Fast Continuous Collision Detection using Parallel Filter in Subspace

Chen Tang*
Graphics and Interaction Lab
Peking University

Sheng Li†
Graphics and Interaction Lab
Peking University

Guopin Wang‡
Graphics and Interaction Lab
Peking University

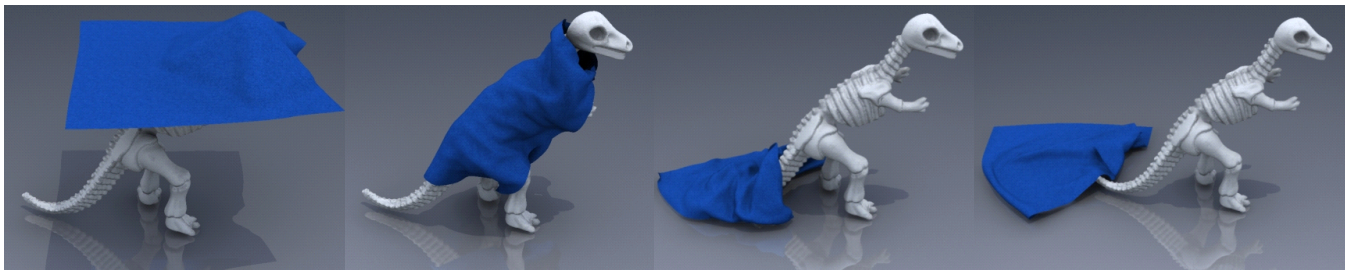


Figure 1: *Clothbone benchmark: a piece of cloth (108K) drops onto a bone of dinosaur (47K) causing large number of self-collision and contact with the bone. The scene consists of 90 frames. We test self-collision of the cloth and collision between cloth and dinosaur. Our method removes 99% of redundant elementary tests and gains almost 6.3x speedup compared with previous approaches on the elementary tests.*

Abstract

In this paper, we present a novel fast Continuous Collision Detection (CCD) method using SIMD capacity of CPU and idea of dimension reduction. We apply a parallel linear filter culling performed in one-dimensional subspace followed by a parallel planar filter culling performed in two-dimensional subspace before each elementary test, which simultaneously and conservatively tests the relative motion of each primitive pairs in various selected subspace. CPU's SIMD capacity is utilized for parallelizing the projection and filtering process in each subspace. Parallel filter culling in subspace removes a large amount of redundant elementary tests with low cost, and improves the overall performance of collision query. We demonstrate the advantages of our approach when comparing with previous alternatives in various dynamic scenes as benchmarks. In experiments, we observe up to 99% removal of false positives, and a huge magnitude of speed improvement on elementary tests (over 3x). Since our method only correlates the elementary test, it is scalable and can be easily integrated with various available single or multicore CPU based CCD algorithm. In addition, the performance of our method is less sensitive to varying step time.

Keywords: collision detection, deforming filter culling, SIMD, subspace

1 Introduction

Continuous Collision detection (CCD) is a crucial and ubiquitous step in physics based animation, haptic rendering and motion planning. CCD focuses on finding out the first *time of contact* (TOC) and respective contact position of primitive pairs. Generally speaking figuring out exact contact information of two deforming triangles needs 6 vertex-face (VF) and 9 edge-edge (EE) elementary tests. Each elementary test requires solving a cubic equation [Provot 1997] which is the most expensive step in CCD. On the other hand, CPU's SIMD capacity like Intel® SSE/SSE2/SSE3 and AMD® 3DNow! is widely used in various fields and brings remarkable performance improvement, but it's still not fully exploited in CCD. In this paper we provide a novel SIMD based parallel acceleration kernel for CCD which removes a large amount of unnecessary elementary tests with low filtering cost.

Accelerating the performance of CCD especially for deformable objects has been extensively studied in the last decade. Most of them use a bounding volume hierarchy (BVH) combined with self-collision detection and culling method, to obtain the Potential Collision Set (PCS) which contains potential collided triangle pairs. Then the low culling approaches are applied in the purpose of removing all the redundant elementary tests among primitive pairs in PCS. After that we obtain *Potential collided feature pairs (VF and EE pairs)* and continue pairwise elementary test for each of them. However, the BVH and self-collision culling is not so effective in CCD especially for deformable objects, which results in very high false positives (above 95% or more). And the computational cost of elementary test is considerably high. Therefore most of the query time is spent on elementary tests. Some methods try to use a low cost approximation as a filter before performing each elementary test such as primitive based bounding volume culling [Hutter and Fuhrmann 2007] [Curtis et al. 2008], and non-penetration filter culling [Tang et al. 2010]. Plus, CPU's SIMD capacity can also be utilized to improve the performance of BVH (SIMD k-DOP) on traverse and refitting, and primitive based bounding volume test [Tang et al. 2010]. This approach is proved effective to reduce the average cost of elementary test. The whole process is illustrated in Fig.2.

Problem: Intuitively, primitive based bounding volume culling is simple and efficient, but it bounds the whole moving trajectory which is over conservative especially for the simulation with large step time. Non-penetration filter removes those VF and EE pairs when they are never coplanar during the time interval. Nevertheless, this culling seems also over conservative since even VF or EE reaches coplanarity at certain moment t where $t \in [0, 1]$ they may

*e-mail: tangchen.cs.pku@gmail.com
†e-mail:lisheng@pku.edu.cn (corresponding author)
‡e-mail:wgp@pku.edu.cn

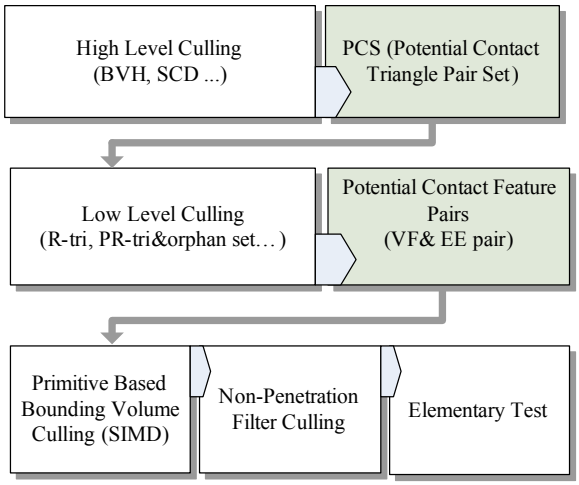
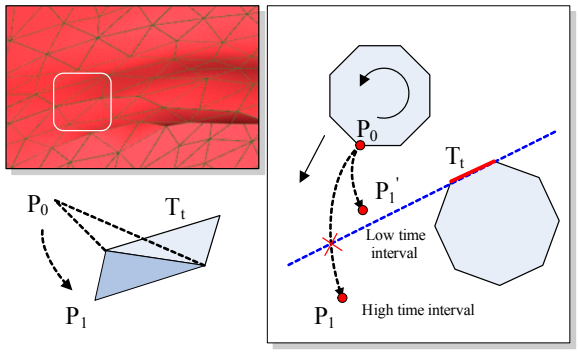


Figure 2: *General deformable CCD flow [Tang et al. 2010].*

still miss colliding with each other, eg. vertex may penetrate the plane of the triangle through outside of the triangle region. Fig.3(a) gives an example about this. And the filer process is difficult to be parallelized, this brings relatively high cost and restrains the overall performance. In addition, both primitive bounding volume culling and coplanarity culling are seriously sensitive to the simulation step time. See Fig.3(b): as the time interval increases, there are more probabilities for certain primitive to penetrate the the plane of the other primitives, which may conduct severe drop of performance when the step time increases.



(a) Penetration on self-collision feature pair

(b) Penetration relies on step time

Figure 3: *Various limitation on traditional filter method. (a) Fail inside test: Adjacent vertex P moving from P_0 to P_1 penetrates the plane of triangle T_t from outside of T_t. (b) Sensitivity on step time: P_0 to P_1 indicates the moving trajectory of the vertex P within a long time interval while P_0 to P_1' indicates the moving trajectory with a shorter step time.*

Main Results: In this paper we propose a novel efficient parallel method for continuous collision detection problem performed by SIMD capacity of CPU and dimension reduction. We found the removal for redundant elementary test is more efficient and feasible in subspace, and CPU's SIMD capacity can be fully exploited to parallelize the process in each subspace. We present a fast parallel linear filter in one-dimensional subspace considering relative motion on each primitive pair (VF and EE pair), and a conservative parallel planar filter test in two-dimensional subspace considering relative motion on each vertex-edge(VE) pair of those primitive pairs for a further culling. CPU's SIMD capacity is utilized for parallelizing projection and filtering in subspace. Parallel filter test in subspace removes a large amount of false positives and elementary tests with low cost, and improves the overall performance of collision query. We gain a huge magnitude of speed improvement on elementary tests (over 3x) compared to previous ones (Fig.1). Our method has high scalability and can be easily integrated with available single or multicore CPU based CCD algorithm. Plus, the performance of our method maintains stable and outstanding performance in various dynamic scenes with no obvious restraint, and is less sensitive to varying step time.

Organization: The rest of paper is organized as follows. First we present the background (Section 2), and an overview about the motivation and principle of parallel filter method proposed by us (Section 3). Next we give an explicit and elaborate description of the parallel linear filter (1D reduced filter, Section 4) and the parallel planar filter (2D reduced filter, Section 5) respectively. Finally we exhibit the experimental statistics and analysis on the overall performance (Section 6).

2 Background

Continuous Collision Detection (CCD) has been extensively studied in various fields including robotics, computational geometry and simulation in the last decade. Approaches for rigid bodies, articulated system and deformable objects involve algebraic equation solving [Canny 1984] [Kim and Rossignac 2003] [Redon et al.], swept volume formulations [Abdel-Malek et al. 2006], adaptive bisection approach [Redon et al. 2002], kinetic data structures (KDS) [Agarwal et al. 2001] [Kim et al. 1998] [Kirkpatrick et al. 2000], conservative advancement [Lin and Canny 1993] [Mirtich 1996] [Zhang et al. 2006] [Zhang et al. 2007] [Tang et al. 2009b], recent deformable CCD approach [Hutter and Fuhrmann 2007] [Curtis et al. 2008] [Tang et al. 2009a] [Tang et al. 2010], GPU and multi-core CPU architecture [Govindaraju et al. 2005] [Sud et al. 2006] [Lauterbach et al. 2010] [Kim et al. 2009].

Deformable CCD needs to figure out the first time of contact between pairwise features, including vertex-face (VF) and edge-edge (EE) elementary tests, which always reduces to solve a cubic equation [Provot 1997]. Due to the complexity of DOF (degree of freedom) of primitives of surface, eg. cloth simulation [Volino and Thalmann 1994] [Bridson et al. 2002], traditional culling approach performs poor on deformable CCD and produce large fraction of false positives. Most recent deformable CCD follows the order of high level culling, low level culling, filter culling and finally elementary test.

High level culling performs triangle-level overlap tests for the purpose of reducing the potential collided pair-wise set (PCS), the most common used approach to build a bounding volume hierarchy (BVH) on the object, and test the BVH against itself. Common bounding volume choices include spheres, axis aligned bounding boxes (AABB), oriented bounding boxes (OBB), and discrete oriented polytopes (k-DOP). To reduce overlap tests in BVH traversal, self collision detection (SCD) method are used to remove large regions with no collided triangle pairs, such as curvature tests [Volino and Thalmann 1994], normal bounds [Provot 1997] [Grinspun and Schroder 2001], its extension on continuous collision detection [Tang et al. 2009a], normal trees [Schvartzman et al. 2009], contour self-intersection [Schvartzman et al. 2010], SCD for reduced deformable models [Barbič and James 2010].

Low level culling removes redundant primitive pairs (VF or EE pair)and reduces elementary tests from PCS. These involve bounding volumes of the primitives to avoid performing elementary tests between different features [Hutter and Fuhrmann 2007] [Curtis

et al. 2008], representation triangle (R-triangle) using masking schemes to remove the redundant elementary tests [Curtis et al. 2008], procedural representation triangles (PR-triangle) for feature pair removal of non-adjacent triangles and orphan set for feature pair removal of adjacent triangles [Tang et al. 2009a], non-penetration filter culling which reduce both false positives and elementary tests by using coplanarity condition of elementary test [Tang et al. 2010]. Low level culling correlates feature pairs and elementary tests only, with no assumption on hierarchy of CCD or previous culling stages. A general low culling process order before pairwise elementary test follows the sequence of removing redundant feature pair, primitive based bounding volume culling, filter culling.

3 Motivation and Principle

In this section, we introduce the notation used in the rest of the paper and give an overview of our approach.

Notation: We use the symbols P , E and T to represent vertices, edges, and triangles, respectively. We denote the time interval, in which we perform CCD, to be $[0, 1]$. We use t_0 and t_1 to denote the start and end moments of the time interval. Assume we know the start and end positions of each vertex during each time interval. The position of each vertex between time steps, is defined as interpolation function $F(t)$. We use symbols P_t, E_t, T_t to denote the positions of specific vertex, edge and triangle at a certain time t where $t \in [0, 1]$, and symbols a_t, b_t, c_t to denote the corresponding vertices of the edge and triangle at time t, eg. $E_t = a_t b_t, T_t = \triangle a_t b_t c_t$. Most available CCD algorithms simplify $F(t)$ as linear interpolation, so that each elementary test is reduced to solve a cubic equation [Provot 1997].

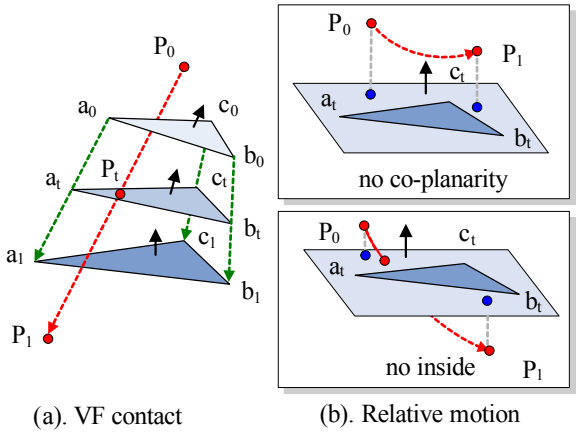


Figure 4: *VF elementary test. (a) shows the VF contact in object space. (b) shows the vertex's relative trajectory in respect to the triangle when it misses contacting the triangle (Up Image: fails the coplanarity condition. Bottom Image: fails the inside condition.)*

Each elementary test can be actually broken into two parts: *coplanarity test* and *inside test*. Coplanarity test figures out the exact contact time t by finding the root of a cubic equation. Inside test checks the validity of t by testing whether the contact position is inside the triangle for VF test or inside each edge for EE test. In order to acquire high culling efficiency, we must consider the relative motion of feature pairs and bounds both coplanarity and inside conditions tightly. However, although the movement of primitives is simplified as linear transformation, the relative motion trajectory of vertex is always considerably complicated. Fig. 4(a) shows the VF contact in object space and Fig. 4(b) shows the vertex's relative trajectory in respect to the triangle when it misses contacting the triangle. It's clear that bounding the elementary test in $\mathbb{R}^3$ is difficult. In this paper we exploit bounding the relative motion of feature pairs in subspace. We found the process in subspace is more feasible and bounds both coplanarity and inside test tightly. Furthermore, the process is very suitable for utilizing CPU's SIMD capacity. We can complete the projection and culling process of various different subspaces independently and simultaneously in order to obtain high culling ratio.

Dimension reduction means representing a vector in high dimensional space $u \in \mathbb{R}^n$ with a corresponding vector in a much lower dimensional space $r \in \mathbb{R}^m, m < n$. While necessarily approximate, such a representation can be quite useful for compressing redundant information in u. To transform between the two spaces we need a projection operator $\psi : u \mapsto r$. Reduction is an increasingly important technique in computer graphics and has been extensively used for optimization in a wide range of problems [Barbič and James 2005] [Treuille et al. 2006] [Barbič et al. 2009] [Barbič and James 2010].

In this paper, we solve CCD problem in subspace of $\mathbb{R}^3$. We orthogonally project primitives in elementary test (vertex, edge, triangle) from $\mathbb{R}^3$ into one of its subspaces which is actually a plane (or a line) through the origin. The projected position is represented in a two-dimensional (or one-dimensional) Euclidean space defined on the plane (or on the line). We call the whole process *projection* from $\mathbb{R}^3$ to lower dimensional space ($\mathbb{R}^2$ or $\mathbb{R}^1$), we use ψ to denote the projector. Orthogonal projection has the property that if deforming vertex $p_t \in \mathbb{R}^3$ does linear deformation, its projection $\psi(p_t)$ in $\mathbb{R}^2$ or $\mathbb{R}^1$ also does linear deformation. Our method is based on the theorem below:

Theorem 1. ***Subspace Culling:*** *For any specific projection ψ to subspace, if there's no contact for two deforming primitives in the subspace defined by ψ during a time interval, there's no contact between the two primitives in their original space.*

We provide a fast parallel deforming linear and planar filter following Theorem 1 for the purpose of removing redundant elementary tests in 1D and 2D subspace. In the following we give an explicit and elaborate representation of the parallel linear filter (1D filter) and the parallel planar filter (2D filter).

4 Parallel Linear Filter (1D Filter)

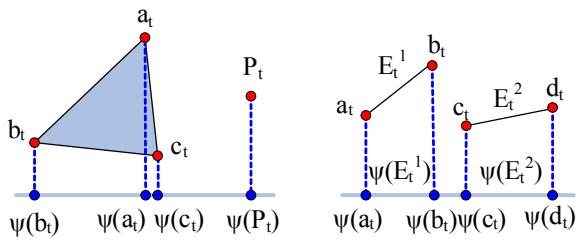
Parallel linear filter is based on the extension of Theorem 1 in one-dimensional subspace:

Corollary 2. ***Linear Bounding Test:*** *For a specific one-dimensional subspace with the projection operator $\psi : \mathbb{R}^3 \mapsto \mathbb{R}^1$,*

Reduced VF bounding: For a vertex P_t and a triangle T_t in $\mathbb{R}^3$, if $\psi(P_t)$ is always on the left (or right) of each vertex of $\psi(T_t)$ during the time interval, there's no contact between P_t and T_t (Fig.5(a)).

Reduced EE bounding: For two edges E_t^1, E_t^2, its projection in $\mathbb{R}^1$ is $\psi(E_t^1)$, $\psi(E_t^2)$, if the two vertices in $\psi(E_t^1)$ is always on the left (or right) of the two vertices in $\psi(E_t^2)$ during the time interval, there's no contact between E_t^1, E_t^2 (Fig.5(b)).

Because each primitive in $\mathbb{R}^3$ does linear deformation in preassumption, the projection of the corresponding vertices in one-dimensional subspace moves with constant velocity. Therefore for any two vertices in the one-dimensional subspace, if one vertex stays at the same side of the other vertex at the start and end of the interval respectively, there's no contact during the entire time interval.



(a). VF test of linear filter (b). EE test of linear filter

Figure 5: *Linear filter in one-dimensional subspace.*

VF Test of Linear Filter: *Given a specific one-dimensional subspace with the projection operator $\psi : \mathbb{R}^3 \mapsto \mathbb{R}^1$, a moving vertex $P_t, P_t \in \mathbb{R}^3$ and a deforming triangle $T_t = \triangle a_t b_t c_t \in \mathbb{R}^3$, supposing their projection through ψ is P'_t, a'_t, b'_t, c'_t respectively. If the scaler values: A, B, C, D, E, F have the same sign, there's no contact between P_t and T_t during the interval.*

$$A = P'_0 - a'_0, B = P'_1 - a'_1 \quad (1)$$

$$C = P'_0 - b'_0, D = P'_1 - b'_1 \quad (2)$$

$$E = P'_0 - c'_0, F = P'_1 - c'_1 \quad (3)$$

EE Test of Linear Filter: *Given a specific one-dimensional subspace with the projection operator $\psi : \mathbb{R}^3 \mapsto \mathbb{R}^1$, two deforming edges $E_t^1 = a_t b_t$ and $E_t^2 = c_t d_t \in \mathbb{R}^3$, supposing their projection through psi is P'_t, a'_t, b'_t, c'_t respectively. If the scaler values: A, B, C, D, E, F, G, H have the same sign, there's no contact between E_t^1 and E_t^2 during the interval.*

$$A = a'_0 - c'_0, B = a'_1 - c'_1 \quad (4)$$

$$C = a'_0 - d'_0, D = a'_1 - d'_1 \quad (5)$$

$$E = b'_0 - c'_0, F = b'_1 - c'_1 \quad (6)$$

$$G = b'_0 - d'_0, H = b'_1 - d'_1 \quad (7)$$

The test process in each subspace is independent and highly parallel. If there's no contact between the feature pair in any selected subspace, there's no contact in original space. We use SIMD k-DOP [Tang et al. 2010] as basic Bounding Volume structure in BVH. Unlike traditional k-DOP, the minimum and maximum coordinates of primitive along each axis in SIMD k-DOP is stored in SIMD vector so that we can use the SIMD instruction to parallelize k-DOP operation (merging, intersection test) and accelerate the BVH traversal and refitting performance [Tang et al. 2010]. 18-DOP is proved effective in CCD [Tang et al. 2009a]. Here we use SIMD 16-DOP which only contains 8 different axes since most CPU's SIMD instructions wrap 4 floating points. Each vertex also maintains a group of coordinates of its start and end position along each k-DOP axis. The coordinates of vertex are stored as SIMD vector and used for refitting BVH and computing primitives' projection in linear and planar filters. eg. A possible projected coordinate of vertex P_t along each axis can be $\psi_i(P_t) = \{x_t, y_t, z_t, x_t + y_t, x_t + z_t, y_t + z_t, x_t - y_t, y_t - z_t\}, i = 1, 2 \ldots 8$. Note that, in SIMD k-DOP or SIMD vector of vertex the order of elements inside each SIMD vector is not random (Specific order constraint is discussed in Section 5.1). This is different from traditional k-DOP and SIMD optimized k-DOP in [Tang et al. 2010]. Since SIMD filtering process is highly sensitive to the cost of data loading and projection, we use each axis in SIMD k-DOP as selected one-dimensional subspace in parallel linear filter so that we can reuse the wrapped coordinates of vertex. In parallel linear filter, the scaler values above in each subspace are calculated and compared simultaneously according to the SIMD instructions of CPU.

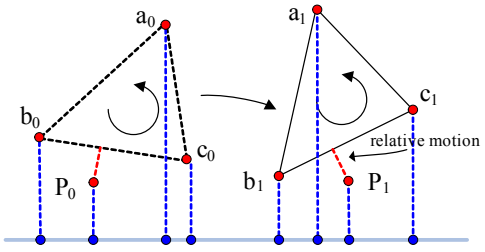
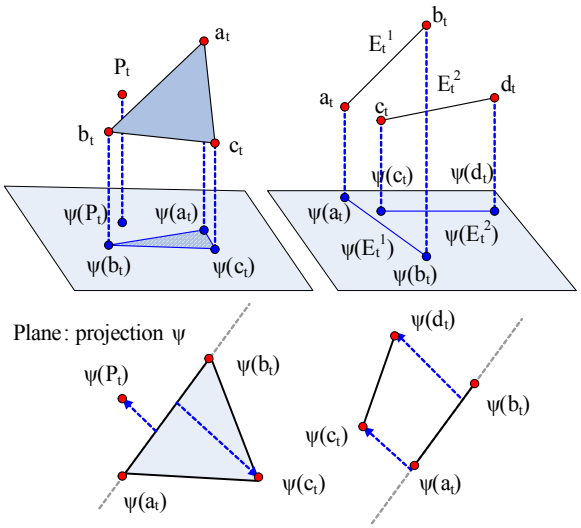


Figure 6: *Relative motion between vertex and edge.*

5 Parallel Planar Filter (2D Filter)

Only projection along fixed axis like linear filter is difficult to remove those accompanied feature pairs who are moving together with rotation or have complicated motion trajectory. The drawback can be ameliorated by considering the relative motion of vertex-edge(VE) pair contained in primitive pairs and using a parallel planar filter test as a post culling stage after performing the parallel linear filter test (Fig.6). The parallel planar filter is based on the extension of Theorem 1 in two-dimensional subspace as follows:



(a). Conservative planar VF filter (b). Conservative planar EE filter

Figure 7: *Conservative Planar Filter Test in two-dimensional subspace.*

Corollary 3. ***Conservative Planar Filter Test:*** *For a specific two-dimensional subspace with the projection operator $\psi : \mathbb{R}^3 \mapsto \mathbb{R}^2$,*

1. Reduced VF bounding: for a moving vertex P_t and a deforming triangle T_t, if there exists one edge of $\psi(T_t)$, $\psi(P_t)$ is always on the same side of $\psi(T_t)$ separated by the edge during the entire time interval, there's no contact between P_t and T_t (Fig.7(a)).

2. Reduced EE bounding: For two edges E_t^1, E_t^2, if there exists one edge $E'_t = a_t b_t$ of them that $\psi(a_t)$ and $\psi(b_t)$ is always on the same side of the other edge $\psi(E''_t)$ during the time interval, there's no contact between E_t^1 and E_t^2. (Fig.7(b))

The conservative planar filter test above bounds both coplanarity and inside test tightly with low cost.

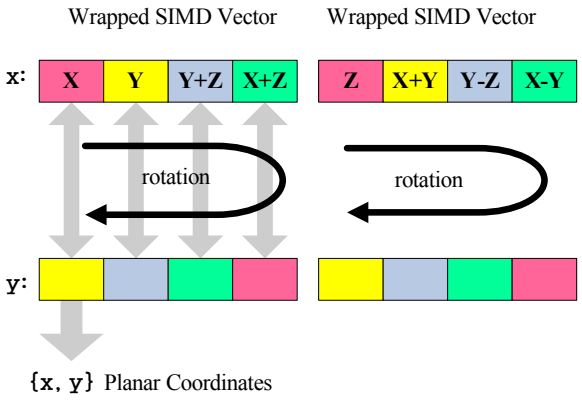


Figure 8: *Projection and data wrapping: The SIMD vectors on the top are the 1D coordinates used in the linear filter. The planar coordinates is obtained directly from the original SIMD vector. The X and Y of planar coordinates come from the original and the rotated SIMD vector respectively. The illustration gives a possible order of projection result inside the SIMD vector which creates 8 independent planes.*

5.1 Projection and Data Wrapping

In order to obtain planar coordinates in each 2D subspace,we reuse the group of 1D projected coordinates of each vertex. We rotate the two SIMD vectors of each vertex respectively and combine them with the original ones, to create the wrapped planar coordinates in each subspace (Fig.8). The two components of each planar coordinate come from the original and newly created SIMD vector respectively. Note that the order of components in SIMD vector is important. Only few specific orders are able to produce 8 different planes (subspace) by such rotation. Fig.8 gives a possible projection and its corresponding order. Rotating the SIMD vector is very efficient using CPU's SIMD instructions. So we can obtain and wrap the planar coordinates effectively from the wrapped 1D coordinates of each vertex.

5.2 Planar VE Test

In Corollary 3, we must decide whether a vertex will penetrate a line during the time interval on plane. Exact VE (vertex-edge) penetration test is costly. We use the following conservative planar VE test to judge whether there's penetration between vertex and edges on plane.

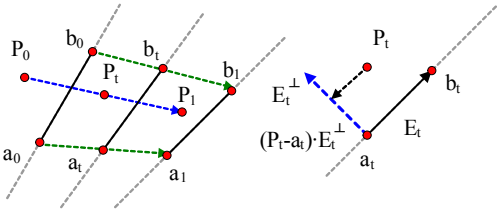
Collinearity Theorem for Planar VE Test: *For a given plane, and an edge $E_t = a_t b_t, a_t, b_t \in \mathbb{R}^2$ and a vertex $P_t \in \mathbb{R}^2$ defined by the start and end positions on one plane during the interval [0, 1], these positions are linearly interpolated in the interval with respect to the time variable, t (Fig.9(a)). If the three scalar values: A, B, C have the same sign, E_t and P_t will not be collinear during the interval. Furthermore, if the corresponding sign is positive, P_t is always inside the left half space of E_t during the interval, otherwise if the corresponding sign is negative, P_t is always inside the right half space of E_t during the interval.(Fig.9(b))*

$$A = (P_0 - a_0) \cdot (b_0 - a_0)^{\perp} \quad (8)$$

$$B = (P_0 - a_0) \cdot (b_1 - a_1)^{\perp} + (P_1 - a_1) \cdot (b_0 - a_0)^{\perp} \quad (9)$$

$$C = (P_1 - a_1) \cdot (b_1 - a_1)^{\perp} \quad (10)$$

where $\perp$ denotes the 90 degree counter-clock wise rotation of a vector. eg. assuming a = $\{a_x, a_y\}$, b = $\{b_x, b_y\}$, there is $a \cdot b^{\perp} = -a_x * b_y + a_y * b_x$.



(a). Planar moving vertex P_t and deforming edge E_t ($a_t b_t$)

(b). $(P_t-a_t)\cdot E_t^{\perp}>0$: left half space
$(P_t-a_t)\cdot E_t^{\perp}<0$: right half space
$(P_t-a_t)\cdot E_t^{\perp}=0$: collinear

Figure 9: *Planar VE test. (a)P_t is a moving vertex and $a_t b_t$ is a deforming edge on plane during the time interval. (b)The sign of $(P_t - a_t) \cdot E_t^{\perp}$ is used to indicate which half space the planar vertex P_t stays.*

Proof. We define $\delta P = P_1 - P_0, \delta a = a_1 - a_0, \delta b = b_1 - b_0$, For the moving vertex $P_t = P_0 + \delta P * t$ and deforming edge $E_t = a_t b_t$ where $a_t = a_0 + \delta a * t, b_t = b_0 + \delta b * t$. Which half space P_t stays depends on the sign of projection along the normal of E_t (Fig.9(b)). The projected distance is:

$$\begin{aligned}(P_t - a_t) \cdot \overrightarrow{E_t}^{\perp} &= (P_t - a_t) \cdot (b_t - a_t)^{\perp} \\ &= ((P_0 - a_0) + (\delta P - \delta a) * t) \\ &\quad \cdot ((b_0 - a_0) + (\delta b - \delta a) * t)^{\perp} \\ &= (P_0 - a_0) \cdot (b_0 - a_0)^{\perp} \\ &\quad + ((P_0 - a_0) \cdot (\delta b - \delta a)^{\perp} \\ &\quad + (\delta P - \delta a) \cdot (b_0 - a_0)^{\perp}) * t \\ &\quad + (\delta b - \delta a) \cdot (\delta P - \delta a)^{\perp} * t^2 \quad (11)\end{aligned}$$

On the other hand, Equation(11) can be represented as

$$(P_t - a_t) \cdot \overrightarrow{E_t}^{\perp} = A * B_0^2(t) + B * B_1^2(t) + C * B_2^2(t) \quad (12)$$

where $B_i^2(t)$ is the i^{th} basis function of Bernstein polynomials. That means $B_0^2(t) = (1-t)^2, B_1^2(t) = 2*t*(1-t), B_2^2(t) = t^2$

By comparing the coefficient of t in Equations(11)-(12), we obtain the representation of symbols A, B and C in Equations(8)-(10)

The scaler values A, B, C in Equations(8)-(10) are actually the control points of quadratic curve created by Equation(11), indicates the relative distance between vertex and the edge. If A, B, C maintain the same sign, P_t can not be collinear with E_t depending on the convex hull property associated with control points of the Bernstein basis. The conservative approximation bounds the result tightly and is more efficient compared with directly solving the root of the equation and checking its validity.

5.3 Planar VF Filter

Both planar VF and EE filter test can be reduced to at most two planar VE tests. We use SIMD instruction to parallelize the projection (Section 5.1) and filtering process in each subspace. Planar VF filter test is as follows:

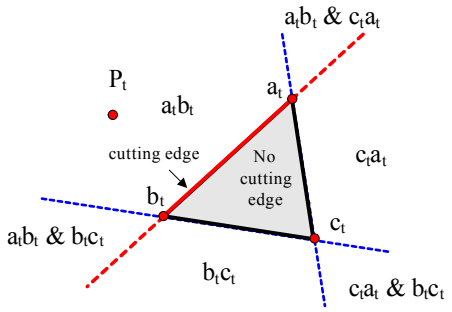


Figure 10: *Space partition by the edges of triangle. The figure shows the space partition by the three edges of triangle and the cutting edge which separates the vertex with the triangle .*

Process: After projection and data preparation, we do filter culling process in different subspace simultaneously using SIMD instructions. Assume the projection of vertex P_t and triangle T_t is $\psi(P_t)$ and $\psi(T_t)$ in planar VF test. Firstly we decide whether the triangle will turn over during the time interval by selecting one vertex and its opposite edge from the triangle and testing VE penetration using the method described in Section 5.2. If the triangle never turns over, we continue to figure out the cutting edge (the edge which separates the triangle and vertex at the start moment) by comparing the sign of scaler value A of VE test between P_t and each edge of T_t, with the sign of scalar value A of the triangle. In most cases there's only one cutting edge existing for a specific vertex and triangle pair, but occasionally two. Fig. 10 shows the space partition and each area's corresponding cutting edge. We select the first one we find. The whole process in one subspace is illustrated in Algorithm(1). In parallel implementation, we use a SIMD vector to trace the status. Only when all subspace routines return *true*, or at least one subspace routine returns *false* in Algorithm(1), the SIMD filter terminates and returns the result whether to go on executing the elementary test.

Algorithm 1 *Single Thread Planar VF Test*: For a given two-dimensional subspace, a moving vertex $P_t \in \mathbb{R}^2$ and a deforming triangle $T_t = \triangle a_tb_tc_t, a_t, b_t, c_t \in \mathbb{R}^2$ on the plane, we decide whether there's a potential penetration between the vertex and the triangle. We use the form $v \Rightarrow e$ to denote VE test between vertex v and edge e. $A(v \Rightarrow e) \sim C(v \Rightarrow e)$ indicates the results of Eq.8∼Eq.10 on the VE test between v and e

$A_0 \leftarrow A(a_t \Rightarrow b_tc_t)$
$B_0 \leftarrow B(a_t \Rightarrow b_tc_t)$
$C_0 \leftarrow C(a_t \Rightarrow b_tc_t)$
if A_0, B_0, C_0 have different sign **then**
  return true \\ *T_t turns over*
end if
for $E \leftarrow a_tb_t, b_tc_t, c_ta_t$ **do**
  \\ *test each edge of T_t respectively*
  $A_1 \leftarrow A(P_t \Rightarrow E)$
  if A_0 and A_1 have different sign **then**
    $B_1 \leftarrow B(P_t \Rightarrow E)$
    $C_1 \leftarrow C(P_t \Rightarrow E)$
    if A_1, B_1, C_1 have the same sign **then**
      return false \\ *no penetration found, should be culled*
    end if
    return true \\ *P_t penetrates E*
  end if
end for
return true \\ *no cutting edge found*

5.4 Planar EE Filter

The process of EE filter test follows the same way. Firstly we figure out the cutting edge (the edge that two endpoints of the other edge are always on the same side of it at the start moment). Then we do VE test between the two vertices and the cutting edge to see if there's penetration between them during the time interval as illustrated in Algorithm (2).

Algorithm 2 *Single Thread Planar EE Test*: For a given two-dimensional subspace, two deforming edges $E_t^1 = a_t^1b_t^1, E_t^2 = a_t^2b_t^2, a_t^i, b_t^i \in \mathbb{R}^2, i = 1, 2$ on the plane, we decide whether there's a potential penetration between E_t^1 and E_t^2.

for $i \leftarrow 1$ to $2, j \leftarrow 3 - i$ **do**
  \\ *test two edges $a_t^1b_t^1, a_t^2b_t^2$ respectively*
  $A_0 \leftarrow A(a_t^j \Rightarrow a_t^ib_t^i)$
  $A_1 \leftarrow A(b_t^j \Rightarrow a_t^ib_t^i)$
  if A_0 and A_1 has the same sign **then**
    $B_0 \leftarrow B(a_t^j \Rightarrow a_t^ib_t^i)$
    $C_0 \leftarrow C(a_t^j \Rightarrow a_t^ib_t^i)$
    if A_0, B_0, C_0 have different sign **then**
      return true \\ *a_t^j penetrates $a_t^ib_t^i$*
    end if
    $B_1 \leftarrow B(b_t^j \Rightarrow a_t^ib_t^i)$
    $C_1 \leftarrow C(b_t^j \Rightarrow a_t^ib_t^i)$
    if A_1, B_1, C_1 have different sign **then**
      return true \\ *b_t^j penetrates $a_t^ib_t^i$*
    end if
    return false \\ *no penetration found, should be culled*
  end if
end for
return true \\ *no cutting edge found*

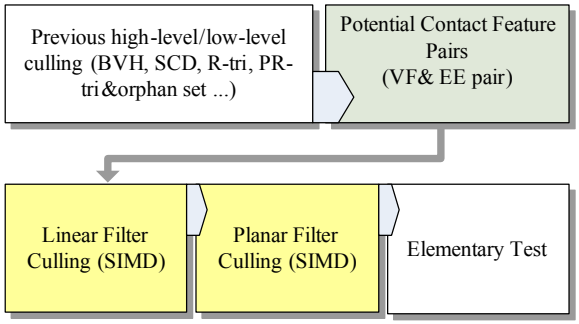


Figure 11: *New flow of CCD: After previous high-level/low level culling, we obtain potential contact feature pair set. Then for each feature pair, we perform the linear and planar filter culling respectively and finally elementary test.*

5.5 Overall Flow for CCD Algorithm

Our method makes no restriction or assumption about previous high-level/low-level culling. After all previous culling stages, we obtain the *Potential Contact Feature Pair Set*. For each feature pair, the parallel linear filter is applied as the 1^{st} level culling, and the parallel planar filter is performed as the 2^{nd} level culling. At last we do exact elementary test. The whole process is illustrated in Fig.11. Since our method is only concerned with the elementary test, it can accelerate most of current available CPU-based CCD algorithm. Compared with previous methods, our approach maintains

higher culling efficiency while relatively low cost, thus significantly improves the overall performance of CCD algorithm.

6 Result and Analysis

In this section we provide the implementation, theoretical and experimental analysis about our method compared against previous approaches on various dynamical scenes.

Implementation: We have implemented all the algorithms on a standard 2.3GHz Intel Duo Core machine with 2GB RAM on 32-bit Windows/XP platform. For clear and explicit comparison, we use a base implementation without filter culling. We use two level DT-BVH as basic hierarchy, 18-DOP as bounding volumes, Continuous Normal Cone (CNC) as high-level culling [Tang et al. 2009a], and representative triangle [Curtis et al. 2008] as low-level culling to remove all redundant feature pairs. The k-DOP operation and primitive BV test is accelerated by SSE2 instruction [Tang et al. 2010]. Linear and planar filters are also implemented by intrinsic SSE2 instructions. We use non-penetration filter [Tang et al. 2010] implementation available at[1] and standard UNC dynamics benchmark[2] for comparison. In order to highlight the acceleration effect for elementary test and exclude the influence of previous basic implementation, we only record the execution time and culling ratio of the last stage after we obtain the potential contact feature pairs.

We compare the performance of different CCD methods on the benchmarks Fig.1, Fig.12(a)∼Fig.12(d). Note that we give two different comparison versions on N-body (the surface has slight deformation). In N-body① we only record the inter-collision (collision between different objects), while in N-body② we make general collision query (involving self-collision). We provide two versions of comparison in order to highlight the performance of inter-collision and also overall collision query on slightly deformable objects, since sometimes it is difficult to pre-determine the deformation property of the surface (whether there's self-collision) in complex dynamic scenes. Our method performs well on both of the two choices.

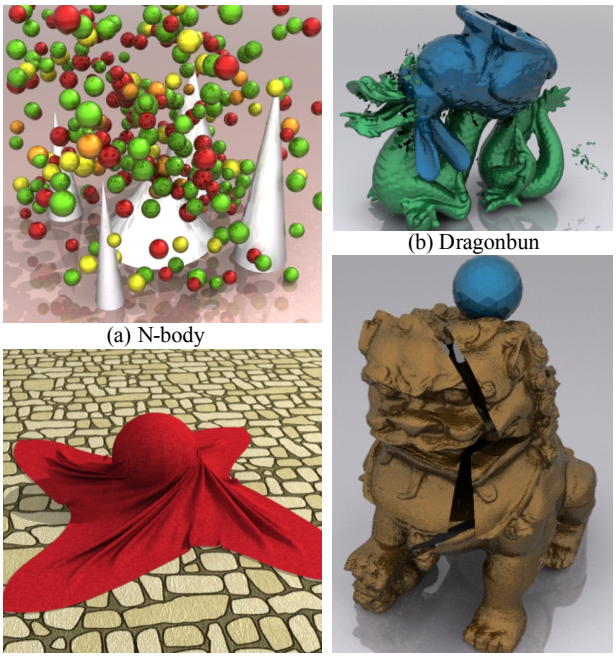
Performance Analysis: The principle of filter culling is to use low costly culling tests to replace part of the high costly elementary tests in order to acquire overall performance advance. The performance of filter algorithm only has relations with its cost and culling efficiency. We use the proportion of cost reduction on elementary test after being optimized with the filter culling method ($C_{reduction}$) to measure the performance of a particular filter method. Suppose C_{elem} is the cost of a single elementary test, C_{filter} and C_{avElem} indicate the cost of filter test and the cost of average elementary test after optimized with the filter test respectively. α denotes the culling ratio.

$$\begin{aligned} C_{reduction} &= (C_{elem} - C_{avElem})/C_{elem} \\ &= (C_{elem} - (C_{filter} + (1-\alpha)C_{elem}))/C_{elem} \\ &= \alpha - \beta \quad where \quad \beta = C_{filter}/C_{elem} \end{aligned} \tag{13}$$

From Eq.13 we see, it's only when the culling ratio α is higher than culling cost proportion β can the filter culling method accelerate the overall performance of the elementary test. The larger the difference is, the more remarkable the acceleration effect appears. Next we will analyze the performance of the linear filter, planar filter, overall performance and the performance with different time intervals respectively.

[1] http://gamma.cs.unc.edu/DNF/

[2] http://gamma.cs.unc.edu/DYNAMICB



(a) N-body (b) Dragonbun (c) Cloth ball (d) Lion

Figure 12: *BenchMarks: (a)(146K, 96 frames) This benchmark consists of hundreds of moving balls that can be slightly deformable. They collide with each other and the ground. (b)(252K, 64 frames) The bunny penetrates the dragon and breaks it into a high number of colliding pieces. (c)(92K, 94 frames) The cloth drops onto a ball with rotation and a high number of self-collisions. (d)(1.6M, 90 frames) The lion gradually breaks into a high number of colliding pieces.*

Parallel Linear Filter Performance: Table.1 shows the culling ratio comparison between combination of SIMD primitive based BV test and parallel deforming filter on various scenes. It's obvious that the culling efficiency of the parallel linear filter is much higher than that of SIMD primitive based bounding volume culling. Although the cost of the linear filter consisting of several substraction appears more expensive than the primitive based bounding volume overlapping test, the cost is considerably low compared with the elementary test. So the the difference between the cost proportion β in respect to the elementary test and the culling ratio α is large, which makes the parallel linear filter much more efficient than SIMD primitive based bounding volume test. Fig.13 illustrates the contrast of the overall performance of various combinations of filters and non-filtering elementary test. It's clear that the overall executed time of the final stage with parallel linear test is even less than the total executed time using the combination of SIMD primitive based bounding volume test and non-penetration filter culling method.

Parallel Planar Filter Performance: As shown in Table.1 you can easily find that the culling ratio of parallel planar filter seems lower than that of non-penetration filter. This is mainly because the culling ratio of the first step (parallel linear filter) is too high, which removes most of the redundant elementary tests before parallel planar filter test. It's clear from the table and Fig.13(a)-(c) that the overall culling ratio of parallel filter is much higher than that of the traditional combination, and parallel planar filter culling provide a further performance improvement on performance after the linear filter culling. In addition, the non-penetration has some restraint. See the contrast between Fig.13(b) and Fig.13(c). The

Table 1: *Culling Ratio Comparison*

Method	clothball	N-body① [5]	N-body② [6]	clothbone	dragon	lion
PrimBV [1]	53%	25%	26%	38%	37%	24%
linear [2]	95%	81%	96%	99%	90%	89%
NonPen [3]	77%	29%	82%	93%	55%	85%
planar [4]	63%	39%	39%	75%	41%	72%
PrimBV+NonPen	89%	46%	87%	96%	72%	87%
linear+planar	98%	89%	97%	99%	94%	97%

[1] SIMD Primitive BV test.
[2] Parallel linear filter
[3] Non-penetration filter.
[4] Parallel planar filter
[5] not include self-collision detection on each ball.
[6] include self-collision detection on each ball.

Table 2: *Performance Comparison on Executed Time (ms)*

Method	clothball	N-body①	N-body②	clothbone	dragon	lion
PrimBV+ NonPen [1]	326	1111	1946	188	680	23489
linear+planar [2]	65	329	513	30	221	1545
acceleration	5x	3.4x	3.8x	6.3x	3x	15x

[1] SIMD Primitive BV + Non-penetration filter.
[2] Parallel linear filter + parallel planar filter

balls in N-body are slightly deformable so that non-penetration filter may achieve high culling efficiency on self-collision, but the lack of motion coherence on N-body makes non-penetration filter perform poor on inter-collisions with low culling ratio (Table.1) and slows down the overall performance. But parallel planar filter method does not have such problem and performs well in both inter-collision and self-collision.

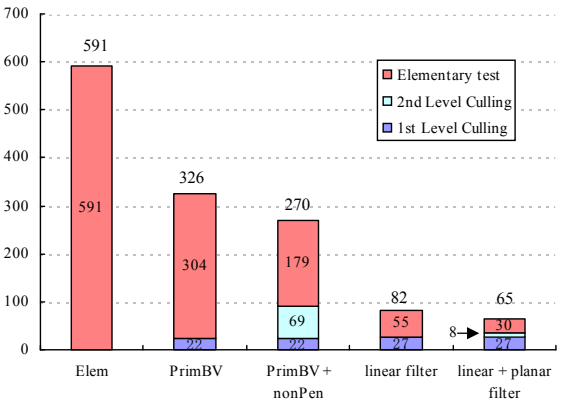
Overall Performance: Table.2 shows the overall speed improvement compared with traditional non-penetration filter and primitive based bounding volume culling. Combined with parallel deforming filter, we observed over 90% elementary test removal and at least 3x performance improvement.

Performance with Different Time Intervals: Table.3 and Table.4 illustrate the performance on N-body① (Collision between different objects) and clothball (self-collision) with varying step time. We simulate varying time interval by increasingly interpolating the original dynamic scenes. From Table.3, it's clear that the culling ratio of the non-penetration filter in N-body system is considerably

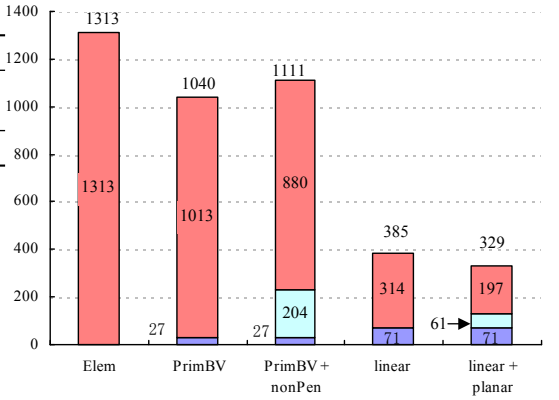
Table 3: *Performance on Nbody① with Varying Time Interval*

Method	1x [5]	1/2x	1/3x	1/4x	1/5x
PrimBV [1]	25%	38%	47%	53%	57%
linear [2]	81%	78%	78%	78%	78%
NonPen [3]	29%	32%	35%	38%	41%
planar [4]	39%	39%	40%	42%	43%
PrimBV+NonPen	47%	58%	65%	70%	74%
linear+planar	89%	87%	87%	87%	87%
speed of PrimBV+NonPen	1111ms	201ms	83ms	47ms	32ms
speed of linear+planar	329ms	83ms	41ms	26ms	19ms

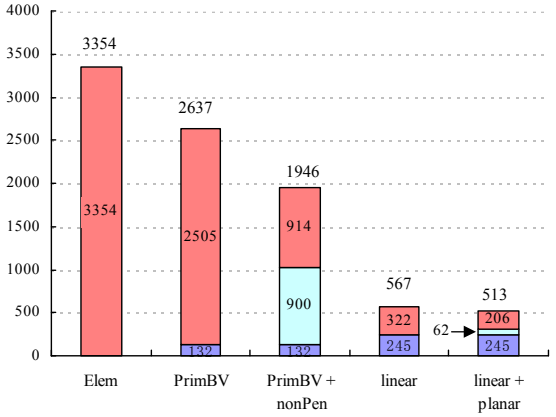
[1] SIMD Primitive BV test.
[2] Parallel linear filter
[3] Non-penetration filter.
[4] Parallel planar filter
[5] Time interval, 1x indicates the time interval of the original dynamic scene. $1/n$x means the model is increasingly interpolated with more frames, and the interval time between each frame is only $1/n$ of the original one.



(a) Executed Time for Cloth Ball



(b) Executed Time for N-Body (Not include self-collision)



(c) Executed Time for N-Body (Including self-collision)

Figure 13: *Overall performance and executed time of each step on various combination of filters.*

low especially when the time interval is large. This is mainly due to the fact that the motion coherence and connection of primitive pair in different objects is weak. As the time interval decreases, the culling ratio rises up. While, on the other hand, the culling ratio of parallel filter is less variable with varying time interval. When it comes to the self-collision (Table.4), the problem is not so obvious in the non-penetration filter, since there's strong coherence for feature pairs on the same surface. However, the overall culling ratio is still much lower than the parallel filter and more sensitive to the time interval. Therefore parallel filter seems more stable with varying time interval. Further more, from the executed time statistics

Table 4: *Performance on Clothball with Varying Time Interval*

Method	1x	1/2x	1/3x	1/4x
PrimBV	53%	65%	71%	75%
linear	95%	94%	93%	93%
NonPen	77%	85%	89%	91%
planar	63%	73%	77%	79%
PrimBV+NonPen	89%	95%	97%	98%
linear+planar	98%	98%	98%	98%
speed of PrimBV+NonPen	326ms	74ms	43ms	30ms
speed of linear+planar	65ms	29ms	21ms	17ms

in Table.3 and Table.4, it's apparent that our method is much faster than non-penetration method with any time interval.

Comparison with Other CCD Algorithm: There are lots of other CCD algorithms available which effectively removes a large fraction of redundant elementary test in high and low levels. *Orphan Set* [Tang et al. 2009a] is quite effective in removing redundant elementary tests between adjacent triangles. *Continuous normal cone* [Tang et al. 2009a] is widely used to remove large flat area from self-collision detection. *Representative triangles* [Curtis et al. 2008] can remove all the redundant elementary tests through pre-computed feature connectivity. *Bounded normal trees* [Schvartzman et al. 2009] bounds the normal trees more tightly and improves the culling ratio of self-collision detection. *Star-Contours* [Schvartzman et al. 2010] gives an effective means to solve the boundary problems in self-collision detection. Since filtering method is executed after all these high/low culling method and correlates elementary test only, it is effective to improve the overall performance by filtering the elementary test. [Tang et al. 2010] gave a elaborate comparison between filtering and non-filtering performance. You can also observe the huge improvement between filtering and non-filtering performance in Fig. 13. We provide a compensation for these available method.

Many other approaches try to use multi-core CPU [Tang et al. 2009c] or CPU and GPU hybrid architecture [Kim et al. 2009] to decompose tasks and put them into different cores (or prepare to send data to GPU). Since each core of CPU maintains independent SIMD capacity, SIMD based filter may also benefit such CCD algorithm and helps to remove unnecessary elementary tests with low cost.

7 Limitation

Low costly culling tests by parallel filter are used to replace part of high costly elementary tests in order to obtain overall performance improvement. The performance is dominated by the culling ratio and the cost of elementary test. So the advantage of our methods may not be so distinctive or even shrink when the cost of elementary test declines.

8 Conclusion and Future Work

In this paper we have proposed a novel fast parallel deforming filter culling method for continuous collision detection (CCD) problem performed by dimension reduction in subspace and exploiting SIMD capacity of CPU. We have presented a fast parallel linear filter considering relative motion on primitive pairs(VF and EE) and a parallel planar filter test performed after linear filter considering relative motion on each vertex-edge(VE) pair of those primitive pairs. CPU's SIMD capacity has been utilized for parallelizing the projection and filtering in subspace. filter culling in subspace removes a large amount of false positives and elementary tests with low cost, and improves the overall performance of collision query. By the combination of linear and planar filter, we get up to 99% removal of redundant elementary tests, and a huge magnitude of speed improvement on elementary tests (over 3x). Our method also has such characteristic that it can be integrated with currently available single or multicore CPU based CCD algorithm easily.

We will optimize our reduced filter and apply our methods to multicore CCD algorithm. We will also exploit high level culling algorithm such as solving self-collision detection problem in subspace in the future.

Acknowledgements

We would like to thank Dr. Tang Min for the useful discussions and UNC Gamma group for the benchmarks.

This research was supported by grant No.2010CB328002 from the National Basic Research Program of China, and by Nos.60703062, 60833007, 60925007 and 90915010 from National Natural Science Foundation of China. It was also supported by grants No.2009AA01Z324 from the National High Technology Research and Development Program of China.

References

ABDEL-MALEK, K., YANG, J., BLACKMORE, D., AND JOY, K. 2006. Swept volumes: fundation, perspectives, and applications. *International Journal of Shape Modeling 12*, 1, 87.

AGARWAL, P., BASCH, J., GUIBAS, L., HERSHBERGER, J., AND ZHANG, L. 2001. Deformable Free Space Tilings for Kinetic Collision Detection. In *Workshop on Algorithmic Foundations of Robotics*, AK Peters, Ltd., 83–96.

BARBIČ, J., AND JAMES, D. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (TOG) 24*, 3, 990.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Transactions on Graphics (TOG) 28*, 3, 53.

BARBIČ, J., AND JAMES, D. L. 2010. Subspace self-collision culling. *ACM Trans. on Graphics (SIGGRAPH 2010) 29*, 3.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, 594–603.

CANNY, J. 1984. Collision detection for moving polyhedra. *Pattern Analysis and Machine Intelligence 8*, 200–209.

CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, 61–69.

GOVINDARAJU, N., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH) 24*, 3, 991–999.

GRINSPUN, E., AND SCHRODER, P. 2001. Normal bounds for subdivision-surface interference detection. In *IEEE Visualization'01*, IEEE Computer Society, 333–340.

HUTTER, M., AND FUHRMANN, A. 2007. Optimized continuous collision detection for deformable triangle meshes. *Proc. WSCG07*, 25–32.

KIM, B., AND ROSSIGNAC, J. 2003. Collision prediction for polyhedra under screw motions. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, ACM, 10.

KIM, D., GUIBAS, L., AND SHIN, S. 1998. Fast Collision Detection Among Multiple Moving Spheres. *IEEE Trans. Vis. Comput. Graph*, 230–242.

KIM, D., HEO, J., HUH, J., KIM, J., AND YOON, S. 2009. HPCCD: Hybrid Parallel Continuous Collision Detection using CPUs and GPUs. In *Computer Graphics Forum*, vol. 28, John Wiley & Sons, 1791–1800.

KIRKPATRICK, D., SNOEYINK, J., AND SPECKMANN, B. 2000. Kinetic collision detection for simple polygons. In *Proceedings of ACM symposium on Computational geometry*, ACM, 322–330.

LAUTERBACH, C., MO, Q., AND MANOCHA, D. 2010. gProximity: Hierarchical GPU-based Operations for Collision and Distance Queries. In *Proc. of Eurographics, to appear*.

LIN, M., AND CANNY, J. 1993. Efficient collision detection for animation and robotics. *University of California, Berkeley*.

MIRTICH, B. 1996. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, Citeseer.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics interface*, vol. 97, Citeseer, 177–189.

REDON, S., KHEDDAR, A., AND COQUILLART, S. An Algebraic Solution to the Problem of Collision Detection for Rigid Polyhedral Objects.

REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. In *Computer graphics forum*, vol. 21, Citeseer, 279–288.

SCHVARTZMAN, S., GASCÓN, J., AND OTADUY, M. 2009. Bounded normal trees for reduced deformations of triangulated surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 75–82.

SCHVARTZMAN, S., PÉREZ, A., AND OTADUY, M. 2010. StarContours for Efficient Hierarchical Self-Collision Detection. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, vol. 29.

SUD, A., GOVINDARAJU, N., GAYLE, R., KABUL, I., AND MANOCHA, D. 2006. Fast proximity computation among deformable models using discrete voronoi diagrams. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH) 25*, 3, 1153.

TANG, M., CURTIS, S., YOON, S., AND MANOCHA, D. 2009. ICCD:Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics 15*, 544–557.

TANG, M., KIM, Y., AND MANOCHA, D. 2009. C2a: Controlled conservative advancement for continuous collision detection of polygonal models. In *Proceedings of International Conference on Robotics and Automation*, 849–854.

TANG, M., MANOCHA, D., AND TONG, R. 2009. Multicore collision detection between deformable models. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, ACM, 355–360.

TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 7–13.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG) 25*, 3, 834.

VOLINO, P., AND THALMANN, N. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Computer Graphics Forum*, vol. 13, Amsterdam: North Holland, 1982-, 155–166.

ZHANG, X., LEE, M., AND KIM, Y. 2006. Interactive continuous collision detection for non-convex polyhedra. *The Visual Computer 22*, 9, 749–760.

ZHANG, X., REDON, S., LEE, M., AND KIM, Y. 2007. Continuous collision detection for articulated models using taylor models and temporal culling. In *ACM Trans. Graph. (ACM SIGGRAPH 2007)*, ACM, vol. 26.