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 8 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．

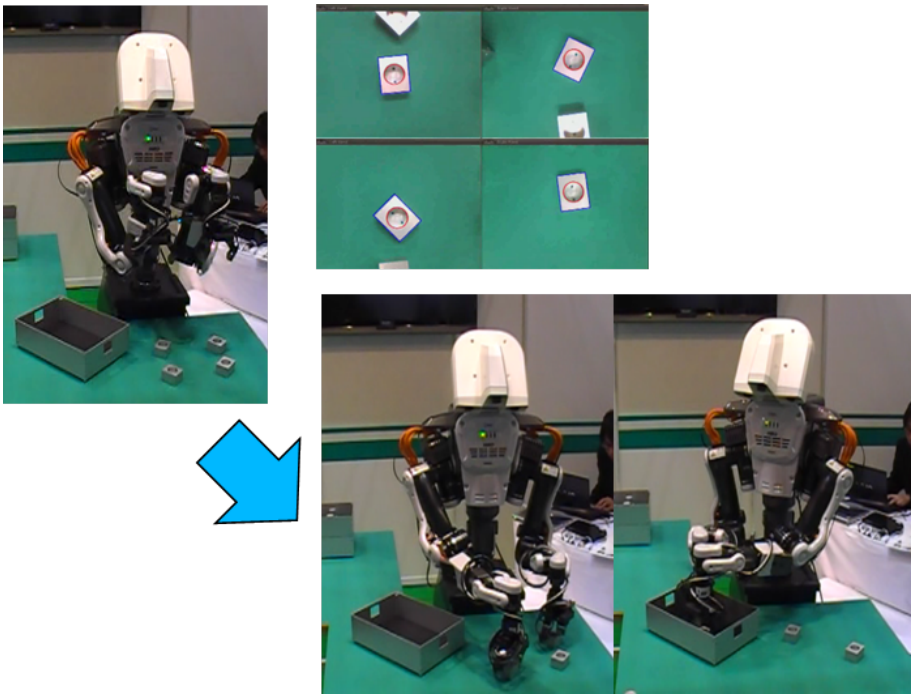

図 1.1: サービスイメージ

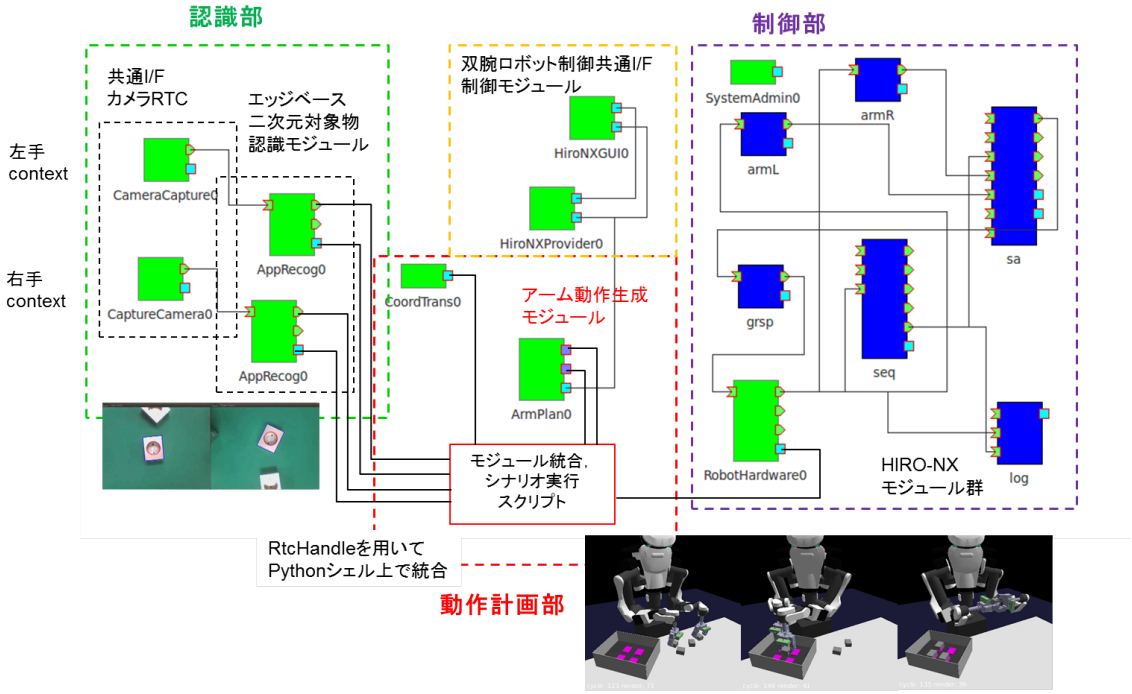

図 1.2: 全体のモジュール構成

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のモジュールである．カメラパラメータはデータポートを通して画像と一緒に送られてくるものを利用する．正しいカメラパラメータが入っていなくても対象物の認識はできるが，位置および姿勢は正しく推定されない．

ダウンロードとコンパイル

- 現状では google code(SVN) から開発版をチェックアウト
- どこかのバージョンの tar を作る予定

```
$ cd app-recog
$ make
```

カスタマイズ

連続的に送られてくる画像に対して認識を行うが，認識結果の時間方向の連続性は考慮せず，各フレームで一番尤度が高い位置を計算し，その尤度が閾値以上であれば検出結果を返す．

- 認識手法の簡単な説明

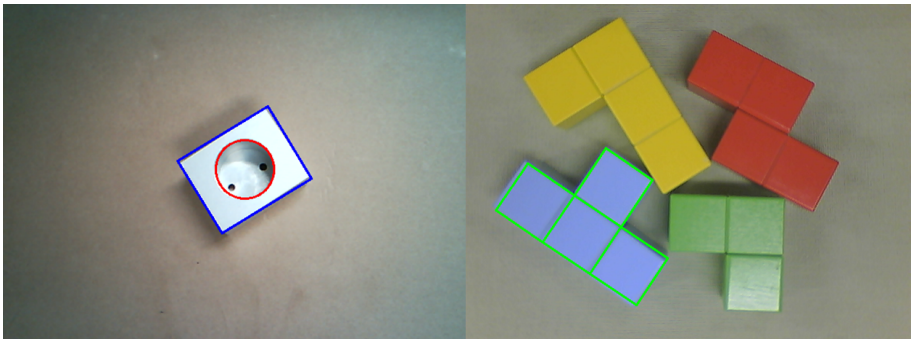

図 1.3: 認識例

- 閾値の変更方法，閾値の意味について
- 状態空間の範囲指定について
- サービスによる認識対象の切り替え
- 認識モジュールにおける対象物モデルの定義の仕方

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，コンパイルする．ログ画像によるテストを行うため，LoadPictureComp モジュールも同様にダウンロードするとよい．

1.2.3 認識部の動作確認

実際にカメラモジュールと接続し，オンラインでテストを行う．このときのモジュール接続は図 1.4 のようになり，実行手順は，以下の通りである．

1. 認識モジュール AppRecog とキャプチャモジュールをそれぞれ実行する．
2. rtshell で画像の入出力を接続する (system editor 上で操作してもよい)．
3. 2 つの RTC を activate する．

```
$ cd CameraComp
$ ./CaptureCameraComp
別端末で
$ cd app-recog/
$ build/bin/AppRecogComp
別端末で（rtctreeでのパスは適当に補完する）
$ rtcon CaptureCamera0.rtc:CameraImage AppRecog0.rtc:InputImage
$ rtact CaptureCamera0.rtc AppRecog0.rtc
```

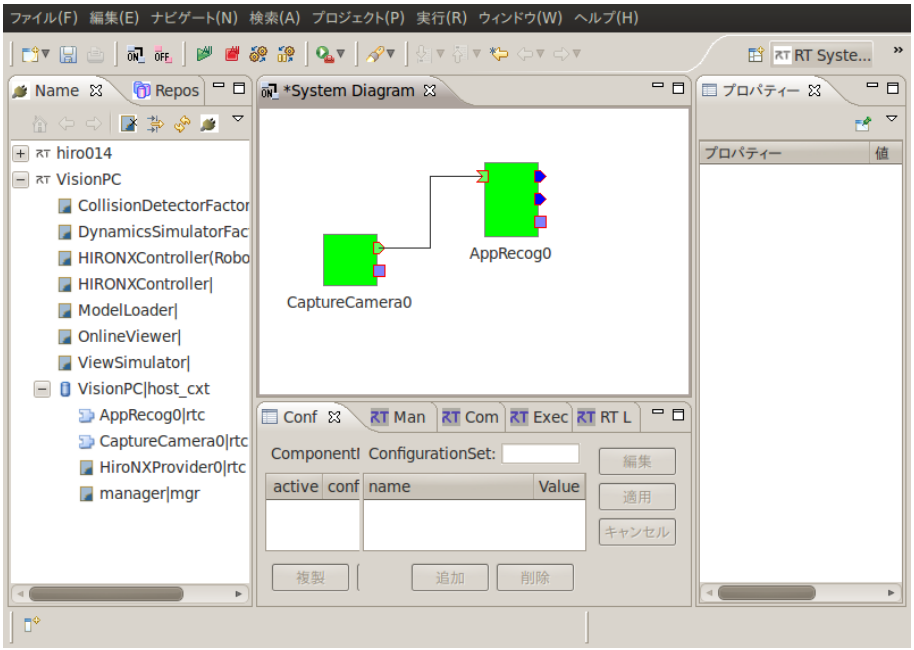

図 1.4: USB カメラを用いたテスト

初期設定で認識範囲のスケールを絞ってあるため，認識できない場合は対象物までの距離をいろいろ変えてみる．また，背景に模様がなく，対象物と異なる色のものを置くと認識しやすくなる．

テストとして，カメラモジュールを LoadPictureComp モジュールに差し替え，あらかじめ撮っておいた画像を用いてテストを行う場合，CaptureCameraComp を LoadPictureComp に置き換えた接続となる．ログ画像の指定は，LoadPictureComp モジュールの LoadPicture.conf で行う．AppRecog モジュール付属の画像 data/parts4.jpg を用いて確認を行うとよい．状態空間における探索範囲は別途指定が可能である．

1.3 動作生成部

1.3.1 (VPython 版)HiroNX 動作生成モジュール

http://openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_HiroAccPrj_5003

Python 対話環境において、幾何モデルを用いた動作生成システムを柔軟に構築するためのスクリプト群です．RtcHandle を用いて対話環境から RTC 構成によるシステムの各モジュールと通信を行うことで，システムの統合を行います．RRT-connect による双腕の干渉を考慮した動作計画機能を提供し，人手による動作記述とプランナによる動作生成をスムーズに統合できます．また，RTC として，作業共通インタフェースを実装する動作生成モジュールとして利用することもできます．

ダウンロードとコンパイル

install.sh (or rosmake –rosdep-install)

```
$ cd iv_plan/src
$ make
```

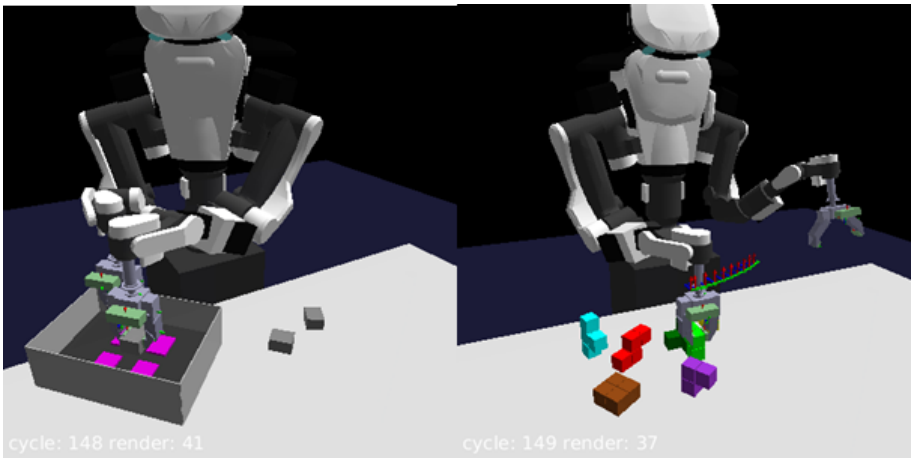

図 1.5: 動作生成モジュール

VPython は Ubuntu 10.04LTC の標準パッケージより新しいものにパッチを当て，コンパイル済みのものを利用する．したがって，標準の python-visual パッケージをインストールしている場合は，それが python 上で先にロードされることがないように注意する必要があります．

基本的な使い方

シミュレーションによるパーツの箱詰め
基本的に，ロボットクラス，ロボットインスタンスごとのカスタマイズは既存クラスを拡張することで行います．

- robot.py から hironx.py
- hironx_params.py 個体差がある各種 transform と，利用する外部モジュール定義
- 他の機能として，VRML のローダ，センサ取り付け位置の推定機能があります

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ある．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．

1.3.3 動作生成部の動作確認

HiroNX の起動は完了しているとする．

```
$ 起動スクリプト
IP/ホスト名の違いはどこを直せばよいか？
$ cd iv_scenario/src
$ ipython demo_wexpo.py
```

1.3.4 動作生成を対話環境で行う場合

1.3.5 動作生成を動作計画モジュールとして分離する場合

シミュレーション環境外部モジュールとの通信動作コマンド送信，状態読み込み

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）

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ある．デフォルト値は HiroNX16 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る.

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する.

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くとよい.

2.2.2　カメラ取り付け位置のキャリブレーション

現状では，チェッカーボードの認識に ROS のノードを使用している．原理について.

- hironx_calib.py を import
- record... 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.
- calibrate キャリブレーションの計算
- play 結果の確認
- 値の反映

第3章　デモの実行

3.1　認識部のモジュール起動と接続

- run.sh を実行 rtc.conf, 左右カメラのデバイス等の設定はできているものとする．左右のカメラが逆になっていないかどうか確認する．
- 動作生成プログラムの起動
- look_for 手先を動かして机上の対象物を認識する
- 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する
- 動作の実行