Fast, Interactive Origami Simulation using GPU Computation
GPU計算を使用した、高速でインタラクティブな折り紙シミュレーション

Amanda Ghassaei, Erik D. Demaine, Neil Gershenfeld

【この文献は、上記の論文を日本語で(簡単に)まとめたものです。】

Abstract
GPUで迅速に計算できる折り紙のシミュレーション方法を提案する。
これまでの研究(折り紙シミュレーション手法)は、物理的なリアリズムに焦点を当て、折り紙の幾何学的・構造的な動作をモデル化した。
本稿では、計算速度と対話性を重視した、数値シミュレーション手法を紹介する。
既存の手法を再構成し、並列GPUアーキテクチャで計算できるようにし、そのメソッドをWebGLアプリとして実装する。
既存のメソッド（Freeform Origami、MERLIN）に対するメソッドのパフォーマンス、安定性、およびスケーラビリティを評価し、従来のGUIと没入型仮想現実を介したリアルタイムの対話の能力を実証する。

1 Introduction
折り紙の物理シミュレーションにより、折り目パターンの変更が折り畳み状態に与える影響を理解できる。
リアルタイムのシミュレーションベースのフィードバックは、折りたたみパターンを編集したり、グローバルな可展性などの所望の幾何学的制約を適用するための、より直感的な方法を提供する。
特に、シミュレーションは逆設計へのアプローチを提供し、計算ツールはユーザーが指定した高レベルの目標に従って有効な設計を生成する。
アルゴリズムの逆設計の代替手段（[Lang 96、Demaine and Tachi 17]など）は、緊密に統合されたシミュレーション機能と組み合わせたインタラクティブな設計ツールを開発することだ[Tachi 10、Tang et al. 16]。
これは、折り紙の計算設計ツールのバックボーンとなる、高速でインタラクティブなシミュレーション環境の構築を目的とする。

2 Simulation Methods
折り紙シミュレーションの数値計算方法では普通、折り目パターンを剛体リンケージまたは有限要素メッシュに離散化し、小さな変位の線形連立方程式を暗黙的に解く。
これらの方法は、反復プロセスを通じて、最初の平らな状態または折りたたまれた状態から、折り紙の大きな非線形変形を計算する。

剛体折り紙シミュレータ[Tachi 06]は、折り紙の剛体運動をモデル化する。
有限要素法（FEM）は、現実的な材料モデルと、プレート/シェル[Schenk et al. 14, Peraza Hernandez et al. 16]や体積要素[Ablat and Qattawi 18]などの高次要素を使用して、折りたたみ状態または部分的に折りたたまれた状態の構造特性を調べる。

バーアンドヒンジモデルは、折り紙の機械的動作を近似し、効率的・抽象的にモデリングする。[Tachi 10、LiuおよびPaulino 16]。
単純であるにもかかわらず、バーアンドヒンジモデルを使用して、折り目間の領域の面外曲げ[Tachi 13, Schenk and Guest 11]や面内せん断[Filipov et al. 17]などの現実的な動作をキャプチャできる。

私たちのソルバーの目的は、これまでに検討されていない物理的な動作をモデル化することではなく、ユーザーと直接対話するアプリケーションの折り紙の折りたたみ状態をすばやく計算することだ。
私たちのソルバーは、以前の暗黙のメソッドを動的で明示的な形式にリフレームし、GPUの高度な並列プロセスで解決される。
準拠した制約を使用して折り紙モデルの折りたたみをガイドし、ユーザーは計算速度と正確さ（またはその逆）のトレードオフを可能にする。

セクション2.1〜2.4は、メソッドの設定の概要を示しています（[Schenk and Guest 11]および[Tachi 10]による以前の作業に基づき、明確にするためにここで繰り返す）。セクション2.5では、明示的な統合方法を紹介する。セクション3では、GPUでのこのメソッドの並列化について説明する。

2.1 Meshing

まず、折り紙の折り目パターンを三角形分割されたメッシュに離散化する。
4つ以上の辺を持つポリゴン面にエッジを追加して、折り目パターンを三角形分割する（図1）。[Schenk and Guest 11]に続いて、これらのエッジを「ファセット折り目」と呼び、折りたたみ時に弾性的に拘束されて平坦なままになるようにする。

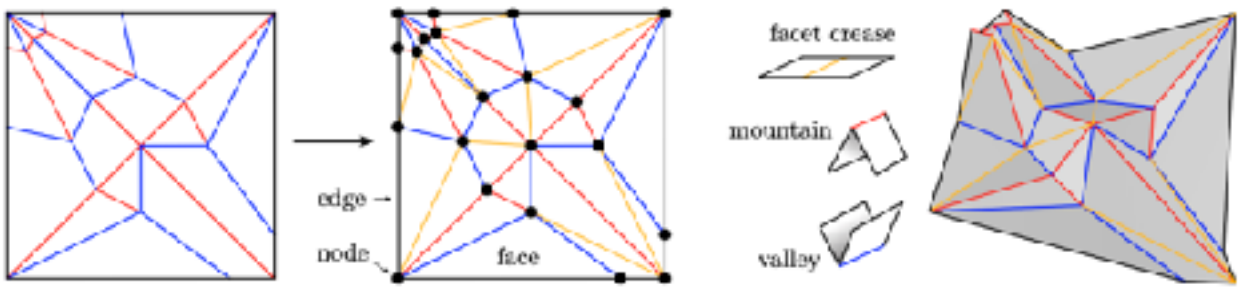

図1：折り紙の折り目パターン（左）の三角形分割メッシュ（中央）への三角形分割。
多角形の面は、自動化されたプロセスで三角形分割するか、ユーザーが直接制御する。

（右）山や谷の折り目とは異なり、三角形分割によって形成されたファセット折り目は、折りたたみ中に弾性拘束によって平坦になるよう働く。

結果として得られるメッシュの各エッジは、ピン結合トラスのビームとしてモデル化され、軸と角度の制約を受ける。

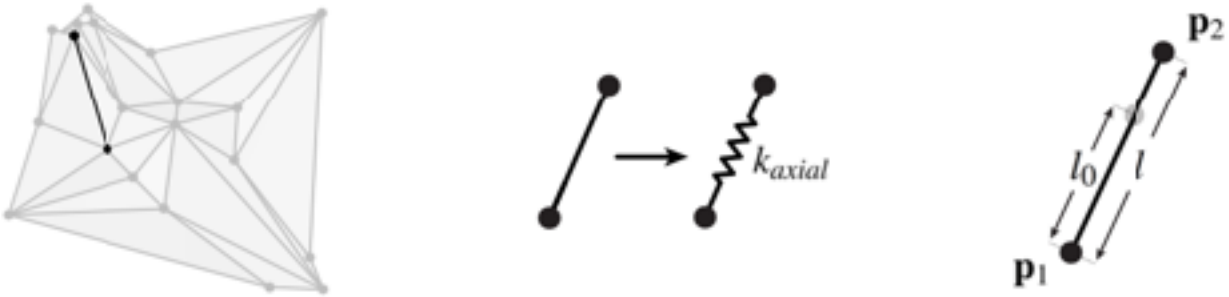

図2：（左）折り目パターンは三角形にされ、ピン結合トラスネットワークとしてシミュレーションされる。
（中央）トラスの梁は線形弾性ばねとしてモデル化されている。
（右）軸拘束の定式化。

2.2 Axial Constraints(軸の制約)

軸拘束により、折りたたみ時に折り紙の表面が伸び縮みするのを防ぐ。
各ビーム要素は、軸方向にのみ力を加える線形弾性ばねのように動作する（図2）。
シミュレーションの各ステップで、端点のノードの位置から各ビームの長さを計算する。
フックの法則$F_l = -k_{axial}(l - l_0)$によって、ビームのローカル座標空間の各ノードに加えられる力を計算する。
ここでF_lは、ビームの軸に沿った力、k_{axial}はビーム要素の軸方向の剛性、lはビームの長さでl_0はビームの通常時の長さである。

ノードに適用する前に、F_lをベクトルF_{axial}に変換する必要がある。

この軸力ベクトルは、$F_{axial} = -\nabla V(p) = -\dfrac{\partial V}{\partial p}$によってノード位置$p$に関連づけられる。

ここで、Vはシステムのポテンシャルエネルギーである。
チェインルールによって、

$$F_{axial} = -\frac{\partial V}{\partial l}\frac{\partial l}{\partial p} = F_l\frac{\partial l}{\partial p}, F_{axial} = -k_{axial}(l - l_0)\frac{\partial l}{\partial p} \quad (1)$$

となる。
各ビームに接続された2つのノードの場合、

$$\frac{\partial l}{\partial p_1} = -\hat{l}_{12}, \frac{\partial l}{\partial p_2} = \hat{l}_{12},$$

$\hat{l}_{12}$はノード1からノード2への単位ベクトルで、p1はグローバル座標空間におけるノード1の3D位置である。

軸の剛性は、$k_{axial} = \frac{EA}{l_0}$によって基板の材料特性に関連付けることができる。
ここで、Eはヤング率、Aはビームの断面積である。
この分析では、軸拘束と角度拘束の相対的な剛性のみを考慮し、それらの物理的な意味を無視する。
モデルに一定のEAを選択し、l_0に基づいて各ビームのkaxisを計算する。

2.3 Crease Constraints(折り目の制約)

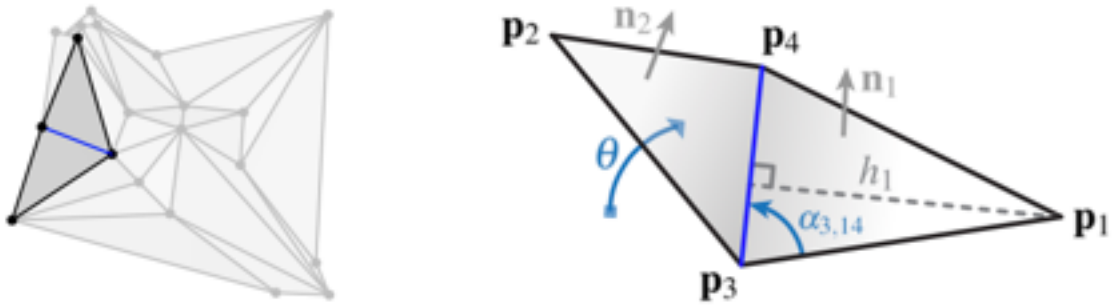

図３：折り目制約の定式化。

距離拘束のみを使用して、摩擦のない球形ジョイントで接続された任意のリンケージをモデル化できる。
メッシュの三角形分割された面間の二面角に制約を追加して、折り畳みを駆動および制約する。
折り目の折り角度（θ）は、隣接する2つの三角形の面の間の二面角の補足である。
角度制約は、隣接する三角形の面を目標の折り曲げ角度に向けて駆動する線形弾性ねじりばねとしてモデル化する。
式1と同様に、角度制約は

$$F_{crease} = -k_{crease}(\theta - \theta_{target})\frac{\partial \theta}{\partial p} \quad (2)$$

によって隣接するノードに力を適用する。
ここで、Fcreaseはグローバル座標空間の3D力ベクトル、θtargetは折り目の望ましい折り角度、pはノードの位置、kcreaseは拘束の剛性。
[Schenk and Guest 11]に従い、タイプに応じてkcreaseを選択し、通常時の長さl_0でスケーリングする。
$k_{crease} = \{l_0 k_{fold}$ 山谷の折り目、$l_0 k_{facet}$ファセット折り目(セクション2.1)、0境界エッジまたは駆動しない折り目。

kaxis» kfoldおよびkfacetとなるように、剛性を選択する。 角度θtargetは、折り目の種類にも依存する。
θ_{target} = {<0 山折り、>0 谷折り、0 ファセット折り目
山/谷の折り目のθtargetの正確な値は、ユーザーが設定する(セクション3を参照)。

図３で示されるように、各角度制約は力を4つの隣接したノードに適用する。
pに関するθの偏導関数は、

$$\frac{\partial\theta}{\partial p_1} = \frac{n_1}{h_1} \quad (3)$$

$$\frac{\partial\theta}{\partial p_2} = \frac{n_2}{h_2} \quad (4)$$

$$\frac{\partial\theta}{\partial p_3} = \frac{-cot\alpha_{4,31}}{cot\alpha_{3,14}+cot\alpha_{4,31}}\frac{n_1}{h_1} + \frac{-cot\alpha_{4,23}}{cot\alpha_{3,42}+cot\alpha_{4,23}}\frac{n_2}{h_2} \quad (5)$$

$$\frac{\partial\theta}{\partial p_4} = \frac{-cot\alpha_{3,14}}{cot\alpha_{3,14}+cot\alpha_{4,31}}\frac{n_1}{h_1} + \frac{-cot\alpha_{3,42}}{cot\alpha_{3,42}+cot\alpha_{4,23}}\frac{n_2}{h_2} \quad (6)$$

で与えられる。ここで、n1とn2は面の法線、h1とh2は折り目から外側のノードへのレバーアームである。 面内角度αは、図3に従って方向付けられる。

2.4 Face Constraints(面の制約)

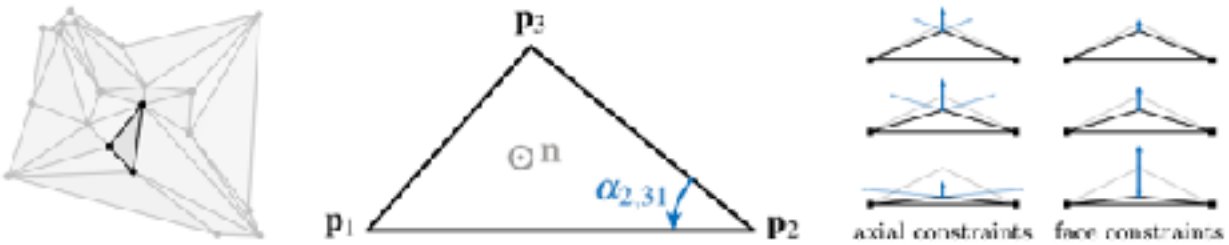

図４：（左と中央）面の制約の定式化。（右）三角形の面（2つの下部ノードが固定されている）が平らになると、隣接する2つの軸拘束によって中央ノードに適用される復元力は、三角形が変形するほど小さくなる。対照的に、三角形が変形するとき、3つの面拘束はより大きな力を適用する。個々の貢献(寄与,contribution)は水色で表示され、正味の力は濃い青で表示され、変形していない状態は灰色で表示され、固定ノードは四角で示される。

ビームと折り目の制約だけで、折りたたみをシミュレートできる。 厳密には必要ないが、**メッシュの三角形の面の各内角に制約を追加すると、特に高アスペクト比の三角形の場合、折りたたみ面のせん断を防ぐのに役立つ**（図4）。 実際には、面の制約により、さまざまな入力折り目パターン全体でシミュレーションの安定性が向上することがわかった。 以前の制約と同様に、これを線形弾性ばねとしてモデル化する。

三角形の面の各内角について、$F_{face} = -k_{face}(\alpha - \alpha_0)\frac{\partial \alpha}{\partial p}$にしたがって３つの隣接ノードに力を加える。ここで、αは現在の角度で、α_0は平坦な状態のその通常の角度で、pは隣接ノードの位置、k_{face}は面制約の剛性である。αに関するpの偏導関数は

$$\frac{\partial p_1}{\partial \alpha_{2,31}} = \frac{n \times (p_1 - p_2)}{||p_1 - p_2||^2},$$

$$\frac{\partial p_2}{\partial \alpha_{2,31}} = \frac{n \times (p_1 - p_2)}{||p_1 - p_2||^2} + \frac{n \times (p_3 - p_2)}{||p_3 - p_2||^2},$$

$$\frac{\partial p_3}{\partial \alpha_{2,31}} = -\frac{n \times (p_3 - p_2)}{||p_3 - p_2||^2}$$

で与えられ、ここでnは三角形の面の法線ベクトルで、αは図４にしたがって定義される。

2.5 Numerical Integration

これまでのところ、前のセクションで説明した支配方程式は、[Tachi 10]と[Schenk and Guest 11]で説明されている方法を組み合わせたものである。我々の方法は、明示的な方法を使用して、軸および角度の制約の下でノードの小さな変位を計算するという点で、以前の研究とは異なる。
これらの計算は、グローバルな剛性/ヤコビ行列なしで、ノードごとに並行して行われる。
ノードに加わる力の合計は、隣接するビーム、折り目、面によって加えられた力の合計として計算される。

$$F_{total} = \Sigma_{beams} F_{beam} + \Sigma_{creases} F_{crease} + \Sigma_{faces} F_{face}$$

ノードの加速度を$a = \frac{F_{total}}{m}$で計算する。ここで、aはグローバル座標空間での３次元加速度ベクトル、mはノードの質量である。この分析では、各ノードの質量を1と仮定する。ノードの質量のより正確な計算は、より現実的なダイナミクスをもたらす。

前方オイラー積分により、各ノードで速度と変位を計算する。

$$v_{t+\Delta t} = v_t + a\Delta t,$$

$$p_{t+\Delta t} = p_t + v_{t+\Delta t}\Delta t,$$

ここで、Δtは小さなタイムステップである。一般に、Δtは、シミュレーションを数値的に安定させるのに十分小さいが、シミュレーションを不必要に遅くするほど小さすぎないように選択する必要がある。

$$\Delta t < \frac{1}{2\pi \omega_{max}} \quad (7)$$

を満たすようにΔtを選択しました。ここで、ωmaxはモデル内の制約の最大固有振動数(??)。
$k_{axial} >> k_{fold}, k_{facet}$及び$k_{axial} > k_{face}$を常に選択するので、この分析では軸制約のみを考慮する。

$$\omega = \sqrt{\frac{k_{axial}}{m_{min}}} \quad (8)$$

ここで、mminはビームの両端の２つのノードの最小質量。

初期条件を仮定する：$v_0 = 0$

上記の式は、数値の散逸(物理学においては、運動などによるエネルギーが、抵抗力によって熱エネルギーに不可逆的に変化する過程をいい、熱力学では自由エネルギーの減少に相当する)が原因で最終的に静的な状態に収束する可能性があるが、これには実用的ではない反復が必要になる場合がある。 メッシュ内の隣接する頂点間に粘性減衰を導入する。

$F_{damping} = c(v_{neighbor} - v)$

ここで、cは粘性減衰(ダンパのこと？)、$v_{neighbor} - v$はノードのその隣接ノード間の相対速度である。シミュレーションサイクルごとの浮動小数点演算を最小限に抑えるために、システム内のすべての制約に対して減衰力を計算するのではなく、この簡略化されたアプローチを使用して減衰する。さらに、全ての制約に臨海減衰があったとしても、構造は全体的に減衰不足[Hiller and Lipson 14]であることがわかった。減衰へのアプローチを徹底することで、より現実的なダイナミクスをもたらす可能性がある。

システムの最も硬い制約(kaxis)に従い、各ノードのcを計算する。

$c = 2\zeta\sqrt{k_{axial}m}$

ここで、ζは減衰比で、mはノードの質量(ここでも1と仮定する)。ほとんどのパターンは、$0.01 \leq \zeta \leq 0.5$で安定してることがわかっている。

3 Implementation

このソルバーは、オープンソース（https://github.com/amandaghassaei/OrigamiSimulator/）、最新のWebブラウザー（http://apps.amandaghassaei.com/OrigamiSimulator/）で実行されるインタラクティブなWebGLアプリに実装された。他の折り紙ソフトウェアとの相互運用性を最大化するために、入出力の内部データ構造としてFOLD形式[Demaine et al. 16]を使用する。また、最終的な折り曲げ角度にマッピングされた不透明度と折り目タイプにマッピングされた色を使用して、SVG入出力をサポートする(**svg形式で展開図を作成し、その際に折り目や折り角度の状態を色やα値で設定できるということ**)。折りたたみ構造のトポロジーについていくつかの条件を仮定することで、ソルバーが折り紙、切り紙、離散化された折り目、駆動されていないヒンジによる制約のないパターン、穴のあるパターン、およびフラットな状態を持たない3D構造をシミュレートできるようにする。剛性と減衰(ダンパ)係数、レンダリングサイクルごとのシミュレーションステップ数など、シミュレーションパラメーターをリアルタイムで制御するためのインターフェイスを提供する（図6）。

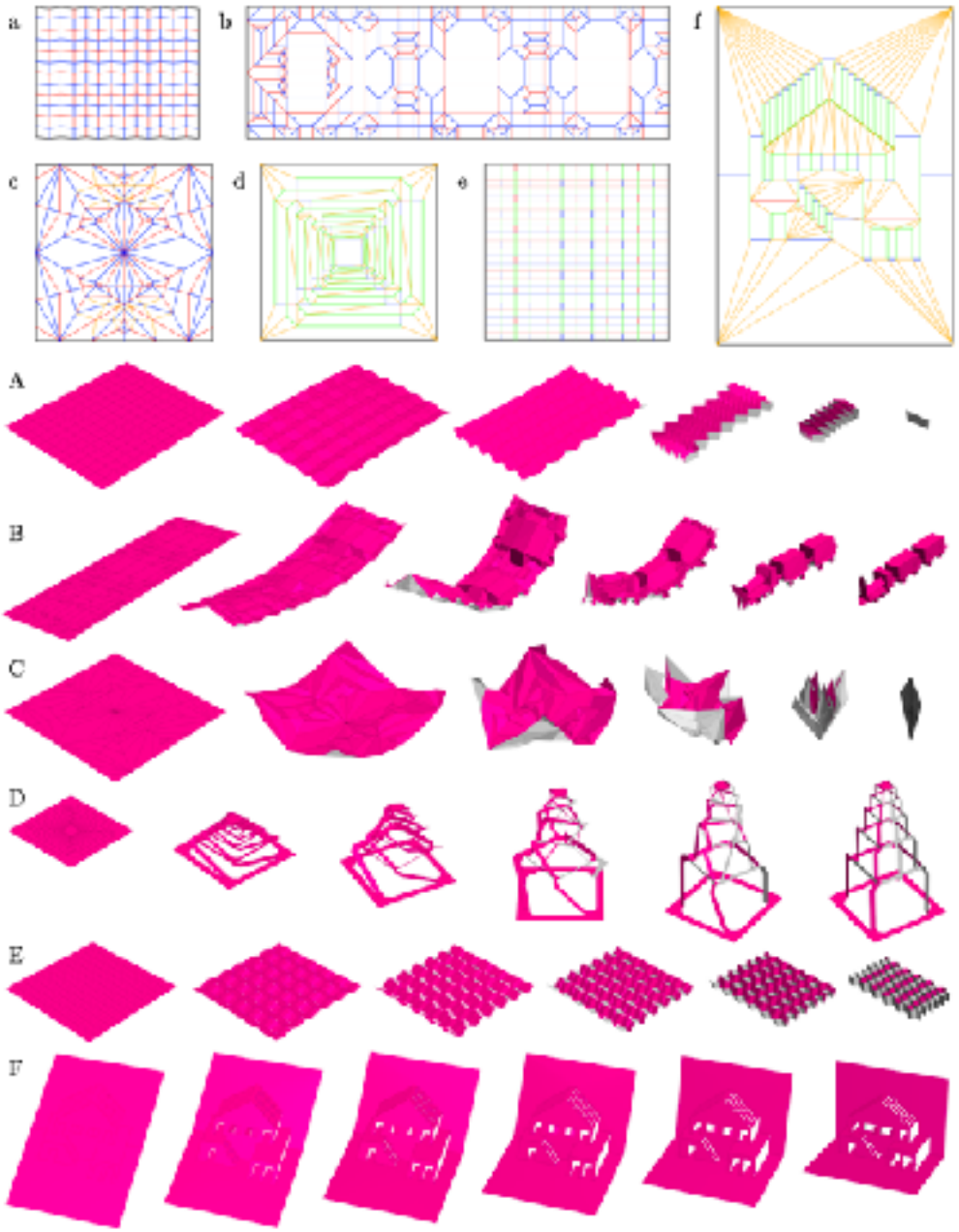

図５：目標折り角度の0, 20, 40, 60, 100%での、シミュレートされた折り紙の静的解。描かれているパターンは、（A）三浦織のテッセレーション、（B）モーザーの列車（ウィリアムガードナーによるリジッドフォールダブルデザイン）、（C）ロバートラングの蘭の花、（D）宮本吉信のRESスクエアタワー、（E）切り紙ハニカム[Saitoh et al. 14]、および（F）Popupologyによるポップアップハウス。上記の対応する折り目パターンと、最終的な折り角度を示す線の不透明度と折り目タイプを示す色：
山（赤）、谷（青）、境界線（黒）、カット（緑）、ユーザー定義のファセットの折り目（オレンジ）。EA = 20、kfold = 0.7、kfacet = 0.7、kface = 0.2。

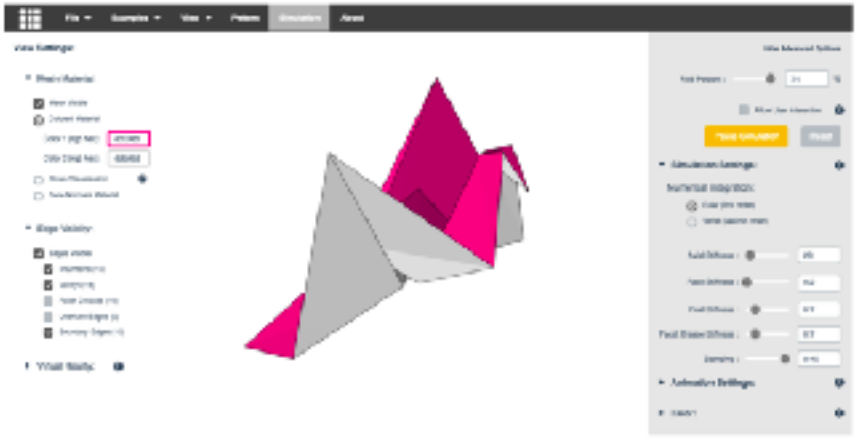

図６：Origami Simulatorアプリケーションのスクリーンショット。

ソルバーが始まる前に、**剛性と減衰係数、および要素間の幾何学的関係のルックアップテーブルを事前に計算**する。**WebGLフラグメントシェーダーで一連のGPUプログラムを実行して、シミュレーションの1ステップを計算**する。

1.メッシュ内の**すべての三角形の面の法線を計算**する（スレッドごとに1つの面）。
2.メッシュの**すべてのエッジの現在の折り曲げ角度を計算**する（スレッドごとに1つのエッジ）。
3.メッシュ内の**すべてのエッジ（スレッドごとに1つのエッジ）について式3〜6の係数を計算**しする。
4.メッシュ内の**すべてのノードの力と速度を計算**する（スレッドごとに1つのノード）。
5.メッシュ内の**すべてのノードの位置を計算**する（スレッドごとに1つのノード）。

他のGPUプログラミングフレームワーク（NVIDIA CUDAなど）では、このソルバーは少ない手順で実装できますが、オペレーティングシステム間の移植性を最大化し、ブラウザーサポートを容易にするために、WebGLシェーダーを使用することを選択肢た。**指定した数のシミュレーションステップの後、ジオメトリを画面にレンダリング**します。図5は、シミュレートされた構造のコレクションを示す。

3.1 Strain Visualization

歪みを視覚化するツールを実装し、ユーザがメッシュ内の局所的に変形した領域を特定できるようにする(図7)。
軸拘束の変位を測定することにより、折り畳まれた折り紙構造の表面全体の工学ひずみを概算します。

ピン結合トラスの所定のビームについて、工学ひずみ ε を$\varepsilon = \frac{l - l_0}{l_0}$で定義します。ここで、lは梁の長さ、$l_0$は梁の公称長さです。

Nビームの節点における工学歪みの大きさは、各ビームからの歪みの絶対値を平均することで計算される:

$$\varepsilon_{node} = \frac{1}{N}\Sigma_{i=1}^{N}\frac{|\Delta l_i|}{l_i}$$

この歪みを青（歪みなし）から赤（高歪み）のスペクトルのRGBカラーに変換し、視覚化のために3Dモデルに適用する。
頂点カラーは、メッシュの面全体で線形補間される。

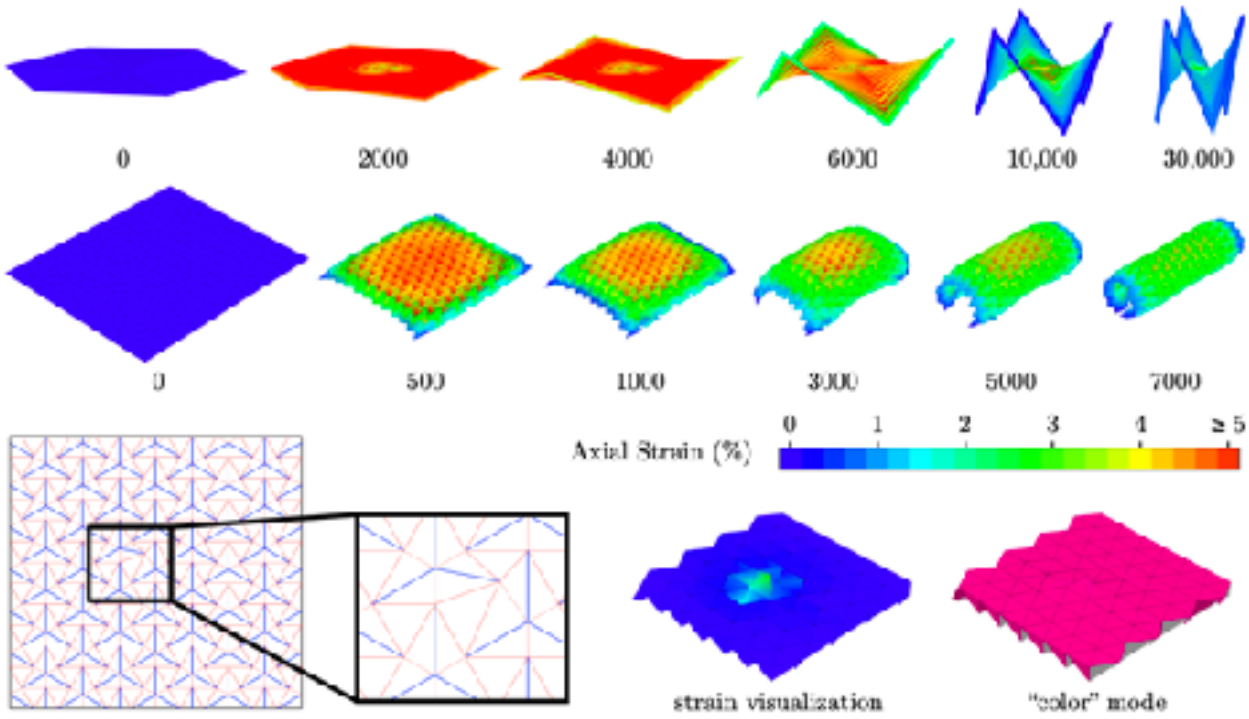

図７：六角形のハイパーバリアント（上部、すべての折り目で±3π / 4に折りたたまれている）およびwaterbombテッセレーション（中央、すべての折り目で±3π / 5に折りたたまれている）の軸ひずみの可視化。これらのシーケンスは、一定の最終折り曲げ角度を時間の増加順に解決する方法を示している。シミュレーションのステップ数は以下の通りである。最初は、両方のモデルが高い内部ひずみのあるほぼ平面の状態に移行し、3次元にカールするにつれてゆっくりと緩和される。（下）Reschテッセレーションの不規則な頂点は、メッシュをマテリアルの「カラー」レンダリングモードでわずかに変形するが、軸歪みの視覚化でははっきりと見える。ハイパーバリアント のシミュレーションパラメータ：
$EA = 100, k_{fold} = 0.7, k_{facet} = 0.7, k_{face} = 1, \zeta = 0.45$。

waterbombとReschテッセレーションのシミュレーションパラメータ：
$EA = 20, k_{fold} = 0.7, k_{facet} = 0.7, k_{face} = 0.2, \zeta = 0.45$。

3.2 User Interaction

シミュレーションの境界条件をリアルタイムで変更することにより、ユーザーインタラクションを紹介する。シミュレーション構造のコンプライアンスにより、ユーザーは双安定パターンの安定状態を切り替えることができる（図8A）。WebVR APIを使用して、HTC ViveおよびOculusバーチャルリアリティ（VR）ヘッドセットとコントローラーでアプリを実行できる（図8B）。このモードでは、ユーザーは両手で折り紙を操作し、没入型3D環境でレンダリングを表示できる。

今後の作業では、設計とシミュレーションの間のループを閉じ、折りたたみ構造をインタラクティブに変更できるようにする(**シミュレーションを止めて、立体形状を何かしら修正できるようにするってことだね**)。

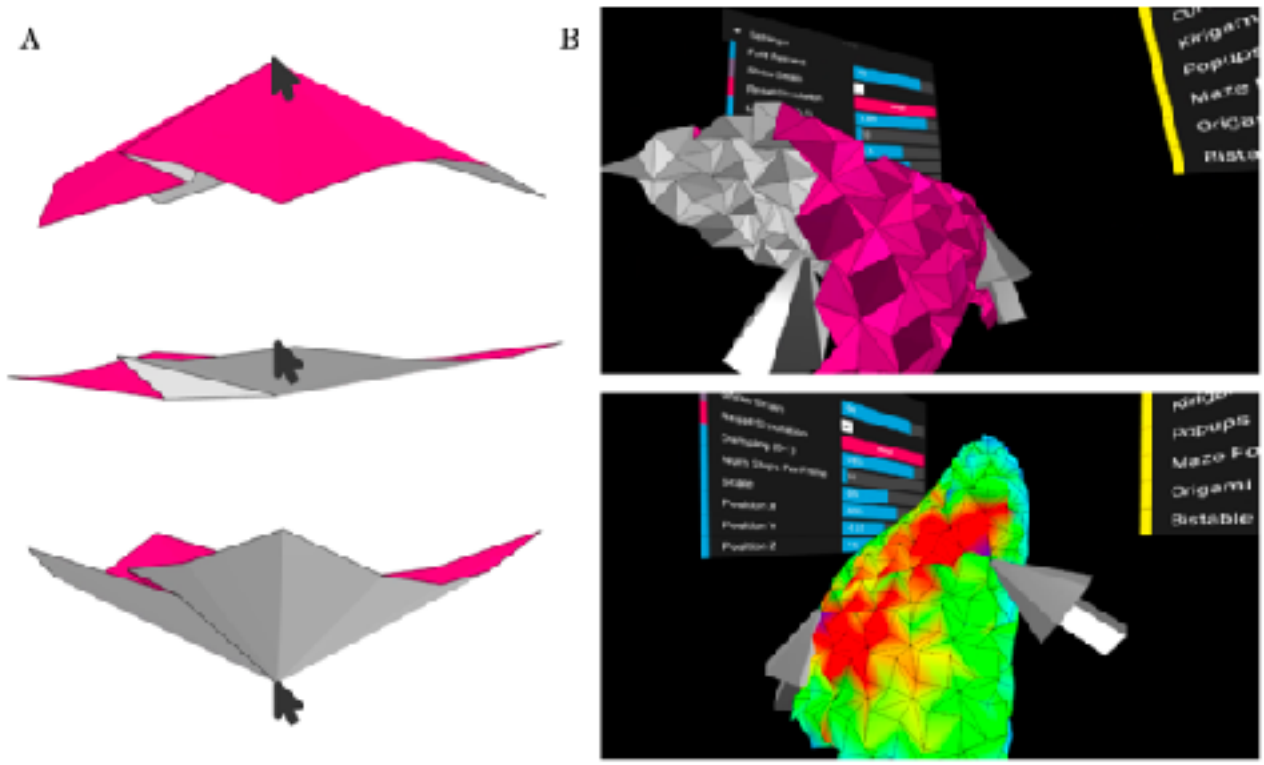

図8：（A）双安定プリーツモデルをある安定状態から別の安定状態にプッシュするユーザーインタラクション。（B）折りたたみハフマン水爆モデルとの直接的なユーザー対話を示す仮想現実インターフェース。（B、下）ひずみの視覚化をオンにすると、ユーザーはひずみをリアルタイムで視覚化しながらメッシュを引っ張ったり押したりできる。

4 Discussion

一連のベンチマークテストを実行して、ソルバーのシミュレーション速度が折り紙モデルのノード数にどのように対応するかを理解した（図9）。 これらのテストは、Chromeブラウザーを使用して、2つの異なるGPUで実行した。348コアと300MHzクロックを備えたIris Graphics 6100と、4096コアと1.1GHzクロックを備えた2つのSLIリンクGeForce GTX 980である。 異なる次

元の一連のn×n三浦織テッセレーションをソルバーにインポートし、レンダリングせずにシミュレーション速度を測定した。
メッシュのノード数（N）が利用可能なGPUコアの数（図9aの青いXで示されている）より少ない間、シミュレーションの1サイクルを計算する時間はほぼ一定であることがわかった。 Nが使用可能なコアを超えると、GPUがスレッドのバッチをキューに入れ、バッチを連続して実行するため、シミュレーション時間のほぼ線形のスケーリングがNで見られる。

4.1 Comparison with Existing Methods

ソルバーの速度を既存の方法であるMERLIN [Liu and Paulino 17]およびフリーフォーム折り紙[Tachi 10]と比較する（図10）。**どちらの方法も、CPU上の三角形の折り紙メッシュの小さな変位を暗黙的に解決**する。

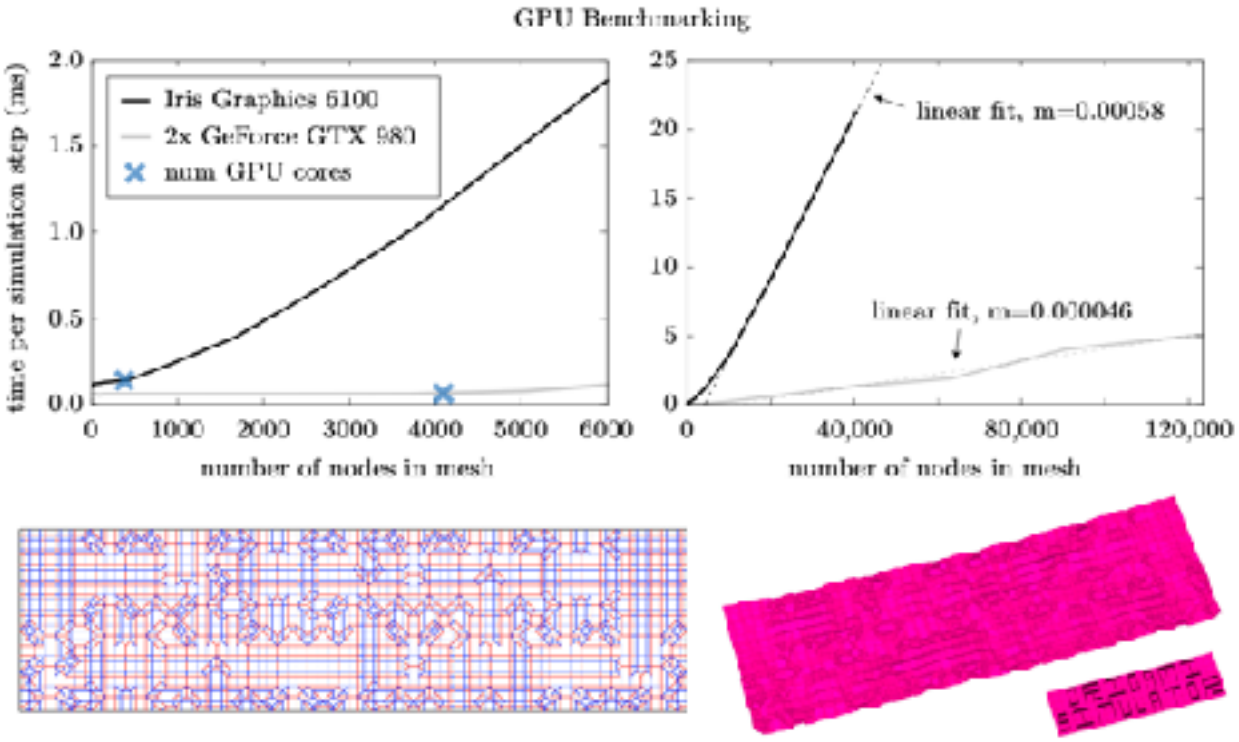

図９：メッシュ内のノード数（N）が増加するときのシミュレーションステップごとのベンチマーク時間。 （左上）ノードの数が使用可能なGPUコアの数（青いXで示されている）より小さい場合、1つのシミュレーションサイクルを完了する時間はほぼ一定です。 （右上）非常に大きなメッシュの場合、シミュレーション時間はNにほぼ線形に比例します。（下）2374ノードのメッシュのリアルタイムシミュレーション。

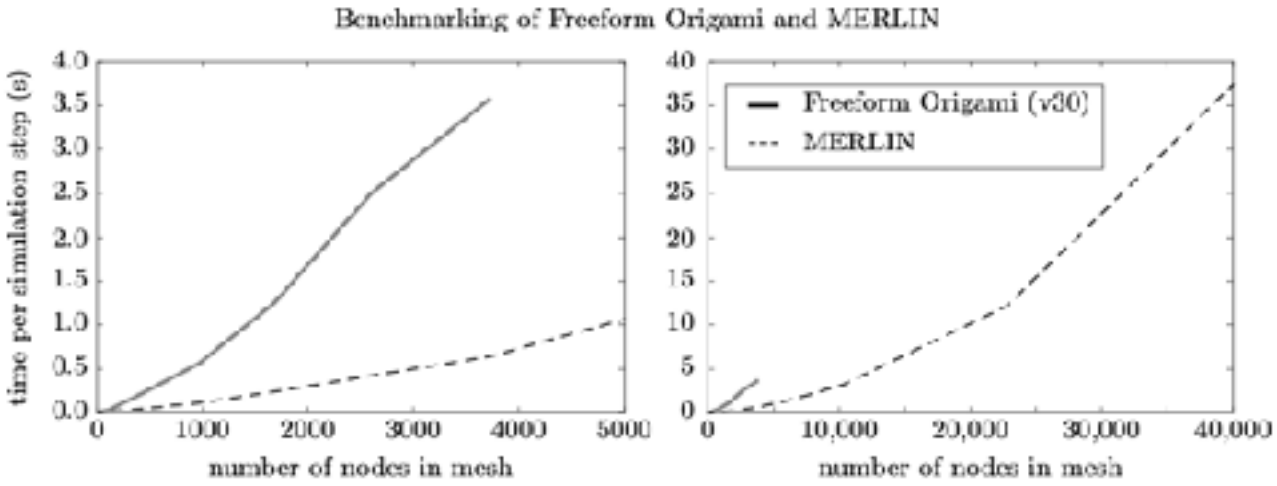

図10：ソルバーの反復時間とフリーフォーム折り紙およびMERLINのメッシュサイズのベンチマーク。

MERLINソルバーの各反復は、グローバルな剛性行列を計算し、修正された一般化変位制御法を使用して、残留力を内部および外部の適用される力に関連付ける方程式系を解く。これらの各ステップを計算する時間は、私たちの方法の単一ステップよりも約1000倍長いが（図10）、MERLINはより硬い材料設定の安定性を維持しながら、より大きなステップを実行する。
(本手法の方が高速、MERLINの方が剛性材料の安定性を維持できる)
10×10セルのミウラオリテッセレーション（121ノード）は、MERLINソルバーの約500回の反復で最終的な折り畳み状態に到達する。（材料の剛性はkfold = 0.1、kfacet = 10,000、EA = 100,000）一方で、我々の方法では、これらのより硬い設定で計算するには約150,000回の反復が必要である。

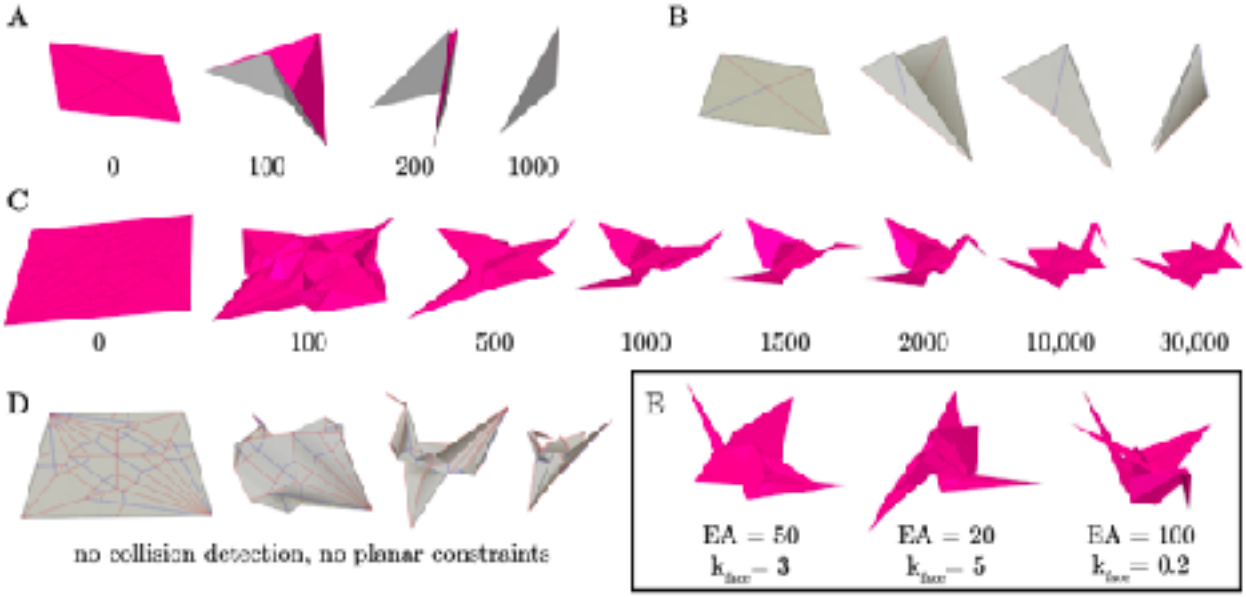

図１１：私たちの方法（マゼンタ）とフリーフォーム折り紙（灰色）の比較[Tachi 10]。
特に明記しない限り、この方法を使用したシミュレーション設定はEA = 20、kfold = 0.7、kfacet = 0.7、kface = 0.2、およびζ= 0.45であり、シミュレーションステップの数はシーケンスの各画像の下に示す。

（A）軸と面の制約のコンプライアンスにより、単純に折り畳まれた頂点を同時に折り畳もうとすると、メッシュの変形が発生する。（B）対照的に、フリーフォーム折り紙は折り目の連続性をより正確に捉えている。
（C）私たちの方法のコンプライアンスにより、ツルなどの非剛体に折りたたみ可能な設計で、最終的な折りたたみ状態を見つけることができる（自己交差を許容）。
（D）フリーフォーム折り紙は、シミュレーションの各ステップでの幾何学的誤差を最小限に抑え、無限に固いメッシュを効果的にモデリングする(**剛体折り可能ってことだね**)。
衝突検出と平面拘束（無限に硬いファセットの折り目）がオフになっていても、フリーフォーム折り紙はクレーンを正しく折りたたむことができない。
（E）同様に、ソルバーの材料剛性を上げると、クレーンが正しい折りたたみ状態に到達できなくなる。

フリーフォーム折り紙は、剛性折り紙シミュレーターである（ただし、ファセットの面外の弾性曲げ変形をモデル化することもできる[Tachi 13]）。図10のデータポイントは、フェイス、エッジ、および頂点のレンダリングが無効で、衝突検出が無効になっているアプリのFPSディスプレイから収集された。Freeform Origamiの各レンダリングサイクルは、共役勾配ソルバーの多くの反復で構成されている。これらは、ベンチマークデータを比較するときに考慮する必要がある。図11は、この方法とFreeform Origamiのモデリングの違いを示しています。Freeformは、単純な頂点の連続した折りたたみをこの方法よりも適切にキャプチャしますが、Freeformの固定拘束により、ツルなどの非固定的に折りたたみ可能なパターンを折りたたむことができない。軸剛性と面剛性が増加すると、我々の方法も同様に失敗する（図11E）。

フリーフォーム折り紙やMERLINの準静的手法とは対照的に、明示的なソルバーの中間ステップは、構造システムのダイナミクスを表現する。質量分布と減衰の現実的なモデルを構築することを第一に考えてはいなが、このソルバーを使用して、妥当な時間依存動作を伴うアニメーションを作成できる（図12）。

Figure 12: *Dynamic animation of a circular pleat pattern with iteration number indicated below. Simulation settings: target fold angle of all mountain/valley creases* $= \pm 7\pi/10$, $EA = 20$, $k_{fold} = 0.7$, $k_{facet} = 0.7$, $k_{face} = 0.2$, *and* $\zeta = 0.2$.

図１２：円形プリーツパターンの動的アニメーション。反復回数は下に示す。
シミュレーション設定：全ての山/谷の折り目の目標折り角度 = =±7π / 10。その他、EA = 20、kfold = 0.7、kfacet = 0.7、kface = 0.2、およびζ= 0.2である。

4.2 Limitations

式7と8によると、幾何学的精度（拘束剛性）とシミュレーション速度（Δt）の間にはトレードオフがあります。各ステップで計算に多くの計算労力が必要な場合でも、ステップサイズを大きく維持しながら、非常に硬いシステムをモデル化できるため、暗黙の定式化はこの点で有利だ。
これを緩和するために、剛性パラメーターへの簡単なアクセスを提供し、モデルが静的解に近いときにユーザーが制約剛性を高めて、シミュレーションの初期段階で不要な計算時間を回避できるようにする。

他のバーアンドヒンジシミュレーション手法では一般的であるように、シミュレーションの動作はメッシュの離散化に敏感である。これは、任意の多角形面を三角形分割する複数の方法が存在する可能性がある（セクション2.1）ファセットのメッシュで特に顕著にみられる。
特定の面外曲げモードが必要な場合は、ユーザーがデザインをアプリにインポートする前に、ファセット折り目を手動で定義できる（図13）。Filipovらは、折り紙のバーアンドヒンジモデルにおける面内および面外変形に対するファセットメッシュの影響をより詳細に説明する[Filipov et al. 17]。

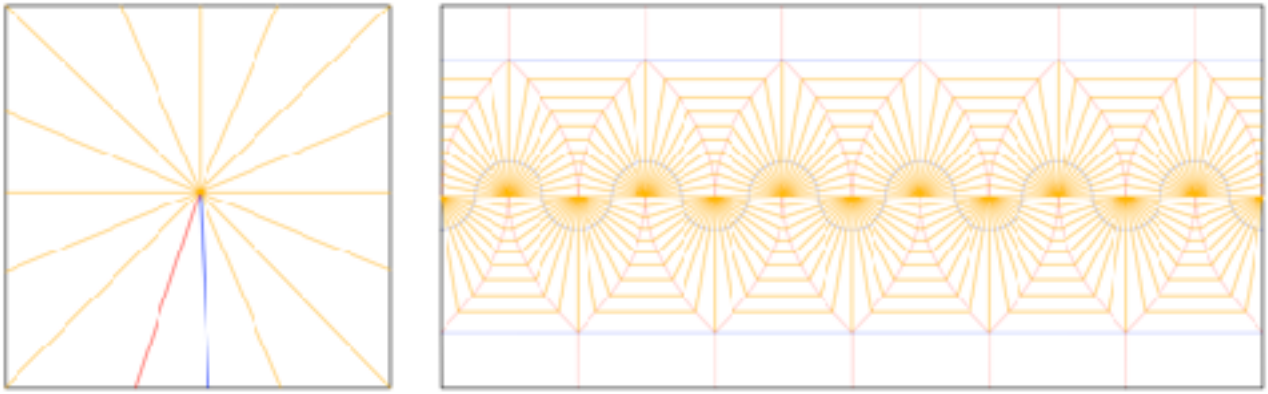

図１３：曲線の折り目パターンのルーリングに沿ったユーザー定義のファセット折り目(黄色いやつ)は、山/谷の折り目間の領域の予想される面外曲げをガイドする。

高アスペクト比の三角形が変形して、2つのエッジの外積で定義される面法線が向きを反転させるほど変形すると、シミュレーションで問題が発生する可能性がある。この問題は私たちの方法に固有のものではないが、制約の遵守によって悪化する可能性がある。この不安定さは、慎重に選択されたユーザー定義のファセット折り目によって、または折り曲げ剛性と比較して軸方向および面剛性を増加させることによって緩和される。

最後に、この方法では、ソルバーへの入力として、各折り目に対して目標の折り曲げ角度が必要だ。ターゲットの折りたたみ角度が正しくない場合、最終的な折りたたみ状態が変形する可能性があるが、これは通常、歪みを視覚化することで理解できる。

5 Future Work

GPUでの並列化は、セクション2.5で導入された明示的な統合方法や一般的なバーアンドヒンジモデルに固有のものではない。 実装が単純であるため、このペーパーで概説されている方法から始めることを選択したが、（4.2で説明したように）現在のアプローチには制限がある。 近い将来、GPUの並列化をバーアンドヒンジモデルの暗黙の定式化に適用して、より剛性の高い構造システムを非常に効率的にモデル化できるようにする予定である。 GPUアクセラレーションにより、より複雑なFEMをリアルタイムで実行することも可能になり、インタラクティブな折り紙の設計、最適化、および分析ツールの新たな可能性が開かれる。

将来の作業も、衝突検出、適応型時間ステップ、より堅牢なメッシュ三角形分割法、およびアプリへのデザインインターフェースの組み込みを含むように成長する可能性がある。