

1. 使用環境

Ubuntu Linux 8.04 で動作確認

OpenHRP (3.1.0, 3.0.x), OpenRTM (0.42, 1.0)がインストール済みのこと (tvmet を使用)。
eclipse、RT System Editorがインストール済みのこと
autotools (automake, autoconf, libtool)などのシステムツールがインストール済みのこと

以下ではpkg-configを使用するので、環境変数PKG_CONFIG_PATHを、インストール先（以下ではprefix）のlib/pkgconfig に指定する。

例えばprefixが /home/myname/develであれば /home/myname/devel/lib/pkgconfigになる。

rtm-configの変更

rtm-config (OpenHRP 経由でインストールした場合、/usr/bin/rtm-config) で、IDL ファイルをコンパイルする際の参照先ファイルが不備なので修正。

```
rtm_idlflags="-bcxx -Wba -nf"
```

を

```
rtm_idlflags="-bcxx -Wba -nf -I/usr/include/rtm/idl"
```

とする。

2. 必要ファイル

- Controller.tar.gz コンポーネント群圧縮ファイル
これを展開すると、以下のディレクトリが生成される。
 - RtmPlannerUtil データ構造定義 IDL
 - RH1Controller RH1 アームコントローラ本体
 - pathProvider 経路供給モジュール
 - trajGen 軌道生成モジュール
 - RH1_model RH1 モデルファイル

3. インストール

- これらの tar.gz ファイルをどこかに置く。インストール先ディレクトリは /home/myname/devel とする(変更可能)。
作業 1 : 以下のすべてについて共通の作業。\$はLinux 端末のプロンプトを表す。
\$ cd package_name
\$./autogen.sh
\$ mkdir build; cd build (システムディレクトリにインストールする場合、root 権限必要)
- rtmPlannerUtil のインストール
作業 1により展開、buildに移動
\$../configure --prefix=/home/myname/devel
\$ make; make install
/home/myname/devel/include/rtmPlannerUtil/idlにIDL ファイルがインストールされる

- RH1Controller のインストール

作業 1 により展開、build に移動

```
$ ../configure --prefix=/home/myname/devel ---with-tvmet=/usr/local
--with-openhrp3=/home/myname/src/OpenHRP3 --with-openhrp31-bin=/usr/local
```

注： --with-tvmet は tvmet がインストールされているディレクトリ (“include” 除く)

--with-openhrp3 は openHRP3 を展開したトップディレクトリ

--with-openhrp31-bin=/usr/local は openHRP3.1 の場合のバイナリインストール先
(とりあえずシミュレーション用なので実機では関係なし)

```
$ make; make install
```

/home/myname/devel/bin/RH1Controller 以下にコンポーネント群がインストールされる

- pathProvider のインストール

作業 1 により展開、build に移動

```
$ ../configure --prefix=/home/myname/devel
```

```
$ make; make install
```

/home/myname/devel/bin/pathProvider 以下にコンポーネント群がインストールされる

- trajGen の展開とインストール

作業 1 により展開、build に移動

```
$ ../configure --prefix=/home/myname/devel
```

```
$ make; make install
```

/home/myname/devel/bin/trajGen 以下にコンポーネント群がインストールされる

4. コンポーネントの接続とテスト

4.1. 軌道の例をテストする

- 立ち上げるコンポーネント (既にネームサーバ等は立ち上がっているとす。ポートは 2809 である)

インストールディレクトリは \$prefix (= /home/myname/devel など) とする。

```
$prefix/bin/RH1Controller/RH1RController/RH1RControllerComp
```

```
$prefix/bin/pathProvider/pathProviderComp
```

実機コントローラモジュール (リファレンスハードウェアアームモジュール)

- コンフィグレーション

```
RH1RController: RH1RBuffer
```

```
pathProvider: RH1RBufTest
```

- 接続 (データポート)

```
pathProvider: outCommandConfigSeq → RH1RController: thetarefBuf
```

```
RH1RController: comBufSize → pathProvider: CrntBufSize
```

```
RH1RController: theta → pathProvider: inCurrentConfig
```

```
RH1RController: vel → 実機コントローラモジュール速度入力
```

```
実機コントローラモジュール角度出力 → RH1RController: angle
```

- 実行

まず実機コントローラモジュール、RH1RController をアクティベート

pathProvider をアクティベートすると、軌道データファイルが読み込まれ、動作が開始される。

実行するデータは \$prefix/bin/pathProvider/RH1RPathBuf-1.dat である。

4.2. 逆運動学の計算

- 立ち上げるコンポーネント
\$prefix/bin/RH1Controller/RH1IK/RH1IKComp
\$prefix/bin/RH1Controller/kineTest/kineTestComp テスト用コンポーネント
- コンフィグレーション
RH1IK: RH1
- 接続 (データポート)
RH1IK: theta → kineTest inData
kineTest: outHandPos → RH1IK: hmat
RH1IK のもう一つの入力データポート initConf は、現在の位置である (オプション)
- RH1IK の機能
入力:
 - ベースからの手先座標 (J7 原点から TCP [Tool Center Point] 分移動した点) の 4x4 行列を入力とし (16 次元の TimedDoubleSeq)
TCP は \$prefix/bin/RH1Controller/data/RH1param.dat に記述
 - 初期姿勢 (6 次元の関節角 TimedDoubleSeq, rad)出力:
 - 初期姿勢に最も近く (ノルム最小)、目標手先位置を実現する関節角 (6 次元 TimedDoubleSeq)
- テスト:
RH1IKComp, kineTestComp をアクティベート。KineTestComp の端末で、

Command list:

```
rhlik (IK test with hand pos and ori)
quit
>
```

と出るので、“rhlik” と入力し、

Please input reference x[mm], y[mm], z[mm], rx[deg], ry[deg], rz[deg]:
と聞かれたら、

0.45 0 0.25 0 90 0 (コンマ「,」はなし)

などと目標位置、姿勢 (ロール・ピッチ・ヨー表現) を入力する。

次の入力後に計算後返ってきた値が表示される。

4.3. 軌道の生成

- 立ち上げるコンポーネント
4.1 のコンポーネントに加え、
\$prefix/bin/trajGen/tranGenComp

- コンフィグレーション
RH1RController: RH1RBuffer
pathProvider: RH1RBuffer
trajGen: RH1R
- 接続
4.1と同様
trajGen: outTrajectory → pathProvider: inPath
trajGenの入力側: ユーザの経路計画モジュール
- 実行:
すべてアクティベートする。
trajGenの入力は"TrajData"
(\$prefix/include/rtmPlannerUtil/idl/PlannerData.idl 参照)である。

関連する PlannerData.idl 上の定義は以下のとおりである。

```

struct CfgElement {
    RTC::TimedDoubleSeq cfg;
    double time;
};

struct TrajData {
    sequence<CfgElement> data;
    RTC::Time tm;
};

```

経路を TrajData の関節角ベクトルの列に値 (RH1 の場合は 6 次元) を入れて構成する。これを trajGen に渡すと、RH1 に適したサンプリングで軌道に直して pathProvider に渡す。生成された軌道は、4.1 と同様に実行される。

5. シミュレーションでの使用

5.1. モデル、コントローラのインストール

現在はアームのみのモデルを使用しているので、それをシミュレーションに使用する。

- RH1_model の展開とインストール

作業 1 により展開、build に移動

```
$ ../configure --prefix=/home/myname/devel ---with-openhrp3=
/home/myname/src/OpenHRP3
```

```
$ make; make install
```

これで、/home/myname/src/OpenHRP3/etc/RH1 に RH1 のモデルがインストールされ、以下のシミュレーションではこれを用いることになる。

- コントローラのインストール

シミュレータのコントローラは、OpenHRP3 のコントローラブリッジを使うので以下のように OpenHRP3.1 のバイナリのパスを指定して configure し、インストールする。

OpenHRP3.1 のインストールが prefix /usr/local に行われている場合 (つまり、

openhrrp-controller-bridge が /usr/local/bin 上にある場合)

3. インストールで --with-openhrp31-bin=/usr/local としておく必要がある。

インストールしなおす場合は、3章の RH1Controller のインストール方法に従い、
make clean;

```
../configure --prefix=/home/myname/devel ---with-tvmet=/usr/local  
--with-openhrp3=/home/myname/src/OpenHRP3 --with-openhrp31-bin=/usr/local
```

注(再掲) :

--with-tvmet は tvmet がインストールされているディレクトリ ("include" 除く)

--with-openhrp3 は openHRP3 を展開したトップディレクトリ

--with-openhrp31-bin=/usr/local は openHRP3.1 の場合のバイナリインストール先
make; make install とする。

5.2. 立ち上げるコンポーネント

GrxUI では、プロジェクト \$prefix/bin/RH1Controller/project/RH1-rtm-1.0.xml を
ロードしておく。アームのみがシミュレーション画面に現れる。

コントローラの立ち上げ

\$prefix/bin/RH1Controller/RH1VController に移動

sh ./BridgeConnect.sh を実行

System Editor 上に RH1VController が現れる

他のコンポーネント

\$prefix/bin/pathProvider/pathProviderComp

5.3. コンフィグレーション

RH1VController: RH1Buffer

pathProvider: RH1VBufTest (立ち上げ時に軌道の例を読み込む場合)

RH1Buffer (trajGen から軌道を受け取る場合)

5.4. 接続

pathProvider: outCommandConfigSeq → RH1VController: thetarefBuf

RH1VController: comBufSize → pathProvider: CrntBufSize

RH1VController: theta → pathProvider: inCurrentConfig

5.5. シミュレーションの開始

- GrxUI の "Start Simulation"  をクリック。
"Controller 'RH1VController' may already exist. Restart it?" と聞かれたら Yes をクリックしてシミュレーションがスタートする。正常に動作している場合はシミュレータ内でアームは動かず、初期状態でサーボがかかった状態となる。
またこのとき、Eclipse の System Editor からコントローラをアクティベートする必要はなく、GrxUI でシミュレーションを開始すれば System Editor 上の RH1VController コンポーネントが緑になる。
- System Editor から pathProvider をアクティベート。サンプルの軌道を読み込まれてロボットが動作する。

5.6. TrajGenを使用する場合 (4.3と同様)

trajGenのコンフィグレーション: RH1V

pathProviderのコンフィグレーション: RH1VBuffer

接続: trajGen: outTrajectory → pathProvider: inPath

pathProviderをアクティベートしておき、trajGenに経路データが与えられると、これをサンプリングした軌道データに変換してpathProviderに渡され、動作が開始される。