2015 IEEE International Conference on Multimedia Big Data

Fast Discrete Intersection Detection for Cloth Penetration Resolution

Juntao Ye
Institute of Automation
Chinese Academy of Sciences
Beijing, China
Email: juntao.ye@ia.ac.cn

Timo R. Nyberg
Dept. of Industrial Eng. and Management
School of Science, Aalto University
Espoo, Finland
Email: timo.nyberg@aalto.fi

Gang Xiong
Cloud Computing Center
Chinese Academy of Sciences
Dongguan, China
Email: xionggang@casc.ac.cn

***Abstract*—Penetrations are often unavoidable in modeling cloth or other deformable surfaces. It could exist as an initial configuration or show up in the middle of a dynamics simulation process. We present a new method for resolving such penetrations. To be adapted to a wide range of applications, this method is based on history-free discrete intersection detection (DID). It is also orientation-free as it does not assume any front- and back- face identification. Our method relies on *dynamic repulsive normal* (DRN) to compute proper displacements to relocate vertices to be intersection-free. First, intersection contours are constructed and classified, followed by a global analysis of the collision configurations. Then for some types of configurations, penetrating (illegal) regions are identified using a heuristic paradigm "small region is illegal". For those surfaces having identifiable illegal regions, proper displacements for incorrectly configured vertices are computed using DRN. For those configurations that do not clearly define legal/illegal regions, displacements are designed to push one mesh to the boundary of the other. The proposed method can also be used in the context of time-dependent simulation of complex deformable surfaces, making it an competitive alternative to the popular CCD-based approach.**

I. Introduction

The majority of cloth and other soft body modeling systems adopt *continuous collision detection* (CCD) to predict impending collisions, then attempt to prevent them from happening by altering the particles' velocities. The success of CCD-based response relies on a hard constraint: an intersection-free state for not only the initial configuration but also the starting of every time interval in the simulation. There are also applications in which an intersection-free initial state is impossible, or the simulation context is subject to external constraints forcing the cloth into illegal states for a number of consecutive steps. Moreover, although every collision event carries a time tag in CCD, they are often neglected, as resolving all the events strictly according to their time order can be too slow to halt the simulation. Thus simultaneous response is often adopted.

Contrast to collision prevention methods, collision repair strategy allows penetrations to occur but try to detect and fix them. It employs *discrete intersection detection* (DID) so it is history-free. DID is expected to be more efficient for two points: bounding volumes are more tight-fitting and no cubic solver is needed. For most cloth meshes there is no way to define inside/outside, this strategy is also orientation-free. So far it has largely been used as a complementary mechanism to CCD-based framework.

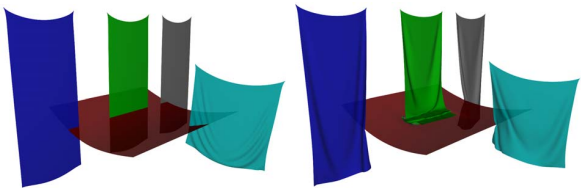

Fig. 1: Resolving five intertwined sheets. Three of them have $2,754$ particles each, and the flat one has $6,561$ and the other has $1,681$.

We envision a DID-based collision correction framework, not only for untangling penetrations, but also as an alternative to the popular CCD-based approaches. However, designing a purely history-free and orientation-free response algorithm is often an ambiguous and ill-defined problem (see Fig 2 for such an example). Recall that in CCD-based response, plane normal plays an important role in detecting if a geometric primitive (edge or vertex) is crossing a specific plane, and is also crucial in defining repulsion force or impulse to stop the approaching. The proper normal direction is so picked that it is consistent with the negative approaching velocity. That is to say the surface orientation is derived from the history information. Yet history information such as velocity is unavailable in DID, so heuristic has to be resorted to identify the penetration regions. Instead of imposing a hard constraints as in CCD, we only make an assumption which is easily satisfied: most of a surface is of legal status at any time of the simulation.

This paper extends the previous correction-based work by introducing *dynamic repulsive normal* (DRN). We first perform an efficient discrete intersection detection (§3). Then the intersection contours are constructed (§4.1), and classified to identify the penetrating/illegal regions (§4.2) by using a heuristic paradigm "small region is illegal". For those meshes having identifiable illegal regions, proper correcting displacements are computed for incorrectly configured vertices using DRN (§5.1). For those configurations that do not clearly define legal/illegal regions, displacements are designed to push one mesh to the boundary of another one (§5.1). In both cases, the displacements are first computed per E-F (edge-face) pair

978-1-4799-8688-0/15 $31.00 © 2015 IEEE
DOI 10.1109/BigMM.2015.22

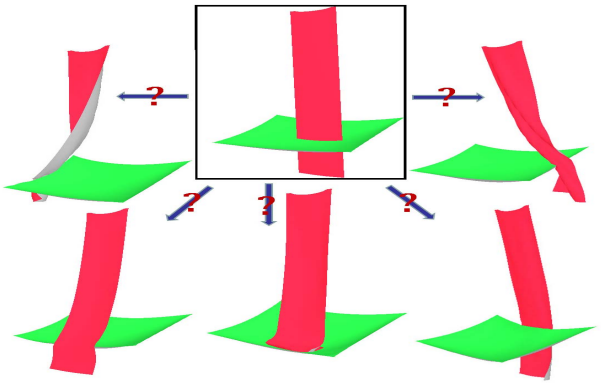

Fig. 2: The ambiguity in resolving the collision configuration in the top-middle image. The heuristic we use will choose the bottom-middle image as the after-collision configuration.

basis, then applied to corresponding vertices via either a local or global scheme (§5.2). The above process is repeated in a detect-correct-detect manner until the vanish of all intersection contours.

II. Related Work

CCD-based method is the dominant solution for cloth collision handling. Provot [1] proposed a cubic solver for detecting penetration of two elastic moving triangles. Bridson et al. [2] presents a comprehensive way of preventing intersections to occur. Asynchronous contact model [3], [4] produces very realistic effects. Another history-based response method for the domain of geometric modeling was proposed in [5]. This method resolves interference by reducing the magnitude of STIV (space-time interference volume swept by the trajectory of the penetrating regions) to zero through a minimization process.

Baraff et al. [6] were probably the first to address the necessity of collision correction in complex simulation environment, and put forward a partial solution. Their method is limited to closed regions and cannot handle intersection involving mesh boundary. Wicke et al. [7] further investigated this problem by taking consideration of boundary collisions. They analyzed all possible collision configurations and presented a classification criterion for intersection contours. Then two response schemes were proposed to handle different types of contours. Unfortunately their method has never been used as standalone response mechanism in complex simulations, thus its effectiveness is unclear. An intersection contour minimization (ICM) method [8] was proposed and it does not suffer from the limitations of open paths. This method resolves intersections by inducing relative displacements which minimize the length of the intersection contour. Aiming at a broader application field, no contour classification is needed and all types of contours are treated uniformly. While this method works fine for open contours which either start from or end at the surface boundary, it often fails for closed interior contours. Ye and Zhao [9] pointed out that the failure or slow convergence of ICM is due to ambiguous normal direction used.

III. The Discrete Intersection Detection

Surfaces intersections are typically detected as intersections between edges and faces/triangles. This elementary test in DID will be called *E-F test* thereafter. Extensive effort has been devoted to CCD over the past few years, and a number of acceleration techniques have been put forward. As CCD, our broad-phase collision detection is performed using Bounding Volume Hierarchies (BVH) with k-DOPs. While BVH can provide high culling efficiency, there is still a very high number of *false positives*, largely caused by BVs of adjacent triangle primitives. Two techniques have been introduced [10], [11] to the CCD pipeline to address this issue. One is called *procedural representative triangles* (PRT) for removing all redundant elementary tests (both V-F and E-E) between nonadjacent triangles. The other is *orphan sets* which exploit the mesh connectivity to eliminate redundant elementary tests between adjacent triangles. Using V-F PRT cuts the V-F test redundancy by 5/6 (assuming the average vertex valence is six), and the E-E PRT cuts the E-E test redundancy by one half, as each edge is shared by two triangle.

Similarly in DID, redundant E-F tests exist because an edge is typically shared by two triangles. We extend the PRT and orphan set techniques to the elementary E-F test in DID, putting forward the concepts of *E-F PRT* and *E-F orphan set*. Using them will successfully cut the E-F test redundancy by a half.

A. Acceleration with E-F PRT

For non-adjacent triangle pairs, we propose the technique of E-F PRT, to guarantee that all necessary E-F tests will be evaluated once and only once. As a common sense, each triangle is identified by a unique index number in the mesh, and the indexing is usually fixed to be unchanged throughout the simulation. Of the two incident triangles to an edge, we designate the one with the smaller index as the PRT for this edge. The fixed indexing guarantees that for a given edge we can always find the same triangle as its PRT. When an E-F test comes from non-adjacent pairs, at least one of the edge's two adjacent triangles is not adjacent to the intersecting face, which means the E-F PRT always make sense. Using E-F PRT, every elementary test will be performed once and only once, thus cuts the E-F test redundancy by a half, as each edge is generally shared by two triangles. The completeness and correctness of PRT are testified in [11].

B. Acceleration with E-F Orphan Set

As adjacent F-F pairs are never culled off in BVH culling, they are treated differently from the non-adjacent pairs. Adjacent pairs have two categories, *co-edge pair*, and *co-vertex pair*. When doing E-F tests out of adjacent pairs, not all six E-F tests are necessary because of the common element shared by the two triangles. Moreover, some E-F pairs between adjacent triangles may be executed in non-adjacent pair phase, thus not necessary to be done again. An E-F pair from a co-vertex triangle pair that do not get performed during the non-adjacent phase is called *E-F orphan pair*. All orphan pairs

constitute *orphan set*, which is a subset of the E-F tests from adjacent triangle pairs. This relationship is similar to the V-F and E-E tests from adjacent pairs in CCD, in which Tang et al. [11] introduce an optimization algorithm based on the work of Govindaraju et al. [12].

Although V-F orphan pairs and E-E orphan pairs exist in a closed mesh for CCD, E-F orphan pairs only appear in meshes with boundary. In a closed mesh, all the E-F tests can be executed in non-adjacent phase, so no E-F test is needed in the adjacent phase. In other words, the edge E in an E-F orphan pair is a boundary edge. This makes the E-F orphan set different from V-F and E-E orphan sets in CCD, in that E-F test can not happen between co-edge triangle pairs. Moreover, only an edge that is not incident to the shared vertex will be part of an E-F orphan pair. That is to say, for a co-vertex triangle pair, at most two E-F orphan pairs can be formed. However, E-F orphan pair does not always exist in a co-vertex triangle pair. It is sufficient to identify all E-F orphan set in a pre-processing step. First, all the co-vertex triangle pairs are constructed. Then for every pair, only the edge that is not incident to the common vertex has potentials to intersect with the other triangle, so only two edges from the pair need to be tested.

IV. Intersection Contour Construction and Classification

A. Contour Construction

The above DID process only outputs a set of E-F intersections. Many collision resolving algorithms [6], [8], [7], [9] rely on the topology of the intersection contours to determine the legal/illegal regions, so a robust intersection contour reconstruction algorithm is necessary. The construction is performed in a bottom-up manner in four stages, as will be detailed in the following.

Register intersection points. For every E-F intersection point, we register it to its *host triangles.* Each point will generally has three host triangles, including the two adjacent triangles incident to the intersecting edge and the corresponding intersecting face in this E-F pair. If the edge is on the boundary, there are only two host triangles. As a result, every triangle has at least two E-F intersection points.

Construct in-face poly-lines. Using the PRT and orphan set techniques imposes a little trouble for intersection contour construction. Without using such tricks, each intersecting F-F pair produces an intersection segment, thus concatenating all these segments head-to-tail will recover the whole contour of two intersecting meshes. With the acceleration techniques, not all six E-F tests are executed in each F-F test, thus we will not get a list of line segments immediately. Instead, we are end up with a set of E-F intersection points without connectivity information. For each face, there may be multiple intersection points registered to it. With the concept of host triangle for every intersection point, two conditions must be satisfied simultaneously if two intersection points in one face are to be connected:

- Two intersection points have two common host triangles.
- None of the two common host triangles is adjacent to the intersecting edge.

Constructing global contours. After all the in-face poly-lines for every single triangle are constructed, these poly-lines must be further connected to form global contours. This is done via a depth-first traversal of every contour. We have ever attempted to do it via width-first traversal with the help of BVH, but found it to be less efficient.

Group end-vertices. Vertices of the intersecting edges along a contour are divided into two groups, one group is in penetrating positions and the other is in valid position. An edge could intersect multiple faces at one time, thus having more than one intersection points. Odd number of intersections implies the two end vertices belong to separate groups.

B. Determining Legal/illegal Regions

The success of the global intersection analysis depends on the classification of the contour. We follow the classification criterion given in [7], as shown in Figure 3. The top row shows the collision configurations, and the bottom row shows the contours drawn on each (part) of the mesh. This classification is applicable as long as the cloth surface is a 2-manifold, whether it is closed or with boundaries. Cases (d) (e) (f) (g) have *loop vertices*, thus they only occur in self-collision. The term loop-vertex was coined in [6] to refer to the common vertex shared by two intersecting adjacent triangles. Contours fold back onto themselves after reaching a loop vertex, so it is often helpful to unfold such contours for classification. There are five types of contours:

- **CL:** closed curve; no loop-vertex.
- **BB:** open; both ends on boundary; no loop-vertex.
- **BLI:** one end on boundary, the other inside; one loop-vertex.
- **BI:** one end on boundary, the other inside; no loop-vertex.
- **II:** open; both ends inside; no loop-vertex.

In a collision configuration, the most common contour pairs are of types CL/CL, BB/II and BI/BI. Some configurations containing loop-vertex do not lead to a pair of contours, but one single contour of type CL or BLI. Figure 3 (d) and (g) are such examples. Note that a BB contour usually comes along with an II contour. Very rarely is the case that a BB can also correspond to a BI or BB contour. These configurations (BB/BI and BB/BB) are unstable due to boundary-boundary intersection. Depending on how the round-off error is handled in the E-F test, they are often classified to either BB/II or BI/BI. For the same reason, BLLB is another unstable contour and is treated as two BLI contours for handling.

Of the five contour types, CL and BB were defined in [7] as *partitioning* contours, i.e. they partition a mesh into two components. However, we find it is true for BB contour only if the mesh has one boundary. BB contours in multi-boundary meshes (e.g. cylindrical surface) are more complicated to handle and will be discussed in §7. For now, we assume the mesh has at most one boundary. Type II always corresponds to BB in which the illegal region can be identified, then II contour

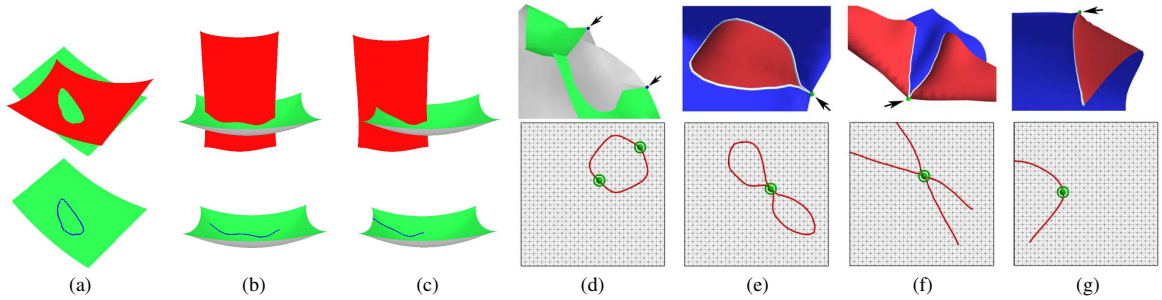

Fig. 3: Contours classification. Loop vertices are marked on the contours. (a) CL/CL pair. (b) BB/II pair. (c) BI/BI pair. (d) Loop-loop contour is unfolded to one CL contour. (e) closed contour with a single loop vertex is split to CL/CL pair. (f) The CROSS is split to BB/II pair. (g) The unfolded BLI contour. Images (e)(f)(g) courtesy of Wicke et al.

will diminish automatically along with the diminishing of BB. BI always corresponds to BI, and the BI/BI configuration is resolved by pushing one mesh to the boundary of another mesh, as will be detailed in §V-A.

It was proposed in [7] that BLI can be treated as a partitioning contour by construct a cone and flatten it. As loop-vertex occurs so rarely and creating a collision scene with a loop-vertex is not easy, we have no chance to validate the effectiveness of their method. And also please note that none of the examples in this paper involves any loop-vertex, and we focus on the first three penetration cases of Figure 3.

With the paradigm "small is illegal", the small region delimited by CL, BB or BLI is regarded as on the wrong side. It is possible that a mesh has multiple contours and is partitioned into more than two components. A fact is that the two components on the different sides of a contour always have opposite legal/illegal status. In the global analysis, these components are divided into two groups and their summed areas are compared. The flood-fill algorithm were typically used to find the smaller region on a partitioned mesh in the literature.

V. Untangling with Position Displacement

A. The Dynamic Repulsive Normals

In dynamics simulation, two triangles are allowed to be close enough but should never penetrate. To fix the illegal status of penetration, the simplest scheme is to re-located the two triangles to an intersection-free configuration. Surface normals play a very important role in bouncing back the colliding geometric primitives. A polygonal mesh, with clearly defined normal directions to denote inside/outside, can be called *oriented surface*. For untangling two oriented surfaces, two factors are considered in developing a scheme. First, one mesh ought to be pushed along the "outside-pointing" normal direction of the other mesh. Second, the length of the intersection contour should be reduced, ultimately leading to a complete disappearance. The task of untangling two intersecting meshes can be broken down into separating a series of E-F intersections. In a collision configuration shown in Figure 4, edge e, shared by two green triangles, intersects a

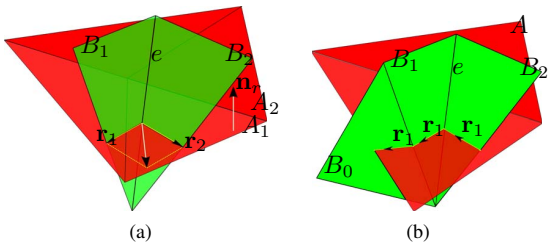

Fig. 4: (a) BB/II configuration where green mesh B is partitioned by the intersection contour. The DRN $\mathbf{n}_r$ is up for face A_1. (b) BI/BI configuration where the intersection contour does not partition either meshes.

red triangle in mesh A. We denote $\hat{\mathbf{r}}_1$ and $\hat{\mathbf{r}}_2$ two unit vectors on the intersection contour, originated from the intersection point. Let us temporarily suppose that the normal vector $\mathbf{n}_r$ unambiguously designates the outside direction of mesh A. According to the above discussion, the displacement applied to e for separation has two components: one along $\mathbf{n}_r$ to push it outside of mesh A, and the other within the plane of A to shorten the length of the contour. The direction of the displacement is thus defined as

$$\mathbf{d}_e = \hat{\mathbf{n}}_r + \lambda(\hat{\mathbf{r}}_1 + \hat{\mathbf{r}}_2). \tag{1}$$

$\mathbf{n}_r$ is called the *repulsive normal*, to differentiate from normals for the rendering purpose. The effect of $\lambda(\hat{\mathbf{r}}_1 + \hat{\mathbf{r}}_2)$ is to straighten the contour until $\hat{\mathbf{r}}_1$ and $\hat{\mathbf{r}}_2$ become collinear. To avoid over-shooting, we choose small value: $\lambda = 0.1$, for the in-plane component. The small λ also decreases the possibility of moving the contour to be stuck at a local minima in the case of concave surfaces.

Now that the direction of $\mathbf{n}_r$ in Equ 1 has to determined. It could be any of the two opposite directions perpendicular to a triangle, but a wrong choice will simply push the colliding geometries further in the wrong side. Naively using the rendering normal of a face as its $\mathbf{n}_r$ does not work for many circumstances. For open surfaces (i.e. surfaces with boundaries), there is no way to define inside/outside, and other "invading" objects could come from either side. Even for a

closed surface, if it involves self-collision, using the outside-pointing render normal as $\mathbf{n}_r$ in Equ 1 is unsafe. Moreover, if a polygonal mesh is penetrated at more than one spots by other objects, each penetration region has its own repulsive normal. Repulsive normal for a region could also vary from time to time, along with changes of the collision configurations. Since it is difficult to foresee the dynamic behavior of deformable surfaces, pre-setting the repulsive normals is impossible. We thus introduce the concept of *dynamic repulsive normals* (DRN), which implies that direction of $\mathbf{n}_r$ is determined on-the-fly and always has the tendency to push any invading objects back to where they came from.

The DRN is set for the face of an E-F intersection, and points to the side that the legal region of the other mesh resides in. In the configuration of Figure 4(a), the contour is of type BB for the green mesh and type II for the red mesh. The green mesh is partitioned and the top part is treated as legal. To resolve the e-A_1 intersection, the DRN for A_1 is set to point up and the displacement vector applied to edge e is computed via Equ 1. Note there is no need to set DRN for face A_2 as it does not involve any E-F intersection.

If a contour pair is of type BI/BI, as shown in Figure 4(b), no mesh is partitioned thus the DRN is undetermined. In this case the DRN vector is set to zero, contributing nothing to the displacement vector $\mathbf{d}_e$. To resolve the E-F intersection, the edge is pushed towards the boundary of the other mesh. Of the two in-plane vectors $\hat{\mathbf{r}}_1$ and $\hat{\mathbf{r}}_2$ for an intersecting edge, from a bookkeeping point of view $\hat{\mathbf{r}}_1$ is picked this way: traveling in the direction of every $\hat{\mathbf{r}}_1$ along the contour will arrive at the boundary of the other mesh. Then Equ.1 degenerates to

$$\mathbf{d}_e = \hat{\mathbf{r}}_1. \tag{2}$$

To conserve the momentum of the whole system, once a vector $\mathbf{d}_e$ is computed, either via Equ 1 or Equ 2, $-\mathbf{d}_e$ is introduced and applied to the corresponding face. This works as a pair of action and reaction forces which are equal but opposite.

Although neither Equ 1 nor Equ 2 is applicable to edges along a type II contour, it is not a concern. As in Figure 4(a), the type II contour in mesh A comes together with the BB contour in mesh B, and the DRN for faces in B are undetermined so vector $\mathbf{d}_e$ for intersecting edges of mesh A cannot be computed. Fortunately face A_1 receives $-\mathbf{d}_e$ from edge e of mesh B, thus the intersection can still be untangled.

B. Applying the Displacement Vectors

Collision response is performed by enforcing the collision constraints through position corrections distributed on mesh vertices. After one pass of correction, intersection detection is perform on the newly positioned meshes, and do another pass of response if needed, and so on in an iterative manner. [8] proposed a scheme for applying displacement vectors. Vectors $\mathbf{d}_e$ and $-\mathbf{d}_e$ are distributed as position changes to corresponding vertices. A vertex p_i is usually shared by multiple intersecting edges or faces, thus receives multiple contributions: $\mathbf{d}_{p_i} = \sum \pm \mathbf{d}_{e_j}$. The magnitude of the position correction vector has to be carefully chosen. Small value means large number of iterations needed for a complete response. Large size could easily crash the simulation, as dynamics simulator is vulnerable to brutal position or velocity changes. A scaled arctangent function is adopted to calculate the actual magnitude, so the final position correction is $h_0 \frac{|\mathbf{d}_{p_i}|}{\sqrt{|\mathbf{d}_{p_i}|^2 + g_0^2}} \hat{\mathbf{d}}_{p_i}$, where g_0 defines a progressive slope of the function, and $\hat{\mathbf{d}}_{p_i}$ is unit vector. [8] also suggested a global scheme that sums up all local corrections and then uniformly apply it to all involved vertices. That scheme is usually converges faster.

VI. Applications and Results

To test the effectiveness of proposed method, we run various examples on a single thread of a 2.13GHz Intel Xeon CPU.

A. Garment Fitting

Garment fitting is to "dress" a human model with an existing garment model. The fitting process may restrict the deformation to be subject to physical simulation or just within the geometric domain. We show an example of the latter case in Fig 5. Two layers of clothes are dressed on a lady, and the penetrations exist between shirt and pants, as well as shirt/body and pants/body. They are resolved according to specified ease-of-allowance.

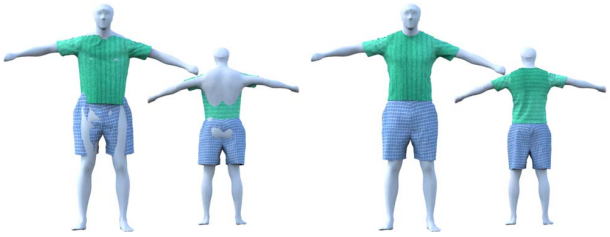

Fig. 5: Garment fitting starts with an initial penetration state.

B. Dynamics Simulation

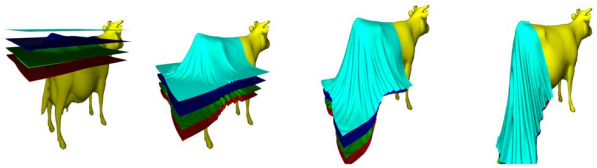

Fig. 6: Four sheets, each modeled by 6,561 particles, fall on a cow model.

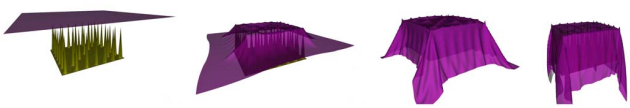

Fig. 7: Simulation of a sheet of 6,561 particles falling on spikes.

The proposed framework has been integrated into a cloth simulator. Figure 1 shows an example of untangling multiple

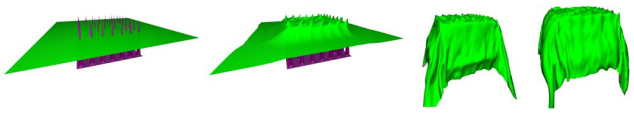

Fig. 8: Simulation of a sheet of 6,561 particles interacting with moving spikes.

intertwined sheets. The initial state consists of different types of collision configurations (e.g. BB/II and BI/BI), and they all have been resolved simultaneously in a reasonable manner. Note resolving the BB/II intersection (between the vertical green sheet and the horizontal one) is challenging as the position adjustments need to counteract the gravity. This is also one of several configurations that cannot be resolved in [8]. Figure 6 demonstrates four sheets falling on a cow model. This example starts with a collision-free state, and resolves all penetrations until collision-free at each simulation step. It involves not only solid/cloth collisions but also extensive cloth/cloth collisions. Figure 7 and 8 highlights our method's ability to handle sharp geometric features. A sheet interacts with a bed of many spikes, with each spike being represented by four triangles. We tested three different scenarios: a flat sheet falling down onto the spikes, initial penetrations being resolved, and moving the spikes with the sheet covering them. The DID and the response algorithm works robustly and the simulated result is free of penetration.

TABLE I: Performance data for examples in Fig.1, 6 , 7 and 8.

Fig.	#vertices cloth/solid	sim. dura.	exec. time
Fig 1	16.5k/0	4.4s	9,068s
Fig 6	26.2k/6.2k	6.4s	9,171s
Fig 7	6.5k/1.7k	3.3s	1,241s
Fig 8	6.5k/1.7k	6.3s	4,511s

Table I shows the computation time for these examples. We have witnessed that majority of the penetrations were resolved in no more than ten detect-correct-detect iterations. As shown in previous experiments, the workload of ten rounds of DID is equal to one CCD. Therefore we believe using the proposed method beats the popular CCD based collision response in these experiments.

VII. Conclusion

We have presented a DID-based response to untangle existing penetrations. When used in context of simulating complex deformable surfaces, it is a competitive alternative to the popular CCD-based approach. The core of the method is to dynamically specify repulsive normal directions. Please be advised in many circumstances surface orientation is either explicitly given or implied in the context (e.g. closed surfaces), and we should not be blind to that. Taking advantage of predefined repulsive normals will save great effort on contour construction and the legal/illegal determination.

An obvious advantage of DID-based over CCD-based approach is that simulator can tolerate the existence of insignificant penetrations at each step, as also being highlighted in [8]. In real production, the artist can opt, in some designated steps, for incomplete response, as long as the intersecting regions do not expand too much and the rendered images do not suffer from visual artifacts.

Limitations. The proposed method also has several limitations. One major problem is that we did not give an effective solution to handle intersection contours containing loop-vertices. Another problem is that the BB contour partitions a surface only if the surface has one boundary. Some models in real production (e.g. T-shirt) have multiple boundaries, and the BB contour is no longer a partition contour. Several tricks can be adopted to avoid dealing with the BB case. Due to the absence of sophisticated friction and contact handling in our simulator, applying the proposed method to simulate more complex scenes like clumped cloth involving massive self-collisions is not easy at this moment. This certainly will be a direction of our future work.

Acknowledgment

This work is supported by NSF China under the grants #61379096, #61233001, and Finnish TEKESs project SoMa2020: Social Manufacturing (2015-2017).

References

[1] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," in *EG Workshop on Computer Animation and Simulation*, 1997, pp. 177–189.

[2] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 594–603, 2002.

[3] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous Contact Mechanics," *ACM Trans. Graph.*, vol. 28, no. 3, 2009.

[4] S. Ainsley, E. Vouga, E. Grinspun, and R. Tamstorf, "Speculative parallel asynchronous contact mechanics," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 151:1–151:8, 2012.

[5] D. Harmon, D. Panozzo, O. Sorkine, and D. Zorin, "Interference aware geometric modeling," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 137:1–137:10, 2011.

[6] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 862–870, 2003.

[7] M. Wicke, H. Lanker, and M. Gross, "Untangling cloth with boundaries," in *Proc. of Vision, Modeling, and Visualization*, 2006, pp. 349–356.

[8] P. Volino and N. Magnenat-Thalmann, "Resolving surface collisions through intersection contour minimization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1154–1159, 2006.

[9] J. Ye and J. Zhao, "The intersection contour minimization method for untangling oriented deformable surfaces," in *Proc. Symp. Computer Animation*, 2012, pp. 311–316.

[10] S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles," in *Proc. Symp. Interactive 3D Graphics and Games*, 2008, pp. 61–69.

[11] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 544–557, 2009.

[12] N. K. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M. C. Lin, and D. Manocha, "Interactive collision detection between deformable models using chromatic decomposition," in *SIGGRAPH*, 2005, pp. 991–999.