I-040

3DCG 映像のための手描き風モーションブラー

Motion Blur for 3DCG Animation with Hand Drawing Style

酒井　健行†
Takeyuki Sakai

1.　はじめに

近年，日本のテレビアニメーションに於いて従来の手描き作画と併用する形で 3DCG（3 Dimensional Computer Graphics）による作画が使われるケースが増えている．この場合，セルアニメ風にレンダリングされた 3DCG 映像が使われるが，3DCG のアニメーションは写実的な効果が多く，セルアニメ風の非写実的アニメーションとは表現効果に大きな隔たりがある．

3DCG アニメーションでは素早く動く物体の動きのブレを表現する手法としてモーションブラー [1]が用いられる．一方手描きアニメーションに於いても動きのブレが描かれるが，これは写実的なモーションブラーとは異なりカートゥンブラー [2] [3] [4]と呼ばれている．

本研究では従来研究で挙げられている 3 種類のカートゥンブラー(図 1)の中でも歪みの効果について，アルゴリズムの検討を行った．歪みの効果は移動する物体の残像を，物体が尾を引くように部分的に引き伸ばすことによって表現する手法で，従来研究では物体の後方にのみ効果を発生させていたが，本研究では物体全体を歪ませることを目標とした．この歪みのカートゥンブラーについて，レンダリングのプロセスから切り離されたポストエフェクトとしてのブラー効果の適用と，従来手法とは異なる表現を生成するためのプロセスを提案する．

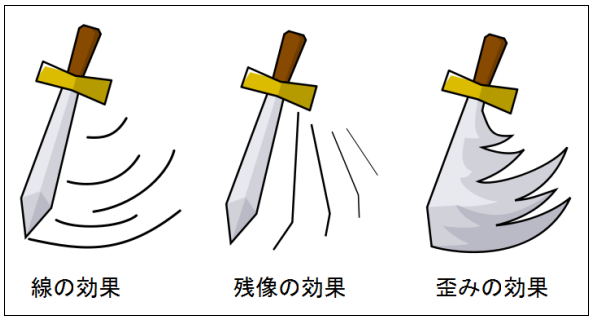

図 1:カートゥンブラーの種類

2.　提案手法

2.1.　処理の概要

提案手法の処理の概要を図 2 に示す．

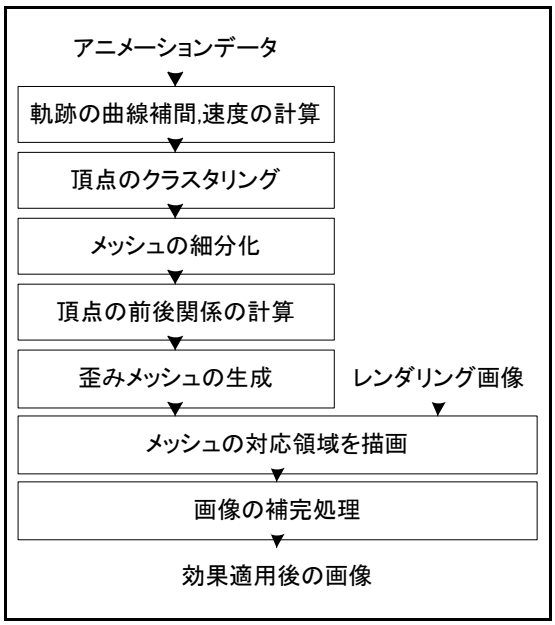

図 2:処理の流れ

まず，与えられたアニメーションデータの各頂点の軌跡を曲線で補完し，各フレームでの速度を求める．その後，頂点群のクラスタリングとメッシュの細分化を行う．移動方向に対する頂点の前後関係を求め，その情報をもとにブレを表す歪んだメッシュを生成する．生成されたメッシュと，元のメッシュ，レンダリング画像から最終的な出力画像を生成する．

2.2.　入力データ

メッシュのアニメーションのデータと，そのアニメーションをレンダリングした画像を入力データとして用いる．画像はトゥーンシェーディングなどを用いて非写実的なレンダリングがされたものを使用する．本研究の実装では，単体または複数の動く物体から構成されたシーンを想定し，背景については考慮していない．

2.3.　軌跡の曲線補間

入力されたアニメーションデータの各頂点の軌跡を非線形な曲線関数を用いて補完する．モーションブラー，カートゥンブラーの軌跡，残像を描画するという効果の特性上，軌跡が直線であれば生成される効果も直線的なものとなってしまう．そのため，曲線関数を用いて補完を行い，その関数を用いて速度を計算する．頂点間を線形に補完した場合と曲線関数で補完した場合の軌跡，速度ベクトルの違いを図 3 に示す．本論文では実装に 3 次の自然スプライン関数を用いて計算した．

†法政大学大学院情報科学研究科
Hosei University Graduate School of Computer and Information Sciences

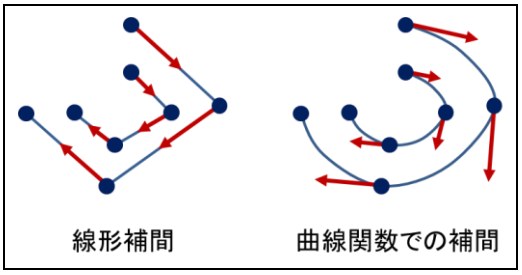

図 3:補間方法による軌跡と速度の違い

2.4. 頂点のクラスタリング

頂点と速度を用い各フレームごとに頂点のクラスタリングを行う．これは後述の前後関係の計算と歪みの生成と関連し，本手法では一つの効果の生成を運動する物体のある纏まった領域全体に対して行うため，クラスタリングによる領域わけを行わなかった場合，画面内に独立した運動を行う複数の物体があったとしてもまとめて一つの効果が生成されてしまう(図 4)．そのため，移動する物体が複数ある，または物体の中で部分的に異なる運動をする場合を考慮して，クラスタリングを行い一度に効果を生成する領域を適切に分ける必要がある．

本論文では線形判別関数を使った二分割クラスタリング [5]を用いた．各頂点のスクリーン上での 2 次元の座標値と 2 次元の速度からなる 4 次元のデータを使用し，分割の停止条件にはユーザが設定したパラメータを閾値として，クラスタ内の速度ベクトルの平均と各頂点の速度ベクトルとのなす角が，閾値以下になる場合を設定した．

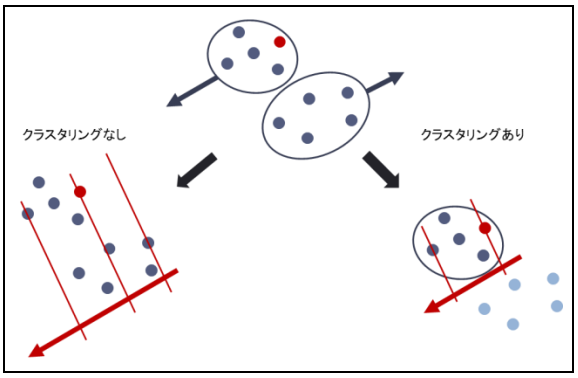

図 4:クラスタリングの有無による違い

2.5. メッシュの細分化

本手法では，メッシュを変形させ歪みを作るため，効果の細かさはメッシュの細かさに影響される．メッシュが粗い場合には，効果が直線的になり，またメッシュを歪ませた際にメッシュのねじれが起こる可能性があるため，メッシュの細分化を行い，歪みを作るために十分な頂点数を与える必要がある．本研究ではメッシュの各辺を再帰的に二等分することで分割を行うアルゴリズムを実装した．

各辺の長さが閾値以下になるまで再帰的に辺の分割と三角形の生成を繰り返すことでメッシュを細分化する．分割の停止条件となる閾値はパラメータとしてユーザが設定することとする．

メッシュ内の閾値以上の長さを持つ辺を二等分し頂点を追加する．各三角形について分割が行われた辺の数に応じて図 5 に示した分割方法でそれぞれ分割しメッシュを更新する．なおメッシュ内に閾値以上の長さの辺が存在する場合には辺の分割に戻り再帰的に分割処理を繰り返す．この手法による分割の実行例を図 6 に示す．

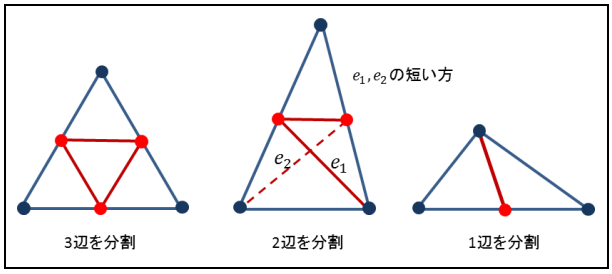

図 5:分割の方法

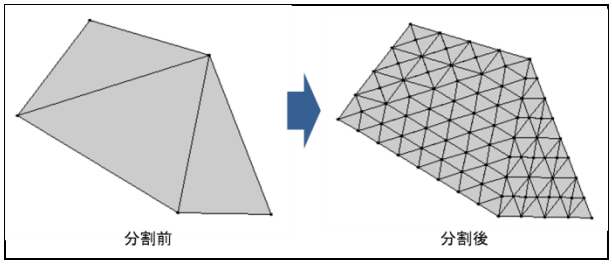

図 6:メッシュの分割例

2.6. 歪みの効果

本手法ではメッシュを移動方向に引き伸ばし，その引き伸ばしに対してブレを加えることで歪みの効果を生成している(図 7)．前後への引き伸ばしは領域の後方の頂点を前のフレームでの位置へ，前方の頂点を後ろのフレームでの位置へ動かすことによって実現している．効果の生成は各独立した運動ごとに行う必要があるため，2.4 で述べたクラスタリングの結果をもとに同一クラスタを一つの領域として扱い効果を生成する．

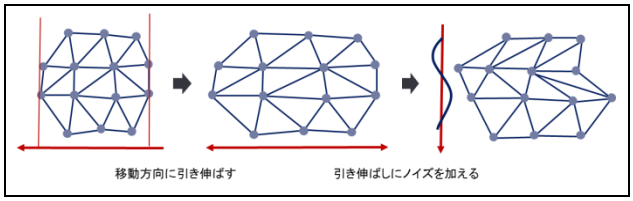

図 7:歪み生成の流れ

移動方向に引き伸ばすためには，各頂点が領域の中でほかの頂点とどのような位置関係にあるのかを調べる必要がある．引き伸ばしは移動方向前後に対して，引き伸ばしに対するブレは移動方向に対する列ごとに加えられるため，移動方向とその垂直なベクトルを基底とした座標系を考える．移動方向を軸とした座標値をk，移動方向に垂直なベクトルを軸とした座標値をlとして，領域内の点が$0 \leq k \leq 1, 0 \leq l \leq 1$となるように各点を投影する．この$k, l$の算出方法については 2.6.1 で述べる．2.6.2 で述べる歪み関数をSとすると歪み適用後の頂点の位置k'はk, lを用いて式(12)となる．

$$k' = S(k, l) \quad (1)$$

このk'をサブフレーム上の位置と考え 2.3 で述べた頂点の軌跡を補完する関数fを用いて歪み適用後の頂点$\boldsymbol{x}'$の位置を求める(式(2)，図 8).

$$\boldsymbol{x}' = f(i + k') \quad (2)$$

i: フレーム

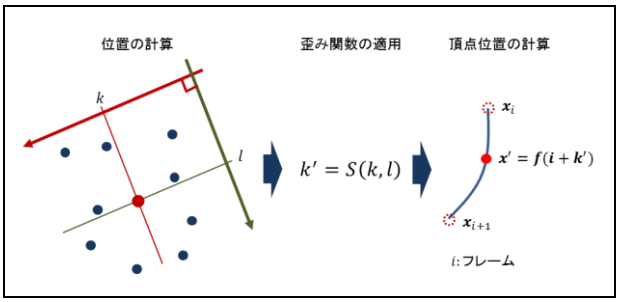

図 8:歪みメッシュの頂点位置計算

2.6.1. 頂点位置の計算

ある頂点$\boldsymbol{x}$に着目した場合，頂点の速度ベクトル$\boldsymbol{v}$を正規化した移動方向のベクトルと，頂点と領域内の他の頂点へのベクトルの内積を取り，その最大値をn，最小値をmとする(式(3)(4)). このn, mを 1~0 にマッピングすると移動方向に対して一番先頭の頂点位置を 1，後方の頂点位置を 0 としたときの頂点位置kを得ることができる(式(5)).

$$n = \max\left\{(\boldsymbol{x_j} - \boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \middle| \forall \boldsymbol{x_j} \in \boldsymbol{X}\right\} \quad (3)$$

$$m = \min\left\{(\boldsymbol{x_j} - \boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \middle| \forall \boldsymbol{x_j} \in \boldsymbol{X}\right\} \quad (4)$$

$$k = \frac{-m}{n-m} \quad (5)$$

$\boldsymbol{x}$: 対象となる頂点の位置　$\boldsymbol{v}$: 対象となる頂点の速度
$\boldsymbol{X}$: 領域内の頂点位置の集合

移動方向に対し垂直なベクトルと，頂点と領域内の他の頂点へのベクトルとの内積を用いて同様に計算すると移動方向に対し垂直な位置lを得ることができる(式(6)).

$$\boldsymbol{u} = \boldsymbol{v}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

$$n' = \max\left\{(\boldsymbol{x_j} - \boldsymbol{x}) \cdot \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \middle| \forall \boldsymbol{x_j} \in \boldsymbol{X}\right\}$$

$$m' = \min\left\{(\boldsymbol{x_j} - \boldsymbol{x}) \cdot \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \middle| \forall \boldsymbol{x_j} \in \boldsymbol{X}\right\}$$

$$l = \frac{-m'}{n'-m'} \quad (6)$$

$\boldsymbol{x}$: 対象となる頂点の位置　$\boldsymbol{v}$: 対象となる頂点の速度
$\boldsymbol{X}$: 領域内の頂点位置の集合

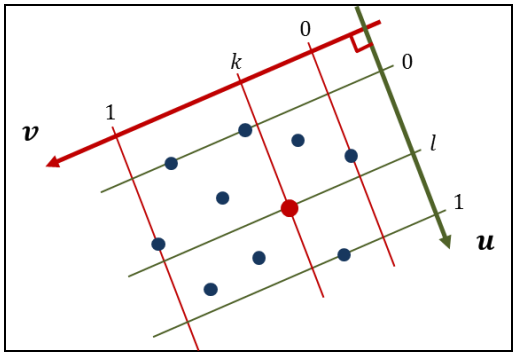

図 9:頂点位置の計算

2.6.2. 歪みメッシュの生成

メッシュを移動方向に対して前後に引き伸ばすだけでは歪みを表すことができない．歪みの効果を表現するためにはでは引き伸ばされた残像にブレを加えることが必要になる．2.6.1 で求めた移動方向に対する前後位置kと移動方向に対する列lから最終的な残像位置になるサブフレームk'を求める関数を歪み関数Sとして，この歪み関数Sはkに対してlの位置ごとにノイズを加えることによって歪んだ残像を表現する(図 10).

本研究では，一様乱数を用いた歪み関数を実装した．長さNの乱数列$r_i (i = 0,1, \dots, N-1)$を等間隔に並べ，点間を線形補間した関数(式 (7))を引き伸ばしのブレとして用いる．

$$R(x) = (1 - x + i)r_i + (x - i)r_{i+1}$$
$$(i \le x \le i + 1,\ i = 0,1, \dots, N-1) \quad (7)$$

この乱数列から作った関数をそれぞれの列に対する歪みの大きさと考え，全体の最大歪み幅sとRから歪み関数Sを式(8)とした．

$$S(k, l) = 1 - s(1 - k)R\big((N-1)l\big) \quad (8)$$

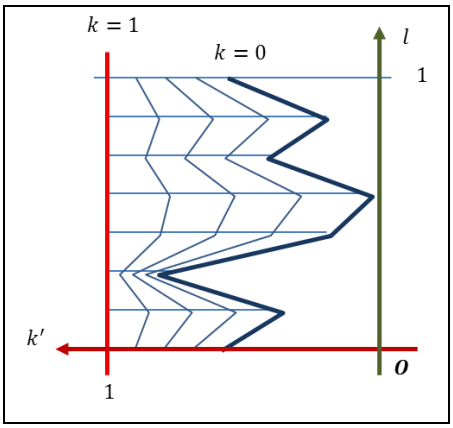

図 10:乱数を用いた歪み

本研究では乱数列の長さNと最大歪み幅sはユーザが設定するパラメータとしている．

2.7. 画像の生成

事前にレンダリングされた画像と歪みメッシュを用いて効果が適用された出力画像を得る．前後フレームのメッシュから歪みメッシュに対してマッピングを行い，生成する画像の色を前後フレームの画像から取得する．こ

の時，前後フレームでの頂点位置と歪みメッシュの頂点位置の距離に応じて色を比例配分することにより色を決定する．

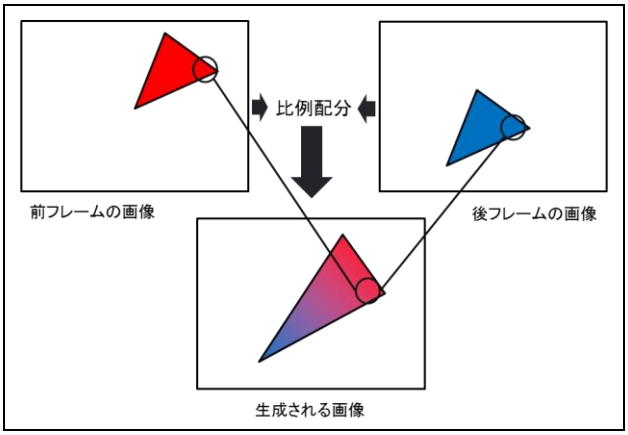

図 11:画像の生成

生成される画像のピクセルが属するポリゴンを特定する．また，そのポリゴンの 3 頂点の座標を比例配分することでピクセルの座標を表し，その配分比を求める．

対象とするピクセルの座標を$\boldsymbol{x}$，ポリゴンの 3 頂点の座標をそれぞれ$\boldsymbol{x}_{\mathrm{v0}},\boldsymbol{x}_{\mathrm{v1}},\boldsymbol{x}_{\mathrm{v2}}$と置く（$\boldsymbol{x},\boldsymbol{x}_{\mathrm{v0}},\boldsymbol{x}_{\mathrm{v1}},\boldsymbol{x}_{\mathrm{v2}}$は全て 2 次元の列ベクトルとする）と，座標の比例配分はu,vを用いて式(9)で表され，配分比は式(10)を解くことで求めることができる．この時のピクセルがポリゴンに属するための条件は式(11)となる．

$$\boldsymbol{x}=(1-u-v)\boldsymbol{x}_{\mathrm{v0}}+u\boldsymbol{x}_{\mathrm{v1}}+v\boldsymbol{x}_{\mathrm{v2}} \tag{9}$$

$$(\boldsymbol{x}_{\mathrm{v1}}-\boldsymbol{x}_{\mathrm{v0}}\quad \boldsymbol{x}_{\mathrm{v2}}-\boldsymbol{x}_{\mathrm{v0}})\begin{pmatrix}u\\v\end{pmatrix}=\boldsymbol{x}-\boldsymbol{x}_{\mathrm{v0}} \tag{10}$$

$$0\le u,0\le v,u+v\le 1 \tag{11}$$

$\boldsymbol{x}$: ピクセルの座標

$\boldsymbol{x}_{\mathrm{v0}},\boldsymbol{x}_{\mathrm{v1}},\boldsymbol{x}_{\mathrm{v2}}$: ポリゴンの頂点座標

配分比を求めることができたら，前後のフレームでの位置を計算する．式(9)と同様に前後のフレームでの位置を求め，レンダリングされた画像の対応するピクセルの色を取得する．取得した前後の色をそれぞれ$\boldsymbol{c}_{\mathrm{previous}},\boldsymbol{c}_{\mathrm{next}}$と置くと最終的なピクセルの色$\boldsymbol{c}$は式(12)の$k'$を用いて式(13)と表すことができる．

$$\boldsymbol{c}=(1-k')\boldsymbol{c}_{\mathrm{previous}}+k'\boldsymbol{c}_{\mathrm{next}} \tag{12}$$

ピクセルが属するポリゴンを特定する際に，アニメーションによっては複数のポリゴンが重なり，描画するべきポリゴンが一意に定まらない場合がある．その場合奥行値等を用いてどのポリゴンを描画するべきか判定する．またオクルージョンが発生する可能性した場合は後述する補完処理を行う必要がある．

2.8. 画像の補完計算

オブジェクト同士の位置関係によっては，前後のフレームのうちどちらかが他のオブジェクトに隠れ，適切な色が取れない場合がある．この時，取得できた片側のフレームの色をそのまま結果の色として使用すると，通常ではフレーム間での色の変化が結果に表れるのに対し，片側のフレームの色のみを使用した領域ではその変化が現れず，結果として周りの領域から分離してしまう．そのため，片側の色情報が取得できなかった場合には，周囲のピクセルの情報から隠れていて取得できなかった色を推定する必要がある．本論文では周囲のピクセルにおけるフレーム間の HSV 色情報の傾斜をもとに推定を行った．

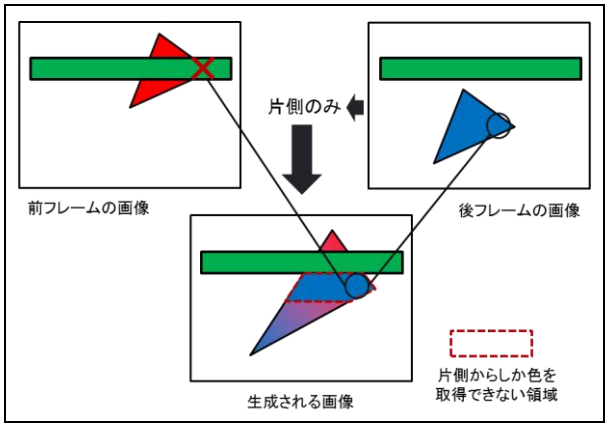

図 12:補完処理が必要なケース

前後のフレームのうち色が取得できた側の色とその周囲のピクセルでの色相，再度，明度のフレーム間の勾配から色を推定する．対象のピクセルとその周囲のピクセルとはフレーム間での色の変化の割合がおおよそ同じだと仮定して、周囲 4 ピクセルのうち色が確定しているピクセルの色の勾配の平均を対象のピクセルに適用して色を決定する．補完する対象となるピクセルの色が取得できたフレームの色情報を$\boldsymbol{c}_{\mathrm{next}}$，周囲 4 ピクセルのうち色が確定しているピクセルの数をN，色を$\boldsymbol{c}_i(1\le i\le N)$，対象ピクセルの色が取得できたフレームでの周囲のピクセルの色を$\boldsymbol{c}_{\mathrm{next},i}(1\le i\le N)$と置くと，補間後の色$\boldsymbol{c}$は式(13)となる．

$$\boldsymbol{c}=\boldsymbol{c}_{\mathrm{next}}+\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{c}_i-\boldsymbol{c}_{\mathrm{next},i}\right) \tag{13}$$

補完計算は周囲に色が確定しているピクセルが多いピクセルから行い，周囲から内側に向かって補完する．周囲に補完に使用できるピクセルがない場合には，計算を行わず色情報が取得できたフレームの色をそのまま用いる．また本研究の実装では前後のフレームどちらからも色情報が取れないピクセルについては，描画されないものとして扱っている．

3. 実装

本手法の各処理は同じ働きをする別のアルゴリズムと置き換えることができるが，効果生成の精度，表現は大きく異なる．今回，本手法を実装するために，各処理に用いたアルゴリズムをそれぞれ表 1 に示す．またどのような効果を適切とするかはアニメーションの制作者の意図によるところが大きいため，様々な効果の生成が可能となるようにパラメータを設定した．

表 1:実装に用いた手法

曲線関数	3 次自然スプライン関数
クラスタリング	BDC-LDF [5]
メッシュの細分化	各辺の長さが閾値以下になるまで再帰的に二分割
歪みの生成	一様乱数によるノイズ
画像の補完	フレーム間の HSV 傾斜

4. 実行結果

メッシュのアニメーションデータとそれをレンダリングした画像(図 13,図 15)を入力として使用した実装結果としてそれぞれ図 14，図 16 を示す．ユーザが指定するパラメータにはクラスタリングの停止条件，メッシュ細分化の閾値，歪みの大きさ，歪みの細かさを設定している．それぞれ使用した値を表 2 に示す．

表 2:使用したパラメータ

クラスタリングの停止条件となる角度	60°
メッシュ分割の停止条件となる辺の長さ	15px
歪みの大きさ	1
歪みに使う乱数列の長さ	43

物体の運動している部分が尾を引くように大きく歪んでいることがわかる．

一方で，図 16 の 2 フレーム目の顔，3 フレーム目の足の内側などピクセルの補完計算の失敗だと思われる不自然な色の部分が存在する．また，メッシュのねじれを防止するために細分化を行っているものの十分であるかは確認できていない．

5. むすび

本研究では，レンダリングされたアニメーション画像と，メッシュのアニメーションデータから，非写実的なモーションブラーの生成を行った．歪みの生成と描画については成功しているが，補完計算の精度は十分とは言えない．

本研究の処理フローでは各処理の実装に用いるアルゴリズムによって結果が異なる．特にクラスタリングと画像の補間については実装したアルゴリズムによって結果の精度が大きく左右される．また今回クラスタリングの停止条件，メッシュの細分化の閾値はユーザが入力するパラメータにより決定したが，これらのパラメータは表現に幅を持たせるためのものではなく効果生成の精度を決定するものである．したがって，これらのパラメータの決定はユーザの負担にすべきではなく，アニメーションデータに合わせパラメータが自動的に決定されることが望ましいと考えられる．今後の課題としては本手法に適した各処理のアルゴリズムの改良，精度に影響するパラメータの自動決定を挙げる．

文 献

[1] T. Akenine-Moller , E. Haines, 著: *リアルタイム レンダリング 第2版*, 加藤諒 , 秋山謙一, 共同編集, ボーンデジタル, 2006, pp. 201-203.

[2] Y. Kawagishi, K. Hatsuyama and K. Kondo, "Cartoon blur: nonphotorealistic motion blur," Computer Graphics International, 2003. Proceedings, 2003.

[3] 大林正一, 近藤邦雄 , 今間俊博, “３Ｄアニメーションのためのカトゥーンブラー,” Ｖｉｓｕａｌ Ｃｏｍｐｕｔｉｎｇ グラフィクスとＣＡＤ合同シンポジウム予稿集, 2005.

[4] S. Obayashi, K. Kondo, T. Konma , K. Iwamoto, “Non-photorealistic motion blur for 3D animation, ” SIGGRAPH '05 ACM SIGGRAPH 2005 Sketches, 2005.

[5] H. Hanaizumi, S. Chino , S. Fujimura, “A binary division algorithm using a linear discriminant function for the cluster analysis of remotely sensed multispectral images.,” Ｐｒｏｃ ＳＰＩＥ Ｉｎｔ Ｓｏｃ Ｏｐｔ Ｅｎｇ, 1995.

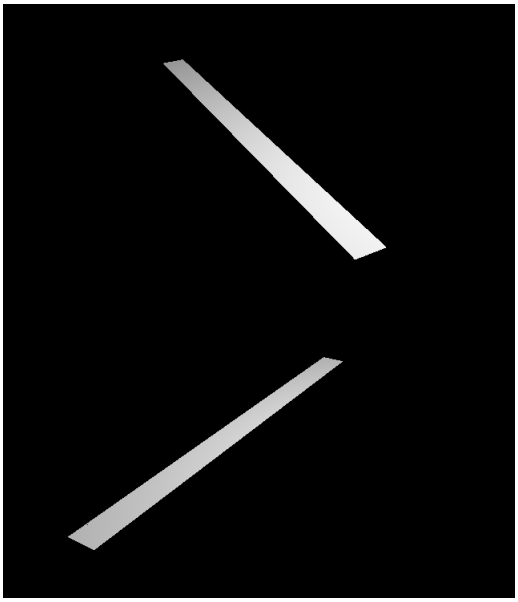

図 13 入力画像 1

図 14 出力結果 1

図 15:入力画像 2

図 16:出力結果 2