移動ロボット知覚制御用RTC群の開発と学生実験での利用

アッタミミ ムハンマド，中村友昭，○長井隆行 (電気通信大学)

Development of RT Components for Perceptual Control of Mobile Robots and Its Application to a Student Laboratory

Muhammad Attamimi, Tomoaki Nakamura, ○ Takayuki Nagai (UEC-Tokyo)

Abstract– We developed RT components for perceptual control of Mobile Robots using the RTM.NET. The developed system consists of the hardware RTCs, the recognition RTCs, and the task RTCs. The system makes it easy for students who have little experience of computer programing to implement intelligent robots. In fact, we use the system for the student laboratory in UEC.

1 はじめに

ロボットは非常に複雑なシステムであり，そこで使われている技術が多岐に渡ることは周知の事実である．特に，近年研究開発されているヒューマノイドロボットや家庭用サービスロボットは，多種多様なセンサによるセンシングやその信号処理，処理結果によって得られる情報を用いたエンドエフェクタの制御，さらには情報の記録や学習など様々なサブシステムが複雑に統合されている．従ってこうしたロボットの研究開発には，ハードウェアや個々のサブシステムはもちろんのこと，それら全体をどのように統合しシステムとして組み上げるかというシステムインテグレーションについての知識や経験が非常に重要である．このシステムインテグレーションを容易に行うための枠組みとして，ミドルウェアという考え方が有用であることは言うまでもないであろう．これは，各サブシステムをモジュール化し，入出力の仕様を定義することでサブシステム間をネットワーク接続する．これによって全体を動作させることができるため，複雑なシステムの開発や統合を容易にする仕組みであると言える．ロボットシステムのためのミドルウェアとしては，RT ミドルウェア (RTM) や ROS などが存在するが，特に RTM は産総研が中心となって進めている日本発の技術であり，これを広めることには大きな意義があろう．

しかしながら，簡単に知能ロボットを作れるほどに整備されているロボットの知覚制御用パッケージは，著者の知る限り存在しない．例えば，OpenINVENT と呼ばれる自律移動制御を行うソフトウエアモジュール群が公開されているが，画像認識や音声認識などは含まれていない．RTM はその柔軟さ故に，使用する言語を始め導入の際の選択肢が多くそうした意味でも敷居が高いのではないかと思われる．RTM のホームページ [1] では，ユーザが作成した RTC が多数公開されており非常に有用であるが，こうしたリポジトリの一般的な問題として，自分で使えるパーツを探す必要がありこれが煩雑となる．特に初心者にとってこれは簡単な作業ではない．

こうしたことから，我々は導入のための敷居が低いことを考慮した，移動ロボット知覚制御用 RTC 群の開発を行った．ポイントは，以下の 4 点である．1)RTM の導入も含めて全てがそろっている（1 つの HP を参照すればすべてが使える），2) 雛形プログラムを修正するだけで知能ロボット（らしきもの）を実現できる（中央制御部の雛形），3)C/C++の知識（プラスアルファ）で作ることができる，4) 必要に応じて複雑なこともできる．

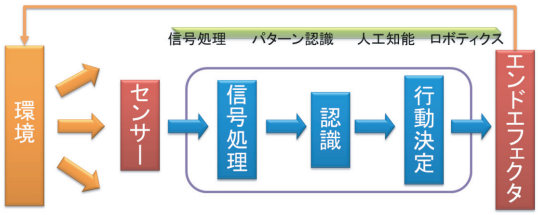

Fig. 1: The simplest model of the intelligent robots

このシステムを作るきっかけは，学部 3 年生の学生実験 [2] であった．実験では，移動ロボットの知覚制御を題材に，センサの処理，ロボットの移動制御から RT ミドルウェアを使った統合までを体験することで，実際の自律ロボットシステム開発について学ぶことを目的としている．本稿では、構築した知覚制御用 RT コンポネント群について述べ，そのシステムを実際に学生実験に導入した例を紹介する．

2 知覚制御用 RTC 群

ここでは，自律移動ロボットの知覚制御用 RT コンポネント群の開発方針と，実際に開発したシステムの詳細について述べる．

2.1 構想

本システムで実現したい，自律型知能ロボットのモデルを図 1 に示す．これは知能ロボットとしては最も単純なモデルであり，観測と行動のループから構成されている．しかし，ロボット全体として正常に機能させるためには，全ての要素を誤りなく動作させる必要がありこれを一人でゼロから開発するには膨大な時間を要する．特に信号処理やパターン認識は理論的背景を押さえていない限り実装することは困難であろう．一方で行動決定部は，処理結果を受けてロボットとしての行動を決定する部分であり，ロボットの振る舞いを大きく変えるものであるにも関わらず，単純なものであれば最も作るのが容易な部分であると言える．そこで本システムの基本方針は，ハードウェアやセンシング，その処理に関する部品をなるべく多く用意し，それらをつなぎ合わせることで基本的な部分を作成し，最も上位となる行動決定部を自由に作ることでシステム作成者の考える自律ロボットを実現するというものである．その際，行動決定部のサンプルも多く用意することで，それらを参考にしながら，もしくはコードを

PortDefine.h

```
#ifndef MODULE_NAME
// モジュールの名前を定義
#define  MODULE_NAME "DiGOROTest"
#endif

// 入力ポートの定義
// INPORT( type , name )
// type : 送信するデータ型
// name : 名前
INPORT( TimedLong^ , InData )
// 出力ポートの定義
// OUTPORT( type , name )
// type : 送信するデータ型
// name : 名前
OUTPORT( TimedLong^ , OutData )
```

Fig. 2: Definition of the data port

書き換えることで実装することがえきるように配慮する．また全てのパーツは同様のひな形プログラムから作ることで，処理に関する部分を改変したい場合にも対応できるようにする．

2.2 設計と実装

RT ミドルウェアの実装にはいくつかの種類が存在するが，我々が試した中で最も使いやすいと感じたものは，株式会社セックが開発している OpenRTM.NET である[3]．OpenRTM.NET は，Microsoft .NET Framework 上で動作する RT ミドルウェア実装であり，IDL を記述する必要がなく，C#や Visual Basic などのプログラミング言語を利用してサービスポートのインタフェースやデータポートで送信するデータ型を定義することができるため，非常に使い勝手が良い．ただし，C++はそのままでは使用できないため，ここでは C++/CLI を用いたひな形コンポーネントを実装し，それをベースにすることで非常に簡単にコンポーネントの開発を行えるようにした．

当初は，GUI を使うことを想定し MFC を使用したひな形プログラムを作成した．しかし，後で述べる学生実験で使用した際に GUI の概念を理解しておらず（マルチスレッドの概念を理解する必要がある），プログラムの構造がわかりにくいということが多く見受けられたため，GUI 版のひな形プログラムとコンソール版のひな形プログラムの 2 種類を用意することとした．特にコンソール版のプログラムは main 関数が明確であり，プログラムの構造が容易に把握できる．また printf で値を表示できるために，デバッグをしやすいという利点もある．何れのバージョンも基本的なプログラムの構造は同じである．我々がひな形プログラムにおいて最もこだわったのは，データポートを利用した送受信の定義をいかに簡単に記述できるかという点である．具体的には，マクロを用いることで非常に簡単に記述可能としている．Fig. 2 に具体例を示す．情報は，「PortDefine.h」というヘッダファイルに記述することになっており，この例では入力ポートは InData と命名され，TimedLong 型のデータを受信することを意味している．また，出力ポートは OutData と命名されており，TimedLong 型のデータが出力される． 一方でポートを使ってデータを送受信する部分は，多くの場合 main 関数内に記述されることになる．main 関数は，「DiGOROClientRTC.cpp」内に記述されており，

DiGOROClientRTC.cpp

```
TimedLong ^inData;  //受信データ用メモリ
TimedLong ^outData; //送信データ用メモリ
 ･･･ 途中省略 ･･･
//受信用関数名は Get + [変数名]
//ただし [変数名] は PortDefine.h で定義
//データの受信 (データがあれば戻り値が true)
if ( m_component->GetInData( inData ) ) {
    //受信したデータに 100 を足して出力する
    outData->Data = inData->Data + 100;
    //本当はここで現在時刻を代入する
    //ここでデータ送信
    m_component->SendOutData( outData );
    //送信用の関数名は、Send + [変数名]
}
```

Fig. 3: A usage example of the data port

先ほどのデータポートの利用は Fig.3 のようになる．

また，コンフィギュレーションやサービスポートに関しても，ひな形プログラムからできる限り簡単に利用できるように作られている．ただし，データポートを用いた送受信さえできれば多くの場合コンポーネントを作ることができるため，特に初心者の場合にはサービスポートなどを使う必要性はあまりないと思われる．ひな形プログラムは，VC++のソリューションとして提供するが，実際にソースコードを書き換える必要があるのは上述の「DiGOROClientRTC.cpp」と「PortDefine.h」だけであり，通信の仕様さえ決まってしまえば，「DiGOROClientRTC.cpp」だけを考えればよい．こうした点も，プログラミングを容易にしているポイントである．Tab.1 には，作成した全てのコンポネントの一覧を示した．これらは大きく，デバイス制御 RTC，処理・認識 RTC，タスク RTC，その他の RTC に分けることができる．デバイス制御に関しては，その名の通りハードウェアを制御するためのものである．ここでは後で述べる実験で，ER1 をロボットとして使用する必要があったため，ロボットに関しては ER1 を用いることが前提となっている．ただし直進速度と回転速度をシリアル通信で与えるだけのプログラムであるため，比較的容易に他のロボットに合わせて改変することができると思われる．また，Kinect としては ASUS の Xtion PRO を使う仕様になっている．画像処理関連に関しては OpeCV を使用しており，音声認識は Julius，音声合成には AquesTalk を用いている．OpenCV に関しては，DLL をダウンロードしパスを通すか，実行ファイルのフォルダにコピーする必要がある．音声認識・合成に関しては全て実行ファイルフォルダに含まれている．タスク RTC は，行動決定部に相当するサンプルプログラムであり，音声認識や画像認識，ジェスチャ認識など，処理・認識 RTC からの情報を応用したサンプルを一通りそろえている．その他の RTC にはひな形プログラムが含まれており，RTC をゼロから作りたい場合，比較的容易に作成できるよう配慮されている．

2.3 使用例

ここでは，開発した RTC 群を使って実際に構築したシステムの例を示す．

2.3.1 音声認識を使ったタスク

音声認識結果を用いてロボットを制御する最も簡単なタスクプログラム（行動決定部）であり，サンプル

Table 1: List of RT components containing in the system; all of these are available at [2)]

デバイス制御 RTC	処理・認識 RTC	タスク RTC	その他の RTC
ロボットの制御	音声認識	音声認識を使ったタスク	RTC を作るための雛形（コンソール）
ゲームコントローラ使用	顔検出	Kinect を使ったタスク	RTC を作るための雛形（GUI）
Kinect からの情報取得	物体認識（検出）	ジェスチャ認識タスク	文字列送信 RTC(テスト用)
–	–	物体認識タスク	–

DiGOROClientRTC.cpp

```
//音声認識の結果を受け取る
if( cmp->GetSpeechRecAudio(datSpRec, -1) )
{
    //受信した文字列のチェック
    str = datSpRec->Data;
    char *p = new char[str.GetLength()+1];
    strcpy(p,str);
    printf("認識文:%s\n",p);

    //条件分岐
    //文字列の比較関数 strcmp の使い方に注意
    //認識結果が"スタート"だったら
    if(!strcmp("スタート", str)) {
        //ロボットの発話
        dataSpeechUtt->Data
            = _T("タスクを実行します");
        //発話命令送信
        cmp->SendTalkAudio(dataSpeechUtt);
        //ここでタスクを実行させる
        //タスク実行関数の呼び出し
        taskExec(cmp);
    //認識結果が"すすめ"だったら
    }else if(!strcmp("すすめ", str )){
        ....
```

Fig. 4: A part of the main function in the task RTC

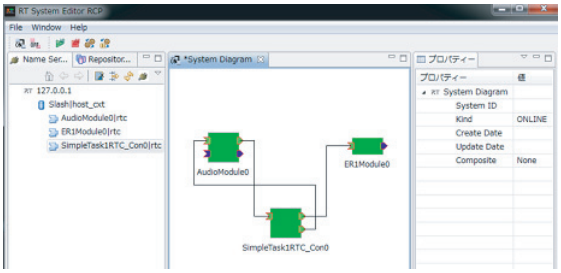

Fig. 5: Configuration of the speech recognition task

として提供しているものである．音声認識 RTC からの文字列を受信し，その文字列によってロボットの直進・回転速度を分岐する．従って出力ポートからは，ロボットの制御 RTC に速度情報を渡すように作られている．またロボットが移動する際には，行動に応じた発話をするために，発話内容の文字列を音声認識 RTC（音声認識 RTC は認識と合成の両方を担当している）に送信する．Fig.4 にコードの一部を，Fig.5 に RTC 接続の様子を示す．

2.3.2 物体認識タスク

物体認識タスク RTC は，事前に登録した物体を検出し，その結果に応じてロボットの移動を制御するものである．従って，物体認識 RTC を使用して物体認識を行う必要がある．物体認識 RTC への物体画像の登録は，画像ファイルを特定のフォルダにコピーするだけであるため容易である．提供しているサンプルには，二つの異なる箱が登録されている．タスク側は，まずロボットを前進させ，二つの物体の内どちらか一方でも認識されたという情報を受信すると，停止してどの物体を認識したのかを発話するようになっている．Fig.6 にタスク実行時のシステムエディタ画面のスクリーンショットを示す．

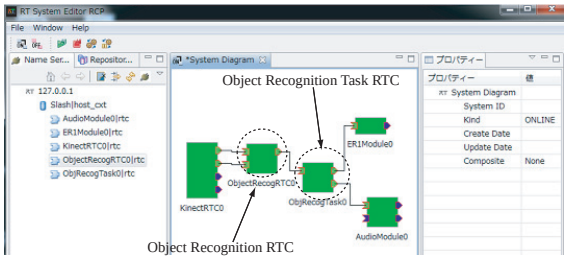

Fig. 6: Configuration of the object recognition task

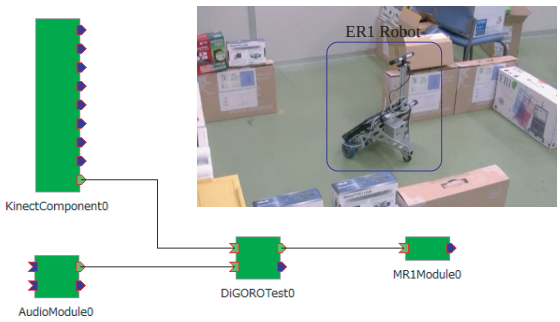

Fig. 7: Example of finding a way out of the maze task

2.3.3 迷路脱出

Kinect の距離情報を使ってロボットの移動を制御し，迷路から抜け出すタスク RTC である．処理の内容としては，キネクトからの距離画像をいくつかの領域に分割し，その領域の距離のパターンによってロボットの直進・回転速度を分岐させるものである．基本的には，最も開けている方向を探索しその方向に進むというアイディアである．また，音声認識コマンドを利用できるため，迷路にはまった際に人が方向を指示できるようにもなっている．Fig.7 にシステムの構成と実行時の様子を示す．

2.3.4 その他

Tab.1 のタスク RTC に示すように，上記の他にも Kinect のボーントラッキング機能を利用したジェスチャ認識などのタスク RTC をサンプルとして用意している．これらの RTC は全てひな形プログラムをベースに作られており，基本構造が分かってしまえばどのプログラムも比較的簡単に内容を把握できる．ここで示したタスク RTC は，サンプルとして提供するための非常に基本的なものである．アイディア次第ではより複雑で面白いロボットを作ることができ，実際後で紹介す

Fig. 8: Scenery of the lab

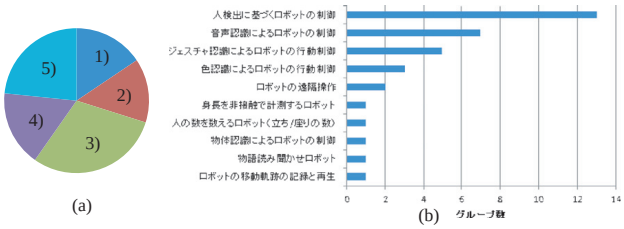

Fig. 9: Some results of the lab; (a) questionnaire results and (b) developed RTCs

る学生実験では，非常に面白いシステムを構築している例が見受けられる．

3　学生実験への適用

本システムは学生実験を一つのターゲットとして構築したものである．具体的には，電気通信大学知能機械工学科 3 年生の必修科目である知能機械工学実験の一つのテーマとして実施しており，今年で 2 年目を迎えている．ここでは，前節で述べた RTC 群を利用した学生実験について具体例を挙げながら概観する．

3.1　実験の方針とスケジュール

学生実験は，3 時間 ×2 日の計 6 時間であり，原則二人一組のグループで行う (これは，機材や実験室の広さといった制約によるものである)．実験の目的は，自律ロボットの構築に必要な要素技術やそれらを統合する仕組みを，実際のプログラミングを通して学習（体感）することである．さらに我々が重視したのは，課題を与えるのではなくそれ自体を自分たちで考えるプロセスを通して，発想することの楽しさや重要さを再認識させ，それを具現化するにはどうすればよいか，について考える機会を作ることである．

3.2　実験の様子

Fig.8 に学生実験の様子を示す．本実験は非常に自由度が高く，自分達でゴールを決める必要があるため，そのことを意識して実験初回の説明も非常に真剣に聞いている様子がうかがえる．また，他のテーマは基本的に与えられたことを実践することが多いため，その新鮮さ故か受講生のモチベーションが高いことを感じることができる．一班は 2〜3 名で構成されるために，自然にチーム内での役割分担ができていることも見受けられる．TA とのやり取りが非常に盛んなのも本実験の特徴であるように思われる．これには，プログラム自体の質問からアルゴリズムに関する議論まで様々なものが含まれる．

3.3　実験の評価

2013 年 9 月現在，延べ 180 名ほどの学生が既にこの実験を受講している．実験では，最終的にレポートを提出しその内容について教員と議論することを義務付けている．現在まで約 90 グループが実験を行っており，実験がうまくいかなかった事例やレポートを提出しなかった事例は発生していない．レポート提出時には，できれば実験に関する感想や意見を書くように指導しており，現在 95 件の回答が集まっている．内容は多岐にわたるが，多くの場合は，プログラムが難しかったが実験自体は楽しかったというものである．

内容を，1）自分の勉強不足を自省するもの，2）もっとやりたかった、3）実験の仕組みがよい、4）RTM や使ったシステムがよい，5）意見（単に難しいというものを含む）の五つに分類した結果を Fig.9 (a) に示す．全体的にはポジティブな意見が多く，RTM やシステムが優れているという意見も比較的多く見られた．例えば,「短時間で音声認識・画像認識・ロボット制御を簡単に行えたので実験教材を精査していると感じました.」という感想があり，これは我々が目指す方向性が素直に評価されているものであろう．実験に関するネガティブな意見としては，時間やＴＡが足りないというものが多い．一方，システムに関する改善の意見はあまり多くないが，プログラミングが苦手であまり理解できないといった感想が多くみられた．これに関しての対応は難しいが，中身がよくわからなくても「なんとなく接続して見よう見まねで作ったら動いてしまった！」というのもミドルウェアの醍醐味なのかもしれない．

実際に学生が構築したシステムで興味深いものには，次のようなものがあった；ジェスチャ認識（サンプルプログラムの基となったもの），おにごっこ（複数班での協力），HMD を利用した遠隔ロボット，人狼ゲーム，だるまさんが転んだ，身長計測ロボット，などである．Fig.9 (b) は昨年半年間で作成されたものを大まかに分類した結果である．このように，アイディア次第で色々なシステムを短時間で作れるのは，開発したシステムによるところが大きいと思われる．

4　まとめ

本稿では，自律ロボットの知覚制御を目的とした RTC 群の開発を行った．これは，環境構築からロボットへの実装までをできる限り全て含む形でパッケージ化することを目標としたものであり，全て無料で利用できる．学部学生実験のシステムとして現在も利用されており，今後も様々な意見に基づき改良していく予定である．実機での動作を目指していたため，現状ではシミュレータとの連携がないが，サイバーフィジカルな環境を構築するための優れたシミュレータである SIGVerse[4] との接続モジュールは既に作成済みであり，公開の準備を行っているところである．

参考文献

1) Open-RTM ホームページ: http://www.openrtm. org/ openrtm/ ja/content/openrtm-aist-official-website

2) 電気通信大学知能機械工学科知能機械工学実験, RT ミドルウェアを用いた移動ロボットの知覚制御実験ホームページ: http://apple.ee.uec.ac.jp/ROBOTEXP/

3) 株式会社セック　OpenRTM.NET, http://www.sec.co.jp /robot/download_rtm.html

4) 稲邑, ”社会的知能研究のためのシミュレーションプラットフォーム:SIGVerse”, 日本ロボット学会誌, 31(3), pp.240-243, Apr.2013