モーションデータベースを用いた実時間モーション合成手法

川地克明*　鈴木宏正*

* 東京大学

E-mail: {kawachi|suzuki}@cim.pe.u-tokyo.ac.jp

本研究では、リアルタイム CG において人間のモーションを合成する手法について述べる。モーションを合成する際には、あらかじめ用意した基本的な短いモーションを外的な拘束条件に応じて編集して接続する。本研究では Spacetime Constraints を用いたモーション編集において、与えられる拘束条件を限定することによって計算量が少なく、実時間でのモーション合成が可能な編集手法を提案する。

Realtime Motion Synthesis with Motion Database

Katsuaki Kawachi*　　Hiromasa Suzuki*

*The University of Tokyo

E-mail: {kawachi|suzuki}@cim.pe.u-tokyo.ac.jp

In this paper we describe a method to synthesize human motion in realtime CG. In creating a motion sequence we prepare basic short motions and concatenate them with transformation in order to satisfy external constraints. We introduce an algorithm for realtime motion transformation utilizing *Spacetime Constraints* with small amount of computation by restricting external constraints.

1　序論

人間のような複雑な構造をもつ物体を計算機上のキャラクタとして表現し、ユーザの要求に従って動作させることはいまだに難しい問題である。映画のようなオフラインでの CG 映像制作では、アニメータの手作業によって必要なモーションの作成が行われる。これに対し、リアルタイム CG では必要なモーションをきわめて短い時間で計算し、運動条件の変化にも柔軟に対応しなければならない。そこで、あらかじめ「歩く」「走る」等の短い動作パターンをモーションデータベースとして作成しておき、これらのモーションを高速に変形・接続するアルゴリズムによって一連のモーションを合成する手法が用いられる [1]。

こういったデータベースには運動状態の遷移が可能なモーションの組み合わせが登録されており (図 1)、状態遷移の途中では 2 つの状態の間で姿勢を表わすパラメータの補間や混合が行われる。リアルタイム CG ではモーションを高速に計算する必要があるため、キャラクタの関節角の線形補間によって状態遷移中のモーション合成を行う単純な手法が一般的に用いられている。このような線形補間によって不自然なモーションが合成される場合には、状態遷移の途中にもう一つ中間の状態を増やすことでこれを緩和することが可能だが、人間のような高い運動の自由度を持つキャラクタに対して様々に変化する外部条件にあらかじめ対応した状態を全て用意しておくことは難しい。

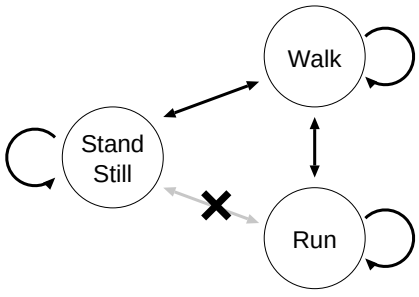

図 1: キャラクタの運動状態遷移図

一方、オフライン CG においてこのようなモーショ

ンの編集を行うための手法として、Motion Editing [2, 3, 4] という手法が提案されている。これらの手法は、元になるモーションの特徴的な動きの成分を保存したまま目的に応じて編集することが可能である。この手法を用いることで、異なる外部環境へのモーションへの変換や、体格が異なる別のキャラクタへの動作のあてはめ、運動の速度の調整といった編集作業を行うことができ、運動状態遷移図 (図 1) での関節角の線形補間を用いた手法よりも柔軟で自然なモーション合成が可能になる。本研究では、このような Motion Editing の手法をリアルタイム CG でのモーション編集に利用してデータベース上のモーションを編集することによって一連のモーションを実時間で合成する手法を提案する。

2 モーション合成手法の概要

本研究では、短い動作パターンをモーションデータベースとして用意し、これを組み合わせて接続することで一連のモーションを合成する。以下ではこのような短い時間区間で定義されるモーションを「運動セグメント」と呼ぶ。

この運動セグメントは、踵や股関節のような身体の運動を代表する特徴点 (handle)、点の間の幾何的な拘束 (skeleton)、モーションの詳細な部分 (texture) の 3 つの部分によって表現される (図 2 上段)。外部から拘束をあてはめて一連のモーション合成を行う場合には、まずはじめに特徴点に対して拘束をあてはめる (1. Transformation)。そして、特徴点の位置からキャラクタのおおまかな姿勢を再構築し (2. Reconstruction)、詳細なモーションを特徴点の間に割り当てて (3. Mapping) 拘束条件を満たすモーションを得る。

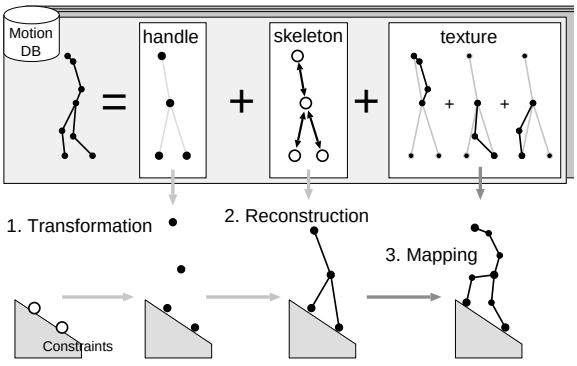

図 2: データベースを用いたモーション合成

3 運動セグメントの編集手法

オフライン CG のための Motion Editing 手法を利用することで、元になるモーションの特徴を保存したままモーションの編集を行い、拘束条件を満たす新しいモーションを得ることができる。これらの手法で用いられる Spacetime Constraints[5] によるモーションの編集では、キャラクタの位置と姿勢を重心の位置と関節角を用いてパラメータ化する手法が一般的に用いられている。このような姿勢を表すパラメータをベクトルとしてまとめ、姿勢ベクトルと呼ぶ。ある時間区間 $t_s \le t \le t_e$ でキャラクタの姿勢の時間的な変化を表す運動セグメントは、このような姿勢ベクトルを与える時間の関数 $\boldsymbol{x}(t)$ $(t_s \le t \le t_e)$ として定義できる。

Spacetime Constraints [5] を用いた手法では、既存の運動セグメント $\boldsymbol{x}(t)$ に対して差分関数 $\boldsymbol{d}(t)$ を加えることで、拘束条件を満たす新しい運動セグメント $\boldsymbol{y}(t)$ を定義する:

$$\boldsymbol{y}(t) = \boldsymbol{x}(t) + \boldsymbol{d}(t) \tag{1}$$

一般に、与えられた拘束条件を満たす差分関数 $\boldsymbol{d}(t)$ を一意に定めることはできない。そこで、Spacetime Constraints による手法では、差分 $\boldsymbol{d}(t)$ を評価する関数 $G(\boldsymbol{d}(t))$ を最小化するような $\boldsymbol{d}(t)$ を解として選ぶというヒューリスティクスを用いる。評価関数 G としては、姿勢の差分 $\boldsymbol{d}(t)$ を実現するために必要な運動エネルギを表す関数などが用いられる。このような評価関数を用いることで差分関数 $\boldsymbol{d}(t)$ はなだらかな曲線を描く関数となり、元になる運動セグメントに含まれる詳細な特徴 (高周波成分) をよく保存する編集操作が期待できる [2]。

このような Spacetime Constraints による編集操作は、全身の関節角をパラメータに含む複雑な評価関数を最小化する問題となり、計算量が多い。また、運動セグメントの時間区間の先頭の時点でセグメント全体に渡る差分関数 $\boldsymbol{d}$ を計算する必要があるので、この時点に計算が集中するという問題がある。したがって、この手法を本研究の目的である実時間モーション合成にそのまま利用することは難しいと考えられる。

そこで、本研究ではキャラクタの姿勢を表わすパラメータとして関節角ではなく、身体の運動を代表する特徴点の位置そのものを用いることで編集操作に要する差分関数の計算を単純化する。また、特徴点に対す

る拘束条件はまず個別の点に関して独立に満足させ、その後全身の特徴点間の幾何的な矛盾を解消してキャラクタの姿勢を再構築する。このように外的な拘束条件と内的な幾何拘束条件とを別々に扱うことにより、全身の運動セグメントの編集の問題を 1 つ 1 つの運動セグメントへの編集手続きに分解し、編集に要する計算量を削減する。さらに、運動セグメントに与えられる外的な拘束条件の種類を制限することで、差分関数 $\boldsymbol{d}$ の計算の大部分を実際のモーション合成の前処理とし、モーションの編集を時間区間の各点で漸進的に行うことを可能にする。

3.1 運動セグメントに対する拘束条件

一般的な Spacetime Constraints の手法では、運動セグメント中の任意の時点で位置や速度の拘束を与えることが可能であるが、本研究では、運動セグメントに対する拘束条件はセグメントの終端点のみで与えるという制約を与え、セグメントの編集に用いる差分関数 $\boldsymbol{d}$ を拘束条件のパラメータに依存しない形で前処理として計算することを可能とする。図 3 は歩行中の足先の位置を表わす特徴点 (図中黒丸 ●) の軌跡を表わしている。本研究の手法が与える拘束は運動セグメントの終端点 (図中白丸 ○) での位置と速度とに限るので、図に示すような足の先端の着地点を持ち上げるような拘束条件は与えることができるが、運動セグメントの途中で足を持ち上げる高さを変えるといった拘束は与えられない。このような拘束を与える場合には、高く足を持ち上げる別の運動セグメントか、持ち上げる点で改めてセグメントを切って拘束を与えられるようにしてある運動セグメントをデータベース中に用意しておく必要がある。

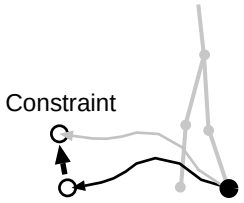

図 3: 運動セグメントに与える拘束条件

3.2 位置の拘束

本手法では、運動セグメントは特徴点の位置の離散的な時系列データ $\mathbf{M}^x = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ によって表す。また、時間間隔は一定であるものとする。このような運動セグメントに対して、時系列の最初の点の位置 $\boldsymbol{x}_0$ を保存したまま時系列の最後の点の位置 $\boldsymbol{x}_n$ を次の運動セグメント $\mathbf{M}^y$ の最初の点 $\boldsymbol{y}_0$ に一致させるという位置拘束を満たす問題 (図 4) を考える。このとき、運動セグメント $\mathbf{M}^x$ の各点に対して、$\mathbf{d} = \{\boldsymbol{d}_0, \boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ なる差分を加えることで位置拘束条件 $\boldsymbol{y}_0 = \boldsymbol{x}_n + \boldsymbol{d}_n$ を満たすものとする。

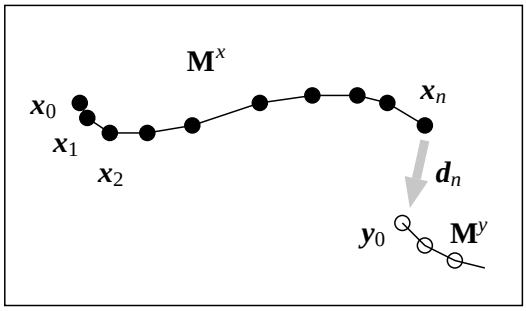

図 4: 運動セグメント終端点の位置の拘束

位置拘束条件からは、差分 $\mathbf{d}$ に関して

$$\begin{cases} \boldsymbol{d}_0 = 0 \\ \boldsymbol{d}_n = \boldsymbol{y}_0 - \boldsymbol{x}_n \end{cases} \tag{2}$$

という条件が得られる。これを Lamouret [1] と同様にして $\mathbf{d}$ の各要素に関して単純に線形補間を行うと、差分は $\boldsymbol{d}_i = (i/n)(\boldsymbol{y}_0 - \boldsymbol{x}_n) \quad (i = 0, 1, 2, \ldots, n)$ となる。このような線形補間を行った場合、元の運動セグメントの特徴とは無関係に $\boldsymbol{d}$ が計算され、もともと速度を持たない点に対して定常的な速度が与えられるといった不自然な編集結果が得られる場合がある。このような運動の特徴の喪失を防ぐために、特徴点の速度を考慮した差分 $\boldsymbol{d}_n$ の分配手法を考える。

ここで、運動セグメント $\mathbf{M}^x$ の各点 $\boldsymbol{x}_i$ に対して、点の持つ速度 $\boldsymbol{v}_i$ を

$$\boldsymbol{v}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \tag{3}$$

によって定義する。各点の位置 $\boldsymbol{x}_i$ に差分 $\boldsymbol{d}_i$ を加え、位置を $\boldsymbol{x}'_i$ に修正した場合の速度 $\boldsymbol{v}'_i$ は、

$$\begin{aligned} \boldsymbol{v}'_i &= \boldsymbol{x}'_{i+1} - \boldsymbol{x}'_i \\ &= \boldsymbol{v}_i + (\boldsymbol{d}_{i+1} - \boldsymbol{d}_i) \\ &= \boldsymbol{v}_i + \Delta\boldsymbol{v}_i \end{aligned} \tag{4}$$

となる。ここで、$\Delta\boldsymbol{v}_i$ は差分 $\boldsymbol{d}_i$ による位置の修正によって与えられた速度の差分を表す。特徴点の元の速

度の大きさ $|\boldsymbol{v}_i|$ が小さい点では、この速度を特徴としてなるべく保存したいので、以下の式

$$\Delta \boldsymbol{v}_i = |\boldsymbol{v}_i| \boldsymbol{c} \tag{5}$$

で表されるように、位置修正による速度の差分が速度の大きさに比例するように $\boldsymbol{d}_i$ を計算する。ただし、 $\boldsymbol{c}$ は定ベクトルである。

このとき、式 (4) (5) と拘束条件 $\boldsymbol{x}'_0 = \boldsymbol{x}_0$ から、

$$\begin{aligned} \boldsymbol{y}_0 = \boldsymbol{x}'_n &= \boldsymbol{x}_0 + \sum_{i=0}^{n-1} \boldsymbol{v}_i + \sum_{i=0}^{n-1} \Delta \boldsymbol{v}_i \\ &= \boldsymbol{x}_n + \left(\sum_{i=0}^{n-1} |\boldsymbol{v}_i| \right) \boldsymbol{c} \end{aligned} \tag{6}$$

また、拘束条件から $\boldsymbol{x}'_n - \boldsymbol{x}_n = \boldsymbol{d}_n$ なので、結局

$$\boldsymbol{c} = \left(\sum_{i=0}^{n-1} |\boldsymbol{v}_i| \right)^{-1} \boldsymbol{d}_n \tag{7}$$

となる。したがって、各点 i ($i = 1, 2, \ldots, n-1$) での位置の差分 $\boldsymbol{d}_i$ は、 $\boldsymbol{d}_0 = 0$ から

$$\begin{aligned} \boldsymbol{d}_i &= \boldsymbol{x}'_i - \boldsymbol{x}_i \\ &= \boldsymbol{d}_{i-1} + \Delta \boldsymbol{v}_{i-1} \\ &= \boldsymbol{d}_0 + \sum_{j=0}^{i-1} \Delta \boldsymbol{v}_j \qquad (8) \\ &= \frac{\sum_{j=0}^{i-1} |\boldsymbol{v}_j|}{\sum_{j=0}^{n-1} |\boldsymbol{v}_j|} \boldsymbol{d}_n \qquad (9) \end{aligned}$$

となる。このように、運動セグメント $\mathbf{M}^x$ の各点に対して、あらかじめ差分係数ベクトル

$$\{\delta_i\} = \left\{ \frac{\sum_{k=0}^{i-1} |\boldsymbol{v}_k|}{\sum_{j=0}^{n-1} |\boldsymbol{v}_j|} \right\} \quad (i = 1, 2, \ldots, n) \tag{10}$$

を計算しておき、実際に運動セグメント終端点の位置 $\boldsymbol{x}_n$ を変化させる時は、運動セグメントの各点の位置 $\boldsymbol{x}_i$ に差分係数 δ_i と終端点での変移ベクトル $\boldsymbol{d}_n$ を乗じたものを加えることで新しい点の位置 $\boldsymbol{x}'_i = \boldsymbol{x}_i + \delta_i \boldsymbol{d}_n$ を得ることができる。

3.3 位置と速度の拘束

前節で述べた位置の拘束に加えて、速度を拘束する手法について述べる。図 5 に示すように、運動セグメント $\mathbf{M}^x$ の最初の点 $\boldsymbol{x}_0$ の位置を保存したまま、時系列の最後の点の位置 $\boldsymbol{x}_n$ と速度 $\boldsymbol{v}_n$ を次の運動セグメント $\mathbf{M}^y$ の最初の点の位置 $\boldsymbol{y}_0$ と速度 $\boldsymbol{w}_0$ に一致させる拘束について考える。

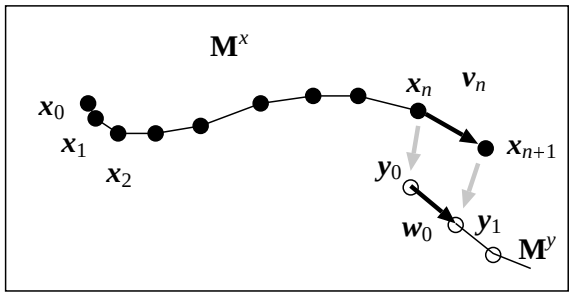

図 5: 終端点の位置と速度の拘束

ここで、運動セグメント $\mathbf{M}^x$ の各点 $\boldsymbol{x}_i$ に対して、各点の加速度 $\boldsymbol{a}_i$ を

$$\boldsymbol{a}_i = \boldsymbol{v}_{i+1} - \boldsymbol{v}_i \tag{11}$$

によって定義する。このとき、加速度 $\boldsymbol{a}_j$ を $\Delta \boldsymbol{a}_j$ だけ変化させて位置拘束と速度拘束を満たすと、 $\boldsymbol{v}_i$ と $\boldsymbol{x}_i$ の変化 $\Delta \boldsymbol{v}_i$ と $\Delta \boldsymbol{x}_i$ は、それぞれ

$$\Delta \boldsymbol{v}_i = \sum_{j=0}^{i-1} \Delta \boldsymbol{a}_j \tag{12}$$

$$\Delta \boldsymbol{x}_i = \sum_{j=0}^{i-2} \left\{ (i-1-j) \Delta \boldsymbol{a}_j \right\} \tag{13}$$

となる。ここで、簡単のために位置拘束と速度拘束を 1 次元での問題と考え、以下では x_i, v_i, a_i はそれぞれ位置、速度、加速度を表すスカラ量であるとする。このとき、式 (12)(13) を整理すると、下のように行列の形に書くことができる。

$$\begin{pmatrix} n-1 & 1 \\ n-2 & 1 \\ \vdots & \vdots \\ 1 & 1 \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} \Delta a_0 \\ \Delta a_1 \\ \vdots \\ \Delta a_{n-1} \end{pmatrix} = \begin{pmatrix} \Delta x_n \\ \Delta v_n \end{pmatrix} \tag{14}$$

この式 (14) は未知数よりも方程式の数が少ないため、これを満たすようなベクトル $\{\Delta a_i\}$ は一意に定まらないが、 $\sum |\Delta a_i|^2$ がもっとも小さくなるように計算を行うことで、運動セグメント全体での加速度の変化を最小にすることができる。ここで、加速度ベクトル $\{\Delta a_i\}$ に関して、前節と同様に元の速度が小さいところほど加速度を与えないという条件を与えるために、速度の大きさの係数 $r_i = |\boldsymbol{v}_i|$ と未知数 p_i を用いて

$\Delta a_i = r_i p_i$ と置き換え、これを式 (14) に代入して、

$$\begin{pmatrix} r_0(n-1) & r_0 \\ r_1(n-2) & r_1 \\ \vdots & \vdots \\ r_{n-2} & r_{n-2} \\ 0 & r_{n-1} \end{pmatrix}^T \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{pmatrix} = \begin{pmatrix} \Delta x_n \\ \Delta v_n \end{pmatrix} \tag{15}$$

とし、この式を満たすようなベクトル $\{p_i\}$ のうち $\sum |p_i|^2$ がもっとも小さくなるものを選ぶ。ここで、左辺の行列を $n \times n$ 正方行列に拡張した行列 $\mathbf{D}$ を定義し、特異値分解を用いてこれを列直交行列 $\mathbf{U}$, 対角行列 $\mathbf{W}$, 直交行列 $\mathbf{V}$ に分解する。$\mathbf{D}$ の階数は高々 2 であり、$\mathbf{W}$ の対角要素には、0 ではない特異値が多くて 2 つ含まれる。これらの特異値を $w_s = W_{ss}$, $w_t = W_{tt}$ とする。また、式 (15) の右辺のベクトルを拡張したものをベクトル $\boldsymbol{b} = (\Delta x_n \quad \Delta v_n \quad 0 \quad \cdots \quad 0)^T$ とすると、$\boldsymbol{p} = \{p_i\}$ は、行列 $\mathbf{W}$ の対角要素を逆数にした行列 $[\mathrm{diag}(1/W_{jj})]$ と行列 $\mathbf{U}$, $\mathbf{V}$ とによって

$$\boldsymbol{p} = \mathbf{V}\ [\mathrm{diag}(1/W_{jj})]\ (\mathbf{U}^T \boldsymbol{b}) \tag{16}$$

と書け、$\sum |p_i|^2$ は最小となる。式 (16) の右辺を展開すると、非 0 の特異値が 2 つしかないことから、

$$\begin{aligned} p_j &= \left(\frac{V_{js}U_{1s}}{w_s} + \frac{V_{jt}U_{1t}}{w_t}\right)\Delta x_n \\ &\quad + \left(\frac{V_{js}U_{2s}}{w_s} + \frac{V_{jt}U_{2t}}{w_t}\right)\Delta v_n \\ &= \alpha_j \Delta x_n + \beta_j \Delta v_n \end{aligned} \tag{17}$$

と整理でき、さらに式 (13)(17) より

$$\begin{aligned} \Delta x_i &= \Delta x_n \sum_{j=0}^{i-2} \left\{\alpha_j r_j (i-1-j)\right\} \\ &\quad + \Delta v_n \sum_{j=0}^{i-2} \left\{\beta_j r_j (i-1-j)\right\} \end{aligned} \tag{18}$$

を得る (α_j, β_j は行列 $\mathbf{D}$ から決まる定数)。このように、あらかじめ運動セグメント $\mathbf{M}^x$ の各点に対して、

$$\gamma_i = \sum_{j=0}^{i-2} \left\{\alpha_j r_j (i-1-j)\right\} \quad (i = 2, 3, \ldots, n)$$

$$\epsilon_i = \sum_{j=0}^{i-2} \left\{\beta_j r_j (i-1-j)\right\} \quad (i = 2, 3, \ldots, n)$$

によって差分係数 γ_i, ϵ_i を計算しておくことで、実際に運動セグメント終端点の位置 $\boldsymbol{x}_n$ と速度 $\boldsymbol{v}_n$ を $\Delta\boldsymbol{x}_n$ と $\Delta\boldsymbol{v}_n$ だけ変化させる時は、新しい点の位置を

$$\boldsymbol{x}'_i = \boldsymbol{x}_i + \gamma_i \Delta\boldsymbol{x}_n + \epsilon_i \Delta\boldsymbol{v}_n \tag{19}$$

によって計算することができる。

ただし、本節で述べた運動セグメントの編集手法を用いて終端点の位置を大きく変化させた場合には、セグメントの途中で不自然な折れ曲がりが発生する場合があることがわかっている。しかし、3.2節で述べた位置だけを拘束する手法を用いた場合にはこのような折れ曲がりは起こらない。このことを利用して、終端点の位置と速度を拘束する場合には、まず 3.2節での手法によって位置だけを拘束し、次に本節での手法によって速度だけを拘束するという手順を踏むことで、不自然さのない編集を行うことができる。

4 姿勢の再構築とモーション合成

本研究では、一般的なコンピュータグラフィックス制作ソフトウェアと同様に、質点をリンクで接続した骨格構造を用いてキャラクタの位置姿勢を計算機上に表現する。キャラクタの腕や足などは伸び縮みしないため、リンクの長さを一定に保つという幾何的な拘束が課せられる。一般的な Motion Editing 手法を用いてモーションの編集を行う場合には、このようなリンク長の拘束を維持したまま姿勢を表わす関節角の変更が行われる。しかし、本研究の手法では、モーション合成に要する計算量を削減するため、身体運動の特徴点に対する拘束条件はまず個別の点の運動セグメントの編集を行って独立に拘束を満足させる。このようにして個別に編集を行った運動セグメントではリンクの長さが一定に保たれないため、特徴点の位置をすべて満足するキャラクタの姿勢が存在しない可能性がある。

そこで、編集された特徴点の位置からキャラクタの位置姿勢を計算するために、本手法では逆運動学 (Inverse Kinematics) の手法の一つである質点系を用いた手法 [6] を利用する。この手法は比較的単純な質点の移動を繰り返す手法によって幾何的な矛盾を含む距離拘束条件を満たす骨格構造の位置と姿勢を計算することができる。ただし、距離拘束条件の適用は繰り返し計算による誤差の最小化によって行われるため、実用的な速度を得るための反復回数で計算を行った場合、得られる結果は必ずしも拘束条件を完全に満たしているわけではない。しかし、この手法では拘束を必ず満たす必要がある点に関しては位置を固定しておくことができるため、計算を途中で打ち切る場合でも必要な拘束を正確に満たしておくことができる。

5 結果

本研究で提案した手法を用いて一連のモーションの編集と合成を行った例を示す。なお、例題で用いた運動データはモーションキャプチャによって得られたものであり、サンプリング周波数は 15Hz である。図 6 は、モーションキャプチャによる一歩分の歩行モーションの最初と最後をなめらかに接続し、ループ化した歩行モーションを合成した例である。また、この歩行モーションに対して足先の接地位置を拘束し、歩幅を次第に狭めた例を図 7 に示す。図 8 は別に測定した段差を降りるモーションを歩行のモーションと組み合わせてモーションを合成した例を示している。

いずれの例でも、1 フレーム当りの姿勢の決定には R10000 (250MHz) を用いて 0.40 msec を要した。例では 15Hz のモーション生成を行っているので、これは 1 フレームの処理に使える時間 (1/15 秒) の 0.60% に相当する。

6 結論

本研究では、簡略化した Spacetime Constraints の手法を用いてモーションの編集を行い、モーションを実時間で合成する手法を提案した。この手法は、編集に必要な計算の大部分を前処理として行い、実際の編集は簡単なベクトルの線形和によってフレーム毎に高速に計算することを可能にした。この手法を用いてモーションキャプチャによって得られたデータを編集して接続し、一連のモーションを合成することで手法の有効性を示した。

また、本研究ではデータベースを離散的な時系列データによって表現しているが、これを曲線などのパラメトリック表現に置き換えることで記憶装置の使用効率を高めることができると考えられる。

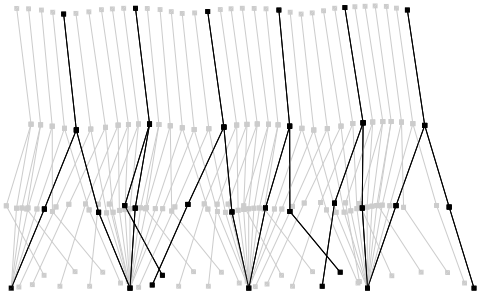

図 6: 歩行のループ化

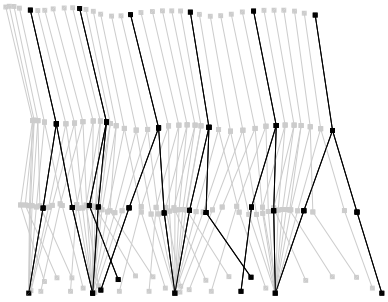

図 7: 歩幅の変更

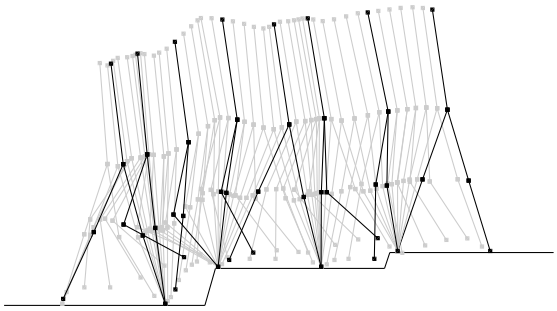

図 8: 段差

参考文献

[1] Alexis Lamouret and Michiel van de Panne. Motion synthesis by example. In *Computer Animation and Simulation '96*, pp. 199–212. Springer Computer Science, 1996.

[2] Andrew Witkin and Zoran Popovic. Motion warping. In *Computer Graphics Proceedings, Annual Conference Series*, 1995.

[3] Michael Gleicher. Retargetting motion to new characters. In *Computer Graphics Proceedings, Annual Conference Series*, pp. 33–42. ACM SIGGRAPH, 1998.

[4] Zoran Popovic and Andrew Witkin. Physically-Based motion transformation. In *Computer Graphics Proceedings, Annual Conference Series*, pp. 147–154. ACM SIGGRAPH, 1999.

[5] Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics Proceedings*, Vol. 22, pp. 159–168. ACM SIGGRAPH, 1988.

[6] Thomas Jakobsen. Advanced character physics. In *Game Developers Conference Proceedings*, pp. 383–401, 2001.