Motion Designator

Contents

Motion designators are similar to action designators, but unlike action designators, motion designators represent atomic low-level motions. Motion designators only take the parameter that they should execute and not a list of possible parameters, like the other designators. Like action designators, motion designators can be performed. Performing a motion designator verifies the parameter and passes the designator to the respective process module.

Since motion designators perform a motion on the robot, we need a robot which we can use. Therefore, we will create a BulletWorld as well as a PR2 robot.

```
from pycram.worlds.bullet_world import BulletWorld
from pycram.world_concepts.world_object import Object
from pycram.datastructures.enums import ObjectType, WorldMode

world = BulletWorld(WorldMode.GUI)
pr2 = Object("pr2", ObjectType.ROBOT, "pr2.urdf")
```

Move

latest

Move is used to let the robot drive to the given target pose. Motion designator are used in the same way as the other designator, first create a description then resolve it to the actual

designator and lastly, perform the resolved designator.

```
from pycram.datastructures.pose import Pose
from pycram.designators.motion_designator import MoveMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    motion_description = MoveMotion(target=Pose([1, 0, 0], [0, 0, 0, 1]))

    motion_description.perform()
```

```
world.reset_world()
```

MoveTCP

MoveTCP is used to move the tool center point (TCP) of the given arm to the target position specified by the parameter. Like any designator we start by creating a description and then resolving and performing it.

```
from pycram.designators.motion_designator import MoveTCPMotion
from pycram.process_module import simulated_robot
from pycram.datastructures.enums import Arms

with simulated_robot:
    motion_description = MoveTCPMotion(target=Pose([0.5, 0.6, 0.6], [0, 0, 0, 1

    motion_description.perform()
```

Looking

Looking motion designator adjusts the robot state such that the cameras point towards the target pose. Although this motion designator takes the target as position and orientation, in reality only the position is used.

```
from pycram.designators.motion_designator import LookingMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    motion_description = LookingMotion(target=Pose([1, 1, 1], [0, 0, 0, 1]))

    motion_description.perform()
```

Move Gripper

Move gripper moves the gripper of an arm to one of two states. The states can be `OPEN` and `CLOSE`, which open and close the gripper respectively.

```
from pycram.designators.motion_designator import MoveGripperMotion
from pycram.process_module import simulated_robot
from pycram.datastructures.enums import Arms, GripperState

with simulated_robot:
    motion_description = MoveGripperMotion(motion=GripperState.OPEN, gripper=Ar

    motion_description.perform()
```

Detecting

This is the motion designator implementation of detecting, if an object with the given object type is in the field of view (FOV) this motion designator will return an object designator describing the object.

Since we need an object that we can detect, we will spawn a milk for this.

```
milk = Object("milk", ObjectType.MILK, "milk.stl", pose=Pose([1.5, 0, 1]))
```

latest

```
from pycram.designators.motion_designator import DetectingMotion, LookingMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    LookingMotion(target=Pose([1.5, 0, 1], [0, 0, 0, 1])).perform()

    motion_description = DetectingMotion(object_type=ObjectType.MILK)

    obj = motion_description.perform()

    print(obj)
```

Move Arm Joints

This motion designator moves one or both arms. Movement targets are a dictionary with joint name as key and target pose as value.

```
from pycram.designators.motion_designator import MoveArmJointsMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    motion_description = MoveArmJointsMotion(right_arm_poses={"r_shoulder_pan_j

    motion_description.perform()
```

World State Detecting

World state detecting is also used to detect objects, however, the object is not required to be in the FOV of the robot. As long as the object is somewhere in the belief state (BulletWorld) a resolved object designator will be returned.

Sine we want to detect something we will spawn an object that we can detect. If you already spawned the milk from the previous example, you can skip this step.

```
milk = Object("milk", ObjectType.MILK, "milk.stl", pose=Pose([-1, 0, 1]))
```

```
from pycram.designators.motion_designator import WorldStateDetectingMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    motion_description = WorldStateDetectingMotion(object_type=ObjectType.MILK)

    obj = motion_description.perform()

    print(obj)
```

Move Joints

Move joints can move any number of joints of the robot, the designator takes two lists as parameter. The first list are the names of all joints that should be moved and the second list are the positions to which the joints should be moved.

```
from pycram.designators.motion_designator import MoveJointsMotion
from pycram.process_module import simulated_robot

with simulated_robot:
    motion_description = MoveJointsMotion(names=["torso_lift_joint", "r_shoulde

    motion_description.perform()
```

The following cell can be used after testing the examples, to close the BulletWorld.

```
world.exit()
```

latest