Student Name: Jessica Short

Student ID: 010500487

Course: C951 – Introduction to Artificial Intelligence

Assignment: Task 2

Date: November 3, 2024

PART A

Describe the disaster recovery environment you chose and the two obstacles you have added to the environment.

The disaster environment that I chose is a house engulfed in flames with a person trapped inside. I added a red cylinder to represent a person. I also added 4 long, purple cuboids to represent the walls of the house, 1 pink cuboid to represent a bed, 1 white cuboid to represent a dining table, and 4 white cylinders to represent dining chairs. All the objects in my environment are collidable, measurable, and detectable. My robot, LifeBot, is designed to locate humans that need immediate assistance and to promptly communicate their existence and location to rescue personnel.

PART B

Explain how the robot will improve disaster recovery in the environment from part A after you have added the two obstacles from part A.

LifeBot assists search-and-rescue teams during a fire emergency. The robot can locate any human trapped inside a burning building and send an alert, including the person's location, to rescue personnel. Rescue workers will be able to focus their efforts on removing survivors from burning buildings and delivering them to safety. The use of LifeBot will significantly improve the efficiency of rescue efforts by eliminating the need to enter and search burning homes that are empty. The use of this robot will also reduce the risk of injury for the rescue workers, since the time spent inside burning buildings will be significantly reduced. In the event of a large fire that has taken over several homes or buildings, multiple LifeBots can be deployed.

The navigation sensor, sensingNose, can detect objects in a building and allow LifeBot to move around those obstructions. This prevents the robot from getting damaged or stuck. LifeBot can navigate around furniture, including those described in part A: walls, a bed, a dining table, and chairs.

PART C

Justify the modifications you made to CoppeliaSim's robot architecture, including two sensors you chose to add, and explain how these sensors will aid the disaster recovery effort.

In the creation of my robot, I modified CoppeliaSim's "bubbleRob" from in their tutorial files. First, I modified the proximity sensor, sensingNose, to improve the robot's mobility. LifeBot uses the sensingNose to navigate; if sensingNose detects something in front of the robot within its detection volume, it will back up, rotate to the left, and then continue moving forward again. I modified sensingNose's detection area to be a small cone with a range of only 0.1 meters. This allows the robot to be able to get into tight spaces. This is a crucial feature since a frightened child trapped inside of a burning house could very likely be hiding behind an item of furniture.

I also added a second blue proximity sensor to detect people, sensingNose_To_Detect. This can be differentiated from sensingNose because sensingNose is yellow. sensingNose_To_Detect turns from blue to green and outputs the message "I found a human!" when a person is detected inside the house. This message will be sent along with the exact location of the human in distress to nearby rescue workers. I made this sensor larger than the navigation sensor, with a range of 0.4 meters, so that it can detect people faster.

PART D

Describe how the robot maintains an internal representation of the environment.

This robot has a reactive view of its environment, rather than a persistent map. LifeBot uses data from its proximity sensor to create an internal representation of its immediate environment. The robot's proximity senor, sensingNose, provides distance measurements to nearby objects. When sensingNose detects an object, the robot switches to backward mode and reduces its speed to avoid obstructions.

The user can see where the robot has been via the yellow path. This path, named robotTrace in the code, is created using the sim.addDrawingObject function, which defines the path as a line strip. This is for the user only; LifeBot does not use robotTrace for navigation.

The function sysCall_sensing measures the distance between the robot and objects in the environment and updates distanceSegment and robotTrace.

These distance measurements to nearby objects can be viewed in the graph steam that pops up when you run the simulation in CoppeliaSim. The graph stream displays bubbleRob_clearance in meters on the y- axis and time in seconds on the x-axis.

backUntilTime monitors whether LifeBot is operating in forward mode or backward mode. If an obstacle is detected, backUntilTime is set to four seconds in the future and activates backward mode. If the current time is less than backUntilTime, the robot moves backwards at reduced speed. If the current time exceeds backUntilTime, the robot moves forward at the specified speed.

The robot's speed can be adjusted by the user via the speed control slider. Initially, the speed is set at the midpoint between minMaxSpeed[1] and minMaxSpeed[2]. The function speedChage_callback adjusts the speed based on the UI slider.

The function sysCall_actutation contains all the logic for the proximity sensor reaction, movement control, and human detection.

PART E

Knowledge: LifeBot collects information about its immediate environment. The robot uses two proximity sensors, one to detect obstacles in its path, and one to detect humans. LifeBot uses this information to navigate around a building and to quickly alert rescue personnel to the location of a person inside a burning building.

Reasoning:

The robot uses the input from its two proximity sensors to make decisions about its direction of movement, speed, and whether to print a message to the console log. If an object is within the detection volume of sensingNose, the robot is programmed to switch to backup mode and move backwards at a slower calculated speed for four seconds. After four seconds, if the path is clear, the robot will move forward again until another obstacle is detected.

LifeBot also makes decisions from the input it receives from sensingNose_To_Detect. The proximity sensor sensingNose_To_Detect is programmed to turn from blue to green and print "I found a human!" to the console when a human is detected. When no human is detected, nothing is printed to the console and the detection volume color remains blue.

Uncertainty:

LifeBot is a great example of an application of uncertainty. Its navigation is reactive, rather than predictive. This robot explores unknown environments by reacting to its immediate surroundings via the use of proximity sensors.

Intelligence:

LifeBot demonstrates intelligence through preforming object detection, object avoidance, mode switching, and simple object recognition. This robot can maneuver through a burning house via the use of a proximity sensor and logic. When the proximity sensor detects an object in the robot's path, the logic switches the robot to backward mode for four seconds. After four seconds, if the path is clear, the logic robot switches back to forward mode.

These capabilities allow LifeBot to successfully alert rescue teams to the locations of humans in distress. This means that people will receive the medical care they need faster and more efficiently. LifeBot will also reduce the injury to fire fighters and other rescue workers, by minimizing the amount of time spent inside burning buildings.

PART F

Explain how the prototype could be further improved, including how reinforced learning and advanced search algorithms can improve the prototype's performance and learning.

This robot protype could be improved by using reinforced learning and advanced search algorithms.

LifeBot could be improved with reinforced learning (RL) in the following ways:

- Using a rewards structure would encourage the robot to act in ways that are safe, efficient, and focused on accomplishing its objectives. The robot would have positive rewards for desired behaviors such as: navigating unexplored areas of a building, avoiding objects, and moving towards people. The robot would have negative rewards for undesired behaviors such as: colliding with objects, becoming immobilized, or revisiting the same areas.
- The robot could learn more advanced obstacle avoidance methods. For example, based on its experience with similar objects, it could experiment with different turning directions, backup angles, and backup mode durations. Because the disaster environment is a bedroom, the objects would mostly be furniture items and would have similar properties and sizes.

- The UI slider could be removed and the robot could learn to adjust its speed dynamically. When it's close to obstacles, the robot could reduce its speed. When the robot is in obstacle-free areas, the robot could increase its speed. This would reduce the occurrence of collisions and positively affect task completion times.

LifeBot could be improved with advanced search algorithms in the following ways:

- The A* algorithm could be implemented. The A* algorithm is an informed search algorithm that finds the shortest path between two points considering obstacles. It is one of the most popular pathfinding algorithms and is used in GPS navigation, pathfinding in video games, and robotics (Source: Kattilaxman, 2023).
- Another algorithm that could be utilized for LifeBot is Greedy Best-First Search. This algorithm can be used to find the most promising path to a target while avoiding obstacles. Greedy Best-First Search is ideal for applications where speed is critical, which makes it an appealing option for LifeBot. Applications of Greedy Best-First Search include video games, robotics, navigation systems, and image processing. (Source: GeeksforGeeks, 2024).

<u>PART G</u>

Please see submission attachments for the code for my robot.

<u>PART H</u>

Please see my submission for my Panopto video recording link.

<u>PART I</u>

The sources that I referenced/used for this project are listed below:

CoppeliaSim
- LifeBot is based off of Coppelia Robotics's bubbleRob (LUA) robot. bubbleRob can be found in the tutorial files of the current edition of CoppeliaSim Edu.
- Date Accessed: 11/3/2024
- Website: https://www.coppeliarobotics.com/

The A* Algorithm Explained
- Author: Kattilaxman for Medium
- Date: 10/30/2023
- Date Accessed: 11/3/2024

- Website: https://medium.com/@kattilaxman4/the-a-algorithm-explained-9cdbce1c8f3c

Greedy-Best-First Search Algorithm

- Author: GeeksforGeeks
- Date: 01/18/2024
- Date Accessed: 11/3/2024
- Website: https://www.geeksforgeeks.org/greedy-best-first-search-algorithm/