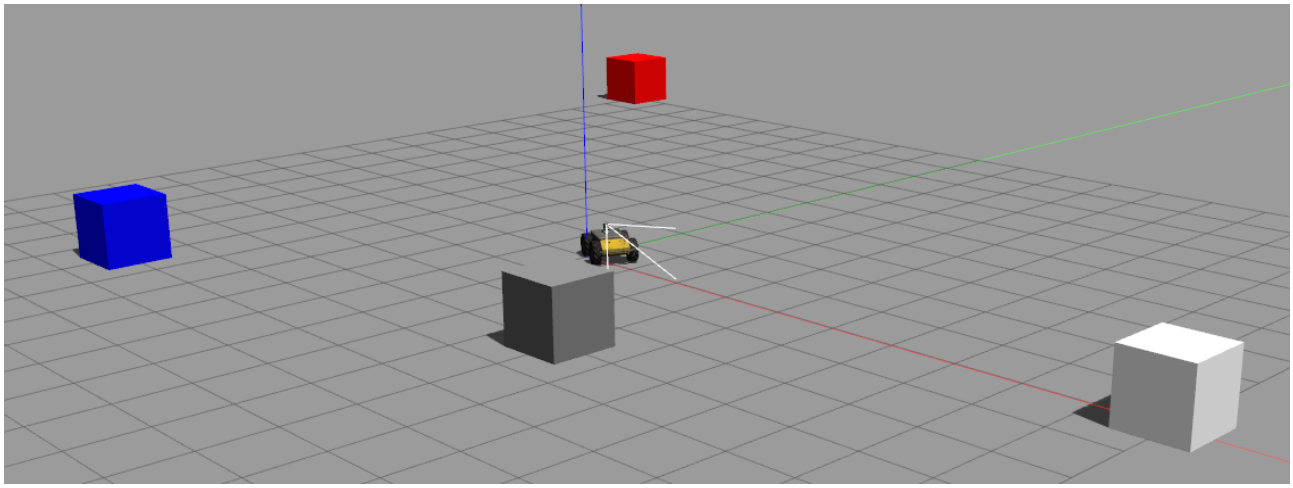
Coding task

description:

The starting point (what is given to you) is a simulated gazebo world with a robot and four boxes. Each of them represents a different type of object: the blue box is a car, the red box a bicycle, the grey box a cat and the white box a dog.



Each of the boxes can emit a sound corresponding to the type of object they represent. Just one object will emit sound at each time. Lets call it the active object. The goal of the task is to implement a controller (in a ros node) that makes the robot approach to or escape from the active object depending on its type. The robot will approach the car and the bicycle and escape from the cat and the dog.

preparatory steps:

- Install ROS in your machine
- Clone the provided catkin workspace at https://github.com/EloyRC/catkin_hbp and build it running catkin_make from the root folder (it has been tested in Ubuntu 20.04 / ROS noetic)
- Add these lines to your .bashrc file and source it:

```
source /opt/ros/noetic/setup.bash
export CATKIN_WS=/replace/with/the/path/to/catkin_hbp
source $CATKIN_WS/devel/setup.bash
export GAZEBO_MODEL_PATH=$CATKIN_WS/src/gazebo_environment/models
```

- You can check that everything is ok by running:

```
roslaunch gazebo_environment hbp_husky.launch
```

gazebo should start correctly and running:

```
rostopic echo /sounds
```

in another terminal should output messages of the type:

```
sound_file:
  data: "car"
source:
  x: 5.0
  y: -5.0
  z: 0.0
```

task instructions:

Information about the active object is provided via the topic */sounds* of type *gazebo_msgs2/SoundWithSource* (defined in a msg file in *gazebo_msgs2* folder). The msgs contain the pose of the active object (source) and a string representing the produced sound (sound_file).

Information about the robot pose is provided via the topic */odom* of type *nav_msgs/Odometry* (ROS defined message).

The robot speed is controlled by publishing to topic */husky/cmd_vel* of type *geometry_msgs/Twist* (ROS defined message)

The implemented ROS node will have to subscribe to */sounds* topic and infer the type of the active object by using the audio file in folder *audio_files* with the same name as *sound_file* string. Ie. to classify the sound as car, bicycle, cat or dog (eg. using a neural network).

Finally, the object and robot pose will be used to compute a proper twist which implements the aforementioned behavior: moving towards the active object if it is a car or a bicycle and against it if it is a cat or a dog.

The computed twist will be published to */husky/cmd_vel* topic. This process will be repeated for every msg coming through */sounds* topic

Assessment criteria:

Completing the task is of course desirable but more important is clarity and quality of code, showing, when possible, good design principle and knowledge about the chosen programming language (you can choose between C++ or python according to your preference). For example it will be valued if the implemented solution is easily extensible to new objects/sounds or new robot behaviors.

If some parts of the task are too complex for you, you can apply some simplifications. For example, you can avoid the inference of the active object type using the audio file and use directly the *sound_file* string. Or implement a simplified robot behavior, eg. for car and bicycle the robot moves forwards and for cat and dog moves backwards. But maximum value will be given to implementations solving the task as in the full description.

Submitting the task:

you can implement your solution in a separate branch of the repository and, after completion, open a pull request, which I will receive. You can additionally send an email to me.