Vrije Universiteit Amsterdam

Software Design

ROBOSEARCH

Deliverable 1

Team number: 4

Team members:

Name	Student Number	Email
Connor de Bont	2599228	connordebont@gmail.com
Basel Sammor	2608511	b.s.a.sammor@student.vu.nl
Yasin Aydin	2657725	yasinaydin@sabanciuniv.edu
Faruk Şimşekli	2657316	faruksimsekli.7@gmail.com

Table of Contents

1. Introduction

Author(s): Yasin Aydin, Basel Sammor, Faruk Simsekli, Connor De Bont.

In this introduction, a short description of our design for the ROBOSEARCH system will be given. The ROBOSEARCH system is a system that simulates a robot search system. This is for you to grasp the ideas behind the design, and assumptions we have made along the way. In ROBOSEARCH system, robots rovers are special kind of robots that are specifically designed for autonomously moving within an environment, sensing information, and acting accordingly. Also, in the system, the mission is to move around the environment by following some strategies that we will mention later on, and search boxes of a particular color. The mission will be over if 70% of the environment is visited. Now, we will try to explain the aspects of our system one by one.

The robots we have in our system are of one type only. They have one camera on the top to take pictures to analyze the pixels, four sonar sensors, and eight bumper sensors around them. For the mission in our system, we will have four robots. They will be connected to each other through the central station only. They will not be able to send and retrieve information without the central station. The robots will be positioned at some starting coordinates by default. After the mission is over, they will return to their initial positions.

The environment we created for this phase was implemented to get familiar with the model itself. Therefore, it is medium-challenging. It can be easily improved for the upcoming phases. Our environment is unknown to the robots and central station at first. The robots will figure out a way to find the boxes. Hence, we have made some assumptions. The first assumption is that all walls in the world will be white only. The reason is that when the robots try to find the boxes they will take photos to analyze the color of boxes. This way robots will be able to distinguish the walls from the boxes. Another one is that the length of the boxes will be an integer to avoid glitches.

Since our environment is unknown, the central station plays a big role in this mission. The role is to get the information from the robots, first of all. This way the central station gets familiar with the environment and gives directions to the robots so that the robots will move to the target and end the mission automatically when 70% of the area is mapped. One thing that is worth to mention is that the central station will be a sort of database. It will keep the information about the environment according to the data taken from the robots. So, every grid point will be assigned as either obstacle, empty, or visited while the robots move around. This way central station will be aware of the grid points. This way robots will move much more efficient in terms of performance. For example, they will not visit a point that has been already visited.

There is a possibility that our robots stuck in the environment even if the central station gives directions. We will try to minimize this unfortunate possibility in the next phases with more sophisticated algorithms perhaps, but if that happens there should be someone who controls the system. This person is the operator. The operator is able to start and end the mission, view the environment from different angles so that in emergency cases, s/he is able to intervene to recover the robots. This will happen through the central station. Even the operator does not have access to the robots directly. The operator gives commands to the central station and the central station gives this information to the relevant robot.

Rover Algorithm:

The rovers shall move in a way that depends on continuous communication with the central station in order to avoid unnecessary movements. The rover starts by checking the four sonar sensors equipped on it for obstacles in the north, south, east, and west of its point and relay that information to the central station. The central station will update it's interior data structure about the environment with the new information only if the map points are still "unknown" or it's updating a point to "visited".

Next, whenever a rover wants to move to a new point, it receives a reply from the central station to find out which of the four directions it is allowed to go. For example, if three of the four points around the rover are marked with "obstacle" or "visited" and the fourth is "empty", it will receive the latter point.
In certain situations where all points are unavailable to the rover, the central station will reply with a random "visited" point to keep the rover moving until it is able to send an "empty" point.

Lastly, after moving to the point specified by the central station, the rover will inform the central station to mark its previous location as "visited". This will always be updated even if a previous point was already visited. [1, 2, 3]

In general, the algorithm will be as follows:

1. Rover checks sonar sensors.
2. Rover determines the four coordinates' status.
3. Rover sends resulting coordinates to the central station.
4. Rover receives new coordinate from the central station.
5. Rover goes to the new coordinate.
6. Rover sends last visited coordinate to the central station.

2. Requirements Specification

Author(s): Yasin Aydin, Basel Sammor, Faruk Simsekli, Connor De Bont.

In this section, we will discuss the requirements of the system.

2.1 Requirements

2.1.1 Functional requirements

The functional requirements will discussed in the following table:

#	Short name	Description
F1	Obstacle avoidance	The rovers shall move freely in the environment and avoid obstacles. If an object is detected, it should give this information to the Central Station.
F2	Goal detection	The central station will detect if the goal is reached when 70% of the map is mapped.
F3	Handle Human Intervention	Each rover shall be able to be controlled by a human actor in case of emergency.
F4	Reporting to the central station	Each rover shall report situational data to the central station on every point they move in the environment.
F5	Movement	The rovers shall move to the coordinate that central station sends.
F6	Photo capturing	The rovers shall capture photos whenever they come near an obstacle for later color identification.
F7	Analyze colored boxes	Photos of obstacles shall be analyzed to see if they match the requested color of the mission.
F8	Obstacle detection	The rovers shall detect obstacles within 1.5 meter radius from their current location with the sensors that are attached on them. This way information about environment will be collected for the mission.
F9	Avoid Unnecessary Movements	The rovers shall not visit the points that are already visited and it shall not try to move to a place where an obstacle was already found by a rover.

2.1.2 Non-functional requirements

The non-functional requirements will discussed in the following table:

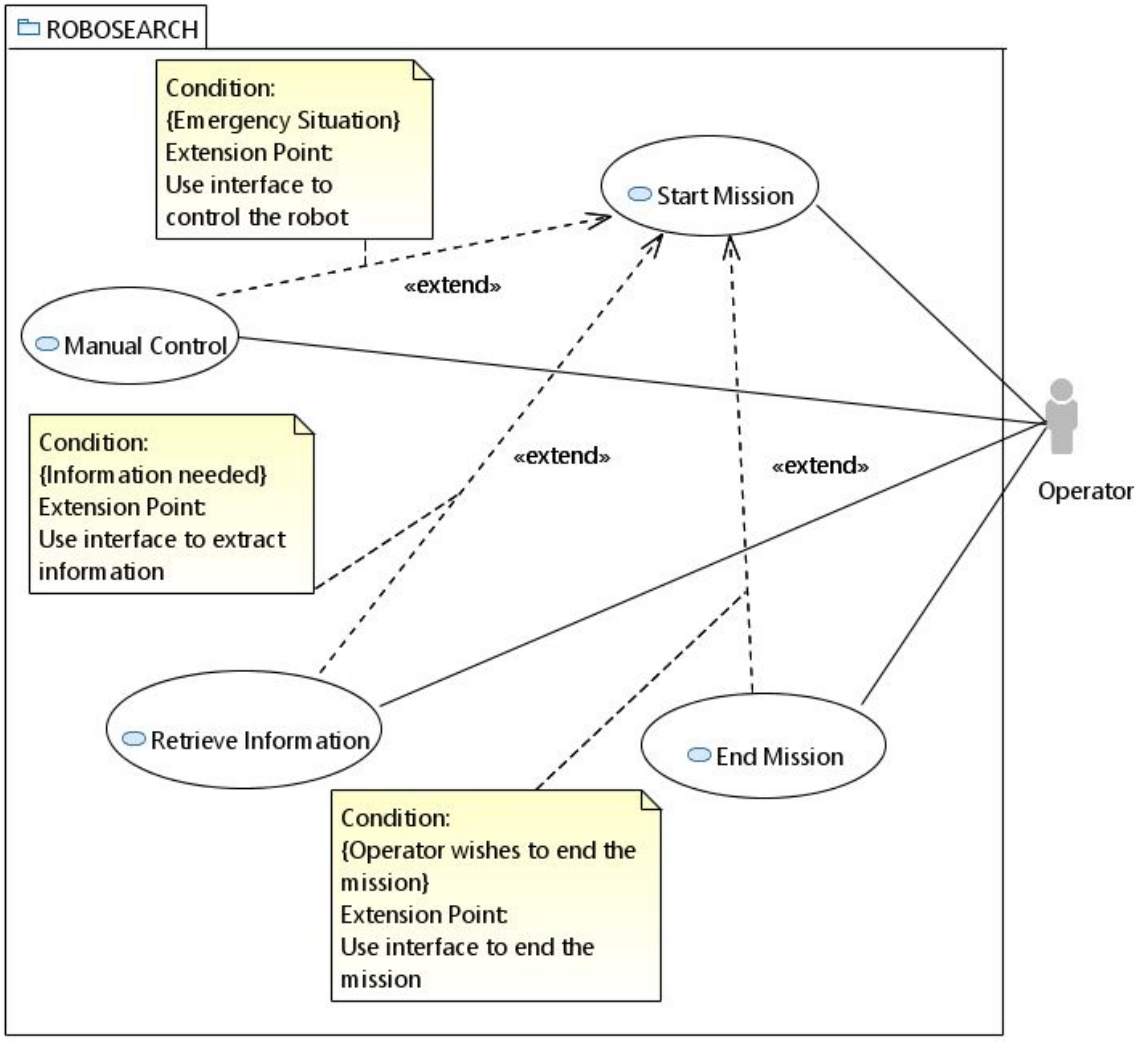
#	Short name	Description
NF1	Ease of Use [Usability]	Non-technical personnel are able to use the system without requiring training and knowledge by interacting with a simple interface, that allows choosing demanded rover to control with ComboBox's and send commands with buttons on the UserInterface.
NF2	Response Time [Performance]	The time of communication between a rover and the central station or deciding on a new direction shall be under 100 milliseconds in order to not slow down the performance of the system.
NF3	Avoid Catastrophes [Safety]	The rovers must move safely, which means there shall be minimum 1 meter distance between rover and obstacle, with the guidance of the central station in order to avoid crashes into obstacles or other rovers. The distance will be calculated with sensors and reported.
NF4	Prioritizing Requests [Availability]	The system shall prioritize manual requests initiated by a human operator and execute them first by suspending autonomous control for the duration of the manual operation.
NF5	Programming and API [Development]	Java and the simbad API shall be used in the development of the system.
NF6	Environment Shape [Environmental]	The environment must be rectangular and of a fixed size of **25x25** while not containing enclosed sub-areas.

2.2 Use Cases

The system has one actor who can perform four actions. The actor can start a mission, end the mission, control a rover manually and retrieve information about found boxes.

2.2.1 Use Case Diagram

In this section use case diagram of the system we have designed is given as follows:



2.2.2 Use Case Descriptions

In this section, descriptions of the use cases will be discussed.

Name	Start Mission
Short description	The operator starts the ROBOSEARCH system.
Precondition	The operator is authorized to start the system and requested the color of the box is chosen.
Postcondition	The mission begins and rovers starts searching in the environment.
Error situations	No box color was chosen or technical issues such as rover malfunctioning arise.
System state in the event of an error	ROBOSEARCH Mission does not begin.
Actors	Operator
Trigger	Operator wants to find specific colored boxes in an environment.
Standard process	(1) Operator chooses the requested box color. (2) Operator confirms the choice and begins the simulation.
Alternative processes	(1') Operator doesn't choose the requested box color. (2')The mission fails to begin due to the forgotten parameter.

Name	End Mission
Short description	The Operator Ends the ROBOSEARCH system.
Precondition	The operator has started the ROBOSEARCH system.
Postcondition	Rovers stop moving and the mission ends.
Error situations	-The ROBOSEARCH system hasn't been already started -The missions fails to end.
System state in the event of an error	The system remains without being started or the mission keeps on executing.
Actors	Operator
Trigger	Operator wants to prematurely end the mission because of errors or technical difficulties.
Standard process	(1) Operator checks if the mission has already started. (2) Operator decides to end the mission due to a specific reason. (3) Operator ends the mission.
Alternative processes	(1')Operator cannot end the mission due to the mission not being in operation.

Name	Manual Operation
Short description	The Operator manually controls a rover or multiple rovers of the ROBOSEARCH system.
Precondition	The mission is ongoing and the operator is authorized to manually control the rovers
Postcondition	Rovers start being manually controlled instead of autonomously moving.
Error situations	Manual Operations cannot be carried out.
System state in the event of an error	Robot stops acting.
Actors	Operator
Trigger	Rovers are stuck, in a loop, or require human intervention to avoid a hazardous scenario.
Standard process	(1) Operator chooses what operation to carry out. (2) Rover performs the manually requested operation. (3) Control is restored to the central station.
Alternative processes	(1') Operator cannot perform the manually requested operation. (2') Robot remains in control.

Name	Retrieve Information
Short description	The Operator retrieves information about ROBOSEARCH system environment regarding the positions of the colored boxes.
Precondition	The operator has started the ROBOSEARCH system.
Postcondition	The system continues to run. The locations of all known colored boxes are given to the operator.
Error situations	The system cannot transmit information to the operator or there is no information to be retrieved.
System state in the event of an error	The system continues to run. Operator will be informed about the situation in the interface.
Actors	Operator
Trigger	Operator wishes to get information regarding the location of colored boxes at that time.
Standard process	(1) Operator starts the system. (2) Operator clicks on retrieve information. (3) Information is shown on the interface for the operator.
Alternative processes	(1') Operator does not start the mission. (2') Operator clicks on retrieve information. (3') Operator doesn't receive information.

3. Implementation remarks

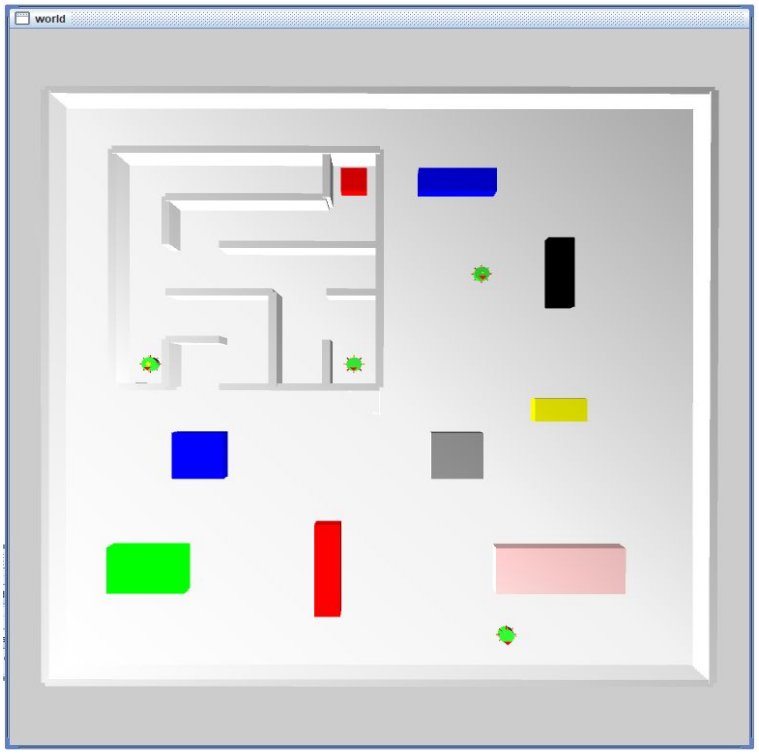
Author(s): Yasin Aydin, Basel Sammor, Faruk Simsekli, Connor De Bont.

In the first iteration of our implementation, we have kept things simple as requested. In our environment class, we have a 25x25 environment with white walls on the edges, added a number of colored obstacles inside the environment, and 4 rovers placed in different positions in the environment. Additionally, we created a simple maze in a small region inside the environment for some robots to move in.

In the Robot class, each rover is equipped with a camera for image capturing, four sonars to detect if obstacles are nearby, and eight bumpers to detect if rovers are in collision with other robots or an obstacle. The rovers move in a straight line with a translation velocity of 0.5 when no obstacles are detected and if another rover is very close, the rovers stop and change direction randomly to avoid each other.

In obstacle avoidance, we faced some issues with collisionDetection() being reliable and to fix that we created a function bumperHit() that checks the bumpers every 20 frames and whenever bumper.hasHit() returns true, we change the Y-axis of the rover by a random value. collisionDetection() is still used to rotate in case of some situations as a fail safe.

You can access the video of our system while running using the following link for this phase: https://youtu.be/WwHDcokn6V8



4. References

[1] “Robots, mazes, and subsumption architecture”, [Online]. Available: https://www.ibm.com/developerworks/library/j-robots/. Accessed February 18, 2019.

[2] “Mobile robot project”, [Online]. Available: http://aeguchi.hogsford.com/projects/mobile-robot-project. Accessed February 19, 2019.

[3] “A Comparison of Robot Navigation Algorithms for an Unknown Goal”, [Online]. Available: https://www.researchgate.net/publication/250822730_A_Comparison_of_Robot_Navigation_Algorithms_for_an_Unknown_Goal. Accessed February 19, 2019.