エッジベース二次元対象物認識モジュール (AppRecog)

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

1　概要

画像から二次元的に対象物を認識，その位置，姿勢およびスケールを出力するモジュールです．通常，カメラ出力共通 I/F で画像および，カメラパラメータを出力するカメラモジュールと組み合わせて利用します．

http://openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_HiroAccPrj_5002

2　ダウンロードとコンパイル

https://code.google.com/p/app-recog/ からダウンロードします．

```
$ tar xvfz AppRecog-0.1.0.tgz
$ cd app-recog
$ make
```

3　開発・動作環境

- Ubuntu Linux 10.04 LTS
- OpenRTM-aist 1.0.0-RELEASE C++版
- OpenCV 2.3

OpenCV のバージョンに注意してください．

4　インタフェース

- データポート
  - 入力: Img::TimedCameraImage (Img.idl)
    画像出力共通インタフェース準拠のカメラモジュールから，画像及び，カメラパラメータを受取ります．
  - 出力: TimedRecognitionResult (Vision.idl)
    認識結果共通インタフェースにしたがい，対象物体の位置姿勢を出力します．
    Img::TimedCameraImage 処理結果を画像として出力します．

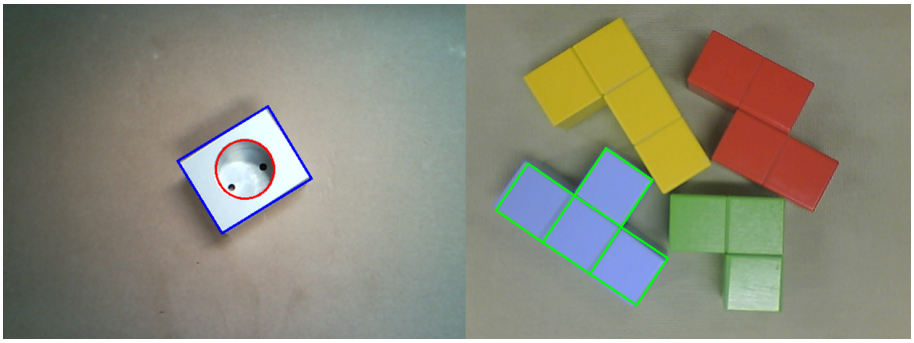

図 1: 認識例

- サービスポート
  認識対象のモデルを設定するために使います．あらかじめ．ModelFiles/ModelList.txt にモデル ID とモデル定義ファイル名を記述し，モデル ID を引数としてサービスコールを行います．setModelID(i) は，i 番のモデルを使用することを意味します．

認識結果は TimedRecognitionResult によって出力されます．具体的な出力内容は以下の通りです．現在，対象物の姿勢以外は入っていません．

```
 0: 0, 1: 0, 2: 0, 3: 0, 4: 0
 5: 0, 6: 0, 7: 0,
 8: R00,  9: R01, 10: R02, 11: Tx
12: R10, 13: R11, 14: R12, 15: Ty
16: R20, 17: R21, 18: R22, 19: Tz
```

5 詳細説明

与えられたモデルと，画像から抽出したエッジの Chamfer マッチングにより類似度を評価します．そして，粒子群最適化により類似度が最大の位置，姿勢，スケールを出力します．連続的に送られてくる画像に対して認識を行いますが，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します．指定した閾値以下の場合は，認識結果は出力しません．

モデルと実画像のマッチングは，画像上で行われます．検出した位置，姿勢，スケールからカメラパラメータを用いて，カメラ座標系における対象物の位置，姿勢が計算されます．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使用します．正しいカメラパラメータが入っていない場合も対象物の検出はできますが，出力される結果は正確ではありません．画像座標での (x, y, θ, s) の探索範囲，検出の閾値は，コンフィグファイル AppRecog.conf で指定できます．また，モデル定義ファイルは ModelFile ディレクトリの中に置き，ファイルは頂点と辺によって表現されるエッジと，円からなります．

6 実行およびテスト

まず，画像をキャプチャするモジュールを用意します．

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

以下の web から，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．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します．

http://www-arailab.sys.es.osaka-u.ac.jp/CameraIF/

6.2 カメラを用いた認識

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います．このときのモジュール接続は図 2 のようになり，実行手順は，以下の通りです．

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
2. rtshell で画像の入出力を接続 (system editor 上で操作してもよい)
3. 2 つの RTC を activate

```
$ 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
```

初期設定で認識範囲のスケールが絞ってあるため，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てください．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ります．

6.3 ログ画像を用いた認識

カメラモジュールを LoadPictureComp モジュールに差し替えることで，あらかじめ撮っておいた画像を用いてテストを行うことができます．RTC の接続は，CaptureCameraComp を LoadPictureComp に置き換えたものとなります．ログ画像の指定は，LoadPictureComp モジュールの LoadPicture.conf で行います．まず，AppRecog モジュール付属の画像 data/-parts4.jpg を LoadPictureComp のディレクトリにコピーし，以下のように LoadPicture-Comp が読み込むように設定してください．

LoadPicture.conf で読み込む画像を指定するには，rtc.conf に

```
Processing Module.LoadPicture.config_file: LoadPicture.conf
```

を，LoadPicture.conf に

```
conf.default.string_file_name: parts4.jpg
```

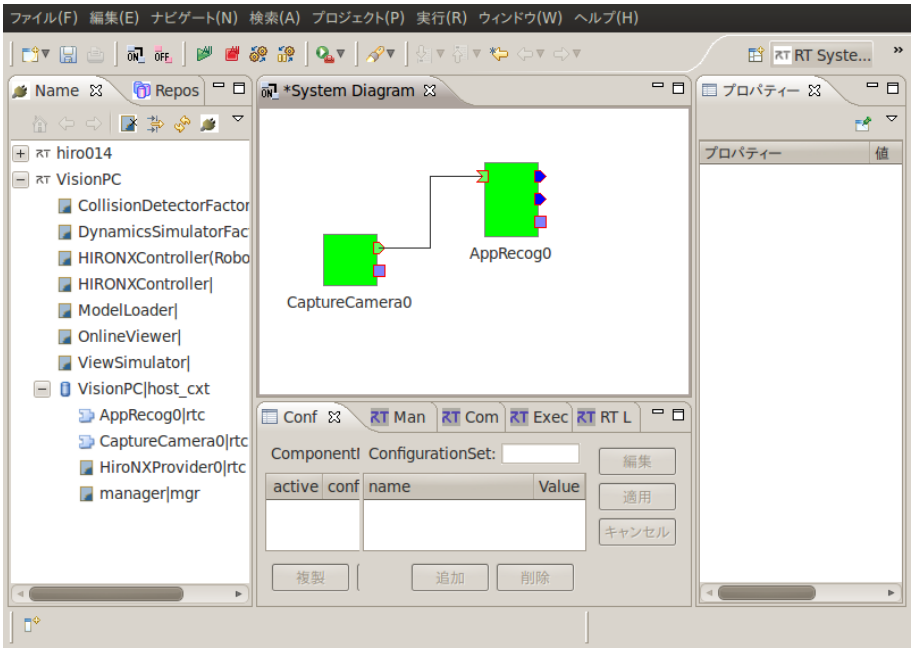

図 2: USB カメラを用いた場合の接続図

を記述します．parts4.jpg は読込みたいファイルの名前を書きます．正しく動作していれば，図 1（左）のようなウィンドウが表示されます．

次に，サービスを利用して認識対象の切り替えを行います．一度，実行しているコンポーネントを終了させた後，以下の設定を行なってください．

1. LoadPicture.conf で読み込む画像を，piece_red.jpg にする
   （画像は上と同様に LoadPictureComp ディレクトリにコピーしておきます）．

2. AppRecog における検出閾値を下げます．AppRecong.conf で

   ```
   conf.default.detection_threshold: 80.0
   ```

   とします．

ここまで設定をした後，LoadPictureComp と AppRecogComp を起動，ポートの接続，activate を行います．この状態で，parts を認識する設定になっています．次に，サービスで認識対象を変更するため，付属のテスト用クライアントを起動し，RTC の接続，activate を行った後，クライアントで認識対象の変更を行います．

```
$ build/bin/AppRecogComp
別端末で
$ rtcon AppRecog0.rtc:Recognition AppRecogConsumer0.rtc:RecognitionService
$ rtact AppRecogConsumer0.rtc

Command list:
 getModelID          : get current model ID.
 setModelID [value]: set model ID.
> setModelID 2
```