FLOW, AND ALL IS THE OUTLIER RATE.

			S-5-clean		S-5-final		K15		
	Ablation	Iter	EPE ↓	ALL ↓	EPE ↓	ALL ↓	EPE ↓	ALL ↓	Mid ↓
A	Base	10k	2.70	11.43	3.23	15.01	4.89	18.95	167.86
B	Base+Ht	10k	2.46	10.78	3.17	14.88	4.63	17.57	153.98
C	Base+Ht+CSCV	10k	2.35	10.03	2.98	13.53	4.28	16.12	143.66
D	Base+Ht+CSCV+Resnet	10k	2.35	10.00	3.17	14.07	4.38	16.47	137.56
E	Base+Ht+CSCV+Resnet*	10k	2.11	8.42	2.81	11.69	4.96	17.13	175.84
F	Base+Ht+CSCV+Resnet*+RNN	10k	2.06	8.11	2.63	11.76	3.51	11.85	127.55
G	Base+Ht+CSCV+Resnet*+GIR	10k	1.87	7.59	2.53	10.51	3.60	11.91	115.03
H	Base+Ht+CSCV+Resnet*+GIR+Init	10k	1.82	7.09	2.29	9.77	3.17	10.31	114.78
		15k	1.72	6.66	2.23	9.24	3.23	10.20	112.73
I	Base+Ht+CSCV+Resnet+GIR+Init	10k	2.04	8.32	2.77	11.66	3.15	10.83	95.52
		15k	1.89	7.68	2.57	10.12	3.26	10.61	96.07

Base Model: We used the previous version Scale-flow as the ablation baseline and replaced the CSCV with the 4D Correlation Volume in RAFT. The rest remains unchanged, using the residual feature extractor and GRU iterative optimizer from RAFT.

Hybrid Training Pipeline (Ht): As shown in rows A and B of Tab. II, the mixed training pipeline Ht comprehensively improves the performance of all indicators. This proves that our proposed hybrid training pipeline can effectively enhance the scene flow performance of the network without increasing training costs.

Cross-Scale Matching (CSCV): As shown in rows B and C of Table II, the model's optical flow and scale flow performance have been significantly improved. Especially in the driving scenario KITTI with significant changes in scale, the error of optical flow decreased from 4.63 to 4.28, which proves the effectiveness of our cross-scale matching strategy.

ResNet and Weight Pre-training: In rows D and E of Table II, we replaced the original residual convolution with standard Resnet and used pre-trained weights from ImageNet. The experimental results show that the performance of Resnet is close to the accuracy of the RAFT original feature extractor, but after using pre-trained weights, the performance of the Sintel dataset is significantly improved, while the KITTI dataset is significantly reduced. This is easy to understand because there are almost no driving scenes in ImageNet, and its data domain is closer to Sintel. We also verified the relationship between training iteration and pre-training weights. As shown in H and I, the method using pre-training weights still maintained significant advantages when more iterations were being trained.

The above experiment proves that appropriate pre-training weights can effectively improve model performance in similar data domains, and this advantage will continue to be maintained with increasing iteration times. Therefore, we recommend using ImageNet weights only on the Sintel dataset in subsequent comparisons.

Iterative Optimizer: We compared the performance between the most commonly used GRU optimizer, RNN optimizer in SEA-RAFT [58], and the GIR optimizer proposed in this paper. As shown in rows E, F, and G in Table II, all optimizers have undergone 6 iterations, and GIR has significant advantages in Sintel scenarios and Mid metrics. Moreover, we also reported the running time of each optimizer in the complete method in Table III. The optimizer proposed in this paper did not significantly increase computational overhead under standard iteration numbers.

Initialize Displacement Field: As shown in row H of

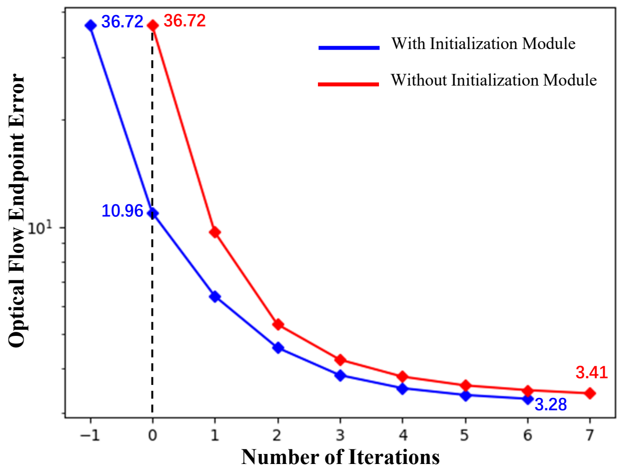

Fig. 8. Initialization Module. We selected the first 100 sequences of K15 to calculate the average optical flow endpoint error (EPE) for each iteration. The blue and red colors in the figure correspond to H and G in Table II, respectively. The difference is that the red has an additional GIR iteration to ensure fair comparison. When the optical flow field is all 0, the initial EPE error is 36.72, and the initialization module reduces the initial error to 10.96, reducing the subsequent optimizer's training difficulty and achieving higher accuracy with fewer iterations.

TABLE III
RUNTIME ANALYSIS OF DIFFERENT OPTIMIZERS. PERFORM 100 CONSECUTIVE INFERENCES ON A 375 × 1242 IMAGE AND TAKE THE AVERAGE. THE GPU USED IS RTX4090.

Optimizer	Single Iteration Time	Refine Iterations	Total Time
GIR (ours)	3.26ms	6	19.56ms
RNN	1.92ms	12	23.04ms
GRU	1.62ms	12	19.44ms

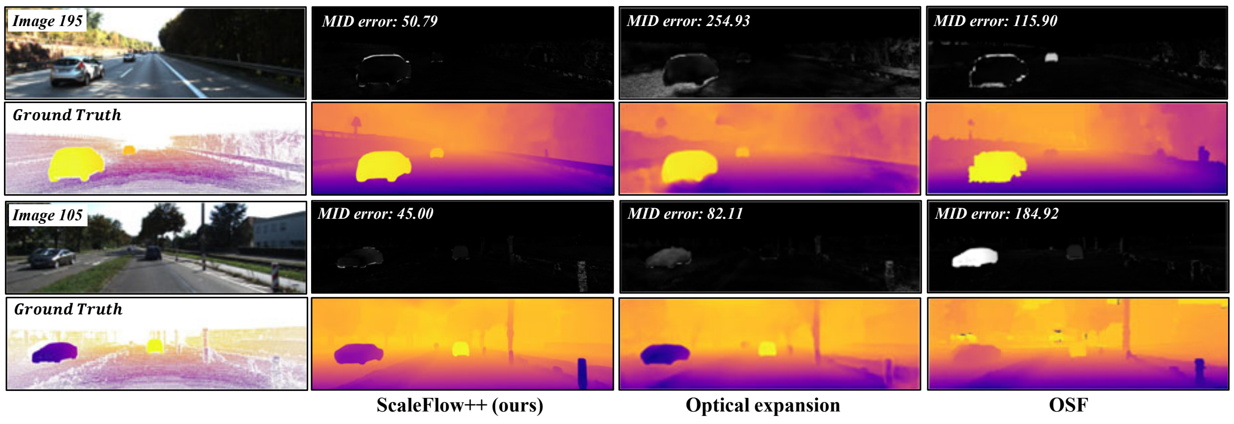

Fig. 9. Motion-in-depth results for images "105" and "195" in the K-40 set. For each of them, Up: input images and the error map, where the whiter the pixels, the greater the error. Down: ground truth and visualization of motion-in-depth. Our method is much more accurate than other methods.

Table II, the initialization module significantly improved the performance of optical flow and slightly enhanced Mid performance, proving the effectiveness of the initialization module.

We also demonstrated a more detailed working process of the initialization module. As shown in Fig. 8, the initialization module reduced the initial optical flow EPE from 36.72 to 10.96, alleviating the task alienation of the optimizer at different iteration stages, reducing the overall number of iterations, and improving accuracy.

TABLE IV
MOTION-IN-DEPTH ESTIMATION ON K-40. THE BEST AMONG ALL ARE BOLDED, AND THE SECOND BEST ARE UNDERLINED. OUR METHOD OUTPERFORMS THE MONOCULAR BASELINES BY A LARGE MARGIN.

Method	Input	Training set	$Mid\downarrow$	Time/s
OSF [47]	Stereo	K-160	115	3000
PRSM [48]	Stereo	K-160	124	300
Hur & Roth [61]	Mono	K-160	115.13	0.1
Binary TTC [21]	Mono	K-160	73.55	2.2
Optical expansion [20]	Mono	K-160	75	0.2
Scale-flow [35]	Mono	K-160	48.9	0.2
TPCV [62]	Mono	K-160	<u>42.84</u>	0.2
ScaleFlow++(ours)	Mono	K-160	**38.44**	0.2

C. Motion-in-depth (MID)

MID is the core indicator of monocular 3D flow methods. In order to prove the superiority of ScaleRAFT in estimating MID tasks, we compared ScaleFlow++ with the stereo-based traditional scene flow methods, state-of-the-art MID methods and scene flow methods on K-40. We use the same criteria as predecessors [20] to define the loss:

$$Mid = ||log(f_3) - log(\boldsymbol{\tau}_{gt})||_1 \cdot 10^4 \tag{24}$$

where $\boldsymbol{\tau}_{gt}$ is the ground truth of MID.

We first compare with the stereo-based traditional approach: OSF and PRSM decompose the pixels into rigid blocks, and iteratively update the optical flow and depth field based on rigidity assumptions and hand-designed regularization terms.To get the MID result, we divide the depth of the second frame by the depth of the first frame. The experimental results show that our error is much smaller (115 v.s. 38.44) and in less time.

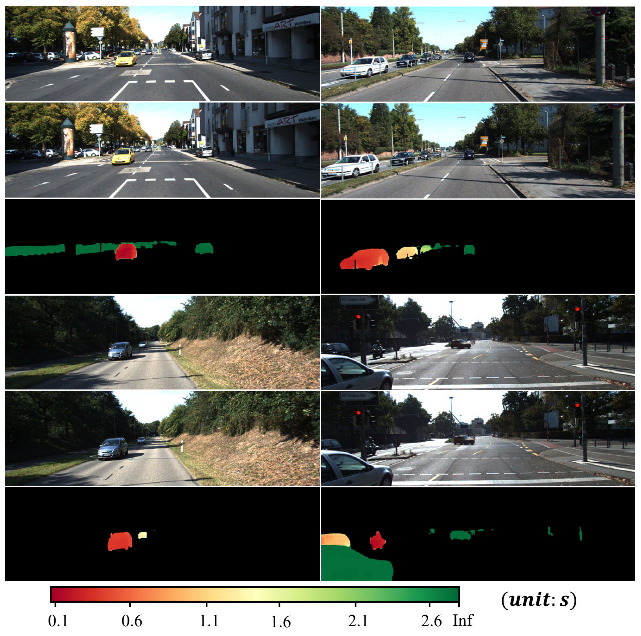

Fig. 10. Visualization of foreground target collision time. We visualized some of the TTC results in K-40; from top to bottom are two consecutive frames and the visualized collision time. The foreground mask is provided by the real-time segmentation algorithm Mseg [63].

Then we compare with the state-of-the-art MID methods Optical expansion, TPCV and Binary TTC. As shown in Table IV, ScaleFlow++ still has apparent advantages (42.84 v.s. 38.44).

Application example: We present an application example of MID in downstream 3D tasks: time-to-collision (TTC).

Time-to-collision estimation is a crucial technique in path planning for autonomous robots [2], [4], [10], [64], which describes the collision time between moving objects and the observer plane. TTC can be estimated indirectly by MID $\boldsymbol{\tau}$:

$$TTC = \frac{Z}{Z - Z'}T = \frac{T}{1 - \frac{Z'}{Z}} = \frac{T}{1 - \boldsymbol{\tau}} \tag{25}$$

where T is the sampling interval between two consecutive frames, for KITTI $T = 0.1s$, Z and Z' are the object's depth

TABLE V
STATE-OF-THE-ART PUBLISHED METHODS ON KITTI SCENE FLOW BENCHMARK. RGB-D MEANS DEPTH AND MONOCULAR INFORMATION, AND MONO MEANS MONOCULAR INFORMATION. D1, D2, FL, AND SF IS THE PERCENTAGE OF DISPARITY, OPTICAL FLOW AND SCENE FLOW OUTLIERS. -BG,-FG AND -ALL REPRESENT THE PERCENTAGE OF OUTLIERS AVERAGED ONLY OVER BACKGROUND REGIONS, FOREGROUND REGIONS AND OVERALL GROUND TRUTH PIXELS. THE BEST AMONG THE SAME GROUP ARE BOLDED, AND THE SECOND BEST ARE UNDERLINED. OUR MONOCULAR METHOD PERFORMS QUITE WELL, NOT ONLY FAR AHEAD OF SIMILAR MONOCULAR METHODS, BUT ALSO OUTPERFORMING RGBD-BASED METHODS IN FOREGROUND SCENE FLOW ESTIMATION, WHICH PROVES THAT THE CROSS-SCALE MATCHING CAN ACCURATELY ESTIMATE THE 3D MOTION.

Method	Typev	D1-bg	D1-fg	D1-all	D2-bg	D2-fg	D2-all	Fl-bg	Fl-fg	Fl-all	SF-bg	SF-fg	SF-all
CamLiFlow [12]	RGBD	1.48	3.46	1.81	-	-	3.19	-	-	4.05	-	-	5.62
CamLiFlow-RBO [12]		1.48	3.46	1.81	1.92	8.14	2.95	2.31	7.04	3.1	2.87	12.23	4.43
RigidMask+ISF [49]		1.53	3.65	1.89	2.09	8.92	3.23	2.63	7.85	3.5	3.25	13.08	4.89
RAFT3D [39]		1.48	3.46	1.81	2.51	9.46	3.67	3.39	8.79	4.29	4.27	13.27	5.77
Optical expansion [20]	Mono	1.48	3.46	1.81	3.39	8.54	4.25	5.83	8.66	6.3	7.06	13.44	8.12
Binary TTC [21]		1.48	3.46	1.81	3.84	9.39	4.76	5.84	8.67	6.31	7.45	13.74	8.5
TPCV [62]		1.48	3.46	1.81	<u>2.29</u>	**7.63**	<u>3.18</u>	<u>4.53</u>	**5.52**	<u>4.69</u>	<u>5.34</u>	**10.60**	<u>6.21</u>
Scale-flow [35]		1.48	3.46	1.81	2.55	8.24	3.50	5.24	5.71	5.32	6.06	11.32	6.94
ScaleFlow++ (ours)		1.48	3.46	1.81	**2.13**	<u>8.02</u>	**3.11**	**3.94**	<u>5.59</u>	**4.21**	**4.81**	<u>10.69</u>	**5.79**

in frame1 and frame2.

We demonstrate the TTC performance of ScaleFlow++ in Fig. 10. Our method can easily distinguish the collision sequence and approximate collision time of different moving objects.

D. Optical Flow

Results are shown in Tab. VI. Compared to the previous version of Scale-flow, ScaleFlow++ has improved the overall performance by at least 18% and 19% on Sintel and KITTI, respectively, demonstrating the effectiveness of the improvements proposed in this paper.

Moreover, compared with existing state-of-the-art optical flow methods, ScaleFlow++ is also very competitive. On the Sintel test, it leads most RAFT-based methods and achieves even better performance on the KITTI test.

TABLE VI
COMPARE OPTICAL FLOW PERFORMANCE WITH OTHER STATE-OF-THE-ART METHODS. THE BEST AMONG ALL ARE BOLDED, AND THE SECOND BEST ARE UNDERLINED. * DENOTES THE USE OF TARTANAIR [65] AS ADDITIONAL TRAINING DATA.

		Sintel-test		K15-test		
Base	Method	Clean↓	Final↓	Fl-all↓	Fl-bg↓	Fl-fg↓
RAFT	SEA-RAFT(M)*	1.44	2.86	4.64	4.47	<u>5.49</u>
	SEA-RAFT(L)*	1.31	2.60	<u>4.30</u>	<u>4.08</u>	**5.37**
	RAFT [22]	1.61	2.86	5.10	4.74	6.87
	CRAFT [66]	1.45	2.42	4.79	4.58	5.85
	MS-RAFT [67]	1.37	2.67	4.58	4.88	6.38
	GMA [13]	1.39	2.47	5.15	-	-
	Scale-flow [35]	1.88	3.16	5.32	5.24	5.71
	ScaleFlow++(ours)	1.33	2.77	**4.21**	**3.94**	5.59
Transformer	GMFlowNet [68]	1.39	2.65	4.79	4.39	6.84
	GMFlow [69]	1.74	2.90	9.32	9.67	7.57
	FlowFormer [70]	<u>1.20</u>	**2.12**	4.68	4.37	6.18
	GMFlow++ [71]	**1.03**	<u>2.37</u>	4.49	4.27	5.60

E. Scene Flow

Correctly estimating the 3D motion of an object is crucial for the robot path planning and action prediction task in a dynamic environment [1], [47]. Scene flow consists of optical

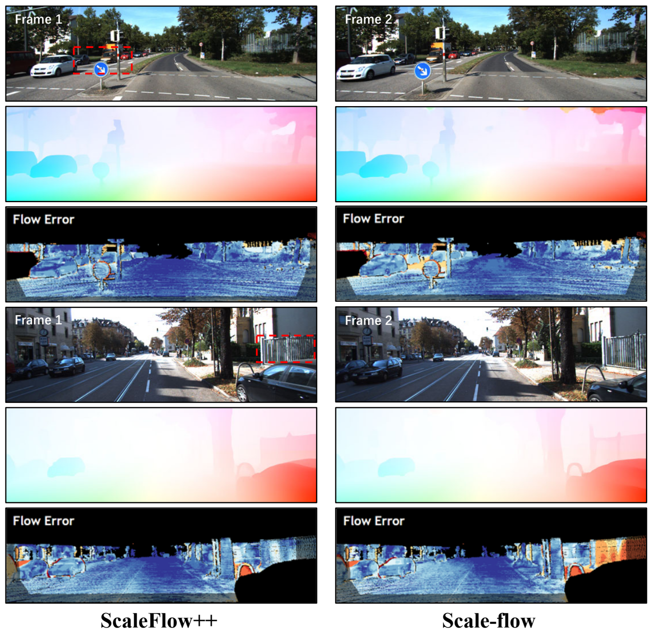

Fig. 11. **Visualization results on the KITTI test.** In each group, from top to bottom, there are two consecutive frames of images, the visualized optical flow, and the visualized optical flow error map. The redder the color of the error map, the greater the error. Observing the red box area in the image, our improved method better optimizes the background over a large area.

flow and depth changes of matched pixels between two frames. In order to unify the evaluation standards with the most advanced methods, we use GA-Net [72] to obtain the depth of the first frame $D1$ and calculate the depth change through MID $\boldsymbol{\tau}$, where $D2 = \boldsymbol{\tau} * D1$. The model was trained on the K-200 and tested on the KITTI public scene flow evaluation website.

We first compare ScaleFlow++ with monocular methods, as shown in Table V, ScaleFlow++ leads in all monocular methods with significant advantages, consistent with the previous superior performance in motion-in-depth tasks. It proves the effectiveness of our ScaleFlow++ framework in the field of monocular 3D motion estimation.

We also compare ScaleFlow++ with the state-of-the-art

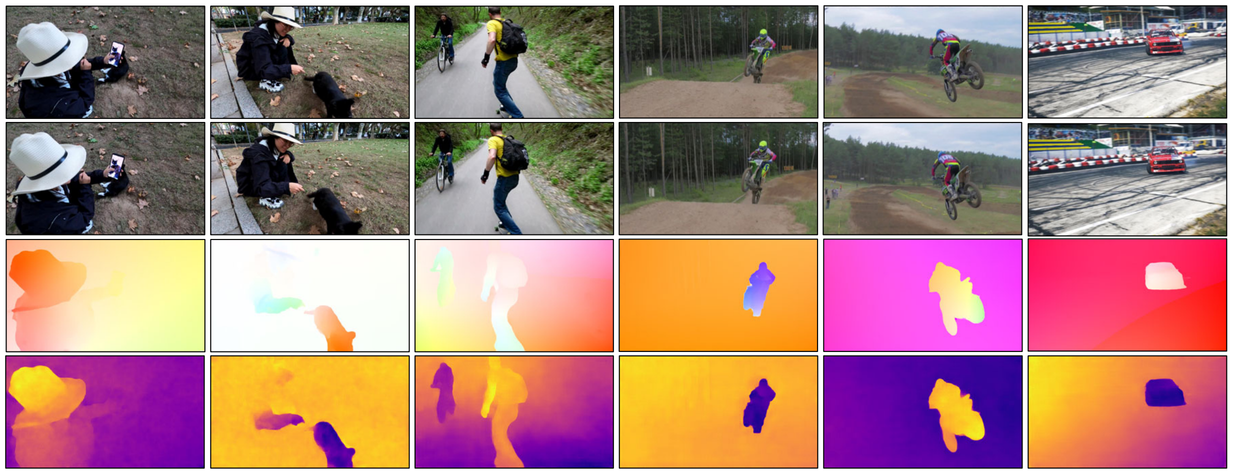

Fig. 12. Self-supervised generalization. ScaleFlow++ has demonstrated stunning results in natural scenes that have never been seen before, demonstrating that our method is robust and easy to generalize.

RGBD-based methods RAFT3D, CamLiFlow, and RigidMask+ISF. As shown in Tab. V, RGBD-based methods tend to get smaller Fl-bg errors, which is thanks to the aid of depth-based rigid assumption to optimize the background optical flow. Nonetheless, our monocular method still achieved competitive results in the core indicator SF-all and significant advantages in foreground estimation (SF-fg) compared to the RGBD-based method (10.69 v.s. 12.23).

F. Motion-in-depth on KITTI Test

Since the KITTI testing platform does not directly indicate motion-in-depth, we indirectly measure the accuracy of motion-in-depth based on the depth error growth (DEG) rate.

$$\mathrm{DEG} = \frac{\mathrm{D2} - \mathrm{D1}}{\mathrm{D1}} \tag{26}$$

where D1 and D2 are the depth outlier rates in the first and second frames, respectively. DEG describes how much error is added to the depth corresponding to the second frame based on the depth estimation of the first frame.

TABLE VII
COMPARISON OF GROWTH RATES OF DEEP MOTION ERROR ON KITTI TEST.

Method	Input	DEG-bg↓	DEG-fg↓	DEG-all↓
CamLiFlow-RBO	RGBD	0.30	1.35	0.63
RigidMask+ISF		0.37	1.59	0.71
RAFT3D		0.70	1.73	1.03
Optical expansion	Mono	1.29	1.47	1.35
Binary TTC		1.59	1.71	1.63
TPCV		<u>0.55</u>	**1.21**	<u>0.76</u>
Scale-flow		0.72	1.38	0.93
ScaleFlow++		**0.44**	<u>1.32</u>	**0.72**

As shown in Tab. VII, our method outperforms all monocular methods in DEG-all and is highly competitive even compared to RGBD methods. In addition, the performance of ScaleFlow++ is particularly impressive in the evaluation of the background area, mainly due to the better global view provided by the GIR module to optimize the background.

G. Generalization in Real World

To assist readers in better applying ScaleFlow++, we trained generalization weights on 60000 real-world image pairs using a 3D flow self-supervised generalization method, ADFactory [59].

As shown in Fig. 12, whether it is optical flow or MID, the ScaleFlow++ trained by self-supervision has shown excellent generalization results in unseen scenarios. We hope that our work can help methods in fields such as human pose recognition, action prediction, and autonomous driving better perceive the three-dimensional world.

V. CONCLUSION

In this article, we propose a new cross-scale recurrent all-pairs field transform for 3D motion estimation (ScaleFlow++), which extends the optical flow matching from the original image-to-image to image-to-scale-space, reducing the interference of scale changes on optical flow matching. Moreover, matching in scale space can also obtain depth motion clues, helping 3D flow methods break away from depth dependence and regression dependence. We also propose a global iterative refinement (GIR) module and an initialization module to address the inherent problems of small network perception range and task alienation in iterative methods. These modules enable the network to perceive global motion and achieve more robust optimizer learning in fewer iterations. The comprehensive evaluation of different datasets shows that ScaleFlow++ performs excellently in various 3D tasks, especially in scenes with large scale changes (driving scenes). We hope this exciting result draws people to pay more attention to 3D motion estimation based on a monocular camera.

References

[1] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *International Journal of Computer Vision*, vol. 130, no. 5, pp. 1366–1401, 2022.

[2] J. Byrne and C. J. Taylor, "Expansion segmentation for visual collision detection and estimation," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 875–882.

[3] A. Manglik, X. Weng, E. Ohn-Bar, and K. KITANI, "Future near-collision prediction from monocular video: Feasibility, dataset, and challenges," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[4] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1750–1757.

[5] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[6] V. Brebion, J. Moreau, and F. Davoine, "Real-time optical flow for vehicular perception with low-and high-resolution event cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 066–15 078, 2021.

[7] M. A. Sormoli, M. Dianati, S. Mozaffari, and R. Woodman, "Optical flow based detection and tracking of moving objects for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[8] H. Shi, Y. Zhou, K. Yang, X. Yin, Z. Wang, Y. Ye, Z. Yin, S. Meng, P. Li, and K. Wang, "Panoflow: Learning 360° optical flow for surrounding temporal understanding," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5570–5585, 2023.

[9] S. Alletto, D. Abati, S. Calderara, R. Cucchiara, and L. Rigazio, "Self-supervised optical flow estimation by projective bootstrap," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3294–3302, 2018.

[10] T. Marinho, M. Amrouche, V. Cichella, D. Stipanović, and N. Hovakimyan, "Guaranteed collision avoidance based on line-of-sight angle and time-to-collision," in *Annual American Control Conference*, 2018, pp. 4305–4310.

[11] K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2021.

[12] H. Liu, T. Lu, Y. Xu, J. Liu, W. Li, and L. Chen, "Camliflow: bidirectional camera-lidar fusion for joint optical flow and scene flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5791–5801.

[13] 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*, 2021, pp. 9772–9781.

[14] Y. Zhang and C. Kambhamettu, "On 3d scene flow and structure estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, 2001, pp. II–II.

[15] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 71–1, 2020.

[16] T. Camus, "Calculating time-to-contact using real-time quantized optical flow," *Nation Institute of Standards and Technology Nistir*, 1995.

[17] D. Muller, J. Pauli, C. Nunn, S. Gormer, and S. Muller-Schneiders, "Time to contact estimation using interest points," in *12th International IEEE Conference on Intelligent Transportation Systems*, 2009, pp. 1–6.

[18] M. Sagrebin, A. Noglik, and J. Pauli, "Robust time-to-contact calculation for real time applications," in *Proc. of 18th International Conference on Computer Graphics and Vision*, 2008, pp. 128–133.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[20] G. Yang and D. Ramanan, "Upgrading optical flow to 3d scene flow through optical expansion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1334–1343.

[21] A. Badki, O. Gallo, J. Kautz, and P. Sen, "Binary ttc: A temporal geofence for autonomous navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 946–12 955.

[22] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*, 2020, pp. 402–419.

[23] D. Sun, X. Yang, M.-Y. Liu, and J. 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*, 2018, pp. 8934–8943.

[24] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European Conference on Computer Vision*, 2004, pp. 25–36.

[25] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *International Journal of Computer Vision*, vol. 67, pp. 141–158, 2006.

[26] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. 33, no. 3, pp. 500–513, 2010.

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer, 2014, pp. 818–833.

[28] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, pp. 79–116, 1998.

[29] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, pp. 43–72, 2005.

[30] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "Unet 3+: A full-scale connected unet for medical image segmentation," in *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2020, pp. 1055–1059.

[31] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "Convnext v2: Co-designing and scaling convnets with masked autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 133–16 142.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[33] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015, pp. 427–434.

[34] 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)*, 2012, pp. 611–625.

[35] H. Ling, Q. Sun, Z. Ren, Y. Liu, H. Wang, and Z. Wang, "Scale-flow: Estimating 3d motion from video," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 6530–6538.

[36] D. Sun, S. Roth, J. Lewis, and M. J. Black, "Learning optical flow," in *European Conference on Computer Vision*, 2008, pp. 83–97.

[37] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[38] 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*, 2021, pp. 9772–9781.

[39] Z. Teed and J. Deng, "Raft-3d: Scene flow using rigid-motion embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8375–8384.

[40] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," *Advances in neural information processing systems*, vol. 32, 2019.

[41] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[42] Y. Lu, J. Valmadre, H. Wang, J. Kannala, M. Harandi, and P. Torr, "Devon: Deformable volume network for learning optical flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2705–2713.

[43] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[44] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015, pp. 2758–2766.

[45] L. Xu, Z. Dai, and J. Jia, "Scale invariant optical flow," in *European Conference on Computer Vision*, 2012, pp. 385–399.

[46] S. Wang, L. Luo, N. Zhang, and J. Li, "Autoscaler: scale-attention networks for visual correspondence," *arXiv preprint arXiv:1611.05837*, 2016.

[47] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[48] C. Vogel, K. Schindler, and S. Roth, "3d scene flow estimation with a piecewise rigid scene model," *International Journal of Computer Vision*, pp. 1–28, 2015.

[49] G. Yang and D. Ramanan, "Learning to segment rigid motions from two frames," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1266–1275.

[50] G. Puy, A. Boulch, and R. Marlet, "Flot: Scene flow on point clouds guided by optimal transport," in *European Conference on Computer Vision*, 2020, pp. 527–544.

[51] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 529–537.

[52] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6954–6963.

[53] W. Wu, Z. Qi, and F. P. Li, "Deep convolutional networks on 3d point clouds. ieee," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9613–9622.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[55] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[57] G. Xu, X. Wang, X. Ding, and X. Yang, "Iterative geometry encoding volume for stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 919–21 928.

[58] Y. Wang, L. Lipson, and J. Deng, "Sea-raft: Simple, efficient, accurate raft for optical flow," *arXiv preprint arXiv:2405.14793*, 2024.

[59] H. Ling, Q. Sun, Y. Sun, X. Xu, and X. Li, "Adfactory: An effective framework for generalizing optical flow with nerf," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 591–20 600.

[60] N. Mayer, E. Ilg, P. Hausser, 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 4040–4048.

[61] J. Hur and S. Roth, "Self-supervised monocular scene flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7396–7405.

[62] H. Ling, Y. Sun, Q. Sun, and Z. Ren, "Learning optical expansion from scale matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5445–5454.

[63] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, "Mseg: A composite dataset for multi-domain semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2879–2888.

[64] D. N. Lee, "A theory of visual control of braking based on information about time-to-collision," *Perception*, vol. 5, no. 4, pp. 437–459, 1976.

[65] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4909–4916.

[66] 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," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 602–17 611.

[67] A. Jahedi, M. Luz, L. Mehl, M. Rivinius, and A. Bruhn, "High resolution multi-scale raft (robust vision challenge 2022)," *arXiv preprint arXiv:2210.16900*, 2022.

[68] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, and D. Metaxas, "Global matching with overlapping attention for optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 592–17 601.

[69] 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*, 2022, pp. 8121–8130.

[70] 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," in *European Conference on Computer Vision*, 2022, pp. 668–685.

[71] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, F. Yu, D. Tao, and A. Geiger, "Unifying flow, stereo and depth estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[72] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 185–194.