Vol. 48 No. 4 情報処理学会論文誌 Apr. 2007

布と高解像度のモデルとのリアルタイム衝突計算

原 田 隆 宏† 越 塚 誠 一†

本論文では変形する高解像度のモデルと布のリアルタイム衝突計算手法を提案する．本手法は布と物体との衝突計算において距離関数を用いる．距離関数の計算において離散境界という概念を導入し，ポリゴンモデルを離散境界に分解することによって境界の次元を下げる．本手法はポリゴン数が距離関数計算時間に与える影響は小さいため，高解像度のモデルを用いても高速に距離関数の計算を行うことができる．またポリゴンモデルのトポロジには制約条件を課さないため，不整合なデータを含むようなポリゴンモデルからでも距離関数を計算することができる．さらに離散境界の空間解像度を変化させることによって計算時間を制御することも可能である．布の時間積分には陽解法を用いたが，陽解法は陰解法のように数値的に安定な状態ではない．そこで計算精度と数値安定性の向上のためサブタイムステップアルゴリズムを導入した．本論文で提案する距離関数計算手法は GPU を用いて高速化することが可能である．また布の計算および衝突計算にも GPU を用いることが可能である．

Real-time Collision Computation between Cloth and Highly-tessellated Models

Takahiro Harada† and Seiichi Koshizuka†

In this paper, we propose a method that can calculate collision responce between a cloth and a deforming polygon model in real-time. Our method uses a distance function for collision computation. We introduce discrete boundaries for calculation of a distance function. Decomposition of polygon model into discrete boundaries reduces the dimension of the boundary. Since the number of polygons has little effect to the computation time in our method, a distance function can be calculated in a short time even from a highly tessellated model. Our method can compute a distance function from the models involving inconsistent data because it does not need any requirement about topology of a polygon model. Moreover, it can control the computation time with varying the spatial resolution of discrete boundaries. For time integration of a cloth an explicit method is employed. An explicit method does not guarantee stable computation although an implicit method does. Thus to improve the calculation accuracy and numerical stability, we introduce a sub-time step algorithm. The distance function computation method proposed in this paper can be accerelated with GPUs. As we are also able to use GPUs for the computation of a cloth motion and collision also.

1. 背　　景

コンピュータグラフィックスにおいて布は様々なシーンで用いられ，物体の形状に合わせて変形する．布を有限個の質点群で近似するならば，この変形の計算には布を構成する各質点に対してシーンに存在する物体との衝突を検出しなければならないため，高コストである．静止している物体と布とのリアルタイムシミュレーションは比較的容易に計算することができるが，布が接している物体がすべて静止している状況は少ない．動きながら形状を変える物体と布とのリアルタイム衝突計算はまだ困難な問題の 1 つである．

† 東京大学
The University of Tokyo

距離関数とはある空間において境界条件が与えられたときに境界の最近点までの距離を表す関数であり，衝突計算に用いることができる．この距離関数は衝突判定を行えるだけでなくこの関数の空間一階微分はその最近点に向かうベクトルであるため，衝突の解消のために加えなければならない力のベクトルを求めることができる．

本論文では変形する物体と布のリアルタイム衝突計算手法を提案する．変形する物体から実時間で距離関数を計算し，これを用いて布との衝突計算を行う．本研究では離散境界という概念を導入する．まず物体の表面を離散境界に分解し，この離散境界を用いて距離関数を計算する．布のシミュレーションでは陽解法を用いるが，陽解法は計算を数値不安定にすることがある．この点を克服するために，サブタイムステップア

ルゴリズムを導入した.

本手法は距離関数の計算においてトポロジに関して制約条件を課さないので，ポリゴンが完全に閉じていない物体やポリゴンの表裏が反転している部位を持つ物体に対しても衝突計算を行うことができる．また本手法は GPU を用いて高速化することが可能である.

2. 関連研究

本提案手法は距離関数をリアルタイムに更新し，布との衝突計算を行っている．本章では距離関数と布の衝突計算についての関連研究を見ていく.

2.1 距離関数

距離関数に関する研究は多く行われており，Jones らがレビュー論文で触れている[8)].

距離関数の計算は計算負荷が高いため，近年 GPU を用いた研究が行われている．Hoff らは 2 次元でのボロノイ図を 3 次元形状のラスタライズによって求める手法を提案し[7)]，ボロノイ領域と距離関数は密接な関係にあることを用いて，距離関数計算を行っている．この手法は 3 次元格子を 2 次元格子の集合に分解し，サイトの形成する 3 次元形状を生成することによって各 2 次元格子でのボロノイ図を求めている．そのため精度の良い距離関数を求めるには十分細かい 3 次元形状を作成しなければならない．この手法の多量のポリゴンを生成しなければならない問題点に対して Sud[16)] らは改善手法を開発した.

Mauch は Characteristics/Scan-Conversion（CSC）algorithm を用いてポリゴンの面を押し出すことで多面体を作成し，距離関数を計算している[10)]．この手法では多量の 3 次元形状を作成しなければならない．Mauch の実装は CPU を用いた実装であったが，Sigg らは GPU を用いてこの手法の高速化を行った[14)]．CSC algorithm では多くの形状を作成しなければならなかったため，Sigg ら[14)] はこれらを減らす手法を提案している．しかしこの手法を用いて作成される多面体内部の距離関数は線形分布をしていないため，GPU を用いた計算ではフラグメントシェーダ内部で複雑な処理を行う必要があった.

このように生成される多面体内部だけでなく，一般的にあるサイトからの平面上の距離関数の分布は非線形である．そこで Sud らはサイトに属する領域においてサイトに向かうベクトルは線形に分布しているということに注目したアルゴリズムを提案している[15)]．この手法は Hoff らの手法[7)] のように高精度の距離関数を求めるために多くのポリゴンを生成する必要がない.

このように距離関数を求めるには多くの形状の構築を行わなければならず，多大な情報を GPU に送り処理しなければならない．またすべての既存研究では距離関数の構築の計算コストは物体を形成しているポリゴン数に比例した処理となっており，ポリゴン数が増えれば計算コストは大きく増えてしまうという共通の欠点がある.

2.2 布の衝突計算

布の研究も多く行われており，レビュー論文でそれらについて触れられている[1),9),17)]．布を質点の集合体で近似すると布と物体の衝突計算を行うためには布を構成する質点ごとに物体の各面に対して衝突判定を行わなければならないため，計算コストが高い.

バウンディングボリュームは剛体の衝突判定によく用いられるが，布など変形する物体の計算にも用いられている．しかし形状の変形にともないバウンディングボリュームを修正もしくは再構築しなくてはならない[12)].

布と衝突する形状をボクセルで表現し，衝突判定を行う手法がある[11)]．物体の内外の情報のみ持つボクセルを用いる場合には衝突を解消するために布に加えなければならない力は別に求める必要がある．また変形する物体との衝突計算においてはボクセルデータを再構築しなければならない．空間を構成する多くのボクセルの物体の内外判定を実時間で行うのは難しい.

布と衝突を計算する物体の形状が球体など単純なものならば，その表面は簡単な式で表すことができ，その物体との衝突計算も容易である．Fuhrmann らは衝突判定では布が衝突する形状をこのような形状のみに限定している[5)]．Cordier らも布と人間の足の衝突計算を，足を円柱と近似して同様な手法を用いて計算している[2)]．これらの手法は複雑な形状の物体との衝突計算を行うことはできない.

Vassilev らはイメージ空間上での判定を行う手法を開発した[18)]．この手法は物体の形状を深度として取得し，布との衝突判定をこの空間で行っている．しかしこの手法は物体の深度を取得するときの方向に物体の重なりが存在している場合に物体のすべての表面位置を取得することができないため，このような場合に用いることはできない.

Fuhmann らは布と物体との衝突計算において距離関数を用いた[4)]．距離関数を用いることでどのような物体の表面形状も表すことが可能なので，それらに対して衝突計算を行うことができる．この論文では距離関数の計算手法が開発されているが，やはり距離関数をリアルタイムで計算するのは難しいため，前処理として距離関数を求めている.

3. 距離関数の計算

ある境界が定義されたとき，空間の点から境界までの距離を表した関数を距離関数という．**図 1** はある境界とそれによって計算される距離関数を示したものであり，距離関数の値を色で示している．

ここで境界 s によって求まる $\mathbf{x}$ での距離関数を $d(\mathbf{x}, s)$ とする．境界 s が 2 つの境界 s_0，s_1 に分解されるとき，ある点 $\mathbf{x}$ での距離関数は最も近い境界までの距離となる．つまり距離関数 $d(\mathbf{x}, s)$ は以下のように分解することができる．

$$\begin{aligned} d(\mathbf{x}, s) &= d(\mathbf{x}, s_0 + s_1) \\ &= g(d(\mathbf{x}, s_0), d(\mathbf{x}, s_1)) \end{aligned} \tag{1}$$

ここで関数 g は以下のように定義される．

$$\begin{aligned} &g(f(\mathbf{x}, s), f(\mathbf{x}, t)) \\ &\quad = \begin{cases} f(\mathbf{x}, s) & d(\mathbf{x}, s) \leq d(\mathbf{x}, t) \\ f(\mathbf{x}, t) & d(\mathbf{x}, t) < d(\mathbf{x}, s) \end{cases} \end{aligned} \tag{2}$$

ここで境界 s によって求まる $\mathbf{x}$ でのある関数を $f(\mathbf{x}, s)$ とする．距離関数 d も関数 f の 1 つである．

この性質を用いて 3 次元ポリゴンモデルを境界とした問題を考える．あるポリゴンモデルの境界 t は n 個のポリゴン $\{f_0, f_1, f_2, \cdots, f_n\}$ で構成されているとする．すると式 (1) より t を境界条件とする距離関数 $d(\mathbf{x}, t)$ は

$$\begin{aligned} d(\mathbf{x}, t) = g(&d(\mathbf{x}, f_0), d(\mathbf{x}, f_1), d(\mathbf{x}, f_2), \\ &\cdots, d(\mathbf{x}, f_n)) \end{aligned} \tag{3}$$

と表すことができる．

同様に 1 つのポリゴン f_i はそれを構成する m 個の点要素 $\{p_{i,0}, p_{i,1}, p_{i,2}, \cdots, p_{i,m}\}$ に離散化することができる．この点要素を離散境界と呼ぶ．この f_i を境界条件とする距離関数 $d(\mathbf{x}, f_i)$ は式 (1) を用いて近似することができる．

$$\begin{aligned} d(\mathbf{x}, f_i) \approx g(&d(\mathbf{x}, p_{i,0}), d(\mathbf{x}, p_{i,1}), d(\mathbf{x}, p_{i,2}), \\ &\cdots, d(\mathbf{x}, p_{i,m})) \end{aligned} \tag{4}$$

この近似による式変形は境界 t を構成するすべてのポリゴンに対して用いることができる．**図 2** に境界の分解の手順を示す．

ここで $d(\mathbf{x}, s)$ を空間微分し

$$\mathbf{v}(\mathbf{x}, s) = \frac{\partial d(\mathbf{x}, s)}{\partial \mathbf{x}} \tag{5}$$

と表す．この $\mathbf{v}(\mathbf{x}, s)$ を勾配ベクトルと呼ぶ．この勾配ベクトルは点 $\mathbf{x}$ から最も近い境界までのベクトルとなることが知られている．勾配ベクトル $\mathbf{v}(\mathbf{x}, s)$ は式 (1) と同様に

$$\begin{aligned} \mathbf{v}(\mathbf{x}, s) &= \mathbf{v}(\mathbf{x}, s_0 + s_1) \\ &= g(\mathbf{v}(\mathbf{x}, s_0), \mathbf{v}(\mathbf{x}, s_1)) \end{aligned} \tag{6}$$

と分解することができる．ポリゴンモデルの境界 t から求まる勾配ベクトルもその要素であるポリゴンを境界条件として求まる勾配ベクトルを用いて

$$\begin{aligned} \mathbf{v}(\mathbf{x}, t) = g(&\mathbf{v}(\mathbf{x}, f_0), \mathbf{v}(\mathbf{x}, f_1), \mathbf{v}(\mathbf{x}, f_2), \\ &\cdots, \mathbf{v}(\mathbf{x}, f_n)) \end{aligned} \tag{7}$$

と表すことができ，ポリゴン f_i の勾配ベクトルも点要素による勾配ベクトルを用いて

$$\begin{aligned} \mathbf{v}(\mathbf{x}, f_x) \approx g(&\mathbf{v}(\mathbf{x}, p_{i,0}), \mathbf{v}(\mathbf{x}, p_{i,1}), \mathbf{v}(\mathbf{x}, p_{i,2}), \\ &\cdots, \mathbf{v}(\mathbf{x}, p_{i,m})) \end{aligned} \tag{8}$$

と近似することができる．

これらの変形は距離関数を求める際の境界条件の次元を落とす変形である．式 (3)，(7) は 2 次元曲面の境界 t を 2 次元平面の境界群に分解する変形である．さらに式 (4)，(8) は 2 次元平面の境界を零次元の境界群に分解する変形である．2 次元曲面の境界条件のもと解かなければならなかった問題が零次元の境界条件を持つ問題に変形されたことにより，解を求める計算が容易になる．零次元要素を境界とする問題は，点と点の距離を計算することのみによって距離関数を求めることができる（**図 3**）．**図 4** に求めた距離関数の一例を示す．

今までの議論は空間において連続に距離関数が分布する場合を前提としてきたが，一般的には計算領域を離散化して離散距離関数を用いることが多い．空間上での 1 枚の平面と 1 点が存在する場合を考える．するとその面上では点を境界とする距離関数は非線形な分布になる．格子点での離散距離関数を求めた後に格

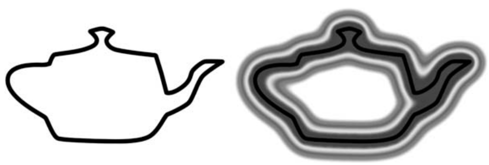

図 1 境界（左）と距離関数（右）．右図は距離関数を色で表しているものである

Fig. 1 Boundary (left) and a distance function (right). A distance function is represented by color in the right figure.

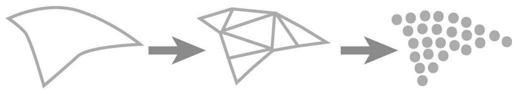

図 2 境界の分解．1 段階目では 2 次元曲面境界を 2 次元平面境界に分解する．2 段階目ではさらに離散境界に分解する

Fig. 2 Decomposition of boundary. The first step decomposes 2 dimensional curved boundary to 2 dimensional plane boundaries. The second step decomposes to discrete boundaries.

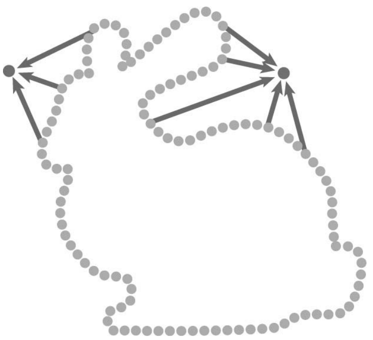

図 3 離散境界を用いた距離関数の計算
Fig. 3 Distance function calculation using discrete boundaries.

図 4 計算された距離関数．ある面での距離関数を値によって色で示す．距離関数は図右下のカラーバーのように距離は色づけされている
Fig. 4 Calculated distance function. This figure shows a distance function at a slice. The distance function is colored with color shown at the bottom right.

子点以外の位置での距離関数を求めるときに，格子点での値から線形補間したのでは正しい値は得ることができない．しかし Sud らが述べているように，平面上での勾配ベクトルは線形に分布している[15]．そのため格子点以外の位置での値は勾配ベクトルを線形補間し，それらの大きさを計算することにより距離関数は正確に計算することができる．**図 5** 右に示すようにある境界 p があり，2 点 v_0，v_1 において勾配ベクトルと距離関数を求める．そして線分 v_0，v_1 での距離関数を勾配ベクトルを線形補間してそのベクトルの大きさとして距離関数を求めたものと，2 点での距離関数を線形補間して求めた解を図左に示す．グラフでは理論解を実線で示している．このように勾配ベクトルを線形補間することによって距離関数は正確に求めることができるが，距離関数の線形補間では誤差が生じる．格子点での距離関数を求めた後，格子以外の位置での距離関数を求めるときに格子点での距離関数を線形補間するか勾配ベクトルを線形補間し距離関数を求めるかは使用事例に応じて使い分けるべきである．

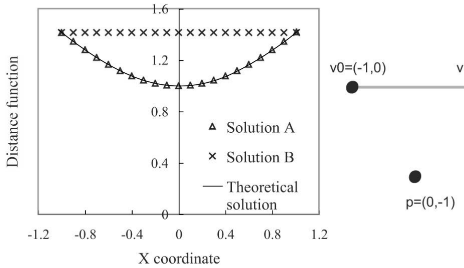

図 5 線分 v0，v1 上での距離関数を勾配ベクトルを線形補間して求めた解（Solution A）と距離関数を線形補間して求めた解（Solution B）．実線は理論解である．
Fig. 5 Solution A is distance function calculated by interpolating gradient vectors and solution B is distance function calculated by interpolating distance values on v0 and v1. The line in the graph shows theoretical solution.

本手法を用いて距離関数を求めるときには離散距離関数を求める．そのため面要素から離散境界を求める際に空間に均一に分布する離散境界を用いるのが，ある計算誤差における分解では最も計算効率が良い．空間的に離散境界の疎密が存在すると全体の計算誤差の最大値は疎な部分におけるものとなり，密な部分を作成しても最大誤差は減少しない．このように空間的に均一な離散境界を求めることによって，距離関数計算のコストは離散境界の数に比例するものとなる．すなわち境界 t の面積に比例する計算である．面の分割の細かさのみ異なる同じ形状を表した 2 つの物体があるとすると，それらの物体から生成される離散境界は同じ数となり，距離関数計算のコストは面の数によらず物体の表面積に比例する計算とすることができる．

4. 布シミュレーション

4.1 モ デ ル

本研究では布のモデルにバネ質点モデルを用いた．このモデルは布を質量を持った粒子群として近似し，それらをバネでつないだものである．粒子間のバネは上下左右の粒子間に働くバネと斜め方向の粒子間に働くバネの 2 種類を用いた（**図 6**）．上下左右の粒子を接続しているバネのバネ定数を k_{adj}，斜めの粒子を接続しているバネのバネ定数を k_{diag} とすると，粒子 i に働く力 F_i は

$$\mathbf{F}_{i,spring} = \Sigma k_{adj}(|\mathbf{x}_{ij}| - l_{adj})\frac{|\mathbf{x}_{ij}|}{\mathbf{x}_{ij}} + \Sigma k_{diag}(|\mathbf{x}_{ij}| - l_{diag})\frac{|\mathbf{x}_{ij}|}{\mathbf{x}_{ij}} \qquad (9)$$

となる．l_{adj}，l_{diag} はそれぞれバネの初期状態の長さである．

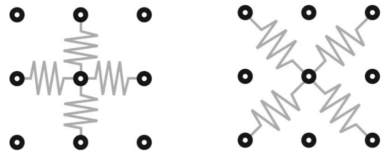

図 6 布モデルに用いた 2 種類のバネ．上下左右の粒子間を接続するバネ（左）と斜めの粒子間を接続するバネ（右）

Fig. 6 Two kinds of springs. Springs connecting adjacent particles are shown in the left and those connecting diagonal particles are shown in the right.

バネ質点モデルを用いた布シミュレーションでは，これらの 2 種類のバネのほかに質点を 1 つ飛ばしたバネを用いることがある[13]．本論文ではリアルタイムシミュレーションに焦点を絞っているので計算コストを下げるため，このバネは導入しなかった．

4.2 時間積分

最も簡単な時間積分法はオイラー陽解法であるが，この手法は数値安定性に欠け不安定な結果を引き起こしやすい．二次精度の Leap-frog 時間積分法は高次精度の時間積分法よりも計算方法が容易であり，オイラー陽解法よりも安定性が高い．そのため，リアルタイムシミュレーションに向いているため，本研究では Leap-frog 時間積分法を用いた．時間積分は以下のように計算される．

$$\mathbf{x}_i^{t+dt} = \mathbf{x}_i^t + d(\mathbf{x}_i^t - \mathbf{x}_i^{t-dt}) + \frac{\mathbf{F}_i dt^2}{m_i} \qquad (10)$$

$\mathbf{x}_i^t$ は質点 i の時間 t での位置，$\mathbf{F}_i$ は質点 i に働く力，m は質点の質量，d は減衰係数である．Leap-frog 時間積分法を用いると質点の速度を用いるかわりに時間 $t-dt$ の位置を用いるため，その座標を保持しておく必要がある．

5. カップリング

距離関数を衝突計算に用いることで，衝突判定と衝突による反応も同時に計算することができる．本手法では Fuhmann らの用いた手法をもとにしており[4]，数値安定性の向上のためサブタイムステップアルゴリズムを導入した．また時間軸上での距離関数とその勾配ベクトルのアンチエイリアシングを行うことで布の挙動の不自然さを軽減させた．

5.1 距離関数を用いた衝突計算

布が物体の表面から ϵ 以下の距離に存在する場合，布が物体に衝突したと判定して処理を行う（**図 7**）．この処理では布が実際に物体の表面に当たっていないため近似計算となるが，このような処理を行うことで布が物体へめり込むのを防ぐことができる．しかしこの手法の欠点は布がつねに物体の表面より少し離れているように見えることである．

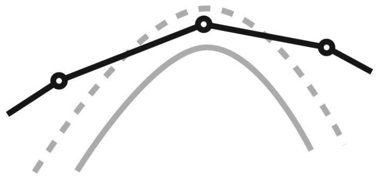

図 7 衝突判定

Fig. 7 Collision detection.

布を構成するある質点 i の位置 $\mathbf{x}_i$ における物体までの距離を d とし，その点での物体の最近点までのベクトル，つまり勾配ベクトルを $\mathbf{n}$ とし，この粒子 i の物体までの距離を ϵ に戻す力を加える．

$$\begin{aligned}\mathbf{F}_{i,resp} &= \frac{m_i}{dt^2}\Delta\mathbf{x}_i \\ &= -\frac{m_i}{dt^2}(\epsilon - d)\mathbf{n} \qquad (11)\end{aligned}$$

この力 $\mathbf{F}_{i,resp}$ を布に衝突した質点 i に加えることで物体の表面から質点までの距離を ϵ に戻すことができる．

物体の表面に衝突した質点には一般的に動摩擦力が働き，その質点に働く力の物体上での接線方向の力を打ち消すように働く．粒子 i に働く動摩擦力は粒子 i に働く力 $\mathbf{F}_i$ の物体の接線方向の力を打ち消すように働くので

$$\mathbf{F}_{i,t} = \mathbf{F}_i - b(\mathbf{F}_i - \mathbf{F}_i \cdot \mathbf{n}) \qquad (12)$$

となる．なお b は動摩擦係数である．本研究では静止摩擦力に関しては考慮していない．

距離関数の計算において 3 次元格子点上の距離関数と勾配ベクトルを求めた．物体と布との衝突計算においては各質点の位置 $\mathbf{x}_i$ でのそれらの値を求めなければならないので，質点を囲む 8 格子点での値を用いて線形補間を行うことで求めた．

5.2 サブタイムステップ

本研究での物体と布の相互作用は 1 方向であり，物体は布に影響を及ぼすが布は物体に影響を及ぼさない．物体は物理的な力を受けないため，どのような時間刻み幅を用いても不安定になることはない．しかし布は布を構成する質点間のバネの力以外にも，物体と衝突するときに力を受けるため，その力の大きさにより数値的に不安定になることがある．また Leap-frog 時間積分法はオイラー陽解法に比べ数値安定性は良いが，陰解法ほど安定ではない．そこで Leap-frog 時間積分法の計算精度と数値安定性の向上のために，サブタイムステップアルゴリズムを導入した．このサブタイムステップアルゴリズムは物体の刻む時間刻み幅を小さな時間刻み幅（サブタイムステップ）に分割する．そ

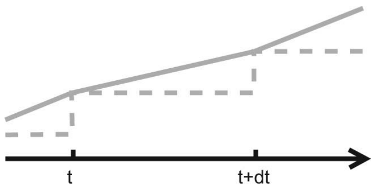

図 8 距離関数の時間アンチエイリアシング
Fig. 8 Temporal antialiasing of a distance function.

してサブタイムステップを用いて布の計算を行い，物体の運動にはタイムステップを用いて計算する．このサブタイムステップを用いることで数値安定性を向上させることができる．

5.3 時間アンチエイリアシング

サブタイムステップアルゴリズムを用いることで布の計算は改善されたが，物体はサブタイムステップを刻む間に位置の更新を行わない．すると距離関数の更新もタイムステップで行うので距離関数とその勾配ベクトルは時間軸上においてステップ状の関数となる．

このような離散的な関数を用いると布の挙動に不自然な結果が生じてしまうことがある．このような結果を防止するためサブタイムステップを刻む間は，求めた距離関数の時間軸上での線形補間を行い，スムーズな関数にする．距離関数と勾配ベクトルをこのように線形補間することで布の挙動の不自然さは解消される．n 回のサブタイムステップを刻むときに j 回目のサブタイムステップでの質点 i の座標 $\mathbf{x}_i$ での距離関数 $d(\mathbf{x}_i)^{t+\frac{j}{n}dt}$ と勾配ベクトル $\mathbf{v}(\mathbf{x}_i)^{t+\frac{j}{n}dt}$ は

$$d(\mathbf{x}_i)^{t+\frac{j}{n}dt} = \left(1-\frac{j}{n}\right)d(\mathbf{x}_i)^t + \frac{j}{n}\,d(\mathbf{x}_i)^{t+dt} \qquad (13)$$

$$\mathbf{v}(\mathbf{x}_i)^{t+\frac{j}{n}dt} = \left(1-\frac{j}{n}\right)\mathbf{v}(\mathbf{x}_i)^t + \frac{j}{n}\,\mathbf{v}(\mathbf{x}_i)^{t+dt} \qquad (14)$$

と求めることができる（**図 8**）．

6. 実　　装

本手法の距離関数の計算は GPU を用いることで高速化することができる．そして距離関数はグラフィックスメモリ上に保持される．布の計算も GPU を用いて高速化することができる．

6.1 離散境界の構築

距離関数の計算を行う前に入力として与えた境界 p を離散境界に分解しなければならない．空間上に均一な離散境界を計算するため，あらかじめ計算領域を格子状に分割し，それらの格子点上に離散境界を配置する．この処理は 3 次元空間内でのポリゴン物体の表面のボクセル化にあたる．ここでは Dong らが開発した表面のボクセル化手法を用いて境界の離散境界への分解を行う[3)]．

Dong らはバッファの RGBA 各チャネルに 8 ビット用意しているが，ブレンディングを行うことができるのは 1 チャネル 16 ビットまでなので，本研究では 16 ビットを用いた．こうすることで 1 枚のバッファに格納できるデータ量が増えるため，レンダリング回数を減らすことができる．ラスタライズでは視線ベクトルと平行な面の情報を抽出することはできないので，この手法では XYZ 3 方向からラスタライズした結果を統合する必要がある．Dong らはビット参照テーブルを用意していたが，16 ビットのバッファを用いるときのビット参照テーブルは $65{,}539 \times 16$ の大きさになってしまう．このデータ量は 1 枚のバッファで表現できず，GPU 上では効率的に処理することができないので，CPU に読み戻しデータの統合を行った．3 次元ボクセルデータはビット圧縮されているので読み戻すデータ量は少ない．

この手法は物体の表面をボクセルの精度で離散化する．そのため，角度を持った面などを正確に表すことができない．より精度の高い計算のために物体の表面をサブボクセルの精度で分割する手法を開発する必要がある．

6.2 距離関数の計算

ポリゴンモデルの境界から離散境界を求めることができたので，これを用いて距離関数を計算する．すべての離散境界において距離関数を求め，境界 m からの距離関数を構築する．GPU は 3 次元格子での値を書き出すことができないので，3 次元格子を複数枚の 2 次元格子の集合として出力する．この 2 次元格子をスライスと呼び，1 枚の大きな 2 次元格子に敷き詰めることにより 3 次元格子を 2 次元格子で表現することができる．ここで 2 次元格子座標 u，v を 3 次元格子 x，y，z に変換する関数 $t(u,v)$ を導入する．x，y，z は $[0,1]$ の値をとるとする．2 次元格子は u，v 軸方向にスライスを l 枚保持し u，v それぞれは $[0,1]$ の値をとるものとすると 3 次元格子座標 x，y，z を出力する関数 $t(u,v)$ は以下のように表される．

$$x = \frac{1}{s}(u - s[u/s]) \qquad (15)$$

$$y = \frac{1}{s}(v - s[v/s]) \qquad (16)$$

$$z = \frac{1}{l^2}(l[v/s] + [u/s]) \qquad (17)$$

ここで $s = 1/l$ である．

これを用いると 3 次元空間内の距離関数は

図 10 Buddha モデルとの計算結果．モデルのポリゴン数は 67,240
Fig. 10 Computation results with a Buddha model. The number of polygons is 67,240.

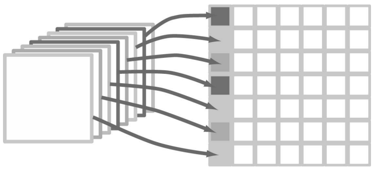

図 9 3 次元格子の 2 次元スライスへの分解
Fig. 9 Decomposition of three-dimensional grid into two-dimensional slices.

$$d(\mathbf{x}) = d(x, y, z) = d(t(u, v)) \quad (18)$$

と変形でき，2 次元空間内での距離関数として求めることができる（**図 9**）．

各離散境界から距離 a 以下の領域において距離関数を求める場合は 1 辺 $2a$ の正方形を描画する．そして生成されたピクセルにおいてピクセルシェーダ内部で離散境界との距離計算を行う．

$$d(\mathbf{x}) = \sqrt{x^2 + y^2 + z^2} \quad (19)$$

同時に勾配ベクトルを求め，距離関数を R チャネル，勾配ベクトルの 3 成分を GBA チャネルに格納することで 1 ピクセルでこれらを表すことができる．また式 (2) の計算は $d(\mathbf{x})$ を深度値として書き出すことで深度テストを用いて行うことができる．

6.3 カップリング

布と物体との衝突判定は質点の座標の距離関数を求めることで行われる．このときに 3 次元の線形補間を行う必要があるが，GPU では 2 次元平面上での補間はコストフリーで行うことができる．3 次元格子上での距離関数は複数枚の 2 次元平面上に保持されているので，Z 軸方向の補間を 1 度行うだけで質点での距離関数と勾配ベクトルを求めることができる．求めたこれらの値を用いて衝突計算をフラグメントシェーダ内部で行う．

6.4 計算の分割

離散境界を用いる距離関数計算は既存研究と比べると高速に計算することができた．計算時間に関しては後述するが，レンダリングを行うすべてのフレームにおいて距離関数を計算することは計算コストが高かった．そのため，距離関数計算を物体の離散境界の計算と離散境界を用いた距離関数の計算の 2 段階に分割し計算を行った．Harris らが雲のシミュレーションで同様な手法を用いた[6]．数フレームごとに距離関数が計算されるため，距離関数の計算コストが複数フレーム間で分散されることになる．

この手法を用いることでシミュレーションのフレームレートは大幅に改善された．距離関数の構築を間引きしても距離関数の計算は 2 つのフレーム間で線形補間するため，布の衝突計算で用いられる距離関数はすべてのフレームで計算したものと大きく異ならない．

7. 結　　果

本手法を Pentium4 3.6 GHz の CPU および 3.0 GB のメモリを搭載した PC 上で実装した．用いた GPU は GeForce7800GTX でありビデオメモリは 256 MB である．プログラムは C++ で OpenGL を用い，Shader プログラムには C for graphics を用いて作成した．

布は 4,906 個の質点で表現した．また計算結果では 128^3 の計算格子内で離散境界へ分解した．用いたスライス数は 128 枚である．**図 10** にポリゴン数 67,240 のモデルをリアルタイムで変形させ布とインタラクションさせた結果を示す．この計算は約 80 FPS で実行できている．また**図 11** にはモデルのポリゴンが計算途中で裏返る物体との計算結果を示す．本手法で用いている距離関数の計算手法はトポロジに制約条件が必要ないのでこのようなモデルも問題なく取り扱うことができている．**図 12** にはモデルの一部を切り取ったものを用いた計算結果を示す．ポリゴンが連続でない部分からの距離関数とその勾配ベクトルも求められているので，そのような部分で不自然さは起こっていない．

表 1 に 3 つのモデルを用いたときの計算時間を示す．Time(a) はポリゴンモデルを離散境界に分解する計算時間であり，Time(b) は離散境界を用いて距離関数を

表 1 3 個のモデルでの計算時間（ミリ秒）
Table 1 Calculation time with three models (in milliseconds).

Model	Number of polygons	Number of discrete boundaries	Time(a)	Time(b)	Time(c)	Time(d)
Teapot	6,320	8,753	14.1	17.1	39	7.9
Rabbit	33,518	8,109	28.1	15.6	46	8.6
Buddha	67,240	9,033	35.9	18.8	63	12.5

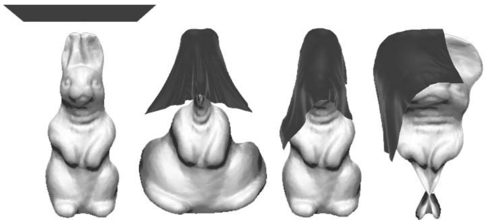

図 11 面の表裏が裏返るポリゴンモデルとの計算結果
Fig. 11 Computation results with a polygon model with an inside-out part.

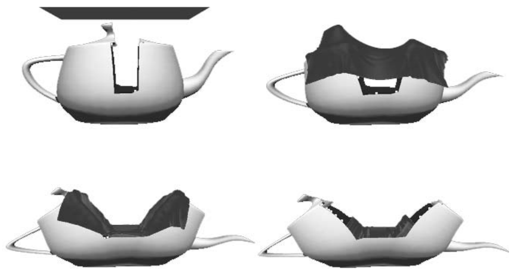

図 12 閉じていないポリゴンモデルとの計算結果
Fig. 12 Computation results with an opened polygon model.

求めるのにかかる計算時間である．そして Time(c) は計算の分割を行わない場合の計算時間の合計である．つまり離散境界の構築，距離関数の計算，布のシミュレーションを行うのにかかる時間である．Time(d) は計算の分割を行った場合の 1 回の描画にかかる計算時間の平均である．これらの結果を見るとモデルのポリゴン数が増加しても計算時間に大きな変化はないことが分かる．これは，距離関数の計算においてポリゴンモデルをまずラスタライズして離散境界に変換しているからであり，この離散境界を用いた距離関数の計算はポリゴン数には関係なくモデルの表面積に比例する計算となっているからである．

8. 結　　論

本論文では動的に変形する高解像度モデルと布との距離関数を用いたリアルタイム衝突計算手法を提案した．距離関数の計算では離散境界という概念を導入し，ポリゴンモデルをまず離散境界に分解することによって境界条件の次元を下げて計算を行った．これにより距離関数の計算コストはポリゴンモデルのポリゴン数ではなくモデルの表面積に依存するようになったため，多数のポリゴンを持つモデルからも高速に距離関数を計算することが可能となった．このようにして計算した距離関数と布のシミュレーションをカップリングすることによって動的に変形するモデルとの衝突を計算することができた．数値安定性の向上のためサブタイムステップアルゴリズムを導入した．またステップ状に変化する距離関数に起因する不自然さを軽減するために，距離関数の時間アンチエイリアシングを導入した．本手法を用いることで今までリアルタイムで衝突計算されたことのない多数のポリゴンを持つポリゴンモデルと布の衝突計算もリアルタイムで行うことが可能になった．

本研究では布との衝突を研究したが，この核となる高速な距離関数計算手法を他の物理計算に用いることができると考えられる．今後これを用いた流体と剛体の衝突計算なども研究していく．

参 考 文 献

1) Choi, K.J. and Ko, H.S.: Research problems in clothing simulation, *ComputerAided Design*, Vol.37, No.6, pp.585–592 (2005).
2) Cordier, F. and MagnenatThalmann, N.: Real-time animation of dressed virtual humans, *Computer Graphics Forum*, Vol.21, No.3, pp.327–336 (2002).
3) Dong, Z., Chen, W., Bao, H., Zhang, H. and Peng, Q.: Real time voxelization for complex polygonal models, *Proc. 12th Pacific Conference on Computer Graphics and Applications*, pp.43–50 (2004).
4) Fuhmann, A., Sobottka, G. and Gros, C.: Distance fields for rapid collision detection in physically based modeling, *Proc. GraphiCon*, pp.58–65 (2003).
5) Fuhrmann, A., Gros, C. and Luckas, V.: Interactive animation of cloth including self collision detection, *Proc. WSCG*, pp.141–148 (2003).
6) Harris, M.J., Baxter, W.V., Scheuer-Mann, T. and Lastra, A.: Simulation of cloud dynamics on graphics hardware, *Proc. ACM Siggraph/Eurographics conference on Graphics*

Hardware, pp.92–101 (2003).
7) Hoff, K.E., Keyser, J., Lin, M., Manocha, D. and Culver, T.: Fast computation of generalized voronoi diagrams using graphics hardware, *Proc. 26th annual conference on Computer graphics and interactive techniques*, pp.277–286 (1999).
8) Jones, M.W., Baerentzen, J.A. and Sramek, M.: 3D distance fields: A survey of techniques and applications, *IEEE Trans. Visualization and Computer Graphics* (2006).
9) Kimmerle, S. and Mezger, J.: Collision detection for cloth simulation, *Eurographics Tutorial*, pp.45–50 (2004).
10) Mauch, S.: A fast algorithm for computing the closest point and distance transform, Technical Report, California Institute of Technology (2000).
11) Meyer, M., Debunne, G., Desbrun, M. and Barr, A.H.: Interactive animation of cloth-like objects for virtual reality, *The Journal of Visualization and Computer Animation*, Vol.12, pp.1–12 (2001).
12) Mezger, J., Kimmerle, S. and Etzmus, O.: Hierarchical techniques in collision detection for cloth animation, *Journal of WSCG*, Vol.11, No.2, pp.322–329 (2003).
13) Provot, X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior, *Proc. Graphics Interface*, pp.147–154 (1995).
14) Sigg, C., Peikert, R. and Gross, M.: Signed distance transform using graphics hardware, *Proc. IEEE Visualization*, pp.83–90 (2003).
15) Sud, A., Govindaraju, N., Gayle, R. and Manocha, D.: Interactive 3D distance field computation using linear factorization, *Proc. ACM Symposium on Interactive 3D Graphics and Games* (2006).
16) Sud, A., Otaduy, M.A. and Manocha, D.: Difi: Fast 3D distance field computation using graphics hardware, *Proc. Eurographics*, Vol.23, pp.557–566 (2004).
17) Teschner, M., Kimmerle, S., Heidelbeger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.P., Faure, F., Magenenat-Thalmann, N., Strasser, W. and Volino, P.: Collision detection for deformable objects, *Computer Graphics Forum*, Vol.24, No.1, pp.61–81 (2005).
18) Vassilev, T. and Spanlang, B.: Fast cloth animation on walking avatars, *Computer Graphics Forum*, Vol.20, No.3, pp.260–267 (2001).

(平成 18 年 6 月 2 日受付)
(平成 19 年 1 月 9 日採録)

原田　隆宏（学生会員）

昭和 56 年生．平成 18 年東京大学大学院工学系研究科システム量子工学専攻修士課程修了．同年東京大学大学院情報学環学際情報学府助手．平成 19 年 4 月助教．計算力学とコンピュータグラフィックスの研究に従事．日本機械学会，ACM 各会員．

越塚　誠一

昭和 37 年生．昭和 61 年東京大学大学院工学系研究科原子力工学専攻修士課程修了．同年東京大学工学部助手．平成 3 年博士（工学）．同年講師．平成 5 年助教授．平成 16 年教授．連続体の力学シミュレーションの研究に従事．特に粒子法の開発を行う．平成 17 年に丸善より『粒子法』を出版．平成 18 年に日本学術振興会賞を受賞．日本原子力学会，日本機械学会，日本流体力学会，日本計算工学会，日本応用数理学会各会員．