Available online at www.sciencedirect.com
ScienceDirect
Computer-Aided Design 39 (2007) 719–731
COMPUTER-AIDED DESIGN
www.elsevier.com/locate/cad

Modeling dynamic developable meshes by the Hamilton principle

Yong-Jin Liu[a,*], Kai Tang[b], Ajay Joneja[c]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing, PR China*
[b] *Department of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong, China*
[c] *Department of Industrial Engineering and Logistics Management, Hong Kong University of Science and Technology, Hong Kong, China*

Received 28 April 2006; accepted 27 February 2007

Abstract

In this paper, a new dynamic developable surface model is proposed. The proposed model represents developable surfaces using triangle meshes. A novel algorithm is proposed to introduce the Hamilton principle into these meshes such that the resulting developable model is dynamic, i.e., it can offer a time-dependent continuous path to deform the model. Applications with examples are presented; these show that the proposed technique can model buckled developable surfaces well, and can offer physically-realistic animations of deformed developable surfaces.

Keywords: Developable surface; Hamilton principle; Physical-based simulation

1. Introduction

A developable surface is a surface that can be flattened into a plane without stretching or tearing. Developable surfaces have found rich applications in industries such as manufacturing sheet-metal and plate-metal for aircraft skins, ship hulls and automobiles; and bending paper, leather, plywood and fabric over a variety of products, including furniture, boxes and other containers.

Because of their wide use in industry, developable surfaces have received considerable attention in computer aided geometric design. Most of existing work exploits the characterizations and properties for free-form surfaces to be developable. Notably two classes of approaches exist: the first uses the primal representation of surfaces [2,3,19,7], and the second uses the dual representation of surfaces [6,15,22]. The primal representation uses a tensor product surface of degree $(1, n)$, and usually solves nonlinear characterizing equations to guarantee developability. A developable surface can also be viewed as the envelope of a one-parameter family of tangent planes, and thus can be treated as a curve in a dual projective 3-space. Given this dual representation, some interpolation and approximation algorithms can be computed efficiently. However, interactively designing a developable surface in the dual projective space is not intuitive, and it could be difficult to handle the singularities and points at infinity in projective space.

All the above discussed developable surfaces are *smooth*, i.e., they have no sharp creases, or only have singularities along the line of striction. Frey [13] discusses an important class of developable surfaces that are *buckled*, i.e., the surface exhibits creases of finite length or localized singular points. For an example, when folding thin-sheet material such as paper into boxes or other similar containers along straight-line creases, the resulting surface is developable, but not smooth across the crease lines. Modeling crumpled paper is well known to be a difficult problem [18,1]. As a practical and efficient solution, Frey [12,13] shows how to approximate buckled surfaces by developable three-dimensional triangulation and discusses some interesting insights.

In this paper, we present an algorithm that also uses triangle meshes to model developable surface. Based on triangle meshes, we make a step further by introducing the Hamilton principle into the model such that the resulting developable mesh is *dynamic*. Besides its novelty in algorithm design, we believe that the perspective offered in this paper, which clearly departs from most existing work, can offer distinct advantages in computer aided design. For example, the paper model used

* Corresponding author. Tel.: +86 10 62780807.
E-mail addresses: liuyongjin@tsinghua.edu.cn (Y.-J. Liu), mektang@ust.hk (K. Tang), joneja@ust.hk (A. Joneja).

0010-4485/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2007.02.013

in origami is clearly developable. However, due to shape complexity, it is not clear from a designer's point of view how a paper model can be folded by a single square planar paper. Computational origami is a new branch of computer science [8], and recently the field has grown significantly [9]. Dynamic developable meshes shed light on a potential solution to these fascinating puzzles. Dynamic developable meshes can also find applications in printer design by the computer simulation of a paper jam process. Detailed applications of our dynamic developable model are presented in Section 6.

2. Dynamic developable model

Our proposed dynamic developable model is a triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with the sets of vertices $\mathcal{V}$, edges $\mathcal{E}$ and faces $\mathcal{F}$, which is treated as a *rigid multi-body system*: each vertex $v_i \in \mathcal{V}$ is associated with a mass m_i, and each edge $e_i \in \mathcal{E}$ is treated as a weightless rigid rod; thus, each triangular face $f_i \in \mathcal{F}$ can only be moved rigidly in order to offer the desired developability to the surface model $\mathcal{M}$.

Consider a piece of surface $\mathcal{M}$ consisting of $n_\mathcal{V}$ particles, subject to given geometric constraints and otherwise influenced by forces which are functions only of the positions of the particles. Let the force acting upon the ith particle at (x_i, y_i, z_i) of the system have the Cartesian components (f_x^i, f_y^i, f_z^i). Here we consider the type of force system – conservative system – for which there exists a single scalar function $V = V(x_1, y_1, z_1, \ldots, x_{n_\mathcal{V}}, y_{n_\mathcal{V}}, z_{n_\mathcal{V}})$ from which the $3n_\mathcal{V}$ force components can be derived as

$$f_x^i = -\frac{\partial V}{\partial x_i}, \qquad f_y^i = -\frac{\partial V}{\partial y_i}, \qquad f_z^i = -\frac{\partial V}{\partial z_i}, \quad i = 1, 2, \ldots, n_\mathcal{V}. \tag{1}$$

The function V is called the *potential energy* of the system. The nonconservative system is discussed in Section 5.

The *kinetic energy* of a particle v_i is defined as $\frac{1}{2}m_i(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2)$.[1] Given a system of $n_\mathcal{V}$ particles, its kinetic energy is

$$T = \frac{1}{2}\sum_{i=1}^{n_\mathcal{V}} m_i(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2). \tag{2}$$

The geometric constraints upon a mesh model $\mathcal{M}$ are partitioned into two classes: external and internal constraints. The *external* geometric constraints consist in the confinement of certain particles to given positions, curves or surfaces. Since the proposed model is a triangulated particle system, the *internal* constraints maintain the constancy of the length of each mesh edge, which are treated as weightless rigid rods. The internal constraints are completely specified by the $n_\mathcal{E}$ consistent and independent equations

$$\|v_j^0 - v_j^1\|^2_{e_j=(v_j^0, v_j^1)} = l_j, \quad j = 1, 2, \ldots, n_\mathcal{E} \tag{3}$$

where $l_j > 0$ is the square of constant length of edge e_j.

2.1. System coordinates and Hamilton principle

Subject to constraints (3), the particle representation of the developable surface $\mathcal{M}$ is a holonomic system with $(3n_\mathcal{V} - n_\mathcal{E})$ degrees of freedom (DOFs). It is more convenient to introduce a set of $n = 3n_\mathcal{V} - n_\mathcal{E}$ independent variables $q_1, q_2, \ldots, q_n$ through which the positions of all $n_\mathcal{V}$ particles are fully and uniquely determined. That is, the equations of constraints (3) can be in effect replaced by an equivalent system of $3n_\mathcal{V}$ equations

$$\begin{cases} x_i = x_i(q_1, q_2, \ldots, q_n), \\ y_i = y_i(q_1, q_2, \ldots, q_n), \\ z_i = z_i(q_1, q_2, \ldots, q_n), \end{cases} \quad i = 1, 2, \ldots, n_\mathcal{V}. \tag{4}$$

The variables $q_1, q_2, \ldots, q_n$ are called *system coordinates*,[2] where x_i, y_i, z_i are *absolute coordinates*. Using the system (4), the potential energy V can be expressed solely in terms of the system coordinates $V = V(q_1, q_2, \ldots, q_n)$. To express the kinetic energy (2) in terms of the system coordinates, we differentiate each of the equations (4) w.r.t. time:

$$\dot{x}_i = \sum_{k=1}^{n} \frac{\partial x_i}{\partial q_k}\dot{q}_k, \qquad \dot{y}_i = \sum_{k=1}^{n} \frac{\partial y_i}{\partial q_k}\dot{q}_k, \qquad \dot{z}_i = \sum_{k=1}^{n} \frac{\partial z_i}{\partial q_k}\dot{q}_k. \tag{5}$$

By expressing potential and kinetic energies solely in terms of $q_1, q_2, \ldots, q_n, \dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n$, the *Lagrangian* of the developable model $\mathcal{M}$ is defined as

$$L(q_1, q_2, \ldots, q_n, \dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n) = T - V. \tag{6}$$

The principle of Hamilton [5] reads that *the actual motion of a system with Lagrangian L is such as to render the Hamilton's integral*

$$I = \int_{t_1}^{t_2} L\,\mathrm{d}t \tag{7}$$

an extremum with respect to continuously twice-differentiable functions $q_1(t), q_2(t), \ldots, q_n(t)$, where t_1 and t_2 are two arbitrary instants of time. From Hamilton's principle, the calculus of variations [25] shows that the system coordinates describing the motion of the developable surface $\mathcal{M}$ must satisfy the system of Euler–Lagrange equations

$$\frac{\partial L}{\partial q_i} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = 0, \quad i = 1, 2, \ldots, n. \tag{8}$$

The system (8), known as *Lagrange's equations of motion*, constitutes a set of n second-order ordinary differential equations, whose solution yields the functions $q_1(t), q_2(t), \ldots, q_n(t)$.

2.2. Mesh resolution vs. degrees of freedom

The number n of system coordinates counts for the DOFs of the developable model. Consider a surface $\mathcal{M}$. We can

[1] In this paper, the superior dot is employed to indicate differentiation with respect to the time variable t.

[2] The word *system coordinate* has other names in the engineering literature, including *generalized coordinate* and *Lagrangian coordinate*.

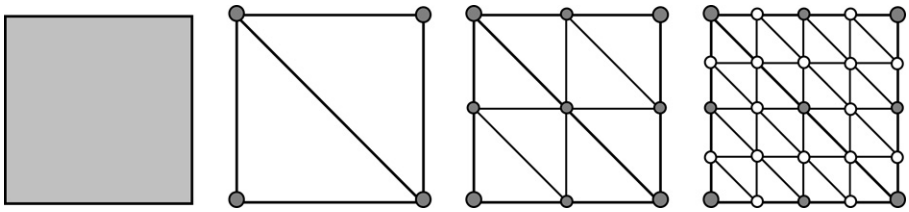


Fig. 1. Mesh refinement with 1–4 splitting rule.

model it by using meshes with different resolutions. For a particular example, we can model surfaces with finer resolution by refining a coarse mesh using the 1–4 splitting rules, as shown in Fig. 1. Intuitively, fine meshes can offer more DOFs than those in coarse meshes. Below, we rigorously affirm this property.

Lemma 1. *Given a developable surface $\mathcal{M}$ modeled by a triangle mesh as a rigid multi-body system, by applying the* 1–4 *splitting rule, at any i (>0)th refinement step, the DOFs of the next level fine mesh $\mathcal{M}^{i+1}$ satisfies the relation*

$$DOF^{i+1} = DOF^i + n^i_{\partial\mathcal{E}}$$

where $n^i_{\partial\mathcal{E}}$ is the number of boundary edges in $\mathcal{M}^i$.

Proof. Consider the surface $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ developing into a plane that has $n_\mathcal{V}$ vertices, $n_\mathcal{E}$ edges and $n_\mathcal{F}$ faces, respectively. Denote its set of boundary edges (each of which is incident to a single face) by $\partial\mathcal{E}$ with cardinality $n_{\partial\mathcal{E}}$. We have $n_{\partial\mathcal{E}} = 2n_\mathcal{E} - 3n_\mathcal{F}$. The Descartes–Euler formula holds

$$n_\mathcal{V} + n'_\mathcal{F} - n_\mathcal{E} = 2 \Rightarrow n_\mathcal{V} + n_\mathcal{F} - n_\mathcal{E} = 1 \tag{9}$$

where $n'_\mathcal{F} = n_\mathcal{F} + 1$ counts the unbounded region in the plane by adding 1. By applying 1–4 splitting rules to a mesh $\mathcal{M}^i$ towards $\mathcal{M}^{i+1}$, we have

$$\begin{cases} n^{i+1}_\mathcal{F} = 4n^i_\mathcal{F} \\ n^{i+1}_\mathcal{V} = n^i_\mathcal{V} + n^i_\mathcal{E} \\ n^{i+1}_\mathcal{E} = 2n^i_\mathcal{E} + 3n^i_\mathcal{F} \end{cases}$$

and

$$\begin{aligned} \mathrm{DOF}^{i+1} &= 3n^{i+1}_\mathcal{V} - n^{i+1}_\mathcal{E} = (3n^i_\mathcal{V} - n^i_\mathcal{E}) + (2n^i_\mathcal{E} - 3n^i_\mathcal{F}) \\ &= \mathrm{DOF}^i + n^i_{\partial\mathcal{E}}. \quad \square \end{aligned}$$

The DOFs of a developable model can be further classified to be either *external* or *internal*. Obviously, the whole model can be rigidly moved in the space $\mathbb{R}^3$; we count these six DOFs as external DOFs. The remaining DOFs of the model are considered as internal DOFs. Refer to Fig. 2. The left model consists of a single triangle and has only six (external) DOFs, i.e., it can only move rigidly in space. The right model has seven DOFs: in addition to the six external DOFs, there is an internal DOF, i.e., the rotating (folding) between the two triangles around their shared edge.

2.3. Mesh pattern vs. folding directions

Consider the mesh refinement pattern shown in Fig. 1. As illustrated in Fig. 3, the resulting meshes can only fold

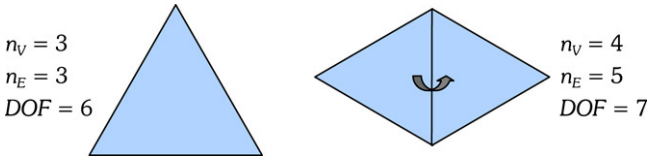


Fig. 2. External vs. internal degrees of freedom.

horizontally, perpendicularly or along the diagonal direction from lower-right to upper-left, but not along the diagonal direction from lower-left to upper-right. We regard this mesh pattern as *anisotropic*. To achieve a uniform distribution of folding directions, we use an *isotropic* mesh refinement pattern, as shown in Fig. 4, for our developable surface model. The following result then holds:

Lemma 2. *Given a developable surface $\mathcal{M}$ modeled by an isotropic triangle mesh, at any i (>0)th refinement step, the DOFs of the next level fine mesh $\mathcal{M}^{i+1}$ satisfy the relation*

$$DOF^{i+1} = 2DOF^i - 3, \quad \textit{with } DOF^0 = 11.$$

Proof. The refinement rules for the isotropic mesh in Fig. 4 can be written as follows. For the upper row of Fig. 4,

$$\begin{cases} n^{i+1}_{Rf} = 4n^i_{Rf}, & \\ n^{i+1}_{\partial Re} = 2n^i_{\partial Re}, & n^0_{\partial Re} = n^0_{Re} \\ n^{i+1}_{IRe} = 2n^i_{IRe} + 4n^i_{Rf}, & n^0_{IRe} = 0 \end{cases}$$

where n_{Rf}, n_{Re} represent the rectangle's faces and edges, respectively, and $n_{\partial Re}, n_{IRe}$ represent the boundary and interior rectangle edges, respectively. For the lower row of Fig. 4,

$$\begin{cases} n^{i+1}_\mathcal{F} = 8n^{i+1}_{Rf} \\ n^{i+1}_\mathcal{E} = 2n^{i+1}_{Re} + 8n^{i+1}_{Rf}. \end{cases}$$

With Eq. (9) and noting that $n_{Re} = n_{\partial Re} + n_{IRe}$, we have:

$$\begin{aligned} \mathrm{DOF}^{i+1} &= 2n^{i+1}_\mathcal{E} - 3n^{i+1}_\mathcal{F} + 3 = 4n^{i+1}_{Re} - 8n^{i+1}_{Rf} + 3 \\ &= 8n^i_{Re} - 16n^i_{Rf} + 3 = 2\mathrm{DOF}^i - 3. \quad \square \end{aligned}$$

2.4. Compatibility condition for the developable model

By enforcing the constraints (3) on the isotropic surface model, each triangle of the mesh can only move rigidly in space. Consider any two triangles that share a common edge e. Refer to Fig. 5. Let $(\alpha(e), \beta(e), \gamma(e))$ be the angle triple associated to one endpoint of e. The superscript "0" is used to indicate the initial (rest) state, and the superscript "t" indicates the state at

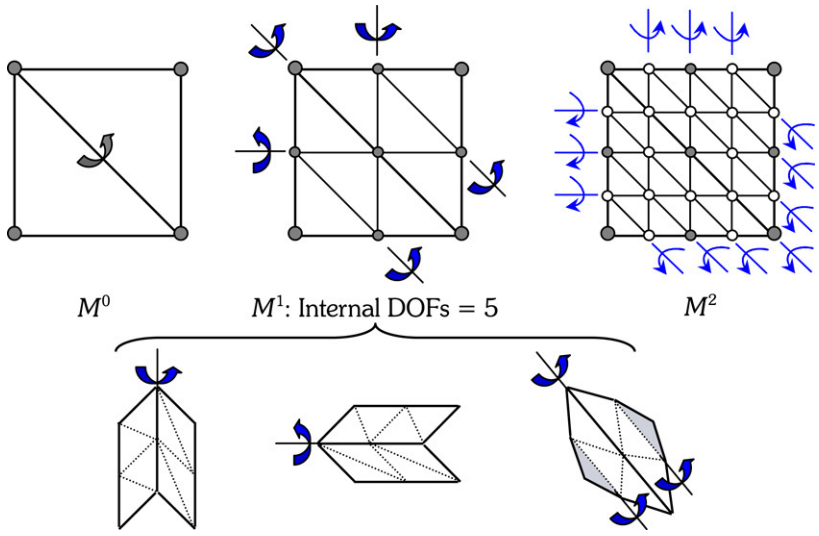


Fig. 3. One of the possible folding directions inherent in the pattern shown in Fig. 1.

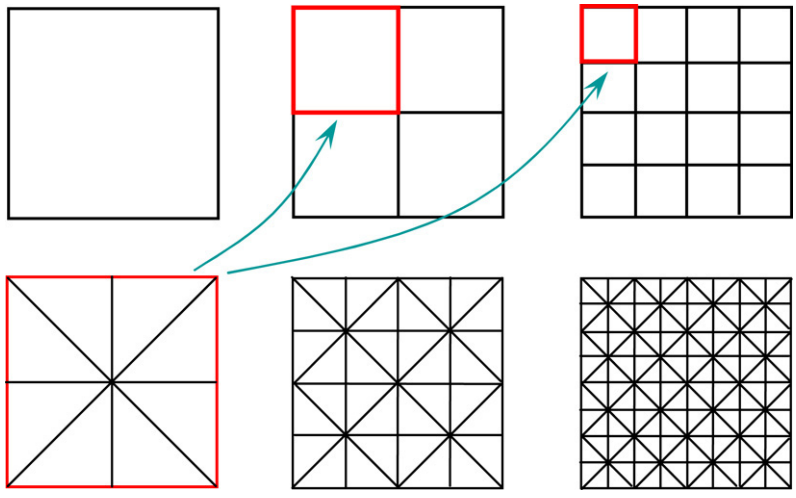


Fig. 4. Isotropic mesh generation and refinement.

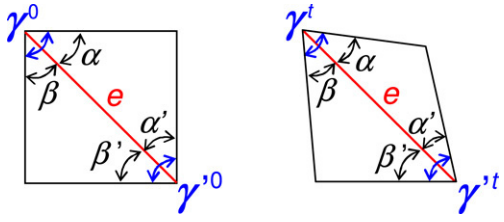


Fig. 5. Compatibility condition of the developable model.

time t. Due to the rigid motion of each triangle, two triangles incident to the edge e can only rotate relatively around e. It is immediately seen that the following geometric constraint must be satisfied:

$$\gamma^t(e) \leq \alpha(e) + \beta(e) = \gamma^0(e). \tag{10}$$

For each internal edge, there are two angle triples corresponding to the two endpoints of that edge; e.g. (α, β, γ) and $(\alpha', \beta', \gamma')$ in Fig. 5. These two triples are equivalent, in the sense that if one triple satisfies relation (10), the other will satisfy (10) automatically.

To present the following theorem, we need the concept of an abstract graph.

Definition. An *abstract graph* is a pair $(\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ is a finite set and $\mathbb{E}$ is a set of unordered pairs of distinct elements of $\mathbb{V}$. Thus, an element of $\mathbb{E}$ is of the form $\{v, w\}$, where $v, w \in \mathbb{V}$ and $v \neq w$. The elements of $\mathbb{V}$ are called *vertices*, and the element $\{v, w\}$ of $\mathbb{E}$ is called the *edge* joining v and w.

Definition. Let $(\mathbb{V}, \mathbb{E})$ be an abstract graph. A *realization* of $(\mathbb{V}, \mathbb{E})$ is a set of points in a real vector space $\mathbb{R}^n$, one point for each vertex, together with straight segments joining precisely those pairs of points which correspond to edges. The points are *vertices* and the segments *edges*; the realization is called a *geometric graph*. It is required that two intersection conditions hold:

(1) Two edges meet, if at all, in a common endpoint;
(2) No vertex lies on an edge except at one of its endpoints.

For more properties about abstract and geometric graphs, the reader is referred to the textbook [14].

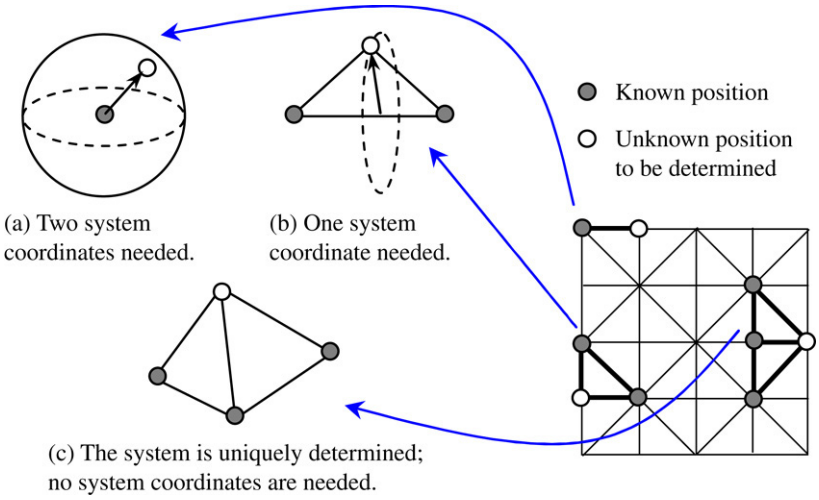


Fig. 6. Basic geometric relations.

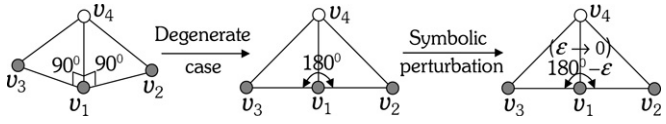


Fig. 7. A degenerate case of the configuration shown in Fig. 6(c).

Let $\mathbb{M}$ be the abstract graph of a developable surface model $\mathcal{M}$. The following result holds:

Theorem 1. *Given a developable surface model $\mathcal{M}$ with a valid initial planar state in $\mathbb{R}^2$, its abstract graph $\mathbb{M}$ has a valid realization $\mathcal{M}(t)$ in $\mathbb{R}^3$ at time t, i.e., satisfying the edge length constraints in Eq. (3), if and only if for every edge e in $\mathbb{M}$, a valid angle triple $(\alpha(e), \beta(e), \gamma(e))$ can be assigned such that the relation (10) is satisfied.*

Proof. That the necessary condition for Theorem 1 holds is readily seen. The sufficient condition is proved in Section 3.2. □

The importance of Theorem 1 is that, if the local compatibility condition by relation (10) holds everywhere, then a valid global realization of the abstract developable model exists.

3. Details of the algorithm

3.1. Basic geometric relations

Consider a valid realization of a developable model which, by Theorem 1, satisfies relation (10) for all internal edges. Refer to Fig. 6:

(a) Given a vertex v with a known position in $\mathbb{R}^3$, it needs two system coordinates, i.e., two spherical coordinates, to locate an incident edge of v (Ref. Fig. 6(a)). Note that the edge length is pre-specified in the model's initial state in $\mathbb{R}^2$;

(b) Given an edge with known positions of both endpoints, it needs one system coordinate to locate an incident triangle (Ref. Fig. 6(b));

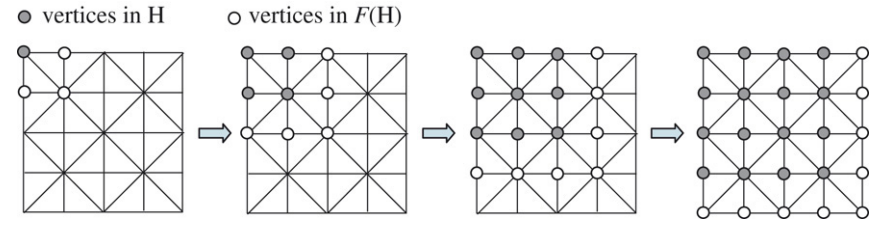
(c) Given two adjacent triangles sharing a common edge, the position of one vertex incident to the common edge is uniquely determined if the positions of the other three vertices are known (Ref. Fig. 6(c)); no system coordinates are needed — this configuration is always valid by Theorem 1, i.e., relation (10) holds. Refer to Fig. 7. A degenerate case of this configuration exists: if the angle γ is exactly π, one system coordinate should be assigned (see also Fig. 15). However, due to floating point computation being used and the dynamic model numerically evolving with a small time step, symbolic perturbation schemes [10] can be applied to remove this kind of degeneracy. Details are presented in [20].

3.2. Proof of the sufficient condition for Theorem 1

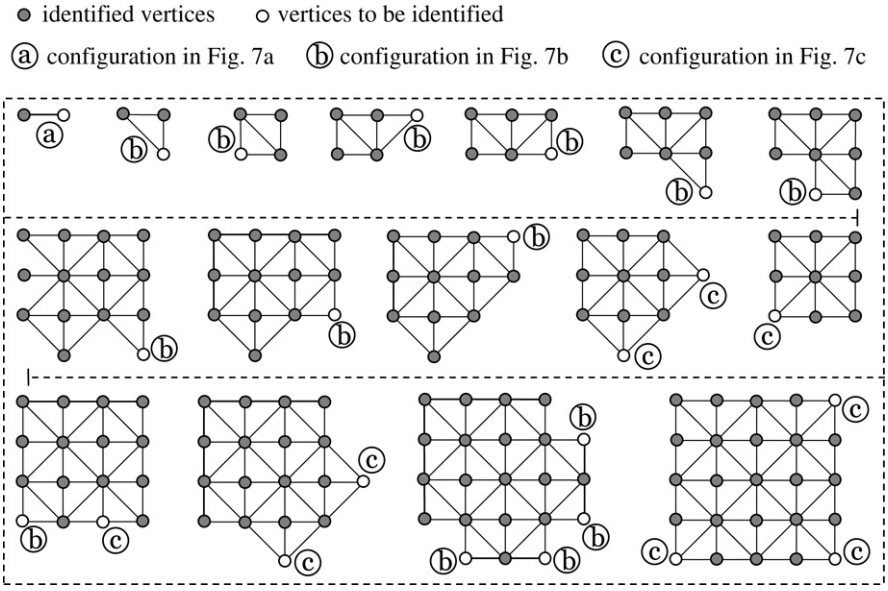
Notation. The abstract developable model can be written as an undirected graph $\mathbb{M} = (\mathbb{V}(\mathbb{M}), \mathbb{E}(\mathbb{M}))$, where $\mathbb{V}$ and $\mathbb{E}$ are finite sets with $\mathbb{E} = \{(v, w) \in \mathbb{V} \times \mathbb{V} : v \neq w\}$. A subgraph of a graph $\mathbb{M}$ is a graph $\mathbb{H} = (\mathbb{V}(\mathbb{H}), \mathbb{E}(\mathbb{H}))$ with $\mathbb{V}(\mathbb{H}) \subseteq \mathbb{V}(\mathbb{M})$ and $\mathbb{E}(\mathbb{H}) \subseteq \mathbb{E}(\mathbb{M})$. The *front* of a subgraph $\mathbb{H}$ is a vertex set $\mathcal{F}(\mathbb{H}) \subseteq \mathbb{V}(\mathbb{M}) \setminus \mathbb{V}(\mathbb{H})$ which satisfies $\forall v \in \mathcal{F}(\mathbb{H})$, there is a vertex $w \in \mathbb{V}(\mathbb{H})$ such that $(v, w) \in \mathbb{E}(\mathbb{M})$.

The strategy we use for this proof is to show that if the condition in Theorem 1 is satisfied, then we can always specify a set of independent system coordinates such that the positions of the particles of the model are uniquely determined. The process of system coordinates assignment is as follows. Without loss of generality, we start at the left-upper corner vertex v_{lu} of the model. Refer to Fig. 8(a). Three system coordinates are needed to specify the location of v_{lu}. Let $\mathbb{H} = \{v_{lu}\}$. The system coordinates assignment takes the following order:

Step 1 Find $\mathcal{F}(\mathbb{H})$;
Step 2 Assign the positions of the front $\mathcal{F}(\mathbb{H})$;
Step 3 $\mathbb{H} = \mathbb{H} \bigcup \mathcal{F}(\mathbb{H})$;
Step 4 If $\mathbb{H} \neq \mathbb{V}(\mathbb{M})$, go to step 1.



(a) Front propagation.



(b) The process of system coordinates assignment.

Fig. 8. The proof of Theorem 1.

To assign system coordinates to a given front $\mathcal{F}(\mathbb{H})$, the step 2 consists of the following substeps (Ref. Fig. 8(b)):

2a Set $\mathcal{F}' = \mathcal{F}(\mathbb{H})$;

2b **while** $\mathcal{F}' \neq \emptyset$, **do**

2ba In $\mathcal{F}'$, identify the configuration shown in Fig. 6(c) to specify the positions of vertices without assigning more system coordinates, remove the identified vertices from $\mathcal{F}'$ and goto step **2b**. If no vertices are identified, goto step **2bb**;

2bb In $\mathcal{F}'$, identify the configuration shown in Fig. 6(b) to specify the position of one vertex which is assigned to one system coordinate, remove the identified vertex from $\mathcal{F}'$ and goto step **2b**. If no vertices are identified, goto step **2bc**;

2bc In $\mathcal{F}'$, identify the configuration of Fig. 6(a) to specify a position of one vertex which is assigned to two system coordinates, remove the identified vertex from $\mathcal{F}'$ and goto step **2b**;

Since: (i) the front $\mathcal{F}(\mathbb{H})$ completely separates the surface model $\mathbb{M}$ into disjoint components; and (ii) assigning system coordinates to $\mathcal{F}(\mathbb{H})$ can only have three cases shown in Fig. 6, the validity of the presented process in Fig. 8 is readily seen.

3.3. System coordinates assignment

System coordinates play a vital role in dynamic developable models. If a set of system coordinates (Ref. Eq. (4)) can be assigned, the Lagrangian of the surface model (Ref. Eq. (6)) is readily obtained, and the system of Euler–Lagrange equations (8) characterizes the dynamic process. The choice of a set of system coordinates for the description of a surface model subject to given constraints, is not unique. Akin to the different bases of a vector space $\mathbb{R}^n$, they are isomorphically equivalent, and the number of such coordinates is definite: it is the number of DOFs.

In the following, we present a detailed algorithm for system coordinates assignment for a developable model subject to a single external point constraint. The situation becomes much more complicated if multiple external point constraints (MEPC for short) exist, since some subsets of MEPC may over-constrain or violate portions of the surface model. For a clear and concise explanation of the mechanism of dynamic simulation on the developable model, we only address the single constraint case here, and we will present the non-trivial extension to the MEPC case in the sequel Liu et al. [20].

The proof in Section 3.2 already suggests the following algorithm to assign the system coordinates. Consider a developable model $\mathcal{M}$ subject to a single external point

constraint that can be located in any node of $\mathcal{M}$. Let the location of the external point constraint be v_{ext} and $\mathbb{H} = \{v_{\text{ext}}\}$. Denote the 1-ring neighbors of a vertex v as the vertices connecting to v by edges. The algorithm of system coordinate assignment can be summarized as follows:

Algorithm: `sys_coord_assnmt_single`

1. Mark all vertices in $\mathbb{M}$ by 0;
2. Mark all vertices in $\mathbb{H}$ by 1;
3. **while** $\mathcal{F}(\mathbb{H}) \neq \emptyset$, **do**
3.1. Set $\mathcal{F}' = \mathcal{F}(\mathbb{H})$;
3.2. **while** $\mathcal{F}(\mathbb{H}) \neq \emptyset$, **do**
3.2.1. $max\_cstr = 0$;
3.2.2. **for** every vertex v in $\mathcal{F}(\mathbb{H})$**do**
3.2.2.1. $cstr =$ the number of vertices marked "1" in the 1-ring neighbors of v;
3.2.2.2. **if** $cstr > max\_cstr$
3.2.2.2.1. $max\_cstr = cstr$; $max\_v = v$;
3.2.3. **if** $max\_cstr = 3$
3.2.3.1. assign configuration shown in Fig. 6(c) to max_v;
3.2.4. **else if** $max\_cstr = 2$
3.2.4.1. assign configuration shown in Fig. 6(b) to max_v;
3.2.5. **else if** $max\_cstr = 1$
3.2.5.1. assign configuration shown in Fig. 6(a) to max_v;
3.2.6. mark max_v by "1" and remove $max_v$from $\mathcal{F}(\mathbb{H})$;
3.3. Set $\mathcal{F}(\mathbb{H})$ be all vertices marked "0" in the 1-ring neighbors of $\mathcal{F}'$.

3.4. Generation of the system (4)

Algorithm `sys_coord_assnmt_single` assigns system coordinates by propagating the front $\mathcal{F}(\mathbb{H})$. The key is that in the loop of Step 3.2, the system coordinate assignment rules can be written down in a progressive manner, i.e., for Step 3.2.3.1,

$$v_{\text{unknown}} = f(v_{\text{known1}}, v_{\text{known2}}, v_{\text{known3}}) \tag{11}$$

for Step 3.2.4.1,

$$v_{\text{unknown}} = g(v_{\text{known1}}, v_{\text{known2}}, q_{\text{assigned}}) \tag{12}$$

for Step 3.2.5.1,

$$v_{\text{unknown}} = h(v_{\text{known}}, q_{\text{assigned1}}, q_{\text{assigned2}}) \tag{13}$$

where f, g, h are vector-valued functions characterizing the configurations in Fig. 6(c), (b) and (a), respectively. For a clear explanation, we write down a simple example (upon a consistent reordering of vertices):

$$\begin{aligned} v_1 &= h(v_{\text{ext}}, q_1, q_2) \\ v_2 &= g(v_{\text{ext}}, v_1, q_3) \\ v_3 &= g(v_{\text{ext}}, v_2, q_4) \\ &\cdots\ \cdots\ \cdots. \end{aligned}$$

if we keep substituting known vertex positions in terms of system coordinates into unknown vertex position functions from the upper rows to lower rows, the system of Eqs. (4) is readily obtained:

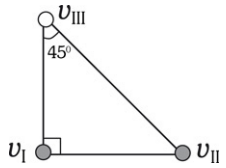


Fig. 9. System coordinate specification by function g.

$$\begin{aligned} v_1 &= h(v_{\text{ext}}, q_1, q_2) \\ v_2 &= g(v_{\text{ext}}, h(v_{\text{ext}}, q_1, q_2), q_3) \\ v_3 &= g(v_{\text{ext}}, g(v_{\text{ext}}, h(v_{\text{ext}}, q_1, q_2), q_3), q_4) \\ &\cdots\ \cdots\ \cdots. \end{aligned} \tag{14}$$

3.5. Linear approximation of system (4)

The function h in Eq. (13) is trigonometric in terms of spherical coordinates, while functions f, g in Eqs. (11) and (12) are compound trigonometric and radical functions in a complex form. Consider the function g, for example. Refer to Fig. 9. Given a triangle $T = \{v_{\text{I}}, v_{\text{II}}, v_{\text{III}}\}$, let the superscript "0" denote its initial state in $\mathbb{R}^2$ and "t" its time-t state. Now given one edge $\{v_{\text{I}}, v_{\text{II}}\}$ known at time t, we want to use a system coordinate q_{assigned} to uniquely locate the position v_{III}. The function g can be determined as follows:

(1) Calculate the angle θ between the two vectors $v_{\text{II}}^0 - v_{\text{I}}^0$ and $v_{\text{II}}^t - v_{\text{I}}^t$;
(2) Calculate and normalize the rotation axis $\mathbf{r} = (v_{\text{II}}^0 - v_{\text{I}}^0) \times (v_{\text{II}}^t - v_{\text{I}}^t)$;
(3) Rotate the triangle around the axis $\mathbf{r}$ to align vector $v_{\text{II}}^0 - v_{\text{I}}^0$ coinciding with vector $v_{\text{II}}^t - v_{\text{I}}^t$. Denote the four Euler parameters by

$$\theta_0 = \cos\frac{\theta}{2}, \qquad \theta_1 = r_1 \sin\frac{\theta}{2}, \qquad \theta_2 = r_2 \sin\frac{\theta}{2}, \qquad \theta_3 = r_3 \sin\frac{\theta}{2}$$

where $\mathbf{r} = (r_1, r_2, r_3)^T$, and the rotation of vector $\mathbf{v}$ around $\mathbf{r}$ with angle θ is given by

$$\mathbf{v}' = \mathbf{A}(\theta)\mathbf{v}$$

(4) Given a value q for q_{assigned}, for the case shown in Fig. 9, rotate the triangle around the axis $v_{\text{II}}^t - v_{\text{I}}^t$ (subject to normalization):

$$v_{\text{III}}^t = g(v_{\text{I}}^t, v_{\text{II}}^t, q) = \mathbf{A}(q)\mathbf{A}(\theta)(v_{\text{III}}^0 - v_{\text{I}}^0) + v_{\text{I}}^t$$

where

$$\mathbf{A}(\theta) = \begin{pmatrix} 1 - 2(\theta_2)^2 - 2(\theta_3)^2 & 2(\theta_1\theta_2 - \theta_0\theta_3) & 2(\theta_1\theta_3 + \theta_0\theta_2) \\ 2(\theta_1\theta_2 + \theta_0\theta_3) & 1 - 2(\theta_1)^2 - 2(\theta_3)^2 & 2(\theta_2\theta_3 - \theta_0\theta_1) \\ 2(\theta_1\theta_3 - \theta_0\theta_2) & 2(\theta_2\theta_3 + \theta_0\theta_1) & 1 - 2(\theta_1)^2 - 2(\theta_2)^2 \end{pmatrix}.$$

To simplify the above formulation, noting that a small time step Δt is used in the numerical integration, we precompute

$\frac{\partial f}{\partial v}, \frac{\partial g}{\partial v}, \frac{\partial g}{\partial q}, \frac{\partial h}{\partial v}, \frac{\partial h}{\partial q}$ and use first-order approximations for the functions f, g, h:

$$\begin{aligned}
v^t &= f(v_{\mathrm{I}}^t, v_{\mathrm{II}}^t, v_{\mathrm{III}}^t) \\
&= f(v_{\mathrm{I}}^{t-\Delta t} + \Delta v_{\mathrm{I}}, v_{\mathrm{II}}^{t-\Delta t} + \Delta v_{\mathrm{II}}, v_{\mathrm{III}}^{t-\Delta t} + \Delta v_{\mathrm{III}}) \\
&= v^{t-\Delta t} + \frac{\partial f}{\partial v_{\mathrm{I}}}\Delta v_{\mathrm{I}} + \frac{\partial f}{\partial v_{\mathrm{II}}}\Delta v_{\mathrm{II}} + \frac{\partial f}{\partial v_{\mathrm{III}}}\Delta v_{\mathrm{III}} \\
&\quad + \text{high order terms} \\
v^t &= g(v_{\mathrm{I}}^t, v_{\mathrm{II}}^t, q^t) = g(v_{\mathrm{I}}^{t-\Delta t} + \Delta v_{\mathrm{I}}, v_{\mathrm{II}}^{t-\Delta t} \\
&\quad + \Delta v_{\mathrm{II}}, q^{t-\Delta t} + \Delta q) \\
&= v^{t-\Delta t} + \frac{\partial g}{\partial v_{\mathrm{I}}}\Delta v_{\mathrm{I}} + \frac{\partial g}{\partial v_{\mathrm{II}}}\Delta v_{\mathrm{II}} + \frac{\partial g}{\partial q}\Delta q \\
&\quad + \text{high order terms} \\
v^t &= h(v_{\mathrm{I}}^t, q_1^t, q_2^t) \\
&= v^{t-\Delta t} + \frac{\partial h}{\partial v_{\mathrm{I}}}\Delta v_{\mathrm{I}} + \frac{\partial h}{\partial q_1}\Delta q_1 + \frac{\partial h}{\partial q_2}\Delta q_2 \\
&\quad + \text{high order terms.}
\end{aligned} \tag{15}$$

Accordingly, the system (14) becomes a set of linear homogeneous functions of $q_1, q_2, \ldots$, which makes the following numerical integration much easier to implement.

4. Numerical integration

The dynamic simulation of our developable model is governed by the system of Euler–Lagrange equations (8), which consists of n second-order ODEs. Rather than directly solving this system of equations, we can introduce a set of auxiliary variables called *system momenta* [25]:

$$p_i = \frac{\partial T}{\partial \dot{q}_i}, \quad i = 1, \ldots, n. \tag{16}$$

By Eqs. (2) and (5), the kinetic energy T is a quadratic form in terms of $\dot{q}_i$. Then in Eq. (16), each p_i is a linear homogeneous function of $\dot{q}_1, \ldots, \dot{q}_n$. Conversely, due to the positive definite characteristic of the energy T, the solution of the n equations in (16) must yield each $\dot{q}_i$ as a linear homogeneous function of $p_1, \ldots, p_n$. Upon substituting each $\dot{q}_i$ in T in terms of $p_1, \ldots, p_n, q_1, \ldots, q_n$, the *Hamiltonian* H of the system is defined as

$$H(q_1, \ldots, q_n, p_1, \ldots, p_n) = T + V. \tag{17}$$

That is, the Hamiltonian of a system is the total energy (including both potential and kinetic energy) of the system. Since T is a homogeneous function of order 2 in terms of $\dot{q}_1, \ldots, \dot{q}_n$, by Euler's homogeneous function theorem, we have

$$2T = \sum_{i=1}^{n} \dot{q}_i \frac{\partial T}{\partial \dot{q}_i} = \sum_{i=1}^{n} p_i \dot{q}_i$$

and with Eq. (6), the following is immediately obtained:

$$H = 2T - (T - V) = \sum_{i=1}^{n} p_i \dot{q}_i - L$$

$$\frac{\partial H}{\partial p_j} = \dot{q}_j + \sum_{i=1}^{n} \frac{\partial \dot{q}_i}{\partial p_j}\left(p_i - \frac{\partial T}{\partial \dot{q}_i}\right) = \dot{q}_j, \quad j = 1, \ldots, n. \tag{18}$$

It can be shown [25] that in terms of the Hamiltonian, applying Hamilton's principle which renders the integral (7) an extremum leads to the *Hamilton equations of motion* (together with identities (18)):

$$\dot{p}_i = -\frac{\partial H}{\partial q_i}, \qquad \dot{q}_i = \frac{\partial H}{\partial p_i}, \quad i = 1, 2, \ldots, n. \tag{19}$$

The system (19) constitutes $2n$ first-order ordinary differential equations, and is much easier to solve numerically than the original system (8). Given a valid initial state of the system, after determining the Hamiltonian function H, the motion of the system can be efficiently obtained by the simple forward Euler method:

$$\begin{cases} p_i(t + \Delta t) = p_i(t) - \dfrac{\partial H(t)}{\partial q_i(t)}\Delta t \\ q_i(t + \Delta t) = q_i(t) + \dfrac{\partial H(t)}{\partial p_i(t)}\Delta t \end{cases} \quad i = 1, 2, \ldots, n. \tag{20}$$

We emphasize that the system (19) is highly non-linear and even discontinuous, due to the collision forces possibly applied at any time in a complex virtual environment. Consequently it is difficult to use an implicit numerical method like in Baraff and Witkin [4] to solve it.

4.1. Fast and stable numerical solutions

To formulate $\frac{\partial H}{\partial q_i}$ and $\frac{\partial H}{\partial p_i}$ in system (19), substitute Eq. (4) into (2):

$$\begin{aligned}
T = \frac{1}{2}\sum_{i=1}^{n_{\mathcal{V}}} m_i \\
\times \left[\left(\sum_{k=1}^{n} \frac{\partial x_i}{\partial q_k}\dot{q}_k\right)^2 + \left(\sum_{k=1}^{n} \frac{\partial y_i}{\partial q_k}\dot{q}_k\right)^2 + \left(\sum_{k=1}^{n} \frac{\partial z_i}{\partial q_k}\dot{q}_k\right)^2\right].
\end{aligned}$$

Then Eq. (16) becomes

$$\begin{aligned}
p_j = \frac{\partial T}{\partial \dot{q}_j} = \sum_{k=1}^{n}\left\{\sum_{i=1}^{n_{\mathcal{V}}} m_i\left[\frac{\partial x_i}{\partial q_k}\frac{\partial x_i}{\partial q_j} + \frac{\partial y_i}{\partial q_k}\frac{\partial y_i}{\partial q_j}\right.\right. \\
\left.\left. + \frac{\partial z_i}{\partial q_k}\frac{\partial z_i}{\partial q_j}\right]\right\}\dot{q}_k, \quad j = 1, \ldots, n.
\end{aligned} \tag{21}$$

Known from Cramer's rule, the system of Eqs. (21) has a unique solution that expresses each $\dot{q}_j$ as a linear homogeneous function of $p_1, \ldots, p_n$. We have

$$\begin{cases} \dfrac{\partial H}{\partial p_j} = \displaystyle\sum_{i=1}^{n} \frac{\partial T}{\partial \dot{q}_i}\frac{\partial \dot{q}_i}{\partial p_j} = \sum_{i=1}^{n} p_i \frac{\partial \dot{q}_i}{\partial p_j} \\ \dfrac{\partial H}{\partial q_j} = \dfrac{\partial V}{\partial q_j} + \dfrac{\partial T}{\partial q_j} + \displaystyle\sum_{i=1}^{n} \frac{\partial T}{\partial \dot{q}_i}\frac{\partial \dot{q}_i}{\partial q_j}. \end{cases} \tag{22}$$

A considerable simplification can be achieved if the first order approximation (15) of system (14) is applied: since each function x_i, y_i, z_i is a linear homogeneous function of

$q_1, q_2, \ldots, q_n$, all terms of $\frac{\partial x_i}{\partial q_j}$, $\frac{\partial y_i}{\partial q_j}$, $\frac{\partial z_i}{\partial q_j}$, $i, j = 1, \ldots, n$, are constants, and system (22) is reduced to

$$\begin{cases} \dfrac{\partial H}{\partial p_j} = \displaystyle\sum_{i=1}^{n} p_i \frac{\partial \dot{q}_i}{\partial p_j} \\ \dfrac{\partial H}{\partial q_j} = \dfrac{\partial V}{\partial q_j} = g \displaystyle\sum_{i=1}^{n_\mathcal{V}} m_i \frac{\partial z_i}{\partial q_j}. \end{cases}$$

To evaluate $\frac{\partial \dot{q}_i}{\partial p_j}$, $i = 1, \ldots, n$, we first calculate the coefficients in system (21)

$$c_{jk} = \sum_{i=1}^{n_\mathcal{V}} m_i \left[\frac{\partial x_i}{\partial q_k}\frac{\partial x_i}{\partial q_j} + \frac{\partial y_i}{\partial q_k}\frac{\partial y_i}{\partial q_j} + \frac{\partial z_i}{\partial q_k}\frac{\partial z_i}{\partial q_j} \right], \quad j, k = 1, \ldots, n$$

at time t. Then the linear function $\dot{q}_i$ of $p_1, \ldots, p_n$, by solving system (21), can be expressed as $\dot{q}_i = \frac{D_{*i}}{D}$, where

$$D = \begin{vmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{vmatrix}$$

and

$$D_{*i} = \begin{vmatrix} c_{11} & \cdots & c_{1(i-1)} & p_1 & c_{1(i+1)} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{n(i-1)} & p_n & c_{n(i+1)} & \cdots & c_{nn} \end{vmatrix}.$$

It is immediately seen that

$$\frac{\partial \dot{q}_i}{\partial p_j} = (-1)^{i+j} \frac{M_{ji}}{D},$$

where M_{ij} is the minor of D formed by eliminating row j and column i from D. To evaluate the determinant of a matrix, recall that the determinant of an LU decomposed matrix is just the product of the diagonal elements; then the determinant evaluation can be efficiently achieved by performing the LU decomposition [23].

4.2. Numerical stability with an adaptive time step

The time step Δt is critical to ensure numerical stability in the scheme (20). To find a desired time step, a classical way with an explicit method is to view $p_i(t)$ and $q_i(t)$ as functions of time t and expand them as Taylor series. By Eq. (19), we have

$$\begin{cases} p_i(t+\Delta t) = p_i(t) - \dfrac{\partial H(t)}{\partial q_i(t)}\Delta t - O\left(\dfrac{\Delta^2 t}{2}\dfrac{\mathrm{d}}{\mathrm{d}t}\left(\dfrac{\partial H}{\partial q_i}\right)\right) \\ q_i(t+\Delta t) = q_i(t) + \dfrac{\partial H(t)}{\partial p_i(t)}\Delta t + O\left(\dfrac{\Delta^2 t}{2}\dfrac{\mathrm{d}}{\mathrm{d}t}\left(\dfrac{\partial H}{\partial p_i}\right)\right) \end{cases} \quad i = 1, 2, \ldots, n. \tag{23}$$

Compared to the scheme (20), the accumulation of the error terms in (23) is bounded by

$$\frac{\Delta^2 t}{2}\frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=1}^{n}\left(\frac{\partial H}{\partial q_i} + \frac{\partial H}{\partial p_i}\right) \le \frac{\Delta^2 t}{2\delta}\frac{\mathrm{d}(\Delta H)}{\mathrm{d}t} \tag{24}$$

Figure	No. vertices /faces	Frames per second	Step size min/max (ms)
11	25/32	33.33	50/250
12	81/128	3.33	18/150
13	289/512	0.152	0.2/10
14	81/128	1.67	2/150
15	244/440	0.435	1.5/30

Fig. 10. Performance summary: all tests are performed on PC with a Pentium III processor running at 937 MHz operating under Microsoft Windows XP.

where $\delta = \arg\min_i \{q_i(t+\Delta t) - q_i(t), p_i(t+\Delta t) - p_i(t), i = 1, \ldots, n\}$. Note that the Hamiltonian H is the total energy of a conservative system and thus, during simulation, ΔH should be zero in an ideal situation. If $|\Delta H|$ increases in the numerical process, this non-physical variation is due to the error terms in (23), and can be controlled by decreasing the time step Δt. More precisely, referring to the bound in (24), the time step should be inversely proportional to the square root of the change of Hamiltonian. To achieve numerical stability and efficiency with a low computational overhead, similar to that achieved in Joukhadar and Laugier [17], the basic idea is to monitor the value of ΔH at each time step, and to choose the largest time step which satisfies $\Delta H < \epsilon$ for a (small) pre-specified value ϵ.

5. Extension to non-conservative system

Although we have not implemented this aspect, we mention that our dynamic model can be extended naturally to a non-conservative system. For a non-conservative extension, we resort to the knowledge of the Hamilton's principle for a non-conservative system of particles [5]. To count non-conservative forces, the basic idea is to define *generalized forces* Q_k:

$$Q_k = -\frac{\partial V^*}{\partial q_k}$$

and the integral (7) becomes

$$I^* = \int_{t_1}^{t_2} (T + Q_k q_k)\,\mathrm{d}t$$

where $\delta W = Q_k \delta q_k$ is called the *virtual work*. Due to the limited scope of this paper, we will present the detailed formulation for non-conservative systems in a following up paper.

6. Applications

We have implemented the algorithms described in Sections 3 and 4. Examples illustrating the cases with both single and multiple external point constraints are presented below. In all experiments, the scene is set up by comparing to the real world: the mesh size of paper (or cloth) is proportional to the true size of real paper (or cloth); the mass of each mesh vertex is proportional to the true unit mass of paper (or cloth); the gravitational acceleration is set as 9.8. We use the dynamic bounding volume hierarchy method to handle collision

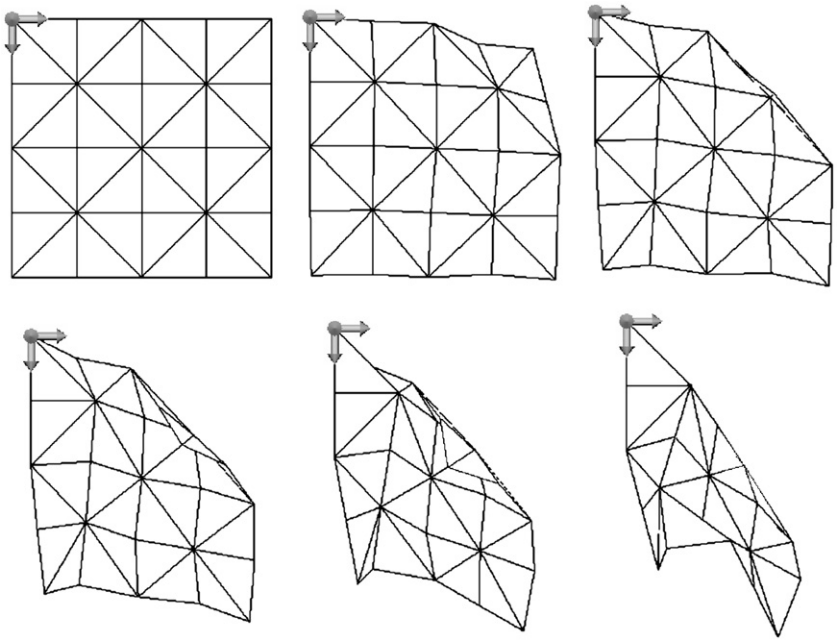


Fig. 11. Crumpled paper simulation using $\mathcal{M}^1$: the paper is holding on the left-upper corner.

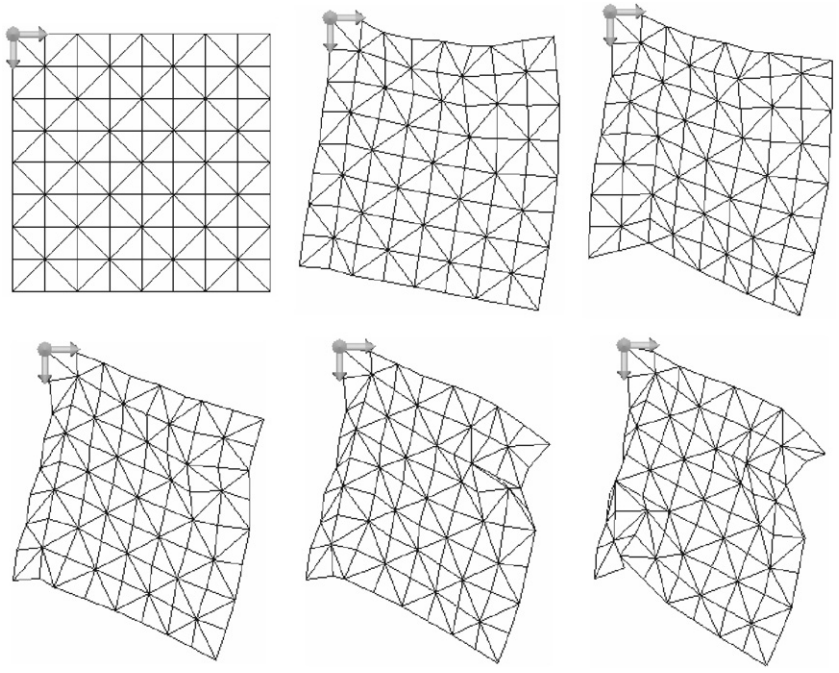


Fig. 12. Crumpled paper simulation using $\mathcal{M}^2$.

detection, including self-intersection. Rather than adding the viscosity at each vertex to simulate damping force that could result in a non-conservative system, we add this damping force to the velocity and we find that this simple scheme can produce good, physically plausible behaviors of the objects. Fig. 10 summarizes the performance of all our simulation results.

6.1. Crumpled paper simulation

The design and fabrication of three dimensional shape models made by paper-like sheet materials has received increasing attention, both in computer aided design [11] and in computer graphics [21]. Here we show two examples using a single external point constraint with different mesh resolutions. Fig. 11 shows an example of a piece of paper $\mathcal{M}^1$ holding on the left-upper corner. Fig. 12 shows the same process using a finer mesh $\mathcal{M}^2$. Two notes follow. First, clearly the fine mesh looks softer than the coarse mesh; this agrees with our statement in Lemma 2 that a finer mesh can offer more DOFs. Secondly, as predicted by Amar and Pomeau [1], the crumpling of a piece of paper should "leave permanent marks, very localized and

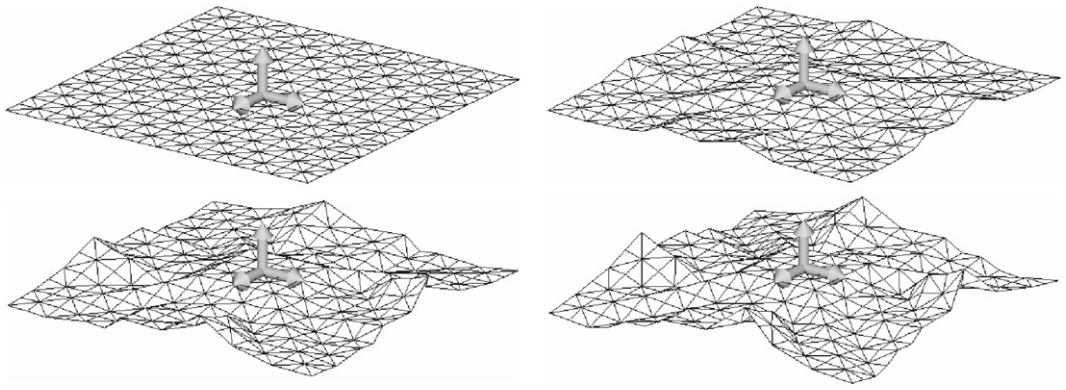


Fig. 13. Buckled developable surface simulation in sheet metal stamping.

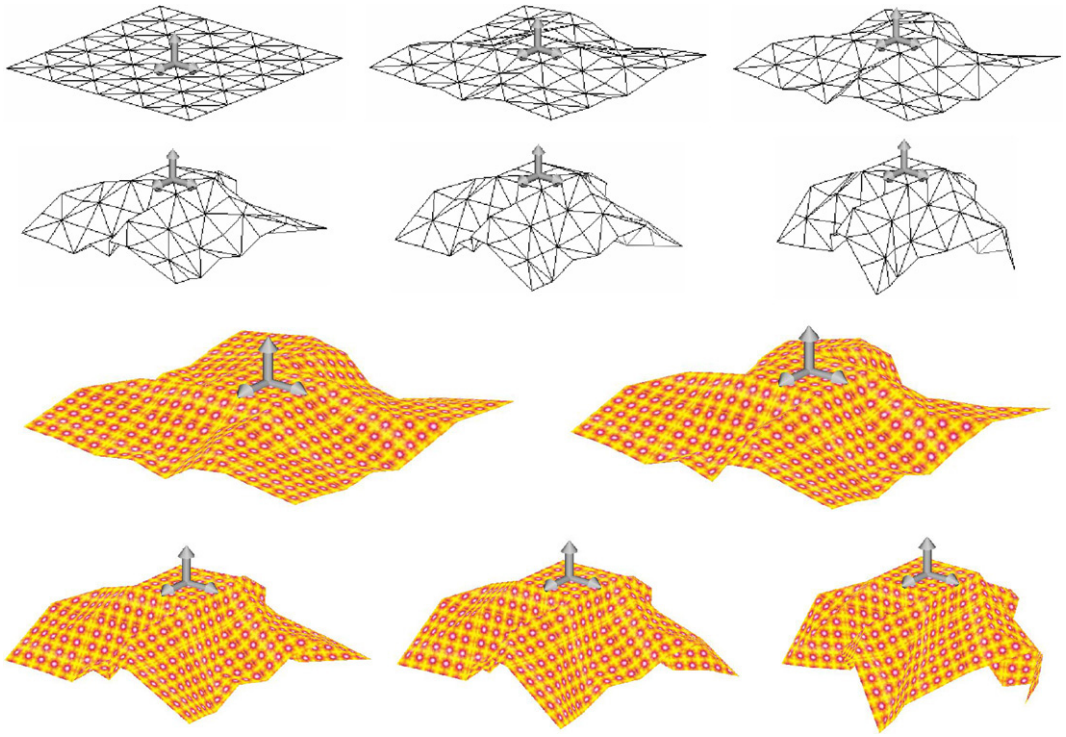


Fig. 14. A rectangular plastic tablecloth draped over a rectangular table using $\mathcal{M}^2$.

with a typical crescent shape"; our experimental results meet this requirement very well.

6.2. Sheet metal stamping

Sheet metal stamping encompasses the wide range of metal-forming operations that bend, or reshape metal without creating chips. The press provides the power that transforms the sheet metal. Specially designed dies determine the final product. In Fig. 13 a square die is used, and it punches upwards. The animation sequence shows clearly how the buckled developable surface is formed.

6.3. Cloth simulation

Cloth modeling has been extensively studied in textile mechanics and engineering communities, as well as by the computer graphics community [16,24]. Two major challenges exist. Firstly, cloth has the macroscopic property of negligibly small strain along with large displacement. Secondly, cloth frequently exhibits plenty of microscopic structures, e.g., folds and wrinkles. Since cloth in most cases undergoes large displacements with negligibly small strains at macro-level, we can use the developability of our dynamic model to characterize large displacements without stretching. If the developable model is refined sufficiently, the resulting dynamic mesh can model cloth structures at the micro-level.

In Figs. 14 and 15 we show two plastic tablecloth draping simulations, each with two different resolutions. In Fig. 14, we simulate a rectangular cloth over a rectangular table and we observe clearly the principal folds formed, from the table corners to the corresponding cloth corners. This property is further verified in the example shown in Fig. 15, in which we simulate an octagonal cloth draping over an octagonal table. If

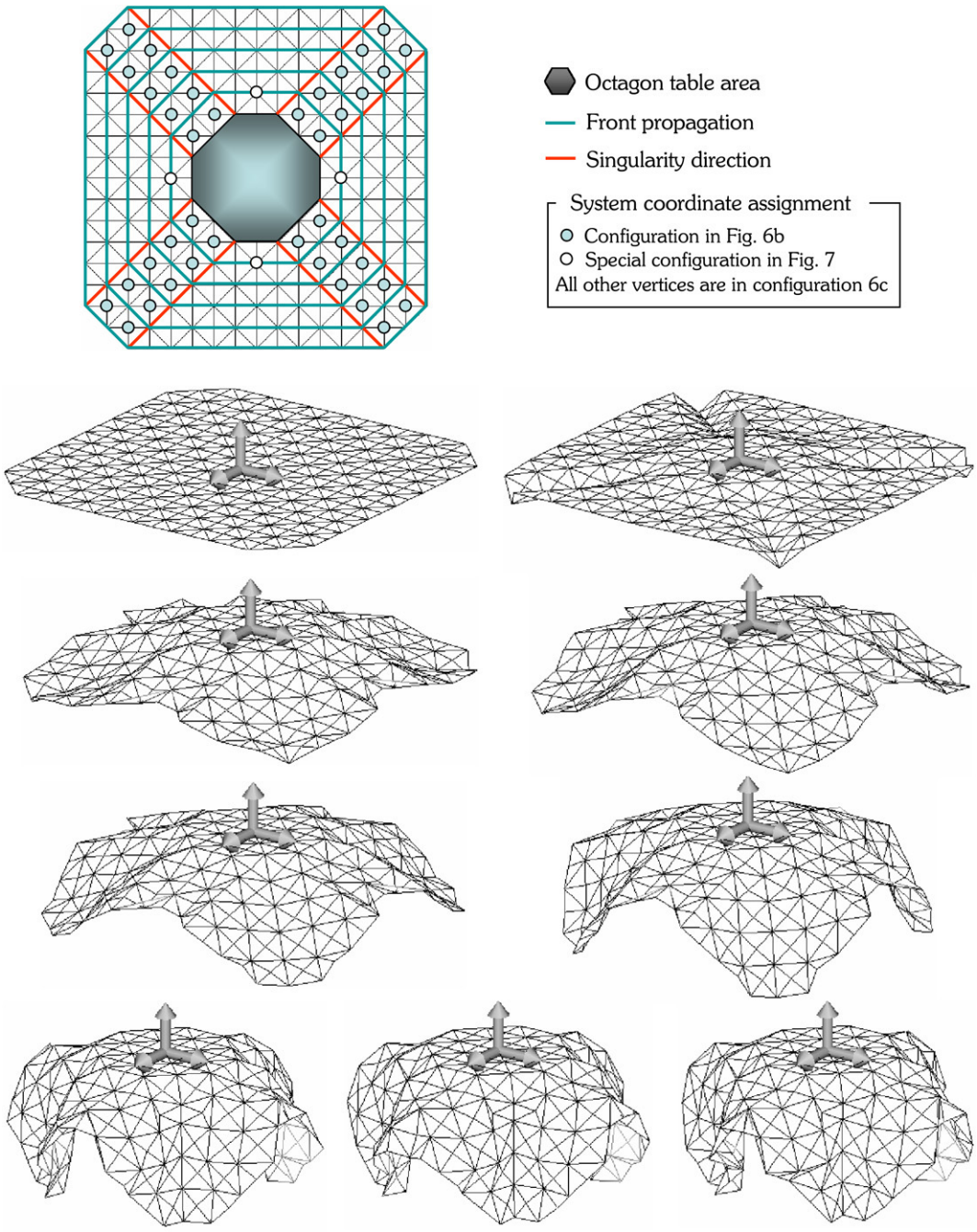


Fig. 15. An octagonal plastic tablecloth model draped over an octagonal table.

we simulate an n-gonal cloth draping over an n-gonal[3] table by refining the mesh and letting $n \to \infty$, we will get the result of a round cloth draping over a round table; in this case, we can infer that uniform folding patterns will occur.

7. Conclusion

Many industrial applications require the free form surfaces used in computer aided design not only to be *developable* but also to have the property of being *dynamic*. Towards this end, in this paper we present a study on dynamic developable surface models using the Hamilton principle. By adopting triangle meshes as the underlying surface representation, similarly to Frey [13] our proposed technique can model buckled developable surfaces. The detailed dynamization mechanism, as well as practical numerical computation, are analyzed in depth with the aid of system coordinates. This mechanism we use is quite different from most existing work on developable surfaces, and shows an interesting new way for modeling developable surfaces.

[3] Akin to rasterizing a circle over a mesh, the resulting n-gon is not necessarily convex.

Acknowledgements

The first author was supported by the National Natural Science Foundation of China (Project Number 60603085), the National High Technology Research and Development Program of China (Project Number 2006AA01Z304) and the National Basic Research Program of China (Project Number 2006CB303104).

References

[1] Amar MB, Pomeau Y. Crumpled paper. Proceedings of the Royal Society of London. Series A 1997;453:729–55.
[2] Aumann G. Interpolation with developable Bezier patches. Computer Aided Geometric Design 1991;8:409–20.
[3] Aumann G. A simple algorithm for designing developable Bezier surfaces. Computer Aided Geometric Design 2003;20:601–19.
[4] Baraff D, Witkin A. Large steps in cloth simulation. In: Proceedings of SIGGRAPH'98. 1998. p. 43–54.
[5] Bedford A. Hamilton's principle in continuum mechanics. Pitman Advanced Publishing Program; 1985.
[6] Bodduluri R, Ravani B. Design of developable surfaces using duality between plane and point geometries. Computer-Aided Design 1993;25: 621–32.
[7] Chu C, Sequin CH. Developable Bezier patches: Properties and design. Computer-Aided Design 2002;34:511–27.
[8] Cipra BA. In the fold: Origami meets mathematics. SIAM News 2001; 34(8).
[9] Demaine ED, Demaine ML. Recent results in computational origami. In: Proc. 3rd inter. meeting of origami science, math and education. 2001. p. 3–16.
[10] Edelsbrunner H, Mucke E. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. ACM Transactions on Graphics 1990;9:66–104.
[11] Elber G. Model fabrication using surface layout projection. Computer-Aided Design 1995;27:283–91.
[12] Frey WH. Boundary triangulations approximating developable surfaces that interpolate a close space curve. International Journal of Foundations of Computer Science 2002;13:285–302.
[13] Frey WH. Modeling buckled developable surface by triangulation. Computer-Aided Design 2004;36:299–313.
[14] Giblin PJ. Graphs, surfaces and homology. Chapman and Hall; 1981.
[15] Hoschek J, Pottmann H. Interpolation and approximation with developable surfaces. In: Dahlen M, Lyche T, Schumaker LL, editors. Mathematical methods for curves and surfaces. 1995. p. 255–64.
[16] House DH, Breen DE. Cloth modeling and animation. A.K. Peters; 2000.
[17] Joukhadar A, Laugier C. Adaptive time step for fast converging dynamic simulation system. In: Proc. of IEEE RSJ int. conf. on intelligent robots and systems. 1996. p. 418–24.
[18] Kergosien YL, Gotoga H, Kunii TL. Bending and creasing virtual paper. IEEE Computer Graph Applications 1994;14:40–8.
[19] Lang J, Roschel O. Developable $(1, n)$-Bezier surface. Computer Aided Geometric Design 1992;9:291–8.
[20] Liu YJ, Tang K, Joneja A. Modeling dynamic developable meshes II: Detailed algorithmic implementation. Manuscript. 2007 [in preparation].
[21] Mitani J, Suzuki H. Making papercraft toys from meshes using strip-based approximate unfolding. ACM Transactions on Graphics 2004;23:259–63.
[22] Pottmann H, Farin G. Developable rational Bezier and B-spline surfaces. Computer Aided Geometric Design 1995;12:513–31.
[23] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C++. Cambridge University Press; 2002.
[24] Volino P, Magnenat-Thalmann N. Virtual clothing: Theory and practice. Springer; 2000.
[25] Weinstock R. Calculus of variations: With applications to physics and engineering. Dover Publishers; 1974.