A Deformation Method for Three-Dimensional Complex Models Based on Semantic Parameters (Postprint)

Authors: Sun Xia, He Kunjin

Date: 2018-06-19T00:00:00+00:00

Abstract

To address the challenge of deforming 3D models containing free-form surfaces, this paper proposes a semantic parameter-based deformation method for complex 3D models, aiming to enable rapid 3D model design. The method utilizes deformation control points as the fundamental elements of model deformation and achieves hierarchical deformation through semantic parameters: deformation control points drive the deformation of grouped components, which in turn facilitate the overall deformation of the 3D model. First, the Hausdorff distance method and mean clustering method are employed to replace the free-form surfaces of the grouped electric vehicle components with quadratic surfaces. Then, based on the relationships among deformation control points and between these points and component vertices, the internal and external constraint conditions for each grouped component are calculated. Finally, by editing and modifying the semantic parameters defined on the deformation control points, customized model deformation is achieved. Verification experiments conducted on a 3D electric vehicle model demonstrate that the proposed method enables users to deform the model by modifying only a small number of semantic parameters, effectively improving the efficiency of personalized 3D model design.

Full Text

Preamble

Title: Research on 3D Complex Model Deformation Method Based on Semantic Parameters

Authors: Sun Xia, He Kunjin† (College of IoT Engineering, Hohai University, Changzhou, Jiangsu 213022, China)

Abstract: This paper proposes a 3D complex model deformation method based on semantic parameters to address the difficulty of deforming 3D models containing free-form surfaces, aiming to enable rapid design of 3D models. This method utilizes deformation control points as basic elements for model deformation and achieves hierarchical deformation by setting semantic parameters: deformation control points drive the deformation of grouped components, which in turn facilitates overall deformation of the 3D model. First, the Hausdorff distance method and mean clustering method are employed to replace the free-form surfaces of an already-grouped electric vehicle with quadric surfaces. Then, based on the relationships between deformation control points and component vertices, internal and external constraints for each grouped component are calculated. Finally, semantic parameters defined on the deformation control points are edited to achieve customized model deformation. Experimental verification using an electric vehicle 3D model demonstrates that this method supports users in deforming models by modifying a small number of semantic parameters, effectively improving the efficiency of personalized 3D model design.

Keywords: cluster analysis; quadric surface; component constraint; semantic parameter

0 Introduction

In recent years, scholars both domestically and internationally have conducted research on 3D model deformation technology, achieving certain results. However, existing methods have limitations when applied to complex models. Some approaches allow users to freely select deformation directions by setting deformation handles, but are not suitable for models composed of complex polygons. Others propose methods for rapid generation of automotive prototypes based on feature semantics, yet the semantic parameters are limited and cannot cover all aspects of volumetric features. While hierarchical parameterization frameworks for free-form surface features enable intuitive editing at the semantic level, they fail to establish robust inter-layer constraint relationships. Methods for local surface reconstruction and featurization based on feature lines can edit local regions but cannot automatically add constraint lines. Some techniques use semantic attributes to create geometric deformations, but excessive attribute types may cause deformation mode failure. Deformation methods based on n-dimensional surface shape spaces offer good robustness for local shape shifting and overlapping, but at high computational cost. Model reconstruction approaches based on landmark depth estimation and shape deformation can produce realistic 3D models, though the precision of 3D landmark estimation affects detail-rich regions. Improved mesh deformation methods based on Delaunay background graphs enhance deformation quality and efficiency.

On the other hand, maintaining original geometric details during 3D model deformation remains a challenge. Global optimization algorithms based on

Laplacian coordinates preserve mesh quality and geometric details well, but are limited to STL file models. Deformation algorithms based on local rigidity constraints solve distortion problems in joint areas during large-scale character model deformation, though complexity increases with skeleton nesting complexity. Skeleton-driven mesh deformation methods based on subdivision are suitable for geometrically rich complex models, but the constructed control mesh cannot fit the model surface shape well. Data-driven shape descriptors address variations and noise in structural deformation, maintaining good robustness even under large deformations.

Although customized products have entered the market, both mass customization and private customization face challenges: the design process is time-consuming and expensive, or lacks high-level semantic parameters with real-time display of product appearance. Therefore, this system establishes functionally different deformation control points and internal/external constraints for model components, providing users with a custom deformation interface. This enables intuitive editing of deformation control point 3D information through high-level semantic parameters, achieving hierarchical model deformation. Using an electric vehicle as the deformation model, this paper validates the proposed method, focusing on free-form surface replacement for each component, setting three types of deformation control points, and calculating constraints during deformation.

1 Electric Vehicle Parametric Deformation Framework

The 3D complex model deformation process based on semantic parameters generally includes replacing free-form surfaces, setting deformation control points, calculating internal and external constraints for grouped components, and parameterizing control points. The workflow is illustrated in Figure 1 [Figure 1: see original paper].

According to the above design process, the electric vehicle parametric deformation steps are as follows: (a) import an existing electric vehicle model and simplify and group its surfaces; (b) cluster surfaces of each grouped component and replace the free-form surfaces constituting the electric vehicle with quadric surfaces; (c) establish constraint conditions, including setting deformation control points and internal/external constraints for grouped components; (d) parameterize control points to achieve electric vehicle deformation.

2 Simplified Electric Vehicle Model Generation

After simplifying and grouping the electric vehicle 3D model, the vehicle is divided into five parts: front, seat, body, front wheel, and rear wheel. First, sub-surfaces of each grouped component are clustered, followed by secondary clustering of sub-surface classes to group geometrically similar surfaces together. Then, the likelihood of free-form surfaces in each class belonging to quadric

surfaces such as planes, spheres, or cylinders is evaluated, and free-form surfaces are replaced with quadric surfaces.

2.1 Clustering Based on Free-Form Surfaces

Each grouped component consists of multiple free-form surfaces. To collect feature information from these surfaces, their categories must be discriminated. Sub-surface clustering groups highly similar surfaces into classes, and mean shift clustering further regroups surfaces with matching geometric feature information.

2.1.1 Sub-Surface Clustering This paper employs the Hausdorff distance method to discriminate pairwise similarity of sub-surfaces within each grouped component. Hausdorff distance describes the similarity between two point sets, defined as the maximum distance from points in one set to their nearest points in the other set. The Hausdorff distance method is used to compare similarity among surfaces in the five grouped components (front, seat, body, front wheel, rear wheel). For any two surfaces A and B in a component, their Hausdorff distance H(A,B) is calculated and compared with a threshold D. When H(A,B) $<$ D, surfaces A and B are considered similar and can be grouped into the same class.

The mathematical definition of Hausdorff distance is:

$$H(A, B) = \max(h(A, B), h(B, A))$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

$$h(B, A) = \max_{b \in B} \min_{a \in A} \|b - a\|$$

The threshold D is calculated as:

$$D = \frac{h(A, B) + h(B, A) + h_1(A, B) + h_1(B, A)}{4}$$

where

$$h_1(A, B) = \min_{a \in A} \min_{b \in B} \|a - b\|$$

$$h_1(B, A) = \min_{b \in B} \min_{a \in A} \|b - a\|$$

After initial sub-surface clustering of each component (e.g., the body), several sub-surface classes c = {c , c , c , ···, c } are obtained. For class c , feature information x = {e , e , e , n, k} is calculated for each sub-surface, where e , e , e are principal component vectors describing vertex distribution; n is the surface normal vector; and k = {k , k , k , k > k > k } represents Gaussian curvature at vertex j.

Gaussian curvature is calculated according to the Gauss-Bonnet theorem:

$$k_j = \left| 2\pi - \sum_i \theta_i \right| / \sum_i A_i$$

where A and are the area and angle of all triangles related to vertex j, respectively. The Gaussian curvature values of all vertices in a sub-surface are used as statistical data (with 10 bins), and k , k , k are the average values within the bin ranges having the highest frequencies in the histogram.

Using the front component containing 15 free-form surfaces as an example, three sub-surface classes c = {c , c , c } are obtained via Hausdorff distance, containing 3, 4, and 8 surfaces respectively. Table 1 shows feature information for each free-form surface in sub-surface class c .

Table 1 Feature information of free-form surfaces in c

e , e , e	k , k , k
-0.52, 0.17, 0.84	0.80, 0.27, 0.54
0.90, 0.50, 0.00	-0.90, -0.10, -0.30
0.31, -0.94, 0.00	0.00, -1.00, 0.00
1.00, 0.00, 0.00	-0.10, -0.10, 0.00
-0.30, -0.90, -0.10	0.00, 0.00, -1.00
0.28, -0.40, -0.87	-0.96, -0.13, -0.25
0.30, -1.00, 0.00	-0.50, -0.80, -0.20
-0.18, 0.90, -0.42	

The standard deviation of point sets for free-form surfaces in class C is calculated as = {0.13, 0.09, 0.13}. Based on probabilities P = 0.99, P = 0.48, and P = 0.66, the free-form surfaces in class C are replaced with planes. Figure 2 [Figure 2: see original paper] shows the simplified electric vehicle model after replacing free-form surfaces with quadric surfaces.

2.1.2 Mean Shift Clustering Secondary clustering is performed on the five grouped components. First, sub-surface classes c = {c , c , c , ···, c } of a component are used as categories to be clustered. Then, mean shift algorithm is applied to compute secondary clustering classes C = {C , C , C , ···, C }.

The mean shift vector is defined as:

$$m(x) = \frac{\sum_{x_i \in N(x)} x_i \cdot e^{-\|x_i - x\|^2}}{\sum_{x_i \in N(x)} e^{-\|x_i - x\|^2}} - x$$

where m(x) is the mean shift vector, x is a 15-dimensional feature vector of a sub-surface, N(x) is the neighborhood of x, and x is any vector in N(x). When m(x) converges to a sufficiently small value, if the distance between the current N(x) center and an existing class center is small enough, they are merged; otherwise, N(x) becomes a new class. Based on sub-surface classes c, the mean shift algorithm yields surface classes C = {C , C , C }, where C , C , C contain 3, 4, and 8 free-form surfaces respectively.

2.2 Replacement Based on Quadric Surfaces

After surface clustering, a grouped component (e.g., the body) generates surface classes C = {C , C , C , ···, C }. Free-form surfaces in these classes are replaced with quadric surfaces, which facilitates setting deformation control points with semantic parameters based on quadric surface features.

For each surface class C , the probabilities of belonging to plane (P), sphere (P), or cylinder (P) are calculated using discriminant formulas:

$$P_p = \frac{n_p}{n_i} \cdot \frac{C}{\sigma_1 + \sigma_2 + \sigma_3}$$

$$P_s = \frac{n_s}{n_i} \cdot \frac{C}{\sigma_1 + \sigma_2 + \sigma_3}$$

$$P_c = \frac{n_c}{n_i} \cdot \frac{C}{|\sigma_1 - \sigma_2 - \sigma_3|}$$

where n is the number of surfaces in class C , and , , are standard deviations of point sets for any surface i in C . If P 0.9, P 0.9, or P 0.9, all surfaces in class C are represented by the quadric surface feature with the highest probability. If all probabilities are < 0.9, surfaces maintain their original free-form characteristics.

Using the front component as an example, surface classes C = {C , C , C } are obtained. The standard deviations for three free-form surface point sets in class C are = {0.13, 0.09, 0.13}, = {0.10, 0.00, 0.60}, and = {0.09, 0.00, 0.05}. With P = 0.99, P = 0.48, and P = 0.66, the free-form surfaces in class C are replaced with planes.

3 Constraint Generation Based on Point Cloud Model

Based on the components to be deformed, three types of deformation control points are defined on the quadric surfaces of each grouped component. Constraints within and between components are calculated to improve deformation rationality.

3.1 Classification of Deformation Control Points

Different types of deformation control points are set on each grouped component of the simplified electric vehicle model. Figure 3 [Figure 3: see original paper] shows deformation control points on the five grouped components, with different shapes representing different types (e.g., dots for body control points). Based on their roles during deformation, control points are classified into three categories:

a) **User-edited control points**: Points operated by users for vehicle deformation, containing defined semantic parameters that are automatically selected according to user requirements for appearance deformation.

b) **System-fixed control points**: Points that automatically maintain model fixation, having global symmetry with user-edited control points to reduce the number of fixed points users must mark.

c) **System-manipulated control points**: Points that automatically realize component deformation, determined jointly by user-edited and system-fixed control points. Their position updates are optimized by:

$$\text{minimize} \sum_{j=1}^{N} \|s_j - m_j\|^2, \text{subject to } t_j = u_j, \forall t_j \in T$$

where s are the updated positions of system-manipulated control points, N is their number, T is the deformation type (translation, rotation, scaling along X, Y, Z axes), t is any deformation category, u is user-specified deformation category, and m , v are the mean and variance of vertex information in the region manipulated by system-manipulated control points.

3.2 Constraint Setting Based on Deformable Components

Deformation design from local (e.g., body component) to global (electric vehicle) involves constraint limitations within single components and between different components to ensure orderly deformation. When a component M deforms, internal constraints limit changes within M, while external constraints include constraints on other components MM = {M , M , M , ···, M } and M' s influence on them. Assuming the body is the deformable component M, then MM = {M , M , M , M } correspond to front, seat, front wheel, and rear wheel components respectively.

3.2.1 Internal Constraints of Deformable Components Figure 4(b) shows internal constraints for the body component. Using the component' s centroid, influence domains for user-edited control point e and system-fixed control point a are divided into Area1 and Area2. The influence factor k of point e on vertex p in Area1 is calculated to obtain p' s new position p , where D(e, p) is the distance between e and p .

The constraint condition F(e, p) is defined as three types: - When D(e, p) < 0.1: k = 1, p = p - When 0.1 < D(e, p) < 1: Find minimum D and maximum D distances in this range, then k = 1/(D + D), p = p + temp · k - When D(e, p) > 1: k = 1/D(e, p), p = p + temp · k

Similarly, the position p influenced by point a on vertex p in Area2 can be obtained.

3.2.2 External Constraints of Deformable Components Using body length increase as an example, constraints between deformation control points p = {e, a, E}, where E = {E , E , E , E }, are calculated. As shown in Figure 4(c), point e on the body is a user-edited control point, point a is system-fixed, and E is the set of system-manipulated control points located on other components: E = {e , e } (front), E = {e , e } (seat), E = {e } (front wheel), E = {e } (rear wheel). Using Equation (5), updated positions for E = {e , e , e , e , e , e } are obtained.

Taking the seat component M as an example, its centroid divides influence domains for e and e as Area21 and Area22. Vertices in different influence domains deform with the seat component according to the same constraints. Other components deform based on their respective constraints.

Establishing constraints between adjacent components effectively reduces overlapping or floating phenomena during deformation. The steps include: (a) Calculate influence regions EA = {EA , EA , EA , ···, EA } of component M on adjacent components, where M' s influence on M , M , M , ···, M corresponds to EA , EA , EA , ···, EA ; (b) Compute constraint conditions between any vertex p in influence region EA and deformation control points on M; (c) Obtain new positions of p in EA after deformation based on these constraints.

Using the influence of body component M on seat component M as an example: First, the influence domain EA is obtained from the 3D overlapping space between M and M . Then, based on user-edited control point e' s constraint F(e, p) on vertex p , constraints between points e, a and p in EA are derived. Similarly, constraints between body component M and front M , front wheel M , and rear wheel M are obtained. Figure 4(d) shows the deformed electric vehicle model after the body component grows by distance temp.

4 Examples

The proposed method was tested on VS2010 and Unity 5.5.3 platforms. Custom parameter-based electric vehicle deformation is illustrated through examples of front wheel and seat component deformation.

Front Wheel Component Parameterization: Figure 5 shows front wheel deformation. First, the simplified model is obtained through surface replacement, and a user-edited control point e is set on the front wheel (Figure 5(a)), where p is any point on the wheel, D is the vector from e to p , and R is the initial radius. Based on the special nature of the wheel' s quadric surface, the wheel with increased radius R is obtained via p = p + D · R (Figure 5(c)). Using Equation (5), updated positions of system-manipulated control points on other components are calculated to achieve their deformation. The influence region on the body component is determined from the front wheel' s 3D space (Figure 5(b)), and internal constraint relationships are applied to constrain local body deformation in this region, preventing penetration or misalignment. Finally, semantic parameters defined at point e are edited to enlarge the wheel while driving changes in other components, achieving hierarchical deformation (Figure 5(d)).

Seat Component Parameterization: Figure 6 shows seat deformation. After obtaining the simplified model and setting different functional control points, user-edited point e and system-fixed point a are placed on the seat component (Figure 6(a)), where D(e, p) is the distance from e to p and L is the initial seat length. First, influence domains for e and a are divided by the seat' s centroid, and the seat component with increased length L is obtained by applying constraints (Figure 6(c)). Then, updated positions of system-manipulated control points on other components are calculated using Equation (5) to deform the front, body, front wheel, and rear wheel components. External constraints are computed for the body component within the influence domain (divided by the seat' s 3D space, Figure 6(b)) to ensure tight fitting during deformation. Finally, semantic parameters at the user-edited point are modified to grow the seat while achieving overall vehicle deformation (Figure 6(d)).

During electric vehicle parametric deformation, users modify semantic parameters defined on user-edited control points, driving overall vehicle deformation through component-level changes and simplifying the complex deformation design process.

5 Conclusion

Based on existing electric vehicle 3D models, this paper proposes a 3D complex model deformation method based on semantic parameters. By replacing free-form surfaces with regular quadric surfaces, this method effectively solves the deformation difficulty of complex models composed of free-form surfaces. Through establishing constraint relationships from the original to target model—including constraints between deformation control points and internal/external con-

straints of components—orderly and hierarchical deformation of electric vehicle models is achieved. However, automatic grouping of electric vehicle components remains challenging, and the defined constraint conditions are limited, making it difficult to establish constraints between every part. Consequently, deformation of detailed electric vehicle parts is insufficient, which will be addressed in future work.

References

[1] Yumer M E, Kara L B. Co-constrained handles for deformation in shape collections [J]. Association for Computing Machinery Transactions on Graphics, 2014, 33 (6): 1-11.

[2] Zhu Danmo, Lu Yang, Xu Xin, et al. Rapid car-body prototype generation method based on feature semantics [J]. Journal of System Simulation, 2013, 25 (9): 1990-1995.

[3] He Kunjin, Zhao Zongxing, Geng Weizhong, et al. Parametric representation and implementation of freeform surface feature based on layered parameters [J]. Journal of Computer-Aided Design and Computer Graphics, 2014, 26 (5): 826-834.

[4] He Kunjin, Wang Lin, Chen Zhengming, et al. Reconstruction and featurization of local region based on CAD surface models [J]. Computer Integrated Manufacturing System, 2014, 20 (10): 2360-2368.

[5] Yumer M E, Chaudhuri S, Hodgins J K, et al. Semantic shape editing using deformation handles [J]. Association for Computing Machinery Transactions on Graphics, 2015, 34 (4): 86.

[6] Demisse G G, Aouada D, Ottersten B. Deformation based curved shape representation [J]. IEEE Trans on Pattern Analysis & Machine Intelligence, 2017, PP (99): 1-1.

[7] Liu P, Hong M, Wang M, et al. 3D face reconstruction via landmark depth estimation and shape deformation [J]. Multimedia Tools & Applications, 2017, 76 (2): 2749-2767.

[8] Jiang Bo, Zhu Mucheng, Zeng lei, et al. Improved mesh deformation method based on Delaunay background map [J/OL]. Application Research of Computers, 2018, 35 (4). (2017-03-31) [2017-03-31]. http://www.arocmag.com/article/02-2018-04-005.html.

[9] Xu bin, Li lǜke. Laplacian mesh optimization algorithm based on STL file [J]. Application Research of Computers, 2013, 30 (5): 245-248.

[10] Xu bin, Yu Yong, Tan Ying. Differential mesh deformation algorithm based on local rigidity constraint [J]. Application Research of Computers, 2015 (5): 1580-1585.

[11] Zhang Xiangyu, Li ming, Ma Xiqing. Skeleton-driven mesh deformation technology based on subdivision [J]. Journal of Computer Applications, 2015, 35 (3): 811-815.

[12] Dai G, Xie J, Zhu F, et al. Learning a discriminative deformation-invariant 3D shape descriptor via many-to-one encoder [J]. Pattern Recognition Letters, 2016, 83: 330-338.

[13] Tian Baozhen, Yu Suihuai, Wang Shuxia, et al. Module partition method of multi-constrained for customized products [J]. Computer Engineering and Applications, 2016, 52 (19): 234-240.

[14] Kim Y J, Oh Y T, Yoon S H, et al. Efficient Hausdorff distance computation for freeform geometric models in close proximity [J]. Computer-Aided Design, 2013, 45 (2): 270-276.

[15] Carreira-Perpiñán Miguel Á. A review of mean-shift algorithms for clustering [J]. arXiv: 1503.00687, 2015.

Note: Figure translations are in progress. See original paper for figures.