Vol.2016-CG-164 No.11
2016/9/21

# Air Meshes for Robust Collision Handling 【CG 技術の実装と数理 2016】研究報告

中川 展男[1,a)]

**概要**：筆者は SIGGRAPH 2015 で M. Müller らが発表した論文 “Air Meshes for Robust Collision Handling” [1] を実装し【CG 技術の実装と数理 2016】で論文概要の紹介を行った．この論文は，Air Mesh と呼ばれるオブジェクトの周りのメッシュを考慮することで，衣服など複雑で変形するオブジェクトの衝突検出と応答を堅牢に行う事を可能とする．Air Mesh を構成する 3 角形 (2D) あるいは 4 面体 (3D) に対し符号付き面積 (2D)，体積 (3D) を求め，この符号の反転を防ぐために単一方向の制約を考慮する．この際に大きな回転と並進運動がある場合は，3 角形 (2D)，4 面体 (3D) の品質を求め，この品質が閾値を下回った際に Air Mesh の再分割が必要となる．この再分割は 3D の場合, 計算コストが高いため，モーションが限定的な場合での使用に限られる．

# Air Meshes for Robust Collision Handling

**“Mathematics and Implementation of Computer Graphics Techniques 2016” Technical Report**

Nobuo Nakagawa[1,a)]

## 1. はじめに

### 1.1 背景と目的

衝突検出と衝突応答はコンピュータ・グラフィックスにおいて活発な研究領域となっているが，この論文 [1] では，薄いレイヤーの衣服が絡みあったような複雑な衝突検出を堅牢に行い，ビデオゲーム向けの高速な衝突検出・応答手法を提案している．

### 1.2 関連研究

本論文は，M. Müller らによって近年多くの発表がなされている Position Based Dynamics(PBD) [2] 関連の研究である．これは既存の加速度，速度を求めオブジェクトに働く力を求める既存のアプローチと異なり，直接位置を更新する手法である．これまで PBD の制約条件として距離制約や曲げ制約 [2] などが提案されてきたが，本研究では，Air Mesh と呼ばれるオブジェクトの周りのメッシュを考慮し，このメッシュを構成する 3 角形 (2D) あるいは 4 面体 (3D) の符号付き面積 (2D)，体積 (3D) が，正になることを強制するような単一方向の制約条件を PBD のフレームワークに加えている．

## 2. 手法

手法は，以下の 3 ステップで構成される．
(1) オブジェクト間の Air Mesh のテセレーション
(2) 要素毎の制約の適用
(3) メッシュ最適化
以下ではこの 3 つのステップの詳細を確認していく．

### 2.1 オブジェクト間の Air Mesh のテセレーション

事前計算として，オブジェクト間の Air Mesh を構築する．このステップはシミュレーションが開始する前に一度だけ実行すればよいので計算時間は深刻な問題とはならない．論文 [1] では，Air Mesh の生成手法として TetGen が挙げられていたが，今回の筆者の実装では，外部ツールに

[1] 株式会社ポリフォニー・デジタル
Polyphony Digital Inc.
http://www.polyphony.co.jp
[a)] nakagawa@polyphony.co.jp

依存せず動くものとしたかったため独自にドロネー 3 角形分割の実装を行っている．

### 2.2 要素毎の制約の適用

シミュレーションの間中，Air Mesh は，その符号付き面積 (2D) あるいは体積 (3D) が正である限りはいかなる処理も行わないが，符号が負になった時のみ，その面積あるいは体積が 0 になるように復元するような力を加える．このような単一方向の制約によりシステムの構築が可能となっている．PBD のフレームワークで考えると式 (1) で 2D の場合，式 (2) で 3D の場合を考慮することができる．ここで $p_1, p_2, p_3, p_4$ は，3 角形あるいは 4 面体を構成する頂点となる．また図 1 に，3 角形 (2D) の場合の Air Mesh の体積制約のための勾配ベクトルを示す．運動量は PBD の更新の際に保存される．例えば赤いパーティクルに近い青いパーティクルはより大きく更新される．

$$C_{air} = |(p_2 - p_1) \times (p_3 - p_1)| \geq 0 \tag{1}$$

$$C_{air} = \det[p_2 - p_1, p_3 - p_1, p_4 - p_1] \geq 0 \tag{2}$$

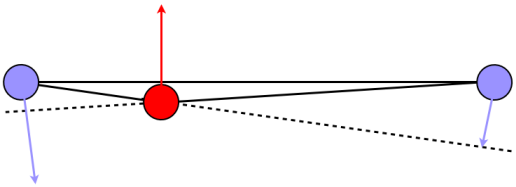

**図 1** Air Mesh の体積制約勾配ベクトル表示 (2D)

### 2.3 メッシュ最適化

Air Mesh は，オブジェクト間に衝突がない場合でも反転する場合がある．この問題はオブジェクトが比較的大きな回転や移動を行った際に生じる．この問題を避けるために 3 角形あるいは 4 面体の品質の測定を行い，一定以下の値になった時に辺のフリップを実行し問題を回避する．この際に式 (3) を 3 角形の品質測定に用いる，ここで $A$ は，3 角系の面積で $l_1, l_2, l_3$ は，3 角形の 3 辺の長さを示す．同様に 4 面体は式 (4) を用い．$V$ が 4 面体の体積となり $l_{rms}$ が，4 面体を構成する辺の長さ $l_1, l_2, l_3, l_4, l_5, l_6$ の二乗平均平方根となる．図 2 では，メッシュ最適化の例 (2D) を示している．(左) 動的な黄色い箱が，長方形の境界に囲まれており，3 角形の面積が反転することを防ぐことで堅牢に壁との衝突応答を行うことができる．(中) 黄色の箱が大きく回転することで三角形がロックしてしまう状況．ここで三角形の質を計算しフリップを実行する．(右) フリップによる Air Mesh の最適化により黄色い箱が自由に回転した後の状態を示す．

$$q_{triangle} = \frac{4}{\sqrt{3}} \frac{A}{l_1^2 + l_2^2 + l_3^2} \tag{3}$$

$$q_{tetrahedron} = \frac{12}{\sqrt{2}} \frac{V}{l_{rms}^3}$$

$$l_{rms} = \sqrt{\frac{l_1^2 + l_2^2 + l_3^2 + l_4^2 + l_5^2 + l_6^2}{6}} \tag{4}$$

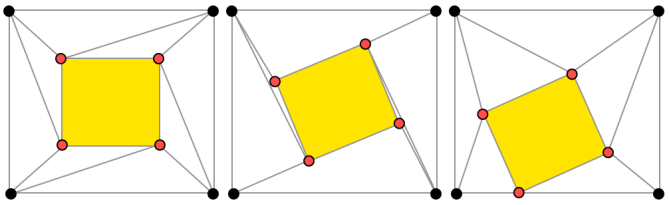

**図 2** Air Mesh の最適化の様子 (2D)

## 3. 実装

問題の複雑さを避けるため，計画的に段階的な実装を行うことにした．具体的には，第 1 回 ( 7 月) の概要発表に向けて 2D 版の実装を行い，その後第 2 回 (9 月) の成果発表に向けて 3D 版の実装を行っている．実装は C++ と OpenGL を用い Windows と OSX の両方の環境で動作を確認できるようにした．またベクトル演算等には glm ライブラリを用いた．筆者による実装のソースコードを github で公開している [3]．また関連論文 [1], [2] の日本語参考訳も筆者の承諾を得て同様に github で公開した [3]．また 7 月の研究発表会のポスター発表時に，この実装例の実機デモを行った．実装結果のスクリーンショットを図 3 に示す．

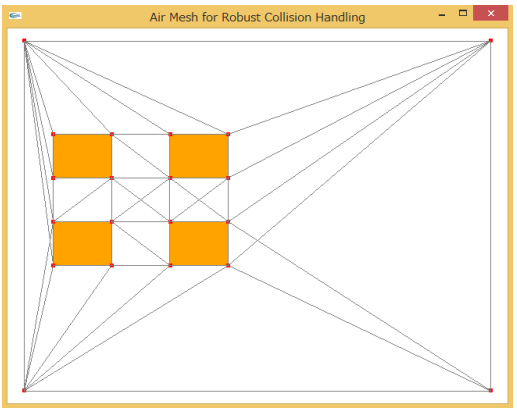


**図 3** 実装結果

## 4. 実験

論文で実験されている状況と同様の条件で筆者が独自に実装したシステムで再現実験をおこなったところ提供されている動画と同様の堅牢な結果が得られ手法の有用性が確認できた．

## 5. おわりに

### 5.1 議論

論文 [1] 内では詳細な記述がないが，この手法を効率的に 3D の領域に適用する場合には，対象領域をうまく絞り込む必要があると感じた．周辺領域全てに Air Mesh を構築するとコストが高くなりすぎるためである．また同様に 3D の場合に効率的なメッシュ最適化を行い不必要なエッジのフリップを回避できるなら，大きな回転運動や並進運動を含むダイナミックなシーンでの使用も期待できる．しかし現状では 3 次元の場合は Air Mesh の再分割を含まない限定的なケースでのみ，ビデオゲームなどのリアルタイム処理において十分なパフォーマンスで実行できると感じた．したがって利用できるケースは限られるが，例としてあげられている複数レイヤーを含むクロスシミュレーションなどの状況は，他の手法では高速かつ安定的に解くことが難しく本手法の有用性をよく示す極めて魅力的な例となっている．2D の場合は計算コストの問題が小さくなり幅広い応用が期待されるが，とりわけ Air Mesh の面積が 0 になるような完全に圧縮して潰れるようなケースや，そのような状況でも上下の位置関係が維持される特性を活かした場面で極めて有用である言える．

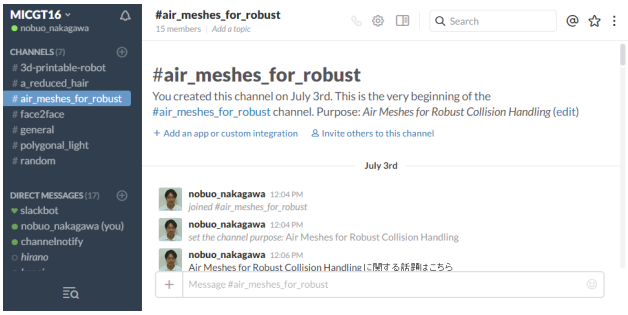


**図 4** Slack の様子

### 5.2 研究会に参加して

7 月の研究会では口頭発表の後にポスター発表の時間が設けられ，論文を読んで疑問に感じていた箇所に関して参加者と議論することができ非常に有意義であった．また 7 月と 9 月の間にもオンラインで議論を継続させるために参加者で構成されるプライベートな slack グループを作成し各自の進捗報告を行えるようにした．slack グループでは，論文ごとにスレッドを作成し，議論が継続できるようにした (図 4)．私の告知が不十分だったこともあり参加者はそれほど多くなかったが一定の成果はあったと考えている．

## 参考文献

[1] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim and Miles Macklin.: *Air Meshes for Robust Collision Handling.*, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2015, Volume 34 Issue 4, August 2015 Article No. 133, (2015).
[2] Matthias Müller, Bruno Heidelberger, Marcus Hennix and John Ratcliff.: *Position based dynamics*, Journal of Visual Communication and Image Representation Volume 18 Issue 2, April, 2007, Pages 109-118, (2007)
[3] 筆者の github ページ, 入手先 ⟨https://github.com/nobuo-nakagawa⟩ (2016.08.28).