Temporal Interpolation of Dynamic Digital Humans using Convolutional Neural Networks

Irene Viola*, Jelmer Mulder*, Francesca De Simone* and Pablo Cesar*†
*Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
†TU Delft, Delft, The Netherlands
Email: {name.surname}@cwi.nl

***Abstract*—In recent years, there has been an increased interest in point cloud representation for visualizing digital humans in cross reality. However, due to their voluminous size, point clouds require high bandwidth to be transmitted. In this paper, we propose a temporal interpolation architecture capable of increasing the temporal resolution of dynamic digital humans, represented using point clouds. With this technique, bandwidth savings can be achieved by transmitting dynamic point clouds in a lower temporal resolution, and recreating a higher temporal resolution on the receiving side. Our interpolation architecture works by first downsampling the point clouds to a lower spatial resolution, then estimating scene flow using a newly designed neural network architecture, and finally upsampling the result back to the original spatial resolution. To improve the smoothness of the results, we additionally apply a novel technique called neighbour snapping. To be able to train and test our newly designed network, we created a synthetic point cloud data set of animated human bodies. Results from the evaluation of our architecture through a small-scale user study show the benefits of our method with respect to the state of the art in scene flow estimation for point clouds. Moreover, correlation between our user study and existing objective quality metrics confirm the need for new metrics to accurately predict the visual quality of point cloud contents.**

***Index Terms*—Cross Reality, point cloud, temporal interpolation, digital humans, scene flow estimation**

I. Introduction

Cross Reality (XR), a collective term for a range of techniques, such as Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR), has seen in recent years a surge of popularity, thanks to the technological innovations that have made it possible to bring together the digital and physical world. One representation that is frequently used in XR technologies is that of a point cloud: a structure that models volumetric visual data as a set of individual points in space [1], [2].

Points clouds do not require 3D reconstruction to be rendered, thus making them suitable for real-time systems. However, due to their large volume of data, concessions have to be made either in spatial resolution (the amount of points in each frame) or in temporal resolution (the frame rate) in order to match the bandwidth requirements of existing transmission systems. Low frame rate sequences are generally considered to be less pleasant to look at than higher frame rate sequences [3], and jerkiness is often perceived in them. Thus, simply reducing the temporal resolution is likely to have a strong negative effect on the Quality of Experience (QoE). However, temporal interpolation can be used to increase the frame rate, by predicting frames in between existing frames. It then becomes possible to transmit or capture a point cloud sequence at a relatively low frame rate, and perform temporal interpolation on the other end of the channel in order to achieve the desired frame rate. Such reconstruction can drastically reduce the bandwidth requirements of streaming and capturing dynamic point cloud sequences, allowing this bandwidth to be used for transmitting higher spatial resolution.

In this paper, we are interested in how temporal interpolation of digital humans, represented using point clouds, can be achieved. In particular, the research question we aim to address is how to effectively design a neural network architecture capable of performing temporal interpolation on point cloud data representing humans. Temporal interpolation through neural networks has been extensively used in 2D video coding [4]. However, performing temporal interpolation on digital humans presents a unique set of challenges, which makes the adoption of existing techniques particularly complex. Specifically, learning on point cloud data requires rethinking of common neural network approaches, in order to be adapted for unstructured data. Moreover, the high number of points composing a point cloud, and the fact that such number is usually varying within different frames of a dynamic sequence, may pose a challenge. Some networks have been proposed that deal with learning with point cloud data such as networks for classification and segmentation [5], [6]. However, their approach might not be suitable for our needs, as they were designed and optimized with different tasks in mind. Notably, temporal interpolation of human movement requires different approaches than for rigid motion, which is where most of the previous research has been focused [7]. In order to effectively learn the characteristics of human movement, a large data set of dynamic digital humans is needed. Yet, real-world dynamic point cloud data sets are limited in availability. Furthermore, in order to implement a supervised-learning solution, a reliable metric to estimate the goodness of the performed interpolation is needed. Several objective quality metrics for point clouds have been proposed in the past; nonetheless, studies have repeatedly shown how they poorly correlate with users' perception [8].

In this paper, we report the design of a neural network

This paper was partly funded by the European Commission as part of the H2020 program, under the grant agreement 762111, "VRTogether" (http://vrtogether.eu/).

architecture that is capable of performing temporal interpolation on dynamic point cloud sequences. In order to train the network, we created a synthetic data set by applying animations to models of human bodies. We compare our results against the state-of-the-art in scene flow estimation for point clouds. We also compare our results against the non-interpolated input sequences, in order to analyze the benefit of temporal interpolation. Lastly, we conduct a small-scale user study for a subjective evaluation of our algorithm. Results demonstrate that our method can be successfully used to achieve temporal interpolation of digital humans, showing clear gains with respect to state-of-the-art approaches in point cloud scene flow estimation. To the best of our knowledge, this is the first work focusing on performing scene flow estimation for non-rigid human motion using point cloud representation.

II. Related work

Temporal interpolation has already been successfully used in 2D video coding to increase the frame rate of existing sequences. In particular, neural network-based approaches have been adopted to tackle the problem [9]–[12], and now represent the state-of-the-art in video frame interpolation [4]. However, traditional neural network approaches, like convolution, are not always directly applicable on data that does not exhibit a regular, grid-like structure, such as point clouds. Several strategies have been implemented in the literature to deal with learning on point clouds. Some of them involve converting the irregular structure into an intermediate format on which traditional learning techniques can be applied. Among others, view-based methods generate a set of 2D views of the input point cloud, on which 2D-based learning techniques can be applied [13]. Similar to view-based methods, volumetric methods can learn on point cloud data by first converting the point clouds to an intermediate format. In their case, this intermediate format is a volumetric representation, for example an occupancy grid [5], [6] or a KD-tree [14].

Other methods have focused on learning directly on irregular point cloud data. PointNet [15] works by first feeding all the input points through a shared Multi-Layer Perceptron (MLP), creating a local feature vector for each point. Next, a symmetric function (for example, max pooling) is applied along the first axis, resulting in a global feature vector that is invariant to the order of the input. In the case of the classification network, this global feature vector is fed through another MLP which then produces the output scores. In the case of the segmentation network, the global feature vector is concatenated to each of the points, which are then fed through more MLPs to generate the output scores. This ensures that the segmentation scores take into consideration both local features and global features. Its successor, PointNet++ [16], leverages local neighbourhood structure by first partitioning the input sets into overlapping local regions, after which the original PointNet architecture is used as a building block to extract features from these local regions. This process is repeated hierarchically to generate increasingly high-level features. While PointNet++ achieved state-of-the-art results at the time, it still uses PointNet, which means points in local regions are still processed independently and, as a consequence, it does not consider relationships among the points. Dynamic Graph CNN (DGCNN) [17] aims at improving performance by applying convolutions on both input points and local neighbourhoods, which are constructed through a k-nearest neighbour graph. PointCNN [18] uses k-nearest neighbour search and MLPs to learn a transformation χ to weigh the input features and permute the input points into a latent and potentially canonical order. After the input points have been transformed using the learned χ-transformation, convolution can be applied.

Learning on point clouds has been successfully adopted to perform scene flow estimation, which can be used to achieve temporal interpolation. FlowNet3D [7] uses convolution layers from PointNet++ [16] to learn point features from two input point cloud frames. The features are then merged using a novel flow embedding layer, and the scene flow is refined through an up-convolution layer. The network is trained with synthetic scene flow ground-truth data, which proves to transfer well to the real-world data from the KITTI scene flow data set [19]. However, the model is only evaluated on data featuring rigid motions, thus not including local deformation. Rigid Scene Flow [20] focuses on point clouds obtained from 3D LiDAR scans. Again, the assumption is that all transformations are rigid, and thus that there is no local deformation. This assumption holds up for LiDAR scans, where the focus might typically be on moving objects such as cars. While the algorithm is also tested on non-rigid data, there is no convincing comparison to other work.

III. Interpolation architecture

The high-level overview of the interpolation architecture is depicted in Figure 1. Two 6-dimensional point clouds, pc_1 and pc_2, each having spatial position and color information, are fed to the network. As the interpolation network uses nearest-neighbour search, and thus scales $\mathcal{O}(n^2)$ with n points in computation time, only point clouds of limited spatial resolution can be used. For this reason, pc_1 and pc_2 are first uniformly downsampled to $n_d = 2048$ points each. The interpolation network then takes the two downsampled point clouds and estimates the corresponding scene flow. Next, the scene flow estimation is upsampled back to the original resolution using 3D interpolation. Lastly, a neighbour snapping algorithm is applied to increase the smoothness of the scene flow estimation.

In the following paragraphs, we will present the components of our algorithm. Subsection III-A introduces the main components of our interpolation network, subsection III-B details the 3D upsampling algorithm, while subsection III-C focuses on the neighbour snapping algorithm.

A. Interpolation network

The interpolation network is the central part of our architecture. It consists of two modules, namely point matching and flow refinement, which compute the scene flow estimation between two input point clouds.

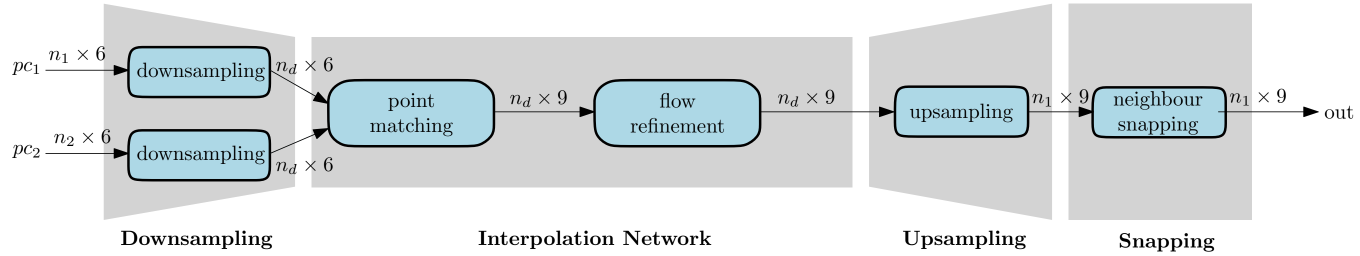


Fig. 1: High-level structure of the temporal interpolation architecture.

The point matching module takes as input two point clouds, $\mathcal{P} = \{p_1, ..., p_{N_p}\}$ and $\mathcal{Q} = \{q_1, ..., q_{N_q}\}$, and learns a soft mapping between points from $\mathcal{P}$ to points from $\mathcal{Q}$. That is, it learns a weight $w_{i,j}$ for all pairs $i \in \{1, ..., N_p\}$ and $j \in \{1, ..., N_q\}$, where weight $w_{i,j}$ describes how much point p_i is matched to point q_j. We impose that $\forall i \in \{1, ..., N_p\} \mid \sum_{j=1}^{N_q} w_{i,j} = 1$. The weights are used to compute the first scene flow estimation $\mathcal{S}$:

$$\mathcal{S} \leftarrow \left\{ \sum_{j=1}^{N_q} w_{i,j} * (q_j - p_i) \right\}_{i=1}^{N_p} \tag{1}$$

To learn the weights, we first find for each point p in pc_1 the k-nearest neighbours, where k is a hyper-parameter determining the neighbourhood size. Higher values of k mean more neighbours have to be considered, and thus require more computation power. Lower values of k mean smaller neighbourhoods, making it less likely to find the true semantically corresponding point, in which case the network will likely not output a good scene flow estimation. After empirical testing, the value $k = 50$ was selected, to provide a good balance between computational complexity and robust scene flow estimation. For each point p_n belonging to the neighbourhood of p, we select the features $\{p, p_n, p_n - p\}$. The features p and p_n provide information about the absolute position of the points, whereas the feature $p_n - p$ provides information about the relative position of the neighbouring points. These features are fed through multiple MLPs. In the last MLP layer, we use only a single filter, in order to obtain an output of dimensions $n_1 \times k$. We normalize the sum of outputs to 1 for each base point, and then use these values as weight w to derive the estimated scene flow as described in Equation 1.

The flow refinement layer takes as input one point cloud $\mathcal{P} = \{p_1, ..., p_{N_p}\}$ with scene flow estimation $\mathcal{S} = \{s_1, ..., s_{N_p}\}$, and refines the scene flow estimation. Similarly to the point matching layer, it does this by learning a set of weights, where weight $w_{i,j}$ describes how much the refined scene flow estimation of point p_i will depend on the original scene flow estimation s_j. The refined scene flow estimation $\mathcal{S}'$ is then calculated:

$$\mathcal{S}' \leftarrow \left\{ \sum_{j=1}^{N_p} w_{i,j} * s_j \right\}_{i=1}^{N_p} \tag{2}$$

The weights are learned by first applying EdgeConv [17] to learn high-level features for each scene flow estimation, in order to detect and correct outliers. After the EdgeConv layer, two MLP layers are applied. Similarly to the point matching module, the sum of outputs is normalized to 1 for each base point, and the values are used as weight w to compute the refined scene flow as described in Equation 2.

B. Upsampling

In order to obtain a high resolution scene flow estimation, we upsample our low resolution estimation through 3D interpolation [16]. Given a high resolution point cloud $\mathcal{P} = \{p_1, ..., p_{N_p}\}$, the downsampled low resolution point cloud $\mathcal{P}' = \{p'_1, ..., p'_{N_{p'}}\}$, and the low resolution scene flow estimation $\mathcal{S}' = \{s'_1, ..., s'_{N_{p'}}\}$, we calculate the upsampled scene flow estimation $\mathcal{S}$ as described in Equation 3. Here w is a normalized inverse-distance weight function, assigning higher weights to points that are closer, and $\mathcal{N}_k(p_i)$ describes the k-nearest neighbourhood of p_i.

$$\mathcal{S} \leftarrow \left\{ \sum_{p'_j \in \mathcal{N}_k(p_i)} w(p_i, p'_j) * s'_j \right\}_{i=1}^{N_p} \tag{3}$$

C. Neighbour snapping

In order to provide more smoothness in the temporal interpolation, we propose neighbour snapping, a technique that makes an existing scene flow estimation fully-connected. The pseudo-code of the algorithm is shown in Algorithm 1.

Considering two point cloud frames $\mathcal{P}^i$ and $\mathcal{P}^{i+1}$, and their estimated scene flow $\mathcal{F}^{i \rightarrow i+1}$, to achieve forward-connectedness, we find for each point in $\mathcal{P}^i$ the point in $\mathcal{P}^{i+1}$ closest to its scene flow target location, and let that point be the new scene flow target location. Subsequently, to ensure backward-connectedness, we find all points in $\mathcal{P}^{i+1}$ that are not backward-connected yet, and we change their spatial location to that of the nearest point in $\mathcal{P}^i$. The scene flow for these points is adjusted accordingly, so that the target location remains the same.

IV. Validating Experiment

In this section, we outline the procedure followed to evaluate the performance of our architecture. Specifically, section IV-A illustrates how we acquired the data set used for training and testing, whereas section IV-B details how the training was carried out. In section IV-C, the test conditions are presented,

Algorithm 1 Neighbour snapping

```
procedure SNAP(pc1,pc2)
    for point p1 in pc1 do
        target ← p1.xyz + p1.sceneflow
        target_nn ← nearest_neighbour(pc2, target)
        p1.sceneflow ← target_nn.xyz − p1.xyz
    end for
    for point p2 in pc2 do
        nn ← nearest_backward_connected_neighbour(pc2, p2)
        p2.sceneflow ← p2.xyz + p2.sceneflow − nn.xyz
        p2.xyz ← nn.xyz
    end for
end procedure
```

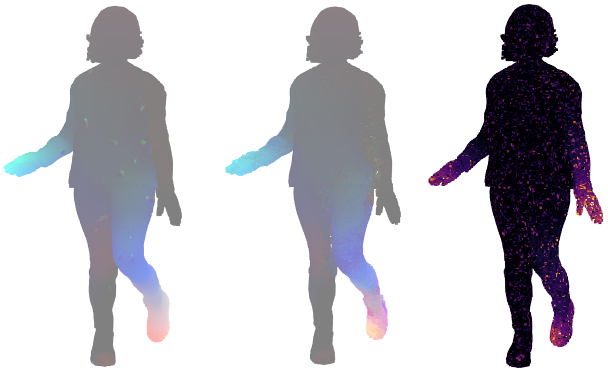
followed by a description of the objective quality metrics and the user test in sections IV-D and IV-E, respectively. Finally, section IV-F describes the results of the performance evaluation.

A. Data set

Due to the lack of sufficiently large point cloud datasets for training and evaluation, we created a synthetic data set of animated human bodies, using the online service Mixamo[1]. In Mixamo, it is possible to select one between a number of default character models (or upload a new one), and apply one of the many available animations. After, we used the 3D animation software Blender[2] to render the animations and to obtain one mesh per frame. Finally, the meshes were converted to point clouds by randomly sampling points from the faces of the mesh. For instructions on how to use our data set, refer to our Github page[3]. The training data set consists of 5 models, each having 12 animations applied on them, for a total of 60 sequences. Each sequence ranges in length from 30 to 250 frames, for a total of 10 590 individual frames, giving 10 522 pairs of consecutive frames. Furthermore, data augmentation was applied on the data, by flipping, scaling, rotating, shuffling and subsampling the frames.

B. Training and hardware

The training of the network was split in two phases. In the first phase, only the first module of the interpolation network, namely the point matching layer, was trained; whereas in the second phase, the flow refinement module was trained, after freezing the weights from the point matching layer. The End Point Error (EPE), defined as the average L_2 distance between the estimated and ground truth scene flow, was selected as the loss function for both learning phases. Out of the 10 522 pairs of consecutive frames forming the dataset, 8152 pairs were used as training set, 500 for validation, and 1870 were reserved for the test set.



(a) Ground truth (b) Estimated (c) Error

Fig. 2: Visualization of the ground truth scene flow (left), along with the estimated scene flow (middle), and relative error (right).

The architecture was implemented in Python using the Tensorflow framework, and is publicly available on Github[3]. It contains modules for training, evaluation and inference. All experiments were carried out on a machine equipped with an Intel i7-7800X CPU running at 3.50 GHz, 16GB RAM, and an NVIDIA GeForce RTX 2080 Ti.

C. Test conditions

In order to test our scene flow estimation algorithm, 8 additional sequences, obtained from 2 models with 8 unique animations applied to them, were used. Namely, the animations *Dancing*, *HipHop*, *LookingAround* and *WaveDance* were applied to the model *shae*, whereas the animations *fight*, *roar*, *samba* and *yelling* were applied to the model *malcolm*. The test sequences at full frame rate, 48 frames per second (fps), with relative ground truth scene flows, were used as references for the performance assessment. The sequences were reduced to a lower frame rate (8 fps) to perform the temporal interpolation.

We compare the results of our temporal interpolation algorithm to the state of the art in point cloud scene flow estimation, namely Flownet3D [7]. The network has been retrained using the same training data set used in our architecture. When reporting their results, we apply their scene flow estimation algorithm on the downsampled point clouds, since Flownet3D cannot process large point clouds in reasonable time. Their estimated scene flow is then fed to our upsampling and neighbor snapping algorithm, to obtain the final result. Moreover, in order to assess whether temporal interpolation would be beneficial in improving the visual quality of the scene, we assess our method against the low frame rate sequence, which was used as input in both interpolation algorithms.

D. Objective quality evaluation

In order to evaluate the performance of our scene flow estimation algorithm, several objective metrics were used. In particular, the average EPE was computed between the estimated and the ground truth scene flow (see Figure 2).

[1] https://www.mixamo.com
[2] http://www.blender.org
[3] https://github.com/jelmr/pc_temporal_interpolation

TABLE I: Objective metrics, computed on the original point cloud size, with respect to the ground truth.

	Low fps	Flownet3D	Ours
`EPE`	**1.295**	2.183	1.533
`Accuracy (0.5 cm)`	**0.350**	0.194	0.323
`Accuracy (1.0 cm)`	**0.546**	0.400	0.522
`Accuracy (2.0 cm)`	**0.806**	0.637	0.753
`po2point_xyz_PSNR`	56.18	53.01	**58.35**
`po2point_rgb_PSNR`	**74.93**	72.14	73.75
`po2plane_PSNR`	94.19	88.95	**104.75**
`VIF (projection)`	0.710	0.653	**0.809**

Furthermore, the accuracy was computed as the portion of points for which the EPE is less than a certain threshold, in our case fixed to $0.5, 1.0$ and 2.0 cm. Accuracy ranges from 0.0 to 1.0, with a higher score being better. Moreover, several point cloud objective metrics were taken from the literature. In particular, the PSNR of the spatial component (xyz) and the color component (RGB) of the point-to-point distortion metric (po2point_xyz_PSNR and po2point_rgb_PSNR), and the PSNR of the point-to-plane distortion metric were computed between the original and interpolated point cloud frames, whereas the VIF metric was used to compute the projected distortion between original and interpolated frames [8].

E. Subjective quality evaluation

In order to better assess the performance of the algorithms with respect to the ground truth, we performed a small user study with 8 participants. Four categories were evaluated: ground truth (original point cloud sequences at 48 fps), Low fps (no interpolation at 8 fps), Flownet3D, and Ours, both interpolating from 8 to 48 fps. Participants were shown video sequences depicting pre-rendered dynamic point cloud sequences, and were asked to rate the quality of the motion of the video between 1 (terrible) and 7 (excellent). They were instructed to focus on the quality of the motion, rather than the visual quality of the model.

The Mean Opinion Score (MOS) was computed for each sequence by averaging all the scores given by the subjects, along with the Confidence Intervals (CIs) at 95% level. To understand whether the performance of the algorithms with respect to one another would bear statistical significance, we additionally performed a one-sided Welch's t-test at 5% significance level, with the following hypotheses:

$$H0 : MOS_A \leq MOS_B$$
$$H1 : MOS_A > MOS_B,$$

in which A and B are the algorithms under comparison. The test was performed separately for each sequence. If the null hypothesis were to be rejected, then it could be concluded that algorithm A performed better than algorithm B for the given sequence, at a 5% significance level.

F. Results

Table I summarises the results of the objective performance assessment. It can be seen that for the metrics based on scene flow error, i.e. `EPE` and `Accuracy`, Low fps is the clear winner, followed by the results from Ours. Regarding point cloud metrics, values obtained with the point-to-point metrics are very close, with ours being slightly ahead on the spatial component, and Low fps again being ahead on the color component. However, Ours has the best performance when considering both the point-to-plane metric and the VIF projection metric, which has been reported by Torlig et al. [8] to have the strongest correlation with subjective user ratings on point clouds of human bodies.

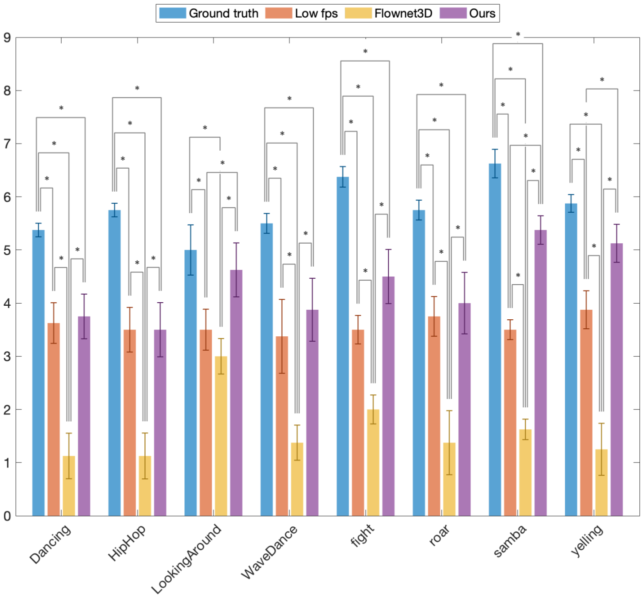


Fig. 3: MOS with relative CIs for each sequence. The asterisk indicates statistical difference at 0.05 significance level.

Results of the user evaluation for each test sequence are shown in Figure 3. Results clearly show that state-of-the-art algorithms for scene flow estimation, such as Flownet3D, perform sub-optimally when given the task of interpolating digital humans. The lackluster performance can be attributed to the fact that the algorithm was designed for rigid motions, and adapts poorly to scene flow estimation for human movement, even when trained with human motion sequences. Our proposed algorithm is shown to outperform it for all the sequences. Results also show that our method is considered either equal or better than the Low fps solution. In particular, the Low fps solution is never considered better than our proposed method in a statistically significant way, whereas our solution is rated as significantly better for 37.5% of the contents, according to the Welch's t-test applied on the scores. Moreover, our solution is considered statistically equivalent with respect to the ground truth for 25% of the contents.

V. Discussion

In this section, the results presented in the previous section are discussed and analysed. Specifically, in section V-A we examine the correlation between objective metrics and user perception. In section V-B, we debate on the use of transfer learning for temporal interpolation on point cloud data. Finally,

TABLE II: Performance metrics computed between objective metrics and user scores, for each regression model. Best performance for each model is highlighted in bold.

	Linear				**Cubic**			
	PLCC	**SRCC**	**RMSE**	**OR**	**PLCC**	**SRCC**	**RMSE**	**OR**
EPE	0.439	0.347	1.131	0.625	0.547	0.471	1.053	0.750
Accuracy (0.5 cm)	0.382	0.413	1.163	0.750	0.387	0.413	1.160	0.792
Accuracy (1.0 cm)	0.358	0.342	1.175	0.667	0.483	0.409	1.102	0.792
Accuracy (2.0 cm)	0.403	0.325	1.152	**0.583**	0.594	0.547	1.012	0.708
po2point_xyz_PSNR	0.576	0.627	1.029	0.750	0.632	**0.682**	0.975	0.667
po2point_rgb_PSNR	0.373	0.412	1.168	0.792	0.537	0.492	1.061	**0.583**
po2plane_PSNR	**0.636**	**0.660**	**0.972**	0.750	**0.645**	0.666	**0.962**	0.708
VIF (projection)	0.575	0.620	1.030	**0.583**	0.641	0.612	0.966	0.667

in section V-C we clarify and discuss limitations and future steps.

A. User perception and objective metrics

Results of the performance evaluation through objective metrics are not in agreement with the data collected through our user study. While the user study shows that the QoE of the users can be improved by performing temporal interpolation, results from the objective metrics would discourage such an attempt, as the non-interpolated sequence often comes as the top performing one. In fact, it might be considered surprising that the Low fps approach (which simply repeats the last frame) would achieve the highest score for scene flow metrics. However, the phenomenon can be explained when considering the actual meaning of scene flow error metrics: calculate the average of some notion of error over all points. In digital human motion, the largest movements will likely be circumscribed to a subset of points (e.g., the points forming hands or feet), whereas the majority of the points are unlikely to present large motion (e.g., the torso). Thus, the Low fps method, which sets the scene flow to 0, will have a good performance on a large number of points, which can dominate over the small amount of points with large error. Results can then be justified considering that non-rigid motion might lead to a large number of points presenting small displacements, which skews the performance of the metrics to favour conservative movement estimation. Moreover, as temporal interpolation involves geometry more significantly than color, it is justifiable that color metrics such as po2point_rgb_PSNR would prefer the Low fps approach.

To better understand the performance of the objective metrics in predicting the user perception of temporal interpolation for point clouds, several performance indexes were computed on the data. In particular, Pearson Linear Correlation Coefficient (PLCC), Spearman Rank Correlation Coefficient (SRCC), Root Mean Square Error (RMSE) and Outlier Ratio (OR) were chosen to account for linearity, monotonicity, accuracy and consistency, respectively, following ITU-T Recommendations P.1401 [21]. Linear and cubic fitting was applied on the results of the objective metrics before computing the indexes. Ground data results were excluded from the computation.

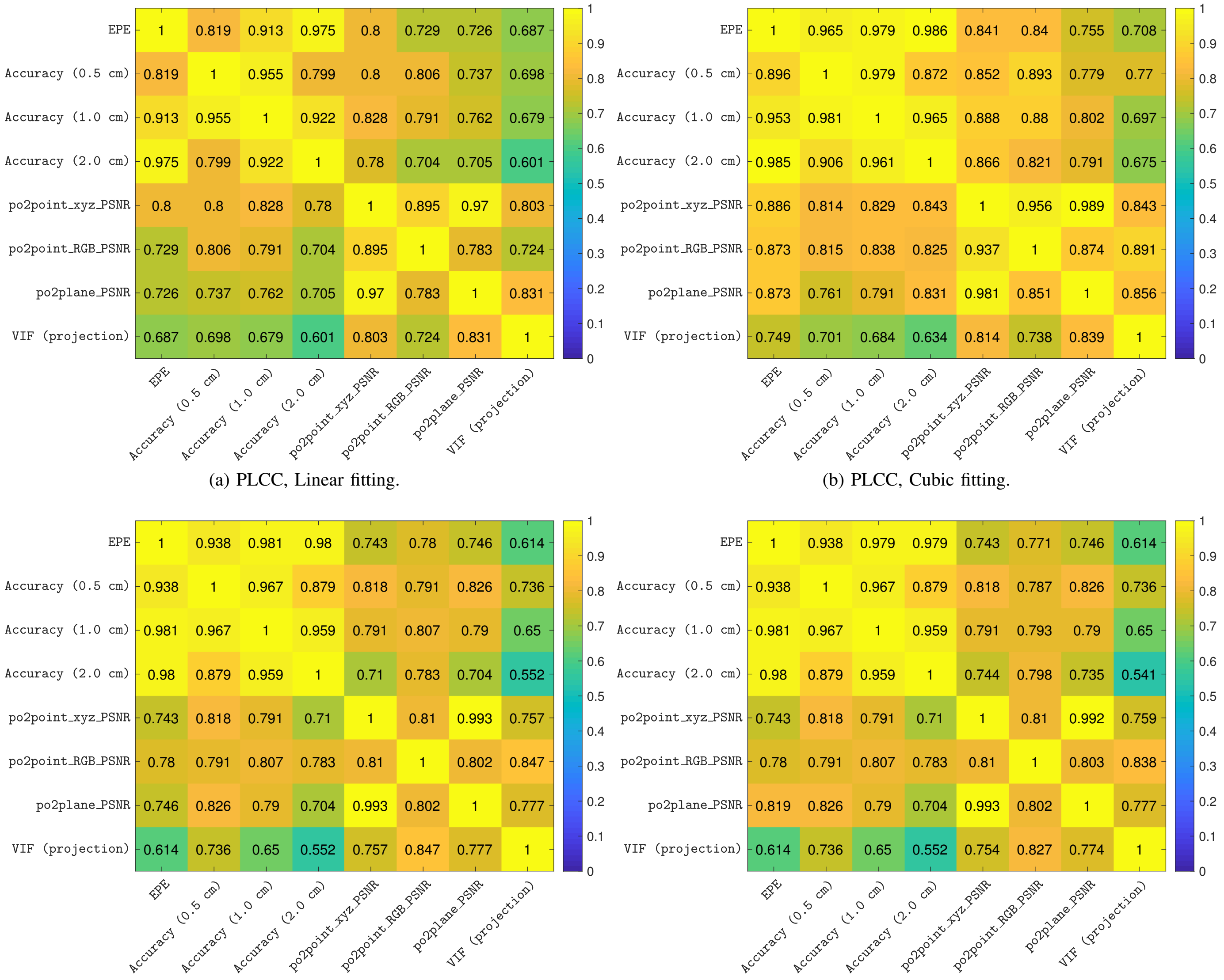
Results from the performance indexes are summarised in Table II. It can be observed that all metrics seem to correlate poorly with the results of the user study. In particular, scene flow error metrics, such as EPE and Accuracy, present very low correlation indexes, for both regression models. Metric po2point_rgb_PSNR also seems ill-suited to predict the visual quality of temporally interpolated point clouds. Among the point cloud geometry metrics, po2plane_PSNR appears to be the one that better correlates with the results of the user study, although the values obtained (PLCC of 0.645 and SRCC of 0.666 for cubic regression) are far from optimal.

Figure 4 depicts the PLCC and SRCC indexes computed between pairs of objective metrics, for linear and cubic fitting. As could be expected, scene flow error metrics are closely correlated. However, their correlation with point cloud metrics is not as high. In fact, for the two best-performing metrics in terms of correlation with user scores, the performance indexes indicate a level of disagreement with scene flow error metrics.

B. Reusability of network architecture

In the context of deep learning, transfer learning (i.e., reusing a model developed for one task as a starting point for completion of another task [22]) is often used to exploit well-know trained networks to achieve good performance. Many tasks lend themselves well to transfer learning. In the context of point clouds, for instance, many classification and segmentation solutions share the same core architecture, and vary only in the last couple layers. Examples of such solutions are PointNet [15] and DGCNN [17].

However, reusing such architectures to perform temporal interpolation led, in our experience, to poor results. This could be due to the fact that classification and segmentation are tasks that demand high level inference: the network will need to aggregate the local information of individual points into higher level features, which can then be used to classify or segment the point cloud. In the case of temporal interpolation with non-rigid motion, however, the local features play an important role, as different sets of points will exhibit different motion behaviour. This could explain why architectures that focus on inference of high level features do not show good performance when tackling the problem of temporal interpolation of human motion. Even the Flownet3D [7] architecture, which has been designed for scene flow estimation, does not perform



(a) PLCC, Linear fitting.

(b) PLCC, Cubic fitting.

(c) SRCC, Linear fitting.

(d) SRCC, Cubic fitting.

Fig. 4: Correlation coefficients computed between the objective metrics, for each regression model.

well on our data set of human bodies. We suspect this is because Flownet3D has been designed and evaluated for scene composed of multiple objects with rigid motion. Such data sets would require to separate the objects using segmentation, and then to estimate the motion of each object. Thus, when applying Flownet3D to non-rigid motion, the network will attempt to move the entire human body in one direction, leading to poor performance.

C. *Future work*

One of the main issues we faced in training a neural network architecture for temporal interpolation is the limited availability of public dynamic point cloud data sets representing human characters. In order to train a convolutional neural network, large amounts of data are needed. To solve the issue, we created our own synthetic data set, which allowed us to train and evaluate our architecture. However, the point clouds in our synthetic data set inevitably have different properties than real-world captured point clouds may have. For example, in our data set, each point in a frame has a unique one-to-one mapping to another point in other frames (that is, each point has exactly one semantically corresponding point in other frames). In real-world data sets, this will generally not be the case. Moreover in our data set, points will never change color, as they will only be translated spatially, whereas in real-world data sets colors are likely to change, for example due to variable lighting conditions.

The performance of neural networks depends on the type of data that is used in the training process. In our particular case, using a synthetic data set might lead the network to exploit the simplification brought by the properties of the data set, and thus not to learn how to interpolate real-acquired point clouds. Larger data sets of digital humans, acquired in a variety of settings, including different types of motions and variable lightning conditions, are thus needed to improve the performance of temporal interpolation architectures.

It is worth mentioning that scene flow error metrics, which were used to train our network, exhibit very low correlation with respect to the data gathered from the user study. In fact, the metrics seem to be overly conservative, as can be demonstrated by the Low fps solution being preferred for this type of metric. In order to counteract the effect of the metric, the training of our network was split in two separate phases (point matching and flow refinement), as training it in one step was preventing the point matching layer from learning properly.

More generally, the performance of neural network approaches will be affected by the choice of the loss function used to train it. Considering that none of the objective metrics considered in this study presented strong correlation with users' perception, it becomes apparent that existing metrics might not suitable to approximate the QoE of temporally interpolated human motion. Thus, adopting them in the training process of neural network approaches might lead to visually unpleasant results. New metrics need to be designed to better predict the visual quality of dynamic digital humans, in order to lead to better performing temporal interpolation networks.

VI. Conclusions

In this paper, we report the design of an architecture capable of performing temporal interpolation on dynamic point clouds representing digital humans. By transmitting point clouds in a lower frame rate and successively upsampling their frame rate using such an architecture on the receiving side, the bandwidth requirements of streaming dynamic point clouds can be reduced. Results of our performance evaluation show that temporal interpolation seems to be a promising solution to improve the QoE of digital humans. Further steps need to be implemented in order to improve the performance of existing algorithms, in order to achieve an acceptable QoE. We also show that objective metrics fail at capturing how users perceive temporal interpolation for digital humans. This is a particularly critical issue, since objective metrics are needed in order to train and evaluate interpolation architectures; the wrong choice of metric can thus lead to very unpleasant results. New, improved metrics are needed to successfully estimate the QoE of the interpolated sequences.

Acknowledgment

The work presented in this paper is based on the master thesis of Jelmer Mulder, which can be found in full at the following link: https://github.com/jelmr/pc_temporal_interpolation. This paper was partly funded by the European Commission as part of the H2020 program, under the grant agreement 762111, "VRTogether" (http://vrtogether.eu/).

References

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokua, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, March 2019.

[2] R. Mekuria, K. Blom, and P. Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, April 2017.

[3] A. Banitalebi-Dehkordi, M. T. Pourazad, and P. Nasiopoulos, "The Effect of Frame Rate on 3D Video Quality and Bitrate," *3D Research*, vol. 6, no. 1, p. 1, Dec 2014.

[4] H. Men, H. Lin, V. Hosu, D. Maurer, A. Bruhn, and D. Saupe, "Technical Report on Visual Quality Assessment for Frame Interpolation," *arXiv preprint arXiv:1901.05362*, 2019.

[5] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 922–928.

[6] T. Le and Y. Duan, "PointGrid: A Deep Network for 3D Shape Understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[7] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning Scene Flow in 3D Point Clouds," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] E. M. Torlig, E. Alexiou, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A novel methodology for quality assessment of voxelized point clouds," in *Applications of Digital Image Processing XLI*, vol. 10752. International Society for Optics and Photonics, 2018, p. 107520I.

[9] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[10] ——, "Video frame interpolation via adaptive separable convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[11] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1701–1710.

[12] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.

[13] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 945–953.

[14] R. Klokov and V. Lempitsky, "Escape From Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5099–5108.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[18] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution On X-Transformed Points," in *Advances in Neural Information Processing Systems*, 2018, pp. 828–838.

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

[20] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Rigid scene flow for 3D LiDAR scans," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1765–1770.

[21] ITU-T P.1401, "Methods, metrics and procedures for statistical evaluation, qualification and comparison of objective quality prediction models," International Telecommunication Union, July 2012.

[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.