技術論文

並列処理型センサベースト マニピュレーションシステム：匠

北垣高成*1 末広尚士*2 小笠原司*1 劉雲輝*3

Sensor Based Parallel Processing Manipulation System : TAKUMI

Kosei Kitagaki*1, Takashi Suehiro*2, Tsukasa Ogasawara*1 *and* Yun-Hui Liu*3

This paper presents a novel parallel processing manipulation system, developed to provide extensibility and capability of real-time response. The system has been organized based on multi-agent using many transputers. The advantages of the system are 1) capability to integrate manipulator controls from higher intelligent levels through lower servoing levels in which real-time response is strongly desired, and 2) flexibility to accumulate various methods and to utilize them any time without bad effects to the system. First, in this paper, an important issue to be considered for system design is introduced as a result of a classification of data processing from sensors to actuators. Next, guides to design the system are presented based on the issue. Third, system hardware and software are described in detail. Finally, a prototype system for direct drive manipulator, ETA 3 s, is shown. The effectiveness of the system is demonstrated with two extended systems for different studies of adaptive control and pseudo contact position monitoring, developed with low cost.

Key Words: Manipulation System, Parallel Processing, Multi-agent, Sensor Fusion, Transputer

1. はじめに

ロボットに器用な作業を行わせるには，作業状態を絶えず監視し，様々な観点から作業状況を判断してロボットを動作させる必要がある．そのためには，センサやセンサ情報処理手法のインプリメント，および蓄積が容易であり，さらに，蓄積された手法を統合できるようなシステムが望まれる[1]．

従来の産業用マニピュレータの専用コントローラは，位置決め制御用に特化されていることが多く，新たなセンサや機能をインプリメントするための拡張性や，センサ情報処理のリアルタイム性が十分に考慮されているとは言えない．オープンシステムを有する産業用マニピュレータコントローラは少ない[2]．

制御アルゴリズムの研究開発用としていくつかのシステムが開発されている．筆者らは研究成果の蓄積が可能なサーボレベルの制御実験用システム ARS/A を開発している[3]．下倉らは複数のプロセッサを用いた接触状態変化検出機能を有するマニピュレーションシステム[4]を提案している．これらは上位レベルとの結合に関して具体的には触れていない．

拡張性を確保するには，オープンシステムであることが必要とされる．オープンシステムは，専用でないコンピュータ，標準的なオペレーティングシステム，およびプログラム言語を用い，標準バス上のハードウェアに基づいて構築され，そして，ネットワーク利用可能であることが望ましい[5]．しかし，汎用のハードウェアだけでは，所要の機能実現，とくにリアルタイム性確保がむずかしい場合もあるため，ロボットの逆運動学などの計算に適した再構成可能並列プロセッサ[6]や，センサフュージョンを実現するための実時間用並列計算機 CODA[7]などが提案されている．また，ソフトウェアも同様であり，所要の機能を実現するために全体的にあるいは部分的に特化されたソフトウェアが開発されている．例えば，CHIMERA は基本的に UNIX ライクであるが，リアルタイム性を確保するためにカーネルは別途開発されている[8]．

従来より，筆者らはスキルに基づくマニピュレーションシステム[9]を用いて様々な研究を進めてきた[10]～[12]．同システムはネットワークを介することで上位システムと容易に結合可能であるが，新たなセンサデータをサーボレベルでフィードバック可能なほどのリアルタイム性は有していなかった．そこ

原稿受付 1995年11月10日
*1電子技術総合研究所
*2新情報処理開発機構組合
*3香港中文大學
*1Electrotechnical Laboratory
*2Real World Computing Partnership
*3The Chinese University of Hong Kong

で，上位から下位までの拡張性とリアルタイム性を同時に満たすシステムの実現を目指し，並列処理型センサベーストマニピュレーションシステムを新たに構築してきた[13]～[19]．本システムはトランスピュータを用いたエージェントベースのシステムである．トランスピュータを用いたシステムとしては，ロボット制御のための運動学や逆動力学計算[20][21]，また，適応制御[22][23]，計算トルク制御[24]などの制御アルゴリズムを細粒度で並列分散化することで高速化を図る手法や自動分散化手法[25]が提案されているが，これらはそれ自体で閉じており，他の機能との接続法については言及されていない．

本論文の構成は以下の通りである．まず，センサからアクチュエータまでのデータ処理を分類し，それぞれの入出力関係を考察することにより，システムの拡張性を確保するために考慮すべき点を2章に示し，それに基づいた設計指針を3章に示す．つぎに，本システムのハードウェア，ソフトウェアについて，4章，5章に示す．6章では，ダイレクトドライブマニピュレータETA3s用プロトタイプシステムについて示し，本システムの特徴を生かして行われた制御実験のための拡張システム例を示す．7章はまとめである．

2. データ処理モデル

プロセス間の結合の度合いを確認し，処理の分散並列化を検討するための，センサ信号や目標値の入力からモータなどアクチュエータへの出力へのデータ処理の流れのモデルを示す．

システムへの入力情報はいくつかのプロセスを経て出力される．ここではプロセスを5つに分類し，それらの入出力の特徴について述べる．**Fig. 1** に提案するモデルを示す．図中，データは左から右へ，上から下へと流れる．ただし，ここではセンシングデータの流れに注目するため，各プロセスにおけるパラメタの変更やプロセス自体の制御を行うためのデータの流れは省略した．それぞれのプロセスを以下に示す．

（1） 入力プロセス：I/Oを介してエンコーダ，ポテンショメータや力覚センサ，視覚センサなどの内界，外界センサからのセンシングデータを得る．システムに対して与えられる目標値も含まれる．この段階ではフィルタリングなどの，入力データの情報量が減少するような処理は行わない．

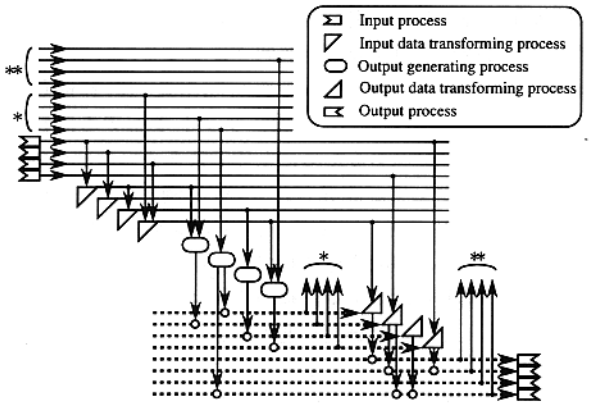


Fig. 1 Process flow model

（2） 入力情報変換プロセス：入力プロセスからのデータおよび(4)，(5)のプロセスへのデータ(Fig. 1 中* 印，** 印)を(3)，(4)で利用可能な形式にするための処理を行う．センシングデータのフィルタリングや座標変換などの下位レベルの処理から，接触状態遷移の検出[12]など上位レベルの処理まで含まれる．上位レベルの処理は下位レベル処理結果を用いることができる．ロジカルセンサ[26]はこのレベルで実現することができる．

（3） 出力生成プロセス：(1)，(2)からのデータおよび(4)，(5)のプロセスへのデータ(Fig. 1 中* 印，** 印)をもとに上位レベルの動作決定を行い，結果を出力情報変換プロセスや直接出力プロセスに出力する．スキル[9]はこのレベルのプロセスとして位置付けられる．

（4） 出力情報変換プロセス：(1)～(3)からのデータを前処理し(5)へ出力する．座標変換などのほか，フィードバック制御もこの段階のプロセスに含まれる．

（5） 出力プロセス：(3)，(4)からのデータに演算を施しI/Oを介してモータなどにデータを出力する．

各プロセスは拡張される可能性があるため，インプリメント時にはそれぞれの入出力関係すべてが決定されているとは限らない．例えば，(1)のプロセスへの入力はそれぞれ決められた入力装置(センサ)からのデータ以外入力される可能性はない．一方，出力は(2)～(4)のプロセスからアクセスされるが，(2)～(4)のプロセスは拡張の可能性があるためインプリメント後に新たに生成されるプロセスからアクセスされる可能性がある．まとめると，(1)のプロセスのインプリメント時にその入力先は決定されている，すなわち固定であるが，出力先はすべてが決定されているとは限らないということになる．各プロセスの入出力先がインプリメント時に固定であるかそうでないかをまとめて **Table 1** に示す．拡張性を持たせるために考慮すべき点は，未知の出力先や入力先に対してターンアラウンドタイムを変化させずに結合できるようにあらかじめ準備しておくという点にある．

3. システム設計指針

本システムは，「蓄積されてきた機能のリアルタイム性を損なわずに，機能拡張することのできるシステム」の実現を目指したものである．2章で示した処理モデルに基づき以下のような設計指針がたてられた．

・システム全体に拡張性を持たせる

既存システムに新たなセンサを追加装備する場合を考えてみよう．単にセンサとのインタフェースが準備されているだけで

Table 1 Features of input and output for an each process at its implement

Processes	Input	Output
Input process	Fix	Non-fix
Input data transforming process	Fix	Non-fix
Output generating process	Fix	Fix
Output data transforming process	Non-fix	Fix
Output process	Non-fix	Fix

は十分ではなく，センサデータ処理や処理データを用いた制御則も同時にインプリメント可能でないと，新たなセンサを装備した意味がなくなる．既存の機能のリアルタイム性を失わないようにしつつ，新機能を付化できるためには，システム全体に拡張性が求められる．

・処理モジュールを並列分散化する

とくにセンシングデータはパイプライン処理を行うことの可能なものが多く，並列処理を行うことによりリアルタイム性を高めることができる．また，それまでのシステムの性能を損なうことなく新たなセンサ処理手法等をシステムに組み込むことができる点で有利である．

・モジュール間の結合度の柔軟性を持たせる

単にモジュール化すればよいというものではなく，モジュール間の結合度を十分考慮した上でシステムをモジュール化しないと拡張性，リアルタイム性の低下を招くことがある．リアルタイムモニタリングを実現しようとする場合，センシングデータ取得とその処理，動作の決定までのプロセスの結合は密であることに注意する必要がある．

・内界センサと外界センサの区別をしない

内界センサ情報は位置・力制御などのフィードバック制御に用いられるだけではなく，状況認識などの上位レベルの処理にも用いられる可能性が高い点にも考慮する必要がある．従来のようにサーボシステムだけが内界センサデータを取得している場合には，外界センサのデータはサーボと独立に取得できるにもかかわらず，内界センサのデータのサンプリング時間はサーボに依存してしまい，それより短いサンプリング時間での処理を行うことはできない．

・上位レベルから下位レベルまでの制御を統一的に扱えるようにする

理想的にはすべての制御レベルで同一の実行環境であることが望ましいが，実際的にはそうではないこともある．とくに，下位レベルでは非常に短い制御周期を実現するための例外的なテクニックを用いることがある．したがって，すべてのレベルで完全に統一することはむずかしいが，ある程度の例外処理を認めた上で，できるだけ統一性を保つことができるようにしておく必要がある．

これらの指針を踏まえ，本システムでは，全体の拡張性を確保するために多数台結合させることの容易なメッセージ伝達型プロセッサトランスピュータを用い，処理を機能別にモジュール化しプロセッサに割り当てることにより並列分散化している．Ethernetからメモリ間データ伝送まで広帯域のプロセッサ間通信速度をサポートすることでモジュール間結合度の柔軟性を確保している．センサデータはできるだけ短いサンプリングで不特定の外部からアクセスできるよう接続システムを準備している．また，モジュールをエージェントとして統一的に扱いソフトウェアシステムを構築することで統一性を確保している．

4. ハードウェア

VME等の汎用バスによる共有メモリを介したデータの通信を行う方法はよく用いられるが，多数の処理を並列に行わせようとするとバスの通信能力がネックとなる場合がある[27]．各々のデータに独立したバスを割り当てることができれば理想であるが現実的ではない．そこで，本システムではメッセージ伝達型プロセッサを用いた．メッセージ伝達型プロセッサはバス結合型と比較し，プロセッサ数を容易に増やすことができるため大規模システムの構築が比較的容易であること，また，データを流すための通信ラインを独立に持たせることが可能であることなどの点で有利と考えた．

メッセージ伝達型プロセッサによるシステム構築において，考慮すべき点は各プロセッサ間のデータ通信速度である．より計算能力の高いプロセッサが次々と開発され市場に出回っているが，オーバーヘッドも含めたプロセッサ間通信速度の点で優れているトランスピュータを本システムでは採用している．

システムの統一性という観点からは，同一プロセッサでシステム全体を構築することが望ましいが，従来より蓄積されている汎用ワークステーションやパーソナルコンピュータの膨大なソフトウェアを生かせなくなることがある．そこで，本システムでは，それら汎用機を単なるプログラム開発機としてではなく，ロボット制御のための処理モジュールと位置づけ，用いている．

4.1 通信速度による階層構造

所要の機能を実現するためには，各モジュール間のデータ通信時間を考慮する必要がある．本システムでは，様々な時間的要求に応じることができるよう **Fig. 2** に示されるような階層構造となっている．

・I層：Ethernetに代表される汎用ネットワークやバスを用いた階層．

・II層：トランスピュータのシリアルリンクによる通信を用いた階層．

・III層：高速データ伝送ボード(4.3節参照)によるメモリ間通信を用いた階層．

I→IIIの順で通信速度が速くなるが，逆に汎用性が減少するためプログラムが特化する．このトレードオフを考慮しつつ，ユーザは処理時間の要求に応じて，その通信手段を自由に選択することができる．なお，それぞれの階層はトランスピュータを介してオーバーラップしているため階層間で通信の遅れはほ

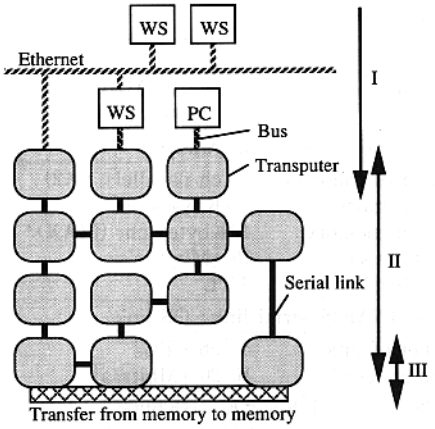


Fig. 2 Hierarchic structure with respect to data transfer rate

とんど生じない.

4.2 高速インタフェース

ロボットを制御するには，センサ，アクチュエータとのインタフェースの高速性が要求される．本システムでは，マニピュレータ等とのインタフェースとして，高速性を要求されない一部を除いて，メモリマップドI/Oを用いることで，プロセッサからの高速アクセスを可能とした．また，各インタフェースは機能別に独立させているため，センサやアクチュエータを追加装備しても，干渉は生じない.

4.3 高速データ伝送ボード

トランスピュータネットワークを用いることでより汎用的かつ高速にデータを伝送することができるという長所を持つ．しかし，トランスピュータの通信用リンクが4本という制約のため，他のプロセッサ上のプロセスに影響を及ぼさないようにするためにはルータとしてのプロセッサが大量に必要となること，また，データが複数のプロセッサを通過するため通信時間の遅れが大きくなることなどの問題が生じる．また，0.1〜1[ms]程度の短い周期で更新されるセンサデータを同一周期で処理する場合，そのデータ伝送のために費やされるタスク切替えに伴うオーバーヘッドが大きな割合を占めるようになりネックとなることがある．処理周期の方が長い場合には常にセンサで得られた最新データを利用できるようブロードキャスト機能があれば都合がよいが，トランスピュータのチャネルは一対一に対応しているためセンサデータをたれ流すことは許されない.

本ボードはそのような更新周期の短いセンサデータのリアルタイム並列処理を実現するために開発されたものであり，OS link[28]よりも高速なデータ伝送，データ送受信のためのCPU負担軽減，データのたれ流し，さらにデータアクセスの競合軽減を可能とするものである．前バージョンのボードでの伝送速度は5[Mbytes/s]であったが[15]，今回，改良を加えることにより伝送速度10[Mbytes/s]を実現した．仕様を**Table 2**に示す．本ボードはVMEダブルハイト2スロット幅のサイズであり4枚1組で使用される．それぞれのボード上にはトランスピュータが2個ずつ搭載されておりOS linkに加えメモリを介したデータ通信を行うことができる．ただし，各センサデータ処理はそれぞれ一定の周期で行われることから，データアクセスの競合が予想されるため，ここでは一般的な共有メモリ方式は採用していない.

・データ伝送機構

Fig. 3に示されるように，それぞれのプロセッサには書込領域1つと読込領域7つがあり，書込領域のデータは他の7個のプロセッサの同じアドレスの読込領域に8ビットパラレルのラインを介して1バイトずつ並列に伝送される．伝送機構はCPUとは独立したハードウェアで構成されており同一クロックで動作する．伝送ラインは領域ごとに独立しており，伝送領域量はボードの設定により可変である．データ伝送は連続的に行われており，プロセッサの伝送領域へのアクセス時間だけ伝送は停止する．各プロセッサにおけるタスクとは同期していないことが文献[29]と異なる点である.

・OS linkとの伝送時間の比較

1つのトランスピュータから7つのトランスピュータに対してデータをOS linkのみを用いて伝送した場合，および，本ボードを用いて伝送した場合のメモリアクセス時間も含めた伝送時間を**Fig. 4**に示す．メモリアクセス時間も参考のため示す．これらのデータは実測値をもとに計測のためのオーバーヘッドを考慮して得られた推定値である．本ボードでの通信時間が連続的でないのは，伝送データバイト数の設定が離散的なた

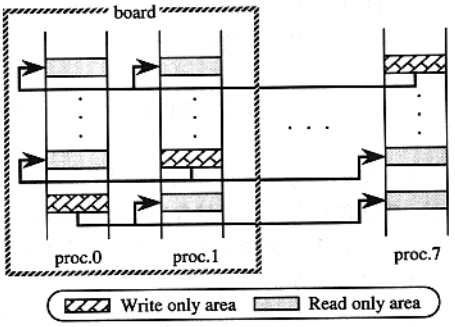


Fig. 3 Mechanism of developed high rate transfer board

Table 2 Specifications of a developed fast transfer rate board

CPU	
Transputer	T 805-30 [MHz] (×2)
Memory	1 [Mbytes/CPU]
Interface (high-speed data transfer)	
Number of channel	8 [ch parallel] (×2)
Total memory	4 [Kbytes/ch]
Transfer memory	4 [Kbytes/ch] (MAX)*
Transfer speed	10 [Mbytes/s]
Signal level	TTL
Interface (INMOS serial link : OS link)	
Number of link	4 [ch/CPU]
Transfer speed	10/20 [Mbits/s]

*32/64/128/256/512/1[K]/2[K]/4[Kbytes]. The simultaneousness of each 32[bytes] can be guaranteed.

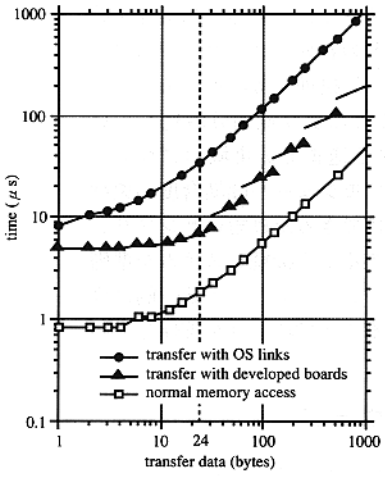


Fig. 4 Transfer bytes v.s. time

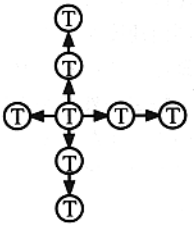


Fig. 5 Transputer connection for comparison

めである．本測定でのOS linkを用いた通信は**Fig. 5**に示されるような結合により行われた．

本ボードを用いたときの伝送時間はOS linkを用いたときと比較して1/5.6〜1/1.4である．伝送データ数が数バイトの場合には速度の差が小さいがこれは最小伝送データが32バイトのためである．力覚データ一組(力とモーメント計6個の要素，1要素あたり浮動小数点4バイトとしたとき)計24バイトデータを他のトランスピュータに送る場合，伝送時間は約1/5であることがFig. 4よりわかる．

・伝送周期とセンサデータ更新周期

ここで考慮されるセンサデータの更新周期は0.1[ms]〜であるのに対し，例えば伝送メモリ量を32バイトに設定した場合には，3.2[μs](=32[bytes]×0.1[μs/byte])周期でメモリ上のデータがすべて伝送されることになり，更新周期に対する伝送周期の割合は小さい．したがって，読込領域のデータはほぼ最新のものとみなすことができる．

5. ソフトウェア

システム全体を統一的にプログラミングすることができれば理想的である．本システムでは2章で示したデータ処理モデルにおけるひとつの機能を持つプロセスをエージェントと呼んでいる．各エージェントには機能処理に加え，受付可能なコマンド(おもに機能処理に対する入出力コマンド)を定義しコーディングしておく．目的のエージェントにコマンドを伝送することで他のエージェントからその機能を利用することが可能となる．システムを構成する各々の機能をこのような形態でエージェント化することで，システム全体を統一的に扱うことができる．

マニピュレータ関節角度取得エージェントを例にとると，機能として一定時間ごとのエンコーダカウント値の読み込みおよび関節角度への変換，コマンドとして関節角度データを返すコマンドを定義しコーディングしておく．その結果，あるエージェントで関節角度データが必要になった場合にはこのエージェントに関節角度要求コマンドを伝送するだけで必要なデータを得ることができるようになる．

エージェント間通信方法として，次節に示すルータを利用した通信とそうでない通信がある．

・ルータを利用した通信

開発されたシンプルなルータソフトウェアを，エージェントとともにプロセッサ上で並列に稼働させることにより行う通信である．本ルータによる通信を利用することにより，並列分散プログラムが汎用的になり，さらに，数[ms]程度の制御ループをも実現することが可能となった．

・ルータを利用しない通信

特定のエージェントどうしをラインを専用化して行う通信で，通信周期が非常に短い場合などに用いられる．例えば，0.5[ms]ごとに更新されるセンサ情報を連続的に通信させる場合に用いる．高速性は満たされるが，乱用するとシステムの透明度が落ちる．

5.1 ルータ

トランスピュータにはシリアルリンクが装備されており，それらを電気的に結合することでハードウェアとしてのトランスピュータネットワークを形成することができる．しかし，そのままではすべてのデータ伝送について通信プログラムを各々開発する必要がありプログラミングが煩雑となる．そこで筆者らは，ネットワークを介したデータ伝送を容易に実現するためにルータソフトウェアを開発した．本ルータはネットワーク構造を考慮して作成されるルーティングテーブルに従い，データを宛先のエージェントに転送する．ルータ間を流れるデータは元のデータに宛先エージェントなどのヘッダを付化した形式(以下NETプロトコルと呼ぶ)である(**Fig. 6**)．なお，本ルータはOccam 2[30]で記述されている．

ルータを用いた通信に要する時間の測定結果を**Fig. 7**に示す．通信時間の変化はデータ長，ホップ数にほぼ比例しており，プログラムのオーバーヘッドをA，ルータでの遅延をB，ルータとエージェント(プロセッサ内部)の通信遅延をC，ルータ間(リンク)の通信遅延をD，データ長(往復のヘッダ+付加データ)をl，ホップ数をnとすると実験での通信時間は，

$$\tau=2A+2(1+n)B+2lC+nlD \qquad (1)$$

と表せる．これに基づき最小二乗近似を行うと，$A=38.76$, $B=45.66$, $C=0.065$, $D=0.801$となった．このとき，残差の標準偏差は6.83であった．実際にはリンクの遅延はボード間通信を行うと大きくなるし，また，同じプロセス内ではプログラムのオーバーヘッドが大きくなる．これらを考慮して近似すると残差の標準偏差はより小さくなるが，遅延時間などのパラメタはそれほど大きく変化しない．

測定結果を基に，データ長をl，ホップ数をnとして片道通信時間を評価すると，

$$\tau=A+(1+n)B+2lC+nlD \qquad (2)$$

と見積もることができる[16]．

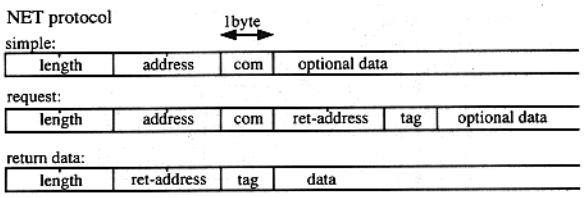


Fig. 6 Typical three types of NET protocol definition

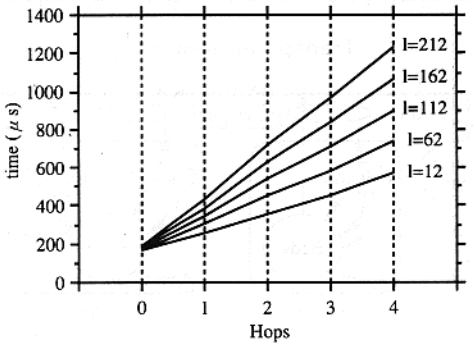


Fig. 7 Hops for transfer v.s. time

5.2 ルータを用いたシステム構成

本ルータを用いたシステム構築の際，以下の点を設計指針とした．

- ネットワーク内を流れるデータはNETプロトコルデータに限定．
- INMOS社供給のホストI/Oライブラリが利用可能．
- iserver(INMOS社供給のサーバ)とそれを利用するエージェントの入出力部のみでiserver用プロトコル(以下，ISERVERプロトコルと呼ぶ)データを使用．
- 任意のエージェントが接続先として任意のiserverを選択可能．

さらに，当所で開発されたオブジェクト指向Lisp：EusLisp[31]との接続を実現するため次に列挙する点も合わせて指針とした．

- EusLispもネットワーク上のエージェントの一つとみなすことが可能．
- 複数のiserverとEusLispが共存可能．

これらの指針により**Fig. 8**に示されるような構成のシステムが開発された．図中，四角で囲まれているのはそこを流れるデータプロトコルを示している．eus-server，iserverはひとつずつのみ描かれているが，ハードウェアで許せば複数個接続することが可能である．また，複数のソケットをオープンできるようeus-serverを改造すれば，そこに複数のEusLisp，iserverを接続することも容易にできる．i.net.serverはプロトコル変換エージェントである．

本ルータで通信を行うエージェントは，大別してFig. 8下部に示されるような3つのタイプに分類することができる．

Type 1はネットワークと直接入出力を行う最も単純な形のエージェントである．

Type 2はHeadとBodyからなるエージェントである．

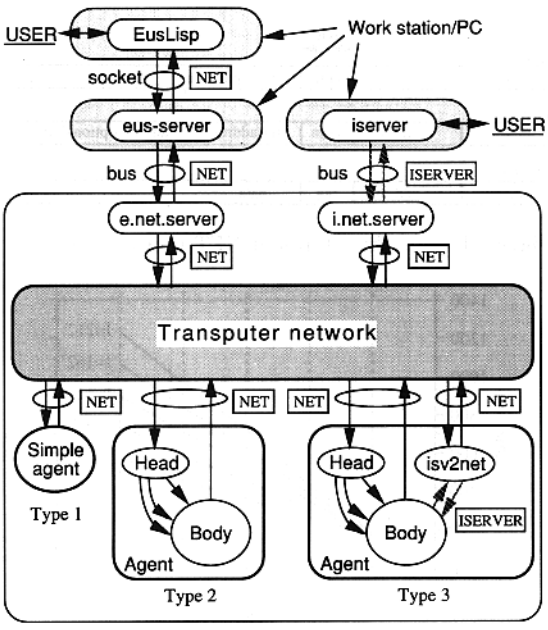


Fig. 8 Multi-agent network with transputers and EusLisp

Bodyではそのエージェント固有のデータ処理を行う．Headではネットワークから送られてくるデータの種類(他のエージェントからのリクエストであるのか，自らが他のエージェントにリクエストしたデータであるのか)を判断し，適切なbodyとのチャネルにデータを流す．HeadとBodyは並列で稼働させる．

Type 3はType 2にさらにisv2netというプロセスが並列で稼働するエージェントである．isv2netではISERVERプロトコルデータをNETプロトコルデータに変換する．このプロセスを用いることにより，INMOSのホストI/Oライブラリを利用することができる．デバッグの際に非常に強力な機能である．

5.3 EusLispの利用

ネットワーク上のエージェントの一つとしてEusLispを扱うことにより，EusLispの持つ豊富な機能を利用しつつ，各エージェントに向けてコマンドを発行することによりロボットを制御することが可能となっている[17]．

EusLispからeus-serverの起動されているホストとそのソケットのポートとを指定してソケットをオープンすることによりトランスピュータネットワークとの通信が可能になる．

- データの変換

トランスピュータネットワーク上では通信効率を上げるため型のないバイナリデータで通信を行っている．そのためデータは送り側，受け側で整合を取って解釈されなくてはならない．バイナリデータからEusLispのデータへの変換については，データの型を表す数値リストに基づいてバイナリデータと数値リストの変換を行う単純なEusLispの関数をC言語で作ってある．現在はバイトデータ，2バイト整数，4バイト整数をEusLispの整数に，4バイト実数をEusLispの実数に対応させた変換をサポートしている．トランスピュータはlittle-endianであり，現在EusLispを走らせているSunのSPARCはbig-endianなのでその変換もここで行っている．これは使用するワークステーションごとに注意してコーディングしなくてはならない部分となる．さらに数値リスト，ベクトル，配列などへの変換の関数はEusLispで記述されている．

- ネットワーク上のエージェントの表現

EusLispの機能を活かしてトランスピュータネット上のエージェントに対応してEusLisp上ではオブジェクトが作られている．各エージェントへの命令はそれぞれオブジェクトのメソッドとしてまとめられている．これによりトランスピュータ上のエージェントに対する命令をEusLispのオブジェクトへの命令として自然な形で書くことができる．またデータの型の指定，返答のあるなしなどのエージェント個別の処理をオブジェクトごとに安全に記述することができる．

その他，ネットワーク全体やトランスピュータプロセッサに対応したオブジェクトも作られていて実際のネットワークと相似な構造がEusLisp上に作られている．現在はプログラマの責任で実際のネットワークとの整合性を保っている．

5.4 開発支援ユーティリティ

これまで述べたようなエージェントベースのシステムを構築する際には，

・プロセッサアドレス，各プロセッサにおけるエージェントのアドレス，および，各エージェントにおけるコマンド番号の管理
・ルーティングテーブルの作成
が必要である．システムが小規模の場合には開発者が手で行うことも可能であるが，大規模になると間違いなく行うことはむずかしくなる．そこで，本システムでは，これらを半自動で行うためのユーティリティが準備されている．

本ユーティリティは，INMOS 社の供給している開発環境において作成する必要のあるコンフィギュレーション記述ファイルで定義されているプロセッサラベル，各プロセッサ間のリンク結合情報等を利用し，さらに，同ファイルのコメント部に記述されたコマンド名とその入出力データの型，ルーティングパス情報から，複数のファイルを生成する．出力ファイルは，本システムで使用している ANSI C[32]および Occam 2 で記述されたプロセッサアドレス，エージェントアドレス，コマンド番号の記述されたそれぞれのヘッダファイル，それらが EusLisp で記述されたファイル，および，Occam 2 で記述されたルーティングテーブルファイルである．さらに同ユーティリティは，EusLisp において各エージェントをオブジェクト，各コマンドをメソッドとして利用するための定義が記述されたプログラムも同時に生成し，EusLisp 用出力ファイル内のプロセッサアドレス等の記述の後に付加する．

本ユーティリティを利用することにより，プロセッサの追加などのシステム拡張を行う際に，システム全体の整合性を容易に保つことが可能となった．また，メソッドの自動生成機能は追加されたコマンドの EusLisp での利用を容易にした．参考までに，プロセッサ 40 個，コマンド 151 個のシステムにおいて同ユーティリティで生成されたファイルは，Occam 2 用ヘッダファイル：283 行，C 用ヘッダファイル：297 行，EusLisp 用ファイル：1351 行であった．なお，本ユーティリティは bison および flex により作成された．

6. ETA3s 用プロトタイプマニピュレーションシステム

本システムはダイレクトドライブマニピュレータ(ETA3s)，力覚センサ(ピー・エル・オートテック社：F/T 5/50 HSS)，ホストコンピュータ (Sun microsystems 社：SPARC station 2)，トランスピュータ(INMOS 社：T 800, T 801, T 805)複数台，およびデバッグ用モニタ(NEC 社：PC 9801 DA)から構成されるマニピュレーションシステムである．ETA 3 s は ETA 3[9]の改造機であり，モータの一部を高出力のものに置き換えたマニピュレータである．ハードウェア構成図，コントロールシステム概観を **Fig. 9**, **10** に示す．

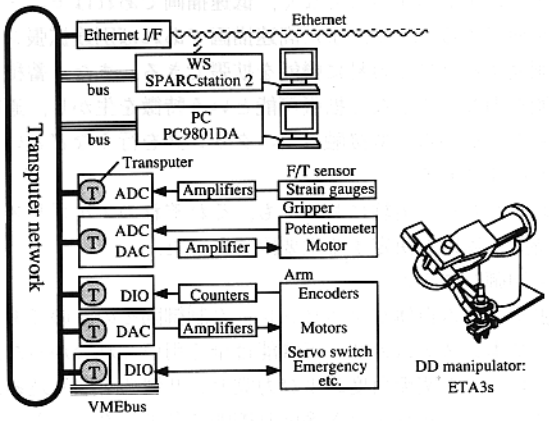


Fig. 9 Hardware system configuration

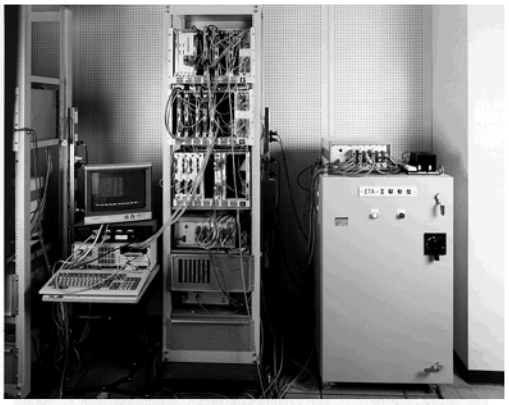


Fig. 10 Controller (left: Ethernet interface with a monitor, center: rack including transputers, right: servo driver)

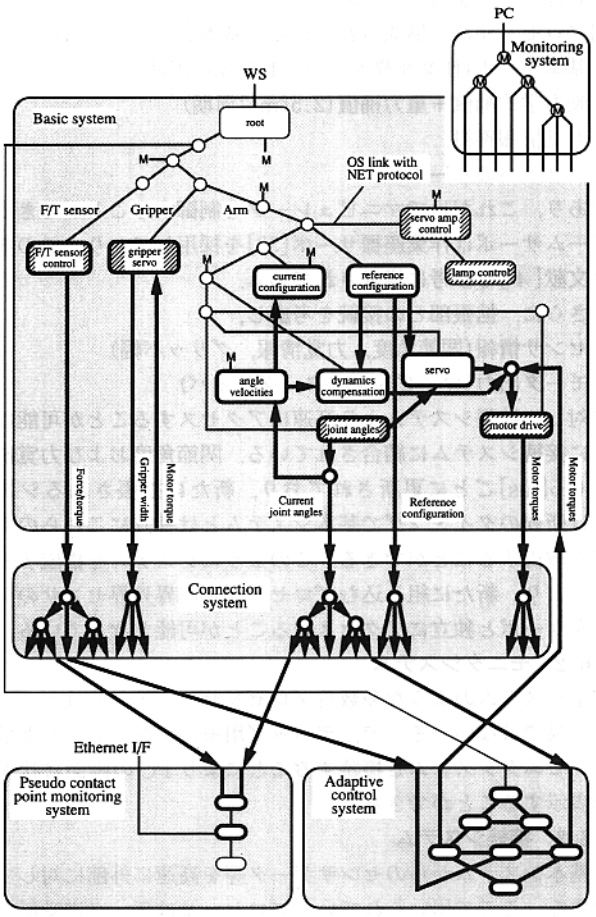


Fig. 11 Transputer network configuration

コントロールシステムは基本システム，モニタシステム，接続システム，および拡張システムより構成される．トランスピュータネットワーク構成図を **Fig. 11** に示す．図中の丸や四角はどちらもプロセッサを表し，それぞれを結んでいる線はリンクを表す．四角枠はルーティング以外にタスクを持つプロセッサである．丸は現在ルーティングのみを行っているが，これはトランスピュータのリンクが4本であるという制約から生じたものである．斜線で示されるものはI/Oと結合しているプロ

れるデータが固定のもの，矢印なしリンクはNETプロトコルを用いたデータ伝送ラインである．'M' はモニタシステムに接続されていることを示す．

太線で表されたプロセッサとリンクはそれぞれの計算能力，伝送速度を考慮し時間的にほんど余裕のない状態で用いられている．このように時間制約の厳しい状況での並列処理においては，少しの変更も全体に悪影響を及ぼすことがしばしばある．したがって，この部分は容易には変更しないようにしている．

システム記述言語はOccam 2とANSI Cであり，混在させて用いている．通信部分はOccam 2，計算部分はANSI Cを中心に記述されている．マニピュレータはワークステーション上のEusLispからコマンドを目的のエージェントに送ることにより動作させることができる．

6.1 基本システム

処理タスクを持つ12個のプロセッサと12個のルーティング専用プロセッサから構成されている．基本システムの機能は，

- 基本サーボ(作業座標サーボ：1.5[ms]周期)
- 動力学的補償＋重力補償(2.5[ms]周期)
- 力覚情報獲得
- グリッパのサーボ

であり，これだけでマニピュレータを制御することができる．アームサーボは作業座標サーボ[33]を採用しており，この部分は文献[4]を参考に構成されている．

さらに，拡張部との接続を考慮し，

- センサ情報(関節角度，力覚情報，グリッパ幅)
- モータ出力(マニピュレータ，グリッパ)

に対し，拡張システムより高速にアクセスすることが可能なように接続システムに結合されている．関節角度および力覚情報は0.5[ms]ごとに更新されており，新たに拡張されるシステムは所要のタイミングで基本システムとは独立にこれらのデータを取得することができる．このようにシステムを構成することにより，新たに組み込むプロセスから内界外界センサのデータをサーボと独立にアクセスすることが可能となっている．

6.2 モニタシステム

本システムのような多数台プロセッサシステムではデバッグが容易ではない．そこで，デバッグ用モニタシステムを準備した．モニタシステムと接続することによりPCの画面に状態等を表示することができる．

6.3 接続システム

基本システムからのセンサデータ等を高速に外部に対して供給する．ここで用いられているプロセッサはデータ送受信のみを可能な限り高速に行う．高速データ伝送ボードの伝送機能は本システムに含まれる．本システムは，2章で示したモデルにおいてポイントとなる，未知の入出力先に対してターンアラウンドタイムを変化させない結合を実現するためのシステムである．

6.4 拡張システム

拡張システムは，基本システムや蓄積されている拡張システムの機能だけでは実行できない処理を行わせるため必要に応じて開発され，基本システムに接続して用いられる．具体的な手

タ)を準備し実行したい処理アルゴリズムをコーディングし，基本システムや接続システムにケーブルで接続するだけである(Fig. 11 下部)．開発された拡張システムはシステムの資産として蓄積される．

これまで後述の2つの拡張システムが構築されており，ETA 3 sを用いて実験が行われた．実験に必要なデータ入出力周期を満足するようなシステムを用いれば，本システムでなくともこれらの実験を行うことは可能と思われる．しかし，制御実験では入出力周期がアルゴリズムの計算時間から決定されることが多いため，当初のシステム設計段階になかった機能，例えばデバッグのため動作中の関節角度データや関節トルクをリアルタイムで画面に表示する機能などを追加することは従来のシステムではむずかしく，可能であったとしてもシステムを再構成する必要がある．本システムを用いれば開発された制御実験システムを改造することなく，低速描画であればモニタシステムを利用することで，また高速描画であれば別途拡張システムを開発するだけで容易に機能を拡張できる．また，蓄積された機能を損なうことなく拡張可能という特徴を生かし，適応制御実験をしながら疑似接触点モニタリングを行うなどといったことも可能である．

なお，どちらの実験においても，それぞれのアルゴリズムの計算時間が周期を決定する主要因であった．

- 適応制御システム

拡張システム自体がマニピュレータ制御則そのものであったため，基本システムのサーボ機能は全く用いられなかった．マニピュレータの関節角度および力覚センサ情報を取り込み，拡張システム内で検証すべき適応制御則で各モータへのトルク指令を計算しマニピュレータを制御する実験を行った[34]．なお，同システム内のひとつのプロセッサと基本システム上部の

Fig. 12 Experiment of adaptive control

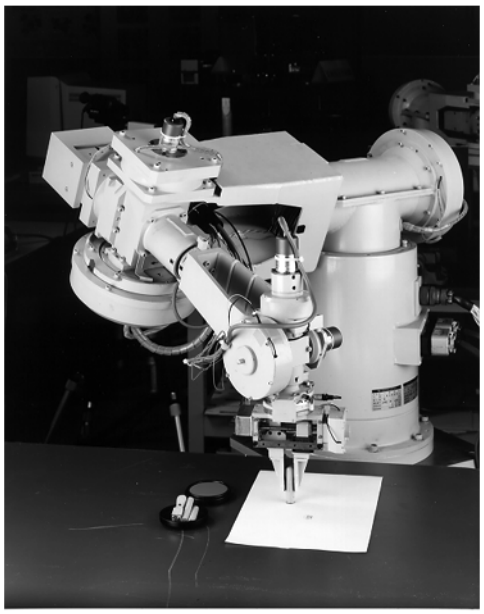


Fig. 13　Experiment of pseudo contact point monitoring

プロセッサとが接続しているが，このラインはホストとの通信のために確保されたものである．**Fig. 12** に実験の様子を示す．制御ループ周期が2[ms]と短かったため，従来であれば，システム全体を本実験用に再構成する必要があったが，本システムを利用することにより，基本システムに何も手を加えず開発された拡張部を接続しただけで実験を遂行することができた．

・疑似接触点モニタリングシステム

関節角度および力覚情報を利用し，捺印作業における疑似的な接触点位置を検出する実験が行われた[35]．本システムは，Ethernetインタフェースを介してホストコンピュータと結合している．実験の様子を **Fig. 13** に示す．本実験でのサンプリング周期は6[ms]であった．現在，検出された接触点位置をもとにマニピュレータを制御することを予定している．この場合には基本システムのサーボ機能を利用することで遂行することができると思われる．

7. ま　と　め

上位から下位までの拡張性とリアルタイム性を同時に有するシステムの実現を目指して開発されてきた並列処理型センサベーストマニピュレーションシステムについて，その設計指針，具体的なシステムハードウェア，ソフトウェアを示した．そして，同システムを利用して行われた研究用のシステム構成について述べた．

本システムはトランスピュータを用いたエージェントベースのシステムである．上位レベルのインテリジェントシステムから下位レベルのリアルタイム性を強く要求されるサーボシステムに渡る総合的なロボット制御が可能であること，技術の蓄積およびその利用が可能であるような拡張性を有することが特徴として挙げられる．現在，リアルタイム性の確保，および，データルーティングにおけるデッドロック回避に関してはユーザの責任で行う必要があるが，将来，それらを解決する手法が提案されれば本システムに組み込むことが可能である．

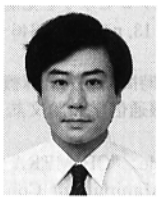
電子技術総合研究所では，空気圧ゴム人工筋アーム制御用システムを始めとした，本システムを利用した研究[36]が進行中であり，今後，さらに多くの研究のためのプラットフォームとして利用される予定である．

最後に，本研究を進めるにあたり，御助力頂いた高瀬前知能システム部長(現電気通信大学教授)，築根知能システム部長に深く感謝致します．また，日頃よりご討論頂いている電子技術総合研究所ロボットグループの皆様に感謝の意を表します．なお，本研究の一部はリアルワールドコンピューティングプログラムの一環として行われた．

参　考　文　献

[1] 石川正俊："センサフュージョンの課題"，日本ロボット学会誌，vol. 8, no. 6, pp. 735-742, 1990.
[2] 大西典子，大西献："可搬式汎用知能アーム登場！—オープンロボットの提案—"，日本ロボット学会誌，vol. 12, no. 8, pp. 1137-1142, 1994.
[3] 北垣高成，内山勝："研究用ロボット制御システム ARS/A"，日本ロボット学会誌, vol. 7, no. 5, pp. 475-481, 1989.
[4] 下倉健一朗，武藤伸洋："接触状態変化検出機能を有するマニピュレーションシステムに関する検討"，日本ロボット学会誌, vol. 12, no. 6, pp. 837-845, 1994.
[5] W. E. Ford："What is an Open Architecture Robot Controller ?," Proc. of IEEE Int. Symp. on Intelligent Control, pp. 27-32, 1994.
[6] 藤岡与周，亀山充隆，苦米地宣裕："再構成可能並列プロセッサと知能ロボット制御への応用"，日本ロボット学会誌, vol. 13, no. 6, pp. 846-853, 1995.
[7] 西田健次，戸田賢二，高橋栄一，山口喜教："実時間用並列計算機CODAのプロセッサアーキテクチャ"，電子情報通信学会論文誌, J78-D-1, no. 8, pp. 777-787, 1995.
[8] D. Schmits, P. Khosla, R. Hoffman and T. Kanade："CHIMERA : A Real-Time Programming Environment for Manipulator Control," Proc. of Int. Conf. on IEEE Robotics and Automation, pp. 846-852, 1989.
[9] 末広尚士，高瀬國克："スキル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. 8, no. 5, pp. 551-562, 1990.
[10] T. Hasegawa, T. Suehiro, T. Ogasawara, T. Matsui, K. Kitagaki and K. Takase："An Integrated Tele-Robotics System With a Geometric Environment Model and Manipulator Skills," Proc. of IEEE International Workshop on Intelligent Robots and Systems, pp. 335-342, 1990.
[11] T. Ogasawara, K. Kitagaki, T. Suehiro, T. Hasegawa and K. Takase："Model Based Implementation of A Manipulation System with Artificial Skills," 2nd International Symposium on Experimental Robotics, 1991.
[12] K. Kitagaki, T. Ogasawara and T. Suehiro："Methods to Detect Contact State by Force Sensing in an Edge Mating Task," Proc. of 1993 IEEE Int. Conf. on Robotics and Automation, pp. 701-706, 1993.
[13] 北垣高成，末広尚士，小笠原司："拡張性の高いセンサ情報並列処理ロボットシステム"，第10回日本ロボット学会学術講演会予稿集，pp. 1035-1036, 1992.
[14] 北垣高成，末広尚士，小笠原司："並列処理型センサベーストマニピュレーションシステムの構築－高速データ伝送ボードの開発－"，ROBOMEC' 94 講演論文集, pp. 673-674, 1994.
[15] 北垣高成，末広尚士，小笠原司："力覚データ並列処理システムの構築"，第12回日本ロボット学会学術講演会予稿集, pp. 671-672, 1994.
[16] 末広尚士，北垣高成："ネットワーク用ルータソフトウェア"，第11回日本ロボット学会学術講演会予稿集, pp. 901-902, 1993.

[17] 北垣高成, 末広尚士: "EusLispによるロボット制御用トランスピュータネットワークの柔軟な利用環境の実現", 第12回日本ロボット学会学術講演会予稿集, pp. 1055-1056, 1994.
[18] 末広尚士, 北垣高成: "トランスピュータネットワークを用いた柔軟なDDマニピュレータの制御システムの構築", 第12回日本ロボット学会学術講演会, pp. 1107-1108, 1994.
[19] T. Suehiro and K. Kitagaki: "A Multi-Agent Based Implementation of Task Coordinate Servo for the DD Manipulator: ETA3," Proc. of Int. Conf. on Intelligent Robots and Systems, pp. 459-464, 1995.
[20] P. M. Sharkey, R. W. Daniel and P. Elosegui: "Transputer Based Real Robot Control," Proc. of 29th Conf. on Decision and Control, pp. 1161-1162, 1990.
[21] H. Tinone and N. Aoshima: "Implementation of Distributed Robot Control System with Transputers," Trans. of the SICE, vol. 30, no. 6, pp. 636-640, 1994.
[22] H. Lecocq and Z. Zheng: "Parallel Processing for Real-Time Robot Manipulator Control," Proc. of 1994 IEEE Conf. on Control Applications, pp. 1043-1048, 1994.
[23] P. K. Sinha and P. Ho: "Transputer-Based Real-Time Parallel and Network Inferential Control of Robots," IFAC Algorithms and Architectures for Real-Time Control, pp. 81-85, 1991.
[24] I. P. W. Sillitoe and A. M. Tyrrell: "Evaluation of Cost Effective Transputer Architecture for the Implementation of the Computed Torque Method for Robotic Manipulators," Proc. of Int. Conf. on Control'91, pp. 885-860, 1991.
[25] Y. Kokusho, N. Doi, T. Ogasawara and H. Tsukune: "Parallel Computation for Manipulator Control Using the Double-Layered Load-Distribution Mechanism on Multi-Transputer Networks," Proc. of Int. Conf. on Intelligent Autonomous Systems, pp. 473-480, 1995.
[26] T. Henderson, C. Hansen and B. Bhanu: "The Specification of Distributed Sensing and Control," J. of Robotic Systems, vol. 2, no. 4, pp. 387-396, 1985.
[27] P. Woodbury, A. Wilson, B. Shein, B. Gertner, P. Y. Chen, J. Barttlet and Z. Aral: "Shared Memory Multiprocessors: The Right Approach to Parallel Processing," Proc. of 34th IEEE Computer Society Int. Conf. -COMPCON, pp. 72-80, 1989.
[28] Transputer Databook, INMOS Ltd., Redwood Burn Ltd., 1989.
[29] 出口光一郎, 森下巖, 小笠原司, 小野輝, 平沢裕, 渡辺淳: "単一バス同期データ変換型マルチコンピュータシステムの高能率化", 情報処理学会論文誌, vol. 20, no. 4, pp. 299-305, 1979.
[30] Occam2 Reference Manual, INMOS Ltd., Prentice Hall, 1988.
[31] T. Matsui and I. Hara: "EusLisp Reference Manual version 8.00," ETL-TR-95-2, 1995.
[32] ANSI C Toolset Reference Manual, INMOS Ltd., 1992.
[33] 末広尚士, 高瀬國克: "直接計算方式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.
[34] Y. Liu, S. Arimoto and K. Kitagaki: "Adaptive Control for Holonomically Constrained Robots: Time-Invariant and Tine-Variant Cases," Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 905-912, 1995.
[35] 北垣高成, 末広尚士, 小笠原司: "疑似接触点位置を利用したファインマニピュレーション", 第13回日本ロボット学会学術講演会, pp. 115-116, 1995.
[36] 宮腰清一, 北垣高成, 小笠原司, 築根秀男: "空気圧ゴム人工筋アームによるジャグリング", 第13回日本ロボット学会学術講演会, pp. 23-24, 1995.



北垣高成 (Kosei Kitagaki)

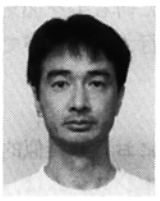
1961年5月27日生. 1984年東北大学工学部精密工学科卒業, 1989年同大学院工学研究科博士課程修了. 同年通産省工業技術院電子技術総合研究所に入所, 知能システム部行動知能研究室に配属. 現在, 主任研究官. 知能ロボット, 力覚センシング, ロボットシステムなどの研究に従事. 工学博士. 計測自動制御学会, 日本機械学会, IEEEの会員.

(日本ロボット学会正会員)

小笠原司 (Tsukasa Ogasawara)

1955年10月16日生. 1978年東京大学工学部計数工学科卒業. 1983年同大学院情報工学専門課程博士課程修了. 同年通産省工業技術院電子技術総合研究所入所. 現在, 知能システム部行動知能研究室室長. 1993-94年, ドイツ, カールスルーエ大学客員研究員. 知能ロボットの研究に従事. 工学博士. 計測自動制御学会, 情報処理学会などの会員.

(日本ロボット学会正会員)



末広尚士 (Takashi Suehiro)

1955年11月30日生. 1978年東京大学工学部電子工学科卒業. 1980年同大学院修士課程修了. 同年電子技術総合研究所に入所. 以来知能ロボットの研究に従事. 1990年東大工学博士. 1990-91年米国CMU客員研究員. 1993年3月新情報処理開発機構に出向. 現在, 同機構つくば研究センタ能動知能研究室室長. 計測自動制御学会の会員.

(日本ロボット学会正会員)

劉 雲輝 (Yun-Hui Liu)

1965年3月9日生. 1985年北京理工大学応用力学工学科卒業. 1986年10月来日留学. 1987年大阪大学大学院基礎工学系修士課程修了. 1992年東京大学大学院工学系博士課程修了. 同年通産省工業技術院電子技術総合研究所入所. 1995年2月香港中文大学機械自動化工学科講師, 現在に至る. 1994年日本ロボット学会論文賞受賞. ロボットの運動計画と制御の研究に従事. 工学博士. 計測自動制御学会, IEEEなどの会員.

(日本ロボット学会正会員)