モジュールロボットの Plug-and-Play 機構の設計

Designing of Plug-and-Play System for Modular Robot

○　小島 佑太 (慶大)　佐々木 貴宏 (慶大)　平 哲也 (慶大)　山浦 康史 (慶大)　山崎 信行 (慶大)

Yuta Kojima. Takahiro Sasaki. Tetsuya Taira. Kouji Yamaura. Nobuyuki Yamasaki.
Keio university. Hiyoshi 3-14-1, Kouhoku-ku, Yokohama, Kanagawa.

This paper describes the Plug-and-Play for Modular robot. The Plug-and-Play for Modular robot is not only to recognaize plug-in/out mechanism. It includes buildings network for Modular robot, having shared information and creating a situation for being able to collaborate each other. But There are many problem for Plug-and-Play on Modular robot. So I propose that the concrete definition of Plug-and-Play for Modular robot. And then I implement this mechanism to evaluate it. The result show that our Plug-and-Play mechanism is effective for modular robot.

Key Words: plug-and-play, destributed modular robot

1　序章

近年ロボットは, あらゆる現場に登場するようになった. それは, 環境的適応性や機能性が求められていることを意味する. 今日, 実用化されているロボットは、ある目的のために最初から単体に作り込む設計がなされている。それでは、機能的な拡張性を保持できずに様々な現場への環境適応力に今後限界が出てくると考えられる. そこで, ある環境に求められている機能や形状にあったものを適宜組み合わすことができるモジュールロボットの必要性がでてくる. ここでのモジュールロボットとは, それぞれが計算機を持ち, 様々な形状や機能を個別に保有しているものとする. 本研究ではこれを機能別モジュールロボットと呼ぶことにする.

モジュールロボットは, 組み合わせることが可能なロボットである. その着脱はそれを使用する誰もが煩しい作業をすることなくできることが望ましい (Plug-and-Play 機構). この Plug-and-Play 機構は, 物理的な着脱認識をハードウェアやソフトウェア認識するのみではない. トポロジに関係なくモジュールのネットワークを動的に構成し, 共有情報を持ち合うことでロボットを安定させ, 最終的にはモジュール同士の協調動作を始められる状態を作り出すことまでを指す.

そこで本研究では, モジュールロボットのための Plug-and-Play 機構を定義し, 現段階における Plug-and-Play 機構の問題点と提案を述べる. そして Plug-and-Play 機構における一連の流れを設計し, 実装を行った.

2　背景と関連研究

モジュールロボット形態には個々のモジュールが同型のものとそうでないものとにわけられる. 個々のモジュールが同型なものには, 動的再構成可能ロボット [3] や合体変形ロボット [4] などがある. 一方では個々のモジュールを機能単位で設計しているロボットがある [5]. 前者は, 環境に適応するために自己の構造を動的変えることを視野に入れている. しかし、形状の規格からそれにのせる機能やアプリケーションに限界が出てくる. そこに機能別モジュールロボットの研究の必要性が出てくる.

モジュールロボットの着脱に関する研究は行われているが, モジュール同士がどのような情報を共有し, どのように協調動作を行っていくのかという問題を抱えているのが現状である.

3　提案

上述したように, 機能別モジュールロボットにおける Plug-and-Play 機構は, 物理的な着脱をハードやソフトウェアで認識するだけでない. 本章では, 機能別モジュールロボットのための Plug-and-Play 機構の定義を明記する. さらに Plug-and-Play 機構の中で, モジュール同士がどのような, どの程度の情報を共有させればいいのかを明確にするために, モジュール間共有情報という枠組みで述べる. それ以外の情報を相手指定型共有情報とすることで, Plug-and-Play 機構においてどこまでの情報を共有させるかを明らかにしたい.

3.1　Plug-and-Play 機構の定義

以下の一連の流れを機能別モジュールロボットのための Plug-and-Play 機構とする.

1. モジュールの物理的接続のハードとソフトでの認識
2. 接続相手モジュールとの隣接関係の確立
3. トポロジに関係なくモジュールの通信ネットワークの動的構築
4. 着脱の安定に必要な情報の共有
5. モジュール同士が協調動作できる状態

3.2　モジュール間共有情報

すべての接続しているモジュールに共有情報を持たせることは, Plug-and-Play 機構の一部である.Plug-and-Play 機構はモジュール同士の協調動作することが可能な状態を作り出すまでを指すので, それまでに時間をかけることはそれ以降の制御に支障が出る. よって, 共有情報は最低限に抑えることで,Plug-and-Play を早めに完了させたい.

モジュール間共有情報は, モジュールロボットの着脱の際に自動的にすべてのモジュールが共通して持つ情報のことである. 共有情報の内容としては, 機能別モジュールロボット同士の着脱後にもバランスを保つための情報と, ロボット全体の安全のための可動領域情報を含むべきであると考える. 前者の方は, 重心や, 着脱による新たな付加で発生するモーメント情報を共有情報としたい. 後者の方は, すべてのモジュールの可動領域を直方体ベースの空間で表すことで, ロボット全体がどの程度の可動領域があるかを共有情報とする. ここで可動領域を直方体ベースで表す理

由は, 送る情報が簡素なものですむからである.Plug-and-Play をすばやく行うためである, 細かな情報はついた瞬間の安全性を守るためには必要ない. それ以上のモジュールに関する細かい情報が必要な場合は次に記す相手指定型共有情報に委ねる.

3.3 相手指定型共有情報

相手指定型共有情報とは, 全体で共有するのではなく送信先もしくは送信元を指定し, 情報を伝達してそのモジュール同士のみで共有する情報のことを指す. さらに, この相手指定型共有情報は Plug-and-Play 機構には含まれていない. 全体で情報を共有する必要がない情報や共有情報についてさらに細かな情報が必要となった場合などの情報がこれに当たる. モジュール間共有情報と明確にこれを区別することで, 全体で共有する情報は, Plug-and-Play にかかる時間を少なく抑えるために限定していることを概念上で確実に区別する. 今回は、Plug-and-Play 機構の設計であるために, この情報に関しては, 設計していない.

仮に移動中モジュール同士の着脱も視野にいれると, 相手指定型共有情報も含めて Plug-and-Play 機構としなければならない. その上で, 相手指定型共有情報として何を含めなくてはならないかも問題となる. そういったシチュエーションでは, ロボットは使われないであろうという判断のもとで本研究は, 移動中のモジュール同士の着脱は考慮しない.

3.4 まとめ

モジュールの着脱における Plug-and-Play 機構をスムーズに行うために共有される情報は, 最低限におさえなければならない. また, 機能別モジュールロボットの Plug-and-Play 機構の中ですべてのモジュールが共有される情報は, 全体が共有することで初めて意味が生まれるものである. つまり, 個々のモジュールがどのような情報を必要としているかには主眼をおいていない.

4 設計

実装は *Responsive Processor*[2] を使用する. *Responsive Processor* は,4 つのポートを持つ. それぞれのモジュールをノードとして設計する. それぞれのノードは, 自分のどのポートにどのノードとリンクされているのかを表す Link-State と呼ばれる情報を持つ. ひとつのサブネットワークには一意のルートノードが存在し, それによりネットワークの区別ができる.

4.1 ハード・ソフトウェアの認識

ここでは, モジュールの物理的な着脱を認識する機能を設計する.

初期状態においては, 自己ノード (モジュール) をルートノードと設定する. また, モジュール (ノード)ID は, 初期状態で個別に一意に割り振る.

接続ケーブル (*Responsive Link[1]*) の着脱によっておこ電圧変化を *Responsive Processor[2]* についている AD コンバータを使用してデジタルで表された電圧値を読み取る. その際, 電圧の上下の差を 3.3V にして流し込む. その上下の間を 1000 分割し,350 を閾値とし, それを越える値ならば接続とし, それ以下ならば未接続状態とする. 着脱状態を知るというよりは. 着脱の変化を認識するためのものなので, プログラムとしては, 未接続から接続, もしくは接続から未接続と変化があった場合にそのルーチンから抜けて次の制御に移るという仕組みをとっている. スレッドが実行された最初に AD コンバータから読み取れる値は 0 となっているため, 初めから接続した状態で電源を入れても接続を認識することができる.

4.2 隣接モジュールの認識

着脱認識をハードウェア, ソフトウェアの両方で認識した後, 相手がどんなモジュールなのか情報交換を行う.

まず, モジュールは自分のノード ID と自分のルートノードを接続相手に投げる. 次に, 接続相手のノード ID 接続相手ルートノード ID を受け取る.

受け取ったら, 接続相手ノード ID と比較する. 相手より ID が大きいならば MASTER となり, 小さいならば SLAVE となる. 各モジュールは自分の持つ 4 つのポートに対して MASTER か SLAVE を決める. 接続相手のノード ID がわかった時点で自分の持っている Link-State を書き換える.Link-State 書換えた後は,MASTER となったノードが SLAVE となった接続相手に Link-State とルートノード ID を送る. 受け取った接続相手ノード (SLAVE 側) は,Link-State を書き換える. 書き換えたら,SLAVE 側ノードは書き換え後の Link-State を MASTER に送り返す. さらにこれを受け取った MASTER は, 自分の Link-State を書き換える. この作業で物理的に接続したノード同士は共通の Link-State を保持する. この流れを figure1 に示す.

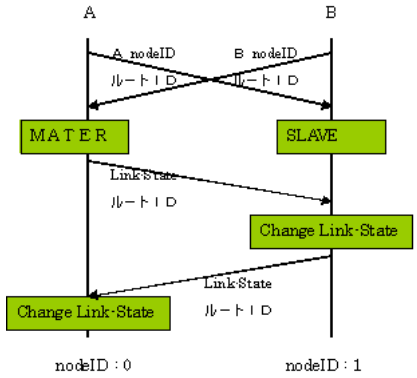

Figure 1: 隣接関係の確立

4.3 ルーティング

このセクションでは, トポロジに関係なくモジュールロボットのネットワークを構成するための設計を行う.

接続認識したノードはルートノード ID を接続相手のノードと比較している. そこで比較した結果により, 以下のパターン分けができる.

1. 接続相手のルートノード ID の方が大きい場合

   これは, 接続相手のノード (モジュール) が属すネットワークのルートノード ID より小さい結合ということである. つまり, 自ノードのルートノードが接続相手のルートノードに比べ優先されるということである.

   MASTER,SLAVE ともに接続されたノード (モジュール) 以外の隣接ノードに対して Link-State 情報の flooding 処理を行う. これは, ネットワーク上すべてのモジュールの持つ Link-State が統一されるまで行う.

2. 接続相手のルートノード ID の方が小さい場合

   これは, 接続相手のノード (モジュール) が属すネットワークのルートノード ID のより大きい結合ということである. つまり, 相手ノードのルートノードが自ノードのルートノードより優先させるということである.

   MASTER,SLAVE ともに接続されたノード (モジュール) 以外の隣接ノードに対して Link-State とルートノード情報の flooding 処理を行う. ルートノード情報を送る理由は, 接続相手のルートノードが優先されることで自ノードのルートノード ID が変わってしまったからである. この場合も, ネットワーク上すべてのモジュールの持つ Link-State とルートノードが統一されるまで行われる.

3. 接続相手とルートノード ID が等しい場合

   これは, 接続相手のノードと同じネットワークない同士の結合という事である.

   同じネットワーク内同士の結合であるため,MASTER,SLAVE ともに, ルートノードの flooding は必要ない. ここでは,Link-State の flooding を行う.

ここで,flooding 処理を以下の Figure2 の様に設計した.

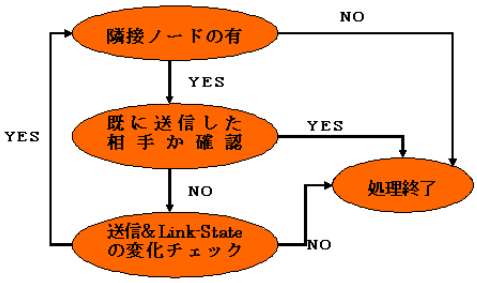

Figure 2: flooding 処理

ここまでの処理を終えた各ノード (モジュール) は,Link-State を元に最短パスツリーの作成を行う. 作成には SPF(Shortest Pass Fast) を使用する. 帯域幅を考慮す必要がないので, メトリックはホップ数を使用した.

最短パスツリーを作成したノードは, それをもとに, ルーティングテーブルを作成する. ルーティングテーブル作成後のプログラムは着脱認識へと戻る.

4.4 モジュール間共有情報

ここでは, 本研究におけるモジュール間共有情報について述べる. モジュール間共有情報は, Link-State を全体で統一する作業と同時に行われる. つまり接続したと同時に必要な情報である. ゆえに処理を早めに終わらせたい. 本研究では, 共有情報として Link-State やルートノード以外に, モジュールの直方体ベースの大きさで縦横高さ, 直方体ベースの可動領域空間の縦横高さをモジュール間共有情報として各ノードが保持する.

5 実装対象

5.1 共通事項

モジュール同士の通信規格としては *Responsie Link*[1] を使用する. それぞれのモジュールロボットには *Responsive Processor*[2] を搭載した.OS には *RT-Frontier*[7] を使用した. それぞれ分散リアルタイム処理用の通信規格,Processor,OS であるためにロボット制御においてリアルタイム処理を行うことを可能としている.

5.2 車輪ロボット

Figure3 の左図である [6]。Responsive Processor を搭載. 駆動輪は二つ. 地図生成機能を持つ. 自己位置推定機能を持つ. 可動型赤外線センサを前方と左右に 3 個保持.

以下の情報を保持している.

5.3 胴体ロボット

Figure3 の右図である.Responsive Processor を搭載.

以下の情報を保持している.

Table 1: 車輪ロボットの情報

nodeID	number	
大きさ	縦	70cm
	横	50cm
	高さ	30cm
可動領域	縦	70cm
	横	70cm
	高さ	30cm

Table 2: 胴体ロボットの情報

nodeID	number	
大きさ	縦	50cm
	横	80cm
	高さ	30cm
可動領域	縦	95cm
	横	95cm
	高さ	30cm

6 評価

6.1 評価環境

実装対象の車輪モジュールロボット, 胴体モジュールロボットを組み合わせる合体させることで Figure4 の左のようになる. また, Figure4 の右のような障害物が 3 つあるような環境で評価をとる.

評価は, 提案で定義した機能別モジュールロボットのための Plug-and-Play 機構の一連の処理ができているかを実装することで行う.

6.2 評価方法

- ロボットの初期位置を決める
- 障害物の地図情報を車輪モジュールロボットに与える
- 車輪モジュールロボットを単体で直進させる
- 幅の広い胴体型モジュールロボットを接続する
- ディスプレイを用いてどのような情報がやり取りされているかを見る
- モジュールロボットを合体させた状態で先ほどと同様の直進をさせる
- その際, 車輪モジュールロボット単体で直進させた場合と胴体型モジュールロボットを合体させた場合の動作の違いを見る
- 胴体モジュールを車輪モジュールからはずす

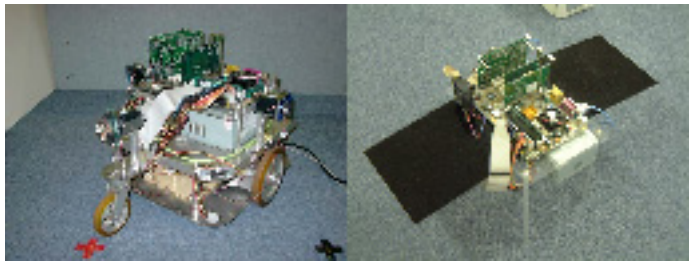

Figure 3: 車輪ロボットと胴体ロボット

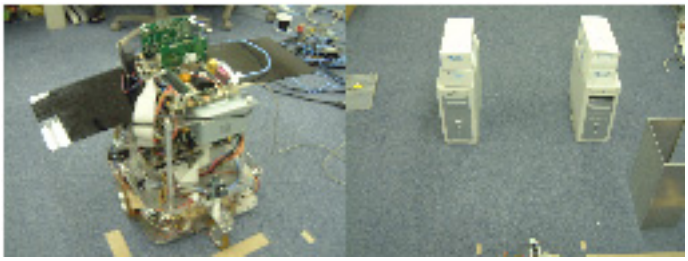

Figure 4: モジュールロボットと評価環境

6.3 評価結果

まず, 車輪ロボット単体で動作させた結果は, 単純にそのまま通れることをはじめから与えてある地図情報を元に判断し, 直進した. この簡素状態においても着脱認識のスレッドは回っている状態である.

次に胴体ロボットを車輪ロボットに装着させた. それと同時に物理的な着脱を *Responsive Processor* の AD コンバータの電圧値変化で認識し, それをソフトウェア側が接続と認識した. これはディスプレイに出力させることによって確認することができた.

次にディスプレイに出力して試したものは, モジュール同士でどのような情報を交換しているかである. 接続相手のノード ID やルートのノードの情報が入ってくる. 次に MASTER もしくは,SLAVE が決まる. その後は, 接続相手のモジュール間共有情報が出力された. この時点でわかることは, 相手モジュールとの隣接関係の確立, 通信ネットワークの動的確立, モジュール間共有情報の保持ができているということである. つまり, モジュール同士の協調動作をはじめられる状態ができあがったということである.

最後は, モジュール間共有情報の有用性についてである. まず, 車輪ロボットと胴体ロボットを合体させた状態で,Figure4 の評価環境のもとで, 先ほどと同じように直進させた. 正面の二つの障害物の間は,60cm であり胴体ロボットは, 通ることはできない. そのことを与えてあった地図情報から判断し停止した. 正面の通路が通れない場合は, 左からの迂回路を与えておくとそちらに向かった. 一時停止した後, 迂回路を選択するために左に 90 度回転する. その際に胴体モジュールの横に延びた部分が障害物に当たることなく回転を行えた. これは, モジュールが持っている可動領域の情報より, その範囲に障害物が入らないように止まることができた事がわかる.

この一連の評価によって本研究の目的であり, かつ提案で定義した機能別モジュールロボットのための Plug-and-Play 機構が実装でき, その有用性を示すことができた.

7 結論

機能別モジュールロボットの Plug-and-Play が実現されるためには, 物理的な結合をハード, ソフトウェアでの認識, 接続相手のモジュール情報を取得, モジュール間ネットワークの構築, 共有情報の取得, 協調動作できる状態を作ることであった. しかし、これを実現するには様々な問題があることを示した. 本研究はその問題に対する解決策として, モジュールロボットが共有すべき情報がどうあるべきかを述べ, その上で Plug-and-Play 機構の設計と実装を行った. また, 上述した評価を行うことでこの機構の動作を確認しその有用性を確かめることができた. 今後は, モジュールの着脱が行われたことでどのようなモーメントが働くかという情報を共有情報として持つことで着脱におけるロボットの安定性を確保できるような研究を行っていきたい.

謝辞

本研究は独立行政法人新エネルギー・産業技術総合開発機構 (NEDO), 慶應義塾大学大学院高度化推進研究費の支援による. また, 本研究の開発にあたり, 三和金属株式会社 野田正弘氏には大変お世話になりました. ここに深く謝意を表します.

References

[1] 山崎信行. 分散制御用リアルタイム通信 *Responsive Link* の設計および実装. 情報処理学会論文誌, Vol. 45, No. SIG3, pp. 50–63, 2004.

[2] 山崎信行. 並列分散リアルタイム制御用レスポンシブプロセッサ. 日本ロボット学会誌, Vol. 19, No. 3, pp. 68–77, 2001.

[3] Wei-Min Shen and Will P. Docking in Self-Reconfigurable Robots. In *IEEE/RSJ International Conference on Intelligent Robotsand Systems*, pp. 1–2, 2001.

[4] Kurokawa H, Kamimura A, Yoshida E, Tomita K, Kokaji S, and Murata S. M-TRAN Ⅱ:metamorphosis from a four-legged walker to a caterpillar. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 2454–2459, 2003.

[5] Tetsuya Taira, Nobuhide Kamata, and Nobuyuki Yamasaki. Design and Implementation of Reconfigurable Mudular Humanid Robot Architecture. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 1071–1076, 2005.

[6] 山浦康史, 鎌田展秀, 平哲也, 山崎信行. 可動型赤外線センサを用いた動的スキャンによる地図生成システムの設計と実装. 第 23 回日本ロボット学会学術講演会, pp. 1–4, 2005.

[7] Hidenori Kobayashi and Nobuyuki Yamasaki. A Real-Time Operating System for Practical Imprecise Computation. In *10th IEEE Real-Time and Embedded Techinology and Applications Symposium*, pp. 255–264, 2004.