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], [VLMBO07], [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 [DDZ2W12]. 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 proposed 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 \( A = [M_1, M_2, \ldots, M_n] \) as a sequence of \( n \) static meshes consisting of \( k \) vertices. Each one of these meshes can be represented by two different sets \( M = (\mathcal{V}, \mathcal{F}) \) corresponding to the vertices \( \mathcal{V} \) and the indexed faces \( \mathcal{F} \) of the mesh. Each vertex can be represented as a point in the Euclidean space. Let us define with \( v = [x, y, z] \) a vector of vertices in a 3D coordinate space denoted as \( x, y, z \in \mathbb{R}^{k \times 1} \), \( v = [v_1, v_2, \ldots, v_k] \in \mathbb{R}^{k \times 3} \) and \( \mathcal{V} = \{v_1, v_2, \ldots, v_k\} \) 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 \( \mathcal{F} = \{f_1, f_2, \ldots, f_m\} \).

The set of edges \( \mathcal{E} \) can be directly derived from \( \mathcal{V} \) and \( \mathcal{F} \), 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 \( A' = [M'_1; M'_2; \ldots; M'_n] \) where \( M'_i \subset M_i \forall i = 2, n \). Each incomplete dynamic mesh is described by a matrix of dimension \( 3k \times n \):

\[
A' = \begin{bmatrix}
M'_1 & V_{11} & V_{12} & V_{13} & \ldots & V_{1k-1} & V_{1k} \\
M'_2 & 0 & V_{22} & 0 & 0 & \ldots & 0 & 0 \\
M'_3 & V_{31} & 0 & 0 & 0 & \ldots & 0 & 0 \\
M'_4 & 0 & 0 & V_{43} & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
M'_n & 0 & 0 & 0 & 0 & \ldots & V_{nk-1} & 0
\end{bmatrix}
\]

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 [WJHBO7]. 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 \( R \in \mathbb{R}^{k \times k} \) the adjacency matrix which is estimated as described below:

\[
R_{ij} = \begin{cases} 
1 & \text{if } i, j \in E \\
0 & \text{otherwise}
\end{cases}
\] (1)

The matrix \( R \) is binary and it is used for the creation of the Laplacian matrix defined as:

\[
L = D - R
\] (2)

\( D = \text{diag}\{d_1, \ldots, d_k\} \) 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 \( R \). 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.

Figure 2: Representation of two connected cells.
Temporal process for the animation reconstruction

\[ A^t = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_{n-1} & a_n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & a_n \end{bmatrix} \]

\[ \forall t = 2, n \]

Spatial iterative process for the mesh reconstruction

\[ M^t \]

Reconstructed Mesh

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 \( c_j \) 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 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 \( s_{ij} = 1/||v_i - v_j||^2 \).

(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 \( v_j \) \( j = 1, k - k' \) we define a weighted matrix \( S_j = [s_{1j} \ s_{2j} \ \cdots \ s_{nj}] \) 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 \( W = [w_1 \ w_2 \ \cdots \ w_{k'}] \) 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 \( (p + 1) \)
Estimate the weighted distance via Eq. (4):

\[ \text{d}_{ij} = \sum_{p=1}^{n} s_{ij} w_{ij} \left| d_{ij} - d_{ij}^{(p)} \right| \]

where

\[ d_{ij} = \left| x_{i} - x_{j} \right|, \quad d_{ij}^{(p)} = \left| x_{i}^{(p)} - x_{j}^{(p)} \right|, \quad \text{and } d_{ij}^{(p)} = \left| z_{i}^{(p)} - z_{j}^{(p)} \right| \]

The \( d_{ij} \) represents the Euclidean distance between the \( i \) and \( j \) points.

**Algorithm 1: Reconstruction of 3D animated model**

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

**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 scenarios that include rapid changes between sequential frames or in cases 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, 8 GB RAM. The algorithms have been implemented using the Julia scientific language.

**Metrics**

The quality of the reconstructed results is 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]:

\[ NMSVE = \frac{1}{2k} \sum_{t=1}^{k} \left( \left\| v_{t} - v_{t}^{\prime} \right\|_{2} + \left\| \bar{z}_{t} - z_{t}^{\prime} \right\|_{2} \right) \]

**Heatmap.** To efficiently highlight the visual difference between reconstructed and original mesh we use heatmap visualization of \( |M_{t} - \tilde{M}_{t}| \) \( \forall i = 1, n \).

**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 \( [L^{T} H]_{x} \tilde{x}_{p} = \left[ 0^{T}, x_{p}^{2} \right] \), for \( p = 1, \ldots, n \), 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 \( \Delta f \) 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 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 coordinates of the missing vertices. After that a weighted
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


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)