



# Toward a robot swarm protecting a group of migrants

Maxime Vaidis<sup>1</sup> · Martin J.-D. Otis<sup>2</sup>

Received: 18 June 2019 / Accepted: 6 February 2020 / Published online: 20 February 2020  
© The Author(s) 2020

## Abstract

Different geopolitical conflicts of recent years have led to mass migration of several civilian populations. These migrations take place in militarized zones, indicating real danger contexts for the populations. Indeed, civilians are increasingly targeted during military assaults. Defense and security needs have increased; therefore, there is a need to prioritize the protection of migrants. Very few or no arrangements are available to manage the scale of displacement and the protection of civilians during migration. In order to increase their security during mass migration in an inhospitable territory, this article proposes an assistive system using a team of mobile robots, labeled a rover swarm that is able to provide safety area around the migrants. We suggest a coordination algorithm including CNN and fuzzy logic that allows the swarm to synchronize their movements and provide better sensor coverage of the environment. Implementation is carried out using on a reduced scale rover to enable evaluation of the functionalities of the suggested software architecture and algorithms. Results bring new perspectives to helping and protecting migrants with a swarm that evolves in a complex and dynamic environment.

**Keywords** Mobile robots · CNN · Swarm · Interaction · Migrants

## 1 Introduction

The socio-political climate of insecurity over the last year has led to an increase in the number of asylum applications and political refugees in many countries, including Germany, Canada, France, etc. Governments are facing a real international migration crisis. Indeed, restrictive policies and civil wars in many countries around the world have increased refugee claim situations. These migrants who arrive massively at our borders risk retaliation from their countries and their lives. It is, therefore, relevant to conduct studies that could provide some solutions with respect to ensuring their protection. However, the limited resources of governments and the large surveillance areas near the

borders of several countries make the task almost impossible. For these reasons, our objective is to develop an autonomous system that enables supervision and protection of migrants. During their journey, migrants move mainly by foot with few means of protection exposing themselves to risky situations. In order to protect them, we need to gather information about their movement and the environment around them. Many studies focused on retrieving this information with different types of networked sensors [1–4]. These networked sensors use mainly WiFi communication [1–3], but they can also use other protocols like XBee and Bluetooth [4]. These modes of communication are relatively easy to set up and can be very useful in transmitting data between robotic platforms. Therefore, the data gathered can be used to protect people using groups of rovers. Automatic management of a robot swarm for better efficiency and to reduce human involvement in potentially dangerous terrain, often covering long distances, is a major challenge to this type of project. In addition, external factors must be taken into account to provide a safety zone around the group of migrants. With respect to this challenge, some studies have been conducted regarding a mobile group of moving rovers interacting with external factors that have an impact on the group's decision [5–8]. The suggested system should track the migrants, compute some optimal way-points around the

**Electronic supplementary material** The online version of this article (<https://doi.org/10.1007/s11370-020-00315-w>) contains supplementary material, which is available to authorized users.

✉ Martin J.-D. Otis  
Martin\_otis@uqac.ca  
Maxime Vaidis  
vaidis.maxime@gmail.com

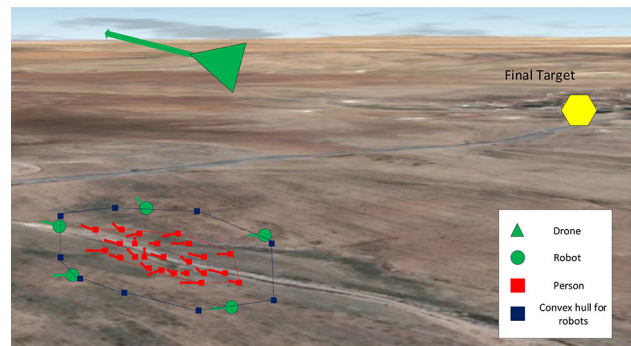
<sup>1</sup> Université Laval, 2325 Rue de l'Université, Ville de Québec, QC G1V 0A6, Canada

<sup>2</sup> Université du Québec à Chicoutimi, 555 Boulevard de l'Université, Chicoutimi, QC G7H 2B1, Canada

migrants and manage the robot's faults. The goal of the research works is to provide a safety area around the migrants and real-time assistance. This paper presents only the optimal way-points and robot faults management algorithm, including the rover control. For solving migrants tracking, some existing solutions could be implemented, such as using fixed-wing unmanned aerial vehicle (UAV), such as suggested in [9–11]; however, this subject is out of the scope of this research work.

Merging data gathered from different robotic systems allows us to follow-up what is happening on the ground and protect migrants. To process them, our suggested system acts in three steps: (1) analyze the data of migrants' behavior and give a position to reach for each rover, (2) lead the rover to this position, keep a motion vector conforming to the movement of the group of migrants and (3) manage the faults (abnormal behavior of the rover). The abnormal behavior of the rover is classified using a convolutional neural network (CNN) and is compared to a physical model. We chose to detect four situations, one normal and three abnormal states: a normal move of a rover on stones, a fall of one rover, a collision with an obstacle, and a skid on sand. Methods of human-swarm interaction mostly rely on orders given by operators via different interfaces, like computers and smart watches [5–8]. The level of automation is reduced with such interface. Conversely, the aim of this paper is to present a wireless body area network (WBAN) comprising rovers, migrants and the different algorithms created to control the swarm. Our contribution is the design of a new system labeled SROPRAS (Swarm RObots PRotection Algorithms System). With the collected data from this network, we wish to identify consistent patterns enabling their processing using fuzzy logic and CNN. These patterns can be used to detect automatically several events in order to improving the effectiveness of the swarm and the protection of migrants. Therefore, migrants do not need to send orders or commands to SROPRAS. The rovers will choose by themselves their paths and target positions with SROPRAS. This ensures a high level of swarm transparency and reduction in intrusion for the migrants.

Based on a review of our general approach to set up the system and the tools used to do the research work, we described the primary contribution of this paper, which is the design of SROPRAS based on three different algorithms created to control the swarm of rovers, including the rover positioning algorithm, the rover planning trajectory algorithm, the rover moving algorithm, and abnormal behavior management. Then, we explain how we can detect the different states of each rover to know how the swarm will react if there is an issue. The SROPRAS simulations show encouraging results, which are discussed.



**Fig. 1** Context of the use of the proposed system (simulation video available in Electronic supplementary material)

## 2 Related work

An example of the context of our research work is shown in Fig. 1. Our tool integrates different sensors in order to track migrants using inertial measurement unit and re-organize the swarm in real time. Therefore, this section presents some applications of such sensors in swarm robotics.

Inertial measurement unit (combination of accelerometers, gyroscopes, and magnetometers) is widely used in the field of robotics to help localization algorithms. The received data can be analyzed: motion measurement of robotics arm or mobile robots [12,13], tracking people or robotic systems [14,15], gait analysis [16,17], and inertial navigation or positioning for mobile robots [18,19]. Its affordability and ease of use on multiple robotic platforms make its integration in robot swarm possible [20].

Swarms of mobile robots enable the execution of many tasks faster and more effectively than a lone robot, for example, in field exploration [21,22], search for a target of surveillance [23,24] or rescue [25,26]. This is possible because of their number as well as their group intelligence, which allows distribution of tasks between robots in the swarm. Depending on its level of autonomy, the swarm can perform more or less complex tasks. Most modern mobile swarms are controlled by one or more operators [7]. They must follow the evolution of the robots and influence their performance if necessary, usually by assigning them a different goal to achieve [27]. However, this interaction depends on the communication mode between robots and humans [28]: speech, gesture, joystick or all of them. The point is that interactive exposure to a robot can change a user's perspective of expectations from a robot's behavior. Therefore, the communication mode between robots and humans should be carefully chosen to achieve desired objectives. Moreover, this communication mode should be transparent to provide an autonomous swarm.

However, improving the interaction between robots and humans is not enough to achieve effective autonomous

swarms. The structure of decision-making should find an optimal balance between the individual command of a robot and the overall performance of the swarm. The robot must have enough liberty to be capable of performing its actions, but it must comply with the aims of the swarm. Some rovers can follow a direction [29], perform mechanical actions [30], and measure environmental conditions [31], while some others can assign goals and trajectories [19–21]. In this article, our swarm should be able to coordinate all these actions while following a group of migrants. Other research works have been designed to help the operator select a robot in particular. This leads to a simplification of the interaction between the operator and the swarm [32]. But a complex swarm that needs to perform different actions, like that of our research work, it is complex for a human to control all robots separately while carrying out their tasks. Moreover, the suggested swarm should react as a function of the human behavior and not as a function of a command.

Some solutions have been suggested to solve this issue. One of them is to define two different roles among the robots that make up the swarm. One or many robots will be the leader(s) of the swarm that give instructions and collect all the information, and the others will only communicate with the leader to give all the information needed and carry out their instructions [33]. The operator has only the relevant information and can control or influence the leader to interact with the swarm [34]. Another solution is to define the operator as the leader of the swarm; this will allow him to directly assign goals to the swarm who will adapt his local command through their robots [35,36].

In order to implement this system, some frameworks for swarm control were suggested and designed [37], such as Robotic Operating System (ROS) [38,39]. ROS is a meta-operating system that can run on one or more computers, and it provides several features: hardware abstraction, low-level device control, implementation of commonly used features, transmission of messages between the processes, and the management of the installed packages [40,41]. In other words, ROS is a server-client able to provide an implementation of distributed artificial intelligence algorithms inside the swarm; therefore it will take all the team decisions related to the swarm's actions [42,43].

Artificial intelligence is the most modern technology in the field of robotics. With precise control and less computing time, it can be easily implemented in mobile robot. Moreover, it has the advantage of overcoming some mathematical problems in motion and path generation. One issue it can answer is the recognition of a robot's state due to situations they can encounter and the solution they can execute to deal with it. Currently, many sensors allow us to gather information on their environment and some algorithms can analyze these data [44,45]. However, they are specific to some situations and cannot be used in a more general case. For this reason,

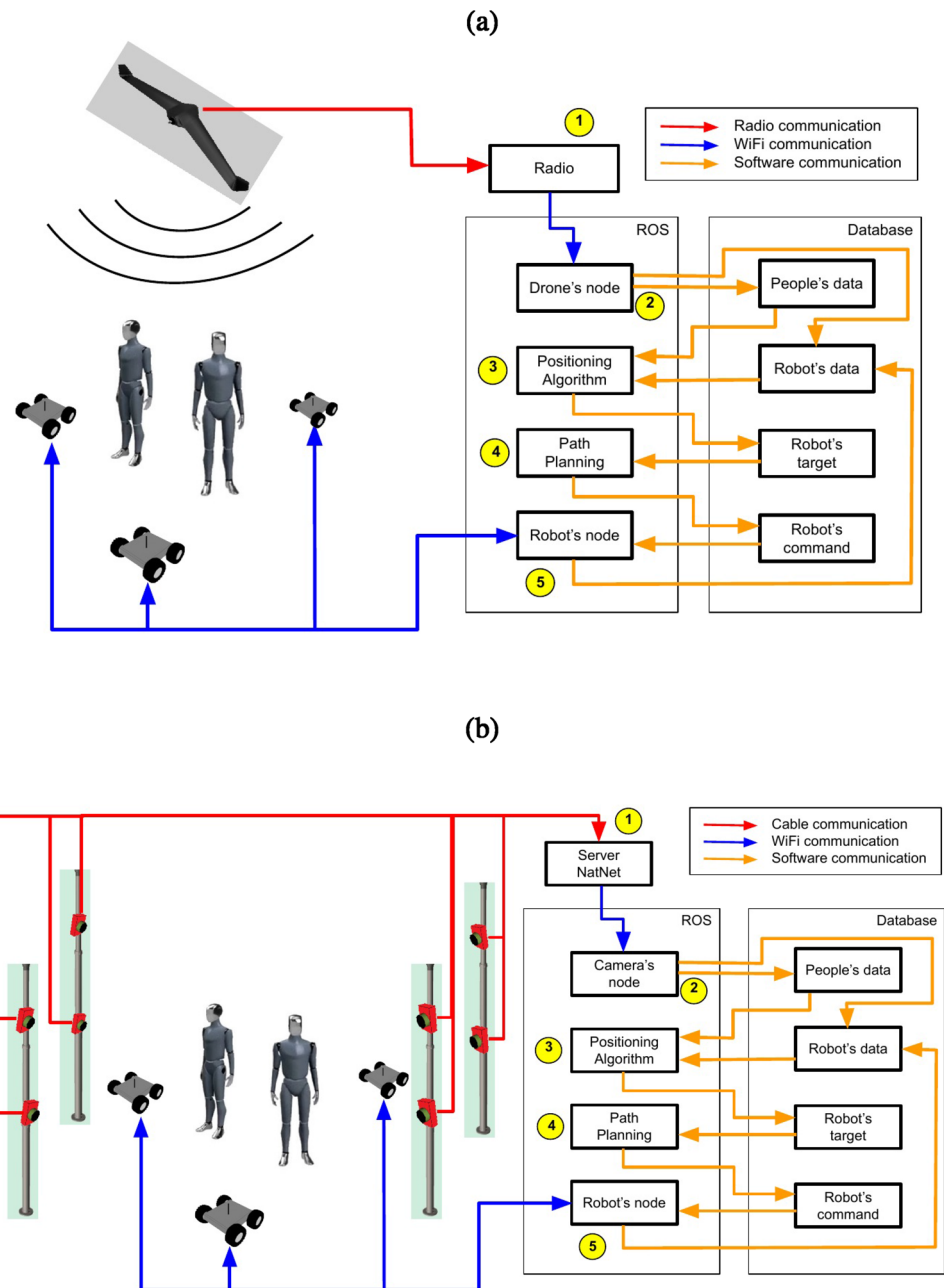
many studies focus on implementation of deep learning algorithms in mobile robot. This type of algorithm works for the majority of situations encountered. They are found in many fields of mobile robotics, including obstacle avoidance [46], cooperation between several robotic systems [47] and detection of a robot's state using camera [48]. On the basis of these studies, we can use deep learning to evaluate the state of a robot with data gathered from different sensors. Some studies used a multilayer perceptron with data from an accelerometer and gyroscope as input to detect the fall of a robot [49]. Specifically, some other studies used convolutional neural network (CNN) to process the data gathered from sensors so as to detect different human activities [50] and recognize some hand gestures by using an IMU [51]. We decided to apply this principle based on [52] and [53] on the rover to detect its state. According to this algorithm, the swarm will have all the necessary information to carry out some choices and adapt itself to the environmental constraints.

The management of multi-robots could be applied in harsh environments, such as mountains and valleys, which contain many different kinds of obstacles [5–8]. On these difficult fields, some properties of swarms allow them to adapt their behavior to face situations like damage of one of their robots. To this end, their capacity to deal with information through a decentralized system gives them an important advantage during military operations. If one of their robots is damaged or destroyed, the swarms will normally continue to operate with the remaining robots. Then, they will plan their deployment according to their aims and situations encountered. Moreover, they will take into account the movement constraints of their robots to optimize their results [54]. Indeed, depending on to the type of robot, they will have more or less degree of freedom to move [55]. Finally, the utilization of WiFi and XBee allows us to send the required data in a short period of time, facilitating and reducing the time of decision-making [56–58].

### 3 Suggested swarm of rovers

We suggest a swarm of rovers to evaluate the control algorithm proposed in the paper. First, we present the rovers used for the swarm. Second, we discuss the wireless body area network and the motion tracking used with the rovers to communicate and track people during their motion. For outdoor applications, we suggest the usage of a drone to have the position of migrants and rovers as shown in Fig. 2a. The evaluation of the swarm will be done indoors for logistic reasons. Figure 2b shows the suggested framework used in our research work. The motion tracking was done by using eight NaturalPoint OptiTrack cameras, replacing drone with a technology of localization, such as lidar and IR cameras for migrant localization and GPS for rover localization (1).

**Fig. 2** **a** Suggestion of setup in outdoor environment, **b** experimental setup in the laboratory



For the two suggested systems, it is always the same software configuration. The localization of robots and people is achieved by a camera's node in ROS (2). Then, the different algorithms (positioning (3), path planning (4) and state detection (5)) presented in Sects. 4.1, 4.2, and 4.4 are applied. The results are used by the robot drive controller presented in Sect. 4.3 to drive the rovers to their targeted positions in real time (5). The idea of the system comes from a swarm of rovers and UAV for Mars exploration [59]. However, our challenge will be migrants' detection and the control algorithm in order to protect the migrants and provide a safety area.

### 3.1 Rovers

In order to evaluate the suggested algorithms and architecture, we used a reduced scale rover composed of two servomotors and four wheels (Lynxmotion Aluminum A4WD1 Rover). Their battery capacity is up to 3 h, 1 h at full speed. The maximum speed is approximately 2 m per second. However, a full scale implementation will need a rover such as those available at Boston Dynamic or Argo-XTR (including J8 XTR: 30km/h with a 1250lb payload that can be used for a gasoline generator and a solar panel kits). Any kind of rover

could work with our algorithm (humanoid, legged, wheeled, hexapod, quadruped, etc.).

The reduced scale rover selected in this research work has been used in many different swarm projects, including the student competition organized by NASA Swarmathon to collect materials such as water ice and useful minerals from the Martian surface to be used by astronauts who land on Mars in future missions [60]. One servomotor controls the speed of the robot, and the other controls its rotation. It is also built with three different types of sensors: three ultrasonic sensors to detect very near obstacles, a laser sensor (lidar) in order to avoid all collisions in the swarm's environment, and an angular sensor (IMU) to allow the computation of the orientation of each robot. The entire components work and communicate by mean of Arduino hardware. The system is implemented on Ubuntu 14.04 LTS OS with the use of Robot Operating System (ROS) [38] and [39].

### 3.2 Wireless body area network (WBAN)

To establish a protocol of communication in our swarm using ROS, we chose to use WiFi through the ESP8266. ESP8266 is a micro-controller with an integrated WiFi module. Each robot need to be connected with one ESP in order to create a network of WiFi module that communicates a fairly large amount of information within a short period of time. The ROS community has already suggested a node for the ESP, and it is, therefore, possible to develop an integrated platform for the swarm.

### 3.3 Motion tracking

Many projects use drones to follow-up people or vehicles; the method depends on the environment of the drones and their missions. Some of these methods target accurately a person and can differentiate many groups of people [61] and [62]. Dual rotating infrared (DRIR) sensor was suggested as a new technology to track multiple targets moving unpredictably [63]. Moreover, differentiation between animal, adult human, and children is suggested in [64]. A data fusion algorithm is used with unattended ground sensor (UGS) to make this differentiation. UGS is also used to differentiate vehicles.

This aspect is still a research subject, which we will study in our future works. Since our algorithms are evaluated indoors in a laboratory environment, we used a motion capture system, a network of eight Optitrack Flex3 camera, to determine the positions of people and robots in real time. The position obtained from the motion capture acts as the drone to provide the data to the rovers. Data are sent in real time to a ROS server that will analyze them and store them into a database including the data given by the rovers.

## 4 Suggested algorithms to command the swarm and rovers

The following section details our algorithms that control the positions of the rovers and drive them around a group of migrants. The suggested command and control algorithms are executed in four steps: i) an algorithm takes information from the motion capture camera and gives every rover a target to reach; ii) a second algorithm takes the precedent result and computes a trajectory for each rover; iii) a fuzzy logic bloc coded on the rover will receive the information and drive it to the target with the appropriate motion vector; and iv) a convolutional neural network (CNN) differentiates some events related to the state of a rover.

### 4.1 Rover localization algorithm

In order to provide a trajectory to the rovers around the group of migrants, the algorithm searches a database for the localization of each person. Each position is represented by a point in a 3D coordinate (world) frame, which allows us to define the cloud of points made by the group of migrants or rovers. In order to provide a safety area, rovers should be around the group. The convex hull of the cloud of points of migrants is used to place our rovers beyond this one.

Considering the few number of persons during our experiments, we computed each point part of the convex hull with the gift wrapping algorithm [65], as shown in Fig. 5. For higher number of persons, another algorithm, like the Graham scan algorithm [66] or the Chan algorithm [67], should be used.

Of course, some exception exists and is taken into account, as shown in Fig. 3:

- For less than three migrants, we cannot apply an algorithm to search a convex hull because three non-aligned points are needed at least. In that case, the gravity center of the localization of migrants is used as a circle center, which will have a radius higher than the safety distance between migrants and rovers.
- If the number of migrants is more than three, the algorithm will search to define the convex hull of the group

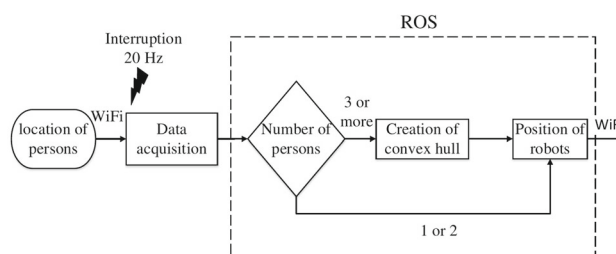


Fig. 3 Rovers position algorithm around a convex hull

(labeled inside convex hull). Once this inside convex hull is defined, we expand it by a certain safety distance predefined by the operator. This will create an outside convex hull where the rovers will be located.

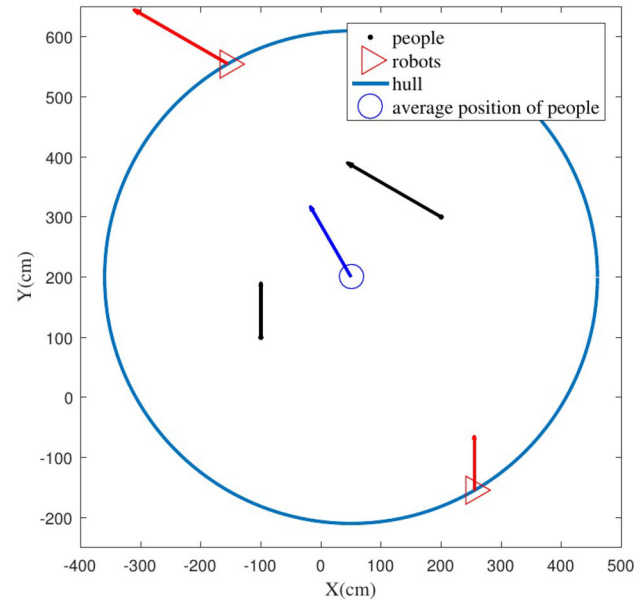
In both cases, rovers will be placed uniformly around the migrants. This algorithm gives as output the predicted position of the rover around people. It is the next two algorithms, the path planning and the robot drive control, that will drive the rover until it reaches its target. The gravity center of the group is used as the center of uniform angular cutting off a plane; the number of sections depends on the number of rovers available to perform the mission. We chose to begin the first section from the mean direction of the group of people. Consequently, the first rover of the swarm will be always located in front of the group (vector direction of the group motion) on the outside convex hull and becomes the leader of the swarm. If we have two rovers, the second will be placed behind the group, bringing up the rear of the group. For three rovers, these will be on the second convex hull and each should have a position doing, respectively, an angle of  $120^\circ$  and  $-120^\circ$ , respectively, with the first rover in front of the group and the gravity center of the people. An additional rover will divide the angle.

As a result of the group motion, each target of the rovers is moved accordingly. The rovers will be moved in such a way to achieve their targets and constantly surround the group during the mission. It is also possible to give each rover a motion vector determined by the group motion vector near them. A scalar product of the motion vector and the normal vector of the rover on the inside convex hull is used to know if a migrant will go out of the convex hull or not. When a migrant is going out of the inside convex hull, the rover near him takes the motion vector of the person that has the maximum value of scalar product of his motion vector and the normal vector of the rover. Otherwise, the rover will take the motion vector of another person near him. If nobody is less than 6 m from the rover, it will take the average motion vector of the group.

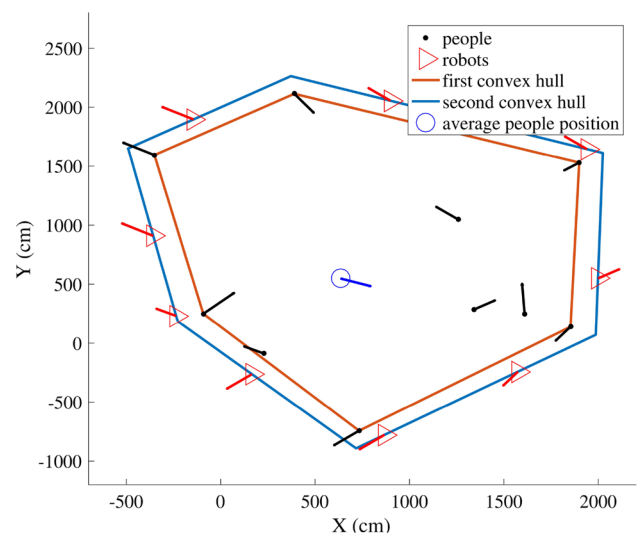
To link the computed positions to the rover, we associate them by the nearest distance between each other. Two different results are shown in Figs. 4 and 5. The predicted positions given by the algorithm are sent in a database to be used by the path planning algorithm presented in the following section.

## 4.2 Path planning

Many global path planning and local path planning methods exist in real time [44]. In our case, we chose the local VFH (vector field histogram) method which provides us real-time processing while avoiding trajectory oscillation issues present in the VFF (virtual force field) method. The result will be coupled with the robot's control drive algorithm, which



**Fig. 4** Simulation given by the algorithm for two persons and two rovers: each arrow provides the direction and amplitude of displacement



**Fig. 5** Simulation given by the algorithm for ten persons and nine rovers: each arrow provides direction and amplitude of displacement

allows the rovers to move in their environment to reach their target and avoid obstacles at the same time [68].

The algorithm will work on each rover's node on ROS, allowing us multiprocessing at the same time for all connected rovers. The node searches a database for all positions of obstacles situated in a local map defined by a square of  $2 \times 2$  m around the rover, including any migrants who are assimilated as obstacles. The size of the square is chosen in order to anticipate new obstacles and avoid them near the rover without being too wide to save computing time and

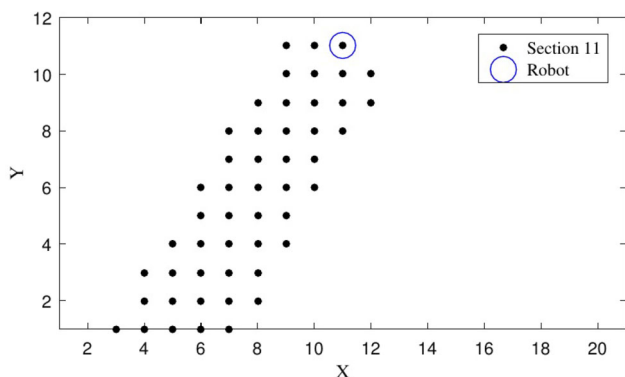


Fig. 6 Simulation of a section given by the vector field histogram method

memory during the process. The map and each obstacle coordinate are in centimeter resolution. This resolution generates a certain amount of data. Consequently, we chose an array of  $20 \times 20$  representing the square of the map; in other words, each case of this local map contains the obstacles in an area of  $10 \times 10$  cm. This will substantially reduce the processing time.

When obstacles are detected inside a case of the map, the value of this case is set to one. Otherwise, the case is set to zero. Then, we apply the local VHF method to compute a trajectory. The matrix composed of  $20 \times 20$  cases is divided into 15 rectangular sections of width adapted to the rover (40 cm), allowing the rover to pass between two obstacles if it is possible. This division is done once at the beginning of the program and is stored in the global variables to reduce the processing time. An example of a section is shown in Fig. 6.

Once the sections around the rover are created, the algorithm verifies the presence of obstacles in each of them. Every section without obstacles is memorized and represented as a potential goal to reach by the rover with a middle point located at their border. The direction chosen will be the one whose middle point is closer to the target that is to be reached, i.e., the point given by the control position algorithm. This direction will be sent to the rover by a local drive control algorithm. If the target to be reached is inside the map of the rover, the algorithm will check if it is possible to reach it directly. If this is the case, it defines it as the final direction to reach. An example is shown in Fig. 7. The black rectangular blocs represent obstacles around the rover, and the coordinates targeted are (21, 8). The selection of the target depends on the average motion vector of the group (direction and velocity).

### 4.3 Robot drive control

Once the positions are found for the trajectories, the server sends the results to the available rovers in the mission. A local drive control algorithm receives the current position of the

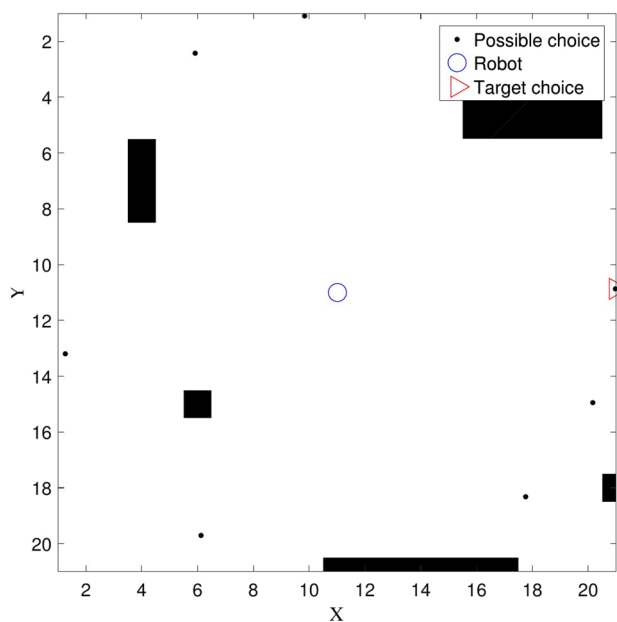


Fig. 7 Simulation of a section’s algorithm: black rectangular blocs represent obstacle

rover and the setpoints: the desired position and the desired motion vector until the rover reaches the target. The rover does not have a linear drive control but needs to take some decisions in real time. To overcome this issue, a fuzzy logic bloc architecture was developed to drive it [69] and [70].

We chose three inputs that can give information about how to reach the target: (1) the distance between its position and the desired position, (2) the angular difference between the current rover’s orientation and the angle created by his current position and the desired position (target to reach), and finally (3) the distances between the rover and obstacles measured by the lidar. Because each rover has two servomotors, the system has two outputs: a speed command and a rotation command.

Each variable is presented in Table 1. We chose to use triangular functions to describe each of them: (1) distance between the rover and the target in centimeters (zD = zero Distance, nD = near Distance, mD = middle Distance, fD = far Distance), (2) angle in degrees (zA = zero Angle, nA = near Angle, mA = middle Angle, fA = far Angle), (3) distance between rover and obstacles in centimeters (nO = near Obstacle, mO = middle Obstacle, fO = far Obstacle), (4) speed command (nV = not Velocity, sV = slow Velocity, V = Velocity, fV = fast Velocity), and (5) angle command (nO = not Orientation, sO = slow Orientation, O = Orientation, fO = fast Orientation).

With these variables, we can write some rules as follows:

$$R_i : \text{if } x_1 \text{ is } X_1^i \dots \text{ and } x_n \text{ is } X_n^i \\ \text{Then } y_1 \text{ is } Y_1^i \dots \text{ and } y_m \text{ is } Y_m^i. \tag{1}$$

**Table 1** Fuzzy logic variables used for the rover motion

Rover drive control		
Variables	Membership functions and values	
<i>Input</i>		
Distance	zeroDistance	tri([0 0 0 15])
	nearDistance	tri([10 20 20 50])
	middleDistance	tri([40 65 65 90])
	farDistance	tri([80 90 90 190])
Angle	zeroAngle	tri([0 0 0 20])
	nearAngle	tri([15 30 30 60])
	middleAngle	tri([50 75 75 100])
	farAngle	tri([90 100 100 200])
Obstacle	nearObstacle	tri([0 0 0 40])
	middleObstacle	tri([30 50 50 70])
	farObstacle	tri([60 70 70 190])
<i>Output</i>		
Speed command	notVelocity	tri([0 0 0 5])
	slowVelocity	tri([0 10 10 15])
	Velocity	tri([12 21 21 27])
	fastVelocity	tri([25 31 31 37])
Angle command	notOrientation	tri([0 0 0 8])
	slowOrientation	tri([5 11 11 17])
	Orientation	tri([15 21 21 27])
	fastOrientation	tri([25 31 31 37])

There are 768 possible rules for this system. To reduce processing time, we selected 28 of them that allow us to drive the rover in all rover statuses. The rules selected, and the results are presented in Table 2. At the defuzzification step, the output variables are sent to the drive control. The direction to turn is determined earlier according to the position of the rover and the position of the target.

The integration of these three algorithms enables each rover to move around the convex hull while avoiding collision with its environment. However, if one rover has some issues, the swarm should be aware of the situation to adapt itself. For this reason, we chose to analyze all possible states of each rover. These states are presented in the next section.

#### 4.4 CNN for states differentiation

In order to protect migrants in a complex environment, our system should be able to know the state and activity of each rover. This is essential to ensure appropriate distribution of the rover around the group. Indeed, if one of the rovers has an issue and cannot follow its desired position target, the swarm should be able to adapt itself to protect the sector uncovered by the rover that is not available. To detect some of these situations, we used a CNN according to a library of states (events) defined in the following section.

**Table 2** Fuzzy logic rules

Fuzzy logic rules					
Rules	Inputs			Outputs	
	Distance	Orientation	Obstacle	Speed	Angle
1	zD	zA	nO/mO/fO	nV	nO
2	nD	zA	nO/mO/fO	sV	nO
3	mD	zA	nO	sV	nO
4	mD	zA	mO/fO	V	nO
5	fD	zA	nO	sV	nO
6	fD	zA	mO	V	nO
7	fD	zA	fO	fV	nO
8	zD	nA	nO/mO/fO	nV	sO
9	zD	mA	nO/mO/fO	nV	O
10	zD	fA	nO/mO/fO	nV	fO
11	nD	nA	nO/mO/fO	sV	sO
12	nD	mA	nO	sV	sO
13	nD	mA	mO/fO	sV	O
14	nD	fA	nO/mO/fO	nV	fO
15	mD	nA	nO	sV	sO
16	mD	nA	mO/fO	V	sO
17	mD	mA	nO	sV	O
18	mD	mA	mO/fO	V	O
19	mD	fA	nO/mO	nV	fO
20	mD	fA	fO	sV	fO
21	fD	nA	fO	fV	sO
22	fD	nA	mO	V	sO
23	fD	nA	nO	sV	sO
24	fD	mA	fO	fV	O
25	fD	mA	mO	V	O
26	fD	mA	nO	sV	O
27	fD	fA	nO	sV	fO
28	fD	fA	mO/fO	V	fO

(1) *States definitions for one rover:* To detect the events that will generate a rover state, we created a library of different events and states that the rover will encounter:

- State 0: the rover is not connected yet to the server (the leader); it is not a part of the swarm.
- State 1: The rover is fully operational and is a part of the swarm, receiving and executing the orders which there are sent to him.
- State 2: This state indicates that the rover experienced an unexpected network disconnection from the server. As we do not have any control over it, it is temporarily removed from the swarm awaiting a potential re-connection. The swarm adapts itself to this situation by changing the distribution of the rover around the outside convex hull.



- State 3: The rover experienced a serious fall. This fall has been detected by the CNN due to the data of the IMU. Thereafter, if the server detects that the rover is moving to its targeted position, its state returns to 1. Otherwise, the rover is removed from the swarm awaiting the intervention of an operator. The swarm adapts itself to this situation as described in state 2.
- State 4: In some environments, such as the desert or temperate forests, there is a risk of the wheels of the rover skidding in sand, snow, ice or mud. This state refers to this situation. The skidding is detected both by the CNN and the server, who sees that the rover is not moving despite its commands. Then, the rover is removed from the swarm awaiting the intervention of an operator.
- State 5: It is possible that a lidar did not detect some obstacles in the current trajectory of the rover. This state indicates a collision. The collision is detected by the CNN due to the data given by the IMU. The server disconnects the rover from the swarm while it is trying to get around the obstacle. When the rover evaluates that it can reach the outside convex hull, it is reinstated into the swarm. The obstacle that was not detected is registered to avoid this situation with other rovers.
- State 6: This state indicates that the rover is trapped in some branches of trees or some obstacles that it cannot overcome and escape from. The CNN detects this state, and the rover is removed from the swarm awaiting the intervention of an operator.
- State 7: Considering that the rover could be used for several hours, it should be recharged or fullled frequently whenever solar panels are used. This state enables the system to predict maintenance using a predictive algorithm.
- State 8: The rover has the aim of protecting migrants in risky areas. This state indicates that the rover experienced an explosion or irreparable damages. The rover is removed from the swarm and should be replaced or destroyed.
- State 9: While the rover is traveling, it could experience some damages preventing it from pursuing its mission. This state indicates to the operator that the rover needs to be fixed. It is temporarily removed from the swarm.

This library of events allows us to follow in real time the state of each rover and the swarm evolution. The operator can act if one of them needs an intervention. The swarm is also kept updated about any rover that cannot be used. It can remove any rover in real time to adapt itself to the situation. In the next part, we will present the CNN that can detect these events. Considering that we evaluated the swarm in a laboratory environment, we could not test all the situations above. We selected four states: (1) the rover is moving normally on

small stones; (2) it is falling, (3) it has experienced a collision and is trying to escape; and (4) the rover has skidded in sand.

(2) *Suggested CNN architecture*: To detect these different states, many studies have been conducted. Anti-skid systems have been developed over the years, such as the one that is presented in [71]. The gyroscope and magnetometer give some data, which are processed by a Kalman filter to get a correct orientation after the skid. Then, some kinematics equations are applied on the result to control the trajectory of the robot. Also, some fall detection processes were implemented. For instance, the Nao robot uses a deep learning approach to predict a fall [49]. It uses a multilayer perceptron composed of 100 values of the x and y-axis of the gyroscope concatenate in one vector for the input. The output is defined by two states: the robot is stable or unstable. For collision avoidance, sensors are used to detect obstacles to be avoided by implementing a planned trajectory. Collision occurs when the sensors fail to detect objects (cross-talk, absorption, refraction, reflection, etc.). Bumpers are adapted to indoor motion as suggested in [72] and do not work in mud, sand, and gravel; we prefer IMU information, which does not interfere with the composition of the environment. Moreover, the dynamic model of the rover is not available, and we prefer to use an IMU sensor. Other research works specialized in the detection of one event, whereas our work is concerned with the classification and differentiation of each state.

As we shown in the Related Work section, many robotic projects rely on CNN to process the data obtained from sensors. One advantage of CNN is that it learns directly from features identified to differentiate events. Moreover, it can deal with more data with fewer weight in comparison with a multilayer perceptron. This allows it to use less memory, and it is easier to implement in robotic systems.

Our network relies on data from the IMU, sampled at 20Hz. A sliding window of 13 samples over 2 s in each axis of the accelerometer and the gyroscope is memorized in order to extract some features from them. An overlap of 12 samples is used to slide the window. More than 2400 acquisitions were realized to create a dataset that will train the network. With these raw data describing the four (4) situations selected previously, we calculated some features and kept nine (9) of them that we can use to differentiate the different states. These features are shown in Table 3 (where  $A_{x,i}$ ,  $A_{y,i}$ ,  $A_{z,i}$  are measurements taken in the axis of accelerometer, and  $G_{x,i}$ ,  $G_{y,i}$ ,  $G_{z,i}$  are measurements of the gyroscope;  $i$  is the sample of the window). With these measurements, we created an image of  $9 \times 9 \times 1$  pixels as input to the CNN based on the method used in [52] and [53]. To do that, we took the value of the feature and duplicated it on all the lines, one line for one feature. The choice of this spatial representation is arbitrary. We wanted to have a small image to be processed by a small number of filters (2 convolutional filters in our

**Table 3** Features from the data of IMU

IMU	Features
Accelerometer	$\text{Mean}(A_{x,i})$
	$\text{Std}(\sqrt{A_{x,i}^2 + A_{y,i}^2 + A_{z,i}^2})$
	$\text{Std}(A_{x,i})$
	$\text{Skewness}(\sqrt{A_{x,i}^2 + A_{y,i}^2 + A_{z,i}^2})$
Gyroscope	$\text{Mean}(\sqrt{G_{x,i}^2 + G_{y,i}^2 + G_{z,i}^2})$
	$\text{Mean}(G_{y,i})$
	$\text{Std}(G_{y,i})$
	$\text{Kurtosis}(G_{z,i})$

case). Other techniques exist such as [50]. In this case, the input of the neural network is directly the raw data of the IMU in a short window of time. The CNN should find the features and classify the different activities. In our case, we found some explicit features to differentiate each state of the robot. In order to obtain a better result there, we directly build the image with them.

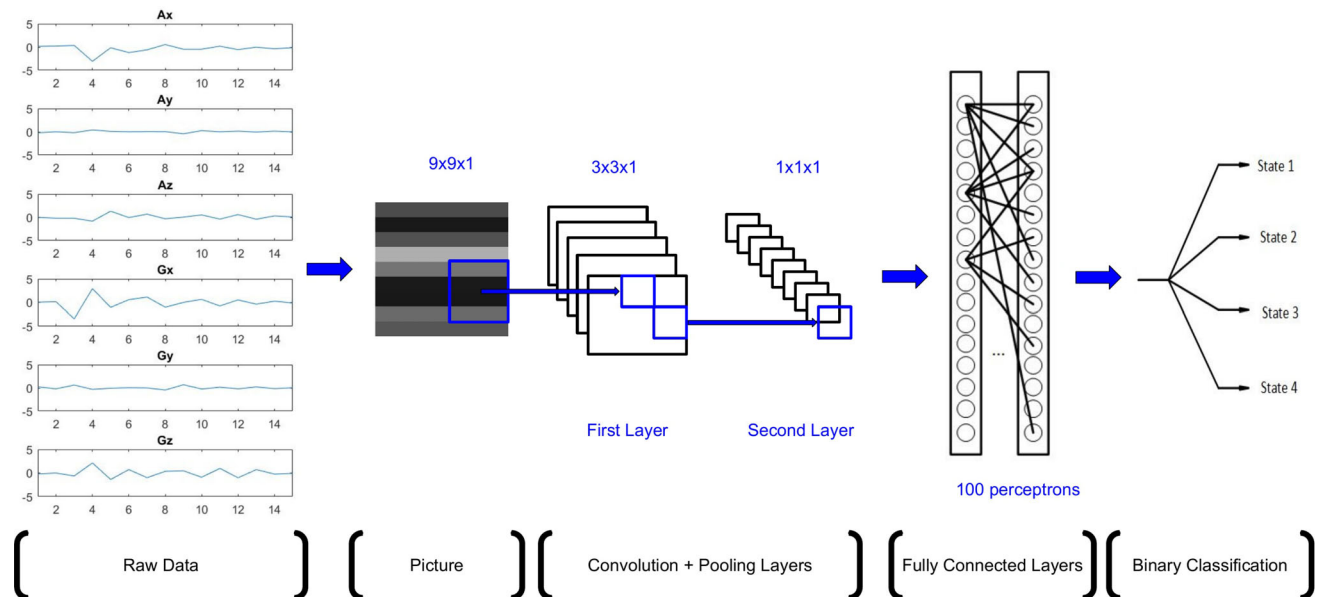
The network used in our research work to detect the four different situations is composed of two convolutional layers, as shown in Fig. 8. First, the network initializes with random weights the convolutional layers. The first layer of convolution is composed of a kernel with a dimension of  $4 \times 4 \times 10$  and the dimension of the second layer kernel is  $2 \times 2 \times 20$ . Then, the input takes a picture of  $9 \times 9 \times 1$  pixels based on features extracted from the data of the IMU. The output is given by a multilayer perceptron to classify the picture from the features. It has 100 perceptrons and four outputs which

represent each of the four situations selected. The result is given by the Softmax function with a threshold of 0.6. At each iteration, the output is compared to the desired result, and the weights are adjusted by the back-propagation method to obtain the target results. The learning rate decreases linearly over the time. We chose this configuration because of the small size of the picture and the memory size constrain of our system.

The dataset is composed of 2873 pictures to evaluate the network. We chose to use 86% of them for the training part and 14% to test our results. The repartition of the folds was determined randomly. Regarding the 2468 pictures used for the training, 2033 of them were used to change the weights, and the other 435 pictures were used to validate the result during the training. Each of them is annotated in XML files indicating the situation that they are representing in the training. Four hundred and five (405) pictures were not used for the training, and we tested our network with them. The results are shown in Sect. 5.2. In this research, all the CNN algorithms were developed using TensorFlow (Python) and performed on a computer equipped with a 2.66 GHz Intel(R) Xeon(R) W3520 CPU, a NVIDIA GeForce GTX 1080 and 10GB RAM. The time of a training was two (2) hours for 500,000 steps.

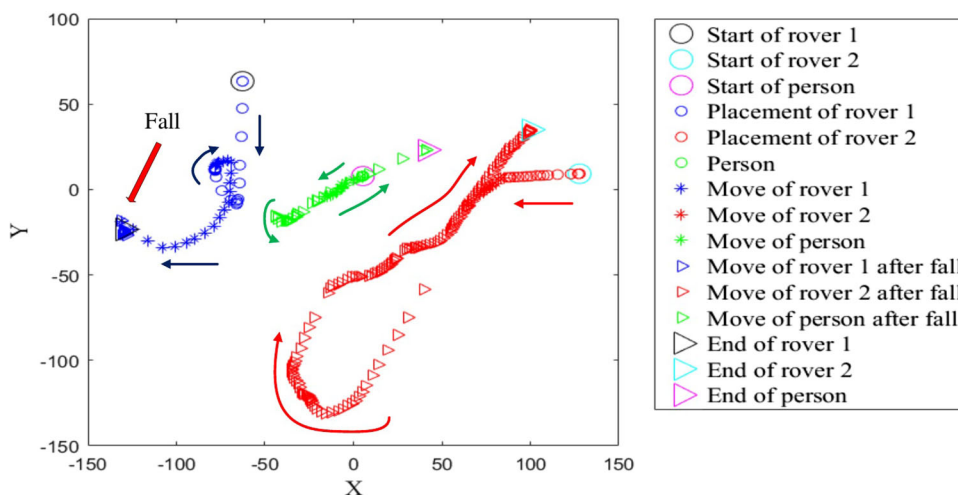
### 5 Experimental results

Considering that the evaluation is executed indoor, only some selected situations defined previously are used. We will present one of them in the next part and the result given by the



**Fig. 8** CNN for state detection

**Fig. 9** Movements of the rovers and the migrants during the test



CNN for all the four situations detected. Two reduced scale rovers were used during these tests. A reduced scale rover is enough to achieve a clear demonstration of the suggested architecture and algorithms for this critical application.

### 5.1 Swarm of rover reaction

In this part, we present the results obtained during the fall of one of the two rovers given the mission to protect one migrant. Then, the operational rover adapts itself and changes its behavior to lead the mission.

In order to test our system, we used an indoor area of 3 × 4 m surrounded by eight cameras, which allow us to know the position of the simulated migrants and the rovers. Due to the room configuration, we set the minimal distance between the rovers and the simulated migrants to 75 cm. This choice gives the rovers enough space to move without annoying the simulated migrants.

The motion of the simulated migrants and the rovers during this test is described as follows:

- (1) The rovers surround the migrant. Once they are around him, the migrant moves.
- (2) The rovers follow him, and then the leader in front of the group (Rover 1 in Fig. 9) falls into a hole. After the rover falls, the migrant turns to his left, and the operational rover (Rover 2 in Fig. 9) adapts itself to the situation, taking the place of Rover 1 in front of the migrant.
- (3) At the end, the migrant retraces his steps with the rover. This behavior is presented in Fig. 9.

The data of the IMU of each rover are sent to the server and used to detect the fall of the rover during its trajectory. Figure 10 refers to the rover in front of the migrant. It is composed of two parts: one part is the data of the IMU

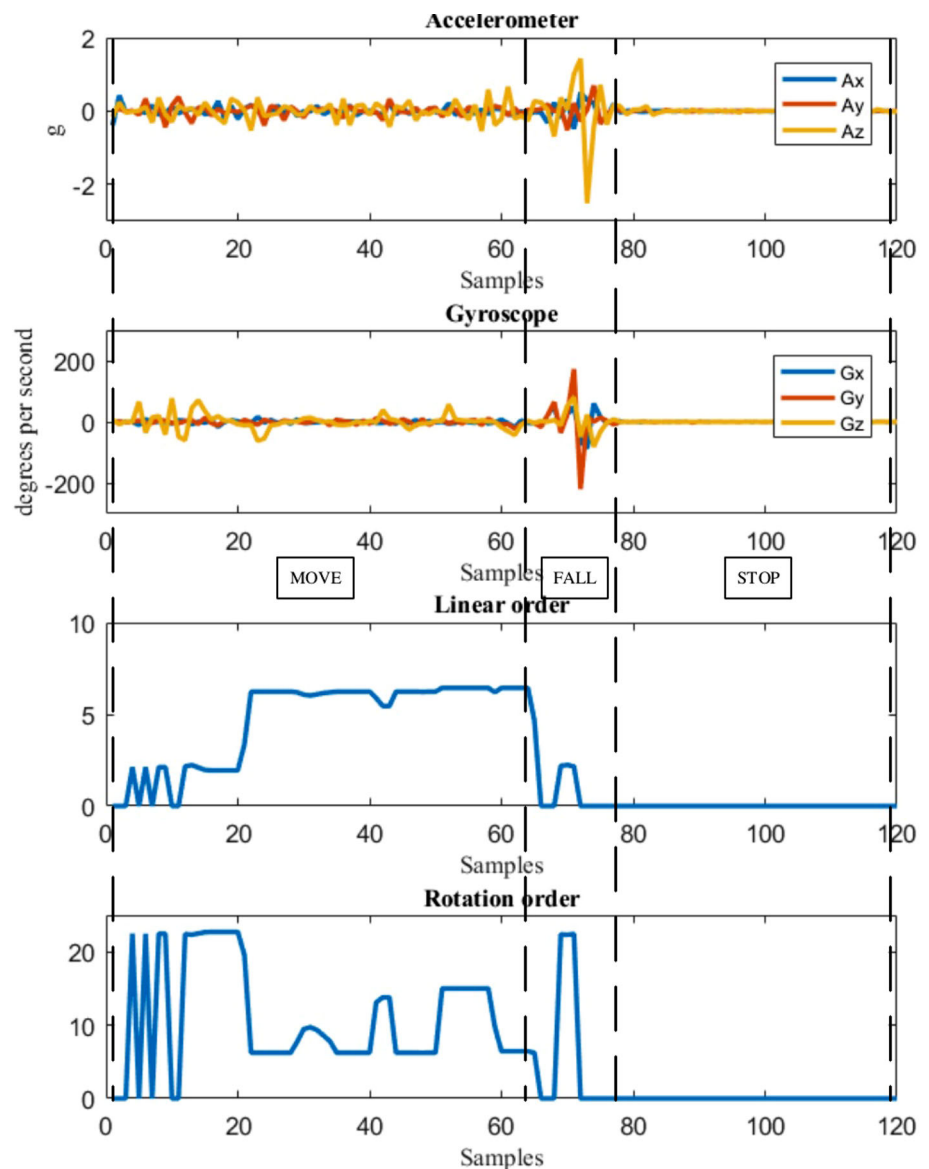
(accelerometer and gyroscope), and the second part is the command sent to the motors (linear speed and rotation). Three events are presented as the following:

- (1) The first event is the robot moving to reach its target position. As shown in Fig. 9, the blue path corresponds to this robot. It detects its target position and goes in front of the migrant.
- (2) The second event is the fall of the robot into a hole as detected by the CNN, which is very different from the first event.
- (3) The fall is detected by the CNN, which leads to the third event: the robot stops its motors because it is jammed. This reaction is observable with the command sent to the motor. Just after the fall, they are set to zero.

Figure 11 refers to the second robot that is behind the migrant at first. It is also composed of two parts: one part is the data of the IMU (accelerometer and gyroscope), and the second part is the distance to the targeted position and the command sent to the motors (linear speed and rotation). Three events are also presented:

- (1) The first event is the robot moving to reach its target position. The red path on Fig. 9 corresponds to this robot. It detects its target position and goes behind the migrant.
- (2) The second event is the fall of the robot in front of the migrant. When the fall is detected by the CNN, the fallen robot is removed from the swarm because it cannot continue its mission. The robot behind the migrant becomes the only robot in the “swarm,” and its targeted position changes. It should now be in front of the migrant. Therefore, the command sent to the motor changes quickly.

**Fig. 10** Interaction between Rover 1 and the migrant



- (3) The third event is that the second robot reaches its new position in front of the migrant. At sample 72, the distance to the target has increased because of the change in target. Therefore, the command of the linear speed increases to adapt the rover to this new destination.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

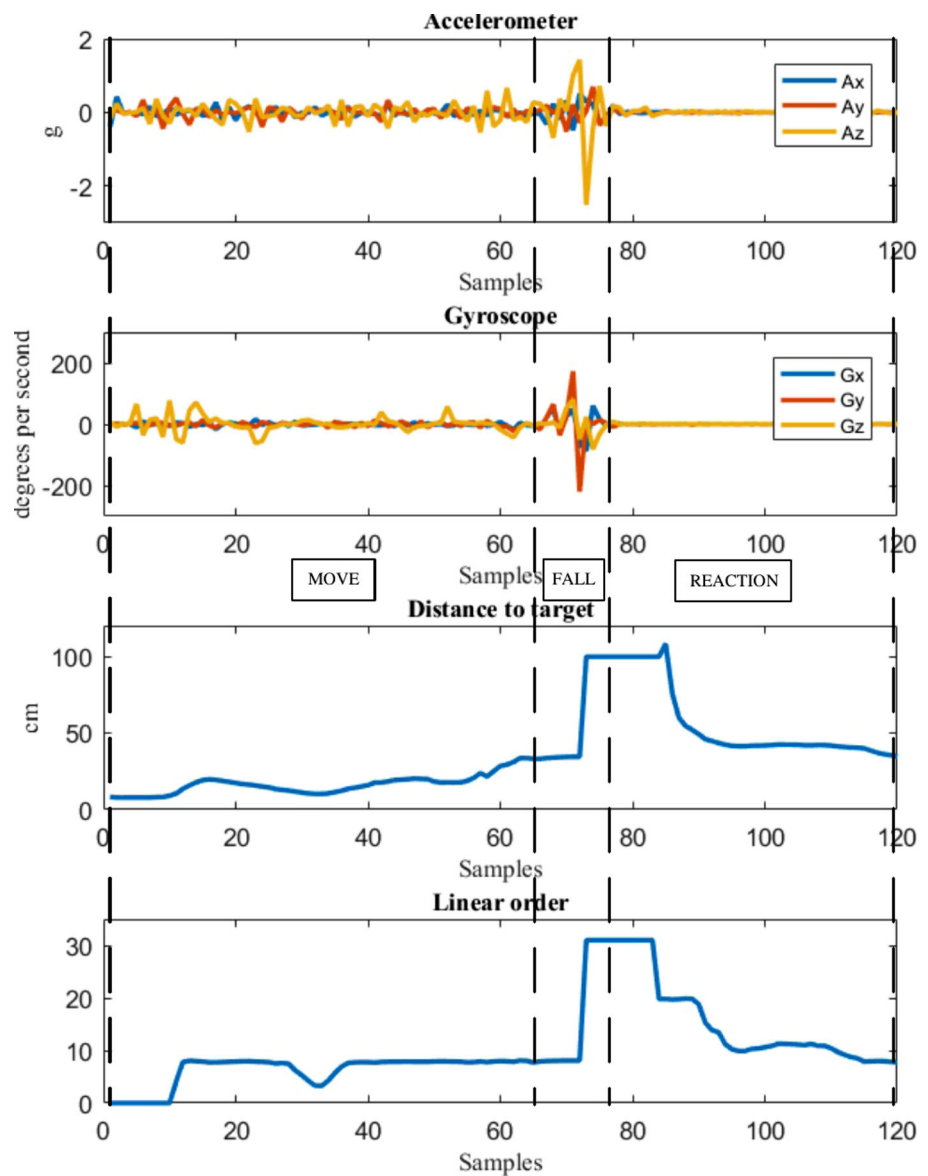
## 5.2 Performance of state estimation with CNN

In order to measure the performance of our CNN in classifying some situations, we calculate two indicators: (1) the precision (i.e. fraction of the relevant situations among those found) and (2) the recall (i.e. fraction of the relevant situations that were found over the total amount). Both measures are currently used to evaluate the performance of classification systems [73]. Their values can be computed using Eqs. (2) and (3):

where, TP is the number of true positives, FN is the number of false negatives and FP is the number of false positives.

To measure these two indicators, we used a dataset of 405 pictures obtained from the data given by the IMU: 223 for a rover in motion on stones, 87 for a collision between a rover and an obstacle, 52 for the fall of a rover, and 43 for a rover that skidded on sand. These data were not used for the training of the CNN in order not to bias the results. With the CNN described previously, we obtained the results shown in Table 4.

**Fig. 11** Detection of the fall and its effect on the Rover 2



The state of the rover is classified with precision and recall rates of the CNN between 91 and 100%. Despite not being able to recognize all states with 100% accuracy, our CNN architecture can provide some information with several benefits to the swarm.

Compared to [49], which detected the fall of a robot with a multi-perceptron layer, our results seem better. Our suggested algorithm does not detect only a fall, and the kind of robot is different (humanoid vs. rover). Regarding 52 falls, we obtained a recall and precision of 98%. In [49], at the end of their training, they obtained a precision of 89.84% and a recall of 98.37%. Our method seems to give better results in the estimate of true falls. This is probably due to the fact that we used nine features (five from accelerometer and four from gyroscope) to detect a fall, while other research projects used only two inputs from the gyroscope.

Also, the processing time is very short. Therefore, the swarm reacts quickly to an issue. Further, we can add as many situations as we want to be detected as long as we have data to make the training of the CNN. Then, the swarm will be able to adapt itself to many situations. Having a real-time monitoring system in place to monitor rover state in an outdoor environment provides a mechanism to improve rover behaviors and increase the protection of migrants by the swarm.

## 6 Conclusions

The follow-up of a group of migrants using rovers (mobile robots) is challenging regarding the autonomy of the swarm in the interaction between migrants and rovers. This project

**Table 4** Detection results with a testing dataset and a threshold of 0.6

Situations	Number in data	Correctly detected situations	Incorrectly detected situations		Precision	Recall
			Not detected (FN)	Mis-detected (FP)		
Stones	223	221	1	1	99.5	99.5
Collision	87	79	1	7	91.9	98.8
Fall	52	50	1	1	98	98
Skid	43	43	0	0	1	1

concerns mainly the integration of trajectory planning using convex hulls to provide a safety area around the migrants as well as a strategy to manage the swarm. Some of these methods are used in very specific domains but have never been implemented in a mission for the protection of migrants.

The rover's states, identified by using a CNN, allow us to follow their possible issues and to improve the rover's behavior. This enables the swarm to adapt itself to the environment through its evolution. In this research work, we were able to validate and demonstrate the effectiveness of the approach, both at the level of the logic of the system and its response. Through this evaluation, some improvements can be suggested such as localization of migrants using a fixed wings UAV. Geo-localization of migrants is a very difficult task since it mainly depends on differentiation of migrants from other objects or obstacles in the environment.

Furthermore, better models for positioning robots around the group could be studied, implemented, tested and evaluated. Instead of arranging them in a uniform way, we could position them, for example to cover a zone more dangerous than the others. Improvements in the algorithm of planning can also take place. One could seek to validate the path to its final target by verifying that we do not encounter problems with trajectory oscillations.

## 7 Future works

As this application is critical and should require the use of human participants with a research ethical approval from a Research Ethics Board (REB) for an adequate performance evaluation, this paper presents only the overall design, including a process that can be used for such an application. Of course, using human participants will demonstrate the commercial version of this research work. Moreover, many other experiments could be performed: (1) drone and migrants detection and localization, (2) drone tactical autonomous decision for target following, (3) true interaction with participants and (4) gas filling strategy of the swarm by a third party drone. Each of these projects will be presented in other research works.

Finally, it should be interesting to identify some motion patterns of the group in order to anticipate their trajectory and help the swarm optimize their position for improving protection. Some issues related to the state of a rover should be solved by the swarm itself by cooperation between the rovers. For example, when one of them has a difficulty, we can imagine other rovers helping it. This could avoid the intervention of an operator.

These developments will continue to refine the SROPRAS system of protecting a group of migrants with a swarm of robots in order to avoid operators' intervention in dangerous situations.

**Acknowledgements** We would like to thank the Department of Applied Sciences, UQAC, Canada for allowing access to the rovers of the LAR.i Laboratory. Francis Deschênes and Danny Ouellet gave us precious advices related to the technical design and maintenance of the rovers. While performing this project, Maxime Vaidis received a scholarship from REPARTI Strategic Network supported by Fonds québécois de la recherche sur la nature et les technologies (FRQ-NT). This work is financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), Discovery grant, under Grant Number RGPIN-2018-06329.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Gisin L, Doran HD, Gruber JM (2016) RoboBAN: a wireless body area network for autonomous robots. In: ICINCO 2016—proceedings of the 13th international conference on informatics in control, automation and robotics, vol 2, pp 49–60
- Salayma M, Al-Dubai A, Romdhani I, Nasser Y (2017) Wireless body area network (WBAN): a survey on reliability, fault tolerance, and technologies coexistence. In: ACM computing surveys, 50(1), Art. no. 3

3. Yi WJ, Saniie J (2013) Smart mobile system for body sensor network. In: IEEE international conference on electro information technology
4. Paschalidis IC, Dai W, Guo D (2014) Formation detection with wireless sensor networks. *ACM Trans Sens Netw* 10(4), Art. no. 55
5. Kim MS, Kim SH, Kang SJ (2017) Middleware design for swarm-driving robots accompanying humans. *Sens Switz* 17(2), Art. no. 392
6. Garzón M, Valente J, Roldán JJ, Cancar L, Barrientos A, Del Cerro J (2016) A multirobot system for distributed area coverage and signal searching in large outdoor scenarios\*. *J Field Robot* 33(8):1087–1106
7. Kamegawa T, Sato N, Hatayama M, Uo Y, Matsuno F (2011) Design and implementation of grouped rescue robot system using self-deploy networks. *J Field Robot* 28(6):977–988
8. Mouradian C, Sahoo J, Glitho RH, Morrow MJ, Polakos PA (2017) A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters. In: 2017 13th International wireless communications and mobile computing conference (IWCMC 2017), pp 1909–1914
9. Amanatiadis A, Bampis L, Karakasis EG, Gasteratos A, Sirakoulis G (2018) Real-time surveillance detection system for medium-altitude long-endurance unmanned aerial vehicles. *Concurr Comput Pract Exp Concurr Comput* 30(7):e4145
10. Khaleghi AM et al (2013) A DDDAMS-based planning and control framework for surveillance and crowd control via UAVs and UGVs. *Expert Syst Appl* 40(18):7168–7183
11. Sara M, Jian L, Son Y-J (2015) Crowd detection and localization using a team of cooperative UAV/UGVs. In: Proceedings of the 2015 industrial and systems engineering research conference
12. Stival F, Michieletto S, De Agnoi A, Pagello E (2018) Toward a better robotic hand prosthesis control: using EMG and IMU features for a subject independent multi joint regression model. In: Proceedings of the IEEE RAS and EMBS international conference on biomedical robotics and biomechanics, pp 185–192
13. Ishac K, Suzuki K (2017) Gesture based robotic arm control for meal time care using a wearable sensory jacket. In: IRIS 2016—2016 IEEE 4th international symposium on robotics and intelligent sensors: empowering robots with smart sensors, pp 122–127
14. Yi C, Ma J, Guo H, Han J, Gao H, Jiang F, Yang C (2018) Estimating three-dimensional body orientation based on an improved complementary filter for human motion tracking. *Sensors* 18:3765
15. Chan TK, Yu YK, Kam HC, Wong KH (2018) Robust hand gesture input using computer vision, inertial measurement unit (IMU) and flex sensors. In: 2018 IEEE international conference on mechatronics, robotics and automation (ICMRA 2018), pp 95–99
16. Ding S, Ouyang X, Liu T, Li Z, Yang H (2018) Gait event detection of a lower extremity exoskeleton robot by an intelligent IMU. *IEEE Sens J* 18(23):9728–9735 Art. no. 8469017
17. Caramia C et al (2018) IMU-based classification of Parkinson's disease from gait: a sensitivity analysis on sensor location and feature selection. *IEEE J Biomed Health Inform* 22(6):1765–1774 Art. no. 8434292
18. Alatise MB, Hancke GP (2017) Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended Kalman filter. *Sensors* 17:2164
19. Alessandro F, Niko G, Simone N, Pallottino L (2016) Indoor real-time localisation for multiple autonomous vehicles fusing vision, odometry and IMU data. *Model Simul Auton Syst* 9991:288–297
20. Li J, Bi Y, Li K, Wang K, Lin F, Chen BM (2018) Accurate 3D localization for MAV swarms by UWB and IMU fusion. In: IEEE international conference on control and automation (ICCA, 2018), pp 100–105
21. Salih SQ, Alsewari ARA, Al-Khateeb B, Zolkipli MF (2019) Novel multi-swarm approach for balancing exploration and exploitation in particle swarm optimization. *Adv Intell Syst Comput* 843:196–206
22. Sánchez-García J, Reina DG, Toral SL (2018) A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario. *Fut Gener Comput Syst* 90:129–148
23. Garcia-Aunon Pablo, Cruz AB (2018) Comparison of heuristic algorithms in discrete search and surveillance tasks using aerial swarms. *Appl Sci* 8(5):711
24. de Moraes RS, Freitas EPd (2017) Distributed control for groups of unmanned aerial vehicles performing surveillance missions and providing relay communication network services. *J Intell Robot Syst Theory Appl* 92(3–4):645–656
25. Din A, Jabeen M, Zia K, Khalid A, Saini DK (2018) Behavior-based swarm robotic search and rescue using fuzzy controller. *Comput Electr Eng* 70:53–65
26. Ranaweera DM, Hemapala KTM Udayanga, Buddhika AG, Jayasekara P (2018) A shortest path planning algorithm for PSO base firefighting robots. In: Proceedings of the 4th IEEE international conference on advances in electrical and electronics, information, communication and bio-informatics (AEEICB 2018)
27. Kapellmann-Zafra G, Chen J, Groß R (2016) Using Google glass in human-robot swarm interaction. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 9716, pp 196–201
28. Abich J, Barber DJ (2017) The impact of human robot multimodal communication on mental workload, usability preference, and expectations of robot behavior. *J Multimodal User Interfaces* 11(2):211–225
29. Hacothen S, Shoval S, Shvalb N (2017) Multi agents' multi targets mission under uncertainty using probability navigation function. In: IEEE international conference on control and automation (ICCA), pp 845–850
30. Zhen W, Kang X, Zhang X, Dai J (2016) Gait planning of a novel metamorphic quadruped robot. *Jixie Gongcheng Xuebao/J Mech Eng* 52(11):26–33
31. Li L, Wang D, Wang P, Huang J, Zhu D (2015) Soil surface roughness measurement based on color operation and chaotic particle swarm filtering. *Nongye Jixie Xuebao/Trans Chin Soc Agric Mach* 46(3):158–165
32. Alonso-Mora J et al (2015) Gesture based human—multi-robot swarm interaction and its application to an interactive display. In: Proceedings of the IEEE international conference on robotics and automation
33. Walker P et al (2014) Human control of robot swarms with dynamic leaders. In: IEEE international conference on intelligent robots and systems
34. Nagi J et al (2014) Human-swarm interaction using spatial gestures. In: IEEE international conference on intelligent robots and systems
35. Zhang L, Vaughan R (2016) Optimal robot selection by gaze direction in multi-human multi-robot interaction. In: IEEE international conference on intelligent robots and systems
36. Bevacqua G et al (2015) Mixed-initiative planning and execution for multiple drones in search and rescue missions. In: Proceedings international conference on automated planning and scheduling (ICAPS)
37. Mtshali AE Mbali (2010) Robotic architectures. *Def Sci J* 60(1):15–22
38. West Andrew, Arvin Farshad, Martin Horatio, Watson Simon, Lennox B (2018) ROS integration for miniature mobile robots. *Towards Auton Robot Syst* 10965:345–356
39. Straszhaim T, Gerkey B, Cousins S (2011) The ROS build system. *IEEE Robot Autom Mag Short Surv* 18(2), Art. no. 5876218
40. Conte G, Scaradozzi D, Mannocchi D, Raspa P, Panebianco L, Screpanti L (2018) Development and experimental tests of a ROS multi-agent structure for autonomous surface vehicles. *J Intell Robot Syst Theory Appl* 92(3–4):705–718

41. Veloso MVD, Filho JTC, Barreto GA (2017) SOM4R: a middleware for robotic applications based on the resource-oriented architecture. *J Intell Robot Syst* 87(3–4):487–506
42. Hönig W, Ayanian N (2017) Flying multiple UAVs using ROS. *Stud Comput Intell* 707:83–118
43. Otsuka A et al (2015) Algorithm for swarming and following behaviors of multiple mobile robots. In: IECON2015—41st annual conference of the IEEE industrial electronics society
44. Wu ZS, Fu WP (2014) Review of path planning method for mobile robot. *Adv Mater Res* 1030–1032:1588–1591
45. Lv W, Kang Y, Qin J (2019) Indoor localization for skid-steering mobile robot by fusing encoder, gyroscope, and magnetometer. *IEEE Trans Syst Man Cybern Syst* 49(6):1241–1253
46. Zhang K, Niroui F, Ficocelli M, Nejat G (2018) Robot navigation of environments with unknown rough terrain using deep reinforcement learning. In: 2018 IEEE international symposium on safety, security, and rescue robotics (SSRR 2018)
47. Geng M, Li Y, Ding B, Wang AH (2018) Deep learning-based cooperative trail following for multi-robot system. In: Proceedings of the international joint conference on neural networks
48. Mišeikis J et al (2018) Robot localisation and 3D position estimation using a free-moving camera and cascaded convolutional neural networks. In: IEEE/ASME international conference on advanced intelligent mechatronics, AIM, pp 181–187
49. Hofmann M, Schwarz I, Urbann O, Ziegler F (2016) A fall prediction system for humanoid robots using a multi-layer perceptron. In: 10th International Cognitive Robotics Workshop (CogRob-2016), vol 3, pp 3–6
50. Mascret Q, Biemann M, Fall CL, Bouyer LJ, Gosselin B (2018) Real-time human physical activity recognition with low latency prediction feedback using raw IMU data. In: Proceedings of the annual international conference of the IEEE engineering in medicine and biology society, EMBS, pp 239–242
51. Ma Y et al (2017) Hand gesture recognition with convolutional neural networks for the multimodal UAV control. In: 2017 Workshop on research, education and development of unmanned aerial systems, RED-UAS 2017, pp 198–203
52. Fakhruddin AH, Fei X, Li H (2017) Convolutional neural networks (CNN) based human fall detection on body sensor networks (BSN) sensor data. In: 2017 4th International conference on systems and informatics (ICSAI, 2018)-Janua(Icsai), pp 1461–1465
53. Zhu R, Xiao Z, Li Y, Yang M, Tan Y, Zhou L, Wen H (2019) Efficient human activity recognition solving the confusing activities via deep ensemble learning. *IEEE Access* 7:75490–75499
54. Quinonez Yadira, Ramirez Mario, Lizarraga Carmen, Tostado Ivan, Bekios J (2015) Autonomous robot navigation based on pattern recognition techniques and artificial neural networks. *Bioinspir Comput Artif Syst* 9108:320–329
55. Saab W, Rone WS, Ben-Tzvi P (2018) Robotic tails: a state-of-the-art review. *Robot Rev* 36(9):1263–1277
56. Yie Y, Solihin MI, Kit AC (2017) Development of swarm robots for disaster mitigation using robotic simulator software. In: Ibrahim H, Iqbal S, Teoh SS, Mustafa MT (eds) 9th International Conference on Robotic, Vision, Signal Processing and Power Applications. Springer, Singapore, pp 377–383
57. Kapellmann-Zafra G et al (2016) Human–robot swarm interaction with limited situational awareness. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), pp 125–136
58. Bamberger RJ et al (2004) Wireless network communications architecture for swarms of small UAVs. In: Collection of technical papers—AIAA 3rd “Unmanned-Unlimited” technical conference, workshop, and exhibit
59. Nowak S, Krüger T, Matthaei J, Bestmann U (2013) Martian swarm exploration and mapping using laser SLAM. In: International archives of the photogrammetry, remote sensing and spatial information sciences—ISPRS archives, vol 40, pp 299–303
60. N.A.a.S. Administration (2019) Nasaswarmathon. <http://nasaswarmathon.com/>. Accessed 15 Oct 2019
61. Minaeian S, Liu J, Son YJ (2015) Crowd detection and localization using a team of cooperative UAV/UGVs. In: IIE annual conference and expo, pp 595–604
62. Minaeian S (2017) Effective visual surveillance of human crowds using cooperative unmanned vehicles. In: Proceedings—Winter simulation conference, pp 3674–3675
63. Lee G, Chong NY, Christensen H (2010) Tracking multiple moving targets with swarms of mobile robots. *Intell Serv Robot* 3(1):61–72
64. Pannetier B, Moras J, Dezert J, Sella G (2014) Study of data fusion algorithms applied to unattended ground sensor network. In: Proceedings of the SPIE—the international society for optical engineering, 9091, art. no. 909103
65. Sugihara K (1994) Robust gift wrapping for the three-dimensional convex hull. *J Comput Syst Sci* 49(2):391–407
66. Kong X, Everett H, Toussaint G (1990) The Graham scan triangulates simple polygons. *Pattern Recognit Lett* 11(11):713–716
67. Chan TM, Chen EY (2010) Optimal in-place and cache-oblivious algorithms for 3-d convex hulls and 2-d segment intersection. *Comput Geom Theory Appl* 43(8):636–646
68. Siddaiyan S, Arokiasamy RW (2012) DVFH—VFH\*: reliable obstacle avoidance for mobile robot navigation coupled with A\*algorithm through fuzzy logic and knowledge based systems. Presented at the international conference on computer technology and science (ICCTS), Singapore
69. Benbouabdallah K, Qi-dan Z (2013) A fuzzy logic behavior architecture controller for a mobile robot path planning in multi-obstacles environment. *Res J Appl Sci Eng Technol* 5(14):3835–3842
70. Bayar V, Akar B, Yayan U, Yavuz HS, Yazici A (2014) Fuzzy logic based design of classical behaviors for mobile robots in ROS middleware. Presented at the 2014 IEEE international symposium on innovations in intelligent systems and applications (INISTA) proceedings
71. Peng ST, Sheu JJ (2004) An anti-skidding control approach for autonomous path tracking of a 4WS/4WD vehicle. In: 2004 5th Asian control conference, vol 1, pp 617–622
72. Hasan KM, Abdullah-AI-Nahid Reza KJ (2014) Path planning algorithm development for autonomous vacuum cleaner robots. In: 2014 International conference on informatics, electronics and vision (ICIEV), pp 1–6
73. Powers DMW (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J Mach Learn Technol* 2(1):37–63

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.