日本機械学会論文集(C編)
63巻609号(1997-5)

論文 No. 96-0865

機能適応形マニピュレータ Fun-ARM における耐故障制御*

琴坂信哉*1, 浅間一*2
嘉悦早人*2, 遠藤勲*3

Fault-Tolerance of Functionally Adaptive and Robust Manipulator

Shin'ya KOTOSAKA, Hajime ASAMA,
Hayato KAETSU and Isao ENDO

Robots are required to have the ability to adapt their function according to the tasks to be carried out in an unexpected environment, and to execute tasks even if a part of the system is malfunctions. Fault Tolerance is a significant factor of functional adaptability. In this paper, a fault-tolerant control method with a proxy control strategy for a distributed manipulator is proposed. A Byzantine fault model is assumed in the method, where in the behavior of the faulty part cannot be predicted. The method focuses on malfunction of CPU (central processing unit) which is the controller of the manipulator. The method consists of procedures for fault detection, localization, containment, system reconfiguration and error recovery. The fault detection procedure is based on communication using shared memory. A voting algorithm for fault location is proposed. The fault-tolerance control method is implemented in a distributed manipulator with modular architecture, called Fun-ARM (functionally adaptive and robust manipulator). A reaching motion experiment with a CPU pseudo fault is shown, and the proposed fault-tolerant control method is verified.

Key Words: Robot, Fault Tolerance, Byzantine Fault Model, Distributed Manipulator

1. 緒　言

近年，技術の発達や人間の活動範囲の拡大により，宇宙空間や放射線環境といった環境下でさまざまな作業を行いたいという要求が出てきた．しかし，これらの環境は，人間にとって危険な環境であるため，作業のロボット化が希求されている．ところが，これらの環境下での作業は，さまざまな作業から構成されるため高い機能性や適応性，そして，高い安全性や信頼性が必要とされる．また，このような危険な環境では，故障したからといって人間作業員が直ちに修理に赴くといったことはできない．そのため，一部の故障であれば，作業を続行できるか，もしくは修理が可能な場所まで自力で帰還できるといった耐故障性が要求される．本研究では，このような作業環境の変化や自身の状況の変化に対して，自らの機能を再構成することによって適応させることのできる性質を機能適応性[1]と呼んでいる．現在，この機能適応性をもったマニピュレータとして，機能適応形マニピュレータ Fun-ARM(Functionally Adaptive and Robust Manipulator)[1][2]の開発を行っている．本論文では，機能適応性の中でも，特に自身の故障に対応できる性質，すなわち耐故障性について議論する．

これまで，ロボットの耐故障性の研究として，機構学的な冗長性を活用する方法[3]~[5]，機能縮退制御技術[6]などの研究が報告されている．ただし，これらの研究では，ロボットの制御中枢である情報処理機構の故障に関しては，まだ十分には検討されていない．

また，耐故障性実現の別な方向からのアプローチとして，自律分散形システムの研究がある[7][8]．これらの研究では，複数台のモジュール/ロボットを用いることによって，一部のモジュール/ロボットが故障したとしても，他のモジュール/ロボットが作業を引き継ぐことにより，システム全体としては，作業を完遂することができるという耐故障性が期待されている．特に，自律分散形ロボットシステム[8]では，その制御を分散的に行うことにより，情報処理機構の故障にも対処できることが期待できることからロボットの耐故障性実現の手段として，非常に有効な方法であると考えられる．しかしながら，ロボットの故障をどのように検出するのかといった具体的な手法に関しては，まだ十分に検討されていないのが現状である．また，こ

* 原稿受付 1996年5月30日．
*1 正員，(株)ATR人間情報通信研究所 (〒619-02 京都府相楽郡精華町光台2-2)．
*2 正員，理化学研究所 (〒351-01 和光市広沢2-1)．
*3 理化学研究所．

れまでの耐故障性に関する研究は，故障したロボット/モジュールの挙動に関しては，暗黙的に Fail-Stop 故障，すなわち故障した機器要素は完全に沈黙するということを仮定している．しかし，実際の故障においては，故障した要素が何らかの異常な動作を行う可能性もあり得る．

作業実行の側面からとらえた Fault-Tolerance の研究[9]もあるが，これはロバストな作業の実現を目的としたもので，ロボット自身の故障を取扱ったものではない．

一方，計算機科学の分野では，以前から耐故障性に関する研究が行われてきた[10][11]．近年，分散形システムによる耐故障性の研究も行われている[12][13]．しかし，実空間内を移動することによって作業目的を達成するロボットシステムでは，計算機システムと異なり，考慮しなければならない二つの制約条件が存在する．一つは，物理的な移動を可能にするため，システムの重量や体積の制約を受けることである．また，実時間で行動するために故障に対しても実時間での対処が要求される．計算機科学の分野で提案されている手法では，分散化された要素間の結合が非常に密であったり，通信機構が複雑であったりするため，そのままモジュール形ロボットや自律分散形ロボットシステムに適用しようとすると，モジュールの重量，体積の増加や通信路への負担増を招くという問題がある．

本論文では，情報処理機構の故障を想定した場合のマニピュレータにおける耐故障性について検討を行う．特に情報処理機構の故障には，故障した要素の挙動に仮定をおかないビザンチン故障モデル[14]を採用する．そして，上記制約条件を考慮した耐故障制御手法を提案する．この耐故障制御手法を，本研究で開発した機能適応形マニピュレータ Fun-ARM[1] に搭載し，その有効性の確認を行ったので報告する．

2. 分散形ロボットシステムによる耐故障性の実現

分散形ロボットシステムは，その柔軟性，頑健性などに着目されて多く研究が行われている．その特長の一つとして，耐故障性が指摘されている[15][16]．本研究でも，自らの構成を状況に応じて変更することにより，さまざまな作業，環境あるいは故障といった自身の状況の変化にも適応，対処することのできる能力をもった分散形マニピュレータとして，機能適応形マニピュレータ Fun-ARM の開発を行ってきた[1]．現在，本研究では，Fun-ARM を用いてロボットシステムにおける耐故障性の研究を行っている[2]．

分散形マニピュレータでは，その作業性能を向上させるために，制御系の小形化，モジュールの重量の低減が重要である．そのため，耐故障制御手法の開発にあたっては，次の点に留意することが必要である．

（1） 分散形マニピュレータにおいては，搭載できる機器は小形軽量であることが望ましいことから，小形の CPU でも処理できる程度に，機能なり情報処理量を少なくして簡素化することが必要である．そこで，分散化された情報処理機構間の通信やおのおのにおける処理負担が可能な限り小さい手法であることが望ましい．

（2） マニピュレータは，外界の物体との接触を行いながら作業を行うことが多い．その際の異常な動作は，作業の失敗，あるいは作業対象物や自分自身の損壊につながりかねない．そこで，実時間で故障に対処できることが必要である．

3. 耐故障制御手法

3・1 故障モデルとシステムに対する仮定 本研究では，情報処理機構，すなわちマニピュレータを制御する CPU の故障を対象とした耐故障制御手法の開発を行う．本研究では，手法を開発するにあたり次のような仮定を設けた．なお，これ以降 CPU をエージェントと呼ぶことにする．

（1） エージェント間の通信は，保証されている．

（2） システムを構成する要素は，十分に信頼性が高いと考え，複数箇所が同時に故障したり，故障に対処するための手続きを行っている最中に，新しく故障が発生したりしないことを仮定する（単一故障仮定）．

（3） 故障したエージェントは，計画的に他のエージェントの行動を妨害するような行動はとらないと仮定する（Random 故障仮定）．

（4） エージェントの故障に関して，先の（2），（3）の仮定に反しない限りビザンチン故障モデル[14]を仮定する．すなわち故障を起こしたエージェントは，他のエージェントに対して虚偽の報告を行ったり，異常な命令をコントローラに発したりする可能性があることも考慮する．

3・2 耐故障制御戦略 本研究では，耐故障性を実現する手段として分散制御系の採用と代替制御方式による耐故障制御手法を提案する．

故障時に，マニピュレータの作業性能を維持するためには，その自由度を維持することが最も重要である．代替制御方式では，マニピュレータの各自由度は，それぞれ独立した情報処理機構により分散的に制御される．一部の情報処理機構が故障した場合には，他の

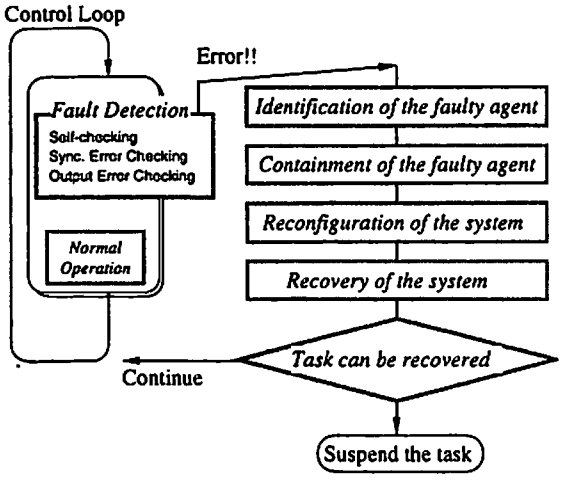

Fig. 1　Strategy for fault tolevance

正常な情報処理機構がその制御を代替することにより，自由度を縮退させることなく作業を継続する．図1に，本研究で提案する耐故障制御手法の流れを示す．

図1左側に示したように，通常マニピュレータは，故障検出を含む制御ループを実行している．特に故障が検出されなければ，制御出力算出，センシング処理を実行する．もし，故障が検出された場合には，図1右側の処理に流れが移る．右側の処理の流れでは，まず故障箇所の同定が行われる．これは，本手法ではビザンチン故障モデルを想定しているため，故障が原因となってエージェントが誤判断をしている可能性を考慮したためである．故障箇所の同定後，故障箇所の隔離を行う．これも同様に，故障したエージェントによる異常な制御指令の出力等による異常動作を防ぐためである．そして，残りの正常なエージェントで，隔離されたエージェントが担当していた処理をどのエージェントに引き継がせるのかを決定する．これが，システムの再構成である．最後に，故障によりシステムの動作に回復不能な影響が出ていないかを確認し，必要に応じて修復処理を行い作業へと復帰する．

3・3　耐故障制御手法

3・3・1　故障検出　本手法では，実時間での対処を保証するため，制御サイクルごとに相互に正常性(故障の有無)を検査することにより故障検出を行う．通信量の低減のため，各エージェントは，相互に正常性の検査に必要な情報を一定時間おきに報告しあう．それぞれのエージェントでは，この情報に基づき，次の3種類からなる手法により正常性の検査を行う．

(1)　自身の正常性の検査を自分自身で行う(Self-Checkingによる故障検出手法)．

(2)　すべてのエージェントで同一の処理を行い，その処理結果を比較することにより正常性の検査を行う(同一処理結果の比較による故障検出手法)．

(3)　他のエージェントの制御出力値，センサ信号処理結果を取込み，ロボットの設計仕様からそれが許容あるいは想定される正常範囲に収まっているかどうかを検査することにより正常性の検査を行う．これは，CPUの故障時には正常範囲を外れた出力が現れることを想定している(出力比較による故障検出手法)．

3・3・2　故障箇所の同定　本手法ではビザンチン故障を仮定しているので，検出された故障そのものが，故障したエージェントによる誤判断の可能性がある．そこで，本手法では，検出された誤りの種類により以下に述べる故障箇所の同定手続きを行う．

(1)　Self-Checking手法により検出された場合

故障箇所は，あきらかに自分自身である．

(2)　同一処理結果の比較，もしくは出力比較による故障検出手法により検出された場合

故障箇所を同定するため多数決を利用した投票手続きによる同定手続きを行う．この投票手続きは，Lamportの Voting Algorithm[17]を参考に作成した．故障を検出した時点で，これまで正常とされていたエージェントの台数(n)により手順が異なる．

(2・a)　$n \geqq 3$ である場合

単一故障仮定より，最低限3台のエージェントが存在すれば，正しい故障箇所の同定が可能である．投票手続きでは，まず故障を検出したエージェントは，他のエージェントすべてに故障を検出したエージェントのIDと検出された誤りの種類を通知する．各エージェントは，一定時間以内に誤りが指摘されたエージェントの誤り検出結果を他のエージェントすべてに報告する．それぞれのエージェントにおいて，報告された結果を集計し，自分の意見を含めて(そのとき正常とされていたエージェントの台数 -1)台の賛成が得られたのであれば，その検出結果が正しいと判断する．

(2・b)　$2 \geqq n$ であった場合

n が2台以下である場合は，多数決が成立しないため，故障の検出時点で，その判断が正しいとする．

3・3・3　故障箇所の隔離　故障したエージェントが異常な命令をモータコントローラ等に発することができないように，故障したエージェントをシステムのバスから切り離す隔離操作を行う．

3・3・4　システムの再構成　故障したエージェントの行っていた処理は，その時点で最小のIDをもつエージェントが行うこととする．ただし，処理負担の均等化のため，最小のIDをもつエージェントがすで

に代替制御を行っていた場合には，次に小さい ID をもつエージェントが代替をすることとした．

3・3・5 故障の影響の修復，作業への復帰 代替を行うエージェントは，代替制御の対象となるモジュールから関節角の値を読み取り，故障による悪影響がないかどうかを判断する．故障の影響がなければ，代替エージェントは，制御指令を出力し，作業を続行する．故障により手先軌道がずれていた場合には，いったんマニピュレータの動作を停止し，再度軌道計画をやり直し作業を再開する．

4. 機能適応形マニピュレータ Fun-ARM への実装

4・1 Fun-ARM のシステム構成 図2に Fun-ARM のシステム構成を示す．

Fun-ARM は，分散化された複数のモジュールから構成されたモジュール形マニピュレータである．各モジュールには，LCS(Local Control System)と呼ばれる，CPU(Toshiba Corp. TMPZ84C015BF-10)，センサ I/F(National Semiconductor ADC0848)，モータ・コントローラ(Hewlett-Packard HCTL-1100)，分散形共有メモリ，BCU(Bus Control Unit)から構成される各モジュールの制御に必要な電装系が組み込まれている(図3)．Fun-ARM では，モータの制御，センサ信号処理などのローカルにできる処理は，この LCS が担当するという分散制御方式を用いている．各モジュール内の CPU は，コモン・バスによって接続された分散形共有メモリを介して情報交換が可能である．この分散形共有メモリは，各モジュールに分散的に配置された Dual Port RAM(Fujitsu MB8421)から構成され，コモン・バス側，および各 LCS 内のローカル・バスからのアクセスが可能なように設計されている．分散形共有メモリを用いることにより，各モジュール間の情報交換の効率を向上できる．また，一部の共有メモリが使用不能になっても，他のモジュール間の通信には影響を与えずにすむという利点がある．各 LCS 上の BCU には，モジュール内の CPU が故障した場合に，他のモジュール内の CPU がその制御を代替して行うことができるように，バス・アビトレーション機能，各分散形共有メモリへのアクセス制御機能，必要に応じて CPU を停止し，隔離を行うための HALT 回路が集約されている．この HALT 回路は簡単なラッチ回路で，入力はコモン・バス上のアドレス空間に割当てられており，その出力はそれぞれのモジュールの CPU の HALT 信号入力端子に接続されている．この HALT 回路にコモン・バスより書き込みを行うと，HALT 信号を出力するようになっており，この回路を用いて CPU を強制的に停止させることができるようになっている．また，マニピュレータ基部のベース・モジュールには，PC/AT 互換機があり，その I/F カードはコモン・バスに接続されている．マニピュレータと外部との情報交換は，この計算機を介して行われる．

4・2 耐故障制御手法の実装 3章で述べた耐故障性制御手法の実装は，以下のように行った．

(1) 故障検出手法の実装

・Self-Checking による故障検出手法

各モジュールの自身の正常性の検査は，CPU に内蔵された WDT(Watch Dog Timer)を用いて行う．ハードウェアによる方法のため，システムに対する処理負担をかけることなく，信頼性の高い検査が可能である．

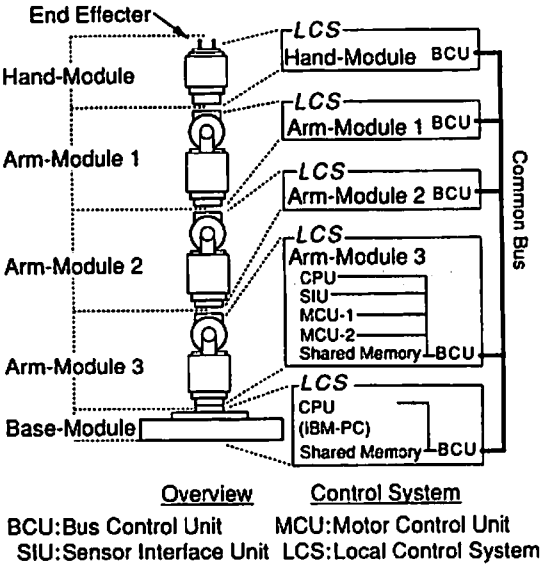

Fig. 2 Structure of Fum-ARM

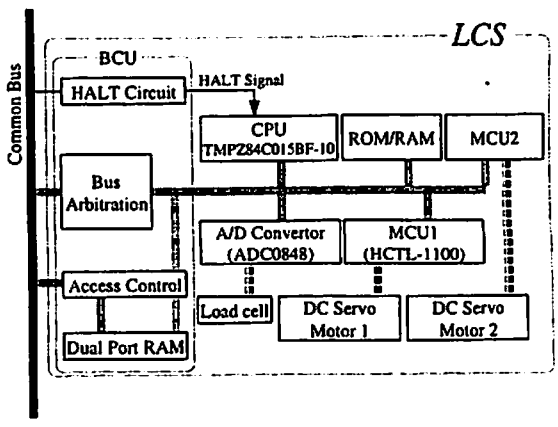

Fig. 3 Blockdiagram of bocal control System

・同一処理結果の比較による故障検出手法

分散形制御システムでは，統合的な動作を行うために各エージェント間で同期が取れていることが重要である．この同期をとる処理を“すべてのエージェントで行われる同一の処理”としてとらえ，同期誤りがあるかどうかを検査することにより正常性の検査を行う．しかし，1台のエージェントのクロックに依存した同期手法では，耐故障性が損なわれてしまう．そこで，本手法ではソフトウェアによる緩やかなモジュール間同期機構を用いた．モジュール間の同期，および同期誤りの検出に用いられる同期カウンタは，Lamport の方法[18]を用いて実現されている．各エージェントは，自身の同期カウンタの値を更新するごとに，自身の共有メモリに書き込む．それぞれのエージェントは，他のエージェントのカウンタの値を読み取り，同期動作を行う．

・出力比較による故障検出手法

制御サイクルごとに，モータコントローラに出力される速度指令値を検査することによって行う．

制御プログラムの作成にあたっては，通常の制御指令の算出に必要とされる時間に加えて，故障検出処理に必要とされる時間，故障箇所を発見した場合の故障箇所の同定，システムの再構成，および代替制御を行うための制御出力の計算を行うための時間の余裕をもたせてある．これは，代替制御を行う場合に，滑らかに制御サイクルをつなぐために必要な余裕である．

（2） 故障箇所の隔離操作

隔離操作は，ハードウェア的に行う必要がある．厳密には，この隔離操作も多数決回路により起動されることが望ましいが，回路が繁雑になること，故障したエージェントによって隔離のための回路が起動されることは稀であると考え，多数決回路は省略することにした．エージェントは，故障したエージェントを HALT 回路を用いて，強制的にバスから切り離す．

（3） 故障からの修復，作業への復帰

実際のエンコーダ値と，その制御サイクルで予測されるエンコーダ値との差がクロックの最大ずれ量から想定される差よりも小さい場合には，故障の影響はないと判定し，ただちに作業へ復帰する．

5. 耐故障制御実験

本研究で提案した耐故障制御手法の有効性を評価するために，以下のような耐故障制御実験を行った．実験には，ベース・モジュール，アーム・モジュールを2台，ハンド・モジュールの計4台のモジュールを用いて行った．ID 番号は，マニピュレータの根元側からベース・モジュールを 0 番としてアーム・モジュールをそれぞれ 1，2 番，ハンド・モジュールを 3 番とした．実験では，ベース・モジュールに他のモジュールの関節角計測用のプログラムを載せ，関節角変化の計測を行った．

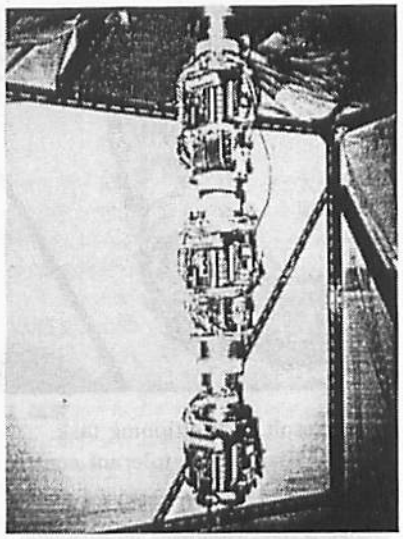

Fig. 4 Initial Pose

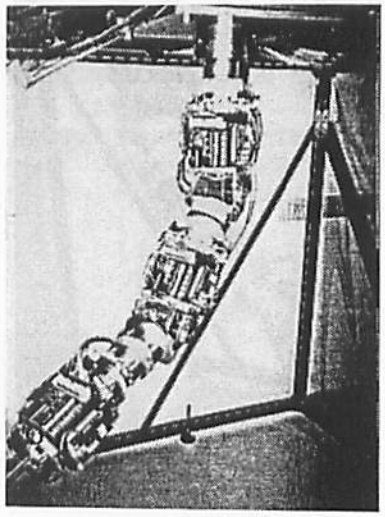

Fig. 5 Target Pose

実験では，手先を図4に示した初期位置から，図5に示される位置に移動させることを作業目標とする．作業目標点は，ベース・モジュール上の共有メモリ上に置かれる．各エージェントは，この作業目標点を読み込み作業を実行する．実験では，この作業目標点は，あらかじめオフラインで計算された作業目標位置に至るまでの経由点における関節角目標値，目標到達時間の組で表現されている．各モジュールは，これらの値に従って，制御サイクルごとに速度指令を算出し，各軸のモータ・コントローラに出力する．

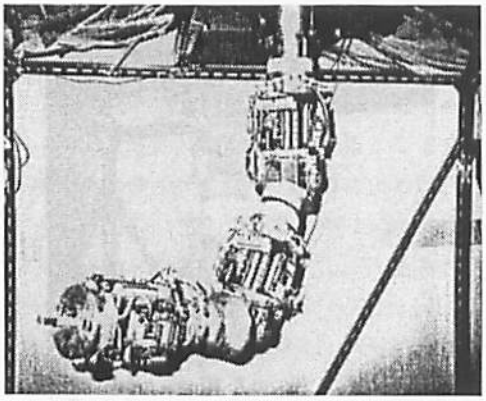

Fig. 6 Result of Positioning task
(without fault-tolerant control)

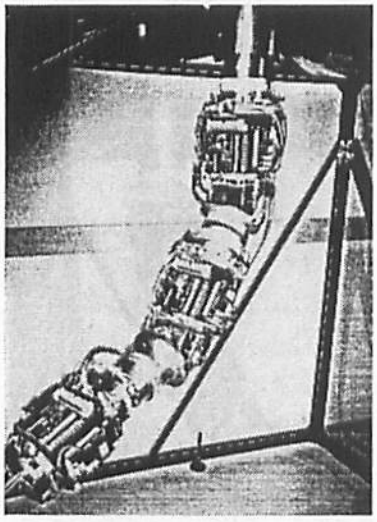

Fig. 7 Result of Positioning task
(by using fault-tolerant contlrol)

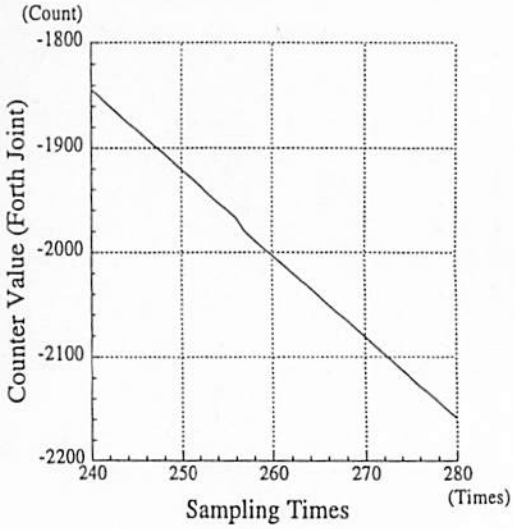

Fig. 8 Transition of joint-4 angle

また，出力比較による故障検出のための正常な速度指令値の範囲は，関節保護のために設定した最大加速度と実験に採用した作業内容から求めた最大速度より，速度指令値がC0(H)以上，40(H)以下(2の補数表示)であれば正常と判断することにした．なお，速度指令値のフルスケールは，80(H)から7F(H)である．

実時間での対処が可能かどうかを判断するために，本手法を組込んだ制御プログラムの実行時間の計測を行った．まず，制御プログラムの1サイクルタイムは，21.5 msである．そのうち，同期動作および故障検出処理に用いられている時間は，0.6 msであることが確認できた．また，1 msごとにカウントアップされる同期カウンタを用いて，各モジュール間で最大誤差1.5 ms以内で同期がとれることが確認できた．

続いて，第3関節(回転関節)，および第4関節(屈曲関節)を制御するアーム・モジュール(ID: 2)のCPUが故障することを想定して，上述の手先位置決め作業を実行する実験を行った．まず図6に，耐故障制御手法を用いない場合を示す．作業の途中でアーム・モジュール(ID: 2)のCPUが停止することを想定している．CPU停止後，アーム・モジュール(ID: 2)は，停止する寸前の速度指令が有効のまま動作を続けたため，第4関節が目標関節角を大幅に超えてしまった．最終的には，リミットスイッチが入り，モータコントローラが強制的に第4関節の動作を停止させている．図6では，手先が上向きに上がりすぎていることがわかる．

次に，耐故障制御手法を用いた場合を図7に示す．図6に示した状態と異なり，アーム・モジュール(ID: 2)の屈曲関節(第4関節)が行きすぎずに，目標関節角で停止していることがわかる．図7においても，アーム・モジュール(ID: 2)のCPUが停止するも，アーム・モジュール(ID: 1)が代替制御を行うことにより，手先は目標位置に位置決めすることができた．このとき，各モジュールは，停止したアーム・モジュール(ID: 2)を同期誤りにより異常を検出し，故障箇所同定手法により，故障エージェントを同定した．このとき，第1，第2関節を担当するアーム・モジュール(ID: 1)が，一番小さなIDをもつことから(実際にはベース・モジュールが最小のIDをもつが，この実験では関節角計測を行わせるため，ベース・モジュールには代替制御は行わせないことにした)，代替制御を行う．アーム・モジュール(ID: 1)は，アーム・モジュール(ID: 2)のモータ・コントローラから現在関節角を読み取り，制御可能範囲を逸脱していないことを確認した後，アーム・モジュール(ID: 2)に出力する

制御指令を算出し出力している．

図8に，代替制御が行われた第4関節の制御サイクルごとに計測した関節角の変化を示す．縦軸は，エンコーダパルスのカウント値，横軸は，制御サイクルごとの，すなわち 21.5 ms ごとのサンプリング回数をあらわす．第4関節の制御は，256 times の時点で切り替わっている．グラフの傾きを見てわかるように，ほぼ滑らかに制御が切り替わっていることが確認できた．切り替わり時に若干の変化が見られるが，これは代替を引き受けたエージェントと代替制御を受けるモジュールの間のわずかな同期のずれが原因と思われる．

以上の実験より，耐故障制御手法を用いることにより，一部のモジュールの CPU が故障したとしても，他のモジュールの CPU がその制御を引き継ぐことにより，作業が達成できることが確認できた．また，実時間で対処可能であることも確認でき，耐故障制御手法が有効であることが確認された．

6. 結　　言

本研究では，ロボットマニピュレータを対象として，その制御に用いられる CPU の故障に着目し，分散制御系を用いることにより耐故障性を実現する耐故障制御手法の提案を行った．特に本手法では，故障モデルとしてビザンチン故障モデルを想定し，より現実的な故障に対処できるようになっている．実際には機能適応形マニピュレータ Fun-ARM に搭載し，耐故障制御実験を行うことによりその有効性の確認を行った．本手法は，実時間で対処を行うので，耐故障性の向上とともに故障した際の異常な動作も未然に防ぐことができ安全性の向上も期待できる．

ただし，本論文で提案した方法では，WDT による Self-Checking や同期誤り，出力誤りの検査による故障検出手法を用いたが，将来的には，他の原因による故障も検出できるように故障検出手法を拡張していく予定である．また，本手法では実時間での対応に重点を置いたために，一過性の故障からの復帰に関しては考慮していない．一時的にロボットを作業から退避させ，自己診断を行い，正常が確認された CPU に関しては復帰させるなどの機能を付加したいと考えている．また，本論文では CPU の故障のみに着目したが，例えば Paredis の手法などを統合することにより，他の部位の故障にもある程度対処できると考えている．

本研究は，「原子力プラント内保全作業用ロボットシステムの開発研究」に基づき理化学研究所で行われた．また，機能適応形マニピュレータの開発にあたっては，(株)東芝　原子力技術研究所の多大なるご協力を頂きました．ここに感謝いたします．

文　　献

(1) 琴坂伯哉・浅間　一・嘉悦早人・大森弘亨・遠藤　勲・佐藤勝彦・岡田　敏・中山良一・園部博之，機能適応型マニピュレータの開発(その1)-Fun-ARM の基本設計と内蔵型触覚センサの試作-，日本ロボット学会第2回ロボットシンポジュウム予稿集，(1992)，157-162.

(2) 琴坂伯哉・浅間　一・嘉悦早人・大森弘亨・遠藤　勲・佐藤勝彦・岡田　敏・中山良一，機能適応型マニピュレータの開発(その4)-Fun-ARM における故障検出-，同定，隔離のアルゴリズム，日本ロボット学会第3回ロボットシンポジュウム予稿集，(1993)，357-362.

(3) Paredis, C. J. J. and Khosla, P. K., Global Trajectory Planning for Fault Tolerant Manipulators, *Proc. of the 1995 IEEE/RSJ Int Conf. on Intelligent Robots and Systems*, 2 (1995), 428-434.

(4) Yung Ting, Sabri Tosunoglu, Benito Fernäez, Control Algorithms for Fault-Tolerant Robots, *Proc. of the 1994 IEEE Int. Conf. on Robotics and Autom.*, 1 (1994), 910-915.

(5) 木村真一・土屋　茂・鈴木良昭，自律分散的アルゴリズムによる冗長多関節マニピュレータの制御，第34回計測自動制御学会学術講演会講演論文集，(1995)，505-506.

(6) 稲坂重孝・清水祐次郎・林　哲司，極限作業用ロボットの機能縮退制御技術の開発，日本ロボット学会誌，8-6(1990)，48-58.

(7) 森　欣司・宮本提二・井原廣一，自律分散概念の提案，電気学会誌，104-12，C(1984)，303-310.

(8) Fukuda, T. and Nakagawa, S., A Dynamically Reconfigurable Robotic System (Concept of a System and Optimal Configuration), *Porc. of IECON '87*, (1987), 588-595.

(9) Lueth, T. C. and Laengle, T., Fault-Tolerance and Error Recovery in an Autonomous Robot with Distributed Controlled Components, *Distributed Autonomous Robotic Systems*, (1994), 41-49, Springer-Verlag.

(10) Hopkins, A. L., Smith, T. B. and Lala, J. H., FTMP-A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft, *Proc. IEEE*, 66-10 (1978), 1221-1239.

(11) Smith, P. M. M. and Schwartz, R. L., Formal Specification and Mechanical Verification of SIFT: A Fault-Tolerant Flight Control System, *IEEE Trans. Computers*, C31-7 (1982), 616-629.

(12) Hariri, S., Choudhary, A. and Sarikaya, B., Architectural Support for Designing Fault-Tolerant Open Distributed Systems, *Computer*, 25-6 (1992), 50-62, IEEE.

(13) Ayache, J. M., Courtiat, J. P. and Diaz, M. REBUS, A Aault-Tolerant Distributed System for Industrial Real-Time Control, *IEEE Trans. Computers*, C31-7 (1982), 637-647.

(14) Lamport, L., Shostak, R. and Pease, M., The Byzantin Genral Problem, *ACM Trans. Programming Languages and Systems*, 4-3 (1982), 382-401.

(15) 浅間　一，マルチエージェントから構成された自律分散型ロボットシステムとその協調的活動，精密工学会誌，57-12(1991)，2117-2122.

(16) 小鍛治繁，フォールトトレラントな機械，精密機械，48-1(1982)，115-118.

(17) Pease, M., Shostak, R. and Lamport, L., Reaching Agreement in the Presence of Faults, *J. ACM*, 27-2

(1980), 228-234.
(18) Lamport, L. and Smith, P. M. M., Synchronizing clocks in the presence of faults, *J. ACM*, 32-1 (1985), 52-78.