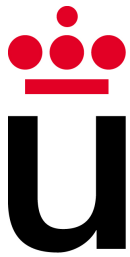
Aplicaciones de robótica de servicio en la plataforma académica Robotics Academy

Lucía Lishan Chen Huang

Grado de Ingeniería de Robótica Software
Escuela de Ingeniería de Fuenlabrada

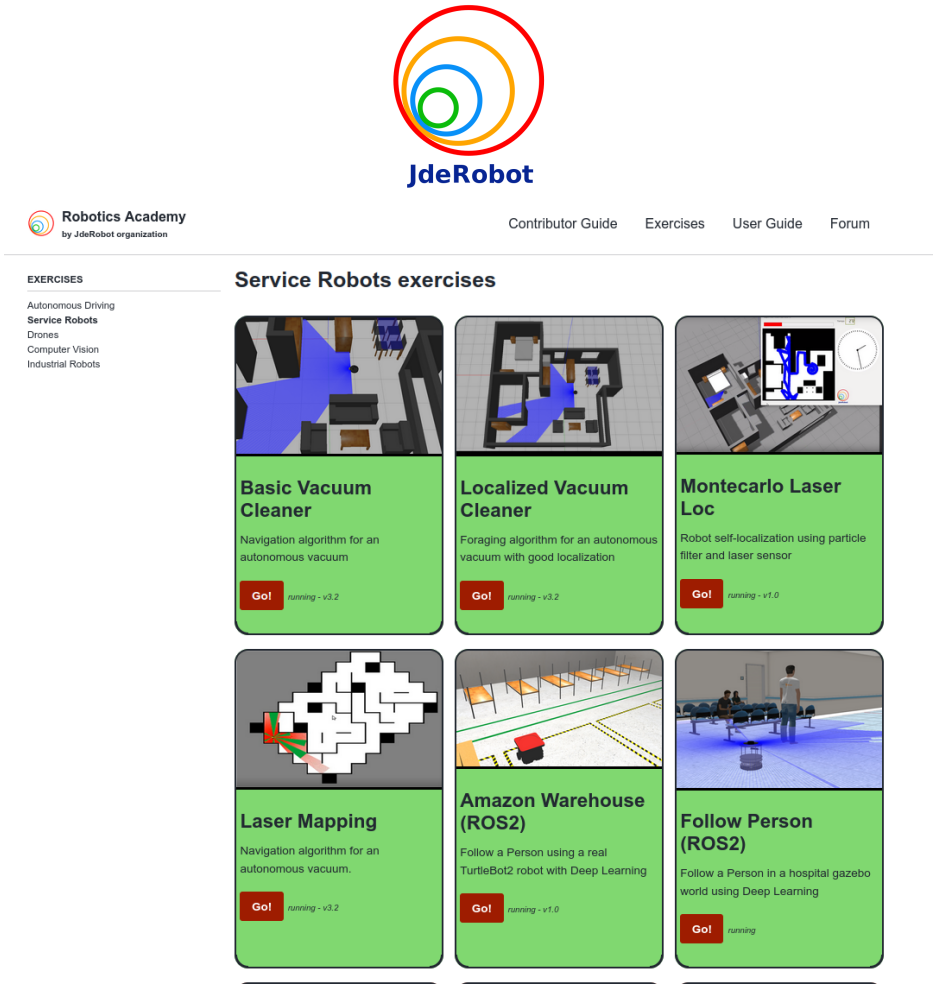
July 20, 2024



Universidad Rey Juan Carlos

Introducción

- Robótica de Servicio
  - Robótica social
  - Robótica en logística
- Robótica Educativa
  - Robotics Academy



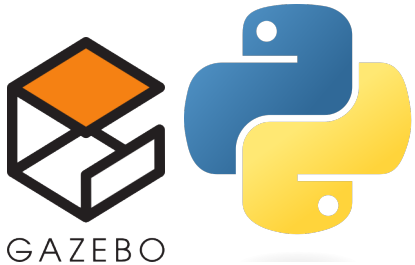
Objetivos

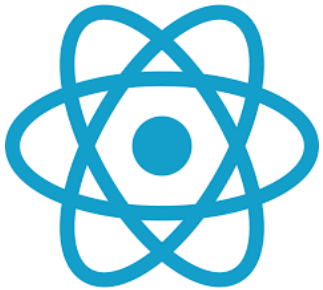
- Mejorar y ampliar la gama de ejercicios de Robotics Academy con dos ejercicios
  - Ejercicio Sigue-Persona
  - Ejercicio Almacén-Amazon

Ingredientes de los ejercicios:

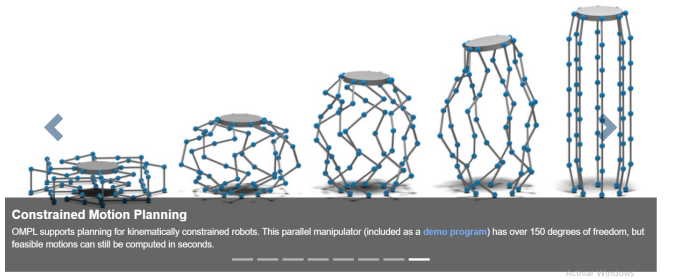
- Infraestructura
- Plantillas Python, frontend web y backend
- Solución de referencia
- Documentación de apoyo al usuario

Herramientas utilizadas



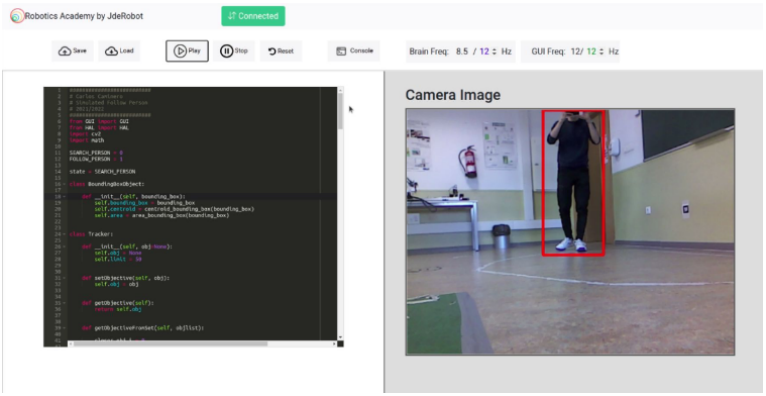


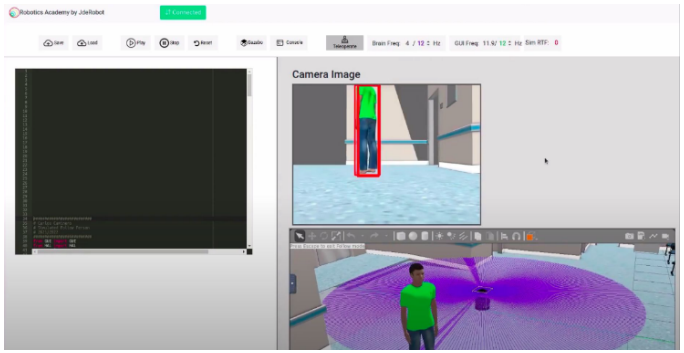
The Open Motion Planning Library



Versión antigua del ejercicio Sigue-Persona

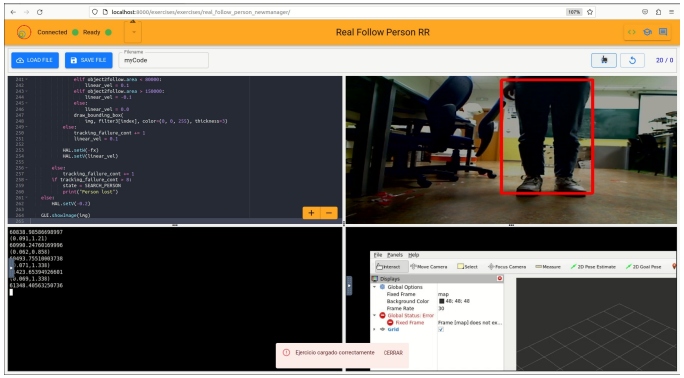
- Robotics Academy versión 4.x
- Infraestructura
  - Entorno simulado de un hospital
  - Robot TurtleBot2 en ROS 2 Foxy (simulado y real)
  - Persona simulada a seguir (autónoma o teleoperada)
- Plantillas Python
- Página web basada en plantillas JavaScript
- Solución de referencia: tracking, VFF y máquina de estados

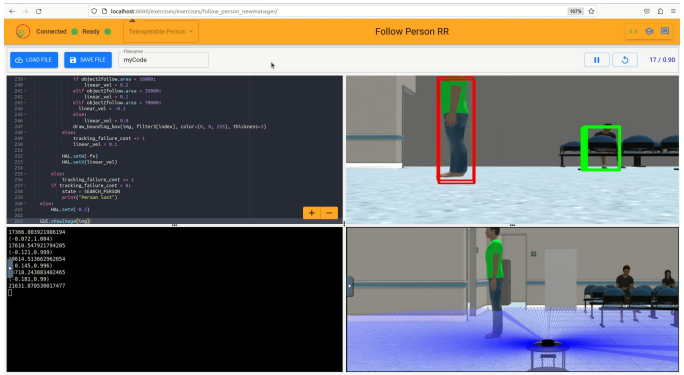




Mejoras

- Actualización del ejercicio a Robotics Academy v5.1
  - Soporte del TurtleBot2 en ROS 2 Humble
  - Reimplementación de la comunicación con el teleoperador
  - Nuevas plantillas Python
  - Nuevo diseño de la página web, plantillas React
- Mejoras en la solución de referencia: controlador PD

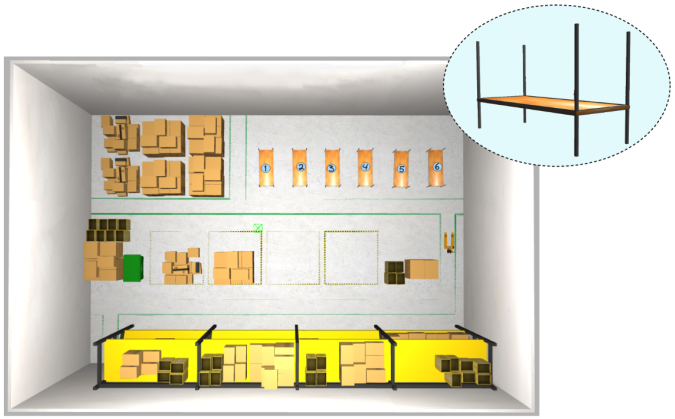




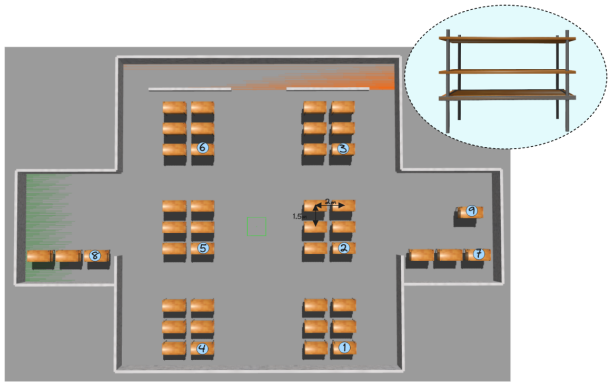
Ejecución típica en `https://youtu.be/VeRk_F9IsDM`

Infraestructura: Entorno simulado

1. Almacén pequeño



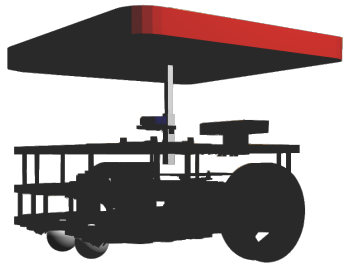
2. Almacén grande



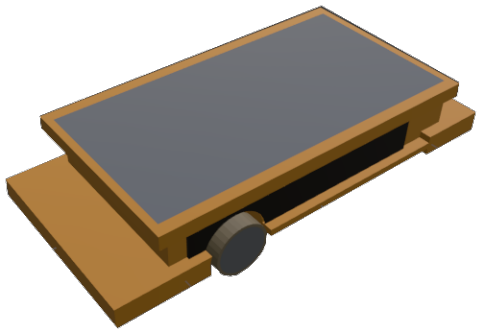
```
<?xml version='1.0' encoding='utf-8'?>
<sdf version="1.6">
  <world name="default">
    <gravity>0 0 -9.8</gravity>
    <physics default="0" name="default_physics" type="ode">
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1</real_time_factor>
      <real_time_update_rate>1000</real_time_update_rate>
    </physics>

    <model name="aws_robomaker_warehouse_ShelfF_01_001">
      <include>
        <uri>model://aws_robomaker_warehouse_ShelfF_01</uri>
      </include>
      <pose frame="">-5.795143 -0.956635 0 0 0 0</pose>
    </model>

    <model name="aws_robomaker_warehouse_WallB_01_001">
      <include>
        <uri>model://aws_robomaker_warehouse_WallB_01</uri>
      </include>
      <pose frame="">0.0 0.0 0 0 0 0</pose>
    </model>

    <model name="pallet_0">
      <include>
        <uri>model://warehouse_pallet3</uri>
      </include>
      <pose frame="">3.728 0.57943 0 0 0 1.588</pose>
    </model>

  [...]
  </world>
</sdf>
```

Infraestructura: Robots simulado

1. Robot logístico holonómico

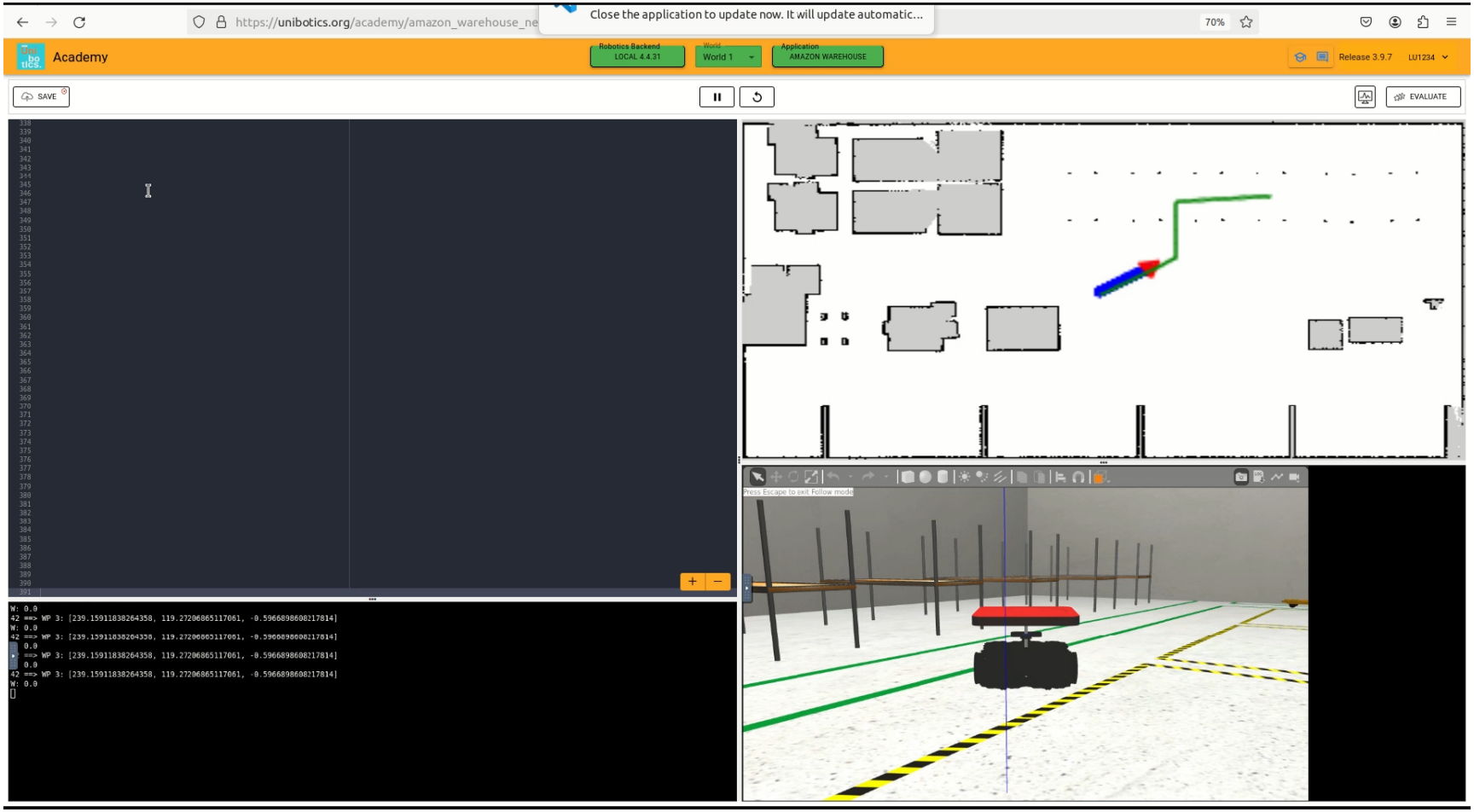


2. Robot logístico Ackermann



```xml
<?xml version="1.0"?>
<sdf version="1.5">
  <model name="ackermann_logistic_robot">
    <pose>0 0 0.0 0 0 0</pose>
    <link name='base_link' />
    <joint name='chassis_link_joint' type='fixed'>
      <pose>0.0 0.0 0.0 0 0 0</pose>
      <parent>base_link</parent>
      <child>chassis_link</child>
    </joint>
    <link name='chassis_link'>
      <inertial>
        <pose>1e-06 0.07307 0.65096 0 -0 0</pose>
        <mass>1580</mass>
        <inertia>
          [...]
        </inertia>
      </inertial>
      <collision name='chassis_link_fixed_joint_lump__chassis_link_collision'>
        <pose>0 0 0.0 0 -0 0</pose>
        <geometry>
          <mesh>
            <scale>0.5 0.5 0.5</scale>
            <uri>model://ackermann_logistic_robot/meshes/chassis.dae</uri>
          </mesh>
        </geometry>
        <surface>
          [...]
        </surface>
      </collision>
      <visual name='chassis_link_fixed_joint_lump__chassis_link_visual'>
        <pose>0 0 0.0 0 -0 0</pose>
        <geometry>
          <mesh>
            <scale>0.5 0.5 0.5</scale>
            <uri>model://ackermann_logistic_robot/meshes/chassis.dae</uri>
          </mesh>
        </geometry>
      </visual>
      <self_collide>1</self_collide>
    </link>
    [...]
    </model>
</sdf>
```

Página Web


Close the application to update now. It will update automatic...
https://unibotics.org/academy/amazon_warehouse_ne
Academy
Robotics Backend
LOCAL 4.4.31
World
World 1
Application
AMAZON WAREHOUSE
Release 3.9.7
SAVE
EVALUATE

Página Web

```
{% extends "react_frontend/exercise_base.html" %}
{% load react_component %}

{% block exercise_header %}
{% endblock %}

{% block react-content %}
    {% if deployment %}
        {% react_component exercise/amazon_warehouse_newmanager/AmazonWarehouseRR %}
            {% react_component components/containers/MaterialBox id="exercise-container" %}
            {% react_component components/common/MainAppBar exerciseName="Amazon Warehouse" url="https://jderobot.github.io/RoboticsAcademy/exercises/MobileRobots/amazon_warehouse/"%}
            {% react_component RA_links/WorldSelectorRA %} {% end_react_component %}
            {% end_react_component %}
            {% react_component components/containers/MaterialBox id="content" %}
                {% react_component components/containers/MaterialBox id="content-exercise" %}
                    {% react_component components/containers/ExerciseControl %}{% end_react_component %}
                    {% react_component components/containers/FlexContainer row %}
                    {% react_component components/containers/FlexContainer console %}
                    {% react_component components/editors/AceEditorRobot %}{% end_react_component %}
                        {% react_component components/visualizers/VncConsoleViewer%}{% end_react_component %}
                        {% end_react_component%}
                        {% react_component components/containers/FlexContainer%}
                            {% react_component exercise/amazon_warehouse_newmanager/SpecificAmazonWarehouse %}{% end_react_component %}
                            {% react_component components/visualizers/GazeboViewer%}{% end_react_component %}
                        {% end_react_component %}
                {% end_react_component %}
            {% end_react_component %}
            {% react_component components/message-system/Loading %}{% end_react_component %}
            {% react_component components/message-system/Alert %}{% end_react_component %}
{% end_react_component %}
{% end_react_component %}
    {% end_react_component %}
    {% else %}
        {% react_component exercise/amazon_warehouse_newmanager/AmazonWarehouseRR %}
            {% react_component components/wrappers/MaterialBox id="exercise-container" %}
                {% react_component components/layout_components/MainAppBar exerciseName="Amazon Warehouse" url="https://jderobot.github.io/RoboticsAcademy/exercises/MobileRobots/amazon_warehouse/"%}
                {% react_component components/visualizers/WorldSelector %} {% end_react_component %}
                {% end_react_component %}
            {% react_component components/wrappers/MaterialBox id="content" %}
                {% react_component components/wrappers/MaterialBox id="content-exercise" %}
                    {% react_component components/layout_components/ExerciseControl %}{% end_react_component %}
                    {% react_component components/wrappers/FlexContainer row %}
                    {% react_component components/wrappers/FlexContainer console %}
                    {% react_component components/editors/AceEditorRobot %}{% end_react_component %}
                    {% react_component components/visualizers/ConsoleViewer%}{% end_react_component %}
                        {% end_react_component%}
                        {% react_component components/wrappers/FlexContainer%}
                            {% react_component exercise/amazon_warehouse_newmanager/SpecificAmazonWarehouse %}{% end_react_component %}
                            {% react_component components/visualizers/GazeboViewer%}{% end_react_component %}
                        {% end_react_component %}
                {% end_react_component %}
            {% end_react_component %}
            {% react_component components/message_system/Loading %}{% end_react_component %}
            {% react_component components/message_system/Alert %}{% end_react_component %}
    {% end_react_component %}
    {% end_react_component %}
    {% end_react_component %}
    {% endif %}
{% endblock %}
```

Plantillas Python

1. HAL API
   - setV(), setW()
   - getPose3d()
   - getSimTime()
   - lift(), putdown()
2. GUI API
   - showPath()
   - getMap()

```python
import rclpy
import sys
import cv2
import threading

from interfaces.motors import PublisherMotors
from interfaces.pose3d import ListenerPose3d
from interfaces.laser import ListenerLaser
from interfaces.platform_controller import PlatformCommandListener
from interfaces.platform_publisher import PublisherPlatform

# Hardware Abstraction Layer
class HAL:
    def __init__(self):
        rclpy.init(args=sys.argv)
        rclpy.create_node('HAL')

        self.motors = PublisherMotors("/amazon_robot/cmd_vel", 4, 0.3)
        self.pose3d = ListenerPose3d("/amazon_robot/odom")
        self.laser = ListenerLaser("/amazon_robot/scan")
        self.platform_listener = PlatformCommandListener()
        self.platform_pub = PublisherPlatform("/send_effort")

        # Spin nodes so that subscription callbacks lift topic data
        executor = rclpy.executors.MultiThreadedExecutor()
        executor.add_node(self.pose3d)
        executor.add_node(self.laser)
        executor.add_node(self.platform_listener)
        executor_thread = threading.Thread(target=executor.spin, daemon=True)
        executor_thread.start()

    @classmethod
    def initRobot(cls):
        new_instance = cls()
        return new_instance

    def getPose3d(self):
        return self.pose3d.getPose3d()

    def getLaserData(self):
        return self.laser.getLaserData()

    def setV(self, velocity):
        self.motors.sendV(velocity)

    def setW(self, velocity):
        self.motors.sendW(velocity)

    def lift(self):
        self.platform_pub.lift()

    def putdown(self):
        self.platform_pub.putdown()
```

Soluciones referencia

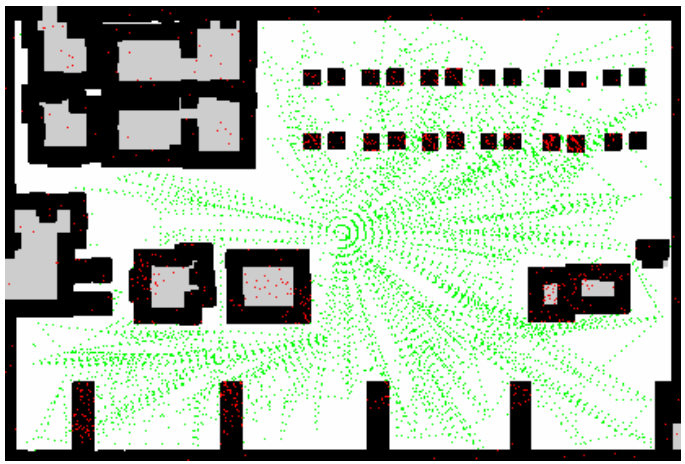
1. Planificación espacial con robot holonómico y geometría "circular"
2. Planificación espacial con robot holonómico y geometría "rectangular"
3. Planificación basado en controles con robot Ackermann

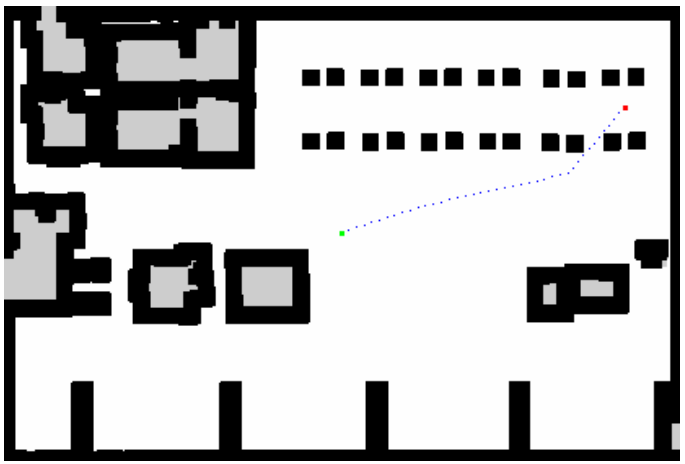
Soluciones referencia

- Fases
  - Planificación de la ruta con OMPL:
    - Espacio de estados (y espacio de controles)
    - Criterio de validación de estados
    - Estado inicial y estado final
    - Planificador (y objetivo de optimización)
    - Resolución del problema $\rightarrow$ lista de *waypoints*
  - Ejecución de la ruta
- Máquina de estados

Soluciones referencia

1. Planificación espacial con robot holonómico y geometría "circular"
   - *RealVectorStateSpace*, (x, y)
   - *PathLengthOptimizationObjective* → Ruta óptima





   - Indexado por posición → recorriendo secuencia (x, y)

Soluciones referencia

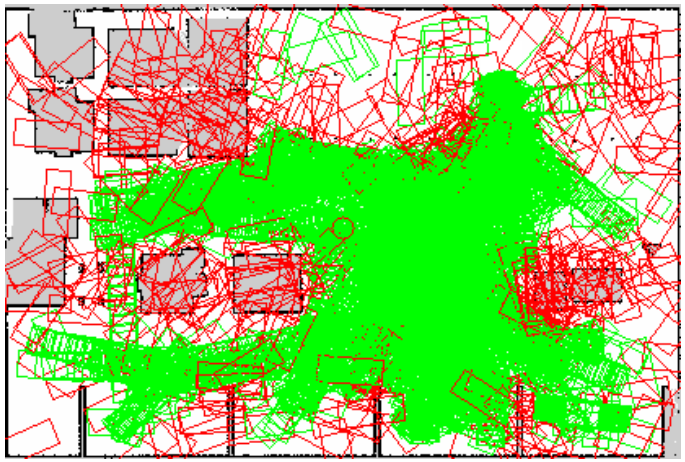
1. Planificación espacial con robot holonómico y geometría "circular"

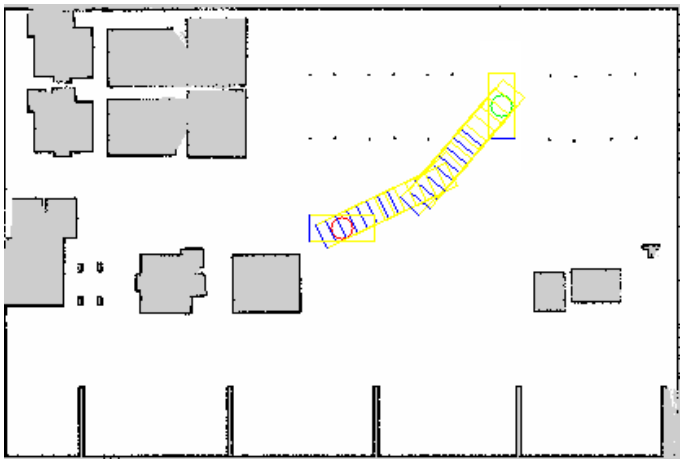
```
1   def isStateValid(state):
2       w = min(int(state[0]), MAP_WIDTH - 1)
3       h = min(int(state[1]), MAP_HEIGHT - 1)
4
5       c = eroded_map[h, w]
6       free = c[0] * 255 > 250 and c[1] * 255 > 250 and c[2] * 255 > 250
7       return free
```

Ejecución típica en `https://youtu.be/EVt9vYqEoDg`

Soluciones referencia

2. Planificación espacial con robot holonómico y geometría "rectangular"
   - *DubinsStateSpace*, (x, y, yaw)
   - *PathLengthOptimizationObjective* → Ruta óptima





   - Indexado por posición → recorriendo secuencia (x, y)

Soluciones referencia

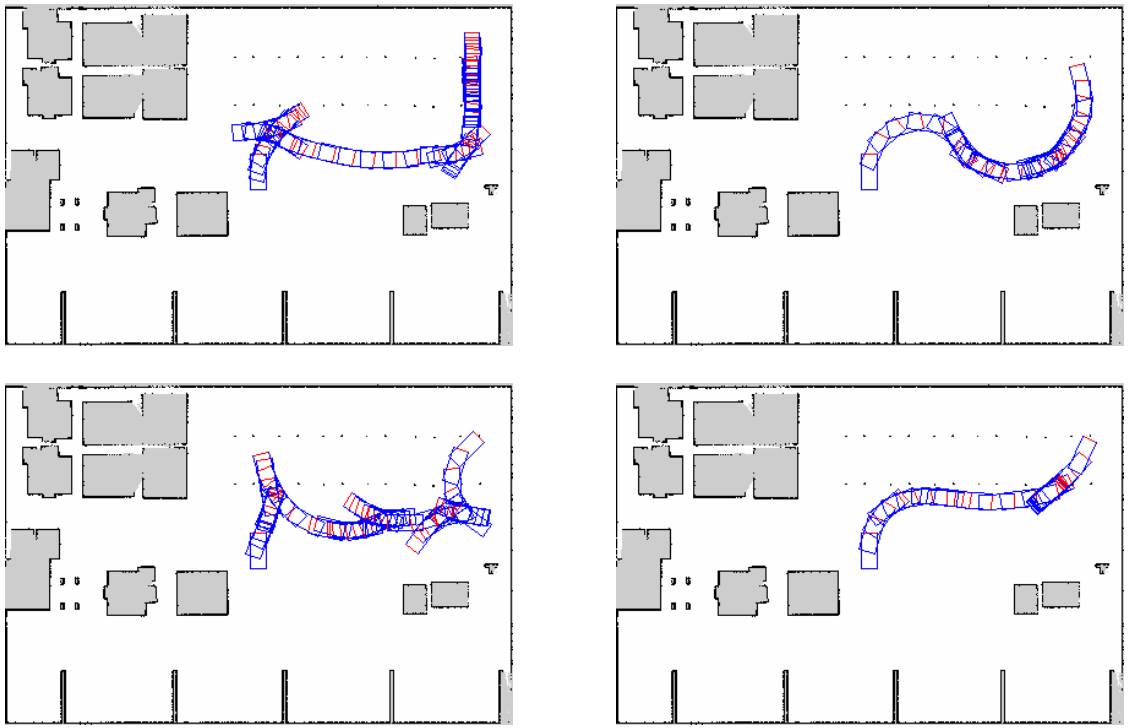
2. Planificación espacial con robot holonómico y geometría "rectangular"

```
1   def isStateValid(state):
2       w = min(int(state.getX()), MAP_WIDTH - 1)
3       h = min(int(state.getY()), MAP_HEIGHT - 1)
4       yaw = state.getYaw()
5
6       for x in range(-robotw, robotw, 1):
7           for y in range(-robotl, robotl, 1):
8               newx, newy = rotationMatrix(x, y, w, h, yaw)
9               if 0 < newx < MAP_WIDTH and 0 < newy < MAP_HEIGHT:
10                  c = mapImg[newy, newx]
11                  free = c[0] * 255 > 250 and c[1] * 255 > 250 and c[2] * 255 > 250
12              else:
13                  free = False
14              if not free:
15                  break
16          if not free:
17              break
18      return free
```

Ejecución típica en `https://youtu.be/-2D90I-wZKs`

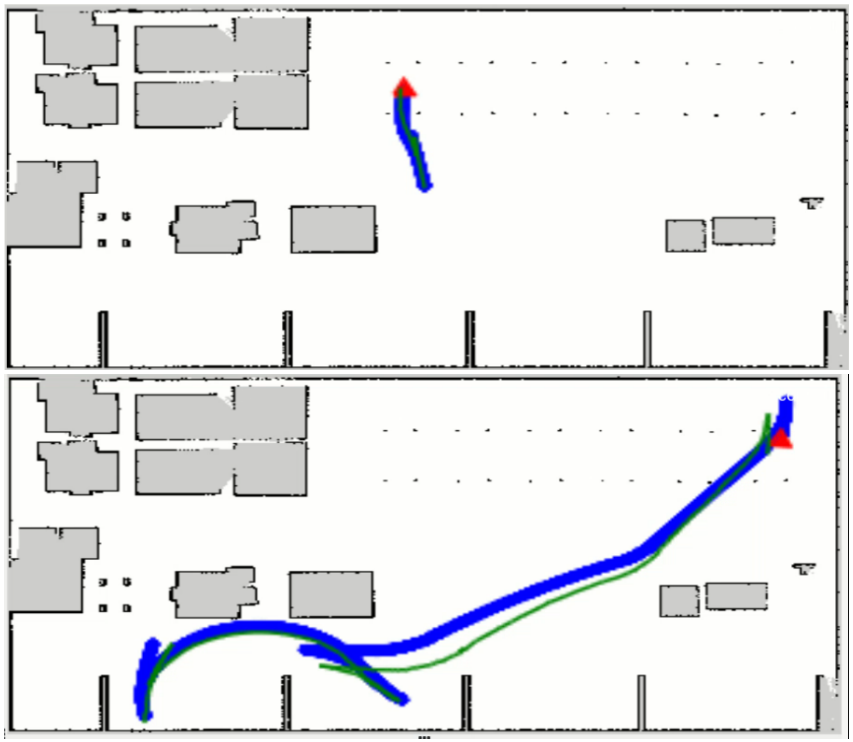
Soluciones referencia

3. Planificación basado en controles con robot Ackermann
   - OMPL app: *setEnviromentMesh()*, *setRobotMesh()*
   - *KinematicCarPlanning*: *SE2StateSpace*, (x, y, yaw) + *control space*,(v, w, duration)
   - Ruta aproximada → Ruta subóptima



Soluciones referencia

3. Planificación basado en controles con robot Ackermann
   - Indexado en el tiempo → recorriendo secuencia (v, w, duration)



Ejecución típica en `https://www.youtube.com/0P04S80mn30`

Conclusiones: objetivos cumplidos

- Actualización del ejercicio Sigue-Persona a RA v5.1
- Creación de un nuevo ejercicio Almacén-Amazon

Conclusiones: líneas futuras

- Mejoras en la solución
- TurtleBot2 + Almacén-Amazon

Gracias por su atención