BOAT CONTROLLER STRATEGY: TARGET DETECTION AND OBSTACLES AVOIDANCE WITH RGB-D CAMERA

MOBILE ROBOTICS PROJECT

Emanuele Venturelli - VR509826

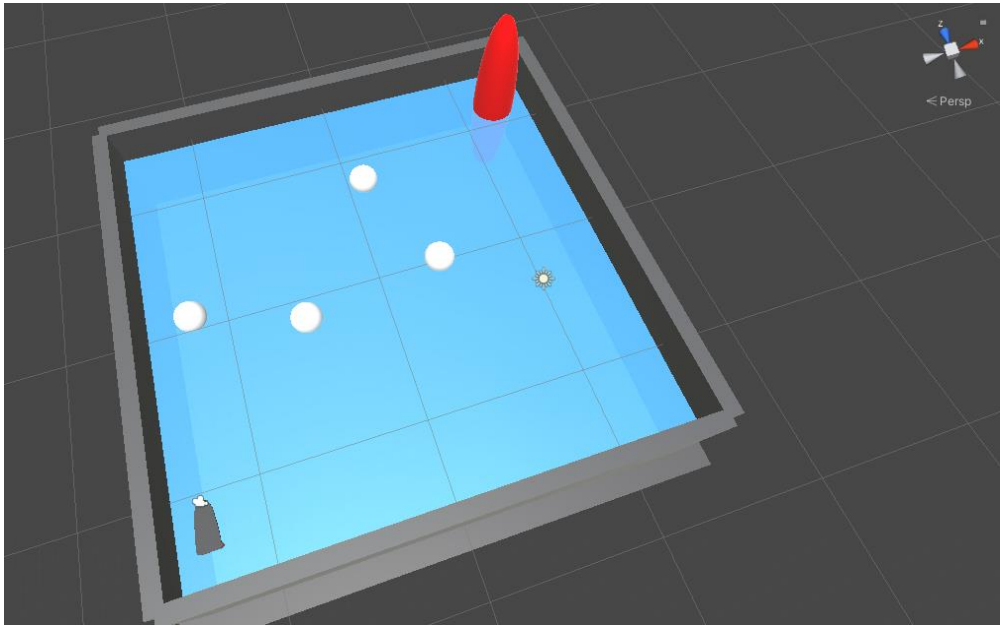
TARGET FOLLOWING AND OBSTACLE AVOIDANCE FOR ROBOTIC BOAT

This code implements the control of a robotic boat for detecting a red target and avoiding obstacles. The structure of this presentation will be:

- Main components of the scene
- Sensor and add-ons used
- RGB camera and target detection
- Depth camera and obstacles avoidance

MAIN COMPONENTS OF THE SCENE

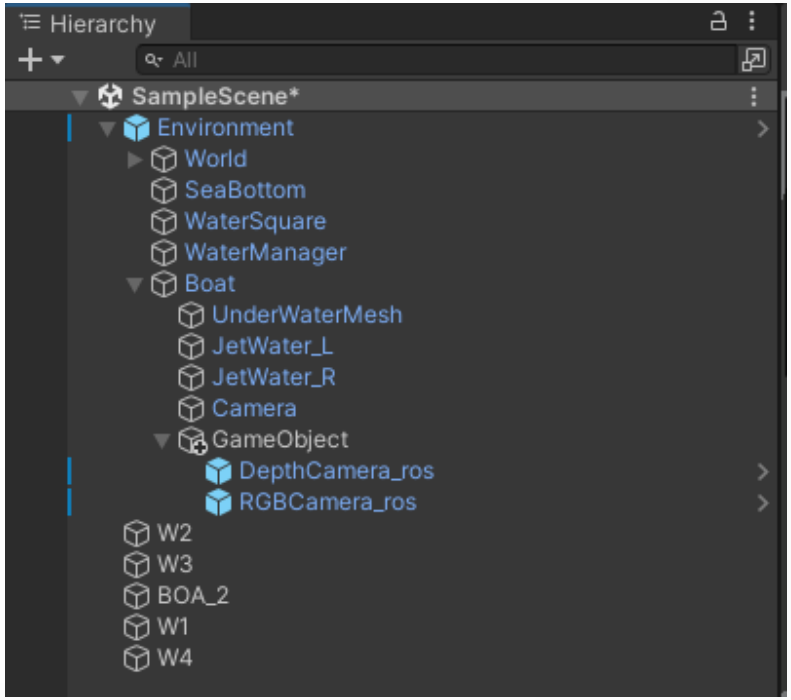
- Boat
- Obstacles with spherical shape (buoys)
- Red target



Snapshot of the scene

SENSORS & ADD-ONS

- RGB Camera: to detect the target
- Depth Camera (PointCloud2): to detect obstacles
- To retrieve the data from the sensors and to analyze them in ROS2 we use two packages:
  - ROS-TCP-Connector
  - ROS-TCP-Endpoint



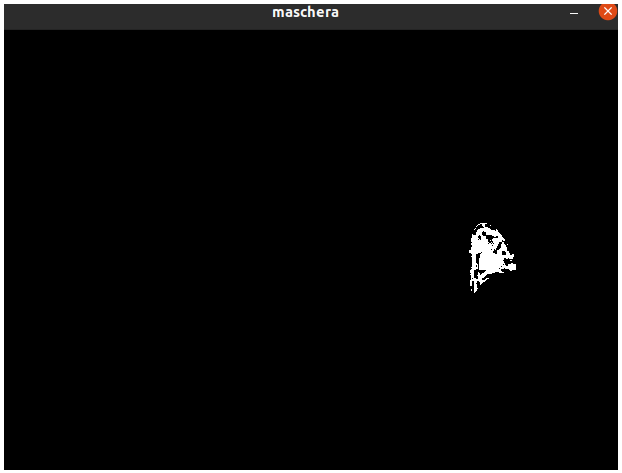
Structure of the hierarchy

RGB CAMERA

- The RGB camera is mounted in front of the boat, and it is used to detect the target
- The camera collects images of size 640*480, with a frequency of 10 fps
- OpenCV library is used to detect a red object, which is our target
- The topic subscribed by the RGB camera is: `/camera/color/image/compressed`

RGB CAMERA – TARGET DETECTION

- The image obtained from the RGB camera is converted into HSV format with OpenCV library
- A mask that collects all the pixels inside a certain color range is created



Snapshot of the mask taken from the RGB camera

HSV color model

RGB CAMERA – TARGET DETECTION

- The range is fixed based on the color you want to find, in our case RED
- We compute the moment of the image and the coordinate of the centroid along X axis (which corresponds to the center of our target)
- Knowing the X coordinate of the centroid, we compute the target offset with respect to the center of the image. Then, if the module of offset is higher than an certain threshold, the robot will:
  - Turn right if the target is on the right, giving more power to the left motor and less power to the right motor
  - Turn left if the target is on the left, giving more power to the right motor and less power to the left motor

```
min_distance > SECURITY_DISTANCE and abs(self.target_offset) > 0.03:
left_motor.data = FORWARD_SPEED + 0.06 * self.target_offset # turning towards the target
right_motor.data = FORWARD_SPEED - 0.06 * self.target_offset
```

DEPTH CAMERA

- The depth camera is mounted in front of the boat, together with the RGB camera, and it is used to detect the obstacles (buoys)
- The sampling frequency is equal to 10 fps
- The topic subscribed by the depth camera is: `/camera/depth/points`
- From this sensor we collect pointcloud data and we filter them to obtain only the closest points in front of the boat

```
# filter the points in front of the robot
front_points = points[
(np.abs(points[:, 1]) < 0.5) & # filter lateral points, where index 1 is equal to y axis
(points[:, depth_index] > 0.1) & # do not consider points to close
(points[:, depth_index] < 10.0)] # do not consider points to far
```

DEPTH CAMERA - OBSTACLES AVOIDANCE

- After filtering the pointcloud, we compute the closest point to the boat and we calculate on which side of the boat it is and its magnitude
- If it is negative it will be on the left side of the boat, otherwise, if it is positive, it will be on the right side
- If the point is closer than a certain threshold, called **SECURITY_DISTANCE**, the boat will start the 'obstacles avoidance' procedure

```
front_points_min = front_points[pos_min_distance, 1]

elif min_distance <= SECURITY_DISTANCE:
if float(front_points[pos_min_distance, 1]) >= 0:
self.get_logger().warn(f"Obstacle found at: {min_distance:.2f}m! Going right...")
else:
self.get_logger().warn(f"Obstacle found at: {min_distance:.2f}m! Going left...")
left_motor.data = FORWARD_SPEED + 0.025 * (1.2 - float(front_points_min)) * np.sign(float(front_points_min))
right_motor.data = FORWARD_SPEED - 0.025 * (1.2 - float(front_points_min)) * np.sign(float(front_points_min))
```

DEPTH CAMERA - TARGET REACHED

- When the boat reaches the target, it stops, and the motors have to stabilize it in order to keep the yaw angle under a certain threshold (using the X centroid position)
- The boat stops when:
  - the percentage of area of the mask retrieved from the RGB camera covered by the target is over than 2.8%
  - The closest point is in front of the boat
- If no target is detected, the boat rotates 360° looking for one

CONCLUSION

The code allows the robotic boat to:

- Follow a red target using an RGB camera
- Avoid obstacles using the depth camera
- Stop when the target is reached
- Stabilize itself to keep the yaw angle low once the target is reached