Software design
Team project – Deliverable 1

Team number: **28**

Team members: 5

Name	**Student Nr.**	**E-mail**
Fabiola Loffredi	2617929	f.loffredi@student.vu.nl
Carmen Toyas Sánchez	2622278	c.toyassanchez@student.vu.nl
Roshini Ramasamy	2610864	r.ramasamy@student.vu.nl
Cristiano Milanese	2600169	c.milanese@student.vu.nl
Anna Claire Favié	2626585	a.c.favie@student.vu.nl

Table of Contents

. Introduction

Author(s): Carmen Toyas Sánchez

For our version of the ROBOSEARCH system, our goal is to implement a program that consists of a robot that will randomly explore around a given area, of which properties are unknown, searching for differently colored boxes that are spread around the field. Scattered on the area there are 15 objects of different colors, possibly overlapping, of which 4 are red. We will have the robot randomly exploring the area and it will be looking for the mentioned 4 red cubes. Despite the robot having no knowledge of the objects being there, with the help of a camera it will be looking for obstacles and, when one is detected, the robot will check for the pixels of the captured image to see if it is of the wanted color. Given that it does, it will move towards the obstacle in order to assess its coordinates in the space, which are in turn stored for the user to be able to retrieve the information once the mission is completed. This procedure resembles the one used by real-world robot rescuers, able to sense the environment, detect pre-specified objects and determine their location: our final aim is to represent such situations within the simulator, possibly with different actors. Map the floor during mission execution becomes useful in terms of efficiency because smart area coverage allows rapid intervention in real-world scenarios, something that in our implementation is still missing, at this stage of the development.

When our robot encounters an obstacle, using its sensors it will recognize which direction is closer to the object to move accordingly, while using the camera to search for the proposed color. If it does not find the color that we have set for the objects, the robot will continue the search around the given area.

We will be using one robot for this idea, this robot will be equipped with a camera, bumpers, and sonars to detect the obstacles and their distance, and also to detect the colors of the certain obstacles that it has found along the way around the given area. This robot will not be operating outside of the given playground, it will be restricted to only that perimeter by the surrounding walls. Previous works [1] investigated the importance of multi-robot scenarios, affecting the outcome of a given mission thanks to the collaboration these agents provided to each other and to the eventual control tower orchestrating the bots.

All cases taken into consideration were first played through a simulator, crucial in the testing phase to assess behaviors, avoiding the problems arising from faulty devices interacting with the environment. Furthermore, the use of such tools allows developers to test Human-Computer Interaction features [2], under which our project falls thanks to its user interface and it is within the scope of this project to delve into its properties and enhance its details.

2. Requirements Specification

Author(s): Anna Claire Favié, Roshini Ramasamy, Cristiano Milanese

2.1 Requirements

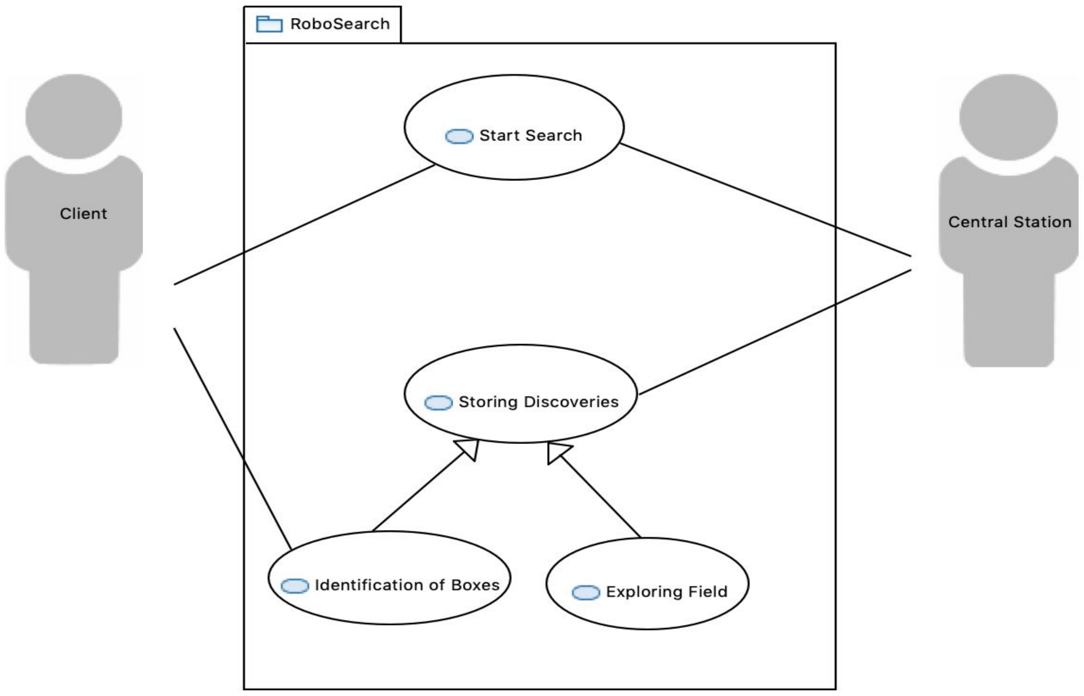
Functional requirements

#	Short Name	Description
F1	Obstacle avoidance	The rover shall move freely in the environment and avoid obstacles autonomously.
F2	Exploring Field	The rover shall locate the boxes in the field. When the rover establishes contact with an object, it will classify the object being either a box or a wall. When it makes contact with the object, the rover shall then rotate 5 degrees in each direction, continuing its search
F3	Identification of boxes	The rover shall identify the different boxes
F4	Storing discoveries	When a box is identified, its relative coordinates will be stored in an array, to make sure no boxes will be rediscovered
F5	Continuing Search	The rover shall turn around and keep searching for new objects after a box is discovered

Non-functional requirements

#	Short Name	Description & reasoning
NF1	Obstacle avoidance [Performance]	The rover shall react to the presence of an obstacle within 250 milliseconds.
NF1	Obstacle avoidance [Safety]	A rover shall always be at least 0.3 meters from obstacles, other robots, and human beings.
NF2	Exploring field [safety]	The rover is set out into the environment and move freely and randomly to explore the field for boxes.
NF3	Identification of boxes [reliability]	The rover shall classify the object as a box if red pixels are detected by the camera using the central pixel method
NF4	Storing discoveries [performance]	The rover has identified a box its coordinates will be send to central station where they will be stored in order for the client to know the location.
NF5	Continuing Search [performance]	The rover shall keep looking for new objects; if it finds a new object it will compare the new coordinates with the stored one; if the coordinates are the same the object will not be classified as it is already considered as a box.

2.2 Use Cases- This section has been updated



(Our use case diagram includes two actors, the client and the central station. Although initially in our code for this deliverable we have not included a central station, we had been thinking of implementing it and catered all our other functionalities to have included it, this can be seen by the easy integration of the central station in deliverable 2. This is why we thought it appropriate to include it in the use case diagram to show our understanding and get feedback according to the updated code.)

Name	Start Search
Short description	This triggers the environment to be set up and for the rovers to start roaming around freely.
Precondition	The client wants to identify the boxes.
Postcondition	The search starts.
Error situations	There are no boxes present in the environment.
System state in the event of an error	The client will not be able to find any boxes, since there are none present.
Actors	Client, Central Station
Trigger	Client wants to locate boxes in the environment.

Standard process	1) The client starts the search
Alternative processes	1’) The client starts the search but no boxes are present 2) The environment is regenerated until there are boxes 3) The client restarts the search

Name	Storing Discoveries
Short description	Once the rovers have located and identified the coordinates of the boxes, they are sent and stored in the central station.
Precondition	A box has been identified by a rover.
Postcondition	The coordinates are stored in the central station.
Error situations	There are no boxes present in the environment.
System state in the event of an error	The central station does not store any coordinates, as there are none present. (This is taken care of in the initial start search as if there are no boxes, the environment is regenerated until there are boxes)
Actors	Central Station
Trigger	The rovers identify boxes in the environment and the central station needs to save the coordinates for the client to know the location.
Standard process	1)The rovers locate a box 2)They retrieve the coordinates 3)The coordinates get stored in the central station
Alternative processes	1’) The rovers do not locate any boxes 2) The environment is regenerated until there are boxes 3) The rovers start the search again

Name	Identification of Boxes
Short description	Once the rovers are set out into the environment, they are able to distinguish between walls, other objects and the boxes that need to be identified.
Precondition	The client starts the search and the field has been explored.
Postcondition	The object found has been identified as a box.
Error situations	There are no boxes present in the environment.
System state in the event of an error	The rovers cannot identify any boxes as there are none present.(This is taken care of in the initial start search as if there are no boxes, the environment is regenerated until there are boxes)

Actors	Client
Trigger	The client starts a search and once the rover identifies a box, the client is notified of the amount of boxes that have been found.
Standard process	1)The client starts the search 2)The rovers are set out into the environment 3)The rovers distinguish between different objects and identifies boxes.
Alternative processes	3′) The rovers do not locate any boxes 4) The environment is regenerated until there are boxes 5) The rovers start the search again

Name	Exploring field
Short description	The rovers are set out into the environment and move freely and randomly.
Precondition	The client starts the search.
Postcondition	The objects found are identified.
Error situations	-
System state in the event of an error	-
Actors	Central Station
Trigger	The client starts the search, wants to identify the boxes and the environment has been created.
Standard process	1)The client starts the search 2)They environment is created 3)The rovers move around freely
Alternative processes	-

3. Implementation remarks

Author(s): Fabiola Loffredi

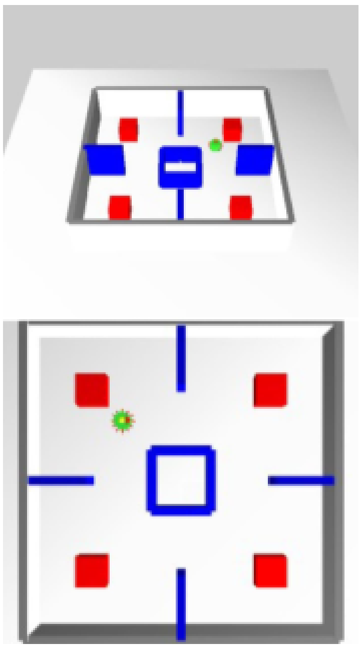


Fig. 1: "Fixed environment"

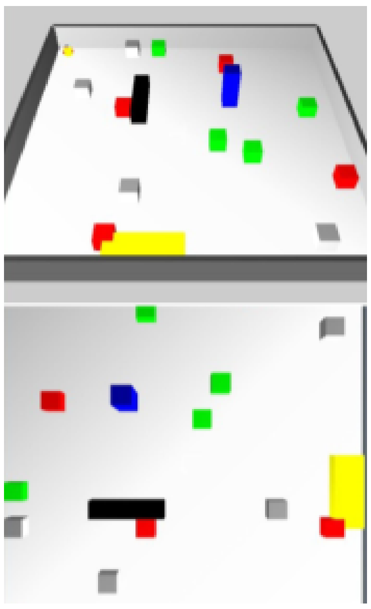


Fig. 2: Randomly generated environment

Our system consists of a robot that travels in the space in search of a specific class of objects.

The system was designed for a randomly generated environment (*Fig.2*), however, in the initial stages, we tested on a defined environment (*Fig.1*), where we could determine the position and the color of the boxes and the obstacles.

The search robots were provided with three different types of sensors: bumpers and sonars, used for obstacles avoidance, and a camera, used for object recognition.

The robots do not follow a specific algorithm to move around the environment, but they just wander and change their orientation at regular intervals of time. However, the sonars guide the robot in avoiding eventual obstacles. The robot was given a total of 8 sonars, but those that were mostly used were the Front, Right, Top-Right, Left, and Top-Left. When the two side sonars detect the safe distance, the robot turns to the opposite direction, while if the Front sonar detects something, the robot simply turns around. In the eventuality that the sonars fail to detect an obstacle and the robot hits it, the bumpers will detect the hit and will instruct the robot to turn around

During the search, the camera performs a color recognition, described by the function `boxDetected()`. For the implementation, we did not consider the entire picture frame when searching for the color. Instead, we only considered the central pixel (50,50), as we know that the image size is 100x100 pixels.

The method used to classify the different colors is the RGB model, in which every color is a combination of the colors red, green and blue, where each of them assumes a value between 0 and 255.

For our system, we decided to only consider red objects, therefore the only RGB values that we had to consider were only those relative to the red color, while the values for blue and green were set to 0. If the camera detects that the central pixel has a red value different than 0, the function returns true.

In this case, the robots will then proceed to go towards the object. In the case where an obstacle presents itself between the object and the robot, the latter will then proceed to avoid it according to the aforementioned method; this also implies that the robot will stop to look for that object and will continue to go around the environment until a new red object is detected. Otherwise, the robot will get as close as possible to the object until the sonar detects the safe distance. Then, the system returns the coordinates of the robot, while the robot will turn around and keep searching for a new red object.

During the implementation, we had to find an appropriate way to determine the coordinates of the newfound box, as there is no direct way to do it. In the first version of the code, we tried to estimate the coordinates by rounding them to the closest integer, then we added a margin of error (which in our case was the safe distance detected by the sonar) to each value.

This part was initially tested in our "fixed" environment so that we could easily determine if the coordinates were right or not, however, we encountered two main issues:

1. The coordinates changed significantly depending on the direction of the robot was coming from, meaning that the measurement, even if approximate, were not very reliable.

2. If we were to test this method on a random environment, we could not verify if the coordinates were accurate enough, as we could not set specific coordinates for the boxes.

To solve this problem to the best of our ability, we decided to consider the coordinates as a simple indication for the placement of the box. In addition, the coordinates of each box are stored in an array. Always taking into consideration the first stated issue, we did not differentiate the boxes, meaning that sometimes different coordinates may refer to the same object. We plan on solving this issue in future implementation.

4. References

[1] J.L. Baxter, "Multi-Robot Search And Rescue: A Potential Field Based Approach" in *Autonomous Robots and Agents*, 2007, M. Subhas Chandra, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, 9-16.

[2] R.R. Murphin, "Human-Robot Interaction in Rescue Robotics" *IEEE Robotics and Automation Magazine*, 3, 16, 3-5, 2005.