ベクトルポテンシャルを用いた流体流れ場の編集

Editing Fluid Flow using Vector Potential

佐藤 周平 [1†]　土橋 宜典 [2,1]　楽 詠灝 [3]　岩崎 慶 [4,1]　西田 友是 [5,1]

Syuhei SATO[1†]　Yoshinori DOBASHI[2,1]　Yonghao YUE[3]　Kei IWASAKI[4,1]　Tomoyuki NISHITA[5,1]

1 UEI リサーチ　1 UEI Research
2 北海道大学, JST CREST　2 Hokkaido University, JST CREST
3 コロンビア大学　3 Columbia University
4 和歌山大学　4 Wakayama University
5 広島修道大学　5 Hiroshima Shudo University

E-mail: † syuhei.sato@uei.co.jp

1. はじめに

近年，映画やゲームなどの映像制作において，写実的な流体映像を作成するために，物理ベースのシミュレーション[2]がよく利用される．しかし，物理ベースのシミュレーションは非常に計算コストが高い．そのため，所望の流体映像を得るためには異なるパラメータセットで何度もシミュレーションを繰り返す必要があり，映像制作全体にかかる時間が非常に長くなる．再度シミュレーションを実行せずに，流体の流れ場を編集できれば，このような問題の解決が期待できる．しかし，単純に流れ場を編集した場合，流体の非圧縮性が保たれず，意図しない場所から流体が湧き出したり，消えたりする．

そこで，本研究では，非圧縮性を保持した流体流れ場の編集手法を提案する．このような編集を実現するために，我々はベクトルポテンシャルを用いて，流れ場を表現する．提案手法では，まず，入力の速度場からベクトルポテンシャル$\mathbf{\Psi}$を計算する．次に，ユーザが編集操作を行い，ベクトルポテンシャルに編集に対応した処理を行う．そして，編集後のベクトルポテンシャルに対して，curl 演算子($\nabla\times$)を適用することで，編集が適用された速度場を得る．任意のベクトルポテンシャル$\mathbf{\Psi}$について，$\nabla\cdot\nabla\times\mathbf{\Psi}=\mathbf{0}$が成り立つため，我々の手法は常に非圧縮性を保証することができる．提案手法を用いることで,ユーザは流れ場を編集して,様々な流れ場を作成することが可能である．編集要素としては，流れ場の変形と障害物の追加を扱う．

2. 関連研究

流体シミュレーション：Stam は，Navier-Stokes 方程式を安定に解くための手法を提案した[23]．Stam の手法以降，様々な流体現象をシミュレーションするための手法が数多く提案されている[6-8,11,16]．一般的に，所望の流体アニメーションを得るためには，シミュレーションの実行とパラメータの調整を繰り返し行う必要がある．これは非常に煩雑な作業である．

流体制御：所望の形状の流体アニメーション[5,22,26]やユーザ指定の曲線に沿って動く流体[14]を作成するために，いくつかの制御手法が提案されている．これらの手法は，外力を追加することによって流体の動きを制御する．これらの手法を用いて，様々な流体アニメーションを作成する場合，ユーザは複数回シミュレーションを実行する必要がある.一方我々の手法では，再度流体シミュレーションを実行する必要なく，様々な流体アニメーションを作成することが可能である．

モデルリダクション：流体シミュレーションを高速化するための一つのアプローチとして，モデルリダクション手法が提案されている[25,27].これらの手法では，様々な初期条件，パラメータでシミュレーションをして得られた速度場のセットを用意し，その速度場に対し主成分分析を適用する．そして，主成分を基底関数として用いることで，Navier-Stokes 方程式を高速に計算できる．しかし，これらの手法では，基底の線形和により流れ場を表現しているため，入力データから大きく異なる流れ場は作成することができない．また，様々な流れ場を作成可能とするためには，多くの速度場データが必要であり，その結果として，データベースを構築する段階で，繰り返し流体シミュレーションを実行しなければならない．上記の手法をベースとして，流体の高速な再シミュレーションを可能とする手法が提案されている[13]．この手法では，流体シミュレーションにより得られた単一の速度場のセットに対し主成分分析を適用し，基底関数を得る．そして，その基底が張る空間で Navier-Stokes 方程式を解くことで，異なるパラメータ設定の流れ場を効率的に計算することができる．しかし，この手法でも，[25,27]の手

法と同様にデータの線形和として表現できない流れ場は作成できない．従って，提案手法が目標とするような, 全体的な流れを変更するといったことはできない．

手続き的手法：手続き的な手法は，比較的低コストで所望の流れ場を作成することができる．このような手法の一例として，様々な炎のアニメーションを作成するための手法が提案されている[9,15]．これらの手法では，炎の経路を表す曲線を変形することで，所望の結果を生成することができる．しかし，物理的な正確さは考慮していないため，不自然な結果が作成される場合がある．また，流れ場は非圧縮性を満たさない．Pighin らは，シミュレーションされた流れ場を，advected radial basis functions により表現し，流れ場を編集するための手法を提案した[18]．しかし，この手法も結果の流れ場に対し非圧縮性を保証していない．

シミュレーション結果の再利用：Reveendran らはシミュレーションにより作成された複数の水のアニメーションからその間の状態を補間する手法を提案した[19]．この手法では，水の表面を表すメッシュデータから time-space mesh を作成し, non-rigid iterated closest point method (non-rigid ICP)によりデータ間の対応をとることで，尤もらしい中間の状態のメッシュを得ることができる．しかし，この手法は水面を表すメッシュに対して適用するものであり，我々が扱うような格子に格納されたデータを扱うことは考慮されていない．我々はこれまで，流れ関数を用いることで，非圧縮性を保持して 2D の流れ場を変形するための手法を提案した[20]．本稿では，ベクトルポテンシャルを用いることで, 3D の流れ場に対してそのような変形を実現するための手法を提案する．

ベクトルポテンシャルの計算：速度場からベクトルポテンシャルを求める方法は，数値流体力学の分野にも存在する．例えば，渦法では，速度場を渦度のベクトルポテンシャルであるとみなし，渦度場から速度場へ変換する[4,10,17,28,29]．我々の知る限りでは，我々の手法は流体の編集のためにベクトルポテンシャルを使用した初の方法である．本稿では，我々の問題に適切な境界条件と方程式について議論する．

3. 入力と仮定

提案手法の入力は，非圧縮な速度場の単一のセット$\mathbf{v}(t)$である．ここで，$t(=0,1,\cdots,T-1)$はフレーム番号を表し，Tは入力の速度場のフレーム数である．提案手法では，$\mathbf{v}(t)$が空間的に十分なめらかであり，単連結な閉領域$\Omega \subset \mathbb{R}^3$($\Omega$は種数 0)に定義されていると仮定する．また，$\mathbf{v}(t)$のシミュレーション空間の境界$\partial\Omega$の法線方向成分が$0$であると仮定する：$\mathbf{v}(t)\cdot\mathbf{n}=0(\mathrm{at}\ \ \partial\Omega)$．ここで，$\mathbf{n}$は境界の法線である．

また，密度場として表現される流体が$\mathbf{v}(t)$に従って移流すると仮定する．提案手法では，速度場を置き換え，新しい速度場に従って流体を移流させることで，様々なアニメーションを作成する．以下では，簡潔な表記のためにtを省略する．

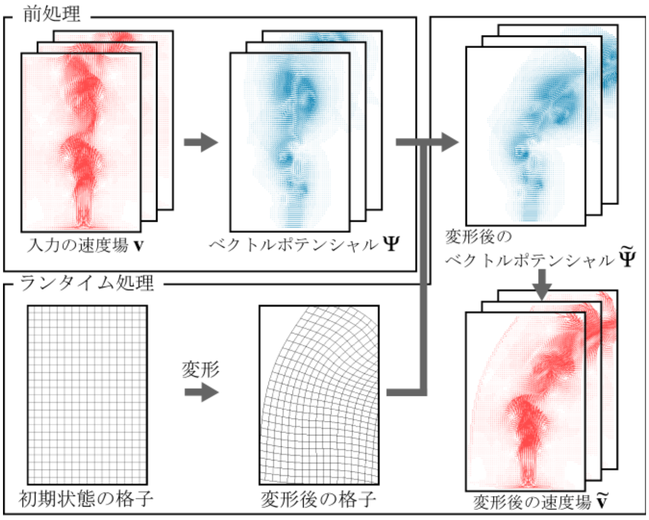


図 1：提案手法の概要

4. 提案手法の概要

速度場や密度場を直接操作する代わりに，提案手法では，ベクトルポテンシャルを用いることで，編集された流れ場に対し，非圧縮性を保証する．単連結な閉領域において，Helmholtz -Hodge 分解の定理[1,24]により, 十分になめらかなベクトル場$\mathbf{v}$は次のように分解できる．

$$\mathbf{v}=\nabla\times\mathbf{\Psi}+\nabla p, \tag{1}$$

ここで，$\mathbf{\Psi}$はベクトルポテンシャル，pはスカラー場である．$\nabla\times\mathbf{\Psi}$のシミュレーション空間の境界の法線方向成分は$0$である:

$$(\nabla\times\mathbf{\Psi})\cdot\mathbf{n}=0\ \ \mathrm{at}\ \ \partial\Omega, \tag{2}$$

また，∇pは境界に垂直である．

我々の入力の速度場$\mathbf{v}$は非圧縮なので，$\nabla p=0$であり，従って，

$$\mathbf{v}=\nabla\times\mathbf{\Psi}. \tag{3}$$

そして，恒等式$\nabla\cdot\nabla\times\mathbf{\Psi}=0$が任意のベクトル場$\mathbf{\Psi}$に対して満たされるため，ベクトルポテンシャルの回転は常に非圧縮条件を満たす．そのため，ベクトルポテンシャルに対し編集を行うことで，結果の流れ場に非圧縮性を保証することができる．

図 1 に提案手法の概要を示す．視覚的にわかりやすくするため，全ての 3 次元ベクトル場を，2 次元ベク

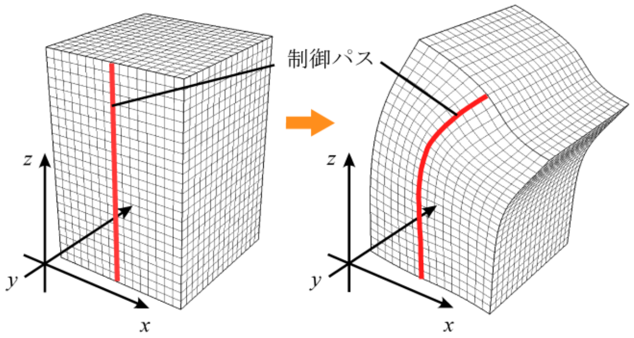


図 2：格子の変形

トル場で図示した．前処理では，まず，格子ベースの方法により，非圧縮性 Navier-Stokes 方程式を解くことで，入力の速度場$\mathbf{v}$を作成する．そして，各フレームの速度場$\mathbf{v}$をベクトルポテンシャル$\boldsymbol{\Psi}$へ変換する．ランタイムでは，ユーザが編集操作を行い，その編集に対応する処理をベクトルポテンシャルに適用することで，編集後のベクトルポテンシャル$\tilde{\boldsymbol{\Psi}}$を得る．そして，curl 演算子を$\tilde{\boldsymbol{\Psi}}$に適用することで，編集後の速度場$\tilde{\mathbf{v}}$を生成する．最後に，$\tilde{\mathbf{v}}$に従って密度場を移流させる．

5. ベクトルポテンシャルの計算

Helmholtz-Hodge 分解の定理に従い，入力の非圧縮な速度場$\mathbf{v}$は，ベクトルポテンシャル$\boldsymbol{\Psi}$を用いて，式(3)のように表すことができる．ただし，このベクトルポテンシャル$\boldsymbol{\Psi}$は通常一意に求めることができない．これは，$\boldsymbol{\Psi}$が$\mathbf{v} = \nabla \times \boldsymbol{\Psi}$を満たすと仮定した場合，あるベクトルポテンシャル$\boldsymbol{\Psi}' = \boldsymbol{\Psi} + \nabla q$も，任意のスカラー場$q$に対して，定義$\nabla \times \nabla q = 0$より，$\mathbf{v} = \nabla \times \boldsymbol{\Psi}'$を満たすためである．我々は，時間変化するベクトル場の変形に適するよう，この自由度を拘束する．以下では編集操作のうち変形を例として説明する．

まず，式(1)に従い，ベクトルポテンシャル$\boldsymbol{\Psi}$を$\boldsymbol{\Psi} = \nabla \times \boldsymbol{\Phi} + \nabla s$のように分解する．ここで，$\boldsymbol{\Phi}$はベクトル場，$s$はスカラー場である．そして，変形を表す関数$F$により$\boldsymbol{\Psi}$を変形する．$F$は，例えば，各位置ベクトル$\mathbf{X}$を新しい位置$\mathbf{x}$へマップする関数を表す．この変形では，ベクトル値関数$\mathbf{g}(\mathbf{X})$は$\tilde{\mathbf{g}}(\mathbf{x}) = (\partial \mathbf{x}/\partial \mathbf{X})\mathbf{g}(\mathbf{X})$のように変換され，$F(\mathbf{g}) = \tilde{\mathbf{g}}$と記述することとする．これにより我々は，ベクトル値関数$\mathbf{g}$と$\mathbf{h}$に対して，$F(\mathbf{g}+\mathbf{h}) = \tilde{\mathbf{g}} + \tilde{\mathbf{h}} = F(\mathbf{g}) + F(\mathbf{h})$を得る．そして，変形後のベクトルポテンシャルは以下のようになる．

$$F(\boldsymbol{\Psi}) = F(\nabla \times \boldsymbol{\Phi} + \nabla s) = F(\nabla \times \boldsymbol{\Phi}) + F(\nabla s). \quad (4)$$

上式に curl 演算子を適用した場合，任意の変形に対して，$\nabla \times F(\nabla s)$は常に 0 になるとは限らない．そのため，もし$\nabla s$に時間的なコヒーレンスがない場合，$\nabla \times F(\nabla s)$は結果の流れ場に意図しない揺らぎを引き起こす可能性がある．従って，我々はそのような問題を避けるために，∇sの値を拘束する．本稿では特に，以下の設定により$\nabla s = 0$を強制する．

$$\boldsymbol{\Psi} \cdot \mathbf{n} = 0 \ \text{at} \ \partial\Omega, \quad (5)$$

$$\nabla \cdot \boldsymbol{\Psi} = 0 \ \text{at} \ \Omega. \quad (6)$$

速度場$\mathbf{v}$から所望のベクトルポテンシャル$\boldsymbol{\Psi}$を得るために，我々はまず，式(3)の両辺に curl 演算子($\nabla \times$)を適用し次式を得る．

$$-\nabla^2 \boldsymbol{\Psi} = \nabla \times \mathbf{v}, \quad (7)$$

我々はこのポアソン方程式を境界条件$\boldsymbol{\Psi} \cdot \mathbf{n} = 0$，$(\nabla \times \boldsymbol{\Psi}) \cdot \mathbf{n} = 0$，$\nabla \cdot \boldsymbol{\Psi} = 0$の下で解くことにより，ベクトルポテンシャル$\boldsymbol{\Psi}$を得る（付録 A に証明を示す）．式(7)を数値的に解くために，biconjugate gradient stabilized method (BiCGSTAB)を本稿では採用した．

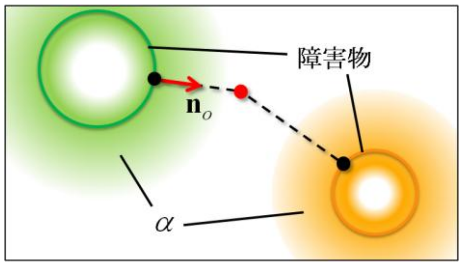


図 3：障害物の追加

6. 編集後の速度場の生成

本手法はベクトルポテンシャルに対して，ユーザの編集操作に対応した処理を適用する．本稿では，編集要素として変形および障害物の追加を扱う．

流れ場の変形：本稿では，制御点や制御パス（図 2 は制御パスの例）を用いて，2 次元（図 2 のxz平面）で変形操作を行い，それを 3 次元格子のy方向の各 2 次元スライスに適用する．変形手法には移動最小二乗法に基づく方法[21]を用いた．そして，変形後のベクトルポテンシャル$\tilde{\boldsymbol{\Psi}}$を得るために，我々はまず$\boldsymbol{\Psi}$が格納されている変形後の格子を，$\tilde{\boldsymbol{\Psi}}$を格納するための直交格子で再サンプリングする．そして変形後のベクトルポテンシャルの正確な方向を得るために，ベクトルポテンシャルの値に変形に対応する局所的な回転を適用する．提案手法では，[21]の方法に限らず，画像やメッシュを変形するための任意の変形手法を適用できる．しかし，格子が裏返るような変形が適用された場合，裏返りがベクトル場に不連続を引き起こす可能性があり，その結果大きな速度が生成される場合がある．また，変形の度合いが大きい場合，速度が意図せず大きく変化してしまう．これは，変形が大きくなると，格子点間のエッジの長さが変化し，$\nabla \times \boldsymbol{\Psi}$の値が大きく変化することが原因である．

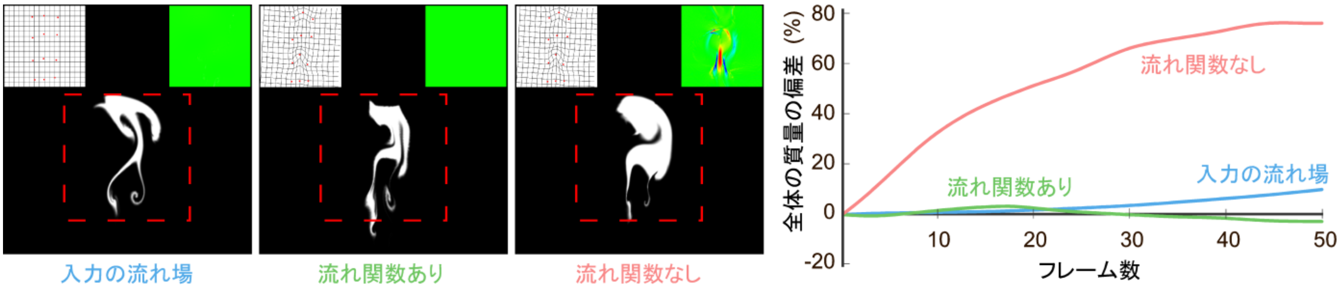


図 4：非圧縮性を保った変形の重要性

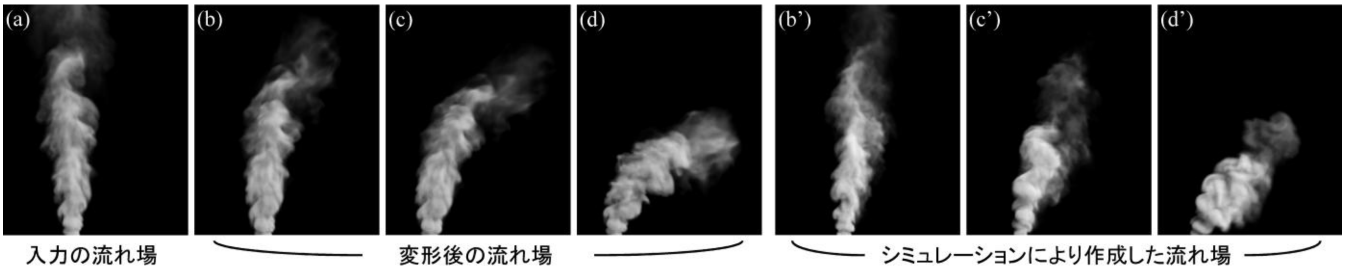
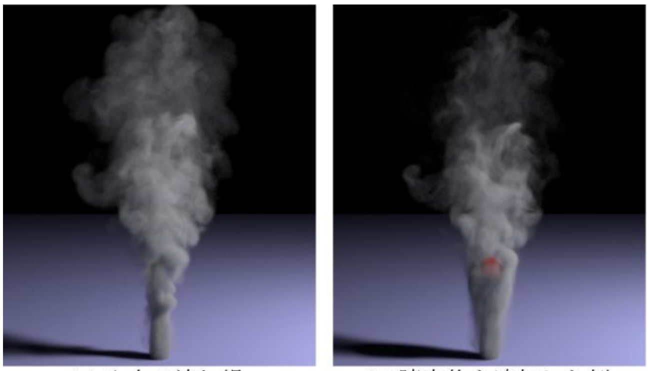


図 5：提案手法と流体シミュレーションによる結果との比較



(a) 入力の流れ場　(b) 障害物を追加した例

図 6：障害物の追加の例

障害物の追加： 障害物の追加は，Bridson らの方法[3]を用いる．この論文では，ノイズ関数を用いてポテンシャル場を生成することで，手続き的に流体のような流れ場を作成できる．そして，ポテンシャル場に対し以下の式を適用することで，非圧縮性を保ったまま障害物の影響を考慮した流れ場を生成できる．

$$\widetilde{\mathbf{\Psi}} = \alpha\mathbf{\Psi} + (1-\alpha)\mathbf{n}_o(\mathbf{n}_o \cdot \mathbf{\Psi}), \tag{8}$$

ここで，$\mathbf{n}_o$はある格子点（ここでは図 3 の赤点）から最も近い距離にある障害物上の点の法線（図 3 赤矢印）を表す．また，αは格子点と最も近い障害物上の点との距離に応じて変化する係数であり，障害物に近い格子点ほど 0 に近くなる（図 3 緑と橙色のグラデーション部分：各色が濃いほどα が 0 に近い）．上記の方法について，詳細は文献[3]を参照していただきたい．

ユーザが上記の編集操作を行い，編集後のベクトルポテンシャル$\widetilde{\mathbf{\Psi}}$が計算された後，$\widetilde{\mathbf{\Psi}}$に curl 演算子を適用することで，編集後の速度場$\tilde{\mathbf{v}} = \nabla \times \widetilde{\mathbf{\Psi}}$を得る．結果の速度場は，$\nabla \times (\nabla \times \widetilde{\mathbf{\Psi}})$が 0 となるため，常に非圧縮性を満足する．

表 1：格子数および計算時間

図	格子数	T_{ns}	T_p	T_r
5	256x128x384	146	298	2.9
7,9	192x192x512	234	407	4.4

7. 実験結果

計算に使用した PC は，CPU が Intel Core i7 3930K であり，メモリは，32GB である．図 6,7,8 および図 9 の右の画像は物理ベースレンダラの"Mitsuba"[12]を用いてレンダリングした．なお本節のアニメーション例については，補足資料の動画ファイルを参照していただきたい．

非圧縮を保った変形の重要性： 速度場の非圧縮性を保持することは，全体の質量を保存するために重要な要素である．図 4 にこの重要性を示す．視覚的にわかりやすくするため，2D の流れ場を用いた．2D においてベクトルポテンシャルは流れ関数と呼ばれるスカラー関数として表される．本実験では，我々の従来手法[20]を用いて，2D の速度場から流れ関数を計算した．

この例では，ゼロでない密度場が 2D の流体シミュレーションにより生成された時間変化する速度場にしたがって移流される．シミュレーション空間の中央下端には，毎フレーム一定の上向きの速度がセットされ

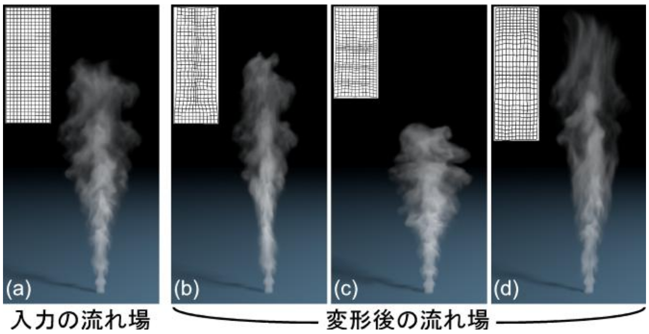


図 7：様々なアニメーションの作成例

図 8：複数の煙突から立ち上る煙の例

る．また，ランタイムにおいて，密度の追加や除去は行わない．図 4 の左 3 つの画像は，密度場を可視化したものであり，各画像の左上の図は変形に用いた格子である．また，右上の図は赤色の矩形で示した領域の速度場の発散を示したものであり，青，緑，および赤色はそれぞれマイナス，ゼロ，およびプラスの発散を表す．図 4 右のグラフは全体の質量の偏差を表す．偏差は(m_t/m_0-1)のように算出し，mはシミュレーション領域全体にわたって密度を積分することで算出される．流れ関数を使わずに直接速度場を変形した場合，結果の速度場に非ゼロの発散が生じ，その結果全体の質量が初期値から大幅に逸脱してしまう．これに対し，非圧縮性が保たれている場合，質量の時間的な偏差を十分に減少させることができる．また，我々は，3D の流れ場の変形についても同様の傾向を確認している．

編集結果とシミュレーション結果との比較：図 5 に提案手法により作成された結果と流体シミュレーションにより作成された結果の比較を示す．図 5a は入力の煙のアニメーションであり，図 5b'-d'は流体シミュレーションにより作成された結果である．シミュレーションには，煙が右に流れるよう，シミュレーション空間の左端に一様な外力を適用している．図 5b-d は提案手法により作成された結果であり，変形により図 5b'-d'のような流れを模倣できることを示す．変形が小さい

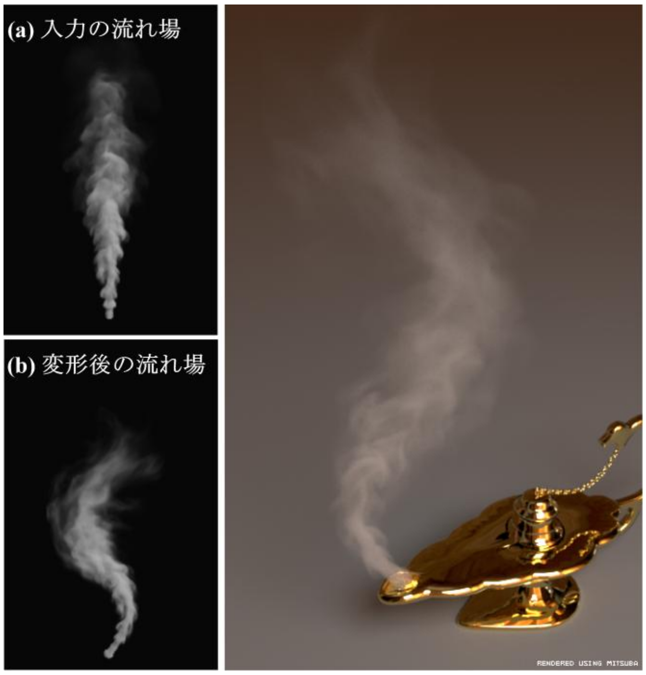


図 9：魔法のランプから立ち上る煙の例

場合，我々の結果はシミュレーションに近い結果が作成できる．また，比較的大きな変形でも尤もらしいアニメーションを作成できる．

3D の流体の編集:表 1 に各結果の格子数と計算時間を示す．計算時間の単位は[sec/frame]である．T_{ns}は流体シミュレーションの時間，T_pおよびT_rはそれぞれ提案手法における前処理とランタイムの計算時間である．提案手法では，シミュレーションの時間に比べ，前処理に約 2 倍の計算時間がかかっている．しかし，ランタイムの計算は，約 50 倍高速になっており，所望の映像を作成する際の試行錯誤を効率的に行うことができる．また，提案手法で行う計算は，前処理での Navier-Stokes 方程式とポアソン方程式，ランタイムでの変形処理と curl 演算であり，並列化が有効である．

図 6 は，障害物を追加した場合の例である．格子数は 256x256x384 である．図 6a は，入力の流れ場であり，図 6b は，a に障害物（赤球）を追加しその影響を考慮した流れ場である．障害物が追加されたことで，球の下に煙が滞留しており，密度が濃くなっているのがわかる．また，その影響で，球の上方に立ち上る煙の量が入力に比べて少なくなっており，障害物を考慮した影響を確認することができる．

単一の流れ場（図 7a）から，提案手法は様々な流れ場を作成することができる．この例では格子を水平方向や鉛直方向に縮めたり，鉛直方向に引き延ばしたりすることで，入力と比べ，細い煙（図 7b），太い煙（図 7c），高い煙（図 7d）のアニメーションを作成できる．

各画像の左上の図は変形に用いた格子である．

図 8 は，図 5b-d の結果を使用して，複数の煙突から立ち上る煙のシーンを作成した例である．このように，提案手法では，単一のデータから複数の流体アニメーションを効率的に作成できる．

図 9 に魔法のランプから立ち上る煙の例を示す．図 9a は入力の流れ場であり，b は提案手法により得られた結果である．図 9a では，煙が垂直に立ち上っている．図 9b では，a から蛇行するような流れを作成した．また，図 9 右の画像は b の結果を用いて作成した．提案手法では，このようなシミュレーションのパラメータ調整だけでは作成が難しい流れも作成できる．

リミテーション：提案手法はベクトルポテンシャルを変形するため，速度場において変形結果が直観的でない場合がある．例えば，ベクトルポテンシャルの x, y, z 成分が，$\mathbf{v} = \nabla \times \boldsymbol{\Psi}$において組み合わされているため，平面においてベクトルポテンシャルを変形した場合，平面に垂直な方向にも速度場が変化する．また，変形が非常に大きくなると，同様に直観的な変形が難しい．そのため，今後変形の度合いや種類に応じて起こる変化を定量化する予定である．

8. まとめと今後の課題

本稿では，非圧縮性を保ちつつ流体の流れ場を編集する手法を提案した．非圧縮性は入力の流れ場をベクトルポテンシャルに変換することで満足される．また，curl-free な成分を含まないようベクトルポテンシャルにおける自由度を拘束するための方法も提案した．

今後の課題として，より直観的な変形の実現が挙げられる．また，もう一つの流体の物理法則である運動量保存も満たした編集方法の開発が挙げられる．

文　献

[1] Bhatia, H., Norgard, G., Pascucci, V., Bremer, P.: The helmholtz-hodge decomposition - a survey. IEEE Transactions on Visualization and Computer Graphics 19(8), pp. 1386-1404 (2013)

[2] Bridson, R.: Fluid Simulation for Computer Graphics. AK Peters (2008)

[3] Bridson, R., Hourihan, J., Nordenstam, M.: Curl-noise for procedural fluid flow. ACM Transactions on Graphics 26(3), Article 46 (2007)

[4] Cottet, G.H., Koumoutsakos, P.D.: Vortex Methods: Theory and Practice. Cambridge University Press (2000)

[5] Fattal, R., Lischinski, D.: Target-driven smoke animation. ACM Transactions on Graphics 23(3), 439–446 (2004)

[6] Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In Proceedings of ACM SIGGRAPH 2001, pp. 15–22 (2001)

[7] Feldman, B.E., O'Brien, J.F., Arikan, O.: Animating suspended particle explosions. In Proceedings of ACM SIGGRAPH 2003, pp. 708–715 (2003)

[8] Foster, N., Fedkiw, R.: Practical animation of liquids. In Proceedings of ACM SIGGRAPH 2001, pp. 23–30 (2001)

[9] Fuller, A.R., Krishnan, H., Mahrous, K., Hamann, B., Joy, K.I.: Real-time procedural volumetric fire. In Proceeding of the 2007 symposium on Interactive 3D graphics and games, pp. 175-180 (2007)

[10] Gamito, M.N., Lopes, P.F., Gomes, M.R.: Two-dimensional simulation of gaseous phenomena using vortex particles. In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation, pp. 3–15. Springer (1995)

[11] Hong, W., House, D.H., Keyser, J.: Adaptive particles for incompressible fluid simulation. The Visual Computer 24(7–9), 535–543 (2008)

[12] Jakob, W.: Mitsuba renderer (2010). Http://www.mitsuba-renderer.org

[13] Kim, T., Delaney, J.: Subspace fluid re-simulation. ACM Transactions on Graphics 32(4), Article 62 (2013)

[14] Kim, Y., Machiraju, R., Thompson, D.: Path-based control of smoke simulations. In Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 33–42 (2006)

[15] Lamorlette, A., Foster, N.: Structural modeling of flames for a production environment. ACM Transactions on Graphics 21(3), 729–735 (2002)

[16] Nguyen, D.Q., Fedkiw, R., Jensen, H.W.: Physically based modeling and animation of fire. ACM Transactions on Graphics 21(3), 721–728 (2002)

[17] Park, S.I., Kim, M.J.: Vortex fluid for gaseous phenomena. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 261–270 (2005)

[18] Pighin, F., Cohen, J., Shah, M.: Modeling and editing flows using advected radial basis functions. In Proceedings of the 2004 ACM SIGGRAPH /Eurographics Symposium on Computer Animation, pp. 223–232 (2004)

[19] Raveendran, K., Wojtan, C., Thuerey, N., Turk, G.: Blending liquids. ACM Transactions on Graphics 33(4), Article 137 (2014)

[20] Sato, S., Dobashi, Y., Iwasaki, K., Yamamoto, T., Nishita, T.: Deformation of 2D flow fields using stream functions. In Proceedings of SIGGRAPH Asia 2014 Technical Briefs, Article 4 (2014)

[21] Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Transactions on Graphics 25(3), pp. 533–540 (2006)

[22] Shi, L., Yu, Y.: Taming liquids for rapidly changing targets. In Proceedings of the 2005 ACM SIGGRAPH /Eurographics symposium on Computer animation, pp. 229–236 (2005)

[23] Stam, J.: Stable fluids. In Proceedings of ACM SIGGRAPH 1999, Annual Conference Series, pp. 121–128(1999)

[24] Tong, Y., Lombeyda, S., Hirani, A.N., Desbrun, M.: Discrete multiscale vector field decomposition. ACM Transactions on Graphics 22(3), 445–452 (2003)

[25] Treuille, A., Lewis, A., Popovic, Z.: Model reduction for real-time fluids. ACM Transactions on Graphics

25(3), 826–834 (2006)

[26] Treuille, A., McNamara, A., Popovic, Z., Stam, J.: Keyframe control of smoke simulations. ACM Transactions on Graphics 22(3), 716–723 (2003)

[27] Wicke, M., Stanton, M., Treuille, A.: Modular bases for fluid dynamics. ACM Transactions on Graphics 28(3), Article 39 (2009)

[28] Yaeger, L., Upson, C., Myers, R.: Combining physical and visual simulation-creation of the planet Jupiter for the film “2010”. In Proceedings of ACM SIGGRAPH'86, pp. 85–93 (1986)

[29] Zhang, X., Bridson, R.: A PPPM fast summation method for fluids and beyond. ACM Transactions on Graphics 33(6), 206:1–206:11 (2014).

A 式(7)と境界条件より$\nabla\cdot\mathbf{\Psi}=0$となることの証明

式(7)から，我々は$\nabla\cdot(-\nabla^2\mathbf{\Psi})=\nabla\cdot\nabla\times\mathbf{v}=0$を得る．$\mathbf{\Psi}$は十分になめらかであるため，$\nabla\cdot(-\nabla^2\mathbf{\Psi})=-\nabla^2(\nabla\cdot\mathbf{\Psi})$となる．従って$\nabla^2(\nabla\cdot\mathbf{\Psi})=0$となる．$G=\nabla\cdot\mathbf{\Psi}$と書くこととすると, 我々は$\nabla^2 G=0$を得る. 境界において$\nabla\cdot\mathbf{\Psi}=0$なので, Gもまた境界において$G=0$となる. そのため, 領域全体について$G=0$となり，結果領域全体について$\nabla\cdot\mathbf{\Psi}=0$を得る.