対話型3次元アプリケーションのための幾何制約解消法

細部博史†

近年，コンピュータ技術の進展により，パーソナルコンピュータ上で高速で高精細な3次元グラフィクスが可能になり，3次元アプリケーションの開発が容易化されている．しかし，それらの技術のほとんどが，3次元グラフィクスにおける配置や振舞いの面を十分にサポートしているとはいえない．一般に，幾何制約は，グラフィカルオブジェクトの配置や振舞いを記述する強力な道具であり，対話型2次元グラフィカルユーザインタフェースや専用の3次元グラフィクスパッケージで用いられている．本論文では，対話型3次元アプリケーションを対象とした幾何制約解消法を提案する．本手法の新規な点は，幾何制約解消において座標変換を処理することで，シーングラフに対応できる点である．本手法によってプログラマは，幾何配置や，制約付きドラッグ，インバースキネマティクスなどの種々の目的のために3次元幾何制約を利用できるようになる．本論文では，本手法を実装した制約解消系 Chorus3D を対話型3次元アプリケーションに適用した例を与えることで，本手法の有効性を示す．

Geometric Constraint Satisfaction for Interactive 3D Applications

Hiroshi Hosobe†

Recent computer technologies have enabled fast high-quality 3D graphics on personal computers, and also have made the development of 3D applications easier. However, most of such technologies do not sufficiently support layout and behavior aspects of 3D graphics. Geometric constraints are, in general, a powerful tool for specifying layouts and behaviors of graphical objects, and have been applied to interactive 2D graphical user interfaces and specialized 3D graphics packages. In this paper, we present a geometric constraint solving method for interactive 3D applications. Its novel feature is to handle scene graphs by processing coordinate transformations in geometric constraint satisfaction. It enables programmers to use 3D geometric constraints for various purposes such as geometric layout, constrained dragging, and inverse kinematics. We demonstrate its usefulness by presenting sample interactive 3D applications using Chorus3D, a constraint solver which implements our method.

1. はじめに

近年，パーソナルコンピュータにおける3次元グラフィクス機能の高速化および高精細化が進行している．また，VRML や Java 3D などのソフトウェア技術が，3次元グラフィクスの開発を容易化している．しかし，それらの技術のほとんどが主に3次元グラフィクスのレンダリング面に重点を置いており，配置や振舞いの面を十分にサポートしているとはいえない．

一般に，**幾何制約**は，グラフィカルオブジェクトの配置や振舞いを記述するのに，きわめて強力な道具である．描画エディタなどの対話型2次元グラフィカルユーザインタフェースで図形の幾何配置や振舞いの記述に制約が役立つことは広く認められており，このような目的のための**制約解消系**の研究が数多く行われている[3),7),9)~11),13)~16),20),22),23),25)]．また，専用の3次元グラフィクスパッケージでも，制約やそれに類似する機能を用いることで，オブジェクトの配置や振舞いを指定できるものは多い．

対話型3次元アプリケーションにおいても幾何制約が有効であると考えることは自然である．そして，そのためには，従来の2次元を対象とした制約解消系を単純に3次元を扱えるように修正すればよいように思われるかもしれない．しかし，実際には，2次元グラフィクスと3次元グラフィクスを記述するうえでの本質的な相違のために，それだけでは不十分である．2次元グラフィクスでは，通常，単純な座標系のみを扱うのに対して，3次元グラフィクスでは，**シーングラフ**を扱うために，回転などの複雑な関係を持つ複数個の座標系を扱う必要があるためである．

本論文では，対話型3次元アプリケーションに幾何

† 国立情報学研究所
National Institute of Informatics

制約を統合するための制約解消法を提案する．本手法では，幾何制約解消における**座標変換**の処理を可能にすることで，シーングラフをともなった 3 次元グラフィクスに対応している．本手法は，本研究者が以前に提案した 2 次元幾何制約解消の枠組み[15),16)]を基礎とするものである．

本手法はまた，その基礎の枠組みから**モジュール機構**[15),16)]を継承することで，ユーザによる制約の種類の定義を可能にしている．この機構によってプログラマは，以下を含む種々の目的で 3 次元幾何制約を利用できる．

幾何配置： 本手法の典型的な用途は，グラフィカルオブジェクトの配置である．たとえば，オブジェクトを平行または垂直に配置することが，必要なパラメータを前もって計算することなく実現できる．また，スプリングモデル[17)]に基づくグラフ配置も可能である．

制約付きドラッグ： 本手法によって，配置のための制約をともなったオブジェクトのドラッグが可能である．たとえば，オブジェクトを球体の表面に制約しながらドラッグできる．3 次元グラフィクスにおいて制約付きドラッグは，通常のマウスドラッグ操作を 3 次元空間に容易に対応させることができるため，重要である．

インバースキネマティクス： 本手法は,「関節」を持つオブジェクトの望ましい形状を求める問題であるインバースキネマティクス[1),26)]に対して適用可能である．関節を持つオブジェクトを座標変換により指定しておくと，制約を満たす座標変換のパラメータを自動計算できる．この方法は，移動するオブジェクトを目標とするカメラ制御に対しても適用可能である．

本手法は，別の重要な特徴として，**強さ**と呼ばれる階層的な優先度を持つ「軟らかい」制約を処理する機能（**制約階層**[7)]）を基礎の枠組みから継承しており，グラフィカルオブジェクトのデフォルトの配置や振舞いの記述に適している．本手法では，解の決定において，矛盾を生じる弱い制約を満たさないようにすることで，できるだけ多くの強い制約を充足することができる．

本手法は，制約解消系 Chorus3D としてすでに実装されている．本論文では，Chorus3D を適用した対話型 3 次元アプリケーションの例を与えることで，本手法の有効性を示す．

本論文は以下の構成からなる．まず 2 章で，本提案手法と関連する従来研究について紹介する．次に 3 章で，対話型 3 次元アプリケーションで制約を利用するための本手法のアプローチについて述べる．4 章では，本手法が基礎とする制約の枠組みを述べる．5 章では，本手法において座標変換処理を実現するための技法を提案する．6 章では本手法の実装である Chorus3D 制約解消系について述べ，7 章で対話型 3 次元アプリケーションに対する適用例を示す．8 章で本手法に関する種々の議論の後，9 章で結論と今後の課題を述べる．

2. 関 連 研 究

これまでに，制約やそれに類似する機能を 3 次元グラフィクス言語に統合することで，グラフィカルオブジェクトの指定を容易化する研究が存在する．たとえば，VRML[4)]におけるイベント伝播機構を 1 種の制約として見なすことが可能である．また，VRML に単方向制約と有限領域制約を導入する研究[5)]も行われている．しかし，これらは Euclid 幾何制約のような強力な非線形制約を連立して扱うことができない．

グラフィカルユーザインタフェースの分野では，多くの制約解消系が開発されている[3),7),9)~11),13)~16),20),22),23),25)]．しかし，それらのほとんどは 2 次元インタフェースを対象としており，3 次元グラフィクスに必要な座標変換処理機構を提供していない．例外的に，Gleicher が提案した differential アプローチ[9),10)]では，3 次元幾何制約と座標変換がサポートされている．このアプローチは，ユーザによる幾何制約の種類の定義を可能にしており，その意味では本論文の提案手法と同様の目的を持っていると見なすことができる．しかし，本論文の提案手法と異なり，このアプローチは仮想的な力学シミュレーションを実行することで制約解消を実現しているものである．そのため，このアプローチは，本論文の提案手法と解の振舞いの点でまったく異なり，また解を制御するインタフェースも大きく異なる．一方，本論文の提案手法を実装した制約解消系 Chorus3D は，従来の制約解消系である Cassowary[3)]や Chorus[15),16)]との互換性の高いインタフェースを提供している．

CAD などの分野において，記号的な手法に基づく幾何制約解消の研究が行われている[8),19),21)]．記号的手法は通常，点や直線などの幾何的オブジェクトに関する Euclid 幾何制約を扱い，制約系の解析に基づいてオブジェクトを順に適切に配置していく構成的な方法によって制約を充足する．記号的手法は高速かつ正確な制約解消を実現できるが，軟らかい制約や不等式制約を処理することや，Euclid 幾何制約以外の幾何制

約を扱うことが困難である．

コンピュータグラフィクスやロボティクスの分野では，インバースキネマティクスの研究が行われている[1),26)]．しかし，インバースキネマティクスは通常，専用のソフトウェアとして実装されており，限られた種類の幾何制約しか扱うことができない．

3. 本研究のアプローチ

本論文では，対話型 3 次元アプリケーションに幾何制約を統合するための制約解消法を提案する．その実現のために，本研究者が以前に開発した 2 次元幾何制約解消の枠組み[15),16)] を 3 次元へ拡張する．その際の重点は，階層構造からなる座標系群を制約解消において処理できるようにすることである．

座標系の階層をサポートするために，以下のような制約のモデルを新たに導入する．

3 次元変数： 各 3 次元変数（実数値をとる制約可能変数の 3 つ組からなる）は 1 つの座標系に結び付けられ，その値は局所座標として表される．

幾何制約： 3 次元変数に関する幾何制約は，3 次元変数の世界座標を用いて評価される（ただし，距離や角度などを表す 1 次元変数を参照する際に，その値を直接用いることも可能である）．なお，1 つの制約が，異なる座標系に属する 3 次元変数を参照することも可能である．

座標変換： 座標変換のパラメータは制約可能変数として与えられ，制約解消系は制約を適切に充足するために，そのようなパラメータを変更できる．

このモデルに基づくことで，シーングラフを利用して幾何的オブジェクトをモデル化できると同時に，制約を利用して幾何的関係を管理できるようになる．

本手法の実装である Chorus3D 制約解消系では，以下の基本的な 3 種類の座標変換を提供している．

平行移動： 座標変換は 3 つの変数 t_x，t_y，t_z で特徴付けられ，ベクトル (t_x, t_y, t_z) による平行移動を指定する．

回転： 座標変換は 4 つの変数 r_x，r_y，r_z，r_w をパラメータとし，軸 (r_x, r_y, r_z) の周りの角度 r_w の回転を指定する☆．

スケール： 座標変換は 3 つの変数 s_x，s_y，s_z で表現され，原点を中心とする軸ごとのスケール (s_x, s_y, s_z) を指定する．

☆ 一般に 3 次元空間における回転の自由度は 3 であるため，4 つの変数を用いるこの回転の表現方法は冗長である．しかし，この方法は，回転の軸をベクトルで直接的に表現できるために使いやすく，実際に VRML などで利用されている．

このような基本的な座標変換を用いることで，現実に有用な座標変換の多くを表現できる．実際に，VRML の `Transform` ノードによる座標変換はすべて，これら 3 種類の変換を組み合わせることで実現可能である[4)]．

4. 制約の枠組み

本章では，本提案手法において制約を扱うための基礎的枠組みを簡潔に述べる．これは本研究者が以前に開発した 2 次元幾何制約解消の枠組みと同様である．その詳細については文献 15), 16) を参照されたい．

4.1 定 式 化

最初に，制約と制約系の定式化を与える．以下では，変数を表現するために n 個の変数からなる変数ベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ を，また変数の値を表すために n 個の実数からなる変数値ベクトル $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ を導入する（v_i は x_i の値を意味する）．

種々の幾何制約を統一的に表す手段として，**誤差関数**を導入する．誤差関数 $e(\boldsymbol{x})$ は通常 1 つの算術制約に対応しており，変数値ベクトルを入力とし，非負実数で表される誤差を出力とする関数として定められる．すなわち，$e(\boldsymbol{v})$ は，対応する制約の $\boldsymbol{v}$ に対する誤差を与える．誤差関数の結果が 0 であることは，制約が正確に満たされていることを示す．たとえば，制約 $x_i = x_j$ に対しては，$e(\boldsymbol{x}) = (x_i - x_j)^2$ を用いることができる．また，各 $e(\boldsymbol{x})$ について，その勾配 $\nabla e(\boldsymbol{x})$ が既知であると仮定する☆☆．

$$\nabla e(\boldsymbol{x}) = \left(\frac{\partial e(\boldsymbol{x})}{\partial x_1}, \frac{\partial e(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial e(\boldsymbol{x})}{\partial x_n} \right).$$

本枠組みが扱う制約系は，制約階層[7)] と同様に，強さの等しい制約を含む有限個のレベルからなる．最も強いレベルの制約は必須制約（硬い制約）であり，必ず満たされる（不可能な場合は解なしになる）．一方，それ以外の制約は選好制約（軟らかい制約）であり，より強い制約に矛盾する場合に緩和されることがある．

制約系の解は以下のように定義される．$e_{i,j}(\boldsymbol{x})$ を，レベル i $(0 \leq i \leq l)$ の j 番目 $(1 \leq j \leq m_i)$ の制約の誤差関数とする（直観的には，i が大きいほど，レベル i の制約は弱くなる）．このとき，解 $\boldsymbol{v}$ は，次の

☆☆ 一般には，任意の誤差関数において，その勾配が既知であるとは限らない．したがって，この仮定は本論文の提案手法の適用可能範囲を制限するものである．このような制限を課す理由は，本手法が準 Newton 法などの非線形数値最適化手法を用いており，その効率的な実現のためには，誤差関数の勾配を利用できるほうが有利であるためである．

最適化問題で決定される．

$$\begin{aligned} &\underset{\boldsymbol{v}}{\text{minimize}} \;\; E(\boldsymbol{v}) \\ &\text{subject to} \;\; e_{0,j}(\boldsymbol{v}) = 0 \;\; (1 \le j \le m_0). \end{aligned}$$

ただし，E は次のように定められる目的関数である．

$$E(\boldsymbol{x}) = \sum_{i=1}^{l} \sum_{j=1}^{m_i} w_i e_{i,j}(\boldsymbol{x}).$$

ここで，w_i は，強さ i に対応する重みであり，$w_1 \gg w_2 \gg \cdots \gg w_l$ である．この定式化において，レベル0の制約は必須制約に，それ以外は選好制約に対応する．直観的には，重い（強い）選好制約ほど，より良く満たされることになる．

この定式化は，制約階層を近似するものである．特に誤差関数の定義に制約違反の2乗を用いる場合，この定式化により，least-squares-better[3),20)]という基準で解かれる制約階層の近似解を得ることができる．最大の違いは，この定式化では，本来無視されるべき，強い制約に矛盾する弱い制約を多少考慮してしまう点である．

本枠組みの実装であるChorus3D制約解消系では，強さとして`required`（必須），`very strong`，`strong`，`medium`，`weak`，`very weak`の6つを採用している．ただし，`very strong`と`very weak`は制約解消系が内部的に使用する強さであり，前者は非線形または不等式制約を近似的に処理するために，後者は新たな解を以前の解にできるだけ近付けるために用いられる．実際の重みとしては，`very strong`，`strong`，`medium`，`weak`，`very weak`に対して，それぞれ 32^4，32^3，32^2，32^1，1を割り当てている．これらの重みは，実際に用いる数値最適化手法の精度に応じて決定する必要がある．上述の重みを用いる場合，たとえば`strong`の制約 $x = 0$ と`medium`の制約 $x = 100$ からなる制約系に対して，解 $x = 3.0303\cdots (= 100/33)$ が得られる．したがって，制約階層の近似として十分な精度であるとはいえないものの，ほとんどのグラフィカルアプリケーションにおいて制約の強さの違いは明白である．

4.2 制約解消アルゴリズム

上述の制約系の解を実際に求めるためには，対応する最適化問題を解く必要がある．そのために本枠組みでは，数値最適化手法と遺伝的アルゴリズムを組み合わせることで制約解消アルゴリズムを実現している．このアルゴリズムでは，局所解を求めるために数値最適化を利用し，大域解を求めるために遺伝的アルゴリズムを採用している．

数値最適化のために，現在の実装ではBroyden-Fletcher-Goldfarb-Sahnno（BFGS）公式による準Newton法[2),6)]を用いている．これは超1次収束する高速な反復法として知られているもので，過去の履歴を利用することで無駄な探索を避けるため，素朴なNewton法よりも通常，高速である．

遺伝的アルゴリズムを導入した理由は，幾何制約を数値的に充足する場合に局所解に陥りやすいという欠点[13),19)]を補うためである．一般に，遺伝的アルゴリズムとは，複数の解候補の集団から次の世代の解候補の集団を生成する処理を繰り返す，確率的な探索手法である[12),18)]．本枠組みにおいて遺伝的アルゴリズムを必要とするのは，通常，最初の解を計算する場合に限られる．一方，修正された制約系を解き直す場合には，通常，遺伝的アルゴリズムを用いずに，数値最適化を直前の解に対して適用すればよい．

5. 座標変換処理

本章では，前章の制約の枠組みに対して座標変換の処理を統合するための技法を提案する．

3章で述べたように，3次元幾何制約を評価するために，点の世界座標を用いる．これを単純に実現する方法として，すべての祖先の座標系において3次元変数を複製したうえで，それらの間に座標変換を表現する必須制約を課す方法が考えられる．しかし，この方法は，非線形な必須制約を処理する最適化ルーチンを必要とするため，実際に利用可能な数値手法の選択肢を狭めることになる（事実，この場合に準Newton法は利用できない）．また，この方法では変数と制約を数多く生成してしまうことから，多くの記憶領域が余分に必要とされる．

以下では，より広範囲に適用可能な座標変換処理技法を提案する．その特徴は，最適化ルーチンに対して座標変換を隠蔽する点である．本技法では，座標変換を誤差関数に埋め込むことで，これを実現する．

5.1 モデル

まず，新しい変数ベクトル $\boldsymbol{x}' = (x'_1, x'_2, \ldots, x'_n)$ を導入する．これは，$\boldsymbol{x}$ に含まれる3次元変数の局所座標を表す変数を，対応する世界座標を表す変数に置き換えたものである（ただし，1次元変数は不変とする）．この処理は，以下のように数学的にモデル化できる．まず，s 個の座標変換の列を考える．

$$\boldsymbol{y}_0 \,(= \boldsymbol{x}) \xrightarrow{\boldsymbol{t}_0} \boldsymbol{y}_1 \xrightarrow{\boldsymbol{t}_1} \cdots \xrightarrow{\boldsymbol{t}_{s-1}} \boldsymbol{y}_s \,(= \boldsymbol{x}').$$

ここで，$\boldsymbol{y}_0$ と $\boldsymbol{y}_s$ はそれぞれ $\boldsymbol{x}$ と $\boldsymbol{x}'$ に等しく，各 $\boldsymbol{y}_k$ $(1 \le k \le s-1)$ は「中間的」な変数ベクトルであり，各 $\boldsymbol{t}_k$ $(0 \le k \le s-1)$ は $\boldsymbol{y}_k$ を $\boldsymbol{y}_{k+1}$ に変換する

関数である．直観的に，$\boldsymbol{t}_k$ は座標変換に対応し，対象とする 3 次元変数を元の座標系から目的の座標系へと変換するものである．なお，一般には複数の座標変換が階層的に構造化されている（木構造である）が，それらを適切な順序で「直列化」すれば，このような直線的な列を見つけることがつねに可能である点に注意されたい．

このような座標変換を用いることで，$\boldsymbol{x}'$ を以下のように計算できる．

$$\boldsymbol{x}' = \boldsymbol{t}_{s-1}(\boldsymbol{t}_{s-2}(\cdots(\boldsymbol{t}_1(\boldsymbol{t}_0(\boldsymbol{x})))\cdots)) \equiv \boldsymbol{t}(\boldsymbol{x}).$$

ここで，$\boldsymbol{t}$ はすべての座標変換を合成したものとして定められる．以下では，$y_{k,i}$ によって $\boldsymbol{y}_k$ の第 i 要素を，$t_{k,i}$ によって $\boldsymbol{t}_k$ の第 i 要素を示す．すなわち，次式が成立する．

$$\begin{aligned}\boldsymbol{y}_{k+1} &= (y_{k+1,1}, y_{k+1,2}, \ldots, y_{k+1,n}) \\ &= (t_{k,1}(\boldsymbol{y}_k), t_{k,2}(\boldsymbol{y}_k), \ldots, t_{k,n}(\boldsymbol{y}_k)) \\ &= \boldsymbol{t}_k(\boldsymbol{y}_k).\end{aligned}$$

5.2 処理技法

幾何制約は 3 次元変数の世界座標を用いることで評価される．これは，その誤差関数が $e(\boldsymbol{x}')$ として定義されることを意味する．合成された座標変換を用いることで，これは

$$e(\boldsymbol{x}') = e(\boldsymbol{t}(\boldsymbol{x}))$$

として計算できる．その実現においては，実際に利用される変数に対して，必要な座標変換のみを適用することで効率化が可能である．

さらに，$e(\boldsymbol{t}(\boldsymbol{x}))$ の勾配

$$\nabla e(\boldsymbol{t}(\boldsymbol{x})) = \left(\frac{\partial e(\boldsymbol{t}(\boldsymbol{x}))}{\partial x_1}, \ldots, \frac{\partial e(\boldsymbol{t}(\boldsymbol{x}))}{\partial x_n}\right)$$

を計算する必要がある．基本的に，偏微分 $\partial e(\boldsymbol{t}(\boldsymbol{x}))/\partial x_i$ は，連鎖律を繰り返すことで原始的な式へ分解することが可能である．しかし，単純に連鎖律を適用する方法は，大量の式を生成してしまうことになるため，避ける必要がある．

以下では，上述の偏微分の分解を効率的に処理するための技法を与える．この技法は，連鎖律を適切に整理することで，不要な計算を行わないようにするものである．まず，$\partial e(\boldsymbol{t}(\boldsymbol{x}))/\partial x_i$ を以下のように分解する．

$$\begin{aligned}\frac{\partial e(\boldsymbol{t}(\boldsymbol{x}))}{\partial x_i} &= \sum_{j'} \frac{\partial e(\boldsymbol{x}')}{\partial x'_{j'}} \frac{\partial t_{s-1,j'}(\boldsymbol{y}_{s-1})}{\partial x_i} \\ &= \sum_{j'} \frac{\partial e(\boldsymbol{x}')}{\partial x'_{j'}} \sum_{j_{s-1}} \frac{\partial t_{s-1,j'}(\boldsymbol{y}_{s-1})}{\partial y_{s-1,j_{s-1}}} \\ &\qquad \times \frac{\partial t_{s-2,j_{s-1}}(\boldsymbol{y}_{s-2})}{\partial x_i} \\ &= \sum_{j_{s-1}} \left\{ \sum_{j'} \frac{\partial e(\boldsymbol{x}')}{\partial x'_{j'}} \frac{\partial t_{s-1,j'}(\boldsymbol{y}_{s-1})}{\partial y_{s-1,j_{s-1}}} \right\} \\ &\qquad \times \frac{\partial t_{s-2,j_{s-1}}(\boldsymbol{y}_{s-2})}{\partial x_i} \\ &= \sum_{j_{s-1}} \frac{\partial e(\boldsymbol{x}')}{\partial y_{s-1,j_{s-1}}} \frac{\partial t_{s-2,j_{s-1}}(\boldsymbol{y}_{s-2})}{\partial x_i}.\end{aligned}$$

ここで，$\partial e(\boldsymbol{x}')/\partial x'_{j'}$ が幾何制約の定義によって与えられており，また $\partial t_{s-1,j'}(\boldsymbol{y}_{s-1})/\partial y_{s-1,j_{s-1}}$ も 1 つの座標変換 $\boldsymbol{t}_{s-1}$ の勾配に含まれる偏微分であることに注意されたい．したがって，$\partial e(\boldsymbol{x}')/\partial y_{s-1,j_{s-1}}$ が得られる．また，この処理を繰り返すことで，それぞれの k に対して

$$\frac{\partial e(\boldsymbol{t}(\boldsymbol{x}))}{\partial x_i} = \sum_{j_k} \frac{\partial e(\boldsymbol{x}')}{\partial y_{k,j_k}} \frac{\partial t_{k-1,j_k}(\boldsymbol{y}_{k-1})}{\partial x_i}$$

を計算でき，最終的に

$$\frac{\partial e(\boldsymbol{t}(\boldsymbol{x}))}{\partial x_i} = \sum_{j_1} \frac{\partial e(\boldsymbol{x}')}{\partial y_{1,j_1}} \frac{\partial t_{0,j_1}(\boldsymbol{x})}{\partial x_i}$$

が得られる．ここで，$\partial t_{0,j_1}(\boldsymbol{x})/\partial x_i$ は $\boldsymbol{t}_0$ の勾配の成分である．したがって，$\partial e(\boldsymbol{t}(\boldsymbol{x}))/\partial x_i$ が決定される．

さらに，$\partial e(\boldsymbol{x}')/\partial y_{k,j_k}$ を実際に計算する回数を削減することが可能である．上述の計算から以下の観察が得られる．

- 各変数 $x_{j'}$ に対して，$\partial e(\boldsymbol{x}')/\partial x'_{j'} \neq 0$ であるのは，変数 $x_{j'}$ が制約の評価に実際に必要な場合に限られる．
- $\boldsymbol{t}_k$ に対応する座標系に変数 x_i が由来する場合（すなわち，x_i がその局所座標であるか，または座標変換のパラメータであるかの場合），$y_{k,i} = x_i$ であり，$\partial t_{k,j}(\boldsymbol{y}_k)/\partial x_i$ が計算可能である．したがって，$\partial e(\boldsymbol{x}')/\partial x_i$ はただちに計算可能である．

これらの観察により，$\partial e(\boldsymbol{x}')/\partial y_{k,j}$ を次のステップに引き渡す必要があるのは，x_j が制約の評価に必要な座標に対応し，かつ，それが由来する座標系に処理がまだ到達していない場合のみである．また，上記の処理は各点について個別に進めることが可能であることから，再帰呼び出しの際に，点の各成分に対応する 3 つの偏微分 $\partial e(\boldsymbol{x}')/\partial y_{k,j}$ を引き渡すような 1 本道の再帰関数として上述の処理を実装できることが分かる．

6. 実　　装

本研究では，本提案手法を制約解消系 Chorus3D として実装した．Chorus3D は，本研究者が以前に開発した 2 次元幾何制約解消系 Chorus [15),16)] を 3 次元に

拡張したものである．Chorus3D は C++のクラスライブラリとして実装され，さらに Java のクラスライブラリとして利用するためのネイティブメソッドインタフェースを提供している．

Chorus3D を利用するプログラマは，そのモジュール機構を用いることで，新しい種類の算術制約（たとえば Euclid 幾何制約）を追加できる．これは，その誤差関数を評価するメソッドを備えた新しい制約のクラスを実装することで行う．また，プログラマは，新しい種類の非算術（擬似）制約（たとえばグラフ配置制約）を導入できる．これは，与えられた制約の集合のための「集約的」な誤差関数を計算する新しい評価モジュールを開発することで行う．

現在の実装では，線形等式・不等式制約と，変数の値を繰り返し更新する edit 制約，変数の値を固定する stay 制約に加えて，平行，垂直，距離の一定などの関係を与える Euclid 幾何制約，スプリングモデル[17]に基づくグラフ配置制約を提供している．線形等式・不等式制約は，1 次元変数（3 次元変数の要素を含む）のみを参照でき，一方，edit および stay 制約は 1 次元変数または 3 次元変数を参照できる．Euclid 幾何制約は通常，3 次元変数を参照するが，角度や距離を表す 1 次元変数を必要とすることもある．グラフ配置制約はグラフのエッジを表現し，結び付けるべきノードの位置として 2 つの 3 次元変数を参照する．なお，すでに述べたように，1 つの制約が，異なる座標系に属する 3 次元変数を参照することが可能である．

Chorus3D のアプリケーションプログラミングインタフェースは，Chorus と同様，最近の線形制約解消系である Cassowary[3] との互換性に配慮して設計されている．Cassowary や Chrous と同様に，Chorus3D を利用するプログラマは，変数と制約をオブジェクトとして生成し，制約オブジェクトを制約解消系オブジェクトに対して追加・削除することで処理を記述する．これに加えて，Chorus3D では，座標変換をオブジェクトとして扱い，それらを階層的に配列するためのインタフェースを提供している．

7. 適 用 例

本章では，Chorus3D 制約解消系を適用することで，幾何制約を対話型 3 次元アプリケーションに導入する 3 つの例を与える．すべての例は Java で実装され，Chorus3D のネイティブメソッドインタフェースを利用し，グラフィクスプログラミングインタフェースとしては Java 3D を採用している．また，以下では，これらの例における制約解消に要した計算時間を与え，最後の例ではその制約プログラムを示す．

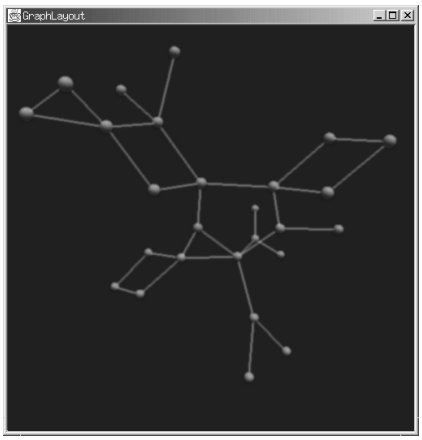

図 1 一般のグラフ構造の 3 次元幾何配置
Fig. 1 A 3D geometric layout of a general graph structure.

7.1 グラフ配置

最初の例は，**図 1** に示すように，一般のグラフ構造を持つ点の集合を 3 次元空間に配置するアプリケーションである．このアプリケーションでは，ユーザがグラフノードをマウスでドラッグすることができる☆．採用しているグラフ配置手法は，スプリングモデル[17]を 3 次元に拡張したものである．この種の 3 次元グラフ配置手法は情報視覚化に有効であり，既存のシステム[24]で実際に採用されている．

このグラフ配置の制約系は，26 個の 3 次元変数（78 個の実数値の変数）と，31 個のグラフ配置制約，3 個の線形等式制約（3 次元変数の 1 つを原点に固定）からなる．これを Linux 2.2.16 で動作する 866 MHz の Pentium III 上で実行した場合，Chorus3D で最初の配置を求めるのに 456 ミリ秒を要した．また，ユーザがグラフノードをドラッグした場合の制約解消は，通常 250 ミリ秒以内に処理が完了した（ただし，ドラッグ操作が急激であるなどの場合に，この時間を超えることがある）．

7.2 制約付きドラッグ

第 2 の例は，他のオブジェクトの表面に制約されているオブジェクトをユーザがドラッグできるアプリケーションである．**図 2** に示されるこのアプリケーションでは，小さい球体のオブジェクトが，大きいワイヤフレームの球体のオブジェクトの表面に制約されている．ここでは強さ `strong` の Euclid 幾何制約が

☆ 次の制約付きドラッグの例と異なり，このマウス操作は単純に Java 3D の `PickMouseBehavior` クラスを用いることで実装されている．

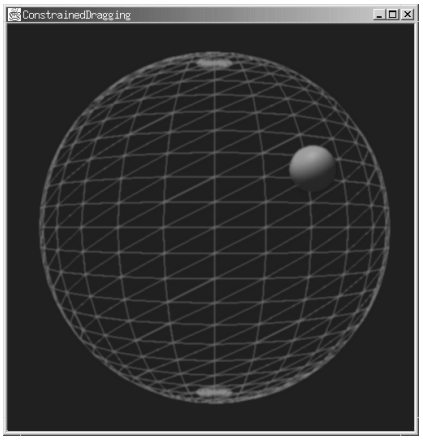

図 2 球面上に制約されているオブジェクトのドラッグ
Fig. 2 Dragging an object constrained to be on a sphere.

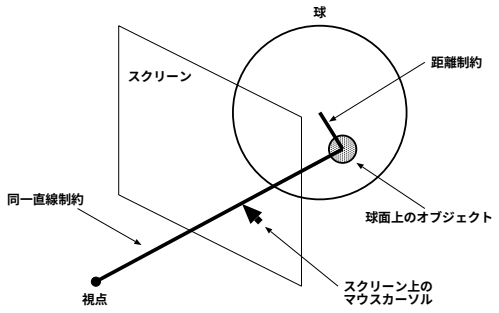

図 3 制約付きドラッグの実装
Fig. 3 Implementation of constrained dragging.

宣言され，これらのオブジェクトの中心間の距離が一定となるように指定されている．ユーザが小さいオブジェクトをマウスでドラッグしようとすると，アプリケーションは，**図 3** に示されるように，視点と，マウスカーソルの 3 次元の位置（スクリーン上に存在する），ドラッグされるオブジェクトの中心を同一直線上に並べる強さ `medium` の Euclid 幾何制約を追加する．この同一直線制約によって，マウスの動きが，ドラッグされるオブジェクトの位置に反映される．同一直線制約が最初の Euclid 幾何制約よりも弱いために，ユーザは小さいオブジェクトを大きいオブジェクトの外側へドラッグすることができない．

このアプリケーションは最初に，2 つの球体の中心を表現する 2 個の 3 次元変数を用いて，それらの間の距離を一定とする Euclid 幾何制約を 1 個宣言する．制約解消系はこれを最初の例と同じコンピュータ上で 1 ミリ秒で解消した．ユーザが小さいオブジェクトをドラッグしようとする場合，1 個の Euclid 幾何制約と，視点とマウスの位置のための 2 個の edit 制約が追加される．制約解消系はこの制約系の処理を通常 2 ミリ秒以内で完了した．

7.3 インバースキネマティクス

最後の例は，制約を利用することで，インバースキネマティクスを仮想的なロボットアームに適用するものである．これまでの例と異なり，この例では座標変換を利用することで制約系を表現している．

図 4 (a) に示されるように，ロボットアームは，土台，肩，上腕，前腕という 4 つの部品からなる．インバースキネマティクスのための制約解消は，可能ならば，手（前腕の先端）が，目標となるオブジェクトの場所に位置するようにし，それが不可能な場合，手が目標に最大限に近付くようにする．図 4 (b)〜(f) はロボットアームの動きを示す．図 4 (b)〜(e) において，手は適切な関節の角度を用いることで目標と同じ場所に位置している．一方，図 4 (f) において，手は目標に届かないため，代わりに目標に向かって伸びている．

図 5 はロボットアームアプリケーションで用いられる制約プログラムを示す．制約解消系 `s` を構築した後，6 個の座標変換 `shldrTTfm`，`shldrRTfm`，`uarmTTfm`，`uarmRTfm`，`farmTTfm`，`farmRTfm` を生成する．ここで，回転変換 `shldrRTfm`，`uarmRTfm`，`farmRTfm` の回転角度パラメータは，制約解消系によって変化させられうる変数として実際に働くものである．次に，手の位置を表す 3 次元変数 `handPos` を生成し，選好的な edit 制約 `editHandPos` を用いることで，手に対して目標の位置を指示する．最後に，制約解消系を実行し，回転変換の望ましい角度 `shldrAngle`，`uarmAngle`，`farmAngle` を得る．これらの角度は Java 3D ライブラリに引き渡され，適切な形状に整えられたロボットアームがレンダリングされることになる．

このプログラムが生成する制約系は，3 個の平行移動および 3 個の回転変換，1 個の明示的な 3 次元変数，座標変換のための 6 個の 3 次元変数および 3 個の 1 次元変数，1 個の edit 制約からなる．制約解消系は最初の解を得るのに 18 ミリ秒を要し，フレーム更新のための新しい解の計算を通常 10 ミリ秒以内に完了した．

8. 議　　論

一般に，本手法で扱っているような非線形幾何制約の役割は，2 次元インタフェースにおいてよりも，3 次元アプリケーションにおいて重大であると考えられる．特に重要な点として，2 次元インタフェースではオブジェクトの回転が利用されることが少ないのに対して，3 次元グラフィクスでは必要とされることが多い点があげられる．その主な理由として，3 次元空間においては通常，すべての「水平」方向が平等に扱われ

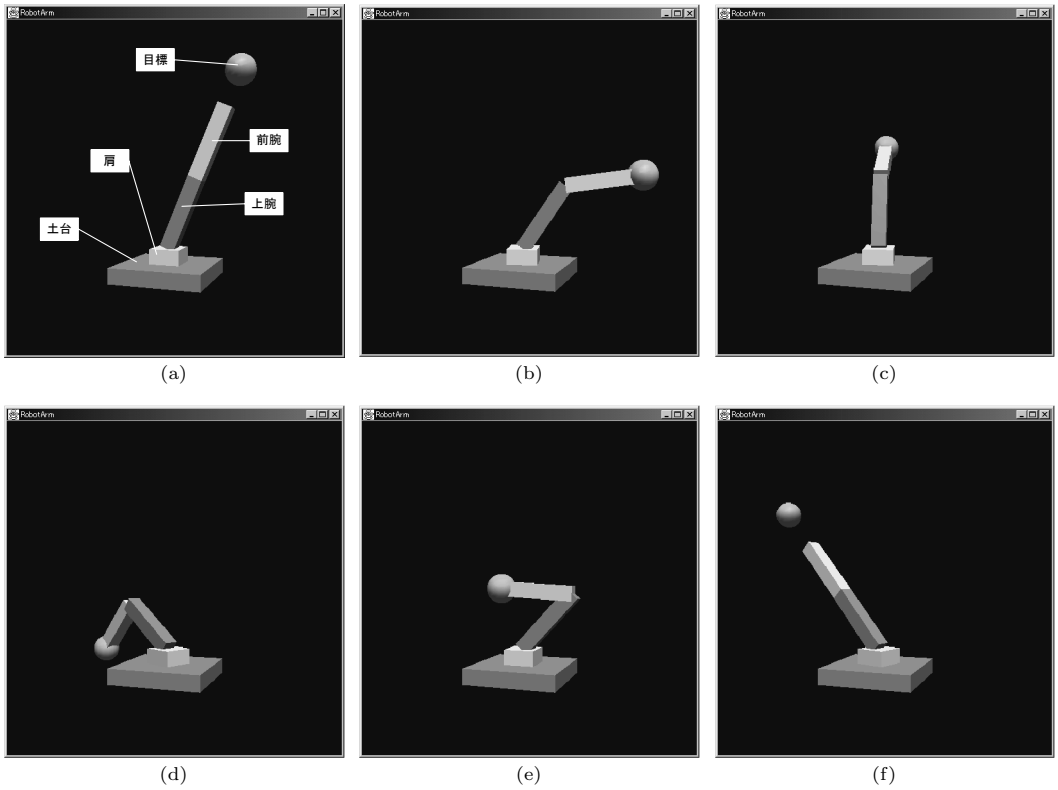

図 4 インバースキネマティクスを処理するロボットアームアプリケーション
Fig. 4 A robot arm application which performs inverse kinematics.

る（「垂直」方向との区別はなされる）ことが指摘できる．このために，本手法のような非線形制約解消系が 3 次元アプリケーションには適しているといえる．さらに，本手法のように制約解消系において座標変換を明示的にサポートすることが，現実のアプリケーションにおいてオブジェクトの回転処理のために利用されることが多いため，有効である．

本手法における制約解消アルゴリズムには 2 つの制限がある．1 つは，制約系が大きい場合の解消の速度に関するものであり，もう 1 つは，選好制約が含まれる場合の解の精度に関するものである．これらの制限は主に，必須制約に加えて，複数レベルの選好制約を処理しなければならないこと，すなわち，制約階層の実現に起因するものである．多くの数値最適化手法が数理計画法の分野で提案されている[2),6)]が，それらのほとんどは選好制約に対応していない．このような制限を緩和するために，複数レベルの選好制約を処理する，より洗練された制約解消アルゴリズムを開発することは今後の課題である．

本研究では，C++または Java プログラムで利用可能なクラスライブラリとして制約解消系 Chorus3D を実装した．しかし，より高水準なオーサリングツールを開発することが，3 次元グラフィクスの構築のためには有用である．その 1 つのアプローチとして，VRML[4)]を拡張して幾何制約をサポートすることが考えられる．標準的な VRML は，複雑な配置や振舞いの実現のために，Java または JavaScript で書かれたスクリプトを必要とする．対照的に，VRML に制約を導入すれば，そのような追加のスクリプトなしに，より広範囲のアプリケーションを構築できるようになると考えられる．

9. おわりに

本論文では，対話型 3 次元アプリケーションに幾何制約を統合するための制約解消法を提案した．本手法では，幾何制約解消における座標変換の処理を可能にすることで，シーングラフをともなった 3 次元グラフィクスに対応した．本手法は制約解消系 Chorus3D として実装されており，これによってプログラマは，幾何配置や，制約付きドラッグ，インバースキネマティクスなどの種々の目的で幾何制約を利用できる．

今後の課題として，本手法の有効性をさらに実証す

```
// 制約解消系
s = new C3Solver();
// 肩のための平行移動変換：(0, .1, 0) に固定
shldrTTfm = new C3TranslateTransform(
    new C3Domain3D(0, .1, 0));
// shldrTTfm は世界座標系を親とする
s.add(shldrTTfm);
// 肩のための回転変換：軸を (0, 1, 0) に固定;
// 角度の範囲を [−10000, 10000] とする
shldrRTfm = new C3RotateTransform(
    new C3Domain3D(0, 1, 0), new C3Domain(-10000, 10000));
// shldrRTfm は shldrTTfm を親とする
s.add(shldrRTfm, shldrTTfm);
// 上腕のための平行移動変換：(0, .1, 0) に固定
uarmTTfm = new C3TranslateTransform(
    new C3Domain3D(0, .1, 0));
// uarmTTfm は shldrRTfm を親とする
s.add(uarmTTfm, shldrRTfm);
// 上腕のための回転変換：軸を (0, 0, 1) に固定;
// 角度の範囲を [−1.57, 1.57] とする
uarmRTfm = new C3RotateTransform(
    new C3Domain3D(0, 0, 1), new C3Domain(-1.57, 1.57));
// uarmRTfm は uarmTTfm を親とする
s.add(uarmRTfm, uarmTTfm);
// 前腕のための平行移動変換：(0, .5, 0) に固定
farmTTfm = new C3TranslateTransform(
    new C3Domain3D(0, .5, 0));
// farmTTfm は uarmRTfm を親とする
s.add(farmTTfm, uarmRTfm);
// 前腕のための回転変換：軸を (0, 0, 1) に固定;
// 角度の範囲を [−3.14, 0] とする
farmRTfm = new C3RotateTransform(
    new C3Domain3D(0, 0, 1), new C3Domain(-3.14, 0));
// farmRTfm は farmTTfm を親とする
s.add(farmRTfm, farmTTfm);
// 手の位置を表す変数：座標系を farmRTfm とする;
// 位置を (0, .5, 0) とする
handPos = new C3Variable3D(
    farmRTfm, new C3Domain3D(0, .5, 0));
// 手の位置のための強さ medium の edit 制約
editHandPos = new C3EditConstraint(handPos, C3.MEDIUM);
s.add(editHandPos);
// 手が目標と同じ位置になるように指示
editHandPos.set(getTargetWorldCoordinates());
// 制約系を解消
s.solve();
// 解を得る
double shldrAngle = shldrRTfm.rotationAngle().value();
double uarmAngle = uarmRTfm.rotationAngle().value();
double farmAngle = farmRTfm.rotationAngle().value();
```

図 5 ロボットアームアプリケーションのための制約プログラム
Fig. 5 Constraint program for the robot arm application.

るために，他の種類の幾何制約を実現する予定である．特に，重複回避制約[15] を実装し，グラフィカルオブジェクトの衝突解決のために Chorus3D を利用できるようにすることを計画している．また，本手法における制約解消のスケーラビリティと精度を改善することや，本手法を採用する 3 次元グラフィクスオーサリングツールを開発することも今後の課題である．

参 考 文 献

1) Badler, N.I., Phillips, C.B. and Webber, B.L.: *Simulating Humans: Computer Graphics, Animation, and Control*, Oxford University Press (1993).
2) Bertsekas, D.P.: *Nonlinear Programming*, 2nd edition, Athena Scientific (1999).
3) Borning, A., Marriott, K., Stuckey, P. and Xiao, Y.: Solving Linear Arithmetic Constraints for User Interface Applications, *Proc. ACM UIST*, pp.87–96 (1997).
4) Carey, R., Bell, G. and Marrin, C.: The Virtual Reality Modeling Language (VRML97), ISO/IEC 14772-1:1997, VRML Consortium (1997).
5) Diehl, S. and Keller, J.: VRML with Constraints, *Proc. Web3D-VRML*, pp.81–86, ACM (2000).
6) Fletcher, R.: *Practical Methods of Optimization*, 2nd edition, Wiley (1987).
7) Freeman-Benson, B.N., Maloney, J. and Borning, A.: An Incremental Constraint Solver, *Comm. ACM*, Vol.33, No.1, pp.54–63 (1990).
8) Fudos, I.: Constraint Solving for Computer Aided Design, Ph.D. Thesis, Dept. Comput. Sci., Purdue Univ. (1995).
9) Gleicher, M.: A Graphical Toolkit Based on Differential Constraints, *Proc. ACM UIST*, pp.109–120 (1993).
10) Gleicher, M.: A Differential Approach to Graphical Manipulation (Ph.D. Thesis), Tech. Rep. CMU-CS-94-217, Sch. Comput. Sci., Carnegie Mellon Univ. (1994).
11) 服部隆志：編集操作におけるマクロと制約の統合，インタラクティブシステムとソフトウェア IV—日本ソフトウェア科学会 WISS'96，レクチャーノート/ソフトウェア学，Vol.16, pp.41–49, 近代科学社 (1996).
12) Herrera, F., Lozano, M. and Verdegay, J.L.: Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis, *Artif. Intell. Rev.*, Vol.12, No.4, pp.265–319 (1998).
13) Heydon, A. and Nelson, G.: The Juno-2 Constraint-Based Drawing Editor, Res. Rep. 131a, Digital Systems Research Center (1994).
14) Hosobe, H.: A Scalable Linear Constraint Solver for User Interface Construction, *Principles and Practice of Constraint Programming—CP2000*, LNCS, Vol.1894, pp.218–232, Springer (2000).
15) Hosobe, H.: A Modular Geometric Constraint Solver for User Interface Applications, *Proc. ACM UIST*, pp.91–100 (2001).
16) 細部博史：対話型インタフェースのための幾何制約解消の枠組み，情報処理学会論文誌，Vol.42, No.6, pp.1424–1434 (2001).
17) Kamada, T. and Kawai, S.: An Algorithm for Drawing General Undirected Graphs, *Inf. Pro-*

cess. Lett., Vol.31, No.1, pp.7–15 (1989).
18) 北野宏明（編）：遺伝的アルゴリズム，産業図書 (1993).
19) Kramer, G.A.: A Geometric Constraint Engine, *Artif. Intell.*, Vol.58, No.1–3, pp.327–360 (1992).
20) Marriott, K., Chok, S.S. and Finlay, A.: A Tableau Based Constraint Solving Toolkit for Interactive Graphical Applications, *Principles and Practice of Constraint Programming—CP98*, LNCS, Vol.1520, pp.340–354, Springer (1998).
21) Owen, J.C.: Algebraic Solution for Geometry from Dimensional Constraints, *Proc. ACM Solid Modeling*, pp.397–407 (1991).
22) Sannella, M.: SkyBlue: A Multi-Way Local Propagation Constraint Solver for User Interface Construction, *Proc. ACM UIST*, pp.137–146 (1994).
23) Suzuki, T. and Tokuda, T.: The DeltaUp Constraint Solver: Minimizing the Number of Method Selections in DeltaBlue, *Softw. Pract. Exper.*, Vol.31, No.14, pp.1351–1361 (2001).
24) Takahashi, S.: Visualizing Constraints in Visualization Rules, *Proc. CP2000 Workshop on Analysis and Visualization of Constraint Programs and Solvers* (2000).
25) Vander Zanden, B.: An Incremental Algorithm for Satisfying Hierarchies of Multi-Way Dataflow Constraints, *ACM Trans. Prog. Lang. Syst.*, Vol.18, No.1, pp.30–72 (1996).
26) Zhao, J. and Badler, N.I.: Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures, *ACM Trans. Gr.*, Vol.13, No.4, pp.313–336 (1994).

(平成 14 年 6 月 10 日受付)
(平成 14 年 12 月 3 日採録)

細部 博史（正会員）

1969 年生．1993 年東京大学理学部情報科学科卒業．1995 年同大学大学院理学系研究科情報科学専攻修士課程修了．1998 年同専攻博士課程修了．博士（理学）．日本学術振興会特別研究員-PD，文部省学術情報センター助手を経て，2000 年より国立情報学研究所助手．制約プログラミング，ユーザインタフェース，対話型グラフィクス，情報視覚化等に興味を持つ．日本ソフトウェア科学会，ACM 各会員．