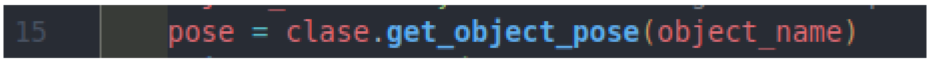
API

Environment Information

- `get_object_list()` - Return the name list of all objects.

  Se puede sacar asi, objetos=clase.object_list
  siempre que sean objetos añadidos a esa lista.
  Siendo añadidos por una funcion add_objects() que en realidad no permite crear objetos, simplemente se ejecuta y crea los objetos ya definidos en ella al instanciar la clase.
  Se crean en RVIZ

- `get_object_pose(object_name)` - Return the pose of the object.
  Solo funciona con los objetos añadidos a traves de la lista en la funcion add_object
  

```
pose = clase.get_object_pose(object_name)
```

- `get_object_info(object_name)` - Return the pose, height, width, length, shape, color of the object in order.

- `get_target_list()` - Return the name list of all targets.
- `get_target_position(target_name)` - Return the position of target where we are going to place the objects.

Convert Pose Message

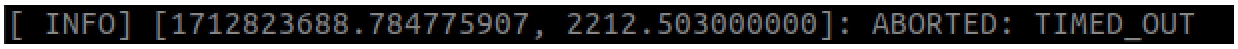
- `pose2msg(roll, pitch, yaw, x, y, z)` - Convert pose to Pose message. The unit of roll, pitch, yaw is radian.
-
- `pose2msg_deg(roll, pitch, yaw, x, y, z)` - Convert pose to Pose message. The unit of roll, pitch, yaw is degree.

- `msg2pose(pose)` - Convert Pose message to pose, return roll, pitch, yaw, x, y, z in order. The unit of roll, pitch, yaw is radian.
-
- `msg2pose_deg(pose)` - Convert Pose message to pose, return roll, pitch, yaw, x, y, z in order. The unit of roll, pitch, yaw is degree.

Basic Robot Movement

- `move_pose_arm(pose_goal)` - Command the robot with Pose message to make its end effector frame move to the desired pose with inverse kinematics.
- `move_joint_arm(joint_0,joint_1,joint_2,joint_3,joint_4,joint_5)` - Command the robot joints to move to desired joints value
- `move_joint_hand(joint_value)` - Command the gripper joint to move to desired joint value.
- `back_to_home()` - Command the robot arm and gripper to move back to the home pose.

Estos funcionan, lo único, que si le das una posición lejana puede ser que tengas el error de



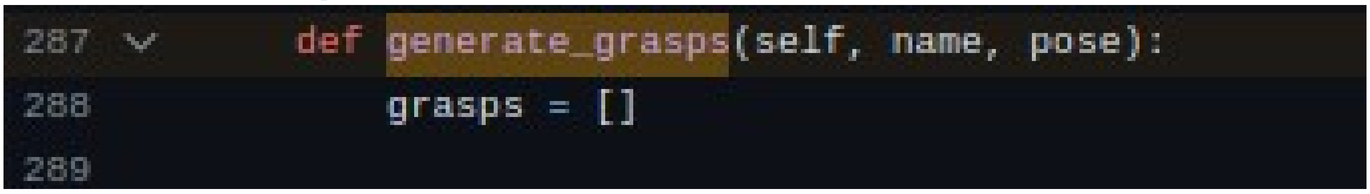
```
[ INFO] [1712823688.784775907, 2212.503000000]: ABORTED: TIMED_OUT
```

Mismo error que yo tenia y por lo que daba puntos intermedios para hacer mi trayectoria

Setup Grasp message

Grasp message contains the informations that the MoveIt pick function requires.

- `generate_grasp(object_name, eef_orientation, position, [width, roll, pitch, yaw, length])` - Returns the specified Grasp message according to related setup.
  Podemos ver que no tiene las mismas variables de entrada



```
    def generate_grasps(self, name, pose):
        grasps = []

```

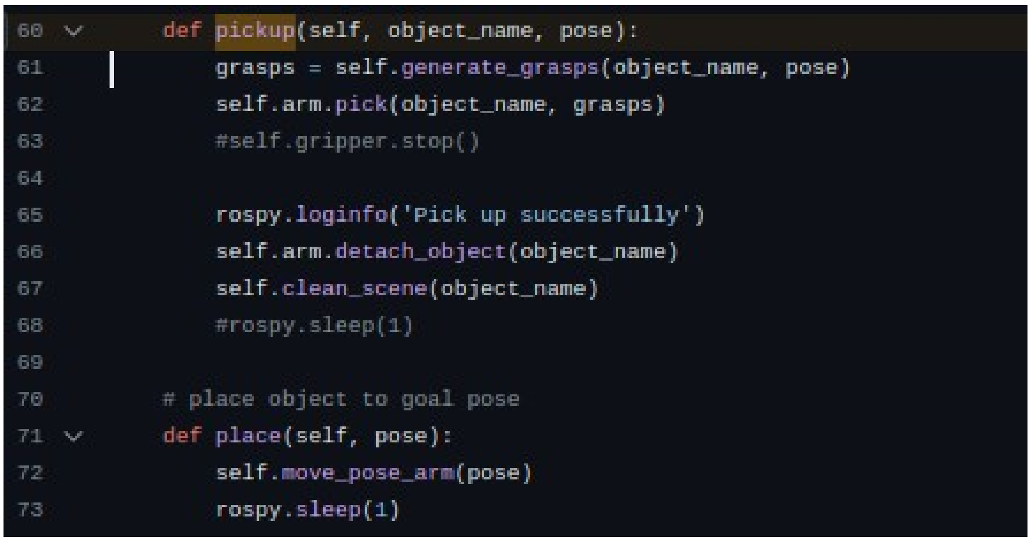
- `eef_orientaion` is to clarify the desired end effector orientation.
  - `horizontal`: grasp the object with a horizontal gripper orientation. (default value: roll = pitch = yaw = 0. `pitch` is setable.)
  - `vertical`: grasp the object with a vertical gripper orientation. (default value: roll = 0, pitch = 90°, yaw = 0. `yaw` is setable.)
  - `user_defined`: grasp the object with a user defined gripper orientation. (default value: roll = pitch = yaw = 0. `roll`,`pitch`,`yaw` are all setable)
- `position` is the position of the end effector when grasping the object
- `width` is the value the gripper joints should move to grasp the object with a range of [0, 0.8]. If you keep the default value 0, the `eef_orientation` is `horizontal` or `vertical` and the default rpy angle values are kept, this width value will be set depend on the object width.
- `roll`,`pitch`,`yaw` are optional parameters.
- `length` is the offset length from the your desired gripper position to robot tool center. When you input grasping pose, you are specifying the pose of the desired gripper position. The default value is 0, which means you are specifying the pose of the tool center of the robot arm when the robot is grasping the object.

The default minimum grasp distance and desired distance are set to be 0.2(m) and 0.1(m). The default approach direction is set to be (0, 0, -0.5). You can keep them or modify them with following functions:

- `set_grasp_distance(min_distance, desired_distance)` - Set the minimum distance and desired distance the gripper translates before and after grasping.
- `set_grasp_direction(x, y, z)` - Set the direction of the gripper approach translation before grasping. Retreat translation distance will set to be the opposite direction of the approach direction.

Pick and Place

Simplemente con comparar las variables de entrada ya podemos ver que son diferente, ademas el pickup de la foto incluso genera el mensaje grasp dentro



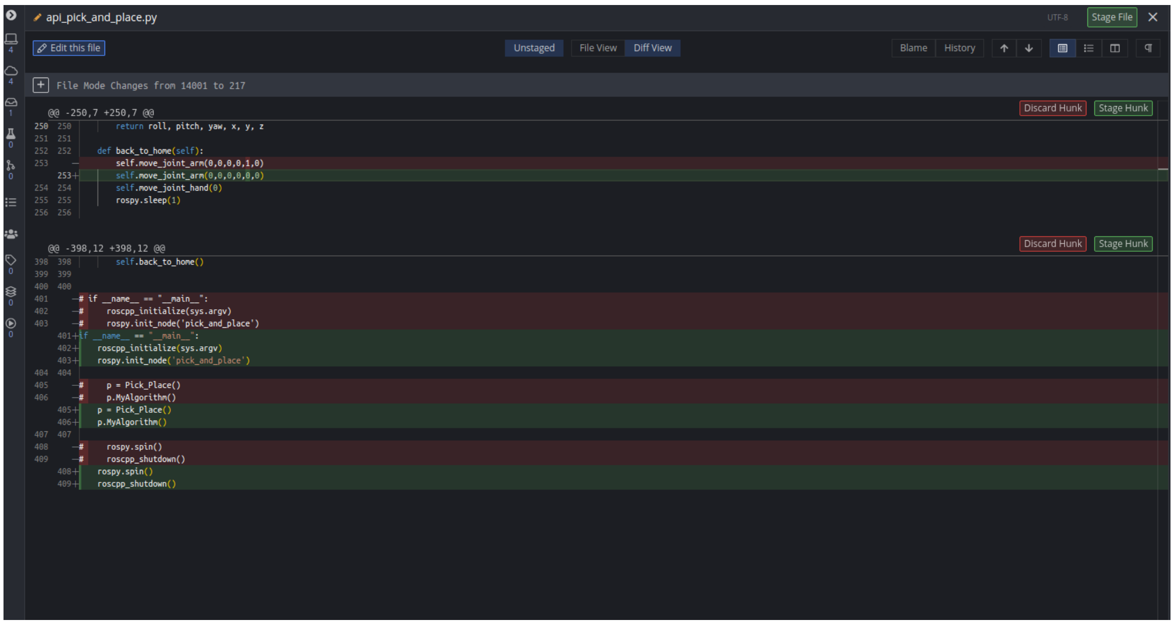
```
    def pickup(self, object_name, pose):
        grasps = self.generate_grasps(object_name, pose)
        self.arm.pick(object_name, grasps)
        #self.gripper.stop()

        rospy.loginfo('Pick up successfully')
        self.arm.detach_object(object_name)
        self.clean_scene(object_name)
        #rospy.sleep(1)

    # place object to goal pose
    def place(self, pose):
        self.move_pose_arm(pose)
        rospy.sleep(1)
```

- `pickup(object_name, grasps)` - Command the industrial robot to pick up the object with a list of genrated Grasp messages.
- `place(eef_orientation, position[, distance, roll, pitch, yaw])` - Command the industrial robot to place the currently holding object to goal_position with desired end effector orientation.
    - `eef_orientaion` is to clarify the desired end effector orientation.
        - `horizontal`: grasp the object with a horizontal gripper orientation. (default value: roll = 0, pitch = 0, yaw = 180°. `pitch` is setable.)
        - `vertical`: grasp the object with a vertical gripper orientation. (default value: roll = 0, pitch = 90°, yaw = 180°. `yaw` is setable.)
        - `user_defined`: grasp the object with a user defined gripper orientation. (default value: roll = 0, pitch = 0, yaw = 180°. `roll`,`pitch`,`yaw` are all setable)
    - `position` is the position of the end effector when placing the object
    - `distance` is the distance the robot arm will move in z axis when placing objects. The default value is 0.1(m)

Robot API

Si volvemos al principio de la explicación ya las hemos visto todas, a excepción de la ultima que tampoco la tenemos

- `HAL.back_to_home()` - Command the robot arm and gripper to move back to the home pose.
- `HAL.pickup()` - to set the linear speed
- `HAL.place()` - to set the angular velocity
- `HAL.move_pose_arm(pose_goal)` - Command the robot with Pose message to make its end effector frame move to the desired pose with inverse kinematics.
- `HAL.move_joint_hand(joint_value)` - Command the gripper joint to move to desired joint value.
- `HAL.generate_grasp(object_name, eef_orientation, position, [width, roll, pitch, yaw, length])` - Returns the specified Grasp message according to related setup.
- `HAL.get_object_pose(object_name)` - Return the pose of the object.
- `HAL.get_target_position(target_name)` - Return the position of target where we are going to place the objects.



```
File Mode Changes from 14001 to 217

@@ -250,7 +250,7 @@
             return roll, pitch, yaw, x, y, z

         def back_to_home(self):
-            self.move_joint_arm(0,0,0,0,1,0)
+            self.move_joint_arm(0,0,0,0,0,0)
             self.move_joint_hand(0)
             rospy.sleep(1)


@@ -398,12 +398,12 @@
             self.back_to_home()


-# if __name__ == "__main__":
-#     roscpp_initialize(sys.argv)
-#     rospy.init_node('pick_and_place')
+if __name__ == "__main__":
+    roscpp_initialize(sys.argv)
+    rospy.init_node('pick_and_place')

-#     p = Pick_Place()
-#     p.MyAlgorithm()
+    p = Pick_Place()
+    p.MyAlgorithm()

-#     rospy.spin()
-#     roscpp_shutdown()
+    rospy.spin()
+    roscpp_shutdown()
```

https://github.com/JdeRobot/IndustrialRobots/blob/master/industrial_robots/irb120_robotiq85/irb120_robotiq85_gazebo/src/api_pick_and_place.py

SON iguales