Introduction to Artificial Intelligence (C951)

Performance Assessment Task 2  
**“DISASTER RELIEF ROBOT”**

Daniela Vidal

Western Governors University

WGU student ID# 001172091



Contents

[A. Problem 3](#_Toc40624329)

[B. Improved disaster recovery 3](#_Toc40624330)

[C. Architecture 3](#_Toc40624331)

[D. Goal seeking 4](#_Toc40624332)

[E. Advantages and limitations 4](#_Toc40624333)

[F. Testing and implementation plan 5](#_Toc40624334)

[G. Improving the prototype 6](#_Toc40624335)

[E. Robot Code 6](#_Toc40624336)

[F. Panopto Recording 7](#_Toc40624337)

[G. Sources 7](#_Toc40624338)

# A. Problem: description the disaster recovery environment with two more additional obstacles.

Robots are physical agents that perform tasks by manipulating the physical world (Russell S. J., Introduction, 2010), which makes them useful for tasks that humans may not be able to perform. The prototype represented in this work is precisely doing that, accessing a dangerous environment to perform a task that can’t be done by humans due to inaccessibility or danger.

The environment on which BubbleRob starts represents a building after an earthquake has hit the area. The building in the layout is a closed space inaccessible for humans at the time, but there is a very important object somewhere between the scattered debris of the earthquake and it needs to be located. The cylinders represent the limits of the building, which makes it a closed space, but they also may represent scattered debris on the floor. The rescue team sends this robot inside to recover the object (represented by a cube), once the robot recovers it, it sends a signal “Object found”. For this task, the robot is equipped with 2 extra sensors, one specifically for the location of the stated object, and the other to avoid hitting objects when going backward.

# B. Improved disaster recovery: explanation of how the robot will improve disaster recovery in an environment with two or more additional obstacles.

After an earthquake, the environment is unpredictable (due to aftershocks or falling pieces of debris) and therefore it is unethical to expose any human being to such conditions and uncertainty, that’s why a disaster recovering robot is so helpful, it may perform very dangerous tasks without putting at risk the lives of human beings, and also may provide very useful information about the building state and its soundness by sending pictures or videos to the rescue team.

# C. Architecture: justification of the modifications to the robot’s architecture and explanation on how the additional sensors will aid the disaster recovery effort.

The basic model of BubbleRob is very useful on this context, the ability it has to avoid obstacles makes it the ideal prototype to any task that implies the avoidance of some obstacle (in this case, rubble found after an earthquake), but that ability is not enough by this own, that’s why the default sensor was slightly modified and 2 additional sensors were added.

Nose Sensor modification:

The nose sensor, although very useful, had a small range of sensitivity, and therefore it had to be increased. With a small range, BubbleRob still had issues when its wheels stump with objects that the sensor didn’t perceive.

Cube Detector:

To be a functional robot in a disaster environment, it’s not enough to avoid obstacles, BubbleRob should also be able to recognize and recover determined objects. To accomplish this task a "Cube detector" was added to the robot. This extra sensor will enable the robot to specifically detect objects and will trigger an “object found” signal once it does along with the object position.

It’s important also to mention that the “cube detector” sensor will have a wider angle than the proximity sensor, to make the search faster and more effective.

Back Sensor:

When BubbleRob goes backward to avoid an obstacle, it may hit something behind it and flip over. Considering this and the fact that hitting something in a post-earthquake environment may trigger a building collapse, it became evident that a back sensor with a wider angle needed to be added to the robot. When BubbleRob is in reverse and this back sensor “sees” something that it may hit, it will resume forward motion to avoid hitting any obstacle behind it. This virtually removes all unintentional reverse collisions and flips over.

# D. Goal seeking: description of how optimization principles are implemented in the prototype and explanation of how these principles include the concepts of reasoning, knowledge, representation, uncertainty, and intelligence.

For a robot, intelligence is measured by its ability to carry out its goals. BubbleRob makes decisions based on its reasoning about the environment. Equipped with two types of sensors, one to determine the route, and one being specifically a “cube detector” (to detect the object that needs to be found) it gains knowledge about the environment and its capacity to distinguish between different types of figures (Russell S. J., Robotics Feature Engineering, 2010). This feature is fundamental in an unknown, changing and uncertain environment (Russell S. J., Robotics Feature Engineering, 2010), but to get a successful competition of the task, it is also fundamental that BubbleRob stays on a constant search to find its objective. Hence, it was equipped with a wider angle for the “cube detector” sensor, allowing it to “look” for its objective more easily, which will necessarily translate to a reduced seeking time, and therefore, improving the probability of a successful task.

# E. Advantages and limitations: explanation of the advantages and limitations of the robot and description of the criteria for assessing whether the prototype solves the problem.

Advantages:

* Task competition without endangering human lives
* Possibility to get an “inside picture” for rescue teams which may be fundamental on decision making
* Location of specific objects on dangerous situations
* Possibility to work in a simulated situation to prevent undesirable outcomes

Limitations:

* The possibility that the robot runs into a more unstable wall and provoke a collapse
* It may take a long time to find the object, or it may not find it if is in a too inaccessible place
* Prone to retake routes without completing a full search first
* May end in a loop of tracks

Success criteria:

To consider this task as successful, the robot should be able to find the object in less than 10 minutes ideally (to avoid the probability of aftershocks or collapses that may endanger the object and the robot itself). Nevertheless, as stated before, the scenario on which BubbleRob will work is constantly changing and all the possible situations cannot be represented for this prototype, so we will assume that no aftershocks and no collapse of walls happen to consider the task successful.

# F. Testing and implementation plan: outline of the testing plan for the robot and the implementation plan.

To assess the functionality of BubbleRob, the best approach would be the use of a platform to model and simulate its behavior on a determined environment, this will help in the design improvement, to avoid the waste of physical resources and time. Nevertheless, it is important to mention that this scenario is considered ideal, with no new debris falling into it and no collapsing walls whatsoever.

The software selected for this task is the CoppeliaSlim platform. The scenario created for the testing phase includes an object that the robot needs to find that is located along with different obstacles that need to be avoided. The robot is located inside the building when the simulation starts, and it will be given 10 minutes to find the objective.

Future testing will consider dynamic environments, this type of environment changes while an agent is deliberating on the next action to perform (Russell S. J., 2010). In this case, it would mean the addition of new debris once the robot starts looking in the floor plan along with collapsing walls and different unpredictable events to state if BubbleRob is capable to overcome these changes. Also, in the future, we will consider the addition of multiple objects to be in this uncertain environment.

Once stated that the robot can perform its task in a dynamic environment, the prototype build may begin. And physical tests may begin with static and dynamic environments also before getting a final product.

# G. Improving the prototype: explanation of how the prototype can be further improved and it discussion on how reinforced learning can optimize the prototype’s performance.

This model may be improved in several ways. For example, BubbleRob already has 3 types of sensors (2 looking for obstacles and 1 looking for the object), but it may be improved by the addition of side sensors. This side sensors may assist BubbleRob by looking for obstacles in all directions and determine the most optimum and less dangerous route.

Th robot may also be added a memory which would make it able to map the path already searched and therefore avoid path already taken. This improvement would be significant considering that on this kind of situations the time used in the search may be a critical factor.

It is important also to mention that BubbleRob may use machine learning practices based on feedback, like unsupervised learning on which the robot may learn patterns in the input it receives (Russell S. J., Forms of Learning, 2010) which may be especially useful on dynamic environments.

# E. Robot Code: Inclusion of the robot code.

|  |
| --- |
| function speedChange\_callback(ui,id,newVal)  speed=minMaxSpeed[1]+(minMaxSpeed[2]-minMaxSpeed[1])\*newVal/100  end  function sysCall\_init()  -- This is executed exactly once, the first time this script is executed  bubbleRobBase=sim.getObjectAssociatedWithScript(sim.handle\_self) -- this is bubbleRob's handle  leftMotor=sim.getObjectHandle("bubbleRob\_leftMotor") -- Handle of the left motor  rightMotor=sim.getObjectHandle("bubbleRob\_rightMotor") -- Handle of the right motor  noseSensor=sim.getObjectHandle("bubbleRob\_sensingNose") -- Handle of the proximity sensor  backSensor=sim.getObjectHandle("bubbleRob\_backSensor") -- Handle of the back sensor  cubeSensor=sim.getObjectHandle("bubbleRob\_cubeSensor") -- Handle of the cube sensor  minMaxSpeed={50\*math.pi/180,300\*math.pi/180} -- Min and max speeds for each motor  backUntilTime=-1 -- Tells whether bubbleRob is in forward or backward mode  -- Create the custom UI:  xml = '<ui title="'..sim.getObjectName(bubbleRobBase)..' speed" closeable="false" resizeable="false" activate="false">'..[[  <hslider minimum="0" maximum="100" onchange="speedChange\_callback" id="1"/>  <label text="" style="\* {margin-left: 300px;}"/>  </ui>  ]]  ui=simUI.create(xml)  speed=(minMaxSpeed[1]+minMaxSpeed[2])\*0.5  simUI.setSliderValue(ui,1,100\*(speed-minMaxSpeed[1])/(minMaxSpeed[2]-minMaxSpeed[1]))  end  function sysCall\_actuation()  --Read the proximity sensors  resultFront=sim.readProximitySensor(noseSensor)  resultBack=sim.readProximitySensor(backSensor)  cubes=sim.readProximitySensor(cubeSensor)  if (resultFront>0) then backUntilTime=sim.getSimulationTime()+1 end  if (backUntilTime<sim.getSimulationTime()) then  -- When in forward mode, we simply move forward at the desired speed  sim.setJointTargetVelocity(leftMotor,speed)  sim.setJointTargetVelocity(rightMotor,speed)  elseif (backUntilTime>=sim.getSimulationTime()) and (resultBack <= 0) then  -- When in backward mode, we simply backup in a curve at reduced speed  sim.setJointTargetVelocity(leftMotor,-speed/2)  sim.setJointTargetVelocity(rightMotor,-speed/8)  elseif (resultBack > 0) then  sim.setJointTargetVelocity(leftMotor,speed/4)  sim.setJointTargetVelocity(rightMotor,speed/2)  end    -- Cube detector actions  if (cubes>0) then  position = sim.getObjectPosition(cubeSensor, sim\_handle\_parent)  print("Object found at:")  print(position)  end  end  function sysCall\_cleanup()  simUI.destroy(ui)  end |

# F. Panopto Recording: Demonstration of the robot’s functionalities to stakeholders that are non-practitioners addressing all of the required points.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=0bd77088-75b7-444e-8401-abc001646326

# G. Sources: in-text citations for sources.

Russell, S. J. (2010). Introduction. In *Artificial intelligence: A modern approach* (3rd ed.). Pearson. doi:9780136042594

Russell, S. J. (2010). Robotics Feature Engineering. In *Artificial intelligence: A modern approach.* Pearson. doi:9780136042594

Russell, S. J. (2010). The Nature of Environments. In *Artificial intelligence: A modern approach.* Pearson. doi:9780136042594