ロボットにおけるセンサとマルチプロセッサ並列信号処理に関する研究

北川　秀夫

ロボットにおけるセンサとマルチプロセッサ並列信号処理に関する研究

北川 秀夫

目　次

第1章　序論

1.1　ロボットにおける情報処理

現在，ロボットに要求される作業は日増しに高度になっており，この要求を実現するために，ロボット自身に認識や判断の機能を持たせる知能化の研究が進められている．当初，ロボットの知能化に関する研究は，ロボットビジョンやエキスパートシステムといった人工知能（AI）の研究として始まった．AI手法では，外界の情報をロボットの知能である中枢コンピュータで集中的に認識する方法[1, 2]を用いるので，ロボットには外界の知識や認識・行動のアルゴリズムなどを，予めルールとして与えておかなければならないが，どんな状況にも対応できるルールを矛盾なく記述することは不可能であることが指摘されている．最近，ロボット自身が様々な環境での行動から，後天的にルールを獲得する手法が提案された．すなわち，「それぞれ自分の認識と判断で勝手に動こうとする独立な多数の要素行動間の競合・強調」の結果として知能ロボットの動作を表現する方法[3, 4]で，サブサンプションアーキテクチャと名付けられている．この手法では，問題を知覚，モデル化，計画，運動制御といった機能モジュールで分割するのではなく，衝突回避，探索，地図作成といった独立なエージェントの集合体として能力レベル別に分割し，その階層構造で制御システムを構成している．このような手法を実現するための技術としては，AIのトップダウン的な教示だけではなく，ニューラルネットワーク，ファジィ理論，遺伝アルゴリズムといったボトムアップ的な知識獲得のための技術に期待が寄せられ，それらの応用を目指した多数の研究が行われている．

このように，ロボットの知能化のために，各種アーキテクチャが提案されているが，基本的にはセンサによって得られる外界の情報を基に，アクチュエータを駆動して外界に働きかけるという所に共通点を見出すことが出来る．従って，ロボットの知能化のためには，センサ，コンピュータ，アクチュエータの各要素技術の高性能化，または新しいアルゴリズムの開発等による情報処理能力の向上が不可欠である．

後者の情報処理能力について，センサに関しては，センサ単体のハードウェアの性能を上げるのではなく，複数のセンサと情報処理によって能力を向上させようという研究が盛んに行われている[5]．しかし，一般にロボットに装着されたセンサの測定範囲，分解能を向上することには，計算量の増加が伴う．さらに，センシングによって対象に関するより高度な知識を得ようとした場合，装着するセンサの数が増加し

たり，センサ出力信号から情報を引き出すための信号処理が複雑になりがちである．その結果，必要とされる計算量の増加はさらに顕著になり，リアルタイムでの情報処理が困難になる傾向がある．

また，情報処理能力について，アクチュエータに関しては，マニピュレータの位置，力の制御等のために必要となる計算量の増加が問題となる．マニピュレータに，前述のセンサ情報等に基づいたフレキシブルな動作を行わせるためには，リアルタイムで逆運動学方程式，逆動力学方程式等を計算する必要があるが，それらの計算には多大な計算量が要求されるため，通常のマイクロコンピュータを用いただけでは満足する結果を得ることができない．

これらのリアルタイム制御を達成するためには，高速計算可能な信号処理アルゴリズムの開発及び高速処理を行うコンピュータシステムの利用が必要である．本研究では，ロボットにおける情報処理に焦点をあて，把握制御用センサ及びマルチプロセッサ並列処理手法の開発を行った．

1.2 ロボットの把握制御用センサ

ロボットのセンサは主に外界センサ，内界センサ及び効果器センサに分類されている [5]．外界センサとは，ロボットの動作する環境を認識するためのセンサで，画像センサ，距離センサがその代表である．内界センサとは，ロボット自身の状態を知るためのセンサで，ロボットの各関節に装着したエンコーダ，タコメータがその代表である．そして，効果器センサとは，対象物とロボットの手先との相対的な状態を知るために，手先効果器（ロボットハンドのように，対象物に対して作業を行うため，マニピュレータの先端に装着する機器）に装着するセンサで，触覚センサ，力覚センサがその代表である．本研究では，効果器センサの中で，未知物体を把握する際に使用するセンサの開発を行った．

ロボットの基本動作の一つに，把握動作がある．重量，形状，材質が未知の対象物体を把握する動作を確実に行うためには，把握を行う前に，ロボットハンドの座標系で対象物体の位置・姿勢が分かっていることが必要である．さらに，把握を行う際に，把握力が弱すぎて物体をすべり落としてしまったり，逆に把握力が強すぎて，物体に損傷を与えてしまったりすることのないように，把握力を制御することが必要とされる．

それらの動作を達成するために，

1) 視覚センサ，触覚センサ，近接覚センサ等を用いて，未知把握対象物体の位置・姿勢を検出し，物体の把握位置を決定する

2) スリップセンサを用いてすべりを検出し，把握力制御を行う

という二段階の方法が考えられている．それぞれの段階について，現在までの研究状況を以下にまとめる．

1) の物体の位置・姿勢の検出方法については，現在までに以下のような方法が提案されている．

視覚センサを用いる方法の代表的なものは，複数のカメラを用いて物体の特徴点を検出して，それを対応づけることによって距離計測を行い，物体の位置・姿勢を判別するものである [6]．この方法では，複数の画像間の対応点を決めることが容易でないため，大まかな位置・姿勢判別には向いているが，組立作業などの高度な作業には，精度的に問題が多い．また，装置が大がかりで複雑になるという欠点もある．複数のカメラによる三次元認識を簡略化する手法として，スリットを通した垂直の光を物体に投影し，物体上の反射光をカメラで観測することによって，物体上のスリット光の三次元位置を求める方法がある [7]．この方法は，高速に多数の点を入力するのに適するが，装置が大きいという欠点を持つ．

触覚センサを用いる方法 [8] は，ロボットハンドのフィンガーの内側にセンサを取り付け，物体に触れた時のハンドの位置から，物体の位置を求めるものである．これは接触型のセンサであるため，物体に触れた際に動かしてしまったり，損傷を与えてしまったりすることが考えられる．

近接覚センサのうち，オンオフタイプのセンサを用いる方法 [9] は，ロボットハンドの先端にセンサをつけて，物体に接近した時のオンオフの出力により，物体の位置・姿勢を求める方法である．このセンサは距離情報を用いないため，位置・姿勢の検出精度に問題がある．また，近接覚センサのうち，距離センサを用いる方法 [10, 11] は，センサを取り付けたマニピュレータを動かして，物体面を構成する各点の距離を検出することにより，物体の全体の形状，位置・姿勢を検出する方法である．一回の計測で一点の位置しかわからないため，物体の位置・姿勢を判別するには時間がかかるが，ロボットの自由度を利用することにより物体の全体の形状を知ることができる．

2) のすべりを検出するスリップセンサは，接触計測を行う方法と非接触計測を行う方法に分類することができる．それぞれのタイプについて，現在までに以下のような方法が提案されている．

接触計測を行う方法には，振動検出型 [12, 13]，ボール回転型 [14]，回転ローラ型 [15, 16, 17, 18, 19]，ディンプルボール型 [20]や，分布型触覚センサを用いた方法 [21] がある．これらは，センサを直接把握物体に押し当てなければならず，その押し当てる力が強すぎる場合には，物体に損傷を与えてしまい，また，押し当てる力が弱い場合には，正確にすべりを検出することができない．さらに，物体の形状によって，大きく左右されるという欠点も持つ．

非接触計測を行う方法には，超音波を利用した方法 [22] や，空間フィルタ法 [23]，二点間相互相関法 [24]，レーザ・ドップラ法 [25]を利用した速度計がある．超音波を利用した方法は，方向，変位の計測が困難である．空間フィルタ法を用いる方法は，分解能の点で問題がある．二点間相互相関法を用いる方法は，計測する二つの地点を，物体がすべる方向に合わせないと計測できない，すなわち，斜め方向のすべりに対

応することができないという点で問題がある．また，外部光等測定環境の影響を受け易いという点がある．レーザ・ドップラ法は，ドップラ周波数がレベル変動とノイズの影響を強く受けるという性質を持つため，信号処理が複雑となる．また，レーザ光として，時間的，空間的に高いコヒーレンスが必要であるため，安価な半導体レーザでは波長安定度が問題となる．

物体の位置・姿勢検出及びすべり検出を，別々のセンサで行うことを考えると，センサのシステムが大がかりになる．また，ハンドに装着できるセンサの大きさ，個数には制限があるため，装着することができないといった問題も起こりうる．従って，物体の位置・姿勢検出とすべり検出を共通のセンサで行うことが出来ると有効である．

近年，半導体レーザの自己混合効果を用いて，物体の速度と距離の両方を計測できるセンサ [26, 27, 28] が研究されている．このセンサは，半導体レーザダイオード，内蔵のフォトダイオード，及び物体に投光するためのレンズより構成される．速度をもつ物体に対して，半導体レーザ光を投光すると，ドップラ偏移された戻り光が再びレーザダイオード内に入る．自己混合効果により内蔵のフォトダイオードにビート信号が現れるので，これを信号処理することにより速度が求められる．また，三角波電流でＦＭ変調すると物体によって構成される外部共振器でモードホップが生じるので，このモードホップの時間間隔から距離を求めることができる．しかし，このセンサの距離測定範囲は，10cm から 80cm 程度 [28] であるので，フィンガーに取り付けて把握動作の制御を行う用途には適していない．

本論文では，ロボットハンドのフィンガーへの装着が可能で，単体ですべり及び距離検出を両方行うセンサを開発し，その信号処理アルゴリズムに関する検討を行う [29, 30, 31]．

1.3 センサ出力の高速信号処理

ロボットのセンサを実用化するための条件として，実時間性があげられる．特に，広い測定範囲，高分解能を要求する場合，あるいは複数のセンサ信号を処理する場合に計算量が増加し，リアルタイム制御が困難になる傾向がある．

リアルタイム制御を達成するためには，高速計算可能な信号処理アルゴリズムの開発，及び高速処理を行うコンピュータシステムの利用が必要である．後者については，DSP などの高速演算プロセッサによる計算，または複数のプロセッサを用いた並列処理などの手法があげられる．このうち，複数のプロセッサを用いた並列処理手法について次節に述べる．

1.4 並列処理によるロボットの逆運動学及び逆動力学計算の高速化

ロボットに要求される作業が複雑かつ高精度になるにつれ，その制御に必要な計算量は増大する．そのような制御計算をリアルタイムで処理するためには，高速処理能力を持つコンピュータが必要になる．しかし，そのような高価なコンピュータを，実用的なロボットコントローラに搭載する事は困難であるため，安価なマイクロプロセッサを複数用いた並列処理技術を導入し，価格性能比の優れた制御システムを開発することが期待されている．

並列処理計算機の構成方式としては，演算パイプライン方式，マルチプロセッサ方式，データフロー方式などがあげられるが，パイプライン・コンピュータは小マトリクス・ベクトル演算が苦手であり，データフローマシンは，その非決定的なデータ駆動型実行のために，リアルタイム・オンライン制御に要求されるサンプリング・タイムの管理が困難である．従って，リアルタイム・ロボット制御には，マルチプロセッサ方式が最も適していると考えられる [32]．

マルチプロセッサによる並列処理には，以下のような問題点があげられる．まず，与えられた問題を，いかに定式化するかという解法の決定については，並列処理で特に問題となる，通信量や通信のオーバヘッドを考慮した，解の導出アルゴリズムが明らかにされていない．次に，与えられた問題を，いかに処理単位であるタスクに分割するかというタスク分割については，大まかにタスク粒度を選択する方法が開発されているだけであり，与えられた解法を，処理時間の短いタスク集合に分割する手法の開発が不十分である．さらに，スケジューリングについては，従来の方法では，組合せ最適化問題の複雑さを軽減するために，プロセッサ間のデータ通信に要する時間が無視されていたが，ハードウェアの進歩，乗算器の内蔵等により，マイクロプロセッサの演算能力が飛躍的に向上しているため，相対的にプロセッサ間のデータ転送によるオーバーヘッドが無視できない状況になっている．これらの問題に関して以下に詳細に述べる．

マニピュレータの逆動力学問題 [33, 34] に関して，Luh et al. [35] は，マニピュレータの各関節にプロセッサを 1 つずつ割り付け，各関節に関する計算をそれぞれのプロセッサに行わせることによって，スケジューリング問題の簡略化を行っている．この方法では，各関節単位でスケジューリングを行っているため，結果がかなり最適解から離れてしまい，さらにプロセッサ数も可変に出来ないという欠点がある．また，笠原ら [36, 37] は，処理時間の異なる任意の数のタスクを，任意の数のプロセッサに割り付ける汎用的なスケジューリングアルゴリズムを開発し，これをマニピュレータの逆動力学問題に適用している．そこでは，リストスケジューリング [38] に基づく，ヒューリスティックアルゴリズム CP/MISF(Critical Path/Most

Immediate Successors First)，及びこれを分枝限定法と結び付けた DF/IHS(Depth First/Implicit Heuristic Search) の 2 種類の方法が提案されている．CP/MISF はスケジューリングに要する時間が短いが，近似解しか得ることができない．これに対して，DF/IHS はスケジューリングに要する時間は長くなるが，高い精度の解を得ることができるという特徴を持つ．さらに，Chen et al. [39] も，同じ特徴を持つ 2 種類の方法を提案し，マニピュレータの逆動力学問題に適用している．これらの方法の共通点として，分割されたタスクを複数のプロセッサに割り当てるに際して，組合せ計算の複雑さを軽減するために，データ通信に要する時間を無視しているという点があげられる．

マニピュレータの構造に一致するトポロジーとして，各関節に 1 チップずつプロセッサを割り当て，各関節に関する計算はそのプロセッサで行い，隣接関節間でのデータの授受のみを行うという方法もある．しかし，逆動力学方程式の手順としては，マニピュレータの台座側から，手先のリンクに向かって順に運動学計算を行った後，手先側から台座側に向かって順に動力学計算を行うため，各関節単位で処理を行った場合，各関節間の従属関係がそのまま現れてしまい，並列度が非常に低く，そのままでは並列処理の効果があまり現れない．Binder et al. [40]，橋本ら [41] は，並列度を抽出して処理の高速化を図るために，予測子を用いているが，時間遅れからくる誤差の影響が避けられない．さらに，プロセッサ数を関節数に一致させなければならないため，その自由度が少ないという欠点もある．

実際に処理時間の短縮を大きくするためには，上述のスケジューリングとともに，その前処理としてのタスク分割法を確立する必要がある．一般に，与えられた問題からタスク集合を生成する際，タスク分割を細かくする程タスク集合の並列度は高くなる．しかし，一方，タスクの細分化によってプロセッサ間の通信量が増加すると，並列処理の効率は低下する．従って，全体の並列処理時間を短くするためには，タスク集合の並列度とプロセッサ間の通信量の双方を考慮して，タスク集合を生成しなければならない．タスク分割については，一般的には演算レベル，方程式レベル等で行われ，通信に要する時間に対する比率により，おおまかにタスク粒度（タスクの大きさ）が選択されているが，それのみでは，各問題に固有の複雑度（並列度，先行関係等）が加味されていないため，十分とはいえない．

本論文では，スケジューリングの問題に対して，プロセッサ間の通信時間を考慮したスケジューリングアルゴリズムを提案し [42, 43]，解法の決定の問題に対して，ロボットマニピュレータの逆運動学問題を対象とした，並列処理向きの解法を示した後，タスク分割の問題に対して，逆動力学問題を対象としたプロセッサ間の通信時間を考慮したタスク分割法を示す [44]．

1.5 本論文の目的と構成

ロボットの知能化のためには，センサ，コンピュータ，アクチュエータの各要素技術の高性能化だけではなく，情報処理能力の向上が重要である．本論文では，ロボットにおける情報処理をテーマとして，センサからマニピュレータまでに関して，3つの主要な問題を取り上げる．

まず，センサに関して，単体で2種類の計測が可能な複合センサの一例として，距離センサへの拡張が可能なスリップセンサを開発する．

次に，センサの複合化，高性能化のためには，演算量が増加する傾向にあるということについて述べ，その演算を高速に行うためのマルチプロセッサ・スケジューリングアルゴリズムを示す．

さらに，このスケジューリングアルゴリズムをマニピュレータの逆運動学及び逆動力学に適用するとともに，マルチプロセッサ並列処理において必要となる，並列処理向き解法の決定及びタスク分割法の開発を行う．

本論文は以下の5章から構成される．

第1章では，ロボットにおける情報処理の重要性について述べ，把握用センサ及びマルチプロセッサ並列信号処理について従来の研究状況を概説した．

第2章では，ロボットハンドに装着可能で，非接触で把握対象物体のすべり速度を計測することが可能な，格子像投影型非接触スリップセンサを開発し，センサ出力波形の自己相関関数を用いた信号処理を行い，距離計測への適用も可能な二光束干渉型非接触スリップセンサへの拡張について述べた後に，スリップセンサの把握力制御への応用について述べる．

これらのロボット用センサを実用化するための条件として，実時間性があげられる．特に，広い測定範囲，高分解能を要求する場合，あるいは複数のセンサ信号を処理する場合に計算量が増加し，リアルタイム制御が困難になる傾向がある．

第3章では，第2章で述べたセンサ信号処理の高速演算法として，マルチプロセッサによる並列処理手法について述べる．処理対象を任意として，プロセッサ間の通信時間を考慮したスケジューリングアルゴリズムを提案した後に，実際の問題に対してシミュレーションを行い，処理時間を評価する．

第4章では，第3章で述べたマルチプロセッサスケジューリングアルゴリズムを，ロボットマニピュレータの逆運動学計算及び逆動力学計算に適用する．ロボットマニピュレータの逆運動学問題を対象として，通信のオーバヘッドを考慮した，並列処理向きの解法を示した後，逆動力学問題を対象として，通信量と通信のオーバヘッドを考慮したタスク分割について考察する．また，それぞれに対して，第3章で提案した

スケジューリングアルゴリズムを適用してシミュレーションを行い，処理時間を評価する．

最後に，第5章では，本論文の総括を行い，本研究で提案した方法の今後の課題と展望について述べる．

参考文献

[1] S.A.Shafer, A.Stentz and C.E.Thorpe, "An Architecture for Sensor Fusion in a Mobile Robot", *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.2002-2011, 1986.

[2] 石川正俊, "並列処理を用いた能動的センサシステム", 計測自動制御学会論文集, Vol.24, No.8, pp.251-255, 1988

[3] R.A.Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE J. Robotics and Automation*, Vol.RA-2, No.1, pp.14-23, March 1986.

[4] 吉田典晃, "サブサンプションアーキテクチャを用いたロボットの制御", 日本ロボット学会誌, Vol.11, No.8, pp.1118-1123, 1993

[5] 山崎弘郎, 石川正俊, "センサフュージョン", コロナ社, 1992

[6] 安江利一, 白井良明, "物体認識のための両眼立体視", 電子技術総合研究所報, Vol.37, No.12, pp.1101-1119, 1973

[7] 辻三郎, 白井良明, 諏訪基, "距離情報を用いた立体認識の試み", 電気四学会連合大会論文集, pp.3245, 1970

[8] 増田良介, 尾崎一義, 長谷川健介, 田口伸夫, "触覚を用いた工業ロボットの対象物体中心把握法", 電気学会全国大会予稿集, pp.1848-1849, 1973

[9] 花房秀郎, 浅田春比古, "空気式センサをもつロボットハンドの制御方式　－長方形物体の位置，姿勢探索－", 第 17 回自動制御連合講演会予稿集, pp.409-410, 1974

[10] 辻村健, 薮田哲郎, 森光武則, "マニピュレータ用超音波距離センサを用いた三次元物体の形状認識システム", 第 2 回日本ロボット学会学術講演会予稿集, pp.83-84, 1984

[11] 木下源一郎, 出澤正徳, "光学的距離センサによる物体形状計測", 第 4 回日本ロボット学会学術講演会予稿集, pp.153-156, 1986

[12] M.Ueda, K.Iwata and H.Singu, "Tactile Sensors for an Industrial Robot to Detect a Slip", *Proc. 2nd ISIR*, pp.63-76, 1972.

[13] 土屋朋隆, 森下議雄, 武藤英一, "すべりセンサを有する把握機構", 第4回日本ロボット学会学術講演会予稿集, pp319-320, 1986

[14] R.Tomovic and Z.Stojilikovic, "Multifunctional Terminal Device with Adaptive Grasping Force", *Automatica*, 11, pp.567-571, 1975.

[15] 増田良介, 長谷川健介, 尾嵜一義, "工業ロボットのすべり覚とその応用", 電気学会論文誌, 96-10, 1975

[16] 増田良介, 長谷川健介, 尾嵜一義, "工業ロボットのすべり感覚器", 第11回SICE学術講演会予稿集, 3204, 1972

[17] R.C.Luo and J.Y.Chang, "Robot Grasping Force Control Using Force/Slip Sensory Feedback System", *Proc. IECON*, Vol.1, 1985.

[18] 本田智, 由利昌明, "すべり覚センサによる把握力制御", 第4回日本ロボット学会学術講演会予稿集, pp.321-322, 1986

[19] Huang Xin-han, Li Jun-yuan and Chen Jin-jiang, "A robot slide sense control system", *Proc. ICAR*, pp.161-168, 1985.

[20] A.K.Bejczy, "Sensors, Controls, and Man-Machine Interface for Advanced Teleoperation", *SCIENCE*, Vol.208, No.4450, pp.1327-1335, 1980.

[21] 鈴木豊一, 増田良介, "分布型触覚センサによるロボットの力制御", 第4回日本ロボット学会学術講演会予稿集, pp.311-312, 1986

[22] A.E.Brennemann, R.L.Hollis, M.A.Lavin and B.L.Musits, "Sensors for Robotic Assembly", *Proc. IEEE Int. Conf. Robot Automation*, Vol.3, pp.1606-1610, 1988.

[23] J.T.Ator, "Image-velocity sensing with parallel-slit reticles", *J. Opt. Soc. Am.*, 53-12, pp.1416-1422, 1963.

[24] M.H.Butterfield, G.f.Bryant and J.Dowsing, "A new method of strip-speed measurement using random-waveform correlation", *Trans. Soc. Instr. Technol.*, 13, pp.111-123, 1961.

[25] 塩野幸策, "レーザ・ドップラ方式による非接触速度, 速度ムラ, 回転ムラ測定システム", 計測技術,'87.12, pp.72-77, 1987

[26] S.Shinohara, A.Mochizuki, H.Yoshida and M.Sumi, "Laser Doppler Velocimeter using the Self-Mixing Effect of a Semiconductor Laser Diode", *APPLIED OPTICS*, Vol.25, No.9, pp.1417-1419, 1986.

[27] H.W.Jentink, F.F.M.de Mul, H.E.Suichies, J.G.Aarnoudse and J.Grere, "Small Laser Doppler velocimeter based on the self-mixing effect in a diode laser", *APPLIED OPTICS*, Vol.27, No.2, 1988.

[28] 篠原茂信ら, "FM変調半導体レーザの自己混合による距離計測", 平成元年度電気関係学会東海支部連合大会講演論文集, pp.595, 1989

[29] 松田文夫, 森元裕志, 北川秀夫, 服部秀三, 上田実, "物体散乱光信号の自己相関処理を用いたすべり検出法", 日本ロボット学会誌, Vol.4, No.6, pp.602-606, 1986

[30] F.Matsuda, H.Morimoto, H.Kitagawa, S.Hattori and M.Ueda, "Control of an Industrial Robot with Nontactile Slip Sensor Using Auto Correlation Proccessed Laser Scattering Signal", *Proc. 17th ISIR*, pp.6·1-6·13, 1987

[31] 黒田敏秋, 北川秀夫, 松田文夫, 内川嘉樹, 服部秀三, "格子像投影型非接触スリップセンサの開発", 日本ロボット学会誌, Vol.9, No.4, pp.466-470, 1991

[32] 笠原博徳, 藤井博文, 岩田雅彦, 成田誠之助, "最適マルチプロセッサスケジューリングアルゴリズムを用いたロボットダイナミックスシミュレーションの並列処理", 電子情報通信学会論文誌D, Vol.J70-D, No.9, pp.1783-1790, 1987

[33] J.Y.S.Luh, M.W.Walker and R.P.C.Paul, "On-line Computational Scheme for Mechanical Manipulators", *ASME Trans. J. Dynamic Syst.,Measurement and Contr.*, Vol.102, pp.69-76, June 1980.

[34] 吉川恒夫, "ロボット制御基礎論", コロナ社, 1988

[35] J.Y.S.Luh and C.S.Lin, "Scheduling of Parallel Computer for a Computer-Controlled Mechanical Manipulator", *IEEE Trans. Syst.,Man and Cybern.*, Vol.SMC-12, No.2, pp.214-234, March 1982.

[36] 笠原博徳, 成田誠之助, "マルチプロセッサ・スケジューリング問題に対する実用的な最適及び近似アルゴリズム", 電子情報通信学会論文誌D, Vol.J67-D, No.7, pp.792-799, 1984

[37] 笠原博徳, 成田誠之助, "マルチプロセッサ・スケジューリング・アルゴリズムを用いたロボット制御計算の並列処理手法", 日本ロボット学会誌, Vol.2, No.5, pp.387-401, 1984

[38] 富田眞治, 末吉敏則, "並列処理マシン", オーム社, 1989

[39] C.L.Chen, C.S.G.Lee and E.S.H.Hou, "Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System", *IEEE Trans. Syst.,Man and Cybern.*, Vol.18, No.5, pp.729-743, Sep.1988.

[40] E.E.Binder and J.H.Herzog, "Distributed Computer Architecture and Fast Parallel Algorithm in Real-Time Robot Control", *IEEE Trans. Syst.,Man and Cybern.*, Vol.SMC-16, pp.543-549, 1986.

[41] 橋本亮一, "逆ダイナミクスの新しい並列計算法", 計測と制御, Vol.30, No.5, pp.406-411, 1991

[42] H.Kitagawa, M.Ito, F.Matsuda, Y.Uchikawa and S.Hattori, "Scheduling of parallel processing including communication time", *Proc. Autotech Asia '89*, pp.62-79, 1989

[43] 北川秀夫, 松田文夫, 内川嘉樹, 服部秀三, "通信時間を考慮に入れたマルチプロセッサスケジューリングアルゴリズム", 電子情報通信学会論文誌, Vol.J73-D-I, No.10, pp.812-817, 1990

[44] 北川秀夫, 包原孝英, 長南功男, 松田文夫, "タスクの再分割による多関節型マニピュレータの逆動力学並列計算の高速化", 日本ロボット学会誌, Vol.11, No.5, pp.710-716, 1993

第2章　把握制御用スリップセンサ及びその自己相関信号処理

2.1　はじめに

ロボットが物体を把握する際，物体に不必要な力を加えることなく，かつ，物体を落とすことなく確実に把握することが必要である．しかし，物体の重量，形状，材質が様々で，事前に予知できない場合，予め把握力を設定することは困難である．さらに，ロボットが物体を把握した状態で，加速動作する場合，それに応じて把握力を調整する必要がある．このために，スリップセンサをロボットハンドに装着して，物体がすべったことを検出し，この信号によって把握力を増加させて，すべりを止める把握力制御が考えられている．スリップセンサを把握力制御に用いるには，リアルタイムですべりを検出することができ，しかもロボットハンドに装着できるよう小型であることが必要である．

これまでに様々なスリップセンサが報告されているが，それらは接触計測型と非接触計測型とに大別することができる．接触計測型のスリップセンサとしては，振動検出型[1]，ボール回転型[2, 3]，回転ローラ型[4, 5, 6]，ディンプルボール型[7]といったものがある．これらは，センサを直接把握物体に押し当てなければならず，その押し当てる力が強すぎる場合には，物体に損傷を与えてしまい，逆に，押し当てる力が弱い場合には，正確にすべりを検出することができない．さらに，物体の形状によって，測定が不可能になるという欠点も持つ．また，分布型接触センサを用いてすべりを検出するセンサ[8]は，分解能の点で問題がある．

非接触計測型スリップセンサとしては，超音波を用いてすべりを検出するセンサ[9]があるが，動きは検出できるものの，方向，変位の計測は困難である．また，レーザ・ドップラ方式を利用したもの[10]もあるが，時間的，空間的に高いコヒーレンスのレーザ光が必要であるため，安価な半導体レーザでは，波長安定度が問題となる．さらに，半導体レーザの自己混合効果を利用して速度を計測するセンサ[11]もあるが，至近距離では戻り光がレーザの性能に影響を与えてしまうため，フィンガに取り付けて把握動作の制御を行う用途に使用することは困難である．

本章では，これらのスリップセンサの欠点を補った，格子投影型非接触小型スリップセンサ，及び自己相関関数を用いた信号処理法を示し，さらに，距離計測への適用も可能な，二光束干渉型非接触スリップ

センサについて述べる [12, 13, 14].

2.2 格子像投影型非接触スリップセンサ

2.2.1 測定原理

格子像投影型非接触スリップセンサの原理を Fig.2.2.1 に示す．レーザ光を回折格子に通し，$2f_L$（f_L：焦点距離）離れたところにおいたレンズにより集光すると，レンズからさらに $2f_L$ 離れた位置に回折格子と同じピッチの像が結ばれる．この回折格子像をハンドが把握する物体の表面上に結ぶと，物体面の微小な凹凸のために格子像の明るい部分の光は散乱される．物体が格子像の 1 ピッチ p_G だけ動くと，物体表面上の各点は再び同じ強度の光を散乱することになるので，物体付近に設置されたフォトディテクタでの受光信号は一定の周波数

$$f_S = \frac{1}{\tau} = \frac{v}{p_G} \tag{2.1}$$

をもつことになる．ここで，v は物体のすべり速度であり，受光信号の周期 τ より

$$v = \frac{p_G}{\tau} \tag{2.2}$$

によって求めることができる．

2.2.2 実験装置

実験装置の構成を，Fig.2.2.2 に示す．レーザ光は He-Ne レーザ（波長 $632.8nm$，出力 $5mW$），回折格子はピッチ $p_G = 25\mu m$ のものと $p_G = 125\mu m$ のもの，レンズは焦点距離 $f_L = 69.3mm$，直径 $D = 42mm$ のものを使用し，受光素子としてフォトトランジスタ（TOSHIBA TPS601），測定対象となる動く物体として，ギア付き DC サーボモータの回転軸面を使用した．

回折格子により回折されたレーザ光は，レンズを通して集光され，レンズより距離 $2f_L$ 離れたモータ回転軸面に格子縞像を結ぶ．モータ回転軸面を速度 v で動かすことにより，周期 $\tau = p_G/v$ の散乱光信号が得られるので，その信号をフォトトランジスタで受光し，フォトトランジスタの出力信号を，A/D 変換器（分解能 8bit，変換時間 $500ns$）を通してコンピュータ（NEC PC-9801VM）に入力して，その周期を求める．

2.2.3 実験結果及び考察

DC モータの軸面の回転速度 v と受光信号の周波数 f_S との関係を，Fig.2.2.3 に示す．図中，実線は理論式 (2．1) を表し，点は次節で示す自己相関信号処理によって得られる実験値を表す．この図からわかる

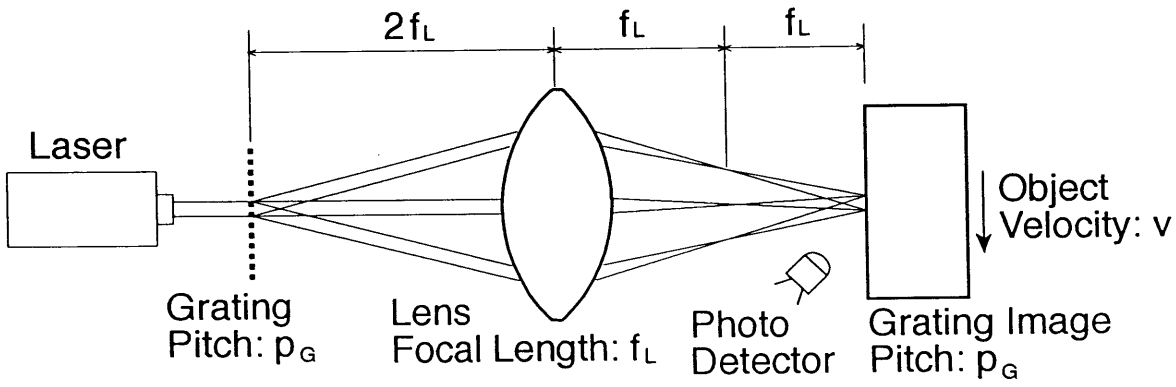


Fig.2.2.1. Principle of the slip detection.

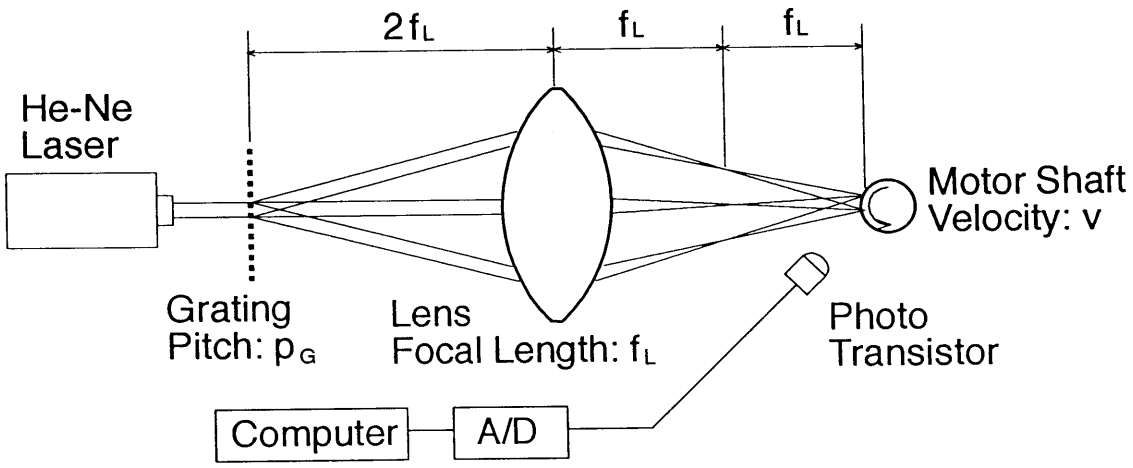


Fig.2.2.2. Experimental setup.

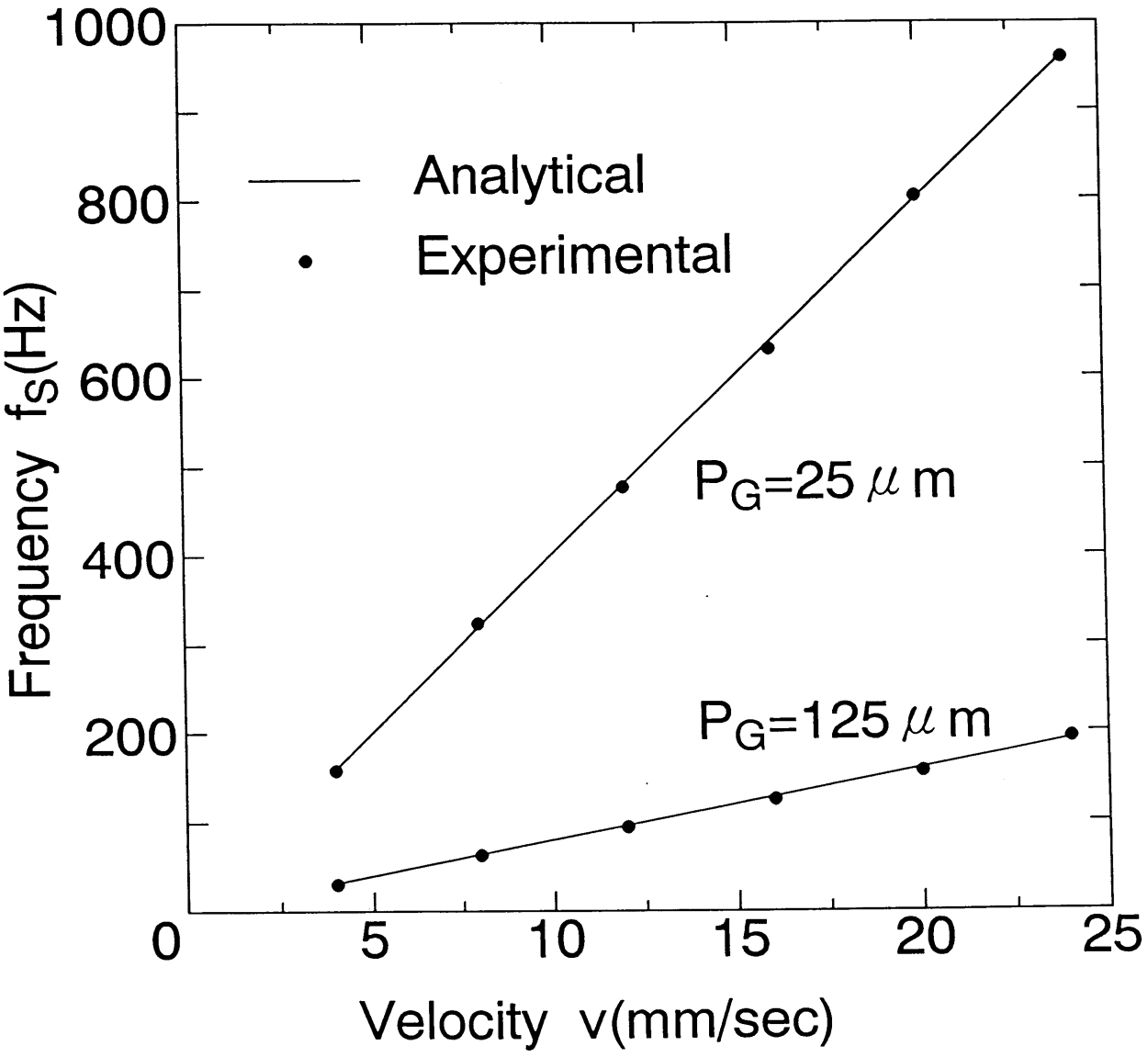


Fig.2.2.3. Frequency of the scattering light signal.

ように，v と f_S は比例し，その傾きはそれぞれ理論式 (2．1) に示す $1/p_G$ とほぼ一致している．

対象物体の材質を変化させて同等の実験を行ったところ，鏡面に近い表面特性を持つ物体以外では，同様の信号が観測された．鏡面の場合は，散乱光成分が少なかったために，信号が得られなかったと考えられる．

格子像投影型スリップセンサには以下のような利点がある．

1. 半導体レーザを用いることが可能であり，センサの小型化が容易である．
2. ビームスプリッタやミラーを用いる必要がないため，光学系の構成及び光軸調整が簡単になる．
3. 斜め方向のすべりにも対応可能である．
4. 縞の方向をずらした2つのセンサを用いることにより，すべり速度の大きさだけではなく方向の検出も可能である．
5. 単純な自己相関関数の計算を行うだけで十分であるので，計算部分のハードウェア化が容易である．

速度の測定誤差は，ディジタル信号処理の条件によっても変わる．その結果及び考察を次節に記す．

2.3 自己相関信号処理によるすべり速度の計算

本センサによって得られる受光信号は，周期成分を持ってはいるが，信号波形から直接その信号周期を求めるのは困難である．そこで，信号周期を求めるための信号処理方法として，自己相関を用いる．自己相関関数 $\Phi(i)$ は，サンプリング周期 Δt の離散的な関数として，次式のように与えられる．

$$\Phi(i) = \frac{1}{N-i}\sum_{n=i}^{N-1} F(n)F(n-i) \tag{2.3}$$

$$i = 0, 1, 2, \ldots, N-1$$

ここで，N は観測時間 T の間での総サンプル数（$N = T/\Delta t$）を表し，$F(n)$ は第 n サンプリング点でのサンプル信号強度を表す．i は自己相関をとる場合の時間の遅れを，サンプリング数で表したものであり，$N-1$ はその最大値である．

式 (2．3) で求めた $\Phi(0), \Phi(1), \Phi(2), ..., \Phi(N-1)$ の中で，極大値をとるものを選び出す．極大値が L 個得られたとすると，それぞれの極大値 $\Phi(P_0), \Phi(P_1), ..., \Phi(P_{L-1})$ に対応する，$P_0, P_1, ..., P_{L-1}$ が得られ

る．ここで，

$$\Phi(P_k) = k\_th\ peak\ value\ of\ \Phi(i) \tag{2.4}$$

である．このとき，隣合う極大値 $\Phi(P_k), \Phi(P_{k-1})$ のサンプリング数 $P_k, P_{k-1}(k = 1, 2, ..., L)$ の差の平均にサンプリング周期 Δt をかけたものが信号の周期 τ となる．すなわち，

$$\tau = \frac{1}{L}\sum_{k=1}^{L}(P_k - P_{k-1})\Delta t = \frac{1}{L}(P_L - P_0)\Delta t \tag{2.5}$$

さらに，$P_0 = 0$ より τ は

$$\tau = \frac{P_L \Delta t}{L} \tag{2.6}$$

で表される．そして，物体のすべり速度 v は

$$v = \frac{p_G}{\tau} \tag{2.7}$$

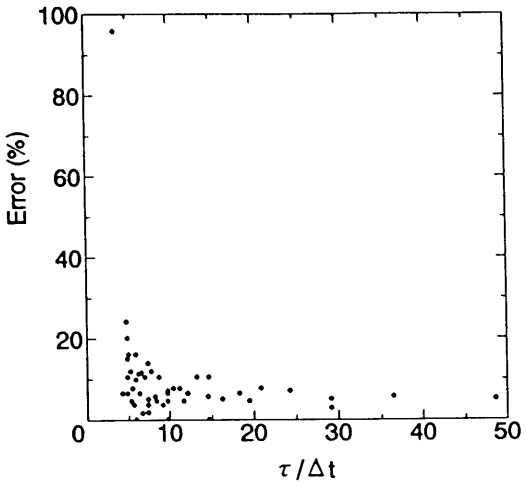
で求められる．

信号の周期を求めるための信号処理に自己相関関数を用いた理由としては，以下の点があげられる．

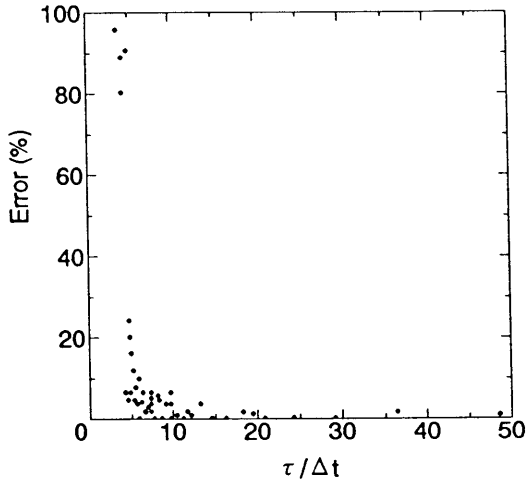
1. 本センサを通常の速度計ではなくスリップセンサとして使用するためには，速度測定範囲（ダイナミックレンジ）を大きくし，かつ信号の観測時間を短くする必要があるため，低い周波数の信号に対しては数周期の観測信号から速度を測定しなければならない．

2. 分解能を上げるためには，フーリエ変換では信号の観測時間を長くとる必要があるが，自己相関処理ではサンプリング周波数を上げるだけで，観測時間を長くとる必要がない．観測時間を長くとりすぎるとその間に物体がすべり落ちてしまうため，スリップセンサには不向きであるが，サンプリング周波数を上げるためには A/D 変換器を高速化すればよい．

自己相関関数の精度は，信号のサンプリング周期 Δt の値によって変化する．丁度，信号のピークの部分にサンプリング点があれば，正確に信号の周期を求めることができるが，サンプリング点の間にピークがくると，理論的に最大 $\Delta t/2$ の誤差が出る．また，信号の観測時間 T が長いほど，フィルタリング効果が効いて測定精度が上がる．そこで，周波数 $880Hz$ の信号に対して Δt 及び T の値を変えて自己相関関数の計算を行い，計算された信号周期の理論値 $(1/880sec)$ からの誤差をグラフにしたものを Fig.2.3.1 に示す．このグラフより，観測する周期数 T/τ を 3 以上，1 周期あたりのサンプル数 $\tau/\Delta t$ を 10 以上（即ち，観測時間を信号の 3 周期分以上，サンプリング周期を信号の周期の 1/10 以下）にとれば，信号周期



(a) $T/\tau = 3$



(b) $T/\tau = 5$

Fig.2.3.1. Quantizing error.

の誤差はほぼ 10% 以内に収まることがわかる．本センサを速度センサではなくスリップセンサとして使用する場合，速度計測の精度を上げる以上に，処理速度を上げることが重要となる．さらに，すべり速度によって把握力を増加させるスリップセンサの場合，把握対象物の表面特性は不特定であるため，速度計測の精度を必要以上に上げても，対象物の差異によるばらつきに吸収されてしまう．従って，速度計測誤差の目安を 10%とした．

必要となるサンプル数 N は，以下の式によって計算される．

$$N = n_r \frac{T}{\tau} \frac{\tau}{\Delta t} \tag{2.8}$$

ここで，n_r とは測定する最低速度と最高速度の比である．一例として，上述のように観測周期数 $T/\tau = 3$，1 周期あたりのサンプル数 $\tau/\Delta t = 10$ として，$n_r = 10$ とすれば，必要となるサンプル数 $N = 300$ が算出される．

自己相関関数の計算に必要とされる乗算，加算数は共に $N^2/2$ となるので，積和演算 1 ステップで $100nsec$ 程度の積和器を用いた場合，$N = 300$ として $10 \sim 20msec$ で計算が可能であり，スリップセンサへの応用が十分に可能である．なお，速度計として使用する場合は，高速応答性よりも精度が優先する場合があるが，その場合はサンプル数を多くとれば良い．

上述のように，自己相関関数 $\Phi(i)$ は Δt で離散的に求められるため，より精度をあげるためには，$\tau/\Delta t$ を大きくする方法の他に，各サンプル点間を補間する方法がある．

自己相関関数 $\Phi(i)$ から，信号の周期を求める 3 種類の方法を Fig.2.3.2 に示す．$\Phi(i)$ の中で極大値をとるものを選び，これを $\Phi(j)$ とすると，$\Phi(j)$ 及びその前後の $\Phi(j-1), \Phi(j+1)$ の 3 点（図中，Δt の時間間隔でサンプリングされている 3 点）から以下のようにピークとなる値を求める．なお，図中の黒丸は以下の方法から計算されたピーク値を表す．

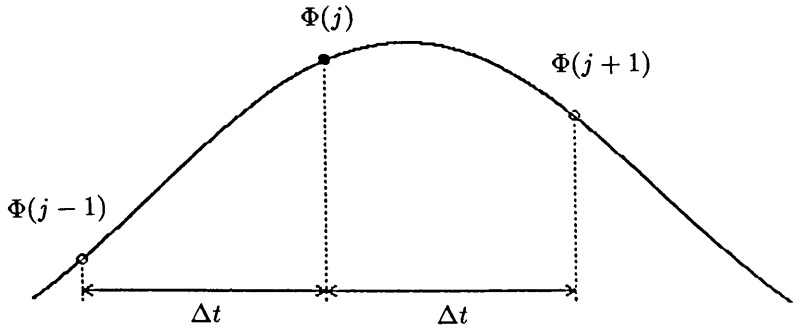
(a) ピーク検出法

単純に $\Phi(j)$ をピークとして，そのサンプリング数 j にサンプリング周期 Δt をかけた値

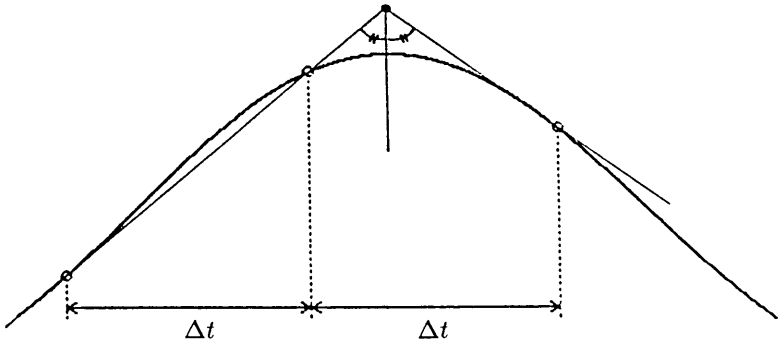
$$\tau = j\Delta t \tag{2.9}$$

を信号の周期とする．

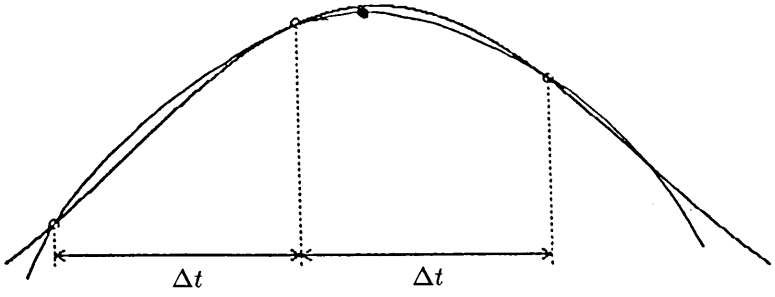
(b) 直線補間法



(a) Peak detection



(b) Linear interpolation



(c) Least squares

Fig.2.3.2. Methods of interpolation.

Fig.2.3.2(b) のように 3 点を用いて，直線で内挿することによってピークを求める．ピークの時間を，相当するサンプリング値 p に換算すると，

$$p = \begin{cases} j + \dfrac{\Phi(j+1) - \Phi(j-1)}{2\{\Phi(j) - \Phi(j-1)\}}, & for \quad \Phi(j-1) \leq \Phi(j+1) \\ j + \dfrac{\Phi(j+1) - \Phi(j-1)}{2\{\Phi(j) - \Phi(j+1)\}}, & for \quad \Phi(j-1) > \Phi(j+1) \end{cases} \tag{2.10}$$

となるので，方法 (a) における j のかわりに，この p の値を用いて

$$\tau = p\Delta t \tag{2.11}$$

を信号の周期とする．

(c) 最小二乗近似法

3 点を二次曲線で最小二乗近似し，その極大値をピークとする．二次曲線を

$$\Phi(t) = at^2 + bt + c \tag{2.12}$$

として，最小二乗法により係数 a,b,c を求め，その値より

$$\tau = -\frac{b}{2a}\Delta t \tag{2.13}$$

を信号の周期とする．

Fig.2.3.3 に上記の 3 方法による誤差の比較を示す．横軸は信号の周期 τ に対するサンプリングピッチ Δt の割合，即ち 1 周期中のサンプル数の逆数を示し，縦軸はその時の誤差を示す．また，グラフの一点鎖線は上記の方法 (a)，実線は方法 (b)，点線は方法 (c) でピークを求めた場合の誤差である．図中，方法 (a) で所々誤差が 0 に近くなっている点があるが，これは τ が Δt の整数倍になって，丁度真のピークの位置にサンプル点が重なったためである．

A/D 変換速度が遅い場合，あるいは積和演算の増加により処理時間が著しく増加する場合は 1 周期あたりのサンプル数 $\tau/\Delta t$ を大きくすることが困難であるので，単純に方法 (a) でピークを求めるよりも，方法 (b) または方法 (c) を用いて速度計測精度を上げることが有効である．

2.4 ロボットハンドに装着可能な小型スリップセンサ

2.4.1 小型スリップセンサの製作

2.2節で示したように，本スリップセンサは光源，回折格子，レンズ，受光素子からなる．本センサをロボットハンドに装着可能となるように小型化するためには，投光器として半導体レーザを用いればよい．さ

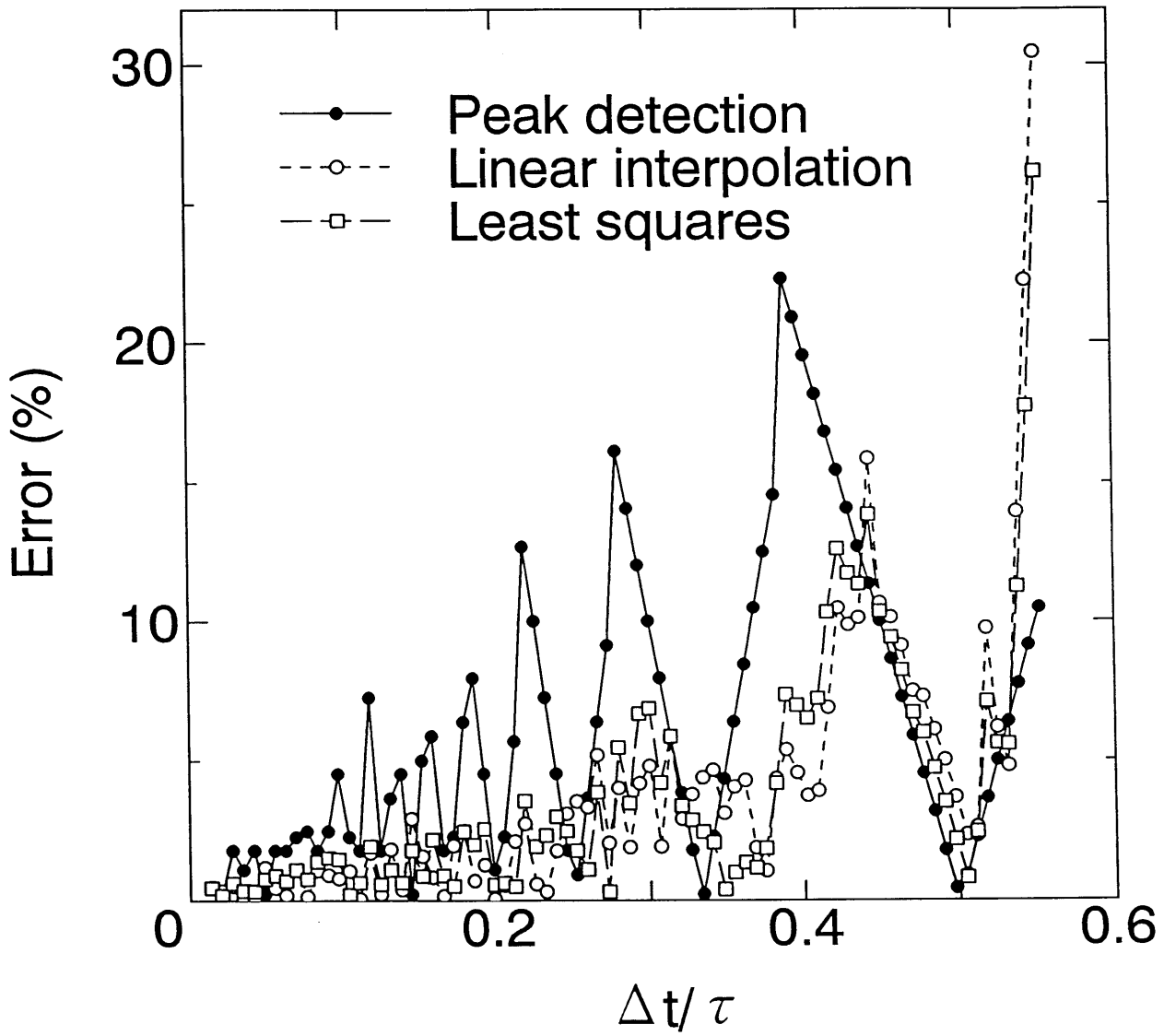


Fig.2.3.3. Error in the peak detection and interpolation processing.

らに，投光器と受光器を一体にして製作することによっても，センサの一層の小型化を行うことが可能である．

Fig.2.4.1 に小型化を試みたスリップセンサの構成， Fig.2.4.2 にセンサの写真， Fig.2.4.3 にその部品写真を示す．直径 $25mm$，長さ約 $100mm$ のアルミニウムの円筒の中に投光器と受光器を組み入れて製作したものである．以下に主な構成部品を示す．

- 光源 ..... 半導体レーザ（SHARP LT015MD 波長 $830nm$, 出力 $40mW$）
- 回折格子 ..... ピッチ $25\mu m$
- 集光レンズ ..... 焦点距離 $15.0mm$, 直径 $14.5mm$
- 受光素子 ..... フォトトランジスタ（TOSHIBA TPS601）

半導体レーザ光は，一定の広がり角をもって光を放射するので，レンズによりコリメートして平行ビームとした後に回折格子に通した．

今回試作したスリップセンサは，ロボットハンドのフィンガの大きさ，製作，調整の容易さを考慮して，上記の大きさにしたが，さらに小型化する必要がある場合は，フォトトランジスタのサイズ，レンズの焦点距離などを小さくすることで，小型化を実現することができる．

2.4.2 実験結果及び考察

試作したスリップセンサを用いて，ロボットハンドが把握する物体がすべった場合の，すべり速度を検出する実験を行った．本実験の目標は,「置いてある物体を確実に，最小の力で把握して持ち上げる」という動作の達成である．ハンドが把握する対象物体として，アルミニウムの角材 ($20mm \times 20mm \times 120mm$) を使用した．すべり速度の設定を容易にするために， Fig.2.4.4 のように，アルミ片は動かさずにベースに固定しておき，スリップセンサを装着したロボットハンドを上下運動させた時のすべり速度を検出した．フォトトランジスタでの受光信号を A/D 変換器（分解能 8bit，変換時間 $500ns$）を通して，コンピュータ（NEC PC-9801VM）に入力し，自己相関関数を計算し，すべり速度を求めた．

Fig.2.4.5 は，物体のすべり速度 $v = 15mm/s$ の時の散乱光信号である． Fig.2.4.6 は， Fig.2.4.5 に示した散乱光信号を $60\mu sec$ でサンプリングして，自己相関関数を計算したものである．この波形のピーク $\Phi(k)$ を求めることで，周期 τ が得られ，式 (2 . 2) を用いることによりすべり速度 v が求められる．

ハンドの速度設定値を $1mm/sec \sim 40mm/sec$ とした時の実験結果を Fig.2.4.7 に示す．実線はハンドの速度設定値を表し，点は受光信号から計算した速度の値を表す．図から，上記の速度範囲では有効にす

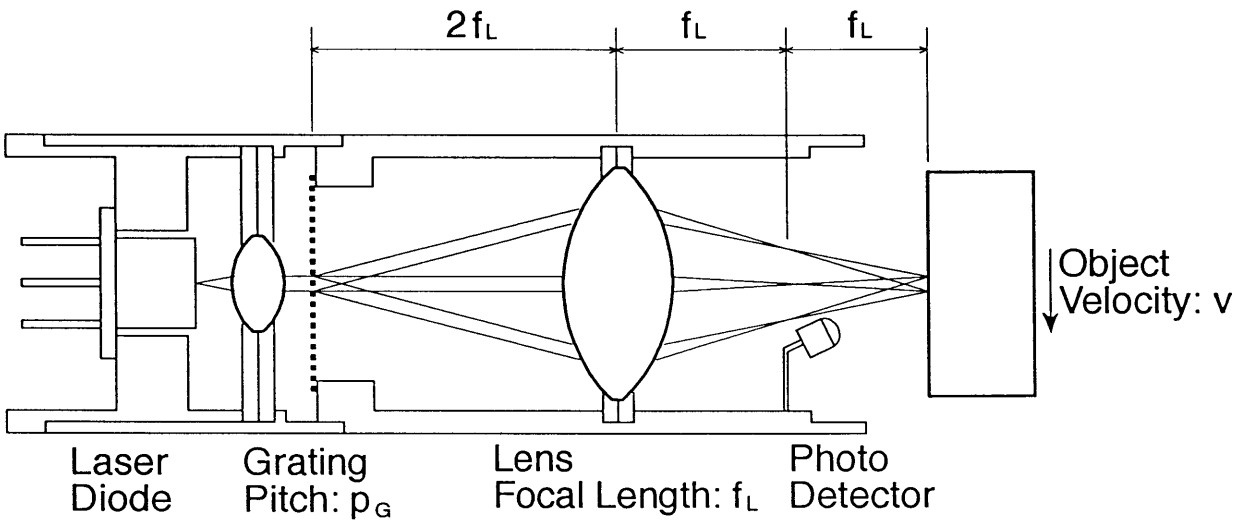


Fig.2.4.1. Construction of the slip sensor.

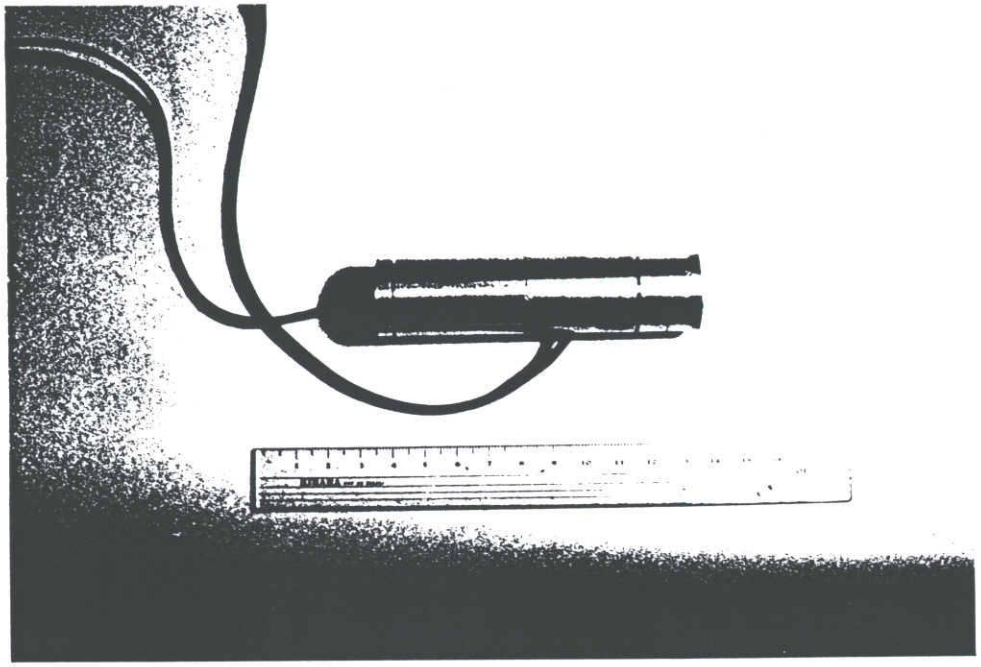


Fig.2.4.2. Compact slip sensor.

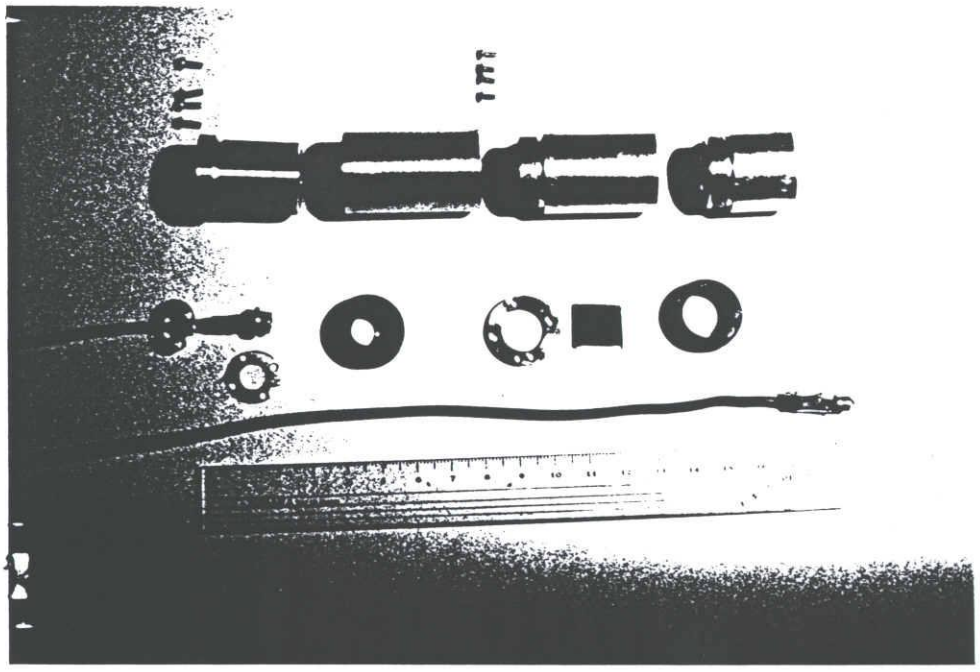


Fig.2.4.3. Parts of the slip sensor.

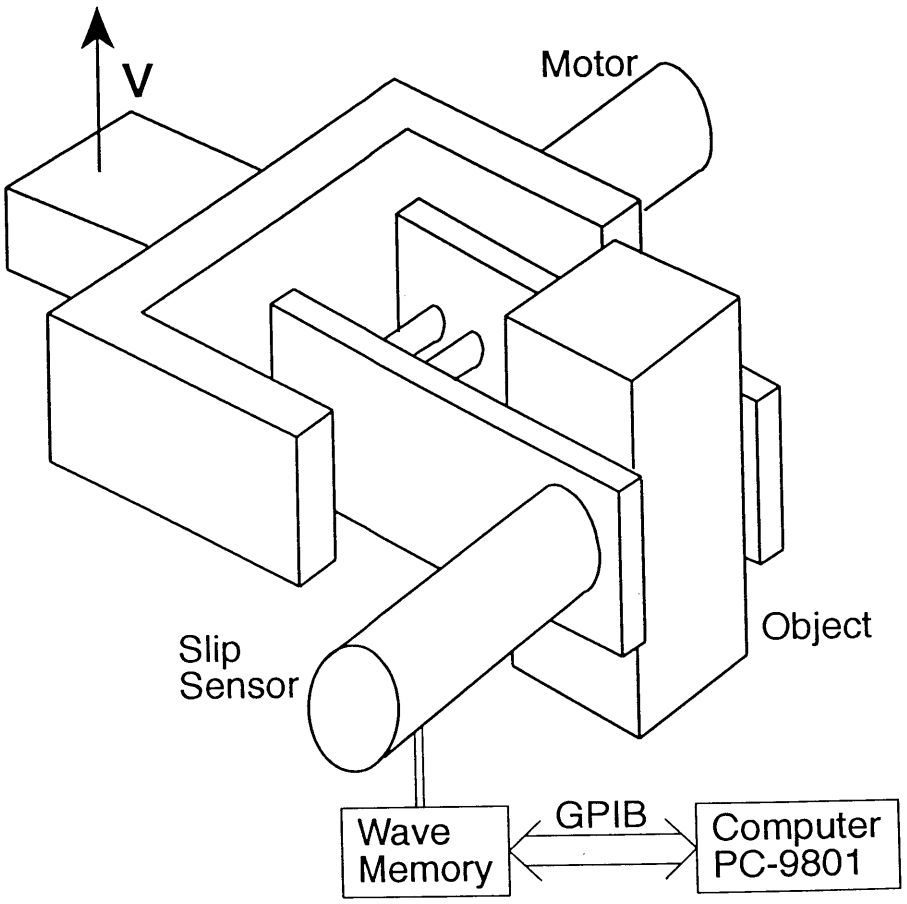


Fig.2.4.4. Schematic diagram of the experimental setup.

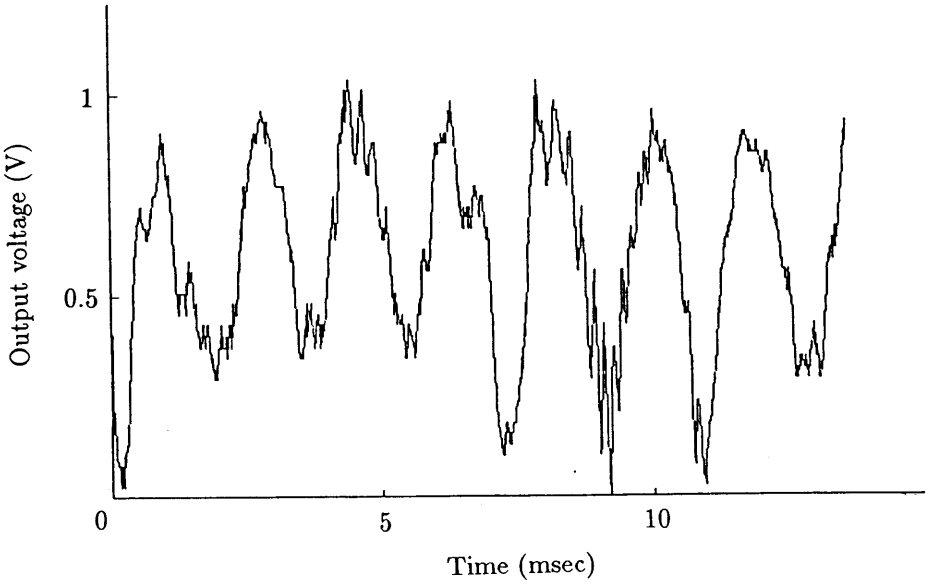


Fig.2.4.5. An example of a scattering light signal.

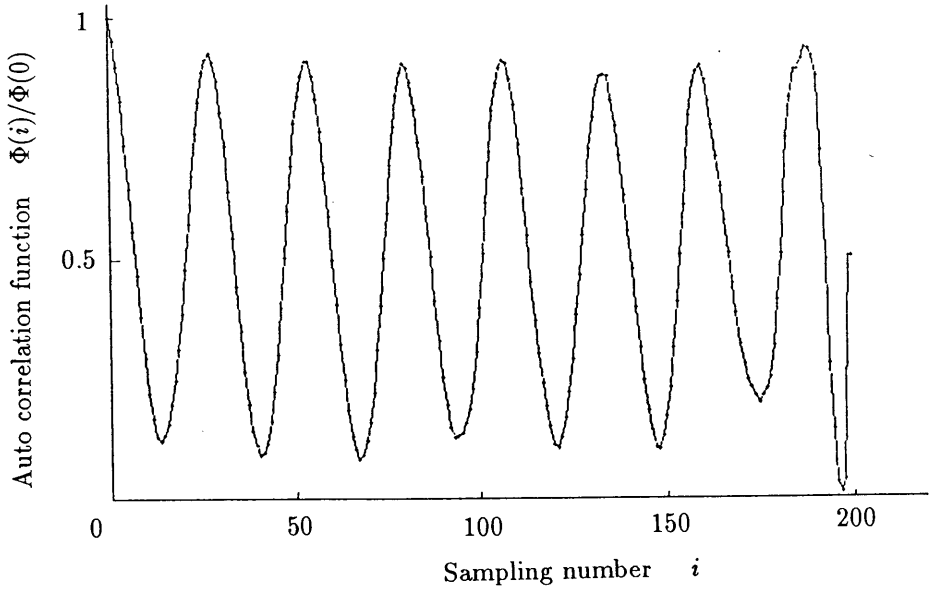


Fig.2.4.6. Auto correlation function.

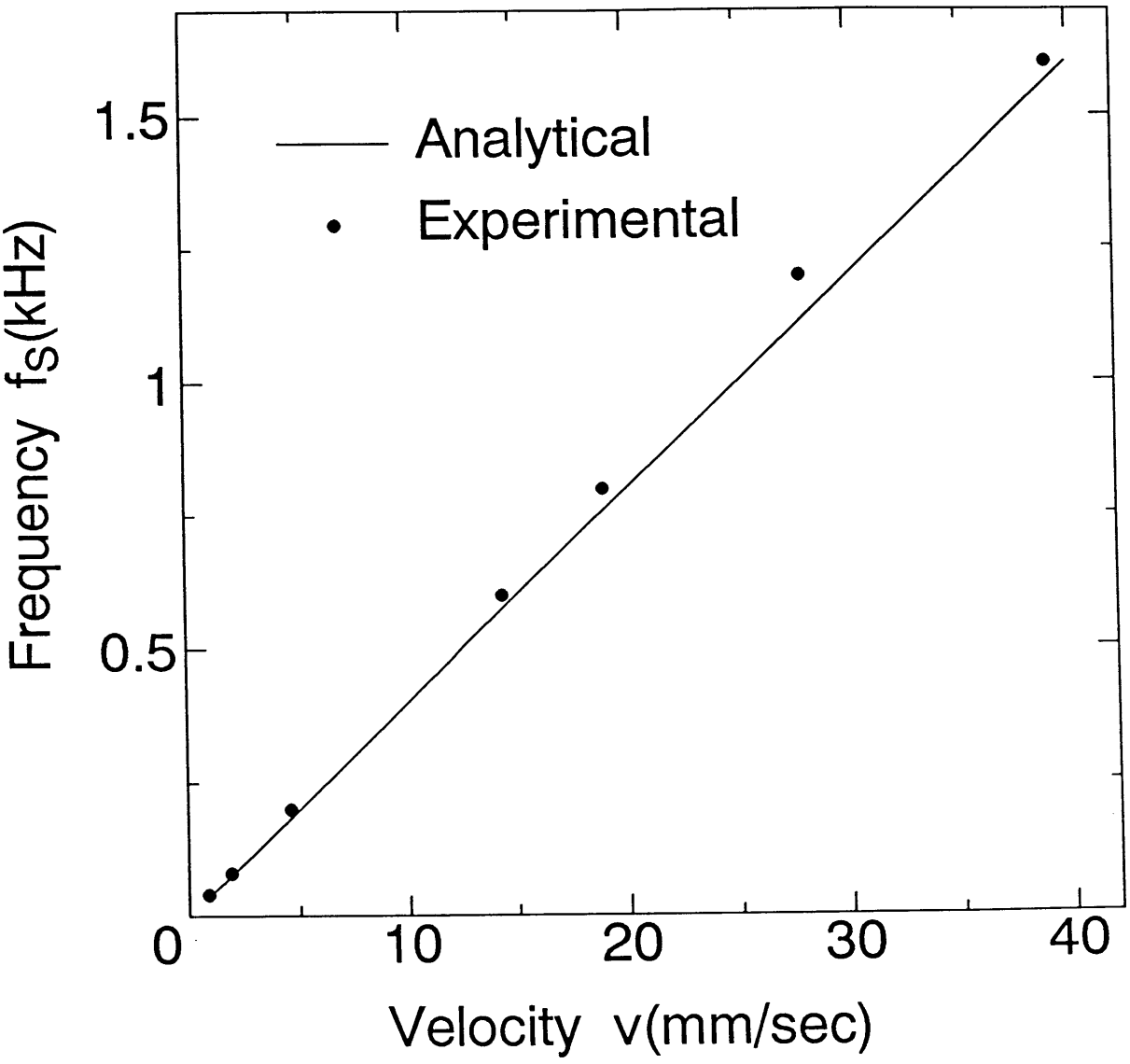


Fig.2.4.7. Frequency of the scattering light signal.

べり速度が検出できることがわかる．検出速度のばらつきは，主に，物体を設置した状態が理想的でないために，物体表面に結ぶ縞のピッチが回折格子のピッチと異なっていることにより生じる誤差，散乱光をサンプリングし，自己相関関数を計算して周期を求める際に生じる誤差，また，ハンドの速度の誤差の影響によるものと考えられる．

本実験では，回折格子としてピッチが $25\mu m$ のものを使用しているが，より小さな速度を高精度で検出するためには，ピッチのより細かい回折格子を用いて，受光信号の周期が大きくなりすぎないようにすればよい．

現在のシステムでは，スリップセンサの信号を A/D 変換器でメモリに取入れた後，コンピュータ PC-9801 で自己相関関数を計算しているが，受光信号を DSP 等の高速演算プロセッサを用いて計算すると，高速にすべり速度を検出することが可能となる．例えば，サンプリング数として 300 個程度を取って計算すると $10 \sim 20msec$ ですべり速度が求められる．

Table 2.4.1 に試作スリップセンサの仕様及び特性をまとめて示す．

2.5 二光束干渉型非接触スリップセンサ

前節までで示してきた格子像投影型スリップセンサは，物体表面上に回折格子の像を投影する方法であるため，対象物までの距離が変わると像がぼけてしまうという問題点がある．これは対象物上のレーザ投影面が平面でない場合には大きな問題となる．本節では，格子像投影型非接触スリップセンサを，距離変化に対して対応が可能なように改良した，二光束干渉型非接触スリップセンサについて示す．

2.5.1 測定原理

Fig.2.5.1 に示すように，二点から同波長，同位相の光を放射すると，二つの光の重なり合う部分に図のような干渉縞が生じる．図に示すように，二つの点光源の中心に原点をとり，干渉縞の中心軸（光軸方向）に x 軸，点光源を結ぶ直線方向に y 軸，それらに垂直に z 軸をとる．また，点光源の間隔を r，光の波長を λ とする．この時，２つの点光源からの光が強め合う条件は，

$$y^2 = \frac{m^2}{r^2 - m^2}x^2 + \frac{m^2}{r^2 - m^2}z^2 + \frac{m^2}{4} \tag{2.14}$$

$$m = n\lambda \ (n = 0, \pm 1, \pm 2, \ldots)$$

という式で表される．

この縞のピッチを表す式を次のようにして求める．

Table 2.4.1 Specification and caracterictics of slip sensor.

Type	Nontactile sensor
Principle	Velocity $\rightarrow$ Period of scattering signal
Components	Laser diode, Optical grating, Lens, Photo transistor
Signal	Analog output
Process	Amplification, A/D conversion, Auto correlation
Range	$1 \sim 40mm/sec$
Accuracy	$\pm 5\%$
Size	$\phi 25mm \times 100mm$

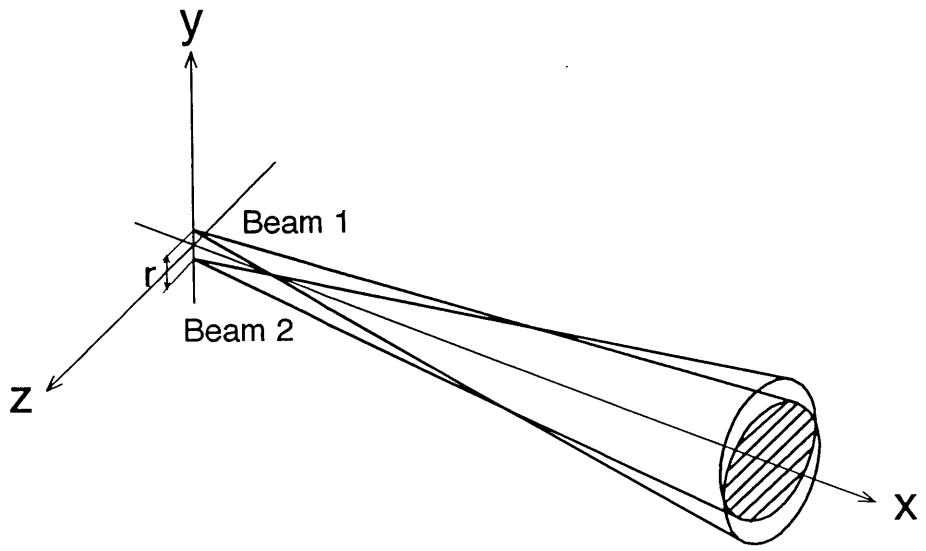


Fig.2.5.1. Interference fringes.

Fig.2.5.2 に，yz 断面の様子を示す．点光源の位置から x 方向に距離 L をとる．距離 L が大きくなるにつれて，縞のピッチ p は大きくなる．縞の中心付近では，ピッチ p と距離 L の関係は，

$$p \propto L \tag{2.15}$$

と近似することができる．

以上の原理によって生成した縞を利用して，2.2 節と同様のスリップセンサを構成する．Fig.2.5.3 に示すように，レーザ光を放射する点光源を物体との接触面から距離 L だけ離してフィンガ内に設置すると，ロボットハンドが物体を把握する時には，点光源と物体の距離が一定となる．二光束の干渉縞のピッチを物体までの距離 L の関数 $p(L)$ で表し，物体のすべり速度を v，散乱光の周期を τ とすると，これらの関係は，2.2 節の方式と同じように

$$v = \frac{p(L)}{\tau} \tag{2.16}$$

と表される．

Fig.2.5.1 に示したような 2 つの点光源を得るために，Fig.2.5.4 のような光学系を用いる．光学系の設定は基本的に前述の方式と同じであるが，各次数の回折光のうち ± 1 次光のみを用いて干渉縞を物体面上に投影する．物体の把握位置を前述の方式と同じ所に設定すると，縞のピッチは

$$p(f_L) = \frac{p_G}{2} \tag{2.17}$$

となり，式 (2 . 16) からすべり速度 v は

$$v = \frac{p_G}{2\tau} \tag{2.18}$$

となる．

2.5.2 実験結果及び考察

2.4節の小型スリップセンサを用いてこの方式を実現するために，それぞれの次数の回折光が一点に集光している後焦点面に，Fig.2.5.5 のような銅板（フィルタ）を取り付けて ± 1 次光のみを通過させる．ハンドが把握する対象物体として，アルミニウムの角材 ($20mm \times 20mm \times 120mm$) を使用した．実験の諸条件は 2.4節の実験と同じである．

フィルタを装着した状態の小型スリップセンサを Fig.2.5.6 に，ハンドの速度設定値を $2mm/sec \sim 130mm/sec$ として実験を行った結果を Fig.2.5.7 に示す．図中，横軸はハンドの速度設定値を表し，縦軸は受光信号の周波数を表す．実線はその理論値であり，点は受光信号から得られた値である．図から，上記

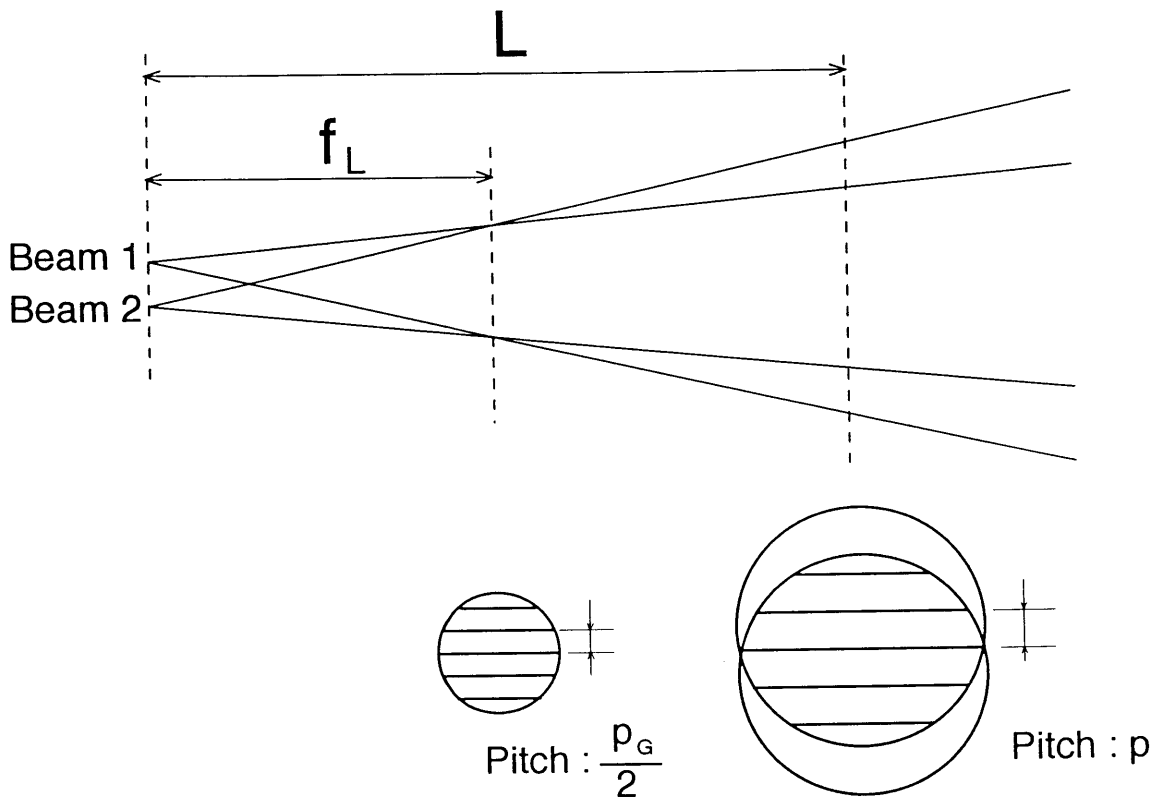


Fig.2.5.2. Pitch of the interference fringes.

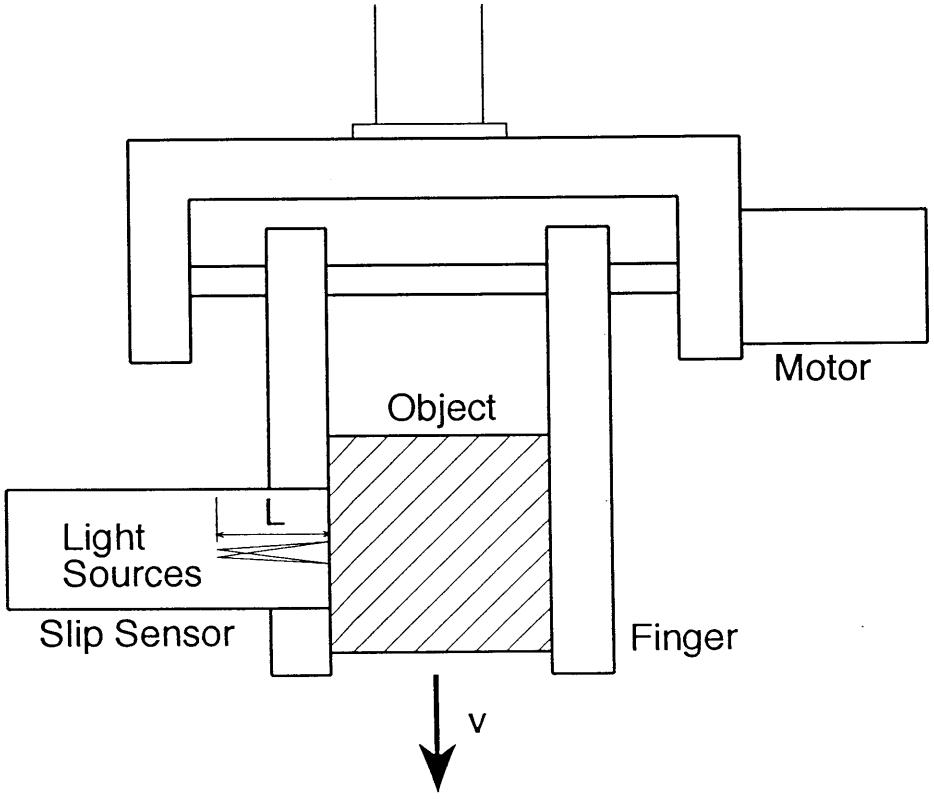


Fig.2.5.3. Slip sensor attached to the robot hand.

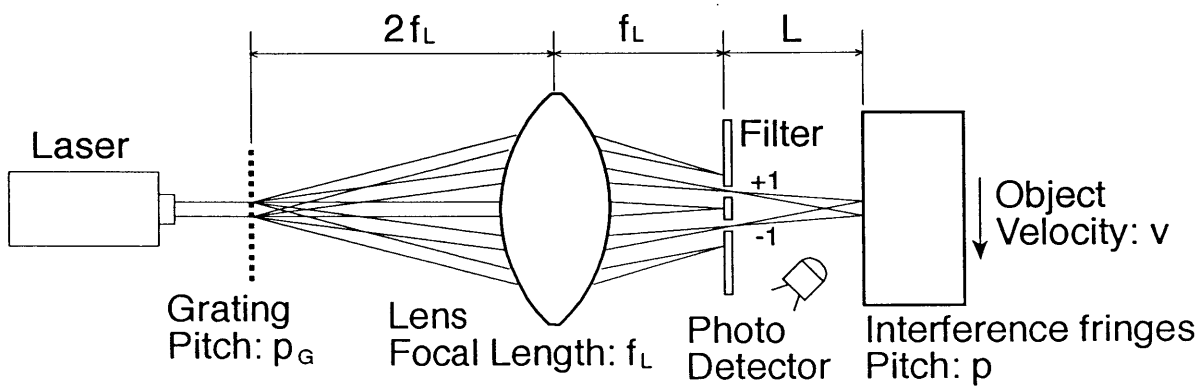


Fig.2.5.4. Principle of the slip sensor.

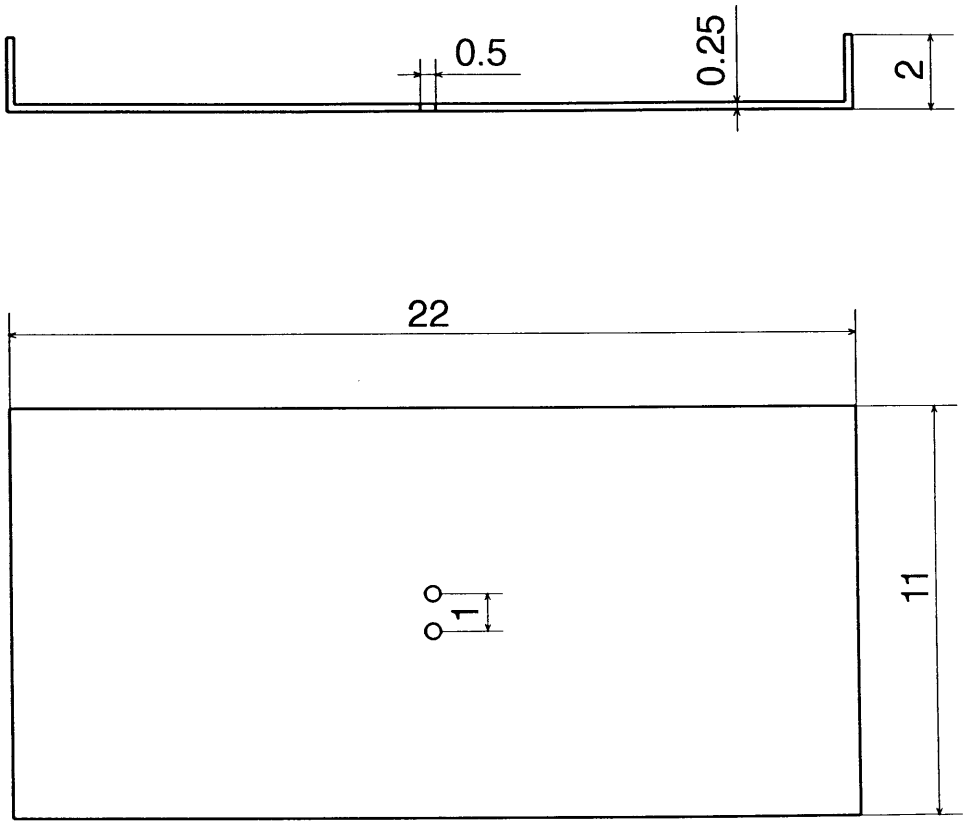


Fig.2.5.5. ± 1 order diffraction beam filter.

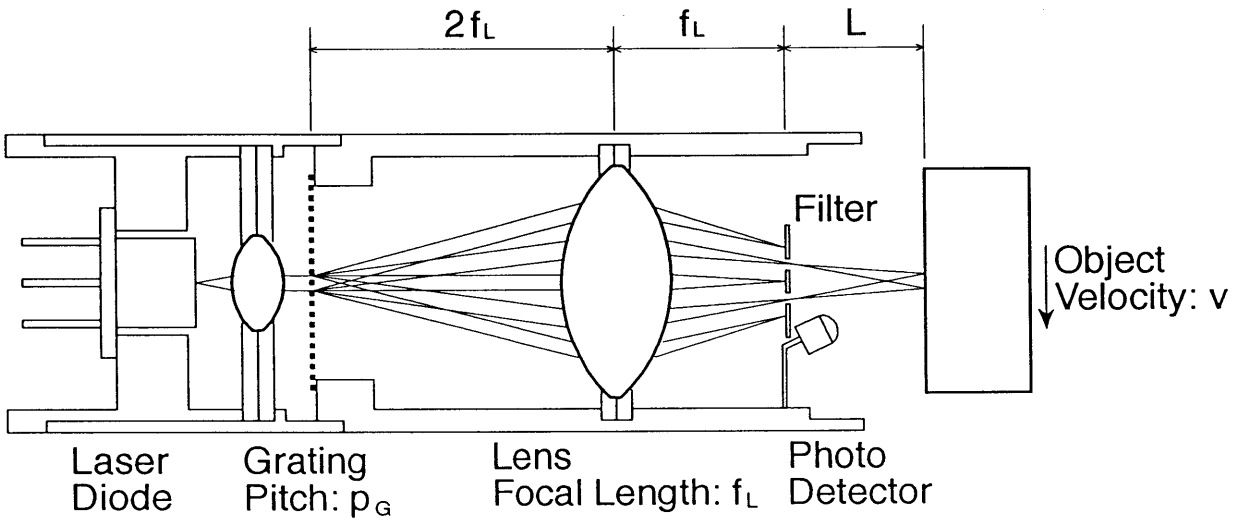


Fig.2.5.6. Construction of the slip sensor.

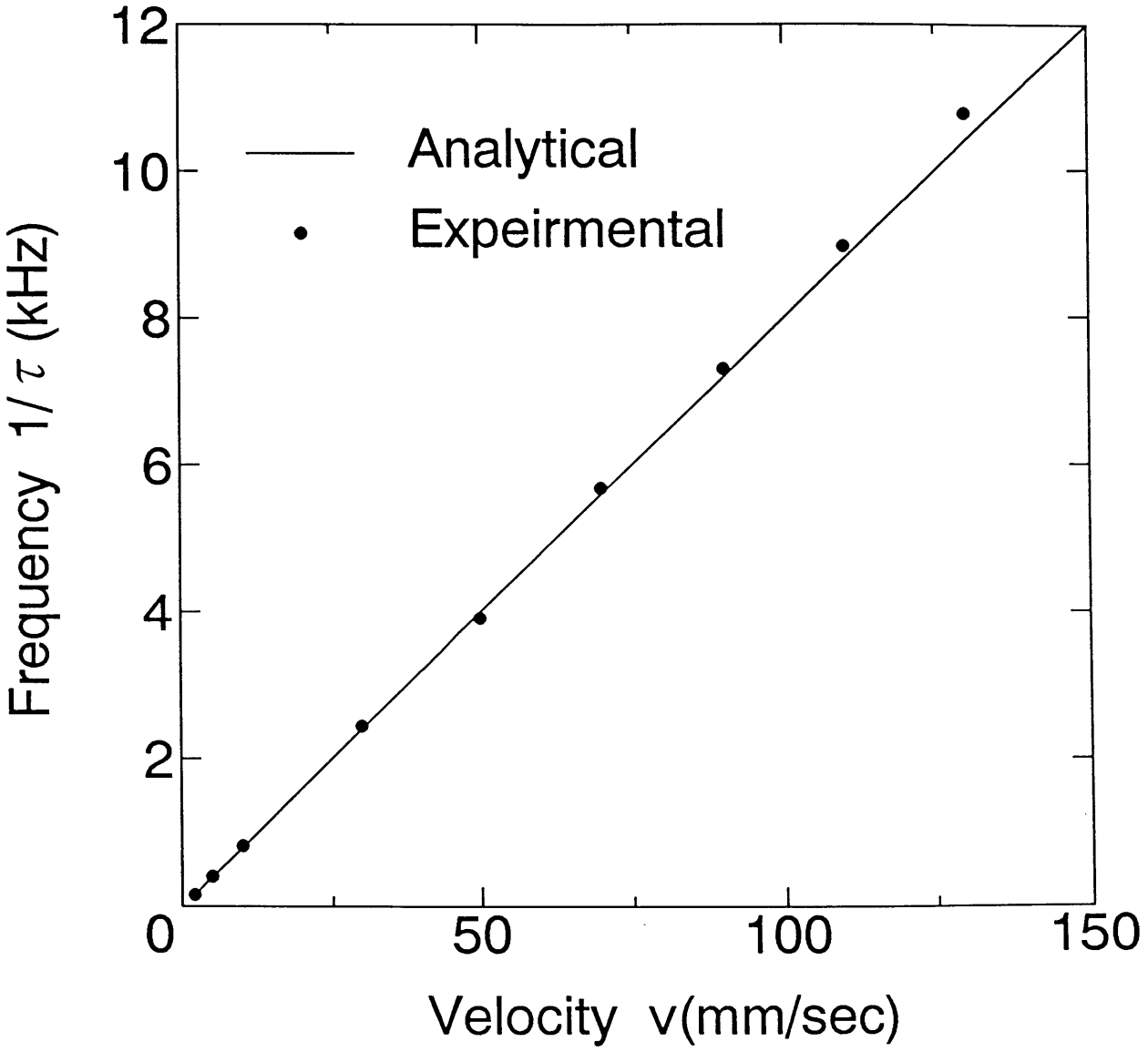


Fig.2.5.7. Frequency of the scattering light signal.

の速度範囲では有効にすべり速度が検出できることがわかる．検出速度のばらつきは，主に，①物体を設置した状態が理想的でないために，物体表面に結ぶ縞のピッチが回折格子のピッチと異なっていることにより生じる誤差，②散乱光をサンプリングし，自己相関関数を計算して周期を求める際に生じる誤差，③ハンドの速度の設定値からの誤差，および，④式 (2．15) の近似式の理論値からの誤差によるものと考えられる．

フィルタ（後焦点面）から物体面までの距離は $15mm \sim 120mm$ の範囲で速度検出が可能であるので，物体形状の変化に十分対応できることがわかる．前述の格子像投影型スリップセンサに対する本方式の特徴として，以下の点が上げられる．

1. 焦点位置からの光軸方向の距離のずれにも対応できるため，物体形状の変化に対応できる．

2. ハンドを一定速度で物体面に平行かつ縞に垂直な方向に移動させることによって，受光信号の周期から距離計測が可能になる．

2.6 把握力制御への適用

ロボットが未知の物体を把握する場合，その物体を破壊することなく，かつすべり落とすことのない最小の把握力（最適把握力）で把握する必要がある．スリップセンサ及び自己相関信号処理によって求めた物体のすべり速度を利用した把握力制御は以下の手順で行われる．

i. 物体を破壊しない微少な力（初期把握力）で把握する．

ii. 把握力が不足している場合，物体がすべるので，そのすべり速度に比例して増加させた把握力信号を D/A 変換器に送り，モータを駆動する．

iii. すべり速度が 0 になるまで前項を繰り返す．

Fig.2.6.1 にその概要図を， Fig.2.6.2 にその制御フローチャートを示す．

今後，この手順で把握力制御を行う．なお，この制御は置いてある物体を把握して持ち上げる場合の他に，物体を把握した状態でロボットが加速動作する場合にも必要である．

2.7 まとめ

把握制御を効果的に行うためには，非接触に微少すべりを計測するセンサが必要とされる．格子像を物体表面上に投影したときの散乱光信号の変化から把握物体のすべりを非接触に検出するセンサの開発を行った．

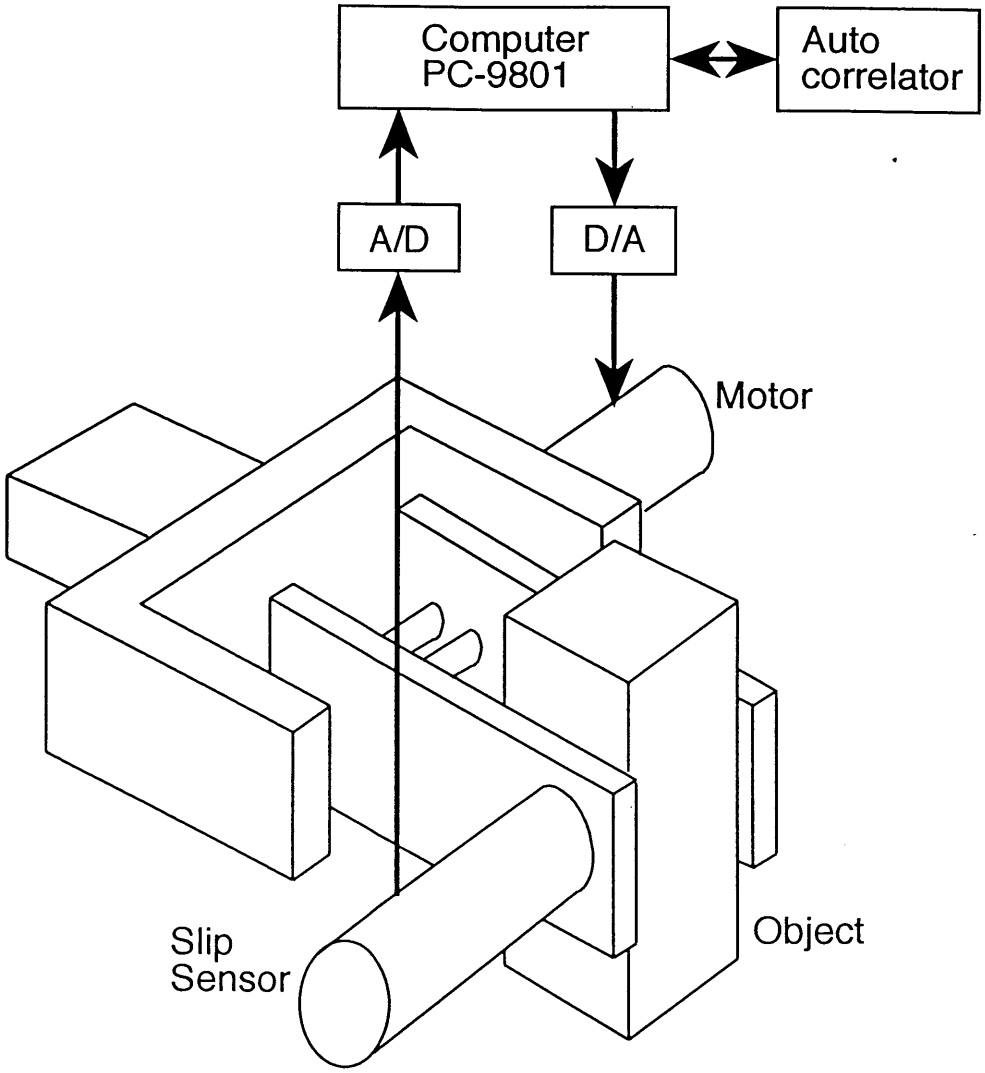


Fig.2.6.1. Schematic diagram of the grasping force control.

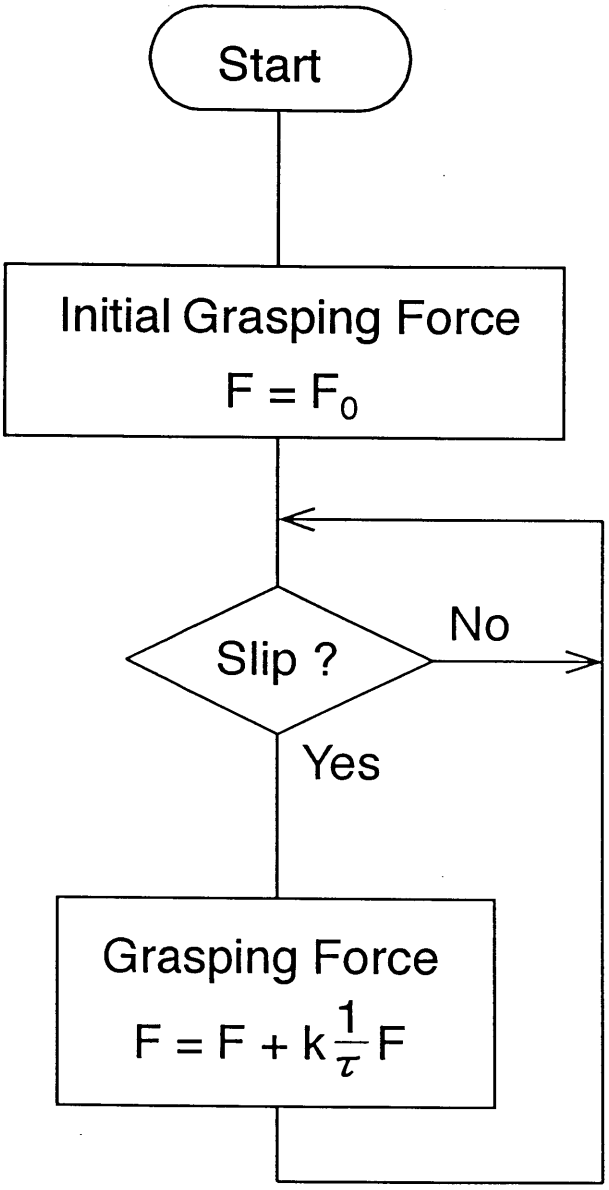


Fig.2.6.2. Flow chart of the grasping force control.

まず，格子像投影型スリップセンサの基礎実験を行い，自己相関関数を用いた信号処理手法を示した．次に，ロボットハンドに装着可能な小型スリップセンサを製作した．さらに，このセンサを改良して，物体形状の変化に対応可能な二光束干渉型スリップセンサを開発し，最後に，スリップセンサを用いた把握力制御の方法について概説した．

以下に本センサの特徴を示す．

1. 非接触である．

2. 半導体レーザを用い，受光素子と一体化しているため小型である．

3. 物体形状による影響が少ない．

4. 鏡面，黒体面を除くあらゆる物質に適応可能である．

5. 微少すべりが検出可能である．

6. 距離センサへの拡張が可能である．

2.8 今後の課題

本章では，ロボットハンドに装着可能なスリップセンサを開発し，その基礎実験を行った．今後，本センサを用いて把握制御を行うための課題としては，以下の点があげられる．

1. 本センサを距離センサとして使用し，対象物体までの距離計測を行う．

2. センサをロボットハンドに装着し，距離計測を行いながら対象物体の回りを一回転させることにより，物体の輪郭データを得る．

3. 得られた輪郭データより，物体の最も把持しやすい位置を推定する．

4. 推定した位置にロボットハンドを移動させる．

5. 同じセンサをスリップセンサとして使用し，2.6節で示した把握力制御を行う．

また，本センサは速度センサとして，他のロボット応用にも用いることができる．その応用例の一つとして，移動ロボットのための2次元対地速度センサがあげられる．

本スリップセンサでは，対象物体に投影した干渉縞に対して斜め方向に物体が移動した場合，干渉縞に垂直な方向の速度成分のみが信号として得られるため，干渉縞の方向をずらした2台のセンサを用いるこ

とにより，2台のセンサの出力の速度ベクトルを合成して，2次元の速度検出を行うことが可能となる．また，本センサの性質上，干渉縞の水平方向付近の速度検出は困難であるが，方向を調整した（例えば 60° ずつずらした）3台のセンサを設置することにより，常に信号の得易い出力を選択することが可能になる．

参考文献

[1] M.Ueda, K.Iwata and H.Singu, "Tactile Sensors for an Industrial Robot to Detect a Slip", *Proc. 2nd ISIR*, pp.63-76, 1972.

[2] A.K.Bejczy, "Sensor System for Automatic Grasping and Object Handling", *Colloques IRIA*, pp.279-299, 1978.

[3] R.Tomovic and Z.Stojilikovic, "Multifunctional Terminal Device with Adaptive Grasping Force", *Automatica*, 11, pp.567-571, 1975.

[4] 増田良介, 長谷川健介, 尾嵜一義, "工業ロボットのすべり覚とその応用", 電気学会論文誌, 96-10, 1975

[5] 増田良介, 長谷川健介, 尾嵜一義, "工業ロボットのすべり感覚器", 第 11 回 SICE 学術講演会予稿集, 3204, 1972

[6] R.C.Luo and J.Y.Chang, "Robot Grasping Force Control Using Force/Slip Sensory Feedback System", *Proc. IECON*, Vol.1, 1985.

[7] A.K.Bejczy, "Sensors, Controls, and Man-Machine Interface for Advanced Teleoperation", *SCIENCE*, 208-4450, pp.1327-1335, 1980.

[8] 鈴木豊一, 増田良介, "分布型触覚センサによるロボットの力制御", 第 4 回日本ロボット学会学術講演会予稿集, pp.311-312, 1986

[9] A.E.Brennemann, R.L.Hollis, M.A.Lavin and B.L.Musits, "Sensors for Robotic Assembly", *Proc. IEEE Int. Conf. Robot Automation*, Vol.3, pp.1606-1610, 1988.

[10] 塩野幸策, "レーザ・ドップラ方式による非接触速度, 速度ムラ, 回転ムラ測定システム", 計測技術,'87.12, pp.72-77, 1987

[11] S.Shinohara, A.Mochizuki, H.Yoshida and M.Sumi "Laser Doppler Velocimeter using the Self-Mixing Effect of a Semiconductor Laser Diode", *APPLIED OPTICS*, Vol.25, No.9, pp.1417-1419, 1986.

[12] 松田文夫, 森元裕志, 北川秀夫, 服部秀三, 上田実, "物体散乱光信号の自己相関処理を用いたすべり検出法", 日本ロボット学会誌, Vol.4, No.6, pp.602-606, 1986

[13] F.Matsuda, H.Morimoto, H.Kitagawa, S.Hattori and M.Ueda, "Control of an Industrial Robot with Nontactile Slip Sensor Using Auto Correlation Proccessed Laser Scattering Signal", *Proc. 17th ISIR*, pp.6·1-6·13, 1987

[14] 黒田敏秋, 北川秀夫, 松田文夫, 内川嘉樹, 服部秀三, "格子像投影型非接触スリップセンサの開発", 日本ロボット学会誌, Vol.9, No.4, pp.466-470, 1991

第3章　高速演算のためのマルチプロセッサ・スケジューリングアルゴリズム

3.1　はじめに

前章で述べたように，広い測定範囲，高分解能のセンサを実現するためには，一般に計算量の増加が伴う．さらに，装着するセンサの数が増加した場合，この計算量の増加はさらに顕著になり，リアルタイム制御が困難になる傾向がある．

リアルタイム制御を達成するためには，高速計算可能な信号処理アルゴリズムの開発，及び高速処理を行うコンピュータシステムの利用が必要である．後者については，DSP などの高速演算プロセッサによる計算または複数のプロセッサを用いた並列処理などの手法があげられる．DSP とは，主に信号処理等で用いられる繰り返し計算を，高速に行うために開発されたプロセッサであり，積和演算に関しては，パイプラインを構成することによって高速化が可能になる．ただし，プログラムの開発環境が未発達で，プログラミングの困難さに問題点がある．本章では，複数のプロセッサを用いた並列処理手法について述べる．

複数のプロセッサを用いた並列処理手法としては，演算パイプライン方式，マルチプロセッサ方式，データフロー方式などがあげられるが，パイプライン・コンピュータは小マトリクス・ベクトル演算が苦手であり，データフローマシンは，その非決定的なデータ駆動型実行のために，リアルタイム・オンライン制御に要求されるサンプリング・タイムの管理が困難である．従って，リアルタイム・ロボット制御には，マルチプロセッサ方式が最も適していると考えられる．

マルチプロセッサによる並列処理の手順は，単一ジョブの高速処理を目的とする場合，一般に Fig.3.1.1 に表されるように，解法の決定，タスク分割，スケジューリングの順で行われる．しかし，各段階においてそれぞれ以下のような問題点があげられる．まず，与えられた問題をいかに定式化するかという解法の決定については，並列処理で特に問題となる通信量や通信のオーバヘッドを考慮した，解の導出アルゴリズムが明らかにされていない．次に，与えられた問題をいかに処理単位であるタスクに分割するかというタスク分割については，与えられた解法を処理時間の短いタスク集合に分割する手法が未開発である．さらに，スケジューリングについて，従来の方法 [1, 2] では組合せ計算の複雑さを軽減するためにプロセッサ間のデータ通信に要する時間が無視されていたが，ハードウェアの進歩，乗算器の内蔵等により，マイクロ

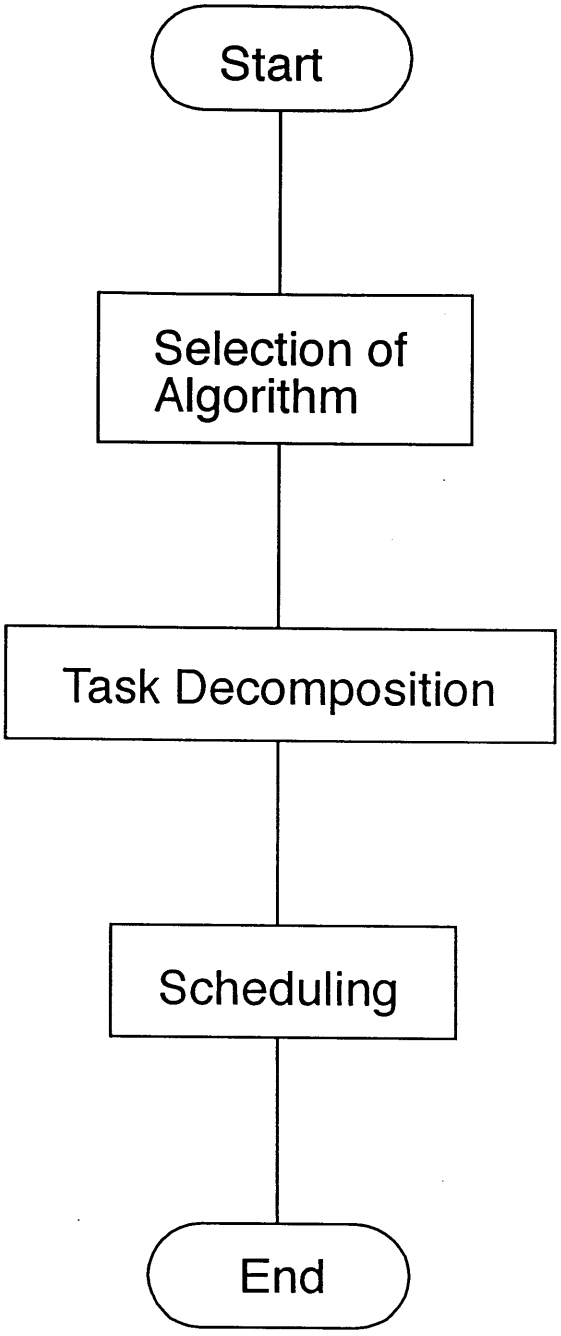


Fig.3.1.1. Flow chart of multiprocessor parallel processing.

プロセッサの演算能力が飛躍的に向上しているため，相対的にプロセッサ間のデータ転送によるオーバーヘッドが無視できない状況になっている．

本章では，上述の各段階のうち，スケジューリング問題をとり上げる．まず，処理対象を任意として，プロセッサ間の通信時間を考慮した，スケジューリングアルゴリズムを提案する．次に，提案したスケジューリングアルゴリズムを用いてシミュレーションを行い，処理時間を評価する [3, 4]．

なお，本章で対象とするのは，MIMD(Multiple Instruction/Multiple Data Stream) 方式のマルチプロセッサシステムにおいて，相互結合された処理能力の等しい任意の数のプロセッサに，処理時間が既知である任意の数のタスクを静的に割り付けるという問題である．各プロセッサは相互に直結された完全網を構成し，共有メモリを持たず，プロセッサ間の通信はメッセージ転送によってのみ行われるものとする．

ターゲットコンピュータとして，疎結合・メッセージ転送型プロセッサであるトランスピュータ（IMS T800）[5, 6] を使用する．トランスピュータは，処理内容に対する柔軟性を持った汎用性と規模拡張性のある高速実時間用並列プロセッサであり，通信用の高速シリアルリンクを他チップのリンクと結合することによって，容易にマルチプロセッサシステムを構築することが可能である．

Fig.3.1.2 に示すように，このプロセッサとローカルメモリをプロセッシングエレメント（PE）とし，PE 間の通信は高速シリアルリンクを通しメッセージ転送によって行う．このマルチプロセッサシステムは共有メモリもバスも持たないため，構成が簡単で拡張性が良いという特徴を持つ．Table 3.1.1 にシミュレーションに用いたトランスピュータの処理時間を示す．

以上の設定について，まずトランスピュータを用いた理由としては，モジュール性及び拡張性の良さがあげられる．また，プロセッサの結合方式等の設定理由としては，ロボットの逆運動学，逆動力学問題等への適用を考えて，効率の良い（無駄な通信，同期オーバヘッドの少ない）システムを構築できる点があげられる．

3.2 マルチプロセッサによる並列処理

以下に，Fig.3.1.1 で示した単一ジョブの高速化を目的とした，マルチプロセッサによる並列処理の一般的な手順の詳細を示す [1, 7]．そして，この中のスケジューリング問題について，従来考慮されていなかった通信時間を入れたアルゴリズムを 3.3節で提案する．

・解法の決定　まず，与えられた問題を解くアルゴリズムを求める．このとき，並列処理に向いたアルゴリズムを求めることが望まれる．タスク分割を行ったとき，得られるタスク集合の並列度が高い（先行制約が少なく，同時に実行可能なタスク数が多い）程，そのアルゴリズムは並列処理向きであると

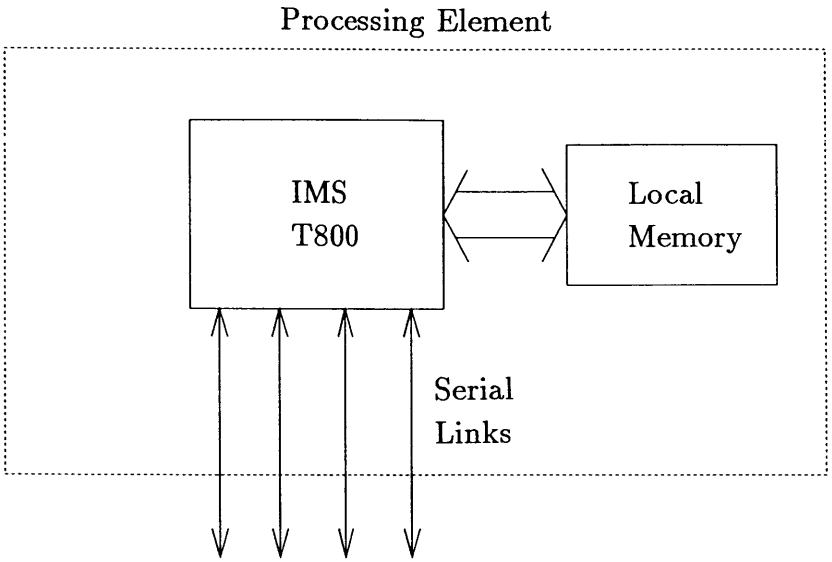


Fig.3.1.2. Architecture of the target computer.

Table 3.1.1. Processing time of the target computer.

Operation	Processing time (μs)
Addition	1.1
Subtraction	1.3
Multiplication	1.4
Division	1.7
Data transmission	5.0

いえる．

・タスク分割　スケジューリングの前処理段階において，対象とする計算式は処理単位となる複数のタスクに分割される．分割されたタスク間の従属関係（先行制約）は Fig.3.2.1 に示すような無サイクル有向タスクグラフ $G=(V,E)$ によって表される．ただし，V,E はそれぞれ以下に示すような節点集合，枝集合を表している．図中の円（節点）は分割されたタスクを，その中の数字は各タスクのタスク番号を表し，円の横の数字はタスクの処理時間を表す．これらは節点集合 $V=(T_1,T_2,\ldots,T_n)$ で表される．また，節点間を結ぶ枝はそのタスク間の先行制約，即ち枝の上側のタスクが終了するまでは下側のタスクは実行できないという条件を表す．これらは枝集合 $E=(e_1,e_2,\ldots)$ で表される．T_i から T_j へ到る枝集合が存在する場合，T_j を T_i の後続タスク（successor）と呼び，T_i を T_j の先行タスク（predecessor）と呼ぶ．さらに，T_i から直接 T_j に到る枝が存在する場合，T_j は T_i の直接の後続タスク（immediate successor）であると言う．

タスクグラフの各タスクにはその処理時間が重みとして与えられていて，タスク間のパス長はその重みの和で定義される．タスクグラフにおける最長のパスをクリティカルパスといい，その長さをクリティカルパス長という．クリティカルパス長はその問題における処理時間の下限（プロセッサ数をいくら増加させても，先行制約のためにそれ以上は処理時間が短くならないという限界の時間）を意味している．処理時間がクリティカルパス長に近づいた場合，それ以上プロセッサ数を増加させても，タスク間の先行制約及び通信に要する時間のために，処理時間が短縮できない [8]．また，各タスクについて，そのタスクから終端タスクまでの最長のパス（そのタスク以後のクリティカルパスに相当するもの）をそのタスクのレベルと言う．

与えられた問題に対する解法が同じ場合でも，このタスク分割の仕方によって生成されるタスク集合が異なるため，スケジューリング結果の並列処理時間に違いが現れる．

・スケジューリング　マルチプロセッサにおけるスケジューリングとは，タスク分割によって得られた複数のタスクをどのプロセッサに割り付け，どのような順序で実行するのかを決定することである．ただし，本章で対象とするのは，プログラムの実行前にタスク集合が完全に与えられる決定性の問題に対する静的スケジューリングである．

一般に，スケジューリングの結果は Fig.3.2.2 に示すようなガントチャートと呼ばれるタイムチャートによって表される．図中，T_i は Fig.3.2.1 におけるタスク i の実行を示し，斜線の部分はプロセッサが先行タスクの終了待ちで空き状態になっている時間を示す．本節では，次節で提案するスケジュー

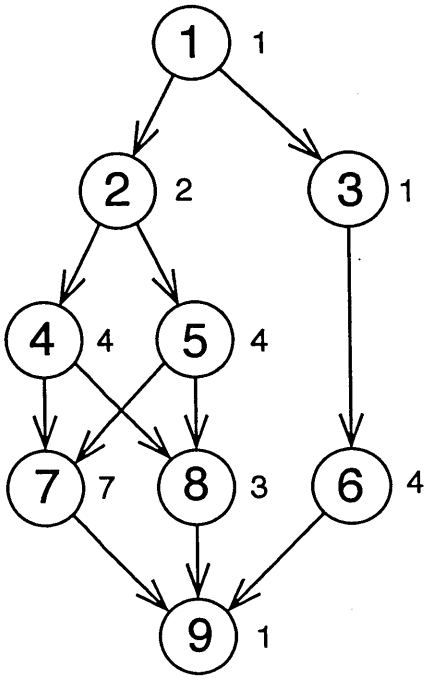


Fig.3.2.1. An example of a task graph.

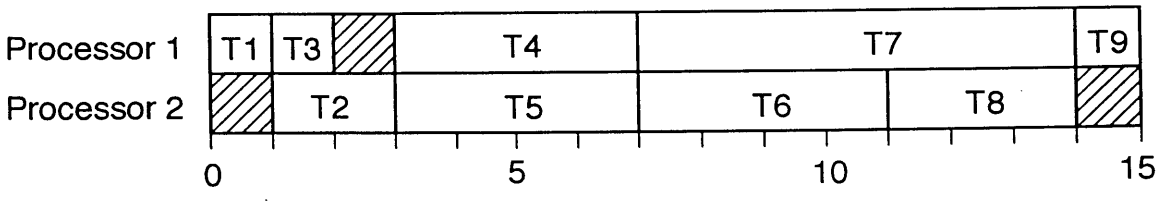


Fig.3.2.2. Gantt chart.

リングアルゴリズム，及びこれまで提案されている多くのスケジューリングアルゴリズムの基礎となっている，リストスケジューリングと呼ばれるスケジューリング法について説明する．

リストスケジューリングの手順は次の通りである．

1) 優先順位リストの作成

タスクを優先順位の高い順に並べたリストを作成する．

2) プロセッサへの割り付け

プロセッサが空き状態になるたびに，処理可能なタスクの中から優先順位リストの先頭に最も近いタスクを選び，そのプロセッサに割り付ける．

また，1) の優先順位リストの作成アルゴリズムは，基本的に以下の通りである．

i. タスクグラフの終端ノードにあるタスクに最も低い優先順位をつける．

ii. 優先順位を付けられたタスクの直接の先行タスクすべてに対して，昇順の優先順位を付けていく．

iii. すべてのタスクに優先度が付けられるまで，前項を繰り返す．

iv. 各タスクを，付加された優先順位の高い順にリストにつなぐ．

3.3 通信時間を考慮に入れたスケジューリングアルゴリズム

3.3.1 スケジューリングアルゴリズム

本節では，分割された個々のタスクを，システム内のどのプロセッサに割り付けるかを決定するスケジューリングアルゴリズムについて述べる．予め分割されたタスクは以下のステップにより，各プロセッサに割り付けられる．

1) 優先順位リストの作成

タスクには最初に適当なタスク番号が付けられているが，全てのタスクについて式 (3．1) の評価関数を計算し，その降順にタスクを並べ換えてタスク番号を付け直す．

［タスク評価関数］

$$f(i) = \frac{t_{lev}(i)}{t_{total}} + \mathrm{S}n_{is}(i) + k\frac{t_{comm}}{t_{avg}}\frac{n_s(i)}{n_{total}} \tag{3.1}$$

$$(i = 1, 2, \ldots, n;\ n : \text{タスク数})$$

ここで，各変数は以下の意味を持つ．

$t_{lev}(i)$ ： タスク i のレベル

t_{total} ： 全タスクの処理時間の合計

S ： 直接後続タスク係数

$n_{is}(i)$ ： タスク i の直接の後続タスク数

k ： 通信係数

t_{comm} ： データ 1 語の転送に要する時間

t_{avg} ： タスクの処理時間の平均

$n_s(i)$ ： タスク i の後続タスク数

n_{total} ： 全タスク数

以降，全てのタスクについて以下のステップを繰り返す．

2) *target task* の選定

未処理（これまでの段階でまだ特定のプロセッサに割り付けられていない）タスクで，その先行タスクがすべて処理されているもの（*ready task*）の中から，前項で並べ換えられたタスク番号の一番小さいものを選び，次に処理するタスク（*target task*）とする．

3) *target prosessor* の選定

全てのプロセッサについて下式の評価関数を計算し，その値の一番小さいものを選び，*target task* を割り付けるプロセッサ（*target processor*）とする．

［プロセッサ評価関数］

$$g(i) = max(t_{ready}, t_{end}(i)) + n_{data}(i)t_{comm} \tag{3.2}$$

$$(i = 1, 2, \ldots, m;\ m : \text{プロセッサ数})$$

ここで，各変数は以下の意味を持つ．

t_{ready} ： *target task*の全先行タスクの終了時刻

$t_{end}(i)$ ： プロセッサ i が空き状態になる時刻

$n_{data}(i)$ ： プロセッサ i への転送データ数

t_{comm} ： データ 1 語の転送に要する時間

4) タスク及び通信時間の割り付け

target task を *target processor* に割り付け，さらにプロセッサ間の通信時間を送受信プロセッサに割り付ける．

以上のステップによって，全てのタスクが各プロセッサに割り付けられる．そのフローチャートを Fig.3.3.1 に示す．

式 (3．1) のタスク評価関数における $t_{lev}(i)/t_{total}$ はタスク i のレベルを全タスクの処理時間で正規化したものである．また，第 2 項の S は第 1 項の値が等しいときのみ直接の後続タスク数 $n_{is}(i)$ によって差が出るように設定した十分に小さい定数である．さらに，第 3 項の t_{comm}/t_{avg} はデータ通信時間とタスクの平均実行時間との比であり，$n_s(i)/n_{total}$ はタスク i の後続タスク数を全タスク数で正規化したものである．この二つの積 $(t_{comm}/t_{avg})(n_s(i)/n_{total})$ が大きい程，データ転送の単位時間とデータ転送数の積が大きくなり，データ転送の要素が大きくなる．k の値は通信の要素への重み付けを示し，$k=0$ のとき，この評価関数による優先順位リストは，通信の要素を無視した場合のものと等価になる．k の値を可変にしてスケジューリングを行い，結果として得られた計算時間のうち最も短いものを解とする．

式 (3．2) のプロセッサ評価関数における第 1 項は，ターゲットタスクがプロセッサ i に割付け可能になる時刻を表す．第 2 項は，ターゲットタスクをプロセッサ i に割付けた場合に必要となるデータ通信に要する時間である．

式 (3．1) の第 3 項，および式 (3．2) の第 2 項を導入することによって，プロセッサ間の通信時間を考慮したスケジューリングが可能となる．

本スケジューリングアルゴリズムの特徴としては以下の点があげられる．

1. プロセッサ間通信時間を考慮に入れている．
2. リストスケジューリングを基調としていて，組み合わせ計算を行う必要がないため，スケジューリングに要する計算量が少ない．

プロセッサが空き状態になった瞬間に直ちにレディタスクを割り付けるよりも，一時的にプロセッサを空き状態のままにしておいた方が，処理時間が短くなる可能性がある．例えば，Fig.3.2.2 において，タス

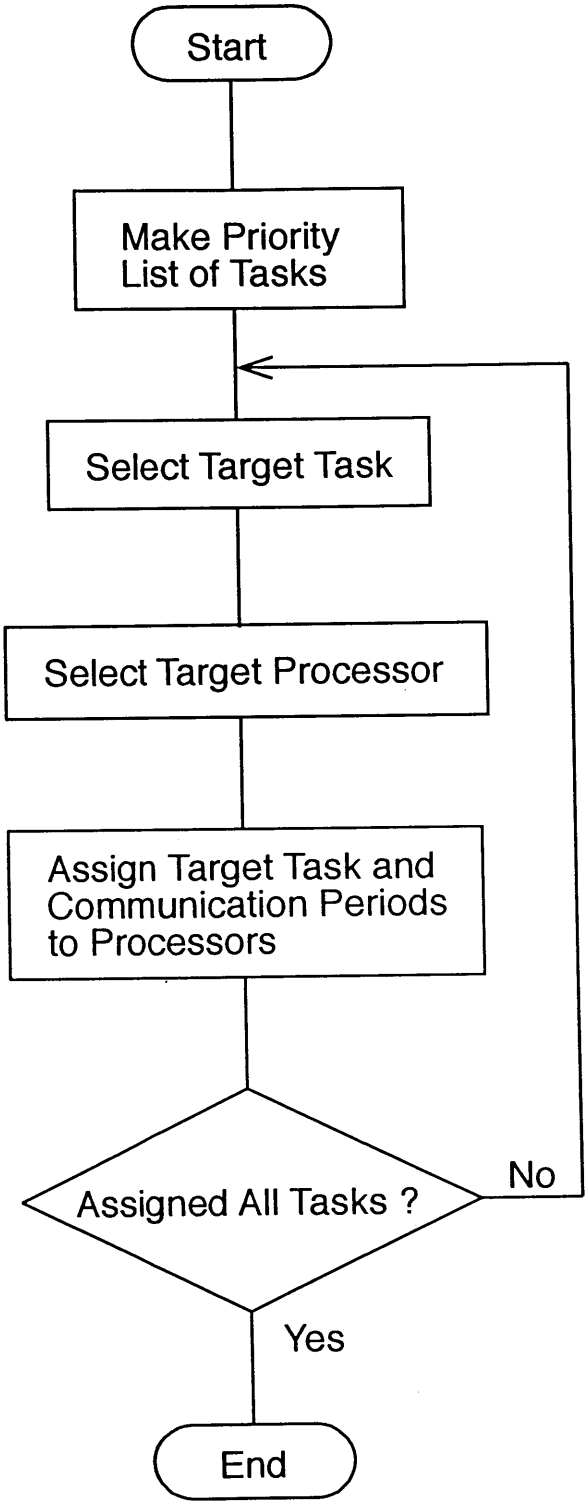


Fig.3.3.1. Flow chart of the scheduling algorithm.

ク 3 の終了後に直ちにタスク 6 を割り付けてしまうと，タスク 4 またはタスク 5 の実行が遅れてしまい，その結果，タスク 7,8,9 の実行も遅れてしまう．一時的にプロセッサを空き状態にする手法を，通信時間を考慮にいれたスケジューリングに取り込むためには，本手法を分枝限定法に拡張するなどの方策が有効であるが，この場合，生成されるノード数の増大が避けられない．

元来複雑であるスケジューリング問題に更に通信時間を含める場合には，本手法のようにリストスケジューリングを用いることによって，実用的な時間内での求解を可能とする方法が有効である．

3.3.2 シミュレーション結果及び考察

タスク数 30 ～ 100 ，タスクの平均処理時間 5 ～ 100μsec ，平均直前先行タスク数 1.0 ～ 3.0 としてランダムに発生させた 1150 種類のタスク集合に対して，本アルゴリズムを適用した．プロセッサ数 2 ～ 7 の場合に，通信時間を考慮しないリストスケジューリングである CP/MISF 法に対して，本アルゴリズムでは処理時間が平均約 6% 短縮されている．特に，タスク数が多く，通信時間に対するタスクの処理時間が短く，直前先行タスク数が多い程，本アルゴリズムの有効性が示されている． Fig.3.3.2 に本章で提案したスケジューリングアルゴリズム，及び従来の通信時間を考慮しないリストスケジューリングとして CP/MISF 法を適用した結果の一例を示す．図中，横軸は結合されるプロセッサ数を示し，縦軸は処理時間を示す．提案したスケジューリングアルゴリズムを用いることによって処理時間が短縮されていることが分かる．なお，プロセッサ数が 6 の時の処理時間が 5 の時より長くなっているのは，スケジューリングアルゴリズムが高速化のための準最適解法であることに起因すると考えられる．

本スケジューリングアルゴリズムでは，通信オーバヘッドを考慮することによって，現実に近いスケジューリングを行っている．しかし，このことはただでさえ複雑なスケジューリング問題を，より複雑なものにしているため，リストスケジューリングを行うことによって現実的な計算時間で解を求めることが出来るようにしている．最適スケジュールを行うためにはプロセッサをアイドル状態にすることが必要な場合があるが，スケジューリングに要する時間が増大しても最適解が必要な場合は，分枝限定法に式 (3 . 1) を適用し，アイドルタスクを取り入れれば良い．

3.4 まとめ

プロセッサ間の通信時間を考慮に入れたマルチプロセッサスケジューリングアルゴリズムを提案し，その有効性を示すために MIMD 方式の相互結合されたプロセッサを対象としたシミュレーションを行った．ランダムに発生させたタスク集合に対して本アルゴリズムを適用した結果，通信時間を考慮しないリスト

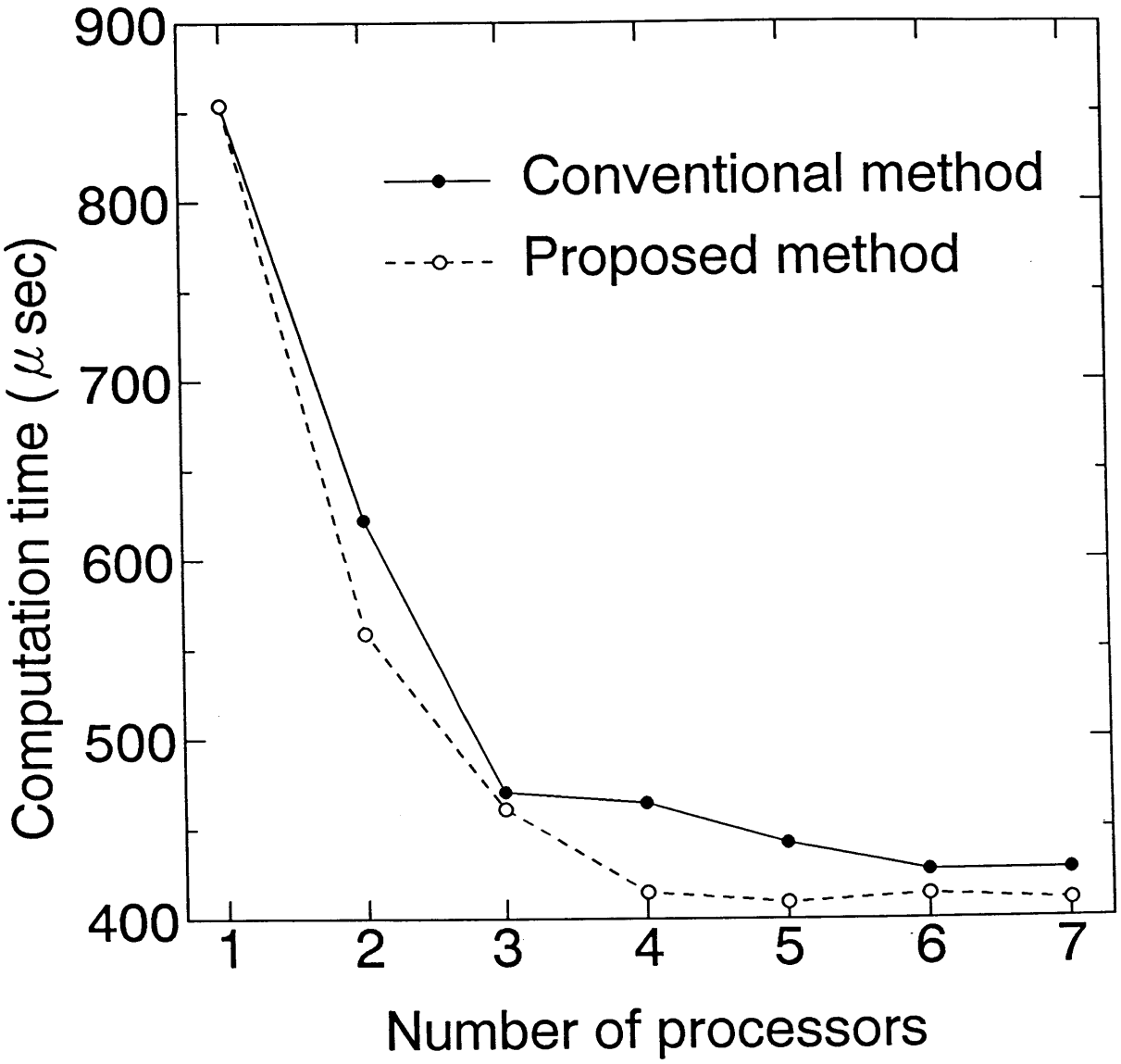


Fig.3.3.2. Computation time of example equations.

スケジューリング法に対して，処理時間が平均約 6% 短縮された．本アルゴリズムは以下のような特徴を持つ．

1. プロセッサ間のデータ通信に要する時間をパラメータとして考慮に入れているため，より実システムに即したスケジューリングが可能になる．

2. 準最適解法であるリストスケジューリングを基調としていて，組み合わせ計算を行う必要がないため，スケジューリングに要する時間が短くなる．

本章では，マルチプロセッサシステムの結合方式として完全網を用いたが，異なる結合方式を用いた場合のスケジューリングアルゴリズムの修正，及びその評価も必要である．非閉塞網等のプロセッサ間距離（スイッチ段数）が一定の結合網に対しては，本章で提案した手法がそのまま適用可能である．また，2 進 n-キューブ網等のプロセッサ間距離が静的に決定している回路網に対しても，各プロセッサ間距離を評価関数に入れることにより適用可能である．

また，今後のロボット応用では，外界の状況に対応するための実時間性が要求される．さらに，複数のロボットが分散的に通信，協調しながら行動する場合も多くなるため，モジュール性を持ち，かつプロセッサ間の通信能力も高いトランスピュータのようなプロセッサを用いた，分散型コンピュータアーキテクチャが重要になると考えられる．

参考文献

[1] 笠原博徳, 成田誠之助, “マルチプロセッサ・スケジューリング問題に対する実用的な最適及び近似アルゴリズム”, 電子情報通信学会論文誌D, J67-D, No.7, pp.792-799, 1984

[2] C.L.Chen, C.S.G.Lee and E.S.H.Hou, “Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System”, *IEEE Trans. Syst.,Man and Cybern.*, Vol.18, No.5, pp.729-743, Sep.1988.

[3] H.Kitagawa, M.Ito, F.Matsuda, Y.Uchikawa and S.Hattori, “Scheduling of parallel processing including communication time”, *Proc. Autotech Asia '89*, pp.62-79, 1989

[4] 北川秀夫, 松田文夫, 内川嘉樹, 服部秀三, “通信時間を考慮に入れたマルチプロセッサスケジューリングアルゴリズム”, 電子情報通信学会論文誌, Vol.J73-D-I, No.10, pp.812-817, 1990

[5] “Transputer Reference Manual”, Inmos, Oct.1986.

[6] 山崎弘郎, 石川正俊, “センサフュージョン”, コロナ社, 1992

[7] 富田眞治, 末吉敏則, “並列処理マシン”, オーム社, 1989

[8] 坂井修一, “並列計算機におけるスケジューリングと負荷分散” 情報処理, Vol.27, No.9, pp.1031-1038, 1986

第4章　ロボットの逆運動学計算と逆動力学計算の高速化

4.1　はじめに

マニピュレータに，センサ情報等に基づいたフレキシブルな動作を行わせるためには，リアルタイムで逆動力学方程式などを計算する必要がある．しかし，それらの計算には多大な計算量が要求されるため，通常のマイクロコンピュータを用いただけでは，満足する結果を得ることができない．このような計算を高速に行わせる方法として，複数のマイクロプロセッサによる並列処理が上げられる．

その他の方法として，処理速度を上げるために固定小数点乗算器 [1, 2]，DSP [3] や専用ハードウェア [4, 5] を用いる方法も提案されているが，処理の簡略化により精度の低下を招いてしまったり，プログラムの開発に時間と労力がかかるために，仕様変更に柔軟に対応できないという欠点がある．

本章では，第 3 章で述べたマルチプロセッサによる並列処理を，ロボット制御計算に適用する．なお，ターゲットシステムとして，並列処理専用プロセッサであるトランスピュータ [6] を用いるため，高級言語 Occam によるプログラミングが可能となる．

第 3 章で示したように，マルチプロセッサによる並列処理の手順は，単一ジョブの高速処理を目的とする場合，一般に Fig.3.1.1 のように表される．第 3 章では，このうちのスケジューリング問題をとり上げて，通信時間を考慮に入れたスケジューリングアルゴリズムを提案し，その有効性を示した．しかし，その前処理としての解法の決定については，通信オーバヘッドを考慮した解の導出アルゴリズムが明らかにされていないし，タスク分割についても，コマンド，演算，方程式等の単位でおおまかにタスク粒度（タスクの大きさ）を選択する方法が開発されているだけである．

そこで，本章では，ロボットマニピュレータの逆運動学問題を対象として，通信のオーバヘッドを考慮した並列処理向きの解法を示す．次に，逆動力学問題を対象として，通信量と通信のオーバヘッドを考慮したタスク分割法を示す．それらの逆運動学，逆動力学問題に関して，それぞれ第 3 章で提案したスケジューリングアルゴリズムを適用してシミュレーションを行い，処理時間を評価する [7]．

なお，本章で対象とするのは，第 3 章と同じく，トランスピュータを用いたマルチプロセッサシステムである．このシステムでは，各プロセッサが相互に直結された完全網を構成し，共有メモリを持たず，プ

ロセッサ間の通信はメッセージ転送によってのみ行われる.

4.2 逆運動学問題の並列処理向き解法

与えられた問題を解くアルゴリズムによって，得られるタスクグラフの構造は異なり，それに応じて並列処理時間も変わる．従って，与えられた問題の解法としては，処理時間を小さくするという意味で，並列処理に適したタスクグラフが得られるような解法が望ましい．そこで，本節ではロボットマニピュレータの逆運動学問題に対する並列処理向き解法を提案し，シミュレーションによってその有効性を示す.

4.2.1 ロボットの逆運動学問題

逆運動学問題とは，ロボットの手先の位置，姿勢ベクトルを入力とし，その位置，姿勢を実現するための各関節の変位を出力として求める問題である．なお，本節では，スタンフォード・マニピュレータを例として用いるが，以下に述べる手法は一般の多関節型マニピュレータに適用可能である.

Fig.4.2.1 にスタンフォード・マニピュレータ [8]，Fig.4.2.2 にそのリンク構成を示す．6 自由度垂直多関節型マニピュレータであるが，第 3 関節（図中の d_3）のみ直動関節である．Fig.4.2.3 にスタンフォード・マニピュレータの逆運動学方程式を示す．入力は，手先の位置を表示するベクトル $\boldsymbol{p}$，及び手先の姿勢を表示する 3 つの単位直交ベクトル $\boldsymbol{n}, \boldsymbol{o}, \boldsymbol{a}$ を用いて，位置姿勢行列

$$\boldsymbol{T}_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

で表され，出力は関節変位

$$\boldsymbol{q} = \begin{bmatrix} \theta_1 & \theta_2 & d_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}^T \tag{4.2}$$

で表される.

4.2.2 並列処理における解法の決定

Fig.4.2.3 の各式及び θ_1 の第 1 項，第 2 項を，それぞれ 1 つのタスクとするタスク集合を考える．このうち，S_i, C_i の値はそれぞれ θ_i の正弦 $(\sin\theta_i)$ 及び余弦 $(\cos\theta_i)$ であり，$i = 1, 2, 4$ の場合，それぞれ関数 $\tan^{-1}()$ によって求められた θ_i の値を用いて計算されている．しかし，この S_i, C_i は θ_i を求める関数 $\tan^{-1}()$ の引数から直接計算可能であり，この場合，S_i, C_i は Fig.4.2.4 の式で与えられる．以後，

解法 I： $\tan^{-1}()$ の結果を用いて S_i, C_i を計算する解法

解法 II： $\tan^{-1}()$ の引数から直接 S_i, C_i を計算する解法

として，その並列処理について考える．

Fig.4.2.5 に，それぞれの解法に対するタスクグラフを示す．図中，タスク番号 0 はタスクグラフの入口ノード，タスク番号 28 はタスクグラフの出口ノードであり，共に処理時間は 0 となる．また，タスク番号 1 ～ 27 のタスクは Table 4.2.1 の通りである．但し，タスク番号 3 以降は Fig.4.2.3 の左辺の記号をそのままタスク名として用いている．このうち，タスク θ_i, S_i 及び C_i 間のデータの流れは Fig.4.2.6 で表される．S_i, C_i は解法 I では θ_i に従属している（θ_i は S_i, C_i の先行タスクである）が，解法 II ではそれぞれが独立しており，解法 II の方が並列度の高い解法であることが分かる．各タスクをそれぞれ一つのプロセッサで処理する場合を考えると，そのガントチャートは Fig.4.2.7 のようになる．プロセッサ間のデータ通信に要する時間を t，また，タスクの処理時間を簡単化のためにすべて T とした．図中，斜線はプロセッサが空き状態であることを示し，ドット（・）はデータ通信が行われていることを示す．Fig.4.2.7 のようなスケジューリングを行うとき，解法 II の処理時間が解法 I よりも短い条件は，ガントチャートより $T > 2t$ である．このとき，通信のオーバヘッド（データ待ちの時間）が減少することにより，全体の処理時間が短くなる．特に，プロセッサ間のデータ通信に要する時間 t がタスクの処理時間 T に比べて十分小さいとき，解法 II は常に解法 I より処理時間が短くなる．

4.2.3 シミュレーション結果及び考察

Fig.4.2.8 に，解法 I 及び解法 II のそれぞれに対して，第 3 章で提案したスケジューリングアルゴリズムを適用して求めた処理時間を示す．ここで，解法 II' で示されるデータは，解法 I と解法 II のタスクグラフの構造変化による処理時間の変化を純粋に比較するために，タスクグラフの形状は解法 II のままでタスク θ_i, S_i 及び C_i の処理時間のみを解法 I と同じにしたものである．

解法 II' では，タスクグラフから分かるように，並行して処理可能なタスク数が増加したために，解法 I に比べて処理時間が短縮されている．

さらに，解法 II では，S_i, C_i の計算にかかっていた時間が短縮されたため，解法 I に比べて処理時間が平均 46.9%，最大 58.3% 短縮されている．

なお，プロセッサ数 6, 7 あたりで処理時間の短縮が見られなかったのは，問題自体の並列度が低かったこと及びマニピュレータのシリアルリンク機構に起因する先行制約が多かったことが原因と考えられる．

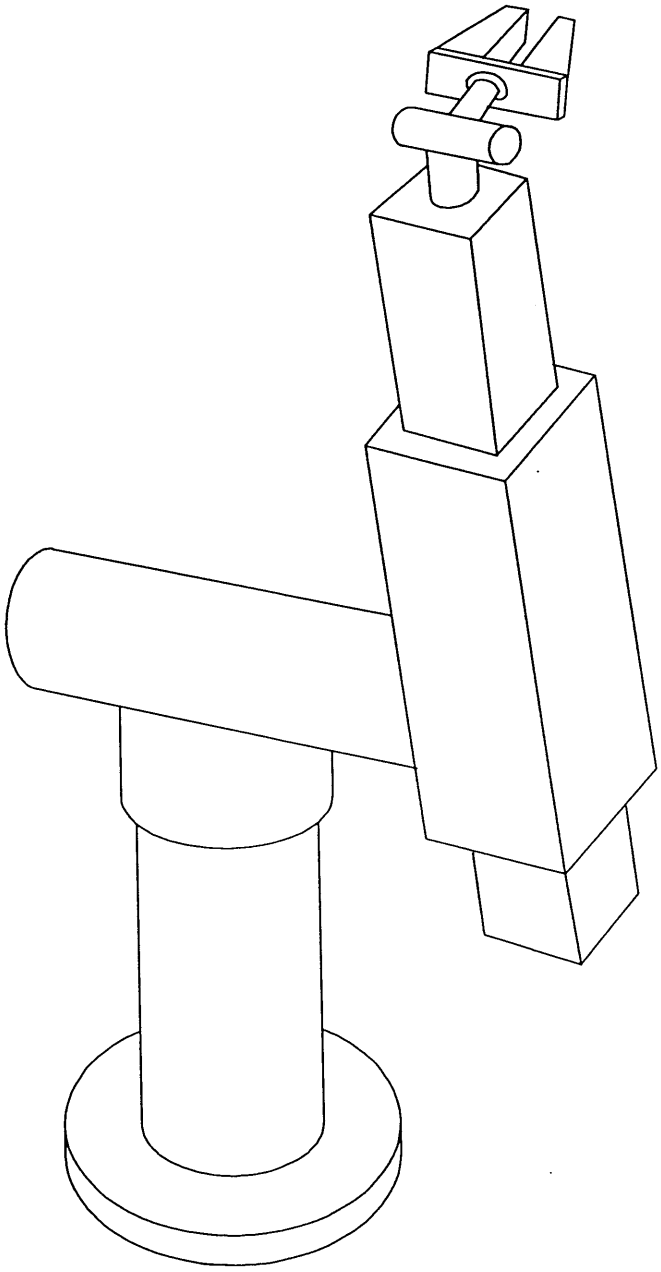


Fig.4.2.1. Stanford manipulator.

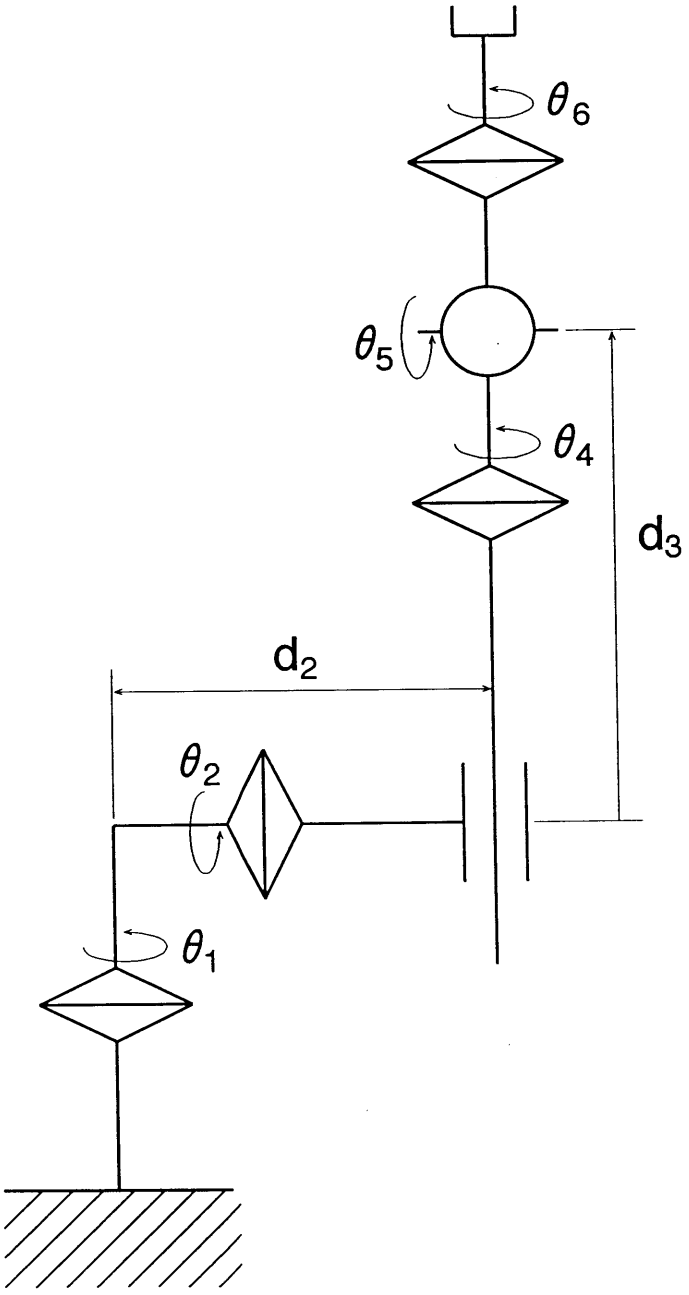


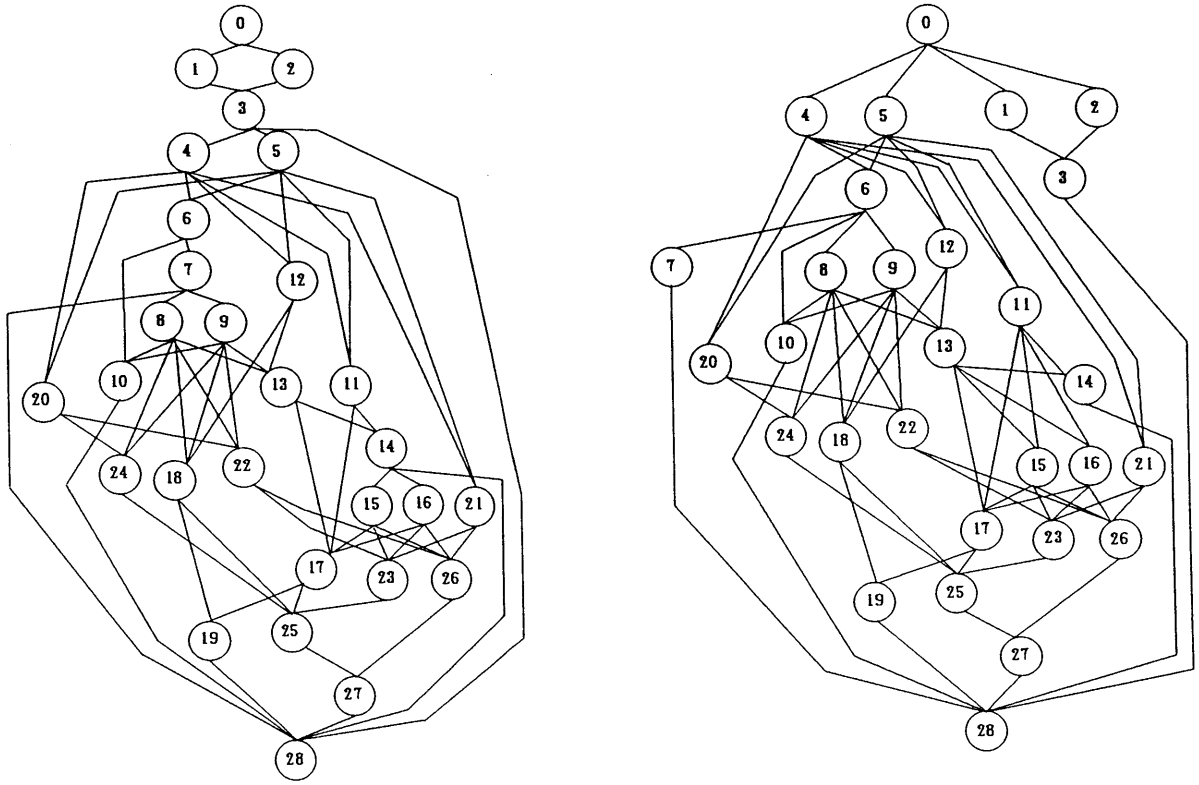
Fig.4.2.2. Skeleton model of the Stanford manipulator.

$$
\begin{aligned}
\theta_1 &= \tan^{-1}\frac{p_y}{p_x} \\
&\quad -\tan^{-1}\frac{d_2}{\sqrt{{p_x}^2+{p_y}^2-{d_2}^2}} \\
S_1 &= \sin\theta_1 \\
C_1 &= \cos\theta_1 \\
M_1 &= C_1 p_x + S_1 p_y \\
\theta_2 &= \tan^{-1}\frac{M_1}{p_z} \\
S_2 &= \sin\theta_2 \\
C_2 &= \cos\theta_2 \\
d_3 &= S_2 M_1 + C_2 p_z \\
M_2 &= -S_1 a_x + C_1 a_y \\
M_3 &= C_1 a_x + S_1 a_y \\
M_4 &= C_2 M_3 - S_2 a_z \\
\theta_4 &= \tan^{-1}\frac{M_2}{M_4}
\end{aligned}
\qquad
\begin{aligned}
S_4 &= \sin\theta_4 \\
C_4 &= \cos\theta_4 \\
S_5 &= C_4 M_4 + S_4 M_2 \\
C_5 &= S_2 M_3 + C_2 a_z \\
\theta_5 &= \tan^{-1}\frac{S_5}{C_5} \\
M_5 &= C_1 o_x + S_1 o_y \\
M_6 &= -S_1 o_x + C_1 o_y \\
M_7 &= C_2 M_5 - S_2 o_z \\
M_8 &= C_4 M_7 + S_4 M_6 \\
M_9 &= S_2 M_5 + C_2 o_z \\
S_6 &= -C_5 M_8 + S_5 M_9 \\
C_6 &= -S_4 M_7 + C_4 M_6 \\
\theta_6 &= \tan^{-1}\frac{S_6}{C_6}
\end{aligned}
$$

Fig.4.2.3. Inverse kinematic equations of the Stanford manipulator.

$$
\begin{aligned}
S_1 &= \frac{p_y\sqrt{{p_x}^2+{p_y}^2-{d_2}^2}-p_x d_2}{{p_x}^2+{p_y}^2} \\
C_1 &= \frac{p_x\sqrt{{p_x}^2+{p_y}^2-{d_2}^2}+p_y d_2}{{p_x}^2+{p_y}^2} \\
S_2 &= \frac{M_1}{\sqrt{{p_z}^2+{M_1}^2}}
\end{aligned}
\qquad
\begin{aligned}
C_2 &= \frac{p_z}{\sqrt{{p_z}^2+{M_1}^2}} \\
S_4 &= \frac{M_2}{\sqrt{{M_4}^2+{M_2}^2}} \\
C_4 &= \frac{M_4}{\sqrt{{M_4}^2+{M_2}^2}}
\end{aligned}
$$

Fig.4.2.4. Solutions of the task S_i and C_i of the inverse kinematic equations.



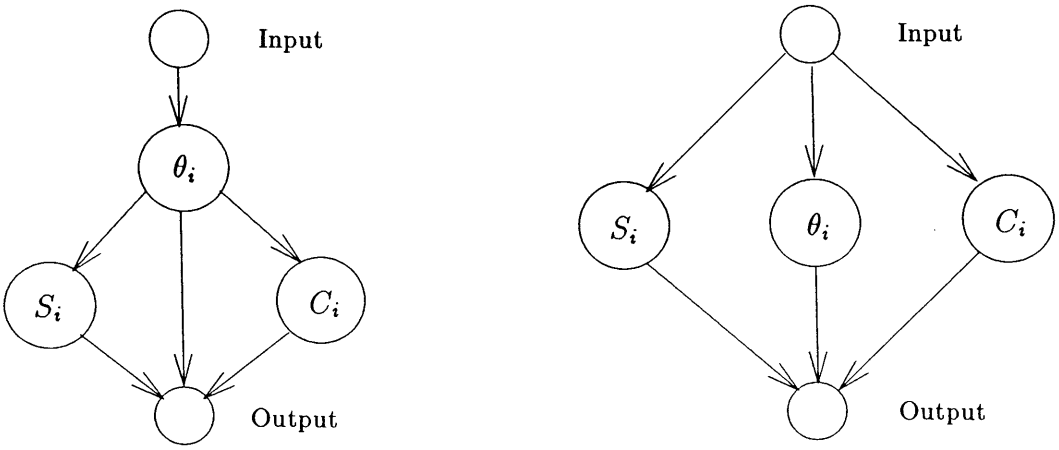
(a) Method I

(b) Method II

Fig.4.2.5. Task graph for the inverse kinematic equations of the Stanford manipulator.

Table 4.2.1. Task number of the inverse kinematic equations.

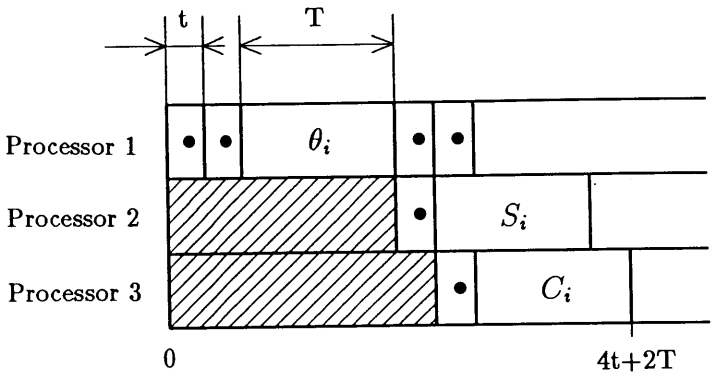
Task number	Task	Task number	Task
1	$\frac{p_y}{p_x}$	14	θ_4
2	$\frac{d_2}{\sqrt{p_x{}^2 + p_y{}^2 - d_2{}^2}}$	15	S_4
		16	C_4
3	θ_1	17	S_5
4	S_1	18	C_5
5	C_1	19	θ_5
6	M_1	20	M_5
7	θ_2	21	M_6
8	S_2	22	M_7
9	C_2	23	M_8
10	d_3	24	M_9
11	M_2	25	S_6
12	M_3	26	C_6
13	M_4	27	θ_6



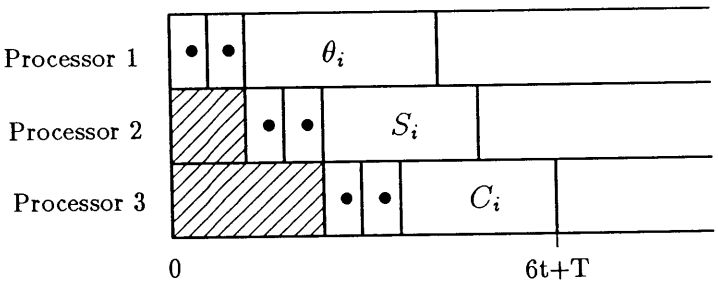
(a) Method I

(b) Method II

Fig.4.2.6. Precedence constraints of the task θ_i, S_i and C_i of the inverse kinematic equations.



(a) Method I



(b) Method II

Fig.4.2.7. Gantt chart of the inverse kinematic equations.

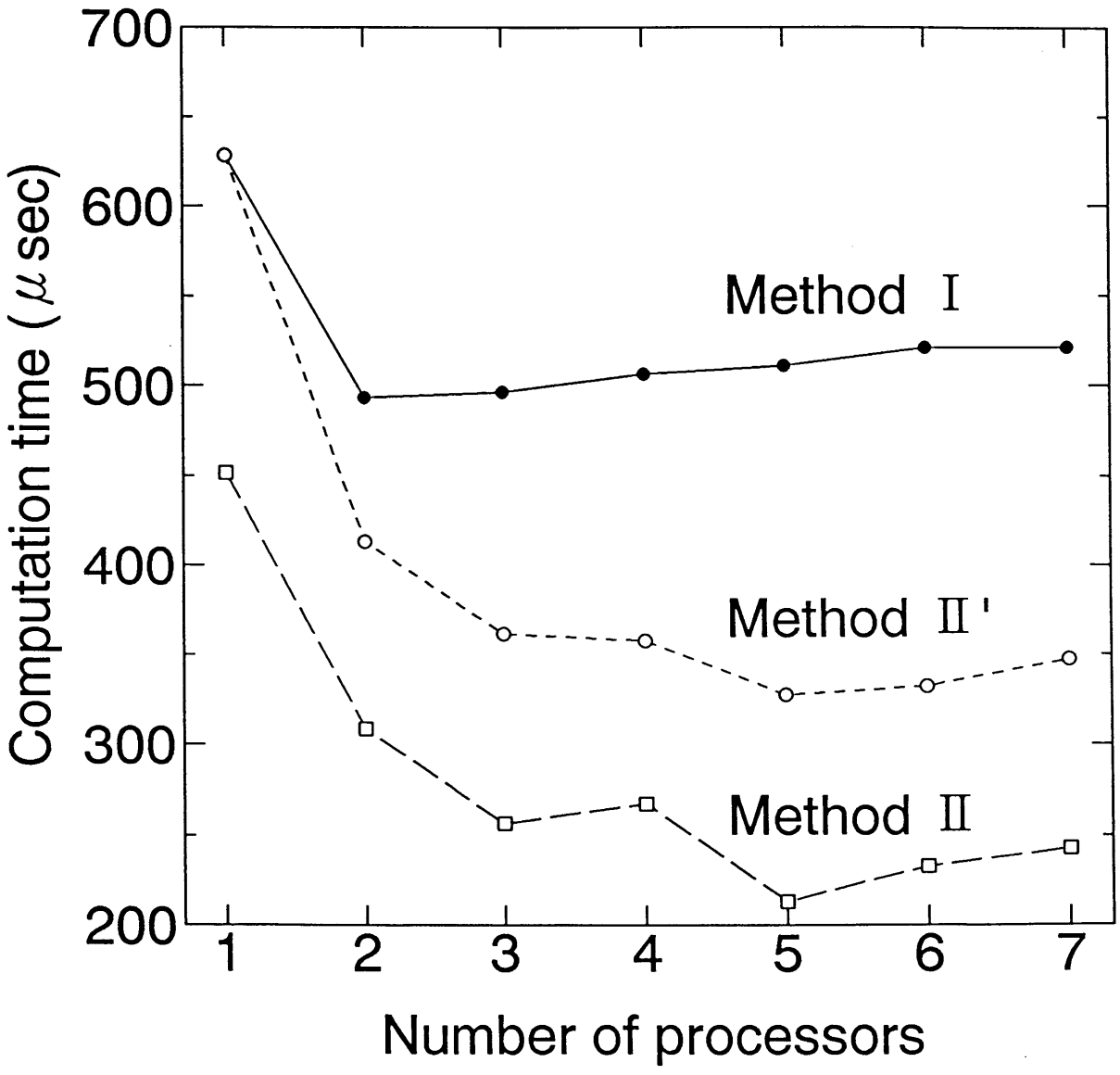


Fig.4.2.8. Computation time of the inverse kinematic equations.

4.3 逆動力学問題とタスク分割法

解法の決定の次の段階として，処理時間の短縮を大きくするためには，第3章で述べたようにタスク分割法を確立する必要がある．一般に，与えられた問題からタスク集合を生成する際，タスク分割を細かくする程タスク集合の並列度は高くなる．しかし，一方，タスクの細分化よってプロセッサ間の通信量が増加すると，並列処理の効率は低下する．従って，全体の処理時間を短縮するためには，タスク集合の並列度とプロセッサ間の通信量の双方を考慮してタスク集合を生成しなければならない．そこで，本節ではロボットマニピュレータの逆動力学問題に対するタスク分割法を提案し，シミュレーションによってその有効性を示す．

4.3.1 ロボットの逆動力学問題

逆動力学問題とは，ロボットマニピュレータの各関節の変位，速度，加速度を入力とし，その運動を実現するための，各関節の駆動トルクを出力として求める問題である．Fig.4.3.1 に Newton-Euler 法によるロボットマニピュレータの逆動力学方程式 [9, 10] を示す．ここで，添字 i（$i = 1 \sim n$; n : マニピュレータの軸数）はリンク i または関節 i に関する変数であることを示し，R は回転関節，P は直動関節を表す．計算の手順としては，マニピュレータの台座側から手先のリンクに向かって，順に運動学計算を行った後，手先側から台座側に向かって順に動力学計算を行う．式中の各変数はそれぞれ以下の意味を持つ．

$\boldsymbol{q}_i$: 第 i 関節の変位

$\dot{\boldsymbol{q}}_i$: 第 i 関節の速度

$\ddot{\boldsymbol{q}}_i$: 第 i 関節の目標加速度

${}^i\boldsymbol{\omega}_i$: 第 i リンクの基準座標系からみた回転角速度

${}^i\dot{\boldsymbol{\omega}}_i$: 第 i リンクの基準座標系からみた回転角加速度

${}^i\ddot{\boldsymbol{p}}_i$: 第 i リンクの基準座標系からみた並進加速度

${}^i\ddot{\boldsymbol{s}}_i$: 第 i リンクの基準座標系からみた第 i リンクの質量中心の並進加速度

${}^i\hat{\boldsymbol{f}}_i$: 外部から第 i リンクの質量中心に加えられるべき並進力

${}^i\hat{\boldsymbol{n}}_i$: 外部から第 i リンクの質量中心に加えられるべき回転モーメント

${}^i\boldsymbol{f}_i$: 第 i 関節において加えられるべき並進力

$^{i}n_i$ ： 第 i 関節において加えられるべき回転モーメント

$^{n+1}f_{n+1}$ ： 把握対象物などから手先に加えられる並進力

$^{n+1}n_{n+1}$ ： 把握対象物などから手先に加えられる回転モーメント

τ_i ： 第 i 関節において加えられるべき駆動力

なお，対象とするマニピュレータは前節と同じ6自由度スタンフォード・マニピュレータであるが，以下に述べる手法は一般の多関節型マニピュレータに適用できる．

本節では，ロボットマニピュレータの逆動力学方程式に関して，各関節単位 [11, 12, 13]ではなく，より汎用的なタスク分割及びスケジューリングを行うため，第3章で述べたスケジューリングアルゴリズムを適用する．

4.3.2 並列処理におけるタスク分割法

一般的なタスク分割は，コマンド，演算，方程式等の単位で行われ，通信に要する時間に対する比率によりおおまかにタスク粒度（タスクの大きさ）が選択されているが，それのみでは各問題に固有の複雑度（並列度，先行関係等）が加味されていないため，十分とはいえない．

本論文で提案する方法では，まずコマンドレベル，演算レベル，方程式レベル等の一般的なタスク分割を行った後，さらにタスクの再合成，再分割を行うという方法で，より有効なタスク分割を行う．

本節では，対象とするプロセッサの計算時間・通信時間比及び逆動力学方程式の並列度から，方程式レベルでの分割をタスク分割の第1段階とし，分割されたタスクの集合をタスク集合 A と呼ぶ．第2段階としては，タスク集合 A において，以下の各ステップでタスクの再合成，再分割等を行うことによって，より一層の高速化を図る．これらのタスク集合を，それぞれタスク集合 B, C, D と呼ぶ．

1) タスクの合成

小さい（処理時間の短い）タスクどうしを合成する．ただし，合成に際しては以下の条件に適合するタスクを選ぶ．

- 合成する各タスクの処理時間が短い
- 合成する各タスクの先行タスクのうちで共通するタスクの割合が大きい
- 合成する各タスクの後続タスクのうちで共通するタスクの割合が大きい
- 合成による先行タスクの処理時間の増加が少ない

$$
{}^{i}\omega_i = \begin{cases} {}^{i-1}R_i{}^T\,{}^{i-1}\omega_{i-1} + e_z\dot{q}_i & \text{(if R)} \\ {}^{i-1}R_i{}^T\,{}^{i-1}\omega_{i-1} & \text{(if P)} \end{cases}
$$

$$
{}^{i}\dot{\omega}_i = \begin{cases} {}^{i-1}R_i{}^T\,{}^{i-1}\dot{\omega}_{i-1} + e_z\ddot{q}_i + ({}^{i-1}R_i{}^T\,{}^{i-1}\omega_{i-1}) \times e_z\dot{q}_i & \text{(if R)} \\ {}^{i-1}R_i{}^T\,{}^{i-1}\dot{\omega}_{i-1} & \text{(if P)} \end{cases}
$$

$$
{}^{i}\ddot{p}_i = \begin{cases} {}^{i-1}R_i{}^T[{}^{i-1}\ddot{p}_{i-1} + {}^{i-1}\dot{\omega}_{i-1} \times {}^{i-1}\hat{p}_i \\ \quad + {}^{i-1}\omega_{i-1} \times ({}^{i-1}\omega_{i-1} \times {}^{i-1}\hat{p}_i)] & \text{(if R)} \\ {}^{i-1}R_i{}^T[{}^{i-1}\ddot{p}_{i-1} + {}^{i-1}\dot{\omega}_{i-1} \times {}^{i-1}\hat{p}_i \\ \quad + {}^{i-1}\omega_{i-1} \times ({}^{i-1}\omega_{i-1} \times {}^{i-1}\hat{p}_i)] \\ \quad + 2({}^{i-1}R_i{}^T\,{}^{i-1}\omega_{i-1}) \times (e_z\dot{q}_i) + e_z\ddot{q}_i & \text{(if P)} \end{cases}
$$

$$
{}^{i}\ddot{s}_i = {}^{i}\ddot{p}_i + {}^{i}\dot{\omega}_i \times {}^{i}\hat{s}_i + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\hat{s}_i)
$$

$$
{}^{i}\hat{f}_i = m_i\,{}^{i}\ddot{s}_i
$$

$$
{}^{i}\hat{n}_i = {}^{i}I_i\,{}^{i}\dot{\omega}_i + {}^{i}\omega_i \times ({}^{i}I_i\,{}^{i}\omega_i)
$$

$$
{}^{i}f_i = {}^{i}R_{i+1}\,{}^{i+1}f_{i+1} + {}^{i}\hat{f}_i
$$

$$
{}^{i}n_i = {}^{i}R_{i+1}\,{}^{i+1}n_{i+1} + {}^{i}\hat{p}_{i+1} \times ({}^{i}R_{i+1}\,{}^{i+1}f_{i+1}) + {}^{i}\hat{n}_i + {}^{i}\hat{s}_i \times {}^{i}\hat{f}_i
$$

$$
\tau_i = \begin{cases} e_z{}^T\,{}^{i}n_i & \text{(if R)} \\ e_z{}^T\,{}^{i}f_i & \text{(if P)} \end{cases}
$$

Fig.4.3.1. Newton-Euler equations.

2) タスクの分割

大きい（処理時間の長い）タスクを分割する．ただし，分割に際しては以下の条件に適合するタスクを選ぶ．

- 分割するタスクの処理時間が長い
- 分割後の各タスクの先行タスクのうちで共通するタスクの割合が小さい
- 分割後の各タスクの後続タスクのうちで共通するタスクの割合が小さい
- 分割による先行タスクの処理時間の減少が多い

3) タスクの全プロセッサでの計算

データ転送の回数を減少させるために，以下の条件に適合するタスクを，予め全プロセッサで計算しておく．

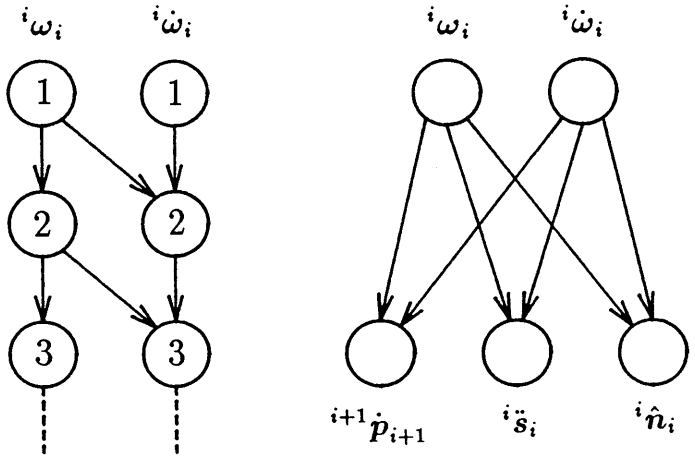
- タスクの処理時間が短い
- 先行タスク数が少ない
- 後続タスク数が多い

本節では，汎用的なタスク分割法よりも更に効果をあげるために，6軸（うち1軸は直動）のスタンフォード・マニピュレータの逆動力学問題という，具体的な問題に対する分割法を採用する．特に2)のタスクの分割は，タスク内の各部分の従属関係が未知であるために，汎用的な方法では困難であるが，このような手法を用いることにより可能になる．

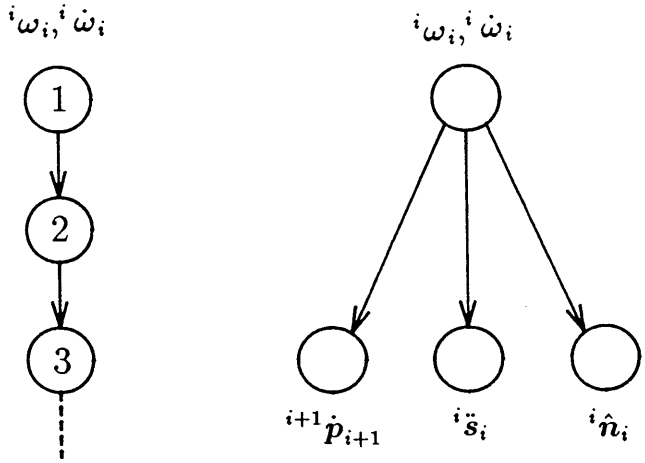
具体的に上記のタスク分割法を逆動力学方程式に適用すると，以下のようになる．

1) タスクの合成

Fig.4.3.1 に示す逆動力学方程式の中で ${}^{i}\omega_i$, ${}^{i}\dot{\omega}_i$ に注目し，${}^{i}\omega_i$, ${}^{i}\dot{\omega}_i$ 間及び他タスクとの関係を Fig.4.3.2(a) に示す．ただし，図中の数字はそれぞれ i の値を示している．${}^{i}\omega_i$, ${}^{i}\dot{\omega}_i$ はこの2種のタスクだけですべてのリンクについて計算でき，他のタスクからのデータを必要としない．また，他のタスクが ${}^{i}\omega_i$, ${}^{i}\dot{\omega}_i$ のデータを必要とするときは，2つのタスクの結果を両方とも参照している．即ち，${}^{i}\omega_i$ と ${}^{i}\dot{\omega}_i$ は互いに結びつきが強く，他のタスクへのデータ依存の無いタスクである．さらに，${}^{i}\omega_i$ は他のタスクに比べてサイズが小さい．そこで，タスク集合 A のうち ${}^{i}\omega_i$ と ${}^{i}\dot{\omega}_i$ を合成し，こ



(a) Task set A



(b) Task set B

Fig.4.3.2. Successors of the task $^i\omega_i$ and $^i\dot{\omega}_i$.

れをタスク集合 B とする．このとき，${}^i\omega_i, {}^i\dot{\omega}_i$ 間及び他タスクとの関係は Fig.4.3.2(b) のようになり，プロセッサ間の通信によるオーバーヘッドが減少する．

2) タスクの分割

次に，タスクサイズの大きいタスク ${}^i n_i$ に着目する．${}^i n_i, {}^i \hat{f}_i, {}^i \hat{n}_i$ 間のデータの流れを Fig.4.3.3(a) に示す．Fig.4.3.3(a) より，タスク ${}^i n_i$ を Fig.4.3.4 のように 2 つのタスクに分割すれば，Fig.4.3.3(b) に示すように ${}^i n_i$ 間の先行制約から u_i の部分が解放されるため，並列度を上げることが可能となる．さらに，サイズの大きいタスク ${}^i \ddot{s}_i$ も Fig.4.3.5 のように分割することによって並列度を上げることが可能となる．タスク集合 B に対して以上のような分割を施したものをタスク集合 C とする．

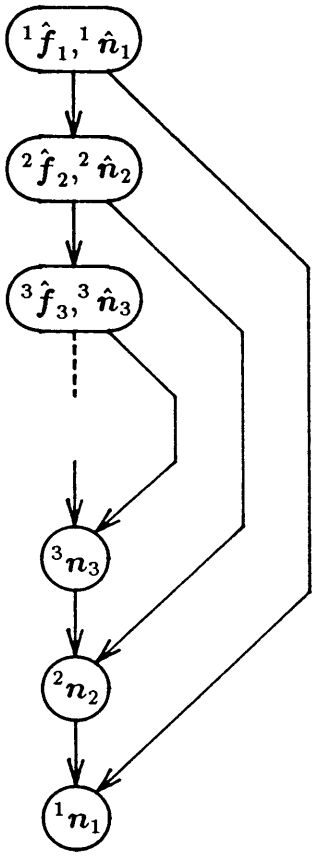
3) タスクの全プロセッサでの計算

${}^i\omega_i, {}^i\dot{\omega}_i$ のデータ（特に i が小さいもの）は他の計算の先行制約となることが多く，それ自身は他の計算の結果を必要とせず，しかもタスクサイズが比較的小さいため，1 つのプロセッサで計算した後に他のプロセッサに転送するよりも，予め全プロセッサで計算しておいて，通信オーバーヘッドを減らした方が有利である．${}^1\omega_1, {}^1\dot{\omega}_1$ は入力として与えられるため，${}^2\omega_2, {}^2\dot{\omega}_2$ を計算するタスクを最初に全プロセッサで行う．前述のタスク集合 C に対して以上のような操作を施したものをタスク集合 D とする．

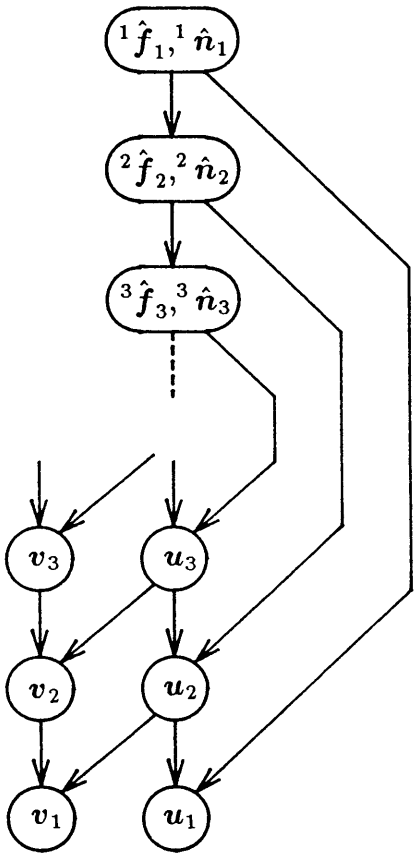
本手法は，スタンフォード・マニピュレータだけではなく，PUMA 型等の他の垂直多関節型マニピュレータや，水平多関節（SCARA）型マニピュレータにもそのまま適用できるため，多くの場合のロボット制御において有効である．また，他の一般的な問題に対しても同様の観点にたったタスク分割法が適用できるため，広範な応用が考えられる．

4.3.3 シミュレーション結果及び考察

Fig.4.3.1 に示した逆動力学問題について前項で述べたタスク集合 A, B, C, D のそれぞれに対して，第 3 章のスケジューリングアルゴリズムを適用した結果を Fig.4.3.6 に示す．図中，横軸は結合されるプロセッサ数を示し，縦軸は処理時間を示す．また，Fig.4.3.7 には Fig.4.3.6 におけるタスク集合 A の処理時間からタスク集合 B, C, D の処理時間を引いた時間差，即ち短縮された時間を示す．図から分かるように，タスク集合 A に比べて ${}^i\omega_i$ と ${}^i\dot{\omega}_i$ を合成したタスク集合 B の方が処理時間が短くなっている．また，プロセッサ数 3 及び 4 の場合を除き，タスク ${}^i n_i$ 及び ${}^i s_i$ を分割したタスク集合 C の方がタスク集合 B に



(a) Task set A, B



(b) Task set C

Fig.4.3.3. Successors of the task $^i\hat{f}_i$ and $^i\hat{n}_i$.

[Task set A, B]

$$
{}^{i}\boldsymbol{n}_i = {}^{i}\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{n}_{i+1} +^{i} \hat{\boldsymbol{p}}_{i+1} \times ({}^{i}\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{f}_{i+1}) +^{i} \hat{\boldsymbol{n}}_i +^{i} \hat{\boldsymbol{s}}_i \times^{i} \hat{\boldsymbol{f}}_i
$$

[Task set C]

$$
\begin{aligned}
\boldsymbol{v}_i &= {}^{i}\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{n}_{i+1} +^{i} \hat{\boldsymbol{p}}_{i+1} \times ({}^{i}\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{f}_{i+1}) \\
\boldsymbol{u}_i &= {}^{i}\hat{\boldsymbol{n}}_i +^{i} \hat{\boldsymbol{s}}_i \times^{i} \hat{\boldsymbol{f}}_i \\
{}^{i}\boldsymbol{n}_i &= \boldsymbol{v}_i + \boldsymbol{u}_i
\end{aligned}
$$

Fig.4.3.4. Division of the task ${}^{i}\boldsymbol{n}_i$.

[Task set A, B]

$$
{}^{i}\ddot{\boldsymbol{s}}_i = {}^{i}\ddot{\boldsymbol{p}}_i +^{i} \dot{\boldsymbol{\omega}}_i \times^{i} \hat{\boldsymbol{s}}_i +^{i} \boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times^{i} \hat{\boldsymbol{s}}_i)
$$

[Task set C]

$$
\begin{aligned}
\boldsymbol{t}_i &= {}^{i}\dot{\boldsymbol{\omega}}_i \times^{i} \hat{\boldsymbol{s}}_i +^{i} \boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times^{i} \hat{\boldsymbol{s}}_i) \\
{}^{i}\ddot{\boldsymbol{s}}_i &= {}^{i}\ddot{\boldsymbol{p}}_i + \boldsymbol{t}_i
\end{aligned}
$$

Fig.4.3.5. Division of the task ${}^{i}\ddot{\boldsymbol{s}}_i$.

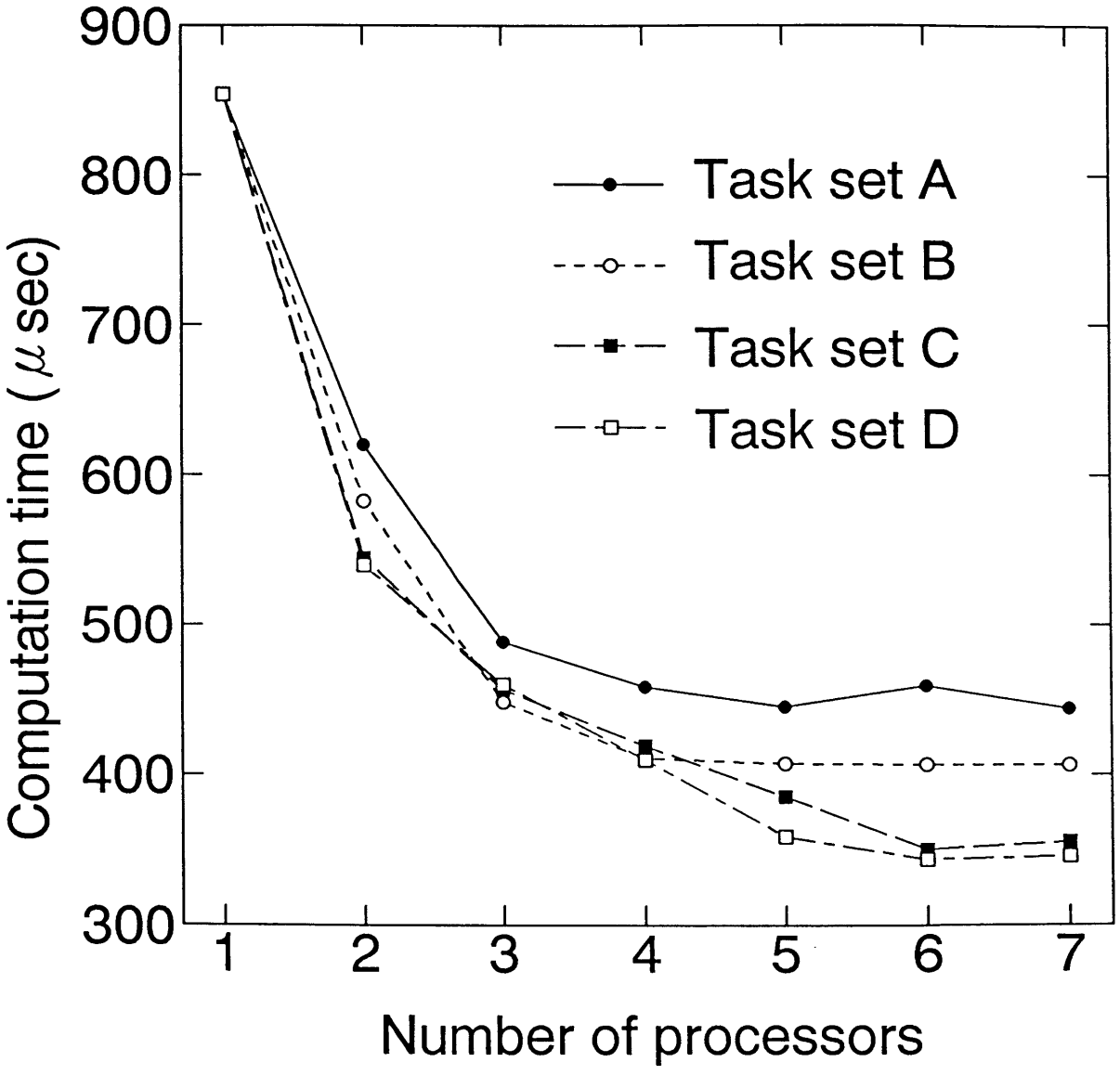


Fig.4.3.6. Computation time of the inverse dynamics equations.

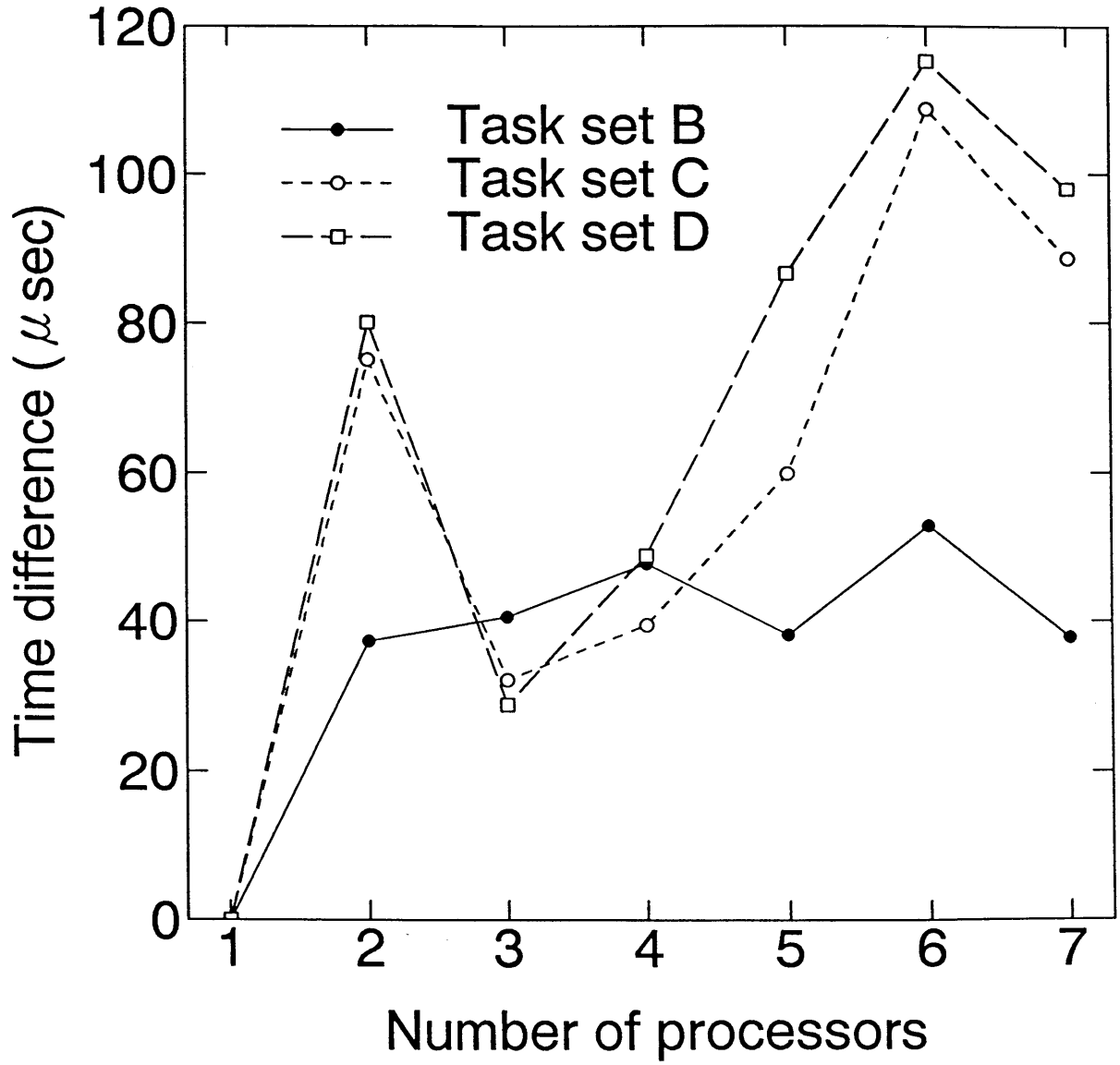


Fig.4.3.7. Time difference between the task set A and other task set in Fig.4.3.6.

比べて処理時間が短い．さらに，プロセッサ数3の場合を除き，${}^2\omega_2, {}^2\dot{\omega}_2$ を計算するタスクを予め全プロセッサで行ったタスク集合Dの方がタスク集合Cに比べて処理時間が短い．そこで，プロセッサ数3の時のみタスク集合Bを用い，他のプロセッサ数の時はタスク集合Dを用いれば，より効果的な並列処理を実現できることが分かる．

最終的なタスク集合Dでは，方程式レベルでのタスク分割を行ったタスク集合Aに比べて処理時間が平均 16.0%，最大 25.1%短縮された．

全般的に 1) のタスクの合成による処理時間の短縮が最もよく表れている．この手法は汎用的なタスク分割法を適用する場合にも採用が容易であることから，最も有効な手法であると考えられる．

全般的に，処理時間の短縮率が低かったのは，マニピュレータのシリアルリンク機構に起因する先行制約が多かったことが原因と考えられる

4.4 まとめ

本章では，マルチプロセッサを用いたロボット制御のための，並列処理における計算の高速化手法について述べ，シミュレーションによりその有効性を示した．まず，ロボットマニピュレータの逆運動学問題を対象として，通信のオーバヘッドの低減により，並列処理時間が短くなる解法を示した．さらに，ロボットマニピュレータの逆動力学問題を対象として，プロセッサ間の通信量及び通信のオーバヘッドを考慮し，並列処理時間が短くなるタスク分割ついて述べた．そして，それぞれの問題に対して，第3章で提案したスケジューリングアルゴリズムを適用して，シミュレーションを行った．その結果を以下に示す．

1. スタンフォード・マニピュレータの逆運動学問題に並列処理向きの解法を採用することによって，処理時間が平均 46.9%，最大 58.3% 短縮された．

2. 逆動力学問題に対して，方程式レベルでのタスク分割を再分割することによって，処理時間が平均 16.0%，最大 25.1% 短縮された．

マニピュレータに，センサ情報等に基づいたフレキシブルな動作を行わせるためには，リアルタイムでのセンサ情報処理，逆運動学計算，及び逆動力学計算が必要である．Luh et al. [11]は逆動力学計算の計算時間間隔を，ロボットの固有振動数を考えて 1/60 秒以下にすることが必要であるとした．しかし，センサ情報に応じて軌道を修正しつつ，高速動作を行わせるためには，さらにその時間間隔を短くする必要がある．本章で示された計算時間は，逆運動学計算やその他の処理にかかる時間を含めて，その 1/10 程度になっている．この時間短縮により，センサ情報に基づいた，よりフレキシブルな動作が可能になる．

今後は，センサ情報の多様化により，センサ情報処理にかかる時間も増大することが考えられるため，センサ情報処理と逆運動学，逆動力学計算の分散処理と各要素内での並列処理のバランスを考えて，より効率的なプロセッサの配置を検討する必要がある．

参考文献

[1] 川島伯彦, 末広尚士, 高瀬国克, "作業座標軸サーボ系のインプリメンテーション", 日本ロボット学会誌, Vol.1, No.2, pp.72-74, 1983

[2] 末広尚士, 高瀬国克, "直接計算法による作業座標サーボに基づくマニピュレーションシステム", 日本ロボット学会誌, Vol.3, No.2, pp.95-105, 1985

[3] 吉川恒夫, 中村仁彦, 横小路泰義, "DSP による多関節マニピュレータの動的制御のための実時間計算", 日本ロボット学会誌, Vol.6, No.3, pp.175-183, 1988

[4] 亀山充隆, 江上秀樹, 樋口龍雄, "ロボット制御用超高速座標逆変換プロセッサの構成", 日本ロボット学会誌, Vol.6, No.1, pp.3-13, 1988

[5] 亀山充隆, 樋口龍雄, "LSI 向きロボット制御用プロセッサ", 計測と制御, Vol.25, No.1, pp.30-36, 1986

[6] "Transputer Reference Manual", Inmos, Oct.1986.

[7] 北川秀夫, 包原孝英, 長南功男, 松田文夫, "タスクの再分割による多関節型マニピュレータの逆動力学並列計算の高速化", 日本ロボット学会誌, Vol.11, No.5, pp.710-716, 1993

[8] R.P.Paul, 吉川恒夫 訳, "ロボットマニピュレータ", コロナ社, 1984

[9] 吉川恒夫, "ロボット制御基礎論", コロナ社, 1988

[10] J.Y.S.Luh, M.W.Walker and R.P.C.Paul, "On-line Computational Scheme for Mechanical Manipulators", *ASME Trans. J. Dynamic Syst.,Measurement and Contr.*, Vol.102, pp.69-76, June 1980.

[11] J.Y.S.Luh and C.S.Lin, "Scheduling of Parallel Computer for a Computer-Controlled Mechanical Manipulator", *IEEE Trans. Syst.,Man and Cybern.*, Vol.SMC-12, No.2, pp.214-234, March 1982.

[12] E.E.Binder and J.H.Herzog, "Distributed Computer Architecture and Fast Parallel Algorithm in Real-Time Robot Control", *IEEE Trans. Syst.,Man and Cybern.*, Vol.SMC-16, pp.543-549, 1986.

[13] 橋本亮一, "逆ダイナミクスの新しい並列計算法", 計測と制御, Vol.30, No.5, pp.406-411, 1991

第5章　結論

5.1　要約

本論文では，ロボットにおける情報処理をテーマとして，ロボットのためのスリップセンサからマニピュレータまでの信号処理，及びその並列演算法について述べた．ロボットの高度な知能化のためには，センサ，コンピュータ，アクチュエータの各要素技術の高性能化だけではなく，アルゴリズムの開発等による情報処理能力の向上が重要である．

センサに関しては，センサ単体のハードウェアの性能を上げるだけではなく，複数のセンサと情報処理によって能力を向上させることが有効である．そこで，効果器センサの一例である距離／スリップセンサの必要性を示した後，距離センサへの拡張が可能なスリップセンサの開発について述べた．本センサは，単体で距離とすべりの2種類の計測が可能であるため，省スペースの観点からも有効である．

次に，センサの複合化，高性能化のためには，演算量が増加する傾向にあるということについて述べ，その演算を高速に行うためのマルチプロセッサ・スケジューリングアルゴリズムを示した．本手法は，プロセッサ間の通信時間をパラメータに取り入れているため，より実システムに近いスケジューリングが可能である．

さらに，このスケジューリングアルゴリズムをマニピュレータの逆運動学，及び逆動力学計算に適用するとともに，マルチプロセッサ並列処理において必要となる，並列処理向き解法の決定，及びタスク分割法の開発を行った．

本研究によって得られた結果を以下に要約する．

第2章では，ロボット用センサの1つとして，把握制御を効果的に行うためのスリップセンサの開発を行った．以下にその要約を示す．

1. 格子像を物体表面上に投影したときの散乱光信号の変化から，把握物体のすべりを非接触に検出する格子像投影型スリップセンサを開発した．

2. このスリップセンサは鏡面，黒体面を除くあらゆる物質に適応可能であることを明らかにした．

3. 自己相関関数を用いた信号処理手法を示し，必要となるサンプル数についての検討を行った．

4. 半導体レーザを用い，受光素子と一体化することによって，ロボットハンドに装着可能な小型スリップセンサを開発した．

5. 格子像投影型スリップセンサを改良して，物体形状の変化に対応可能な二光束干渉型スリップセンサを開発した．なお，このセンサは単体ですべり速度と物体までの距離の両方を検出できる距離／スリップセンサへの拡張が可能である．

このようなロボット用センサを実用化するための条件として，実時間性があげられる．特に，広い測定範囲，高分解能を要求する場合，あるいは複数のセンサ信号を処理する場合に計算量が増加し，リアルタイム制御が困難になる傾向がある．

第3章では，第2章で述べたセンサ信号処理の高速演算法として，マルチプロセッサによる並列処理手法を開発した．以下にその要約を示す．

1. 高速演算処理のために，プロセッサ間の通信時間を考慮に入れた，マルチプロセッサ・スケジューリングアルゴリズムを提案した．通信時間をパラメータとして考慮に入れているため，より実システムに即したスケジューリングが可能である．なお，本アルゴリズムでは，準最適解法であるリストスケジューリングを基調としていて，組み合わせ計算を行う必要がないため，スケジューリングに要する時間も短くなる．

2. ターゲットとして，並列処理専用プロセッサであるトランスピュータを想定し，シミュレーションによってその有効性を示した．本アルゴリズムを採用することによって，通信時間を考慮しないリストスケジューリングである CP/MISF 法に対して，処理時間が平均 5.87% 短縮されることを明らかにした．

第4章では，第3章で述べたマルチプロセッサスケジューリングアルゴリズムをロボットマニピュレータに適用した．以下にその要約を示す．

1. マニピュレータの逆運動学問題を対象として，プロセッサ間の通信量及び通信オーバヘッドを考慮して，並列処理時間を短縮する解法を示した．

2. スタンフォード・マニピュレータの逆運動学問題に，並列処理向きの解法を採用して，第3章で提案したスケジューリングアルゴリズムによってシミュレーションを行い，処理時間が平均 46.9%，最大 58.3% 短縮されることを明らかにした．

3. マニピュレータの逆動力学問題を対象として，プロセッサ間の通信量及び通信オーバヘッドを考慮し，方程式レベルで分割したタスクを再分割することによって，並列処理時間を短縮するタスク分割法を示した．

4. スタンフォード・マニピュレータの逆動力学問題にタスク再分割法を採用して，第3章で提案したスケジューリングアルゴリズムによってシミュレーションを行い，処理時間が平均 16.0%，最大 25.1% 短縮されることを明らかにした．

5.2 今後の課題と展望

本論文の各章で提案した手法について，今後の課題と展望を以下に述べる．

第2章で述べたスリップセンサを実用化するにあたっては，リアルタイムですべりを検出し，把握力のフィードバックを行う必要がある．そのためには，DSP などの高速演算プロセッサによる計算，またはマルチプロセッサによる並列処理が必要となる．後者の一例であるトランスピュータによる並列処理については，第3章で記述している．

物体の位置・姿勢検出とすべり検出を個別のセンサで行った場合，センサシステムが大がかりなものとなる．また，ハンドに装着できるセンサの大きさ，個数には制限があるため，装着することができないといった問題も起こりうる．このため，物体の位置・姿勢検出とすべり検出を共通のセンサで行うことが要求される．2.5 節で述べた二光束干渉型スリップセンサは，そのままで距離センサとして使用することも可能であるので，単体での距離／スリップセンサとしての利用が可能であり，安定した未知物体把握が期待される．

把握制御を行う場合，まず本センサを距離センサとして動作させる．未知対象物体の回りを，本センサを装着したロボットハンドを回転させることによって，物体の輪郭座標を得て，その結果から最適把握位置を求める．次に，本センサをスリップセンサとして動作させる．物体を持ち上げる時の把握力が不足した際に起こるすべりを検出して，把握力にフィードバックさせることにより，最適把握力を求める．

位置検出よりもすべり速度の検出精度を優先する場合は，2枚のレンズを使用することによって距離によらず一定間隔の（光軸方向の変位によって間隔の変わらない）明瞭な干渉縞を作ることも可能である．その応用例としては，2次元の速度検出があげられる．本スリップセンサでは，対象物体に投影した干渉縞に対して斜め方向に物体が移動した場合，干渉縞に垂直な方向の速度成分のみが信号として得られるため，干渉縞の方向をずらした2台のセンサを用いることにより，2台のセンサの出力の速度ベクトルを合成して，2次元の速度検出を行うことが可能となる．また，本センサの性質上，干渉縞の水平方向付近の速度検

出は困難であるが，方向を調整した（例えば 60° ずつずらした）3 台のセンサを設置することにより，常に信号の得易い出力を選択することが可能になる．この応用方法は移動ロボットのデッドレコニング（エンコーダ等の内界センサからの情報のみを用いて行う自己位置計測）等に特に有効である．

第 3 章ではマルチプロセッサシステムの結合方式として完全網を用いたが，異なる結合方式を用いた場合のスケジューリングアルゴリズムの修正，及びその評価も必要である．非閉塞網等のプロセッサ間距離（スイッチ段数）が一定の結合網に対しては，本論文で提案した手法がそのまま適用可能である．また，2 進 n-キューブ網等のプロセッサ間距離が静的に決定している回路網に対しても，各プロセッサ間距離を評価関数に入れることにより適用可能である．

第 4 章で述べた方法を実用化するための課題として以下の点があげられる．

まず，並列処理向きの逆動力学問題の解法の開発を行う．そして，実際にマルチプロセッサ・ロボットコントローラに適用して，有効性を実証する．また，第 4 章で定性的な評価に基づいて行ったタスクグラフの生成を，定量的な評価関数に基づいて自動的に行うタスク生成アルゴリズムの開発を行うことも必要である．その方策としては，予め小さめに分割しておいたタスクどうしを処理時間，後続タスクなどを考慮して合成する方法が有効である．

本研究で取り上げたロボットにおける情報処理全般について，今後の展望を以下に述べる．

未知の環境あるいは動的に変化する環境において，ロボットが自律的に行動するためには，複数のセンサからの情報を利用した環境認識が必要となる．特に環境がより複雑になるに従って，あるいは高速性や信頼性がより要求されるようになるに従って，多数のセンサからの情報を利用するケースが現れる．それに伴い，並列高速演算及びセンサデータ間の融合アルゴリズムの開発が必要となる．また，それと同時に，センサをより集積化するために，マイクロセンサの開発やセンサの多機能化も要求される．

このようなマルチセンサ情報処理においては動的なタスクが生成されるため，ロボットに搭載されるコンピュータには，外界の状況に対応するための実時間性が要求される．一方，複数のロボットが分散的に通信，協調しながら行動する場合も多くなるため，モジュール性を持ち，かつプロセッサ間の通信能力も高い分散型コンピュータ・アーキテクチャの開発も重要である．

謝　　辞

本論文をまとめるに当たりご指導を賜りました名古屋大学工学部電子情報学科 内川嘉樹教授に謹んで感謝の意を表します.

本論文の作成に当たりご教示を賜りました名古屋大学工学部電子情報学科 後藤俊夫教授ならびに名古屋大学工学部電気工学科 大熊繁教授に深く感謝の意を表します.

本研究の端緒をお作り下さり，ご指導を賜りました服部秀三 名古屋大学名誉教授に謹んで感謝の意を表します.

本研究の遂行に当たり終始ご援助とご指導を賜りました岐阜工業高等専門学校電子制御工学科 松田文夫教授に深く感謝の意を表します.

本研究の遂行に当たりご支援とご助言を頂きました名古屋大学工学部電子情報工学科 伊藤昌文助手，多くのご協力を頂きました株式会社中埜酢店 森元裕志氏，名古屋大学工学部電子機械工学科旧服部研究室の黒田敏秋氏，包原孝英氏，富永英嗣氏をはじめとする卒業生諸氏に深く感謝の意を表します.

最後に，本論文をまとめるに当たりご支援下さいました岐阜工業高等専門学校電子制御工学科 長南功男助教授ほか電子制御工学科教職員各位に厚く御礼申し上げます.

研 究 業 績

	論文題目	公表の方法及び時期	著者
I	学会誌等		
1.	物体散乱光信号の自己相関処理を用いたすべり検出法	日本ロボット学会誌, 4 巻, pp.602-606 (1986)	松田文夫 森元裕志 北川秀夫 服部秀三 上田 實
2.	Nontactile Slip Sensor Using Method of Auto Correlation with Reflected Light Signal from Object.	Advanced Robotics, vol.2, pp.361-368 (1988)	F.Matsuda H.Morimoto H.Kitagawa S.Hattori M.Ueda
3.	離散直交座標における Arctangent の連続算出法	計測自動制御学会論文集, 25 巻, pp.723-725 (1989)	松田文夫 白谷晴永 山川陽一 伊藤昌文 北川秀夫 服部秀三
4.	通信時間を考慮に入れたマルチプロセッサスケジューリングアルゴリズム	電子情報通信学会論文誌, J73-D-I 巻, pp.812-817 (1990)	北川秀夫 松田文夫 内川嘉樹 服部秀三
5.	格子像投影型非接触スリップセンサの開発	日本ロボット学会誌, 9 巻, pp.466-470 (1991)	黒田敏秋 北川秀夫 松田文夫 内川嘉樹 服部秀三
6.	Multiprocessor Scheduling Algorithm with Communication Time.	Systems & Computers in Japan, vol.22, pp.14-20 (1991)	H.Kitagawa F.Matsuda Y.Uchikawa S.Hattori

	論文題目	公表の方法及び時期	著者
7.	タスクの再分割による多関節型マニピュレータの逆動力学並列計算の高速化	日本ロボット学会誌, 11 巻, pp.710-716 (1993)	北川秀夫 包原孝英 長南功男 松田文夫

II 国際会議

1.	Nontactile Slip Sensor Using Method of Auto Correlation with Signals of Reflecting Light.	Proc. of the 16th Int. Symp. on Industrial Robots, pp.591-598 (1986)	F.Matsuda H.Morimoto H.Kitagawa S.Hattori M.Ueda
2.	Control of an Industrial Robot with Nontactile Slip Sensor Using Auto Correlation Proccessed Laser Scattering Signal.	Proc. of the 17th Int. Symp. on Industrial Robots, pp.6·1-6·13 (1987)	F.Matsuda H.Morimoto H.Kitagawa S.Hattori M.Ueda
3.	Scheduling of parallel processing including communication time.	Proc. of the Autotech Asia '89, pp.62-79 (1989)	H.Kitagawa M.Ito F.Matsuda Y.Uchikawa S.Hattori