electronics

MDPI

Article

Fusing Skeleton-Based Scene Flow for Gesture Recognition on Point Clouds

Yahui Liu and Jiajia Jiao *

College of Information Engineering, Shanghai Maritime University, Shanghai 201200, China; 202230310060@stu.shmtu.edu.cn
* Correspondence: jiaojiajia@shmtu.edu.cn

Abstract: Dynamic gesture recognition has recently aimed to learn static and motion features by exploiting point clouds from depth images. However, the weak correlation between some pixels and hand gestures makes the extracted dynamic features redundant. When search points and adjacent points in a larger feature space maintain movement consistency, more detailed movements are ignored. To improve the ability to capture fine-grained dynamic features and improve the relevance of point clouds and gestures, we propose a novel method of fusing skeleton-based scene flow for gesture recognition (FSS-GR) for higher recognition accuracy. Firstly, skeletons are automatically converted into pairs of point clouds. Based on the time interval between source and target point clouds and scene flow measurement indicators, four scene flow estimators are obtained. To minimize the additional cost of capturing fine-grained information, scene flow is used as datasets before fusion. Then, the coarse-grained dynamic features from depth images are fused with the obtained scene flow using different strategies, so that the flexible tradeoffs between model complexity and recognition performance are available for various scenarios. The comprehensive experiments and ablation study on SHREC'17 and DHG demonstrate that FSS-GR achieves a higher accuracy than state-of-the-art works.

Keywords: dynamic gesture recognition; scene flow; point cloud; skeleton

Academic Editors: Jungpil Shin, Md. Al Mehedi Hasan and Hoang D. Le

Received: 6 January 2025
Revised: 25 January 2025
Accepted: 29 January 2025
Published: 31 January 2025

Citation: Liu, Y.; Jiao, J. Fusing Skeleton-Based Scene Flow for Gesture Recognition on Point Clouds. *Electronics* **2025**, *14*, 567. https://doi.org/10.3390/electronics14030567

1. Introduction

Dynamic gesture recognition, as a popular task in computer vision, is widely used in various scenarios such as autonomous driving, smart home, and smart healthcare. Recent related works [1–9] have achieved good performance in terms of common gesture datasets [10,11].

According to the modality of inputs, the existing work on dynamic gesture recognition can be classified into three categories: image [5,11–13], skeleton [4,7–10,14–17], and point clouds [1–3,6]. The RGB or RGB-D images are easily obtained and the large data size makes the models converge faster. However, images are susceptible to irrelevant factors, such as occlusion and background. Recently, more and more work [1–3] has begun to focus on the 3D point cloud. The point cloud contains the latent spatial information of objects and maintains shape features. The first step of those works [1–3,6] is to convert depth images to point clouds without using skeletons. The data size of the depth image (for example, 128 points) is larger than that of the skeleton (22 joints) [10]. The bigger data size is positive for making the models converge. However, the neighborhood of the search point in the images is composed of some pixels which are joint points and other pixels that have less correlation with hand gesture. Instead, the skeleton information from the coordinates of the hand joints is more robust to irrelevant factors such as lighting or occlusion [4].

The correlation between skeletons and hand movements is greater. Therefore, our work explores the point cloud from both skeletons and depth images for better recognition performance. We convert the skeleton information of each gesture into pairs of point clouds. Every gesture is uniformly divided into different groups along the time dimension. The source point cloud comes from the first frame of each group, and its corresponding target point cloud depends on the time interval. Then, dynamic features are captured from pairs of point clouds.

Due to the irregularity of the point clouds, it is challenging to track the point-by-point correspondence between different frames [1]. Previous works [1–3,6] indirectly represent motion features by capturing the features of adjacent points. Kinet [1] keeps the adjacent points in the same feature-level ST-surfaces, and the normal vector of the surface can represent the dynamic features well. PointLSTM [2] optimizes the long short-term memory (LSTM) by combining information from the current frame and neighbors from the past frame while preserving the spatial structure. However, as shown in Figure 1, most of the works extract the dynamic features of the search point in the feature space of Neighborhood 1. In Neighborhood 1, the adjacent points of the search point are learned from the pixels on depth images. These points are related to the search radius of the ball query, the number of samples for adjacent points, and the time interval Δt_c. The works suppose that search point and its neighborhood keep consistency in movement. Therefore, the learned motion features of adjacent points (gray) are similar. And, detailed information is ignored that changes rapidly in time, such as fine-grained features (orange).

Some works aim to learn fine-grained dynamic features for higher accuracy while capturing coarse-grained features. MAE [7] is self-supervised in skeleton data to generalize with different hand gestures. Graph convolutional network (GCN)-based approaches [3–6,16] are widely explored by defining spatio-temporal graphs to incorporate spatial connectivity. SOGTNet [6] introduces multi-head attention to extract global features and an improved DGCNN to capture local features. ST-SGCN [5] pays more attention to directional or sparse interactions between the hand joints when learning subtle interactions. And, they introduce the cross-spatio-temporal module. FPPR-PCD [3] captures local features using DGCNN and the global position using DenseNet [18]. However, they only convert depth images to point clouds or extract skeletons from images. And, those models have a lower performance on SHREC'17 [10], which is a mixture of fine and coarse gestures. That is because coarse gestures lack distinctive features. And, precision in directed interactions is important when identifying fine gestures. Therefore, it is necessary to express the subtle gesture changes more accurately.

Scene flow estimation can provide the map of each 3D point in the previous frame to each point in the next frame. In other words, scene flow estimation aims to capture the motion vector of each point [19]. By converting the skeleton data into scene flow, we can capture dynamic information such as the trajectory, speed, direction, and acceleration of joint movements. This not only enables researchers to intuitively analyze actions, but also allows for the capture of the details of dynamic gestures. There have been many mature scene flow estimators [20–28]. Scene flow estimation is widely used in those areas such as robotics (path planning, collision avoidance), autonomous driving (tracking vehicles or people) and visual surveillance, which contain multiple tracked objects and complex spatial information. Recently, works have aimed to build self-supervised scene flow estimators that do not use ground-truth flow as labels [23,24,29,30]. Finding point cloud correspondences is widely applied to self-supervised scene flow estimators [23,31]. These works are inspired by Lang's work [32]. Lang et al. [32] suggests using latent space similarity and point features rather than regressing the true flow. However, to the best of our knowledge, they have not been used in gesture recognition in dynamic point clouds [1]. And, no

work has generated ground-truth flow in the existing gesture datasets. In addition to the irregularity of point clouds that makes the training time and inference time longer [23], the other primary reason is that most scene flow estimators rely on large datasets with ground-truth flow (FlyingThings3D [33] and KITTI [34,35]) to make the regression learning network converge.

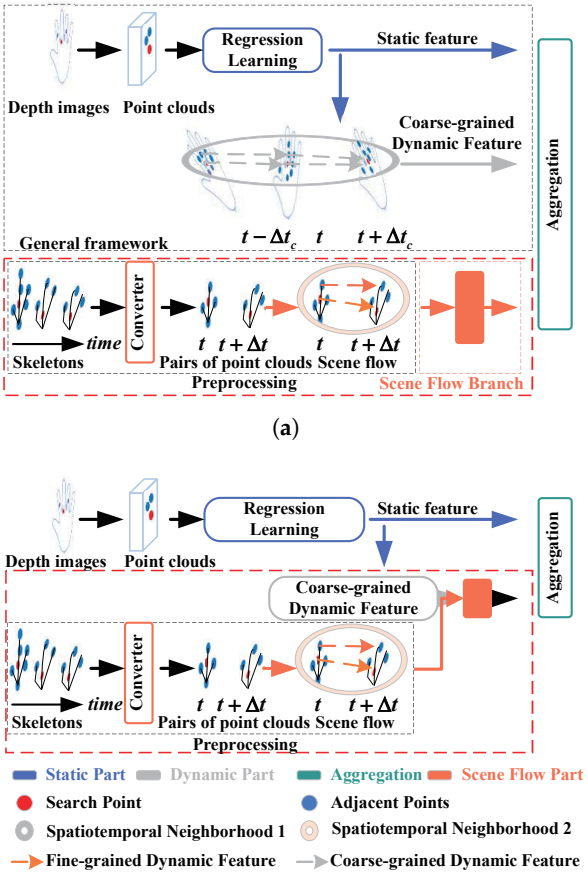

Figure 1. The basic idea of FSS-GR. (**a**) Multi-stream FSS-GR. (**b**) Two-stream FSS-GR. Compared with the general framework based on point clouds and FSS-GR, FSS-GR not only takes advantage of both the depth image and skeleton, but also integrates 3D motion features at different fine-grained levels. Most works learn the dynamic features of search point in Neighborhood 1. Our work fuses the fine-grained features represented by scene flow in smaller Neighborhood 2. Δt_c is the frame time interval between search points and adjacent points during the learning of coarse-grained features. Δt is the frame time interval between search (source) points and adjacent (target) points during the extraction of fine-grained features.

The paper proposes fusing skeleton-based scene flow for gesture recognition on point clouds (FSS-GR) for accurate recognition, as shown in Figure 1. We aim to use scene flow to measure the fine-grained feature of dynamic gestures represented by skeletons. Then, the scene flow and coarse-grained features are fused with different strategies. It is necessary to design an automatic converter to convert skeletons. In our work, gesture skeletons are converted into pairs of point clouds. And, point clouds are fed into self-supervised estimators to obtain scene flow. According to the time interval between different source and target point clouds and different scene flow measurement indicators, four different scene flows are estimated. The different time intervals and indicators are related to the feature space of spatial and temporal neighborhoods. Our work learns fine-grained features in Neighborhood 2, as shown in Figure 1. In Neighborhood 2, the features of adjacent points are captured from hand joints, which have a stronger correlation with gestures. The size of

Neighborhood 2 is related to the number of samples for adjacent (target) points and the time interval Δt. By setting Δt and the number of samples for source points, the size of Neighborhood 2 is smaller than that of Neighborhood 1. If we extract the scene flow in Neighborhood 2, the feature space of every gesture will be grouped into more different neighborhoods. The estimated 3D scene flow is more detailed and represents the subtle differences between the search point (red) and the adjacent point (blue). To avoid the high time cost of the scene flow estimator, the estimated scene flow is printed as the scene flow dataset.

As shown in Figure 1, multi-stream FSS-GR adds an independent scene flow branch to learn fine-grained motion features. The prediction scores from three branches are averaged as the final outputs during the testing phase. Two-stream FSS-GR fuses scene flow before the fusion of low-level static features and high-level dynamic features to supplement the fine-grained dynamic features.

In this paper, the main contributions are as follows:

- Point clouds used in previous gesture recognition methods are generated from depth images. FSS-GR explores the point cloud from both skeletons and depth images. Skeletons and depth images form an effective information supplement. And, bones with small data volumes are highly correlated with hand movements. Compared with works that feed skeletons to GCN-based networks, this work is the first to transform the skeleton information into pairs of point clouds. The time interval between source point clouds and target point clouds and fewer hand joints make the feature space of a search point smaller;
- We measure fine-grained features using scene flow. Then, the scene flow and coarse-grained features are fused with different strategies. An automatic converter is designed to convert skeletons, and four scene flow datasets are obtained: SHRECsft, SHRECsfe, SHRECsfe2, and SHRECsfe3. Those datasets can be fused with other static and dynamic features in gesture recognition to reduce the time cost. FSS-GR fuses scene flow and coarse-grained dynamic features with two strategies. Multi-stream FSS-GR includes an independent scene flow branch. Two-stream FSS-GR fuses the scene flow before the fusion of low-level static features and high-level dynamic features. The code is available at https://github.com/shawn-fei/fss-gr.git (accessed on 25 January 2025);
- Comparative experiments are conducted on various datasets to show efficacy in the performance, efficiency in parameters, and computational complexity of FSS-GR. Experiments are conducted on the SHREC'17 dataset [10] and DHG [14]. Noticeably, on SHREC'17, multi-stream FSS-GR obtains 1.4% and 0.8% accuracy gains in comparison with Kinet [1] and SOGTNet [6].

The structure of this paper is presented as follows. In Section 2, we analyze the related work on dynamic gesture recognition and flow estimation. In Section 3, we introduce FSS-GR. Initially, preprocessing is described in Section 3.1. In Section 3.2, two strategies are designed to fuse the scene flow with dynamic features. In Section 4, the experimental setup, results and cost analysis are presented, discussed, and analyzed. In Section 5, we conclude the contributions and limitations of FSS-GR.

2. Related Work

2.1. Deep Learning on Dynamic Gesture Recognition

Most previous works capture motion features using deep learning methods instead of manual features. That is because a deep learning method can be used for processing complex gestures.

The often-used deep learning methods for dynamic gesture recognition include a 3D convolutional neural network (CNN) [36–38], recurrent neural network (RNN) [11],

long short-term memory (LSTM) [17], and self-attention [3,39]. Three-dimensional CNN-based recognition methods [36–38] use 3D convolution and 3D pooling to extract temporal and spatial features from the original video in a parallel way. In the combination of 3D CNN and RNN, each video segment is usually sent to 3D CNN to extract spatio-temporal features, and the spatio-temporal information is sent to RNN to make up for the loss of time information due to video segmentation. The above works on 3D CNN and RNN are more concerned with capturing short-term relationships and using skeletons or depth images as input. Depth image-based methods are sensitive to occlusion and image resolution. In some skeletal-based methods [7,15], joint coordinates are fed into the network to learn hand characteristics. However, these methods ignore the spatial structure and motion features of skeletons. Some studies [4,5,16] define spatio-temporal maps based on skeletons or images. However, these methods face challenges with the mixture of coarse and fine features.

Most recent works have focused on extracting motion features from point clouds [1–4,6,40–42]. The point cloud contains the latent spatial information of objects for accurate recognition. Kinet [1] solves the normal vector of feature-level ST-surfaces to encode dynamically. PointLSTM [2] gathers current information and information from previous neighbors to avoid tracking the relationship between points of contact. FPPR-PCD [3] fuses local and global features in an LSTM to capture dynamic features. DG-STA [4] constructs a fully connected graph from a hand skeleton, and uses spatial and temporal self-attention to capture node features and edges. However, these works extract features without explicitly finding the corresponding relationship of joints. The proposed method FSS-GR fuses scene flow for considering the corresponding point of every source point. SOGTNet [6] has a similar motivation that extracts fine-grained features while capturing global gesture features. However, they only convert depth images to point clouds. This makes learned features redundant. In our work, point cloud sequences as input can be from both skeletons and depth images. We transform skeletons into pairs of point clouds to make point clouds and gestures more relevant.

2.2. Flow Estimation

Two-dimensional optical flow and 3D scene flow are often used to capture motion features. Two-stream network based on 2D optical flow [43,44] is composed of a spatial branch for extracting spatial features and a time branch for extracting dynamic features. However, the optical flow cannot represent the real motion of an object, and is actually the real motion projected onto images.

Scene flow estimators are used in autonomous driving [29,34], motion segmentation [30,45], free-viewpoint video [46,47], and visual tracking [48,49]. According to whether the model uses ground-truth flow, recent scene flow estimators are divided into supervised [20,45,46,49] and self-supervised methods [23,24,29,30]. FlowNet3D [20] is a classic model that uses flow regression, and the regression model relies on large datasets to make the network converge because of the irregularity and disorder of point clouds. FLOT [31] does not use flow regression in the training phase, but estimates points corresponding to source point cloud through spatial similarity. However, in the testing phase, the method still optimizes the flow of the initial scene with flow regression. In the testing phase, SCOOP [23] optimizes the initial scene flow with custom distance loss and smooth loss. This method reduces the training cost, but it relies on two large datasets synthesized in the middle, FlyingThings3D [33] and KITTI [34,35]. To the best of our knowledge, the ground-truth flow of gestures does not appear in the publicly available gesture datasets. This is because the scene flow at each joint point is complex and the high cost of annotation does not lead to efficient identification. This makes it difficult to generalize the previous approach to unknown test environments. Scene flow can play a complementary role in the recognition

of point cloud sequences, where hand skeletons are converted into pairs of point clouds, and the estimated scene flow is used as an intermediate synthetic dataset for 3D motion feature fusion.

3. Proposed Method

Dynamic gesture recognition on point clouds seeks to output the category label for a specific gesture. As shown in Figure 1, the proposed method named FSS-GR contains two parts:

- Preprocessing: scene flow is from hand skeletons. As described in Section 3.1, skeletons are converted into point clouds, and scene flow is estimated based on the different pairs of point clouds and metrics;
- Feature information extraction: static features and dynamic features are extracted to identify different types of gestures. It is the key to fuse the learned scene flow with the dynamic features captured from depth images.

Two strategies are designed to fuse scene flow with the dynamic features, as described in Section 3.2. A multi-stream network is called multi-stream FSS-GR and a two-stream network is called two-stream FSS-GR.

3.1. Preprocessing: Learn Scene Flow from Hand Skeletons

In this section, the goal is to input the 3D skeletal information of gestures at different moments, and output a scene flow of each joint.

Given a dynamic gesture of T_0 frames, the skeletons of each frame are represented by the 3D coordinate vectors of N_0 hand joints. In order to obtain the scene flow of joints, firstly source point clouds X are constructed from the skeletons of T key frames in the gesture. Then, target point clouds Y are constructed from the skeletons of T frames that are distinct from the T frames of source point clouds X so that each dynamic gesture has T pairs of point clouds as described in Section 3.1.1. Finally, the flow field about point-by-point motion from X to Y is estimated by using the self-supervised scene flow estimator. Because different metrics are used to build target point clouds and measure scene flow, different types of scene flow datasets can be extracted for gesture recognition, as described in Section 3.1.2.

3.1.1. Constructing Pairs of 3D Point Clouds on Skeletons

In this section, our goal is to build T pairs of point clouds. T pairs of point clouds consist of the source point cloud X and the corresponding target point cloud Y. For clarity, this work denotes a source point cloud, a target point cloud and pairs of point clouds as follows:

- A source point cloud: $X^t = \left\{ X_i{}^t \middle| i = 1, 2, \ldots N \right\}$;
- A target point cloud: $Y^t = \left\{ Y_j^t \middle| j = 1, 2, \ldots N \right\}$;
- Pairs of point clouds in one hand gesture: $\left\{ (X^t, Y^t) \middle| t = 1, 2, \ldots T \right\}$.

X is a sequence of T key frames. $X^t \in R^{N \times 3}$ is the tth source point cloud in chronological order, where t is an integer from 1 to t and N is the number of hand joints in the source point clouds X. $Y^t \in R^{N \times 3}$ is the tth target point cloud that corresponds to X^t.

Firstly, the point cloud X^t is composed of N disordered 3D points and represents static features. Each point is represented by the 3D coordinates of hand joints. There are N_0 hand joints in each frame of dynamic gesture. In order to reduce the computational cost on the premise of retaining most of the skeletal information, this work chooses to discard some points directly and retain the N hand joints. N points are sampled uniformly from the N_0

points of the original gesture. Therefore, static features represented by point clouds are the raw coordinates of joints without interference from irrelevant factors.

Secondly, T source point clouds are selected from every gesture and correspond to the T key frames of gestures. Similarly to Kinet [1] and PointLSTM [2], T frames are sampled uniformly from the T_0 frames of the original gesture.

Finally, in order to extract the dynamic features of each point in the source point cloud X^t, it is also necessary to obtain a target point cloud to form a pair of point clouds, which is part of dynamic scene.

This work assumes that X^t and a target point cloud Y^t form a pair of point clouds (X^t, Y^t). Then, the T source point clouds and the corresponding T target point clouds form T pairs of point clouds in each gesture.

The frame time interval Δt between the source and target point clouds will affect the scene flow and further influence the accuracy of gesture recognition. The larger Δt is, the more the learned scene flow characterize the long-term point-by-point correspondence. In this paper, in order to capture fine-grained features, only a small value is taken for Δt.

Figure 2 shows the process of extracting source point clouds and target point clouds from skeletons.

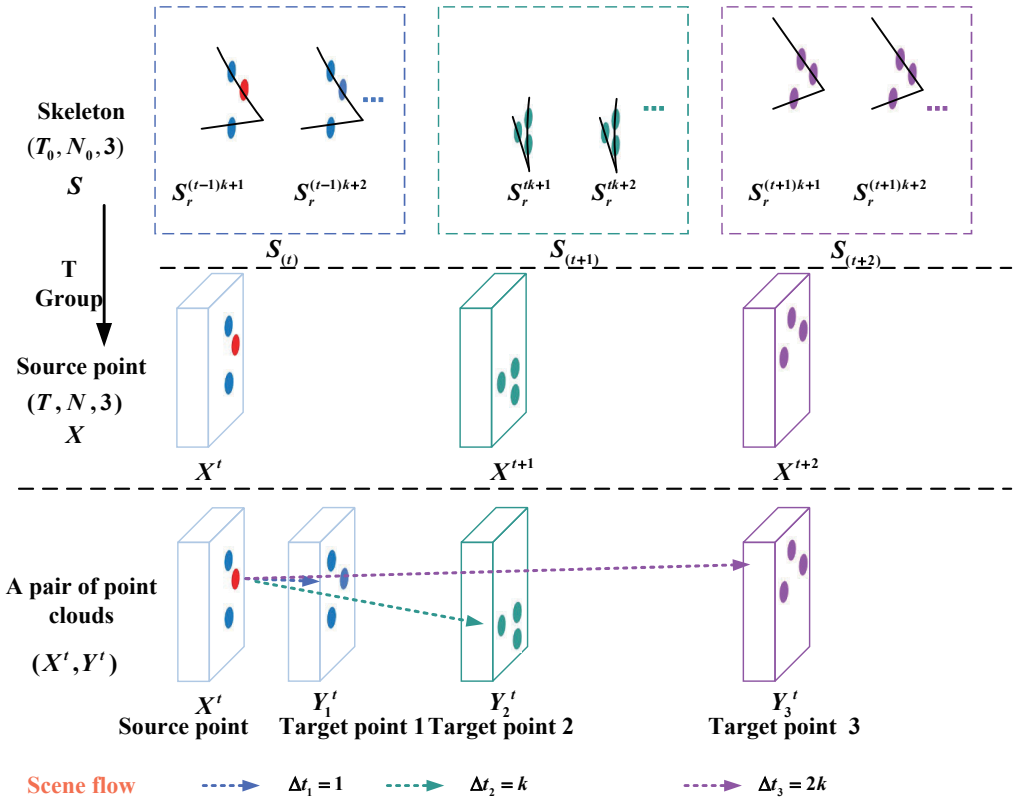

Figure 2. A pair of point clouds includes source point cloud and target point cloud. There are three methods to choose the target point cloud based on Δt. Δt is the frame time interval between the source and target point clouds. k is the average number of frames in every grouped gesture. When Δt is set to Δt_1, the scene flow of the search point(red) is learned from points(blue) in $S_{(t)}$. When Δt is set to Δt_2, the scene flow of the search point(red) is learned from points(green) in $S_{(t+1)}$ and points in X^t. When Δt is set to Δt_3, the scene flow of the search point(red) is learned from points(purple) in $S_{(t+2)}$ and points in X^t.

The initial gesture composed of T_0 frames is represented by S_{raw}, as shown in Equations (1). S_r^p is the tth frame in chronological order, where p is an integer from 1 to T_0.

$$S_{raw} = \left\{ S_r^p \middle| p = 1, 2, \ldots, T_0 \right\}. \quad (1)$$

Similarly to traditional methods for sampling, the T_0 frames in a gesture are uniformly divided into T(32) groups along the time dimension. The grouped gesture S_{group} is shown in Equation (2). In the sequence S_{group} consisting of T groups, $S_{(m)}$ is the mth group in chronological order, and m is an integer from 1 to T. Each group contains k frames. It is worth noting that $\frac{T_0}{T}$ is not necessarily an integer, so we round $\frac{T_0}{T}$, as shown in Equation (3). The first group $S_{(1)}$ and the mth group $S_{(m)}$ are shown in Equation (4) and (5). $S_{(m)}$ is a sequence consisting of k frames, and $S^q_{(m)}$ is the qth frame in chronological order. In particular, the last group $S_{(T)}$ is shown in Equation (6).

$$S_{group} = \left\{ S_{(m)} \mid m = 1, 2, \ldots, T \right\}. \tag{2}$$

$$k = \left[\frac{T_0}{T} \right]. \tag{3}$$

$$S_{(1)} = \left\{ S^q_{(1)} \middle| q = 1, 2, \ldots, k \right\} = (S^1_r, S^2_r, \ldots, S^k_r). \tag{4}$$

$$S_{(m)} = \left\{ S^q_{(m)} \middle| q = 1, 2, \ldots, k \right\} = (S^{(m-1)k+1}_r, S^{(m-1)k+2}_r, \ldots, S^{mk}_r). \tag{5}$$

$$S_{(T)} = (S^{(T-1)k+1}_r, S^{(T-1)k+2}_r, \ldots, S^{T_0}_r). \tag{6}$$

Then, the T source point clouds X are composed of the first frame of each group. For example, X^t is shown in Equation (7).

$$X^t = S^1_{(t)} = S^{(t-1)k+1}_r = S^{tk-k+1}_r. \tag{7}$$

The target point cloud Y^t depends on the frame time interval Δt. As shown in Figure 2, there are three ways to select target point clouds. In three methods, Δt is denoted as Δt_1,Δt_2 and Δt_3. The target point cloud Y^t_1 is from the next frame of the source point cloud X^t. It is as follows:

$$Y^t_1 = S^2_{(t)} = S^{(t-1)k+1+\Delta t_1}_r = S^{tk-k+2}_r, \tag{8}$$

where Δt_1 is set to one frame time. The target point cloud Y^t_2 is from the first frame in the next group of the source point cloud X^t. It is as follows:

$$Y^t_2 = S^1_{(t+1)} = S^{(t-1)k+1+\Delta t_2}_r = S^{tk+1}_r, \tag{9}$$

where Δt_2 is the k frame time. If $tk + 1$ is greater than T_0, the target point cloud is set to the last frame $S^{T_0}_r$. When Δt_3 is set to $2k$ frame time, target point cloud Y^t_3 is as Equation (10). If $tk + k + 1$ is greater than T_0, then the target point cloud is set to the last frame $S^{T_0}_r$.

$$Y^t_3 = S^1_{(t+2)} = S^{(t-1)k+1+\Delta t_3}_r = S^{tk+k+1}_r. \tag{10}$$

The value of Δt suggests that a pair of point clouds (X^t, Y^t) on skeletons represents a short segment of a dynamic gesture for fine-grained feature extraction.

3.1.2. Different Scene Flow Datasets Based on Different Metrics

Pairs of point clouds are fed into self-supervised scene flow estimator. Estimated scene flow has two representations. Based on different representation and pairs of point clouds, four scene flow datasets are automatically generated by our converter.

Since there is no ground-truth flow in gesture datasets, this work uses the self-supervised scene flow estimator SCOOP [23] as the backbone. Ground-truth flow is not used as supervision during the train and test phases. In the train phase, a softly matched

target point $\hat{Y}$ is obtained for each source point ${X_i}^t$ based on the similarity of the points in the feature space. Finally, the initial scene flow $F = \hat{Y} - {X_i}^t$ corresponding to each source point is obtained. In the test phase, considering the consistency of neighboring source points $N_X({X_i}^t)$ and flow field, $\hat{Y}$ should be as close as possible to $N_Y({X_i}^t)$, where $N_Y({X_i}^t)$ is a neighborhood of ${X_i}^t$ in $Y^{t+\Delta t}$. The initial scene flow F is refined, and the final scene flow F^* is obtained.

Taking into account the computational cost of scene flow and the impact on classification performance, two representations of the scene flow f will be used. One representation is the initial scene flow F learned during the train phase, and the other representation is the optimized scene flow F^* during the test phase. The computational cost of F^* is higher than F, but F^* is a refinement of F.

Finally, according to the different methods of target point cloud construction and the different ways of describing scene flow, four scene flow datasets are obtained, as shown in Table 1. The scene flow datasets, which characterize the fine-grained dynamic features of gestures, can be fused with coarse-grained dynamic features.

Table 1. Scene flow dataset. Δt is the frame time between the source point cloud and target point cloud. F and F^* are two ways of describing scene flow. F is the initial scene flow learned during the train phase. F^* is the optimized scene flow during the test phase.

Scene Flow Dataset	Δt	Representation of Scene Flow
SHRECsft	1	F
SHRECsfe	1	F^*
SHRECsfe2	k	F^*
SHRECsfe3	2k	F^*

3.2. Fusing Skeleton-Based Scene Flow for Point Cloud-Based Gesture Recognition

In this section, the goal is to input the scene flow of gestures and a point cloud sequence based on depth images, and output gesture categories. This paper uses Kinet as the backbone. It inputs the point cloud sequence into a mature static network for extracting the spatial features, and then utilizes the static spatial features to learn normal vectors at each point. However, the local neighborhood of a center point lies on the same ST-surface orthogonal to the normal vector. This work aims to combine the learned scene flow with the coarse-grained normal vector seamlessly. And, two architectures (multi-stream FSS-GR, two-stream FSS-GR) are proposed.

3.2.1. Multi-Stream FSS-GR (M-FSS-GR)

The multi-stream FSS-GR consists of three parts, as shown in Figure 3 :

- Static branch (blue) captures spatial features;
- Dynamic branch (gray): the dynamic feature is extracted by using the static feature learned by the static branch. And, it is the normal vector of the ST-surface and is coarse-grained;
- Scene flow branch (orange): The scene flow dataset generated in Section 3.1 is used as input to the deep learning network, which is designed to predict gesture classes using fine-grained motion features.

The scene flow is represented by Equation (11).

$$F = \{f_i^t | i = 1, 2, \ldots N.t = 1, 2, \ldots, T\}, f_i^t = (f_{x_i}^t, f_{y_i}^t, f_{z_i}^t). \tag{11}$$

As shown in Figure 3, module SA takes the 3D scene flow F/F^* of each gesture as input and outputs a d_1-dimensional feature vector representing the fine-grained features

of that gesture. In detail, group N points a gesture into a group, then sets (0, 0, 0) as the search point, and the flow field of the N points becomes the feature space of the gesture. Then, the 3D scene flow is fed into the network to obtain the high-dimensional features and obtains one feature vector $f_{sa} \in R^{d_1}$ of the entire dynamic gesture. We formulate the mechanism as Equation (12):

$$\widetilde{F} = g(F), f_{sa} = maxpool\left(\widetilde{F}\right). \tag{12}$$

This work denotes the operation of training the 1 × 1 convolution, BN (batch normalization), and ReLU (activation function, linear rectification function) twice as function g. To further obtain high-level dynamic features $f_c \in R^{d_3}$ and reduce the dimension of the feature vector from d_3 to 24 (number of gesture classes), the M-FSS-GR mechanism is designed as Equation (13). In Equation (13), $g(f_{sa}) \in R^{d_2}$. The mechanism of M-FSS-GR' is designed as Equation (14).

$$f_c = g(g(f_{sa})), f_{sf} = FC(f_c). \tag{13}$$

$$f_c = g(f_{sa}), f_{sf} = FC(f_c). \tag{14}$$

The final fine-grained feature f_{sf} of scene flow branch is obtained, as shown in Figure 3.

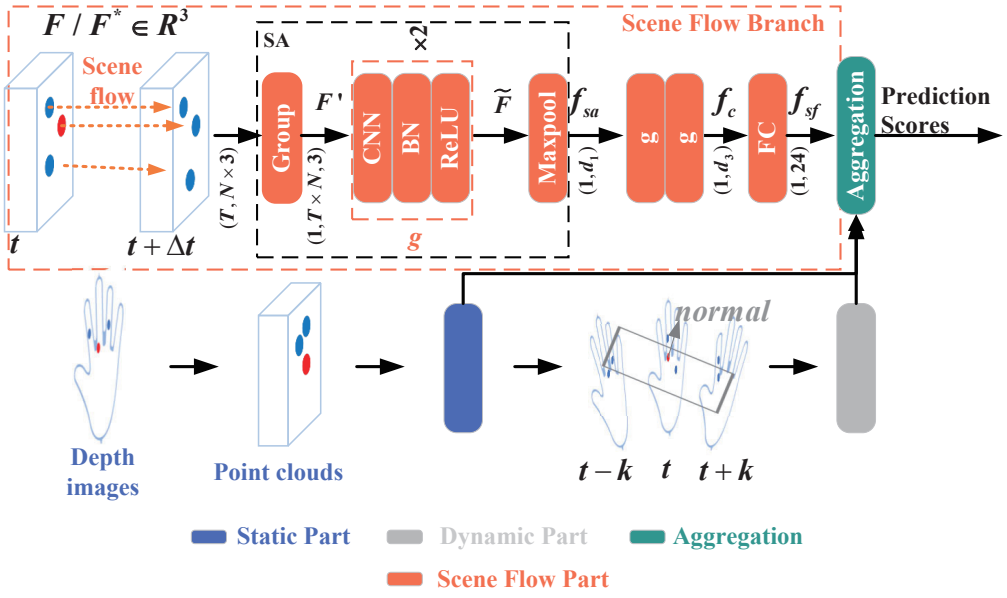

Figure 3. Multi-stream FSS-GR framework. In terms of input modality, the input for static and dynamic branches are depth images, and the initial input modality for scene flow is skeleton. The input of this branch is the scene flow dataset in Section 3.1. CNN is a 1 × 1 convolution. FC is the fully connected layer.

In the test phase, category predictions from static, dynamic, and scene flow streams are fused in different proportions. M-FSS-GR optimizes the three branches through cross-entropy loss. The total loss is defined as Equation (15). The ratio of the three branches—scene flow, static branch, and dynamic branch—is denoted as r_{sf}, r_s, and r_n.

$$L_{total} = r_{sf}L_{sf} + r_sL_s + r_nL_n. \tag{15}$$

3.2.2. Two-Stream FSS-GR (T-FSS-GR)

In contrast to the new scene flow stream of multi-stream FSS-GR, two-stream FSS-GR fuses scene flow to supplement the learning of fine-grained motion features in dynamic stream. In other words, we use early fusion in T-FSS-GR.

The architecture of T-FSS-GR is shown in Figure 4. The static feature is a coordinate vector that represents static space based on the depth images. The normal vector is a multi-dimensional vector that represents local motion features based on the static feature, so it is essentially a feature learned from the depth images. In order to take into account skeletons and depth images with complementary information, the scene flow (orange) is fused with a static feature and normal vector.

First, two different features are concatenated in the direction of the point:

$$f_{ss} = concat(F, f_s), n_s = concat(F', n), \tag{16}$$

which doubles the number of feature points in each frame. Next, we fuse $2N$ points into N feature points. In order to further capture the time information, f'_{ss}/n'_s is connected with time. The learning mechanism is formulated as Equation (17).

$$f_{sst} = concat(t, g(f_{ss})) = concat(t, f'_{ss}), n_{st} = concat(t, g(n_s)) = concat(t, n'_s). \tag{17}$$

After the dimension alignment with another g, the static and dynamic features (f'_{sst} *and* n'_{st}) which learn scene flow are obtained. Eventually, f'_{sst} and n'_{st} are forwarded to SA and FC, followed by the last aggregation of static and dynamic results. In T-FSS-GR, the total loss is defined as Equation (18). The ratio of two branches (dynamic and static branch) is denoted as r_d and r_s.

$$L^{(2)}_{total} = r_d L_d + r_s L_s. \tag{18}$$

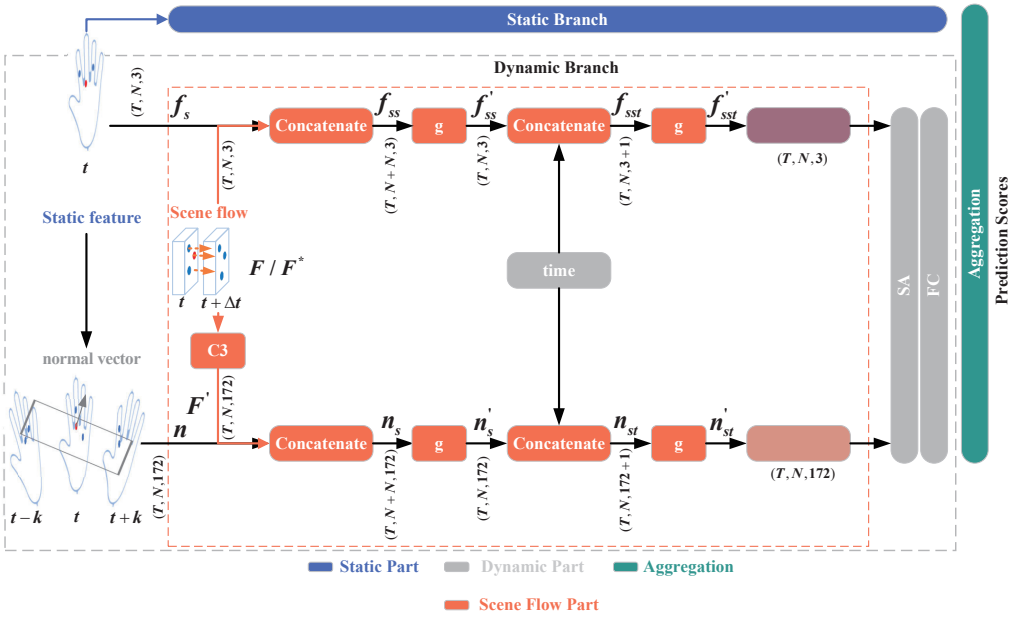

Figure 4. Two-stream FSS-GR framework. Two-stream FSS-GR consists of a static branch that extracts spatial features and a dynamic branch that captures dynamic features. Scene flow is the supplementary learning of normal vectors in the dynamic branch. In T-FSS-GR, the input of module SA are 3D static features and multi-dimensional normal vectors that fuses fine-grained scene flow. f'_{sst}(brown) is the static feature which learns scene flow. n'_{st}(pink) is the dynamic feature which learns scene flow.

4. Experiment

To evaluate the performance of FSS-GR, comprehensive experiments are conducted on the dynamic gesture recognition dataset SHREC'17 and DHG. And, FSS-GR is compared with the state-of-the-art works on dynamic gesture recognition from the perspective of recognition performance and cost. An ablation study is also performed to verify the effectiveness of each module in FSS-GR.

4.1. Experimental Setup

4.1.1. Datasets

SHREC'17 [10]: SHREC'17 is a public dynamic gesture dataset that not only provides the coordinates of 22 hand joints in the 3D world, but also provides depth images. It contains 2800 videos composed of 28 types of gestures, including 1960 videos (70%) in the training set and 840 videos (30%) in the test set.

SHRECsft, SHRECsfe, SHRECsfe2, and SHRECsfe3 are four datasets about the scene flow of dynamic gestures. These datasets are generated based on the *Skeletons_world* provided by SHREC'17. The 3D coordinates of 22 joints per frame for each dynamic gesture are provided in *Skeletons_world*. In this paper, the skeletal information is processed into 32 pairs of point clouds, each of which contains a source point cloud and a target point cloud. However, the ground truth flow corresponding to the source point cloud is not provided in *Skeletons_world*. Therefore, a self-supervised scene flow estimator is used to extract the scene flow for each joint. Each scene flow dataset consists of a 3D scene flow of joints for 2800 videos. There are two aspects of differences between the four datasets: time interval and the metrics of scene flow, as detailed in Table 1.

DHG [14]: This dataset includes 2800 dynamic gesture sequences. There are 28 types of dynamic gesture that were conducted by 20 subjects using the whole hand. The modalities of the sequences are skeletons and depth images. The skeletons are the coordinates of 22 hand joints in the 3D world. Because there is no official division between the training set and test set, this work follows the common means of evaluation, which is leave-one-subject-out cross-validation.

4.1.2. Experimental Configuration

Experiments about scene flow estimation are conducted on GPU equipped with V100-32G. The manufacturer of V100-32G is NVIDIA, whose headquarters is located in Santa Clara, California, United States. We equip the GPU with a GPU Cloud service provider named Gpushare Cloud. Gpushare Cloud is located in Shanghai, China.To obtain the same key frames, when processing the initial skeletons and depth images, we uniformly sample along the timeline of each dynamic gesture, following common practice. The quantity of key frames for each gesture, denoted as T, is set to 32, which aligns with the prevalent number of key frames utilized in recent works. Before the fusion of scene flow and coarse-grained features, the number of feature points in each frame, denoted as N, is set to 16. During the learning of coarse-grained features, the greater the time interval Δt_c, the larger the search radius. For example, in the first layer, the search radius is an arithmetic sequence ranging from 0.5 to 0.6. The search radius in the second layer is twice that of the first layer. The number of samples for adjacent points is 64. Regarding the dynamic characteristics of source points in the source point cloud X, adjacent points have similar features. The size of the neighborhood for each source point is denoted as K_f. In other words, each source point has similar motion features with adjacent K_f points. In our work, K_f is set to 12. For the dimensions of points and the hyperparameters of losses in the feature space, this paper retains the settings of the original work [23]. In the training of the flow estimator SCOOP [23], the batch size is set to 16 and the epoch is set to 100. In the training of the

FSS-GR, the batch size is set to 8, the epoch is set to 250, the learning rate is 0.001, the decay step is 200,000, and the decay rate is 0.7.

Concerning the number of channels for MLP in M-FSS-GR, d_1 is set to 128 and d_3 is set to 1024. In M-FSS-GR, d_2 is 256.

FSS-GR is implemented in TensorFlow. All experiments are conducted on GPU equipped with V100-16G. To make a fair comparison, a static branch is trained first and followed by freezing, and finally, mainly a dynamic branch and a scene flow branch are trained. For other hyper-parameters such as learning rate and decay step, our work keeps the same settings of the original work [1]. Classification accuracy is selected for the performance evaluation index as in the previous works. FLOPs (floating-point operations) and Params (the number of parameters) are the evaluation indicators of the time complexity and spatial complexity of model.

4.2. Performance

4.2.1. Comparison with the State-Of-The-Art (SOTA)

FSS-GR is compared with recent advanced methods. We replicate approaches [1–3,6] on point clouds on the same GPU equipped with V100-16G.

As shown in Table 2, FSS-GR achieves the best performance with 95.2% accuracy on SHREC'17, with gains of 1.4% and 0.8% compared to advanced works Kinet [1] and SOGTNet [6]. As shown in Table 3, FSS-GR achieves the best performance with accuracy of 93.5% in DHG, with gains of 0.9% and 0.3% compared to the advanced works using SOGTNet [6] and Shen's annotation framework [8]. This results from the fusion of fine-grained scene flow and coarse-grained dynamic features on point clouds. Moreover, as shown in Table 4, the proposed method, M-FSS-GR, achieves gains of 0. 5% and 0. 8% in both static and dynamic branches compared to the method using point clouds in depth images. The difference between M-FSS-GR and M-FSS-GR' is the number of channels for MLP (multi-layer perceptron). In M-FSS-GR, by fusing heterogeneous information through more branches, we can completely extract features from different granularities, so the accuracy can be improved.

As shown in Table 5, we compare FSS-GR with advanced GCN-based approaches in recent years. From the perspective of modality, FSS-GR not only converts the depth images into point clouds, but also converts the raw skeleton data into point clouds. To our knowledge, in dynamic gesture recognition, FSS-GR is the only model to convert skeletons into point clouds. And, it has the highest accuracy on two datasets. This shows that the conversion of bones into point clouds is helpful to improve the accuracy of gesture recognition.

From the perspective of different modules of each model, FSS-GR and other GCN-based approaches have similar motivations. The fine-grained features of dynamic gestures are extracted from the scene flow branch of FSS-GR. The improved DGCNN in SOGTNet and FPPR-PD captures local features. The accuracy of the scene flow branch is low, but the accuracy of FSS-GR is the highest after integrating different types of features. This shows that the fusion of scene flow and coarse-grained features is effective.

These results indicate the following two aspects:

- Compared to SOTA works, FSS-GR performs more accurate gesture recognition by capturing the scene-flow-assisted 3D static features and motion features;
- FSS-GR characterizes the 3D motion of dynamic gestures more completely from skeletons and depth images, due to an automatic converter and fusion between different data granularities.

Table 2. Performance comparison (%) on SHREC'17 dataset. The results of 28 gestures are reported, where pc is the abbreviation for point cloud; and di is the abbreviation for depth image.

Method	Modality	Accuracy (%)
Key frames [10]	di	71.9
SoCJ+HoHD+HoWR [14]	skeleton	81.9
Res-TCN [15]	skeleton	87.3
STA-Res-TCN [15]	skeleton	90.7
ST-GCN [16]	skeleton	87.7
MAE [7]	skeleton	90.0
DG-STA [4]	skeleton	90.7
Shen's Annotation Framework [8]	skeleton	92.6
ST-SGCN [5]	RGB → skeleton	92.9
PointLSTM [2]	di → pc	93.4
FPPR-PCD [3]	di → pc	93.8
SOGTNet [6]	di → pc	94.4
Kinet [1]	di → pc	93.8
FSS-GR (ours)	di+skeleton → pc	95.2

Table 3. Performance comparison (%) on DHG dataset. The results of 28 gestures are reported, where pc is the abbreviation for point cloud; and di is the abbreviation for depth image.

Method	Modality	Accuracy (%)
SoCJ+HoHD+HoWR [14]	skeleton	80.0
CNN+LSTM [17]	skeleton	81.1
Res-TCN [15]	skeleton	83.6
STA-Res-TCN [15]	skeleton	85.0
ST-GCN [16]	skeleton	87.1
HPEV [50]	skeleton	88.9
DG-STA [4]	skeleton	88.0
MS-ISTGCN [51]	skeleton	91.2
TD-GCN [52]	skeleton	91.4
SBI-DHGR [53]	skeleton	91.8
Shen's Annotation Framework [8]	skeleton	93.2
FPPR-PCD [3]	di → pc	91.7
SOGTNet [6]	di → pc	92.6
FSS-GR (ours)	di+skeleton → pc	93.5

Table 4. Performance comparison (%) at different branches. The inputs of FSS-GR are skeletons and depth images. If the modality is used in methods, we mark it with a tick (✓). Otherwise, we use an '✗' to mark it.

	Modality: Point Clouds		Accuracy (%) on SHREC'17			
Method	Skeleton	Depth Image	Total	Static	Dynamic	Scene Flow
PointLSTM [2]	✗	✓	93.4	-	-	-
FPPR-PCD [3]	✗	✓	93.8	-	-	-
SOGTNet [6]	✗	✓	94.4	-	-	-
Kinet [1]	✗	✓	93.8	87.6	91.7	-
M-FSS-GR (ours)	✓	✓	95.2	88.1	92.5	3.0
M-FSS-GR' (ours)	✓	✓	95.1	88.1	92.3	3.0
T-FSS-GR (ours)	✓	✓	94.5	88.1	88.3	-

Table 5. Comparison with GCN-based approaches. The accuracy (%) of 28 gestures are reported, where pc is the abbreviation for point cloud; di is the abbreviation for depth image.; d-coarse denotes the dynamic coarse-grained feature; d-fine; denotes the dynamic fine-grained feature; and FSS-GR-static denotes the static branch of FSS-GR.

Method	Modality	Feature	SHREC(%)	DHG(%)
FSS-GR-static	di → pc	static	88.1	86.3
FSS-GR-dynamic	di → pc	d-coarse	92.5	90.5
FSS-GR-scene flow	skeleton → pc	d-fine	3.0	3.3
FSS-GR	di+skeleton → pc		95.2	93.5
SOGTNet-SA	di → pc	global	70.7	68.1
SOGTNet-PointNet++	di → pc	global	74.8	71.4
SOGTNet-OA	di → pc	local	72.7	69.3
SOGTNet-DGCNN	di → pc	local	74.0	72.1
SOGTNet [6]	di → pc		94.4	92.6
ST-SGCN [5]	RGB → skeleton		92.9	-
FPPR-PCD-DenseNet	di → pc	global	86.0	-
FPPR-PCD-DGCNN	di → pc	local	91.6	-
FPPR-PCD [3]	di → pc		93.8	91.7

4.2.2. Comparison at Different Branches with Different Weights

It is also observed from Table 4 that there are differences between the results of the static, dynamic, and scene flow branches of the same method. By comparing the overall accuracy after the aggregation of different branches, the four following points are concluded.

- The accuracy of aggregation is 6.4% to 7.1% higher than that of the static branch. The gain obtained by M-FSS-GR is 7.1%, which is 0.9% higher than the gain from Kinet [1];
- The accuracy of the two methods of multi-stream FSS-GR is 4% higher on dynamic branches than on static branches. In this case, the gain obtained by M-FSS-GR on dynamic branches is 4.4%, which is 0.3% higher than the gain from Kinet [1];
- The overall accuracy of M-FSS-GR is up to 2.7% higher than that of dynamic branches, and the gain is 0.6% higher than the gain from work [1];
- The accuracy of the scene flow branch of T-FSS-GR is about 3.0%.

The first three observations further validate that the proposed method is better than previous methods due to the improvement in the ability to learn spatial information and capture temporal information. However, as described in the last observation, it is inefficient to use scene flow alone in classification. This work attributes this inefficiency to the fact that tracking the movements of each hand joint over very short time intervals allows the weighting of small motions to be increased, and these small motions have less correlation with the type of gesture. However, these tiny motions can be complemented with normal vectors that focus on the coarse-grained motion features, allowing the network to capture 3D motion more completely. Based on the above analysis, this work suspects that the learning of dynamic features should take into account both coarse-grained motion features and fine-grained motion features, and fine-grained motion features should take up a small proportion in gesture recognition.

In order to verify the impacts of the proportion of scene flow branches on the recognition accuracy during aggregation, first the proportion of scene flow branches is set from 0.1 to 0.5 during aggregation, and the spatial branch proportions are 0.5, 0.4, and 0.3. Then, the accuracy of M-FSS-GR and M-FSS-GR' is compared with different proportions, and the

results are shown in Figure 5 and Table 6. The two following points can be observed from Figure 5.

- When the static ratio is 0.3, the performance of M-FSS-GR and M-FSS-GR' is better;
- When the scene flow branch ratio is set to 0.1, 0.2, or 0.3, the model performance is improved. When the static ratio is 0.3, the scene flow branch ratio is 0.3, and when the dynamic ratio is 0.4, the performances of M-FSS-GR and M-FSS-GR' reach their best.

The above results demonstrate that FSS-GR is effective in combining different fine-grained dynamic features.

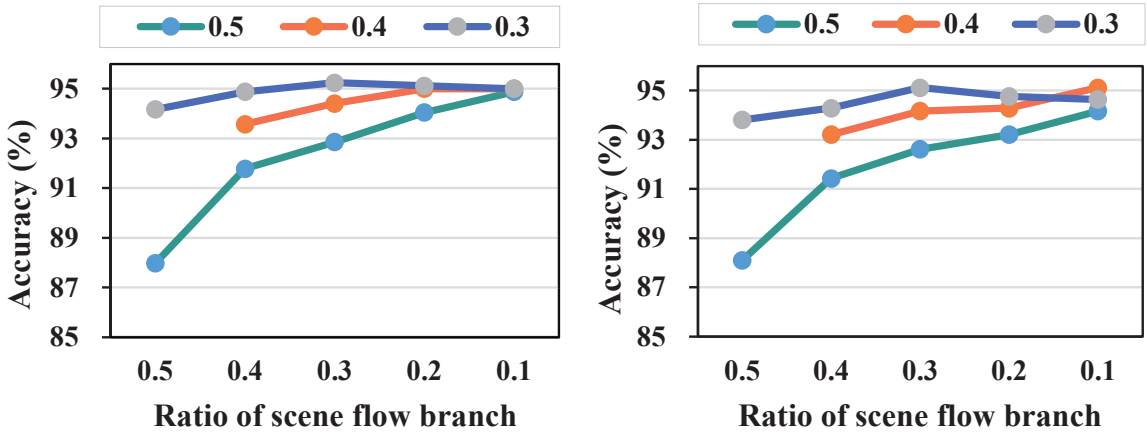

Figure 5. Performance comparison (%) of multi-stream FSS-GR under different scene flow branch ratio on SHREC'17: (**a**) M-FSS-GR; and (**b**) M-FSS-GR'. The x axis is the ratio of scene flow branch, while y axis is the total accuracy (%).

Table 6. Accuracy (%) of multi-stream FSS-GR under different scene flow branch ratio on SHREC'17.

	M-FSS-GR			M-FSS-GR'		
	Static = 0.3	**Static = 0.4**	**Static = 0.5**	**Static = 0.3**	**Static = 0.4**	**Static = 0.5**
flow = 0.5	94.17		87.98	93.81		88.10
flow = 0.4	94.88	93.57	91.79	94.29	93.21	91.43
flow = 0.3	95.24	94.40	92.86	95.12	94.17	92.62
flow = 0.2	95.12	95.00	94.05	94.76	94.29	93.21
flow = 0.1	95.00	95.00	94.88	94.64	95.12	94.17

4.2.3. Comparison on Different Scene Flow Datasets

When using different scene flow datasets, the performance of the same method is shown in Table 7, and the results show the impact of time interval Δt and scene flow metrics on recognition accuracy.

Differences between the performance of M-FSS-GR and M-FSS-GR':

- The accuracy of M-FSS-GR is 1% higher on SHRECsft and SHRECsfe compared to SHRECsfe2 and SHRECsfe3;
- The accuracy of M-FSS-GR' is nearly 0.8% higher on SHRECsfe or SHRECsfe3 compared to SHRECsfe2;
- M-FSS-GR is the best performing method on SHRECsft, SHRECsfe, and SHRECsfe2. Compared to other methods, its accuracy is improved by 0.4 to 0.6. It indicates that the tracking motion of points in a shorter period is beneficial for the network to learn the motion of different fine-grained sizes;
- On SHRECsfe3, M-FSS-GR' has the highest accuracy and outperforms the other two methods by more than 1%.

T-FSS-GR: Compared to Kinet [1], the accuracy of T-FSS-GR is 0.7% higher in SHRECsfe. However, its performance is degraded on the other three datasets in which scene flow is denoted by F^*. And, its performance deteriorates as the time interval Δt increases.

Performance degradation results from the fact that, when fusing different fine-grained motion features, N(16) feature points not only fail to retain the fine-grained dynamic features, but also lose the coarse-grained dynamic features. F^* describes the movement of joints more accurately than F, so the difference between F^* and normal vector is greater, which makes the new fused motion features have a greater shift. This work tries to connect coarse-grained or fine-grained motion features after the new motion features, but the results show that the performance was only improved to the same level as that of SOTA, but the computational cost of the network was already higher than that of the SOTA.

Table 7. Comparison on different scene flow datasets. The results of 28 gestures are reported.

Method	Scene Flow Dataset	Accuracy (%)
M-FSS-GR	SHRECsft	95.0
M-FSS-GR	SHRECsfe	95.2
M-FSS-GR	SHRECsfe2	94.0
M-FSS-GR	SHRECsfe3	93.8
M-FSS-GR′	SHRECsft	94.4
M-FSS-GR′	SHRECsfe	94.8
M-FSS-GR′	SHRECsfe2	94.0
M-FSS-GR′	SHRECsfe3	95.1
T-FSS-GR	SHRECsft	94.5
T-FSS-GR	SHRECsfe	93.9
T-FSS-GR	SHRECsfe2	93.7
T-FSS-GR	SHRECsfe3	93.2

The above results show that the performance of M-FSS-GR is superior to that of T-FSS-GR. This is because it is better to add a scene flow branch than to fuse normal vector and scene flow to form a new motion feature. The two motion features with different fine-grained dimensions lose less information during the training phase.

4.3. Cost Analysis

Table 8 gives the floating-point operations (FLOPs) and the number of parameters (Params) for FSS-GR. From the perspective of Params, the number of parameters of our model (M-FSS-GR, M-FSS-GR′, and T-FSS-GR) is kept at a low level. In particular, the Params of T-FSS-GR is only 1.6 M. Compared with MAE and ST-SGCN, the Params of T-FSS-GR is reduced by 89.3% and 84.6%. This shows that FSS-GR requires less memory for training. The Kinet has the smallest number of parameters, which is 1.5 M. The Params of T-FSS-GR is very close to that of Kinet, which means that FSS-GR has a lower risk of overfitting and has a stronger generalization ability.

Although FSS-GR introduces computationally expensive scene flow estimation, and scene flow estimation is not synchronized with gesture recognition. The estimated scene flow is used as an intermediate synthetic dataset. However, from the perspective of FLOPs, the computational complexity of FSS-GR is high. The FLOPs of M-FSS-GR are 1.9% higher than those of Kinet. The FLOPs of M-FSS-GR′ and T-FSS-GR decrease slightly but remain in the same order of magnitude. This indicates that FSS-GR requires more computing resources and time to recognize gestures.

In terms of recognition performance, the accuracies of the models M-FSS-GR, M-FSS-GR′, and T-FSS-GR are the highest among all models. Although FLOPs are relatively high, the Params is comparatively low. This indicates that these models control the complexity

of the model to a certain extent while maintaining high performance. Compared to Kinet, M-FSS-GR has an improvement in recognition accuracy, and the Params only increases by 0.8M. When comparing M-FSS-GR with other models, it demonstrates an improvement in recognition accuracy, and the Params has decreased significantly. By comparing the FLOPs, Params, and accuracy of different gesture recognition models, we observe that, while maintaining a high recognition accuracy, the Params of FSS-GR is relatively low, which validates the effectiveness of FSS-GR. In particular, the T-FSS-GR achieves a balance between recognition accuracy and computational cost. This indicates the scalability of T-FSS-GR.

In summary, FSS-GR has a high accuracy and a low number of parameters, which is suitable for deployment on equipment with limited resources. However, its computational efficiency needs to be improved. In the future, we will explore lightweight architectures to balance accuracy and FLOPs, which will be beneficial to improve the practicability of FSS-GR.

Table 8. Comparison of FLOPs (floating-point operations) and Params (the number of parameters).

Method	FLOPs	Params
MAE [7]	-	15 M
FPPR-PCD [3]	-	4.6 M
PointLSTM-late [2]	30.6 G	2.2 M
ST-SGCN [5]	-	10.4 M
Kinet [1]	255.4 G	1.5 M
M-FSS-GR	260.3 G	2.3 M
M-FSS-GR'	259.9 G	2.2 M
T-FSS-GR	256.0 G	1.6 M

5. Conclusions

The paper proposes a novel dynamic gesture recognition method FSS-GR that fuses fine-grained features from skeleton-based scene flow and coarse-grained dynamic features. In the past, the point cloud generated from depth images is usually used as the input of gesture recognition. However, some pixels have little correlation with gestures, making the learned features redundant. And, previous methods often feed features within a spatio-temporal neighborhood into complex networks and do not directly track point-by-point correspondences. Scene flow is a powerful tool for extracting real 3D motion. In our paper, the skeletons of a gesture are converted into the pairs of a point clouds to estimate the scene flow. Depending on the time interval and the criterion for measuring the scene flow, four scene flow datasets are generated by a new automatic converter in the preprocessing stage. The learned scene flow and coarse-grained motion features are fused in two different ways. Multi-stream FSS-GR adds a new scene stream branch, while T-FSS-GR fuses coarse-grained features with scene flow in the dynamic branch. Extensive experiments demonstrate that the scene flow estimated in the preprocessing stage is an effective complement with coarse-grained motion features on point clouds. Interestingly, fine-grained motion features take up a small proportion, but have some impacts on recognition accuracy improvement and cannot be ignored. Therefore, our proposed method FSS-GR can take advantage of fine-grained motion features from skeletons and coarse-grained features from deep images together for the best performance gesture recognition over SOTA works. However, each pair of point clouds that evaluates fine-grained dynamic features includes only two frames of spatial information, which makes some of the learned scene flow almost consistent with the static features. In the future, we will design pairs of point clouds including multi-frame skeletons in a short time to learn complex fine-grained features. In addition, we intend to realize the adaptation of Δt to various gesture datasets.

Author Contributions: Conceptualization, Y.L. and J.J.; methodology, Y.L. and J.J.; software, Y.L.; validation, Y.L. and J.J.; formal analysis, Y.L.; investigation, Y.L.; resources, Y.L. and J.J.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, J.J.; visualization, Y.L.; supervision, J.J.; project administration, J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Shanghai Pujiang Program with grant number 21PJD026.

Data Availability Statement: The dataset SHREC'17 used in this study is available at http://www-rech.telecom-lille.fr/shrec2017-hand/ (accessed on 25 January 2025), reference number 10.2312/3dor.20171049 [10]. These data are available in the public domain. The datasets (SHRECsft, SHRECsfe, SHRECsfe2, and SHRECsfe3) presented in this study are available upon request from the corresponding author due to privacy.

Acknowledgments: The authors gratefully acknowledge the financial support from Shanghai Pujiang Program with grant number 21PJD026.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FSS-GR	Fusing Skeleton-based Scene Flow for Gesture Recognition
M-FSS-GR	Multi-stream FSS-GR
T-FSS-GR	Two-stream FSS-GR
SOTA	State-of-the-art
FLOPs	Floating-point operations
Params	The number of parameters

References

1. Zhong, J.X.; Zhou, K.; Hu, Q.; Wang, B.; Trigoni, N.; Markham, A. No Pain, Big Gain: Classify Dynamic Point Cloud Sequences with Static Models by Fitting Feature-level Space-time Surfaces. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 8500–8510. [CrossRef]
2. Min, Y.; Zhang, Y.; Chai, X.; Chen, X. An Efficient PointLSTM for Point Clouds Based Gesture Recognition. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 5760–5769. [CrossRef]
3. Bigalke, A.; Heinrich, M.P. Fusing Posture and Position Representations for Point Cloud-Based Hand Gesture Recognition. In Proceedings of the 2021 International Conference on 3D Vision (3DV), London, UK, 1–3 December 2021; pp. 617–626. [CrossRef]
4. Chen, Y.; Zhao, L.; Peng, X.; Yuan, J.; Metaxas, D.N. Construct Dynamic Graphs for Hand Gesture Recognition via Spatial-Temporal Attention. *arXiv* **2019**, arXiv:1907.08871.
5. Ikne, O.; Slama, R.; Saoudi, H.; Wannous, H. Spatio-Temporal Sparse Graph Convolution Network for Hand Gesture Recognition. In Proceedings of the 2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG), Istanbul, Turkey, 27–31 May 2024; pp. 1–5. [CrossRef]
6. Qiu, F.; Zhou, J.; Peng, D. Dynamic gesture recognition method based on multi-scale feature fusion. In Proceedings of the 2024 5th International Conference on Computer Vision, Image and Deep Learning (CVIDL), Zhuhai, China, 19–21 April 2024; pp. 390–394. [CrossRef]
7. Ikne, O.; Allaert, B.; Wannous, H. Skeleton-Based Self-Supervised Feature Extraction for Improved Dynamic Hand Gesture Recognition. In Proceedings of the 2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG), Istanbul, Turkey, 27–31 May 2024; pp. 1–10. [CrossRef]
8. Shen, J.; Xu, X.; Tan, R.; Karlson, A.; Strasnick, E. Boosting Gesture Recognition with an Automatic Gesture Annotation Framework. In Proceedings of the 2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG), Istanbul, Turkey, 27–31 May 2024; pp. 1–10. [CrossRef]
9. Cheng, J.; Shi, D.; Li, C.; Li, Y.; Ni, H.; Jin, L.; Zhang, X. Skeleton-Based Gesture Recognition with Learnable Paths and Signature Features. *IEEE Trans. Multimed.* **2024**, *26*, 3951–3961. [CrossRef]

10. De Smedt, Q.; Wannous, H.; Vandeborre, J.P.; Guerry, J.; Saux, B.L.; Filliat, D. 3D hand gesture recognition using a depth and skeletal dataset: SHREC'17 track. In Proceedings of the 3Dor'17 Workshop on 3D Object Retrieval, Lyon, France, 23–24 April 2017; pp. 33–38. [CrossRef]
11. Molchanov, P.; Yang, X.; Gupta, S.; Kim, K.; Tyree, S.; Kautz, J. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4207–4215. [CrossRef]
12. Yang, X.; Molchanov, P.; Kautz, J. Making Convolutional Networks Recurrent for Visual Sequence Learning. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6469–6478. [CrossRef]
13. Abavisani, M.; Joze, H.R.V.; Patel, V.M. Improving the Performance of Unimodal Dynamic Hand-Gesture Recognition with Multimodal Training. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 1165–1174. [CrossRef]
14. De Smedt, Q.; Wannous, H.; Vandeborre, J.P. Skeleton-Based Dynamic Hand Gesture Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1206–1214. [CrossRef]
15. Hou, J.; Wang, G.; Chen, X.; Xue, J.H.; Zhu, R.; Yang, H. Spatial-Temporal Attention Res-TCN for Skeleton-Based Dynamic Hand Gesture Recognition. In Proceedings of the Computer Vision—ECCV 2018 Workshops, Munich, Germany, 8–14 September 2018; Proceedings, Part VI; Springer: Berlin/Heidelberg, Germany, 2019; pp. 273–286. [CrossRef]
16. Yan, S.; Xiong, Y.; Lin, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; AAAI'18/IAAI'18/EAAI'18.
17. Núñez, J.C.; Cabido, R.; Pantrigo, J.J.; Montemayor, A.S.; Vélez, J.F. Convolutional Neural Networks and Long Short-Term Memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognit.* **2018**, *76*, 80–94. [CrossRef]
18. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
19. Muthu, S.; Tennakoon, R.; Hoseinnezhad, R.; Bab-Hadiashar, A. A Survey of CNN-Based Techniques for Scene Flow Estimation. *IEEE Access* **2023**, *11*, 99289–99303. [CrossRef]
20. Liu, X.; Qi, C.R.; Guibas, L.J. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 529–537. [CrossRef]
21. Liu, X.; Yan, M.; Bohg, J. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9245–9254. [CrossRef]
22. Min, Y.; Chai, X.; Zhao, L.; Chen, X. FlickerNet: Adaptive 3D Gesture Recognition from Sparse Point Clouds. In Proceedings of the British Machine Vision Conference, Cardiff, UK, 9–12 September 2019.
23. Lang, I.; Aiger, D.; Cole, F.; Avidan, S.; Rubinstein, M. SCOOP: Self-Supervised Correspondence and Optimization-Based Scene Flow. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 5281–5290. [CrossRef]
24. Shen, Y.; Hui, L.; Xie, J.; Yang, J. Self-Supervised 3D Scene Flow Estimation Guided by Superpoints. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 5271–5280. [CrossRef]
25. Fan, H.; Yu, X.; Yang, Y.; Kankanhalli, M. Deep Hierarchical Representation of Point Cloud Videos via Spatio-Temporal Decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 9918–9930. [CrossRef]
26. Liu, H.; Lu, T.; Xu, Y.; Liu, J.; Li, W.; Chen, L. CamLiFlow: Bidirectional Camera-LiDAR Fusion for Joint Optical Flow and Scene Flow Estimation. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5781–5791. [CrossRef]
27. Wang, Z.; Wei, Y.; Rao, Y.; Zhou, J.; Lu, J. 3D Point-Voxel Correlation Fields for Scene Flow Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13621–13635. [CrossRef] [PubMed]
28. Li, Z.; Yang, X.; Zhang, J. GAMAFlow: Estimating 3D Scene Flow via Grouped Attention and Global Motion Aggregation. In Proceedings of the ICASSP 2024—2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 3955–3959. [CrossRef]
29. Luo, C.; Yang, X.; Yuille, A. Self-Supervised Pillar Motion Learning for Autonomous Driving. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 3182–3191. [CrossRef]

30. Andreas Baur, S.; Josef Emmerichs, D.; Moosmann, F.; Pinggera, P.; Ommer, B.; Geiger, A. SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 13106–13116. [CrossRef]
31. Puy, G.; Boulch, A.; Marlet, R. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XXVIII; Springer: Berlin/Heidelberg, Germany, 2020; pp. 527–544. [CrossRef]
32. Lang, I.; Ginzburg, D.; Avidan, S.; Raviv, D. DPC: Unsupervised Deep Point Correspondence via Cross and Self Construction. In Proceedings of the 2021 International Conference on 3D Vision (3DV), London, UK, 1–3 December 2021; pp. 1442–1451. [CrossRef]
33. Mayer, N.; Ilg, E.; Häusser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; Brox, T. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4040–4048. [CrossRef]
34. Menze, M.; Geiger, A. Object scene flow for autonomous vehicles. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3061–3070. [CrossRef]
35. Menze, M.; Heipke, C.; Geiger, A. Joint 3D Estimation of vehicles and scene flow. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *II-3/W5*, 427–434. [CrossRef]
36. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning Spatiotemporal Features with 3D Convolutional Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chille, 7–13 December 2015; pp. 4489–4497. [CrossRef]
37. Li, Y.; Miao, Q.; Tian, K.; Fan, Y.; Xu, X.; Li, R.; Song, J. Large-Scale Gesture Recognition with a Fusion of RGB-D Data Based on Saliency Theory and C3D Model. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 2956–2964. [CrossRef]
38. Miao, Q.; Li, Y.; Ouyang, W.; Ma, Z.; Xu, X.; Shi, W.; Cao, X. Multimodal Gesture Recognition Based on the ResC3D Network. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 3047–3055. [CrossRef]
39. Jin, B.; Ma, X.; Zhang, Z.; Lian, Z.; Wang, B. Interference-Robust Millimeter-Wave Radar-Based Dynamic Hand Gesture Recognition Using 2-D CNN-Transformer Networks. *IEEE Internet Things J.* **2024**, *11*, 2741–2752. [CrossRef]
40. Fan, H.; Yang, Y.; Kankanhalli, M. Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 14199–14208. [CrossRef]
41. Fan, H.; Yang, Y.; Kankanhalli, M. Point Spatio-Temporal Transformer Networks for Point Cloud Video Modeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 2181–2192. [CrossRef] [PubMed]
42. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the NIPS'17: 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, Long Beach, CA, USA, 4–9 December 2017; pp. 5105–5114.
43. Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *Proceedings of the Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 20–36.
44. Simonyan, K.; Zisserman, A. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of the Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K., Eds.; Curran Associates, Inc.: Cambridge, MA, USA, 2014; Volume 27.
45. Behl, A.; Jafari, O.H.; Mustikovela, S.K.; Alhaija, H.A.; Rother, C.; Geiger, A. Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios? In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2593–2602. [CrossRef]
46. Luthra, A.; Gantha, S.S.; Song, X.; Yu, H.; Lin, Z.; Peng, L. Deblur-NSFF: Neural Scene Flow Fields for Blurry Dynamic Scenes. In Proceedings of the 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2024; pp. 3646–3655. [CrossRef]
47. Park, K.; Sinha, U.; Barron, J.T.; Bouaziz, S.; Goldman, D.B.; Seitz, S.M.; Martin-Brualla, R. Nerfies: Deformable Neural Radiance Fields. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 5845–5854. [CrossRef]
48. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361. [CrossRef]
49. Li, X.; Jiao, Z.; Zhang, X.; Zhang, L. Visual SLAM in Dynamic Environments Based on Object Detection and Scene Flow. In Proceedings of the 2023 IEEE International Conference on Mechatronics and Automation (ICMA), Harbin, Heilongjiang, China, 6–9 August 2023; pp. 2157–2162. [CrossRef]

50. Liu, J.; Liu, Y.; Wang, Y.; Prinet, V.; Xiang, S.; Pan, C. Decoupled Representation Learning for Skeleton-Based Gesture Recognition. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 5750–5759. [CrossRef]
51. Song, J.H.; Kong, K.; Kang, S.J. Dynamic Hand Gesture Recognition Using Improved Spatio-Temporal Graph Convolutional Network. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 6227–6239. [CrossRef]
52. Liu, J.; Wang, X.; Wang, C.; Gao, Y.; Liu, M. Temporal Decoupling Graph Convolutional Network for Skeleton-Based Gesture Recognition. *IEEE Trans. Multimed.* **2024**, *26*, 811–823. [CrossRef]
53. Narayan, S.; Mazumdar, A.P.; Vipparthi, S.K. SBI-DHGR: Skeleton-based intelligent dynamic hand gestures recognition. *Expert Syst. Appl.* **2023**, *232*, 120735. [CrossRef]