

Article

Implementation of Machine Learning Algorithms on Multi-Robot Coordination

Tuncay Yiğit [†]  and Şadi Fuat Çankaya ^{*,†} 

Faculty of Engineering, Computer Engineering, Süleyman Demirel University, Isparta 32260, Türkiye; tuncayyigit@sdu.edu.tr

* Correspondence: sadicankaya@sdu.edu.tr; Tel.: +90-0545-398-8000

† These authors contributed equally to this work.

Abstract: Occasionally, professional rescue teams encounter issues while rescuing people during earthquake collapses. One such issue is the localization of wounded people from the earthquake. Machines used by rescue teams may cause crucial issues due to misleading localization. Usually, robot technology is utilized to address this problem. Many research papers addressing rescue operations have been published in the last two decades. In the literature, there are few studies on multi-robot coordination. The systems designed with a single robot should also overcome time constraints. A sophisticated algorithm should be developed for multi-robot coordination to solve that problem. Then, a fast rescuing operation could be performed. The distinctive property of this study is that it proposes a multi-robot system using a novel heuristic bat-inspired algorithm for use in search and rescue operations. Bat-inspired techniques gained importance in soft-computing experiments. However, there are only single-robot systems for robot navigation. Another original aspect of this paper is that this heuristic algorithm is employed to coordinate the robots. The study is devised to encourage extended work related to earthquake collapse rescue operations.

Keywords: multi-robot coordination; navigation; bat-algorithm; heuristic algorithms



Citation: Yiğit, F.; Çankaya, Ş.F. Implementation of Machine Learning Algorithms on Multi-Robot Coordination. *Electronics* **2022**, *11*, 1786. <https://doi.org/10.3390/electronics11111786>

Academic Editor: Jaha Ryu

Received: 27 April 2022

Accepted: 1 June 2022

Published: 4 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the event of a natural disaster, rescue teams race against time. The chance of survival of the surviving injured is significantly reduced if not rescued in 72 h [1]. In disaster areas, the work of rescuers is challenging because entering such areas can be dangerous for the lives of the injured. Therefore, rescue teams have a limited time window for search and rescue activities. In such cases, the work of rescue teams can be assisted by using robot technology [2]. The use of heavy machinery in collapsed structures poses a significant risk for the injured trying to survive due to the inadequacy of the space between pieces of rubble. The spot-location of those under the debris is tried to be determined by the combined and separate use of search dogs, cameras, and listening devices. Many robot systems that use wireless communication technology have been used for search and rescue activities in recent years [3–5]. These robots should transmit vital data to the relevant people accurately and quickly to speed up search and rescue activities. However, robots used for search and rescue are costly, and there is no government incentive or funding for the construction and production of such robots [6]. Therefore, this is not an option for most underdeveloped countries. The time constraint and difficulty of post-earthquake search and rescue operations underline the need for a less costly and fast-operating casualty location detection system. Robots developed using Arduino boards are not very old [7,8]. There are many documents and books on the usage of these boards. In addition, the sensors compatible with Arduino boards, such as infrared, distance, humidity, and temperature sensors, are inexpensive and accessible because of their widespread use [9]. In this study, a team consisting of multiple search and rescue robots was designed; a multi-robot coordi-

nation algorithm based on the Bat Algorithm (BA), one of the nature-inspired optimization algorithms that have expanded in recent years, was tested on these robots.

Studies on BA have shown that this algorithm gives better results than the others, such as particle swarm optimization, firefly algorithm, and harmonic search [10]. It is expected that the study will contribute significantly to the literature with the improvements made for pathfinding and robot coordination. There are studies on using robot technology for search and rescue efforts in disaster situations. These studies generally performed e-puck or Arduino-based improvements [11]. Various sensing and wireless technologies are used in disaster areas, including remote and image control [12]. Rescue teams can remotely control search and rescue robot systems using these technologies when natural disasters occur instead of entering the devastated area [12]. However, there are very few studies in our country on determining the location of the people under collapsed buildings after an earthquake using robots. Furthermore, BA's prevalence and usage areas increase day by day. Its popularity has increased even more because many studies have proved its efficiency and effectiveness. Thus, a bat-inspired algorithm was desired to be developed for multi-robot coordination.

This study has two objectives. The first is to develop a system that can detect the location of people in rubbles after an earthquake. The second is to develop a new multi-robot coordination system based on a bat-inspired algorithm. In this section, these two objectives and sub-objectives are detailed. One of the study's objectives is to determine the people's positions in rubbles with a low error margin after the earthquake. Some coercions limit manageability in signal transmission. However, the signal enhancer and signal filtering techniques used in the study helped overcome these issues. Another objective is to achieve the desired objective with a robot system that is less costly than existing systems. Thus, an Arduino-based robot platform, which is lower than an e-puck in cost, is preferred in the study.

Multi-robot coordination is a method used in situations where more than one autonomous robot is used for operations such as navigation, mapping, and patrolling [12]. There are two forms of coordination: centralized and decentralized [13]. The studies should be evaluated according to these features. Developing a bat-inspired algorithm for multi-robot coordination is one of the main objectives. The desired result is achieving a method superior to its counterparts in terms of time, complexity, and effectiveness. The study aimed to accelerate search and rescue activities in natural disasters. The multi-robot coordination algorithm developed in this study will set an example for future studies in this field. Sheng et al., offered a multi-robot coordination model for limited communication situations [14]. Perception and mapping were the key aspects of the model. The direction of the robots was determined using an information acquisition function. In addition, the study contained a special algorithm for map synchronization. The method was tested with six robots in a simulation setting and successfully used the available data. However, this study did not compare the method used for coordination with other methods. The outcomes included only the communication range and the test results [15]. Talebpour et al. [15] examined multi-robot coordination in dynamic environments containing humans. Multi-robot coordination was also used to map unknown environments [16]. The algorithms for searching and finding the optimum solution, such as A* and greedy, were used. They failed to reduce the number of idle robots in coordination entirely. Huang et al. [17] developed an algorithm that scans an area and measures the robot's exploration performance for an unknown environment.

Kandhasamy et al., developed a route planning method for decentralized multi-robot control and supported it with a special messaging system [18]. The robot positioning and mapping system developed by Tchuiev et al., produced promising results in simulation environments regarding time [19]. However, the performance should be remeasured in environments with factors affecting communication between robots. In addition, it would be more effective to compare the system with similar methods instead of evaluating it only

in itself. One of the collaboratively developed methods for solving complex problems was found in the study of Queralta et al. [20].

There are many studies in the literature about rescue robots. One of them presented an improved network system for the rescue robot [21]. In addition, although it produced successful results in a simulation environment, it was not a sound system for multiple robots. Rescue zones are not entirely flat; therefore, it is necessary to develop robot systems that overcome the obstacles. Bai et al., designed a new transformable wheeled-legged robot mechanism for search and rescue missions in complex terrains. There is a need for such systems in this field.

Moreover, collaboratively using more than one robot is desirable in search and rescue activities [22]. Joseph et al., developed an Arduino-based rescue robot to detect people stranded in land or water areas affected by natural disasters, serving the additional purpose of carrying emergency food and medical aid to people [23]. The performance of such robots is limited when used for both search and rescue purposes. Human intervention is needed in cases where robot capability is limited. Sound pickups are also required on rescue robots. Mae et al., developed rescue robots for search and rescue operations in times of large-scale disaster. The robot they developed is used to search for survivors in disaster areas by capturing their voices with its microphone array. Performing in a simulated disaster area, they confirmed that a survivor could sense the direction of their position using a noise reduction technique [24]. One of the improvements made on Arduino boards belongs to Kiran et al. [25]. They have developed a search robot equipped with various sensors that can be controlled via wireless using Arduino Board (Mega and Uno). The system was cost-effective, but a user should manage it entirely, which is its downside. Today, there is a need for autonomous robots that can act independently and collect vital information from a single point. Buzduga et al., developed a system that detects the number of people in earthquake rubbles using Arduino Uno [26]. The stability of the developed system was insufficient due to the distance constraints of the sensors used. Akkhar et al., offered a robot system developed for search and rescue purposes [27]. The system developed with Arduino Mega could be controlled remotely with Bluetooth and includes an arm with good mobility. The system was not self-directing and did not work for distances longer than 10 m. An algorithm that plans robot paths was developed in 2018 [28]. It investigated the possibility of using a new evolutionary-based technique based on Teaching-Learning Based Optimization (TLBO) to solve a mobile robot's navigation problem in a wild environment. The results of the algorithm were validated by comparing them with the outcomes of other intelligent algorithms such as particle swarm optimization (PSO), weed optimization (IWO), and biogeography-based optimization (BBO). The heuristic Q-learning method was presented in 2015 [29]. This method was richly compared with different methods. However, this study has tested the robot in a known setting. The method should be renewed for a completely unknown setting and multi-robots. The ABACO algorithm [30], developed by Ajeil et al., based on the ant colony algorithm, was tested in many experimental settings and compared with many methods. However, multi-robot coordination was not considered in this study.

Regarding theoretical richness, one of the significant studies tried to establish multi-robot coordination with hierarchical clustering [31]. This study is different from the others in terms of the originality of the method and simulation. In Muthukumaran et al. [32], a new meta-heuristic optimization technique called Dragonfly Algorithm (DA) is used to navigate an autonomous mobile robot in an unknown complex setting full of static obstacles. Various static settings were modeled in the method, and the algorithm was tested both in simulation and experimentally. They showed that the robot reached the target with the proposed algorithm without hitting any obstacles and created a smooth optimal trajectory. Apart from Arduino, the E-puck robot system has also been used in robot navigation studies [33]. One of its essential advantages is that the sensors that will be needed are integrated. The cost of this robot, which occupies a small space, is higher than Arduino. Vijay Kumar and Dinesh Kumar presented a comprehensive review of the Firefly Algorithm

(FA) [34]. They examined various variants of FA, such as binary, multi-objective, and hybrid, with other meta-heuristics. They provided applications and performance improvement metrics. Luke et al., simulated the results using binary classifiers to determine performance criteria in classifying unbalanced datasets and developed a systematic analysis based on the binary confusion matrix [35].

Gürgöze et al. [36] researched and comparatively analyzed algorithms used in robotic systems. Ahmet Sami Dog̃ru and Tolga Eren examined the nature algorithms [37]. They concluded that the error rate of BA is less than other algorithms and gives more accurate results. Lee et al., performed a sinogram-based map assembly study [38].

Ramm et al., explained applied tomography's theoretical, computational, and practical aspects with Radon transform [39]. Huang presented a brief review of different optimization strategies used in the simultaneous localization and mapping (SLAM) problem [40]. Jain et al., provided an overview of research progress on Particle Swarm Optimization (PSO) between 1995–2017. They discussed the progress, improvements, modifications, and applications of PSO [41]. Mirjalili explained the general structure of a genetic algorithm and applied it to several studies to observe its performance [42]. Yang et al., reviewed BA and its derivatives and examined some of the sampled real-world optimization applications [43]. Moshayedi et al., discussed various shapes and applications of all existing service robots. They summarized the key points of the research progress, such as robot dynamics, types of robots, and different dynamic models of different types of service robots. The study can be accepted as a starting point for all researchers on this subject [44]. Moshayedi et al., investigated the relationship between the methods considered on the AGV robot instead of introducing the PID controller and seeing the effect of each parameter on the whole system. This review shows various PID tuning methods used according to system requirements through a humoral neural network called Lyapunov Direct Method, traditional Ziegler Nichols, and Fuzzy supervisory human immune system [45]. Moshayedi et al., reviewed the contributions and new applications of deep learning. The study aimed to summarize significant points so that academicians can analyze applications and algorithms. Also, the advantages of using the deep learning method and its hierarchical and non-linear operation were introduced and compared with traditional algorithms in typical applications [46]. Moshayedi et al., aimed to establish a much more efficient and cost-effective system with minimal effort through a secure SCADA for monitoring. They showed that having a territorial idea helps the system achieve the purpose of safe sensor monitoring without any outside interference. As a result, they showed a successful design and implementation of the zone idea and excellent use of the Raspberry Pi as a small and reliable mainframe [47]. Moshayedi et al., introduced and tested a simple algorithm for Persian vocabulary and implemented a robot structure based on Arduino. The critical point of the study was the use of sound signals in the Persian language, which has less than twenty years of history. The success rate exceeded 70% in the trials [48]. Fozuni et al., analyzed the stability and convergence of particle dynamics in the standard BA version, addressed the limitations, and proposed new update relations [49]. In addition, the dynamics of the algorithm were investigated, and sufficient stability conditions were obtained using Lyapunov stability analysis. Belge et al., used the hybrid HHO-GWO algorithm in their work, distinguished by its avoidance of local minimums and speed convergence, to successfully obtain a feasible and efficient path [50]. The experimental results showed that the proposed algorithm creates a fast and safe optimal path without getting stuck with local minima. The quadcopter followed the path created with minimum energy and time consumption.

This study compared machine learning algorithms for multiple robots that work concurrently. Thanks to data sharing between collaborative robots through sensors, they would not use the paths that other robots scanned, allowing them to reach the desired output sooner. Earthquake zones were chosen for testing machine learning algorithms. Regarding the literature, no configuration used BA in robot systems that work in integration with each other. From this point of view, it is expected that this study will take its place as an original study in the literature.

In this study, a robot system was designed within the scope of multi-robot coordination. First, a robot with more than one sensor was designed, and then algorithms deciding the robot's movement were developed. Studies on BA were reviewed. BA provided better results than the other algorithms covered in the study, which was a reason for preferring to use BA. Virtual and concrete settings have been built, and the robots determined the location and direction by working collaboratively. The performances of BA and some other machine learning algorithms were compared, and the results were interpreted.

2. Materials and Methods

The system for implementing multi-robot coordination is designed as follows: First, the robot was designed. Then, the Bat-Inspired Multi-Robot Coordination Algorithm (BIMRCA) was developed. The robot moved following algorithm rules. BIMRCA coordinated the robots moving in the determined area. Other machine learning algorithms were also tested in addition to BIMRCA. Machine learning algorithms and features that can be used in multi-robot coordination systems are listed below (Gürgüze et al., 2019).

1 Simultaneous Localization And Mapping (SLAM):

- The algorithm's operating principle:
 1. The proposed algorithm was observed to make better predictions in prolonged periods.
 2. The objective was to test whether the proposed algorithm could provide more convenient navigation than the ordinary PSO.
 3. It aimed to reach the destination in the most convenient way possible despite static obstacles.
 4. As a result, the proposed algorithm kept the path length and reached the target within the time limit.
- Performance against uncertain and changing environmental conditions:
 1. It is 6% more successful than methods such as genetics and fuzzy.
- Usage Areas:
 1. It is used in driverless cars, unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers, domestic robots and even inside the human body.

SLAM Algorithm moves the robot that maps the setting and uses that map to calculate its position. Figure 1 below shows how the SLAM algorithm works.

According to the figure above, all sensor data is read from the used robot in time "k". X represent the robot's position information, " U " control input information, m the entire set of signals, and Z marks observations. The formula generally used for the algorithm is below:

$$P(XK, m|Z0 : K, U0 : K, X0) \quad (1)$$

X is the robot's position information, U control input information, m is the entire cluster of signals, and Z is the entire mark observation cluster.

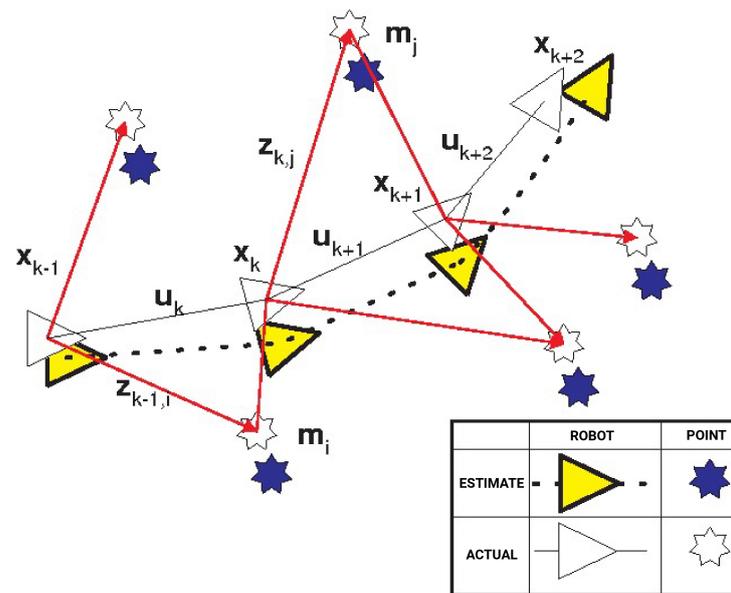


Figure 1. A picture of a gull.

2 Random Particle Swarm Optimization (RPSO):

- Its usage areas and formulation are the same as PSO.
- PSO is explained below.

3 Particle Swarm Optimization (PSO):

- Understanding the hand gestures using an SVM (support vector machine) optimized with particle swarm algorithm:
 - The success rate was high, around 77–81%, but it decreased to 65% when a small population was selected for PSO.
- Proper path planning:
 - The collision avoidance rate was higher in short periods
- The ability to predict the leader robot’s position in robot swarms and continue to move:
 - Despite the interruption between seconds 10 and 20, followers continued to follow the leader.
- Movement of a moving robot in a multi-robot and moving-target environment:
 - The time to reach the target was recorded with a maximum of 6.9 s without hitting any obstacles.
- Motion test in a closed and unfamiliar environment:
 - Showed a good movement performance with less ambient knowledge.
- Usage Areas:
 - It is used in many fields such as function optimizations, training artificial neural network models, fuzzy logic systems and image processing.

The formula of the algorithm is expressed as follows:

$$V_{k+1}^i = wv_k^2 + C_1rand \frac{(p^i - x_k^i)}{\delta t} + C_2rand \frac{(p_k^g - s_k^i)}{\delta t} \tag{2}$$

The velocity (V) of particle “ I ” at time $k + 1$ is given above. W refers to the inertia factor, C_1 is the particle’s coefficient, and C_2 is the swarm-related coefficient. P_i is the best point of the particle, and p_k^g is the best point of the swarm. These values are multiplied by a random number and summed up to determine the next move. The new position $X_k^i + 1$ is obtained with this movement [41]. The flowchart of the algorithm is shown in the Figure 2.

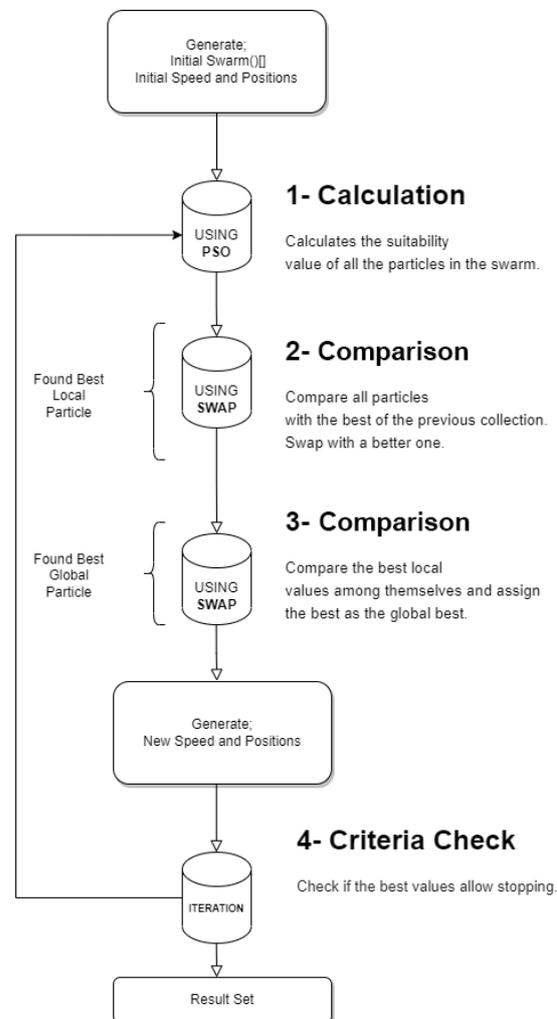


Figure 2. Flow Chart of PSO Algorithm.

4 Genetic Algorithm (GA):

- Starting point:
 - An improved GA approach was proposed to find suitable paths in multiple moving robots.
- Uptime and path planning:
 - Compared with algorithms such as A-star, PSO, PRM, and B-RRT, the uptime and path planning performance were better than theirs.
- Usage Areas:
 - There are three application areas: classification systems, practical industrial applications, and optimization in experimental studies.

The general implementation formula of the algorithm can be expressed with the following code (pseudocode) :
START

Create population
 Calculate Compliance
 START CYCLE
 (Selection)
 (Crossover)
 (Mutation)
 (Compute fitness)
 STOP CYCLE Is the termination criterion met?
 STOP

GA is a family of evolution-inspired computational models. These algorithms encode a possible solution to a particular problem on a simple chromosome-like data structure and apply recombination operators to these structures to preserve critical information. GA is often seen as a function optimizer, even though applied to many problems.

Process steps of GA are illustrated in Figure 3:

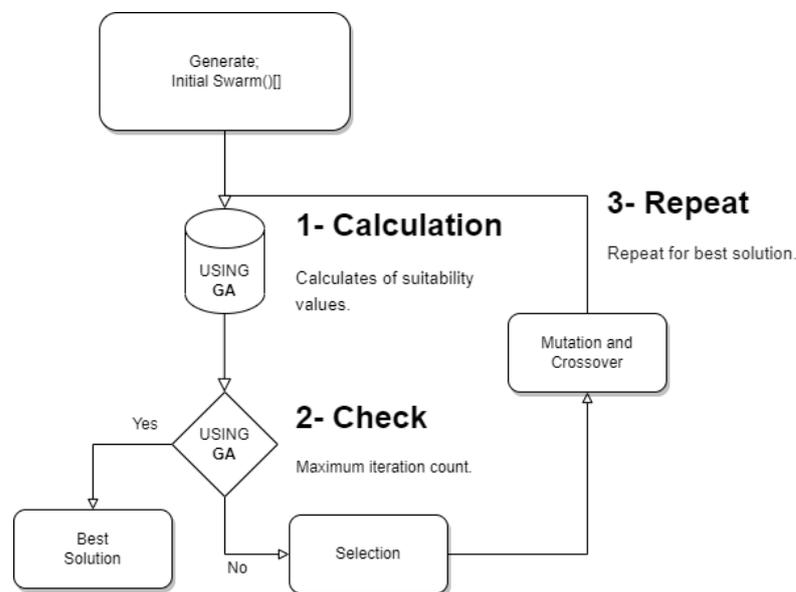


Figure 3. Processing Steps of GA.

5 Bat Algorithm (BA):

- Starting point:
 - Bats use echolocation to compute their distance from their prey and distinguish between prey/food and objects/obstacles.
 - Bats were observed to find their targets.
- Hybrid approach:
 - Performance efficiency was determined as 69.4% for GA, 74% for PSO, and 79.8% for GAPSO hybrid.
 - A hybrid approach including GA and PSO was offered to increase the performance of robots in industrial lines.
- Usage Areas:
 - It is successfully applied to solve problems in almost all optimization areas and appears to be very efficient.
- Stability Analysis:
 - Stability analysis was performed using the Lyapunov stability concept. Lyapunov stability analysis is based on the idea that if the total energy in the system continuously decreases, the system will asymptotically reach

the zero energy state associated with the equilibrium point. The results reveal that the Lyapunov energy function decreases with time in the stability range of the algorithm's parameters, and the particles' trajectory shows the asymptotic stability of the particle dynamics [49].

Bats gave appropriate values that affect the direction and subsequent movements of the entire population, randomly distributed in the area. Bats determine their next position depending on a specific frequency (f_i) and the solution value (x) of the best individual in the population for the speed (v_i) they generate. BA is formulated with the following equations [43]:

$$\begin{aligned} f_i &= f_{min} + (f_{max} - f) \\ v_{it} &= v_{it-1} + (x_{it} - x^*) \\ x_{it} &= x_{it-1} + v_{it} \end{aligned}$$

Accordingly, f_i represents the bat's frequency; f_{min} and f_{max} the minimum and maximum frequencies, respectively; x_{it} represents a randomly distributed value in the range $[0, 1]$; x^* represents the best individual value in the population at time t ; v_{it} , represents t : instance and the speed of the i . individual [43]. According to the above formulas, the algorithm's complexity in its most optimized form is $O(n)$. The operational steps of the algorithm can be illustrated as follows:

2.1. Robot Design

One of the most critical points considered in designing a robot is that the robot should move easily in uneven areas. Nowadays, many regions of the world are affected by natural disasters. Disasters include exceptional and unstoppable natural events such as earthquakes, forest fires, floods, or events triggered by human errors like building collapses. This situation reveals the importance of search and rescue robot systems in the emergency field [11]. The developed robot has sensors such as infrared, distance, temperature, RP-lidar, an encoder motor, and transceivers that support broadband communication for inter-robot communication. Figure 4 shows the picture of the equipped robot that has been designed and operated.

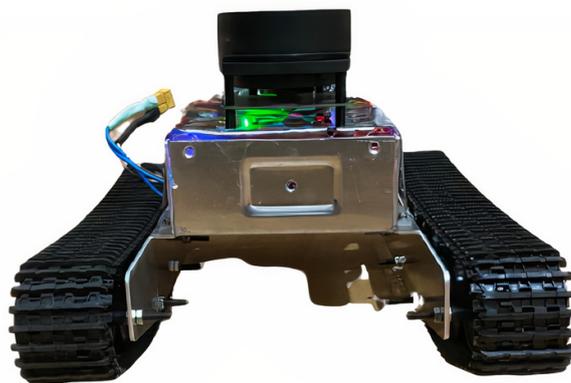


Figure 4. Sample of the Designed Robot.

The UGVs (Unmanned Ground Vehicles) used in real life are shown above. The robots were named NE-1001, SE-1002, and E1003, where NE—northeast, SE—southeast, and E—east. Mybot was used in the mapping of the simulated setting. Mybot is a robot with an RP-lidar on it. It is capable of listening to the developed software.

The hardware is made of all-steel material with proper holes to keep it cool. Thus, it is protected against falling, overturning, and overheating. The algorithms process the data

collected by the sensors, and the robot's behavior is decided after the design takes its final shape. If a robot has scanned a path during the search process, other robots do not repeat this task. Thus, the search is carried out faster. The information received from the sensors was first recorded in a database and then sent to the algorithm. A new task was defined for the robot that completes a task. After completing these steps, the software makes a decision based on the parameters collected by the robot searching for life under the debris and recognizing the human body. These parameters are temperature, the distance between the robot and the object, size, and motion. Figure 5 shows the robot reaching the target point using the distance and temperature data.

The algorithm ensures a new path is drawn by covering all the existing lines and checked boxes on the grid, as shown in the figure above. A path is set for the second robot, shown with a second line on the grid. The important thing here is to increase the starting points to enhance the communication between the robots and increase the endpoints. Increasing the number of robots, which also means increasing starting points, causes a competitive approach. On the other hand, the robot's movement was reviewed if direction changes occurred regarding the obstacle's size while the robot progressed towards the target. The times that the robot got out of the obstacles and moved in different directions regarding the obstacle size were recorded in three different settings.

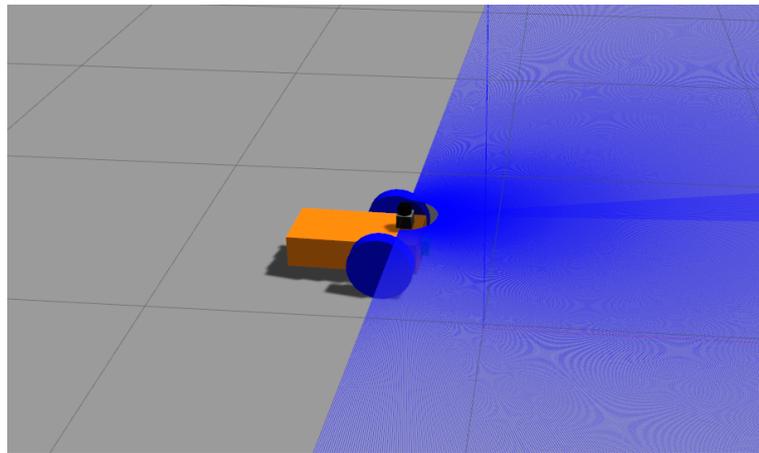


Figure 5. The Image of the Designed Robot in the Simulated Setting.

The times of determining a new direction for getting out of the obstacles in three different settings are given in Figure 6. The barrier density of the medium is calculated as in Equation (1). Accordingly, Environment 1 is an environment with an obstacle density of 25%. Environment 2's obstacle density is 50%, and Environment 3 has an obstacle density of 75%. Obstacle crossing time increases quadratically in Environment 3. The main reason is randomly generated motor speeds in the method while determining the new direction. The random motor speed generation function is repeated when they get the same values as the old direction, which increases the time to cross the obstacle. There is no quadratic increase for Environment 1 and Environment 2 because there are not enough obstacles to increase the obstacle clearance time in these two environments.

The algorithm developed in the robot's course of motion to any targeted destination is shared in Algorithm 1, where speed1 and speed2 are the variables used to change the robot's direction. Speed1 and speed2, which allow the robot to move straight, are set to 80. The robot constantly controls the distance between itself and the obstacle while moving on a straight course. If the variables that contain data from the ultrasonic sensors fall below 40 cm for distance and distance2, then the motor speed values directly become 0, and the robot is stopped. Subsequently, random motor speeds are assigned, and the robot is expected to cross the obstacle. The processes described so far work in the same way for the ultrasonic distance sensors on the back of the robot. These sensors are named distance3 and distance4 and are defined in the algorithm.

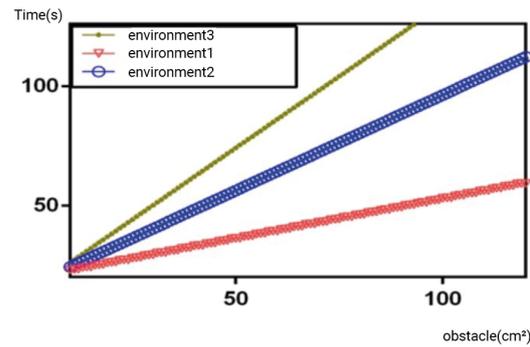


Figure 6. Increase in Target Finding Time by Obstacle Density.

Algorithm 1 Robot distance algorithm

Procedure

Require: $speedOne \geq 0$
Require: $speedTwo \geq 0$
Ensure: $distanceOne \neq distanceTwo$
 $speedOne \leftarrow 80$
 $speedTwo \leftarrow 80$
if $distanceOne \leq 40$ **or** $distanceTwo \leq 40$ **then**
 $m1Speed \leftarrow 0$
 $m2Speed \leftarrow 0$
else if $speedOne \geq 0$ **then**
 $randomNumberOne \leftarrow Random(-400, 0)$
else
 $randomNumberOne \leftarrow Random(400, 0)$
end if
if $speedTwo \geq 0$ **then**
 $randomNumberTwo \leftarrow Random(-400, 0)$
else
 $randomNumberTwo \leftarrow Random(400, 0)$
 $m1Speed \leftarrow randomNumberOne$
 $m2Speed \leftarrow randomNumberTwo$
end if
if $distanceThree \leq 40$ **or** $distanceFour \leq 40$ **then**
 $m1Speed \leftarrow 0$
 $m2Speed \leftarrow 0$
else if $speedOne \geq 0$ **then**
 $randomNumberOne \leftarrow Random(-400, 0)$
else
 $randomNumberOne \leftarrow Random(400, 0)$
end if
if $speedTwo \geq 0$ **then**
 $randomNumberTwo \leftarrow Random(-400, 0)$
else
 $randomNumberTwo \leftarrow Random(400, 0)$
 $m1Speed \leftarrow randomNumberOne$
 $m2Speed \leftarrow randomNumberTwo$
end if
End Procedure

If the algorithm is run step by step for the sample data, the initial distance data for the two sensors in the front are distance = 60 and distance2 = 60. When the algorithm was run step by step, it worked as follows:

- In the first iteration the following values are assigned; distance and distance2 60 and randNumber = −35 randNumber2 = 20. M1Speed and M2Speed values will change according to these values.
- In the second iteration the following values are measured after the robot’s movement; distance = 45, distance2 = 70 and randNumber = −10 randNumber2 = 40. M1Speed and M2Speed values will change according to these values.
- In the third iteration, the following values are measured; distance = 35, distance2 = 55. In this case, the first condition of the algorithm is met, and the motor speed becomes 0.
- In the fourth iteration, motor speeds are assigned as 50, and distance values are read as 65 and 75.
- In the fifth iteration, randNumbers are set as follows: randNumber = −5 randNumber2 = 55. Accordingly, M1Speed and M2Speed become 60 and 130.
- In the sixth iteration, distance = 5, distance2 =15. In this case, the motor speed reset.
- These processes continue until the robot scans the entire area.

2.2. BIMRCA

BIMRCA was developed in this part of the study. Path planning methods have been developed for 3D and 2D environments using BA, which has never been used for multi-robot coordination before. Considering the promising results of BA studies, the researchers decided to develop BIMRCA. The projected algorithm uses more than one robot in the system concerning their location, effectively determining survivors’ location. The data obtained from the infrared, distance and temperature sensors are used and processed in the algorithm to decide the robot’s behavior. As a result, the search is carried out faster. The setting is mapped while the robot is in motion. This modeling can be examined in a grid structure, as seen in Table 1. The representation in Table 1 is also called the proximity grid. Suppose that R steps will be taken from the distance grid information in Table 1. Then;

$$Nx = Xmax / R \tag{3}$$

is calculated by the number of partitions for each row. Similarly, the number of partitions in each column can be represented by;

$$Ny = Ymax / R \tag{4}$$

Here y represents the column. Each grid has its own coordinate and is represented by an Id.

The coordinates around the robot also change as a result of its movement. For example, the new proximity grid formed after the robot reaches a new position by moving one step on the x-axis and one step on the y-axis is shown in Figure 7.

Bearing all these in mind, a unique algorithm inspired by bats has been developed to coordinate multiple robots.

Table 1. Change of Coordinates around the Robot as a Result of Its.

(x,y)	(x,y+1)	(x, y+2)
(x+1,y)	(x+1,y+1)	(x+1,y+2)
(x+2,y)	(x+2,y+1)	(x+2,y+2)

The implementation of BIMRCA is shown in Algorithm 2. The details and steps can be explained as follows. RobotSensorDegerMatrix is a matrix containing the information about the temperature of the object the robot encounters, the ambient temperature, and the distance of the robot from the closest object at its right, left, bottom, and top, respectively. Wavelength refers to the distance between successive crests of a wave. Wavelength is obtained by dividing robot speed by frequency in this algorithm. Frequency refers to the robot’s motion to the target. The obstacle the robot encounters or the distance to the target is the loudness. Pulse is a positive value that increases as the robot approaches the

target. First, the variables are set, assigned to the variables in the algorithm and kept in the matrix. Then, the wavelength is set as $(\text{randNumber} + \text{randNumber2})/2$. The robot's upper-left distance is set as the frequency, and the sum of the robot's bottom-left, upper-right, and bottom-right values is set as the default loudness. If the ambient temperature is between 27 and 340 °C, the pulse value increases by 0.1, and the loudness value decreases by 1 due to the favorable conditions.

Algorithm 2 Bat-Inspired Multi-Robot Coordination Algorithm

Input: *robotSensorValueMatrix*, *waveLength*, *frequency*, *loudness*, *pulse*

Procedure:

```

envHeat ← robotSensorValueMatrix[2][1]
rightUp ← robotSensorValueMatrix[2][2]
leftUp ← robotSensorValueMatrix[2][3]
rightDown ← robotSensorValueMatrix[2][4]
leftDown ← robotSensorValueMatrix[2][5]
waveLength ← (randNumberOne + randNumberTwo)/2
loudness ← leftUp
frequency ← rightUp + leftUp + rightDown + leftDown
while Counter ≤ 5000 do
  if envHeat ≥ 27 and envHeat ≤ 34 then
    bestLocalOne ← randNumberOne
    bestLocalTwo ← randNumberTwo
    pulse ← pulse + 0.1
    loudness ← loudness − 1
  end if
  if envHeat ≥ 35 then
    bestLocalOne ← randNumberOne
    bestLocalTwo ← randNumberTwo
    m1Speed ← 0
    m2Speed ← 0
    pulse ← 1
    loudness ← 0
  end if
  if rightUp ≤ x2Best or leftUp ≤ y2Best then
    pulse ← pulse + 0.1
    loudness ← loudness − 1
    x2Best ← rightUp
    y2Best ← leftUp
  end if
  if rightDown ≤ x2Best or leftDown ≤ y2Best then
    pulse ← pulse + 0.1
    loudness ← loudness − 1
    x2Best ← rightDown
    y2Best ← leftDown
  end if
  if rightUp ≥ x2Best or leftUp ≥ y2Best then
    pulse ← pulse − 0.1
    loudness ← loudness + 1
  end if
  if rightDown ≥ x2Best or leftDown ≥ y2Best then
    pulse ← pulse − 0.1
    loudness ← loudness − 1
  end if
  return rightUp, rightDown, leftUp, leftDown
end while
End Procedure

```

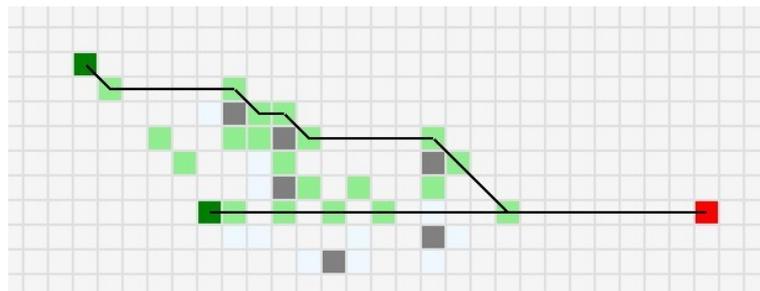


Figure 7. Modeling of Robot Environment in Grid Structure.

The pulse value is positive because the robot is close to finding the object it is looking for. The pulse takes a value between 0–1 here. In cases where the ambient temperature is 340 °C or higher, the engine stops immediately, the loudness is set as 0, and the pulse value as 1. It can be interpreted as the robot finding the object it is looking for, stopping in this position, and increasing the pulse value to the highest value, 1. Finally, thanks to the sensors on the robot, the upper-left, bottom-left, upper-right, and bottom-right values collected by the robot are compared with each other, and the pulse and loudness values are adjusted. The robot can sense whether it is approaching the target or not with this setting.

The table below shows the variables’ changes during the algorithm’s testing.

According to Table 2, two random numbers were generated in each iteration and assigned to the randNumber and randNumber2 variables. The average of these variables was set as the wavelength. The robot approached the target step by step and stopped its operation when the pulse value became 1 in the 7th iteration when it reached the target.

Table 2. Execution of Algorithm 2 with Test Data..

	S1U	S1A	SgU	SgA	W.length	Fre.	Loudness	Tempature	Pulse
1.Iteration	30	25	28	22	45	30	75	28	0.5
2.Iteration	32	22	25	18	40	32	65	27	0.6
3.Iteration	24	17	25	13	37	24	55	26	0.7
4.Iteration	36	15	28	12	38	36	55	26	0.7
5.Iteration	15	9	15	11	60	15	35	28	0.8
6.Iteration	20	13	12	5	50	20	30	30	0.9
7.Iteration	22	6	8	3	45	22	0	32	1

3. Findings

Virtual settings were created to test the algorithm and compare it with the others. Various environmental conditions and obstacles (roughness, heat source) were tested with different scenarios and forced the algorithm within the simulation settings. This study will interpret the findings obtained in two virtual and one concrete simulation setting.

Simulation 1: A virtual setting was created. Three virtual robots were used, and the training ground was a flat surface. Wall-like structures were added to the setting to limit the robots’ motion range. The movement of the robots was monitored. Robots have started their movements using sensors to scan the entire area as quickly as possible. Each robot transferred the information of the scanned areas to each other every two seconds so that they did not repeat the search on scanned areas. As seen in Figure 8, the position of Robot 1 will give the optimum result if it moves straight and turns to the right. At this point, Robot 1, which was advancing straight and could not make any progress due to the obstacle (wall) in front of it, randomly determined a new direction. The algorithm said to turn to the upper left, and the robot has decided to move in that direction. However, the controls conducted during the robot’s movement detected an obstacle (wall) at the top left, and the robot had to be redirected. As a result of the determined redirection, the robot decided to move to the right and continued its move.

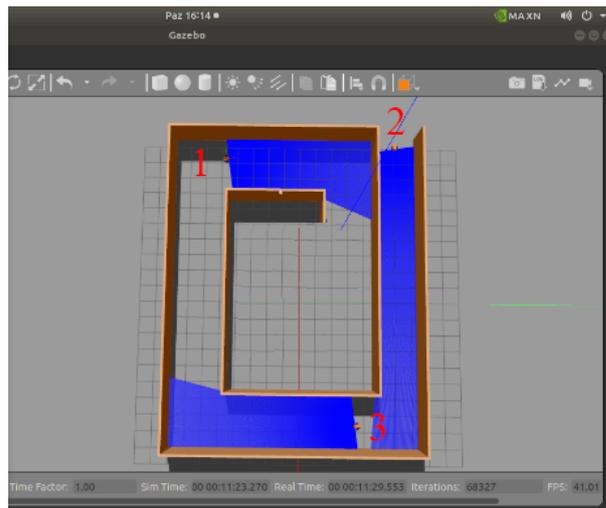


Figure 8. Environment and Motion Images for Simulation 1.

Simulation 2: After getting the optimum result from the first simulation, a new simulation setting was created, adding a heat source and a target to be recovered to the setting. Regarding the setting shown in Figure 8, robot1 moved to the middle zone. The obstacle detection function started to work slowly as the data from the temperature sensor increased above 34 °C as it approached the heat source it faced. Then the robot reduced its movement and determined a new direction. In this way, the entire area that could be explored by staying away from the heat source was scanned. When it approached the target shown as a black dot, it informed the others that there was an entity to be rescued here using temperature sensors. Robot2 and robot3 used the other areas and came near the overlapping areas. Regarding the information received from robot1, they decided that they had reached the target and ended their movements. The scanned area is shown in Figure 9. The area scanned by Robot 1 is shown orange, the area scanned by Robot2 with green, and the area scanned by Robot 3 with blue by the simulation.

The numbers 1, 2 and 3 appearing in Figures 8 and 9 represent the robots in different positions. In addition, they collect map information obtained by simultaneous scanning of robots at locations. Then, this collected data is combined to form a single map.

Tests were conducted with different algorithms in similar simulation settings to validate the collaborative work of the robots in finding targets. Error rates of the tests from collected accuracy, sharpness, sensitivity, and F criterion data are shown in the results section. The robots and the algorithm were tested in a concrete setting after completing the observations in the simulated setting. The setting used to obtain the data has been designed to liken actual structures to a certain extent by using brick, concrete and wooden materials. The robot navigation area shown in Figure 10 consisted of 8 units. Seven of these units were designed to be large enough for the robot to enter, and one unit had a curved obstacle surface. The units were numbered as shown in the figure. Apart from these, there were also obstacles interspersed among them. The robot can enter the sheltered area in two moves using two front distance sensors for the nook entrance assembly. If there is no departure zone behind the sheltered area, scanning continues in this area.

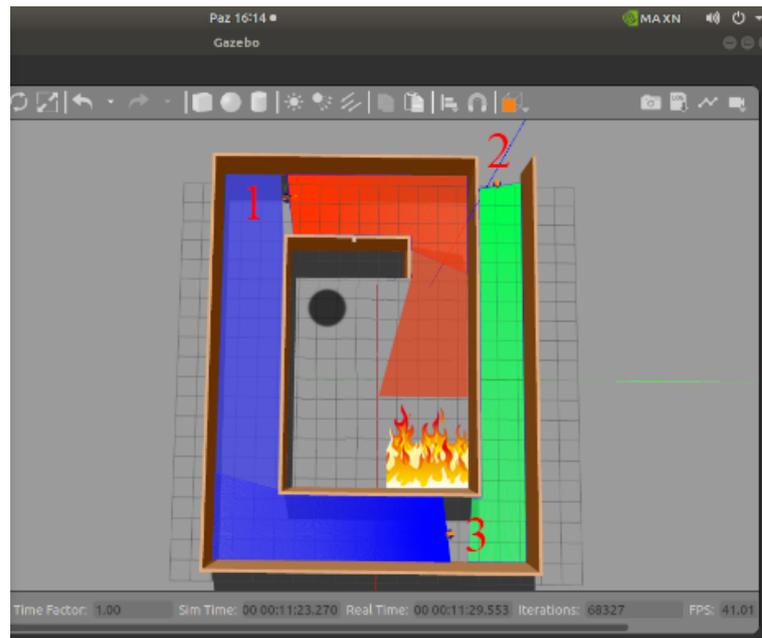


Figure 9. Environment and Motion Images for Simulation 2.



Figure 10. The Training Platform in detail.

Since all hardware is made of steel, it must be kept cold. Therefore, the temperature rise above a certain level poses a problem for the healthy operation of the algorithm. Figure 11 shows the delay in the obstacle detection function after detecting the heat. Delays of up to 34 °C are acceptable, depending on the function. The proposed algorithm reorientated the robot since the engine stops above this temperature.

The numbers from 1 to 8 indicated in Figure 10 consist of the real-life adaptation of the houses in the simulation environment.

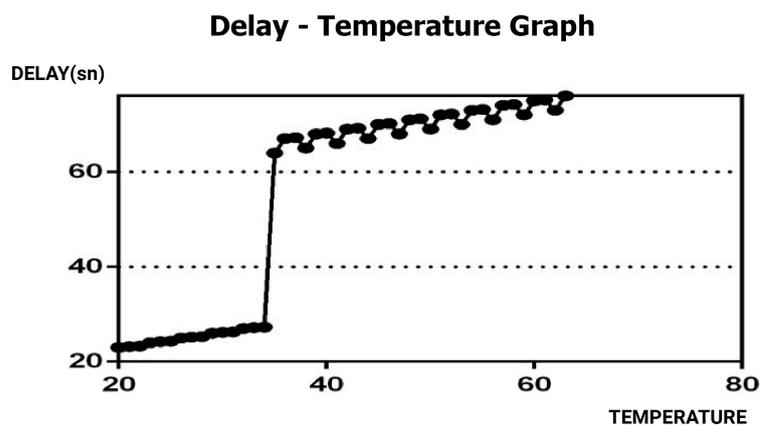


Figure 11. Variation of the delay time according to temperature.

4. Results

In this study, BIMRCA was suggested as an algorithm for casualty detection in earthquake collapses. An intuitive coordination system was developed using the robot's sensor values in the bat-inspired algorithm. The designed robots determine new directions by moving randomly according to the position of obstacles such as roughness, wall, and temperature. Tests were made for these scenarios in simulated settings and the created training platform. The robots were observed to determine a random direction according to the obstacles in their path. They also share information with other robots via wireless communication when they detect an injured using temperature sensors. Two algorithms have been developed for both the obstacle and the intuitive movement in this context. The main challenge during the experiment was the ultrasonic distance sensors' coverage area because the coverage of the sensors was kept narrow to prevent possible collisions with walls or other obstacles. This situation makes it challenging to map the environment. A new motion is determined in cases when the data from the sensors indicate a distance below 40 cm. Another challenge was the inability to create an educational model for direction determination with the old data. Therefore, a new direction was determined t times depending on the obstacle density for each encountered obstacle. For the algorithm to give the best performance, the frequency should be less than 35, the loudness greater than 30, the temperature should be 270 °C or above, and the wavelength should be greater than 40. Because, regarding the algorithm results, progress was observed in iterations with these values. In future studies, the direction can be determined faster by introducing the obstacle in the environment through the education process.

In addition, the image control can be made with a camera and a mobile device when the robot stops, and the algorithm working principle can be altered with user intervention. Depending on the snapshots taken from the camera on the robot, the variables can be changed so that the algorithm can enter a different state and work differently. In addition to these, maps with different characteristics can be analyzed, the optimum number of robots required in disaster areas can be determined, and the resources can be allocated and used more efficiently. Bat-Algorithm (BA), Simultaneous Localization and Mapping (SLAM), Random Particle Swarm Optimization (RPSO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) were tested in simulated and concrete settings. The data were created by taking the average of 50 simulations, one concrete setting, and the written scenarios of simulated tests. Test results are shown in Figure 12.

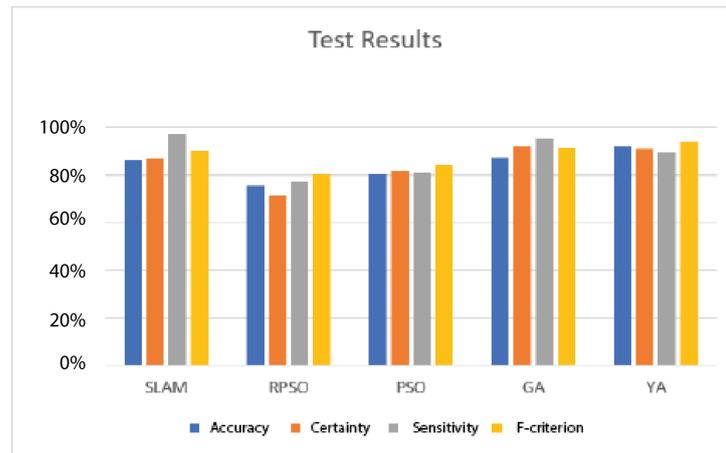


Figure 12. Algorithm Test Results.

The Accuracy, Sharpness, Sensitivity and F-Measure data in Table 3 are taken from the complexity matrix, a table often used to describe the performance of a classification model.

Table 3. General Form Of Complexity Matrix.

Complexity Matrix		Estimated Class	
		Positive	Negative
True Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Equations of the concepts used in performance evaluation are below.

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{5}$$

$$\text{Sharpness} = \frac{TP}{TP + FP} \tag{6}$$

$$\text{Accuracy} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{F-measure} = \frac{2 * \text{Sensitivity} * \text{Sharpness}}{\text{Sensitivity} + \text{Sharpness}} \tag{8}$$

As a result of the simulation, the most successful algorithm for positioning the robots was BA. The robots have been observed to transfer relative location information without any problems using BA. The simulations of the robots’ location information are given in Figures 13 and 14. The blue areas show the areas within the robot’s line of sight.

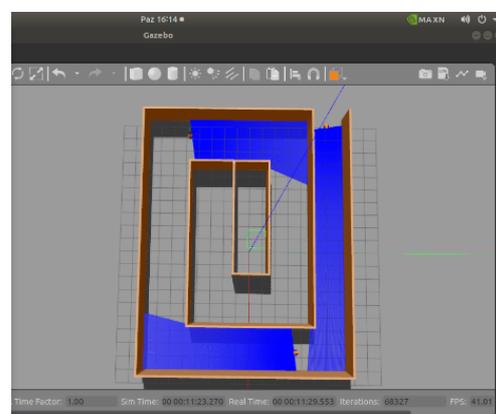


Figure 13. Top view.

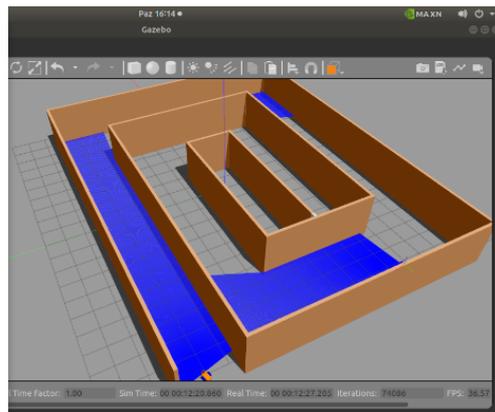


Figure 14. Side view.

The simulation showed that the SLAM algorithm was the most successful in 2D mapping. The communication between the robots is continuous in this method. Uninterrupted communication can also be achieved by using a large number of robots.

BA tests compared the performance and processing load requirements of SLAM algorithms. The effects of measuring noise and motion noise on BA have been demonstrated, showing that the increase in system performance shortens the time spent in BA. Regarding the experiments conducted in the concrete setting, robots created their zones and mapped them without entering each other's area, shared maps, and merged them on a common map. Robots transfer the information of the scanned area to each other through various sensors and continue their movements with the information of these scanned areas. Therefore, a robot knows whether an area has been scanned or not and proceeds accordingly. The maps created by the robots are shown in Figure 15. The red and black areas in the images show the areas detected by the sensors as obstacles.

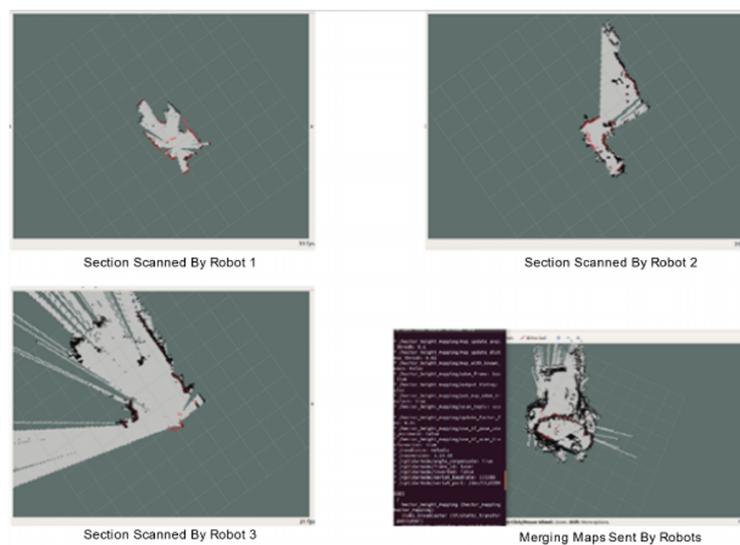


Figure 15. Maps Created as a Result of Simulation.

After the maps were created, they were merged by the Radon transform algorithm. Radon transformation is used to extract geometric information from a map according to offsets and rotation [39]. Sinogram-based map aggregation was performed using Radon transform. Sinogram-based map assembly consists of two consecutive parts, as shown in Figure 16.

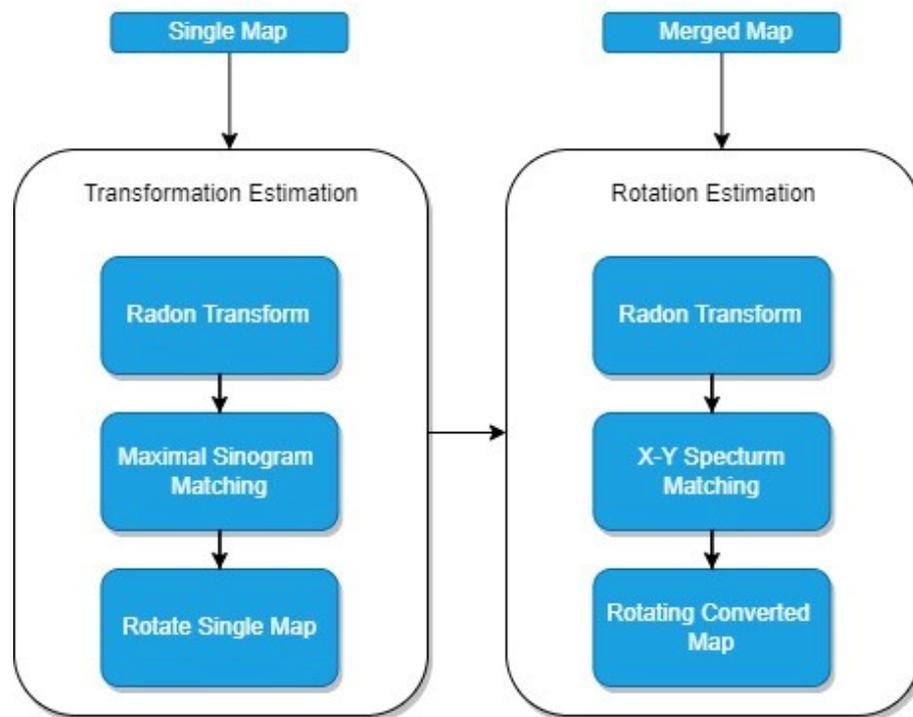


Figure 16. Structure of sinogram-based map aggregation.

The rotation angle and the offset amounts are estimated by maximum sinogram matching and partial sinogram matching in the sinogram-based map merging. First, the map rotation between the two maps is estimated: Delta Theta and then Delta x and Delta y offsets between them are estimated. The Radon transform of the Dirac delta function is often referred to as a sinogram because the graph of a sine wave supports its distribution. Let $f(x, y)$ be a continuous function in R^2 disappearing outside a large disk. $f(x, y)$ is a continuous function zeroing on circular surface R^2 . A straight line in this plane, denoted by L , is parameterized as follows:

$$(x(t);y(t)) = ((t \sin a + s \cos a); (-t \cos a + s \sin a)) \tag{9}$$

where t is a parameter for the parametric form of L , s is L 's distance from the origin, and Alfa is the normal vector's angle to the x -axis. In Radon transform, RTf is a function on the space of straight lines of L in R^2 , defined by a line integral along each such line

$$\begin{aligned} RT_f &= \int_{-\infty}^{+\infty} f(x(t),y(t))dt \\ &= \int_{-\infty}^{+\infty} f(t \sin(\alpha) + s \cos(\alpha), (-t \cos(\alpha) + s \sin(\alpha)))dt \end{aligned} \tag{10}$$

Sinograms of the scanned areas are shown in Figure 17. The structure of a sensor on the robot is shown visually and graphically in Figures 18 and 19.

They were tested with three robots in a virtual environment. Robots can create a new map by sharing the RP-lidar information they obtained by communicating with each other. Using the BA, they send instant location information to other robots every 2 s, enabling them to share location information simultaneously. In this way, a robot can perform mapping more quickly by avoiding the areas where the other robot has already scanned. The error function in Equation (11) is used to calculate the error between the estimated node positions and the actual node positions while mapping.

$$E = 1N \sum \sqrt{(x_i - X_i) + (y_i - Y_i)^2} \tag{11}$$

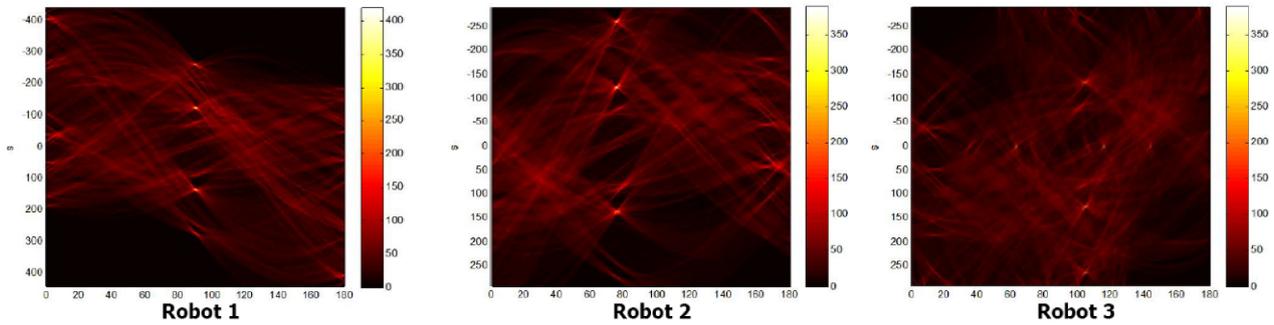


Figure 17. Sinograms of the Scanned Areas.

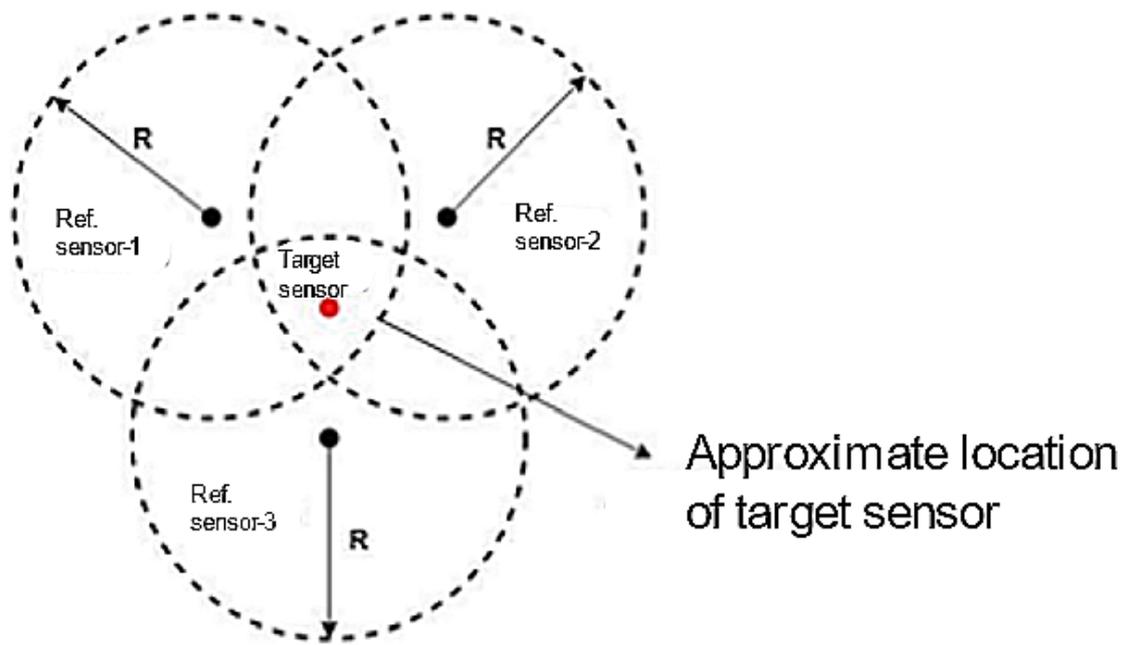


Figure 18. Structure of a Sensor on the Robot Graphically.

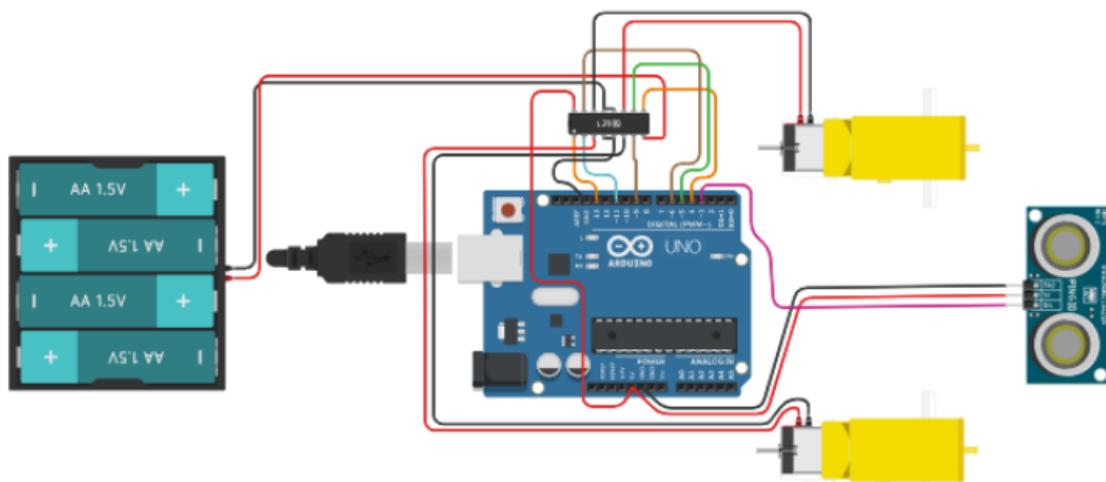


Figure 19. Structure of a Sensor on the Robot Visually.

Here (N) represents the number of target nodes, (Y_i) the estimated target node coordinates, and (x_i, y_i) the target node's coordinates. The coordinate information obtained from the simulated setting is used in this formula, and different algorithms' error rates have been calculated. The positions of 20 predicted and actual nodes in simulation1 and the neighbouring connection obtained by their combination are shown in Figure 20.

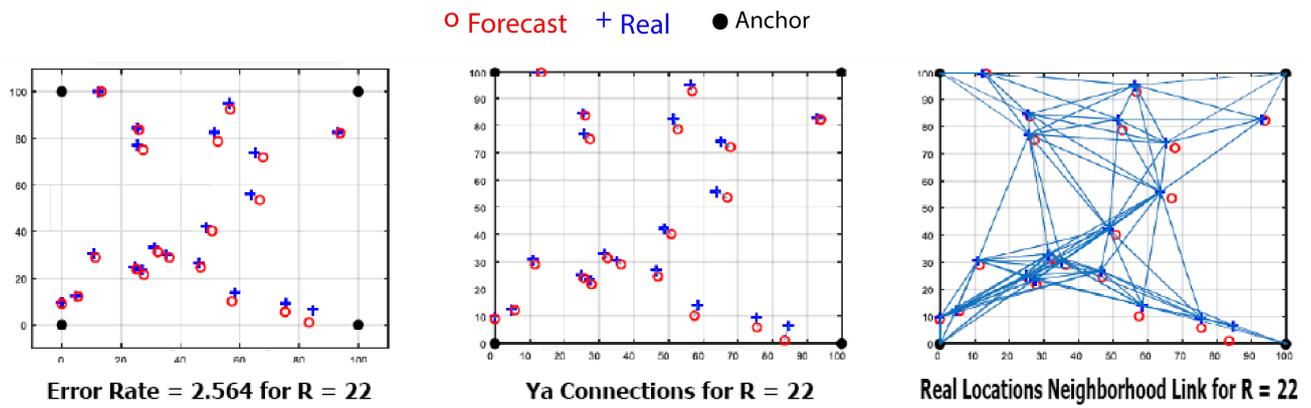


Figure 20. Obtaining Error Rate for Bat Algorithm.

The iteration results used for different simulation settings are shown in Figure 21, and the average error rates for 50 simulations are shown in Figure 22.

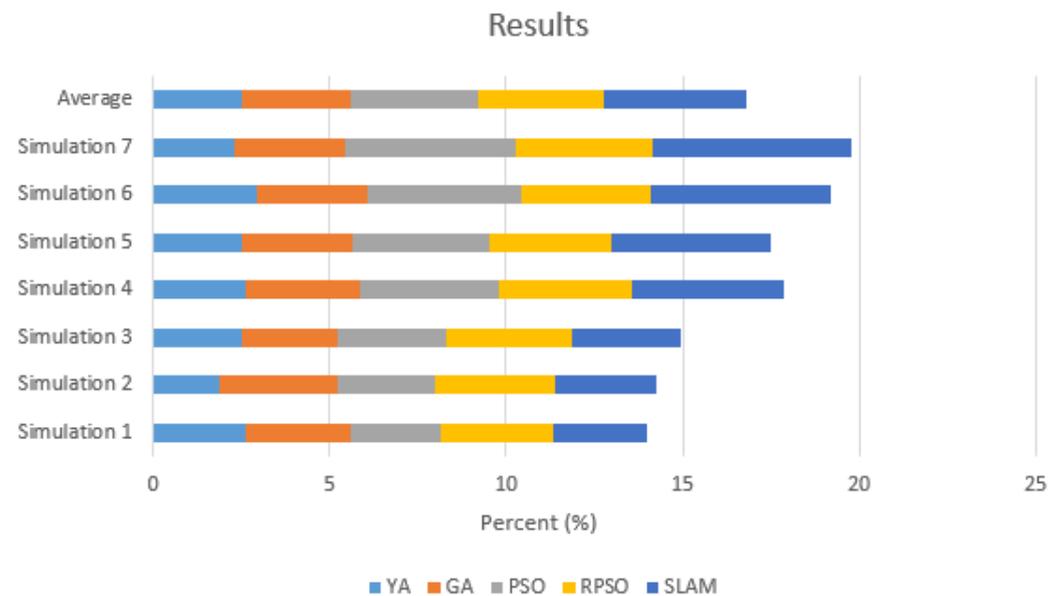


Figure 21. Error Rates for Different Simulation Environments.

The scope of the scanning process can be increased by using more advanced sensors in the future to improve the effectiveness of the study. In addition, an educational model can be developed using the old data, making direction determination automatic. In the long term, autonomous flying robots can be integrated into the team in areas where robots cannot move (For example, in an environment with flooding or high water level). Joint search and rescue activities can be carried out with the Disaster and Emergency Management Presidency (AFAD), established in our country in 2009, to implement the system described in the study.



Figure 22. Average error rates for 50 simulations.

5. Conclusions

Multi-robot coordination algorithm is proposed for casualty detection in earthquake collapses. An intuitive coordination system was developed by adapting the robot's sensor inputs to the bat-inspired algorithm. The robots determine new directions according to the position of obstacles such as roughness, wall, and temperature by moving randomly. These scenarios were tested both in simulated and concrete settings. The robots were observed to determine their direction randomly regarding the obstacles in their path.

According to the simulations, the BA has the lowest error rate, with an average of 1.5%. After being randomly distributed into space, bats' fitness values affect the direction of their next movement and the subsequent movement of the whole population. This is why BA's success rate is higher than that of other algorithms. The robots share information with other robots utilizing wireless communication when temperature sensors detect the injured. Two algorithms have been developed for the obstacle and the intuitive movement in this context. The main challenge during the experiment was the ultrasonic distance sensors' coverage because it was kept narrow to prevent possible collisions with walls or other obstacles.

Author Contributions: Conceptualization, T.Y. and Ş.F.Ç.; writing—review and editing, Ş.F.Ç.; project administration, T.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Disaster and Emergency Management Presidency (AFAD) [UDAP-C-18-05] and Scientific Research Coordinatorship at Süleyman Demirel University [FDK-2020-7439].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We want to thank the Disaster and Emergency Management Presidency (AFAD) for aiding in carrying out this study with financial support under the project code of UDAP-Ç-18-05. We also would like to thank Scientific Research Coordinatorship at Süleyman Demirel University for supporting our work morally and financially under the project code FDK-2020-7439.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berawi, M.A.; Siahaan, S.A.O.; Miraj, P.; Leviakangas, P. Determining the prioritized victim of earthquake disaster using fuzzy logic and decision tree approach. *Evergreen* **2020**, *7*, 246–252. [[CrossRef](#)]
2. Williams, A.; Sebastian, B.; Ben-Tzvi, P. Review and analysis of search, extraction, evacuation, and medical field treatment robots. *J. Intell. Robot. Syst.* **2019**, *96*, 401–418. [[CrossRef](#)]
3. Ranjan, A.; Sahu, H.B.; Misra, P. Wireless robotics networks for search and rescue in underground mines: Taxonomy and open issues. In *Exploring Critical Approaches of Evolutionary Computation*; IGI Global: Hershey, PA, USA, 2019; pp. 286–309.
4. Alam, M.N.; Saiam, M.; Al Mamun, A.; Rahman, M.M.; Hany, U. A Prototype of Multi Functional Rescue Robot Using Wireless Communication. In Proceedings of the 2021 5th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT), Mirpur, Dhaka, 18–20 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
5. Zhou, J.; Li, L.; Zhang, X. Wireless Control System Design for Mine Rescue Robot. *J. Phys. Conf. Ser.* **2019**, *1237*, 042052. [[CrossRef](#)]
6. Sharmin, S.; Salim, S.I.; Sanim, K.R.I. A Low-Cost Urban Search and Rescue Robot for Developing Countries. In Proceedings of the 2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON), Dhaka, Bangladesh, 29 November–1 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 60–64.
7. Plaza, P.; Sancristobal, E.; Carro, G.; Blazquez, M.; García-Loro, F.; Martin, S.; Perez, C.; Castro, M. Arduino as an educational tool to introduce robotics. In Proceedings of the 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), Wollongong, NSW, Australia, 4–7 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.
8. Margolis, M. *Make an Arduino-Controlled Robot*; O'Reilly Media, Inc.: Newton, MA, USA, 2012.
9. Babu, S.R.; Kumar, K.S.; Narayanan, G.S.; Raja, R.; Anbazhagan, K.; Sathab, H.Y. Design and conception of Arduino based quadruped robot. *Mater. Today Proc.* **2021**, in press.
10. Nguyen, T.T.; Pan, J.S.; Dao, T.K. A compact bat algorithm for unequal clustering in wireless sensor networks. *Appl. Sci.* **2019**, *9*, 1973. [[CrossRef](#)]
11. Nosirov, K.; Begmatov, S.; Arabboev, M. Analog Sensing and Leap Motion Integrated Remote Controller for Search and Rescue Robot System. In Proceedings of the 2020 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 4–6 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5.
12. Nosirov, K.K.; Shakhobiddinov, A.S.; Arabboev, M.; Begmatov, S.; Togaev, O.T. Specially Designed Multi-Functional Search And Rescue Robot. *Bull. Tuit Manag. Commun. Technol.* **2020**, *2*, 1–5.
13. Verma, J.K.; Ranga, V. Multi-robot coordination analysis, taxonomy, challenges and future scope. *J. Intell. Robot. Syst.* **2021**, *102*, 10. [[CrossRef](#)]
14. Jin, L.; Li, S.; La H.M.; Zhang, X.; Hu, B. Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach. *Automatica* **2019**, *100*, 75–81. [[CrossRef](#)]
15. Talebpour, Z.; Martinoli, A. Multi-robot coordination in dynamic environments shared with humans. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 4593–4600.
16. Batinović, A.; Oršulić, J.; Petrović, T.; Bogdan, S. Decentralized strategy for cooperative multi-robot exploration and mapping. *IFAC-PapersOnLine* **2020**, *53*, 9682–9687. [[CrossRef](#)]
17. Huang, Y.C.; Lin, H.Y. Development and Implementation of a Multi-Robot System for Collaborative Exploration and Complete Coverage. In Proceedings of the 2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Municipality of Las Palmas, Spain, 26–29 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 472–479.
18. Kandhasamy, S.; Kuppasamy, V.B.; Krishnan, S. Scalable decentralized multi-robot trajectory optimization in continuous-time. *IEEE Access* **2020**, *8*, 173308–173322. [[CrossRef](#)]
19. Tchuiev, V.; Indelman, V. Distributed consistent multi-robot semantic localization and mapping. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4649–4656. [[CrossRef](#)]
20. Queraltá, J.P.; Qingqing, L.; Schiano, F.; Westerlund, T. VIO-UWB-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems. *arXiv* **2020**, arXiv:2011.00830.
21. Alam, S.S.; Ahmed, T.; Islam, M.S.; Chowdhury, M.M.F. A Smart Approach for Human Rescue and Environment Monitoring Autonomous Robot. *Int. J. Mech. Eng. Robot. Res.* **2021**, *10*, 209–215. [[CrossRef](#)]
22. Bai, L.; Guan, J.; Chen, X.; Hou, J.; Duan, W. An optional passive/active transformable wheel-legged mobility concept for search and rescue robots. *Robot. Auton. Syst.* **2018**, *107*, 145–155. [[CrossRef](#)]
23. Joseph, J.; Berchmans, N.; Varghese, M.R.; Krishnan, M.; Arya, B.S.; Antony, M. Arduino based automatic human seeker robot for disaster management. In Proceedings of the 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT), Kannur, India, 5–6 July 2019; IEEE: Piscataway, NJ, USA, 2019; Volume 1, pp. 770–773.
24. Mae, N.; Yamaoka, K.; Mitsui, Y.; Matsumoto, M.; Makino, S.; Kitamura, D.; Ono, N.; Yamada, T.; Saruwatari, H. Ego noise reduction and sound localization adapted to human ears using hose-shaped rescue robot. In Proceedings of the International Workshop on Nonlinear Circuits, Communications and Signal Processing, Honolulu, HI, USA, 4–7 March 2018; pp. 371–374.
25. Kiran, V.V.; Santhanalakshmi, S. Raspberry Pi based Remote Controlled Car using Smartphone Accelerometer. In Proceedings of the 2019 International Conference on Communication and Electronics Systems (ICES), Coimbatore, India, 17–19 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1536–1542.

26. Buzduga, C.; Graur, A.; Ciufudean, C.; Vlad, V. System for the detection earthquake victims—Construction and principle of operation. In *New Developments in Circuits, Systems, Signal Processing, Communications and Computers*; Springer: Berlin/Heidelberg, Germany; pp. 110–115.
27. Akkhar, S.A.; Sharma, P.; Jamali, M.F. *Design and Development of a Search and Rescue Robot*; DSpace Repository: Atlanta, GA, USA, 2019.
28. Aouf, A.; Boussaid, L.; Sakly, A. TLBO-based adaptive neurofuzzy controller for mobile robot navigation in a strange environment. *Comput. Intell. Neurosci.* **2018**, *2018*, 3145436. [[CrossRef](#)] [[PubMed](#)]
29. Low, E.S.; Ong, P.; Cheah, K.C. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robot. Auton. Syst.* **2019**, *115*, 143–161. [[CrossRef](#)]
30. Ajeil, F.H.; Ibraheem, I.K.; Azar, A.T.; Humaidi, A.J. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. *Sensors* **2020**, *20*, 1880. [[CrossRef](#)]
31. Koreitem, K.; Shkurti, F.; Manderson, T.; Chang, W.D.; Higuera, J.C.G.; Dudek, G. One-shot informed robotic visual search in the wild. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 January 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 5800–5807.
32. Muthukumar, S.; Sivaramkrishnan, R. Optimal path planning for an autonomous mobile robot using dragonfly algorithm. *Int. J. Simul. Model.* **2019**, *18*, 397–407. [[CrossRef](#)]
33. Adam, Y.M.; Sariff, N.B.; Algeelani, N.A. E-puck Mobile Robot Obstacles Avoidance Controller Using the Fuzzy Logic Approach. In Proceedings of the 2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Virtual, NJ, USA, 15–16 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 107–112.
34. Kumar, V.; Kumar, D. A systematic review on firefly algorithm: Past, present, and future. *Arch. Comput. Methods Eng.* **2021**, *28*, 3269–3291. [[CrossRef](#)]
35. Luque, A.; Carrasco, A.; Martín, A.; de Las Heras, A. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognit.* **2019**, *91*, 216–231. [[CrossRef](#)]
36. Gürgüze, G.; Türkoğlu, İ. Robot Sistemlerinde Kullanılan Algoritmalar. *Türk Doğa Fen Derg.* **2019**, *8*, 17–31.
37. Doğru, Sami, A.; Eren, T. Kablosuz Sensör Ağlarında Konum Belirlemede Parçacık Sürü Optimizasyonu, Yarasa Algoritması, Diferansiyel Gelişim Algoritması ve Ateşböceği Algoritması Yöntemlerinin Karşılaştırılması. *Int. J. Eng. Res. Dev.* **2020**, *12*, 52–64.
38. Lee, H.; Lee, S. Grid map merging with insufficient overlapping areas for efficient multi-robot systems with unknown initial correspondences. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1427–1432.
39. Ramm, A.G.; Katsevich, A.I. *The Radon Transform and Local Tomography*; CRC Press: Boca Raton, FL, USA, 2020
40. Huang, S. A review of optimisation strategies used in simultaneous localisation and mapping. *J. Control. Decis.* **2019**, *6*, 61–74. [[CrossRef](#)]
41. Jain, N.K.; Nangia, U.; Jain, J. A review of particle swarm optimization. *J. Inst. Eng. India Ser.* **2018**, *99*, 407–411. [[CrossRef](#)]
42. Mirjalili, S. Genetic algorithm. In *Evolutionary Algorithms and Neural Networks*; Springer: Cham, Switzerland, 2019; pp. 43–55.
43. Yang, X.S.; Slowik, A. Bat algorithm. In *Swarm Intelligence Algorithms*; CRC Press: Boca Raton, FL, USA, 2020; pp. 43–53.
44. Moshayedi, A.J.; Roy, A.S.; Sambo, S.K.; Zhong, Y.; Liao, L. Review on: The Service Robot Mathematical Model. *Eai Endorsed Trans. Robot.* **2022**, *1*, 1–19. [[CrossRef](#)]
45. Moshayedi, A.J.; Roy, A.S.; Liao, L. PID Tuning Method on AGV (automated guided vehicle) Industrial Robot. *J. Simul. Anal. Nov. Technol. Mech. Eng.* **2019**, *12*, 53–66.
46. Moshayedi, A.J.; Roy, A.S.; Kolahdooz, A.; Shuxin, Y. Deep Learning Application Pros and Cons Over Algorithm. *Eai Endorsed Trans. Robot.* **2022**, *1*, 1–13. [[CrossRef](#)]
47. Moshayedi, A.J.; Roy, A.S.; Liao, L.; Li, S. Raspberry Pi SCADA Zonal based System for Agricultural Plant Monitoring. In Proceedings of the 2019 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, 20–22 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 427–433.
48. Moshayedi, A.J.; Agda, A.M.; Arabzadeh, M. Designing and Implementation a Simple Algorithm Considering the Maximum Audio Frequency of Persian Vocabulary in Order to Robot Speech Control Based on Arduino. In *Fundamental Research in Electrical Engineering*; Springer: Singapore, 2019; pp. 331–346.
49. Fozuni, S.M.; Nikanjam, A.; Aliyari, S.M. Stability analysis of the particle dynamics in bat algorithm: Standard and modified versions. *Eng. Comput.* **2021**, *37*, 2865–2876. [[CrossRef](#)]
50. Belge, E.; Altan, A.; Hacıoğlu, R. Metaheuristic Optimization-Based Path Planning and Tracking of Quadcopter for Payload Hold-Release Mission. *Electronics* **2022**, *11*, 1208. [[CrossRef](#)]