

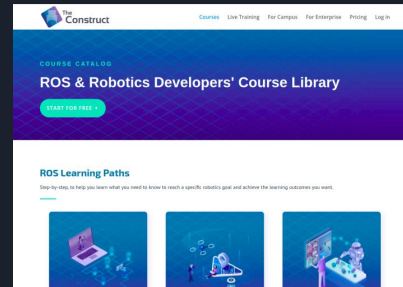
Ejercicios Sigue-Persona para la plataforma académica Robotics Academy, usando un robot real y simulado en ROS2.

Trabajo Fin de Grado (2021/2022)

Alumno: Carlos Caminero Abad
Tutor: Jose María Cañas Plaza

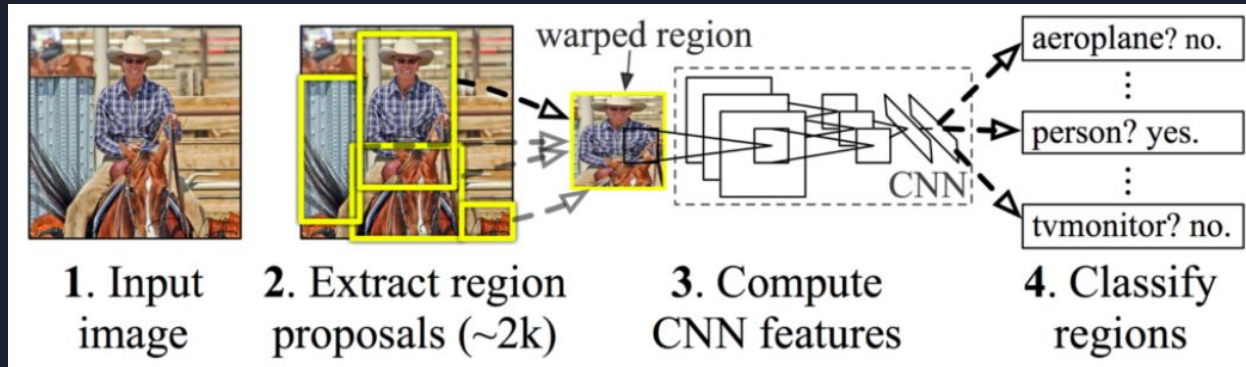
Introducción

- La robótica aporta cada vez más servicios útiles a la sociedad.
- Es importante la formación profesional del ingeniero robótico.
- Robótica Educativa
 - The Construct
 - GIRS de la ETSIT (URJC)
 - Robotics Academy y Unibotics



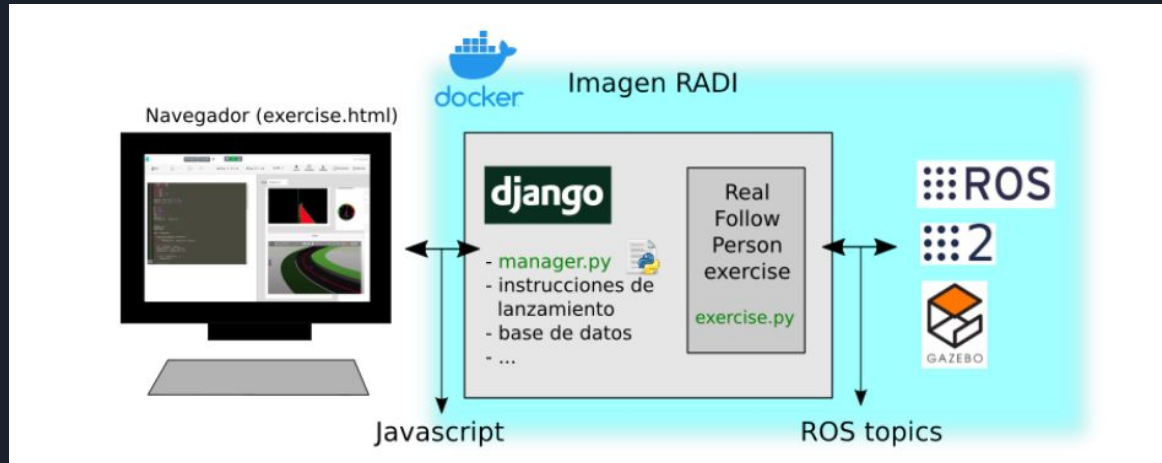
Motivación

- Deep Learning abre un abanico de aplicaciones robóticas.
- Redes Neuronales Convolucionales basadas en Regiones (R-CNN).
- De la clasificación pasamos a la detección de objetos.



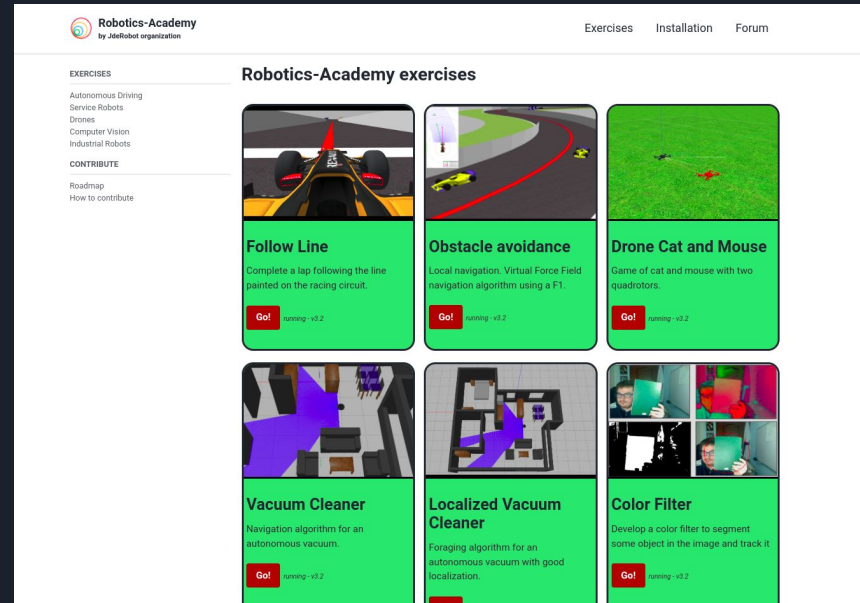
Motivación

- RNA óptima para sistemas computacionales de bajo rendimiento (SSD Inception V2).
Ejemplos: PC sin GPU, contenedores Docker.
- Programación de un robot real + capa de abstracción (HAL y GUI).

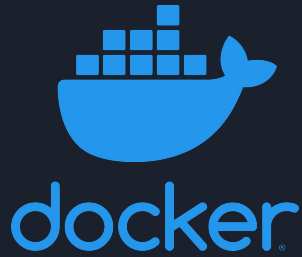


Objetivos

- Desarrollar dos ejercicios para Robotics Academy.
- Soporte de un robot TurtleBot2 simulado y real (ROS2 Foxy).
- Los alumnos programan un algoritmo de seguimiento.
- Utilización de Deep Learning.



Herramientas utilizadas.



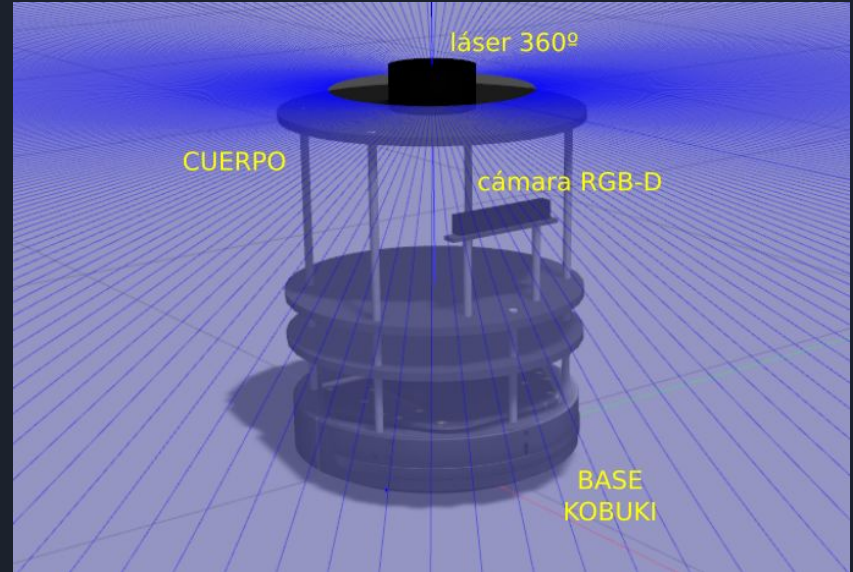
URDF

Xacro



Soporte del TurtleBot2 en ROS2 Foxy

- Drivers de la base Kobuki real en ROS2 (más *wrappers*).
- ¿y para la simulación?
- Diseño del Turtlebot2 en Gazebo más simulación de los sensores y actuadores.
- `kobuki_gazebo` + cuerpo del robot en Xacro.



Infraestructura de los dos ejercicios

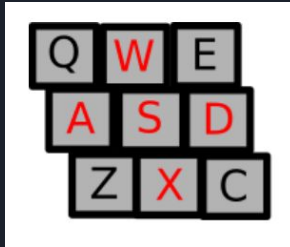
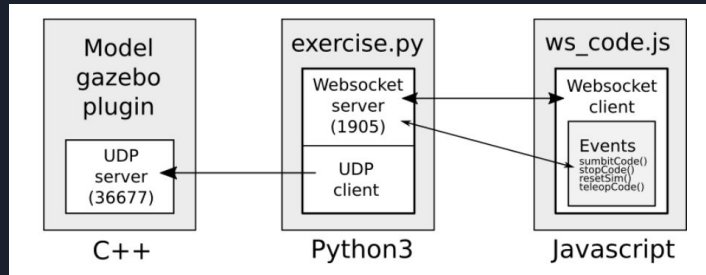
- Desarrollo del frontend de cada ejercicio
 - plantillas web.
- Desarrollo del entorno simulado
 - *plugin* (teleoperación - C++) e integración hospital AWS.
- Creación de *Launchers ROS2*.
- Comunicación con ROS *topics*
 - Plantillas python (HAL/GUI e *interfaces*) adaptadas a ROS2.



Infraestructura de los dos ejercicios

Plugin de Teleoperación

- Controlar un modelo simulado



ws_code.js

```
// Key events to move the model
window.onkeydown = (e) => {
  if (activate_teleop) {
    key_pressed = e.key;
    websocket_code.send("#key_"+key_pressed);
  }
}
```

exercise.py

```
if (message[:4] == "#key"):
    mode = message[message.find('_')+1:]
    if mode == "w":
        self.model_client.sendto(str.encode("UVF"), self.model_address)
    elif mode == "s":
        self.model_client.sendto(str.encode("UVB"), self.model_address)
```

person.cpp

```
void ServerThreadLoop(void)
{
    char msg[3];

    while (true) {
        recv_message(this->sockfd, this->addr, &msg, sizeof msg);
```

Infraestructura de los dos ejercicios.

HAL API - ROS topics

- `setV()` y `setW()` :
 - `/cmd_vel`
 - `/commands/velocity`
- `getLaserData()`:
 - `/scan`
- `getImage()`:
 - `/depth_camera/image_raw`
 - `/image_raw`
- `getPose3d()`:
 - `/odom`
- `getBoundingBoxes()`:
 - Llama a SSD Inception

```
class HAL:

    def __init__(self):
        # init node
        rclpy.init()

        self.motors = PublisherMotors("cmd_vel", 4, 0.3)
        self.laser = ListenerLaser("scan")
        self.camera = ListenerCamera("/depth_camera/image_raw")
        self.odometry = ListenerPose3d("/odom")

        self.listener_executor = MultiThreadedExecutor(num_threads=4)
        self.listener_executor.add_node(self.laser)
        self.listener_executor.add_node(self.camera)
        self.listener_executor.add_node(self.odometry)

        self.net = NeuralNetwork()

        # Update thread
        self.thread = ThreadHAL(self.listener_executor)

    def start_thread(self):
        self.thread.start()

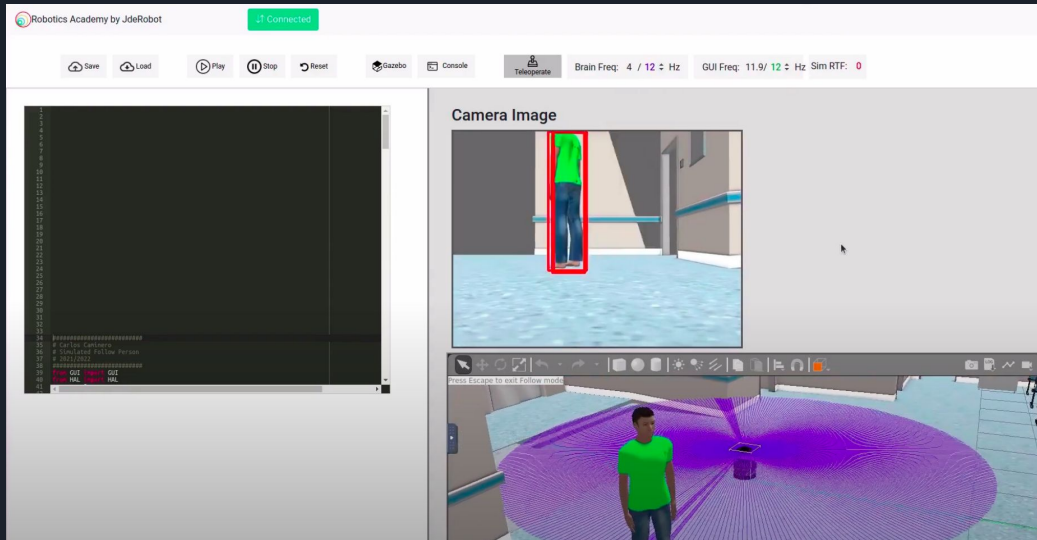
    def setV(self, velocity):
        self.motors.sendV(velocity)

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

    def getLaserData(self):
        values = self.laser.getLaserData().values
        return values[90:0:-1] + values[0:1] + values[360:270:-1]

    def getImage(self):
        return self.camera.getImage().data
```

Plantilla web Sigue-Persona simulado



- editor texto
- botón teleoperación
- *canvas* imagen
- ventana simulador
- ventana terminal


Plantilla web Sigue-Persona Real

Robotics Academy by JdeRobot Connected

Save Load Play Stop Reset Console Brain Freq: 8.5 / 12 Hz GUI Freq: 12/ 12 Hz

```
1 #####
2 @ Simulacrum
3 # Simulated Follow Person
4 # 2021/2022
5 #####
6 from GUI import GUI
7 from HAL import HAL
8 import cv2
9 import math
10
11 SEARCH_PERSON = 0
12 FOLLOW_PERSON = 1
13
14 state = SEARCH_PERSON
15
16 class BoundingBoxObject:
17     def __init__(self, bounding_box):
18         self.bounding_box = bounding_box
19         self.centroid = centroid_bounding_box(bounding_box)
20         self.area = area_bounding_box(bounding_box)
21
22
23
24 class Tracker:
25     def __init__(self, obj=None):
26         self.obj = None
27         self.limit = 50
28
29
30     def setObjective(self, obj):
31         self.obj = obj
32
33
34     def getObjective(self):
35         return self.obj
36
37
38     def getObjectiveFromSet(self, objList):
39         return objList[0]
```

Camera Image



- editor texto
- *canvas* imagen
- ventana terminal



Programación del algoritmo Sigue-Persona

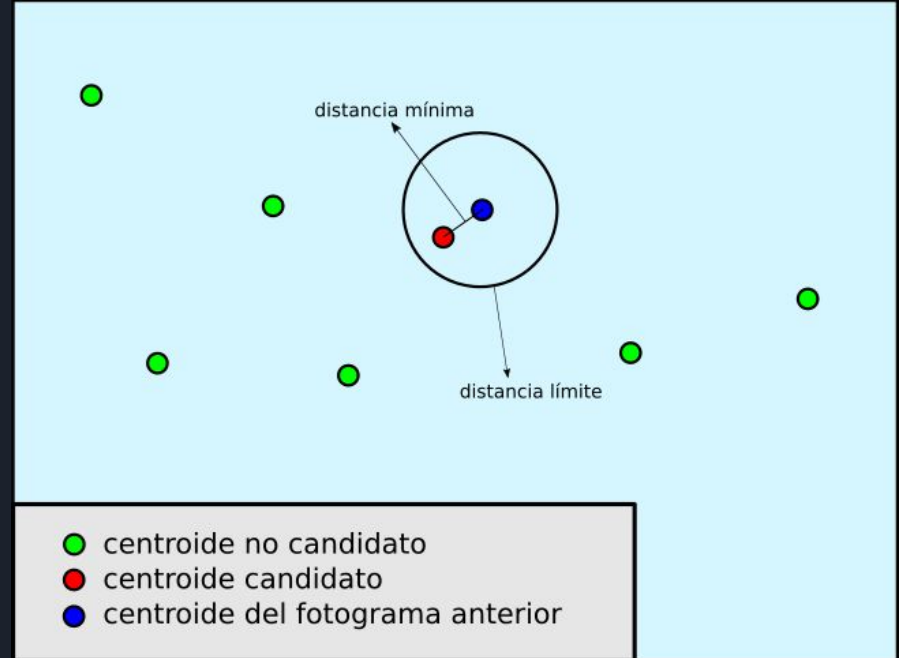
- Dos soluciones de referencia.
- 3 pasos: *tracking*, VFF y Máquina de Estados.
- Diferente comportamiento en real y en simulado.

Variantes alternativas:

- Control de velocidad basado en casos (franjeas).
- Control de velocidad basado en un PID.
- Detección por color de ropa.

Algoritmo de seguimiento visual (tracking)

- Uso de R-CNN
- filtros: *score*, clase, área.
- Clase Tracker (Actualiza objetivo)
- Tolerancia a la pérdida de detecciones en fotogramas.

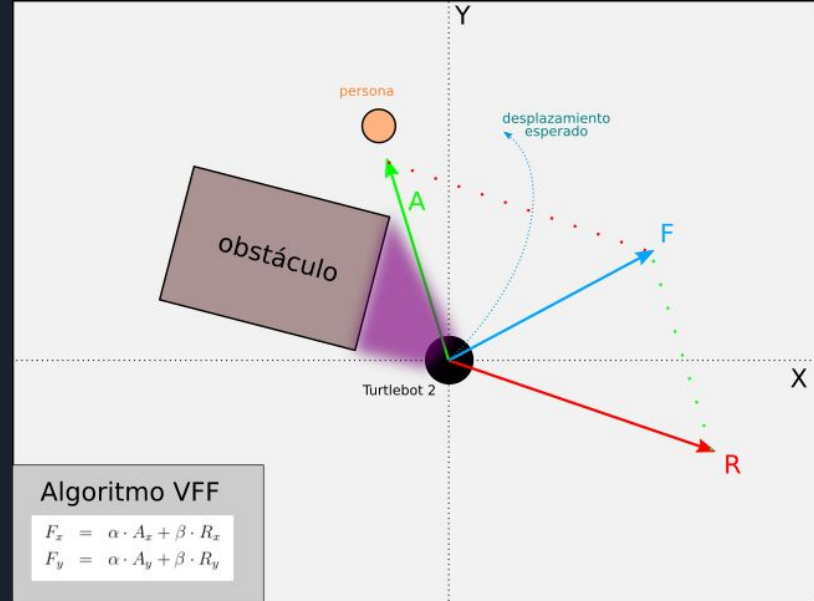


Campo de Fuerzas Virtuales (VFF)

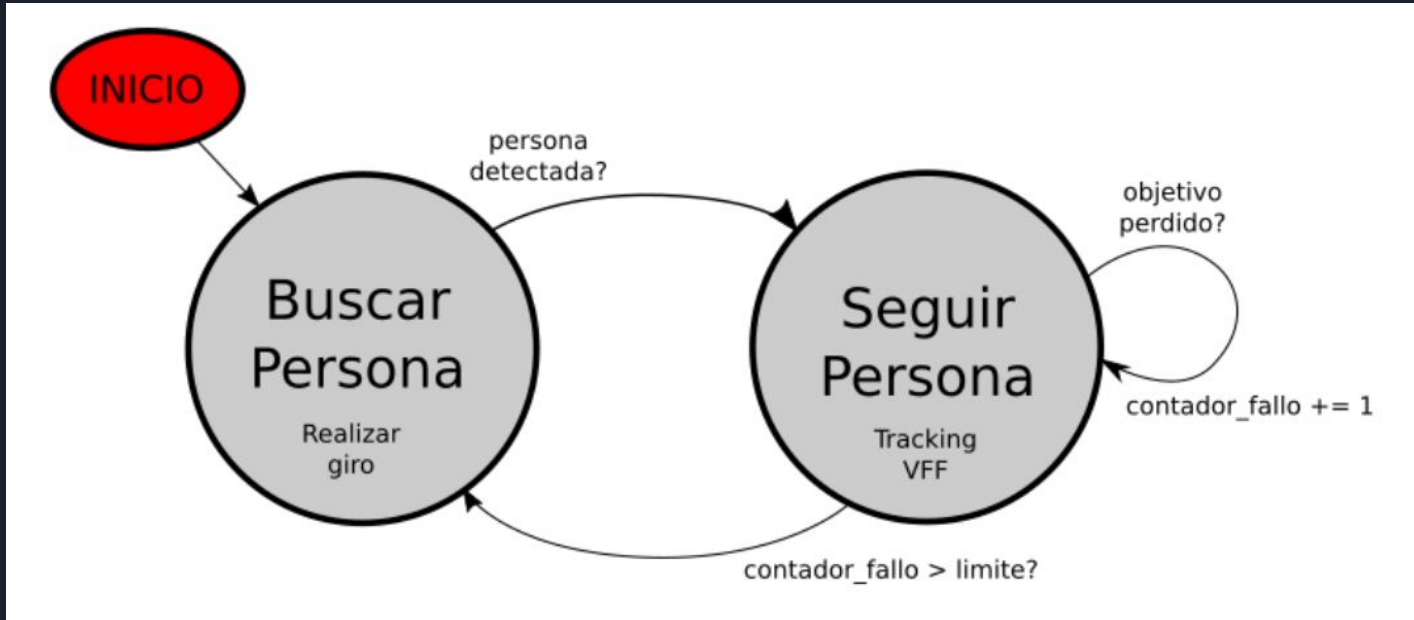
- Movimiento = suma vectorial:
 - Vector A -> objetivo (FOV)
 - Vector R -> Repulsión del láser.
- Dos parámetros que actúan como pesos de los vectores:
 - α - Vector de Atracción.
 - β - Vector de Repulsión.

$$F_x = \alpha \cdot A_x + \beta \cdot R_x$$

$$F_y = \alpha \cdot A_y + \beta \cdot R_y$$



Máquina de Estados Finitos (FSM)



Validación experimental Sigue-Persona Simulado

Simulated Follow Person Robotics Academy



[https://www.youtube.com/
watch?v=fDAU465eVxQ](https://www.youtube.com/watch?v=fDAU465eVxQ)

Validación experimental Sigue-Persona Real

**Real Follow Person
Robotics Academy**



<https://www.youtube.com/watch?v=54Jb4KJwyDM>



Conclusiones. Aportaciones

- TurtleBot2 real y simulado en ROS2 Foxy.
- Dos nuevos ejercicios para Robotics Academy.
- Nuevo escenario simulado y plugin de teleoperación.
- Un robot real en un ejercicio de RA.
- Soluciones de referencia.
- Red Neuronal SSD Inception en RADI.



Conclusiones. Líneas Futuras

- Modelos R-CNN ligeros en más ejercicios de RA.
- Nuevos ejercicios con el TurtleBot2
 - Navegación global en el Laboratorio.
- Soporte de Cámaras RGB-D.
- Sigue-Persona basado usando la profundidad.
 - Transformadas
 - PointCloud



Fin

¿PREGUNTAS?
¡MUCHAS GRACIAS!

