RTMSafety のシステム構成を以下に示す。

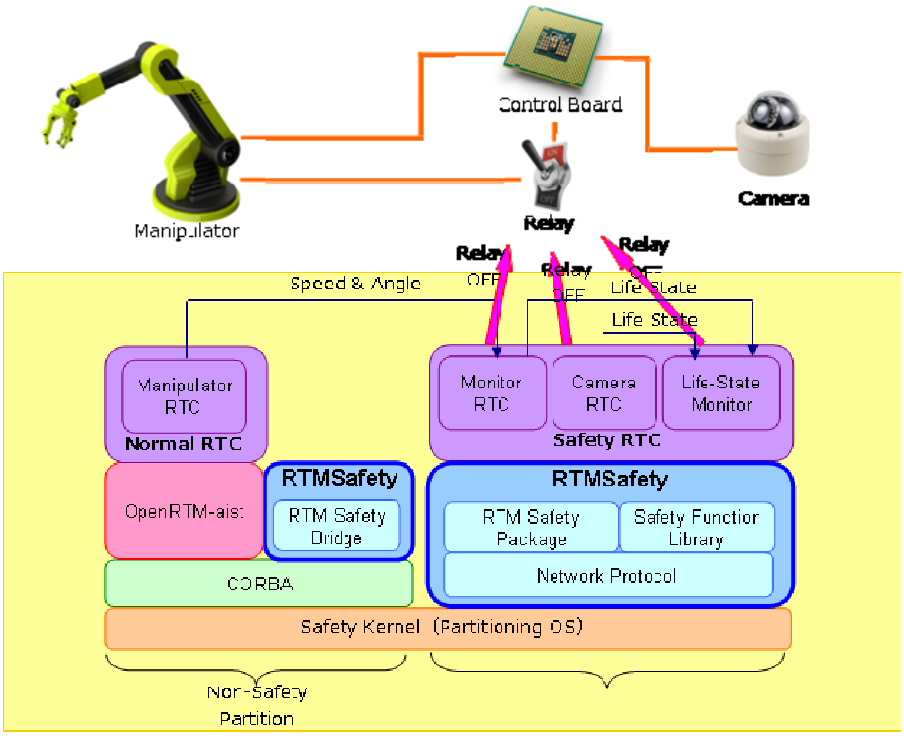


図 95　RTMSafety のシステム構成

RTMSafety は、OMG の RTC 仕様の Lightweight RTC に準拠した RT ミドルウェアである。この RT ミドルウェアの基本機能は RTMSafety Package により提供している。さらに、機能安全のための機能として、RTC の生存状況を監視する機能や CPU 負荷を均一化し、リアルタイム性を保証するフレームワークを Safety Function Library として提供している。

ここで、先に述べたとおり、機能安全なソフトウェアモジュールを開発するのはコストがかかるため、機能安全を保証する箇所は、最小限に留めることが望ましい。そこで我々は、機能安全が必要なミッションクリティカルな部分（安全関連系）と、従来までの RT ミドルウェアで構築した知能モジュール群（非安全関連系）を組み合わせてサービスロボットを構成し、それらが協調して動作するモデルを提案する。そのため、安全関連系と非安全関連系で通信が行えるよう、OMG の GIOP 仕様の CDR 準拠の軽量通信プロトコルを実装し、様々なネットワークプロトコルに対応可能な機能を Network Protocol として提供している。また、RTM Safety Bridge を利用することで、OpenRTM-aist とシームレスに連携することも可能である。

RTMSafety では、ロボットシステムを安全関連系と非安全関連系に分け、それぞれを独立した領域（OS のパーティションなど）で実行することにより、非安全関

連系が安全関連系に影響を及ぼさないよう考慮している。また、OMG の RTC 仕様に基づくコンポーネント指向を採用していることで、安全関連系のコンポーネント（モジュール）間の依存関係はデータの授受だけであり、その仲介も RTMSafety が担うことで、機能安全モジュールの独立性も保証されている。これらの特徴により、機能安全対応のロボット開発におけるコストダウンや機能安全モジュールの再利用性を実現している。

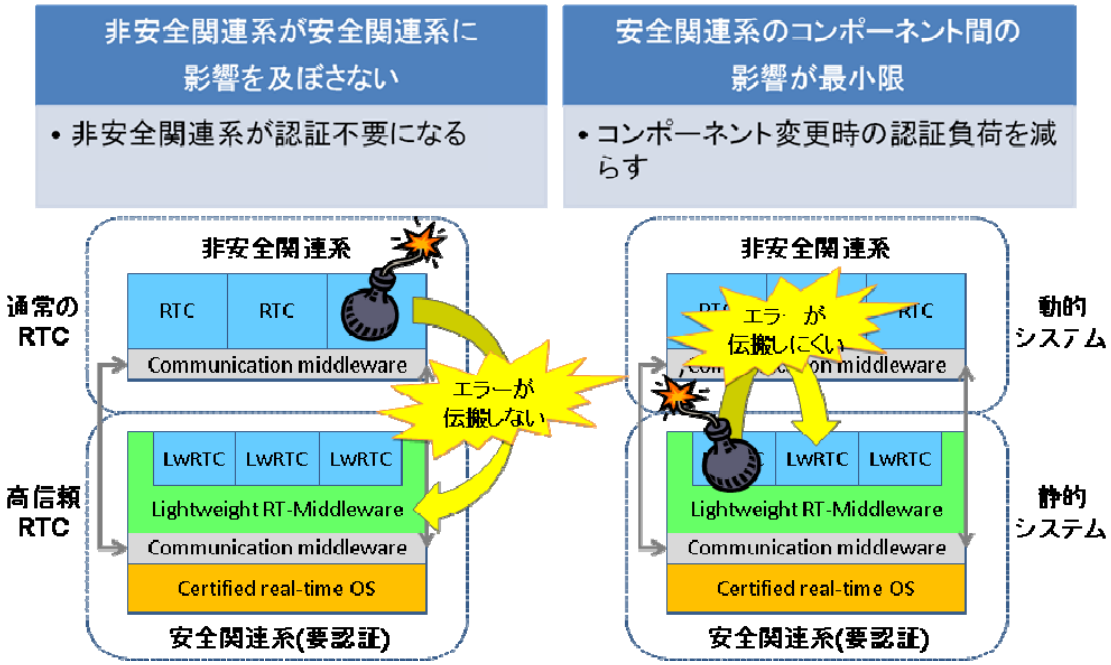


図 96　RTMSafety の構成とメリット

まとめ

RTMSafety は､機能安全に対応した OS である μITRON と QNX の 2 つのプラットフォームで動作するものを開発した。さらに、今後のプラットフォーム拡充のため、VxWorks への移植性も評価し、移植が可能なことを確認した。

本研究項目は、当初の実施計画にはなかったが、生活環境で動作するロボットにとって安全認証は必要不可欠であることが明らかになり実施したものである。

RTMSafety は、認証取得後に、世界初の安全コンセプトをもったロボット用ミドルウェアとして、セックより製品販売を計画している。

② 応用ソフトウェア開発支援機能

(a) 作業シナリオ設計ツールの開発

ロボットに組み込まれるアプリケーションロジックの実現を容易化するための仕組みとして、下記の 3 系統の作業シナリオ設計ツールを研究開発した。3 つのツールと実行系とも、OpenRTM-Aist1.0.0 Java 版に準拠し、すべてのコードを

Java6.0(JRE6.0)によって実装し、特にツール部分については本事業で実現される他の開発ツールとの連携できるように Eclipse プラグインとして実装した。本研究項目の最終目標は、後述の 3 つのシナリオ設計ツールとそれらの実行系を開発することであった。

(a-1)長期時間駆動型シナリオ設計ツール

長時間の役務に従事するロボットへの適用を想定し、ロボットの役務の実行スケジュールを、主として分単位以上の時間の粒度で記述する作業シナリオを開発するためのツールを開発した。具体的には、後述する短期時間駆動型シナリオ設計ツールで作成された短時間のシナリオを画面の時間軸上に部品としてレイアウトしこれを連結して長時間のシナリオを出力するように実現した。

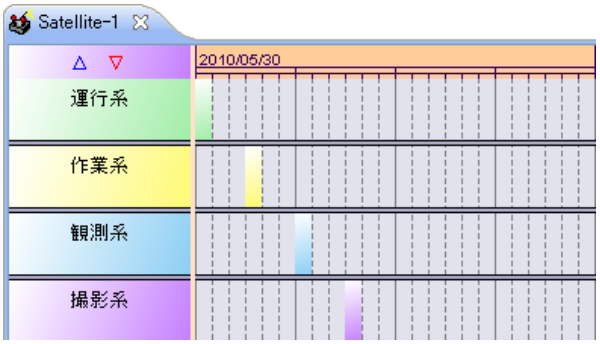


図 97　長時間駆動型シナリオ設計ツールの画面

画面上は縦軸にロボットの部位、横軸に時間が配置される。任意の位置を指示し、後述の短期時間駆動型シナリオ設計ツールで作成した短期時間シナリオを格納したファイル名を指定すると、同ファイルで記述された短期時間シナリオを表すボックスが表示される。これを繰り返し、長期時間シナリオを作成していく。最後に結果のエクスポートのコマンドを選ぶと、全ての短期時間シナリオが指定された時間順に連結された長期時間シナリオファイルが生成される。上記の図では一目盛りあたり 1 時間を表しているが、時間の刻みはカスタマイズすることも可能である。

(a-2)イベント駆動型シナリオ設計ツール

ほとんどのロボットへの適用を想定し、他の RT コンポーネントから出力されるイベント(非同期通知)を受信し、そのイベントへの反応として、別の RT コンポーネントのサービスを呼び出す作業シナリオを開発するためのイベント駆動型シナリオ開発ツールと、そこで開発された作業シナリオをインタプリタ形式で実行することができるイベント駆動型シナリオ実行 RT コンポーネントを開発し、EPL ライセンスに準拠したオープンソースとして一般公開した。また、詳細な英文ドキュメント(200 ページ)を作成し、公開パッケージに同梱した。

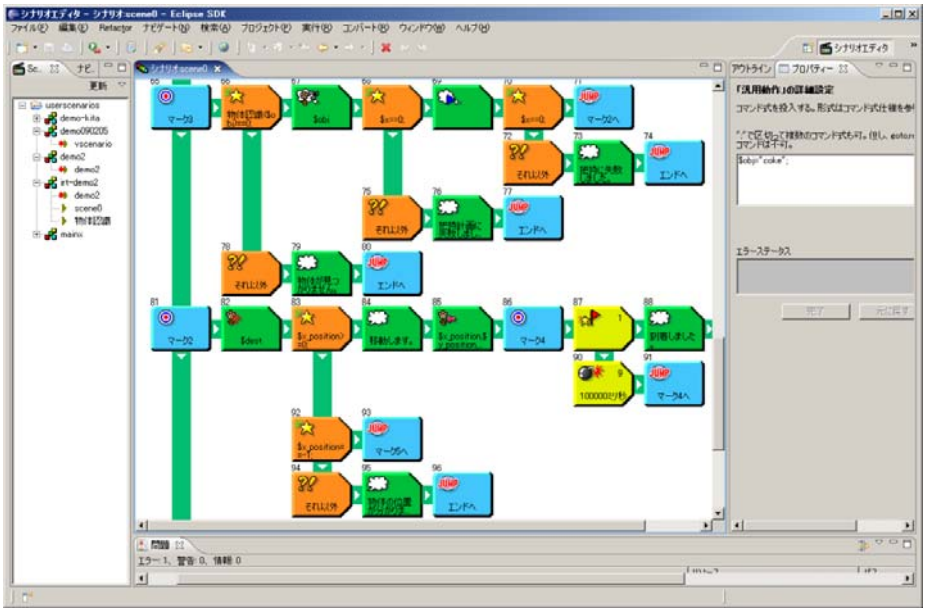


図 98　イベント駆動型シナリオ設計ツールの画面

イベント駆動型シナリオでは、ビジュアル部品と呼ぶ箱をビルディングブロックのメタファを用いて、シナリオをフローチャートのように組み立てていくことができる。ビジュアル部品の追加は下記のように追加する場所を選んでメニューを開き、メニューに表示されたビジュアル部品の種別を選択し引数を入力することで行うことができる。

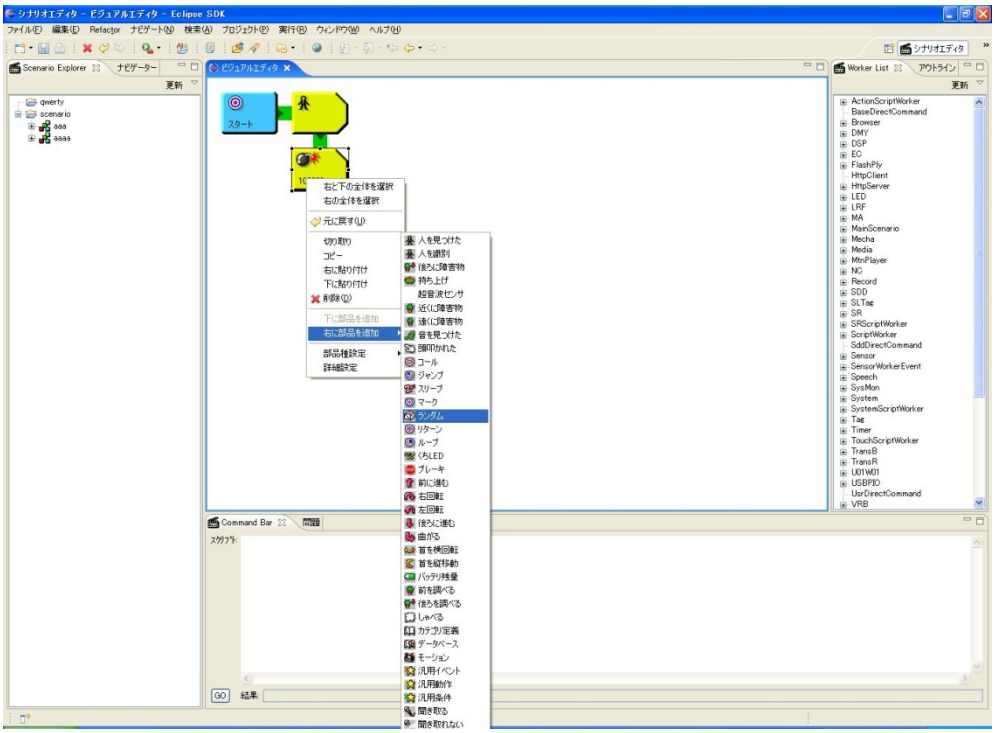


図 99　GUI によるビジュアル部品の追加

ビジュアル部品には 4 つの種類がある。

(1)　動作部品

ロボットに何かのアクションをさせるための部品である。水平方向に付け加えていくことができる。部品の色はミドリで示している。

(2)　動的条件部品

ロボットに外部イベントを判断させたいときに使う部品である。垂直方向に付け加えていくことができる。部品の色はきいろで示している。

(3) 静的条件部品

ロボットにイベントでない条件を判断させたいときに使う部品である。垂直方向に付け加えていくことができる。部品の色はきみどりで示している。

(4) ジャンプ部品

マーク部品まで処理の流れをジャンプさせたいときに使う。水平方向に付け加えていくことができる。部品の色はみずいろで示している。

(5) マーク部品

ジャンプ部品の飛び先を定義するための部品である。水平方向に付け加えていくことができる。部品の色はみずいろで示している。

このうち、動作部品、静的条件部品、動的条件部品の 3 種類は本ツールを用いてシ

ナリオを作成するロボットに合わせてカスタマイズすることができる。以下はカスタマイズを簡単化するためのカスタマイズツールの画面である。このカスタマイズツールはイベント駆動型シナリオ設計ツール内のコマンドから随時呼び出すことができる。

図 100 ビジュアル部品カスタマイズツールの画面

また、本ツールによって作成されるイベント駆動型シナリオを入力し、その記載にしたがって、他の RT コンポーネントを使役し、ロボット全体の動きを生成していくシナリオプレーヤを RT コンポーネントの形で開発した。その構成を RT システムビルダの画面で示す。

シナリオプレーヤ RT コンポーネントは、1 つの入力ポートと N 個の出力ポート(図では 10 個。コードの変更により増減できる)を持つ。出力ポートにはシナリオプレーヤが使役する 1 つの RT コンポーネントがコマンドとイベントを受け付ける入力ポートを結線する。入力ポートには、シナリオプレーヤが使役する全ての RT コンポーネントがコマンド、イベント、レスポンスを出力する出力ポートを結線する。

シナリオプレーヤ RT コンポーネントは、実行するシナリオファイルの記載に従い、コマンドないしイベントを出力ポートを通じて、使役対象の RT コンポーネントに伝え、結果として使役対象の RT コンポーネントから戻ってくるレスポンスなどに従って、シナリオファイルに記載されたフローチャートを辿っていくという動作を行う。

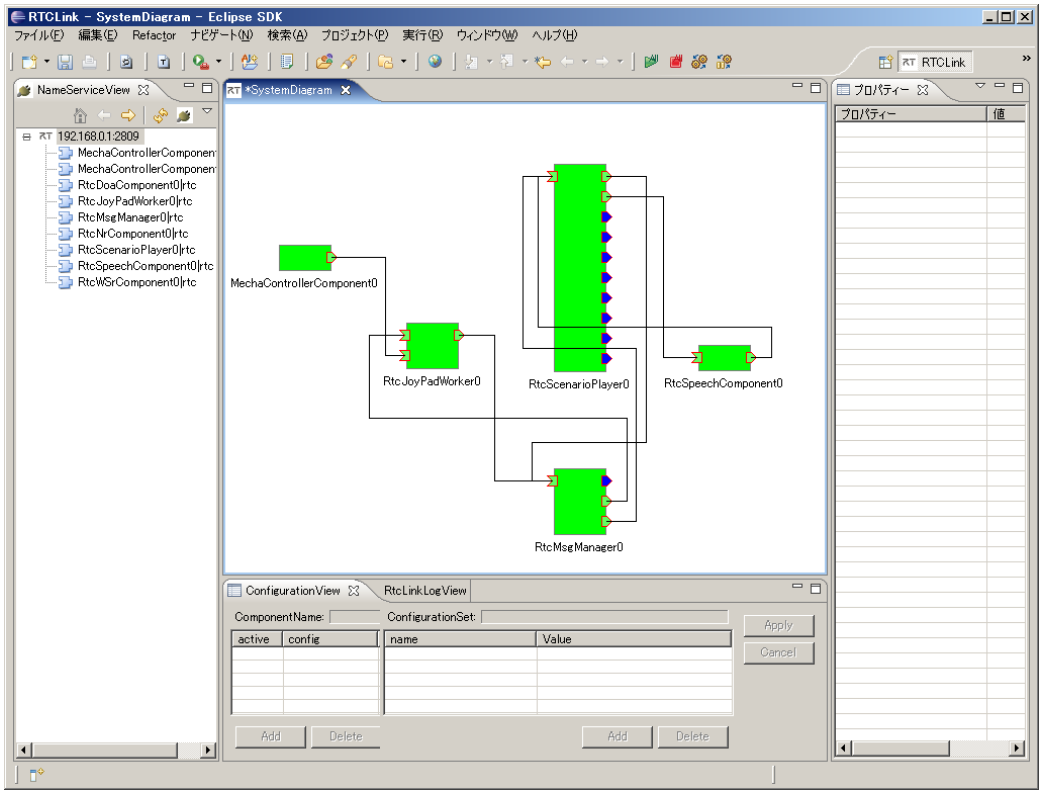


図 101 シナリオプレーヤ RT コンポーネントの結線例

(a-3)短期時間駆動型シナリオ設計ツール

仕草などロボットの複数の部位を連動させる必要のあるロボットに適用されることを想定し、主として分単位以下の時間の粒度で、ロボットが有する複数の動作手段(手や足の動き、発話など)の同期・組み合わせを記述する作業シナリオを開発するための短期時間駆動型シナリオ設計ツールを開発した。

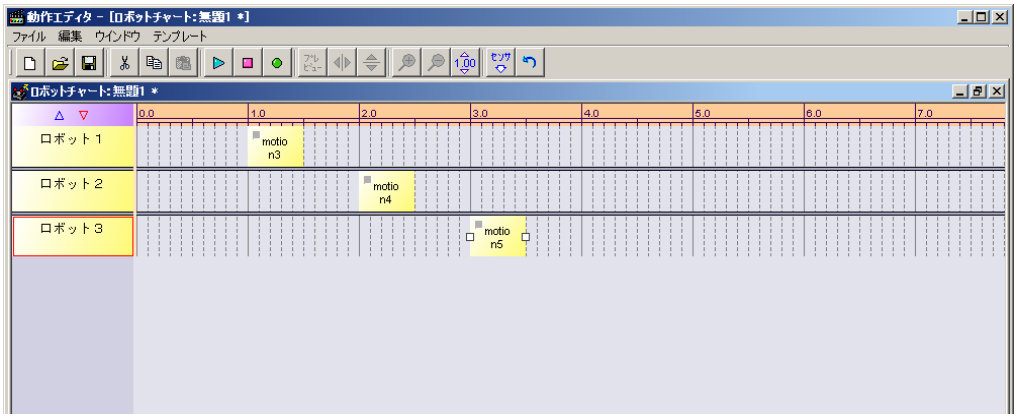


図 102 短期時間駆動型シナリオ設計ツールの画面

画面上は縦軸にロボットの部位、横軸に時間が配置される。任意の位置を指示し、クリックするとダイアログボックスが表示されるので引数を記入して確定させると、引数で指定された動作を示すボックスが表示される。これを繰り返し、短期時間シナリオを作成していく。最後にファイルへの格納のコマンドを選ぶと、動作列が時間順に整理された短期時間シナリオファイルが生成される。
ここで生成された短期時間シナリオファイルと長期時間シナリオファイルを入力し、実行していくシナリオプレーヤを RT コンポーネントとして開発した。

まとめ

以上、ロボットに組み込まれるアプリケーションロジックを容易に作成するための作業シナリオ設計ツールを開発し、プロジェクト参加組織内に提供した。
また、シナリオ編集系・実行系ともに作業シナリオ仕様記述方式で既定したシナリオメッセージ規約、ワーカ定義規約に基づいて実装している。特に、ビジュアルシナリオ編集系ではターゲットロボットに併せて、ビジュアルブロックをカスタマイズできる構成を持っており、リファレンスハードウェア向けのビジュアルブロックを作成し、平成 20 年度に実施した先行デモおよび後述の検証用知能モジュールの検証デモにおいて、その有効性を実証した。

(b)動作設計ツールの開発

(b-1)動作パターン設計ツール

多関節を有するロボット対して、関節角軌道として表現されるロボットの「動作パターン」をインタラクティブに編集するためのツールを開発した。編集した動作パターンは、作業における基本動作や、コミュニケーションにおけるジェスチャ、エンタテインメントコンテンツとしての活用及び機構設計時の検証用途に用いることを想定している。本研究項目の最終目標は、本事業内で開発された他のツールとの連携機能を強化し、事業化を検討することであった。以下、開発した動作パターン設計ツールについて述べる。

概要

多関節を有するロボット対して、関節角軌道として表現されるロボットの「動作パターン」をインタラクティブに編集するためのツールの開発を行った。編集した動作パターンは、作業における基本動作や、コミュニケーションにおけるジェスチャ、エンタテインメントコンテンツとしての活用及び機構設計時の検証用途に用いることを想定している。以下に開発したツール全体のスクリーンショットを示す。

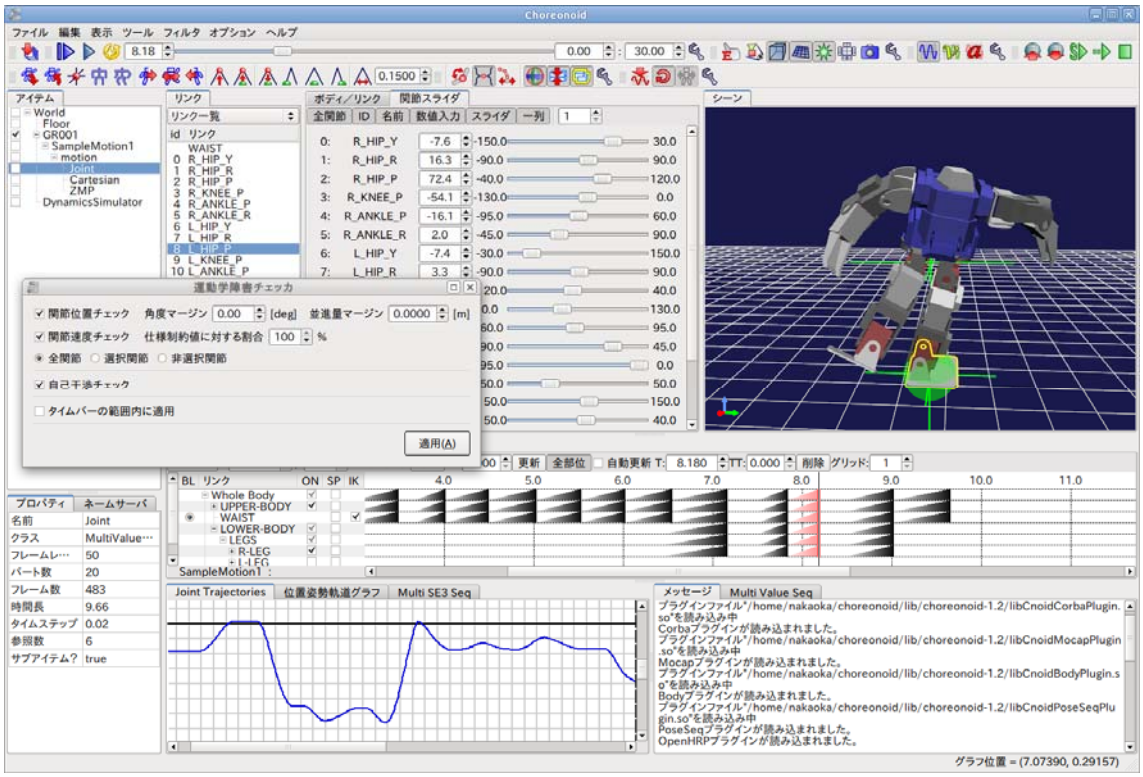


図 103 動作パターン設計ツールのスクリーンショット

基本設計

本ツールの基本的な設計は以下の方針に基づいて行った。

- ✧ 内部計算処理だけでなく、3D レンダリングを含む可視化、アニメーション、およびユーザ入力と内部計算との連携も含めて、コンピュータの性能を最大限活用可能な設計とする。
- ✧ 必要に応じてユーザが機能を柔軟に拡張可能とする。
- ✧ ロボットや計算機の専門家ではないユーザにも使いやすいツールとする。

これらの方針に基づき、具体的に以下のような設計を行った。

C++と非分散設計の採用

本ツールの想定する基本機能を実現するためには、グラフィックの描画や内部計算処理、およびそれらとユーザ入力との連携に関して、十分な高速性を確保する必要がある。なぜなら、動作の作成を直感的に行うためには、ツール上での編集操作を即座に動作データの更新と提示に結びつけることが重要だからである。従って、ツールの土台となる部分の設計は、内部計算処理だけでなく、3D レンダリングを含む可視化、アニメーション、およびユーザ入力と内部計算との連携も含めて、コンピュータの性能を最大限活用可能な設計となっている必要がある。

しかしながら、プラットフォームが標準 GUI として採用している Eclipse 環境

は、多くの面で優れた環境であるものの、上記の点に関してはその要求を十分に満たせるものではない。なぜなら、Eclipse は Java 言語で構築された環境であり、C/C++で記述されたロボットモデルの運動学や動力学に関する内部計算処理と連携を行う際のオーバーヘッドが大きいからである。
具体的には、Java の JNI 機構を用いて C の関数を呼び出す場合、そのためのラッパインタフェース実装は煩雑であり、呼び出し速度も当然各言語内における関数呼び出しと比べて遅くなる。C++の場合は C 以上に煩雑なラッパ実装が必要となる。また、動力学シミュレータで採用しているように、CORBA を用いてネットワーク経由で呼び出す場合、JNI と比べてさらにオーバーヘッドは大きくなってしまう。シミュレータにおいては基本的に動力学計算サーバの出力する剛体の位置姿勢情報を表示に反映できればよく、そのようなデータはデータ量もさほどではなく、そのやりとりも動力学計算全体からみると微々たるものなので、大きな問題とはなっていない。しかし、GUI 上での入力操作と連携させるにあたっては、このオーバーヘッドが問題となってくる。さらに、シミュレータの場合はやりとりすべきデータはほぼ確立されたシンプルなものであるが、動作データ作成に関しては多様な動作データをやりとりする必要が生じるため、JNI や CORBA で接続するためのラッパ実装のコストも開発において大きな負担となってしまう。

これに関して、そもそもロボットモデルの計算処理自体を Java 言語で記述するという方法も考えられる。そのようにすれば、少なくとも関数呼び出しのオーバーヘッドは解消できる。また、Java は JIT コンパイル機能を備えており、状況によっては C++と同等以上の実行速度が得られる可能性もある。

それでもなお、ロボットモデル計算処理を Java で記述することには問題が多い。まず、JIT によって実行速度が速くなるとしても、その程度は予測・制御ができない。また、Java では基本型以外のオブジェクトはすべてヒープ上に個別に確保されるため、例えば C++のようにベクトル型や行列型をスタック上に確保したり、コンテナ上に直接（ポインタを介さず）配置したりするということもできない。Java のガベージコレクションは便利ではあるものの、その実行タイミングは制御しづらく、実行時には一定の負荷がかかってしまう。さらに、ネイティブコードに比べて新たなプロセスの起動が遅い。以上のような Java の性質は、やはり実行速度の最適化という点で C++に大きく見劣りするものである。これは例えばロボット制御時に短い周期でリアルタイム実行をしなければならない場合や、シミュレーションや動作計画等において大量の計算処理を高速に行う必要がある際に、問題となる。また、C++では演算子オーバーロードを用いて例えば行列・ベクトルについて数学的に自然な式記述をすることも可能であり、テンプレートを用いて多様な型に対してコードの共有と最適化を両立することも可能である。このような機能はロボットの計算処理の開発効率や保守性を高めるにあたって大変有用で

あるが、Java ではそのような機能がない。さらに、実際にロボティクスにおいて有用な多くの既存ライブラリが C/C++にて記述されている。それらを直接利用できるという点も、C++を用いる重要な利点のひとつである。そして、そのような既存ライブラリ資産は Java よりも C++が充実しているという事実自体、上記の考察を裏付けるものである。

以上まとめると、以下の２点が言える。

- 動作設計ツールにおいては、ロボットモデルの可視化や GUI 操作に関して、ロボットモデルの内部計算処理との高速かつ多様な連携が求められる。
- ロボットモデルの計算処理においては Java よりも C++が適しており、実際に多くの状況で C++が使われている。計算処理において有用な既存のライブラリ資産についても、C++の方が充実している。

以上を考慮した結論として、本ツールにおいて我々は Eclipse ではなく、自前の C++実装をフロントエンド部分に適用し、内部計算処理と直接リンクする非分散型の設計とすることにした。これに関して、Eclipse 上に GUI ツールを統一するという理想に反するものではないかという議論も行った。しかし、それを重視するあまり各ツールの目的を十分に達成できなければ本末転倒であり、さらにこのような性質を必要とする他のツールが必要となることも今後考えられることから、本プラットフォームの可能性を広げるためにも、本ツールのような設計をプラットフォームにおいてひとつ示しておくことは、重要であるという結論に至った。

ウィンドウシステム

動作パターン設計ツールにおいては、多くの機能を GUI 上に構築する必要がある。このため、GUI のウィンドウやツールバーといったコンポーネントについて、多様な種類のものをまとめる必要がある。そしてこれは作業内容に応じて適切な配置が行えることが望ましい。以上を考慮して、以下の設計のウィンドウシステムを採用することとした。

- ビューは基本的にひとつのメインウィンドウ上に配置される。
- ツールバーは機能ごとにまとめられ、各ツールバーの配置はユーザが自由に変更可能とする。
- ツールバーよりも広い領域を持ち、各種操作や可視化、データの編集等を行う矩形領域を「ビュー」とし、ビューについてはメインウィンドウ上に分割して配置する。
- 当面の作業に必要でないビューは、他のビューと同じ領域にて、他のビューの背後に隠すことができる。ある領域に含まれるビューはタブで管理され、任意のビューを表に出すことが可能である。
- 上記のビューの分割とタブ配置についても、ユーザが自由に設定可能とす

る。

以上の設計は基本的には Eclipse と同様のものであり、Eclipse 上に構築された他のツールと操作感を統一することが可能である。

汎用アイテムツリー設計

本ツールでは、多様な種類のデータを同時に扱うことが想定される。例えば、ロボットのモデルデータ、環境のモデルデータ、ロボットの動作を生成するキーフレームデータ、さらに生成されたデータを格納する動作軌道データなどである。さらに、これらのデータに加えて、ロボットの動作と同期して再生する音声データや、ロボットの動作データを時系列に並べて順番に再生するシナリオデータ、ロボットモデルと実機のロボットを接続するためのコントローラなども追加することが想定される。

本ツールの開発においては、そのような多様なデータをシンプルかつ統一的に扱うための「汎用アイテムツリー」を設計した。この設計のポイントを以下に示す。

- 本ツール上で明示的に操作対象となるモデルやデータは、「アイテム」として抽象化され、「アイテムツリー」にツリー構造で格納される。
- アイテムの生成・読み込み・保存、あるいは基本的な属性の表示や編集といった、共通となる機能については、共通の関数やインタフェースで行うことが可能である。
- 各アイテムをツリーのどの位置に配置するかは、基本的にユーザの自由とする。
- ある機能を実行する際にアイテム間の関係を知る必要がある場合、ツリー内での位置（親子関係も含む）によってその関係を判断する。
- どのような位置関係をアイテム間の関係ととらえるかは、各機能が自由に判断するものとする。
- 上記の関係性に関する操作体系の統一は、あらかじめ定められた厳密なルールに従うのではなく、慣習的なものとする。
- アイテムツリーの状態は、「プロジェクト」として一括して保存、読み込みが可能とする。

このようなツリーの扱いは、必ずしも今までのソフトウェアで一般的なものではない。確かにツリー構造は多くのソフトウェアで一般的に使われているが、そのほとんどは定められた特定の構造に従うか、ある特定のデータタイプのみを含むようなものである。例えばファイルツリーはファイルやディレクトリという特定の型のみを対象としたツリーである。あるいは、グラフィックソフトウェアやシミュレータ等においてよく見られる、3D 描画オブジェクトの階層構造を表現す

る「シーンツリー」は、描画に関連するオブジェクトのみを扱うものであり、取り得る親子関係についてもオブジェクトごとに定められている。あるいは、編集対象のいくつかの種類のデータをツリーで扱うソフトウェアもあるが、それらは多くの場合、特定の型ごとにグループ分けされた親ノードに対応する型を格納するものとなっている。

一方で、本ツールで設計した上記のツリーの扱いは、それら既存のツリーインタフェースと比べて、より柔軟性の高いものとなっている。ただし、ツリー配置のルールが厳密に定められていないため、配置の慣習を知らなければデータの関係性を適切に指定できないことにもなる。ただし、実際にこの運用法を実装して動作パターンの設計を多様なユーザにテストしてもらった結果、このことは大きな問題にならないことが分かった。つまり、関係性の記述は親子関係などをベースに人間にとって自然なかたちで運用されるので、そのような慣習はすぐにユーザが把握できている。一方で、この運用法の特徴である柔軟性の高さは、多様なモデルやデータを同時に扱い、さらに機能を拡張して新たなモデルやデータを追加していくにあたって、有効に機能することが確認できた。

以下はアイテムツリーの実際の利用例を示す図である。

図 104　アイテムツリー

この例では、まずトップレベルのノードとして”World”というアイテムを配置している。このアイテムはひとつの仮想世界を示すものであり、このアイテムの子アイテム（子孫アイテム）としてロボットや環境のモデルに対応するアイテムを配置することにより、それらのモデルが同一ワールドに存在するものであることを表現する。そして、例えばモデル間の干渉チェックなどは、この関係を考慮して行われることになる。ロボットモデルの組み合わせについて、いくつかに分けて編集を進めたい場合は、この World アイテムを複数作成し、それぞれに適当なモデルアイテムを配置すればよい。

次に、World アイテムの子アイテムとして配置している、”Floor”, ”G ロボット 1”, “G ロボット 2”というアイテムは、それぞれ床と、G ロボットというロボットモデルに対応している。ここでは”G ロボット 1”のみに左端のチェックを入れているため、3D ビューに表示されるのは”G ロボット 1”のみとなっている。

そして、”G ロボット 1”について、関連するアイテムを小アイテムとして配置し

ている。まず”GRobot コントローラ”というのは、実機の G ロボットを動作させる機能をアイテムとして抽象化したものである。実際にどのロボットを動かすかは、このアイテムの属性として設定可能である。そして、どのモデルがどの実機に対応するかは、アイテムの親子関係で表現される。ここでは、”G ロボット”が 2 体あり、それぞれのコントローラはそれぞれの親アイテムになっているモデルの動きに合わせて動くことになる。

“シナリオ“というアイテムは、いくつかの動作データや音楽データ等を定められたシナリオで順番に実行・再生していく機能を抽象化したアイテムである。自身の子アイテム（子孫アイテム）として配置されている動作データのアイテムや音楽データのアイテムを、ツリー上で上から順番に実行していく。このように、単にアイテムの配置を適切に行うだけで、何を、どの順番で実行していくか、という関係性を指定できていることになる。

“キーフレームデータ 1”というのは、後述するキーフレーム列に対応するデータである。本ツールではこのデータを編集することで、ロボットの動作パターンを作成可能となっている。また、このアイテムについても、どのモデルアイテム以下に格納されているかによって、データとモデルの組み合わせが指定されている。これにより、複数の動作データと複数のモデルを自由に組み合わせて同時に動作編集を進めることが可能であり、これをシンプルなツリーシステムで実現できていることが本ツールの大きな特徴である。

キーフレームアイテムは、自身の子アイテムとして”motion”というデータアイテムを持つ。これについてはユーザがツリー内で自由にアイテムの位置を変えられるということの例外となっており、親アイテムが自前で生成して管理しているアイテムとなっている。このようなアイテムを、「サブアイテム」と呼ぶ。サブアイテムは、その配置をユーザが自由に帰られないという以外は、通常のアイテムと同じである。ここではキーフレームを補間して生成される動作軌道データを格納する目的で使われている。”motion”アイテム自体は、そのような動作軌道データを格納するアイテム型となっており、もともとこの型のアイテム用に用意された機能がそのまま利用可能である。例えば、実際の軌道をグラフ上で確認したい場合は、このアイテムを対象としてグラフを表示すればよい。また、動作データの再生機能や実機ロボットの制御などは、軌道データを対象に作成しておけば、キーフレームデータに対して新たに作成する必要はない。さらに、”motion”アイテムの子アイテムである”Joint”と”Cartesian”は、”motion”アイテムの子アイテムとなっている。それぞれ関節角軌道とリンクの位置姿勢軌道に対応する。このように必要に応じてひとつのデータを複数のアイテムで構成することで、機能の柔軟な共有が可能となる。

ここでは、キーフレームデータアイテムの子アイテムとして、さらに”音楽 1”と

いうデータアイテムが配置されている。これは音楽データに対応するアイテムで、選択して再生を行うと音楽が再生される。ここではこのアイテムをシナリオアイテム以下にキーフレームデータと並べて配置することで、シナリオにおける各動作にそれぞれ個別の音楽を対応付け再生するということを行っている。このような少々複雑な設定も、単にツリー内の適切な位置に音楽アイテムを配置するというシンプルかつ統一された操作で実現できる。

以上のように、本アイテムツリーの設計により、多様なデータをシンプルかつ統一的な操作で扱うことが可能となっている。

機能拡張のためのプラグインシステム

本ツールでは、上記のアイテムツリーの例でも示したように、使用するロボットごとにその実機をツール上から動かすためのコントローラアイテムを追加したり、動作データだけでなく音楽データも読み込めるようにしてロボットの動作と合わせて再生したりと、様々に活用することが想定される。これを実現するためには、想定される全ての利用法やロボットに対してあらかじめ機能を開発することは困難であり、ツールを利用者が自由に拡張できることが望ましい。

このため、本ツールの基本設計として、機能拡張のためのプラグインシステムを構築した。プラグインは共有ライブラリとして実装され、プラグインオブジェクトを生成するエントリ関数によってプラグインが生成される以外は、通常の共有ライブラリと同等のものである。従って、プラグインは他のライブラリの機能が利用できるだけでなく、他のプラグインの機能も利用できるようになっている。この場合、プラグイン間の依存関係を指定する機能も利用可能である。この設計は、特定の関数のみ追加を許すような、一般的なプラグインシステムと比べて非常に自由度の高いものとなっている。従って拡張の種類も特に限定されておらず、ユーザは必要な機能を自由に追加していくことが可能である。その際、追加したモデルやデータと既存のデータとの連係も、上述の柔軟なツリーシステムによって簡便かつ効果的に行うことが可能である。

このように本ツールのプラグインシステムは自由度の高いものであり、ツールの基本機能についてもプラグインとして実装することで、システム全体の統一性を高めることができる。このため後述するように、ツールの基本機能の多くが実際にプラグインとして実装されている。

実際のプラグインのコーディング例を以下に示す。ここでは、”Increment”と”Decrement”という２つのボタンをツールバーに追加し、”Increment”ボタンを押すと選択されているロボットモデルの全ての関節角度が一定値増加し、結果の姿勢がモデルの状態や表示に反映される。”Decrement”については同様に一定し減少させる。これらの結果として、3D ビューのロボットの描画や、関節角スライダ

の状態など、モデルと関連する全ての機能が影響を受ける。あまり意味のないプラグインではあるが、このように、「対象となるモデルを取得し」、「モデルに操作を加えて」、「その結果を全ての関連するアイテムやビューに反映させる」という比較的複雑な処理が、以下のシンプルなコードで実現できる。

```
class SamplePlugin : public Plugin {
public:
    SamplePlugin() : Plugin("Sample") { depend("Body"); }
    virtual bool initialize() {
        ToolBar* bar = new ToolBar("Sample1");
        bar->addButton("Increment ")
            ->sigClicked().connect(bind(&onButtonClicked, +0.04));
        bar->addButton("Decrement ")
            ->sigClicked().connect(bind(&onButtonClicked, -0.04));
        addToolBar(bar);
        return true;
    }
};
void onButtonClicked(double dq){
    ItemList<BodyItem> bodyItems =
        ItemTreeView::mainInstance()->selectedItems<BodyItem>();
    for(size_t i=0; i < bodyItems.size(); ++i){
        BodyPtr body = bodyItems[i]->body();
        for(int j=0; j < body->numJoints(); ++j){
            body->joint(j)->q += dq;
        }
        bodyItems[i]->notifyKinematicStateChange(true);
    }
}
```


図 105　プラグインのコーディング例

本コードを簡単に解説すると、まず”Plugin”クラスを継承したクラスを定義することで、プラグインを実装する。そして、コンストラクタの”depend”関数で本プラグインが”Body”という名前のプラグインに依存していることを示している。”Body”プラグインはロボットモデルを扱う基本的なクラスや関数を実装したプラグインとなっている。

次に”initialize”関数をオーバーライドし、初期化処理を記述している。ここでは”Increment”と”Decrement”の２つのボタンを有するツールバーを作成して追加している。各ボタンが押されたときに実行する関数をこのコードのようにシンプルに指定することが可能である。ここでは”sigClicked”というボタンが押されたときに発行されるシグナルに対して、”onButtonClicked”という関数と関数実行時のパラメータ（関節角の増減値）を結びつけている。

次に”onButtonClicked”関数内にて、ボタンが押されたときの処理を記述してい

る。ここでは前述のアイテムツリービューに対応する”ItemTreeView”のインスタンスに対して、”selectedItems<BodyItem>()” を実行することにより、ユーザが選択しているロボットモデルのアイテムを取得している。そして、取得した各ロボットモデルに対して、関節数分のループを回し、各関節の関節角を dq 分だけ増減させることで、モデルの状態を更新している。最後に、モデルのアイテムに対して”notifyKinematicStateChange” を実行することで、このモデルの運動学的状態が更新されたというシグナルを発生させている。モデルの状態と関連しているビューなどはこのシグナルに対して自身を更新する関数を結びつけているため、”notiyKinematicStateChange”をひとつ呼ぶだけで、関連する全てのビュー等の更新が可能となっている。これにより、他にどのようなビューが関連してるかを特に気にせずとも、それらのビューと連携することが可能であり、拡張性の高い設計となっている。

以上述べたように、本ツールのプラグインシステムによって、必要最小限の労力で機能を拡張し、既存の機能と連携させることが可能となっている。

モジュール構成

本ツールは以下の図に示すモジュール構成で構築されている。

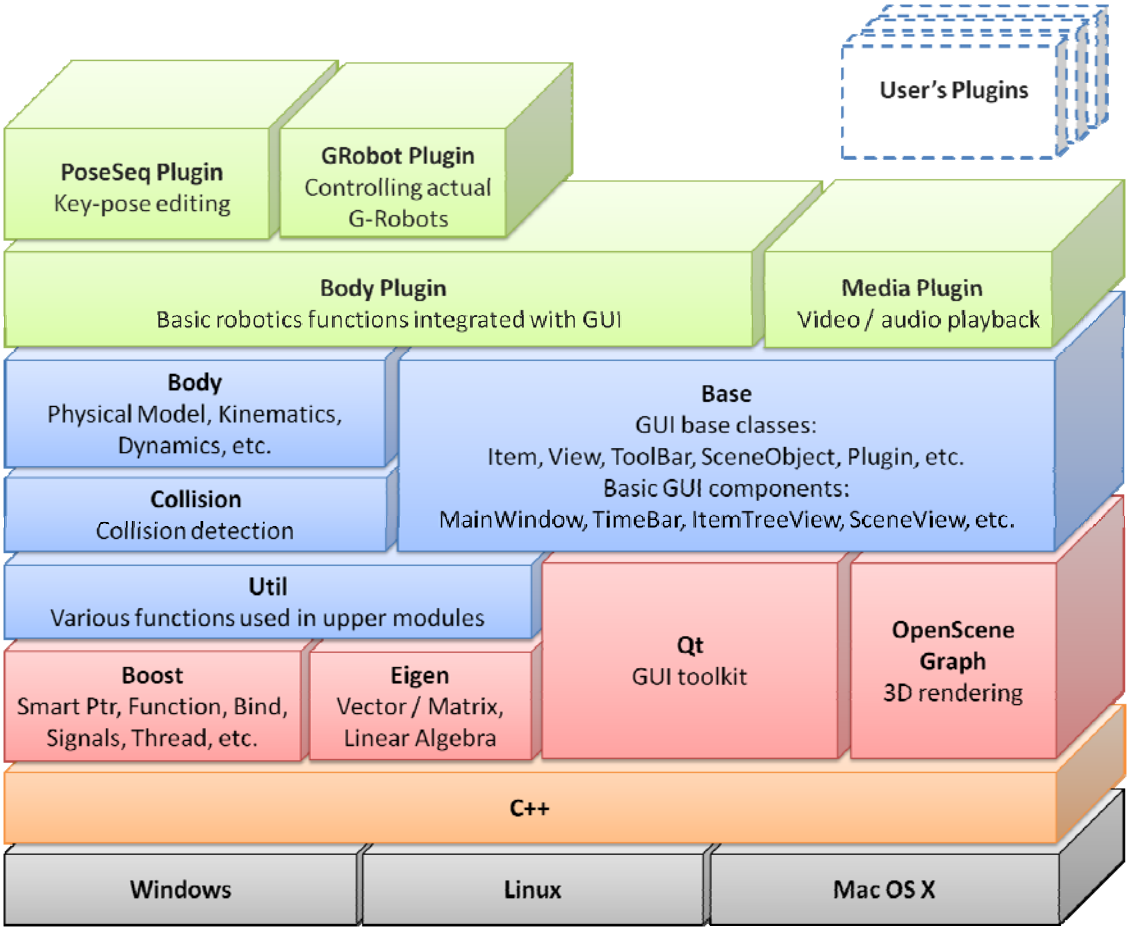


図 106　動作パターン設計ツールのモジュール構成

まず、本ツールはマルチプラットフォーム対応としており、Windows, Linux, および Mac OS X で動作可能としている。また、言語については、前述のように C++ を基盤としている。

GUI ツールキットや 3D シーングラフ描画、行列・ベクトル演算などに関しては、自前で全て開発するのではなく、既存のライブラリを用いることにしている。代表的なものとして以下のライブラリを利用している。

Boost C++ Libraries: C++の汎用的なライブラリ集で、標準ライブラリがカバーしない部分をカバーする大変有用なライブラリ集である。本ツールでは、柔軟な拡張性を実現するために、本ライブラリ集の Signals, Bind, Function といった関数の格納や呼び出しに関するライブラリが大きな役割を果たしている。それ以外にも多くのライブラリが本ツールの実装で使用されている。

Eigen: 行列・ベクトル演算をはじめとする線形代数処理を扱う C++のテンプレートライブラリである。演算子オーバーロードと式テンプレートの活用により、数

学における記述に近い記述を実際のコードでも実現でき、結果として生成されるコードもオーバーヘッドの少ない非常に高速なものとなっている。さらに SSE などの拡張 CPU 命令を扱う機能も備えており、手動のコード最適化に匹敵する最適化を得ることができる。また、固定サイズの行列・ベクトルと可変サイズの行列・ベクトルを同じ API で扱うことが可能であり、効率の高さと柔軟性を兼ね備えている。以上のような性質から非常に使いやすい上に高速で、大変有用なライブラリである。行列・ベクトル演算に関するライブラリとしては、他に tvmet, uBlas, Blas, Lapack 等が挙げられる。Tvmet については、固定サイズのみを扱うライブラリで、可変サイズを扱うことができない。uBlas は逆に可変サイズに特化して設計されたものであり、固定サイズの演算については十分な最適化が出来ない。Blas については元々Fortran のライブラリであり、可変サイズを対象として関数呼び出し形式で使うため、比較的小さいサイズの行列・ベクトルに対しては効率が悪い。また API も数学的に自然な記述ができるものではない。Lapack は Blas を基盤とする線形代数ライブラリで、多様な線形代数演算ができる。これは Eigen がカバーしていない線形代数演算を行う際には有用であるが、本ツールの実装においては Eigen がカバーするもので十分となっている。以上の理由により、本ツールの開発においては Eigen を採用することとした。

Qt: GUI ツールキットライブラリで、多様で高品質な GUI 部品を提供する。GUI ツールキットとしては、他に wxWidgets, Gtkmm(Gtk+) 等がよく知られている。実際に本ツールの開発においては、我々は当初 wxWigets を採用し、次に Gtkmm と入れ替え、最終的に Qt を採用するに至った。当初 wxWidgets を採用した理由は、各 OS のウィンドウシステムの比較的ハイレベルな API を直接利用するライブラリは wxWidgets のみであり、さらにライセンスも LGPL ベースで本プラットフォームの運用に問題なかったためである。この時点では Qt は商用ライセンスか GPL ライセンスとなっており、どちらもオープンかつ商用利用も可能なプラットフォームの運用では問題があった。また、Gtkmm については Windows や Mac OS X への対応の面で他のライブラリに劣ることが懸念された。しかしながら、実際に wxWidgets で実装を進めたところ、我々が必要とする GUI 機能が十分に実現できないことが分かった。また、そのような機能を工夫して実現しようとすると、プラットフォームによって挙動が異なり、結局複数のプラットフォーム間でコードを共有できないという問題も生じた。次に Gtkmm に乗り換えた理由は、少なくとも Windows のサポートには改善がみられ、Qt は依然としてライセンスに関する問題があったからである。実際、Gtkmm は我々の要求を概ね満たすものであった。しかしながら、Gtk+バージョン 2.19 以降から再び Windows のサポートが悪くなり、Windows においてウィンドウが正常に描画されないなどの問題が生じた。そして、この時点までに Qt のライセンスが LGPL に変更されていたため、

Qt の採用に至った。Qt はもともと商用ライブラリとして開発されてきただけに、他の２つのライブラリと比較して細かい点までよく作り込まれており、現状では最適な選択となっている。

OpenSceneGraph: 3 次元 CG 描画をシーングラフというハイレベルな API で行うためのライブラリ。OpenGL をベースに構築されており、必要であれば OpenGL の API を直接用いることも可能である。

これらはいずれもよく知られて広く使われている定評あるライブラリである。また、いずれもマルチプラットフォーム対応となっており、本ツールをマルチプラットフォーム対応させるにあたっても重要な役割を果たしている。

以上のライブラリの上に、本ツールの基盤となる以下のモジュールを構築している。

Util: ツール実装の各所で使われるクラスや関数をまとめたユーティリティライブラリ。

Collision: 干渉チェック機能を提供するライブラリ。

Body: ロボットモデルの定義や各種計算処理を扱うライブラリ。

Base: ツールの GUI に関連する基盤機能をまとめたモジュール。

そして、これらのモジュールの上に、プラグインが構築される。本ツールが標準で提供するプラグインとして、以下がある。

BodyPlugin: ロボットモデルや動作データに関する GUI や処理を実装したプラグイン。主に Body ライブラリの機能を GUI 上で扱うためのコードを記述している。

PoseSeqPlugin: ロボットのキーフレーム編集を行う機能を実装したプラグイン。

GRobotPlugin: HPI ジャパン株式会社の小型ロボット”G-Robot”の実機をモデルの動作と連動させて動かすための機能を実装したプラグイン。同様のプラグインを書くことで、他のロボット実機への対応も可能となる。本プラグインはそのためのサンプルとして開発した。

MediaPlugin: ビデオや音声をロボットの動きと合わせて再生するためのプラグイン。このプラグインのように、直接はロボットと関係のない多様な機能をプラグインとして自由に追加することが可能となっている。

動作パターン作成のための基本機能

本ツールの本題である動作パターン作成機能に関しては、以下を開発目標として掲げていた。

- モデル読み込み機能

- 姿勢提示・編集機能
- 姿勢シーケンス編集機能
- 動作提示機能
- 制約適応支援機能

以下ではそれぞれの目標に対して、実際に開発した内容を述べる。

モデル読み込み機能

目標： 編集対象となるロボットについて、その関節構造や形状、各関節の仕様などの情報は、本プロジェクトで策定するハードウェア仕様記述に基づくモデルファイルから読み込むものとする。これにより、モデルファイルを用意することで、任意の形状・構造のロボットの動作パターンを編集可能とする。

本目標については、「ハードウェア仕様記述方式」にて述べた、VRML97 に基づくロボットハードウェア記述方式に対応させており、問題なく実現することが出来た。なお、「IDL によるハードウェア仕様記述方式」については、現状では対応していない。これは、2.1 で述べた「C++で非分散型」の設計思想に基づくものである。もちろん、IDL への対応は必要に応じて追加プラグインのかたちで行うことが可能である。

姿勢提示・編集機能

目標： ロボット全身の姿勢を提示・編集するためのインタフェースとして、各関節角の数値表示によるものと、３Ｄグラフィック表示によるものを実装する。３Ｄグラフィックにおいては、表示されているロボットのリンクをドラッグすることで効率的に姿勢を編集可能なインタフェースを実現する。

本目標についても対応する機能の開発を行った。まず、下図に示す２つのビューは姿勢を数値形式で提示・編集するためのインタフェースである。

左側のビューは”BodyLinkView”であり、ロボットの各リンクについて、その関節角、リンク位置、姿勢等を数値形式で閲覧・編集することが可能である。また、リンク間の干渉状態についてもリンク名に対応する文字列で明確に確認することが出来る。

右側のビューは、”JointSliderView”であり、複数の関節の関節角度を一括して確認・編集できるビューである。数値による確認・編集に加えて、関節角に対応するスライダも提供しており、これらを使い分けることで効率的な関節角編集が可能である。

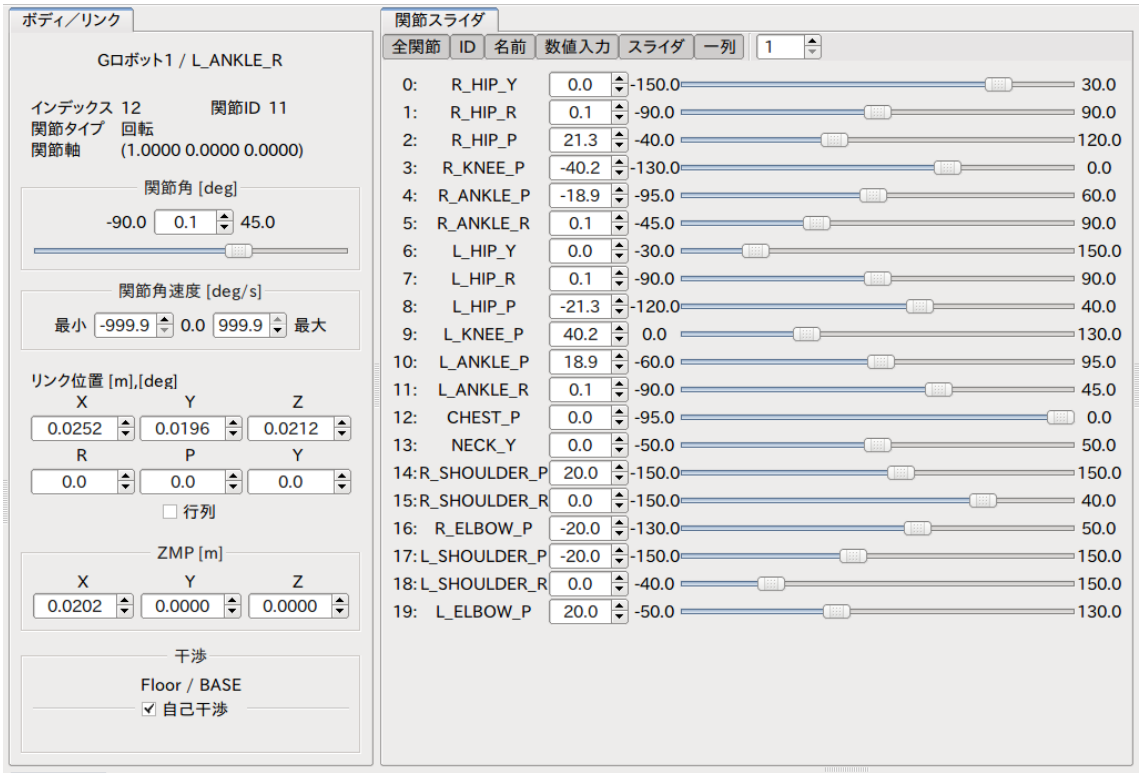


図 107　BodyLinkView と JointSliderView

3D グラフィック表示による姿勢の確認・編集は、本ツール全体の 3D 描画を行う”SceneView”に統合されている。以下に姿勢編集中の SceneView の状態を示す。

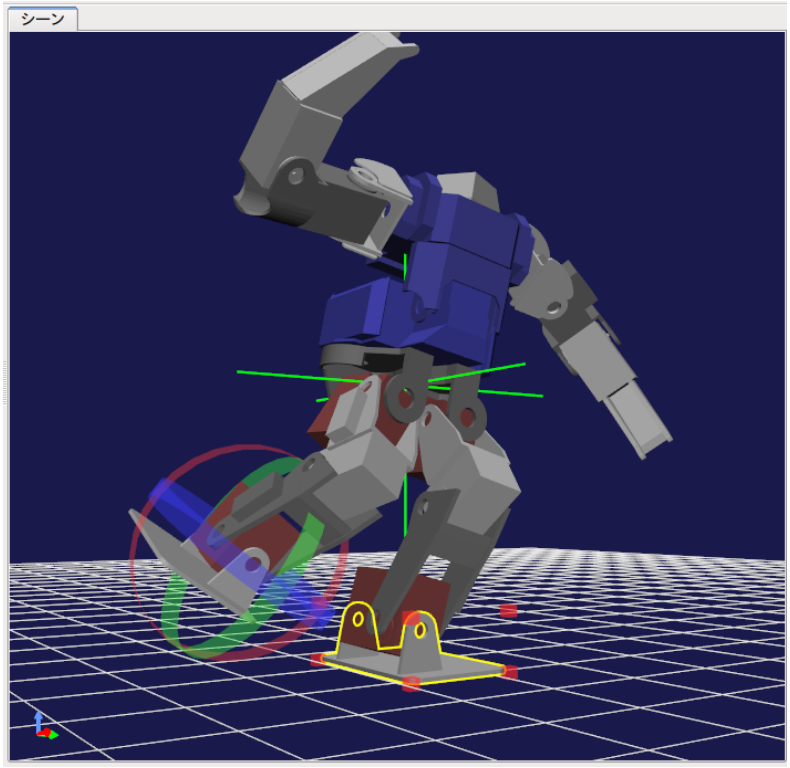


図 108　SceneView

このビュー上では、図から分かるようにロボットの姿勢を 3D グラフィックで確認できる。もちろん、作成した動作もアニメーション表示によって確認することが出来る。そして、3D モデルの各リンクをマウスでドラッグすることで、そのリンクの位置姿勢や、関節角度をダイレクトに編集することも可能である。編集には順運動学モード、逆運動学モード、およびハイブリッドモードが利用可能であり、これらを使い分けることで効率的な姿勢編集が可能である。また、それらの編集操作を支援するいくつかのツールバーも提供しており、モードの切り替えや、特定のパターンの姿勢の変更などは、それらのツールバーが提供するボタンを押すことで簡単に実現できる。

姿勢シーケンス編集機能

目標： 姿勢編集機能で作成した各姿勢を時間軸上に並べ、姿勢のシーケンスから動作を構成するためのインタフェースを実現する。ロボットの動作パターンとなる関節角軌道はユーザの提示した姿勢シーケンスからシステムが生成するものとする。

本目標を実現する機能として、主に”PoseSeq”アイテムと”PoseRoll”ビューを開発した。

まず、PoseSeq アイテムは、姿勢シーケンスを格納するデータ構造に対応したアイテムとなっている。そして、このアイテムの内容は、以下に示す PoseRoll ビューを用いて編集可能である。

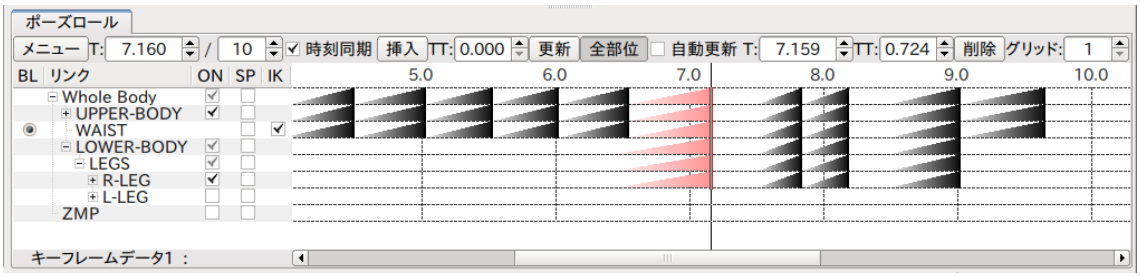


図 109　PoseRollView

PoseRoll ビューでは、横方向が時間軸となっており、適当な時間位置にて「挿入」ボタンを押すことで、現在のモデルの姿勢がキーフレームとして追加される。各キーフレームは遷移開始位置からキーフレーム到達位置までに渡る三角形で表示される。また、既存のキーフレームを選択状態にした上で姿勢編集を行うことで、各キーフレームの姿勢を変更することも可能である。さらに、ビュー上部の各種数値ボックスや、キーフレームに対応する三角形領域をドラッグすることなどにより、キーフレームの時刻や遷移時間などを効率的に編集することも可能である。

また、PoseRoll ビューの縦方向は各身体部位に対応している。ビュー左部のツリーは、リンクの階層構造と一致しており、ここに表示されている身体部位と、その右側に続くキーフレームが、それぞれ対応する。すなわち、キーフレームは全身に対して一括して設定することも出来、その場合は身体部位ツリーの全ての領域がキーフレーム表示で占められることになる。一方で、ある身体部位のみにキーフレームを設定することもでき、その場合は設定した身体部位に対応する縦位置のみでキーフレームが表示される。例えば、上半身と下半身、あるいは右手と左手でキーフレームのタイミングが異なる場合、あるいは単に首だけを動かしたい、といった場合には、それぞれの身体部位に対してキーフレームを設定すればよい。この設定はツリーのチェック状態で行うことが出来、後で変更することも可能である。また、身体部位ツリーの各ノードは状況に応じて展開したりまとめたりすることが出来る。細かい部位レベルで設定を行いたい場合、例えば足首関節だけ動かしたい、といった場合には、そのレベルまでツリーを展開すればよいし、逆にあるまとまった部位に対して設定したい場合は、ツリーの展開を閉じることで、細かい部分を気にせずに編集を進めることができる。このようにして、身体部位を考慮した効率的な姿勢シーケンスの編集が可能となっている。

PoseSeqアイテムのキーフレームからは対応する軌道データが自動で生成され、PoseSeq アイテムのサブアイテムである BodyMotion アイテムに格納される。単にアニメーションを確認したい場合には、元の PoseSeq アイテムを選択して確認することも可能であるが、軌道レベルの細かい処理、例えばグラフを用いた軌道の確認などを行いたい場合や、軌道データとしてファイルの保存を行いたい場合などは、サブアイテムである BodyMotion アイテムに対して操作を行えばよい。

動作提示機能

目標: 姿勢シーケンス編集機能で作成された動作パターンを３Ｄグラフィックスのアニメーションとして提示、確認することができる機能を実現する。ツール内で実行される運動学シミュレーションに加え、シミュレータと連携した動力学シミュレーションも容易に実行可能とする。

アニメーション提示について、目標とする機能を実現した。下図に示すタイムバーを操作することで、作成した動作のアニメーション再生が可能である。

図 110 タイムバー

また、動力学シミュレーション機能についても、「動力学シミュレータ」において開発した動力学エンジンを組み込み、作成した動作パターンを本ツール上で直接動力学シミュレーションする機能を実現した。これにより、作成した動作パターンの力学的整合性を容易にチェックすることが可能である。

制約適応支援機能

目標： ロボットの制約条件を満たした動作パターンを効率的に作成できるようにするため、インタラクティブな編集操作と連動して制約とその逸脱を提示する機能を構築する。この機能により、関節仕様値の逸脱や力学的な安定性が編集操作の節々で確認可能となり、ユーザはロボットの制約条件を満たしたかたちで直接動作パターンを編集していくことが可能となる。

本目標に対応する機能として、関節角度や関節角速度の逸脱をグラフにてチェックする機能が利用可能である。リミット値の境界はグラフ上で明示的に表示されるので、グラフ上でリミット値を超えている部分が容易に分かるようになっている。また、グラフはキーフレームの更新時に連動して更新されるため、ユーザはリミット値を超えていないかを常に確認しながら作業を進めることが可能である。

力学的安定性に関しては、単に条件を満たしていないことを提示するだけでは、安定な動作を作成することは困難であることが判明した。これにより、力学安定性の制約逸脱を提示するだけの機能については開発を見送った。ただしこれに関しては、本ツールのテスト公開後に、産総研知能システム研究部門ヒューマノイド研究グループにて二足歩行ロボットのバランスを本ツール上で自動補正する機能が追加プラグインとして開発され、これを用いることでより発展的にこの問題を解決できるようになっている。これについては後述する。

利用状況

プラグイン開発事例

上で述べたように、本ツールはプラグインシステムによって機能を拡張可能となっている。そして、開発中の本ツールのテスト公開を平成 21 年 7 月に行ってから、いくつかのプラグインが開発されている。ここではそのうちの 2 点について報告する。

まず、二足歩行ロボットのバランスを自動補正するプラグインが、産総研知能システム研究部門ヒューマノイド研究グループにて開発された。これにより、バランスのとれた全身動作を作成することが難しい二足歩行ロボットに関しても、動作パターンを作成することが容易となった。

また、本プロジェクトの「HIRO 加速案件」の一環として”graspPlugin”という把持計画プラグイン群が開発された。これを用いることで、アームの先端にハンドが取り付けられたロボットシステムに対して、把持計画、動作計画、作業計画などを行うことができる。

以上のように、本ツールの動作パターン作成機能をベースとした拡張機能が開発され、ツールの応用範囲が広がってきており、本ツールがプラットフォームツールとして有効に活用されていることが確認できている。

公開状況

本ツールは開発中のものを平成 21 年 7 月よりテスト公開を行い、希望者には必要な情報を提供してきた。上記のプラグイン開発事例はその枠組みの中で行われたものである。

そして、平成 22 年 10 月 16 日に、上述のバランス補正プラグインと組み合わせた成果として、”Choreonoid（コレオノイド)”という命名で産総研プレスリリースを行った。その後平成 23 年 11 月 8 日より本ツールの公式サイトを立ち上げ、一般への公開を開始した。平成 24 年 4 月 11 日現在、計 2024 のダウンロードが確認されている。

まとめ

上述のように動作パターン設計ツールは、プラグイン機能を実装したことにより他のツール群との連携することが可能になっている。また、C++言語により実装を行ったことで十分な性能が実現でき、他のコンソで開発する知能モジュールのベースとしても利用されている。このツールはオープンソースライセンスで開発、公開されているが、独自のプラグインを開発することで様々なロボットアプリケーションへの応用することが可能であるため、外部企業による独自の事業化展開も期待できる。

(b-1)移動動作設計ツール

移動動作設計ツールとは、ロボットの移動能力に応じた2次元平面上での移動動作を、a)ユーザがGUIを用いてインタラクティブに設計する方法、b) ロボットが計画エンジンを呼び出して自律的に設計する方法、の2つの方法で設計可能なツールである。本研究項目の最終目標は、本事業内で開発された他のツールとの連携機能を強化し、事業化を検討することであった。以下、移動動作設計ツールの成果について述べる。

設計

ツールの全体設計

本ツールの利用方法として以下のような 2 つを想定する。

- 環境及びロボットの幾何モデルが既知の場合に、グラフィカルユーザインタフェース（以下 GUI）を用いて対話的に移動動作を設計し、それをシミュレーションや実際のロボットに適用するオフラインの使用方法。
- ロボットが実際に動作している状態で計測結果等に応じて環境モデルを構築・更新し、その時々の状況に応じた経路を計画し動作制御に使用するオンラインの使用方法。

以上のような二通りの使用方法を実現するため、本ツールは下図に示すような内部構造を持つ。

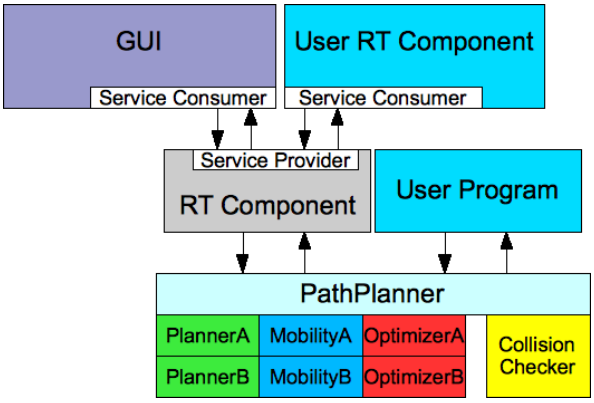


図 111　移動動作設計ツールの内部構造

大きく分けて以下の 3 つの部分からなる。

a)　実際に経路計画を行う計画エンジン部（上図下段）

b)　計画エンジン部を制御し、計画条件の設定、計画実行、計画結果の取得をサービスとして提供する RT コンポーネント（上図中段左の RT Component）

c)　RT コンポーネントのサービスコンシューマを内蔵し、計画条件の設定等を対話的に行い、計画結果を可視化する GUI（上図上段左の GUI）

これにより、前者の利用方法では GUI を介して RT コンポーネントを操作することでオフラインでの移動動作設計を行い、後者の利用方法ではユーザが開発した RT コンポーネント（上図中段右の User RT Component）からサービスを使用する、あるいはライブラリとして提供される計画エンジンを直接ユーザプログラム（上図中段右の User Program）にリンクすることでオンラインでの設計が可能となる。

計画エンジン部の設計

計画エンジン部は干渉検査部と 3 種類のアルゴリズムで構成される。干渉検査部はロボットの位置が指定された場合にロボットと環境との間に干渉が発生しているか否かを判定する。 3 種類のアルゴリズムはそれぞれの種類毎に複数のアルゴリズムを登録できるようにし、移動動作設計対象のロボットや使用方法等に応じて種類毎に適切なアルゴリズムを選択し、組み合わせて使用する。

3 種類のアルゴリズムはそれぞれ以下のような部分機能を担当する。

- 移動アルゴリズム：始点と終点（回転角を含む）が指定された時に、干渉の有無は考慮せずに 2 点間を移動する局所的な経路を計画するアルゴリズムで

ある。平面上の 2 点間の移動の可否や移動する経路はロボットの移動機構による制約や移動動作制御アルゴリズムによって異なる。このために一つのアルゴリズムで全てのロボットを対象とすることは不可能であるので、この部分を独立させ、対象とするロボットに応じたアルゴリズムに差し替えられるようにする。

- 経路計画アルゴリズム：移動アルゴリズムを用いて生成される局所的な経路に対して干渉検査を行い、障害物と干渉しないものを組み合わせてロードマップを生成し、ロードマップから始点と終点を結ぶ経路を探索するアルゴリズムである。
- 経路最適化アルゴリズム：経路計画アルゴリズムによって計画された経路を修正し、経路の総延長等、何らかの評価基準に用いて最適化を行うアルゴリズムである。経路計画アルゴリズムにランダムサンプリングベースのアルゴリズムを用いた場合には不要な経路が含まれる場合があるため、そのような場合にこのアルゴリズムを適用する。

実装

前節で述べた設計に基づいて移動動作設計ツールの実装を行った。下図に GUI の実行画面を示す。

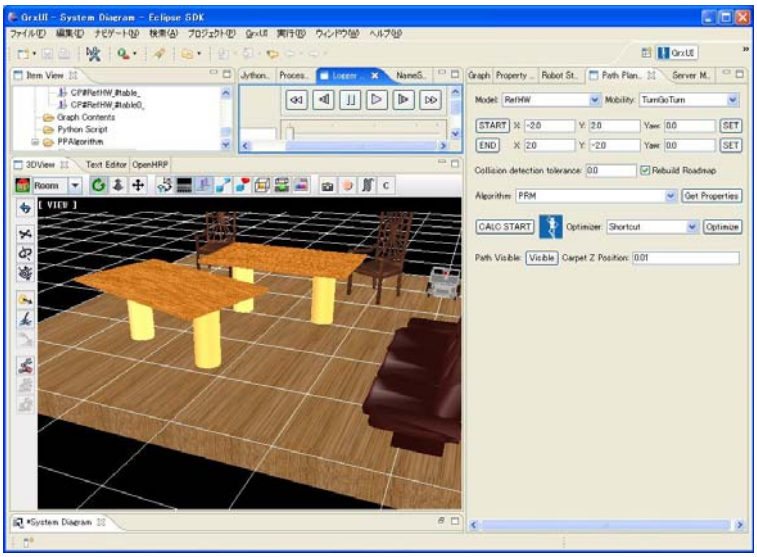


図 112　移動動作設計ツールの GUI

本 GUI は OpenRTP の一部である動力学シミュレータ、OpenHRP の GUI に計画条件の設定を行うビューを追加したものとなっており、3 次元グラフィックスの表示部や、環境やロボットの幾何モデルや計画を行うシーンを記述するファイル形式等を共有している。これによりシミュレーションと移動動作の設計をシーム

レスに行うことが可能となっている。

計画エンジンを構成する各種アルゴリズムはそれぞれの機能を抽象化した C++の抽象クラスに対して実装を与えることで新たなアルゴリズムの追加を行う仕組みとしている。現在の所、移動アルゴリズムとしてはその場旋回と直進を組み合わせて移動するアルゴリズムと回転しながら直線移動するアルゴリズム、経路計画アルゴリズムとしては RRT と PRM、経路最適化アルゴリズムとしては経路を構成する局所経路の接続点の内ショートカット可能な点を削除するアルゴリズムが標準で組み込まれている。これらのアルゴリズムはユーザが抽象クラスを継承して実装することで追加することが可能であり、ユーザが保有するロボットに応じた移動アルゴリズムを追加したり、オープンソースのモーションプランニング開発ツールである OOPSMP や OpenRAVE, MPK や商用の開発ツールである Kineo Path Planner に含まれる高度な計画アルゴリズムを追加したりすることが可能である。

下図に経路計画アルゴリズムとして RRT を用いた場合の一例を示す。ロボットが右下から左上へ移動する経路を計画した結果である。緑が障害物、青が移動可能な空間を表す。赤細線で示したのが生成されたロードマップ、半透明の太線が計画された経路であり、左が最適化前、右が最適化後である。

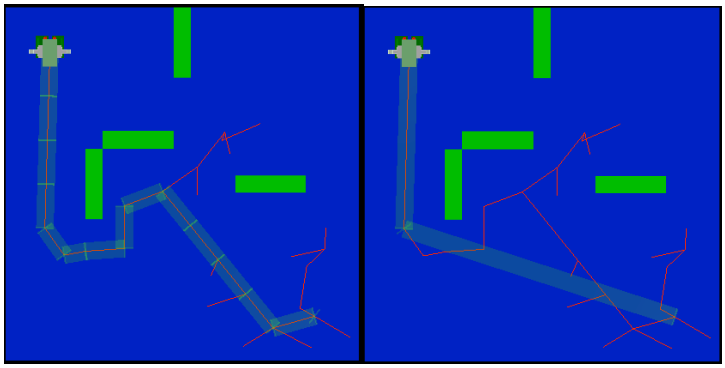


図 113 RRT を用いた経路計画の例

干渉検査部分に関しても OpenHRP に含まれる干渉検査機能を用いている。OpenHRP の干渉検査機能はオープンソースの干渉検査ライブラリ OPCODE をベースとしたものであり、三角形集合間の干渉を検査し、干渉している三角形のペアと干渉点、干渉法線、干渉深さを算出するものである。この干渉検査機能を用いた場合、干渉していない状態は全て同列として扱われるために障害物をかすめるような経路が計画される場合がある。そこで SSV(Swept Sphere Volume)を用いた近接距離計算機能を追加し、これを用いることで障害物までの安全距離を

設定できるようにしている。

下図に安全距離を 0 とした場合（左）と 0.3[m]とした場合（右）のそれぞれで PRM を用いて生成されたロードマップを示す。緑で示しているのが長辺 0.8[m]の障害物であり、安全距離が 0.3[m]の場合は障害物同士の距離が近いために障害物間をすり抜けるようなパスが生成されていないことが分かる。

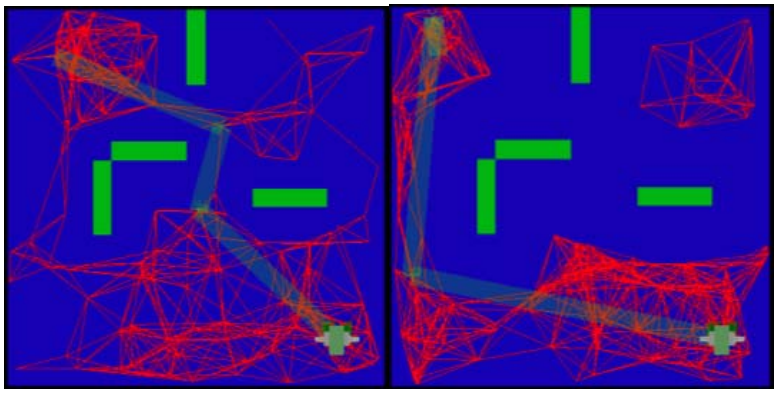


図 114　SSV を用いた経路計画の例

移動動作設計ツール利用手順

以下に移動動作設計ツールを用いて移動動作を設計する手順を示す。

初期設定

Eclipse 起動とパースペクティブの表示

通常通り Eclipse を起動し、GrxUI パースペクティブを表示する。GrxUI パースペクティブが表示されていない場合、「パースペクティブを開く(図 115 内[a])→「その他」をクリックし一覧から選択する。（図 115）

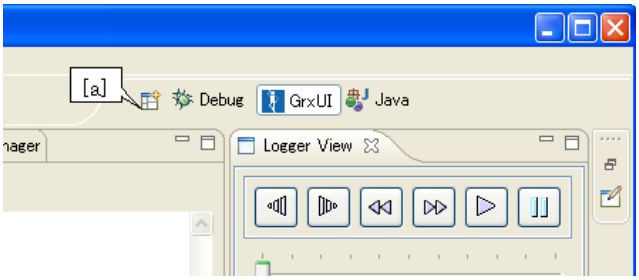


図 115　パースペクティブの選択

必要なビューの確認

3DView、Path Planning、Logger View、Property View、Item View など必要なビューが表示されていない場合、メニューバーの「ウィンドウ」→「ビューの表示」→「その他」をクリックし、追加する。（図 116）

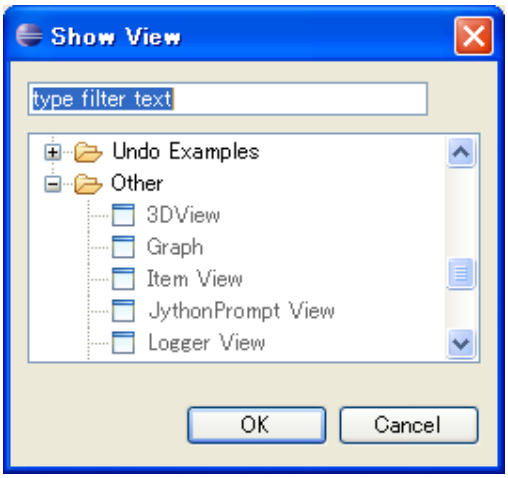


図 116　必要なビューの追加

シーン設定

Item View から”Model”を右クリックし「load」を選択する。(図 117)
ファイル選択ダイアログが開くので、読み込む VRML ファイルを選択する。

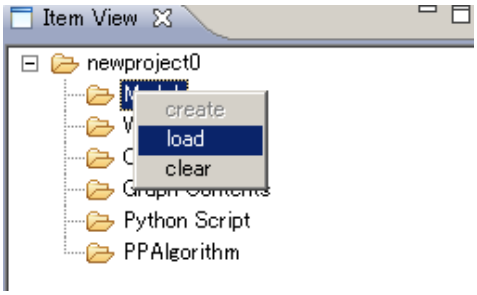


図 117　モデルのロード

この手順を繰り返してシーンに存在する全ての物体を読み込んだ後、干渉チェックを行うペアの設定を行う。以上のシーン構築手順は動力学シミュレータにおいてシーンを構築する手順と同一であり、動力学シミュレーション用に作成したシーン設定をそのまま移動動作設計に利用することも可能である。

計画アルゴリズムの設定

パラメータセットの作成

Item View から”PPArgolithm”を右クリックし「create」を選択する。(図 118)

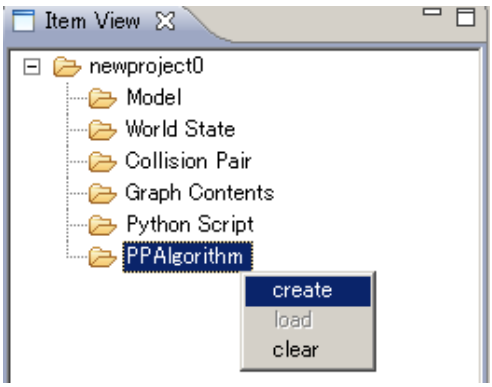


図 118　アルゴリズム設定の作成

計画アルゴリズムの設定

設定を変更したい PPArgolithm を Item View で選択し、Property View からパラメータを変更する。(図 119)

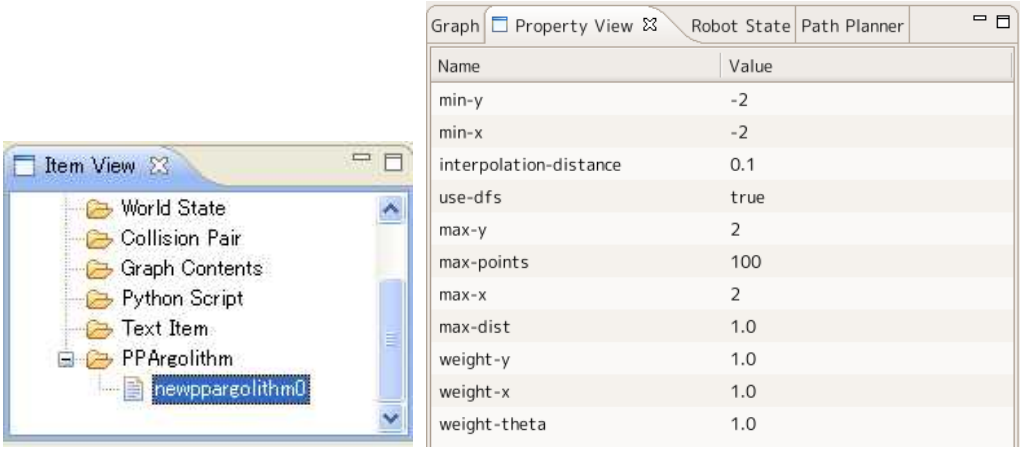


図 119　アルゴリズムの選択(左)と、そのプロパティ(右)

PRM、RRT の双方ともに共通するプロパティは以下のとおりである。

乱数最大点　・・・　ランダムサンプリングを行う範囲の最小 (min-x, min-y)

乱数最小点　・・・　ランダムサンプリングを行う範囲の最大 (max-x, max-y)

距離の重み　・・・　距離を算出する際の重み(weight-x, weight-y, weight-theta)

補間距離　・・・　干渉チェックを行う際に隣接する 2 点間の距離がこの値以下になるように補間される(interpolation-distance)。小さな値を設定すると 2 点間を細かく分割して干渉チェックを行うために干渉の発生を見落としづらくなる一方で計算時間がかかる。逆に大きな値を設定すると、干渉の発生を見落とす場合が生じる。

ランダムサンプリングを行う範囲は(-2,-2)～(2,2)の範囲に設定され、補間距離は 0.1、距離の重みは全て 1 に設定される。

PRM アルゴリズムは以下のプロパティを持つ。
近傍距離 ・・・ エッジを結ぶ近傍の距離（max-dist)。 2 点間の距離がこの値以下の場合、干渉を検査して干渉がなければ 2 点を結ぶエッジをロードマップに追加する。
サンプリング数 ・・・ ランダムサンプリングを行う数（max-points)
デフォルト値では、近傍距離が 1.0、最大点数が 100 個となっている。

RRT アルゴリズムは以下のプロパティを持つ。
近傍距離 ・・・ RRT の追加されるノードの距離（eps)
最大試行回数・・・ 探索を打ち切る回数（max-trials)
デフォルト値では、近傍距離が 0.1、最大試行回数が 10000 回となっている。

計画条件の設定

下図の PathPlanning ビューを用いて計画条件の各種設定を行う。

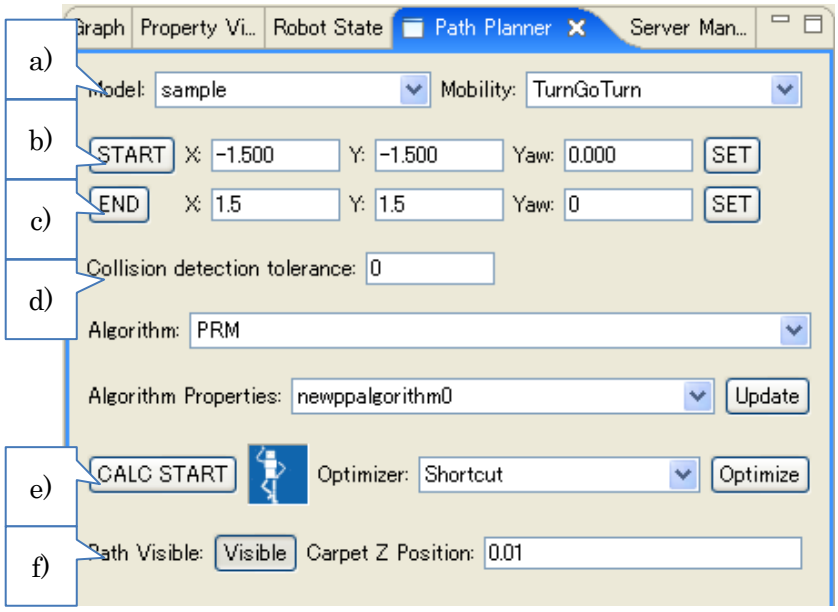


図 120 PathPlanning ビュー

a) ロボット及びその移動アルゴリズムの選択
動作させるロボットを、モデル選択ボックスから選択する。またそのロボットが持つ移動アルゴリズムを選択する。

b) 始点の指定
ロボットを希望の始点まで移動させ（図 121)、始点設定ボタン(2)を押す。
もしくは始点入力エリアへ数値を直接入力する。

c) 終点の指定

始点の指定同様、ロボットを移動させ（図 121）、終点設定ボタン(3)を押す。もしくは終点入力エリアへ数値を直接入力する。

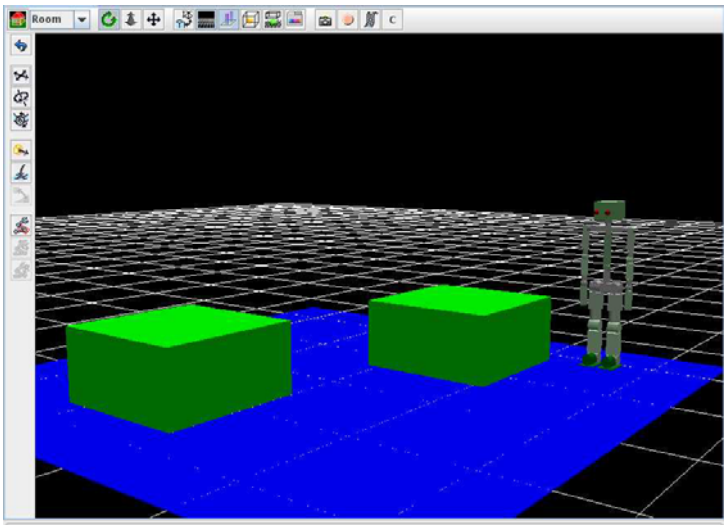


図 121　ロボットの移動

d) 干渉チェック時のトレランス

干渉チェック時のトレランスを設定する。干渉チェックを行う 2 つの物体の間の距離がこの値よりも小さい時、干渉が発生しているものと見なされる。これによって障害物との間に常に一定以上の距離を保つことが可能となる。

e) 計算開始

計算インジケータが表示され、計画中であることを示す。計算を中止する場合は再度ボタンを押す。

f) 計算結果の表示

計算が終了すると、経路表示ボタン(7)が選択可能になり、経路の表示・非表示を切り替えることができる。（図 122）

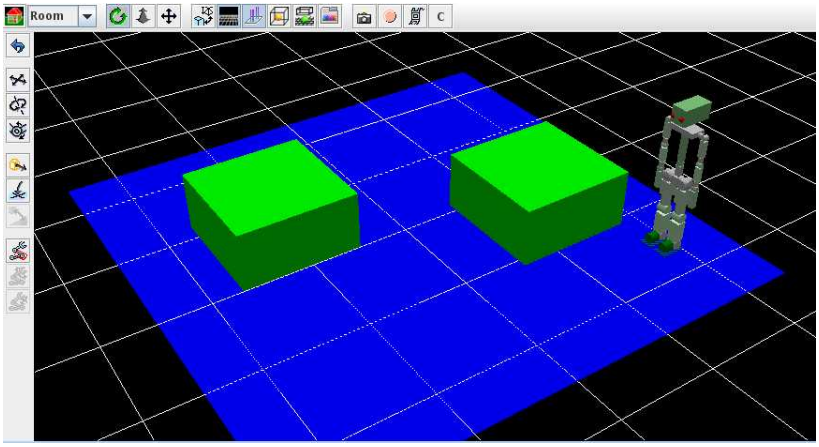


図 122　オフ状態の経路表示ボタン(左)と経路表示をオフにした状態(右)

また、Logger View 上のボタン（低速逆回し、高速逆回し、低速再生、高速再生、再生、ポーズの六種）およびスライダを動かすことで、経路上を移動させることができる。（図 123）

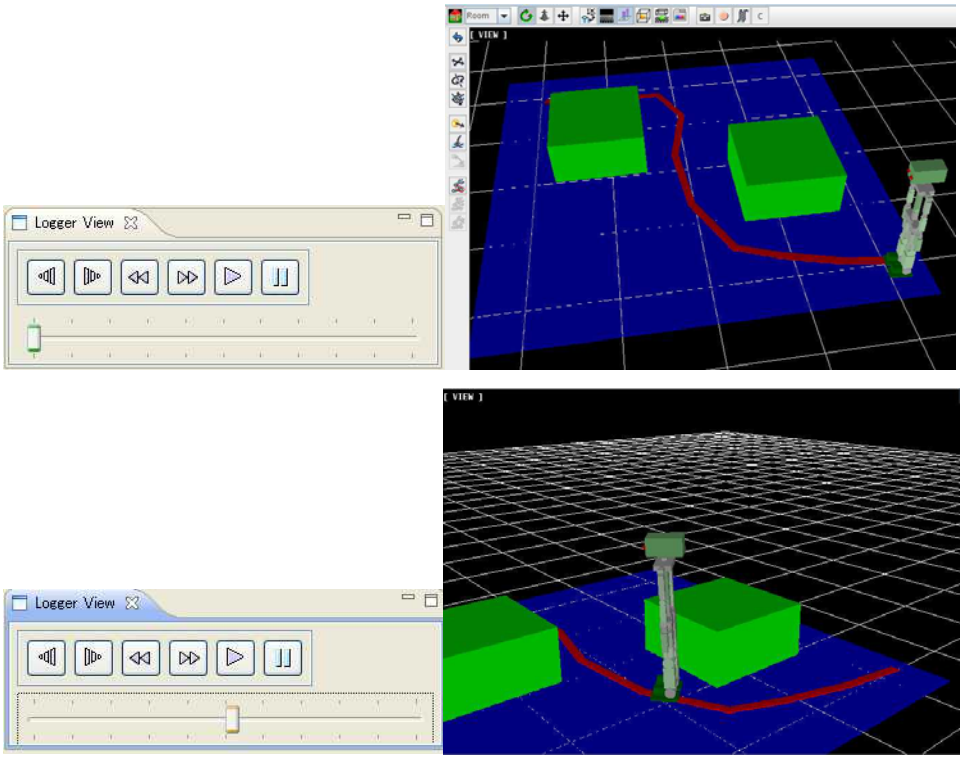


図 123　Logger View による経路上の移動

まとめ

開発した移動動作設計ツールは、動力学シミュレータに対する拡張モジュールとして実装されており、そのソースコード及びバイナリは動力学シミュレータと共

に一般に公開・配布されている。本ツールを用いることでリファレンスハードを始めとした様々なロボットで移動動作の計画が可能である。また計画エンジンは移動動作に限定された実装にはなっていないため、マニピュレータの動作計画等にも利用することが可能である。

(c) シミュレータ

作成されたロボットの動作及び作業シナリオの正当性・妥当性を、仮想世界を用いて検証するためのソフトウェアである動力学シミュレータ、およびで様々なロボットで利用が想定されている各センサの RT 部品機能のシミュレーション機能モジュール(RT コンポーネントシミュレータ)の開発を行った。

(c-1) 動力学シミュレータ

動力学シミュレータは、ロボットの動作及び作業シナリオの正当性、妥当性を仮想世界において検証するためのソフトウェアである。ロボットおよび作業環境を模擬可能な 3 次元シミュレーションを実現しており、他のツール群と連携して動作可能にし、一般公開を行った。本研究項目の最終目標は、ソフトウェアプラットフォームの他の機能との連携を強化し、事業化を検討することであった。

シミュレータのベースには、科学技術連携施策群の効果的・効率的な推進「次世代ロボット共通プラットフォーム技術の確立」において平成 19 年度に開発を完了した「分散コンポーネント型ロボットシミュレータ」OpenHRP3 を利用し、他のツール群との連携を容易にすると共に使い勝手の向上、機能強化を行った。以下、成果の詳細を述べる。

モデルローダの改良

OpenHRP3 においては、ロボットの機構・形状・物理パラメータ等をモデルファイルとして記述し、このファイルは「モデルローダ」というコンポーネントを用いて読み込まれるようになっている。モデル読み込みに関わる処理の効率と保守性を向上させることと、本プロジェクトで新たに策定したハードウェア仕様記述方式へと対応させることを目的として、モデルローダの改良を行った。

改良前のモデルローダとそれに関わるシステムでは、以下の問題があった。まず、モデルファイルの読み込み処理が非常に重く、複雑なモデルを扱うシミュレーションを実行する際の操作性が良くなかった。また、モデルローダはグラフィック描画する際に必要な情報の全ては提供しないため、システムにおいてモデル描画を必要とする各コンポーネントはモデルローダとは別個に直接モデルファイルを読み込む処理が必要となっていた。これにより、モデル読み込みに関わる処理を複数のコンポーネント間で共有化しシステム全体の保守性を向上させるというモデルローダの役割が、十分果たされていなかった。

以上の問題の解決と、新たに策定するモデルファイルのフォーマットへの対応を行うため、 モデルローダに関わるコンポーネントの改良を行った。
まず、改良前のモデルローダは Java で記述されたものであったが、この実装を検討しなおし、 C++言語でモデルローダのベースを記述し直した。これにより、ベース処理の高速化を実現し、ソースの見通しもよくなった。さらにこのベースに対して、拡張的な VRML 要素への対応や、プリミティブ幾何形状を三角形メッシュベースの統一的な幾何形状表現へと変換する機能の追加を行った。また、モデルローダの IDL はハードウェア仕様記述方式に基づいて更新し、新しい IDL へ他のコンポーネントを対応させる作業も行った。

GUI の Eclipse プラグイン化

OpenHRP3 の GUI は Java の標準 GUI ライブラリである Swing を用いて実装されていた。ただし、本プロジェクトで作成する他のツールのうち、Java で実装されるものは基本的に Eclipse のプラグインとしており、ツール間の統一性や連携を考慮すると、シミュレータの GUI も Eclipse のプラグインとなることが望ましい。このため、GUI 部分を Swing ベースのものから Eclipse のプラグインへと移植する作業を行った。 下図にユーザインタフェースの実行画面を示す。

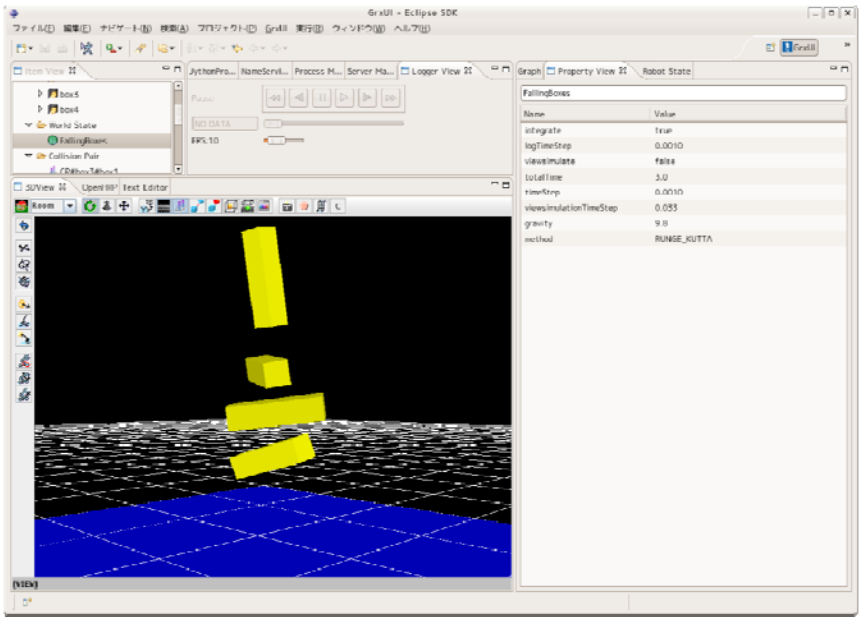


図 124　OpenHRP3 のユーザインタフェース

画面右の部分には、各種パラメータの一覧が表示され、編集が可能である。画面左下の部分にはシミュレーション世界の状況が 3 次元表示される。これらの他にグラフ表示を行うパネル等があり、タブで切り替えが可能となっている。上記画面の構成は一例であり、ユーザがウィンドウの配置やサイズ等を変更可能である。

距離センサシミュレーション機能

当初距離センサのシミュレーションは、OpenHRP の視野画像シミュレーション機能を用いて視野画像生成時のデプスバッファを取得し、距離を計算することで実現していた。しかしこのシミュレーション方法には以下のような問題があった。

- デプスバッファのデータ長が 32bit であり、距離の分解能が不足していた
- Java3D のバージョン 1.5 以降ではデプスバッファ取得部分にバグがあり、デプスバッファを取得するためには古いバージョンを使う必要があった
- Java3D によるオフスクリーンレンダリングが正しく機能しないため、視野画像を生成するためのウィンドウを常に手前に出しておく必要があった

以上の問題から、新たに半直線とポリゴン集合の干渉チェック機能を用いて距離センサのシミュレーションを行なうことが出来るようにした。OpenHRP では干渉チェックライブラリとして OPCODE を使用しており、これに含まれる干渉チェック機能を利用して実装している。

OpenHRP ではロボットや環境オブジェクトの記述に、VRML97 をベースにセンサ等を記述するためのプロトタイプノードを追加したものを用いている。距離センサを記述するためのプロトタイプノードとして以下のノードを追加した。

```
PROTO RangeSensor [
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField MFNode     children    [ ]
  exposedField SFInt32    sensorId    -1
  exposedField SFFloat    scanAngle   3.14159
  exposedField SFFloat    scanStep    0.1
  exposedField SFFloat    scanRate    10
  exposedField SFFloat    maxDistance 10
]
{
  Transform {
    rotation    IS rotation
    translation IS translation
    children    IS children
```

```
    }
}
```

このノードを距離センサが取り付けられているリンクの子ノードとして定義し、搭載されているセンサの仕様に応じたパラメータ設定を行なうことで距離センサシミュレーションが可能となる。センサの出力は scanRate で設定した周期で仮想世界に存在するロボットに対応する RT コンポーネントの出力ポートから出力される。下図に距離センサを用いたシミュレーション中の実行画面を示す。半透明で示されているのは距離が検出されている部分である。

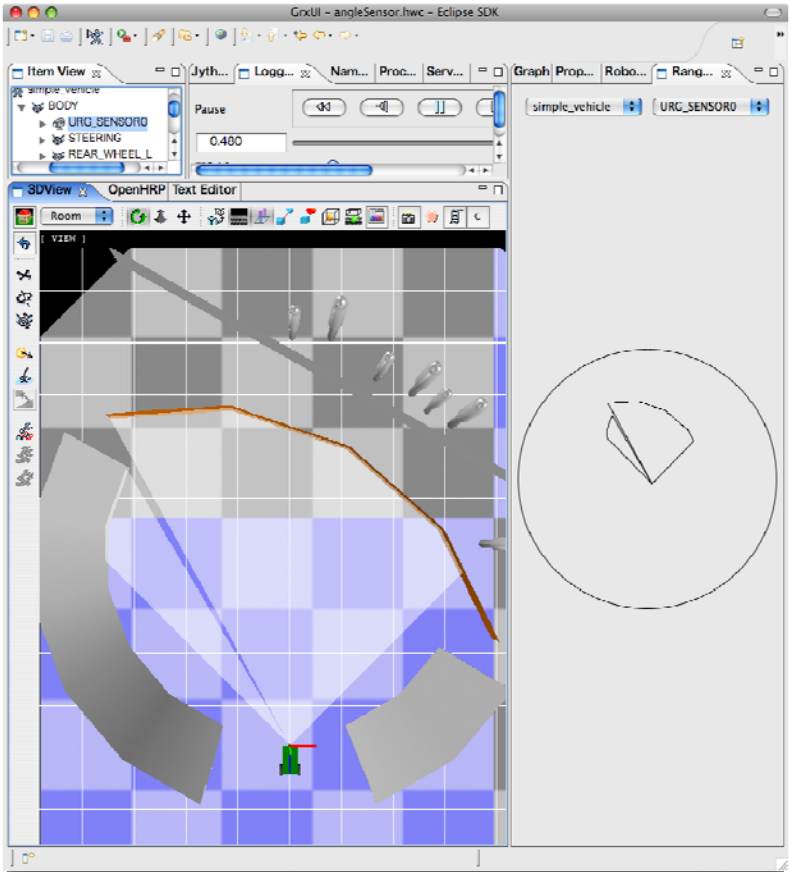
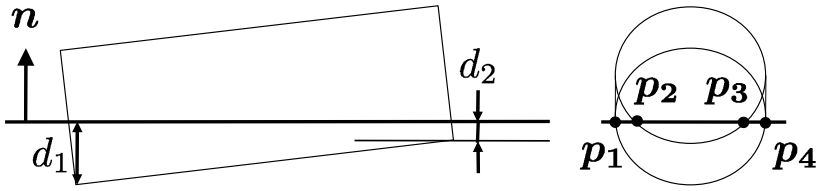


図 125　距離センサを用いたシミュレーション中の実行画面

円筒・平面干渉チェック機能

OpenHRP ではあらゆる形状は 3 角形分割され、物体間の干渉チェックはこの 3 角形ポリゴン集合に対して行なわれてきた。直方体や球といったプリミティブ形状しか扱うことの出来ないシミュレーション環境が多い中で、ポリゴン集合を用いることでどのような形状でも扱えるという利点があった。一方で車輪型移動ロ

ボットのシミュレーションをする場合に、本来真円であるはずの車輪を多角形で近似するために車輪が回転すると車体が上下に振動するという問題が発生した。細かくポリゴン分割を行なって真円に近づけることで振動は軽減できるが、処理するポリゴン数が多くなるためにシミュレーション速度が低下してしまう。
この問題に対処するため、円筒と平面の干渉チェック機能を追加した。円筒と平面の間の干渉チェックが呼び出された場合に、円筒をポリゴン分割せずに干渉点、干渉法線、干渉深さを計算する機能である。干渉点、干渉法線、干渉深さはそれぞれ以下のように決定される下図参照）。



- [干渉点] 円筒の両端の円が平面と交わる点（p1～p4）
- [干渉法線] 平面の法線(n)
- [干渉深さ] 円筒の両端の円上の点で平面からの距離が最も遠い点と平面との距離(d1, d2)

図 126　円筒と平面の干渉チェック

これにより、車輪形状を多角形で近似することなくシミュレーションを行うことが可能となった。この機能は干渉チェックを行う 2 つの物体の形状データが円筒と平面として与えられた場合に適用される。

ハイブリッドシミュレーション機能

OpenHRP の動力学計算エンジンでは関節（モータ等で駆動される回転、並進関節だけでなく、機構ツリーのルートにあたる自由関節も含む）のシミュレーション方法として 2 種類の方法を提供している。1 つ目は関節トルクを入力として与える方法、2 つ目は関節の位置、速度、加速度を与える方法である。前者をトルク制御モード、後者をハイゲインモードと呼んでいる。ハイゲインモードの場合、その関節は与えられた軌道通りに動くことになる。
モバイルマニピュレータを用いてマニピュレーションを主な対象としたシミュレーションを行なう場合、車輪の制御部分は省略して台車部分は計画した軌道通りに移動することとして、マニピュレーション部分のみを精密に検証する、といった利用法が考えられる。このような利用方法を可能とするため、ハイゲインモードとトルク制御モードで制御される関節が混在する環境のシミュレーションを可能とした。

モデルファイル作成機能

ロボットや環境オブジェクトのファイル形式は VRML97 を用いたものであるが、出力形式として VRML97 を扱える CG や CAD ソフトは存在するものの、VRML97 を読み込んで編集して再度出力することが出来るようなソフトウェアは皆無であり、従来はテキストエディタで編集するしかない状態であった。

そこで OpenHRP のグラフィカルユーザインタフェースにモデルファイルの作成機能を追加した。前述のように VRML97 形式で出力可能な CG, CAD ソフトウェアは存在するため、形状データに関してはあらかじめこれらのソフトウェアを用いて作成しておくこととし、この作成機能の主な役割は、それらのパーツを組み上げて機構ツリーを作成し、質量や重心といったパラメータを入力し、ロボットに搭載されたセンサを定義することとした。

以下にモデル生成の手順例を示す。

a) 新たに作成するロボットに対応するモデルノードを作成する（下図左）。ノードを作成するとその子ノードとして一つのリンクが自動的に生成される（下図右）。

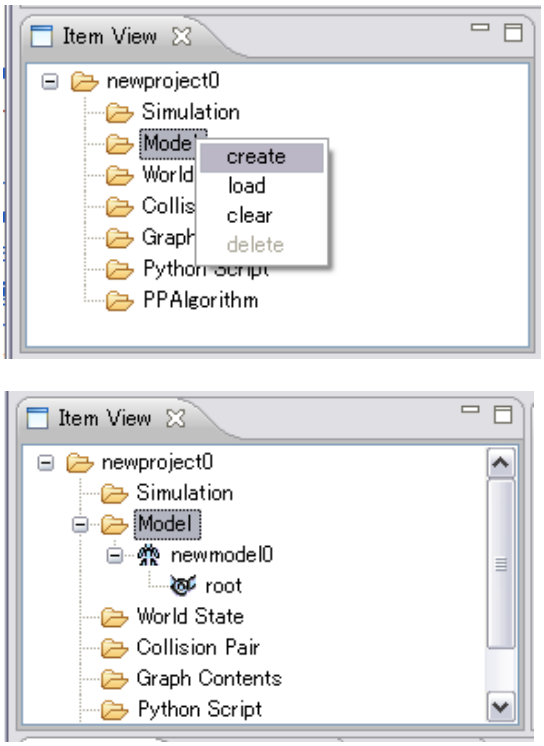


図 127　ロボットのモデルノードの作成

b) 機構ツリーを作成する。自動的に生成されたリンクに対して子リンクを追加

したり、そのリンクに取り付けられているセンサを追加したりしながら機構ツリーを構築する。また移動台車とマニピュレータ等、既存のロボットを組み合わせて新たなロボットを作成する場合にはそれらのモデルファイルを読み込んで利用することも可能である。

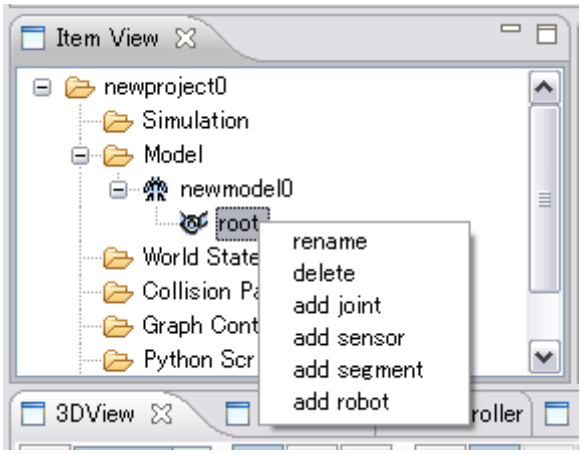


図 128　モデルファイルの読み込み

c)　形状データを設定する。形状データの設定時にはプリミティブ形状を利用する方法と CAD ソフト等で生成した VRML ファイルを利用する方法がある。プリミティブ形状を利用する場合は直方体、円錐、円筒、球が形状として利用できる。

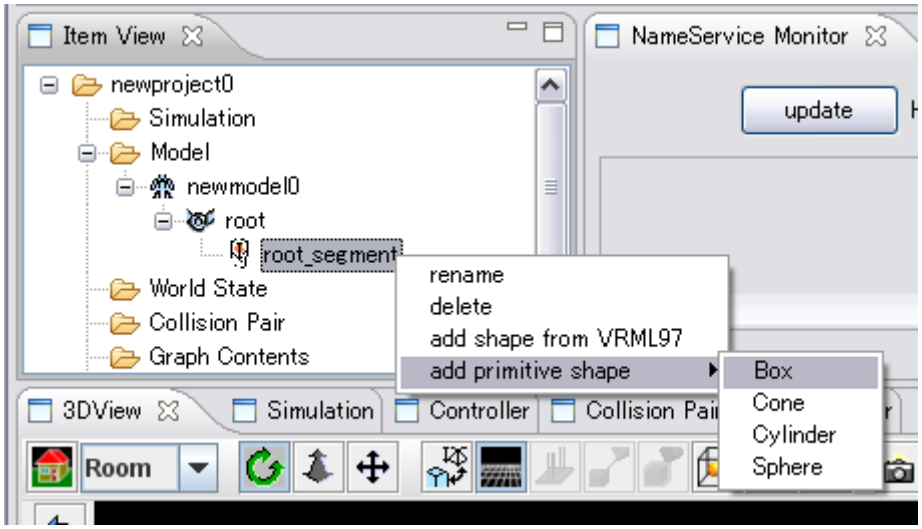


図 129　プリミティブ形状によるモデルの作成

d)　各種パラメータを設定する。質量、重心位置、慣性モーメント等の動力学パラメータやリンク間の相対位置、姿勢、センサの特性情報といった各種パラメータをプロパティパネルを用いて設定する。下図は関節に対するプロパテ

ィである。

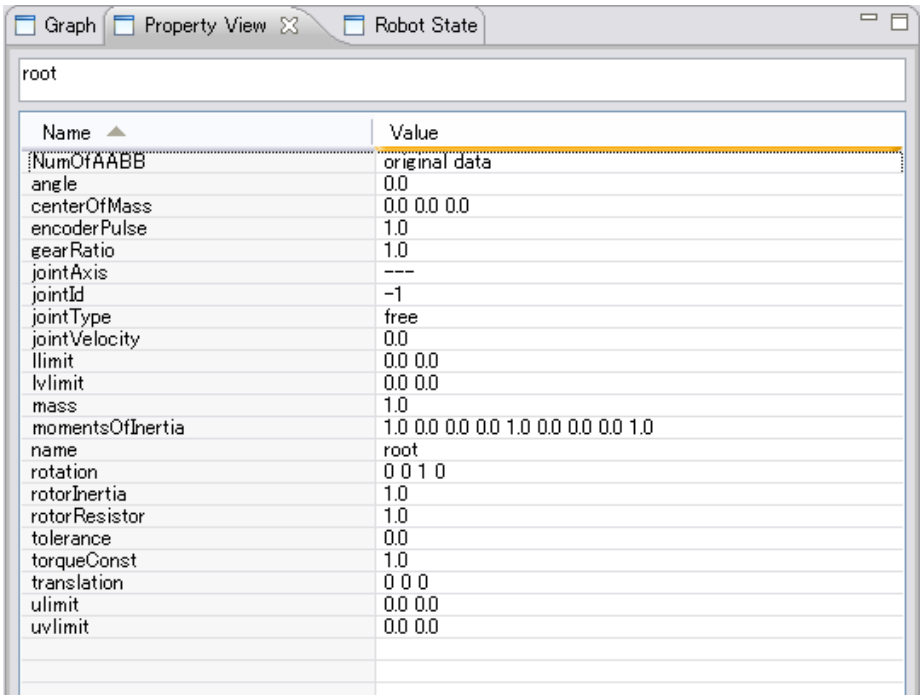


図 130　関節角に対するプロパティパネルの例

最後に作成したモデルをファイルに保存して完了である。下図に車輪移動型ロボット（下図左）のモデルを作成した際の最終的なツリー（下図右）を示す。

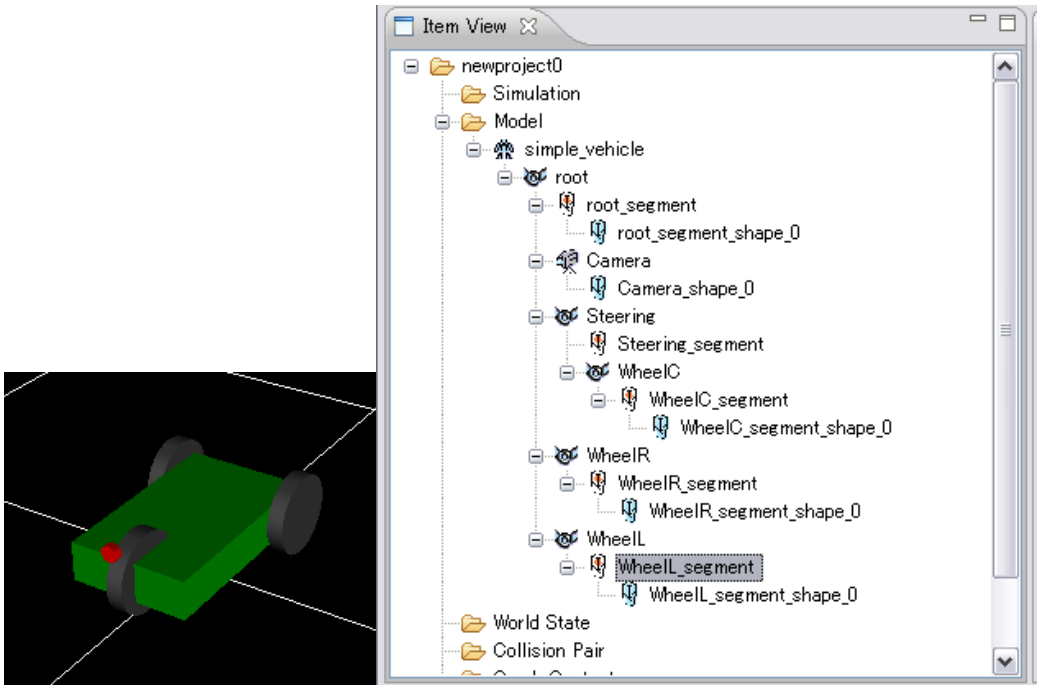


図 131　車輪移動型ロボットのモデルの例

近似形状を用いた干渉チェック

CAD モデル等から生成されたロボットの形状モデルは、形状が精密に表現される一方でポリゴン数が多くなり、シミュレーション時の干渉チェック処理時間を増大させる要因となる。ロボットのソフトウェア開発の初期段階においては形状の正確さよりもシミュレーションが短時間で実行できることが必要となる。そこで近似形状を用いて干渉チェックを行うことで処理時間を短縮する機能を追加した。近似形状をユーザが与える方法も考えられるが、ユーザの負担が増大するため、シミュレータが自動的に近似形状を生成することとした。近似形状の生成には、干渉チェックを効率的に行うために使用されているデータ構造 BVH（Bounding Volume Hierarchy）を利用した。これはロボットの形状を BV（Bounding Volume）のツリーによって近似したものであり、開発したシミュレータの場合には BV は AABB（Axis Aligned Bounding Box）で表現される。このツリーはルートに近づく程、少数の AABB によって形状が表現される。ユーザはロボットの形状をいくつの AABB で近似するかを指定することができ、開発の段階に応じて近似度合いを変更することが可能である。

下図に AABB の個数を設定している画面を示す。①の部分で個数を指定する対象となるロボットのリンクを選択し、②の NumOfAABB というプロパティに対して所望の AABB の個数を入力することで形状の近似が行われる。③は近似形状の表示の ON/OFF を切り替えるボタンである。下図では近似形状が白線のワイヤフレ

ームで表示されている。

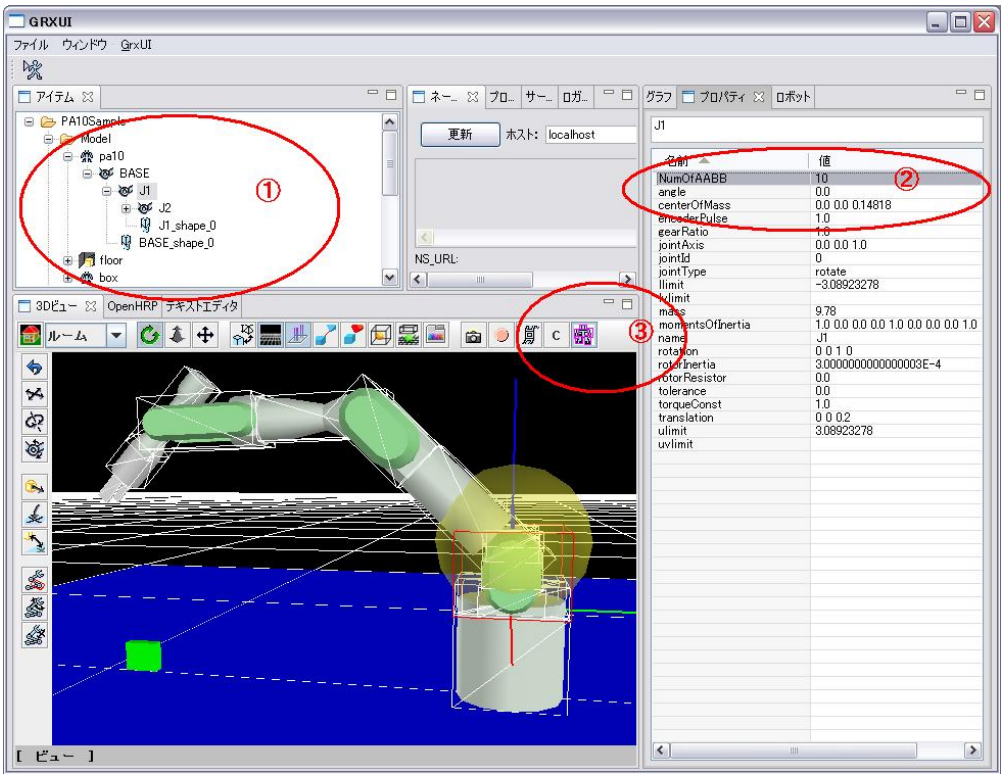


図 132　近似形状を用いた干渉チェック

シミュレーションに同期した RT コンポーネントの実行機能

RT コンポーネントで構成されたロボットのコントローラと組み合わせてシミュレーションを行なう場合には、RT コンポーネント群はシミュレーション時間に同期して実行される必要がある。シミュレーションの実行速度が実時間に対して大きく異なっている場合に RT コンポーネントが通常通り実際の時間に従って動作していたのでは、シミュレーション時と実験時で挙動が異なってしまうことが予想されるためである。

RT コンポーネントの実行を制御する部分、実行コンテキストはユーザが自由に入れ替えることが可能となっているため、新たにシミュレーション時刻に同期して RT コンポーネントを実行する実行コンテキストを開発し、シミュレータ本体にこれらの実行コンテキストに対してクロックを供給する機能を追加することでシミュレーションと RT コンポーネントの同期実行を可能とした。

シミュレータ本体に追加したサーバ機能のインタフェースを以下に示す。

```
module OpenHRP {
  interface ClockGenerator{
    void subscribe(in OpenRTM::ExtTrigExecutionContextService ec,
```

```
                    in double period);
    void unsubscribe(in OpenRTM::ExtTrigExecutionContextService ec);
  };
};
```

新たに作成した実行コンテキストはこのサーバ機能のクライアントとして動作する。実行コンテキストが生成される際に subscribe()を呼び出して自分を実行して欲しい時間間隔と共に登録する。シミュレータはシミュレーションを実行するとともに登録されている時間間隔で実行コンテキストに対してクロックを送信し、実行コンテキストが RT コンポーネントを駆動する。

まとめ

開発した動力学シミュレータはオープンソース形式で一般に無償配布を行なっている。インストールの手間を軽減するためバイナリパッケージでの配布も行なっている。

（c-2）RT コンポーネントシミュレータ

本研究項目では、RT コンポーネントシミュレータとして、RT 構成要素のシミュレーションが行えるシミュレータを開発し、動作設計ツール、作業シナリオ作成ツールによって生成された作業シナリオ、動作を仮想環境内でシミュレートし、検証するための環境を提供することを目標とした。最終目標は、中間目標成果物に対して、プラットフォームのバージョンアップ対応、バグフィックスを行うとともに、新規デバイス向けのＲＴコンポーネントシミュレータ及び基本 RT コンポーネントを４種以上追加し、本プロジェクトの外部に対して公開、事業化を行うことであった。以下、開発した RT コンポーネントシミュレータについて述べる。

成果の詳細

DADS や ADAMS のような商用ソフトウェアや ODE のようなオープンソースソフトウェアにおいて動力学シミュレーション機能は提供されているが、ロボットのアプリケーションを開発する上で必要なセンサのシミュレーション機能は十分でない場合が多い。そこで、我々は、動力学計算によってシミュレート可能な加速度センサ、ジャイロ、力センサ等に加えて視覚センサや距離センサ等のシミュレーション機能を開発し、RT コンポーネントとして提供した。

さらに、仮想環境内で検証した RT システムを実環境にシームレスに移行できるよう、RT コンポーネントシミュレータと同じ仕様を持つ、実デバイスを用いた基本 RT コンポーネントの開発も行った。実デバイスは、レーザレンジファインダやネ

ットワークカメラなど、知能ロボットシステムでよく利用される各種センサを対象とした。これらを RT ミドルウェア対応のロボットシステムで利用可能なようにモジュール（RT コンポーネント）化している。さまざまなロボットシステムで利用されている各種センサをモジュール化することで、ロボットシステムの開発効率の向上が期待できる。また、センサ種別毎に共通インタフェースを定義しているため、センサモジュールのバージョンアップ、交換も容易なものとした。モジュール化したセンサの実デバイスの例を下図に示す。

図 133　モジュール化した実デバイス例

RT コンポーネントシミュレータは、「分散コンポーネント型ロボットシミュレータ」OpenHRP3 と連携して動作するセンサシミュレーション用の RTC である。ロボットシステムにおいて、各種センサデバイスはなくてはならないものであるといえる。移動ロボットの分野では環境認識のために距離センサや加速度センサが用いられ、産業用ロボットの分野では物体認識のために力覚センサなどが組み込まれることが多い。OpenHRP3 では、センサデバイスモデルが出力する力覚データや加速度データなどの物理データを動力学計算によりシミュレーションすることが可能である。OpenHRP3 単独でも、各種デバイスの情報を出力することが可能であるが、RT コンポーネントシミュレータを併用すれば、センサ固有の機能や特性を考慮してデータを出力することが可能になる。OpenHRP3 から得られるセンサシミュレーションデータは、出力形式が予め決められている。ロボットシステム構築において、センサ類は独自開発のものではなく、センサメーカーが販売している既製品を利用することが少なくない。RT コンポーネントシミュレータは、これらの既製品を利用しているロボットシステムに対し、より正確なシミュレーションを実現するために有効な手段となる。RT コンポーネントシミュレータでは、センサ毎に固有のデータ分解能や測定範囲といった特性を考慮しており、図に示すように OpenHRP3 が出力するデータを RTC 内で加工する役割を果たし

ている。

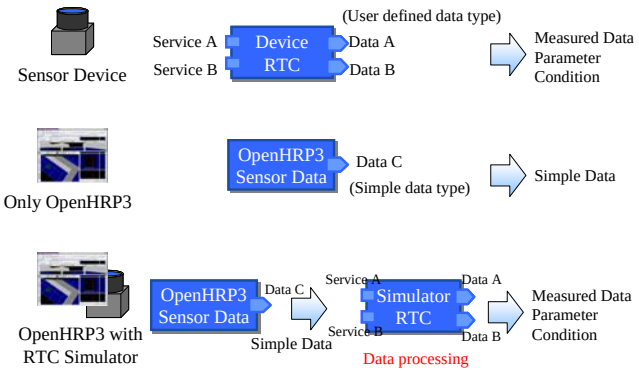


図 134　RT コンポーネントシミュレータの機能概念図

また、RT コンポーネントシミュレータのデータ出力インタフェースは、実デバイス RTC のインタフェースと同一にしているため、仮想環境（OpenHRP3）内で検証した RT システムを実環境へシームレスに移行することも可能である。

OpenHRP3 は、平成 20 年度末に OpenHRP3.1.0-beta がリリースされた。GrxUI が Eclipse プラグインとして動作するようになったほか、モデルローダ機能の実装を Java から C++に移行するなどの変更が行われた。中でも、RT コンポーネントシミュレータに関連する大きな変更として、距離センサ機能（Range Sensor View）の追加が挙げられる。

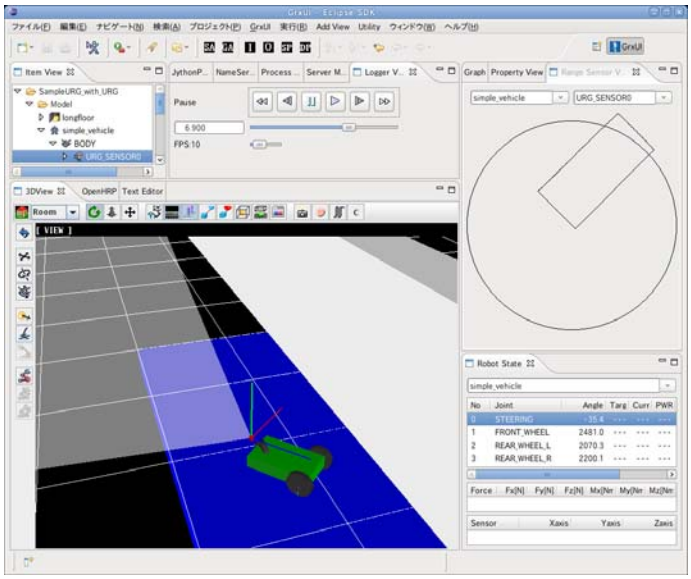


図 135　OpenHRP3.1.0-beta GrxUI

OpenHRP3.0.3 までは、OpenHRP3 において利用可能なセンサシミュレーション機能は動力学シミュレーションにもとづくデータのみであったため、距離情報については、直接取得することができなかった。そこで、RT コンポーネントシミュレータの距離センサシミュレーションとしては、Java3D の機能を利用して

3DView に表示されるモデルのデプス値（深度）を取得していた。このデプス値を以下の式により距離情報に変換し、RTC から出力する方式を採用していた。

$$Z = \frac{fn}{d(f-n)-f}$$

Z: distance(z-axis)、 f: far clipping、 n: near clipping、 d: depth value

しかし、このデプス値は式からも分かるように、計測する物体が遠くになればなるほど、距離データの精度が落ちるという性質がある。距離センサのうち、例えば SICK 社の LMS2xx シリーズは最大で 80m の距離まで計測可能であるが、デプス値を使った方法ではシミュレーション世界で十分な精度が得られない。また、Java3D に起因する制約から、3DView（Vision Sensor Window）を PC の画面上で前面に表示していなければ、距離情報を取得できないという問題もあった。
そこで、OpenHRP3.1.0-beta では、モデル間の干渉検出を行うための干渉チェック機能を用いて物体までの距離情報を取得する方法が採用された。3DView に依存しない形式となったため、動力学計算で得られる加速度情報などと同じように、Controller Bridge から距離情報を出力可能となった。この OpenHRP3 の機能向上に伴い、RT コンポーネントシミュレータとしても、その実現方法を見直すこととした。OpenHRP3 の Controller Bridge のデータ出力は、コントローラ RTC（Virtual Robot RTC）の OutPort から出力される。そこで、RT コンポーネントシミュレータをコントローラ RTC の OutPort と接続可能な InPort を持つ RTC として実装した。システム構成を以下に示す。

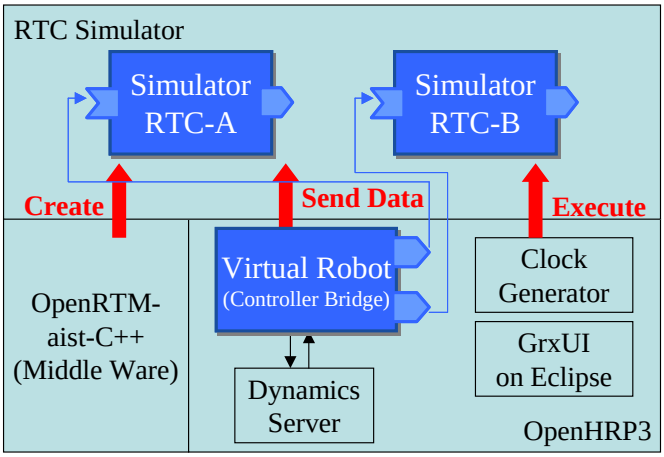


図 136　RT コンポーネントシミュレータのシステム構成

図中の RTC Simulator は RT コンポーネントシミュレータ本体であり、RT ミドルウェアの C++実装である OpenRTM-aist-C++の RTC として動作する。Virtual Robot (Controller Bridge)は、OpenHRP3 とのインタフェースを担うコンポーネ

ントであり、RT コンポーネントシミュレータでは、距離情報を Controller Bridge から取得している。

まとめ

ここで開発した RT コンポーネントシミュレータは、ロボット知能ソフトウェアプラットフォームの有効性検証の検証用移動知能モジュール群の開発および検証で利用され、開発／検証効率の向上に寄与した。プロジェクト期間中、RT コンポーネントシミュレータは、OpenRTM-aist、OpenHRP3 のバージョンアップ等に対しての改修を継続して行った。また、実デバイスの RTC のソフトウェアとマニュアルは、セックのロボットサイトにて無償公開している。NEDO 次世代ロボット知能化技術開発プロジェクトの参画機関を始め、その他の大学や企業でも多数利用されており、RT ミドルウェアの普及や開発の効率化に貢献している。

(d) 実時間ソフトウェア設計ツール

本開発項目では、実時間シミュレーション機能、実時間制御機能、実時間シミュレーションデバッグ機能の 3 つの機能を実現するため、「分散コンポーネント型ロボットシミュレータ」の成果を Eclipse プラグインとして実装し、本事業内で実装する他のツールとの連携をすることである。最終目標として、実時間ソフトウェア設計ツールに関しては、複数スレッド上での設計・デバッグ機能を追加し、Linux Preemptive OS 上で、機能を検証することであった。以下、成果の詳細を述べる。

実時間ソフトウェア設計ツールは実時間性を確認する部分を RT コンポーネントのうちの複合コンポーネントを仮定し、システムに組み込まれた RTC 群が与えられた時間内に終了するかどうかを検討するためのツールである。OpenRTM の利点はコンポーネント同士を容易に接続することができる点にある。しかし机上で様々な RT コンポーネントをシステムに導入し、機能的要求仕様を満足させたとしても、実際にシステムを動かしてみると各コンポーネントの実行時間と通信オーバーヘッドにより、期待した実時間性能が満たされないケースがでてくる可能性がある。この場合、正確な見積りは難しいとしても、およそループが 1 周するのにかかる時間などを知ることができれば、想定しているタスクが実現可能か、もしくは環境の変化に合せた動作が可能かどうかの判断が可能となる。

今後 RT コンポーネントの種類が増加していく中で、各 RTC の実行時プロファイルのデータベースが蓄積されれば、RT コンポーネントを組合せた RT システムの実現の可否を知る事はもちろん、同種の機能を持った別の RTC に切り替えた場合のパフォーマンスと比較することで、システムを構成する場合に必要十分な機能を持った構成にしたり、実行時間により余裕を持たせた構成にしたりといった設計時の選

択が容易に行える事が期待できる。

実時間シミュレーション機能

実時間シミュレーション機能は、分散コンポーネント型ロボットシミュレータ（以下 OpenHRP3）の成果を利用して、同シミュレータ上で動作する RTC 化されたコントローラフレームワークをベースとして開発を行った。ここで必要となる機能は以下のようになる。

- RTC をシミュレーションで実行する機能
- シナリオデータ読込み機能
- シミュレーション結果表示機能
- シミュレーション結果保存機能

RTC をシミュレーションで実行する機能

双腕ロボットをプラットフォームとして開発されたロボット制御ソフトウェアでは、RTC 群からなる上位側コントローラと、デバイスとの通信を行う下位側コントローラが存在し、両者は共有メモリを介して状態量や関節角度指令情報を共有している。上位側コントローラでは、実時間で同期が必要なロボットの運動制御を行う RTC 群を単一の実行コンテキスト上で動作させており、下位側のコントローラはシングルスレッドで実行される。この二つの実行コンテキストは、下位側から上位側へのシグナルを利用して同期をとりながら協調動作している。

OpenHRP3 でこれを実現するために、以下の実装を行った。

- 下位側コントローラの RTC への置き換え
- 周期毎に複数の RTC に対する tick 送信機能をシミュレーションスケジューラに追加

シナリオデータ読み込み機能

ある状態で実行可能な動作リストからなるシナリオリストを読み込んで、現在実行可能な動作ボタンのみを配置させ、対話的にシナリオを実行する簡易 GUI を作成した。簡易 GUI は python で記述しており、ロボットの起動、初期化ルーチンも含めたサンプルとして、パッケージ化し、ユーザ向けに公開した。

シミュレーション結果表示機能

各 RTC の実行時間計測結果を取得し、表示するために、Eclipse Plugin として以下の View を実装した。

Benchmark Operator View … RTC の実行時間計測結果を取得・表示する

Benchmark Explorer View … 過去の計測結果を選択する

Timing Chart View …　実行コンテキスト毎の計測結果をグラフ表示

下図はその実行画面である。

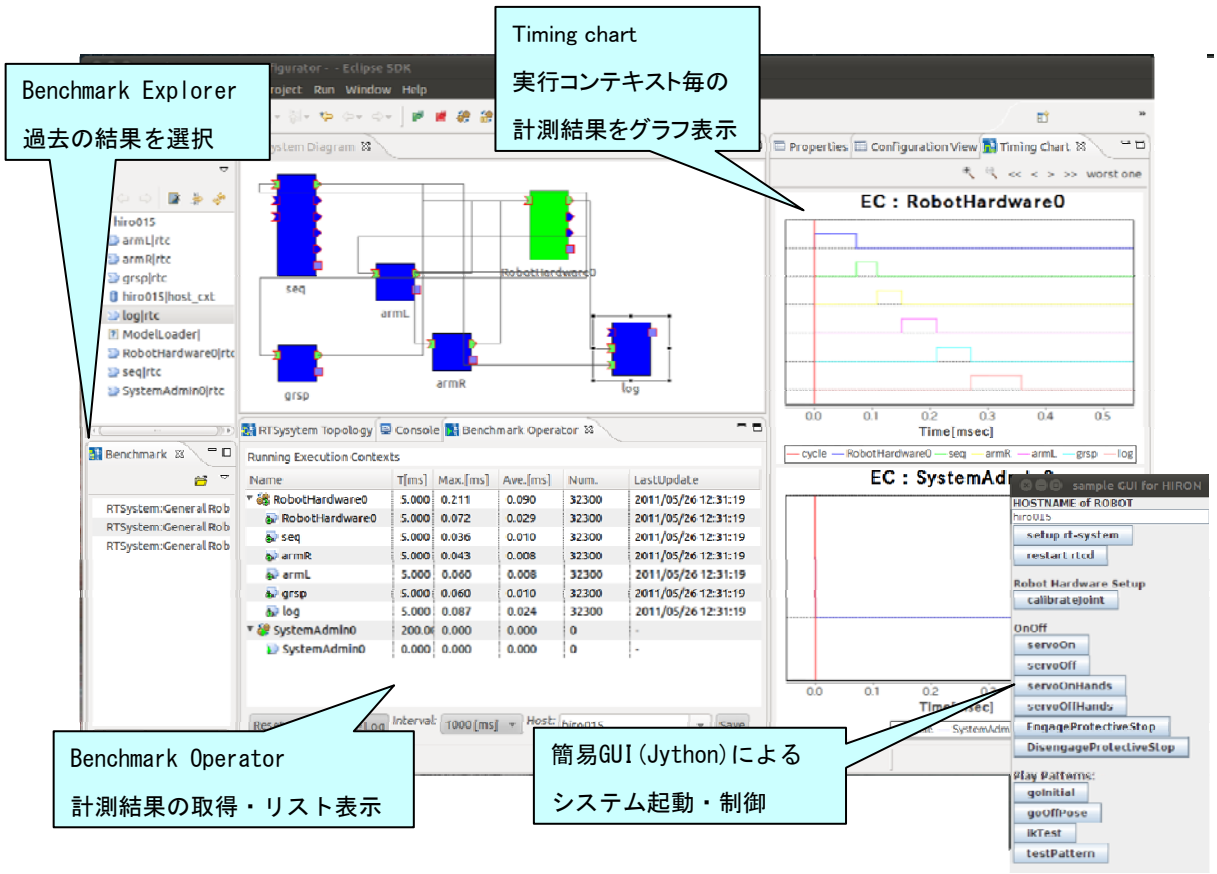


図 137　実時間ソフトウェア設計ツールの実行画面

シミュレーション結果保存機能

シミュレーション結果の保存機能は、Benchmark Operator View の Save Log ボタンとして実装した。保存したデータは、Benchmark Explorer 上で選択する事で、Benchmark Operator View や Timing Chart View に表示される。

実時間制御機能

実時間制御機能はロボットハードウェアを制御するための機能である。具体的な機能として以下のようなものが上げられる。

- シナリオデータ読込み機能
- RTC をロボットで起動するための機能
- ロボットで実行されている RTC からデータを読み出して表示する機能

シナリオデータ読み込み機能

RTC をシミュレーションで実行する機能の追加により、ロボット実機の下位側コントローラを置き換えた RTC 以外の RTC については、シミュレーションと同じバイナリを利用可能とした。下位側コントローラを置き換えた RTC では、スクリプトからは参照されないため、シミュレーションと実機とで、全く同じスクリプトを利用可能とした。

RTC をロボットで起動するための機能

シミュレーションでは、シミュレーション開始毎にコントローラが起動する。一方ロボット実機では、下位側コントローラはロボットホスト起動時から実行開始されシャットダウンの瞬間まで動作しつづけるが、上位側コントローラについては、はじめに RTC を統括する RTC マネージャのみが起動し、その後スクリプト言語 python によって各種 RTC のセットアップが実行される。RTC をロボットで起動するためには、RTC マネージャにアクセスして、RTC の読み込み・インスタンス生成、RTC 間の実行コンテキスト共有、ポート間接続等を実行する必要がある。これを python、もしくは jython で実行可能とするためのユーティリティスクリプトは、オープンソースプロジェクトとして産業技術総合研究所から配布されている。

hrpsys-base プロジェクト：http://code.google.com/p/hrpsys-base/

各 RTC を簡単に起動できるようにするために、このユーティリティスクリプトを利用したサンプルスクリプトと簡易 GUI を作成し、ネットワークインストールできるようにソフトウェアパッケージ化してユーザ向けに公開した。スクリプトはシミュレーションと実機とで完全に互換性があり、接続先のホスト名を変更するだけで、シミュレーションでも実機でも利用できるようになっている。スクリプトのカスタマイズ方法については、弊社サポートサイトのオンラインマニュアルにてユーザ向けに公開した。

ロボットで実行されている RTC からデータを読み出して表示する機能

ロボットで実行されている RTC からデータを読み出して表示する機能として、ツール側には以下を実装した。

- 単一 RTC の実行時間情報を取得する機能
- 複合 RTC の実行時間情報を取得する機能
- 取得した実行時間情報を元に、各 RTC の動作の実時間性を検証する機能

実行時間情報を取得する機能については、OpenRTM 側の仕組みが存在しないため、以下の機能を実行コンテキスト側に新たに実装した。その結果、実行コンテキストを専用のものに切り替える必要はあるものの、ユーザの RTC 自体を修正することなく実行時間の取得が可能となっている。

- 登録された RTC の実行時間をロギングする機能
- 外部クライアント側にロギングデータを提供する機能

各 RTC 動作の実時間性検証に関しては、実行コンテキストに設定された実行周期を基準として、以下の機能を前述の Eclipse Plugin に対して付加した。

- RTC 群の実行時間合計が、設定時間の 1/2 倍を超える場合には数値を黄色で表示
- 設定時間を超える場合には赤色で表示
- Timing Chart View 上に、実行コンテキストの実行周期を表示

実時間シミュレーションデバッグ機能

実時間シミュレーションデバッグ機能とは、シナリオから生成したスクリプトを実行シミュレーション結果に基づいて、実時間性を実現させるための微調整等を行なう機能である。これを実現させるためには次の機能が必要となる。

- シナリオで使用する RTC の読み出し機能
- RTC のデータ保存機能
- シナリオ中のクリティカルパスを実行するために必要な時間を計算する機能
- クリティカルパス実行中に他のクリティカルでないサービスに割り当てられる時間の調整

シナリオで使用する RTC の読み出し機能

対象ロボットのホスト名を BenchmarkOperator View 上の host 欄に入力し、Update ボタンを押すと、上で起動した RTC 群は一定時間毎に全て読み出され、実行時間計測結果を表示する BenchMarkOperator View 上に一覧表示される。

RTC のデータ保存機能

読み出された RTC 群は、BenchmarkOperatorView に実装された RTC のデータ保存機能により、実行時間計測結果や実行環境情報と共にファイルに保存される。保存された計測結果は、Benchmark Explorer 上で選択する事により、後から内容を閲覧可能となる。

クリティカルパス実行に必要な時間を計算する機能

シナリオ中のクリティカルパスを実行するために必要な時間は、対象となる実行コンテキスト上に追加された RTC 群の実行時間を全て足し合わせ、BenchmarkOperatorView 上に表示させた。複数の View から参照できるように、OpenRTM の eclipse 向けライブラリをベースとしてデータモデルを構築し、表示対象が切り替わる度に実行結果を再計算する。

まとめ

以上、今回開発した実時間ソフトウェア設計ツールについて述べた。開発したツールは、Eclipse plugin として実装しており、下記の EclipsePlugin リポジトリにオープンソースライセンスで公開している。ソースコードは、下記の google code サイトにプロジェクトとして登録済みである。

Eclipse repository：http://www.generalrobotix.com/update/
GoogleCode Project:
http://code.google.com/p/hrprtc-grx-realtimesystem-configurator/

Eclipse Plugin、OpenHRP3、各種ロボット制御 RTC を含めた全てのソフトウェアは、簡単なコマンドでネットワークインストール可能とし、導入時のユーザの負担を大幅に軽減させた。本ツールは、後述する双腕ロボットプラットフォームの実装に利用され、Linux Preemptive OS 上で機能検証ができた。

(e)双腕プラットフォーム向けソフトウェア整備

本研究項目は、プロジェクト内で開発された知能モジュールの統合検証システムの 1 つとして平成 22 年度から実施された「HIRO 加速案件」にて利用される双腕ロボットプラットフォームのソフトウェア整備を実施したものである。本研究項目は、当初の実施計画にはなかったがプロジェクトの加速化に寄与するために実施した。以下、実施したソフトウェア整備について述べる。

OpenRTM を商用実時間 OS へポーティング

OpenRTM を商用実時間 OS へのポーティングを行い、双腕ロボット向けのソフトウェアコントローラを構築した。
コントローラを構成するコンポーネントの多くをオープンソースでユーザに提供し、ユーザから見てブラックボックスとなっているコンポーネントをなくし、ユーザにコンポーネント開発を促すことも目指した。
コントローラシステムの基本コンセプトは次のようになる。

- コンポーネント制御方法によりコントローラ内で置く場所を決める
- ビジョンを使用したサーボをしやすくするため、新しい指令の実行を優先させる
- ユーティリティのような RTC も積極的に用意する

OpenRTM のポーティング作業は次のようなステップで構成される。

- OpenRTM が依存する CORBA のポーティング
- CORBA のユニットテストで動作確認
- 商用実時間 OS 上で実行している CORBA サーバ・クライアントの動作確認
- 商用実時間 OS 上で実行している CORBA サーバ・クライアントと、外部 OS 上で実行中のクライアント・サーバとの動作確認
- OpenRTM のポーティング作業
- OpenRTM のユニットテストの動作確認
- OpenRTM のサンプルの動作確認
- 商用実時間 OS 上で実行している OpenRTM コンポーネント同士の動作確認
- 商用実時間 OS 上で実行している OpenRTM コンポーネントと外部 OS で実行している OpenRTM コンポーネントの動作確認

ポーティング作業時には OpenRTM が開発途中であったため、一部の動作確認は OpenRTM の主要な実行環境である Linux でのみ実施した。
作業が終了した段階で改めて有効な変更かどうかを吟味した後、パッチを提出した。

シミュレーションおよび実機向けカメライメージ取込み RTC を作成・公開

ビジョン関連のユーティリティコンポーネントを作成した。双腕プラットフォームは頭部および両手にカメラが設置されているため、カメラを選択しやすいように Video Stream RTC を作成した。このコンポーネントはオープンソースで公開しており、下記 Google Code サイトからダウンロード可能である。URL は http://code.google.com/p/hrprtc-grx-video-stream である。
使用する時には下図のように接続する。

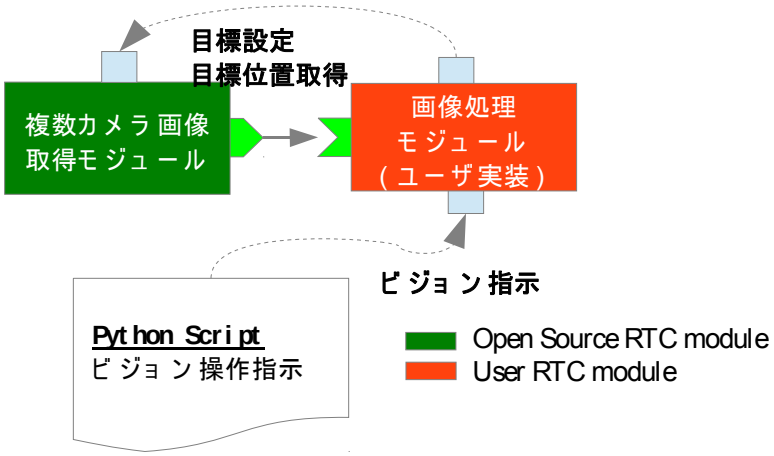


図 138　Video Stream RTC の接続

この RTC のインタフェースは本プロジェクトで作成した共通インタフェースに基づいている IDL ファイル、Img.idl で定義されているものを使用している。サービスポートを通してデータポートに流すデータを選択するようになっている。共通インタフェースには接続するカメラを定義する方法が出ていないため、conf ファイルを使って接続するカメラを定義するようになっている。この RTC を使ってシミュレーションからも画像を取り出すことも可能である。

下図は平成 23 年国際ロボット展で展示したデモのビジョンをシミュレーションで実行しているところで緑色のテープが張られている部分を OpenCV で検知しているところである。左がカメラ画像、中央が OpenCV が検知したテープを張られた場所で、右がテープの場所を赤く示したものである。

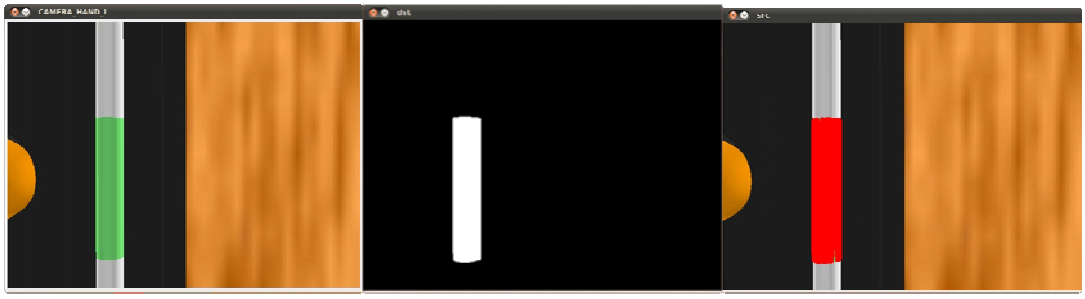


図 139　ビジョン処理のシミュレーション例

さらにユーティリティ RTC として座標変換に使用する RTC を作成した。

多くのビジョンシステムは検知しているデータをカメラ座標で返すようになっている。このままでは物体を持ち上げる手先などで使用することができないため、座標系を手先を制御するためのものに変換する必要がある。双腕プラットフォームでは

稼動する頭部と手先にカメラを設置しているため、変換するためには現在の軸角度が必要となる。
この機能を満たすのが Coordinate Mapper RTC である。OpenHRP の使い方のよい例にもなるため、Video Stream RTC と同様に Google Code でソースを公開している。公開しているサイトの URL は
http://code.google.com/p/hrprtc-grx-coordinate-mapper である。

カメラとロボット本体等の座標変換用 RTC を作成・公開

Coordinate Mapper RTC には 2 つのメソッドが用意されている。
calcLocal2Global に引数でローカル座標での位置・姿勢、そしてローカル座標のついている関節の名前を渡すと、この情報を使い位置・姿勢をグローバル座標に変換して返すようになっている。
calcGlobal2Local に引数にグローバル座標での位置・姿勢と変換先のローカル座標が属しているジョイント名を渡すと、指定したローカル座標に変換された位置と姿勢を返すようになっている。
使用上の注意としては変換する時の関節角度が重要なため、動いている時に使用することはできない。新しい関節角度を受け取るために下図のような接続を構築し、最新の関節角度を RTC に送る必要がある。

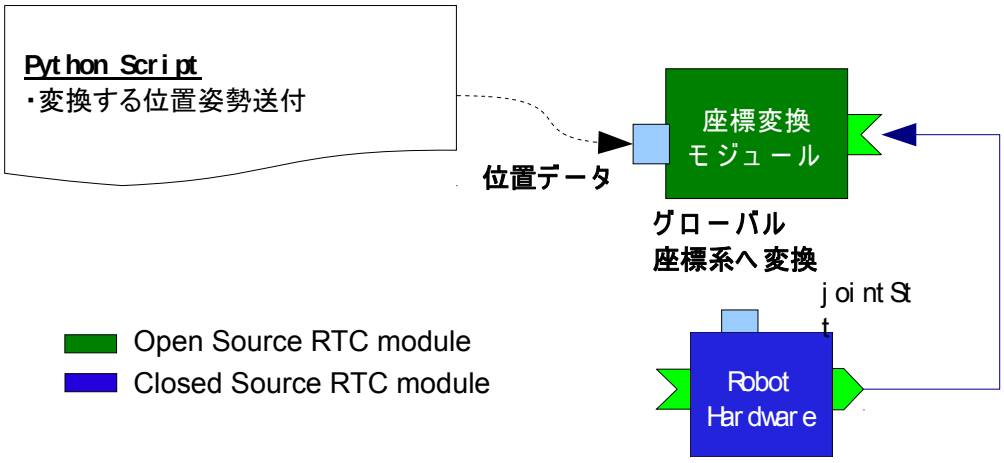


図 140　Coordinate Mapper RTC の接続

商用実時間 OS 向け RTC を Linux Preemptive へポーティング

商用実時間 OS 用の RTC を Linux Preemptive へもポーティングを行った。ポートした Linux のバージョンは Ubuntu Lucid で対応しているバージョン 2.6.31 である。
この段階の Linux Preemptive は普通の Linux よりはリアルタイム性能は高いが、商用の実時間 OS と違いハードウェアの能力の影響を受けジッタが起きやすい。Hackbench を利用してシステムに負荷をかけた時にジッタを計測した結果が下記

のグラフである。

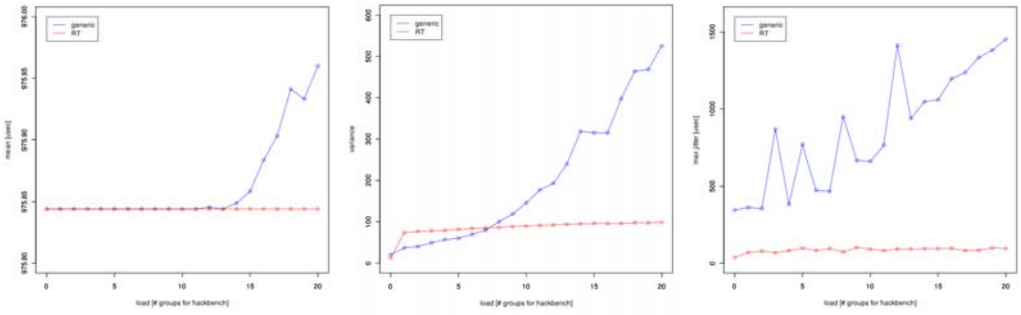


図 141　Hackbench を利用してシステムに負荷をかけた時にジッタの計測結果

左が平均、中央が分散、右が最大のジッタをプロットしたものである。RT カーネルは 10%くらいのジッタに抑えてあるが、使用した CPU が非力(Atom)だったため、ジッタが大きく出ている。Linux の Preemptive カーネルの開発を追っていたが、2.6.31 から 2.6.39 くらいまでいろいろな不具合の影響でなかなか次のバージョンがリリースされず、試すことはできなかった。去年発表されたバージョン 3.0 をベースにしたバージョンは 1 年近く OSDL(Open Source Development Labs)で動作確認しており、これを用いた動作確認の結果を見る限り、リアルタイム性能がさらに上がっている。一般にリリースするためには、少なくとも 4 月以降にリリースされる Ubuntu Linux 12.04 を待つ必要があると考えられる。

ロボット角度補間 RTC に加速度台形制御の追加

双腕ロボットではビジョンを使用したサーボもできるよう、新しく入ってきた命令に追従することを狙いとしている。目標角度を変える時に把持しているワークを落とさないようにするためには、次の目標位置までスムーズに動くことが重要である。全身の動作を補間し再生する RTC をベースに動作補間 RTC を開発し、新たな動作が入力された際に古い動作をキャンセルして、加速度の制限値で一旦停止した後、新しい動作に移行する機能を追加した。この機能の追加により、ビジュアルサーボ制御を実行する事が可能となった。

Python から OpenHRP を使用可能にした統合環境を構築するベースを作成・公開

今までは Java で動作する Python, Jython をスクリプトで使用してきた。このスクリプト言語を使用する時に Python 用の OpenRTM を使用するのか、Java 用の OpenRTM を使用するのかハッキリせず、混乱しているユーザが多かった。さらに Jython では動作しない Python の拡張が多く、Python を使用する利点がかなり減ってしまうという欠点があった。

Java を使用し、Eclipse との親和性が高い Jython にも利点もあった。しかし最近では Java がバンドルされていることは少なく、ユーザが自分でインストールする必要がある。さらに、正式な Java が存在しない ARM を使用することも増えてきている。プラットフォームの自由を考えると、純粋に Python で動くシステムの重要性が上がってきていると言える。

ユーザが自由にメニューを追加することができるという特性をそのままに、システムがどのような状態になっているかについて、メッセージをただ表示するだけではなく、対策を表示することにより把握しやすくした。
ユーザの意見を反映させ、メニュー方式は、1 つの窓にすべての機能を並べる方式と、現在適応できる項目だけ表示する方式の 2 つから選択可能とした。

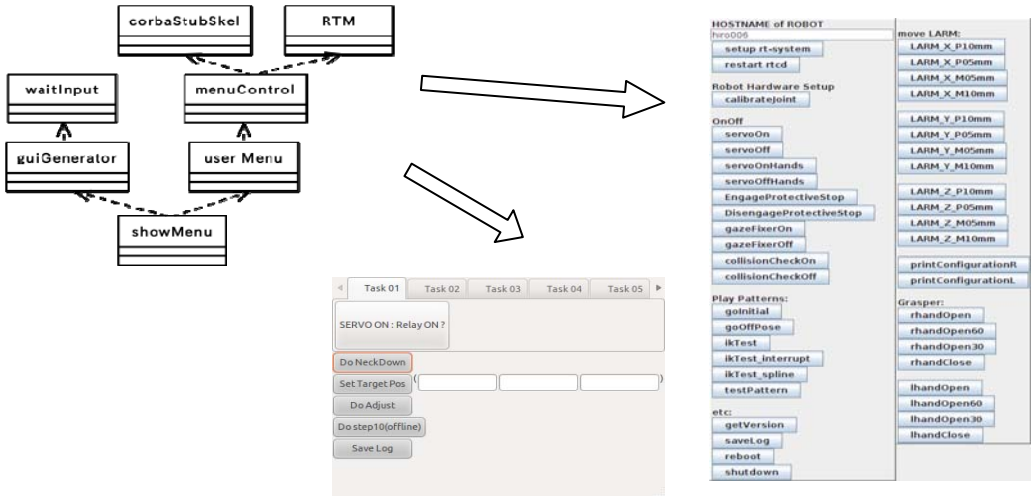


図 142　メニューのカスタマイズ

上図右側の形式が 1 つのメニューにすべての項目を並べる方式で、中央にあるのが現在適応可能なメニューを表示している方式である。この中で基本的にユーザが機能を追加する場合に編集するのは user Menu クラスの中のリストのみであるが、必要であれば menuControl 内の関数を追加することが可能である。どちらの方式を使用するかはメニューを構成するリストの形式でスイッチングするようにしている。
双腕ロボットの場合はロボットを目の前にして実験することが多いため、ジョイントアングルなどの情報はここに表示していない。これらは eclipse の GUI に表示されるため、このインタフェースは軽量にロボットを制御するためという位置づけのものである。

まとめ

以上、実施した双腕ロボットプラットフォームのソフトウェア整備について述べた。本研究項目は、単に RT コンポーネントで動作するロボットプラットフォームの整備のみならず、前述の実時間ソフトウェア設計ツールや後述のロボットシステム構築ツールが活用され、両ツールの機能検証にも用いられた。また、本研究項目は、当初の実施計画にはなかったがプロジェクトの加速化に大きく貢献した。

③ロボットシステム設計支援機能

RT コンポーネントを再利用し、組み合わせてロボットシステムの構築を支援するためのツールとして、ロボットシステム設計ツールと RT リポジトリの開発を行った。

(a) ロボットシステム構築ツール

ロボットシステム構築ツールは、ロボットのハードウェア及び RT リポジトリにより実現されるソフトウェア情報を組み合わせてロボットシステムを構成し、目標とする作業シナリオを実現することが可能かどうか、さらにそのシナリオを実現する動作が実現可能かどうかをシミュレーションにより検証し、ロボットシステムの設計を支援する機能の実現を目指す。本研究項目の最終目標は、モデルに属する各種パラメータを編集するために必要な GUI 機能を追加し、検証を行い、マニュアルやサンプルを充実させ事業化するための準備を行うことである。以下、開発したツールの詳細について述べる。

ロボットシステム構築ツール

OpenHRP 用のシミュレーション環境をベースにロボットシステム構築ツールを開発した。新たに追加した機能として、従来用いてきた VRML に加え、Collada の読み込み機能および書き出し機能も追加している。VRML の難点の一つは、VRML の長所でもあるフォーマットの自由さである。編集している VRML モデルをそのままではシミュレーションで使用できず、CAD データからシミュレーションで使用可能になるまでに数ステップの調整を要していた。また、近年 VRML をサポートしているソフトウェアが減少しつつあるため、ユーザが導入しやすい新たなデータフォーマットが必要となっていた。

Collada は OpenGL などの標準化を行っている Khronos Group が管理しているグラフィクスデータフォーマットで、もともとソニーのプレイステーションのゲームなどに使用されているものである。ソニーはこの技術を社内からスピンアウトし、広く使用することができるように第三の管理機関に委託している。もともとゲームで使用されているということもあり、ロボットのシミュレーションに必要なデータタイプがそろっている上、様々なモデリングツールがこのデータフォーマットをサポートしている。

これら機能を使って VRML を Collada に変換することも可能である。Collada には VRML に含まれない情報が保存されるため、情報を破棄することになる Collada から VRML への変換は行わない。これは Collada を上位フォーマットとして位置づけるためである。
当初は Collada ではなく ModelicaXML を使用するという案が挙がっていたが、Modelica はどちらかというと物理的な特性のシミュレーションを目的とする言語で、現在使用している VRML の代替としては適切ではない。また、現状では Modelica データに対応している 3D モデリングソフトは存在しないため、ユーザに取っての利点もほぼ皆無と言える。Modelica を使用する利点はライブラリの作成にあるが、これは本開発のスコープから大きく外れてしまうため、これは行っていない。
Collada を使用しているロボット関連ソフトウェアが存在するため、双腕ロボットプラットフォームのモデルデータも Collada に変換し、下記サイトにて公開した。

モデル公開サイト：http://code.google.com/openhrp-aist-grx/downloads/list

下図のようにユーザインタフェースを通してアクチュエータやセンサを配置することが可能である。

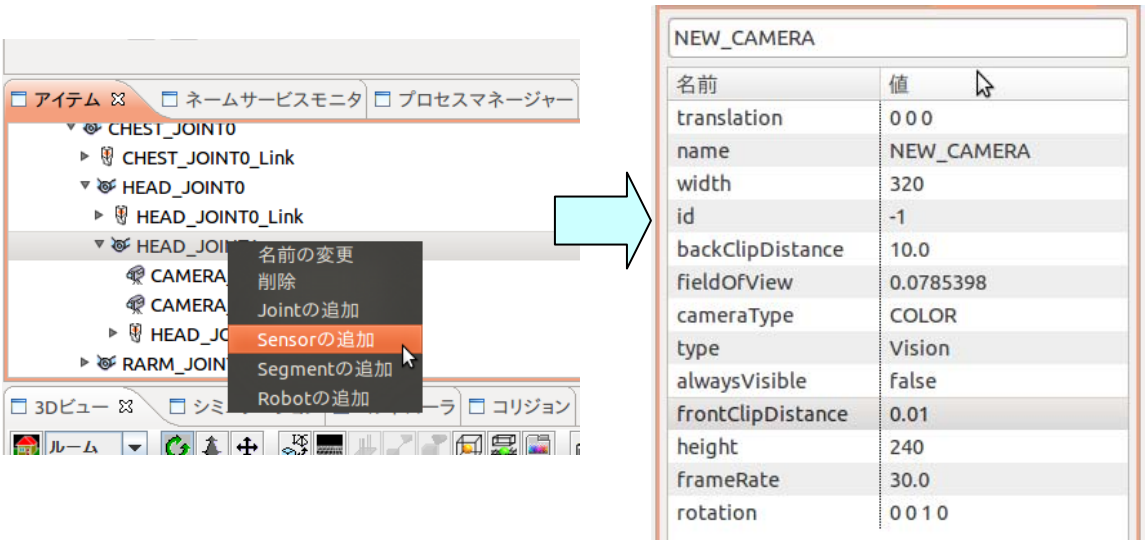


図 143　ロボットシステム構築ツールの操作例

また、追加したセンサの有効範囲を表示す機能も実装した。

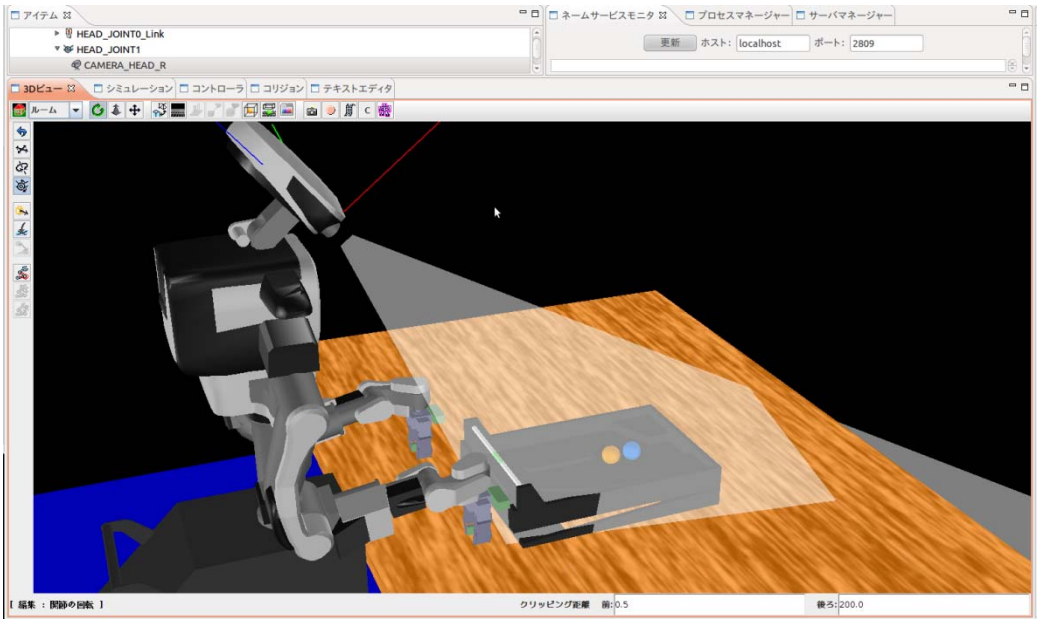


図 144　センサの有効範囲の表示

これらの成果は ROBOMEC2010 で発表している。

加速プロジェクトの双腕ロボットによって、上記機能の有効性を確認した。スケジュールの問題から、同時進行していた双腕ロボットのハンドの詳細モデル決定を待たずに、簡易なモデルによってテストをスタートした。大まかな自由度配置が決まった時点で、本インタフェースを使用し、簡易モデルでテストをスタートした。モデルは非常に簡単なものでスタートし、新しい情報にあわせて随時モデルを更新して確認テストを継続した。下の左図は、最終段階の簡易モデルである。

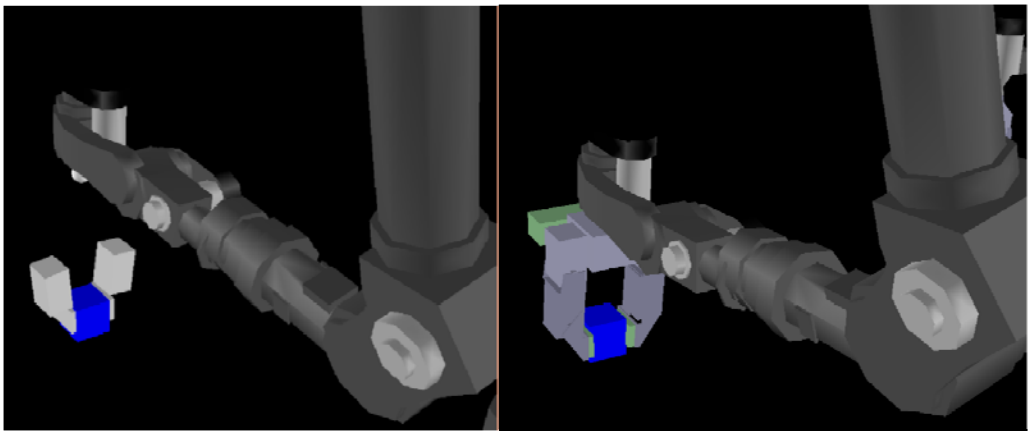


図 145　簡易なモデルによるテスト

ハンドのデザインの詳細が決まった後、右のような CAD モデルに基づいたデータと交換した。この体験は ROBOMEC2011 で発表している。

まとめ

以上、開発したロボットシステム設計ツールについて述べた。本ツールは、双腕ロボットプラットフォームの開発に利用され、その有効性が確認された。また、本ツールは、オープンソースライセンスとして開発し、マニュアルやサンプルのモデルについても公開を行っている。

(b) 分散型データベース

本研究項目は、「RT コンポーネント開発支援機能」において決定した知能モジュール・ハードウェアの仕様記述及び知能モジュールをコンテンツとする分散型データベースを開発することである。最終目標は、ＲＴリポジトリおよび、ＲＴポータルサイトが完成し、本プロジェクトの外部に対して公開するとともに、事業化を行うことである。以下、開発した RT リポジトリについて述べる。

RT リポジトリ

ロボットがロバスト性をもって稼働するためには、ロボットの環境・状況認識能力や自律的な判断能力及び作業の遂行能力の向上が必要であるとともに、ロボットの知能要素をモジュール化し、その蓄積・管理及び組み合わせ等を可能とすることが重要になる。これを実現するためには、ロボット知能ソフトウェアプラットフォーム上で動作するロボットの知能モジュール（RT コンポーネント（RTC））およびその仕様記述（RT プロファイル）を蓄積・管理し、容易に利用するための枠組みが必要となる。この枠組みは、開発者がロボットシステムを開発する際に、有用な RT コンポーネントを選定するために利用されるだけでなく、ロボットシステムが稼働中に、環境や状況に合わせて、必要な RT コンポーネントを動的に選択し、ロボットシステムに組込み、動作できることも望まれる。

そこで、我々は、「RT コンポーネント開発支援機能」において決定した知能モジュールの仕様記述及び知能モジュールをコンテンツとする分散型データベースとして、RT リポジトリを開発した。

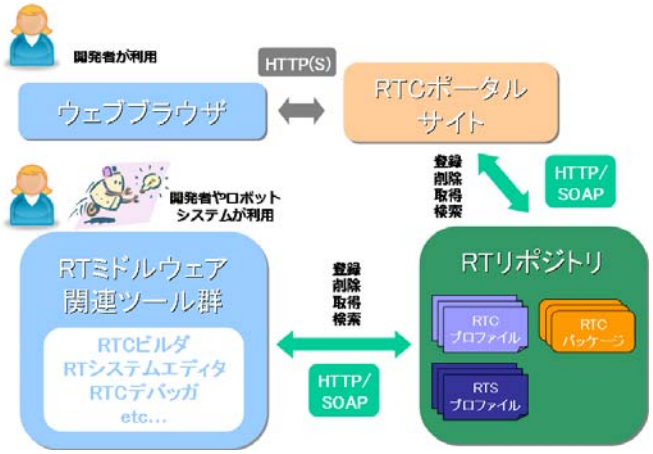


図 146　RT リポジトリ概念図

これまで、RTC は一般公開される機会が少なく、モジュール化の利点が活かされていなかった。ロボットシステムの効率的な開発には、散逸している資源を共有化し、誰でもアクセス可能な状態にすることが不可欠である。RT リポジトリは、これを実現するためのデータベースであり、ロボットシステムを構成するために必要な各種仕様記述（RT プロファイル）や RTC を登録し、一般に公開することができるようになった。これらの操作は、ウェブブラウザや GUI ツールを介して、容易に行うことができる。

RT リポジトリには、2 種類のコンテンツを登録することができる。一つは RTC のパッケージ、一つは RT プロファイルである。RT プロファイルとして、知能モジュール仕様記述である RTC プロファイルと、ロボットシステム仕様記述である RT システムプロファイルを管理できるようにした。RT リポジトリに登録されたコンテンツは、ウェブブラウザや GUI ツールを通じて、検索・ダウンロードすることができる。

一例として、あるセンサを自身のロボットシステムに組み込もうとした場合を挙げる。そのセンサを制御するための RTC をゼロから開発するのではなく、RT リポジトリに登録されている RTC を利用すれば、効率的に開発を進めることができる。また、ソースコードが公開されている RTC を選択すれば、それをベースにより目的に合致した RTC を作成でき、短期間でロボットシステムの品質を高めることができる。RT リポジトリにより、知能モジュールの再利用の仕組みが実現できるのである。

ユーザは、RT リポジトリに対して、ウェブブラウザや RTC ビルダ、RT システムエディタ等のツールからアクセスする。これらのツールが、容易に RT リポジトリにアクセスできるよう、「アクセスライブラリ」を開発し、提供した。また、ウェブサービスでは、インタフェースが WSDL で規定されているため、ユーザ独自のアク

セスライブラリを作成することも可能である。以下の図に示す通り、RT リポジトリではウェブサービス実行環境としてオープンソースの Axis を採用しているが、その他のフレームワークを利用することも可能である。
アクセスライブラリを通じて RT リポジトリに送信されたコンテンツは、RT プロファイルは XML データベースに、RTC はファイルシステムに格納する。RT リポジトリで使用する XML データベースには、検索の高速性を考慮しセック製の Karearea を採用したが、大学や研究機関で無償利用することも考慮し、オープンソースの Xindice も利用可能である。

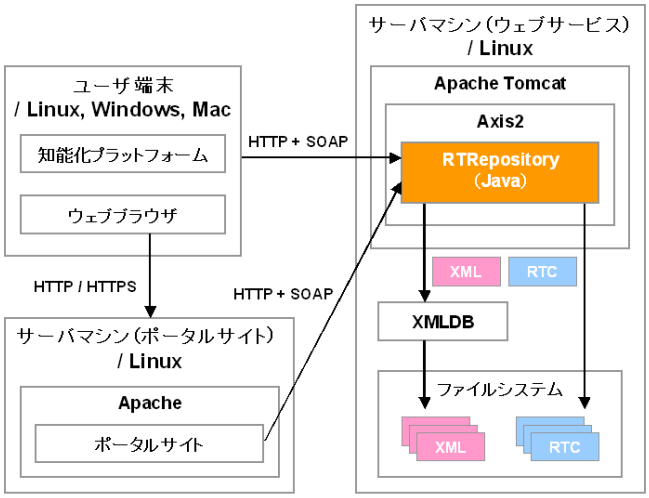


図 147　RT リポジトリのシステム構成

まとめ

RT リポジトリは、産業技術総合研究所および、RTC 再利用技術研究センターに無償提供・評価し、知能モジュールの蓄積、再利用を促進するための再利用 Web 実現に貢献した。

さらに、我々は、RT リポジトリの仕様について、OMG での国際標準として提案を進めている。「Deployment and Dynamic Configuration (DDC) of Robotic Technology Components (DDC4RTC)」は、RTC の動的配置（Deployment）および設定（Configuration）に関する標準仕様である。OMG の RTC 標準仕様では、コンポーネントレベルでの相互運用性を考慮し、基本的なコンポーネントのモデルのみを規定している。一方で、OpenRTM-aist や OpenRTM.NET や韓国 OPRoS プロジェクトの OPRoS コンポーネントなど、複数ベンダより RTC 標準仕様に基づいた、異なるフレームワークが提供されており、RTC の開発や運用時に使用するツール群の共通化および相互運用性の確保が必要となりつつある。DDC4RTC の標準仕様化により、RT リポジトリをはじめ、各ベンダがそれぞれに実装した開発ツー

ルやシステム管理ツールの共通化や相互運用性の向上が期待される。

3.1.2 ロボット知能ソフトウェアプラットフォームの有効性検証

本節では、ロボット知能ソフトウェアプラットフォームの有効性検証について述べる。この実施内容は、前述のロボット知能ソフトウェアプラットフォームの開発の成果物であるツール群および仕様記述方式に関して、実際のロボットシステムを研究開発する上で十分な機能を有するか否かを検証するために実施した知能モジュール群の開発及び検証ロボットシステムの研究開発である。検証用知能モジュール群は、作業知能、移動知能、コミュニケーション知能を含む知能モジュール群であり、知能化プロジェクトで開発される知能モジュール群の先行事例的な知能モジュール群である。また、この検証用知能モジュール群に関しては、ロボット知能ソフトウェアプラットフォームの基本的なツールの実装の完了およびRTC再利用技術研究センターの本格的活動開始にともない平成21年度にその研究開発を終了した。

また、次世代ロボット知能化開発プロジェクトにおいてオープンソースライセンスで開発が行われ知能モジュール群について、その再利用を促進するために、開発者とRTC再利用技術研究センター等の機関と共同して知能モジュールに関するドキュメント作成および知能モジュールの特許侵害調査を実施した。

① 検証用知能モジュール群の開発

検証用知能モジュール群は、リファレンスハードウェアに搭載することで、介助犬が行っているような室内で人の生活活動を支援するロボットを実現することを応用イメージとして、作業知能、移動知能、コミュニケーション知能を含む知能モジュール群である。ただし、ここで開発した知能モジュール群は、生活支援分野のみではなく、他分野の知能ロボットにも利用可能な汎用性を有するように留意したものである。

図 148 に開発した検証用知能モジュール群と検証ロボットシステムの全体構成を示す。

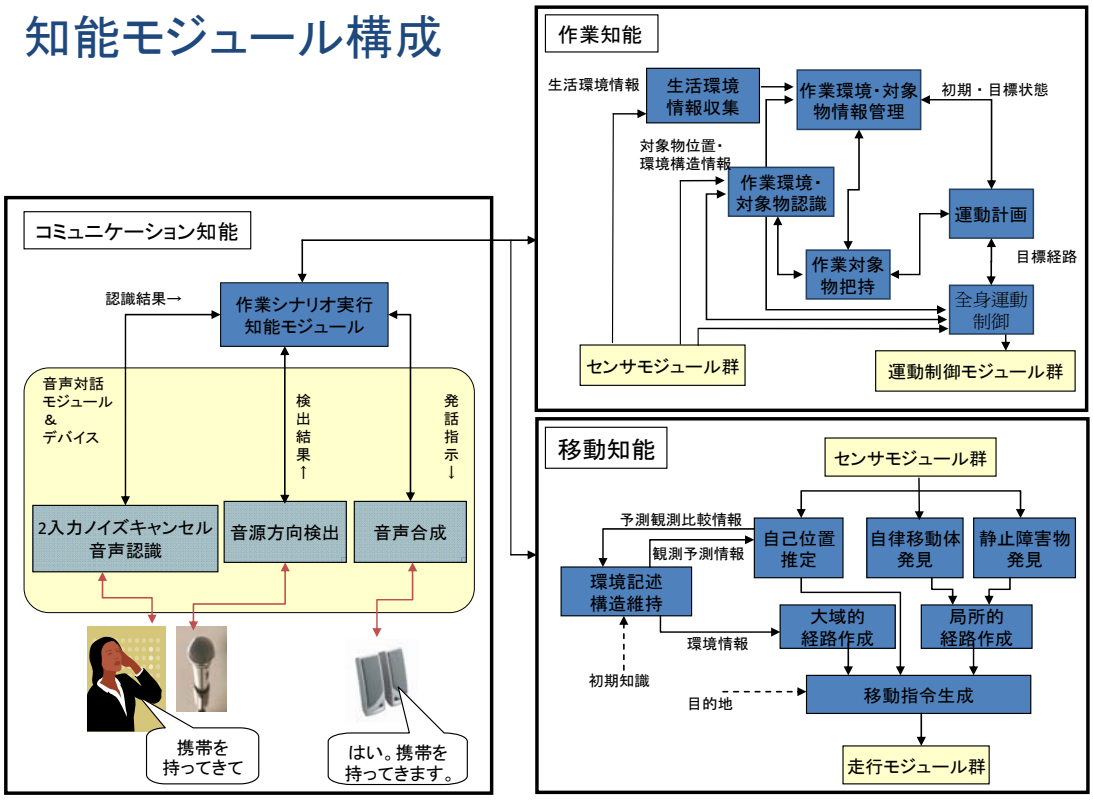


図 148　検証用知能モジュールおよび検証ロボットシステムの構成

また、ここで開発した検証用作業知能モジュール群ならびに検証用移動知能モジュール群は、ロボット知能ソフトウェアプラットフォームの RT コンポーネント開発ツール（RT コンポーネントビルダ、RT システムエディタ）を用いて開発されており、その知能モジュールは知能モジュール仕様記述方式で記述可能であることの検証を行っている。さらに、作業動作生成や移動動作生成に関する知能モジュールの動作検証には、動力学シミュレータを用いることにより、ハードウェアを使用せずとも効率よく開発することの検証も実施した。

さらに、検証用ロボットシステムの動作検証のための移動動作の設計やデモシナリオの作成には、ロボット知能ソフトウェアプラットフォームのツールの 1 つである作業シナリオ設計ツールを使用し、その動作の検証を行った。

なお、検証用知能モジュール群として開発した RT コンポーネント（以降 RTC と略記）は一部を除いてオープンソースライセンスにて開発しており、再利用 WEB に登録し本プロジェクトの各研究項目実施機関に公開した。以下、具体的に開発した知能モジュール群について述べたあと、開発した検証用知能モジュール群を統合して実施した統合検証について述べる。

(a)検証用作業知能モジュール群

(a-1) 作業環境・対象物認識知能モジュール群

この知能モジュール群は、ロボットが障害となるものを回避しながら日常物をマニピュレーションするために必要な情報（種類、位置・姿勢、状態等）を必要な精度で認識する機能及び視覚センサでは捉えることのできない作業対象物との接触や、作業対象物の重さや環境との拘束状態を認識する機能作業対象物を把持できているかどうか判定する機能を実現したものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 幾何学特徴・アピアランス特徴視覚認識モジュール
ステレオ画像から物体の位置姿勢を検出するモジュールであり、③の検証用作業知能モジュール群接続図における ObjectRecognitionRTC として実装した。

(2) 触覚認識モジュール
対象物との接触状態を認識するモジュールであり、1.4 節の検証用作業知能モジュール群接続図における触覚認識 RTC として実装した。

(a-2) 作業環境・対象物情報管理知能モジュール群

この知能モジュール群は、作業中に得られた作業環境・対象物認識モジュール群による観測情報を用いて、ロボットが作業をするために有用な知識（環境・物体モデル）の記述を更新・管理し、さらに、マニピュレーションするために必要な情報（物体の位置、姿勢）が不足している場合には、それを獲得するためのセンシングプランを生成する探索機能を実現する。具体的には、以下の知能モジュールの開発を実施した。

(1) 作業環境知識記述モジュール
(2) 物体知識記述モジュール
(3) 認識結果に基づく環境物体知識更新モジュール
(4) 作業環境・対象物表示モジュール

上記のモジュールは③の検証用作業知能モジュール群接続図における InfoServerRTC として実装した。

(a-3) 運動計画知能モジュール群

運動計画知能モジュール群により最終的に実現する機能は、認識可能な固定・移動障害物の存在する環境で、初期・目標位置間の移動体の衝突のない軌道の探索を行うことである。設定された時間内に解の有無を報告し、軌道が計画されたら出力することとする。

さらに、作業環境・対象物情報管理知能モジュール群などの外部モジュールにより検出された環境変化により軌道の実行が不可能となった場合、再探索を行う機能も実装する。
開発する知能モジュール群は以下の通りである。

(1) 基本軌道計画モジュール
与えられた環境で、初期位置から目標位置への移動体の衝突のない円滑な軌道を計画する。このモジュールは、動作中に環境の変化が検出された場合にも、必要に応じて軌道の再計画を行うことができる。
このモジュールは③の検証用作業知能モジュール群接続図における MultiDevPlannerRTC として実装した。

(2) ロボット運動指令生成モジュール
基本軌道計画モジュールの出力である経路を、ロボットの関節角、車輪の回転角などロボットシステムに適応した制御指令に変換する
このモジュールは③の検証用作業知能モジュール群接続図における PathProviderRTC として実装した。

(a-4) 作業対象物把持知能モジュール群
この知能モジュール群は、把持対象物・作業環境・ロボットの情報（種類、形状、位置・姿勢、状態など）を入力として、適切な把持形態の選定や、対象物へのアプローチから持ち上げまでの一連の把持動作を計画する機能を実現したものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 物体カテゴリーモデルデータベース
実際の把持対象物と把持の観点から抽象化した物体幾何形状（物体カテゴリーモデル）とを対応付けを行うためのデータベースであり、6 種類以上の形状の異なる日常物に関する情報を格納している。

(2) 可能把持形態抽出モジュール
与えられた物体カテゴリーモデルや作業環境、ロボットの情報をもとに可能な把持形態を抽出する機能を実現する。
このモジュールは③の検証用作業知能モジュール群接続図における可能把持形態抽出 RTC として実装した。

(3)　把持動作計画モジュール

与えられた状況に応じて適切な把持点やアプローチ方向、持ち上げ方向を計画する。

このモジュールは③の検証用作業知能モジュール群接続図における把持動作計画 RTC として実装した。

(4)　把持確認モジュール

触覚認識 RTC と連動して、把持の成功・失敗の判定を行う。このモジュールは③の検証用作業知能モジュール群接続図における把持確認 RTC として実装した。

最終的には、リファレンスハードウェアに実装し、障害物のある作業環境において異なる日常物の把持を実現できることを確認した。

可能把持形態抽出モジュール、把持動作計画モジュールについては、複数の形状の異なる日常物についての把持を実験検証した。また、触覚認識 RTC と連動して、把持の成功・失敗の判定を行う機能を実装した。

(a-5) 全身運動制御知能モジュール群

全身運動制御知能モジュール群は、運動計画知能モジュール群で計画された運動指令に基づき、ロボットの全身運動を制御するとともに、作業環境・対象物認識知能モジュール群によって検出された作業環境・対象物との接触に対しても、運動計画知能モジュール群で計画されたように適切な応答を取ることのできる機能を実現するものである。具体的には、以下の知能モジュールの開発を実施した。

(1) アーム単体でのグリッパ位置・姿勢制御モジュール
(2) アーム・台車自己干渉回避モジュール
(3) 移動台車とアームを協調させたグリッパ位置・姿勢制御モジュール

なお、全身運動制御知能モジュール群のアーム・台車自己干渉回避モジュール、移動台車とアームを協調させたグリッパ位置・姿勢制御モジュールについては、改良版基本軌道計画モジュールおよびロボット運動指令生成モジュール（③の検証用作業知能モジュール群接続図における MultiDevPlannerRTC と PathProviderRTC）の機能として組み込む形で実現した。さらに、リファレンスハードウェア試作 1 機、2 号機のシミュレーションモデルと実機で動作する全身運動制御モジュールを③の検証用作業知能モジュール群接続図における RH1ControllerRTC として実装した。

(a-6) 生活環境情報収集知能モジュール群

生活環境情報収集知能モジュール群の項目では、生活環境内で作業を実行するロボットが、生活環境の様々な情報を収集し、生活環境における異常状態の可能性を検出したり、温度・明るさ等の変化を検知したりすることができる知能モジュール群を開発する。また、室内の人の大まかな場所や動きの検出が可能な知能モジュール群を開発する。これらにより、温度や明るさの変化（寒くなった、暗くなった等）に応じて、窓を閉めたり照明を調節したりすることを促すメッセージを発するなどの機能や、人の生活パターンに異常がある可能性を検出する機能など、「見守り」の機能を実現できる。

具体的には、温度、湿度、気圧、におい（ガス）、照度、焦電等の、様々な生活環境情報を知覚できるセンサを用いた生活環境情報収集知能モジュール群の開発と、これらのセンサ利用知能モジュールを連携させることにより生活環境を総合的に認識する知能モジュール群の開発を行った。生活環境情報収集知能モジュール群で用いるセンサに関しては、安価なセンサと汎用なマイコン等を用いて知能モジュールソフトウェアをRTコンポーネントとして実装可能なPC等への接続が容易な形で公開した。

焦電センサ・温度センサ・湿度センサ・においセンサ・照度センサを搭載したユニット（長期間電池駆動可能）と、焦電センサアレイを搭載した生活環境情報収集モジュールを開発した。これにより人のいる方向を認識して、人の方向が変化すると、そちらを向き、照度センサ、温度センサ、湿度センサ、においセンサの状態が変化すると、それに応じて発話行動を行う知能モジュールを実現した。

また、熱源方向認識・24時間ログの各知能モジュール改良と、変化パターン記憶・異常状態検出の各知能モジュールβ版の開発を行い、試作したセンサ旋回用可動機構及び駆動モジュールにより検証した。

(b) 検証用移動知能モジュール群

検証用移動知能モジュール群は、リファレンスハードウェアのように車輪機構を有するロボットが屋内を安全に目的地まで移動するための機能を実現する知能モジュール群である。これらの知能モジュール群は、動力学シミュレータおよびリファレンスハードウェアに搭載し、動作検証が可能になるように開発を実施した。

また、以下の知能モジュール群は、OpenINVENTとして、事業期間中にオープンソースライセンスの知能モジュール群として公開を実施した。

(b-1) 自己位置推定知能モジュール群

この知能モジュール群は、自己位置と 3D 点&直線群により構成される地図を状態ベクトルとして管理し、運動モデルによる予測、単眼及びステレオ画像入力による能動観測、さらにはオドメトリに基づき、拡張カルマンフィルタあるいはパーティクルフィルタで状態ベクトルを更新し、環境中の 3D 点&直線を選択的に地図に追加・削除することにより、照明条件や設置物が変化する環境においても実時間で頑健に自己位置を推定しつつ地図を構築できる機能を実現したものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 自己位置・3D 点群地図からなる状態管理モジュール
(2) 拡張カルマンフィルタによる運動モデルに基づく状態予測更新モジュール
(3) 予測状態に基づく 3D 点の単眼画像観測モジュール
(4) 拡張カルマンフィルタによる単眼 3D 点観測誤差に基づく状態更新モジュール
(5) 画像特徴量に基づく安定 3D 点特徴選択モジュール
(6) 単眼画像による 3D 点特徴初期化モジュール
(7) 拡張カルマンフィルタによるオドメトリに基づく状態予測更新モジュール
(8) 単眼画像による拡張カルマンフィルタに基づく自己位置推定地図構築モジュール
(9) オドメトリ計算モジュール
(10) 自己位置推定融合モジュール

なお、自己位置・3D 点群地図からなる状態管理モジュール、拡張カルマンフィルタによる運動モデルに基づく状態予測更新モジュール、予測状態に基づく 3D 点の単眼画像観測モジュール、拡張カルマンフィルタによる単眼 3D 点観測誤差に基づく状態更新モジュール、画像特徴量に基づく安定 3D 点特徴選択モジュール、単眼画像による 3D 点特徴初期化モジュールは、モジュール化の粒度の検討を詳しく行った結果、単一の RT コンポーネントとして実装することが妥当であると結論し、単一の RT コンポーネントとして視覚による自己位置同定モジュールとして実装した（本 RTC に限り公開はしていない）。また、拡張カルマンフィルタによるオドメトリに基づく状態予測更新モジュールを実現するにあたり、実施計画にはないオドメトリ計算モジュール（③の検証用移動知能モジュール群接続図における OdometryRTC として実装）、自己位置推定融合モジュール（③の検証用移動知能モジュール群接続図における LocalizationRTC として実装）を開発した。

オドメトリ計算モジュール、視覚による自己位置同定モジュール、自己位置推定融合モジュールをリファレンスハードウェアに搭載し30平米の環境で実験検証した。

(b-2) 環境記述構築維持知能モジュール群

この知能モジュール群は、環境DBから得た間取り図などから初期化した静的3D環境記述を、静止障害物などの情報に基づき更新し、他の移動知能モジュールに現状情報を提供する機能を実現するものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 環境記述初期化モジュール、
このモジュールは③の検証用移動知能モジュール群接続図における MapBuilderRTC として実装した。

(2) 静的３D環境記述更新モジュール
このモジュールは③の検証用移動知能モジュール群接続図における MapMaintenanceRTC として実装した。

これらの知能モジュール群をリファレンスハードウェアに搭載し、30 平米の環境で実験検証を実施した。これによって、実験環境において、間取り図あるいは平面図を初期知識として、移動に必要な障害物情報を自律的に獲得し、次に述べる静止障害物発見知能モジュール群が検出した環境条件の変化を維持できることが実現できた。

(b-3) 静止障害物発見知能モジュール群

この知能モジュール群は、センサから得られた距離情報に基づき移動経路上の静止障害物の検出と形状計測する機能を実現するものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 距離情報利用仮想バンパーモジュール
このモジュールは③の検証用移動知能モジュール群接続図における BumpDetectionRTC として実装した。

(2) 距離センサパンチルト制御モジュール
このモジュールは③節の検証用移動知能モジュール群接続図における PanTiltControlRTC として実装した。

(3)　　距離情報利用移動経路上障害物候補検出

このモジュールは③の検証用移動知能モジュール群接続図における ObstacleDetectionRTC として実装した。

これらの知能モジュール群をリファレンスハードウェアに搭載し、30 平米の環境で実験検証を行い、床面からの高さ 10cm 以上で不透明で鏡面反射なく適度な模様を持つ障害物が 1 個／平米以内の場合を、実験環境において起こり得る 10 ケース以上抽出し、全ケースにおいて速度 0.5m/秒以上で走行中に障害物を認識しその位置を計測できることを確認することができた。

(b-4) 自律移動体発見計測知能モジュール群

この知能モジュール群は、画像情報に基づき移動経路上の移動体の検出と計測を行う機能を実現したものである。この知能モジュール群は、実施計画ではカメラからの画像処理により自律移動体の発見と計測を実装する予定であったが、前述の、静止障害物発見知能モジュール群の距離情報利用仮想バンパーモジュールを利用し、観測範囲内の 3 次元グリッドの障害物存在確率を計算することで移動体検出が行えることがリファレンスハードウェアに搭載した実装実験で明らかになった。これにより、自律移動体発見計測知能モジュール群は、静止障害物発見知能モジュール群と統合し、実装を行った。

また、実証実験を通じて、床面からの高さ 10cm 以上で不透明で鏡面反射なく適度な模様を持つ障害物が進路内に 1 個で速度 0.5m/秒以下で加速度ほぼない場合を、実験環境において起こり得る 10 ケース以上抽出し、全ケースにおいて速度 0.5m/秒以上で走行中に障害物を認識しその位置と速度を計測できることが確認できた。

(b-5) 大域的経路作成知能モジュール群

この知能モジュール群は、現在地から目的地までの大域的格子状経路を生成・更新する機能を実現するものである。具体的には、下記の知能モジュールの開発を実施した。

(1)　　拡張版静的 2D 環境記述に基づく大域的格子状経路生成モジュール

このモジュールは③節の検証用移動知能モジュール群接続図における GlobalPathPlanningRTC として実装した。

なお、拡張版静的 2D 環境記述に基づく大域的格子状経路生成モジュールは、静的 3D 環境記述に基づく大域的格子状経路生成モジュールを、処理の高速性の観点

から静的 2D 環境記述に基づく大域的格子状経路生成モジュールとして実装したものを拡張し実装したものである。この大域的格子状経路生成モジュールでは、リファレンスハードウェアを用いた 30 平米の環境で実験検証を行い、実験環境において任意の現在地と目的地の組み合わせに対して経路を生成できることが確認できた。

(b-6)局所的経路作成知能モジュール群

この知能モジュール群は、大域的格子状経路をベースにして、障害物情報などに基づいて局所経路を生成・更新する機能を実現するものである。具体的には、以下の知能モジュールの開発を実施した。

(1) 静的 2D 環境記述に基づく局所的格子状経路生成モジュール
このモジュールは③の検証用移動知能モジュール群接続図における LocalPathPlanningRTC として実装した。

なお、静的 2D 環境記述に基づく局所的格子状経路生成モジュールは、静的 3D 環境記述に基づく局所的格子状経路生成モジュールを、処理の高速性の観点から静的 2D 環境記述に基づく局所的格子状経路生成モジュールとして改良し、実装したものである。

この知能モジュールをリファレンスハードウェアに統合し、30 平米の環境で実験検証したところ、実験環境において障害物発見に関して設定したすべての出現ケースに関して、障害物の位置情報を受け取って、1 秒以内に目的地への経路を局所的に動作再計画できることが確認できた。

(b-7)移動指令生成知能モジュール群

この知能モジュール群は、現地から目的地への格子状及び滑らか経路を維持し、経路に沿って安全かつ効率良く走行するための移動速度指令を生成する機能を実現するものである。具体的には、以下の知能モジュールの実装を行った。

(1) 経路形状と障害物情報に基づく移動速度指令生成モジュール
(2) 経路形状と障害物情報に基づく移動共通速度指令生成モジュール
(3) ジョイスティック操作に基づく移動共通速度指令生成モジュール
(4) 移動共通速度指令生成モジュール
(5) リファレンスハードウェア 0 号機台車部用移動指令変換モジュール
(6) リファレンスハードウェア 1 号機台車部用移動指令変換モジュール

これらのモジュール群は③の検証用移動知能モジュール群接続図における DriveControlRTC に相当する。

なお、経路形状と障害物情報に基づく移動共通速度指令生成モジュールとジョイスティック操作に基づく移動共通速度指令生成モジュールは実施計画にはなかったが、平成 20 年に実施した先行共同デモのために必要であったため開発した。移動共通速度指令生成モジュールは、平成 21 年の国際ロボット展でのデモのために目標点列が逐次与えられる場合に拡張した。
また、(4)、(5) のリファレンスハードウェアに関する知能モジュールは、ロボット知能ソフトウェアプラットフォームの実機による検証のために開発したものである。
これらの知能モジュール群をリファレンスハードウェアに実装し、30 平米の環境で実験検証を行ない、実験環境において現在地と目的地と障害物出現の想定しうる 10 種以上の組み合わせに対して、平均速度 0.5m/秒以上で安全にスムーズに移動できることが確認できた。

(c) 検証用コミュニケーション知能モジュール群

この知能モジュール群は、実環境において人間とロボットの自然なコミュニケーションを実現するために必要な対話者の方向検出や音声認識機能を実現するためものである。さまざまな実環境において安定した音声認識を実現するために、複数のマイクロホンを使って周辺雑音を取り除いたり、対話者の方向を検出したりするための機能を実現するものである。検証用コミュニケーション知能モジュールは、ロボットシステムでは、ユーザからの命令を効率よく入力するためのインタフェースを実現するための知能モジュールである。
以下に述べる知能モジュールは、平成 20 年度には、共通機能であるワーカフレームワーク（Linux 版、OpenRTM-aist-0.4.2 準拠、ライブラリ）を実現して、これを組み込むことで実現を行った。ワーカフレームワークは、「作業シナリオ仕様記述方式」に基づいて実装され、これを RT コンポーネントに組みこんで利用することにより、作業シナリオ実行系との規約に従ったメッセージングを実現している。本機能は、作業シナリオ実行ツールの実現にも利用された他、平成 21 年 1 月に実施した先行デモにおけるリファレンスハードウェアに組みこまれた作業や移動のための RT コンポーネントの実現でも利用され実証された。
ワーカフレームワークは産総研の RT ミドルウェア（OpenRTM-aist-1.0）の機能を拡張し、RT ミドルウェアに NEC のロボット技術のノウハウを組み込んだものである。ワーカフレームワークを導入することにより、複数のロボットアプリケーション（シナリオ）を随時切替ながら動作するようなロボットシステムを容易

に構築できる。特に音声認識や音声合成のコンポーネントと組み合わせることにより、ユーザとのインタラクションを行う対話型のシナリオを実現するのに適する。

ワーカフレームワークの概念図を以下に示す。ワーカフレームワークは、ロボットの行動決定を制御するためのシナリオスクリプトの実行環境である、シナリオプレーヤコンポーネントを中核とし、メカ、音声認識、音声合成などのロボット各部の機能を制御するワーカコンポーネントが RT ミドルウェアの枠組みで結合した構成を取る。

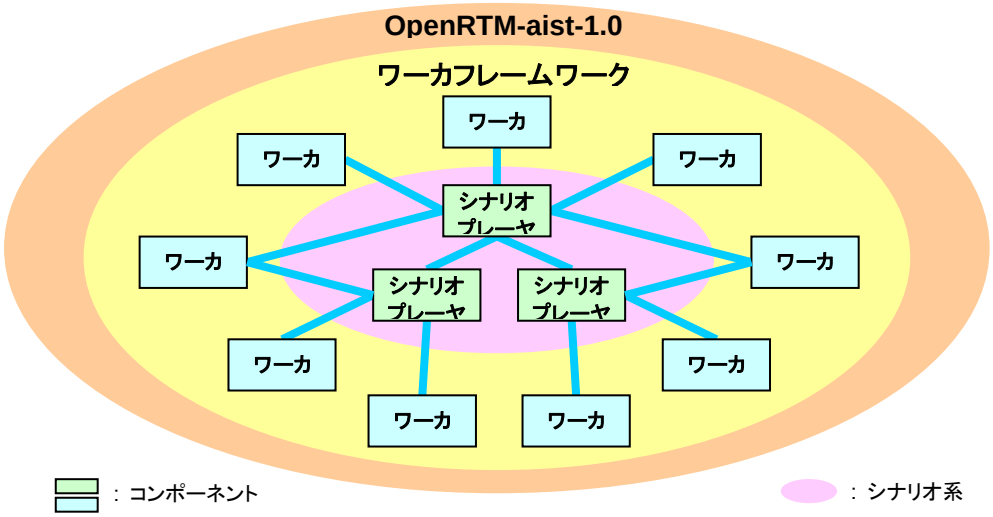


図 149　ワーカフレームの概念図

人間に例えると、ワーカが手足を直接司る小脳に相当するのに対し、シナリオプレーヤは手足を使った行動を司る大脳に対応する。

ワーカフレームワークのモジュール構成図を以下に示す。RT ミドルウェアは CORBA を利用した分散ミドルウェア環境であるため、その上に構築されたワーカフレームワークも分散ミドルウェアの性質を持ち、複数のコンピュータにまたがって稼動する。

モジュール構成図の最上位層にはシナリオスクリプトが位置し、これがロボット全体の行動決定を司る。シナリオスクリプトはシナリオプレーヤによって解釈・実行され、必要に応じてシナリオプレーヤから各ワーカにロボット各部の動作要求が発行される。

ワーカフレームワークは、これらの連係を効率良く行うためのメッセージマネージャや XML 変数空間のしくみを提供する。

以上のワーカ、シナリオプレーヤ、メッセージマネージャなどは全て RT ミドルウェアのコンポーネントとして実装され、RT ミドルウェアのポートを介して相互に

通信・連携し合う。

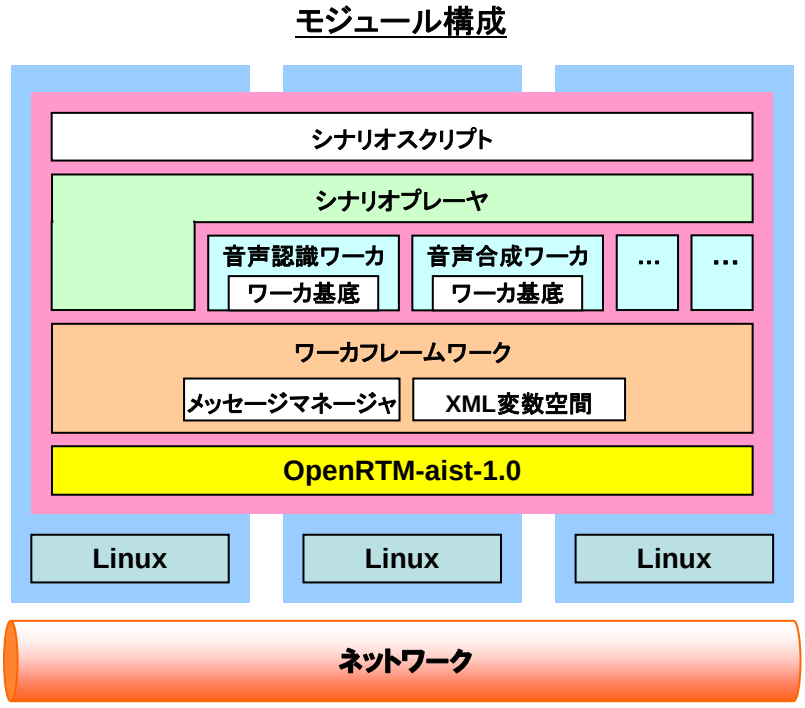


図 150　ワーカフレームワークのモジュール構成図

次に、ワーカフレームワークのクラス階層図を以下に示す。
RT ミドルウェアのコンポーネントの機能を活用するため、コンポーネントの基底クラス（DataFlowComponentBase）を継承するワーカ基底クラス（RtcWorkerComponentBase）を用意されている。ワーカ作成者は、このワーカ基底クラスを継承したワーカ派生クラス（RtcXXWorker）を作成する。
ワーカ基底クラスは入力用／出力用のデータポートをそれぞれ１つずつ持ち、外部との入出力は全てこの２つのポートを介して行う。即ち、他のコンポーネントへのメッセージ送信には出力用のデータポートを、他のコンポーネントからのメッセージ受信には入力用のデータポートを用いる。ただしメッセージマネージャは出力用のポートを複数所有する。
ワーカフレームワークでは、ポートにバッファリングされたデータに対して優先度管理などの独自の処理を行うため、上記データポートは RT ミドルウェアの標準のデータポートから派生させた、専用のデータポートクラスとする（MsgInPort／MsgOutPort）。
またワーカフレームワークのマネージャにはメッセージマネージャ（RtcMsgManager）とコンフィグマネージャ（RtcConfigManager）の２つがあ

り、前者はメッセージの送受信の管理を、後者はワーカのインタフェース定義の管理を行う。

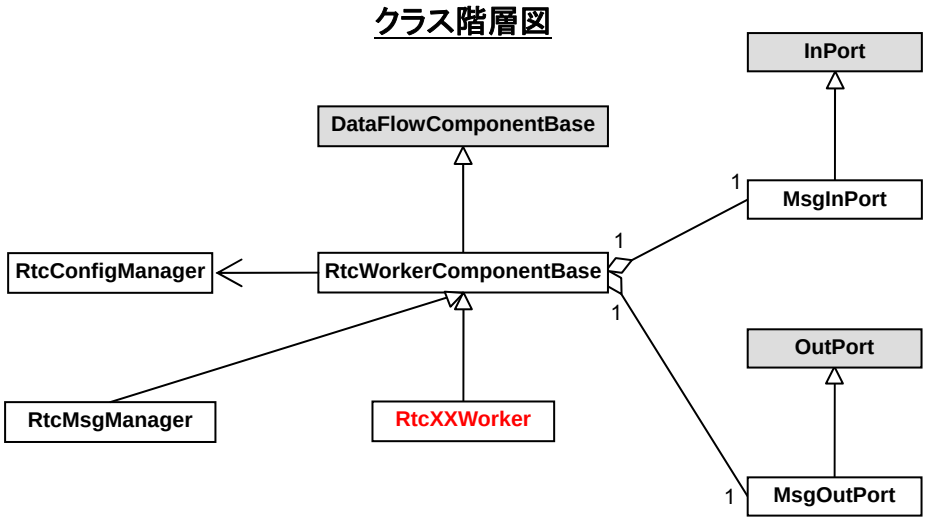


図 151　ワーカフレームワークのクラス階層図

ワーカフレームワークを構成する各コンポーネント（ワーカ、シナリオプレーヤ、メッセージマネージャ）は専用の起動プロセスの管理下にあり（*.a/*.so）、フレームワーク全体の起動はこの起動プロセスが担う。
起動プロセスは、まず配下の各コンポーネントモジュールを初期化したのち、コンポーネントのインスタンスを生成する（図 152 ①）。次に、コンポーネントのリンク情報を読み込み、その指示に従ってコンポーネントのポート同士をリンクする（図 152 ②）。リンク情報の詳細は、OpenRTM-aist-1.0 の仕様に準ずる。
その後、コンポーネントを順次 Active 化していく（図 152 ③）。この時にはメッセージマネージャ→ワーカ→シナリオプレーヤの順に Active 化を行う。ワーカフレームワークが複数のコンピュータにまたがる場合は、初期化・生成は各々のコンピュータの起動プロセスが行うが、リンク・Active化はメインの起動プロセスが行う。

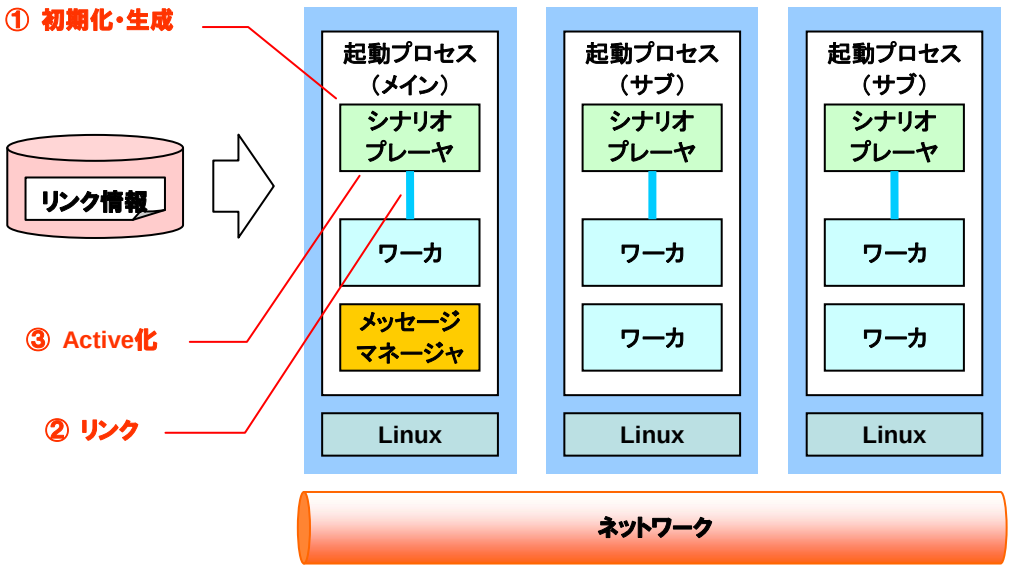


図 152　ワーカフレームワークの起動

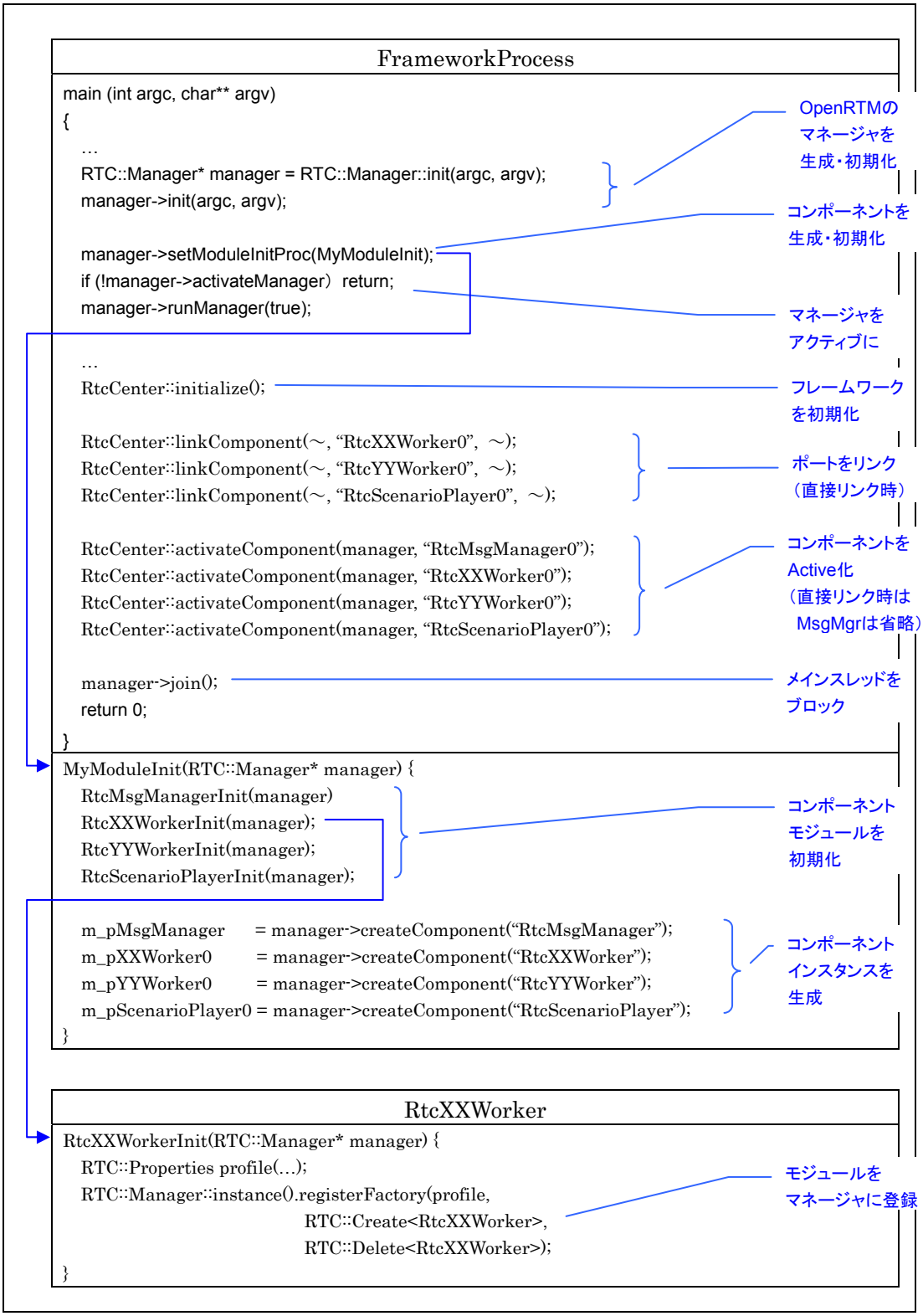


図 153　ワーカフレームワークの起動プロセスの例

ワーカフレームワークにおけるメッセージングは、基本的に全てデータポートを介して行う（サービスポートは各ワーカの事情により使っても良いが、ワーカフレームワークはサービスポートについては関知しない）。
フレームワークを構成する各ワーカは、データ入力ポートとデータ出力ポートをそれぞれ１つずつ所有し、コマンド／レスポンス／イベントといったメッセージは全てこの２つのポートを介して授受される。
通常の場合、ワーカから他のワーカにメッセージを送る場合は、メッセージマネージャが仲介する。ただしシステムの構成が非常にシンプルな場合や、高速なメッセージ応答性を要求される場合は、ワーカ同士を直接リンクすることも許す。
メッセージには宛先の概念があり、自分宛でないメッセージを受信したワーカは、単にそれを破棄する。例えばコマンドは通常、特定のワーカを宛先にして送られ、他のワーカがそのコマンドを受信した場合には破棄される。同様に、コマンドに対するレスポンスもコマンドの要求元を宛先として送られ、それ以外のワーカがレスポンスを受信しても破棄される。例外はイベントで、通常は全てのワーカに対して同報配信される。
以上のメッセージは全て XML 文字列の形で送受信される。

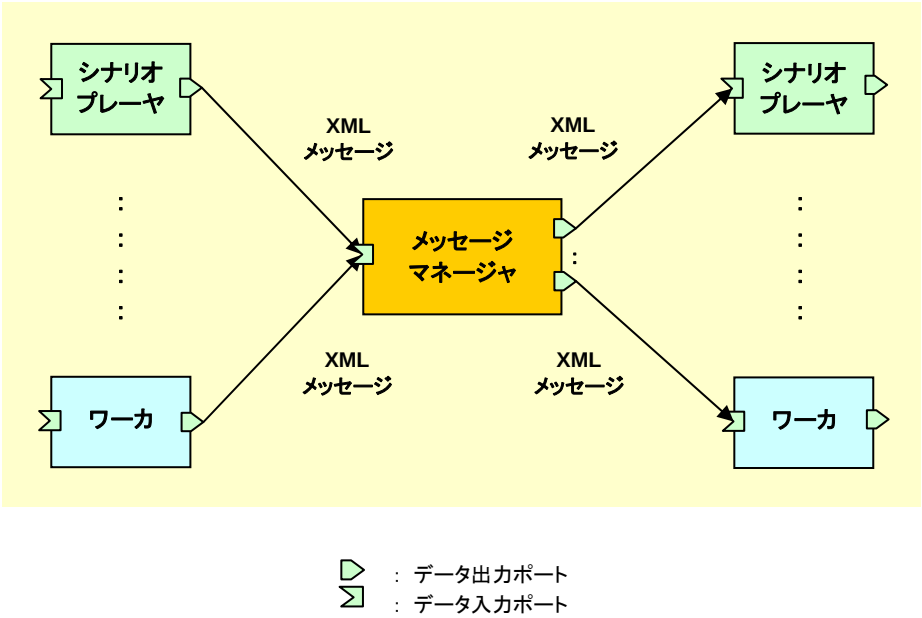


図 154　メッセージングの全体像

(c-1) 2 入力ノイズキャンセル音声認識モジュール

この知能モジュールは、正面マイクと後方マイクの２つのマイク入力を用いて後方雑音抑制をしつつ、日本語単語音声認識する機能を実現するものである。特定実験環境における耐雑音下での認識精度 70%以上で認識することを目標とした。本モジュールは共通基盤プロジェクトで開発された音声対話モジュール上で専用に動作するモジュールとして実装する。共通基盤プロジェクトでの実装に含まれる RoboStudio フレームワークを再構成し、OpenRTM-aist-1.0 版への対応を行った。音声対話モジュールの形でリファレンスハードウェアに組み込み、他のモジュール群と連携動作させることを目標とした。

このモジュールが提供する単語音声認識機能は、「離散単語音声認識」という音声認識手法に基づいた音声認識エンジンを使用し、予め決められた単語セットのみを音声認識する機能である。音声認識させる単語（以降、認識語と呼ぶ）セットを記述したものを音声認識辞書と呼ぶ。アプリケーション開発者は、開発する音声認識アプリケーションに応じた音声認識辞書を作成する必要がある。また、音声認識は常時行うものではなく、認識を行うタイミングは AP 側で適宜制御する必要がある。

音声認識の一般的な流れのイメージ図を以下に示す。音声認識機能は、音声認識機能を司る制御プログラムを指す。本音声認識機能は、シングルセッションで行う仕様であり、複数のモジュールから同時に音声認識処理を呼び出すことはできない。

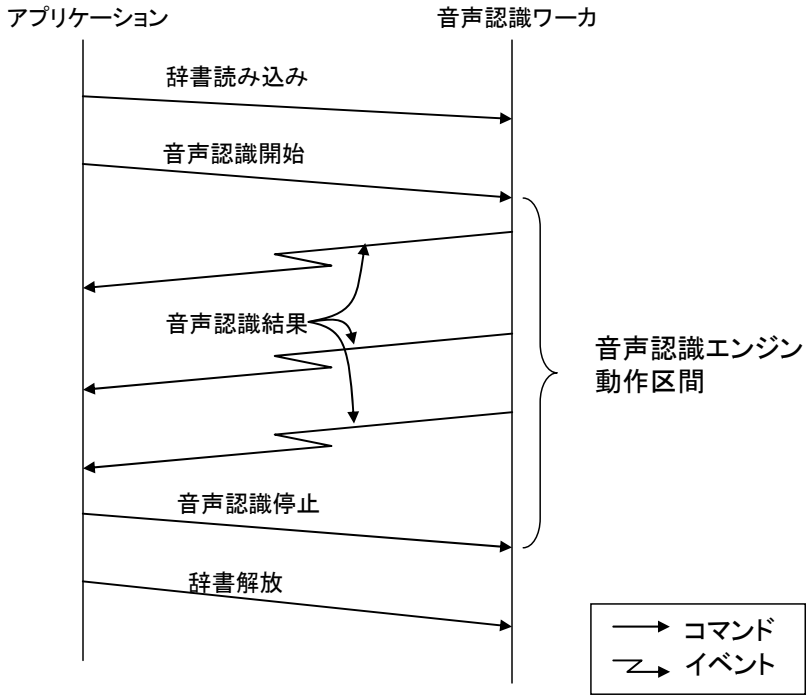


図 155　音声認識の一般的な処理の流れ

音声認識機能のコマンドの概要を以下に示す。

表 23　音声認識機能のコマンド

API 名	説明
Start	指定した認識ルールを用いて、音声認識処理を開始する。音声認識開始コマンドに相当。
Stop	音声認識を停止する。音声認識停止コマンドに相当。
LoadGrammar	音声認識辞書を音声認識エンジンにロードする。辞書読み込みコマンドに相当。
ReleaseGrammar	読み込み済音声認識辞書をアンロードする。辞書解放コマンドに相当。
GetStatus	音声認識状態を取得する。
GetUser	使用中の話者データを取得する
SetUser	話者データを変更する。 共通の話者データを設定済であり、変更不可とする。

以下に、音声認識機能のイベントの詳細を示す。音声認識イベントは、音声認識エンジンが処理を行った結果、何らかの認識語であるという認識結果が出た場合のみ要求元のモジュール側へ返送される。音声認識処理においてリジェクトされた場合にはイベントの返送は行わない。音声認識結果は、下記の要素を持った XML 構造のテキストデータとして RTM 経由で返送される。

表 24　音声認識機能のイベント

SpeechRecognized			
要素名	型	要素数	説明
status	int	1	エラーステータス。
pron	var	3	認識結果の第1～第3 候補の読み。
grapheme	var	3	認識結果の第1～第3 候補の表記。
group	var	3	認識結果の第1～第3 候補の意味。
score	int	1	認識結果のスコア。

また、平成 21 年度には、前述のワーカフレームワーク（OpenRTM-aist-0.4.2 ベース）を、音声対話モジュール H/W(ARM9)に移植し、これをベースに、音声認識ＲＴコンポーネントを移植した。また、ＲＴコンポーネントの性能評価を完了し、特定実験環境における耐雑音下での認識精度 70%以上で認識することを確認

した。

(c-2) 音源方向検出モジュール

この知能モジュールは、ロボットの周囲の水平同心円上に３つ、ないし４つの周囲マイクを配置し、それらからの入力を用いて、音源方向（360 度）の検出機能を実現するものである。静環境下での方向推定精度 80%以上で検出することを目標とした。本モジュールは共通基盤プロジェクトで開発された音声対話モジュール上で専用に動作するモジュールとして実装した。共通基盤プロジェクトでの実装に含まれる RoboStudio フレームワークを再構成し、OpenRTM-aist-1.0 版への対応を行った。音声対話モジュールの形でリファレンスハードウェアに組み込み、他のモジュール群と連携動作させることを目標とした。

音源検出機能のコマンドの概要を以下に示す。

表 25　音源検出機能のコマンド

コマンド	説明
start	音源方向検出処理を開始し、イベント送信を行う。
stop	音源方向検出処理を停止する。

音源方向検出機能のイベントの詳細を以下に示す。

表 26　音源方向検出機能のイベント

DoaDetected		
付加情報名	型	説明
Type	int	1:確定イベント 0:不定イベント
Direction	int	検出した音源の方向（初期値:－186） レンジ：-180～180　　不定値：-186
Volume	int	検出した音源の音量の平均値（初期値:－1） レンジ：0～32768
機能詳細	音がしたと判定したことを通知する。イベントの種別、方向、音量の情報を持つ。	

平成 21 年度には、平成 20 年度に開発を行った「2 入力ノイズキャンセル音声認識モジュール」、「音源方向検出モジュール」、「音声合成モジュール」の共通機能

であるワーカフレームワークを、音声対話モジュール H/W(ARM9)に移植し、これをベースに、音源方向検出 RT コンポーネントを移植した。また、RT コンポーネントの性能評価を完了し、水平同心円上に３つの周囲マイクを配置し、それらからの入力を用いて、音源方向（360 度）の検出において、静環境下での方向推定精度 80%以上で検出可能であることを確認した。

(c-3) 音声合成モジュール

この知能モジュールは、日本語発話テキストをコマンドポートから入力し、言語解析、音素合成を行って、サウンド出力より合成発話を実現するものである。本モジュールは共通基盤プロジェクトで開発された音声対話モジュール上で専用に動作するモジュールとして実装した。共通基盤プロジェクトでの実装に含まれる RoboStudio フレームワークを再構成し、OpenRTM-aist-1.0 版への対応を行った。音声対話モジュールの形でリファレンスハードウェアに組み込むことを目標とした。

以下の表に音声合成モジュールが備えるコマンドとイベントの一覧とその概要を示す。

表 27　音声合成モジュールが備えるコマンド

コマンド	説明
StartSpeak	引数文字列の発話/再生を行う
StopSpeak	発話/再生を停止する
SetDefaultSpeaker	デフォルトの話者の設定を行う
GetSpeakStatus	発話/再生状態の問い合わせを行う
SetSpeakVolume	デフォルトのボリュームを設定する
GetSpeakVolume	デフォルトボリュームを取得する
SetDefaultEffect	再生時の効果を指定する
SetDefaultEffectChange	再生時効果の大きさを指定する
SetDefaultPitch	デフォルトのピッチを設定する
SetDefaultSpeckSpeed	デフォルトの再生速度を設定する
SetDefaultIntonation	デフォルトの抑揚の設定を行う

表 28　音声合成モジュールにおけるイベント

イベント	説明
SpeckReceived	発話再生が開始されたことを通知する。
SpeckFailed	発話再生開始時にエラーを検出したことを通知する。
SpeakStatus	発話再生の終了を通知する。正常時・異常時ともこのイベントを使用する。

この知能モジュールは、平成 20 年度には、上述のように「2 入力ノイズキャンセル音声認識モジュール」、「音源方向検出モジュール」とともに共通機能であるワーカフレームワーク（Linux 版、OpenRTM-aist-0.4.2 準拠、ライブラリ）で実現を行った。

② リファレンスハードウェアの開発

本研究項目は、本プロジェクトで開発される各種知能モジュール群を適宜装着することで、これらの機能を検証することが可能なプラットフォームロボット（リファレンスハードウェア）を開発することである。最終目標として、量産機の開発を完了して一般向けにリファレンスハードウェアを販売できる体制を整え、事業化を図ることである。以下、本研究開発の詳細を述べる。

開発の背景および目的

本プロジェクトでは、次世代ロボットシステムを効率よく開発するためのロボット知能ソフトウェアプラットフォームの研究開発を行い、また、一般家庭の屋内で活動する介助犬の作業を想定し、この機能を実現するための各種知能化モジュールを開発し、これらを研究用ロボットに実装することで、ソフトウェアプラットフォームの有効性の検証および改良を行っていく。本研究ではその一環として、ソフトウェアプラットフォームで開発された知能化モジュールを実機で検証するためのロボット「リファレンスハードウェア」を開発した。

研究用ロボットとして要求されるのは、ユーザ側で自由にソフトウェアが構築できるライブラリ、API が公開されていることである。しかしながらこれらの条件を満たした市販の研究用ロボットは海外製のものが多く、日本国内で購入すると数百万円と高価であり、容易に導入するのが難しい。

そこで、既存のプラットフォームロボットに較べて低コストで導入可能で、かつ知能化モジュールを随時追加しながらこれらの動作を検証することが可能な「リファレンスハードウェア」を開発することとなった。

リファレンスハードウェアの基本仕様の策定

開発に先立ち、屋内で活動する介助犬が行う作業について、同じコンソーシアムに参画している産業技術総合研究所検証用知能モジュール群開発チームに検討いただいたところ、以下の 2 点が挙がった。

- 500[ml]ペットボトル飲料を床から 700[mm]上方の机の上まで持ち上げる能力および可動範囲
- 平らな床面で 800[mm]の安全通路をその場旋回できる移動能力

それに伴い、関係機関と共同でリファレンスハードウェアの仕様を検討した結果、ロボットの構成は直列リンク型多関節アーム 1 台と独立 2 輪移動機構による電動台車 1 台とし、アームのリンク長、および関節軸、動輪構成を図 1 と定めた。また、マニピュレータ先端での可搬重量はエンドエフェクタと 500[ml]ペットボトルとハンドに装着するセンサ類を合わせた重量を想定して 2[kg]、一方台車の可搬重量についてはマニピュレータユニットと制御用 PC を搭載したことを想定して 20[kg]とし、それに伴い各軸の可動範囲、目標定格トルクおよび目標最大回転数を表 1 のように定めた。

試作 1 号機の開発

図 2 に試作 1 号機の外観を示す。マニピュレータユニットおよび電動台車ユニットはモジュール化を図っているため、台車からマニピュレータユニットを取り外してそれぞれ単独での動作も可能であり、移動機構またはハンドリング機構が不要な研究用途でも個別に対応できるようにした。なお、各ユニットの電源電圧は DC24[V]である。以下に各ユニットの詳細について述べる。

マニピュレータユニット

まず、関節各軸には他社製に比べると小型でありながら高出力で、様々なロボットで採用実績のある Maxon 製コアレス DC モータを動力源として採用した。この場合、モータの出力は労働安全衛生規則での産業用ロボットの適用除外となるよう、全軸 80[W]未満に抑えた。また、全軸ハーモニックギアを用いて高減速比化および関節機構の小型化を図り（図 3)、図 4 のように 2 軸、3 軸には非常停止時にマニピュレータが自重で動かないよう無励磁ブレーキ機構を実装した。各部の寸法はなるべく人間の腕に近い印象を持たせるようスリム化を図ったが、マニピュレータ自体小型化させると電機系統の内部での配線処理が困難となるため、図 5 に示す超小型モータドライバ mrsvm100 を各軸ごとに配置し、RS485 を介してデイジーチェーン接続を行った。mrsvm100 はリファレンスハードウェア専用として(有)アールラボによって開発されたモータドライバで、外形寸法 40[mm]× 40[mm]、コアレス DC モータについて電圧 DC24[V]最大 10[A]のドラ

イブ能力を有している。また、RS485 を介したコマンドによってモータの速度制御、位置制御が可能である。
これによって省配線化が実現しケーブル全てをマニピュレータ内部で引き回すことが可能となり、図 6 のように最大幅 102[mm]（2 軸周辺）以内でスリムな印象を持たせることに成功した。アーム本体の材質にはアルミニウム合金を採用し、またカバーには ABS 樹脂を採用して、本体重量を 8.2[kg]まで軽量化させた。カバー作成に際しては 3 次元プリンタを利用することで、少ロットでも複雑なデザインにも対応して製作することができた。
なお、マニピュレータの各関節の仕様は表 2、可動範囲は図 7 の通りである。

台車ユニット

図 8 に台車ユニットの外形図を示す。800[mm]の通路を旋回できることをふまえ、幅 500[mm]、全長 543[mm]とした。また、上記マニピュレーションユニットの要求諸元として挙げた「床から 700[mm]上方の机までペットボトルを持ち上げる」という事項を考慮して、アームの取付ベース高さを床面から 250[mm]、最大高さを 290[mm]とした。本体重量はバッテリー含めて 45.5[kg]である。車輪構成は旋回半径をできるだけ小さくとれるようにし、かつある程度の段差乗り越えにも対応できるよう、前輪 2 駆動輪、後方 2 キャスタとした。また、10[mm]までの段差乗り越えや平地からスロープへ移動する際に起こりがちな空転への対策として、図 9 に示すラバースプリング（製品名ロスタ）を適用したサスペンション機構を考案し、実装した。
動力源にはマニピュレータユニット同様Maxon 製コアレスDC モータを採用し、また非常停止時に確実に停止できるよう、モータ後端軸側に無励磁ブレーキ機構を追加した。
モータドライバはマニピュレーションユニットと同様 mrsvm100 を採用し、RS485 を介して左右輪個別に制御が可能である。台車ユニット後方の蓋を開けると B5 ノートまたはネットブック PC を収納できるブラケットがあり（図 10）、オプションでハーフサイズ 4 スロット PCI バスラックにも交換可能とした。電源として 12[V]16[Ah]の密閉型鉛蓄電池を採用し、底面に横置きして低重心化を図った。
外装は当初マニピュレータユニット同様 3 次元プリンタを利用して ABS 樹脂で製作したが、強度不足であるのとマニピュレータユニットのカバーに比べて大型化して製作コストおよび製作時間がかかったため、焼き付け塗装を施した鋼板に変更した(図 11)。
台車前方にあるマニピュレータの取り付け台座では、マニピュレータを前方中心の他、図 12 のように右または左にオフセットして取り付けられるようにした。

その他、非常停止対策として、前方および後方にタッチスイッチが内蔵されたラバーバンパーを装着するとともに、台車後方から有線で非常停止スイッチボックスを引き出し、実験で異常動作が見られた際に実験補助者 1 名がスイッチを押せる状態にした。

試作エンドエフェクタ

エンドエフェクタはユーザごとで仕様が多種多様となることから、マニピュレータユニット先端のフランジに装着、取り外しできる構造とし、ユーザで使用希望するエンドエフェクタがある場合はそのフランジに合わせた取り付け部品を用意していただくこととした。今回は多種多様なエンドエフェクタの一例として、500[ml]ペットボトル飲料を把持することを想定した、開閉 1 自由度型のエンドエフェクタを試作した。図 13 に外観および外形図を示す。まず、グリッパの形状は固定側が 1 本、可動側が 2 本指とし、3 本指で対象物を把持できる。これらのグリッパは取り外し可能な構造で、ユーザ各自が考案・設計した他の形状のものへ交換することもできる。また、ペットボトルからちり紙といった多様な把持対象を想定して開閉幅は 0 ～ 75[mm]、把持力は 10[N]とした。モータドライバにマニピュレータユニットと同様 mrsvm100 を採用することで、エンドエフェクタ自体 RS-485 を介して 1 つのモジュールとして単独動作できるようにした。固定側グリッパの根元には 3 軸ベクトルセンサ（ミネベア製 MX020）を内蔵させることで、把持力の計測ばかりでなく固定側グリッパ先端を触覚のように用いて把持対象の位置や形状を探ることも可能である。さらに、図 14 のように上面には小型 USB カメラを取り付けることができ、ハンドアイビジョンも実現可能である。本体の重量は 0.7[kg]である。

試作 1 号機のシステム構成

図 15 にリファレンスハードウェア試作 1 号機のシステム構成を示す。マニピュレータユニットと台車ユニットはそれぞれ RS485 を介して通信を行っているが、2 つのユニット同時に制御するには 2 ポート確保する必要がある。また、RS485 インタフェースを標準で実装する PC が存在しないため、現実には USB-RS485 変換ケーブルまたはインタフェースが必要となる。検証用知能モジュールは Linux で開発が進められる一方で、USB-RS485 変換器が Windows でしか動作しないものがほとんどだという問題も存在した。その結果、Linux でもデバイスドライバが用意されおり、入手性がよいなどの理由から moxa 製インタフェース（Uport1130 または Uport1450。図 16 参照）が以後本プロジェクトで標準インタフェースとして利用されることとなった。

試作 1 号機用の試用状況について

試作 1 号機は 2008 年度に 4 セット製作し、2009 年 3 月に産業技術総合研究所検証チームへ 2 セット供給した。供給の際にはモータドライバ開発会社である（有）アールラボより、モータドライバ制御ライブラリのサンプルとして提供いただいたリファレンスハードウェア制御ソフトウェアを同梱し、産総研検証チームではこれを参考にしてリファレンスハードウェア用 RTC を開発していただいた。なお、供給したソフトウェアは Linux（カーネル 2.6、Ubuntu7 または 8.04LTS で動作確認）で動作し、以下の 3 種類である。

- calib_arm：マニピュレータ原点位置出し動作
- joydrive:マニピュレータユニットをゲーム用ジョイパッドで制御できる（マニピュレータの動作確認に利用）
- joybase：台車ユニットをゲーム用ジョイパッドで制御できる（台車ユニットの動作確認に利用）

残りは同じコンソーシアムの東京農工大へ 2009 年 8 月に、コンソーシアム外ではあるが RTC 再利用技術研究センターへ 2009 年 6 月に引き渡した。
なお、1 セットを 2009 国際食品工業展（FOOMA2009）前川製作所ブースにて展示デモを行った。展示では床に置いた 500[ml]のペットボトルを右側面のテーブルに置くというシーケンス動作を行った。図 16 に当時の状況を示す。

試作 1.5 号機の開発

産総研で試用いただいている間にいただいた要望を 2 号機の設計へフィードバックするためと、奈良先端大学院大学での利用が平成 21 年 4 月に決まり、1 セット追加で必要になったことから、試作 2 号機先行検証用という位置付けて試作 1.5 号機を 1 セット製作した。マニピュレータユニット、台車ユニットとも 1 号機と外形およびシステム構成は変わらないが、アーム関節軸のメカリミット部の強化と（位置決めピンからブロックへ変更）、4 軸（旋回）のトルク強化が図られた。平成 21 年 7 月に完成し、その後奈良先端大へ貸し出した。なお、リファレンスハードウェア専用に開発されたモータドライバ mrsvm100 の開発元である（有）アールラボが平成 21 年 3 月末で廃業したため、試作 1.5 号機以降のモータドライバの改良および販売窓口はゼネラルロボティックス（株）へ移管された。

試作 2 号機の開発

試作 1 号機、1.5 号機を各機関で試用いただいている間に以下の問題点が上がった。

- マニピュレータユニットの 5 軸(J5)の強度が弱く、動作中に障害物と干渉し

た際に 5 軸が破損することがある。

- バッテリーを充電するには、台車ユニット本体から一度バッテリーを取り出さなくてはならず、作業に手間がかかる。
- バッテリーの放電状況が確認できないため、最適な充電時期が把握できず、場合によっては実験中電圧不足で突然停止することがあった。

また、試作 1 号機ではデザイン先行で設計を行ったため、部品の一部に複雑な形状のものが存在し、部品製作費のコストがかかっていた。そこで、平成 21 年度はこれらの問題への改善を図った試作 2 号機の開発を行った。

マニピュレータユニット

図 17 に試作 2 号機の外形図を示す。試作 1 号機の 5 軸、6 軸については当初の要求仕様にあげた定格トルクは満たしていたものの、実験中操作ミス、プログラムのバグ等で暴走して障害物と干渉した際、その衝撃に耐えられず破損することが見られたため、図 18 に示すように 5 軸、6 軸を強化し、関節軸トルクも表 3 のように変更した。これによって、試作 2 号機を試用いただいている機関で当該箇所が故障したというトラブルはその後見られなかった。なお、その他の外形、関節軸の仕様等には変更はない。

台車ユニット

試作 1 号機からの大きな変更点は、電源系統の改良である。まず、バッテリーを本体から取り出さなくても充電ができるよう、図 17(a)のように台車ユニット左側面に充電コネクタを新設した。充電コネクタへの電気系統はメカニカルリレーを利用してメインスイッチが切れるとともにバッテリーとコネクタが接続され、メインスイッチが入ると充電系統が遮断される構成とした。また、充電コネクタの上方にアナログ電圧計を設置し、バッテリーの電圧状況を目視できるようにした。一方、右側面には図 17(b)のように外部からの電源供給コネクタおよび電源系統切り替えスイッチを新設した。これにより、リファレンスハードウェア用ソフトウェアのデバッグを行う際、外部から電源を確保すればバッテリー切れを気にすることなく長時間稼働させることが可能となった。その他、試作 1 号機、1.5 号機では測域センサを前面底面付近に搭載することを想定して前面下方に開口部が設けられていたが、実際にはマニピュレータユニットを側方へ寄せて正面台座に測域センサを設置するといった使用方法がとられることが多く、測域センサを台車ユニット前面下部には設置しないとの意見が多数であったため、前面の開口部を廃止した。なお、外形寸法、移動機構などその他のハードウェアについては試作 1 号機から大きな変更はない。

エンドエフェクタ

試作 2 号機では試作 1 号機開発時に試作したベクトルセンサ内蔵版の他に、ベクトルセンサを省略版を新たに開発し、試作 2 号機配布先の要望に応じてどちらか一方を装着した。

ベクトルセンサ省略版ではベクトルセンサ設置スペースが空いた分、可動範囲が 75[mm]から 90[mm]へと拡大した。

試作 2 号機改の開発

平成 21 年度後半の予算追加により、3 セット追加製作することが平成 21 年 9 月に決まったため、基本性能はそのままで、台車ユニットの組立やすさの改善を図った試作 2 号機改を開発した。

大きな変更点は台車ユニットの電気機器の配置で、試作 2 号機では上述のように電気系統が改良されたものの、その一方で 1 号機に比べて電気機器が増加し、台車内部の空いた箇所に随時機器を配置したため、電気配線の引き回しが複雑になった。そこで 2 号機改では作業を容易にするため、図 18 のように台車天板カバーをあけるとすぐ電機機器にアクセスできるよう配置させた。

試作 2 号機の試用状況について

試作 2 号機は 6 セット製作し、まず平成 21 年 10 月に完成した後、産業技術総合研究所へ先行で 1 セット搬入し、平成 21 年 11 月に開催された国際ロボット展 2009 にて展示およびデモ対応で利用された。図 19 に当時のデモの様子を示す。その後大阪大学、東京理科大学、東京農工大学、RTC 再利用技術研究センターへ搬入した。また、平成 22 年 7 月には、試作 3 号機を試用する予定になっていた東芝にも 1 セット、試作 3 号機が完成後交換という条件で一時貸し出した。なお、東京理科大で使用していたセットのうち、マニピュレータユニットのみ東京理科大と同じコンソーシアムに参画している筑波大学へ平成 22 年 6 月に移籍し、また東京農工大で使用していたセットについては、平成 23 年 1 月に RTC 再利用技術研究センターへ移籍した。特に RTC 再利用技術研究センターでは、知能モジュール統合検証実験において、知能化プロジェクト参画機関から収集した知能モジュールを統合検証するための検証用ロボットとして利用いただいている。

一方、試作 2 号機改は産業技術総合研究所、奈良先端科学技術大学院大学へ 1 セットずつ引き渡し、残り 1 台は前川製作所で他機関での故障時のバックアップ用として保管している。

試作 3 号機の開発

平成 22 年度では試作 2 号機改までの改良機という位置づけで試作 3 号機を開発

し、知能モジュールの有効性検証の実験を行う機関（RTC 再利用技術開発センターなど）へ配布することとなった。この場合、各機関で再利用性の実証を迅速に行えるようにするため、ソフトウェア開発パッケージ、下位の制御システムの仕様を統一し、さらにセンサなど各機関共通で必要なオプションも併せて追加して、リファレンスハードウェア標準モデルとして提供することとなった。なお、ソフトウェア関連についてはモータドライバを開発、提供いただいているゼネラルロボティックス(株)が担当することとなった。

産総研､RTC 再利用技術開発センターなど関係機関で試作 3 号機に装着する共通オプ

ションを検討した結果、以下の通りとなった。

- 測域センサ：北陽電機製 Top-URG（UTM-30LX）
- ハンドアイステレオカメラ：IEEE1394b 単眼カメラ× 2 台や IEEE1394b ステレオカメラ× 1 台等、複眼かつ高速で同期撮影が可能な製品
- スピーカ：USB 接続のステレオスピーカ
- その他：ハンド取り付け部に 6 軸力覚センサを取り付けられるよう対応

しかしながら現状の仕様でこれらオプションを装備すると、マニピュレータの可搬重量を超えるため、マニピュレータユニットのトルク仕様を再策定した。

一方、下位制御システムの仕様統一に関連して、台車ユニットに Mini-ITX 仕様の PC 基板を内蔵させ、リファレンスハードウェア専用制御 RTC（ゼネラルロボティックス製）を稼動させることとなったため、台車内部の構造を見直すこととなった。

マニピュレータユニット

図 20 にマニピュレータユニットの外形図を示す｡まず標準オプションを搭載することを前提に、エンドエフェクタの質量を 1.5 倍、6 軸先端に力覚センサを取り付けたことによるエンドエフェクタの先端までの延長分を+50[mm]という条件で各関節軸のトルクを見直した結果、表 4 の通りになった。

特に屈曲軸(J2, J3, J5)のトルク増大が約 1.5 ～ 2 倍必要となるため、その対応案として以下の 2 案が出された。

(a)使用するモータの出力を上げる

(b)使用するモータはそのままで、ギヤ比を上げる

上記(a)案ではギヤ比の変更を必要としないため 2 号機までの関節速度は維持でき、また 2 号機までの外見的な設計変更はあまり必要としない。一方、(b)案ではギヤ比を上げた分だけ速度が低下し、またハーモニックギヤや前段のプーリの大型化などに伴う大幅な設計変更が必要であり、それに伴いデザイン変更、外形大型化

という懸念がなされた。しかしながら(a)案では 2 軸で 80[W]以上のモータを必要とすることになり、研究用途では適用外ではあるが、他の用途に将来利用した場合労働安全衛生規則に抵触する可能性がある。さらに、2 号機までの関節速度では速すぎるという意見も挙がった。同じコンソーシアムである産業技術総合研究所と対応案を検討した結果、大幅な改良となるものの(b)案で開発を進めることにした。表 5 にマニピュレータユニットの各関節の仕様を示す。

トルクの増大化やリンク長、可動範囲の見直しを行った結果試作 1 号機、2 号機と比較すると外形が一回り大きくなり、最も外形が増えたのが 2 軸関節周辺の幅で 80[mm]から 105[mm]へ、全長も 755[mm]から 785[mm]へ増加した。それにもかかわらず図 21 の通り基本的なデザインを維持できた。また本体重量も増えて 10.2[kg]となった。その他、マニピュレータユニットへの通電状況が目視で把握できるよう、電源コネクタ近辺に通電を知らせるための LED を追加した。

台車ユニット

図 22 に外形図を示すこれまでの試作 1 号機、2 号機から大きく変更となった点は、Mini-ITX 仕様の PC 基板を搭載して、台車ユニット自体が下位制御システムとしての機能を持たせた点である。搭載する PC の仕様については RTC 再利用技術研究センターで試作 2 号機に搭載して使用していた機器と同一とすることにし、表 6 に示す仕様の製品を採用することとした。

また、2 号機改に続いて電機機器配置の見直しを行い、これまで台車ユニット用の電機機器が配置されていたスペースの上方に Mini-ITX 基板を搭載した。図 23 に PC 基板搭載に伴う変更箇所を示す。ストレージは走行時の衝撃等を考慮してソリッドステートディスクを採用することとしたが、バックアップ等交換作業を容易にするため、後方に最大 2 ドライブまで収納可能なストレージラック（但しホットスワップには未対応）を搭載した。PC 用の電源は DC24[V]から ATX 規格の電源を供給できる DC-DC 変換基板 M4-ATX（mini-box 製，250[W]）を採用した。本基板には電源監視用の USB 端子が設けられているため、PC に接続して電源のマネジメントが可能である。その他、台車ユニット背面に PC 起動スイッチおよび通電通知用 LED を設置し、PC 起動スイッチには実験時の誤操作を防止するため、スイッチカバーを設けた。図 24 に試作 3 号機台車ユニットの外形図を示す。

これら PC 関連機器を搭載した結果、台車ユニットの外形寸法のうちこれまでの試作機と比較して全高が 20[mm]増えた。逆に本体重量は製造コスト低減の観点から部品点数を減らした結果、2 号機までの台車ユニットよりも若干軽くなり、44.9[kg]となった。
オプションの装着については、前方正面に測域センサを取り付け、その右側に USB 接続のステレオスピーカを配置した。この場合、図 24 のようにマニピュレータの姿勢（特に床へアプローチする姿勢）よっては測域センサと干渉する。そこで、測域センサを上下逆に取り付けて、万が一マニピュレータがセンサに干渉した際に測域センサの取り付けブラケットが先に当たるようにした。

試作 3 号機のシステム構成

図 25 に試作 3 号機のシステム構成を示す。試作 3 号機では台車に内蔵された PC をメインコントローラと位置付け、USB-RS485 変換インタフェースを介して台車ユニット、マニピュレータユニットと通信を行う。また、PC には OS としてプリエンプティブカーネルパッチが施された Ubuntu10.04LTS(64 ビット版)がインストールされており、ゼネラルロボティックス社が開発した制御用ソフトウェアもインストールされている。開発当初は 4 ポート USB-RS485 インタフェースである Uport1405i(MOXA 製)を台車ユニットに内蔵し、マニピュレータユニット、台車ユニットを USB ケーブル 1 本で一括して通信しようと試みたが、リファレンスハードウェア制御ソフトウェアでマニピュレータを動作させたところステッピングモータによる間欠駆動のような不連続動作となったため、2 号機以前と同様 1 ポート版 USB-RS485 変換ケーブル Uport1130(MOXA 製)2 本で個別通信することとなった。また、試作 3 号機の電気系統図を図 26 に示すが、PC を搭載したことによって 1 号機、2 号機に比べて電気系統が複雑になっている。測域センサなどオプションの給電に関しては、台車左側面にオプション給電用電源端子を介して行う。電圧は 24[V]（バッテリーから直接出力）、12[V]、5[V]（台車に内蔵された DC-DC コンバータで変換）3 種類存在する。非常停止ボタンを入れた際には、各ユニットの動力系統、オプション機器の電源が切れるが、PC への給電は常時供給するようにした。

試作 3 号機の試用状況

平成 22 年当初の計画では RTC 再利用技術研究センター、東芝、奈良先端大学院大学、国際電気通信基礎技術研究所（ATR）で試用される予定であり、平成 22 年 10 月に 4 セット完成して配布準備を進めていた。しかしながら、同時に配布される予定であったゼネラルロボティックス製リファレンスハードウェア専用ソフト

ウェアの開発が遅れたため、まず平成 22 年 11 月に RTC 再利用技術研究センターへ先行で引き渡すとともに、RTC 再利用技術研究センターらと共同でソフトウェアを中心とした動作検証を進めてきた。その後平成 23 年 1 月に計画変更で ATR への試用が取りやめとなり、代わりにリファレンスハードウェア制御用ソフトウェア開発検証用にゼネラルロボティックスへ 1 セット引き渡した。その後平成 23 年 2 月末にソフトウェアが完成したのを受けて、3 月目処で他機関へ配布準備を進めていた。しかしながら平成 23 年 3 月 11 日に東日本大震災に見舞われ、エネルギー供給不足や計画停電等の影響で交通手段や運搬手段の確保が困難になったため 3 月下旬にずれ込んだが、年度内には東芝および奈良先端大への引き渡しが完了した。

目標に照らした成果の達成度および今後の展開について

当該年度内でのプロジェクトの達成度については、まず試作 1 号機の段階でプロジェクト開始当時に掲げた要求諸元を満たす動作を実現し、その後 2 号機で信頼性、使いやすさの向上を図り、3 号機では要求諸元の大幅な見直しが入ったものの、2 号機までの開発ノウハウを生かしつつ基本的デザインを維持することに成功した。また 1 号機から 3 号機まで合計 18 セットをプロジェクト期間中に製作を完了させ、産総研、RTC 再利用技術センターをはじめ、知能化プロジェクトに参画している研究機関へ配布した。さらに知能化プロジェクトで作成されたすべての知能モジュールを検証する機関である RTC 再利用技術研究センターでは知能モジュール統合検証実験（来訪者受付システム）でリファレンスハードウェアが活用され、日本機械学会ロボティクス・メカトロニクス講演会、日本ロボット学会学術講演会、計測自動制御学会 SI 部門講演会といったロボット関連学会にてリファレンスハードウェアを活用した研究成果が発表されるなど（表 7 参照）、知能モジュール研究用プラットフォームロボットとしての役割を果たしていると考える。以上のことより、プロジェクト実施期間中での目標を達成した。

今後の展開については、現在社内で製造、販売体制を構築するとともに、リファレンスハードウェア組立マニュアルの作成など、製造への引き継ぎ作業を行っている。平成 23 年 9 月開催の日本ロボット学会学術講演会などで販売のアナウンスを行い、平成 23 年 12 月頃に web での受注開始というスケジュールで進める予定である。販売価格は海外製研究用マニピュレータロボットと電動台車ロボット 1 セット分の半額以下（300 万円台）に押さえるようにしたい。

③ ロボット知能ソフトウェアプラットフォームの検証

(a) 検証用知能モジュール群の統合検証

ロボット知能ソフトウェアプラットフォームの機能検証の一環として、開発した検証用知能モジュール群の統合システムの統合検証を実施した。

平成 20 年度には、NEC の開発したコミュニケーション知能モジュール、シナリオ実行モジュールとともに前川製作所が開発したリファレンスハードウェア試作 0 号機台車に市販ロボットアーム Katana を搭載したロボットに搭載し、その有効性を平成 21 年 1 月 27 日に芝浦工業大学豊洲校舎で行われた先行デモにて検証した。また、検証用移動知能モジュール群については芝浦工大コンソ、日本 SGI コンソとともに先行共同デモを行い共通仕様に基づくモジュール化の有効性を示した。

先行デモに引き続き、前川製作所の開発したリファレンスハードウェア試作 1 号機に平成 21 年度に開発したいくつかの検証用知能モジュールを搭載し、その有効性を検証した。

また、検証用移動知能モジュール群については平成 21 年 11 月に開催された国際ロボット展において芝浦工大コンソ、日本 SGI コンソとともに共同デモを行い共通仕様に基づくモジュール化の有効性を示した。

平成 21 年度最終四半期には、検証用知能モジュール群をリファレンスハードウェアシステム試作 2 号機に RT コンポーネントとして搭載し、図 156 に示す面積 30 平米の検証環境において介助犬の行う種類の介助動作（図 157 参照）を図に示したように開発した検証用知能モジュール群を統合することで実行できることを検証した。

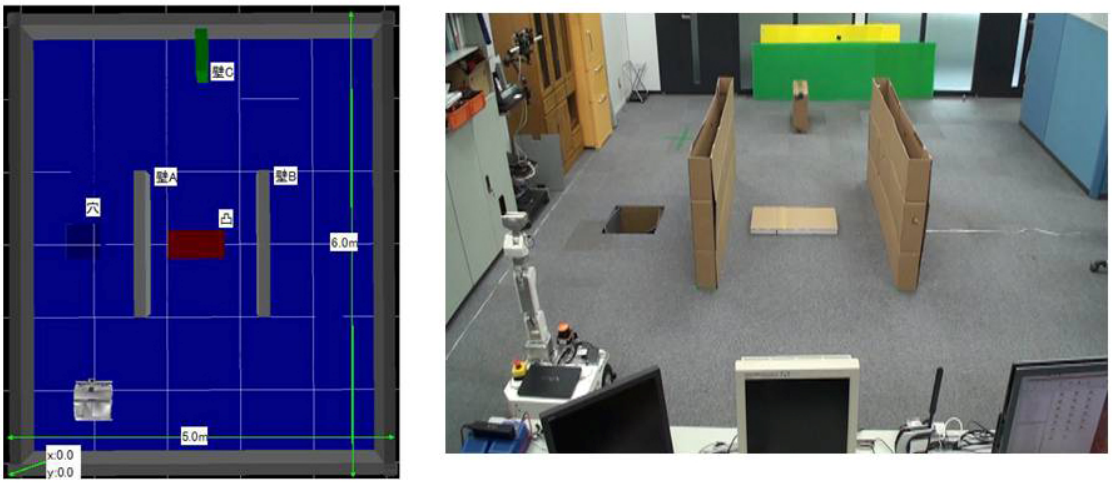


図 156　面積 30 平米の検証環境

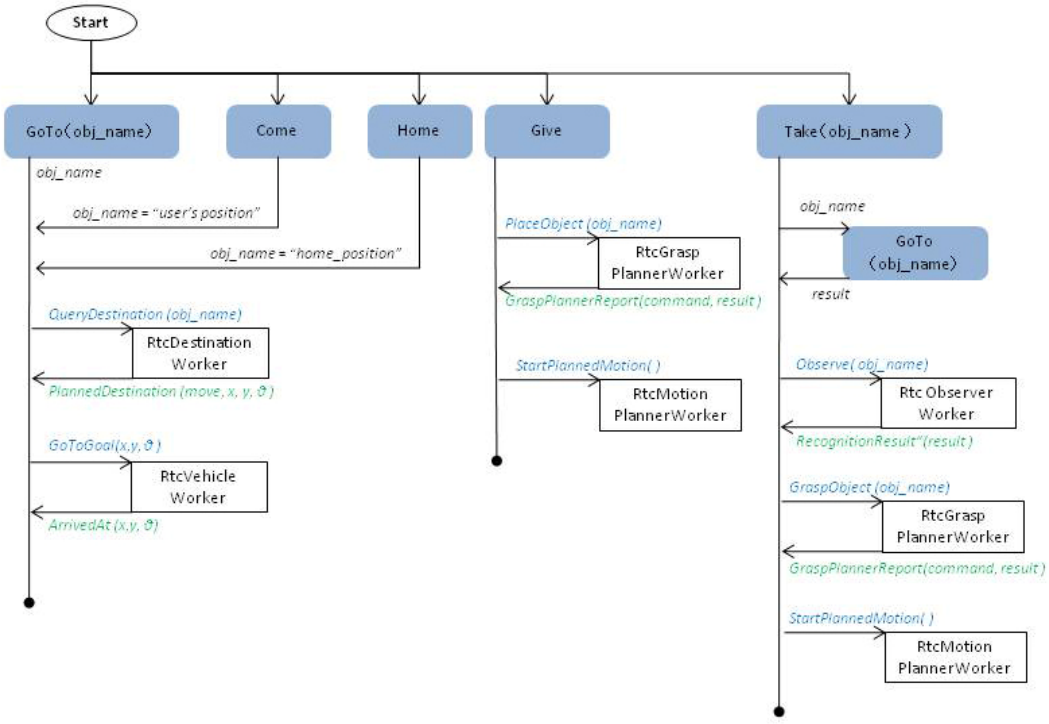


図 157　介助犬の行う 5 種類の介助動作

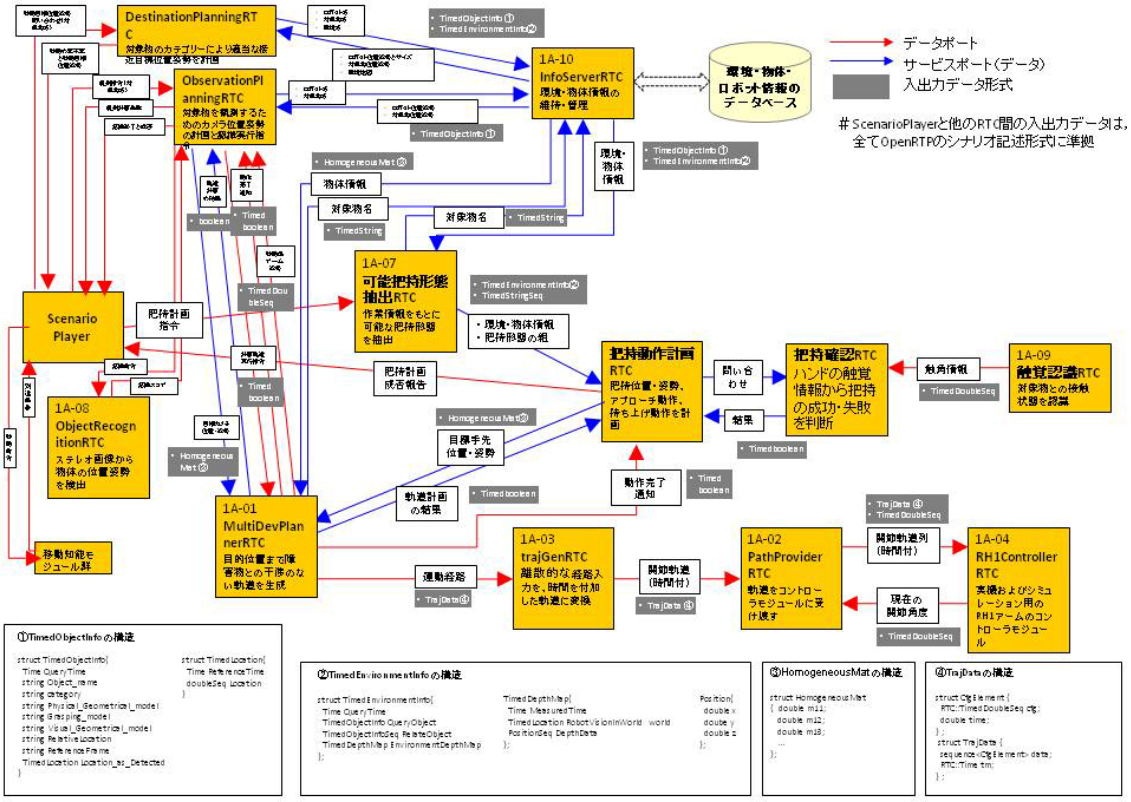


図 158　検証用作業知能モジュール群接続図

(b) オープンソースライセンスの知能モジュール群のドキュメント作成

本プロジェクト内で開発されたオープンソースライセンスの知能モジュール群について、その再利用を促進するために、開発者と RTC 再利用技術研究センター等の検証機関で共同してドキュメント作成を行うためのシステムを構築し、知能モジュールおよび知能モジュールを組み合わせた RT システムの構築に関するドキュメントの作成を実施した。本研究開発項目は、当初の実施計画にはなかったが、知能モジュールの検証と、再利用性を向上させるために実施した。

本プロジェクトに対する期待として、マニュアルや仕様書の書式統一だけでなく、ソフトウェア品質作り込み基準、プログラミング作法、モジュール入出力仕様の考え方などの統一を図ることで、利用者が混乱することなく容易に利用可能で、かつ信頼性を担保したソフトウェアモジュールの実現が望まれていた。さらに、RT ミドルウェアの利用者の拡大に向けた施策（マニュアル作成、プラットフォームの使い勝手と信頼性の向上、情報開示の拡大、啓蒙、サポート）に力点を移していく必要性もあった。

そこで、我々は、知能モジュール群やそれらを組み合わせたロボットシステムのマニュアル、仕様書や開発の統一基準の策定、再利用性確保のためのガイドラインを整備するとともに、利用者の拡大、普及促進に向けた初心者向けパンフレットやチュートリアルの作成を行った。

パンフレットは、本プロジェクトの内容やロボットのモジュール化の意義、移動知能、作業知能、コミュニケーション知能の代表的なモジュール例とモジュールの構成例を、一般のロボット開発者に分かりやすいものとなるように作成した。このパンフレットは、国際平成 23 ロボット展にて、2000 部を一般配布し、「概念が理解できた」、「モジュール化にイメージが分かった」など、高評価を得た。

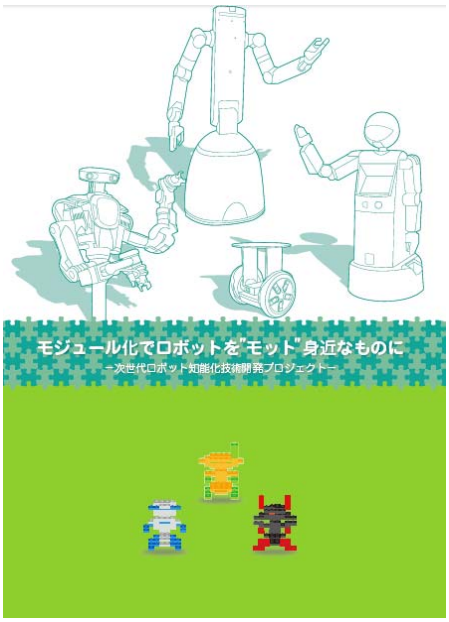


図 159　パンフレット「モジュール化でロボットをモット身近なものに」

次に、RT ミドルウェアを用いたロボットシステムの開発や、知能モジュールを再利用するための敷居を下げるため、初心者向けのチュートリアルを作成した。チュートリアルでは、移動知能、作業知能、コミュニケーション知能の各種モジュールを組み合わせた統合ロボットシステムについて、シミュレータ上で開発を体験できる内容とした。さらに、知能モジュールの利用の裾野を広げるために、市販の 2 足歩行ホビーロボットへの RT ミドルウェアの適用方法についてもチュートリアルとしてまとめた。

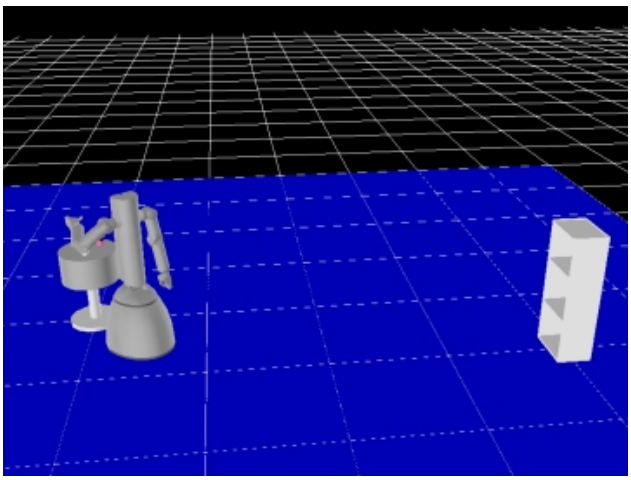


図 160　統合ロボットシステムのチュートリアル

知能ロボットシステムのマニュアルとしては、自律移動システム、双腕ヒューマノイド作業システム、音声対話システムなど 8 種類ロボットシステム（チュート

リアルを含む）の利用手順をまとめた。また、知能モジュール群のマニュアルとしては、移動知能、作業知能、コミュニケーション知能の 9 つのモジュール群（74 モジュール）の仕様、利用手順をマニュアル化した。これにより、それぞれの知能ロボットシステムや知能モジュール群の再利用性が向上した。さらに、これらのマニュアルをオープンに公開することで、マニュアルの共通書式として、マニュアルの記述レベルや品質を再利用することが可能になった。

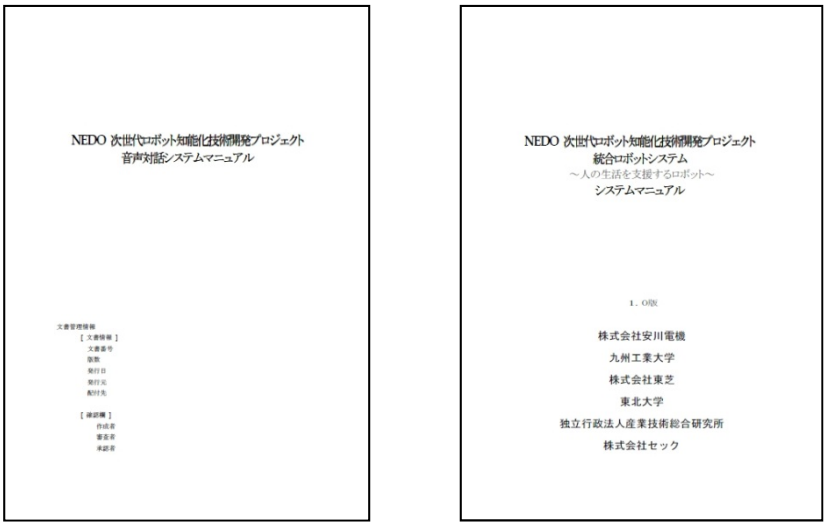


図 161　知能ロボットシステムおよび、知能モジュール群のマニュアル例

プログラミング作法やモジュール入出力仕様の共通化などの知能モジュールの再利用性確保に向けては、RTC 再利用技術開発センターと共同で、知能モジュールの共通インタフェース（I/F）仕様書を作成した。この共通 I/F 仕様では、本プロジェクトで共通化が図れた移動共通 I/F、作業共通 I/F、コミュニケーション I/F、カメラ画像共通 I/F、作業用画像認識共通 I/F、双腕ロボット共通 I/F の 6 つの共通インタフェースについて、仕様を取りまとめ、一般に公開することができた。この共通 I/F に準拠した知能モジュールも多数公開されており、モジュールの再利用性の向上に寄与することができた。

図 162　知能モジュール共通インタフェース仕様書例

作成したパンフレット、チュートリアル、各種マニュアル、共通インタフェース仕様書については、OpenRTM-aist サイトや各機関、セックのロボットサイトにて、一般公開している。これらのドキュメントのライセンスはクリエイティブ・コモンズ「表示 2.1 ライセンス」で配布しており、ドキュメント自体の再利用も可能となっている。

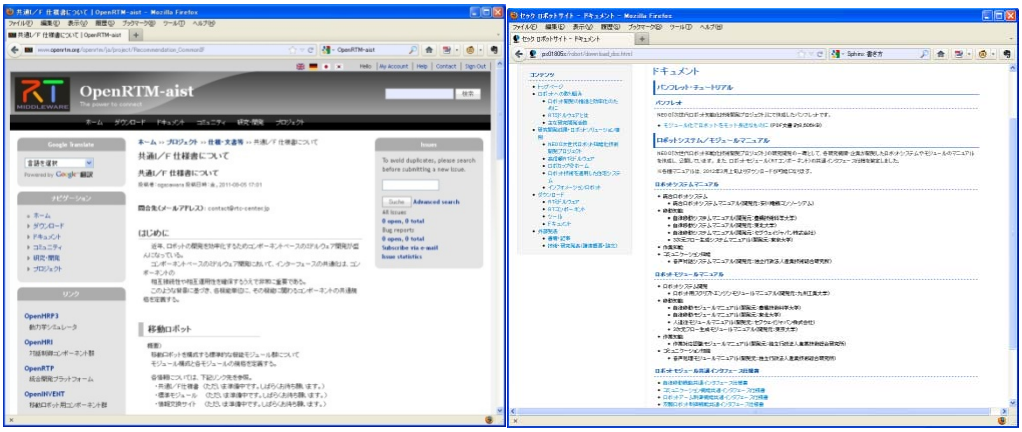


図 163　ドキュメントの公開サイト

(c) オープンソースライセンスの知能モジュール群の特許侵害調査

本プロジェクト内で開発されたオープンソースの知能モジュール群について、その再利用を促進するために、作業知能、移動知能、コミュニケーション知能の一部について、特許侵害調査を実施した。これは、当初の実施計画には含まれなかったがオープンソースライセンスの知能モジュールを広く再利用し、産業化を促進するために、平成 23 年に実施した。

本調査は、知能モジュール群で用いているアルゴリズム等に関して、日本、米国、

欧州を中心に既存の第三者特許に関する権利侵害の可能性を調査したものである。この調査の対象としては、RTC 再利用技術研究センターにおいて実施した統合検証システムで用いた移動知能モジュール群とコミュニケーション知能モジュール群の中でもオープンソースライセンスによって開発されたものに限定をした。対象としたプログラムを特徴づける技術内容（技術仕様）について、既存の国内、米国の特許について、権利侵害の可能性を技術仕様との関連性の度合として捉え、ランク分けを行い、傾向分析を実施した。

また、この調査の目的は、オープンソースで開発された知能モジュール群が広く利用されるために、そのプログラムで利用されている技術が、すでに登録または出願済の特許又は実用新案の権利侵害可能性を明らかにすることとした。このような侵害予防調査では、知財ビジネスでいう製品化前調査、サービス実施前調査と同様であり、プログラム仕様と第三者特許を対比して比較・分析する手法が適用でき、一般にはクレームチャート分析、関連性評価等といわれている。しかし、本調査に当たって、調査対象となる技術仕様書は製品販売前の製品仕様や実施前のサービス仕様として記載することが困難であったため、仕様規定を包括的に記載したものを使用した。したがって、このような場合の侵害予防調査では、上記のクレームチャート分析をベースに、関連性の評価ランクの設定を工夫し、関連性の高い特許を包括的な判断によって抽出する必要がある。関連性には、一致/類似性、包含性、基本性等が含まれる。

この調査では、大きくは 3 ランク（その上位、下位に 3 ランクを設定）での絶対評価を行い、個々に評価の観点や対比コメントを付すことによる分析を行った。

移動知能モジュールに関する技術的な特徴は以下のように整理した。

表 29　移動知能モジュールに関する技術的な特徴

	技術区分	技術要素	主なキーワード
自己位置推定	オドメトリ	既知の初期位置から移動車輪の単位時間当たりの回転角を積算して位置を算定	移動車輪ロボット 移動ロボット
	天井画像マッチング	● 移動範囲全体にわたる天井画像（蛍光灯の配置パターンを利用）を地図とする ● 広角レンズ撮影画像と地図を比較（テンプレートマッチング）	天井画像 蛍光灯配置 広角レンズ撮影画像 テンプレートマッチング

		● 並進移動と回転移動を推定	併進移動、回転移動
	LRF 距離データを用いた自己位置推定	● 距離センサにて周囲の障害物までの距離を測定 ● 占有格子地図のどこの地点にいるかを推定 ● 事後確率最大化またはベイズ推定	測距センサ、LRF 占有格子地図 モンテカルロ位置推定（パーティクルフィルタ）
経路計画	所与の経路地図で、現在位置から目的地までの二次元の地図上で移動すべき最短経路をダイクストラ法により検索	● 経路地図の表現形式 ● 最短経路の検索方式	経路地図 最短経路の検索 ダイクストラ法
経路追従	経路の追従制御	直進移動とその場回転の組み合わせ移動	直進移動 その場回転
障害物検知	ある高さの二次元平面上を測域センサでスキャン	● 二次元平面 ● 測域センサでスキャン	二次元平面 測域センサスキャン
	ロボットの中心座標及び位置を計算	中心座標及び位置計算	位置計算
	ロボットの中心から一定距離以内の円筒物体を障害物として検知	自己中心から一定距離内の障害物検知	中心から一定距離内円筒物体
障害物回避	ロボット移動系のキネマティックスから、所定の加速度・角加速度の範囲内で停止可能か否かを判断	キネマティックス（時間経過に伴う空間的なロボットの運動（変位）利用	キネマティックス 停止可能性判断

	障害物と衝突せず、かつ当初の目標速度に近い移動速度を出力 回避経路の生成では、回避優先か減速優先かいずれかを選択可能	● 回避優先では移動速度を維持しつつ回避経路を生成 ● 減速優先では角度を維持しつつ減速する動作生成	障害物回避 回避優先 減速優先

また、コミュニケーション知能モジュールに関する技術的な特徴は以下のように整理した。

表 30　コミュニケーション知能モジュールに関する技術的な特徴

	技術区分	技術要素	主なキーワード
システム系	ネットワーク分散コンポーネント技術による音声入力・音声認識・対話処理システム	● 音声入力・音声認識・対話処理システムがコンポーネント化 ● ネットワーク上の複数 PC で処理分散 ● ロボット依存	コンポーネント、ネットワーク、分散処理
音声認識系	HMM による音響モデルを用いた音声認識システム	HMM と有限状態オートマトンによる記述文法	HMM、有限状態オートマトン、音響モデル、記述文法
	単語信頼度を算出する音声認識システム	2 パス探索アルゴリズムによる単語信頼度を計算	2 パス探索アルゴリズム、単語信頼度
	GMM を用いた音声区間検出処理、デコーダベースの音声区間検出処理		GMM、デコーダベース 音声区間検出
音声認識・対話系	認識文法を動的に切り替える音声認識・対話処理システム	● 対話処理で有限状態オートマトンによる文脈管理情報を生成 ● 音声認識文法の切り替え	有限状態オートマトン、文脈管理、認識文法切り替え

		● ロボット依存	
	音声認識の第一候補だけでなく第 2 候補以降を利用	● 第 2 候補以降選択 ● ロボット依存	単語信頼度を用いた対話処理
入力フィルタリング系	適応フィルタによるエコーキャンセル処理	雑音低減処理	適応フィルタ エコーキャンセル処理
	MMSE-STSA 法に基づく雑音低減処理	雑音低減処理	MMSE-STSA 法 雑音低減処理
	SRP-PHAT 法に基づくマイクロホンアレイを用いた音源定位処理	マイクロホンアレイを用いた音源定位	SRP-PHAT 法 音源定位処理
	音線の位相差を用いた雑音抑制処理	● 音源定位結果に基づくマイクロホンアレイの各マイクに入力される音線信号処理 ● 明瞭な音声信号抽出 ● ロボット依存	位相差、雑音抑制処理

この調査では、下記の結果が得られた。

移動知能モジュールに関して、調査母集団の選定・抽出は、技術仕様書を精査し、技術的なポイントに合致する特許を検索できる特許分類（IPC、FI、F ターム、OSPC）とプロダクトの特性を示すキーワードの組み合わせによって行った。

その結果、国内特許 1657 件、米国特許 1325 件、計 2982 件を抽出し、技術仕様書との対比調査をおこない、一次スクリーニングで 130 件、二次スクリーニングで 35 件を抽出し、その特許の請求項と技術仕様書の関連性について詳細調査をおこなった。

関連性が高いと評価された（評価 4～5）公報の出願人傾向、出願の年次推移を調査した。

また、コミュニケーション知能モジュール群に関しては、調査母集団の選定・抽出は、技術仕様書を精査し、技術的なポイントに合致する特許を検索できる特許分類（IPC、FI、F ターム、USPC）と産総研プロダクトの特性を示すキーワード

の組み合わせによって行った。
その結果、調査母集団として国内特許 1219 件、米国特許 1294 件、計 2513 件を抽出し、技術仕様書との対比調査を行った。さらに、1 次スクリーニングで 194 件、2 次スクリーニングで 20 件を抽出し、その特許の請求項と技術仕様書の関連性について詳細調査を行った。
関連性が高いと評価された（評価 4～5）公報の出願人傾向、出願の年次推移を調査した。
今回の調査では、さらに以下のような課題が残されている。関連性に関しては、今回はプログラム全体の基本仕様、概念設計仕様のレベルで対比した。今後、侵害可能性を事前に把握するには、今回提示された各技術項目について、場合によってはクレームチャート等の手法を使った詳細な検討が必要となる。
今回の調査では、外国特許として主に米国特許に絞った調査を行った。今後、侵害予防の観点からは、欧州をはじめ、中国や韓国への出願特許調査をも考慮する必要がある。
近年、新製品の販売開始を機に、パテントトロールといわれる企業が、製品に関する基本特許の保有を主張するケースが増えている。この傾向から、予防措置としてこの種の調査は、技術の産業界への普及の促進する作用が期待され、継続的な調査が必要となる。
この調査の詳細に関しては、各知能モジュール群の公開サイトに公開した。

3.1.3 結び

本研究開発では、次世代ロボットシステム開発に必要な作業、移動、コミュニケーションを行うための知的機能を共通部品化し、それを組み合わせることでさまざまなタイプのロボットシステムを効率的かつ低コストで実現しうるロボット知能ソフトウェアプラットフォームの開発を行った。開発したロボット知能ソフトウェアプラットフォームは、主に、様々なロボット知能化技術を RT コンポーネントとしてモジュール化し、これらを統合して次世代ロボットシステムのシステム設計、シミュレーション、動作生成、シナリオ生成を効率よく実施するためのツール群、開発した知能モジュール群を多種多様な実行環境で動作させるためのミドルウェアから構成されている。このロボット知能ソフトウェアプラットフォームは、プロジェクトの他の研究機関で開発される知能モジュール群の基盤となるため、プロジェクトの前半において知能モジュール開発に必要な機能の実装を行い、プロジェクト内の他の研究機関への提供する必要性があったため、本研究開発においては、作業知能、移動知能、コミュニケーション知能を含む検証ロボットシステム開発を、ソフトウェアプラットフォームのツール群の開発と並行して実施することで、より使いやすく安定したツール群に開発を実現することができた。また、ここで開発を実施した検証用知能モジュール群は、

そのほとんどをオープンソースライセンスで開発を実施することで、ソフトウェアプラットフォームの機能検証のみならず、他の研究項目で開発されている知能モジュール群との共通のインタフェースを実装していたため、研究開発が終了したプロジェクトの後半でも他の研究項目において活用され続けていた。
ロボット知能ソフトウェアプラットフォームに関しては、プロジェクト前半において知能モジュール群を開発するための基本ツール群をほぼ完成することができ他の研究項目で活用されてきた。また、プロジェクト後半においては、プロジェクト内の他の研究実施機関からの要求に応じて、当初の実施計画に含まれないツール群の研究開発や統合システムのためのプラットフォーム開発を実施し、プロジェクト全体の進行と社会環境の推移にしたがって出てきた双腕ロボットプラットフォームの開発や高信頼 RT ミドルウェア、組込機器対応の RT ミドルウェア研究開発等を実施した。このようにプロジェクトの各研究開発項目の開発状況に応じて柔軟に研究開発項目の追加の実施を行うことで、より充実したソフトウェア開発基盤を実現することができた。
本事業における基本計画における最終目標は、以下の通りであった。

① 次世代ロボットシステムの応用ソフトウェアの開発が、ロボット知能ソフトウェアプラットフォームを用いて効率よく実施できること。
② 本プロジェクトで開発される、作業知能モジュール、移動知能モジュール、コミュニケーション知能モジュールのすべてが、ロボット知能ソフトウェアプラットフォームに組み込み可能となること。
③ 次世代ロボットシステムの設計を支援する機能が、ロボット知能ソフトウェアプラットフォーム上に実現すること。

本事業で行ったロボット知能ソフトウェアプラットフォームのツール群および検証用知能モジュールの研究開発等の成果より、上記の最終目標は達成できたと考えられる。
また、基本計画にはなかったが、プロジェクト内で開発された知能モジュール群の再利用、普及を促進するための活動の一環として、プロジェクト内で開発されたオープンソースの知能モジュール群についてその開発者と RTC 再利用技術研究センター等の機関と共同して知能モジュールに関するマニュアル、解説等のドキュメント作成整備や特許侵害の調査を実施した。これらは、今後、プロジェクトの成果を広く普及させるための重要な成果であると考えられる。