Search and Rescue Operations with Mesh Networked Robots

Tristan Brodeur
Department of Mathematics
University of Nevada, Las Vegas (UNLV)
Las Vegas, USA
brodeurtristan@gmail.com

Paulo Regis
Department of Computer Science
University of Nevada, Reno (UNR)
Reno, USA
pregis@nevada.unr.edu

David Feil-Seifer
Faculty of Computer Science
University of Nevada, Reno (UNR)
Reno, USA
dave@cse.unr.edu

Shamik Sengupta
Faculty of Computer Science
University of Nevada, Reno (UNR)
Reno, USA
ssengupta@unr.edu

***Abstract*—Efficient path planning and communication of multi-robot systems in the case of a search and rescue operation is a critical issue facing robotics disaster relief efforts. Ensuring all the nodes of a specialized robotic search team are within range, while also covering as much area as possible to guarantee efficient response time, is the goal of this paper. We propose a specialized search-and-rescue model based on a mesh network topology of aerial and ground robots. The proposed model is based on several methods. First, each robot determines its position relative to other robots within the system, using RSSI. Packets are then communicated to other robots in the system detailing important information regarding robot system status, status of the mission, and identification number. The results demonstrate the ability to determine multi-robot navigation with RSSI, allowing low computation costs and increased search-and-rescue time efficiency.**

***Index Terms*—Wireless Mesh Networks (WMN); ZigBee; RSSI; Disaster Relief; Multi-robot systems;Rescue robots**

I. INTRODUCTION

According to the United Nations, 54% of the worlds population currently resides in urban areas. This number is expected to grow to 66% by the year 2050 [1]. With a majority of the population moving to dense, urban environments, major onset natural disasters such as tsunamis and earthquakes targeting such areas will result in a larger percentage of the global population being affected. These natural disasters damage large amounts of infrastructure, oftentimes causing residents to become trapped within the resulting debris. In these cases, a highly effective search and response team would be needed to allow swift discovery and assistance to those affected.

The proposed search and rescue model is a team of highly specialized unmanned aerial vehicles (UAV's) and unmanned ground vehicles (UGV's) that communicate in a mesh-network topology, and localize via the received signal strength indicator (RSSI) of adjacent nodes. A multi-robot search-and-rescue operation will allow the distribution of any necessary payload required for a rescue operation, thus resulting in more time alloted for search as a result of better energy efficiency.

978-1-5386-7693-6/18/$31.00 ©2018 IEEE

Fig. 1: **UAV and UGVs.** An array of UAVs and UGVs allow an efficient response time in disaster relief efforts.

Wireless meshed robots benefit from the flexibility and self-organization of wireless mesh networks, allowing many advantages for search-and-rescue missions. One benefit is the removal of the dependency on a single control network point. Instead, wireless meshed robots allow relaying messages between different nodes of the system, removing the possibility for failures of entire systems as a result of a single point of failure. Wireless meshed robots also possess the capability to flexibly communicate with each other by dynamically discovering other surrounding nodes without any fixed architecture [2].

Large-area search-and-rescue techniques are of primary concern in disaster relief. One search-and-rescue method typically employed by fire departments is a *Left-Hand* searching method. This method typically deems one firefighter as a coordinator, who will then determine the path by following along a wall on the left side of the building. Several other firefighters then interlock via an elastic band to expand the amount of area searched, while also ensuring the location and safety of other firefighters within the building. This approach to search-and-rescue equates to a faster and safer response and rescue time.

We employ a similar *Left-Hand* search with an array of UAVs and UGVs. Each robot will ensure communication and positioning of other robots within the network via the RSSI of each robot's radio transmission device. Adjacent nodes' RSSI act as the band that determine future actions and ensure communicability between nodes.

Several techniques based on RSSI positioning have been developed. One such approach [5] discusses positioning based on a coupling of RSSI and a wireless sensor network (WSN), whereby nodes of the WSN are placed in a grid pattern and the robot coordinates itself via the four closest nodes' RSSI. However, due to the uninformed nature of disaster sites, we take a different approach that allows robots to navigate relative to each other rather than via pre-determined positioning of nodes.

The remainder of this paper is organized as follows. Section II will review prior work related to the topics of mesh-networked robots, multi-robot path planning, and path planning using RSSI. In Section III, the hardware platforms are discussed, along with software components utilized. Section IV presents models used for experimentation, including the proposed path planning method in detail. Section V will detail simulation and experimental results for evaluation of the proposed method. Finally, we conclude with our findings and future improvements in Section VI.

II. RELATED WORK

Our work builds on a rich history of past work in the fields of mesh-network communication architectures, ZigBee-based communications, and multi-robot system navigation. We utilize past work to both improve upon current standards and ensure appropriate decision making when choosing technologies to work with.

A. Mesh-Networked Robotics

Several multi-robot navigation approaches utilizing mesh networks have been explored [6], [8]. These methods utilize wireless mesh networks to allow multi-robot communication or localization. Cartano, in [8], discusses a shared database between multiple robots in a system. Each robot locally hosts a separate version of the database, allowing blocks of data to be private and public, with private blocks storing local data vital to a particular node in the system. Public blocks are then passed between robots that detail information vital to the group of robots.

In [6], static wireless access points organized in a mesh network topology allow a configuration of robots to determine their location based on the averaging of η access points and comparing it to a locally stored database. A *leader* robot then communicates its velocity command to the remaining *follower* robots, who proceed to imitate the received velocity commands in order to stay in line with the *leader* robot, reducing the computational load of path finding from several agents to one agent. In [10], a localization method is applied to mobile nodes based on a target node's RSSI. This method utilizes a floor-plan of the navigable area to determine positioning data, and statically placed nodes' RSSI to assist in positioning correction. Experiments are then run in simulation and in physical to determine average localization errors in order to determine the feasibility of their model. Both utilize static nodes to increase accuracy in localization estimates, which is not possible for our application due to the unknown nature of our navigable environment.

B. ZigBee Based Communication

Research into applications of ZigBee-enabled devices, especially in the field of robotics, has been extensively researched.Distance measurement techniques based on the RSSI of a ZigBee device are evaluated in [9]. One approach discussed in the paper proposes a Gauss model to remove small probability events in order to account for the instability of RSSI measurements. An approximate distance measurement based on the accumulated RSSI values can then be determined.

An array of devices using the ZigBee communication protocol is discussed in [12]. Several tests are conducted detailing information regarding message-latency and maximum data rate of the system. From the literature, the observed average time-of-flight is recorded to be 7 ms per hop. This data is taken into account in determining the efficient threshold value for our system.

When compared to other wireless standards such as WiFi or Bluetooth, the IEEE 802.15.4 standard (ZigBee standard) is the conclusive choice for our system. As discussed in [13], ZigBee demands lower power requirements compared to WiFi and Bluetooth, while also allowing greater range when compared to Bluetooth. In addition, ZigBee does not require scheduling special wake-up events in order to communicate and maintain synchronization. For these reasons, we have decided to utilize ZigBee enabled modules to transmit data across our devices.

C. Multi-Robot Navigation

In [3], several multi-robot navigation techniques are discussed. Of the methods discussed, two are of particular importance due to their applicability in our system architecture. Balch terms a *Leader-Referenced* approach as a multi-robot navigation system that determines each robots formation in reference to a leader robot. A *Neighbor-Referenced* approach is a multi-robot navigation technique that determines each robots action via the actions of its neighbor. We'll utilize both techniques within our system architecture. The *Leader-Referenced* approach will be utilized to ensure a constant velocity amongst each robot based on the velocities of the *Coordinator* robot. A *Neighbor-Referenced* approach will be utilized to ensure each robot takes appropriate actions based on their neighbor robot's RSSI value.

A multi-robot navigation system consisting of UAVs and UGVs is discussed in [4]. In this system, the authors test various policy-based navigation techniques on a set of pursuer robots attempting to capture a group of evading robots. A *Global-Max* policy is tested against a *Local-Max* policy, and is found to be more efficient in the metric of capture time. While a policy-based searching method is more time-efficient with

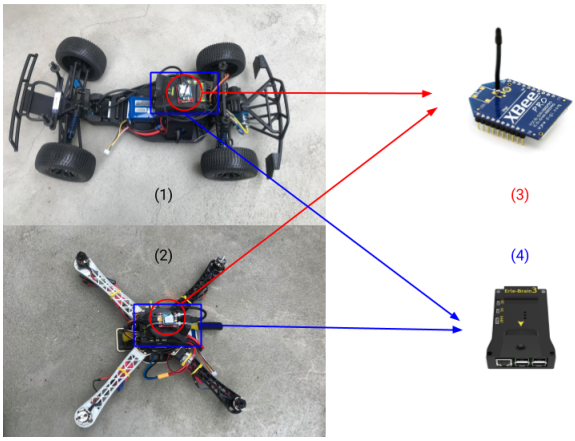

Fig. 2: **Robot Hardware**. (1) Unmanned ground vehicle (UGV) from Erle Robotics. (2) Unmanned aerial vehicle (UAV) from Erle Robotics. (3) XBee Pro. radio transceiver device compatible with the 802.15.4 standard (4) Erle Brain3, an embedded Linux computer from Erle Robotics

a smaller group of vehicles, our *Left-Hand* searching method allows a more time-efficient searching method with a large group of unmanned vehicles. A large system of vehicles basing their actions on a single parameter also lends itself to a more computationally efficient multi-robot navigation technique.

III. HARDWARE AND SOFTWARE ARCHITECTURE

To ensure a fast and efficient response time, multiple robotic platforms are used. The main design philosophy for the discussed model is one where each robot within the system is able to operate as its own unit, while also being able to act harmoniously as part of a larger group. Variation in traversal method is also taken into account, with the system being able to work both in air and on land.

The hardware platforms used are shown in Fig. 2. Vehicle (1) is an Erle Robotics UGV, whereas vehicle (2) is an Erle Robotics UAV. Both platforms are outfitted with an Erle Brain3 (4), an embedded Linux computer that incorporates sensor measurements from an on-board inertial measurement unit (IMU), a pressure sensor, and a temperature sensor. The software stack of the Erle Brain utilizes the Robot Operating System (ROS) [11] to communicate sensor data and motor commands. The software is implemented on ROS Indigo on top of an Ubuntu version 14.04 operating system.

The main communication and localization device is the XBee module(3). This module is a radio transceiver device compatible with the Zigbee protocol. XBee's are designed for high-throughput applications that require low latency. Each robot in the system is equipped with a XBee module. This module gives each robot the capability to communicate to other robots integral information regarding positioning data, system status, and missions status.

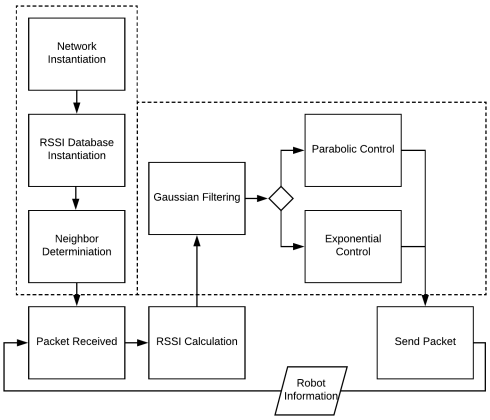

Fig. 3: **Algorithm Overview**

IV. PROPOSED METHOD

For a swarm of independently operated robotic systems to work efficiently in a search and rescue mission, each robot must be within communication range of other robots in the system, while also ensuring optimal coverage of an area. An ensured communication method between individual nodes in the system is also of paramount importance.

A mesh network topology is the primary communication method employed in our system. The non-hierarchical and self-healing design of mesh networks benefit multi-robot systems, as it reduces the probability of failures of the entirety of a system as a result of the failure of a single node within the system. Our *Rubber-Band* approach to communication and path-determination is based on the RSSI calculated from nodes organized in a mesh-network topology.

A. System Architecture

Due to the nature of a ZigBee mesh-network architecture, one node in the system is allocated as the coordinator node, whereby all of the initial system routing is done, while every other node is determined as a router. After initialization of all nodes in the system, the coordinator node acts in the same manner as a router node, and data can be communicated freely with no reliance on a master node.

The RSSI of each node is utilized for two components of the system. One component utilizes respective nodes' RSSI to determine system architecture such as the number of nodes in the system, which will assist in determining operable territory. Another component will utilize RSSI to update a robot's positioning relative to the node to the left of it, deemed *Node Rely*.

B. Node Initialization

At the start of the search, each node determines other nodes in the network via an established communication frequency determined by the coordinator node. After all nodes are established, the coordinator determines the positioning of each robot by sending a pre-determined amount of packets to each node

requesting an acknowledgement. Each router node requesting an acknowledgement then sends a packet of its calculated RSSI after the acknowledgement is received. The coordinator then stores the RSSI data received from each node in the RSSI database column corresponding to the vehicles *Node ID*.

After the database is instantiated, the coordinator node then sends the updated table to all router nodes within the network. After each vehicle receives the database, a *Node Rely* and *Node Send* are determined for each vehicle. *Node Rely* is defined as the node to the left of a vehicle, and will help determine positioning. *Node Send* is defined as the node to the right of a vehicle, and will be the node to which packets of data will be sent, allowing the *Node Send* to determine its RSSI respective to the sending node. Algorithm 1 shows a general outline of our node instantiation algorithm.

Algorithm 1 Node Initialization

```
Node Discovery
if this.node is Coordinator then
    for node in network do
        reqAck() → node
        while packet is None do
            packet ← AvailablePackets
        end while
        RSSI ← packet.data
        initRssiTable(packet.node, packet.data)
    end for
else if this.node is Router then
    while packet is None do
        packet ← AvailablePackets
    end while
    if packet.data then
        RSSI ← getRSSI()
        sendData(packet.node, RSSI)
    end if
end if
broadcastRssiTable()
```

C. RSSI Thresholding

To ensure no robot loses communication with the others in the system, we measure the radio devices range, and set a threshold limit ω that falls below the radio's maximum transmission range. If and when a robot has hit a maximum threshold, the robot will locate the closest node to the left of the vehicle via it's locally stored RSSI database.

This database is updated at every time step, as every router within the system will broadcast a packet containing its identification number (ID), time-to-failure (TtF), their RSSI respective to the requesting node, and a Boolean detailing the status of the mission. These data help determine the robots next actionable steps.

Based on research done in [9], we approximate a RSSI measurement using a Gaussian model and determine a useful threshold;

$$P(x) < \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

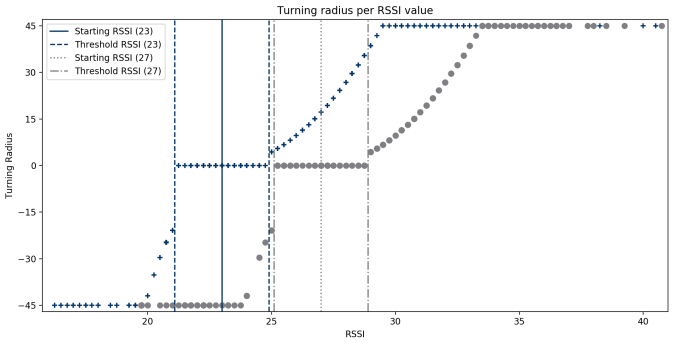

Fig. 4: **RSSI-Based Control.** Two $RSSI_{start}$ values of two seperate vehicles are graphed, and a threshold is determined. After a threshold is reached, the vehicle calculates a turning radius based on the calculated RSSI.

where $P(x)$ is a threshold value determining the critical point (past which an RSSI value can be evaluated as a high probability event), μ is the average of all received RSSI values for a particular node, and σ^2 is the variance. Using this model, we abandon small probability RSSI values in order to increase measurement reliability.

Equation 1 is utilized within a sliding window algorithm of RSSI values. This sliding window approach allows us to only take into account a certain subset of recently received RSSI values, removing older RSSI values in favor of more recent RSSI recordings. This allows us to filter out RSSI noise caused by either exterior ZigBee-based devices or energy level spikes caused by the on-board battery.

D. Robot Control

Algorithm 2 shows a general outline of our control loop. Both the steering angle and velocity commands of each vehicle are determined by the data received in the packets sent from a vehicles respective *Node Rely* and the calculated RSSI after a received packet from *Node Rely*.

Steering angle of the robots hosting the routers is determined via the RSSI. A magnitude is determined from the $RSSI_{start}$ and the $RSSI_{current}$, and a either a parabolic or exponential is used to determine the unscaled steering value. We then scale these values to the $\text{steer}_{\text{max}}$ and $\text{steer}_{\text{min}}$ values of the vehicle. The steering angle is updated 30 times a second, the rate at which each ZigBee module sends an acknowledge packet to its respective *Node Send*.

Fig. 4 shows the turning radius for a given RSSI value, given a $RSSI_{start}$ value and a pre-determined RSSI threshold. An exponential is used to determine the turning radius of the vehicles when the RSSI value falls below the lower threshold, ensuring safety from collision of approaching robots. A parabolic equation is used in the case when the upper threshold is reached, lending to a gradual advance towards the respective vehicle's *Node Rely*.

Algorithm 2 Robot Control Loop

```
nodesInit ← initNodes()
if nodesInit then
    while ROS and not MissionComplete do
        sendAcknowledge()
        packet ← readData()
        if packet then
            throttle ← determineRSSI(packet)
            broadcastBatteryStatus(batteryStatus)
        end if
        currentRSSI ← filterRSSI(window, rssi)
        coordinateVelocities(throttle, currentRssi)
    end while
end if
```

V. EXPERIMENTS AND RESULTS

To evaluate the feasibility of our system in its intended application, we performed a number of experiments. First, we conducted tests on the initialization sequence of each node in the system. Then, we tested the propagation of node-state information throughout the system. Tests were then conducted in-lab to ensure the safety of a predetermined path. From there, the remaining tests were done in an open field adjacent to the lab, testing all aspects of the proposed architecture.

A. Node Initialization

We conducted experiments on the initialization sequence of η+1 nodes in the system when configured in their starting positions. Table I shows the minimum, maximum, and average time required for initialization of η+1 nodes in the system. The measured time (in seconds) includes node finding and the time-of-flight for the initial acknowledge package.

TABLE I: Measured time requirements for the instantiation algorithm with N=2 nodes and N>2 nodes.

	N = 2			**N > 2**		
	Mean (s)	Min (s)	Max (s)	Mean (s)	Min (s)	Max (s)
Trial 1	3.38	2.94	3.81	8.68	4.81	12.67
Trial 2	6.03	3.70	8.36	9.88	6.79	13.32
Trial 3	8.77	5.73	11.80	4.67	0.77	9.16
Trial 4	2.09	1.39	2.79	5.13	3.84	7.18
Trial 5	8.51	5.35	11.66	5.69	1.24	11.18

Ensuring the nodes are correctly initialized and appropriately positioned in the RSSI table as they are positioned relative to the coordinator node is a great deal of importance. To test the accuracy of node positioning after the initialization sequence, we assess the ordering of nodes in the RSSI table after the initialization sequence and compare it to the physical ordering of the robots on the field. Robots and their radio transmission devices are spaced by approximately $1\,\mathrm{m}$. We test the accuracy with and without varying power levels applied to each ZigBee module to determine power level significance in RSSI-based distance determination. Table II shows the accuracy after three trials of initialization of η+1 nodes.

TABLE II: Initial positioning accuracy for η+1 nodes

Modules	No Power Level Variation (%)	Power Level Variation (%)
2	100	100
3	100	93.3
4	100	86.7

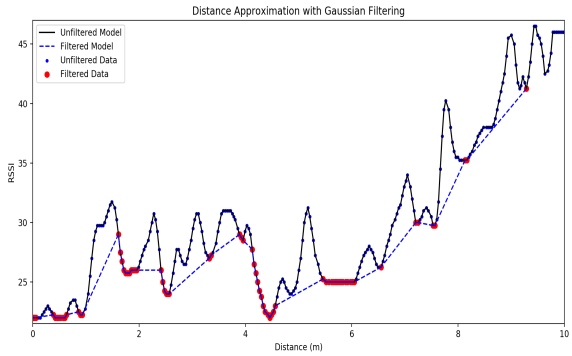

Fig. 5: **Gaussian Filtering of Observed RSSI Measurements**

B. RSSI Distance Approximation

We also conducted tests on the relationship between RSSI measurement and distance approximation to test the feasibility of RSSI-based navigation techniques. Figure 5 shows the unfiltered data points (in blue) collected while a robot moves from a starting position χ, to an endpoint $10\,\mathrm{m}$ away from point χ.

The maximum range for our radio transmission devices was determined to be $1600\,\mathrm{m}$. To account for our testing parameters, we manually default the maximum transmission range to $10\,\mathrm{m}$ by changing the power level of the XBee module. The test helps us determine our maximum threshold value, ω, and confirm the observed exponential relationship between an increase in RSSI values and the determined distance. Our maximum threshold value helps determine the point at which a robot must begin taking actionable steps to correct its positioning.

To account for noise in the environment, we use the Gaussian filtering method described in Section IV-C to filter the data from the last η calculated RSSI values. Figure 5 shows (in red) the data points that are considered high probability events, and therefore used in our navigation algorithm to direct steering angle of the robot at a time period τ, with filtered RSSI value $P(\tau)$.

C. Robot Control Experiments

We conducted extensive experiments with our hardware in an open grass field near our university. Fig. 7 shows the environment in which the robots were tested. Due to our focus on RSSI-based navigation, we do not implement any obstacle avoidance techniques into our navigation arrangements, although this will be considered in future work. The path for the coordinator robot is pre-determined, while each η+1 unmanned router vehicle situated after the coordinator robot

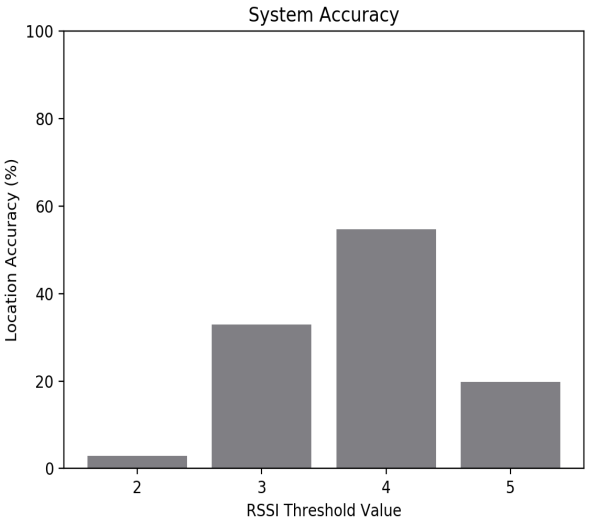

Fig. 6: **Accuracy of various RSSI thresholds**

determines its path using the RSSI-based navigation technique discussed in Section IV.

Figure 6 details the accuracy rates of several trial runs of our system at four different thresholds. In the graph, accuracy is defined as the deviation in turning radius required at a received RSSI value. At a threshold of 2 RSSI deviations from the starting RSSI, the system becomes over-reactant to small changes in movements of a vehicle's *Node Rely*. Due to our control function and its reliance on a starting RSSI value and a corresponding RSSI threshold, a threshold of 2 leads to erratic behaviour when signal dispersion and non-native received signals are taken into account.

Figure 7 shows several snapshots of the results of multiple test runs with the worst and best performing RSSI threshold values. In Panel 1 of Figure 7, the robots are in their starting positions and running the initialization sequence documented in Section IV-B. After both the reference RSSI value and the positioning of each node in the system is determined, the coordinator robot takes off.

Panel 2 of Figure 7 demonstrates a situation in which the coordinator robot is turning in the direction of the corresponding router robots path. With this configuration, 66% of the tests ran with a threshold value of 2 would result in the router robot diverging after a threshold was hit, and being unable to find the coordinator robot after some time. This inaccuracy is largely due to the threshold being set too low, resulting in large corrections to account for less response time. However, only 12% of the tests run with a RSSI threshold of 4 would result in a collision. The router robot makes the appropriate correction to stay in line with the coordinator robot, without losing connection.

Panel 3 of Figure 7 demonstrates a situation in which the coordinator robot is turning away from the corresponding router robots. This configuration coupled with a RSSI threshold of 2 would result in collision 46% of the time. This high inaccuracy

Fig. 7: **RSSI Threshold Results: 2, 4**

is due to the low threshold value set. The router robot gradually corrects its turning radius after each RSSI update, however is unable to correct itself after the threshold is hit. In tests ran with this configuration and a threshold of 4, no collisions resulted. The largest issue with this configuration and a RSSI threshold of 4 was signal dispersion, which led to an overall decrease in the accuracy rate of the system.

We've found that increasing the RSSI threshold gradually increases the overall accuracy of the system, until we observe diminishing returns after a threshold of 4 RSSI. In a system with a larger array of robots, with each robot spaced farther apart, higher accuracy rates would be observed. This is due to more constraints put on each robot within the system, leading to fewer egregious corrections.

VI. CONCLUSION

In this paper, we introduce a multi-robot search-and-rescue technique based on the received signal strength (RSSI) of radio transmission devices installed on each robot. For this system, we utilize a control algorithm that determines actions based on both the RSSI and system status data shared via packets

broadcast out from each router to other routers in the system organized in a mesh-network topology.

Several thresholding values were tested and assessed based on overall accuracy relative to a defined model. A threshold deviation of 4 was determined to be the most accurate, with diminishing returns observed after that point. We concluded that a better filtering method, along with a refined control algorithm, would lead to a higher accuracy rate for each threshold value. Future work includes adding obstacle-avoidance techniques to the coordinator robot, which would allow for a truly autonomous system. Another path of interest would be to allow the robot to train its parameters to increase accuracy via simulation.

VII. ACKNOWLEDGMENT

This research was supported by the National Science Foundation under award number IIS-1757929.

REFERENCES

[1] Population Division of the United Nations Department of Economic and Social Affairs *Urban Prospects* UN Department of Public Information, 2018

[2] C. E. Perkins and E. M. Royer, *Ad-hoc on-demand distance vector routing,* Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[3] T. Balch and R. Arkin, *"Behavior-based formation control for multirobot teams,"* IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 926-939, 1998.

[4] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, *"Probabilistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation,"* IEEE Transactions on Robotics and Automation, vol. 18, no. 5, pp. 662-669, 2002.

[5] N. Zhou, X. Zhao, and M. Tan, *RSSI-based Mobile Robot Navigation In Grid-pattern Wireless Sensor Network,* Chinese Automation Congress, November, 2013.

[6] S. Srivastava and B. Manoj, *Path Planning Algorithms for Mesh Networked Robots based on WiFi Geo-location* 2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS), December, 2012.

[7] S. Liu, L. Mao, J. Yu, *Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems* 2006 International Conference on Mechatronics and Automation, June, 2006.

[8] L. Caetano, T. Nascimento, M. Oliveira, G. Rocha, *Expanding the Coverage Area of a Formation of Robots Through a Mesh Network and a Real Time Database Middleware* 2014 Brazilian Symposium on Computing Systems Engineering, November, 2014.

[9] Z. Jianwu, Z. Lu. *Research on distance measurement based on RSSI of ZigBee* 2009 ISECS International Colloquium on Computing, Communication, Control, and Management, August, 2009.

[10] H. Lee, W. Jeon, D. Jeong. *A Practical Indoor Localization Scheme for Disaster Relief* 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), June, 2017

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, *ROS: an open-source robot operating system* In ICRA workshop on open source software, 2009

[12] R. Fitch, and R. Lal, *Experiments with a ZigBee Wireless Communication System for Self-Reconfiguring Modular Robots* IEEE International Conference on Robotics and Automation, 2009

[13] Z. M. Fadlullah, M. M. Fouda, N. Kato, A. Takeuchi, N. Iwasaki and Y. Nozaki, *Toward intelligent machine-to-machine communications in smart grid,* IEEE Communications Magazine, vol.49, no.4, pp.60-65, April 2011.

View publication stats