IDEA-Net: Dynamic 3D Point Cloud Interpolation via Deep Embedding Alignment (Supplementary Materials)

Yiming Zeng[1] Yue Qian[1] Qijian Zhang[1] Junhui Hou[1†] Yixuan YUAN[1] Ying He[2]
[1] City University of Hong Kong [2] Nanyang Technological University
ym.zeng@my.cityu.edu.hk, jh.hou@cityu.edu.hk

In this supplementary material, we first provide a demo video to comprehensively show the interpolation results in Section 1. Then, we elaborate on the implementation details and training strategy of our IDEA-Net in Section 2. We provide the details of the dataset used in our experiments in Section 3. In Section 4, we visualize the learned point-wise temporal consistency (i.e., matrix $\mathbf{A}$) by the dual- and single- branch models. In Section 5, we provide more visual comparisons.

1. Video Demo

We refer the readers to the video demo at https://github.com/ZENGYIMING-EAMON/IDEA-Net.git, where we show the interpolation results ($k_{\text{train}} = k_{\text{test}} = 3$) of our method and the compared methods.

2. Implementation Details of IDEA-Net

We provide the implementation details in Table 1. We omit the BatchNorm layers and ReLU activation functions that are carried in each Conv layer, except for the Conv layer at the end of each module. In the feature representation module, we replace ReLU with LeakyReLU (negative slop is 0.2).

During the training process, we use the EMD loss for both branches and take the average of them as the final training loss. We use Pytorch [2] to implement our model on a single GPU (GeForce RTX 3090). We use Adam with the initial learning rate 0.0001 to optimize the model. For all the models, we train 1000 epochs with the batch size set to 14. For the feature embedding module in the table, we follow the original settings of DGCNN [3].

3. Details of Constructed Datasets

In this section, we provide more details for the DHB dataset and DFAUST dataset used in our paper. As shown in Table 2, both datasets are composed of 14 sequences, in which the four sequences (*Longdress, Loot, Redandblack, Soldier*) are 3D point clouds, and others are 3D meshes. We uniformly sampled 1024 points from each individual frame to construct dynamic point cloud sequences. For the DHB dataset, we used the top 8 sequences to form the training set and the remaining as the testing set. For the DFAUST dataset, we used used the top 11 sequences for training and the rest for testing. All the datasets listed in the table will be released along with our code.

4. Visual Illustration of the Learned Point-wise Temporal Consistency

As mentioned in Section 4.4(f) of the manuscript, we conducted the ablation study to demonstrate the advantage of the dual-branch design over the single-branch design. Here, we also visualize the learned point-wise temporal consistency, (i.e., matrix $\mathbf{A}$) to illustrate the difference between the two designs. Keeping consistent with other ablation studies, we used the mixed data training mechanism, as mentioned in the last paragraph of Section 4.2 of the manuscript, to train and test our model on the DHB dataset and we fixed $k_{\text{test}} = 3$. We fed a random sample pair of the sequence *swing* into the network and obtained matrix $\mathbf{A}$. As shown in Fig. 1, it can be seen that the learned matrix $\mathbf{A}$ by the single-branch model has many values distributed in the same column (Fig.1b), meaning that multiple points of $\mathbf{P}_0$ are aligned to an identical point of $\mathbf{P}_1$. However, such an observation rarely appear in the matrix $\mathbf{A}$ (Fig.1a) by the dual-branch model, which is credited to the

Table 1. Implementation details of our network structure. The input and output dimensions are in parentheses.

Module	Layer / Operation	Output
Feature Embedding	getGraphFeat (3, 6), Conv2d (6, 64), MaxPooling	x1
	getGraphFeat (64, 128), Conv2d (128, 64), MaxPooling	x2
	getGraphFeat (64, 128), Conv2d (128, 128), MaxPooling	x3
	getGraphFeat (128, 256), Conv2d (256, 256), MaxPooling	x4
	Concat (x1+x2+x3+x4, 512), Conv1d (512, 512)	x5
	MaxPooling	x6
	AvgPooling	x7
	Concat (x6+x7, 1024)	y1
	Concat(x5+y1, 1536), MLP(1536, 512, 256, 128)	y2
Temporal Consistency	$\mathbf{P_0}, \mathbf{P_1}$→shared feature embedding	$\mathbf{F_0}, \mathbf{F_1}$
	$\mathbf{F_0}, \mathbf{F_1}$ →inverse distance	$\widetilde{\mathbf{A}}$
	$\widetilde{\mathbf{A}}$ → Conv1d (1024, 1024 + 128)	-
	→ row normalization	$\mathbf{A}$
Trajectory Compensation	$\mathbf{P_0}, \mathbf{P_1}, \mathbf{F_0}, \mathbf{F_1}$ → aligned by $\mathbf{A}$	-
	→ linear interpolation	$\mathbf{P}_{0\to t}, \mathbf{F}_{0\to t}, \mathbf{P}_{1\to t}, \mathbf{F}_{1\to t}$
	$\mathbf{F}_{0\to t}, \mathbf{F}_{1\to t}$ → {Conv1d(1024+128, 1152, 576, 288, 3), Tanh}	$\mathbf{\Delta}_{0\to t}, \mathbf{\Delta}_{1\to t}$
	$\mathbf{\Delta}_{0\to t}, \mathbf{\Delta}_{1\to t}$ → compensate $\mathbf{P}_{0\to t}, \mathbf{P}_{1\to t}$	$\mathbf{O}_{0\to t}, \mathbf{O}_{1\to t}$

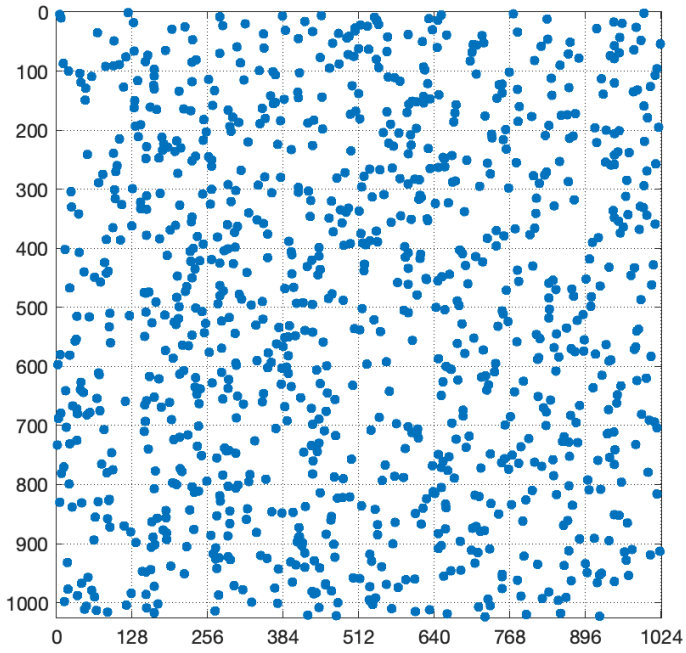
Table 2. Details of the DHB dataset and DFAUST dataset in our experiment.

DHB sequences	# frames	DFAUST sequences	# frames
Bouncing	175	chicken_wings	216
Crane	175	hips	697
Handstand	175	jiggle_on_toes	240
Jumping	150	jumping_jacks	461
March_1	250	knees	500
March_2	250	light_hopping_loose	234
Samba	175	light_hopping_stiff	214
Squat_1	250	one_leg_jump	541
Squat_2	250	one_leg_loose	264
Swing	150	punching	303
Longdress	300	running_on_spot	331
Loot	300	shake_arms	230
Redandblack	300	shake_hips	243
Soldier	300	shake_shoulders	255

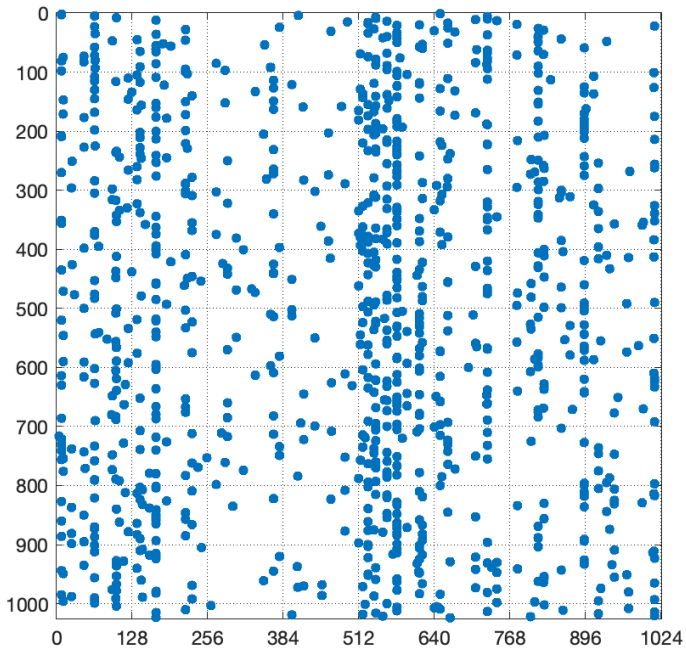
regularization effect of the dual-branch design Note that the ground-truth point-wise consistency is not available, and thus we cannot quantitatively measure the accuracy of $\mathbf{A}$.

5. More Visual Results

In this section, we illustrate more visual comparisons with the state-of-the-art flow-based method PointINet [1] on the DHB dataset. As shown in Figs. 2, 3, and 4, we provide results of two methods on the sequences named *swing*, *soldier* and *squat_2*, respectively. Both methods were evaluated under the setting $k_{\text{train}} = 3$ and $k_{\text{test}} = 3$. It can be seen that for the sequence with large motion and rotations, i.e., *swing* (Fig. 2), PointINet [1] produces broken limbs, which do not appear in our method. For the sequences with smooth motion, i.e., *soldier* (Fig. 3) and *squat_2* (Fig. 4), our method produces fewer artifacts and holes than PointINet [1].

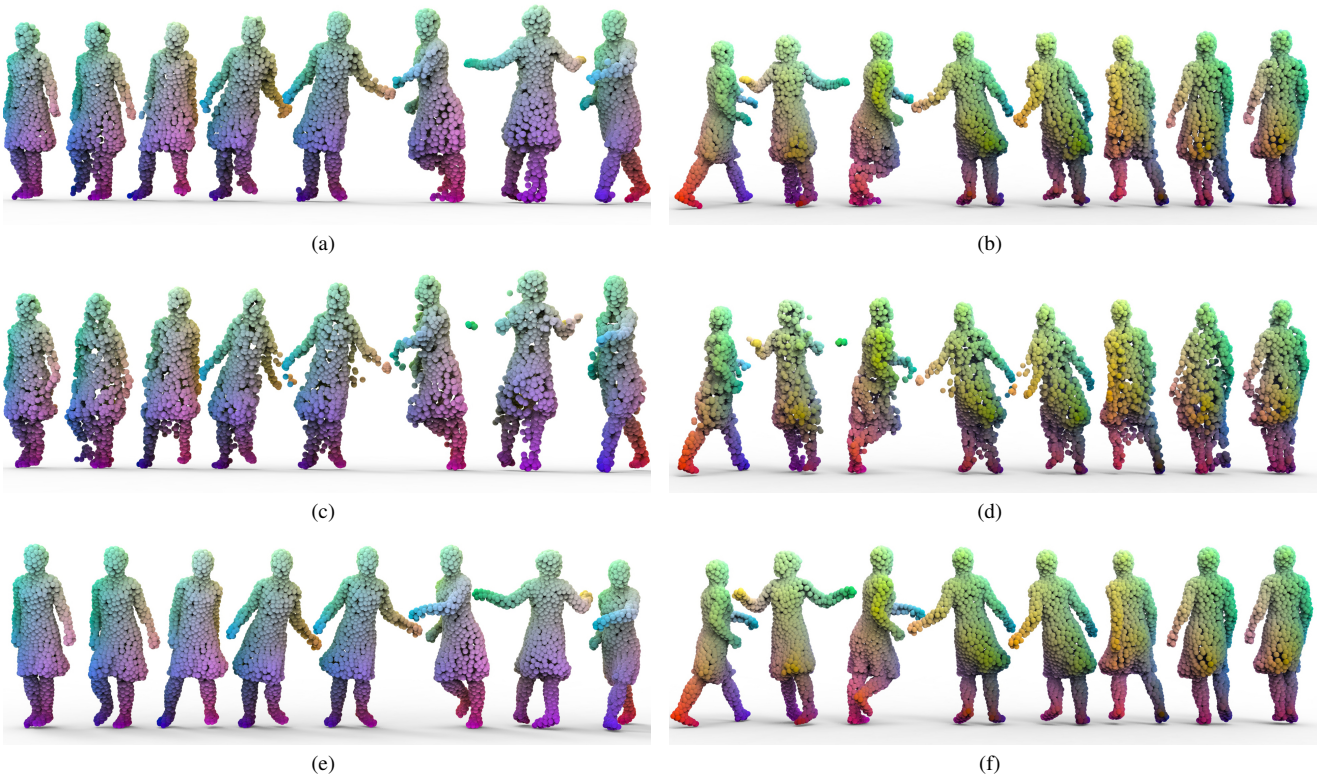


(a)



(b)

Figure 1. Visual illustration of learned $\mathbf{A}$. (a) the dual-branch design. (b) the single-branch design.



(a) (b) (c) (d) (e) (f)

Figure 2. Visual comparisons of the interpolated frames on the test sequence *Swing* by our method (the top row), PointINet (the middle row), and the ground-truth (the bottom row). (a), (c) and (e): front views; (b), (d) and (f): back views.

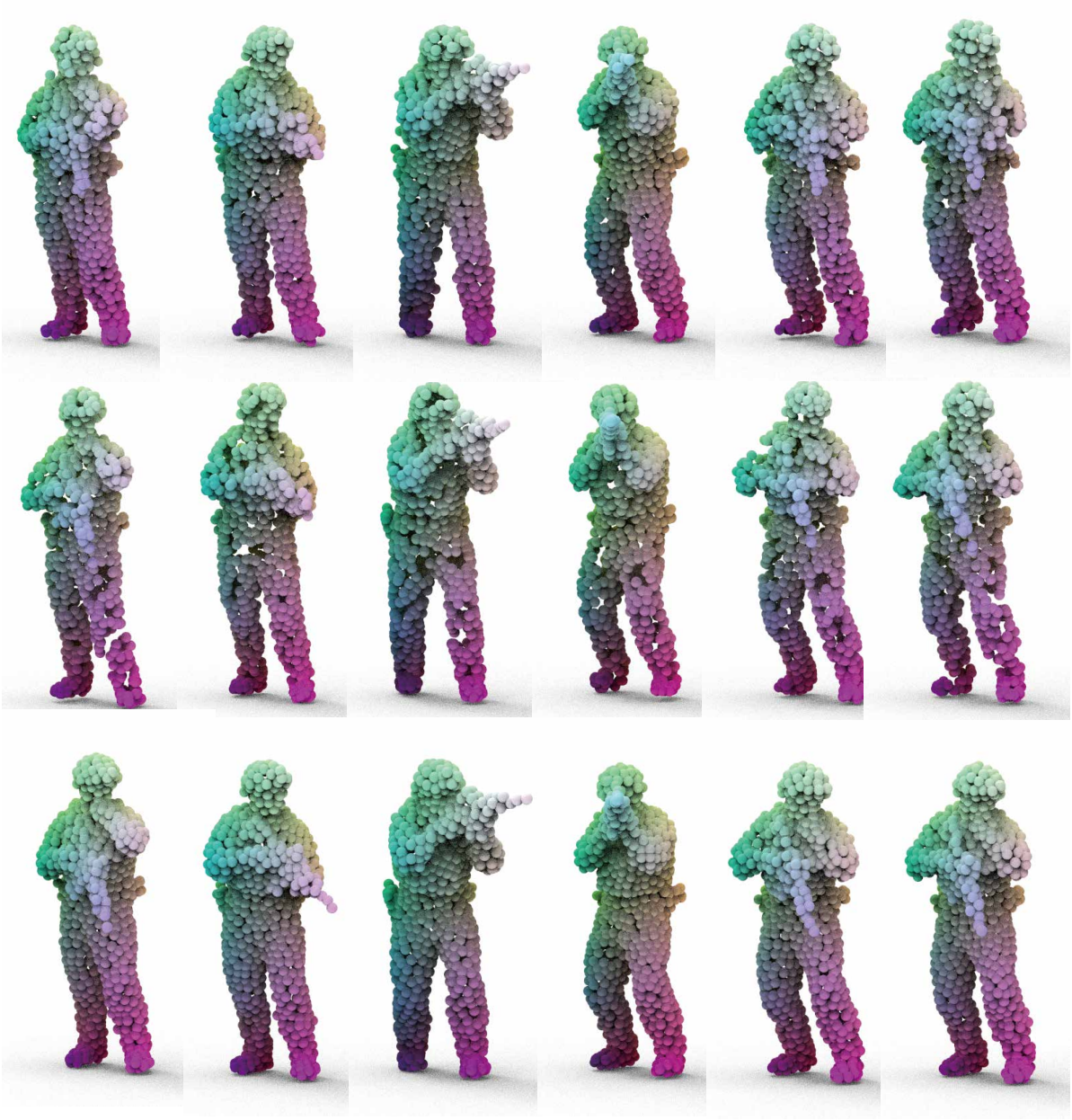


Figure 3. Visual comparisons of the interpolated frames on the test sequence *soldier* by our method (the top row), PointINet (the middle row), and the ground truth (the bottom row).

Figure 4. Visual comparisons of the interpolated frames on the test sequence *squat* by our method (the top row), PointINet (the middle row), and the ground truth (the bottom row).

References

[1] Fan Lu, Guang Chen, Sanqing Qu, Zhijun Li, Yinlong Liu, and Alois Knoll. Pointinet: Point cloud frame interpolation network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 1

[3] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019. 1