http://code.google.com/p/rtm-ros-robotics/

エージェントシステム

- 認識行動プログラミング -

2011/06/01

岡田　慧

http://code.google.com/p/rtm-ros-robotics/

準備(1)

- ソースツリーの更新

```
$ rosinstall ~/prog/rtm-ros-robotics /opt/ros/diamondback http://rtm-ros-robotics.googlecode.com/svn/trunk/agentsystem_ros_tutorials/rtm-ros-robotics.rosinstall
$ rosrun roseus generate-all-msg-srv.sh
```

- ROSの起動に必要な環境変数は~/prog/rtm-ros-robotics/setup.bashにかかれる. したがって, ~/.bashrcの最後に以下の2行を追加する.

```
source ~/prog/rtm-ros-robotics/setup.bash
export PATH=$PATH:`rospack find roseus`/bin
```

- 講義の理解に必要なパッケージをコンパイルする

```
$ rosmake roseus_tutorials --rosdep-install --rosdep-yes
```

- ターミナルを2つ立ち上げ, それぞれ以下を立ち上げる. カメラ画像が表示できれば成功

```
$ roslaunch roseus_tutorials usb-camera.launch
$ roslaunch roseus_tutorials image-view.launch
```

http://code.google.com/p/rtm-ros-robotics/

準備(2)

- 顔画像認識プログラムの起動

```
$ roslaunch roseus_tutorials face-detector-mono.launch
```

- 立ち上げたノードや出力しているトピックの調べ方

```
$ rosnode list
$ rosnode info /face_detector_mono
$ rostopic echo /face_detector_mono/faces
```

- クライアントプログラムの例

```
$ rosrun roseus_tutorials face-detector-mono.l
```

- データを受け取りコールバック関数が呼ばれている.

```
$ (ros::subscribe "face_detector_mono/faces"
   face_detector_mono::RectArray #'facedetect-cb)
```

http://code.google.com/p/rtm-ros-robotics/

facedetect-cb関数

```
(defun facedetect-cb (faces)
  (let ((mrk (instance image_view2::ImageMarker2 :init)) ret)
    (warning-message 1 "facedetect callback~%")
    (dolist (f (send faces :rects))
      (format t "cx:~3d, cy:~3d~%" (send f :x) (send f :y)))

    (when (send faces :rects)
      (let* ((f (car (send faces :rects)))
             (cx (send f :x)) (cy (send f :y))
             (w  (send f :width))  (w/2 (/ w 2))
             (h  (send f :height)) (h/2 (/ h 2)))
        (format t "cx:~3d, cy:~3d, w:~3d, h:~3d~%" cx cy w h)

        (send mrk :type image_view2::ImageMarker2::*POLYGON*)
        (send mrk :points (list
                           (instance geometry_msgs::Point :init
                                     :x (- cx w/2) :y (- cy h/2))
                           (instance geometry_msgs::Point :init
                                     :x (- cx w/2) :y (+ cy h/2))
                           (instance geometry_msgs::Point :init
                                     :x (+ cx w/2) :y (+ cy h/2))
                           (instance geometry_msgs::Point :init
                                     :x (+ cx w/2) :y (- cy h/2))))
        (ros::publish "image_marker" mrk)))))
```

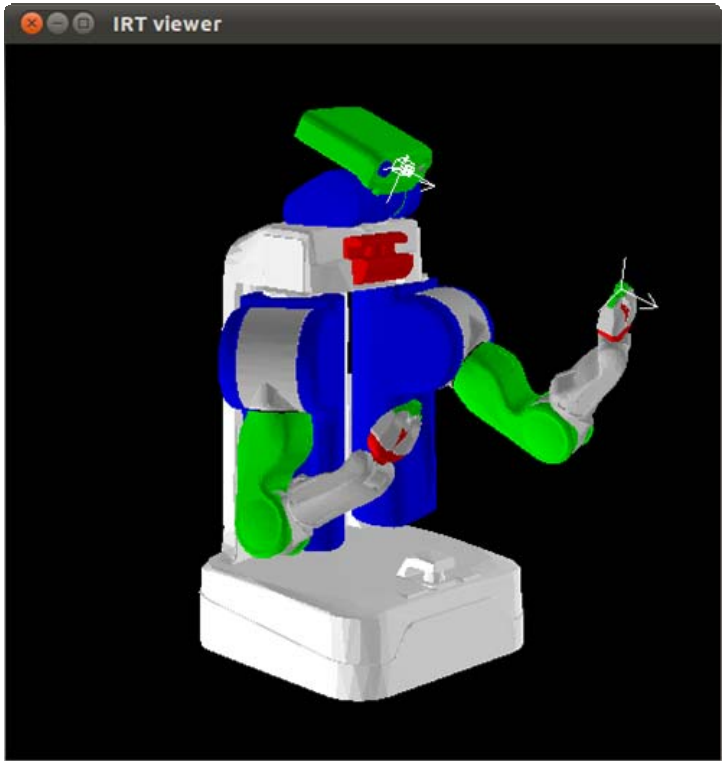
- プログラムを理解するのに便利なツール

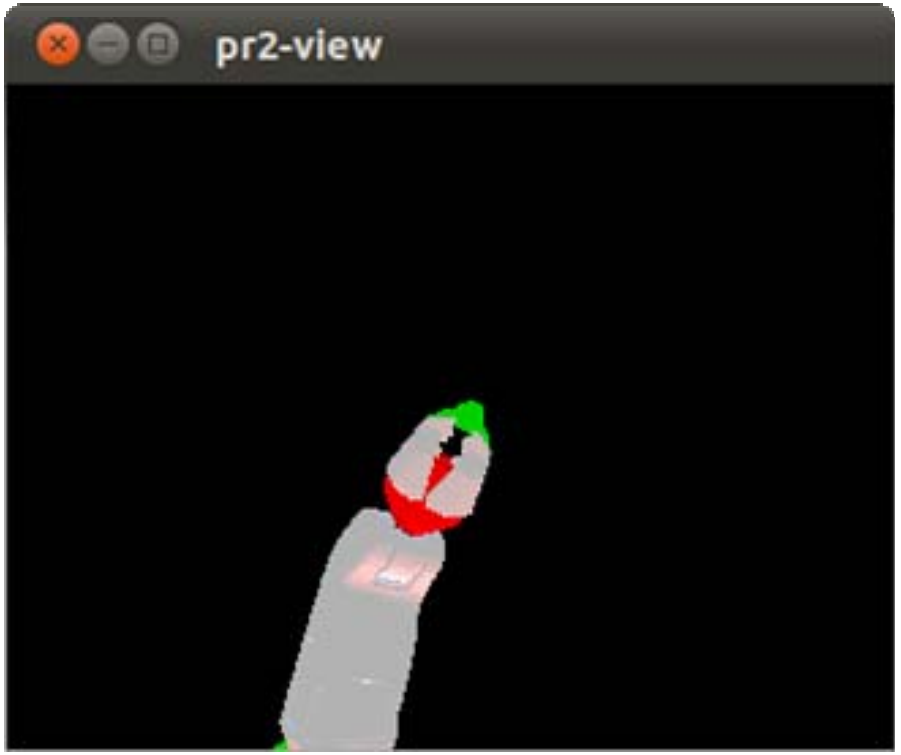
$ rosmsg show face_detector_mono/RectArray

http://code.google.com/p/rtm-ros-robotics/

認識行動プログラム例

- ２次元画像座標系を用いるもの
- ３次元空間座標系を用いるもの





http://code.google.com/p/rtm-ros-robotics/

二次元画像座標を用いた例

- サンプルプログラムの起動

```
$ rosrun roseus_tutorials vision-action-example1.l
```

- facedetect-cb関数の中身

```
(if (> cx 320)
      (send *pr2* :head :neck-y :joint-angle 30)
  (send *pr2* :head :neck-y :joint-angle -30))
(send *ri* :angle-vector (send *pr2* :angle-vector) 1000)
(send *pr2* :head :neck-p :joint-angle 30)
(send *ri* :angle-vector (send *pr2* :angle-vector) 1000)
(send *pr2* :head :neck-p :joint-angle 0)
(send *pr2* :head :neck-y :joint-angle 0)
(send *ri* :angle-vector (send *pr2* :angle-vector) 1000)
```

http://code.google.com/p/rtm-ros-robotics/

三次次元画像座標を用いた例(1)

- サンプルプログラムの起動

```
$ rosrun roseus_tutorials vision-action-example2.l
```

- facedetect-cb関数の中では変数camにカメラのモデルをセットし，(send cam :ray cx cy)とすることで，このカメラの画像座標上の点が絶対座標空間ではどの方向にあるか，という情報を得る

```
;; calc 3d ray from camera model and image coordinates
(setq cam (send *pr2* :camera "wide_stereo/right"))
(setq p (send cam :worldpos))
(setq v (send cam :ray cx cy))
(format t "ray = ~A~%" v)
```

- これとカメラの現在の位置pから，カメラの1000mm先に対象物体が存在するとした場合の世界座標情報を計算でき，(send *pr2: :head :look-at [pos])メソッドを用いて，対象物を見る，という動作を生成する.

```
;; look-at the end of ray
(send *pr2* :head :look-at (v+ p (scale 1000 v)))
(send *ri* :angle-vector (send *pr2* :angle-vector) 1000)
(send *viewer* :viewsurface :3d-line p (v+ p (scale 1000 v)))
(send *viewer* :viewsurface :flush)
```

http://code.google.com/p/rtm-ros-robotics/

三次次元画像座標を用いた例(2)

- サンプルプログラムの起動

```
$ roslaunch roseus_tutorials checkerboard-pose.launch
$ rosrun roseus_tutorials vision-action-example3.l
```

- checkerboard-cb関数の中でcheckerboard-poseから出力されるposeメッセージを座標系オブジェクトに変換している.

```
(defun checkerboard-cb (pose)
  (let ((mrk (instance image_view2::ImageMarker2 :init))
        cam target-local target-world)
    (setq target-local (ros::tf-pose->coords (send pose :pose)))
```

- 次はデバッグ用表示用のコードになる. ここでimage_view2に結果が描かれる.

```
;; for display
(send mrk :type image_view2::ImageMarker2::*FRAMES*)
(send mrk :frames (list "/checkerboard_pose_frame"))
(send *tfb* :send-transform
      target-local (send pose :header :frame_id)
   "/checkerboard_pose_frame")
(ros::ros-info "~A" target-local)
(ros::publish "image_marker" mrk)
```

http://code.google.com/p/rtm-ros-robotics/

三次次元画像座標を用いた例(2)．．．続き

- target-localはカメラ座標系からみた対象の座標なので，以下の様にして世界座標系に変換する.

```
(setq target-world (send target-local :transform (send
   cam :worldcoords) :parent))
```

- 最後のこの場所に手を伸ばすような動作を生成している．:rotation-axis のオプションは，手先の姿勢の拘束を表している．nilは，拘束無しを指定しており，その結果手先の位置のみの3自由度を目標とした逆運動学を解いている

```
(send *pr2* :rarm :inverse-kinematics
       target-world :rotation-axis nil
       :stop 3 :revert-if-error nil :warnp nil)
(send *ri* :angle-vector (send *pr2* :angle-vector) 1000)
(send (send target-world :copy-worldcoords) :draw-on :flush t
   :size 200)
```

http://code.google.com/p/rtm-ros-robotics/

宿題

- １：vision-action-example{1,2,3}.lを実行せよ
- ２：顔認識，チェッカボード認識以外の画像処理プログラムを用いたロボットの行動プログラム例を作成せよ.
- ３：PR2ロボット以外での認識行動プログラム例を作成せよ.
- 提出は，プログラム実行中の画像，あるいは動画をメール貼付，wikiに貼り付け，宿題サイトに貼り付けのいずれかでよい.
- 締め切り：次回講義開始時まで.