筑波大学大学院博士課程

システム情報工学研究科修士論文

スクリーン空間での複雑な屈折を考慮した液体の実時間描画

今井拓也

修士（工学）

（コンピュータサイエンス専攻）

指導教員　三谷 純、金森 由博

2016年3月

概要

ゲームなどの対話的なアプリケーションにおいて、粒子の集合として表現された液体の動きをシミュレートし、その結果を描画するということが行われている。この液体の描画では高速化のために液体前面でのみ屈折が計算されてきたが、前面のみの屈折では液体の厚みを表現できず写実性を損なうという問題がある。

そこで本論文では、粒子の集合として表現された液体に対して、複雑な屈折を考慮した実時間描画手法を提案する。液体を表現した粒子の集合から液面を抽出する既存手法を拡張し、液体粒子を液体本体と飛沫に分け、それぞれの部分から液面を抽出する。抽出した液面に対し、2 回屈折を考慮して実時間描画を行う既存手法を拡張して適用し、液体本体における前背面の液面と、飛沫部分における前背面の液面の計 4 枚の液面を利用して最大 4 回までの屈折を考慮した液体の描画を実現する。さらに液面抽出において滑らかな液面を抽出するために行う平滑化について、本論文では新たに局所的な平面フィッティングによる平滑化を提案し、既存手法で発生していた問題の解決を図る。提案手法による結果を既存手法と比較し、提案手法の有効性および課題を明らかにした。

目 次

図 目 次

第1章　序論

1.1　研究背景と目的

コンピュータグラフィックス (CG) では現実の風景と勝るとも劣らない写実的な画像を生成することが目標の一つとされてきた。そのため、コンピュータの発展とともに写実性を高めるための理論が提案されてきた。さらに物理シミュレーションによって物理法則を再現することで、より現実に即した画像を生成できるようになった。近年ではハードウェアの高性能化に伴い、リアルタイムでより写実的な画像を生成することに需要があり、この目標を達成すべく様々な研究が行われてきた。自然現象の中でも液体は日常生活で特に目にするものであるため CG として表現する需要が非常に高く、盛んに研究されている。ゲームのような対話的なアプリケーションでは液体を有限個の粒子の集合として離散的に表現し、各粒子の挙動を計算してシミュレーションを行う。粒子の動きのみを計算すればよいため、炎や煙を生成するために使われているパーティクルシステムと相性がよく、パーティクルシステムに実装されて使われている。

しかし対話的なアプリケーションにおける液体の描画結果は写実的であるとはいえない状況にある。一般に液体を表現するには光の反射と屈折が重要であるが、写実性を高めるためには液体内部での光の反射と屈折を複数回計算する必要があり、時間がかかる。そのため対話的なアプリケーションでは液体内部での光の屈折や反射の計算を省略し、最初に光が液体に入射した際にのみ光の反射や屈折を計算をしていることが多い。しかし液体の厚みを表現できず写実性を損ねてしまう問題がある。

卒業研究 [1] ではこの問題を解決するため、液体を表現した粒子を入力として液体の前面と背面での 2 回屈折を考慮した、液体の実時間描画を実現する手法を提案した。しかしこの方法では粒子が密集している、まとまった液体本体となるべき部分と孤立した粒子である飛沫を区別せずに扱っている。そのため液面を抽出したときに両者が混ざってしまい、屈折が正しく計算できないという問題が発生する。また液面抽出において滑らかな液面を得るために行う平滑化について、平滑化後の輪郭付近の液面が視点に向かって反ってしまうことで誤った屈折ベクトルが計算されてしまうとともに、十分な滑らかさの液面を得るためには平滑化に時間がかかる問題がある。

本研究ではこれらの問題を解決するために、粒子が密集しているまとまった液体本体となるべき部分と孤立した粒子の集合である飛沫部分に分け、それぞれの前面および背面での屈折を考慮した液体の実時間描画手法を提案する。さらに粒子から液面を抽出する際に行っている平滑化の計算において、新たに平面フィッティングを利用した平滑化手法を用いることで輪郭付近の液面での屈折の精度を高めるとともに、平滑化にかかる時間を短縮する。提案手

法によって生成した結果と既存研究による結果を比較し、提案手法の有用性および問題点を明らかにする。

1.2 論文の構成

本論文は全 6 章からなる。第 2 章では関連研究について概説する。第 3 章では卒業研究 [1] で提案した描画手法について説明する。第 4 章では本研究が提案する手法の詳細な手順を解説する。第 5 章では提案手法に対する結果を提示し、結果に対しての考察を述べる。第 6 章では本研究の結論と今後の課題について記す。

第2章　関連研究

本章では本研究に関連のある、液体を表現した粒子の集合から液面を抽出するための研究と、ポリゴンメッシュに対し 2 回屈折を考慮した実時間描画を行う研究について解説する。

2.1　液面抽出に関する研究

液面抽出に関する研究は、液体粒子の表現方法により二つに分けられる。一つは液体粒子をメタボールとして表現し、メタボールから液面を抽出する方法であり、もう一つは液体粒子を球として描画し、その結果を基に液面を抽出する方法である。

まず、液体粒子をメタボールとして表現し、メタボールから液面を抽出する既存研究について述べる。Kanamori ら [2] は Bézier Clipping 法と Depth Peeling を用いた液面を抽出する方法を発表した（図 2.1）。しかし Kanamori らの手法は液体に対し 1 回しか屈折を考慮していない。Gourmel ら [3] はレイトレーシングによって液面を高速に抽出する方法を発表した（図 2.2）。Fraedrich らは SPH 法によってシミュレートした結果に対し、ボリュームレンダリングによって液面を抽出することで高品質な液面を生成した（図 2.3）。Akinci ら [5] は複数の CPU および GPU を用いたマーチングキューブ法を提案しより高速に液面を抽出した（図 2.4）。Yu と Turk[6] は局所的な粒子分布から計算された異方性カーネルを用いて液面を抽出することで滑らかな液面を実現した（図 2.5）。これらの手法はより写実性の高い結果を作成できる特徴があるが、処理に時間がかかってしまい、実時間描画が難しい。本研究では写実性の高い実時間描画結果を目指す。

次に液体粒子を球として描画し、その結果を基に液面を抽出する既存研究について述べる。Müller ら [7] は SPH 法によって実時間で液体をシミュレートし、スプラッティングを利用して液面を描画する手法およびマーチングキューブ法によって液面を抽出する方法を提案した（図 2.6）。Adams ら [8] は Müller らのスプラッティングを利用して液面を描画する手法を改良し、複数の異なる色の液体の混合の描画を可能にした（図 2.7）。Müller ら [9] はスクリーン空間において液体を表現した粒子の集合からポリゴンメッシュとして液面を実時間で抽出した（図 2.8）。これに対し Cords と Staadt[10] は各粒子を球として描画した後、得られたデプスマップに平滑化を施すことでスクリーン空間において実時間で液面を抽出した（図 2.9）。van der Laan ら [11] は Cords と Staadt の手法を拡張し、平滑化に Mean Curvature Flow と呼ばれる液面の平均曲率を最小化する操作を行うことで、より滑らかな液面を実時間で抽出した。さらに液体の厚みを疑似的に計算し、厚みに応じて液体の色の濃さを変化させることで液体の厚みの表現を目指した（図 2.10）。Bagar ら [12] は van der Laan らの手法に対し、粒子を液体

図 2.1: Kanamori らの手法による結果. 画像出典: 文献 [2].

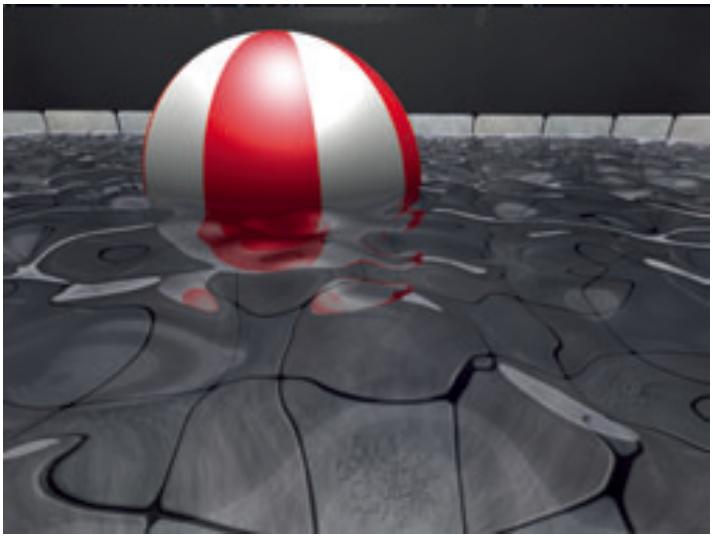


図 2.2: Gourmel らの手法による結果. 画像出典: 文献 [3].

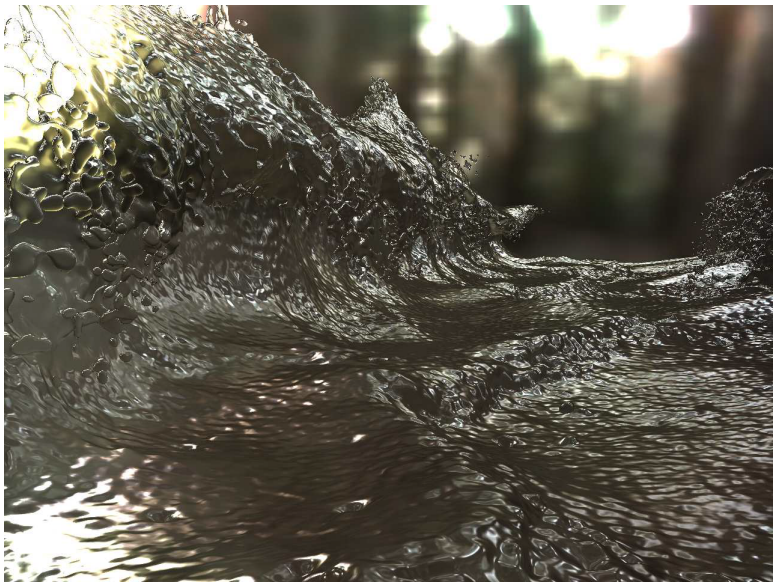


図 2.3: Fraedrich らの手法による結果. 画像出典: 文献 [4].

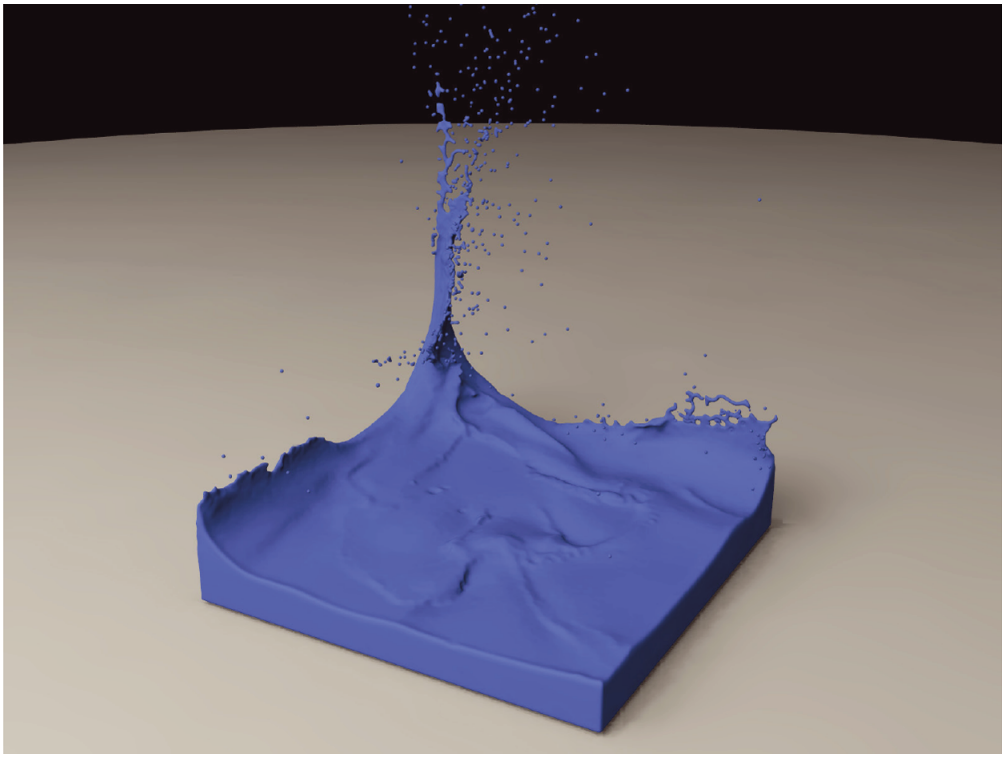
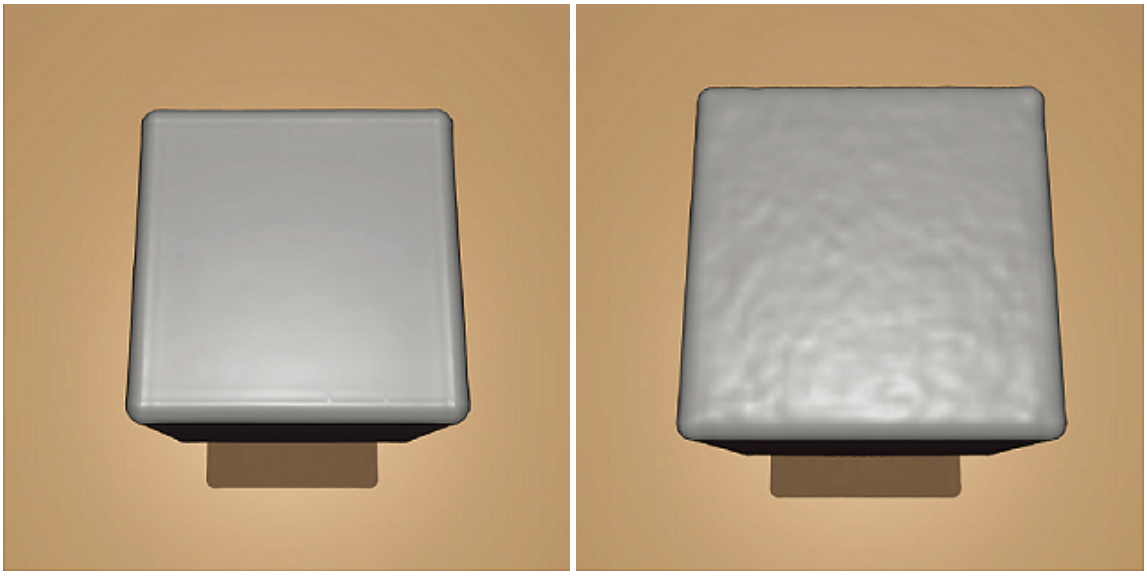
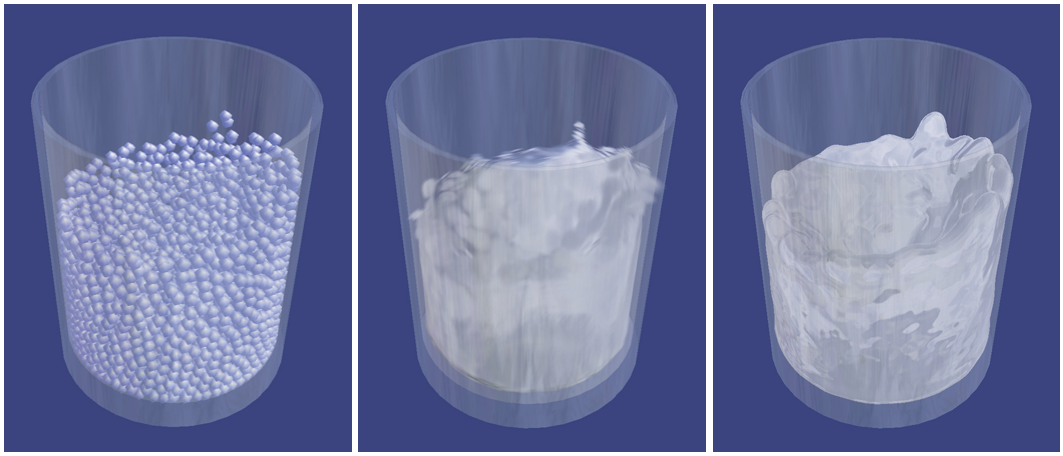


図 2.4: Akinci らの手法による結果. 画像出典: 文献 [5].



(a) Yu と Turk の手法により液面を抽出した結果　　(b) 等方性カーネルを用いて液面を抽出した結果

図 2.5: Yu と Turk の手法による結果. 局所的な粒子分布をもとに計算された異方性カーネルを用いて滑らかな液面を抽出することができる. 画像出典: 文献 [6].



(a) 粒子を表示した結果　　(b) スプラッティングを利用して液面を描画した結果　　(c) マーチングキューブ法によって液面を抽出し描画した結果

図 2.6: Muller らの手法による結果. 画像出典: 文献 [7].

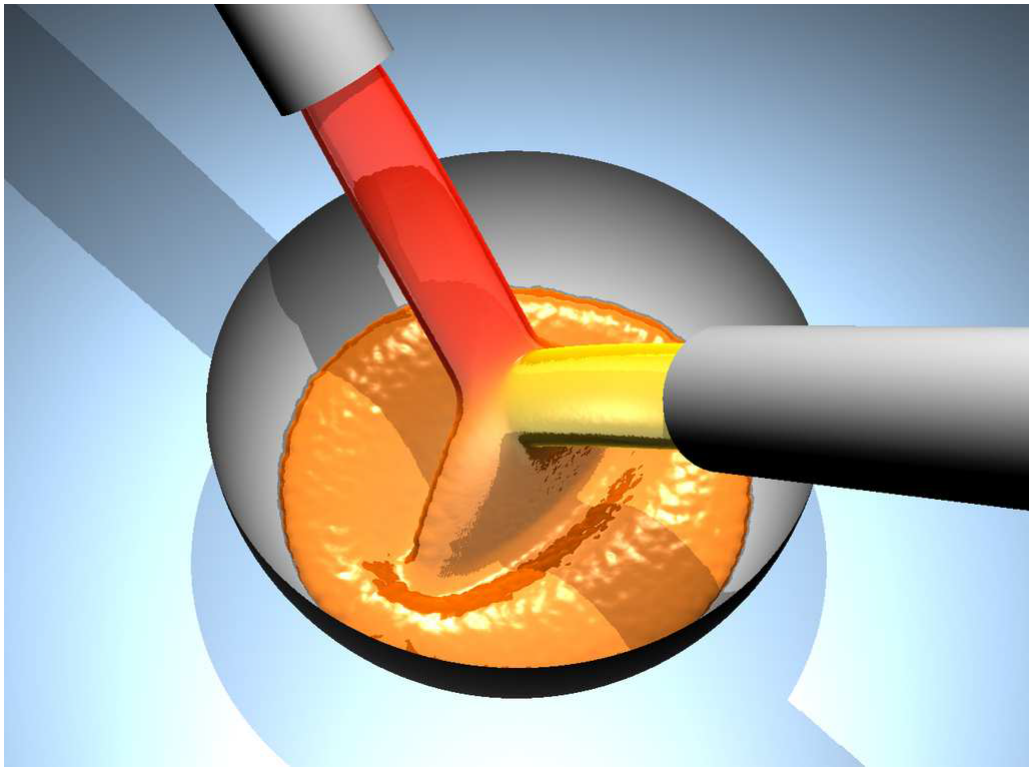


図 2.7: Adams らの手法による結果. 画像出典: 文献 [8].

(a) 最終的な描画結果　　(b) メッシュを表示した結果

図 2.8: Müller らの手法による結果. 画像出典: 文献 [9].

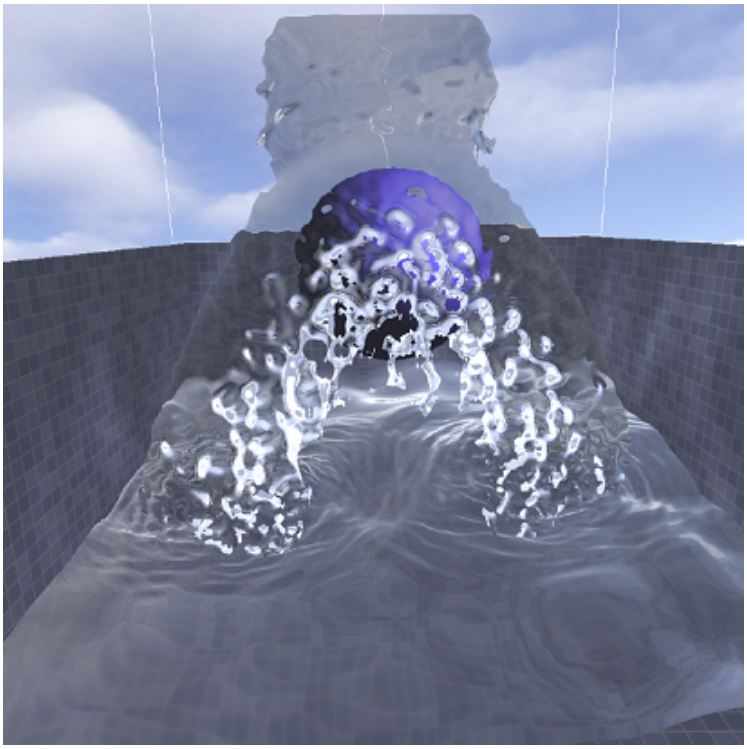


図 2.9: Cords と Staadt の手法による結果. 画像出典: 文献 [10].

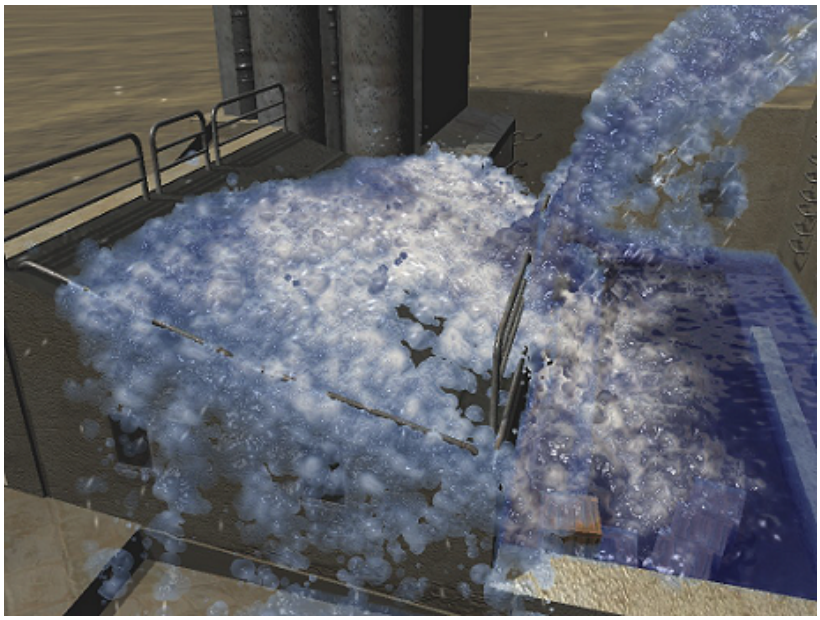


図 2.10: van der Laan らの手法による結果. 画像出典: 文献 [11].

図 2.11: Bagar らの手法による結果. 画像出典: 文献 [12].

粒子と気泡粒子に分け、それぞれの粒子の厚みによって液体の色を変化させるとともに、平滑化手法を改善することでさらに液体らしい表現を目指した（図 2.11）。これらの手法は液体を表現する粒子から液面を抽出する手法であるが、液体前面のみを対象とした手法であり、2 回以上の屈折を考慮していない。

2.2　2 回以上の屈折を考慮した実時間描画に関する研究

一般に、光の屈折を計算するにはレイトレーシング法が用いられる [13]。しかしレイトレーシング法で複数回の光の屈折を計算すると負荷が高くなってしまう。Wyman[14] はポリゴンメッシュモデルに対し、前面および背面での 2 回屈折を計算すれば十分に写実的な結果が得られることを示した（図 2.12）。Wyman の手法はスクリーン空間において、事前に計算した値を用い、対話的な速度で計算可能である。しかし Wyman の手法では事前にジオメトリの各頂点での法線方向の奥行を計算する必要があり、ジオメトリが変形するような物体には適用できない。この問題に対し Oliveira と Brauwers[15] は、視線レイとジオメトリの背面との交点をスクリーン空間において二分探索法を用いて反復的に計算して求め、変形するポリゴンメッシュモデルに対して 2 回屈折を考慮した効率的な実時間描画を実現した（図 2.13）。一方で Davis と Wyman[16] はポリゴンメッシュモデルに対し、全反射を考慮した実時間描画手法を実現した（図 2.14）。これらの研究は 2 回または複数回の屈折を考慮して実時間描画を実現しているが、ポリゴンメッシュモデルを対象としている。そのため粒子の集合として表現された液体に対して直接適用するためには別途液体のデプスを計算する必要がある。

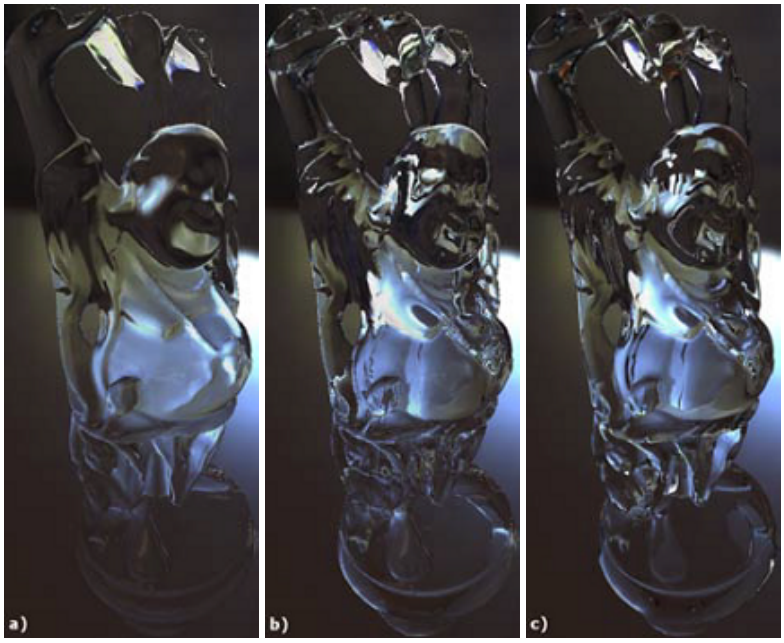
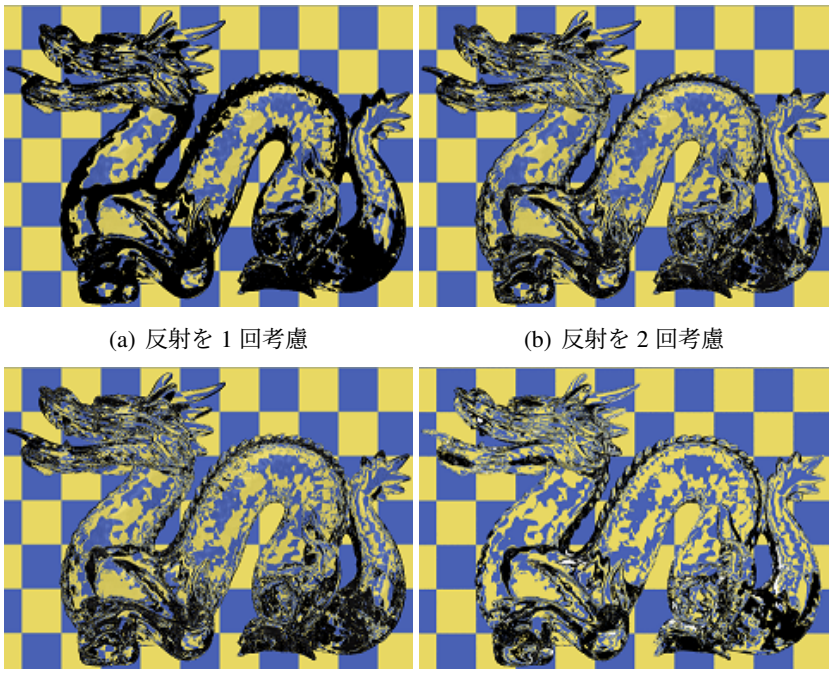


図 2.12: Wyman の手法による結果. (a) 前面のみ屈折を計算した結果. (b) Wyman の手法による 2 回屈折を考慮した結果. (c) レイトレーシングによる結果. 前面および背面での 2 回屈折を計算すればレイトレーシングによる結果に近い結果が得られる. 画像出典: 文献 [14].

(a) (b) (c)

図 2.13: Oliveira と Brauwers の手法による結果. (a) Oliveira と Brauwers の手法による 2 回屈折を考慮した結果. (b) 前面の屈折のみ考慮して描画した結果. (c) レイトレーシング法による結果. 2 回屈折を考慮した結果はレイトレーシング法による結果に似ているが, 全反射を計算していない点で結果が異なる. 画像出典: 文献 [15].



(a) 反射を 1 回考慮　(b) 反射を 2 回考慮

(c) 反射を 3 回考慮　(d) レイトレーシング法による結果

図 2.14: Davis と Wyman の手法による結果. (a), (b), (c) Davis と Wyman による結果. (d) レイトレーシング法による結果. 反射回数を増やし全反射を正確にシミュレートすればするほどレイトレーシング法による結果に近づく. 画像出典: 文献 [16].

第3章　スクリーン空間での2回屈折を考慮した液体の実時間描画手法

3.1　全体の流れ

卒業研究 [1] ではスクリーン空間での 2 回屈折を考慮した液体の実時間描画手法を提案した。スクリーン空間での 2 回屈折を考慮した液体の実時間描画手法を説明する。この手法では SPH 法など液体を有限個の粒子として表現し、そのシミュレーション結果を対象とする。入力として粒子それぞれの位置と半径を用いる。Cords と Staadt[10] の手法を利用し液面を抽出する。液体粒子を球として描画し液体前面および背面のデプスマップを作成し、その後デプスマップを平滑化を計算して法線を計算する。このとき液体の前背面が一度に計算できるように拡張し、デプスマップの平滑化にはバイノミアルフィルタではなくバイラテラルフィルタを用いる。続いて Oliveira と Brauwers[15] の手法を利用し、反復法によって視線レイと液体背面との交点を計算し、液体の前面および背面における屈折を計算する。高速化のため、Oliveira と Brauwers の手法で用いられた二分法ではなく割線法を用いる。その後、液体前面および背面における反射による色と液体背面における屈折による色を混ぜ合わせることで最終的な結果を生成する。これらの処理は GPU を用いて計算できる。

卒業研究の手法の流れを以下に示す (図 3.1)。

1. Cords と Staadt の手法を基に液体を表現した粒子の流合から液面を抽出する。
   - 1- 1 粒子を球として描画し、液体前面および背面のデプスマップを作成する。
   - 1- 2 Cords と Staadt の手法とは異なり、バイラテラルフィルタを用いて平滑化する。
   - 1- 3 平滑化されたデプスマップから法線マップを作成する。
2. Oliveira と Brauwers の手法を基に 2 回屈折を考慮して描画を行い、最終結果を作成する。
   - 2- 1 液体前面における視線レイの屈折ベクトルと反射ベクトルを計算する。
   - 2- 2 Oliveira と Brauwers の手法とは異なり、液体前面において屈折された視線レイと液体背面との交点を割線法を用いて計算する。
   - 2- 3 液体背面における視線レイの屈折ベクトルと反射ベクトルを計算する。
   - 2- 4 液体前面において反射された視線レイと、液体背面において反射された視線レイおよび屈折された視線レイを基にそれぞれ環境マップから色を取得し、両者をブレンドして最終結果を描画する。

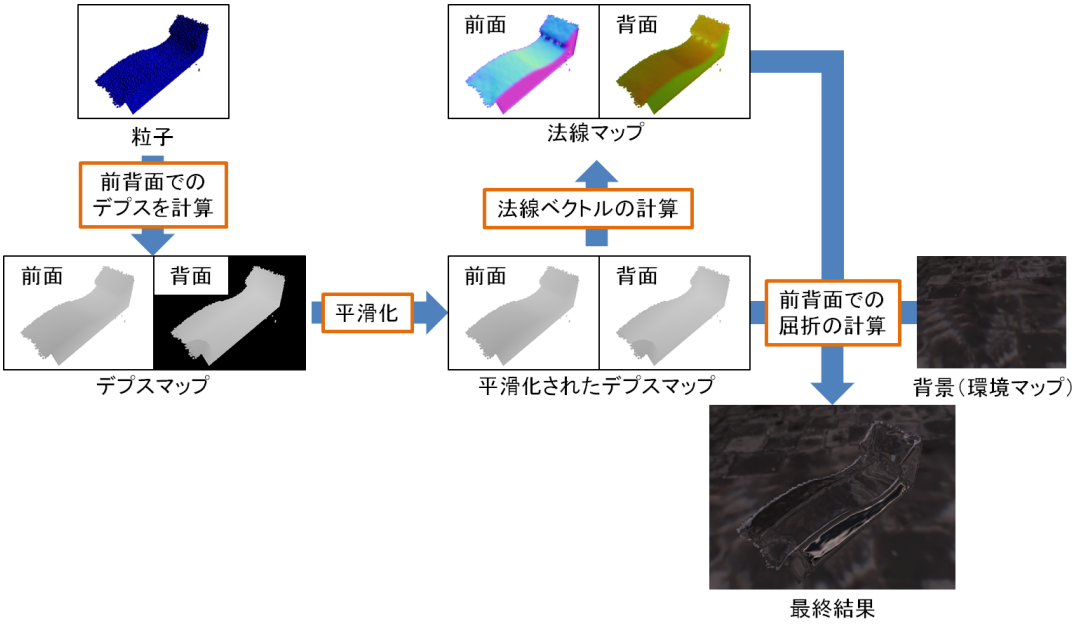


図 3.1: スクリーン空間での 2 回屈折を考慮した液体の実時間描画手法における計算の流れ.

3.2 液面の抽出

Cords と Staadt の手法を基に液面を抽出する。最初に液体を構成する粒子を球として描画することで奥行を計算し、デプスマップを作成する。卒業研究の手法では、最適化された球の描画手法である Kanamori ら [2] の手法を導入し、粒子を描画する。Kanamori らの手法では、バーテックスシェーダにおいて粒子を正方形の領域として処理を行い、フラグメントシェーダにおいては正方形の領域の各ピクセルにに対し粒子と視線レイが交差するか判定し、交差する領域のみ描画を行う。

次にデプスマップを平滑化する。これは粒子を球として描画すると本来の液体にはない凹凸が出てきてしまい液体らしく見えないため、滑らかな液面を得るために行う。Cords と Staadt は平滑化に用いるローパスフィルタとしてバイノミアルフィルタを用いていたが、卒業研究の手法ではバイラテラルフィルタを利用した。これはデプスマップ中に存在する、隣接するピクセルにおいてデプス値に大きな差のあるエッジ部分を保ちつつ平滑化を行うためである。入力画像 f の座標 (i,j) における値を $f(i,j)$、出力画像 g の座標 (i,j) における値を $g(i,j)$ とすると、バイラテラルフィルタによるフィルタリングは次式で表される。

$$g(i,j) = \frac{\sum_{n=-w}^{w}\sum_{m=-w}^{w} f(i+m,j+n)\exp\left(-\frac{m^2+n^2}{2\sigma_1^2}\right)\exp\left(-\frac{(f(i,j)-f(i+m,j+n))^2}{2\sigma_2^2}\right)}{\sum_{n=-w}^{w}\sum_{m=-w}^{w}\exp\left(-\frac{m^2+n^2}{2\sigma_1^2}\right)\exp\left(-\frac{(f(i,j)-f(i+m,j+n))^2}{2\sigma_2^2}\right)} \tag{3.1}$$

ここで w はカーネル半径、σ_1 と σ_2 はバイラテラルフィルタのパラメータである。バイラテラルフィルタを複数回適用することでデプスマップを十分平滑化する。卒業研究の手法では r を粒子の半径とし、$\sigma_1 = \dfrac{w}{2\sqrt{2}}, \sigma_2 = 2\sqrt{2}r$ として計算する。

デプスマップを平滑化した後、平滑化したデプスマップから法線マップを作成する。スクリーン座標系の点 (x, y) における液体前面での法線ベクトル $\mathbf{n}(x, y)$ の計算式を以下に示す。

$$\mathbf{n}(x, y) = \frac{\mathbf{n}'(x, y)}{\|\mathbf{n}'(x, y)\|} \tag{3.2}$$

$$\mathbf{n}'(x, y) = \frac{\partial}{\partial x}\mathbf{P}(x, y) \times \frac{\partial}{\partial y}\mathbf{P}(x, y) \tag{3.3}$$

$$\mathbf{P}(x, y) = \begin{pmatrix} \dfrac{x - \frac{W_x}{2}}{F} \\ \dfrac{y - \frac{W_y}{2}}{F} \\ 1 \end{pmatrix} D(x, y) \tag{3.4}$$

ここで $\mathbf{P}(x, y)$ はスクリーン空間上の点 (x, y) に対応する視点空間上の点であり、$D(x, y)$ はスクリーン座標系上の点 (x, y) における液体前面のデプスを表す。W_x, W_y はそれぞれスクリーン空間の幅と高さを表し、F は焦点距離を表す。液体背面における法線を計算するためには式 (3.3) の右辺に -1 をかければよい。

3.3 割線法による視線レイと液体背面との交点の計算

液体背面での屈折を計算するためには液体前面で屈折された視線レイと液体背面との交点を計算する必要がある。Oliveira と Brauwers の手法では液体前面で屈折された視線レイとポリゴンメッシュの背面との交点を計算するために二分探索法を用いた。二分探索法は下記のように反復的に計算される。

1. 背面デプスマップの最大値 Z_{max} と最小値 Z_{min} を用いて次式で表される視点空間上の 2 点 $\mathbf{S}, \mathbf{E} \in \mathbf{R}^3$ を計算し、$\mathbf{S}, \mathbf{E}$ をテクスチャ座標に変換した点 $\mathbf{t_S}, \mathbf{t_E} \in \mathbf{R}^2$ を得る。

$$\mathbf{S} = \mathbf{P}_1 + (Z_{min} - P_1.z)(\hat{\mathbf{T}}_1 / \hat{T}_1.z) \tag{3.5}$$

$$\mathbf{E} = \mathbf{P}_1 + (Z_{max} - P_1.z)(\hat{\mathbf{T}}_1 / \hat{T}_1.z) \tag{3.6}$$

ここで $\mathbf{P}_1$ はポリゴンメッシュ前面と視線レイとの交点、$P_1.z$ は点 $\mathbf{P}_1$ での z 座標の値である。$\hat{\mathbf{T}}_1$ はポリゴンメッシュ前面で屈折された視線レイの単位方向ベクトルであり、$\hat{T}_1.z$ はベクトル $\hat{\mathbf{T}}_1$ の z 成分である。

2. 2 点 $\mathbf{S}, \mathbf{E}$ の中点 $\mathbf{P}$ を計算し、点 $\mathbf{P}$ をテクスチャ座標に変換する。点 $\mathbf{P}$ のテクスチャ座標 $\mathbf{t_P} \in \mathbf{R}^2$ は次式で計算される。

$$\mathbf{t_P} = (1 - \alpha)\mathbf{t_S} + \alpha\mathbf{t_E} \tag{3.7}$$

$$\alpha = \left(\frac{1}{P.z} - \frac{1}{S.z} \right) \frac{S.zE.z}{S.z - E.z} \tag{3.8}$$

ここで $S.z$ は点 $\mathbf{S}$ の z 成分、$E.z$ は点 $\mathbf{E}$ の z 成分、$P.z$ は $\mathbf{P}$ の z 成分を表す。

3. 点 $\mathbf{t_P}$ における背面デプスマップの値と $P.z$ を比較する。$P.z$ の方が小さく、かつ一定の範囲内に収束していれば $\mathbf{P}$ をポリゴンメッシュ背面での交点とする。それ以外の場合、$P.z$ が小さければ $\mathbf{P}$ を新しく $\mathbf{S}$ に、そうでなければ $\mathbf{P}$ を新しく $\mathbf{E}$ にし、手順 2 へ戻る。

Oliveira と Brauwers の二分探索法では初期点に背面デプスマップの最大値および最小値から計算された点を用いている。このとき背面デプスマップの最小値と最大値を計算するための処理を別途行う必要がある。そこで卒業研究の手法においては、背面デプスマップの最小値と最大値を利用せずに液体背面と視線レイとの交点を計算するために割線法を用いる。割線法とは、一次式で関数を近似しその根を求める処理を繰り返すことで近似解を求める方法である。液体前面と視線レイとの交点を $\mathbf{P}_f$、液体前面で屈折された視線レイの単位ベクトルを $\mathbf{T}_f$ とすると、パラメータ $t\ (\geq 0)$ を用いて液体前面で屈折された視線レイ上の点 $\mathbf{P}(t)$ を次式で表現できる。

$$\mathbf{P}(t) = \mathbf{P}_f + t\,\mathbf{T}_f \tag{3.9}$$

点 $\mathbf{P}(t)$ をスクリーンに投影したときの点を $\mathbf{P}'(t) \in \mathbf{R}^3$ とする。視線座標系での背面のデプス値を記録したデプスマップを D_{back}^{view} とし、$\mathbf{P}'(t) = (P'(t).x, P'(t).y, P'(t).z)$ に対し $(P'(t).x, P'(t).y)$ におけるデプス値を $D_{back}^{view}(P'(t).x, P'(t).y)$ と記す。点 $\mathbf{P}$ の z 成分 $P(t).z$ と $D_{back}^{view}(P'(t).x, P'(t).y)$ との差（以下、$\mathbf{P}(t)$ と背面デプスマップとの差と表現する）$\Delta z(t)$ を次のように定義する。

$$\Delta z(t) = P(t).z - D_{back}^{view}(P'(t).x, P'(t).y) \tag{3.10}$$

液体前面で屈折された視線レイ上の 2 点 $\mathbf{P}(t_s), \mathbf{P}(t_e)\ (t_s < t_e)$ と背面デプスマップとの差 $\Delta z(t_s), \Delta z(t_e)$ に対し、2 点 $(t_s, \Delta z(t_s))$, $(t_e, \Delta z(t_e))$ を通るような変数 t の一次関数 y を考える。

$$y = \frac{\Delta z(t_e) - \Delta z(t_s)}{t_e - t_s}(t - t_s) + \Delta z(t_s) \tag{3.11}$$

$y = 0$ となるような t の値 t^* を求めると、

$$t^* = t_s - \frac{t_e - t_s}{\Delta z(t_e) - \Delta z(t_s)} \Delta z(t_s) \tag{3.12}$$

になる。求めた t^* における液体の前面で屈折された視線レイ上の点 $\mathbf{P}(t^*))$ と背面デプスマップとの差 $\Delta z(t^*)$ を求め、$\Delta z(t^*)$ が一定の範囲に収まっていればそのときの $\mathbf{P}(t^*)$ を液体前面で屈折された視線レイと液体背面での交点とする。それ以外の場合、$\mathbf{P}(t_s)$ か $\mathbf{P}(t_e)$ のどちらかを $\mathbf{P}(t^*)$ で置き換え、更に $t_s < t_e$ となるように $\mathbf{P}(t_s)$ と $\mathbf{P}(t_e)$ を入れ替えて次の t^* を計算していく。しかしスクリーン座標系の点 $\mathbf{P}'$ に粒子の領域が存在しない場合、背面デプスマップに粒子のデプス値が書き込まれず、設定していた初期値が書き込まれ無効な値になる。無効な値が格納された点 $\mathbf{P}'$ を参照した場合、デプスが有効な値になるまで t の値を半分する操作を繰り返し行う。

3.4 出力する色の計算

出力する色 $\mathbf{C}$ は次式を用いて計算される。

$$\mathbf{C} = F(\mathbf{v}_f, \mathbf{n}_f)\mathbf{C}_{fr} + (1 - F(\mathbf{v}_f, \mathbf{n}_f))(F(\mathbf{v}_b, \mathbf{n}_b)\mathbf{C}_{br} + (1 - F(\mathbf{v}_b, \mathbf{n}_b))\mathbf{C}_{bt}) \tag{3.13}$$

ここで $\mathbf{v}_f$ は視線ベクトル、$\mathbf{v}_b$ は液体前面で屈折された視線ベクトル、$\mathbf{n}_f$ は液体前面での法線ベクトル、$\mathbf{n}_b$ は液体背面での法線ベクトル、$\mathbf{C}_{fr}$ は液体前面での反射ベクトルを基に環境マップから取得した色、$\mathbf{C}_{br}$ は液体前面での反射ベクトルを基に環境マップから取得した色、$\mathbf{C}_{bt}$ は液体背面での屈折ベクトルを基に環境マップから取得した色である。また F はフレネル係数を表している。ここではフレネル係数として次式で表される Schlick[17] の近似式を用いる。

$$F(\mathbf{v}, \mathbf{n}) = r_0 + (1 - r_0)(1 - c)^5 \tag{3.14}$$

$$c = -\frac{\mathbf{v} \cdot \mathbf{n}}{\|\mathbf{v} \cdot \mathbf{n}\|} \tag{3.15}$$

$$r_0 = \left(\frac{n-1}{n+1}\right)^2 \tag{3.16}$$

ここで n は相対屈折率を表す。

第4章　提案手法

4.1　全体の流れ

本研究で提案する手法の全体の流れについて解説する。提案手法では粒子の位置と半径を入力とする。卒業研究の手法と同様に、Cords と Staadt[10] の手法を基に粒子の集合として表現された液体から液面を抽出し、Oliveira と Brauwers[15] の手法を基に描画を行う。

今回提案する手法では以下の改善を行う。卒業研究の手法ではすべての液体粒子の奥行を同じデプスマップに書き込むため、飛沫が飛んでいるようなシーンでは液体本体と飛沫が一緒になってしまい、屈折が正しく計算できない問題がある。この問題を解決するために液体粒子を液体本体となるべき部分の粒子と孤立している粒子に分け、それぞれの部分ごとに前面と背面で 2 枚ずつ、合計 4 枚のデプスマップを利用して最大 4 回の屈折計算を行うことで問題の軽減を図る。液体粒子が液体本体となる粒子であるか飛沫となる粒子であるかは、その粒子の近傍粒子の密度で判断する。近傍粒子の密度が密である粒子の集合を液体本体、近傍粒子の密度が疎である粒子を飛沫と定義し、飛沫となる粒子の集合を描画することによって得られるデプスマップや液面を便宜上、飛沫部分のデプスマップや飛沫部分の液面、もしくは飛沫部分におけるデプスマップや飛沫部分における液面などと呼称する。生成した 4 枚の液面から屈折を計算する方法は卒業研究の手法を拡張して実現する。また、粒子を描画して得られたデプスマップを平滑化する際、スクリーン空間での 2 回屈折を考慮した液体の実時間描画手法ではバイラテラルフィルタを用いていたが、平滑化を繰り返し適用する必要があるため平滑化に時間がかかり、また平滑化によって液面の輪郭付近の液面が反ってしまうという問題がある。提案手法ではこれらの問題を解決するために局所的な平面フィッティングによる平滑化を提案する。提案手法ではさらに、Yu と Turk[6] の手法を用いて異方性カーネルを作成し、Gumhold[18] の手法やより汎用的な Sigg ら [19] の手法を用いて液体粒子を楕円体に変形して描画することでより滑らかな液面が得られるように改善を行う。

提案手法における計算の流れを下記に示す (図 4.1)。

1. 液体を表現した粒子について、液体本体の粒子か飛沫となる粒子かを決定する。同時に、Yu と Turk の手法を用いて楕円体として描画するための異方性カーネルを計算する。

2. 液体粒子から液面を抽出する。

   2- 1 粒子を楕円体として描画し、液体本体と飛沫部分における前面および背面のデプスマップを得る。

   2- 2 液体本体のデプスマップを局所的な平面フィッティングを利用して平滑化する。

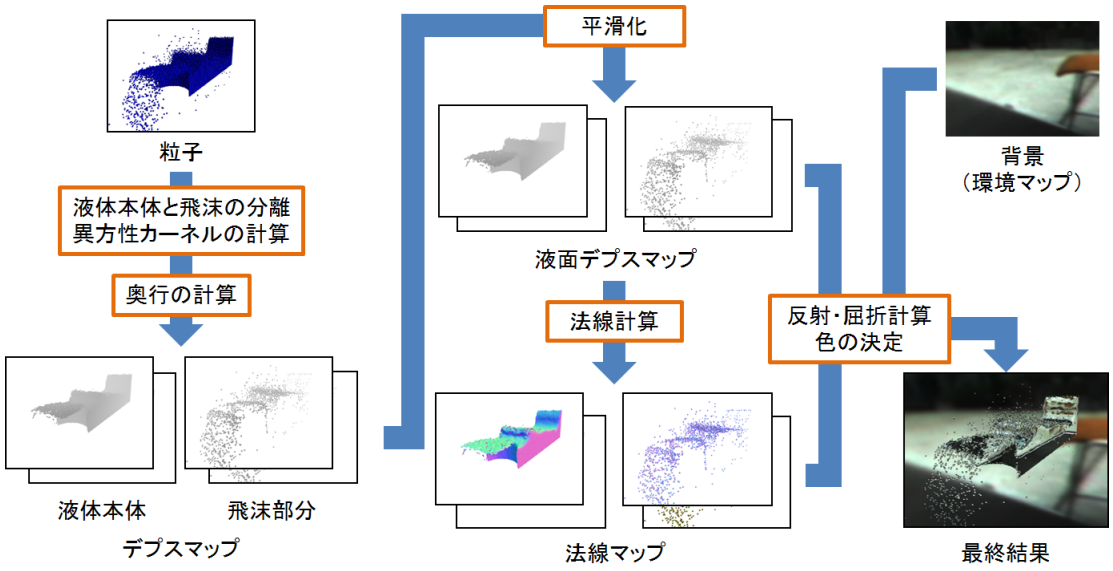


図 4.1: 提案手法における計算の流れ. 「液体本体と飛沫の分離」および「異方性カーネルの計算」以外の処理は GPU で実装した.

2- 3 4 枚のデプスマップからそれぞれの法線を計算し法線マップを作成する。

3. 4 枚の液面に対する光の反射および屈折を考慮して描画を行い、最終結果を生成する。

3- 1 液体本体の前面デプスマップのデプス値と、飛沫部分の前面デプスマップのデプス値を比較し、4 枚の液面に対する屈折計算の順序を決定する。

3- 2 選択された液面における視線レイの屈折ベクトルおよび反射ベクトルを計算する。

3- 3 次に屈折が発生する液面において、屈折された視線レイと液面との交点を割線法を用いて計算する。

3- 4 3- 2 に戻り、次の屈折が発生する液面とその交点を計算する。最後の液面における屈折ベクトルを計算し終えた場合、屈折ベクトルと今まで計算した反射ベクトルを基に環境マップから色を取得し、すべてをブレンドして最終結果を描画する。

これらの処理のうち、2 以降はすべて GPU で実装した。

4.2 液体粒子の区別

ある液体粒子が液体本体に属するか飛沫部分に属するか決めるには、その粒子の近傍の粒子の数が設定した閾値を超えているかで判断する。閾値を超えている場合その液体粒子は液体本体に属しているとみなし、そうではない場合その液体粒子は飛沫であるとみなす。この

ような液体粒子の判別方法は、液体粒子のシミュレーションを実装するパーティクルシステムに簡単に組み込むことができる。

4.3 局所的な平面フィッティングを利用した平滑化

スクリーン空間での 2 回屈折を考慮した液体の実時間描画手法では液面の平滑化にバイラテラルフィルタを用いた。しかし図 4.2 に示すように、バイラテラルフィルタによる平滑化を適用した輪郭付近の液面が反ってしまう。これは輪郭部分の液面にバイラテラルフィルタを適用した場合、有効な値がある部分の重みが大きくなる一方で無効な値の部分が重みが 0 になり、デプス値がより有効な値がある部分に偏るためである。

バイラテラルフィルタによる平滑化で発生する問題を解決するため、提案手法では平面フィッティングを利用した平滑化手法を提案する。平滑化対象のデプスマップにおいて、注目する画素とその近傍の画素のデプス値を基に視点座標系上の点に変換し、これらの点にフィッティングする平面を求める。注目する画素のデプス値を視点座標系に変換した点を通る視線レイと、求めた平面との交点を計算し、この値を利用することで平滑化を行う。注目する画素の座標とそのデプス値から変換した視点座標系の点 $\mathbf{P}_e = (x_e, y_e, z_e)$ と、注目する画素の近傍の座標とそのデプス値から変換した視点座標系の N 個の点 $\mathbf{P}_i = (x_i, y_i, z_i)\ (i = 1, 2, ..., N)$ は局所的に平面で近似できると仮定する。その平面を A と呼ぶことにし、平面 A の式を $ax+by+c+d = 0$ (ただし、 $a^2 + b^2 + c^2 = 1$) とおく。以下、 A の式における未知変数 a, b, c, d を求める。平面 A と点 $\mathbf{P}_i$ の誤差 e_i を次式で定義する。

$$e_i = ax_i + by_i + cz_i + d \tag{4.1}$$

さらに誤差の重み付き二乗の総和 E を次式で計算する。

$$E = \sum_i^N w_i e_i^2 \tag{4.2}$$

$$w_i = \frac{w(z_e, z_i)}{\sum_i^N w(z_e, z_i)} \tag{4.3}$$

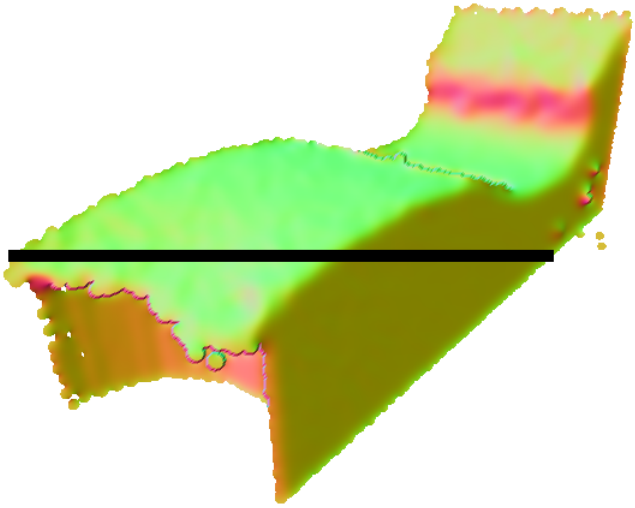
$$w(z_e, z_i) = \exp\left(-\frac{(z_e - z_i)^2}{\sigma^2}\right) \tag{4.4}$$

ここで σ は重み付けのパラメータである。さらに $G = a^2 + b^2 + c^2 - 1$ とおく。$G = 0$ のもとで E を最小化するためにラグランジュの未定乗数法を用いる。ラグランジュ乗数 λ を導入して、

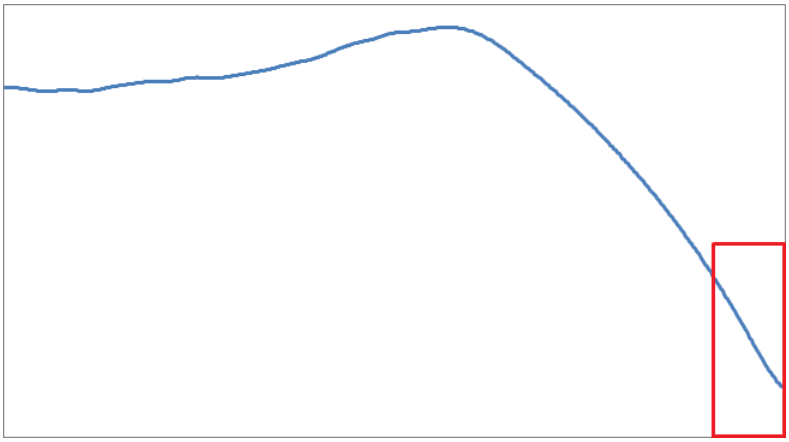
$$F = E - \lambda G \tag{4.5}$$

とおき、F を最小化する。F を最小化する a, b, c, d を a^*, b^*, c^*, d^* とおくと、点 $\mathbf{P}_e$ への単位視線ベクトル $\mathbf{V} = (x_v, y_v, z_v)$ と、平面 A との交点の z 座標は次式で与えられる。

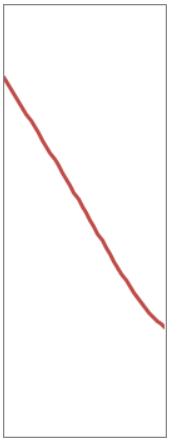
$$z = -\frac{d^*}{a^* x_v + b^* y_v + c^* z_v} z_v \tag{4.6}$$



(a) バイラテラルフィルタによって平滑化を適用した液面のデプスマップ. デプスマップの色はその座標における法線の向きを基に決定している.



(b) 図 4.2(a) の黒線部のデプスマップの変化をグラフで表現.



(c) 図 4.2(b) の赤の矩形領域を拡大.

図 4.2: バイラテラルフィルタによって平滑化を適用した液面のデプスマップの問題点. 図 4.2(c) を見るとデプスマップの輪郭部分が反っていることが確認できる.

交点の z 座標をデプス値に変換することで平滑化したデプス値を得られる。

場合によってはフィッティングの結果得られた平面が視線レイと平行になり、フィッティングの結果得られた平面と視線レイとの交点が、注目している画素の座標とそのデプス値から変換した視点座標系の点から著しく離れてしまうことがある。この場合得られたデプス値を利用してしまうと意図した通りの屈折が計算できない。そこでデプス値を計算した際、得られたデプス値と元のデプス値と差を調べ、差が一定の範囲になければ元のデプス値を利用することで影響を抑える。

4.4 4 枚の液面を用いた屈折の計算

卒業研究の手法では前面と背面の 2 枚の液面しか抽出しなかったため、前面の液面で屈折した視線レイは液体背面で屈折することが明らかであった。一方、提案手法では 4 枚の液面を用いて屈折計算を行うため、屈折された視線レイが次にどの液面のどの位置で交差するのかを計算する必要がある。

4 枚の液面と屈折の順番について考える。液面は液体本体の前面および背面、飛沫部分の前面および背面の 4 枚ある。これらは前面と背面で組になっており、前背面のデプス値は一度の描画で同時に計算しているため、前面のデプス値に値が書き込まれていれば同じ位置の背面のデプスマップにも値が書き込まれており、前面のデプス値よりも背面のデプス値の方が視点から遠ざかっていることが保証されている。ある部分の前面と背面の間の部分をその部分の内部と表現する。本来であれば液体の内部には別の液面が存在しないはずである。したがって視線レイは、液面前面で液体に入射し、液面背面で液体から出射することを繰り返す。ゆえに、視線レイは液体本体 → 液体本体、液体本体 → 飛沫、飛沫 → 液体本体、飛沫 → 飛沫のいずれかの順に通過する。提案手法では、計算時間を減らすために液体本体の液面 → 飛沫の液面、もしくは飛沫の液面 → 液体本体の液面の 2 通りに絞って屈折を計算する。屈折する液面の並び順を決定するために、液体本体前面のデプス値と飛沫部分前面のデプス値を比較し、より視点に近いデプス値をもつ液面を決定する。決定した液面が先頭に来るような並び順を選択し、その先頭の液面から順に屈折計算を行う。交点計算においては、卒業研究の手法で利用した割線法を用いて計算を行う。

4.5 出力する色の決定

最終的に出力される色 $\mathbf{C}$ は次式で計算される。

$$\mathbf{C} = F(\mathbf{v}, \mathbf{n}_1)\mathbf{b}(\mathbf{r}_1) + (1 - F(\mathbf{v}, \mathbf{n}_1))\mathbf{c}(\mathbf{t}_1) \tag{4.7}$$

$$\mathbf{c}(\mathbf{t}_k) = F(\mathbf{t}_k, \mathbf{n}_{k+1})\mathbf{b}(\mathbf{r}_{k+1}) + (1 - F(\mathbf{t}_k, \mathbf{n}_{k+1}))\mathbf{c}(\mathbf{t}_{k+1}) \quad (k = 1, 2, 3) \tag{4.8}$$

$$\mathbf{c}(\mathbf{t}_4) = \mathbf{b}(\mathbf{t}_4) \tag{4.9}$$

ここで $\mathbf{v}$ は視線ベクトル、$\mathbf{t}_k$ は k 回目の屈折ベクトル、$\mathbf{r}_k$ は k 枚目の液面で計算した反射ベクトル、$\mathbf{n}_k$ は k 枚目の液面の法線ベクトル、$\mathbf{b}(\mathbf{x})$ はベクトル $\mathbf{x}$ を基に背景から取得した色

である。また F はフレネル係数を表しており、提案手法においても卒業研究の手法と同様に Schlick の近似式を用いた。3 回目の交点が見つからなかった場合には $\mathbf{c}(\mathbf{t}_2) = \mathbf{b}(\mathbf{t}_2)$ として計算する。

液体内部での光の軌跡の長さを計算できるため、提案手法ではさらに液体による吸光を再現する Beer-Lambert の法則を考慮して描画を行った。液体背面の色を $\mathbf{C}_b$ とおくと、Beer-Lambert の法則を考慮した色 $\mathbf{C}'_b$ は次式で計算される。

$$\mathbf{C}'_b = \begin{pmatrix} \mathbf{C}_b.r\ \exp(-al\mathbf{C}_a.r) \\ \mathbf{C}_b.g\ \exp(-al\mathbf{C}_a.g) \\ \mathbf{C}_b.b\ \exp(-al\mathbf{C}_a.b) \end{pmatrix} \tag{4.10}$$

ここで a は吸光係数、l は液体内部での光の長さ、$\mathbf{C}_a.r$, $\mathbf{C}_a.g$, $\mathbf{C}_a.b$ はそれぞれ赤、緑、青おける減衰の強さである。提案手法では $a = 1.0$ を利用する。

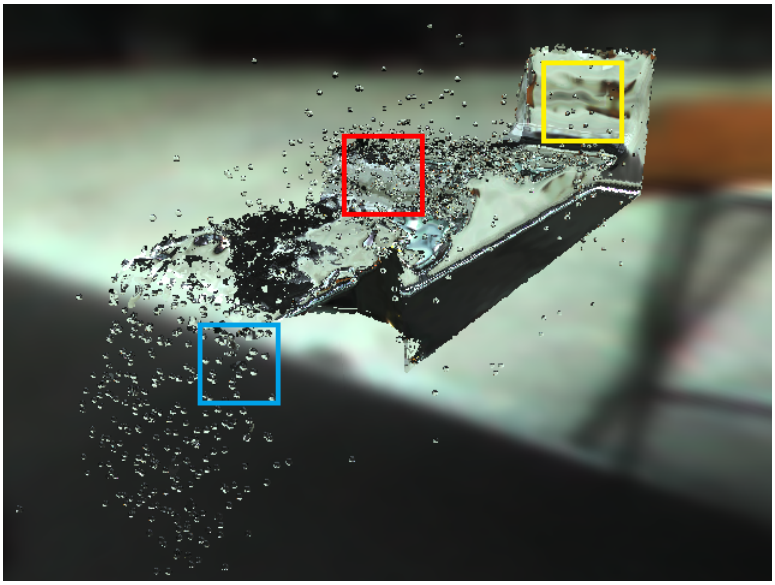
第5章　結果

本研究において行った、描画手法ごとの結果の比較実験および平滑化手法ごとの結果の比較実験について述べる。提案手法はC++を用いて実装した。ライブラリとしてOpenGL, GLUI, GLUT, GLEW, Eigenを使用した。シェーダにはGLSLを用いた。実行環境としてIntel Core i7-4770 3.40GHzのCPU、8GBのメモリNVIDIA GeForce GTX TITANのGPUを搭載したコンピュータを使用した。約10万個の粒子を事前にシミュレートしたデータを用いた「波」シーンおよび5万個の粒子をシミュレートした「段差」シーンを用いて実行結果を計算し、640×480と1024×768の解像度で出力した。結果にはシミュレーションの計算時間、液体粒子を液体部分か飛沫部分のどちらに属するのか判定する時間、および近傍の粒子から楕円体として描画するための異方性カーネルを作成する時間は含まれていない。

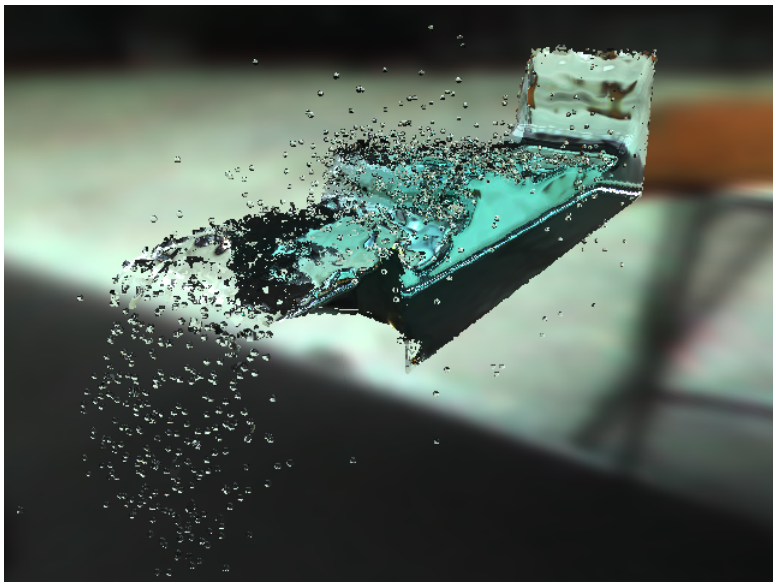
描画手法ごとの結果の比較実験について述べる。この実験では提案手法による描画結果、卒業研究の手法による描画結果、および提案手法で作成した液面をポリゴンメッシュ化しレイトレーシングによって描画した結果を比較する。局所的な平面フィッティングによる平滑化は9×9の大きさのカーネルを用いて3回平滑化を適用した。バイラテラルフィルタは7×7の大きさのカーネルを用いて22回平滑化を適用した。これらはパラメータを変化させて描画を行った際に得られた最適なパラメータである。レイトレーシングでは最大4回まで再帰的に液面での反射や屈折を計算した。提案手法を用いた場合の「波」シーンの描画結果を図5.1に、「段差」シーンの描画結果を図5.2に、提案手法を用いた場合の描画にかかる時間の内訳を表5.1に示す。また卒業研究の手法を用いた場合の「波」シーンの描画結果を図5.3に、「段差」シーンの描画結果を図5.4に、卒業研究の手法を用いた場合の描画にかかる時間の内訳を表5.2に示す。提案手法で作成した液面をポリゴンメッシュ化し、それをレイトレーシングによって描画した「波」シーンにおける結果を図5.5に、「段差」シーンにおける結果を図5.6に示す。レイトレーシングで描画した結果にノイズが含まれているのは、作成した液体本体前面と液体本体背面のポリゴンメッシュがそれぞれ独立しているためその間にわずかな隙間が存在してしまい、屈折もしくは反射した視線レイがその隙間を通ってしまうため、周囲の液面と異なる色になってしまうものと考えられる。表5.1と表5.2を比較して、全体としては卒業研究の手法よりも高速に描画できることが分かる。これはスクリーン空間での2回屈折を考慮した液体の実時間描画手法において、デプスマップの平滑化を複数回適用しているため計算に時間がかかっていたが、提案手法で採用した局所的な平面フィッティングによる平滑化によって平滑化一回当たりの計算時間は増えているものの、平滑化全体にかかる時間が短縮されたためである。一方、デプスマップの平滑化以外の項目では描画時間が増加している。提案手法でデプスマップの作成に時間がかかるようになった理由としては、粒子を異方

性カーネルによって楕円体に変形し描画しているためであると考えられる。法線マップの作成については作成する法線マップの数が卒業研究の手法に比べて倍になっているためである。屈折の計算については提案手法では卒業研究の手法よりも複雑な計算を行っているためである。図 5.1(a)、図 5.3 および図 5.5 における青、黄、赤の矩形領域の拡大図をそれぞれ図 5.7、5.8、5.9 に示す。図 5.7 のような飛沫のみの描画結果や、図 5.8 のような液体本体と飛沫の描画結果においては飛沫の描画が向上しているのが分かる。卒業研究の手法では、飛沫となるべき粒子も他の粒子と同じように描画し平滑化するため平坦な板のように描画されていたが、提案手法では飛沫は描画したデプスマップを平滑化していないため立体的に描画することができた。一方で図 5.9 のような大量の飛沫が存在する部分の描画結果においては、提案手法を利用しても意図した通りの結果が得られずノイズの多い結果になった。提案手法で作成した液面をポリゴンメッシュ化し、それをレイトレーシングで描画した場合においても同様の結果になったため、提案手法を利用しても大量の粒子が存在するシーンの描画には向かないことが分かった。

続いて平滑化の手法ごとの比較実験について述べる。この実験では平滑化に局所的な平面フィッティングを利用した平滑化を用いた場合（提案手法）および平滑化としてバイラテラルフィルタを用いた場合（卒業研究の手法）を比較する。描画手法ごとの比較実験と同様に、局所的な平面フィッティングを利用した平滑化は 9×9 の大きさのカーネルを用いて 5 回平滑化を適用し、バイラテラルフィルタは 9×9 の大きさのカーネルを用いて 20 回平滑化を適用した。局所的な平面フィッティングによる平滑化を用いた場合の「波」シーンの描画結果を図 5.10 に、バイラテラルフィルタを用いた場合の「波」シーンの描画結果を図 5.11 に示す。また、図 5.10 および図 5.11 における赤の矩形領域の拡大図を図 5.12 に示す。局所的な平面フィッティングを利用した平滑化においてパラメータを変えて描画した結果を図 5.13 に、その際平滑化にかかった計算時間を表 5.3 に示す。バイラテラルフィルタの平滑化のパラメータを変えて描画した際の結果を図 5.14 に、その際平滑化にかかった計算時間を表 5.4 に示す。図 5.11 より、バイラテラルフィルタを用いた場合輪郭付近において意図した通りに屈折が計算されず、輪郭付近の液面の色が周囲の液面の色と異なってしまう問題があった。図 5.10 より、局所的な平面フィッティングを利用した平滑化を導入することで、バイラテラルフィルタを用いた場合に発生していた問題を解決することができる。

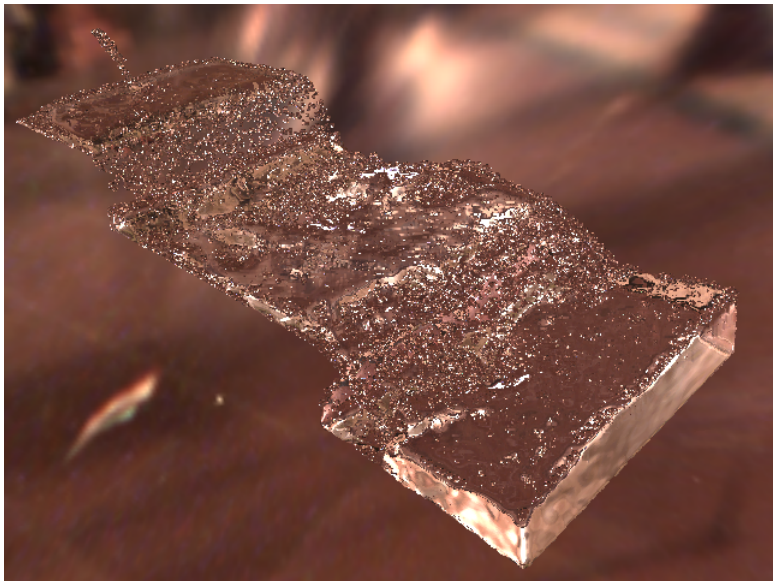


(a) Beer-Lambert の法則を考慮しなかった場合

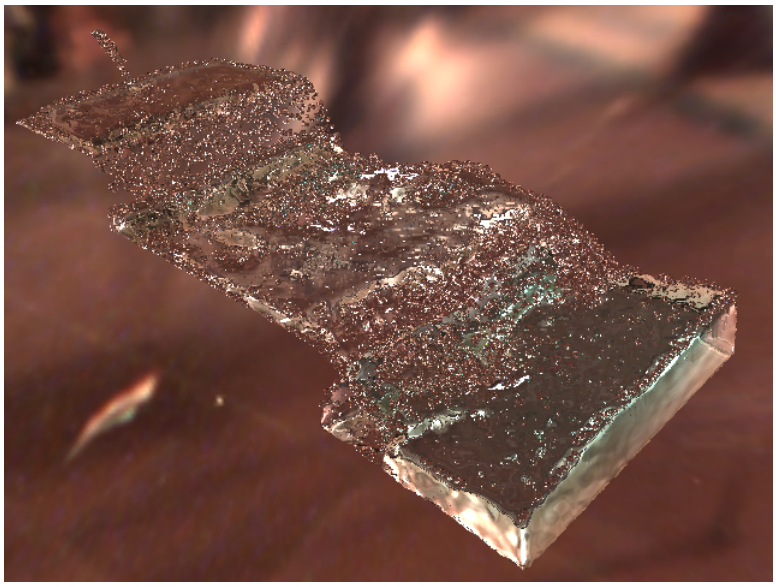


(b) Beer-Lambert の法則を考慮した場合

図 5.1: 提案手法による「波」シーンの描画結果.



(a) Beer-Lambert の法則を考慮しなかった場合



(b) Beer-Lambert の法則を考慮した場合

図 5.2: 提案手法による「段差」シーンの描画結果.

表 5.1: 提案手法を用いて描画する際にかかるフレームあたりの計算時間. 単位はミリ秒である. 括弧内のパーセンテージは描画にかかる時間全体に占める処理時間の割合を表す.

	640 × 480 の 解像度での計算時間	1024 × 768 の 解像度での計算時間
「波」シーン		
総描画時間	4.21	9.54
デプスマップの作成	1.4 (33.25%)	3.06 (32.08%)
デプスマップの平滑化	2.16 (51.31%)	4.9 (51.36%)
法線マップの作成	0.18 (4.28%)	0.45 (4.72%)
屈折の計算	0.47 (11.16%)	1.13 (11.84%)
「段差」シーン		
総描画時間	3.87	8.97
デプスマップの作成	0.56 (14.47%)	1.1 (12.26%)
デプスマップの平滑化	2.47 (63.82%)	5.94 (66.22%)
法線マップの作成	0.18 (4.65%)	0.45 (5.02%)
屈折の計算	0.66 (17.05%)	1.48 (16.50%)

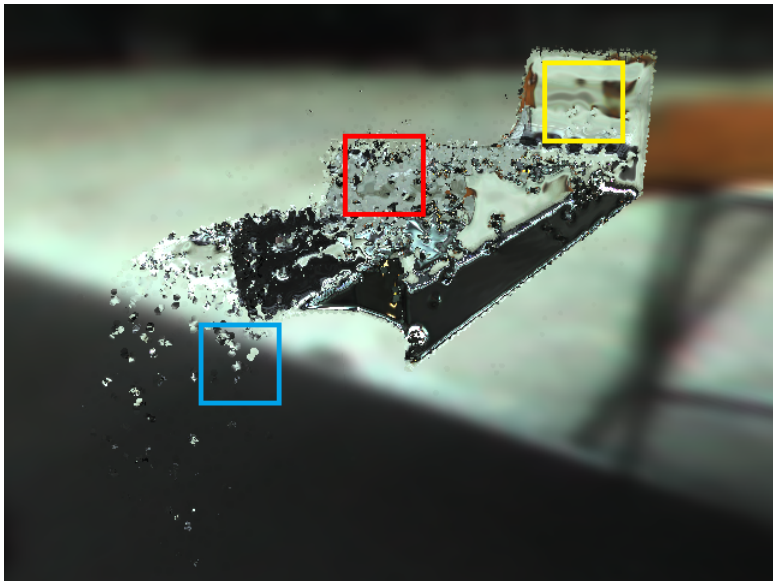


図 5.3: 卒業研究の手法による「波」シーンの描画結果.

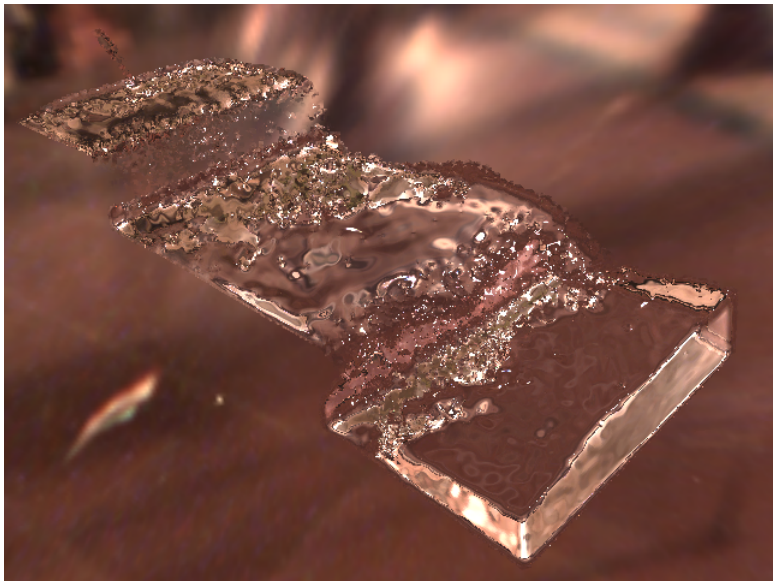


図 5.4: 卒業研究の手法による「段差」シーンの描画結果.

表 5.2: 卒業研究の手法を用いて描画する際にかかるフレームあたりの計算時間. 単位はミリ秒である. 括弧内のパーセンテージは描画にかかる時間全体に占める処理時間の割合を表す.

	640 × 480 の解像度での計算時間	1024 × 768 の解像度での計算時間
「波」シーン		
総描画時間	6.26	15.5
デプスマップの作成	0.52 (8.31%)	1.16 (7.48%)
デプスマップの平滑化	5.54 (88.50%)	13.83 (89.23%)
法線マップの作成	0.09 (1.44%)	0.23 (1.48%)
屈折の計算	0.11 (1.76%)	0.28 (1.81%)
「段差」シーン		
総描画時間	6.03	14.89
デプスマップの作成	0.28 (4.64%)	0.53 (3.56%)
デプスマップの平滑化	5.54 (91.87%)	13.83 (92.88%)
法線マップの作成	0.09 (1.49%)	0.23 (1.54%)
屈折の計算	0.12 (1.99%)	0.30 (2.01%)

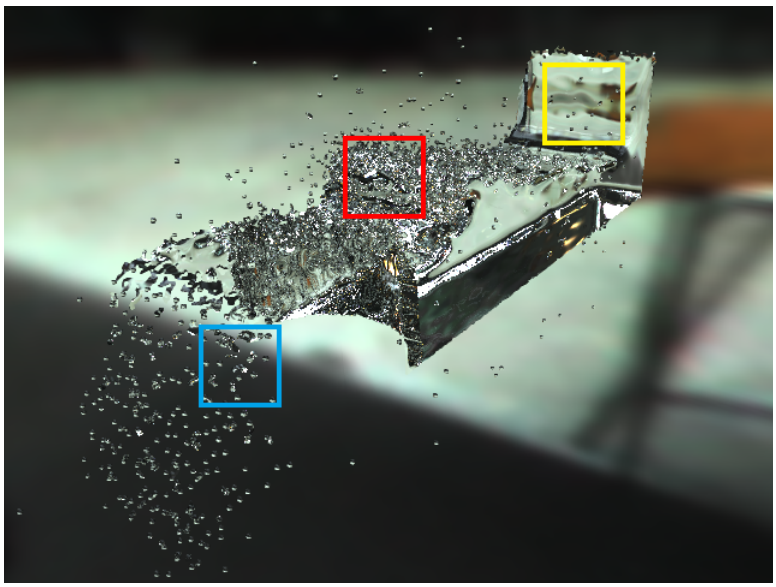


図 5.5: 「波」シーンにおいて提案手法で作成した液面をポリゴンメッシュ化し、レイトレーシングを用いて描画した結果.

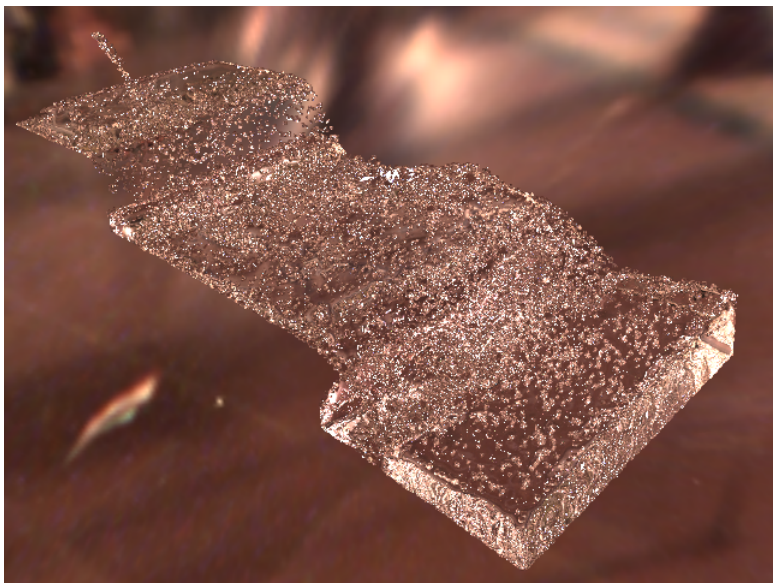
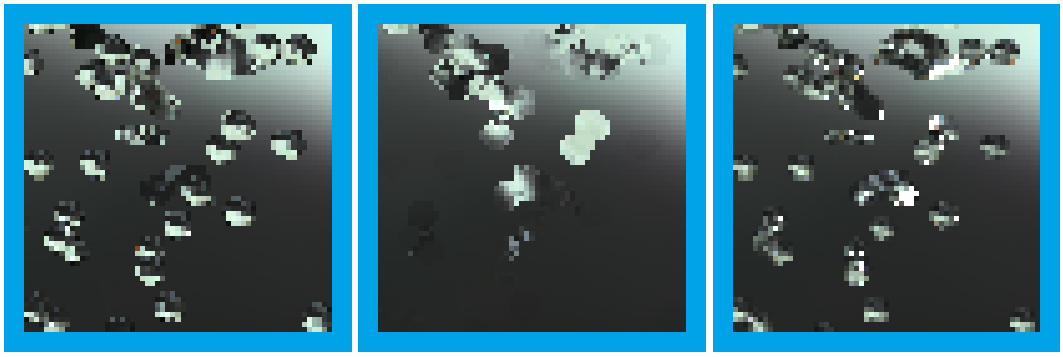


図 5.6: 「段差」シーンにおいて提案手法で作成した液面をポリゴンメッシュ化し、レイトレーシングを用いて描画した結果.

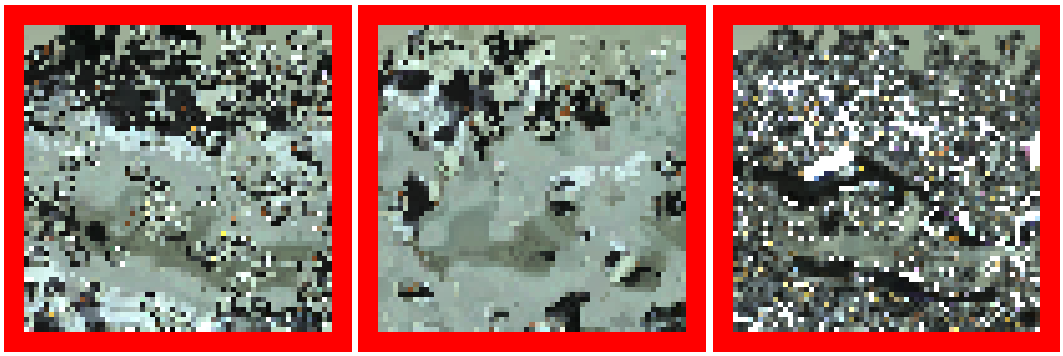


(a) 図 5.1(a) の拡大図（提案手法）(b) 図 5.3 の拡大図（卒業研究の手法）(c) 図 5.5 の拡大図（レイトレーシング）

図 5.7: 図 5.1(a), 図 5.3, 図 5.5 における青の矩形領域の拡大図.

(a) 図 5.1(a) の拡大図（提案手法）(b) 図 5.3 の拡大図（卒業研究の手法）(c) 図 5.5 の拡大図（レイトレーシング）

図 5.8: 図 5.1(a), 図 5.3, 図 5.5 における黄の矩形領域の拡大図.

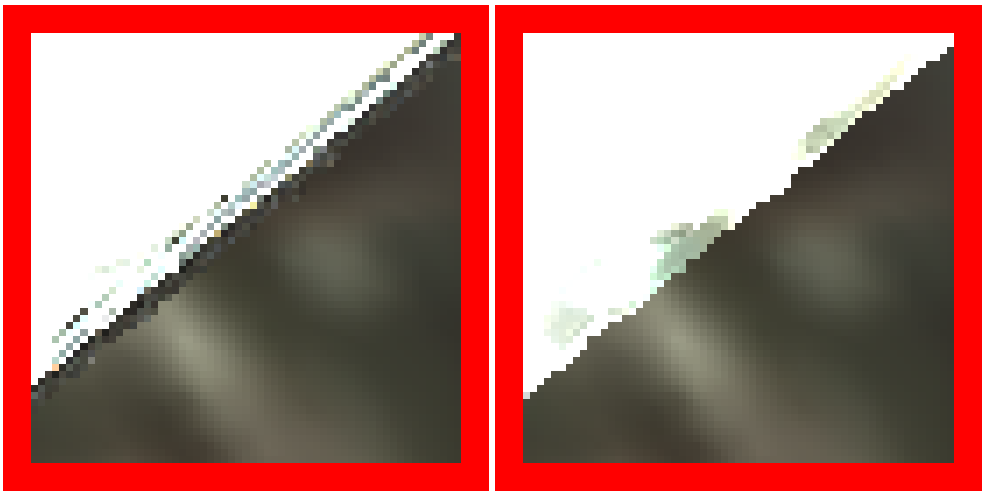


(a) 図 5.1(a) の拡大図（提案手法）(b) 図 5.3 の拡大図（卒業研究の手法）(c) 図 5.5 の拡大図（レイトレーシング）

図 5.9: 図 5.1(a), 図 5.3, 図 5.5 における赤の矩形領域の拡大図.

図 5.10: 局所的な平面フィッティングによる平滑化を用いた場合（提案手法）の「波」シーンの描画結果.

図 5.11: バイラテラルフィルタを用いた場合（卒業研究の手法）の「波」シーンの描画結果.



(a) 図 5.11 の拡大図（バイラテラルフィルタを用いた場合）

(b) 図 5.10 の拡大図（局所的な平面フィッティングによる平滑化を用いた場合）

図 5.12: 図 5.10 および図 5.11 における赤の矩形領域の拡大図.

表 5.3: 局所的な平面フィッティングを利用した平滑化においてパラメータを変化させた際における 640×480 の解像度での平滑化にかかる時間. 単位はミリ秒.

	平滑化回数			
カーネルサイズ	1	2	3	4
7×7	0.55	1.09	1.63	2.15
9×9	0.87	1.71	2.56	3.40
11×11	1.28	2.55	3.81	5.08
13×13	1.77	3.51	5.23	6.98

表 5.4: バイラテラルフィルタのパラメータを変化させた際における 640×480 の解像度での平滑化にかかる時間. 単位はミリ秒.

	平滑化回数			
カーネルサイズ	11	22	33	44
5×5	1.49	2.98	4.48	5.96
7×7	2.72	5.43	8.13	10.98
9×9	4.02	8.14	12.38	17.29
11×11	5.96	13.01	19.81	26.01

図 5.13: 局所的な平面フィッティングを利用した平滑化においてパラメータを変化させた際の描画結果.

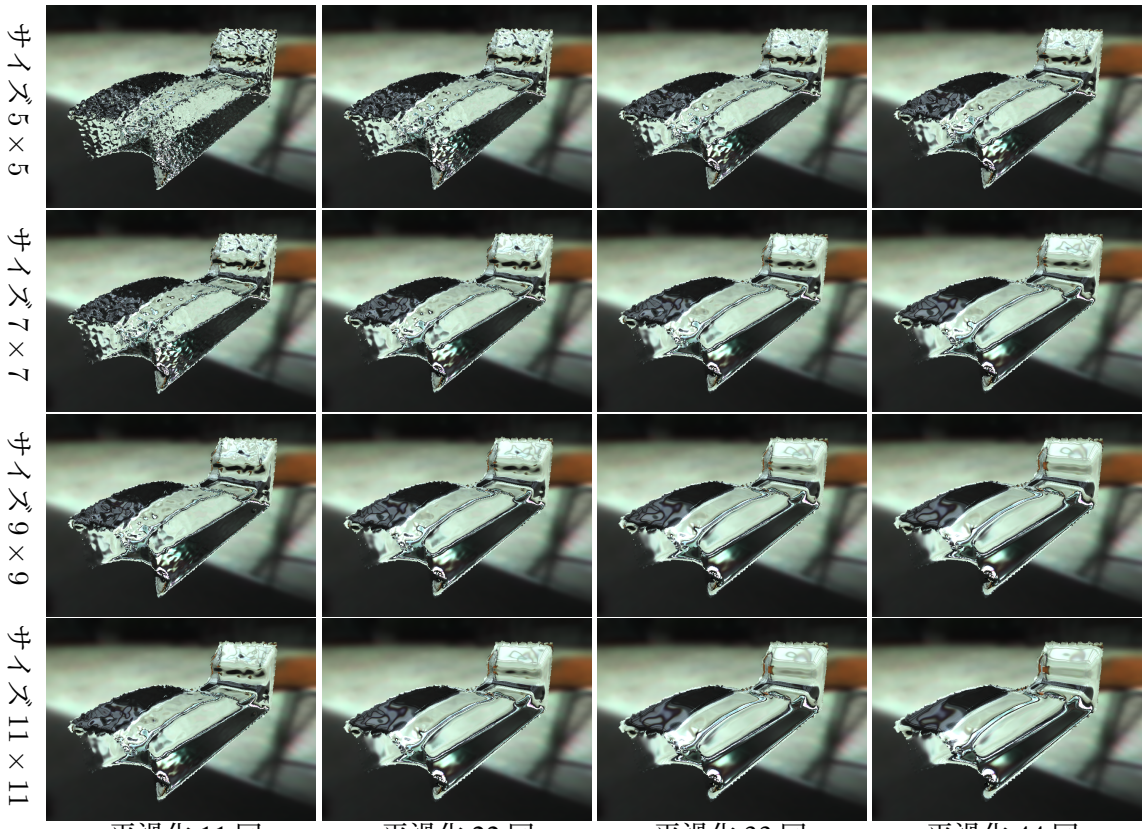


図 5.14: バイラテラルフィルタのパラメータを変化させた際の描画結果.

第6章　結論

本論文では粒子の集合として表現された液体に対し複雑な屈折を考慮した実時間描画手法を提案した。液面を抽出する既存手法とポリゴンメッシュに対し 2 回屈折を考慮して実時間描画を行う既存手法を組み合わせた卒業研究の手法 [1] を利用した。さらに描画する粒子をその近傍にある粒子の数により液体部分と飛沫部分に分け、それぞれに属する粒子から液面を抽出してそれらを屈折計算に利用することで複雑な屈折を計算できるように拡張した。また、バイラテラルフィルタによる液面の平滑化で発生した問題点を修正するため、局所的な平面フィッティングを利用した平滑化を導入した。提案手法は既存のパーティクルシステムに組み込むことで利用でき、描画に関する部分は既存のプログラム可能な GPU を用いて実現できる。

本手法を改善しさらに写実性を高める方法として、Davis と Wyman[16] の手法を導入し液面の反射に対しても再帰的に計算していくことが挙げられる。

提案手法では飛沫を含むシーンにおいて卒業研究の手法よりも高品質な描画結果を生成することに成功し、また描画にかかる時間を短縮することに成功した。一方で大量の飛沫を含むシーンにおいてはより高品質な結果を得ることができなかった。この問題を解決するためには、例えばそれぞれ独立した液面として作成した飛沫の液面を利用した屈折計算の実現が挙げられる。Dual Depth Peeling[20] を繰り返し利用すれば複数枚のデプスマップを取得し液面を得ることができるが、すべての飛沫の液面をデプスマップとして保持することは現実的ではない。さらに効果的な液体の描画手法を考えていく必要がある。

謝辞

本研究を進めるにあたりご指導とご協力をいただいた本学システム情報系の三谷純教授および金森由博助教に深く感謝いたします。また、本論文の執筆にあたりご協力いただいた非数値処理アルゴリズム研究室の皆様、研究生活を支えてくださった私の家族に御礼を申し上げます。

参考文献

[1] 今井拓也. スクリーン空間での 2 回屈折を考慮した液体の実時間描画. 筑波大学情報学群情報メディア創成学類卒業論文, 2014.

[2] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum*, Vol. 27, No. 2, pp. 351–360, 2008.

[3] Olivier Gourmel, Anthony Pajot, Mathias Paulin, Loïc Barthe, and Pierre Poulin. Fitted BVH for fast raytracing of metaballs. *Computer Graphics Forum*, Vol. 29, No. 2, pp. 281–288, 2010.

[4] Roland Fraedrich, Stefan Auer, and Rüdiger Westermann. Efficient high-quality volume rendering of SPH data. *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 16, No. 6, pp. 1533–1540, 2010.

[5] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. Parallel surface reconstruction for particle-based fluids. *Comput. Graph. Forum*, Vol. 31, No. 6, pp. 1797–1809, 2012.

[6] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, Vol. 32, No. 1, pp. 5:1–5:12, 2013.

[7] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pp. 154–159. Eurographics Association, 2003.

[8] Bart Adams, Toon Lenaert, and Philip Dutré. Particle splatting: Interactive rendering of particle-based simulation data. Technical report, Departement Computerwetenschappen, KU Leuven, 2006.

[9] Matthias Müller, Simon Schirm, and Stephan Duthaler. Screen space meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pp. 9–15, 2007.

[10] Hilko Cords and Oliver Staadt. Instant liquids. In *Poster proceedings of ACM Siggraph/Eurographics symposium on computer animation*, 2008.

[11] Wladimir J. van der Laan, Simon Green, and Miguel Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, pp. 91–98, 2009.

[12] Florian Bagar, Daniel Scherzer, and Michael Wimmer. A layered Particle-Based fluid model for Real-Time rendering of water. *Computer Graphics Forum*, Vol. 29, No. 4, pp. 1383–1389, 2010.

[13] コンピュータグラフィックス編集委員会. コンピュータグラフィックス. CG-ARTS 協会, 2004.

[14] Chris Wyman. An approximate image-space approach for interactive refraction. *ACM Trans. Graph.*, Vol. 24, No. 3, pp. 1050–1053, 2005.

[15] Manuel M. Oliveira and Maicon Brauwers. Real-time refraction through deformable objects. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pp. 89–96, 2007.

[16] Scott T Davis and Chris Wyman. Interactive refractions with total internal reflection. In *Proceedings of Graphics Interface 2007*, pp. 185–190, 2007.

[17] Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, Vol. 13, No. 3, pp. 233–246, 1994.

[18] Stefan Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proceedings of the Vision, Modeling, and Visualization Conference 2003 (VMV 2003)*, pp. 245–252, 2003.

[19] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus Gross. GPU-based ray-casting of quadratic surfaces. In *Proceedings of the 3rd Eurographics/IEEE VGTC conference on Point-Based Graphics*, pp. 59–65, 2006.

[20] Louis Bavoil and Kevin Myers. Order independent transparency with dual depth peeling. Technical report, NVIDIA, 2008.