Exp_assignment2

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 robot_to_ball.image_feature Class Reference

Public Member Functions

- def __init__ (self)
- def callback (self, ros_data)

Public Attributes

- **image_pub**
- **vel_pub**
- **subscriber**

4.1.1 Detailed Description

Definition at line 27 of file robot_to_ball.py.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def robot_to_ball.image_feature.__init__ (
              self )
```

Initialize ros publisher, ros subscriber

Definition at line 29 of file robot_to_ball.py.

```
29     def __init__(self):
30         '''Initialize ros publisher, ros subscriber'''
31         rospy.init_node('image_feature', anonymous=True)
32      # topic where we publish
33         self.image_pub = rospy.Publisher("/output/image_raw/compressed",
34                                          CompressedImage, queue_size=1)
35         self.vel_pub = rospy.Publisher("/robot/cmd_vel",
36                                        Twist, queue_size=1)
37 
38         # subscribed Topic
39         self.subscriber = rospy.Subscriber("/robot/camera1/image_raw/compressed",
40                                            CompressedImage, self.callback,  queue_size=1)
41 
42 
```

4.1.3 Member Function Documentation

4.1.3.1 callback()

```
def robot_to_ball.image_feature.callback (
               self,
               ros_data )

Callback function of subscribed topic.
Here images get converted and features detected
```

Definition at line 43 of file robot_to_ball.py.

```
43     def callback(self, ros_data):
44         '''Callback function of subscribed topic.
45         Here images get converted and features detected'''
46         if VERBOSE:
47             print ('received image of type: "%s"' % ros_data.format)
48 
49 
50         np_arr = np.fromstring(ros_data.data, np.uint8)
51         image_np = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)  # OpenCV >= 3.0:
52 
53         greenLower = (50, 50, 20)
54         greenUpper = (70, 255, 255)
55 
56         blurred = cv2.GaussianBlur(image_np, (11, 11), 0)
57         hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
58         mask = cv2.inRange(hsv, greenLower, greenUpper)
59         mask = cv2.erode(mask, None, iterations=2)
60         mask = cv2.dilate(mask, None, iterations=2)
61         #cv2.imshow('mask', mask)
62         cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
63                                 cv2.CHAIN_APPROX_SIMPLE)
64         cnts = imutils.grab_contours(cnts)
65         center = None
66         # only proceed if at least one contour was found
67         if len(cnts) > 0:
68             # find the largest contour in the mask, then use
69             # it to compute the minimum enclosing circle and
70             # centroid
71             c = max(cnts, key=cv2.contourArea)
72             ((x, y), radius) = cv2.minEnclosingCircle(c)
73             M = cv2.moments(c)
74             center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
75 
76             # only proceed if the radius meets a minimum size
77             if radius > 10:
78                 # draw the circle and centroid on the frame,
79                 # then update the list of tracked points
80                 cv2.circle(image_np, (int(x), int(y)), int(radius),
81                            (0, 255, 255), 2)
82                 cv2.circle(image_np, center, 5, (0, 0, 255), -1)
83                 vel = Twist()
84                 vel.angular.z = 0.002*(center[0]-400)
85                 vel.linear.x = -0.01*(radius-100)
86                 self.vel_pub.publish(vel)
87 
88         else:
89             vel = Twist()
90             vel.angular.z = 0.5
91             self.vel_pub.publish(vel)
92 
93         # update the points queue
94         # pts.appendleft(center)
95         cv2.imshow('window', image_np)
96         cv2.waitKey(2)
97 
98         # self.subscriber.unregister()
99 
100
```

The documentation for this class was generated from the following file:

- /home/sara/catkin_ws/src/exp_assignment2/scripts/robot_to_ball.py

4.2 state_machine.image_feature Class Reference

Istance called in play state to track the ball.

Public Member Functions

- def __init__ (self)

  initialization
- def callback (self, ros_data)

  callback to subscribe to camera1 when the state is Play: when the ball is found, a parameter (camera_rotate) becomes 1 and the robot starts to rotate the head, otherwise, a parameter becomes 0 and the robot search around for a bit time.

Public Attributes

- image_pub

  publisher
- **vel_pub**
- subscriber

  subscriber

4.2.1 Detailed Description

Istance called in play state to track the ball.

Definition at line 166 of file state_machine.py.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def state_machine.image_feature.__init__ (
              self )
```

initialization

```
Initialize ros publisher, ros subscriber
```

Definition at line 168 of file state_machine.py.

```
168     def __init__(self):
169         '''Initialize ros publisher, ros subscriber'''
170
171
172         self.image_pub = rospy.Publisher("/output/image_raw/compressed",
173                                          CompressedImage, queue_size=1)
174
175         self.vel_pub = rospy.Publisher("/robot/cmd_vel",
176                                        Twist, queue_size=1)
177
178
179
180         self.subscriber = rospy.Subscriber("/robot/camera1/image_raw/compressed",
181                                            CompressedImage, self.callback,  queue_size=1)
182
```

4.2.3 Member Function Documentation

4.2.3.1 callback()

```
def state_machine.image_feature.callback (
              self,
              ros_data )
```

callback to subscribe to camera1 when the state is Play: when the ball is found, a parameter (camera_rotate) becomes 1 and the robot starts to rotate the head, otherwise, a parameter becomes 0 and the robot search around for a bit time.

Definition at line 184 of file state_machine.py.

```
184     def callback(self, ros_data):
185         global count
186         global camera_rotate
187         count = rospy.get_param('count')
188 
189         while count == 360:
190             time.sleep(1)
191 
192 
193 
194 
195         if VERBOSE:
196             print ('received image of type: "%s"' % ros_data.format)
197 
198 
199         np_arr = np.fromstring(ros_data.data, np.uint8)
200         image_np = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
201 
202         greenLower = (50, 50, 20)
203         greenUpper = (70, 255, 255)
204 
205         blurred = cv2.GaussianBlur(image_np, (11, 11), 0)
206         hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
207         mask = cv2.inRange(hsv, greenLower, greenUpper)
208         mask = cv2.erode(mask, None, iterations=2)
209         mask = cv2.dilate(mask, None, iterations=2)
210 
211         cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
212                                 cv2.CHAIN_APPROX_SIMPLE)
213         cnts = imutils.grab_contours(cnts)
214         center = None
215 
216         if len(cnts) > 0:
217 
218             # find the largest contour in the mask, then use
219             # it to compute the minimum enclosing circle and
220             # centroid
221 
222 
223 
224 
225             c = max(cnts, key=cv2.contourArea)
226             ((x, y), radius) = cv2.minEnclosingCircle(c)
227             M = cv2.moments(c)
228             center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
229 
230             # only proceed if the radius meets a minimum size
231             if radius > 10:
232 
233                 # draw the circle and centroid on the frame,
234                 # then update the list of tracked points
235                 cv2.circle(image_np, (int(x), int(y)), int(radius),
236                            (0, 255, 255), 2)
237                 cv2.circle(image_np, center, 5, (0, 0, 255), -1)
238                 vel = Twist()
239 
240                 vel.angular.z = 0.002*(center[0]-400)
241                 vel.linear.x = -0.01*(radius-100)
242 
```

```
243                 self.vel_pub.publish(vel)
244
245                 if (vel.linear.x < 0.01) & (vel.angular.z < 0.01):
246                     vel.angular.z = 0
247                     vel.linear.x = 0
248
249                     self.vel_pub.publish(vel)
250                     rospy.set_param('camera_rotate', 1)
251
252
253
254         else:
255
256             vel = Twist()
257             vel.angular.z = 0.5
258             self.vel_pub.publish(vel)
259             count = count + 1
260
261             rospy.set_param('count', count)
262
263
264             if rospy.get_param('count') == 259:
265
266                 vel.angular.z = 0.0
267                 self.vel_pub.publish(vel)
268                 self.subscriber.unregister()
269
270
271
272
273
274
275         cv2.imshow('window', image_np)
276         cv2.waitKey(2)
277
278
279
280
```

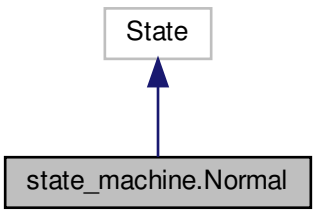
The documentation for this class was generated from the following file:

- /home/sara/catkin_ws/src/exp_assignment2/scripts/state_machine.py

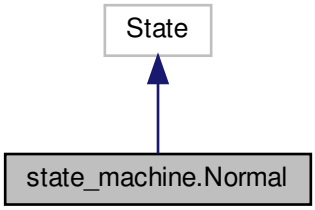
4.3 state_machine.Normal Class Reference

Normal state definition.

Inheritance diagram for state_machine.Normal:



Collaboration diagram for state_machine.Normal:



Public Member Functions

- def __init__ (self)
  inizialization
- def execute (self, userdata)
  execution

Public Attributes

- command
  2 outcomes defined

4.3.1 Detailed Description

Normal state definition.

Definition at line 282 of file state_machine.py.

4.3.2 Member Data Documentation

4.3.2.1 command

`state_machine.Normal.command`

2 outcomes defined

switch in sleep state

command is sleep

read command (choice random between move random or go to sleep)

command is searchball

Definition at line 288 of file state_machine.py.

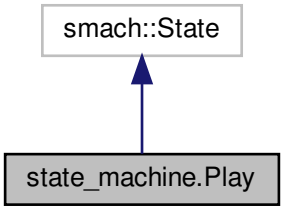
The documentation for this class was generated from the following file:

- /home/sara/catkin_ws/src/exp_assignment2/scripts/state_machine.py

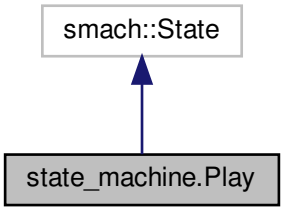
4.4 state_machine.Play Class Reference

Play state definition.

Inheritance diagram for state_machine.Play:



Collaboration diagram for state_machine.Play:



Public Member Functions

- def __init__ (self)

  initialization
- def execute (self, userdata)

  Execution.

Public Attributes

- angle_pub

  1 outcome defined : Normal

4.4.1 Detailed Description

Play state definition.

Definition at line 359 of file state_machine.py.

4.4.2 Member Data Documentation

4.4.2.1 angle_pub

`state_machine.Play.angle_pub`

1 outcome defined : Normal

publisher for move the head

Definition at line 365 of file state_machine.py.

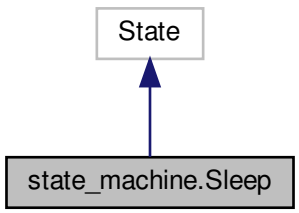
The documentation for this class was generated from the following file:

- /home/sara/catkin_ws/src/exp_assignment2/scripts/state_machine.py

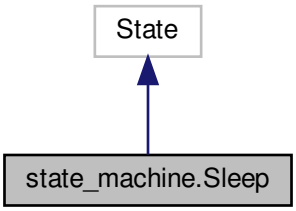
4.5 state_machine.Sleep Class Reference

Sleep State definition.

Inheritance diagram for state_machine.Sleep:



Collaboration diagram for state_machine.Sleep:



Public Member Functions

- def __init__ (self)
  Initialization.
- def execute (self, userdata)
  Execution.

4.5.1 Detailed Description

Sleep State definition.

Definition at line 340 of file state_machine.py.

The documentation for this class was generated from the following file:

- /home/sara/catkin_ws/src/exp_assignment2/scripts/state_machine.py

Chapter 5

File Documentation

5.1 /home/sara/catkin_ws/src/exp_assignment2/scripts/go_to_point_ball.py File Reference

This node is actionlib server that permits to move the ball.

Functions

- def go_to_point_ball.clbk_odom (msg)
  Define callback.
- def **go_to_point_ball.change_state** (state)
- def go_to_point_ball.go_straight_ahead (des_pos)
  define function to go straight
- def go_to_point_ball.done ()
  define function to stop when the goal is reached
- def go_to_point_ball.planning (goal)
  define planning funcion
- def go_to_point_ball.main ()
  Main function.

Variables

- go_to_point_ball.position_ = Point()
  ball state variables
- **go_to_point_ball.pose_** = Pose()
- int **go_to_point_ball.yaw_** = 0
- int go_to_point_ball.state_ = 0
  machine state
- go_to_point_ball.desired_position_ = Point()
  goal
- int go_to_point_ball.yaw_precision_ = math.pi / 9
  parameters
- int **go_to_point_ball.yaw_precision_2_** = math.pi / 90
- float **go_to_point_ball.dist_precision_** = 0.1

- float **go_to_point_ball.kp_a** = 3.0
- float **go_to_point_ball.kp_d** = 0.5
- float **go_to_point_ball.ub_a** = 0.6
- float **go_to_point_ball.lb_a** = -0.5
- float **go_to_point_ball.ub_d** = 2.0
- float **go_to_point_ball.z_back** = 0.25
- go_to_point_ball.pub = None

  publisher
- **go_to_point_ball.pubz** = None
- go_to_point_ball.act_s = None

  action_server

5.1.1 Detailed Description

This node is actionlib server that permits to move the ball.

5.2 /home/sara/catkin_ws/src/exp_assignment2/scripts/go_to_point_robot.py File Reference

This node is actionlib server that permits to move the robot.

Functions

- def go_to_point_robot.clbk_odom (msg)

  Callback function.
- def **go_to_point_robot.change_state** (state)
- def go_to_point_robot.normalize_angle (angle)

  function to compute the norm
- def **go_to_point_robot.fix_yaw** (des_pos)
- def go_to_point_robot.go_straight_ahead (des_pos)

  function to go straight
- def go_to_point_robot.done ()

  function to stop the robot when the goal is achieved
- def go_to_point_robot.planning (goal)

  define planning funcion
- def go_to_point_robot.main ()

  Main function.

Variables

- go_to_point_robot.position_ = Point()
  robot state variables
- **go_to_point_robot.pose_** = Pose()
- int **go_to_point_robot.yaw_** = 0
- int go_to_point_robot.state_ = 0
  machine state
- go_to_point_robot.desired_position_ = Point()
  goal
- **go_to_point_robot.z**
- int go_to_point_robot.yaw_precision_ = math.pi / 9
  parameters
- int **go_to_point_robot.yaw_precision_2_** = math.pi / 90
- float **go_to_point_robot.dist_precision_** = 0.1
- float **go_to_point_robot.kp_a** = -3.0
- float **go_to_point_robot.kp_d** = 0.2
- float **go_to_point_robot.ub_a** = 0.6
- float **go_to_point_robot.lb_a** = -0.5
- float **go_to_point_robot.ub_d** = 0.6
- float **go_to_point_robot.z_back** = 0.25
- go_to_point_robot.pub = None
  publisher
- **go_to_point_robot.pubz** = None
- go_to_point_robot.act_s = None
  action_server

5.2.1 Detailed Description

This node is actionlib server that permits to move the robot.

5.3 /home/sara/catkin_ws/src/exp_assignment2/scripts/move_ball_around.py File Reference

This node permits to the ball to move around, wait for a while and disappear.

Functions

- def move_ball_around.move_ball (target)
  client of actionlib server for moving the ball
- def move_ball_around.main ()
  Main function.

5.3.1 Detailed Description

This node permits to the ball to move around, wait for a while and disappear.

5.4 /home/sara/catkin_ws/src/exp_assignment2/scripts/state_machine.py File Reference

This node implements a state machine which permits to move around and to search for a ball, go to sleep, and play with the ball when the last one is found.

Classes

- class state_machine.image_feature

  Istance called in play state to track the ball.
- class state_machine.Normal

  Normal state definition.
- class state_machine.Sleep

  Sleep State definition.
- class state_machine.Play

  Play state definition.

Functions

- def state_machine.user_action ()

  User Action function.
- def state_machine.move_dog (target)

  client of actionlib server for moving the dog robot
- def state_machine.Normal_clbk (ros_data)

  Callback for Normal State for subscribe to a Camera1 : it checks if the ball is on the arena, if it is, a parameter (DetectedBall) becomes 1 and the robot goes to play, otherwise the parameter becomes 0 and the robot moves randomly.
- def state_machine.main ()

  Main Function definition.

Variables

- bool **state_machine.VERBOSE** = False
- **state_machine.vel** = Twist()
- **state_machine.angle_camera** = Float64()
- state_machine.image_pub

  Publisher.
- **state_machine.vel_pub** = rospy.Publisher("/robot/cmd_vel", Twist, queue_size=1)

5.4.1 Detailed Description

This node implements a state machine which permits to move around and to search for a ball, go to sleep, and play with the ball when the last one is found.

5.4.2 Function Documentation

5.4.2.1 Normal_clbk()

```
def state_machine.Normal_clbk (
              ros_data )
```

Callback for Normal State for subscribe to a Camera1 : it checks if the ball is on the arena, if it is, a parameter (DetectedBall) becomes 1 and the robot goes to play, otherwise the parameter becomes 0 and the robot moves randomly.

Definition at line 97 of file state_machine.py.

```
97 def Normal_clbk(ros_data):
98
99     global count2
100     global DetectedBall
101     global SearchBallSub
102
103     while rospy.get_param('count2') == 360:
104         time.sleep(1)
105
106
107
108     np_arr = np.fromstring(ros_data.data, np.uint8)
109     image_np = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)  # OpenCV >= 3.0:
110
111     greenLower = (50, 50, 20)
112     greenUpper = (70, 255, 255)
113
114     blurred = cv2.GaussianBlur(image_np, (11, 11), 0)
115     hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
116     mask = cv2.inRange(hsv, greenLower, greenUpper)
117     mask = cv2.erode(mask, None, iterations=2)
118     mask = cv2.dilate(mask, None, iterations=2)
119
120
121     cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
122                             cv2.CHAIN_APPROX_SIMPLE)
123     cnts = imutils.grab_contours(cnts)
124     center = None
125
126
127
128
129
130     if len(cnts) > 0:
131         c = max(cnts, key=cv2.contourArea)
132         ((x, y), radius) = cv2.minEnclosingCircle(c)
133         M = cv2.moments(c)
134         center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
135
136
137         if radius > 10:
138
139
140             cv2.circle(image_np, (int(x), int(y)), int(radius),
141                        (0, 255, 255), 2)
142             cv2.circle(image_np, center, 5, (0, 0, 255), -1)
143
144
145             rospy.set_param('DetectedBall',1)
146
147
148     else:
149
150         vel_ = Twist()
151
152         vel_.angular.z = 0.5
153         vel_pub.publish(vel_)
154         count2 = rospy.get_param('count2')
155         count2 = count2 + 1
156         rospy.set_param('count2', count2)
157
158
159
160     cv2.imshow('window', image_np)
161     cv2.waitKey(2)
162
163
164
```

5.4.3 Variable Documentation

5.4.3.1 image_pub

state_machine.image_pub

Initial value:

```
1 =  rospy.Publisher("/output/image_raw/compressed",
2                                       CompressedImage, queue_size=1)
```

Publisher.

Definition at line 90 of file state_machine.py.

Index