Multi-Agent Mapping with Imperfect Communication on Commodity Hardware

Clifford Bakalian and Justin Goodman

I. INTRODUCTION

The field of robotics has grown exponentially in the past few decades. As such, there exist many problem domains, such as mapping and navigation, in the robotics community. We notice, however, that oftentimes researchers solve robotics problems using simulations or expensive custom-built robots. In this paper, we challenge these practices by actually building low-cost robots and evaluating their performance under a real-life mapping task.

Specifically, our research goal is to evaluate the viability of state-of-the-art distributed mapping algorithms with hobby-tier hardware in real life. Oftentimes, as we are doing here, pushing the bounds on current literature will provide insight towards future work. As an added layer of complexity, we introduce mapping under imperfect communication to our experimental set-up. Imperfect communication results from real-world, and not necessarily simulated-world, effects such as malicious actors, bad sensors, and physical obstacles. While we ran out of time to achieve our goal, mapping under imperfect communication contributes saliently to future work and we plan to apply it on our experimental set up in the future.

II. RELATED WORK

In 2010, the Australian and US Departments of Defence put on MAGIC – the Multi Autonomous Ground-robotic International Challenge [5]. Among other challenges, teams had to develop distributed robots to autonomously map out a 500m $\times$ 500m area. Approaches included LiDAR sensors, omnidirectional vision, and laser scanners for mapping and GPS for positioning. Most teams implemented SLAM algorithms for indoor mapping. For this paper, we do not assume the same hardware access as these teams. Modules like LiDAR and laser scanners are significantly more expensive than analogous commodity hardware. Additionally, GPS requires an unobstructed view of the sky and has unfavorable error for small agents. Thus for the context of this paper, this would not allow for commodity hardware, nor accurate indoor mapping.

Work has been done related to bounded communication which can assist with imperfect wireless communication. Feng Wu et. al. solve multi-agent DEC-POMDP planning under restricted communication using linear programming [9]. We do not use this approach however we do believe this would be something interesting to include for future works.

Mapping is a rich research topic in robotics. Specific to distributed mapping, [2] use pose graph optimization. We want to go a simpler route since we have limited hardware and so out approach will use occpancy grid mapping using logs odds.

Joseph Rothermich et. al. use robotic swarms for localization and mapping [8]. In their study, their robots lack range sensors and instead rely on bumping for mapping. Localization is done using distance from global landmarks emitting global positions. We want to avoid reliance on global beacons and so we take a different approach to localization by using sound. We plan on using a speaker and 2 microphones to determine where each agent is relative to the others.[6], [4]

Distributed localization and mapping is generally an active research area. Especially for large environments, single-node CPU and limited RAM resources become too sparse. Robert G. Reid wrote his 2016 doctorate thesis on the topic where he utilized GPUs to re-mesh each agent's local occupancy grid [7]. Since we are focusing on a smaller scale we do not need large computational power to merge sub-maps, since we are starting out knowing where each agent begins relative to the others, (making the corresponding sub-maps relative as well). Additionally our approach uses intermittent merging rather than a single merge at the end.

III. PROBLEM FORMULATION

Suppose that a team of robots $n \geq 2$ and a central server are deployed to map an unknown environment. The robots may communicate to each other via audio signal and to the server via Wi-Fi. Both signal streams may contain noise. Furthermore, both signal streams may close, and re-open, at any point with nonzero probability. Finally, assume the server knows the relative positioning of each robot to themselves, and the robots will send pose updates after movement commands. For this paper, we evaluate the feasibility of solving this problem given low-budget commodity hardware.

IV. APPROACH

When planning, our distributed mapping algorithm must account for multiple agents *and* communication drop-outs. We leave the latter to future work, as we ran out of time working on the former.

Our first task is to localize each agent. Without GPS or a beacon-friendly environment, we appealed to acoustic localization. We discuss this project separately for another class. For the scope of this paper, assume the server has multilaterated each agent to obtain a relative positioning of all agents with respect to a leader agent.

The server can begin mapping once it knows each agent's relative coordinates. We implement this using occupancy grid mapping. Server-side, each agent acts as an API – a fancy sensor, if you will – to the environment. The server can ask an agent for distance readings, and it can tell an agent to move with two degrees of freedom. These both enable mapping and motion respectively.

Our second task is to account for communication drop-outs. Given more time, our approach would have been as follows. The server generates a *trajectory* for each agent. The server sends this trajectory, and listens for when the agent is finished. If communications drop out at this step, we retry whenever we can. Next, the agent is always listening for new trajectory updates unless it is currently performing one step in a trajectory. The agent keeps a local log of events, just as if the server were pinging it every time it finishes a motion command. Upon receipt of a new trajectory, the agent will send back its local log of events to the server, and wait for a corrected trajectory (the server can only guess at where the agent is at this point). The server receives the log, plays it forward to reconstruct and merge the local occupancy grid map, and calculates and sends a new trajectory for the agent. If communications drop out at any point, both parties retry. We did not have time to evaluate what should happen if an agent finishes its local trajectory before receiving an updated one. One approach may be to roam semi-randomly, while another may be to perform some local path planning. Either way, the agent is restricted to the computational capabilities of its host micro-controller. For multiple agents, occupancy grid mapping abstracts neatly – we treat overlapping different-agent sensor readings as two successive readings from one sensor.

With all of this in mind, we built a real team of $n = 3$ 25cm $\times$ 15cm mobile robots from commodity hardware. Altogether, each agent costs \$58. We believe this is well within reason for an online, localization-capable, and mapping-capable, low-cost build. A parts list is presented in table I, and a wiring schematic can be found in our GitHub Repository [1]. We chose ultrasonic sensors due to their low cost and high availability. Our motion-processing units contain gyroscopes and accelerometers. We use the ultrasonic sensor for range finding and travel distance judgment, and the gyroscope for accurate rotation measurements.

Note that we use two micro-controllers (micro control units – MCUs) for this project. We require this because we need wireless communications (Wi-Fi), and a handful of analog inputs for audio and motion capturing. The boards communicate via serial communication. Board synchronization is vital due to our placement of sensors – one board contains access to the ultrasonic sensor, while the other contains access to the motion-processing unit and motors.

We plan on placing these robots in an enclosed area with multiple obstacles made of wood. Imagine a maze, roughly scaled to the robot size – 5-10m $\times$ 5-10m area. The obstacles can be moved around randomly before the start of each run. We will measure success by comparing the accuracy of the global map generated by the agents to the actual known world environment.

Once the robots are placed and "localized", we begin our occupancy grid mapping routine. Note that any path planning algorithm would suffice here to decide where to send the robots, so long as it accounts for noisy sensor readings. One might frame our mapping problem as a Markov decision process (MDP) and use any well-known solver (eg: Monte-Carlo Tree Search) for deciding on robot trajectories. Future work is required for exact implementation details with respect to imperfect mapping.

We present our entire algorithm in 1 and 2. Our actually-implemented mapping policy is as follows. Randomly pick K random points within the occupancy grid map which are within some restricted radius of the robot i which needs planning. Choose the point K_p with the smallest magnitude occupancy grid value (which corresponds to the "most unknown" point). Then, perform breadth-first search from K_p to i and yield a trajectory. Send commands to i such that it follows the trajectory. Once K_p is reached, repeat. Note that, due to high-noise commodity hardware, our server can only *estimate* where each robot actually is.

Algorithm 1: Multi-Agent Mapping – Server

```
wait for robots to register;
localize();
// begin mapping;
while within computation budget do
    $c_i$ ← retrieveRobotCommunication();
    if $c_i$ is forward move result then
        $u$ ← requestUltrasonic($i$);
        updateGrid($i, u$);
    end
    planTrajectory($i$);
end
```

Algorithm 2: Multi-Agent Mapping – Robot

```
setupBoard();
syncBoard();
while alive do
    $e$ ← receiveSeverTransmission();
    if $e$ is move request then
        doMove($e$);
    end
    if $e$ is ultrasonic request then
        takeMeasurement($e$);
    end
    transmitResult();
end
```

V. RESULTS

Our prototype robots can be seen in figure 1. Our mapping and exploration code can be seen in our GitHub repository [1]. Most of our time in this project was spent wiring our robots and writing control code.

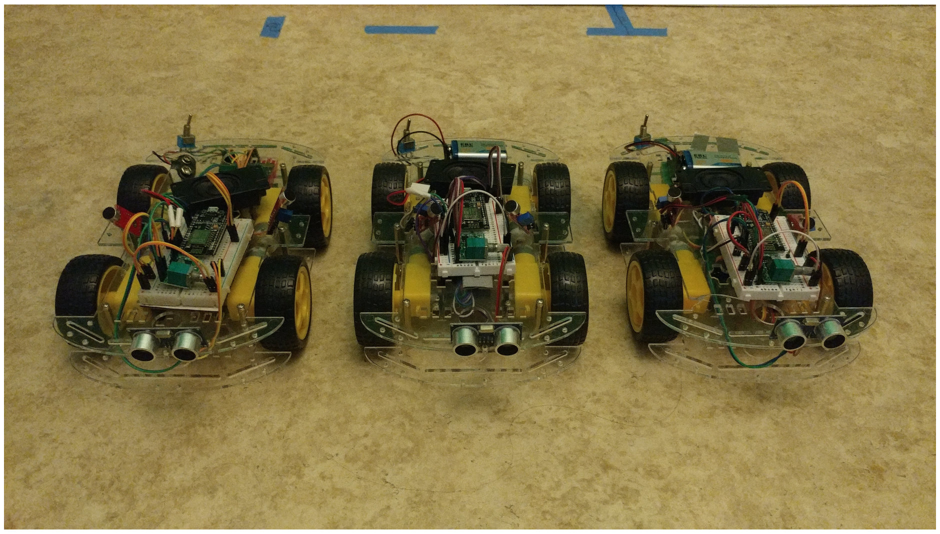

Fig. 1. All three robots.

Our results leave much to be desired, yet are hopeful towards future work. Due to time constraints, we could not test our robots inside a large maze. Figure 2 displays one of our tests. Our robots can indeed sense ranges, however

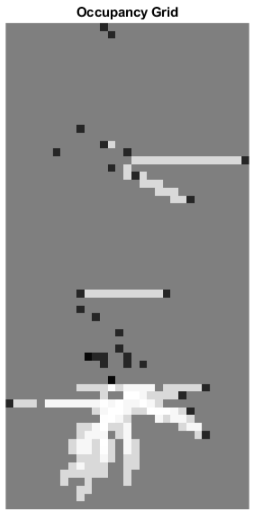

Fig. 2. Example occupancy grid map generated by two agents. The robots were placed in a 230cm $\times$ by 190cm arena. Barring walls, there were no obstacles in the environment. Each grid cell represents 10 centimeters. We used an odds value of 0.85. As in standard occupancy grid mapping, darker colors mean the cell is more likely to be occupied. We did not include our third robot in this test due to technical failure. The demo that created this map is available in our GitHub repository [1].

the resulting map is highly inaccurate. Most of this error is caused by our inaccurate movement sensors and our lack of environmental sensors.

VI. LIMITATIONS

Most of our error is caused by inaccurate motion sensors, and low-quality motors. Furthermore, our wheels lacked traction in our tested environment. If we increased our budget, we would use higher quality motors, a motor controller module, or even use a tread approach as seen in [3]. This would greatly reduce the error obtained during movement. Additionally, for more accurate occupancy values, we would purchase more (inexpensive) ultrasonic sensors. Alternatively we could double our budget and consider different types of environment sensors like LiDAR. This is less favorable, though, as it takes away from the main point of the paper.

For future work, we may make our mapping algorithm more efficient and knowledgeable. A greedy approach to mapping may be more time efficient, yet be unfavorable for inaccurate sensors and robot battery life. A learning approach may provide better path planning. We may also take better advantage of our agents being distributed. One consideration we had not implemented, yet is crucial for optimal planning, is to ensure our robots implement better robot-robot collision avoidance.

VII. CONCLUSION

While we may not have achieved our research goal, we believe that we have still shown significant progress towards our end goal of creating a low cost robot which can map an enclosure. We highly encourage the reader to view our server and control code available in our GitHub repository [1].

VIII. CONTRIBUTIONS

We list our contributions, and associated author, in table II.

Author	Contribution
Cliff	robots algorithm design
Justin	server algorithm design

TABLE II
AUTHOR CONTRIBUTIONS.

References

[1] https://github.com/jugoodma/818bw-project.

[2] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I Christensen, and Frank Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *The International Journal of Robotics Research*, 36(12):1286–1311, 2017.

[3] David Gonzalez-Arjona, Alberto Sanchez, Fernando López-Colino, Angel De Castro, and Javier Garrido. Simplified occupancy grid indoor mapping optimized for low-cost robots. *ISPRS International Journal of Geo-Information*, 2(4):959–977, 2013.

[4] J. Hornstein, M. Lopes, J. Santos-Victor, and F. Lacerda. Sound localization for humanoid robots - building audio-motor maps based on the hrtf. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1170–1176, 2006.

[5] Ani Hsieh and Simon Lacroix. Editorial: For the jfr special issue on "multiple collaborative field robots". *Journal of Field Robotics*, 29(5):687–688, 2012.

[6] P. Karimian, R. Vaughan, and S. Brown. Sounds good: Simulation and evaluation of audio communication for multi-robot exploration. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2711–2716, 2006.

[7] Robert George Reid. Large-scale simultaneous localization and mapping for teams of mobile robots, 2016.

[8] Joseph A. Rothermich, M. İhsan Ecemiş, and Paolo Gaudiano. Distributed localization and mapping with a robotic swarm. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, pages 58–69, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[9] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487 – 511, 2011.

#	Part
1	ESP8266 NodeMCU
1	Arduino Nano
1	HC-SR04 Ultrasonic Sensor
2	KY-038 Microphone
1	PAM8403 5V Amplifier
1	4Ω 3W Speaker
1	GY-521 MPU-6050 Module
1	L9110S H-Bridge
4	3-6V Motor
1	DIY Robot Chasis
1	Toggle Switch
1	9V 600mAh Li-ion Rechargeable battery

TABLE I

Parts list. Note that our robot chassis included the four DC motors, as well as mounting hardware. Note that our Wi-Fi board is an entire micro-controller made by HiLetgo (ESP8266 CP2102 ESP-12E). Note that we use the LAFVIN Nano with Atmel Atmega328P-AU micro-controller unit and CH340G USB chip. We omit our specific wire and breadboard counts, but our most recent wiring schematic is available in our GitHub repository [1].