Exercise 4: ROS Services

With this exercise, we introduce you to the environment (robot, world, launchers,...) that you will use for the assignments. Thus, it is important that you follow these instructions now, so that you won't be unprepared once we start with the exams:

1. Initialize a dedicated workspace:
   ‘’’bash
   ```
   mkdir ws_$(group_number)_assignments
   cd ws_$(group_number)_assignments
   mkdir src
   cd src
   git clone https://github.com/PieroSimonet/ir_2526.git
   cd ..
   colcon build
   source install/setup.bash
   ```
   “”
2. Create a dedicated package for this exercise and for each future assignment
   “”
   ```
   cd src
   ros2 pkg create --build-type ament_cmake --license Apache-2.0 $(group_number)_ex4
   ros2 pkg create --build-type ament_cmake --license Apache-2.0 $(group_number)_assignament_1
   ros2 pkg create --build-type ament_cmake --license Apache-2.0 $(group_number)_assignament_2
   ```
   “
3. **ONLY MODIFY THE FILES IN YOUR PACKAGES, DO NOT EDIT OURS IN ir_2526 !!!!!!! (You will fail the exams otherwise)**

4. You will need to create a git repository for each assignment. More on this in the assignment instructions.

5. Launch the initialization of the exercise with:
   ‘’’bash
   ```
   ros2 launch ir_launch exercise_4.launch.py
   ```
   “”

6. Tip: Write your own launch file that automatically calls the initialization launch file.

In this exercise, you will deal with the turtlebot Robot and its simulator. The turtle can communicate with its burrow and get updates about the state of its resources. The turtle can use its LIDAR to find new resources.

In this exercise:

- Develop a node representing the burrow and one node representing the turtlebot. The two **must** communicate **by means of a service**.
- The burrow node **(client)** sends a **request** to the turtlebot node**.**
- The request indicates the state of its resources (e.g. n apples) and the size of the burrow itself (s). Randomly generate these two values so that n<s.
- The turtlebot node **(server)** subscribes to the lidar topic and keeps checking the number of apples (in the simulation, apples are represented as spheres) in its surroundings.
- When the request arrives, the turtlebot sends a **response** True if it can find enough apples to refill the burrow in its whole size s, False when apples are not enough.
- The turtle must **publish on a topic /apples** the poses (geometry_msgs/msg/PoseArray) of the apples found relative to base_link.

Build the code so that multiple scenarios of the resources state are explored (n < s, n=s, enough/not enough apples found).

All the phases of the communication must be shown to the user by printing info in the terminal window.

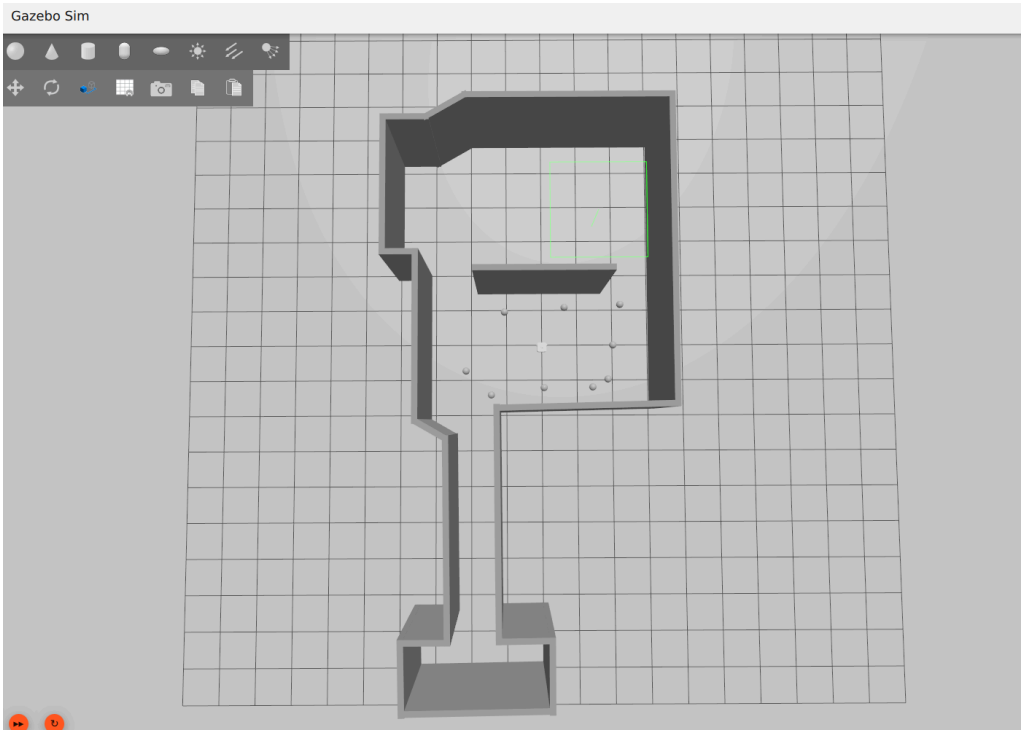

Figure 1: This is the environment and the robot that you should see when launching the simulation.

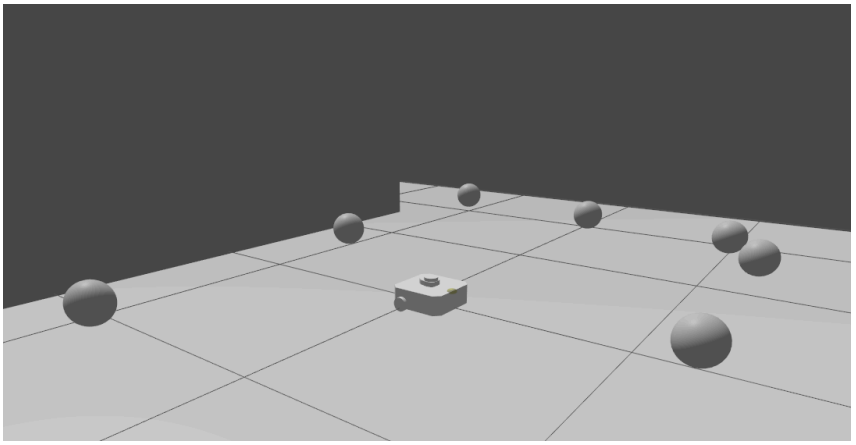

Figure 2: This is the turtlebot, surrounded by apples.

Question:
How do you recognize the presence of an apple? Are there any geometrical properties that you can use in order to estimate their position relative to the turlebot?