Fast and Effective Dynamic Mesh Completion

Gerasimos Arvanitis Aris S. Lalos Konstantinos Moustakas Nikos Fakotakis
Dept. of Electrical and Computer Engineering
University of Patras, Rio, Patras, Greece
{arvanitis, aris.lalos, moustakas}@ece.upatras.gr, fakotaki@upatras.gr

ABSTRACT

We introduce a novel approach to support fast and efficient completion of arbitrary animation sequences, ideally suited for real-time scenarios, such as immersive tele-presence systems and gaming. In most of these applications, the reconstruction of 3D animations is based on dynamic meshes which are highly incomplete, stressing the need of completion approaches with low computational requirements. In this paper, we present a new online approach for fast and effective completion of 3D animated models that estimates the position of the unknown vertices of the current frame by exploiting the connectivity information and the current motion vectors of the known vertices. Extensive evaluation studies carried out using a collection of different incomplete animated models, verify that the proposed technique achieves plausible reconstruction output despite the constraints posed by arbitrarily complex and motion scenarios.

Keywords
3D animated meshes, missing vertices, weighed iterative function

1 INTRODUCTION

Recently, there has been increasing interest in real-time 3D capture enabling the acquisition of dynamically deforming shapes at sustained ”video” rates. Although resolution and accuracy of 3D scanners are constantly improving, they are still unable to capture the full surface at once. Even in scenarios where multiple sensors are placed around the subject, most scanned shapes are likely to exhibit large holes, noise and outliers due to occlusions [BTSAL14], limited sensor range capabilities, high light absorption and low surface albedo.

Although a large number of prior works [SGP03] has investigated the problem of completion in static geometries, resulting in excellent filled static meshes, their direct application to every frame separately usually causes incorrect topologies and temporally incoherent surfaces. A fast and efficient approach for reconstructing surfaces from a set of known points have been proposed in [SC04] where the authors reconstruct meshes with a prescribed connectivity that approximate a set of control points in a least-squares sense.

Building on this direction, in this work we introduce a novel technique for reconstruction of highly incomplete dynamic meshes[1], by exploiting a known connectivity and set of motion vectors corresponding to the known points in an online setting. In this setting the animation sequence in not known a priori and at each time our method exploits information that has been presented so far. An extensive analysis has demonstrated that the high-frequency details of the animated model can be adequately recovered from a highly incomplete geometry dataset at very fast execution times.

2 RELATED WORKS & CONTRIBUTIONS

In recent years, a lot of research has been carried out into the field of 3D mesh reconstruction, having presented excellent results applied to incomplete static meshes. However, little attention has been given to the reconstruction of animated meshes. Traditional methods usually cause temporally incoherent surfaces when they are directly applied to each frame individually. These methods do not take advantage of previous frames knowledge, as a result they basically deal with n individual meshes instead of a sequence of temporally coherent meshes. A common approach to produce a temporal consistent dynamic mesh is to use a template prior [ZA04], however, this approach is not ideal for real-time applications because the entire captured animated mesh is required before the execution of the process.

The authors in [ACSTD07] focus on reconstructing watertight surfaces from unoriented point sets using a

[1] A dynamic mesh is a common term defined as a series of static, mainly triangular, meshes representing a 3D animation.

Voronoi-based variational approach, while the method in [SLS07] tries to handle the missing points by trying to infer topological structures in the original surface at the potential expense of retaining geometric fidelity. The researchers in [DGQ12] perform reconstruction only on the available information, effectively preserving the boundaries from the scan. Recently, a new signal processing technique known as matrix completion (MC) [CAN12] has been successfully applied to several computer vision problems, including the recovery of occluded faces/dynamic meshes [DDZ11], [VLMB07], [SWG08] and the face image alignment [PGWXM12]. It has been also used for the fusion of point clouds from multiview images of the same object [DDZW12]. In [RPMR13], it is applied on RGB-D data for the simultaneous tracking and reconstruction of 3D objects.

The common limitation of all the aforementioned approaches is the high computational complexity that significantly affects the execution time and renders them inappropriate for real time applications. These limitations motivated us to search for a fast and effective approach that can satisfy the reconstruction efficiency supporting at the same time real time applications. In summary, the main contributions of our work are:

- A general out-of-core approach to dynamic mesh completion ideally suited for fast and accurate filling (in spatial or temporal space) of incomplete arbitrary mesh sequences.
- An extensive experimental evaluation under different configurations and mesh animations showing that our approach achieves the highest reconstruction quality offering at the same time faster execution times as compared to previous methods.

The rest of this paper is organized as follows: Section 2 includes a detailed summary of prior art. Section 3 presents an overview of our method. Section 4 presents our experimental results and discusses the advantages and limitations of the proposal method. Section 5 draws conclusions and identifies future directions.

3 OVERVIEW OF OUR METHOD

Initial Assumptions and Preliminaries

In this section we present the basic assumptions and preliminaries related to animated meshes. Firstly, a dynamic mesh is defined $A = [M_1;\ M_2;\ \dots\ M_n]$ as a sequence of n static meshes consisting of k vertices. Each one of these meshes can be represented by two different sets $M = (\mathscr{V}, \mathscr{F})$ corresponding to the vertices $(\mathscr{V})$ and the indexed faces $(\mathscr{F})$ of the mesh. Each vertex can be represented as a point in the Euclidean space. Let us define with $\mathbf{v} = [\ \mathbf{x},\ \mathbf{y},\ \mathbf{z}\ ]$ a vector of vertices in a 3D coordinate space denoted as $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \Re^{k\times 1}$, $\mathbf{v} = [v_1,\ v_2,\ \dots\ v_k] \in \Re^{k\times 3}$ and $V = \{v_1,\ v_2,\ \dots\ v_k\}$ is the corresponding set of vertices. Additionally, each face is represented as a set of 3 connected vertices $f_i = [v_{i1},\ v_{i2},\ v_{i3}]\ \forall\ i = 1, m$ where $m > k$ and the corresponding set of faces is denoted by $F = \{f_1,\ f_2,\ \dots\ f_m\}$. The set of edges $\mathscr{E}$ can be directly derived from $\mathscr{V}$ and $\mathscr{F}$, corresponds to the connectivity information.

Let us assume that A' is a highly incomplete dynamic mesh. In other words, each mesh of the animation has only a subset of known vertices while the rest have been removed. The incomplete animated model is represented by a sequence of incomplete meshes $A' = [M_1;\ M_2';\ \dots\ M_n']$ where $M_i' \subset M_i\ \forall\ i = 2, n$. Each incomplete dynamic mesh is described by a matrix of dimension $3k \times n$:

$$A' = \begin{bmatrix} M_1 \\ M_2' \\ M_3' \\ M_4' \\ \vdots \\ M_n' \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & \dots & v_{1k-1} & v_{1k} \\ 0 & v_{22} & 0 & 0 & \dots & 0 & 0 \\ v_{31} & 0 & 0 & v_{34} & \dots & 0 & 0 \\ 0 & 0 & v_{43} & 0 & \dots & v_{4k-1} & v_{4k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & v_{n2} & 0 & 0 & \dots & v_{nk-1} & 0 \end{bmatrix}$$

The incomplete meshes are created by randomly removing points from the original ones. Fig. 1 depicts some indicative frames (meshes) assuming different densities of known points.

Adjacency and Laplacian Matrix

To estimate the coordinates of the missing points, we initially use a prescribed connectivity information (i.e., adjacency matrix), constructed from the faces of the mesh. Despite the fact that the position of vertices is changing, the adjacency matrix remains fixed over time, since we assume that every mesh has the same connectivity [WJHB07]. This observation allows us to estimate the adjacency matrix only once and use it repeatedly for any subsequent mesh of the same model. Moreover, we assume that we have full knowledge of the first mesh M_1 of the sequence. We define as $\mathbf{R} \in \Re_{k\times k}$ the adjacency matrix which is estimated as described below:

$$\mathbf{R}_{ij} = \begin{cases} 1 & if\ i, j \in E \\ 0 & otherwise \end{cases} \tag{1}$$

The matrix $\mathbf{R}$ is binary and it is used for the creation of the Laplacian matrix defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{R} \tag{2}$$

$\mathbf{D} = diag\{d_1, \dots, d_k\}$ is a diagonal matrix with $d_i = \sum_{j=1}^{k} r_{ij}$, being the degree of its node.

Spatial Classification of Each Frame Vertices

We create k cells of nodes $c_i\ \forall\ i = 1, k$ using the knowledge of the adjacency matrix $\mathbf{R}$. Each cell c_i rep-

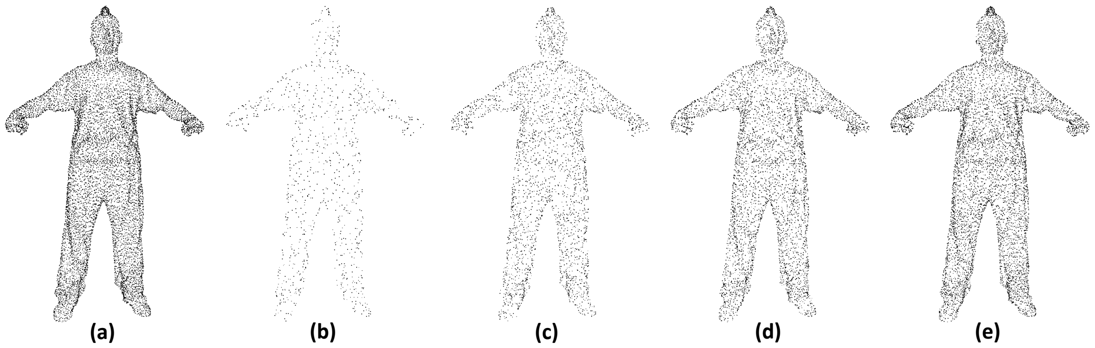

Figure 1: Indicative incomplete frames of the animated sequence: (a) original mesh (10002 points), (b) 10% of original points, (c) 30% of original points, (d) 50% of original points, (e) 70% of original points

resents the first ring area of each vertex i. We define as $\mathscr{C} = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_k]$ the set of k cells where $\mathbf{c}_i = [\dot{c}_{i1}\ \dot{c}_{i2}\ \ldots\ \dot{c}_{ij}]\ \forall\ i = 1,k$. However, it is worth mentioning here that each cell has different connectivity valence denoted by j. The element $\dot{c}_{ip}$ represents the index of p-th connected vertex with the vertex i. The set $\mathbf{C}$ remains fixed for every frame (mesh) and it is used for recovering the missing points. Fig. 2 illustrates an example of two connected cells $\mathbf{c}_1, \mathbf{c}_2$ with their related connections, cell c_1 has $j = 6$ connected neighbors (valence), while c_2 has a valence $j = 5$.

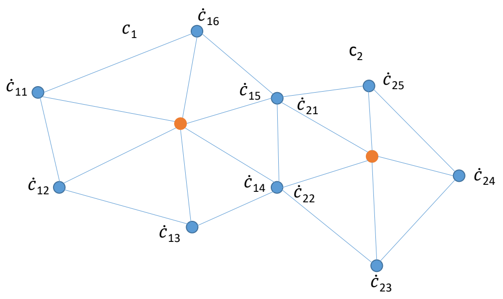

Figure 2: Representation of two connected cells

After the definition of cells we focus on the classification of vertices for each frame $M_i^{\prime}\ \forall\ i = 2,n$. This process is executed in a sequential manner for each mesh starting from the second mesh until the end of the animation sequence. We assume three different classes for each vertex:

- **Anchor vertices** are the known vertices of the mesh.
- **Satellites vertices** are those belonging to a cell of an already known vertex. If a vertex belong to a cell of an anchor vertex then is defined as first generation satellite otherwise is defined as second generation satellite and so on.
- **Unknown vertices** are the vertices that do not belong to a cell of an already known vertex. Those vertices will be recovered and placed in the mesh at a future iteration.

The classification procedure is an iterative process that tries to eliminate any remaining unknown vertex so that the mesh will be composed of only anchor or satellite vertices. This means that the coordinates of an unknown vertex are evaluated in a following iteration. This method has been proved to be robust and all the missing vertices are always recovered. The method starts with the assumption that if the position of a vertex is known then it is called anchor point, $i\ \forall\ i = 1,k^{\prime}\ k^{\prime} < k$. All the vertices that belong to the cell $\mathbf{c}_i$ are classified as satellite vertices (first generation satellite). The rest vertices are classified as unknown. At the following iteration the position of the satellite points are taken into account and their cells are used for identifying new satellites (second generation satellites).

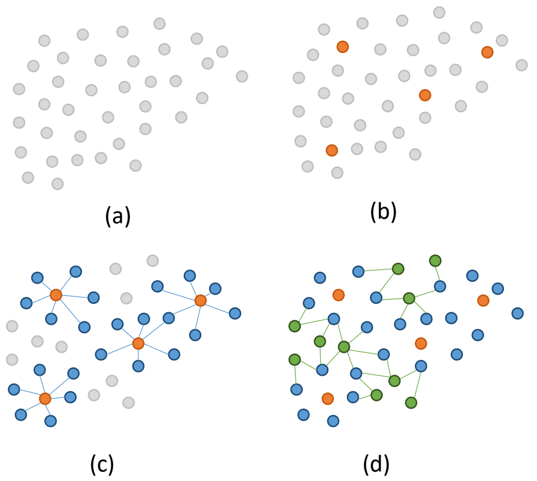

Figure 3: (a) Part of mesh with unknown points, (b) Anchors are indicated with red (Iteration 1), (c) First generation satellites identified based on anchors cells (Iteration 2), (d) Second generation satellites are set based on first generation satellites cells (Iteration 3).

Fig. 3 illustrates the aforementioned classification procedure. Red points represent the known vertices (anchor points). Blue points represent the satellites (first generation) which are connected with the anchor points, and correspondingly green point represent the satellites

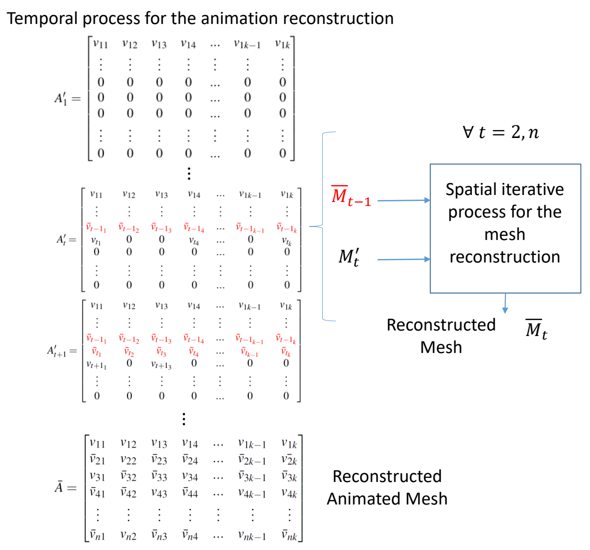
Temporal process for the animation reconstruction
$A'_1 =$
$A'_t =$
$A'_{t+1} =$
$\bar{A} =$
$\forall\, t = 2, n$
$\overline{M}_{t-1}$
M'_t
Spatial iterative process for the mesh reconstruction
Reconstructed Mesh
$\overline{M}_t$
Reconstructed Animated Mesh

Figure 4: The animation is reconstructed frame by frame (temporal process) taking into account the previous reconstructed mesh and the current incomplete (spatial iterative process).

(second generation) that are connected with the first generation satellites. It is important to mention here, that a satellite point can be also a satellite point of more than one anchor point while, an anchor point can be a satellite point for another neighbor anchor point.

A cell c_i can indicate the existence of satellite vertices in a first ring area of vertex i, nevertheless it does not specify their real position. The classification procedure is used for assigning weights and then estimating the unknown vertex position using a weighted filtering approach based on their previous position and the motion vector of the anchor points. A detailed description of the method is provided in the following section.

Geometry Completion Based on Topological Characteristics

In this section we present the main steps of our interpolation procedure which is divided in two stages. In the first stage, a spatial iterative process is executed for reconstructing the mesh using only the known vertices of the current mesh. In the second stage, a temporal process tries to reconstruct the entire animated mesh using knowledge of the previous frame. Fig. 4 presents the process while the reconstruction takes place gradually, starting from the second mesh and continues until the end of the animation sequence rendering the method appropriate for online setups.

As mentioned earlier, a cell **c** represents the first ring area where points (satellite vertices) are connected with an anchor vertex. Therefore we decided to build upon the assumption that the satellite vertices are expected to move towards the direction of an anchor keeping their

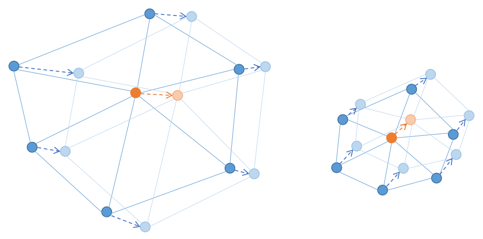

Figure 5: Example of large and small variations in cell movements.

common topological characteristics (e.g., distances between each other) unchanged. However, some satellites are connected with more than one anchors, meaning that their new position will be affected by the motion vectors of every connected anchor.

For making the estimation of coordinates more accurate we suggest a weighted reconstruction function which is defined by exploiting the following observations. As it was mentioned earlier, when a satellite is connected with more than one anchors then its new position is affected by the motion vectors of all the anchors. However, each anchor contributes with a different weight that is related to the relative distance between anchors and satellites. Smaller cells are more rigid so that their satellite points are expected to follow the motion vectors of the closest anchor, as shown in Fig. 5.

The second rule that we apply, is based on the fact that some points are more trustworthy than others. In other words we give more emphasis on the anchor points instead of the satellites due to their known position. Subsequently, we give more emphasis in the first generation satellites rather than the next generation because they are connected directly with the anchor points so that their estimated position is expected to be more accurate. According to the aforementioned observations, we distinguish two different weighted factors:

(a) the weighting factor s_{ij} that represents the inverse distance between points j (new discovered satellite) and i (already known point) such as $s_{ij} = 1/||\mathbf{v}_i - \mathbf{v}_j||_2^2$.

(b) the weighting factor w_i that represents the prioritization weight of vertex i. More specifically, anchors prioritization weights have the highest value while the last generation satellites have the smallest one.

For each vertex $v_j \forall\ j = 1, k - k'$ we define a weighted matrix $\mathbf{S}_j = [s_{1j}\ s_{2j}\ \cdots\ s_{nj}]$ that consists of n' elements that represent the distance between the current vertex and the known vertices that are connected with the vertex v_j. The matrix $\mathbf{W} = [w_1\ w_2\ \cdots\ w_{k'}]$ is universal and can be used by every vertex v_j. However, in each iteration the values of their elements increase by one, while in every new element we assign a unit weight. For each satellite j we estimate its new coordinates in the $(p+1)$

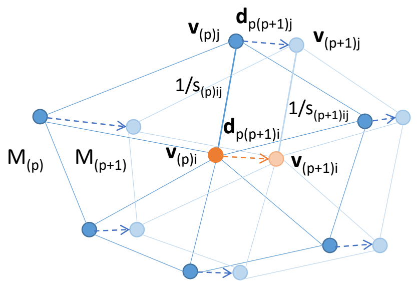

Figure 6: Anchor and satellites movement in a cell of two sequential meshes.

mesh, by updating its previous coordinates in (p) mesh, based on the following equation:

$$v_{(p+1)j} = v_{(p)j} + \mathbf{d}_{p(p+1)j} \tag{3}$$

where

$$\mathbf{d}_{p(p+1)j} = \frac{\sum_{i=1}^{n} s_{ij} w_i \mathbf{d}_{p(p+1)i}}{\sum_{i=1}^{n} |s_{ij} w_i|} \tag{4}$$

The $\mathbf{d}_{p(p+1)j}$ represents the motion vector of vertex v_i from (p)-th mesh to $(p+1)$-th mesh (see Fig. 6), i represents the known vertex v_i (anchors and satellites) and $\mathbf{d} = [d_x = |x_{(p+1)} - x_{(p)}| \;\; d_y = |y_{(p+1)} - y_{(p)}| \;\; d_z = |z_{(p+1)} - z_{(p)}|]$ is a distance vector. An overview of the proposed method is briefly presented in the following Algorithm 1.

Algorithm 1: Reconstruction of 3D animated model

```
Function: 3D mesh reconstruction based on previous
          complete mesh
Input   : Animated 3D model $A'$ with missing data.
Output  : A reconstructed animated model $\bar{A}$.
1 Find the connectivity $\mathbf{R}$ of $M_1$;
2 for $i \leq n$ do
3     while Number of known vertices $< k$ do
4         Search for satellate points using the connectivity of
          $\mathbf{C}$;
5         Estimate the wighted distance via Eq. (4);
6         Update vertex based on its previous frame
            coordiantes via Eq. (3) ;
7     end
8 end
9 return Reconstructed animated model $\bar{A}$;
```

4 RESULTS

In this section, we present an experimental analysis of the proposed completion approach on different dynamic meshes. The evaluation of both the execution time and the reconstruction quality shows the effectiveness of our method even in complex motion scenario that include rapid changes between sequential frames or in cased with a small percentage of known points.

Experimental Setup

In all the experiments we have used a PC Intel core i7-4710HQ CPU @ 2.50GHz 2.50GHz, 8 GB RAM. The algorithms have been implemented using the Julia scientific language.

Metrics

The quality of the reconstructed results are evaluated using the metrics that are briefly presented below:

NMSVE. Normalized mean square visual error is used in order to evaluate the reconstruction quality of results, by capturing the average distortion between the original and the approximated frame [CG04]:

$$NMSVE = \frac{1}{2k} \sum_{j=1}^{k} \left(\left\| v_i - \tilde{v}_i \right\|_2 + \left\| GL(v_i) - GL(\tilde{v}_i) \right\|_2 \right) \tag{5}$$

$$GL(v_i) = v_i - \frac{\sum_{j \in \mathbf{N}_i} d_{ij}^{-1} v_j}{\sum_{j \in \mathbf{N}_i} d_{ij}^{-1}} \tag{6}$$

d_{ij} denotes the Euclidean distance between i and j.

Heatmap. To efficiently highlight the visual difference between reconstructed and original mesh we use heatmap visualization of $|M_i - \bar{M}_i| \; \forall \; i = 1, n$.

Dataset

Two types of 3D animated models were used in our experiments. These models represent different case studies because of their inherent properties and the fact that they target on different applications (e.g., immersive tele-presence systems, gaming). Specifically, (a) Handstand has many smooth areas, while there are no abrupt temporal changes (175 frames, 10002 vertices and 20000 triangles). (b) Ocean on the other hand is full of repetitive abrupt changes (1500 frames 2500 vertices and 4802 triangles).

Comparison Methods

For comparison purposes, we have also employed conventional techniques for the reconstruction of the animation models, namely the least-square meshes (LSM) algorithm [SC04] and the Laplacian interpolation approach (LIA) [OOH89]. LSM is described as the solution of an extended system of equations $[L^T I_{n \times k'}^T]^T x_p = [0^T, x_{k',k}]$, for $p = 1, \cdots, n$, for k' known (anchor) vertices in the p-th frame. Laplacian interpolation is described as a fast and effective method with a lot of similarities with LSM. According to [OOH89] a way to interpolate a triangulated mesh is by putting constraints on the Laplacian Δf of the function and trying to minimize its Euclidean norm.

Experimental Results

The processing time is related to the number of initial known vertices. Specifically, the execution time increases linearly with the number of unknown vertices. In Fig. 7 we present the processing time for the reconstruction of each mesh of the animated sequence using different initialization schemes. The required number of iterations for a complete mesh reconstruction depends on the percentage of known vertices. Additionally, we can observe that the two models have a similar behavior.

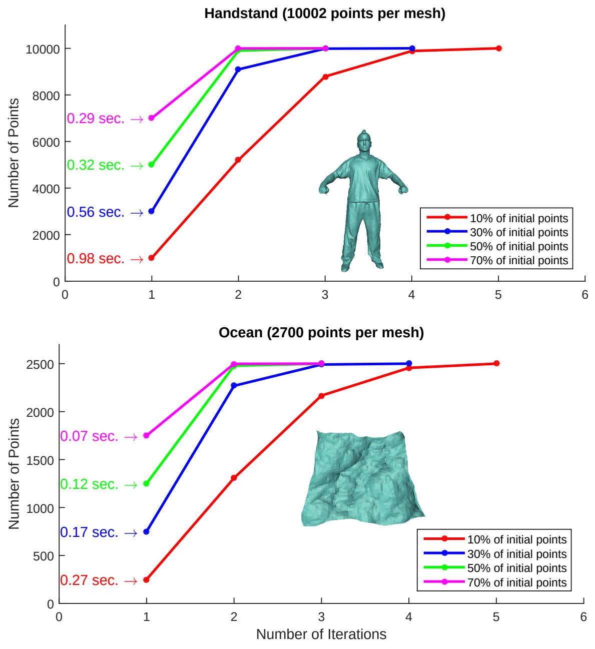

Figure 7: Number of iteration and processing time for a full mesh reconstruction.

In Fig. 8 the missing vertices are visualized with red color and the known vertices with blue (1st and 3rd row). A heatmap visualization is also offered presenting the squared difference between original and reconstructed mesh for different numbers of known vertices (2nd and 4th row). The compared results of our approach and LIA are presented in Fig. 9 showing that our method outperforms LIA in both reconstruction quality and execution time. A major disadvantage of LIA is the smoothed results even in cases with a high percentage of remaining points. In terms of execution time, our method becomes faster when more remaining points are used, because of the less iterations that are required, contrary to LIA where the execution time increases because of the larger matrix operations. Fig. 10 illustrates some indicative reconstructed frames of the Handstand model after using the aforementioned approaches. Our method seems to outperform the others while LIA and LSM have similar performance. Fig. 11 presents some indicative reconstructed frames of different animated

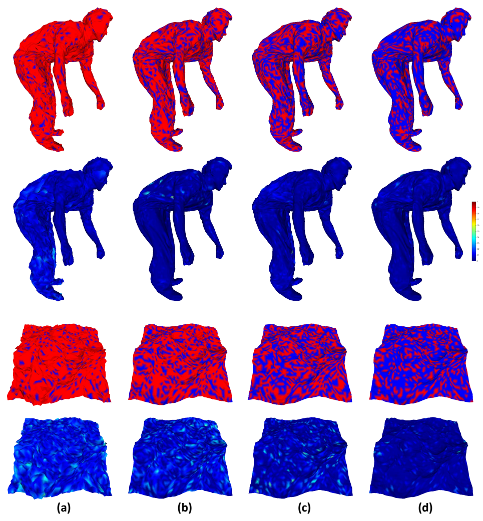

Figure 8: Visualized missing data and heatmap visualization for different density of points (a) 10% of original points, (b) 30% of original points, (c) 50% of original points, (d) 70% of original points. (Handstand frame 110, Ocean frame 1500).

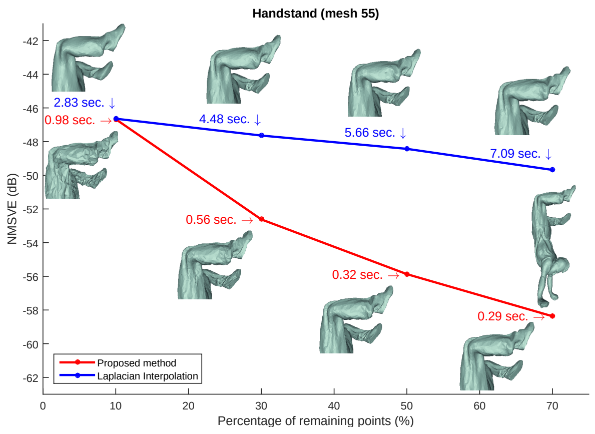

Figure 9: NMSVE and processing time results for the two compared methods.

models. For the sake of completeness, the NMSVE values are also illustrated under each reconstructed mesh. Finally, it should be noted that despite the high motion variance of animated trajectories, the perceptual quality of the reconstructed dynamic meshes when the density of the known point is higher than $> 30\%$ is considerably high. This totally satisfies the main goal of this work which is the design and implementation of fast and effective dynamic mesh reconstruction approaches.

5 CONCLUSIONS AND FUTURES EXTENDS

In this work we introduced a fast and effective method for reconstructing animated 3D models with missing data. The proposed method takes advantage of the adjacency matrix information in order to identify the coordinated of the missing vertices. After that a weighted

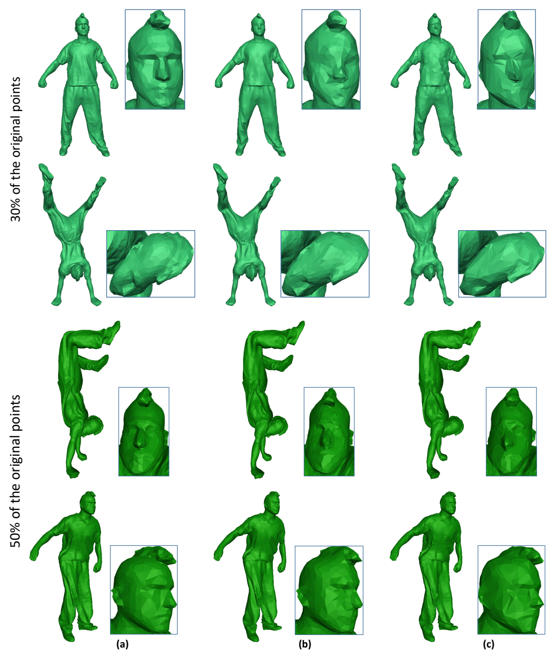

Figure 10: Handstand with 30% of original points (Frames 150 & 80) and with 50% of the original points (Frames 55 & 120) (a) our method, (b) LIA, (c) LSM.

iterative procedure estimates the position of missing vertices based on their previous position and the motion vectors of the connected anchors. An extensive evaluation study using a collection of different 3D animation models verified that the proposed technique achieve plausible reconstruction output and fast execution times.

6 REFERENCES

[BTSAL14] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, et al.. State of the Art in Surface Reconstruction from Point Clouds. Eurographics 2014 - State of the Art Reports, Apr 2014, Strasbourg, France. 1 (1), pp.161-185, 2014, EUROGRAPHICS.

[SGP03] P. Liepa. 2003. Filling holes in meshes. In Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 200-205

[OOH89] Thom F Oostendorp, Adriaan van Oosterom and Geertjan Huiskamp, Interpolation on a triangulated 3D surface. Journal of Computational Physics, 80: 331-343, 1989.

[CG04] Z. Karni and C. Gotsman, Compression of soft-body animation sequences, Computers Graphics, vol. 28, pp. 25-34, 2004.

[WJHB07] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling, Reconstruction of Deforming Geometry from Time-Varying Point Clouds, EUROGRAPHICS, 2007.

[DDZ11] Y. Deng, Q. Dai, and Z. Zhang, Graph laplace for occluded face completion and recognition, IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2329-2338, Aug 2011.

[PGWXM12] Y. Peng, A. Ganesh, J. Wright,W. Xu, and Y. Ma, Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2233-2246, Nov 2012.

[CAN12] Candes, Emmanuel, and Benjamin Recht. "Exact matrix completion via convex optimization." Communications of the ACM 55.6 (2012): 111-119.

[DDZW12] Y. Deng, Y. Liu, Q. Dai, Z. Zhang, and Y.Wang, Noisy depth maps fusion for multiview stereo via matrix completion, IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 5, pp. 566-582, Sept 2012.

[SWG08] J. Süßmuth, M. Winter, G. Greiner, Reconstructing animated meshes from time-varying point clouds, Eurographics Association, vol. 27, no. 5, pp. 1469-1476, 2008.

[RPMR13] C. Ren, V. Prisacariu, D. Murray and I. Reid, STAR3D: Simultaneous Tracking And Reconstruction of 3D Objects Using RGB-D Data, IEEE International Conference on Computer Vision, 2013.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, M. Desbrun, Voronoi-based variational reconstruction of unoriented point sets. In Computer Graphics Forum (Proc. of the Symposium on Geometry Processing), 2007.

[SLS07] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo and D. Cohen-Or, Interactive topology-aware surface reconstruction. ACM Trans. Graph. (Proc. SIGGRAPH), 2007.

[DGQ12] T. K. Dey, X. Ge, Q. Que, I. Safa, L. Wang, Y. Wang, Feature-preserving reconstruction of singular surfaces, In Computer Graphics Forum, 2012.

[SC04] Least-Squares Meshes. In Proceedings of the Shape Modeling International 2004 (SMI '04). IEEE Computer Society, Washington, DC, USA, 191-199.

[VLMB07] E. Vlachos, A. Lalos, K. Moustakas, K. Berberidis, Efficient graph-based matrix completion on incomplete animated models, IEEE International Conference on Multimedia and EXPO (ICME) 2017, At HONG KONG.

[ZA04] Li Zhang et. al., 'Spacetime faces: High resolution capture for modeling and animation,' ACM Trans. Graph., vol. 23, no. 3, pp. 548-558, Aug. 2004.

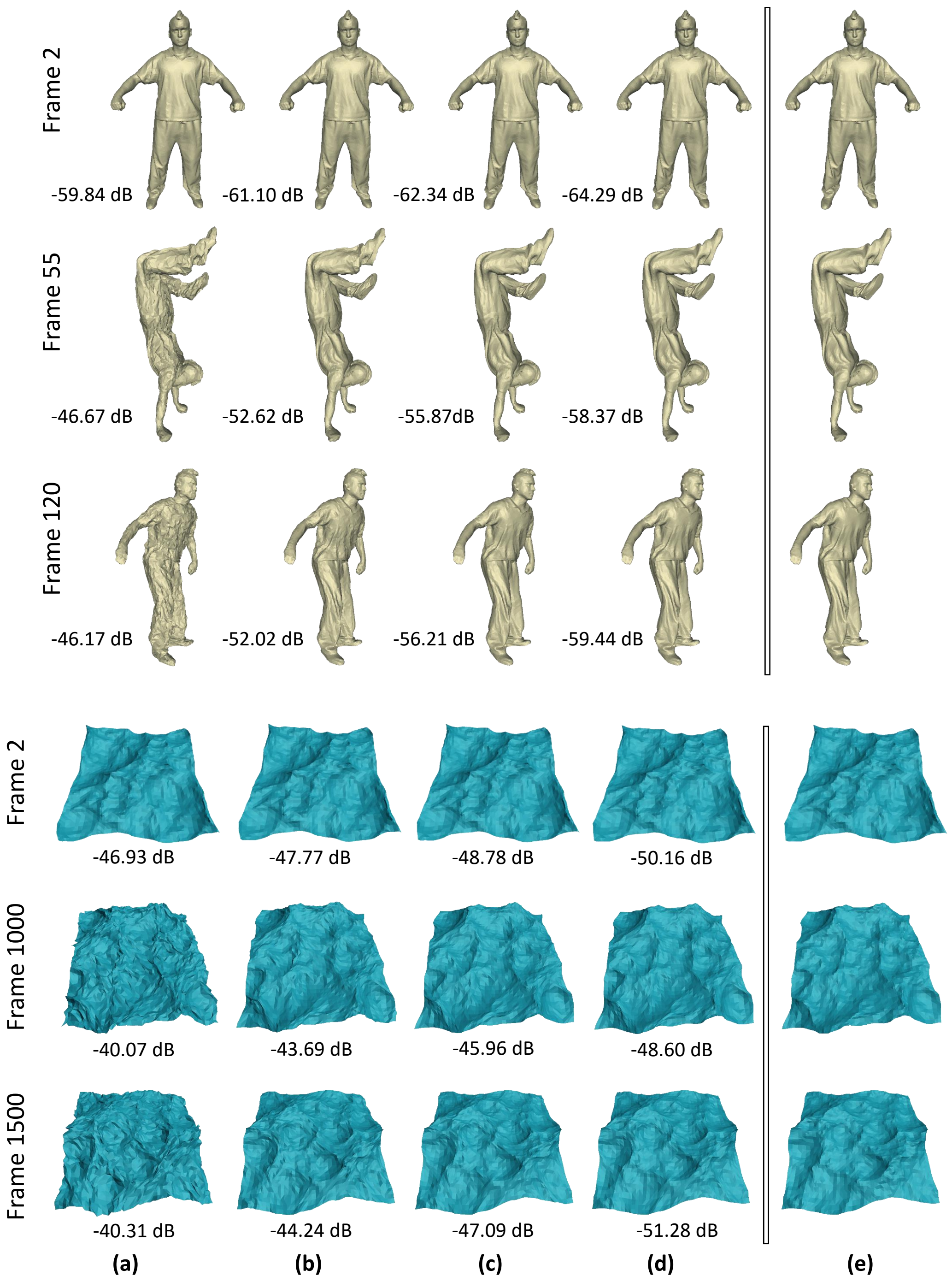

Figure 11: Reconstructed meshes for different density of remaining points (a) 10% of original points, (b) 30% of original points, (c) 50% of original points, (d) 70% of original points, (e) original mesh. (Handstand & Ocean)