A Review of Two Approaches for Joining 3D Meshes

Anh-Cang Phan[1(✉)], Romain Raffin[2], and Marc Daniel[2]

[1] VinhLong College of Economics and Finance, Vinh Long , Vietnam
pacang@vcef.edu.vn
[2] Aix-Marseille University, CNRS, LSIS UMR 7296, 13009 Marseille, France

Abstract. The construction of smooth surfaces of 3D complex objects is an important problem in many graphical applications. Unfortunately, cracks or holes may appear on their surfaces caused by the limitation of scanners or the difference in resolution levels and subdivision schemes between adjacent faces. In this paper, we introduce two approaches for joining 3D meshes of different resolutions to remove the cracks or holes. These approaches use a wavelet transform and a RBF local interpolation or a tangent plane local approximation. They guarantee that the discrete continuity between meshes is preserved and the connecting mesh can change gradually in resolution between coarse and fine mesh areas.

Keywords: Triangular meshes · Mesh connection · Smoothness · Local approximation · RBF local interpolation · B-spline wavelets

1 Introduction

3D object models with complex shapes are usually generated by a set of assembled patches or separate meshes which may be at different resolutions, even with different subdivision schemes. A generic problem arising from subdividing two meshes initially connected along a common boundary is the occurrence of cracks or holes if they are separately subdivided by different schemes (i.e. Butterfly [1], Loop [2], etc.). In order to deal with these drawbacks and particularly cracks, we propose two new approaches joining two selected meshes of a 3D model so that the "continuity" between these meshes can be preserved. It means that the curvatures must be "continuous" on the boundaries, which must be studied in terms of discrete curvatures, the latter being not presented here. We aim at constructing a high quality connecting mesh linking two meshes of different resolutions. The connecting mesh is constructed by adding triangle strips to each boundaries up to the time they are close enough to be linked.

2 Previous Work

Some research [3–5] are related to the incremental subdivision method with Butterfly and Loop schemes. The main goal of these methods is to generate a smooth

P.C. Vinh et al. (Eds.): ICTCC 2016, LNICST 168, pp. 82–96, 2016.
DOI: 10.1007/978-3-319-46909-6_9

surface by refining only some selected areas of a mesh and remove cracks by simple triangulation. However, this simple triangulation changes the connectivity, the valence of vertices, and produces high valence vertices leading to long faces. This not only alters the limit subdivision surface, but also creates ripple effects on the subdivision surface and therefore reduces its smoothness. In addition, several research [6,7] relevant to joining meshes along boundary curves are of immediate practical interest. These methods consist in connecting the meshes of a surface at the same resolution level which adopt various criteria to compute the planar shape from a 3D surface patch by minimizing their differences. In general, they are computationally expensive and memory consuming. Besides, the algorithms do not mention the continuity and the progressive change in resolution between meshes after joining. Meanwhile, the main challenge in designing a mesh connection algorithm is to guarantee these features. From the above motivation, we propose in this paper two new methods based on mesh connection approaches to overcome these drawbacks and particularly cracks and holes.

3 Background

3.1 Wavelet-Based Multiresolution Representation of Curves and Surfaces

Wavelets has been applied successfully in the areas of computer graphics [8,9]. The combination of B-splines and wavelets leads to the idea of B-spline wavelets [10]. Taking advantage of the lifting scheme [12], the Lifted B-spline wavelet [11] is a fast computational tool for multiresolution of a given B-spline curve with a computational complexity linear in the number of control points. The Lifted B-spline wavelet transform includes the forward and backward B-spline wavelet transforms. From a fine curve at the resolution level J, C^J, the forward wavelet transform decomposes C^J into a coarser approximation of the curve, C^{J-1}, and detail (error) vectors. The detail vectors are a set of wavelet cœfficients containing the geometric differences with respect to the finer levels. The backward transform synthesizes C^{J-1} and the detail vectors into a finer curve, C^J. In our approach, we apply the Lifted B-spline wavelet transform for multiresolution of discrete boundary curves of a connecting mesh.

3.2 Radial Basis Function (RBF) Local Interpolation

In order to extrapolate local frames (tangents, curvatures) between two meshes, we need a local interpolation method on the points that will be projected. In this section, we choose the RBF local interpolation [13,14] to construct an expected surface in crack removal and hole filling from subsets of nearest neighboring points because it provides local details of the interpolated surface and exploits the characteristics of flexibility and accuracy. The basic idea of the RBF local interpolation is to find a local interpolation function which implicitly defines a surface (denoted CM) using a set of local control points. The signed distance

function $f(x)$ is represented as the signed distance from x to the closest point on CM. If point x lies on CM, $f(x)$ vanishes ($f(x) = 0$). Point x is called "on-surface point". In contrast, it is called "off-surface point" and $f(x)$ is not zero. Given a set of N data points $X = \{x_k = (a_k^x, a_k^y, a_k^z)\}_{k=1}^N \subset \mathbb{R}^3$. For each $x_k \in X, k = 1, ..., N$, we determine:

- A set of local control points: $X_k = \{x_k\} \cup \left\{x_i \in \mathbb{R}^3; x_i \in Neighbors(x_k)\right\}$ corresponding to a set of the signed distance function values F_k, where $Neighbors(x_k)$ is the nearest neighboring points of x_k. For each point of X_k, we compute one off-surface point to specify a set of the local control points in X_k so that the number of points in X_k is multiplied by 2.
- Distance function values: $F_k = \{f_i = f(x_i), i \in I_k\} \subset \mathbb{R}$, where I_k is the set of indexes of X_k and $N_k = |X_k|$ is the number of the local control points in X_k. The signed distance function f is defined by equation:

$$\begin{cases} f(x_i) = 0, \text{ if } x_i \text{ are on-surface points} \\ f(x_i) = d \in \mathbb{R}, \text{ if } x_i \text{ are off-surface points} \end{cases} \tag{1}$$

  with d is a parameter predefined by the user.

A RBF local interpolation function s_k: $\mathbb{R}^3 \to \mathbb{R}$ on X_k is expressed:

$$s_k(x) = \sum_{c \in I_k} \lambda_c \phi(\|x - x_c\|) \tag{2}$$

where $\phi(\|x - x_c\|)$ are the radial basis functions (RBFs); x_c are the control points and are also the nearest neighboring points of x_k; λ_c are the RBF weights; $\|x\|$ is the Euclidean norm. The user needs to find $s_k(x)$ such that it satisfies the constraints:

$$s_k(x_i) = f_i = \sum_{c \in I_k} \lambda_c \phi(\|x_i - x_c\|), i \in I_k \tag{3}$$

The basis function is normally chosen from the family of spline functions. Typically, the Gaussian function $\phi(r) = e^{-\left(\frac{r}{h}\right)^2}$ is suggested in our method because we want that the RBF local interpolation function $s_k(x)$ provides a local approximation of data points x. The user should choose h as the average distance between x and control points x_c [15]. Combining (2) and (3) leads to the linear system expressed in a matrix form:

$$\begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,N_k} \\ \vdots & \ddots & \vdots \\ \phi_{N_k,1} & \cdots & \phi_{N_k,N_k} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_{N_k} \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_{N_k} \end{pmatrix} \tag{4}$$

Equation (4) may be re-written in simplified matrix form:

$$\Phi_{X_k} \Lambda_{X_k} = F_{X_k} \tag{5}$$

where $\phi_{i,c} = \phi(||x_i - x_c||)$, $\Phi_{X_k} = (\phi_{i,c})$ with $i, c \in I_k$, $\Lambda_{X_k} = (\lambda_1, \lambda_2, ..., \lambda_{N_k})^T$, $F_{X_k} = (f_1, ..., f_{N_k})^T$. After solving the linear system (5) to compute Λ, a set

of data points on CM is simply reconstructed by computing $s_k(x)$ at $x \in X_k$ using (2). The RBF local interpolation give good results for surface reconstruction but it is not adequate for data points with abrupt and large changes within small distances. In addition, it requires much more estimations of the shape parameter (h), off-surface constraints. Therefore, we need to find other feasible and reliable methods such as an implicit surface fitting with tangent planes [17] that produces high quality surfaces in our work.

3.3 Tangent Plane Local Approximation for Implicit Surface Reconstruction

We describe here a tangent plane local approximation proposed by Hoppe et al. [16] for implicit surface reconstruction from 3D point cloud. Given a set of data points $P = \{p_i\} \in \mathbb{R}^3$ of a surface CM, to determine data points p_{new} on CM, the authors estimate a set of local tangent planes $Tp(p_i)$ represented as local linear approximations of CM, and then find the projection p_{new} of an arbitrary point $p \in \mathbb{R}^3$ onto CM. The estimation of $Tp(p_i)$ and the projection of p onto CM are described as follows:

Estimation of a tangent plane: Let $Tp(p_i)$ be the tangent plane corresponding to point p_i and passing through a centroid point o_i. An arbitrary point p is projected onto tangent plane $Tp(p_i)$ which has point o_i closest to point p. Tangent plane $Tp(p_i)$ is determined by passing through point o_i with unit surface normal n_i as follows:

- Find local neighbors of each data point:
  For each point $p_i \in \mathbb{R}^3$, the user finds a set of nearest neighbors of p_i denoted $Neighbors(p_i)$.
- Compute a centroid point on a tangent plane: For each point $p_i \in \mathbb{R}^3$, the user computes the centroid point o_i based on all nearest neighbors of p_i:

$$o_i = \frac{\sum_{p_j \in Neighbors(p_i)} p_j}{N} \tag{6}$$

  where N is the number of the neighbors of p_i.
- Estimate a normal vector of a tangent plane: The principal component analysis (PCA) method is used to estimate normal n_i of $Tp(p_i)$. The point covariance matrix $CV_i \in \mathbb{R}^{3\times3}$ from the neighbors of p_i is first computed:

$$CV_i = \sum_{p_j \in Neighbors(p_i)} (p_j - o_i) \otimes (p_j - o_i) \tag{7}$$

  where $\otimes$ denotes the outer product vector operator: if x and y have components x_i and y_j respectively, the matrix $x \otimes y$ has $x_i y_j$ as its ij-th element.
  Eigenvalues $\lambda_{i,1} \geq \lambda_{i,2} \geq \lambda_{i,3}$ of CV_i are then determined corresponding to unit eigenvectors $v_{i,1}, v_{i,2}, v_{i,3}$. Since normal n_i is the eigenvector corresponding to the smallest eigenvalue, the user chooses to be either $v_{i,3}$ or $-v_{i,3}$. The choice determines the tangent plane orientation [16].

Projection of p onto Tp(p_i): The projected point p_{new} is the orthogonal projection of point p onto $Tp(p_i)$. Let $f(p)$ be a signed distance from an arbitrary $p \in \mathbb{R}^3$ to CM:

$$f(p) = dist(p, p_{new}) = (p - o_i).n_i \tag{8}$$

Then, the projected point p_{new} is computed by:

$$p_{new} = p - (f(p)\ n_i) \tag{9}$$

4 Overview of Two Methods for Joining Meshes

4.1 Notation

In order to lighten notations, we decide not to use vectorial notations for all the notations or equations having vectorial relations. Moreover, we denote the position vector $\overrightarrow{Op}$ of a vertex p by p, where O is the frame origin. Each multiplication of a scalar value and a vector is understood as the vector components multiplied by the scalar value.

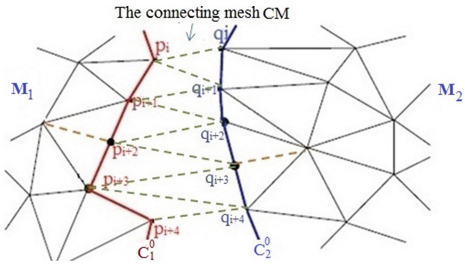

Fig. 1. Topology representation of the algorithm.

Let M_1 and M_2 be two meshes of different resolutions, and p_i, q_k their vertices. An edge connecting p_i to q_k is denoted e_i or p_iq_k. An edge is usually shared by two faces. If it is shared by only one, it corresponds to a boundary edge and its end vertices are called boundary vertices. We need to construct a connecting mesh CM between M_1 and M_2 so that the continuity between them can be preserved as illustrated in Fig. 1. First we will introduce the notations used in the algorithms:

- s: number of newly created boundary curves of CM created between M_1 and M_2. It is a user parameter computed based on the distance between two original boundaries of M_1 and M_2 and it controls the resolution of CM.
- j: order number of the decomposition step to create intermediate discrete curves, also called the level. Since two boundary curves between M_1 and M_2 will be created at each level j, j is in $[1, \frac{s}{2}]$.

- C_1^j and C_2^j: two boundary curves of CM at level j. C_1^0 and C_2^0 are the two original boundary curves of meshes M_1 and M_2.
- $N(C_1^j)$: number of vertices of boundary curve C_1^j at level j. It corresponds to the density of vertices of boundary curve C_1^j.
- p_i^j, q_i^j: vertices i on boundary curves C_1^j and C_2^j. ($p_i^0 = p_i$ and $q_i^0 = q_i$)
- L_1^j: list of the boundary vertex pairs (p_i^{j-1}, q_k^{j-1}); L_2^j: list of the pairs (q_k^{j-1}, p_i^{j-1}).

4.2 Our General Algorithm for Joining Meshes

The idea is to generate the connecting mesh CM consisting of newly created boundary curves using the Lifted B-spline wavelet transform and the local RBF interpolation (for CM2D-RBFW method) or the tangent plane local approximation (for CM2D-TPW method). The general algorithm of our proposed methods consists of the following main steps detailed in the next sections.

A General Algorithm for Joining Meshes
Input: a crack between two meshes of different resolutions and schemes.
Output: a high quality connecting mesh CM and a smooth connection surface.

- **Step 1**. Boundary detection: read the input model of two meshes M_1 and M_2. Detect and mark boundary vertices of the two boundaries C_1^0 and C_2^0 of M_1 and M_2.
- **Step 2**. Boundary vertex pairs and boundary curve creation: for each level j, we pair the boundary vertices of C_1^{j-1} and C_2^{j-1} based on the distance between them. If this distance is too narrow (smaller than a certain threshold), we go to Step 3 to connect the boundary curve pair (C_1^{j-1}, C_2^{j-1}). In contrast, we create two new boundary curves C_1^j, C_2^j from the paired boundary vertices by a linear interpolation and a RBF local interpolation (CM2D-RBFW method) or a tangent plane local approximation (CM2D-TPW method). It finally refines or coarsens these new boundary curves applying wavelet transforms and operations of vertex insertion or deletion.
- **Step 3**. Boundary curve connection: perform a boundary triangulation for each boundary curve pair (C_1^{j-1},C_1^j) and (C_2^{j-1},C_2^j).
- **Step 4**. Repeat steps 2 and 3 until both mesh areas M_1 and M_2 have been connected or patched by all newly created triangles.

5 Boundary Curve Creation and Connection

5.1 Boundary Vertex Pairs

This work is to find all vertices of a boundary curve closest to vertices of a remaining boundary curve to pair. In order to create boundary curves between two meshes M_1 and M_2 by interpolating previously created boundary curves, we pair the boundary vertices $p_i^{j-1} \in C_1^{j-1}$ with $q_k^{j-1} \in C_2^{j-1}$ and vice versa based on the distances between them. Since the densities of vertices of both boundary

curves are different, we need to create two lists of the closest boundary vertex pairs L_1^j and L_2^j. Assume that j is the current level, for each boundary vertex $p_i^{j-1} \in C_1^{j-1}$, we search for and insert into L_1^j the corresponding paired vertex $q_k^{j-1} \in C_2^{j-1}$ such that:
$\left(\forall q \in C_2^{j-1}, dist(p_i^{j-1}, q_k^{j-1}) \leq dist(p_i^{j-1}, q)\right)$, where the notation $dist(p_i^{j-1}, q_k^{j-1}) = ||p_i^{j-1} - q_k^{j-1}||$ is the Euclidean distance between p_i^{j-1} and q_k^{j-1}. The list of boundary vertex pairs L_2^j is created similarly.

5.2 Boundary Curve Creation

5.2.1 CM2D-RBFW Algorithm

The idea of our algorithm is to create the discrete boundary curves C_1^j and C_2^j between M_1 and M_2 from the paired vertices of two previously created boundary curves C_1^{j-1} and C_2^{j-1} using the RBF local interpolation and the Lifted B-spline wavelet transform. Then, we connect each new curve C_1^j to C_1^{j-1}, and C_2^j to C_2^{j-1}. Therefore, it is called the algorithm of ***connecting mesh in two directions based on the RBF local interpolation and the Wavelet transform*** (CM2D-RBFW). Details of the algorithm are introduced as follows:

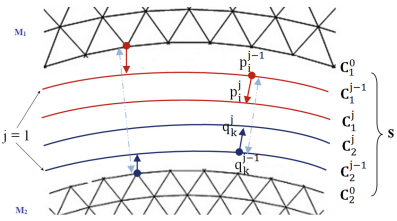

Fig. 2. Creation of new boundary curves.

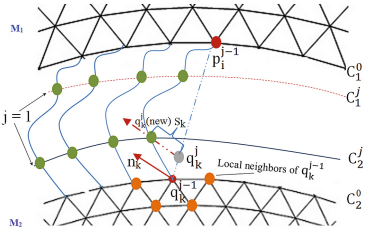

Fig. 3. Projection of the vertices onto CM.

We assume $N(C_1^0) \leq N(C_2^0)$ and let the density of vertices of the two boundary curves C_1^j and C_2^j be two functions $N(C_1^j)$ and $N(C_2^j)$ defined by:

$$\begin{aligned} N(C_1^j) &= N(C_1^0) + \frac{j}{s+1}[N(C_2^0) - N(C_1^0)] \\ N(C_2^j) &= N(C_2^0) - \frac{j}{s+1}[N(C_2^0) - N(C_1^0)] \end{aligned} \tag{10}$$

The boundary curve creation is computed in three phases:

Phase 1: Create vertices of two new boundary curves by a linear interpolation.

– Create new vertices $p_i^j \in C_1^j$ at each level j (see Fig. 2): for each boundary vertex pair $(p_i^{j-1}, q_k^{j-1}) \in L_1^j$, we apply the linear interpolation equation (11)

to create new boundary vertices $p_i^j \in C_1^j$.

$$p_i^j = p_i^{j-1} + \frac{j}{s+1}(q_k^{j-1} - p_i^{j-1}) \tag{11}$$

where i are the subscripts of boundary vertices of C_1^j, $1 \leq i \leq N(C_1^{j-1})$, and k are the subscripts of boundary vertices of C_2^{j-1}, $1 \leq k \leq N(C_2^{j-1})$.

– In the same way, we create new vertices $q_k^j \in C_2^j$ at each level j: for each boundary vertex pair $(q_k^{j-1}, p_i^{j-1}) \in L_2^j$, we apply the linear interpolation equation (12) to create new boundary vertices $q_k^j \in C_2^j$.

$$q_k^j = q_k^{j-1} + \frac{j}{s+1}(p_i^{j-1} - q_k^{j-1}) \tag{12}$$

where k are the subscripts of boundary vertices of C_2^j, $1 \leq k \leq N(C_2^{j-1})$, and i are the subscripts of boundary vertices of C_1^{j-1}, $1 \leq i \leq N(C_1^{j-1})$.

Equations (11) and (12) have been chosen with a local linear expansion classically used in marching methods. Starting from C_1^0 and C_2^0 when $j = 1$, we recursively compute (11) and (12) based on vertices of the curves C_1^{j-1} and C_2^{j-1} but not C_1^0 and C_2^0. In addition, since C_1^{j-1} and C_2^{j-1} are then refined or coarsened by wavelet transforms, their resolutions are increased or reduced respectively.

Phase 2: Project created boundary vertices onto CM using the RBF local interpolation.

The goal of phase 2 is to improve the resulting surface CM after applying phase 1. Since new boundary vertices p_i^j and q_k^j of curves C_1^j and C_2^j are created by a linear interpolation in phase 1, they lie on a line through two vertices p_i^{j-1} and q_k^{j-1} but not on the expected surface CM. As a result, the boundary curves are produced without respect to local curvatures when CM is a complex curved surface. Therefore, the generated connecting mesh CM will not respect the expected continuity between the meshes. This problem is overcome by an implicit surface reconstruction with a RBF local interpolation. We project new vertices $p_i^j \in C_1^j$ and $q_k^j \in C_2^j$ created in phase 1 onto CM by a RBF local interpolation as shown in Fig. 3. Projecting the created vertices $q_k^j \in C_2^j$ onto CM is performed as follows: first, for each vertex q_k^j, we find the closest vertex $q_k^{j-1} \in C_2^{j-1}$ and its neighbors to choose as a set of the local control vertices of q_k^j. Then, we compute the weights of the RBFs (using (4)) and the local interpolation function values s_k for q_k^j (using (2)). Next, we project them onto CM with the projection distances s_k along normals at the vertices q_k^{j-1} (see Fig. 3). The normals at the vertices q_k^{j-1} are estimated via the principal component analysis (PCA) method. Finally, we update the vertices q_k^j as their projections. Similarly, we also perform the same for the created vertices $p_i^j \in C_1^j$.

A problem shows that if we only use on-surface vertices directly to solve for the weights of the radial basis functions, the system Eq. (4) becomes trivial. This

problem can be overcome by creating additional off-surface vertices to construct the RBF local interpolation function. Typically, for each vertex q_k^j, we choose a set of local control vertices Q_k^j corresponding to the signed distance function values $f(q) = 0$ for $q \in Q1_k^j$ and $f(q) = d$ for $q \in Q2_k^j$ to construct the RBF local interpolation function, where $Q_k^j = Q1_k^j \cup Q2_k^j$. $Q1_k^j$ and $Q2_k^j$ are computed by:

- $Q1_k^j$ is referred to as a set of on-surface vertices. It is defined by: $Q1_k^j = \{q_k^{j-1} \in C_2^{j-1}\} \cup \{q_1 \in \mathbb{R}^3; q_1 \in Neighbors(q_k^{j-1})\}$, where $Neighbors(q_k^{j-1})$ is a set of the local neighbors of q_k^{j-1} which have edges connecting to q_k^{j-1}.
- $Q2_k^j$ is referred to as a set of off-surface vertices. It is defined in form: $Q2_k^j = \{q_2 \in \mathbb{R}^3; q_2 = q_1 + d\ n(q_1), \forall q_1 \in Q1_k^j\}$, where d is the estimate of the signed distance to the surface (also called the signed projection distance) defined by the user. However, d can't be too small, otherwise, the matrix of the linear system in Eq. (4) will be ill-conditioned. On the other hand, if d is too large, wrong off-surface points could be created. This can result in an incorrect surface; $n(q_1)$ is the normal vector at the vertex q_1.

When the curves C_1^{j-1} and C_2^{j-1} are close together, the neighbors are chosen from both sides of model to make a set of on-surface vertices. That means, for each vertex q_k^j, we take the two closest vertices $p_i^{j-1} \in C_1^{j-1}$ and $q_k^{j-1} \in C_2^{j-1}$ along with their local neighbors to specify a set of on-surface vertices of q_k^j. This implies that we take into account the local curvatures on both sides to finish CM and have a nice join.

Phase 3: Refine or coarsen new boundary curves with the wavelet transforms. Since the densities of vertices of C_1^j and C_2^j are now $N(C_1^{j-1})$ and $N(C_2^{j-1})$, we need to increase and reduce their densities to be $N(C_1^j)$ and $N(C_2^j)$. Taking advantage of the Lifted B-spline wavelet transform presented in Sect. 3.1, we apply this transform for the multiresolution analysis of the curves C_1^j and C_2^j to refine the curve C_1^j, coarsen the curve C_2^j. Then, we perform operations of the vertex insertion or deletion to control the densities of vertices of C_1^j and C_2^j. Thus, the created curves C_1^j, C_2^j, and the associated connecting mesh CM are changed gradually in resolution between both mesh areas.

CM2D-RBFW method requires too much estimation consisting of the off-surface constraints, the choice of the user parameter values d and h, and the solution of linear systems for an interpolation problem. Thus, in the next section we propose a more reliable method for mesh connection called CM2D-TPW method which allows us to construct a high quality connecting mesh CM and a continuous surface. The goal is to gain in both time of computation and surface quality.

5.2.2 CM2D-TPW Algorithm

Similar to CM2D-RBFW method, we create new curves C_1^j and C_2^j from the paired vertices in each level j based on a tangent plane local approximation and

a wavelet transform as shown in Fig. 4. We also assume $N(C_1^0) \leq N(C_2^0)$ and the density of vertices of the two boundary curves C_1^j and C_2^j are two functions $N(C_1^j)$ and $N(C_2^j)$ defined by Eq. (10). The boundary curve creation is produced in three phases.

Phase 1: Create vertices of two new boundary curves by a linear interpolation.

This phase is similar to phase 1 of CM2D-RBFW method. We recursively compute (11) and (12) based on vertices of the curves C_1^{j-1} and C_2^{j-1} starting from C_1^0 and C_2^0 when $j = 1$.

Phase 2: Project created boundary vertices onto CM using a local approximation.

When CM is a complex curved surface, the newly created vertices p_i^j and q_k^j can not lie on CM because we do not consider the curvature information in phase 1. As a result, the produced connecting mesh will not ensure the expected continuity between two meshes. To solve this problem, we need a tangent plane local approximation [16,17] on which points will be projected as provided in Sect. 3.3 to extrapolate local frames (tangents, curvatures) between two meshes. We first apply phase 1 (linear interpolation) to create new boundary vertices, and then project these vertices onto local tangent planes corresponding to a C^1 or C^2 continuous surface as shown in Fig. 4. As a result, the new vertices are positioned on the expected surface CM with respect to the variation of the tangent planes. It implies that we take into account the local curvatures to have a nice join and a smooth transition with at least C^1 continuity (tangent continuity). Projecting the created vertices $q_k^j \in C_2^j$ onto surface CM is performed as follows: First, for each vertex q_k^j, we find the closest vertex $q_k^{j-1} \in C_2^{j-1}$ and its local neighbors $Neighbors(q_k^{j-1})$ which have edges connected to q_k^{j-1} to determine the local control vertices of q_k^j (see Fig. 4). Next, we estimate the local tangent plane $Tp(q_k^{j-1})$ which is a local linear approximation of surface CM. The plane $\mathrm{Tp}(q_k^{j-1})$ passes through the centroid vertex o_k^{j-1} (using (6)) with unit normal vector n_k^{j-1} (using (7)). From that, we compute $f(q_k^j)$ using (8) whose value is referred to as the signed projection distance between q_k^j and $Tp(q_k^{j-1})$. Then, we use (9) to project them onto CM with the projection distances $f(q_k^j)$ along normals of the local tangent planes represented as surface normals (see Fig. 4).

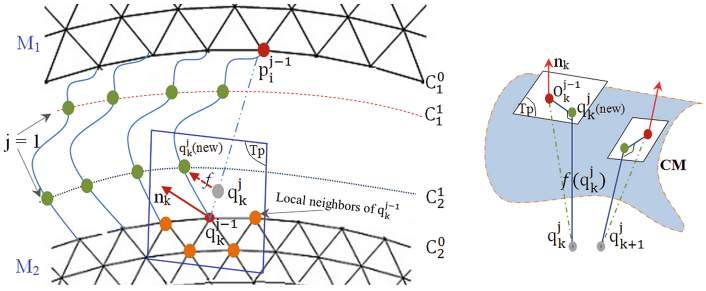

Fig. 4. Projection of the vertices onto CM with a tangent plane local approximation.

Finally, we update vertices q_k^j by their projections. Similarly, we perform the same operation for vertices $p_i^j \in C_1^j$.

When the two curves C_1^{j-1} and C_2^{j-1} are close together, we take the neighboring vertices from both curves to define the set of local neighboring vertices. For each vertex q_k^j, we keep the two closest vertices $p_i^{j-1} \in C_1^{j-1}$ and $q_k^{j-1} \in C_2^{j-1}$ with their neighbors. It permits us to take into account the local curvatures on both sides.

Phase 3: Refine or coarsen the new boundary curves with wavelet transforms. This phase is similar to phase 3 of CM2D-RBFW method.

5.3 Boundary Curve Connection

After creating two boundary curves C_1^j and C_2^j, we connect each new boundary curve to each previously created boundary curve, C_1^{j-1} to C_1^j and C_2^{j-1} to C_2^j, based on the method of stitching the matching borders proposed by G. Barequet et al. [6].

6 Results and Comparisons

We first provide experimental results of CM2D-TPW and CM2D-RBFW methods and then compare these methods with various types of 3D objects. Both methods have been implemented in Matlab on a PC 2.27 GHz CPU Core i5 with 3 GB Ram to make possible their comparisons. To understand the quality of the results, we plot the images of the connecting mesh, surface and Gaussian curvature map.

In Fig. 5, CM2D-TPW method produces a smooth connecting mesh CM with the progressive change in resolution between M_1 and M_2 defined by subdivision schemes (Loop and Butterfly), each mesh being at a different level of subdivision. Based on a set of tests, $s = 4$ is an empirical good value to apply this method for joining M_1 and M_2.

As we seen, Gaussian curvatures in Fig. 6c is respected better than these in Fig. 6b because CM2D-TPW method is possible to constrain the surface to have specified tangent planes at subsets of control vertices to be extrapolated. The newly created vertices are located on the expected surface CM with respect to the variations of tangent planes. This leads to a smooth transition between boundary faces and faces of CM. Therefore, the Gaussian curvatures are well respected and are "continuous" on the boundaries.

In order to draw comparisons, we have chosen examples of a sphere to have accurate evaluations of the error and runtime. We have developed a test on four density-based discretizations of the sphere, since analytical description permits to compute the exact surface and relative errors. The numbers of vertices are 240, 3840, 61440, 983040 and the numbers of vertices of the removed strips are 66, 720, 5982, 70743, respectively. In this way, both meshes M_1 and M_2 have the same density of vertices for each given discretization level, and the process to obtain

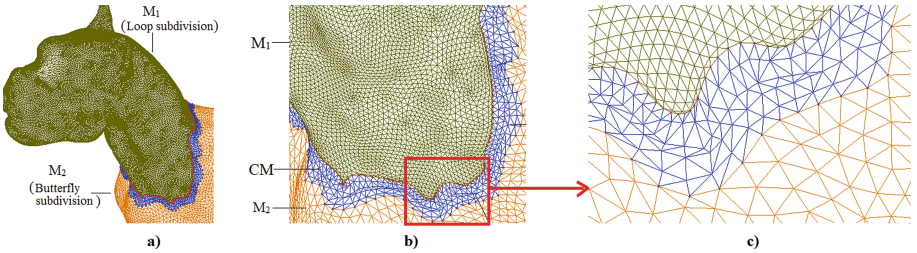

Fig. 5. The Tiger model with CM2D-TPW algorithm: (a) The connecting mesh CM produced with $s = 4$; (b) Zoom of CM; (c) Zoom of one of the interesting parts of CM.

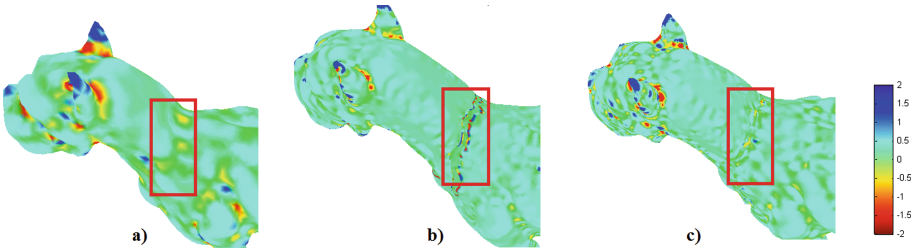

Fig. 6. Gaussian curvature map of the Tiger model: (a) Before crack; (b) After removing crack by CM2D-RBFW algorithm; (c) After removing crack by CM2D-TPW algorithm.

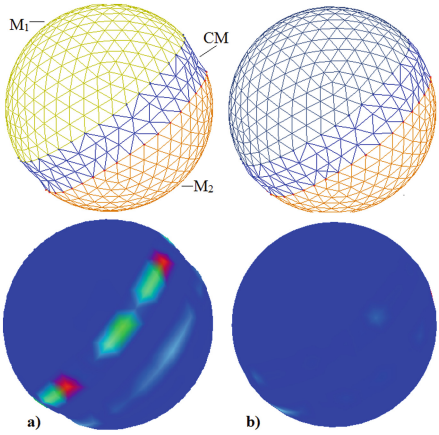

Fig. 7. Mesh connection with model of Sphere 2: (a) CM is produced by CM2D-RBFW with $s = 2$ and $d = 0.004$; (b) CM is produced by CM2D-TPW with $s = 2$.

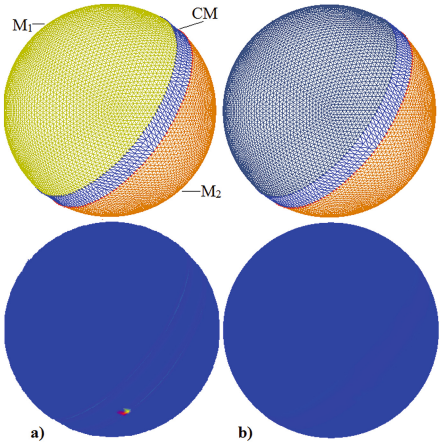

Fig. 8. Mesh connection with model of Sphere 3: (a) CM is produced by CM2D-RBFW with $s = 2$ and $d = 0.004$; (b) CM is produced by CM2D-TPW with $s = 2$.

the compatible number of vertices of CM is the same for both methods. Hence, we define the errors E_{dist} and E_{max} as follows: $E_{dist} = \sqrt{\frac{\sum_{p_i \in CM}(R - dist(c,p_i))^2}{N}}$; $E_{max} = sup(|R - d_i|)$, $1 \leq i \leq N$; where: $d_i = dist(c, p_i)$ is the Euclidean distance between c and vertices p_i of CM; R, c are the radius and center of the sphere, respectively (in our tests, $c = (0, 0, 0)$ and $R = 10$). N is the number of vertices of CM.

Figures 7, 8 and Table 1 summarize the results. First, we use CM2D-RBFW method for the discretization models of the sphere with $s = 2$ as illustrated in Figs. 7a and 8a. Then, we also apply CM2D-TPW method on these models (see Figs. 7b and 8b). Obviously, CM2D-TPW method can position the newly inserted vertices on the expected surface with respect to the variation of tangent planes (phase 2) without destroying the Gaussian curvature and altering the original meshes. As a result, it gives the high quality connecting meshes and smooth surfaces. Figure 8b shows the sphere with a nice join between meshes since Gaussian curvature maps of meshes are virtually the same.

Figures 9a–b show the connecting mesh and surface CM produced with linear interpolation by applying phase 1 and 3 of CM2D-TPW algorithm without phase 2. As a result, CM is hyperbolic and the surface continuity is not guaranteed. While Figs. 9c–d present CM after applying all phases of the algorithm. Obviously, CM2D-TPW method generates a smooth surface with natural shape where continuity between meshes is preserved.

According to these experimental results, we can see that CM2D-TPW method gives better results compared to CM2D-RBFW method since errors to the real surface are smaller (see Table 1) and Gaussian curvatures are much better respected (see Figs. 6, 7 and 8). In addition, a well-known drawback of RBF based reconstruction methods is the difficulty to provide abrupt changes

Table 1. Comparison of errors and runtimes of CM2D-RBFW and CM2D-TPW algorithms for spheres with center $c = (0, 0, 0)$, and radius $R = 10$; the numbers of vertices and faces of CM are in columns V and F.

Model	CM		E_{dist}		E_{max}		Runtime (secs)	
	V	F	RBFW	TPW	RBFW	TPW	RBFW	TPW
Sphere 1	38	40	0.927	0.758	2.366	2.048	0.406	0.316
Sphere 2	159	240	0.202	0.065	0.486	0.222	0.459	0.376
Sphere 3	639	960	0.034	0.015	0.058	0.029	1.386	0.917
Sphere 4	2641	3963	0.063	0.033	0.145	0.076	14.496	9.022

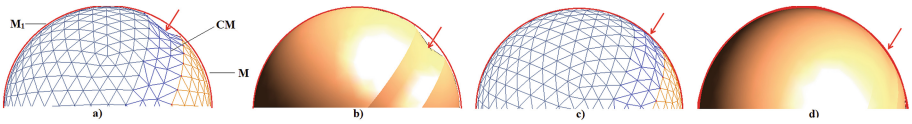

Fig. 9. The surface continuity of Sphere preserved after applying CM2D-TPW method with $s = 2$: (a)–(b) CM produced by linear interpolation; (c)–(d) CM produced by CM2D-TPW method.

in a small distance. It requires much more estimation which includes estimating the linear constraints on the control vertices as well as the off-surface constraints to construct and solve a linear system for each interpolated vertex. Therefore, the time of computation will be inevitably longer or the memory requirements may exceed the capacity of the computer. As a consequence, the runtime of this algorithm is rapidly increasing when the vertex numbers of the models increase as illustrated in Table 1. We have applied the algorithm to various 3D objects with complex shapes. The runtime increases quadratically. Moreover, the most critical disadvantage is that it is very important for the user to make a decision on the choice of the basis functions and the user parameter values, i.e. d-the signed distance and h-the shape parameter. This leads to the fact that the user chooses them by a rather costly trial and performs their numerical experiments over and over again until they end up with a satisfactory result consisting of the well-chosen values and a surface with a natural shape. In order to overcome these disadvantages, we have proposed a more reliable method, CM2D-TPW method. It produces surfaces of good approximation, computationally more efficient and occupied less memory compared to the C2MD-RBFW method.

7 Conclusion

We have introduced new simple and efficient mesh connection methods which join two meshes of different resolutions while maintaining the surface continuity and not destroying local curvatures. The wavelet transform and the methods of local approximation or interpolation are applied to position newly inserted vertices on the expected surface. Additionally, our methods keep the original bound-

aries of the meshes and the closest faces around these boundaries while connecting them. The connecting mesh is changed gradually in resolution between coarse and fine areas. CM2D-TPW method gives better results compared to CM2D-RBFW method since it improves the reconstruction capability of the connecting surface as illustrated by Gaussian curvatures and error evaluations. The advantages of CM2D-TPW method are: (1) It is simple, efficient, and local; (2) It generates smooth connecting surfaces; (3) There is no need to solve a system of linear equations. As a consequence, our algorithm is then numerically stable.

References

1. Dyn, N., Levin, D., Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. Graph. **9**, 160–169 (1990)
2. Loop, C.: Smooth Subdivision Surfaces Based on Triangles. Master's thesis (1987)
3. Husain, N.A., Bade, A., Kumoi, R., Rahim, M.S.M.: Iterative selection criteria to improve simple adaptive subdivision surfaces method in handling cracks for triangular meshes. In: Proceedings of the VRCAI 2010, pp. 207–210. ACM, USA (2010)
4. Husain, N.A., Rahim, M.S.M., Bade, A.: Iterative process to improve simple adaptive subdivision surfaces method with Butterfly scheme. World Acad. Sci. Eng. Tech. **79**, 622–626 (2011)
5. Pakdel, H., Samavati, F.F.: Incremental subdivision for triangle meshes. J. Comput. Sci. Eng. **3**(1), 80–92 (2007)
6. Barequet, G., Sharir, M.: Filling gaps in the boundary of a polyhedron. Comp. Aided Geometric Des. **12**(2), 207–229 (1995)
7. Hongbo, F., Tai, C.-L., Zhang, H.: Topology free cut and paste editing over meshes. In: GMP, pp. 173–184 (2004)
8. Stollnitz, E.J., DeRose, T.D., Salesin, D.H.: Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann Publishers, San Francisco (1996)
9. Mallat, S.G.: A Wavelet Tour of Signal Processing. Academic Press (1998)
10. Bertram, M., Duchaineau, M.A., Hamann, B., Joy, K.I.: Generalized B-spline subdivision-surface wavelets for geometry compression. IEEE **10**, 326–338 (2004)
11. Bertram, M.: Biorthogonal wavelets for subdivision volumes. In: Proceedings of the SMA 2002, pp. 72–82. ACM, New York (2002)
12. Sweldens, W.: The lifting scheme: a construction of second generation wavelets. SIAM J. Math. Anal. **29**, 511–546 (1998)
13. Casciola, G., Lazzaro, D., Monte-fusco, L.B., Morigi, S.: Fast surface reconstruction and hole filling using radial basis functions. In: Numerical Algorithms (2005)
14. Branch, J., Prieto, F., Boulanger, P.: Automatic hole-filling of triangular meshes using local Radial Basis Function. In: 3DPVT, pp. 727–734 (2006)
15. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Vis. Comp. Grap. **9**(1), 3–15 (2003)
16. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. SIGGRAPH Comput. Graph. **26**, 71–78 (1992)
17. Alexa, M., Rusinkiewicz, S., Adamson, A.: On normals and projection operators for surfaces defined by point sets. In: Eurograph Symposium on Point-Based Graph, pp. 149–155 (2004)