ACM Symposium on Solid Modeling and Application (2004)
G. Elber, N. Patrikalakis, P. Brunet (Editors)

Developability-Preserved Free-Form Deformation of Assembled Patches

Charlie C. L. Wang

Department of Automation and Computer-Aided Engineering,
The Chinese University of Hong Kong,
Shatin, N.T., Hong Kong
cwang@acae.cuhk.edu.hk

Kai Tang

Department of Mechanical Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, N.T., Hong Kong
mektang@ust.hk

Abstract
A novel and practical approach is presented in this paper that solves a constrained free-form deformation (FFD) problem where the developability of the tessellated embedded surface patches is preserved during the lattice deformation. The formulated constrained FFD problem has direct application in areas of product design where the surface developability is required, such as clothing, ship hulls, automobile parts, etc. In the proposed approach, the developability-preserved FFD problem is formulated as a constrained optimization problem. Different from other contained FFD approaches, the positions of lattice control points are not modified in our algorithm – as their control is insufficient in regarding to the developability of all the nodes in the mesh. Moreover, the optimization is performed on the parameters of the mesh nodes rather than directly modifying the 3D coordinates, which avoids the time-consuming inverse calculation of the parameters of every node in a non-parallelepiped control lattice when further deformations are required.

Categories and Subject Descriptors: I.3.5 [**Computational Geometry and Object Modeling**]: Curve, Surface, solid and object representation; Modeling packages. J.6 [**Computer- Aided Engineering**]: Computer-Aided Design (CAD).

1. Introduction

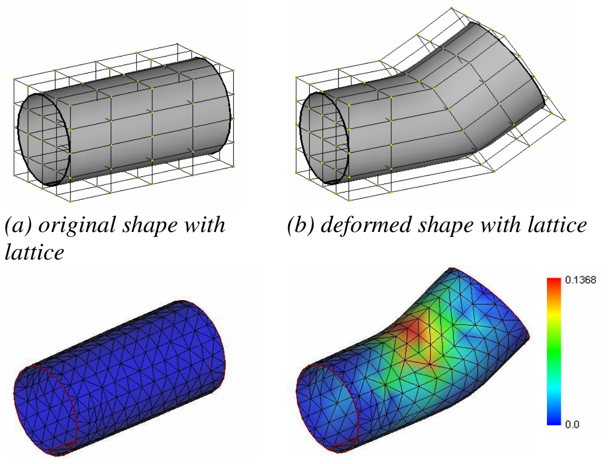
The design of complex models is a key problem in geometric modeling. One of the most powerful tools for deforming a complex model is free-form deformation (FFD) first introduced by Sederberg and Parry [SP86]. The idea of the FFD method is to deform a complex object by deforming a simple flexible solid (called *embedding volume* or *solid*) in which the given object is embedded so that the deformation of the simple solid is propagated to the embedded object. This tool is very useful for designing product shapes, especially when the topology of the product is required to be retained. Usually such an FFD is *constrained*: a set of points on the original embedded object are specified and their new locations in space are given; the embedding volume is deformed in such a way that under the parametric mapping of the deformed embedding volume the specified set of points take the given new locations.

After Sederberg and Parry [SP86] firstly introduced the free-form deformation (FFD) method, many variants have been reported. The original FFD method adopts a trivariate parallelepiped Bézier volume as the embedding solid. In [Coq90, CR94], the control lattices were extended to non-parallelepiped; in [MJ96], a further improvement was proposed to support lattices with arbitrary topology. Trivariate B-Splines or NURBS solids are the most popular types of embedding volume [GP89, LW94]. To overcome the difficulty in getting the deformed object pass through desired points precisely, the direct manipulation FFD methods were proposed. Hsu et al. [HHK92] adopted least-square fitting approach to determine the movement of the control points, and the method of Hu et al. [HZT01] is based on a constrained optimization scheme. As some specified points are required to be on the object after deformation, the methods in [HHK92, HZT01] are constrained FFD approaches. The algorithm presented in [HML00] is another kind of constrained FFD, where the total volume of a solid undergoing FFD is preserved.

However, under an FFD many geometrical surface properties of the embedded object will have some change, very often undesirable. Among these properties one particularly important one is the *developability* of a surface. Informally, a surface is developable if it can be flattened into a plane without any distortion [Car76]. This is a highly desirable property in sheet manufacturing industry, where the stretch or compression in the sheet material is not wanted, as they make the product more prone to damage since internal strains and stresses are generated. As an example shown in Figure 1a, the original design of the duct has a developable shape which can thus be rolled by a metal sheet. After redesigning it by FFD, its shape becomes the one as shown in Figure 1b which is non-developable. The elastic energy maps of the duct surface before and after the FFD are given in Figure 1c and Figure 1d respectively. As clearly shown, a great amount of elastic energy is generated because of the FFD which will translate into stretch and compression if the newly designed duct is to be manufactured by metal sheet. Manufacturers thus require eliminating this kind of stretch and compression at the design stage. This requirement exists in many applications (e.g., clothing, ship hulls, ducts, shoes, aircraft and automobile parts). This leads to the concept of *developability-preserved FFD* which is the theme of this paper.

No prior research on developability-preserved FFD has been found in literature. In our approach, the original given surface to be deformed is developable and has been tessellated into triangular mesh surface patch. The developability of a surface relates to every point on it, by this simplification, we only need to consider about the developability at the mesh vertices – the triangles linking them are of course developable. The mesh patches are assembled together by sharing some triangular edges. After analyzing the developability property of a tessellated surface, the developability-preserved FFD problem is formulated as a constrained optimization problem, where the parameters of each triangular node inside the embedding volume are chosen as variables to be optimized.

Different from other constrained FFD approaches [HHK92; HML00; HZT01], in our approach the positions of the lattice control points of the embedding volume are *not* modified. This is because changing the positions of the lattice control points can only lead to coarse-scale modification; however, the developability preservation needs fine-scale modification of the deformed surface. Furthermore, we do *not* directly optimize the positions of triangular nodes either, based on the reasoning that usually successive deformations are performed and they require inversely computing the corresponding parameters of the relocated nodes, which is extremely tedious and very time-consuming. The constrained optimization problem is numerically solved by a penalty function based scheme, and the optimization scheme is further enhanced to maintain the continuity between the assembled patches. A NURBS solid is adopted as the embedding volume function in our implementation; however, our method can be easily extended to any lattice-based free-form deformation functions.



(a) original shape with lattice *(b) deformed shape with lattice*

(c) elastic energy map of the original duct *(d) elastic energy map of the newly shaped duct*

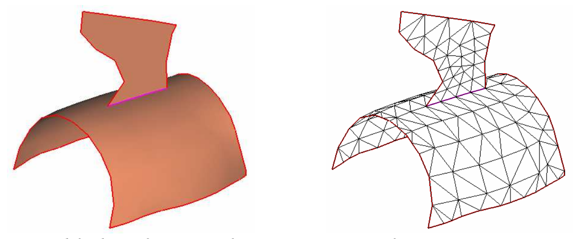
Figure 1: *example I – a deformed duct leads to stretch in manufacturing*

2. Mathematical Formulation

In this section, the mathematical formulation of the developability-preserved free-form deformation for assembled tessellated surface patches is presented.

2.1. Deformation volume

Deforming a shape via embedding it into a (usually uniform) volume such as a NURBS volume and then modifying its control points is a popular approach. Here, we utilize a NURBS solid $Q(u,v,w)$ with a control net of uniformly distributed control points $P_{i,j,k}$ as the deformation volume. The object to be deformed is denoted by O which is a set of assembled surface patches M_i. Each surface patch M_i is a piecewise linear triangular mesh either provided by the user or generated using a standard tessellation algorithm, and the surface patches are assembled together by sharing some common triangular edges (as shown in Figure 2). All patches of the object O are embedded in $Q(u,v,w)$ by determining the parameters (u,v,w) of every triangular node on O with reference to $Q(u,v,w)$. Then, after adjusting the control points, weights, or knot vectors of the NURBS volume $Q(u,v,w)$, and substituting the parameters (u,v,w) of every point on O into the adjusted $Q(u,v,w)$, the object O is deformed. The parameters (u,v,w) can be quickly derived through simple linear transformation if the contribution of the control points is uniform and paralleled.



assembled surface patches *mesh representation*

Figure 2: *Assembled surface patches after tessellation*

2.2. Developability of a tessellated surface patch

Let us first recall the following theorem in differential geometry [Car96].

Theorem 2-1 At regular points, the Gaussian curvature of a developable surface is identically *zero*.

By this theorem, one can easily detect whether a surface is developable according to its overall Gaussian curvature. However, Gaussian curvature is not well defined mathematically on a piecewise linear polygonal mesh surface. An extension of theorem 2-1 is required to detect whether a piecewise linear polygonal surface patch is developable. Thus, the following proposition arises for this purpose.

Proposition 2-1 At any internal point of a developable piecewise linear surface, the summed inner angle is identically 2π.

Proof. For a point q_i on a developable triangular mesh surface patch M, if θ_j is an inner angle adjacent to q_i before flattening and θ_j^F is the corresponding inner angle flattened on the 2D plane, as illustrated in Figure 3, the inner angles satisfy $\theta_j = \theta_j^F$ if the surface at this point can be flattened without stretching. In the 2D plane, $\sum_j \theta_j^F$ equals 2π for an internal vertex. When M is developable, which makes $\theta_j = \theta_j^F$ at every point on M, we have $\sum_j \theta_j = 2\pi$.

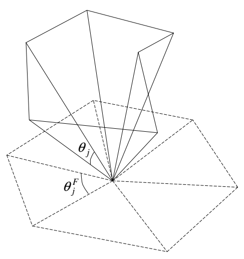


Figure 3: *The inner angles before and after flattening the triangles around a vertex*

The approximation Gaussian curvature formula in [KBB00] on an internal triangular node q_i is

$$\kappa_{q_i} = \frac{2\pi - \sum_j \theta_j}{\frac{1}{3}\sum_j A_j}, \tag{2-1}$$

where θ_j are the inner angles adjacent to q_i, and A_j are the corresponding triangle areas. When utilizing the above approximation of Gaussian curvature to detect the developability of the given surface patch M, by Theorem 2-1, we have $\kappa_{q_i} = 0$, which also leads to $\sum_j \theta_j = 2\pi$. Q.E.D.

For an internal vertex, we call it a *developable point* when $\sum_j \theta_j = 2\pi$ is satisfied at this point; otherwise, it is called a *non-developable point*. Using Proposition 2-1, we can detect whether a given mesh patch M is developable by checking every internal vertex. However, simply stating whether a surface is developable or not is insufficient to identify the degree of developability of the surface. Thus, we define the *developability function* on a tessellated surface to describe it.

Definition 2-1 The developability function of a tessellated surface patch M is defined as

$$D[M] = \frac{1}{A}\sum_i \delta(2\pi - \theta_{sum}(q_i))A_{q_i}$$

where $\delta(t)$ is the impulse function, $A_{q_i} = \frac{1}{3}\sum_j A_j$ is the sum area of the adjacent triangle of a vertex q_i on M, and A is the area of M. $\theta_{sum}(q_i)$ is either the sum of inner angles adjacent to q_i when q_i is an internal vertex, or set to 2π when q_i is on the boundary of M.

The value of the developability function gives a progressive estimate of the developable property of a surface patch. When $D[M]$=1, all internal vertices on this surface are developable points; in other words, M is developable. When $D[M]$=0, it means that we cannot find any developable point on the surface – M is absolutely non-developable. For any $D[M] \in (0,1)$, there are some developable points on M. The larger the value of $D[M]$, the more developable the surface M is.

2.3. Problem definition

For a given surface M with n vertices and $D[M]$=1, the parameters of the n vertices form a $n \times 3$ matrix

$$X = \begin{bmatrix} u_0 \; u_1 & u_{n-1} \\ v_0 \; v_1 \cdots & v_{n-1} \\ w_0 \; w_1 & w_{n-1} \end{bmatrix}^T$$

when it is embedded in the control lattice Q. After M is deformed into a new M' when Q is changed to Q', the developability function value $D[M']$ usually decreases if the same X is kept. For an embedded object with multiple patches, the parameters of all vertices on the object compose the final X. The problem we are to solve here is to find a new matrix X^* so that an optimized M^* is determined by Q' and this X^*. The M^* should be developable (i.e., $D[M^*]=1$), and the difference between M^* and M' should be minimized.

The reason we do not contemplate computing an optimized M^* by directly changing the positions of control points in Q' is that such a modification is always of coarse-scale nature. The density of control points on the FFD lattice is usually much less than that of the triangular mesh vertices. Thus, by the NURBS properties of the deformation volume, when the position of a control point in Q' is adjusted, it changes the positions of more than one mesh vertices on M'. However, the developability-preserved optimization needs fine-scale modification of M' to eliminate some non-developable points locally. When the parameters of one mesh vertex q_i are adjusted, only the position of this vertex is modified – there is no effect on the developability of other vertices not adjacent to q_i. A fine-scale modification is thus achieved.

Another question to answer is why not directly changing the positions of vertices on M' to get M^*? When a new deformation is required after the positions of the vertices on M' are altered, we need to compute the new parameters of these vertices in a non-parallelepiped control lattice. This inverse calculation is extremely difficult and very time-consuming. Furthermore, trying to optimize the positions of vertices on M' may lead some vertices moving outside the deformation volume. In such a case, no further deformation can be expected.

When determining the optimized X^* of M^* to achieve $D[M^*]=1$, we attempt to minimize the surface discrepancy between M' and M^* since the shape of M' is what the designer wants. An elastic energy $E(X^*)$ is defined below to quantify this change,

$$E(X^*) = \sum_j \left(\left\| q_{j,s}(X^*) q_{j,e}(X^*) \right\| - l_j^0 \right)^2 \tag{2-2}$$

where j is the index of a triangular edge, $q_{j,s} \in M$ and $q_{j,e} \in M$ are the vertices of the edge, and l_j^0 is the length of the triangular edge j on M'. This energy function simulates a spring network in which every spring follows along a triangular edge on M^*. The energy measures the change of length on every triangular edge between M^* and M'. Our goal is to find a new configuration X^*, which preserves the developability of the embedded surface patch while at the same time minimizes the incurred discrepancy between M^* and M'. Therefore, we formulate the problem as a *constrained optimization problem*, where we search for the minimum energy configuration of X subject to the constraint of developability preservation:

$$\textbf{\textit{min}}\ E(X^*)\ \textbf{subject to}\ D[M^*]=1. \tag{2-3}$$

In the definition of the developability function, there is an impulse function which may lead to irregularity during the optimization. Here, we define a new *developability detect function* $G[\cdots]$ to take place of the developability function $D[\cdots]$ as

$$G[X] = \sum_i (g(q_i(X)))^2 \tag{2-4}$$

where $q_i(X)$ is the position of a triangular vertex $q_i \in M$ determined by the parameter configuration X, and the function $g(q_i)$ is a *vertex developability detect function* given as

$$g(q_i) = \begin{cases} 2\pi - \sum_k \theta_k & (q_i \notin \mathrm{B}) \\ 0 & (q_i \in \mathrm{B}) \end{cases} \tag{2-5}$$

where B is the set of triangular vertices on the boundary of the given mesh patch M. It is not hard to verify that when $G[X^*]=0$, the sum of the inner angles at every internal vertex equals 2π, hence $D[M^*]=1$ is satisfied. Thus, we replace the developability constraint by this new one and the constrained optimization problem is redefined as

$$\textit{min}\ E(X^*)\ \textbf{subject to}\ G[X^*]=0\,. \qquad (2\text{-}6)$$

It is important to state that the optimization formulation of (2-6) pertains to a single patch M_i on the embedded object O. Since O is usually made of several surface patches assembled together, the continuity constraint should also be added when these patches are optimized individually. This will be discussed in the following section.

3. Numerical Scheme

Recall our constrained optimization problem:

$$\textit{min}\ E(X^*)\ \textbf{subject to}\ G[X^*]=0\,,$$

we can convert it into an unconstrained optimization problem by adding the constraint as a penalty term on the objective function [MS92]. As a result, the objective function to be optimized becomes

$$J(X^*)=E(X^*)+\frac{\rho}{2}(G(X^*))^2 \qquad (3\text{-}1)$$

where ρ is the coefficient to balance the weight between $E(X^*)$ and $G(X^*)$. The choice of ρ is by no means trivial; for smaller ρ, the computing procedure converges slowly to $G[X^*]=0$; for larger ρ, the shape of the surface patch after optimization usually deviates too much from the one before optimization. For any starting optimization point X^0, the procedure begins to minimize $J(X^0)$ with $\rho=\frac{1}{n_e(G[X^0])^2}\sum_j(l_j^0)^2$, where n_e is the number of triangular edges. After applying the conjugate gradient method to minimize the value of $J(X)$ with a fixed number of iteration steps (which is empirical and is 5 in our implementation), we obtain a new point X^1. Then, we use X^1 as a starting guess for the minimum of $J(X)$ with $\rho=\frac{1}{n_e(G[X^1])^2}\sum_j(l_j^0)^2$ and obtain X^2, and so on. In actual computation, we stop the process either when the constraint violation is less than a given threshold or when changes in $J(X)$ become insignificant. Theoretically, we arrive at X^* in the limit as ρ tends to infinity.

In the object O consisting of assembled mesh patches M_i ($i=1,\cdots,m$), a vertex shared by more than one patches is called an *assembling vertex*. Associated with an assembling vertex q_p, we define a *linked vertex set* L_{q_p} which contains all the mesh vertices in O coincidental at q_p; also, for any vertex $q_q\in L_{q_p}$, there is an associated linked vertex set L_{q_q} where we have $q_p\in L_{q_q}$. The cardinality of the linked vertex set of a vertex is exactly the number of patches sharing the vertex. By means of these linked vertex sets, the connectivity information of assembled patches is stored. However, this connectivity is ignored when the shape of every $M_i\in O$ is optimized individually – for two coincidental triangular nodes belonging to two different patches, their (u, v, w) parameters are adjusted independently since the parameters are in different rows in X; consequently, cracks will appear at places where two patches originally met. For example, in Figure 4, the object with assembled patches is deformed from the shape in (a) to the one in (b). (The color map on the surface indicates the value of vertex developability detection function $g(\cdots)$ at each triangular node – called *developability detection map*; a linear interpolation is utilized to compute the values at non-vertex points on the surface. The blue color indicates full developability while other colors symbolizes non-developability in different degrees.) After applying the conjugate gradient method to determine the optimized X, a crack appears on the optimized shape, as shown in Figure 4c; an enlarged mesh about this crack is given in Figure 4d.

The numerical scheme then needs to be enhanced to take into consideration of preserving the position continuity of O. The basic idea is to make the linked vertices consistent during the optimization. To achieve this consistency, the formulas of computing gradients at the assembling vertices are modified. When changing the position of an assembling vertex q_a, the positions of vertices in L_{q_a} should be maintained the same as q_a. Thus, the gradient of E with respect to q_a relates to not only $\sum\left(\|q_aq_j\|-l_{aj}^0\right)^2$ but also all the other terms $\sum\left(\|q_pq_q\|-l_{pq}^0\right)^2$ ($q_q\in L_{q_a}$) in E. Also, the numerical gradient of G with respect to q_a should be changed to

$$\frac{\partial G}{\partial q_a}=\frac{G_{PA}(q_a+h)-G_{PA}(q_a-h)}{2h}$$

instead of $\frac{\partial G}{\partial q_a}=\frac{(g(q_a+h))^2-(g(q_a-h))^2}{2h}$, where

$$G_{PA}(q_a)=(g(q_a))^2+\sum_q(g(q_q))^2+\sum_j(g(q_j))^2$$

with q_j being either the adjacent vertices of q_a or the adjacent vertices of q_q ($q_q\in L_{q_a}$).

When calculated with the above prescribed method, the gradients of the linked vertices become consistent with each other. Therefore, while searching the optimum along the conjugate direction, the update of their positions is also consistent, which in turn then ensures the position continuity (e.g., the result shown in Figure 4e and 4f).

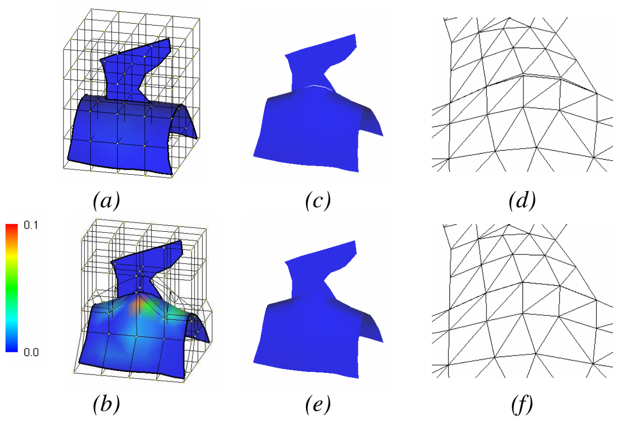


Figure 4: *Example II – optimization without vs. with continuity preservation: (a) before FFD; (b) after FFD; (c) result without preserving continuity; (d) mesh view – crack occurs; (e) result with preserving continuity; (f) mesh view – continuity preserved.*

So far, the prescribed optimization process is applied indiscriminately to the parameters of all vertices on the given object O. However, it is intuitive to conjecture that, as also observed in some of our experiments, usually only a subset of the vertices need to be optimized. For example, in the object shown in Figure 4, after free-form deformation, the area with blue color (which indicates full developability) takes more than half of the object's total area. By this observation, we could improve the algorithm as follows: the parameters of a vertex q_i are inserted into X only if its $g(q_i) > \lambda$ or it is adjacent to a vertex whose developability detection function $g(\cdots)$ returns a value greater than λ. In all of our testing examples in this paper the λ is set to 10^{-8}.

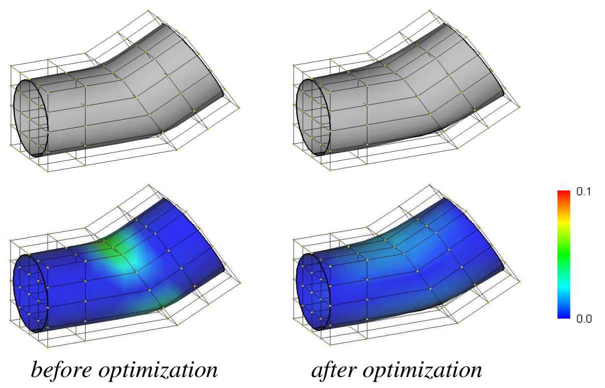


Figure 5: *Example I – a duct: using our new approach, the developability of resultantant surface is preserved.*

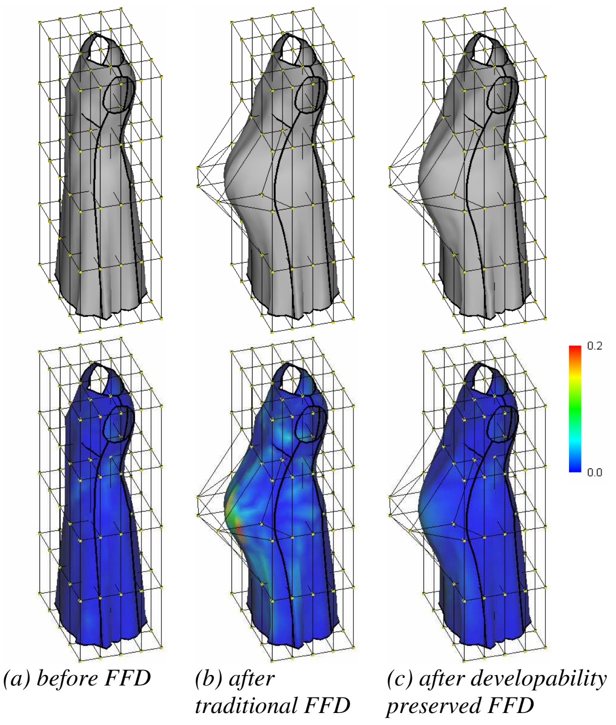

0.2
0.0
(a) before FFD
(b) after traditional FFD
(c) after developability preserved FFD

Figure 6: *Example III – the style modification of lady dress: the developability is preserved in our FFD.*

4. Experimental Result

We have implemented the proposed optimization algorithm and tested it on a number of examples of which four are given here. Figure 5 shows the optimization result of example I. In example III and IV, real samples from apparel industry are used. Garment pattern assembly and cloth simulation techniques have been studied for a long time [VCM95, CK02] and recently people have been exploring the use of the free-form deformation technique in style design of a cloth model with assembled 3D patterns. However, as aforementioned at the beginning of this paper, the FFD technique applied here ought to be constrained to preserve the developability of the assembled surface patches. Figure 6 – 7 shows the experimental results of our approach in this regard. The computational statistics is listed in Table 1.

During the iteration of optimization, the value of constraint function decreases while the number of iteration increases. Here, these two factors are utilized together to give the terminal criterion of iterations. Thus, our terminal condition is either $\left\| G[X^i] - G[X^{i-1}] \right\| / G[X^0] < \varepsilon$ or the iteration number is greater than $N_{\max}$, where $G[X^i]$ is the value of the constraint function in the ith iteration (current value), $G[X^0]$ is the value of the constraint function before optimization, $N_{\max}$ is the maximum iteration number, and ε is a small number (we choose $\varepsilon = 0.01\%$ in all of our testing examples).

Table 1 *Computation statistics of the examples*

Example	Developability		Node No.	Time Cost	Total Steps
	$G[X^0]$	$G[X^*]$			
I (Fig.1, 5)	0.0541	0.0129	328	21s	200
II (Fig.2, 4)	0.0257	0.0005	149	5s	175
III (Fig. 6)	0.4778	0.0396	1170	104s	200
IV (Fig. 7)	0.1930	0.0139	1494	135s	200

*All with $N_{\max}$=200 on a PIII 900 PC with a program in C++.

5. Conclusion and Discussion

In this paper, we consider the problem of preserving the developability of a surface under FFD and propose a practical solution to it. In our approach, the developability-preserved FFD problem is formulated as a constrained optimization problem, where the parameters of each node on the mesh of the embedded surface are chosen as variables for optimization. Optimizing by modifying the parameters rather than the position of each mesh node benefits the successive deformation – the time-consuming process of determining the parameters of the nodes in a non-parallelepiped control lattice is avoided. The popular NURBS solid is adopted as the deformation function in our implementation; however, our method can be easily generalized to any variants of lattice-based free-form deformation.

Since our main objective is to model the developability preservation problem in the FFD as a functional optimization problem, the adoption/development of the exact numerical solution for solving such an optimization is rather pedagogical and a very simple one is used in our current implementation. But even using this very primitive numerical method, the speed of our developability preserved free-form deformation has already been seen to be acceptable (see Table 1), at least for design purpose. It is believed that with more efficient and elaborate numerical optimization algorithms (e.g., projected polyhedron algorithm [PM02] or GPD-based optimization approaches [BFG03]) and with the increasing processing power available on the desktop, the running time can be further shortened. Also the multi-level optimization can be considered. Transforming a surface to a developable one will destroy some desirable geometric characteristics of the surface (e.g., smoothness). This work could be taken into account in our future reseach.

References

[BFG03] Bolz J., Farmer I., Grinspun E., Schröder P.: Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid, *SIGGRAPH 2003*.

[Car76] do Carmo M.P.: *Differential Geometry of Curves and Surfaces*, Englewood Cliffs, N.J.: Prentice-Hall, 1976.

[CK02] Choi K.J., Ko H.S.: Stable but resposive cloth, *SIGGRAPH 2002 Conference Proceedings*, pp.604–611, 2002.

[Coq90] Coquillart S.: Extended free-form deformations: A sculpting tool for 3D geometric modeling, *Computer Graphics*, 24(4): 187-196, 1990.

[CR94] Chang Y.K., Rockwood A.P.: A generalized de Casteljau approach to 3D free-form deformation, *Computer Graphics*, 28(4): 257-260, 1994

[GP89] Griessmair J., Purgathofer W.: Deformation of solids with trivariate B-splines, *EUROGRAPHICS '89*, pp.137-48, Amsterdam, Netherlands, 1989.

[HHK92] Hsu W., Hughes J., Kaufmann H.: Direct manipulations of free-form deformations, *Computer Graphics*, vol.26, no.2, pp.177-184, 1992.

[HML00] Hirota G., Maheshwari R., Lin M.C.: Fast volume-preserving free-form deformation using multi-level optimization, *Computer-Aided Design*, 32(8-9): 499-512, 2000.

[HZT01] Hu S.M., Zhang H., Tai C.L., Sun J.G.: Direct manipulation of FFD: efficient explicit solutions and decomposible multiple point constraints, *The Visual Computer*, 17(6): 370-379, 2001.

[KBB00] Kobbelt L.P., Bischoff S., Botsch M., Kähler K., Rössl C., Schneider R., Vorsatz J.: Geometric modeling based on polygonal meshes, *EUROGRAPHICS 2000 Tutorial.*

[LW94] Lamousin H.J., Waggenspack W.N.: NURBS-based free-form deformation, *IEEE Computer Graphics and Application*, 14(6): 59-65, 1994.

[MJ96] MacCracken R., Joy K.: Free-form deformations with lattices of arbitrary topology, *Computer Graphics*, 30(4): 181-189, 1996.

[MS92] Moreton H.P., Sequin C.H.: Functional optimization for fair surface design, *Computer Graphics*, vol.26, no.2, July 1992, pp.167-76.

[PM02] Patrikalakis N.M., Maekawa T.: *Shape interrogation for computer aided design and manufacturing*, Berlin; New York: Springer, 2002.

[SP86] Sederberg T., Parry S.: Free-form deformations of solid geometric models, *Computer Graphics*, vol.20, pp.151-160, 1986.

[VCM95] Volino P., Courchesne M., Magnenat-Thalmann N.: Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects, *SIGGRAPH'95 proceedings*, pp.137-144, 1995.

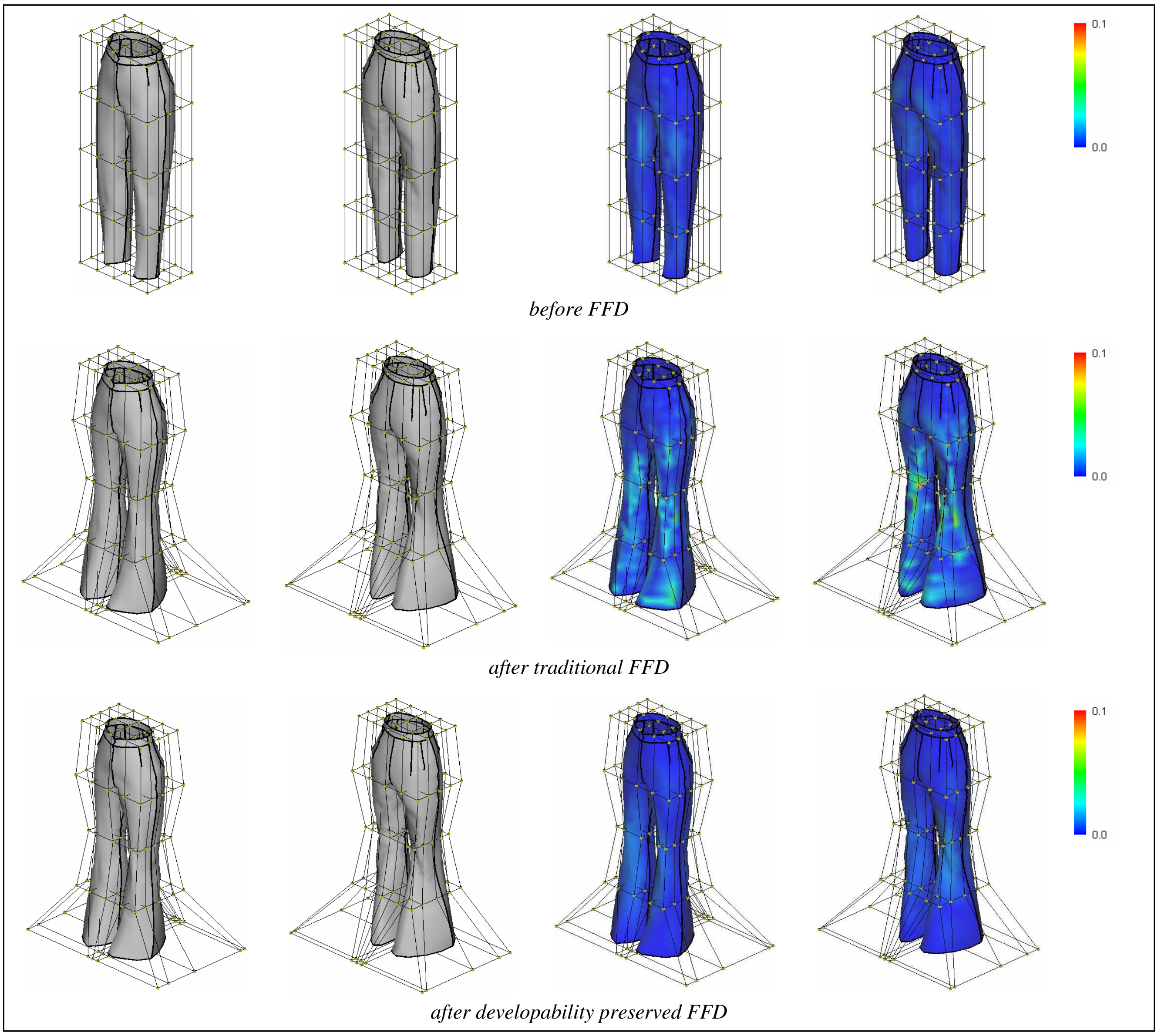


Figure 7: *Example IV – the style modification of lady pants: using our new approach, the developability of resultant surface is preserved comparing to the traditional FFD approach.*