

# Aplicaciones de robótica de servicio en la plataforma académica Robotics Academy

Lucía Lishan Chen Huang

Trabajo Fin de Grado  
Grado de Ingeniería de Robótica Software

22 de julio de 2024



## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

## 1 Introducción

## 2 Objetivos

### 3 Herramientas utilizadas

## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

# Introducción

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



The screenshot shows the JdeRobot Robotics Academy website. At the top right are links for 'Contributor Guide', 'Exercises', 'User Guide', and 'Forum'. On the left, there's a sidebar with 'EXERCISES' and categories like 'Autonomous Driving', 'Service Robots', 'Drones', 'Computer Vision', and 'Industrial Robots'. The main area is titled 'Service Robots exercises' and displays six green cards, each representing a different exercise:

- Basic Vacuum Cleaner**: Navigation algorithm for an autonomous vacuum.
- Localized Vacuum Cleaner**: Foraging algorithm for an autonomous vacuum with good localization.
- Montecarlo Laser Loc**: Robot self-localization using particle filter and laser sensor.
- Laser Mapping**: Navigation algorithm for an autonomous vacuum.
- Amazon Warehouse (ROS2)**: Follow a Person using a real Turtlebot2 robot with Deep Learning.
- Follow Person (ROS2)**: Follow a Person in a hospital gazebo world using Deep Learning.

Each card has a 'Get' button and a 'running -v1.0' link.

## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

# 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
- Frontend y backend web, plantillas Python
- Solución de referencia
- Documentación de apoyo al usuario

## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

## 4 Ejercicio Sigue-Persona

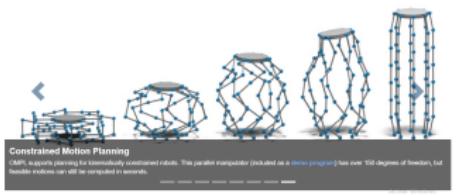
## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

# Herramientas utilizadas



The Open Motion Planning Library



## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

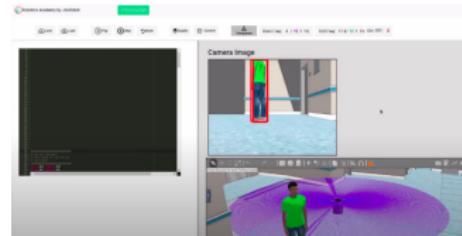
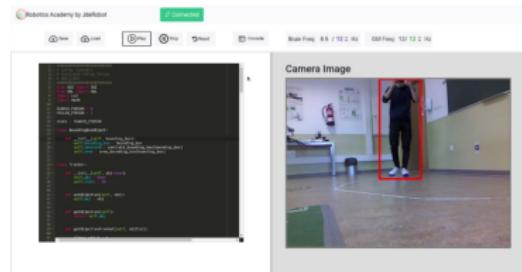
## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

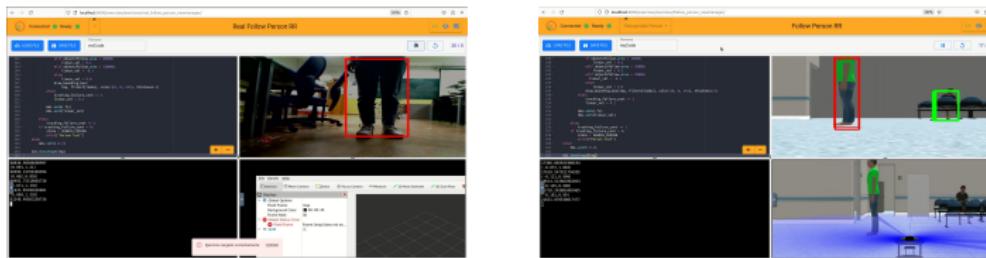
# Versión antigua del ejercicio Sigue-Persona

- 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](https://youtu.be/VeRk_F9IsDM)

## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

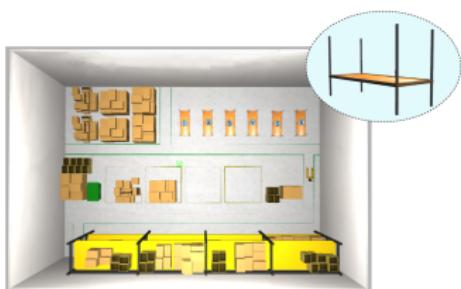
## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

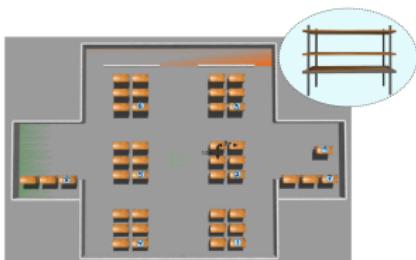
## 6 Conclusiones

# Infraestructura: Entorno simulado

## ① Almacén pequeño



## ② Almacén grande



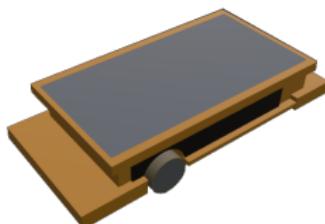
```
1 <?xml version='1.0' encoding='utf-8'?>
2 <sdf version="1.6">
3   <world name="default">
4     <gravity>0 0 -9.8</gravity>
5     <physics default="0" name="default_physics" type="ode">
6       <max_step_size>0.001</max_step_size>
7       <real_time_factor>1</real_time_factor>
8       <real_time_update_rate>1000</real_time_update_rate>
9     </physics>
10
11   <model name="aws_robomaker_warehouse_ShelfF_01_001">
12     <include>
13       | <uri>model://aws_robomaker_warehouse_ShelfF_01</uri>
14     </include>
15     <pose frame="">-5.795143 -0.956635 0 0 0 0</pose>
16   </model>
17
18   <model name="aws_robomaker_warehouse_WallB_01_001">
19     <include>
20       | <uri>model://aws_robomaker_warehouse_WallB_01</uri>
21     </include>
22     <pose frame="">0.0 0.0 0 0 0 0</pose>
23   </model>
24
25   <model name="pallet_0">
26     <include>
27       | <uri>model://warehouse_pallet3</uri>
28     </include>
29     <pose frame="">3.728 0.57943 0 0 1.588</pose>
30   </model>
31
32   [...]
33 </world>
34 </sdf>
```

# Infraestructura: Robots simulado

## ① Robot logístico holonómico

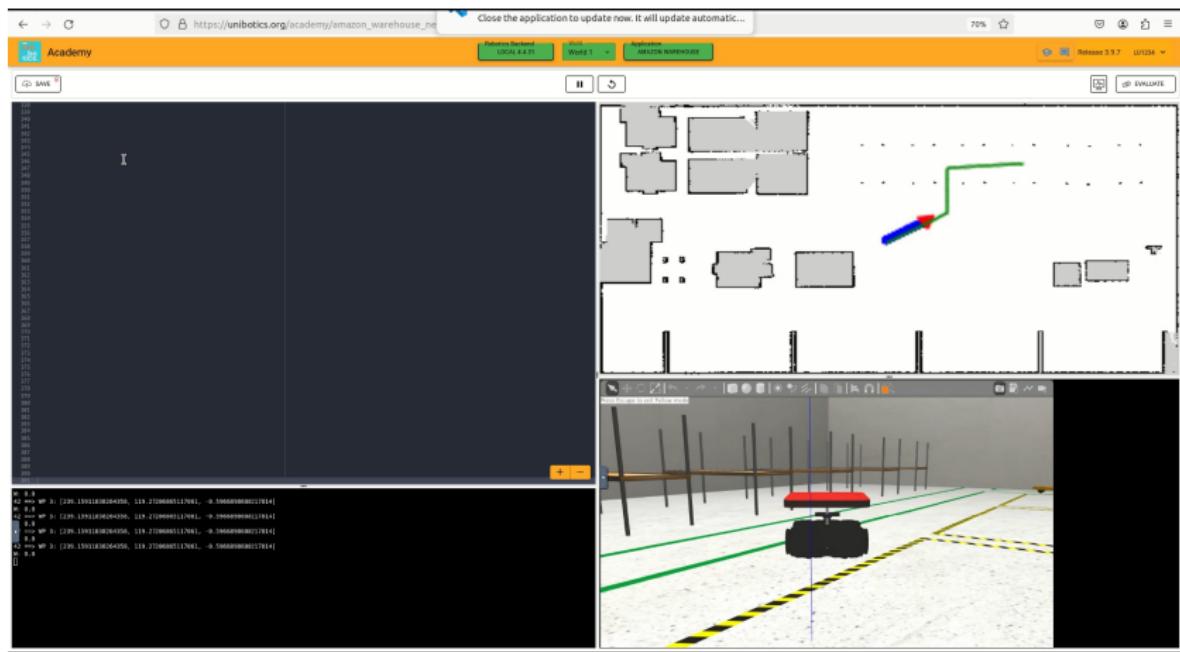


## ② Robot logístico Ackermann



```
1  <?xml version="1.0"?>
2  <sdf version="1.5">
3   <model name="ackermann_logistic_robot">
4     <pose>0 0 0 0 0</pose>
5     <link name="base_link" />
6     <joint name="chassis_link_fixed_joint" type="fixed">
7       <pose>0 0 0 0 0 0</pose>
8       <parent>base_link</parent>
9       <child>chassis_link</child>
10      </joint>
11      <link name="chassis_link">
12        <inertial>
13          <pose>1e-06 0.07307 0.65096 0 -0 0</pose>
14          <mass>1500</mass>
15          <inertia>
16            [...]
17          </inertia>
18        </inertial>
19        <collision name="chassis_link_fixed_joint_lump_chassis_link_collision">
20          <pose>0 0 0 -0 0</pose>
21          <geometry>
22            <mesh>
23              <scale>0.5 0.5 0.5</scale>
24              <uri>model://ackermann_logistic_robot/meshes/chassis.dae</uri>
25            </mesh>
26            <geometry>
27              <surface>
28                [...]
29              </surface>
30            </geometry>
31            <visual name="chassis_link_fixed_joint_lump_chassis_link_visual">
32              <pose>0 0 0 -0 0</pose>
33              <geometry>
34                <mesh>
35                  <scale>0.5 0.5 0.5</scale>
36                  <uri>model://ackermann_logistic_robot/meshes/chassis.dae</uri>
37                </mesh>
38                <geometry>
39                  <visual>
40                    <self_collide>1</self_collide>
41                  </visual>
42                  [...]
43                </geometry>
44              </visual>
45            </geometry>
46          </collision>
47        </link>
48      </model>
49    </sdf>
```

# Página Web



# Página Web

```
1  [% extends 'react_frontend/exercise_base.html' %]
2  [% load react_component %]
3
4  [% block exercise_header %]
5  [% endblock %]
6
7  [% block react-content %]
8  [% if deployment %]
9      [% react_component exercise/amazon_warehouse_newmanager/AmazonWarehouseR %]
10     [% react_component components/containers/MaterialBox id="exercise-container" %]
11     [% react_component components/common/MainAppBar exerciseName="Amazon Warehouse" url="https://jderobot.github.io/RoboticsAcademy/exercises/mobileRobots/amazon_warehouse/"%]
12     [% react_component NA_links/NavBarSelectorRA %] [% end_react_component %]
13     [% end_react_component %]
14     [% react_component components/containers/MaterialBox id="content" %]
15     [% react_component components/containers/MaterialBox id="content-exercise" %]
16     [% react_component components/containers/exerciseControl %] [% end_react_component %]
17     [% react_component components/containers/lexContainer row %]
18     [% react_component components/containers/lexContainer console %]
19     [% react_component components/editors/AceEditorRobot %] [% end_react_component %]
20     [% react_component components/visualizers/ConsoleViewerWork %] [% end_react_component %]
21     [% end_react_components %]
22     [% react_component components/containers/FlexContainer %]
23     [% react_component exercise/amazon_warehouse_newmanager/SpecifyAmazonWarehouse %] [% end_react_component %]
24     [% react_component components/visualizers/GazeboViewer %] [% end_react_component %]
25     [% end_react_component %]
26     [% end_react_component %]
27     [% end_react_component %]
28     [% react_component components/message-system>Loading %] [% end_react_component %]
29     [% react_component components/message-system/Alert %] [% end_react_component %]
30
31 [% end_react_component %]
32 [% end_react_component %]
33 [% end_react_component %]
34 [% react_component exercise/amazon_warehouse_newmanager/AmazonWarehouseR %]
35     [% react_component components/wrappers/MaterialBox id="exercise-container" %]
36     [% react_component components/layout_components/MainAppBar exerciseName="Amazon Warehouse" url="https://jderobot.github.io/RoboticsAcademy/exercises/mobileRobots/amazon_warehouse/"%]
37     [% react_component components/visualizers/WorldSelector %] [% end_react_component %]
38     [% end_react_component %]
39     [% react_component components/wrappers/MaterialBox id="content" %]
40     [% react_component components/wrappers/MaterialBox id="content-exercise" %]
41     [% react_component components/layout_components/ExerciseControl %] [% end_react_component %]
42     [% react_component components/wrappers/FlexContainer row %]
43     [% react_component components/wrappers/FlexContainer console %]
44     [% react_component components/editors/AceEditorRobot %] [% end_react_component %]
45     [% react_component components/visualizers/ConsoleViewer %] [% end_react_component %]
46     [% end_react_components %]
47     [% react_component components/wrappers/FlexContainer %]
48     [% react_component exercise/amazon_warehouse_newmanager/SpecifyAmazonWarehouse %] [% end_react_component %]
49     [% react_component components/visualizers/GazeboViewer %] [% end_react_component %]
50     [% end_react_component %]
51     [% end_react_component %]
52     [% end_react_component %]
53     [% react_component components/message_system>Loading %] [% end_react_component %]
54     [% react_component components/message_system/Alert %] [% end_react_component %]
55
56 [% end_react_component %]
57 [% end_react_component %]
58 [% endif %]
59 [% endblock %]
```

# Plantillas Python

## ① HAL API

- setV(), setW()
- lift(), putdown()
- getPose3d()
- getSimTime()

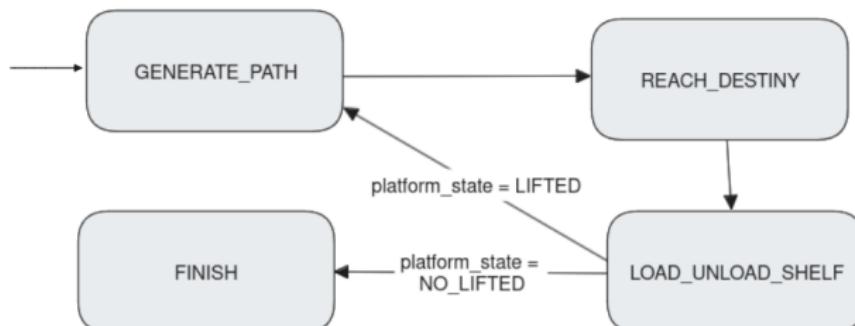
## ② GUI API

- getMap()
- showPath()

```
1 import rclpy
2 import sys
3 import cv2
4 import threading
5
6 from interfaces.motors import PublisherMotors
7 from interfaces.pose3d import ListenerPose3d
8 from interfaces.laser import ListenerLaser
9 from interfaces.platform_controller import PlatformCommandListener
10 from interfaces.platform_publisher import PublisherPlatform
11
12 # Hardware Abstraction Layer
13 class HAL:
14     def __init__(self):
15         rclpy.init(argv=sys.argv)
16         rclpy.create_node('HAL')
17
18         self.motors = PublisherMotors("/amazon_robot/cmd_vel", 4, 0.3)
19         self.pose3d = ListenerPose3d("/amazon_robot/odom")
20         self.laser = ListenerLaser("/amazon_robot/scan")
21         self.platform_listener = PlatformCommandListener()
22         self.platform_pub = PublisherPlatform("/send_effort")
23
24         # Spin mode so that subscription callbacks lift topic data
25         executor = rclpy.executors.MultiThreadedExecutor()
26         executor.add_node(self.pose3d)
27         executor.add_node(self.laser)
28         executor.add_node(self.platform_listener)
29         executor_thread = threading.Thread(target=executor.spin, daemon=True)
30         executor_thread.start()
31
32 @classmethod
33 def initRobot(cls):
34     new_instance = cls()
35     return new_instance
36
37 def getPose3d(self):
38     return self.pose3d.getPose3d()
39
40 def getLaserData(self):
41     return self.laser.getLaserData()
42
43 def setv(self, velocity):
44     self.motors.sendV(velocity)
45
46 def setw(self, velocity):
47     self.motors.sendW(velocity)
48
49 def lift(self):
50     self.platform_pub.lift()
51
52 def putdown(self):
53     self.platform_pub.putdown()
```

# Soluciones referencia

- Etapas
  - 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 → lista de *waypoints*
  - Ejecución de la ruta
  - Carga/descarga de la estantería
- Máquina de estados



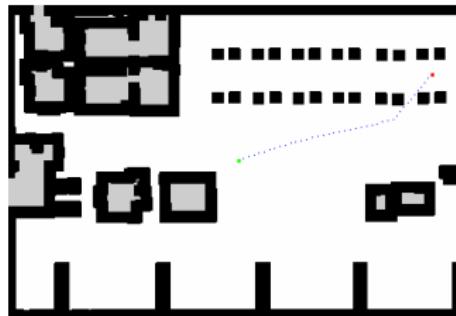
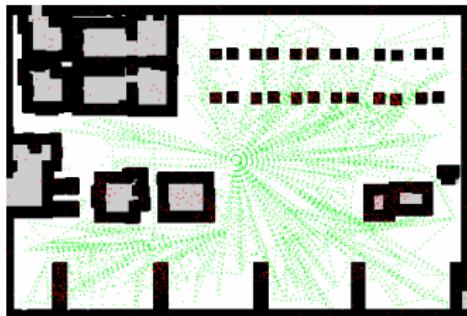
# Soluciones referencia

- 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

## ① Planificación espacial con robot holonómico y geometría “circular”

- *RealVectorStateSpace, (x, y)*
- *PathLengthOptimizationObjective* → Ruta óptima



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

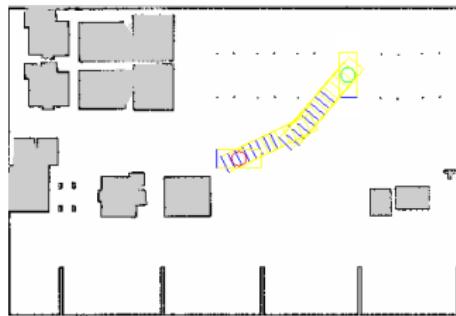
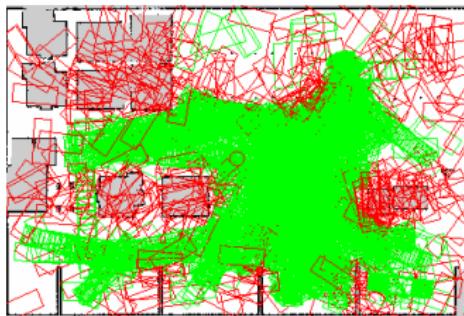
# Soluciones referencia

## 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
```

## 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

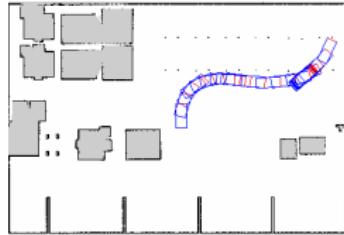
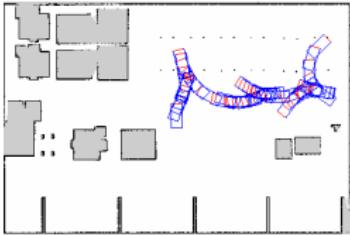
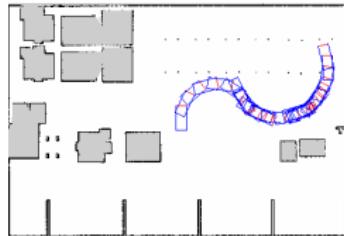
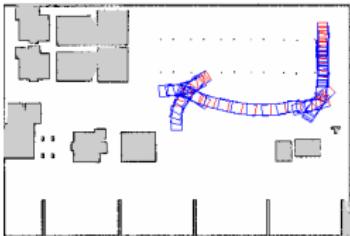
## ② Planificación espacial con robot holonómico y geometría “rectangular”

```
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
```

## Soluciones referencia

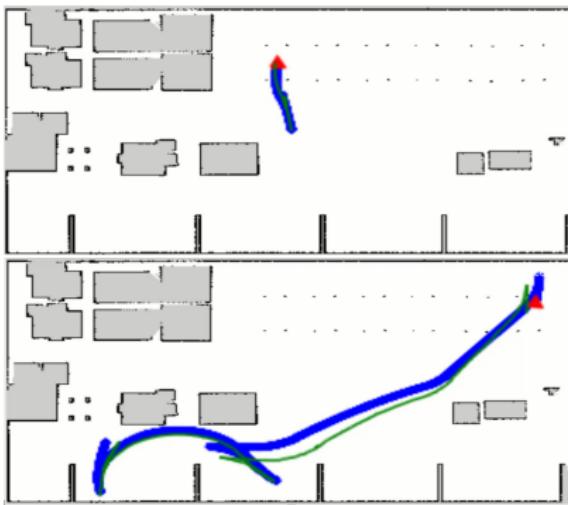
### ③ Planificación basado en controles con robot Ackermann

- OMPL app
  - *KinematicCarPlanning*:  $SE2StateSpace$ ,  $(x, y, \text{yaw}) + control space, (v, w, \text{duration})$
  - *setEnvironmentMesh()*, *setRobotMesh()*
- Ruta subóptima (ruta aproximada)



## Soluciones referencia

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



# Soluciones referencia

Ejecución típica en:

- ① <https://www.youtube.com/EVt9vYqEoDg>
- ② <https://www.youtube.com/-2D90I-wZKs>
- ③ <https://www.youtube.com/0P04S80mn30>

## 1 Introducción

## 2 Objetivos

## 3 Herramientas utilizadas

## 4 Ejercicio Sigue-Persona

## 5 Ejercicio Almacén-Amazon

## 6 Conclusiones

## Conclusiones: objetivos cumplidos

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

## Conclusiones: líneas futuras

- Mejorar el algoritmo de navegación de la solución basada en controles
- Adaptar la solución basada en controles para el modelo del robot holonómico
- Adaptar una versión del ejercicio Amazon Warehouse para que sea utilizable con robots TurtleBot2.

*Gracias por su atención*