分散オブジェクト技術を用いたオープンロボットコントローラ

Open Robot Controller Using Distributed Object Technology

尾崎　文夫　OZAKI Fumio
大明　準治　OAKI Junji

ロボット産業の活性化のためには，使いやすいロボットコントローラの出現が待ち望まれる。使いやすいとは，拡張性に富み，簡単に他のロボットや周辺機器とつないでシステムを構築できるようなロボットコントローラのことである。コントローラをソフトウェア・ハードウェアの両面でオープン化することにより，このようなコントローラが可能になる。当社では，分散オブジェクト技術を用いて，いろいろな周辺装置（画像処理装置や音声認識装置など）を簡単に接続できるようにしたオープンロボットコントローラの枠組みを開発し，適用例を通してその有効性を確認した。

To cope with various demands for robots, they must be much easier to use. We propose an open robot controller framework using PC/AT compatibles. The openness achieved makes it possible to build up many peripheral elements into a robot system without difficulty. The controller software is made up of object oriented technology and distributed object technology for extendibility. Toshiba has applied the framework to an experimental open controller to verify its validity.

1 まえがき

ユーザーの広範な要求に対応するため，あるいは新規技術をすばやく導入しユーザーに提供できるようにするためにいろいろな分野でオープン化が進み始めており，NC（Numerical Control）[1]あるいはロボットコントローラ[2]の分野でも同様の状況にある。オープン化とはソフトウェア・ハードウェアの改造や拡張を容易にしようという動きで，ここでは，ソフトウェア・ハードウェアのモジュール間インタフェースを標準化していくことと定義する。

この論文では，最近，急速に進んでいるいろいろな分野でのオープン化の流れに対して，ロボットコントローラ分野での当社研究開発センターのオープン化への取組み[3, 4]について述べる。コントローラをオープン化することによりネットワークを介して簡単にいろいろなシステムを接続でき，容易にシステムアップを図ることができる。

2章でオープン化の利点ついて述べ，3章で分散オブジェクト技術を利用したオープンコントローラを適用した例を示す。

2 オープン化の利点

オープン化することによる利点を，ソフトウェア・ハードウェアについて見ていく。

ソフトウェアの面では次のような利点がある。すなわち，新しいシステムが加わっても共通のインタフェースで利用できる，技術の蓄積・再利用が可能になる，ユーザーが自由にGUI（Graphical User Interface）や制御アルゴリズムのカスタマイズを行える，また，他業種からの参入が可能になるので産業のすそ野の拡大及び大幅なコストダウンが期待できる，などである。更に，再利用できる技術が蓄積されていくことにより，使いやすいロボットを提供できるようになる。多くのユーザーがいろいろなロボットプログラムを作成し，その中から再利用できる部分を抽出しまとめ上げていく作業をすることで，ソフトウェア工学におけるプログラムの再利用性と同じくロボットプログラムの再利用性を高めることができる。

ハードウェアの面では，豊富で安価な汎用のCPUボードやI/O（Input/Output）ボードを利用してコントローラを構築できるようになる。また，ネットワークを介して，簡単にいろいろな機器を接続し，相互に利用することができるようになる。オープン化により市場が活性化すれば，ソフトウェアと同様にサードベンダーの参入が可能となり，共通に使えるロボット用のボードの投入が期待できる。

ここでは，オープン化を達成するために分散オブジェクト技術を用いたコントローラについて説明する。

3 オープンコントローラの適用例

この章では，オープンコントローラを複数のロボットから成るシステムに適用した例について述べる。オープンコントローラを使用すると，アームなどの構成要素をつなぎ合わせてロボットシステムを構築できることを示す。まず，システムの構成を示し，次にオープンコントローラのインタフェース，リアルタイムOS（基本ソフトウェア）の選択について，最後に

コントローラの動作について説明する。

3.1 オープンコントローラを用いたシステムの構成

オープンコントローラを適用したシステム例を図1に示す。このシステムは，図右の移動台車の上に白い小型アームが搭載された移動ロボットと左の固定ロボット，ヒューマンインタフェース（HI）パソコン（PC），画像処理PC（移動ロボット上に設置）から成り，それらがLANにより相互接続されている。移動ロボットについては無線LANにより接続されている。ロボットはHI PCから共通のコマンドを介して動作させる。固定ロボットと移動ロボットの組合せにより，1台ではできなかった作業への適用などを考えている。

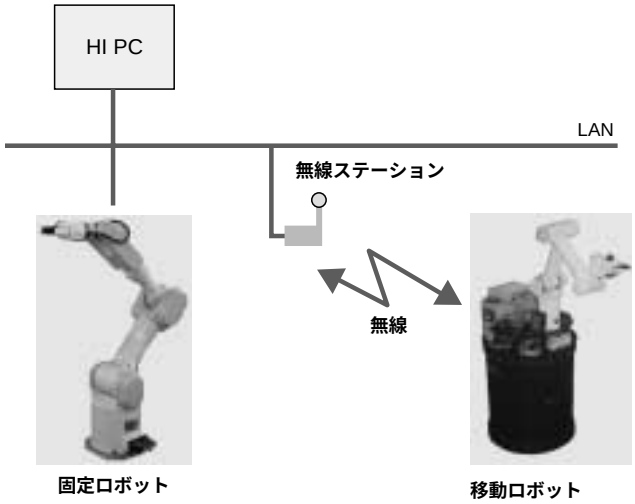

図1．オープンコントローラを採用したシステム　移動ロボットと固定ロボットから成り，これらがHI PCとLAN（有線・無線）で接続し，システムを構成する。
Experimental system using open robot controller framework

コントローラの構成を図2に示す。破線で囲んだ二つの部分がそれぞれ固定ロボット側と移動ロボット側のLAN構成を表している。それら二つの部分は無線ステーションを介して，無線LANで接続されている。また，それぞれの部分で網掛けの大きい四角の囲みが一つのPCあるいはコントローラを表し，その中の小さい四角は機能を表している。以降では大きい四角をサブシステムと呼ぶ。

それぞれのサブシステムは，オープンコントローラの構成法に従って作られている。オープンコントローラの構成法とは，図2の各サブシステム中の機能構成に示すように，最上位にコマンドサーバと呼ぶコマンド受付機能を設け，その下に各処理機能を付加した構成である。各処理機能とはサブシステムが動作する小型アームや移動台車なら軌道生成・サーボ制御の機能であり，サブシステムが画像処理装置なら画像処理機能である。

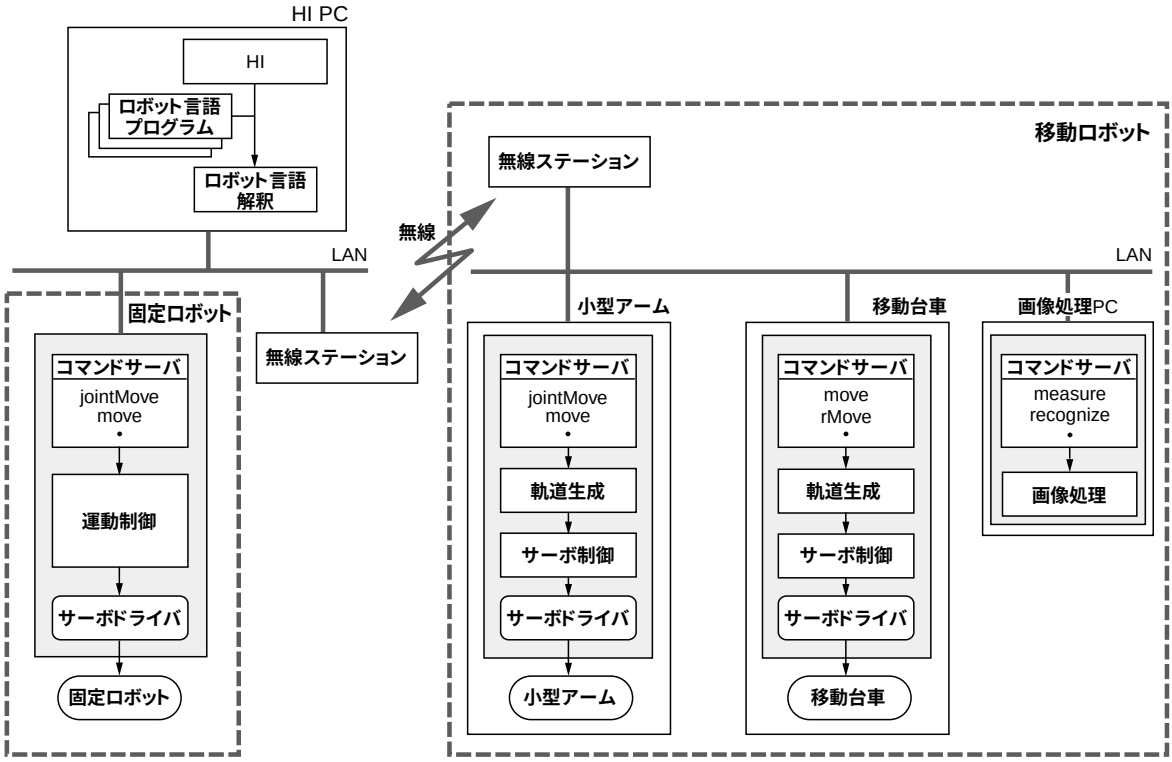

図2．コントローラの構成　固定側としてHI PCと固定ロボットをLANで接続し，移動ロボット内では小型アーム，移動台車，画像処理PCをLANでつないでいる。この二つのLAN間を無線LANでつないだ構成とする。実行時には，HI PCでロボット言語の流れに従って各サブシステム（固定ロボット，小型アーム，移動台車，画像処理）を制御する。
Configuration of experimental open robot controller

各サブシステムに使用しているOSとしては，HI PCにはMicrosoft®Windows®(注1)，画像処理PCにはLinux，左側のロボットにはWindows®，移動台車にはLinux，小型アームには産業技術総合研究所で開発中のリアルタイムLinuxであるART-Linux[5]を用いた。オープンコントローラの性格上，開発環境もフリーでだれにでも使えるものが良いと考えたからである。オープンコントローラに使用できるリアルタイムOSの選択については3.3節で説明する。

HI PC上のソフトウェアはJava(注2)で作成した。いろいろなプラットフォームで走ることが可能である，製品レベルの開発ツールその他が無料で手に入る，言語レベルでスレッドをサポートしている，などのJavaの特長がオープンなロボットコントローラに適していると考えたからである。

3.2 オープンコントローラのソフトウェアインタフェース

オープンコントローラのソフトウェアインタフェースはコマンドサーバ部に実装し，コマンドサーバ部は分散オブジェクト技術を用いて作成した。分散オブジェクト技術とは，遠方に存在するソフトウェアをネットワークを介してあたかもローカルにあるソフトウェアのように使う技術である。この技術を用いてオープンコントローラのインタフェースを作成することで，ロボットのように多種多様なインタフェース（コマンド）を持つシステムに対しても柔軟に対応可能となり，変更・拡張が容易な構成とすることができる。分散オブジェクトを採用することで，作成したコマンド（オブジェクトに対するメソッド）をどこからでも利用できるようになる。

各サブシステムが一つのオブジェクトとなり，サブシステムに対してコマンドを出すことがオブジェクトに対するメッセージ送信に対応する。サブシステムに対するコマンドは，“サブシステム．コマンド（引数）”という形になる。例えば，小型アームに関節角座標系での動作をさせようとするならば“小型アーム．jointMove（目標関節角）”になる。

サブシステム間の通信にはHORB[6]（分散オブジェクト技術のJavaによる実装）を利用しているため，ネットワーク上のソフトウェアオブジェクトがローカルに存在するように通信を意識することなくソフトウェアを作成することができる。実際，HI PCではHI部で受け付けた要求やロボット言語プログラムのコマンドを，ロボット言語解釈部が分散オブジェクト技術を用いて，対応するサブシステムのコマンドを呼び出している。サブシステムのオブジェクトに対するメッセージが，いわゆるロボット言語のコマンドとなっている。

移動ロボット上には三つのサブシステムがあり，これらがネットワークで結ばれている。画像処理PCも含めて分散オブジェクト技術を用いて作成してあるので，インタフェースがオブジェクトに対するメッセージ送信という形で統一的に扱える。ソフトウェアとしては，HI PC上のソフトウェアに移動ロボット上のサブシステムのインタフェースソフトウェア（HORBにより自動生成）を追加するだけでよい。

小型アームや移動台車は，コマンドサーバからサーボ制御までの三つの機能で構成されている。コマンドサーバはHI PCからのコマンドを待ち受け，コマンドがきたら対応する軌道生成を行い，この軌道に沿うようにサーボ制御する。サーボ制御部はサーボドライバに指令値を出し，ロボットを動かす。コマンドとしては，手先の位置や移動先の位置を指定しての移動命令（move）や関節角を指定する移動命令（jointMove）などがある。

3.3 リアルタイムOSの選択について

われわれが提案するオープンロボットコントローラでは，コントローラのCPUでサーボ制御から，HORBに基づく分散オブジェクトまでまかなうことになるので，Javaがサポートできるリアルタイム OSが必要になる。現状では，リアルタイムLinux（RT-Linux[7]やART-Linuxなど）やQNX-RTP[8]，VxWorks[9]などが候補に挙げられる。このシステムではART-Linuxを用いており，CPUパワーを必要とする欠点はあるが，通常のLinuxプログラミングの範囲で扱えるので導入の敷居は低い。一方，RT-LinuxはCPUパワーは要求しないが，カーネル空間でのプログラミングになるため，特殊なシステムコール体系になっており，ソフトウェアのバグによるメモリ保護もされないという欠点がある。これらのリアルタイムLinuxはオープンソースである。QNX-RTPは，最新のPOSIX（UNIX(注3)の規格）に基づくアーキテクチャを持ち，評価だけなら自由に使うことができる。VxWorksは組込分野では大きなシェアを占めるが，高価で閉じたシステムであるためここでは選択肢から外した。

3.4 コントローラの動作

サブシステムはHI PCからのコマンドを受け取り，動作する。そのコマンドは，各サブシステム上のソフトウェアオブジェクトに対するメッセージとして発行する。例えば，移動台車を（x, y）の位置へ速度velで移動させるときには“vehicle.move（x, y, vel）”のように“サブシステム．コマンド（引数）”あるいはオブジェクト指向のことばで書けば“オブジェクト．メソッド（引数）”という形のコマンドになる。vehicle.move（x, y, vel）というコマンドは，HI PCのロボット言語解釈部で分散オブジェクトを利用して解釈され，vehicleに対するコマンドなので無線ステーションを介して移動ロボットに送られる。移動ロボット側で更に移動台車へコマンドを送り，移動台車のコマンドサーバがmoveコマンドに対する軌道生成ルーチンを呼び出す。ここで軌道が生成され，この軌道に沿ってサーボ制御することで移動台車が目標位置へ移動する。

このコマンドは，JavaやC++などのオブジェクト指向汎

（注1） Microsoft，Windowsは，米国Microsoft Corporationの米国及びその他の国における登録商標。

（注2） Javaは，米国SunMicrosystems社の商標。

（注3） UNIXは，商標。

用プログラム言語で開発したAPI（Application Programming Interface）群を，汎用オブジェクト指向スクリプト言語であるPythonから利用する形で提供する。ここでは，Java上のPythonであるJPython[10]を利用し，JavaのAPIを呼び出す。ユーザーは，HI PC上のHIからコマンドあるいはロボット言語プログラムをロボット言語解釈部に渡し，ここで解釈されたコマンドが対応するサブシステムに送信され，ロボットに望みの動作をさせる。HORBを用いているため，ユーザーはネットワーク構成を気にすることなく対象のサブシステム(Javaのオブジェクトとして存在している)に対して，メッセージを送るという形で呼び出すだけでロボット言語APIを起動することができる。分散オブジェクトを使う利点の一つはこれで，通常ならば動作コマンドをHIからサブシステムに送り，サブシステムでこれを解釈して対応する動作を実現する。動作コマンドの追加の場合には，HI側とサブシステムの受け側双方のソフトウェアの修正が必要になる。分散オブジェクトを使えば，新しいコマンドに対応するメソッドを追加するだけでよい。ネットワークを介したソフトウェアの開発でもっともバグが出やすい送受信部を代行してくれるので，開発効率が上がる。

Pythonで書いたロボット言語の一例を図3に示す。Java側でHORBへの接続部分が書いてあるのでPythonからはロボットを初期化（図の最初の4行に示すような宣言）するだけで，以降はまったくネットワークを気にせずに，初期化されたロボットオブジェクトに対してメッセージを送るという形で動作を記述できる。

図の5行目fixedArm.move（・・・）から下が動作プログラムである。fixedArm.move（x1, y1, z1, a1, b1, c1, v1）は，固定ロボットオブジェクトに対して位置（x1, y1, z1, a1, b1, c1）へ速度v1で移動せよというメッセージを送っている。次に，smallArm.move（・・・）は小型アームに対しての移動命令である。次のvehicle.move（x3, y3, v3）は，位置（x3, y3）へ速度v3で移動車を移動させる命令である。その下のfor文は，Pythonのループ文で3回の繰返しを指定している。ここでは，vehicle.rMove（・・・）という移動台車の相対移動命令の後，objで指定する対象物までの距離を画像計測し，その結果が100 mm以下であればループを抜ける。そうでなければ，ループの先頭に戻って繰り返す。

```
fixedArm = FixedArm()              #固定ロボットの初期化
smallArm = SmallArm()              #小型アームの初期化
vehicle = Vehicle()                #移動台車の初期化
image = Image()                    #画像処理装置の初期化

fixedArm.move(x1,y1,z1,a1,b1,c1,v1)        # fixedArm手先移動
smallArm.move(x2,y2,z2,a2,b2,c2,v2)        # smallArm手先移動
vehicle.move(x3,y3,v3)                     # vehicle移動
for i in range(0, 3):                      # 3回繰返し
    vehicle.rMove(x4,y4,v4)            # vehicle相対移動
    ans = image.measure(obj)           #画像による計測
    if ans < 100 :                     #計測結果による条件分岐
        break
```

図3．ロボットプログラムの例　JPythonを用いたロボットプログラムの例を示し，JavaとHORBを用いて定義したロボットオブジェクトにメッセージを送る形で使用する。
Example of robot program

4 あとがき

分散オブジェクト技術を用いることで，簡単にネットワークを介して接続できるオープンコントローラの枠組みを開発し，適用例を通してその有効性を確かめた。今後もいろいろなコントローラに適用し，オープンコントローラの有効性を実証していく。なお，このシステムは1998年度NEDO（新エネルギー・産業技術総合開発機構）即効型提案公募の資産を利用している。

文　献

(1) CNCオープン化はここまで来た．精密工学会第261回講習会テキスト．2000.
(2) 特集 ロボットシステムのオープン化．ロボット．日本ロボット工業会, No.121, 1998.
(3) 大明準治, ほか．“ロボットのパソコンコントローラ”，日本ロボット学会第59回講習会テキスト．1999, p.9－20.
(4) 大明準治, ほか．“リアルタイムLinuxと分散オブジェクト技術を用いたオープン指向PCコントローラ”．ロボティクス・メカトロニクス講演会．2000.
(5) ART-Linux, http://www.movingeye.co.jp/~you1/art-linux/download.htm
(6) HORB, http://www.horb.org/horb-j/
(7) RT-Linux, http://www.rtlinux.org/
(8) QNX-RTP, http://get.qnx.com/
(9) VxWorks, http://www.windriver.com/
(10) JPython, http://www.jpython.org/

尾崎　文夫　OZAKI Fumio
研究開発センター　機械・システムラボラトリー主任研究員。ロボットのソフトウェアシステムの研究・開発に従事。日本ロボット学会会員。
Mechanical Systems Lab.

大明　準治　OAKI Junji
研究開発センター　機械・システムラボラトリー研究主務。ロボット制御の研究・開発に従事。日本ロボット学会，計測自動制御学会会員。
Mechanical Systems Lab.