博士論文

# 柔軟性可変な脊椎構造を有する
# 多自由度全身行動ロボットシステム

2001 年 12 月

水内 郁夫

# 目次

# 第 1 章

# 序論

## 1.1 研究の背景と目的

ヒューマノイドなどの全身型・人間型のロボットの開発・研究が盛んである [12, 24] が，人間に比べ構造・動きの硬さが感じられる場面が多い．ロボットは今後ますます人と接することが想定される環境で活動するようになると予想でき，そこでは構造・動きの柔かさは非常に重要になる．本研究では，全身型ロボットに柔軟性を組み込むことを提案する．特に，人間の脊椎構造に着目し，柔軟な脊椎構造を持つ全身型ロボットに取り組んでゆく．

人間と接する環境を想定するとき，柔軟性は安全性の面から重要になるが，もう一つ多様性というキーワードも重要になる．動作の多様性と環境の多様性である．これまでのロボットは全ての対応すべき状況に応じたセンサ・アクチュエータ・制御モデルを持つという立場に立っていた．しかし，全ての起こりうる状況を想定することはできない．従来のロボットは，想定されていない状況への対応は困難であった．本研究では，この問題の解決への貢献を試みる．ロボットの身体構造として，従来のロボットは種類・量ともに必要な自由度・必要なセンサを備えていたが，本研究では，多種・多数のセンサ・アクチュエータを搭載し，センサやアクチュエータがいくつか不調であったり故障したりしていても，行動を行うことができるような枠組みの構築を目指す．

本研究の目的は，柔軟な脊椎を持つ全身型ロボットの，

- 脊椎構造 (体幹) のあり方
- 脊椎構造の駆動系の構成法
- 全身行動の生成法，動作・行動の制御法・拡張法
- 全身行動のためのシステム構成法

に関し，課題を整理し，解決への道筋を示すことである．

産業用ロボット，サービスロボット，ヒューマノイドと新化してきたロボットが，いずれも金属のかたい機械構造により作られてきた．こうしたかたい構造による人間との親和性や安全性の面での違和感を無くすためには，ロボットの身体構造を根本的に見なおすことは，大きなブレークスルーにつながる可能性がある．本研究は，こうした “ロボットの身体構造 (作り方) の根本的な見直し” ができる (あるいは，すべき) 状況が訪れつつあるという視点から，萌芽的テーマに多面的に取り組んだものであると言える．

## 1.2 本論文の構成

本論文の構成は全 7 章から成る．**第 1 章 序論** (本章) では，研究の背景と目的に関し述べた．**第 2 章 柔軟性可変な脊椎構造を有する多自由度全身型ロボット**では，多自由度全身型ロボットが柔軟性可変な脊椎構造を持つと，どのようなトピックが生まれてくるのかに関し考察する．人間の脊椎の構造と役割の分析，これまでに行われてきた関連研究の分析を行う．**第 3 章 脊椎構造の設計 — 基本姿勢と復元力を有する脊椎構造 —**では，脊椎構造を変形に関する変数と柔軟性調節に関する変数の観点に分けて捉え，試作と実験を行う．そして，人間の脊椎の構造を参考にした全身型ロボットのための脊椎構造の検討をし，それらをふまえた上での可変な柔軟性を持つ脊椎構造の設計に関し述べる．問題の本質はいかにして柔軟性と自由度が増加する構造を実現するかという点である．本研究では，基本姿勢と復元力というキーワードを挙げて，実際に全身型ロボットのための脊椎構造を設計・製作を行い，脊椎構造のあり方を論じる．**第 4 章 拮抗筋駆動による柔軟性を有する脊椎の姿勢制御**では，第 3 章で設計・製作した脊椎構造をいかに動かすかに関し研究する．人間に近い複雑性を有する構造の制御として，人間の制御の方針に近いと思われる方針をとり，ある程度の幾何学的制御量 (関節角度等) の制御法の提案と，センサフィードバックによるタスクの実現法を示し，その有効性を示した．**第 5 章 柔軟な脊椎を有するロボットの全身行動の実現**では，全身動作・全身行動の生成法およびその制御法を論じる． GA や NN と 3 種類のシミュレーション環境，モーションキャプチャの利用，姿勢データベースに基づく行動拡張の枠組みなどを示す．また，従来の全身型ロボットのソフトウェア環境と透過的な環境を用いて，様々な脊椎を利用した全身行動実験に関し述べる．**第 6 章 多自由度多センサシステムの設計と実現**では，多入力・多出力な全体システムをどのように設計するかに関し述べる．不完全性を前提として，自律的診断調節系，従来のヒューマノイドソフトウェアとの透過性を持ったソフトウェア環境などに関し述べる．多入力・多出力な全体システムの構成法としては，体内 LAN・ハードウェア抽象化層・オブジェクト指向の上位層からなる三層構造を基本としたシステムの構築法を示す．**第 7 章 結論**では，本研究のまとめをし結論を述べる．さらに今後の展望を述べる．

# 第 2 章

# 柔軟性可変な脊椎構造を有する多自由度全身型ロボット

本章では，全身型ロボットの身体構造はどうあるべきか，今後どうなってゆくべきかを考察する．全身型ロボットは人間の生活の場での活動が想定され，そこでは，人間に対する安全性・動作の多様性・環境の多様性といったキーワードが求められてくると考えられる．これに対する解として，本研究では柔かいボディ・柔かい動きの実現を目指す．特に人間の脊椎機構の効果に着目し，全身型ロボットにこれを組み込む．また，多様性へのアプローチとして，多種多数のセンサ・アクチュエータを搭載した多自由度型の複雑なロボットシステムの構成法として，全ての状況を想定するのは不可能であるという前提に立ち，不完全な状態を前提とするシステムが必要であることを指摘する．

## 2.1 全身型ロボットの身体構造

本論文では，胴体 (体幹， torso) の他に四肢 (またはそれ以上) に相当するリム (limb) を有するロボットで，これらのリムの一部または全てを用いて移動をはじめとするタスクを行うことを想定したロボットを指して全身型ロボットと呼ぶものとする．

全身型ロボットの特長に汎用性がある．自由度が多くセンサも多いロボットは，活動の目的が単一でなく様々なタスクを様々な環境でこなすことが期待される．今後こうしたロボットはますます活動の場を人間と共有するようになってゆくだろう．そこでは，人間に対する安全性と，動作の多様性，環境の多様性への対応が重要になる．ここでは， 安全性 と 多様性 という二つのキーワードをもとに全身型ロボットの身体構造に関し考察する．

### 2.1.1 硬さと柔らかさ

現在のヒューマノイドのボディは硬く，人間との衝突はあってはならないことと想定されている．動きも人間など動物の動きと比較すると硬さが感じられる．特に上半身の動きに関しては，何か制限されているような感覚を感じさせられることもある．本研究では，全身型ロボットの身体構造を根本的に変えてゆくことにチャレンジする．柔かいボディの全身型ロボットを作り，その動きを柔かくする．人間の持つ身体的な柔かさは様々な側面があるが，本研究では特に人間の脊椎機構の効果に着目し，この効果を全身型ロボットに持たせることを考えてゆく．人間の活動の場で活動するようになってゆくであろう全身型ロボットは安全性を備えることが不可欠である．柔軟性のある身体構造というのは，安全性を実現することを考えるときに一つの鍵となる．

### 2.1.2 動作・タスクの多様性

ロボットはますます人間の活動の場で活動するようになってゆくと述べたが，そこでのロボットに課されるタスクは，非常に多岐にわたる．家庭に入るロボットは家事のできるだけ多くの種類を行えることが求められるだろう．こうした汎用性が求められるからこそ，全身型ロボットがその解となりうるポテンシャルを有すると言える．全身型ロボットは，遂行すべきタスクも多様なものになり，ロボットがすべき動作も多様なものになるだろう．

多様な動作・タスクの実行を可能にするために重要になるのは，自由度の多さである．可動部が多いことや可動範囲が広いことは，実現可能なタスクが広がることにつながる．たとえ，少ない自由度でも実現可能な動作・タスクであっても，一般に自由度が多い方が効率良く短時間で行うことができる．本研究では，自由度の増加という観点を重要視したロボットの身体構造を考えてゆく．

### 2.1.3 環境の多様性

人間の活動の場というのは，複雑な環境でもある．特に家庭というのは非常に多様性の大きい環境である．従来のロボットは多様性に対応するために，想定される全ての状況に対応することができるセンサ・アクチュエータ・制御モデルを有するように設計されてきた．逆に言えば想定されていない状況に対応することは困難であった．

しかし，環境の多様性が増せば増すほど，全ての状況を想定することは不可能に近くなる．古くは産業用ロボットから現在のヒューマノイドまで，作業環境や条件をロボットのために整えるというやり方は，家庭や人間社会のように環境が複雑になってくると，破綻をきたすことが想像できる．本研究では，多様な環境 (と多様なタスク・動作) に対応できるために，多種多数のセンサ・アクチュエータを搭載し，多種多様なセンサ・アクチュエータを適応的に利用して，不完全な状態でも行動を行うことができるようなロボットシステムとしての枠組みをねらう．すなわち，センサ・アクチュエータの数・種類はできるだけ多くするが，システムとしては不完全性に対応できるという点を重視する．

### 2.1.4 柔軟性可変な脊椎を持つ全身型ロボット

2.1.1節， 2.1.2節， 2.1.3節に述べた要件を満たしたロボットへのアプローチとして，本研究では人間や脊椎動物の脊椎構造に着目した．全身型ロボットが柔軟性可変な脊椎を持つと以下のような特徴が生まれると考えられる．

### 柔かさ・しなり

衝突時の衝撃吸収を行うことができるようになる．また人間や環境との接触において安全性を実現することができる．柔らかさによって，衝突時に生じる力の最大値が小さくなることが期待できる．ロボット自身が壊れることを防ぐ意味もある．

また，脊椎の柔らかさは，動きの柔らかさにもつながる．物理的な柔らかさにより，動作のしなやかさを増すことができる．例えば投球動作やテニスのサービスのような運動の際に「しなり」が重要であると言われることがあるが，この「しなり」という動きは運動の根元から順に動かし先端がやや遅れるように動いてくることにより，手先の速度の最大値を「しなり」が無い場合に比べて大きくするような動きを指す場合が多い．また，鞭の「しなり」も同様に先端の速度の最大値が大きくなる動きを指すが，弾性を持つ構造が力を蓄積して，それを開放する時に瞬間的に大きな力が出ているとみなすこともできる．体の動きに「しなり」を使うというのも，体の弾性要素の材料的特性を積極的に利用していると言うこともできる．

### 状況に応じた柔軟性の調節

柔軟性を持つことにより，前項に指摘したような柔らかさ・しなりといった特長の効果を利用することができるが，脊椎構造は上体を支持する構造材の役割も担っている．したがって，構造を支える力が要求される場面では，柔軟性を調節して硬い状態になることができることが望ましい．

人間や動物の場合，筋肉による拮抗型の駆動系の構造になっている．そして，筋肉はそれ自体の剛性が可変である．そのため，硬くする必要があるときは，筋肉にグッと力をこめて剛性を上げることができる．

### 自由度の増加 (可動範囲の向上)

これにより，体を折らなければ作業できないような狭い場所での作業や，障害物を避けての作業など，制約条件の多い状況での作業を遂行する能力が向上する．自由度の増加による可動範囲の向上は，冗長性の向上でもある．つまり，脊椎の自由度は不可欠な自由度ではなく，脊椎を持たないロボットが同じタスクをするためには様々な問題が生じるというような種類の自由度である．

例えば， **Fig. 2.1** は，従来型のヒューマノイドが椅子に座った状態で床にあるものを拾おうとしている様子である．脊椎構造が無いロボットは，股関節の roll 軸回りの自由度を回転させなければ手先位置を床に近づけることができない．左図は股関節を可動範囲ギリギリまで回したところであるが，床の物に手が届いていない．

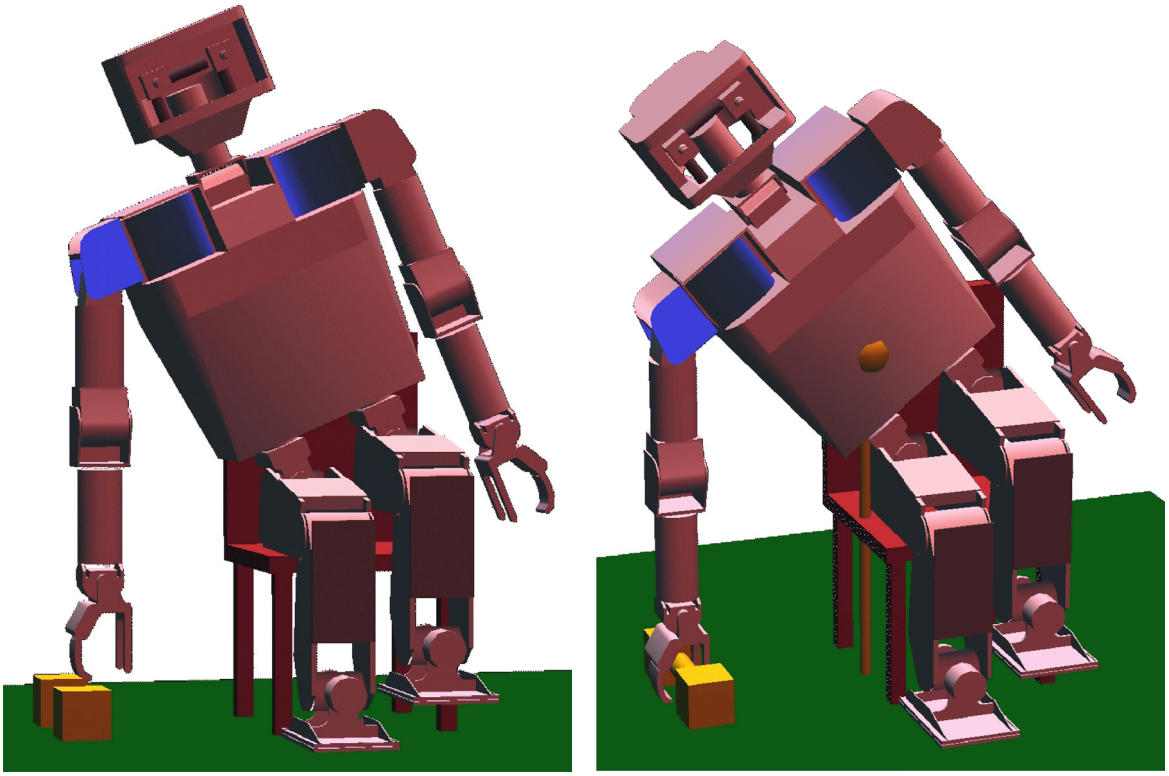


**Fig. 2.1** 従来型ヒューマノイドの可動範囲
Movable area of a conventional humanoid

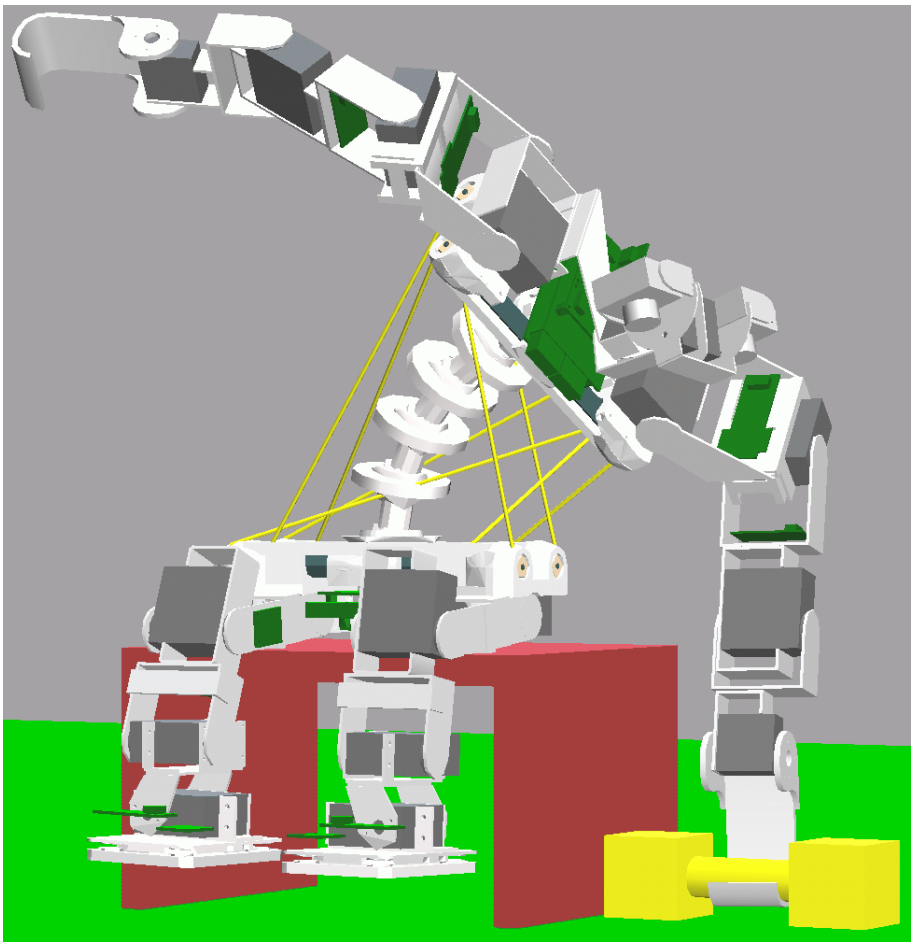


**Fig. 2.2** 可動な脊椎を持つ人間型ロボットの可動範囲
Movable area of a flexible spine humanoid

もし仮にもう少し股関節が広かった場合はどうなるだろうか．右図は床の物に手が届くところまで股関節を回転させた場合である．この場合は今度はロボットの重心位置が椅子からはみ出しそうになっている．このままではおそらく床に倒れてしまうだろう．

これに対し，脊椎を持つ人間型ロボットの場合どうなるかを **Fig. 2.2** に示す． **Fig. 2.1** のヒューマノイドは床に足裏が着く高さの椅子に座っていたが， **Fig. 2.2** のロボットは足裏が床に着かない高い椅子に座っている．しかし，それでも脊椎の自由度を使えば余裕を持って床の物まで手が届く．

もちろん，ヒューマノイドが椅子から立ち上がり横に移動してしゃがめば床の物を拾うことは可能であり，脊椎の自由度は不可欠な自由度ではないかも知れない．脊椎の自由度向上とはそういった種類のものである．ノンホロノミックビークルは直接横に移動する自由度を持っていなくても，横に移動できないわけではないが，横に移動する自由度を持っている全方位

ビークルは横への移動がしやすい．脊椎によりもたらされる自由度とは便利になる自由度なのである．

#### 人に与える印象

上半身のみを持つ人間型ロボットで全身動作・全身行動の研究や開発を行った経験がある人の意見で，脊椎があるとないでは，できる動作，与える印象が全く異るという意見を耳にしたことがある．

また， 3D Studio MAX などの CG アニメーションソフトウェアでは，人間の動作や行動の CG を作成できる．サンプル動作として，人間の動作をモーションキャプチャしたデータ等もついている．こうしたソフトウェアにおける人間の動作の表現形式はいくつかあるが，そのほとんどに脊椎の動き情報が入っている．脊椎の動き情報が無いと，自然な人間らしい動きを再現するのが難しくなるということの証左ということができるのではないだろうか．

### 2.2 人間の脊椎構造

生理学的知見 [35, 87] から，人間の脊柱の特徴を整理した．なお，脊椎という単語は，脊柱すなわち背骨全体という意味と，一つ一つの節 (椎骨) という意味の，二つの意味がある．本論文では， “脊椎” は “脊柱” と同じ意味で用いる．人間の脊椎の構造・役割を知る事は，ロボットにおける体幹の構造のひとつの理想像を議論するための土台となる．

#### 2.2.1 人間の脊柱の構造

脊柱は，腰から頭まで多数の節からなる．各節は，椎骨とその間に椎間板が挟まる形で構成される．各椎骨は上部から順に頸椎 (けいつい)・胸椎 (きょうつい)・腰椎 (ようつい) などに分類される．他にも，仙椎 (せんつい) と尾椎 (びつい) と呼ばれる椎骨もあるが，これらは骨盤 (腰骨) を形成する骨とみなすこともできる． (仙椎は成人になるころには結合して一個の骨になり，仙骨と呼ばれる．) 脊柱の節の数は，仙椎 (5 節)・尾椎 (3 〜 5 節) も含めると 32 〜 34 節となるが，主要な部分は，頸椎 7 節・胸椎 12 節・腰椎 5 節で合計 24 節である． **Fig. 2.3** は人間の背骨である． 4 の部分が頸椎， 3 が胸椎， 2 が腰椎にあたり， 1 の部分が骨盤を形成する仙椎・尾椎にあたる．

この 24 節からなる脊柱は，横から見ると S 字に湾曲している ( **Fig. 2.3** 左)．医学の分野ではこの S 字の湾曲は生理湾曲と言われ，湾曲の度合いが大きすぎても小さすぎても腰痛などの原因となる．また，この湾曲はそれに加わった長軸方向への圧力への抵抗 (抗力) を増加させる [35]．人間の場合は腰のすぐ上の腰椎部分 ( **Fig. 2.3** の 2) と首の辺りの頚椎 ( **Fig. 2.3** の

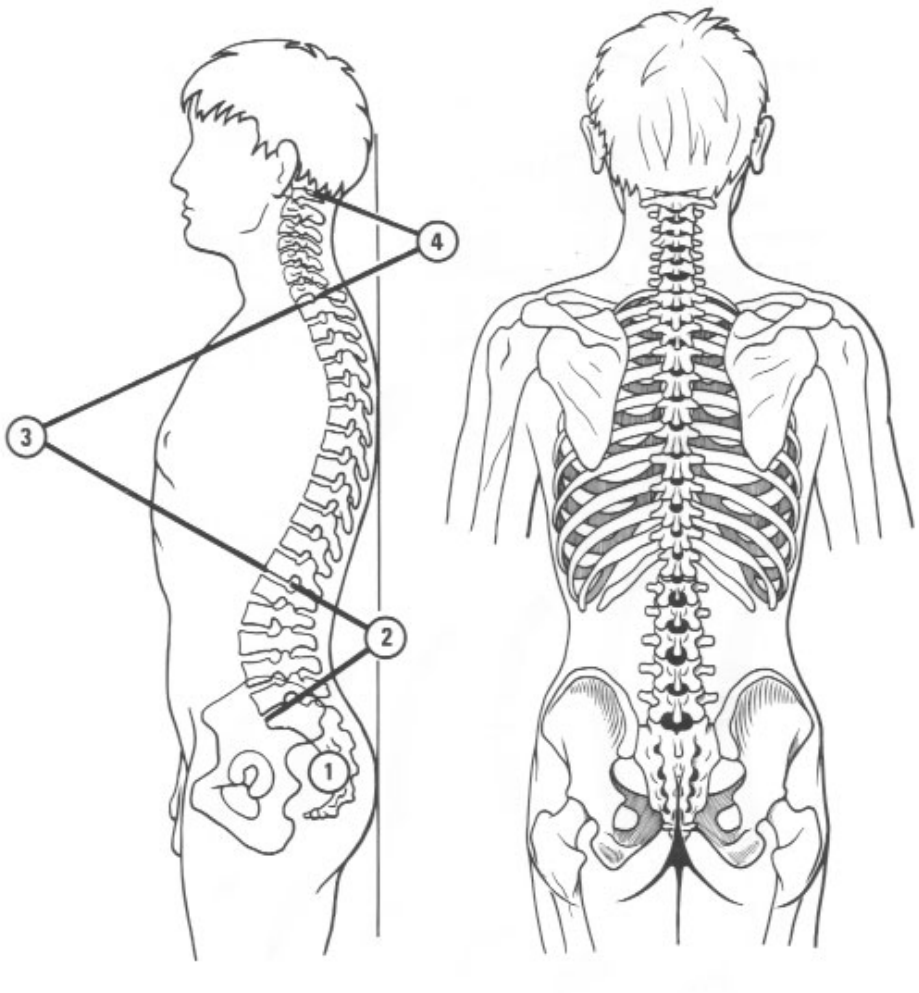



**Fig. 2.3** 人間の背骨 (「カパンディ関節の生理学 I 体幹・脊柱」 [35] より)
Human's backbone

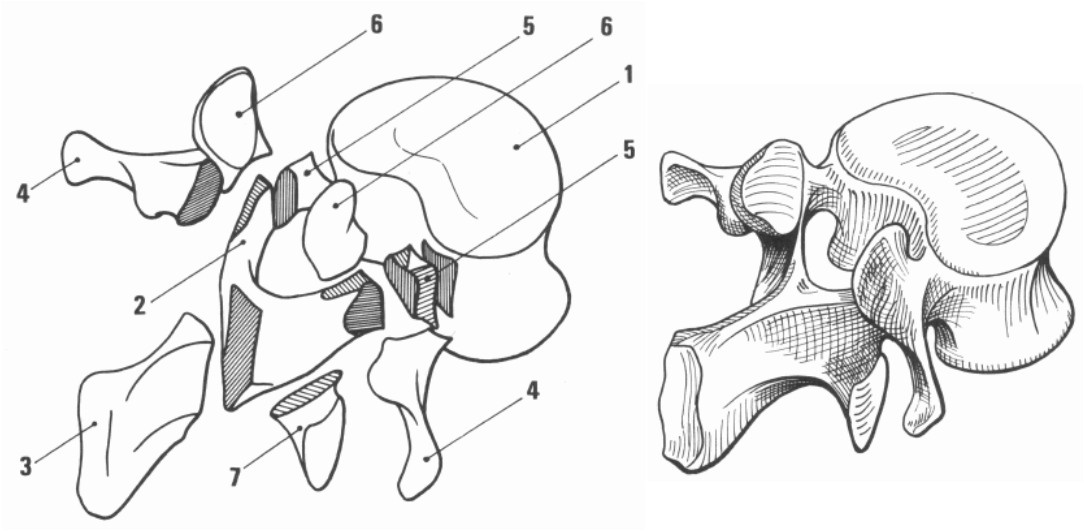



**Fig. 2.4** 人間の椎骨 (「カパンディ関節の生理学 I 体幹・脊柱」より)
A vertebra of human

4) は前方に凸で，中間部の胸椎 ( **Fig. 2.3** の 3) は後に凸となっている．このため，腰椎部分および頚椎部分は胴体の中心近くを通り，胸椎部分は後部 (背中に近い部分) を通っている．胸椎は胴の中心部を通らない代わりに，胸椎 12 節の各節に肋骨が接続されている．

各椎骨は，上下面は緩い凹面状で臼のような形状をしている ( **Fig. 2.4** の 1，**Fig. 2.6** 参照)．各椎骨と椎骨の間には椎間板が存在する．椎間板は，　髄核と呼ばれる弾力性・吸水性のある球 ( **Fig. 2.6** の 9) と，その球を取り囲む形で線維輪と呼ばれる線維状のもの ( **Fig. 2.6** の 8) が入っている．髄核は圧縮された状態で臼と臼の間にあり，上下の椎骨間の距離を離す方向に常に力を発している．そのまわりの線維輪は，上下の椎骨の臼状面に接続され引っ張られた状態になっている．これらの力のバランスにより，各節は常に初期状態に戻ろうとする力 が働いている．

典型的な椎骨の形状は **Fig. 2.4** のようになっている．椎骨の髄核と椎間板が接している部分 (椎体．図の 1 の部分) より後 (背中側) には三方向に突起があり，後突起 (3)・横突起 (4) と呼ばれる．それぞれの突起は，靱帯・筋肉が接続されている．また，上向きに二つ (上関節突起．**Fig. 2.4** の 6) と下向に一つ (下関節突起．**Fig. 2.4** の 7) の突起は，上下の椎骨の間の相対的運動を制限する働きを持っており，特に後方 (背中側) に傾いた場合はある程度の傾きで上の椎骨の下関節突起が下の椎骨の上関節突起の間に入る形でロックし，それ以上傾かないようになると同時に，水平面内の回転も制限されるようになる ( **Fig. 2.5** )．この特徴は，腰椎などの下部の椎骨に特に見られ，胸椎・頸椎においてはこうした制限が少なくなっている．

脊柱の自由度 (変形のパラメータ数) を明確にするのは容易ではない．一節あたりの自由

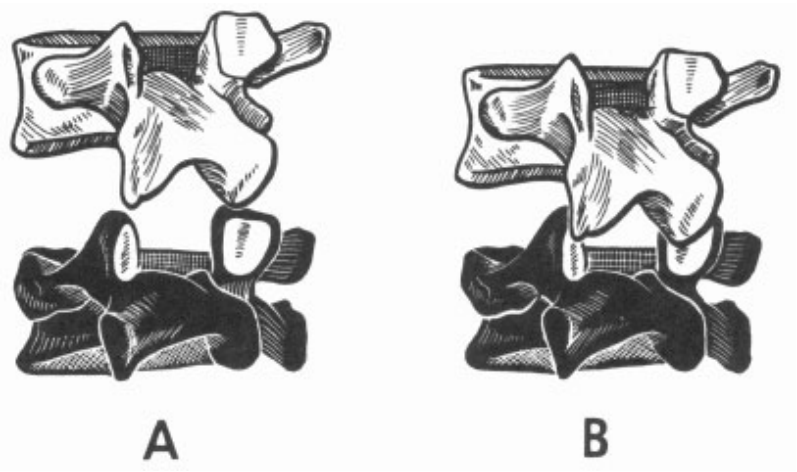



**Fig. 2.5** 二節の椎骨間の動作制限 (「カパンディ関節の生理学 I 体幹・脊柱」より)
Relation between two vertebrae

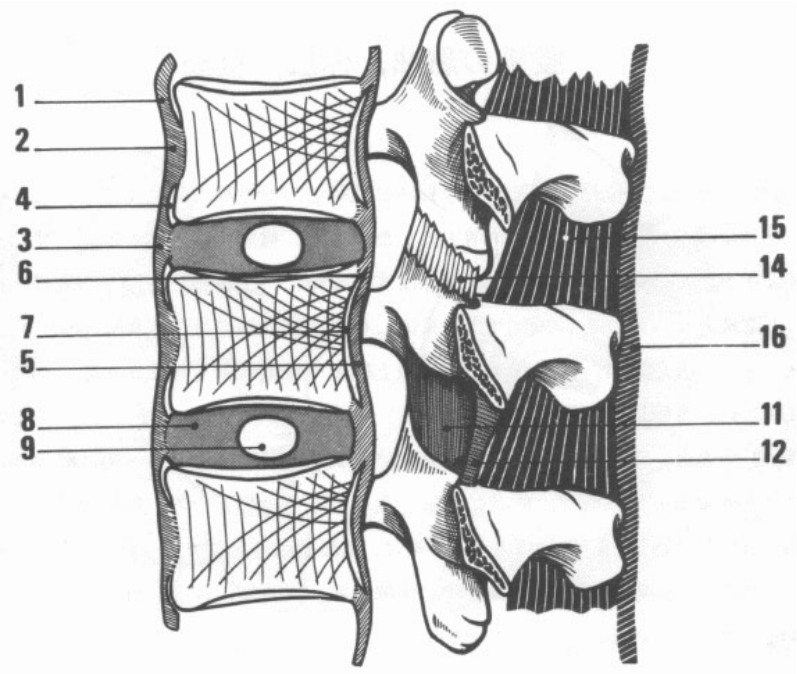



**Fig. 2.6** 人間の脊椎 (「カパンディ関節の生理学 I 体幹・脊柱」より)
Human's vertebrae

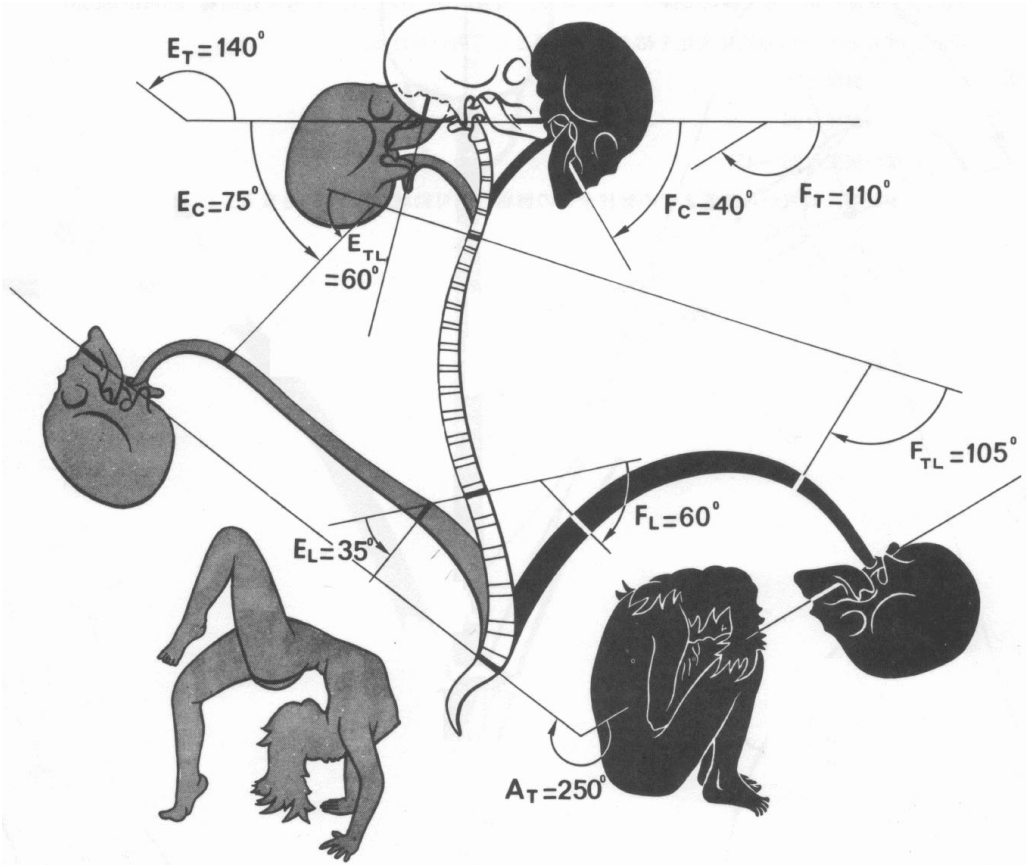



**Fig. 2.7** 人間の脊椎の可動範囲 (「カパンディ関節の生理学 I 体幹・脊柱」より)
Movable area of human spine

度は剛体の自由度 6 と考えると，回転の自由度は前後屈，左右側屈，長軸に沿ったねじりの 3 自由度全て可動である．並進の自由度は，可動範囲は小さいが，前後，左右，上下やはり 3 自由度可動であり，6 自由度全てが可動である．ただし，並進の自由度の可動範囲は極めて小さく，回転の 3 自由度のみ可動であると考えることもできる．回転の 3 自由度も一節一節の可動範囲はかなり小さく，数度〜十数度であるが，多節構造であるため，脊柱全体では， **Fig. 2.7** のような可動範囲を持つ (図は矢状面内の動きのみを示している).

**Fig. 2.7** よりわかるように，部位によって可動量は異る．腰椎部は屈曲 (前屈)($F_L$)=60 °，伸展 (後屈)($E_L$)=35 °，腰胸椎部は全体として屈曲 ($F_{TL}$)=105 °，伸展 ($E_{TL}$)=60 °，頸椎部は屈曲 ($F_C$)=40 °，伸展 ($E_C$)=75 ° である．腰椎・胸椎の部分は，前述のロック機構がはたらくためあまり可動範囲が大きくないと考えられる．なお，これらの数値は年齢と共に大きく変化する．ここに示したのは最大値である．

### 2.2.2 人間の脊柱 (脊椎) の役割

人間の脊椎は以下のような役割を担っている.

**固定軸としての働き** 頭と両腕を支える役割. 作業をする時, 両腕はなんらかの動作を行うことが多く, その動作に対する反力や両腕にかかる外力を安定に支えることのできるベースとしての役割を持つ.

**体軸および脊髄の保護器官としての働き** 腰と肩 (首) の部分では脊椎は体の前後方向の中心付近を通っているが, その間 (体幹の中央部分・胸椎部分) は脊椎の S 字湾曲により体の中心より後方 (後 1/4 以内) を通っている. この部分は前側には肋骨があり, 肋骨と脊椎の間の空間には心臓や肺をはじめとする内臓があり, 肋骨とあわせて内蔵の保護器官の役割を持つ. そして, 脊椎は同時に脊髄の保護器官の役割も担っている. 脳から下りてくる延髄・脊髄と呼ばれる大量の神経配線の束を保護している.

**屈曲・回旋動作を担う自由度としての働き** 本節でも述べてきたように, 体幹の屈曲・回旋動作を担うというのは, 脊椎の最も重要な役割の一つである. この自由度により, 狭所での作業, 効率の良い動作・行動, 手足・脊椎を利用した全身行動 (第 2.3節) がが可能になる.

脊椎構造は強靭性と柔軟性という二つの矛盾した力学的要求に対処しなければならない. そこには脊椎の重要性と難しさがあると言える. 本研究では, 矛盾した力学的要求に対処できるしくみと, 脊椎の役割のロボットへの組み込みの双方の実現を目指している.

## 2.3 全身行動

全身行動 [32] とは, アームや脚を複数持つ全身型ロボット (第 2.1節) において, アームや脚を含む全身の自由度を, ある行動のために用いるような行動を言う. 全身型ロボットの代表例は人間型ロボットである. 人間型ロボットは, 脚先をベースとし手先をエンドエフェクタとする冗長マニピュレータとみなすこともできるが, そうみなせるような種類の行動はこれまでのロボットの技術の延長と考えることができる. しかし, 例えば, 手足を使って瓦礫の山をよじ登るような行動や, 転んだ時に起き上がったり寝返りを打ったりというような行動は, これまでのロボットの技術の延長とは別の技術が必要になると考えられる. また, 物を投げる行動や重量上げのような行動は, 全身の自由度の動きをどのように組み合わせてどのようなタイミングで実行するかといったことも重要になってくる.

脊椎構造を持つロボット (や人間や動物) による全身行動は, 人間型ロボットによる全身行動よりさらに発展した形態になる. 脊椎の自由度が加わることにより, 例えば, 机の下にもぐ

りこむような行動や，入り組んだ配管の間を縫ってバルブを操作するなど，体幹の変形が不可欠であるような狭い場所での作業等の全身行動も可能になる．

## 2.4 関連する研究

### 2.4.1 柔軟性を持つロボット

柔軟性可変な体幹を持つ全身型ロボットの研究はこれまで行われてこなかった．類似した研究としては，構造材が柔軟性を持つロボット [78, 79, 4] や関節の剛性を機械的に調節できるロボット [62] などの研究が行われてきた [80].

#### 非可変柔軟性・マニピュレータ

一般にフレキシブルロボットとは，構造が機械的柔軟性を有することを前提とするロボット全般を指すが，特にこれまでにフレキシブルロボットと呼ばれる研究分野 [99] では，例えば軽量マニピュレータや高速な動作などのように，リンクが剛体であるとみなせないような場合の扱い，すなわち振動を制御したり先端のエンドエフェクタの位置・姿勢を制御したりということが行われてきた．空気圧人工筋を用いて柔軟性を持たせる例もあり [75]，人間の筋肉に近い性質を得られる．

#### 可変柔軟性

柔軟性を可変にする研究は，古くから行われている方法は制御によるものであった．しかし，衝突時の衝撃吸収などを想定すると，ソフトウェアによる制御によるには制御周期の限界があり，瞬間的な衝撃力を和らげることができない．この問題を解決するためには機械的に柔軟性を持つ (剛性の低い) 素材を用いる必要がある．機械的柔軟性を可変にする構造の研究はこれまでにいくつかのトライアルが行われている [46]．一つの方法は，腱駆動の拮抗式にして，腱に直列に非線形バネ要素やダンパ要素を組み込むというものがあり [17]，こうした方法で 7 自由度の腱駆動アームも開発された [81]．回転関節部分に直接機械的柔軟性を調節できる機構を組み込んだ例もある． MIA(Mechanical Impedance Adjuster) と呼ばれる，板バネとブレーキ (擬似ダンパ要素) により回転関節に機械的柔軟性を調節できる機構を設け，これを組み合わせて 7 自由度のアームを構成した [63]．脊椎構造の柔軟性調節と MIA の違いは， MIA が関節の剛性 ( ・粘性) を調節するのに対し，脊椎構造は構造の柔軟性を調節する．関節の剛性を調節してもリンクの慣性が大きいと衝突時の衝撃が大きくなるが，構造を柔軟にすると構造の変形により衝撃吸収が可能になるため，大きな慣性力が生じにくい．脊椎の柔軟性はフレキ

シブルマニピュレータの柔軟性に近い.

### 2.4.2 脊椎のようなロボット (Spine Robot)

脊椎構造を持つロボットとして，スウェーデンの The Spine Robot [44, 45] がある．このロボットは楕円球面の接触面を持つ円盤状のユニットを積み重ねて脊椎構造を形成し，その接触面にギアを刻むことで，ユニット同士がずれるのを防いでいる．この構造の利点は，各ユニットを安価に高強度に作れると点にある．駆動方法として，ワイヤ駆動を行っているが，アクチュエータ一個で曲げ伸ばしの両方を行っている．この方法はワイヤが一本でも切断すると全体が制御できなくなるという難点がある． The Spine Robot は脊椎が椎骨のみで構成されたロボットであったが，これに加えて椎間板を有するロボットとしては，液体圧を動力として組み込み，椎間板にアクチュエータの役割を持たせたもの [91] や椎間板にモータとクランクを３つ組み込んだもの [97, 98] がある．脊椎構造に似た構造を持つロボットとしては，付け根にモータを集中配置したワイヤ駆動の Space Crane[102] や，ユニバーサルジョイントを連結し，各ジョイントについたプレートにワイヤを取り付け，ジョイントごとに全自由度を制御できる Scripps Tensor Arm [1] や，各節の自由度をリンクで拘束することで，根本の一つの節を少し動かすだけで，大きな動きを作り出せる Articulating Mechanism[64] がある．

これらの脊椎のような機構を持つロボットは，しかし，全身型ではなかった．つまり手足や四肢を持つ構造ではなかった．また，柔軟性という観点もあまり考慮されていない.

### 2.4.3 全身型ロボット

全身型ロボットの研究は，これまで数多く行われており，全身を持つ人間型のロボット (ヒューマノイド) は，その代表的なものと言える． 1970 年代に最初の全身を持つ人間型ロボット WABOT-1[36] が開発されてから，国内外で様々な全身型ヒューマノイドの研究がされてきたが，特に近年それは盛んであり， WABIAN[11]， P2[12]， H6[74]，リモートブレイン [20] を始めとして多くの研究・開発が進み，それぞれのプロジェクトはさらに進化を進めている．第 2.3節に述べたような全身をフルに使った全身行動という概念は，特にリモートブレインロボットにおいて研究がされている [25].

全身型ヒューマノイドの中には，体幹に自由度を持つものもあるが [104, 90]，体幹の自由度は回転中心が一点の集中関節型であり脊椎とは異なる構造である．また，柔軟性はあまり考慮されていない.

# 第 3 章

# 脊椎構造の設計 ── 基本姿勢と復元力を有する脊椎構造 ──

本章では，第 2章における可変柔軟性を持つ構造の従来研究について概説・考察をふまえた上で本研究における可変な柔軟性を有する脊椎機構の設計と実際の製作に関し述べる．第 3.1節で脊椎構造の実現のポイントを整理し，第 3.2節以降に実際の設計・製作に関し述べる．前半 (第 3.2節〜第 3.3節) では，変形の自由度・柔軟性調節可能自由度の数を限定した脊椎を試作し全身型ロボットに組み込み，動作実験を通して変形の自由度・柔軟性調節可能自由度の持つ意味を整理する．そして，第 3.4節では，変形・柔軟性調節の両自由度を拡大した脊椎構造に関し人間の脊椎構造との比較を通して考察し，全身型ロボットのための脊椎構造のあり方を検討する．後半の節 (第 3.5節〜第 3.7節) では，変形と柔軟性調節の自由度を拡大した第 3.4節 における主張に基づいた脊椎構造の設計・製作とそれらを組み込んだ全身型ロボットの開発に関し述べる．

## 3.1 多自由度全身型ロボットのための柔軟性可変な脊椎構造

### 3.1.1 問題の本質

問題の本質は，脊椎の特徴である 柔軟性 と 自由度増加 をどのような構造で実現するか，ということである． 柔軟性 は衝突時の衝撃吸収により環境や自分自身に危害を加えることを防止する機能や，人間との接触を伴うような作業・行動において人間の受ける恐怖感を減らすという役割を持つ． 自由度増加 により，狭所での作業や脊椎が変形しなければ届かない場所へのアクセスなどが可能になる．例えば机の下に潜り込む必要があるような場合である．また，運動性能の向上も期待される．つまり，四肢の動きを脊椎の動きと協調させることで，四肢の動きのみでは達成できない動作の達成や，効率や性能の向上を見込むことができる．

### 3.1.2 脊椎構造の変形と柔軟性

柔軟性と可動自由度の増加は，独立に扱うことができる．柔軟性には， (1) 常に柔軟， (2) 常に剛， (3) 可変柔軟性，の三種類がある．厳密には可変と不変の二種類だが，ここでは便宜的に三種類として捉えて話を進める． (1) 〜 (3) は，二つの剛体の相対変位を規定する 6 自由度に対応した 6 自由度それぞれに関し (1) 〜 (3) のいずれかの状態が想定できる．したがって，柔軟性調節の自由度とでも呼ぶべき量を定義することができる．脊椎構造により柔軟性と自由度が増加すると述べたが，この二つはそれぞれ自由度という観点で捉えることができる．すなわち，脊椎構造を有するロボットは，体幹部に変形に 6 種類のパラメータと，柔軟性調節に 6 種類のパラメータを有することになる．体幹の変形は多節構造であれば 6 自由度以上の自由度数を持ち得るが，その両端の位置・姿勢という観点から捉えれば 6 つの独立なパラメータにより記述できる．柔軟性も同様である． 6 つのパラメータは固定か否かで定数か変数に分けられる．変数のうち操作 (制御) 可能なものの数を自由度と呼ぶ．

脊椎構造の設計のポイントは，変形に関する自由度と柔軟性に関する自由度をいかに多く制御可能にするかということである．

### 3.1.3 アプローチ

本研究でとるアプローチに関し述べる．まず始めに，変形に関する調節可能なパラメータ (能動変形の自由度) 数，柔軟性に関する調節可能なパラメータ数 (剛性調節可能な自由度) が，比較的少ない機構の脊椎を試作し，全身型ロボットに組み込む．さらにこれらのロボットによる実験を通じて，各自由度の持つ意味を整理しそれらのパラメータの重要性を確認し知見を得る．そしてそこで得られた情報と人間の脊椎構造の特徴の解析により，全身型ロボットの脊椎の構造がどうあるべきかを議論する．この議論に基づき，変形・柔軟性調節の両自由度を拡大した脊椎構造を，三段階に分けて設計・製作・全身型ロボットへの組み込みを行う．

## 3.2 変形 3 変数 (2 自由度)・柔軟性 3 変数 (0 自由度) の脊椎

### 3.2.1 脊椎の構造

変形に関する変数 3(回転 3)，そのうち操作可能な変数の数 (自由度数) が 2 であり，柔軟性に関する変数 3(回転 3)，そのうち操作可能な変数の数 (自由度数) が 0 であるような，脊椎構造を試作した．製作した脊椎の構造を **Fig. 3.1** に示す [51].

脊椎部は上下及び左右の端部に穴がある椎骨に模した剛体の部品 A とその間をつなぐ柔軟な部品 B(メモリフォーム[1)]) が交互に並んだ構造をなしている．

部品 A の左右の穴にはスチロール樹脂の円柱 (ポール) が，上下の穴にはワイヤーが通されており，各々別のサーボモジュールに固定されている．製作した脊椎の動作について説明する．サーボモジュール A を駆動することにより脊椎全体を通る円柱 (ポール) の長さに左右で差が生じるので，脊椎が左右方向に屈曲する (スチロールポールは伸縮はしないが容易に曲がる)．上下方向も同様にワイヤー長に差が生じるので，サーボモジュール B によって上下方向の屈曲動作が実現される．上述した構造により脊椎全体では二方向の屈曲動作を実現している．

左右方向と上下方向について別の構造を用いた理由として，実際の動作時に左右方向にかかる衝撃力は上下方向と比較して小さいことが挙げられる．そのために左右方向は制御性を重

[1)] 本研究では山光油業株式会社の M-41 を使用した

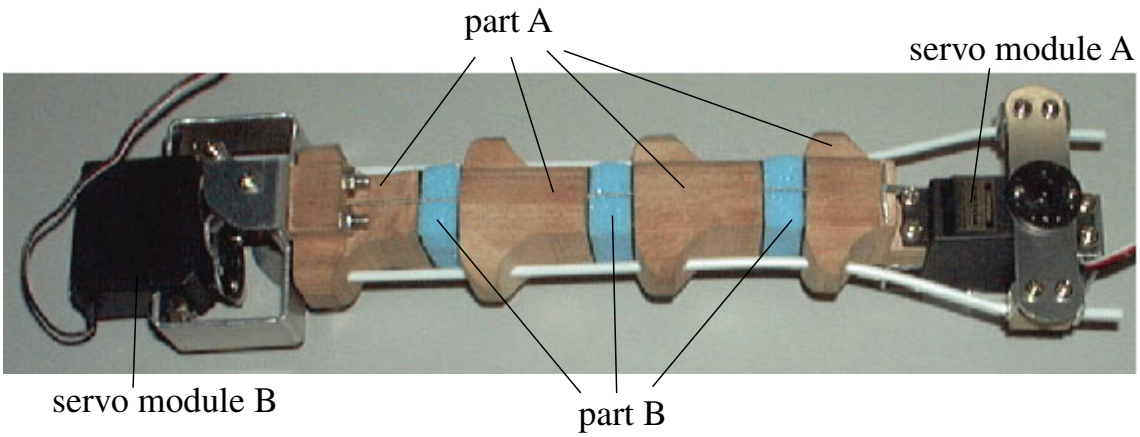



**Fig. 3.1** 柔軟性が可変でない 2 自由度の脊椎
2-DOF spine with fixed flexibility

視し円柱 (ポール) を用い，上下方向は受動的な柔軟性を重視してワイヤを用い，上述したような構造を採用することにした.

### 3.2.2 四脚ロボット “SQ43” への組み込み

人間等の胴体部は脊椎機構により任意の方向に屈曲する動作が可能である．このような機構を四脚ロボットの胴体部に組み込むことにより以下に記述するような動作が可能になると考えられる．まず，従来のもの比較してより広範囲な脚配置が可能になるので，旋回性能の向上，不整地における歩行可能領域の拡大，より高い段差の乗り越え動作などが実現できると考えられる．また胴体部と脚を連動させることにより，跳躍，走行等の動的な動作を実現する可能性が増すと考えられる．なお四脚ロボットの胴体部に可変機構を組み込んだ研究例としては岩本らの研究 [23] が，また脊椎機構については川原らの研究 [37] が挙げられるが，実際の四脚ロボットに脊椎機構を組み込んだ例はなかった.

3.2.1節に述べた脊椎を組み込んだ四脚ロボット “SQ43” を製作した．製作した四脚ロボットの外観図と自由度配置を **Fig. 3.2** に示す．自由度配置は各脚に 3 自由度, 胴体部 (脊椎部) に 2 自由度を配し, ロボット全体で 14 自由度を有する．重量は 1.9[kg] である.

姿勢センサの代わりに 3 軸加速度センサを脊椎の前後に 1 個ずつ計 2 個搭載した．センサ情報の収集とモータ制御指令は小型プロセッサボード H8-01[42] によって行い，リモートのホストコンピュータとシリアル通信で結んだ.

### 3.2.3 有限要素法のシミュレーションの設計への活用

最初に製作した脊椎構造は柔軟性がやや大きすぎた．特にねじれ方向の変形を拘束する力が非常に弱く，対角二脚を曲げて持ち上げるような動作を行おうとしても，脊椎がねじれ脚を持ち上げることができなかった．この特性は，不整地においても脚を全て接地する姿勢を受動的に取ることができるという利点があるが，前進動作もままならないというのはちょっと柔軟性が大きすぎると言える．そこで，脊椎の形状を調整してねじれ方向の剛性を上げる改良を，有限要素法と最急降下法を用いて行った．

#### 有限要素法

有限要素法を用いた理由として，

- 構造解析の手法として一般的である．
- 市販の機構解析ソフトウェアと有限要素ソフトウェアを容易に組み合わせることが可能である．

が挙げられる．一番目の理由に関してはすでに建築等の大型構造物からマイクロまで構造解析の分野等で有限要素法は最も多く使用されている解析手法である．またすでに多数の商用パッケージが存在しており，実用段階に入っていると言える．二番目の理由に関しては，今回使用する機構解析ソフトウェアと有限要素解析ソフトウェアとには両者を組み合わせて解析する機能が含まれており，それ以上手を加える必要がないということが大きな長所となる．

本研究では有限要素法を用いて動力学シミュレーション環境を作成し [51]，落下時の衝撃の比較 [52] や，形状の最適化，動作生成 (第 5.4節参照) に利用した．本節では，有限要素法シミュレーションの形状最適化への利用に関し述べる．

#### 設計パラメータの定義

製作した脊椎は柔軟性が幾分大きすぎた．特に捻じれ方向は変形の自由度を持たない受動変形軸で，その方向の剛性はもう少し大きいべきであった．そこで，脊椎の剛性をいくらか増すために脊椎形状の微調節を行った．脊椎形状のパラメータとして **Fig. 3.3** に示す 4 個の部品 A(図の網掛けで示されている部品) の水平面となす角 $a_i(i = 1 \ldots 4)$ を用いた．

なお以下の様な拘束条件を加えるので，実際の設計変数の数は 3 になる．

- parts1 と parts4 の z 軸の位置は同一にする．つまり脊椎機構の前半に付く部分と後半につく部分は同一の高さとなる．

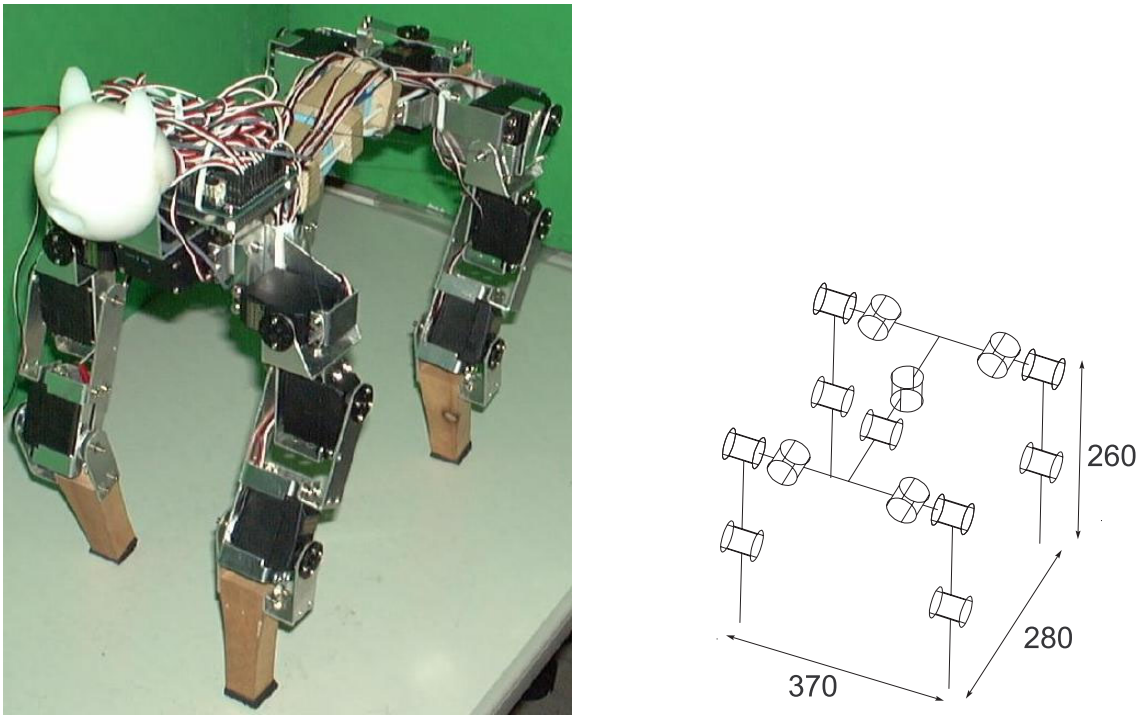



**Fig. 3.2** SQ43 の外観と自由度配置
Appearance and DOF arrangement of SQ43

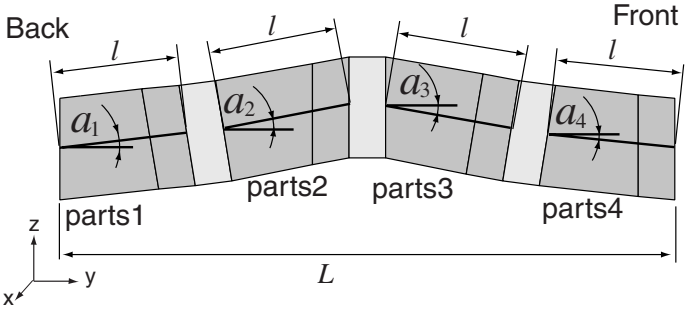



**Fig. 3.3** SQ43 の脊椎形状の設計変数
The design parameters of the spine of SQ43

- 各部品の相対角度 $a_i(i = 1 \ldots 4)$ は $\pm 30^\circ$ の範囲とする．
- 各部品の y 方向長さは全ての部品で同一とする．

従って脊椎の全長を $L$，各部品の y 方向の長さを $l$ とすると，上記の拘束条件は以下の式で表される．

$$\left.\begin{aligned} \sin a_1 + \sin a_2 + \sin a_3 + \sin a_4 &= 0 \\ l\cos a_1 + l\cos a_2 + l\cos a_3 + l\cos a_4 &= L \end{aligned}\right\} \tag{3.1}$$

但し実際には椎骨部品間をつなぐ部品 B(メモリフォーム) が存在するので，少し値を変更する必要がある．部品 B の y 方向の長さは全て 10[mm] とする．実際の最適化においては設計変数として， $a_1$， $a_2$， $a_3$ を用いることとした．

### 最適化の方法

有限要素法 (FEM:Finite Element Method)[41, 16] および最急降下法を用いて改良を行う．まず，設計変数に対する評価値の求め方について述べる．今回の形状の改良ではねじり剛性を大きくする方向に形状を変更することを目的とする．しかし直接ねじり剛性を計算するのは困難なので，ねじり剛性を近似的に表す評価値として，四脚支持の姿勢から左前脚を上方に引き上げる動作を行った際の左前脚先端の地面からの距離を評価値として定める．実際の手順としては， (1) 前項で述べた設計変数から生成された形状の脊椎の有限要素モデルをロボットのモデルに組み込む． (2) 組み込まれたモデルを使用して上記の動作をシミュレーションで解析する． (3) ある時刻における左前脚の脚先端の z 軸方向の位置を評価値として求める．という手順で行った．

最適化の手法としては最急降下法を用いた．設計変数 $(a_1, \ldots, a_m)$(但し$m$は設計変数の数) に対する評価値を $f(a_1, \ldots, a_m)$，また 1 ステップ毎の設計変数の変化量を $\delta a_i/\delta t$(但し$i = 1 \ldots m$) と定める．ここでステップ $t$ における設計変数が $(a_1(t), \ldots, a_m(t))$ であるとする．この時，次のステップ $t + \delta t$ における設計変数は以下のように求める．まず，現在の設計変数の近傍の設計変数に対する評価関数の値を，すなわち

$$\begin{gathered} f(a_1(t), \ldots, a_m(t)) \\ f(a_1(t) + \delta a_1/\delta t, \ldots, a_m(t)) \\ f(a_1(t) - \delta a_1/\delta t, \ldots, a_m(t)) \\ \vdots \\ f(a_1(t), \ldots, a_m(t) + \delta a_m/\delta t) \\ f(a_1(t), \ldots, a_m(t) - \delta a_m/\delta t) \end{gathered}$$

を全て求める．求めた各々の設計変数に対する評価関数 $f$ の値に関して，この値を最大にする設計変数をステップ $t+\delta t$ の設計変数と定める．この作業を繰り返し，

$$f(a_1(t),\ldots,a_m(t)) = f(a_1(t+\delta t),\ldots,a_m(t+\delta t)) \tag{3.2}$$

が満たされた時，すなわち近傍の全ての設計変数に対する評価関数の値よりも，現在の設計変数に対する評価関数の値が大きい時に最適化は終了となり，$(a_1(t),\ldots,a_m(t))$ が評価関数 $f$ に対する最適な設計変数となる．

| | $a_1$ ° | $a_2$ ° | $a_3$ ° |
|---|---|---|---|
| 設計変数 1 | 5.0 | 5.0 | 5.0 |
| 設計変数 2 | -5.0 | -5.0 | 5.0 |
| 設計変数 3 | 15.0 | 5.0 | -15.0 |
| 設計変数 4 | 5.0 | -15.0 | 5.0 |

Table 3.1　最適化開始の設計変数
The initial values of design parameters

この手法には探索空間が広い，すなわち設計変数の数が多い場合には，計算時間の増大しやすいことや，局所解に陥ることなどの問題点が存在する．しかし今回行う最適化は設計変数の数が 3 個と少ないのでこの手法を用いるには適していると判断した．また局所解の問題に関しては解析を開始する設計変数を複数用意することで，問題の解決を図った．今回の最適化では設計変数の各ステップに対する変化率を $\delta a_1/\delta t = \delta a_2/\delta t = \delta a_3/\delta t = 0.5°$ とした．また最適化を開始する設計変数として Table 3.1 に示す 4 個を用いた．

### 最適化の結果

最適化の過程における評価値の変化を **Fig. 3.4** に示す．グラフから確認できる通り，開始点を設計変数 2 とした場合の最終評価値が最も高い値となっている．また開始点が設計変数 1 及び設計変数 3 の場合は，開始後すぐに極小値に陥っていることが分かる．また設計変数 4 を開始点とした場合は最適化の途中から，設計変数 1 の場合の最適化とほぼ同一の経路をたどっており，ほぼ等しい評価値で収束していることが分かる．そこで今回は一番目の経路で収束した際の設計変数を最適値と定めることにする．最適化の結果の脊椎の形状の有限要素モデルを **Fig. 3.5** に示す．

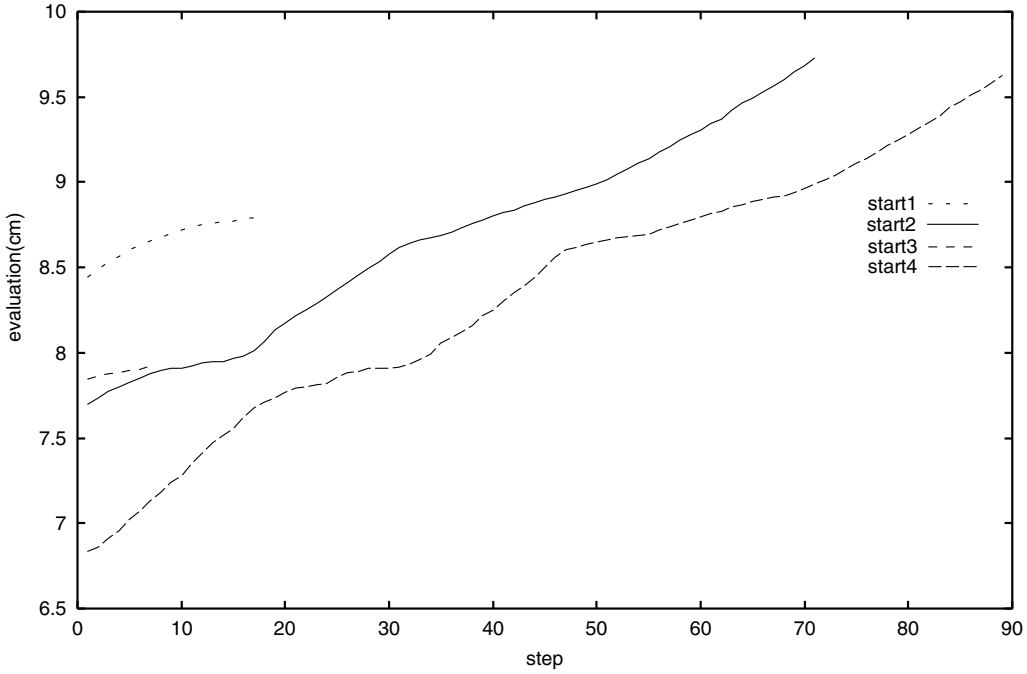



**Fig. 3.4** 評価値の変化
Change of evaluation

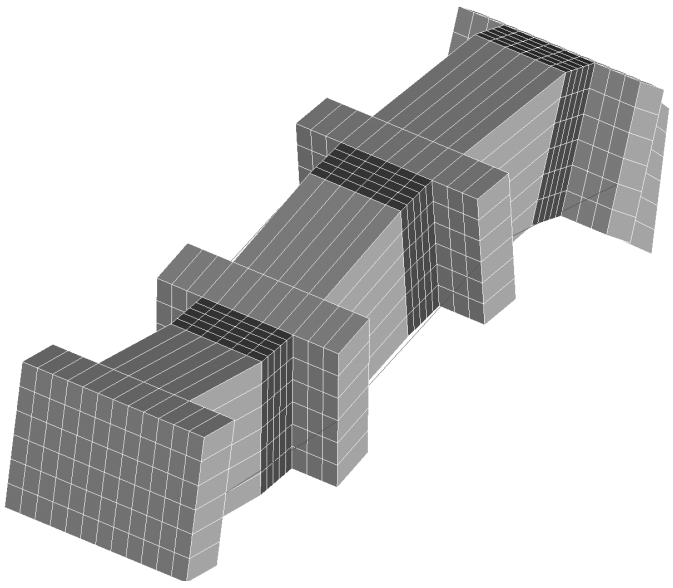


**Fig. 3.5** 最適化された脊椎の有限要素モデル
The FEM-model of the optimized spine

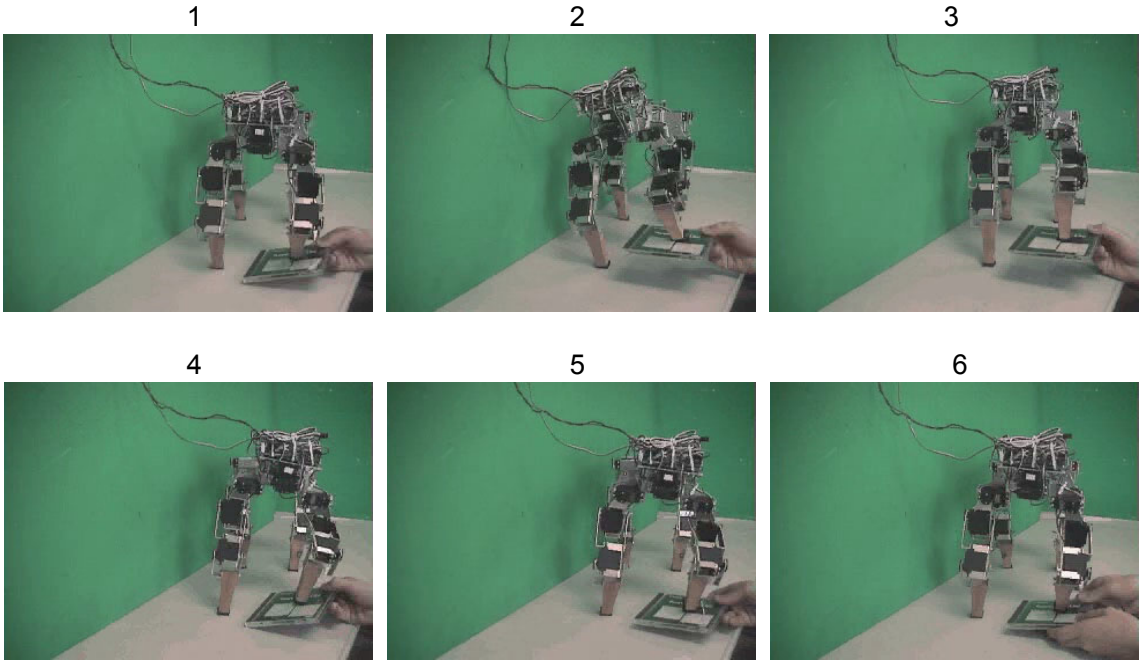



**Fig. 3.6** SQ43 による水平保持動作
SQ43's keeping horizontal

### 3.2.4 SQ43 の動作例

#### 姿勢制御

3 軸加速度センサの情報を利用して重力方向を検知し，脊椎の両端の部位を水平に保つ制御を行った． **Fig. 3.6** はその様子を前方より見た写真である．1 から 2 へ左前脚の乗った板を持ち上げると，それによって脊椎上部 (前部) の肩に相当する部分が右下がりに傾く．加速度センサは動作中は動作に付随する加速度を含んだ値が得られて重力方向がずれるため，センサの値が定常になった時の値のみを姿勢情報として利用する． **Fig. 3.6** の 2 から 3 で肩部分を水平になるように左前脚の曲げ角を調節している．左前脚の乗った板を下げると，今度は **Fig. 3.6** の 4 にように左前が下がるが，やがて左前脚を伸ばし (5) 水平に戻る (6).

#### クロール歩容による四脚歩行動作

脚を 1 本ずつ前に運ぶクロール歩容による四脚歩行動作の実験を行った **Fig. 3.7** はその様子を示す． SQ43 は四脚で直立した状態から 1 脚を持ち上げようとすると，脊椎のねじれ方向の柔軟性により体幹がねじれ，持ち上げようとした脚の部分が沈む形になり，脚を持ち上げることができない．そこで，持ち上げようとする脚から重心をなるべく遠ざけた姿勢をとり

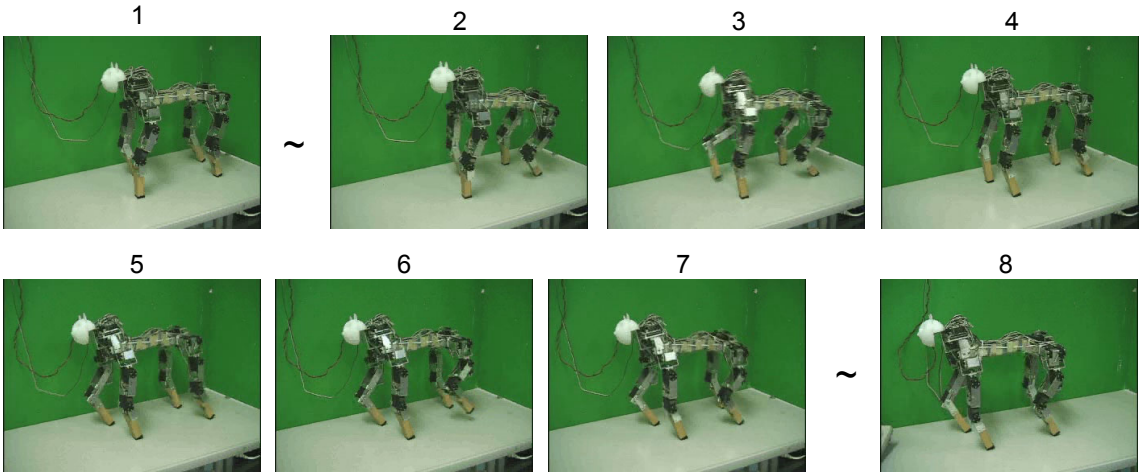



**Fig. 3.7** SQ43 によるクロール歩容
SQ43's crawl walking

( **Fig. 3.7** の 2: 右前脚を持ち上げるために重心を左後ろ脚の真上付近に近づけるような姿勢をとっている)，素早く脚を持ち上げると ( **Fig. 3.7** の 3: 右前脚を持ち上げたところ)，脚を持ち上げることができる．しかし，その姿勢を保持はできず，持ち上げた脚の方に傾いてゆく．そこで，素早く脚を上げ，脊椎がねじれて上げた脚が地面につかないうちに素早くその脚を前に運ぶ ( **Fig. 3.7** の 4: 右前脚を素早く前に運び地面に下ろしたところ)． **Fig. 3.7** の 5,6,7 も同様の方法で左後ろ脚を前に運んでいる．その動作の繰り返しで一脚ずつ脚を前に運ぶクロール歩容を実現した.

### 不整地歩行

前々項の肩を水平に保つ姿勢制御と前項のクロール歩容を組み合わせて，地面の形状がわからない不整地を歩行する実験を行った． **Fig. 3.8** はその様子を示す．基本的には前項に示したクロール歩容を行うが，脚を接地した時に地面が平らではないことを検知すると (3 軸加速度センサにより重力方向を算出)，歩容を中断し肩・腰の部分が水平になるように前脚・後脚の曲げ角度を調節する．肩・腰が水平になったらその時の各脚の曲げ角を記憶し，そこからの相対量でクロール歩容の動きを行う．体幹が剛体である場合は，未知の不整地では四脚全てを接地することが難しい (センサを利用するなどが必要) が，脊椎を持つ四脚ロボット SQ43 の場合，脊椎のねじれ方向の柔軟性により，不整地においてもある程度までの不整地であれば地面になじむように脊椎が変形し，四脚全てを接地することができる． **Fig. 3.8** では，高さの違う地面に脚を付き肩が水平でなくなった時 (2) に，加速度センサによりそれを検知し，左前脚を曲げて肩を水平にする (3)．左前脚の高さは保ったままそこからの相対量でクロール歩容を続け，

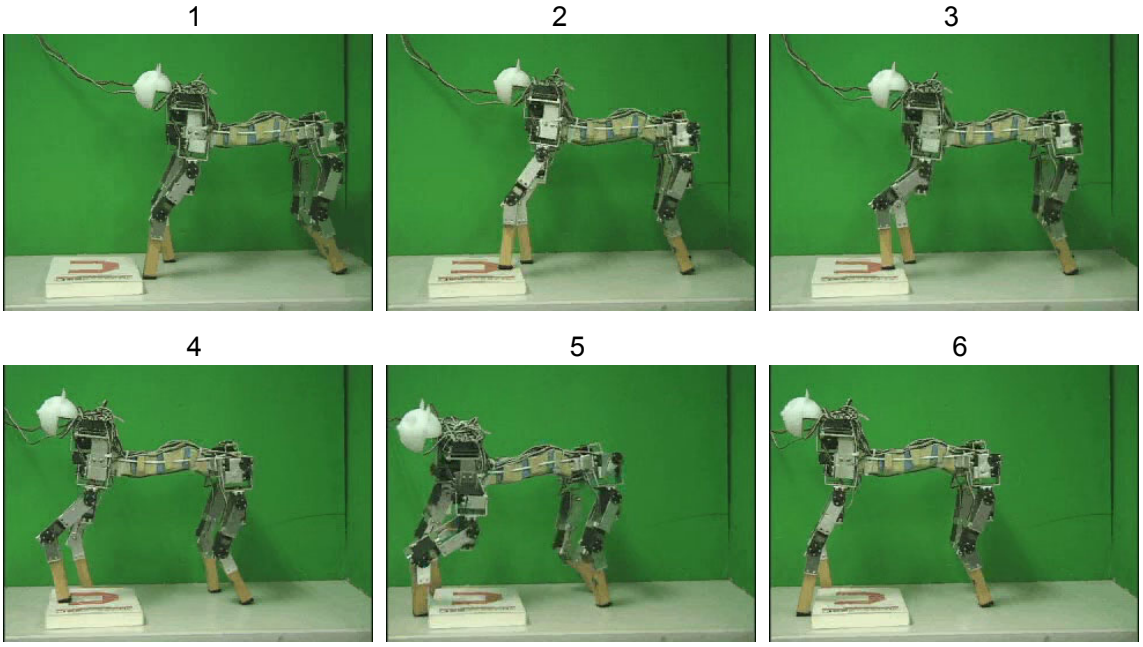



**Fig. 3.8** SQ43 による不整地歩行
SQ43's walking through unknown uneven place

また高さが変わったら (5)，肩が水平でなくなったことを検知し左前脚を伸ばし，肩が水平になるようにする (6).

### 立ち上がり動作

脊椎の動きを利用した動作として，四脚で床に座った状態から立ち上がり四脚で直立した状態になる立ち上がり動作の実験を行った． **Fig. 3.9** にその様子を示す．各脚のアクチュエータのトルクがそれほど大きくないので， **Fig. 3.9** の 1 の状態からいっせいに脚を伸ばしても立ち上がることはできない (8 の状態にはならない)．そこで，前脚と後脚を交互に少しずつ伸ばすような動きを作り，脚を伸ばす際には脊椎がその力を補助するような動きをするようにした．これらの一連の動きは，人間が試行錯誤で少しずつ姿勢を動かして，腹這いの状態から立ち上がる動作を実現した．試行錯誤の時のログを記録しておき，成功した時にそのシークェンスを繰り返すことにより，動作の自動化を行った.

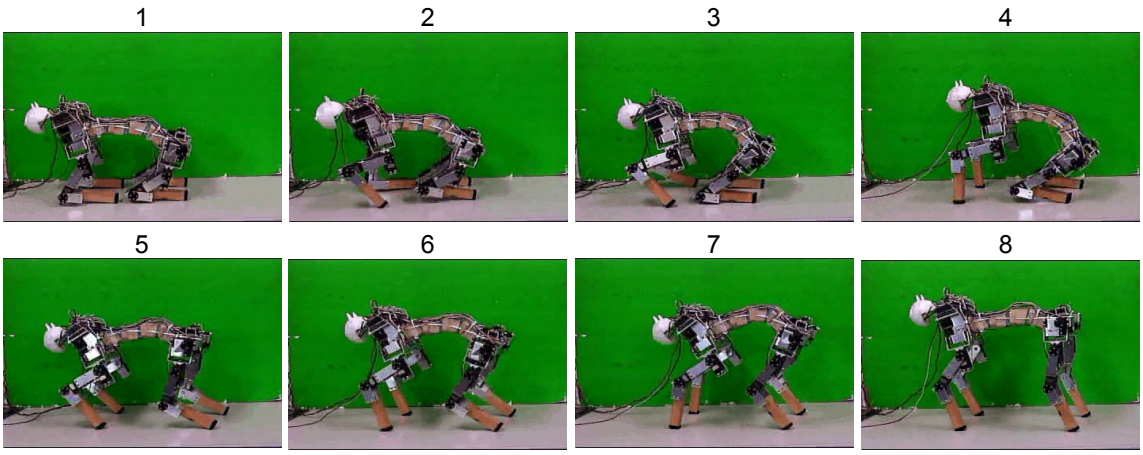



**Fig. 3.9** SQ43 の床からの立ち上がり動作
SQ43's standing up from the floor

### 3.2.5 SQ43 の実験結果

#### 受動変形の問題

SQ43 は四脚直立状態から対角二脚を持ち上げようとしても脊椎がねじれて脚が上がらなかった．四脚歩行動作を行う際にこの問題を解決するために，遊脚になる予定の脚から重心がなるべく遠くなるように他の三脚の姿勢を制御し，遊脚が上がったら素早く前に運ぶという動作を行った．しかし，実際に脚を持ち上げた姿勢を保つことを必要とする行動は様々なものが想定でき，脚が上げた姿勢を保持できないのは大きな問題である．この問題を解決するには，

1. 柔軟性調節の自由度

2. ねじり変形を可操作にする (変形自由度を増やす)

のいずれかが必要である．このいずれかが満たされれば，脚を持ち上げた姿勢を状態を保持することは可能になるが，もしいずれか一つだけを満たすとすれば，1. を採用すべきである．なぜなら，2. のみを満たした場合，脚を上げることはできるようになるが，メカニカルな柔軟性は損なわれることになり，不整地への適応などの柔軟性の利点が享受できなくなる可能性があるからである．

## 3.3 変形 1 変数 (0 自由度)・柔軟性 1 変数 (1 自由度) の脊椎

人間型ロボットに柔軟構造を持たせる試みとして，椎骨に相当する部品を積み重ねる構造で，メモリーフォームによる粘弾性を持つ体幹を備えた人間型ロボットを試作した研究があるが [43]，弾性係数・粘性係数といった柔軟性を示す特性は不変であった．また，これまでのロボットの柔軟性の研究にはセンサフィードバックによる例や機械的柔軟要素を利用する例があるが，関節レベルの柔軟性を調節できるのみで，構造材の柔軟性を可変にする試みはほとんど行われていない．いわゆるフレキシブルマニピュレータなどの柔軟な構造材を利用する研究は，柔軟性が可変ではなかった (常に一定の柔軟性を持っていた)．これはフレキシブルマニピュレータが柔軟性を利用するよりもむしろ軽量化のために生じる柔軟性をいかに扱うかにポイントを置く場合が多いためと考えられる．

人間型ロボットは柔軟性が重要であるが，構造材に柔軟性を持たせた場合は，柔軟性が有効な場面と支持材としての機能が求められる場面がある．つまり，状況に応じて柔軟性を調節できる機構が必要である．人間が，筋肉の力を調節することにより関節レベルの柔軟性だけでなく構造的な柔軟性も可変であるように，人間型ロボットに柔軟性を持つ構造を組み込む場合にも，柔軟さを調節できる機構を持つことが望ましい．

### 3.3.1 脊椎の構造

柔軟性可変な構造としては，弾性体の変形領域を調節することのできる構造や，柔軟な素材に拮抗筋などを取りつけ筋の張力の変化により柔軟性を可変にする構造などが考えられる．

本研究では，板バネを利用した 1 自由度の柔軟性可変な構造を製作した [54]．製作した脊椎の構造を **Fig. 3.10** に示す．板バネをカムのような形をした剛性の高い板で挟む構造で，カムを調節することで板バネの有効長が変化し柔軟性を変えることができる．カムは，角度変位と板バネの有効長が比例するようにした．

これは極めて単純なものだが，この構造を拡張して，例えば，弾性体の円柱を剛性の高い円筒で包むようにし円筒を上下させるような構造をにすれば，2 方向の変形に対する柔軟性を可変にすることができる．また，長手方向の柔軟性を可変にする構造としては，コイルばねと円盤を利用した剛性可変バネの試作例 [77] などがある．こうした構造を利用することにより，多自由度に対する可変な柔軟性が実現できると考えられる．

また，この構造は弾性係数を可変にしているが，粘性係数を調整できる構造ではない．拮抗筋やダンパを利用することにより，能動的に変形させることや粘性係数を調整することができると考えられる．

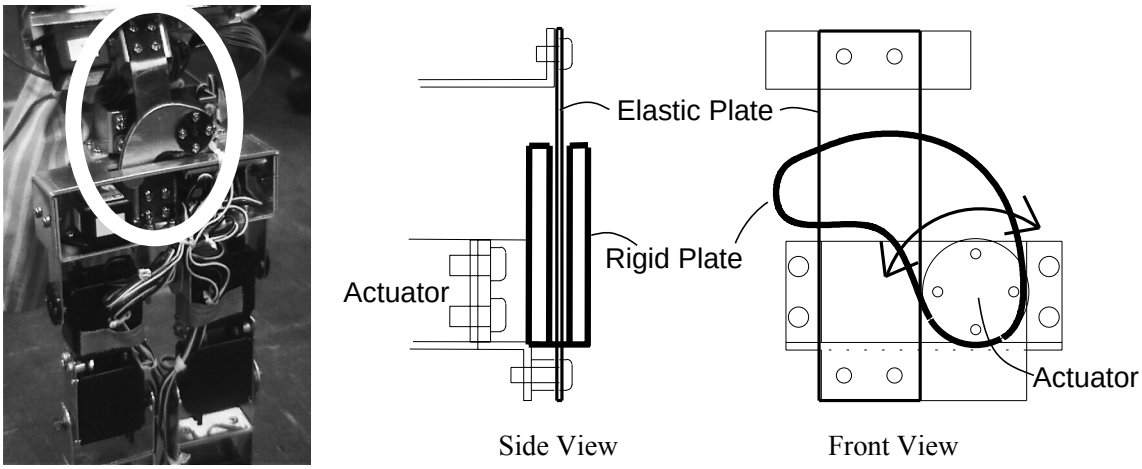



**Fig. 3.10** 板バネを利用した 1 自由度の柔軟性可変構造
1-DOF variable flexible structure using spring plate

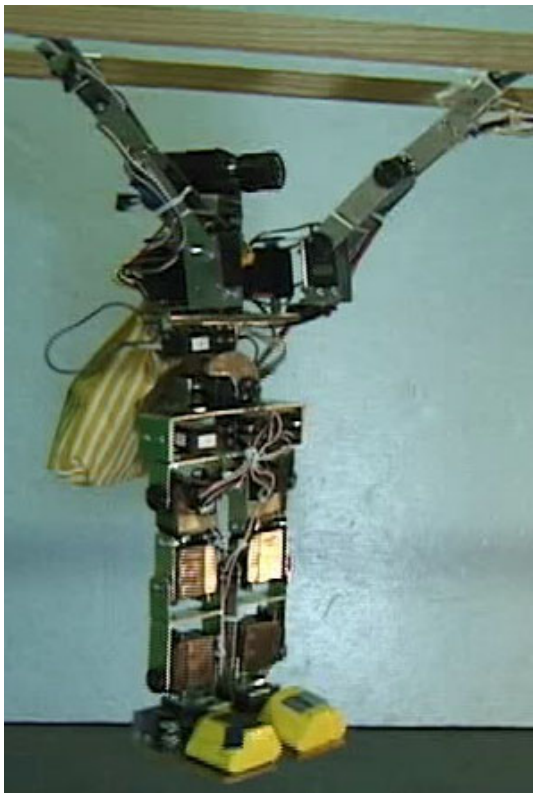


**Fig. 3.11** HanzouS の外観
Appearance of HanzouS

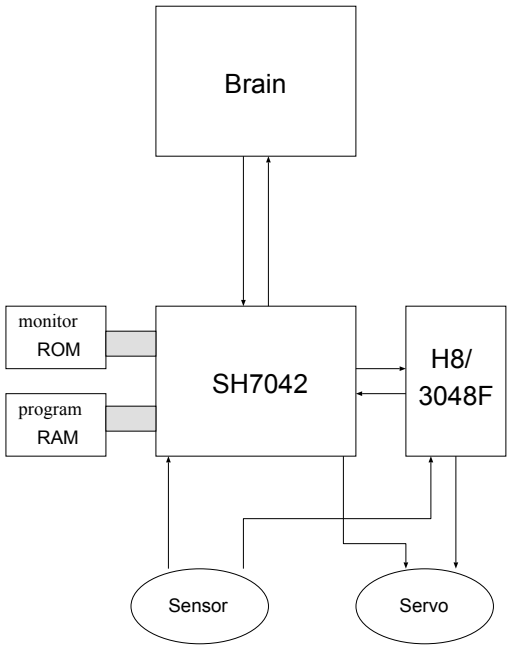



**Fig. 3.12** HanzouS の信号処理系
The signal processing
system of HanzouS

### 3.3.2 人間型ロボット HanzouS への組み込み

3.3.1節で述べた板バネを用いた構造を体幹に用いた 18 関節の人間型ロボット “HanzouS” を製作した ( **Fig. 3.11** ). 弾性係数を調整するモータを含めて合計 19 個のラジコンサーボモジュールと, 3 軸加速度センサを体幹の上下に 1 個ずつ, ジャイロスコープを 1 個, CCD カメラ 1 台を搭載した. 画像以外のセンサ情報と制御信号はボディに搭載した SH-2,H8/3048 の 2 個のマイクロコントローラで入出力を行い, シリアル通信でワークステーションと結んだ ( **Fig. 3.12** ).

### 3.3.3 HanzouS によるブラキエーション動作

人間型ロボットの動作における可変柔軟構造の影響を示すために, 3.3.1節に述べた人間型ロボットを用いて, ブラキエーション動作を例にとり実験を行った. 次のバーを把持した直後の下半身の慣性から生じる衝撃を, 体幹の柔軟構造を利用して吸収する実験を行った ( **Fig. 3.13** ).

動作中の体幹部の柔軟性は以下のように調整した. 下半身を後ろに引いて準備する際には最も剛性の高い状態にした. **Fig. 3.13** の 2 はその時の瞬間だが, 両足を後ろに引き体幹部のカム構造が最も上の位置になっているのがわかる. 衝撃がかかるのは次のバーを把持した直後 ( **Fig. 3.13** の 7) であると予測できるので, その直前 ( **Fig. 3.13** の 6) に最も柔軟な状態に調節した. 実際には右手が後のバーから離れた瞬間 ( **Fig. 3.13** の 3) から次バーに届く直前 ( **Fig. 3.13** の 6) までの間に素早く脊椎の柔軟性調節機構を動かしている. 右手が次のバーを把持した後は, 体幹が柔軟に変形し ( **Fig. 3.14** の 8,9), 衝撃力をの最大値を和らげている様子がわかる.

### 3.3.4 HanzouS の実験結果

#### 衝撃吸収

ブラキエーション動作のバー把持時の衝撃力を脊椎の柔軟性により緩和されている. 柔軟性を持つ脊椎が衝撃吸収に有効である事がわかる ( **Fig. 3.14** ).

#### 可変柔軟性

足を後に引いて準備する際には, 体幹を最も剛な状態にしておくことにより, 重心を後に持ってくることができている. 状況に応じた柔軟性の調節機能の有効性がわかる.

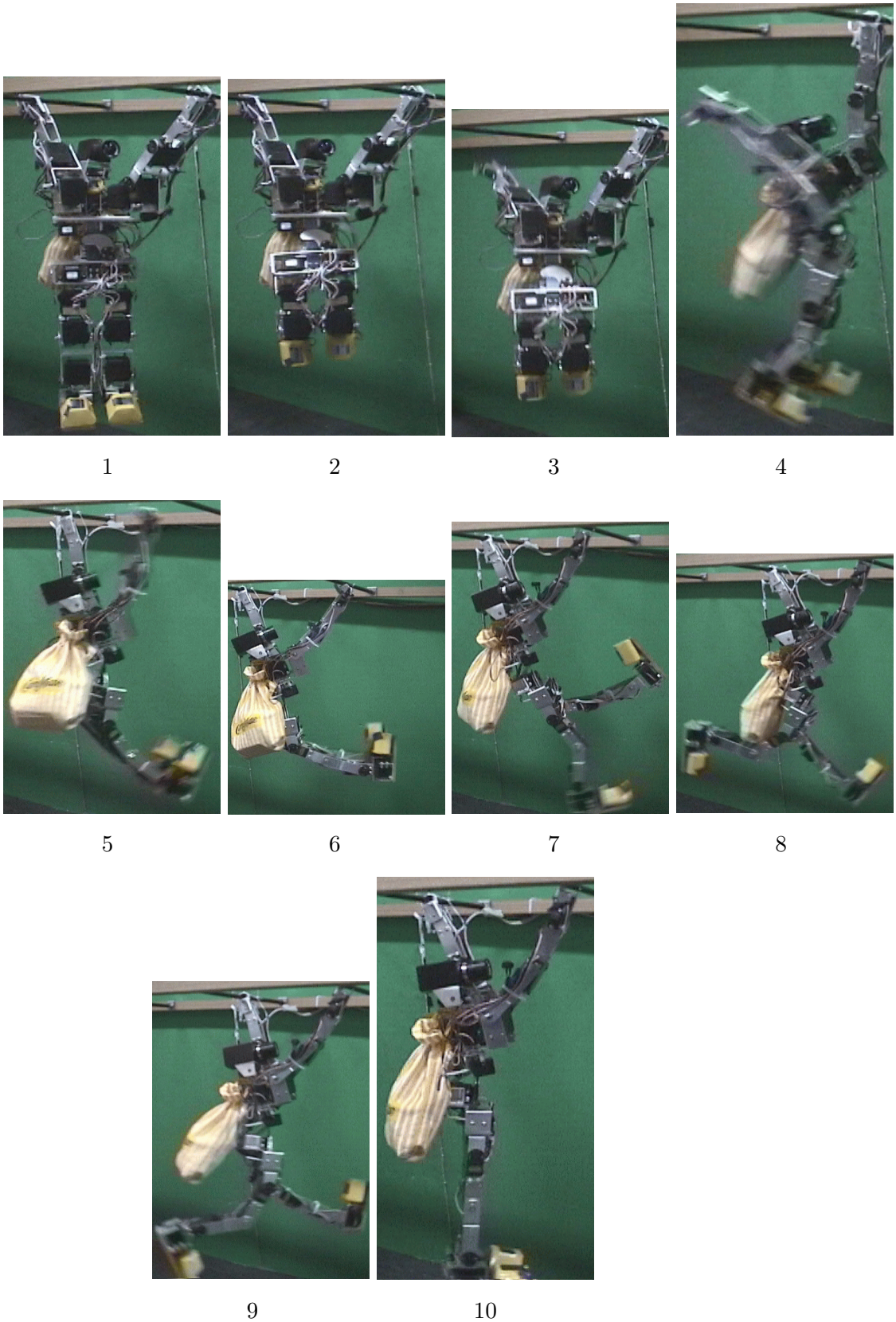



**Fig. 3.13** HanzouS のブラキエーション動作の様子
Brachiation motion of HanzouS

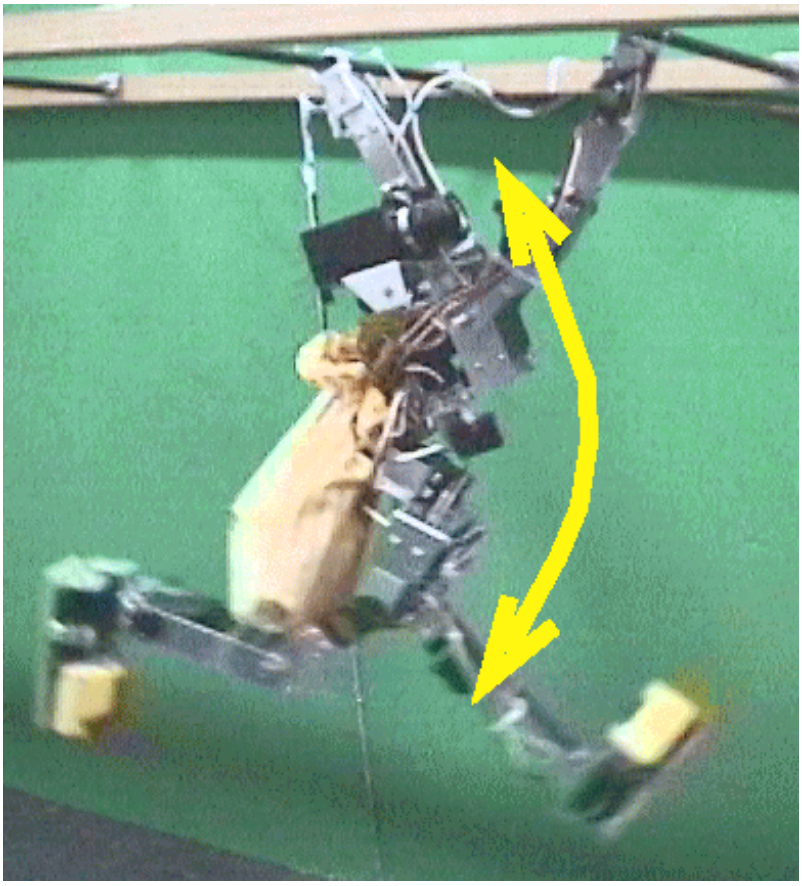


**Fig. 3.14** ブラキエーション動作における柔軟構造の影響
The effect of flexible structure during brachiation

#### 能動変形自由度

脊椎構造の大きな利点は柔軟性と可動範囲の拡大であるが，能動変形の自由度を持たない場合，可動範囲の拡大の利点を享受することができない．能動変形と，柔軟性調節の両自由度を調節できることがベターである．

## 3.4 変形・柔軟性調節の自由度を拡大するには

第 3.2節，第 3.3節より，変形の自由度及び柔軟性調節の自由度が限られている点を改善することが求められるということが言える．人間の脊椎は変形・柔軟性 (剛性) 調節の自由度ともに最大である．その構造に学ぶことは，全身型ロボットのための脊椎構造を検討する上で有効

であると考えられる.

### 3.4.1 人間の脊椎の構造

#### 各節の自由度

人間の脊椎は，上下面が臼状である椎骨が連なり，各隣り合う椎骨の間に海面質の球 (髄核) が挿入された構造になっており ( **Fig. 2.3** )，各節は回転 3 自由度，並進 2 自由度 (軸方向以外) と言われている [35]．各節の可動範囲は部位により異るが，一般に並進の 2 自由度の変形量は小さい.

#### 椎間板

隣り合う椎骨に挟まれた領域で髄核以外の間隙は弾性体の椎間板により埋められている．各椎間板は椎骨の上下面に固着されており，椎間板の自然長より椎骨間の距離の方が大きいので，初期荷重 (プリテンション) のかかった状態になっている．このプリテンションにより人間の脊椎は，変形すると元の姿勢に戻ろうとする 復元力 を持つ．また，椎間板のテンションが釣り合った状態である 初期姿勢 を持つ事になる．人間の場合の脊椎 S 字彎曲 (2.2.1節) が初期姿勢である.

直立状態を基本姿勢とすると，復元力を持つ事により，上体傾斜時に重力による荷重に対する反力を生じる事になる．脊椎を筋肉により駆動する場合，筋肉が出力しなければならない力のうち，重力荷重に抗して上体を支持するための力が復元力により補助され，その分筋肉は姿勢変化のために力を利用することができる.

また，椎間板があることにより，ある節だけが付近の節より大きく変形するという状態が起こりにくくなり，ある程度均等に変形が分散される形になり，体幹全体の変形として見たときに滑らかな曲線に近くなるという特徴もある．人間の筋肉の本数は非常に多く，構造の変形に対し充分な冗長性を持っているが，もし仮に少ない本数であったとしても (あるいは少ない本数になったとしても)，椎間板の釣り合い点でどこかに姿勢が定まることができる．人間のような脊椎機構をロボットに組み込む際に，筋肉の本数を人間ほどには多くできなくとも，少ない筋でも姿勢の安定性は保ちやすい構造であると言える．例えば， **Fig. 3.15** に示すような構造に沿って筋を配置し上下の両端から引き，両端の部品間の距離を拘束したとしても，途中節の姿勢を拘束することができないが， **Fig. 3.15** (b) に示すように弾性要素 (椎間板) を組み込むことにより，途中節の姿勢が滑らかになる．具体的には，仮に外部からの力が無いとすると，筋の張力を $F$，弾性部品の変形量を $dx$，その時の弾性力を $f(dx)$ とすると，直列に配置された各弾性要素には均等に $F$ が加わるので，全ての弾性部品が筋の張力と変形反力が釣り合

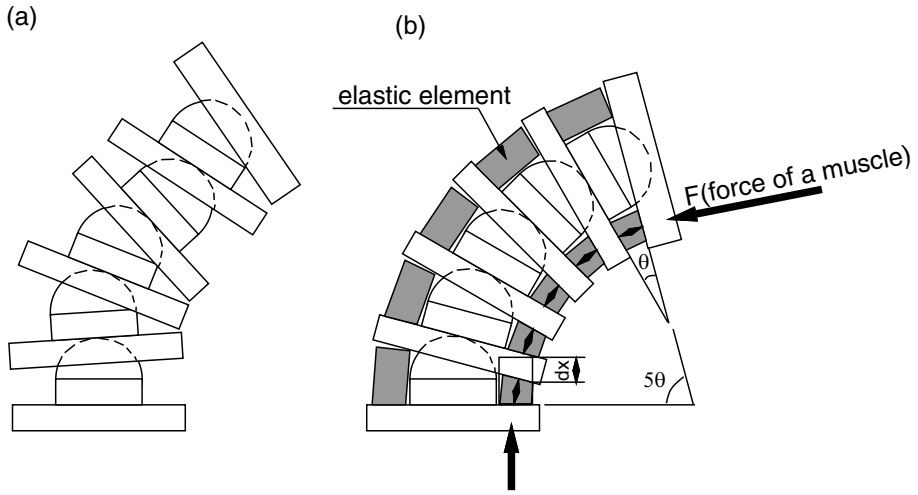



**Fig. 3.15** 多節構造の脊椎の変形における弾性要素 (椎間板) の影響
Effect of elastic elements (intervertebral disk) on the deformation of multi-jointed spine

うような ($F = f(dx)$ となるような) 変形をする.

### 小可動範囲の節が連なる多節構造

人間の脊椎は 24 節あり (2.2.1節参照), 各節の可動範囲は部位によるがいずれもあまり大きくはない. ±数度から±十数度程度である. 各節の可動範囲が小さいことにより, 全体として曲線の形にち近い形での変形になる. また, 機構的冗長性が増すことにより, 姿勢の多様性が増加する. これは, 狭所での作業, 例えば腹部を引かなければ机に腹部が干渉するような状況での腹部を引いての作業などの作業において有効となる. 姿勢の多様性の増加によるもう一つの利点は, より効率の良い動作・ピーク出力が小さい動作が可能になることである. 例えば, 仰向けに横たわった状態で上体を起こす動作を行う場合, 腰の集中関節のトルクにより上体を起こす場合と比較して, 首に近い関節から順に起こすようにすると, 最大トルクが小さくて済む.

多節構造は多数の可動軸を持ち多数の変形パラメータを有するが, 人間の脊椎においてはこのパラメータの数をはるかに上回る, 充分に冗長な本数の筋肉により駆動することにより, 変形の自由度・柔軟性 (剛性) 調節の自由度を実現している. 多節構造の各節が筋肉の終端を有し, 肩から腰までを結ぶ長い筋から隣り合う節どうしを結ぶ短い筋まで, 全てのリンク (節と節の間の剛体と見なせる部位単位) の全てに近い組合わせにおいて, 接続する筋を有する. 筋により駆動される多節構造を多様な姿勢に制御するためには, 中間節もできるだけ多く筋の接続を有することが望ましいと言える.

### 3.4.2 人間の脊椎構造から示唆を得た全身型ロボットの脊椎構造

人間の脊椎構造の特徴から取り入れる事が有効であると思われる要素を組み入れた全身型ロボットの脊椎構造のあり方を整理すると以下のようになる.

- **多節構造**
  姿勢の多様性を実現する.
- **各節は回転 3 自由度**
  (1) 機構的に不安定になりやすいこと, (2) 各節は回転自由度のみであっても多節にすることで全体として並進自由度が実現可能であること, などの理由から, 各節に並進の変形機構を設ける事はせず, 各節回転 3 自由度 (roll,pitch,yaw) とする.
- **コンプライアンス可変な筋による拮抗駆動**
  筋を, 可変コンプライアンス若しくは張力制御可能とすることで剛性調節を実現する. 本数は少なくとも 4 本 (3 自由度を駆動する最小限の筋数) 以上とする.
- **弾性要素の組み込み (椎間板)**
  これにより, (1) 基本姿勢, (2) 復元力, (3) 滑らかな全体変形形状, (4) 姿勢の安定性, を実現する.
- **中間節への筋の接続**
  中間節へ接続される筋が多くなればなるほど, 多節構造による姿勢の多様性の制御可能性が増す. 中間節に接続された筋を有する事により, 一様な変化ではないような変形, すなわち変曲点を持つような姿勢を取ることが可能になる.
- **多くの筋による駆動**
  多節構造にして構造の変形を多様にし, 各節が一様に変形するような姿勢のみならず変曲点を有するような姿勢も実現するためには, 途中節に留められるような筋を持つ必要がある. そして各節が同じような変形をする場合は, これらの筋は協調しあうことができる. すなわち, 多くの筋により駆動することにより, 姿勢の多様性と, 干渉駆動 [14, 15] のような筋力の協調が可能になる.

こうした項目を考慮して, 全身型ロボットの脊椎構造の基本構造は,

- 各節が二つの臼の間に球を挟む代わりに, 二つの球面凹みの間に球が挟まる形.
- 複数の直列に接続された節からなる多節構造

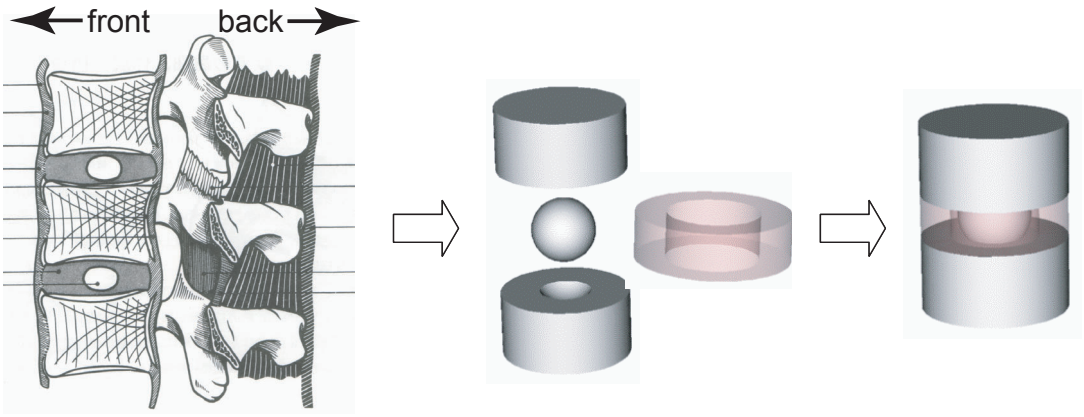



**Fig. 3.16** 全身型ロボットのための脊椎の基本型
The basic spine structure for whole-body robots

- 弾性要素 (椎間板) は，プリテンションでなくプリコンプレッションがかかるようにする．プリテンションは弾性要素と剛体材料の固着に伴う，固着が弱い・分解できないなどの問題がある．プリコンプレッションはプリテンションと同様の特性 (基本姿勢への復元力) を実現する．

とした．全身型ロボットのための脊椎の基本構造を **Fig. 3.16** に示す．

### 3.4.3 変形・柔軟性調節の両自由度を拡大した脊椎構造実現へのアプローチ

人間の脊椎構造に示唆を得て，以下に挙げる三段階の脊椎の製作を通して，変形の自由度及び柔軟性調節の自由度を拡大した脊椎構造を実現した．

1. 変形自由度と簡易な柔軟性調節機能を持つ脊椎構造

   **変形部の機構** 球面関節 (回転 3 自由度)×5 節．

   **駆動方式** 6 本の筋による拮抗駆動．

   **柔軟性調節** 張力センサを備えないがゴムの非線形性を利用した簡易な柔軟性調節機能．

2. 張力制御可能な筋を持つ脊椎構造

   **変形部の機構** 球面関節 (回転 3 自由度)×5 節．

   **駆動方式** 8 本の筋による拮抗駆動．

   **柔軟性調節** 各筋は張力センサを持ち張力制御可能．

3. 中間節へ接続する筋を有する脊椎構造

**変形機構** 球面関節 (回転 3 自由度)×10 節.

**駆動方式** 40 本の筋による拮抗駆動.

**柔軟性調節** 各筋は張力センサを持ち張力制御可能.

**駆動方式** 40 本の筋による拮抗駆動.

**姿勢の多様性** 変曲点を持つような姿勢，例えば腹部を引きながら肩の姿勢は水平を保つような姿勢が可能になる.

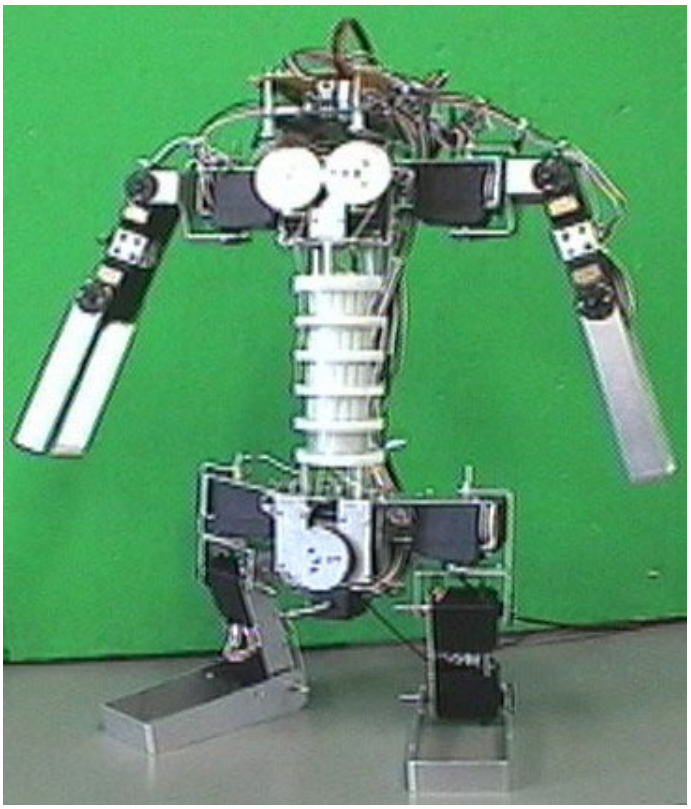


**Fig. 3.17** 柔軟な脊椎を持つ全身型ロボット Rabbit
A flexible spine whole-body robot: Rabbit

## 3.5 変形自由度と簡易な柔軟性調節機能を持つ脊椎構造 —Rabbit の脊椎 —

変形の自由度と柔軟性調節の自由度を拡大した脊椎構造実現へのアプローチ (3.4.3節) の第一段階として，変形自由度と簡易な柔軟性調節機能を持つ脊椎構造を製作し，この脊椎を組み込んだ四肢を持つ全身型ロボット “Rabbit” を製作した [9, 10]( **Fig. 3.17** ).

### 3.5.1 脊椎の構造

#### 機構の概要

製作した脊椎の構造及び外観を **Fig. 3.18** に示す．各節前後・左右・ねじりの回転 3 自由度の球面関節構造で，これを直列に 5 節積み重ねた形で脊椎機構が成り立っている．これを 6 本の筋により駆動する方式の脊椎構造である [9, 10]．球面関節には，エンジニアリングプラスチックを半球形に NC 加工した物を使用した ( **Fig. 3.19** ).

可動部の長さは 120[mm]，直径 50[mm]，重量 650[g] である．体幹部は生物の椎骨を模し

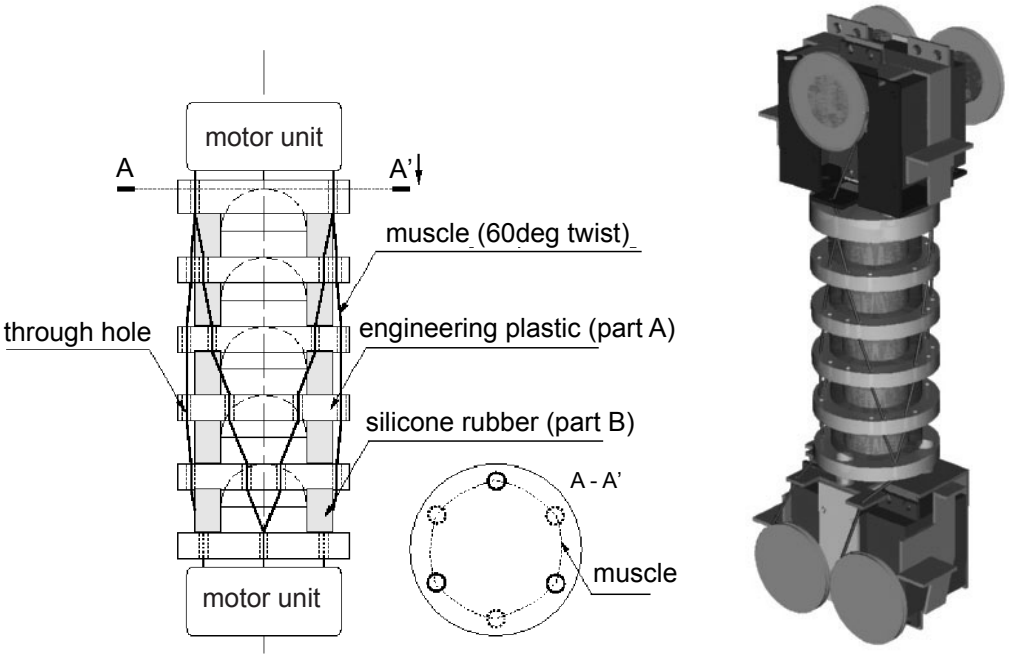



**Fig. 3.18** Rabbit の脊椎の構造
The structure of Rabbit's spine

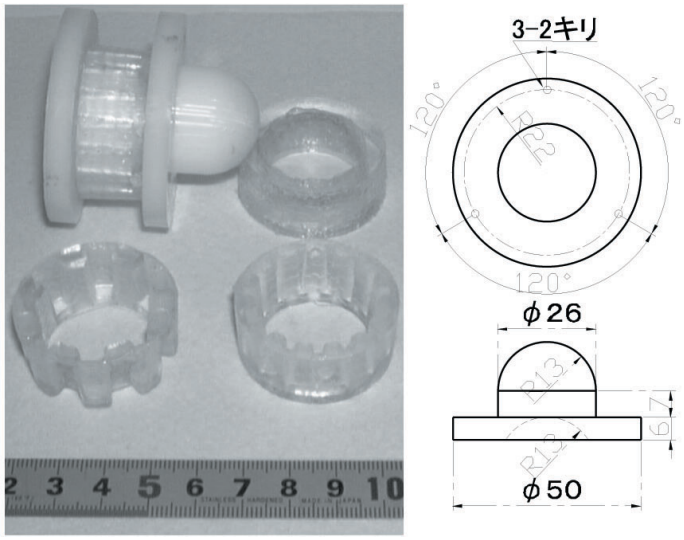



**Fig. 3.19** 球面関節とシリコンゴム製の弾性要素
Ball joint and elastic element made of silicone rubber

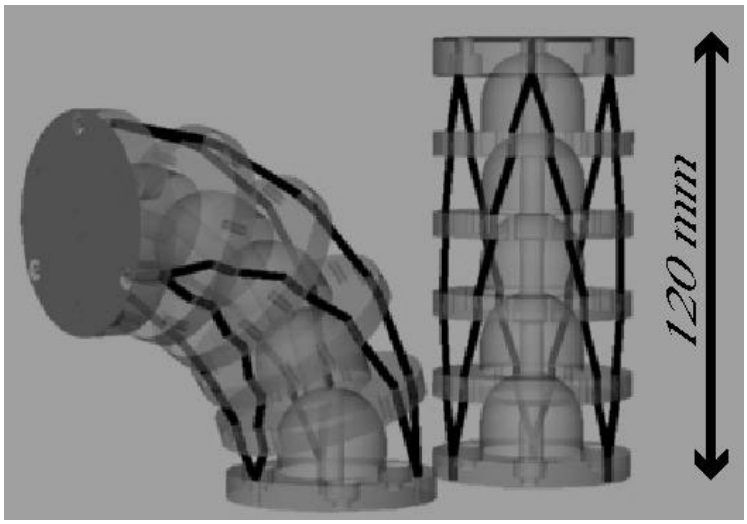



**Fig. 3.20** Rabbit の脊椎の筋配置
The arrangement of muscles (tendons) of Rabbit's spine

た剛体部品 A を 5 個直列に配置し，各剛体部品の間にシリコンゴムを用いた円環形状の弾性部品 B(椎間板) を挟んだ構造を成している．部品 A は半球状の凸面部を上面に，同半径の半球状の凹面部を下面に持つ．凸面部と隣の部品の凹面部とを組み合わせることで，球面関節を構成する．

両端の部品 A と反対側に配置されたモータユニットとを 6 本の筋で結び，モータユニットで牽引し，各筋の長さを独立に制御できる機構になっている．全ての筋は，両端の部品の脊柱まわりに 60° ずつずれた部位を結び，中間の部品 A の周囲にあけられた通過穴を通して，均等に円周上に配置されている．中間の部品は通過穴を通る筋によってのみ拘束される．

駆動方法には，筋とラジコンサーボモジュールの組合わせを採用した．球面関節の上下にある，円形の椎骨の円周上にあけた穴に，6 本の筋を通した．体幹の両端にラジコンサーボモジュールを配置し，これらで 6 本の筋を独立に引き，変形と柔軟性を調節した． 6 本の筋は 60° の傾きを持たせて等間隔で配置した ( **Fig. 3.20** )．これによって，少ない筋で前後，左右，ねじり方向に能動変形できる．また，すべての自由度に対して筋が拮抗するように配置し，柔軟性を調節可能にした (3.5.3節参照)．

サーボモジュールは， Futaba S9204 のギヤ比を調節したものを利用した．可動範囲は 180° である．脊椎の可動範囲が roll,pitch,yaw 各軸とも全体で 90° となるように筋のストロークを 60[mm] とし，筋の牽引用プーリは直径 20[mm] とした ( **Fig. 3.21** )．これにより体幹の変形可能範囲は確保できたが，プーリの径が大きくなったため筋の張力は比較的小さくなった．筋の最大張力は，サーボモジュールの最大トルク (12.6[kgf·cm]) から計算すると，

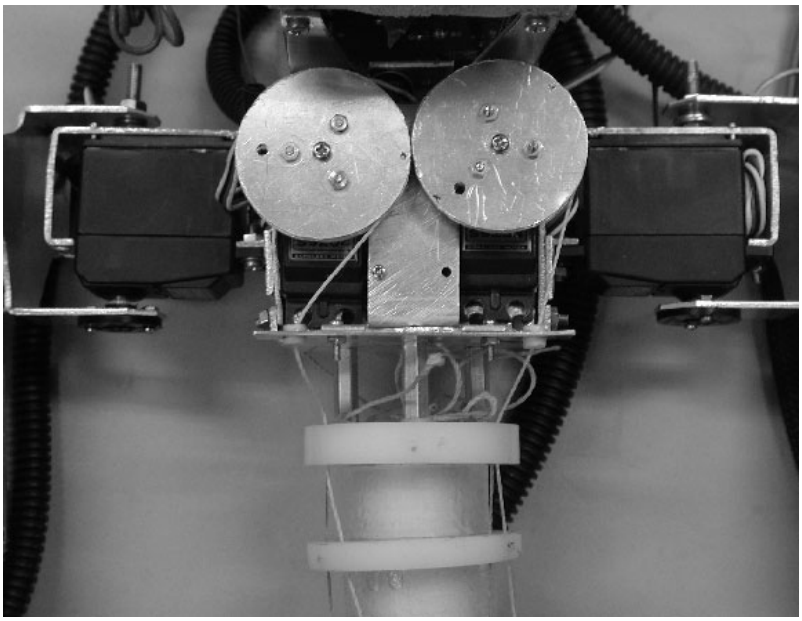


**Fig. 3.21** Rabbit の脊椎を駆動するプーリ
The pulleies of Rabbit's spine

12.6[kgf] である.

この体幹構造では各 3 自由度の球面関節を 5 個配置しているため，体幹のとりうる姿勢には 15 自由度あるが， 6 本の筋だけではそれらをすべて制御できない．そこで，各椎骨間にシリコンゴム[2)]で成形した椎間板 ( **Fig. 3.19** ) を組み込み，球面関節に弾性要素を持たせた．各椎間板に常に圧縮荷重があるように，椎間板は椎骨間隔より大きく設計した．この設計により，各関節は，関節に加わるトルクと弾性反力が釣り合うように変形することになり，少ない筋での拘束が可能である．また，筋を拮抗させて両側から引くことで，椎間板の圧縮荷重を変化させることができる．これによって体幹全体の柔軟性を調節することが可能になった (3.5.3節参照).

### 筋の配置方法

脊椎全体を前後・左右・ねじり方向に姿勢変化させることができるように，各筋は脊椎の片端から反対側の端まで 60° のねじれを持ち，脊椎に巻き付くような形に配置してある．途中の各節では，それぞれ隣り合う節の穴の位置に対して 12° のねじれを持つような位置に椎骨の穴を配置してある (正確には， 60° 以外のねじれ配置も可能なように，いくつか使われない穴もあいている)． 6 本の筋のうち 3 本は時計回りに，残りの 3 本は反時計回りに配置した．脊椎

[2)] 信越シリコーン KE-106, 硬さ (JIS-A)50, 伸び 120%

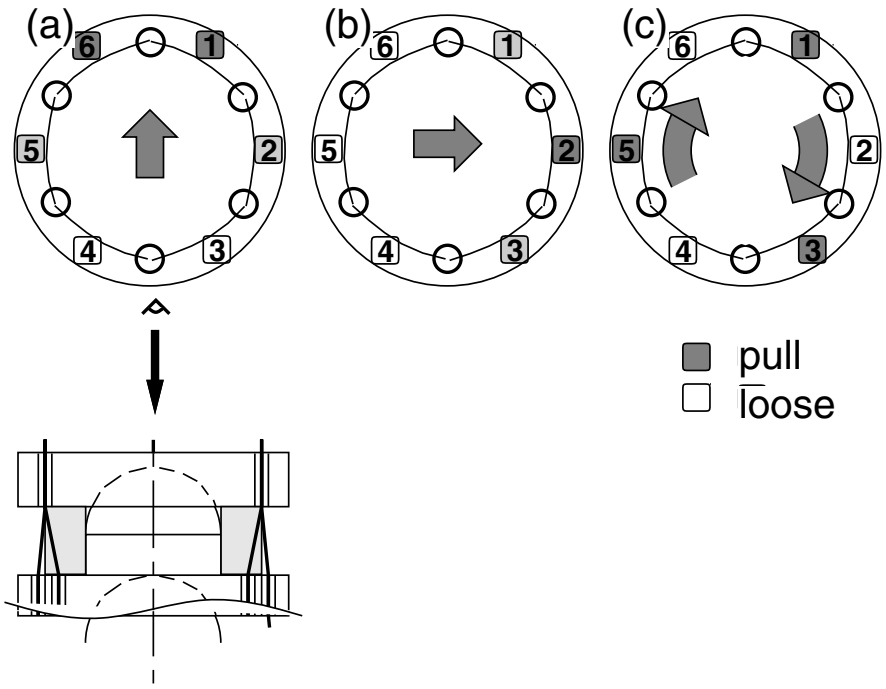



**Fig. 3.22** 筋の伸縮による姿勢の変化
Changing postures by pulling and loosening the muscles

を pitch,roll,yaw 各軸周りに姿勢変化させる際の筋の駆動方法を **Fig. 3.22** に示す．それぞれ

**(a)** 前面の筋 (3,4) を伸ばし，後面の筋 (1,6) を引くことによる，後ろ方向への変形

**(b)** 左側面の筋 (4,5,6) を伸ばし，右側面の筋 (1,2,3) を引くことによる，右方向への変形

**(c)** 反時計廻りにずれた筋 (2,4,6) を伸ばし，時計廻りにずれた筋 (1,3,5) を引くことによる，ねじり変形

を示している．

### 3.5.2 全身型ロボット Rabbit への組み込み

#### 二脚直立状態と四脚状態

3.5.1節に述べた脊椎構造を，四肢を持つ全身型ロボット Rabbit に組み込んだ ( **Fig. 3.17** ). 全身型ロボットとする際に，脊椎構造の利用法・利点として以下に挙げるような点を考慮した．

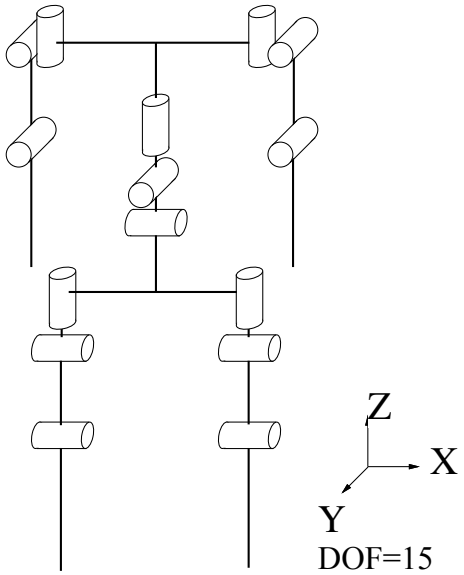



**Fig. 3.23** Rabbit の自由度配置
DOF arrangement of Rabbit

- 二脚直立状態から体幹を曲げる，もしくはねじることで上半身の取り得る姿勢を増やすことができる．これは上肢の自由度が十分に多くない場合に，その自由度不足を補い，多用な動作を可能とする．

- 傾いた場所や不安定な場所で二脚直立する必要がある場面において，体幹を変形し，上半身の重心を移動することができるということは，安定した姿勢をとる上で重要である．下肢だけを用いたバランス補償に比べて，体幹の重心移動を用いたバランス動作は，下肢に負担をかけない．

- 四脚で歩行する時に，上肢と下肢とを支える体幹部を変形させることで，四肢の付け根が取り得る位置関係を変えることができる．これにより，従来の 4 脚歩行よりも歩幅を広げる，不整地に対する適応性を上げるなどの利点が生まれる．

これらの 3 つの利点のうち先の 2 つは，二脚直立状態で上半身が自由に動かせるときに顕著に現れる．またの最後の 1 つは四脚歩行において上肢と下肢との位置関係を変えることが必要なときに有効である． Rabbit においては，二脚直立状態と，四脚歩行状態とを併用することを想定した四肢の構造にした．

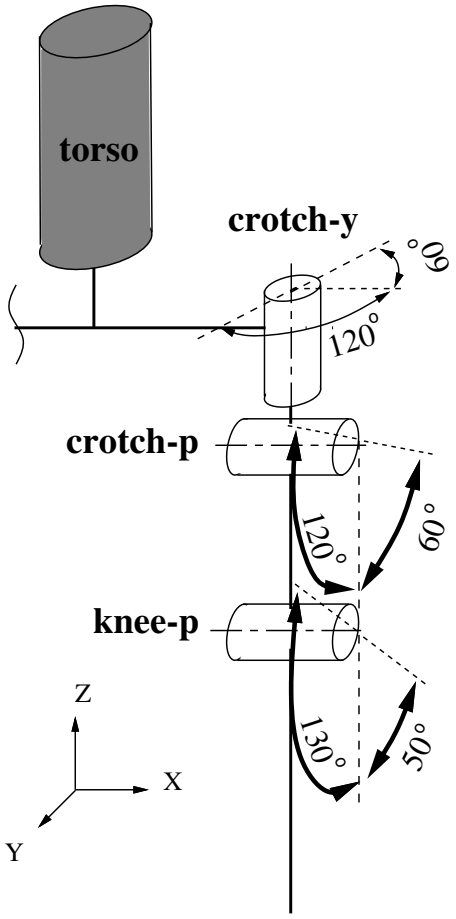



**Fig. 3.24** Rabbit の下肢の自由度
DOF of a lower leg of Rabbit

### 四肢の構造

ロボット全体の自由度配置図を **Fig. 3.23** に示す．自由度配置は各脚に 3 自由度，体幹部に 3 自由度を配し，ロボット全体で 15 自由度を有する．四脚ほふく状態の肩高は 200[mm]，二脚直立状態の体長は 410[mm]，総重量は 2.0[kg] である．

下肢の関節名と可動範囲を **Fig. 3.24** に示す．この構造は， crotch-x 関節を背側に 30° 回転させ， knee 関節を腹側に 120° 回転することで，二脚座姿勢 ( **Fig. 3.25** a) が可能である．これにより，上半身を自由に動かす動作が可能である．また， crotch-x 関節を腹側に 90° 回転し， knee 関節を伸ばし，上肢を前方に伸ばすことで四脚歩行状態 ( **Fig. 3.25** b) が可能である．

各四肢の関節駆動のためのアクチュエータには，脊椎の筋駆動用モータと同型の模型用サーボモジュール (Futaba S9204 ギヤ比調節) を採用した．各サーボモジュールの可動範囲は

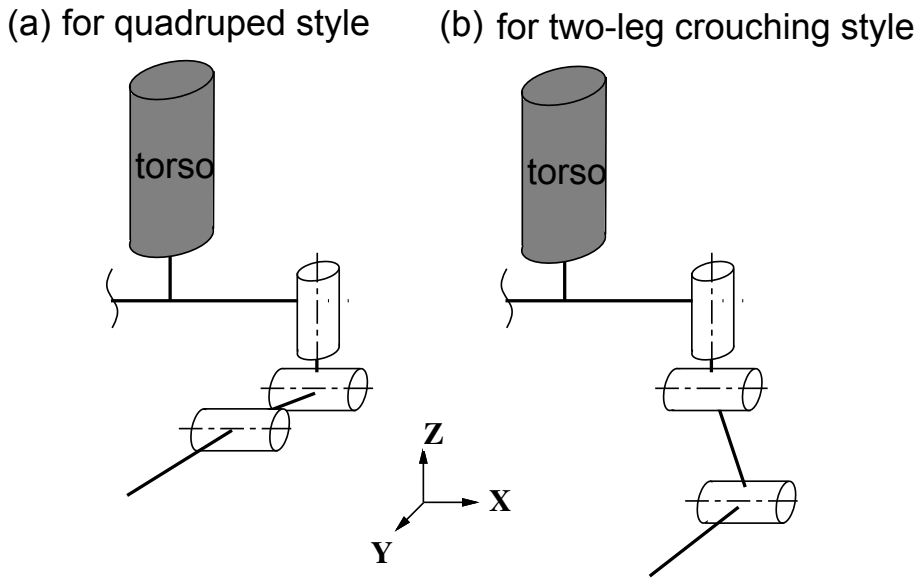



**Fig. 3.25** 二脚座状態と四脚歩行状態を使い分ける下肢
The lower leg is used for either two-leg sitting state and four-leg walking state

180° ある.

#### センサとシステム構成

システム構成は，オンボディのワンチップマイコンとリモートのワークステーションをシリアル通信で結びセンサ情報とアクチュエータ情報を送受信する形を採った．ロボット搭載用小型ワンチップマイコンボードを設計・製作した.

センサは，脊椎の上部および下部に 3 軸加速度センサ (Analog Devices, ADXL05EM-3) を取り付けた．この加速度センサを利用することで，上半身および下半身の加速度を計測する．動作中以外なら重力加速度の方向を計測することができ，上下の二つの加速度センサの出力の差分を取ることで，脊椎全体の姿勢変形を検出することができる.

### 3.5.3 Rabbit の脊椎の柔軟性調節

開発した脊椎機構は，外部からの力に対し柔軟に変形可能である．これは生物の脊椎の機能の一部であり，衝撃吸収などの場面において全身型ロボットに応用すべきものである．しかし，全身型ロボットの脊椎構造として組み込むためには，ただ柔らかいだけではなく，必要に応じて堅くなることができるという柔軟性調節機能が必要である．この機構全体の柔軟性は弾性部品の弾性係数によって決まる．よって，この脊椎構造で柔軟性調節機能を実現するために

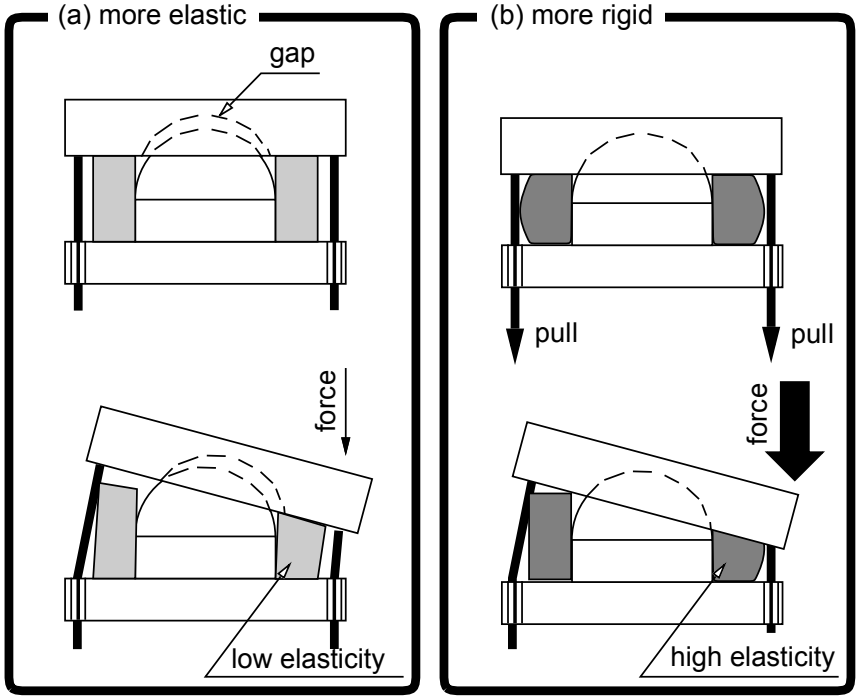
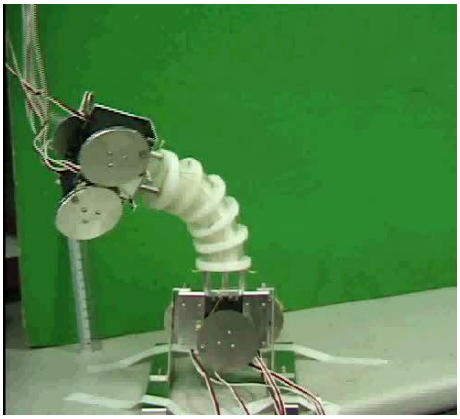



**Fig. 3.26** Rabbit の脊椎機構の柔軟性調節
The change of elasticity of Rabbit's spine

は，弾性部品の弾性係数を能動的に調節すればよい．本研究で椎間板の材料として使用した弾性部品であるシリコンゴムの弾性は非線形であり，変形が大きくなるほど弾性係数が大きくなる．この性質を利用して弾性係数を能動的に調節する方法を以下に説明する．

まず弾性部品の脊椎軸方向寸法を，節の間隔よりも大きく設計し，節を構成する剛体部品に挟む ( **Fig. 3.26 (a)**) 形にする．これにより，球面関節に隙間が生じる．この状態に外部から力を加えた場合，弾性部品はその自然長付近で変形し，前節で説明したような挙動を示す．これを挙動 1 とする．次に体幹を硬化させる必要がある場面では，6 本の筋を全て引き，弾性部品に初期変形を与える ( **Fig. 3.26 (b)**)．この状態で外部から力を加えた場合，弾性部品はすでに圧縮されているため，挙動 1 の場合よりも多く圧縮される．これを挙動 2 とする．シリコンゴムの非線形性のため，挙動 2 における弾性部品の弾性係数は挙動 1 におけるそれよりも大きい．その結果挙動 2 の脊椎は，挙動 1 の脊椎よりも硬くなる．

Rabbit の実験においては，この機構を用い，場面に応じた柔軟性の調節を行なった．衝撃吸収や外界への適応などの，柔軟な体幹が要求される場面では，各弾性要素の初期変形を少なくするために，関節間隔が大きいものとして，全ての筋長を長めに計算し，制御する．逆に構造体としての，堅い体幹が要求される場面では，各弾性要素の初期変形を多くするために，全



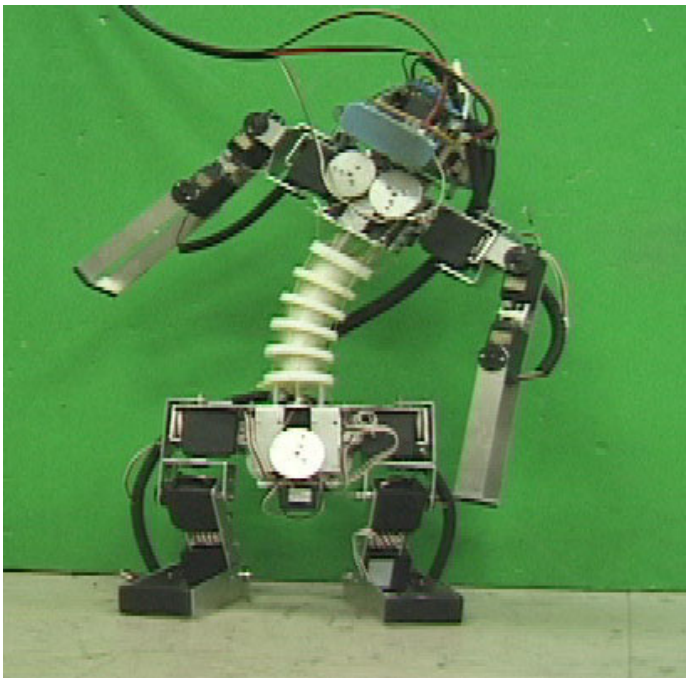


**Fig. 3.27** 脊椎の旋回動作
Rotation of the spine

ての筋長を短く計算して，制御する．このように，筋の長さ制御だけを用いて，脊椎全体の柔軟性を制御することができる．

拮抗駆動と非線形ばね要素を用いた柔軟性調節機構の研究は，これまでに様々な形で行われている [17, 81, 85, 18]．これらは拮抗筋に非線形バネを組み込んだ柔軟性制御であるが，ここでは筋ではなく回転関節に，非線形バネ要素を組み込まれていると捉えることができる．

### 3.5.4 Rabbit の脊椎の動き

Rabbit の脊椎の動きの様子を **Fig. 3.27** に示す．脊椎構造が, 上半身の重量を十分支えて旋回を行うことができることを確認した.

さらに，柔軟性を調節する実験を行った．脊椎を直立させた状態で 6 本の筋を引き，弾性部品に初期変移を与えた．これによって，各関節の弾性係数が変化し，外部からの力に対しての剛性が増す．このことを脊椎を横から押すことで確認した ( **Fig. 3.28** ).

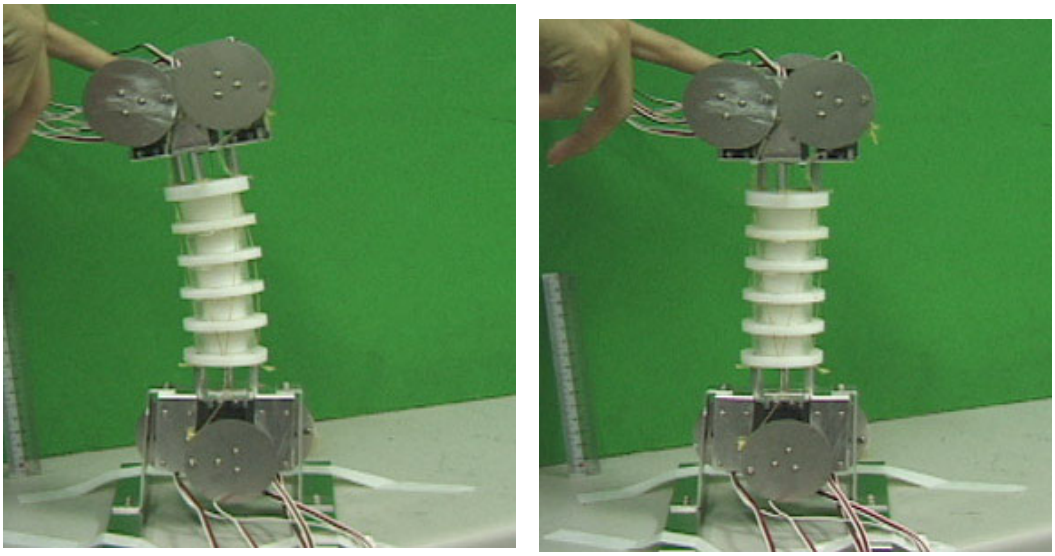


**Fig. 3.28** 筋長調節による柔軟性変化 (左: 柔，右: 剛)
Changing softness by changing muscle-lengths (left:soft, right:hard)

全身を利用した動作や行動に関しては，第 5章で詳しく述べる．

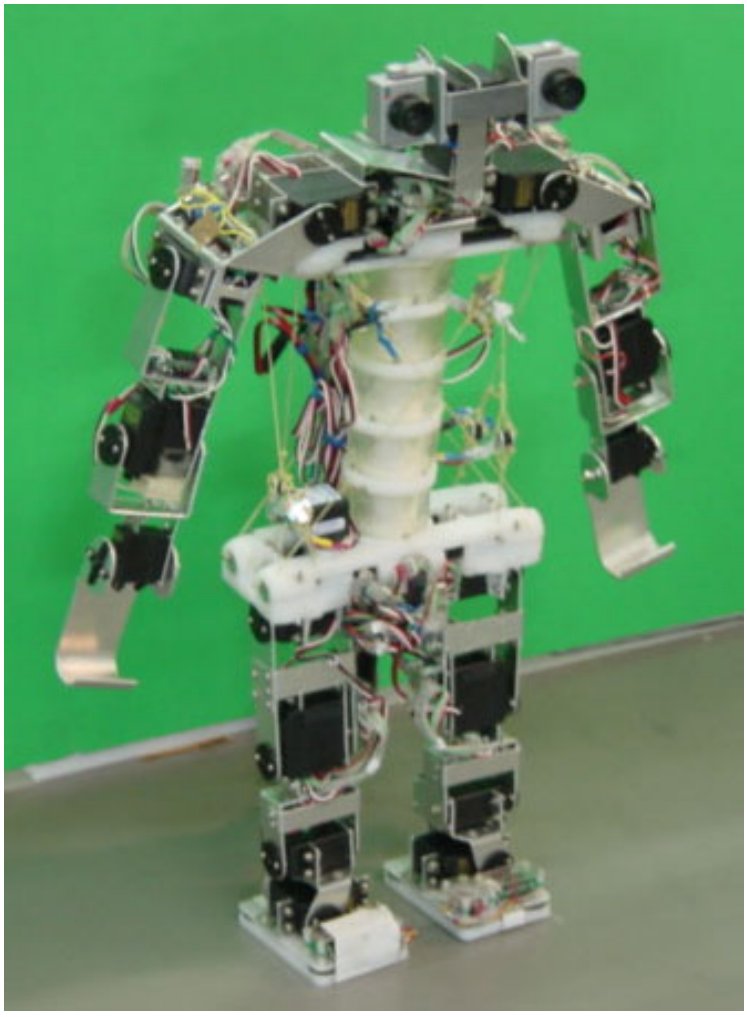


**Fig. 3.29** Cla の全身
Whole body of Cla

## 3.6 張力制御可能な筋を持つ脊椎構造 —Cla の脊椎 —

変形の自由度と柔軟性調節の自由度を拡大した脊椎構造実現へのアプローチ (3.4.3節) の第二段階として，張力制御可能な筋を持つ脊椎構造を製作し，この脊椎を組み込んだ二足二腕を持つ人間型ロボット “Cla” を製作した [59]( **Fig. 3.29** ).

### 3.6.1 脊椎の構造

製作した脊椎の構造を **Fig. 3.30** に示す [58]．脊椎の構造は Rabbit の脊椎 (3.5.1節) と類似した機構を持つ．特徴を以下に説明する．

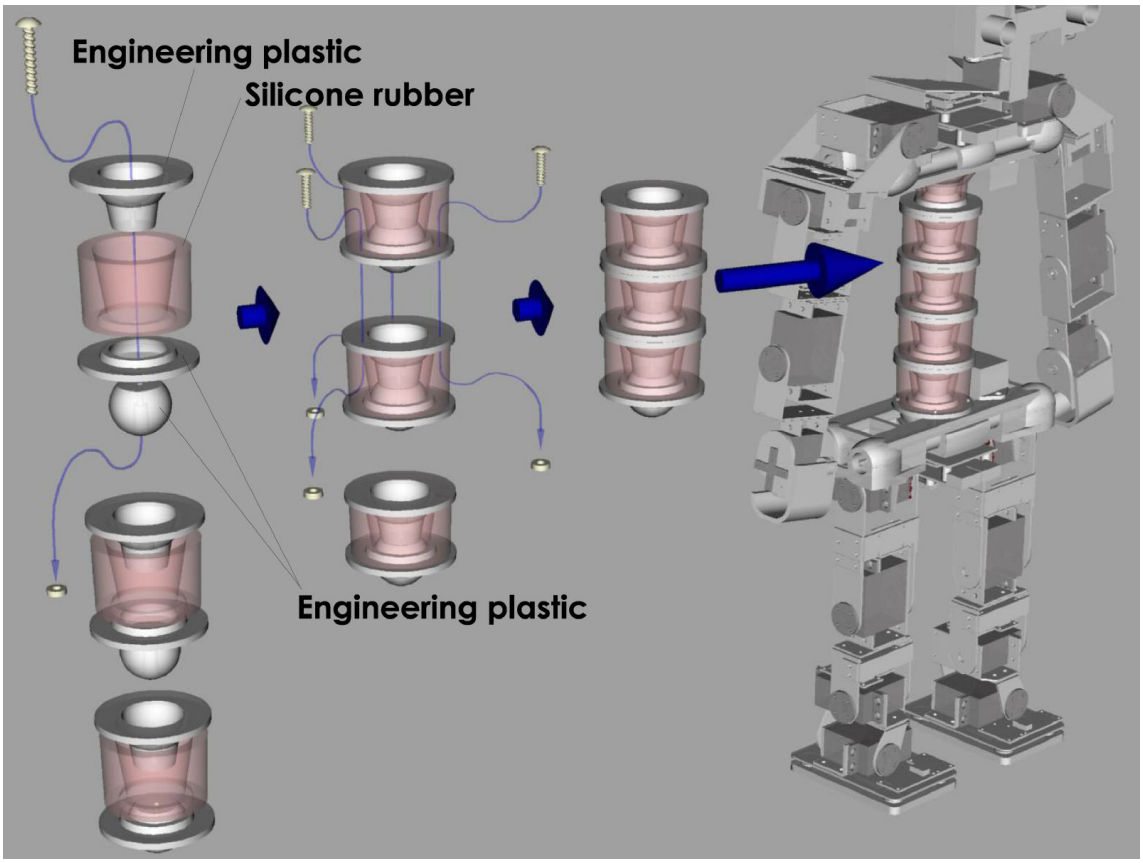



**Fig. 3.30** Cla の脊椎構造
The structure of Cla's spine

### Rabbit の脊椎と同様な点

回転 3 自由度を持つ球面関節からなる節が 5 節直列に積み重なる構造である．各節の構造部材はエンジニアリングプラスチックを NC 加工，椎間板となる弾性部材はシリコンゴム[3)]を型成形したものである．シリコンゴムの椎間板は，プリコンプレッションがかかるように節間の隙間より大きい．可動範囲は，各節各軸 (roll,pitch,yaw)±18° 程度で，脊椎全体で roll,pitch,yaw 各軸 ±90° 程度の可動範囲を持つ．

### 筋の本数・配置

脊椎を駆動するための筋は 8 本とした．脊椎全体が前後・左右・ねじりの回転 3 自由度と前後・左右の並進 2 自由度の計 5 自由度であると考えると，これを駆動する筋は最少で 6 本必

[3)] 信越シリコーン KE-109, 硬さ (JIS-A)20, 伸び 150%

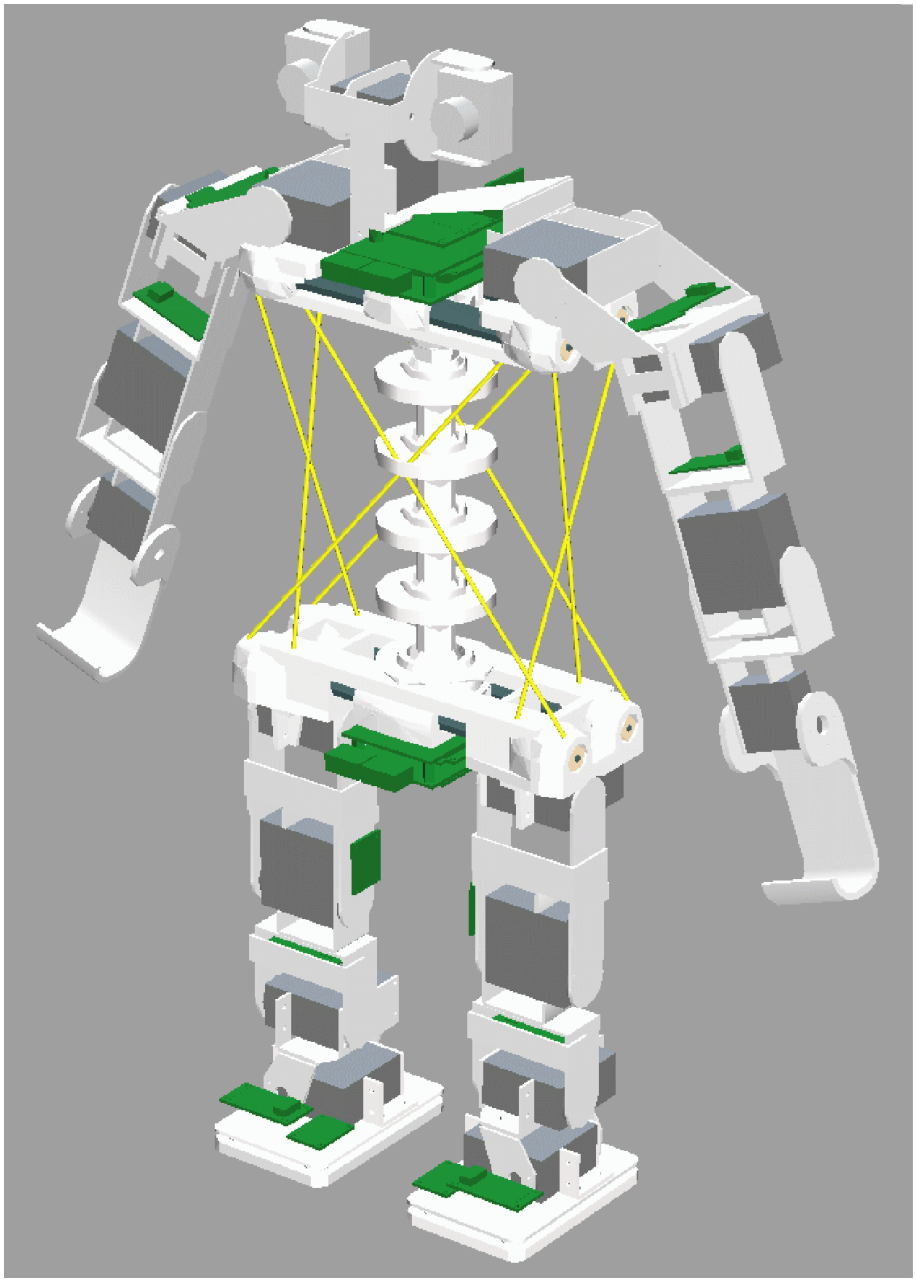


**Fig. 3.31** Cla の筋配置
The arrangement of Cla's muscles

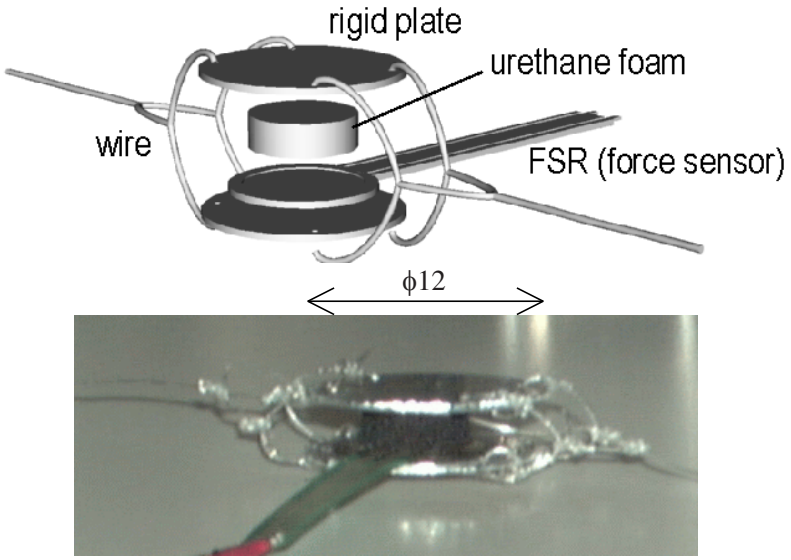



**Fig. 3.32** Cla の筋の張力センサの構造
The structure of developed tension sensor for Cla's muscles

要である．各節前後・左右・ねじりの回転 3 自由度で 5 節脊椎全体では 15 自由度であると考えると，これを駆動する筋は最少で 16 本必要である． 3.4.1節で述べたように，筋が足りなくても，外力が無ければ椎間板の復元力と筋張力が釣り合いにより姿勢が決まる．

8 本という本数の理由は,

- 筋の本数が多ければ多いほど，姿勢制御がきめ細かくできるようになる，すなわち多節構造の姿勢の多様性を制御しやすくなる (3.4.1節参照),
- Cla の肩骨と腰骨の形状を上から見て幅広で長方形に近い形状であるため (下記**筋の引っ張り位置**の項参照)， Rabbit の脊椎の両端のような点対称な形状ではなく線対称な形状になり，計 6 本 (片側 3 本) の配置より各端で偶数本の筋を牽引する方が自然な配置になる．
- 少ない本数でも (外力無しであれば) 不安定にはならないので，脊椎の制御法の研究におけるテストベッドとしての利用を想定すると，できるだけシンプルな構造の方が良く，筋は少ない本数の方が望ましい．しかし， 4 本では roll,pitch,yaw 各軸を制御できる最少本数であるため，最適配置でなければならない．また， x,y(前後・左右) の並進などの制御を不可能にする可能性がある．

などである．

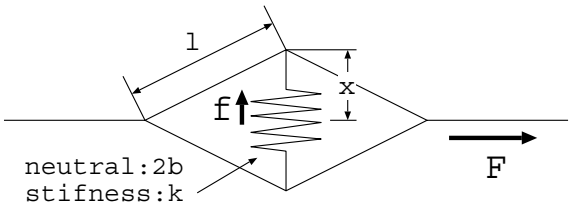



**Fig. 3.33** 筋張力と圧力センサにかかる力の関係
The relation between the muscle's tension and the force on pressure sensor

また，脊椎全体で yaw 軸回りの姿勢変形を可能にするために，体幹の前後・左右に筋を体幹を横切るように斜めに配置した． **Fig. 3.31** に Cla の筋の配置を示す．

各筋は張力センサを備え，ソフトウェアにより張力制御が可能である． **Fig. 3.32** に示す構造の張力センサを製作した．並行平板の間に弾性体と圧力センサ (FSR:force sensing register) を重ねて挟み，ワイヤ張力に応じて並行平板の間隔を縮める方向に力がかかる．力が強くなってもワイヤの伸びに上限がある構造である．ワイヤ張力 $F$ と圧力センサにかかる力 $f$ の関係は， **Fig. 3.33** より，

$$f^2 l^2 = (F^2 + f^2)(b - f/k)^2 \tag{3.3}$$

となる．張力が小さいほど解像度が高く，また筋も伸びやすい (剛性が低い)．逆に張力が大きくなるほど解像度が低く，また筋も延びにくい (剛性が高い) という特性である．

### 張力センサ

直径約 8mm(#400),18mm(#402) の 2 種類のセンサの張力と 10bit,5V の AD 変換の値の計測結果を **Fig. 3.34** に示す．センサ出力は張力を加え始めてから 10 秒後のデータを採った．小さい力の領域で分解能が高く大きい力の領域で分解能が粗い事がわかる．経時変化もあり絶対値の高精度は得られないが，基本姿勢の調整や各腱張力バランスなどように，各センサの出力の相対的関係を利用するには適用可能であると考えられる．

### 筋駆動用アクチュエータ

筋の駆動には， DC モータとプーリを利用した．モータとしては， maxon motor 製の定格出力 2.5[W] の DC モータを遊星ギヤヘッド・ロータリエンコーダと組み合わせたものを利用した．プーリは，内径 6[mm] 外形 12[mm] のものを製作した． Cla の脊椎の筋駆動系の主な特性を Table 3.2 に示す．

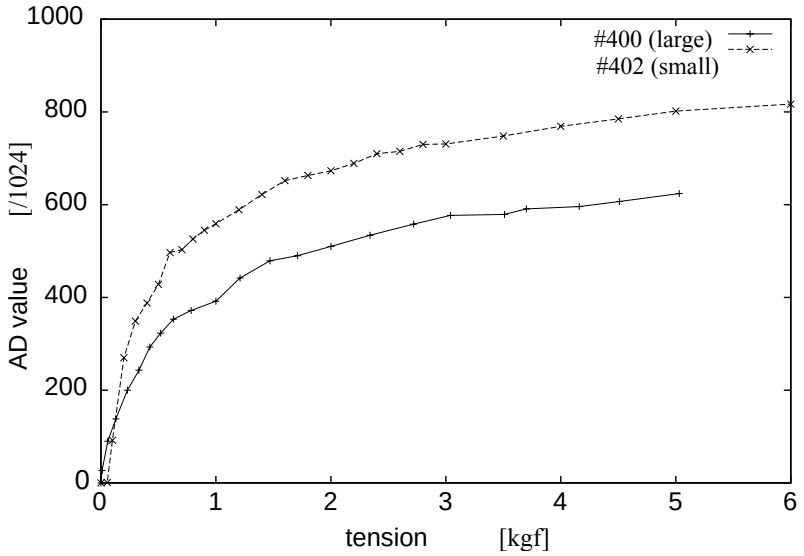



**Fig. 3.34** 張力センサの特性 (実際の張力と力センサの値 (AD) 出計測) との関係
The relation between the tension and force sensor's value (AD)

Table 3.2 Cla 脊椎の筋駆動系の主な特性
The specification of the actuator of the muscles of Cla's spine

| | | |
|---|---|---|
| モータシリーズ | RE$\phi$13mm, 貴金属ブラシ, 2.5Watt | |
| 定格出力 | 2.5 | W |
| 公称電圧 | 24.0 | V |
| 無負荷回転数 | 17800 | rpm |
| 停動トルク | 14.2 | mNm |
| ギヤ比 | 67.49 | |
| 無負荷回転数 (ギヤ後) | 264 | rpm |
| 停動トルク (ギヤ後) | 0.958 (9.78) | Nm (kgf·cm) |
| エンコーダ分解能 | 16 | カウント / 回転 |
| プーリ内径 | 6 | mm |
| 最大筋張力 | 320 (32.6) | N (kgf) |

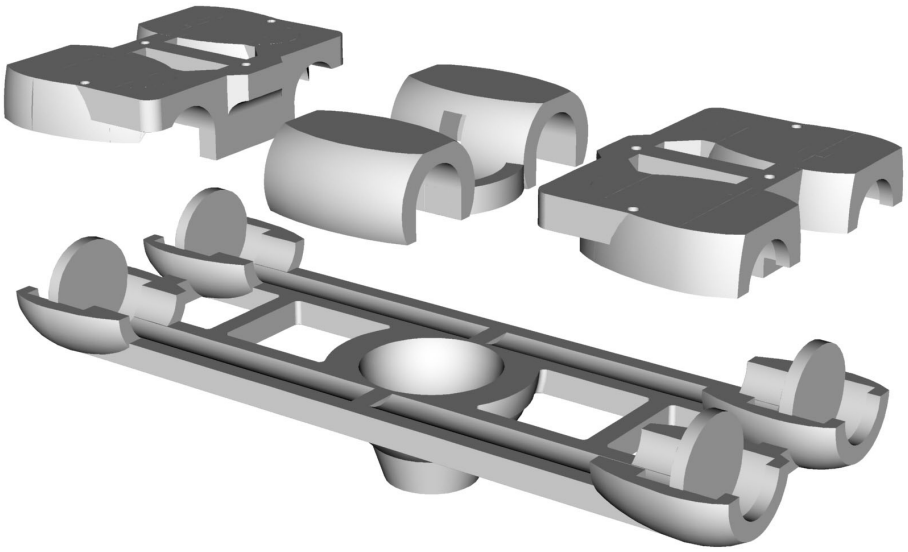


**Fig. 3.35** 鎖骨部品
Clavicle (collarbone) part

各アクチュエータは，オンボディのワンチップマイクロコントローラにより制御する．各筋に取りつけた張力センサの信号も同じマイクロコントローラに内蔵する AD 変換器で読み取ることにより，制御ループの短い (1[ms] など) 制御が可能になる．

### 筋の取り付け位置・引っ張り位置

幅の広い肩骨 (clavicle) と腰骨 (pelvis) を持つ．肩骨の三次元 CAD モデルを **Fig. 3.35** に示す．肩骨・腰骨にそれぞれ 4 本の DC モータを横向きに埋め込み，内径 6[mm] 外形 12[mm] のプーリを介して肩・腰の両端から筋を引く構造になっている．各筋は脊椎の前面，後面，側面のいずれかを斜めに横切り，肩のモータにつながる筋は対角線上の腰骨の端付近に取り付けられている．これにより，脊椎の回転中心から筋を離した構造になり，脊椎全体の変形力は脊椎の中心付近を筋が通る Rabbit の脊椎のような構造と比較して大きくなっている．なお，その際，必要な筋の引っ張り速度とストロークは，比較的大きくなる．筋は，当初脊椎構造のみの姿勢制御実験の際には最大張力 12[kgf] のテグスを使用したが，全身を持つ人間型ロボット “Cla” の形に組み上げてからは最大張力 60[kgf] のケブラーストリング (ポリアミド繊維) を使用している．

#### 各節の構造

Rabbit の各節は，隣り合う節同士の間の距離は一定ではなく，筋が緩めば分解してしまう構造になっているが，これは Rabbit の脊椎が張力センサを備えていないので，簡易な柔軟性調節機能を実現するためにこのような構造になっていた (3.5.3節および **Fig. 3.26** 参照)． Cla の脊椎では，球面関節の構成部品である球 (髄核に相当) が， **Fig. 3.30** に示すような構造により，隣接する部分から外れない構造になっている．すなわち，球を受ける球面状の凹みが半球より小さいと球が外れ得るので，半球より大きくなるように複数の部品とネジを用いる構造で設計し，ネジを用いて組み立てるようにした．これにより，筋に張力をかけなくても椎間板に初期圧力 (プリコンプレッション) を生じさせることができている．

### 3.6.2 人間型ロボット Cla への組み込み

3.6.1節で述べた脊椎構造を組み込んだ，全身を持つ人間型ロボット Cla を開発した ( **Fig. 3.29** ).

#### 脚腕の自由度配置

直立した状態を基本姿勢とする事を想定し，両脚は腰骨から脊椎の軸に平行な方向に伸びる形，両腕は肩の両端から脊椎の軸に並行する形とした ( **Fig. 3.30** 右端)．自由度は，脚・腕とも各 4 自由度である．自由度配置は， **Fig. 3.36** 左に示すように，脚は股関節・膝関節が pitch 軸のみで足首関節が pitch・ roll 軸，腕は肩関節が roll・ pitch 軸で肘および手首関節が pitch 軸のみとなっている．

#### 脚椀のアクチュエータ

四肢の関節駆動用アクチュエータには， Futaba 製ラジコンサーボモジュールを改造して利用した． Futaba 製ラジコンサーボモジュールには比例制御による位置制御の回路が組み込まれているが，その制御回路を取り外し，自作した小型モータドライバ JSK-D02 を搭載し，利用してワンチップマイコンによるソフトウェアサーボを行う構成にした．これにより，制御方式の変更 (PID 制御など)・制御パラメータ (ゲインなど) の動的変更・他センサを利用した多様な制御を可能にした．また，力制御などの制御周期が重要になるような制御法も可能になる．元になるラジコンサーボモジュールは， Rabbit に用いたものと同じ型番の S9204 のギヤ比を調節したものを用いた (3.5.2節).

#### センサ

Cla は以下のセンサが組み込まれている．

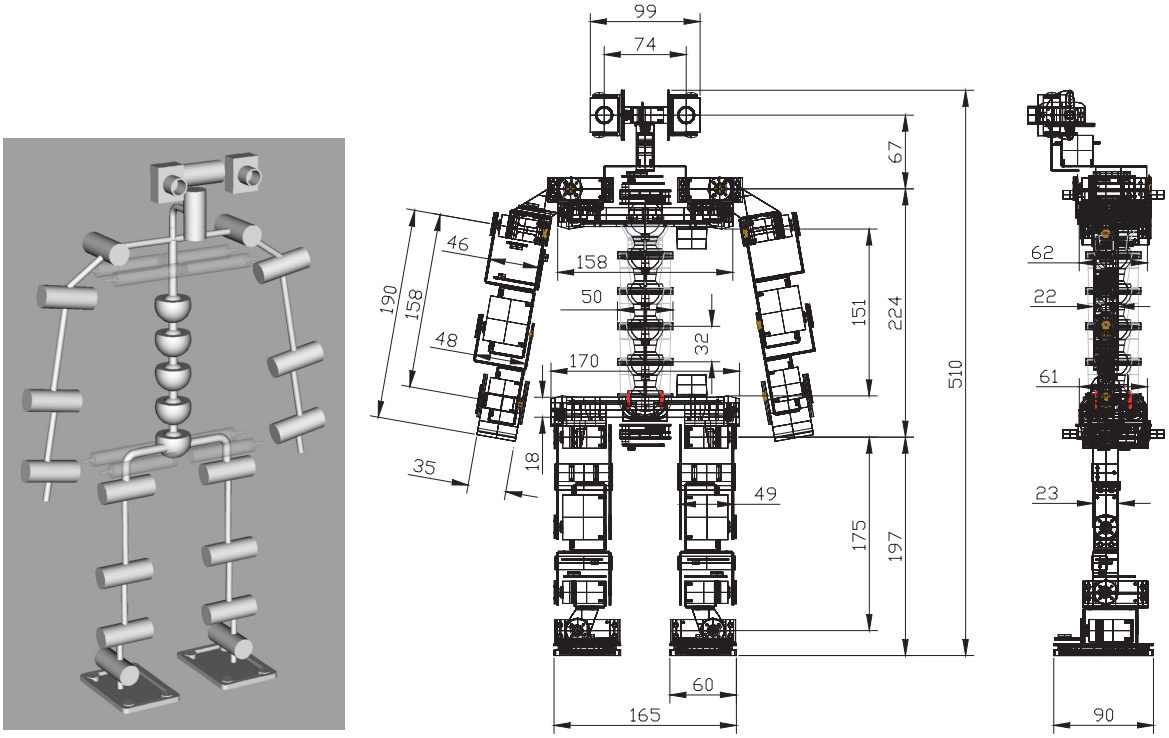



**Fig. 3.36** 左: Cla の自由度配置, 右: Cla の正面図と側面図
Left: DOF arrangement of Cla, Right: front and side view

- モータ角度検出センサ (ロータリエンコーダ・ポテンショメータ)
  脊椎の筋を駆動する各モータはロータリエンコーダが組み込まれ，手足の関節を駆動する各モータはポテンショメータが組み込まれている (手首と首を駆動するモータは角度センサの信号は取り出されていない)．ロータリエンコーダはパルスを計測することで相対角度を計測する．ポテンショメータは抵抗値の変化に応じた電圧を計測することで絶対角度を計測する．

- 張力センサ
  各筋に張力センサを組み込んだ．詳しくは 3.6.1節に述べられている．

- 3 軸加速度センサ
  腰と肩に 1 個ずつ合計 2 個の 3 軸加速度センサを搭載した．これにより重力の方向を検出することにより姿勢を計測することができる．ただし，動作中は自分の動作に起因する加速度を拾ってしまう．また，重力方向の検出なので水平面内の回転に関しては検出

できない．肩と腰に取り付けることにより，その差分を計算することで脊椎の変形量を計測することができる (4.7.2節参照)．腰・肩の 3 軸加速度センサにより計測した重力の方向をそれぞれ $\boldsymbol{g}_h$・$\boldsymbol{g}_c$ とすると，腰から見た肩の姿勢 $\boldsymbol{\theta}_s$ は次式により計算される．

$$\boldsymbol{\theta}_s \quad = \quad \boldsymbol{g}_c - \boldsymbol{g}_h \tag{3.4}$$

- 足裏力センサ

  両足にそれぞれ 4 個ずつ反力センサを搭載した．使用したセンサは FSR(Force Sensing Resister) というもので，センサ面に加わる力に応じて抵抗値が変わる． FSR と固定抵抗を直列に接続し電圧を加え分圧点の電圧を計測することで足裏力を計測する．センサ出力と力の関係は非線形なので， 10 点ほど計測しテーブルを作成し，線形補間することで力を計測する ( **Fig. 3.34** 参照)．足裏の四隅に配置することで，力のバランスから ZMP(zero moment point) を計測することができる (5.2.2節参照)． FSR の取り付け位置を $\boldsymbol{p}_1$， $\boldsymbol{p}_2$， $\boldsymbol{p}_3$， $\boldsymbol{p}_4$，検出された反力を $f_1$， $f_2$， $f_3$， $f_4$ とすると， ZMP($\boldsymbol{p}_z$ とする) は次式により算出される．

$$\boldsymbol{p}_z \quad = \quad \frac{\sum_{i=1}^{4} f_i \boldsymbol{p}_i}{\sum_{i=1}^{4} f_i} \tag{3.5}$$

- モータ電流センサ

  モータドライバ基板 JSK-D02 は，モータ駆動パワーラインに $R = 0.1[\Omega]$ の抵抗を直列に配置してあり，その電圧降下 $V$ を計測することでモータ電流 $I$ を計測することができる．

$$I \quad = \quad \frac{V}{R} \tag{3.6}$$

- CCD カラーカメラ

  頭部には 2 個の CCD カラーカメラを搭載した． PC に画像キャプチャボードを 2 枚配置し，信号を取り込む．テンプレートマッチングに基づくトラッキング・オプティカルフローの検出・ステレオ視・色検出などが可能である．

### システム構成

Cla には，合計 10 個のオンボディプロセッサを搭載した．各プロセッサは局所的にセンサ情報収集・アクチュエータ制御を行う．全てのオンボディプロセッサは $\mathrm{I}^2\mathrm{C}$-bus からなる体内

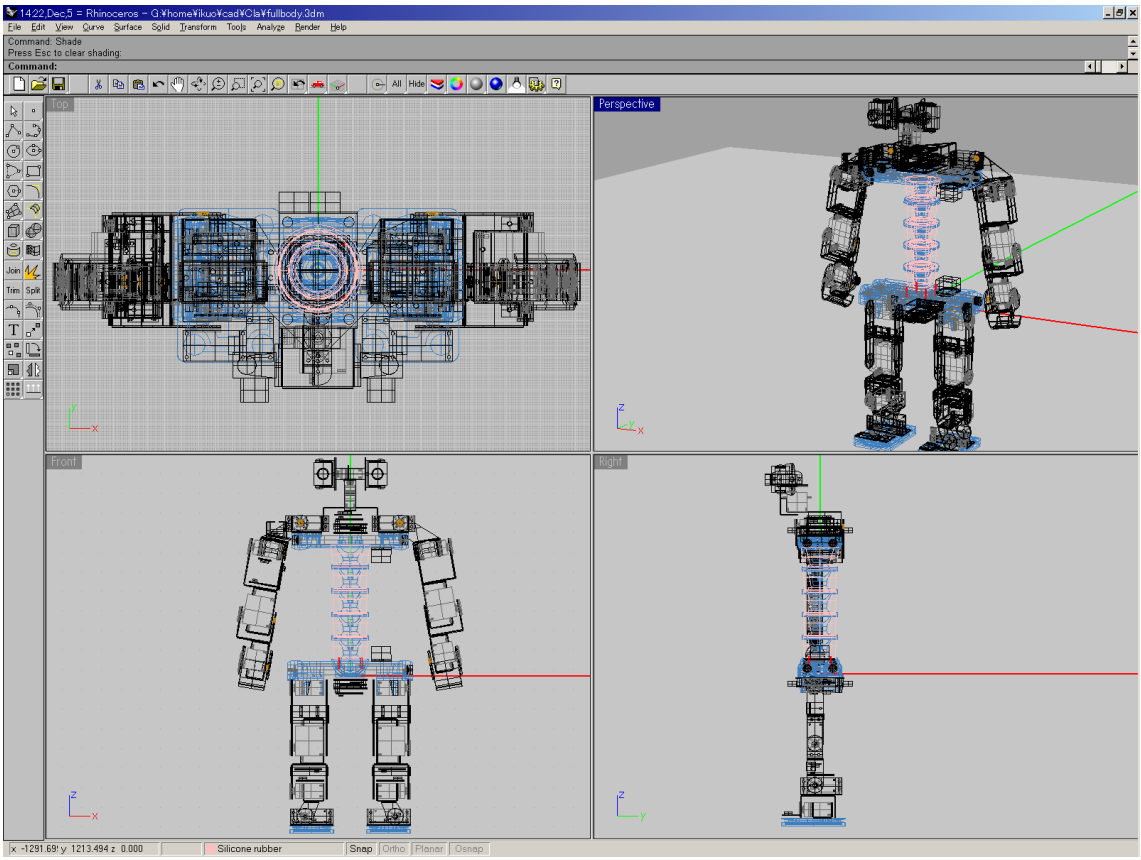



**Fig. 3.37** Rhinoceros での設計
Design on Rhinoceros

LAN に接続し，1 個の HUB プロセッサが全てのプロセッサの情報を集約し，遠隔のホストコンピュータとシリアル通信でデータを送受信する．また，小型モータドライバ基板 (1 枚で 2 モータを駆動) を 12 枚搭載し，各モータとオンボディプロセッサの間でモータの駆動を行う．システムの詳細は第 6章に述べる．

### 三次元 CAD ソフトウェアを利用した全身設計

脊椎のような複雑な立体形状の設計においては，立体形状の作成に便利な三次元のデザインツールが望ましい．ロボット “Cla” の設計においては，三次元モデリングソフトウェアである Rhinoceros [4)]を使用し，背骨の部品を含むロボット全体の設計を行った ( **Fig. 3.37** ). Rhinoceros 上で作成した三次元形状を実際に製作する際には，以下に示す二つの方法を用いた．

[4)] http://www.rhino3d.com/, http://www.rhino3d.co.jp/

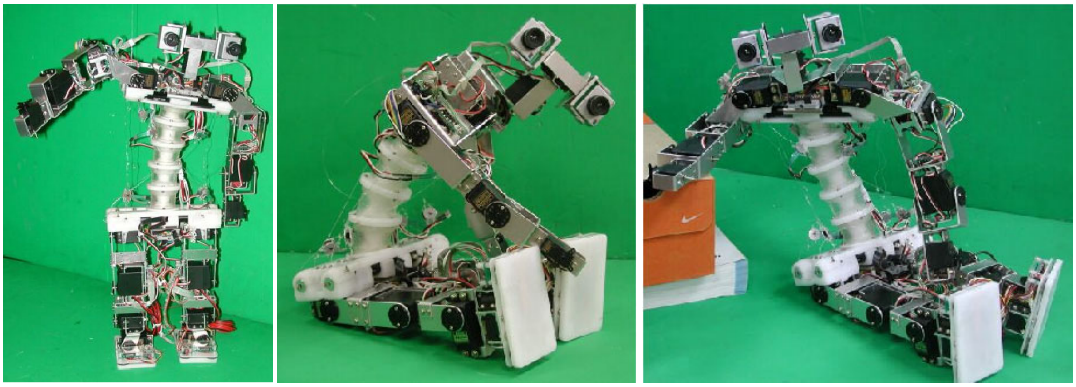


**Fig. 3.38** Cla の様々な姿勢
Various poses of Cla

**板金加工による製作** 腕のリンク部分の構造部材などの単純な形状の部品は， Rhinoceros 上の部品を現尺で印刷したものをもとにボール盤，板金折り曲げ機等を用いて製作した．

**NC 工作機械による製作** 脊椎を構成する部品などの複雑な形状の部品は， Rhinoceros から汎用な三次元形状データフォーマット (DXF 形式) で出力し，それを CAM ソフトウェアと NC フライス盤を用いて製作した．これにより，自由度の高い設計のできる三次元デザインソフトウェアで作成した形状を，できるだけそのままの形で製作することができた．例として，肩骨 (鎖骨) に当る部品の形状を **Fig. 3.35** に示す．ちなみに，ロボット Cla の名称は，鎖骨 (clavicle) が特徴的であることが一つの由来である．

### 3.6.3 Cla の脊椎の動き

Cla の様々な姿勢を **Fig. 3.38** に示す．ねじりや側屈の稼働範囲により，脊椎が無ければ不可能であるような姿勢を可能にしている．脊椎の制御法に関しては，第 4章に詳しく述べる．

## 3.7 中間節へ接続する筋を有する脊椎構造 — 腱太の脊椎 —

変形の自由度と柔軟性調節の自由度を拡大した脊椎構造実現へのアプローチ (3.4.3節) の第三段階 (最終段階) として，第一・第二段階の特徴を備えつつ中間節への筋を有する脊椎構造を設計・製作し，この脊椎を持つ全身腱駆動型ヒューマノイド “腱太”(Kenta) を設計・製作した [21, 60, 93, 106, 22] ( **Fig. 3.39** ).

### 3.7.1 腱太の脊椎の構造

**Fig. 3.40** に，腱太の脊椎の構造の概要を示す [60]．腱太の脊椎は，能動変形・柔軟性調節の自由度を最大に有し，途中節へ接続する筋を持つ多節構造であるため，多様な姿勢を実現することができる．腱太の脊椎は 2 種類 9 個の椎骨からなり，腰骨ブロック・肩骨ブロックとの間の関節も含めて合計 10 節の球面関節で構成される．椎骨のうち 3 個は肋骨が付いている．各節間には椎間板と靭帯に相当する構造がある．

#### 脊椎 S 字彎曲

人間の脊椎は 24 節からなり，横から見ると S 字に彎曲しており (生理彎曲)，長軸方向の力に対し構造材が破損しにくくなっている (3.4.1節参照)．腱太では，二種類の傾斜を持つ椎骨 ( **Fig. 3.40** 左端) を組み合わせて，脊椎 S 字彎曲を再現した ( **Fig. 3.40** 右端)．二種類の椎骨はそれぞれ傾斜は 10° で，後側 (背側) が低くなっているもの ( **Fig. 3.41** 左) と前側 (腹側) が低くなっているもの ( **Fig. 3.41** 右) がある．

#### 各椎骨と椎間板

各椎骨は上面に球状の突起，下面に球面状の凹みがあり，節間に球 (髄核に相当) が挟まる構造である．各球を囲むようにシリコンゴム[5)] を型で成形し，椎間板とした ( **Fig. 3.40** )．圧縮された状態で挟まり脊椎の柔軟性に貢献する．

球面関節部分の構造は Cla の脊椎 (3.6.1節参照) のように球が球面状の凹みの中に入り外れない形ではなく， Rabbit の脊椎 (3.5.1節参照) のように球面状の凹みは半球より小さくそのままでは崩れてしまう形である．腱太の脊椎では，椎骨の球面関節を構成する部分の中心に上下を貫通する穴をあけておき，腰から肩まで通った 1.2[mm] 径のステンレスワイヤが各椎骨を貫通する形にしてある．中心を貫通するワイヤは腰のブロックの構造材に固定され，肩のブロックで簡易テンショナーにより張力をかけられている．

[5)] 信越シリコーン KE-106, 硬さ (JIS-A)50, 伸び 120%

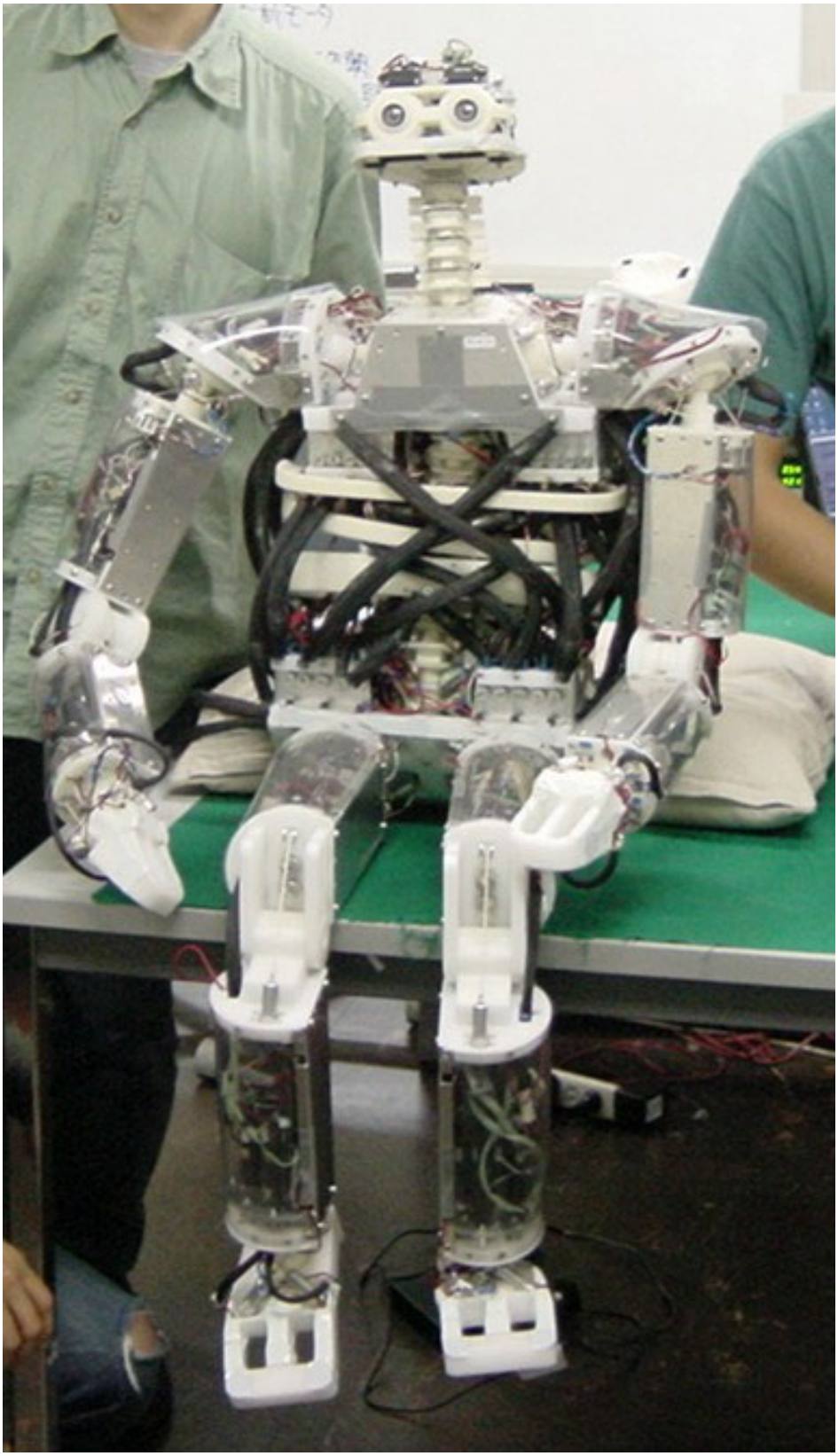


**Fig. 3.39** 脊椎を持つ全身腱駆動型ヒューマノイド腱太
Kenta, a whole-body tendon-driven humanoid with spine

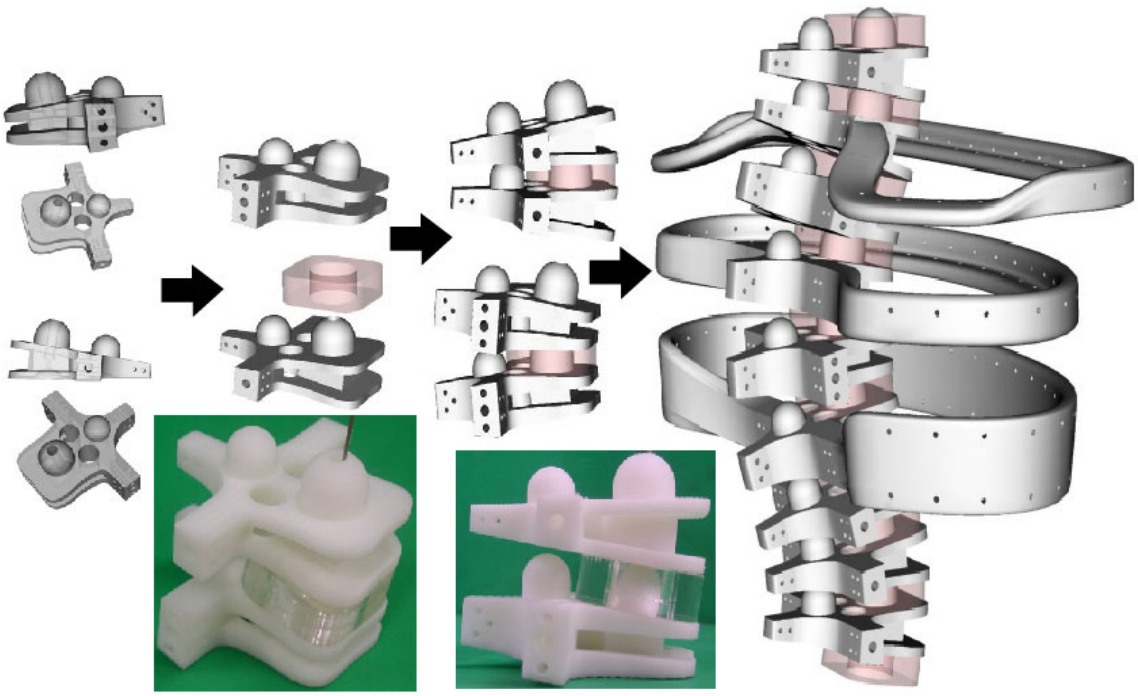


**Fig. 3.40** 腱太の脊椎の構造
The structure of Kenta's spine

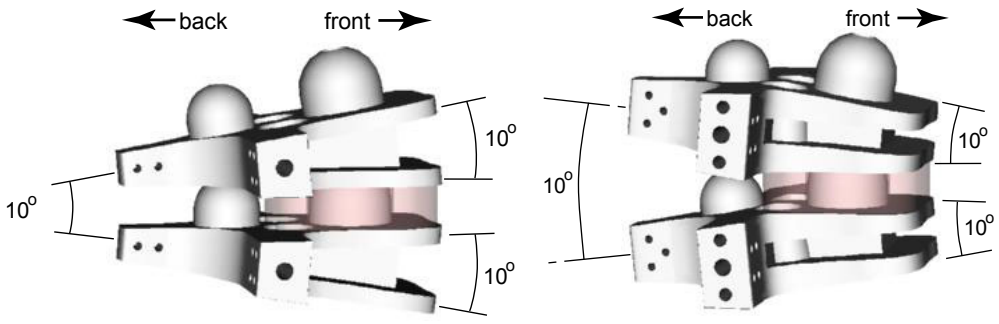



**Fig. 3.41** 二種類の傾斜の椎骨
Two kinds of inclination of vertebrae

### 椎骨後突起・横突起・靭帯

人間の椎骨には後と左右に突起があり，上下の突起間は靭帯で接続され，構造を保持する力と柔軟性と復元力を得ている．腱太の椎骨にも突起を設け ( **Fig. 3.40** )，靭帯の機能を期待して，隣り合う椎骨は，横突起同士・後突起同士をそれぞれ引きバネで接続してある．各節合計 6 本の引きバネの靭帯を有する構造である．各引バネの種類は部位によって異り，下部の節ほどばね定数が高いものを，前に凸な部分は自然長が短く後に凸な部分は自然長の長いものを選定した．合計 7 種類の自然長・ばね定数の引きバネを利用した．

各椎骨には，上面に球面関節を構成する突起の他にもう一つ突起があり，下面にも球面関節を構成する球面状の凹みの他にもう一つ球面状の凹みがある．腱太の脊椎は 10 節と多節であるため，一節に屈曲力が集中し材料力学における応力集中のような状態になり一つの節だけが大きく曲がってしまう可能性がある．こうした状態を防止するために，各節には過変形防止のハードウェア的なロック機構を設けた．これが上記の “もう一つの” 突起と凹みである．人間の脊椎も構造は異るが同様の機能を持つ機構を有する [35].

### 肋骨

肋骨は内蔵を守ると同時に，脊椎を駆動する筋の取り付け点という役割もあると考えられる．肋骨に筋を固定することにより脊椎の回転中心から筋を離し，筋の張力による脊椎の変形力を増していると考えられる．そこで，腱太の脊椎にも肋骨を持たせ，そこに筋を固定する構造にした．

人間の脊椎には 24 節の椎骨と片側 12 ずつの肋骨がある．腱太では左右の肋骨を一つにまとめて環状にし，肋骨の数は合計 3 とした．各肋骨には，筋を留めることができるように周囲に多数の穴をあけた ( **Fig. 3.40** 右端).

人間の脊椎では，肋骨と椎骨との間に関節構造があり，肋骨は椎骨と肋骨の接続点を中心として上下に動くことができる．腱太の脊椎における肋骨も椎骨との間に関節を設ける事も検討したが，本研究で着目している脊椎構造の腰から肩を結ぶ線の姿勢は肋骨と椎骨の間の関節の影響を受けないので，不要な自由度を可動部を増やすことになると考え，ここに関節は設けなかった．肋骨と椎骨の間に関節を設けないので，肋骨と椎骨は一体で設計・製作した ( **Fig. 3.42** ).

### 可動範囲

可動範囲は各節 roll,pitch,yaw の 3 軸とも $\pm 10^\circ$ 程度とした．ただし pitch 軸周りに関しては，後に凹な部分 (腰椎と胸椎上部，**Fig. 3.40** 参照) は過変形防止の小球により可動範囲は

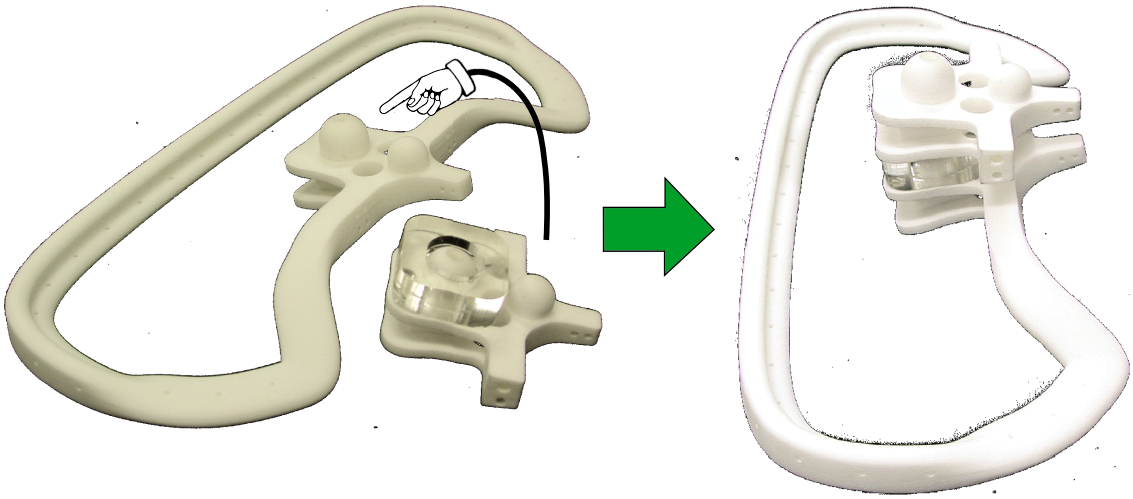


**Fig. 3.42** 椎骨と一体になった肋骨
The one-body rib-vertebra

$-3^\circ + 10^\circ$ 程度である．脊椎全体の可動範囲は， roll,pitch,yaw に各 $\pm 90^\circ$ 程度である．腱太の脊椎のリンク配置を **Fig. 3.43** に示す.

#### 肩と腰

脊椎を構成する椎骨は全部で 9 個あり，その両端は腰と肩のブロックになっている．隣り合う椎骨同士の間は合計 8 節の球面関節となり，両端の椎骨と腰・肩のそれぞれのブロックの間で上下計 2 節の球面関節が構成され，脊椎全体では 10 節となっている．腰と肩の部品は，脊椎との接続部分の形状を椎骨の上面 (肩は下面) の形状と同様に球状の突起 (肩は球面状のの凹み) を設けてある ( **Fig. 3.44** ).

### 3.7.2 腱太の脊椎の駆動系

#### 筋の駆動方法

柔軟性調節の実現のためには，コンプライアンス可変な筋が必要である (3.4.2節)．コンプライアンス可変な筋としては，空気圧人工筋 [75, 100, 39] (4.8.2節)，非線形バネ要素を持つ筋 [17]，張力センサを備える筋等が考えられる．空気圧人工筋と非線形バネ要素を持つ筋は，機械的柔軟性 (メカニカルコンプライアンス) が可変であるのに対し，張力センサを備える筋は，ソフトウェアによる力制御によりコンプライアンスを可変にする．空気圧人工筋は， (1) 空気圧源としてのコンプレッサや出力調節器としての電磁弁などが必要で装備が大掛かりになる， (2) ストロークが比較的短く脊椎全体の可動範囲を狭める原因になる， (3) 柔軟性 (剛性)

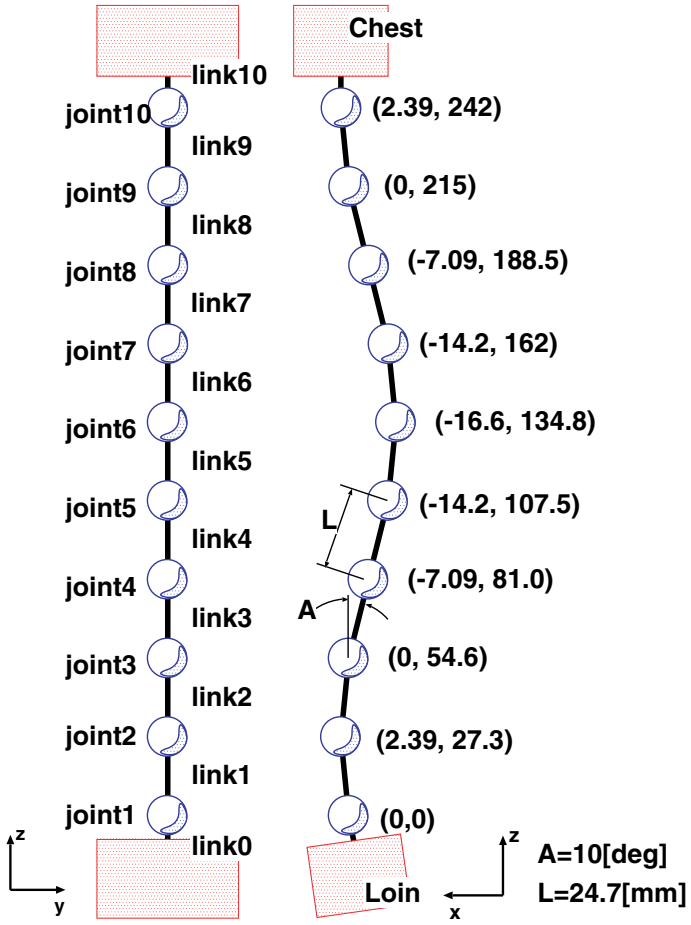



**Fig. 3.43** 腱太の脊椎の各リンクの位置
Positions of all links of Kenta's spine

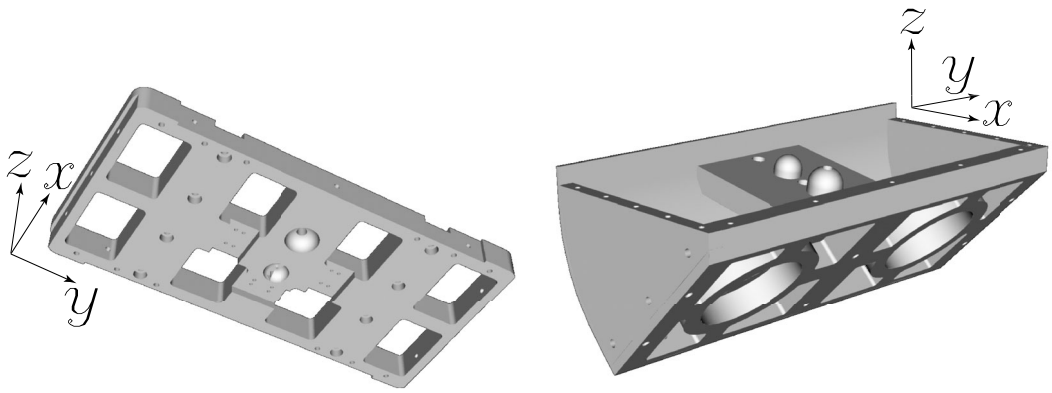



**Fig. 3.44** 左: 肩の部品 (下から見た図)， 右: 腰のブロック
left: a chest part (from below), right: hip block

と長さを同時に制御できない，ゆえに (4) 空気圧人工筋のみでは自然長を変えることができない，などの理由で採用しなかった．非線形バネ要素を筋に直列に組み込むことは，機械的柔軟性を自由に調節できる点は衝撃吸収と上体支持機構としての役割をあわせ持つ脊椎には機能的には適しているが，機構をコンパクトにするのが困難であり重量の増加にもつながるため，多数の筋を持つ腱太の脊椎のような構造の駆動に利用するのは現時点ではふさわしくないと判断した．そこで，コンプライアンスはソフトウェアによる調節になるが，衝撃吸収のようなソフトウェアでは対応しきれないような柔軟性の利用も，脊椎自体に機械的柔軟性を利用することができると考え，張力センサを備えたワイヤを DC モータとプーリを介して引くという形態を採った．

### モータの配置

腱太の脊椎は，各節 roll,pitch,yaw の 3 自由度 ×10 節 =30 自由度とすれば，最少で 31 本の筋が必要である． 3.4.1節で述べたように，復元力を持つ脊椎なので筋の本数が足りなくても姿勢は定まる．さて，腱太の筋の本数・配置はどのようにして決めればいいのだろうか．無数に存在する本数・配置の案の中から決定しなければならない．

まず，モータをどこに配置するかという観点から話を進めたい．筋が脊椎の変形力を発生する時，その両端で構造材に力を及ぼしているが，本論文ではその両端のうち，筋を構造材に固定する点を “取り付け点”，(構造材に固定された) モータ (プーリ) により引かれる側の点を “引っ張り点” と表現する．モータは筋の引っ張り点が属するリンク (を構成する構造材) に固定しなければならない．人間の筋肉は動力源を中に持ち，引っ張り点・取り付け点の区別をす

る必要が無い．空気圧人工筋なども引っ張り点・取り付け点の区別が必要無く，動力源の配置場所は筋の両端の二つのリンクのいずれかに固定する必要が無い．しかし，モータにより筋に張力を発生させる場合，取り付け点・引っ張り点の区別が必要で，引っ張り点のリンクにモータは固定されなければならない．腱太の脊椎のように途中節に固定された筋を持つ場合，モータにより筋に張力を発生させるなら，(1) 途中リンクに引っ張り点を設けそのリンクにモータを固定するか，(2) 途中リンクには取り付け点を設けモータはその筋の反対側の端 (引っ張り点) のリンクに固定するか，の選択肢がある．例えば，腰のブロックと肋骨の間に張る筋を駆動するモータは，(1) 腰ブロックまたは (2) 肋骨に固定しなければならないし，肋骨と椎骨横突起の間に張る筋を駆動するモータは，(1) 肋骨または (2) 椎骨横突起に固定しなければならない．

人間の脊椎を駆動する筋には非常に多くの種類の筋が存在するが，腱太の脊椎を駆動する筋はすべてを実現するのは現実的でないので，肋骨・椎骨間の筋，椎骨・椎骨間の筋は省略することにした．すなわち，筋の両端のいずれか一方の端または筋の両端は，腰のブロック (骨盤に相当) または肩のブロック (肩胛骨・鎖骨に相当) に固定することにした．さらに別の表現をすれば，筋は全て腰・肩のブロックのいずれか若しくは両方に引っ張り点又は取り付け点を持つという事である．肋骨または椎骨 (**A** とする) と腰のブロックまたは肩のブロック (**B** とする) の間に筋を張ることになるので，モータは **A** 若しくは **B** のいずれかに配置することになる．腰・肩のブロック (**B**) の方が肋骨・椎骨 (**A**) より空間的余裕があるので，モータは腰・肩のブロックに配置することにした．

### 筋の本数・配置・接続

筋の本数は多ければ多いほど，実現できる姿勢の多様性が増す．肩と腰にモータを全て配置する事にしたので，肩と腰のブロックにできるだけ多くのモータを配置することとする．肩・腰共にモータを前後方向に並べ，腹筋を引くモータは前面側にプーリが来る向きに，背筋を引くモータは背面側にプーリが来る向きに配置することにした．肩・腰ともに，左右両側にこのモータの配列を設け，前向きのモータを左右各 5 本後ろ向きのモータを左右各 5 本配置した．肩・腰ともに，前後・左右それぞれ 5 本ずつで計 20 本である．肩と腰をあわせて，合計 40 本のモータを配置することにした． **Fig. 3.45** は腰と肩を含む脊椎構造を斜め後方から見た図である，腰と肩のブロックに背骨の左右両側にモータを 5 本ずつ 2 段に並べた計 10 本のモータアレイが設けられている (モータは緑色で示されている)．

合計 40 本の筋の配置・接続は，人間の脊椎を駆動する筋肉のエッセンスを取り込んで決定した．人間の場合，背筋側は背骨を斜めに横切る筋はほとんど無く，椎骨横突起にほとんどの筋は接続されている．その内訳は脊椎に沿った筋，椎骨と肩胛骨を結ぶ筋，椎骨と骨盤

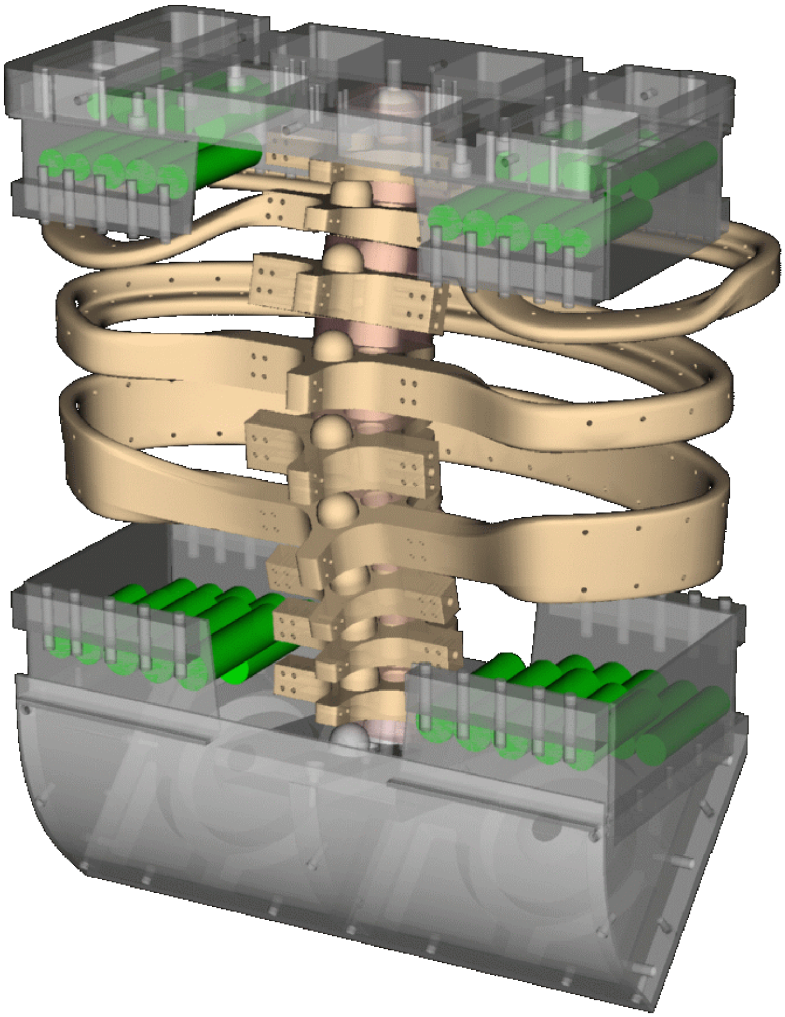


**Fig. 3.45** 腱太の脊椎の筋駆動用モータの配置 (背面から見た図)
The arrangement of the motors to pull the muscles for Kenta's spine

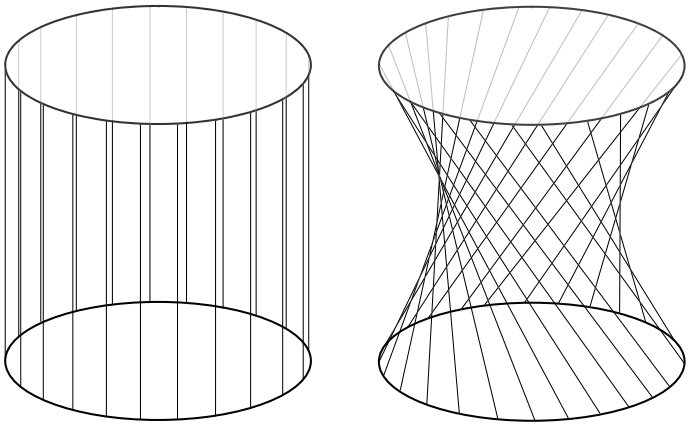


**Fig. 3.46** 腹筋とウェストのくびれ (左: 直行した腹筋，右: クロスした腹筋)
A model of abdominal muscles (left:straight, right:cross)

を結ぶ筋，椎骨と肋骨を結ぶ筋などがある．腱太の脊椎の背中側の筋も脊椎を横切る筋は設けず，肩・腰のブロックから椎骨横突起へ接続する筋と，肩ブロックと腰ブロックを直接接続する筋を設けた．背筋が脊椎を横切らないのに対し，人間の腹側の筋肉は腹を斜めに横切る筋が多い．特に腹筋の部分は筋が斜めにクロスするように張られており，それによりウェスト部分がくびれるようになっている ( **Fig. 3.46** ) ．これにより旋回 (yaw 軸回りの動き) も実現できる．腱太の脊椎の前面の筋配置はこれを参考に，全ての筋が斜めに体幹を横切る形に配置した．腱太の脊椎を駆動する合計 40 本の筋の配置・接続を以下に述べる ( **Fig. 3.47** ， **Fig. 3.48** ).

- 腰骨後端から出る筋肉 (左右各 5 本計 10 本)
  偶数番目の椎骨の横突起 (または肋骨付け根) に接続する．内側 (背骨に近い側) のモータほど下の方の椎骨に接続する．つまり，
  - 1 番外側の筋肉は肩骨に,
  - 2 番目は上から 2 番目の椎骨横突起 (または第一肋骨付け根) に,
  - 3 番目は 4 番目の椎骨横突起 (または第二肋骨付け根) に,
  - 4 番目は 6 番目の椎骨横突起 (または第三肋骨付け根) に,
  - 5 番目 (最も内側) の筋肉は 8 番目 (下から 2 番目) の椎骨横突起に,

  それぞれ接続する.

- 腰骨前端から出る筋肉 (計 10 本)

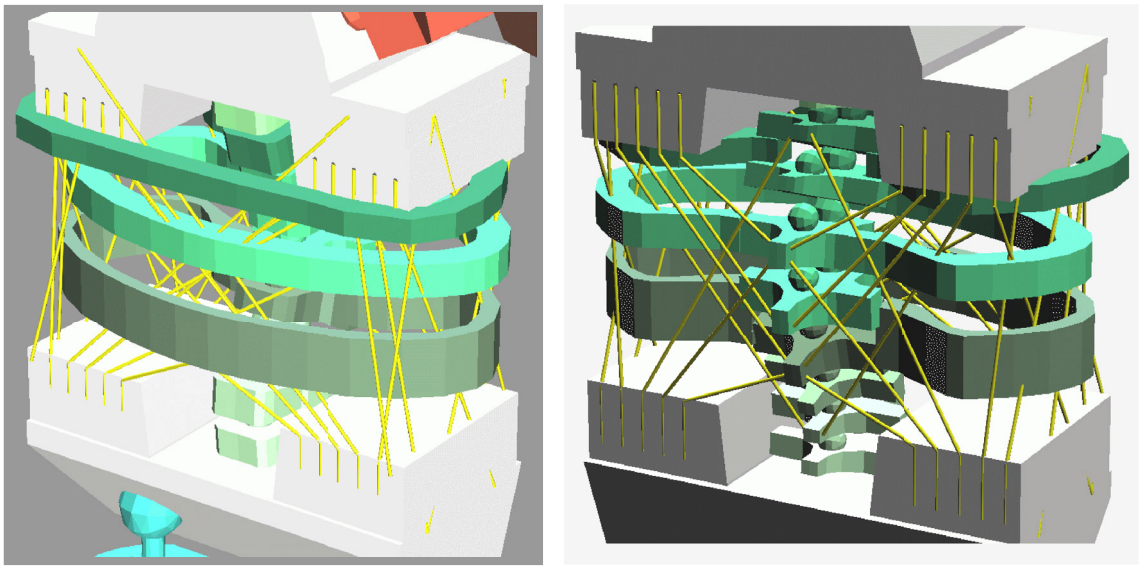


**Fig. 3.47** 腱太の脊椎の筋配置 (左: 前面, 右: 背面)
The arrangement of muscles of Kenta's spine (left:front, right:back)

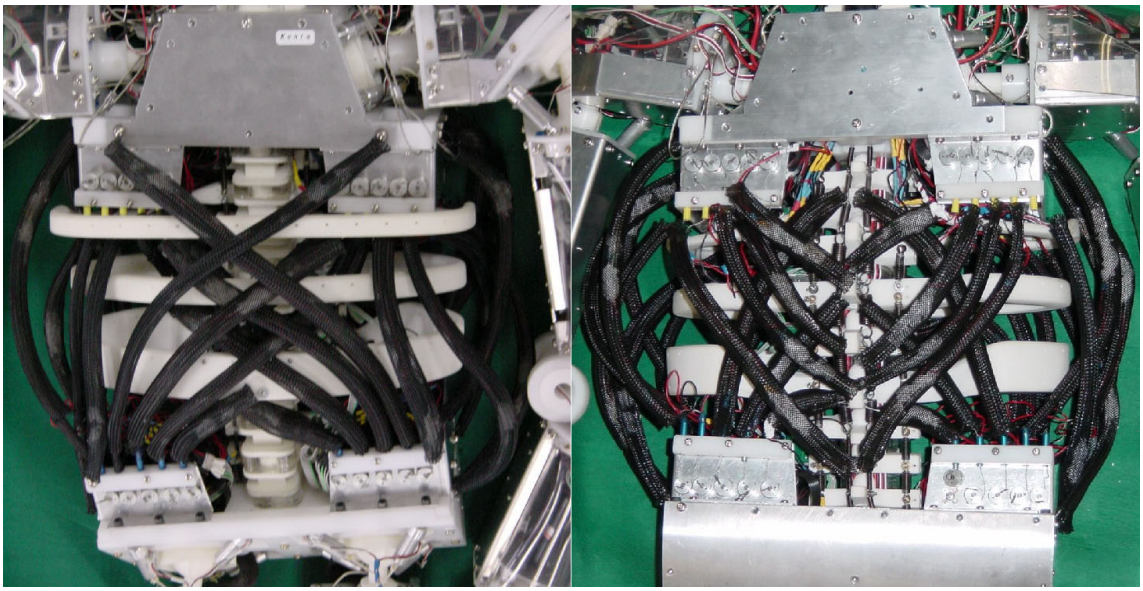


**Fig. 3.48** 腱太の脊椎の筋配置 (左: 前面, 右: 背面)
The arrangement of muscles of Kenta's spine (left:front, right:back)

内側 3 本は各肋骨に接続する (腰内側の筋ほど下部の肋骨に接続)．外から 2 番目は肩骨前端に接続する． 1 番外側の筋肉を肩骨の後側外端に接続する (体の横を斜めに通る)．

- 肩骨後端から出る筋肉 (計 10 本)
  偶数番目の椎骨の横突起 (または肋骨付け根) に接続．肩内側 (背骨に近い側) のモータに接続された筋ほど上の方の椎骨に接続．左側から出る筋肉は右側の横突起に接続 (体の後を斜めに通る)．

- 肩骨前端から出る筋肉 (計 10 本)
  内側三つは各肋骨に接続．外から二番目は腰骨前端に接続．それぞれ左側から出る筋肉は右側に接続 (体の前を斜めに横切る)．一番外側の筋肉を腰骨の後側外端に接続 (体の横を斜めに通る)．

この配置は，長い筋肉ほど回転中心から遠いところを通る．これにより，全体としての動きに関わるほど (姿勢変形力／筋張力) の値が大きくなる．

#### モータの選定

モータは引く筋の場所・役割に応じていくつかの種類のモータを用意した．腰・肩のブロックの内側 (脊椎に近い側) の筋と外側の筋を比較すると，内側の筋の方が脊椎の回転中心に近い場所を通っており強い張力が要求され，外側の筋の方が回転中心から離れているため張力より速度が必要とされると思われる．また，脊椎は肩のブロックを含む上半身を支持しなければならないので，肩のブロックはできるだけ重量が増さないようにする方が良い．そこで， Table 3.3 に示すように，背筋・腹筋の外側 2 つ，背筋下部中央 3 つ，腹筋中央 3 つと上部腹筋・背筋中央 3 つで，合計 3 種類のモータを選定した．

### 3.7.3 センサとその処理系

モータ制御とセンサ情報収集・処理を行う 28 個のプロセサをボディに分散配置し， $I^2C$ バスによる体内 LAN で結んだ．体内 LAN のハブがシリアル通信で PC と通信している．脊椎機構に搭載したセンサを以下に示す．

**張力センサ** ウレタン・力覚センサを二枚の並行平板間に挟み，張力に応じて平板の間隔を縮める方向にかかる力を計測する構造のセンサ [58]

**触覚センサ** 腱太の全身の各リンクに 6 点程度， FSR(Force Sensing Resister) という力覚センサを，合計 60 点取り付けた．環境との接触状態を検知できる他，人間がセンサを直接触

Table 3.3　腱太の脊椎駆動用モータの選定
The selection of motors for Kenta's spine

| 配置場所 | 背筋・腹筋の外側 2 個ずつ (腰・肩とも) (合計 16 個) | 腰部から出る背筋内側 3 個 (合計 6 個) | 腹筋内側 3 個ずつ (腰・肩とも) (計 12 個) および肩部から出る背筋内側 3 個ずつ (計 6 個) | 単位 |
|---|---|---|---|---|
| 最大筋張力 | 65.2 (6.65) | 532 (54.3) | 240 (24.4) | N (kgf) |
| 最大引張り速度 | 337 | 52.3 | 82.8 | mm/s |
| 張力 5kgf 時の引張り速度 | 83.5 | 47.5 | 65.9 | mm/s |
| モータ型式 ※1) | RE$\phi$13mm, PB, 2.5W | RE$\phi$16mm, GB, 4.5W | RE$\phi$13mm, PB, 2.5W | |
| 外径寸法 | $\phi$13×59.5 | $\phi$16×72.5 | $\phi$13× | mm |
| 定格出力 | 2.5 | 4.5 | 2.5 | W |
| 公称電圧 | 24.0 | 24.0 | 24.0 | V |
| 無負荷回転数 | 17088 | 14000 | 17800 | rpm |
| 停動トルク | 14.2 | 34.7 | 14.2 | mNm |
| ギヤ比 | 16.58 | 84 | 67.49 | |
| 無負荷回転数 — ギヤ後 — | 1073.6 | 166.7 | 264 | rpm |
| 停動トルク — ギヤ後 — | 0.195 (1.99) | 2.13 (21.7) | 0.958 (9.78) | Nm (kgf·cm) |
| エンコーダ分解能 | 16 | 16 | 16 | cpr※2) |
| プーリ内径 | 6 | 6 | 6 | mm |

※ 1) PB: 貴金属ブラシ， GB: グラファイトブラシ

※ 2) cpr: カウント / 回転

ることによって，直接教示状態へのトリガの役割をさせることもできる．また，アナログ値を取れるセンサなので，触り方の強弱によって異る反応をさせることもできる．

**3 軸加速度センサ** 椎骨 9 個のうち 5 個に，3 軸加速度センサを取りつけた．重力加速度の方向を各節で計測することで，脊椎全体の状態 (姿勢) を推定する．

**電流センサ** モータ電流を計測し過負荷を検知する．

**エンコーダ** 受動的変形時にも筋長を記録できる

柔軟性を持つ構造を含むロボットには，構造の破壊を防ぐためにも過負荷対策が不可欠である．緊急事態を検知するセンサと共に，即応できる反射系が必要とされる．各分散プロセッサは局所的にセンサ情報 (張力・電流) を監視し，過負荷時にモータ出力を停止できる．

### 3.7.4 脊椎を持つ全身腱駆動型ヒューマノイド腱太の全身設計

腱太の設計コンセプトは，大きく二つあった．一つは脊椎を持つ全身型ヒューマノイドであるということ，もう一つは全ての関節が腱駆動[6)]であるということである．脊椎構造を筋により駆動する機構も，腱駆動の範疇にはいる．腱太の手足の関節は肘・膝関節は 1 自由度，それ以外の肩・手首・股・足首の各関節は全て 3 自由度を持つ球面関節により構成した [93, 94]．以下に脚部，腕部，頭部の設計の概要を述べる．

#### 脚部

腱太の脚部の自由度配置とを **Fig. 3.49** に示す．股関節 3 自由度，膝関節 1 自由度，足首関節 3 自由度の 6 自由度構成になっている．股関節と足首関節は球面関節機構を用い，可動範囲はロール，ピッチ角 55°，ヨー角 45° となっている．

それぞれの球面関節は 4 本の筋 (直径 0.8[mm] のケブラーワイヤ) で駆動されている．それぞれの筋配置は，**Fig. 3.50** に示すように，2 本ずつ交差させ，その組を相対して配置した．このように筋を交差させて配置することで，ヨー方向の稼働範囲を大きくしている．また，筋の中継には小径のプーリを用い，伝達路での摩擦による筋の切断を防いでいる．

股関節の付け根部分は水平から 45° 傾けている．これは人間の股関節の稼働範囲が後側へ小さく (10° 程度)，前側に大きい (90° 程度) であることを再現するためのものである．

膝の 1 自由度は，2 本の筋によって駆動されている．1 自由度であればプーリを用いる方法が一般的であるが，ここでは機構がより簡便になるすべり軸受を用いている．膝関節は後

[6)] 本論文では「腱」という言葉を用いず「筋」と記述している．

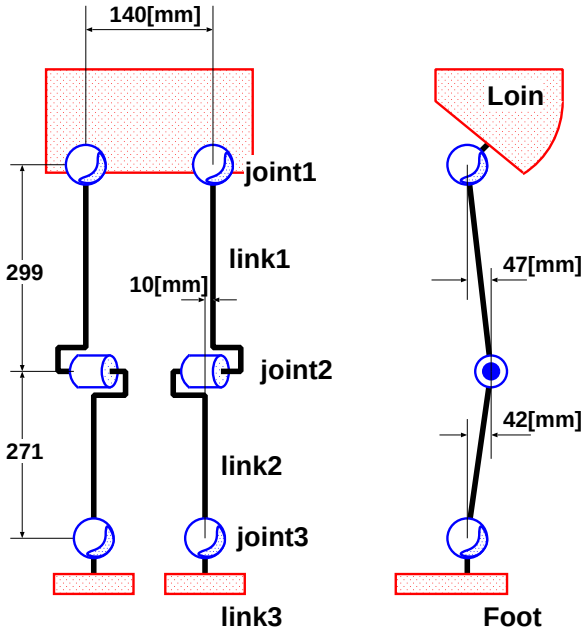



**Fig. 3.49** 腱太の脚の機構
The mechanical design of Kenta's leg

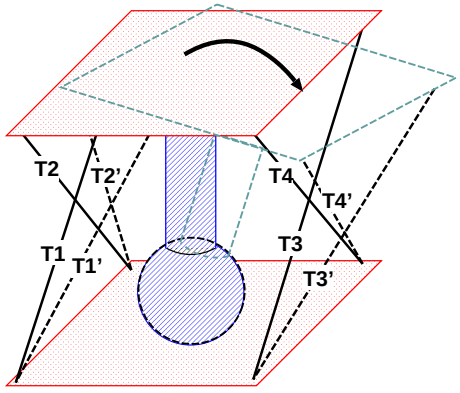



**Fig. 3.50** 脚・球面関節の筋配置
Assignmnent of muscles for spherical joint

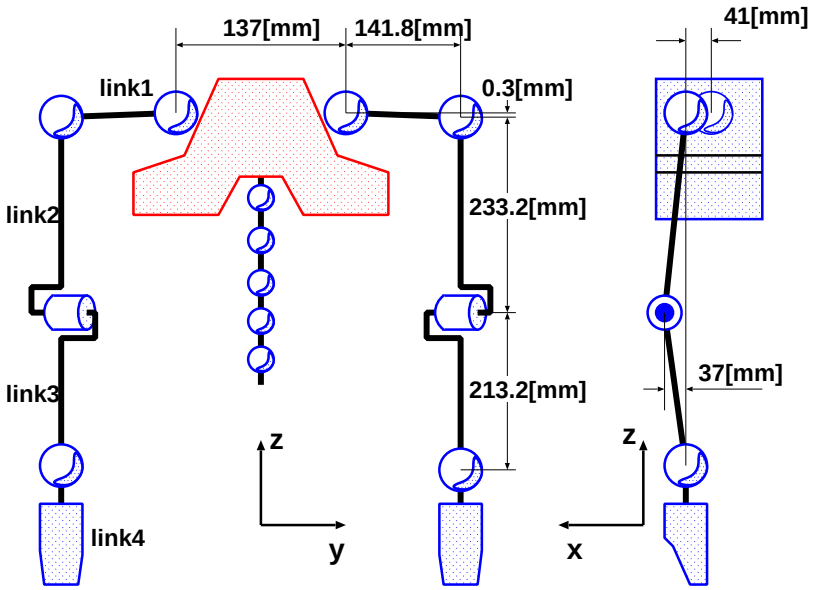



**Fig. 3.51** 腱太の腕の機構
The mechanical design of Kenta's arm

部に軸半径分だけオフセットされているが，これは膝の可動範囲を 180° まで広げるためである．

各筋には，後述するバネ要素を持った張力センサが筋末端側に取りつけられている．筋にバネ要素を付加することで，筋への過大な張力が発生することを防ぐことができる．また，筋へが非線形バネ要素を持っていることで，剛性の調節が行なえることがわかっている．

用いているモータは減速比 64:1 遊星ギヤヘッド，エンコーダ付きの 4.5[W]DC モータである．筋巻きとりに用いているプーリは $\phi$8[mm] のものであり，これにより，停動トルク発生時の筋張力は 73[kgf] となる．モータは上腿部に 6 本 (股関節駆動用 4 本，膝関節駆動用 2 本)，下腿部に 4 本 (足首関節駆動用) を配置している．

### 腕部

腱太腕部の自由度配置を **Fig. 3.51** に示す．人間の肩関節は複雑なリンク機構により，擬似的に 5 自由度を実現していると言われている．そのため， 3 自由度の球面関節 1 つでは人間の肩関節と同様の可動範囲を実現することはできない．そこで胸部中央近くと，上腕付け根部分に一つずつの球面関節を配置し， 6 自由度構成とすることにした．筋の配置については，脚部と同様である．

用いているモータは，手首部は減速比 64:1 遊星ギヤヘッド，エンコーダ付きの出力 3[W]DC

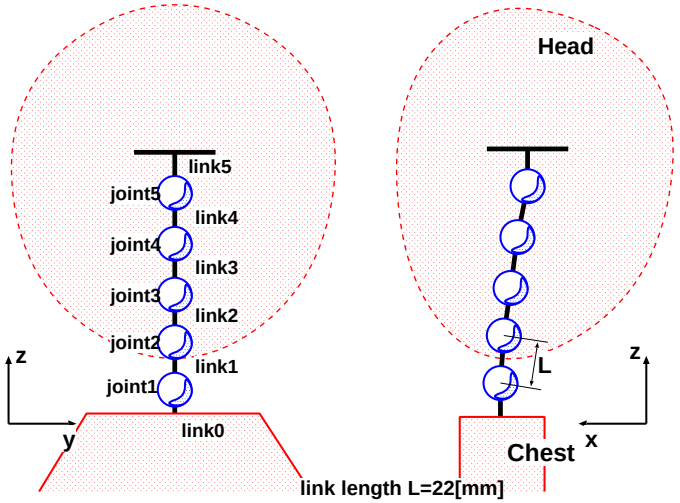



**Fig. 3.52** 腱太の頭部の機構
The mechanical design of Kenta's head

モータである．筋巻きとりに用いているプーリは $\phi$6[mm] のものであり，これにより，停動トルク発生時の筋張力は 25[kgf] となる．また肩部，肘部のモータは減速比 256:1 遊星ギヤヘッド，エンコーダ付きの出力 3[W]DC モータである．モータは肩リンク部分に 4 本 (胸部関節駆動用)，上腕部に 6 本 (肩関節駆動用 4 本，肘関節駆動用 2 本)，前腕部に 4 本 (手首関節駆動用) 配置されている．

### 頭部

腱太の頭部の自由度配置を **Fig. 3.52** に示す．頚椎の構成は脊柱を縮小したものである．頭部に配置した 6 本のモータでステンレスワイヤ ($\phi$0.8[mm]) を牽引して駆動している．脊柱とは異なり，アルミ板による筋のガイドがそれぞれの頚骨に固定されており，筋はそのガイドの穴を滑ることになる．この機構はリンクの自由度が 15(球面関節 5 個) あるにもかかわらず駆動筋は 6 本であるため，それぞれの関節角度を個別に決定することはできない．しかし，リンク同士は非線形なバネ要素 (シリコンゴムによるスペーサ) で結合されているため，全体としての姿勢を限られた範囲で決定することは可能である．

また眼球部分は球面関節の機構で，球面関節内に小型カラー CCD カメラが組みこまれている．眼球部は 2 つの眼球を継いだ閉リンク機構を 2 つのラジコン用サーボモジュールにより駆動し，パン，チルト角度の制御が可能である．カメラからの画像はホストコンピュータの画像キャプチャカードに入力することで，両眼立体視，色抽出などの処理が可能である．

### 3.7.5 腱太の部品の形状設計と製作

椎骨や肋骨は人間のそれを再現する必要はないが， 3.7.1節に述べたような要件を満たす形状は複雑な形状になる．腱太の脊椎の部品の形状設計・製作においては， Rabbit， Cla の脊椎部品の設計と同様の手法，すなわち三次元 CAD を用いた形状設計と NC 工作機械を用いたエンジニアリングプラスチックの削り出しという手法 (3.5.1節， 3.6.1節参照) では，後者の工程が非常に複雑になる．特に，軽量化のため各部品の強度に余裕のある部分にえぐりを入れる等の加工や，面の角度精度などを出すためには，チャック法の検討・治具の作成など，加工プロセスと密接に関わりを持った形状設計が求められる．また，このような方法で形状設計・製作を行うと形状変更→試作というループを繰り返してゆくような開発手法が取りづらくなる．

腱太の脊椎の構成部品の製作においては，前段落に述べたような従来型の加工プロセスではない加工法の模索も行った．まず，椎骨や肋骨などの複雑な三次元形状の設計においては，三次元 CAD の利用が有効である．腱太の設計においては，全ての部品の形状設計において三次元 CAD[7)]を利用した． Rabbit， Cla で経験のある椎間板の作成は， Rabbit， Cla と同様の手法により NC 工作機械でエンジニアリングプラスチックから削り出しにより作成した型を用いて，シリコンゴムを成形することで製作した．肩・腰のブロックを構成する部品や，脚部・腕部の部品は，電子データ[8)]のやりとりが可能な加工業者に依頼し， NC 工作機械や従来型の工作機械を用いた加工法により製作した．椎骨・肋骨・眼球などの複雑な形状は，従来型の加工プロセスでは加工が困難あることがわかり，最新のプロトタイピング技術を利用した．光造形による成形より短時間・安価な成型法である FDM(Fused Deposition Modeling) 成形法， SLS(Selective Laser Sintering) 成形法により製作した． FDM 法は，椎骨の試作に利用した．最終的に腱太の脊椎に利用した椎骨・肋骨・頸椎・頭部・眼球の部品は， SLS 法により製作した．

### 3.7.6 腱太の脊椎の動き

腱太の脊椎を利用した動作の様子を **Fig. 3.53** に示す．詳しい制御法は第 4章に述べる．全身の動きを利用した人間の動きに近づいた動きを可能にしている様子がわかる．

---

[7)] 脊椎・首・頭部の部品は主に Rhinoceros，肩や腰のブロック及び脚部・腕部は主に Autodesk Mechanical Desktop を利用した．

[8)] 広く用いられている汎用なデータ型式である IGES 型式によった．

**Fig. 3.53** 腱太の脊椎の動き
A motion of Kenta's spine

## 3.8 脊椎構造の構成まとめ

- **冗長性・多様性を実現する多節構造**
  連続した球面関節からなる多節構造にすることで，変曲点を持つような姿勢を実現.
- **多く筋による拮抗駆動**
  多様な姿勢の実現．筋はコンプライアンス (機械的もしくはソフトウェア) が可変.
- **筋の経路**
  - 多節構造の中間節に取り付ける筋.
  - 中心軸から離すことでモーメントアームを大きくし，姿勢変形力向上.
- **基本姿勢を持ち，復元力を持つ構造**
  各節間に弾性要素の椎間板を挟むことにより構成．基本姿勢に戻ろうとする復元力を構造が有することにより，重力に逆らう運動時にアクチュエータの必要力を低減.

# 第 4 章

# 拮抗筋駆動による柔軟性を有する脊椎の姿勢制御

本章では脊椎構造の制御に関し述べる．問題の本質は，(1) 制御量は何か (何を制御するか)，(2) 制御法は何か (いかに動かすか) という点であると考え，従来のロボットアームや脚の制御法と比較して，共通する点と異る点を整理する．そして，脊椎構造を持つ多自由度ロボットの制御の基本的立場として，(1) 従来型の全身型ロボットと透過的に扱える制御，(2) タスク依存のセンサフィードバックによる制御，の二点が重要であるという立場を主張する．さらにこうした立場から，(1) ある程度の幾何学的制御量の制御を行う方法の提案，(2) センサフィードバックによるタスクの実現へのトライアルを行い，主張の妥当性を示した．ロボットをどう動かすかという観点を階層化して考えると，姿勢 — 動作 — 行動というレベルが考えられる．第 4章 (本章) では，姿勢制御に関し述べ，第 5章で動作，および行動に関し述べる．

## 4.1 問題の整理

### 4.1.1 姿勢 — 動作 — 行動

ロボットをどう動かすかという観点を階層化して考えると，姿勢 — 動作 — 行動というレベルが考えられる．

**姿勢 (posture)** 各関節の変位により決まる．運動学・静力学の計算．

**動作 (motion)** 姿勢の時系列．その実現方法は関節変位の軌道に基づくものと，センサフィードバックによるものがある．

**行動 (action, behavior)** 一連の動作の組合わせ．目的 (タスク) を持った動作．状況判断とそれに応じた動作の選択・決定のプロセスを含む．

本章では脊椎構造を持つロボットの姿勢制御に関し述べ，第 5章で脊椎構造を持つロボットの動作および行動に関し述べる．

### 4.1.2 制御量と制御法

多節の脊椎構造の各節の位置・姿勢といった幾何学的な制御量を正確に制御する事は，それが可能ならばできるに越したことはないが，それが困難である場合が多く，それらの制御量を正確に制御する事以外の制御が重要になる．それは，結局はタスク依存になるはずだと考えられる．例えば，物を移動するというタスクであれば対象物を把持できることが求められるし，物体を追跡するというタスクであれば対象物が視野の中心に見えるように制御できることが求められる．幾何学的制御量をいかに制御してタスクを実現するかではなく，別の方法でいかにタスクを実現するかが問題であると考えられる．

これまでのロボットは，三次元ユークリッド空間の位置情報を主に利用して物体を操作していた．しかし，人間はそうではない．自身や環境との相対的位置，それもユークリッド空間で表現される位置よりはむしろ (視覚・平衡感覚などの) 内界センサのセンサ特性に基づく操作を行っていると考えられる． 6 自由度や 7 自由度程度の自由度で，関節軸がはっきりしたアームなどを利用する運動とは異る．対象物を把持するためには，物体の位置情報を視覚系から三次元直交座標系に変換しハンドをユークリッド空間で表現される位置・姿勢に持ってゆく制御をする方法以外に，大まかな幾何学計算に基づく軌道計画とセンサフィードバックによる調整により物体を把持する，ということを人間は行っている．それは，自由度が大きくはっきりした関節軸を持たないような身体構造であるからこその方法であると考えられる．脊椎構造のような自由度の多い身体構造のロボットにおいては，従来のロボットの方法論よりもむしろ人間のやり方に近い方法論が必要となるのではないだろうか．本研究では，こうした視点に立ち，ユークリッド空間に基づく軌道は大まかな実現ができればよく，センサフィードバックにより差を埋めるというアプローチを取るべきであるという立場をとる． ( Table 4.1 )

Table 4.1　従来型ロボットと人間の制御方針の比較
The comparison of control between current robot and human

| 比較項目 | 従来のロボット | 人間 |
|---|---|---|
| 自由度 | 6 自由度や 7 自由度程度 | 多自由度 |
| 関節軸 | 明確な関節軸 | 関節軸は明確でない (面接触) |
| 制御量 | 幾何学的な制御量 (x,y,z,roll,pitch,yaw) | センサベーストな制御量 (内界センサ) |
| 制御のゴール | 正確に制御 | タスク依存 (例えば物体追跡なら視野の中心に見えるように制御) |
| 実際の制御 | 正確な幾何学・動力学計算とセンサフィードバックによる外乱の吸収 | 大まかな幾何学・動力学計算とセンサベーストな制御 (センサフィードバックの比率が高い) |
| モデル可変性 | 幾何学的モデル (運動学・静力学・動力学. (各順逆)). 基本的に不変. | センサベーストモデル．自己モデル形成．モデルを常に更新し続ける. |

また，モデルの可変性という観点から考えると，従来のロボットは基本的にモデルは不変であった．しかし，人間に近いロボットの場合，特性の径時変化や，摩擦などの不確定要素も大きい．また，多センサ・多アクチュエータが全て正常に動いている状態を保つのは難しくな

り，いくつか故障していても全体としては動き続けることができることも必要になる．したがって，モデルがセンサ空間を基準にするというだけでなく，日々，時々刻々，モデルが更新されてゆくような仕組みが必要になってくると考えられる ( Table 4.1 ).

本章では， x,y,z,roll,pitch,yaw といった従来のロボットの制御量をある程度制御できる手法を構築すると同時に，センサフィードバックによるタスクの実現というアプローチにも取り組んでゆく．

### 4.1.3 直接教示

4.1.2節とは別の視点から見ると，複雑な身体構造を有するロボットを動かす方法の一つとして，直接教示と倣い作業 [88, 65] を考慮に入れる必要がある．人間も複雑な姿勢あるいは，複雑な姿勢を含む動作などをできるようなる時，直接教示・関節教示を受ける効果は大きい．また以前のロボットは，教示 (直接でなくても)・再生という方法は有効であった [2]．直接教示のためのアクチュエータの制御法に関しても本章で述べる．

直接教示には，次のような特徴がある．

**モデル不要** 人間が直接ロボットに取らせたい姿勢を指示するため，ロボットの関節角度・筋の長さ等と，目的とする姿勢の間の関係を記述するようなモデルが必要ない．

**筋の干渉問題が無い** 第 4.3節に述べるように，幾何学的な脊椎のモデルを構築する際に，筋同士の物理的な干渉や筋とボディ (構造材) の干渉は問題である．直接教示により姿勢を指示する場合は，実現可能な筋の長さの組み合わせ・関節角度の組み合わせであることが保証できるので，筋の干渉問題を考慮する必要が無い．

**教示していない姿勢は再生不可能** 直接教示のみにより姿勢・動作の作成を行うと，教示していない姿勢・動作を行うことは困難である．

**計算機上での姿勢・動作生成が困難** 前項と関わるが，脊椎の姿勢を人間が直接教示した場合，その時の脊椎の姿勢はモデル無しでは角度や位置などの幾何学的な数値情報を得ることができない．そのため動作軌道の生成などを計算機上のみで行うことができない．実機が実際にとった姿勢の組み合わせでの軌道生成が困難である．

### 4.1.4 脊椎構造を持つ多自由度全身型ロボットのための制御法

本研究ではで 4.1.2節述べた考え方に基づき，脊椎構造を持つ多自由度全身型ロボットの制御法として，幾何学ベースの脊椎構造の制御法の提案と，センサベースの脊椎構造の制御法の

トライアルを行った．さらに，計算機上で直感的に把握しづらい複雑な姿勢等の実現に利用可能な，直接教示・再生の枠組みを実現した．

- 直接教示・再生
  筋張力を一定に保つ機能と，筋長を目標長さに保つ機能を用意した．さらに，いくつかの姿勢を直接教示し，その再生により姿勢の制御ができることを確認する実験を行った．

- 幾何学ベースの制御法
  脊椎構造の姿勢を各節の roll,pitch,yaw 角で表現し，その姿勢を実現するための各筋の制御法を提案する．さらに，提案した制御法に基づき姿勢制御実験を行った．

- センサベースの制御法の試行
  例として視覚情報に基づくセンサベーストな一制御法を提案する．さらに提案した制御法による視覚情報の目標量を制御する実験を行った．

## 4.2 直接教示・再生による姿勢制御

姿勢を制御するための一つ方法として考えられるのは，ロボットを人間が物理的に動かして，姿勢を教えることがある．つまり直接教示 (direct teaching) である．人間でも複雑な姿勢や，複雑な姿勢を含む動作などをできるようになる時，教示を受ける効果は大きい．また制御法が洗練されていなかった初期のロボットは，教示 (直接でなくても)・再生という方法は有効であった．脊椎を持つロボットの姿勢制御の一つとして，直接教示・倣い作業というのは有効な手法となる．

### 4.2.1 姿勢の教示

人間が直接物理的に脊椎を動かして姿勢を教示する際，必要となるのは筋をたるまずきつ過ぎずの状態に保つことである．これには筋を張力制御できれば良い．例えば，全ての筋を 1[kgf] の張力になるように制御しておけば，人間が外から加えた力によって張力が大きくなる筋は長さを長くすることで張力を 1[kgf] に保ち，逆に張力が小さくなりたるみそうになる筋は長さを短くすることで張力を 1[kgf] に保つ．そして，ロボットが目的の姿勢になったところで，各筋の長さを記録する．その姿勢を再現するには，各筋の長さを記録した長さになるように制御すれば良い．

姿勢の直接教示・再現を実現するために必要な機能は，

- 各筋の張力制御機能
- 各筋の長さ計測機能
- 各筋の長さ制御機能

である．張力制御を行うためには筋張力センサが必要となり，長さ計測及び長さ制御を行うには筋長センサが必要となる．本研究で製作した人間型ロボット Cla(第 3.6節) および腱太 (第 3.7節) においては，脊椎を駆動する全ての筋に張力センサを組み込み，筋長はモータに取り付けられたエンコーダによりプーリの回転角を計測することで筋長センサの代用とした．これらのロボットを用いて，筋張力一定制御と筋長記録による教示と，その時に記録した長さに筋長制御することにより姿勢を再現する実験を行った．

### 4.2.2 動作の教示

動作は複数の姿勢を連続的に再現することにより実現可能である (4.1.1節)．以下に示す 2 種類の方法により動作の教示と再生の実験を行った．

#### 複数の姿勢の教示による動作の実現

実現したい動作の途中姿勢のいくつかを 4.2.1節の方法により教示・記録し，いくつかの姿勢を途中を補間しながら再生することにより動作を実現する．補間の方法に関しては，いくつかの補間法を実装した (6.3.4節参照)．

腱太 (第 3.7節参照) に三種類の姿勢を教示し，それを順に繰り返し再生する様子を **Fig. 4.1** に示す．

#### 動作の直接教示

直接人間が触って動かしたその動きを全て記憶し再生する．すなわち，直接動かされている時に，あるサンプリング周波数で各筋の長さをすべて記録してゆき，それを順に再生する．補間の必要がないため，確実に中間姿勢も再現されるやすいが，長時間の動作だと記録するデータが膨大になる可能性がある．また，記録したデータを忠実に再生することしかできず，記録したデータの再利用は難しい．この問題の解決に，本研究では姿勢データベースという手法を提案した (第 5.9節参照)．

Cla および腱太を用いて，全筋を一定張力になるように制御した状態で，人間が触って動作を教示し，その時のエンコーダのログを記録した．再生時には，ログのデータを順にエンコーダの目標値として送り，動作を再生した． Cla の実験において，再生時にログから 1 つ

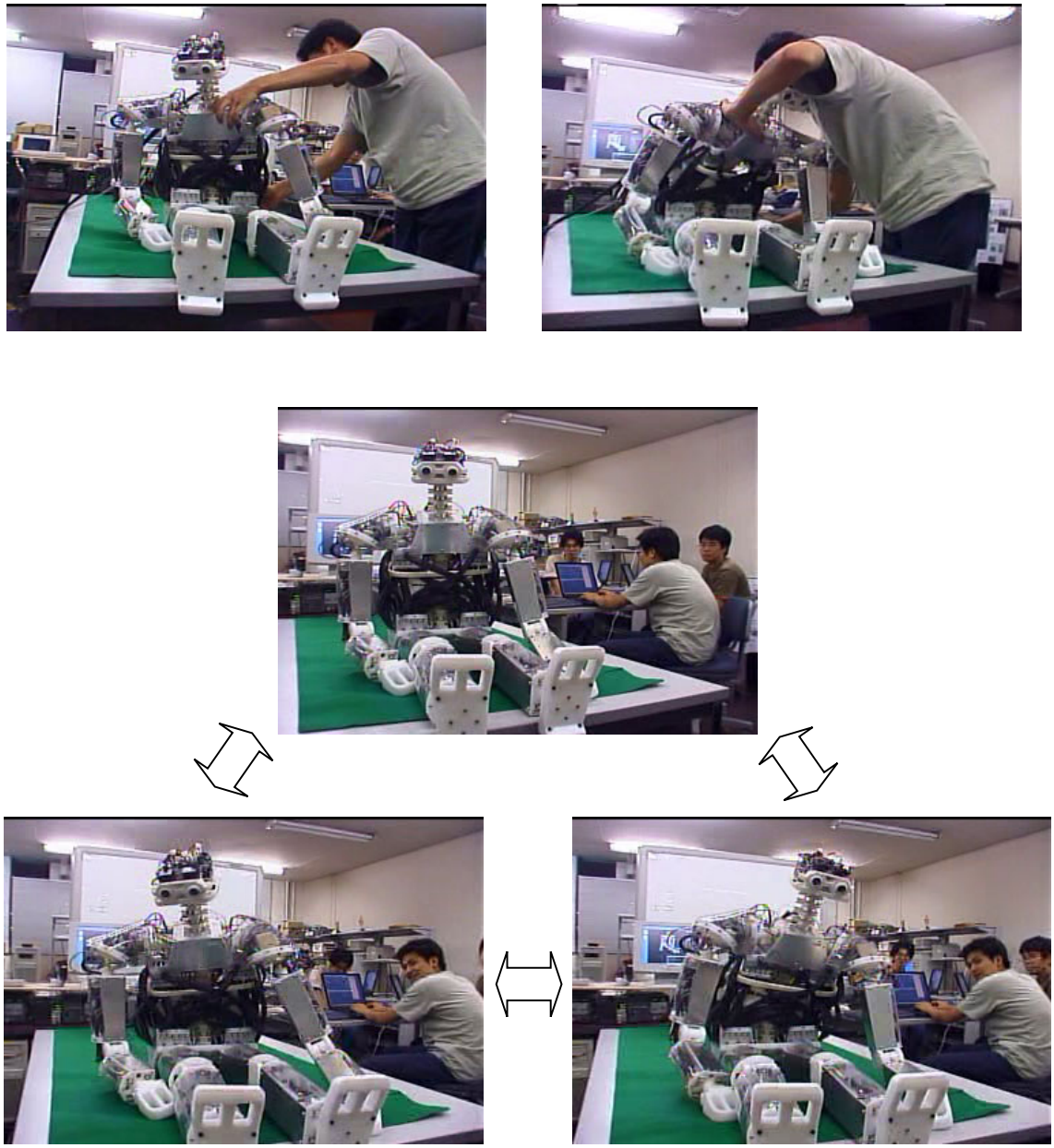


**Fig. 4.1** 複数の姿勢の教示による動作 (ロボット: 腱太)
Teaching postures and playing back them (robot:Kenta)

おきにデータを取り出し再生すると倍の速さで動作した．2 つおきにすると 3 倍になった．4 倍・8 倍も試行したところ，4 倍は 4 倍の速さで動作したが，8 倍はモータの最高速度を超えたようで，速い動作の部分は追従しきれなかった．

## 4.3 脊椎構造の姿勢制御における幾何モデルの在り方

本節では，脊椎構造の姿勢を幾何学的な表現で扱い，ある程度幾何学的表現に基づく制御量を制御するための枠組みを検討する．

### 4.3.1 姿勢の表現

本論文で提案する脊椎構造 (3.4.2節) の姿勢を表現する際に用いる幾何学的数値としては，脊椎を構成する各節の変位を用いて表現する．各節は並進の変位を持たない (3.4.2節参照) ので roll,pitch,yaw 軸回りの角度変位のみで表現できる．脊椎の第 $i$ 節の変位 (roll,pitch,yaw) を $\phi_i$， $\theta_i$， $\psi_i$ とし，脊椎の全 $n$ 節の変位を，

$$\boldsymbol{\theta} \quad = \quad (\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2, \cdots, \phi_n, \theta_n, \psi_n)^T \tag{4.1}$$

と表すことができる．これは，脊椎構造の姿勢を表現する．

### 4.3.2 筋長と関節変位の関係

$j$ 番目の筋の長さを $m_j$ とし，全筋の筋長ベクトルを，

$$\boldsymbol{q} \quad = \quad (l_1, l_2, \cdots, l_m)^T \tag{4.2}$$

とすると，

$$\boldsymbol{\theta} \quad = \quad f\left(\boldsymbol{q}\right) \tag{4.3}$$

$$\boldsymbol{q} \quad = \quad f^{-1}\left(\boldsymbol{\theta}\right) \tag{4.4}$$

なる $f$ および $f^{-1}$ は，それぞれ，筋長から姿勢への運動学 (kinematics) および姿勢から筋長を求める逆運動学 (inverse kinematics) を表す．一般に $f$ および $f^{-1}$ は非線形である．

本研究では，第 4.1節に述べたように，従来のロボットと類似の方法論に基づく制御とセンサベーストな制御の両面により，脊椎構造を持つロボットの制御を考えてゆく．本節 (第 4.3節) では，幾何学ベースの制御ということで，式 (4.3)，式 (4.4) がどこまで正確にできるようになるかという問題に取り組んでゆく．

### 4.3.3 関節変位 (関節座標系) と位置・姿勢 (直行座標系) の関係

ところで， 4.3.2節の筋長空間 (筋長座標系) と関節変位空間 (関節座標系) の関係は，関節変位表現 (関節座標系) と位置・姿勢表現 (直交座標系) の関係の扱い方と類似点がある．筋駆

動ではないロボットにおける関節空間と三次元直交座標空間の関係の取り扱いを簡単に整理しておく．

あるリンク (0) から見たあるリンク $(i)$ の位置・姿勢 (直交座標空間の表現) を

$$ {}^{\mathrm{o}}\boldsymbol{x}_i \;=\; (x\,,\; y\,,\; z\,,\; \phi\,,\; \theta\,,\; \psi)^T \tag{4.5} $$

とし，その間にある関節の変位 (関節空間表現) を $\boldsymbol{\theta}_{\mathrm{o}i}$ とすると，

$$ {}^{\mathrm{o}}\boldsymbol{x}_i \;=\; g\,(\boldsymbol{\theta}_{\mathrm{o}i}) \tag{4.6} $$

$$ \boldsymbol{\theta}_{\mathrm{o}i} \;=\; g^{-1}\quad ({}^{\mathrm{o}}\boldsymbol{x}_i) \tag{4.7} $$

なる， $g$ は関節変位空間からリンク位置・姿勢への順運動学を表し， $g^{-1}$ はその逆運動学を表す．一般に $g$ および $g^{-1}$ は非線形である．

一般に $g$ は回転と平行移動の演算の組み合わせで表現できるが，その逆関数である $g^{-1}$ を一般に求めることはできない． $g$ は非線形であるが，微小変化の範囲で線形近似することにより，関節変位の変化率 (関節角速度) と姿勢の変化率 (直行座標系での速度・角速度) の関係は次のように表せる．

$$ {}^{0}\dot{\boldsymbol{x}}_i \;=\; \boldsymbol{J}_j\;\dot{\boldsymbol{\theta}}_{\mathrm{o}i} \tag{4.8} $$

$\boldsymbol{J}_j$ は $\boldsymbol{\theta}_{\mathrm{o}i}$ の関数であり，関節変位とリンク位置・姿勢の間のヤコビアンである[1)]．

$$ \boldsymbol{J}_j(\boldsymbol{\theta}_{\mathrm{o}i}) \;=\; \begin{pmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \cdots & \frac{\partial x}{\partial \theta_i} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \cdots & \frac{\partial y}{\partial \theta_i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \psi}{\partial \theta_1} & \frac{\partial \psi}{\partial \theta_2} & \cdots & \frac{\partial \psi}{\partial \theta_i} \end{pmatrix} \tag{4.9} $$

$\boldsymbol{J}_j$ の各要素は数値的に求めることができる．関節角度空間から直交座標空間への変換 (順運動学 $g$) が，回転移動・平行移動の組み合わせで表現できるので，関節角度を微小変化させた時の直交座標空間の数値 $(x,\ y,\ z,\ \phi,\ \theta,\ \psi)$ の微小変化量を求めればよい．

$$ \frac{\partial x}{\partial \theta_1} \;=\; \lim_{\delta\theta_1 \to 0} \frac{\delta x}{\delta \theta_1} \tag{4.10} $$

$\boldsymbol{J}_j$ の疑似逆行列 (pseudo inverse matrix) [2)] ${\boldsymbol{J}_j}^{+}$ を用いて，

$$ \dot{\boldsymbol{\theta}}_{\mathrm{o}i} \;=\; {\boldsymbol{J}_j}^{+}\quad {}^{0}\dot{\boldsymbol{x}}_i \tag{4.11} $$

---

[1)] 添字の $j$ は joint を示す．

[2)] $AA^{+}A = A$， $A^{+}AA^{+} = A^{+}$， $(AA^{+})^T = AA^{+}$， $(A^{+}A)^T = A^{+}A$ が成り立つ行列 $A^{+}$ を $A$ の疑似逆行列と呼ぶ．擬似逆行列は， $(A^TA)^{-1}A^T$ により求められる．

により，直交座標系における速度から関節座標系における速度への変換が計算できる．直交座標空間の自由度 ($^0\boldsymbol{x}_i$ の要素数) より関節座標空間の自由度 ($\boldsymbol{\theta}_{\mathrm{o}i}$ の要素数) の方が大きい場合，関節速度 $\dot{\boldsymbol{\theta}}_{\mathrm{o}i}$ のノルムが最少となる解を与える．

ヤコビアンは，直交座標系における力・モーメント (リンク間力) と関節座標系における力・トルク (関節力) の関係を求めるために利用することができる．例えば，あるリンク (0) から見たあるリンク ($i$) に発生する力・モーメント $^0\boldsymbol{f}_i$・$^0\boldsymbol{n}_i$ に等価な関節動力 $\boldsymbol{\tau} = (\tau_1\,, \tau_2\,, \cdots\,, \tau_i)^T$ は，ヤコビアンの転置行列を用いて，

$$\boldsymbol{\tau}_{\mathrm{o}i} = \boldsymbol{J}_j{}^T \begin{bmatrix} {}^0\boldsymbol{f}_i \\ {}^0\boldsymbol{n}_i \end{bmatrix} \tag{4.12}$$

により計算される [13].

### 4.3.4 筋長ヤコビアン

関節角度空間と直交座標空間の間の非線形関数 (式 (4.6)，式 (4.7) の $g$，$g^{-1}$) を微分により線形に近似する (式 (4.8)，式 (4.11) という手法を，筋駆動ロボットの筋長空間と関節角度空間の間の非線形関数 (式 (4.3)，式 (4.4) の $f$，$f^{-1}$) に関しても利用することを考える．筋長の微少変化 $\delta\boldsymbol{q}$ とそれに対応した関節変位の微少変化率 $\delta\boldsymbol{\theta}$ の関係を，

$$\delta\boldsymbol{q} = \boldsymbol{J}_m\, \delta\boldsymbol{\theta} \tag{4.13}$$

と表せる．$\boldsymbol{J}_m$ を筋長ヤコビアンと呼ぶ[3]．$\boldsymbol{J}_m$ は筋長 $\boldsymbol{q}$ の関数である．$n$ 節の脊椎を $m$ 本の筋により駆動する場合，脊椎の各節の変位は roll,pitch,yaw($\phi$，$\theta$，$\psi$) の 3 つのパラメータで表現されるので，$\boldsymbol{J}_m$ は $3n$ 行 $m$ 列の行列であり，

$$\boldsymbol{J}_m(\boldsymbol{q}) = \begin{pmatrix} \frac{\partial l_1}{\partial \phi_1} & \frac{\partial l_1}{\partial \theta_1} & \frac{\partial l_1}{\partial \psi_1} & \frac{\partial l_1}{\partial \phi_2} & \cdots & \frac{\partial l_1}{\partial \psi_n} \\ \frac{\partial l_2}{\partial \phi_1} & \frac{\partial l_2}{\partial \theta_1} & \frac{\partial l_2}{\partial \psi_1} & \frac{\partial l_2}{\partial \phi_2} & \cdots & \frac{\partial l_2}{\partial \psi_n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial l_m}{\partial \phi_1} & \frac{\partial l_m}{\partial \theta_1} & \frac{\partial l_m}{\partial \psi_1} & \frac{\partial l_m}{\partial \phi_2} & \cdots & \frac{\partial l_m}{\partial \psi_n} \end{pmatrix} \tag{4.14}$$

により計算される．$\boldsymbol{J}_m$ の各要素は，脊椎の姿勢を微小変化させ，その時の筋の長さの微小変化を計算することで，算出することができる．

$$\frac{\partial l_1}{\partial \phi_1} = \lim_{\delta\phi_1 \to 0} \frac{\delta l_1}{\delta \phi_1} \tag{4.15}$$

[3] 添字の $m$ は muscle を示す．

ただし，関節空間→直交座標空間の場合式 (4.6) の $g$ が回転移動・平行移動の演算により求められるのに対し，筋長空間→関節空間や関節空間→筋長空間の場合筋同士・筋とボディの物理的な干渉等の問題があり，式 (4.3) の $f$ や式 (4.4) の $f^{-1}$ を正確に求めるのは難しい．特に，$f$ は解がない場合もある (4.4.1節).

$\boldsymbol{J}_m$ を求める事ができると，式 (4.11) のように疑似逆行列 ${\boldsymbol{J}_m}^+$ を用いて，

$$\delta\boldsymbol{\theta} \;=\; {\boldsymbol{J}_m}^+\,\delta\boldsymbol{q} \tag{4.16}$$

により，$\delta\boldsymbol{\theta}$ のノルムを最少にするような解を算出することができる.

筋長ヤコビアンの擬似逆行列を利用することで，関節空間での力・トルクを実現するための筋長力組み合わせを求めることができる． $\boldsymbol{q}$ に対応した各筋の張力を,

$$\boldsymbol{T} \;=\; (T_1\,,\,T_2\,,\,\cdots\,,\,T_m)^T \tag{4.17}$$

とし， $\boldsymbol{\theta}$ に対応した関節発生力を,

$$\boldsymbol{\tau} \;=\; (\tau_1\,,\,\tau_2\,,\,\cdots\,,\,\tau_{3n})^T \tag{4.18}$$

とすると，仮想仕事の原理より,

$$\begin{aligned} &\delta\phi_1\tau_1 + \delta\theta_1\tau_2 + \delta\psi_1\tau_3 + \cdots + \delta\psi_n\tau_{3n} \\ =\quad &-\,(\;\delta l_1 T_1 + \delta l_2 T_2 + \cdots + \delta l_m T_m\;) \end{aligned} \tag{4.19}$$

すなわち,

$$\dot{\boldsymbol{\theta}}^T\boldsymbol{\tau} \;=\; -\,\dot{\boldsymbol{q}}^T\boldsymbol{T} \tag{4.20}$$

となる．ただし，負号は筋張力を引っ張り方向を正と定義したことによる．式 (4.20) と式 (4.13) より， $(AB)^T = B^TA^T$ を利用して,

$$\dot{\boldsymbol{\theta}}^T\boldsymbol{\tau} \;=\; -\dot{\boldsymbol{\theta}}^T{\boldsymbol{J}_m}^T\boldsymbol{T} \tag{4.21}$$

よって,

$$\boldsymbol{\tau} \;=\; -{\boldsymbol{J}_m}^T\boldsymbol{T} \tag{4.22}$$

となり，筋張力 $\boldsymbol{T}$ から関節駆動力 $\boldsymbol{\tau}$ を求めることができる． $\boldsymbol{J}_m$ が行フルランク行列であれば式 (4.22) から,

$$\boldsymbol{T} \;=\; -({\boldsymbol{J}_m}^T)^+ + \boldsymbol{Z}\boldsymbol{\zeta} \tag{4.23}$$

が求められる．ここで，

$$\boldsymbol{Z} = \boldsymbol{I}_m - (\boldsymbol{J}_m{}^T)^+ \boldsymbol{J}_m{}^T \quad (4.24)$$

($\boldsymbol{I}_m$ は $m$ 行 $m$ 列の単位行列)

であり，$\boldsymbol{\zeta}$ は $m$ 次元の任意ベクトルである．筋張力 $\boldsymbol{T}$ の各要素は負にすることはできないが，式 (4.23) の第一項は $\boldsymbol{T}$ のノルム最少の解であり，負の要素を含む可能性がある．そこで，零空間上のベクトル $\boldsymbol{Z\zeta}$ を足して張力が正になるようにする [18]．$\boldsymbol{\zeta}$ は $\boldsymbol{T}$ のどの要素も負にならない最少の値になるように決める．

## 4.4 本研究での脊椎構造の姿勢制御における幾何モデル

### 4.4.1 概要

第 4.3節に述べた要素のうち，本研究ではどこまで実現するか．まず，脊椎の幾何学的数値 $\boldsymbol{\theta}$ で与えられた目標姿勢になるように，ある程度制御できることが求められるであろう．すなわち，式 (4.4) の $f^{-1}$ が最低限必要である．これがあれば，姿勢 $\boldsymbol{\theta}$ を実現するための筋長 $\boldsymbol{q}$ を求めることができる．しかし，筋同士の物理的干渉・筋とボディ構造材との物理的干渉を計算するためには，筋の太さ・経路・張力バランス・椎間板の反力を考慮しなければならない．また筋の干渉問題には外力も影響があるので，干渉問題をどう扱うか，どこまで干渉を考慮するか，という問題がある．

目標姿勢と実現された姿勢の誤差を求める，シミュレーション環境を利用する，等の場合には，式 (4.3) の $f$ も必要となる．こちらは，逆問題になり一般的な解を求めることは困難である．筋長の組合わせによっては解が無い場合もある．簡単な例を示す．回転 1 自由度の関節を拮抗 2 筋により駆動する場合，2 筋とも同時に引くということは物理的に不可能である．

本研究では，上述の問題に対する解決方法として，$f^{-1}$ は物理的干渉を無視した計算を行い筋単位の制御モードの切り替えにより干渉の回避を行い，$f$ は干渉を無視した $f^{-1}$ のモデルを教師としてニューラルネットワーク (NN) の学習により実現した．

### 4.4.2 姿勢情報から筋長情報への変換

幾何モデルには，筋の取り付け点・引っ張り点 (3.7.2節) の幾何的位置情報を持たせ，関節姿勢 $\boldsymbol{\theta} = (x, y, z, \phi, \theta, \psi)^T$ が変化すると，取り付け点・引っ張り点はそれが存在するリンク $i$ の座標系の変化に伴い変化する．リンク $i$ に存在する筋 $j$ の第 $k$ 番目の取り付け点・引っ張り点のリンク $i$ の座標系での位置を，${}^i\boldsymbol{p}_{jk} = ({}^ix_{p_{jk}}, {}^iy_{p_{jk}}, {}^iz_{p_{jk}})^T$ ・ ${}^i\boldsymbol{q}_{jk} = ({}^ix_{q_{jk}}, {}^iy_{q_{jk}}, {}^iz_{q_{jk}})^T$

と表現すると，物理的な干渉を無視した筋 $j$ の長さ $l_j$ は,

$$l_j \;=\; \sum_k |\,{}^0\boldsymbol{p}_{jk} - {}^0\boldsymbol{q}_{jk}\,| \tag{4.25}$$

により計算される．筋の物理的干渉の問題の解決に関しては第 4.7節に述べる．

### 4.4.3 筋長情報から姿勢情報への変換

各筋の長さが与えられたとき脊椎の姿勢を求める計算 (式 (4.3) の $f$) は一般には解けない．ここで求めるのは，実機の筋長を計測したときにその時の姿勢を推定することである．その方法として考えられる手法を以下に挙げる．

#### ヤコビアン逆行列を利用

筋長ヤコビアンの疑似逆行列 ${\boldsymbol{J}_m}^+$ を用いることで筋長の変化速度 $\dot{\boldsymbol{q}}$ から関節姿勢の変化速度 $\dot{\boldsymbol{\theta}}$(ノルム最少) を求めることができる (式 (4.16))．初期姿勢を $\boldsymbol{\theta}_{\rm o}$，初期筋長 (初期姿勢の時の筋長) を $\boldsymbol{q}_{\rm o}$ とすると，筋長 $\boldsymbol{q}$ のときの姿勢 $\boldsymbol{\theta}$ は,

$$\boldsymbol{\theta} \;=\; \boldsymbol{\theta}_{\rm o} + \int_{\boldsymbol{q}_{\rm o}}^{\boldsymbol{q}} \left(\boldsymbol{J}_m\left(\boldsymbol{\theta}\right)\right)^+ d\boldsymbol{q} \tag{4.26}$$

により算出できる．これを数値的に求めるために，適当な $\delta\boldsymbol{q} = (\boldsymbol{q} - \boldsymbol{q}_{\rm o})/(n-1)$ に分割して,

$$\boldsymbol{\theta} \qquad \boldsymbol{\theta}_{\rm o} + \sum_{i=1}^{n} \left(\boldsymbol{J}_m\left(\boldsymbol{\theta}_{i-1}\right)\right)^+ \delta\boldsymbol{q} \tag{4.27}$$

により算出できる．ただし， $\boldsymbol{J}_m(\boldsymbol{\theta}_{i-1})$ は,

$$\boldsymbol{J}_m(\boldsymbol{\theta}_{i-1}) \;=\; \boldsymbol{\theta}_{i-2} + \left(\boldsymbol{J}_m\left(\boldsymbol{\theta}_{i-1}\right)\right)^+ \delta\boldsymbol{q} \tag{4.28}$$

の漸化式により計算する．

姿勢が大きく変化する場合は，分割数 $n$ を大きくとらなければならない．ここでは，分割数 $n$ として $|\,\boldsymbol{q} - \boldsymbol{q}_{\rm o}\,|$ の整数部分を用いた．ただし， $\boldsymbol{q}$ の各要素の単位は筋長 [mm] である．5 節の脊椎を持つ人間型ロボット Cla の脊椎 (第 3.6節) において，この手法で筋長から関節変位を計算して， 4.4.2節の式 (4.25) の変換と比較した．つまり， $\boldsymbol{R}_a = (\phi_a, \theta_a, \psi_a)^T$ として，$\boldsymbol{\theta}_a \;=\; ({\boldsymbol{R}_a}^T, {\boldsymbol{R}_a}^T, {\boldsymbol{R}_a}^T, {\boldsymbol{R}_a}^T, {\boldsymbol{R}_a}^T)^T$ なる関節変位 $\boldsymbol{\theta}_a$ から式 (4.25) により求めた筋長 $\boldsymbol{l}_a$ を式 (4.27) により出力 $\boldsymbol{\theta}_b$ を計算し，最初の $\boldsymbol{\theta}_a$ と比較した．結果を Table 4.2 に示す．

表からわかる通り，誤差は非常に大きく，計算時間も非常にかかる．誤差が大きい原因は，おそらく筋長の変化を単純に $n$ 分割しているため，筋長の変化が一様になってしまい，$\delta\boldsymbol{\theta}$ の変化が大きく $\delta\boldsymbol{q}$ を細かくすべきところでもおなじ割合であるためというのが一つの原

Table 4.2 ヤコビアン疑似逆行列と数値積分による筋長から姿勢への変換
The calculation of joint-angles from muscle-lengths using pseudo inverse of Jacobian matrix and integration

| 正しい値 $\boldsymbol{R}_a$ deg/ 節 | 計算結果 (各節平均) $(\overline{\phi}_a, \overline{\theta}_a, \overline{\psi}_a)$ deg/ 節 | ‖誤差‖ の平均 $(\overline{\lvert\phi_E\rvert}, \overline{\lvert\theta_E\rvert}, \overline{\lvert\psi_E\rvert})$ deg/ 節 | ‖誤差‖最大となる誤差 $(\phi_{max}, \theta_{max}, \psi_{max})$ deg/ 節 | 分割数 $n$ | 計算時間 $T_c$ s |
|---|---|---|---|---|---|
| $\phi_a$=1 | (3.50, 1.79, −7.01) | (73.9, 67.6, 146) | (115, 161, −215) | 6 | 57.3 |
| $\theta_a$=0 | (0.483, 0.279, −0.533) | (7.51, 13.6, 4.39) | (−12.8, 30.3, −9.62) | 12 | 115 |
| $\psi_a$=0 | (0.537, 0.409, −0.613) | (144, 108, 107) | (−181, 168, 193) | 23 | 218 |
| $\phi_a$=0 | (0.706, −2.48, −1.76) | (76.6, 25.6, 14.0) | (130, 53.3, 30.5) | 2 | 19.4 |
| $\theta_a$=1 | (1.26, −0.377, 0.945) | (25.8, 10.9, 10.2) | (54.4, −25.0, 23.1) | 3 | 28.9 |
| $\psi_a$=0 | (4.89, 3.35, 8.74) | (25.5, 23.6, 20.8) | (41.4, 53.1, 63.3) | 6 | 57.4 |
| $\phi_a$=0 | (−3.00, −4.14, −107) | (192, 103, 105) | (248, 180, −189) | 2 | 19.5 |
| $\theta_a$=0 | (−70.4, −20.6, −65.4) | (71.6, 87.0, 133) | (−222, −166, −186) | 4 | 38.5 |
| $\psi_a$=1 | (5.57, 19.3, 4.34) | (73.8, 49.8, 32.6) | (−134, 61.5, −92.4) | 8 | 76.3 |
| $\phi_a$=5 | (117, 108, 108) | (207, 190, 0.0) | (360, 360, 0) | 29 | 277 |
| $\theta_a$=0 | (36.0, 24.5, 108) | (244, 169, 177) | (293, −258, 272) | 57 | 539 |
| $\psi_a$=0 | (−10.8, 63.8, −5.15) | (45.4, 116, 312) | (126, −360, 360) | 113 | 1072 |
| $\phi_a$=0 | (0.308, 64.9, 167) | (117, 101, 85.5) | (−360, −360, 360) | 8 | 76.7 |
| $\theta_a$=5 | (108, 77.6, 36.0) | (216, 197, 216) | (360, 360, 360) | 16 | 152 |
| $\psi_a$=0 | (16.8, 6.64, 7.32) | (169, 164, 158) | (232, 243, 225) | 31 | 297 |
| $\phi_a$=0 | (−15.8, 180, −28.7) | (304, 115, 195) | (360, 360, −360) | 10 | 95.6 |
| $\theta_a$=0 | (36.0, 61.8, 108) | (216, 220, 216) | (360, 360, 360) | 20 | 190 |
| $\psi_a$=5 | (36.0, 108, 36.0) | (176, 172, 171) | (−232, 202, 212) | 40 | 380 |
| $\phi_a$=10 | (108, 1.30, 108) | (0.0, 181, 0.0) | (0.0, −360, 0.0) | 56 | 528 |
| $\theta_a$=0 | (36.0, 42.1, 108) | (0.0, 210, 216) | (0.0, 360, 360) | 112 | 1056 |
| $\psi_a$=0 | (36.0, −6.58, 67.4) | (144, 155, 103) | (−360, −360, −360) | 223 | 2100 |
| $\phi_a$=0 | (36.0, 36.0, 36.0) | (216, 179, 72.0) | (360, 360, −360) | 16 | 151 |
| $\theta_a$=10 | (82.1, 108, −36.0) | (242, 148, 288) | (360, 297, 360) | 32 | 302 |
| $\psi_a$=0 | (−36.0, 8.69, −108) | (216, 140, 175) | (360, 360, −360) | 63 | 594 |
| $\phi_a$=0 | (180, −17.8, 108) | (144, 198, 216) | (360, 360, 360) | 20 | 189 |
| $\theta_a$=0 | (108, 108, 108) | (170, 144, 144) | (360, 360, 360) | 40 | 378 |
| $\psi_a$=10 | (36.0, 40.0, 36.0) | (216, 175, 144) | (360, −360, 360) | 80 | 755 |

因ではないかと考えられる．また，ヤコビアンを用いて積分により求めるという手法は，小さい誤差が生じると次のステップですぐに発散していきやすい．こうした問題は，$\delta \boldsymbol{q}$ をもっと小さくすることにより解消されるかも知れないが，すでに $\delta \boldsymbol{q}$ のノルムを 0.5 としても ( Table 4.2 中で三種類ある分割数のうち最も分割数が粗い場合) 計算時間は数十秒〜数百秒かかるため，これ以上分割数を細かくすることは非現実的である．

### ニューラルネットワーク (NN)

4.4.2節の式 (4.25) により計算される姿勢 (関節変位) →筋長を教師データとして NN により学習し，筋長→姿勢 (関節変位) を算出する．脊椎の姿勢空間 (関節変位空間) は $n$ 節の脊椎の場合 $3n$ 次元になる．教師データをここから均一にサンプリングすると，節数に依るが膨大な教師データの量になるので，ここでは各節が roll,pitch,yaw 軸回りに均等に曲がった姿勢のみを教師データとすることにした．すなわち，$\boldsymbol{\theta} = (\phi_1, \theta_1, \psi_1, \cdots, \psi_n)^T$ において，

$$\begin{cases} \phi_1 = \phi_2 = \cdots = \phi_n = \phi \\ \theta_1 = \theta_2 = \cdots = \theta_n = \theta \\ \psi_1 = \psi_2 = \cdots = \psi_n = \psi \end{cases} \tag{4.29}$$

となる $\tilde{\boldsymbol{\theta}} = (\phi, \theta, \psi, \cdots, \psi)^T$ の場合のみを教師データとした．

5 節の脊椎を持つ人間型ロボット Cla の脊椎 (第 3.6節) において，この手法で式 (4.25) の変換と比較した．条件設定は次のようにした．

- **入出力層**
  入力は筋 8 本の長さとし，出力は各節 3 自由度 ×5 節で 15 の関節変位とした．筋長は [mm] 関節変位は [deg] で表されるが， NN への入出力は正規化して行った．

- **教師データ**
  各節均等に曲がるとして， $\phi$， $\theta$， $\psi$ をそれぞれ $(-15, -10, -5, 0, 5, 10, 15)$[deg] の 7 種類ずつで，全組合わせは $7^3 = 343$ 種類の姿勢とその時に式 (4.25) により計算した筋長を教師データとして用意した．

- **中間層**
  中間層は，合計 8 種類のタイプを試した．ここでは， NN の層構成を (入力層のニューロン 数 – 中間層のニューロン数 – 出力層のニューロン数) の形で表す．中間層を二つ持つ 4 層構造を 1 種類 (8-10-10-15)， 3 層構造を 7 種類 ((8-12-15)， (8-20-15)， (8-30-15)， (8-40-15)， (8-50-15)， (8-70-15)， (8-100-15)) を試行した．カッコ内は入力から出力までの各層のニューロン数である．

- **学習パラメータ**

  学習はバックプロパゲーションにより行った．各ニューロンは $i$–$j$–$k$ で $j$ が自分の時,

$$W_{ij}(t+1) \quad = \quad W_{ij}(t) + \eta\delta_j O_i \tag{4.30}$$

  により重みを更新した． $\delta_j$ は次のように計算される．

$$\begin{aligned} \delta_j &= \lambda O_j(1-O_j)(T_j - O_j) \quad \text{(in output layer)} \\ \delta_j &= \lambda O_j(1-O_j)\sum_{k=0}^{n}\delta_k W_{jk} \quad \text{(in hidden layer)} \end{aligned} \tag{4.31}$$

  各パラメータの意味を以下に示す．カッコ内は今回計算に利用した値である．

$$\begin{cases} W_{ij} & \text{重み} \\ \delta_j & \text{重み更新量} \\ O_j & \text{出力} \\ \eta & \text{重み更新のゲイン} \quad (1.0) \\ T_j & \text{教師信号} \\ \lambda & \text{シグモイド関数の傾斜を表すパラメータ} \quad (1.0) \end{cases}$$

  である．

各場合の学習結果を **Fig. 4.2** 〜 **Fig. 4.9** に示す．横軸は学習回数を示し，縦軸は出力と教師信号との誤差 (関節変位 (単位 [deg])15 個分) のノルムを示す．試行した順番は，まず，層構成が (8-10-10-15)， (8-20-15)， (8-50-15)， (8-100-15) の 4 つの場合を始めに試行した．4 層構成の (8-10-10-15) は,

- 学習の途中から誤差が再び大きくなっている．
- 汎化能力が 3 層構成のものに比べて劣るように見える (後述)．
- 3 層構成と 4 層構成で大きな差は無い．
- 3 層構成なら層構成の検討の際に中間層のニューロン数のみを変えて試せば良いが， 4 層構成だと中間層が二層になるので試行するべきパターンが大幅に増大する．

といった理由から， 3 層構成で中間層のニューロン数を検討する事にした． 3 層構成の中で最も良かった (8-50-15) 近辺で (8-30-15)， (8-70-15) の 2 つの場合を試行した．誤差のノルムの最小値が (8-70-15) ＞ (8-50-15) ＞ (8-30-15) の順だったので，最後に (8-20-15) と (8-40-15) を試行した結果， (8-40-15) が最も良い結果が得られた．各層構成の場合の誤差のノルムの最小値を Table 4.3 に示す．

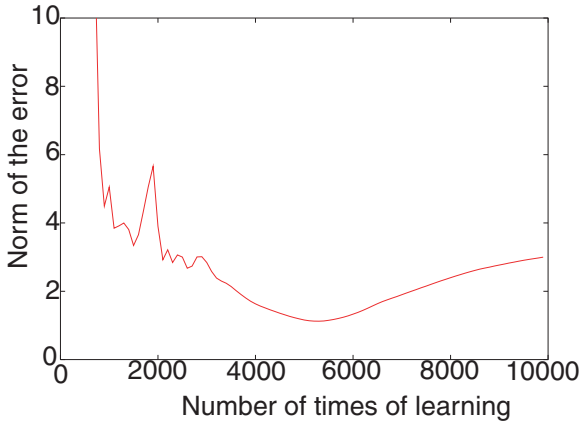



**Fig. 4.2** 中間層 2 層 (8-10-10-15)
Layer structure (8-10-10-15)

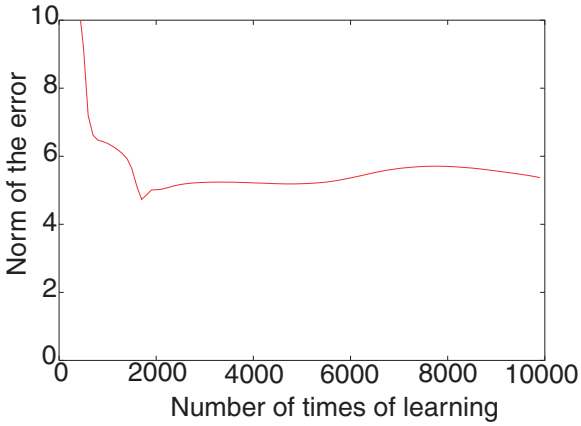



**Fig. 4.3** 中間層 1 層 (8-12-15)
Layer structure (8-12-15)

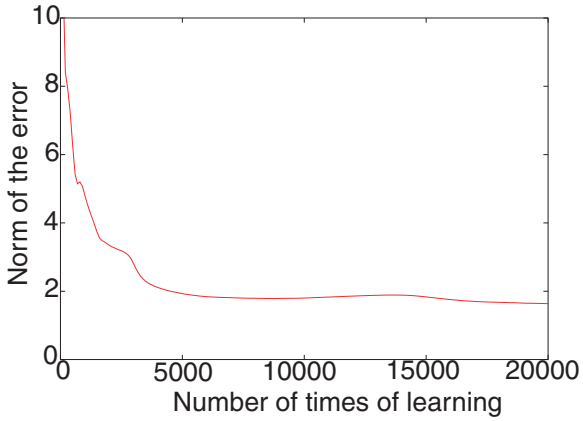



**Fig. 4.4** 中間層 1 層 (8-20-15)
Layer structure (8-20-15)

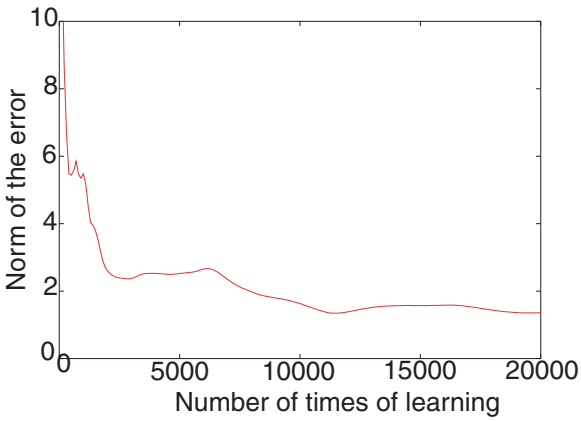



**Fig. 4.5** 中間層 1 層 (8-30-15)
Layer structure (8-30-15)

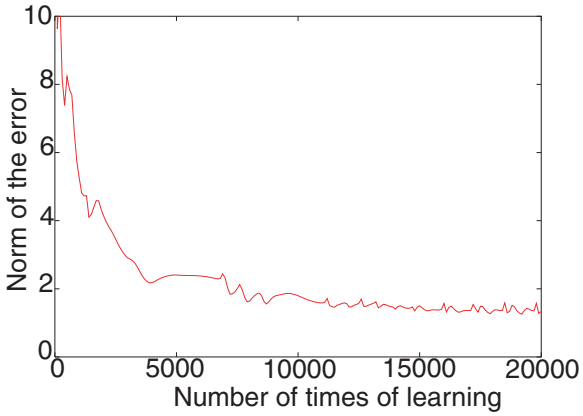



**Fig. 4.6** 中間層 1 層 (8-40-15)
Layer structure (8-40-15)

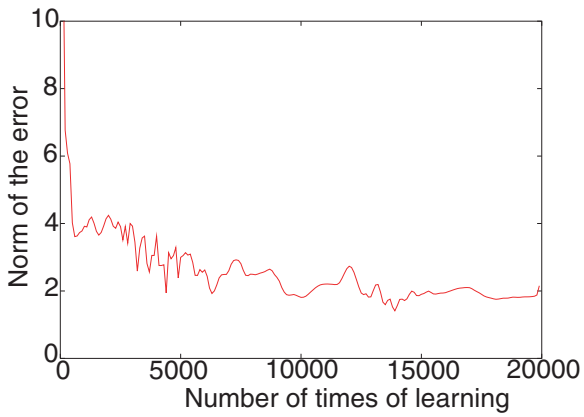



**Fig. 4.7** 中間層 1 層 (8-50-15)
Layer structure (8-50-15)

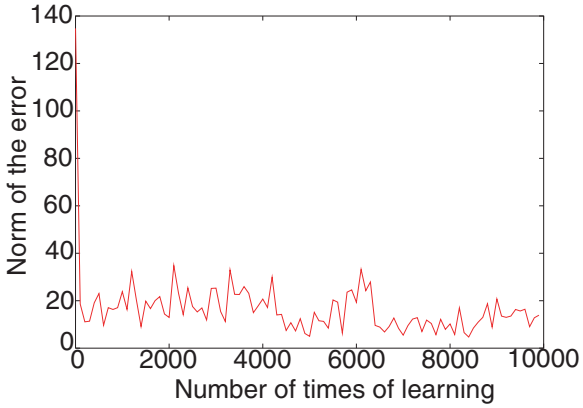



**Fig. 4.8** 中間層 1 層 (8-70-15)
Layer structure (8-70-15)

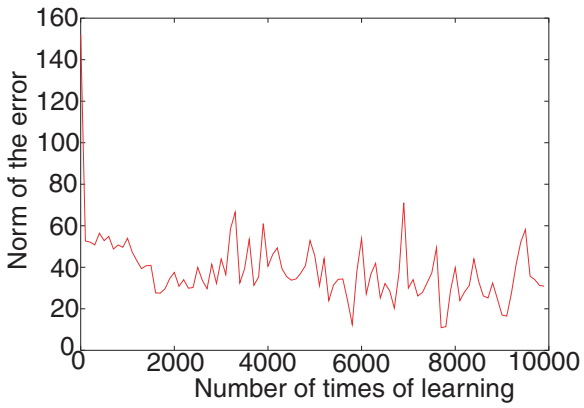



**Fig. 4.9** 中間層 1 層 (8-100-15)
Layer structure (8-100-15)

Table 4.3　各層構成の場合の誤差のノルムの最小値
Minimum of norm of the error

| 層構成 | 誤差のノルムの最小値 | 備考 |
|---|---|---|
| 8-10-10-15 | 1.12 | ランダム入力時の誤差のノルムの平均は 2.51 |
| 8-12-15 | 4.73 | |
| 8-20-15 | 1.58 | |
| 8-30-15 | 1.35 | |
| 8-40-15 | 1.26 | ランダム入力時の誤差のノルムの平均は 1.95 |
| 8-50-15 | 1.75 | |
| 8-70-15 | 4.71 | 収束せず |
| 8-100-15 | 10.9 | 収束せず |

層構成が (8-10-10-15) の場合と (8-40-15) の場合で，誤差のノルムが最少になった時の NN を使用して，ランダムで 50 種類の正しいデータ与えて誤差を計測したランダムに 50 種類の各関節が均等に曲がった姿勢 $\boldsymbol{\theta}_t$ を生成し，式 (4.25) により筋長を計算し，それを入力として与えたときの出力 $\boldsymbol{\theta}_s$ と $\boldsymbol{\theta}_t$ の差を計算した)．その誤差のノルムの平均値を比較すると ( Table 4.3 備考欄)， 4 層構成の (8-10-10-15) の方が 3 層構成の (8-40-15) より教師データに対する再現性は高いが，教師データの中間値などの入力があった場合に (8-40-15) の方が (8-10-10-15) より正解に近い出力をするという傾向がわかる．つまり (8-10-10-15) より (8-40-15) の方が汎化能力が高いと考えられる．

上記の考察より， Cla の脊椎構造の筋長→姿勢の変換に NN を利用する場合は， (8-40-15) の層構成が適していると言える．なお，ここまでの議論は 15 の関節角度誤差のノルムを利用した．他に指標として考えられるのは， (1) 誤差の平均値， (2) 誤差の最大値，等があるが， (1) は大きな誤差を持つ節が一つあると誤差の平均が小さくても姿勢としてはあまり正しくない， (2) は誤差の最大値が小さくても誤差の平均が大きいと，全体としてあまり正しくない姿勢になる，という予想に基づいて，誤差のノルムを評価指標に利用した．なお，関節誤差の最大値は，各節が均等に曲がる (式 (4.29) を満たす) 入力に対しては， (8-40-15) の層構成の場合， 0.5° 未満程度である．

**各手法の比較**

ヤコビアン逆行列を用いる手法は，非常に大きな計算時間がかかる．特に初期姿勢から離れた姿勢になると，実用からは程遠い時間 (数十分等) かかる．また，誤差も NN を利用する場合と比較して大きい．よって，本研究では，筋長空間から関節変位空間への変換 (式 (4.3) の $\boldsymbol{f}$) は，関節変位空間から筋長空間への変換 (式 (4.4) の $\boldsymbol{f}^{-1}$) を教師信号として利用して学習したニューラルネットワークによるものとする．

### 4.4.4 モデルのインタフェース

実際に脊椎構造を持つロボットの幾何モデルを利用する機会は，姿勢生成・動作生成やシミュレーション・センサ状態のモデルへの反映などが挙げられる．モデルを利用するためのインタフェースは，既存のロボットの姿勢・動作の生成センサ情報の利用の際と透過的なインタフェースを持つことが好ましい．モデル利用時の入出力としての， $\boldsymbol{\theta} = (\phi_1, \theta_1, \psi_1, \cdots, \psi_n)^T$ の入力と $\boldsymbol{q} = (l_1, l_2, \cdots, l_m)^T$ の出力，その逆の $\boldsymbol{q}$ の入力と $\boldsymbol{\theta}$ 出力がある．また，脊椎構造全体の姿勢の指示 ($\boldsymbol{\theta}$ 入力) のために， $\boldsymbol{\Theta} = (\phi_s, \theta_s, \psi_s)^T$ のような形で脊椎各節は均等に曲がるように姿勢を入力，あるいは，センサ情報の反映として $\boldsymbol{\Theta}$ の形で出力するインタフェースも備えるとよい．筋長 $\boldsymbol{q}$ は直接アクチュエータに与える，もしくはアクチュエータの現在地を直接受け取ることができるように，アクチュエータで扱う物理量と対応した値の列であることが求められる．また，本研究では詳細には扱わないが，ヤコビアンを用いた関節トルク制御のための筋張力制御のための力情報の計算のためのインタフェースが求められる．

並列オブジェクト指向 Lisp である EusLisp[49] を利用した，幾何モデルをベースにした人間型ロボットのソフトウェア環境 [26] があるが，本研究ではこの環境に柔軟な脊椎を持つ全身型多自由度ロボットのための機能を追加してゆくことで，ソフトウェアシステムを構築していく．ソフトウェアシステムの詳細は第 6章で述べるが，ここでは脊椎の姿勢制御に関わる部分に関し述べる．

ロボットのソフトウェア環境はオブジェクト指向に基づいて記述されている．回転 3 自由度を持つ脊椎の各節は`joint` クラスを継承し`qjoint` というクラスを定義した (6.5.2節)． Table 4.4 に`qjoint` クラスの基本インタフェースを示す．

また，脊椎を駆動する筋は新たに`general-wire` というクラスを定義した． Table 4.5 に `general-wire` クラスの基本インタフェースを示す．`:init` により引っ張り点・取り付け点はそれが属するべきリンクにアタッチされ，`qjoint` クラスの`:joint-angle` により脊椎の姿勢が更新されるとそれにともない位置が更新される．`length` は常に最新の姿勢における筋長を計算する．また，`bodies` を利用することにより，ビューワに表示されたロボットの画像に現

Table 4.4　脊椎各節を記述するクラス (qjoint) の基本インタフェース
The basic interface of the class of spine-joints (qjoint)

| メッセージ名 | 引数 | 機能 |
|---|---|---|
| :init | [名前，接続リンク等] | オブジェクト生成．初期化処理. |
| :joint-angle | [関節姿勢 $\boldsymbol{\theta}$] | $\boldsymbol{\theta}$ が与えられなければ内部変数の値を返す．与えられれば内部変数を書き替え子リンクを回転. |

Table 4.5　各筋を記述するクラス (general-wire) の基本インタフェース
The basic interface of the class of muscles (general-wire)

| メッセージ名 | 引数 | 機能 |
|---|---|---|
| :init | 駆動する節，[取り付け点・引っ張り点リスト等] | オブジェクト生成．初期化処理．引っ張り点・取り付け点 (座標系クラスオブジェクト) の生成とリンクへのアタッチ. |
| :length | —— | 引っ張り点・取り付け点のリストから経路を辿り，筋の全長を算出する. |
| :tension | [筋張力 $T$] | $T$ が与えられなければ内部変数の値を返す．与えられれば内部変数を書き替える. |
| :bodies | [半径，色] | 筋を可視化する. |

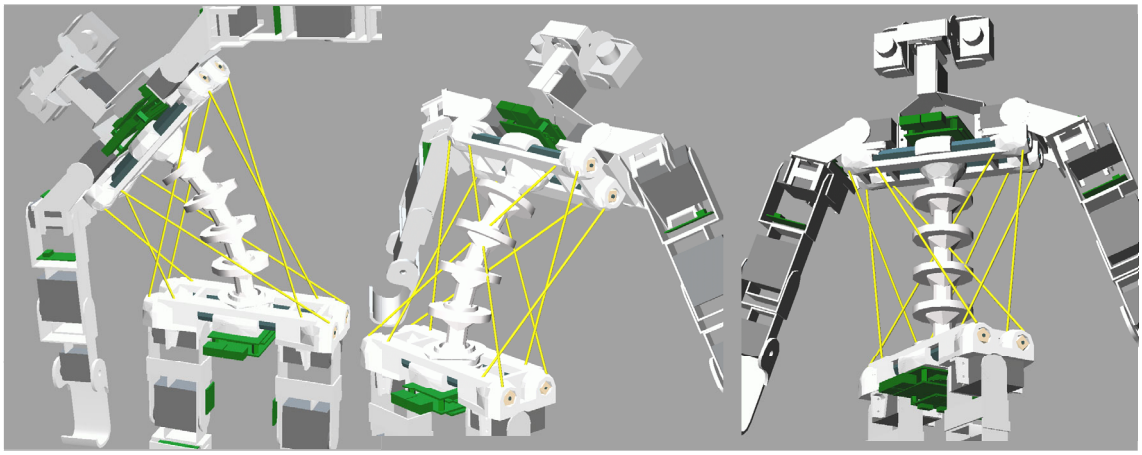


**Fig. 4.10** 脊椎・筋の幾何モデル (ロボット:Cla)
The geometric model of spine and muscles (robot:Cla)

在の筋の経路がモデル上でどのようになっているかを視覚的に確認することができる.

5 節の脊椎を持つ人間型ロボット Cla の脊椎 (第 3.6節) の姿勢を roll,pitch,yaw にそれぞれ均等に変化させたときの，モデル上での筋の様子を`:bodies` により可視化 ( Table 4.5 ) した図を **Fig. 4.10** に示す.

## 4.5 柔軟性の調節

### 4.5.1 剛性行列

これまでロボットの柔軟性の調節の研究においては，剛性行列が用いられてきた．関節の剛性行列 $\boldsymbol{K}_j$ は，関節発生力，関節変位の微少量をそれぞれ $\delta\boldsymbol{\tau}$， $\delta\boldsymbol{\theta}$ はとして，

$$\delta\boldsymbol{\tau} \quad = \quad \boldsymbol{K}_j\delta\boldsymbol{\theta} \tag{4.32}$$

と定義される．一方，筋駆動の構造の場合は，筋 $i$ の剛性 (バネ定数) を $k_i$ とし，全筋の剛性を,

$$\boldsymbol{K}_m \quad = \quad \begin{pmatrix} k_1 & 0 & \cdots & 0 \\ 0 & k_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_m \end{pmatrix} \tag{4.33}$$

という対角行列で表すと，微少な筋長変化 $\delta\boldsymbol{q}$ に対応した筋張力の微少変化量 $\delta\boldsymbol{T}$ は,

$$\delta\boldsymbol{T} \quad = \quad -\boldsymbol{K}_m\delta\boldsymbol{q} \tag{4.34}$$

という関係式で表すことができる．関節剛性が式 (4.32) により表現されるのに対し，いわゆる腱駆動型ロボットにおいては，腱長ヤコビ行列 (本論文では “筋長ヤコビアン” と記述) $\boldsymbol{J}_m$(式 (4.14)) を用いて，

$$\boldsymbol{K}_j \quad = \quad \boldsymbol{J}_m{}^T \boldsymbol{K}_m \boldsymbol{J}_m \tag{4.35}$$

となり，筋の剛性調節により関節の剛性を調節することができる [18]．筋の物理的な剛性 $\boldsymbol{K}_m$ を変更するためには，非線形ばね要素を持つ筋を利用する必要がある．

### 4.5.2 脊椎構造の柔軟性の調節

#### 筋の剛性調節

筋の柔軟性 $\boldsymbol{K}_m$ を調節することで，剛性行列に基づいて脊椎構造の柔軟性を関節レベルで調節することができる．筋の柔軟性調節は，ハードウェアによるものとソフトウェアによるものが考えられる．

ハードウェアによる筋の柔軟性調節は，衝突などソフトウェアでは応答が間に合わないような用途に有効だが，複雑な機構や重量の増加につながりやすい．方法としては，非線形ばね要素を直列に組み込むという方法 [17, 85, 18] や，バネ定数可変なばね機構を組み込むという方法 [77, 63, 54] などが考えられる．

ソフトウェアによる筋の柔軟性調節は，上に述べたような応答性の問題やアクチュエータの慣性の問題があり，非常に速い応答が求められる用途への利用は難しいが，人と接する際の柔かさなどには充分に利用できる程度に制御機器の性能は高くなっている．ソフトウェアによる場合の利点は，(1) 機構が比較的単純になりやすい，(2) パラメータの調節等が容易，(3) 様々な制御のモードを設けることができ状況に応じて制御法を切り替えることができる，等が挙げられる．本研究では，これらの利点をとり，主に張力センサとソフトウェアによる柔軟性の調節を行うことにした．

#### ソフトウェアによるばねの挙動の実現

式 (4.33)$\boldsymbol{K}_m$ を調節するためには，各筋のバネ定数 $k_i$ を調節できる必要がある．張力センサの情報を利用して $k_i$ を変更するための制御はどのようにするのがよいか検討する．ばねは，(1) 力が加わっていない時は自然長 ($l$ とする) になる，(2) 力 ($F$ とする) が加わると長さが $\delta l$ だけ変わり $k\delta l$($k$ はバネ定数) の力を $F$ と反対方向に発生する，(3)$F = k\delta l$ となる $\delta l$ の位置が釣り合いの位置になる，といった特徴がある．

この挙動を再現するために， **Fig. 4.11** に示すようなブロック図により筋の制御を考えた．内側のループは張力制御を行っており (4.6.2節および **Fig. 4.19** 参照)，筋張力が目標値

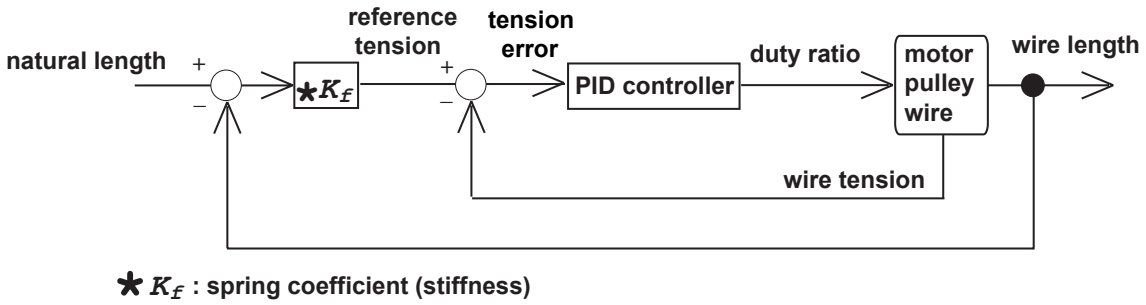



**Fig. 4.11** ソフトウェアによるばねモードのブロック図
Block diagram of spring mode by software

(reference tension ($T_r$ とする) ) になるように制御する．$T_r$ は，自然長 $l_0$ と現在のワイヤ長 $l$，およびバネ定数 $k_s$ によって次式により決定する．

$$T_r \quad = \quad k_s(l_0 - l) \tag{4.36}$$

筋張力制御のループは PD 制御である．ソフトウェアで動的に指定できるのは，バネ定数 $k_s$，自然長 $l_0$，それに筋張力制御の PD controller の部分のパラメータである力制御の P,D のゲインである．

ここで，張力センサを用いずに筋の長さを比例制御し，その比例ゲインを調節することにより，位置制御目標値が自然長に対応し，比例制御ゲインがバネ定数に対応するようできるのではないかという可能性を検討しておく．これはつまり，比例制御ゲインの調節により筋の剛性行列 $\boldsymbol{K}_m$ を調節し，式 (4.35) により脊椎の姿勢の剛性行列を調節できるのではないかということである．比例制御により筋長制御 (位置制御) を行う場合は **Fig. 4.12** のようなブロック図になる．図中にも書いてある通り，モータへの入力信号である PWM のデューティ比とモータの出力トルクは比例関係ではない．さらに，モータの出力トルクと筋張力も摩擦などの影響で必ずしも比例するとは限らない．柔軟性調節を式 (4.35) により剛性行列を利用して行うために，筋張力と自然長からのずれの関係が線形 (比例定数 $k$) になるようにするためには，張力センサの情報を利用したばねモードが有効であると言える．

### ソフトウェアばね制御の様子

5 節の脊椎を持つ人間型ロボット Cla(第 3.6節) の脊椎を駆動する筋を，全てソフトウェアばね制御により制御し，自然長を初期姿勢 (基本姿勢) の時の長さとしバネ定数を 4 種類に変えて，人間が触った硬さがどのように変わるかを確認した． **Fig. 4.13** に，全筋ソフトウェアばね制御の状態で人間が脊椎の姿勢を変えて手を離すと戻る実験を行う様子を示す．

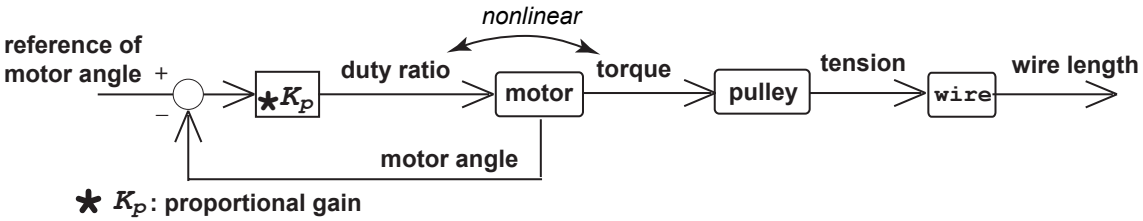



**Fig. 4.12** 比例制御による位置制御 (筋長制御) のブロック図
Block diagram of proportional position control (length control)

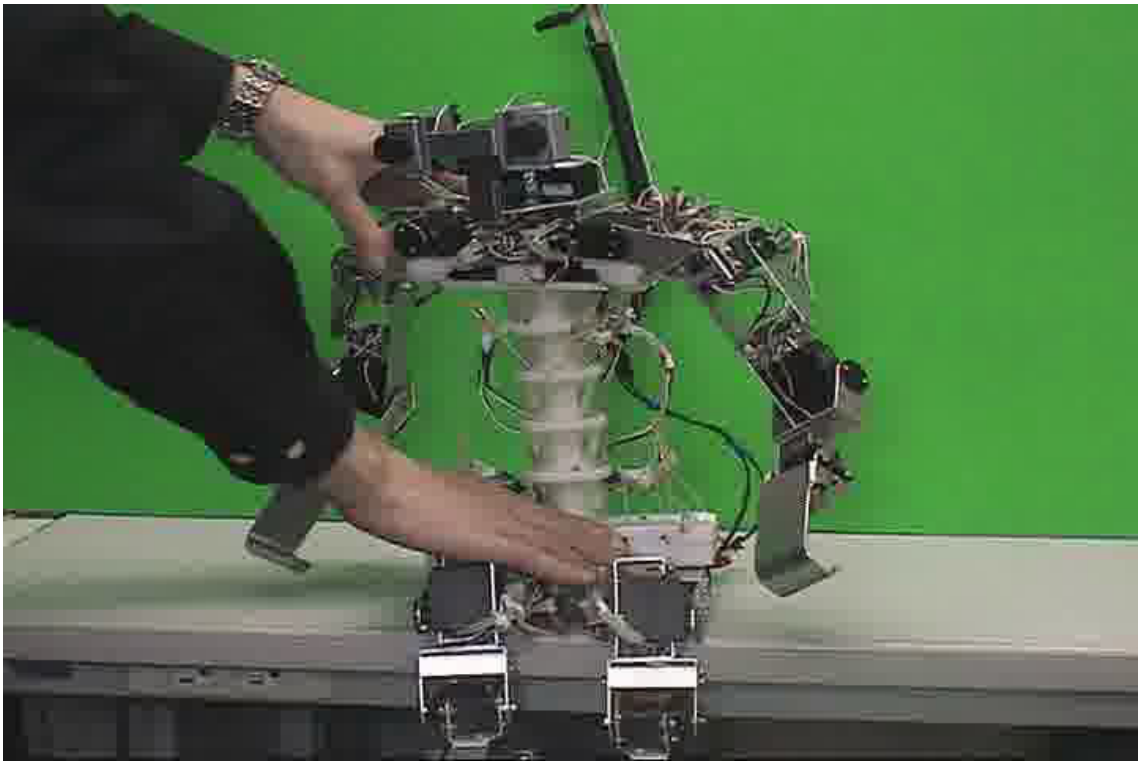


**Fig. 4.13** 全筋ソフトウェアばね制御の様子
All muscles of Cla are in software spring mode.

**Fig. 4.14 ～ Fig. 4.17** は，バネ定数を 4 種類に変えて同様の実験を行った際に，各筋の自然長からのずれ (Error) と張力 (Tension) がどのように変化しているかを示している．ここでは，バネ定数の単位は張力センサから得られる AD 変換の数値とモータエンコーダから得られるパルスカウントの数値の比である．バネ定数が大きいほど筋長の変化が小さくても張力の変化は大きくなる様子がわかる．

#### 椎間板の復元力

本研究で提案する復元力を持つ脊椎構造 (3.4.2節) にかかるに力は,

(1) 筋による変形力,
(2-1) 椎間板の弾性力,
(2-2) 椎間板の粘性力,
(3) ロボットの構造材にかかる重力等により脊椎構造に加わる力,
(4) ロボットの運動により生じる慣性力,
(5) 人間や環境からの外力,

がある．したがって，筋の剛性 $\boldsymbol{K}_m$ を用いて脊椎の剛性行列 $\boldsymbol{K}_j$ を制御しようとしても (式 (4.35))，脊椎構造全体のいわゆる剛性を正確に制御するためには，椎間板 (3.4.1節参照) の発生する力等を考慮しなければならない．

椎間板は，静力学だけを考慮しても非線形な弾性体であり，大変形とそれに伴う急激な反力と変位の変化もあり，さらにヒステリシスもある．時間要素を含めて考慮すれば粘性の影響も無視できない．粘性の無視できる線形なばね要素の組み込みを検討するという考えもあるが，本研究の基本姿勢はある程度の従来型の制御法での制御とセンサベーストな手法によるというものであり (第 4.1節)，厳密な意味での剛性調節まではここでは踏み込まない．

### 4.6 筋の制御のまとめ

ここまでに述べた脊椎の制御において言及した筋の制御方式が複数あった．本節ではこれらを列挙し，その方式・特徴・利用法などについて述べる．さらに，これらの制御法をまとめて統合的な制御方式が可能であるか検討する．

#### 4.6.1 筋長制御 (位置制御)

張力センサの情報を利用せず，プーリを駆動するモータのロータリエンコーダの値を制御目標とし， PID 制御を行う．したがって正確には筋長制御ではなく筋巻き取りプーリ回転角度

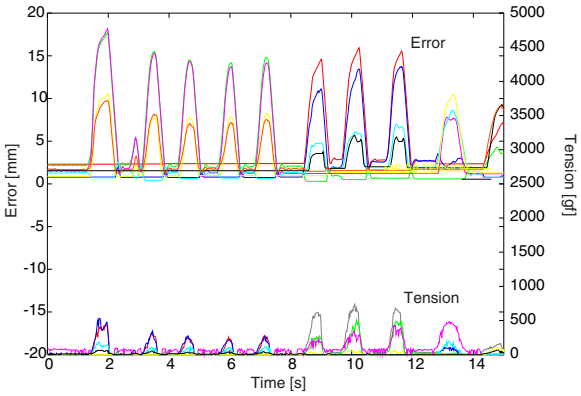



**Fig. 4.14** バネ定数 50
spring coeff. 50

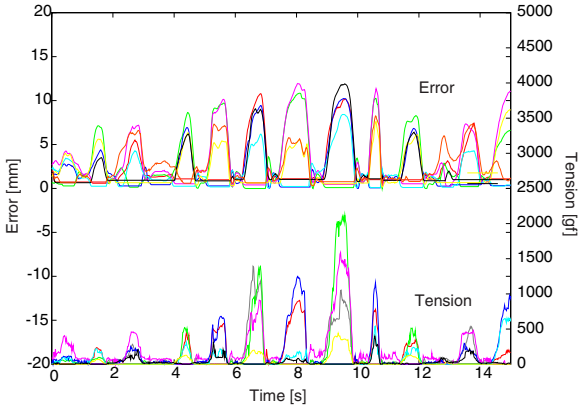



**Fig. 4.15** バネ定数 100
spring coeff. 100

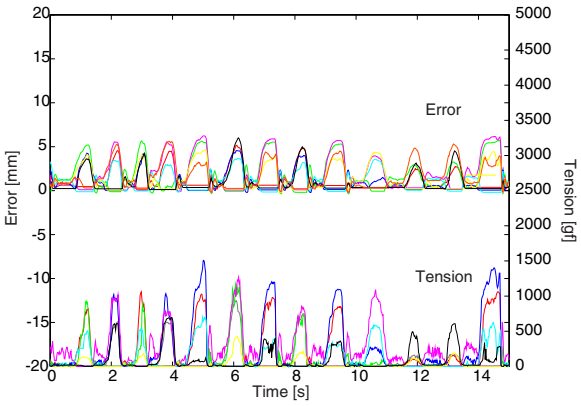



**Fig. 4.16** バネ定数 200
spring coeff. 200

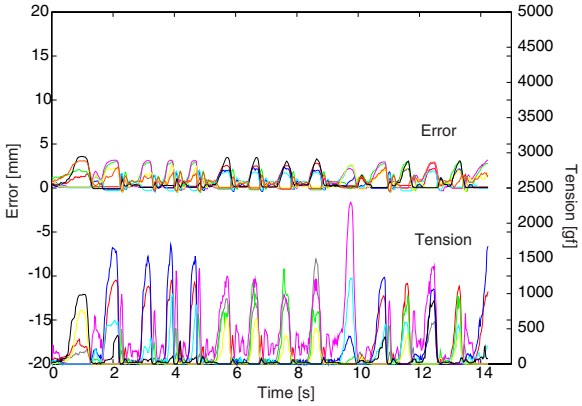



**Fig. 4.17** バネ定数 400
spring coeff. 400

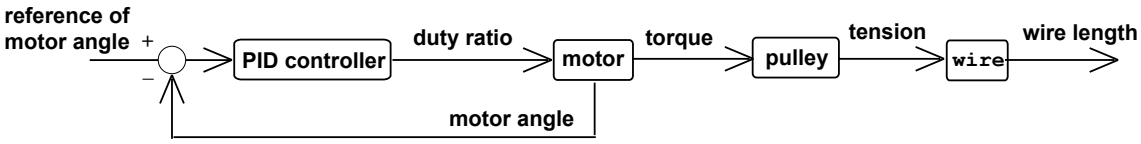



**Fig. 4.18** 筋長制御のブロック図
Block diagram of muscle-length control

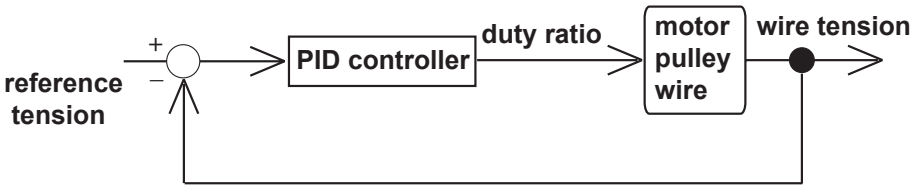



**Fig. 4.19** ソフトウェア張力制御のブロック図
Block diagram of tension control by software

制御である．プーリに巻かれた筋の平均半径を元に筋長を計算している．ソフトウェアにより動作中に変更可能なパラメータは， P・I・D の各ゲイン ($K_{P_l}$・$K_{I_l}$・$K_{D_l}$) である．

$$o = -K_{P_l}(l - l_r) - K_{I_l}\int (l - l_r)dt - K_{D_l}\frac{dl}{dt} \tag{4.37}$$

によりモータへの出力信号 (PWM) のデューティ比 $o$ を決定する．ただし， $l$ は筋長 (モータのエンコーダの値)， $l_r$ は目標筋長である．筋長制御のブロック図を **Fig. 4.18** に示す．

Cla(第 3.6節)・腱太 (第 3.7節) では絶対角度のとれないエンコーダを利用しているので，初期状態のキャリブレーションが必要である．第 4.4節に述べたモデルから筋の長さを求め (第 4.4節)，その長さになるように筋巻き取り量を制御することにより，ある程度の姿勢制御が可能である．筋同士・筋とボディ構造材の物理的な干渉の問題に関しては第 4.7節に述べる．

### 4.6.2 筋張力制御

**Fig. 4.19** にソフトウェアによる張力制御のブロック図を示す．目標張力 $T_r$ を入力として，モータの出力 (PWM のデューティ比 $o$) を直接調節することで，ワイヤ張力を一定に制御する．制御は PD 制御で，張力制御の P・D の各ゲイン ($K_{P_t}$・$K_{D_t}$) を調節可能である．また，振動防止のため，目標張力からある程度 ($\pm\, T_t$) の範囲は不感帯を設けてある． $K_{P_t}, K_{D_t}$, $T_t$ は全てソフトウェアで動的に調節できる． $o$ を計算する式は次のようになる．

$$o = \begin{cases} -K_{P_t}(T - T_r) - K_{D_t}\dfrac{dT}{dt} & (\ |\, T - T_r \,| \geq\ T_t\ ) \\ 0 & (\ |\, T - T_r \,| <\ T_t\ ) \end{cases} \tag{4.38}$$

ただし，$T$ は張力センサの値，$T_r$ は目標筋張力である．

張力制御により，周囲になじむ，あるいは周囲の環境に危害を加えにくいという特徴を持たせる事ができる．また，全筋を張力制御し，全筋の目標張力を同じ張力に設定した場合，筋の張力はつりあって体幹の動きには影響を与えなくなる．つまり，シリコンゴムから成る椎間板のばね成分で初期姿勢に戻ろうとする力が生じているだけの状態になる．全筋の目標張力を同じにするのではなく，その配分を調節する事により，外力の内状態であれば直立以外の姿勢に保つ事ができるはずである．ある姿勢のときの椎間板により生じる内力を求める事ができれば，それにつりあうように筋張力分配を決める事により，張力制御による姿勢制御が可能になるはずである．しかし，椎間板に生じる内力と変形量の関係の定式化あるいはモデル化は困難であると予想され (4.5.2節参照)，椎間板に直接圧力センサなどを組み入れて内力を求めるという方法が妥当な案として考えられる．

### 4.6.3 ソフトウェアばね制御

制御の方法は 4.5.2節に詳しく述べた．制御ブロック図は **Fig. 4.11** に示す．筋の剛性 (ばね定数 $k_i$) をソフトウェア的に調節することで，脊椎の柔軟性を調節する．モータの PWM のデューティ比 $o$ を決める式は，

$$\left.\begin{array}{rcll} o & = & \left\{\begin{array}{ll} -K_{P_t}(T-T_r)-K_{D_t}\dfrac{dT}{dt} & (\ |\ T-T_r\ |\ \geq\ T_t\ ) \\ 0 & (\ |\ T-T_r\ |\ <\ T_t\ ) \end{array}\right. & \\ T_r & = & k_s(l_0-l) & \end{array}\right\} \tag{4.39}$$

のようになる． $k_s$ はバネ定数である．式 (4.38) における $T_r$ を式 (4.36) により決定する形である．ソフトウェアで動作中に変更できるパラメータは，張力制御のゲイン，不感帯幅，バネ定数，バネ自然長 ($K_{P_t}$， $K_{D_t}$， $T_t$， $k_s$， $l_0$) である．

### 4.6.4 張力制限付き筋長制御

4.6.1節の制御に張力制限を設けたモードである． 4.6.1節の P・I・D の各ゲインと，限界張力 $T_l$ を設定する事ができる．張力 $T$ が $T_l$ を超えると，モータ出力 (PWM のデューティ比 $o$) を 0 にする． $o$ を決める式を以下に示す．

$$o \ = \ \left\{\begin{array}{ll} -K_{P_l}(l-l_r)-K_{I_l}\displaystyle\int(l-l_r)dt-K_{D_l}\frac{dl}{dt} & (T<T_l) \\ 0 & (T\geq T_l) \end{array}\right. \tag{4.40}$$

このモードでは次に示す利点が考えられる．

**安全性 (フェイルセイフ)** 大きな外力が加わった時に受動的になる．衝突時などの安全性という点と，試験的な実験時に誤って極端な姿勢になろうとするといった事を防止する事が可能である．

**姿勢の教示** 体幹部の変形自由度が多いため，第 4.4節に示したようなコンピュータ上のモデルを利用した数値的な姿勢情報では，直感的にわかりにくい．実機を直接動かしての指示が可能であれば，姿勢の教示が容易になる．

張力制限付き位置制御モードにおいて，限界張力を比較的小さい値に設定し全ての筋の目標長さを短くすることにより，各ワイヤが限界値の張力でつりあった状態になる．その状態で実機を直接動かそうとすると，たるんだ筋は張力が小さくなり目標長さに近づこうと筋長が短くなり，反対側の引かれた筋は限界張力を超えモータ出力を停止する．モータ出力が停止すると，ある程度以上の力をかけていればモータのバックドライバビリティにより受動的に筋を繰り出す事可能であるため，人間が直接触って姿勢を調節する事ができる．

外力 (人間の力) と重力による力を区別する事はできず，ある姿勢を保持することができるのは，筋にかかる力がモータが受動的に動き始める (バックドライブされる) 力より小さい場合のみである．

**初期姿勢のキャリブレーション (エンコーダのゼロ点調節)** 前項の姿勢の教示と同じように，限界張力を筋がピンと張る程度の張力に設定し，筋の目標長さを短くする事により，全ての筋が同程度の大きさの張力の状態になる．この状態で初期姿勢にし，その時のエンコーダの値をゼロ点に設定する事により，筋駆動形式でしばしば問題となるエンコーダのゼロ点調節が可能である． 4.6.2節の張力制御モードでも可能だが，張力制御モードより張力制限付き位置制御モードのほうが人間が直接動かす際の反力がある程度あり (姿勢を保とうとする性質が強い)，初期姿勢を作った時にそれを保持しやすい．

### 4.6.5 複数の制御モード

これらをまとめて一つの統合的な制御方式は可能だろうか．ここまでに述べた制御方式は， 4.6.1節筋長制御 (L 制御と呼ぶ)， 4.6.2節筋張力制御 (T 制御と呼ぶ)， 4.6.3節ソフトウェアばね制御 (S 制御と呼ぶ)， 4.6.4節張力制限付き筋長制御 (LL 制御と呼ぶ) の 4 種類があった．これらを表す式 (4.37)，式 (4.38)，式 (4.39)，式 (4.40) をまとめて一つの式にすると，

$$o = -\mathrm{f}\left(T_l - T\right)\left(K_{P_l}\left(l - l_r\right) + K_{I_l}\int\left(l - l_r\right)dt + K_{D_l}\frac{dl}{dt}\right)$$

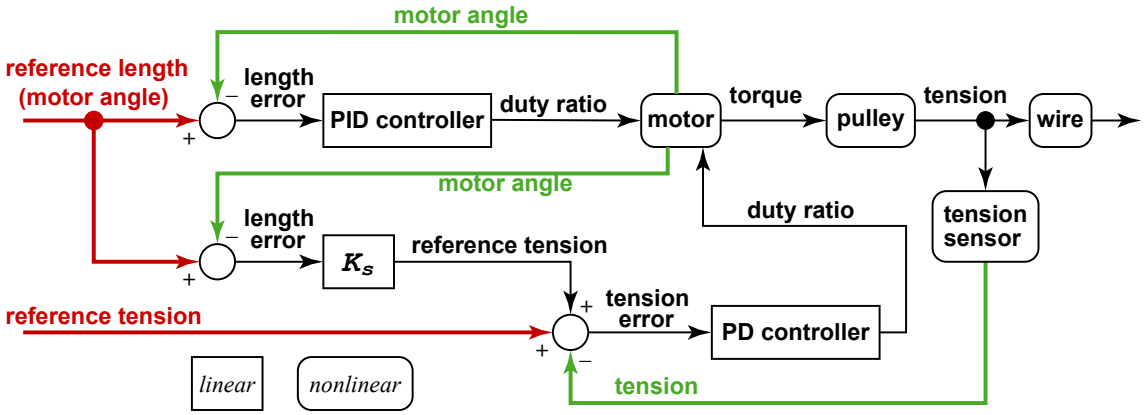



**Fig. 4.20** 全ての制御モードをまとめたブロック図
Block diagram of all control-modes

$$- \mathrm{f}\left( \left| T - (T_r + k_s\,(l_0 - l)) \right| - T_t \right) \left( K_{P_t}\,(T - (T_r + k_s\,(l_0 - l))) + K_{D_t} \frac{dT}{dt} \right) \tag{4.41}$$

のようになる．ただし，

$$\mathrm{f}\,(x) \equiv \begin{cases} 1 & (0 < x) \\ 0 & (x \leq 0) \end{cases} \tag{4.42}$$

である．ブロック図にすると **Fig. 4.20** にようになる． ソフトウェアで調節可能なパラメータは，$l_r$，$T_r$，$K_{P_l}$，$K_{I_l}$，$K_{D_l}$，$K_{P_t}$，$K_{D_t}$，$T_l$，$T_t$，$k_s$ である．各制御モードにおいて各パラメータを利用できるか，0 にしておかなければならないか，意味を持たないかに分類した ( Table 4.6 ).

式 (4.41) を統合された制御方式と呼ぶには式のシンプルさエレガントさに欠るように感じられる． L(LL),T,S という制御モードに分類したが，これらは冗長ではないだろうか．筋の制御系は制御すべき制御量が長さと張力の二種類あり，これらは独立である．ばねモード (S 制御) の場合この二種類の量ではなく，自然長とバネ定数という二種類の量を制御する．これは長さと張力という二種類と独立ではないが，式 (4.35) を利用して脊椎の姿勢の剛性行列を調節する時には，姿勢を長さにより制御しつつ筋の剛性 $\boldsymbol{K}_m$(式 (4.33)) を直接調節できるソフトウェアばね制御 (S 制御) は有効であり，やや複雑なこの制御法が必要であると考えられる．

また，式 (4.41) の形をコントローラのプログラムとして実装するのは適切ではない．力制御を含む制御はできるだけ制御周期を短くしなければならないが，式 (4.41) のような式を制御周期毎に計算するのは計算量が多くなる．コントローラのプロセッサは一般に非力であり浮動小数点演算器が搭載されていない場合も多く，計算ルーチンの最適化は非常に重要であ

Table 4.6 制御モード別各パラメータの利用
The use of parameters depending on control-mode

| パラメータ | 記号 | L 制御 | T 制御 | S 制御 | LL 制御 |
|---|---|---|---|---|---|
| 目標筋長 | $l_r$ | V | — | V | V |
| 目標筋張力 | $T_r$ | — | V | 0 | — |
| 筋長制御 P ゲイン | $K_{P_l}$ | V | 0 | 0 | V |
| 筋長制御 I ゲイン | $K_{I_l}$ | V | 0 | 0 | V |
| 筋長制御 D ゲイン | $K_{D_l}$ | V | 0 | 0 | V |
| 張力制御 P ゲイン | $K_{P_t}$ | 0 | V | V | 0 |
| 張力制御 D ゲイン | $K_{D_t}$ | 0 | V | V | 0 |
| 制限張力 | $T_l$ | — | — | — | V |
| 不感帯幅 | $T_t$ | — | V | V | — |
| バネ定数 | $k_s$ | — | 0 | V | — |

※記号の意味
- V　指定しなければならない.
- 0　0でなければならない.
- —　任意の値(意味を持たない).

り，しばしばアセンブリ言語レベルのプログラミングが行われるほどである．したがって，L,T,S,LL の各制御モード別に計算ルーチンを作り，システムの上位層から制御モードを切り替えられるようにするという構成が適切である．

## 4.7 筋の物理的な干渉

### 4.7.1 筋の物理的干渉の問題

第 4.4節に述べた幾何モデルは，式 (4.25) に基づいて脊椎の姿勢と筋の長さの関係を計算している．逆計算にヤコビアンや NN を利用する場合も式 (4.25) を利用して逆を求めている．式 (4.25) は，筋と筋，筋と構造材の物理的な干渉を無視している．しかし，実際は脊椎をねじればクロスして張ってある筋同士は物理的に干渉するし，前後・左右に屈曲すれば筋と構造材は干渉する．これはいわゆる干渉駆動 [14, 15] のように複数の筋の張力が協調して 1 本の筋では出せない力を出すというような種類の干渉ではなく，物理的に経路の妨害となり本来出せるはずの張力を出せなくする，あるいは本来出せるはずの力とは違った方向の力が発生するなどといった，ネガティブな意味での干渉である．本節 (第 4.7節) では，筋の干渉問題を解決するための方法を検討する．

解決方法として一つ考えられるのは，筋の干渉をモデル化し干渉まで考慮したモデルで姿勢 $\longleftrightarrow$ 筋長の計算を行うことである．しかし，4.4.1節に述べたように，干渉を考慮したモデルを構築するためには，筋の太さ・経路，椎間板の発生力・上半身から受ける力・外力・筋張力の間のバランス，などを計算できることが求められ，こうしたモデルを構築することは困難であると思われる．さらに，本研究では複雑な身体構成の機械を動かすための方針として，厳密ではなくある程度の幾何学的なモデルとセンサフィードバックによるのが適しているとの立場をとっている (第 4.1節)．

本研究では，干渉のモデルを構築するアプローチではなく，干渉の存在を想定しセンサを利用して干渉の問題を解決するというアプローチをとる．

### 4.7.2 張力センサを利用した制御

第 4.6節には，4 種類 (L,LL,T,S) の筋の制御モードに関して述べた．筋の干渉の問題を解決するために，こうした制御モードのいずれを用いるのが適切か，あるいは別の制御法がふさわしいかを検討する．式 (4.25) は直線距離で筋の長さを計算するので，筋が物理的に他の筋や構造材と干渉すると，その筋は式 (4.25) で求めた長さより長くなると思われる．式 (4.25) で求めた長さを目標筋長 $\boldsymbol{l}_r$(S 制御においては自然長) として L,LL,S の各制御モードで筋を制御すると，張力が大きい筋ほど $\boldsymbol{l}_r$ とのずれが大きくなると予想できる．

ここで，現在の姿勢 ($\boldsymbol{\theta}_0$ とする) の目標筋長 ($\boldsymbol{l}_{r0}$ とする) から，姿勢を $\boldsymbol{\theta}_1$ に変える場合を考える．式 (4.25) により計算される目標筋長 ($\boldsymbol{l}_{r1}$ とする) と $\boldsymbol{l}_{r0}$ の各要素を比較し，後者の方が小さくなるもの，すなわち筋長が短くなる筋と，逆に筋長が長くなる筋とで，別々の制御

モードを利用することを考える．姿勢を変えて筋長が短くなるということは，その筋の張力は姿勢変形力に貢献していると考えられ，逆に筋長が長くなる筋は前者ほど姿勢変形力に貢献しないという仮定をする事ができるだろう．そこで，仮に筋長が長くなる筋を “伸び筋”，短くなる筋を “縮み筋” と呼ぶと，縮み筋は L 制御により式 (4.25) で計算された通りの値に制御し，伸び筋は張力の制限がかかるような制御モード (LL,T,S のいずれかの制御) により制御することで，伸び筋群の $\boldsymbol{l}_{r1}$ とのずれにより干渉する筋の力を逃がすことができるのではないだろうか．

以上の考察に基づき，

1. 全筋を L 制御
2. 全筋を LL 制御
3. 全筋を S 制御 (バネ定数中)
4. 全筋を S 制御 (バネ定数大)
5. 縮み筋は L 制御，伸び筋は LL 制御
6. 縮み筋は L 制御，伸び筋は T 制御
7. 縮み筋は L 制御，伸び筋は S 制御 (バネ定数中)
8. 縮み筋は L 制御，伸び筋は S 制御 (バネ定数大)

の 7 種類の方法により筋を制御し，$\boldsymbol{\theta}_1$ と実際の姿勢とのずれを計測する実験を行った．実験は，5 節の脊椎を持つ人間型ロボット Cla の脊椎 (第 3.6節 参照) を用いて行った．両腕を真上に上げた状態で，脊椎が pitch 軸回りに $0^\circ \Rightarrow -10^\circ \Rightarrow -20^\circ \Rightarrow -30^\circ \Rightarrow -20^\circ \Rightarrow -10^\circ \Rightarrow 0^\circ \Rightarrow 10^\circ \Rightarrow 20^\circ \Rightarrow 30^\circ \Rightarrow 20^\circ \Rightarrow 10^\circ \Rightarrow 0^\circ$，次に roll 軸回りに $0^\circ \Rightarrow -10^\circ \Rightarrow -20^\circ \Rightarrow -30^\circ \Rightarrow -20^\circ \Rightarrow -10^\circ \Rightarrow 0^\circ \Rightarrow 10^\circ \Rightarrow 20^\circ \Rightarrow 30^\circ \Rightarrow 20^\circ \Rightarrow 10^\circ \Rightarrow 0^\circ$ の順になるように目標姿勢を与え，実際の姿勢を計測した．実際の姿勢の計測には，腰と肩にそれぞれ組み込まれている 3 軸加速度センサ (3.6.2節参照) を利用した．肩と腰に取り付けた 3 軸加速度センサから重力の方向を計測し，式 (3.4) により肩と腰の差分を計算した．ただし，動作中はその動作で生じる加速度がセンサ情報に含まれるので，上記シークェンスの各姿勢を数秒間ずつとるようにした．各モードの場合の 3 軸加速度センサにより計測した roll,pitch 各軸回りの姿勢の変化を，**Fig. 4.21** 〜 **Fig. 4.25** に示す．ただし，上記項目のうち L 制御と T 制御の組み合わせ (6.) の時は，脊椎が座屈してしまったため，データを載せていない．

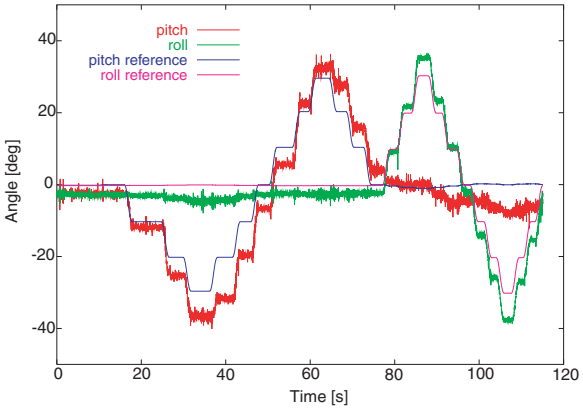



**Fig. 4.21** 全筋 L 制御
L-mode

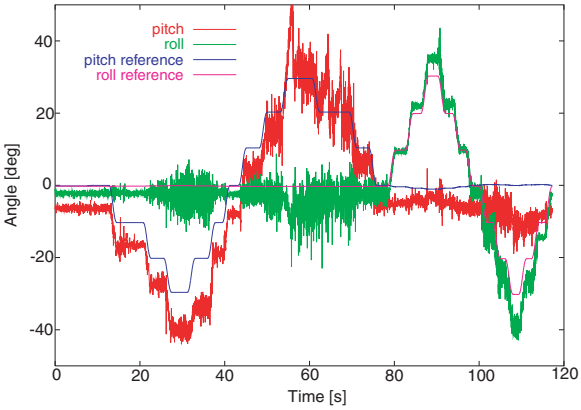



**Fig. 4.22** 全筋 LL 制御
LL-mode

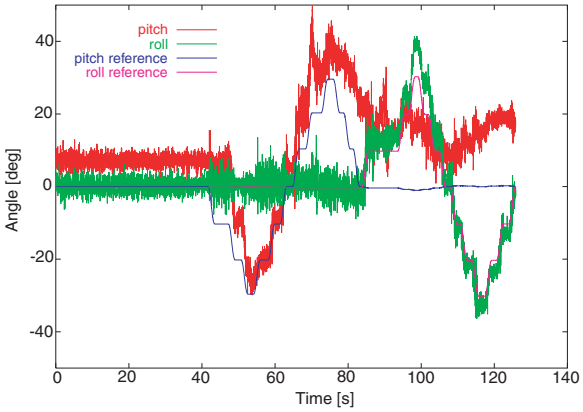



**Fig. 4.23** 全筋 S 制御 (バネ定数中)
S-mode (middle stiffness)

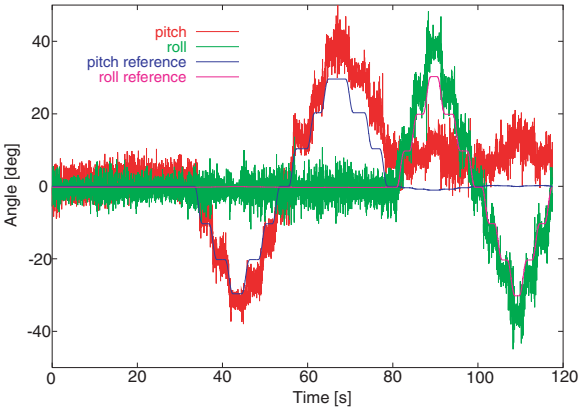



**Fig. 4.24** 全筋 S 制御 (バネ定数大)
S-mode (high stiffness)

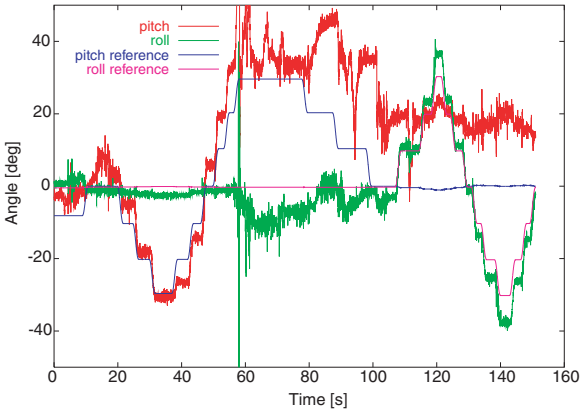



**Fig. 4.25** 組み合わせ (L と LL)
L-mode and LL-mode

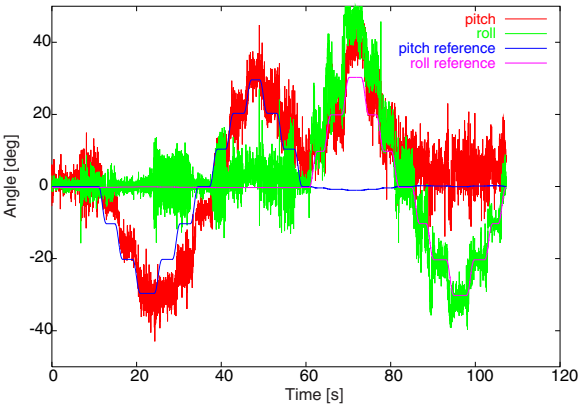



**Fig. 4.26** L と S(バネ定数中)
L and S(middle stiffness)

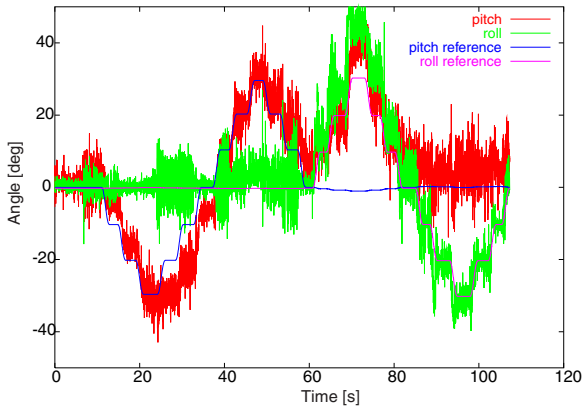



**Fig. 4.27** L と S(バネ定数大)
L and S(high stiffness)

また，同じ動作の時の，各筋の張力を張力センサにより計測した．各筋の張力の様子を **Fig. 4.28** ～ **Fig. 4.32** に示す．

さらに，同じ動作の時の，各筋を駆動するモータの電流をモータドライバ基板の電流センサにより計測 (直列の 0.1[Ω] の抵抗の電圧変化を計測) した．各モータの電流の様子を **Fig. 4.35** ～ **Fig. 4.39** に示す．

**Fig. 4.21** ～ **Fig. 4.25** の中の pitch–reference と roll–reference はモデル上での姿勢角を表し， pitch と roll は肩と腰に取り付けられた 3 軸加速度センサから計測した重力方向を表すベクトルの (肩と腰の) なす角により求めた姿勢角を示す．モデル上での姿勢角を次のステップに進める際はキーボード入力をトリガにしたので，各姿勢間の時間間隔は一定ではない．また， pitch 軸回りは roll 軸回りに比べて筋が脊椎の回転中心近くを通っているためか，姿勢がふれやすい様子がわかる．また，一部のデータは，脊椎が姿勢を保持しきれなくなって大きく曲がってしまい，手で支えられた部分がある ( **Fig. 4.25** 等).

**Fig. 4.21** ～ **Fig. 4.25** のグラフの pitch–reference と roll–reference は，モデル上での姿勢角を表す．モデル上での姿勢角と，肩と腰の 3 軸加速度センサの差分により計測した実際の姿勢角の間の誤差の，各姿勢における平均値を求めた． Table 4.7 に示す．ただし， L 制御と T 制御の組み合わせ (122ページ 6.) の時は，脊椎が座屈してしまったため，データがない．また，表中で — となっている部分は，姿勢を保持できずに大きく崩れてしまい実験者が手で支えたためデータが取れなかった部分である．

各制御モードにおける各筋の張力の平均値・最大値と，筋を駆動する各モータの消費電流の平均値・最大値を付録 Aに掲載した．

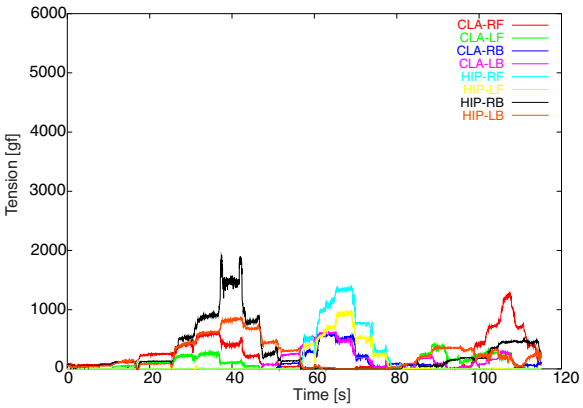



**Fig. 4.28** 全筋 L 制御
L-mode

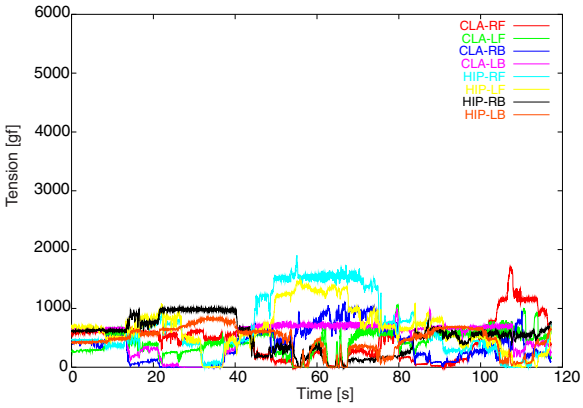



**Fig. 4.29** 全筋 LL 制御
LL-mode

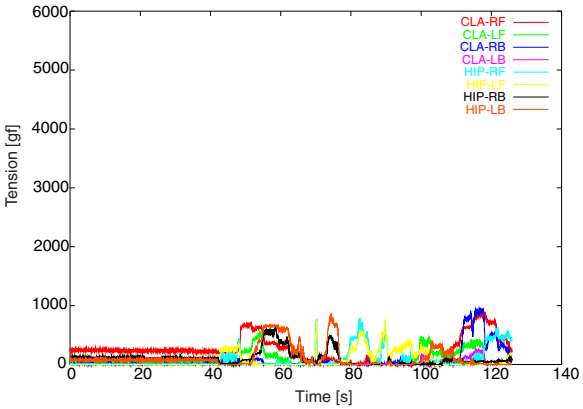



**Fig. 4.30** 全筋 S 制御 (バネ定数中)
S-mode (middle stiffness)

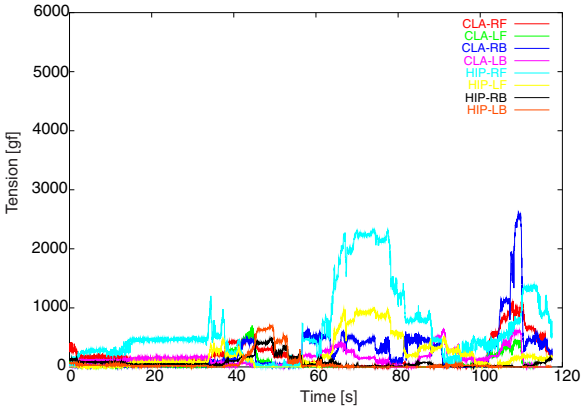



**Fig. 4.31** 全筋 S 制御 (バネ定数大)
S-mode (high stiffness)

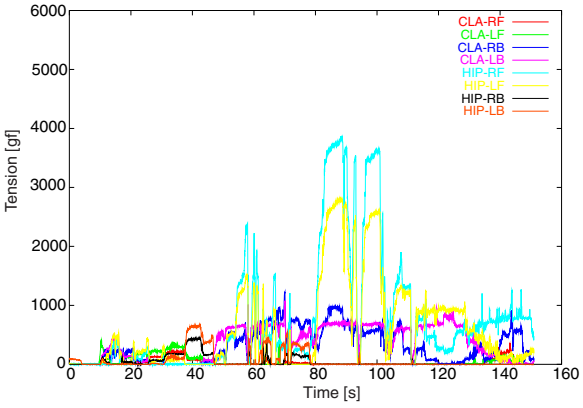



**Fig. 4.32** 組み合わせ (L と LL)
L-mode and LL-mode

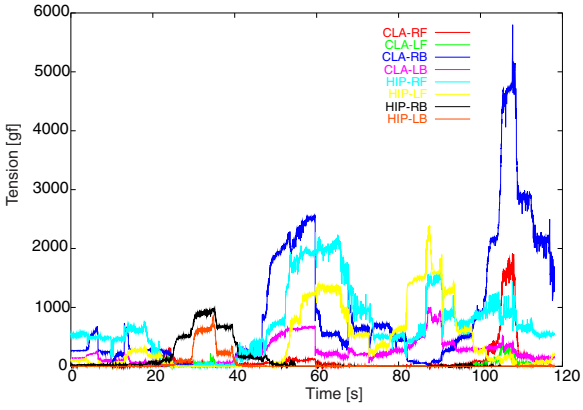



**Fig. 4.33** L と S(バネ定数中)
L and S(middle stiffness)

Table 4.7　各制御モードにおける姿勢角の平均値 (3 軸加速度センサによる)
The average of the posture (angle) at each control-mode

| 姿勢 | 全筋 L 制御 | 全筋 LL 制御 | 全筋 S 制御 (バネ定数大) | 全筋 S 制御 (バネ定数中) | L 制御 + LL 制御 | L 制御 + S 制御 (バネ定数中) | L 制御 + S 制御 (バネ定数大) |
|---|---|---|---|---|---|---|---|
| pitch | | | | | | | |
| -10 | 0.65 | 2.24 | 1.50 | 8.37 | 0.26 | 1.20 | 3.52 |
| -20 | 2.83 | 1.01 | 0.91 | 3.46 | 1.27 | 0.86 | 2.44 |
| -30 | 4.17 | 3.65 | 3.03 | 1.76 | 3.57 | 0.10 | 1.89 |
| -20 | 9.33 | 9.50 | 7.75 | 7.31 | 7.95 | 8.98 | 11.36 |
| -10 | 7.17 | 2.68 | 6.27 | 6.37 | 2.80 | 13.54 | 13.34 |
| 0 | 4.20 | 2.89 | 3.72 | 1.70 | 1.22 | 10.58 | 8.67 |
| 10 | 2.04 | 0.14 | 2.09 | 3.38 | 6.35 | 3.33 | 4.04 |
| 20 | 4.98 | — | 3.27 | 6.07 | — | 0.67 | 0.74 |
| 30 | 4.95 | — | 6.52 | — | — | 6.23 | 0.57 |
| 20 | 9.79 | — | 9.34 | — | — | 5.29 | 2.87 |
| 10 | 8.29 | — | 9.65 | — | — | 7.15 | 4.26 |
| 0 | 6.42 | — | 7.08 | — | — | 6.71 | 5.30 |
| roll | | | | | | | |
| 10 | 1.81 | 3.07 | 2.14 | 3.53 | 1.60 | 3.93 | 2.13 |
| 20 | 4.41 | 6.38 | 5.84 | 5.88 | 4.28 | 7.15 | 6.14 |
| 30 | 8.01 | 8.69 | 9.92 | 10.28 | 7.39 | 18.48 | 14.45 |
| 20 | 6.04 | 5.97 | 6.88 | 9.59 | 4.80 | 4.11 | 11.71 |
| 10 | 2.97 | 3.39 | 3.71 | 4.99 | 2.45 | 4.30 | 5.91 |
| 0 | 0.70 | 1.06 | 0.77 | 1.98 | 0.54 | 1.24 | 1.95 |
| -10 | 1.34 | 1.39 | 1.70 | 0.03 | 0.96 | 1.74 | 0.59 |
| -20 | 3.20 | 3.17 | 3.07 | 0.28 | 1.29 | 3.93 | 1.54 |
| -30 | 4.97 | 5.42 | 4.11 | 1.56 | 6.69 | 5.15 | 1.70 |
| -20 | 4.32 | 4.11 | 4.31 | 2.64 | 0.04 | 2.72 | 1.20 |
| -10 | 2.69 | 2.29 | 3.47 | 1.91 | 2.18 | 1.46 | 1.21 |

※単位： degree

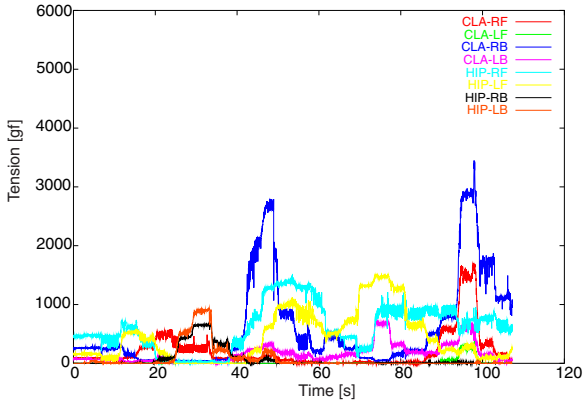



**Fig. 4.34** L と S(バネ定数大)
L and S(high stiffness)

### 4.7.3 制御モードの組合わせによる筋の物理的干渉問題の解決

以上の結果をまとめると Table 4.8 に示すようになる．モデルとの誤差の平均は， Table 4.7 から誤差の平均を求めた．最大姿勢保持角は，モデル上の姿勢を変更していったときに，どの姿勢まではその制御モードで保持できたかの限界姿勢角度を示す．今回は， pitch,roll に $\pm 30^\circ$ の範囲内だったので，その範囲ではすべて姿勢が保持できた場合は (30 以上) と示した．最大姿勢保持角の欄で — となっている部分 (7. と 8.) は， roll 軸回りの回転を行ったところ， pitch 軸回りの姿勢が不安定になり，前か後に倒れてしまった部分である．平均張力は，付録 Aに掲載した表 ( Table A.1 〜 Table A.7 ) より計算して求めた．平均電流も，付録 Aに掲載した表 ( Table A.15 〜 Table A.21 ) より計算して求めた．

まず，伸び筋を L 制御・縮み筋を T 制御による場合 (6.) は，姿勢を安定に保つことができなかったので，姿勢制御の方法としては考慮の対象から除外する．伸び筋は干渉する可能性が大きいので何らかの形でモデル上の筋長より長くする必要があると思われるが，そこを T 制御 (筋張力制御) によるといくらでも伸びることができるため，脊椎が重力に負けて崩れてしまうときにも伸び続けてしまう．そのため伸び筋を T 制御によるのは適していないと考えられる．

次に，伸び筋を L 制御・縮み筋を S 制御似寄る場合 (7. と 8.)，残りの制御モードによる場合と比較して，モデルとの誤差の平均や平均消費電流が大きくなる傾向がある． S 制御 (ソフトウェア疑似ばね制御) は，比較的細かくモータへの出力を調節するので，常に電流が流れてしまいがちな傾向がある．これは，制御周期が遅い (1[ms]) ことも原因ではないかと考えられるが，浮動小数点演算器を持たない組み込みプロセッサを用いているため，これ以上の制御周期で制御を行うのは通信などのルーチンが不調になる可能性もありやや難しい．

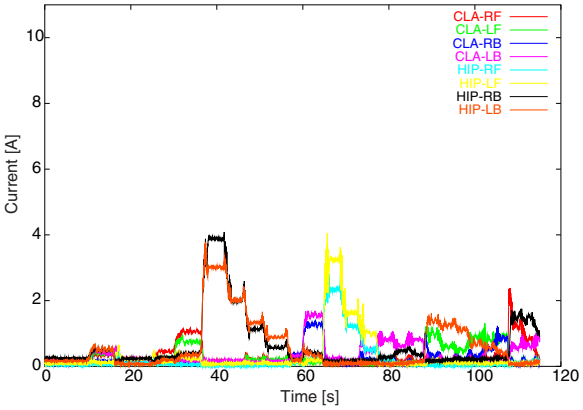



**Fig. 4.35** 全筋 L 制御
L-mode

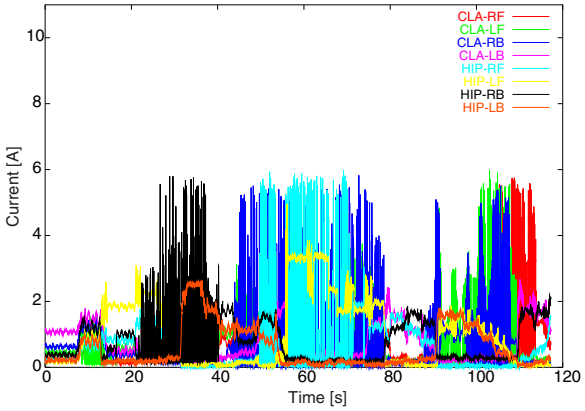



**Fig. 4.36** 全筋 LL 制御
LL-mode

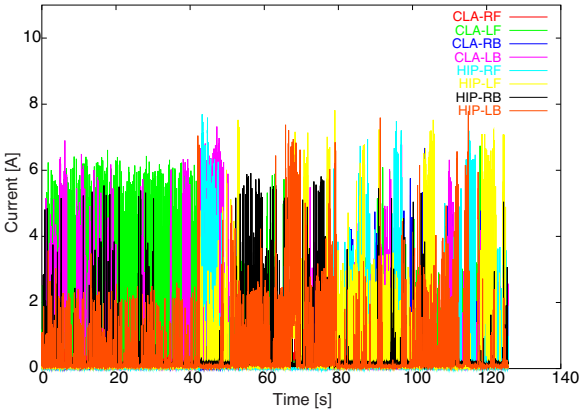



**Fig. 4.37** 全筋 S 制御 (バネ定数中)
S-mode (middle stiffness)

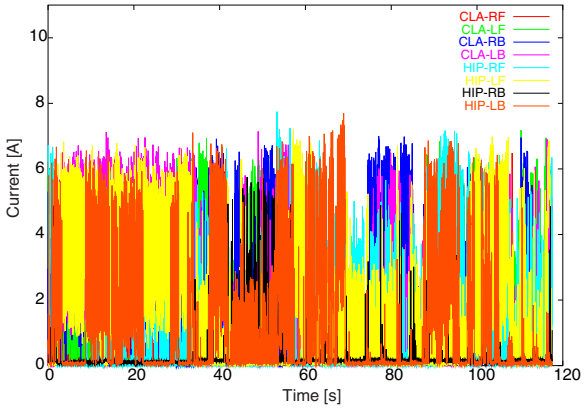



**Fig. 4.38** 全筋 S 制御 (バネ定数大)
S-mode (high stiffness)

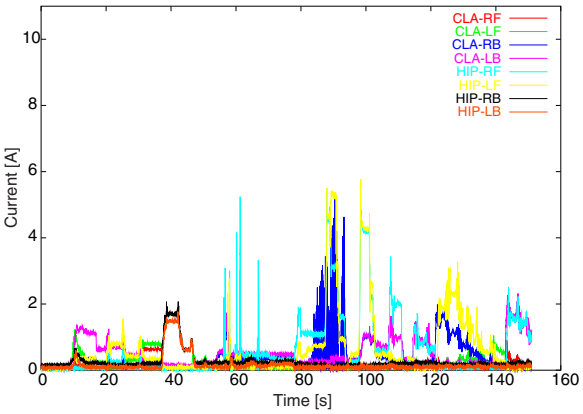



**Fig. 4.39** 組み合わせ (L と LL)
L-mode and LL-mode

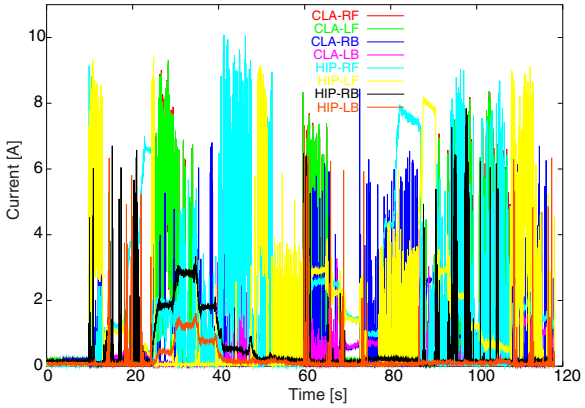



**Fig. 4.40** L と S(バネ定数中)
L and S(middle stiffness)

Table 4.8　筋の物理的干渉問題解決のための実験の結果
The result of the experiment for the interfer-problem of muscles

| 制御モード | モデルとの誤差の平均 | | 最大姿勢保持角 | | 平均張力 | | 平均電流 | |
|---|---|---|---|---|---|---|---|---|
| | pitch $\theta$ deg | roll $\phi$ deg | pitch $\theta$ deg | roll $\phi$ deg | pitch $T$ kgf | roll $T$ kgf | pitch $I$ A | roll $I$ A |
| 1. 全筋 L 制御 | 4.73 | 3.68 | (30 以上) | (30 以上) | 0.20 | 0.13 | 0.41 | 0.33 |
| 2. 全筋 LL 制御 | 2.85 | 2.93 | 10 | (30 以上) | 0.48 | 0.47 | 0.59 | 0.64 |
| 3. 全筋 S 制御 (バネ定数中) | 4.83 | 4.06 | 20 | (30 以上) | 0.13 | 0.13 | 0.49 | 0.42 |
| 4. 全筋 S 制御 (バネ定数大) | 3.86 | 4.17 | (30 以上) | (30 以上) | 0.23 | 0.25 | 0.98 | 0.79 |
| 5. 縮み筋 L, 伸び筋 LL | 3.66 | 4.09 | 10 | (30 以上) | 0.09 | 0.24 | 0.25 | 0.38 |
| 6. 縮み筋 L, 伸び筋 T ※ 1 | — | — | 0 | 0 | — | — | — | — |
| 7. 縮み筋 L, 伸び筋 S (バネ定数中) | 5.88 | 4.93 | (30 以上) | (30 以上) ※ 2 | 0.26 | 0.41 | 0.86 | 0.99 |
| 8. 縮み筋 L, 伸び筋 S (バネ定数大) | 6.87 | 4.47 | (30 以上) | (30 以上) ※ 2 | 0.25 | 0.33 | 1.29 | 1.04 |

※ 1： 6. は，座屈してデータ取れず.

※ 2：ただし，姿勢角度が大きくなると誤差も大きくなる.

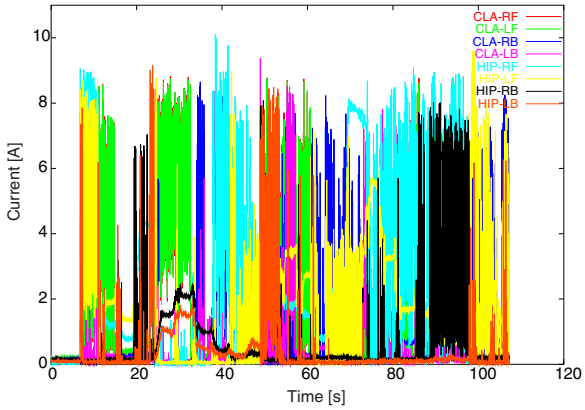



**Fig. 4.41**　L と S(バネ定数大)
L and S(high stiffness)

残る 5 種類を比較すると，モデルとの平均誤差の点では全筋 LL 制御 (2.) が最も優れている．しかし， pitch 軸回りの動作に関しては，全筋 LL 制御 (2.)，全筋 S 制御 (バネ定数中) (3.)，縮み筋を L 制御・伸び筋を LL 制御 (5.) の三種類はあまり大きな姿勢変化になると重力に逆らって姿勢を保持することができない．（これらの実験は Cla の長い両腕を真上に上げて行ったので，姿勢変化に対応した重力による負荷が大きい．）さらに，平均消費電流を比較すると， pitch 軸回りは L 制御 +LL 制御 (5.) が最も良く， roll 軸回りは L 制御 +LL 制御 (5. と全筋 L 制御 (1.) が優れている． roll 軸回りの運動時は， Cla の筋配置 ( **Fig. 3.31** 参照) からして，干渉が起きないのではないかと予想できる．そのため， roll 軸回りの運動に限れば全筋 L 制御 (1.) が最も良い結果になったのだと考えられる．

平均張力が最も小さいのは，全筋 S 制御の二つ (3. と 4.) である．しかし，これらは平均消費電流は比較的大きい． S 制御の振動しやすさから来るものであろうと考えられる．また姿勢の誤差は小さくはない．

これらのことから考えると，状況に応じて制御モードを切り替えて利用するのがいいと考えられる．以下に，制御モードをどのように使い分けるべきかを示す．

**干渉が少ない時、負荷が大きい時**　roll 軸回りのみのの動作や，基本姿勢付近の姿勢など，干渉が無い，あるいは少ないと予想できるときは，全筋 L 制御を利用する．

**負荷が大きくない時**　両手を下げた状態での動作や，それほど変形量が大きくない姿勢など，負荷が大きくないことが予想できるような時は，モデルとの誤差が最も少ない全筋 LL 制御による．

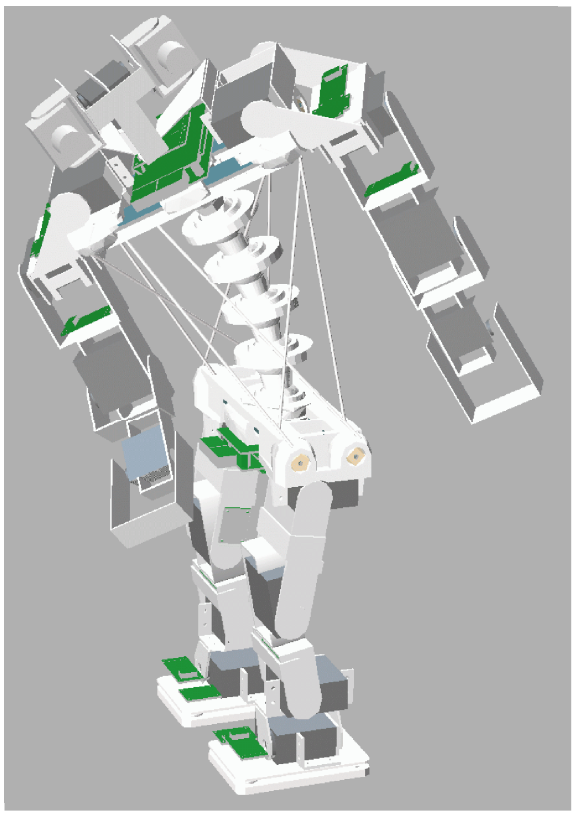


**Fig. 4.42** Cla の幾何モデル上での姿勢
A posture of Cla on the geometric model

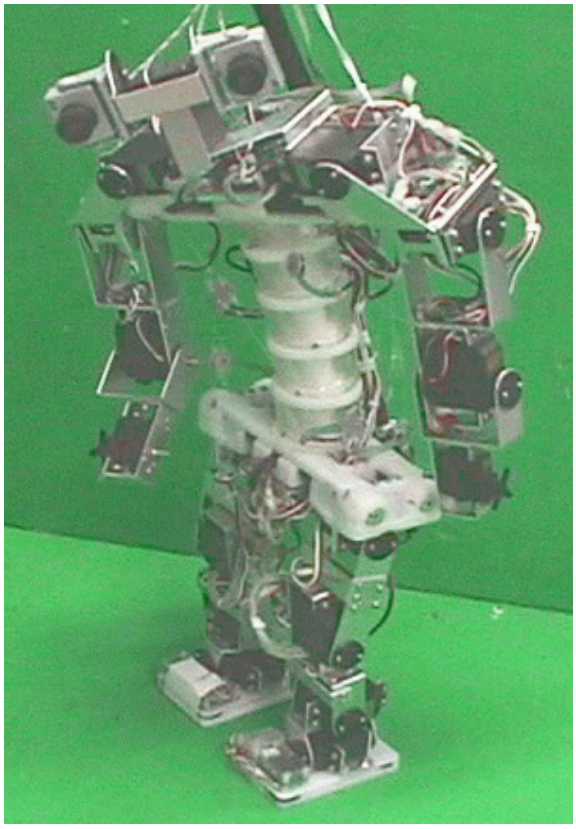


**Fig. 4.43** 制御された実機の Cla の姿勢
The controlled posture of real Cla

**消費電流を少なくしたい時** 張力や電流をあまり大きくしない動作を行うときで，脊椎の幾何学的姿勢の絶対角度がそれほど重要ではないときは，縮み筋は L 制御・伸び筋は LL 制御による.

### 4.7.4 実際の脊椎の制御

5 節の脊椎を持つ人間型ロボット Cla(第 3.6節) の脊椎を roll,pitch,yaw で与え，姿勢を制御した様子を **Fig. 4.42** ， **Fig. 4.43** に示す.

## 4.8 センサベーストな制御法のトライアル

### 4.8.1 概要

人間のように多自由度で複雑な身体構造を持つと，ロボットの制御法はモデルベーストなアプローチからセンサベーストな方法論へ移行してゆくだろうと指摘した (第 4.1節)．本研究では，従来のロボットが持っていた幾何的なモデルや動力学モデルの構築自体が非常に困難であるような，人間に近い身体構造を持つロボットのプロトタイプを製作し，幾何モデル・動力学モデルを使わずセンサベーストな制御のトライアルを行った．

具体的には，人間の脊椎と同じ数の節を持つ実物大医学用の脊椎のモデルと，空気圧の人工筋 [75, 76]36 本を用いて，人間の脊柱とそれを駆動する筋肉の配置を模倣した脊柱型ロボット “BeBe” ( **Fig. 4.44** ) を製作した [38]．そして，頭部に取り付けたカメラからの視覚情報を利用するアドホックな制御則に基づき，物体のトラッキング動作を行う実験を行った [39]．

### 4.8.2 脊柱型ロボット BeBe の設計・製作

#### 設計方針

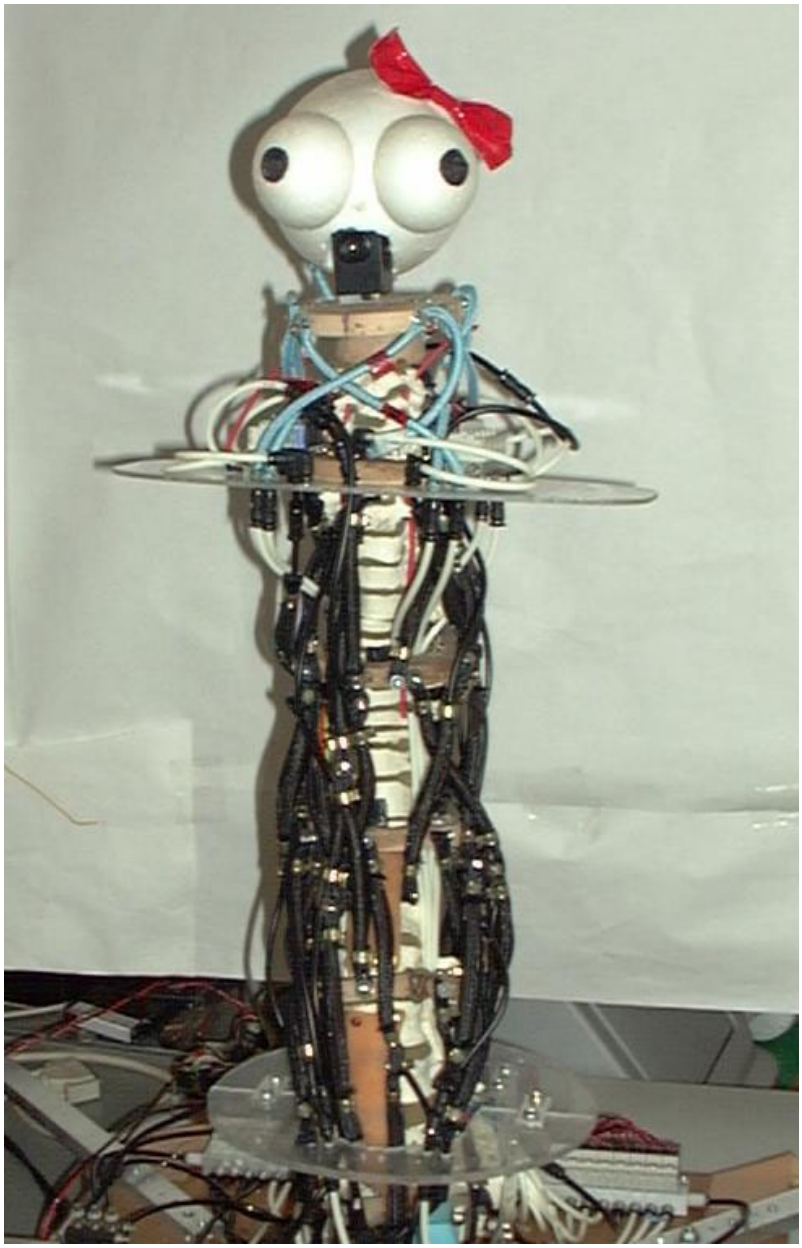
多自由度で複雑な身体構造として，人間の脊椎構造を模倣するという設計方針をとった．人間の筋肉に相当する働きが可能なアクチュエータの選択や，人間の背骨の形状や筋肉の張られ方といった脊柱構造の解析を行なった．

#### アクチュエータの選定

ロボット用のアクチュエータには，古くから一般に利用されているものとして電気式，油圧式，空気圧式等があり，比較的新しいアクチュエータとして形状記憶合金アクチュエータ [84]，水素吸収合金アクチュエータ [89, 101]，高分子ゲルアクチュエータ [92] などがある [95]．ここでは，人間の筋肉に相当する動きをするアクチュエータが求められ，

- 柔軟性，制御性，耐久性
- ストローク大 (収縮率 30% 程度，数 [cm])
- 小型, 軽量

といった特性が求められる．新しいアクチュエータは人工筋へ向けた研究として位置づけられているものもあるが，ストロークや制御性の面でここで開発するロボットに容易に応用できる段階まで到っていない．そこで，ここでは従来からある電気式，油圧式，空気圧式のアクチュエータに注目した．この三者を定性的に比較すると Table 4.9 のようになる．



- 空気圧人工筋 36 本, すべて on-off 制御, 使用圧力 4 気圧.
- 頭部に小型 CCD カメラを搭載.
- 全長 約 720mm

**Fig. 4.44** 脊椎型ロボット BeBe
Spine robot BeBe

Table 4.9 電気，油圧，空気圧アクチュエータの比較
The comparison between electric, hydraulic, and pneumatic actuators

| | | 電気式 | 油圧式 | 空気圧式 |
|---|---|---|---|---|
| 駆動系 | 直線運動 | 普通 | 容易 | 容易 |
| | 応答性 | 極めて良 | 良 | 普通 |
| | 駆動力 | 小〜大 | 中〜極大 | 小〜大 |
| | 過負荷の対処 | 困難 | 普通 | 容易 |
| | 構造 | やや複雑 | やや複雑 | 簡単 |
| | 取付の自由度 | 中 | 大 | 極めて大 |
| 制御系 | 位置制御 | 可能 | 可能 | やや困難 |
| | 力制御 | 可能 | 困難 | 可能 |
| | 対振動性 | 悪 | 普通 | 普通 |
| コスト | | 普通 | 高い | 低い |

空気圧式アクチュエータは，制御性で電気式アクチュエータに劣るものの，過負荷に強いという柔軟性を実現するために重要な特性を備えている．このため， BeBe には空気圧アクチュエータを利用することにした.

### 空気圧人工筋

BeBe で使用した空気圧人工筋はマッキベン型と呼ばれ，ゴムチューブをスリーブで覆った構造になっている ( **Fig. 4.45** )．スリーブは繊維コードを網状に編んで作られており，内側のゴムチューブに圧力が加わると，外側のスリーブがゴムチューブの体積の増加を軸方向の収縮に変換し，収縮力を発生する [47].

BeBe には，自作した人工筋と，イギリスの Shadow 社[4)]から販売されている人工筋 (Shadow Air Muscle) を用いた ( **Fig. 4.46** )．各々の内圧一定での収縮力と収縮率の関係を **Fig. 4.47** に示す.

空気圧人工筋の以下のような特徴から，人間の筋配置を模倣する際に用いるアクチュエータとして最適である.

- 柔軟な素材のみから構成される

[4)]Shadow Robot Company Ltd., http://www.shadow.org.uk/

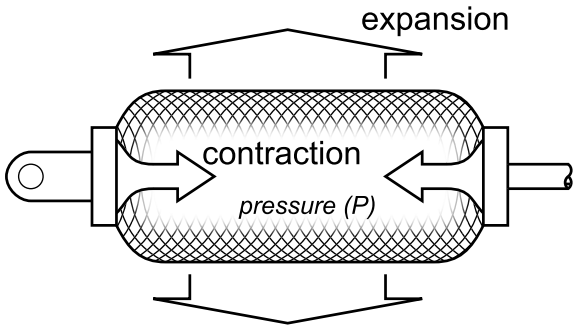



**Fig. 4.45** マッキベン式空気圧人工筋の構造
The structure of makkiben artificial air muscle

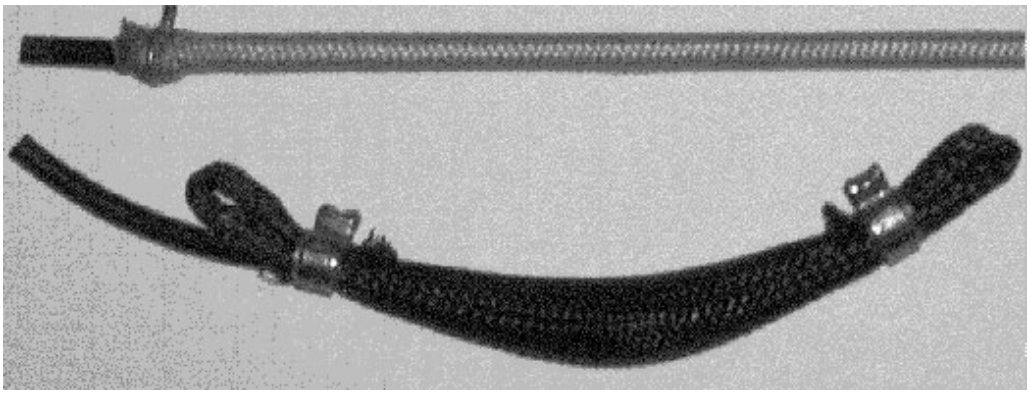


**Fig. 4.46** 自作人工筋 (上) と Shadow Air Muscle(下)
Artificial air muscles (hand-made and Shadow Air Muscle)

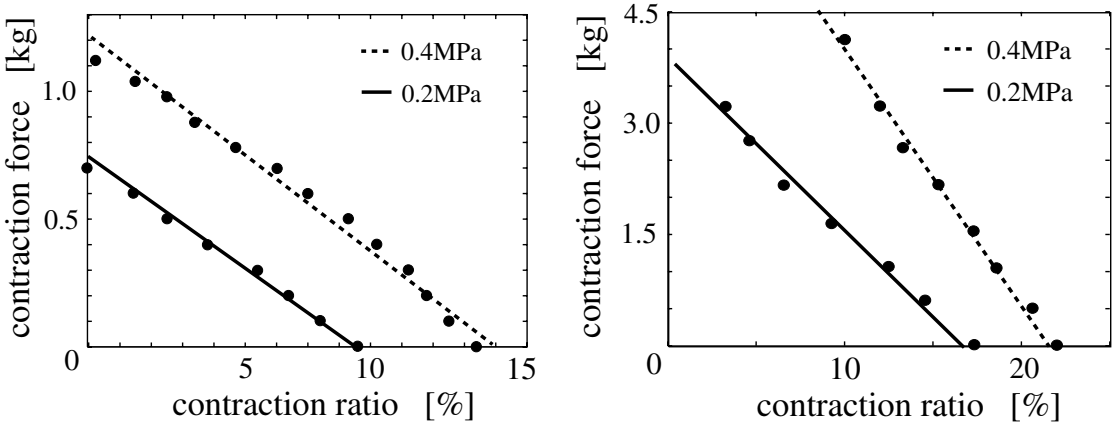



**Fig. 4.47** 自作人工筋の特性 (左) と， Shadow Air Muscle の特性 (右)
Feature of handmade air muscle (left) and Shadow Air Muscle(right)

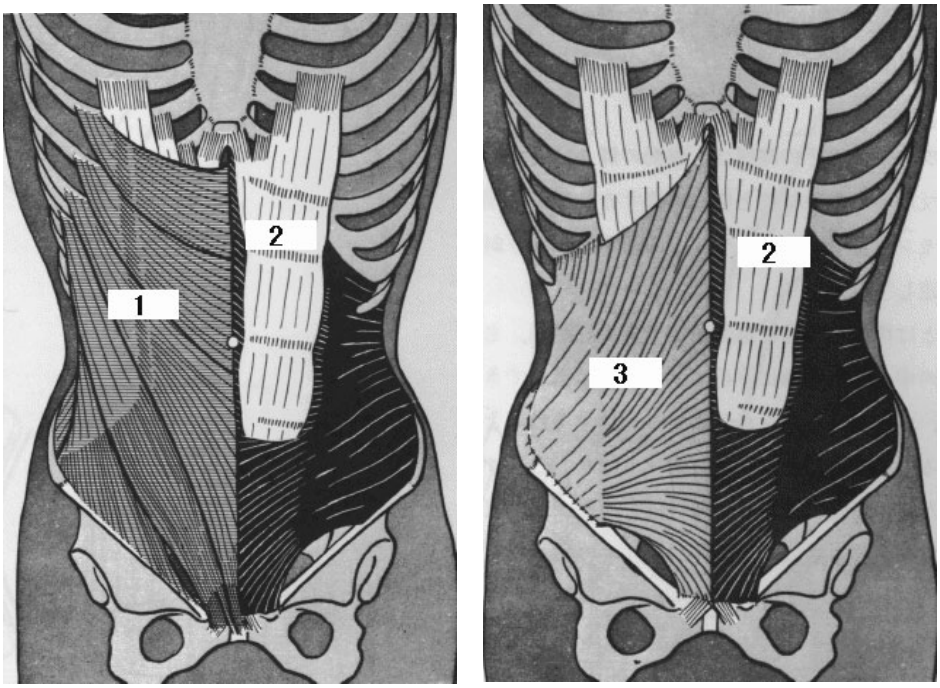



**Fig. 4.48** 1: 内腹斜筋, 2: 腹直筋, 3: 外腹斜筋
(「カパンディ関節の生理学 I 体幹・脊柱」 [35] より)
1: internal oblique, 2: rectus muscles, 3: external oblique

- 人間の筋肉と同じく，収縮力を発生する.
- 小型，軽量であり，小さなスペースに多数の人工筋の取り付けが可能である.
- 取り付け，取り外しが容易なため，筋配置の変更が簡単.

### 人工筋の配置

人間の身体において，脊柱を駆動する筋肉の代表的なものとして **Fig. 4.48** の内腹斜筋, 外腹斜筋，腹直筋がある [35]. 腹直筋により前後の屈曲，内腹斜筋と外腹斜筋により側方の屈曲と回旋が行なわれる. 内腹斜筋と外腹斜筋は，側方屈曲時は協調して働き，回旋時は拮抗して働く. これらの筋肉の配置をもとに， BeBe の人工筋配置は **Fig. 4.49** のようにした.

途中節は，全 24 節に接続を持つには 36 本では不足するため， **Fig. 4.50** に示すように, 頭部・肩部・胸部・腹部 1・腹部 2・腰部の 6 部位に大きく分け，これら 6 部位に筋取り付け点を設け， Table 4.10 に示すように 36 本の筋を配置した.

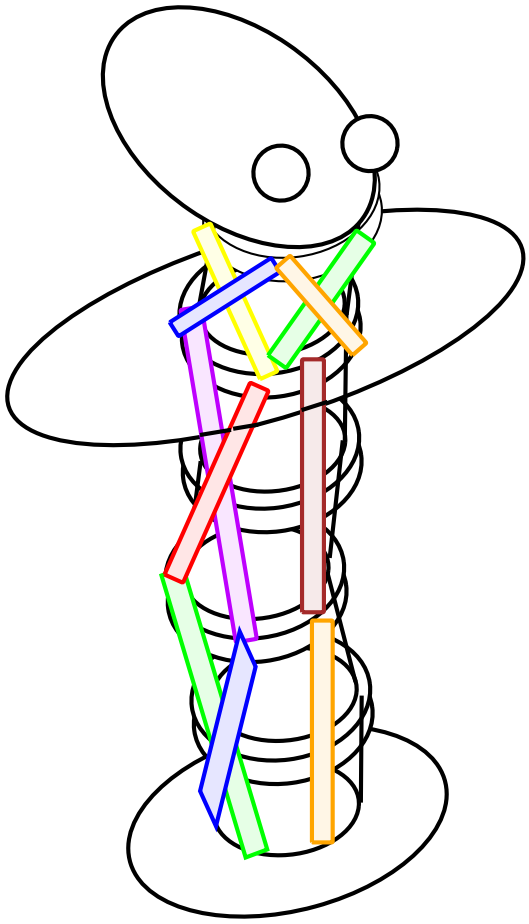


**Fig. 4.49** 人工筋の配置
Arrangement of artificial muscles

Table 4.10　BeBe の筋配置
The arrangement of BeBe's muscles

| 人工筋番号 | 取り付け位置 始点 | 取り付け位置 終点 | 拮抗する人工筋 |
|---|---|---|---|
| 1 | 腹 2–2 | 腰 –2 | 4 |
| 2 | 腹 2–6 | 腰 –6 | 3 |
| 3 | 腹 2–3 | 腰 –3 | 2 |
| 4 | 腹 2–5 | 腰 –5 | 1 |
| 5 | 腹 1–1 | 腰 –1 | 6 |
| 6 | 腹 1–4 | 腰 –4 | 5 |
| 7 | 胸 –1 | 腰 –1 | 8 |
| 8 | 胸 –4 | 腰 –4 | 7 |
| 9 | 腹 1–3 | 腰 –2 | 10 |
| 10 | 腹 1–5 | 腰 –6 | 9 |
| 11 | 腹 1–2 | 腰 –3 | 12 |
| 12 | 腹 1–6 | 腰 –5 | 11 |
| 13 | 胸 –3 | 腰 –2 | 14 |
| 14 | 胸 –5 | 腰 –6 | 13 |
| 15 | 胸 –2 | 腰 –3 | 16 |
| 16 | 胸 –6 | 腰 –5 | 15 |
| 17 | 頭 –3 | 肩 –2 | 18 |
| 18 | 頭 –5 | 肩 –6 | 17 |
| 19 | 頭 –6 | 肩 –2 | 23 |
| 20 | 頭 –6 | 肩 –5 | 21 |
| 21 | 頭 –2 | 肩 –3 | 20 |
| 22 | 頭 –2 | 肩 –6 | 24 |
| 23 | 頭 –5 | 肩 –3 | 19 |
| 24 | 頭 –3 | 肩 –5 | 22 |
| 25 | 肩 –1 | 腹 1–1 | 35 |
| 26 | 肩 –2 | 胸 –3 | 34 |
| 27 | 肩 –2 | 腹 1–3 | 30 |
| 28 | 肩 –3 | 胸 –2 | 36 |
| 29 | 肩 –5 | 腹 1–6 | 32 |
| 30 | 肩 –6 | 腹 1–5 | 27 |
| 31 | 肩 –4 | 腹 2–4 | 33 |
| 32 | 肩 –3 | 腹 1–2 | 29 |
| 33 | 肩 –1 | 腹 2–1 | 31 |
| 34 | 肩 –6 | 胸 –5 | 26 |
| 35 | 肩 –4 | 腹 1–4 | 25 |
| 36 | 肩 –5 | 胸 –6 | 28 |

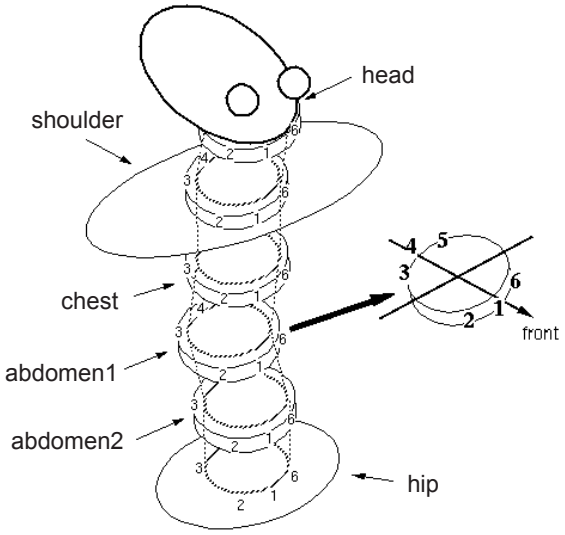



**Fig. 4.50** BeBe の部位
BeBe's tipical parts

### 空気圧人工筋の駆動

空気圧アクチュエータは圧縮空気を用意するエネルギー源が必要で，一般にはコンプレッサが利用される．空気圧の制御には一般に電磁弁が利用されることが多い．筋の圧力を制御するためには圧力センサと弁が必要である． BeBe の 36 本の筋を全て圧力制御するためには，圧力調節サーボ系を 36 セット用意しなければならない．人工筋のような流量の少ない圧力制御系は，弁の高速動作が要求され，高速動作弁または流量調節器の利用などが必要になる．圧力計測センサも 36 個必要になる． BeBe の駆動系としては，

- 冗長性の大きいシステムなので一本の筋は弁の開閉の二状態で制御するのでも，脊椎全体としては状態空間は比較的広くなりセンサベーストな制御のトライアルという目的のためには充分なのではないか.
- 大掛かりな装置は将来的にロボットに搭載する事も見込めず，自立型への発展が困難.

といった理由から，各筋は，汎用の電磁弁の開閉による，収縮しているかしていないか (on-off) の二状態で制御することにした.

### システム構成

BeBe のシステムは，空気圧系，電気系，画像処理部の三部分と，全体を統合する PC から構成される． PC からワンチップマイコン (日立， H8/3334YF-ZTAT) を通して小型電磁弁を

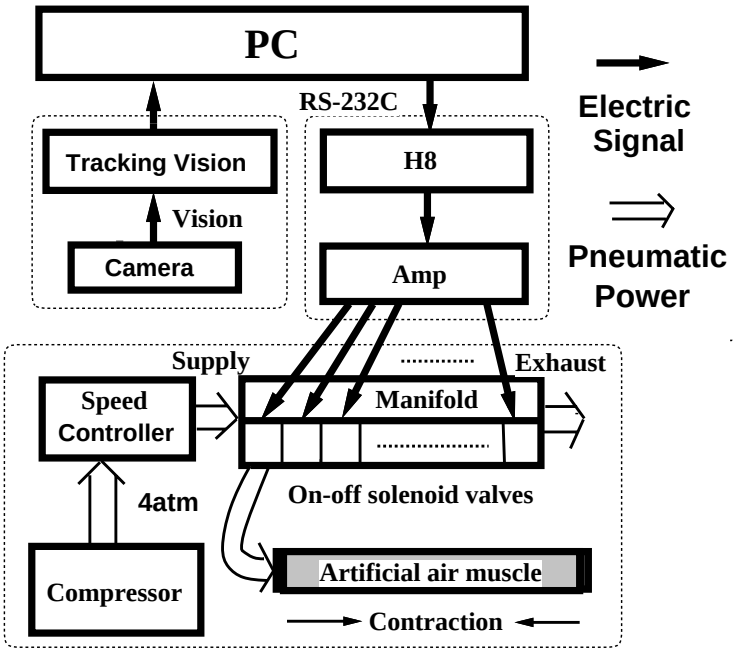



**Fig. 4.51**　システム構成図
System structure

駆動し，コンプレッサから供給される約 4 気圧の高圧空気を制御している． また，画像処理ボード (富士通トラッキングビジョン) を用いて，頭部のカメラ画像上の物体の追跡処理を行なっている ( **Fig. 4.51** ).

### 4.8.3　BeBe の制御モデル

#### 順運動学・逆運動学

まず，幾何学的な制御量と従来型ロボットの制御法により， BeBe を制御することを考えてみる．まず， BeBe の筋長・関節変位の間の運動学，すなわち筋の長さから関節変位を求める計算を考える．筋の長さを $\boldsymbol{q} = (m_1, m_2, \cdots, m_{36})^T$ とし，各関節の変位を $\boldsymbol{\theta} = (x_1, y_1, z_1, \phi_1, \theta_1, \psi_1, \cdots, x_{24}, y_{24}, z_{24}, \phi_{24}, \theta_{24}, \psi_{24})^T$ とする ($x_n$， $y_n$， $z_n$ もいくらか変化する) と，運動学が線形だと仮定すれば,

$$\boldsymbol{\theta} = \boldsymbol{J}\boldsymbol{q} \tag{4.43}$$

と表せる．疑似逆行列を用いて筋張力と関節発生力の関係も求める事ができる．

しかし， BeBe のボディでは筋の干渉もあり，摩擦も大きく，筋自体にヒステリシスもある．そのため，運動学は線形と仮定できない．さらに，構造的なヒステリシスも見られるた

め， $\boldsymbol{q}$ だけからは $\boldsymbol{\theta}$ が決まらない．したがって，動く前の姿勢・筋長にも依存するとすると，

$$\boldsymbol{\theta}(t) \quad = \quad \mathrm{f}\left( \boldsymbol{q}(t),\ \boldsymbol{q}(t-1),\ \boldsymbol{\theta}(t-1) \right) \tag{4.44}$$

なる関数 $\mathrm{f}$ を求めなければならない．これを計算的に求めるのは困難である．動力学も同様に，計算的に求めるのは困難である．

### 幾何モデルを用いないセンサベーストな制御

そこで考えられるのは，実際に筋を駆動しボディを動かして，その結果の蓄積から上記 $\mathrm{f}$ またはそれに相当する計算モデルを導き出すというのが一つの方法である．さらに，4.1.2節に述べたように，そのモデルは常に更新されてゆくものであるべきで，結果の蓄積とモデルの更新の仕組みを持つ事が望ましい．本研究では，こうした仕組みの提案例として，姿勢データベースを用いた行動生成環境の提案も行っている (第 5.9節)．ここでは，モデル自己生成・自動更新といった枠組みではなく，センサ情報に基づく幾何学的なモデルを利用しない制御法のトライアルに関し述べる．

ここでのアプローチは，まず BeBe の代表姿勢を何通りか計測し，得られたデータから人工筋の on-off 状態と全身の姿勢の汎用性のある対応関係を求め，それを上の $\mathrm{f}$ に相当するものとして制御に利用しようというものである．

その対応関係として，人工筋 1 本を on にした時の全身の変形量を計測し，複数の人工筋が on になった時の変形量が 1 本を on にした姿勢の線形和になるかどうかを考える．例えば，筋 1 のみを on にした時の姿勢を $\boldsymbol{\theta}_1$，筋 2 のみを on にした時の姿勢を $\boldsymbol{\theta}_2$ とすると，筋 1 および 2 を on にした時の姿勢は， $\boldsymbol{\theta}_{12} = \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2$ になるかどうかである．しかし，実験の結果，BeBe は単純な重ね合わせは適用できない構造であることがわかった [38]．しかし， on か off の二値であっても 36 本あると， $2^{36} = 68,719,476,736$ 通りの組合わせがあり，全ての状態を計測することは不可能である．しかも，ヒステリシスもある．

そこで， BeBe のトラッキング動作を行うためには，次のような制御法を考案した．まず，全ての筋が off の初期状態から筋 1 本を on にしたときの BeBe の姿勢を 36 通り計測する．そして，実際の動作を行う際にはその 36 パターンの入出力計測結果を利用した，汎用なフィードバック則に基づいて制御を行った．

**Fig. 4.52** に示すようなセットアップで，初期状態で対象物が視野の中央になるようにして，筋 $i$ を 1 本だけ on にした時に，視野内での対象物の移動 $\boldsymbol{v}_i$ を計測した． **Fig. 4.53** は各筋に対応する $\boldsymbol{v}_i$ をプロットした図である．

**Fig. 4.53** のデータを実際の動作に利用する際には，次のようなアルゴリズムによった．まず，対象物の位置が視野座標系内でどの位置にあるかを計測する．そして，視野の中心から

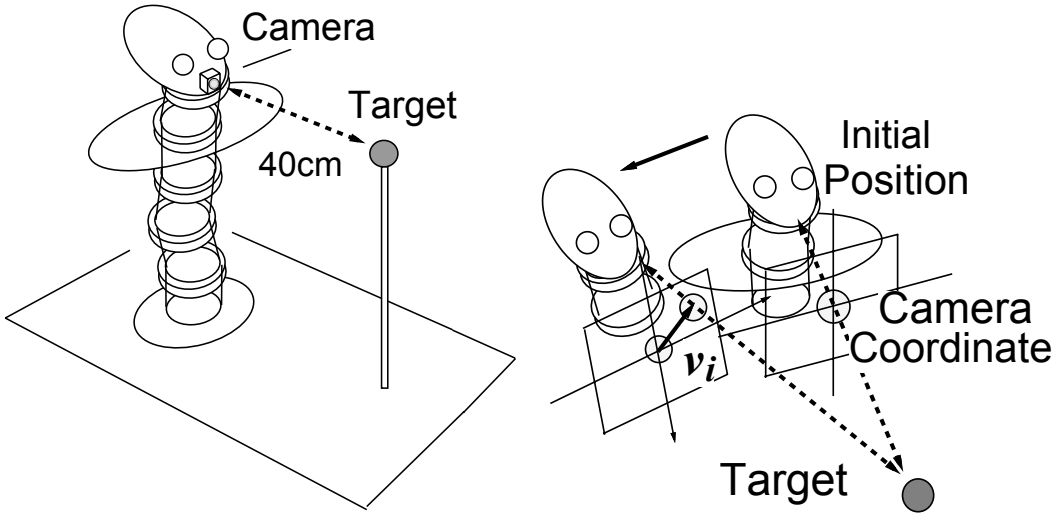



**Fig. 4.52** 筋を 1 本だけ on にした時の画像の変化の計測
Measuring the visual movement when one muscle is on

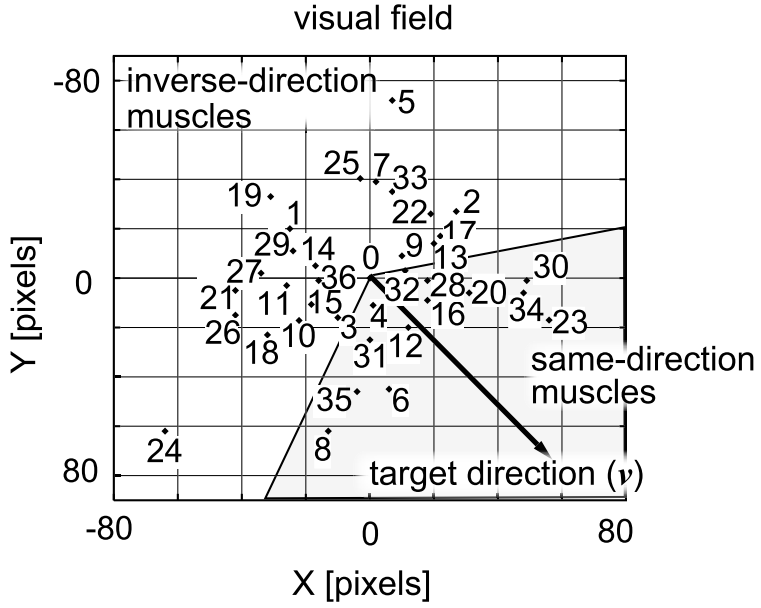



**Fig. 4.53** 視覚計測の結果と順・逆方向筋の定義
The result of visual-measurement and definition of inverse and same-direction muscles

その位置の方向 $\boldsymbol{v}$ を計算し，それを目標方向 (target direction) とする．そして，筋 $i$ を 1 本だけ on にしたときの視野のずれの方向 $\boldsymbol{v}_i$ と比較する． $\boldsymbol{v}$ と $\boldsymbol{v}_i$ のなす角が一定範囲内にあるような筋 $i$ の群を，ロボットを目標方向に動かし得る筋とみなして順方向筋 (same-direction muscles) と呼ぶことにする．そして順方向筋に属さない筋を逆方向筋 (inverse-direction muscles) と呼ぶことにする ( **Fig. 4.53** )．そして， 1 制御周期毎に，

1. 画像入力から $\boldsymbol{v}$ を算出する．
2. 逆方向筋のうち on のものがあれば，そのうちで $\boldsymbol{v}$ となす角が最も大きいものを 1 本 off にする．
3. 2. に当てはまるものが無い場合，順方向筋のうち off であるものがあれば，そのうちで $\boldsymbol{v}$ となす角が最も小さいものを 1 本 on にする．

という手順の判断を行う．順方向筋と逆方向筋を決定する閾値は $\boldsymbol{v}$ と $\boldsymbol{v}_i$ のなす角を $\cos\theta_i$ とすると， $\cos\theta_i = 0.2$ という数値を用いた．この値は経験的に求めた． on にする操作より off にする操作が優先するのは，発散し全ての筋が on になってしまうのを防ぐためである．また，中心付近に経験的に定めた不感帯を設け，対象物がその領域内にある時は操作を行わないものとした．

### 4.8.4 トラッキング動作実験

#### 通常のトラッキング動作

4.8.3節に述べたセンサベーストな制御法により， BeBe のトラッキング動作の実験を行った．その様子を **Fig. 4.54** に示す． 画像フィードバックの周期は 66[ms] である．人間が普通に対象物を動かす程度の速さであればしっかり対象物を追跡し続けることができた．時折画像の中心を往復する形で二つの姿勢を繰り返す事もあるが，それは前項に述べた不感帯の面積内に解が無いためであると考えられる．

#### 外乱を受けながらのトラッキング動作

4.8.3節に述べたアルゴリズムの特長の一つに，外乱を受けながらでもタスクを遂行できることがある．外乱があって計測データとの整合性があまり無いような状況でも，計測した方向と操作する筋による姿勢の変化方向のずれが 180° 以上にならなければ，向かうべき方向に真っ直ぐ向かわないにしても目標に近づく方向に変化を起こすことができる．そこで，冗長性を活用して，タスクの遂行が実現できる．人が脊椎の中ほどを手で押して外乱を与えたときに，トラッキング動作を続ける様子を **Fig. 4.55** に示す．

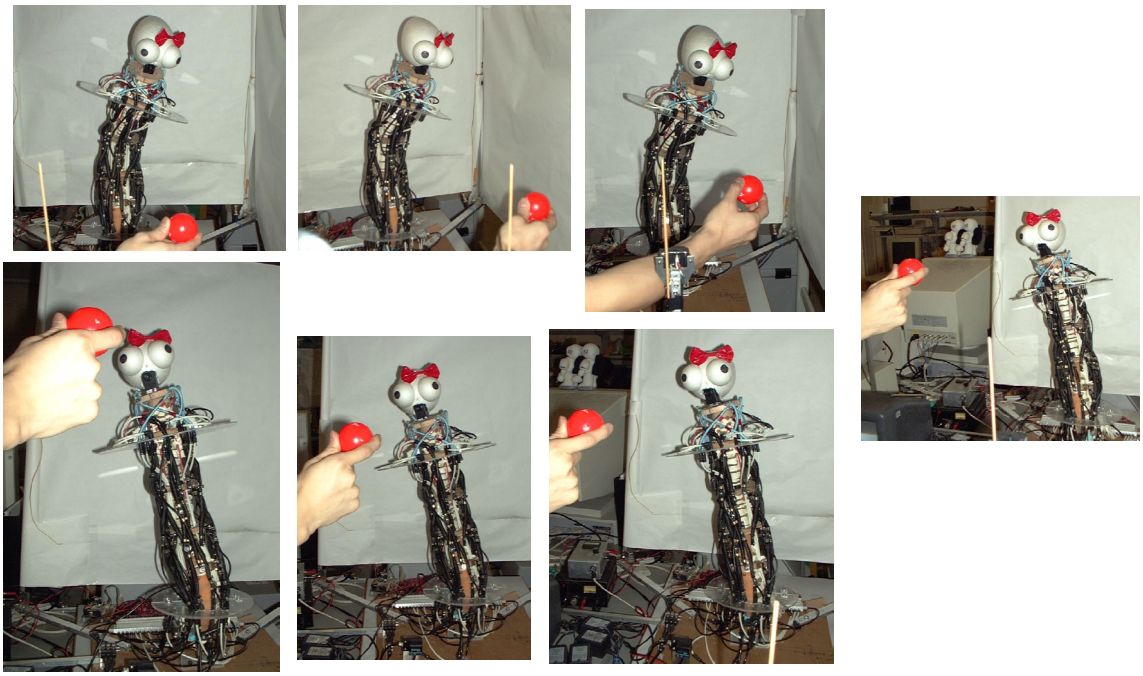


**Fig. 4.54** BeBe の視覚フィードバックによる物体追跡動作
BeBe's tracking an object by visual feedback

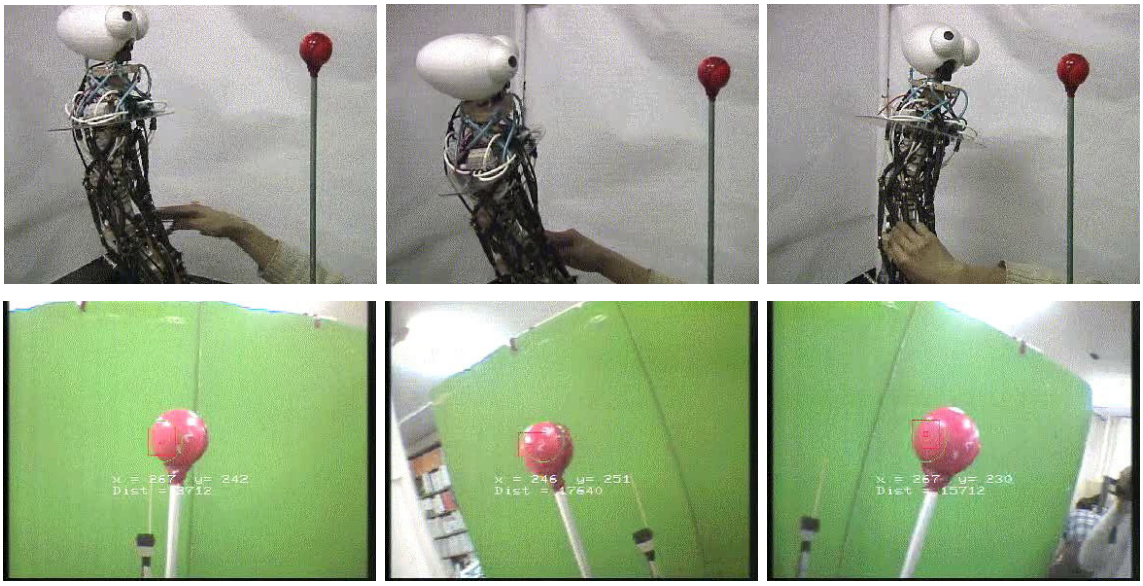


**Fig. 4.55** 外乱を受けながらの BeBe の物体追跡動作
BeBe's tracking while being disturbed

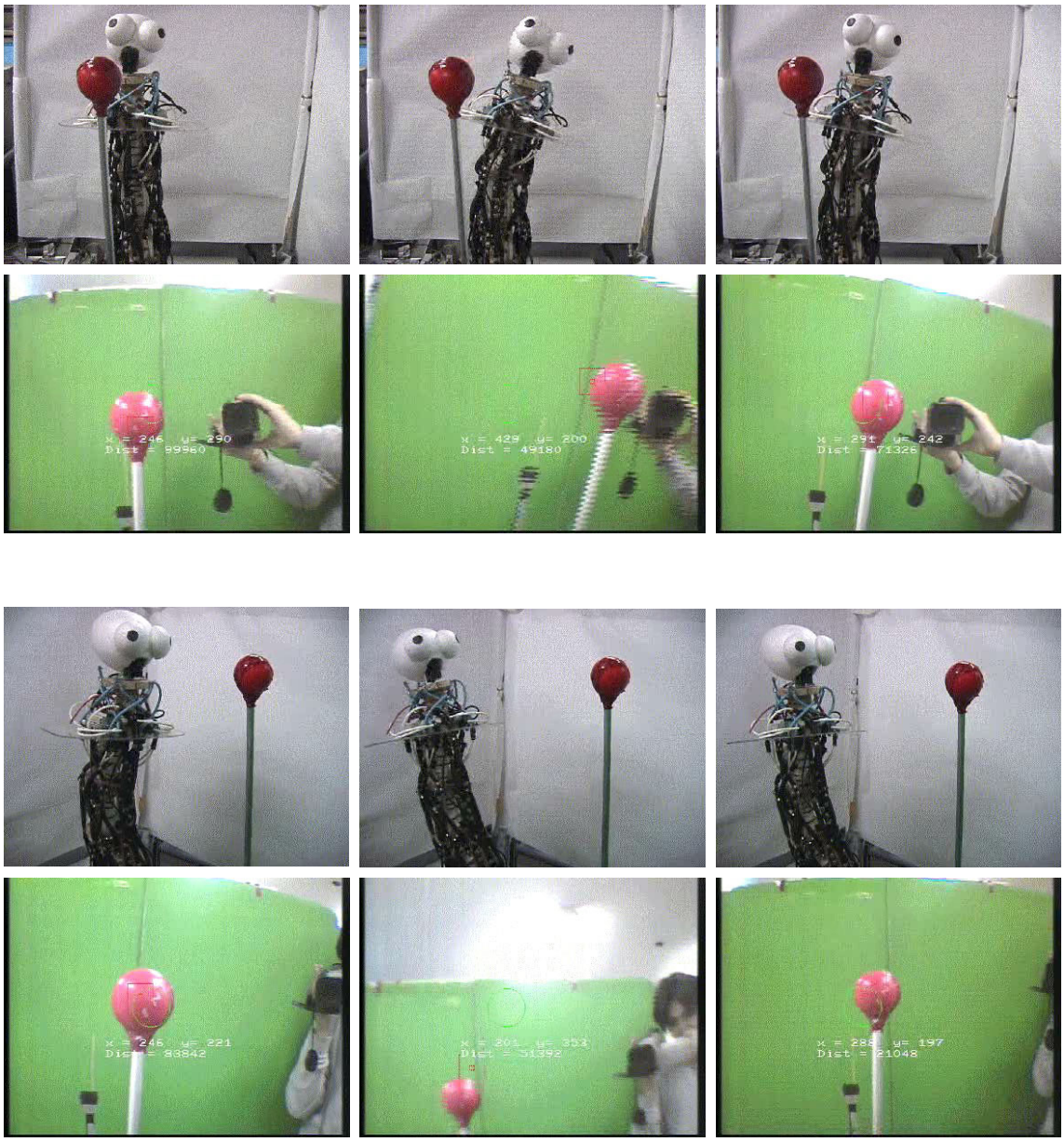



**Fig. 4.56** 制約条件下での BeBe の物体追跡動作
BeBe's tracking under some restriction

**Fig. 4.55** の下段は内部画像である．人間に手で左右に押されながらも，対象物を視野の中心に保っている様子がわかる．

#### 制約条件下でのトラッキング動作

4.8.3節に述べたアルゴリズムのもう一つの特長として，姿勢や筋に制約条件が課された状態でも，冗長性を利用してタスクを遂行することができるという点が挙げられる．例えば，人間が机の前の椅子に座っており，机の奥の方にある物を掴もうと手をのばす時，脊椎を前屈させなければ届かないという状況を想定する．この時，腹部は机の縁に接触するため脊椎の下部 1/3 程は曲げることができないとする．その際，脊椎の残りの上部 2/3 程を曲げることで物体に手が届くようにする．これは，人間の脊椎の冗長性により，下 1/3 の姿勢を動かさないという制約条件の下にも，タスクを遂行することができるという例である．

BeBe において，こうした利点を再現する実験を行った．制約条件として，一部の筋肉は常に on でなければならないという制約を課し，残りの制約の課されていない筋を 4.8.3節に述べた同じアルゴリズムで制御することで，冗長性を活用し制約条件下でトラッキングを遂行する． **Fig. 4.56** に実験の様子を示す．図の上部の 2 段は，脊椎を左に傾ける筋をいくつか on にするという制約条件の下にトラッキング動作を行う様子を示し，最上段が外部画像，二段目が内部画像を示す．左端の図から始まり，制約条件の一部筋を on にしたところ対象物が中心から外れ (上二段中央の図)，左端の図では制約の課されていない筋群のみを操作して再び対象物が視野の中心になるような姿勢を取っている様子がわかる．図の下部の 2 段は，脊椎を後に傾ける筋をいくつか on にするという制約条件の下にトラッキング動作を行う様子を示し，三段目が外部画像，最下段が内部画像である．上二段と同様に，左端の姿勢から始まり，中央の図で視野の中心から外れた対象物が，左端の図では視野の中央に来るように姿勢が修正されている．

## 4.9 可変柔軟な脊椎を持つロボットの基礎的な制御のまとめ

### 4.9.1 幾何学ベーストとセンサベースト

本章では，第3章に述べた脊椎構造をいかに制御するかに関し述べた．まず，これまでのロボットが幾何学的な制御量を計算的手法によって制御していたのに対し，人間の身体構造に近づいた複雑な身体構造を持つロボットにおいては，ある程度従来型の方法を踏襲した制御インタフェースを確立しつつ，最終的にロボットが活動する場面においてはセンサ情報に基づいたタスク依存な制御量を，センサ空間におけるフィードバック・フィードフォワードにより制

御する方法に重点が移ってゆくであろうということを指摘した．これは，言い換えれば幾何学ベースからセンサベーストへとモデルが移行してゆくという表現もできよう．

こうした立場に基づいて，幾何学的制御量に基づく完全ではない姿勢制御の枠組みと，センサ情報を利用した不完全な部分の補間，そして，全面的にセンサ情報に基づいた制御法のトライアルを行った．以下にその概要を述べる．

**教示・再生による制御** 全筋とも筋張力制御 (T 制御) により一定張力を保つ状態にして，人間が姿勢・動作の直接教示を行う．教示時の筋長の記録を取り，全筋とも筋長制御 (L 制御) により教示した姿勢・動作を再現する．

**幾何モデル** 筋の干渉を考慮せず，直線距離で姿勢 ⇒ 筋長を求める (式 (4.25))．筋長 ⇒ 姿勢への変換は，姿勢 ⇒ 筋長の変換を教師とし NN で学習することで獲得した．

$$l_j \quad = \quad \sum_k | \, {}^0\boldsymbol{p}_{jk} - {}^0\boldsymbol{q}_{jk} \, | \tag{4.25}$$

**張力センサを利用した制御モード** 脊椎構造の姿勢制御および柔軟性制御のために，筋の制御モードとして，筋長制御 (L 制御)，張力制限付き筋長制御 (LL 制御)，筋張力制御 (T 制御)，疑似ばね制御 (S 制御) の四種類の制御モードが必要であることを述べた．さらに，姿勢を変えるときに伸びる筋を伸び筋，縮む筋を縮み筋と呼ぶと， (1) 通常の動作時は縮み筋を L 制御・伸び筋を LL 制御， (2) 硬くする必要がある時・力が必要な時は全筋 L 制御， (3) 柔軟性が必要な時は全筋 S 制御でバネ定数を調節する (式 (4.35))，という制御モードの使い分けが適していることを実験により示した．

$$\boldsymbol{K}_j \quad = \quad {\boldsymbol{J}_m}^T \boldsymbol{K}_m \boldsymbol{J}_m \tag{4.35}$$

**センサベーストな制御法のトライアル** 36 本の on,off 制御の筋により 24 節の脊椎を駆動する構造のロボットを製作し，初期状態から筋を 1 本だけ on にした時の視覚情報の変化を計測・記録することで得られた 36 種類のデータを利用して，汎用性のある簡易なセンサフィードバック則に基づき，物体のトラッキング動作の実験を行った．姿勢の幾何学的情報量を全く利用しないタスク依存なセンサ情報を利用した制御法により，タスクを遂行した．さらに，脊椎の冗長性の利点を活用し，外乱や制約条件の存在する状況においてもタスクを遂行できることを実験により示した．

### 4.9.2 モデルの理想像

自己センサの情報形式に基づく入出力モデルを自己形成するのが理想であると考える．人間がユークリッド空間に基づくモデル構成を検討・実装するのではなく，モデルの自己生成が

可能な枠組みを模索するのが理想であると考える．ユークリッド空間に基づくことを否定するのではない．ユークリッド空間における量を得るセンサを備えた身体構造を持つ場合は，それに基づくモデルにより制御されるのが自然な形である．従来のロボットの手法はこれに該当する場合が多いのではないだろうか．人間場合は，そういった制御量をきちんと計測できるセンサを備えておらず，身体構造も不確定性を持つ構造であり，それを制御するのに充分な精度と密度のセンサを備え，こうした構造に対応したセンサ空間のモデルに基づいて制御を行っているのではないだろうか．脊椎構造を持つロボットのように，人間の身体構造に近づきつつあるようなロボットの場合，人間の手法に近づいてゆくという方向性は間違ってはいないのではないだろうか．

# 第 5 章

# 柔軟な脊椎を有するロボットの全身行動の実現

本章では，全身動作・全身行動の生成法およびその制御法を論じる．人が試行錯誤により動作・行動を作成するために必要な枠組み (インタフェース)，シミュレータと GA を用いる動作生成方法，姿勢データベースというしくみの提案・構築とそれに基づく全身行動の生成法・制御法などに関し述べる．シミュレーションと GA あるいは NN を用いて全身動作を生成することで，脊椎構造を持つ全身型ロボットという挙動予測がしづらいロボットにおいて，実機での動作可能なパターンに近い解を得られるということだけでも大きな有効性がある．シミュレーションの環境は複数種類作成した．最も精度が高く解析時間のかかる有限要素法シミュレータ ANSYS，最も解析時間が短く精度はそれほど正確性は期待できない MathEngine，その中間的位置付けとなる DADS である．それぞれの使い分けを示す．さらに，教示や動作の実行中のセンサ状態を常に監視し，初めて現れた状態を検出するとデータベースに追加し，データベースを利用して動作・行動の生成・制御を行うための枠組みを，姿勢データベースというキーワードを用いて提案し，その有効性を示す．

## 5.1 脊椎を利用した全身行動

### 5.1.1 従来型の全身型ロボットを扱うソフトウェアとの透過性

詳しくは第 6.5節に述べるが，従来型の全身型ロボットを扱うソフトウェアと透過性を持ったシステムになっている．従って従来の全身型ロボットの動作の作成と同様の手順で全身動作を作成することができる．

ユークリッド空間の表現に基づく幾何モデルを中核とするソフトウェア環境 [26] である．計算機上で軌道を作成し実機に適用することで全身行動を作ってゆくことができる．

また，センサフィードバックを利用するような動作を記述する場合は，センサ入力を読み，それに基づいて幾何モデル上でのロボットの姿勢に変更を加え，その情報をアクチュエータに指令として渡すという手順で，記述することができる．

### 5.1.2 直接教示

前節の方法で動作を作成していても，幾何モデルベースの制御が破綻するような姿勢をとらせたい場面も出てくる．また，コンピュータモデル上では直感的にどこをどう回せば目的の姿勢にできるかがわかりにくい場合もある．

こうした場合は，全筋張力制御モードにして直接教示を行い，その時の筋長の状態を記録しておき，ロボットの行動の流れの中で必要な部分だけ，直接教示により得た情報を混在させることができる．

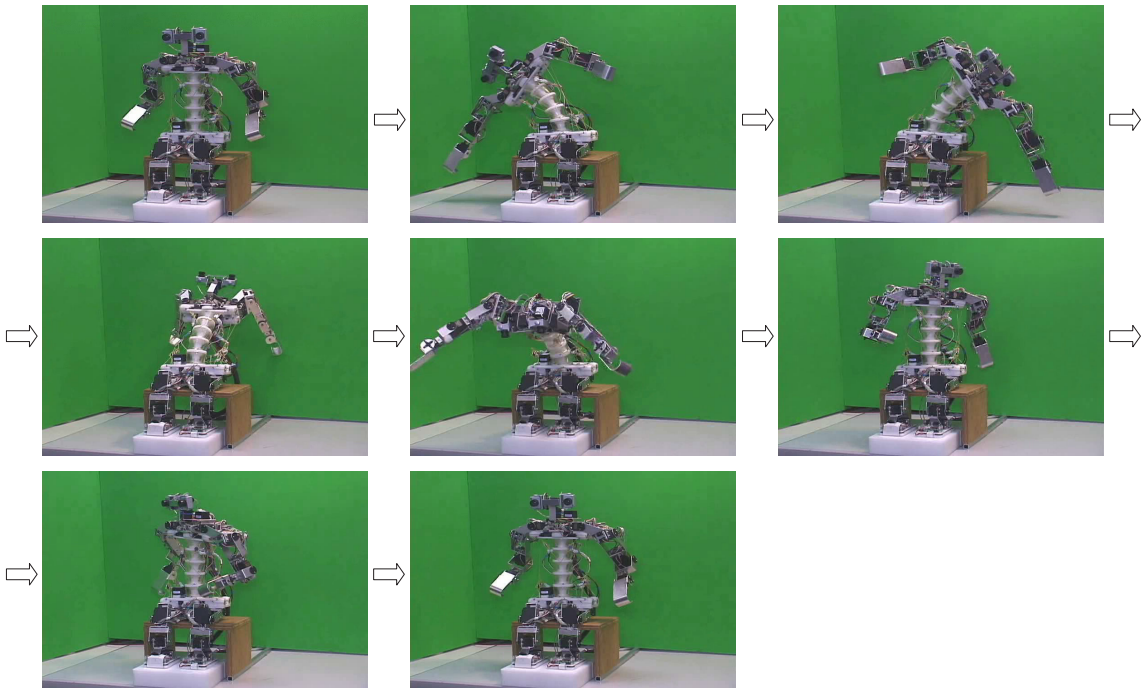


**Fig. 5.1** Cla の前後・左右・ねじり動作
Roll, pitch, and yaw motion of Cla

### 5.1.3 複雑な身体構造のロボットの全身行動

脊椎構造を持つロボット特有の動作生成法もある．複雑な動的動作の実現には，従来型ロボットと同じ幾何モデル環境だけでは困難である．そこで，本研究では，

1. 柔軟構造を扱える仮想環境を構築し GA などの探索手法を用いて複雑な動的動作を生成する．

2. 人間に教わる．

という二種類の解決法を示した． 1. に関しては，解析時間と解析精度のトレードオフがあり，解析精度を優先した環境と高速性を優先した環境とこれらの中間的な環境の三種類のシミュレーション環境を構築した．そして， GA の遺伝子に対応し多数のマシンを利用して並列に GA を行う環境を構築した．また，センサベーストな動的動作を自動獲得するために， NN のコントローラの重みを仮想環境と GA により最適化する実験も行った [66, 105]． 2. に関しては，直接教示により動作を実現するのも一つの方法だが，もう一つは人間の動作から採取したモーションキャプチャのデータを脊椎構造を持つロボットの動作に変換する環境を構築した．

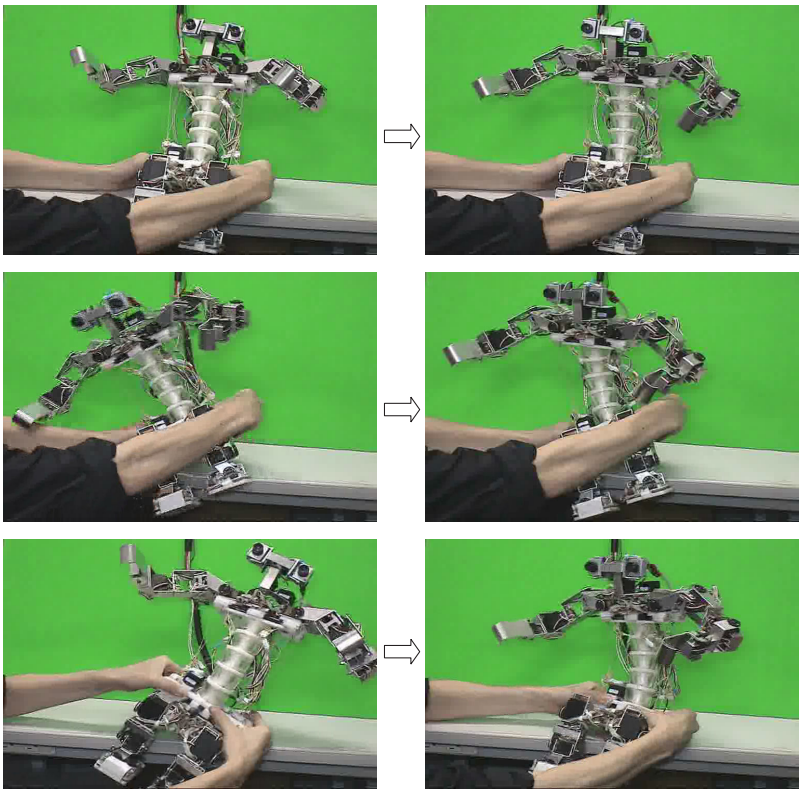


**Fig. 5.2** Cla による加速度センサを利用したバランス制御動作
Cla balancing using 3D accelerometers

## 5.2 従来型全身型ロボットと同様の方法による全身行動

### 5.2.1 Cla の前後，左右，ねじり動作実験

脊椎を持つ人間型全身ロボット Cla(第 3.6節参照) が左右，前後，ねじり動作を行う様子を **Fig. 5.1** に示す．左右・前後・ねじりとも腕の動きと協調させた動作を行うことで，全身を利用した動作になっている．腕の動きを協調させなければ，左右に屈曲するときに腕がボディに干渉するため，腕の動きを脊椎と協調させている．

図よりわかるように，足が地面に着かない程度の高さの椅子に座り左右に脊椎を屈曲した時には，手先がほぼ足裏の高さと同程度の高さまで到達している．

### 5.2.2 Cla による加速度センサを利用したバランス制御行動

**Fig. 5.2** に， Cla が 3 軸加速度センサを利用したバランス制御動作を行う様子を示す． 3 軸加速度センサにより重力方向を検出し，肩が水平でなければ，脊椎の姿勢を制御して水平に

**Fig. 5.3** 脊椎を利用して物を投げる Cla
Cla throwing an object using spine

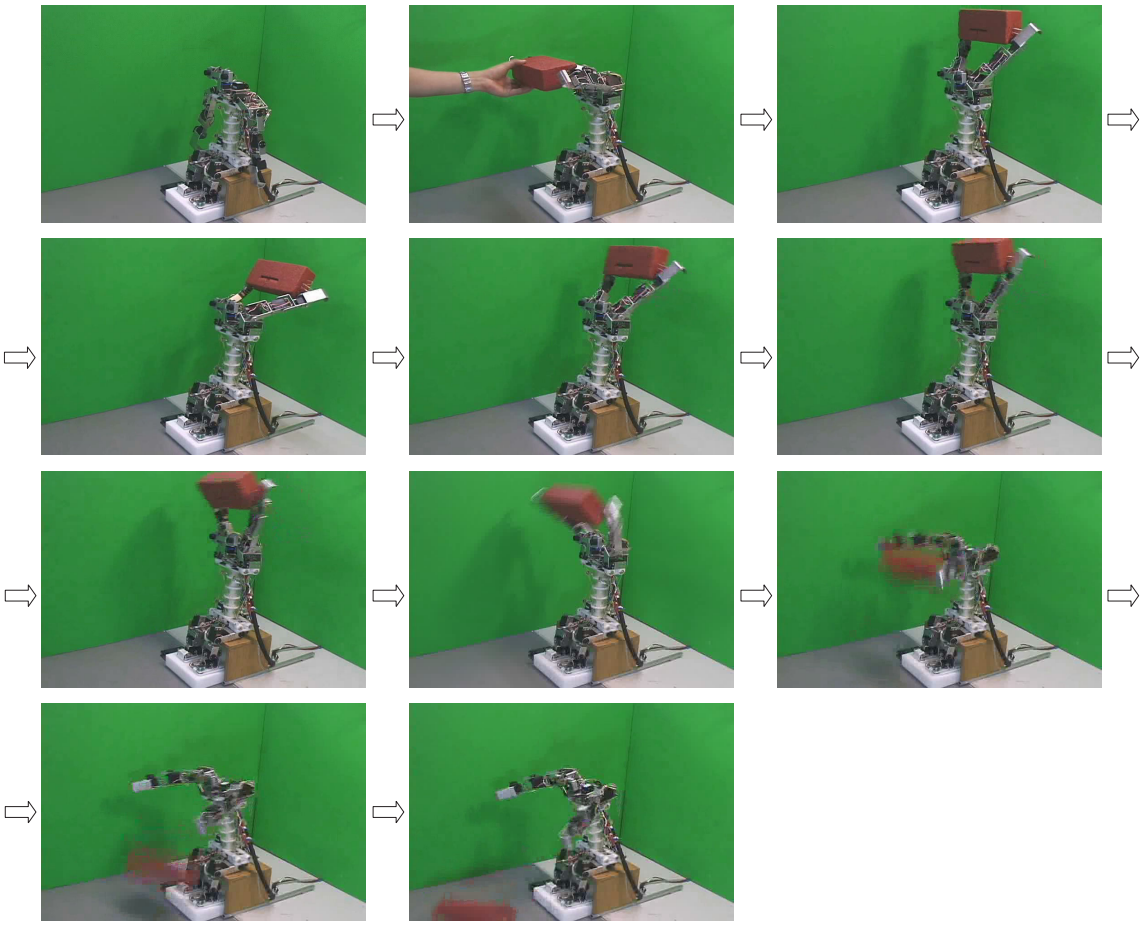


**Fig. 5.4** 脊椎を利用しないで物を投げる Cla
Cla throwing an object WITHOUT using spine

なるように動作する.

動作中は，加速度センサが自分の動作に起因する加速度を検出してしまうので，姿勢の調節とセンサ情報の読み取りはフェイズを分けて，姿勢検出→脊椎調節量算出→動作 という作業を順に繰り返すようにした．脊椎調整量の算出時には， 4.4.3節に述べたニューラルネットワークを利用した筋長情報→脊椎の姿勢情報の変換を利用した.

### 5.2.3 Cla による物を投げる行動

**Fig. 5.3** には， Cla が脊椎を利用してものを投げる様子を示す． **Fig. 5.4** は，同じ動作を脊椎を動かさずに行う様子を示す.

手先の瞬間最大速度を大きくするためには，脊椎根元部から順に動作を開始し，肩，肘という順に動作を開始することで，慣性の大きい部位が最大速度になる瞬間と手先が最大速度に

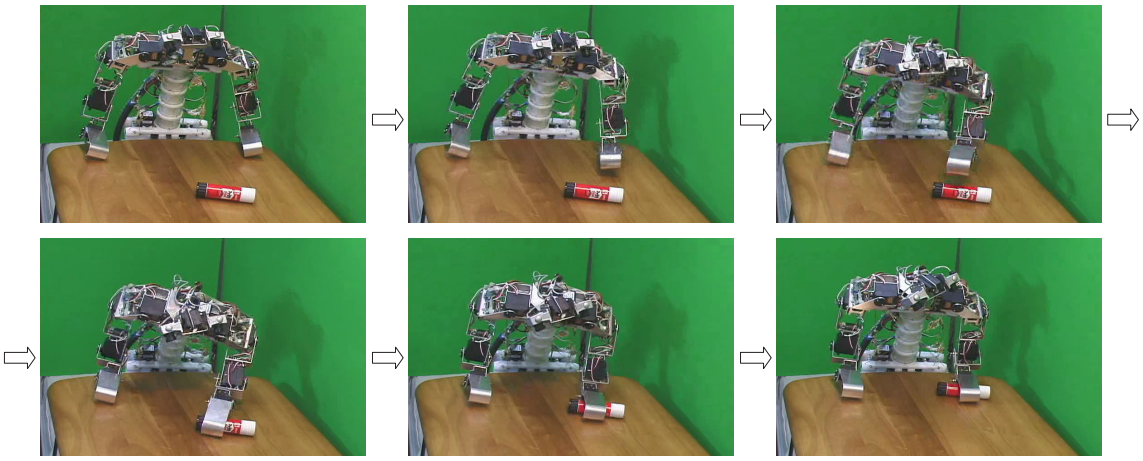


**Fig. 5.5** 机の上の物を引き寄せる Cla(脊椎利用)
Cla reaching to an object on the table using spine

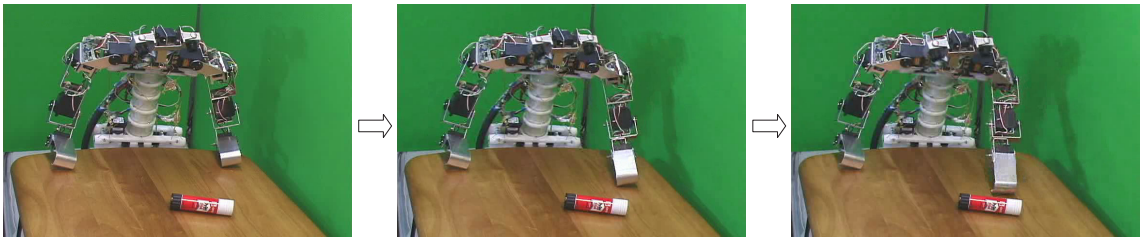


**Fig. 5.6** 机の上の物を引き寄せようとする Cla (脊椎利用せず失敗)
Cla trying to reach to an object on the table without using spine

なる瞬間を同じくすることができ，これにより物をより強く投げることができる．脊椎を利用しない場合は肩から手先までの関節しか加速に利用できないため，脊椎を利用する場合に比べ手先の最大速度は小さくなる．

### 5.2.4 Cla による机の上の物を引き寄せる行動

**Fig. 5.5** に， Cla が机の上のものを引き寄せる様子を示す．脊椎を利用しないで同様の動作を行おうとすると， **Fig. 5.6** に示すように，手が対象物に届かない．脊椎の働きが必要なデモンストレーションである．なお，脊椎の姿勢は実機で調整し決定している．

### 5.2.5 Cla による椅子に座った状態で床に置いてある物を拾う行動

椅子に座った状態で床に置いてある物を拾おうとするとき，手を伸しただけでは床に手が届かず拾うことができない．脊椎の動きを利用して脊椎を側屈させればそれが可能になる．同

様の動作を脊椎でなく股関節の自由度等で行おうとすると， **Fig. 2.1** に示すように股関節の roll 軸が不自然に曲がり，重心が外れてしまう． **Fig. 2.2** および， **Fig. 5.7** に示すように，Cla は脊椎の roll 方向の自由度を用いる事で，重心を椅子の上に置いたまま床の物を拾うことができる．

### 5.2.6 Cla による椅子に座った状態で机の上に物を置く行動

拾ったものを机の上に置くときも，座った状態では，脊椎を動かさなければ机の上に置くことができない ( **Fig. 5.7** の 9 コマ目)．脊椎をねじることで人間のような動作によって椅子に座ったまま机の上に手を伸ばして物を机の上に置くことができた．

### 5.2.7 Cla による立った状態でバランスをとりながらの全身動作

**Fig. 5.8** に， Cla が立った状態でバランスをとりながら全身動作を行う様子を示す．

最初に椅子に座った状態から立ち上がる際には，人が両手を持って少し持ち上げると，両肩のサーボエラーを監視し，サーボエラーが小さくなるように少しずつ立ち上がる．

完全に立ち上がって人が手を離すと，ゆっくりと左右の脊椎を屈曲する運動を行う． Cla は足裏が小さいのと機械的なガタなどの影響で，前後方向のバランスは非常に微妙である．そこで常に足裏の前後方向の中心付近に重心の投影点が来るように，両腕を前後にゆっくりと動かしながらバランスをとる．足裏の重心投影点が足裏中心より前方に来たら，両腕を後に動かし重心を戻そうとする．逆に重心投影点が後方に行ったら，両腕を少しずつ前に出し重心が足裏中心付近に来るまで少しずつ動かす．こうした反射を，左右の側屈の動作と並列に行うことで，立った状態での運動ができるようになった．

### 5.2.8 Cla による寝返り行動

**Fig. 5.9** は， Cla が脊椎を利用して寝返り動作を行っている様子を示す．股関節や肩のアクチュエータのパワーが不足しているため，股関節の力・肩の力だけでは体を捻って半身を起こすことができない．脊椎を yaw 軸回りに回転させる動きと肩・股関節の動きを同時に行うことで，肩・股のアクチュエータのパワーと脊椎の回転力を合わせて，半身を起こすことができている．

これにより，仰向けからうつぶせ，うつぶせから仰向けと，両方向の寝返りができるようになった．

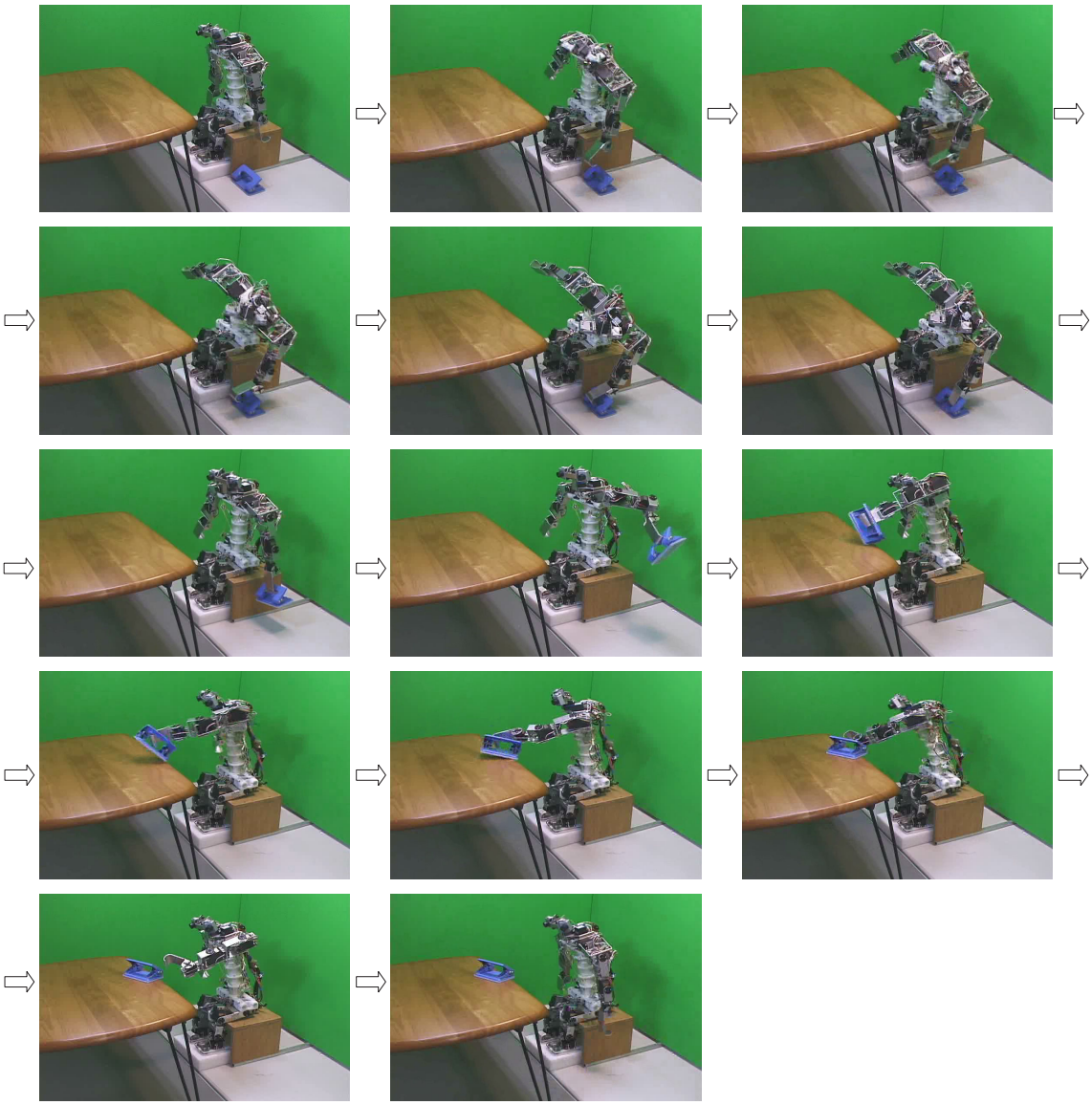


**Fig. 5.7** 床の物を拾い机の上に置く Cla
Cla picks up an object from the floor and puts it on the table

**Fig. 5.8** 足裏力センサを利用してバランスをとりながら立って全身動作をする Cla
Standing motion of Cla, balancing using foot sensors

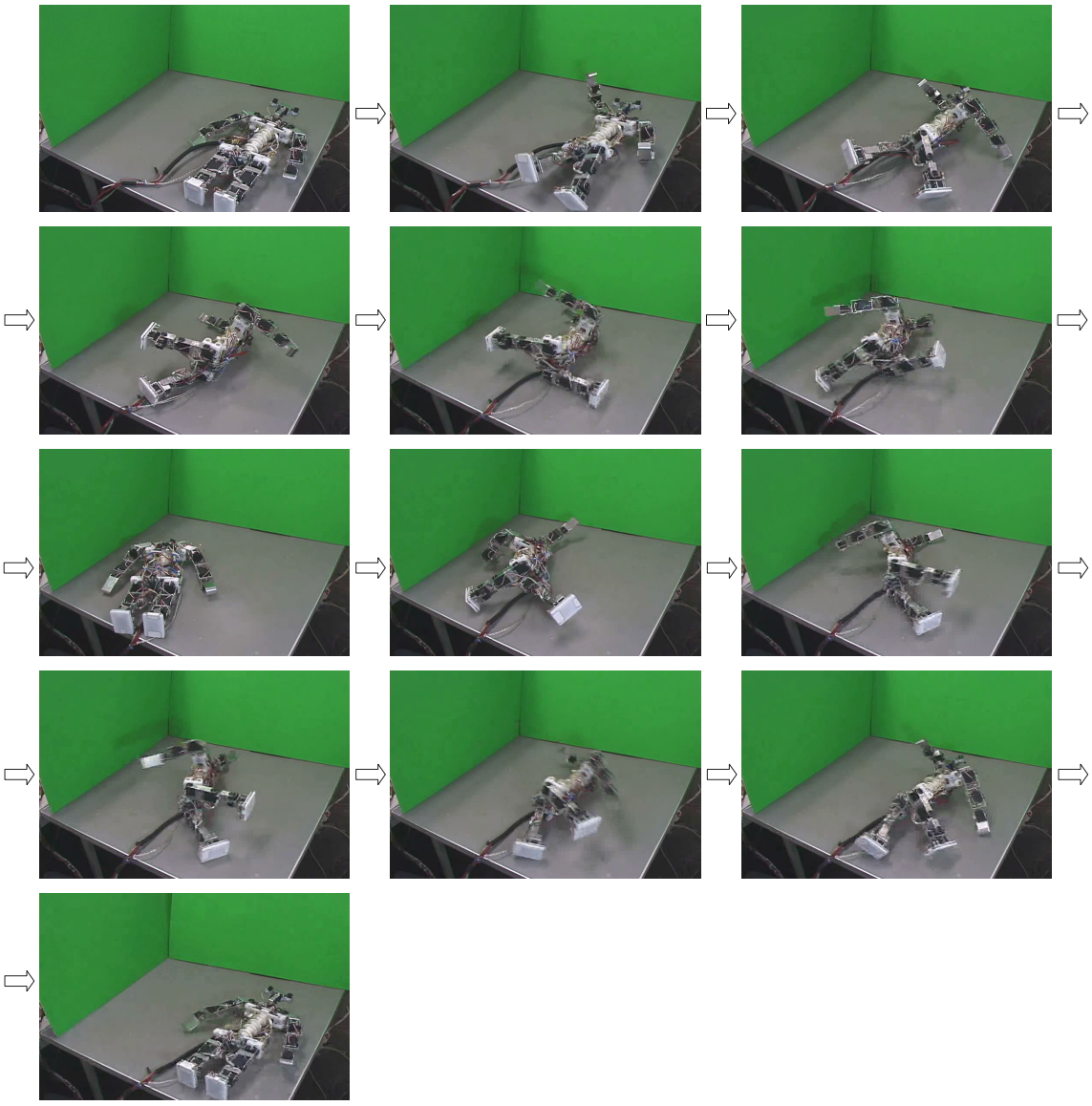


**Fig. 5.9** Cla の寝返り動作
Cla turning over on the floor

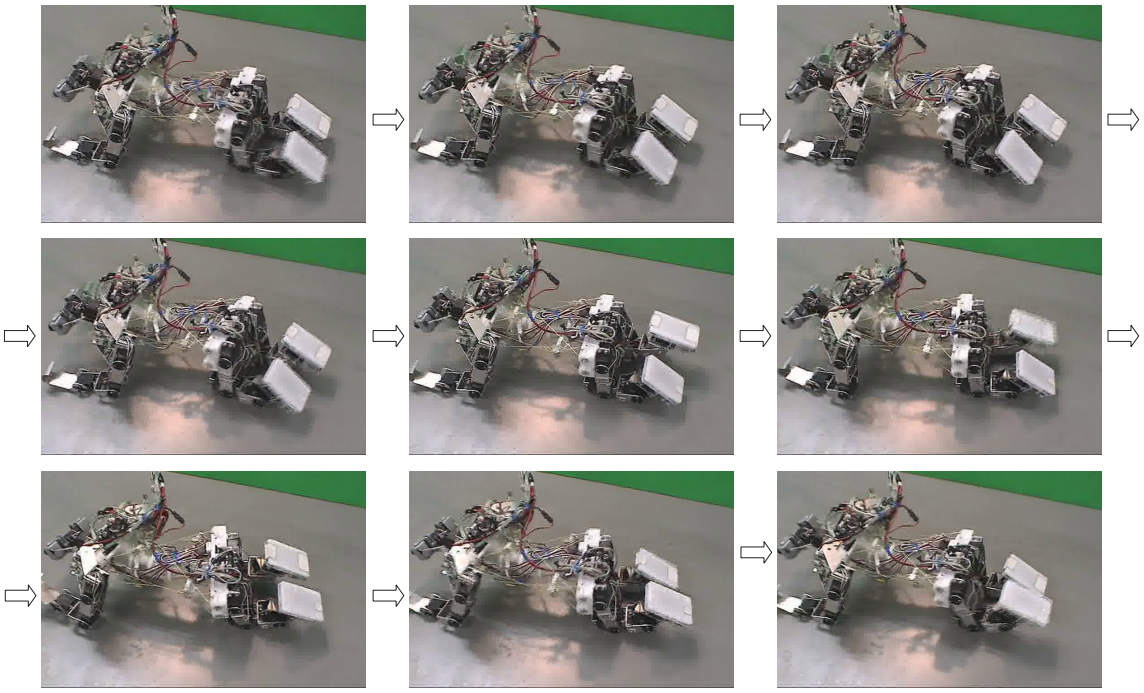


**Fig. 5.10** はいはいする Cla
Cla crawling

### 5.2.9 Cla による這い這い (はいはい) 行動

**Fig. 5.10** には， Cla がはいはいをする様子が示されている．うつぶせで腰・肩が床に完全に着いた状態から，腰と肩を持ち上げる時には，前項の寝返り動作のシークェンスの前半部分を利用している．寝返り動作と同様に，脊椎を捻る力が無ければ腰と肩を持ち上げることができない．腰と肩を持ち上げてからは，手足の動きと連動して脊椎を roll 軸回りに回し， 1 ストロークで進む距離を大きくする努力をしている．

ここまでの実験は全て人が動作パターンを EusLisp のモデル上で作成し，それを実ロボット Cla で実行していた．しかし，はいはいの動作は，床の摩擦の状況にも依るが，なかなか速く進む動作パターンを作ることができなかった．また，真っ直ぐに進まずに曲がって進む事も多かった．

そこで，はいはいの動作を GA により最適化し，少なくともシミュレーション環境の中でうまく動作できる動作パターンを探索することにした．シミュレーション環境には， 5.4.5節に述べる高速動力学演算ライブラリによる環境を利用した．

最適化するパラメータは脊椎の動きのみに絞り，手足の動作は常に同じ動きをして，脊椎の roll 方向の変形量と変形のタイミング (手足の動きは脊椎の変形タイミングにあわせて，ス

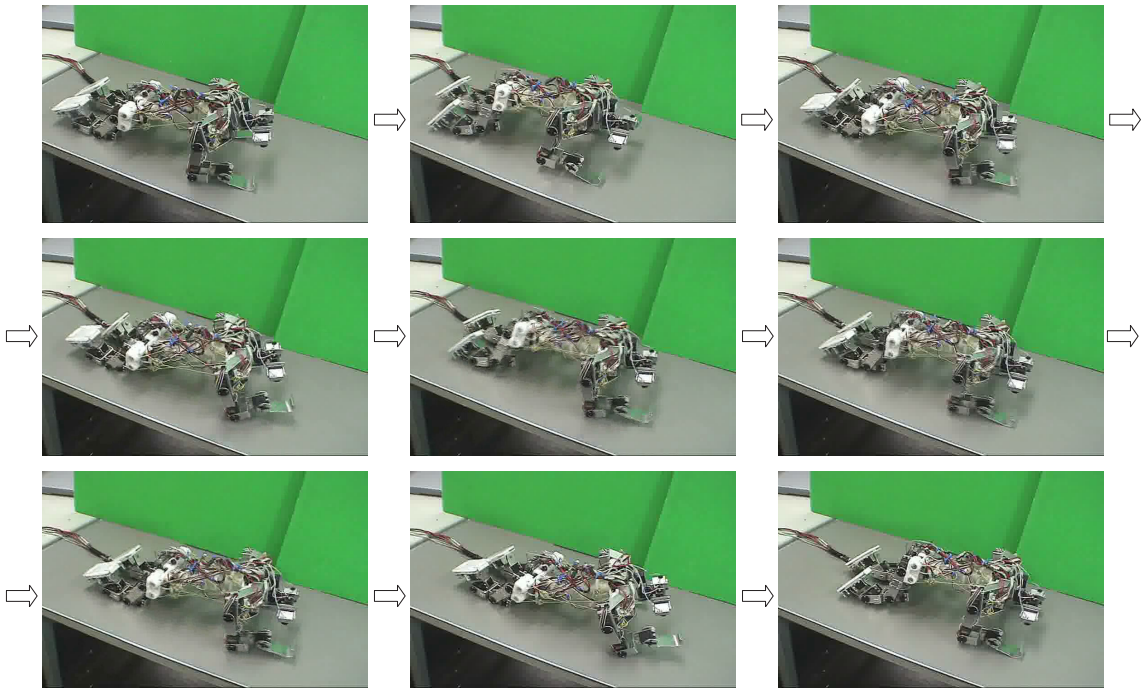


**Fig. 5.11** GA による Cla のはいはい動作の最適化
Cla crawling motion by GA

ピードが変わるようにした) を正規化し，これを遺伝子とし，(1) 前進量，(2) トルクの積分値，(3) 動作中の最大トルク，の三種類の評価関数の線形和で，fitness を求めるようにした．

GA により (少なくともシミュレーション環境用には) 最適化された動作パターンにより，はいはいを行う様子を **Fig. 5.11** に示す．

### 5.2.10 Cla による脊椎と首を協調させたトラッキング行動

**Fig. 5.12** には，Cla が脊椎と首を協調させて，視覚により対象物のトラッキング動作を行う様子を示す．視野中の対象物の位置を色領域検出により検出し，(1) 首の pan(yaw), tilt(pitch) の姿勢角をそれに応じて調整するルーチンと，(2) 脊椎の現在姿勢を現在の筋長から求め (4.4.3節)，対象物のずれに応じた角度だけ pitch 軸，yaw 軸回りに脊椎を変形する動作を行うルーチン，とを並列に実行し，(1)，(2) のゲインを調節することで実現した．

### 5.2.11 腱太による脊椎・首・両腕を同時に動かす行動

全身腱駆動型ヒューマノイド腱太 (第 3.7節参照) が，脊椎の動きと首の動き・両腕の動きを同時に行っている様子を **Fig. 5.13** に示す．

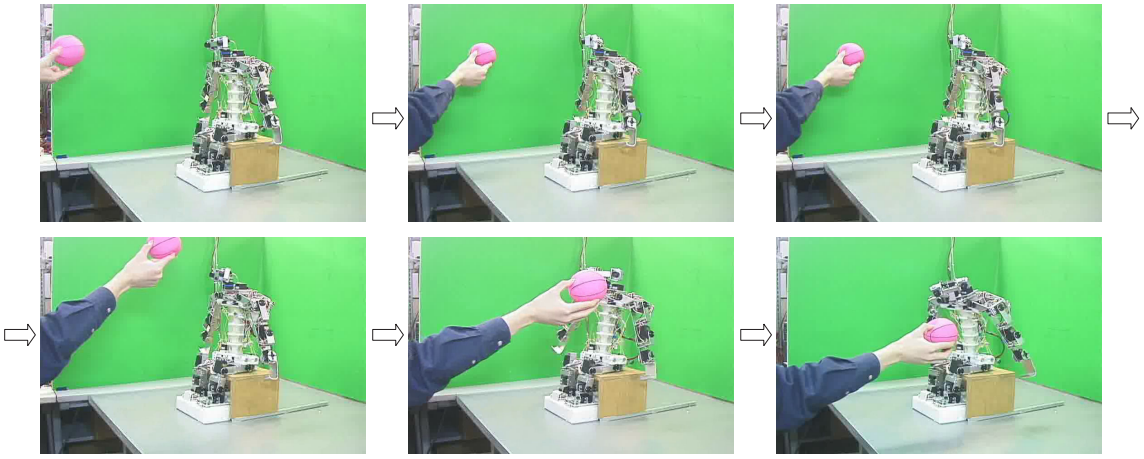


**Fig. 5.12** Cla の脊椎と首を協調させた視覚によるトラッキング動作
Cla tracking an object by visual feedback using spine and neck cooperatively

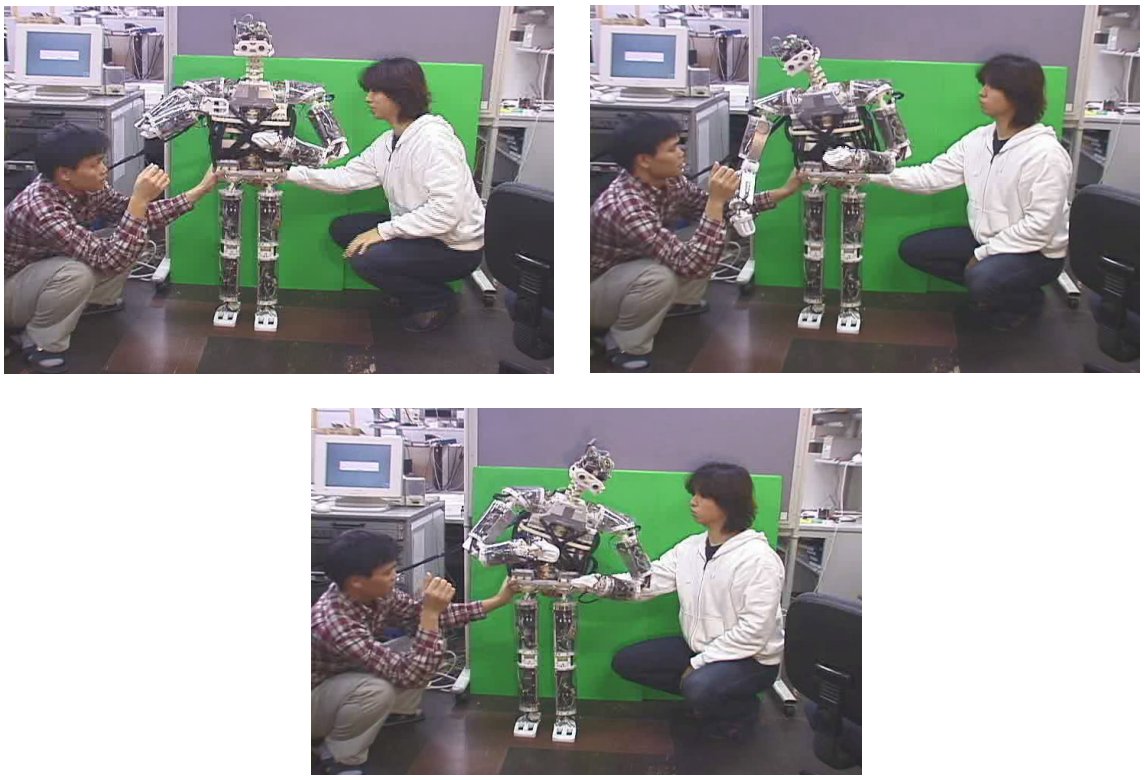


**Fig. 5.13** 腱太による脊椎・首・両腕を同時に動かす動作
Kenta's motion using the spine, neck and arms

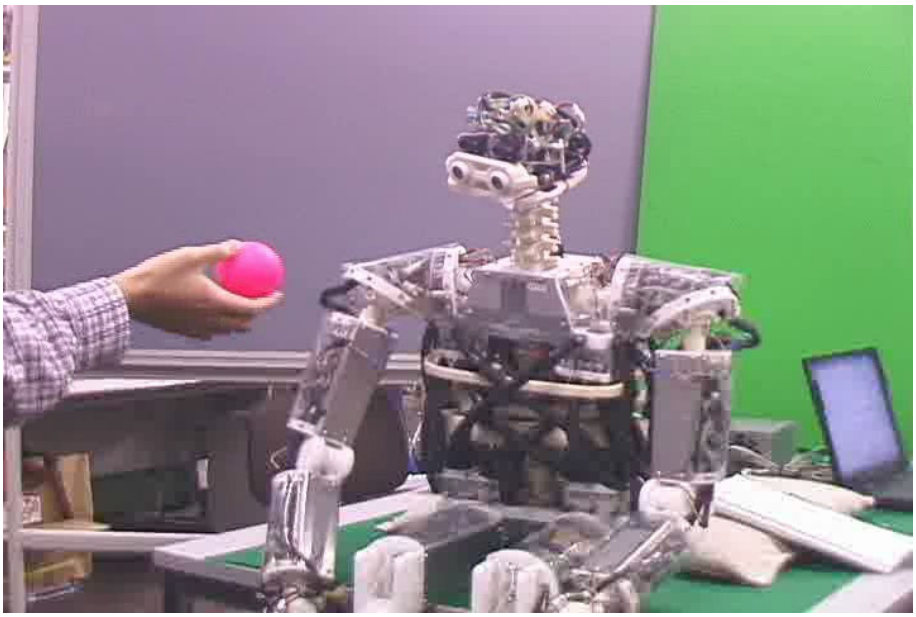


**Fig. 5.14** 腱太による眼球，首，脊椎を協調させた視覚によるトラッキング動作
Kenta's tracking an object using vision by coordination of eye-balls, neck, and spine.

### 5.2.12 腱太による眼球，首，脊椎を協調させた視覚によるトラッキング動作

腱太が眼球，首，脊椎を協調させた視覚によるトラッキング動作を行う様子を， **Fig. 5.14** に示す．眼球・首・脊椎をそれぞれビジュアルフィードバックで動かす周期の異なる 3 つの制御ループを並列に実行することにより実現している． 5.2.10節の行動を拡張する形である．

## 5.3 全身行動の自動生成法

ヒューマノイドなどの全身行動は，上肢のみあるいは下肢のみではなく，上肢・下肢を複合的に作用させるような行動である．こうした動作・行動を自動生成するための方法論に関する研究は，これまでにいくつか行われてきている．本節では，柔軟構造を持たない従来型のヒューマノイドなどにおける全身行動の生成と，柔軟構造を持つ全身型ロボットにおける全身行動生成法に関しに述べる．

### 5.3.1 柔軟構造を持たないロボットの全身行動の自動生成

ひとつの方法は，目的とする全身行動にある程度近い動作シークエンスを何らかの方法により作成し，それを雛形としてそれを元に力学的整合性を含めてロボッとが物理的に動作可能なシークェンスに適応化することにより，全身運動を実現するという手法である．金広らは，雛形の作成をコンピュータモデル上でのロボットの姿勢をいくつかピックアップして，各姿勢間の初期時間間隔を運動の概略記述というかたちで与え，遺伝的アルゴリズム (GA) と動力学シミュレータを用いて力学的整合性のとれた動作を獲得した [29, 32]．長阪らは，モーションキャプチャにより得られた実際の人間の運動パターンを元に雛形を作成し，同じく遺伝的アルゴリズムと動力学シミュレータを用いて実機で動作可能な運動パターンを生成した [67]．人間の動作には，解析不可能な多くの知識が含有されており，モーションキャプチャにより人間から収集した動作パターンを初期状態として用いることは，その知識を利用する方法の一つであると考えることもできる．遺伝的アルゴリズムを用いる他にも，ニューラルネットや最急降下法等を利用することも考えられるが，ヒューマノイドの運動パターンの探索空間の広さを考慮して遺伝的アルゴリズムを採用したと考えられる．さらに，遺伝的アルゴリズム以外にも，最適勾配法と呼ばれるアルゴリズムを用いた動歩行パターンの生成も行われている [68, 70]．

上記の研究例では運動パターンは純粋に関節角度情報 (あるいは関節目標角度情報) の時系列シークェンスを生成するだけで，センサ情報に基づいた適応的な全身行動を生成することはできない．実際の動作時に，センサ情報に基づく補正を加えることにより，より適応的な全身動作にすることは可能だが [69]，センサ情報と動作情報がより密に結合した全身行動も考えられる．センサ情報とアクチュエータ情報がより密に結合した適応的な全身行動を生成する方法の例としては，反射行動の組合わせで全身行動を生成するという手法 [55] がある．この手法では，全身行動をいくつかの反射行動要素に分けて，それらを重ね合わせることにより全身行動を実現している．全身行動を，多入力・多出力のシステムの入出力を決定する問題と捉えれば，全入力空間から全出力空間への写像を決定する際の組合わせの爆発を回避するための手段の一つであるとも考えられる．

### 5.3.2 柔軟構造を持つロボットの全身行動の自動生成

柔軟構造を持つ全身型ロボットの全身行動は，柔軟構造を持たないロボットの全身行動の生成と同様の方法の適用が容易ではない．その違いは，柔軟構造を持たないロボットの場合は実機との間の誤差の非常に小さい動力学シミュレータをできるが，柔軟構造を持つロボットの場合はそこに困難があるという事である．すなわち，柔軟構造部分の挙動を数式化して予測するシミュレータが現在のシミュレーション技術では困難である．ある入力 (アクチュエータ情報) を与えたときに，どのような出力 (柔軟部分を含むロボットの挙動) が得られるかをコンピュータの中で導き出せないため，遺伝的アルゴリズムや最急降下法などの適用ができない事になる．

## 5.4 シミュレーション環境の構築

### 5.4.1 シミュレーションを導入する理由

柔軟構造を持つロボットの全身行動の生成においては，剛体を仮定した従来のヒューマノイドの歩行動作などのように解析的に動作を生成することが困難である．なぜなら，柔軟性を持つ部分の挙動に関して完全に数式化を行うのは不可能に近いからである．実機における挙動との差が許容できる程度に小さい近似を行う方法が確立されれば，解析的に動作を生成することも可能であるが，現時点ではそこまで高精度の近似は難しいと考えられる．

そのため，柔軟構造を持つロボットの動作生成においては，どうしても実機を用いた繰り返し実験 (try and error) を行わざるをえなくなる．しかし，多様な全身行動を生成するにあたり全ての動作に関して実機を用いた試行錯誤的手法に依るのは非現実的であり，動作生成を自動化するためのなんらかの方法論が求められる．また，落下動作などの，外部からの衝撃が大きく実機で動作を行うと故障などの危険があるような動作の生成および解析を行う場合には，実機を用いずに行うことが求められる．さらには， 3.2.3節で述べたように，機構や形状を様々に変更してその挙動を確認したいなどといった場面も多い．こうしたことから，本研究では，柔軟構造を持つロボットの全身行動の生成においては，シミュレーションの利用は不可欠であると考えた．

いわゆる動力学解析シミュレーションは大抵は剛体のリンクを明確な固定関節軸により拘束したモデルを用いている．弾性や粘性を持つ構造材のモデルを用いないのは，構造材の柔軟性を仮定するとシミュレーションの計算量が膨大になるためである．柔軟性を持つ構造のシミュレーションには，有限要素法が用いられる場合が多い．有限要素法は弾性や粘性を持ち形状変形を伴うような構造の解析が可能だが，一般に計算量は非常に多い．従来の全身型ロボッ

トは，その身体構造も剛体リンクの仮定が充分適用可能な程度の剛性を有していた．本研究で扱うような，脊椎構造の椎間板などの構造を含む身体構造を有する全身型ロボットは，どのようなシミュレーション環境を用意すれば良いだろうか．

市販ソフトウェアなどの動力学解析シミュレータの多くは剛体リンクを仮定しているが，剛体リンク間の関係 (関節) に弾性・粘性の挙動を持たせることはできるものが多い．本研究で議論してきた脊椎のような構造の挙動をシミュレートする一つの方法は，こうしたリンク間の弾性・粘性のパラメータを利用することである．もう一つは，全身の中で脊椎部分のように柔軟性を有する部位の特性のみを有限要素法解析でパラメタライズし，それ以外の部分は剛体リンクのシミュレーションを用いるという方法がある．

また，柔軟性を有する構造は摩擦やヒステリシスがあることも多く，シミュレーションではどうしても実機の挙動を予測しきれない面がある．それにも関わらず，柔軟性を考慮に入れると計算量は膨大になり解析時間が非常に長くなりやすい．それならば始めから，実機の挙動との整合性はほどほどにして，計算量を減らした効率の良い高速動力学シミュレータなどを用いることも，実機で実験を始める際の初期軌道を求めるためというような用途に利用することを考えてよい．

本研究では，

1. 脊椎部分のみを有限要素法により解析し全身の挙動は剛体リンクを扱う動力学解析シミュレータにより解析する．
2. 動力学解析シミュレータを利用し，脊椎部分は関節の粘性・弾性を定義して解析する．
3. 高速動力学解析シミュレータを用いて，柔軟構造の挙動を考慮しない全身のシミュレーションを行う．

という 3 種類のシミュレーション環境を構築し，それぞれの特色を活かす用途を示す． 3. より 1. の方が正確性が高いが，計算時間は非常に長い． 2. は正確性・計算時間とも 3. と 1. の中間に位置づけることができる．

### 5.4.2 有限要素法を用いた体幹部の挙動のシミュレート

#### 有限要素法

柔軟部分の挙動の完全なモデル化は困難であるが，何らかの方法で近似的にでも挙動を予測しシミュレーションを行うことが必要である．本研究では，柔軟部分の解析のための方法としていくつかの方法を試行した．ここでは，そのうちの一つである有限要素法によるシミュレーションに関し述べる (3.2.3節参照)．有限要素法を用いるのは， 3.2.3節に述べたように構

造解析の手法として一般に利用されているという理由があるが，もう一点として商用のソフトウェアの中には，ロボットの動力学解析ソフトウェアと組み合わせることが可能なものもあるという点にも着目した．実際に本研究で用いた市販ソフトウェアは，動力学解析ソフトウェア DADS(Dynamical Analysis and Design System)[7] [1)]と，有限要素解析ソフトウェア ANSYS[40][2)]である．

### 解析手法

大まかな手順は，

1. 柔軟部のみを有限要素法を使用して変位モードを求める解析
2. 柔軟部の有限要素モデルの組み込み
3. 柔軟部を含むロボット全体の動作の解析

となる．

まず手順の式 (5.1) において柔軟部の固有変位モードを有限要素法のモーダル解析を用いて求める．固有変位モードはその物体に変形が生じる際に現れる，その物体に特有の変位の仕方で，今回用いる手法では固有変位モードの線形和で柔軟部の変形が表現される．具体的には，求める固有変位モードの数 $m$ この解析で求められる固有変位モードを $\boldsymbol{\phi}_i$ $(i = 1 \ldots m)$，モード座標を $q_i$ とすると，ロボット全体の解析では柔軟部の変形 $\boldsymbol{u}$ は式 (5.1) で表現される．

$$\boldsymbol{u} = \sum_{i=1}^{m} q_i \boldsymbol{\phi}_i \tag{5.1}$$

手順の 2. では，手順の一番目で求められた柔軟部の有限要素モデルをロボット全体の機構解析のモデルに組み込む．最後に (3.) ロボット全体の動作を機構解析ソフトウェアを用いて動解析を実行することによりシミュレートする．

この手法の利点として，ロボット全体の解析の際に柔軟部に有限要素法を用いる解析手法と比較すると，柔軟物体を少数の変数で表現できるので計算資源の小型化，及び計算時間の短縮が図れることが挙げられる．一方，問題点として変位モードの線形和で表現された解析上での変形と実際の変形との誤差がある．これは例えば使用する変位モードの数が少ない場合などに起こりうる問題である．またこの手法ではそもそも材料学的あるいは構造的な非線形性を解析の中に組み込むことができないという問題がある．今回製作した脊椎では，柔軟性を持たせ

[1)] http://www.cadsi.com/Software/

[2)] http://www.ansys.com/

るためにメモリーフォームを脊椎の部品として使用しているが (3.2.1節参照)，この材料の粘弾性項が挙動に対して支配的な場合には誤差が生じる可能性がある．従って，この部分をより忠実に再現しようと考えると，各変位モードに対して各々適切な減衰項を設定することにより，擬似的に材料学的非線形性を表現する手法を採ることも考えられる．但し減衰項を設定すると計算時間が大きくなるので，この手法のメリットを失うことになる．

### 5.4.3 動力学解析シミュレータと有限要素法ソフトウェアの統合

本研究では実際のシミュレーションを行うソフトウェアとして市販のソフトウェアを利用する．複数の市販のソフトウェアを統合する環境のみを作成することにより，全てのソフトウェアを作成する場合と比較すると迅速に，しかも容易にシミュレーション環境を構築，利用することが可能になる．また環境自体は新たに作成するので，これまで自分で作成してきた既存のソフトウェア資産との統合も容易に行うことが可能である．

本研究で利用する環境では以下のソフトウェアを使用して，シミュレーション環境を構築する．

- 三次元幾何モデル及び動作の作成 - EusLisp[48, 50]
- 有限要素解析用ソフトウェア - ANSYS[40]
- 機構解析用のソフトウェア - DADS(Dynamical Analysis and Design System)[7]
- 全体の環境の統合 - EusLisp

また，構築した環境の全体の流れを **Fig. 5.15** に示す．

#### 幾何モデル及び動作生成環境としての EusLisp

EusLisp 上での作業は解析を行うロボットの 3 次元幾何モデル及びそのロボットの動作を作成することである．実際にロボットを動作させる際に利用するシステムも EusLisp を使用しており (第 6章参照)，シミュレーション及び実機の動作の生成に共通の環境が利用できる．したがって実機の動作をシミュレーションで確認する，あるいはシミュレーションから生成した動作を実機で試行する，などの作業が効率的に実行することができるという長所もある．ここでの EusLisp 上で行う作業は作成したロボットの 3 次元幾何モデルデータ及び動作のトラジェクトリデータを DADS 用の形式に変換するところまでである．

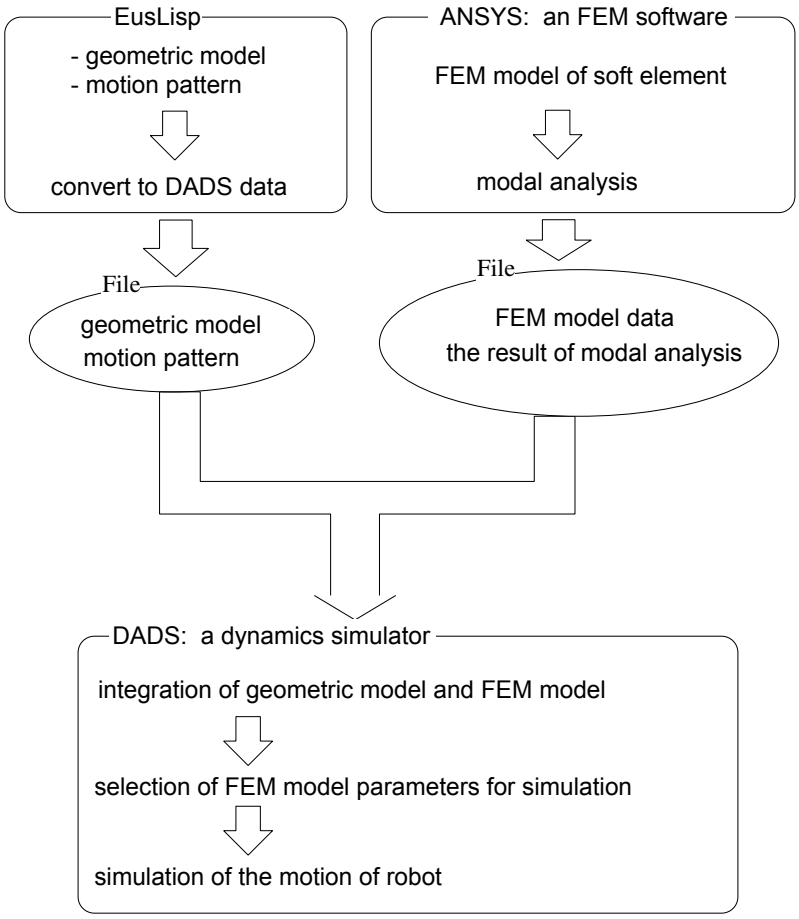



**Fig. 5.15** DADS と ANSYS の統合によるシミュレーション環境
The simulation environment by the integration of ANSYS and DADS

### 脊椎の特性解析のための ANSYS

ANSYS 上での作業は柔軟部のモーダル解析を行い，固有変位モードを求めることである．まず柔軟部として取り扱う部位の有限要素モデルを作成する．この作業に先んじて，どの部分を柔軟部として扱うかを決定する必要がある．これはロボットの機構や，実際の挙動等を観察することにより決定する．作成した有限要素モデルを用いてモーダル解析を実行し，固有変位モードを求める．この解析で求める固有変位モードには，

- 静的変位モード
- 動的変位モード

の二種類が存在し，前者を求める解析と後者を求める解析は別個に行う必要がある．

静的変位モードは物理的には以下のように説明される．物体内のある一節点に関して x, y, z 各方向の位置及び回転の合計 6 自由度に関して拘束を加える．この時，他の節点にある方向の単位応力または単位モーメントが加わった際の変形を静的固有変位モードと定義する．具体的には式 (5.2) の $\boldsymbol{\psi}_a$ で表現される．

$$\boldsymbol{K}_a\boldsymbol{\psi}_a = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{O} \end{pmatrix} \tag{5.2}$$

$$\text{ただし} \begin{cases} \boldsymbol{K}_a & : \text{有限要素モデルの剛性マトリクス} \\ \boldsymbol{\psi}_a & : \text{静的固有変位モード} \\ \boldsymbol{I} & : \text{力を表す単位行列} \end{cases}$$

結果として，ある節点に関する固有変位モードは 6 個求められることになる．ここで求められる 6 個の固有変位モードの重ね合わせを利用することにより，任意の方向の外力 (含むモーメント) に対する変形を求められることになる．

一方，動的変位モードはモデルの振動モードを表現している．すなわちあるモデルに自然振動が生じる際に，その振動周波数に応じて現れる変形が動的変位モードと定義され，具体的には式 (5.3) の$\boldsymbol{\varphi}$で表現される．

$$(\boldsymbol{K}_e - \lambda\boldsymbol{M}_e)\boldsymbol{\varphi} = \boldsymbol{O} \tag{5.3}$$

$$\text{ただし} \begin{cases} \boldsymbol{K}_e & : \text{物理剛性行列} \\ \boldsymbol{M}_e & : \text{質量行列} \\ \lambda & : \text{固有値} \\ \boldsymbol{\varphi} & : \text{固有ベクトル} \end{cases}$$

### 全身の挙動解析のための DADS

DADS 上の作業は主に柔軟部を含むロボット全体の挙動をシミュレートすることである．まずロボットの 3 次元幾何モデルに柔軟部の有限要素モデルの組み込みを行う．つまり， EusLisp から変換したロボットのモデルに含まれる柔軟部を ANSYS から変換した有限要素のモデルに変更する．この際，柔軟部の有限要素モデルのどの節点と機構解析のモデルのどの点を接続するか，また接続する際にどのような拘束条件を与えるかが重要になる．

次に柔軟部においてロボット全体の解析で使用する固有変位モードを選択する．この際に良好な解析結果を得るためには変位モードの選択が重要となるが，選択するべき固有変位モードはロボットの形状や動作，ロボットに加わる外力の大きさや方向等の解析条件に依存するため，一般的な選択手法は存在しない．そこで，まずどの固有変位モードを使用するかを見積もるために，柔軟部を剛体と仮定したロボットで動作の解析を行い，その挙動を観察することでどの固有変位モードの係数が大きくなるかをあらかじめ予測しておく．その結果を元に，固有変位モードを選択することで適切に解析を進めることが可能になる．

上記の作業を行った後に，ロボット全体の動作の解析を実行する．

### 環境構築のための EusLisp

上記の 3 つのソフトウェアを統合する環境を構築するためのソフトウェアとしても EusLisp を使用する．この役割のために EusLisp を用いるメリットとして，

- 幾何モデル部との親和性
- 外部プログラムの起動しやすさ
- データ，ファイルの処理能力

などが挙げられる．とくに三番目の項目に関しては，例えば学習による最適化を行う場合に，結果ファイルから評価値の計算などにおいて，わざわざ外部プログラムを起動せずに環境自身が計算することで余計な手間を省くことができる．なお実際の環境では前に述べた「幾何モデル及び動作生成環境としての EusLisp」とこの節で述べる EusLisp は同一のプロセスとなっており，必要に応じて DADS 及び ANSYS らの外部プログラムを立ち上げている．なお DADS， ANSYS ともに Solaris 及び WindowsNT の両方の環境で実行することが可能であり，必要に応じて EusLisp が動いているマシンも含めて複数のマシンから選択して解析を実行する (5.5.2節参照)．

### 5.4.4 市販の動力学解析シミュレータの修正による動的な剛性調節のシミュレート

5.4.2節の有限要素法ソフトウェアと動力学解析シミュレータを利用した環境は，(1) 計算量が多く計算時間がかかる，(2) 動作中に動的に柔軟性を変更するような動作の解析ができない，などの問題があった．本節では，これらの問題を解決し，幾分シミュレーションの精度の悪化は伴うかもしれないが，動作中の柔軟性の動的変更も含めて，ロボット全体の挙動を解析できるシミュレーション環境の構築と，実機の動作データを元にした体幹部が硬い場合と軟らかい場合の衝撃力の比較に関し述べる．さらに，実機の実験 [54] の際の動作と同様に動作中に柔軟性を変更した場合の挙動に関してもシミュレーションを行った [57, 56]．柔軟性を持つ部分の解析には，剛体リンク間の関節に弾性要素を組み込み (5.4.1節参照)，柔軟性に関するパラメータを動的に変更可能にするための市販ソフトウェアの改造を行うことで，可変な柔軟構造の脊椎を持つ全身型ロボットの挙動をシミュレートできるようにした．

もともと DADS には BEAM 要素，FORCE 要素などの弾性体や力要素のモデルがある．BEAM 要素は HanzouS(第 3.3節) の脊椎のモデル化に利用できると思われるが，BEAM 要素に定義できる弾性係数やヤング率などの物理定数は固定であり，HanzouS の脊椎のように動作中に柔軟性を変更することができない．筆者は DADS の開発元である CADSI Inc.[3)] に問い合わせ，これらを動的に変更する方法を相談した．その結果，BEAM 要素を定義する剛性行列の各要素を直接変更する方法に関する助言を受け，DADS のソルバ (FORTRUN で記述) のソースを修正し，動作中に動的に剛性行列を更新するインタフェースを新たに開発した．修正したソースを再コンパイルしてできた修正済ソルバを用いて解析を行うことにより，動作中の柔軟性調節をシミュレートできる環境を構築した ( **Fig. 5.17** 参照)．

### 5.4.5 ゲーム用高速動力学演算ライブラリを利用したシミュレーション環境

5.4.2節～ 5.4.4節に述べた手法の問題は，計算時間がかかることである．いずれの手法でも実時間のシミュレーションは不可能である．精度が幾分悪くとも実時間並みの計算速度でシミュレーションを行うことができる環境が有効な場面もある．例えば，GA による動作生成 (第 5.5節) を行うような場合，5.4.3節の手法によりシミュレーションと探索を行うと収束に数日かかるような場合でも，高速な動力学演算ルーチンを利用すれば数時間で収束するかも知れない．脊椎構造を持つ全身型ロボットの場合，いずれにせよ得られた動作軌道を実機にそのまま適用しても，うまく実機で動作が可能でない場合は多いが，おそらく前者の手法 (精密・計算量多) により得られた結果の方がより実機で動作可能な解に近い動作軌道が得られることが期待できるが，逆に，後者の手法 (精度低・高速) を用いることで得られた動作軌道は前者の

[3)] http://www.cadsi.com/ ． 2001 年 12 月現在，LMS INTERNATIONAL ( http://www.lmsintl.com/ )

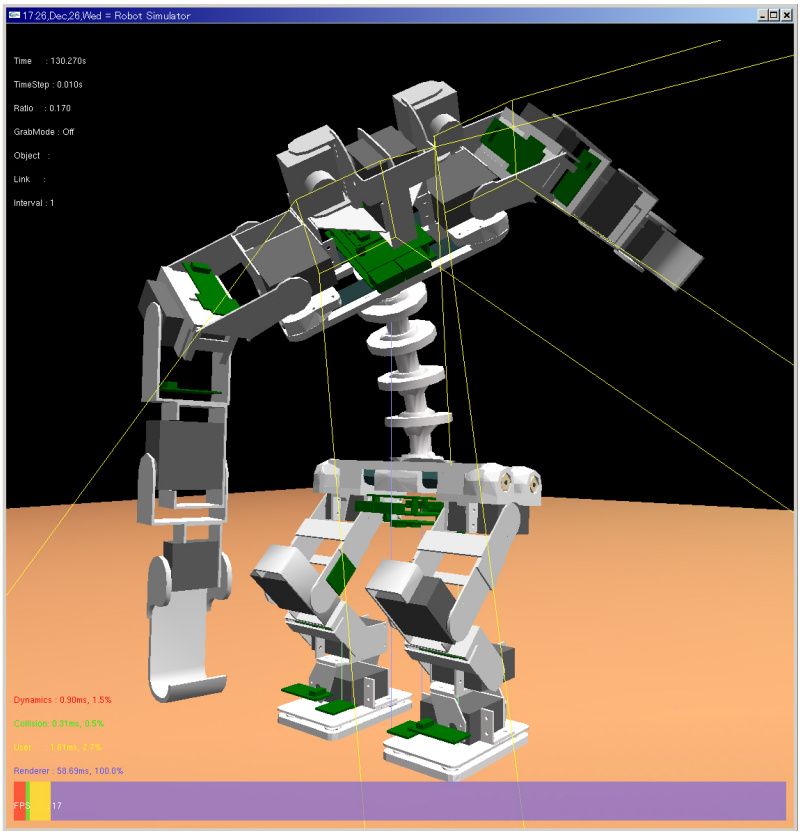



**Fig. 5.16** FAST 上の Cla
Cla in the FAST

結果よりは実機で動作可能な解から遠いかも知れないが，最初から人間が数時間試行錯誤するよりは解に近い結果が得られるかも知れない．そうすれば，高速な環境を利用して解から少し遠い軌道が得られたとしても，そこから人間がパラメータ調節をする事で，精密な環境を利用して数日かけるより早く実機での動作が実現できることを期待することができる．実際，DADS で力要素を組み込まず動力学計算のみを行った場合でも，18 自由度のヒューマノイドロボットが地面から起き上がる約 6 秒間の動作のシミュレーションに UltraSparc II 300MHz の CPU を搭載した計算機上で約 3 分間かかっている例もある [30]．また，制御アルゴリズムの開発の初期段階や，高次の行動制御ソフトウェアの開発時などにも，正確性よりも高速性が要求される場面はあると考えられる．

こうした考察から本節では，正確性より高速性を重視した動力学演算ソフトウェアを利用したシミュレーション環境の構築に関し述べる．本研究では，同様の観点から金広らが開発した [33, 34] 高速シミュレーション環境 FAST(FAst Simulation environmenT) を，脊椎構造を持つロボットに適用可能にするための拡張を行うことで，高速なシミュレーション環境を開発

した． FAST は Critical Mass Labs 社[4)]のツールキット Vortex を利用している． FAST ではこれを利用し，回転関節と剛体から成る全身型ロボットの解析のためのインタフェースを整えてある．本研究では，これに脊椎の 3 自由度球面関節のクラスとそれを扱うためのいくつかのクラスを追加することで，脊椎を持つ全身型ロボットのシミュレーションを行うことができるようにした．また，シミュレーション環境のインタフェースは，実ロボットを扱う際の，アクチュエータ目標角度・センサ (3 軸加速度センサ・足裏力センサ・視覚センサ等) の入出力と同じインタフェースを備え，シミュレーション環境で生成あるいは検証した動作を，そのままのソフトウェアでインタフェースを切り替えるだけで実ロボットに適用することができる．

**Fig. 5.16** に Cla(第 3.6節参照) をこの環境で動かしている様子を示す．

## 5.5 GA を用いた柔軟構造を持つロボットの動作生成

### 5.5.1 柔軟構造を持つロボットの動作生成に GA を採用する理由

5.3.1節で述べたように，従来のヒューマノイドなどの動作生成の研究においては，動力学解析と探索手法・最適化手法を組み合わせた手法により全身行動を生成している例がある．探索手法・最適化手法には，遺伝的アルゴリズム (GA)，ニューラルネットワーク，最急降下法，最適勾配法などが用いられているが，本研究では，柔軟構造を持つロボットの動作生成における探索手法として GA を採用する． GA 以外の最適化手法・探索手法では，探索のステップ毎に次に最適化方向を示すベクトルを求めなければならない．そのためには各ステップにおける最適化手法・探索手法により調整されるパラメータと，評価関数の間の関係を数式化する必要がある．例えば，最適勾配法 (Optimal Gradient Method)[70] の手法では，調整するパラメータベクトルが探索の $k$ 番目のステップにおいて

$$q^{(k)} = (q_1^{(k)} q_2^{(k)} \cdots q_N^{(k)})$$

で，その時の評価関数が，最適化したい値 $p$ の目標値を $p_{ref}$ として

$$E(q^{(k)}) = \int (p - p_{ref})^2 dt$$

だとすると， $k+1$ 番目の探索点を，

$$q^{(k+1)} = q^{(k)} - \alpha^{(k)} \nabla E(q^{(k)})$$

などのように求める．すなわち評価関数 $E(q^{(k)}$ の勾配 $\nabla E(q^{(k)}$ を求めることが必須である．

[4)] http://www.cm-labs.com/

しかし，柔軟構造を持つロボットの場合，ほとんどの場合調整されるパラメータと評価関数との関係が明確ではない．そのため，探索ステップ毎のパラメータの調整量を計算することが難しい．こうしたことから， GA やランダムサーチ等の，探索ステップ毎のパラメータの変更量を，評価関数との関わりに無関係に決定することのできる探索アルゴリズムを用いることが有効となる．本研究では， GA と第 5.4節で述べたシミュレーション環境を組み合わせた手法を開発し，柔軟構造を持つ四脚ロボットのトロット歩行のパラメータの最適化や，柔軟性可変な構造を持つ人間型ロボットのブラキエーション動作の生成において，これを適用した．

### 5.5.2 EusLisp による並列 GA 環境の構築

本研究で開発した 3 種類のシミュレーション環境 (第 5.4節) のうち，有限要素法を利用するもの (5.4.3節) と動力学解析シミュレータのソルバを改造するもの (5.4.4節) は，比較的計算時間がかかる．一方 GA のプロセスにおいては，動作パターンやパラメータが少しずつ変えて， GA の個体数に対応する多数の動作の解析を行う必要があるが，各個体に対応した解析は相互に完全に独立であるので， GA の 1 世代を構成する個体数分の並列化がアルゴリズム的に可能である．そこで， GA による探索の高速化のために， GA の各個体に対応する解析を並列に行う環境を構築した． **Fig. 5.17** は， 5.4.4節に述べた DADS のソルバを修正して動作中に柔軟性調節を行える環境を利用した並列 GA のソフトウェア構成を示す． 5.4.3節に述べた有限要素法と動力学解析シミュレータを組み合せた環境を利用する場合は，図の Solver(DADS) の部分において有限要素解析の結果を利用することになる ( **Fig. 5.19** 参照).

生殖・淘汰・交差・突然変異などの GA のオペレーションを行うサーバを EusLisp 上に置き， GA の個体数以下のできるだけ多くのクライアントを生成し，サーバから各個体の遺伝子に対応した動作パターンやパラメータを各クライアントに送る．各クライアントは，サーバから与えられた指示に従い，動力学解析エンジン ( **Fig. 5.17** では solver(DADS)) を起動し，解析エンジンはその個体に対応した動作の解析を行い，クライアントが GA の評価値 (fitness) を計算するための，センサ情報などの解析から得られる情報をクライアントに与える．各クライアントは解析エンジンから得た情報を元に評価関数に基づいて評価値を計算し，それをサーバに返す．サーバは，このようにして集まった各遺伝子に対応する評価値に基づいて GA のオペレーションを行い，次の世代の遺伝子を計算し，再びクライアントに動作パターンやパラメータを送る．これを繰り返すことで動作パターンやパラメータの最適化を行う．

**Fig. 5.17** における解析エンジンである DADS は，コマンドインタプリタを備えているので，遠隔からの利用が可能である．また， OS は SPARC Solaris2.6 および WindowsNT4.0 に対応している．そこで， WindowsNT にフリーソフトウェアの rsh サーバや NFS サーバをインストールし，遠隔から DADS のコマンドインタプリタの利用や，モデル定義ファイルや解

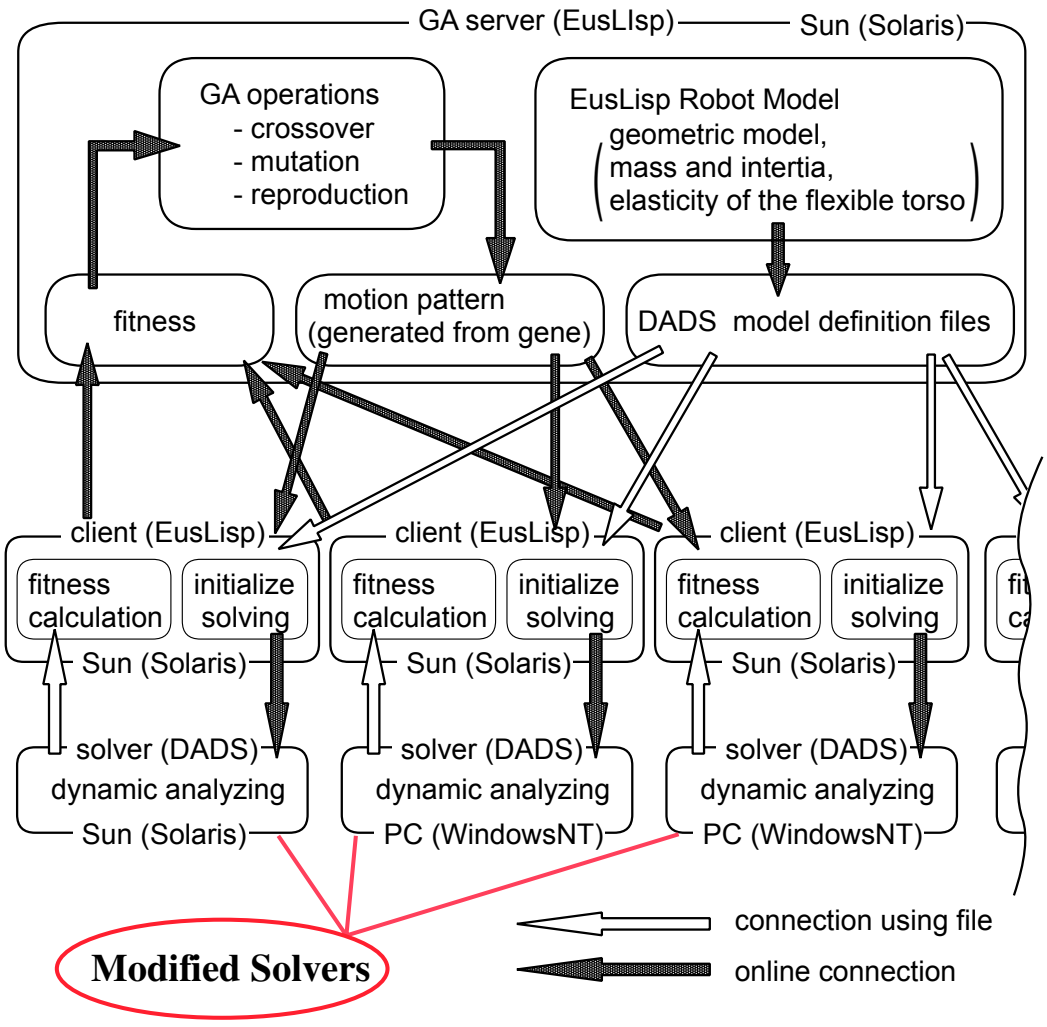



**Fig. 5.17** 並列 GA のためのソフトウェア構成
Software structure for parallel GA

析結果格納ファイルのアクセスをできるようにした．これにより，並列 GA のクライアントが利用する解析エンジンとして，多数の PC やワークステーションを利用することができるようになった．

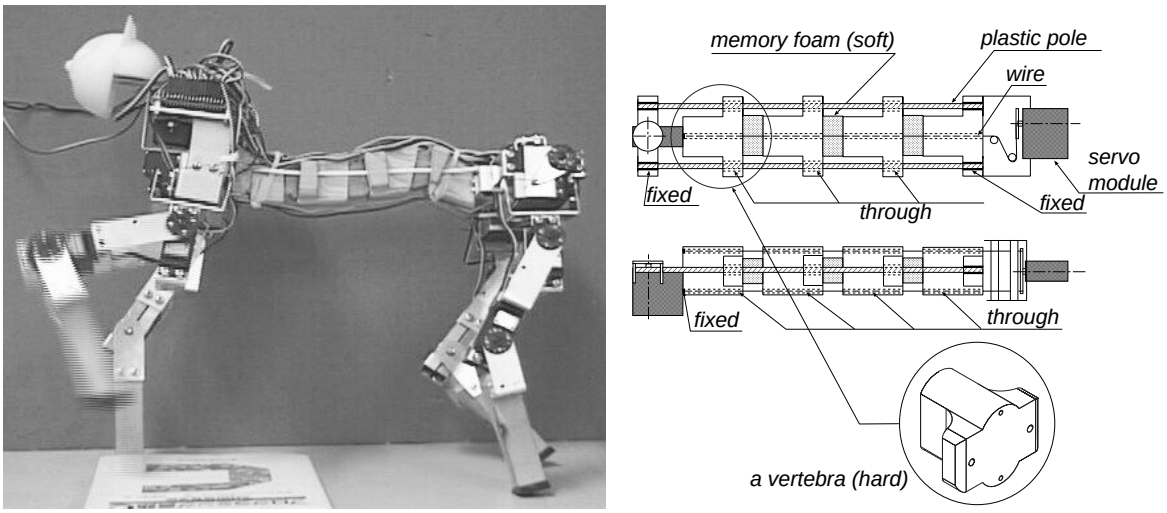



**Fig. 5.18** 脊椎構造を持つ四脚ロボット SQ43 とその脊椎機構
Flexible spine quadruped robot SQ43 and its spine structure

## 5.6 柔軟な脊椎を持つ四脚ロボット SQ43 のトロット歩行動作生成

### 5.6.1 ソフトウェア構成

シミュレーション環境は，5.4.3節に述べた環境によった． **Fig. 5.18** 右に示す非線形な柔軟構造の脊椎機構の部分の挙動は有限要素法ソフト ANSYS を用いて解析し，その結果を動力学解析ソフト DADS に取り込むことにより，全体の挙動をシミュレートできる環境を構築した．さらに，この環境を統一的に扱うソフトウェア環境を EusLisp[50] を利用して構築し，これを利用して GA による動作生成を行った．ソフトウェア構成を **Fig. 5.19** に示す．

各ソフトウェアはソケット通信またはファイルを介して接続され， EusLisp 上からコントロールする． EusLisp 上の GA サーバが， Solaris 及び WindowsNT 上で走る DADS のソルバを多数のクライアントとして生成し，並列に解析を行う ( **Fig. 5.17** 参照).

### 5.6.2 GA による歩行パターンの最適化

#### 遺伝子パラメータから歩行パターンへの変換

遺伝子パラメータから動作を生成する方法を説明する． **Fig. 5.20** は，各脚先位置を横から見た簡略図である．脚先位置の経路を DCBA の台形形状とし，以下のようにパラメータを設定した.

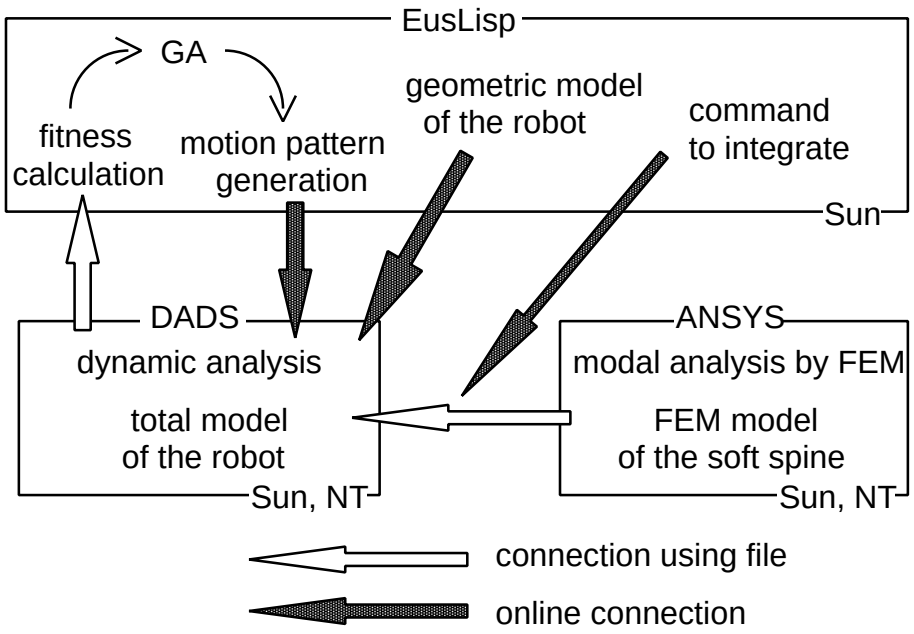



**Fig. 5.19** SQ43 のトロット歩行動作生成のためのソフトウェア構成
Software structure for generation of trot walking by SQ43

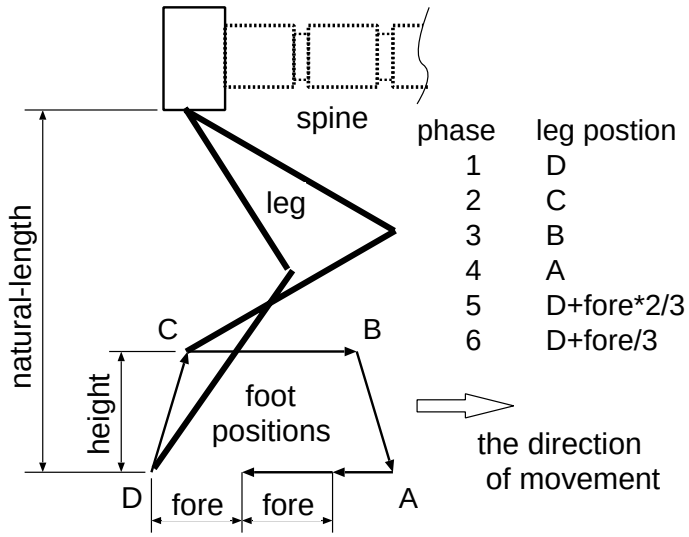



**Fig. 5.20** 歩行動作パターンの生成パラメータ
Parameters for trot walking generation

1. 全脚共通パラメータ

   **cycle** 1 サイクルの時間

   **fore** 1 サイクルで進む距離

   **natural-length** 脚が接地しているときの脚先と脚根本の鉛直方向の距離

   **height** 脚を前方に運ぶ際，脚を上方に引き上げる距離

   **duty** duty 比 (脚が接地している割合)

2. 各脚毎のパラメータ

   **start-time** 1 サイクルの内で何秒後に前方に脚を運ぶ動作を行うか

   **leg-center** 脚先の前後方向の中心位置

これらのうち， **cycle** (2.0 〜 4.0[sec]), **fore** (40.0 〜 80.0[mm]), **natural-length** (208.0 〜 228.0[mm]), **height** (5.0 〜 25.0[mm]), **leg-center** (前脚，後脚．それぞれ −30.0 〜 30.0[mm]) を GA の遺伝子に対応させ，これらのパラメータからトロット歩行動作パターンを生成した． **duty** は 0.5 に， **start-time** は 0.0, 1.5[sec] に固定した．

#### 適応度の設定

適応度は，解析結果から計算される前進量，ボディのねじれの平均，遊脚の高さの重みつきの和を用いた．それぞれの値はシミュレーションから得られる．

#### GA の結果

上に述べた条件で，遺伝子長 6，個体数 20，交差率 0.2，突然変異率 0.02，淘汰率 0.4 で GA を行った．解析結果を **Fig. 5.21** に示す．パラメータが多くないからか，それほど世代数をかけずに学習が収束している．解析時間は 1 個体の数秒間の運動の解析に約 30 分程度の時間がかかる．

### 5.6.3 実機でのトロット歩行実験

#### 実験

**Fig. 5.22** に，シミュレータ上での動作と実機の動作の様子を示す．実機では，シミュレータ上の動作と若干異り，脚を引きずりがちであり，また歩行の方向もシミュレータと完全に一致はしなかったが，歩行に成功した．

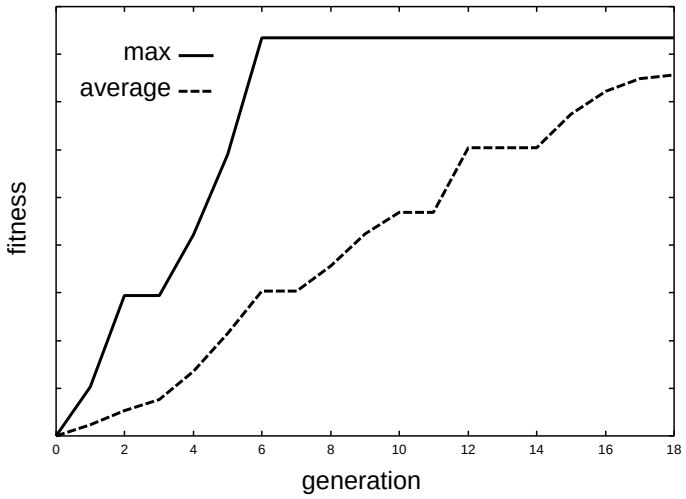



**Fig. 5.21** GA における世代数と fitness の関係
The relation between generation and fitness during GA

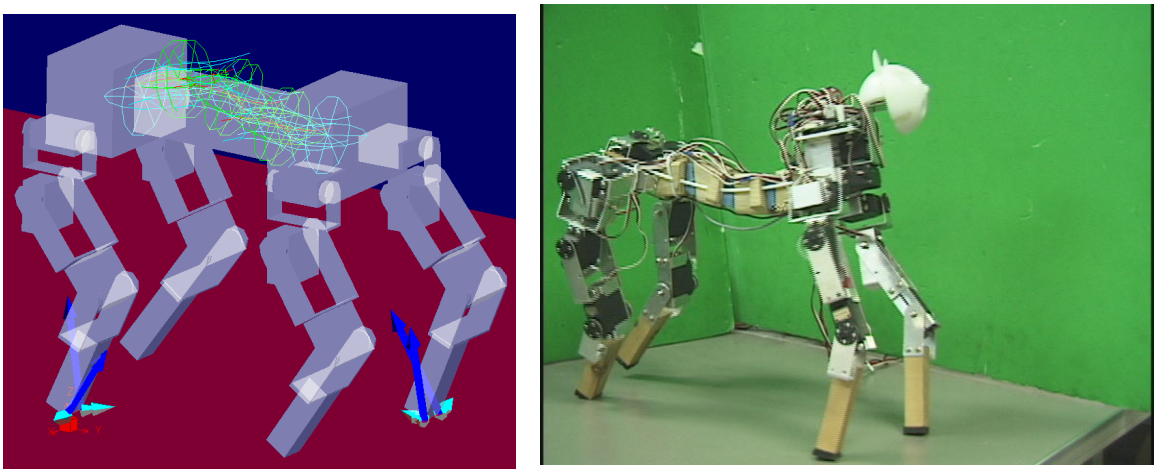


**Fig. 5.22** シミュレーションと実機の実験
Experiments on the simulation environment and real environment

**考察**

シミュレータを利用する場合，モデル化誤差を完全に無くすことは困難である．例えば今回のシミュレーションでは，床面と脚先の接触は DADS の接触モデルを利用しそのパラメータは試行錯誤で決めたが，実機の接触を忠実にモデル化しているとは言いきれない．シミュレータと GA 等の探索手法を利用して動作の生成を行うのは，本ロボットのように動作パターンの生成に解析的手法や試行錯誤を用いるのが困難な問題に対し有効である．目的の動作が可能な解にある程度近い動作パターンが生成された後は，実機を用いた適応的調整機能とセンサフィードバックを用いるべきであると考えられる．

## 5.7 可変な柔軟構造を持つ人間型ロボットのブラキエーション動作生成

### 5.7.1 ソフトウェア構成

シミュレーション環境は，5.4.4節に述べた環境を利用した．ロボットの幾何モデルを EusLisp[50] 上に構築するソフトウェア環境 [26] を拡張し，ロボットの形状・質量・慣性モーメントの他に，縦弾性係数やポアソン比といった柔軟構造の部分のパラメータを含む幾何モデルを，動力学解析シミュレーションソフトである DADS のモデルに変換するモジュールを追加し，柔軟構造を含む EusLisp 上のモデルを DADS の定義ファイルに変換する．

DADS の解析及び解析結果の解釈を EusLisp からオンラインで行うモジュールを作成した． EusLisp 上の GA の遺伝子から生成される動作パターンを自動的に DADS で解析し，結果を EusLisp 上で解釈し適応度の計算を行う． Solaris 及び WindowsNT で走る多数の DADS のソルバに，並列に解析を割り当てる並列 GA 環境を構築し，プラットフォームの異なる多数のマシンを計算エンジンとして，高速な探索を行う．

### 5.7.2 柔軟性の動的変更のシミュレート

柔軟性の動的調節は，5.4.4節の環境によった． 1 自由度の可変柔軟性を持つ脊椎の部分は DADS の BEAM 要素を利用し，弾性係数 $E$，ビームの長さ $l$ を動作中に動的に変更できるようにした．すなわち，DADS 内部で利用されている剛性行列 $\boldsymbol{K}$ の要素を更新するように FORTRUN のソースに修正を加えた．

$$\boldsymbol{F} = \boldsymbol{K}\ \boldsymbol{X} \tag{5.4}$$

$$\text{ただし,} \begin{cases} \boldsymbol{F} & : & \text{body1からbody2へ作用する力} \\ \boldsymbol{X} & : & \text{body1とbody2の相対変位} \\ \boldsymbol{K} & : & 6 \times 6\text{の剛性行列} \end{cases} \tag{5.5}$$

$$\boldsymbol{F} = (F_x, F_y, F_z, T_x, T_y, T_z)^T \tag{5.6}$$

$$\boldsymbol{X} = (P_x, P_y, P_z, \theta_x, \theta_y, \theta_z)^T \tag{5.7}$$

$$\boldsymbol{K} = \begin{pmatrix} \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12E\mathrm{I}_z}{l^3} & 0 & 0 & 0 & \frac{-6E\mathrm{I}_z}{l^2} \\ 0 & 0 & \frac{12E\mathrm{I}_y}{l^3} & 0 & \frac{6E\mathrm{I}_y}{l^2} & 0 \\ 0 & 0 & 0 & \frac{G\mathrm{I}_x}{l} & 0 & 0 \\ 0 & 0 & \frac{6E\mathrm{I}_y}{l^2} & 0 & \frac{4E\mathrm{I}_y}{l} & 0 \\ 0 & \frac{-6E\mathrm{I}_z}{l^2} & 0 & 0 & 0 & \frac{4E\mathrm{I}_z}{l} \end{pmatrix} \tag{5.8}$$

### 5.7.3 シミュレータ上におけるブラキエーション動作の生成

構築したソフトウェア環境を用いて，GA によりシミュレーション上のブラキエーション動作を生成した．3.3.3節に述べた実機のブラキエーション動作のシミュレーション上での再現である．

#### GA の設定

実機のブラキエーション動作実験 [54] の際のシーケンスを初期姿勢列として，各姿勢の関節の調整量と姿勢間の遷移時間を遺伝子として GA を行った [29].

適応度は以下の評価基準の重みつき和を用いた.

- 運動中の体幹部の鉛直方向の位置（高さ）の平均が大きいほど良い.
- 右手の接触反力がある.
- 両手の接触反力の変動が小さい方が良い.
- 体が前に進んだ距離が多いほど良い.

調節する関節としては，両手の肩・肘，片手のグリッパ関節，両足の股関節・膝関節の計 9 関節とし，初期姿勢列は 8 個の姿勢からなる列をスプライン補間したものとしたので,

$$(9\text{関節の調節量} + \text{時間ステップの調節量}) \times 7\text{ステップ}$$

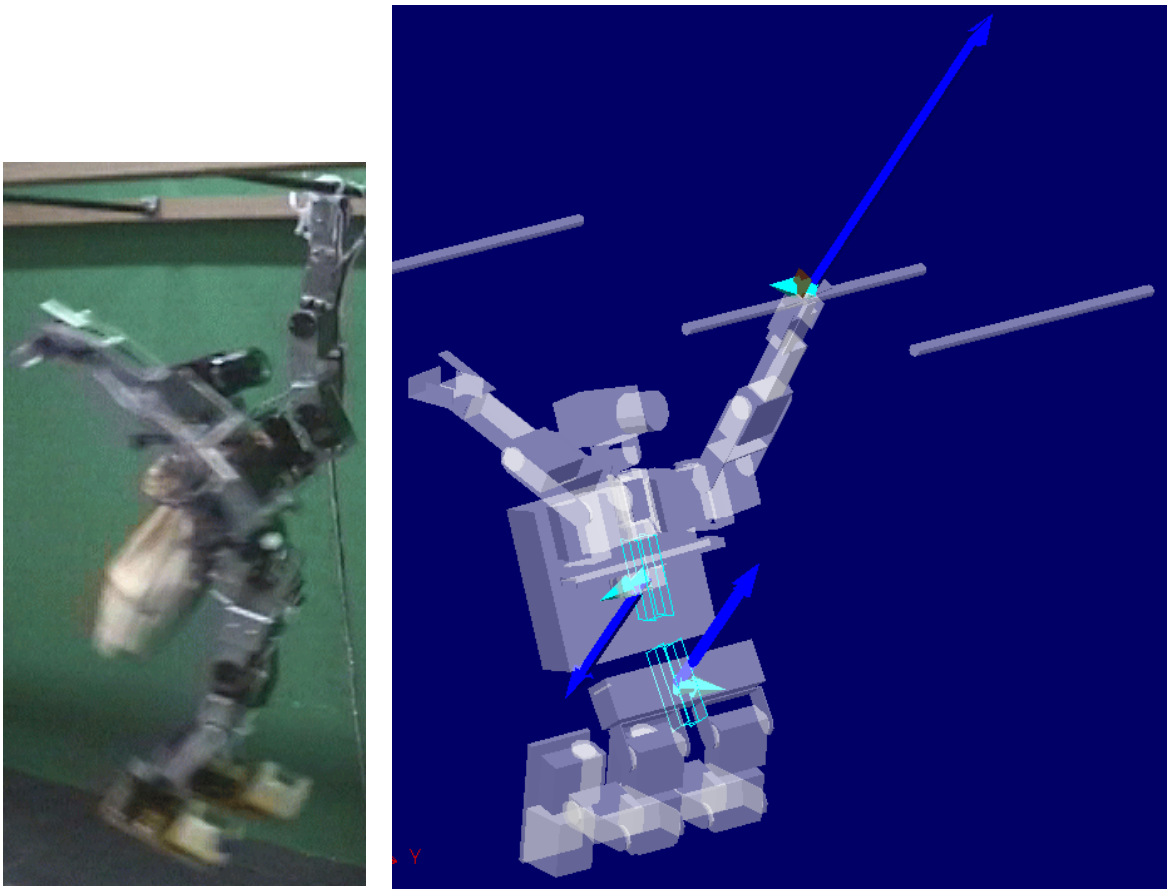


**Fig. 5.23** ブラキエーション動作 (実機とシミュレーション)
Brachiation motion (real environment and the simulation environment)

となり，計 70 の長さの遺伝子を用いた．その他 GA のパラメータは，個体数 105，交差率 0.2，突然変異率 0.02，淘汰率 0.4 とした．

### 5.7.4 実機とシミュレーションのブラキエーション動作

**Fig. 5.23** に，実機とシミュレーションによるブラキエーション動作を示す．実機を完全にモデル化できてはいないので，実機の動作とシミュレーション上の動作と若干の違いがあるが，柔軟性を変えた時の衝撃力の違いを見ることはできる．

**Fig. 5.24** は，体幹部が硬い場合と柔らかい場合の，体幹部の根元に生じる内力の大きさをシミュレーションで計測したグラフを示す．硬い場合の方がピークの値が大きいことがわかる．数値の大きさの正確性はモデル化の精度に依存するが，剛と柔の両方の場合での比較の意

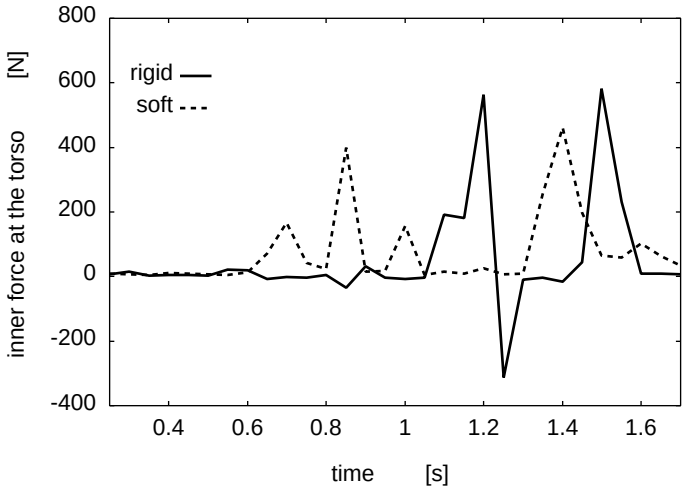



**Fig. 5.24** 動作中の体幹部における内力
Inner force at the torso during the motion

味では，その差が検証できることが重要である．

### 5.7.5 柔軟性を動的に変更しながらのシミュレーション

柔軟性を動作中に動的に変更する場合のシミュレーションを行うこともできるようになった．比較のために，(1) 脊椎が常に硬い状態，(2) 脊椎が常に柔かい状態，(3) 実機での動作と同様のシークェンス (足を後に引くときは硬く，手を離して次のバーを把持するまでの間に最も柔かい状態にする) で柔軟性を調節，の 3 種類の柔軟性で，5.7.3節に述べた方法によりシミュレーション上でブラキエーションが成功する動作シークェンスを，GA を利用して生成した．生成された動作を **Fig. 5.25** ～ **Fig. 5.27** に示す．

**Fig. 5.25** は常に脊椎が硬い状態でのブラキエーション動作，**Fig. 5.26** は常に脊椎が柔かい状態でのブラキエーション動作，**Fig. 5.27** は実機の実験と同様のシークェンスで柔軟性を調節しながらのブラキエーション動作を示す．

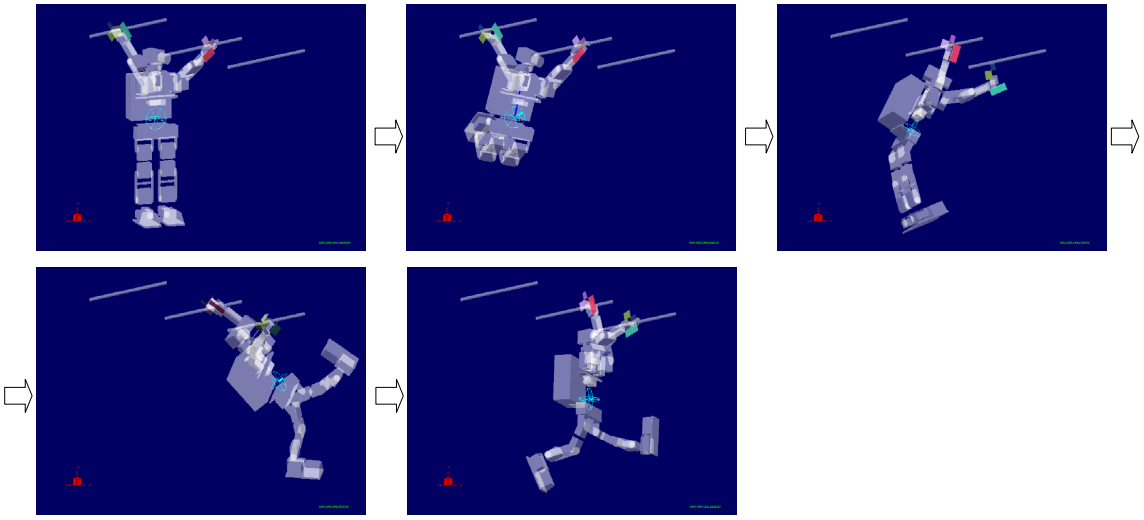


**Fig. 5.25** 脊椎が常に剛な状態のブラキエーション動作のシミュレーション
The spine is always rigid during the brachiation motion

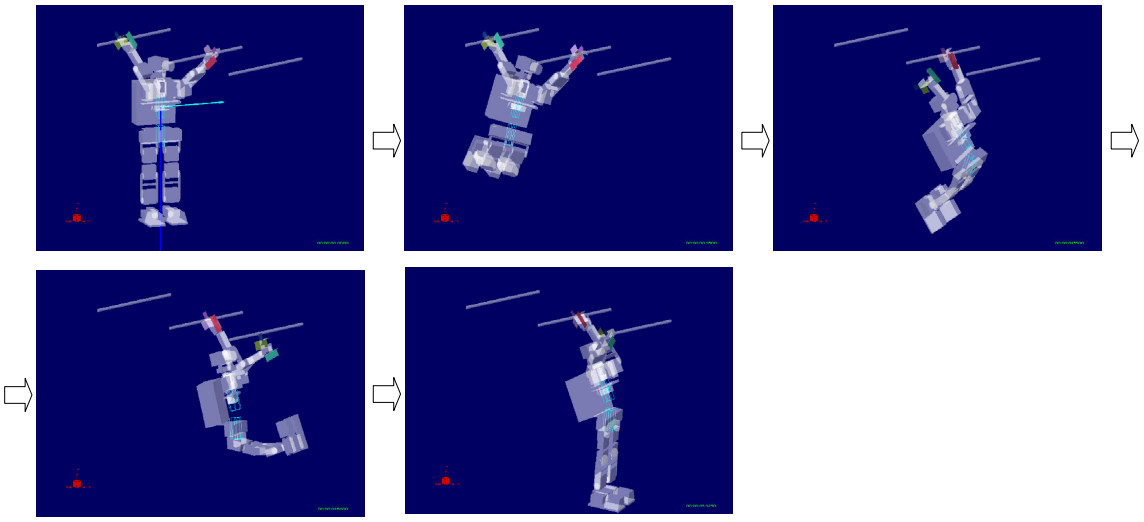


**Fig. 5.26** 脊椎が常に柔な状態のブラキエーション動作のシミュレーション
The spine is always soft during the brachiation motion

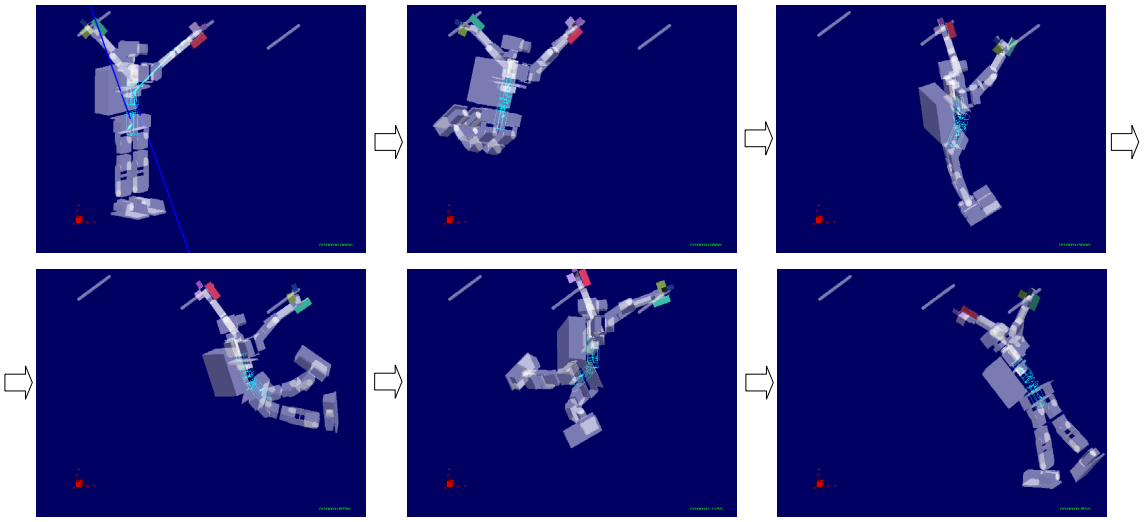


**Fig. 5.27** 脊椎の柔軟性を動的に変更しながらのブラキエーション動作のシミュレーション
The softness of the spine is dynamically changed during the brachiation motion

## 5.8 モーションキャプチャデータの利用

### 5.8.1 モーションキャプチャ

人間のモーションキャプチャデータには，脊椎の動きが含まれる場合が多い．これは，人間らしい動きには脊椎の動きが重要な役割を果たす場合が多いという事を意味していると考えられる．本研究で扱うような人間の脊椎構造に近い構造の脊椎を持つ全身型ロボットの動作生成において，モーションキャプチャデータを利用することは有効であると考えられる．本節では，モーションキャプチャデータを用いて脊椎を持つ人間型ロボットの脊椎の動きを含むような動作を生成するための枠組みに関し述べる．

これまでに，人間型ロボットの動作生成において，モーションキャプチャのデータを利用する試みはあった [67]．しかし，人間のような脊椎を有する全身型ロボットでモーションキャプチャのデータを利用する試みは例が無い．

### 5.8.2 BVH 形式

モーションキャプチャのデータの形式として業界標準と言ってよい形式である BVH 形式[5)]は，多くのモーションキャプチャ関連のツールで扱うことができる．BVH 形式はプレイ

[5)] BioVision 社 (http://www.biovision.com/) のモーションキャプチャ形式である．

ンテキストファイルで，ボディの構造の定義と運動情報を格納するための形式が定義されている．ボディの構造の定義 ( “HIERARCHY” で始まる) は各リンクの枝分かれ構造の定義 (“{”, “}” により入れ子構造にすることで，親子・枝分かれを定義)，それぞれの隣接するリンク間の初期相対変位の定義 (“OFFSET X Y Z” の形式)，そして各リンク間の自由度の定義 (“CHANNELS Zrotation Xrotation Yrotation” の形式) により，ボディの構造を定義する．動作情報 (“MOTION” で始まる) は，フレーム数，フレームタイムの記述の後，各フレームにおける関節の姿勢を羅列する．

この形式から情報を抽出し， EusLisp のロボットモデル環境 (第 6.5節) 上のロボットモデルに適用するためのツールを作成した．まずボディの構造の定義を読み出し，自由度の数と配置をロボットのモデルと比較する．そして，ロボットと BVH のデータの双方に存在する自由度を抽出する．これは後の動作情報を読み込む際に使う．大抵の場合は BVH 形式ファイルの方がロボットより多くの自由度を持っているが，腱太 (第 3.7節) のように，脊椎に 10 節もあると BVH より自由度が多い場合もある．その場合は， BVH ファイルに無い自由度は動かさないことになる．ボディ構造の擦り合わせが終ったら，続いて動作情報の読み込みを行う．双方に存在する自由度の分だけをロボットの関節角度として設定し， 1 フレームに 1 つずつ姿勢を読み込む．読み込んだ姿勢は`robot-state` クラスのオブジェクトとしてリストにつないでおく．

全ての動作情報の読み込みが終ったら，姿勢のリストの中から任意の姿勢を取り出してロボットにその姿勢をとらせることにより，人間がとった姿勢をロボットが真似をすることができる．リストにつながった順に取り出しては実ロボットに送れば，動作の再生ができる．Cla，腱太，ヒューマノイド HRP，小型ヒューノイド Akira を表示し，同じ bvh ファイルからデータを用いて，動作を行う様子を EusLisp のビューワに表示した． **Fig. 5.28** にそのシーンを示す．

**Fig. 5.28** は，コンピュータアニメーション作成ソフトウェアである 3D Studio MAX[6)] に付属のサンプル BVH ファイルを読み込んで，動作をさせたものである．腱太と Cla は脊椎を捻って，後ろに宙返りジャンプをする構えになっているが，脊椎を持たないヒューマノイドは体を捻っていない (捻る自由度がない)．

脊椎はあくまで冗長な自由度を追加するもので，脊椎が無いとできない動作というのはあまり無い．しかし，将来人間型ロボットがアクロバットのような動作をできるようになった時に，その動作を動力学的に可能にするためには脊椎構造の変形や力を利用しなければ不可能であるような動作も出てくるだろう．現在のヒューマノイドの運動能力は，脊椎の動きが不可欠なほど微妙な動作を求められる段階にはまだまだ遠いという状況であると考えることもできる

[6)] http://www.discreet.com/products/3dsmax/

**Fig. 5.28** モーションキャプチャデータから生成した動作例
A motion from motion-captured data

のではないだろうか.

## 5.9 姿勢データベース

第 4章で，教示・再生による制御，幾何モデルを利用した制御に関して述べたが，脊椎に様々な姿勢を取らせるためには，いずれの手法だけでも不十分な面がある．幾何モデルを利用した制御は，筋の干渉の問題を完全には解決できず，また，モデルと同じ姿勢に制御できるとは限らない．直接教示・再生による制御は，教示していない姿勢に制御することができない．こうした問題の解決への取り組みとして本節では，「姿勢データベース」と呼ぶ手法を提案する．

### 5.9.1 姿勢データベースの概要

姿勢データベースを次のように定義する．

**データベースの要素** 筋長ベースでセグメンテーションし，データベースのいずれの要素からも一定距離以上離れたデータは，新しい要素としてデータベースに追加する．

**動作生成への利用** 動作を幾何モデル上の姿勢列で与えられたら，そこから計算した筋長配列に最も近い (姿勢データベース中の) 姿勢から成る列に直して動作を実行する．必ず一度は取ったことのある (筋長の組合わせとしての) 姿勢の列になるので，筋の干渉などを問題にすることの無い動作が実現されることが期待できる．

**行動空間の拡張** データベースは常に更新．触覚により追加教示可能．求める動作と実行された動作に差があったら，追加教示することで行動空間を広げてゆく．また，これまでに取ったことが無いような姿勢が指示された場合も追加教示することで，以後はその姿勢を含む動作も可能になる．

**姿勢履歴の保存** これまでにデータベース中のどの要素付近を通ってきたかの履歴を保持する．同じ姿勢を何度も通ることもあれば，一度だけ通った姿勢というのもあり得る．

**内界センサの状態** 姿勢データベースのノードが保持する情報は，筋長だけではない．姿勢センサの情報などの内界センサの情報もノードが生成された時のセンサ状態をノードに記録する．ノード間の距離を計算する際に，基本的には筋の数と同じ次元の多次元空間におけるユークリッド距離を用いるが，姿勢センサを持つロボットの場合姿勢センサの情報も距離判別に利用する．姿勢センサの情報を筋長の多次元空間の一つの軸にするのではなく，姿勢センサ情報は姿勢センサ情報同士で距離を計算する．$(\phi, \theta, \psi)$ により表される姿勢センサの値を，他のノードに記録された姿勢センサの値と比較し，閾値を超えていたら新たなノードを生成する．

**Fig. 5.29** は，姿勢データベースの概念を 2 次元的に表現した図である．例えば，直接教示により pose1 から順に pose2 ⇒ pose3 ⇒ pose4 ⇒ pose5 ⇒ pose6 ⇒ pose7 ⇒ pose8 ⇒ pose2 ⇒ pose9 と姿勢を辿ったとする．姿勢履歴にはこのリストが保存される．ここで，動作指令として A という姿勢から開始し B という姿勢で終わるような動作をモデル上で与えられたとする．その時の 4.4.2節の幾何モデルに基づく筋長を計算し，姿勢データベース中で A および B に最も近い pose を探索する．探索の結果 A に最も近いのは pose9 で B に最も解のは pose6 だったとする．そうすると pose9 ⇒ pose3 ⇒ pose6 という順で筋長を制御する事により動作 AB が実現する．その途中でデータベースのどの要素からも一定距離以上離れた筋長空

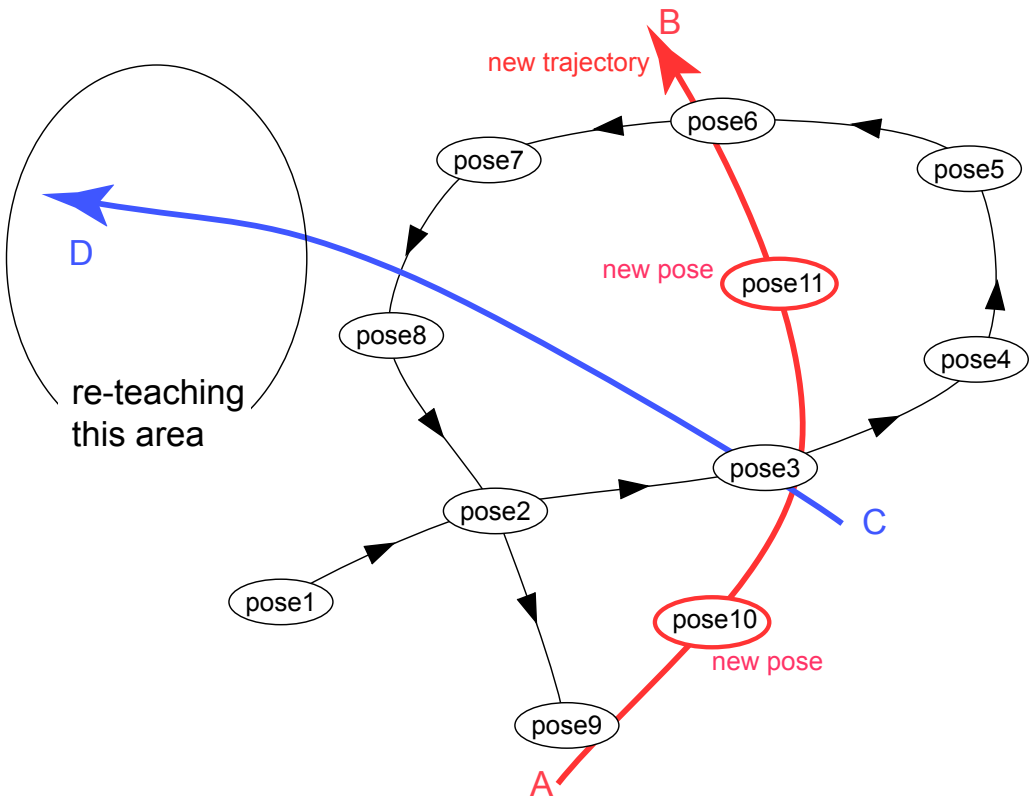



**Fig. 5.29** 姿勢データベースの概念
The concept of pose-database

間上の点を通ることで，新たな要素 (pose10,pose11) がデータベースに追加される．このようにして行動空間を広げてゆくことができる．

次に C から D という別の動作をモデル上で与えられたとする．そうすると，今度は pose3 ⇒ pose8 という経路から先はデータベースの要素が無いため， D 付近までたどり着けない．これがわかったところで，さっと直接教示モードに切り替え，新たに目的姿勢 D 付近の姿勢を追加教示することにより，データベースが更新され， C ⇒ D という動作を実現できるようになる．

### 5.9.2 姿勢履歴を利用した動作

姿勢データベースを利用した最もシンプルな動作は，姿勢履歴の再生である．姿勢履歴の再生により， 4.2.2節の「動作の直接教示」の項で述べたような大量のデータを記録することなく，教示動作の再生が可能になる．言い換えれば量子化頻度を下げることができる．

脊椎を持つ全身腱駆動ヒューマノイド腱太 (第 3.7節) の脊椎構造は，筋の干渉も強く (78ペー

ジ **Fig. 3.47** および 78ページ **Fig. 3.48** 参照)，幾何モデルに基づく姿勢制御ではどうしてもたるむ筋や張力が過大になる筋が何本か出てきてしまう．こうしたロボットにこそ，直接教示とそれに基づく動作生成法が有効である．

**Fig. 5.30** は，腱太による姿勢履歴を利用した動作の様子を示す．各行の上段は教示時の様子で，特徴的な姿勢を教示した順に示してある．各行の下段は動作時の様子で，上段の姿勢に対応すると思われる姿勢を下段の対応する場所に示してある．

4.2.2節の「動作の直接教示」の場合は，一定のサンプリング周期 (20[ms]) で筋長を記録するので，再生時には (倍速・ 3 倍速等も可能だが) 教示時の動作と相似な動作になる．しかし，ここで示した姿勢履歴を利用した動作の場合，教示データのサンプリング周期は姿勢がデータベース中のどのノードからも一定距離以上離れる毎なので，一定の速さで姿勢履歴を辿ると，教示時の動作とは時間軸は相似でない動作になる．

### 5.9.3 触覚センサの利用

腱太は全身に分布した触覚センサを備えている (3.7.3節)．これらの触覚センサの情報を教示状態のトリガに利用する．アナログ値のセンサなので，強く押されたかどうかによって挙動を変える．具体的には，以下のようにした．

#### 触覚センサの配置

触覚センサは，関節の自由度 ×2 の数だけ配置した．例えば，膝関節・肘関節は 2 個の触覚センサを配置し，股関節・足首・肩・手首等は 3 自由度の球面関節なので合計 6 点配置した．配置場所は，その関節の子リンクに当たるリンクに，その関節に対応した触覚センサを配置した．配置はその関節の自由度をどちらの方向に回転させようとしているかが判別できるようにした．例えば，肘の関節は前腕の表側と裏側に配置し，裏側が押されれば肘を曲げる方向の指示，表側が押されれば肘をのばす方向の指示であることがわかるようにした． 3 自由度の球面関節の子リンクには，その関節を roll 軸の正方向に回す時に力をかける方向と触覚センサを押す方向がほぼ一致するようになるような場所を選んで触覚センサを配置し， roll 軸負方向と pitch,yaw の各軸回りの正負回転に対応させたものと合わせて計 6 点の触覚センサがある．

脊椎や首は多節構造で，その各節に触覚センサを配置するとなると膨大な数のセンサが必要になる．実際には多節構造の単独の節を操作する必要があるような姿勢の教示は稀であるから，その際は触覚センサを利用せずに教示することにして，各節にセンサを取り付けることはしなかった．その代わり，脊椎の両端に各 6 点計 12 点の触覚センサを配置した．roll,pitch,yaw 各軸回りの正負に対応して 6 点である．両端 (腰と肩) にあるのは冗長に見える

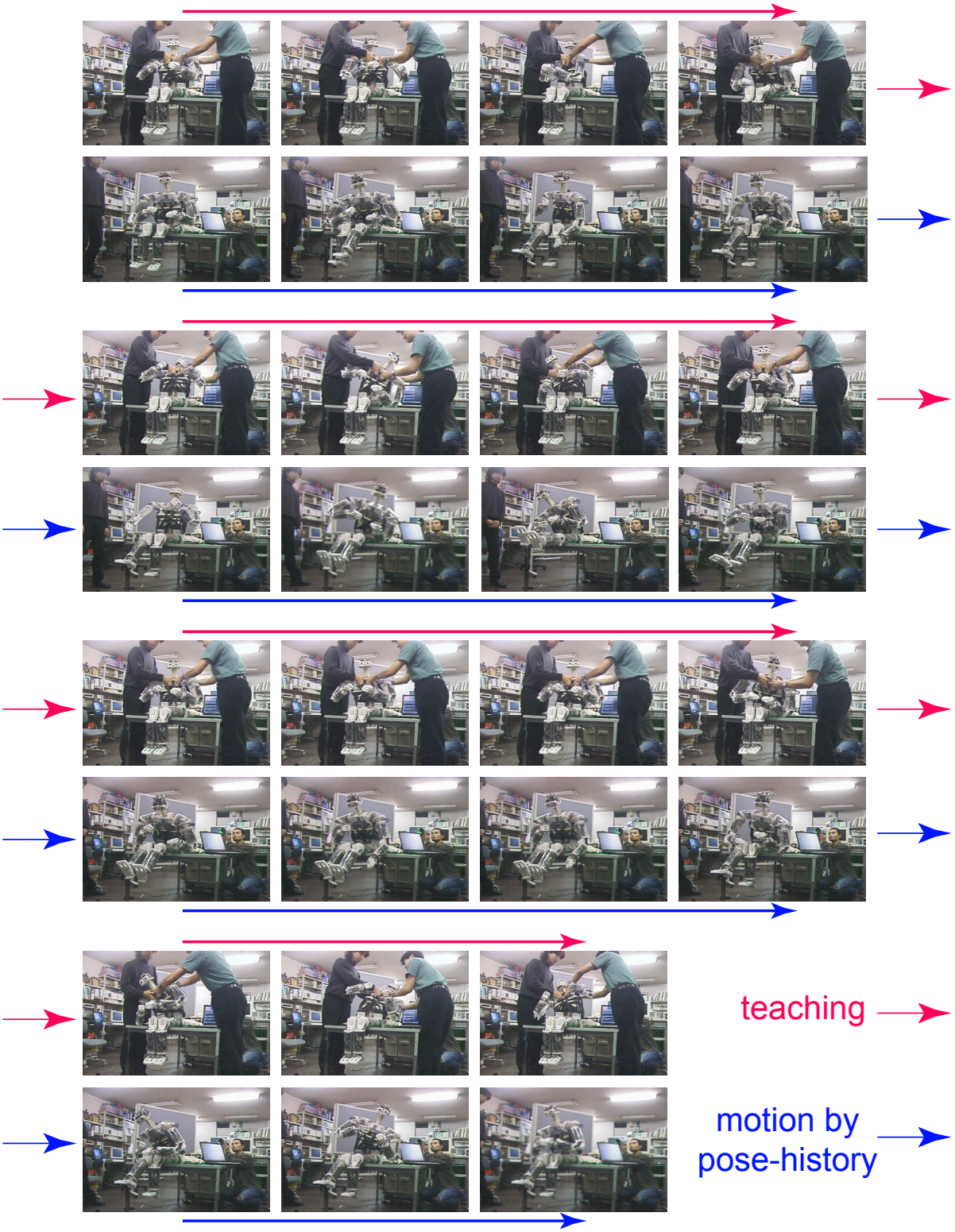



**Fig. 5.30** 姿勢履歴を利用した動作
A motion using pose history

がそうではない．肩と腰の位置関係は平行移動もある．例えば，腰ブロックから見て肩ブロックが前方に平行移動するような姿勢を指示するときは，腰の前面のセンサと肩の背面のセンサを同時に押す．本来腰の前面のセンサは脊椎が pitch 軸回りに負の回転の指示，肩の背面のセンサは脊椎が pitch 軸回りに正の回転の指示に対応している．

#### 触覚センサが弱く押された時

触覚センサを押す強さによって挙動を変える．弱く押されたときは相対的な姿勢の指示とみなし，強く押されたときは絶対的な姿勢の指示とみなす．相対的な指示の時は，現在のモデル上の姿勢からセンサを押す力に応じた角度だけ対応する関節の角度を変化させる．脊椎部分は，現在の姿勢の推定は 4.4.3節に述べたようにニューラルネットワークを利用する．そして求めた現在の姿勢から触覚センサの力に応じた角度だけ姿勢を変化させ， 4.4.2節の方法で目標筋長を求め，第 4.7節の主張にしたがった制御法で姿勢を制御する．脊椎や首以外の関節は幾何学的に関節姿勢角を求めることができるが，実機のモータエンコーダに基づく筋長計算は，巻き取りプーリの内径の変化や筋の伸びなどがあり，やはりそれほど正確には求めることができない．

#### 触覚センサが強く押された時

触覚センサが強く押されたときは，幾何学モデルは利用せず，関わりのある関節の全筋を一定張力制御モード (T 制御) にする．するとその関節は脱力状態になり，人間の力でその関節を直接動かすことで教示を行う事ができる．

#### 姿勢データベースとの兼ね合い

姿勢データベースは常に筋長 (と内界センサ情報) を監視し続けていて，幾何モデルベースに動かした時にも，弱い力で触覚センサを押して教示したときにも，触覚センサを強く押して教示モードに入ったときにも，データベースに無い姿勢 (筋長の組合わせ) を経由すれば，その点を新しいデータとしてデータベースを更新してゆく．

### 5.9.4 姿勢データベースの課題

**Fig. 5.30** に示した実験では， **Fig. 5.29** の A $\Rightarrow$ B という軌道は実験をしていない． **Fig. 5.29** の pose9 と pose3 の「距離」が新ノードが一つ入る程度の大きさであれば，pose10 はそれほど無理な姿勢ではなくなることが期待でき， A $\Rightarrow$ C の動作はある程度可能で

あろうと予測できる．新しいノードが二つ入る距離ではどうか，いくつまでなら動作可能かというのは，今後の課題である．

ノードのセグメンテーションに，多種センサの情報も利用してゆくというのも課題である．同じポーズであっても壁に寄りかかっている時と，床に座っている時では，適切な筋長が異なることはありうると考えら得る．そのため，触覚センサや力覚センサ・張力センサなどもデータに含めてゆくことが必要になると予想され，どのような形で利用するセンサを増やしてゆくかということは今後の課題である．

# 第 6 章

# 多自由度多センサシステムの設計と実現

本章では，システム構成法全般と本研究におけるシステム構成法を，全身行動との兼ね合いの観点を中心に論じる．また，EusLisp の幾何モデルを中心とする全身型ロボットのためのシステムアーキテクチャに関し述べ，筋駆動型ロボットのための拡張と，従来型の全身型ロボットの研究において蓄積されたソフトウェアを透過的に扱うための仕組み等に関し述べる．また，多自由度・多センサの全身型ロボットに特有の発展をしてゆくことのできるシステムとして，不完全性に対応可能であるためのシステム構築法を示す．複雑性ゆえに完全な状態を常に保つことは困難であるという認識の元に，できるだけ多種・多数のセンサ・アクチュエータを搭載し，一部のコンポーネントが故障していても動作可能なシステム構築法を示す．

## 6.1 脊椎構造を持つ多自由度全身型ロボットシステム

本研究で扱うような脊椎構造を持つ多自由度・多センサの全身型ロボットは，例えば腱太では 96 個のモータ， 94 個の張力センサ・ 8 個の 3 軸加速度センサ・ 62 個のアナログ触覚センサ・ 94 個の電流センサ・ 2 個の CCD カラーカメラ等，多種・多数のセンサが搭載されている．こうしたロボットのシステムに必要な要件を以下に挙げる．本章では，これらの要件を満たすロボットシステムの全体構成法を示す．

### 6.1.1 多入力・多出力を扱うロボットシステム

多自由度・多センサの全身型ロボットシステムに特有の発展をしてゆくことのできるシステムが求められる．重要な点は，頑健性，フレキシビリティ，多入力・多出力への対応を考慮したシステム構成である．こうした点を，多入力多出力なロボットシステムのポイントであると考え，多入力多出力なロボットシステムの全体設計を行う．

以下に示す項目により，多入力・多出力を扱うロボットシステムを実現した．

**頑健性** 末端に故障個所があってもシステムは停止しない．エラーリカバリ機能 (ウォッチドッグタイマ等) を有する．

**フレキシビリティ** 設定の変更自由度が大きい．多種類のロボットを扱える．ロボット毎に設定ファイルを用意し，アクチュエータの数やセンサの種類・個数を記述する．

**多入力・多出力への対応** 体内分散プロセッサネットワーク (体内 LAN)[53] により，配線を減らす，センサやアクチュエータの追加・移動・取り外しなどへの対応が容易，などの利点が得られる．

### 6.1.2 自律診断調節系

複雑性ゆえ，完全な状態を常に保つのは困難な対象であると捉えて，不完全性に対応できるためのシステムはどうあるべきかを検討するべきである．多数のセンサ・アクチュエータが組み込まれているので，センサ・アクチュエータの故障は常に生じる．全てのセンサ・アクチュエータが正常な状態である日は少ない．不完全性に対応するために，センサの状態を常に監視し，自己状態を診断する系を持つ事が有効である．

不完全性を考慮して，多種センサの状態を常に監視し，自己状態を診断する機能，いわば自律調節系とでもいうものが非常に有効である．常に完全な状態であることを想定しない (人間も調子の悪い部位があるものである)．自律調節系の例を以下に挙げる．

- 筋のたるみ検出・修復機能
- 過電流検出・モータ停止機能
- 張力センサ故障検出機能
- エンコーダ故障検出機能
- 体内 LAN 通信ハング検出復帰機能

### 6.1.3 従来のヒューマノイドのソフトウェア環境と透過な環境

過去に蓄積されてきた膨大なソフトウェア [26] を利用できる部分は利用し，脊椎や多自由度・多センサに対応して新たに追加しなければならない部分は追加する．重要なことは，これまでのソフトウェア環境と透過的に扱える環境を構築することである．詳しい実装に関しては第 6.5節に述べるが，ポイントは関節角度列による姿勢表現を用いて記述できるようにすることである．脊椎構造は，関節角度とアクチュエータ変位が 1 対 1 に対応していないが，第 4章に述べたような関節角度→アクチュエータ変位の変換と，アクチュエータ変位→関節角度の変換を実装することにより，動作情報を関節角度のシークエンスで記述することができ，従来の人間型ロボットのソフトウェア環境を利用できるようになる．

## 6.2 全身型ロボットのためのシステム構成法

### 6.2.1 ロボットシステムの構成法の研究

ロボットの研究は，運動制御，認識，プランニングなど様々な側面に焦点を当てて進められてきたが，それらを統合して実世界で知的に自律行動するロボットを実現することは依然と

して困難である．その理由の一つに，ロボットの情報システムに要素機能を組み込み，統合する方法論が十分に確立していないことが挙げられる．機能統合の方法を確立するには，システムアーキテクチャが重要であり，近年，これまで行われてきた様々な機能の研究を組み込むための，ロボットシステムの研究が盛んに行われている．本論文では，こうした近年のロボットシステム構成法の研究を参考に，全身型ロボットによる全身動作を行うためのロボットシステムの構成法を考察する．

### 実世界との相互作用による機能の実現

実世界の知能を実現するシステムとして，環境との相互作用の重要性の認識が広まっている． Brooks は，センサ情報の局所的で即応的なフィードバックを持つ非同期並列システムによって移動ロボットが実世界で自律的な振舞いを見せることを示し，感覚と運動による環境との相互作用の重要性を主張した [6].

こうした主張は，ロボットシステムの設計論に多大な影響を与えた．それまでのロボットシステムは，タスク→行動計画→動作手順→動作という階層的で構造化されたトップダウンなアプローチが主流だった．これに対し， Brooks の提案したサブサンプション・アーキテクチャ [5] は，感覚と運動を密接に結び付ける実世界との相互作用による “機能” をモジュール化し，複数の機能の上にロバストなタスク遂行能力ができあがるという，ボトムアップなアプローチにより構築しようという枠組だった．これは，逐次的に分解されていたそれまでの行動計画法とは違って，機能により分解された行動要素が並列に動作するという形態で，統合された行動を実現するという意味でも，斬新な枠組だった．多指ハンドの制御にこうした枠組を利用する例もあり [71, 73]，全身型ロボットの全身動作にも適用できると考えられる．

感覚と行動を直接結んだいわば “反射” のみからなる機能の重畳より生じる知的に見える行動に，知能の本質があるのではないかとも思わせたが，その後その単純とも言える枠組だけでは様々な問題点があることも指摘され，改善するための様々な試みが提案された．

サブサンプション・アーキテクチャの問題点を改善する試みの一つに，記述方法の改善があった．実世界という複雑な環境との相互作用による機能モジュールは，ハードウェアに依存する機能とやや抽象度の高いモジュールが混在し得る．こうした問題を改善するために，各機能モジュールを抽象化し，記述と実装を明確に分離すべきであるという主張がある．岡による BeNet は，ハードウェアに依存しない概念上のモジュール記述モデルである [82].

また， Brooks の主張は，従来の階層化されていたアーキテクチャを，プレーンな構造にするものであったが，ある程度高次の知能は実現が困難であるという側面を持っていた．こうした問題を解決するため， Brooks の主張とは矛盾するが，サブサンプション・アーキテクチャの枠組を持ちつつ，階層的に構造化されたアーキテクチャの提案もなされている．

Connell は， SSS という時間・空間がそれぞれ連続かどうかの組合せで， 3 つの層の分け方を定義したシステムを提案した [8]． Bonasso は， 3T という時間・バンド幅などのいくつかの dimension において，その粗さでどの層に分けるかを決めるという形のシステムを提案した [3]．いずれも三層構造のアーキテクチャで，一番下の層がサブサンプションの層である．

### 階層的システム構成

6.2.1での議論は，機能の実現の方法論であった． Brooks は，階層的構造を取り払うことによって実時間性を持ちつつ，巧妙に機能モジュールを組み立てることによりロバストに行動することができることを示したが， 6.2.1の最終パラグラフに述べたように，ある程度以上の知能の実現には，それだけでは不足であるとする主張もある．すなわち，環境の変化に素早く対応することのできるための枠組を下位に持ちつつ，従来の人工知能で行われているような記号処理なども上位で行えるような，階層的な枠組も必要ではないかということである．

こうした階層的な構成法は，実時間処理と記号処理などの相反する要素を両立させるための枠組と言え，各層における処理を他の層を意識せずに扱うことのできるシステム構成法である．ソフトウェアレベルだけでなく，ロボットシステム全体に関しても階層的構成法の議論があてはまる．すなわち，本研究で採用しているリモートブレイン・アプローチ [19] も，システムをボディとブレインという形で大きく二層に分離することにより，アクチュエータや構造部材などのボディの設計に関する議論と，ソフトウェアに関する議論を完全に独立に扱うことのできる枠組であると言うことができる．

ところで，人間の脳は階層構造になっているように見えるが，その構造は一意には決まらない部分も多々あり，ある面から見た層構造は，別の面から見るとまた別な層構造が考えられる場合もある．また，明確に層が分離しているのではなく，層として捉えた構造を跨ぐ結合の存在もある．そして，層構造も決して一次元的なものではない．脳が理想的なシステムであるかはわからないが，高度な知能の実現のためには，非常に複雑な構造が必要なのではないかと思わせる．

ロボットのシステムを層構造で構成するのは，知能の実現のためというよりはむしろもっと現実的な，わかりやすく扱いやすい形でシステムを構築するための手法であると捉えることもできる．

### 自己組織化

6.2.1および 6.2.1では，システムの構造に関する議論を行ったが，本節では，所望の行動を実現する方法について，自己組織化などの探索的手法の利用に関する議論を行う．

(i) 機能を実現するソフトウェアの導出

6.2.1では，“機能” を実現するソフトウェアをいかにして導出するかについては議論しなかった．機能を実現するソフトウェアを導出する手段として考えられる方法を以下に挙げ，その特徴を述べる．

**解析的手法**

理論的な計算に必要なモデル化や，仮定を設けて，それに基づいて解析的に解を求め，それを機能として実装するという手法である．

例として人間型ロボットのバランス制御という機能を考えると，ロボットの幾何学的構造や重量などの物理的なモデルと，姿勢センサやトルクセンサなどの情報から，バランスを維持するための制御を理論的に計算し，それをインプリメントするということを意味する．

解析的手法は，外乱や不確定要素の無い場合は，非常に有効であり，ロボットの機能を実現する手段として最も頻繁に利用されてきた．

**探索的手法**

理論的なモデル化の困難な機能に対しては，自己組織化などの利用が有効な場合がある．この場合，解析的に導出した教師ソフトウェアを利用する教師あり自己組織化と，所望する機能の達成度を評価する基準を設ける教師なし自己組織化が考えられる．

**実験的手法**

何らかの方法でソフトウェアのプロトタイプを作成し，それを実験により調整することにより，所望の機能を実現するという手法である．あまり確実性があるとは言えないが，モデル化や解析が困難である場合，自己組織化のための初期データとして，こうした手法により得たソフトウェアを利用することができると考えられる．

実験的手法は人間の助けを借りてソフトウェアを導出することになり，解析が困難な場合や不確定要素に対応しなければならない場合に有効であるが，様々な機能を一つ一つ実験的に実現していくのは，あまり現実的ではない．

したがって，未知物体の操作を視野に入れる場合，探索的手法すなわち自己組織化の枠組を導入できるようにシステムを構成することが必須となると考えられる．

(ii) 階層構造における自己組織化

階層構造における自己組織化は，各階層内での自己組織化と，階層構造の自己組織化が考えられる．

**各階層内での自己組織化**

前項で述べたのは，世界との相互作用による機能を担う層の内部の自己組織化である．同様に，例えば動作の手順を担う層があるとすると，そのどのような順序でどの動作を行えば，所望の結果が得られるかを，自己組織的に獲得するというのが，各層における自己組織化である．全ての層においてこうした探索的手法が有効かどうかは一概には言えず，各層ごとに最適な手法を選択しなければならない．

**構造自体の自己組織化**

適切な構造を探索するという考え方である． 6.2.1で述べた BeNet[82] を，構造の変化が可能なように拡張した例 [28] は，こうした考え方の導入が可能なシステムの提案であると言える．構造の変化を可能にすることはできても，この考え方は構造の評価基準の設定が難しく，そもそも最適な構造というものがわからないために，様々な構造が提案されているという現状を考えると，実際に構造自体の自己組織化という概念を導入するにはまだまだ課題が多いと言える．

### 6.2.2 本論文で提案するシステム構成法

6.2.1節で議論したロボットシステムの構成法の研究を考慮し，本研究では全身型ロボットのシステム構成を以下のように構成する．

1. 行動の指令は抽象的なレベルで与え，動作・機能の多くは自律的に行うことを目指した枠組であること
2. 感覚と行動が密接に結び付く機能を持つこと
3. 自動獲得する仕組みを組み込むことができること

### 6.2.3 自律的機能を担う系を持つ二層構造の脳システム

抽象レベルの高い部分を扱う上位と，抽象レベルの低いレベルを扱う下位という，少なくとも二層の階層的システム構成が必要である．

つまり，バランス維持・反射動作などの，人間が無意識に行っているような下位の自律的機能を，脳の最下位層に埋め込むことによって，そうした機能を意識せずに上位の行動を記述することができるようになるための，最小限必要な構造である．同時に，そうした条件を考慮せずに脳の上位の枠組を議論することができるようになるための枠組でもある．

本研究では，ロボットの脳のシステムの最も基本的な枠組として，最下位層に自律系を持つ脳の構成法を提案する ( **Fig. 6.1** ).

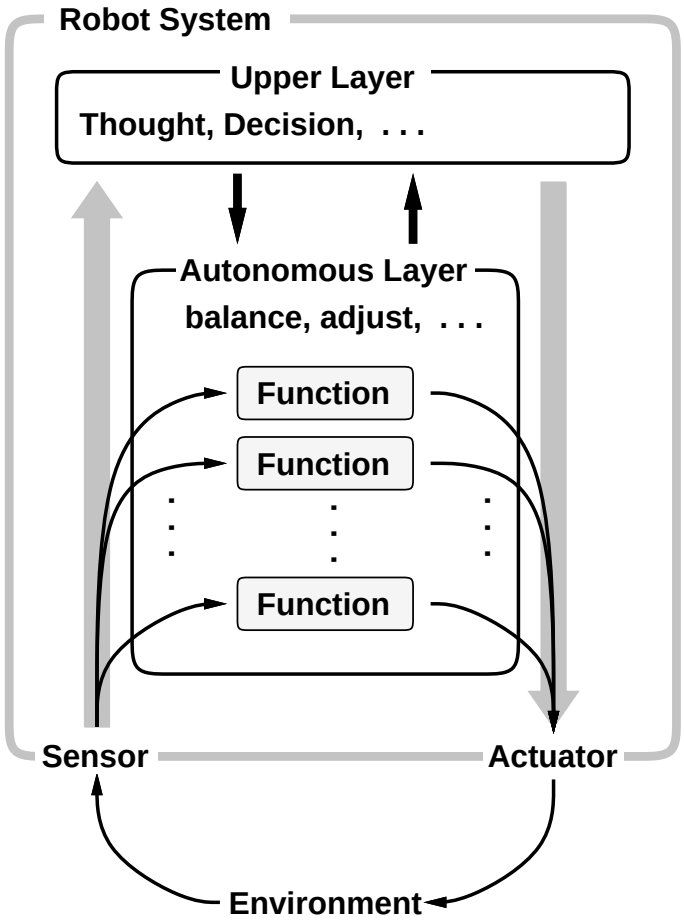



**Fig. 6.1** 自律系を備えたロボットシステム
Robot system architecture with the autonomous system

### 6.2.4 自律的機能を担う系の枠組

本研究では 6.2.3節で述べた自律的機能を担う系を，センサ・アクチュエータが密接に結び付いた反射動作を行う自律機能モジュールが，必要に応じて複数並列に実行されるという形で構成した．さらに，並列に出力される自律機能モジュールからのアクチュエータ情報の出力は，絶対的な値ではなく現在値との差分とし，それらを現在値に足し合わせる形にした．

これは，

- 各単機能モジュールは，入力情報・出力を計算する関数・出力情報という，最小限必要な枠組だけを持つ．
- 上位からモジュールの起動・終了・パラメータの変更が自由にできる．

- 競合するモジュールの出力がある場合，サブサンプション・アーキテクチャの場合は優先度により調停を行うが，本手法は差分の和という形をとる．

という特徴を持つ． 3 点目に挙げた特徴は，競合する出力が単純な和の形になり両方が生かされるという利点があるが，矛盾する出力を和の形で問題なく解決できるかという疑問もある．差分の和の形はまだまだ改善の余地があり，重みつきの和を用いるなどの手法 [72, 73] が考える．

この特徴により，

- 必要な機能を実現する入出力関係・パラメータを，自律機能モジュール内で学習する．
- 行動レベルで，どういう順序・パラメータで自律機能モジュールを起動・終了すれば，所望の行動になるかを自己組織化する．

という二つの，自己組織化のための可変性を持つ．

## 6.3 中間層のソフトウェア構成 — 自律機能を担う系を実現する nervous—

自律系の実現には “nervous” (付録 Cを参照) と呼ぶソフトウェアを開発した ( **Fig. 6.6** 参照)． nervous は神経系 (nervous system) を意味する． nervous はロボットボディと上位層との間に位置するプログラムで，主に以下のような役割を担っている．

### 6.3.1 ハードウェア抽象化層

nervous は，上位層からロボットのハードウェアを抽象化する役割を担っている．ロボットとのインタフェースの仕様 (API) として，以下の 4 つを定義した．

**初期化** `int OpenInterface(int, char **);`

**センサ情報の取り込み** `RobotState GetRobotState(RobotState *);`

**アクチュエータ情報の送信** `vector SetAngleVector(vector);`

**終了処理** `int CloseInterface(void);`

インタフェースとしては具体的には，シリアル通信やソケット通信等を定義し，上記 API を実装すれば新たなインタフェースの追加も可能な構成である．

これにより，ロボットと上位層を結ぶインタフェースの物理的仕様に関わらず， nervous より上位の層からは同様に扱うことができるようになり，ハードウェアを抽象化することができた．

本研究では，腱太以外のロボットにおいては， 1 ポートのシリアル通信のインタフェースを利用した．通信速度は Cla は 153.6[kbps]，それ以外は 38.4[kbps] である．腱太に関しては 1 ポートでは帯域幅が足りなかったため，複数ポートのシリアル通信を並列して行うインタフェースをマルチスレッドライブラリを用いて新たに開発し， 307.2[kbps] のシリアル通信を 4 系統利用する構成にした.

### 6.3.2 ロボット抽象化

多種類のロボットが利用できるように，ロボット毎に設定ファイルを作成することで，nervous 本体は同じプログラムを利用できるようにした．具体的には,

- ロボットの持つアクチュエータ数・制御パラメータ (PID ゲインなど) の種類と数・搭載されたセンサの種類と数などが定義された C 言語のヘッダファイル． EusLisp の幾何モデルから変換できるようにした.
- ロボットの幾何モデルを C 言語で記述した C のソースファイル． EusLisp の幾何モデルから変換できるようにした.
- ロボットの逆動力学などを計算するための関節情報，リンク質量・慣性モーメント情報などが記述された C 言語のソースファイル． EusLisp の幾何モデルから変換できるようにした (EusLisp 上に動力学計算のための情報が定義されたロボットに限る).
- オンボディプロセッサの ID とその ID のプロセッサに接続されたセンサ・アクチュエータの種類と数がそれぞれ記述された C 言語のヘッダファイル.

などがある.

これにより，上位の層からは nervous との通信オブジェクトを解して，ロボットの種類に依存しない共通の関数を利用してロボットとのアクセスを記述することができる.

### 6.3.3 上位層とのインタフェース

nervous はソケットサーバの機能も果たしており，上位層のプログラムからのマルチコネクションを受け入れ，センサ情報・アクチュエータ情報の受け渡し，自律調節機能プラグイン (6.3.4節参照) の起動・終了・パラメータ調節などのコマンドインタフェースも提供する.

nervous の上位層とのインタフェースは，キャラクタベースのアクチュエータ指令・センサ情報のやり取りとコマンド解釈系からなる．コマンド解釈系にしたので，インタラクティブに情報を操作・閲覧できると同時に，上位層はソケットインタフェースを備える EusLisp で

記述されており，ソケット通信でも同様の操作・閲覧ができるようなインタフェースを構築した．双方向ソケットを開き，センサ周期と同じ周期でセンサ情報クラスのインスタンスを生成するストリングをソケットに送信しつつ，コマンドやアクチュエータ情報を非同期に受信しボディに送信するデータを更新する．

以下に EusLisp から nervous に接続し，センサ情報の取得，モータ指令値の変更を行う様子を示す． nervous が走っている “host” に接続し (1 行目)，張力情報を取得し (2 行目)，モータ角度情報を指令する (3 行目)．

```
1: (setq *ns* (open-nerve :io :host "host"))
2: (send *ns* :state :tension-vector)
3: (send *ns* :angle-vector angles)
```

### 6.3.4 プラグインアーキテクチャ

nervous の機能拡張手段としてプラグインの形態を用意した． nervous 内部で，取り込んだセンサ情報を利用してアクチュエータ情報を操作するような機能モジュールを，自由に追加・取り外しができる環境を開発した．これにより，上位層まで上がらない制御ループを構成することができ，自律機能モジュールや反射機能モジュールをここに実装することができる．plugin 構造体を定義し，メンバ関数にコンストラクタ・処理ルーチン・デストラクタを定義し，動作開始の指示があるとコンストラクタ関数を実行し，センサ情報が入力されるたびに処理ルーチン関数を一度実行し，動作終了の指示があるとデストラクタ関数を実行する．

最も良く利用されるプラグインとして，補間・姿勢列管理プラグイン (seqplugin) を定義した (付録 C参照)．目標姿勢 (アクチュエータ変位の列) と目標に達するまでの時間を引数とし，指定された時間をかけて現在のアクチュエータ状態から目標状態まで連続的にアクチュエータ変位が変わるように，目標アクチュエータ変位を更新してゆく．補間方法は，線形補間・躍度最小補間を実装した．補間中に新たな目標値が与えられるとキューに溜まってゆき，順に実行される．オプションとして，速度を与えられるようにしてあり，途中姿勢における速度の指定も可能になっている．指定が無ければ各途中姿勢において速度は 0 となる．

## 6.4 システムの実装

### 6.4.1 ハードウェアシステム構成

ボディにマイクロコントローラを分散配置し，それぞれのコントローラが数自由度・数センサ分の処理を担当する形にし，それぞれのコントローラ間の通信によりデータのやりとりを

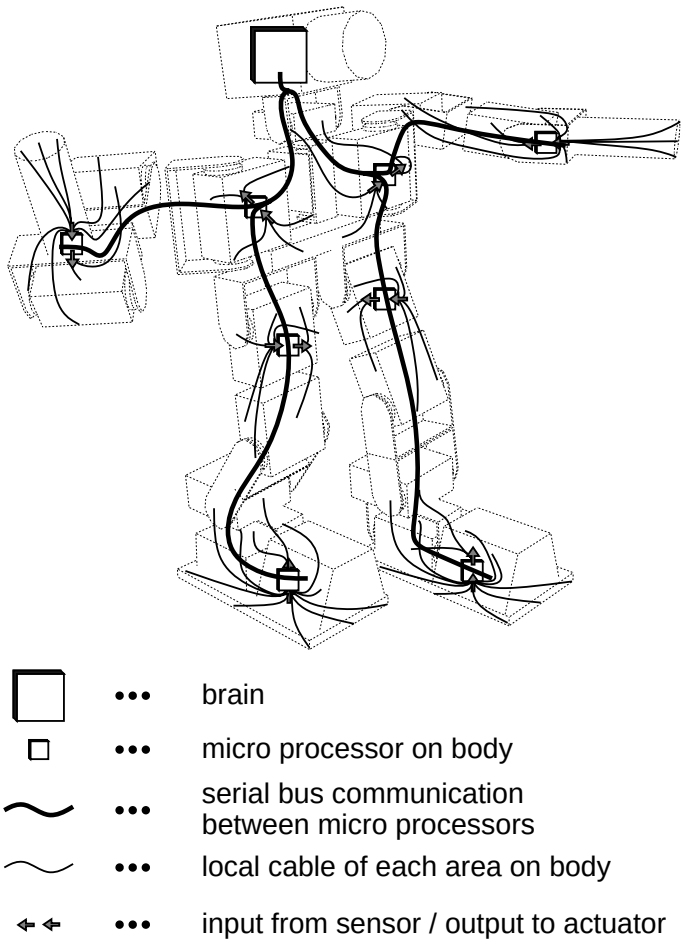



**Fig. 6.2** 体内 LAN の概念図
The concept of onbody-LAN

可能にするために体内 LAN [53, 27] を構築した． **Fig. 6.2** は体内 LAN の概念図である (ただし， brain は外部にある場合も想定できる).

体内 LAN の通信は， Philips 社の提唱する $I^2C$(Inter IC) バス [86] を選択した．主な理由は，ワンチップマイコンに搭載されている通信プロトコルの中では比較的速かったこと，クロック・データ・グランドの 3 線のバスに 128 チップまで接続できる省線であること，マルチプロセッサで利用できること，などである．

体内 LAN のノードとしては，日立のワンチップマイコンの H8/3334Y， H8S/2128， H8S/2138, SH7042 を利用して小型基板を開発してきた． SQ43， Rabbit には H8/3334Y の搭載された基板を開発し利用した． HanzouS には H8/3334Y と Sh7042 を利用した． Cla には H8S/2128 を搭載された基板を 2 種類開発し利用した．腱太には， H8S/2128 及び小型モータドライバ 4

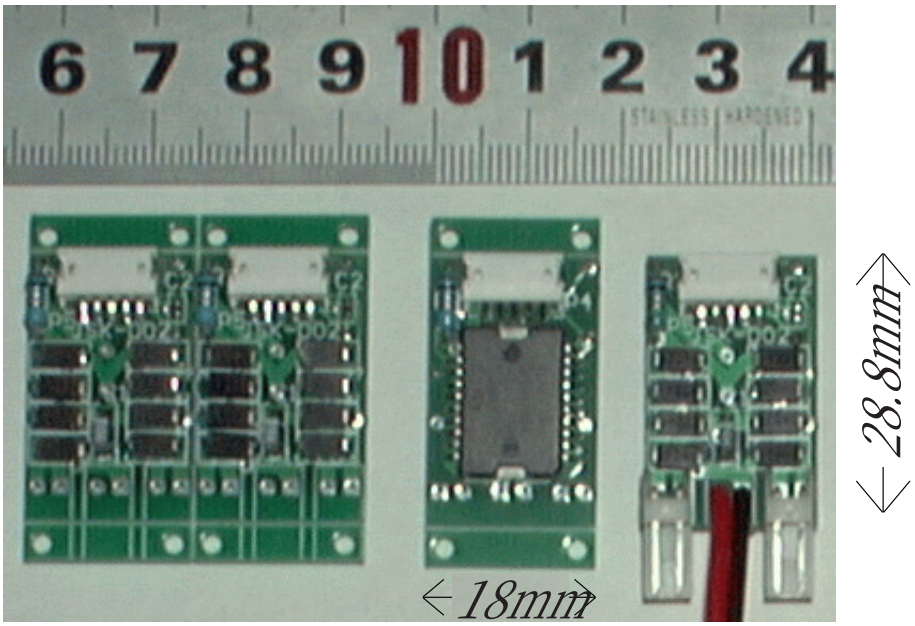



**Fig. 6.3** JSK-D02 の外観 (4 組)
Appearance of JSK-D02 (4 sheets)

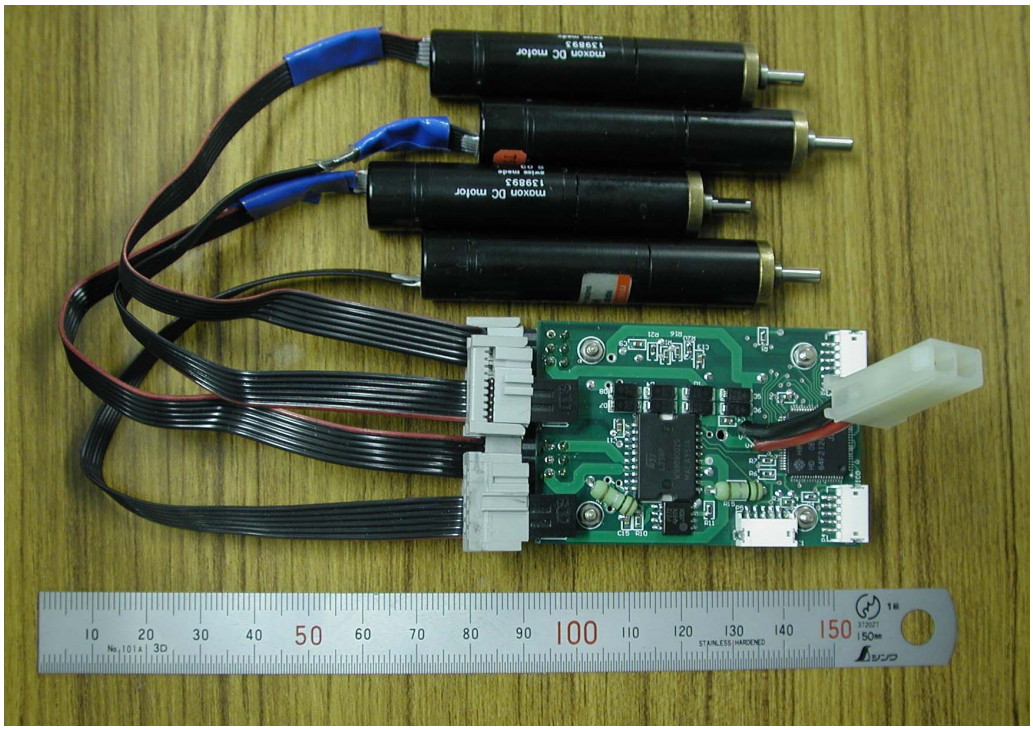



**Fig. 6.4** H8-08D の外観
Appearance of H8-08D

軸分を搭載した基板 (H8-08D, **Fig. 6.4** ) と, H8S/2138 を搭載した基板を混在して利用した.

体内 LAN のノードであるオンボディコントローラは, AD 変換器やパルスカウンタによってセンサ情報の収集が可能であり, 汎用 I/O ポートや PWM 出力を利用してモータを駆動することも可能だが[1)], こうしたモータでは制御パラメータや制御周期の変更ができず, 張力制御等やセンサベーストな制御を行おうとする場合, 二重のループができることになり好ましくない. 張力センサを利用した制御を行うロボットのために, モータドライバを搭載した基板を開発した. **Fig. 6.3** は 2 軸の超小型モータドライバ基板, **Fig. 6.4** は H8S/2128 を 1 台と 4 軸のモータドライバを搭載した基板である.

### 6.4.2 ソフトウェアシステム構成

ソフトウェアは大きく分けて三層構造とした ( **Fig. 6.6** ).

1. 最上位層

   ロボットの幾何モデルや, 行動制御等を記述する. **Fig. 6.6** の EusLisp の部分である. 最上位層の中の構成は更に可能であるし, 並列ユニット群 (BeNet[82, 83]) の構成にすることも可能である. ここでは, ソフトウェアの実装としては EusLisp により記述される最上位層と表現する. 従来の人間型ロボットのソフトウェアシステム環境との透過性などを含めた詳細に関しては第 6.5節に述べる.

2. ハードウェア抽象化層

   第 6.3節に述べた nervous の環境である. 中間層にあたり, ハードウェア抽象化, ロボット抽象化, 解釈系を有する上位層との汎用インタフェース, 並列センサ・モータモジュール [61] であるプラグインアーキテクチャ [31] などからなる.

3. オンボディプロセッサ群

   6.4.1節に述べたように, 体内にプロセッサを分散配置し, 各プロセッサは局所的なセンサ情報収集とアクチュエータ制御を行う. 全プロセッサは体内 LAN を介して体内 HUB( **Fig. 6.5** 参照) と通信し, HUB は遠隔のホストとシリアル通信をする. この構成により, センサ・アクチュエータの追加・取り外しへの対応, HUB がマスタをすることにより, 分散プロセッサに故障プロセッサが発生しても, それ以外のプロセッサは正常動作ができる. なお, $I^2C$ バスは, H8S/2128 及び H8S/2138 を利用した場合, 最高クロック約 714[kHz] で通信し, 実効速度も数百 [kbps] になる.

[1)] ラジコンサーボモジュールのようにパルスにより駆動できる種類はマイクロコントローラで直接駆動できる.

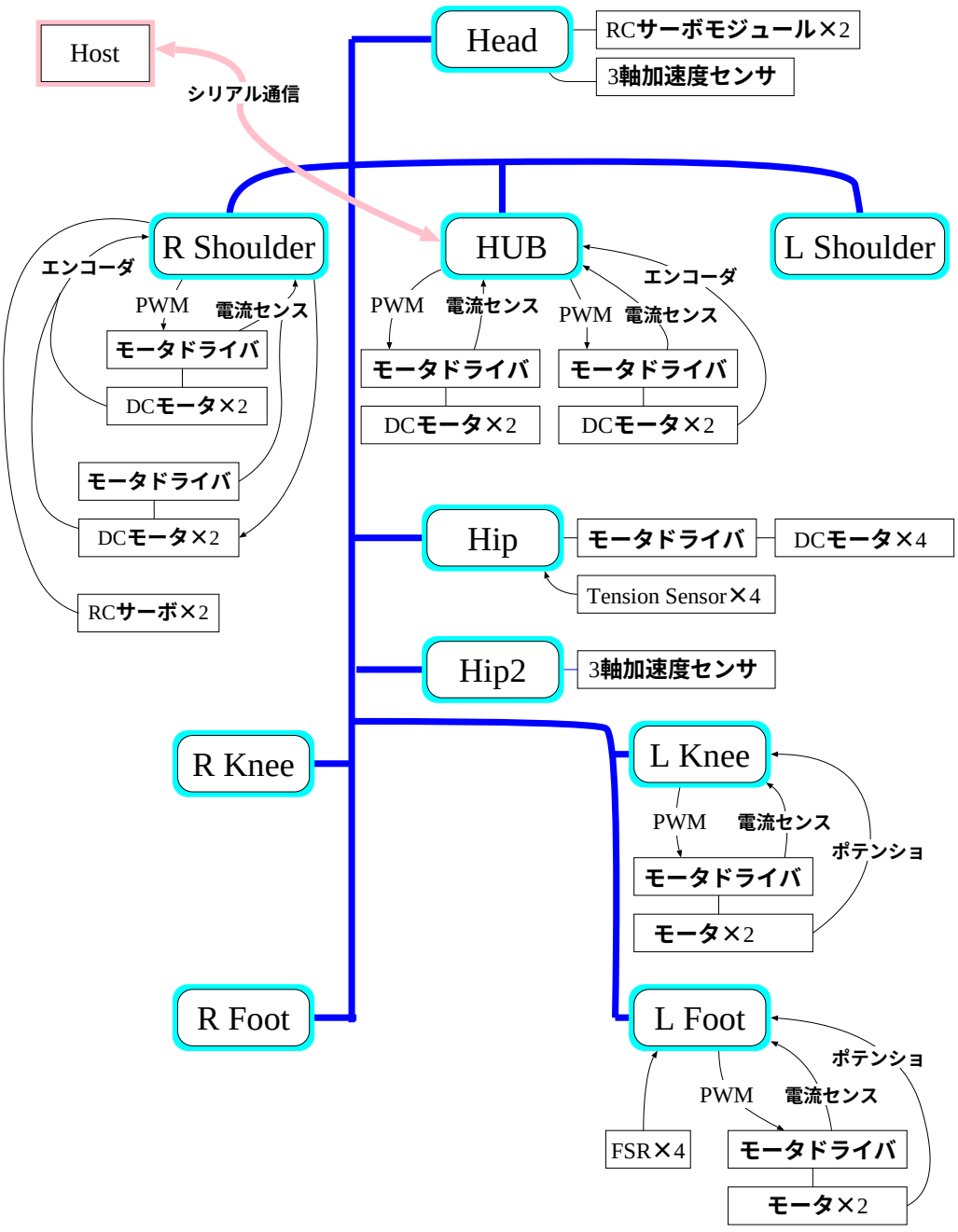



**Fig. 6.5** 体内 LAN のプロセッサ配置 (Cla の例)
Arrangement of onbody processors (the case of Cla)

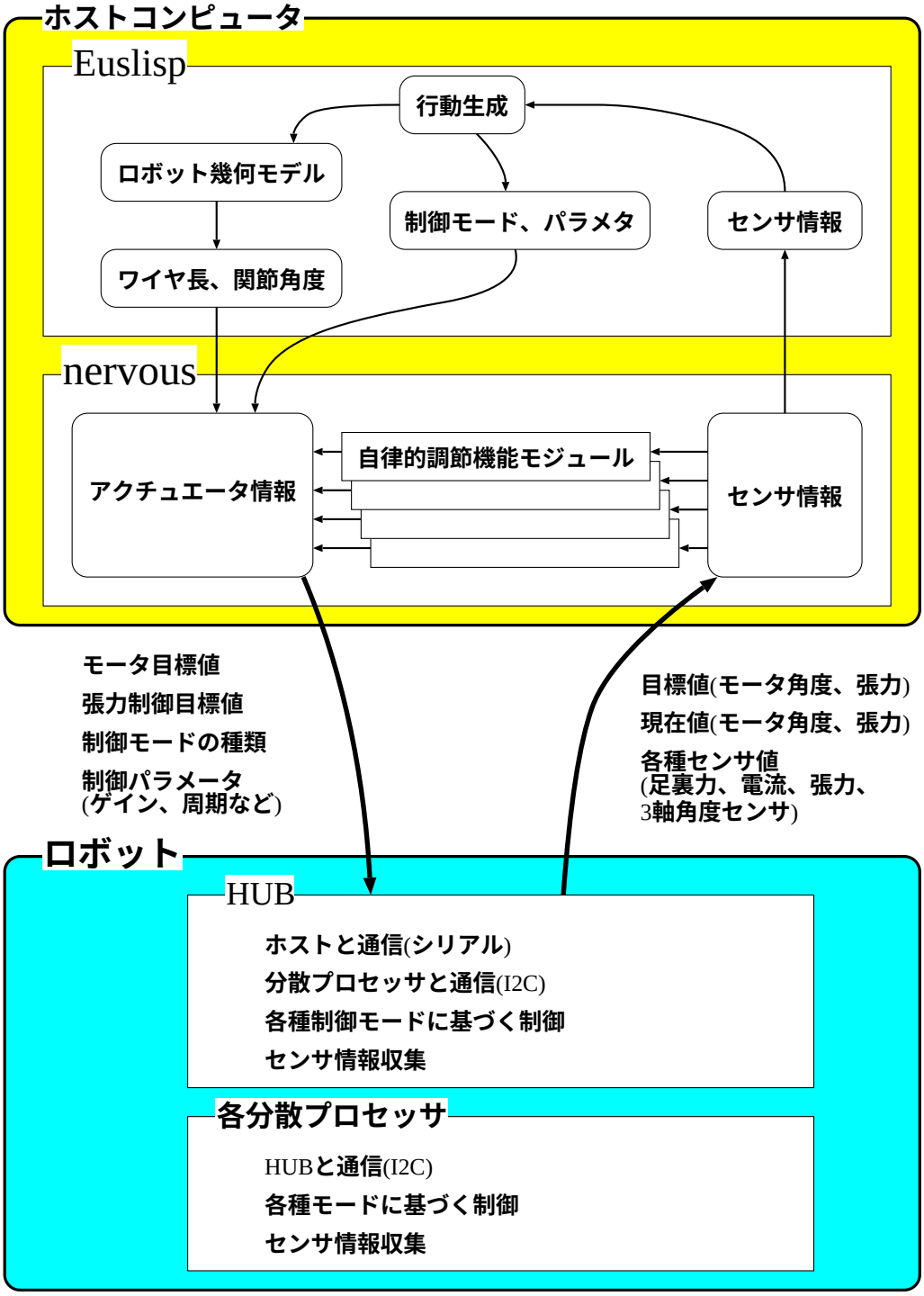



**Fig. 6.6** ソフトウェアシステム構成
Software structure

自律的診断調節系は，これらの各層に組み込みが可能である．

- 上位層に組み込み (イベント駆動型並列実行モジュール)

  上述の BeNet により，イベント駆動型並列実行モジュールのネットワークを構成することで，センサ入力周期にあわせて診断処理を行うモジュールを記述できる．張力センサ故障検出モジュールや過電流検出モジュールなど，多くの自立的診断調節機能モジュールは，まず始めにこの最上位層で BeNet のノード (BU: behavior unit) として記述・開発・調整し，仕様が固まると順に下の層に移植するという手順を採った．最上位層に組み込む場合の特徴は，ソフトウェアをフレキシブルに変えてゆけるという利点と，即応性が多少犠牲になるという欠点がある．最上位層の基本システムである EusLisp はインタプリタであるため，システム稼動中に機能を変更しシステムを停止することなく試行を行うことができるという，非常に優れた開発環境である．

- オンボディプロセッサに組み込み (組み込みソフトウェア)

  前項の特徴と正反対の特徴を有する．つまり，即応性 (1[ms] の制御周期毎に実行可能) という利点を持つが，劣悪な開発・デバッグ環境のためソフトウェアの改変は上位層のように容易ではないという欠点がある．この層に属する自律診断調節機能の例としては，モータ回転角度のリミッタ機能や，過電流防止機能などがある．

- 中間層に組み込み (プラグインアーキテクチャ)

  上記 2 項目の中間的特徴を有する． EusLisp の最上位層に組み込むよりは実行周期を見積もることが可能で，ある程度の即応性を得られるが，ソフトウェアの改変にはソースの編集と再コンパイルが必要である．ただし，プラグインはダイナミックロード・アンロードが可能な実装 (so: shared object や， dll: dynamic link library) になっているので，システム全体は停止せずにソフトウェアの変更・試行が可能である．

## 6.5 幾何モデルを中心とする従来型ロボットのシステムとの透過性

人間型全身行動ロボットのためのソフトウェア環境は，従来研究がある [26]．本節では，こうしたソフトウェア環境を用いる場合と同様な枠組みで，柔軟な脊椎を持つ全身型ロボットのソフトウェア環境を如何に構築するかに関し考察し，本研究で構築したソフトウェア環境を説明する．

### 6.5.1 従来の人間型ロボットのソフトウェアの拡張

ロボットのソフトウェア環境はオブジェクト指向に基づいて記述されている．ロボットの関節とアクチュエータの関係はドライバクラスにより記述され，その関係は基本的に **Fig. 6.7** 上部に示すように一対一の関係であった．今回の拡張，脊椎構造のように複数の関節を複数のアクチュエータにより動かすという多対多の関係になる．

**Fig. 6.8** に示すような新しいクラスを`actuator`，`driver`，`joint` の各クラスのサブクラスとして定義することで，従来のソフトウェアとの透過性を保ちながら，脊椎を持つロボットにも対応可能にした．図中で新しく定義したクラスには * 印を付してある．

なお， **Fig. 6.8** の下の矢印 backward-drive は 4.4.2節の式 (4.25) を利用し，図の上の矢印 forward-drive は 4.4.3節に述べた NN の計算に依っている．

これらの新しいクラスにより，脊椎を有する全身型ロボットの姿勢の記述は従来の人間型ロボットの姿勢の記述と同様の方法で記述することができるようになった．以下に，脊椎を有するロボットの行動の記述例を示す． 1 行目で nervous と接続， 2 行目でロボットのモデルのインスタンスを生成， 3 行目でモデル上で脊椎を動かす， 4 行目でモデル上で肩を動かす， 5 行目で実際のロボットをモデルの姿勢になるように 1000[ms] かけて動かす， 7 行目で実際のロボットの姿勢をモデルに反映させる．

```
1: (setq *ns* (open-nerve :io :host "host"))
2: (setq *sr* (spine-robot))
3: (send *sr* :spine :pitch 5)
4: (send *sr* :arms :shoulder-p :joint-angle 30)
5: (send *ns* :angle-vector (send *sr* :angle-vector) 1000)
6: ...
7: (send *sr* :angle-vector (send *ns* :angle-vector))
```

### 6.5.2 脊椎の各節を記述するクラス

球面関節のクラスの定義は次のように行った．これまで利用していた`joint` クラスを一般化し，これまでの 1 自由度の回転関節を`rotational-joint` として新たに`joint` クラスのサブクラスとして定義しなおした．そして，新しく`qjoint` クラスを作成した．

```
1: (defclass joint
2:   :super propertied-object
3:   :slots (parent-link child-link joint-angle
4:           driver min-angle neutral-angle max-angle)
5:   )
```

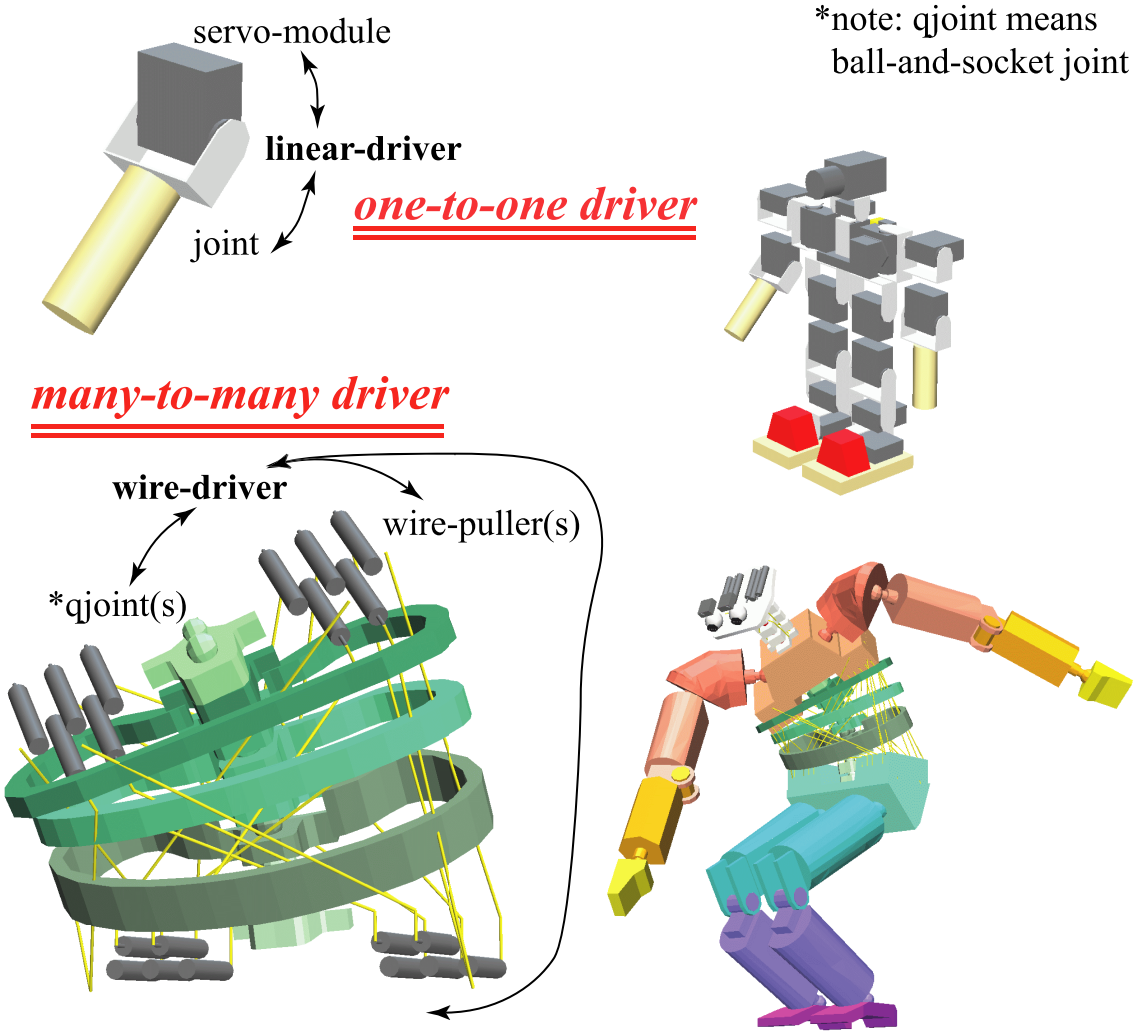



**Fig. 6.7** 従来型ソフトウェア環境との透過性
The transparency between current humanod software environment

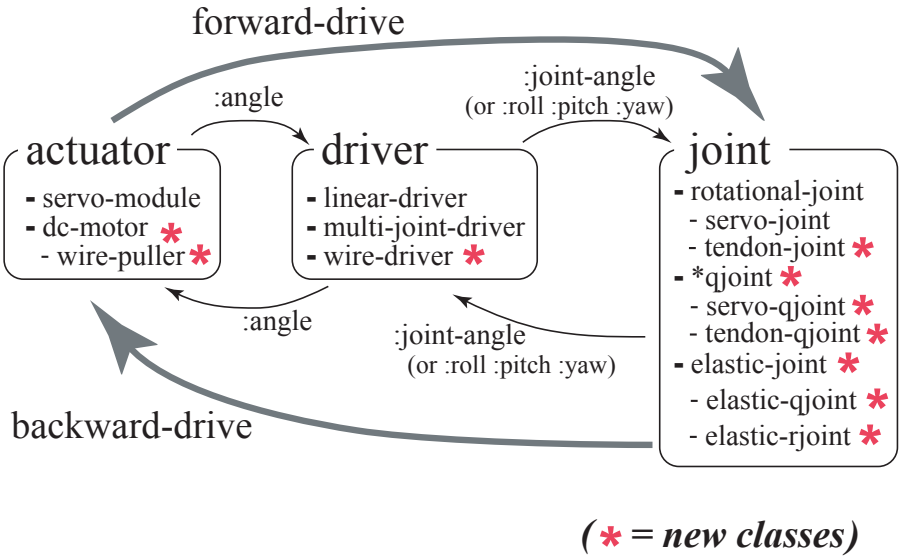



**Fig. 6.8** ドライバクラスの拡張
The expansion of driver classes

上記のプログラムコードの 1 行目から 5 行目までは，`joint` クラスの定義である．スロット変数として，親リンク・子リンク，関節角度，最大・最小・初期角度などが定義される (3,4 行目)． Lisp なので，それぞれの変数にバインドされる値は，数値またはオブジェクトである．例えば，親リンク・子リンクはそれぞれ `robot-link` クラスのオブジェクトで，関節角度などは， 1 自由度関節の場合は数値， 3 自由度の球面関節の場合は浮動小数点ベクトルなどを値に持つ．

```
 6: (defmethod joint
 7:   (:init (&key (name :joint)
 8:                ((:child-link clink))
 9:                ((:parent-link plink))
10:                (min) (max) (neutral)
11:                &allow-other-keys)
12:          (send self :name name)
13:          (setq parent-link plink child-link clink
14:                min-angle min neutral-angle neutral max-angle max)
15:          self)
16:   )
```

6 行目から 16 行目までは，`joint` クラスの初期化メソッド (`:init`) の定義である．与えられたキーワードパラメータ (7 行目`&key` 以降 10 行目まで) に基づいて各スロット変数に値を設

定する (12 〜 14 行目) が，関節の抽象クラスなので各変数に設定される値のデータ型 (数値かベクトルか他のオブジェクトか) は明示的にしていされていない.

```
17: (defclass rotational-joint
18:   :super joint
19:   :slots (axis
20:           )
```

18 行目から 21 行目は，1 自由度の回転関節のクラス (rotational-joint クラス) の定義を示す．joint クラスをスーパークラスとし (19 行目)，スロット変数に axis が加わっている (20 行目)．axis は，関節の回転軸の方向を表す :x,:y,:z もしくは回転軸の方向ベクトルを示す 3 次元浮動小数点ベクトルが与えられる.

```
21: (defmethod rotational-joint
22:   (:init (&rest args &key ((:axis ax) :-z) &allow-other-keys)
23:          (setq axis ax)
24:          (setq joint-angle 0.0)
25:          (send-super* :init args)
26:          ;; set default value
27:          (if (null neutral-angle) (setq neutral-angle 180))
28:          (if (null min-angle) (setq min-angle -90))
29:          (if (null max-angle) (setq max-angle (+ 180 min-angle)))
30:          self)
31:   (:joint-angle
32:    (&optional v &key relative (reflect t))
33:    (when v
34:      (let ((relang))
35:        (cond
36:         (relative
37:          (setq relang v)
38:          (setq joint-angle (+ v joint-angle))
39:          )
40:         (t
41:          (setq relang (- v joint-angle) joint-angle v)))
42:        (send child-link :rotate (deg2rad relang) axis))
43:      (if (and reflect driver) (send driver :backward-drive self))
44:      )
45:    joint-angle)
46:   )
```

rotational-joint クラスの初期化メソッド :init (23 〜 31 行目) では，スロット変数 joint-angle に初期値として数値である 0.0 がセットされる (25 行目)．neutral-angle，min-angle，max-angle 等も同様にデフォルト値として数値が設定される (28,29,30 行目)．関節の値 (joint-angle) へのアクセッサメソッドである:joint-angle の定義 (32 〜 46 行

目) では，回転角度がオプショナル引数として与えられると (32 行目でv≠nil)，子リンク (child-link) は，回転軸axis 回りにv 度回転する (42 行目).

```
47: (defclass qjoint
48:   :super joint
49:   :slots ()
50:   )
```

これに対し，3 自由度の球面関節を定義するqjoint クラス (47 〜 50 行目) では，rotational-joint クラスと同様にjoint クラスのサブクラスとして定義されているが (48 行目)，axis スロット変数は追加されていない (49 行目).

```
51: (defmethod qjoint
52:   (:init
53:    (&rest args)
54:    (setq joint-angle (float-vector 0 0 0))
55:    (send-super* :init args)
56:    (if (null neutral-angle) (setq neutral-angle (float-vector 0 0 0)))
57:    (if (null min-angle) (setq min-angle (float-vector -90 -90 -180)))
58:    (if (null max-angle) (setq max-angle (float-vector 90 90 180)))
59:    self)
60:   (:joint-angle
61:    (&optional v)
62:    (unless v
63:      (return-from :joint-angle joint-angle))
64:    (setq joint-angle v)
65:    ;; 子リンクの姿勢
66:    (when child-link
67:      (send child-link
68:            :rpy
69:            (deg2rad (elt joint-angle 2))
70:            (deg2rad (elt joint-angle 1))
71:            (deg2rad (elt joint-angle 0))))
72:    joint-angle)
73:   )
```

初期化メソッド :init(52 〜 59 行目) では，スロット変数joint-angle に初期値として浮動小数点ベクトル $(0, 0, 0)$ が設定されている (54 行目)．neutral-angle，min-angle，max-angle 等も同様に，デフォルト値として数値ではなく浮動小数点ベクトルが設定される (56 〜 58 行目)．そして，関節の値 (joint-angle) へのアクセッサメソッドである :joint-angle の定義 (60 〜 72 行目) では，オプショナル引数v は roll, pitch, yaw の 3 つの角度を要素とする浮動小数点ベクトルで与えられ，child-link は，その roll, pitch, yaw の姿勢に回転される (67 〜 71 行目).

### 6.5.3 筋を記述するクラス

脊柱全体を駆動するワイヤおよびアクチュエータも，これまでの EusLisp によるモデル化環境 [26] には定義されていなかった．しかし，脊柱の姿勢制御にはワイヤの長さの計算は不可欠で，できればワイヤ張力に関するモデル，椎間板のゴムの力に関するモデルも定義されていることが望ましい．そこで，ワイヤを表現するwire クラスおよびgeneral-wire クラスを定義した．

```
 1: (defclass wire
 2:   :super propertied-object
 3:   :slots ((pv-list    :type list)    ;; 引張り点の list
 4:           (qv-list    :type list)    ;; 取付け点の list
 5:           (joint-list :type list)    ;; 駆動される joint の list
 6:           (force      :type integer) ;; 張力 [gf] (常に > 0 )
 7:           )
 8:   )
```

1 〜 8 行目はwire クラスの定義である．pv-list， qv-list は，それぞれ引っ張り点 (float-vector(浮動小数点ベクトル))，取付け点 (float-vector) のリストである．連結リンクを 1 から n とすると，i 番目のリンクの取付け点 (pv) はpv-list の i 番目 (i-1 番目のリンクの座標系から見る)，i 番目のリンクの引っ張り点 (qv) はqv-list の i 番目 (i 番目のリンクの座標系から見る) である．下記 (10 〜 29 行目) に初期化メソッド (:init) の定義を示す．pv-list とqv-list は，それぞれ joint-list と同じ要素数でなければならない．

```
 9: (defmethod wire
10:   (:init (sj-list &rest args
11:           &key ((:force f) 0) ((:pv-list pvl)) ((:qv-list qvl))
12:           &allow-other-keys)
13:    (let ()
14:      (setq joint-list sj-list)
15:      (setq force (if (> f 0) f 0))
16:      (if (and (= (length pvl) (length joint-list))
17:               (= (length qvl) (length joint-list)))
18:          (progn
19:            (setq pv-list pvl)
20:            (setq qv-list qvl))
21:        (progn
22:          (error "ERROR: pv and qv must be same length with joint-list.~%")
23:          (setq pv-list (make-list (length joint-list)
24:                                   :initial-element (float-vector 0 0 0)))
25:          (setq pv-list (make-list (length joint-list)
26:                                   :initial-element (float-vector 0 0 0))
27:                qv-list (make-list (length joint-list)
28:                                   :initial-element (float-vector 0 0 0)))))
29:      self))
30:   )
```

```
31: (defmethod wire
32:   (:jacobian-vector ()
33:    (let (rqv ret tmp tmatrix)
34:      (do*
35:       ((tempp pv-list (cdr tempp))
36:        (tempq qv-list (cdr tempq))
37:        (tempj joint-list (cdr tempj))
38:        (p (car tempp) (car tempp))
39:        (q (car tempq) (car tempq))
40:        (j (car tempj) (car tempj)))
41:       ((null tempp) nil)
42:       (if (null j) ;; ワイヤに関係ない joint のとき
43:           (push 0 tmp)
44:         (progn
45:           (setq tmatrix (send j :child-link :rot))
46:           (setq rqv (transform tmatrix q))
47:           (if (= (send j :dof) 1) ;; 回転関節のとき
48:               (push
49:                (elt (scale (/ 1 (norm (v- p rqv))) (v* p rqv)) 2)
50:                tmp)
51:             ;; 球面関節のとき
52:             (push
53:              (scale (/ 1 (norm (v- p rqv))) (v* p rqv))
54:              tmp)
55:             ))))
56:      ;; push で逆順になったので直す
57:      (nreverse tmp)
58:      ;; float-vector にまとめる
59:      (dolist (v tmp)
60:        (if (float-vector-p v)
61:            (setq ret (concatenate float-vector ret v))
62:          (setq ret (concatenate float-vector ret (float-vector v)))
63:          ))
64:      ret))
65:   )
```

上記 32 〜 64 行目は，関節トルクと正負を考慮した張力の関係 $(-G^t)$ からヤコビ行列を求め，自身に対応する $G$ の行ベクトルを返すメソッド:jacobian-vector の定義である，これを利用して，張力 (下記 67 行目:force メソッドにより設定する) の値から，:yacobian-vector を利用して，joint-list の各関節に発生するトルクの値を計算して返すメソッド :torque-vector が定義されている．

```
66: (defmethod wire
67:   (:force      (&optional v) (if v (setq force (if (> v 0) v 0)) force))
68:   (:torque-vector () (scale force (send self :jacobian-vector)))
69:   )
```

さらに，:length メソッド (下記 71 〜 85 行目に定義) は，各経由点間の現在のワイヤの長

さの和を求めるメソッドである．

```
70: (defmethod wire
71:   (:length ()
72:    (let (rqv (ret 0) tmatrix)
73:      (do*
74:       ((tempp pv-list (cdr tempp))
75:        (tempq qv-list (cdr tempq))
76:        (tempj joint-list (cdr tempj))
77:        (p (car tempp) (car tempp))
78:        (q (car tempq) (car tempq))
79:        (j (car tempj) (car tempj)))
80:       ((null tempp) nil)
81:       (setq tmatrix (send j :child-link :rot))
82:       (setq rqv (transform tmatrix q))
83:       ;; (format t "rqv =  ~A~%" rqv)
84:       (setq ret (+ (norm (v- rqv p)) ret)))
85:      ret))
86:   )
```

次に，general-wire クラスの定義 (〜行目) と初期化メソッド:init(下記 1 〜 4 行目) を下に示す．スロット変数 pv-list, qv-list の要素は，float-vector ではなくcascaded-coords クラスのオブジェクトである．cascaded-coords クラスは，連結座標系を表現するクラスで，:parent メソッドで (または，初期化時にキーワード引数:parent で) 親座標系を指定すると，それ以降親座標系の移動や回転に伴い自動的に移動・回転するようになる．

```
1: (defclass general-wire
2:   :super wire
3:   :slots ()
4:   )
```

下記 6 〜 26 行目は general-wire クラスの初期化メソッド (:init) である．pv-list，qv-list に，float-vector のリストが渡されると，それが示す位置のcascaded-coords クラスのオブジェクトを生成し，joint-list の対応する joint の child-link に結合される (14 〜 16 行目および 20 〜 22 行目)．cascaded-coords クラスのオブジェクトをpv-list，qv-list に引数として渡す場合，それが取りつけられている物体に連結されているオブジェクトを渡す必要がある．

```
 5: (defmethod general-wire
 6:   (:init
 7:    (js &rest args &key ((:pv-list pvs)) ((:qv-list qvs))
 8:        ((:force f) 0) name &allow-other-keys)
 9:    (unless (= (length pvs) (length qvs))
10:      (error "the length of pv and qv must be the same"))
11:    (if (vectorp (car pvs))
12:        (dotimes
13:         (i (length pvs))
14:         (setf (elt pvs i)
15:               (make-cascoords :pos (elt pvs i)
16:                               :parent (send (elt js i) :child-link)))))
17:    (if (vectorp (car qvs))
18:        (dotimes
19:         (i (length qvs))
20:         (setf (elt qvs i)
21:               (make-cascoords :pos (elt qvs i)
22:                               :parent (send (elt js i) :child-link)))))
23:    (setq pv-list pvs qv-list qvs joint-list js)
24:    (setq force (if (> f 0) f 0))
25:    (send self :name name)
26:    self)
27:   )
```

メソッド:length(下記 29 〜 35 行目) はワイヤの全長を求めるメソッドである．:bodies メソッド (下記 37 〜 43 行目) は，ワイヤの可視化のためのメソッドで，そのワイヤの経路に沿って半径radius(キーワード引数，デフォルト値は 1， 37 行目) の円筒のリストを生成し，それを戻り値として返す． 40 行目で使われている関数 make-point-to-point-cylinder の定義は，下記 45 〜 57 行目に示す．

```
28: (defmethod general-wire
29:   (:length ()
30:    (let* ((ret 0))
31:      (dotimes
32:       (i (length pv-list))
33:       (incf ret (distance (send (elt pv-list i) :worldpos)
34:                           (send (elt qv-list i) :worldpos))))
35:      ret))
36:   ;;
37:   (:bodies (&key (radius 1))
38:    (let* (bs b axis theta normal)
39:      (dotimes (i (length pv-list))
40:               (push (make-point-to-point-cylinder
41:                      (elt pv-list i) (elt qv-list i) :radius radius)
42:                     bs))
43:      (nreverse bs)))
44:   )
```

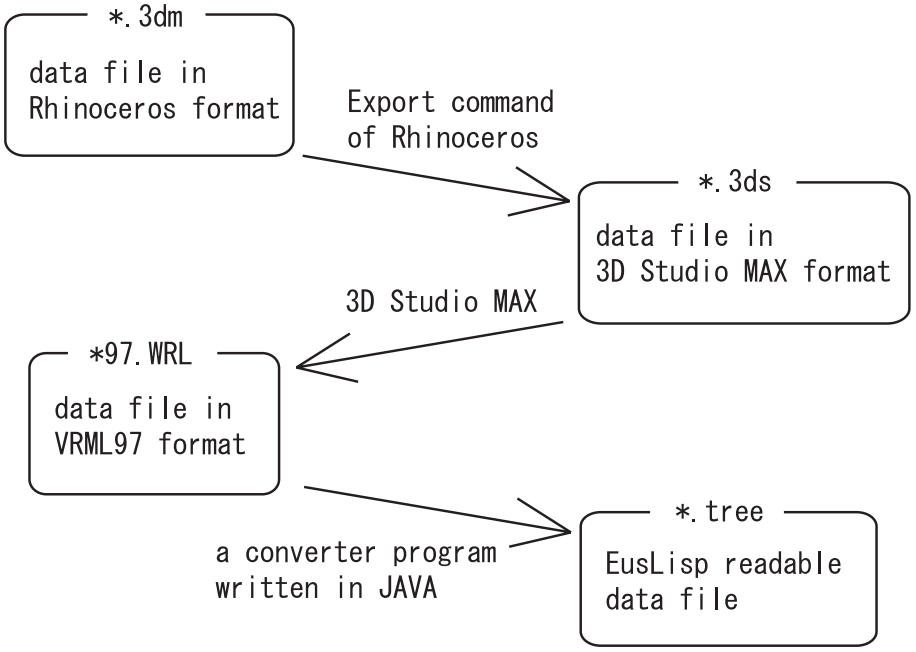



**Fig. 6.9** CAD データの変換
conversion of CAD data files

```
45: (defun make-point-to-point-cylinder
46:   (p1 p2 &key (radius 1))
47:   (let* (b axis theta normal)
48:     (setq b (make-cylinder radius (distance (send p2 :worldpos)
49:                                             (send p1 :worldpos))))
50:     (send b :locate (send p1 :worldpos) :world)
51:     (setq axis (v- (send p2 :worldpos) (send p1 :worldpos)))
52:     (unless (= 0.0 (norm (setq normal (v* (float-vector 0 0 1) axis))))
53:       (setq normal (normalize-vector normal))
54:       (setq theta (vector-angle (float-vector 0 0 1) axis normal))
55:       (send b :rotate theta normal)
56:       )
57:     b))
```

### 6.5.4 幾何モデルの作成

体幹部の制御においては，ワイヤ長やワイヤ張力の決定のために，体幹部を含めて幾何学的な構造をコンピュータ上のモデルとして作成することが必要である．これまでに，研究室内で開発・利用されてきたロボットの幾何学的モデルおよび動力学的モデルの環境 [26] を，本研究における柔軟性を持つ多節構造などの構造も扱えるように拡張することで，過去に開発され

た膨大なソフトウェアライブラリの大部分を利用可能になる．これらのソフトウェアは主にオブジェクト指向の Lisp である EusLisp [50] で記述されている．

これまでに新しいロボットが製作された際には，その都度そのロボットの物理形状に基づいて，モデルを記述したファイルを作成していた．今回製作したロボット Cla もそうした方法でモデルを作成することもできるが，複雑な形状を持つ部品を EusLisp 上でもう一度作り直すのではなく，三次元 CAD ソフトウェアである Rhinoceros で作成した部品の形状を EusLisp に変換[2)]することができると効率が非常に良い．今回，Cla のモデル化においては，Rhinoceros 上のデータを利用した．Rhinoceros には，DXF 形式や IGES 形式などの汎用な CAD 形式での出力 (エクスポート) 機能があるが，EusLisp で直接読み込める形式の出力機能は無い．一方，VRML97 形式のファイルを EusLisp で読み込める形式に変換する JAVA プログラム (付録 Dにソースを添付) が利用可能だったので，そこで，Rhinoceros から 3D Studio MAX [3)]形式で出力し，それを 3D Studio MAX から VRML97 形式で出力し，最後に JAVA プログラムで変換するという方法をとった ( **Fig. 6.9** ).

[2)] EusLisp 上では各部品は`body` クラスのインスタンスで表される．

[3)] http://www.ktx.com/3DSMAX/

# 第 7 章

# 結論

## 7.1 結論

本研究では，従来のヒューマノイドなどのロボットの問題点として，(1) ボディの硬さ・動きの硬さ，(2) 冗長性の多くない自由度構成，(3) 想定されない状況には対応できないようなソフトウェア構成，があると考えた．そして，これらの問題を解決するためには全身型ロボットの身体構成を根本的に見直す必要があると考えた．このような視点から，本研究では，全身型ロボット・人間型ロボットに柔軟構造を組み込むことを提案した．特に，人間の脊椎構造の効果に着目し，柔軟性可変な脊椎構造を多自由度全身型ロボットに組み込むことに取り組んだ．脊椎構造を組み込むことにより，上記の (1) 〜 (3) の問題に対し，(1) 柔らかいボディ・柔らかい動きを目指す，人間の脊椎構造の効果を全身型ロボットに持たせる，(2) 冗長性と脊椎の自由度により多様な動作・自然で効率的な動作の実現を目指す，(3) できるだけ多くの状況に対応できるように多種多数のセンサ・アクチュエータを搭載し，なおかつ不完全な状態でも行動が可能な枠組みを目指す，という解決方法を提案した．脊椎構造を有することにより，自由度の増加 と 柔軟性 という利点が得られる．自由度の増加により，狭い場所での作業・効率の良い動作，(人間に近い) 自然な動作などが可能になる．柔軟性を持つことにより衝撃吸収などの人間や環境との接触における安全性などが得られる．また，脊椎は上体を支える支持構造の役割も有するため，常に柔軟ではなく状況に応じて柔軟性を変えられることが必要である．

本研究では，柔軟性可変な脊椎構造を有する全身型ロボットの，

1. 脊椎構造 (体幹) のあり方
2. 脊椎構造の駆動系の構成法
3. 脊椎を利用する全身行動の生成法，動作・行動の制御法・拡張法
4. 脊椎を利用する全身行動のためのロボットシステム構成法

に関し，課題を整理し，解決への道筋を示すことを目的とした．上記の課題をどのように整理し解決への道筋を示したかを以下の各節に述べる．

### 7.1.1 脊椎構造のあり方

脊椎構造のあり方に関しては主に第 3章で論じた．第 2章における可変柔軟性を持つ構造の従来研究について概説・考察や人間の脊椎構造の解析をふまえた上で，可変な柔軟性を有する脊椎構造の設計と実際の製作を通し知見を整理し，全身型ロボットのための脊椎構造のあり方

を示した． **Fig. 7.1** に本研究で設計・製作・実験を行ったロボットを示す．第 3章で得た結論をまとめると以下のようになる．

- 冗長性・多様性を実現する多節構造

  連続した球面関節からなる多節構造にすることで，変曲点を持つような姿勢を可能にする．体をよじって障害物を避けるような姿勢や，机の下にもぐり込むような姿勢が可能になる．

- 椎間板による基本姿勢と復元力 (重力を前提)

  各節の間に初期圧力をかけた状態の弾性要素 (椎間板に相当) を挟むことにより，基本姿勢を持ち変形すると元に戻ろうとする復元力が生じる．直立した姿勢を基本姿勢とすると，基本姿勢は重力のポテンシャルエネルギーが高い状態になり，姿勢を変形した状態から基本姿勢に向かう場合にアクチュエータに必要とされる力を補助する形で復元力が生じる．

- 多くの筋による駆動

  多節構造で変曲点を有するような姿勢も実現するためには，途中節に留められるような筋も有する必要がある．同じ方向に変形する場合は各筋は協調しあうこともできる．すなわち，多くの筋により駆動することで，姿勢の多様性と，筋力の協調が可能になる．

- 張力センサを備えた筋による拮抗駆動 (柔軟性調節)

  脊椎は上体の支持構造の役割も有し状況に応じて柔軟性を変えられることが求められる．そのために，柔軟性を調節できる筋により駆動する．メカニカルに筋の柔軟性を調節できるようにする機構は幾分複雑になるので，多くの筋により駆動するため，様々な制御法に柔軟に対応できるために，張力センサを備えた筋により駆動する構成とする．

### 7.1.2 脊椎構造の駆動法

脊椎構造の駆動法に関しては主に第 4章で論じた．問題の本質は，制御量は何か，制御法は何かという点である．従来のロボットの制御は関節角度や手先のデカルト座標系における位置・姿勢等の制御量を用いて記述されていた．人間の場合は関節中心が明確でなく，こうした制御量よりもむしろ，タスク依存で自己センサ空間表現の制御量を利用していると考えられる．脊椎構造を有する全身型ロボットの場合も，身体構造は人間に近づいてゆくと考えられる．また，直接教示・再生による制御も有効である．そこで本研究では，脊椎構造の駆動法として，直接教示・再生による制御，幾何学的制御量をある程度の制御できる方法論，センサ

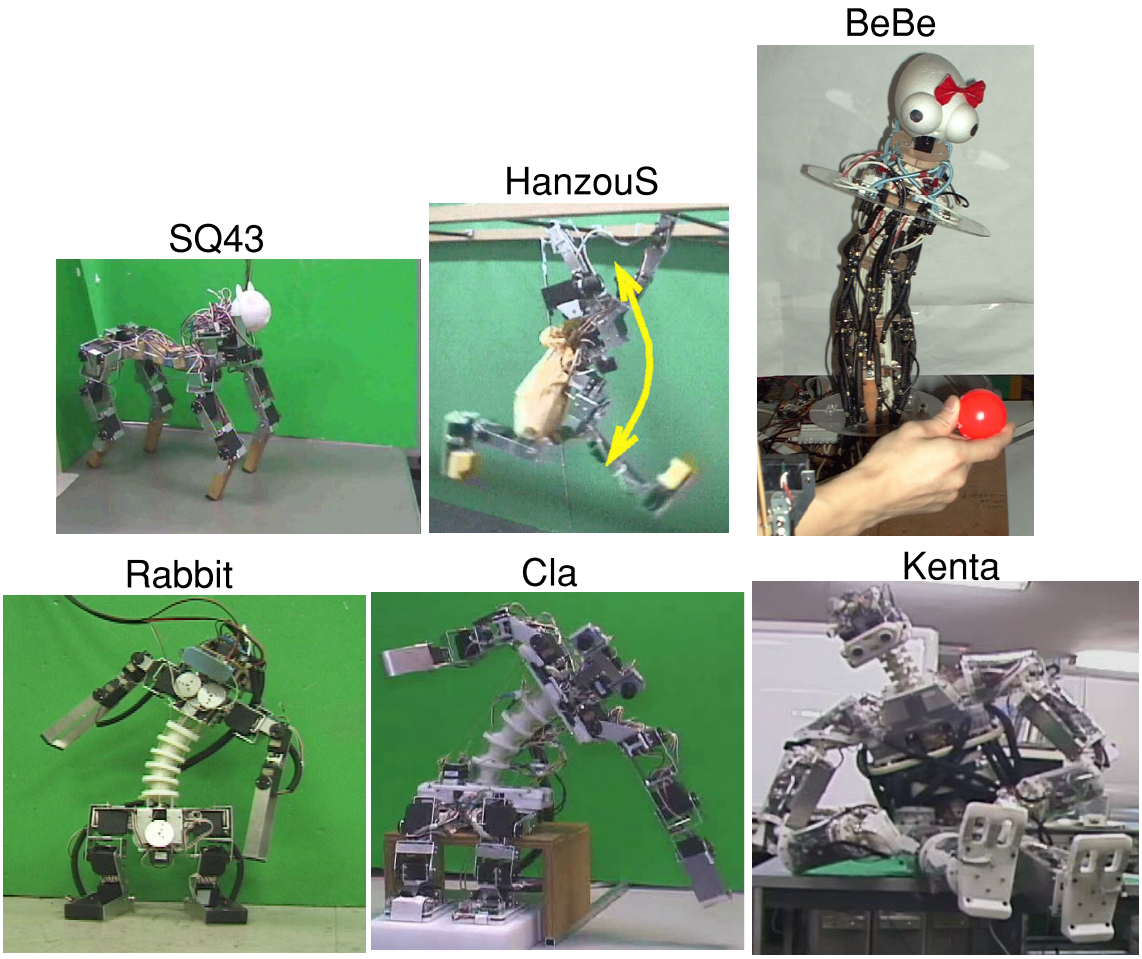



**Fig. 7.1** The developed robots in this research
本研究で扱ったロボット

フィードバックによるタスクの実現，などを示した．第 4章 で得た結論をまとめると以下のようになる．

- 筋制御モードの切り替えによる姿勢・動作の教示・再生

  全筋を一定張力に保つような制御を行うことにより，ロボットを直接触って姿勢を教示することができる．姿勢が変わっても各筋は張力が一定になるように，緩みや張りを無くすよう制御される．この時の各筋の長さを記録しておき，長さ制御に切り替えることで，教示した姿勢・動作の再生が可能である．

- 幾何モデルによる筋長計算

干渉を考慮しない幾何モデルによる姿勢→筋長の変換と，それを教師データとしニューラルネットにより学習した筋長→姿勢の変換を実現した．これらにより，計算機上での姿勢・動作・行動の作成と，姿勢センサに基づく制御の時などに筋長情報から現在の姿勢の推測が可能になった．

- 複数の筋制御モードにより筋の干渉問題を解決

  前項の幾何モデルの計算は筋同士や筋と構造材の干渉を考慮していないため，過大な張力が発生する危険がある．これを回避するために，張力センサを利用し，状況に応じて各筋の制御モードを変えることで，過大な張力の発生を抑えつつできるだけ姿勢を目標どおりに制御する方法を示した．

- ばね制御による脊椎の柔軟性調節

  脊椎構造の剛性行列を定義し，筋長変化と姿勢変化の微小量の間の関係を表す筋長ヤコビアンを用いて，筋の柔軟性 (弾性係数) と脊椎構造の柔軟性の間の変換を示した．モータの非線形性を考慮し，張力センサを用いて筋がばねの挙動を示すような制御法を提案し，これにより脊椎構造の柔軟性 (剛性行列) を調節する手法を示した．

- 幾何モデルを用いないセンサベーストな制御法

  幾何学的制御量を用いずにセンサベーストな制御法として，24 節の医学用脊椎モデルを 36 本の空気圧人工筋の on,off により駆動する脊柱型ロボット BeBe を製作し，汎用性のあるアルゴリズムにより幾何モデルを利用せず視覚フィードバックのみを用いた物体のトラッキング動作実験を行い，センサフィードバックのみによる制御が可能なこと・冗長性により外乱や制約条件があってもタスク遂行が可能なことを示した ( **Fig. 7.1** ).

### 7.1.3 脊椎を利用する全身行動

第 5章では，全身動作・全身行動の生成法およびその制御法を論じた．人が試行錯誤により動作・行動を作成するために必要な枠組み (インタフェース)，シミュレータと GA を用いる動作生成方法，姿勢データベースというしくみの提案・構築とそれに基づく全身行動の生成法・制御法などを示した．シミュレーションと GA あるいは NN を用いて全身動作を生成することで，脊椎構造を持つ全身型ロボットという挙動予測がしづらいロボットにおいて，実機での動作可能なパターンに近い解を得られるということだけでも大きな有効性がある．

- 従来の人間型ロボットのソフトウェアと透過性のある行動記述

姿勢・動作・行動の記述法を従来の人間型ロボットとの透過性を有するように，関節角度列による姿勢表現を用いて記述できるようにした．この環境を利用して従来のロボットの行動記述法と同様の方法で，脊椎を利用して床の物を拾う行動・脊椎を利用して物体を両手で投げる行動・脊椎を利用した寝返り行動などを実現した．また，姿勢センサや足裏力センサを利用した姿勢制御行動・視覚センサを利用した物体追跡行動などの，センサ情報を利用する行動も実現した．

- 三種類の仮想環境を用いた探索による動作生成

  脊椎構造の挙動をシミュレートするために，有限要素法ソフトウェア ANSYS を用いた手法，動力学解析ソフトウェア DADS を用いた手法，ゲーム用高速動力学演算ライブラリ MathEngine の三種類の市販ソフトウェアを，解析時間と精度のトレードオフを考慮して必要な精度・計算時間に応じて使い分けることが有効であることを示した．この環境を用いて， GA による動作パターンの生成， GA による NN コントローラの最適化，の二種類の手法により脊椎を有する全身型ロボットの全身行動を生成した．

- 姿勢データベースによる直接教示・再生・行動空間拡大の枠組み

  行動及び動作の制御及び拡張のための枠組みとして，姿勢データベースト呼ぶ仕組みを提案した．筋長空間または姿勢センサ空間でセグメンテーションし，データベースのいずれの要素からも一定距離以上離れたデータは，新しい要素としてデータベースに自動的に追加され，ある目標姿勢 (状態) が与えられたら現状からそこへ至る経路を，データベースの要素を辿る形で求めるという枠組みである．

### 7.1.4 脊椎を有する全身型ロボットシステムの全体構成法

第 6章では，脊椎を有する全身型ロボットシステムの全体構成法を論じた．分散配置オンボディプロセッサにより構成される体内 LAN，インタプリタ・オブジェクト指向などの特徴を持つ EusLisp により構成される上位層，この二つの層を結ぶ中間層である nervous，という 3 つの層からなる基本的な階層構造による構成を，基本構成として示した．不完全性に対応可能な仕組みとして，自律的診断調整機能モジュールを提案・実装した．最上位層には，従来の人間型ロボットのソフトウェア環境と透過性を有するシステムを構築した．

- 従来型ソフトウェア環境と透過性を持ったシステム

  従来の人間型ロボットでは関節姿勢 (角度) とアクチュエータ変位は 1 対 1 に対応していたが，脊椎の場合複数の関節を複数のアクチュエータ (筋) で駆動するという構成になる．第 4章に述べた計算法を利用し，関節角度列とアクチュエータ変位列の関係を求める

ドライバと呼ばれるクラス (システムはオブジェクト指向である) を新たに定義することによりこれを実現した.

- 体内 LAN による多入力・多出力システムの実現

  マイクロプロセサをボディに分散配置し，局所的にセンサ情報収集・アクチュエータ制御を行い，各プロセッサを高速のマルチチップのシリアルバスで結ぶことにより体内 LAN を構成．体内の HUB がバスマスタの役割を担うことで，分散プロセッサの一部が故障してもシステム全体は動きつづけることが可能なシステムとした． HUB にデータを集約し遠隔のホストと通信することで，多種多数のセンサ・アクチュエータを有するロボットにおいても，省配線・拡張性を実現した.

- ハードウェア抽象化の機能を有する中間層による多種ロボットに対応したシステム

  様々なロボットのハードウェア構成に設定ファイルにより対応し，小型のロボットから腱太のような大規模・複雑なロボットでも利用できるシステムを構築した．また，中間層にプラグインアーキテクチャにより拡張性を持たせ，即応性と開発環境をそれぞれ程よく満たした環境を構築した.

- 自己状態診断・修正機能 (自律調節系)

  不完全性に対応可能な仕組みとして，自律的診断調整機能モジュールを提案・実装した．この自律機能は，上位層の EusLisp 上では BeNet の形で，中間層の nervous ではプラグインの形で，下位層の分散オンボディプロセッサでは 1[ms] 周期の組み込みソフトウェアの形で，各層に実装可能とした．即応性と開発環境のトレードオフあるいは仕様の流動度で，どの層に実装するかを選択する.

## 7.2 展望

本研究で取り組んだのは，複雑な身体構造を製作して，いかにそれを動かすかという問題である．今後の方向性として，腱太のような，更に進んで人間のような，複雑な構造をいかに扱えるようにしてゆくかということだと考えられる．それは，身体構造をモデル化，単純化するという方向性ではなく，最終的に目指すゴールは人間並みのソフトウェアである．人間は腱太よりはるかに複雑性の高いボディをしっかりと意思の通りに扱っている.

一つには，人間はセンサの種類・性能・密度が非常に高いということが理由として挙げられる．ロボットの身体構造がこのような方向に進化してゆくと，ソフトウェアとしてはセンサ情報空間に基づいたモデルのようなものを自己形成するという枠組みが求められると考えられ

る．そして，そのセンサベーストモデルは，行動のために利用されつつ，常に更新されてゆくような仕組みが必要となろう．

複雑な全身行動の実現方法も課題である．本研究では，複雑な全身行動を実現するための様々な手法を示したが，完成度の面ではまだまだである．シミュレーションの探索過程におけるパラメータの変化の様子を解析することで，実機でのパラメータ調整の指針のようなものを得ることができる枠組みが可能なのではないだろうか．すなわちシミュレーションと実環境の間に差はあるが，パラメータの改良過程には類似性があることが期待できるという可能性を追求することも今後の課題である．

複雑な全身行動の実現方法としてもう一つ考えられるのは，異なるロボット間 (あるいは人間ロボット間) での，行動情報の共有である．単純な身体構造のロボットによる行動から複雑な身体構造のロボットの行動への変換方法や，複雑な身体構造を巧みに扱う方法を会得している人間の行動情報を変換する方法が実現できることが複雑な全身行動の実現方法の一つとして考えられる．

よりデペンダビリティの高いハードウェアの構築法も課題として挙げておく．本研究では，ハードウェアの構成法に関して脊椎構造のあり方を一定のレベルで示したが，更に軽量・多自由度で壊れにくいの脊椎構造の構成法の研究が求められる．

最後に，本研究が全身型ロボット研究の進歩における新しい一歩となることを願う．

# 謝辞

本論文は筆者が東京大学大学院 工学系研究科 機械情報工学専攻 の情報システム工学研究室において，稲葉雅幸教授の御指導のもとで行なった研究をまとめたものです．

稲葉教授には，1996 年に研究室に入った時から，学会発表，修士論文，博士論文と，大変お世話になりました．研究の進め方，研究に対するポリシーを構成するための考えの進め方など，直接・間接に御指導いただきました．脊椎構造というユニークな研究を提案していただき，このような形でまとめることができるまで，至らない筆者を厳しくご指導くださり，本当にどうもありがとうございました．また，組み込み用コントローラのソフトウェア・電子基板から知能のレベルを実現する Lisp 言語・3 次元 CAD など，本当に様々な技術的な経験を積む機会を与えてくださいました．当初，こうした様々な技術的な事項を軽んじて考えがちだった筆者に，自分の関わる全ての技術に深く接することの重要性を認識させて下さいました．

同じ研究室の教官である井上教授 (東京大学) にも大変お世話になりました．筆者が博士課程に進学し，修士課程のテーマからこの博士論文のテーマに変える時に，悩んでいた筆者の背中を稲葉教授と共に押してくださいました．毎週の研究会や，午前中に実験室にいらした時などに，筆者が返答に窮するような鋭いご質問やご意見を下さり，ともすれば視野が狭くなりがちな筆者に様々な思考のきっかけを与えてくださいました．どうもありがとうございます．

佐藤知正教授 (東京大学)，中村仁彦教授 (東京大学)，下山勲教授 (東京大学) には，本論文をまとめるにあたり，何度もお時間をいただき，非常に有意義なご意見をいただきました．本論文を仕上げるにあたり，論点の整理・つめの甘さを指摘していただきました．また，中村教授，下山教授には，本研究と密接に関わりのあった未来開拓研究「マイクロ・ソフトメカニクス統合による高度生体機能機械の研究」のメンバーとして，研究会等でも様々なご意見・議論をいただきました．どうもありがとうございます．同研究プロジェクトの，広瀬茂男教授 (東京工業大学)，岡田昌史さん (当時東京大学リサーチアソシエート) にも，研究会等で鋭いご意見をいただいたり議論をしてくださいました．どうもありがとうございます．

研究室の OB の方々にも大変お世話になりました．金広文男さん，加賀美聡さん，長嶋功一さん，柴田智広さん，現在は同じ研究室の助教授になられました國吉康夫助教授，これらの方々にも，機会あるごとに研究に対するご意見をいただいたり，研究以外の面でも色々とお世話になりました．ありがとうございました．

最後の一年間でしたが，脊椎を持つ人間型ロボット Cla をテーマに研究を進めてきた吉田成徳君には，大変お世話になりました．締切りに追われないとペースがあがらない筆者に付合って，IROS2001 の出発前丸 2 日間以上全く眠らずに一緒に実験をやってくれました．その他にも数えきれないほど色々と助けになってもらいました．本当にどうもありがとう．

腱太グループの，但馬竜介君，吉海智晃君，佐藤大輔君に感謝いたします．長嶋さん，稲葉教授とともに 2001 年 3 月の発表前の苦しいときに共に頑張ったことはいい思い出・経験に

なるでしょう.

これまでに同じグループになった後輩の方々に感謝いたします．松木健君，川島俊嗣君，原朗人君は， SQ43， BeBe， Rabbit の各ロボットの研究を進めてすばらしい修士論文や卒業論文を仕上げられました．ともに議論し壁を乗り越えてゆけた経験を誇りに思います．そして，椛澤光隆君，和井田寛則君，クァン君とも同じグループ・同じ実験室で研究を進め，色々と意見をいただきました．みなさんどうもありがとう．みんなに共通するキーワードは何かと考え思いついたのが「しなやか」という単語でした．本当に「しなやか」なロボットの実現はまだまだ先の長い道のりだと思います.

修士課程から同じ学年であった同期の方々に感謝いたします．大変楽しいメンバー，できるメンバーが揃っており，研究でもプライベイトでも刺激のある大変楽しい時間をすごせたと思います．どうもありがとう．博士課程でも同じ学年だった星野由紀子さん，香山健太郎君とは，苦しい時に励ましあうことができ大変ありがたかったと思います．田宮幸春君，坂井克弘君，吉池孝英君，小屋迫光太郎君，野田卓郎君とは修士課程の 2 年間を共にすごし，研究に遊びにと活動的な時間を共有できました.

筆者の研究の環境を支えてくださった研究室のスタッフの方々に感謝いたします．どうもありがとうございました．そのほかにも同じ研究室の先輩・後輩や，同じ専攻の友人達にも色々とお世話になりました．どうもありがとうございました.

筆者が卒業論文でお世話になった早稲田大学理工学部機械工学科の方々に感謝いたします． 3 年生の時の故・加藤一郎教授のゼミと演習 (エンジニアリング・プラクティス) で，筆者はロボットの世界に入りたい気持ちで一杯になりました．卒業論文を途中からご指導くださった菅野先生には，卒業後も学会その他様々な機会にお世話になりました．どうもありがとうございます．尾形哲也さん，森田寿郎さん，渋谷恒司さんにも，学会等で助言や議論をいただくなど，いろいろとお世話になりありがとうございました.

研究の苦しい時に気分転換を図る機会を与えてくださった友人達に感謝いたします．どうもありがとう.

両親と妹達に感謝いたします．博士課程進学に快く賛成してくださり，生活面・精神面での支えになってくださいました．どうもありがとう.

最後に，筆者の研究に対する姿勢を理解して励ましてくれた妻・香織に感謝します．いろいろどうもありがとう.

修士課程 1 年から数えて 5 年と 9ヶ月の長い大学院生生活でした．途中，ネガティブになることもありましたが，振り返ってみると，本当に多くの方々に支えられて，この素晴らしい研究室で誇りに思える研究をし博士論文をまとめることができたことは，非常に幸福なことだと思います．お世話になった皆様に改めて感謝いたします．どうもありがとうございました.

# 参考文献

[1] V. Anderson. “Tensor Arm Manipulator”. U.S. Patent 3,497,083, , Filed 10 May 1968, Issued 24 February 1970.

[2] 浅田春比古, 出海晴生. “作業者の動作計測によるハイブリッド制御のための作業教示とプログラム生成”. 日本ロボット学会誌, Vol. 5, No. 6, pp. 452–458, 1987.

[3] R. Peter Bonasso and David Kortenkamp. “Characterizing an Architecture for Intelligent, Reactive Agents”. In *AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, 1995.

[4] W. J. Book, O. Maizza-Neto, and D. E. Whitney. “Feedback Control of Two Beam, Two Joint Systems with Distributed Flexibility”. *Journal of Dynamic Systems Measurement, and Control*, Vol. 97, No. 4, pp. 424–431, 1975.

[5] Rodney A. Brooks. “A Robust Layered Control System For A Mobile Robot”. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14–23, 1986.

[6] Rodney A. Brooks. “A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network”. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989.

[7] Computer Aided Design Software, Inc. “*Dynamic Analysis and Design System*”, 1993.

[8] J. H. Connell. “SSS: A Hybrid Architecture Applied to Robot Navigation”. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2719–2723, 1992.

[9] 原朗人, 水内郁夫, 稲葉雅幸, 井上博允. “拮抗腱駆動型多関節体幹を有する全身行動体”. 第 18 回日本ロボット学会学術講演会予稿集, 第 3 巻, pp. 1433–1434, 2000.

[10] 原朗人. “拮抗腱駆動型多関節体幹を有するロボットの全身行動に関する研究”. 修士論文, 東京大学, 2001.

[11] 橋本周司, 成田誠之助, 小林哲則, 高西淳夫. “プラットフォームとしての 2 足歩行型ヒューマノイド: WABIAN”. 第 15 回日本ロボット学会学術講演会予稿集, 第 3 巻, pp. 763–764, 1997.

[12] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. “The Development of Honda Humanoid Robot”. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 1321–1326, 1998.

[13] 広瀬茂男. “ロボット工学 (改訂版)”. 裳華房, 1987.

[14] 広瀬茂男, 佐藤幹夫. “多自由度ロボットの干渉駆動”. 日本ロボット学会誌, Vol. 7, No. 2, pp. 20–27, 1989.

[15] 広瀬茂男, 馬書根. “ワイヤ干渉駆動型多関節マニピュレータの開発”. 計測自動制御学会論文集, Vol. 26, No. 11, pp. 1291–1298, 1990.

[16] 久田俊明. “有限要素法によるモデリング”. ロボット学会誌, Vol. 16, No. 2, pp. 140–144, March 1998.

[17] 兵藤和人, 小林博明. “非線形バネ要素を持つ腱制御手首機構の研究”. 日本ロボット学会誌, Vol. 11, No. 8, pp. 1244–1251, 1993.

[18] 兵頭和人, 小林博明, 大鐘大介, 山本圭治郎. “冗長腱を持つ腱駆動ロボット機構の剛性調節”. 日本ロボット学会誌, Vol. 17, No. 4, pp. 493–502, 1999.

[19] M. Inaba. “Remote-Brained Robotics: Interfacing AI with Real World Behaviors”. In *Proceedings of 1993 International Symposium on Robotics Research*, 1993.

[20] Masayuki Inaba. “Remote-brained humanoid project”. *Advanced Robotics*, Vol. 11, No. 6, pp. 605–620, 1998.

[21] 稲葉雅幸, 長嶋功一, 水内郁夫, 但馬竜介, 吉海智晃, 國吉康夫, 井上博允. “脊椎を持つ全身腱駆動ヒューマノイド「腱太」の開発 — 脊椎を持つ全身腱駆動ヒューマノイド「腱太」 (その 1) —”. 第 19 回日本ロボット学会学術講演会講演論文集, pp. 775–776, 2001.

[22] Masayuki Inaba, Ikuo Mizuuchi, Ryosuke Tajima, Tomoaki Yoshikai, Koichi Nagashima, and Hirochika Inoue. “Building Spined Muscle-Tendon Humanoid”. In *Proceedings of 10th International Symposium of Robotics Research*, 2001.

[23] 岩本和世, 植田真嗣, 石原好之, 戸高敏之. “胴体可変機構を有する 4 脚歩行ロボット”. 日本ロボット学会第 5 会学術講演会予稿集, pp. 359–360, 1987.

[24] Satoshi Kagami, Koichi Nishiwaki, Tomomichi Sugihara, James J. Kuffner, Masayuki Inaba, and Hirochika Inoue. “Design and Implementation of Software

Research Platform for Humanoid Robotics: H6". In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, pp. 2431–2436, 2001.

[25] 金広文男, 稲葉雅幸, 井上博允. "人間型ロボットの全身行動と動作制御". 第 1 回ロボティクスシンポジア予稿集, pp. 11–16, 1996.

[26] 金広文男, 稲葉雅幸, 井上博允. "幾何モデルを中核とした人間型全身ロボットのソフトウェア環境". 第 15 回日本ロボット学会学術講演会予稿集, pp. 1013–1014, 1997.

[27] 金広文男, 水内郁夫, 垣内洋平, 稲葉雅幸, 井上博允. "体内LANによる神経系を持つリモートブレインロボットの開発". 第 15 回日本ロボット学会学術講演会予稿集, pp. 1023–1024, 1997.

[28] 金広文男, 稲葉雅幸, 井上博允. "適応行動記述のための構造操作可能な並列プロセスネットワークの実現". 日本機械学会ロボティクス・メカトロニクス講演会'97 講演論文集, 第 A 巻, pp. 547–548, 1997.

[29] 金広文男, 服部雄高, 稲葉雅幸, 井上博允. "概略記述の適応化による人間型ロボットの運動獲得". 第 16 回日本ロボット学会学術講演会予稿集, pp. 829–830, 1998.

[30] Fumio Kanehiro, Masayuki Inaba, and Hirochika Inoue. "Action Acquisition Framework for Humanoid Robots based on Kinematics and Dynamics Adaptation". In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1038–1043, 1999.

[31] 金広文男, 水内郁夫, 稲葉雅幸, 井上博允. "プラグインアーキテクチャに基づく発展的ソフトウェア環境の実装". 第 18 回日本ロボット学会学術講演会予稿集, pp. 1237–1238, 2000.

[32] 金広文男. "人間型全身行動ロボットシステムの発展的構成法". 博士論文, 東京大学, 2000.

[33] 金広文男, 稲葉雅幸, 井上博允. "ゲーム用高速動力学演算パッケージを用いたロボットボディの仮想化". 日本機械学会ロボティクス・メカトロニクス講演会'01 講演論文集, pp. 2P2–H3, 2001.

[34] 金広文男. "高速シミュレーション環境 FAST を用いたヒューマノイドシミュレーション". 第 19 回日本ロボット学会学術講演会講演論文集, pp. 931–932, 2001.

[35] I. A. Kapandji. “カパンディ関節の生理学 III 体幹・脊柱”. 医歯薬出版株式会社, 1986.

[36] Ichiro Kato, Sadamu Ohteru, Hiroshi Kobayashi, Katsuhiko Shirai, and Akihiko Uchiyama. “Information-Power Machine with Senses and Limbs(WABOT 1)”. In *First CISM - IFToMM Symposium, on Theory and Practice of Robots and Manipulators*, Vol. 1, pp. 11–24. Springer-Verlag, 1974.

[37] 川原, 河村, 中沢. “ゴム人工筋を用いた脊椎動物型背骨機構の制御”. ロボティクスメカトロニクス講演会’95 講演論文集 (Vol.A), pp. 17–20, 1995.

[38] 川島俊嗣. “空気圧人工筋を用いた柔軟な脊柱を持つロボットに関する研究”. 卒業論文, 東京大学, 1999.

[39] 川島俊嗣, 水内郁夫, 山口博明, 加賀美聡, 稲葉雅幸, 井上博允. “空気圧人工筋を用いた冗長性をもつ脊椎型ロボット”. 日本機械学会ロボティクス・メカトロニクス講演会’99 講演論文集, pp. 2A1–47–081. JSME, 1999.

[40] C. Ketelaar. “Ansys: Engineering Software with the Design and Analysis Answers.”. *Structure Anaysis System:Software, Hardware, Capability, Compatible, Application*, Vol. 1, pp. 31–39, 1986.

[41] 菊地文雄. “有限要素法概説”. サイエンス社, 1980.

[42] 小屋迫光太郎, 水内郁夫, 稲葉雅幸, 井上博允. “全身型ロボットの体内 LAN 用小型プロセッサモジュールの設計・開発”. 第 15 回日本ロボット学会学術講演会予稿集, pp. 1021–1022, 1997.

[43] 小屋迫光太郎, 金広文男, 水内郁夫, 稲葉雅幸, 井上博允. “脊椎構造を持つ人間型ロボットシステムの開発”. 日本機械学会ロボティクス・メカトロニクス講演会’98 予稿集, pp. 1CI1–6, 1998.

[44] O. Larson and C. Davidson. “Flexible Arm, Particularly a Robot Arm”. U.S. Patent 4,393,728, , Filed 23 February 1982, Issued 19 July 1983.

[45] O. Larson and C. Davidson. “Flexible Arm, Particularly a Robot Arm”. U.S. Patent 4,494,417, , Filed 29 September 1992, Issued 22 January 1985.

[46] K. F. L-Kovitz, J. E. Colgate, and S. D. R. Carnes. “Design of Components for Programmable Passive Impedance”. In *Proceedings of the 1991 IEEE International*

*Conference on Robotics and Automation*, pp. 1476–1481, Sacramento, California, April 1991.

[47] 李湧権, 下山勲. “人工手及びソフトアクチュエータに関する研究”. 第 16 回日本ロボット学会学術講演会予稿集, pp. 1359–1360, 1998.

[48] Toshihiro Matsui. “*EusLisp Reference Manual (Version 7.27)*”. Electrotechnical Laboratory, Tsukuba, Ibaraki 305, JAPAN, 1992.

[49] Toshihiro Matsui and Masayuki Inaba. “EusLisp: An Object-Based Implementation of Lisp”. *Journal of Information Processing*, Vol. 13, No. 3, pp. 327–338, 1990.

[50] Toshihiro Matsui and Masayuki Inaba. “EusLisp: an Object-Based Implementation of Lisp”. *Journal of Information Processing*, Vol. 13, No. 3, pp. 327–338, 1990.

[51] 松木健, 水内郁夫, 加賀美聡, 稲葉雅幸, 井上博允. “脊椎構造を持つ四脚ロボットとそのシミュレーション環境”. 第 16 回日本ロボット学会学術講演会予稿集, 第 1 巻, pp. 85–86, 1998.

[52] 松木健, 水内郁夫, 加賀美聡, 稲葉雅幸, 井上博允. “柔軟な脊椎機構を持つ四脚歩行ロボットの動作生成と行動制御”. 日本機械学会ロボティクス・メカトロニクス講演会’99 講演論文集, pp. 1P2–42–063, 1999.

[53] 水内郁夫, 小屋迫光太郎, 稲葉雅幸, 井上博允. “シリアルバスを用いたロボットボディプロセッサネットワーク－リモートブレインロボット 春プロジェクト 96：その５－”. 第 14 回日本ロボット学会学術講演会予稿集, pp. 357–358, 1996.

[54] 水内郁夫, 松木健, 加賀美聡, 稲葉雅幸, 井上博允. “可変な柔軟構造の体幹部を持つ人間型ロボットへの取り組み”. 第 16 回日本ロボット学会学術講演会予稿集, 第 2 巻, pp. 825–826, 1998.

[55] Ikuo Mizuuchi, Masayuki Inaba, and Hirochika Inoue. “Adaptive pick-and-place behaviors in a whole-body humanoid robot with an autonomous layer based on parallel sensor-motor modules”. *Robotics and Autonomous Systems*, Vol. 28, pp. 99–113, 1999.

[56] 水内郁夫, 稲葉雅幸, 井上博允. “可変柔軟構造を持つ人間型ロボットのシミュレーション環境”. 第 17 回日本ロボット学会学術講演会予稿集, 第 3 巻, pp. 1197–1198, 1999.

[57] 水内郁夫, 松木健, 稲葉雅幸, 井上博允. “柔軟構造を含む人間型ロボットのシミュレーション環境と実ロボットの動作生成への適用”. 日本機械学会ロボティクス・メカトロニクス講演会’99 講演論文集, pp. 2A1–47–085, 1999.

[58] 水内郁夫, 原朗人, 稲葉雅幸, 井上博允. “多自由度脊柱を持つ全身行動体のための体幹の腱駆動拮抗制御”. 第 18 回日本ロボット学会学術講演会予稿集, 第 3 巻, pp. 1459–1460, 2000.

[59] Ikuo Mizuuchi, Masayuki Inaba, and Hirochika Inoue. “A Flexible Spine Human-Form Robot — Development and Control of the Posture of the Spine —”. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2099–2104, 2001.

[60] 水内郁夫, 稲葉雅幸, 長嶋功一, 但馬竜介, 吉海智晃, 國吉康夫, 井上博允. “全身型ヒューマノイドの多自由度柔軟脊椎構造の設計と制御 — 脊椎を持つ全身腱駆動ヒューマノイド「腱太」 (その 2) —”. 第 19 回日本ロボット学会学術講演会講演論文集, pp. 777–778, 2001.

[61] Ikuo Mizuuchi, Masayuki Inaba, and Hirochika Inoue. “Adaptive Pick-and-Place Behaviors Based on Parallel Sensor-Motor Modules in a Whole Body Humanoid”. In *Proceedings of the International Conference on Intelligent Autonomous Systems 5*, pp. 504–510, 1998.

[62] 森田寿郎, 菅野重樹. “メカニカルインピーダンス調節機構による関節制御 — 機構の提案とロボット指への適用 —”. 日本ロボット学会誌, Vol. 14, No. 1, pp. 131–136, 1996.

[63] Toshio Morita and Shigeki Sugano. “Development and Evaluation of Seven-D.O.F. MIA Arm”. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 462–467, 1997.

[64] R. Mosher. “Articulating Mechanism”. U.S. Patent 3,580,099, , Filed 24 September 1969, Issued 25 May 1971.

[65] 武藤伸洋, 下倉健一朗. “触覚情報センシングを用いた倣い作業の教示と制御”. 日本ロボット学会誌, Vol. 15, No. 5, pp. 744–751, 1997.

[66] 長阪憲一郎, 稲葉雅幸, 井上博允. “GA と NN を用いた二足二腕型ロボットによる視覚誘導型ブランコ動作”. 日本機械学会ロボティクス・メカトロニクス講演会’95 講演論文集, pp. 1151–1153, 1995.

[67] 長阪憲一郎, 稲葉雅幸, 井上博允. “ヒューマノイドにおける人間規範型遺伝的動作獲得の研究”. 第 16 回日本ロボット学会学術講演会予稿集, pp. 827–828, 1998.

[68] 長阪憲一郎, 稲葉雅幸, 井上博允. “最適勾配法を用いた人間型ロボットの動歩行パターン生成”. 日本機械学会ロボティクス・メカトロニクス講演会’99 講演論文集, pp. 2P1–78–110, 1999.

[69] 長阪憲一郎, 稲葉雅幸, 井上博允. “動力学的動作変換フィルタ群を用いた人間型ロボットの全身行動設計”. 第 17 回日本ロボット学会学術講演会予稿集, pp. 1207–1208, 1999.

[70] Ken’ichirou Nagasaka, Masayuki Inaba, and Hirochika Inoue. “Dynamic Walking Pattern Generation for a Humanoid Robot Based on Optimal Gradient Method”. In *1999 IEEE International Conference on Systems, Man, and Cybernetics*, pp. FA22–3, 1999.

[71] 中村仁彦, 山崎友敬, 鈴木雄一. “反射行動の組合せによるロボットハンドの制御”. 日本機械学会ロボティクス・メカトロニクス講演会’95 講演論文集, 第 B 巻, pp. 1166–1169, 1995.

[72] 中村仁彦, 山崎友敬. “行動の重ね合わせ理論の研究 — 多指ハンドの把持動作への応用—”. 第 14 回日本ロボット学会学術講演会予稿集, 第 3 巻, pp. 1047–1048, 1996.

[73] 中村仁彦, 山崎友敬. “反射行動の重ね合わせ理論とその多指ハンドの反射的把握動作への応用”. 日本ロボット学会誌, Vol. 15, No. 3, pp. 448–459, 1997.

[74] Koichi Nishiwaki, Tomomichi Sugihara, Satoshi Kagami, Fumio Kanehiro, Masayuki Inaba, and Hirochika Inoue. “Design and Development of Research Platform for Perception-Action Integration in Humanoid Robot : H6”. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’00)*, Vol. 3, pp. 1559–1564, 2000.

[75] 則次俊朗, 和田力. “ゴム人工筋のロボット制御への応用”. 日本ロボット学会誌, Vol. 9, No. 4, pp. 502–506, 1991 1991.

[76] 則次俊朗. “空気圧アクチュエータ”. 日本ロボット学会誌, Vol. 15, No. 3, pp. 43–47, 4 1997.

[77] 尾田十八, 王安麟, 松本徳之. “剛性可変ばねの試作とその変位制御問題への応用”. 日本機械学会論文集 (C 編), Vol. 59, No. 564, pp. 262–267, 1993.

[78] 日本ロボット学会. “特集「フレキシブルアーム」”. 日本ロボット学会誌, Vol. 6, No. 5, pp. 51–103, 1988.

[79] 日本ロボット学会. “特集「フレキシブル・マニピュレータ」”. 日本ロボット学会誌, Vol. 12, No. 2, pp. 1–62, 1994.

[80] 日本ロボット学会. “特集「ソフトロボティクス」”. 日本ロボット学会誌, Vol. 17, No. 6, pp. 1–52, 1999.

[81] 大鐘大介, 兵藤和人, 小林博明. “非線形バネ要素を持つ 7 自由度腱制御アームの機構と制御”. 日本ロボット学会誌, Vol. 14, No. 8, pp. 1152–1159, 1996.

[82] 岡哲資, 稲葉雅幸, 井上博允. “BeNet: 自律ロボットの情報システム記述のための並列情報処理モデル”. 日本ロボット学会誌, Vol. 15, No. 6, pp. 878–885, 1997.

[83] 岡哲資, 稲葉雅幸, 井上博允. “並列モジュール記述モデルに基づく自律ロボットの情報システム開発環境”. 日本ロボット学会誌, Vol. 15, No. 7, pp. 1050–1059, 1997.

[84] 大方一三, 大塚和弘. “形状記憶材料 — アクチュエータとしての形状記憶合金 —”. 日本ロボット学会誌, Vol. 13, No. 2, pp. 189–1192, 1995.

[85] 小澤隆太, 小林博明. “腱に非線形弾性を持つ腱駆動システムの制御”. 日本ロボット学会誌, Vol. 17, No. 2, pp. 275–281, 1999.

[86] Philips. “*The $I^2C$-bus Specification*”, 1989.

[87] Johannes W. Rohen, 横地千仭. “解剖学カラーアトラス”. 医学書院, 1985.

[88] 酒井勝. “ロボットの直接教示”. 日本ロボット学会誌, Vol. 13, No. 5, pp. 627–628, 1995.

[89] 佐々木忠之, 川島稔夫, 青山英樹, 伊福部達, 小川孝寿. “水素吸蔵合金を利用したアクチュエータの開発”. 日本ロボット学会誌, Vol. 4, No. 2, pp. 119–122, 1985.

[90] SONY. http://www.sony.co.jp/SonyInfo/News/Press/200011/00-057a/.

[91] I. Sutherland. “Robot Arm Structure”. U.S. Patent 4,900,218, , Filed 18 June 1985, Issued 27 February 1990.

[92] 田所諭. “柔かいアクチュエータ”. 日本ロボット学会誌, Vol. 15, No. 3, pp. 318–322, 1997.

[93] 但馬竜介, 水内郁夫, 吉海智晃, 長嶋功一, 國吉康夫, 稲葉雅幸, 井上博允. “球面ジョイントを用いたヒューマノイドの四肢構造 — 脊椎を持つ全身腱駆動ヒューマノイド「腱太」(その 3) —”. 第 19 回日本ロボット学会学術講演会講演論文集, pp. 779–780, 2001.

[94] 但馬竜介. “全身型多自由度拮抗腱駆動ロボットシステムの研究”. 博士論文, 東京大学, 2002.

[95] 高森年. “アクチュエータ材料”. 日本ロボット学会誌, Vol. 13, No. 2, pp. 193–196, 1995.

[96] 田宮幸春, 稲葉雅幸, 井上博允. “操縦型二足二腕ロボットにおける片足立脚時の全身によるバランス機能の実現”. 第 15 回日本ロボット学会学術講演会予稿集, pp. 777–778, 1997.

[97] S. Ubhayaker. “Robotic Arm Systems”. U.S. Patent 4,964,062, , Filed 16 February 1988, Issued 16 October 16 1990.

[98] S. Ubhayaker. “Robotic Arm Systems”. U.S. Patent 4,964,062, , Filed 2 October 1989, Issued 4 September 1990.

[99] 内山勝, 近野敦. “フレキシブルロボット”. 日本ロボット学会誌, Vol. 16, No. 7, pp. 921–923, 1998.

[100] Robert B. van Varseveld and Gary M. Bone. “Accurate Position Control of a Pneumatic Actuator Using On/Off Solenoid Valves”. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 1196–1201, 1997.

[101] 脇坂裕一. “水素吸蔵合金アクチュエータ”. 日本ロボット学会誌, Vol. 15, No. 3, pp. 347–350, 1997.

[102] F. E. Wells. “Remote Control Manipupator for Zero-Gravity Enviroment”. U.S. Patent 3,631,737, , Filed 18 September 1970, Issued 4 January 1972.

[103] 八重樫剛史. “ポータブルロボットによるインターネットを用いた遠隔対話システムに関する研究”. 修士論文, 東京大学, 1998.

[104] Jin’ichi Yamaguchi, Sadatoshi Inoue, Daisuke Nishino, and Atsuo Takanishi. “Development of a Bipedal Humanoid Robot Having Antagonistic Driven Joints and Three DOF Trunk”. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 96–101, 1998.

[105] 吉田成徳. “脊椎構造を持つ人間型ロボットの制御と全身動作に関する研究”. 卒業論文, 東京大学, 2002.

[106] 吉海智晃, 稲葉雅幸, 水内郁夫, 但馬竜介, 長嶋功一, 和井田寛則, 國吉康夫, 井上博允. “脊椎を持つヒューマノイドにおける頭部構造と注視制御 — 脊椎を持つ全身腱駆動ヒューマノイド「腱太」 (その 4) —”. 第 19 回日本ロボット学会学術講演会講演論文集, pp. 781–782, 2001.

# 付録 A

# 各種制御モードにおける張力・消費電流の平均値・最大値

## A.1 本章の説明

### A.1.1 制御モード

**L 制御** 筋長制御．詳しくは 4.6.1節 を参照.

**T 制御** 筋張力制御．詳しくは 4.6.2節を参照.

**S 制御** ソフトウェアばね制御．詳しくは 4.6.3節を参照.

**LL 制御** 張力制限付き筋長制御．詳しくは 4.6.4節を参照.

### A.1.2 姿勢制御実験における制御モードの種類

姿勢制御実験を行った制御モードの種類は，122ページにある通り，

1. 全筋を L 制御
2. 全筋を LL 制御
3. 全筋を S 制御 (バネ定数中)
4. 全筋を S 制御 (バネ定数大)
5. 縮み筋は L 制御，伸び筋は LL 制御
6. 縮み筋は L 制御，伸び筋は T 制御
7. 縮み筋は L 制御，伸び筋は S 制御 (バネ定数中)
8. 縮み筋は L 制御，伸び筋は S 制御 (バネ定数大)

である.

### A.1.3 表の見方

表の最上段の CLA-RF 〜 HIP-LB は，筋の種類を示す．“CLA-” は肩のブロックのモータによって引かれる筋を意味し，“HIP-” は腰のブロックのモータによって引かれる筋を意味する．“-RF”，“-LF”，“-RB”，“-LB” は，それぞれブロックの右前，左前，右後，左後の位置に引っ張り点がある筋を示す ( **Fig. 3.31** (58ページ) 参照).

左端の列の数字は，pitch,roll 各軸回りに脊椎全体で何度曲がる姿勢をモデル上でとったかを示す．例えば pitch−30 なら，脊椎を構成する 5 つの節がそれぞれ pitch 軸回りに $-6^\circ$ ずつ曲がる姿勢をモデル上でとったことを意味する.

## A.2 各種制御モードにおける張力・消費電流の平均値・最大値

Table A.1 〜 Table A.7 は平均張力, Table A.8 〜 Table A.14 は最大張力, Table A.15 〜 Table A.21 は平均消費電流, Table A.22 〜 Table A.28 は最大消費電流を示す.

なお欄がない姿勢は, 重力に負けて脊椎が崩れてしまい手で支えたためデータが取れなかった姿勢である.

また, 縮み筋は L 制御・伸び筋は T 制御による制御は, 制御できた姿勢が無かったためデータを取ることができなかった.

Table A.1　筋張力の平均 (全筋 L 制御)
The average of muscle-tension (all L-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.07 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 | 0.03 |
| -10 | 0.24 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.13 | 0.08 |
| -20 | 0.41 | 0.22 | 0.00 | 0.00 | 0.00 | 0.00 | 0.52 | 0.43 |
| -30 | 0.58 | 0.26 | 0.00 | 0.00 | 0.00 | 0.00 | 0.88 | 0.61 |
| -20 | 0.41 | 0.11 | 0.00 | 0.00 | 0.00 | 0.00 | 1.50 | 0.84 |
| -10 | 0.22 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.80 | 0.68 |
| 0 | 0.07 | 0.01 | 0.01 | 0.07 | 0.00 | 0.00 | 0.25 | 0.45 |
| 10 | 0.04 | 0.00 | 0.09 | 0.25 | 0.00 | 0.00 | 0.13 | 0.31 |
| 20 | 0.02 | 0.00 | 0.29 | 0.51 | 0.49 | 0.39 | 0.00 | 0.02 |
| 30 | 0.02 | 0.00 | 0.58 | 0.61 | 1.12 | 0.69 | 0.00 | 0.02 |
| 20 | 0.00 | 0.00 | 0.51 | 0.47 | 1.34 | 0.93 | 0.00 | 0.00 |
| 10 | 0.03 | 0.00 | 0.21 | 0.19 | 0.77 | 0.52 | 0.00 | 0.00 |
| 0 | 0.04 | 0.01 | 0.04 | 0.04 | 0.29 | 0.24 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.03 | 0.00 | 0.09 | 0.01 | 0.00 | 0.01 | 0.00 | 0.03 |
| 20 | 0.00 | 0.01 | 0.06 | 0.08 | 0.00 | 0.01 | 0.09 | 0.09 |
| 30 | 0.00 | 0.25 | 0.07 | 0.18 | 0.00 | 0.01 | 0.04 | 0.22 |
| 20 | 0.00 | 0.37 | 0.00 | 0.09 | 0.00 | 0.00 | 0.05 | 0.35 |
| 10 | 0.01 | 0.16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.15 | 0.35 |
| 0 | 0.04 | 0.15 | 0.00 | 0.03 | 0.00 | 0.00 | 0.19 | 0.33 |
| -10 | 0.42 | 0.25 | 0.00 | 0.08 | 0.00 | 0.00 | 0.18 | 0.22 |
| -20 | 0.72 | 0.27 | 0.00 | 0.16 | 0.00 | 0.00 | 0.35 | 0.30 |
| -30 | 1.19 | 0.24 | 0.00 | 0.25 | 0.00 | 0.00 | 0.45 | 0.18 |
| -20 | 0.70 | 0.01 | 0.03 | 0.05 | 0.00 | 0.00 | 0.46 | 0.05 |
| -10 | 0.39 | 0.00 | 0.06 | 0.00 | 0.00 | 0.00 | 0.44 | 0.21 |

Table A.2　筋張力の平均 (全筋 LL 制御)
The average of muscle-tension (all LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.60 | 0.29 | 0.42 | 0.64 | 0.46 | 0.68 | 0.62 | 0.43 |
| -10 | 0.61 | 0.36 | 0.76 | 0.26 | 0.59 | 0.76 | 0.77 | 0.59 |
| -20 | 0.54 | 0.25 | 0.00 | 0.03 | 0.73 | 0.79 | 0.98 | 0.64 |
| -30 | 0.60 | 0.29 | 0.00 | 0.00 | 0.40 | 0.45 | 0.98 | 0.76 |
| -20 | 0.48 | 0.42 | 0.03 | 0.00 | 0.06 | 0.00 | 0.98 | 0.82 |
| -10 | 0.53 | 0.38 | 0.24 | 0.28 | 0.41 | 0.43 | 0.97 | 0.77 |
| 0 | 0.61 | 0.44 | 0.49 | 0.68 | 0.46 | 0.53 | 0.61 | 0.59 |
| roll | | | | | | | | |
| 10 | 0.16 | 0.40 | 0.47 | 0.63 | 0.78 | 0.64 | 0.22 | 0.44 |
| 20 | 0.05 | 0.43 | 0.22 | 0.66 | 0.73 | 0.74 | 0.51 | 0.53 |
| 30 | 0.03 | 0.57 | 0.09 | 0.69 | 0.53 | 0.80 | 0.64 | 0.64 |
| 20 | 0.07 | 0.57 | 0.09 | 0.66 | 0.28 | 0.53 | 0.57 | 0.68 |
| 10 | 0.14 | 0.58 | 0.23 | 0.61 | 0.30 | 0.38 | 0.42 | 0.68 |
| 0 | 0.44 | 0.61 | 0.27 | 0.69 | 0.43 | 0.52 | 0.57 | 0.65 |
| -10 | 0.77 | 0.71 | 0.33 | 0.69 | 0.31 | 0.47 | 0.58 | 0.48 |
| -20 | 1.09 | 0.60 | 0.26 | 0.69 | 0.20 | 0.40 | 0.55 | 0.25 |
| -30 | 1.24 | 0.60 | 0.71 | 0.25 | 0.00 | 0.03 | 0.52 | 0.07 |
| -20 | 1.15 | 0.74 | 0.29 | 0.37 | 0.20 | 0.09 | 0.58 | 0.24 |
| -10 | 0.94 | 0.48 | 0.24 | 0.39 | 0.22 | 0.19 | 0.59 | 0.40 |

Table A.3　筋張力の平均 (全筋 S 制御 (バネ定数中))
The average of muscle-tension (all S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.24 | 0.06 | 0.01 | 0.00 | 0.00 | 0.00 | 0.12 | 0.08 |
| -10 | 0.23 | 0.02 | 0.06 | 0.00 | 0.09 | 0.26 | 0.01 | 0.00 |
| -20 | 0.67 | 0.22 | 0.05 | 0.00 | 0.00 | 0.04 | 0.08 | 0.02 |
| -30 | 0.62 | 0.47 | 0.09 | 0.00 | 0.00 | 0.00 | 0.19 | 0.32 |
| -20 | 0.36 | 0.16 | 0.01 | 0.00 | 0.00 | 0.00 | 0.55 | 0.65 |
| -10 | 0.28 | 0.07 | 0.00 | 0.00 | 0.01 | 0.00 | 0.37 | 0.60 |
| 0 | 0.13 | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 | 0.14 | 0.23 |
| roll | | | | | | | | |
| 10 | 0.03 | 0.01 | 0.01 | 0.00 | 0.12 | 0.25 | 0.01 | 0.01 |
| 20 | 0.01 | 0.00 | 0.04 | 0.00 | 0.06 | 0.36 | 0.01 | 0.00 |
| 30 | 0.01 | 0.09 | 0.00 | 0.00 | 0.00 | 0.18 | 0.00 | 0.02 |
| 20 | 0.01 | 0.35 | 0.00 | 0.14 | 0.00 | 0.20 | 0.00 | 0.10 |
| 10 | 0.01 | 0.19 | 0.00 | 0.00 | 0.03 | 0.01 | 0.00 | 0.30 |
| 0 | 0.09 | 0.15 | 0.00 | 0.00 | 0.01 | 0.00 | 0.08 | 0.20 |
| -10 | 0.34 | 0.22 | 0.06 | 0.00 | 0.00 | 0.00 | 0.08 | 0.06 |
| -20 | 0.63 | 0.32 | 0.78 | 0.12 | 0.02 | 0.00 | 0.04 | 0.03 |
| -30 | 0.84 | 0.36 | 0.91 | 0.22 | 0.11 | 0.00 | 0.05 | 0.02 |
| -20 | 0.71 | 0.01 | 0.49 | 0.00 | 0.45 | 0.00 | 0.05 | 0.00 |
| -10 | 0.45 | 0.00 | 0.33 | 0.00 | 0.46 | 0.04 | 0.06 | 0.00 |

Table A.4 筋張力の平均 (全筋 S 制御 (バネ定数大))
The average of muscle-tension (all S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.15 | 0.02 | 0.05 | 0.12 | 0.37 | 0.04 | 0.06 | 0.03 |
| -10 | 0.23 | 0.05 | 0.09 | 0.07 | 0.53 | 0.27 | 0.04 | 0.00 |
| -20 | 0.42 | 0.26 | 0.13 | 0.03 | 0.23 | 0.18 | 0.10 | 0.03 |
| -30 | 0.49 | 0.54 | 0.44 | 0.06 | 0.13 | 0.13 | 0.18 | 0.22 |
| -20 | 0.29 | 0.08 | 0.06 | 0.00 | 0.01 | 0.00 | 0.41 | 0.64 |
| -10 | 0.18 | 0.01 | 0.02 | 0.02 | 0.00 | 0.00 | 0.23 | 0.43 |
| 0 | 0.14 | 0.00 | 0.04 | 0.12 | 0.06 | 0.00 | 0.17 | 0.08 |
| 10 | 0.06 | 0.00 | 0.51 | 0.19 | 0.45 | 0.04 | 0.04 | 0.02 |
| 20 | 0.03 | 0.00 | 0.52 | 0.27 | 0.72 | 0.18 | 0.06 | 0.02 |
| 30 | 0.01 | 0.00 | 0.45 | 0.32 | 1.89 | 0.74 | 0.02 | 0.01 |
| 20 | 0.01 | 0.00 | 0.42 | 0.15 | 2.24 | 0.92 | 0.01 | 0.00 |
| 10 | 0.01 | 0.00 | 0.30 | 0.09 | 2.15 | 0.86 | 0.01 | 0.00 |
| 0 | 0.06 | 0.00 | 0.10 | 0.01 | 1.25 | 0.56 | 0.01 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.01 | 0.00 | 0.46 | 0.01 | 0.79 | 0.20 | 0.03 | 0.00 |
| 20 | 0.00 | 0.00 | 0.45 | 0.18 | 0.78 | 0.33 | 0.08 | 0.00 |
| 30 | 0.00 | 0.01 | 0.37 | 0.50 | 0.45 | 0.32 | 0.03 | 0.00 |
| 20 | 0.01 | 0.01 | 0.00 | 0.28 | 0.12 | 0.27 | 0.01 | 0.03 |
| 10 | 0.01 | 0.01 | 0.00 | 0.13 | 0.31 | 0.20 | 0.01 | 0.03 |
| 0 | 0.08 | 0.08 | 0.01 | 0.12 | 0.42 | 0.04 | 0.02 | 0.02 |
| -10 | 0.55 | 0.19 | 0.36 | 0.19 | 0.40 | 0.01 | 0.01 | 0.00 |
| -20 | 0.84 | 0.34 | 1.12 | 0.40 | 0.55 | 0.05 | 0.01 | 0.00 |
| -30 | 0.93 | 0.42 | 2.52 | 0.60 | 0.78 | 0.16 | 0.00 | 0.00 |
| -20 | 0.66 | 0.03 | 0.45 | 0.02 | 1.34 | 0.18 | 0.03 | 0.00 |
| -10 | 0.52 | 0.01 | 0.39 | 0.00 | 0.89 | 0.14 | 0.07 | 0.00 |

Table A.5　筋張力の平均 (L 制御 +LL 制御)
The average of muscle-tension (L-mode and LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| -10 | 0.05 | 0.18 | 0.07 | 0.02 | 0.10 | 0.21 | 0.00 | 0.00 |
| -20 | 0.13 | 0.23 | 0.00 | 0.00 | 0.00 | 0.06 | 0.06 | 0.00 |
| -30 | 0.21 | 0.31 | 0.00 | 0.00 | 0.00 | 0.08 | 0.18 | 0.12 |
| -20 | 0.08 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.43 | 0.64 |
| -10 | 0.01 | 0.07 | 0.04 | 0.07 | 0.00 | 0.00 | 0.16 | 0.38 |
| 0 | 0.00 | 0.00 | 0.18 | 0.50 | 0.14 | 0.16 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.26 | 0.61 | 1.31 | 1.20 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.08 | 0.67 | 0.47 | 0.91 | 0.00 | 0.00 |
| 30 | 0.00 | 0.00 | 0.01 | 0.68 | 0.27 | 0.94 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.01 | 0.78 | 0.24 | 0.92 | 0.00 | 0.00 |
| 10 | 0.00 | 0.00 | 0.03 | 0.65 | 0.56 | 0.91 | 0.00 | 0.00 |
| 0 | 0.00 | 0.01 | 0.14 | 0.39 | 0.56 | 0.57 | 0.00 | 0.00 |
| -10 | 0.00 | 0.01 | 0.31 | 0.25 | 0.65 | 0.45 | 0.00 | 0.00 |
| -20 | 0.08 | 0.06 | 0.13 | 0.15 | 0.74 | 0.28 | 0.00 | 0.00 |
| -30 | 0.18 | 0.11 | 0.48 | 0.06 | 0.78 | 0.10 | 0.00 | 0.00 |
| -20 | 0.09 | 0.07 | 0.56 | 0.02 | 0.80 | 0.07 | 0.00 | 0.00 |
| -10 | 0.00 | 0.01 | 0.17 | 0.01 | 0.78 | 0.16 | 0.00 | 0.00 |

Table A.6 筋張力の平均 (L 制御 +S 制御 (バネ定数中))
The average of muscle-tension (L-mode and S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.00 | 0.00 | 0.33 | 0.15 | 0.51 | 0.06 | 0.02 | 0.00 |
| -10 | 0.01 | 0.01 | 0.28 | 0.06 | 0.62 | 0.27 | 0.03 | 0.00 |
| -20 | 0.08 | 0.05 | 0.27 | 0.04 | 0.31 | 0.17 | 0.06 | 0.00 |
| -30 | 0.24 | 0.10 | 0.22 | 0.04 | 0.04 | 0.05 | 0.12 | 0.02 |
| -20 | 0.07 | 0.01 | 0.04 | 0.01 | 0.00 | 0.00 | 0.51 | 0.10 |
| -10 | 0.04 | 0.04 | 0.03 | 0.00 | 0.01 | 0.00 | 0.90 | 0.62 |
| 0 | 0.00 | 0.00 | 0.03 | 0.05 | 0.01 | 0.00 | 0.67 | 0.23 |
| 10 | 0.03 | 0.00 | 0.30 | 0.18 | 0.27 | 0.00 | 0.15 | 0.01 |
| 20 | 0.06 | 0.00 | 1.86 | 0.49 | 0.71 | 0.03 | 0.03 | 0.00 |
| 30 | 0.11 | 0.00 | 2.32 | 0.64 | 1.86 | 0.99 | 0.01 | 0.00 |
| 20 | 0.02 | 0.00 | 0.54 | 0.21 | 2.01 | 1.32 | 0.00 | 0.00 |
| 10 | 0.00 | 0.00 | 0.45 | 0.17 | 1.60 | 0.93 | 0.00 | 0.00 |
| 0 | 0.00 | 0.00 | 0.64 | 0.20 | 0.95 | 0.54 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.69 | 0.20 | 0.51 | 0.36 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.48 | 0.27 | 0.50 | 0.62 | 0.00 | 0.00 |
| 30 | 0.00 | 0.00 | 0.06 | 0.46 | 0.51 | 1.47 | 0.00 | 0.00 |
| 20 | 0.01 | 0.00 | 0.09 | 0.79 | 1.47 | 1.59 | 0.01 | 0.00 |
| 10 | 0.00 | 0.00 | 0.29 | 0.39 | 0.88 | 1.28 | 0.01 | 0.00 |
| 0 | 0.03 | 0.00 | 0.50 | 0.28 | 0.65 | 0.61 | 0.01 | 0.00 |
| -10 | 0.08 | 0.03 | 0.91 | 0.29 | 0.87 | 0.26 | 0.00 | 0.00 |
| -20 | 0.32 | 0.10 | 2.15 | 0.31 | 1.05 | 0.11 | 0.00 | 0.00 |
| -30 | 1.63 | 0.26 | 4.67 | 0.34 | 0.93 | 0.16 | 0.00 | 0.00 |
| -20 | 0.18 | 0.04 | 2.86 | 0.17 | 0.66 | 0.01 | 0.00 | 0.00 |
| -10 | 0.06 | 0.00 | 2.13 | 0.14 | 0.53 | 0.07 | 0.00 | 0.00 |

Table A.7 筋張力の平均 (L 制御 +S 制御 (バネ定数大))
The average of muscle-tension (L-mode and S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.08 | 0.00 | 0.26 | 0.06 | 0.44 | 0.13 | 0.00 | 0.01 |
| -10 | 0.11 | 0.00 | 0.15 | 0.03 | 0.60 | 0.53 | 0.00 | 0.00 |
| -20 | 0.27 | 0.00 | 0.05 | 0.04 | 0.33 | 0.42 | 0.00 | 0.00 |
| -30 | 0.50 | 0.06 | 0.08 | 0.08 | 0.01 | 0.14 | 0.08 | 0.01 |
| -20 | 0.25 | 0.01 | 0.04 | 0.00 | 0.02 | 0.00 | 0.43 | 0.52 |
| -10 | 0.25 | 0.01 | 0.04 | 0.00 | 0.02 | 0.00 | 0.65 | 0.90 |
| 0 | 0.09 | 0.00 | 0.06 | 0.01 | 0.01 | 0.00 | 0.33 | 0.23 |
| 10 | 0.06 | 0.00 | 0.27 | 0.12 | 0.35 | 0.00 | 0.11 | 0.08 |
| 20 | 0.09 | 0.00 | 1.80 | 0.24 | 0.73 | 0.20 | 0.01 | 0.05 |
| 30 | 0.13 | 0.00 | 2.65 | 0.31 | 1.28 | 0.63 | 0.06 | 0.19 |
| 20 | 0.06 | 0.00 | 0.85 | 0.18 | 1.37 | 0.99 | 0.00 | 0.01 |
| 10 | 0.02 | 0.00 | 0.35 | 0.11 | 1.30 | 0.81 | 0.00 | 0.00 |
| 0 | 0.01 | 0.00 | 0.21 | 0.07 | 1.03 | 0.67 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.45 | 0.11 | 0.51 | 0.51 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.26 | 0.16 | 0.44 | 0.70 | 0.00 | 0.00 |
| 30 | 0.02 | 0.00 | 0.09 | 0.23 | 0.25 | 1.32 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.05 | 0.68 | 0.91 | 1.48 | 0.01 | 0.00 |
| 10 | 0.01 | 0.00 | 0.15 | 0.32 | 0.90 | 1.27 | 0.01 | 0.00 |
| 0 | 0.01 | 0.00 | 0.21 | 0.19 | 0.82 | 0.64 | 0.00 | 0.00 |
| -10 | 0.14 | 0.00 | 0.50 | 0.18 | 0.89 | 0.39 | 0.00 | 0.00 |
| -20 | 0.58 | 0.05 | 0.83 | 0.33 | 0.91 | 0.23 | 0.01 | 0.00 |
| -30 | 1.50 | 0.27 | 2.87 | 0.62 | 0.70 | 0.28 | 0.01 | 0.00 |
| -20 | 0.33 | 0.00 | 1.68 | 0.15 | 0.76 | 0.10 | 0.00 | 0.00 |
| -10 | 0.16 | 0.00 | 1.11 | 0.05 | 0.61 | 0.11 | 0.00 | 0.00 |

Table A.8 筋張力の最大値 (全筋 L 制御)
The maximum of muscle-tension (all L-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.07 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.06 | 0.04 |
| -10 | 0.26 | 0.11 | 0.00 | 0.00 | 0.00 | 0.00 | 0.13 | 0.09 |
| -20 | 0.43 | 0.24 | 0.00 | 0.00 | 0.01 | 0.00 | 0.55 | 0.44 |
| -30 | 0.61 | 0.27 | 0.00 | 0.00 | 0.01 | 0.00 | 0.93 | 0.62 |
| -20 | 0.48 | 0.12 | 0.00 | 0.00 | 0.01 | 0.00 | 1.90 | 0.850 |
| -10 | 0.23 | 0.05 | 0.00 | 0.00 | 0.01 | 0.00 | 0.82 | 0.69 |
| 0 | 0.08 | 0.01 | 0.01 | 0.08 | 0.00 | 0.00 | 0.28 | 0.46 |
| 10 | 0.05 | 0.00 | 0.10 | 0.26 | 0.01 | 0.00 | 0.15 | 0.33 |
| 20 | 0.03 | 0.00 | 0.32 | 0.53 | 0.55 | 0.42 | 0.00 | 0.04 |
| 30 | 0.02 | 0.00 | 0.60 | 0.61 | 1.21 | 0.71 | 0.01 | 0.02 |
| 20 | 0.00 | 0.00 | 0.54 | 0.48 | 1.35 | 0.95 | 0.00 | 0.00 |
| 10 | 0.03 | 0.00 | 0.27 | 0.22 | 0.78 | 0.54 | 0.00 | 0.00 |
| 0 | 0.05 | 0.01 | 0.04 | 0.04 | 0.31 | 0.26 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.03 | 0.00 | 0.09 | 0.02 | 0.02 | 0.01 | 0.00 | 0.03 |
| 20 | 0.02 | 0.06 | 0.07 | 0.09 | 0.01 | 0.01 | 0.90 | 0.09 |
| 30 | 0.00 | 0.26 | 0.07 | 0.18 | 0.00 | 0.01 | 0.04 | 0.23 |
| 20 | 0.00 | 0.38 | 0.00 | 0.09 | 0.00 | 0.00 | 0.07 | 0.38 |
| 10 | 0.01 | 0.20 | 0.00 | 0.00 | 0.01 | 0.00 | 0.16 | 0.36 |
| 0 | 0.05 | 0.17 | 0.00 | 0.03 | 0.05 | 0.00 | 0.19 | 0.35 |
| -10 | 0.45 | 0.26 | 0.00 | 0.08 | 0.00 | 0.00 | 0.19 | 0.23 |
| -20 | 0.74 | 0.28 | 0.00 | 0.17 | 0.00 | 0.00 | 0.36 | 0.32 |
| -30 | 1.27 | 0.27 | 0.00 | 0.29 | 0.01 | 0.00 | 0.46 | 0.23 |
| -20 | 0.73 | 0.05 | 0.05 | 0.05 | 0.00 | 0.00 | 0.47 | 0.09 |
| -10 | 0.40 | 0.00 | 0.07 | 0.00 | 0.01 | 0.00 | 0.46 | 0.22 |

Table A.9　筋張力の最大値 (全筋 LL 制御)
The maximum of muscle-tension (all LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.61 | 0.30 | 0.43 | 0.65 | 0.47 | 0.71 | 0.64 | 0.43 |
| -10 | 0.69 | 0.60 | 0.10 | 0.33 | 0.76 | 0.88 | 0.98 | 0.63 |
| -20 | 0.61 | 0.27 | 0.01 | 0.05 | 0.78 | 0.90 | 0.99 | 0.65 |
| -30 | 0.62 | 0.31 | 0.00 | 0.00 | 0.42 | 0.53 | 1.01 | 0.81 |
| -20 | 0.50 | 0.44 | 0.04 | 0.00 | 0.08 | 0.01 | 1.00 | 0.84 |
| -10 | 0.57 | 0.41 | 0.25 | 0.32 | 0.43 | 0.48 | 1.00 | 0.80 |
| 0 | 0.63 | 0.60 | 0.66 | 0.69 | 0.60 | 0.57 | 0.69 | 0.61 |
| roll | | | | | | | | |
| 10 | 0.23 | 0.93 | 0.57 | 0.64 | 0.83 | 0.67 | 0.26 | 0.47 |
| 20 | 0.05 | 0.44 | 0.24 | 0.66 | 0.74 | 0.76 | 0.58 | 0.56 |
| 30 | 0.03 | 0.64 | 0.09 | 0.69 | 0.60 | 0.83 | 0.67 | 0.68 |
| 20 | 0.12 | 0.60 | 0.09 | 0.68 | 0.30 | 0.54 | 0.60 | 0.69 |
| 10 | 0.17 | 0.17 | 0.26 | 0.66 | 0.39 | 0.46 | 0.46 | 0.69 |
| 0 | 0.51 | 0.69 | 0.29 | 0.69 | 0.45 | 0.53 | 0.59 | 0.66 |
| -10 | 0.79 | 0.78 | 0.41 | 0.71 | 0.36 | 0.50 | 0.60 | 0.49 |
| -20 | 1.15 | 0.76 | 0.42 | 0.72 | 0.29 | 0.51 | 0.61 | 0.45 |
| -30 | 1.34 | 0.69 | 0.79 | 0.32 | 0.06 | 0.05 | 0.61 | 0.10 |
| -20 | 1.16 | 0.85 | 0.33 | 0.40 | 0.22 | 0.13 | 0.62 | 0.34 |
| -10 | 1.02 | 0.83 | 0.25 | 0.43 | 0.24 | 0.23 | 0.63 | 0.47 |

Table A.10　筋張力の最大値 (全筋 S 制御 (バネ定数中))
The maximum of muscle-tension (all S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.27 | 0.08 | 0.04 | 0.68 | 0.01 | 0.00 | 0.15 | 0.09 |
| -10 | 0.27 | 0.05 | 0.10 | 0.00 | 0.24 | 0.29 | 0.02 | 0.00 |
| -20 | 0.68 | 0.23 | 0.07 | 0.00 | 0.01 | 0.08 | 0.08 | 0.02 |
| -30 | 0.64 | 0.52 | 0.10 | 0.00 | 0.01 | 0.01 | 0.21 | 0.41 |
| -20 | 0.37 | 0.19 | 0.01 | 0.00 | 0.01 | 0.00 | 0.60 | 0.66 |
| -10 | 0.30 | 0.09 | 0.01 | 0.00 | 0.04 | 0.00 | 0.43 | 0.61 |
| 0 | 0.14 | 0.01 | 0.01 | 0.00 | 0.01 | 0.00 | 0.16 | 0.35 |
| roll | | | | | | | | |
| 10 | 0.13 | 0.09 | 0.10 | 0.01 | 0.72 | 0.75 | 0.12 | 0.05 |
| 20 | 0.01 | 0.00 | 0.05 | 0.00 | 0.15 | 0.39 | 0.03 | 0.00 |
| 30 | 0.01 | 0.10 | 0.00 | 0.00 | 0.00 | 0.19 | 0.01 | 0.02 |
| 20 | 0.02 | 0.42 | 0.00 | 0.14 | 0.01 | 0.22 | 0.01 | 0.13 |
| 10 | 0.02 | 0.21 | 0.00 | 0.00 | 0.07 | 0.01 | 0.03 | 0.33 |
| 0 | 0.10 | 0.17 | 0.00 | 0.00 | 0.02 | 0.00 | 0.09 | 0.23 |
| -10 | 0.36 | 0.24 | 0.09 | 0.00 | 0.00 | 0.00 | 0.09 | 0.08 |
| -20 | 0.64 | 0.34 | 0.82 | 0.13 | 0.08 | 0.00 | 0.05 | 0.03 |
| -30 | 0.87 | 0.43 | 0.94 | 0.24 | 0.20 | 0.00 | 0.06 | 0.02 |
| -20 | 0.73 | 0.01 | 0.52 | 0.00 | 0.48 | 0.01 | 0.06 | 0.00 |
| -10 | 0.47 | 0.00 | 0.38 | 0.00 | 0.47 | 0.05 | 0.07 | 0.00 |

Table A.11 筋張力の最大値 (全筋 S 制御 (バネ定数大))
The maximum of muscle-tension (all S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.41 | 0.11 | 0.11 | 0.17 | 0.53 | 0.09 | 0.14 | 0.08 |
| -10 | 0.27 | 0.10 | 0.12 | 0.08 | 0.76 | 0.32 | 0.04 | 0.00 |
| -20 | 0.43 | 0.34 | 0.19 | 0.06 | 0.27 | 0.21 | 0.12 | 0.04 |
| -30 | 0.58 | 0.70 | 0.50 | 0.08 | 0.18 | 0.17 | 0.19 | 0.31 |
| -20 | 0.31 | 0.13 | 0.07 | 0.01 | 0.04 | 0.00 | 0.42 | 0.66 |
| -10 | 0.22 | 0.01 | 0.05 | 0.05 | 0.03 | 0.00 | 0.27 | 0.48 |
| 0 | 0.16 | 0.00 | 0.10 | 0.14 | 0.20 | 0.00 | 0.17 | 0.14 |
| 10 | 0.08 | 0.00 | 0.65 | 0.20 | 0.47 | 0.11 | 0.06 | 0.03 |
| 20 | 0.04 | 0.00 | 0.56 | 0.31 | 0.86 | 0.21 | 0.08 | 0.03 |
| 30 | 0.02 | 0.00 | 0.61 | 0.42 | 2.28 | 0.98 | 0.06 | 0.03 |
| 20 | 0.02 | 0.00 | 0.47 | 0.16 | 2.26 | 0.95 | 0.01 | 0.00 |
| 10 | 0.02 | 0.00 | 0.47 | 0.13 | 2.19 | 0.88 | 0.02 | 0.00 |
| 0 | 0.08 | 0.00 | 0.17 | 0.032 | 1.51 | 0.63 | 0.01 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.002 | 0.00 | 0.51 | 0.03 | 0.81 | 0.23 | 0.04 | 0.00 |
| 20 | 0.00 | 0.00 | 0.51 | 0.19 | 0.80 | 0.37 | 0.08 | 0.00 |
| 30 | 0.00 | 0.01 | 0.41 | 0.51 | 0.47 | 0.34 | 0.03 | 0.01 |
| 20 | 0.02 | 0.01 | 0.01 | 0.30 | 0.21 | 0.33 | 0.02 | 0.04 |
| 10 | 0.01 | 0.06 | 0.00 | 0.15 | 0.44 | 0.23 | 0.01 | 0.03 |
| 0 | 0.09 | 0.13 | 0.01 | 0.14 | 0.48 | 0.10 | 0.02 | 0.02 |
| -10 | 0.60 | 0.23 | 0.42 | 0.19 | 0.46 | 0.01 | 0.02 | 0.01 |
| -20 | 0.92 | 0.41 | 1.18 | 0.45 | 0.64 | 0.09 | 0.02 | 0.00 |
| -30 | 1.02 | 0.45 | 2.60 | 0.61 | 0.84 | 0.17 | 0.00 | 0.00 |
| -20 | 0.68 | 0.04 | 0.49 | 0.04 | 1.38 | 0.19 | 0.03 | 0.00 |
| -10 | 0.56 | 0.01 | 0.45 | 0.00 | 0.97 | 0.15 | 0.07 | 0.00 |

Table A.12 筋張力の最大値 (L 制御 +LL 制御)
The maximum of muscle-tension (L-mode and LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.00 | 0.17 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| -10 | 0.06 | 0.20 | 0.08 | 0.06 | 0.19 | 0.26 | 0.02 | 0.00 |
| -20 | 0.13 | 0.23 | 0.01 | 0.00 | 0.00 | 0.08 | 0.07 | 0.02 |
| -30 | 0.22 | 0.37 | 0.00 | 0.00 | 0.01 | 0.13 | 0.18 | 0.15 |
| -20 | 0.08 | 0.12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.44 | 0.65 |
| -10 | 0.01 | 0.07 | 0.04 | 0.09 | 0.00 | 0.00 | 0.17 | 0.39 |
| 0 | 0.00 | 0.01 | 0.222 | 0.570 | 0.18 | 0.20 | 0.00 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.28 | 0.62 | 1.36 | 1.27 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.09 | 0.77 | 0.58 | 1.01 | 0.00 | 0.00 |
| 30 | 0.00 | 0.00 | 0.02 | 0.69 | 0.29 | 0.99 | 0.00 | 0.00 |
| 20 | 0.00 | 0.00 | 0.01 | 0.87 | 0.26 | 0.93 | 0.00 | 0.00 |
| 10 | 0.00 | 0.00 | 0.06 | 0.68 | 0.58 | 0.92 | 0.00 | 0.00 |
| 0 | 0.00 | 0.01 | 0.21 | 0.41 | 0.61 | 0.60 | 0.00 | 0.00 |
| -10 | 0.00 | 0.01 | 0.36 | 0.28 | 0.71 | 0.51 | 0.00 | 0.00 |
| -20 | 0.10 | 0.09 | 0.20 | 0.18 | 0.94 | 0.48 | 0.00 | 0.00 |
| -30 | 0.21 | 0.17 | 0.56 | 0.09 | 0.92 | 0.15 | 0.00 | 0.00 |
| -20 | 0.11 | 0.09 | 0.62 | 0.03 | 0.94 | 0.14 | 0.00 | 0.00 |
| -10 | 0.00 | 0.01 | 0.23 | 0.02 | 0.79 | 0.18 | 0.00 | 0.00 |

Table A.13　筋張力の最大値 (L 制御 +S 制御 (バネ定数中))
The maximum of muscle-tension (L-mode and S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.01 | 0.01 | 0.68 | 0.25 | 0.60 | 0.15 | 0.03 | 0.00 |
| -10 | 0.02 | 0.01 | 0.28 | 0.07 | 0.70 | 0.29 | 0.04 | 0.01 |
| -20 | 0.09 | 0.07 | 0.28 | 0.05 | 0.43 | 0.20 | 0.08 | 0.01 |
| -30 | 0.30 | 0.14 | 0.26 | 0.06 | 0.07 | 0.09 | 0.20 | 0.03 |
| -20 | 0.10 | 0.03 | 0.04 | 0.01 | 0.01 | 0.00 | 0.55 | 0.11 |
| -10 | 0.12 | 0.22 | 0.03 | 0.00 | 0.08 | 0.00 | 0.96 | 0.68 |
| 0 | 0.00 | 0.00 | 0.34 | 0.07 | 0.05 | 0.00 | 0.68 | 0.26 |
| 10 | 0.05 | 0.00 | 0.88 | 0.24 | 0.47 | 0.00 | 0.26 | 0.03 |
| 20 | 0.07 | 0.00 | 2.02 | 0.57 | 0.81 | 0.10 | 0.07 | 0.00 |
| 30 | 0.16 | 0.00 | 2.56 | 0.68 | 2.19 | 1.36 | 0.11 | 0.00 |
| 20 | 0.04 | 0.00 | 0.59 | 0.25 | 2.16 | 1.36 | 0.01 | 0.01 |
| 10 | 0.01 | 0.01 | 1.67 | 0.32 | 1.85 | 1.03 | 0.01 | 0.01 |
| 0 | 0.00 | 0.00 | 0.67 | 0.22 | 1.23 | 0.56 | 0.00 | 0.01 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.72 | 0.25 | 0.55 | 0.40 | 0.00 | 0.00 |
| 20 | 0.01 | 0.00 | 0.53 | 0.29 | 0.57 | 0.67 | 0.00 | 0.00 |
| 30 | 0.01 | 0.00 | 0.09 | 0.64 | 0.72 | 1.54 | 0.00 | 0.00 |
| 20 | 0.02 | 0.00 | 0.10 | 0.89 | 1.56 | 1.60 | 0.01 | 0.00 |
| 10 | 0.01 | 0.00 | 0.42 | 0.41 | 0.98 | 1.32 | 0.01 | 0.00 |
| 0 | 0.04 | 0.00 | 0.55 | 0.30 | 0.78 | 0.64 | 0.04 | 0.00 |
| -10 | 0.09 | 0.05 | 1.04 | 0.30 | 0.94 | 0.29 | 0.00 | 0.00 |
| -20 | 0.45 | 0.15 | 2.21 | 0.34 | 1.24 | 0.14 | 0.02 | 0.00 |
| -30 | 1.90 | 0.31 | 5.18 | 0.39 | 1.45 | 0.22 | 0.00 | 0.00 |
| -20 | 0.23 | 0.08 | 2.99 | 0.20 | 0.80 | 0.05 | 0.00 | 0.01 |
| -10 | 0.07 | 0.01 | 2.28 | 0.16 | 0.56 | 0.10 | 0.00 | 0.01 |

Table A.14 筋張力の最大値 (L 制御 +S 制御 (バネ定数大))
The maximum of muscle-tension (L-mode and S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.09 | 0.00 | 0.30 | 0.09 | 0.51 | 0.18 | 0.08 | 0.05 |
| -10 | 0.13 | 0.00 | 0.21 | 0.04 | 0.66 | 0.54 | 0.00 | 0.01 |
| -20 | 0.28 | 0.00 | 0.06 | 0.05 | 0.41 | 0.50 | 0.01 | 0.00 |
| -30 | 0.58 | 0.09 | 0.08 | 0.11 | 0.03 | 0.22 | 0.10 | 0.04 |
| -20 | 0.32 | 0.01 | 0.05 | 0.00 | 0.03 | 0.00 | 0.45 | 0.58 |
| -10 | 0.49 | 0.02 | 0.04 | 0.01 | 0.04 | 0.00 | 0.66 | 0.92 |
| 0 | 0.10 | 0.00 | 0.07 | 0.03 | 0.03 | 0.00 | 0.39 | 0.33 |
| 10 | 0.07 | 0.00 | 0.45 | 0.16 | 0.44 | 0.00 | 0.12 | 0.09 |
| 20 | 0.10 | 0.00 | 2.15 | 0.26 | 0.93 | 0.25 | 0.03 | 0.09 |
| 30 | 0.14 | 0.00 | 2.80 | 0.34 | 1.33 | 0.80 | 0.10 | 0.23 |
| 20 | 0.07 | 0.00 | 0.93 | 0.20 | 1.43 | 1.07 | 0.01 | 0.02 |
| 10 | 0.03 | 0.00 | 0.59 | 0.18 | 1.35 | 0.99 | 0.01 | 0.01 |
| 0 | 0.03 | 0.00 | 0.23 | 0.09 | 1.25 | 0.75 | 0.01 | 0.00 |
| roll | | | | | | | | |
| 10 | 0.00 | 0.00 | 0.51 | 0.12 | 0.55 | 0.52 | 0.00 | 0.00 |
| 20 | 0.02 | 0.00 | 0.37 | 0.17 | 0.46 | 0.80 | 0.00 | 0.00 |
| 30 | 0.03 | 0.00 | 0.10 | 0.26 | 0.32 | 1.34 | 0.00 | 0.00 |
| 20 | 0.02 | 0.00 | 0.05 | 0.70 | 1.16 | 1,49 | 0.02 | 0.00 |
| 10 | 0.02 | 0.00 | 0.19 | 0.35 | 0.99 | 1.30 | 0.01 | 0.00 |
| 0 | 0.03 | 0.00 | 0.23 | 0.22 | 0.95 | 0.73 | 0.01 | 0.00 |
| -10 | 0.15 | 0.00 | 0.53 | 0.20 | 0.99 | 0.43 | 0.02 | 0.00 |
| -20 | 0.64 | 0.07 | 0.91 | 0.36 | 0.99 | 0.26 | 0.02 | 0.00 |
| -30 | 1.70 | 0.34 | 3.19 | 0.67 | 0.94 | 0.34 | 0.02 | 0.00 |
| -20 | 0.41 | 0.01 | 1.84 | 0.17 | 0.84 | 0.13 | 0.01 | 0.01 |
| -10 | 0.18 | 0.00 | 1.14 | 0.06 | 0.75 | 0.14 | 0.00 | 0.01 |

Table A.15　モータ電流の平均 (全筋 L 制御)
The average of motor-current (all L-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.19 | 0.11 | 0.06 | 0.18 | 0.04 | 0.09 | 0.24 | 0.16 |
| -10 | 0.17 | 0.10 | 0.13 | 0.25 | 0.10 | 0.23 | 0.24 | 0.07 |
| -20 | 0.46 | 0.25 | 0.11 | 0.15 | 0.06 | 0.14 | 0.28 | 0.19 |
| -30 | 1.06 | 0.75 | 0.14 | 0.16 | 0.06 | 0.24 | 0.38 | 0.35 |
| -20 | 0.16 | 0.11 | 0.08 | 0.16 | 0.02 | 0.08 | 3.82 | 3.03 |
| -10 | 0.14 | 0.09 | 0.08 | 0.16 | 0.03 | 0.09 | 2.00 | 2.01 |
| 0 | 0.20 | 0.20 | 0.06 | 0.13 | 0.02 | 0.09 | 1.13 | 1.33 |
| 10 | 0.18 | 0.20 | 0.11 | 0.15 | 0.02 | 0.08 | 0.57 | 0.89 |
| 20 | 0.15 | 0.13 | 0.39 | 0.33 | 0.10 | 0.12 | 0.23 | 0.21 |
| 30 | 0.14 | 0.12 | 1.20 | 1.49 | 0.42 | 0.35 | 0.34 | 0.26 |
| 20 | 0.11 | 0.12 | 0.14 | 0.23 | 2.35 | 3.21 | 0.16 | 0.08 |
| 10 | 0.10 | 0.11 | 0.07 | 0.17 | 1.23 | 1.60 | 0.17 | 0.09 |
| 0 | 0.09 | 0.08 | 0.20 | 0.48 | 0.50 | 1.00 | 0.17 | 0.10 |
| roll | | | | | | | | |
| 10 | 0.18 | 0.08 | 0.07 | 0.81 | 0.21 | 0.06 | 0.27 | 0.07 |
| 20 | 0.16 | 0.09 | 0.05 | 0.63 | 0.07 | 0.06 | 0.47 | 0.09 |
| 30 | 0.15 | 0.12 | 0.05 | 0.81 | 0.05 | 0.06 | 0.35 | 0.14 |
| 20 | 0.08 | 1.01 | 0.32 | 0.15 | 0.03 | 0.18 | 0.16 | 1.39 |
| 10 | 0.10 | 0.61 | 0.15 | 0.15 | 0.03 | 0.11 | 0.20 | 1.26 |
| 0 | 0.08 | 0.47 | 0.20 | 0.15 | 0.03 | 0.11 | 0.22 | 1.13 |
| -10 | 0.16 | 0.90 | 0.31 | 0.15 | 0.03 | 0.11 | 0.21 | 0.69 |
| -20 | 0.23 | 0.82 | 0.53 | 0.15 | 0.04 | 0.10 | 0.22 | 0.58 |
| -30 | 0.53 | 0.82 | 0.74 | 0.17 | 0.04 | 0.11 | 0.26 | 0.34 |
| -20 | 1.26 | 0.09 | 0.06 | 0.59 | 0.09 | 0.08 | 1.50 | 0.07 |
| -10 | 0.81 | 0.09 | 0.08 | 0.64 | 0.11 | 0.08 | 1.49 | 0.10 |

Table A.16　モータ電流の平均 (全筋 LL 制御)
The average of motor-current (all LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.29 | 0.45 | 0.63 | 1.08 | 0.23 | 0.18 | 0.38 | 0.22 |
| -10 | 0.15 | 0.56 | 0.37 | 0.51 | 0.84 | 1.86 | 0.89 | 0.19 |
| -20 | 0.40 | 0.48 | 0.22 | 0.36 | 1.30 | 2.21 | 0.71 | 0.18 |
| -30 | 0.81 | 0.78 | 0.15 | 0.23 | 0.78 | 1.30 | 0.81 | 0.27 |
| -20 | 0.14 | 0.20 | 0.07 | 0.13 | 0.02 | 0.07 | 0.79 | 2.53 |
| -10 | 0.14 | 0.19 | 0.12 | 0.21 | 0.03 | 0.16 | 0.93 | 1.75 |
| 0 | 0.77 | 0.95 | 0.57 | 0.33 | 0.08 | 0.07 | 1.62 | 1.17 |
| roll | | | | | | | | |
| 10 | 0.26 | 0.17 | 0.12 | 1.55 | 1.53 | 0.18 | 1.17 | 0.15 |
| 20 | 0.20 | 0.19 | 0.19 | 1.16 | 1.12 | 0.08 | 1.66 | 0.19 |
| 30 | 0.14 | 0.62 | 0.68 | 0.71 | 0.81 | 0.05 | 1.36 | 0.26 |
| 20 | 0.08 | 0.81 | 1.15 | 0.16 | 0.03 | 1.22 | 0.29 | 1.59 |
| 10 | 0.09 | 0.85 | 1.04 | 0.20 | 0.03 | 1.49 | 0.27 | 1.58 |
| 0 | 0.15 | 0.45 | 0.83 | 0.24 | 0.04 | 1.32 | 0.27 | 1.32 |
| -10 | 0.25 | 0.77 | 0.59 | 0.25 | 0.04 | 0.98 | 0.27 | 0.92 |
| -20 | 0.36 | 1.95 | 1.98 | 0.28 | 0.02 | 0.87 | 0.27 | 0.50 |
| -30 | 1.00 | 0.79 | 0.73 | 0.35 | 0.05 | 0.18 | 0.30 | 0.29 |
| -20 | 1.93 | 0.27 | 0.15 | 1.97 | 0.41 | 0.06 | 1.55 | 0.08 |
| -10 | 1.35 | 0.22 | 0.13 | 1.62 | 0.56 | 0.08 | 1.76 | 0.17 |

Table A.17　モータ電流の平均 (全筋 S 制御 (バネ定数中))
The average of motor-current (all S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.27 | 2.34 | 0.07 | 0.75 | 0.03 | 0.05 | 0.27 | 0.22 |
| -10 | 0.38 | 1.47 | 0.04 | 3.71 | 3.06 | 0.32 | 0.16 | 0.06 |
| -20 | 0.59 | 0.42 | 0.07 | 0.48 | 0.03 | 0.72 | 0.16 | 0.07 |
| -30 | 0.93 | 0.70 | 0.08 | 0.08 | 0.03 | 1.76 | 0.33 | 0.44 |
| -20 | 0.53 | 0.42 | 0.08 | 0.07 | 0.02 | 0.06 | 0.96 | 1.17 |
| -10 | 0.42 | 0.79 | 0.08 | 0.06 | 0.03 | 0.06 | 0.38 | 0.77 |
| 0 | 0.24 | 0.15 | 0.08 | 0.06 | 0.03 | 0.07 | 0.33 | 0.50 |
| roll | | | | | | | | |
| 10 | 0.33 | 0.57 | 0.20 | 0.41 | 0.38 | 0.84 | 0.20 | 0.28 |
| 20 | 0.13 | 0.15 | 0.08 | 0.28 | 3.84 | 0.34 | 0.16 | 0.06 |
| 30 | 0.13 | 0.16 | 0.08 | 0.06 | 0.03 | 0.25 | 0.17 | 0.07 |
| 20 | 0.12 | 0.90 | 0.16 | 0.07 | 0.03 | 0.20 | 0.17 | 0.96 |
| 10 | 0.13 | 0.39 | 0.08 | 0.06 | 0.03 | 3.68 | 0.12 | 0.33 |
| 0 | 0.23 | 0.48 | 0.08 | 0.07 | 0.02 | 0.05 | 0.17 | 0.40 |
| -10 | 0.45 | 0.57 | 0.06 | 2.20 | 0.02 | 0.05 | 0.17 | 0.37 |
| -20 | 0.78 | 0.53 | 0.30 | 0.28 | 0.53 | 0.06 | 0.17 | 0.11 |
| -30 | 1.12 | 0.77 | 0.37 | 0.17 | 2.16 | 0.05 | 0.17 | 0.07 |
| -20 | 0.88 | 0.15 | 0.08 | 0.06 | 0.18 | 3.66 | 0.12 | 0.07 |
| -10 | 0.63 | 0.15 | 0.08 | 0.23 | 0.32 | 0.18 | 0.17 | 0.07 |

Table A.18　モータ電流の平均 (全筋 S 制御 (バネ定数大))
The average of motor-current (all S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.38 | 1.89 | 2.51 | 3.93 | 0.54 | 3.46 | 0.12 | 1.46 |
| -10 | 1.18 | 4.33 | 1.61 | 0.42 | 0.62 | 0.36 | 0.16 | 0.08 |
| -20 | 0.44 | 1.96 | 3.73 | 0.17 | 0.02 | 0.36 | 0.16 | 3.66 |
| -30 | 0.57 | 0.63 | 1.60 | 0.32 | 0.04 | 0.25 | 0.30 | 0.35 |
| -20 | 0.34 | 2.29 | 0.90 | 0.09 | 0.02 | 0.06 | 0.65 | 0.94 |
| -10 | 0.41 | 0.20 | 4.82 | 1.36 | 0.03 | 0.05 | 0.71 | 0.46 |
| 0 | 0.27 | 0.13 | 0.65 | 4.12 | 3.71 | 0.05 | 0.38 | 3.10 |
| 10 | 1.51 | 0.13 | 0.29 | 0.82 | 0.18 | 3.75 | 0.11 | 0.21 |
| 20 | 0.13 | 0.14 | 0.81 | 0.38 | 2.53 | 0.30 | 0.16 | 2.44 |
| 30 | 0.14 | 0.14 | 0.48 | 0.32 | 0.87 | 1.01 | 0.18 | 3.00 |
| 20 | 0.13 | 0.14 | 0.22 | 0.31 | 1.08 | 1.02 | 0.15 | 0.07 |
| 10 | 0.12 | 0.14 | 2.92 | 1.00 | 0.98 | 0.96 | 0.15 | 0.06 |
| 0 | 0.16 | 0.14 | 4.16 | 2.75 | 0.79 | 0.64 | 0.15 | 0.07 |
| roll | | | | | | | | |
| 10 | 0.12 | 0.14 | 4.74 | 0.46 | 0.25 | 0.06 | 0.16 | 0.07 |
| 20 | 0.13 | 0.14 | 0.32 | 0.54 | 0.20 | 0.47 | 0.16 | 0.07 |
| 30 | 0.12 | 0.52 | 0.61 | 0.38 | 0.09 | 0.38 | 0.16 | 3.57 |
| 20 | 0.13 | 0.39 | 0.47 | 0.06 | 3.91 | 0.76 | 0.16 | 3.54 |
| 10 | 0.09 | 2.34 | 1.56 | 0.06 | 2.11 | 0.26 | 0.16 | 0.06 |
| 0 | 0.15 | 4.19 | 2.30 | 0.14 | 0.27 | 3.31 | 0.11 | 0.07 |
| -10 | 0.63 | 0.49 | 0.29 | 0.27 | 1.23 | 4.33 | 0.11 | 0.98 |
| -20 | 0.93 | 0.97 | 0.59 | 0.39 | 0.63 | 3.80 | 0.11 | 0.07 |
| -30 | 1.31 | 1.15 | 0.91 | 0.34 | 0.35 | 0.26 | 0.16 | 0.07 |
| -20 | 0.83 | 1.42 | 2.10 | 0.12 | 0.46 | 0.29 | 0.16 | 0.07 |
| -10 | 0.47 | 0.90 | 0.08 | 0.37 | 0.42 | 0.39 | 0.16 | 0.07 |

Table A.19　モータ電流の平均 (L 制御 +LL 制御)
The average of motor-current (L-mode and LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.10 | 0.15 | 0.08 | 0.15 | 0.03 | 0.08 | 0.19 | 0.08 |
| -10 | 0.09 | 0.11 | 0.15 | 0.70 | 0.25 | 0.81 | 0.16 | 0.10 |
| -20 | 0.14 | 0.31 | 0.12 | 0.46 | 0.10 | 0.43 | 0.17 | 0.09 |
| -30 | 0.63 | 0.79 | 0.13 | 0.27 | 0.07 | 0.33 | 0.21 | 0.06 |
| -20 | 0.09 | 0.10 | 0.08 | 0.15 | 0.04 | 0.07 | 1.70 | 1.49 |
| -10 | 0.09 | 0.10 | 0.06 | 0.13 | 0.05 | 0.08 | 0.63 | 0.61 |
| 0 | 0.11 | 0.23 | 0.17 | 0.21 | 0.02 | 0.11 | 0.20 | 0.10 |
| roll | | | | | | | | |
| 10 | 0.11 | 0.12 | 0.09 | 0.98 | 1.99 | 0.29 | 0.20 | 0.09 |
| 20 | 0.15 | 0.13 | 0.14 | 0.90 | 0.86 | 0.07 | 0.23 | 0.10 |
| 30 | 0.13 | 0.12 | 0.31 | 0.46 | 0.67 | 0.05 | 0.21 | 0.10 |
| 20 | 0.10 | 0.15 | 1.61 | 0.13 | 0.06 | 1.88 | 0.18 | 0.10 |
| 10 | 0.11 | 0.16 | 1.12 | 0.13 | 0.10 | 2.33 | 0.16 | 0.10 |
| 0 | 0.11 | 0.30 | 0.75 | 0.15 | 0.13 | 1.37 | 0.17 | 0.09 |
| -10 | 0.11 | 0.33 | 0.52 | 0.21 | 0.09 | 0.98 | 0.17 | 0.09 |
| -20 | 0.09 | 0.30 | 0.28 | 0.21 | 0.16 | 0.56 | 0.19 | 0.09 |
| -30 | 0.13 | 0.61 | 0.20 | 0.18 | 0.09 | 0.28 | 0.18 | 0.09 |
| -20 | 0.27 | 0.12 | 0.05 | 1.62 | 1.56 | 0.05 | 0.20 | 0.08 |
| -10 | 0.15 | 0.09 | 0.06 | 1.36 | 1.42 | 0.07 | 0.21 | 0.10 |

Table A.20 モータ電流の平均 (L 制御 +S 制御 (バネ定数中))
The average of motor-current (L-mode and S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.13 | 0.19 | 0.17 | 0.13 | 0.06 | 0.12 | 0.17 | 0.07 |
| -10 | 0.13 | 0.18 | 0.21 | 0.29 | 1.22 | 1.05 | 0.40 | 0.06 |
| -20 | 0.12 | 0.19 | 0.25 | 0.30 | 3.63 | 0.68 | 0.69 | 0.25 |
| -30 | 0.45 | 0.53 | 0.30 | 0.45 | 6.58 | 0.80 | 0.18 | 0.13 |
| -20 | 5.72 | 5.72 | 0.19 | 0.06 | 1.70 | 0.05 | 1.85 | 0.43 |
| -10 | 1.04 | 1.09 | 0.08 | 0.06 | 0.90 | 0.05 | 2.85 | 1.25 |
| 0 | 0.15 | 0.21 | 0.47 | 0.06 | 0.45 | 0.05 | 1.80 | 0.77 |
| 10 | 0.15 | 0.21 | 0.45 | 0.47 | 5.06 | 0.07 | 0.51 | 0.15 |
| 20 | 0.16 | 0.21 | 0.88 | 0.88 | 1.01 | 5.13 | 0.21 | 0.12 |
| 30 | 0.25 | 0.31 | 1.70 | 1.23 | 0.12 | 0.85 | 0.20 | 0.11 |
| 20 | 2.42 | 2.45 | 0.51 | 0.29 | 2.57 | 2.91 | 0.13 | 0.06 |
| 10 | 0.13 | 0.19 | 0.25 | 0.38 | 2.50 | 2.24 | 0.15 | 0.06 |
| 0 | 0.14 | 0.19 | 0.64 | 0.62 | 1.47 | 1.43 | 0.15 | 0.06 |
| roll | | | | | | | | |
| 10 | 0.15 | 0.21 | 0.47 | 0.71 | 0.97 | 0.06 | 0.22 | 0.06 |
| 20 | 0.15 | 0.21 | 0.44 | 0.45 | 4.31 | 0.65 | 0.22 | 0.06 |
| 30 | 0.15 | 0.21 | 0.82 | 0.16 | 7.58 | 1.55 | 0.18 | 0.08 |
| 20 | 0.13 | 0.19 | 2.65 | 0.37 | 0.23 | 7.92 | 0.04 | 0.09 |
| 10 | 0.13 | 0.19 | 1.40 | 0.30 | 0.91 | 2.92 | 0.13 | 0.11 |
| 0 | 1.82 | 1.85 | 0.90 | 0.20 | 4.85 | 2.15 | 0.72 | 0.09 |
| -10 | 0.20 | 0.26 | 0.96 | 0.37 | 0.55 | 1.20 | 0.17 | 0.09 |
| -20 | 4.71 | 4.75 | 1.03 | 0.42 | 1.81 | 0.69 | 0.25 | 0.09 |
| -30 | 0.75 | 0.79 | 1.06 | 0.51 | 1.46 | 0.58 | 0.16 | 0.14 |
| -20 | 0.18 | 0.24 | 0.40 | 1.41 | 1.82 | 4.87 | 0.13 | 0.05 |
| -10 | 0.13 | 0.19 | 0.49 | 1.07 | 1.44 | 0.99 | 0.17 | 0.06 |

Table A.21　モータ電流の平均 (L 制御 +S 制御 (バネ定数大))
The average of motor-current (L-mode and S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 2.06 | 2.09 | 0.13 | 0.17 | 2.37 | 2.07 | 0.14 | 0.21 |
| -10 | 2.99 | 3.03 | 0.17 | 0.24 | 1.21 | 1.90 | 0.15 | 0.06 |
| -20 | 0.35 | 0.40 | 0.24 | 0.13 | 0.78 | 1.38 | 0.16 | 0.06 |
| -30 | 0.68 | 0.75 | 0.32 | 0.26 | 4.22 | 0.89 | 0.78 | 1.51 |
| -20 | 6.01 | 6.01 | 0.08 | 0.08 | 0.07 | 0.05 | 1.54 | 1.02 |
| -10 | 5.56 | 5.58 | 0.08 | 0.06 | 0.05 | 0.05 | 2.09 | 1.52 |
| 0 | 0.18 | 0.25 | 1.99 | 0.57 | 0.03 | 0.06 | 1.01 | 0.47 |
| 10 | 0.17 | 0.23 | 4.51 | 0.22 | 5.67 | 0.06 | 0.41 | 0.28 |
| 20 | 0.20 | 0.27 | 1.02 | 0.76 | 3.21 | 0.97 | 0.24 | 0.25 |
| 30 | 0.40 | 0.45 | 1.55 | 1.04 | 1.17 | 0.91 | 0.32 | 0.67 |
| 20 | 0.64 | 0.70 | 0.56 | 0.32 | 2.26 | 3.27 | 0.13 | 4.47 |
| 10 | 3.76 | 3.81 | 2.62 | 4.08 | 1.81 | 3.43 | 0.14 | 0.11 |
| 0 | 3.05 | 3.08 | 0.27 | 0.24 | 1.50 | 2.77 | 0.16 | 0.06 |
| roll | | | | | | | | |
| 10 | 0.16 | 0.22 | 0.54 | 0.49 | 0.81 | 0.06 | 0.19 | 0.06 |
| 20 | 0.16 | 0.21 | 0.52 | 0.36 | 2.25 | 0.84 | 0.20 | 0.06 |
| 30 | 0.16 | 0.21 | 0.52 | 0.19 | 7.79 | 1.32 | 0.19 | 0.08 |
| 20 | 0.16 | 0.22 | 2.15 | 0.06 | 2.61 | 5.55 | 0.17 | 0.09 |
| 10 | 0.13 | 0.19 | 1.12 | 0.35 | 4.27 | 3.26 | 0.13 | 0.10 |
| 0 | 1.45 | 1.49 | 0.66 | 0.12 | 4.96 | 1.70 | 0.15 | 0.09 |
| -10 | 1.36 | 1.41 | 0.61 | 0.17 | 3.36 | 1.51 | 0.74 | 0.10 |
| -20 | 1.10 | 1.15 | 0.83 | 0.30 | 3.42 | 0.77 | 3.61 | 0.16 |
| -30 | 0.83 | 0.89 | 1.66 | 0.37 | 3.75 | 0.87 | 2.64 | 0.15 |
| -20 | 0.62 | 0.67 | 1.24 | 0.70 | 1.53 | 2.25 | 0.20 | 0.06 |
| -10 | 0.27 | 0.33 | 0.09 | 0.74 | 1.28 | 0.47 | 0.20 | 0.07 |

Table A.22　モータ電流の最大値 (全筋 L 制御)
The maximum of motor-current (all L-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.20 | 0.15 | 0.08 | 0.21 | 0.06 | 0.12 | 0.27 | 0.20 |
| -10 | 0.21 | 0.13 | 0.17 | 0.27 | 0.13 | 0.26 | 0.26 | 0.10 |
| -20 | 0.51 | 0.27 | 0.13 | 0.18 | 0.09 | 0.21 | 0.30 | 0.22 |
| -30 | 1.15 | 0.93 | 0.16 | 0.20 | 0.08 | 0.34 | 0.45 | 0.43 |
| -20 | 0.20 | 0.14 | 0.12 | 0.21 | 0.05 | 0.12 | 4.09 | 3.37 |
| -10 | 0.20 | 0.11 | 0.10 | 0.19 | 0.06 | 0.11 | 2.09 | 2.12 |
| 0 | 0.25 | 0.24 | 0.07 | 0.14 | 0.04 | 0.10 | 1.18 | 1.38 |
| 10 | 0.25 | 0.26 | 0.13 | 0.18 | 0.07 | 0.11 | 0.74 | 1.04 |
| 20 | 0.20 | 0.20 | 0.64 | 0.57 | 0.13 | 0.15 | 0.29 | 0.48 |
| 30 | 0.17 | 0.15 | 1.36 | 1.64 | 1.90 | 2.23 | 0.45 | 0.40 |
| 20 | 0.13 | 0.15 | 0.19 | 0.29 | 2.80 | 3.32 | 0.18 | 0.11 |
| 10 | 0.13 | 0.13 | 0.09 | 0.21 | 1.52 | 1.70 | 0.20 | 0.12 |
| 0 | 0.12 | 0.11 | 0.25 | 0.55 | 0.55 | 1.07 | 0.19 | 0.12 |
| roll | | | | | | | | |
| 10 | 0.21 | 0.10 | 0.10 | 0.86 | 0.23 | 0.07 | 0.29 | 0.09 |
| 20 | 0.20 | 0.12 | 0.08 | 0.78 | 0.10 | 0.09 | 0.51 | 0.11 |
| 30 | 0.16 | 0.15 | 0.07 | 0.86 | 0.07 | 0.07 | 0.38 | 0.16 |
| 20 | 0.12 | 1.17 | 0.41 | 0.16 | 0.07 | 0.20 | 0.18 | 1.49 |
| 10 | 0.15 | 0.76 | 0.23 | 0.21 | 0.07 | 0.16 | 0.23 | 1.43 |
| 0 | 0.10 | 0.65 | 0.28 | 0.19 | 0.05 | 0.15 | 0.24 | 1.23 |
| -10 | 0.18 | 0.97 | 0.35 | 0.19 | 0.04 | 0.13 | 0.22 | 0.79 |
| -20 | 0.25 | 0.91 | 0.60 | 0.16 | 0.06 | 0.12 | 0.24 | 0.68 |
| -30 | 2.36 | 0.99 | 0.94 | 0.40 | 0.07 | 0.15 | 0.46 | 0.50 |
| -20 | 1.40 | 0.13 | 0.14 | 0.71 | 0.12 | 0.10 | 1.67 | 0.08 |
| -10 | 0.89 | 0.12 | 0.10 | 0.74 | 0.15 | 0.11 | 1.57 | 0.13 |

Table A.23　モータ電流の最大値 (全筋 LL 制御)
The maximum of motor-current (all LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.36 | 0.51 | 0.70 | 1.15 | 0.31 | 0.23 | 0.46 | 0.27 |
| -10 | 0.20 | 0.75 | 0.48 | 0.63 | 0.99 | 1.97 | 1.14 | 0.26 |
| -20 | 0.46 | 0.56 | 0.27 | 0.42 | 1.42 | 2.28 | 3.04 | 0.26 |
| -30 | 0.96 | 0.93 | 0.21 | 0.29 | 0.96 | 1.44 | 5.80 | 0.34 |
| -20 | 0.18 | 0.21 | 0.10 | 0.16 | 0.04 | 0.09 | 5.71 | 2.57 |
| -10 | 0.17 | 0.21 | 0.15 | 0.26 | 0.05 | 0.19 | 2.31 | 1.84 |
| 0 | 0.98 | 1.57 | 2.59 | 0.40 | 0.23 | 0.13 | 1.96 | 1.35 |
| roll | | | | | | | | |
| 10 | 0.43 | 0.27 | 0.29 | 1.74 | 1.68 | 0.24 | 1.29 | 0.20 |
| 20 | 0.23 | 0.22 | 0.22 | 1.26 | 1.27 | 0.10 | 1.70 | 0.23 |
| 30 | 0.18 | 1.59 | 1.95 | 0.93 | 1.08 | 0.08 | 1.59 | 0.33 |
| 20 | 0.18 | 1.78 | 1.46 | 0.20 | 0.06 | 1.55 | 0.34 | 1.75 |
| 10 | 0.19 | 3.02 | 1.35 | 0.26 | 0.08 | 1.81 | 0.39 | 1.77 |
| 0 | 0.19 | 2.73 | 1.75 | 0.29 | 0.07 | 1.40 | 0.32 | 1.40 |
| -10 | 0.35 | 5.98 | 2.66 | 0.35 | 0.06 | 1.35 | 0.32 | 1.12 |
| -20 | 1.94 | 5.91 | 5.39 | 0.38 | 0.07 | 1.19 | 0.32 | 0.71 |
| -30 | 5.76 | 3.64 | 0.92 | 1.24 | 0.10 | 0.23 | 0.34 | 0.48 |
| -20 | 4.69 | 0.34 | 0.20 | 2.32 | 0.46 | 0.09 | 1.81 | 0.14 |
| -10 | 1.48 | 0.35 | 0.23 | 1.76 | 0.67 | 0.14 | 2.01 | 0.21 |

Table A.24　モータ電流の最大値 (全筋 S 制御 (バネ定数中))
The maximum of motor-current (all S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 2.31 | 6.62 | 0.12 | 6.90 | 0.07 | 0.10 | 5.54 | 3.57 |
| -10 | 2.06 | 6.60 | 0.09 | 7.32 | 7.59 | 2.97 | 0.19 | 0.09 |
| -20 | 1.58 | 2.33 | 0.11 | 5.90 | 0.04 | 6.01 | 0.18 | 0.10 |
| -30 | 2.26 | 3.26 | 0.10 | 1.11 | 0.04 | 7.53 | 5.29 | 2.67 |
| -20 | 1.64 | 2.91 | 0.10 | 0.34 | 0.04 | 0.07 | 5.91 | 2.44 |
| -10 | 1.82 | 5.32 | 0.09 | 0.09 | 0.12 | 0.09 | 3.64 | 2.68 |
| 0 | 1.06 | 0.17 | 0.32 | 0.08 | 0.04 | 1.15 | 3.45 | 3.10 |
| roll | | | | | | | | |
| 10 | 5.42 | 6.09 | 4.75 | 4.93 | 6.95 | 7.24 | 5.45 | 7.59 |
| 20 | 0.16 | 0.20 | 0.11 | 4.08 | 7.49 | 2.95 | 0.20 | 0.08 |
| 30 | 0.15 | 0.27 | 0.10 | 0.09 | 0.06 | 3.08 | 0.19 | 0.09 |
| 20 | 0.16 | 3.08 | 0.92 | 0.08 | 0.05 | 1.70 | 0.20 | 5.55 |
| 10 | 0.15 | 2.86 | 0.10 | 0.09 | 0.04 | 7.18 | 0.19 | 2.54 |
| 0 | 1.70 | 3.00 | 0.11 | 0.09 | 0.04 | 0.07 | 0.18 | 3.32 |
| -10 | 1.58 | 4.23 | 0.18 | 6.32 | 0.04 | 0.07 | 0.20 | 2.19 |
| -20 | 1.43 | 3.36 | 1.93 | 2.94 | 5.46 | 0.08 | 0.21 | 2.44 |
| -30 | 1.78 | 5.51 | 2.04 | 2.15 | 6.47 | 0.09 | 0.19 | 0.10 |
| -20 | 1.51 | 0.18 | 0.10 | 0.09 | 3.92 | 7.09 | 0.17 | 0.09 |
| -10 | 1.59 | 0.17 | 0.10 | 2.62 | 3.68 | 2.90 | 0.19 | 0.09 |

Table A.25　モータ電流の最大値 (全筋 S 制御 (バネ定数大))
The maximum of motor-current (all S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 5.26 | 6.33 | 6.29 | 7.13 | 6.73 | 6.83 | 4.88 | 6.68 |
| -10 | 5.79 | 6.56 | 5.33 | 5.04 | 5.29 | 4.72 | 0.18 | 0.86 |
| -20 | 1.52 | 6.34 | 6.92 | 1.97 | 0.05 | 2.72 | 0.20 | 6.75 |
| -30 | 1.98 | 4.00 | 6.26 | 2.75 | 1.50 | 3.14 | 5.08 | 2.89 |
| -20 | 2.04 | 6.29 | 4.55 | 3.22 | 0.04 | 0.10 | 5.59 | 2.69 |
| -10 | 3.55 | 2.81 | 6.72 | 5.58 | 0.04 | 0.07 | 5.33 | 2.55 |
| 0 | 1.63 | 0.16 | 5.00 | 6.77 | 7.22 | 0.09 | 4.93 | 6.84 |
| 10 | 5.22 | 0.16 | 1.34 | 5.47 | 3.50 | 6.90 | 0.18 | 3.63 |
| 20 | 0.22 | 0.16 | 4.67 | 2.63 | 6.20 | 3.82 | 0.38 | 6.54 |
| 30 | 1.02 | 0.16 | 2.59 | 2.92 | 4.83 | 3.55 | 3.22 | 7.49 |
| 20 | 0.67 | 0.18 | 1.21 | 2.66 | 5.02 | 3.32 | 0.18 | 0.10 |
| 10 | 0.15 | 0.18 | 6.52 | 5.79 | 4.66 | 3.09 | 0.18 | 0.09 |
| 0 | 1.67 | 0.16 | 6.70 | 6.49 | 4.88 | 4.30 | 0.19 | 0.12 |
| roll | | | | | | | | |
| 10 | 0.16 | 0.16 | 6.99 | 3.69 | 3.55 | 0.07 | 0.20 | 0.10 |
| 20 | 0.15 | 0.17 | 1.65 | 3.22 | 3.72 | 3.26 | 0.18 | 0.09 |
| 30 | 0.14 | 3.33 | 1.57 | 2.81 | 2.23 | 2.89 | 0.20 | 6.29 |
| 20 | 1.00 | 4.04 | 1.50 | 0.09 | 7.17 | 5.24 | 1.30 | 6.85 |
| 10 | 0.14 | 6.10 | 5.51 | 0.11 | 6.12 | 2.69 | 0.18 | 0.08 |
| 0 | 1.02 | 6.49 | 6.28 | 2.81 | 4.44 | 6.64 | 0.17 | 0.29 |
| -10 | 2.01 | 2.34 | 1.14 | 2.47 | 5.49 | 6.99 | 0.59 | 5.58 |
| -20 | 1.72 | 4.66 | 2.05 | 2.88 | 5.29 | 6.52 | 0.17 | 0.09 |
| -30 | 2.51 | 3.50 | 1.56 | 2.45 | 5.11 | 2.48 | 0.18 | 0.10 |
| -20 | 1.81 | 5.90 | 6.55 | 2.42 | 5.32 | 2.93 | 0.18 | 0.08 |
| -10 | 2.51 | 5.41 | 0.10 | 3.78 | 3.98 | 3.47 | 0.18 | 0.09 |

Table A.26 モータ電流の最大値 (L 制御 +LL 制御)
The maximum of motor-current (L-mode and LL-mode)

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.20 | 0.85 | 0.20 | 0.34 | 0.09 | 0.23 | 0.32 | 0.18 |
| -10 | 0.10 | 0.14 | 0.18 | 0.76 | 0.31 | 0.88 | 0.17 | 0.15 |
| -20 | 0.16 | 0.35 | 0.15 | 0.55 | 0.15 | 0.65 | 0.18 | 0.12 |
| -30 | 0.67 | 0.84 | 0.17 | 0.30 | 0.10 | 0.42 | 0.24 | 0.07 |
| -20 | 0.11 | 0.12 | 0.10 | 0.17 | 0.05 | 0.12 | 1.73 | 1.51 |
| -10 | 0.12 | 0.12 | 0.10 | 0.14 | 0.07 | 0.14 | 0.68 | 0.74 |
| 0 | 0.15 | 0.27 | 0.21 | 0.23 | 0.04 | 0.16 | 0.24 | 0.13 |
| roll | | | | | | | | |
| 10 | 0.13 | 0.13 | 0.15 | 1.07 | 2.09 | 0.32 | 0.21 | 0.12 |
| 20 | 0.32 | 0.26 | 0.23 | 1.31 | 1.14 | 0.26 | 0.27 | 0.14 |
| 30 | 0.15 | 0.15 | 0.46 | 0.52 | 0.75 | 0.07 | 0.23 | 0.13 |
| 20 | 0.13 | 0.17 | 1.71 | 0.15 | 0.09 | 1.93 | 0.20 | 0.12 |
| 10 | 0.17 | 0.24 | 1.28 | 0.17 | 0.13 | 2.60 | 0.18 | 0.12 |
| 0 | 0.15 | 0.43 | 0.95 | 0.18 | 0.15 | 1.5 | 0.19 | 0.12 |
| -10 | 0.17 | 0.43 | 0.63 | 0.26 | 0.12 | 1.57 | 0.20 | 0.13 |
| -20 | 0.10 | 0.43 | 0.35 | 0.26 | 0.23 | 0.63 | 0.21 | 0.13 |
| -30 | 0.15 | 0.79 | 0.23 | 0.20 | 0.12 | 0.40 | 0.20 | 0.13 |
| -20 | 0.29 | 0.13 | 0.08 | 1.67 | 1.70 | 0.07 | 0.25 | 0.13 |
| -10 | 0.17 | 0.10 | 0.09 | 1.42 | 1.51 | 0.10 | 0.24 | 0.13 |

Table A.27　モータ電流の最大値 (L 制御 +S 制御 (バネ定数中))
The maximum of motor-current (L-mode and S-mode (middle coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 0.17 | 0.24 | 0.87 | 5.57 | 7.21 | 7.31 | 1.31 | 0.10 |
| -10 | 0.15 | 0.23 | 0.26 | 0.36 | 1.31 | 1.10 | 6.71 | 0.10 |
| -20 | 0.15 | 0.23 | 0.29 | 0.37 | 3.79 | 0.92 | 6.58 | 5.82 |
| -30 | 6.35 | 6.32 | 1.79 | 7.68 | 6.72 | 8.67 | 1.29 | 3.33 |
| -20 | 9.22 | 9.32 | 5.27 | 0.13 | 2.07 | 0.08 | 1.97 | 0.49 |
| -10 | 6.82 | 6.89 | 0.10 | 0.13 | 5.68 | 0.09 | 3.05 | 1.45 |
| 0 | 0.18 | 0.24 | 6.81 | 0.10 | 3.42 | 0.07 | 1.86 | 0.87 |
| 10 | 0.20 | 0.26 | 6.51 | 3.29 | 10.08 | 2.58 | 0.76 | 0.27 |
| 20 | 0.21 | 0.26 | 0.95 | 0.96 | 7.59 | 9.16 | 0.35 | 0.16 |
| 30 | 8.32 | 8.34 | 1.91 | 1.65 | 2.67 | 5.88 | 0.26 | 0.18 |
| 20 | 7.40 | 7.37 | 6.16 | 1.45 | 2.68 | 3.00 | 0.16 | 0.09 |
| 10 | 0.15 | 0.23 | 4.02 | 3.28 | 2.59 | 2.31 | 0.56 | 0.07 |
| 0 | 0.83 | 0.81 | 8.44 | 0.73 | 1.57 | 1.79 | 0.19 | 0.22 |
| roll | | | | | | | | |
| 10 | 0.18 | 0.24 | 6.29 | 0.76 | 1.09 | 0.07 | 0.23 | 0.11 |
| 20 | 0.18 | 0.26 | 5.64 | 0.62 | 4.39 | 4.41 | 0.25 | 0.08 |
| 30 | 0.70 | 0.71 | 6.55 | 4.17 | 7.84 | 4.43 | 0.23 | 1.83 |
| 20 | 0.15 | 0.23 | 2.69 | 3.04 | 3.08 | 8.01 | 0.06 | 0.11 |
| 10 | 0.16 | 0.22 | 1.50 | 1.21 | 5.33 | 3.03 | 0.16 | 0.13 |
| 0 | 7.64 | 7.59 | 0.98 | 1.30 | 9.03 | 2.26 | 6.45 | 0.11 |
| -10 | 5.21 | 4.72 | 1.03 | 1.45 | 7.65 | 1.28 | 1.45 | 0.12 |
| -20 | 8.35 | 8.37 | 1.12 | 1.71 | 7.79 | 0.77 | 6.05 | 0.11 |
| -30 | 6.61 | 6.45 | 4.93 | 1.90 | 7.87 | 2.34 | 0.18 | 0.19 |
| -20 | 0.26 | 0.34 | 5.45 | 1.52 | 1.96 | 8.92 | 0.23 | 0.09 |
| -10 | 0.18 | 0.26 | 6.70 | 1.20 | 1.53 | 6.15 | 0.21 | 0.27 |

Table A.28 モータ電流の最大値 (L 制御 +S 制御 (バネ定数大))
The maximum of motor-current (L-mode and S-mode (high coeff.))

| 姿勢 | CLA-RF | CLA-LF | CLA-RB | CLA-LB | HIP-RF | HIP-LF | HIP-RB | HIP-LB |
|---|---|---|---|---|---|---|---|---|
| pitch | | | | | | | | |
| 0 | 8.30 | 8.33 | 0.20 | 5.15 | 9.06 | 8.77 | 0.21 | 7.86 |
| -10 | 7.56 | 7.63 | 0.21 | 0.30 | 1.43 | 1.99 | 0.17 | 0.11 |
| -20 | 0.40 | 0.48 | 0.27 | 0.17 | 0.92 | 1.44 | 0.18 | 0.12 |
| -30 | 7.10 | 7.07 | 2.04 | 5.91 | 4.84 | 7.46 | 7.04 | 9.16 |
| -20 | 8.82 | 8.75 | 0.10 | 1.02 | 1.35 | 0.09 | 1.62 | 1.17 |
| -10 | 8.80 | 8.72 | 0.20 | 0.11 | 1.07 | 0.09 | 2.20 | 1.71 |
| 0 | 0.21 | 0.29 | 8.65 | 5.69 | 0.70 | 0.10 | 1.17 | 0.60 |
| 10 | 0.20 | 0.29 | 8.20 | 1.60 | 8.97 | 0.09 | 0.49 | 0.38 |
| 20 | 0.24 | 0.33 | 1.11 | 0.89 | 8.20 | 7.53 | 0.29 | 0.34 |
| 30 | 7.68 | 7.76 | 1.67 | 9.38 | 4.44 | 4.91 | 0.40 | 6.90 |
| 20 | 7.23 | 7.42 | 5.62 | 2.89 | 2.48 | 3.44 | 0.15 | 8.50 |
| 10 | 8.70 | 8.68 | 8.01 | 8.51 | 2.12 | 3.62 | 1.49 | 5.95 |
| 0 | 8.30 | 8.39 | 0.32 | 0.28 | 1.66 | 6.38 | 1.53 | 0.08 |
| roll | | | | | | | | |
| 10 | 0.20 | 0.27 | 5.87 | 0.52 | 0.92 | 0.09 | 0.21 | 0.13 |
| 20 | 0.21 | 0.25 | 6.89 | 0.48 | 2.31 | 4.15 | 0.23 | 0.08 |
| 30 | 0.39 | 0.43 | 6.40 | 1.04 | 7.98 | 4.77 | 2.35 | 1.89 |
| 20 | 2.59 | 2.94 | 2.28 | 0.09 | 7.37 | 5.67 | 5.71 | 0.13 |
| 10 | 0.31 | 0.35 | 1.23 | 2.63 | 8.44 | 3.35 | 0.15 | 0.13 |
| 0 | 7.48 | 7.37 | 0.75 | 0.72 | 9.09 | 1.82 | 0.18 | 0.11 |
| -10 | 7.76 | 7.58 | 0.70 | 0.65 | 8.28 | 1.59 | 7.56 | 0.12 |
| -20 | 7.12 | 7.20 | 1.01 | 1.51 | 8.61 | 0.88 | 8.02 | 0.20 |
| -30 | 6.65 | 6.48 | 1.82 | 2.39 | 8.91 | 7.66 | 7.82 | 2.44 |
| -20 | 0.75 | 0.81 | 7.13 | 0.90 | 1.62 | 7.99 | 0.26 | 0.43 |
| -10 | 0.29 | 0.37 | 1.21 | 0.89 | 1.34 | 4.54 | 0.23 | 0.51 |

# 付録 B

# H8 Library

## B.1 各ライブラリの詳説

### B.1.1 $I^2C$ 通信ライブラリ

$I^2C$ バスは，一つのバスに多数のデバイスを接続し，任意の 2 つの間，または，一つから残り全て (ブロードキャスト) の形で，シリアル通信できるプロトコルである．

**Reference**

`int initI2C(i2c_id_t id, unsigned char *buf, const struct i2c_bufsize *bufsize)`

$I^2C$ 関連の初期化．`id` には自分の ID を入れる． ID は 7bit(0x00 〜 0x7f)．クロック，ID，モードの設定，バッファの初期化，割り込み関数の設定など．`buf` にはマスタ・スレーブ各送受信バッファとして使うメモリ領域へのポインタ，`bufsize` にはバッファサイズの内訳を記述した構造体，

```
struct i2c_bufsize {
    i2c_msttxbuf_size_t     mst_txbuf;
    i2c_msttxpack_size_t    mst_tpack;
    i2c_mstrxbuf_size_t     mst_rxbuf;
    i2c_mstrxpack_size_t    mst_rpack;
    i2c_slvtxbuf_size_t     slv_txbuf;
    i2c_slvtxpack_size_t    slv_tpack;
    i2c_slvrxbuf_size_t     slv_rxbuf;
    i2c_slvrxpack_size_t    slv_rpack;
};
```

へのポインタを渡す．デフォルトのバッファサイズで良ければ，`NULL` を渡す．

`int process_i2c(void);`

現在の $I^2C$ の状況によって必要な機能を実行する関数．
送信バッファにデータがあればバスフリーかチェックし，フリーなら送信を開始する．
アービトレーション・エラーなどが起こっていれば，その処理．終了条件を出すべき状況ならば，終了条件を出しバスを解放するなど．
メインループで周期的に呼ぶべきである．特に，`i2c_putc(); i2c_puts(); i2c_put_pack();` などを実行した後は，`process_i2c();` をコールしなければ実際の送信は始まらない．

`i2c_data_t i2c_getc(void);`

```
i2c_data_t *i2c_gets(i2c_data_t *data, int n);
i2c_data_t *i2c_getall(i2c_data_t *data);
```

データを読み込む.

```
i2c_id_t i2c_get_id(void);
```

次にi2c_getc(); するデータの送信元のID を返す.

```
int i2c_putc(i2c_data_t c, i2c_id_t dest);
int i2c_puts(i2c_data_t *data, int n, i2c_id_t dest);
int i2c_put_pack(i2c_data_t *data, int n, i2c_id_t dest);
```

dest で指定した ID のプロセッサにデータをマスタ送信バッファに転送し、c を返す. その後process_i2c(); が呼ばれた時にバスが空いていれば, マスタ送信を開始します. i2c_puts(); とi2c_put_pack(); は, data から始まるn バイトのデータをバッファに転送し、バッファに転送したバイト数を返します。i2c_puts(); は、他のプロセッサと送信先が重なった時受信されるデータが混ざり合う可能性がありますが、n に限度はありません。i2c_put_pack(); は、他のプロセッサと送信先が重なった時も、受信されるデータはdata からn バイトは必ず連続になりますが、n に現在のマスタ送信バッファの空きバイト数以上を指定することはできません (バッファには何も転送されず、 0 が返る)。

```
i2c_msttxbuf_size_t i2c_txcount(void);
i2c_slvrxbuf_size_t i2c_rxcount(void);
i2c_slvtxbuf_size_t i2c_slv_txcount(void);
i2c_mstrxbuf_size_t i2c_mst_rxcount(void);
i2c_msttxpack_size_t i2c_number_of_tpacks(void);
i2c_slvrxpack_size_t i2c_number_of_rpacks(void);
i2c_mstrxpack_size_t i2c_number_of_reqpacks(void);
i2c_mstrxpack_size_t i2c_number_of_satisfied_requests(void);
i2c_slvrxbuf_size_t i2c_stored_amount_of_the_pack(void);

int i2c_set_slave_transmit_data(i2c_data_t *data, int n, i2c_id_t dest);

i2c_id_t i2c_get_request_data(i2c_data_t *data);

int i2c_request_data(int n, i2c_id_t src);

int i2c_rxbuf_flush(void);

int i2c_ungetc(i2c_data_t c);
int i2c_unget2bytes(i2c_data_t pre, i2c_data_t prepre);
```

### B.1.2 調歩同期式シリアル通信ライブラリ

ソースファイルは， serial.c
2 チャンネルあるシリアル通信インタフェース (SCI0, SCI1) を利用して， WS などのシリアルポートを通じて調歩同期式シリアル通信をするためのライブラリ.
通信条件は， 8bit，パリティなし，ストップビット 1．ボーレートは， H8 のクロック周波数が， 7.3728[MHz] または 14.7456[MHz] の時は， 9600 〜 115200[bps] に設定できる．(ハードウェアの最高速は， 14.7456[MHz] の時に 460800[bps])

#### Reference

以下の関数名中の`sci` の後ろにに`0` または`1` をつけて SCI0, SCI1 のどちらのチャンネルを利用するかを指定する．もちろん両方利用することも可能.
(例)`init_sci0(); sci1_getchar();` など

```
void init_sci(unsigned long bps,
              unsigned char *buf,
              const unsigned int bufsize[]);
```

シリアル通信の初期化．`bps` にはボーレートを渡す．`buf` には送受信バッファとして使うメモリ領域へのポインタ，`bufsize` にはバッファサイズの内訳を記述した`const` 変数を渡す．デフォルトのバッファサイズで良ければ，`NULL` を渡す.
その他，レジスタやバッファの初期化，割り込み時の関数の設定などを行う.

```
int sci_putchar(char c);
char *sci_puts(char *msg);
```

送信バッファにデータを格納.
`sci_puts()` は，`NULL` が来るまで送信バッファにデータを格納.

```
int sci_getchar(void);
int sci_getchar_unlock(void);
```

受信バッファからデータを収納.
受信データが無い場合，`sci_getchar()` は 1[byte] 受信するまでブロックし，
`sci_getchar_unlock()` は，`EOF` を返す.

```
int _sci_putchar(char c, void (*needed_processes)());
```

`sci_putchar()` と同機能だが，送信バッファフルでブロックしている間に，
`needed_processes` を実行する．`_sci_getchar()` および`_sci_puts()` も同様.

```
int sci_rxcount(void);
int sci_txcount(void);
```

送受信バッファに残っているデータのバイト数を返す.

**使用例**

エコーバックのプログラム．受信したデータをそのまま送信する．

```
#include <serial.h>
#define TBUF        64
#define RBUF        128
unsigned char       sci_buf[ TBUF + RBUF ];
const unsigned int sci_bufsize[2] = { TBUF, RBUF };

main()
{
    initSCI0(38400, sci_buf, sci_bufsize);
/*  または,
    initSCI0(38400, NULL, NULL); */

    int_enable();    /* 全割り込み許可 */

    while(1)
        if (sci0_rxcount() > 0) sci0_putchar(sci0_getchar());
}
```

**実装**

1[byte] 受信完了割り込み， 1[byte] 送信完了割り込みをそれぞれ利用している．送受信それぞれ，独立のリングバッファを持ち，putchar()，puts() は，バッファにデータを書き込み，getchar() は，バッファからデータを読み込む.
受信バッファフルになった時は，受信完了割り込みをマスクし，受信バッファに空きができた時にマスクを解除する．送信完了割り込みは送信バッファが空の時はマスクされている．送信バッファにデータが書き込まれると， 1[byte] 送信命令を実行後マスク解除され，バッファが空になるとマスクされる.

### B.1.3 AD 変換ライブラリ

ソースファイルは， ad.c
8 チャンネルある 10bit AD 変換器を利用するライブラリ.

**Reference**

```
void init_AD();
```

初期化． AD 変換のモード設定，割り込み関数の設定などを行う．

```
int new_ad_data(void);
```

新しいデータが入ったかどうかを判定する．ad_round_flg のクリアをする．

```
int get_ad_data(unsigned char n, unsigned short buf[]);
```

AD 変換されたデータをとってくる．n はチャンネル，buf[] はデータ．

**実装**

チャンネル 0 〜 3(またはチャンネル 4 〜 7) を順に AD 変換するスキャンモードを利用する．
4 チャンネル終了毎に割り込みがかかり，レジスタに格納された変換結果をメモリに転送する．全 8 チャンネルを変換し終ったら，ad_round_flg をセットする．
get_ad_data() 関数で，メモリにバッファされたデータを読み出す．

```
void init_AD(void)
{
    ・・・
    inttab[ADI] = (int)ad_finish;    /* 割り込みベクタ設定 */
    ・・・
}

static void ad_finish(void)
{
    static char group = 0 ;
    ADCSR &= (~ADST & ~ADF);    /* AD ストップ */
    if (group == 0) {
        group = 1;
        ad_buf[0] = ((unsigned short) ADDRA) >> 6;
        ・・・
        ADCSR |= (GROUP1 | ADST);    /* グループ→1, AD スタート */
    } else {
        group = 0;
        ad_buf[4] = ((unsigned short) ADDRA) >> 6;
        ・・・
        ad_round_flg = 1;
        ADCSR &= ~GROUP1;    /* グループ→0, (AD ストップしたまま) */
    }
}

int new_ad_data(void)
{
    int flg;
    if ((flg = ad_round_flg) == 1) ad_round_flg = 0;
    return flg;
}

int get_ad_data(unsigned char n, unsigned short buf[])
{
    int i;
    if (n >= ad_channels) {
        for (i=0; i<ad_channels; i++) buf[i] = ad_buf[i];
        for ( ; i<n; i++) buf[i] = 0;
    } else {
        for (i=0; i<n; i++) buf[i] = ad_buf[i];
    }
    ADCSR |= ADST;           /* AD スタート */
    return i;
}
```

# 付録 C

## nervous

## C.1 概説

### C.1.1 nervous とは

nervous とは，WS または PC がロボットのボディのプロセッサと通信するための C のプログラムである (nervous の名前の由来は神経系 (nervous system) である)．現在，Solaris, Solaris X86, Linux, Win32 で使用することができる．socket を通してさらに上位のプログラム (主に EusLisp) と通信することができる．ロボットのボディとの通信のインタフェースは，他の部分と分離されている．ここでは，筆者が nervous ライブラリのうち構築に関わってきた部分に関して解説する．

### C.1.2 nervous の make

make するには rbrapp というディレクトリ (環境変数 RBRAPPDIR をセットする) で，

```
% make ROBOT= ロボット名
```

とすると，メインのプログラムができる．

```
% setenv ROBOT ロボット名
```

としてもよい．通信のインタフェース部分は，“$RBRAPP/nervelib/ インタフェース名 /” というディレクトリで，

```
% make ROBOT= ロボット名
```

とする．(インタフェース名は serial, dummy, tp, sock, rtlinux, eus, ctrv などがある．) 実行ファイルなどのバイナリはすべて，“${RBRAPPDIR}/bin/ ロボット名 / アーキテクチャ名 /” の下に作られる．

### C.1.3 新しいロボットを作ったら

ディレクトリ ${RBRAPPDIR}/robot/ の下に，“ロボット名.h”，“ロボット名.c”，“ロボット名 model.c”，“ロボット名 InvDyn.c” というファイルを作る．さらに，オンボディの H8 と通信する場合は，“ロボット名 h8conf.h” というファイルも作る．EusLisp のモデルを持つロボットなら，EusLisp 上で “${RBRAPPDIR}/common/robot2c.l” をロードして，`(robot2c *robot-object*)` とすれば，これらのファイルが生成される．ロボットが，Cla なら，

```
% jskrbeus
1.jskrbeus$ (load (format nil ‘‘~A/common/robot2c.l’’
                          (unix:getenv ‘‘RBRAPPDIR’’)))
2.jskrbeus$ (setq robot (Cla))
3.jskrbeus$ (robot2c robot)
4.jskrbeus$ (exit)
% make ROBOT=Cla
```

### C.1.4 使い方

“${RBRAPPDIR}/bin/ ロボット名 /.nervousrc” というファイルをつくり，最初に読み込まれる命令を書く．最小限，次の一行があれば動く．

```
set-interface <interface 名>
```

<interface 名> は，serinerve，dummy など．そして，${RBRAPPDIR}/bin/ ロボット名 / アーキテクチャ名 / に移動し，

```
% ./nervous
```

とする．オプションの説明を見るには，

```
% ./nervous -help
```

とする．ヘルプを見るには，nervous> のプロンプトが出てから，? リターン とする．

```
% ./nervous
nervous> ?
```

## C.2 EusLisp と nervous の間の通信

### C.2.1 概要

nervous は， socket(と標準入力) からの入力を解釈し，一行単位で，

1. “#” で始まる → 何もせず
2. “(“で始まる → CommandEusNoRet() を実行
3. 最初の 1 単語は登録されているコマンドである → そのコマンドを実行

4. 最初の 1 単語は登録されているプラグインモジュール名である → そのプラグインモジュールがなんらかのアクションを実行 (メソッド)

5. 1. 〜 4. でない → シェルに渡す.

という処理をしている.

EusLisp からロボットを動かすときは, 3. か 4. を使う. つまり, nervous 内部のロボットボディに向かう値 (グローバル変数) を, 書き換えるプラグインを作り, 3. か 4. の方法で実行する.

```
10.jskrbeus$ (send *ns* :angle-vector #f(0 1 2))
```

とすると,

```
sequence propocmd #f(0 1 2) 1
```

という一行が socket に送られ, 4. により, `seqplugin` が, ロボット (3 自由度) の関節角度を, 1[ms] かけて, 0,1,2 にする.

### C.2.2 詳細

**EusLisp と nervous の通信**

EusLisp → nervous は, socket を通して通信している. nervous は, socket 経由でも, プロンプトから人が打ったコマンドでも, どちらも, `interpreter(char *line)` という関数が, それを解釈し実行する. 一つの nervous に二つ以上のソケットをつなぐ事もできるが (EusLisp が二つ等), 一つの接続毎にそこを`read` し`interpret` するスレッドが生成される. (`socket_reader()`)

plugin に対するコマンドも同じように interpreter を介して渡される.

```
11.jskrbeus$ (send *ns* :angle-vector #f(0 1 2 ... 23))
```

を実行すると, `:raw-propocmd` が呼ばれ, ソケット (スロット変数`strm`) に

```
(format strm "sequence propocmd ~A ~A~%" v time)
```

とされる. コマンドラインから,

```
nervous> sequence propocmd #f(0 1 2 ... 23) 1
```

等と打っても同じことがおきる．(最後の 1 はその姿勢まで 1[ms] で移行することを指示している．)

これは，“${RBRAPPDIR}/app/nerve/seqplugin.c” の propocmd() が実行されている．つまり，nervous において，seqplugin が走っていないと，

```
11.jskrbeus$ (send *ns* :angle-vector)
```

は実行できないということである．大抵の場合，“.nervousrc” に，

```
load-plugin seqplugin
create-plugin sequence sequence
start sequence
```

と書いてあり，これが，seqplugin をロードし走らせている．

デフォルトで nervous が受け付けることができるコマンド (load-plugin 等) の他に，以前は，add_command() という関数を使って，コマンドを追加する事が出来た．プラグインが独自のコマンドをadd_command していた．現在も add_command() は残っていてそれを使っているプラグインもあるが (servoplugin 等)，コマンドとして登録されていなくても，create-plugin した時につけた名前 (モジュール名) が入力されると，そのプラグインにコマンドを送るようになっている．

ロボットボディに送信するスレッドは，ロボットボディからデータが 1 セット受信される度に，そのグローバル変数の値を 1 セット送信している．

### seqplugin

EusLisp で，

```
11.jskrbeus$ (send *ns* :angle-vector #f(0 1 2))
```

とすると，

```
sequence propocmd #f(0 1 2) 1
```

という一行が送られ，4. により，seqplugin が，ロボット (3 自由度) の関節角度を，1[ms] かけて，0,1,2 にする．

つまり，“${RBRAPPDIR}/app/nerve/seqplugin.c” の関数propocmd() が実行される．

```
int propocmd(plugin *p, char *ptr)
{
    int fd=get_socket_fd();
    vector buf_v=MakeVector(DOF);
    double time;
    int overwrite=FALSE, withvel=FALSE;
    int i, options;

    ptr = read_vector(ptr, buf_v, DOF);
    ptr = read_token(ptr, "%lf", &time);
    options=check_arg(ptr);
    for (i=0;i<options;i++) {
      if (strncasecmp(ptr,":clear",6)==0) overwrite=TRUE;
      else if (strncasecmp(ptr,":with-velocity",6)==0) withvel=TRUE;
      else fprintf(stderr, "unknown option %s\n", ptr);
      ptr=skip_string(1,ptr);
    }

    if (overwrite) initSequence(Mem(seq),DOF);
    Mem(fds)[Mem(seq)->next]=fd;

    PushNext(Mem(seq), buf_v, time/1000.0); /* [ms]->[s] */

    DelVector(buf_v);

    return 0;
}
```

PushNext() は，躍度最小補間の姿勢の時系列の最後尾に，buf_v を付け加える．もっと low には，target_av[DOF] を書き換えれば，ボディに送信されるデータが変わる．(例：“${RBRAPPDIR}/app/hold.etc/pForce.c”)

姿勢の補間中に新たな目標が与えられた場合，通常はキューに追加され，前の補間が終了するのを待ってから，新たな姿勢へと移行する．以前のキューを破棄し新たな目標を与えたい場合には，overwrite というメソッドが実装されている． EusLisp から，

```
2.jskrbeusgl$ (send *ns* :angle-vector #f(0 1 2) :overwrite t)
```

とすると，

```
sequence propocmd #f(0 1 2) 1 :clear
```

という一行が送られ，propocmd() 内のinitSequence() によりキューが初期化され，PushNext() により新たな目標が与えられる．

補間方法は，躍度最小補間 (minjerk，minjerksequence) と線形補間 (linear) が実装されている．初期状態では躍度最小補間 (minjerk) で，速度 0 で始まり速度 0 で終るように補間をする．

姿勢列を再生したいような場合，補間方法がminjerk やlinear では各姿勢の境目で速度がゼロになるが，minjerksequence にすると，動作中だった場合 (その姿勢がキューの 2 番目

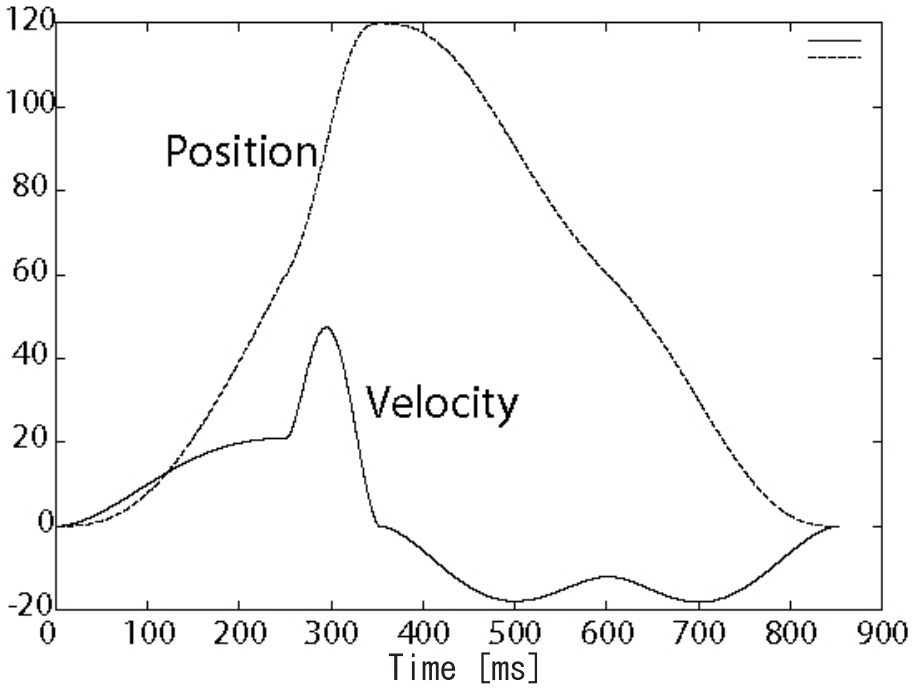



**Fig. C.1** minjerksequene での速度と位置の推移
Transition of the velocity and position with minjerksequence interpolation

以降になった時)，その初期速度を 0 でなく前のキューとバランスするような速度にする．時間 0 位置 0 ，時間 250 位置 60 ，時間 350 位置 120 ，時間 600 位置 60 ，時間 850 位置 0 という目標を一度に与え， minjerksequence で動作したときの位置・速度の推移を **Fig. C.1** に示す.

補間方法を切り替えるには， nervous のプロンプトで,

```
nervous> sequence switch newval
```

とするか， EusLisp から,

```
1.jskrbeus$ (send *ns* :interpolation-method newval)
```

とし，`newval` の値を`linear`，`minjerk`，`minjerksequence` で切り替える.

## C.3 ソースツリー

### C.3.1 ディレクトリ構成

ディレクトリ構成は，以下のようになっている.

```
$\{RBRAPPDIR\}/common/
           /conf/
           /include/
           /robot/

           /app/nerve/
           /app/march/
           /app/balance/
           /app/viewer/

           /nervlib/dummy/
           /nervlib/h8/
           /nervlib/serial/
           /nervlib/i2c/
```

### C.3.2 ${RBRAPPDIR}/common/

アーキテクチャやロボットの種類に依存しない共有ライブラリ.

**servo.c** サーボモータ関連.

**state.c** センサ情報構造体.

**vector.c** 浮動小数点ベクトル.

**arith.c** ベクトル演算・行列演算など.

**dynmodel.c** Newton-Euler 方による逆動力学計算.

**model.c** 剛体リンク構造のモデル計算.

**hoffarbib.c** 躍度最小補間.

**interpolator.c** スプライン補間.

**filter.c** バタワースフィルタ.

**sequencer.c** 姿勢列キュー処理.

**ADXL.c** 加速度センサ.

**FSR.c** 力センサ (FSR). 足裏反力センサ. 張力センサ.

**ForcePlate.c** 足裏反力センサなどの, 平板間力センサ.

**GYRO.c** ジャイロ.

**TACTILES.c** 触覚センサ. センサスーツ.

**interface.c** ロボットとの物理的通信インタフェースの抽象化.

**plugin.c** プラグイン.

**sockcom.c** ソケット通信.

**interpreter.c** ソケットおよびコマンドラインのインタプリタ.

**lispinterface.c** EusLisp とのインタフェース.

**euslayer.l** EusLisp とのインタフェース (lisp).

**ceus.c** EusLisp を起動しコマンドを実行.

**mtinterface.c** マルチスレッド.

**mthread_posix.c** POSIX マルチスレッドライブラリの Solaris スレッドへのラッパー.

**mthread_win32.c** Windows マルチスレッドライブラリの Solaris スレッドへのラッパー.

**robot2c.l** EusLisp のモデルからロボット定義ファイル生成.

### C.3.3 ${RBRAPPDIR}/conf/

アーキテクチャ別の make ファイル.

### C.3.4 ${RBRAPPDIR}/include/

アーキテクチャやロボットの種類に依存しない共有ライブラリのヘッダーファイル.

### C.3.5 ${RBRAPPDIR}/robot/

ロボット別の定義ファイル

**${ROBOT}.c** センサの変換パラメータやサーボの初期値等のセンサ・アクチュエータ情報.

**${ROBOT}.h** センサ・アクチュエータの種類・数・パラメータの初期値等.

**${ROBOT}model.c** ロボットの幾何モデル.

**${ROBOT}InvDyn.c** ロボットの逆動力学モデル.

**${ROBOT}h8conf.h** オンボディのセンサ・アクチュエータ情報の個数・配列の管理 (並べ替え等).

### C.3.6 ${RBRAPPDIR}/app/

**${RBRAPPDIR}/app/nerve/**

nervous 本体と各種プラグイン.

**commander.c** nervous のコマンドライン.

**flowplugin.c** 富士通カラートラッキングビジョン用プラグイン.

**logplugin.c** ログ保存に関するプラグイン.

**nerve.l** EusLisp とのインターフェイス (lisp).

**nervous_ex.c** プラグインの中で使うために export してある nervous の関数.

**nullplugin.c** テスト用プラグイン.

**robotview.c** EusLisp 上のロボットモデルの状態を nervous の情報に基づいて表示する. (viewer の昔の版)

**rotadjustplugin.c** 加速度センサによる姿勢制御.

**screen.c** nervous のターミナル制御用ルーチン.

**seqplugin.c** 姿勢補間用のプラグイン.

**servoplugin.c** サーボモータに関するプラグイン .

**smoothplugin.c** 急激な指令値の変化を滑らかにするプラグイン.

**swingplugin.c** ブランコの励振運動に関するプラグイン.

**util.c** ユーティリティ.

**${RBRAPPDIR}/app/march/**

足踏みに関するライブラリ.

**main.c** 足踏み制御用ルーチン.

**march.c** 足踏み制御に関する nervous 用プラグイン.

**${ROBOT}marchConfig.h** ロボットの足踏み姿勢定義.

**marchcalib.c** ${ROBOT}marchConfig.h を作成.

**${RBRAPPDIR}/app/balance/**

オートバランサー [96] に関するライブラリ.

**${RBRAPPDIR}/app/viewer/**

ロボットの状態を PC で表示するライブラリ.

**viewer.c** ロボットの状態を PC で表示する.

**viewerplugin.c** ロボットの状態を PC に表示する nervous 用プラグイン.

### C.3.7 ${RBRAPPDIR}/nervelib/

**${RBRAPPDIR}/nervelib/dummy/**

実機と通信しないで nervous の挙動を試すもの.

**${RBRAPPDIR}/nervelib/serial/**

シリアル通信とのインターフェイス (主に H8).

**${RBRAPPDIR}/nervelib/i2c/**

PC のパラレルポートからインテリジェントモータドライバと $I^2C$ 通信を行い，モータを制御する [103].

**${RBRAPPDIR}/nervelib/msmt/**

ロボット 1 機に対し，複数のシリアル通信を可能にしたもの.

**${RBRAPPDIR}/nervelib/rtlinux/**

RT-Linux で H5 を動かした.

**${RBRAPPDIR}/nervelib/tp/**

Transputer でリモートブレインの古いロボットを動かしていた.

**${RBRAPPDIR}/nervelib/jsim/**

シミュレーションとの通信.

# 付録 D

# VRML97 形式から EusLisp で読み込める形式への変換プログラム

## D.1 概要

ここでは，VRML97 形式 (*97.WRL) から EusLisp で読み込める形式 (*.tree) への変換プログラムのソースを掲載する．この JAVA プログラムは，研究室の OB の金広文男氏 (現在電子技術総合研究所) により書かれたものである．利用方法は，

```
% java VRML97toEus file97.WRL > file.tree
```

とする．以下に注意事項を示す．

- JAVA3D が必要． http://www.javasoft.com/products/java-media/3D/index.html
- VRML97 パーサが必要． http://www.web3d.org/TaskGroups/x3d/sun/cvs.html
- JDK1.2 / JRE が必要． http://www.sun.com/solaris/java/

## D.2 JAVA で記述されたプログラムソース

```
 1: // Java
 2: import java.io.*;
 3: import java.util.*;
 4: import java.awt.*;
 5: import java.awt.event.*;
 6:
 7: // Java3D
 8: import javax.media.j3d.*;
 9: import javax.vecmath.*;
10: import com.sun.j3d.utils.behaviors.mouse.*;
11: import com.sun.j3d.utils.behaviors.picking.*;
12: import com.sun.j3d.utils.geometry.*;
13: import com.sun.j3d.utils.universe.SimpleUniverse;
14:
15: // VRML
16: import com.sun.j3d.loaders.vrml97.*;
17:
18: public class VRML97toEus{
19:   private static String wrlfile;
20:   private void _traverse(Node node, int depth) {
21:
22:     int GROUP = 0;
23:     int BRANCHGROUP = 1;
24:     int TRANSFORMGROUP = 2;
25:
```

```
26:     // Is the node Group or a subclass of Group ?
27:     if (node instanceof Group) {
28:       int nodetype = GROUP;
29:       try {
30:         BranchGroup bg = (BranchGroup)node;
31:         nodetype = BRANCHGROUP;
32:       } catch (ClassCastException exbg) {
33:         try {
34:           TransformGroup tg = (TransformGroup)node;
35:           nodetype = TRANSFORMGROUP;
36:         } catch (ClassCastException extg) {
37:         }
38:       }
39:
40:       // Is the node Group ?
41:       if (nodetype == GROUP){
42:         Group g = (Group)node;
43:         g.setCapability(Group.ALLOW_CHILDREN_READ);
44:         g.setCapability(Group.ALLOW_CHILDREN_WRITE);
45:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
46:         //System.out.println("----------");
47:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
48:         //System.out.println("Group");
49:         for (int i = 0; i < g.numChildren(); i++) {
50:           _traverse(g.getChild(i), depth + 1);
51:         }
52:       }
53:
54:       // Is the node BranchGroup ?
55:       else if (nodetype == BRANCHGROUP) {
56:         BranchGroup bg = (BranchGroup)node;
57:         bg.setCapability(BranchGroup.ALLOW_CHILDREN_READ);
58:         bg.setCapability(BranchGroup.ALLOW_CHILDREN_WRITE);
59:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
60:         //System.out.println("----------");
61:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
62:         //System.out.println("depth = " + depth);
63:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
64:         //System.out.println("BranchGroup ");
65:         for (int i = 0; i < bg.numChildren(); i++) {
66:           _traverse(bg.getChild(i), depth + 1);
67:         }
68:       }
69:
```

```
 70:       // Is the node TransformGroup ?
 71:       else if (nodetype == TRANSFORMGROUP) {
 72:         TransformGroup tg = (TransformGroup)node;
 73:         tg.setCapability(TransformGroup.ALLOW_CHILDREN_READ);
 74:         tg.setCapability(TransformGroup.ALLOW_CHILDREN_WRITE);
 75:         tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
 76:         tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
 77:         //for (int i = 0; i < depth; i++) { System.out.print("  "); }
 78:         //System.out.println("TransformGroup");
 79:         Transform3D t3=new Transform3D();
 80:         tg.getTransform(t3);
 81:         Matrix3f m3=new Matrix3f();
 82:         Vector3f v3=new Vector3f();
 83:         t3.get(m3,v3);
 84:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
 85:         System.out.println("((:pos #f("+v3.x+" "+v3.y+" "+v3.z+"))");
 86:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
 87:         System.out.println("(:rot #2f(("+m3.m00+" "+m3.m01+" "+m3.m02+")("
 88:                             +m3.m10+" "+m3.m11+" "+m3.m12+")("
 89:                             +m3.m20+" "+m3.m21+" "+m3.m22+")))");
 90:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
 91:         System.out.println("(:descendants (");
 92:         for (int i = 0; i < tg.numChildren(); i++) {
 93:           _traverse(tg.getChild(i), depth + 1);
 94:         }
 95:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
 96:         System.out.println(")");
 97:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
 98:         System.out.println("))");
 99:       }
100:
101:     }
102:
103:     // Is the node Link ?
104:     else if (node instanceof Link) {
105:       Link l = (Link)node;
106:       SharedGroup sg = l.getSharedGroup();
107:       //for (int i = 0; i < depth; i++) { System.out.print("  "); }
108:       //System.out.println("Link");
109:       for (int i = 0; i < sg.numChildren(); i++) {
110:         _traverse(sg.getChild(i), depth + 1);
111:       }
112:     }
113:
```

```
114:     // Is the node Shape3D ?
115:     else if (node instanceof Shape3D) {
116:       //for (int i = 0; i < depth; i++) { System.out.print("  "); }
117:       Shape3D sh = (Shape3D)node;
118:       sh.setCapability(Shape3D.ALLOW_GEOMETRY_READ);
119:       sh.setCapability(Shape3D.ALLOW_APPEARANCE_READ);
120:       Geometry gm = sh.getGeometry();
121:       if (gm instanceof TriangleArray){
122:         TriangleArray ta = (TriangleArray)gm;
123:         ta.setCapability(TriangleArray.ALLOW_COUNT_READ);
124:         ta.setCapability(TriangleArray.ALLOW_COORDINATE_READ);
125:         Point3f p3=new Point3f();
126:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
127:         System.out.println("((:shape ((");
128:         for (int i = 0; i<ta.getVertexCount(); i++){
129:           ta.getCoordinate(i, p3);
130:           for (int j = 0; j < depth; j++) { System.out.print("  "); }
131:           System.out.println("#f(" + p3.x+" "+p3.y+" "+p3.z+")");
132:         }
133:         for (int i = 0; i < depth; i++) { System.out.print("  "); }
134:         System.out.println(")");
135:       }
136:       Appearance ap = sh.getAppearance();
137:       Material mt = ap.getMaterial();
138:       Color3f diff = new Color3f();
139:       mt.getDiffuseColor(diff);
140:       for (int i = 0; i < depth; i++) { System.out.print("  "); }
141:       System.out.println("#f("+diff.x+" "+diff.y+" "+diff.z+"))))");
142:     }
143:
144:     // else
145:     else {
146: //       System.out.println("* The node " + node.toString() +
147: //                          " is not supported.");
148:     }
149:
150:   }
151:   public VRML97toEus(){
152:     VrmlLoader loader = new VrmlLoader();
153:     VrmlScene scene = null;
154:     try{
155:       scene = (VrmlScene)loader.load(wrlfile);
156:     }catch(java.io.FileNotFoundException ex){
157:       System.out.println("* FileNotFoundException occurred");
158:     }
159:     BranchGroup bg = scene.getSceneGroup();
160:     Hashtable ht = scene.getNamedObjects();
161:     _traverse(bg,0);
162:   }
```

```
163:   public static void main(String argv[]){
164:     if (argv.length == 0){
165:       System.out.println("Usage: java VRML97toEus <wrlfile>");
166:       System.exit(0);
167:     }
168:     System.out.println(";;"+argv[0]);
169:     wrlfile = argv[0];
170:     VRML97toEus ts = new VRML97toEus();
171:   }
172: }
173:
```

**著者紹介**

水内 郁夫 (みずうち いくお)
1972 年 6 月 1 日 東京都に生まれる．早稲田大学理工学部機械工学科，東京大学大学院工学系研究科機械情報工学専攻修士課程を経て， 2001 年 12 月東京大学大学院工学系研究科機械情報工学専攻博士課程単位取得の上退学． 2000 年 1 月より 2001 年 12 月，日本学術振興会特別研究員． 2002 年 1 月より東京大学大学院情報理工学系研究科 科学振興特任教員． 2001 年 9 月「多自由度脊柱を持つ全身行動体のための体幹の腱駆動拮抗制御」により，日本ロボット学会第 16 回研究奨励賞を日本ロボット学会より受賞．日本ロボット学会学生会員．元日本ロボット学会学生編集委員．研究の興味は，ロボットの知能，柔軟なロボット，人間型ロボット，ロボットシステム構成法など．人間や生物の知能に，異るアーキテクチャを持つ機械でどこまで迫ることができるかを追求することをライフワークとしたいと考えている．

博士論文: 「柔軟性可変な脊椎構造を有する多自由度全身行動ロボットシステム」

2001 年 12 月　　水内 郁夫