pycram.designators.location_designator

Print to PDF

Contents

- Classes
- Module Contents

Classes

`Location`	Default location designator which only wraps a pose.
`ObjectRelativeLocation`	Location relative to an object
`CostmapLocation`	Uses Costmaps to create locations for complex constrains
`AccessingLocation`	Location designator which describes poses used for opening drawers
`SemanticCostmapLocation`	Locations over semantic entities, like a table surface

Module Contents

class pycram.designators.location_designator.**Location**(*pose: pycram.datastructures.pose.Pose*, *resolver=None*)

Bases: `pycram.designator.LocationDesignatorDescription`

Default location designator which only wraps a pose.

class **Location**

Bases: `pycram.designator.LocationDesignatorDescription.Location`

Resolved location that represents a specific point in the world which satisfies the constraints of the location designator description.

pose: *pycram.datastructures.pose.Pose*

ground() → `Location`

Default specialized_designators which returns a resolved designator which contains the pose given in init.

Returns:

A resolved designator

latest

class pycram.designators.location_designator.**ObjectRelativeLocation**(*relative_pose: pycram.datastructures.pose.Pose = None, reference_object: pycram.designators.object_designator.ObjectDesignatorDescription = None, resolver=None*)

Bases: pycram.designator.LocationDesignatorDescription

Location relative to an object

class **Location**

Bases: pycram.designator.LocationDesignatorDescription.Location

Resolved location that represents a specific point in the world which satisfies the constraints of the location designator description.

relative_pose: *pycram.datastructures.pose.Pose*

Pose relative to the object

reference_object: *pycram.designators.object_designator.ObjectDesignatorDescription.Object*

Object to which the pose is relative

relative_pose: *pycram.datastructures.pose.Pose*

reference_object: *pycram.designators.object_designator.ObjectDesignatorDescription*

ground() → Location

Default specialized_designators which returns a resolved location for description input. Resolved location is the first result of the iteration of this instance.

Returns:

A resolved location

__iter__() → typing_extensions.Iterable[Location]

Iterates over all possible solutions for a resolved location that is relative to the given object.

Yield:

An instance of ObjectRelativeLocation.Location with the relative pose

class pycram.designators.location_designator.**CostmapLocation**(*target: typing_extensions.Union[pycram.datastructures.pose.Pose, pycram.designators.object_designator.ObjectDesignatorDescription.Object], reachable_for: typing_extensions.Optional[pycram.designators.object_designator.ObjectDesignatorDescription.Object] = None, visible_for: typing_extensions.Optional[pycram.designators.object_designator.ObjectDesignatorDescription.Object] = None, reachable_arm: typing_extensions.Optional[pycram.datastructures.enums.Arms] = No resolver: typing_extensions.Optional[typing_extensions.Callable] = None*)

latest

Bases: pycram.designator.LocationDesignatorDescription

Uses Costmaps to create locations for complex constrains

class **Location**

Bases: pycram.designator.LocationDesignatorDescription.Location

Resolved location that represents a specific point in the world which satisfies the constraints of the location designator description.

reachable_arms: *typing_extensions.List[pycram.datastructures.enums.Arms]*

List of arms with which the pose can be reached, is only used when the 'rechable_for' parameter is used

target: *typing_extensions.Union[pycram.datastructures.pose.Pose, pycram.designators.object_designator.ObjectDesignatorDescription.Object]*

reachable_for: *pycram.designators.object_designator.ObjectDesignatorDescription.Object*

visible_for: *pycram.designators.object_designator.ObjectDesignatorDescription.Object*

reachable_arm: *typing_extensions.Optional[pycram.datastructures.enums.Arms]*

ground() → Location

Default specialized_designators which returns the first result from the iterator of this instance.

Returns:

A resolved location

__iter__()

Generates positions for a given set of constrains from a costmap and returns them. The generation is based of a costmap which itself is the product of merging costmaps, each for a different purpose. In any case an occupancy costmap is used as the base, then according to the given constrains a visibility or gaussian costmap is also merged with this. Once the costmaps are merged, a generator generates pose candidates from the costmap. Each pose candidate is then validated against the constraints given by the designator if all validators pass the pose is considered valid and yielded.

Yield:

An instance of CostmapLocation.Location with a valid position that satisfies the given constraints

class pycram.designators.location_designator.**AccessingLocation**(*handle_desig: pycram.designators.object_designator.ObjectPart.Object, robot_desig: pycram.designators.object_designator.ObjectDesignatorDescription.Object, resolver=None*)

Bases: pycram.designator.LocationDesignatorDescription

Location designator which describes poses used for opening drawers

class **Location**

Bases: pycram.designator.LocationDesignatorDescription.Location

Resolved location that represents a specific point in the world which satisfies the constraints of the location designator description.

arms: *typing_extensions.List[pycram.datastructures.enums.Arms]*

List of arms that can be used to for accessing from this pose

handle: *pycram.designators.object_designator.ObjectPart.Object*

robot: *pycram.designators.object_designator.ObjectDesignatorDescription.Object*

ground() → Location

Default specialized_designators for this location designator, just returns the first element from the iteration

Returns:

A location designator for a pose from which the drawer can be opened

__iter__() → Location

Creates poses from which the robot can open the drawer specified by the ObjectPart designator describing the handle. Poses are validated by checking if the robot can grasp the handle while the drawer is closed and if the handle can be grasped if the drawer is open.

Yield:

A location designator containing the pose and the arms that can be used.

class pycram.designators.location_designator.**SemanticCostmapLocation**(*urdf_link_name*, *part_of*, *for_object=None*, *resolver=None*)

Bases: pycram.designator.LocationDesignatorDescription

Locations over semantic entities, like a table surface

class **Location**

Bases: pycram.designator.LocationDesignatorDescription.Location

Resolved location that represents a specific point in the world which satisfies the constraints of the location designator description.

urdf_link_name: *str*

part_of: *pycram.designators.object_designator.ObjectDesignatorDescription.Object*

for_object: *typing_extensions.Optional[pycram.designators.object_designator.ObjectDesignatorDescription.Object]*

ground() → Location

Default specialized_designators which returns the first element of the iterator of this instance.

Returns:

A resolved location

__iter__()

Creates a costmap on top of a link of an Object and creates positions from it. If there is a specific Object for which the position should be found, a height offset will be calculated which ensures that the bottom of the Object is at the position in the Costmap and not the origin of the Object which is usually in the centre of the Object.

Yield:

An instance of SemanticCostmapLocation.Location with the found valid position of the Costmap.