Abstract—This paper presents a novel robot-environment interaction in navigation tasks such that robots have neither a representation of their working space nor planning function, instead, an active environment takes charge of these aspects. This is realized by spatially deploying computing units, called cells, and making cells manage traffic in their respective physical region. Different from stigmegic approaches, cells interact with each other to manage environmental information and to construct instructions on how robots move.

As a proof-of-concept, we present an architecture called AFADA and its prototype, consisting of modular cells and robots moving on the cells. The instructions from cells are based on a distributed routing algorithm and a reservation protocol. We demonstrate that AFADA enables a robot to move efficiently in a dynamic environment that stochastic changes its topology, comparing to self-navigation by a robot itself. This is followed by several demos, including multi-robot navigation, highlighting the power of offloading both representation and planning from robots to the environment. We expect that the concept of AFADA contributes to developing the infrastructure for multiple robots because it can engage online and lifelong planning and execution.

I. INTRODUCTION

Representation, planning, execution, and their smooth integration are essential factors for developing intelligent systems. In particular, representation, i.e., how to abstract and model the world, has been a central issue for AI and robotics [1]; however, there is room to consider whether robots themselves should have a representation of their working environment. As famously argued by Brooks [2], another view is to “use the world as its own model”.

Navigation: We can see derivative concepts, direct use of the world as the representation, in navigation tasks. Navigation is a fundamental robotics challenge that enables autonomous robots to reach their destinations. In general, robots use internal maps as a representation of the environment; however, this entails some serious difficulties, still making navigation challenging [3], e.g., accurate and robust self-localization, working in a dynamic environment where maps are updated frequently, the necessity of a map whenever robots enter a new workspace, and lack of consistency between the internal maps of different robots in multi-robot scenarios. Such problems mostly stem from discrepancies between external physical objects and internal representation.

Passive environment: The use of the environment itself as the representation can be a powerful tool for navigation. This is realized by spatially deploying sensors or tags. In such workspaces, robots without explicit internal maps can navigate individually. For instance, in the early development of self-driving cars, they were guided by electronic devices embedded in roadways [4]. Sensor networks can help navigation, e.g., Verma et al. [5] used a static sensor network to guide a robot in an unknown environment. Signals from RFID tags surrounding the environment have been used to navigate a robot [6], [7], [8]. In nature, stigmergy [9], a mechanism of indirect coordination through the environment between agents, achieves unexpected collective behaviors akin to coordination of social insects [10]. Stigmergic approaches in multi-robot systems have been popular, e.g., navigating robots by artificial pheromones dropped into the environment, taking the form of chemical compounds, embedded RFID tags, etc [11], [12], [3]. These studies can also be seen as examples of spatial computing [13].

Active environment: This paper aims at offloading not only the representation that robots have of the environment, but also the planning function, aiming at a smooth integration of representation, planning, and execution. In navigation tasks, the offloading is embodied as follows. A robot has a destination but does not know the environment or its own position. Instead, the environment actively and repeatedly issues instructions, i.e., planning, instructing the robot where to move. This is realized by deploying static computing units (neither simple sensors nor tags) and by coordinating between these units. The active environment is different from just setting cameras on the ceiling then monitoring robots [14], [15] in the sense that the deployed units themselves form an environment like stigmergy.

Rationale: The rationale of this concept addresses the following three aspects: 1) functional separation: Robots are relieved from the burden of collecting relevant information and planning their trajectories, letting them focus on other tasks. This also implies that even robots lacking expensive sensors can navigate the environment. 2) response to dynamic environment: The system allows for faster response to
multi-robot navigation, highlight the benefits of offloading. That stochastic changes its topology. Several demos, including
ables a robot to move efficiently in a dynamic environment
nor planning. We empirically demonstrate that AFADA en-
from the cells, i.e., robots necessitate neither representation
coordination between cells. Robots just follow instructions
In multi-robot scenarios, collision avoidance is achieved by
spite the addition, removal, or even after the failure of cells.
locations of robots or routing information for navigation, de-
collectively manage the environmental information such as
and a robot on the cell. Using local communication, cells
and maintained collectively in cells, in the form of
routing
packet routing
A.
representation
and/or the set of robots
priori.
Furthermore, we assume that $G$ changes dynamically,
in response to the addition, removal, or crash of cells. The
set of robots is also dynamic, in the sense that, robots can
enter or leave the system.
Navigation: Each robot is assigned a list of destination
cells. The navigation problem consists in making robots visit
all destinations in their list. The problem has many variants,
such as, whether it requires to visit the destinations in the
given sequence, or whether or not all goals must initially
exist in $G$.
Note that, because it is common to use a discrete rep-
resentation of the environment in conventional navigation
problems, the problem defined here is adaptable to a wide
range of problems by simply regarding a cell as a vertex.

III. ELEMENTS OF AFADA

This section presents two key logical aspects of AFADA:
representation and its maintenance, and integration of plan-
ning and execution.

A. Representation

In AFADA, the representation of the environment is stored
and maintained collectively in cells, in the form of routing
tables maintained at each cell. Similar to packet routing
in communication networks, each intermediate cell guides
the robot to the neighbor cell that will bring it closer to
its final destination. Routing tables maintain the information
necessary to make these decisions.
At each cell, the routing information is constructed and
maintained by communicating only with adjacent cells. Since
the environment is initially unknown and can also change dy-
amically, it is essential to properly maintain this information
throughout the lifetime of the system.
Centralized routing algorithms are obviously inadequate, so most known routing algorithms are inherently distributed, such as distributed dynamic programming [25], [26], or NetChange [27], [28] to name just a few.

In this context, the notion of self-stabilization [29], [30] is particularly attractive due to its inherent robustness. Self-stabilizing algorithms are designed in a way that, starting from any possible global state (valid or invalid), the system is always guaranteed to reach a global valid state (or a cycle of valid states) after a finite number of transitions and then permanently remain in valid states in the absence of external influence (e.g., failures, state corruptions, topology changes).

In self-stabilizing routing [28], [24], a valid global state is one in which the combination of all routing tables defines a path from any cell to any other cell. Such algorithms are attractive because of their inherent adaptivity and robustness. Since AFADA is fully decentralized and entails many sources of uncertainty (e.g., cell addition/removal, message loss, crashes), its routing algorithm (Algorithm 1) is designed to be self-stabilizing. The algorithm is inspired from classical self-stabilizing routing [28], [24] but additionally copes with the addition/removal of cells at runtime.

Algorithm 1 Update routing table (for cell $i$)

- $dist_i$: the estimated distance table
- $next_i$: the next-cell table

1. (repeat followings periodically)
2. broadcast $dist_i$ to $i$'s neighbors
3. clear $dist_i$, $next_i$, then $dist_i[i] ← 0$, $next_i[i] ← i$
4. for each neighbor $j$ and each key $k$ in $dist_j$ do
5. if $dist_j[k] ≥ D$ then continue
6. if $k ∉ dist_i$, keys or $dist_j[k] + 1 < dist_i[k]$ then
7. $dist_i[k] ← dist_j[k] + 1$, $next_i[k] ← j$
8. end if
9. end for

In Algorithm 1, each cell $i$ maintains two tables: $dist_i$ estimates the distance to other cells, and $next_i$ determines the next neighbor cell on the path to other cells. Cell $i$ periodically broadcasts $dist_i$ to its neighbors [line 2] (with 2 s period in our prototype), then updates both tables according to the minimal estimated distance of neighbor cells to target cells [lines 3–9].

If $G$ remains unchanged for long enough, and cells $i$ and $j$ are connected in $G$, then $dist_i[j]$ eventually holds the length of the shortest path connecting $i$ and $j$. Furthermore, for every cells $i, j, k$, if $i ≠ j$ and $k = next_i[j]$ then $dist_k[j] = dist_i[j] − 1$. This implies that the routing tables correctly guide the robots. The formal model and the proofs are beyond the scope of this paper.

B. Integrated Planning and Execution

Planning must ensure that the movements of robots are collision-free, for which routing tables alone are insufficient. To this end, it is important to also consider how the robots interact with the cells.

We present the basic behavior of robots and cells and their interaction, based on the time-independent model [31] that copes with multiple agents on a graph moving towards their destinations without any timing assumptions. The model is event-based in the sense that any change in the environment (e.g., start/end of movement, cell addition/removal, cell or robot failure, variable change) defines a new configuration (or global state). In a distributed environment such as AFADA, the notion of “simultaneity” is difficult to realize [32]. Robots should not rely on timings, rather should rely on events such as receiving messages or arriving at a new cell, and this justifies the use of the model.

Typically, the system repeats the following steps. Before a robot can move, the cell requests and reserves the next cell on the path to the destination on behalf of the robot. The robot moves upon receiving confirmation from the cell, which is released by the next cell upon arrival of the robot. The reservation prevents collisions with other robots.

More precisely, assume that a robot $r$ is on a cell $i$ and its destination is $g$. The procedure is as follows:

1. $r$ requests instructions from $i$ on how to go to $g$.
2. After receiving the request, $i$ enquires a neighbor cell $j$ about its availability (e.g., occupied, unoccupied).
3. $j$ replies its availability to $i$, according to whether another robot occupies $j$. If available, $j$ becomes reserved by $r$.
4. If $j$ was unoccupied, continue from Step 6 where $i$ instructs $r$ to move to $j$.
5. Otherwise, go back to Step 2 with another $j$ or with the same $j$ but after waiting for some arbitrary time.
6. When $r$ receives the instruction from $i$, it starts moving towards $j$.
7. When $r$ arrives at $j$, it notifies $j$ which sends a release message to $i$ which is thus released.
8. Repeat from Step 1 with $j$ as the new $i$.

In Step 2, $j$ is typically selected based on the routing table $next_i[g]$. Alternatives include selecting $j$ randomly, in an attempt to break cycle of requests and thus potential deadlocks.

Variants can be considered, such as the multi-step reservation seen in the demo section (§VI-B.1). A comparison with other protocols is outside the scope of this paper.

IV. HARDWARE PROTOTYPE

This section describes the hardware implementation of robots (§IV-A) and cells (§IV-B). The architecture is shown in Fig. 2. The robots use line tracing to move from a cell to the next. The cells communicate with each other via a serial interface and with the robots via NFC (Near Field Communication) to avoid radio interference. Cells can dynamically be attached or detached thanks to magnetic connectors. Connections between cells also supply power.

A. Robot

1) Requirements: Robots need to identify an area of each cell to move from one cell to another cell correctly. Robots need to communicate with underlying cells.
2) **Structure**: A robot consists of a Zumo 32U4 robot (Zumo) as a base with an interface board (Robot-Main-Board) holding an ESP32 microcontroller (ESP32-Board), an NFC controller (PN7150 chip), and LEDs. An NFC antenna is placed under the robot. Each cell measures $100 \times 100 \times 42$ mm$^3$ and weighs about 290 g.

Functionally, the ESP32 works as the main controlling unit and runs FreeRTOS to support multi-tasking. The robot’s motion is controlled by the ATmega microcontroller located on the Zumo. Upon receiving instructions from the ESP32, the Zumo uses the line sensor array to follow patterns drawn on the cells and guide the robot to an adjacent cell.

### B. Cell

1) **Requirement**: A cell needs to communicate with robots as well as adjacent cells, including a connection detection protocol to judge whether its neighbor exists or not. It has sufficient space to hold one robot. It is also critical to establish the power supply serving numerous cells while keeping the platform safe.

2) **Structure**: A cell consists of several printed circuit boards (Cell-Main-Board, LED-NFC-Antenna, Connection-Board), magnetic connectors, plastic parts (ABS Case), and an acrylic roof patterned for line tracing. The mainboard (Cell-Main-Board) is equipped with the same ESP32 microcontroller as the robot, an NFC controller, and a serial port expander supporting four serial channels. Magnetic connectors, which are installed on each side of the cell and connected to the mainboard via cables, are used for both communication between cells and shared power supply. The cell has a size of $160 \times 160 \times 41.5$ mm$^3$ and weighs about 440 g. The average power consumption of a cell is about 1.25 W. We now describe the functional aspects of cells.

  a) **Cell-Cell Communication**: In order to send a message from one cell to another, first, the ESP32 microcontroller sends a message to a PIC (Peripheral Interface Controller; PIC24FJ64GA306-I/PT) in the cell, through SPI (Serial Peripheral Interface). Then, the PIC relays the message to the adjacent cell via UART (Universal Asynchronous Receiver/Transmitter) communication. Both UART and SPI are bidirectional serial interfaces.

  b) **Connection Detector**: Cells must detect the addition or removal of other cells at runtime to update the environmental information. To monitor connections, the prototype uses two-stage detection: physical and virtual. Physical detection uses an adapted version of UART from that of USB (Universal Serial Bus), detecting the connection by the voltage switching on the signal line as a trigger [33]. Virtual detection uses heartbeats [34]. When two cells are actually connected, they exchange heartbeats periodically (period of $3$ s in the demo). When the cell fails to receive heartbeats for a certain time (timeout $10$ s in the demo) from some channel, the cell regards this channel as disconnected.

  c) **Power**: There are two ways to power a cell: a) connecting the cell directly to a power adapter (supply voltage: 24 V), or, b) connecting the cell to a powered cell. The system can be powered from several power adapters (e.g., 6 adapters for 100 cells). The 24 V line is connected to an onboard DC/DC converter, which provides a 5 V line used to supply power to the components in the cell. For safety, the prototype includes a switching circuit with relays so that the 24 V line, dangerous to humans, is not live unless an adjacent cell is connected.

### V. Evaluation

To evaluate the idea of offloading representation and planning, we first consider a single robot navigating in a dynamic environment. As baseline, we used a robot navigating by itself while collecting information about the environment.

#### A. Task

The task consists of multiple round-trips between two cells. To emulate a dynamic environment, several cells stochastically switch their status between correct and failed. Failed cells cease to communicate and robots cannot pass through them. Surrounding cells of the failed cell recognize a disconnection to the cell, resulting in an update of the routing tables. Every second, correct cells change their status to failed with a probability $p$ if no robot is present. Failed cells change their status to correct with a recovery probability $q$. The metric counts the total number of steps taken by the
robot (a step is a movement from a cell to a neighbor). Fewer steps is better and means less overhead.

Three carefully designed fields (simple-loop, two-bridge, and two-loop) were used, shown in Fig. 3, which includes annotations of target cells and potentially failed cells. The numbers of round trips are three times both in simple-loop and in two-bridge, and two times in two-loop. Even though one cell crashes, all fields have detours for round trips, hence moves approaching failed cells result in unnecessary steps. In two-bridge and two-loop, target cells are potentially disconnected by two failed cells; the robot should not move during disconnection to avoid unnecessary steps. We fixed \( q = 0.01 \) but set \( q \) in two conditions, 0.01 or 0.05, then run the robot with 30 repetitions for each condition.

**B. Comparison**

As a baseline for comparison, we tested a robot navigating by itself according to its internal map, using the Zumo robot base. Assume that the robot does not know the availability of locations until it approaches. In this experiment, the robot can detect failed cells only when adjacent to them. The status of cells is simulated within the robot.

The robot first plans the path for the trip (outward or return) using the shortest path algorithm, then moves. When it meets a failed cell on its way, it replans the path around it, and resumes movement. Herein, we call this style self-nav. Note that self-nav never interacts with active cells.

**C. Results**

The results are summarized in Fig. 3. In general, AFADA achieved efficient robot moves compared to self-nav when the recovery probability \( q \) was small, i.e., failed cells existed frequently. We describe the analysis as follows. The statistical significance threshold was set to 0.05.

1) simple-loop: AFADA failed to complete the task four times when \( q = 0.05 \) and three times when \( q = 0.01 \) due to message loss of the reservation protocol between cells. When \( q = 0.05 \), median scores were both 26 in AFADA and self-nav; the distributions in the two groups did not differ significantly (Mann–Whitney \( U = 362.5 \), \( n_1 = 26 \), \( n_2 = 30 \), \( P = 0.24 \), two-tailed). When \( q = 0.01 \), median scores were 29 in AFADA and 30 in self-nav; we observed significant difference (\( U = 283.0 \), \( n_1 = 27 \), \( n_2 = 30 \), \( P = 0.03 \)).

2) two-bridge: AFADA failed several times. When \( q = 0.05 \), median scores were 36 in AFADA and 38 in self-nav; the distributions did not differ significantly (\( U = 298.0 \), \( n_1 = 25 \), \( n_2 = 30 \), \( P = 0.08 \)). When \( q = 0.01 \), median scores were 42 in AFADA and 45 in self-nav; there was a significant difference (\( U = 258.5 \), \( n_1 = 27 \), \( n_2 = 30 \), \( P < 0.01 \)).

3) two-loop: In this case, AFADA failed in almost half of the trials; 12 times when \( q = 0.05 \) and 14 times when \( q = 0.01 \). When \( q = 0.05 \), median scores were both 34 in AFADA and in self-nav; the distributions did not differ significantly (\( U = 269.5 \), \( n_1 = 18 \), \( n_2 = 30 \), \( P = 0.5 \)). When \( q = 0.01 \), median scores were 44 in AFADA and 50 in self-nav; we observed significant difference (\( U = 135.0 \), \( n_1 = 16 \), \( n_2 = 30 \), \( P < 0.01 \)).

**D. Discussion**

AFADA can achieve efficient robot moves because the environment “knows” which cells are available before the robot moves. Note that the update of the graph topology stored in cells also requires time; due to this, we observed that AFADA did not always result in optimal moves. AFADA failed sometimes due to message loss. Our prototype allows operations of cell addition or removal physically; as a disadvantage, the connection might be faulty. Since the protocols, except for routing, are neither fault-tolerant nor self-stabilizing, such unexpected events affect the system critically. As seen in two-loop, when the number of cells increases, this problem cannot be ignored. Solutions include developing robust message channels or self-stabilizing planning and execution protocols, which we plan for future work.

**VI. Case Studies**

This section presents demos showing the potential of AFADA. A supplementary movie\(^2\) presents them as well as other demos.

**A. Navigation in Reconfigurable Environment**

As illustrated in Fig. 4, cells can guide robots correctly in spite of environmental changes due to adding/removing cells. A task is a round trip between two cells \( s \) and \( g \). Cells are added or removed during the execution. At first,

\( ^2 \)Available at https://dfg-lab.github.io/afada/
$g$ is unreachable and the cell informs the robot to wait in place. The robot starts moving after the cells detect the connection of $g$. Next, we added a new cell to create a new path, while removing a cell previously used by the robot. The environment updates the routing tables, and successfully guides the robot back to $s$ via the new path. This illustrates the self-reconfiguration of the system in the face of physical changes in the environment.

B. Multi-robot Navigation

In multi-robot scenarios, colliding is fatal; AFADA provides collision-free moves of robots.

1) Multi-robot Path Planning and Execution: Initially, all robots have distinct locations and destinations. Fig. 5 shows a configuration used in the demo. A solution is to make robots move to their destinations. In addition to the single-step reservation protocol in Section III-B, this demo uses a multi-step reservation; let cells reserve multiple cells ahead before a robot moves. When vacant cells receive a request from neighbors, they forward the request to the next adjacent cells. This forwarding of the request continues until the request is rejected by an occupied cell. Upon receiving a rejection, these cells then relay acknowledgments in the reverse order. As a result, a robot reserves several cells ahead and releases them one-by-one as it moves. Both single-step and multi-step reservation styles used a random choice of cells when receiving a rejection of the request to break deadlocks.

2) Automated Parking: As an application, we emulate an automated parking system (Fig. 5). The parking lot is modeled as a $4 \times 4$ grid. Cells in the leftmost/rightmost columns are parking space, and the two center columns are aisle. Robots can enter or leave the parking lot from the bottom row of the environment. These cells and the aisle are programmed to guide robots in one-way traffic using the reservation protocol to avoid collisions. The robot greedily looks for a vacant parking space following the one-way traffic. If the robot successfully enters a vacant parking space, it waits for a while (corresponding to the driver running an errand). Then, the robot leaves the parking lot, heads to the exit, and leaves the environment.

VII. DISCUSSION AND CONCLUSION

We presented AFADA, a novel fully decentralized scheme with robot-environment interactions for robot navigation, with its implementation as proof-of-concept. The key concept consists in offloading both representation and planning from robots to the environment with computing units embedded spatially in the form of active cells. Robots do not plan their moves alone, rather, cells communicate with neighboring cells and collectively take charge of robots’ movements. In the experiment, we demonstrated that AFADA achieved efficient robot moves using the navigation task in the dynamic environment, followed by several demos such as path planning and execution for multiple robots. We envision that the concept can be applied to many domains of multi-robot systems as infrastructure, such as automated warehouses, smart parking, and the cooperative behavior of autonomous cars.

Quite a few challenges remain to apply the concept to realistic scenarios, e.g., introducing an active environment on a large scale is costly. One important issue is that the system is inherently distributed and fully decentralized. This implies that we cannot expect a perfect execution devoid of any unexpected events such as failure, particularly when the system components increase. An example can be seen in navigation experiments of two-loop in Section V. This is where paradigms and principles of distributed computing are attractive, just like we used self-stabilizing algorithms.

Future work includes the following.

1) Develop cell-driven path planning algorithms. We only presented a basic protocol to prevent collisions between robots, however, it does not yet ensure desirable properties such that all robots eventually reach their destinations. Furthermore, there is considerable room for improvement in the efficiency of trajectories for multiple robots.

2) Address large robots. We currently limit the size of robots to be smaller than a cell for simplicity. Addressing large robots that occupy several cells simultaneously, is fruitful for coordination between heterogeneous robots. This is closely related to diagonal or any-angle moves of robots in a grid environment.

ACKNOWLEDGMENTS

This work was partly supported by the NTT FACILITIES Collaborative Research Project and by JSPS KAKENHI Grant Number 20K11685. The authors thank Mattias Evalds-son, Daniel Gstöhl, and Shoma Mori for their support in the early phases of development; as well as Kazuya Iimuro, Haruka Katahira, Wataru Kinota, and Xin Cen, for their help with building the numerous cells.
REFERENCES

[1] R. Davis, H. Shrobe, and P. Szolovits, “What is a knowledge representation?” AI magazine, vol. 14, no. 1, pp. 17–17, 1993.
[2] R. A. Brooks, “Intelligence without representation,” Artificial intelligence, vol. 47, no. 1-3, pp. 139–159, 1991.
[3] A. A. Khaliq and A. Saffiotti, “Stigmergy at work: Planning and navigation for a service robot on an rfid floor,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 1085–1092.
[4] K. Bimbraw, “Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology,” in 2015 12th international conference on informatics in control, automation and robotics (ICINCO). IEEE, vol. 1, 2015, pp. 191–198.
[5] A. Shakaff, L. Kamarudin, and F. Saad, “Mobile robot localization on Autonomic Communication paradigm for autonomic computing and communication,” in 2015 12th International Conference on Pervasive Computing and Communications. IEEE, 2005, pp. 41–50.
[6] W. Gueaieb and M. S. Miah, “An intelligent mobile robot navigation technique using rfid technology,” IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 9, pp. 1908–1917, 2008.
[7] M. Kim and N. Y. Chong, “Direction sensing rf reader for mobile robot navigation,” IEEE Transactions on Automation Science and Engineering, vol. 6, no. 1, pp. 44–54, 2008.
[8] S. Park and S. Hashimoto, “Autonomous mobile robot navigation using passive rf in indoor environment,” IEEE Transactions on industrial electronics, vol. 56, no. 7, pp. 2366–2373, 2009.
[9] G. Theraulaz and E. Bonabeau, “A brief history of stigmergy,” Artificial life, vol. 5, no. 2, pp. 97–116, 1999.
[10] M. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraulaz, Self-organization in biological systems. Princeton university press, 2003.
[11] R. Johansson and A. Saffiotti, “Navigating by stigmergy: A realization on an rfid floor for minimalistic robots,” in 2009 IEEE International Conference on Robotics and Automation. IEEE, 2009, pp. 245–252.
[12] R. Fujisawa, S. Dobata, K. Sugawara, and F. Matsuno, “Designing pheromone communication in swarm robotics; Group foraging behavior mediated by chemical substance,” Swarm Intelligence, vol. 8, no. 3, pp. 227–246, 2014.
[13] F. Zambonelli and M. Mamei, “Spatial computing: An emerging paradigm for autonomic computing and communication,” in Workshop on Autonomic computing and communication, 2004, pp. 44–57.
[14] R. Visvanathan, S. Mambuh, K. Kamarudin, A. Yeon, A. Zakaria, A. Shakaff, L. Kamarul, and F. Saad, “Mobile robot localization system using multiple ceiling mounted cameras,” in 2015 IEEE SENSORS. IEEE, 2015, pp. 1–4.
[15] A. Sinha, P. P. Mallya, N. Nath, et al., “An approach towards automated navigation of vehicles using overhead cameras,” in 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICICC). IEEE, 2017, pp. 1–8.
[16] A. Torrecho, E. Onaindia, A. Komenda, and M. Stoiba, “Cooperative multi-agent planning: A survey,” ACM Computing Surveys (CSUR), vol. 50, no. 6, pp. 1–32, 2017.
[17] K. Dresner and P. Stone, “A multiagent approach to autonomous intersection management,” Journal of artificial intelligence research, vol. 31, pp. 591–656, 2008.
[18] L. Chen and C. Englund, “Cooperative intersection management: A survey,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 570–586, 2015.
[19] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” AI magazine, vol. 29, no. 1, pp. 9–9, 2008.
[20] H. Ma, W. Höning, L. Cohen, T. Uras, H. Xu, T. S. Kamar, N. Ayaniyan, and S. Koenig, “Overview: A hierarchical framework for plan generation and execution in multirobot systems,” IEEE Intelligent Systems, vol. 32, no. 6, pp. 6–12, 2017.
[21] O. Salzman and R. Stern, “Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems,” in Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1711–1715.
[22] G. Valentini, A. Antoun, M. Trabattoni, B. Wiandt, Y. Tamura, E. Hochard, V. Trianni, and M. Dorigo, “Kilogram: a novel experimental environment for the kilobot robot,” Swarm Intelligence, vol. 12, no. 3, pp. 245–266, 2018.
[23] M. Rubenstein, A. Cornejo, and R. Nagpal, “Programmable self-assembly in a thousand-robot swarm,” Science, vol. 345, no. 6198, pp. 795–799, 2014.
[24] T. T. Johnson and S. Mitra, “Safe and stabilizing distributed multi-path cellular flows,” Theoretical Computer Science, vol. 579, pp. 9–32, 2015.
[25] D. Bertsekas, “Distributed dynamic programming,” IEEE transactions on Automatic Control, vol. 27, no. 3, pp. 610–616, 1982.
[26] Y. Shoham and K. Leyton-Brown, Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press, 2008.
[27] W. D. Tajibnapis, “A correctness proof of a topology information main-