空間的キーフレーム法によるキャラクターアニメーション

Spatial Keyframing for Character Animation

五十嵐　健夫

Summary. This paper introduces "spatial keyframing", a technique for desinging character animation quickly. In traditional "temporal" keyframing, key poses are defined at specific time points. In contrast, key poses are defined at specific key positions in 3D space in spatial keyframing. The user can control a character by controling the position of a control cursor, where the pose of the character is given as an interpolation of the nearby key poses. As a result, the user can make a complicated motion in real time, and the resulting motion can be recorded as an animation sequence. We have been developing a prototype system, and found that one can design interesting animations using our technique.

1　はじめに

現在利用されているアニメーション生成手法としては，各時間における３D キャラクタの姿勢を手作業で細かく指定し，その間を計算機が補間するキーフレーム法がほとんどを占めている．キーフレーム法以外には，体中にセンサーを取り付けて人間の動作を直接取り込むモーションキャプチャと呼ばれる手法もあるが，高価な設備を必要とし個人で使用するものではない．近年，仮想的な３次元世界の中で物理シミュレーションを実行することで自然な動きを構成する手法が活発に研究されているが，これまでの所，プロダクションレベルで高度なアニメーションを生成することに主眼がおかれている．

我々は，このような既存のアニメーション生成手法に代わり，初心者でも直感的に利用できる手法として，ユーザによる操作をそのまま実時間で記録してアニメーションとする手法を提案し，それを支える諸技術の開発を行っている．本稿では特に，マウスのような通常の入力デバイスを用いて，多数の関節を持つ３D キャラクタに対して複雑な動きを与えることのできる空間的キーフレーム法について紹介する．具体的には，ユーザはまず，３D キャラクタの姿勢と，３次元空間中のハンドルの位置を結びつける「空間的キーフレーム」を設定する．あとは，ハンドルの位置をインタラクティブに変化させることで，複雑なアニメーションを手軽に生成することができるようになる．以下，関連研究について簡単に述べた後，実装したプロトタイプシステムの動作について説明する．

Takeo Igarashi, 東京大学情報理工学系研究科, John F. Hughes ブラウン大学計算機科学科

2　関連研究

本手法の基本的なアイデアは，いくつかキーとなるポーズを指定して，それらを補間するというものであり，関連する研究は数多く存在する．

Wiley らは，空間中にグリッド上に密に並べられた点にキャラクタのポーズを関連付け，それらを線形補間する手法を提案している[11]．本手法では，Raidal Basis Function を用いることで，より少ない数のキーから自然なポーズを生成することを可能としている．

Raidal Basis Function を利用したキャラクターのポーズの補間については，Rose らによって提案されている[8]．しかし，ここでの補間は，happy - sad といった形容詞に基づいてモーションキャプチャーしたデータを変形するために利用されている．

Ngo らは，２次元のベクトルグラフィクスの変形に，キーとなる形状の補間を用いている[5]．彼らのシステムにおいては，キーは，キャラクタが存在する３次元空間中ではなく，キーの間の関係を明示的に指定した Simplicial Configuration と呼ばれる特定の位相構造をもった空間に埋め込まれており，補間は線形補間として行われている．

Rademacher らは，カメラの位置によって，３次元キャラクタの見え方が変化するといった効果を実現するために，カメラ方向に基づいて用意された形状の補間を行っている．ここでも補間には線形補間が利用されている[7].

以上，既存の補間を用いた手法と比較した場合，本手法の特徴は，キーがキャラクタと同じ３次元空間に存在している点，補間に Radial Basis Function を

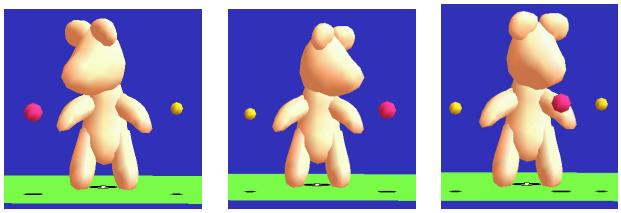
用いることにより少ない点で自然な動作を実現している点にあるといえる．ただし，本提案の最大の特徴は，キーの補間を利用することによって手早くアニメーションを設定できるようなシステム全体のデザインにある．

補間とは異なるが，Kurlander らは，ユーザの与えた複数の例を元に，長さ一定や角度一定といった制約を，自動的に抽出する手法を提案している[4].

3　プロトタイプシステムの動作

本プロトタイプでは，モデリングソフトウェア[1]で作成した階層的な３次元モデル(剛体パーツが回転可能なリンクで結合されたもの)を読み込み，ドラッグ操作により関節角を変化させることで，姿勢を設定することができる．画面上には，ピンク色のハンドルが存在しており，マウスドラッグにより三次元空間中での位置を操作することができる．ハンドルそのものをドラッグした場合には画面に平行な平面内を移動し，ハンドルの影をドラッグした場合には地面に平行な平面内を移動する[3]．set ボタンを押すことで，ハンドルのある場所にキーを打つことができる．キーの設定された場所は黄色いボールで示され，ボタンが押された時点でのキャラクタの姿勢がその点に関連付けられる．いくつかのキーを３次元空間中に設定後，ハンドルをドラッグすることで，キャラクタの姿勢が連続的に変化する(図 1)．キャラクタの姿勢は，ハンドルの周りにあるキーに関連付けられている姿勢を滑らかに混ぜ合わせたものとなる．混ぜ合わせには，RadialBasisFunction を用いた補間(Variational Interpolation)を利用している*．ハンドルを動かすことで，リアルタイムでいろいろな動きを表現できるほか，その動きを記録することでアニメーションが生成できる．具体的な例として，ジャグリング・キック・ダンス・尻文字などの例を作成して有効性を確認した(図 2)．

本手法の特徴としては，まず第一に，ハンドルの位置の操作だけで，一度に多くの関節角を同時に操作することが可能な点が挙げられる．通常は，関節角を一つずつ動かすことが必要であり，リアルタイムで豊かな動作を実現することは難しい．第二に，キーの追加が非常に手軽に行える点が挙げられる．関連研究[11]では，グリッド上に多数のキーを配置しなければならず，また，補間の範囲は，それらの内部に限られている．本手法の場合には，少数の点を指定するだけで，それらの内部に止まらない広範な範囲における補間を行うことができる．



a)　b)　c)

図 1.　２つのキーを設定した場合の例．左の点が図 a のポーズに，右の点が図 b のポーズに関連付けられている．設定後，ピンクのハンドルを左から右へドラッグすると図 c のように中間のポーズが生成される．ピンクのハンドルは，画面に平行な平面上を移動するが，地面の影をドラッグすることで地面に平行にも移動できる．より多くのキーが設定されている場合にも同様に周囲のキーを元に適切なポーズを生成する．

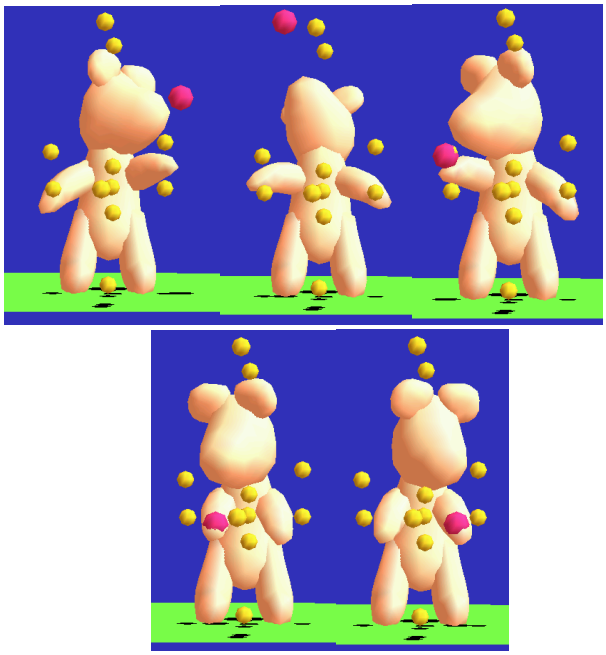


図 2.　３次元の空間的キーフレームによるジャグリング．黄色い点それぞれに固有のポーズが関連付けられている．システムは，ピンク色のハンドルの位置に応じて周囲の黄色い点に関連付けられているポーズを補間して適切なポーズを生成する．ユーザはピンクのボールをドラッグしていくことで，アニメーションを生成することができる．

3.1　インバースキネマティクスとの組み合わせ

インバースキネマティクス[2][10]は，指先や足先といったキャラクタの関節の先の点が指定された位置に来るように，途中の関節角を自動的に計算する手法である．しかし，通常のインバースキネマティクスを利用した場合，端点の位置は正確に制御できるが，途中の関節の形状が思い通りにならないといっ

*補間方法としては他の手法も考えられるが，本プロトタイプでは，少ない制約数でグローバルに滑らかな結果が得られ，かつ高速で実装が簡単なために RBF を用いている．他の補間手法との詳細な比較は今後の課題である．

た問題がある(図 4 中央)．また，前フレームの形状を元に現在の指定された端点へ「引き寄せる」作業になるため，端点が同じ場所であっても，直前の姿勢によって出力される姿勢が異なるといったことも起こる．通常は，このようなことを避けるために，関節の回転角や重みといった付加情報を与えているが，これらはあまり直感的でなく，デザイナーは試行錯誤に時間を費やしがちである．

このような場合に，空間的キーフレームとインバースキネマティクスを組み合わせることで問題を解決することが可能となる．すなわち，操作対象としたい端点をハンドルとして利用することとし，いくつか代表となる点とその点における姿勢をキーフレームとして与える．これに対して，空間的キーフレーム法を適用することで，希望するものに近い姿勢が得られるが，この時点では，端点は正確にハンドルの位置に重なっていない．そこで，空間的キーフレームによって計算された姿勢を初期状態としてインバースキネマティクスによって端点をハンドルへと「引き寄せる」ことによって，正確な追従が実現される(図３)．このようにすることで，端点以外の関節部分の形状を意図どおりに制御し，かつ，一定のハンドル位置に対して前フレームでの姿勢に関わらず常に同じ結果を得ることが可能となる(図４)．

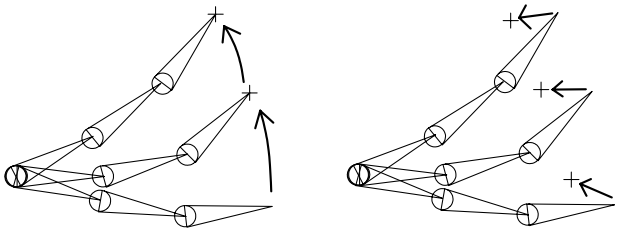


図 3. 通常のインバースキネマティクス（左）と空間的キーフレーム法との組み合わせ（右）．前者の場合には，前フレームの姿勢を元に，現在の端点へと「引き寄せる」作業を行う．後者の場合には，まず，現在の端点をハンドルとして，空間的キーフレーム法により姿勢の概形を得る．その後，その姿勢を元に，目標とする端点へ「引き寄せる」ことで，正確な追従を行う．

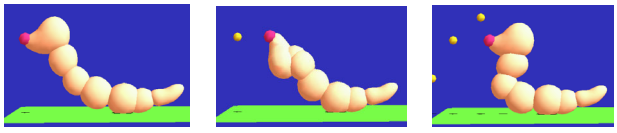


図 4. 初期状態(左)と通常のインバースキネマティクスによる結果（中央）と空間的キーフレーム法との組み合わせによる結果（右）．通常のインバースキネマティクスの場合には，端点の位置は正確なものの，残りの関節の形状の制御が困難である．空間的キーフレームと組み合わせることにより，少数の例(キー)を設定することで，望みどおりの姿勢を得ることができる．

3.2　歩行動作の実現

歩行動作のように，キャラクタ自身が移動する場合には，ハンドルを空間中に固定して本体の位置のみを変化させることで，ハンドルのキャラクタの中心に対する相対的な位置を変化させて，適切なアニメーションを生成することができる(図５)．通常の時間的なキーフレーミングによって歩行を表現した場合には，足先が地面の上を滑らないようにするために，再生速度と移動速度の関係を注意して調整しなくてはならない．一方，空間的キーフレーミングを利用した場合には，足先と地面の相互の位置関係によって自動的に適切なポーズが設定されるので，そのような心配がない．たとえば，キャラクタを早く移動させればそれに合わせて足の動きも自動的に早くなる．

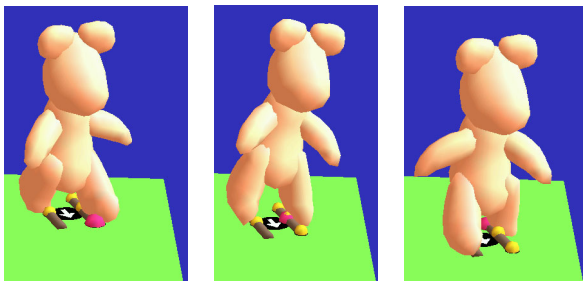


図 5.　歩行動作の実現例．足先の場所に応じて，４つのキーが設定されている．ピンク色のハンドルを地面に対して固定し，キャラクタの位置を移動させることで，ハンドルの相対的な位置が変化し，ポーズがそれに準じて変化している．

4　アルゴリズム

本手法のアルゴリズムの核は，ハンドルの位置(x,y,z 座標)を入力として受け取り，姿勢(関節の変換行列の組)を返す関数である．ここで，各関節の姿勢をどのようにして表現するかが問題となる．もっとも一般的な方法は，オイラー角や４元数(quaternion) [10]を用いる方法であるが，この方法では，外挿したときに，回転が続いてしまうといった問題や，一周するときに不連続が生じるといった問題がある．具体的例を以下に示す．

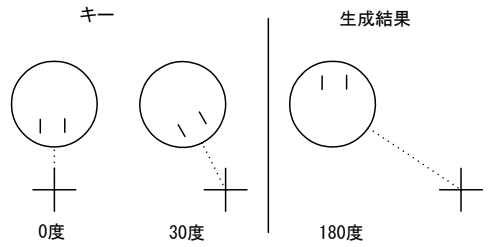


図 6. 角度により姿勢を表現した場合の問題点１．生成結果においては，なるべくキャラクタの顔がハンドルの方を向くのが望ましいが，角度を利用すると，徐々に離れていってしまう．さらにハンドルを右へ動かしていくと，回転が続いてしまう．用途によっては，角度を用いた結果が望ましい場合もあるが，空間的な対応を考えると，このような結果は望ましくない．同様のことは４元数を利用した場合でも発生する．(左側のキーが無回転として４元数が(1,0,0,0)，右側のキーが x 軸周りの３０度回転で４元数が(0.97,0.26,0,0)とすると，ハンドルを右へ右へと持っていくと，外挿して正規化した結果ほぼ(0,1,0,0)へと近づく．これは x 軸周りの 180 度回転であり，図の右のような結果になる)

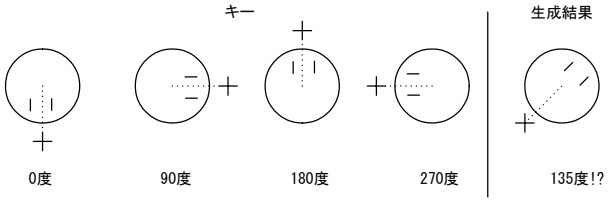


図 7. 角度により姿勢を表現した場合の問題点 2．生成結果においては，なるべくキャラクタの顔がハンドルの方を向くのが望ましいが，角度を利用すると，0 度と 270 度の中間の 135 度となり，不自然な結果になる．これは角度表現を用いた場合には 0 度と 360 度の間に不連続が存在するためである．同様のことは４元数を利用した場合でも発生する(x 軸周りの回転とすると，４元数は左から順に(1,0,0,0), (0.7,0.7,0,0), (0,1,0,0), (-0.7,0.7,0,0)となる．これらから計算して正規化すると右の位置のハンドルに対しては，ほぼ(0.7,0.7,0,0)となるがこれは 90 度の回転に近い．).

このような問題を解決するために，各関節の姿勢を回転行列としてあらわし，その行列の９要素を直接 RadialBasisFunction を用いた Variational Interpolation によって評価する方法を取る[6][9]. これは言い換えると，回転した後の座標系を決定する３つの基底ベクトル(x, y, z) をもって，回転を表現するものである．この場合，３つの基底ベクトルは独立した Variational Interpolation により計算するので，３つが必ずしも正規直交となるわけではない．よって，Variational Interpolation によってまず計算したあとに，それらにもっとも近い正規直交基底に強制的に変換する処理を行う(補遺参照)．このようにすることによって，図 6，7 のような例でも，生成結果として直感的に正しい結果を得ることができる．

より具体的な Variational Interpolation の計算は以下のようになる[9]．まず，入力として，n 個のキーの 3 次元座標 ci とその位置での姿勢を表す回転行列の要素 hi の組が与えられる．これをもとに，ハンドルの座標 x から, 回転行列の要素 h を計算する関数 f(x) を求める．この関数は，$f(\boldsymbol{x}) = \sum_{j=1}^{j=n} d_j \Phi(\boldsymbol{x} - c_j) + P(\boldsymbol{x})$ の形とする．ここで

$\Phi(\boldsymbol{x}) = |\boldsymbol{x}|$ とする†.

これが与えられたキーの制約(キーの位置 ci と対応する出力 hi)を満たすことから

$$hi = \sum_{j=1}^{j=n} d_j \Phi(c_i - c_j) + P(c_i)$$

これは以下のような線形システムとして表される．

$$\begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1n} & 1 & c_1^x & c_1^y & c_1^z \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \Phi_{n1} & \cdots & \Phi_{nn} & 1 & c_n^x & c_n^y & c_n^z \\ 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & \cdots & c_n^x & 0 & 0 & 0 & 0 \\ c_1^y & \cdots & c_n^y & 0 & 0 & 0 & 0 \\ c_1^z & \cdots & c_n^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ \vdots \\ d_{1n} \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ \\ h_n \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

ただし $c_i = (c_i^x, c_i^y, c_i^z)$ ，$\Phi_{ij} = \Phi(c_i - c_j)$ ，

$$P(\boldsymbol{x}) = p_0 + p_1 x + p_2 y + p_3 z$$

これを解くことによって係数 が得られるので, 最終的に f(x)が求められる．

上記によって求められる関数 f(x)は，位置関節につき 9 個必要になるので，関節が m 個ある場合には，9m 個の関数を利用することになる．

5 まとめ

本稿では，キャラクターアニメーションを手軽に作成するための手法として，空間的キーフレーム法について紹介した．空間的キーフレーム法の利用方法として，本稿ではユーザが直接ハンドルを操作してそれを記録する方法を紹介したが，ハンドルの動き

† 他に $\Phi(\boldsymbol{x}) = |\boldsymbol{x}|^3$ とすることも可能であるが，こうすると，より滑らかになる分，キーから離れていったときの動きが大きくなりがちで扱いにくいため，本実装では避けた．どのような関数が適切かは利用者および状況によって異なるものと考えられる．

を簡単なスクリプトで設定したり，物理シミュレーションを適用して自動的に生成したりすることによって，より多様なアニメーションを作成することが可能である．

今後の課題については，まず現時点では個別に独立した動作を作成することしかできないので，異なる動作が連続するようなアニメーションを作成できるように拡張を行っていきたい．また，複数のキャラクタの動作をまとめてデザインしたり，特定のキャラクタに対して設定されたアニメーションを他のキャラクタのアニメーションに転用したりする手法についても研究していく予定である．また，実際のユーザによる操作実験も行っていく必要がある．

参考文献

[1] 五十嵐 健夫, Dennis Cosgrove, Randy Pausch, スケッチによる可動階層構造付き 3 次元モデル生成手法, インタラクティブシステムとソフトウェア VII, pp.7-12, 1999.

[2] Girard, M., and Maciejewski, A. A. Computational modeling for the computer animation of legged figures. In Computer Graphics (July 1985), pp. 263・70. Proceedings of SIGGRAPH 85.

[3] K. P. Herndon, R. C. Zeleznik, D. C. Robbins, D. B. Conner, S. S. Snibbe, A. van Dam. Interactive Shadows. 1992 UIST Proceedings, pp. 1-6, 1992.

[4] D. Kurlander, S. Feiner. Inferring constraints from multiple snapshots, ACM Transactions on Graphics, 12 (4), pp. 277-304, October 1993.

[5] T. Ngo, D. Cutrell, J. Dana, B. Donald, L. Loeb, S. Zhu, Accessible Animation and Customizable Graphics via Simplicial Configuration Modeling, SIGGRAPH 2000 Conference Proceedings, pages -. 1999.

[6] Powell, M. J. D. Radial basis functions for multivariable interpolation: A review. In Algorithms for Approximation, J. C. Mason and M. G. Cox, Eds. Oxford University Press, Oxford, UK, pp. 143-167, 1987.

[7] P. Rademacher. View-dependent geometry. SIGGRAPH 99 Conference Proceedings, pages 439-446. 1999.

[8] Rose, C., B. Bodenheimer, and M. Cohen, Verbs and Adverbs: Multidimensional motion interpolation using radial basis functions. IEEE CGAA, 1998.

[9] G. Turk and J. F. O'Brien. Variational implicit surfaces. Technical Report GITGVU 9915, Georgia Institute of Technology, May 1999.

[10] A. Watt, M. Watt, Advanced Animation and Rendering Techniques: Theory and Practice, Addison-Wesley, 1992

[11] Wiley, Doug and Hahn, James, "Interpolation Synthesis of Articulated Figure Motion", IEEE Computer Graphic and Applications, November/December 1997, Volume 17, No. 6, pp. 39-45.

補遺　正規直交基底への変換手法

ここでは，独立に補間により計算された３つの基底ベクトルをもとに，それらに近い正規直交基底を生成する計算方法について述べる．なお，この計算方法は，実験的にうまく動くように設計したもので，数学的に厳密に定義された値を最小化するものではない．

まず，入力基底ベクトルを $\vec{x}_0, \vec{y}_0, \vec{z}_0$ として，まずそれらをそれぞれ正規化する．する．次に，$\vec{u}_0 = \vec{y}_0 \times \vec{z}_0, \vec{v}_0 = \vec{z}_0 \times \vec{x}_0, \vec{w}_0 = \vec{x}_0 \times \vec{y}_0$ を計算し，それぞれ正規化する．最後に $\vec{x}_1 = (\vec{x}_0 + \vec{u}_0)/2$ ， $\vec{y}_1 = (\vec{y}_0 + \vec{v}_0)/2$ ，$\vec{z}_1 = (\vec{z}_0 + \vec{w}_0)/2$ を計算し，それぞれ正規化する．これを繰り返し，残渣 $r = (\vec{x}_n \cdot \vec{y}_n)^2 + (\vec{y}_n \cdot \vec{z}_n)^2 + (\vec{z}_n \cdot \vec{x}_n)^2$ が一定値(0.000001)以下になるか，繰り返し数が 10 を超えたところで停止する．結果として得られるものは，厳密には正規直交基底とはならないが，ほとんどの場合，実用上問題ない結果が得られる．どうしても収束しない場合や，不連続性が現れる場合には，そもそも与えられたキーの設定では自然な結果が得られない状態であると考えられるので，ゆがんだ結果を示してユーザにキーの再設定を要求する．