基本速度場を用いた高解像度な流体映像の生成

-A Data-driven Approach for Synthesizing High-resolution Fluid Animation-

森田拓也　佐藤周平　土橋宜典　山本強

Takuya MORITA, Syuhei SATO, Yoshinori DOBASHI and Tsuyoshi YAMAMOTO

北海道大学大学院情報科学研究科

Graduate School of Information Science and Technology, Hokkaido University

E-mail: {morita,sato,doba,yamamoto}@ime.ist.hokudai.ac.jp

1　はじめに

コンピュータグラフィックス (CG) がもつ表現力は著しく成長を遂げており，様々な対象物の描写においてその写実性が向上している．これらの技術進歩は，映画やゲームをはじめとするビジュアルコンテンツにおいて確認することができる．流体の CG 表現に関しても，煙や水，炎などを対象として様々な手法が研究されている [1]．なかでも近年では，流体力学に則った物理シミュレーションを用いた表現手法を利用することで非常に写実的な映像の生成が可能となり，様々な映像制作アプリケーション等においてもこの技術が活用されている．CG による流体映像は屋内外を問わず様々なシーンにおいて活躍し，その高い写実性や動きの複雑さなどから映像に大きなインパクトを与えている．

しかし，流体映像の生成には未だ非常に膨大な計算コストや煩雑なパラメータ設定を要し，映像制作に活用する際の実用性を大きく低下させている．流体映像の制作者は所望のシーンを描くための試行錯誤を繰り返す必要があるため，従来手法では作業に大きな遅延をもたらしてしまう．

そこで本研究では，高品質な流体映像を低コストかつ高速に生成する手法を提案する．提案手法では，低解像度の 3 次元流体シミュレーションの結果を元に，データベース化された速度場を合成することで高解像度の 3 次元流体映像を高速に生成する．提案手法は流体シミュレーションにより算出された速度場のみを元に高解像度化を図っているため，様々な種類の流体に対して適用することができる．本稿では，炎，煙，雲を対象として実験を行う．

2　関連研究

1999 年，Stam は Navie-Stokes 方程式を解く際における安定性の問題を解決し，流体シミュレーションを実用化することに成功した [2]．これをきっかけとして，様々な流体を対象としたビジュアルシミュレーションに関する研究が始まった．流体シミュレーションの最たる問題点として，その計算コストの高さが挙げられる．CG の研究者は様々なアプローチによってこの問題の解決に取り組んでいる．Horvath らの手法 [3] では炎の映像生成を対象としており，3 次元空間を複数の 2 次元スライスに分割し，各スライス上でシミュレーションを行うことで効率化を図っている．この手法ではメモリ利用効率の向上には成功しているが，複数回の 2 次元シミュレーションを必要とするため，処理時間の問題は根本的には解決されていない．

本研究と同様に，低解像度のシミュレーション結果を元に高解像度の映像を生成する研究も存在する．Yuan らの手法 [4] では，低解像度シミュレーションから大局的な流れ場を抽出し，それに従うように高解像度シミュレーションの流れ場を拘束している．この手法では低解像度でのデザインに従う映像を生成することは可能であるが，最終結果の出力には非常に高い計算コストを要する．Kim らの手法 [5] では，乱流ノイズ関数と低解像度シミュレーションを組み合わせることで細部の情報を付加した高解像度映像を出力しているが，実際の高解像度シミュレーションと比べてノイズ感の強い映像が生成されてしまう．

これらの研究では，ノイズ付加により疑似的に複雑さを付加するか，もしくは高解像度シミュレーションを部分的に効率よく行うことで高解像度化を図っているが，本研究ではそれらの間に位置する手法を提案している．提案手法では，まず高解像度の 2 次元シミュレーションにより速度場のデータベースを作成し，それを低解像度シミュレーションに対して手続き的に使用することで高解像度化を図る．Treuille ら [6] や Wicke[7] らもデータベースを活用した流体シミュレーションの高速化を提案している．しかし，提案手法はシミュレーション自体を高速化するものではなく，低解像度シミュレーションを高解像度な結果に変換するという点で上記の手法とは異なる．さらに，上記の手法ではデータベース構築に 20〜30 時間を要するが，提案手法では 2 次元シミュレーションによるデータベース構築を行っているため，前処理は 1 時間以内で終えることができる．

3　流体シミュレーション

本研究で対象とする流体は煙，雲，炎の 3 種類であるが，いずれにおいても格子法によるシミュレーションを行い，基本的な大気流体モデルは Stam の手法 [2] に従う．格子

法では，シミュレーション空間を格子で分割し，各格子点に格納された情報を時々刻々と更新することで流体解析を行う．本稿では，簡略化のため流体の密度を一定とし，また非圧縮，非粘性を仮定した次に示す Navie-Stokes 方程式を解く．

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \bigtriangledown)\mathbf{u} - \frac{1}{\rho} \bigtriangledown p + \mathbf{f} \tag{1}$$

$$\bigtriangledown \cdot \mathbf{u} = 0 \tag{2}$$

ここで，$\mathbf{u}$ は流体の速度ベクトル，ρ は流体の密度，p は圧力である．$\mathbf{f}$ は外力全般を表しており，重力や浮力，風などの様々な影響を考慮している．式 (1) は流体の時間発展を表す方程式であり，右辺第 1 項から移流項，圧力項，外力項を指す．この方程式は，非圧縮性を仮定した場合のみ式 (2) に示した連続の式が成立する．

上記の方程式を高速に処理するため，Crane らが考案した GPU 処理モデル [8] を用いる．さらに，乱流成分を疑似的に付加するため Fedkiw らの手法 [9] をもとに式 (3) に示す渦補正力を外力項に加えている．

$$\mathbf{f}_{conf} = \epsilon(\mathbf{N} \times \boldsymbol{\omega}) \tag{3}$$

ここで，ϵ は渦補正係数を表し，$\mathbf{N}$ は $\boldsymbol{\omega} = \bigtriangledown \times \mathbf{u}$ を用いて表現すると $\mathbf{N} = \bigtriangledown|\boldsymbol{\omega}|/|\bigtriangledown|\boldsymbol{\omega}||$ となる．

それぞれの流体に対するシミュレーション手法について説明する．煙のシミュレーションに関しては，上記のシミュレーションで算出された速度場で煙密度のみを移流させる．炎のシミュレーションに関しては文献 [8] の手法に基づき，冷却効果を考慮した温度場の移流に加え，温度影響による浮力の発生を外力項に組み込んでいる．雲のシミュレーションに関しては土橋らの手法 [10] に基づき，相転移現象を考慮した上で空気中の蒸気密度と雲密度，および温度場を移流させる．さらに，温度影響による浮力，および重力影響による水滴落下を外力項に組み込んでいる．より詳しい説明に関しては本稿では割愛する．詳細についてはこれまでに提示した参考文献を参照していただきたい．

4 提案手法

提案手法の考え方を図 1 に示す．本手法は大きく分けて前処理とランタイム処理の 2 段階で構成される．前処理では，高解像度の 2 次元シミュレーションから速度場のデータベースを構築する．このとき，シミュレーションにより算出される速度場のデータ量は膨大であるため，すべてをデータベース化するのは非効率的である．提案手法では，保存した全速度場データに対して主成分分析 (Principal Component Analysis:PCA) を適用することでデータ圧縮を図る．以降，PCA を適用した速度場データを基本速度場と呼ぶ．

ランタイム処理では，前処理で作成したデータベースの合成処理によって，高解像度の 3 次元速度場を生成する．このとき生成される速度場は，入力となる低解像度の 3 次元シミュレーション速度場を近似したものである．最後に，生成した速度場に従って流体の温度場や密度場を移流させ，結果をレンダリングすることで映像化する．区別を図るため，本稿では 2 次元シミュレーションを uv 空間，3 次元シミュレーションを xyz 空間で表現している．以降，提案手法の各処理について詳しく述べる．

4.1 データベースの構築

提案手法では，高解像度化に用いるデータベースを前処理によって作成する．データベース構築には 2 次元の流体シミュレーションを利用する．2 次元シミュレーションでは，まず各ステップにおいて生成される速度場データを一時的に保持する．シミュレーションのステップ数が N_{all}，格子数が N_d のとき，すべての速度場データを次のような行列 X で表現できる．

$$X = \begin{pmatrix} u_{0,0} & u_{1,0} & \cdots & u_{N_{all}-1,0} \\ v_{0,0} & v_{1,0} & \cdots & v_{N_{all}-1,0} \\ u_{0,1} & u_{1,1} & \cdots & u_{N_{all}-1,1} \\ v_{0,1} & v_{1,1} & \cdots & v_{N_{all}-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{0,N_d} & u_{1,N_d} & \cdots & u_{N_{all}-1,N_d} \\ v_{0,N_d} & v_{1,N_d} & \cdots & v_{N_{all}-1,N_d} \end{pmatrix} \tag{4}$$

ここで，$u_{n,i}$，$v_{n,i}$ はシミュレーションの n ステップ目における各格子点 i の速度ベクトルがもつ u 成分と v 成分を表す．この行列をすべて保存すると，データ容量が膨大になってしまう．ランタイム処理ではデータベースを常にメモリ上で確保しておく必要があるため，このままの状態で扱うのは非常に困難である．これを回避するために，PCA によるデータ圧縮処理を行う．この処理では，保持した全速度場データを格子数 $N_b = n_b \times n_b$ の正方形ブロックに分割する．そして全ブロックの速度場に対して PCA を適用することで次式に示す基本速度場の行列 U を得る．

$$U = \begin{pmatrix} u_{0,0} & u_{1,0} & \cdots & u_{m-1,0} \\ v_{0,0} & v_{1,0} & \cdots & v_{m-1,0} \\ u_{0,1} & u_{1,1} & \cdots & u_{m-1,1} \\ v_{0,1} & v_{1,1} & \cdots & v_{m-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{0,N_d} & u_{1,N_d} & \cdots & u_{m-1,N_d} \\ v_{0,N_d} & v_{1,N_d} & \cdots & v_{m-1,N_d} \end{pmatrix} \tag{5}$$

ここで，m はユーザが指定した基本速度場の数であり，PCA によって算出した固有ベクトルの数である．式 (5) の各列がそれぞれ 1 つの基本速度場を示しており，1 列目から第

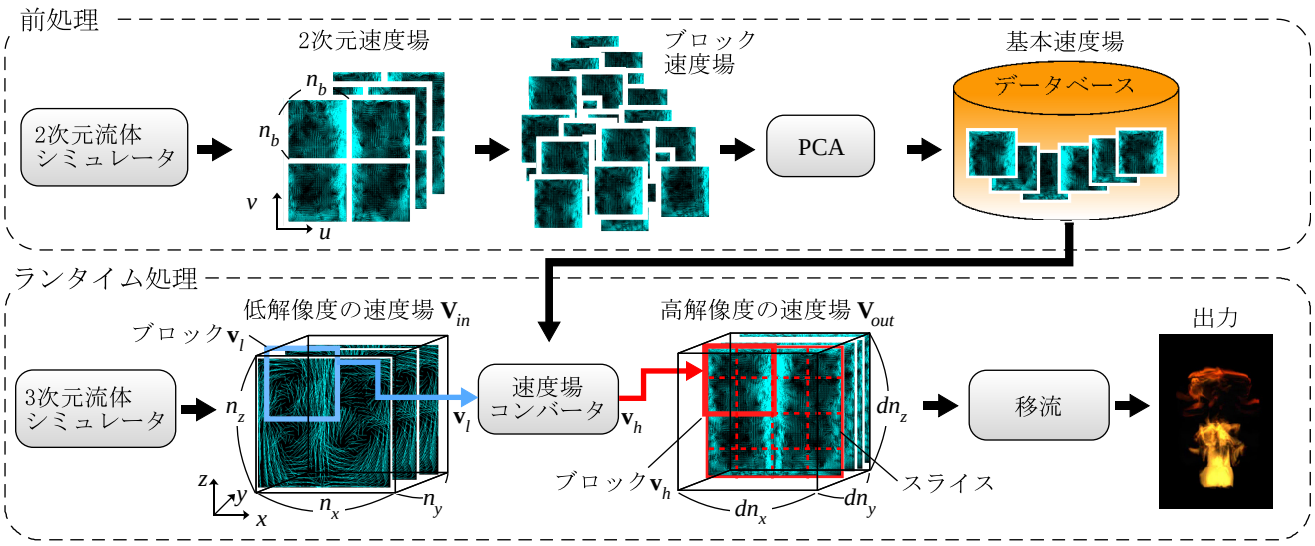

図 1: 提案手法の考え方

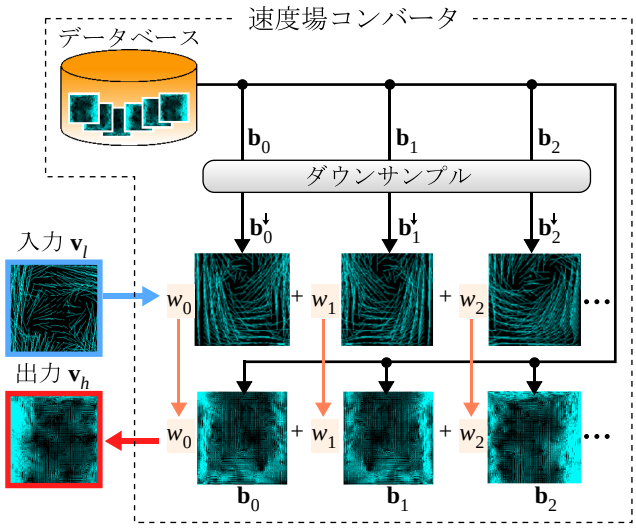

図 2: 変換処理の流れ

1 主成分，第 2 主成分，...，第 m 主成分となる．基本速度場は元の速度場に対して m/N_{all} までデータ数を圧縮することができる．

前処理で行う 2 次元シミュレーションのパラメータやソース配置は，ランタイム処理における 3 次元シミュレーションに使用するものと似た値であることが望ましい．たとえば，地面から立ち昇る煙の 3 次元シミュレーションを対象とした場合，前処理の 2 次元シミュレーションにおいても同じように煙が立ち昇る条件で行う．このとき，2 次元シミュレーションを最終的に出力したい解像度で行うことで，その後の変換処理においてより詳細な乱流成分を付加することができる．提案手法ではデータベースに含まれる以上に複雑な乱流成分を付加することはできないが，データベース構築時に障害物等の影響を考慮する必要はない．

4.2 速度場の高解像度化

ランタイム処理における速度場の高解像度化について説明する．まず，ユーザは求める解像度よりも低い解像度で 3 次元シミュレーションを行い，渦補正係数 ϵ などのパラメータを自由に設定することで，所望する複雑さをもつ流体をデザインする．シミュレーションの各タイムステップでは，生成された 3 次元速度場に対して基本速度場のデータベースを用いた変換処理を施し，高解像度化を図る．ここで，入力となる低解像度の 3 次元速度場を $\mathbf{V}_{in}(n_x, n_y, n_z)$ と表す．これに対して，d 倍の解像度をもつ 3 次元速度場 $\mathbf{V}_{out}(dn_x, dn_y, dn_z)$ を生成することが提案手法の目的である．提案手法では，y 軸方向に対して 2 次元データが均等に並べられているものとして $\mathbf{V}_{in}$ を捉え，それぞれのスライスが独立で処理される構造となっている．

図 1 の $\mathbf{V}_{out}$ 中にある大きな赤枠で囲った 1 枚のスライスをもとに，高解像度化の流れを説明する．まず，1 枚のスライスを小さな赤枠で囲ったブロックでさらに分割する．それぞれのブロック間の不連続性を軽減するために，それぞれのブロックはお互いが重なり合うように配置する．本稿では，それぞれ隣接するブロックが半分ずつ重なるように配置し，ブロックの重複部分に関しては線形補間を適用している．赤枠のブロック速度場 $\mathbf{v}_h$ は，それに対応する青枠の低解像度ブロック速度場 $\mathbf{v}_l$ からの変換処理により生成される．

変換処理の流れを図 2 に示す．この処理ではまず基本速度場に対してダウンサンプリング処理を行い，入力となる低解像度速度場と格子数を合わせる．そして，ダウンサンプリングした基本速度場をそれぞれ重み付けして合成することによって入力速度場 $\mathbf{v}_l$ を近似する．このときの重みの算出方法については後に詳しく述べる．最後に，近似処理

で算出した重みを用いて元の基本速度場を合成し，高解像度の速度場 $\mathbf{v}_h$ を出力する．しかし，基本速度場が 2 次元であることに対し入力速度場は 3 次元であるため，各ベクトル要素に対して 1 対 1 の対応付けができない．したがって提案手法では，入力速度場の水平方向成分 (xy 成分) には基本速度場の水平方向成分 (u 成分)，入力速度場の垂直方向成分 (z 成分) には基本速度場の垂直方向成分 (v 成分) を対応させることで近似処理を行っている．

次に，変換処理における重みの算出方法について説明する．ここでは，uv 成分をもつ基本速度場を $\mathbf{b}_{i,*}(* = u, v)$ で表し，同様に xyz 成分をもつ入力速度場を $\mathbf{v}_{l,\star}(\star = x, y, z)$ で表す．入力速度場を近似するための各基本速度場の重み $w_{i,\star}(\star = x, y, z)$ は次式の最小化問題を解くことで求めることができる．

$$\min_{w_{i,x}} ||\mathbf{v}_{l,x} - \sum_{i=0}^{m-1} w_{i,x}(\mathbf{b}^{\downarrow}_{i,u})||^2 \tag{6}$$

$$\min_{w_{i,y}} ||\mathbf{v}_{l,y} - \sum_{i=0}^{m-1} w_{i,y}(\mathbf{b}^{\downarrow}_{i,u})||^2 \tag{7}$$

$$\min_{w_{i,z}} ||\mathbf{v}_{l,z} - \sum_{i=0}^{m-1} w_{i,z}(\mathbf{b}^{\downarrow}_{i,v})||^2 \tag{8}$$

ここで $\mathbf{b}^{\downarrow}_{i,*}$ は $\mathbf{b}_{i,*}$ に対してダウンサンプリングを行った速度場である．式 (6) を例に挙げて上記の最小化問題における解法について説明する．まず，以下に示す行列方程式を立てる．

$$\mathbf{A}_x \mathbf{w}_x = \mathbf{c}_x \tag{9}$$

ここで，$\mathbf{A}_x$ は $m \times m$ の行列で，$\mathbf{w}_x$ と $\mathbf{c}_x$ は m 次の列ベクトルである．$\mathbf{w}_x$ は未知変数で，各要素には重み $w_{i,x}$ が入る．$\mathbf{A}_x$ の (i, j) 要素を $a_{ij,x}$，$\mathbf{c}_x$ の i 番目の要素を $c_{i,x}$ とおくと，これらの値は次式により与えられる．

$$a_{ij,x} = \mathbf{b}^{\downarrow}_{i,u} \cdot \mathbf{b}^{\downarrow}_{j,u} \tag{10}$$

$$c_{i,x} = \mathbf{v}_{l,x} \cdot \mathbf{b}^{\downarrow}_{i,u} \tag{11}$$

よって，$\mathbf{w}_x$ は $\mathbf{A}_x$ の逆行列と $\mathbf{c}_x$ との行列積によって求めることができる．最後に，次式により高解像度の 3 次元速度場 $\mathbf{v}_{h,x}$ を生成する．

$$\mathbf{v}_{h,x} = \sum_{i=0}^{m-1} \alpha_i w_{i,x} \mathbf{b}_{i,u} \tag{12}$$

α_i は非負の制御パラメータである．PCA の性質上，i が大きいほど基本速度場には細部の乱流成分が格納されている．ユーザはそれらの重みを調整することで，自由に乱流成分を付加することができる．

4.3 再帰的合成処理

これまでに説明した手法では，入力となる 3 次元シミュレーションの解像度に対して目標となる解像度の倍率が一定以上高い場合，速度場が正しく近似されず所望の結果が得られないケースがある．これは，近似処理においてダウンサンプリングされた基本速度場の値が，元の基本速度場の値と比べて大きく異なってしまうことが原因であると考えられる．この問題に対しては，再帰的に本手法を適用することにより解決が可能である．具体的には，本手法で一度高解像度化した 3 次元速度場を入力として，再び高解像度化を行うことでより高い解像度をもつ映像を生成する．このとき，各高解像度化ステップにおいて，対応した解像度のデータベースを再構築する必要がある．

5 実験結果

本稿では，炎，煙，雲に対して提案法を適用した．それぞれのシミュレーションの手順は次に従う．まず，低解像度の速度場に対して提案手法による高解像度化を行う．次に，高解像化された速度場を用いて，煙の密度，炎の温度，雲の密度を移流させる．毎ステップのソース配置に関しては，煙と炎のシミュレーションでは指定した格子点に対して密度，温度を加える．雲のシミュレーションに関しては明示的なソースが存在しないため，水蒸気から水滴への相転移現象をソースとして扱う．以降の実験では，提案法に用いる基本速度場のブロックサイズは 32×32，PCA の主成分数は 32 とする．

提案手法を炎に適用した実験結果を図 3 に示す．図 3(a) は入力となる低解像度シミュレーションの結果で，格子数は $32 \times 32 \times 64$ である．図 3(b)(c)(d) は，図 3(a) の速度場を高解像度化した結果で，解像度は 4 倍の $128 \times 128 \times 256$ である．図 3(b) は線形補間による高解像度化を行っており，図 3(c) は提案手法による高解像度化を行っている．それぞれの結果を比較すると，提案法ではより細かい乱流成分が付加されていることがわかる．図 3(d) では式 (12) の α_i の後半成分を 4 倍にしており，より複雑な炎が生成されている．このように，提案手法ではユーザが α_i を編集することで乱流成分を自由に付加することができる．図 3(e) では従来手法の Wavelet Turbulence[5] を図 3(a) のシミュレーション結果に適用している．この手法でも細かい乱流成分の付加には成功しているが，ノイズ感の強い見た目となっている．図 3(d) と図 3(e) を比較すると，提案手法の方がより写実的であると思われる． 図 4 は風や障害物の影響を考慮した実験結果である．ランタイム処理のシミュレーションにおいて，図 4(a) では x 軸方向に風力を発生させており，図 4(b) では球体の障害物を左右に移動させている．この実験を行うにあたり，あらたなデータベースの構築は

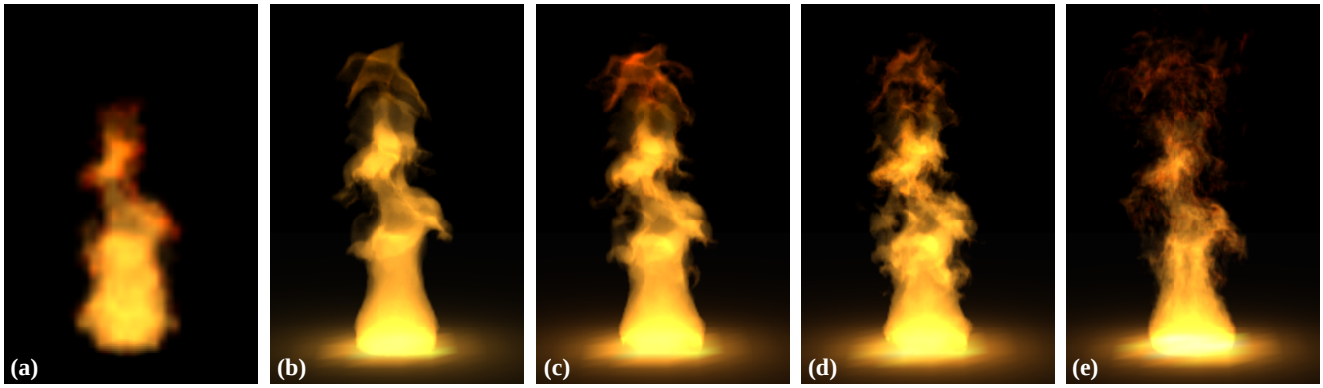

図 3: 提案手法の実験結果および従来手法との比較

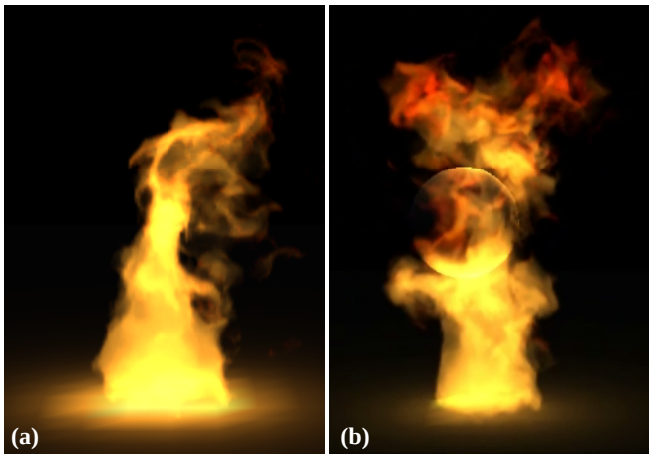

図 4: 風の影響 (a) と障害物の影響 (b) を考慮した実験結果

行っていない．結果より，風や障害物の影響があっても基本速度場による近似処理が可能であることを確認した．

図 5 は再帰的に提案手法を適用することで，より高い解像度をもつ炎を生成した結果である．ここでは，格子数が $32 \times 32 \times 64$ の低解像度シミュレーションをもとに，提案手法を 3 度繰り返すことで，格子数が $256 \times 256 \times 512$ の高解像度な炎を生成している．

図 6 はそれぞれ煙と雲の映像に対して提案法を適用した結果である．境界線より左側には低解像度の結果を表示し，右側には 4 倍の解像度をもつ結果を表示している．高解像度の煙の格子数は $256 \times 128 \times 128$，雲の格子数は $384 \times 84 \times 240$ である．いずれの結果においても，高解像度な映像が生成されていることがわかる．

表 1 ではそれぞれのシミュレーションにおける格子数，および処理時間についてまとめている．それぞれ，低解像度 3 次元シミュレーションの格子数を n_{low}，高解像度化した流体の格子数を n_{high}，前処理におけるデータベース構築の処理時間を T_{pre}，低解像度 3 次元シミュレーションの処理時間を T_{low}，速度場を高解像度化する処理時間を T_{synth}，従来手法による高解像度シミュレーションの処理

図 5: 再帰的合成処理の実験結果

時間を T_{high} で表す．実験環境に用いた CPU は Intel Core i7-2600K，GPU は NVIDIA GeForce GTX 580 である．従来手法と提案手法との処理時間を比較すると，高速化が図れていることが確認できる．なお，従来手法による雲の高解像度シミュレーションに関しては，我々の実験環境ではメモリの確保可能な容量を上回っており実験を行うことができなかったので，処理時間の記載を割愛する．

6　まとめと今後の課題

本稿では，2 次元速度場を PCA により圧縮した基本速度場のデータベースを用意し，低解像度の 3 次元流体シミュレーションの速度場をデータベースの合成処理によって近似的に再現することで，元の速度場よりも高解像度の流体映像を生成する手法を提案した．そして，提案手法が煙，炎，雲などの流体全般に対して適用可能であることを実験

表 1: 各実験におけるシミュレーションの格子数および処理時間

Example	n_{low}	n_{high}	T_{pre}	T_{low}	T_{synth}	T_{high}
炎	$32 \times 32 \times 64$	$128 \times 128 \times 256$	48.72 sec	0.05 sec	0.34 sec	0.91 sec
煙	$64 \times 32 \times 32$	$256 \times 128 \times 128$	48.68 sec	0.04 sec	0.29 sec	0.85 sec
雲	$96 \times 96 \times 60$	$384 \times 384 \times 240$	399.89 sec	0.97 sec	72.51 sec	-

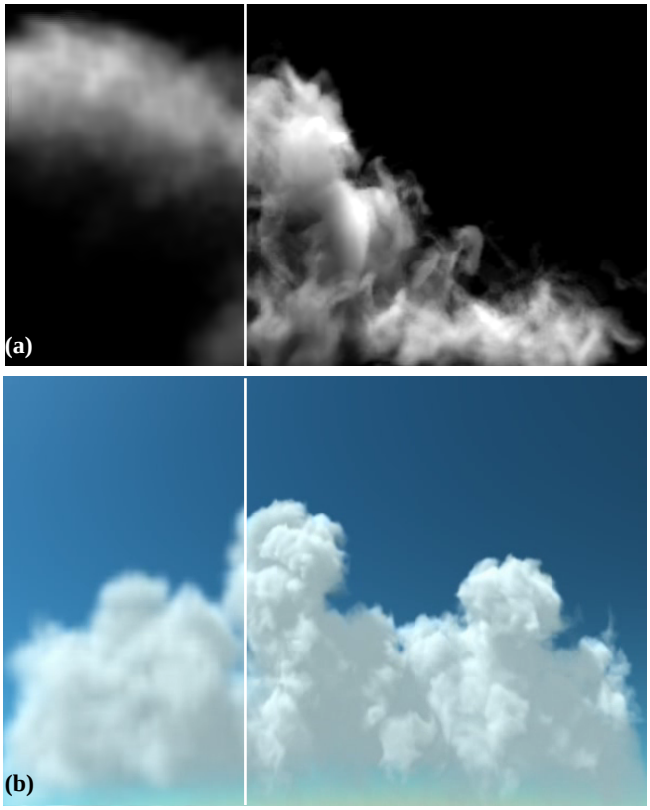

図 6: 煙 (a) と雲 (b) の実験結果

により確認し，従来手法よりも高速，かつ効率的に高解像度な映像が得られることを示した．

今後の課題としては，流体の中でも格子と粒子のハイブリッドシミュレーションにより表現されることの多い水を対象として，その速度場に対する提案手法の適用を検討することが挙げられる．

参考文献

[1] R. Bridson, Fluid Simulation for Computer Graphics, AK Peters, September 2008.

[2] J. Stam, “Stable fluids,” Proceedings of ACM SIGGRAPH 1999, 121-128, Los Angeles, USA, August 1999.

[3] C. Horvath, and W. Geiger, High-Resolution Simulation of Fire on the GPU, ACM Transaction on Graphics 28, Article 41, March 2009.

[4] Z. Yuan, F. Chen, and Y. Zhao, Pattern-Guided Smoke Animation with Lagrangian Coherent Structure, ACM Transaction on Graphics 30, Article 136, June 2011.

[5] T. Kim, N. Thürley, D. James, and M. Gross, Wavelet Turbulence for Fluid Simulation, ACM Transaction on Graphics 27, Article 50, March 2008.

[6] A. Treuille, and Z. Popovič, Model reduction for real-time fluids, ACM Transaction on Graphics 25, pp.826-834, March 2006.

[7] M. Wicke, M. Stanton, and A. Treuille, Modular bases for fluid dynamics, ACM Transaction on Graphics 28, Article 39, March 2009.

[8] K. Crane, I. Llamas, and S. Tariq, Real Time Simulation and Rendering of 3D Fluids, Addison-Wesley, ch.30, 2007.

[9] R. Fedkiw, J. Stam, and H. W. Jensen, “Visual simulation of smoke” Proc. SIGGRAPH 2001, 15-22, Los Angels, USA, August 2001.

[10] Y. Dobashi, K. Kusumoto, T. Nishita, T. Yamamoto, Feedback control of cumuliform cloud formation based on computational fluid dynamics, ACM Transaction on Graphics, Article 94, March 2008.