Sketch Based Mesh Fusion

Juncong Lin
State Key Lab of CAD & CG,
Zhejiang University, Hangzhou
310027, P. R. China
linjuncong@cad.zju.edu.cn

Xiaogang Jin
State Key Lab of CAD & CG,
Zhejiang University, Hangzhou
310027, P. R. China
jin@cad.zju.edu.cn

Charlie C. L. Wang
Department of Automation and
Computer-Aided Engineering,
The Chinese University of Hong Kong
cwang@acae.cuhk.edu.hk

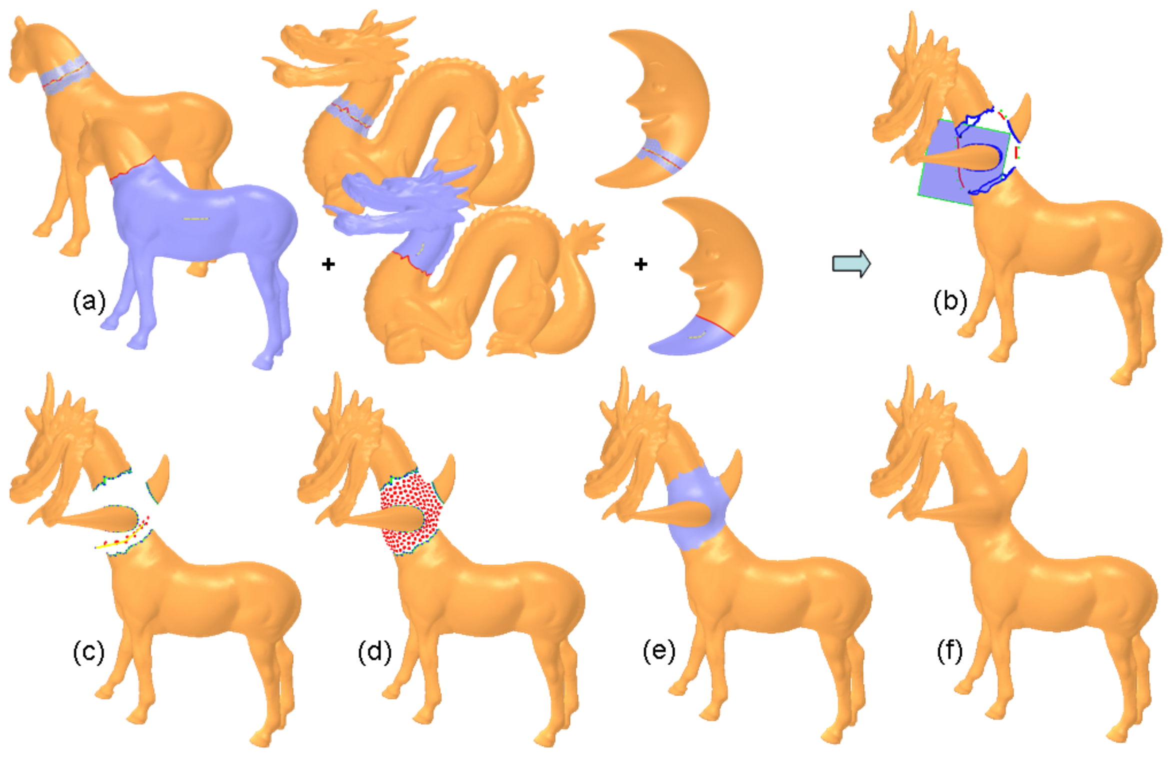

Figure 1: Screenshots of sketch based mesh fusion: (a) scissoring existing models, (b) placing the constructive parts together and sketching the silhouette of fusion, (c) specifying the part of transient surface to be remained, (d) sampling the transient surface by particles, (e) tessellating the particles into a mesh surface patch joining the separated parts, and (f) the final result of mesh fusion.

Abstract

The modelling method for creating 3D models in an intuitive way is far from satisfactory. In this paper, we develop a novel mesh fusion method controlled by sketches, which allows users (even novices) to construct complex 3D polygon models fast and easily. The user first cuts needed parts from some existing objects and puts them in right pose. Then, a radial basis function (RBF) based implicit surface is adopted to smoothly fuse the parts. To achieve better shape control for the transition part in fusion, our method let users using sketches to specify the expected silhouette. After that, the implicit surface is sampled by particles and meshed into a polygonal surface joining the separated parts into one single model. Compared with other previous methods, our mesh fusion approach overcomes the topological limitations and can merge multiple parts together at once.

Keywords: Topological-free mesh fusion, sketching, implicit surfaces, interactive modelling, design by examples

1 Introduction

Researches for providing a user-friendly modelling system to create complex 3D models quickly have been with a long history in computer graphics. A lot of works have been conducted in this area. However, it is still a tedious job even for those well-trained experts to construct desired models by using current commercial modelling system, not to say for novices. This paper aims at providing a simple and intuitive modelling approach for creating a complex 3D mesh model from the existing 3D polygonal objects.

From investigation (ref. [1]), we know that the sketching interface is still the most welcome manner for designers to model their creation in a computer system. Based on this reason, several sketching interfaces have been developed for modelling 3D objects (see [1, 2, 3]). The authors in [1] presented a system using intuitive guesses to govern the construction of geometric objects. Igarashi et al. in [2] extended the idea of [1] to develop a system for modelling 3D freeform objects in the representation of polygon mesh, while the approach in [3] implemented a similar sketching interface using variational implicit surfaces. However, the geometry of objects created in these systems is relatively simple. Later, the authors in [4, 5, 6] introduced several sketch interfaces by using various deformation techniques. Recently, Nealen et al. [7] presented a sketching tool for detail-preserved mesh editing. All these works [4, 5, 6, 7] extend the usage of sketch interface, but they focus on modifying existing models more than constructing a new object.

For geometry modelling, the Constructive Solid Geometry (CSG) provides an efficient way to construct complex geometric shapes by applying operations on primitive objects in a top-down manner. A wide variety of researches have been conducted in the CSG operations on the models represented in numerous methods [8, 9, 10, 11]. Unfortunately, it seems hard to directly apply the Boolean operations of CSG in mesh fusion, especially when there is a large hole between the separated parts to be fused - for instance, the model created in Figure 1. Furthermore, the robust implementation of Boolean operations on complex polygonal models is by no means an easy job.

Recently, several approaches [12, 13, 14, 15, 16, 17, 18, 19] in literature took a *design by example* way to construct complex 3D models from existing ones. The methods presented in [12, 13, 14, 15] focused on the cut-and-paste operation on meshes. They could produce seamless composed models. However, in [12, 13], the joined objects were required to be topologically equivalent to a disk for the necessary mapping between source and target models. [14] implemented a topological free cut-and-paste operation based on their new detail encoding technique, but still cannot overcome the topological limitation about the disk-like boundary. The authors of [15] developed a mesh fusion for topologically incompatible sections by constructing a transient surface between the combining objects; however, their approach required the boundary openings of a model lying on a plane. The authors in [16, 17] used differential approaches to transplant and merge meshes. However, the method in [16] was actually deformation based; thus, boundary openings with similar topological structures are required. In more detail, it is hard to directly fuse two models where one has two openings while the other is only with one opening. For [17], the same topological limitation existed. In [18], input models were cut using intelligent scissors. Then those cut portions were stitched to form a composition result by using the method of [13] - so that the boundaries with similar topology are required. The method presented in [19] was devoted to finding the best place to clip on two well aligned models, then clipping and stitching those retained model parts. To find the best clip place, they borrowed ideas from the graph theoretic-minimal cut operation. To clip and stitch the mesh, they also used the approach of [13]. In summary, there are various topological limitations in all above methods.

In this paper, we present a sketch-based modelling method by a modelling framework of mesh fusion, where the selected parts from complex models are merged through a transient implicit surface. The topological limitation in previous work is overcome in our mesh fusion approach. Our sketching interface developed can be used to construct complex 3D polygon models easily and efficiently. After cutting parts and putting them together, we model a RBF-based implicit surface to smoothly fuse the parts. To better control the shape of fused model, users can use sketches to specify expected silhouettes of the transient surface. At last, the transient surface is sampled and meshed into a polygonal surface patch joining the separated parts into a single model.

More specifically, the major contributions of this paper are as follows.

- A topology-free mesh fusion scheme is introduced, where a variational implicit surface based on RBFs is conducted as the transient surface to interpolate the position and normal constraints on the boundaries of given models. The transient surface overcomes the topological limitation on the openings of given models, so that a topology-free mesh fusion result can be achieved after tessellating the transient surface.

- As only part of the implicit surface belongs to the region to be fused, the conventional tessellation methods (e.g., [20, 21, 22]) cannot be directly applied here. A new tessellation method is developed in this paper to triangulate the transient implicit surface so that the separated polygonal parts are joined smoothly with triangles with good shapes.

- A novel sketching interface is provided to control the silhouette of the transient surface through strokes. The shape of the transient surface will follow the specified silhouette.

In the following, the modelling framework of mesh fusion is first described. After that, the sketching interface for scissoring parts and specifying profiles is introduced. To finally join the constructive parts, the particle-based tessellation scheme is presented in section 4. After showing the results in section 5, our paper ends with the conclusion section.

2 Framework of Mesh Fusion

The basic idea of the mesh fusion framework is to construct a transient surface interpolating the position and the normal constraints on the boundaries of the scissored models to be fused. To overcome the topological limitation on the openings, implicit surface is the best candidate for the transient surface. Here, we use one particular type of variatonal implicit surfaces - the RBF-based implicit surface to represent the transition surface.

Following [23], the RBF-based implicit surface is based on the thin-plate interpolation and can be expressed by a weighted sum of appropriate *radial basis functions* plus an affine term as

$$\Gamma(\mathbf{x}) = p(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \qquad (1)$$

where λ_is are weights, and $\{\mathbf{x}_i = \{x_i, y_i, z_i\}\}_{i=1}^{N}$ are the location constraints, and $p(\cdots)$ is a linear affine function of position in the form of

$$p(\mathbf{x}) = p_0 + p_1 x + p_2 y + p_3 z.$$

For the radial basis functions, we commonly use $\phi(\mathbf{x}) = \|\mathbf{x}\|^3$ in 3D, and $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log(\|\mathbf{x}\|)$ in 2D. For N position constraints, they can be rewritten as N linear equations as

$$\Gamma(\mathbf{x}_i) = p(\mathbf{x}_i) + \delta_i + \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) = f_i \qquad (2)$$

with $i = 1, \ldots, N$ and f_i is the function value shown on the location $\mathbf{x}_i$. The unknown λ_is in above linear equation system can be solved by adding the following compatible conditions

$$\sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} \lambda_i x_i = \sum_{i=1}^{N} \lambda_i y_i = \sum_{i=1}^{N} \lambda_i z_i = 0.$$

It is not difficult to find that in Eq.(2) we add some new scalars, δ_i. These parameters are conducted to control the property of the implicit surface defined by Eq.(2) (ref. [24]). When $\delta_i \neq 0$, the variational implicit function will approximate the corresponding constraint points rather than interpolate them. Every position constraint then becomes a weighted average of the interpolating position and the regularization position (which is actually a low-pass filtering), so that the results are smoothed.

Equipped by the above RBF-based implicit surface modelling method, the framework of topology-free mesh fusion can be elegantly constructed. Taking the models shown in Figure 1 as an example, we want to fuse the scissored models in Figure 1(a) into a single object. The mesh fusion result is determined in three phases.

In the first phase, an initial RBF-based implicit surface Γ is constructed to smoothly interpolate the boundary openings on the given separated models. To achieve the position interpolation, every vertex on the openings leads to a linear equation as shown in Eq.(2), where $\mathbf{x}_i$ is the Euclidean coordinate of the vertex. Since the transient surface should exactly pass all the vertices on openings, the function values of Γ shown on the vertices are zero (i.e.,$f_i = 0$). However, simply setting these position constraints leads to vanishing side conditions, so that the surface Γ cannot be determined by Eq.(2). Based on this reason and also to ensure the smooth interpolation, Turk and O'Brien in [25] added the normal constraints to the linear equation system. Here, we also define normal constraints in the same manner. Briefly, if $\mathbf{v}_i$ and $\mathbf{n}_i$ are the position and the surface normal at a vertex, a normal constraint is added by placing a position constraint at $\mathbf{v}_i + k\mathbf{n}_i$ with the function value $f_i = k$, where k is some small value. We use $k = 0.1$ in all of our examples. The surface normal at a vertex is computed by angle-weighted average of the normals on its adjacent faces [26].

In the second phase, the transient implicit surface Γ is adjusted by the user specified sketches. Our approach allows users to specify the shape of Γ through strokes (e.g, the strokes given in Figure 1(b)). To embed these inputs into our mesh fusion framework, the input strokes are first discreted into sampling points (in the screen plane) and then projected into the 3D space. The transient implicit surface is asked to approximately pass these sampling points. This can be achieved by adding the projected points into the RBF-based linear equation system (i.e., Eq.(2)) as position constraints. Note that in order to remove the dithering which is usually given on the user inputs, we only request the surface Γ to approximate but not to interpolate the sampling point by using $\delta_i \neq 0$.

The transient surface constructed so far is represented implicitly in a scalar function. Therefore, the last phase of our approach is to tessellate the transient surface into meshes so that the fused polygonal object is finally determined. However, as mentioned above, only one part of the implicit surface belongs to the transient region. The transient region and the non-transient region on the implicit surface Γ are separated by the openings on the given models. A user stroke is employed to specify the transient region (Figure 1(c)). By this specification, the tessellation scheme is able to avoid tessellating the non-transient region. Besides, for preventing the artificial results, the tessellation scheme is desired to generate well-shaped triangles that are adaptive to the mesh densities on the models to be fused. Here, we develop a particle sampling based tessellation method to generate the mesh surface smoothly joining the given models. Figure 1(d-e) show the particles sampled and the tessellation result. Details about tessellation will be presented in section 4 after introducing the sketching interface in the following section.

3 Sketching Interface

This section addresses the sketching interface implemented for letting users easily scissor and fuse the given polygonal models. The illustration for all these sketching tools is given in Figure 2.

3.1 Scissoring sketching

The scissoring of existing models is governed by user's strokes. Each stroke is projected onto the surface of the given model with a user-specified width (r in pixels) along the current viewing direction. The strokes actually specify an uncertain region where the cut following the nature seams of the mesh will be computed. Our implementation follows the intelligent scissoring method presented in [18] - computing a shorted path on the edges weighted with their dihedral-angle by using the Dijkstra's algorithm. The same as [18], the cutting path passes along the polygonal edges.

3.2 Sketching for selection

Two types of sketches are implemented in our system for selection. The first one is adopted to specify the part to be cut off after the cutting

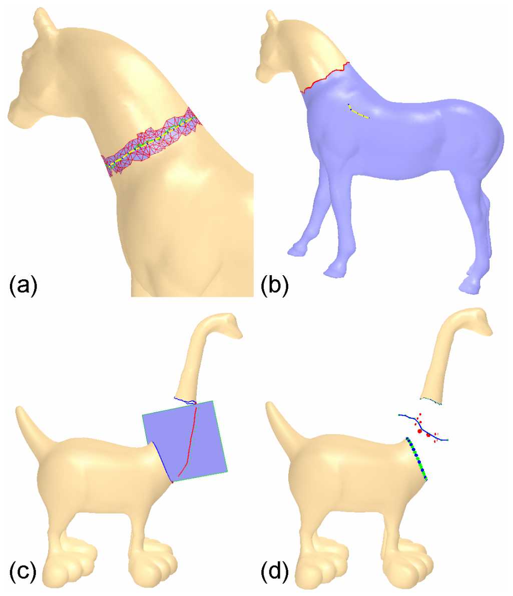

Figure 2: Different sketches in our system: (a) *scissoring sketch* used for cutting input models, (b) *selecting sketch I* for the selection of the remained part, (c) *silhouette sketch* specifying the profile of the transition surface, and (d) *selecting sketch II* used to select the sampling region of tessellation.

path is determined. The user input stroke is projected onto the model along the viewing direction, then the selected faces will serve as seeds in a flooding algorithm to determine the region. When the cutting edges are met, the flooding algorithm stops. The second type of selecting sketches is for the tessellation of the transient implicit surface, Γ. As mentioned above, since only part of Γ needs to be tessellated to fuse the existing models, we should prevent sampling the undesired regions. Therefore, the selecting stroke is first discreted into several points, and then the points are projected onto the implicit surface Γ along the viewing direction. The projected points then serve as seeds in the particle-based tessellation scheme.

3.3 Silhouette sketching

This sketching method is employed to control the silhouette of the transient surface. The user input is gathered as a collection of screen-space points, where the 2D points arrive at a rate that we cannot control. We resample these points so that they are not bunched up too closely. Here, the algorithm of Igarashi et al. [2] is used: when there are too many points close together, we simply ignore some. In detail, if the distance between the previous point and the current one is less than 10 pixels, we do not add the current point into the point list, P_{list}.

The planar coordinates of the sampling points are easily determined by the coordinate transformation between the screen space and the object space. The same as all other sketching interfaces, the most difficult task is to determine the depth coordinates. Our solution is as illustrated in Figure 3. The silhouette sketching is assumed to specify the profile of the transient surface between two openings. First of all, we find the two openings A and B by the closest boundary points to the starting and ending points of P_{list} in the screen plane. The two ending points $\mathbf{P}_A$ and $\mathbf{P}_B$ on the silhouette of these two openings are then searched. After that, a plane Ω passing $\mathbf{P}_A$ and $\mathbf{P}_B$ whose normal vector is closest to the viewing vector is determined. The depth coordinates of all points in P_{list} are finally computed by projecting them onto the plane Ω. The projected points are then added into the RBF-based linear equation system (i.e., Eq.(2)) to control the shape of the transient implicit surface, Γ.

However, only adding the project points of P_{list} is not enough. We cannot ensure that the points falling on the silhouette of Γ. From the definition of silhouette in computer vision, we know that for a point on a silhouette, the surface normal on this point is perpendicular to viewing direction. Therefore, the normal constraints on the points in P_{list} also need to be added. With the tangent vector $\mathbf{t}_p$ at $\mathbf{p}$ in the plane Ω and the viewing vector $\mathbf{n}_v$, the normal on a point $\mathbf{p}$ is defined by $\mathbf{n}_p = \mathbf{t}_p \times \mathbf{n}_v$. The constraint of $\mathbf{n}_p$ is then added following the manner of the normal constrain on openings in the framework of mesh fusion - converted into a set of position constraints with a small offset.

4 Transient Surface Tessellation

After the RBF-based implicit surface Γ has been generated, we need to tessellate the transient region of the surface to construct the final model of mesh fusion. In general, there

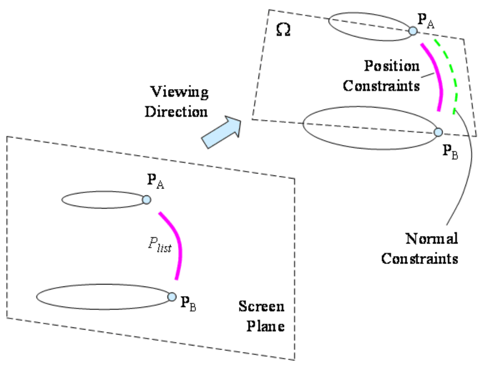

Figure 3: Illustration for determine the depth coordinate for silhouette sketching.

are two categories of methods for tessellating implicit surfaces: the spatial partitioning methods [20, 21, 22] and the continuation methods [27, 28]. Based on the requirements of mesh fusion, the tessellation scheme should provide the following abilities:

- Be able to distinguish the transient and the non-transient regions so that only the transient part is tessellated.
- Be able to construct the mesh compatible to the connectivity of openings on the scissored models.
- Be able to generate well-shaped triangles adaptive to the meshes on the models to be fused.

However, previous works rarely satisfy these requirements. Thus, a new tessellation scheme is developed. Our method first samples the implicit surface using a particle system, and then a triangular mesh surface is reconstructed by these sampling particles. The methodology of this scheme gives the potential for providing the above abilities. More specifically, the first one is satisfied as long as we only sample the transient region, the second ability could be provided by a constrained triangulation, and the last one is satisfied if we adaptively sample the transient surface and conduct an element-shape preserved triangulation method.

4.1 Compound particle sampling

First of all, we need to sample the transient region on Γ. Our particle sampling method borrows some idea from the method of Witkin and Heckbert [29], but with several necessary modifications and extensions. The same, we also diffuse particles on the implicit surfaces using the repulsion energy defined between the particles lying on Γ. For particle i, the energy due to another particle j is defined as:

$$E^{ij} = \alpha exp(-\frac{\|\mathbf{r}^{ij}\|^2}{2(\sigma^i)^2}), \qquad (3)$$

where $\mathbf{r}^{ij} = \mathbf{p}^i - \mathbf{p}^j$ is the vector between the positions of particles, σ^i is the particle repulsion radius, and α is a global repulsion amplitude parameter (we choose $\alpha = 6$ in all our examples). The totally energy at particle i is defined as $E^i = \sum_{j=1}^{n}(E^{ij} + E^{ji})$, which leads the repulsion force be proportional to the gradient of energy with respect to position p^i:

$$F^i = (\sigma^i)^2 \sum_{j=1}^{N} (\frac{\mathbf{r}^{ij}}{(\sigma^i)^2} E^{ij} + \frac{\mathbf{r}^{ij}}{(\sigma^j)^2} E^{ji}) \qquad (4)$$

By this repulsion force, the velocity of particle i in the diffusion on Γ is computed by

$$\mathbf{v}^i = F^i - \frac{\Gamma_x(\mathbf{p}^i) \cdot F^i}{\Gamma_x(\mathbf{p}^i) \cdot \Gamma_x(\mathbf{p}^i)} \Gamma_x(\mathbf{p}^i) \qquad (5)$$

which is just the orthogonal projection of F^i onto the surface's tangent plane at $\mathbf{p}^i$.

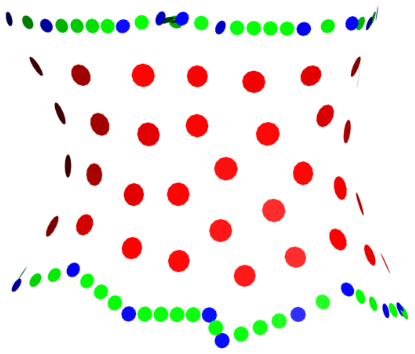

Figure 4: Our compound particle system: red points represent *dynamic particles*, blue ones are *boundary particles*, and green ones are *virtual particles*.

Different from [29], three types of particles are defined in our compound particle sampling scheme:

1. **Boundary Particles** (e.g., the blue ones in Figure 4): the particles coincide with the

boundary vertices on given models - these particles will keep static during the particle diffusion.

2. **Dynamic Particles** (e.g., the ones in red color in Figure 4): the particles that are dynamically moved on Γ driven by the repulsion forces. The particles are also adaptively re-sampled through a fission-death procedure.

3. **Virtual Particles** (coloured green in Figure 4): they will be inserted on the edge between two neighbouring boundary particles. The number of virtual particles between two neighbouring boundary particles is determined by the distance between the two boundary particles. The virtual particles also keep static. These particles and the boundary particles will prevent the dynamic particles from running out the bounded area.

The repulsion energy defined among different types of particles follows the same formula as in Eq.(3).

The boundary and the virtual particles are first generated on the openings of given models. Their repulsion radius are assigned as 0.8 times of the average edge length $\bar{L}_k$ on the corresponding boundary k - so that they can provide enough repulsion forces to prevent the dynamic particles moving outwards the bounded region. The number of virtual particles defined on a boundary edge with length L is computed by $\lceil L/(\varepsilon \bar{L}_k) \rceil$, where ε is a small number usually chosen between 0.05 and 0.1. For better illustration, we adopt $\varepsilon = 0.3$ in Figure 4.

As discussed in section 2, the seed particles generated from user strokes are then defined as dynamic particles. The initial repulsion radius of each particle is defined as the average edge length of all the boundary edges. The dynamic particles are moved on the implicit surface by a diffusion process with the velocity defined in Eq.(5). At the same time, the dynamic particles are processed by the following fission-death process when they reach the equilibrium (e.g.,$\|\mathbf{v}^i\| < 4\sigma^i$):

- **Fission:** a dynamic particle is fission if its repulsion radius is greater than the desired one (i.e.,$\hat{\sigma}^i$). Then, a single particle is split into two with initial radius $\sigma^i/\sqrt{2}$ and a very small random velocity. Also, to ensure particles not been moved beyond the region bounded by boundary/virtual particles, the fission is prevented when a particle is near boundary or virtual particles.

- **Death:** a dynamic particle dies if its repulsion radius is too small (e.g., $\sigma^i < 0.5\hat{\sigma}^i$) and $R = 1$, where R is a uniform random number between 0 and 1 to prevent mass suicide in overcrowded regions. The death particles are removed from the particle system.

In order to let the sampling adaptive to the mesh densities shown on the boundary of given models, we modify the sampling scheme to be resolution-dependent. For a dynamic particle i, a different desired radius is defined by

$$\hat{\sigma}_i = \hat{\sigma} M \sum_{k=1}^{M} (d_k^{-2} \bar{L}_k) / (\sum_{k=1}^{M} \bar{L}_k \cdot \sum_{k=1}^{M} d_k^{-2}) \quad (6)$$

where $\bar{L}_k$ is the average edge length of boundary k, d_k is the distance between particle i and the centroid of boundary k, and $\hat{\sigma}$ is the generally desired radius (we choose $\hat{\sigma}$ as the average edge length of all the boundary edges).

4.2 Triangulation

After sampling the transient surface, we reconstruct the mesh surface by the sampled particles so that the fused model is obtained. The intrinsic-property preserved method in [30] is adopted and modified to generate boundary compatible triangulations. The algorithm starts from the active edge of a seed triangle (i.e., any triangle on the boundary). For each active edge, a particle in its influence area is selected based on an intrinsic property of the point cloud. The active edge and the selected particle form a new triangle. A new active edge is then created while the original one becomes an inner edge. The particle selection and the triangle construction steps are repeated until there is no point left in the influence area of every active edge. When selecting a new particle, the weighted minimal length criterion proposed by Lin et al. [30] is adopted. The new particle for an active edge e_{ij} is selected to minimize the factor

$$k_{ij}\|\mathbf{p}_i-\mathbf{p}_j\|^2+k_{im}\|\mathbf{p}_i-\mathbf{p}_m\|^2+k_{jm}\|\mathbf{p}_j-\mathbf{p}_m\|^2$$

where $\{i,j,k\}$ is the existing triangle face adjacent to the active edge e_{ij} , and $\mathbf{p}_m$ is the selected points. k_{ij} is calculated by

$$k_{ij}=\frac{r_{ik}^2+r_{jk}^2-r_{ij}^2}{A_{ijk}}+\frac{r_{im}^2+r_{jm}^2-r_{ij}^2}{A_{ijm}}$$

and k_{im} and k_{jm} are approximated by

$$k_{im}=k_{jm}=2(r_{im}^2+r_{jm}^2-r_{ij}^2)/A_{ijm}$$

where r_{ij} representing the length of edge $\{i,j\}$ and A_{ijk} denoting the area of triangle $\{i,j,k\}$.

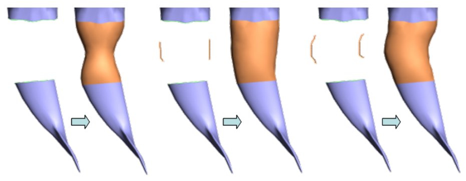

Figure 5: Comparison of fusion with and without silhouette sketching: left column, no silhouette is defined; middle column, the generated transient surface by silhouette sketching is good; right column, more adjustment can be given.

5 Results and Discussion

Our first example shown in this section (Figure 5) is employed to compare the results generated with vs. without silhouette sketching. In the left column of Figure 5, the transient surface tends to shrink due to the reason that the distance between two models to be fused is too far. To improve the shape of the fused object, two strokes are specified in the middle column of Figure 5 to specify the silhouette of the transient surface so that a much better result is obtained. Furthermore, we can even go on modifying the shape by sketching some new profiles (see the right column of Figure 5 - a mermaid with bended tail is produced).

The second test (see Figure 6) is to show the effect of the parameter δ conducted in specifying the silhouettes. As mentioned above, letting $\delta=0$ will generate a surface interpolating the projected silhouette strokes which usually include dithering; when using $\delta\neq 0$, smoother results are obtained.

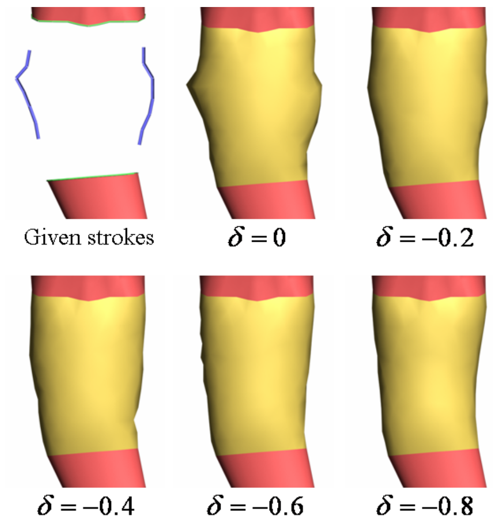

Figure 6: A comparison of approximation and interpolation schemes with different δ.

Figure 7 shows the comparison between the result generated by the uniform particle sampling (letting $\hat{\sigma}_i=\hat{\sigma}$) and the result from the adaptive sampling (using Eq.(6)). It is easy to find that the tessellating result with adaptive sampling gives better triangle-shape near the upper boundary of the fused region (see the part circled in dash lines in Figure 7).

More examples can be found in Figure 8 and 9, where in Figure 8 the bear head is fused onto the body of a rhinoceros and the two wings of the gargoyle are cut and fused onto the killwhale in Figure 9.

We test our examples on a PC with Inter Pentium IV 2.4GHz CPU + 512M RAM. Table 1 summarizes the timing statistics for these examples. Generally, our scheme takes less than 1 second for RBF fitting, and several seconds for tessellation, which satisfies the speed requirement of interactive modelling tools. We use the matcom math library [31] to solve the linear system of RBF fitting (Eq. (2)).

Limitations: The current implementation of our approach presented in this paper has the following limitations.

- Firstly, seed particles in the tessellation scheme are specified by the region selecting strokes. In some extreme cases, it could

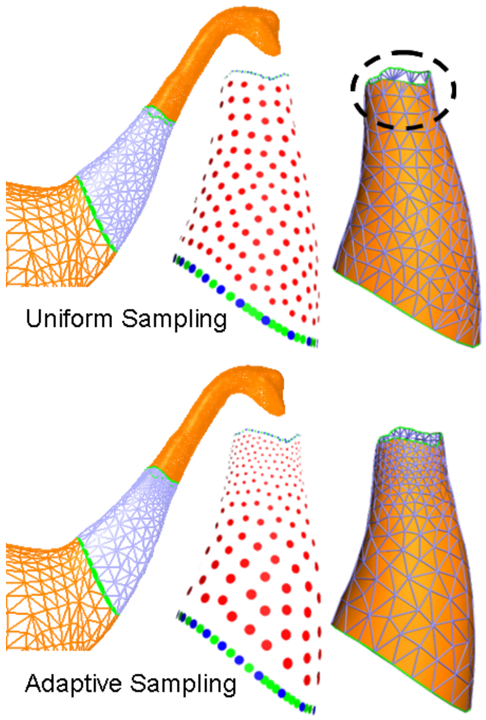

Figure 7: Tessellation results from the uniform sampling vs. the adaptive sampling.

Table 1: Computing time taken in examples

Figure	RBF Fitting		Tessellation	
	Constraint Number	Time (sec)	Particle Number	Time (sec)
Fig. 1	574	0.592	578	4.522
Fig. 8	486	0.409	301	3.810
Fig. 9	330	0.207	250	3.924

be possible that no point from the strokes can be projected onto the transient implicit surface. Our current approach solves this problem in a trial-and-error manner, i.e., warning messages are shown to ask users to sketch again when no successful projection is found. It becomes more dangerous when the transient region consists of two connexions. If seeds are sketched on only one connexion, even the warning message will not be shown. An automatic method to generate seeding particles is expected.

- The second limitation comes from the assumption in the silhouette sketching. Our current method assumes that every stroke generates a curve linking to the endpoints of two openings in the screen plane. This is relative simple. In a more complex interface, one silhouette can be specified by several strokes as what is used in charcoal drawings.

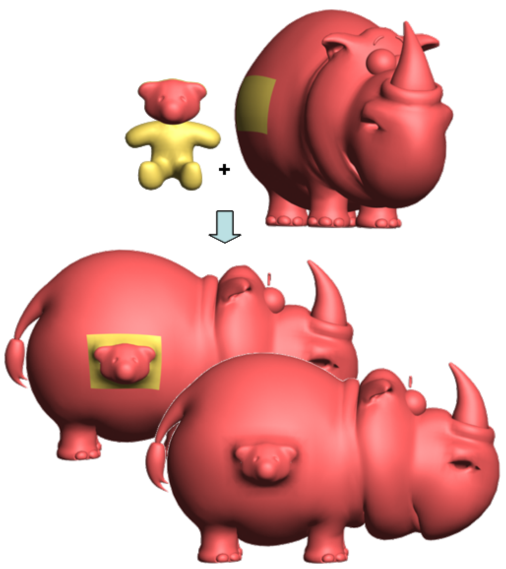

Figure 8: The head of bear is cut and fused onto the rhinoceros.

- The third limitation of our sketch-based mesh fusion is that it always generate smooth transient surfaces lacking complex details. It is caused by the nature of RBF-based implicit surfaces.

6 Conclusions

In this paper, we developed a novel sketch-based mesh fusion interface for creating 3D models from existing ones. The mesh fusion is controlled by sketches which are intuitive and are easy to use even for novices. Complex 3D models can be constructed in a very short time. The developed sketching interface is based on a modelling framework of mesh fusion, where the transient surface interpolating the boundary openings on the given models is first modelled by a RBF-based implicit function and then meshed into a polygonal surface by an adaptive tessellation scheme. Compared to other previous methods, our method overcomes the topological limitation and multiple parts can be merged together at once.

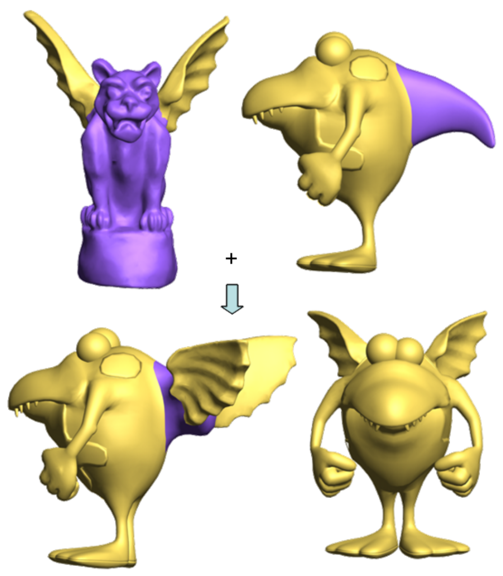

Figure 9: Two wings cut from the gargoyle are fused onto the kill-whale.

Several topics will be further investigated in our future research to overcome the limitations existed in our current implementation. For the automatic generation of seeding particles in tessellation, our first thought is to move all boundary vertices a little bit along the cross-tangents (pointing outwards the existing model) and then project them onto the implicit surface to serve as seeds. For the issue of detail preserved mesh fusion, we are going to treat it as a post-processing. After fusing the models by our current method, we blend the geometry details between fused objects (similar to the method conducted in [17]); or we can add the details back to the smooth transient surface by the means of geometric texture synthesis. Some more complex sketching tools are planned to be developed. For example, we are planning to use strokes to specify feature lines on models like [7]. Since our transient surface is implicitly represented, the anisotropic basis functions in [32] will be helpful to model the features mathematically on an implicit transient surface.

References

[1] R. Zeleznik, K. Herndon, and J. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of SIGGRAPH*, pages 163–170. 1996.

[2] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH*, pages 409–416. 1999.

[3] O. Karpenko, J. Hughes, and R. Raskar. Free form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3):585–594, 2002.

[4] Y. Kho and M. Garland. Sketching mesh deformations. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games*, pages 147–154. 2005.

[5] G. Draper and P. Egbert. A gestural interface to free-form deformation. In *Proceedings of Graphics Interface*, pages 113–120. 2003.

[6] J. Hua and H. Qin. Free-form deformation via sketching and manipulating scalar fields. In *Proceedings of the ACM symposium on Solid Modelling and Application*, pages 328–333. 2003.

[7] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics*, 24(3):1142–1147, 2005.

[8] B. Adams and P. Dutre. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics*, 22(3):651–656, 2003.

[9] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of SIGGRAPH*, pages 185–194. 2001.

[10] K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. *ACM Transactions on Graphics*, 21(3):330–338, 2002.

[11] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics*, 22(3):641–650, 2003.

[12] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution subdivision surfaces. *ACM*

Transactions on Graphics, 21(3):312–321, 2002.

[13] T. Kanai, H. Suzuki, J. Mitani, and F. Kimura. Interactive mesh fusion based on local 3d metamorphosis. In *Graphics Interface*, pages 148–156. 1999.

[14] H. Fu, C.L. Tai, and H. Zhang. Topology-free cut-and-paste editing over meshes. In *Geometric Modeling and Processing*, pages 173–184. IEEE Computer Society Press, 2004.

[15] J. Lin, X. Jin, C.C.L. Wang, J. Feng, and H. Sun. Topology free mesh fusion. In *Pacific Graphics*, pages 38–40. 2005.

[16] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):664–651, 2004.

[17] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rossl, and H.P. Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, pages 179–188. 2004.

[18] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modelling by example. *ACM Transactions on Graphics*, 23(3):652–663, 2004.

[19] T. Hassner, L. Zelnik-Manor, G. Leifman, and R. Basri. Minimal-cut model composition. In *Proceedings of the International Conference on Shape Modelling and Applications*, pages 72–81. 2005.

[20] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[21] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):153–169, 1987.

[22] J. Bloomenthal. An implicit surface polygonizer. In P. Heckbert, editor, *Graphic Gems IV*, pages 234–249. Academic Press, New York, 1994.

[23] M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum*, 24(3):611–621, 2005.

[24] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):346–359, 2002.

[25] G. Turk and J. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of SIGGRAPH*, pages 335–342. 1999.

[26] S. Jin, R.R. Lewis, and D. West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2):71–82, 2005.

[27] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles: range image fusion for complex object modeling. In *International Conference on Image Processing*, pages 381–384. 1996.

[28] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.

[29] A.P. Witkin and P.S. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, 28(2):269–277, 1994.

[30] H.W. Lin, C.L. Tai, and G.J. Wang. A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. *Computer-Aided Design*, 36(1):1–9, 2004.

[31] The MathWorks. http://www. mathworks. com.

[32] H.Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *International Conference on Computer Vision*, pages 606–613. 2001.