4DComplete: Non-Rigid Motion Estimation Beyond the Observable Surface Supplementary Material

A. Network Details

Fig. 1 shows the details of building blocks of 4DComplete. $\oplus$ denotes addition. Convolution, SubmanifoldConvolution, FullyConvolutionalNet, and BatchNormReLU are functions from the SparseConvNet lib (https://github.com/facebookresearch/SparseConvNet.git).

B. Ablation study

Voxel resolution & depth of network (R1). We observed a performance drop in shape estimation when reducing the training volume resolution from $96 \times 96 \times 128$ to $64 \times 64 \times 128$, but with faster training; see Tab. 1.

volume res.	voxel size	SDF error (l_1)	seconds/train-iter.
$96 \times 96 \times 128$	$1.0cm$	**0.53**	2.4
$64 \times 64 \times 128$	$1.0cm$	0.58	**1.8**

Table 1. Ablation study of training volume resolution.

C. Deformingthings4D dataset details

Tab. 2 shows the total number of objects, animations, and frames of each category in our Deformingthings4D dataset. Tab. 3 shows the overview of existing synthetic dynamic dataset. Fig. 2 shows the rendered data for the Mixamo "Aiming Gun" sequence, including color image, depth image, and inter-frame scene flow.

D. Scene Flow estimation for the visible surface

We use FlowNet3D [2] to estimate the scene flow between two consecutive RGB-D frames. As shown in Fig. 3, FlowNet3D directly operates on point clouds. It learns to predict the scene flow as translational motion vectors for each point of the first frame. Fig. 4 shows an example of running FlowNet3D on a pair of point clouds from real-world RGB-D images. Fig. 5 shows the results of scene flow estimation on a real-world RGB-D video recorded from an Azure Kinect camera.

Categories	# Objects	# Animations	# Frames
Man	20	56	8069
Mutant	17	34	4600
Woman	16	48	5096
Zoombie	12	36	2200
Child	11	26	1270
Dragon	10	158	8390
Dear	8	179	10600
Bird	8	71	5000
Fox	5	146	8187
Horse	4	50	3396
Boar	4	180	9654
Moose	4	191	9720
Bear	4	213	10974
Cow	3	50	2014
Wolf	3	149	9060
Rabbit	3	104	6786
Whale	1	5	355
Panthera Onca	1	2	51
Dinosaur	1	3	207
Leopard	1	8	441
Elephant	1	3	198
Tiger	1	15	3479
Racoon	1	25	4448
Puma concolor	1	20	4577
Goat	1	9	700
Pig	1	6	400
Sheep	1	6	665
Crocodile	1	11	1255
Hippo	1	8	768
Rhino	1	9	330
Zebra	1	7	479

Table 2. The number of objects, animations, and frames of each category in our dataset.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012. 2

[2] X. Liu, C. R. Qi, and L. J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, pages 529–537, 2019. 1, 4

[3] 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 *CVPR*, pages 4040–4048, 2016. 2

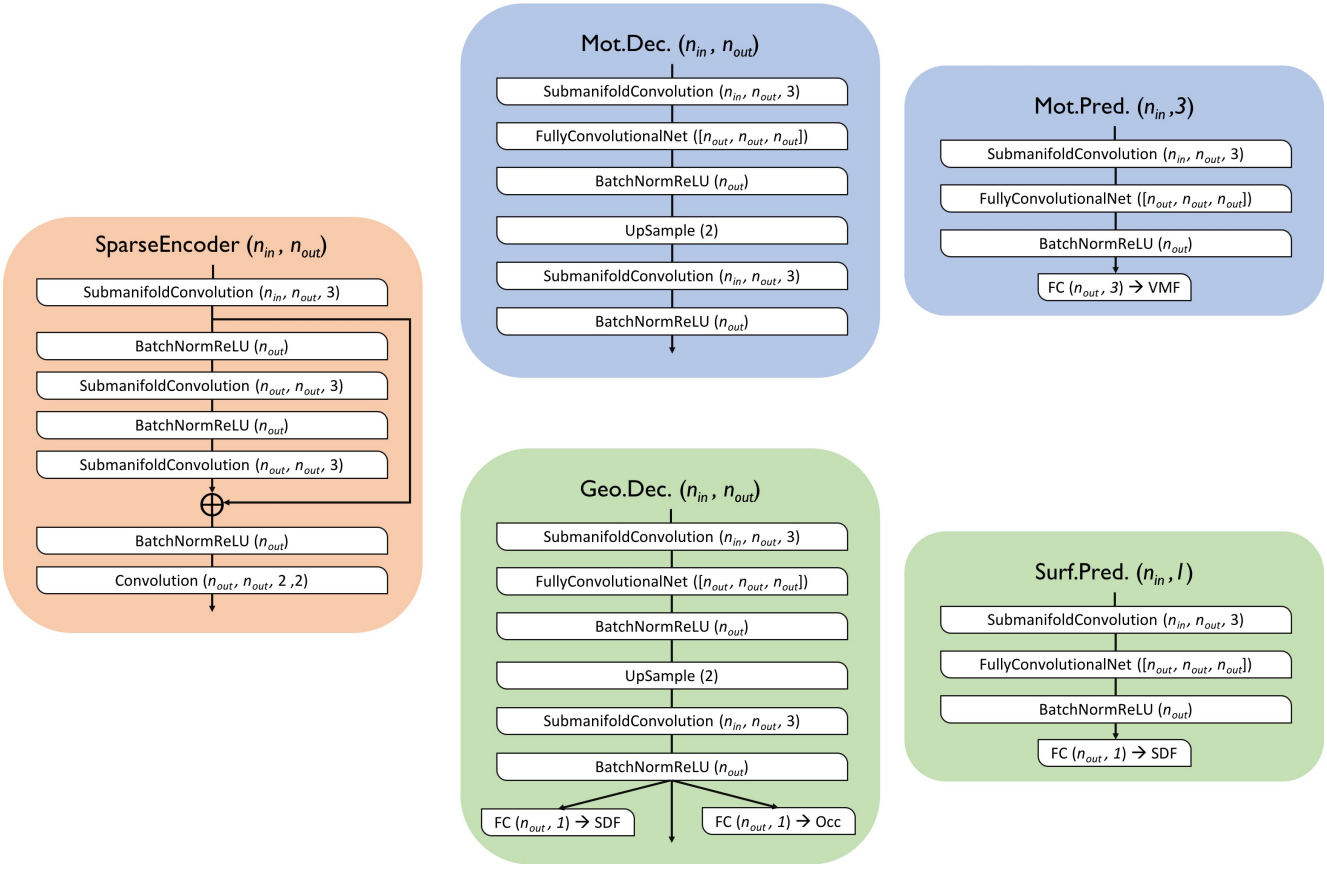

Figure 1. Network building blocks of 4DComplete architecture. $\oplus$ denotes addition. Convolution, SubmanifoldConvolution, FullyConvolutionalNet, and BatchNormReLU are functions from the SparseConvNet lib (https://github.com/facebookresearch/SparseConvNet.git)

Dataset	# Deformable Objects	# Categories	# Frames	Motion	Shape
Sintel [1]	15	3	1,064	Partial	Partial
FlyingThings3D [3]	0	–	21,818	Partial	Partial
Monkka [3]	1	1	8,591	Partial	Partial
DeformingThings4D (Ours)	147	31	123369	Complete	Complete

Table 3. Overview of synthetic dynamic 4D datasets.

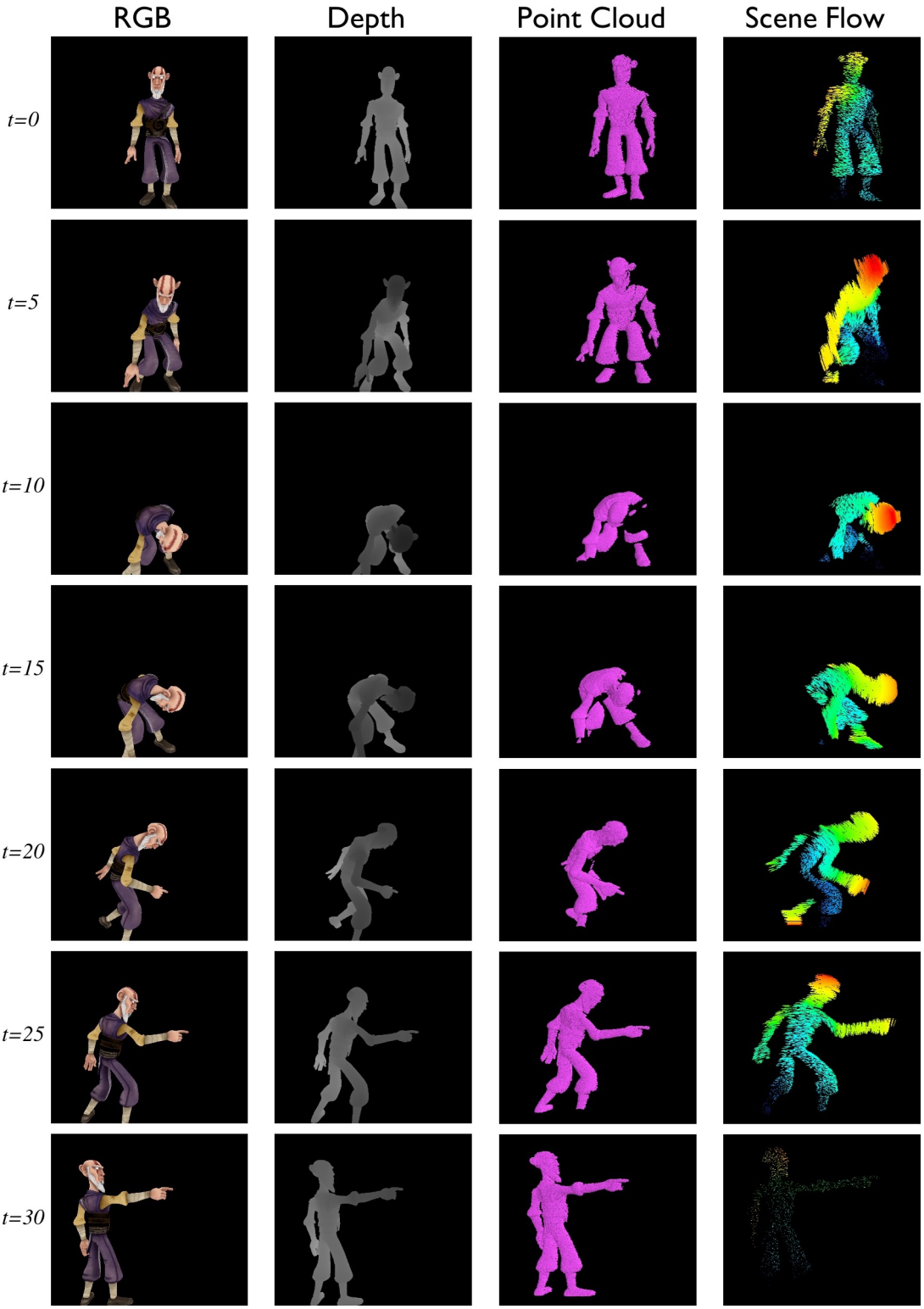

Figure 2. Rendered data for the Mixamo "Aiming Gun" sequence. From left to right: color image, depth image, point cloud (re-projected from depth image), and inter-frame scene flow. The visualization of scene flow and a point cloud is done using Mayavi (https://docs.enthought.com/mayavi/mayavi/).

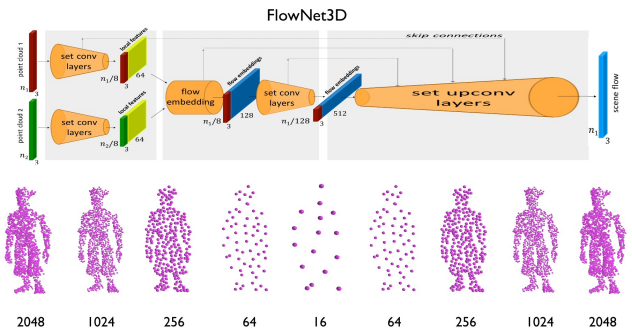

Figure 3. The network architecture of Flownet3D [2]. Given two frames of point clouds, the network learns to predict the scene flow as translational motion vectors for each point of the first frame. The input pointsets are subsampled to 2048 points. And through each encoder/decoder layer, it down-/up-sample the number of points by a factor of 2 or 4. The lower part shows an example of the point sampling through the layers. See the original paper [2] for more details.

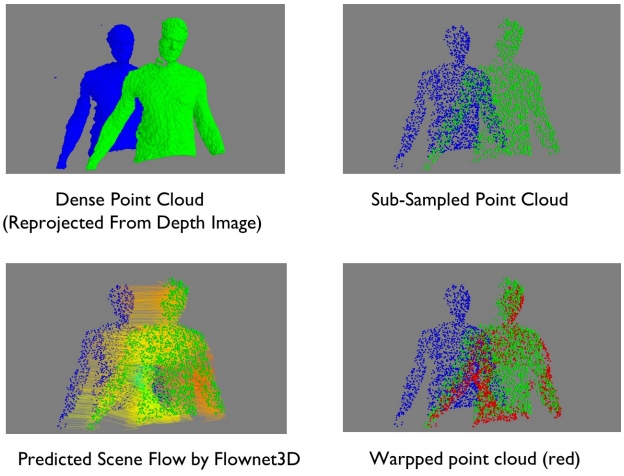

Figure 4. Example of running FlowNet3D on a pair of point clouds from real-world RGB-D images. Top Left: the dense point cloud that is reprojected from the depth images. Top Right: the sub-sampled point cloud input to FlowNet3D, each has 2048 points. Bottom Left: the estimated scene flow vectors which are visualized by the arrows. Bottom Right: warping the source point cloud to the target using the estimated scene flow.

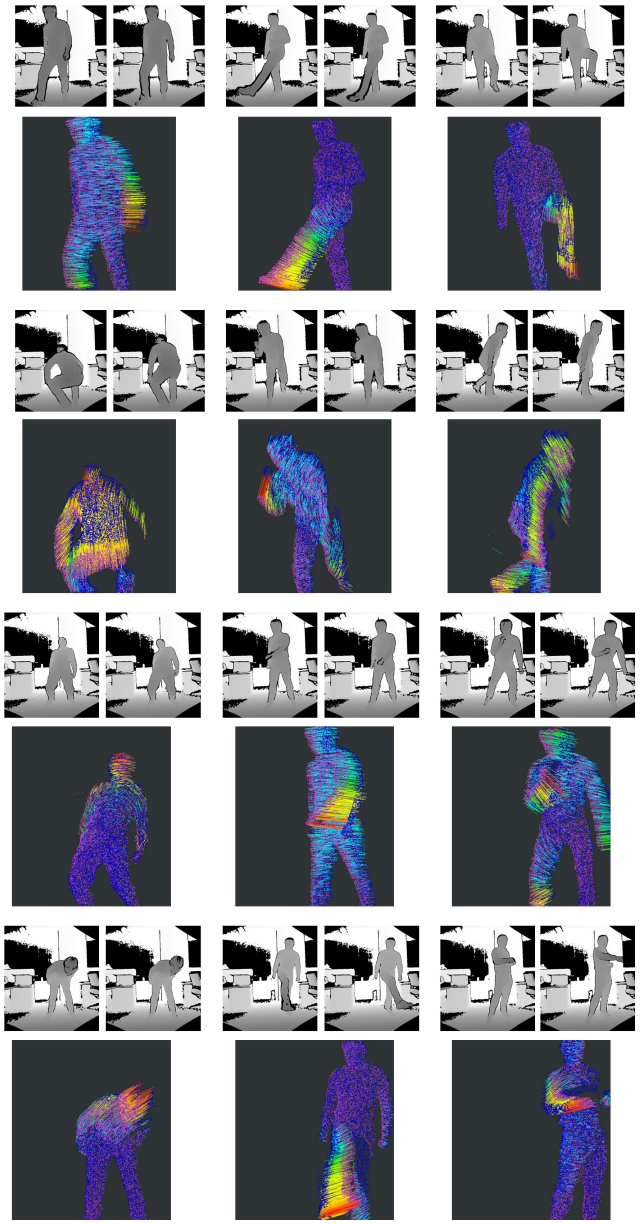

Figure 5. Scene flow estimation results for a real-world RGB-D sequence. The model is FlowNet3D [2] trained on DeformingThings4D dataset. The RBD-G sequence is captured using Azure Kinect. The upper row shows consecutive RGB-D input frames. The lower row shows the scene flow vectors between the two frames, the point cloud sampled from the source frame (given in pink color), and the point cloud sampled from the target frame (given in blue color).