I-009

3 次元流体シミュレーションの速度場のループ再生

Looping of Dynamic Velocity Field generated by 3D Fluid Simulation

谷 翼 [1] 土橋 宜典 [1,2] 佐藤 周平 [1] 山本 強 [1]
TSUBASA TANI[1] YOSHINORI DOBASHI[1,2] SYUHEI SATO[1] TSUYOSHI YAMAMOTO[1]

1. はじめに

近年, 様々な自然現象が CG で表現されるようになった. 特に流体などの自然現象は，物理シミュレーションを用いることにより，写実的な映像を生成することができる．しかし，一般に生成する映像のリアリティや複雑さを追求するほど, 大きな計算コストが必要となってしまう. さらに，生成する映像の時間が長いほど，物理シミュレーションにかかる時間も長くなってしまう．そこで本稿では，流体の物理シミュレーションによって生成した，限られた時間長の速度場データを，連続的にループ再生するための手法を提案する．本手法により生成された速度場を繰り返し再利用することで，シミュレーションにより速度場を新たに計算することなく，任意時間長の流体映像を生成することが可能となる.

2. 従来研究

限られた時間長の映像から，ループ映像を作成する手法はいくつかあり，代表例として Sch¨odl らによる Video Textures[1]が挙げられる．しかし，この手法は動画像が対象であり，3 次元映像には適用していない.

また，Bhat らは，滝や川などのような，全体の形状が大きく変化しない定常的な動作をする流体の動画に対し，部分的な合成処理および編集処理を行うことにより，ループ映像を生成する手法を提案している[2]. Bhat らの手法では, 動画中の流体部分を小さな領域に分割し，その領域単位で合成および編集を行う．また，領域ごとに異なるタイミングでループさせることにより，ループ時に流体全体が一斉に変化することが無く，スムーズに動画をループさせることが可能となっている.

3. 提案手法

提案手法では，Bhat らによるループ映像生成の手法を, 3 次元の速度場へと応用することで， 3 次元流体シミュレーションから出力された限られた時間長の速度場を，スムーズにループさせる. なお, 入力データに用いる速度場は，各格子点が 3 次元の速度ベクトルを記録している，3 次元空間のデータとして表されるものとする.

3.1 出力速度場の生成方法

はじめに，入力データと同じ大きさであり，かつ同じ格子点数を持つ合成用の空間を定義する．以降，この合成用の空間を合成用空間と称し，出力速度場はこの合成用空間において生成する．次に，入力速度場において，流体の流れの中心となる線をユーザが指定する（図 1 参照）. この経路を以降は flow line と称する．そして，flow line 上に n 個の点を均等に発生させる．n はユーザにより指定する．次に，これらの点を中心とする立方体型のブロック領域 d_i（i=1, 2, …n）を定義する．さらに，合成空間においても同様にこれらのブロック領域を定義する．このとき，各ブロック領域 d_i は，入力速度場と合成用空間上で同じ座標に存在している．出力速度場の生成は，これらのブロック領域単位で行われる．具体的には，入力速度場から，各ブロック領域 d_i において取得した速度ベクトルを，合成用空間の対応する領域 d_i にそれぞれコピーする．この時，複数の領域が重なっている範囲は，コピーされる速度ベクトルの平均値を合成空間上における速度ベクトルとする．このようにして，最終的にすべての領域において速度ベクトルをコピーしたものを，出力速度場とする.

上述の方法に従って速度場の単純なループ再生を行う場合，合成用空間の各領域にコピーされる入力速度場のフレーム数は図 2 のように表される. この図は, 縦が flow line 上の領域 d_i，横が出力速度場の時間経過を表す．また表中の数字は，時間 t に取得する入力速度場のフレーム番号を表している．例えば，この表における出力速度場の 1 フレーム目では，全ての領域 d_i（i=1, 2, …n）において，入力速度場の 1 フレーム目から取得した速度ベクトルがコピーされる．すなわち，生成される速度場は入力速度場の 1 フレーム目と同等になる．単純なループ再生の場合，図 2 中の黒い太線で表されるようなループのタイミングにおいて, 各領域 d_i にコピーされる速度ベクトルの全てが同時に，1 フレーム目の速度ベクトルへと不連続にループするため, 出力速度場の不連続性が目立ってしまう.

3.2 本手法における速度場のループ

この問題を解決するために，提案手法では文献[2]の手法に基づいて，図 3 のように，各領域が互いに異なるタイミングで速度ベクトルをループさせる．具体的には，ある領

1 北海道大学
Hokkaido University
2 独立行政法人科学技術振興機構 CREST
JST CREST

域の速度ベクトルがループする，つまり前のフレームから不連続な速度場がコピーされるタイミングでは，他の領域は時間的に連続した速度場がコピーされるようにループのタイミングを階段状にずらしている．なお，ループ幅 K についてはユーザが任意で指定する．図 3 を例に説明すると，出力速度場の 1 フレーム目において，領域 d_1 には入力速度場の 1 フレーム目から速度ベクトルをコピーし，領域 d_2 以降には入力速度場の 4 フレーム目の速度ベクトルをコピーする．2 フレーム目以降についても，同様に図に従って各ブロック領域に速度ベクトルをコピーしていく．このように，flow line 上の各領域で時間をずらして，段階的に速度場をループさせることにより，速度場全体が一斉に変化することが無くなり，違和感を生じないスムーズなループが実現される．

4. 実験結果

3 次元の炎のシミュレーションにより生成された速度場を入力とし，提案法を適用することで得られた速度場に基づいて描画した炎を図 4 に示す．実験環境は，CPU: Intel Core i7-3770 CPU，メモリ: 3.5GB，グラフィックボード: NVIDIA GeForce GTX 660 である．入力速度場の解像度は 50x50x100 であり，ループ幅 K は 60 フレームに，flow line の分割数 n は 50 に設定した．

速度場に対して本手法を適用せずにループ再生を行った場合，速度場全体が同時に不連続に変化してしまい，それに基づいた炎の動きにも，流れる向きが不連続に変わるような不自然な動きが発生した．これに対し，本手法を適用して速度場のループ再生を行うと，速度場の不連続な変化が抑制され，ループの際にも炎は自然な動きを保つことが確認できた．

5. 考察

本論文では，流体シミュレーションにより得られた速度場をループ再生する際に，不連続な変化による不自然さを軽減するための合成手法を提案した．提案手法により，有限のフレーム数の速度場をスムーズにループさせ，任意時間の流体映像生成に用いることが可能となった．

今後の課題としては，flow line を分割する過程の効率化が挙げられる．本手法では隣り合ったブロック領域が 1 フレームずつタイミングをずらして順次ループしている．従って，ある格子点において重複するブロック領域の数が多いほど，つまり flow line の分割数 n を多く取るほど，それらの平均値を取ることにより，ループによる短時間での急激な変化が抑制され，より自然な速度場が生成される．しかし，図 3 に示されるような現在の手法では，flow line の分割数 n を増やした数だけ，合成に用いる入力速度場のフレーム数も多くなってしまい，それに伴い合成に必要とするメモリ容量も膨大になってしまう．また，過度に多くの

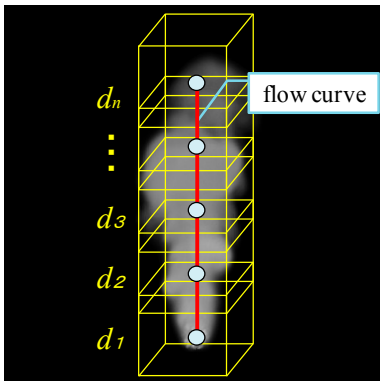

図 1　flow line の分割とブロック領域

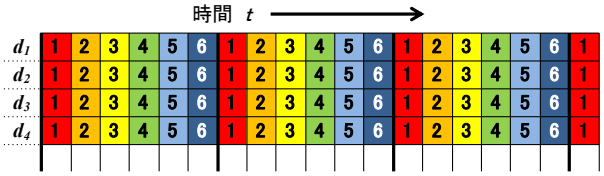

図 2 通常のループ

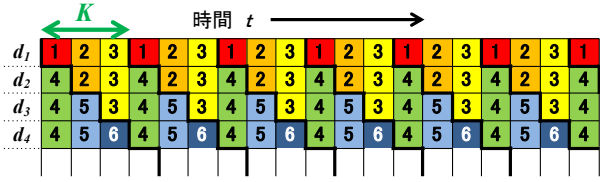

図 3 文献[2]のループ手法

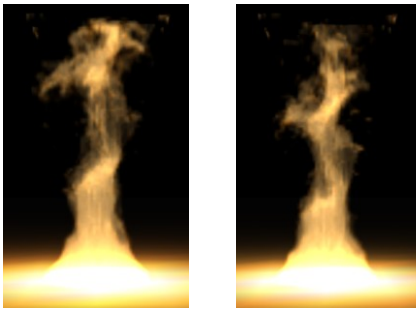

図 4 合成した速度場に従って生成した炎

ブロック領域を重ねると，入力速度場が本来持っていた速度ベクトルの詳細な変化が失われてしまう可能性も考えられる．

従って，速度場の中でもループにより発生する不自然さ，つまり急激な変化がより大きくなる部分は flow line の分割数を増やし，逆にループの際に不連続な変化があまり無い部分は分割数を減らすなど，速度場の変化を考慮した flow line の分割点の設定方法を検討したい．

参考文献

[1]Arno Schödl and Richard Szeliski and David H. Salesin and Irfan Essa. Video textures. In *Proc. SIGGRAPH '00,* pp. 489-498,2000.
[2]Kiran S. Bhat and Steven M. Seitz and Jessica K. Hodgins and Pradeep K. Khosla. Flow-based Video Synthesis and Editing. In *Proc. SIGGRAPH 04*, pp. 360-363, 2004.