Compression and Completion of Animated Point Clouds using Topological Properties of the Manifold

Linfei Pan
ETH Zurich
linpan@student.ethz.ch

Ľubor Ladický
ETH Zurich
lubor.ladicky@inf.ethz.ch

Marc Pollefeys
ETH Zurich / Microsoft
marc.pollefeys@inf.ethz.ch

Abstract

Recent progress in consumer hardware allowed for the collection of a large amount of animated point cloud data, which is on the one hand highly redundant and on the other hand incomplete. Our goal is to bridge this gap and find a low dimensional representation capable of approximation to a desired precision and completion of missing data. Model-less non-rigid 3D reconstruction algorithms, formulated as a linear factorization of observed point tracks into static shape component and dynamic pose, have been found insufficient to create suitable generative models, capable of generating new unobserved poses. This is due to the non-locality of the linear models, over-fitting to the non-causal correlations present in the data, which manifests in the reconstruction containing rigidly behaving not directly connected parts.

In this paper, we propose a new method that can distinguish body parts and factorize the data into shape and pose purely using topological properties of the manifold - local deformations and neighborhoods. To obtain localized factorization, we formulate the deformation distance between two point tracks as the smallest deformation along the path between them. After embedding such distance in low dimensional space, a clustering of embedded data leads to close to rigid components, suitable as initialization for fitting a model - a skinned rigged mesh, used extensively in computer graphics. As both local deformations and neighborhoods of a point are local and can be estimated only from the part of the animation, the method can be used to recover unobserved data in each frame.

1. Introduction

With the emergence of consumer products with stereo and depth cameras, a large amount of animated time-dependent point cloud data can be collected without much effort. Such data is on the one hand highly redundant and on the other incomplete, typically due to occlusion. Our goal is to develop algorithms for compression and completing of animated point clouds by finding suitable low dimensional representation under challenging conditions with fast deformations and movements, and a large portion of missing data, for example, parts not facing the camera. Challenging fast movements suggest that the problem can be solved only as a global optimization, which lays additional obstacles when it comes to practical implementation that does not computationally explode with the number of point tracks and the growing length of the animation.

The simpler variant of the problem assumes that the content and thus the model of the data (template) is known (for example a human body), and the problem can be posed as a mesh registration [6, 26, 7]. Similar formulations have lots of applications in medical imaging [11, 23, 3]. For the more general template-less problems, partially due to the lack of available 3D data, researches have mainly focused on non-rigid reconstruction from 2D point tracks [30, 12, 18, 9, 2, 31, 32]. These approaches formulate the problem as matrix factorization, linearly factorizing the observed point track data into static basis (loosely interpretable as a shape component) and dynamic time-dependent coefficients (loosely interpretable as a pose), effectively reducing the degree of freedom. There has been lots of effort in finding the most suitable reconstruction error term and regularizer [19, 16, 13, 34] and optimization techniques for handling missing data [10]. However, it has been shown that the methods perform poorly if missing data is structured (correlated) [1, 22] as in our case. Furthermore, linear models are not localized and tend to recover correlations in the data that are not causal. For example, for a walking person, movements of a left hand and a right foot are highly correlated, but in a meaningful generative model, capable of sampling new unobserved poses, their dependency is not desired. The same artifacts appear, when the factorization methods are applied to 3D data [20], because they treat observed points as independent disregarding pairwise correlations in the local neighborhoods. To tackle this problem, an ensemble of models (union of subspaces) has been proposed [27, 28], formulating the problem as an optimization with costs per model [14] (label costs), solved approximately

using alpha expansion [8]. To avoid over-fitting, the models have to be very simple (usually planar), and thus a large number of models are required, which makes them unable to extrapolate outside of observed data. Combining global and local models has been proposed in [21], which samples local models only from the global model solution. Generally all mentioned methods are computationally very expensive, as the optimization problem of the size that grows with the number of point tracks and length of the animation has to be solved. This motivated a research on non-rigid reconstruction by iterative warping of estimated geometry into the live frame [24]. The incremental nature of the method makes it applicable only to slow trackable movements of content with perceived fixed topology, as incorrect alignment would result in irrecoverable model corruption. Tracking-based methods as in [25, 5, 15] provides means for motion tracking, yet requiring manual initialization of the skinned model. Recently, deep learning based approaches [4, 33, 17] have been proposed to factorize meshes into pose and static shape.

We propose a novel method that can distinguish body parts and factorize the data into shape and pose purely using topological properties of the manifold - local neighborhoods and their deformations. To obtain such localized factorization, we formulate the deformation distance between two point tracks as the smallest deformation along the path between them. After embedding such distance in low dimensional space, a clustering of embedded data leads to close to rigid components, suitable as an initialization for fitting a model - a skinned rigged mesh, used extensively in computer graphics applications for deformable objects. We show how to estimate all its components - a static mesh, a set of limbs, their transformation matrices in each frame, and association weights of points with each limb. Both inputs of the methods - local deformations and neighborhoods of a point are local, and thus can be estimated only from the part of the animation, where each point is observable. This makes the algorithm applicable in the presence of missing data and can be used for the completion of 3D point tracks. We applied our method on artificial computer graphics models and real-world capture data and demonstrate that the method works robustly in both cases.

2. Non-rigid reconstruction and 3D data completion

2.1. Preliminaries

Deformable meshes in computer graphics applications, such as computer games or movies, are almost exclusively modeled and animated using skinned rigged meshes. This is due to the flexibility of this data representation to model a large class of objects including humans, animals or robots.

A skinned rigged mesh consists of a static mesh with an additional underlying skeleton, containing a set of joints, connected by limbs, forming a tree. Note that joints and limbs are interchangeable when it comes to transformations of the mesh - the transformation of a joint can be interpreted as that of its child limb. To create an animation, artists have to provide (or calculate using inverse kinematics) additional transformation matrices T_m^k for each limb $m \in \{1..M\}$ in each frame $k \in \{1..K\}$, where M is the number of limbs and K the number of frames of an animation.

Each point (vertex) x_i, $i \in \{1..N\}$, on the static input mesh is associated with each limb m, by weights $w_{im} \geq 0$, such that $\forall i \sum_{m \in \{1..M\}} w_{im} = 1$. Only a sparse set (typically 1-3) of weights for each limb is non-zero. Given the matrices T_m^k for a frame k, each point x_i transforms as:

$$x_i^k = f_m^k x_i = \sum_{m \in \{1..M\}} w_{im} T_m^k x_i, \tag{1}$$

where x_i^k is the position of a vertex x_i in the frame k and $f_m^k = \sum_{m \in \{1..M\}} w_{im} T_m^k$. In general, any other interpolation function (for example SLERP interpolation of a rotation component [29]) can be used. The goal of this paper is to recover an underlying set of limbs, static template mesh and weights w_{im} given a set of potentially noisy point tracks x_i^k. This is achieved without any additional assumption about the source and content of the data such as shape templates (it does not have to be human).

2.2. Relative deformation measures

Deformation measure on limbs Movement of limbs is the only source of deformation. Thus, we recover limbs by predicting them based on deformations measured in point data. Relative deformation between two neighboring limbs m and n is defined as follow. Given the transformation matrices T_m and T_n in frames k and l, the deformation is defined as deformation measure (distance) between relative transformations as:

$$D^{kl}(m,n) = |T_m^{kl}, T_n^{kl}|, \tag{2}$$

where $T_m^{kl} = (T_m^k)^{-1} T_m^l$ is the relative transformation of a limb m between k-th and l-th frame, and $|.|$ is any metric defined on transformation matrices. Note that if the transformation of two limbs is rigid, $T_m^{kl} = T_n^{kl}$, and thus their deformation distance is 0.

If we used the same definition (2) for non-neighboring limbs as in [20], we might recover non-causal correlation in the data. Using such low-dimensional models might better approximate data, but would lead to an impractical non-local representation. Intuitively, the deformation between two limbs according to equation (2) should be 0 only if the whole path in the skeleton between these two limbs transforms rigidly. This observation leads to the definition of the deformation between non-neighboring limbs as:

$$D^{kl}(m,n) = \sum_{a \in \{1..A-1\}} D^{kl}(P(a), P(a+1)), \tag{3}$$

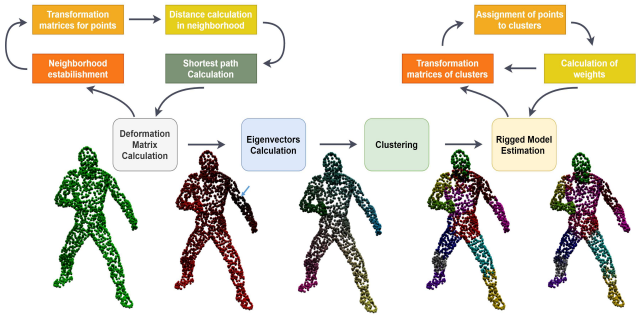

Figure 1. Description of the proposed non-rigid reconstruction algorithm. Given a set of 3D input point tracks the deformation distances between neighboring points are calculated using their estimated transformation matrices. The deformation costs between non-neighboring points are calculated using short paths. The second depicted image contains distances from the selected point on an elbow (pointed by the arrow). The distance matrix is embedded in a low-dimensional space by calculating eigenvectors. The third image shows the top three eigenvectors as RGB values normalized to [0, 255]. Spectral clustering is performed to obtain an initial set of limbs (depicted in the forth image), which are iteratively re-estimated together with association weights of points to each cluster. The final image shows the reconstruction with point colors calculated by blending cluster colors using association weights.

where $(P(1), P(2)..P(A))$ is the path in the skeleton between limbs $P(1) = m$ and $P(T) = n$. In case this deformation distance between two limbs is 0 for each pair of frames, the whole path between them should be recovered as a single rigid limb, as this is indistinguishable from but simpler than a set of rigidly transforming limbs.

Deformation measure on points The definition of the deformation distance can be adjusted to measure deformation between neighboring points in the point tracks x_i^k. We will denote that $x_j \in N(x_i)$ is in the neighborhood of x_i, if $\forall k : |x_i^k - x_j^k| < \epsilon$, where ϵ is a threshold chosen depending on the density of input points. We define the relative transformation t_i^{kl} of a point x_i between frames k and l as:

$$t_i^{kl} = arg\min_{T'} \sum_{j \in N_i} ||x_j^l - T' x_j^k||_0, \tag{4}$$

where $|.|_0$ is an L_0 pseudo-norm maximizing the number of inliers within a certain small δ threshold and can be estimated using RANSAC. The local deformation distance between neighboring points x_i and $x_j \in N(x_i)$ can be defined as:

$$d^{kl}(i,j) = |t_i^{kl}, t_j^{kl}|. \tag{5}$$

The total deformation between neighboring points over the whole animation $\bar{d}_{ij}$ is a simple sum over all frames:

$$\bar{d}_{ij} = \sum_{(k,l) \in P} d^{kl}(i,j), \tag{6}$$

and the total deformation between non-neighboring points x_i and $x_j \notin N(x_i)$ as:

$$\bar{d}_{ij} = \sum_{a \in \{1..A-1\}} \bar{d}_{p(a),p(a+1)}, \tag{7}$$

where $(p(1), p(2)..p(A))$ is the shortest path with respect to the deformation measure between points $p(1) = i$ and $p(T) = j$.

Relation between deformation measure on limbs and points Consider the shortest path between two points i and j, each placed near a pair of neighbouring joints of a limb m and n. We can assume, the points in the neighbourhood of the joint of a limb m transform purely according to the T_m^k with $w_{im} = 1$ and $w_{in} = 0$. As we get closer to joint of a limb n the weights will monotonically change, eventually reaching $w_{jm} = 0$ and $w_{jn} = 1$. The assumption of monotonicity implies that for a distance measure $\bar{d}$, satisfying condition (any L_1-like distance):

$$\bar{d}_{ij}(w_1 t^i + (1-w_1)t^j, w_2 t^i + (1-w_2)t^j) = |w_1 - w_2| \bar{d}_{ij}(t^i, t^j), \tag{8}$$

the relation between deformation $D(.)$ between neighboring limbs m and n, and deformation $\bar{d}_{ij}$ between points i and j purely transforming according to the transformation of limbs m respectively n is:

$$\bar{d}_{ij} = \sum_{(k,l) \in P} D^{kl}(m,n). \tag{9}$$

The same observation can be generalized to the distances between any pairs of points associated purely with a single limb under assumption, the shortest path between them will go through points purely associated with limbs along the path, which is a good approximation in practice. This is a key observation, as the relation between deformation allows us to recover the limbs just by studying and analyzing observed points tracks instead of unobserved limbs.

2.3. Low-dimensional embedding

Next we find a low-dimensional embedding of $\bar{d}$ in a coordinate space Z, in which the euclidian distance is approximately equal to the deformation distance. Finding such embedding is a well understood problem, known as multi-dimensional scaling or principal coordinates analysis. First double centering is applied:

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\bar{d}\mathbf{J}, \tag{10}$$

where $\mathbf{J} = \mathbf{I} - \frac{1}{N}11^T$. Then the matrix $\mathbf{B}$ is factorized using eigen-decomposition and the largest m eigenvalues with corresponding eigenvectors E_m are found. The m-dimensional embedding Z of the data is calculated as:

$$Z = E_m \Lambda_m^{1/2}, \tag{11}$$

where Λ_m is the diagonal matrix of top m eigenvalues The parameter m is chosen depending on the expected level of noise in the data characterized by a threshold λ_{min}, such that $\forall k : \lambda_k > \lambda_{min}$.

Computationally efficient implementation The calculation of eigen-decomposition for large matrices can be too expensive. Computationally, the formulation is identical to ISOMAP dimensionality reduction. The only difference is conceptual, as a different distance measure is used in $\bar{d}$ and in the neighborhood calculation. Thus, we can use techniques, in particular Landmark ISOMAP, originally developed to speed up the ISOMAP method. Instead of building a distance matrix $\bar{d}$ for all pairs of vertices, we calculate the distance of all points to a random subset of *landmark* points. Assuming the size of this subset is still much larger than the expected dimensionality of the low-dimensional embedding, the approximation is expected to be of reasonable quality. First, the matrix $\bar{d}$ is double centered and then eigen-decomposition is solved only for the square matrix of landmark points. After calculating embedding on landmark points only, the coordinates for non-landmark points will be $z_i = -\frac{1}{2}(b_i - s)Z^+$, where b_i is the row of the centered matrix $\mathbf{B}$, s is the mean vector of the landmark sub-matrix and Z^+ is the pseudo-inverse transpose of Z. Note that the landmark approximation would not be possible if the the shortlest path (for example as in [20]) was not used as the distance measure between arbitrary pair of points, making the problem practically infeasible for even average-size point clouds with thousands of points.

2.4. Recovering components of a skinned rigged model

As we show in this section, all components of a skinned rigged mesh data representation can be recovered from principal components of the deformation distance.

Limbs As already stated in our motivation for the definition of deformation distance, if there exists a path between a pair of points that transforms rigidly, the deformation between these points is 0. This suggests that by performing clustering (such as K-means) on the coordinates of the low dimensional embedding (a.k.a. spectral clustering), we get a set of clusters close to rigid. Thus, the clusters can be used as an initial estimate of limbs. In our implementation, the clustering procedure is initialized using combinations of extreme values of each coordinate in the low-dimensional embedding disregarding clusters with too few points. Each cluster m (later associated with a limb) will eventually consist of an initial set of point tracks $\mathbf{x}_m^k$ for each frame k.

Static template and transformation matrices Using the initial clusters (limbs), we estimate the static mesh template $\mathbf{x}$ and transformation matrices T_m^k incrementally by initializing $T_m^1 = I$ and $\mathbf{x}_m^{(1)} = \mathbf{x}_m^1$, where $(.)$ denotes an iteration, aligning points of the cluster in each frame to a template and adjusting the template using aligned points. This step is equivalent to operations described in [15] only differs in that instead of using Gaussian mixture, a skinned rigged model is applied in our case. the The alignment is calculated by minimizing:

$$T_m^k = arg\min_{T'} \sum_{k\in\{1..K\}} \sum_{i\in\mathbf{x}_m} ||T'^{-1}x_i^k - x_i^{(k-1)}||_0, \quad (12)$$

where $|.|_0$ is a L_0 pseudo-norm maximizing the number of inliers using RANSAC. The template is updated as a running mean to

$$x_m^{(k)} = \frac{k-1}{k}x^{(k-1)} + \frac{1}{k}(T_m^k)^{-1}x_m^{(k-1)}. \quad (13)$$

Several iterations over all frames can be performed to improve the quality. In practice the procedure converges after only a few iterations.

Weights associations with limbs The weights w_{im} are estimated by minimizing the reprojection error in all frames as a non-negative least square problem:

$$w_i = arg\min_{w_i'} \sum_{k\in\{1..K\}} (\sum_{m\in\{1..M\}} w'_{im}T_m^k x_i - x_i^k)^2, \quad (14)$$

such that $\forall m : w_{im} \geq 0$ and $\sum_{m\in\{1..M\}} w_{im} = 1$. In practice it is enough to use top three limbs either according to the distance in the low-dimensional embedding or based on their pure reprojection error with $w_{im} = 1$. Using these weight vectors, we can reestimate limb cluster to consist of points with the highest weight associated with it and repeat the whole procedure of estimating a static template, transformation matrices and weights.

2.5. 3D Data completion

In many practical applications, where the source of the data are standard stereo or depth cameras, only a fraction of points is observed in each frame. We show that under a much weaker assumption, that for each limb at least a small fraction of its points are visible in most frames, we are able to recover and complete unobserved data.

To generalize our method to this much more challenging real-world scenario, we have to modify only the definition of basic local topological properties - the notion of neighborhood and total local deformation between neighboring points. Let δ_i^k be an indicator variable denoting whether i-th point is visible in $k-th$ frame. We will denote that $x_j \in N(x_i)$ is in the neighborhood of x_i for incomplete

noise	mean ± SD	median
0.000	0.02415 ± 0.01200	0.02086
0.002	0.02922 ± 0.02232	0.01991
0.010	0.03359 ± 0.01982	0.02708
0.020	0.03318 ± 0.01409	0.02991
0.030	0.04461 ± 0.02285	0.03867
0.040	0.04503 ± 0.01882	0.04169

Table 1. Quantitative evaluation of robustness to gaussian noise. The error is calculated for models scaled to height 1. The noise is in the same units as the reconstruction error. The error of our method is calculated with respect to ground truth with no noise. As seen from the table, the error of the reconstruction converges to the level of noise.

point tracks, if it is in its neighborhood in a nonempty set of all frames where both points are jointly visible:

$$\forall k : |x_i^k - x_j^k| \delta_i^k \delta_j^k < \epsilon. \tag{15}$$

Next we modify total deformation for neighboring points as an average over deformations in frames where both points are visible:

$$\bar{d}_{ij} = \frac{1}{\sum_{(k,l)\in P} \delta_i^k \delta_j^k} \sum_{(k,l)\in P} d^{kl}(i,j) \delta_i^k \delta_j^k. \tag{16}$$

The total deformation for non-neighbouring points will be defined the same way as before without any modification as the shortest path (7) through the generated graph. After performing spectral clustering over modified deformation function, the transformation matrices (12) for each limb in each frame and weights (14) for each vertex are simply calculated only from points visible in the corresponding frames. The location of unobserved points in each frame can be recovered using (1) with estimated transformation matrices and their location in the static mesh.

3. Experiments

We performed experiments on Mixamo and DFAUST dataset [7]. Mixamo dataset is a large free collection of artificial characters and a practically infinite number of parametric animations from the same-named 3d graphics technology company. We tested our algorithm on several characters and animations downloaded from www.mixamo.com and wolf and spider model from 3dhaupt.com. As an input of our algorithm, we generated an animated point cloud by randomly sampling 2000 points on the surface of each mesh. We run our method on two problems - compression and completion. For the first problem, we assumed all points are constantly visible. For the second problem, a camera performed 1080 degree rotation around the vertical axis of the object with only a subset of points facing the camera visible in each frame. The qualitative results of the compression are depicted in Figure 2. As a typical human body has ˜12-20 independent limbs that at this coarse scale, in practice 5 eigenvectors can represent all dominant degrees of freedom. The landmark approximation has been done using 200 landmark points. As seen in the figure, the estimation of 5 eigenvectors is rather robust and often look identical that only differ in order. The first two typically split a body into the left and right part, and the top and bottom part, and the following ones recover individual rigid parts. The next step, spectral clustering, is more random and can change dramatically for the same character and different animation. The final step - iterative optimization of deformation matrices and weight associations smoothes out the initial solution, leading to much more visually pleasant and more robust results. The method generally works well for any complete input with data generated using the same process in the experiments.

The data completion results are shown in Figure 4. Similarly to full point tracks, the estimation of eigenvectors was still robust, because in most cases even a small number of diverse frames is enough to identify deforming parts. The main problem in this step was the estimation of deformation for neighbouring points in highly concave regions which typically led to a higher number of estimated independently moving limbs. Yet the model was still typically able to recover locations of all points, if all limbs were at least partly visible. Quantitative comparisons of reconstruction error with [20] for complete and partial points can be found in Table 3. To get a fair comparison, we fixed the number of clusters to 15 (20 for Spider). The errors for partial tracks are surprisingly only 2x higher than for complete tracks setting. The average runtime for a single animation for complete tracks was around 10 minutes and for partial tracks around 30 minutes since RANSAC procedure for the latter has to be processed on per-frame basis. We performed an ablation study on the robustness of our method to noise by adding Gaussian noise of varying σ to the input and compared reconstruction error with noise-free ground truth. The reconstruction errors are shown in Table 2 which can be observed to converge to the level of noise.

Dynamic FAUST (DFAUST) [7] dataset is a collection of 4D scans with multiple human subjects in motion, recorded in a controlled environment at 60 fps. Similarly to artificial data, we run our method for compression and completion. We generated partial tracks using the same procedure as for artificial data. The qualitative results of compression are depicted in Figure 3. The estimated eigenvectors seemed equally robust to the artificial data, with the exception of performance on fat people that more clusters are needed with independent models for belly and back. The qualitative result for data completion can be found in Figure 5. Similarly to artificial data, typically a larger number of clusters

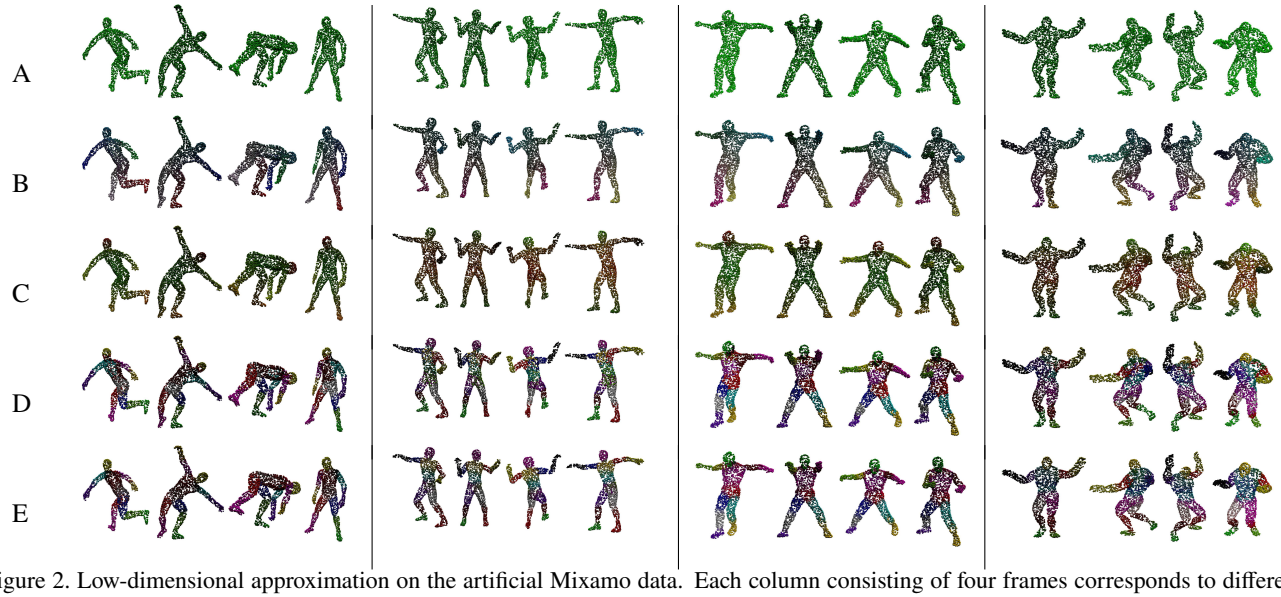

Figure 2. Low-dimensional approximation on the artificial Mixamo data. Each column consisting of four frames corresponds to different animation. A Input point tracks (also ground truth). B First three eigenvector displayed as RGB values normalized to [0-255]. C Fourth and fifth eigenvector, displayed as RG values normalized to [0-255]. D Initial spectral clustering, performed on five eigenvectors. E Reconstruction using low-dimensional model. Point colors calculated by blending cluster colors using weights w.

Figure 3. Low-dimensional approximation on DFAUST dataset [7]. Each column consisting of four random frames corresponds to different animation. A Input point tracks (also ground truth). B First three eigenvector displayed as RGB values normalized to [0-255]. C Fourth and fifth eigenvector, displayed as RG values normalized to [0-255]. D Initial spectral clustering, performed on five eigenvectors. E Reconstructions using low-dimensional model. Point colors calculated by blending cluster colors using weights w.

were needed to model incomplete point tracks. Despite that, our method was robustly capable to reconstruct unobserved data with less than 2x higher error than for complete tracks. Quantitative comparisons with [20] for a fixed number of 15 clusters can be found in Table 4.

The compression ratio for full track points is around 9% for 50 frames and 4% for 300 frames videos and converges to a typically small number proportional to cluster number and reciprocal to the sample number.

4. Conclusions and further work

In this paper, we showed how a generative parametric deformable model can be recovered for a collection of 3D point tracks. Furthermore, we showed how the method can be generalized to perform 3D data completion from partly observed data. As all inputs of the method are local, in

Figure 4. Point track completion on the artificial Mixamo data. Each column consisting of four random frames corresponds to different animation. A Input point tracks. B First three eigenvector displayed as RGB values normalized to [0-255]. C Fourth and fifth eigenvector, displayed as RG values normalized to [0-255]. D Initial spectral clustering, performed on five eigenvectors. E Completed reconstructions using low-dimensional model. Point colors calculated by blending cluster colors using weights w. F Complete ground truth point tracks.

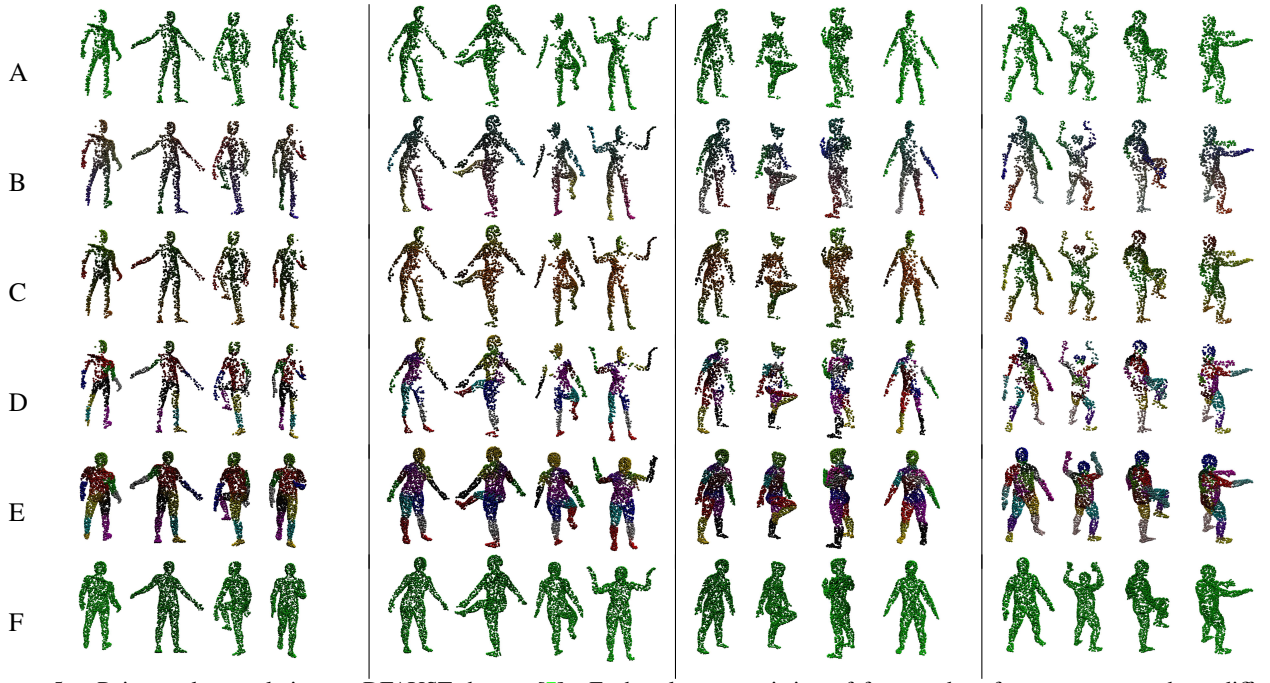

Figure 5. Point track completion on DFAUST dataset [7]. Each column consisting of four random frames corresponds to different animation. A Input point tracks. B First three eigenvector displayed as RGB values normalized to [0-255]. C Fourth and fifth eigenvector, displayed as RG values normalized to [0-255]. D Initial spectral clustering, performed on five eigenvectors. E Completed reconstructions using low-dimensional model. Point colors calculated by blending cluster colors using weights w. F Complete ground truth point tracks.

character	animation	Complete point tracks								Partial point tracks			
		[20] single shot		[20] iterated		Ours single shot		Ours iterated		Ours single shot		Ours iterated	
		mean	median	mean	median	mean	median	mean	median	mean	median	mean	median
Exo	AerialEvade	0.29247	0.15097	0.04441	0.03828	0.03467	0.03422	**0.02089**	**0.01821**	0.0761	0.04966	0.05166	0.04065
Exo	BreakdanceEnd	0.02711	0.02492	0.01921	0.01584	0.0203	0.01873	**0.01244**	**0.00963**	0.02864	0.01935	0.02481	0.01881
Exo	SambaDance	0.02574	0.02437	0.01643	0.01406	0.01912	0.01653	**0.01428**	**0.01027**	0.04055	0.02364	0.03467	0.02307
Exo	StandingDeath	0.04059	0.02981	0.03385	0.02089	0.02384	0.0226	**0.01731**	**0.01345**	0.11457	0.02815	0.03864	0.02705
Exo	WallClimb	0.02546	0.02321	0.024	0.02171	0.02857	0.02818	**0.01792**	**0.01541**	0.03245	0.02958	0.037	0.02875
Prisoner	RoundhouseKick	0.02	0.01865	0.01337	0.00965	0.01681	0.01398	**0.00945**	**0.00563**	0.04896	0.01876	0.01761	0.01363
Prisoner	SitupToIdle	0.03539	0.02981	0.01297	0.01046	0.02431	0.02046	**0.00965**	**0.00785**	0.04876	0.02181	0.02924	0.02034
Prisoner	Swinging	0.03583	0.03173	0.02203	0.02123	0.04052	0.03947	**0.01841**	**0.01754**	0.04419	0.04112	0.03545	0.03307
PumpkinHulk	CastingSpell	0.02522	0.01941	**0.01608**	0.01442	0.02469	0.01958	0.01882	**0.01392**	0.03068	0.02865	0.02249	0.01862
PumpkinHulk	HitOnLegs	0.03534	0.02688	**0.02376**	**0.02014**	0.03207	0.02842	0.02457	0.02041	0.03806	0.02932	0.03218	0.02652
PumpkinHulk	SideKick	0.02571	0.02163	0.01844	0.01327	0.02265	0.01942	**0.01372**	**0.01212**	0.03059	0.02157	0.01845	0.01628
SkeletonZombie	Capoeira	0.02039	0.01966	0.01134	0.00935	0.01329	0.01225	**0.00965**	**0.00857**	0.0171	0.01363	0.0156	0.0124
SkeletonZombie	HipHopDancing	0.03827	0.03275	0.0271	0.02365	0.03669	0.02996	**0.02188**	**0.01727**	0.03612	0.03104	0.03312	0.02917
SkeletonZombie	HipHopShake	0.02808	0.02525	0.01521	0.0143	0.01778	0.017	**0.01074**	**0.00954**	0.0177	0.01557	0.01628	0.01445
Yaku	SalsaDancing	0.03479	0.02175	0.0161	0.01359	0.01666	0.01446	**0.00786**	**0.00617**	0.01969	0.01649	0.01863	0.01608
Yaku	TwoHandSpellCasting	0.02673	0.01704	0.01454	0.01329	0.02191	0.01727	**0.01434**	**0.01305**	0.02526	0.02018	0.0231	0.01899
Parasite	Crawlback	0.03152	0.02976	0.01914	0.01759	0.03213	0.02997	**0.01722**	**0.01435**	0.03906	0.02844	0.03625	0.02712
Parasite	HipHopShake	0.02811	0.02802	0.01502	0.01371	0.01968	0.01816	**0.0121**	**0.01102**	0.0198	0.01767	0.0191	0.01583
Parasite	Jumping	0.02897	0.01954	**0.01513**	**0.0144**	0.02578	0.0235	0.01653	0.01459	0.03506	0.02496	0.03061	0.02298
Wolf	-	0.03477	0.02974	0.02324	0.02179	0.02932	0.02406	**0.02228**	**0.01905**	0.07246	0.04678	0.06801	0.05034
Spider	-	0.06216	0.05748	0.02927	0.02577	0.02288	0.02004	**0.02133**	**0.01847**	0.02675	0.02215	0.02536	0.02091

Table 2. Quantitative comparison on artificial Mixamo dataset and two additional wolf and spider models from 3dhaupt.com with [20]. The error is calculated for models scaled to height 1. The original implementation of [20] using only direct one shot output from eigensolver and we show also comparison with extended iterated version that iteratively re-estimates the limbs and association weights. For comparison we also report completion results on partial tracks. The reconstruction error was only approximately 2x higher than for complete tracks. The median error is typically smaller than mean error because of outliers such as moving fingers.

animation	Complete point tracks								Partial point tracks			
	[20] single shot		[20] iterated		Ours single shot		Ours iterated		Ours single shot		Ours iterated	
	mean	median	mean	median	mean	median	mean	median	mean	median	mean	median
50002	0.03066	0.02398	**0.02072**	0.01913	0.02805	0.02508	0.02093	**0.01815**	0.04261	0.03268	0.03588	0.02524
50004	0.03832	0.03426	0.02417	0.02371	0.03118	0.03028	**0.02285**	**0.02066**	0.06637	0.03586	0.04049	0.03209
50007	0.03296	0.02548	0.01948	0.01812	0.03174	0.02623	**0.01911**	**0.01761**	0.05591	0.03371	0.03291	0.02581
50009	0.03974	0.03154	0.03057	0.02494	0.02837	0.02684	**0.02326**	**0.02169**	0.03761	0.03031	0.03514	0.02647
50020	0.04266	0.03549	0.02938	0.02577	0.0295	0.02817	**0.02188**	**0.01982**	0.0589	0.04087	0.0418	0.03252
50021	0.04295	0.03756	0.03411	0.02951	0.036231	0.03138	**0.02586**	**0.02327**	0.0505	0.03928	0.04867	0.03561
50022	0.05381	0.04015	0.03334	0.02711	0.03018	0.02976	**0.0218**	**0.0215**	0.07112	0.03843	0.04857	0.03628
50025	0.04493	0.04013	0.02901	0.02655	0.03355	0.02974	**0.02832**	**0.02333**	0.04847	0.03825	0.03857	0.03113
50026	0.04113	0.03162	0.02729	0.02397	0.03522	0.0305	**0.02221**	**0.02032**	0.04458	0.03426	0.03425	0.02639
50027	0.05594	0.043	0.02493	0.02126	0.03512	0.03122	**0.02274**	**0.02031**	0.04346	0.03585	0.03473	0.02838

Table 3. Quantitative comparison on DFAUST dataset [7]. The error is calculated for models scaled to height 1. The original implementation of [20] using only direct one shot output from eigensolver and we show also comparison with extended iterated version that iteratively re-estimates the limbs and association weights. The algorithm performed only slightly worse than for artificial data, mainly because the real data is not generated using the estimated data structure. For comparison we also report completion results on partial tracks. The reconstruction error was surprisingly less than 2x higher than for complete tracks.

the future we plan to investigate data completion from 2D point tracks. Furthermore, the local nature makes the 2D convolutions suitable, and thus deep learning applicable to train the deformation directly from the image data.

References

[1] P. M. Q. Aguiar, J. M. F. Xavier, and M. Stosic. Spectrally optimal factorization of incomplete matrices. *Conference on Computer Vision and Pattern Recognition*, 2008. 1

[2] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Nonrigid structure from motion in trajectory space. *Advances in Neural Information Processing Systems*, 2009. 1

[3] J. Alvn, F. Kahl, M. Landgren, V. Larsson, J. Uln, and O. Enqvist. Shape-aware label fusion for multi-atlas frameworks. *Pattern Recognition Letters*, 2018. 1

[4] T. Aumentado-Armstrong, S. Tsogkas, A. Jepson, and S. Dickinson. Geometric disentanglement for generative latent shape models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[5] L. Ballan and G. Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. 2008. 2

[6] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. *Conference on Computer Vision and Pattern Recognition*, 2014. 1

[7] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. Dynamic faust: Registering human bodies in motion. *Conference on Computer Vision and Pattern Recognition*, 2017. 1, 5, 6, 7, 9

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001. 2

[9] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. *Conference on Computer Vision and Pattern Recognition*, 2000. 1

[10] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. *Conference on Computer Vision and Pattern Recognition*, 2005. 1

[11] M. Chupin, E. Gerardin, R. Cuingnet, C. Boutet, L. Lemieux, S. Lehricy, H. Benali, L. Garnero, and O. Colliot. Fully automatic hippocampus segmentation and classification in alzheimer's disease and mild cognitive impairment applied on data from adni. *Hippocampus*, 2009. 1

[12] J. a. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 1998. 1

[13] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 2014. 1

[14] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *Conference on Computer Vision and Pattern Recognition*, 2010. 1

[15] J. Franco and E. Boyer. Learning temporally consistent rigidities. In *CVPR 2011*, pages 1241–1248, 2011. 2, 4

[16] R. Garg, A. Roussos, and L. Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. *Conference on Computer Vision and Pattern Recognition*, 2013. 1

[17] O. Halimi, I. Imanuel, O. Litany, G. Trappolini, E. Rodolà, L. Guibas, and R. Kimmel. The whole is greater than the sum of its nonrigid parts, 2020. 2

[18] N. Ichimura. Motion segmentation based on factorization method and discriminant criterion. *International Conference on Computer Vision*, 1999. 1

[19] Q. Ke and T. Kanade. Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. *Conference on Computer Vision and Pattern Recognition*, 2005. 1

[20] A. G. Kirk, J. F. O'Brien, and D. A. Forsyth. Skeletal parameter estimation from optical motion capture data. *Conference on Computer Vision and Pattern Recognition*, 2005. 1, 2, 4, 5, 8, 9

[21] V. Larsson and C. Olsson. Compact matrix factorization with dependent subspaces. *Conference on Computer Vision and Pattern Recognition*, 2017. 2

[22] V. Larsson, C. Olsson, E. Bylow, and F. Kahl. Rank minimization with structured data patterns. *European Conference on Computer Vision*, 2014. 1

[23] S. Liu, W. Cai, S. Liu, F. Zhang, M. Fulham, D. Feng, S. Pujol, and R. Kikinis. Multimodal neuroimaging computing: the workflows, methods, and platforms. *Brain Informatics*, 2015. 1

[24] R. Newcombe, D. Fox, and S. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. *Conference on Computer Vision and Pattern Recognition*, 2015. 2

[25] R. Plankers and P. Fua. Articulated soft objects for multiview shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1182–1187, 2003. 2

[26] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics*, 2015. 1

[27] C. Russell, J. Fayad, and L. Agapito. Energy based multiple model fitting for non-rigid structure from motion. *Conference on Computer Vision and Pattern Recognition*, 2011. 1

[28] C. Russell, R. Yu, and L. Agapito. Video pop-up: Monocular 3d reconstruction of dynamic scenes. *European Conference on Computer Vision*, 2014. 1

[29] K. Shoemake. Animating rotation with quaternion curves. *ACM Transactions on Graphics*, 1985. 2

[30] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 1992. 1

[31] J. Yan and M. Pollefeys. Articulated motion segmentation using ransac with priors. *Proceedings of the International Conference on Dynamical Vision*, 2007. 1

[32] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *Transactions on Pattern Analysis and Machine Intelligence*, 2008. 1

[33] K. Zhou, B. L. Bhatnagar, and G. Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In *The European Conference on Computer Vision (ECCV)*, August 2020. 2

[34] Y. Zhu, D. Huang, F. D. L. Torre, and S. Lucey. Complex non-rigid motion 3d reconstruction by union of subspaces. *Conference on Computer Vision and Pattern Recognition*, 2014. 1