Supplementary Material for
Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics

Michael Niemeyer[1] Lars Mescheder[1] Michael Oechsle[1,2] Andreas Geiger[1]
[1]Autonomous Vision Group, MPI for Intelligent Systems and University of Tübingen
[2]ETAS GmbH, Bosch Group, Stuttgart
{firstname.lastname}@tue.mpg.de

Abstract

*In this **supplementary document**, we first give a detailed overview of our architectures, baseline implementations and training procedure in Section 1. Details regarding the data generation and preprocessing can be found in Section 2. Finally, we provide additional experimental results, both qualitatively and quantitatively in Section 3.*

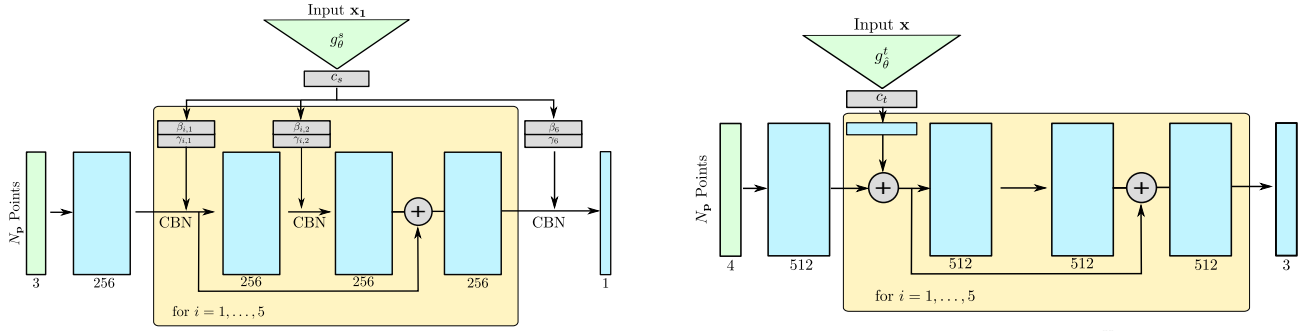
1. Implementation

In this section, we first provide detailed descriptions of the architectures we used for the occupancy and velocity networks as well as the spatial and temporal encoders. Next, we explain implementation details for the baselines and details regarding the training process. Finally, we discuss details regarding mini-batch processing for the forward and backward flow.

1.1. Architectures

1.1.1 Occupancy and Velocity Networks

We use similar architectures for the occupancy and velocity networks $f_\theta^{\mathbf{x}}$ and $\mathbf{v}_{\hat{\theta}}^{\mathbf{x}}$ in all experiments (see Fig. 1). The inputs for both networks are the latent codes c_s and c_t provided by the spatial and temporal encoders g_θ^s and $g_{\hat{\theta}}^t$ as well as N_p points. For both networks, we use five ResNet [6] blocks, each consisting of two fully connected layers with skip connections and ReLUs [5] as activation functions. For the occupancy network, we adopt the conditional batch normalization (CBN) scheme from [9] to inject the latent code c_s. As we experienced difficulties with CBN and the ODE Solver, we pass the latent code c_t



(a) Occupancy Network $f_\theta^{\mathbf{x}}$

(b) Velocity Network $\mathbf{v}_{\hat{\theta}}^{\mathbf{x}}$

Figure 1: **Occupancy and Velocity Network Architectures.** This figure shows the architectures of the occupancy network $f_\theta^{\mathbf{x}}$ and the velocity network $\mathbf{v}_{\hat{\theta}}^{\mathbf{x}}$. Fig. 1a is adopted from [9] and slightly adjusted to our setting. In both figures, light green shows input, cyan fully connected layers, and grey other operations.

provided by the temporal encoder g_θ^t through a fully connected layer and add the result to the activation before every ResNet block for the velocity network. While the occupancy network outputs an occupancy probability for each of the N_p points, the velocity field returns a 3D motion vector for every point in space and time.

1.1.2 Spatial and Temporal Encoders

We use task-specific architectures for the spatial and temporal encoders g_θ^s and g_θ^t (see Fig. 2). For point cloud completion, we use ResNet variants of PointNet [12] for both encoder networks. For the temporal encoder, we concatenate the L input point clouds along the last dimension and adjust the input dimension of the PointNet to $3L$ (Fig. 2a). As the spatial encoder only encodes the first input point cloud, we can keep the input dimension of 3. Similarly to [9], we use four ResNet blocks with additional pooling and expansion layers to enable inter-layer communication. The final feature vectors of dimension 512 are obtained by passing the output of the last ResNet block through a max-pooling operation and a 512-dimensional fully connected layer.

For reconstruction from image sequences, we use a 3D convolutional network with 6 convolutional layers as the temporal encoder (Fig. 2b). For the spatial encoder, we use a pre-trained ResNet-18 [6]. In both networks, we add a final fully connected layer to obtain the final 256-dimensional feature vectors c_t and c_s.

For the correspondence and interpolation experiments, we only use the temporal encoder as no spatial reconstruction has to be performed. We use a single PointNet encoder for both the start and the end point cloud and concatenate the final outputs (Fig. 2c). The weights are shared in siamese fashion. We choose the dimension of the concatenated latent code c_t to be 512. As we use two corresponding point clouds as input for the interpolation task, we can either use the same encoder or the temporal encoder from the point cloud completion experiment with $L = 2$. Again, we choose the dimension of the final output to be 512.

1.2. Baselines

In all reconstruction experiments, we compare against ONet 4D, the 4D variant of Occupancy Networks (ONet) [9]. We keep the architecture from the original publication except that we adjust the input dimension from 3 to 4. This way, we are able to use points sampled in space *and* time as input.

In addition, we compare in all reconstruction experiments against PSGN 4D, the 4D variant of Point Set Generation Networks [4]. We adopt a network architecture consisting of four fully connected layers with hidden dimension of size 512 and ReLUs as activations as it was shown to improve results compared against the more complex 2-branch version from the original publication [9]. We adjust the output dimension from 1024×3 to $1024 \times L \times 3$ because we are interested in producing trajectories of 3D points instead of single 3D points. PSGN is trained by minimizing the Chamfer distance between the predicted and target points, and there are two ways to transfer this to the temporal domain: a) We calculate the Chamfer distances in $\mathbb{R}^3$ for all L time steps individually and sum over them or b) we reshape the predicted points to dimension $3L$ and calculate the Chamfer distance for these predicted and the ground truth trajectories in $\mathbb{R}^{3L}$. Note that while the first variant does not use any correspondences over time during training, the second variant does. To distinguish between the two variants, we name the latter one PSGN 4D (w/ correspond.). We compare against PSGN 4D in all reconstruction experiments and against PSGN 4D (w/ correspond.) in both experiments on the D-FAUST dataset.

1.3. Training Details

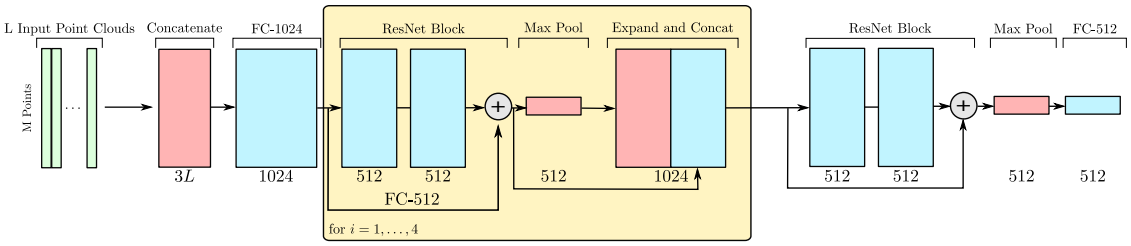
For all experiments, we use the Adam optimizer [8] with learning rate $\gamma = 10^{-4}$ and train with a batch size of 16. We evaluate all models on the validation set every 1,200 iterations during training. To provide a fair comparison in the final evaluation on the test set, we use the model state which performed best on the validation set.

For solving the differential equations, we use the Dormand–Prince method "dopri5" [3] with relative and absolute error tollerances of 10^{-3} and 10^{-5}, respectively.

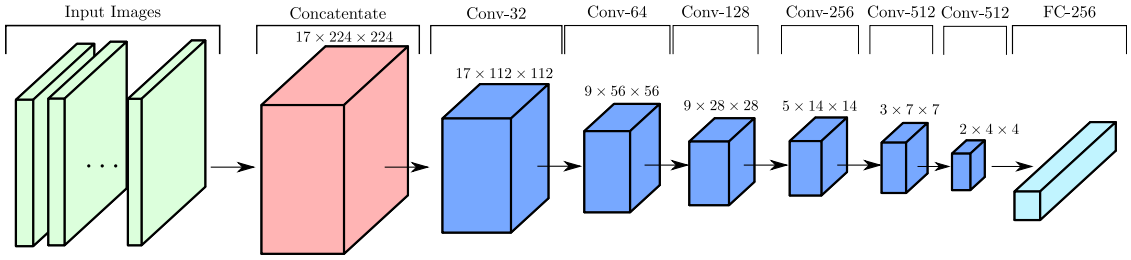
For every training example, we sample a fixed number of points in space and time to calculate the loss. For the reconstruction loss $\mathcal{L}_{recon}$, we sample 512 points at time 0 and 512 points at a random time step $t > 0$. All points at the respective time are sampled uniformly in the bounding volume of the 3D shapes. For the correspondence loss $\mathcal{L}_{corr}$, we sample 100 trajectories each consisting of L points in $\mathbb{R}^3$. The start points of the trajectories are uniformly sampled on the start mesh.

We implemented the OFlow variants and all baselines in PyTorch 1.0.[1]

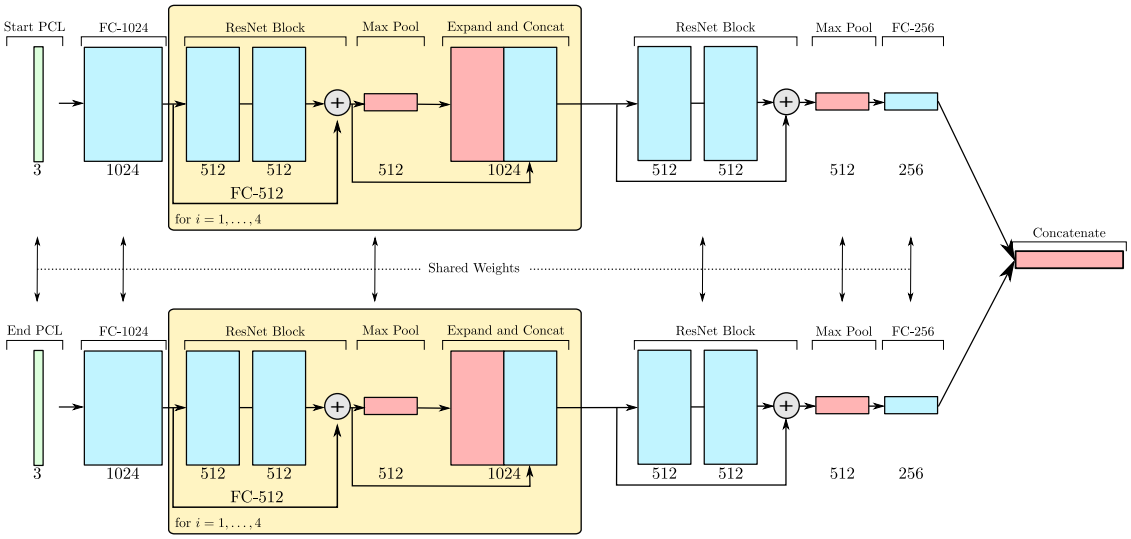
[1] https://pytorch.org/



(a) **Point cloud completion.** We concatenate the L noisy input point clouds each consisting of M points along the last dimension. This tensor is passed through four ResNet blocks with skip connections and additional max-pooling and expansion layers. The final 512-dimensional feature vector is obtained by passing the output of the final ResNet block through a max-pooling operation and a fully connected layer.

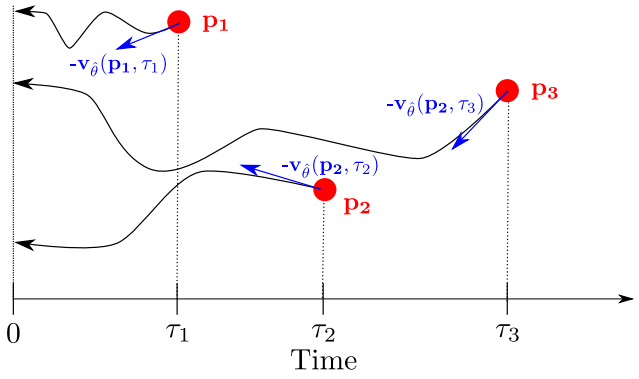


(b) **4D reconstruction from Image Sequences.** The $L = 17$ input RGB images are concatenated to obtain a tensor of size $3 \times 17 \times 224 \times 224$. We pass the tensor through six convolutional layers which downsample the input while increasing the number of feature maps. The final feature vector is obtained by passing the output through a 256-dimensional fully connected layer.

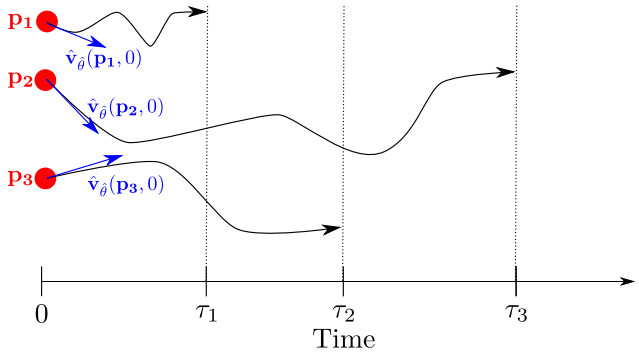


(c) **Shape Matching.** Both the start and the end point cloud is passed through the same PointNet in a siamese fashion. The networks consist of four ResNet blocks with additional max-pooling and expansion operations. We obtain the final 512-dimensional feature vector by concatenating the two latent codes for the start and end point cloud.

Figure 2: **Temporal Encoder.** This figure shows architectures of the temporal encoder $g^t_{\hat{\theta}}$ for the point cloud completion (Fig. 2a), 4D reconstruction from image sequences (Fig. 2b), and the shape matching experiments (Fig. 2c). In all figures, light green shows input, cyan fully connected layers, dark blue convolutional layers, red concatenation and pooling layers, and grey other operations.



(a) Backward flow of velocity network $\mathbf{v}_{\hat{\theta}}$



(b) Forward flow of adjusted velocity network $\hat{\mathbf{v}}_{\hat{\theta}}$

Figure 3: **Batch Processing.** The figure illustrates how we reformulate the backward flow of $\mathbf{v}_{\hat{\theta}}$ as the forward flow of the adjusted velocity network $\hat{\mathbf{v}}_{\hat{\theta}}$ (1) to perform batch processing of points $\mathbf{p}_i$ with different time values τ_i.

1.4. Batch Processing for the ODE Solver

For both $\mathcal{L}_{recon}$ and $\mathcal{L}_{corr}$ we use the adjoint sensitivity method [2, 11] to obtain gradients. We adjust the GPU implementation provided by Chen et al. [2] to our needs.[2] While batch-wise processing is already implemented for the forward flow $\Phi_{\hat{\theta}}$, we make use of a small trick for the backward flow $\Psi_{\hat{\theta}}$ as follows.

Let $\mathbf{p}_i$ be sampled points at random times $\tau_i \in (0, T]$. During training, we get the location of these points for time 0 by calculating the backward flow using the parameterized velocity network $\mathbf{v}_{\hat{\theta}}$. The time values τ_i are not equal for all points $\mathbf{p}_i$, so that batch processing is not as straightforward as for the forward flow. To obtain a batch-wise implementation, we reformulate this problem as solving the forward flow from time 0 to time τ_i using the adjusted velocity network

$$\hat{\mathbf{v}}_{\hat{\theta}}(\mathbf{p}_i, t) := -\mathbf{v}_{\hat{\theta}}(\mathbf{p}_i, \tau_i - t) \tag{1}$$

It results in exactly the same transformation (Fig. 3) while allowing us to use the same start time 0 for all points $\mathbf{p}_i$. This way, we are able to use the same batch-wise implementation as for the forward flow.

1.5. Additional Argument Passing to the ODE Solver

In the reconstruction and interpolation experiments, we condition the velocity network $\mathbf{v}^{\mathbf{x}}_{\hat{\theta}}$ on some input $\mathbf{x}$ in the form of a latent code c_t provided by the temporal encoder $g^t_{\hat{\theta}}$ (see Section 1.1.2). In the following, we describe how this can be done although the implementation of Chen et al. [2] allows only for passing points and time values for which the forward flow is solved.

To condition the velocity network on c_t, we first reshape the points tensor of dimension $|\mathcal{B}| \times N_{\mathbf{p}} \times 3$ to $|\mathcal{B}| \times (3N_{\mathbf{p}})$. Next, we can concatenate the latent code c_t to obtain the input tensor of size $|\mathcal{B}| \times (3N_{\mathbf{p}} + c_{dim})$ where c_{dim} indicates the dimension of c_t. As the ODE Solver expects the same output dimension of the velocity network $\mathbf{v}^{\mathbf{x}}_{\hat{\theta}}$, we reshape its output to $|\mathcal{B}| \times 3N_{\mathbf{p}}$ and fill the remaining values with zeros to obtain an output tensor of the same size $|\mathcal{B}| \times (3N_{\mathbf{p}} + c_{dim})$. The zero values for the coordinates corresponding to the latent code c_t make sure that the ODE Solver only considers the 3D points for evaluating the forward and backward flow and for choosing the appropriate step size.

2. Data Generation and Preprocessing

In this section, we describe the generation process of the *Warping Cars* dataset used in the reconstruction experiments as well as relevant data preprocessing steps.

2.1. Data Generation

To obtain the *Warping Cars* dataset, we use the ShapeNet [1] "car" category and apply random displacement fields to obtain a warping motion. To obtain *continuous* displacement fields, we first sample random Gaussian displacement vectors

[2] available via https://github.com/rtqichen/torchdiffeq

with mean $\mu = 0$ and standard deviation $\sigma = 0.15$ in a $3 \times 3 \times 3 \times 5$ grid. Here, the first 3 coordinates correspond to the spatial dimensions and the last coordinate corresponds to the temporal dimension. We then interpolate between these discrete displacement vectors using radial basis functions (RBFs) [10] to obtain a continuous displacement field in $\mathbb{R}^3 \times [0, T]$. To transform the input meshes, we simply apply this continuous displacement field to each vertex of the mesh individually.

2.2. Data Preprocessing

For detailed descriptions of the sampling process, we refer to [9] and restrict the discussion to the main changes.

To normalize the data while preserving the motion depicted in a sequence, we shift and scale all meshes belonging to the same sequence with respect to the bounding box of the start mesh. In addition, we set the end time $T = 1$ so that the time values lie between 0 and 1. For obtaining trajectories of points on the mesh, we first sample uniformly on the start mesh. We can track them over time by representing the points using trilinear coordinates with respect to the three vertices spanning the face on which the points lie.

As in [9], we perform offline sampling for computational efficiency. To this end, we perform sampling before training, saving the points to files instead of sampling online during training. To do this, we save 100,000 points uniformly sampled in the bounding volume and 100,000 points uniformly sampled on the mesh for all L time steps to files. For the former, we also save the respective occupancy values and for the latter, we save corresponding points so that we obtain corresponding trajectories over time.

3. Experiments

In this section we provide additional quantitative and qualitative results for the representation power (Section 3.1), reconstruction (Section 3.2 and Section 3.3), interpolation (Section 3.4), and generative experiments (Section 3.5). We further investigate differences of the OFlow and ONet 4D representations and additional properties of OFlow in Section 3.6.

3.1. Representation Power

The goal of the representation power experiment is to investigate how well OFlow can represent 3D shapes in motion. To disentangle the influence of the spatial and temporal encoders g_θ^s and g_θ^t from the representation power, we trained OFlow to represent single sequences of length $L = 50$ with no input at all (see main publication) and to represent 10 different sequences of length $L = 50$ with IDs as input. In addition, we also trained a model to represent 32 sequences of length $L = 17$ with point cloud input similar to the 4D point cloud completion training setup without adding Gaussian noise.

Num. Seq.	Seq. Length	IoU	Chamfer	Correspond.
32	17	91.2 %	0.032	0.061
10	50	88.5 %	0.043	0.085

Table 1: **Additional Representation Power Experiments.** We report IoU (higher is better), Chamfer distance (lower is better), and correspondence distance (lower is better) for representation power experiments with different numbers and lengths of sequences.

Table 1 shows quantitative results of the additional representation power experiments with different numbers of sequences and lengths of sequences. Although slightly decreased compared to the single sequence setting from the main publication (IoU of 93%), the results indicate that OFlow can represent the more complex data well (IoU scores above 88%) regardless of the input.

3.2. Point Cloud Completion

In Table 2, 3, and 4 we show detailed results for the point cloud completion experiments from Section 4.2 in the main publication. We evaluate IoU, Chamfer distance, and the correspondence distance in 17 equally spaced time steps between 0 and 1. We observe that the OFlow variants outperform ONet 4D in terms of IoU and achieve lower Chamfer and correspondence distances than PSGN 4D and ONet 4D over all time steps on the D-FAUST dataset. On the synthetic warping cars dataset, we can see that PSGN 4D obtains the lowest Chamfer distance which comes as no surprise because this method, in contrast to OFlow, is directly trained on this metric.

3.3. 4D Reconstruction from Image Sequences

The complete results for the 4D reconstruction from image sequences can be found in Table 5, 6, and 7. The relative results are similar to the point cloud completion experiment with the OFlow variants achieving the best IoU scores and lowest correspondence distances. Depending on the dataset, either OFlow or PSGN 4D obtains the lowest Chamfer distances which comes as no surprise because PSGN 4D directly optimizes this metric. As indicated in Section 4.3 of the main publication,

Figure 4: **4D Point Cloud Completion (D-FAUST).** This figure shows two examples 4a and 4b from the test set of the point cloud completion experiment on the D-FAUST dataset. For both examples, we show the input as well the output of OFlow , ONet 4D, and PSGN 4D for 9 equally spaced time steps between 0 and 1.

	IoU					Chamfer					Correspond.				
time step	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)
0.0	-	-	76.3 %	80.5 %	**83.1 %**	0.110	0.104	0.090	0.070	**0.059**	0.075	0.068	-	0.068	**0.057**
0.0625	-	-	77.1 %	80.8 %	**83.1 %**	0.108	0.103	0.087	0.069	**0.059**	3.466	0.073	-	0.074	**0.062**
0.125	-	-	77.5 %	80.7 %	**82.8 %**	0.107	0.102	0.085	0.069	**0.060**	3.496	0.081	-	0.085	**0.069**
0.1875	-	-	77.8 %	80.5 %	**82.5 %**	0.107	0.102	0.084	0.070	**0.061**	3.447	0.090	-	0.096	**0.077**
0.25	-	-	78.0 %	80.3 %	**82.2 %**	0.106	0.101	0.083	0.071	**0.062**	3.438	0.097	-	0.105	**0.083**
0.3125	-	-	78.2 %	80.1 %	**82.0 %**	0.106	0.101	0.082	0.072	**0.063**	3.450	0.102	-	0.113	**0.088**
0.375	-	-	78.3 %	80.0 %	**81.8 %**	0.106	0.101	0.082	0.072	**0.064**	3.442	0.106	-	0.120	**0.092**
0.4375	-	-	78.4 %	79.9 %	**81.6 %**	0.106	0.101	0.081	0.073	**0.064**	3.415	0.109	-	0.125	**0.095**
0.5	-	-	78.5 %	79.8 %	**81.4 %**	0.106	0.101	0.081	0.073	**0.065**	3.495	0.111	-	0.130	**0.098**
0.5625	-	-	78.5 %	79.8 %	**81.3 %**	0.107	0.101	0.081	0.074	**0.066**	3.399	0.111	-	0.133	**0.101**
0.625	-	-	78.4 %	79.8 %	**81.1 %**	0.107	0.101	0.082	0.074	**0.066**	3.450	0.112	-	0.136	**0.103**
0.6875	-	-	78.4 %	79.8 %	**81.0 %**	0.107	0.100	0.082	0.074	**0.067**	3.409	0.112	-	0.138	**0.106**
0.75	-	-	78.2 %	79.7 %	**80.8 %**	0.108	0.100	0.083	0.075	**0.068**	3.441	0.112	-	0.141	**0.108**
0.8125	-	-	78.1 %	79.6 %	**80.6 %**	0.108	0.100	0.083	0.075	**0.069**	3.424	0.112	-	0.143	**0.111**
0.875	-	-	77.8 %	79.3 %	**80.3 %**	0.109	0.101	0.085	0.076	**0.070**	3.429	**0.113**	-	0.147	0.114
0.9375	-	-	77.4 %	78.8 %	**80.0 %**	0.110	0.101	0.086	0.078	**0.071**	3.385	**0.115**	-	0.152	0.119
1.0	-	-	76.9 %	78.2 %	**79.5 %**	0.113	0.103	0.089	0.081	**0.073**	3.309	**0.120**	-	0.160	0.125
mean	-	-	77.9 %	79.9 %	**81.5 %**	0.108	0.101	0.084	0.073	**0.065**	3.234	0.102	-	0.122	**0.094**

Table 2: **Point Cloud Completion for Seen Individuals (D-FAUST).** We report IoU, Chamfer distance, and correspondence distance for all 17 time steps for the point cloud completion experiment for the test set consisting of seen individuals (**but unseen motions**) of the D-FAUST dataset. As both PSGN 4D variants output point sets without connectivity, IoU cannot be measured for these methods. ONet 4D only predicts occupancy probabilities for 4D points without any correspondence information between time values, so that no correspondence distance can be calculated.

	IoU					Chamfer					Correspond.				
time step	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)
0.0	-	-	64.8 %	70.4 %	**74.2 %**	0.130	0.121	0.148	0.091	**0.077**	0.104	0.093	-	0.093	**0.077**
0.0625	-	-	65.7 %	70.7 %	**74.1 %**	0.129	0.120	0.143	0.090	**0.077**	3.261	0.098	-	0.098	**0.082**
0.125	-	-	66.2 %	70.6 %	**73.8 %**	0.127	0.119	0.141	0.090	**0.078**	3.281	0.105	-	0.108	**0.089**
0.1875	-	-	66.5 %	70.4 %	**73.4 %**	0.127	0.118	0.139	0.091	**0.079**	3.239	0.112	-	0.117	**0.096**
0.25	-	-	66.7 %	70.2 %	**73.0 %**	0.126	0.118	0.138	0.092	**0.081**	3.225	0.120	-	0.127	**0.102**
0.3125	-	-	66.9 %	70.0 %	**72.7 %**	0.125	0.118	0.137	0.093	**0.082**	3.243	0.126	-	0.135	**0.108**
0.375	-	-	67.0 %	69.8 %	**72.4 %**	0.125	0.119	0.137	0.094	**0.083**	3.231	0.130	-	0.143	**0.113**
0.4375	-	-	67.1 %	69.7 %	**72.2 %**	0.125	0.119	0.136	0.095	**0.084**	3.202	0.134	-	0.149	**0.117**
0.5	-	-	67.2 %	69.6 %	**72.0 %**	0.125	0.119	0.136	0.095	**0.085**	3.285	0.137	-	0.155	**0.121**
0.5625	-	-	67.3 %	69.6 %	**71.9 %**	0.125	0.119	0.136	0.096	**0.085**	3.190	0.140	-	0.160	**0.124**
0.625	-	-	67.3 %	69.5 %	**71.8 %**	0.125	0.119	0.136	0.096	**0.086**	3.245	0.142	-	0.164	**0.127**
0.6875	-	-	67.2 %	69.4 %	**71.7 %**	0.125	0.119	0.137	0.096	**0.086**	3.208	0.143	-	0.168	**0.130**
0.75	-	-	67.1 %	69.4 %	**71.5 %**	0.126	0.119	0.138	0.097	**0.087**	3.236	0.145	-	0.172	**0.133**
0.8125	-	-	66.9 %	69.2 %	**71.4 %**	0.126	0.119	0.139	0.098	**0.087**	3.216	0.147	-	0.176	**0.136**
0.875	-	-	66.6 %	68.9 %	**71.3 %**	0.127	0.119	0.141	0.099	**0.088**	3.226	0.149	-	0.181	**0.140**
0.9375	-	-	66.3 %	68.5 %	**71.0 %**	0.128	0.120	0.143	0.100	**0.089**	3.184	0.153	-	0.187	**0.145**
1.0	-	-	65.8 %	67.9 %	**70.7 %**	0.130	0.121	0.146	0.102	**0.090**	3.116	0.157	-	0.195	**0.150**
mean	-	-	66.6 %	69.6 %	**72.3 %**	0.127	0.119	0.140	0.095	**0.084**	3.041	0.131	-	0.149	**0.117**

Table 3: **Point Cloud Completion for Unseen Individual (D-FAUST).** We report IoU, Chamfer distance, and correspondence distance for all 17 time steps for the point cloud completion experiment for the unseen individual of the D-FAUST dataset.

both the quantitative and qualitative results show that reconstruction from image sequences is a harder task than point cloud completion. Fig. 10 shows two failure cases for OFlow: as we sample the view point randomly, extreme lighting conditions and dominant occlusions make it very hard to correctly infer the complex motions. In addition, high frequency motions are not recovered completely. As OFlow is able to recover the static part of the 3D geometry, we assume that incorporating local image features would particularly improve the motion reconstruction.

3.4. Interpolation

The goal of the interpolation experiment is to investigate to which degree OFlow can be used for shape interpolation. The task is to find a continuous transformation between start and end mesh $M_S = \left(\{\mathbf{v}_S^i\}_i, F\right)$ and $M_E = \left(\{\mathbf{v}_E^i\}_i, F\right)$ where the two shapes have corresponding vertices $\mathbf{v}_S^i$ and $\mathbf{v}_E^i$ and the same faces F. As *both*, the start and the end shape are given in this task, we can predict a transformation from M_S to M_E as well as the other way round, from M_E to M_S. The predictions for the first case are given by $M^t_{\text{forward}} = \left(\{\Phi_{\hat{\theta}}(\mathbf{v}_S^i, t)\}_i, F\right)$. For predicting the vertex locations from the end to the start mesh, we propagate the vertices $\mathbf{v}_E^i$ from time $t = 1$ backwards to time t using the backward flow $\Psi_{\hat{\theta}}$. Let

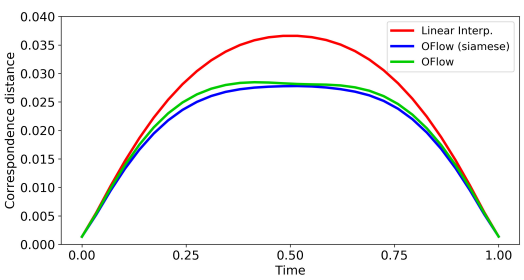
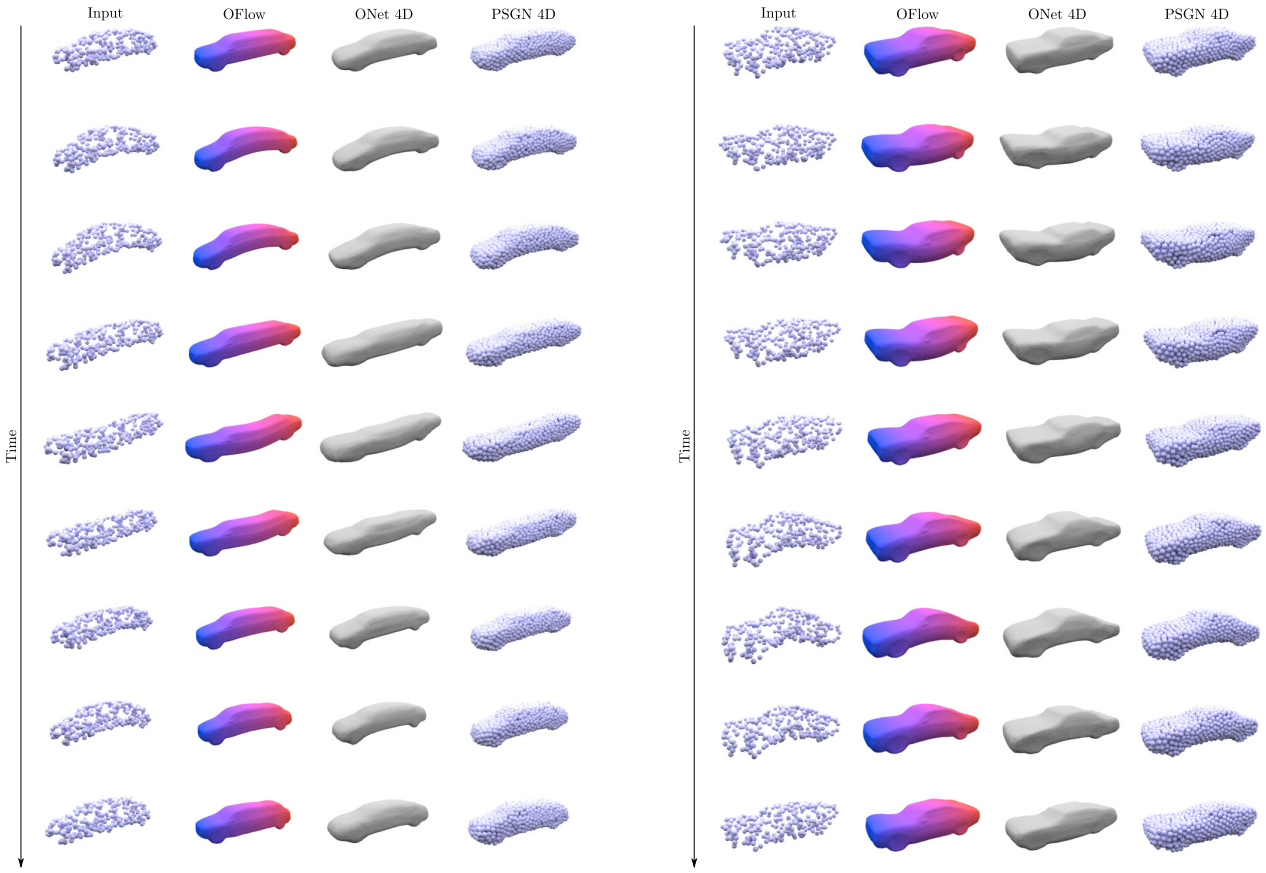


Figure 8: **Interpolation Encoder Comparison.** We show the correspondence distance over time for the interpolation experiment for the linear interpolation baseline and OFlow with two different encoding schemes.

time step	IoU			Chamfer			Correspond.		
	PSGN 4D	ONet 4D	OFlow	PSGN 4D	ONet 4D	OFlow	PSGN 4D	ONet 4D	OFlow
0.0	-	73.7 %	**80.9 %**	0.146	0.165	**0.117**	0.095	-	**0.080**
0.0625	-	73.7 %	**78.4 %**	0.146	0.165	**0.130**	4.188	-	**0.138**
0.125	-	71.6 %	**74.6 %**	0.152	0.178	**0.149**	4.018	-	**0.210**
0.1875	-	69.5 %	**71.9 %**	**0.157**	0.191	0.162	4.103	-	**0.258**
0.25	-	68.9 %	**70.9 %**	**0.159**	0.195	0.166	4.230	-	**0.275**
0.3125	-	69.3 %	**71.3 %**	**0.157**	0.194	0.165	4.138	-	**0.269**
0.375	-	69.7 %	**71.6 %**	**0.157**	0.191	0.164	4.192	-	**0.266**
0.4375	-	69.6 %	**71.1 %**	**0.158**	0.190	0.165	4.219	-	**0.279**
0.5	-	69.2 %	**70.9 %**	**0.159**	0.191	0.166	4.095	-	**0.289**
0.5625	-	69.7 %	**70.9 %**	**0.157**	0.187	0.166	4.167	-	**0.289**
0.625	-	70.1 %	**70.5 %**	**0.156**	0.184	0.169	4.068	-	**0.293**
0.6875	-	**69.3 %**	69.2 %	**0.158**	0.189	0.176	4.140	-	**0.313**
0.75	-	**68.6 %**	68.1 %	**0.160**	0.193	0.182	4.043	-	**0.330**
0.8125	-	**69.2 %**	67.8 %	**0.159**	0.190	0.184	4.151	-	**0.334**
0.875	-	**69.9 %**	67.3 %	**0.157**	0.187	0.188	4.126	-	**0.344**
0.9375	-	**68.1 %**	65.3 %	**0.160**	0.202	0.201	4.022	-	**0.387**
1.0	-	**64.4 %**	62.0 %	**0.172**	0.232	0.222	4.066	-	**0.462**
mean	-	69.7 %	**70.7 %**	**0.157**	0.190	0.169	3.886	-	**0.283**

Table 4: **Point Cloud Completion (Warping Cars).** We report IoU, Chamfer distance, and correspondence distance for all 17 time steps for the test set of the warping cars dataset.



(a) (b)

Figure 5: **4D Point Cloud Completion (Warping Cars).** This figure shows two examples from the test set of the point cloud completion experiment on the warping cars dataset. For both examples, we show the input as well the output of OFlow, ONet 4D, and PSGN 4D for 9 equally spaced time steps between 0 and 1.

	IoU					Chamfer					Correspond.				
time step	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)
0.0	-	-	44.6 %	58.4 %	**60.5 %**	0.252	0.264	0.343	0.181	**0.160**	0.203	0.219	-	0.167	**0.158**
0.0625	-	-	44.7 %	58.4 %	**60.5 %**	0.252	0.263	0.343	0.182	**0.160**	2.717	2.616	-	0.177	**0.163**
0.125	-	-	44.6 %	58.3 %	**60.5 %**	0.252	0.264	0.343	0.182	**0.160**	2.678	2.738	-	0.194	**0.172**
0.1875	-	-	44.6 %	58.0 %	**60.4 %**	0.253	0.264	0.343	0.184	**0.161**	2.732	2.661	-	0.214	**0.182**
0.25	-	-	44.5 %	57.7 %	**60.3 %**	0.254	0.265	0.343	0.186	**0.162**	2.783	2.707	-	0.235	**0.192**
0.3125	-	-	44.4 %	57.4 %	**60.2 %**	0.255	0.265	0.344	0.188	**0.163**	2.761	2.753	-	0.256	**0.203**
0.375	-	-	44.3 %	57.0 %	**60.0 %**	0.256	0.265	0.344	0.190	**0.164**	2.721	2.791	-	0.276	**0.213**
0.4375	-	-	44.2 %	56.7 %	**59.9 %**	0.256	0.265	0.345	0.191	**0.165**	2.775	2.771	-	0.294	**0.222**
0.5	-	-	44.1 %	56.4 %	**59.7 %**	0.257	0.265	0.346	0.193	**0.166**	2.750	2.721	-	0.311	**0.231**
0.5625	-	-	44.0 %	56.1 %	**59.6 %**	0.258	0.266	0.347	0.195	**0.167**	2.710	2.711	-	0.325	**0.240**
0.625	-	-	43.9 %	55.8 %	**59.4 %**	0.259	0.266	0.349	0.197	**0.168**	2.678	2.719	-	0.338	**0.247**
0.6875	-	-	43.8 %	55.6 %	**59.2 %**	0.260	0.266	0.350	0.198	**0.169**	2.725	2.758	-	0.349	**0.254**
0.75	-	-	43.7 %	55.4 %	**59.1 %**	0.261	0.267	0.352	0.199	**0.170**	2.767	2.793	-	0.357	**0.261**
0.8125	-	-	43.5 %	55.3 %	**58.9 %**	0.262	0.267	0.353	0.200	**0.171**	2.707	2.749	-	0.364	**0.267**
0.875	-	-	43.4 %	55.1 %	**58.6 %**	0.263	0.267	0.354	0.201	**0.173**	2.710	2.797	-	0.369	**0.273**
0.9375	-	-	43.2 %	55.0 %	**58.4 %**	0.264	0.266	0.356	0.202	**0.174**	2.704	2.716	-	0.371	**0.278**
1.0	-	-	43.0 %	54.9 %	**58.1 %**	0.265	0.267	0.357	0.203	**0.175**	2.671	2.633	-	0.373	**0.283**
mean	-	-	44.0 %	56.6 %	**59.6 %**	0.258	0.265	0.348	0.193	**0.166**	2.576	2.580	-	0.292	**0.226**

Table 5: **4D Reconstruction from Image Sequences for Seen Individuals (D-FAUST).** The table shows IoU, Chamfer distance, and correspondence distance for the 17 time steps evaluated on the test set of seen individuals (**but unseen motions**) of the D-FAUST dataset. IoU cannot be measured for PSGN 4D due to missing connectivity information and for ONet 4D, correspondence distances cannot be calculated as no correspondence information is predicted.

	IoU					Chamfer					Correspond.				
time step	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	PSGN 4D (w/ cor.)	ONet 4D	OFlow	OFlow (w/ cor.)
0.0	-	-	36.2 %	42.9 %	**43.8 %**	0.302	0.295	0.387	0.285	**0.269**	**0.247**	**0.247**	-	0.286	0.281
0.0625	-	-	36.3 %	43.0 %	**43.8 %**	0.302	0.296	0.388	0.285	**0.269**	2.590	2.505	-	0.293	**0.286**
0.125	-	-	36.3 %	43.0 %	**43.8 %**	0.305	0.298	0.388	0.285	**0.269**	2.556	2.624	-	0.307	**0.296**
0.1875	-	-	36.3 %	42.9 %	**43.8 %**	0.306	0.299	0.389	0.285	**0.270**	2.606	2.553	-	0.324	**0.309**
0.25	-	-	36.3 %	42.9 %	**43.7 %**	0.308	0.301	0.389	0.286	**0.271**	2.652	2.603	-	0.342	**0.322**
0.3125	-	-	36.3 %	42.8 %	**43.7 %**	0.309	0.303	0.390	0.287	**0.271**	2.638	2.644	-	0.361	**0.335**
0.375	-	-	36.2 %	42.6 %	**43.6 %**	0.311	0.304	0.391	0.288	**0.272**	2.605	2.685	-	0.379	**0.349**
0.4375	-	-	36.1 %	42.5 %	**43.5 %**	0.312	0.306	0.392	0.289	**0.273**	2.664	2.667	-	0.396	**0.361**
0.5	-	-	36.0 %	42.4 %	**43.4 %**	0.313	0.308	0.393	0.290	**0.274**	2.642	2.621	-	0.412	**0.374**
0.5625	-	-	35.9 %	42.3 %	**43.3 %**	0.314	0.310	0.395	0.291	**0.275**	2.604	2.617	-	0.426	**0.385**
0.625	-	-	35.8 %	42.2 %	**43.2 %**	0.315	0.311	0.396	0.292	**0.277**	2.575	2.626	-	0.438	**0.396**
0.6875	-	-	35.6 %	42.1 %	**43.1 %**	0.316	0.312	0.397	0.294	**0.278**	2.619	2.660	-	0.449	**0.406**
0.75	-	-	35.5 %	42.0 %	**43.0 %**	0.318	0.314	0.398	0.295	**0.279**	2.658	2.696	-	0.458	**0.416**
0.8125	-	-	35.3 %	41.9 %	**42.9 %**	0.318	0.315	0.400	0.296	**0.280**	2.606	2.660	-	0.465	**0.426**
0.875	-	-	35.1 %	41.8 %	**42.7 %**	0.318	0.316	0.401	0.297	**0.282**	2.611	2.705	-	0.471	**0.434**
0.9375	-	-	34.9 %	41.7 %	**42.6 %**	0.319	0.316	0.402	0.298	**0.283**	2.604	2.632	-	0.476	**0.442**
1.0	-	-	34.7 %	41.6 %	**42.4 %**	0.319	0.316	0.403	0.299	**0.284**	2.578	2.550	-	0.479	**0.449**
mean	-	-	35.8 %	42.4 %	**43.3 %**	0.312	0.307	0.394	0.291	**0.275**	2.474	2.488	-	0.398	**0.369**

Table 6: **4D Reconstruction from Image Sequences on Unseen Individual (D-FAUST).** We show IoU, Chamfer distance, and correspondence distance for all 17 time steps evaluated on the test set of the unseen individual of the D-FAUST dataset.

$\Psi_{\hat{\theta}}(\mathbf{p}, 1, t)$ indicate the position of point $\mathbf{p}$ from time $t = 1$ at time $t \leq 1$ when following the velocity field $\mathbf{v}_{\hat{\theta}}(\cdot, \cdot)$ backwards in time. The backward predictions are then $M^t_{\text{backward}} = \left(\{\Psi_{\hat{\theta}}(\mathbf{v}^i_E, 1, t)\}_i, F\right)$. We obtain our final predictions by interpolating between the two predictions:

$$M^t_{\text{pred}} = \left(\{(1-t) \cdot \Phi_{\hat{\theta}}(\mathbf{v}^i_S, t) + t \cdot \Psi_{\hat{\theta}}(\mathbf{v}^i_E, 1, t)\}_i, F\right) \tag{2}$$

In Fig. 8 we show a comparison of OFlow with two different encoder architectures and the linear interpolation baseline. As discussed in Section 1.1.2, the start and end point clouds can either be encoded in a siamese fashion by encoding each point cloud individually or by concatenating the start and end locations. The figure shows that encoding the point clouds individually with the same network achieves slightly better results. This comes as a surprise as this encoding scheme does not directly use the correspondences between the start and end point clouds.

In Fig. 9 we show additional qualitative results from the test set of the interpolation experiment. They show that, in contrast to linear interpolation, OFlow is able to capture the non-linear motion of the non-rigid body shapes from the D-FAUST dataset.

3.5. Probabilistic Latent Variable Model

We further conducted experiments investigating the generative capabilities of our representation. For this, we slightly adjust the spatial and temporal encoders g^s_θ and $g^t_{\hat{\theta}}$ for 4D point cloud completion to predict means and log standard deviations $(\mu_s, \log \sigma_s)$ and $(\mu_t, \log \sigma_t)$ of Gaussian distributions $q^s_\theta(z|\mathbf{x})$ and $q^t_{\hat{\theta}}(z|\mathbf{x})$ instead of latent codes c_s and c_t. We then obtain the spatial and temporal latent codes z_s and z_t for conditioning the occupancy and velocity networks by sampling from the

time step	IoU				Chamfer				Correspond.			
	PSGN 4D	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	ONet 4D	OFlow	OFlow (w/ cor.)	PSGN 4D	ONet 4D	OFlow	OFlow (w/ cor.)
0.0	-	67.2 %	71.2 %	**72.6 %**	0.191	0.213	0.185	**0.169**	0.144	-	0.136	**0.125**
0.0625	-	62.9 %	66.6 %	**67.7 %**	0.212	0.251	0.218	**0.198**	4.277	-	0.257	**0.250**
0.125	-	57.7 %	60.7 %	**61.0 %**	**0.239**	0.297	0.260	0.240	4.191	-	0.409	**0.405**
0.1875	-	54.6 %	**57.2 %**	56.9 %	**0.257**	0.326	0.286	0.270	4.151	-	**0.508**	**0.508**
0.25	-	53.9 %	**56.5 %**	55.9 %	**0.261**	0.332	0.290	0.277	4.180	-	**0.531**	**0.531**
0.3125	-	54.7 %	**57.6 %**	57.0 %	**0.256**	0.326	0.280	0.269	4.141	-	0.502	**0.499**
0.375	-	55.1 %	**58.2 %**	57.7 %	**0.253**	0.322	0.274	0.263	4.214	-	0.489	**0.482**
0.4375	-	54.3 %	**57.1 %**	56.9 %	**0.258**	0.328	0.281	0.267	4.154	-	0.522	**0.511**
0.5	-	53.7 %	**56.2 %**	**56.2 %**	**0.262**	0.334	0.289	0.273	4.344	-	0.552	**0.542**
0.5625	-	54.0 %	**56.6 %**	56.5 %	**0.259**	0.332	0.287	0.271	4.111	-	0.549	**0.541**
0.625	-	54.5 %	**57.1 %**	56.8 %	**0.255**	0.329	0.285	0.270	4.097	-	0.539	**0.533**
0.6875	-	54.0 %	**56.3 %**	56.0 %	**0.258**	0.335	0.292	0.277	4.254	-	0.555	**0.552**
0.75	-	53.3 %	**55.3 %**	54.9 %	**0.262**	0.341	0.299	0.286	4.201	-	**0.574**	**0.574**
0.8125	-	53.6 %	**55.7 %**	55.2 %	**0.259**	0.338	0.297	0.285	4.092	-	**0.559**	0.561
0.875	-	54.6 %	**56.8 %**	56.4 %	**0.252**	0.329	0.288	0.276	4.160	-	**0.521**	0.524
0.9375	-	54.4 %	**56.4 %**	56.0 %	**0.255**	0.333	0.291	0.279	4.221	-	**0.527**	0.530
1.0	-	52.1 %	**53.5 %**	52.9 %	**0.275**	0.356	0.314	0.303	4.200	-	**0.612**	0.616
mean	-	55.6 %	**58.2 %**	58.0 %	**0.251**	0.319	0.277	0.263	3.949	-	0.491	**0.487**

Table 7: **4D Reconstruction from Image Sequences (Warping Cars).** We report IoU, Chamfer distance, and correspondence distance for all 17 time steps for the test set of the warping cars dataset.

resulting distributions q^s_θ and $q^t_{\hat{\theta}}$. In contrast to having only one latent space, this decoupling of shape and motion allows us to sample spatial and temporal latent codes individually.

We combine the reconstruction and correspondence loss with the KL-divergence between the predicted and prior distributions to optimize a lower bound to the negative log-likelihood of our generative model:

$$\mathcal{L}_{generative}\left(\theta, \hat{\theta}\right) = \mathcal{L}_{recon}\left(\theta, \hat{\theta}\right) + \mathcal{L}_{corr}\left(\hat{\theta}\right) + \frac{\beta}{|\mathcal{B}|}\sum_{\mathbf{x}\in\mathcal{B}}(\mathrm{KL}(q^s_\theta(z|\mathbf{x}), p^s_0) + \mathrm{KL}(q^t_{\hat{\theta}}(z|\mathbf{x}), p^t_0)) \tag{3}$$

We choose the prior distributions p^s_0 and p^t_0 to be Gaussian distributions. The hyperparameter β can be interpreted as a weighting between the reconstruction and the regularization term [7]. In our experiments we set $\beta := 10^{-4}$.

Fig. 11 shows a t-SNE embedding of the motion codes of the training examples of the D-FAUST dataset. Fig. 12 shows latent shape and motion interpolations. We show the interpolation between two shape samples z^0_s and z^1_s with a fixed motion sample z_t in Fig. 12a. Similarly, we show the interpolation between two motion samples z^0_t and z^1_t with a fixed shape sample z_s in Fig. 12b. Finally, in Fig. 13 we show three examples of the motion transfer experiment. For this, we select a shape S and a sequence $\mathbf{x}$ from the D-FAUST dataset. We then encode the motion of sequence $\mathbf{x}$ to latent code z_t (predicted mean of $q^t_\theta(\mathbf{x})$) and apply this motion to shape S. We are able to apply motion z_t directly to the ground truth shape S due to our vector field-based motion representation.

The results show that OFlow is able to learn a meaningful latent space representation. The decoupling of shape and motion allows to encode and sample the two components individually. This flexibility enables OFlow to be used in various tasks ranging from shape or motion interpolation to motion transfer.

A limiting factor of this approach is that motion transfers could not work for completely different shapes, e.g. a human upside-down, as we do not condition the motion on the shape. However, the simple solution to condition the motion distribution on the drawn shape sample would drastically reduce the flexibility and variety of drawn samples. We plan to investigate how to combine these approaches in future work.

3.6. Additional Comparisons and Experiments

In the following we compare the OFlow and ONet 4D representation (Section 3.6.1), analyze the connectivity-preserving property of OFlow (Section 3.6.2) and show that it can be trained on multiple categories simultaneously without degraded results (Section 3.6.3).

3.6.1 Comparison of the ONet 4D and OFlow Representations

The qualitative and quantitative results clearly demonstrate that OFlow performs better than ONet 4D on unseen data while both methods exhibit similar representation capabilities (see Section 4.1 of the main publication). While ONet 4D predicts

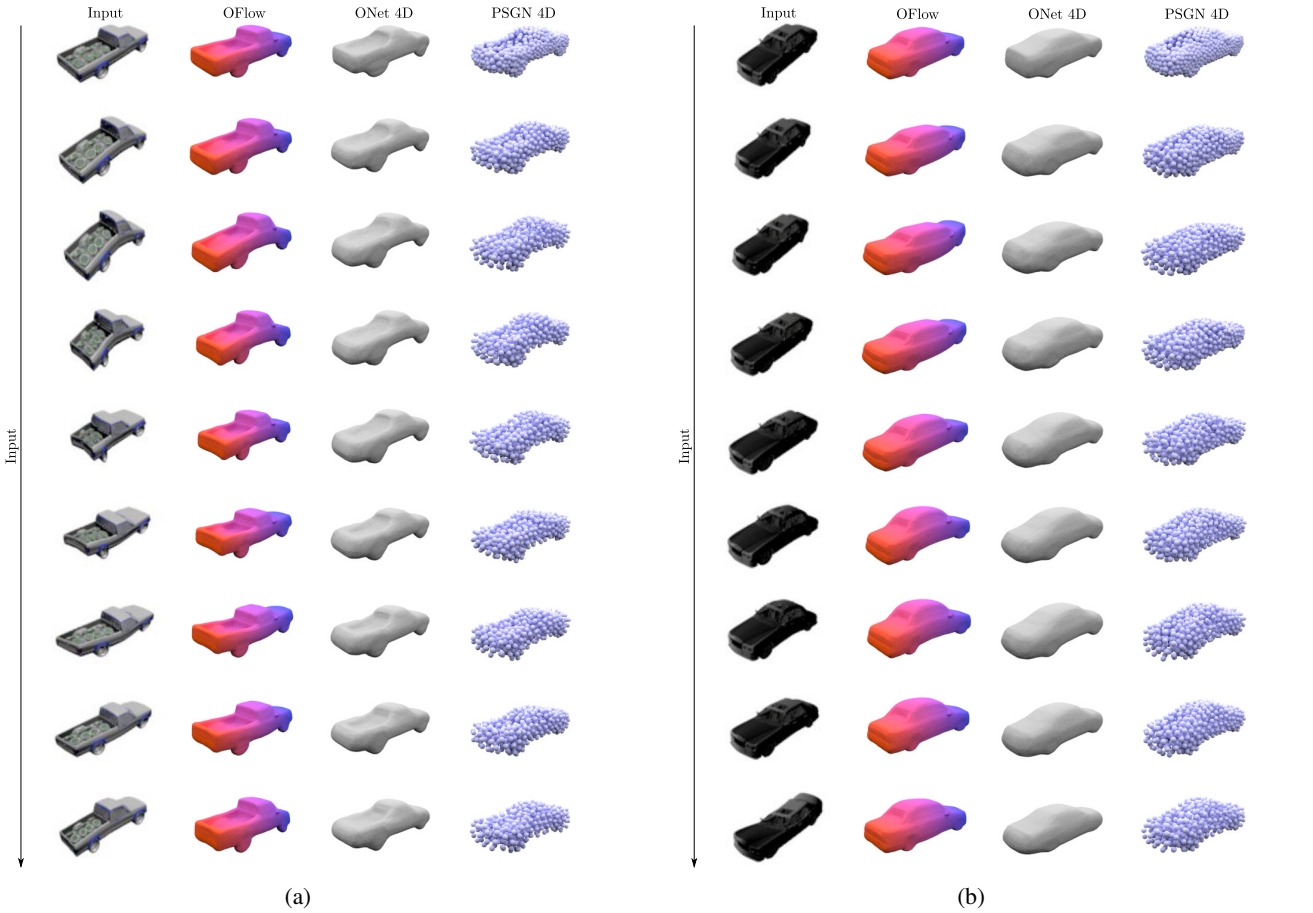


Figure 6: **4D Reconstruction from Image Sequences (Warping Cars).** This figure shows two examples from the test set of the 4D reconstruction from image sequence experiment on the warping cars dataset. For both examples, we show the input as well the output of OFlow, ONet 4D, and PSGN 4D for 9 equally spaced time steps between 0 and 1.

occupancy probabilities in space for all time steps individually, OFlow propagates the predicted 3D shape from time 0 forward by using the parameterized vector field, hence exploiting the physical properties of objects in motion. This way, the tasks of shape and motion reconstruction are decoupled resulting in a more intuitive and simpler problem as the motion vectors, in contrast to occupancy values, often only change slightly over time. Importantly, while the proposed OFlow model is able to exploit the smoothness properties of the motion field, the naïve ONet 4D approach does not, hence leading to higher sampling complexity and degraded results. In Fig. 14 we show an example highlighting the differences of the two approaches: While ONet 4D often produces overly-smooth results, OFlow is able to better preserve the 3D shape over time.

3.6.2 Connectivity-Preserving Property of the OFlow Representation

We model the velocity field as an invertible mapping which preserves connectivity information. To investigate this property, we trained a model to represent a short sequence of touching hands extracted from the MANO dataset [13]. Indeed, the touching hands in Fig. 15 keep their form and do not merge. While OFlow is not able to represent merging of two objects due to its bijectivity, the spatial proximity of the objects in the reconstruction is small enough to realistically model the motion. Depending on the application, this property can also be an advantage, e.g. not leading to self-intersections.

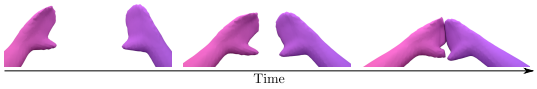


Figure 15: **Reconstruction of Touching Hands.** Note that the objects keep their form/topology and do not merge during touching.

3.6.3 Training on Multiple Object and Motion Categories

Table 8 shows quantitative results of the 4D point cloud completion experiment for OFlow trained and tested on D-FAUST and Warping Cars simultaneously. We observe no or only slight changes wrt. the single category models. This shows that OFlow can be trained with different classes of objects and motions simultaneously without a significant change in performance.

	IoU		Chamfer		Correspond	
	mul. cat.	sin. cat.	mul. cat.	sin. cat.	mul. cat.	sin. cat.
D-FAUST	79.9 %	79.9 %	0.073	0.073	0.122	0.122
War. Cars	68.3 %	70.7 %	0.196	0.169	0.319	0.283

Table 8: **4D Point Cloud Completion on Multiple Categories.** The columns show if the model was trained on a single (sin. cat.) or multiple categories (mul. cat.), and the rows show on which dataset the model is evaluated.

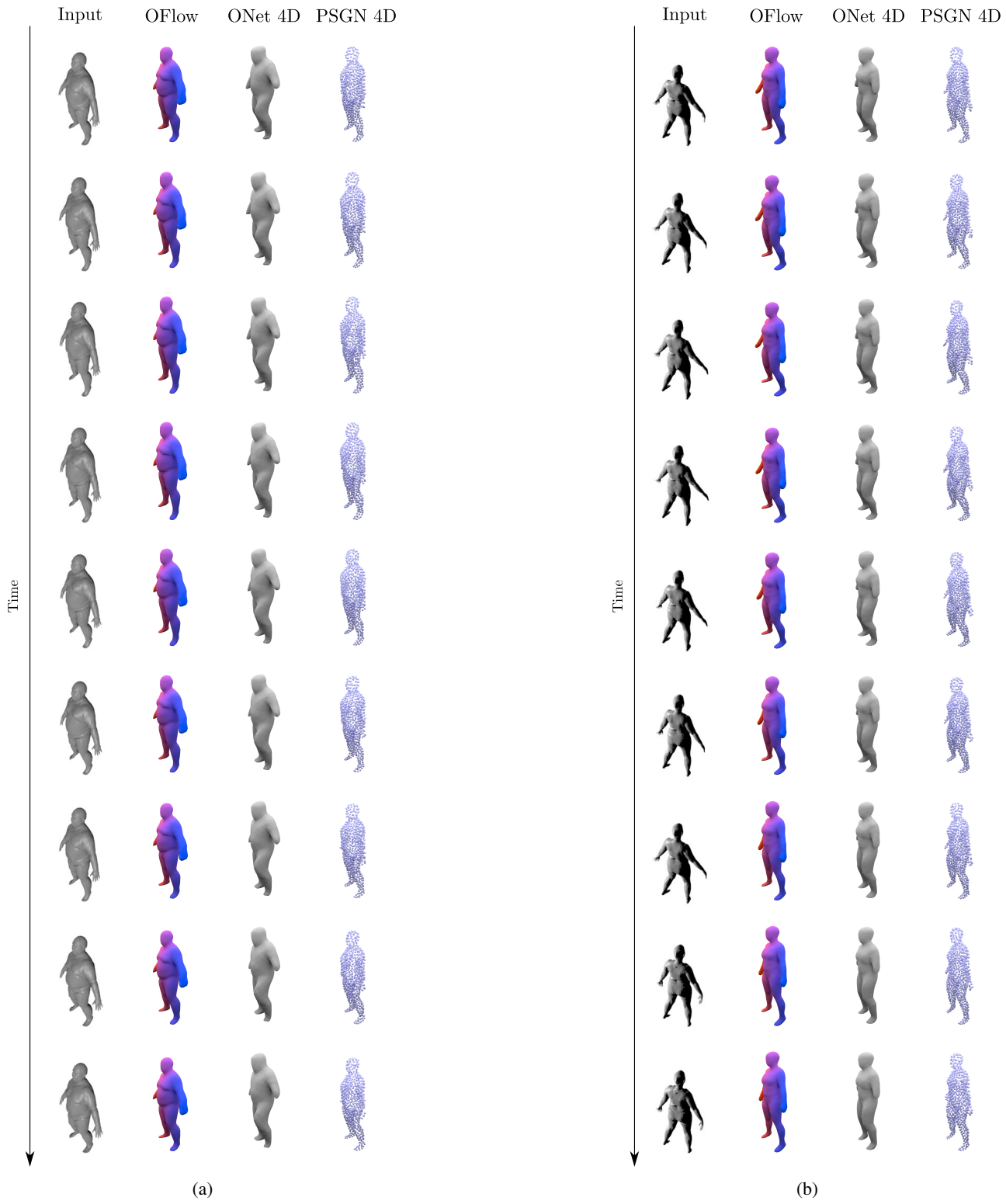


Figure 7: **4D Reconstruction from Image Sequences (D-FAUST).** This figure shows two examples from the test set of the 4D reconstruction experiment on the D-FAUST dataset. For both examples, we show the input as well the output of OFlow, ONet 4D, and PSGN 4D for 9 equally spaced time steps between 0 and 1.

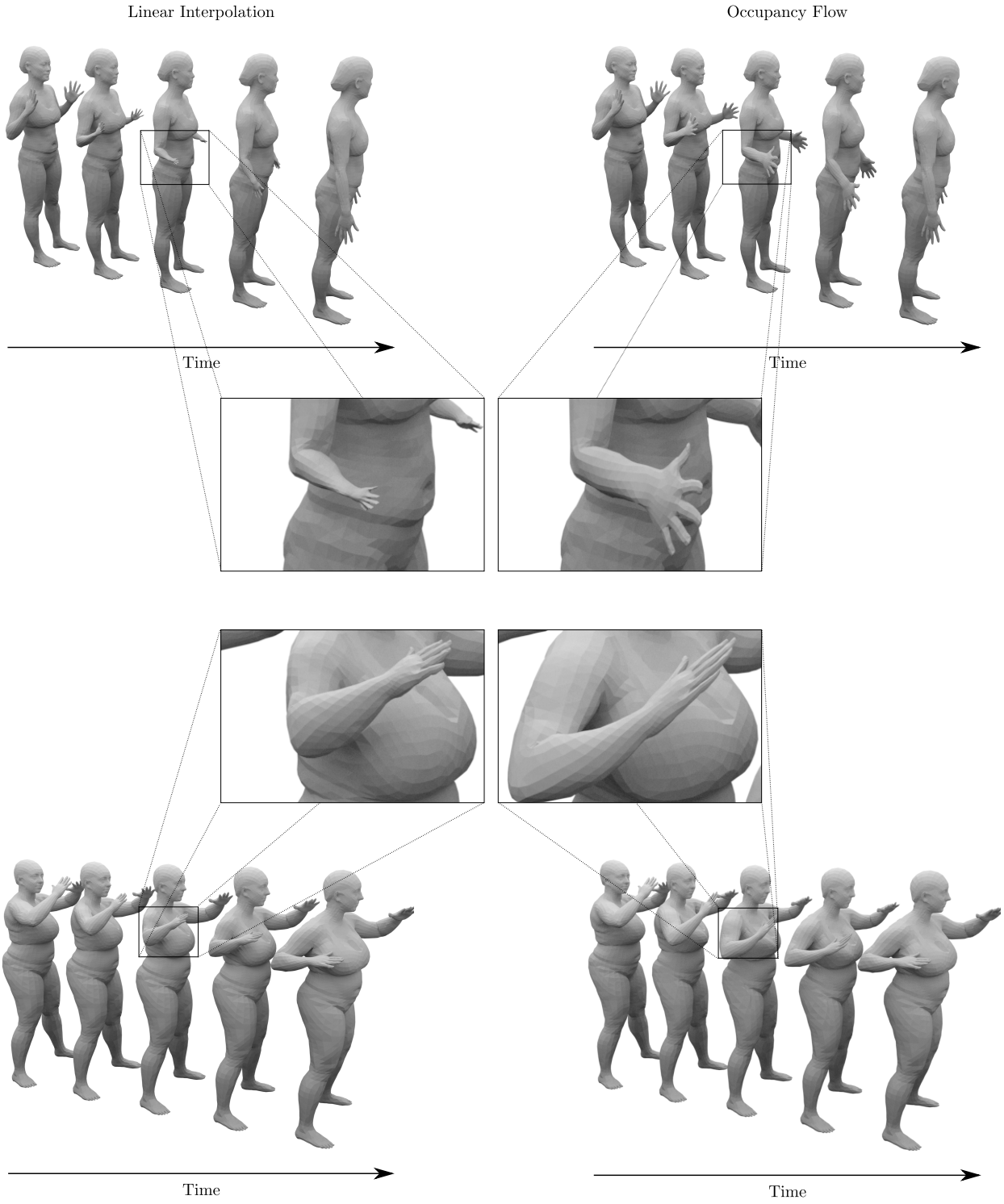
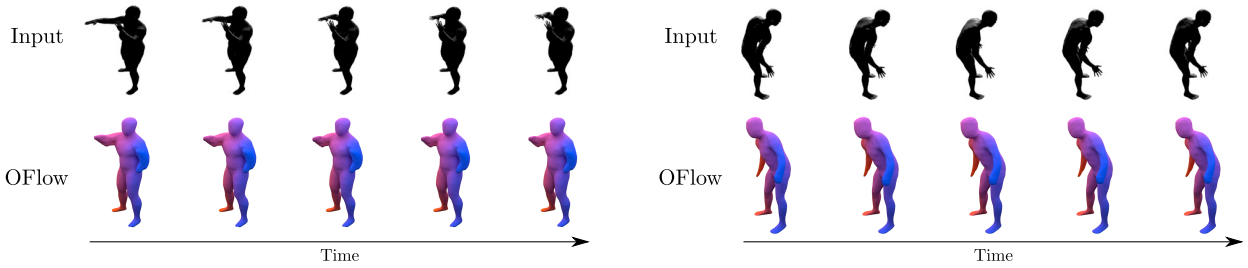


Figure 9: **Interpolation Results.** The figure shows two examples from the test set of the interpolation experiment on the D-FAUST dataset. It shows linear interpolation (left) and the output of OFlow (right) conditioned on the start and end point clouds. For both examples, the qualitative differences are most profound in the arm and hand region. In contrast to linear interpolation, OFlow is able to better preserve the lengths and shapes of the arms and hands over the whole sequence.



(a) High frequency motion failure case.

(b) Extreme lighting condition failure case.

Figure 10: **Failure Cases for 4D Reconstruction from Image Sequences.** We show five equally spaced times steps for the input and the output of OFlow of two examples from the 4D reconstruction from image sequences experiment on the D-FAUST dataset. While in Fig. 10a OFlow is not able to fully recover the high frequency motion, the extreme lighting conditions and the occluded left arm in Fig. 10b cause a nearly static output. As OFlow is able to recover the static part of the 3D geometry, we assume that incorporating local image features would particularly improve the motion reconstruction.

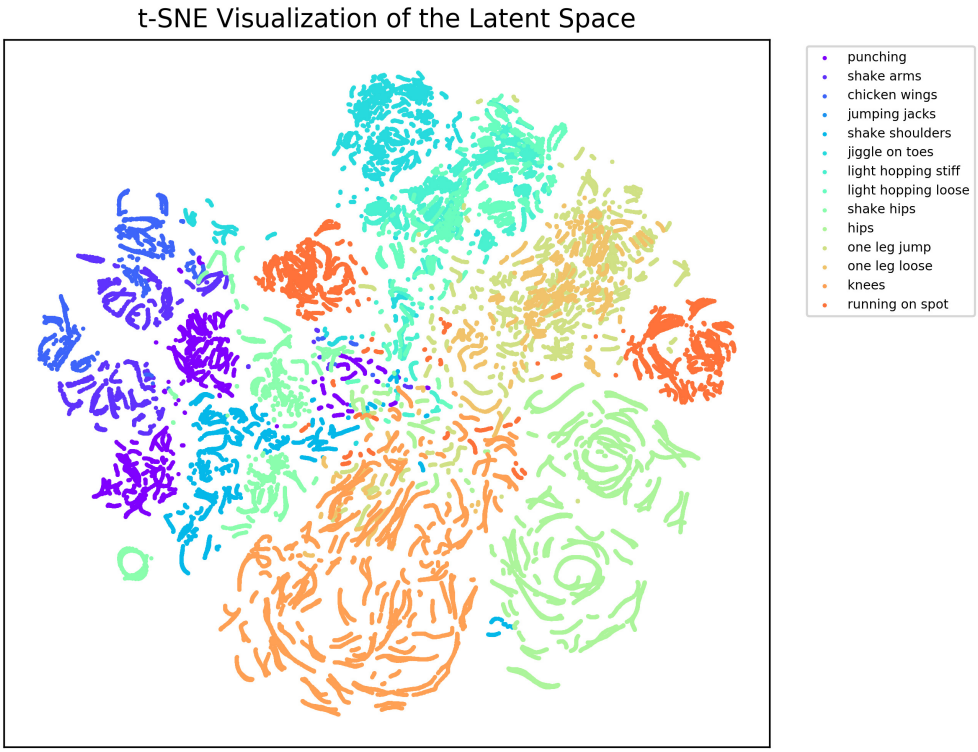
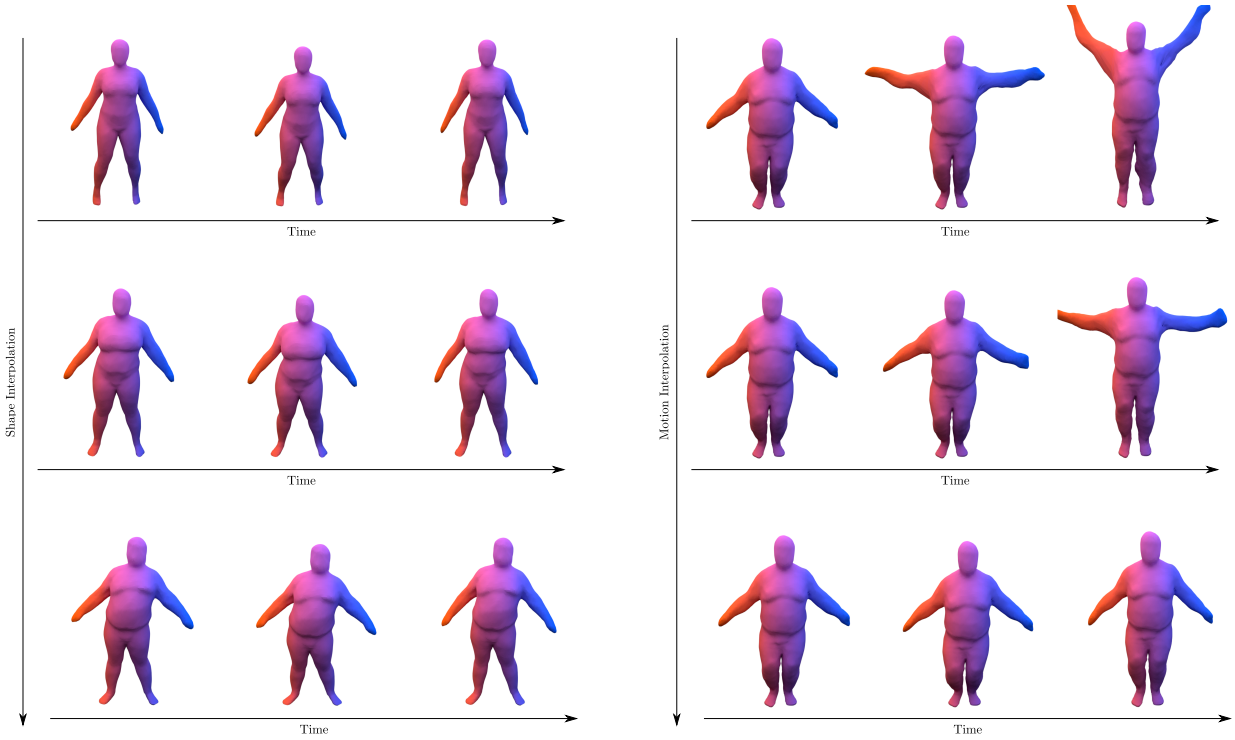


Figure 11: **t-SNE Visualization.** The figure shows a 2D t-SNE embedding of the motion codes of training examples of the D-FAUST dataset. The small line structures are formed by similar sub-sequences which are a result of our sub-sampling process of the D-FAUST dataset. The clustering shows that OFlow is able to learn a meaningful representation of the motions. The overlapping of different clusters often makes sense, e.g. “one leg jump” and “one leg loose” or “light hopping stiff” and “light hopping loose.”



(a) Shape Interpolation with Fixed Motion.

(b) Motion Interpolation with Fixed Shape.

Figure 12: **Latent Space Interpolations.** In Fig. 12a we show three equally spaced steps of a latent shape interpolation with fixed motion. Similarly, in Fig. 12b we show three equally spaced steps of a latent motion interpolation with fixed shape. The figures show that OFlow is able to learn a meaningful latent representation of both the shape and the motion.

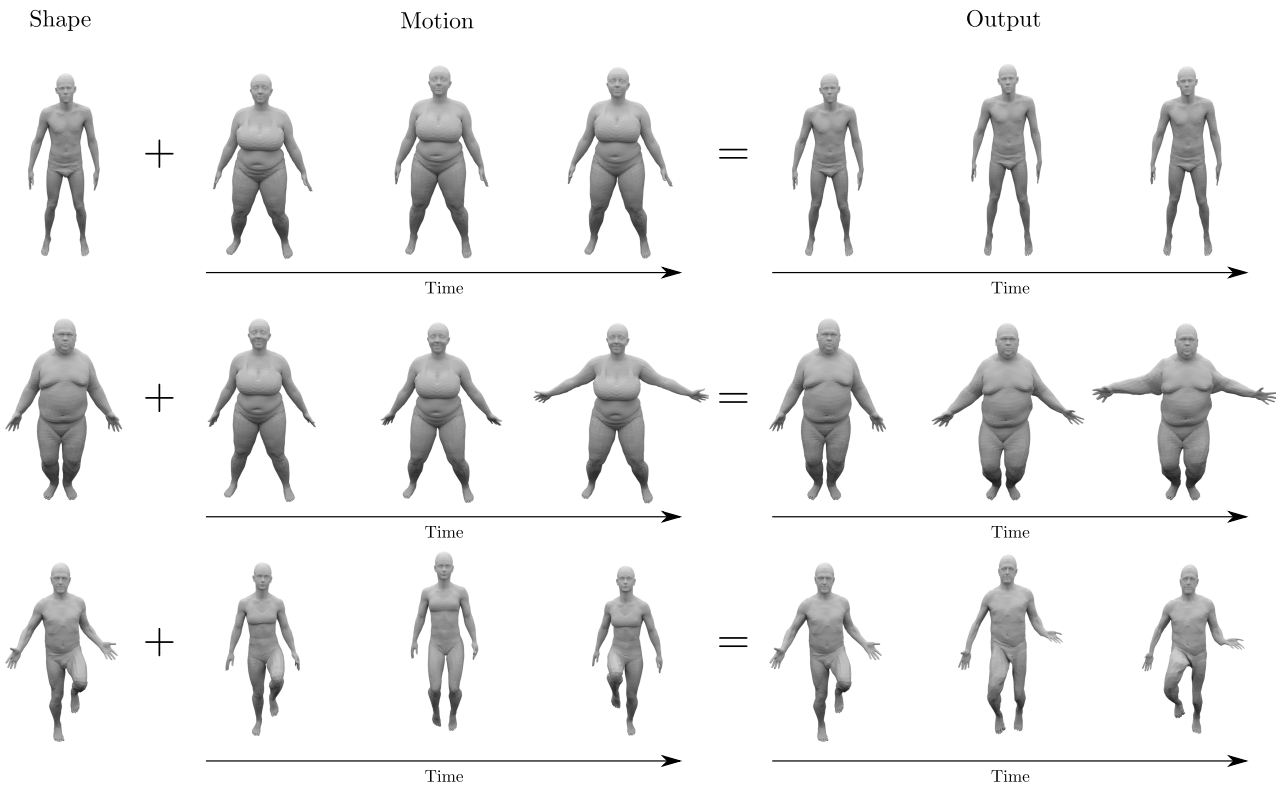


Figure 13: **Motion Transfer.** We show three examples of the motion transfer experiment. We take a start shape (first column) and encode the motion from another sequence (second column) to transfer this motion to the shape (third column). We see that OFlow is able to transfer the motion to another shape reasonably well despite changes in topology and pose.

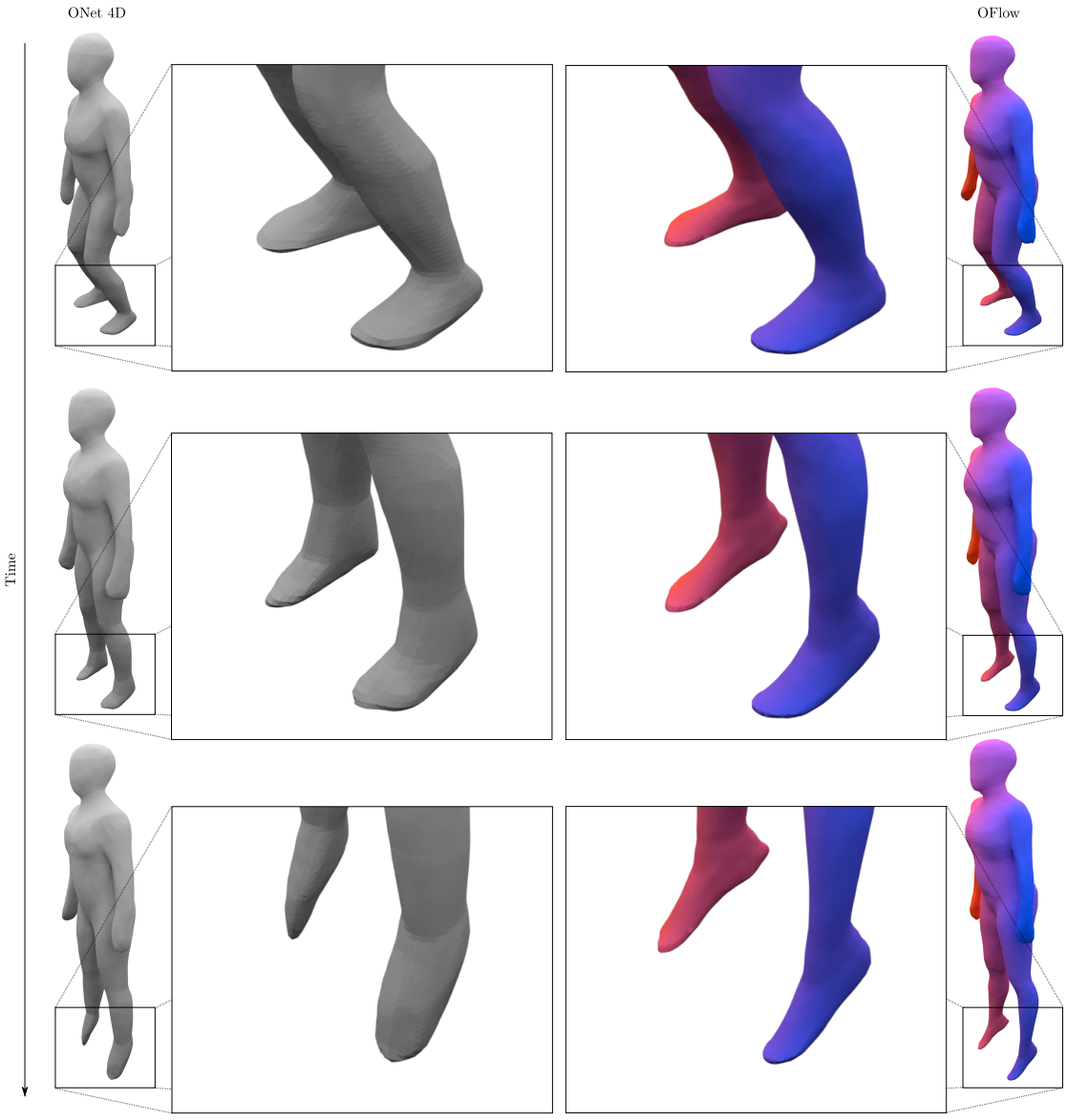


Figure 14: **Comparison of ONet 4D and OFlow.** The figure shows three equally spaced time steps between 0 and 1 of the output of ONet 4D (left) and OFlow (right) for an example from the test set of the point cloud completion experiment on the D-FAUST dataset. While ONet 4D predicts occupancy probabilities in 4D space, OFlow propagates the predicted 3D shape information from time 0 forward in time. As a result, OFlow better preserves the geometry over the whole sequence, here shown in greater detail for the foot area.

References

[1] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 4

[2] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 4

[3] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics (JCAM)*, 6(1):19–26, 1980. 2

[4] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[5] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[7] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. 10

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 2

[9] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 5

[10] Jooyoung Park and Irwin W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991. 5

[11] L. S. Pontryagin, V. G. Boltyanshii, R. V. Gamkrelidze, and E. F. Mishenko. *The Mathematical Theory of Optimal Processes*. John Wiley and Sons, 1962. 4

[12] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[13] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: modeling and capturing hands and bodies together. *ACM Trans. Gr.*, 36(6):245:1–245:17, 2017. 12