Robobo: Aruco Boxes

Intelligent Robotics 1 - Practical Work

Andrés Nó Gómez, Pablo Páramo Telle

1 Morphology of the robot

The robot used for the development of this work is Robobo, consisting of a mobile platform and a smartphone attached to it, which contributes with its detection and actuation capabilities as well as its processing power.

In the simulation of the task, we additionally have a platform attached to the front part of the robot, which is utilized for transporting boxes. The loading of a box is performed automatically when the robot approaches it closely enough, while the unloading is also done automatically when the robot positions itself above the corresponding container.

However, in a real-life implementation, a more complex mechanism would be required to enable loading and unloading functions, such as a pair of robotic grippers.

2 Sensors and Actuators

The robot is equipped with infrared sensors located on the front part of the platform. These sensors are used to measure distances to objects in the environment, such as boxes and walls, allowing the robot to navigate and avoid obstacles. Additionally, the robot utilizes a smartphone camera to detect Aruco codes, which are used for localization and identifying targets within its environment.

The robot's movement is powered by wheel motors, which enable it to move in various directions thanks to their independent operation. This provides the robot with the flexibility to navigate around obstacles and reach its targets. Another actuator is the motor responsible for the pan movement of the smartphone, allowing it to adjust the camera's orientation. The smartphone also has a speaker, which is used to emit sounds or phrases when the robot achieves certain objectives, providing feedback to the user. Finally, the robot's smartphone screen is used to change its facial expression based on the situation, adding a layer of interaction and communication.

3 Description of the Environment and Its Variations

The Aruco Boxes map integrated into the Robobo simulator consists of a square area with a box spawn zone and three zones for depositing them. Each box has a corresponding container, but if a box without an assigned container appears, there is a fourth zone where it can be discarded. Additionally, in this project, this fourth zone has also been considered as a battery charging area.

The correspondence between boxes and containers is predefined according to the IDs of the Aruco codes present both on the faces of the boxes and on the walls above the deposit zones.

The variations in the environment include:

- Box Spawn: Although the boxes always appear in the same area, they will not always have the same Aruco codes, nor will the number of boxes remain the same.
- Containers: While the three main deposit zones are always in the same location with the same Aruco codes, the relative positions of the codes may change. The discard/charging zone is always in the same place with the same Aruco code.

4 Implemented behaviors

We will present the behaviors following the hierarchical order of the subsumed architecture, from the lowest to the highest level. In this approach, the higher levels subsume the lower ones, meaning they take control by inhibiting the transmission of sensory stimuli and replacing their actions. For example, if the robot does not have a box loaded, the priority will be to search for a box instead of searching for a container. In this way, priorities are clearly established between individual tasks, ensuring that the most urgent or relevant tasks are executed first. This hierarchical system allows the behaviors to be combined effectively, resulting in the robot's overall behavior, adapted to the conditions and needs of the environment.

4.1 Search box

The robot moves randomly around the map using a simple strategy: it moves in a straight line until it detects a wall, at which point it turns left to continue its search. This strategy is sufficient because the map is square-shaped, allowing the robot to navigate efficiently without the need for complex planning. If the environment were more complex, with irregular obstacles or dynamic conditions, more advanced search strategies could be implemented. However, in this case, this straightforward approach is enough to locate a box.

The behavior ends when the robot detects an Aruco associated with a box.

4.2 Go to box

Once the Aruco code of a box is detected, this behavior is activated to approach and load the box onto the platform.

To ensure that the robot doesn't deviate while moving toward the box, it repeats the following two steps until it reaches the box: First, it rotates to center the Aruco code in the camera's view, and then it moves straight for a predefined duration, which depends on the distance to the Aruco.

When the robot is very close to the box, it moves slowly for the necessary distance to precisely pick up the box. Once the platform makes contact with the box, it will automatically be catched, and the behavior will end.

4.3 Search container

Once a box is placed on the platform, this behavior is activated to find the corresponding container.

Its operation is similar to *SearchBox*: the robot moves around the map and the behavior ends when the Aruco code of the container associated with the loaded box is detected.

4.4 Go to container

If a box is loaded onto the platform and the Aruco code of the corresponding container has been detected, this behavior directs the robot toward the container to deposit the box.

Its process is analogous to *GoToBox*: it centers the Aruco code in the middle of the camera's view and moves straight toward it repeatedly. Once the robot is close enough, it makes a final move to position itself over the colored area, where the box is unloaded. At this point, the behavior ends.

4.5 Search charge zone

Throughout the entire process of loading and unloading boxes, battery consumption is simulated in a separate thread with a discharge rate of 1% per second. When the battery drops below the 30% threshold, this behavior is triggered to search for the charging area.

Since the battery is crucial for the agent's operation, it interrupts any action the robot is performing, except for the two key last moments in *GoToBox* and *GoToContainer* when the boxes are being loaded or unloaded. This is done to avoid situations where the robot behaves as if it has a box loaded when it does not, and vice versa.

Like *SearchBox* and *SearchContainer*, it moves around the map, avoiding walls, until it detects the Aruco code associated with the battery charging area.

4.6 Go to charge zone

Once the charging zone is located, this behavior activates to move toward it.

As previously explained in *GoToBox* and *GoToContainer*, the robot approaches in short straight-line segments, correcting its orientation after each segment until it is sufficiently close to the charging zone. Then, it makes a final movement forward, positioning itself within the charging zone, where it remains stationary for 7 seconds. The battery is restored to 100%, and the behavior ends.

4.7 Escape

This behavior has the highest priority, as it is activated when the robot is about to collide with a wall. By taking control immediately, it overrides all other behaviors and anticipates the potential impact, moving the robot backward and toward the opposite side of the wall. This way, it prevents both getting stuck in a corner and a possible collision.

5 Results

After several tests, it has been confirmed that the robot performs the task satisfactorily, demonstrating the efficient operation of the designed architecture.

The robot accurately locates the target boxes and moves toward them to pick them up. Despite not having a specific sensor to detect whether the box has been picked up, a method has been developed that positions the robot directly in front of the box at a very close distance to simulate the pickup process. This method has proven to work effectively, reliably detecting whether a box is picked up in all the tests conducted. Then, it reduces its speed while carrying a box and begins searching for the delivery container, toward which it moves once located to deposit the box.

The behavior for locating the charging area is activated when the battery level drops below the established threshold. During this process, the robot adjusts its speed depending on whether it is carrying a box to ensure careful transport if necessary. Once the charging area is identified, the robot goes to it and positions itself over it to recharge the battery.

Additionally, the collision detection behavior, called *Escape*, has proven effective. It activates when the robot gets too close to a wall, taking priority over other behaviors and allowing the robot to free itself from the blocked situation.

In conclusion, the subsumption architecture has proven to be an efficient solution for this task, selecting the most appropriate behavior based on real-time sensory information and effectively prioritizing actions.

6 Existing problems

Since the aruco detection system can only identify one aruco at a time, sometimes, when heading toward a container, the robot loses track of the current target when it detects a neighboring container. This behavior is reflected in the terminal when it occurs, as the robot cancels the *GoToContainer* behavior and reactivates the *SearchContainer* behavior, issuing a warning indicating that it has lost the container it was heading toward.

In spite of this, it is not a critical issue, since the *SearchContainer* behavior moves in a straight line, the previously detected container is identified again as the robot gets closer, as it becomes centered in the camera view.

However, we wanted to point this out as an inconvenience, as it causes unexpected transitions between behaviors.