1. Description of the Disaster Recovery Environment

The disaster recovery environment is designed to represent an area that is on fire. The columns that make up the barrier for the bot are considered inaccessible flame areas. In the layout, the column placement prevents the bot from exiting the closed-space because those areas are entirely engulfed in flames. Whereas the columns inside the area represent potential sections that are dangerous, and that the robot should avoid contact with. The cube represents a person who is trapped. The robot's goal is to find the person and send a notification to the console window. The bot is equipped with two additional sensors that detect the fire columns and stops before touching them. This helps the bot navigate safely through the environment and find a path. The bot is also equipped with a full range sensor to detect the person. When this sensor is tripped, it will send the output, "A person is found!" to the console. A second obstacle is an area in the middle of the map where the person is located. This area represents a building that is also on fire. The bot must maneuver around the building to find the person in most cases.

1. Improved Disaster Recovery

During a fire, the environment is very unpredictable and dangerous for both those caught up in the fire and the associated rescue teams. Having the disaster recovery bot is very helpful because it can create a map of the environment and prevent unnecessary risk for the rescue crews. The robot can map its movements and sensor data to let rescuers know where it chose not to go. Additionally, people who are found in danger can be identified and communicated with through the robot's cameras and sensors. This can be a massive help to people trapped because they know support is on the way. The bot also contributes to improving safety for rescue workers. In a fire, the environment can change rapidly, and rescuers could become trapped as well. If the robot is stuck, it can relay its location so that rescuers know that it is not safe. While there is a risk of the robot being burned or destroyed, it can help improve safety through its actions and prevent additional loss of human lives. Lastly, the camera on the bot can give rescuers first-hand information on the severity of the fire. It may be challenging to determine the structural soundness of a building that is on fire. Sending in a camera can allow experts to identify any people still in the building.

1. Sensor Modifications

To make the bubbleRob model applicable to a disaster recovery setting, we can have to expand the bot's existing tools. We started by adding two proximity sensors on the front of the bot. These additional sensors are modeled after the original sensor but do not include a vision sensor. Next, we had to adjust the Lua code to reduce the turning penalty caused by the activation of each sensor. The original turning mechanism was too severe and caused the bot to sometimes go in circles. This change resulted in a much smoother and fast experience for the bot. At this point, we had a bot that was able to safely navigate the environment without risk of driving into flames or getting stuck. Next, we had to address the issue of identifying people in the background. For this, we added a wide-angle sensor that could only detect the cube. This sensor has a more extensive range than the other three proximity sensors and made it easier to identify the cube during movement operations.

1. Optimization Principles

The use of optimization can be most clearly seen in how the bot's existing tools were adapted to better fit the use case of disaster recovery in a fire. The additional sensors and reduced penalty for tripping a sensor is a perfect example of use case optimization. The bot can reason through most environments by going in a straight line, if possible. Once it runs into something, it can use gained intelligence of its surroundings to adjust course. Additionally, the bot can create knowledge and adapt the environment by discovering blocks and changing their color. This also shows the bot's ability to reason between a human fire, thus proving its ability to turn content into knowledge representation. Lastly, the code, training, sensors, and knowledge gained from exploring the environment allow the bot to operate with some uncertainty. It can continue to work even though it is not aware of its entire ecosystem.

1. Advantages – Limitations – Criteria for Success

Benefits are provided above; however, we offer them here again in bullet point format for brevity and clarity:

* The robot can complete the simulation without damaging any people or itself
* The robot can create a map of the environment and prevent unnecessary risk for the rescue crews
* People who are found in danger can be identified and communicated with through the robot's cameras and sensors
* Use of the robot can help prevent unnecessary loss of human life and reduce the danger of rescuers
* The robot can successfully go through the simulated environment and create helpful knowledge
* The robot can identify and people it comes across

There are also many disadvantages; we outline them below:

* There is a possibility that the robot will back into the fire or malfunction and go off the edge
* It may not detect every person in the environment
* It could continue the simulation infinitely without ever reaching a specific part of the environment
* The robot is not aware of the places it has already visited and is prone to covering its tracks multiple times
* It does not have the ability to extinguish flames
* The robot is not aware of the additional space on the board. It just assumes there is nothing else because it detects flames blocking its path

Success factors for the prototype involves a mix of concrete and abstract goals. We want the bot to be able to explore the environment without getting stuck or falling off the map. The bot must run long enough where we feel comfortable knowing that it has identify most people. We must also give it enough time to work its way out of a situation. If the bot is going in a circle for more than a few minutes than we should abandon the simulation and adjust accordingly. We can determine its success based on the bot's ability to identify every person in the environment. However, this is not practical in a real-world scenario. Therefore, we can also determine the success of the bot by ensuring all the factors mentioned above are indeed correct.

1. Improvements and Reinforcement Learning

Our bot is very simplistic and therefore limited by many constraints. In this section, we outline areas of development and describe what is needed for our bot to work in a real-world environment. Our bot could benefit from machine learning practices such as unsupervised learning and reinforcement learning. In our simulation, we can observe the bot's actions and make supervised changes to ensure its success. In a real-world scenario, our environmental knowledge is limited, and we cannot restart the simulation. Any mistake the bot makes can result in severe consequences. Additionally, there is no feedback loop in place to award the bot for avoiding danger. Employing a rewards system for the bot would be the most necessary improvement. To achieve such development, we would need our bot to identify as a Q-learning based agent. Such an agent can compare expected utility values of its actions and choose the best option without knowing its environment. This agent is likely better equipped to handle real-world situations because it does not need to know the entire ecosystem. In our current simulation, it is likely that a type of utility-based agent would perform the best. This is because "A **utility-based agent**learns a utility function on states and uses it to select actions that maximize the expected outcome utility." We can tell the bot that its expected goal is to identify every person in the environment and avoid getting near the flames. This training would be possible by using reinforcement learning because we cannot adequately supervise the robot in the real-world.

Another improvement we can expand on is using technology that will allow the bot to keep track of everywhere it has been so that it does not cover the same area more than once. This would optimize the performance of the bot significantly. Additionally, we could add more sensors so that the bot was always aware of its entire surroundings. Such a function would allow us to let the bot turn dynamically and directional based on observed information.

Sources

Russell, S. J., & Norvig, P. (n.d.). Artificial Intelligence: A Modern Approach, Ch. 21: Reinforcement Learning, Introduction. Retrieved January 28, 2020, from https://wgu.ucertify.com/?func=ebook&chapter\_no=23#top