2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Lightweight Multi-robot Communication Protocols for Information Synchronization

Murtadha Alsayegh, Ayan Dutta, Peter Vanegas, Leonardo Bobadilla

***Abstract*— Communication is one of the most popular and efficient means of multi-robot coordination. Due to potential real-world constraints, such as limited bandwidth and contested scenarios, a communication strategy requiring to send, for example, all n bits of an environment representation might not be feasible in situations where the robots' data exchanges are frequent and large. To this end, we propose and implement lightweight, bandwidth-efficient, robot-to-robot communication protocols inspired by communication complexity results for data synchronization without exchanging the originally required n bits. We have tested our proposed approach both in simulation and with real robots. Simulation results show that the proposed method is computationally fast and enables the robots to synchronize the data (near) accurately while exchanging significantly smaller amounts of information (in the order of $\log n$ bits). Real-world experiments with two mobile robots show the practical feasibility of our proposed approach.**

I. Introduction

In fundamental mobile robotics applications such as task allocation, environmental monitoring, and search-and-rescue, coordination among the robots via robot-to-robot communication is of utmost importance [1], [6], [17]. As a concrete motivating scenario, suppose that we have a group of robots that are deployed in a contested environment spanning a large geographical area. This environment is highly dynamic, where new obstacles are appearing and disappearing frequently. All the robots need to keep an updated representation of the environment for decision making. However, due to environmental, technical, and adversarial constraints, the bandwidth available for communication is limited. Our testbed motivated by this problem is shown in Fig 1.

Most of the studies on multi-robot communication, such as the one described above, assumes a smart underlying communication framework, designed by an oracle, is available to the robots or that they follow a naive form (having a $O(n)$ complexity) of communication where the robot Alice sends all the n bits of her representation (X) to the robot Bob, and then Bob contrasts this representation with his own. However, this simple protocol will consume a large amount of bandwidth and computational resources. Since changes are constantly happening, sending n bits every time might become a bottleneck. Furthermore, in an environment where the available bandwidth is limited, a high cost of communication such as this might potentially result in a breakdown of the coordination process [17]. Therefore, the objective of the robots should be to exchange their perceived data represented by *less than* n bits.

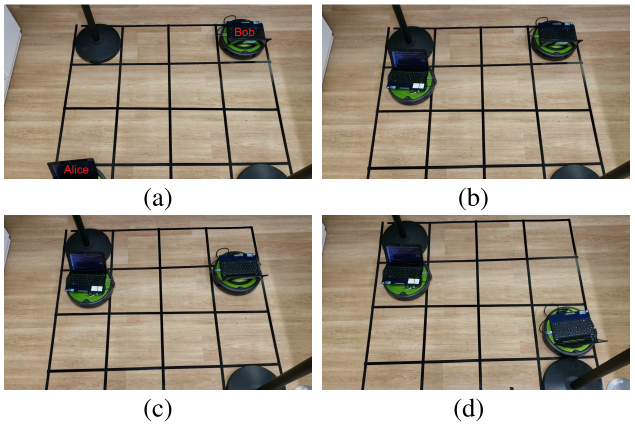


Fig. 1. An experimental demonstration of our lightweight communication protocols with two parties Alice and Bob using two iRobot Create 2.0 platforms: (a) At time $t = 0$ second, Alice and Bob start executing their paths, Alice intends to move north, and Bob intends to move south; (b) At time t=16, Alice encounters, and obstacle and the equality of their representations is checked, they noticed they have different maps; (c) At time $t = 22$, Bob starts his path; (d) At time $t = 29$ Bob finds another obstacle and calls for a testing in their representation, again it is found that their maps are different.

In this paper, we present an approach, inspired by the seminal work of Yao [31], to design protocols in which the robots share less than n bits of data to synchronize their local representations of the environmental state. Our proposed approaches are easy to implement, fast, and need significantly less data-exchange to answer the question: "*Do we have the same representation of the state?*" If the answer is yes, they do not need to take any further action. On the other hand, if they are not the same, using our proposed *error correction* approach, the mismatches in the representations are detected and attempted to be fixed. We have selected a multi-robot map merging case study along with numerical simulations to test our proposed approach. Numerical results show that our proposed techniques are fast (taking in order of a few seconds to synchronize large representations), and they can accurately decide whether the robots have the same map representations. The close-to-reality case study also validates these findings. Our primary contributions in this paper are as follows:

- We employ protocols for reducing the communication between two robots and synchronizing the data afterward, which have significant implications in bandwidth-limited environments.

M. Alsayegh, P. Vanegas, and L. Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA {malsa061@,pvane003@,bobadilla@cs.}fiu.edu
A. Dutta is with the School of Computing, University of North Florida, Jacksonville, FL 32224, USA a.dutta@unf.edu

- We implement and rigorously test our proposed methods with numerical simulations and on a highly applied multi-robot map-merging application. Furthermore, we present a proof-of-concept experiment to show the practical feasibility of our ideas.

II. Related Work

Our ideas take inspiration from the field of *communication complexity* theory [2], [14], [12], [13], [22], [31], which studies the amount of communication (i.e., bits exchanged) required to solve a given problem when the input is split between two (or more) parties. In its original formulation, there are two agents (Alice and Bob), each of which has a binary string of size n – x and y respectively. Alice wants to compute a function $f(x, y)$ that depends on both inputs using the least amount of communication between them. A closely related extension of Yao's results that share the same goals of our research can be found in the area of *document exchange* [5]. Our work is also related to subspace synchronization using coding theory [25] and set reconciliation [18]. However, our ideas differ as in robotics applications, synchronization is required frequently, unlike document exchange, and our goal is to have a practically feasible approach instead of finding theoretical guarantees only.

In a multi-robot system, especially where a large number of robots are present, and the amount of bandwidth is limited, reducing the size of the communicated messages can tremendously benefit the coordination process. One of the most prominent works on multi-robot communication is by Eric Klavins [11], where the scalability of multi-robot algorithms is studied, and ideas of multi-robot communication complexity are introduced. This work took inspirations from the average-case communication complexity considered in [28] for distributed algorithms. The author has proposed four different complexity classes for such systems. However, the complexity was measured in terms of the number of messages passed among the robots. Unlike this, in our approach, we reduce the size of the messages, which can potentially be merged with the reduced message-passing models proposed in [11].

The communication complexity of task allocation was analyzed in [20] using discrete and continuous models of communication. Giridhar and Kumar [10] studied the maximum rate to communicate functions to a sink node. In [17], the authors have proposed a novel technique, with which the robots decide what information to share with the others in a bandwidth-limited environment. Recently, Dutta et al. have proposed an efficient technique to minimize the space complexity required to store the communication graph in a large multi-robot system while keeping the maximum hop count within a bound [7]. Instead of reducing the space requirement for robot-to-robot communication while sharing n bits of data as proposed in [7], our approach in this paper aims to minimize the size shared data itself, which potentially can be combined with the techniques in [7] for magnified impact on multi-robot communication. In [8], the authors have asked the question: "*how the robots should communicate?*". We try to answer this question in this paper partly. Our proposed work advances the state-of-the-art by introducing novel minimalist data synchronization mechanism to reduce the communication complexity, which is currently missing in the multi-robot system literature.

Our effort, especially in the tested case study, has similarities with the problem of *map merging* [3], [4] where individual robots build local maps that then are merged through communication. Our ideas differ from these works and are complementary to them in two important ways. First, although our experiments and case studies focus on detecting and correcting differences in occupancy grids, our approach can be applied to any robot representation that has a binary encoding (e.g., images, cloud points, or geometric maps). Secondly, our main contribution, the minimization of bits transmitted, was not the focus of the map merging process in [3]. Our ideas are also connected to SLAM approaches in communication-constrained environments [23], [15]. Our work also contributes to the recent interest in security and privacy issues in multi-robot networks [9], [24], [32], [21], [30], [16] since minimizing the number of bits exchanged among the robots can lead to better guarantees in privacy and security.

III. Problem Formulation

In our problem formulation, we have two robots, Alice and Bob, that are moving in a two-dimensional workspace $\mathcal{W} \subset \mathbb{R}^2$. $\mathcal{W}$ contains an *obstacle region* $\mathcal{O} \subset \mathcal{W}$. Both Alice and Bob move in the *free space* of the environment, which is defined as $\mathcal{E} = \mathcal{W} \setminus \mathcal{O}$. The robots are modeled as point robots and their positions in $\mathcal{E}$ can be sensed through GPS and other state estimation sensors.

Both robots are equipped with sensors such as laser rangefinders for sensing their surroundings. As the robots move in the environment, they collect information using their sensors from the environment and creates a map, an *occupancy grid* for example [3], [27], over $\mathcal{W}$ where 0 represents a free space (the robots can traverse), and 1 represents an obstacle. The occupancy grid is encoded in a vector of size n, let $X = \{0, 1\}^n$ be the data held by Alice and $Y = \{0, 1\}^n$ be the encoding of Bob's data. Alice and Bob can communicate potentially through a low-bandwidth channel.

As Bob and Alice move and explore their surroundings, their local representations of the same environment change due to the newly discovered obstacles and potential changes in the environment. For this reason, both robots need to check periodically if their representations are the same. However, they do not want to send the whole n bits of their representations due to the communication channel limitations. This scenario motivates our first problem of interest.

Problem 1. Equality Checking: *Given Alice and Bob's representations of the same environment, X and Y, they need to check if they have the same information while minimizing the amount of information sent.*

Besides checking whether they have the same information or not, they need to synchronize their representations and ensure they have the same data after the communication while minimizing the number of bits sent. This motivation leads to our second problem of interest.

Problem 2. Error Correction: *Given Alice and Bob's representations, X and Y, synchronize them to ensure that the robots have the same information while minimizing the number of bits sent.*

IV. Methods

A. Equality Testing

To solve the first problem, we present a procedure based on the results from the field of communication complexity [31], [2], [14]. This sub-field of theoretical computer science studies how to calculate the value of a function $f(x, y)$ by two parties Alice and Bob where Alice holds x and Bob holds y. Both x and y are n-bit strings. The parties want to calculate f using the least amount of communication (i.e., exchanging minimal bits) possible while ensuring that at least one party can calculate the result. Next, sending one more bit will ensure that both parties have the result of the computation.

The field of communication complexity primarily focuses on analyzing protocols based on the worst communication complexity denoted as $D(f)$, which is defined as the minimum number of bits that needs to be exchanged between the parties in the worst case. This metric contrasts to traditional analysis of algorithms since the protocol analysis is not concerned about the computational complexity, nor the amount of memory used but in minimizing the number of bits sent back and forth between the parties.

Protocol 1 Equality Testing

Inputs. Alice's binary vector X and Bob's binary vector Y.

Goal. Both robots, Alice and Bob, will check whether they have the same binary vector.
The protocol:

1) **Setup.** Before their deployment, Alice and Bob are given a database $R = \{r_1, r_2, \ldots, r_{100n}\}$ that contains $100n$ random binary strings of size n each.
2) Bob initializes a *counter* to zero.
3) Repeat k times the following subroutine:
   a) Alice picks a random number $i \in \{1, 2, \ldots, 100n\}$.
   b) Alice calculates the dot product for binary vectors $b = X \cdot r_i$
   c) Alice sends b and the position i to Bob.
   d) Bob calculates the dot product for binary number $b^{'} = Y \cdot r_i$.
   e) Bob compares b and b'. If they are different, Bob increments the counter.
4) If the counter is greater or equal than zero returns 0 (X and Y are different); otherwise return 1 (X and Y are the same).

One canonical example studied in communication complexity is the equality problem (function), that is formally defined as $EQ : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ where $EQ(x, y) = 1$ if and only if $x = y$ and 0 otherwise. It has been proven that the *deterministic* communication complexity of the equality function $D(EQ) = n$, which means that in the worst case, the original n bits need to be sent. This is prohibitive for the robotics application of our interest where occupancy grids can easily be in the order of 10^6. Instead, we will implement a protocol based on *randomization* with better communication complexity than the deterministic worst case of n. This procedure is described in Protocol 1 and uses the idea of computing dot products described in detail in [14].

Proposition 1: Protocol 1 calculates whether two binary vectors X and Y of size n are the same with the communication cost of $k(log100n + 1)$ bits.

Proof:

Sketch A database R, consisting of $100n$ random binary strings to size n, is given to the robots before the mission. Therefore, R does not need to be transmitted every time we are interested in checking the equality between the maps. The protocol runs k times (the subroutine in line 3 in Protocol 1) and the only information sent are the integer i, which can be encoded using $log100n$ bits and b, which has a length of 1 bit, resulting the communication complexity to be $k(log100n + 1)$. The fact that the size of R should be $100n$ is used to calculate the error rate of the protocol [14]. The intuitive idea is that R should be a set of strings with enough randomness to run the protocol. ■

Proposition 2: Protocol 1's probability of correctness is at least $(1 - \frac{1}{2^k})$.

Proof:

Sketch To analyze the probability of correctness of the protocol, we know that if X and Y are the same, the protocol will not fail (the dot products with r_i will be equal). If X and Y are different, there is a probability of $1/2$ that $Y \cdot r_i = X \cdot r_i$, which will give an incorrect result (a detailed explanation can be found in [14]). We will use the subroutine in step 3 as a Monte Carlo algorithm [19] with a one-sided error. As such, the failure probability can be reduced (and the success probability amplified) by running the algorithm k times. Therefore, if X and Y are equal, the answer is always correct, and if they are different, then the answer is correct with a probability of at least $(1 - \frac{1}{2^k})$. ■

Although Protocol 1 sacrifices the determinism achieved by sending n bits, it provides a dramatic reduction in communication complexity, which makes it practical for large maps. We have evaluated the effect of k on the success rate of equality checking through experiments described in Section V.

B. Error Correction

Once Bob and Alice detect that their maps are different after executing the communication protocol described in the last section, a natural next step is to exchange only the bits of information in the map that are different instead of sending the n bits to synchronize them. To this end, we propose an algorithm that uses Protocol 1 as a subroutine for such error correction and is shown in Algorithm 1.

Algorithm 1: ERRORCORRECTION(X, Y,l,u,res)

```
Input: Alice's binary vector $X$, Bob's binary vector $Y$,
       lower-bound $l$, upper-bound $u$, resolution for
       error correction $res$
Output: Positions that are different
1  $midpoint \leftarrow$ FLOOR($\frac{l+u}{2}$)
2  if LEN($X[l \dots u]$) $\leq res$ then
3      if EQUALLINEAR ($X, Y, l, u$) $== 1$ then
4          return FALSE
5      else
6          return $l, u$
7  if PROTOCOL1 ($X, Y, l, u$) $== 1$ then
8      return FALSE
9  else
10     ERRORCORRECTION($X$, $Y$,$l$,$midpoint$,$res$)
11     ERRORCORRECTION($X$, $Y$,$midpoint + 1$,$u$,$res$)
```

Algorithm 1 will return the intervals where X and Y differ. It will work on X (Alice's data) and Y (Bob's data). l and u represent the lower-bound and upper-bound of a sub-array. Initially, the algorithm will be called with $l = 1$ and $u = n$. res represents the resolution in bits at which a difference between X and Y is detected and it is a low constant number that aims to model the packet size which can be exchanged between Alice and Bob.

Line 1 of the algorithm calculates the midpoint of the array. Lines 2-6 are called when the length of the vector is less than the resolution. In this case, it will decide in constant time (EQUALLINEAR) whether the segment of the map from l to u is different and report the positions l and u. A simple linear comparison can be employed to realize the EQUALLINEAR function in practice.

If the currently considered array segment is longer than res, it will go to (line 7) and call a slightly modified version of Protocol 1 that checks equality between positions l to u. If the vectors are equal, the algorithm will stop; otherwise, it will call itself (lines 10 and 11) twice with half the array elements in each call. This divide-and-conquer strategy has similarities to algorithms such as binary search.

There are two points in Algorithm 1 that need the data from Alice and Bob: line 3 (EQUALLINEAR) and line 7 (PROTOCOL 1). This algorithm can be executed in a client-server fashion where Alice (client) runs the algorithm and invokes lines 3 and 7 as needed. Algorithm 1 can be converted into an iterative procedure to facilitate the distributed implementation.

Preliminary Algorithm Analysis. Algorithm 1 corrects the discrepancies between the binary vectors X and Y using $log(\frac{n}{res})k(log100n + 1) + i$ bits for communication in the worst-case, where n is the length of the vector, i is the numbers of bits where X and Y differ, k is the number of iterations of Protocol 1, and res is a small constant.

To analyze Algorithm 1, recall that we are not interested in calculating the space and computational costs of the procedure, as only the communication cost is our primary interest. There are only two places where communication is needed in Algorithm 1 – lines 3 and 7. Line 3 is only called when the segment of length res has differences; this would need to communicate at most i bits. Line 7 is needed at each call of the recursion. The recursion tree will stop growing when the length of the array reaches $res = \frac{n}{2^d}$, where d is the depth of the recursion tree. Solving for d, we have $d = log(\frac{n}{res})$. At each level of the recursion, in the worst-case, Protocol 1 is called with the whole array, which has a communication cost $k(log100n+1)$ following Proposition 1. Multiplying the number of levels of the recursion with the communication at each level, we have $log(\frac{n}{res})k(log100n+1)$. Thus adding the communication costs of Lines 3 and 7 gives the desired result.

This communication cost is an improvement over the naive n-bit communication protocol to correct differences between X and Y in the context of our application where n can be large and the number of bits they differ (i) is comparatively small. Algorithm 1 calls Protocol 1 as a subroutine, and as such, its reliability depends on it.

The reliability of Algorithm 1 in correcting discrepancies between binary vectors X and Y can be modeled as a Binomial Distribution $B(m, \theta)$ where m is the number of segments of size res where the maps differ and $\theta = (1 - \frac{1}{2^k})$. Each of the m different segments will be detected when Protocol 1 is called. According to Proposition 2, the probability of detecting the error when the segments are different is $\theta = 1 - \frac{1}{2^k}$ and each of the calls to check if the segments are different can be modeled as an independent Bernoulli trial.

V. EXPERIMENTAL RESULTS

First, we are interested in testing the quality of our proposed approaches in a wide range of simulation experiments and then transfer them to a real exploration experiment using two robots. The simulation experiments are run on an Ubuntu 18.04 LTS machine with an Intel i7 3.60GHz CPU and 16 GB RAM using Python 3. Each simulation experiment is run 50 times and the average result is presented next along with standard deviation data unless mentioned otherwise.

A. Numerical Simulation of Protocol 1

In these experiments, random binary strings, X and Y, of size $n \in \{16, 32, 64, 128, 256, 512, 1024\}$ are generated with the database R of strings also being random. We are primarily interested in two metrics to test the quality of Protocol 1: success rate and time. Success rate (S.R.) denotes how many times out of 50 test runs Protocol 1 was able to successfully determine whether input strings X and Y were equal or not. The result is shown in Fig. 2. We observe that with the increasing value of k, the S.R. value also increases. This result is supported by the theoretical bound described in Proposition 2. Regardless of the value of k and n, we are always able to achieve a success rate of $> 90\%$ except with $k = 3$ and $n = 128$.

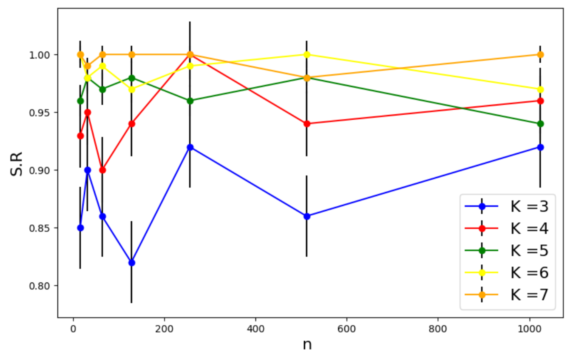


Fig. 2. Success rate of Protocol 1

On the other hand, the protocol's execution time is an important aspect, especially in robotics. The result of this metric is shown in Fig. 3. As can be noticed, time increases linearly with the size of the input strings X and Y, the minimum time being 0.04 sec. with $n = 16$ and the maximum being a negligible 3.17 sec. with $k = 7$ and $n = 1024$. As expected, with a higher value of k, Protocol 1 takes a little more time as more data exchanges are involved.

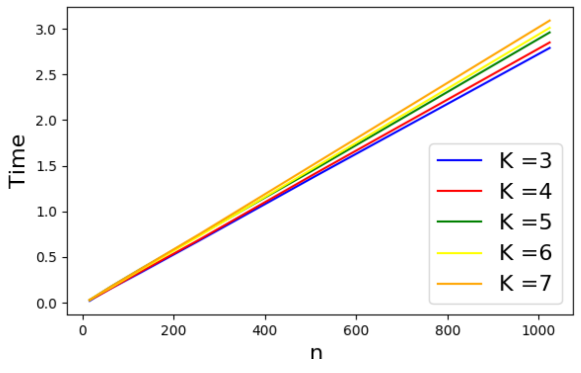


Fig. 3. Execution time of Protocol 1

B. *Numerical Simulation of Algorithm 1*

Next, we adopt the same setting from the previous subsection and test the performance of Algorithm 1 while varying its parameters. We initially implemented and tested a non-distributed version of the algorithm. In the first set of experiments, we fixed the length of the representations n to 128, the number of differences in them to 5, and studied the effect of the number of iterations that Protocol 1 uses by varying $k \in \{7, 8, 9, 10\}$. As shown in Fig. 4, there is an expected increase in the average of detected differences (out of 5 possible).

In the second set of experiments, we have fixed k to 10, differences to 5, and studied the effect of increasing the length of the representations $n \in \{256, 512, 1024, 2048\}$. As shown in Fig. 5, the percentage of corrected errors does not seem to be affected by the length of the representation, which is consistent with our previous analysis.

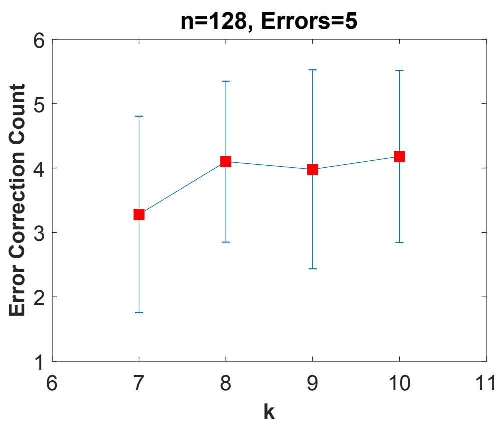


Fig. 4. Number of corrected errors with varying k.

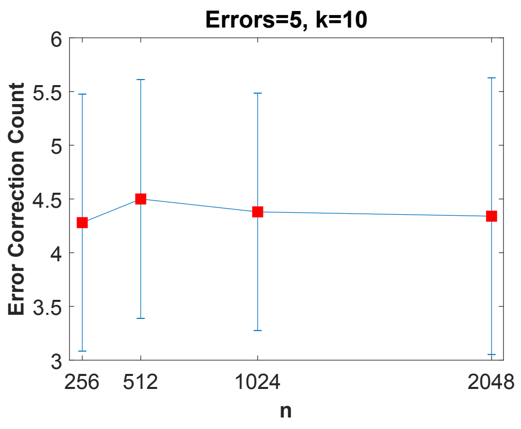


Fig. 5. Number of corrected errors with varying n.

In the third set of experiments, we have fixed k to 10, n to 1024, and studied the effect of varying the number of differences through $\{10, 20, 30, 40\}$. The result is presented in Fig. 6. It can be seen that the percentage of detection slightly decreases as the number of differences increases. However, the rate always remains over 80%.

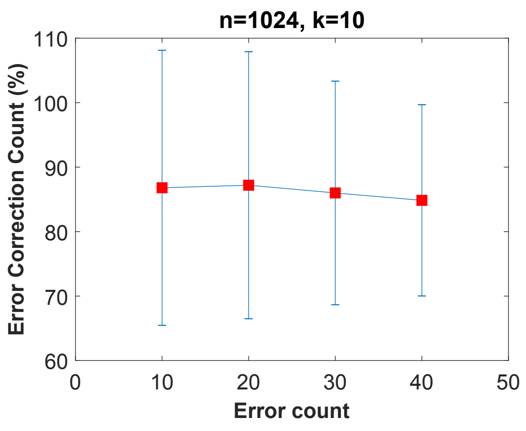


Fig. 6. Percentage of error corrected for varying degrees of errors in X and Y.

C. Case Study: Occupancy Grid Mapping

We are interested in testing our proposed lightweight communication mechanism for occupancy grid mapping using two robots as a proof-of-concept. We implemented this in a single computer. The robots starting from random locations do a random walk in an unknown environment (16×16, 4-connected square grid). We set $k = 10$. The robots can detect any neighboring obstacle when they step into a new cell. The environment is shown in Fig. 7. After making every ten moves, the robots communicate their current local perceptions of the world map represented as a binary string and apply our proposed methods. If their local representations of the environment are not the same, they update them using Algorithm 1. The program stops after 200 steps. In the environment presented in Fig. 7, there are 10 obstacles randomly generated in positions $(12, 15), (6, 13), (6, 9), (3, 15), (12, 3), (12, 13), (5, 5)$ $(13, 7), (14, 1), (6, 5)$ of the grid. Alice starts in position $(11, 7)$, and the initial position of Bob is $(3, 7)$. In step 20 of the simulation when comparing representations using Algorithm 1, it was found that their local representations differ. Alice has found an obstacle at $(6, 9)$, and then their maps are updated. In step 40, Algorithm 1 detects a difference again in their maps since Bob has discovered $(5, 5)$. There are also differences detected at steps 110 and 120. Both robots reliably detect their differences every time they discover an obstacle.

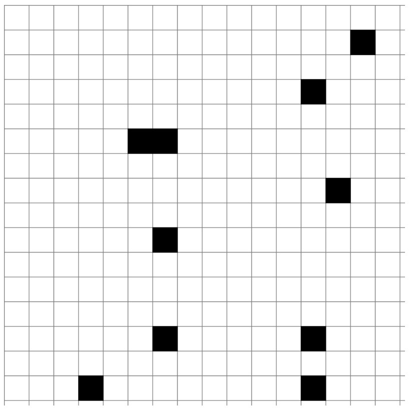
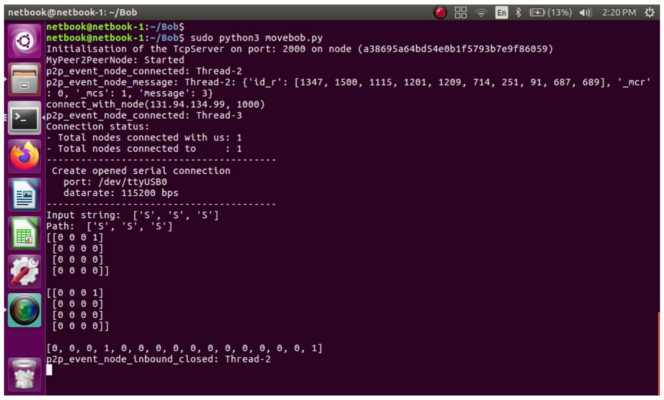


Fig. 7. Environment for the occupancy grid mapping test.

D. Physical Experiment

We have conducted physical experiments with two iRobot Create 2, each connected to an ASUS Eee laptop. Each of the laptops runs Ubuntu 16.04 LTS, with 1.66 GHz Intel Atom N280 CPUs and 1GB of memory. Processes and communications are handled using the Pycreate 2 library [29] and through a decentralized Peer-to-Peer TCP/IP protocol [26]. The robots are placed in a 4×4 grid-world containing two obstacles (Fig. 1). Maps are represented by 4×4 occupancy grid matrices, which are converted to vectors before applying Protocol 1. One of the robots, Bob, remains on stand-by and awaits a signal from the second robot Alice. Alice performs a short move sequence to discover obstacles in the grid-world

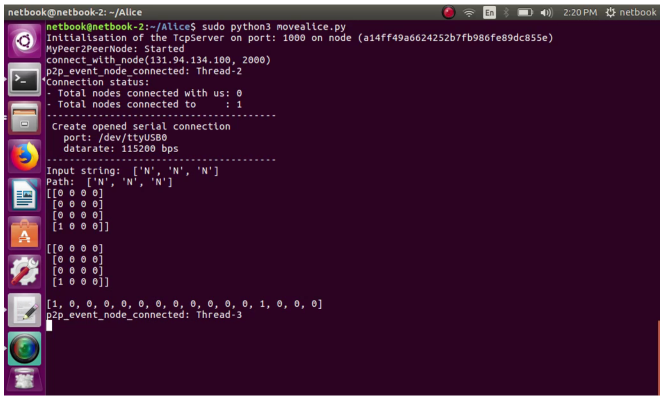


```
netbook@netbook-1: ~/Bob
netbook@netbook-1:~/Bob$
netbook@netbook-1:~/Bob$ sudo python3 movebob.py
Initialisation of the TcpServer on port: 2000 on node (a38695a64bd54e0b1f5793b7e9f86059)
MyPeer2PeerNode: Started
p2p_event_node_connected: Thread-2
p2p_event_node_message: Thread-2: {'id_r': [1347, 1500, 1115, 1201, 1209, 714, 251, 91, 687, 689], '_mcr'
: 0, '_mcs': 1, 'message': 3}
connect_with_node(131.94.134.99, 1000)
p2p_event_node_connected: Thread-3
Connection status:
- Total nodes connected with us: 1
- Total nodes connected to     : 1
----------------------------------------
 Create opened serial connection
   port: /dev/ttyUSB0
   datarate: 115200 bps
----------------------------------------
Input string:  ['S', 'S', 'S']
Path:  ['S', 'S', 'S']
[[0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[[0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
p2p_event_node_inbound_closed: Thread-2
```

(a)



```
netbook@netbook-2: ~/Alice
netbook@netbook-2:~/Alice$ sudo python3 movealice.py
Initialisation of the TcpServer on port: 1000 on node (a14ff49a6624252b7fb986fe89dc855e)
MyPeer2PeerNode: Started
connect_with_node(131.94.134.100, 2000)
p2p_event_node_connected: Thread-2
Connection status:
- Total nodes connected with us: 0
- Total nodes connected to     : 1
----------------------------------------
 Create opened serial connection
   port: /dev/ttyUSB0
   datarate: 115200 bps
----------------------------------------
Input string:  ['N', 'N', 'N']
Path:  ['N', 'N', 'N']
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [1 0 0 0]]

[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [1 0 0 0]]

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
p2p_event_node_connected: Thread-3
```

(b)

Fig. 8. An experimental demonstration of our lightweight communication protocols with two parties Alice and Bob using two iRobot Create 2.0 platforms: Snapshots of (a) Alice's and (b) Bob's execution of the protocol, their paths, and representations are presented.

and, upon colliding with one, stops and sends her current perception to Bob. Bob then runs his discovery until he faces another obstacle, updates his map, and then compares it with Alice. Upon having both comparison copies, both Bob and Alice perform Protocol 1 to determine whether or not they have the same map. Finally, the robots are set to sleep. It took 1 minute to complete the experiment. However, most of the run time is consumed by the robot movements, and our proposed communication protocol was fast. The snapshots of the algorithms running in Alice and Bob's computers are presented in Fig. 8. A video of the physical experiment can be found at https://youtu.be/UiYuZ04t_2c.

VI. Conclusions and Future Work

In this paper, we have studied the problem of minimizing the number of bits transmitted to synchronize two distributed representations of two robots. We have proposed one protocol and one algorithm, implemented them, and tested them in simulation. We have also performed a proof-of-concept experiment to show the practical feasibility of our approach.

We believe that there are exciting avenues of research at the intersection of communication complexity and multi-robot systems, which we would like to explore in the future. In the short term, we want to expand the analysis of Algorithm 1 and its implementation. Although Algorithm 1 is an improvement in cases where n is large and the differences are comparatively small, a more detailed analysis, perhaps studying its average complexity, can give more accurate estimates of its performance. On the implementation side, we have employed a non-distributed testbed. The next step is to test a distributed implementation using the client-server idea proposed and perhaps changing it to an iterative version that will be more convenient in the distributed setting.

Another area of near-term interest is to compare and contrast our ideas with the document exchange problem [5]. Although our motivations are the same, the methodologies employed are different (Hamming distance-based similarity in document exchange and binary dot products in our case). Although we have implemented our protocol in both hardware and software, a more detailed comparison between both approaches is needed to identify scenarios in which one approach performs better than the other. A more detailed comparison with other approaches in communication-constrained SLAM [23], [15] and set synchronization [25], [18] can provide fruitful research directions.

We plan to move away from occupancy grids represented by binary strings to have richer encodings used in robotics applications such as images, geometric maps, and point clouds. Scaling issues will need to be considered, but our preliminary results are encouraging. We have initially studied in detail the two-robot synchronization problems. A natural extension is to consider scenarios where multiple robots simultaneously want to synchronize representations as closely as possible. In these setups, we will take into account the positioning of the robots since communication costs will increase, for example, with distance and presence of obstacles. Interesting scenarios would involve peer-to-peer architectures where all the robots are equal participants and then hierarchical architectures, where some robots have greater computational and communication capabilities.

Acknowledgements

This work is supported in part by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-062000002. Murtadha Alsayegh is supported by the Saudi Arabian Cultural Mission under scholarship number KSP 11822-38375072. Peter Vanegas is supported by the FIU McNair program.

References

[1] T. Arai, E. Pagello, L. E. Parker, et al. Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.

[2] S. Arora and B. Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[3] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397, 2006.

[4] S. Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316, 2008.

[5] G. Cormode, M. Paterson, S. C. Sahinalp, U. Vishkin, et al. Communication complexity of document exchange. In *SODA*, pages 197–206, 2000.

[6] A. Dutta, A. Ghosh, and O. P. Kreidl. Multi-robot informative path planning with continuous connectivity constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3245–3251. IEEE, 2019.

[7] A. Dutta, A. Ghosh, S. Sisley, and O. P. Kreidl. Efficient communication in large multi-robot networks. In *2020 International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[8] B. P. Gerkey and M. J. Matarić. Principled communication for dynamic multi-robot task allocation. In *Experimental Robotics VII*, pages 353–362. Springer, 2001.

[9] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus. Guaranteeing spoof-resilient multi-robot networks. *Autonomous Robots*, 41(6):1383–1400, 2017.

[10] A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. *IEEE Journal on selected areas in communications*, 23(4):755–764, 2005.

[11] E. Klavins. Communication complexity of multi-robot systems. In *Algorithmic Foundations of Robotics V*, pages 275–291. Springer, 2004.

[12] E. Kushelvitz. Privacy and communication complexity. *SIAM Journal on Discrete Mathematics*, 5(2):273–284, 1992.

[13] E. Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

[14] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1996.

[15] M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos, and G. Grisetti. Multi-robot slam using condensed measurements. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1076. IEEE, 2013.

[16] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell. Coordinated multi-robot planning while preserving individual privacy. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2188–2194. IEEE, 2019.

[17] R. J. Marcotte, X. Wang, D. Mehta, and E. Olson. Optimizing multi-robot communication under bandwidth constraints. *Autonomous Robots*, 44(1):43–55, 2020.

[18] M. Mitzenmacher and R. Pagh. Simple multi-party set reconciliation. *Distributed Computing*, 31(6):441–453, 2018.

[19] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[20] N. Nisan and I. Segal. The communication complexity of efficient allocation problems. *Draft. Second version March 5th*, pages 173–182, 2002.

[21] J. M. O'Kane and D. A. Shell. Automatic design of discreet discrete filters. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 353–360, 2015.

[22] C. H. Papadimitriou and M. Sipser. Communication complexity. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 196–200. ACM, 1982.

[23] L. Paull, G. Huang, M. Seto, and J. J. Leonard. Communication-constrained multi-auv cooperative slam. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2015.

[24] A. Prorok and V. Kumar. A macroscopic privacy model for heterogeneous robot swarms. In *International Conference on Swarm Intelligence*, pages 15–27. Springer, 2016.

[25] V. Skachek and M. G. Rabbat. Subspace synchronization: a network-coding approach to object reconciliation. In *2014 IEEE International Symposium on Information Theory*, pages 2301–2305. IEEE, 2014.

[26] M. Snoeren. python-p2p-network. `https://github.com/macsnoeren/python-p2p-network`, Sept 2018.

[27] S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

[28] J. N. Tsitsiklis and G. D. Stamoulis. On the average communication complexity of asynchronous distributed algorithms. *Journal of the ACM (JACM)*, 42(2):382–400, 1995.

[29] K. Walchko. iRobot Create 2. `https://github.com/MomsFriendlyRobotCompany/pycreate2`, May 2017.

[30] Y.-C. Wu, V. Raman, S. Lafortune, and S. A. Seshia. Obfuscator synthesis for privacy and utility. In *NASA Formal Methods Symposium*, pages 133–149. Springer, 2016.

[31] A. C.-C. Yao. Some complexity questions related to distributive computing(preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, page 209–213, New York, NY, USA, 1979. Association for Computing Machinery.

[32] Y. Zhang and D. A. Shell. Complete characterization of a class of privacy-preserving tracking problems. *The International Journal of Robotics Research*, 2018.