Exp_Lab_Assignments

Generated by Doxygen 1.8.11

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 state_manager.coordinates_from_picture Class Reference

Simulates a camera frame and the information it contains.

Public Member Functions

- def **__init__** (self, name)
- def **add_data** (self, img_person_posx, img_person_posy, img_gesture_posx, img_gesture_posy)

Public Attributes

- **person_posx**
- **person_posy**
- **gesture_posx**
- **gesture_posy**

4.1.1 Detailed Description

Simulates a camera frame and the information it contains.

Definition at line 22 of file state_manager.py.

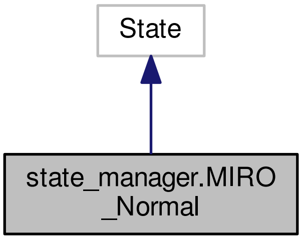
The documentation for this class was generated from the following file:

- src/state_manager.py

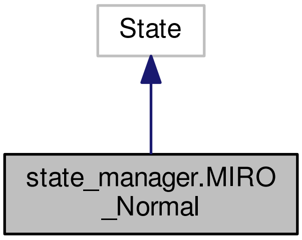
4.2 state_manager.MIRO_Normal Class Reference

Normal state of the smach machine.

Inheritance diagram for state_manager.MIRO_Normal:



Collaboration diagram for state_manager.MIRO_Normal:



Public Member Functions

- def __init__ (self)

  Init function for smach machine normal state.
- def execute (self, userdata)

  Smach machine state normal actions: Listens to user: if user says "Play" or "Hey buddy" it outputs command to enter play state.

4.2.1 Detailed Description

Normal state of the smach machine.

Definition at line 106 of file state_manager.py.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 def state_manager.MIRO_Normal.__init__ (*self*)

Init function for smach machine normal state.

Definition at line 109 of file state_manager.py.

```
109     def __init__(self):
110 
111         smach.State.__init__(self,
112                              outcomes=[’sleep_command’, ’play_command’])
113 
```

4.2.3 Member Function Documentation

4.2.3.1 def state_manager.MIRO_Normal.execute (*self, userdata*)

Smach machine state normal actions: Listens to user: if user says "Play" or "Hey buddy" it outputs command to enter play state.

If user says nothing, it goes to random positions for a while (n loops) then outputs command to enter sleep state.

Returns

c: command to switch between states.

Definition at line 118 of file state_manager.py.

```
118     def execute(self, userdata):
119 
120         # Set state parameter
121         rospy.set_param(’state’, ’NORMAL’)
122 
123         for i in range(0, LOOPS):
124 
125             # Checks if user is speaking
126             user_command = user_says(0)
127 
128             # If user is calling MIRO, enter play state
129             if user_command == ’hey buddy’ or user_command == ’play’:
130                 c = ’play_command’
131                 return c
132 
133             # Else wander around
134             else:
135                 # Wait to be ready
136                 while rospy.get_param(’arrived’) == 0:
137                     time.sleep(1)
138                 rospy.set_param(’arrived’, 0)
139 
140                 normal_command = ’go_rand’
141 
142                 # Publish normal command
143                 pub.publish(normal_command)
144                 time.sleep(3)
145 
146             # Randomly decide to sleep, enter sleep state
147             if random.randrange(0, 5) == 1:
148                 c = ’sleep_command’
149                 return c
150 
151         return ’sleep_command’
152 
```

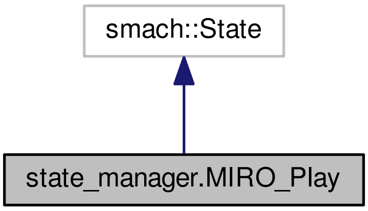
The documentation for this class was generated from the following file:

- src/state_manager.py

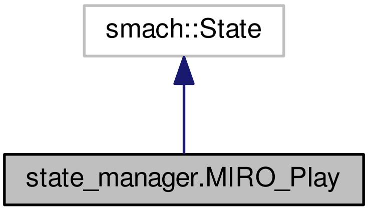
4.3 state_manager.MIRO_Play Class Reference

Play state of the smach machine.

Inheritance diagram for state_manager.MIRO_Play:



Collaboration diagram for state_manager.MIRO_Play:



Public Member Functions

- def __init__ (self)

  Init function for smach machine play state.
- def execute (self, userdata)

  Smach machine state play actions: Looks at user, saves his coordinates as next position, publishes them (goes toward the human).

4.3.1 Detailed Description

Play state of the smach machine.

Definition at line 154 of file state_manager.py.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 def state_manager.MIRO_Play.__init__ (*self*)

Init function for smach machine play state.

Definition at line 157 of file state_manager.py.

```
157     def __init__(self):
158
159         smach.State.__init__(self,
160                              outcomes=[’normal_command’])
161
```

4.3.3 Member Function Documentation

4.3.3.1 def state_manager.MIRO_Play.execute (*self, userdata*)

Smach machine state play actions: Looks at user, saves his coordinates as next position, publishes them (goes toward the human).

It then listens to the user. If user says "go to posx posy", publishes the coordinates (goes to the point). If user says "Hey buddy" or "Play" it waits. If user says nothing, it looks for the user gesture to go somewhere, and publishes the coordinate he receives (goes to the point). This repeates for a while (n loops) then the robot enters normal state again.

Returns

c: command to switch between states.

Definition at line 169 of file state_manager.py.

```
169     def execute(self, userdata):
170
171         # Set state parameter
172         rospy.set_param(’state’, ’PLAY STATE’)
173
174         for i in range(0, LOOPS):
175
176             # Check where user is (assumption:he is there, since he called MIRO)
177             user_camera = user_does()
178             # Save user position
179             user_position = "go to %d %d" % (
180                 user_camera[0], user_camera[1])
181
182             # Wait to be ready
183             while rospy.get_param(’arrived’) == 0:
184                 time.sleep(1)
185             rospy.set_param(’arrived’, 0)
186
187             # Go to user
188             pub.publish(user_position)
189             time.sleep(3)
190
191             # Listen to user
192             user_command = user_says(1)
193
194             # If user says to go somewhere...
195             if ’go’ in user_command and ’to’ in user_command:
196                 check_int = [int(s)
197                              for s in user_command.split() if s.isdigit()]
198                 # ... and he actually gives you two coordinates...
199
200                 if len(check_int) != 2:
201                     rospy.logerr(’Wrong command’)
202                     break
```

```

                # Wait to be ready
                while rospy.get_param('arrived') == 0:
                    time.sleep(1)
                rospy.set_param('arrived', 0)

                # ...Go to position
                pub.publish(user_command)
                time.sleep(3)

            # If user says he wants to play: wait
            elif user_command == 'hey buddy' or user_command == 'play':
                time.sleep(2)

            # If user sayes nothing
            else:
                # Look at user gesture
                user_gesture = user_does()
                user_command = "go to %d %d" % (
                    user_camera.gesture_posx, user_camera.gesture_posy)

                while rospy.get_param('arrived') == 0:
                    time.sleep(1)
                rospy.set_param('arrived', 0)

                # Go to position
                pub.publish(user_command)
                time.sleep(3)

        c = 'normal_command'
        return c

```

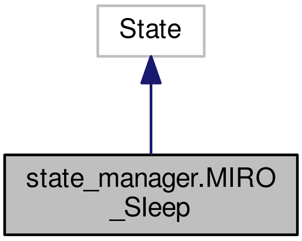
The documentation for this class was generated from the following file:

- src/state_manager.py

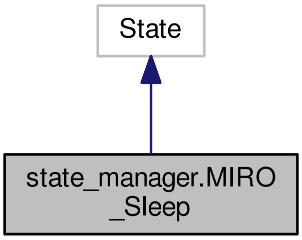
4.4 state_manager.MIRO_Sleep Class Reference

Sleep state of the smach machine.

Inheritance diagram for state_manager.MIRO_Sleep:



Collaboration diagram for state_manager.MIRO_Sleep:



Public Member Functions

- def __init__ (self)

  Init function for smach machine sleep state.
- def execute (self, userdata)

  Smach machine state sleep actions: Publishes "go home" command, waits ("sleeps") and outputs command to enter normal state.

4.4.1 Detailed Description

Sleep state of the smach machine.

Definition at line 73 of file state_manager.py.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 def state_manager.MIRO_Sleep.__init__ (*self*)

Init function for smach machine sleep state.

Definition at line 76 of file state_manager.py.

```
76     def __init__(self):
77 
78         smach.State.__init__(self,
79                              outcomes=['normal_command'])
80 
```

4.4.3 Member Function Documentation

4.4.3.1 def state_manager.MIRO_Sleep.execute (*self*, *userdata*)

Smach machine state sleep actions: Publishes "go home" command, waits ("sleeps") and outputs command to enter normal state.

Returns

c: command to switch between states.

Definition at line 84 of file state_manager.py.

```
84      def execute(self, userdata):
85
86          # Set state parameter
87          rospy.set_param(’state’, ’SLEEP STATE’)
88
89          # Wait to be ready
90          while rospy.get_param(’arrived’) == 0:
91              time.sleep(1)
92          rospy.set_param(’arrived’, 0)
93
94          # Give command home
95          sleep_command = ’go_home’
96
97          # Publish sleep command
98          pub.publish(sleep_command)
99          time.sleep(4)
100
101          # Change state
102          c = ’normal_command’
103          return c
104
```

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

- src/state_manager.py

Chapter 5

File Documentation

5.1 src/geometry_grounding.py File Reference

This node transforms a command into two x,y coordinates.

Functions

- def geometry_grounding.callback (data)

  Callback function for the user command.
- def geometry_grounding.geometry_grounding ()

  Ros node that subscribes to the targcommand topic and publishes on the target_pos topic.

Variables

- **geometry_grounding.pub** = rospy.Publisher('target_pos', Int64MultiArray, queue_size=10)
- **geometry_grounding.pos_to_send** = Int64MultiArray()
- **geometry_grounding.data**

5.1.1 Detailed Description

This node transforms a command into two x,y coordinates.

5.1.2 Function Documentation

5.1.2.1 def geometry_grounding.callback (*data*)

Callback function for the user command.

If the command is a "go to x y" command, it sets the target position as x,y. If the command is a "go home" command, it sets the target postion as home_posx,home_posy. If the command is a "go rand" command, it sets the target position as random coordinates. It then publishes the target position.

Definition at line 24 of file geometry_grounding.py.

```
def callback(data):

    input_string = str(data.data)

    # Save positions in the command, if any
    my_command = [int(s) for s in input_string.split() if s.isdigit()]

    # If command is a "go to" command
    if my_command:
        pos_to_send.data = [my_command[0], my_command[1]]

    # If command is a "go home" command
    elif input_string == "go_home":
        pos_to_send.data = [rospy.get_param(
            'home_posx'), rospy.get_param('home_posy')]

    # If command is a "go rand" command
    elif input_string == "go_rand":
        pos_to_send.data = [random.randrange(10), random.randrange(10)]

    # Publish
    pub.publish(pos_to_send)

```

5.1.2.2 def geometry_grounding.geometry_grounding ( )

Ros node that subscribes to the targcommand topic and publishes on the target_pos topic.

Definition at line 48 of file geometry_grounding.py.

```
def geometry_grounding():

    rospy.init_node('geometry_grounding', anonymous=True)

    rospy.Subscriber("command", String, callback)

    rospy.spin()
    pass


```

5.2 src/printInfo.py File Reference

This node prints informations about target position, reached position, state.

Functions

- def printInfo.printer ()
  Prints the important parameters as loginfo: state, command, robot position.

5.2.1 Detailed Description

This node prints informations about target position, reached position, state.

5.3 src/robot_motion_controller.py File Reference

This node allows to move the robot from the current to the target position.

Functions

- def robot_motion_controller.EuclidianDistance (x_goal, y_goal, x_real, y_real)

  Calculates the euclidean distance between two given points.
- def robot_motion_controller.odom_callback (data)

  Callback function for the robot position.
- def robot_motion_controller.traj_callback (data)

  Callback function for the target position.
- def robot_motion_controller.robot_motion_controller ()

  Ros node that subscribes to the target_pos and odom topic and publishes on the cmd_vel topic.

Variables

- **robot_motion_controller.pub** = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
- **robot_motion_controller.vel** = Twist()
- **robot_motion_controller.x**
- **robot_motion_controller.y**
- **robot_motion_controller.z**
- int **robot_motion_controller.number** = 1
- int **robot_motion_controller.curr_x** = 0
- int **robot_motion_controller.curr_y** = 0

5.3.1 Detailed Description

This node allows to move the robot from the current to the target position.

5.3.2 Function Documentation

5.3.2.1 def robot_motion_controller.odom_callback (*data*)

Callback function for the robot position.

Definition at line 42 of file robot_motion_controller.py.

```
42 def odom_callback(data):
43
44     global curr_x
45     global curr_y
46
47     curr_x = data.pose.pose.position.x
48     curr_y = data.pose.pose.position.y
49
```

5.3.2.2 def robot_motion_controller.robot_motion_controller ( )

Ros node that subscribes to the target_pos and odom topic and publishes on the cmd_vel topic.

Definition at line 89 of file robot_motion_controller.py.

```
89 def robot_motion_controller():
90
91     rospy.init_node(’robot_motion_controlller’, anonymous=True)
92
93     rospy.Subscriber("target_pos", Int64MultiArray, traj_callback)
94
95     rospy.Subscriber(’odom’, Odometry, odom_callback)
96
97     rospy.spin()
98
99     pass
100
101
```

5.3.2.3 def robot_motion_controller.traj_callback (*data*)

Callback function for the target position.

It computes the velocity to send to the cmd_vel topic, by considering an omniwheel robot. when the robot has arrived at desired position, publishes vel=0 and sets the "arrived" and current robot position parameters.

Definition at line 54 of file robot_motion_controller.py.

```
54 def traj_callback(data):
55 
56     global curr_x
57     global curr_y
58     global number
59 
60     target_pos = data.data
61     target_x = target_pos[0]
62     target_y = target_pos[1]
63 
64     while EuclidianDistance(target_x, target_y, curr_x, curr_y) >= 0.001:
65 
66         # omniwheel robot
67         vel.linear.x = (target_x-curr_x)
68         vel.linear.y = (target_y-curr_y)
69 
70         # Publish
71         pub.publish(vel)
72 
73     # omniwheel robot
74     vel.linear.x = 0
75     vel.linear.y = 0
76 
77     # Publish
78     pub.publish(vel)
79 
80     # Set command parameter
81     rospy.set_param(’all’, [target_x, target_y, curr_x, curr_y, number])
82     rospy.set_param(’arrived’, 1)
83 
84     number = number+1
85 
86     time.sleep(2)
87 
```

Index