手先カメラを用いた双腕ロボットによる
マニピュレーションシステム
操作手順書

東京大学　情報理工学系研究科　稲葉研究室

平成 24 年 2 月 14 日

目 次

第1章　システム概要

本サービスは，工場での部品整理をイメージしたものである．具体的には手先のカメラを用いて作業台上の部品を認識し，両手で箱に整理して入れる機能を実現する．手を動かすことで複数の対象物を認識，両手の干渉を考慮して同時にアプローチできる対象物を選択する．

1.1　全体のモジュール構成

以下に，本システムで利用するモジュールの一覧を示す．

- app-recog
- CameraComp
- LoadPictureComp
- iv_plan_hironx
- HiroNXInterface

ファイルシステム上の場所は必ずしも重要ではないが，ディレクトリ構成は揃えておくと本ドキュメントとあわせて理解しやすい．

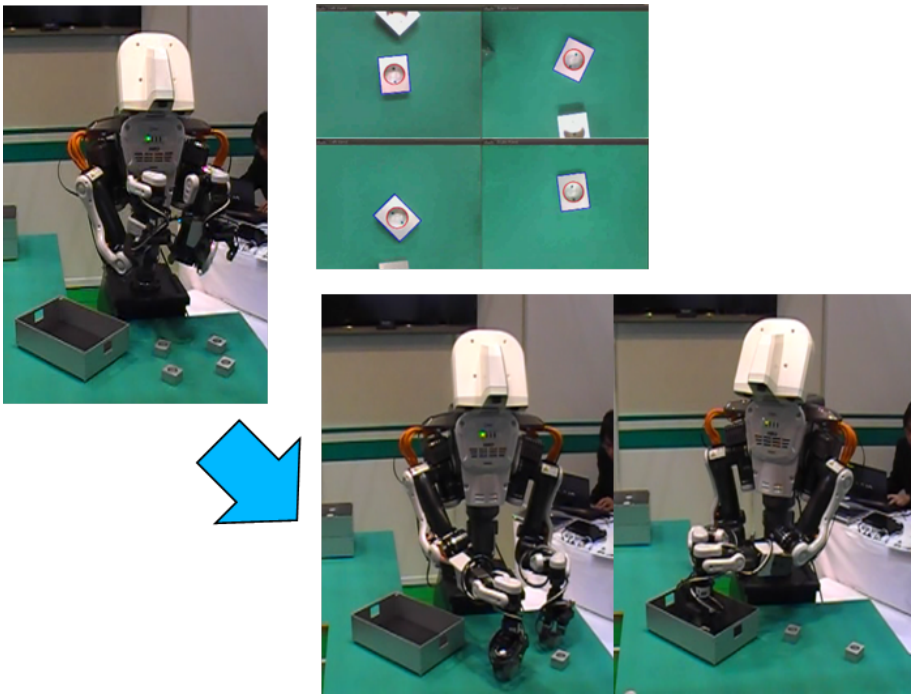

図 1.1: サービスイメージ

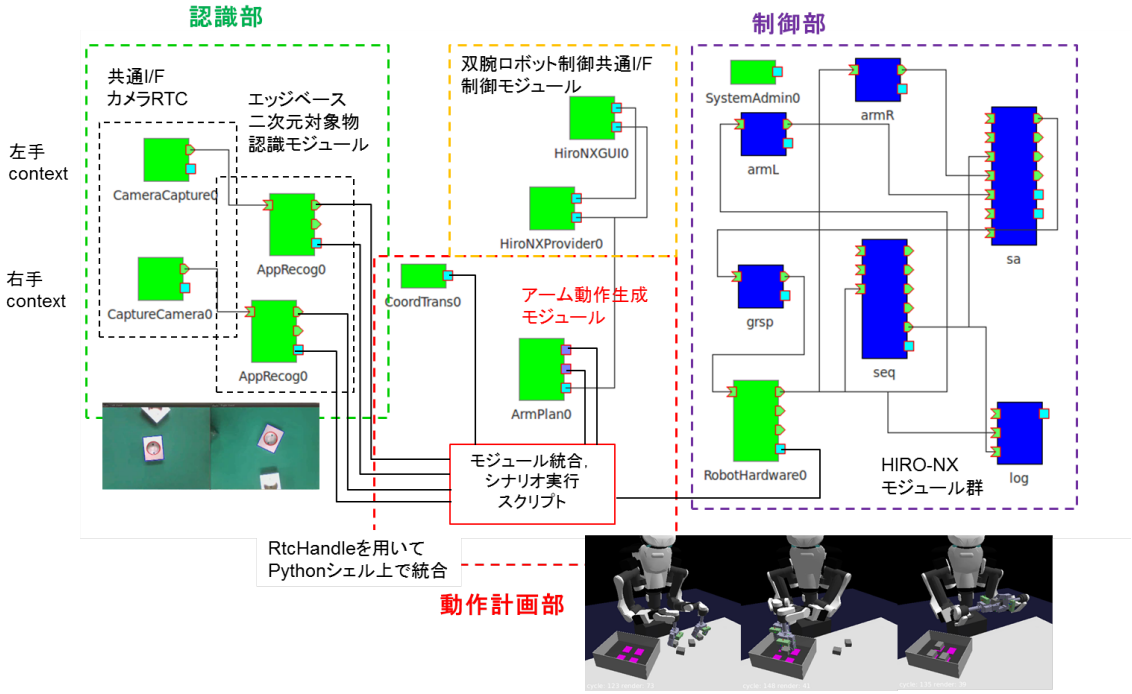

図 1.2: 全体のモジュール構成

1.2 認識部

1.2.1 エッジベース二次元対象物認識モジュール (AppRecog)

http://openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_HiroAccPrj_5002
HiroNX の手先に取り付けられた USB カメラで対象物を認識するために利用します．使い方はモジュール付属のドキュメントを参照してください．

1.2.2 カメラ共通 I/F 準拠の画像キャプチャモジュール (CameraComp)

http://www-arailab.sys.es.osaka-u.ac.jp/CameraIF/
大阪大学により開発され画像キャプチャモジュール CameraComp をダウンロード，コンパイルします．利用方法は，本モジュールのドキュメントおよび，上記 AppRecog 付属のドキュメントを参照してください．

1.3 動作生成部

1.3.1 (VPython 版)HiroNX 動作生成モジュール

認識結果を世界座標への変換，HiroNX の動作の生成，タスク記述を行なうために利用します．詳しくは，モジュール付属のドキュメントを参照してください．

1.3.2 HiroNXInterface

http://www.openrtm.org/openrtm/en/node/4645

双腕ロボットの制御コマンドの共通インタフェースに準拠する HiroNX 用制御モジュールです．詳細については，開発元である産業技術総合研究所のドキュメントを参照してください．

1.3.3 動作生成部の動作確認 (対話環境における動作生成)

ここからは，HiroNX 実機を使います．まず，HiroNXInterface 制御モジュールを起動，activate，接続し，利用可能な状態にします（詳細は HiroNXInterface のドキュメントを参照してください）．HiroNXInterface モジュールは起動後 GUI 上の「RTC Status」が緑になった状態で「Set up Robot」ボタンを押し RTC まわりの初期化を行う必要がある点に注意してください．次に，HiroNX 両手の手先カメラのキャプチャ及び，AppRecog 認識モジュールを起動します．左右で同じ名前のコンポーネント群を起動するため，別々の RTC 名前空間に起動します．

```
$ cd iv_plan/scripts
$ ./run.sh
このスクリプトはROSのツールであるrospack findを使います
もし，コンポーネントが上手く接続およびactivateされない場合
は，./comcon.shおよび./comact.shを再度実行してください．
```

カメラが認識できていない可能性がある場合は，xawtv コマンドなどでカメラ自体が認識されているか確認してください．

認識関係のモジュールを登録するネームサーバは scripts ディレクトリの rtc_left.conf および rtc_right.conf で指定します．

次に，以下のコマンドでデモプログラムを起動し，認識器と通信し認識結果を取得できること，HiroNX 本体のモジュールと通信し，関節角の取得および，動作コマンドの送信を行なうことができることを確認します．

```
$ cd iv_scenario/src
$ ipython demo_wexpo.py

外部モジュールとの通信
>>> rr = MyHiroNxSystem(portdefs) # 外部モジュールとの通信インタフェースオブジェクトを作成

認識できる位置に対象物を置いて
>>> rr.detect(sensor='rhandcam') # 右手ハンドカメラでの認識
>>> rr.detect(sensor='lhandcam') # 左手ハンドカメラでの認識
(対象物を認識できるまでブロックします)

認識結果の世界座標への変換
$ ipython demo_wexpo.py # 内部で上のinterface_wexpo.pyを利用しています．
>>> detect(sensor='rhandcam')
>>> detect(sensor='lhandcam')

>>> f = detect(sensor='rhandcam')
>>> show_frame(f) # 認識結果の表示
```

```
# シミュレータ内で認識位置に箱A0を移動させる
>>> env.get_object('A0').locate(f)

# 作業台上で両手を動かし，対象物を検出する
# 上手く検出できれば，シミュレータ内の箱4つが正しい位置に移動する
>>> look_for()
```

第2章　準備

必要に応じて行うロボットごとに以下の作業を行う.

2.1　モデルファイルの修正（力センサありとなし）

例えば手首に力センサがある場合とない場合で，ロードするモデルを変更する必要があります．サンプルプログラムは，力センサがある場合に手首のリンク長を変更，円筒形状を挿入したものです．力センサがない場合は，`HiroNX` インスタンスを生成するときに，オプション `forcesensor=False` を指定します.

2.2　キャリブレーション

ロボットの個体差を修正する作業です．デフォルト値は HiroNX14 号機のものであり，精度を出すには各機体ごとに行う必要があります.

2.2.1　カメラキャリブレーション

RT-middleware のコンポーネントでも ROS のノードでも何でもよいので単眼カメラのキャリブレーションを行い，CameraComp が読み込めるようにします．本システムにおいて，run.sh スクリプトで RTC を起動する場合，scripts ディレクトリにある camera_left.yml および camera_right.yml にそれぞれのカメラの内部パラメータを記述します．その後，エッジベース二次元対象物認識モジュールにおいて，対象物の位置，距離が正しく出力されているかどうか確認しておいてください.

2.2.2　カメラ取り付け位置のキャリブレーション

チェッカーボードの認識に ROS のノードを使用しています．キャリブレーションプログラムは，学習データとしてロボットの各姿勢におけるチェッカーボードの姿勢を入力とします．チェッカーボードの姿勢は tf として publish されたメッセージを python プログラムで受信します．しがって，準備として，

- ROS のカメラノードにより画像および上記カメラパラメータがトピックとして publish
- checkerboard detection ノードでそれらを subscribe し，推定した tf が publish

される状態にしておきます.

手順

1，机の上にチェッカーボードを置く

2，首を動かして学習データをとる

```
$ ipython hironx_calib.py
>>> res = record_data()
# 手先を動かして画像とそのときの関節角度値を取得する.
# すべての姿勢でチェッカーボードが視野に入り,
# 安定して認識できているかどうかを確認する.
```

3，頭リンクからkinectカメラへのtransformを計算する

```
>>> f = calibrate(res, height=960.0)
```

4，transformの書き換え

```
>>> r.Thd_kinectrgb = f
```

5，確認（正しい位置でキャリブボードのフレームが止まっていれば成功）

```
>>> play_data(res)
```

6，デフォルト値の更新

hironx_params.pyのThd_kinectdepthを書き換える

- 上のコマンドは頭部 kinect の場合なので，ハンドカメラの場合に変更する
- 原理について

第3章　デモの実行

3.1　認識部のモジュール起動と接続

```
$ cd iv_plan/scripts
$ ./run-hironx-interface.sh
$ ./run.sh
```

これにより，両手に対応するカメラ，認識モジュール，HiroNXInterface 制御モジュールが起動，接続され，さらに各モジュールが activate されます．HiroNXInterface モジュールは起動後 GUI 上の「RTC Status」が緑になった状態で「Set up Robot」ボタンを押し RTC まわりの初期化を行う必要がある点に注意してください．

ここで，rtc.conf, 左右カメラの左右のカメラが逆になっていないかどうか確認してください．逆になっている場合は，CaptureCameraLeft.conf と CaptureCameraRight.conf に記述してある conf.default.int_camera_id の番号を逆にして，run.sh を再実行してください．

3.2　動作生成プログラムの起動 (1 つの python プロセス上での実行)

- テーブル上に対象物を 4 つ配置し，look_for() 関数を実行します．手先を動かして机上の対象物を認識テストをします．
- このとき，画像中で対象物が正しく認識されているか，認識結果がシミュレータ中に正しく表示されているかどうかを確認します．
- これで準備が整ったので，実際にデモを実行します．demo() 実行後，ロボットはテーブルをスキャンし，対象物を認識，最初は両手で，残りの 2 つは適宜持ち替えを行なって箱に配置します．