Introducción a la Robótica

Tecnicatura en Automatización y Robótica

FICH - UNL

Trabajo final integrador

Guía de trabajos prácticos 2025 - Entregable 2

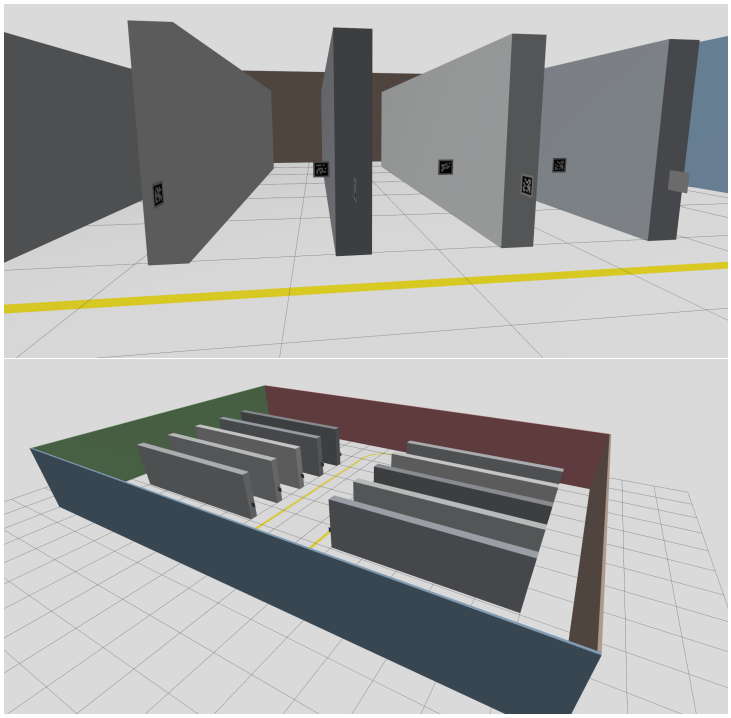
- Fecha de entrega: a convenir con el docente
- Formato de entrega: Archivo comprimido con el código fuente de los paquetes. Utilizar formato pdf en caso de entregar ejercicios de desarrollo o documentos de texto. Proveer en un archivo de texto `README.md` con los comandos necesarios para la ejecución de el/los paquete/s que respondan a la resolución de los ejercicios.
- Archivos disponibles: `entregable-2.zip`
  - 2 carpetas (`AprilTag_36h11` y `Warehouse`) con los modelos necesario para la simulación.

1. Descripción del escenario

La empresa *Distribuidora Central S.A.*, dedicada a la gestión y almacenamiento de productos, enfrenta dificultades para verificar de manera eficiente si los pasillos de su depósito se encuentran despejados antes del inicio de las operaciones diarias. Con el objetivo de automatizar esta tarea, lo contrata a usted para desarrollar un sistema autónomo de inspección selectiva de pasillos.

La empresa pone a disposición un robot móvil diferencial de iguales características al desarrollador en el **entregable 1**. Además, el prototipo cuenta con una cámara **Raspberry Pi Camera Module v2** y un sensor LIDAR **SLAMTEC RPLIDAR C1**, los cuales permiten la detección de marcadores y la identificación de obstáculos en el entorno.

Para las primeras etapas del proyecto, la empresa solicita una demostración simulada del sistema de inspección antes de su implementación real en el depósito. Con este fin, el escenario fue relevado y modelado digitalmente en un archivo *SDF*, representando la disposición y características del espacio físico de la empresa. El entorno de trabajo corresponde a una zona de almacenamiento, conformada por ocho pasillos de estanterías, organizados en dos bloques de cuatro pasillos enfrentados y separados por un corredor central. A lo largo de este corredor se encuentra un trazado de línea guía que servirá como referencia para el desplazamiento del robot. Cada pasillo se encuentra identificado mediante un marcador ubicado en su entrada y otro a metros del inicio, lo que permitirá reconocer el número de identificación. Algunos pasillos pueden incluir obstáculos (por ejemplo cajas), cuya detección forma parte de la tarea de inspección solicitada.



El robot iniciará su recorrido desde la zona de partida ubicada en coordenadas `x: -0.9 y: 0.5` (frame de referencia `world`) con orientación $\pi/2$ y deberá desplazarse hasta el pasillo cuyo identificador coincida con el número recibido como orden de inspección. Una vez frente a la entrada del pasillo designado, el robot deberá detenerse, realizar una inspección utilizando el sensor LIDAR y/o cámara, y determinar si el pasillo se encuentra libre o bloqueado. En caso de detectar un obstáculo, deberá informar la distancia aproximada desde la entrada del pasillo hasta el objeto detectado, como parte del reporte de inspección.

2. Funciones y comportamientos requeridos

1. **Recepción del identificador de pasillo**: El robot deberá recibir un número de identificación de pasillo a través del topic `\inspection_goal` de tipo `std_msgs/Int32`
2. **Seguimiento de línea**: Utilizando la cámara y un algoritmo de seguimiento de linea, deberá desplazarse a lo largo del trazado central, manteniendo la trayectoria indicada con linea amarilla
3. **Identificación de pasillos**: Durante su desplazamiento, el robot deberá detectar los marcadores *AprilTag* ubicados en cada pasillo y reconocer su número de identificación
4. **Inspección del pasillo**: Al identificar el marcador correspondiente al número de pasillo recibido, el robot deberá realizar la inspección del pasillo utilizando los sensores disponibles:
   - En caso de detectar un obstáculo dentro del pasillo, estimar e informar la distancia entre la entrada del pasillo y el obstáculo más cercano
   - En caso contrario, reportar que el pasillo se encuentra despejado

Anexo

Para utilizar el modelo provisto debe descomprimir ambas carpetas del archivo `entregable-2.zip` dentro de una carpeta `models` dentro del paquete a utilizar. Luego

debe modificar el archivo `setup.py` de dicho paquete para poder *instalar* los modelos:

Listado 1 `setup.py`

```
from setuptools import find_packages, setup

import os
from glob import glob

NOMBRE_DEL_PAQUETE = # Completar con el nombre del paquete

setup(
  name=NOMBRE_DEL_PAQUETE,
  version='0.0.0',
  packages=find_packages(exclude=['test']),
  data_files=[
    ('share/ament_index/resource_index/packages',
      ['resource/' + NOMBRE_DEL_PAQUETE]),
    # Install models inside the 'models/*' folder
    *[(os.path.join('share', NOMBRE_DEL_PAQUETE, os.path.split(p)[0]), [p])
      for p in glob(os.path.join('models', '**', '*.*'), recursive=True)]
  ],
  # ...
)
```

Para insertar el modelo dentro del mundo de *Gazebo*:

Listado 2 `gazebo.launch.py`

```
from launch import LaunchDescription
from launch.actions import
  IncludeLaunchDescription, AppendEnvironmentVariable
from launch.substitutions import PathJoinSubstitution
from launch.launch_description_sources import
  PythonLaunchDescriptionSource
from launch_ros.actions import Node
from launch_ros.substitutions import FindPackageShare
# ...

NOMBRE_DEL_PAQUETE = # Completar con el nombre del paquete

def generate_launch_description():
  gz_model_path = AppendEnvironmentVariable(
    'GZ_SIM_RESOURCE_PATH',
    PathJoinSubstitution([
      FindPackageShare(NOMBRE_DEL_PAQUETE), "models"
    ]),
  )

  # Launch Gazebo
  gz_sim = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
      PathJoinSubstitution(
        [FindPackageShare('ros_gz_sim'), 'launch', 'gz_sim.launch.py']
      ),
    ),
    launch_arguments={
      'gz_args': '-r empty.sdf',
    }.items()
  )

  load_warehouse = Node(
    package="ros_gz_sim",
    executable="create",
    arguments=[
      "-entity", "warehouse",
      "-file", "model://Warehouse",
    ],
    output="screen",
  )

  return LaunchDescription([
    gz_model_path,
    gz_sim,
    load_warehouse,
  ])
```