Vertex Block Descent (VBD)

Notes distillées à partir de l'excellent ouvrage Vertex Block Descent.

```
@article{ankachen_2024_VBD,
  author = {Chen, Anka He and Liu, Ziheng and Yin, Yang and Yuksel, Cem},
  journal = {ACM Transactions on Graphics (TOG)},
  number = {116},
  publisher = {ACM New York, NY, USA},
  title = {Vertex Block Descent},
  year = {2024}}
```

> Les auteurs de la publication faisant objet de ce document n'acceptent aucune responsabilité en cas d'erreurs ou d'omissions et n'ont aucune affiliation avec ma personne.
>
> Ce document est informel a été rédigé à titre de support dans le cadre d'une évaluation universitaire de cycle supérieur et représente, simplement et uniquement, les idées exprimées par les auteurs originaux distillées et résumées en français, dans mes propres mots, ce qui m'a permit de comprendre, assimiler et vulgariser leurs idées.
>
> Ce document pourrait contenir des erreurs mathématiques, grammaticales ou de toute autre nature.
>
> Veuillez vous référer à et citer l'ouvrage original.

> Ce document a été rédigé sur Obsidian et contient des balises spécifiques à celui-ci qui pourraient ne pas être valides avec un autre préprocesseur.

Introduction

Contexte du problème

La simulation basée sur la physique est essentielle dans de nombreuses applications graphiques.

Vertex Block Descent (VBD) est un nouveau solveur physique pour la dynamique des corps élastiques qui permet de relever des défis majeurs dans les simulations graphiques, soient:

- une stabilité améliorée;
- une performance de calcul accélérée;

- un réalisme visuel accru.

Le problème est présenté dans le contexte de corps élastiques dynamiques pour des objets présentés par un ensemble de sommets (*vertices*) possédant une masse et un ensemble de forces et contraintes agissant sur ceux-ci.

La méthode peut, toutefois, être généralisée à d'autres problèmes de simulation, comme la simulation de corps rigides (*Rigidbodies*) et les simulations basées sur les particules.

Qu'est-ce qui a motivé la recherche?

Les méthodes existantes ont des défauts, et offrent deux choix de compromis:

- résultats de haute qualité, mais le système est trop glouton;
- réussi à respecter le budget de ressources, mais sacrifie en qualité et réalisme.

La recherche a été motivée par le besoin d'un solveur physique qui peut:

- atteindre la convergence numérique tout en garantissant la stabilité inconditionnelle;
- offrir une performance de calcul supérieure;

Défis techniques

Le principal défi technique consistait à développer une méthode capable d'opérer efficacement au niveau des sommets (*vertices*) tout en garantissant une réduction globale de l'énergie et en maximisant le parallélisme.

Méthodologie

La méthode VBD est basée sur la technique de *Block Coordinate Descent*, et performe des itérations de Gauss-Seidel basées sur les sommets (*vertices*) pour résoudre la forme variationnelle d'Euler implicite.

Tangente #1: Block Coordinate Descent

La descente par blocs de coordonnées (*Block Coordinate Descent*) est un algorithme d'optimisation qui minimise une fonction en mettant à jour itérativement un bloc de variables à la fois, tout en gardant les autres fixes.

1. La fonction objective est divisée en blocs de variables.

2. L'algorithme parcourt chaque bloc de variables.
3. Pour chaque bloc, l'algorithme minimise la fonction par rapport à ce bloc tout en gardant tous les autres blocs fixes.
4. Ce processus se répète jusqu'à ce qu'un critère d'arrêt soit rempli ou que la convergence soit atteinte.

VBD innove en utilisant des blocs de coordonnées basés sur les sommets (*vertices*) au lieu de blocs basés sur les éléments, ce qui permet un bien meilleur parallélisme et des systèmes linéaires locaux plus petits à résoudre, conduisant à une convergence plus rapide

Tangente #2: Forme variationnelle

La forme variationnelle est une méthode alternative de formuler la méthode d'Euler implicite pour une équation différentielle.

La forme variationnelle d'Euler implicite reformule le problème de pas de temps comme un problème d'optimisation.

Au lieu de résoudre directement les équations de mise à jour implicites, elle cherche à minimiser une certaine fonctionnelle (une fonction de fonctions) qui équilibre deux termes concurrents.

Optimization globale

Le problème d'optimization (*minimization*) peut être écrit comme

$$x_{t+1} = argmin_x G(x)$$

où

- $G(x)$ représente l'énergie variationnelle, d'équation

  $$G(x) = \frac{1}{2h^2}\|x - y\|_M^2 + E(x)$$

  où
- $\frac{1}{2h^2}\|x - y\|_M^2$ représente l'inertie potentielle;
  - h représente la taille du pas de temps;
  - $y = x^t + hv^t + h^2 a_{ext}$
    - a_{ext} représente une accélération externe fixe, comme la gravité;
- $E(x)$ représente l'énergie potentielle totale.

> **TL;DR**
> La technique d'optimization proposée tombe dans la catégorie des méthodes de descente de coordonnées, pour minimiser efficacement l'énergie variationnelle, G.

Optimization locale

En modifiant qu'un seul sommet à la fois et en fixant tous les autres sommets, la partie du terme énergétique E(x) qui est affectée ne comprend que l'ensemble des éléments de force Fi qui agissent sur (ou utilisent la position de) le sommet i.

> Rappel:
> Point 3 du *Block Coordinate Descent*, mise à jour des blocs.

Ainsi, l'énergie variationnelle locale Gi autour du sommet i est définie comme:

$$G_i(x) = \frac{m_i}{2h^2}\|x_i - y_i\|^2 + \sum_{j \in F_i} E_j(x)$$

où

- m_i est la masse du sommet i;
- E_j est l'énergie de l'élément de force j.

Il est à noter que

$$G(x) \neq \sum_i G_i(x)$$

car les éléments de force apparaissent plusieurs fois dans cette somme (une fois pour chacun de ses sommets). Cependant, quand la position d'un seul sommet est modifiée, la réduction de G_i est égale à la réduction de G qui en résulte. En d'autres mots, le changement d'énergie de chaque ajustement de la position d'un sommet est cumulé à l'énergie du système.

Enfin, on peut reformuler le problème d'optimization global comme une suite de problèmes d'optimization locale

$$x_i \leftarrow argmin_{x_i} G_i(x)$$

et résoudre le système à l'aide d'itérations de Gauss-Seidel.

Chaque minimisation locale pour un sommet trouve une étape de descente pour G en utilisant les degrés de liberté (*DoF*) du sommet comme un bloc de coordonnées, d'où le nom **Vertex Block Descent** (VBD).

La vélocité résultante est calculée selon la formulation d'Euler implicite:

$$v^{t+1} = \frac{1}{h}(x^{t+1} - x^t)$$

> **TL;DR**
> VBD décompose le problème global en optimisations locales pour chaque sommet.

Solveur de système local

Le système linéaire 3D à résoudre est

$$H_i \Delta x_i = f_i$$

où

- Δx_i est le changement de position,
- f_i est la force totale agissant sur le sommet tel que

$$f_i = -\frac{\partial G_i(x)}{\partial x_i} = -\frac{m_i}{h^2}(x_i - y_i) - \sum_{j \in F_i} \frac{\partial E_j(x)}{\partial x_i}$$

- H_i est la matrice hessienne de G_i tel que

$$H_i = \frac{m_i}{h^2} I + \sum_{j \in F_i} \frac{\partial^2 E_j}{\partial x_i \partial x_i}$$

> La matrice hessienne permet, dans de nombreux cas, de déterminer la nature des points critiques de la fonction f, c'est-à-dire des points d'annulation du gradient.
>
> *source*: https://fr.wikipedia.org/wiki/Matrice_hessienne

Le système est résolu analytiquement, en posant

$$\Delta x_i = H_i^{-1} f_i$$

Les auteurs ont trouvé que pour un aussi petit système (le système local), la solution analytique est efficace et stable. Ils l'ont comparée à des solveurs basés sur la technique de gradient conjugué, ou la décomposition LU/QR.

Damping

L'amortissement numérique entraîné par la méthode d'Euler implicite ne donne pas à l'utilisateur de contrôle sur l'amortissement. L'ajout d'un coefficient d'amortissement est donc hautement désirable.

En ajoutant le coefficient d'amortissement, la matrice hessienne devient

$$H_i = \frac{m_i}{h^2} I + \sum_{j \in F_i} \frac{\partial^2 E_j}{\partial x_i \partial x_i} + (\sum_{j \in F_i} \frac{k_d}{h} \frac{\partial^2 E_j}{\partial x_i \partial x_i})$$

et la force devient

$$f_i = -\frac{m_i}{h^2}(x_i - y_i) - \sum_{j \in F_i} \frac{\partial E_j(x)}{\partial x_i} - (\sum_{j \in F_i} \frac{k_d}{h} \frac{\partial^2 E_j}{\partial x_i \partial x_i})(x_i - x_i^t)$$

Contraintes

Puisque la méthode de VBD manipule directement la position de chaque sommet, la gestion d'une contrainte sur un sommet devient simple.

Les contraintes se répartissent généralement en deux catégories : unilatérales (C(x) ≤ 0) ou bilatérales (C(x) = 0). Avec les contraintes bilatérales, si la position d'un sommet est directement fixée à une valeur spécifique, nous sautons simplement la mise à jour de sa position. Dans le cas contraire, elle est contrainte à un sous-espace (généralement linéaire).

Dans la simulation, les contraintes unilatérales sont utilisées pour prendre en charge les contraintes comme le *world box* (*bounding box* de la simulation).

Collisions & Friction

Les collisions peuvent être gérées en introduisant simplement une énergie de collision quadratique par sommet, basée sur la profondeur de pénétration d, telle que

$$E_c(x) = -\frac{1}{2} k_c d^2$$

où

- $d = max(0, (x_b - x_a) \cdot \hat{n}$
- k_c correspond au coefficient de rigidité
- x_a et x_b correspondent aux deux points de contacts
- $\hat{n}$ correspond à la normale de contact

Deux types de collisions sont supportées pour les maillages triangulaires.

- Les collision arrête-arrête (*edge-edge*) sont prises en charge par détection de collision continue (*continuous collision detection (CCD)*).
    - x_a et x_b correspondent aux points d'intersection sur n'importe quel arrête et la normale de contact est la direction entre celles-ci.

- Les collisions sommet-triangle (*edge-triangle*) sont détectées soit par CCD ou par détection discrète de collision (*Discrete Collision Detection (DCD)*).
    - x_a est le sommet en collision, x_b est le point de collision dans le cas de CCD ou le point le plus proche sur le triangle dans le cas de DCD et $\hat{n}$ est la normale de surface au point x_b.

L'implémentation performe une DCD au début du pas de temps en utilisant x^t pour identifier les sommets ayant déjà été pénétrés, et le reste des collisions utilisent la CCD.

Pour être en mesure de calculer la friction associée à une collision c, il faut considérer le mouvement relatif au point de contact, défini comme

$$\delta x_c = (x_a - x_a^t) - (x_b - x_b^t)$$

La magnitude signée de la force engendrée par la collision c appliquée sur le sommet i est $\lambda_{c,i}$, définie comme

$$\lambda_{c,i} = \frac{\partial E_c}{\partial x_i} \cdot \hat{n}$$

> À noter que le signe de $\lambda_{c,i}$ est différent pour les sommets aux différents pôles de la collision.

On pose u_c, le coefficient de friction.
Alors, la force de friction f_i est

$$f_{c,i} = -u_c \, \lambda_{c,i} \frac{\partial \delta x_c}{\partial x_i} T_c f_1(||u_c||) \frac{u_c}{||u_c||}$$

Le terme hessien de friction est nécessaire, et correspond à la dérivée de la précédente fonction.
Ce terme est approximé en ne différenciant pas par $||u_c||$ pour une formulation plus stable de la force de frottement.

$$\frac{\partial f_{c,i}}{\partial x_i} \approx -u_c \, \lambda_{c,i} \frac{\partial \delta x_c}{\partial x_i} T_c \frac{f_1(||u_c||)}{||u_c||} T_c^T (\frac{\partial \delta x_c}{\partial x_i})^T$$

Finalement, les forces de friction $f_{c,i}$ et leurs dérivées $\frac{\partial f_{c,i}}{\partial x_i}$ sont ajoutées à f_i et H_i respectivement.

Considérations

Premièrement, le solveur analytique ne demande pas que la hessienne soit définie-positive. Comme on cherche un point tel que

$$\frac{d\,G(x)}{d\,x} = 0$$

toute valeur extrême est une solution valide d'Euler implicite. Les auteurs n'ont pas observé de problèmes de stabilité ou convergence engendrés par cette assomption.

Deuxièmement, si le solveur analytique rencontre une hessienne qui est déficiente en rang, on saute l'ajustement du sommet pour cette itération si pour ϵ, un petit seuil prédéfini, on a

$$|| H_i|| \geq \epsilon$$

Comme les sommets voisins vont fort probablement voir leur position changer d'ici la prochaine itération, il est improbable que la hessienne soit déficiente en rang dans les prochaines itérations.

Initialization

Tout solveur itératif demande une approximation initiale de x.

Les auteurs comparent 4 approches, dont leur approche adaptative (d).

a) $x = x^t$
b) $x = x^t + hv^t$
c) $x = x^t + hv^t + h^2 a_{ext}$
d) $x = x^t + hv^t + h^2 \widetilde{a}$

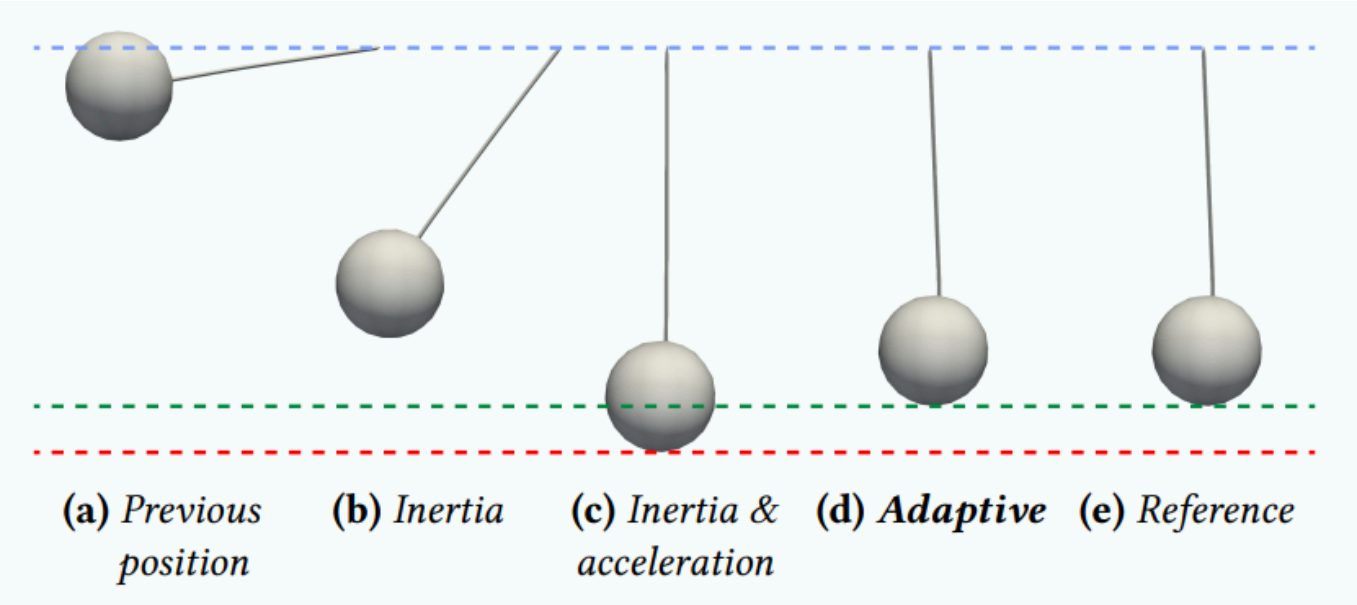
L'approche adaptative est celle qui garde le comportement du système le plus près du comportement de référence à l'initialization.

Le système est donc initialisé avec

$$x = x^t + hv^t + h^2 \widetilde{a}$$

où

- $\widetilde{a} = \widetilde{\alpha}\, a_{ext}$
  - $\widetilde{\alpha} = 1$ si $a^t_{ext} > ||a_{ext}||$
  - $\widetilde{\alpha} = 0$ si $a^t_{ext} < 0$
  - $\widetilde{\alpha} = \frac{a^t_{ext}}{||a_{ext}||}$ sinon



Structure accélératrices

L'approche semi-itérative de Chebyshev est utilisée pour améliorer la convergence de la méthode de VBD.

La méthode de Chebyshev calcule de manière itérative un rapport d'accélération basé sur l'approximation du rayon spectral du système.

Au lieu d'utiliser directement les positions des sommets de Gauss-Seidel $\bar{x}^n$ après l'itération n, elle recalcule les positions à la fin de l'itération:

$$x^{(n)} = \omega_n \left(\bar{x}^{(n)} - x^{(n-2)}\right) + x^{(n-2)}$$

où

- ω_n correspond au ratio d'accélération qui a changé à l'itération n
  - À la première itération $n = 1$, on utilise $\omega_1 = 1$.
  - À la deuxième itération $n = 2$, on utilise $\omega_2 = \frac{2}{2-\rho^2}$.
  - Pour $n \geq 3$, on utilise $\omega_n = \frac{4}{4-\rho^2\,\omega_{n-1}}$
- $\rho \in (0, 1)$ correspond au rayon spectral estimé.

> Le rayon spectral d'une matrice carrée est le maximum des valeurs absolues de ses valeurs propres.
>
> $$\rho(A) = max\{|\lambda_1|,\ \ldots\ |\lambda_n|\}$$
>
> Dans le contexte, le rayon spectral est ajusté manuellement, mais peut être calculé de manière dynamique.

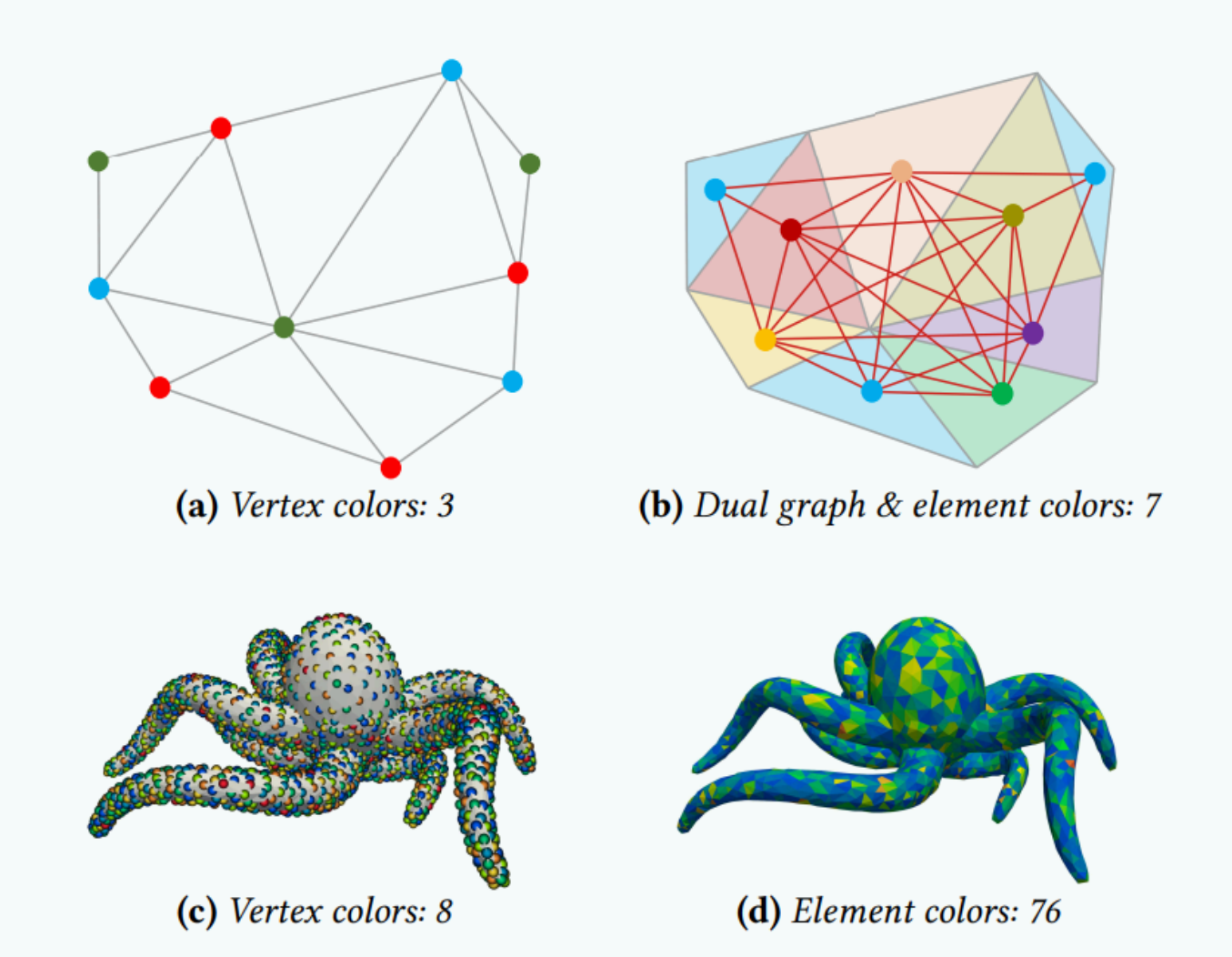
Ce calcul est fait à chaque itération de Gauss-Seidel au niveau global, et non pas après un pas du solveur local.

Finalement, l'accélération de convergence est sautée pour les sommets en collision.
Selon les auteurs, cette approche a un impact minimal sur la convergence de l'élasticité comme typiquement, seul une petite fraction des sommets sont en collision. De plus, pour ces sommets en collision, l'élasticité est généralement dépassée par les forces de collision.

Parallélisation

VBD est parallélisé en colorant le graphe de sommets original, ce qui conduit à un niveau de parallélisme beaucoup plus élevé. Le nombre de couleurs dans le graphe est ainsi minimisé.

> ***Couleurs***
> Groupes de calcul qui doivent être traités de manière séquentielle.
> Les éléments d'un groupe de calcul (***couleur***) peuvent être traités de manière indépendante (parallèle).



L'approche utilisée afin de paralléliser les itérations de Gauss-Seidel fonctionne également avec d'autres types de changements topologiques, tels que les déchirures et les fractures.

La suppression d'éléments de force ne nécessite aucune modification de la coloration des sommets. Lorsqu'un objet est divisé en dupliquant des sommets, comme dans le cas de la déchirure d'un morceau de tissu le long de certains bords, les sommets dupliqués peuvent en toute sécurité hériter les couleurs de leurs sommets d'origine.

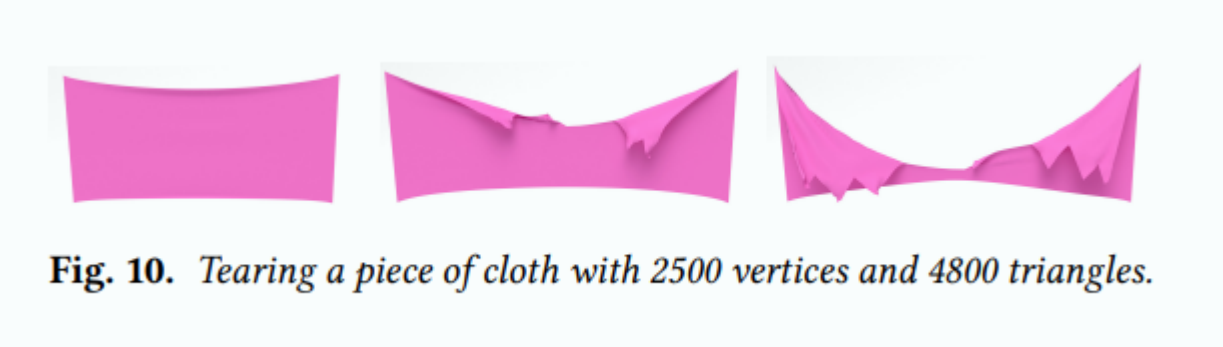


Fig. 10. *Tearing a piece of cloth with 2500 vertices and 4800 triangles.*

Pseudocode

Algorithm 1: VBD simulation for one time step.

Input: $\mathbf{x}^t$: the positions of the previous step; $\mathbf{v}^t$: the velocities of the previous step; $\mathbf{a}_{\text{ext}}$: the external acceleration

Output: This step's position $\mathbf{x}^{t+1}$ and velocity $\mathbf{v}^{t+1}$.

$\mathbf{y} \leftarrow \mathbf{x}^t + h\mathbf{v}^t + h^2\mathbf{a}_{\text{ext}}$
Initial DCD using $\mathbf{x}^t$
$\mathbf{x} \leftarrow$ initial guess with adaptive initialization
for each iteration $n \leq n_{\max}$ **do**
    if $n \bmod n_{\text{col}} = 1$ **then** CCD using $\mathbf{x}$
    for each color c **do**
        // Block-level parallelization
        parallel for each vertex i in color c **do**
            // Thread-level parallelization
            parallel for each $j \in \mathcal{F}_i$ **do**
                // Variables in shared memory
                $\mathbf{f}_{i,j} = -\frac{\partial E_j}{\partial \mathbf{x}_i}$
                $\mathbf{H}_{i,j} = \frac{\partial^2 E_j}{\partial \mathbf{x}_i \partial \mathbf{x}_i}$
            end
            // Local reduction sums
            $\mathbf{f}_i = \sum_{j \in \mathcal{F}_i} \mathbf{f}_{i,j}$
            $\mathbf{H}_i = \sum_{j \in \mathcal{F}_i} \mathbf{H}_{i,j}$
            $\Delta\mathbf{x}_i \leftarrow \mathbf{H}_i^{-1}\mathbf{f}_i$
            $\Delta\mathbf{x}_i \leftarrow$ optional line search from $\mathbf{x}_i + \Delta\mathbf{x}_i$ to $\mathbf{x}_i$
            $\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i + \Delta\mathbf{x}_i$
        end
        // Copy updated positions back to the vertex buffer
        parallel for each vertex i in color c **do**
            $\mathbf{x}_i = \mathbf{x}_i^{\text{new}}$
        end
    end
    // Optional: accelerated iteration
    parallel for each vertex i **do**
        Update $\mathbf{x}_i$ using Equation 18.
    end

```
24      end
25  end
26  v = (x − x^t)/h
27  return x, v
```

Différences avec les méthodes standard

Comme PBD et XPBD, VBD travaille avec les mises-à-jour de positions, mais utilise directement les formulations de force, sans les convertir en contraintes.

De plus, les approximations dans la formulation de XPBD introduisent des erreurs qui font diverger le résultat de la solution d'Euler implicite, ce qui mène à un réalisme diminué, spécialement avec des grands pas de temps et un nombre limité d'itérations; deux facteurs courants en pratique. XPBD a particulièrement du mal avec les rapports de masse élevés. Additionnellement, XPBD dicte une certaine initialization du système.
VBD ne souffre d'aucune de ces limitations.

Finalement, la parallélisation avec XPBD est réalisée en colorant le graphe des contraintes. *Cependant*, le graphe contient beaucoup plus de connexions (en fonction des types de contraintes) que le graphe des sommets, ce qui limite considérablement le niveau de parallélisme. En comparaison, VBD est parallélisé en colorant le graphe original, ce qui conduit à un niveau de parallélisme beaucoup plus élevé.

Discussion

L'article présente VBD comme une méthode innovante pour la simulation basée sur la physique, offrant une stabilité inconditionnelle, une performance de calcul supérieure et une convergence rapide.

Les contributions incluent des formulations pour l'amortissement, les contraintes, les collisions et la friction, ainsi que des techniques d'accélération basées sur le momentum et la parallélisation pour la simulation de corps mous.

Travaux futur

Explorer les stratégies à pas de temps dynamique.

Étude d'approches multi-résolution ou hiérarchiques afin d'améliorer encore les performances pour les simulations à très grande échelle.

Développer des implémentations GPU spécialisées afin d'exploiter pleinement les architectures de calcul parallèle.