二脚ロボットを用いた人のSplit-Belt Treadmill歩容適応モデルの提案

音　田　裕　史

電気通信大学

2009 年 3 月

二脚ロボットを用いた
人の Split-Belt Treadmill 歩容適応モデルの提案

音　田　裕　史

電気通信大学大学院情報システム学研究科

博士（工学）の学位申請論文

2009 年 3 月

二脚ロボットを用いた

人の Split-Belt Treadmill 歩容適応モデルの提案

博士論文審査委員会

主査　高瀬　國克　教授

委員　田中　健次　教授

委員　出澤　正徳　教授

委員　阪口　豊　　准教授

委員　明　　愛国　准教授

著作権所有者

音田　裕史

2009

Proposal of Gait Adaptation Model in Human Split-Belt Treadmill Walking Using a 2D Biped robot

Yuji Otoda

Abstract

Recently, there have been several trials that use robotics as a tool for neuroscience, especially in locomotion studies. In the case of bipedal locomotion, human walking has been investigated extensively over several decades.

A number of studies have measured kinematics, dynamics, and oxygen uptake while a person walks on a treadmill. In particular, during walking on a split-belt treadmill, in which the left and right belts have different speeds, remarkable differences in kinematics are observed between normal subjects and subjects with cerebellar disease.

In order to understand mechanisms behind such phenomena, it is useful to construct the control model of human walking, simulate it using a musculoskeletal model and compare the simulation results with the results of human experiments. But since it is difficult to simulate friction, collision with ground, effects of elastic materials and so on, we would like to carry out experiments using a real machine (robot) rather than computer simulations.

In order to construct a gait adaptation model of such human split-belt treadmill walking, we proposed a simple control model and developed a new 2D biped robot walk on a split-belt treadmill. We combined the conventional limit-cycle based control consisting of joint PD-control, cyclic motion trajectory planning, and a stepping reflex with a newly proposed adjustment of P-gain at the hip joint of the stance leg.

The data obtained in experiments on robot (normal subject model and cerebellum disease subject model) have highly similar ratios and patterns to

data obtained in experiments on normal subjects and subjects with cerebellar disease carried out by Bastian et al. We also showed that the P-gain at the hip joint of the stance leg was the control parameter of adaptation for symmetric gaits in split-belt walking and that P-gain adjustment corresponded to muscle stiffness adjustment by the cerebellum.

Consequently, we successfully proposed a gait adaptation model for human Split-belt treadmill walking and confirmed the validity of our hypotheses and the proposed model using the biped robot.

二脚ロボットを用いた
人の Split-Belt Treadmill 歩容適応モデルの提案

音　田　裕　史

概要

近年，神経生理学における研究のツールとしてロボットを用いる試みがあり，その中で歩行に関する研究がいくつか行われている．二脚歩行に関しては，数十年にわたり人の歩行についての調査が数多くなされてきており，人のトレッドミル歩行における運動学，動力学，代謝研究などが数多く報告されている．本研究では，左右のベルトの速度が異なる split-belt 型のトレッドミルでの歩行パターンの適応現象に注目する．そこでは健常者と小脳疾患患者のあいだに運動学的パターンに大きな相違が現れ，この現象の背後にあるメカニズムを解明することが課題となっている．

このような問題の研究においては，従来，筋骨格系モデルを用いたシミュレーションを行いその結果を人の歩行実験結果と比べることが行われてきた．このアプローチは簡便であるが，地面との摩擦や衝突・伸縮素材の影響などを完全にモデル化しシミュレーションに反映することは困難である．本研究では，二次元二脚歩行ロボット「鉄郎」を開発し，物理的な人の歩行モデルを構成することで，split-belt 上での歩行パターンの適応現象を検証する．

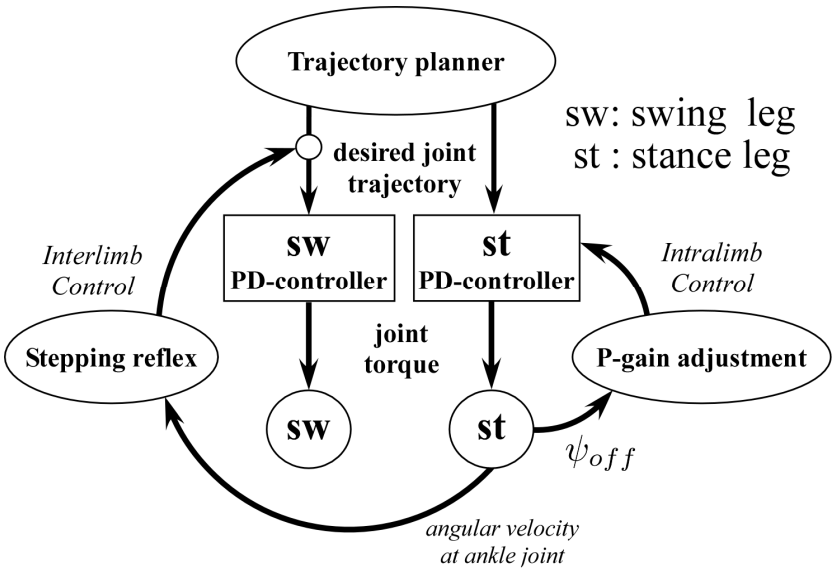
本論文では，最初に健常者と小脳疾患患者の split-belt treadmill 歩行の実験結果を紹介する．次に，関節での PD 制御・脊髄における周期的な運動生成を規範とした倒立振子に基づく軌道計画・脳幹における制御を規範とした遊脚着地角制御（ステッピングリフレックス）から成る従来のリミットサイクルを構成する制御手法について述べる．また，split-belt treadmill 上件での自律的な適応歩行を実現させるために鉄郎に導入した，小脳における運動調節を規範とした支持脚腰関節における P ゲイン調節を提案する．最後に，鉄郎と人(健常者と小脳疾患患者)の歩行パターンを比較し，提案した制御モデルの正当性について議論を行う．測定された指標の比率と歩行パターンの高い類似性は，筆者の

提案した仮説・モデルが正当であることを示唆していると考える．

人の split-belt treadmill 歩行実験を行った Bastian らは，脊髄・脳幹・小脳・運動皮質を含んだ神経構造がさまざまな運動適応の制御を担っていると示唆している．しかしながら，どの神経構造がどのような種類の調節メカニズムによりどの適応機構に貢献しているかは明確に知られていないと言及した．そこで，筆者は，split-belt treadmill 上の二脚歩行に対する調節メカニズムを提案し，健常者や小脳疾患患者の歩容適応モデルを構成し，2 次元二脚歩行ロボットを用いて構成したモデルの正当性を検証した．ロボットの実験で得られたデータが人の実験で得られたデータに近いパターンを示したので，筆者の構成したモデルが二脚歩行における人の神経構造と類似している可能性を示している．

Bastian らは，以下の二つの調節機構が人の split-belt treadmill 歩行において存在すると示唆している．

(a) 脊髄や脳幹における感覚的フィードバック適応機構(ストライド長やデューティ比を調節する intralimb coordination に相当).
(b) 小脳における予見的フィードフォワード適応機構(ステップ長や両脚支持期間比の差を調節する interlimb coordination に相当).



歩容適応モデル図

それに対して，筆者の提案したモデルにおける適応機構(前頁のモデル図参照)は，Bastian らが提案した調節機構と異なり以下のように要約される．

[a] デューティ比はおおむね運動学的拘束により受動的に調節される．
[b] ステッピングリフレックスは interlimb コントロールであるにもかかわらず，intralimb index(ストライド長)を調節する(脳幹における感覚的フィードバック適応機構に相当)．
[c] P ゲイン調節は intralimb コントロールであるにもかかわらず，ステッピングリフレックス(interlimb コントロール) と組み合わさり interlimb indexes(ステップ長と両脚支持期間比の差)を調節する(小脳における感覚的フィードバック適応機構に相当)．

筆者は神経生理学における知見を基に人の split-belt treadmill 歩容適応モデルを構成し，2 次元二脚歩行ロボット「鉄郎」を手段として用い，モデルの正当性を歩行実験により検証した．構成した歩容適応モデルと Bastian らによって行われた健常者や小脳疾患患者における実験結果に比率とキネマティクスパターンにおいて高い類似性が見られた．これは，構成したモデルが正当であることをほのめかしている．

また，支持脚腰関節 P ゲインが split-belt treadmill 歩行における対称性を有する歩容適応の制御パラメータであることと，P ゲイン調節は小脳による physic な筋肉の剛性調節に相当することを示した．したがって，筆者は人の split-belt treadmill 歩行における歩容適応モデルを提案し，二脚ロボットを用いて構成した仮説とモデルの正当性を確認した．

目 次

表 目 次

図 目 次

第1章　序論

1.1　はじめに

近年，神経生理学における研究のツールとしてロボットを用いた試みがある[Schaal:2008]．これらの研究は脳や神経細胞(ニューロン)のメカニズムを解明するために行われている．その試みの中に歩行に関する研究がいくつか行われている．Ijspeertらは脊髄モデルを構成し開発したサンショウウオ型ロボットを用いて遊泳から安定な四脚歩行を実現している[Ijspeert:2007, Ijspeert:2008]．Maufroyらはネコの神経モデルと筋骨格系を用いたシミュレーションを行っている[Maufroy:2008]．二脚歩行の場合，数十年にわたり人の歩行について集中的に調査されてきた．江原らは人の直立状態や歩き始め，平地定常歩行などを力学的に解析している[江原:2006]．人のトレッドミル歩行における運動学，動力学，酸素吸収量などが数多く報告されている[Reisman:2005, Morton:2006, Giese:2007]．高齢者の転倒防止に重要な姿勢制御方式の調査や外乱に対する安定性の解析が行われている[Rogers:2003, Spek:1999]．これらの研究はリハビリテーション分野への応用のために行われている．特に，左右のベルトの速度が異なるsplit-belt条件での歩行パターンの適応を調べると，健常者と小脳疾患患者では運動学的パターンに大きな相違が現れる[Morton:2006]．

人の歩行実験において，身体内でどのような制御が行われているかは，外部観測値からの類推に頼らざるを得ない．この現象の裏にあるメカニズムを解明するためには，筋骨格系モデルを用いたシミュレーションを行いその結果を人の歩行実験結果と比べることが実際的である．しかし地面との摩擦や衝突・伸縮素材の影響などをシミュレートすることが困難であり，かつこれらのダイナミクスは歩行において非常に重要である．そこで我々は，実機を用いて具体的な手法を提案・実現することに意義があると考えた．筆者の知るかぎりsplit-belt条件での二脚歩容適応モデルの構成を行っている他の研究グループは現時点では見当たらないので，本研究が初の試みである．二脚走行に関しては，Bastianら[Bastian:2008]がSLIPモデルを用いてsplit-belt treadmill上のモデルを構成しようと試みている．四脚歩行(ネコ)のsplit-belt条件での適応モデルの構成に関しては，柳原らやGrillnerらなどにより数

多く行われているが，実機を用いた研究では柳原ら[柳原:1999]があげられる．彼らは，ネコの歩行実験から得られた知見より構成した数理モデルをロボットに実装し，歩行適応を実現している．ただ，この研究はロボットの胴体を固定しており姿勢制御を考慮していない．それに対して本研究では，2次元平面内の姿勢制御を考慮した二脚歩容適応モデルを提案する．

本研究では機構的に工夫した胴体のない2次元二脚歩行ロボット「鉄郎」を開発し，人の歩容適応モデル(Figure 2.8参照)を構成し，そのモデルがどのように働いているか検証する．また小脳疾患患者の歩行障害モデルを構成することができれば，将来的にリハビリや歩行介助装置への応用が期待できる．

1.2 従来の二脚歩行ロボット制御手法

80年代から90年代中期にかけて二脚ロボットは盛んに研究された[Miura:1984, 高西:1985, 佐野:1989, 梶田:1996]．90年代中期にホンダのASIMOやソニーのQRIOなどのヒューマノイドロボットが開発され[山口:1996, Hirai:1998, 梶田:2004, 西脇:2007]，二脚ロボットが一般的に知られるようになり，関心はますます強まってきた．現在これらに代表される多くの二脚ロボットでは，ZMP(Zero Moment Point) を指標として生成した軌道に追従させる制御手法(ZMP規範型)を用いている．現在これらのロボットは，様々な機能が加えられ機構もより複雑化してきた．歩行という観点から見ると，整地での歩行・走行，階段昇降，限られた環境での不整地歩行などは実現できていると同時に，実社会に見られる複雑な環境への適応は，機構や制御手法などの理由により非常に困難である．同時に各ステップにおいて全体質量をアクチュエータのみで加減速しており，生成される運動のエネルギ効率がよいとはいえない．また，制御が複雑であることも難点である．

それに対して，機構の特性や重力を有効利用し，できる限りアクチュエータのトルク出力を小さくすることでエネルギ効率の良い歩行を実現する制御手法がある．この制御手法により生成される歩行は「ナチュラルダイナミクスを利用した歩行」と呼ばれている．その代表的な例は，アクチュエータを持たない機構が坂を下る受動歩行である[Collins:2001]．機構の持つダイナミクスを活用できる二脚ロボットは，簡潔な制御で効率的な歩行を実現することに適している．

ロボットが二脚歩行を行うための制御手法として，①幾何的に系が転倒しない軌道を生成しその軌道を追従させる手法と，②制御，機構，環境を全て含んだ系によりリミットサ

イクルを構成する手法がある．様々な環境に適応するためには後者の制御手法が適していると我々は考える．リミットサイクルを構成すると同時に効率的で適応的な歩行を実現するために，リズム生成器や身体との引き込み機能を用いた研究がいくつかある[Taga:1991, 國吉:2003, Endo:2004, 矢野:2005, Aoi:2005, Geng:2006]．

現存する二脚ロボットの制御手法は，ZMP規範型制御手法とリミットサイクルを構成する手法におよそ集約される(Table 1.1参照)．人はZMP規範型制御手法もリミットサイクルを構成する手法も用いることができる．しかし定常歩行において，人はナチュラルダイナミクスを活用するためにリミットサイクルを構成する手法を用いているといわれている[Miura:1984]．ゆえに我々は，ZMP規範型制御手法でなくリミットサイクルを構成する手法を用いて，人のsplit-belt treadmill歩容適応モデルの構成を行った．これら2種類の二脚歩行制御手法を以下に紹介する．

Table 1.1: General classification of bipedal walking control method.

制御系指向	力学系指向
ZMP	Limit Cycle
軌道計画と軌道追従 計画と実行の分離 明示的モデル	機構・運動パターン設計 計画と実行の一体化 非明示的モデル

1.2.1 ZMP規範型制御手法

現在ASIMOやQRIOに代表される多くの二脚ロボット(Figure 1.1,Figure 1.2参照)では，ZMP(Zero Moment Point) を指標として生成した軌道に追従させる制御手法(ZMP規範型)を用いている．ZMPを指標とする制御手法を用いる一般的な二脚ロボットを力学モデルで表すと，Figure 1.3のようになる．ここで，運動は2次元平面内としロボットの質量は胴体中心に集中していると仮定する．歩行中ロボットに加速度aが生じている時，質量mにかかる慣性力と重力の合力により発生するモーメントが床面(x軸)上でゼロとなる点をZMP(zero moment point)という．このZMPがfootの中にあれば，足底は浮き上がることなく転倒しないので安定である．そこで，ZMPがA(かかと)→B(つま先)に移動するようなZMPの目標軌道を立て(Figure 1.4-(a)参照)，それにロボットの実際のZMPを追従させるような各関節の目標軌道を生成･追従させれば安定な歩行を実現できる．この時，目標ZMPに床反力中心点を一致させるために，慣性の大きな体を足首関節(C点)に設置されたアクチュエータのトル

クによって移動させなければならないので，足首関節には大きなモータを必要とする．また，ZMPを一脚支持のfootに確実に確保するために，とりわけZMPを指標とする制御を行う二脚歩行ではfootは大きくなければならない(Figure 1.4-(a)(b)参照)．footが重くなると脚の慣性モーメントが大きくなるために,腰関節の減速比を大きくして大きなトルクを発生させる必要がある．しかし，そうすると逆に関節角速度は小さくなるため素早い動きで目標軌道に追従させることが困難になる．以上の点から素早い動きを必要とする動歩行には，機構的に向いていない．また，不整地環境においての床反力中心点の検出，とりわけ不整地形状が踏むことによって変わってしまうような柔かい路面においての検出は大変困難であるため，不整地歩行にも向いていないと考えられる．その他に，ZMPをfootに保っておくには常に膝を曲げている必要があり，膝を伸ばすと特異点という制御が出来ない状態になってしまうので，人間のような膝を伸ばした歩行は非常に難しいなどの問題もある．

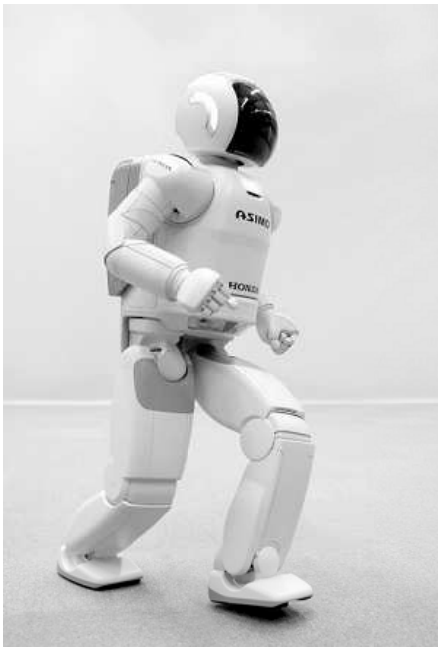


Figure 1.1: Photo of ASIMO.

Figure 1.2: Photo of QRIO.

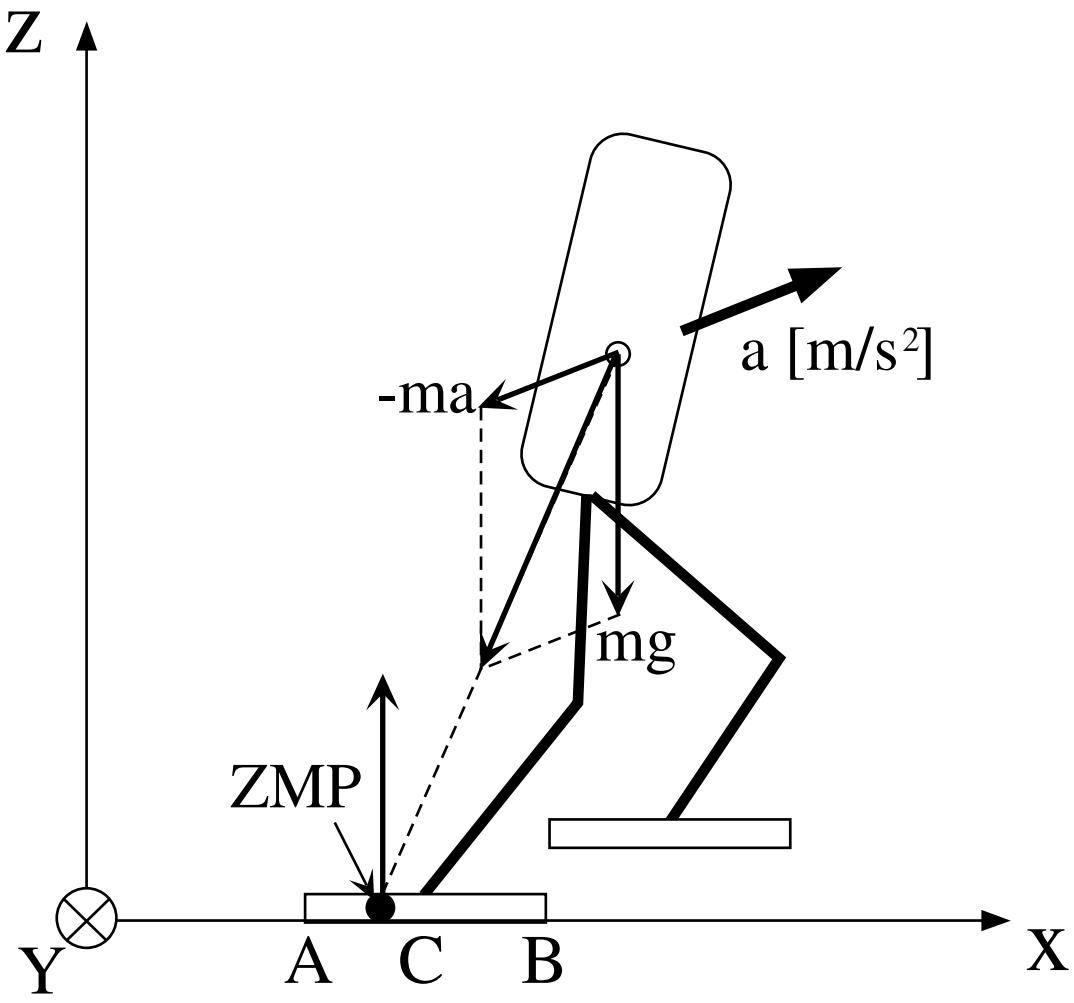


Figure 1.3: Dynamical model of ZMP based control method.

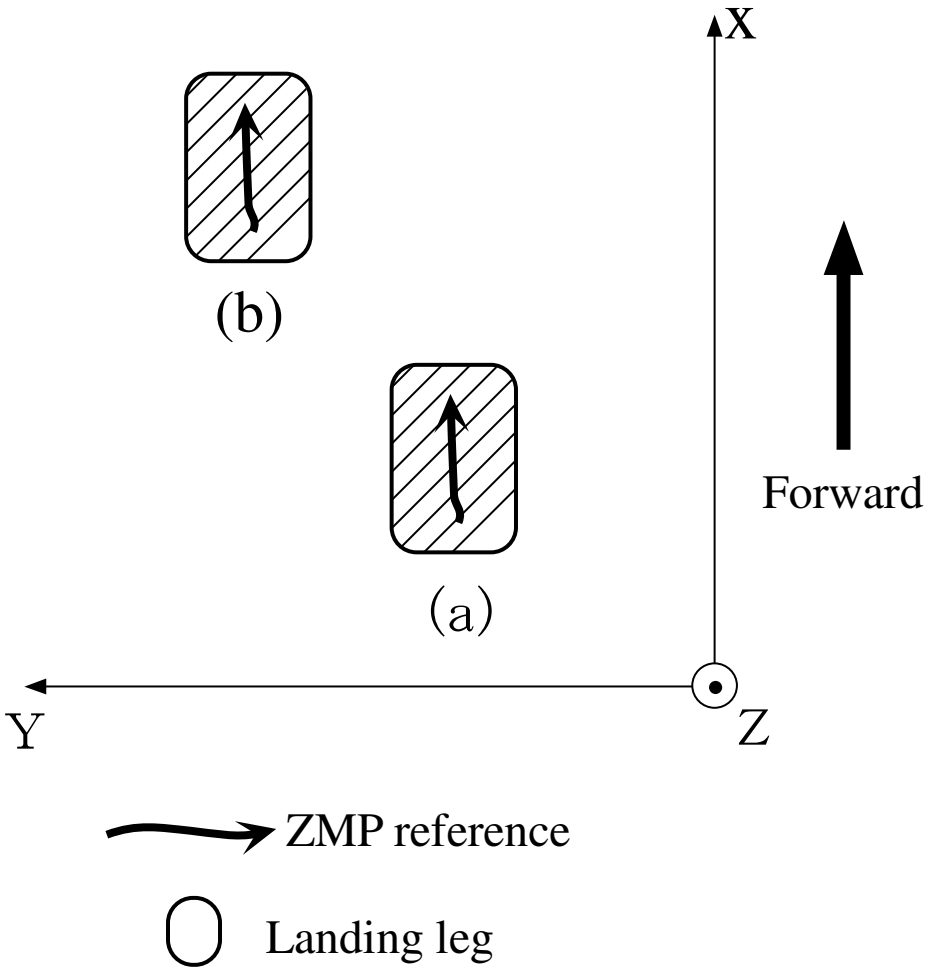


Figure 1.4: Setting of ZMP reference.

ZMPと足首関節トルク出力の関係

Figure 1.5はfootにおける力の分布例である．荷重は接触面全体で同符号なので，足部の境界の内側に存在する点に作用する等価な力Rとしてまとめられる．力ベクトルRが通過するこの作用点をZMPと呼ぶ．ZMPは全床反力の中心点(Center of Pressure; COP)そのものとして定義されている[Vukobratović:1972]．接触面に隣接する足首関節トルクによって

COPを積極的に操作することが可能である．**5.1** 節で，COPを操作する足首関節トルク出力に注目しZMP制御手法への結果としての依存度に関して考察を行った．

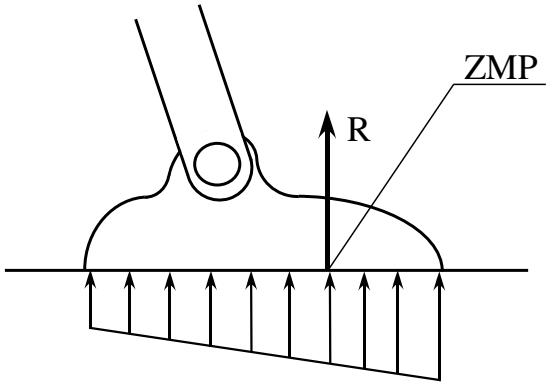


Figure 1.5: Definition of ZMP.

1.2.2　リミットサイクルを構成する制御手法

倒立振子型制御手法

着地しているfootが作る多角形上にZMPを保つことなく歩行が行われる系を，倒立振子モデルとして考えることができる．歩行における倒立振子モデルはFigure 1.3においてx軸上で$A = B = C$となる，つまり支持脚相で点Cにおいて点接地し，x軸歩行に倒れようとする倒立振子となり，そのままの状態で転倒を防ぐためにもう一方の脚を振り出して胴体を支えるという運動を繰り返し行う歩行をいう．この考えに基づいた制御手法を倒立振子型制御手法と言う．この倒立振子状態である時，支持脚足首関節におけるトルク出力はゼロなのでZMPは常に点Cに存在する．この状態は歩行中ほぼ全域において不安定である．しかし，この不安定な状態を支持脚と遊脚の交換を繰り返すことでリミットサイクルを構成し安定化している．また，この制御手法では支持脚足首関節とZMPの安定領域を広げる大きなfootが不要なので，膝下を軽くすることができる．結果として脚の慣性モーメントが小さくなり腰関節モータの減速比を小さく設定できる．これはZMP規範型の二脚ロボットとは対照的である．従って，倒立振子規範型の二脚ロボットは，ZMPを指標とする制御モデルで挙げた，遊脚時の関節目標軌道の追従制御遅れとして最も大きな要因と考えられる機構的な制御遅れの問題から解消されるため，ロボットは素早く脚を動かすことができ，高速歩行に適している．本研究ではこの制御手法を用いた．具体的には，倒立振子モデルを用いて軌道生成を行った(**4.1.3**　節参照)．

下山ら[下山:1981]は支持脚接地点まわりの角速度によって一方の遊脚の接地角度を制御する竹馬型二脚歩行ロボット「Biper」を開発している．Biperの全体画像をFigure 1.6に示す．Raibertら[Raibert:1986]は，直動関節の圧縮度による高さの制御，支持脚収縮時に胴体が地面に対して水平になるようにする胴体制御，および，遊脚着地位置を変化させることによるスピード，安定化制御を用い，1，2，および4脚ロボットの走行(ホッピング)に成功している．その二脚ホッピングロボットの全体画像をFigure 1.7に示す．Wisseら[Wisse:2008]は，歩行速度を変えて目標歩行速度を維持するリミットサイクル型の二脚ロボットを開発している．その二脚ロボットの全体画像をFigure 1.8に示す．加藤ら[加藤:1979]は，ピッチ平面内に拘束され，脚伸縮機構を持つ二脚歩行系に対して，リミットサイクルの持つ安定性を利用した制御系を提案した．古荘ら[古荘:2002]は，ピッチ平面内の足首関節にアクチュエータのない系に対して局所的なフィードバックを行うことにより二脚歩行を実現している．また，「蹴り」動作のために足首関節トルクを出力する場合についても報告している．

他，蹴り動作に関連し[Hodgins:1991, 梶田:2005]などがある．

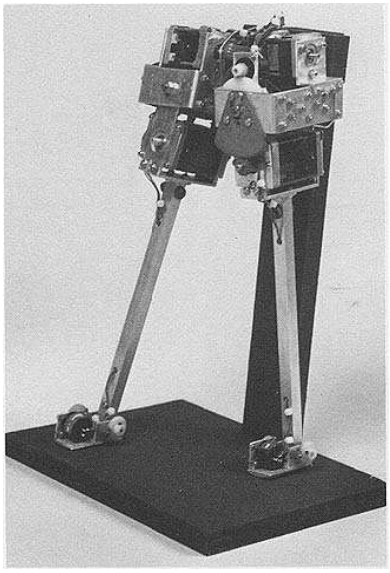


Figure 1.6: Photo of Biper 3.

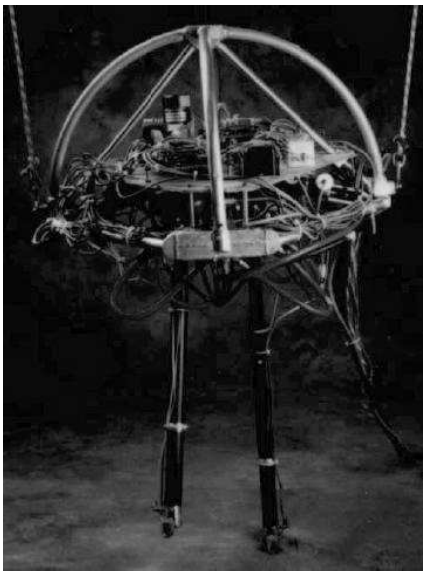


Figure 1.7: Photo of bipedal hopping robot of Raibert.

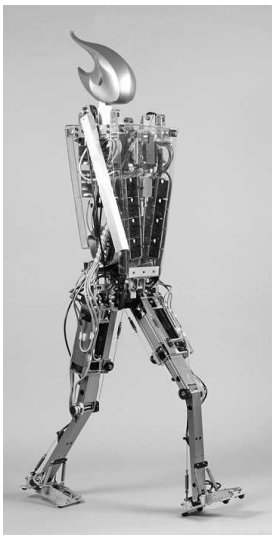


Figure 1.8: Photo of Wisse's 3D bipedal walking robot "Flame".

神経生理学のアプローチで人の歩行や姿勢が環境に対してどのように適応するのかを解明しようとする研究がなされている[Takakusaki:2008]．多賀[多賀:1997]が，生物の脊髄に存在する歩行リズム生成器CPGの数式モデル[Matsuoka:1987]を用いて二脚歩行シミュレーションを成功させて以来，多くの研究者がこのような非線形振動子を用いた創発的歩行生成に取り組むようになった．この原理は非線形振動子の活動リズム，筋骨格系の運動リズム，環境との相互作用のリズムの間にグローバルな「引き込み現象」を発生させることによってアトラクタを形成する[土屋:2005]．その結果として安定で柔軟な歩行運動がリアルタイムに生成される．多賀はこのような原理に基づいたピッチ平面内での二脚歩行シミュレーションを行っている[多賀:1997, Taga:1991, Taga:1995a, Taga:1995b]．このような創発的歩行生成・制御モデルは広い安定領域を有しており，ある程度の外乱や不整地に対しても安定した歩行パターンを生成させる．矢野らは独自に開発したKYS振動子と呼ばれる非線形振動子を用いてピッチ面内での二脚歩行シミュレーションを成功させている[矢野:2003]．その振動子は多賀モデルCPGと比較して引き込み領域が広く，広範囲での周波数調節が可能であると主張している．非線形振動子を用いて歩行を行う実機としては四脚ロボット[福岡:2003, 土屋:2002] が代表的である．二脚では青井ら[Aoi:2005]が実機で3次元平面内，遠藤ら[遠藤:2004]やGengら[Geng:2006]が実機で2次元平面内の歩行を実現させている．遠藤ら，Gengらの二脚ロボットの全体画像をそれぞれFigure 1.9，Figure 1.10に示す．

Passive Dynamic Walkingに基づく制御手法

Passive Dynamic Walking(以後PDW)は，「ナチュラルダイナミクス」に基づいてMcGeer[McGeer:1990]により提案された歩行である．PDWは，機構の持つダイナミクスと位置エネルギを活用することで，エネルギ注入なしで滑らかな下り坂を動的に下る歩行をいう．McGeerは，アクチュエータを持たず膝と腰にpassiveな関節を備えた二脚機構を用いて，制御入力なしで緩い斜面での2次元歩行に成功した[McGeer:1990]．Collinsら[Collins:2001]はそれに上半身を持つ3次元モデルを製作した．そこでは，腕を振ることによってyaw軸回りの回転を小さく押え，より視覚的に人間の歩行に近い歩行を実現している．当然，制御していないので環境の変化に対応することはできないが，歩行は身体と環境とが力学的に相互作用を起こし，引き込み合うことによって発生する非線形振動であることを再確認するに十分な成果で，多くの研究者に影響を及ぼした．Collinsらの二脚ロボットをFigure 1.11に

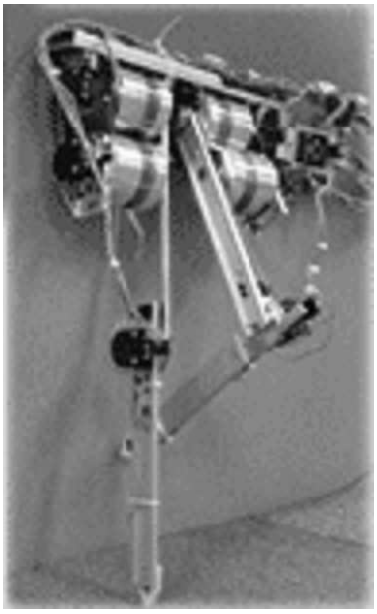


Figure 1.9: Photo of Endo's bipedal walking robot.

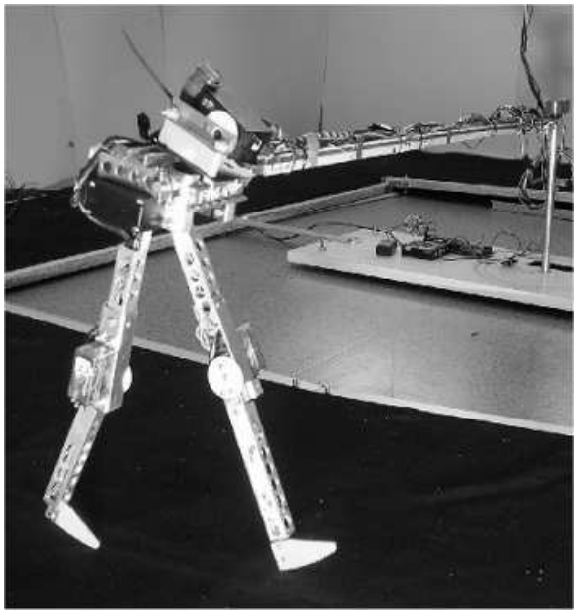


Figure 1.10: Photo of Geng's bipedal walking robot.

示す．機構の持つダイナミクスを活用できる二脚ロボットは，効率的な歩行を実現することに適している[木村:2003]．受動歩行のナチュラルダイナミクスを参考にした二脚歩行制御の研究として，浅野ら[浅野:2004]は目標エネルギ軌道追従制御を，大須賀ら[大須賀:2004]は遅延フィードバック制御を提案している．また，Wisseら[Wisse:2003]や細田ら[細田:2005]は，空気圧人工筋を用いたピッチ面内自立型二脚ロボットにおいて，受動歩行を参考にした効率の良い歩行を実現している．特に田熊ら[Takuma:2006]や辻田ら[Tsujita:2007]は，トレッドミル上でバルブのon/offのタイミングを調節し関節の剛性を変えることにより歩行速度を変化させた．加えてChemori[Chemori:2004]らは，ポアンカレ断面を用いて実現した歩行の安定性を解析している．小野ら[小野:1994, Ono:2004]は腰にアクチュエータを備えた機構で，センサ情報を用いてトルク出力を行う「Self-Excited Walking」をシミュレーションで検証し，実機で実現している．小野らやWisseらの機構は，McGeerが開発した歩行機構と非常に似た構造をしており，三本の脚を二脚のように動作させることで，2次元平面内での二脚歩行を実現している．小野らの用いた二脚機構をFigure 1.12に示す．また杉本ら[杉本:2005]は受動歩行の機械的な安定化の仕組みが，CPGと同様な位相(支持脚・遊脚期間)調節であることを示している．

Figure 1.11: Photo of 3D passive dynamic walking robot of Collins.

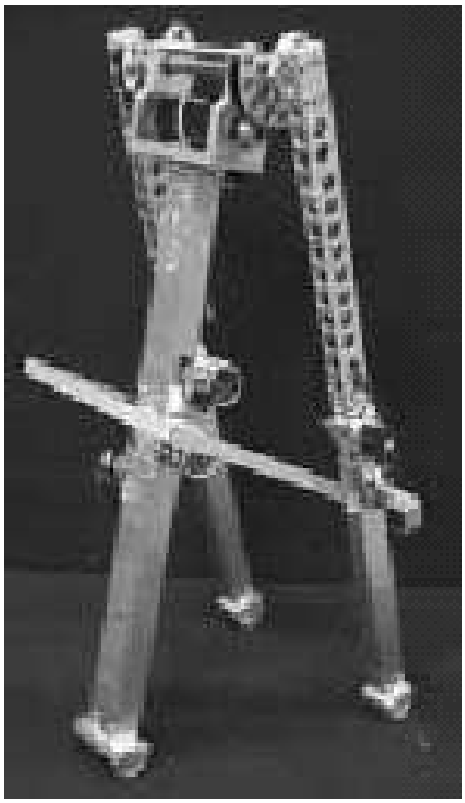


Figure 1.12: Photo of Ono's bipedal walking robot.

1.3 関連する先行研究

これまで人の歩行モデルや走行モデルを構成しようと試みようと，数多くの研究がなされている[内藤:2006, 稲垣:2007]．稲垣ら[稲垣:2007]や青井ら[青井:2007]は筋骨格系と神経系の相互作用による運動生成に基づいた人の歩行シミュレーションを行っている．宮腰[宮腰:2006]は，メモリベーストの多重軌道推移制御を用いた劣駆動二足歩行モデルの運動生成と制御を2次元平面内でシミュレートしている．実機を用いた実験では，倒立振子(Figure 4.3参照)は歩行に，SLIPモデル(Figure 1.13参照) は走行に対してそれぞれ頻繁に用いられている．Bastianら[Bastian:2008]はSLIPモデルを用いてsplit-belt treadmill上での走行モデルを構成しようと試みている．我々は二脚ロボットを用いた人のsplit-belt treadmill歩容適応モデルの構成を行っている．tied-belt treadmillを用いての研究報告[Hirai:1998, Tsujita:2007, Nomura:2006]がいくつかあるが，筆者の知るかぎりsplit-belt treadmill上における歩行モデルの構成を行っている他の研究グループは，現時点では見当たらない．

本研究室における二脚ロボットの研究は，2003年4月より始まった．有村は系の内部的

なフィードバックにより自励振動を発生させ平地での歩行を試みた．高橋[高橋:2005]は強制振動系と同調(引き込み)機能を組み合わせた手法，リズム生成器：CPG(Central Pattern Generator)を用いて歩行させようと試みた．だが，安定した歩行を実現することは困難であった．よって我々は安定な歩行を実現することが最優先であると考え，倒立振子モデルを用いた運動生成法を適用した．

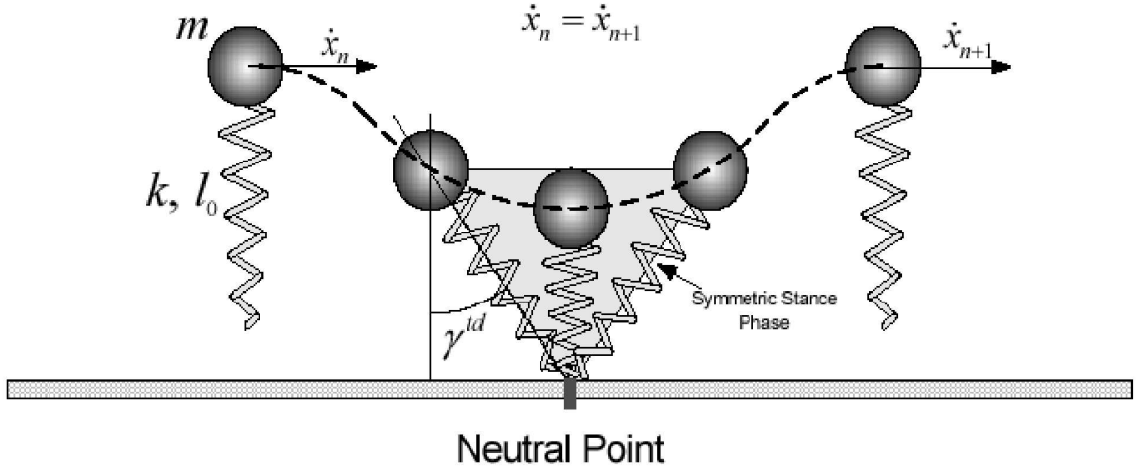


Figure 1.13: SLIP model.

1.4 本研究の目的

本研究の目的は，人(健常者や小脳疾患患者)のsplit-belt treadmill上での歩容適応モデルを二脚ロボットを手段として用いて構築することである．人の歩行の運動学，動力学，酸素吸収量を調べるためにトレッドミル上での歩行実験が数多く行われている．特に，左右のベルトの速度が異なるsplit-belt条件での歩行パターンの適応を調べると，健常者と小脳疾患患者では運動学的パターンに大きな相違が現れる[Reisman:2005, Morton:2006]．この相違の詳細については**2.2** 節，**2.3** 節を参照していただきたい．この現象の裏にあるメカニズムを解明するためには，筋骨格系モデルを用いたシミュレーションを行いその結果を人の歩行実験結果と比べることが実際的である．しかし地面との摩擦や衝突・伸縮素材の影響などをシミュレートすることが困難である．そこで，本研究では2次元二脚歩行ロボット「鉄郎」を開発し，実機ベースで人の歩行モデルを構成し，そのモデルがどのように働いているか検証する．また小脳疾患患者の歩行障害モデルを構成することができれば，将

来的にリハビリや歩行介助装置への応用が期待できる．鉄郎の機構の特徴としては，足首関節にプーリを用いることで遊脚の慣性モーメントをできるだけ小さくしたこと，footの面積を小さくして路面の影響を受けにくくしたこと，ギア比を低くすることで関節を軟らかくしたこと，スポンジなどの伸縮素材を各関節に挿入したことなどがあげられる．制御手法としては**1.2** 節の最後と**2.4** 節で述べた理由より，CPGの単純化モデルとしてリズミカルな運動を発生させる軌道生成・脊髄における緊張性伸張反射の単純化モデルとしての関節におけるPD制御・前庭脊髄反射として前進速度と姿勢の安定化のための脳幹におけるstepping reflexから成る従来のリミットサイクルを構成する制御手法に新たに提案した小脳による適応の単純化モデルとしての支持脚腰関節におけるPゲイン調節を組み合わせた手法を用い，歩容適応を試みた．

1.5 本論文の構成

本論文の構成は以下の通りである．第1章においては，研究背景，従来の二脚歩行制御手法の紹介，関連する先行研究を紹介した後，本研究の目的を述べた．第2章は健常者と小脳疾患患者のsplit-belt treadmill歩行の実験結果を紹介する[Reisman:2005, Morton:2006]．ここでなぜ我々がsplit-belt treadmillに着目したか詳細な理由を述べる．加えて構成した歩容適応モデルについて説明する．第3章は機構的に工夫して開発した二脚ロボット「鉄郎」について述べる．第4章は，関節でのPD制御・脊髄における周期的な運動生成を規範とした倒立振子に基づく軌道計画・脳幹における制御を規範とした遊脚着地角制御（stepping reflex）から成る従来のリミットサイクルを構成する制御手法について述べる．この従来型の制御手法で行ったtied-belt treadmill歩行実験について報告する．そして，鉄郎にsplit-belt条件での自律的な適応歩行を実現させるために導入した，小脳における運動調節を規範とした支持脚腰関節におけるP-gain調節を提案する．この章の最後に鉄郎と人(健常者と小脳疾患患者)の実験結果[Reisman:2005, Morton:2006]を比較し，提案した制御モデルの正当性について議論を行う．第5章は，実験結果についての考察を述べる．測定された指標の比率とパターンの高い類似性は，我々の仮説と提案したモデルが正当であることをほのめかしていると考える．第6章に結論，第7章に今後の展望を述べてまとめとする．

1.6 語彙解説

- 脚—腰関節から下の部分
- 支持脚(相)—地面に足が接触している脚(期間)
- 遊脚(相)—地面に足が接触していない脚(期間)
- クリアランス—脚を振り出す際のつま先と地面の距離
- 大腿リンク—膝関節から上の部分．膝上リンク
- すねリンク—膝関節から下の部分．膝下リンク
- foot—足首関節から下の部分．つま先部とかかと部を合わせてfootと呼ぶ
- 接地—footが地面と接触する状態．瞬間
- 離地—footが地面から離れる状態．瞬間
- ステップ—ここでは歩数とする
- 歩行周期—周期的な歩行において，再度同一の状態になるまでの時間．2ステップ分に相当
- デューティ比—歩行周期に占める支持脚相の割合
- 片脚支持期—片脚が接地している期間
- 両脚支持期—両脚が接地している期間
- 整地—障害物のない平地，平坦な路面
- 不整地—整地以外の路面
- 生物規範型—運動制御，機構などにおいて生物の持つ特性を参考にし，モデル化すること
- 仮想バネ・ダンパ機構—ここでは支持脚足首関節角度・角速度をPD制御により一定にすることで受動的なバネ・ダンパ機構とみなすこと

- 静歩行—歩行中のすべての瞬間において静的に安定な状態を保ちながら行う歩行
- 動歩行—歩行中に静的に不安定な瞬間が存在する歩行
- 適応—与えられた目的を実現するために外界の変化に応じて運動パターンを変化させる機能[土屋:1999]
- 反射—本論文では，センサ情報に基づくトルク出力を反射と呼ぶ
- 屈曲反射—外部からの刺激によって屈筋が収縮，伸筋が弛緩する脊髄反射
- ZMP—Zero Moment Pointの略称．歩行ロボットにおいて一般的に用いられる安定指標．この点から発生する力が重心点を通る場合，回転モーメントが生じないため並進運動のみを生成することができる
- ナチュラルダイナミクス(natural dynamics)を使用した歩行—機構の特性や重力を有効利用しできる限りアクチュエータのトルク出力を小さくすることで実現するエネルギ効率の良い歩行
- PDW—アクチュエータを持たない機構が重力を利用して坂を下ること．本論文ではPassive Dynamic Walking(受動動歩行)の略称としてPDW を用いる
- 創発—力学特性をもつ要素と要素間の力学的相互作用という非線形力学システムの中で，個々の要素の性質に還元できない大域的なパターン(運動)が発生すること
- リミットサイクル—位相空間内における閉ループ．初期値に依存せず，軌道が近傍にて安定構造を持ち，近傍の状態は閉空間に収束する．外乱に適応可能．
- 倒立振子—振り子を逆立ちさせたもの．通常，支点にアクチュエータが存在しないものを指し，不安定な系の典型として使われる
- CPG—Central Pattern Generatorの略称．外部の運動との相互引き込み作用を持つリズム生成器
- Split-belt treadmill—左右独立にベルトの速度調節ができるトレッドミル
- ストライド長—片脚が離地してからその片脚が接地するまでの移動距離

- ステップ長—片脚が接地した時の両脚間の距離
- 後遺症—ここでは左右のベルト速度が変化した直後に残存する症状

第2章　人のSplit-Belt Treadmill歩行の紹介と歩容適応モデルの提案

この章では，Bastianら[Reisman:2005, Morton:2006]による人(健常者と小脳疾患障害者)のsplit-belt treadmill歩行の実験結果を紹介した後，提案した歩容適応モデルについて述べる．健常者のsplit-belt treadmill歩行実験の結論として，Bastianらは互いに独立して存在する2種類の神経制御があると述べている[Reisman:2005]．小脳疾患患者のsplit-belt treadmill歩行実験の結論として，Bastianらは2種類の適応機構があると述べている[Morton:2006]．Bastianらの仮説や生理学的知見に基づき，我々は以下に示すようなメカニズムを二脚ロボットを用いて構成し，split-belt treadmill上における歩容適応モデルを提案する(Figure 2.8参照)．

2.1　実験設定

人は，当然左右のベルトの速度が同じ(tied-belt条件)であっても異なって(split-belt条件)いてもトレッドミル上を歩行できる．トレッドミルの速度は“遅いモード”(0.5 [m/s])か“速いモード”(1.0 [m/s]) とする．tied-belt条件においては双方のベルト速度は遅いモードとする．split-belt条件においては片方のベルト速度は遅いモードとし，もう片方のベルト速度は速いモードとする．実験の時間設定は3つのステージに分けられる(Figure 2.1参照)．baselineステージにおけるトレッドミルの速度は，tied-belt条件とする．adaptationステージにおけるトレッドミルの速度は，split-belt条件とする．post-adaptationステージにおけるトレッドミルの速度は，再びtied-belt条件とする．

トレッドミル歩行の際，4つの運動学的指標を測定する．ストライド長とステップ長の定義をFigure 2.2に示す．ストライド長とステップ長は互いに依存しない指標である．デューティ比を歩行周期における支持脚期間の比と定義する．両脚支持期間比(RDLSP)を歩行周期における両脚支持期間(DLSP)の比と定義する．これらすべての指標は，各脚それぞれに対して測定される．

split-belt条件での“遅いモード”と“速いモード”のベルトに対応するそれぞれの脚を“slow

leg”，“fast leg”とする．baseline・post-adaptationステージにおいて，双方のベルト速度は遅いモードになるが，説明の簡単のためそれぞれの脚の名称をこのsplit-belt条件における脚の名称とする．遊脚におけるfast legが接地しslow legの支持脚期間が終わるまでの両脚支持期間を“slow” DLSPとする．それに対して，遊脚におけるslow legが接地しfast legの支持脚期間が終わるまでの両脚支持期間を“fast” DLSPとする．ストライド長とデューティ比は各脚それぞれに対して測定された値を用いる．よって，これらの変数は各脚独立した指標(intralimb indexes)である．それに対して，ステップ長と両脚支持期間比は各々の「“fast leg”の値と“slow leg”の値の差 」を計算する．よって，これらの変数は各脚が相互依存した指標(interlimb indexes)である．

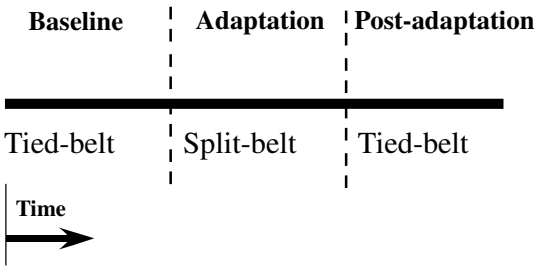
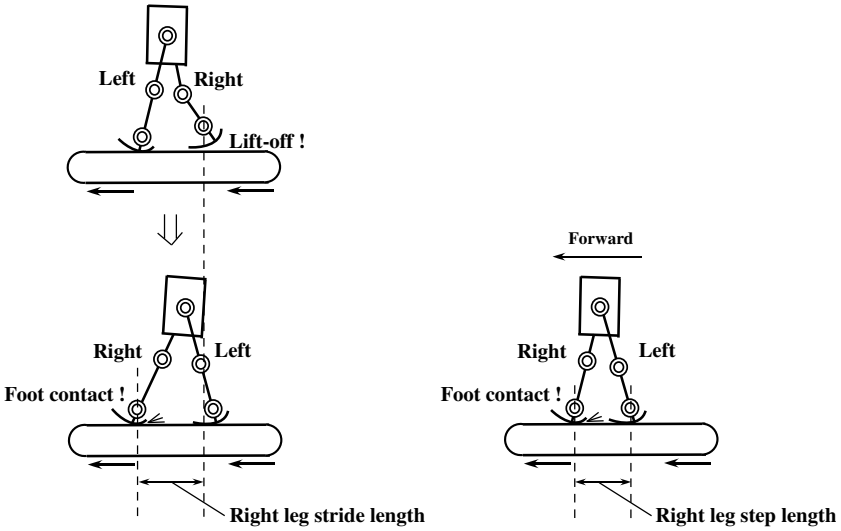


Figure 2.1: Three stages in experiments of split-belt treadmill walking.



(a) Definition of stride length (b) Definition of step length

Figure 2.2: Definitions of the stride length (left) and the step length (right). The stride length is defined as the distance traveled by the ankle joint of one leg from the instant of lift-off to the instant of foot contact of the leg. The step length is defined as the distance between positions of the ankle joints of swing and stance legs at the instant of foot contact of the swing leg.

2.2 実験その1～健常者のSplit-belt treadmill歩行～

Bastianらにより行われた健常者のsplit-belt treadmill歩行のイメージをFigure 2.5に示す．健常者のsplit-belt treadmill歩行実験において計測されたストライド長とデューティ比をFigure 2.3に示し，ステップ長と両脚支持期間比の差をFigure 2.4に示す(これらの図は[Morton:2006]より抜粋し著作権の侵害にならないように修正した)．これらの図のプロット点は，歩行 1 周期において各々計測した指標である．

ストライド長とデューティ比について

Figure 2.3のadaptationステージにおいて，fast legのストライド長は著しく長くなりslow legにおいては少し短くなった．fast legのデューティ比は低くなり，slow legにおいてはほぼ一定で変わらなかった．Figure 2.3におけるこれらの指標は，baselineステージにおいてはほぼ一定で，adaptationステージの始めにおいて素早く変化しその後そのステージ内でほぼ一定を保った．そしてpost-adaptationステージの始めにおいて素早くbaselineステージの元の値に戻った．

ステップ長と両脚支持期間比の差について

Figure 2.4におけるステップ長と両脚支持期間比の差のパターンについて見てみる．baselineステージにおいて，ステップ長の差はほぼ一定で両脚支持期間比の差は一定でないことが見られた[1]．これらの指標はadaptationステージの始めにおいて素早く変化し[2]，その後そのステージ内で徐々に元の値に戻った．また，post-adaptationステージの始めにおいて素早く変化し[3]，その後再びそのステージ内で徐々に元の値に戻った．ここにふたつの興味深い現象が存在する．ひとつはこれらの指標が，adaptationステージの終わりに左右のベルトの速度差があるのにもかかわらず，ほぼゼロに戻ったことである．これは健常者がsplit-belt条件においても対称歩行を好むことを意味している．もうひとつは，post-adaptationステージの始めに左右のベルトの速度が同じであるのにもかかわらず，素早く変化したことである．この現象は，広く知られている「後遺症」のひとつである．

[1] まだその原因を解明できていない．
[2] この時slow legの値はfast legの値より大きくなった．これは非対称歩行であることを意味している．
[3] この時fast legの値はslow legの値より大きくなった．これはもうひとつの非対称歩行である．

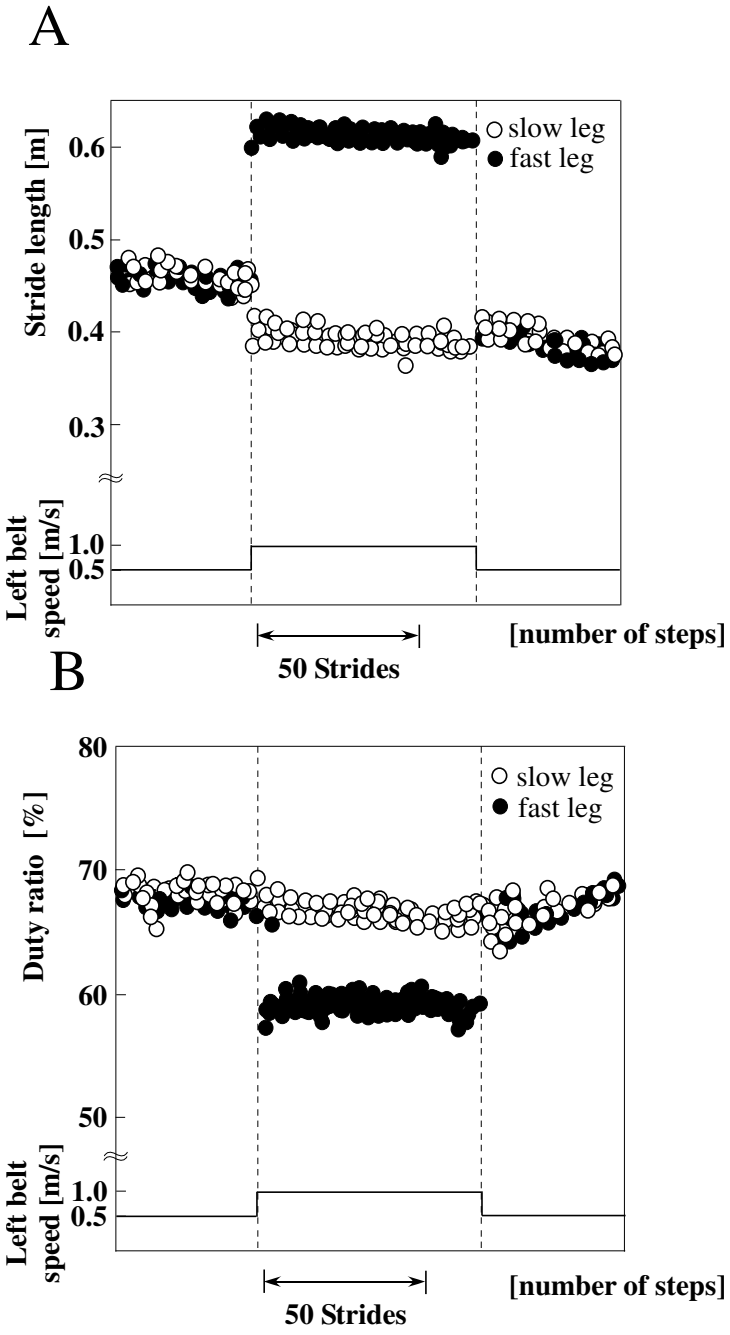


Figure 2.3: The stride length (A) and the duty ratio (B) in normal subject split-belt treadmill walking are shown, where speed of belts were 0.5 m/s at both left and right belts in the baseline stage, 0.5 m/s at the left belt and 1.0 m/s at the right belt in the adaptation stage, and 0.5 m/s at both left and right belts in the post-adaptation stage. Speed of the fast belt is also shown (modified from Morton et al. 2006 [Morton:2006]).

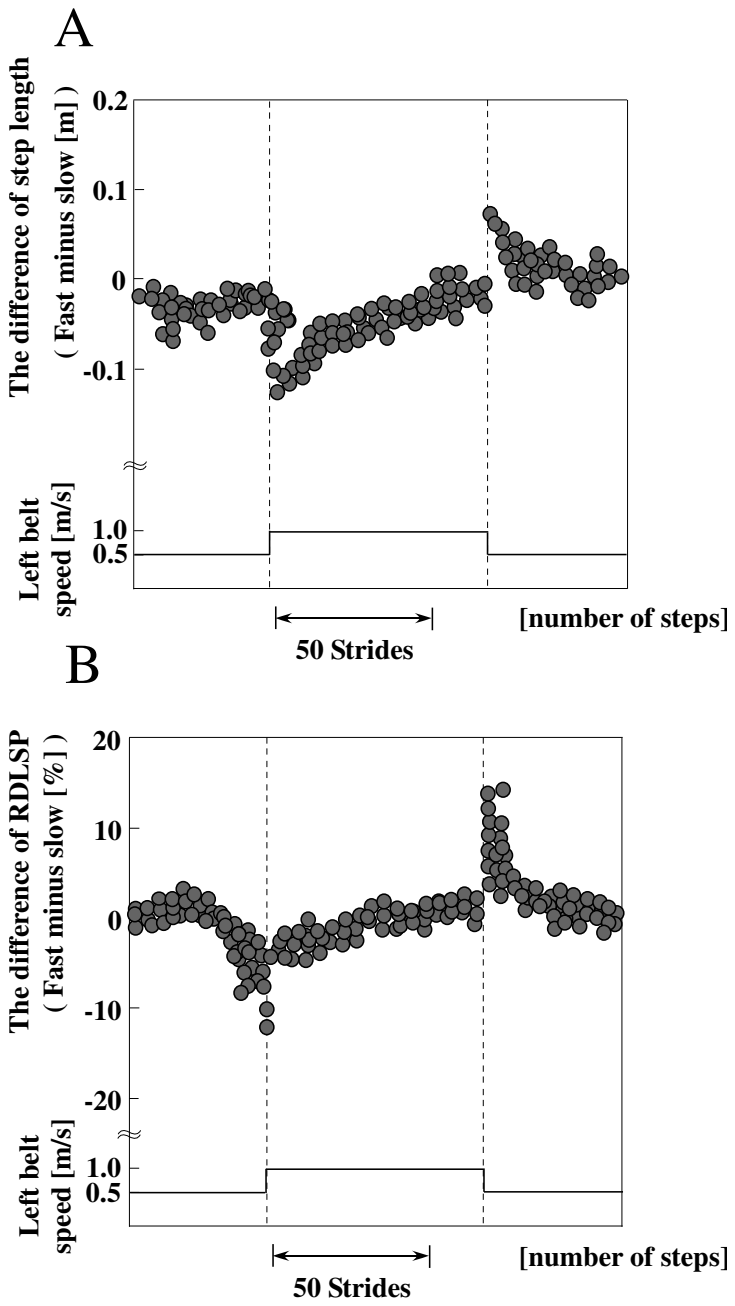


Figure 2.4: The step length difference (A) and RDLSP (ratio of the double legs stance period) difference (B) in normal subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006 [Morton:2006]).

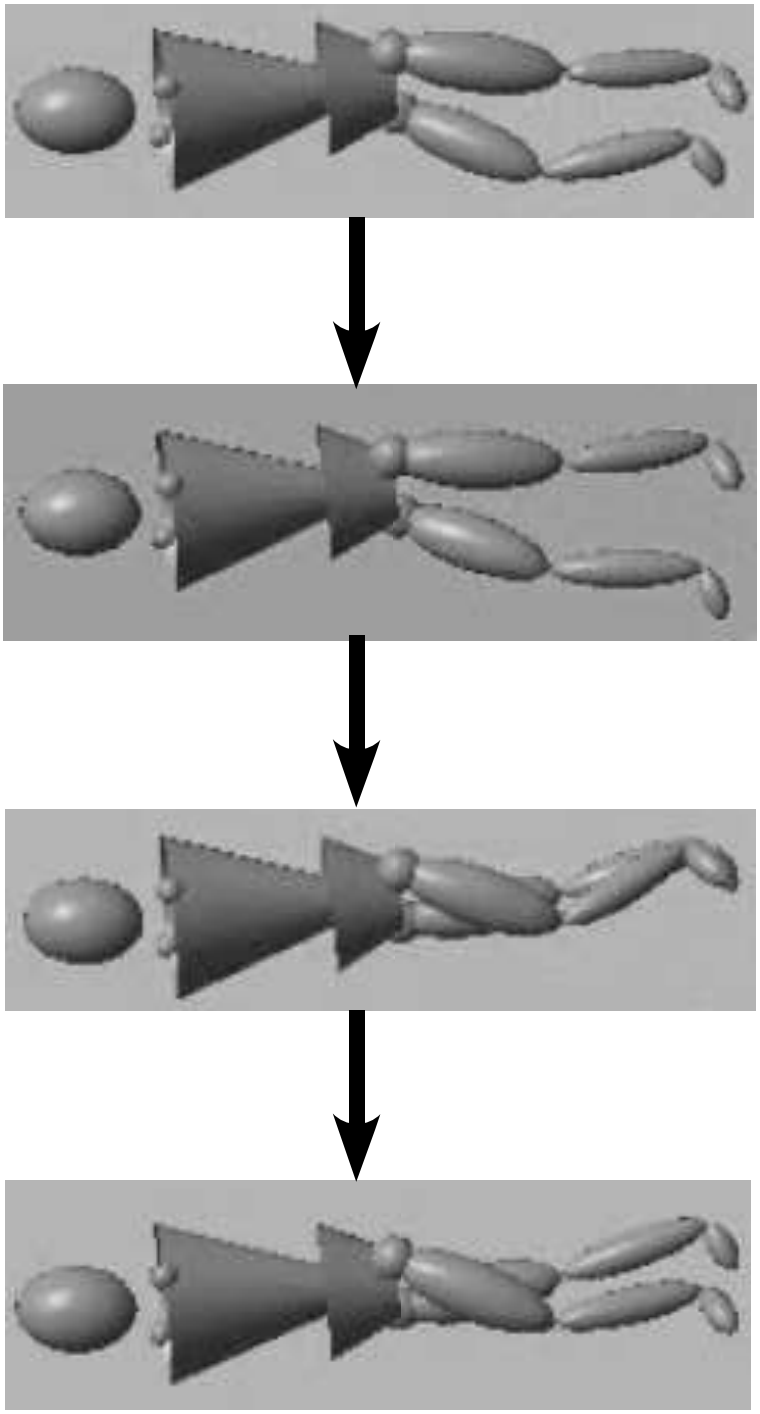


Figure 2.5: Snapshots on human (normal subject) split-belt treadmill walking in the adaptation stage (captured from Morton et al. 2006[Morton:2006]).

2.3 実験その2～小脳疾患患者のSplit-belt treadmill歩行～

小脳疾患患者のsplit-belt treadmill歩行実験において計測されたストライド長とデューティ比をFigure 2.6に示し，ステップ長と両脚支持期間比をFigure 2.7に示す(これらの図も[Morton:2006]より抜粋し修正した).

ストライド長とデューティ比について

Figure 2.6-Aのadaptationステージの始めにおいて，小脳疾患患者は，健常者の変化パターン(Figure 2.3参照) と同様にfast legのストライド長は大幅に長くなる，slow legにおいては少し短くなるような素早い感覚的な調整を行っていることを示した．post-adaptationステージの始めにおいても小脳疾患患者は，健常者の変化パターン(Figure 2.3参照)と同様にbaselineステージのストライド長の値にほぼ戻るような素早い感覚的な変化を示した．Figure 2.6-Bにおいて小脳疾患患者は，デューティ比の値に多少のばらつきがあるものの，健常者の変化パターン(Figure 2.3-B参照)と同様の反応を示した．

ステップ長と両脚支持期間比の差について

小脳疾患患者のステップ長の差(Figure 2.7-A)と健常者のステップ長の差(Figure 2.4-A)を比較すると，adaptationステージに始めに双方に素早い変化が見られる．しかし小脳疾患患者のステップ長の差において，adaptationステージ内の徐々に元の値に戻るパターンやpost-adaptationステージの始めに起きる後遺症が見られない．小脳疾患患者の両脚支持期間比の差(Figure 2.7-B)と健常者の両脚支持期間比の差(Figure 2.4-B)を比較すると，小脳疾患患者のステップ長の差と同様に徐々に元の値に戻るパターンや後遺症が見られない．我々はこのような適応パターンの背景にダイナミクスが存在すると考えた．そこでこの適応に対するモデルを提案し，実機実験を行ってモデルの妥当性を検証した．上記がsplit-belt treadmill歩行に着目している理由である．

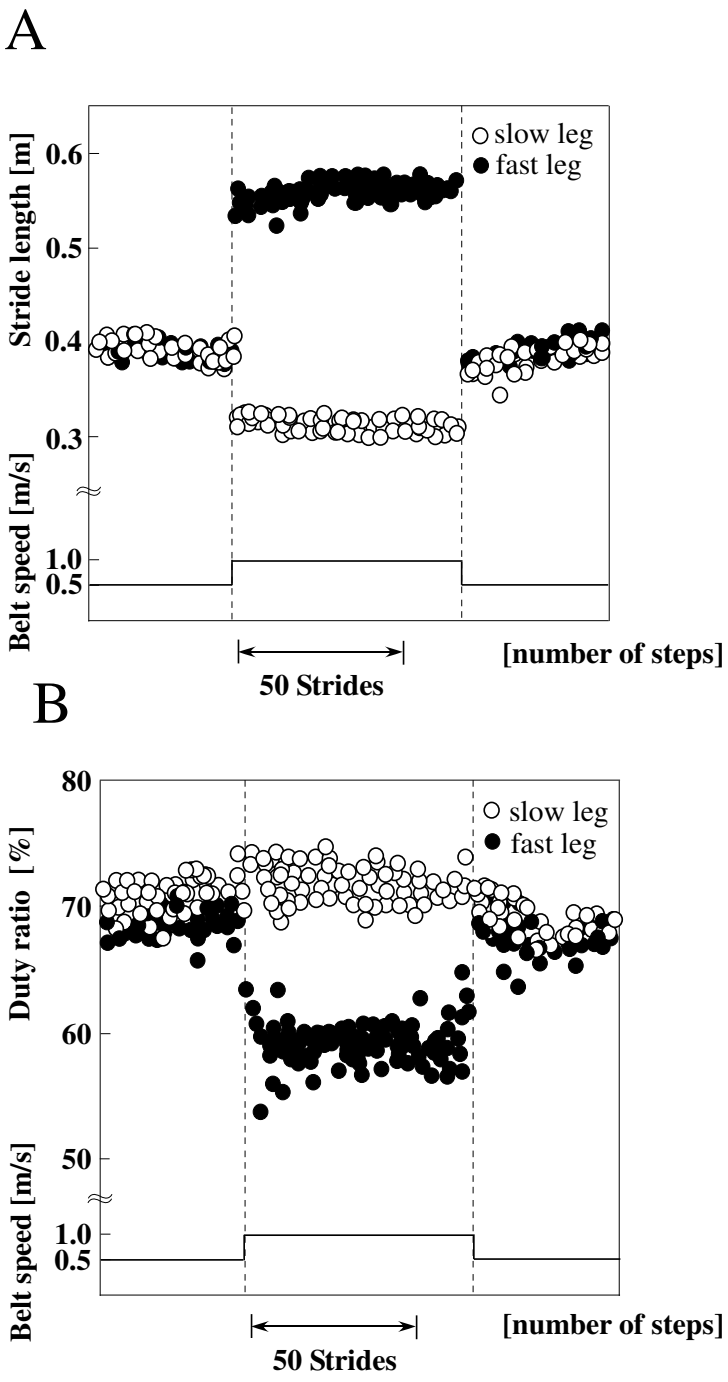


Figure 2.6: The stride length (A) and the duty ratio (B) in cerebellar disease subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006 [Morton:2006]).

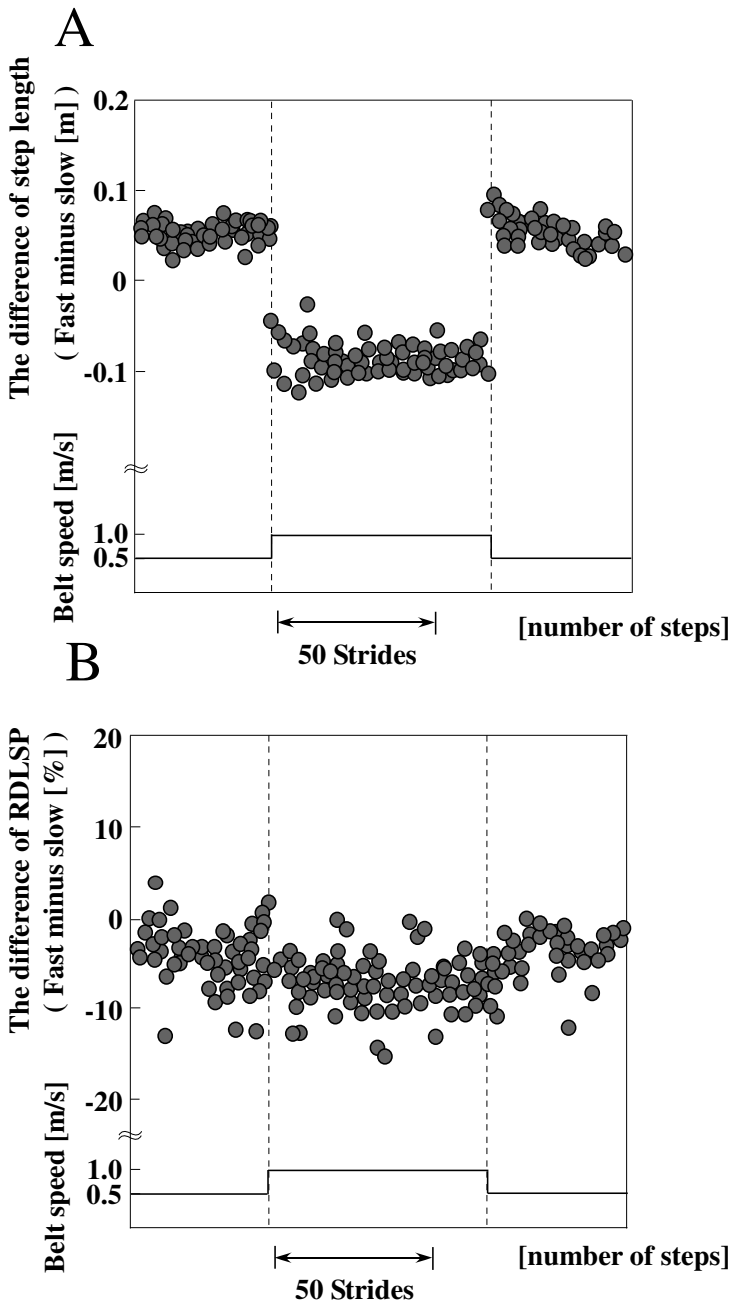


Figure 2.7: The step length difference (A) and RDLSP difference (B) in cerebellar disease subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006[Morton:2006]).

2.4　歩容適応モデルの提案

健常者のsplit-belt treadmill歩行実験の結論として，Bastianらは互いに独立して存在する2種類の神経制御があると述べている[Reisman:2005]．ひとつは，各脚独立にsplit-belt treadmillに素早く対応し後遺症を示さないストライド長やデューティ比を調整するintralimb coordinationである．もうひとつは，adaptationステージにおける遅い段階的な変化やpost-adaptationステージにおける後遺症を示すステップ長と両脚支持期間比の差を調整するinterlimb coordinationである．彼らはこのようなinterlimb coordinationは両肢間の相対運動を最適化すると述べている．さらに，このcoordinationに関連する中枢神経系の分野にふたつの可能性があると示している．ひとつの可能性は脊髄のネットワークである．もうひとつは，小脳もしくは小脳のコントロール下における前庭脊髄経路の脳幹神経であると述べている．

小脳疾患患者のsplit-belt treadmill歩行実験の結論として，Bastianらは2種類の適応機構があると述べている[Morton:2006]．ひとつは，小脳疾患患者においても正常に機能する脊髄や脳幹のような下位の神経中枢により主に制御される反応の速い感覚的フィードバック適応機構である．もうひとつは，小脳疾患により著しく損なわれる予見的フィードフォワード適応機構である．

除脳ネコの実験結果として歩行における両肢間のリズミカルな運動が，脊髄や脳幹における反射によって駆動されるリズム生成器：CPGs(Central Pattern Generators)により発生することが一般的に認められている[Orlovsky:1999, Rossignol:2006]．我々は，このような神経制御が人の二脚歩行においても行われていると仮説を立てる[Giese:2007, Orlovsky:1999, Rossignol:2006]．上記の生理学的知見やBastianらの仮説に基づき，我々は以下に示すようなメカニズムを二脚ロボットを用いて構成し，split-belt treadmill上における歩容適応モデルを提案する(Figure 2.8参照)．

(1) 周期的な運動を行う軌道生成(リズミカルな運動を発生させるCPGの単純化モデル)[4]

(2) 脊髄における緊張性伸張反射の単純化モデルとしての関節におけるPD制御

(3) 前庭脊髄反射として前進速度と姿勢の安定化のための脳幹におけるstepping reflex[5](**4.1.4**

[4]将来的には相互引き込み機能[Orlovsky:1999, Rossignol:2006]を組み合わせる．

[5]我々は前庭情報としてレイトジャイロの出力値を用いず，stepping reflexとして支持脚足首関節の角速度を用いた．なぜならレイトジャイロの値はノイズが乗りやすく，関節角速度のほうがノイズが少なく扱いやすいからである．

節で述べるようにこのメカニズムはinterlimbコントロールである)

(4) 小脳による適応の単純化モデルとしての支持脚腰関節のPゲインの調節(**4.2.1**節で述べるようにこのメカニズムはintralimbコントロールである)

特に小脳は歩行中step-by-stepに筋張力を調節しているという知見[Kandel:1996]に基づいて，我々はこのPゲイン調節をあらたに提案する．これらメカニズムの詳細を第4章に記述する．

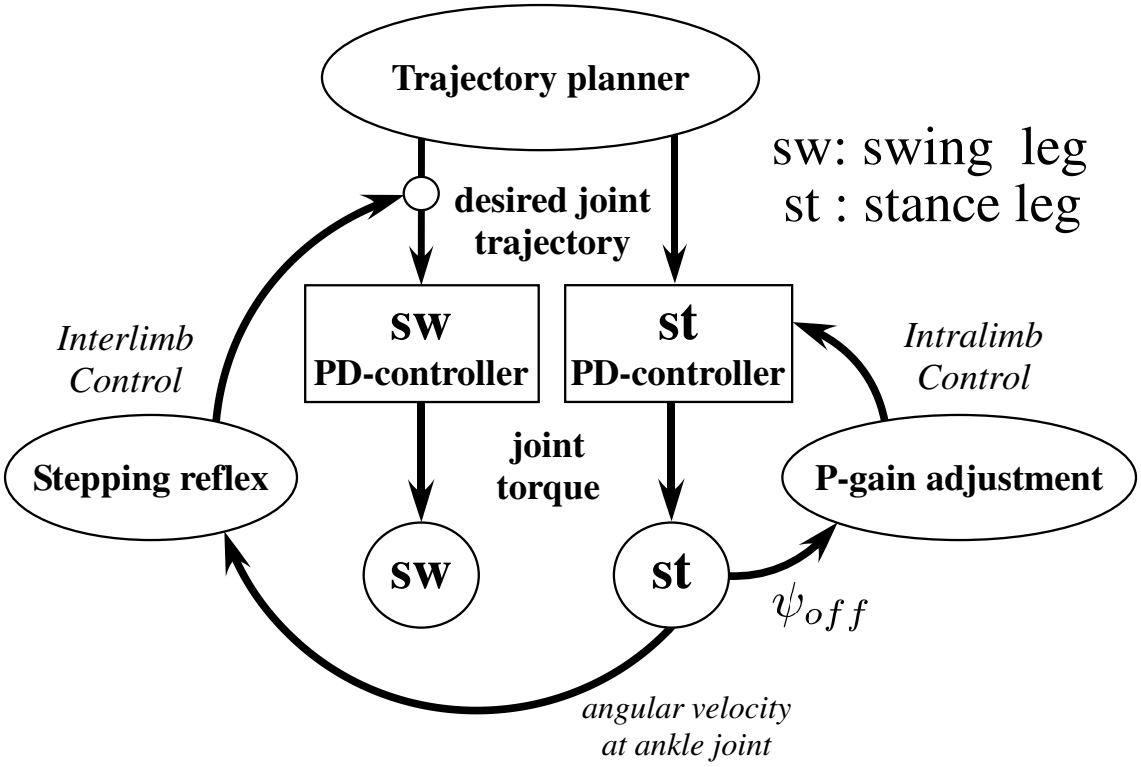


Figure 2.8: Control diagram of biped robot split-belt walking. The stepping reflex acts as interlimb control since hip joint angle of the swing leg is adjusted by ankle joint angular velocity of the stance leg as described in Section **4.1.4**. P-gain adjustment acts as intralimb control since P-gain at hip joint of the stance leg in the next stance phase is adjusted by hip joint angle of the leg at last lift-off as described in Section **4.2.1**.

第3章　二脚ロボット「鉄郎」

人のsplit-belt treadmill歩行の歩容適応モデルを構成するため，加えて簡潔な制御手法でエネルギ効率の良い歩行を行うために開発された，2次元二脚歩行ロボット「鉄郎」を紹介する．機構設計の際，制御上の重要な点を限定し易くするために自由度をできる限り少なくした．

3.1　鉄郎の機構と仕様

鉄郎に搭載したセンサとその用途，仕様，全体画像，足首関節より下の拡大図，簡易設計図をそれぞれTable 3.1，Figure 3.1，Figure 3.2，Figure 3.3，Figure 3.4に示す．脚長は立脚静止時で40[cm]，総質量は2.3[kg]となっている．胴体，上腿，下腿の各リンク長および質量は，それぞれ6[cm]，0.7[kg]，18[cm]，0.3[kg]，20[cm]，0.5[kg]となっている．各脚は，ピッチ軸まわりに回転する腰，膝，足首関節を持つ(Figure 3.4 参照)．本実験では並行した2本のパイプ各々にリニアブシュを通し，それらの間に鉄郎の胴体リンクに取りつけた2枚の拘束板(Figure 3.2参照)を挟むことにより，ロール及びヨー運動が起きないようにした(Figure 4.9 参照)．これらの拘束板とリニアブシュの接触面における摩擦が小さいため，胴体にピッチ軸まわりの3自由度を持たせることができ，鉄郎は2次元平面内で滑らかに動く．ゆえに鉄郎は，遊脚のつまずき[1]や大きな外乱などにより最悪の場合容易に転倒してしまう．鉄郎はピッチ軸まわりに回転する胴体リンクを持つが，その重心は腰関節よりも少し下にあるため上体（トランク）無しのモデルである．

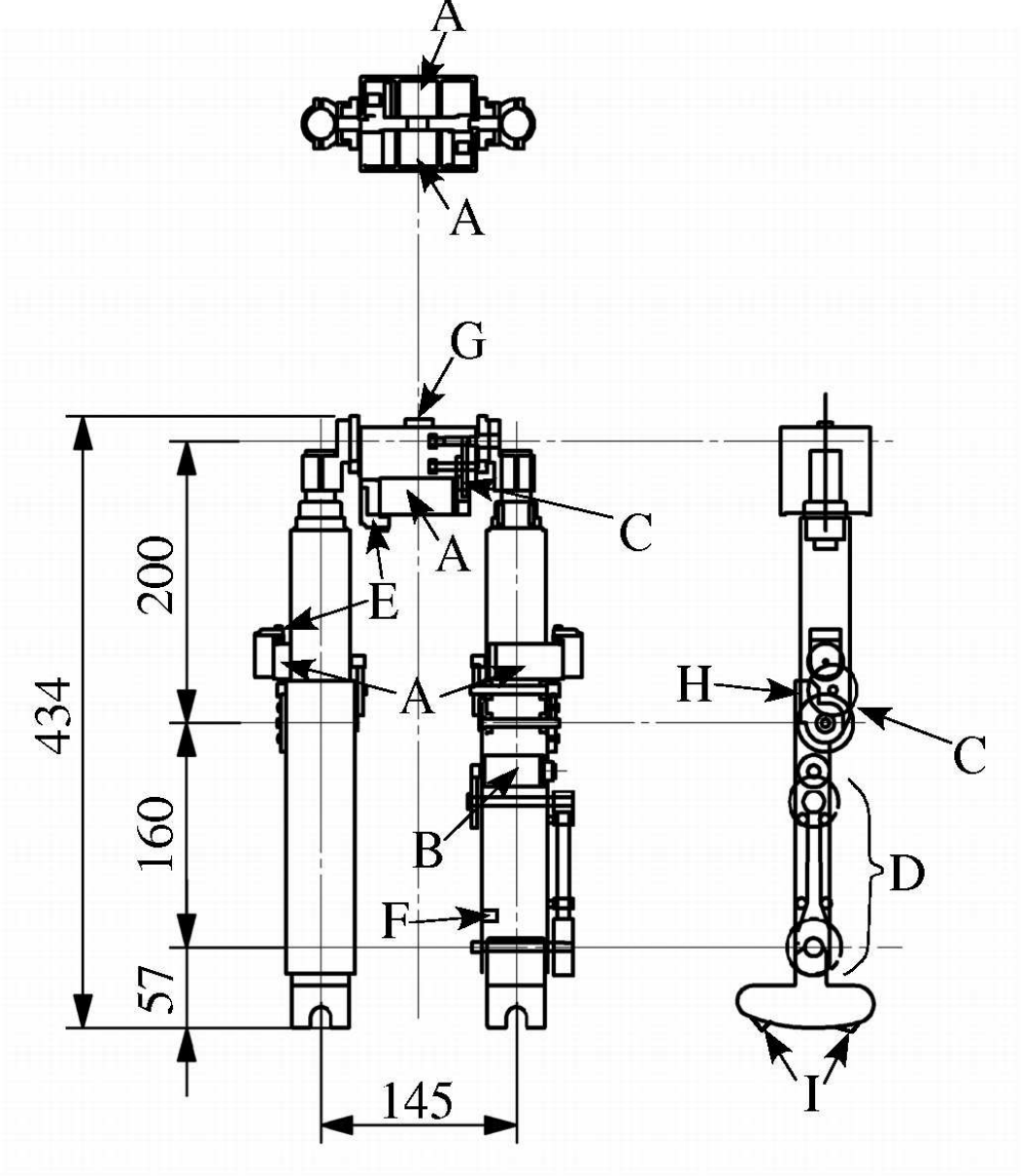
腰，膝，足首関節はそれぞれ20[W]，20[W]，19[W]のDCモータにより駆動され，平歯車により減速されている．減速比はそれぞれ25，25，8.7となっている．減速比を小さく設計して各関節のギア間における摩擦を小さくし，各関節に高いバックドライバビリティを持たせることにより，鉄郎はナチュラルダイナミクスを利用した効率的な歩行を実現することができる[Otoda:2007]．

[1]屈曲反射を実装すれば，鉄郎は遊脚のつまずきによる転倒を避けられる．

各関節に相対関節角検出用の光学式エンコーダ，胴体ピッチ軸まわりにレイトジャイロ(Figure 3.4参照)，足裏部に接地センサとして機械的なスイッチ(Figure 3.3参照)を搭載している．足裏の特徴として，支持脚期間におけるピッチ軸まわりの重心の移動がスムーズに行えるように円柱状の屈曲を持たせた(Figure 3.3参照)．

Table 3.1: Implemented sensors and the usages.

搭載したセンサ	用途
エンコーダ	ピッチ軸まわりの腰・膝・足首関節の角度検出
機械的なスイッチ	つま先部・かかと部に挿入しスイッチのオンオフを検出することにより接地と離地を検出
レイトジャイロ	ピッチ軸まわりの角速度から胴体傾斜角度を算出



A	DC motor of hip and knee pitch joint	MAXON RE25 20[W]
B	DC motor of ankle joint	FAULHABER 2342CR 19[W]
C	Spur Gear	Reduction ratio : hip(25) and knee(25)
D	Pulley of ankle joint	Reduction ratio : 18
E	Digital Encoder of hip and knee joint	MAXON 100PPR, 2channels
F	Digital Micro Encoder of ankle joint	MTL 360PPR,2channels
G	Rate gyro around pitch axis	MURATA GYROSTAR ENC-03J
H	Urethane gel	EXSEAL
I	Mechanical switches on toe and heel	CHERRY

Figure 3.1: Spec of Tetsuro.

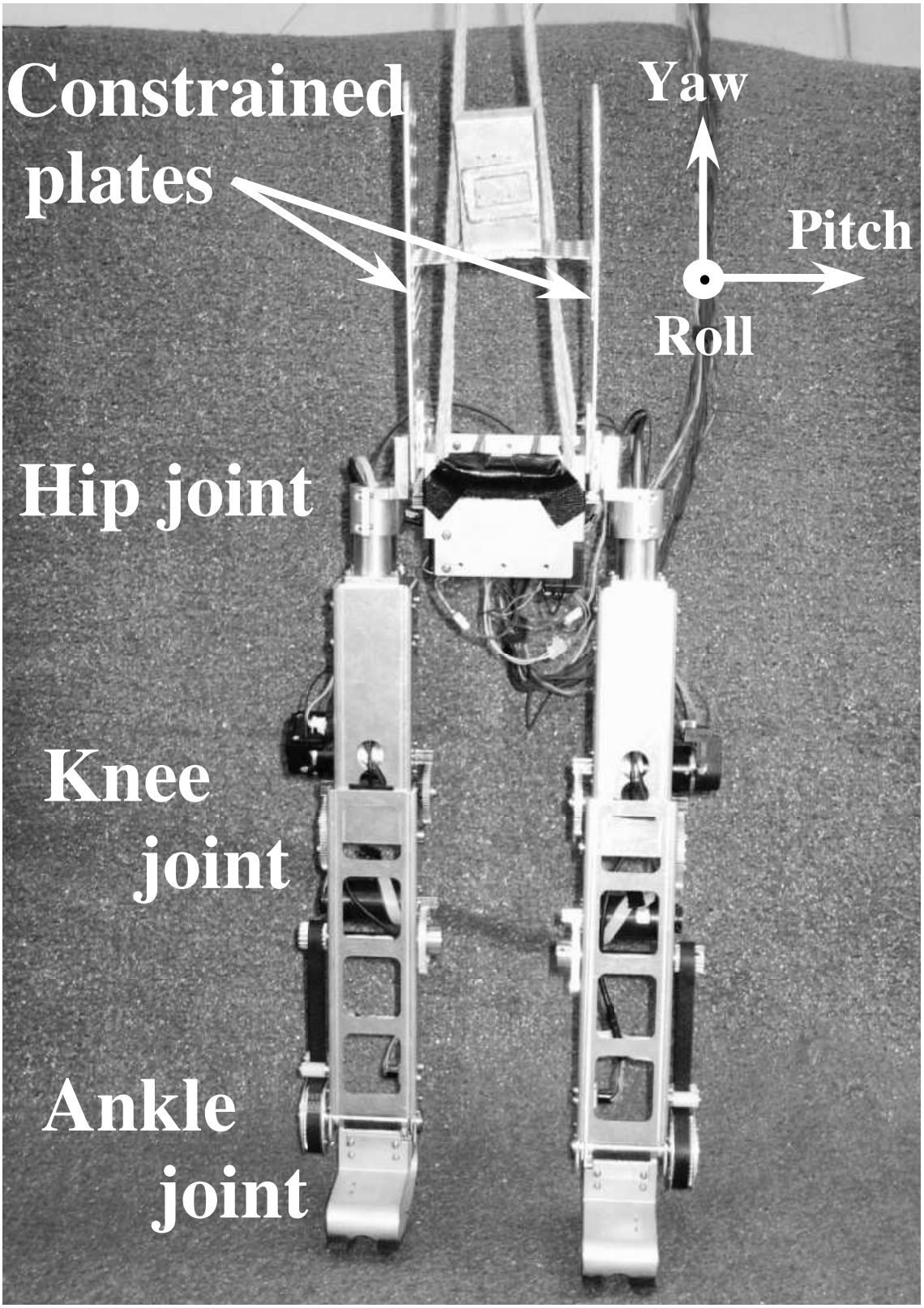


Figure 3.2: Photo of Tetsuro.

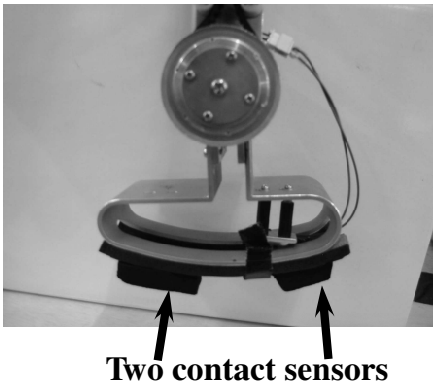


Figure 3.3: Photo of ankle joint, foot and contact sensors beneath the sole.

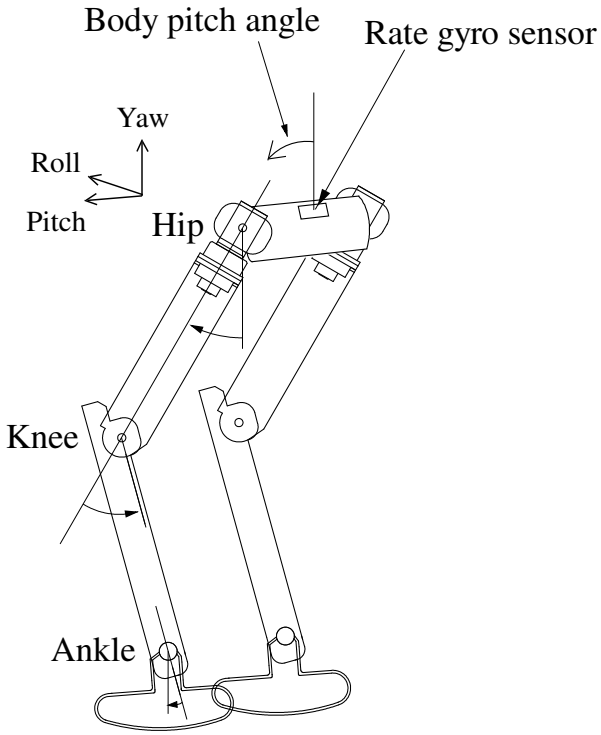


Figure 3.4: Mechanical draft of Tetsuro.

3.2 制御システム

鉄郎の制御ハードウェアの拡大画像をFigure 3.5に，鉄郎に用いられた市販小型ロボットコントローラTITech-Wire[福島:2002]の概略をFigure 3.6に表す．鉄郎に用いられたTITech-Wireは，ロボット各部のセンサ，モータなどを制御する複数の小型コントローラモジュールを単一コードでバス型接続に連結し，通信速度2Mbps以上のシリアル通信によって終端の小型CPUモジュールに接続する小型ロボットコントローラである．CPUモジュールには外部から無線LANを通じてアクセスできる．また，AD module, およびDigital moduleの寸法と質量は，4.3×3.0×2.0[cm]の20[g]，CPU moduleは6.4×9.0×2.0[cm]の90[g]である．Figure 3.5 に表すように，コントローラ(AD module，Digital module，CPU module，モータドライバ)，センサ用フィルタ・増幅回路，電源(24[V] 供給) はロボット外にありケーブルでつながっているが，歩行中のケーブルの影響は小さい．Figure 3.7に鉄郎の実験環境イメージを示す．OSはRT Linuxを用い，1スレッドのリアルタイム制御を行っている．リアルタイムスレッドは主に，

(1) モータ駆動指令

(2) レイトジャイロ値計測・積分

の2つの処理を行っており，サンプリングタイムは1m[sec]である．(2)に関して，胴体に搭載したピッチ·ロール軸回りの胴体傾斜角速度を検出するレイトジャイロの値を1m[sec]ごとに計測し，それを数値積分することで胴体傾斜角度を算出するが，時間とともに積分誤差が蓄積していくので，1m[sec] 間隔でローパスフィルタにより算出した低周波胴体傾斜角度値を用いてそれを補正する．つまり，レイトジャイロにより傾斜角度の高周波成分，ローパスフィルタにより低周波成分をセンシングすることによって，測定値として使用できる程度の胴体傾斜角度のセンシングが可能になっている．

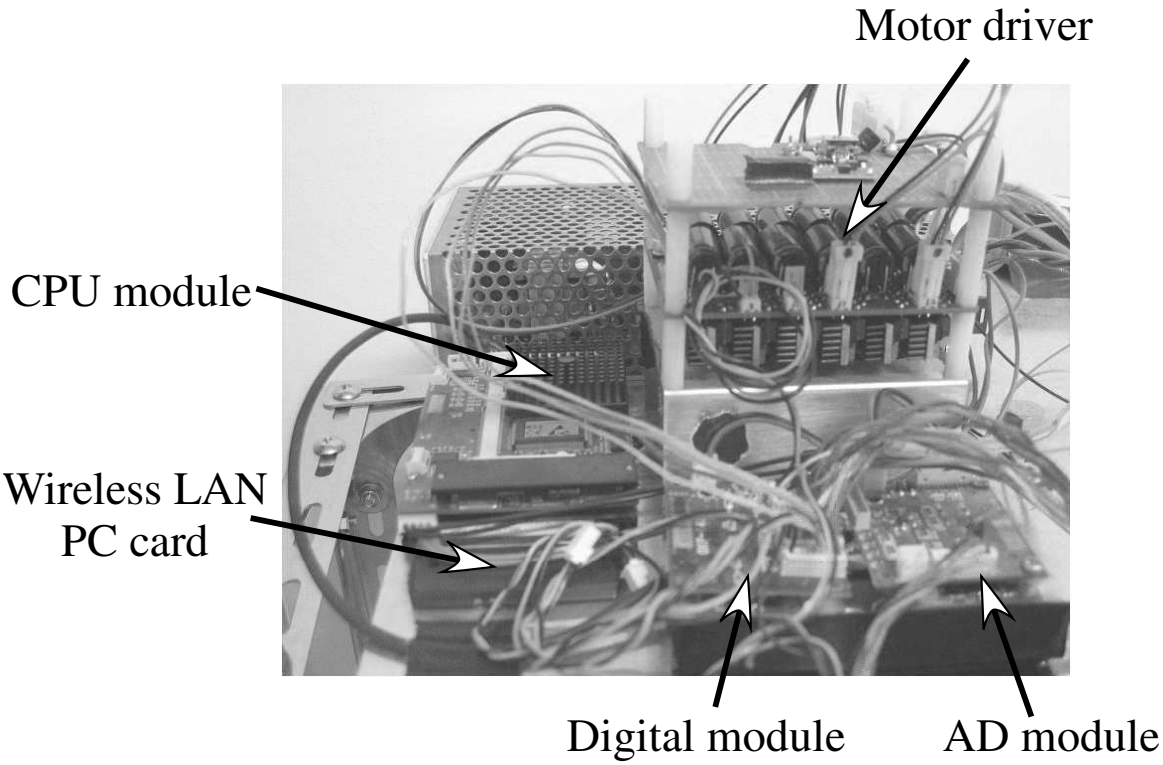


Figure 3.5: Photo of controller for Tetsuro.

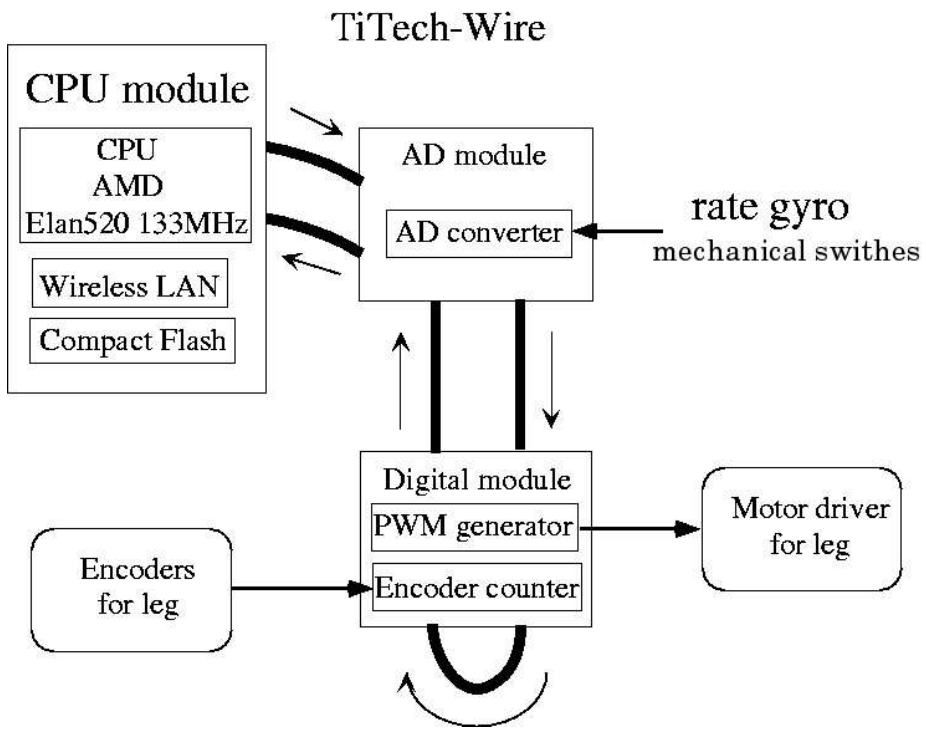


Figure 3.6: Diagram of controller for Tetsuro.

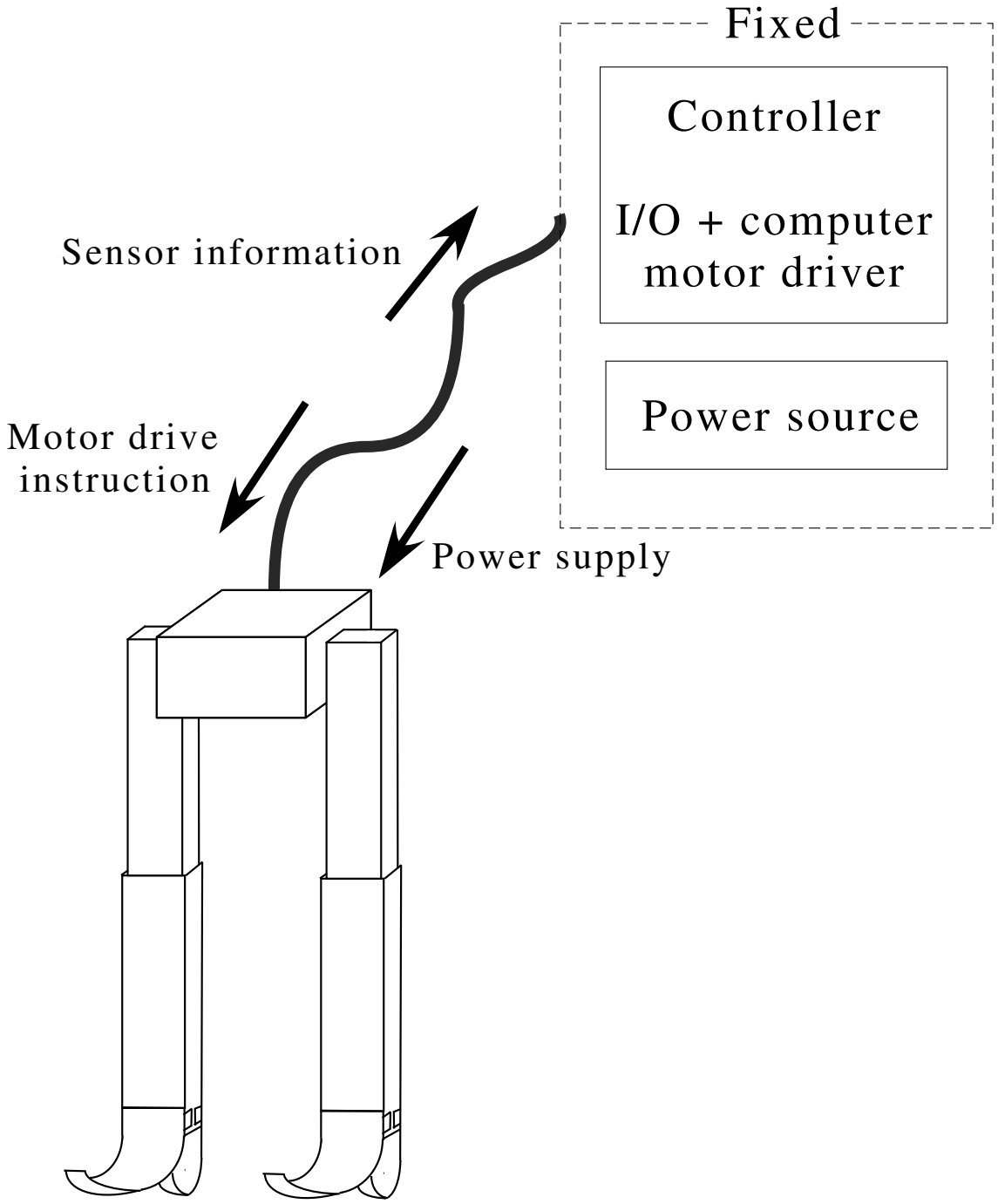


Figure 3.7: Experimental environment of Tetsuro.

3.3 胴体傾斜角度の算出法

レイトジャイロは村田製作所の圧電振動ジャイロENC-03Jを使用する．このセンサは，1個のセンサで1回転軸方向を検出する．センサ部に角速度が作用するとコリオリの法則によって，センサの振動速度に対し直角方向にコリオリ力が発生し，センサにひずみが生じて振動の軸対称性が崩れ，円板振動子の部分ごとでひずみに差が生じる．圧電素子にひずみが発生すると電荷を生じる現象(圧電効果)を利用して角速度値を測定する．これを時間積算して変位角度を求める．このセンサの感度は$0.67[mV/Deg/sec]$であるから，これよりセンサからの出力電圧を角速度に変換する．

ここで時刻tにおける角速度$\omega(t)$，変位角度$\theta(t)$ とすると角速度は

$$\omega(t) = \frac{\Delta\theta(t)}{\Delta t} \tag{3.1}$$

で与えられるので，

$$\Delta\theta(t) = \omega(t) \cdot \Delta t \tag{3.2}$$

従って，

$$\theta(t) = \sum \Delta\theta(t) = \sum \omega(t) \cdot \Delta t \tag{3.3}$$

Δtを一定時間間隔のサンプリング時間T_{cst}とすると

$$\theta(t) = T_{cst} \cdot \sum \omega(t) \tag{3.4}$$

初期条件として，計測開始時刻$t = 0$[s]での変位角度θを0[rad]と設定すれば，任意の時刻tにおける変位角度$\theta(t)$は，センサからサンプリングして得られる角速度を積算し続けることによって理論上得られる．実際上積算値はドリフトの影響を受けるので，これにローパスフィルタをかけることより胴体傾斜角度として算出している．

3.4 膝のロック機構

膝の拡大画像をFigure 3.8に示す．大腿リンクとすねリンクは膝関節でつながっており，すねリンクの上部を大腿リンクの下部に接触させることで，すねリンクがそれ以上前に出ないようにしている．この機構により，すねリンクと大腿リンクを押しつける方向に膝関節のトルクを出すことで，大腿リンクとすねリンクを直線的に保つことが可能となる(膝のロック機構)．また，足先に摂動を受けると摩擦の小さい関節の効果により膝がわずかに曲がる(バックドライバビリティ)ことで衝撃を逃がしている．すねリンクと大腿リンクの接触部分にはウレタンゲルを取り付けることで，接触時の衝撃を吸収するように工夫している．

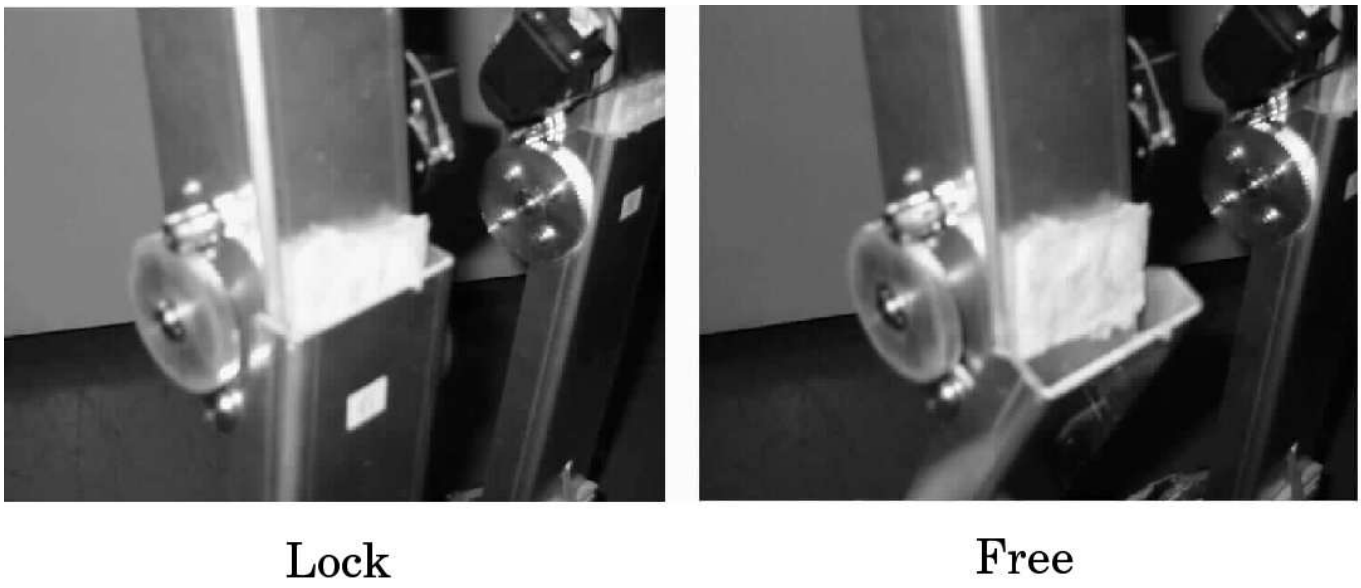


Figure 3.8: Photo of knee joint.

3.5 座標系の定義

鉄郎の座標系をFigure 3.9に示す．各軸方向の胴体傾斜角度はいずれも胴体が地面に対して水平状態にあるときをゼロとする．

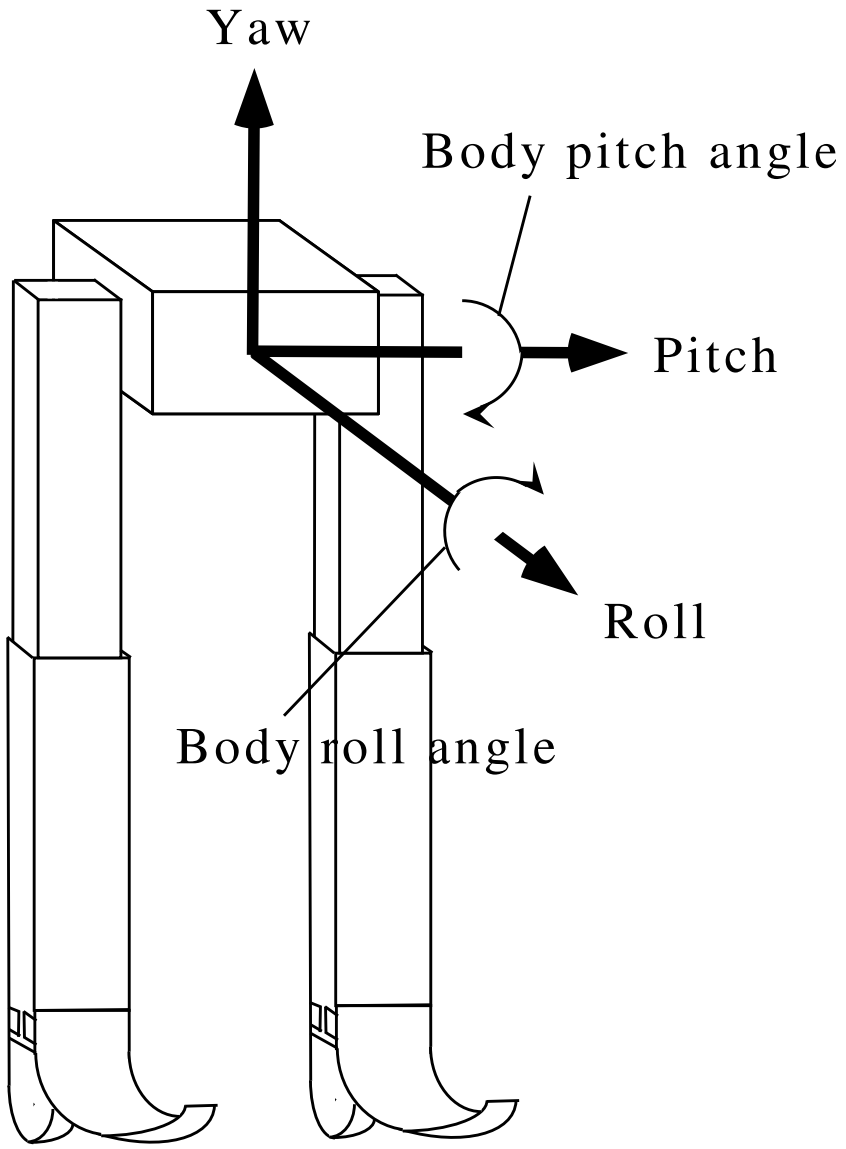


Figure 3.9: Definition of coordination for Tetsuro.

第4章　鉄郎のSplit-Belt Treadmill歩行

4.1　Tied-belt treadmill歩行実験

我々は2次元平面内二脚ロボット「鉄郎」を開発し，tied-belt条件におけるトレッドミル上において外乱下で関節PD制御・周期的な運動を行う軌道生成・stepping reflex から成る従来のリミットサイクルを構成する手法を実装した．鉄郎の機構的な設計やこの制御手法にオリジナリティはないが，以後の節で述べる実験設定や表記を説明するためにこの節を設ける．

4.1.1　PD制御

各関節におけるトルク出力は以下の式のような簡素なPD制御[1]を用いて計算される．

$$\tau_j^{ph} = -k_{j\_p}^{ph}(\theta_{j\_c}^{ph} - \theta_{j\_d}^{ph}) - k_{j\_v}^{ph}(\dot{\theta}_{j\_c}^{ph} - \dot{\theta}_{j\_d}^{ph}) \tag{4.1}$$

τ_j^{ph}，$\theta_{j\_c}^{ph}$，$\theta_{j_d}^{ph}$は，トルク出力，測定された関節角度，目標関節角度である．各関節，脚相をそれぞれ$j \in \{h, k, a\}$，$ph \in \{sw, st\}$と表す．h，k，aはそれぞれ腰，膝，足首関節を示し，sw，stは遊脚相，支持脚相を示す(Figure 4.1参照)．$\dot{\theta}_{j\_c}^{ph}$，$\dot{\theta}_{j_d}^{ph}$は測定された関節角速度，目標角速度である．$k_{j\_p}^{ph}$，$k_{j\_v}^{ph}$ はそれぞれP，Dのフィードバックゲインである．本研究において，$\dot{\theta}_{j_d}^{ph}$を0[rad/s]，$k_{j_v}^{ph}$を0.03[Nm·s/rad]に設定した．

4.1.2　Split-belt treadmill

本研究で使用されるトレッドミルは左右2つのベルトを備えており，それらはDC モータによって駆動される(Figure 4.2参照)．よって各ベルト速度を独立に制御することができる．ベルト速度は0[m/s]から0.6[m/s]まで0.005[m/s]ごとに手動で調節することができる．トレッドミルの加減速度は1.0$[m/s^2]$である．

[1]単脚支持期における遊脚の膝関節はフリーなので，ここではPD制御を用いていない．

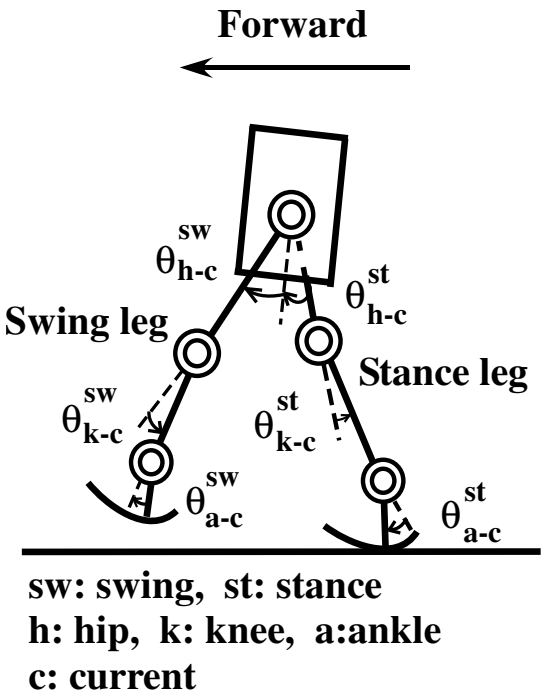


Figure 4.1: Coordinate system in the single leg stance period.

Figure 4.2: Split-Belt treadmill equipped with force sensors.

4.1.3 軌道生成と基礎実験

軌道生成

各関節角度は，絶対角ではなく相対角で測定される(Figure 4.1参照)．各関節において屈曲する回転方向が回転方向の正である．本研究において，立脚静止状態から歩き始めの3ステップまでを“transitional walking”，その後の4ステップから歩行終了までを “steady walking” と分類する．split-belt treadmill歩行においてもこの分類とする．1ステップはプログラム上ふたつの状態に分けられる．ひとつは“stance-swing state”，もうひとつは“landing-exchange state”と名付けた．各状態における期間はそれぞれT_0とΔTである．これより，1ステップの歩行周期は$T_0 + \Delta T$である．

倒立振子に基づく軌道

Figure 4.3のような2次元倒立振子モデルが，transitional walkingにおける支持脚の軌道を生成するため・すべてのステップにおける遊脚の接地角度を決定するために使われる[2]．link1の上端に鉄郎1の全質量が集中していると仮定した．足首関節トルク出力ゼロの倒立振子的な運動を生成するための運動方程式と拘束条件[3]は，以下の式で表される．

$$I_1\ddot{\phi} = m_1 g l_1 \phi \tag{4.2}$$

$$\phi(0) = -\phi(T_0) \tag{4.3}$$

$$\dot{\phi}(0) = \quad \dot{\phi}(T_0) \tag{4.4}$$

ここでI_1はlink1の関節点まわりの慣性モーメント，m_1はリンク1の質量，l_1はlink1の長さ，gは重力加速度である．steady walkingにおける1ステップ内$(0 \leq t \leq T_0)$において，これら3つの式を満たす軌道は以下の式で表される．

$$\phi(t) = (1 + e^{-aT_0})e^{at} - (1 + e^{aT_0})e^{-at}, \qquad a = \sqrt{m_1 g l_1 / I_1} \tag{4.5}$$

実験においては足首関節出力トルクをできるだけおさえ，支持脚に十分な角速度を持たせかつ重力を利用し倒立振子的な運動を行うことでナチュラルダイナミクスを利用した歩行を行う．

[2]Figure 3.3の足裏形状より実際には地面に対して支持脚の足裏自体が回転していると考えられるが，ここではあくまで近似モデルとして軌道計画を行った．

[3]ここで我々は，地面とfootの衝突において角運動量が保存される非弾性衝突を想定している．

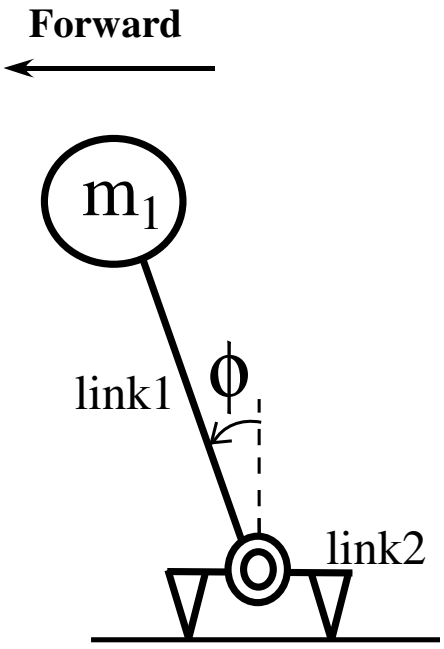


Figure 4.3: Inverted pendulum as a simplified model of Tetsuro. Although the sole of the stance leg rotates with respect to the ground in Tetsuro (Figure 3.3), we ignore such effect in this model.

Stance-swing State

stance-swing stateにおいては，単脚支持期における軌道が生成されている．stance-swing stateのイメージと各関節の目標軌道をそれぞれFigure 4.4とTable 4.1に示す．遊脚において，腰関節の目標角度は式(4.5)を用いて設定される．膝関節はナチュラルダイナミクスを利用するためにフリーにする．足首関節の目標角度は，クリアランスを確保するためのつま先上げを行うように設定される．支持脚において，腰関節の目標角度は式(4.5)を用いて設定される．腰関節はアクチュエータと膝部の機構によってロックされる(Figure 3.8参照)．足首関節の目標角度は，transitional walkingにおいては式(4.5)を用い，steady walkingにおいては一定である．これよりsteady walkingにおいて，支持脚は仮想バネ・ダンパ機構の1リンクの倒立振子となる．Stance-swing stateの期間は$0 < t \leq T_0$である．$t > T_0$となれば次頁Landing-exchange stateへ遷移する．

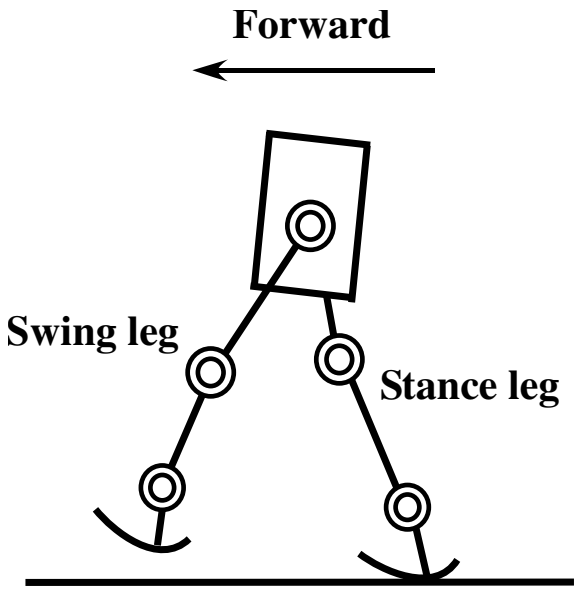


Figure 4.4: The knee joint is free to utilize natural dynamics in the swing leg. The stance leg becomes a single link inverted pendulum with virtually passive spring-dumper in the stance leg of steady walking.

Table 4.1: Desired trajectory of each joint in the stance-swing state.

Desired angles	Desired trajectories [rad]
$\theta^{sw}_{h\_d}$	$\phi(t)$
$\theta^{sw}_{k\_d}$	no control (free)
$\theta^{sw}_{a\_d}$	0.2 (toe raise)
$\theta^{st}_{h\_d}$	$-\phi(T_0)$ (moving the body forward)
$\theta^{st}_{k\_d}$	-0.07 (lock)
$\theta^{st}_{a\_d}$	$\begin{cases} \phi(t) & \text{(in transitional walking)} \\ 0 & \text{(in steady walking)} \end{cases}$

Landing-exchange State

Landing-exchange stateにおいて，振り出した遊脚は接地し，両脚支持期において遊脚と支持脚が切り替えられる(Figure 4.5参照)．Landing-exchange stateにおける各関節の目標軌道をTable 4.2に示す．遊脚が接地しその後胴体を前へ押し出すために，遊脚の後方への引きが，遊脚腰関節において計画される(Figure 4.5参照)．transitional walkingのみにおいて胴体を前へ押し出すことを補助するために,「蹴り動作」が支持脚足首関節で与えられる．

この蹴り動作における軌道は，試行錯誤により得られた．両脚の膝関節は機構的にロックされる．Landing-exchange stateの期間は$T_0 < t \leq T_0 + \Delta T$である．$t > \Delta T$となれば$t$はゼロにリセットされ，Stance-swing stateに戻る．

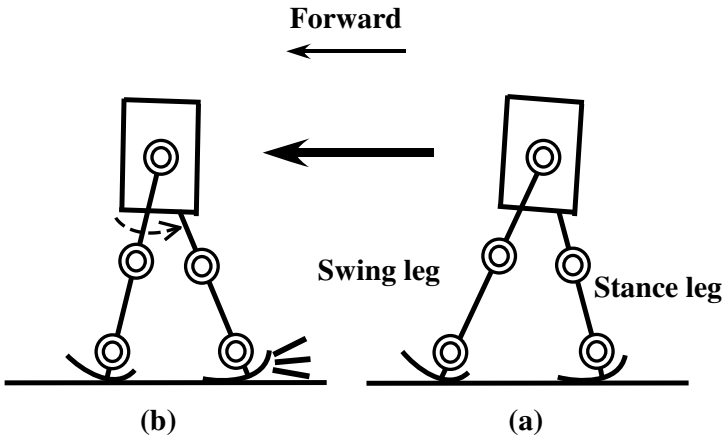


Figure 4.5: When the swing leg touches the ground (a), the double legs stance period appears. Here, we still call the forward leg as "swing leg" and the backward leg as "stance leg." The knee joints of swing and stance legs are mechanically locked. In addition, the desired trajectory as kick motion is given to the ankle joint of the stance leg (b) to help the body move forward only in the transitional walking.

Table 4.2: Desired trajectory of each joint in the landing-exchange state.

Angle at a joint in the phase	Desired trajectories [rad]
$\theta^{sw}_{h\_d}$	$-\phi(t)$ (backward)
$\theta^{sw}_{k\_d}$	-0.07 (lock)
$\theta^{sw}_{a\_d}$	$\begin{cases} \phi(t) & \text{(in transitional walking)} \\ 0 & \text{(in steady walking)} \end{cases}$
$\theta^{st}_{h\_d}$	$-\phi(T_0)$
$\theta^{st}_{k\_d}$	-0.07 (lock)
$\theta^{st}_{a\_d}$	$\begin{cases} \phi(T_0)\cos(\pi t/2T_0) & \text{(kick motion in transitional walking)} \\ 0 & \text{(in steady walking)} \end{cases}$

計画された運動と実際運動の概観

計画された運動と実際運動の時間系列をFigure 4.6に示す．我々は前頁，前々頁(Table 4.1, Table 4.2参照)で述べたようにトップダウン形式の時間系列として鉄郎の運動を計画している．しかし鉄郎の実際運動は，コントローラ・メカニズム・地面との相互作用の結果として生成される．それゆえsplit-belt treadmill歩行時において，脚のデューティ比はベルトの速度に応じて変化する．脚相(支持脚であるか遊脚であるか)は接触センサにより測定される．これより，鉄郎が単脚支持期(SLSP)なのか両脚支持期(DLSP)なのか判別される．**2.1**節で述べたように，両脚支持期は1歩行周期で2回現れる．

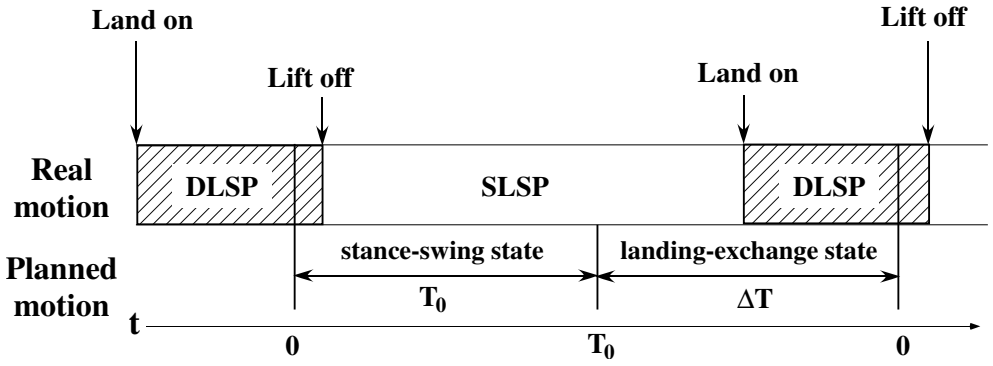


Figure 4.6: Overview of planned motion and real motion. SLSP and DLSP mean the single leg and double legs stance period, respectively. The timings of landing on and lifting off ground depend on the relation between the real motion of Tetsuro and ground. In this figure, the SLSP starts a little delayed from the start of the stance-swing state, and ends much delayed from the end of the stance-swing state. The time t in Table 4.1 and Table 4.2 is reset to zero at $t = T_0 + \Delta T$, and stance and swing legs are exchanged.

Tied-belt treadmill歩行の実験結果

上記の簡潔なPD制御と軌道計画を各関節に用いて，鉄郎はtied-belt treadmill歩行を実現した．ベルトの速度は左右ともに0.15[m/s]とした．$T_0 = 0.32$，$\Delta T = T_0$に設定した．歩行実験における鉄郎の初期状態を立脚静止とした．我々は，トレッドミルが動き始めると自動的に歩行を開始する歩行開始トリガを設けた(付録B参照)．

Figure 4.7-Aのようなスティック線図が，レイトジャイロセンサの出力により算出された胴体ピッチ角度を考慮に入れ，計算された絶対関節角度を用いて描かれる．Figure 4.8に接

地情報と測定された腰・膝関節相対角度を示す．この図より，各ステップの遊脚の腰・膝関節最大角度がそれぞれ0.2[rad]，0.6[rad]程度になっており，脚が地面に引っかからずに周期的な運動を行ったことが分かる．footの運動をFigure 4.7-Bに示す．人の通常歩行のような「かかとで接地してつま先で離地する」運動が見てとれる．実際の歩行周期，両脚支持期間，デューティ比は，それぞれおよそ1.3[s]，0.18[s]，0.72であった．tied-belt treadmill歩行の様子をFigure 4.9に示す．支持脚の膝関節が，人の通常歩行のように曲がっていないことが見られる．

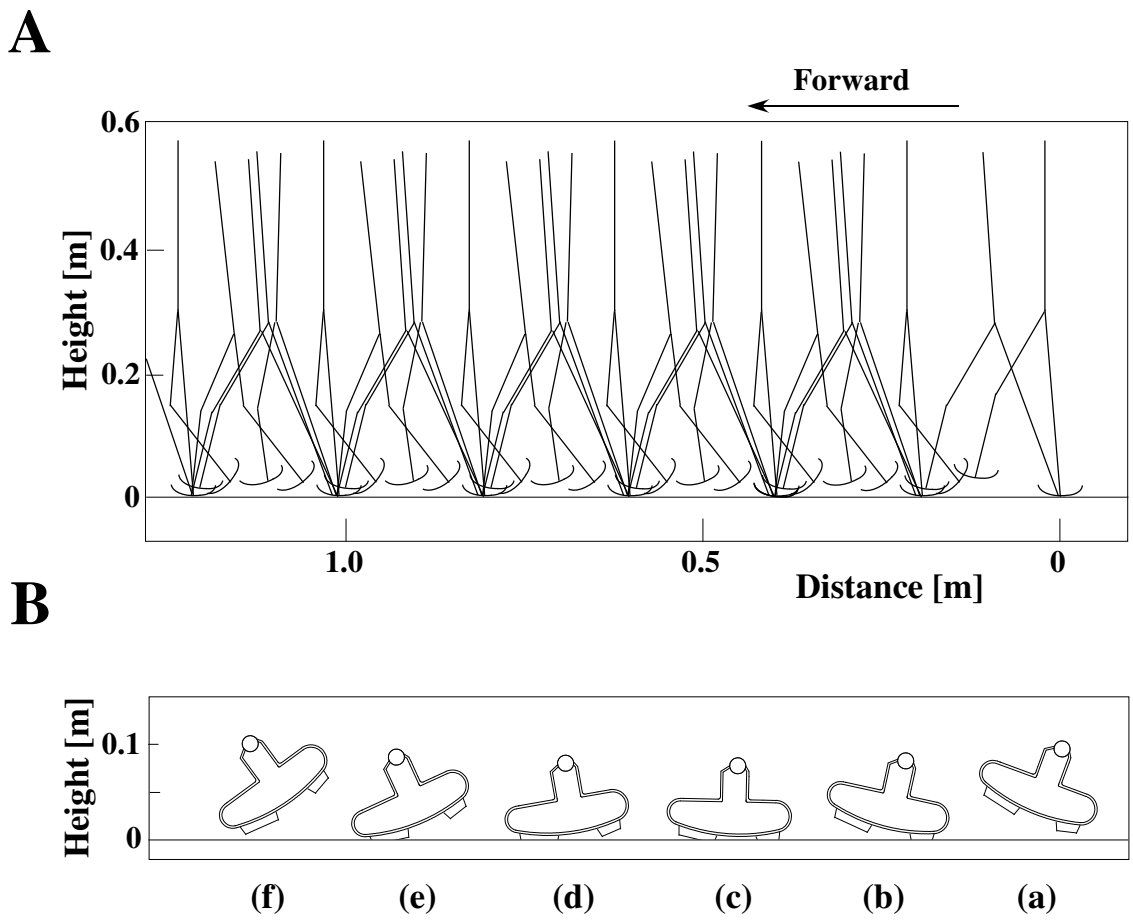


Figure 4.7: Experimental results of robot tied-belt treadmill walking with the belt speed: 0.15 m/s. The stick diagram (A), and motion of a foot (B) in landing (a), stance (b)∼(e) and liftoff (f) are shown. The distance was calculated using measured joint angles and the body pitch angle. Of course, Tetsuro moved forward or backward a little on the treadmill, and belts moved to backward.

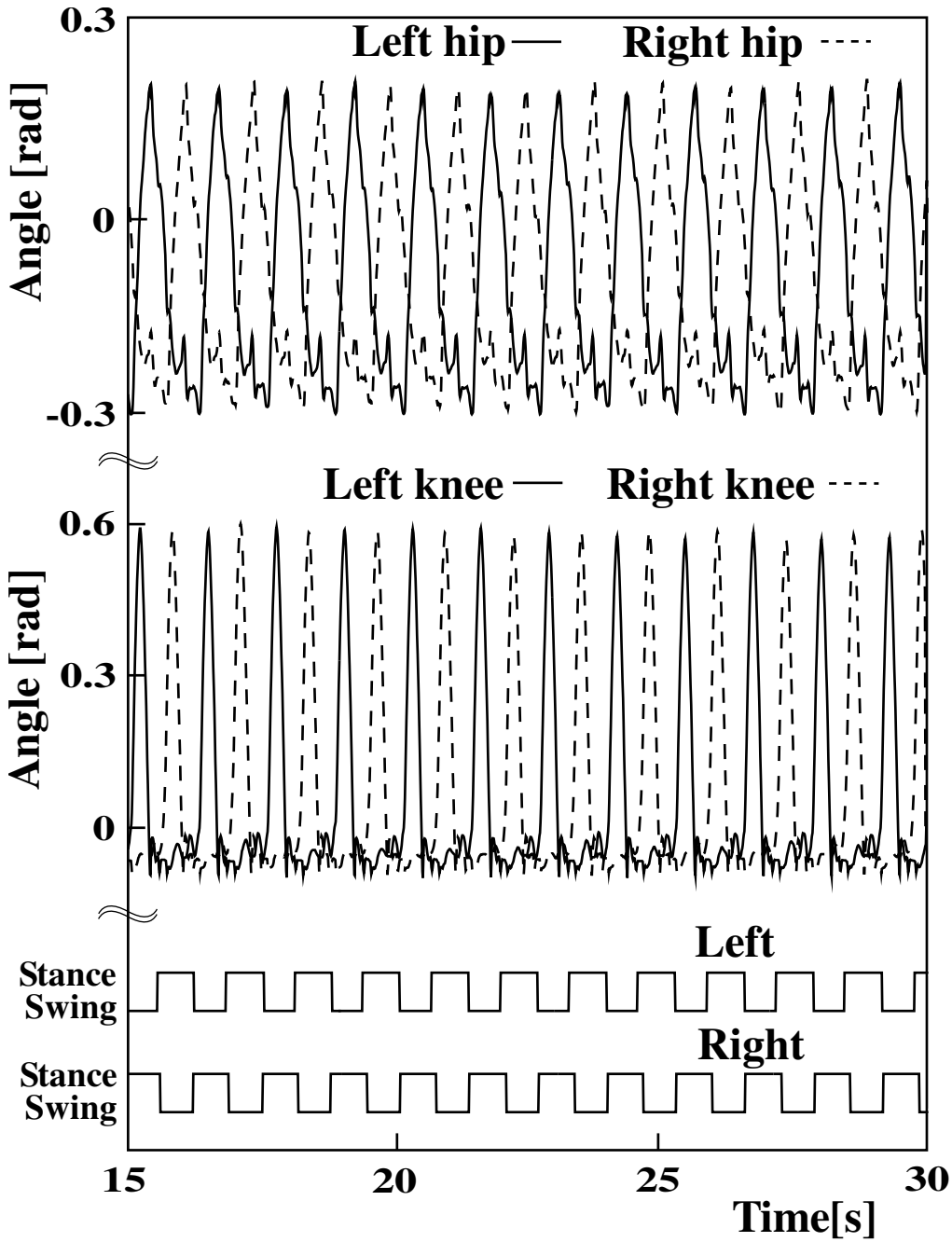


Figure 4.8: Experimental result of tied-belt treadmill walking with the belt speed : 0.20 m/s using PD control and trajectories described in Section **4.1.1** and **4.1.3**.

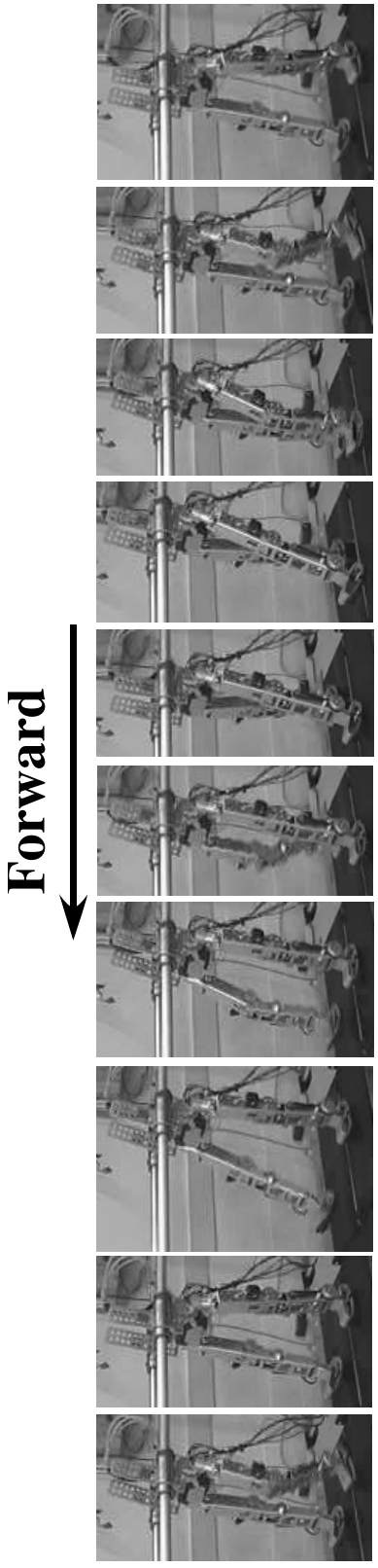


Figure 4.9: Snapshots on tied-belt treadmill walking with the belt speed: 0.15 m/s.

4.1.4 前進速度と姿勢の安定のためのstepping reflex

外乱に対して速い反応を示すinterlimbコントロールとして，我々はFigure 4.10 に示すようなstepping reflexを実装した．式(4.6)は，stance-swing state(Table 4.1参照)における遊脚腰関節の目標軌道に取って使われる．

$$\theta^{sw}_{h\_d} = \phi(t) + k_{sr} \times (\dot{\theta}^{st}_{a\_c} - \dot{\theta}^{st}_{a\_d}) \tag{4.6}$$

ここで$\phi(t)$は倒立振子モデルを用いて生成した軌道(式(4.5)参照)であり，k_{sr} はstepping reflexのゲインである．

この反射制御を用いることにより，ロボットは系の角運動量の変化に応じて遊脚の接地角度を調節することができる．また接地角度が次のステップにおける支持脚の初期角度になるので，ロボットは重力を利用した支持脚の倒立振子的な運動も調整することができる．それゆえstepping reflexは，ZMPを規範とした姿勢制御により用いられる支持脚足首関節トルク調節に比べてエネルギ消費量が少なくてすむ．この反射制御は，後ろから手で押すなどの外乱に対してロボットの前進速度や姿勢を保つことを可能にする．

三浦と下山は，竹馬型二脚ロボットを用いて初めてstepping reflexによる安定化の有効性，および，安定なStepping Reflexゲインの存在範囲を示した[Miura:1984]．本論文の実験におけるk_{sr}の値は0.33であったが[4]，この値は手動で決定された．Raibertのneutral point制御が，このような反射制御による接地角度調節の代表としてよく知られている．また，人の姿勢制御においてもこのような反射制御が重要であると報告されている[Rogers:2003]．

我々は，外乱を加えてstepping reflexの効果を確認する実験を行った．天井から1.4[m]のひもを垂らしその先に0.7[kg]のおもりを付けた振子を用意した．外乱として，鉄郎の腰関節位置における鉛直軸から30[deg.]傾けたところでおもりを放し胴体リンクに非弾性衝突させた．Figure 4.11に実験結果を示す．Figure 4.11-(A)より，左脚が支持期間である時に鉄郎が前方に押されたこと，stepping reflexの効果として左脚の足首関節角速度の増加に応じて右脚の接地角度が増加したことが確認できる．鉄郎はこのような外乱に適応し前進速度や姿勢を保つ．

[4]不適切なゲイン(例えば0.27)に設定すると，鉄郎は外乱に対して容易に転倒する．

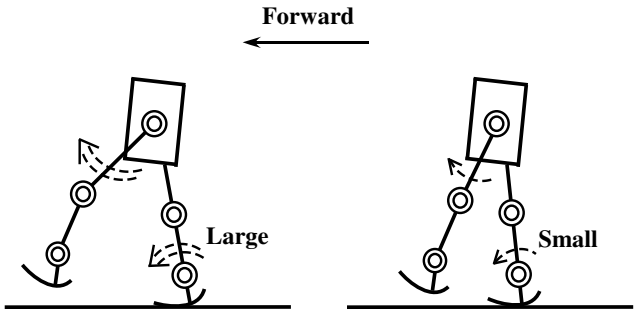


Figure 4.10: Stepping reflex. The robot changes the touchdown angle of the swing leg according to the ankle joint angular velocity of the stance leg.

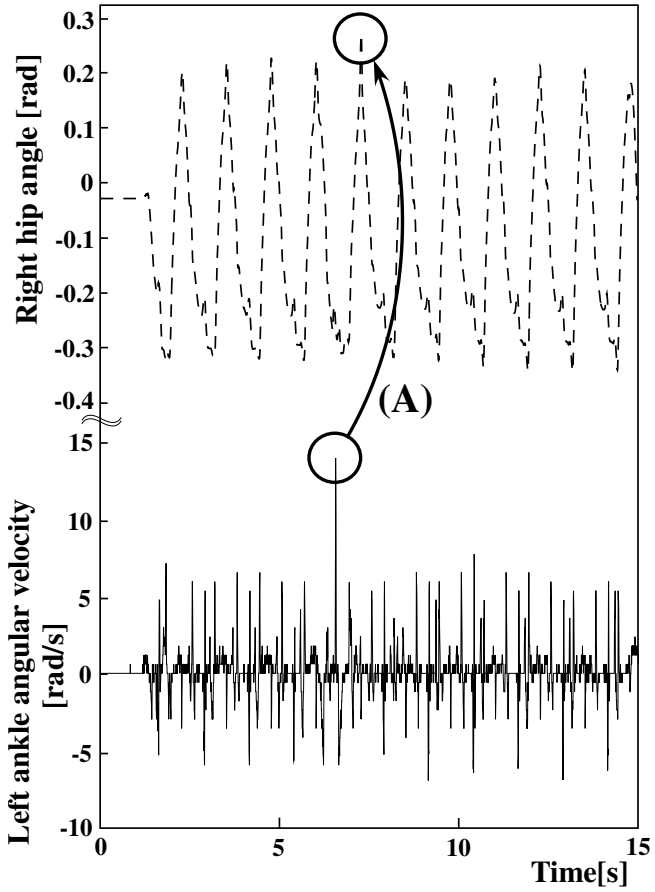


Figure 4.11: Experimental result in tied-belt treadmill walking with the belt speed: 0.20 m/s to see the effectiveness of a stepping reflex against disturbance.

4.2 Split-belt treadmill歩行実験

鉄郎を用いた実験の時間設定は人の実験(Figure 2.1)と同様にした．トレッドミルの速度は“遅いモード”(0.15[m/s])か“速いモード”(0.30[m/s])とした．各ステージの期間は15[s]とした．

4.2.1 歩容適応のためのPゲイン調節

Split-belt treadmill歩行における感覚的なintralimbコントロールとして，支持脚腰関節においてPゲインの調節を行う制御を実装した[5]．Pゲイン調節は各脚独立に実装されるので，それぞれの脚のステップ数を分けて数え，以下の式(4.7)を各脚独立に適用する．Figure 4.12より，ψ_{off}は離地時における後脚の腰関節相対角度である．nステップ目としての次ステップの支持脚相における腰関節のPゲインは，以下の式を用いてadaptationとpost-adaptationステージにおいてstep-by-stepに更新される．

$$k^{st}_{h\_p}[n] = k_{ag} \times (\bar{\psi}_{off} - \psi_{off}[n-1]) + k^{st}_{h\_p}[n-1] \tag{4.7}$$

ここでk_{ag}はPゲイン調節を行うためのゲイン，$\bar{\psi}_{off}$はbaselineステージにおけるψ_{off}の平均値，$k^{st}_{h_p}[n]$はnステップ目の支持脚相における腰関節Pゲインである．$\bar{\psi}_{off}$と1ステップ前のψ_{off}を基に次ステップ支持脚のPゲインの値を更新するので，この調節制御にはフィードフォワードの要素が含まれる．

Pゲイン調節がロボットのsplit-belt treadmill歩行を可能にした理由をFigure 4.13に示す．片方のベルト速度が速くなるadaptationステージにおいて，ベルトによりfast legが支持脚相で後ろへ引っ張られるので，$|\psi_{off}|$が大きくなる(Figure 4.13-(a)参照)．そして次のfast legの支持脚相において，式(4.7)のPゲイン調節によりfast legの腰関節の剛性が低下し，fast legはより後ろへ引っ張られる(Figure 4.13-(b)参照)．これより，fast legの足首関節角速度がより速くなるので，式(4.6)のstepping reflexより遊脚着地角度が大きくなる．その結果，ワールド座標系における胴体の位置がほぼ一定に保たれ，ロボットはsplit-belt treadmill上で歩行を継続することができる．

[5] このPゲイン調節を用いた平地における実験はまだ行われていない．

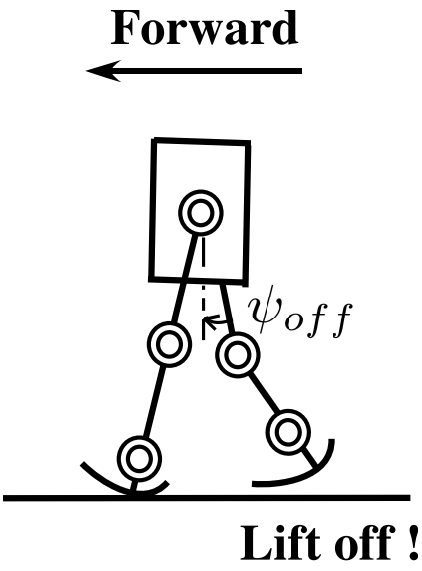


Figure 4.12: Definition of ψ_{off}.

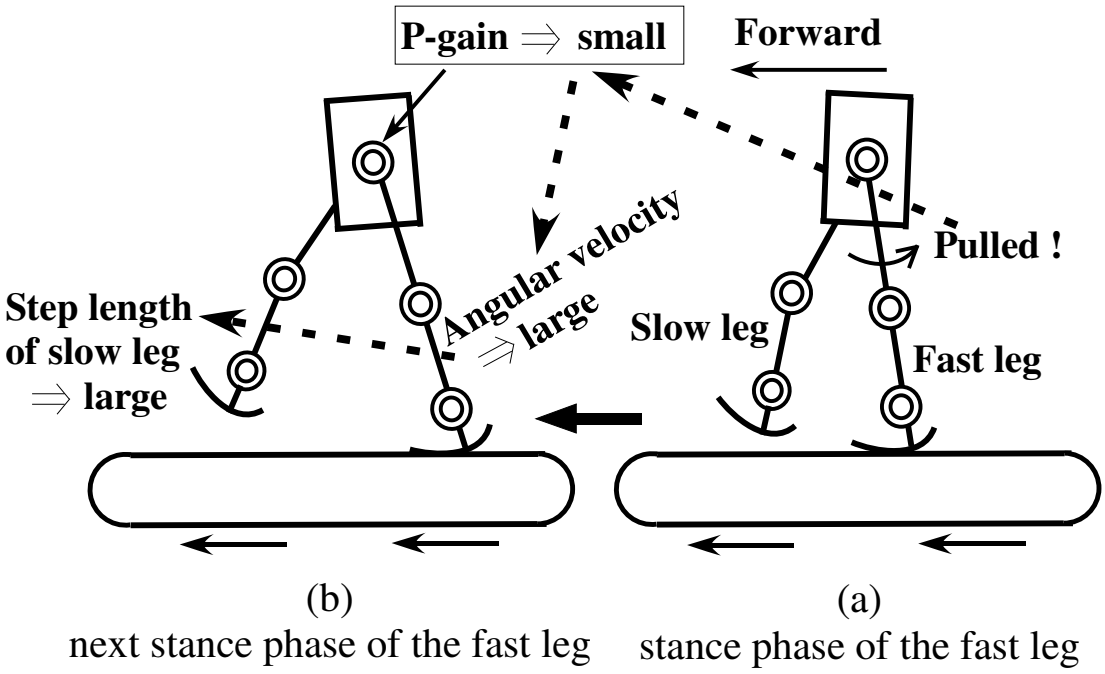


Figure 4.13: P-gain adjustment for split-belt treadmill walking.

4.2.2 健常者モデル

上記の実験設定に**4.2.1**節で述べたPゲイン調節を加え，鉄郎をsplit-belt treadmill上で歩行させた．歩行の様子をFigure 4.20に示す．Figure 4.20-(A)より，支持脚相の左脚(fast leg)がadaptationステージにおいてFigure 4.9と比べてベルトにより後ろへ引っ張られていることが分かる．鉄郎はbaselineステージにおいて$\bar{\psi}_{off}$を計算する．また，ベルトの速度変化による離地のタイミングの変化を検出することで自律的にbaselineステージからadaptationステージへの切り替わりを認識し，式(4.7)で表されるPゲイン調節をスタートさせる．Pゲイン調節は，post-adaptationステージが終了するまでadaptationステージから引き続き用いられる．

ストライド長とデューティ比について

健常者のsplit-belt treadmill歩行実験において計測されたストライド長とデューティ比を再度Figure 4.14に示す．ロボット(健常者モデル)実験から測定されたストライド長とデューティ比をFigure 4.15に示す．これらの図のプロット点は，歩行 1 周期において各々計測した指標である．ロボット実験におけるデータ(Figure 4.15)は，ストライド長を除いて[6] 健常者の実験におけるデータに近い値とパターンを示した．

ステップ長と両脚支持期間比の差について

健常者のsplit-belt treadmill歩行実験において計測されたステップ長と両脚支持期間比の差を再度Figure 4.16に示す．ロボット(健常者モデル)実験から測定されたステップ長と両脚支持期間比の差をFigure 4.17に示す．ロボット実験におけるデータ(Figure 4.17)は，ステップ長の差を除いて健常者の実験におけるデータに近い値とパターンを示した．

Split-belt treadmill歩行における両脚のψ_{off}と$k^{st}_{h_p}$をFigure 4.18に示す．支持脚相におけるfast legの腰関節Pゲイン$k^{st}_{h_p}$が，adaptationステージにおいて式(4.7)により徐々に低下していることが分かる．その結果低下したfast legの$k^{st}_{h_p}$は，ステップ長と両脚支持期間比の差においてFigure 4.17に示すようなpost-adaptationステージの始めに起きる後遺症を引き起こした．adaptationステージにおけるfast legの腰関節目標軌道と実軌道をFigure 4.19に示す．Pゲイン調節によりphysicに支持脚がより後ろに引っ張られていることが見られる．

[6]人とロボットにおけるストライド長とステップ長差の値の違いは，人と鉄郎の脚長の違いによるものである．

これより鉄郎(健常者モデル)と人(健常者)は，split-belt treadmill上のこれらの指標においてよく似た変化パターンを示した．

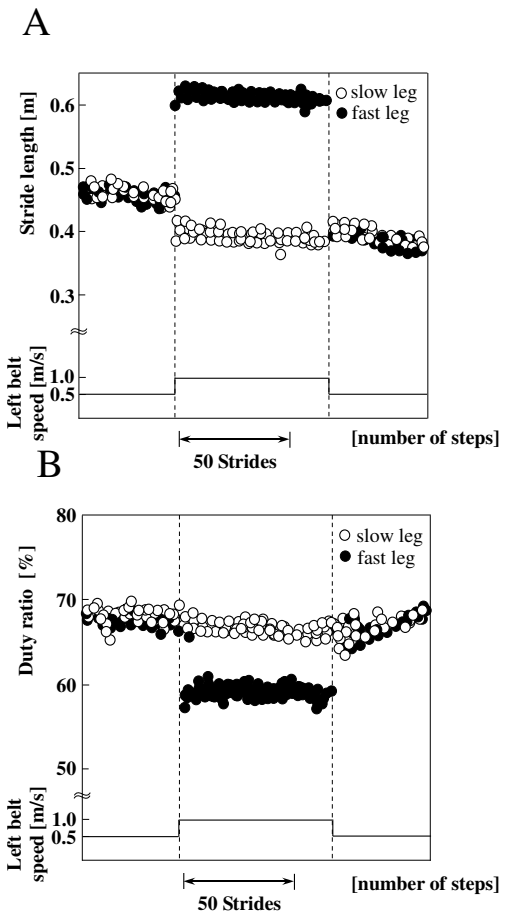


Figure 4.14: The stride length (A) and the duty ratio (B) in normal subject split-belt treadmill walking are shown, where speed of belts were 0.5 m/s at both left and right belts in the baseline stage, 0.5 m/s at the left belt and 1.0 m/s at the right belt in the adaptation stage, and 0.5 m/s at both left and right belts in the post-adaptation stage. Speed of the fast belt is also shown (modified from Morton et al. 2006 [Morton:2006]).

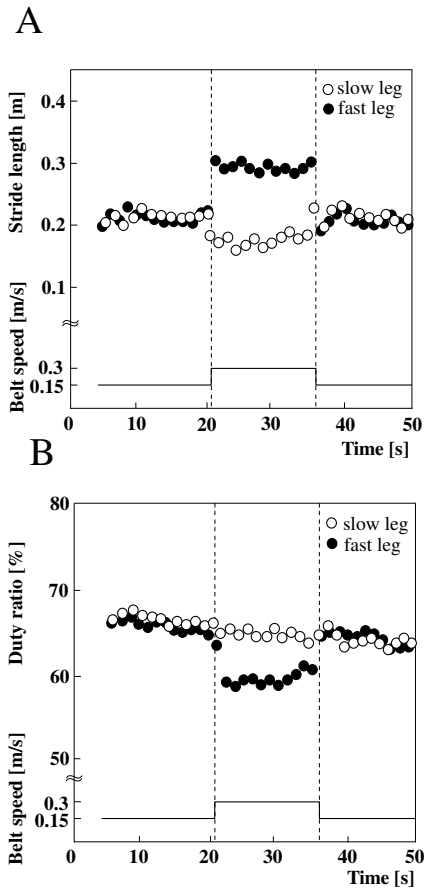


Figure 4.15: The stride length:(A) and the duty ratio:(B) in split-belt treadmill walking of normal subject model are shown, where speed of belts were 0.15 m/s at both right and left belts in the baseline stage, 0.15 m/s at the right belt and 0.30 m/s at the left belt in the adaptation stage, and 0.15 m/s at both right and left belts in the post-adaptation stage. Speed of the fast belt is also shown.

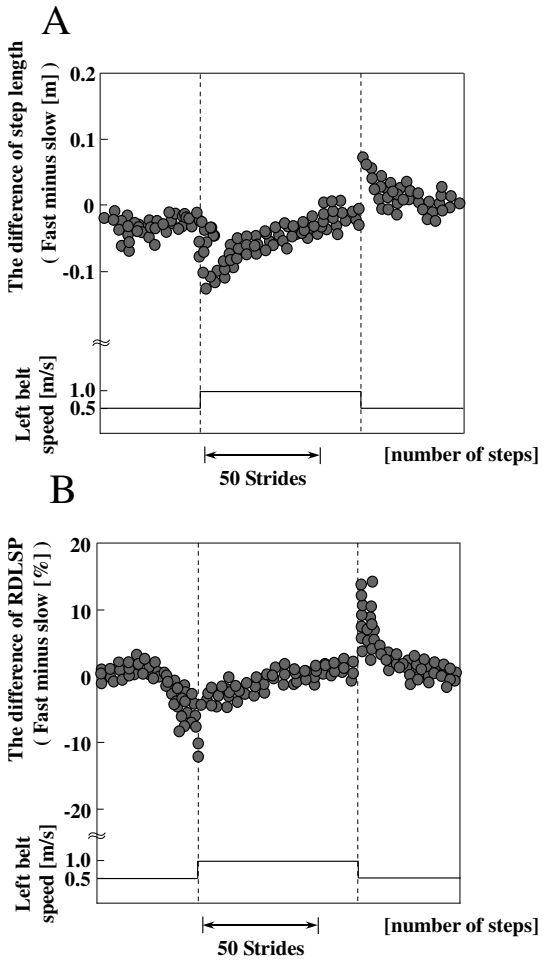


Figure 4.16: The step length difference (A) and RDLSP (ratio of the double legs stance period) difference (B) in normal subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006 [Morton:2006]).

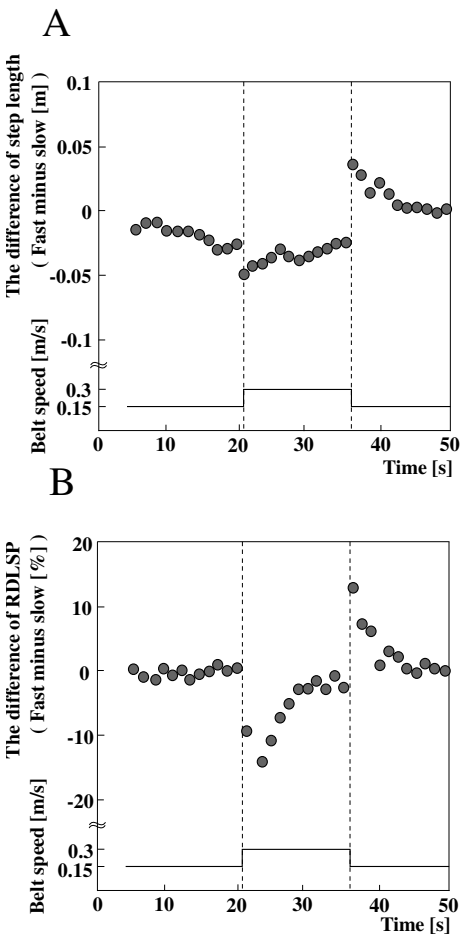


Figure 4.17: The step length difference (A) and RDLSP (ratio of the double legs stance period) difference (B) in split-belt treadmill walking of normal subject model are shown. The time course of belts speed is described in the caption of Figure 4.15.

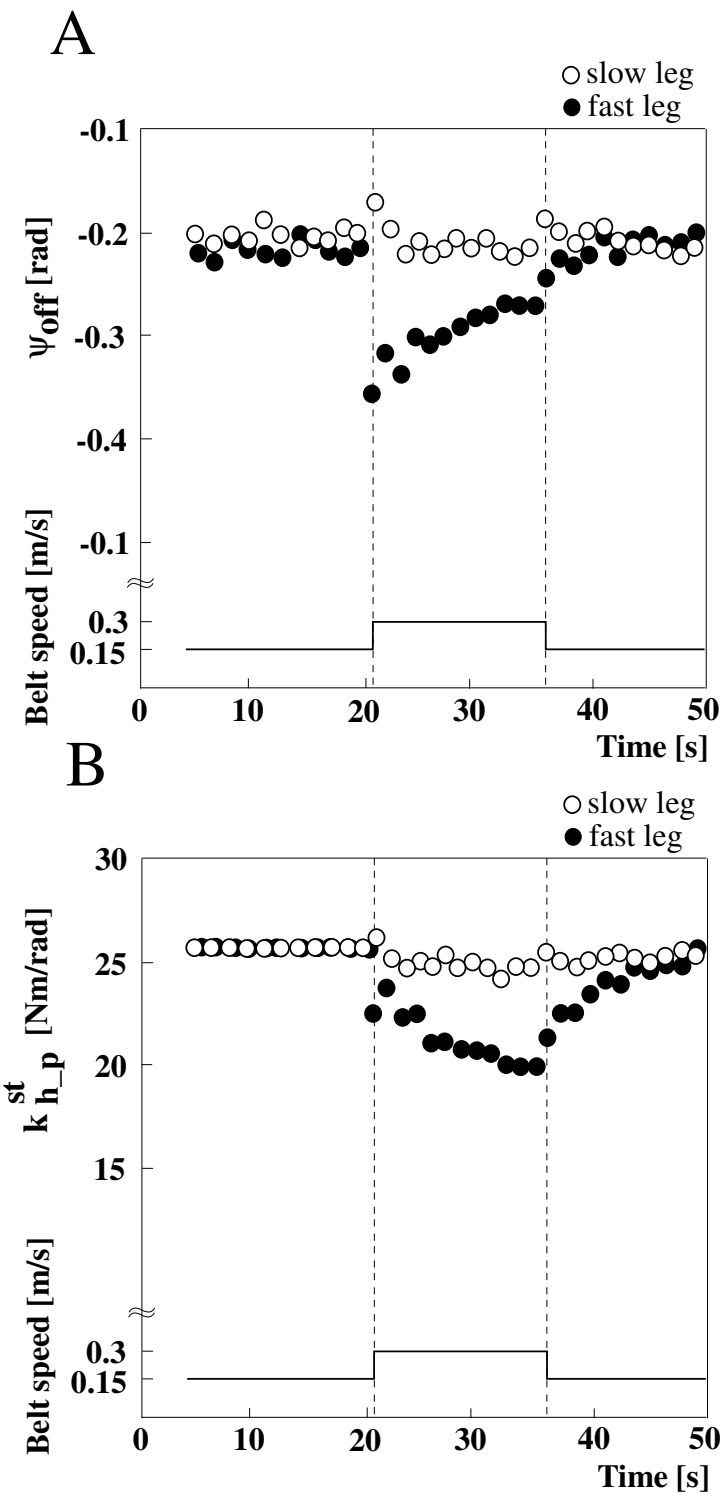


Figure 4.18: The ψ_{off}:(A) and the hip joint p-gain in the stance phase $k^{st}_{h\_p}$:(B) of both fast and slow legs in split-belt treadmill walking of normal subject model are shown. The time course of belts speed is described in the caption of Figure 4.15.

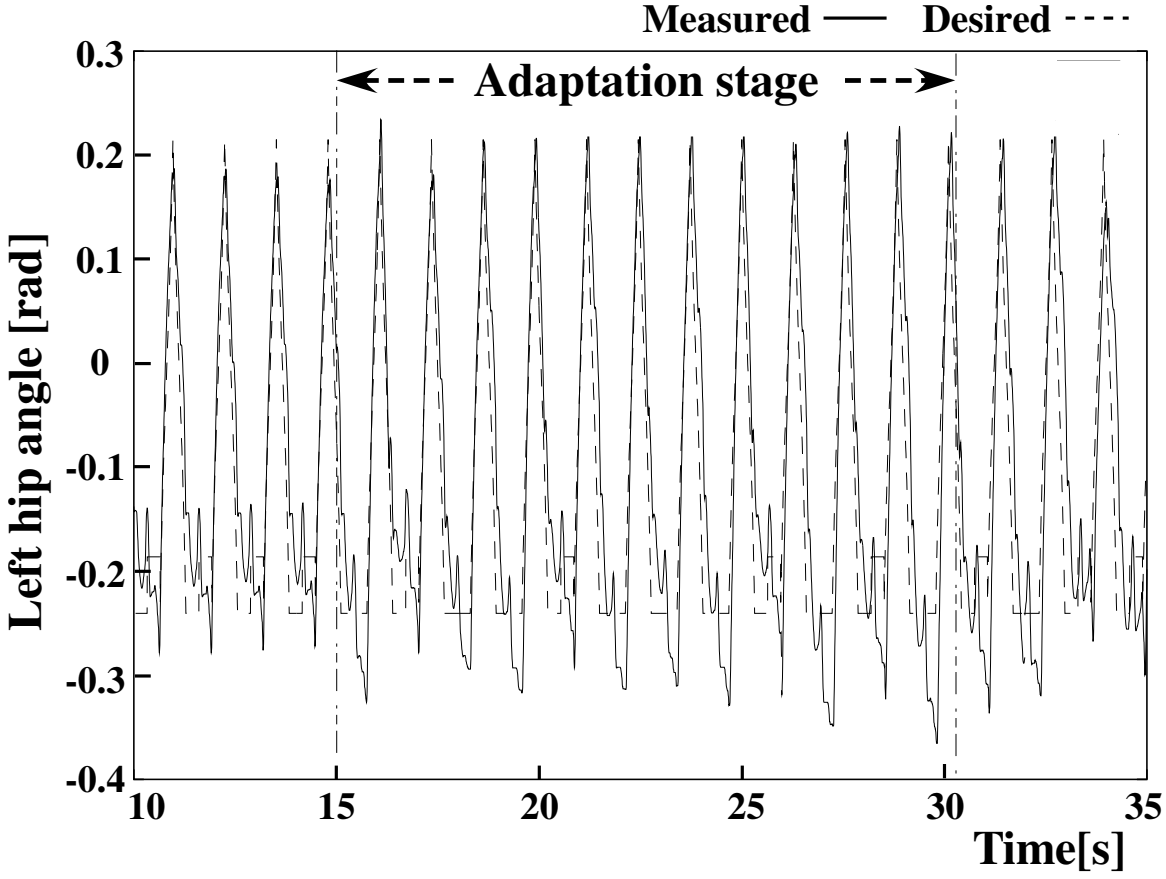


Figure 4.19: The measured and desired hip joint angle of the fast leg in the adaptation stage of split-belt configuration.

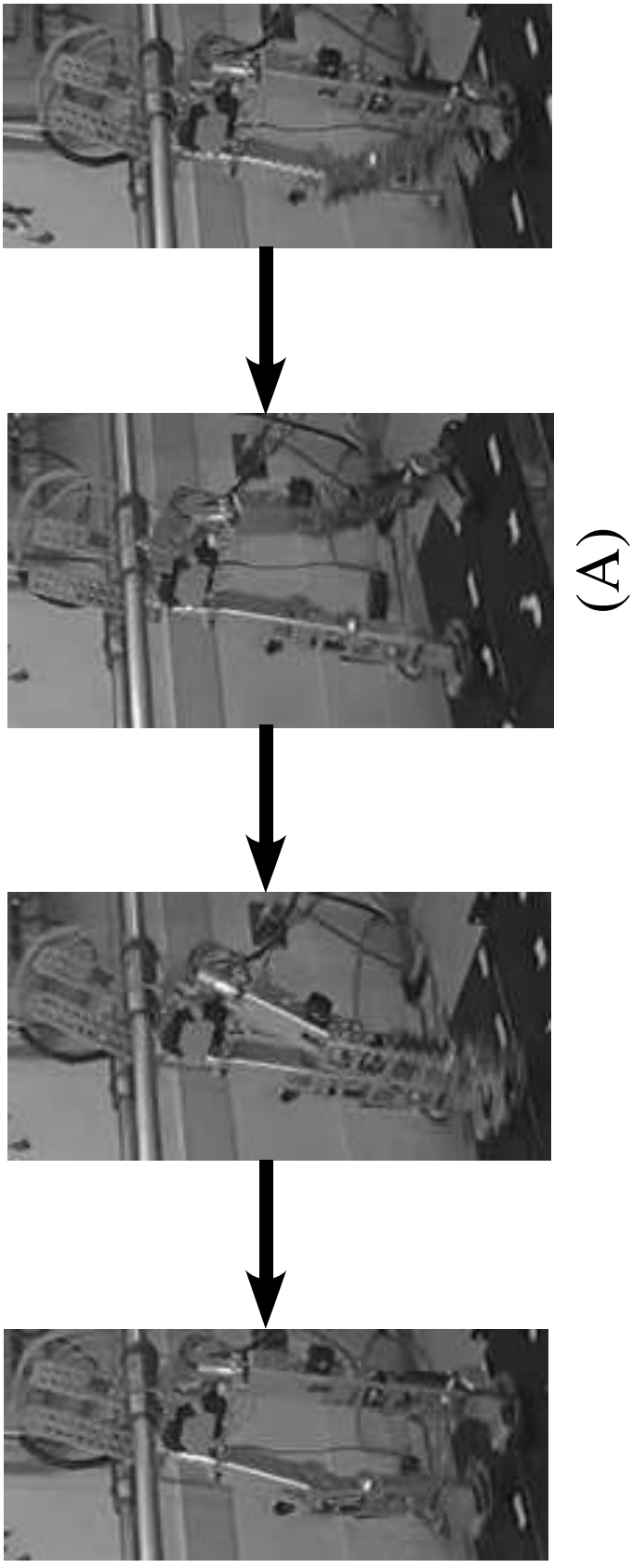


Figure 4.20: Snapshots on Tetsuro (normal subject model) split-belt treadmill walking in the adaptation stage.

4.2.3 小脳疾患患者モデル

Pゲイン調節の効果を確認しつつsplit-belt treadmill上における小脳疾患患者モデルを構成するために，Pゲイン調節を用いずに関節PD制御・周期的な運動を行う軌道生成・stepping reflexのみを用いて鉄郎をsplit-belt treadmill上で歩行させた．Pゲイン調節なしの鉄郎(小脳疾患患者モデル)はsplit-belt treadmill歩行を実現したが，歩行成功確率はPゲイン調節ありの鉄郎(健常者モデル)よりも低下した．加えて，ワールド座標系における胴体の位置が前後により動くようになった．

ストライド長とデューティ比について

小脳疾患患者のsplit-belt treadmill歩行実験において計測されたストライド長とデューティ比をFigure 4.21に示す．ロボット(小脳疾患患者モデル)実験から測定されたストライド長とデューティ比をFigure 4.22に示す．ロボット実験におけるデータ(Figure 4.22)は，小脳疾患患者の実験におけるデータに近いパターンを示した．

ステップ長と両脚支持期間比の差について

小脳疾患患者のsplit-belt treadmill歩行実験において計測されたステップ長と両脚支持期間比をFigure 4.23に示すロボット(小脳疾患患者モデル)実験から測定されたステップ長と両脚支持期間比の差をFigure 4.24に示す．ロボット実験におけるデータ(Figure 4.24)は，小脳疾患患者の実験におけるデータに近いパターンを示した．特に，adaptationステージ内の徐々に元の値に戻るパターンやpost-adaptation ステージの始めに起きる後遺症が，ロボット(小脳疾患患者モデル)実験においても見られないことが分かった．

これより鉄郎(小脳疾患患者モデル)と人(小脳疾患患者)は，split-belt treadmill上のこれらの指標においてよく似た変化パターンを示した．

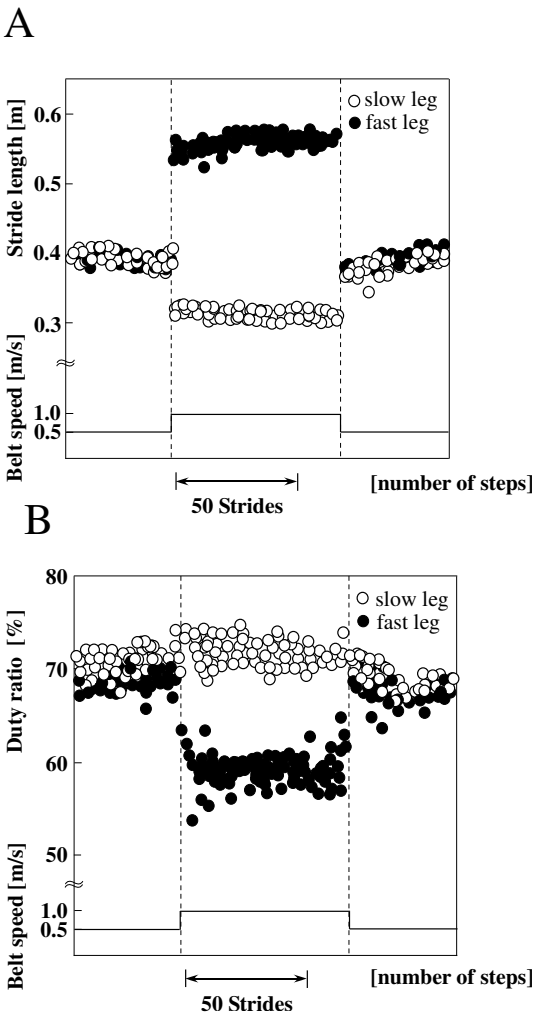


Figure 4.21: The stride length (A) and the duty ratio (B) in cerebellar disease subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006 [Morton:2006]).

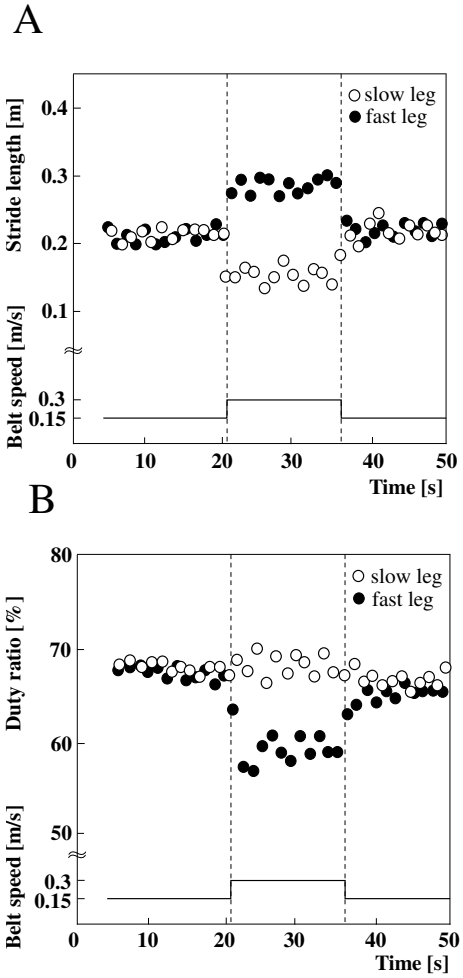


Figure 4.22: The stride length:(A) and the duty ratio:(B) in split-belt treadmill walking of cerebellar disease subject model are shown. The time course of belts speed is described in the caption of Figure 4.15.

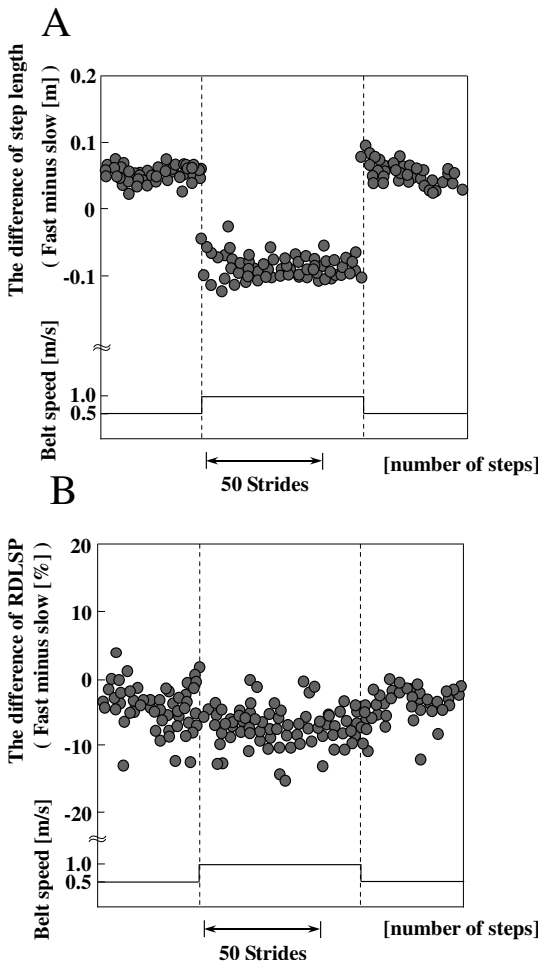


Figure 4.23: The step length difference (A) and RDLSP difference (B) in cerebellar disease subject split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 2.3 (modified from Morton et al. 2006[Morton:2006]).

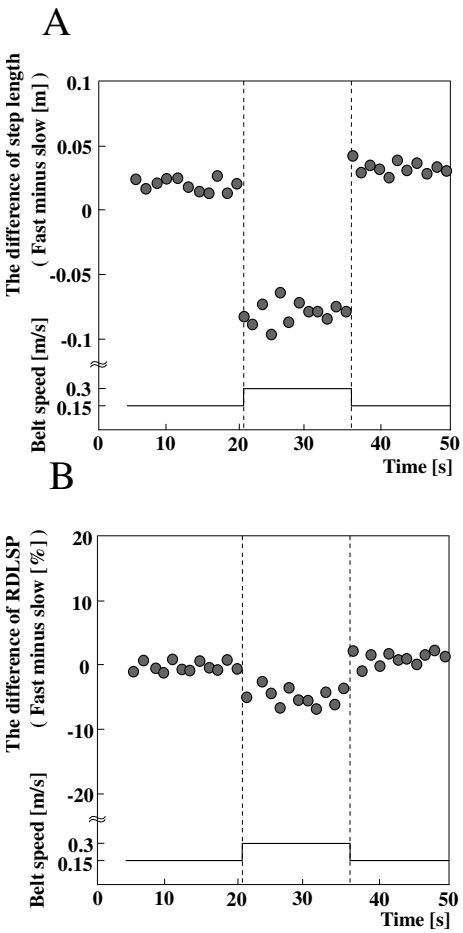


Figure 4.24: The step length difference (A) and RDLSP difference (B) in split-belt treadmill walking of cerebellar disease subject model are shown. The time course of belts speed is described in the caption of Figure 4.15.

第5章　考察

5.1　機構の特性による制御手法への依存度

鉄郎各脚の腰・足首関節トルク出力に注目する．Figure 5.1にtied-belt条件における出力されたトルクパターンを示す．図中より，これらの出力トルクの平均偏差はそれぞれおよそ2.0[Nm]，0.5[Nm]となっている．腰関節トルクは，ナチュラルダイナミクスを利用した(式(4.5))とstepping reflex(式(4.6))により出力されたトルクである．1.2.1節より，足首関節トルク出力はCOPすなわちZMPを操作している．腰・足首関節トルクは，それぞれリミットサイクルを構成する手法とZMP規範型制御手法により出力されたエネルギと考える．よってこのエネルギ配分は，鉄郎のこれら二種類の制御手法への結果としての依存度を表すと仮定する．実験結果より，鉄郎は4:1の割合でそれぞれの制御手法に依存していた．ナチュラルダイナミクスを考慮し効率的な歩行を実現している鉄郎[Otoda:2008]にとって，この比率は妥当である．ZMP規範型制御手法に代表されるASIMO・HRP2は，鉄郎に比べ高出力可能な足首関節を持つ．高出力可能な足首関節を搭載するためには，大きなアクチュエータと減速比が必要となり足首部の質量が大きくなる．足首部の質量が大きくなると脚の慣性モーメントが大きくなるので，腰関節のアクチュエータ・減速比も大きくなる．腰関節のアクチュエータ・減速比が大きくなると，腰部の粘性摩擦が大きくなり機構的な制御遅れのためstepping reflexなどのリミットサイクルを構成する手法を用いにくい．よってASIMO・HRP-2はZMP規範型制御手法への依存度が高い[1]．また竹馬型二足歩行ロボットBiper[Miura:1984]やRaibertのホッピングロボット[Raibert:1986]など，足首関節がない二足ロボットにいたっては足首関節トルク出力によるZMPの操作が不可能なので，リミットサイクルを構成する手法のみへの依存となる．これより，足首関節があるか否か，アクチュエータ軸やギアなどの粘性摩擦，遊脚の慣性モーメントなどの機構的な特性が，これら2種類の制御手法への依存度を決めると考える．

[1]ASIMOはZMP規範型制御だけでなく，stepping reflexと等価な着地位置制御を行っている．

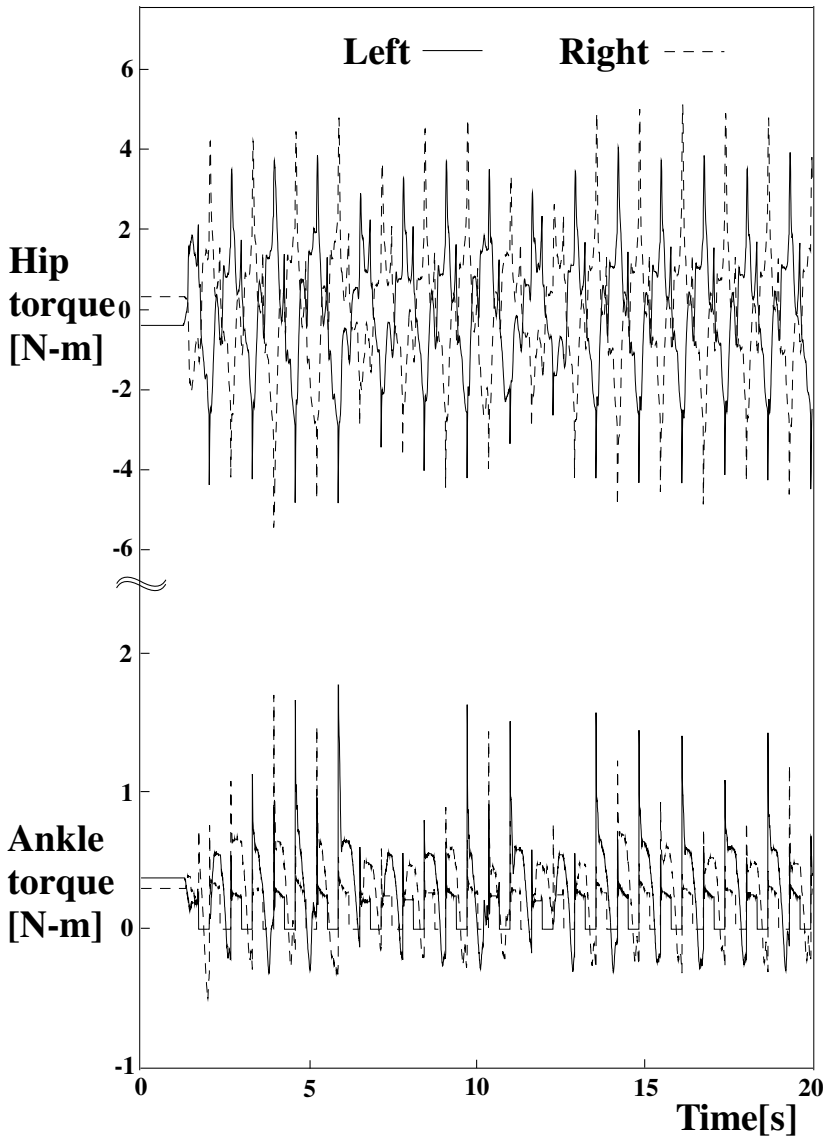


Figure 5.1: Torque pattern of Tetsuro in tied-belt treadmill walking with the belt speed: 0.20 m/s.

5.2 歩容適応モデル

Bastianら[Reisman:2005, Morton:2006]は，脊髄・脳幹・小脳・運動皮質を含んだ神経構造がさまざまな運動適応の制御を担っていると示唆している．しかしながら彼らは，どの神経構造がどのような種類の調節メカニズムによりどの適応機構に貢献しているかは明確に知られていないと言及した．そこで我々は，split-belt treadmill上の二脚歩行に対する調節メカニズムを提案し，健常者や小脳疾患患者の歩容適応モデルを構成し，2次元二脚歩行ロボットを用いて構成したモデルの正当性を検証した．ロボット実験におけるデータが人の実験におけるデータに近いパターンを示したので，我々の構成したモデルが二脚歩行における人の神経構造にある程度近いのではないかと考える．

2.4 節で述べたようにBastianら[Reisman:2005, Morton:2006]は，以下のような調節機構が人のsplit-belt treadmill歩行において存在すると示唆している．

(a) 脊髄や脳幹における感覚的フィードバック適応機構(ストライド長やデューティ比を調節する*intralimb coordination*に相当)．

(b) 小脳における予見的フィードフォワード適応機構(ステップ長や両脚支持期間比の差を調節する*interlimb coordination*に相当)．

それに対して我々の提案したモデルにおける適応機構(Figure 2.8参照)は，このような調節機構と異なり以下のように要約される．

[a] デューティ比はおおむね運動学的拘束により受動的に調節される[2]．

[b] stepping reflexは*interlimb*コントロールであるにもかかわらず，*intralimb index*(ストライド長)を調節する(脳幹における感覚的フィードバック適応機構に相当)．

[c] Pゲイン調節は*intralimb*コントロールであるにもかかわらず，stepping reflex (*interlimb* コントロール) と組み合わさり*interlimb indexes*(ステップ長と両脚支持期間比の差)を調節する(小脳における感覚的フィードフォワード適応機構[3]に相当)．

人とロボットの実験の間で測定された指標の比率とパターンの高い類似性は，我々の仮説と提案したモデルが正当であることをほのめかしている．我々は，支持脚腰関節における

[2]支持脚のPEP(post exterior position)により脚相がリセットされるCPG[Aoi:2005]を実装すれば，デューティ比はBastianらが指摘するように感覚的に調節される．

[3]Pゲイン調節はbaseline ステージにおいて算出された$\bar{\psi}_{off}$を必要とする．そういう意味では目標指向型[Giese:2007]であるといえる．

Pゲインがsplit-belt treadmill歩行における対称性を有する歩容適応の制御パラメータであることを示した．しかし**4.2.1**節で述べたように，我々の構成したモデルにおいてPゲイン調節はstepping reflexと組み合わさって初めて機能するので，式(4.7)のk_{ag}が式(4.6)のk_{sr}に関連して決定されるのは，注目すべきことである．また我々は生物学的知見と実機実験による試行錯誤に基づきsplit-belt歩容適応モデルを構成したが，本研究とは異なるアプローチで(例えば筋骨格系モデルを用いた歩行シミュレーション)split-belt歩容適応に取り組めば他の解は十分に存在しうると考える．

5.3 エネルギ効率の評価

人間，歩行・走行ロボット，車など各種移動体におけるエネルギ効率の比較をFigure 5.2に示す．本論文では，Specific energetic cost of transport(C_{et})とSpecific mechanical cost of transport(C_{mt})を用いてエネルギ効率を評価した[Collins:2005-1, Collins:2005-2]．Figure 5.2の縦軸は(C_{et})である．C_{et}とC_{mt}は以下の式で表せる．

$$C_{et} = \frac{P_e}{mgs} \tag{5.1}$$

$$C_{mt} = \frac{P_m}{mgs} \tag{5.2}$$

ここでm[kg]は移動体の質量，gは重力加速度，s[m]は移動距離，P[W] は消費仕事量である．P_eは総歩行時間における消費エネルギである．P_mは総歩行時間における機械的エネルギ損失である．P_eとP_mは以下の式で表せる．

$$P_e = \int_0^{T_{wt}} \sum_{i=1}^{n} (R_i I_i^2 + \ |\dot{\theta}_i \tau_i|) dt \tag{5.3}$$

$$P_m = \int_0^{T_{wt}} \sum_{i=1}^{n} |\dot{\theta}_i \tau_i| dt \tag{5.4}$$

機械設計とアクチュエータの消費効率の効果を区別するためにC_{et}とC_{mt} を用いた．ここでR_iはアクチュエータの端子間抵抗，I_iはアクチュエータに流れる電流, τ_iはアクチュエータにおけるトルク出力，$\dot{\theta}_i$はアクチュエータに対応する関節の角速度，nはアクチュエータの総数，T_{wt}は総歩行時間である．ここでは電流I_iを以下の式で算出した．

$$I_i = \frac{\tau_i}{\text{トルク定数} \times \text{減速比}} \tag{5.5}$$

算出した鉄郎のC_{et}とC_{mt}をTable 5.1に示す．これにより，移動体の大きさ，移動速度に無関係に消費仕事量の比較ができる．算出した値は人の歩行の抵抗率に比較的近く，本研究で用いた制御手法のエネルギ効率のよさがうかがえる．ただ現時点の鉄郎は胴体がないので，今後胴体を付けた上でエネルギ効率の評価を行い，人の歩行とどの程度差があるのか確認する予定である．またこの図より，脚ロボットはエネルギ効率の点で人や自動車に劣っていることが分かる．唯一勝っているGravity walkerがあるが，これは**1.2.2** 節で紹介したアクチュエータなしで坂を下る歩行機械である．いずれにしても，広く用いられるロボットを開発するためにはエネルギ効率の観点が必要不可欠である．

Table 5.1: Specific const of transport and Specific mechanical cost of transport for selected land vehicles.

Mobile objects	C_{et}	C_{mt}
Human walking	0.2	0.05
ASIMO	3.2	1.6
Tetsuro(tied-belt, split-belt)	0.85, 0.82	0.10, 0.11

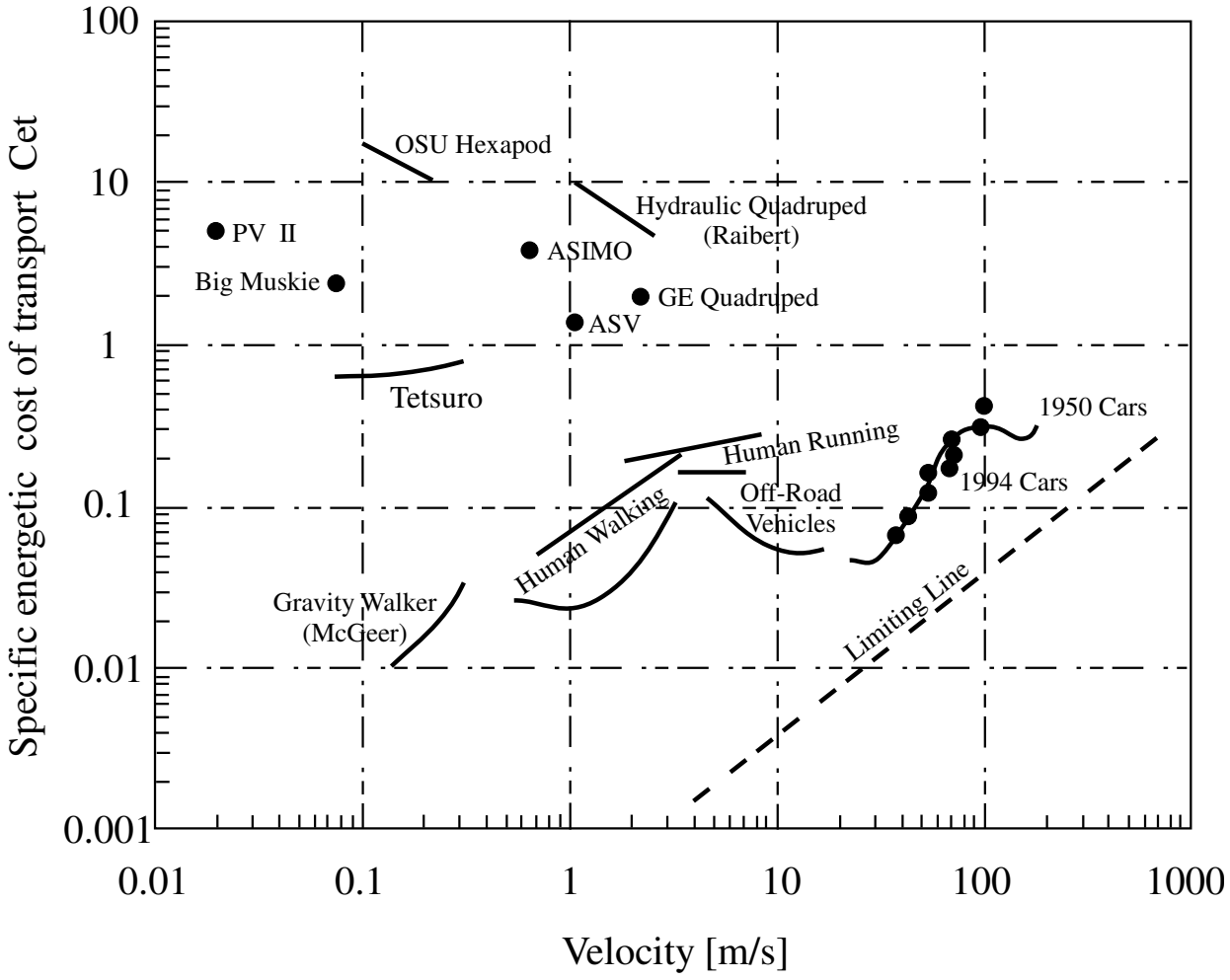


Figure 5.2: Specific const of transport for selected land vehicles (modified from Gregorio et al. 1997[Gregorio:1997]).

第6章　結論

人のsplit-belt treadmill歩行における歩容適応を行うための制御モデル(Figure 2.8参照)を開発した2次元二脚ロボット「鉄郎」を用いて提案した．減速比を小さく設計して各関節のギア間における摩擦を小さくし，各関節に高いバックドライバビリティを持たせること・腰関節まわりの脚の慣性モーメントを小さくすること・腰，膝，足首関節やfootに伸縮素材を装着しコンプライアンスを持たせること・支持脚相における膝ロックなどの機構的な工夫を行った．CPGの単純化モデルとしてリズミカルな運動を発生させる軌道生成・脊髄における緊張性伸張反射の単純化モデルとしての関節におけるPD制御・前庭脊髄反射として前進速度と姿勢の安定化のための脳幹におけるstepping reflexから成る従来のリミットサイクルを構成する制御手法を実装した．この従来型リミットサイクルを構成する制御手法に新規性はない．これより鉄郎はナチュラルダイナミクスを利用した効率的な歩行が可能となった．モータドライバ等電源部に問題がない5分程度，鉄郎はtied-belt treadmill歩行を継続することができる．Specific energetic cost of transport(C_{et})とSpecific mechanical cost of transport(C_{mt})を用いてエネルギ効率を定量的に評価し，本研究で用いた制御手法のエネルギ効率のよさを示した．また，stepping reflexにより鉄郎は後ろから押すような外乱に適応し前進速度や姿勢を保つことができる．リミットサイクルの定義は初期値に依存せず位相平面内で閉ループを形成しかつ外乱に適応可能であることである．この結果をリミットサイクルの観点から評価すると，本研究では倒立振子ベースで軌道を生成しデューティ比0.7程度での歩行を実現している．これは初期値にある程度は依存していること(初期値としての必要条件はロボットが立脚静止していること)・両脚支持期間において系の安定化を計っていることを意味しており，完全に倒立振子的に重力を有効利用した歩行ではない(完全な倒立振子運動はデューティ比が0.5)．ただ，周期的に歩行運動は生成されておりかつstepping reflexにより外乱に適応可能となっているので，その点ではリミットサイクルの定義に則っていると考えられる．

この従来のリミットサイクルを構成する制御手法と新たに提案した小脳による適応の単純

化モデルとしての支持脚腰関節におけるPゲイン調節を組み合わせ，split-belt treadmill上の歩行を実現した．この提案したPゲイン調節･歩容適応モデルに新規性がある．運動学的指標ストライド長･デューティ比･ステップ長の差･両脚支持期間比の差における(健常者モデルと小脳疾患患者モデル)実験データとBastianらによって得られた健常者と小脳疾患患者の実験データに比率とパターンにおいて高い類似性があることを示した．また健常者モデルと小脳疾患患者モデルのinterlimb indexesにおける実験結果より，支持脚腰関節におけるPゲインがsplit-belt treadmill歩行における対称性を有する歩容適応の制御パラメータであることを示し，Pゲイン調節は小脳による筋張力調節に相当した．結果として，我々は人のsplit-belt treadmill歩行における歩容適応モデルを提案し，二脚ロボットを用いて我々の仮説と提案した歩容適応モデル(Figure 2.8参照)の正当性を検証した．最後に，本研究の現状では社会的に貢献できる(役に立つ)ような成果は，得られていない．ただ，本研究は，今後社会に還元できるようなものを提供するためのはじめの一歩であると，我々は考えている．なお本論文で紹介した鉄郎の歩行の映像は，http://robotics.mech.kit.ac.jp/research/Biped/photo-movie-tetsuro1-j.html において公開されている．

第7章　今後の展望

次のステップとして，胴体を付けたうえでロール面内における拘束がない3次元二脚ロボットを用いて提案した制御モデルの拡張を行いたい．2005年4月に鉄郎を左右方向に手で振ることによりロール運動を起こし，そのロール運動と同期したピッチ運動を生成させ，3次元平面内で歩行させようと試みた．当時搭載していたロール軸まわりのレイトジャイロが，胴体のロール軸まわりの角速度を検出していた．これを用いて，鉄郎のロール運動の振動中心の角速度をフィードバックし，ロール運動の運動エネルギが大きい時はピッチ運動の周期を大きく，運動エネルギが小さい時は周期を小さくした．だがそれだけでは姿勢制御が十分でなく，安定した歩行は得られなかった．また，鉄郎の両膝を交互に曲げる運動を生成することでロール運動を起こし，その運動で歩行させようと試みた．だがこの実験においてもそれだけでは姿勢制御が十分でなく，膝を曲げると胴体が後ろにのけぞって倒れてしまい，安定した歩行は得られなかった．鉄郎の胴体リンクの前部におもりを載せることで膝を曲げても後ろに倒れないようにバランスをとろうと工夫を試みたが，膝曲げ運動の際中に鉄郎の重心をfoot内に置くことができなかった．これらの経験より，ロール面内においてもactiveに倒立振子的な運動を起こす必要があるのではないかと考える．そのためにはロール軸まわりにアクチュエータを搭載する必要がある．後任研究者の八木らがこの仕事を現在行っているので，経過を教えていただきたいと考えている．passiveにロール運動を起こすことができればエネルギ効率の観点から言えば申し分ない．そのためには脚の歪曲，ピッチ運動の支持脚の倒れ込みの角度，接地した時のfootのグリップ力を上げる(footが地面をつかむようなイメージ)などの調整を行う必要があるのではないかと考えている．

ストライド長をより正確に算出するため，トーションスプリングなどを用いてsplit-belt treadmill歩行時の胴体の位置変化を計測する必要がある．我々の歩容適応モデルがsplit-belt treadmill上でなぜこれらの指標を調整できるのか，またどのように調節しているのかを詳しく調べるために数理解析を行う必要がある．京都大学の青井先生より，従来型のリミッ

トサイクルを構成する制御手法(Pゲイン調節なし)を用いて実現した歩行ならば，その運動を解析できるとのご教授をいただき，現在試行中である．具体的には，鉄郎の簡単な力学モデルを立て，関節角度・歩行周期・ベルト速度などの変数を定め，4つの指標(ストライド長，デューティ比，ステップ長の差，両脚支持期間比の差)を式で表現しようと試みている．

加えて，我々の仮説や提案したモデルの正当性を検証するために将来的には人のsplit-belt treadmill歩行を我々が実際に行い，3次元での歩行データや床反力などを計測する必要がある．同志社大学の船戸らが現在split-belt treadmill歩行を試行中なので後々ご教授を受けさせていただき，人とロボットの歩行イメージの対比表現の工夫を行いたいと考えている．

謝辞

本研究を行う貴重な機会をくださり，御指導・叱咤激励をしてくださった木村浩教授に感謝致します．変わらぬ配慮・激励・丁寧な御指導をしてくださった高瀬國克教授に感謝致します．また，有用なアドバイスをくださった羽田芳朗様，jia songmin助教，中後大輔助教に感謝致します．

私にとってはもう御一方の指導教官である福岡泰宏講師からは多くを学ばせていただきました．感謝致します．先輩方の張祖光助教・佐藤啓宏様・大西隆之様・覺張陽則様・阿部貴史様・宮腰清一様・小野栄一様には貴重な時間を割いて大変有用な意見を頂きましたことを感謝致します．本研究の先行研究者である有村圭介様・高橋佑治様に感謝致します．お2人と同期の仲手川宗則様にもいろいろお世話になりました．後任研究者になる永本貴史様にも感謝致します．鉄郎をよろしくお願いいたします．博士課程時同期のクリストフ・モフロア様とは様々な議論をさせていただきましたことを感謝致します．修士課程時同期の片渕広生様，そして益田俊樹様をはじめとした後輩の皆様方，雑用等を分担してくれて助かりました．また，他の高瀬研究室の面々，講座，事務, 食堂, 生協の皆様にも感謝致します．

本研究の一部は科研費・特定領域研究「移動知」: 領域番号454の研究助成を受けて行われました．(株)東京精機，(有)図工，(株)応用計測研究所には鉄郎製作を委託しました．ここに感謝の意を表します．何度も実験したにもかかわらず，ほとんど故障しなかった鉄郎に感謝致します．加えて，split-belt treadmillを譲ってくださり本論文作成のための基となる論文を紹介してくださった東京大学大学院広域科学専攻の柳原大准教授に感謝致します．

オーストラリアのニューキャッスル大学に短期留学させていただいたことに感謝致します．その際お世話になった鈴木雅久准教授・新垣モニカ様に御礼申し上ます．

また博士課程1年時，健康を取り戻すきっかけを与えてくださった保険管理センターの長江先生に感謝致します．

的を得た多くのアドバイスをくださり，多くの議論もしてくだったシステムサイエンス研究所代表取締役 柴本巌先生に感謝致します．

その他，多くの方々の助言・御支援により博士課程を経験し修了することができました．誠にありがとうございました．

修士，博士課程と5年間在籍させていただいた電気通信大学大学院に感謝致します．

最後に，博士課程に進学することを承諾してサポートしてくれた母と2人の弟に感謝致します．

参考文献

[青井:2007] 木村 麻衣，青井 伸也，土屋 和雄: ヒトの神経筋骨格モデルの基づく歩行生成，第19回自律分散システム・シンポジウム，pp.207-212, 2007

[浅野:2004] 浅野 文彦, 羅 志偉, 山北 昌毅: 受動歩行を規範とした2足ロボットの歩容生成と制御, 日本ロボット学会誌, vol.22, no.1, pp.130-139, 2004

[稲垣:2007] 稲垣 伸吉，細江 繁幸，鈴木 達也: 振動子- 機械系におけるエネルギー追従制御によるコンパス型歩行ロボットの歩容生成，第19回自律分散システム・シンポジウム，pp.223-226, 2007

[江原:2006] 江原義弘: 人間の歩行，ロボットの歩行, 計測と制御, vol.45, no.12, pp.1018-1023, 2006

[遠藤:2004] 遠藤 玄，森本 淳，中西 淳，ゴードン: 神経振動子を用いた二足歩行運動の実験的検討，日本機械学会ロボティクス・メカトロニクス講演会講演論文集，4月, 1A1-L1-53, 2004

[大須賀:2004] 大須賀 公一, 杉本 靖博, 杉江 俊治:　遅延フィードバック制御に基づく準受動的歩行の安定化制御, 日本ロボット学会誌, vol.22, no.2, pp.193-199, 2004

[大須賀:2005] 大須賀 公一: 移動知理解への力学的接近, 計測と制御, vol.44 no.9, pp.640-645, 2005

[小野:1994] 小野 京右，岡田 徹: 自励振動アクチュエータに関する研究(第3報，自励駆動による二足歩行機構)，日本機械学会論文集C, vol.60, no.579, pp.3711-3718, 1994

[梶田:1996] 梶田 秀司, 谷 和男: 実時間路面形状計測に基づく動的2足歩行の制御, 日本ロボット学会誌, vol.14, no.7, pp.1062-1069, 1996

[梶田:2004] 梶田 秀司，金広 文男，金子 健二，藤原 清司，原田 研介，横井 一仁，比留川博久: 分解運動量制御：運動量と各運動量に基づくヒューマノイドロボットの全身運動生成，日本ロボット学会誌，vol.22, no.6, pp.772-779, 2004

[梶田:2005] 梶田 秀司： ヒューマノイドロボット, オーム社, 2005

[加藤:1979] 加藤 了三, 森 政弘: 安定な閉軌道を持つ力学系を基礎にした2足歩行の制御, バイオメカニズム5, 東京大学出版会，pp.259-268, 1979

[木村:2003] 木村 浩: 生物を規範とした脚式ロボットの不整地適応, 計測と制御, vol.42, no.9, pp.705-711, 2003

[國吉:2003] 國吉 康夫: ロボットの知能-創発実態主義の挑戦-, 計測と制御, vol.42, no.6, pp.497-503, 2003

[佐野:1989] 佐野 明人，古荘 純次: 角運動量制御による2 足歩行ロボットの3次元動歩行, vol.26, no.4, pp.459-466, 1989

[下山:1981] 下山 勲，三浦 宏文: 竹馬型二足歩行ロボットの動的歩行，機械学会講演論文集，vol.817, no.2，pp.73-80, 1981

[杉本:2005] 杉本 靖博: ポアンカレマップ内に存在するフィードバック構造に着目した受動的歩行の安定解析，ロボティクスシンポジア，pp.127-132, 2005

[多賀:1997] 多賀 巌太郎： 歩行の創発, 日本ロボット学会誌, vol.15, no.5, pp.680-683, 1997

[高西:1985] 高西 淳夫, 石田 昌巳, 山崎 芳昭, 加藤 一郎: 2足ロボットWL-10RDによる動歩行の実現，日本ロボット学会誌，vol.3，no.4 pp.325-336, 1985

[高橋:2005] 高橋 佑治，有村 圭介，音田 裕史，木村 浩: 創発型二脚歩行運動生成，計測と制御， 第32回知能システムシンポジウム，pp.301-306, 2005

[土屋:1999] 土屋 和雄： 複雑系の構成原理，計測と制御，vol.38，no.10，pp.605-611，1999

[土屋:2002] 土屋 和雄，辻田 勝吉: Central Pattern Generatorモデルに基づく4脚歩行ロボットの歩行制御，日本ロボット学会誌，vol.20，no.3, pp.243-246, 2002

[土屋:2005] 土屋 和雄: 移動知：脳・身体・環境の生み出す知能, 計測と制御, vol.44, no.9, pp.579, 2005

[内藤:2006] 内藤 尚，長谷 和徳，井上 剛伸，大道 佳昇，相川 孝訓，山崎 伸也，諏訪 基，大日方 五郎: 神経・筋骨格系を有する人体モデルを用いた股義足開発支援シミュレータの開発，バイオリズム学会誌，no.18, pp.113-125, 2006

[西脇:2007] 西脇 光一，加賀美 聡: ヒューマノイドのための短周期オンライン歩行軌道生成更新法， 日本ロボット学会誌，vol.25, no.6, pp.36-43, 2007

[畠:2005] 畠 直輝, 堀 洋一: 歩行補助装具への実装を目的とした歩行安定化制御, 電気学会産業計測制御研究会, IIC-05-11, 名古屋, 2005

[福岡:2003] 福岡 泰宏, 木村 浩: 四足ロボットの生物規範型不整地適応動歩行 – 神経- 機械カップリング系構成法の提案とピッチ運動・CPG・ロール運動間相互引き込みの評価 –，日本ロボット学会誌，vol.21, no.5, pp.569-580, 2003

[福島:2002] 福島 文彦，滝田 健介，広瀬 茂男，中村 亨: 知能ロボット用シリアルバス型制御システムの研究, 第7回ロボティクスシンポジア予稿集，pp.155-160，2002

[古荘:2002] 古荘 純次： 2足歩行のダイナミックスと制御, 生物型システムのダイナミックスと制御, 養賢堂, pp.105-154, 2002

[細田:2005] 細田 耕: 非限定環境に適応する二足歩行ロボット, 計測と制御, vol.44, no.9, pp.609-614, 2005

[宮腰:2006] 宮腰 清一: メモリ・ベースト運動制御による2足歩行の制御，日本ロボット学会誌，vol.24, no.5, pp.623-631, 2006

[柳原:1999] 伊藤 聡，湯浅 秀男，羅 志偉，伊藤 正美，柳原 大: 環境の変化に適応する四足歩行ロボットシステム，日本ロボット学会誌, vol.17, no. 4, pp.595-603, 1999

[矢野:2003] 富田 望，矢野 雅文: 拘束条件生成と拘束充足による二脚歩行のリアルタイム制御, 第15回自律分散システムシンポジウム, pp.22-32，2003

[矢野:2005] 矢野 雅文，冨田 望: 実環境における2足歩行の創発的リアルタイム制御，日本ロボット学会誌，vol.23, no.1, pp.11-16, 2005

[山口:1996] 山口 仁一，木下 昇，高西 淳夫，加藤 一郎: 路上形状に偏差のある環境に対する適応能力を持つ２足歩行ロボットの開発，日本ロボット学会誌, vol.14, no.4, pp.546-559, 1996

[Aoi:2005] S. Aoi and K. Tsuchiya: Locomotion control of a biped robot using nonlinear oscillators, Autonomous Robots, no.19, pp.219-232, 2005

[Bastian:2008] J. Lee, J.T. Choi, S. Carver, A.J. Bastian and N.J. Cowan: Toward a Neuromechanical Model for Adaptation and Control of Human Running, AMAM2008, pp.94-95, 2008

[Chemori:2004] A. Chemori: Generation of Multi-steps limit cycles for Rabbit using a low dimensional nonlinear predictive control scheme, Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems , 2004

[Collins:2001] S.H. Collins, M. Wisse and A. Ruina: A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees, Int. J. of Robotics Research, vol.20, no.7, pp.607-615, 2001

[Collins:2005-1] S.H. Collins: A Bipedal Walking Robot with Efficient and Human-like Gait, ICRA Spain , pp.1995-2000, 2005

[Collins:2005-2] S.H. Collins, A. Ruina, M. Wisse, and R. Tedrake: Efficient bipedal robots based on passive-dynamic walkers, Science, no.307, pp.1082-1085, 2005

[Endo:2004] G. Endo, J. Morimoto, J. Nakanishi and G. Cheng: An Empirical Exploration of a Neural Oscillator for Biped Locomotion Control, Proc. of IEEE ICRA, pp.3036-3042, 2004

[Geng:2006] T. Geng, B. Porr and F.Wörgötter: Fast Biped Walking with a Sensor-driven Neuronal Controller and Real-time Online Learning, Int. J. of Robotics Research, no.25, pp.243-259, 2006

[Giese:2007] W. Ilg, H. Golla, P. Thier and A. Giese: Specific influences of cerebellar dysfunctions on gait, Brain, no.130, pp.786-798, 2007

[Gregorio:1997] P. Gregorio, M. Ahmadi, and M. Buehler: Design, control, and energetics of an electrically actuated legged robot, IEEE Trans. on System, Man, Cybernetics. PartB, Cybernetics, vol.27, no.4, pp.626-634, 1997

[Hirai:1998] K. Hirai, M. Hirose, Y. Haikawa and T. Takenaka: The development of Honda Humanoid Robot, Proc. of IEEE ICRA, pp.1321-1326, 1998

[Hodgins:1991] H. Hodgins and M. Raibert: Adjusting Step Length for Rough Terrain Locomotion, IEEE trans. on Robotics and Automation, vol.7, no.3, pp.289-298, 1991

[Ijspeert:2007] A.J. Ijspeert, A. Crespi, D. Ryczko and J. Cabelguen: From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model, Science, no.9, pp.1416-1420, 2007

[Ijspeert:2008] A.J. Ijspeert: Central pattern generators for locomotion control in animals and robots: A review, Neural networks, no.21, pp. 642-653, 2008

[Kandel:1996] E.R. Kandel, J.H. Schwartz and T.M. Jessell: Principles of neural science, Appleton & Lange, pp.626-646, 1996

[Matsuoka:1987] K. Matsuoka: Mechanisms of Frequency and Pattern Control in the Neural Rhythm Generators, Biolog. Cybern, vol.56, pp.345-353, 1987

[Maufroy:2008] C. Maufroy, H. Kimura and K. Takase: Towards a general neural controller for quadrupedal locomotion, Neural networks, no.21, pp.667-681, 2008

[McGeer:1990] T. McGeer: Passive Dynamic Walking, Int. J. of Robotics Research, vol.9, no.2, pp.62-82, 1990

[Miura:1984] H. Miura and I. Shimoyama: Dynamical walk of biped locomotion, Int. J. of Robotics Research, no.3, pp.60-74, 1984

[Nomura:2006] M. Nakanishi, T. Nomura and S.Sato: Stumbling with optimal phase reset during gait can prevent a humanoid from falling, Biological Cybernetics, vol.95, no.5, pp.503-515, 2006

[Morton:2006] S.M. Morton and A.J. Bastian: Cerebellar Contributions to Locomotor Adaptations during Splitbelt Treadmill Walking, J. of Neuroscience, no.26, pp. 9107-9116, 2006

[Ono:2004] Ono, Furuichi, and Takahasi: Self-Excited Walking of a Biped Mechanism with Feed, Int. J. Robotics Research, no.23, vol.1, pp.55-68, 2004

[Orlovsky:1999] G. Orlovsky, T. Deliagina and S. Grillner: Neural control of locomotion, Oxford Univ. Press, NY, 1999

[Otoda:2007] Y. Otoda, H. Kimura and K. Takase: Efficient Bipedal Walking Using Delayed Stepping Reflex, Proc. of SICE Annual Conference 2007 (SICE2007), Takamatsu, pp.2170-2174, 2007

[Otoda:2008] Y. Otoda: Efficient Bipedal Walking Using Natural Dunamics and Delayed Stepping Reflex, IEEJ Transaction of Electrical & Electric Engineering, vol.3, no.4, pp.413-417, 2008

[Raibert:1986] M.H. Raibert: Legged Robots That Balance, The MIT Press, 1986

[Reisman:2005] D.S. Reisman, H.J. Block and A.J. Bastian: Interlimb Coordination During Locomotion: What Can be Adapted and Stored?, J. of Neurophysiology, no.94, pp.2403-2415, 2005

[Rogers:2003] M.W. Rogers, M.E. Johnson, K.M. Martinez, M. Mille and L.D. Hedman: Step training improves the speed of voluntary step initiation in aging, J. of Gerontology Series A: Biological Sciences and Medical Sciences, no.58, pp.46-51, 2003

[Rossignol:2006] S. Rossignol, R. Dubuc and J. Gossard: Dynamic sensorimotor interactions in locomotion, Physiol. Rev., no.86, pp.89-154, 2006

[Schaal:2008] S. Schaal, Y. Nakamura and P. Dario: Special issue on robotics and neuroscience, Neural networks, no.21, pp.551-552, 2008

[Spek:1999] J.H. van der Spek, M.C. van der Maas and P.H. Veltink: Control of crutch supported standing in paraplegics - a model based study, Proc. of Int. Biomechatronics workshop, pp. 162-167, 1999

[Taga:1991] G. Taga, Y. Yamaguchi and H. Shimizu: Self-organized control of bipedal locomotion by neural oscillators, Biological Cybernetics, nol.65, pp.147-159, 1991

[Taga:1995a] G. Taga: A model of the neuro-musculo-skeletal system for human locomotion I. - Emergence of basic gait, Biological Cybernetics, no.73, pp.97-111, 1995

[Taga:1995b] G. Taga: A model of the neuro-musculo-skeletal system for human locomotion II. - Real-time adaptability under various constraints,
Biological Cybernetics, no.73, pp.113-121, 1995

[Takakusaki:2008] K. Takakusaki and T. Okumura: Neurobiological Basis of Controlling Posture and Locomotion, Advanced Robotics, vol.22, no.15, pp.1629-1663, 2008

[Takuma:2006] T. Takuma and K. Hosoda: Controlling the Walking Period of a Pneumatic Muscle Walker, Int. J. of Robotics Research, no.25, pp. 861-866, 2006

[Tsujita:2007] K. Tsujita, T. Inoura, A. Morioka, K. Nakatani, K. Suzuki and T. Masuda: Oscillator-controlled Bipedal Walk with Pneumatic Actuators, J. of Mechanical Science Technology, no. 21, pp.976-980, 2007

[Vukobratović:1972] M. Vukobratović and J. Stepanenko: On the Stability of Anthropomorphic Systems, Mathematical Bioscience, vol.15, pp.1-37, 1972

[Wisse:2003] M. Wisse,J.van Frankenhuyzen: Design and Construction of MIKE;a 2D autonomous biped based on passive dynamic walking, AMAM2003 International Symposium on Adaptive Motion of Animals and Machines, 2003

[Wisse:2008] D.G.E. Hobbelen and M. Wisse: Controlling the Walking Speed in Limit Cycle Walking, Int. J. of Robotics Research, vol.27, no.9, pp.989-1005, 2008

付録

付録として実験で用いたパラメータ値，歩行を開始トリガの説明，ダイナミクスレベルでの人との比較，transitional walkingにおける軌道計画，鉄郎の自由度についての説明，片方のベルト速度が下がる追加実験，鉄郎の設計図，モータドライバの回路図，鉄郎の制御プログラムを載せる．

A. パラメータ値

Table 7.1: Values of the parameters used in experiments with Tetsuro. SSS and LES mean the stance-swing state and landing-exchange state, respectively.

parameters	value	parameters	value
I_1[kg·m^2]	0.39	$k^{st}_{h\_p}$[Nm/rad] (in LES)	18
m_1[kg]	2.3	$k^{st}_{k\_p}$[Nm/rad] (in LES)	9.0
g[m/s^2]	9.8	$k^{st}_{a\_p}$[Nm/rad] (in LES)	2.0
l_1[m]	0.41	$k^{sw}_{h\_p}$[Nm/rad] (in LES)	20
T_0[s]	0.32	$k^{sw}_{k\_p}$[Nm/rad] (in LES)	9.0
$k^{sw}_{h\_p}$[Nm/rad] (in SSS)	26	$k^{sw}_{a\_p}$[Nm/rad] (in LES)	3.0
$k^{sw}_{k\_p}$[Nm/rad] (in SSS)	0.0	k_{sr}[s]	0.35
$k^{sw}_{k\_v}$[Nm·s/rad] (in SSS)	0.0	k_{ag} [Nm/rad^2]	3.9
$k^{sw}_{a\_p}$[Nm/rad] (in SSS)	2.0	n	6
$k^{st}_{h\_p}$[Nm/rad] (in SSS)	26	T_{wt} [s]	50
$k^{st}_{k\_p}$[Nm/rad] (in SSS)	9.0	A_{th}[rad]	0.16
$k^{st}_{a\_p}$[Nm/rad] (in SSS)	3.0		

B. 歩行開始トリガ

鉄郎を静止状態から定常状態に遷移させるためにトリガを引いた．Figure 7.1に歩行開始トリガのイメージを示す．回転していないトレッドミルに静止した鉄郎を載せ，トレッドミルを回転させると鉄郎も前のめりに回転する．両脚足首関節のエンコーダを用いてその回転を検知するような閾値(A_{th})を設け，振り出し相へ移行させるようなトリガを引いた．

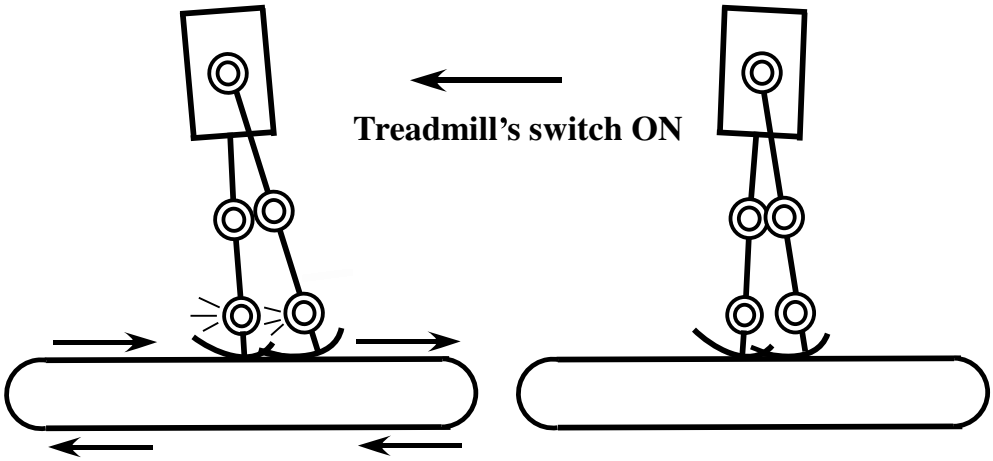


Figure 7.1: Trigger to walk.

C. ダイナミクスレベルでの比較

split-belt treadmill内蔵の力センサを用いて計測したtied-belt treadmill定常歩行時の鉄郎の床反力をFigure 7.2に示す．人の平地歩行のように左右の脚の床反力に二峰性の波形が表れた．しかし，支持脚の膝関節をロックしたこと・蹴り動作が行われていなかったことにより，二峰性がはっきりと見られない．畠ら[畠:2005]によると，蹴り動作を行わない人の歩行においてこの図のような変化パターンがよく見られるという．

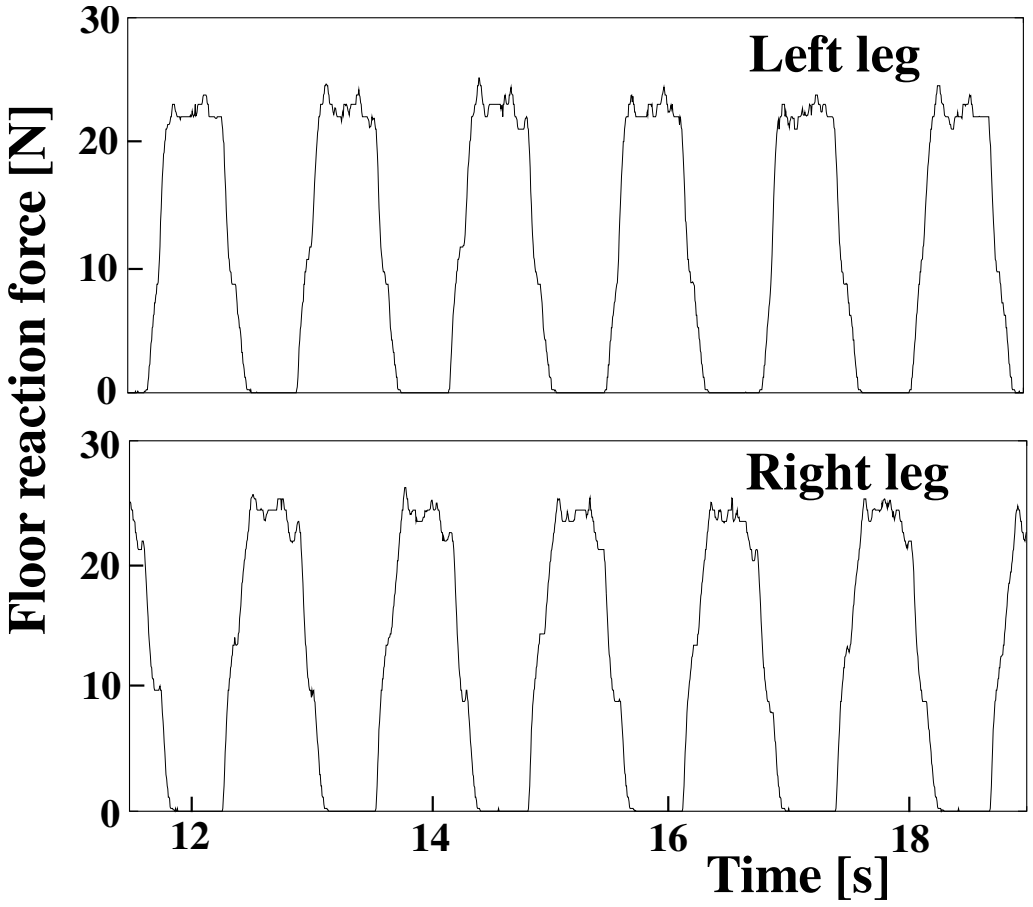


Figure 7.2: The floor reaction force.

D. Transitional walkingにおける軌道計画

1ステップ目

1ステップ目の終端角度と終端角速度が，それぞれsteady walking(3ステップ目以降)における終端角度式(4.3)と終端角速度式(4.4)の半分になる軌道を計画した(Figure 7.3参照).

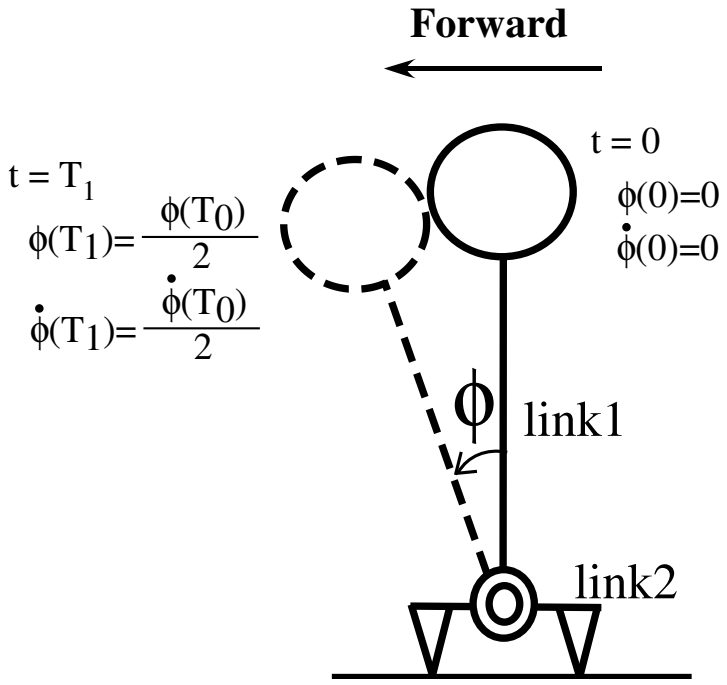


Figure 7.3: First step. T_1 is the period of stance leg.

これらの拘束条件を式で表すと以下のようになる.

$$\phi(0) = 0 \tag{7.1}$$

$$\dot{\phi}(0) = 0 \tag{7.2}$$

$$\phi(T_1) = \frac{\phi(T_0)}{2} \tag{7.3}$$

$$\dot{\phi}(T_1) = \frac{\dot{\phi}(T_0)}{2} \tag{7.4}$$

歩き始めの1ステップ目($0 \leq t \leq T_1$)において，これら4つの式を満たす軌道は以下の式で表される．

$$\phi(t)_{1st} = \frac{\alpha}{2T_1^2}t^2, \qquad T_1 = \frac{2\alpha}{a(\alpha + 2e^{-aT_0} + 2)}\ , \tag{7.5}$$
$$\alpha = e^{aT_0} - e^{-aT_0}$$

ここでT_0はsteady walkingにおける半周期(支持脚期間)である．式(7.5)は，stance-swing state・landing-exchange state(Table 4.1，Table 4.2参照)における目標軌道$\phi(t)$に取って使われる．

2ステップ目

2ステップ目の初期角度と初期角速度は，1ステップ目の終端角度の異符号と終端角速度に相当するとした．2ステップ目の終端角度と終端角速度が，steady walkingにおける終端角度と終端角速度になる軌道を計画した(Figure 7.4参照)．

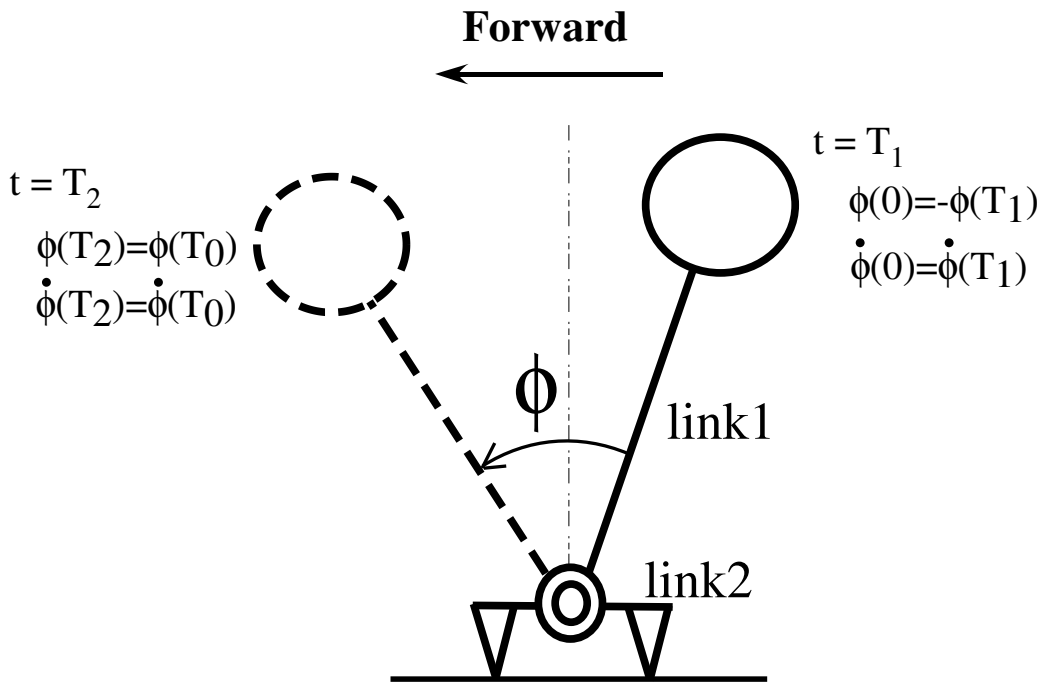


Figure 7.4: Second step. T_2 is the period of stance leg.

これらの拘束条件を式で表すと以下のようになる．

$$\phi(0) = -\frac{\phi(T_0)}{2} \tag{7.6}$$

$$\dot{\phi}(0) = \frac{\dot{\phi}(T_0)}{2} \tag{7.7}$$

$$\phi(T_2) = \phi(T_0) \tag{7.8}$$

$$\dot{\phi}(T_2) = \dot{\phi}(T_0) \tag{7.9}$$

歩き始めの2ステップ目$(0 \leq t \leq T_2)$において，これら4つの式を満たす軌道は以下の式で表される．

$$\phi(t)_{2nd} = \frac{\beta^2}{8\alpha}t^2 + \frac{\beta}{2}t - \frac{\alpha}{2}, \qquad T_2 = \frac{2\alpha}{\beta}\ , \tag{7.10}$$

$$\beta = a(e^{aT_0} + e^{-aT_0} + 2)$$

式(7.10)は，stance-swing state・landing-exchange state(Table 4.1，Table 4.2参照)における目標軌道$\phi(t)$に取って使われる．

E. 鉄郎の自由度

二脚ロボットの歩行時には下記3つの相が存在する．

①宙に浮いている相(浮遊相)

②片脚のみ接地している相(単脚支持相)

③両脚とも接地している相(両脚支持相)

この3つの相における2次元平面内の鉄郎の自由度について述べる．加えて，自由度が拘束される条件，生成する軌道の条件について述べる．

①浮遊相

浮遊相における鉄郎の状態をFigure 7.5に示す．この相における系の自由度，拘束条件，本当の自由度，アクチュエータ数，軌道の条件をTable 7.2に示す．この相は地面からの拘束を受けないので，これら3つの相の中で自由度が一番大きい．

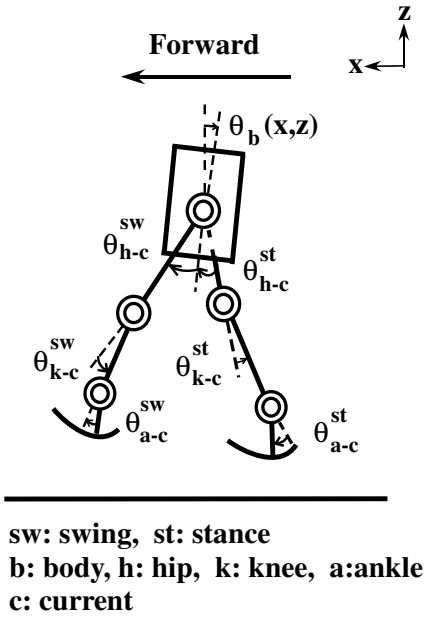


Figure 7.5: Flotage phase.

Table 7.2: Condition of flotage phase.

Degree of freedom of system	3×7 = 21
Restraint condition	(3-1)×6 = 12
Real degree of freedom	21-12 = 9
Number of actuators	6
Condition of trajectory	$\ddot{z} = -g$, $\ddot{x} = 0$ (free fall)

②単脚支持相

単脚支持相における鉄郎の状態をFigure 7.6に示す．この相においての系の自由度，拘束条件，本当の自由度，アクチュエータ数，軌道の条件をTable 7.2に示す．この相は片脚が地面からの拘束を受ける．ここでのZMP拘束は，目標ZMPを着地点が作る多角形内に置くこととする．

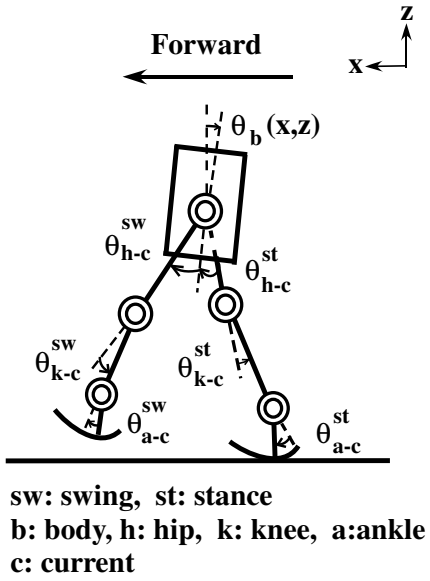


Figure 7.6: Single stance phase.

③両脚支持相

両脚支持相における鉄郎の状態をFigure 7.7に示す．この相においての系の自由度，拘束条件，本当の自由度，アクチュエータ数，軌道の条件をTable 7.4に示す．この相は両脚が地面からの拘束を受けるので，3つの相の中で自由度が一番小さい．

Table 7.3: Condition of single stance phase.

Degree of freedom of system	3×7 = 21
Restraint condition	(3-0)×3+ (3-1)×3 = 15
Real degree of freedom	21-15 = 6
Number of actuators	6
Condition of trajectory	Stance leg is fixed with ground and ZMP restraint.

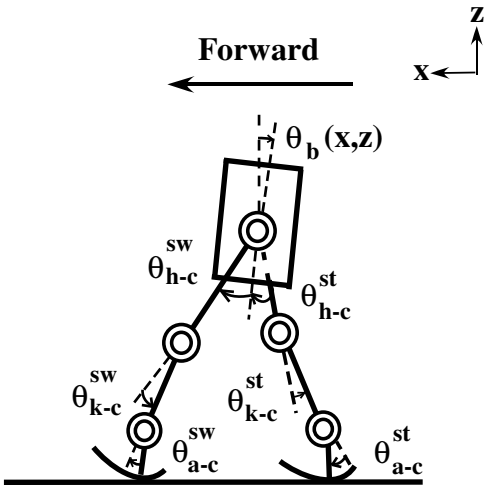


Figure 7.7: Double stance phase.

Table 7.4: Condition of double stance phase.

Degree of freedom of system	3×7 = 21
Restraint condition	(3-0)×6 = 18
Real degree of freedom	21-18 = 3
Number of actuators	6
Condition of trajectory	Both legs are fixed with ground.

F. 追加実験

追加実験として片方のベルト速度を下げるsplit-belt treadmill歩行実験を行った．トレッドミルの速度は“遅いモード”(0.15 [m/s])か“より遅いモード”(0.075 [m/s]) とした．この追加実験の時間設定も人の実験(Figure 2.1)と同様にした．各ステージの期間は15[s]とした．上記の実験設定のもと，従来のリミットサイクルを構成する手法に**4.2.1**節で述べたPゲイン調節を加え，鉄郎をsplit-belt treadmill上で歩行させた．split-belt treadmill歩行における両脚のψ_{off}と$k_{h_p}^{st}$をFigure 7.8に示す．ここでsplit-belt条件での“より遅いモード”と“遅いモード”のベルトに対応するそれぞれの脚を“slow leg”，“fast leg”とした．支持脚相におけるfast legの腰関節Pゲイン$k_{h_p}^{st}$が式(4.7)によりadaptationステージの始めにおいて素早く変化し，その後post-adaptation が終了するまで徐々に元の値に戻ることが見られた．Pゲイン調節はベルト片側の速度が半減するsplit-belt treadmill歩行を可能した．

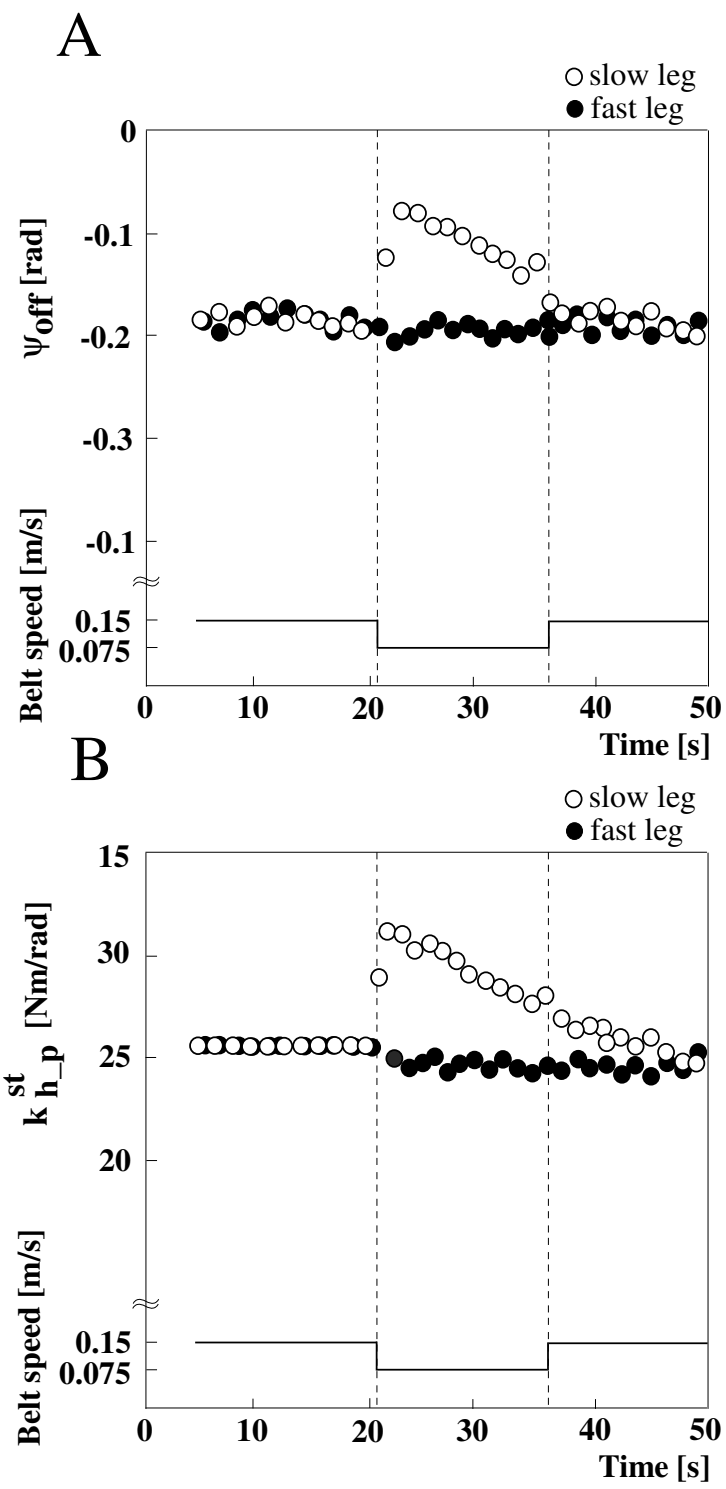


Figure 7.8: Results of robot (normal subject model) experiment. The ψ_{off}:(A) and the hip joint p-gain in the stance phase $k^{st}_{h\_p}$:(B) of both fast and slow legs in split-belt treadmill walking are shown. The time course of belts speed is described in the caption of Figure 4.15.

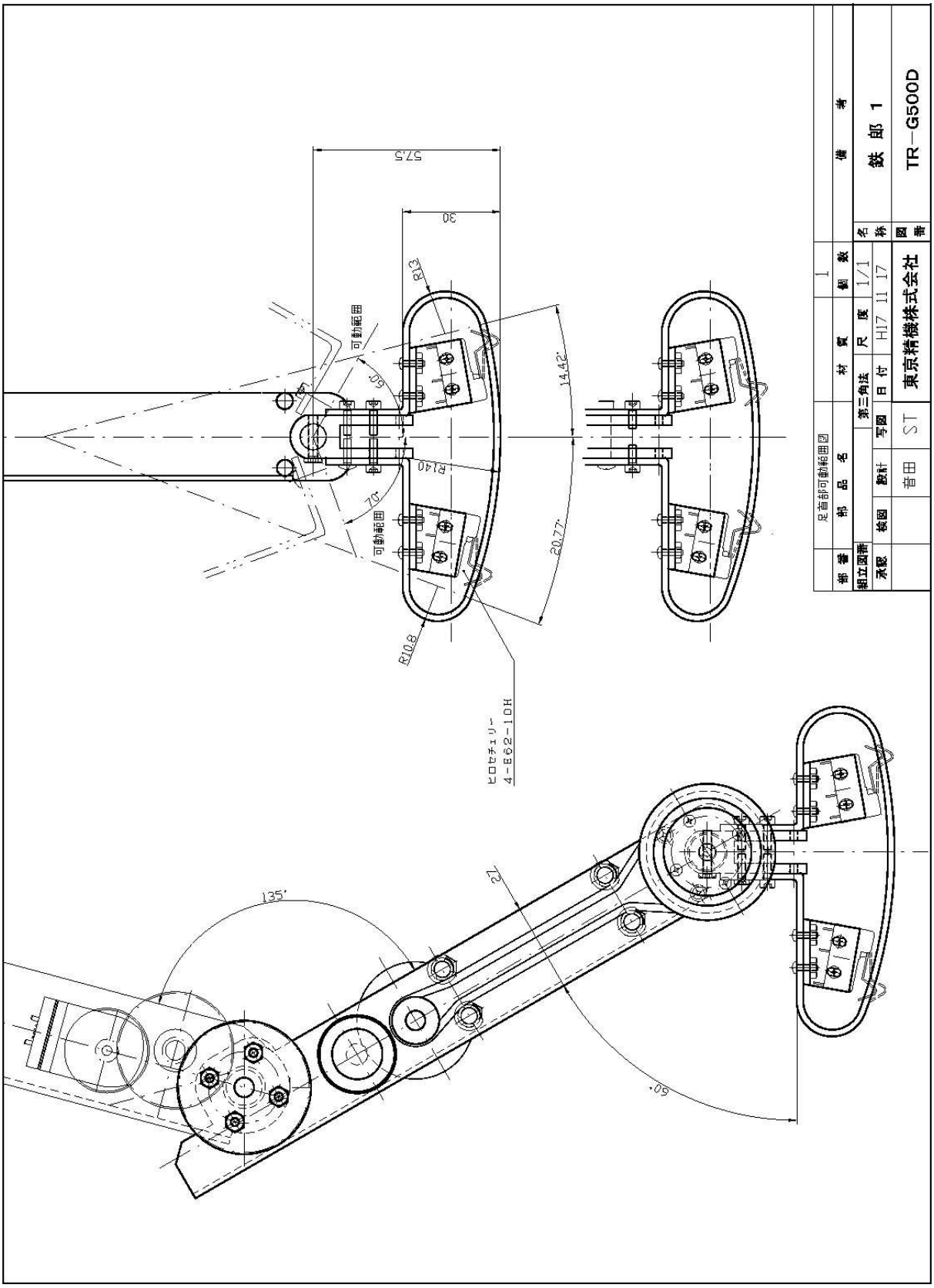


Figure 7.9: Mechanical draft under knee I.

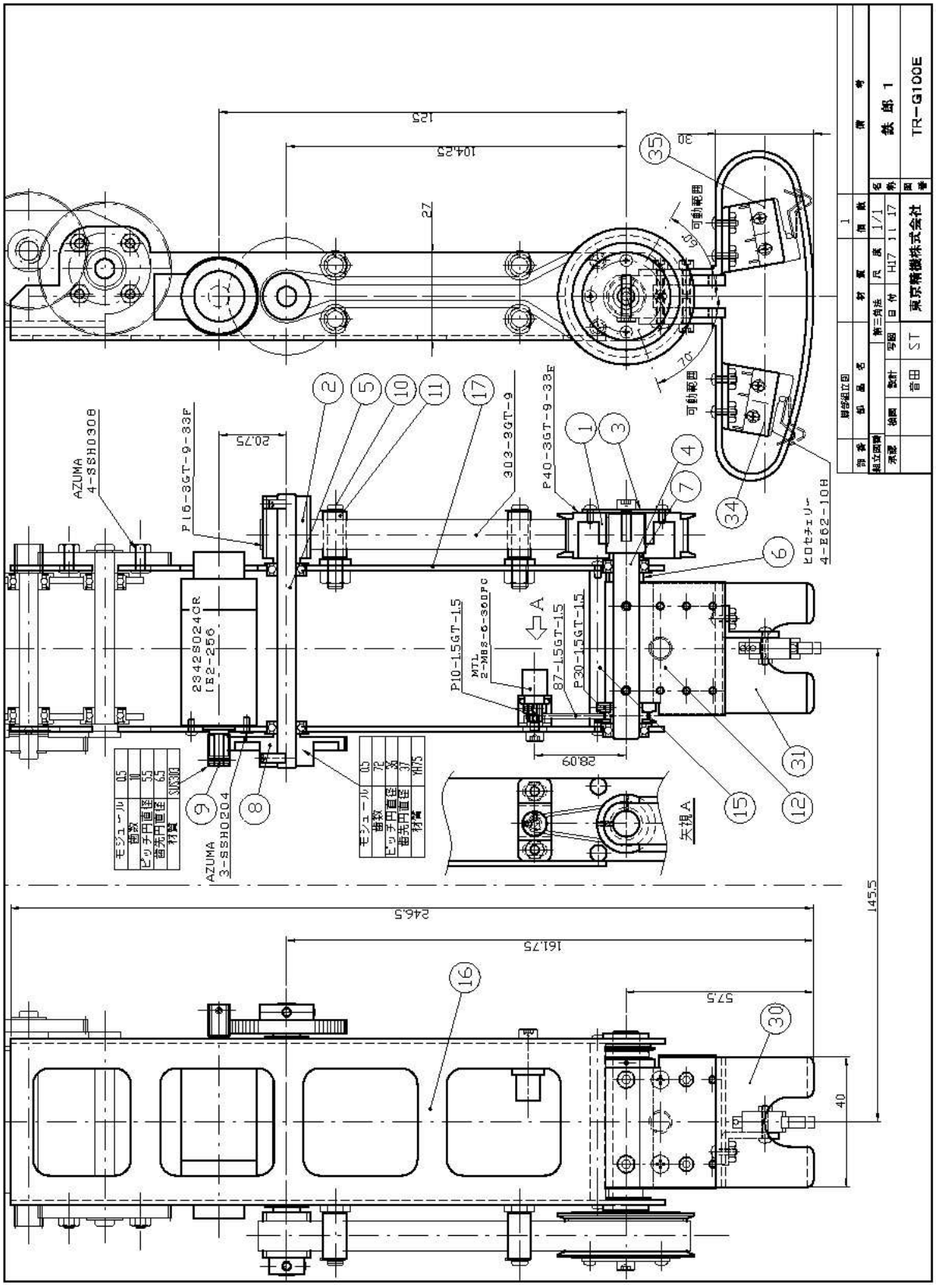


Figure 7.10: Mechanical draft under knee II.

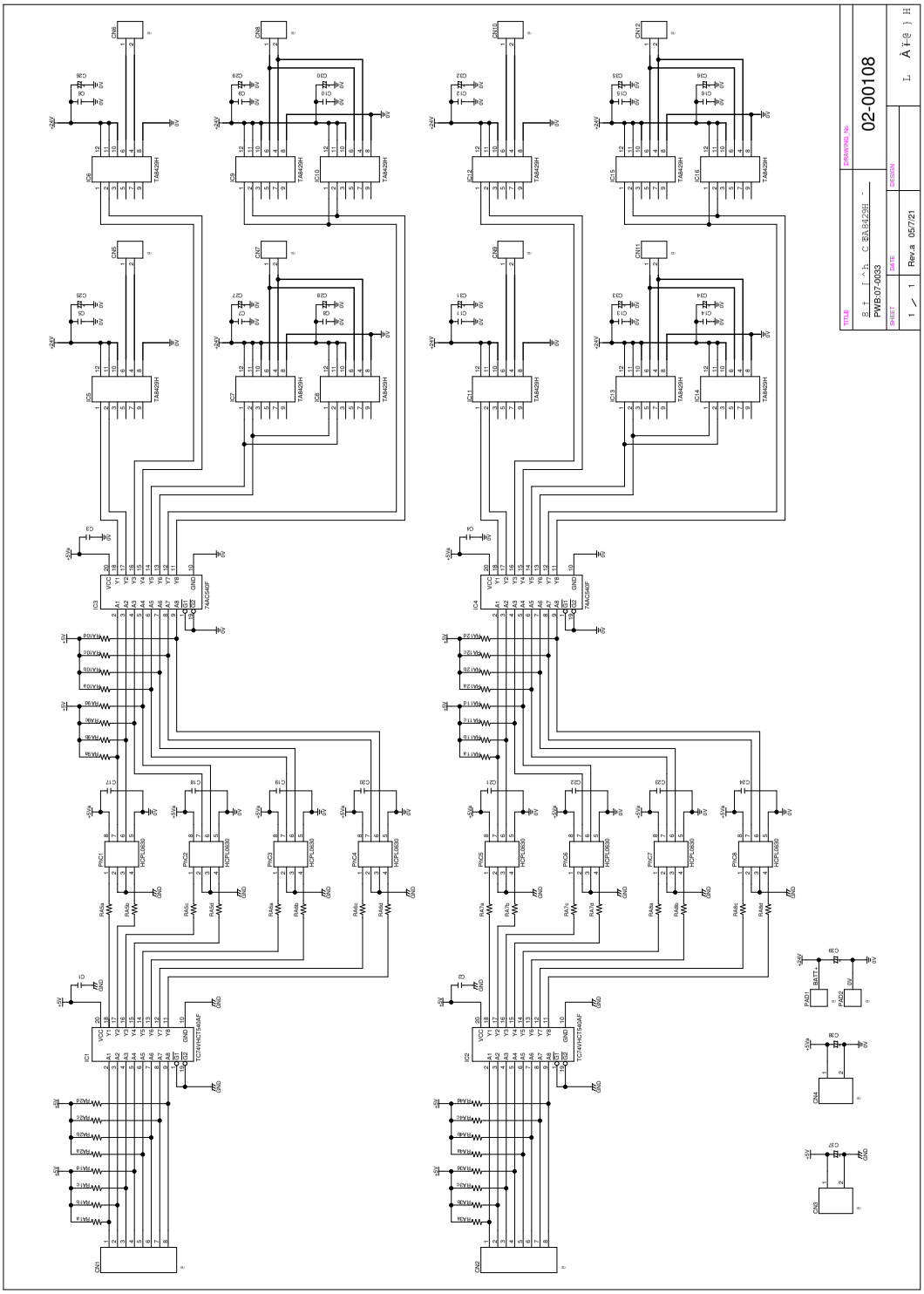


Figure 7.11: Motor driver of circuit diagram.

```
/*                                      Time-stamp: <04/10/03 18:12:21 yujioto> */
/* "init.c"     feed.cでincludeされる */
/* 初期化 */

void initState(stateType *p)
{
    p->x = 0.0;  p->v = 0.0;  p->u = 0.0;
}

void initGain(gainType *p, double k1, double k2)
{
    p->k1 = k1;  p->k2 = k2;
}

void initJoint(enum legs leg, enum joints jnt,
               int encDir, int encCH,
               int motorDir, int digital_bn, int motorCH_plus, int motorCH_minus,
               double k1, double k2)
{
    jointType *p;
    p = &s.leg[leg].joint[jnt];

    initState(&p->present);
    initState(&p->last);
    initState(&p->desired);
    initGain (&p->gain, k1, k2);

    p->ad = 0;  p->da = 0;

    p->encodeDir = encDir; /* エンコーダの回転方向設定 */
    p->encodeCH  = encCH;     /* エンコーダが接続されているchannelナンバー */

    p->motorDir  = motorDir;  /* モータの回転方向設定 */
    p->digital_board_num = digital_bn;
    p->motorCH_plus = motorCH_plus;
    p->motorCH_minus = motorCH_minus;

} /* initJoint */

void initLeg(enum legs leg,
             int encFDir, int encFCH, int encHDir, int encHCH)
{
    legType *p;
    p = &s.leg[leg];

    p->cs[fore].ad = 0;  p->cs[fore].encodeDir = encFDir;  p->cs[fore].encodeCH = enc
FCH;
    p->cs[hind].ad = 0;  p->cs[hind].encodeDir = encHDir;  p->cs[hind].encodeCH = enc
HCH;

    p->phase = stance;
}

void initData(void)
{
    enum legs leg;  enum joints jnt;  int it;

    for (it=0; it<=MaxTime; it++)  {
      for (leg=left; leg<=right; leg++) {
        for (jnt=hip; jnt<=yaw; jnt++) {
          s.data[it].px[leg][jnt] = 0.0;  s.data[it].dx[leg][jnt] = 0.0;
          s.data[it].pv[leg][jnt] = 0.0;
          s.data[it].pu[leg][jnt] = 0.0;
        }
        s.data[it].cs[leg][fore] = 0;
        s.data[it].cs[leg][hind] = 0;
```



```
      }
    }
} /* initData */

/* common.hの構造体sharedType型に定義されたs.〜(feed.cで使われる)の値の初期化 */
void init_shared(void)
{
  /* 共有変数領域の初期化 */
  s.feedFlag=FALSE;  s.storeFlag=FALSE;  s.walkFlag=FALSE;

/* for RT-Linux配線用 @@@@!!!:変更の必要 */
/* initJoint(leg  ,joint, encDir, encCH, mDir, digital_board_num, mCH_plus, mCH_minus
,  k1,   k2)*/
    initJoint( left,  hip,      1,     6,   -1,       D1_NODE_ADDR,         8,
9, 16.0, 0.03);
    initJoint( left,  yaw,      1,     5,   -1,       D1_NODE_ADDR,         6,
7,  6.0, 0.03);
    initJoint( left, knee,     -1,     7,   -1,       D1_NODE_ADDR,        10,         1
1, 15.5, 0.03);
    initJoint(right,  hip,     -1,     1,    1,       D1_NODE_ADDR,         2,
3, 16.0, 0.03);
    initJoint(right,  yaw,     -1,     0,   -1,       D1_NODE_ADDR,         0,
1,  6.0, 0.03);
    initJoint(right, knee,      1,     2,    1,       D1_NODE_ADDR,         4,
5, 15.5, 0.03);

/*  initLeg(leg  , encFDir, encFCH,  encHDir, encHCH) */
    initLeg(left ,       1,      8,        1,      9);
    initLeg(right,       1,      3,        1,      4);

/*-------------------------------------------------------------------------------
-----------*/

    /* データの初期化 */
    initData();

    /* その他初期化 */
    s.it = 0;  s.data_time = 0;
    s.ave_sampt = 0.0;  s.ave_feedt = 0.0;
    s.wt = 0.0;  s.stps = 0;
} /* init_shared */

/* モータ、エンコーダの分解能値←→Nm,radの変換に用いる係数の設定 */
void init_main(void)
{
  BPR[hip ] = BPR_HIP;   BPNM[hip ] = BPNM_HIP;
  BPR[knee] = BPR_KNEE;  BPNM[knee] = BPNM_KNEE;
  BPR[yaw ] = BPR_YAW;   BPNM[yaw ] = BPNM_YAW;
}
```

```
/*                                      Time-stamp: <04/09/25 18:01:56 yujioto> */
/* "biped.h" */
/* biped用物理パラメータ */

/* encoder は４逓倍。よって エンコーダパルス数×減速比×4/2π が以下の値 *//* !!! */
#define BPR_HIP    ((double) 1592.3)  /* bit / rad */  /* 400(pulse) / 2π(rad) * 25
 */
#define BPR_KNEE   ((double) 1592.3)  /* bit / rad */
#define BPR_YAW    ((double) 1712.1)  /* bit / rad */  /* 128(pulse) / 2π(rad) * 84
 */

/* MAX_DUTY_RATIO / (POWER_VOLTAGE/RESISTANCE_SE*TORQUE_CONSTANT_SE*HIP_GENSOKU_RATIO
2) */
#define BPNM_HIP    ((double) 1023.0 / (24.0 / 2.32 * 0.0232 * 25.0))
#define BPNM_KNEE    (BPNM_HIP)
#define BPNM_YAW    ((double) 1023.0 / (24.0 / 7.37 * 0.0200 * 84.0))

/* BIT -> 物理量 */
#define  bit_to_rad(ix,jnt,dir)  (((double) ix)*dir/BPR[jnt])

/* 物理量 -> BIT */
#define  Nm_to_bit(ru,jnt,dir)  (((double) ru)*dir*BPNM[jnt])
```

```
/*                                      Time-stamp: <04/08/11 17:53:50 hiroshi> */
/* define.h
   プログラム内で使われる主な定数を定義している。
 */

#define NOTHING       -1

/*  msg.command */
/*  feed側から見てSEND, RECVを定義 */
#define SYNC                0

#define START_TASK          1
#define STOP_TASK           2
#define STOP_WALK           3
#define START_WALK          4
#define START_STEP          5
#define PATTERN_0           6

#define GET_DX             10
#define GET_DV             11
#define GET_DU             12
#define GET_PX             13
#define GET_PV             14
#define GET_PU             15
#define GET_KX             16
#define GET_KV             17
#define GET_AD             18
#define GET_DA             19
#define GET_CS             20

#define PUT_DX             40
#define PUT_KX             41
#define PUT_KV             42

#define GET_GVAR           45
#define PUT_GVAR           46

#define DATA_OUT           49
#define STORE_DATA         50
#define GET_DATA_TIME      51
#define SAVE_DATA          52

#define GET_AVE_SAMPT      60
#define GET_AVE_FEEDT      61

/* Task ID */
#define FEED_TASK      0    /* not used */

/* FIFO ID */
#define DATA_BUF       1    /* feed -> main (Data Buffer) */
#define CTRL_BUF       4    /* main -> feed (Control Command) */

#define DATA_BUF_SIZE   (2048) /* fifo1は2Kbytesの領域を確保(データ用) */
#define CTRL_BUF_SIZE    (64)  /* fifo3に64bytesの領域を確保(命令用) */

#ifndef TRUE
#       define TRUE  1
#endif

#ifndef FALSE
#       define FALSE 0
#endif

#ifndef NOT
#       define NOT   !
#endif
```



```
#ifndef PI
#define PI    3.1415926543
#endif

/*#ifndef pi
#define pi    3.1415926543
#endif */

#define radian(deg) (double) ((deg) / 180.0 * PI)
#define degree(rad) (double) ((rad) / PI * 180.0)
```

```
/*                                      Time-stamp: <04/08/07 17:48:23 hiroshi> */
#include "define.h"

#define ADC_BASE_ADDRESS 0x180  /* ADC board base address */

#define COMMAND_FIFO 3 /* userプログラム→my_handler間のfifoナンバー */

/* ------------- マクロ定義 -------------- */

/* xが'NaN'でなければ0を返す */
#define isnan(x) ((x) != (x))

/* ------------- 定数の定義 -------------- */
#define feed_samp_msec       (1.0)   /* msec */

/* ストアするデータの最大数 */
#define MaxTime  3000

#define MaxDUTY  1023

typedef struct {
    int command;
    int task, period, ticks;  /* taskは使用されていない */
    float value;

    int leg, jnt;
} ModuleCommand;

/* 各プロセッサで共有される変数 */
/* moniが設定して、feedに転送する */
typedef struct {
  double u0max;   /* 簡略化している */
} gvarType;

enum joints    {hip, knee, yaw};
enum legs      {left, right};
enum phases    {stance, swing};
enum gyros     {pitch, roll};
enum contacts  {fore, hind};

#define num_joints   (yaw+1)
#define num_legs     (right+1)
#define gyro_axis    (roll+1)
#define num_contacts (hind+1)

typedef struct {          /* 状態量 */
    /* 角度 角速度 トルク */
    double  x,   v,   u;
} stateType;

typedef struct {          /* フィードバックゲイン */
    /* 位置  速度 */
    double k1,   k2;
} gainType;

typedef struct {                    /* 各ジョイントの状態を表す */
    stateType  present, last, desired;  /* 指令現在値  指令前回値  指令目標値 実測現
在値 実測前回値 */
    gainType   gain;                /* フィードバック・ゲイン */

    int ad, da;   /* エンコーダカウンタ値, PWMデューティ値 */

    int encodeDir, encodeCH;
    int motorDir, digital_board_num, motorCH_plus, motorCH_minus;

/*   encodeDir  : 関節角の方向 座標系と一致する:1 しない:-1
```



```
     motorDir   : モータの方向 座標系と一致する:1 しない:-1
     encodeCH   : エンコーダのチャネル番号
 */
} jointType;

typedef struct {                        /* 各接触センサの状態を表す */
    int ad;   /* エンコーダカウンタ値 */
    int encodeDir, encodeCH;
} contactSensorType;

typedef struct {
    double px[num_legs][num_joints];
    double pv[num_legs][num_joints];
    double pu[num_legs][num_joints];
    double dx[num_legs][num_joints];

    int cs[num_legs][num_contacts];
} dataType;

typedef struct {                        /* 各脚の状態を表す */
    jointType joint[num_joints];
    contactSensorType cs[num_contacts];

    enum phases phase;
} legType;

typedef struct{
    double gyro[gyro_axis];
    double gap_of_katamuki_by_drift[gyro_axis];
    double katamuki[gyro_axis];
    double filtering_katamuki[gyro_axis];
    int port_pitch;
    int port_roll;
    int incli_port[gyro_axis];
    double init_incli[gyro_axis];
} gyroType;


/* feed.c:大域変数．ここで宣言された変数をすべて初期化する必要がある */
typedef struct {
    legType leg[num_legs];
    gyroType gyro;

    int feedFlag, storeFlag, walkFlag;

    int it, data_time;
    double ave_sampt, ave_feedt; /* msec */

    double wt;  /* 関節目標角度計算のための時刻 (sec) */
    int stps;   /* step数 */

    /* 実験データ */
    dataType data[MaxTime+1];
} sharedType;

/* ------------- 関数のプロトタイプ宣言 -------------- */
/* nothing */
/* ------------- 外部変数の定義 -------------- */
/* nothing */
```

```
/*                                      Time-stamp: <04/08/11 19:11:16 yujioto> */
/* "calc.c"     feed.cでincludeされる */
/* ------------------------- Trajectory(軌道)計算 ---------------------------- */

#define T1 (double) 0.8  /* [sec] 歩行周期 */
#define T2 (T1 / 2)
#define DT (double) 0.1  /* [sec] 少し早めに遊脚を終端に動かす */
#define T3 (T2 - DT)

#define WT (T1 * 50)   /* Max. Walking Time */

#define THT_MAX (double) (10.0 * PI / 180.0)  /* 腰関節・角度振幅：θmax */
#define PHI_MAX (double) (10.0 * PI / 180.0)  /* 膝関節・角度振幅：φmax */
#define PHI_OFS (double) (-6.0 * PI / 180.0)  /* 膝関節・角度振幅：φoffset */

void stop_walking(void)
{
  enum legs leg; legType *p;

  s.walkFlag=FALSE;

  s.leg[left ].phase = stance;  s.leg[right].phase = stance;
  s.wt = 0.0;

  for (leg=left; leg<=right; leg++) {
    p = &s.leg[leg];
    p->joint[hip ].desired.x = 0.0;
    p->joint[knee].desired.x = PHI_OFS;
  }
}

/* ---------- WALKING (歩行) ------------ */
void walk_swing(enum legs leg)
{
  legType *p;  p = &s.leg[leg];

  if (s.wt < T3) {
    p->joint[hip ].desired.x = THT_MAX * sin(PI/T3*s.wt);
    p->joint[knee].desired.x = PHI_MAX * sin(PI/T3*s.wt);
  }
  else {
    p->joint[hip ].desired.x = THT_MAX;
    p->joint[knee].desired.x = 0.0;
  }
} /* walk_swing */

void walk_stance(enum legs leg)
{
  legType *p;  p = &s.leg[leg];

  p->joint[hip ].desired.x = THT_MAX * cos(PI/T3*s.wt);
  p->joint[knee].desired.x = 0.0;
} /* walk_stance */


/* ---------- STEPPING (足踏み) ------------ */
void step_swing(enum legs leg)
{
  legType *p;  p = &s.leg[leg];

  if (s.wt < T3) {
    p->joint[hip ].desired.x = THT_MAX * sin(PI/T3*s.wt);
    p->joint[knee].desired.x = THT_MAX * sin(PI/T3*s.wt) *2;
  }
  else {
    p->joint[hip ].desired.x = 0.0;
```



```
    p->joint[knee].desired.x = 0.0;
  }
} /* step_swing */

void step_stance(enum legs leg)
{
  legType *p;  p = &s.leg[leg];

  p->joint[hip ].desired.x = 0.0;
  p->joint[knee].desired.x = 0.0;
} /* step_stance */


/* --- 時間の管理，支持脚・遊脚の交換，関節目標角度の計算 --- */
void calc_traj(void)
{
  enum legs st, sw;

  if        ((s.leg[left ].phase == stance) && (s.leg[right].phase == swing)) {
    st = left;  sw = right;
  } else if ((s.leg[right].phase == stance) && (s.leg[left ].phase == swing)) {
    st = right;  sw = left;
  } else {
    rtl_printf("phase problem.\n");

    stop_walking();  return;
  }

  s.wt += FEED_SAMPT;  /* 時間を進める */

  if (s.wt > T2) { /* T2：歩行周期の半分 */
    /* 支持脚・遊脚の交換 */
    s.leg[st].phase = swing;  s.leg[sw].phase = stance;

/*  rtl_printf("stps=%d\n", s.stps); */
    s.wt = 0;  s.stps += 1;
    if ((T2 * s.stps) > WT) { stop_walking(); return; } /* WT：最大歩行時間 */
  }

  /* 関節目標角度の計算 */
  switch (s.walkFlag) {
  case START_STEP:
    step_swing (sw);  step_stance(st);
    break;
  case START_WALK:
    walk_swing (sw);  walk_stance(st);
    break;
  case PATTERN_0:
    /* nothing */
    break;
  }
} /* calc_traj */
```

```
/*                                      Time-stamp: <04/09/29 14:01:17 yujioto> */
/* "moni.c"     main.cでincludeされる */
/* ---------------------------------- Monitor ---------------------------------- */
/* ある関節のゲインを表示する */
void gainOf(enum legs leg, enum joints jnt)
{
#define gn p->gain
    printf(" k1:%9.2e k2:%9.2e",GET_FLOAT(GET_KX,leg,jnt),GET_FLOAT(GET_KV,leg,jnt));
}

/* ロボットの状態を表示する */
void displayState(void)
{
    enum legs   leg;
    enum joints jnt;

    printf("   |     ad   da |           present          |           desired
   |\n");
    printf("-------------------------------------------------------------------------
--\n");
    printf("                |     x        v        u    |     x        v        u
   |\n");

    for (leg=left; leg<=right; leg++) {
      printf("---------------------------------%s---------------------------------
-\n", legOf[leg]);
      printf("ContactSens [fore]=%3d, [hind]=%3d \n",
             (int) GET_FLOAT(GET_CS,leg,fore), (int) GET_FLOAT(GET_CS,leg,hind));

        for (jnt=hip; jnt<=yaw; jnt++) {
            printf("%5s ",jointOf[jnt]);
#define pre p->present
#define las p->last
#define des p->desired
            printf("%5d%5d",GET_INT(GET_AD,leg,jnt),GET_INT(GET_DA,leg,jnt));
            printf(" %9.2e %9.2e %9.2e",
                   GET_FLOAT(GET_PX,leg,jnt),GET_FLOAT(GET_PV,leg,jnt),GET_FLOAT(GET_
PU,leg,jnt));
            printf(" %9.2e %9.2e %9.2e",
                   GET_FLOAT(GET_DX,leg,jnt),GET_FLOAT(GET_DV,leg,jnt),GET_FLOAT(GET_
DU,leg,jnt));
            printf("\n");
        }
    }
} /* displayState */

#define SaveTime   MaxTime
/* -- feed側からSTOREされているデータを受けとる -- */
void recvData(void)
{
    int it;

    data_time = GET_INT(GET_DATA_TIME, 0, 0);
    printf("data_time=%d \n ave. samp_time(msec)=%5.3f ave. feed_time(msec)=%5.3f\n",
 data_time, GET_FLOAT(GET_AVE_SAMPT, 0, 0), GET_FLOAT(GET_AVE_FEEDT, 0, 0));

    for (it=0; it<data_time; it++) {
        Send_Com(ctl, SAVE_DATA, FEED_TASK, 0, 0, 0, it);
        Recv_Var (fd1, (double *)&data[it], sizeof(dataType));
    }
} /* recvData */

#define   TIME_ADJUST  (feed_samp_msec/100.0)

#define NUMBER 3 /* saveData()の前半でfp[]に確保した配列の数  */
```



```
/* 各脚、および各関節についてのデータをセーブする */
void saveData(enum legs leg)
{
    enum joints jnt;
    int it, j;
    char *dname = "Data/"; /* データを保存するディレクトリ */
    char *LEG[]   = {"LF", "LH", "RF", "RH", "O", NULL};
    char *JOINT[] = {"S", "E", "Y", "A", "T", "O", NULL};
    char *DATA[]  = {"x", "v", "u", "N_Tr", "ue", "uf", /*  "u0_e", "u0_f", */ /*  "p
ower", *//*  "duty", */ NULL};

    char leg_file[18], w_file[20];
    FILE *fp[NUMBER];

    /* -- SAVEする -- */
    printf("Saving %s's Data    data_time:%d ...\n", LEG[leg], data_time);

    for (jnt = hip; jnt <= yaw; jnt++) {
        strcpy(leg_file, dname);
        strcat(leg_file, LEG[leg]);
        strcat(leg_file, JOINT[jnt]);
        strcat(leg_file, ".");

        for( j = 0; j < NUMBER; j++) {
              strcpy(w_file, leg_file);
              strcat(w_file, DATA[j]); /* ファイル名にデータ判別の拡張子を付加 */

              if ((fp[j] = fopen(w_file, "w")) == NULL) {
                printf("MONI(saveData)>>Can't Open write-mode!! : %s\n", w_file);
              }
        }
        for (it = 0; it <= data_time; it++) {
              /* 各データの出力 */
              fprintf(fp[0], "%f %f\n", TIME_ADJUST*it, data[it].px[leg][jnt]);
              fprintf(fp[1], "%f %f\n", TIME_ADJUST*it, data[it].pv[leg][jnt]);
              fprintf(fp[2], "%f %f\n", TIME_ADJUST*it, data[it].pu[leg][jnt]);
        }
        for( j = 0; j < NUMBER; j++) {
              /* 開いたファイルを閉じる */
              fclose( fp[j]);
         }
    } /* for (jnt = .... */

} /* saveData */

/* コマンドを読み込んで実行する */
int monitor(void)
{
    displayState();

    printf("(d) display state\n(s) stepping\n(w) walking\n(t) stop\n(b) 0->Pattern\nd
ata buffer store\n(v) data save disk\n(q) quit\n");

    while(1){
        printf("Push!!");
        switch (getchar()) {
          case 'd':
            displayState();
            break;

          case 's':/* stepping : その場足踏み */
            Send_Com(ctl, STORE_DATA,  0, 0, 0, 0, 0);  /* データをストア */
            Send_Com(ctl, START_STEP,  0, 0, 0, 0, 0);
            break;

          case 'w':/* walking : 歩行 */
```

```
            Send_Com(ctl, STORE_DATA,  0, 0, 0, 0, 0);  /* データをストア */
            Send_Com(ctl, START_WALK,  0, 0, 0, 0, 0);
            break;

          case 't':/* 中止 */
            Send_Com(ctl, STOP_WALK,  0, 0, 0, 0, 0);
            break;

          case '0':/* Pattern 0 */
            Send_Com(ctl, PATTERN_0,  0, 0, 0, 0, 0);
            break;

          case 'b':/* データをバッファにストアする */
            Send_Com(ctl, STORE_DATA,  0, 0, 0, 0, 0);
            break;

          case 'v':/* データをディスクにセーブする */
            recvData();

            saveData(left);  saveData(right);
            printf("ALL DONE\n");
            break;

          case 'q':
            return(TRUE);
        }
    }
} /* monitor */
```

```
/*                                      Time-stamp: <04/08/11 18:01:07 hiroshi> */
#include <rtl.h>
#include <rtl_sched.h>
#include <rtl_fifo.h>

#include <time.h>
#include <math.h>
#include <linux/errno.h>

#include <titech-io.c>  /* pcdl0:/usr/src/rtlinux-3.0/titech-io-lib/titech-io.c */
#include "common.h"
#include "biped.h"

pthread_t tasks[1];

sharedType s;

double FEED_SAMPT=((double) (feed_samp_msec)/1000.0);
double BPR [yaw+1];  /* (bit/rad) */
double BPNM[yaw+1];  /* (bit/Nm)  */

TwIsaAttr twPort;

/* -- 神経振動子の計算に用いられる*定数* -- (ここでは使用しないので簡略化している by
hiroshi) */
gvarType g=
{
  /* u0max*/
  2.0
}; /* g */

/* すべてのモータにトルクを出力する */
void output2motor(void)
{
    enum legs leg;  enum joints jnt;  jointType *p;

    for (leg=left; leg<=right; leg++)
        for (jnt=hip; jnt<=yaw; jnt++){
            p  = &s.leg[leg].joint[jnt];

            /* Nm -> duty比 への計算*/
            p->da =   Nm_to_bit(p->present.u, jnt, p->motorDir);

            //出力の制限
            if      (p->da >  MaxDUTY)  p->da =  MaxDUTY;
            else if (p->da < -MaxDUTY)  p->da = -MaxDUTY;

            if(p->da > 0){
              TwAbzRcSCtrlSetPWM(&twPort, p->digital_board_num, p->motorCH_plus, p->d
a);
              TwAbzRcSCtrlSetPWM(&twPort, p->digital_board_num, p->motorCH_minus, 0);
            }
            else{
              TwAbzRcSCtrlSetPWM(&twPort, p->digital_board_num, p->motorCH_plus, 0);
              TwAbzRcSCtrlSetPWM(&twPort, p->digital_board_num, p->motorCH_minus, -p-
>da);
            }
        }
}

/*****************************************************************/
/*                          関数feed()関連                       */
/*****************************************************************/

/* ---------------- 初期化 ----------------------- */
#include "init.c"
```



```
void initialize(void)
{
    if(Init_TITech_Wire(&twPort) != 1);

    init_main();
    init_shared();
} /* initialize */

/* 時間を計測する関数。(*nowp - last)実時間を*ave_msecに代入 */
void calc_ave_msec(RTIME last, RTIME *nowp, double *ave_msec)
{
    double msec;

    *nowp = gethrtime();
    msec = ((double)(*nowp - last))/1000000.0;
    *ave_msec = (9.0 * *ave_msec + 1.0 * msec) / 10.0;
} /* calc_ave_msec */

/* ---------------- 制御 ------------------------- */

#include "calc.c"

/* データを１サンプリング分保存する */
void storeData(enum legs leg, int it)    /* 実験データのストア */
{
    enum joints jnt;  jointType *p;

    if (it > MaxTime) return;

    for (jnt=hip; jnt<=yaw; jnt++) {
        p = &s.leg[leg].joint[jnt];
        s.data[it].px[leg][jnt] = p->present.x;  /* 各関節の角度 */
        s.data[it].pv[leg][jnt] = p->present.v;  /* 各関節の角速度 */
        s.data[it].pu[leg][jnt] = p->present.u;  /* 実際に出力されたトルク */
        s.data[it].dx[leg][jnt] = p->present.x;  /* 各関節の角度目標値 */
    }

    s.data[it].cs[leg][fore] = s.leg[leg].cs[fore].ad;
    s.data[it].cs[leg][hind] = s.leg[leg].cs[hind].ad;
}/* storeData() */

/* 関節角の検出、それから関節角速度の計算 */
void sample(void)
{
    enum legs leg; enum joints jnt;  jointType *p;
    long  dataBuf[20];

    /*  エンコーダの値よむ */
    TwAbzRcSCtrlEnc32Input(&twPort, D1_NODE_ADDR, dataBuf);
/*  TwAbzRcSCtrlEnc32Input(&twPort, D2_NODE_ADDR, dataBuf+10); */  /* bipedではディジ
タルモジュールは一つ */

    /* 関節角度 */
    for(leg = left; leg <= right; leg++){
      for(jnt = hip; jnt <= yaw; jnt++){
        p = &s.leg[leg].joint[jnt];
        p->ad = dataBuf[p->encodeCH];

        /* 前回の角度 <- 現在の角度 */
        p->last.x    = p->present.x;

        p->present.x = bit_to_rad(p->ad,jnt,p->encodeDir);

        /* 差分による速度計算 */
        p->present.v = (p->present.x - p->last.x) / FEED_SAMPT;  /* 角速度(rad/s) */
```

```
      }

      /* 接触センサ */
      s.leg[leg].cs[fore].ad = dataBuf[s.leg[leg].cs[fore].encodeCH];
      s.leg[leg].cs[hind].ad = dataBuf[s.leg[leg].cs[hind].encodeCH];
    }
} /* sample */

/* 関節のフィードバック制御 */
void feedback(void)
{
    enum legs leg;  enum joints jnt;  jointType *p;

    if (!s.feedFlag) return;

    sample();

    if (s.walkFlag) calc_traj();

    for(leg=left; leg<=right; leg++) {
      for(jnt=hip; jnt<=yaw; jnt++){
        p = &s.leg[leg].joint[jnt];

        /* PD制御 */
        p->present.u = - p->gain.k1 * (p->present.x - p->desired.x)
                       - p->gain.k2 * (p->present.v - 0.0);
      }
    }

    output2motor();

    if (s.storeFlag) {
      if (s.data_time < MaxTime) {
          storeData(left , s.data_time);    /* 実験データのストア */
          storeData(right, s.data_time);    /* 実験データのストア */
          s.data_time++;
      }
      else
        s.storeFlag = FALSE;
    }
}/* feedback */


/*****************************************************************/
/*                        FIFO経由の通信関連                     */
/*****************************************************************/

/* Float型データを一つmain側に送る */
/* fifoにはFloat型データが直接入る */
void RT_SEND_FLOAT(float value)
{
    float tmp;

    tmp = value;  rtf_put(DATA_BUF, &tmp, sizeof(float));
} /* RT_SEND_FLOAT */

/* doubleで構成された大きさsizeのデータをmain側に送る */
void Send_Var(double *p, int size)
{
    int i;

    for (i=0; i<(size/sizeof(double))/*これは、sizeの中の変数の数を表す*/; i++) {
        RT_SEND_FLOAT((float)*p);
        p++;
    }
} /* Send_Var */
```



```
/* Realtime thread */
/* my_handler→thread0時の通過fifoのnumber */
#define TASK_CONTROL_FIFO_OFFSET 4
void *thread_code(void *t)
{
    legType *pl;  jointType *pj;

    int fifo = (int) t;
    int taskno = fifo - 1;
    int err, i;
    int ret;

    float *p;

    ModuleCommand msg;

    RTIME t0, t1, t2;

    t1 = gethrtime();

    while (1) {
      /* 待ち時間にはいって、他のthreadに切り替わる */
      ret = pthread_wait_np();

      /*スレッドにアタッチされたfifoにデータがあれば、commandなどを取り出す。なければ
、このif内は無視する*/
      if ((err = rtf_get (taskno + TASK_CONTROL_FIFO_OFFSET, &msg, sizeof(msg))) == s
izeof(msg)) {
        pl = &s.leg[msg.leg];  pj = &pl->joint[msg.jnt];

        switch (msg.command) {
              case GET_DX: RT_SEND_FLOAT((float) pj->desired.x);  break;
              case GET_DV: RT_SEND_FLOAT((float) pj->desired.v);  break;
              case GET_DU: RT_SEND_FLOAT((float) pj->desired.u);  break;
              case GET_PX: RT_SEND_FLOAT((float) pj->present.x);  break;
              case GET_PV: RT_SEND_FLOAT((float) pj->present.v);  break;
              case GET_PU: RT_SEND_FLOAT((float) pj->present.u);  break;
              case GET_KX: RT_SEND_FLOAT((float) pj->gain.k1);    break;
              case GET_KV: RT_SEND_FLOAT((float) pj->gain.k2);    break;
              case GET_AD: RT_SEND_FLOAT((float) pj->ad);         break;
              case GET_DA: RT_SEND_FLOAT((float) pj->da);         break;
              case GET_CS: RT_SEND_FLOAT((float) pl->cs[msg.jnt].ad); break;

              case PUT_DX: pj->desired.x = msg.value;             break;

              case GET_AVE_SAMPT:  /* 平均サンプリング時間(msec) */
                RT_SEND_FLOAT((float) s.ave_sampt);                 break;
              case GET_AVE_FEEDT:  /* 平均FEEDタスク実行時間(msec) */
                RT_SEND_FLOAT((float) s.ave_feedt);                 break;
              case GET_DATA_TIME:  /* 保存されたデータ数をmain側に送る */
                RT_SEND_FLOAT((float) s.data_time);                 break;

              case START_TASK:
                pthread_make_periodic_np(pthread_self(), gethrtime(), msg.period);

                init_PWM_value(&twPort);
                s.feedFlag=TRUE;

                rtl_printf("Feed thread is started.\n");
                break;

              case STOP_TASK:
                init_PWM_value(&twPort);
                pthread_suspend_np(pthread_self());
                s.feedFlag=FALSE;
```

```
                break;

              case START_WALK:
              case START_STEP:
              case PATTERN_0:
                s.walkFlag = msg.command;  /* START_WALK or START_STEP */
                s.wt = 0.0;  s.stps = 0;
                s.leg[left ].phase = swing;      /* 左脚から振り出す */
                s.leg[right].phase = stance;

                rtl_printf("Step or Walk is started.\n");
                break;

              case STOP_WALK:
                  stop_walking();

                  rtl_printf("Step or Walk is stopped. stps=%d\n", s.stps);
                break;

              case STORE_DATA:
                s.storeFlag = TRUE;  s.data_time = 0;
                break;

              case SAVE_DATA:       /* 保存されたデータの一部(it=ticks)をmain側に送る
 */
                Send_Var((double *)&s.data[msg.ticks], sizeof(dataType));
                break;

              case GET_GVAR:       /* gvarTypeをmain側に送る */
                Send_Var((double *)&g, sizeof(gvarType));
                break;

              case PUT_GVAR:       /* gvarTypeをmain側から受けとる */
                p = (float *)&g;
                for (i=0; i<msg.jnt; i++) p++;
                *p = msg.value;
                break;

              default:
                rtl_printf("RTLinux task: bad command\n");
                return 0;
            }/* switch */
          }/*  if */

          t0 = t1;
          /*【1】*/
          /* 前回の【1】から今回の【1】までにかかった時間を計っている。
             歩行後の(v)data store時、ディスプレイでチェックするだけ */
          calc_ave_msec(t0, &t1, &s.ave_sampt);

          /* フィードバック */
          feedback();

          /*  【1】からここまでにかかった時間を計っている.
              歩行後の(v)data store時、ディスプレイでチェックするだけ  */
          calc_ave_msec(t1, &t2, &s.ave_feedt);
        }
        return 0;
    } /* thread_code */

    int my_handler(unsigned int fifo)
    {
        ModuleCommand msg;
        int err;

        /* main.c→handler間fifo(COMMAND_FIFO)からコマンドを取り出す */
```



```
        while ((err = rtf_get(COMMAND_FIFO, &msg, sizeof(msg))) == sizeof(msg)) {
          /* handler→指定されたthreadにアタッチされたfifoにコマンドを出力する */
          rtf_put (msg.task + TASK_CONTROL_FIFO_OFFSET, &msg, sizeof(msg));
          /* コマンドが入力されたfifoがアタッチされているthreadをたたきおこす */
          pthread_wakeup_np (tasks [msg.task]);
        }
        if (err != 0) {
          return -EINVAL;
        }
        return 0;
    } /* my_handler */

    /* #define DEBUG  */
    int init_module(void)
    {
            int c[4];
            pthread_attr_t attr;
            struct sched_param sched_param;
            int ret;

            rtf_destroy(1);
            rtf_destroy(2);
            rtf_destroy(3);
            rtf_destroy(4);
            c[0] = rtf_create(1, 6000);
            c[1] = rtf_create(2, 6000);
            c[2] = rtf_create(3, 6000/*  200 */);   /* input control channel */
            c[3] = rtf_create(4, 4000/*  100 */);   /* input control channel */

            pthread_attr_init (&attr);
            sched_param.sched_priority = 4;
            pthread_attr_setschedparam (&attr, &sched_param);
            ret = pthread_create (&tasks[0],  &attr, thread_code, (void *)1);

            rtf_create_handler(3, &my_handler);

            /* TITech-Wire、各変数などの設定と初期化 */
            initialize();

            return 0;
    } /* init_module */

    void cleanup_module(void)
    {
            release_TITech_Wire(&twPort);

            rtf_destroy(1);
            rtf_destroy(2);
            rtf_destroy(3);
            rtf_destroy(4);

            pthread_cancel (tasks[0]);
            pthread_join (tasks[0], NULL);
    } /* cleanup_module */
```

```
/*                                      Time-stamp: <04/08/07 15:03:36 hiroshi> */
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <sys/time.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <rtl_fifo.h>
#include <rtl_time.h>
#include "common.h"
#include "biped.h"

int fd1, ctl;         /* FIFOのファイル記述子 */
char *jointOf[] = {"HIP ", "KNEE", "YAW ", NULL};
char *legOf[]   = {"Left ", "Right", NULL};
dataType data[MaxTime+1];
/*  float data_frank[8]; */

gvarType g;  /* feed.cの方に実体があるので，受信する */
int data_time;

float buf_frank[72];

/* ------------------------ Communication With Feed ------------------------ */
void Send_Com(int ctl, int command, int task, int period, int leg, int jnt, int it)
{
  ModuleCommand msg;
  msg.command = command;  msg.task = task;  msg.period = period;  msg.leg = leg;  msg
.jnt = jnt;  msg.ticks = it;

  if (write(ctl, &msg, sizeof(msg)) < 0) {
      fprintf(stderr, "Can't send a command to RT-task handler\n");
      exit(1);
  }
} /* Send_Com */

#define BUFSIZE 32
/* Float型データを一つだけfeed側から受ける */
/* fifoにはFloat型データが直接入る */
float Recv_Float(int fd)
{
  int v;  float *fp;  char buf[BUFSIZE];  /* fifoからの値のバッファー */

  v  = read(fd, buf, sizeof(float));      /* Float型データを一つづつ読む */
  fp = (float *)buf;   return(*fp);
} /* Recv_Float */

/* doubleで構成された大きさsizeのデータをfeed側から受ける */
void Recv_Var(int fd, double *p, int size)
{
    int i;

    for (i=0; i<(size/sizeof(double)); i++) {
        *p = (double)Recv_Float(fd);
        p++;
    }
} /* Recv_Var */

/* --------- Command and GET&PUT_VALUE ---------- */

float GET_FLOAT(int com, int leg, int jnt)
{
    Send_Com(ctl, com, FEED_TASK, 0, leg, jnt, 0);  return(Recv_Float(fd1));/*  このf
d1はあっとん? */
```



```
} /* GET_FLOAT */

#define GET_INT(com,leg,jnt)  ((int)GET_FLOAT(com,leg,jnt))
#include "moni.c"

int main()
{
    ModuleCommand msg;/*  struct my_msg_struct msg; */

    /* Real time thread1 -> Use application */
    if ((fd1 = open("/dev/rtf1", O_RDONLY)) < 0) {fprintf(stderr, "Error opening /dev
/rtf1\n"); exit(1);}
    /* User application -> Handler */
    if ((ctl = open("/dev/rtf3", O_WRONLY)) < 0) {fprintf(stderr, "Error opening /dev
/rtf3\n"); exit(1);}

    /* now start the tasks */
    Send_Com(ctl, START_TASK, FEED_TASK, feed_samp_msec * 1000000/* [us] */, 0, 0, 0)
;

    /* センサやモータの出力値の表示を行う。その後、歩行スタート、データのファイル書き
込みなどの命令待ち状態になる */
    monitor();

    /* stop the tasks */
    msg.command = STOP_TASK;
    msg.task = FEED_TASK;
    if (write(ctl, &msg, sizeof(msg)) < 0) {
      fprintf(stderr, "Can't send a command to RT-task\n");
      exit(1);
    }
    return 0;
}
```

```
/*************** 傾斜センサ,ジャイロセンサの単位は「ラジアン」 ********************
***********/
/************************** 傾斜センサの傾きを検出 ********************************
******/
void sample_InclinationSensor(int incli_pitch_AD, int incli_roll_AD){

  double incli_pitch = 0.0, incli_roll = 0.0;
  s.last_inclination_sensor[pitch] = s.inclination_sensor_LPF[pitch];
  s.last_inclination_sensor[roll] = s.inclination_sensor_LPF[roll];

  incli_pitch = ((double)incli_pitch_AD - 2082.0);
  if(incli_pitch > 0)
    s.inclination_sensor[pitch] = radian ( incli_pitch * 0.04673 ); /* ラジアンに変換
 */
  else
    s.inclination_sensor[pitch] = radian ( incli_pitch * 0.04739 ); /* ラジアンに変換
 */

  /* ローパスをかける */
  if(s.sensor.inclination_count < 150)
    s.inclination_sensor_LPF[pitch] = s.inclination_sensor[pitch];
  else{
    s.inclination_sensor_LPF[pitch] =  s.inclination_sensor[pitch];
    s.inclination_sensor_LPF[pitch] = (s.last_inclination_sensor[pitch]*700 + s.incli
nation_sensor_LPF[pitch]*1)/701.0;
  }

  incli_roll = ((double)incli_roll_AD - 2184.0);
  if(incli_roll > 0)
    s.inclination_sensor[roll] = radian ( incli_roll * 0.04556 ); /* ラジアンに変換 *
/
  else
    s.inclination_sensor[roll] = radian ( incli_roll * 0.04975 ); /* ラジアンに変換 *
/

  /* ローパスをかける */
  if(s.sensor.inclination_count < 230)
    s.inclination_sensor_LPF[roll] = s.inclination_sensor[roll];
  else{
  s.inclination_sensor_LPF[roll] =  s.inclination_sensor[roll];
  s.inclination_sensor_LPF[roll] = (s.last_inclination_sensor[roll]*700 + s.inclinati
on_sensor_LPF[roll]*1)/701.0;
  }

  s.sensor.inclination_count ++;

} /* void sample_InclinationSensor */

/****************************** ジャイロセンサから角速度s.sensor.gyro[]を検出 ******
************************/

void sample_gyro(int gyro_pitch_AD, int gyro_roll_AD)
{
  //s.sensor.gyro_AD[pitch] = (double)gyro_pitch_AD;
  //s.sensor.gyro_AD[roll]  = s.sensor.init_gyro[pitch]; /*data[it].gyro_pitch_AD[rol
l]にはジャイロピッチ軸の初期値を入れた!!!*/

  /* ジャイロのピッチの座標系の向きが傾斜系のそれと反対 */
    s.sensor.gyro[pitch] = (s.sensor.init_gyro[pitch] - ((double)gyro_pitch_AD))  / 4
.40  / 0.67 ; /* 増幅率の理論値は6.10 検討すべき!!!*/
    s.sensor.gyro[roll]  = (((double)gyro_roll_AD)    - s.sensor.init_gyro[roll]) / 5
.67  / 0.67 ; /* 増幅率の理論値は7.67 検討すべき!!!*/
}

/**** ジャイロセンサの初期角速度を検出,ジャイロセンサからの角速度を積分して傾きs.sens
or.gyro_slope[]を求める ****/
```



```
void integrate_gyro(int gyro_pitch_AD, int gyro_roll_AD)
{
    enum sensors sensor;
    RTIME last_time = 0;
    double delta_time = 0.0, delta_sita  = 0.0;
    double last_gyro[num_sensors];
    double last_gyro_slope[3]={0.0, 0.0, 0.0};

    if(s.motionFlag == INIT_GYRO){/* 静止している状態で、積分するための初期角速度init
_gyro[]を求める。よって、この状態では動かしてはならない。 */

            if(s.sensor.gyro_count < 100) /* ジャイロの初めの値はノイズがのっているの
で計算しない */ //keisuke1008 50
              s.sensor.gyro_count++;

            else if(s.sensor.gyro_count > 99){

              s.sensor.gyro_AD[pitch] = (double)gyro_pitch_AD;
              s.sensor.gyro_AD[roll]  = (double)gyro_roll_AD;

              for(sensor = pitch; sensor <= roll; sensor++){

                s.sensor.inclination_sum[sensor] += s.inclination_sensor[sensor];
                s.sensor.init_inclination[sensor] = s.sensor.inclination_sum[sensor]
/ (double)(s.sensor.gyro_count - 99);/* inclination sensor */
                s.sensor.gyro_sum[sensor] += s.sensor.gyro_AD[sensor];
                s.sensor.init_gyro[sensor] = s.sensor.gyro_sum[sensor] / (double)(s.s
ensor.gyro_count - 99);/* gyro sensor */

              }/* for */
              s.sensor.gyro_count++; /* feed_samp_msec毎にカウントされる */
            }/* else if */
    }/* if */

    else if(s.motionFlag == READY_POSITION){/* 積分を開始して胴体角度を計算する */

      for(sensor = pitch; sensor <= roll; sensor++){
        last_gyro[sensor] = s.sensor.gyro[sensor];
        s.sensor.last_gyro_slope[sensor] = s.sensor.gyro_slope_LPF[sensor];
}
        sample_gyro(gyro_pitch_AD, gyro_roll_AD);
        last_time = pre_time;
        pre_time = gethrtime();

      for(sensor = pitch; sensor <= roll; sensor++){

        //delta_time = ((double)(pre_time - last_time))/1000000.0;// (nsec)から(msec)
へ//gontan0929
        delta_time = ((double)(pre_time - last_time))/1000000000.0;// (nsec)から(sec)
へ//yujioto0224
        delta_sita = (s.sensor.gyro[sensor] + last_gyro[sensor]) / 2.0 * delta_time ;
//gontan0929
        last_gyro_slope[sensor] = s.sensor.gyro_slope[sensor];
        s.sensor.gyro_slope[sensor] += radian ( delta_sita );

        /* ローパスをかける */
        if(s.sensor.gyro_count2 < 100)
        s.sensor.gyro_slope_LPF[sensor] = s.sensor.gyro_slope[sensor];
        else{
          s.sensor.gyro_slope_LPF[sensor] = s.sensor.gyro_slope[sensor];
          s.sensor.gyro_slope_LPF[sensor] = (s.sensor.last_gyro_slope[sensor]*100 + s
.sensor.gyro_slope_LPF[sensor]*1)/101.0;
        }
        /* ドリフト値の計算 */
        s.sensor.drift_value[sensor] = s.sensor.gyro_slope_LPF[sensor] - s.inclinatio
n_sensor_LPF[sensor];
```

```
        /* 胴体角度の補正 */
        s.sensor.revised_gyro_slope[sensor] = s.sensor.gyro_slope[sensor] - s.sensor.
drift_value[sensor];

        if(s.gyroFlag == TRUE){/* 積分開始時に傾斜センサの値をジャイロの初めの値に代
入する */
             s.sensor.gyro_slope[sensor] = s.sensor.init_inclination[sensor];
             s.sensor.gyro_slope_LPF[sensor] = s.sensor.init_inclination[sensor];
          } /* if */

          if(isnan(s.sensor.gyro_slope[sensor]) != 0)
            s.sensor.gyro_slope[sensor] = last_gyro_slope[sensor];
        } /* for */
        s.gyroFlag = FALSE;
    } /* else if */
    s.sensor.gyro_count2++;
} /* void integrate_gyro */

/************************************  接触センサ(スイッチ)  ********************
*********************************/

void sample_contact_sensor_switch(int CSS_LF_AD, int CSS_LB_AD, int CSS_RF_AD, int CS
S_RB_AD){

  int threshold_CSS = 2500;
  //#if 0
  if           (CSS_LF_AD >= threshold_CSS) s.sensor.CSS_LF = OFF;
  else if      (CSS_LF_AD <  threshold_CSS) s.sensor.CSS_LF = ON;

  if           (CSS_LB_AD >= threshold_CSS) s.sensor.CSS_LB = OFF;
  else if      (CSS_LB_AD <  threshold_CSS) s.sensor.CSS_LB = ON;

  if           (CSS_RF_AD >= threshold_CSS) s.sensor.CSS_RF = OFF;
  else if      (CSS_RF_AD <  threshold_CSS) s.sensor.CSS_RF = ON;

  if           (CSS_RB_AD >= threshold_CSS) s.sensor.CSS_RB = OFF;
  else if      (CSS_RB_AD <  threshold_CSS) s.sensor.CSS_RB = ON;

  //#endif
  //  s.sensor.CSS_L = (double) CSS_L_AD;
  //  s.sensor.CSS_R = (double) CSS_R_AD;
}

/************************************************  床反力  ***********************
*****************************/

void sample_floor_reaction_force(int FRF_L_AD, int FRF_R_AD){

  s.sensor.FRF_L = (double)FRF_L_AD / (316.0 / 9.8);
  s.sensor.FRF_R = (double)FRF_R_AD / (344.0 / 9.8);

}
```

```
##                                  Time-stamp: <04/08/07 15:03:17 hiroshi>
IOLIB = /usr/src/rtlinux-3.0/titech-io-lib

all: feed_module.o main

include /usr/src/rtlinux-3.0/rtl.mk

main: main.c moni.c common.h define.h biped.h
        $(CC) ${INCLUDE} ${USER_CFLAGS} -O2 -Wall main.c -o main

#カーネル空間のプロセスはライブラリとダイナミックリンクできないので、こうやって静的リンクしてやらないといけない。
#ここではlibm.aにリンクしてやる
feed_module.o: feed.o
        ld -r -static -o feed_module.o feed.o -L/usr/lib/ -lm

feed.o: feed.c calc.c init.c common.h define.h biped.h
        $(CC) ${INCLUDE} ${CFLAGS} -I${IOLIB} -c feed.c -o feed.o

clean:
        rm -f main *.o

include $(RTL_DIR)/Rules.make
```

関連発表

学術雑誌

Otoda, Y, Kimura, H and Takase, K, Construction of Gait Adaptation Model in Human Splitbelt Treadmill Walking Using a 2D Biped Robot, Advanced Robotics, (in press), 2009

国際会議における発表 (発表者に○印を付す)

○ Otoda, Y, Kimura, H, Takase, K and Songmin, Jia, Adaptive Walking of a 2D Biped Robot during Splitbelt Treadmill, SICE Annual Conference International Conference on Instrumentation Control and Information Technology, University of Electro–Communications, 8.2008

国内学会・シンポジウム等における発表

○ 永本，音田，木村，人のスプリットベルト・トレッドミル歩行制御モデルの構築，SI2008，長良川国際会議場，12.2008

○ 音田，木村，高瀬，人のスプリットベルト・トレッドミル歩容適応モデルの提案，移動知シンポジウム，松島大観荘，3.2009

著者略歴

音田　裕史（おとだ　ゆうじ）

1999 年　3 月　東京都立南多摩高等学校　卒業

2000 年　4 月　成蹊大学工学部電気電子工学科　入学

2004 年　3 月　成蹊大学工学部電気電子工学科　卒業

2004 年　4 月　電気通信大学大学院情報システム学研究科

情報システム運用学専攻　修士課程　入学

2006 年　3 月　電気通信大学大学院情報システム学研究科

情報システム運用学専攻　修士課程　修了

2006 年　4 月　電気通信大学大学院情報システム学研究科

情報システム運用学専攻　博士課程　入学

2009 年　3 月　電気通信大学大学院情報システム学研究科

情報システム運用学専攻　博士課程　修了

2009 年　4 月　国立障害者リハビリセンター研究所

障害工学研究部　流動研究員

日本ロボット学会会員