Secure Encoded Instruction Graphs for End-to-End Data Validation in Autonomous Robots

Jorge Peña Queralta, *Graduate Student Member, IEEE*, Qingqing Li, *Graduate Student Member, IEEE*, Eduardo Castelló Ferrer, and Tomi Westerlund, *Senior Member, IEEE*

***Abstract*—As autonomous robots are becoming more widespread, more attention is being paid to the security of robotic operations. Autonomous robots can be seen as cyber–physical systems: they can operate in virtual, physical, and human realms. Therefore, securing the operations of autonomous robots requires not only securing their data (e.g., sensor inputs and mission instructions) but securing their interactions with their environment. There is currently a deficiency of methods that would allow robots to securely ensure their sensors and actuators are operating correctly without external feedback. This article introduces an encoding method and end-to-end validation framework for the missions of autonomous robots. In particular, we present a proof of concept of a map encoding method, which allows robots to navigate realistic environments and validate operational instructions with almost zero *a priori* knowledge. We demonstrate our framework using two different encoded maps in experiments with simulated and real robots. Our encoded maps have the same advantages as typical landmark-based navigation, but with the added benefit of cryptographic hashes that enable end-to-end information validation. Our method is applicable to any aspect of the robotic operation in which there is a predefined set of actions or instructions given to the robot.**

***Index Terms*—Autonomous robots, cryptography, cyber–physical security, robot security, robotic navigation, secure navigation.**

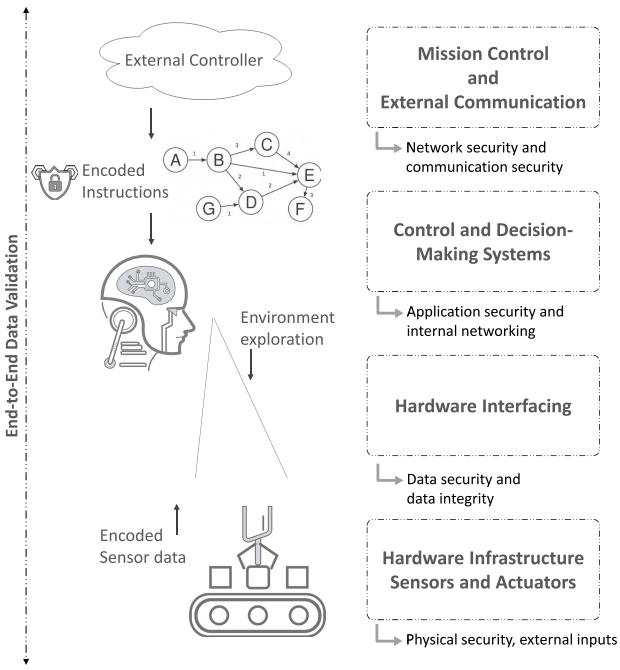


Fig. 1. Classification of data acquisition and analysis processes in autonomous robots and matching security layers.

I. Introduction

As ROBOTS become increasingly common across many domains and application areas, more attention is being paid to the safety and security aspects of robotic operation [1]. The differentiation between safety and security is often ambiguous, with the term *safety* typically being used to refer to human–robot interaction [2] or to the protection of the robot from physical damage [3]. However, what is typically overlooked is that the safe operation of an autonomous robot is inherently tied to having tight security of the data involved, including sensor data and data defining mission instructions. Fig. 1 shows a classification of stages in which information is either collected or processed by an autonomous robot. The classification in Fig. 1 extends the cyberattacks categorization defined in [4]. It also models the software processes as a network [5], which corresponds with many robotic frameworks, such as the robot operating system (ROS) [6]. In this classification in Fig. 1, the acquired sensor data need to be both secured and validated, which is essential from a cybersecurity point of view and represents an unresolved challenge. For the secure operation of autonomous robots, it is essential to be able to validate the data being shared among subsystems and external systems (e.g., a controller or other robots) and the data defining or characterizing the way the robot interacts with its environment.

A relevant precedent in securing multirobot cooperation was introduced by Ferrer *et al.* [7], in which the authors leveraged Merkle trees to improve the security and secrecy of swarm robotics missions. The main novelty of their work is the introduction of a framework for validating data in robots without

Manuscript received 28 May 2021; revised 5 December 2021 and 7 February 2022; accepted 20 March 2022. Date of publication 4 April 2022; date of current version 7 September 2022. This work was supported in part by the Academy of Finland's AutoSOS Project under Grant 328755 and RoboMesh Project under Grant 336061; in part by the European Union's Horizon 2020 Research and Innovation Programme through the Marie Skłodowska-Curie under Grant 751615; and in part by the Finnish Foundation for Technology Promotion under Grant 8089. *(Corresponding author: Jorge Peña Queralta.)*

Jorge Peña Queralta, Qingqing Li, and Tomi Westerlund are with the Turku Intelligent Embedded and Robotic Systems Laboratory, University of Turku, 20500 Turku, Finland (e-mail: jopequ@utu.fi; qingqli@utu.fi; tovewe@utu.fi).

Eduardo Castelló Ferrer is with the MIT Connection Science and MIT Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: ecstll@mit.edu).

Digital Object Identifier 10.1109/JIOT.2022.3164545

relying on the data itself, by encoding mission instructions in Merkle trees. Merkle trees are cryptographic structures that enable the validation of data through cryptographic proofs that do not involve the data itself.

We aim to extend the previous work to build a more general framework: rather than encoding one set of mission instructions [7], we encode all possible ways in which a mission can be completed. In [7], one of the main research questions is whether it is possible to provide the *blueprint* of a robotic mission without describing the mission itself. In this work, we build upon the separation of data verification from the data itself, as introduced in [7], and explore new implicit ways for defining complex robotic workflows. In [7], the robots perform predefined actions when they are able to reproduce some predefined encoded sensor data. In this work, we instead utilize a graph structure with connections between the possible mission instructions or states, in which paths through the graph encode all possible mission flows.

In this manuscript, we first describe the proposed framework and several possible applications, including human–robot interaction and collaborative robots. We then provide a proof of concept in an example navigation mission. The proof of concept demonstrates that our framework has minimal impact on robot behavior and robustness, even when the full mission is encoded. Our framework is therefore viable for real applications and opens the door to the more secure and safe deployment of autonomous robots.

We summarize the current research gap in robust data validation schemes for autonomous or semiautonomous robots with the following key unanswered research questions.

1) Is it possible for a robot to safely and securely interact with its environment, operators, and other robots without *a priori* knowledge of the mission?
2) Can encoded information that reveals no explicit mission instructions maintain the performance of unencoded robotic workflows?
3) Can encoded mission instructions be used to simultaneously validate the integrity of robot actions, the progress of the mission, the sensor data, and the operations of actuators?

The first question applies to a wide variety of situations. For instance, consider whether a robot in a factory can be given assembly instructions that it cannot understand until reaching the respective step in the assembly process, or whether a mobile robot can be given a map that it cannot understand until it is navigating the respective location in its environment. An even more complex situation occurs when a robot can interact with a human or another robot in a certain way only when a series of conditions are met. The latter two questions deal with practical considerations: 1) whether encoded instructions can be adapted to standard robotic algorithms and workflows and 2) whether robots can use encoded instructions to simultaneously validate all processes involved in a mission. In short, we are asking whether a single framework can provide end-to-end data validation and provide encoded instructions that enable a robot to securely and safely interact with its environment.

We consider the validation of a robot's operation from end to end: validating sensor data, the correct operation of actuators, mission instructions, and information received from an external controller. We conduct proof-of-concept experiments in a navigation mission in which limited *a priori* information about the environment is available to the mission controller or the human operator. In this scheme, an operator generates a set of encoded instructions by hashing the description of a set of landmarks in the environment that serve as navigation waypoints. The encoded set of landmarks is given to the autonomous robots, which use them to navigate a realistic environment without any other *a priori* knowledge. The encoded landmarks are nodes in a navigation map that is given to the robots in the form of a graph. The graph edges encode information about how to navigate between consecutive landmarks. Because all information is encoded, we minimize the amount of raw data exposed to the robots *a priori*.

The main contributions of this work are as follows.

1) The definition of an end-to-end validation framework for autonomous robots based on encoded instruction graphs.
2) The definition and validation of a novel approach to use cryptographic hashes to encode a navigation graph that maps environment features.
3) The definition and validation of methods that allow robots to follow encoded mission instructions while validating their sensor data without external feedback.

The remainder of this article is organized as follows. In Section II, we overview previous works that address security issues in robotic systems. Section III then introduces the proposed framework, describes different use cases, and discusses the potentials and limitations of our framework. Section IV presents the implementation details of a proof of concept for robotic navigation, using an encoded navigation graph. Then, Section V describes the methodology we have followed in our simulations and experiments and Section VI presents the experimental results. Finally, Section VII concludes the work and outlines future research directions.

II. Background

There is a growing interest in securing robotic systems, partly due to to the increasing connectivity with which robots are being equipped. There are many data exchange modalities with new attack vectors, for instance, in remote control commands [8], telemetry [9], offloading computation [10], robot-to-robot communication [7], [11], and human–robot interaction [12].

Existing research efforts that study cybersecurity issues in robotics generally consider exclusively virtual or data aspects. Clark *et al.* [4] reviewed and discussed the main security threats to robotic systems, including spoofing sensor data, denial-of-service attacks, malicious code injection, and signal interference. However, this review only considers the cybernetic point of view, not explicitly the physical dimensions in which robots operate. Some works have considered security in the context of sensor data and navigation missions. For instance, in an early work in this direction, Py and Ingrand [13] introduced an execution control framework for an autonomous robot to analyze the data it obtained through its behaviors using a *state checker*. In a more recent work, Tang *et al.* [14]

ensured the convergence of sensor data under the denial-of-service attacks. Similarly, Tiku and Pasricha [15] introduced a methodology for overcoming security vulnerabilities in a deep learning localization method by making use of adversarial training samples. In distributed and multirobot systems, most efforts focus on the analysis and mitigation of security issues from a networking perspective [16]. All of these approaches take the point of view of data security in information systems and do not explicitly involve the cyber–physical nature of autonomous robots and their interaction with their environment, which is the objective of this article. The interaction of a robot with its environment presents new crucial issues that cannot be addressed with existing risk mitigation techniques from the cybersecurity domain.

There have been a few studies on data validation for autonomous robots. Legashev *et al.* [17] defined a generic framework from a legislative point of view, for monitoring, certification, and validation of the operation of autonomous robots, using periodic telemetry data obtained from autonomous vehicles. The framework developed by Legashev *et al.* validates a robot's operation but not the data itself. Validation of the data in this work can only be done through statistical analysis and detection of statistical abnormalities. Data integrity was the subject of one study by Yousef's *et al.* [18] on cyber–physical threats to robotic platforms.

In general, we see a research gap in terms of addressing the physical dimension of security issues in robotic operation. This becomes even more evident when considering widely used robotic frameworks, such as ROS, which has become a standard across both industry and academia. Multiple researchers have studied the security flaws of ROS [19] and proposed approaches to address these issues [20]. Many of these are also being mitigated in the newest version, ROS 2 [21]. However, these efforts are again mostly directed at securing ROS as a distributed and networked system, not at securing robots' interactions with their environment. While it is highly important to provide security for data flow, it is also important to close the gap in securing and validating the way robots receive instructions and interact with their environment.

In this work, we focus on a framework for validating data integrity. Other types of cyberattacks, such as denial-of-service attacks—in which the communication channels are congested—are not considered. However, it is worth noting that our proposed approach can provide some benefits even in the types of cyberattacks we do not directly consider. For instance, while the communication channel used to transmit encoded commands to the robot might be known to an attacker, the sensor data or inputs triggering the different actions will still be unknown to an attacker, as they are unknown even to the robot itself. As another example, the channel utilized to trigger a robot's actions might be disguised within the encoded instruction graph sent to the robot before the mission starts. In these ways, our framework can provide partial mitigation for a broad range of possible cyberattacks.

III. Encoded Instruction Graph's Framework

This section presents the framework and how it can generally be adapted to different application scenarios, before focusing in the remainder of the manuscript on how it can be applied to a specific navigation use case.

A. Encoding Robotic Instructions

We extend the instruction encryption ideas from [7], in which mission instructions are given to a robot by encoding combinations of sensor inputs and robot actions. We do not explicitly consider multirobot cooperation at this point. We describe robust options for encrypting and decrypting mission instructions and evaluate the performance degradation that is inherent to the addition of data encryption to robotic operation.

In order to encrypt a robotic operation, the first step is to encode a set of actions and features from sensor data along with at least one more variable that enables a *hash search*. A hash search is a trial-and-error process in which a robot does not necessarily reproduce a specific hash but instead can try multiple hashes until finding a match. The additional variable that enables the hash search can, for instance, be a spatial or temporal component. The second step is to define a series of encoded actions and encoded states based on a combination of variables (e.g., position, time, sensor data, or other external inputs). By encoding both states and actions, we can wrap the set of encoded information into a graph structure. The encoded information in the edge of the graph gives the robot information about how to arrive to the respective state or process. We call this an encoded instructions graph. A sample encoded instructions graph is shown in Fig. 2. In this graph, the initial instruction is encoded into a hash H_1, which the robot is able to decode by combining a predefined action (e.g., movement in one specific direction) with predefined sensor data (e.g., visual or geometric features extracted from the environment). Most of the nodes in the graph represent states. In the example shown in Fig. 2, most of the nodes are defined *a priori* by the combination of a specific position with specific sensor data (associated to an environment feature, e.g., a predefined landmark).

The information that nodes can encode is not restrictive, as it can be any action or state. The edges, however, need to encode information that enables the robot to transit between the respective nodes. Therefore, an edge needs to encode one of the robot's possible actions or an external input, such as a message from a controller or another robot. An edge can also be empty, e.g., if a robot at the previous node already has enough information to decode the next node, without any intermediate step. In the example in Fig. 2, the first node triggers an action by the robot when its acquired sensor data reproduces the respective hash in the graph. The robot then proceeds to one of two different states in which it must to be able to reproduce both a specific position and a specific sensor reading. The encoded information in the second node (e.g., H_2) can be decoded after a certain period of time. The robot can be forced to go back to the previous state if the sensor data encoded in H_3 cannot be acquired in time, which can, for example, be used as a failsafe. In practice, the robot is not aware of the type of information encoded in each hash and must therefore perform a continuous trial-and-error process to reproduce the hashes in the graph, by any means for which it has been preprogrammed. In our experiments, we show that

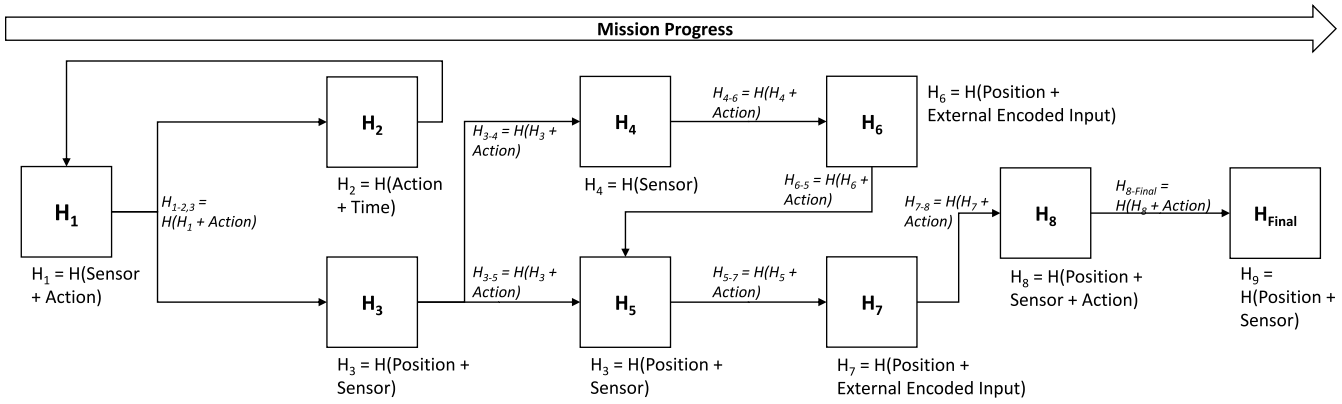


Fig. 2. Sample encoded instruction graph, with hashes that are defined from different environmental information, or combinations of sensor data, localization data, and one or more of the available actions. The mission can be completed successfully only if the hashes are decoded sequentially according to one of the paths encoded in the graph. Note that there is not necessarily a single way of progressing in the mission.

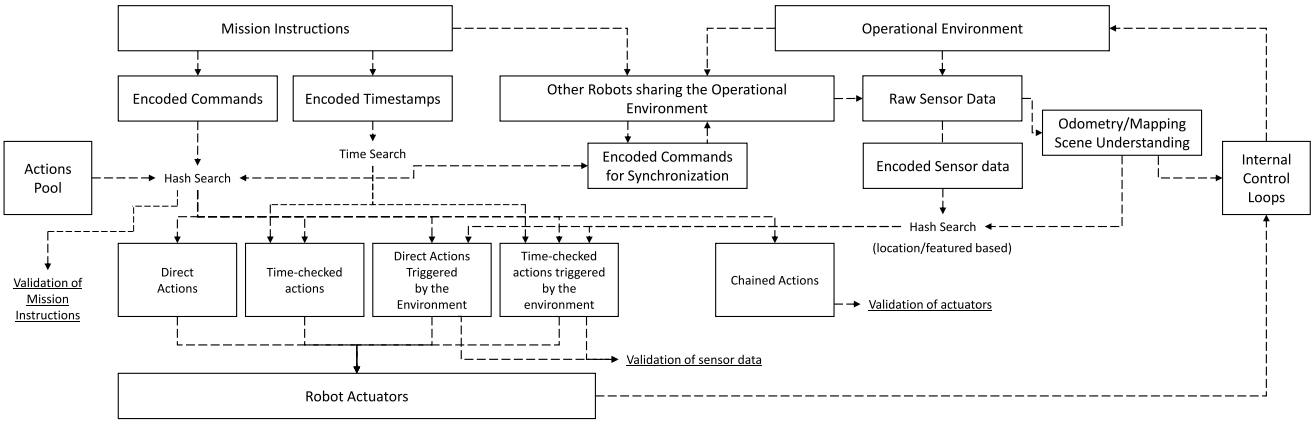


Fig. 3. Different validation modalities and data flows. The same approach can be used for individual mission instructions, event-based commands, chained instructions, or multirobot communication.

this trial-and-error process has a mostly negligible impact on robot behavior, compared to the computational cost of extracting features or processing sensor data. However, the process of deciding how to encode the instructions is not trivial, as they must be reproducible, but also must be concise enough to avoid data mismatches.

The graph depicted in Fig. 2 is directed. In, for example, a mission with only one possible solution in which each step is followed by only one other step, the graph simply represents a linear sequence. In more complex missions, the graph will often remain acyclic, with multiple possible paths to complete the mission, but without the possibility to repeat a step once it has been completed. For instance, in a manufacturing process, the order in which a set of parts are moved to a working bench might not matter, yet every part must be moved exactly once. The graph, nonetheless, can also contain cycles. For instance, in a reconnaissance mission in which a robot has to navigate an area but without a specific path, the graph would include an interconnected set of steps (i.e., a cycle). A robot's higher mission control should understand these situations and provide a planning strategy that is not directly sent by the mission controller (e.g., path planning on the encrypted graph). An example of this is shown in Section VI.

B. Validation Modalities

The proposed approach can be applied to many types of scenarios, because the encoded information in the graph is not necessarily limited to a set of predefined actions and features extracted from sensor data. Other external inputs can also be included, such as variables defining the state of the robot or timing constraints. The addition of these types of variables brings the possibility to enable secure and secret multirobot cooperation as well as new ways of defining the conditions under which human–robot interaction can happen. The node inputs are all encoded, so the information exchanged between robots or utilized as external signals triggering robot actions is only usable when combined with other data and is totally meaningless for an external agent. Therefore, if the data are spoofed or a third party gains access to this communication medium, no meaningful data are actually compromised.

Different possibilities for encoding and decoding instructions are shown in Fig. 3. In the figure, we show the different approaches to encoding a robot's interaction with its environment. For example, these options include simply encoding a predefined action (direct actions); combining actions with a timestamp indicating when they should be carried out (time-checked actions); combining actions with specific sensor data

obtained from the environment (direct actions triggered by the environment); or a combination of all the previous options. These actions can then be chained, such that the previous hash or action is included in the next codified node in the graph. These approaches are described in more detail and put into context in the following sections.

1) Independent Validation: The simplest approach is to encode mission instructions individually and independently. In this case, an encoded instruction set can be sent to the robot, similar to the approach followed in [7]. However, this set of instructions does not define a graph structure and does not represent the main interest of this article.

An additional layer of security can be added by introducing time or spatial constraints. Time constraints (e.g., hashing a timestamp together with the data of interest) provide an extra layer of security against attacks that could spoof the encoded data and reproduce it at a different time. Similarly, spatial constraints can be added by including the robot's location in the hash. This, however, only prevents the replication of the robot's behavior in other locations.

2) Iterative Validation: An iterative validation happens when a robot is able to validate its own actions. This modality is studied in the next sections with the introduction of an encoded navigation graph.

Encoded instruction graphs defining an iterative validation process can contain different types of encoded data in their nodes and edges. For instance, sensor data can be encoded in the graph nodes, which serve the purpose of process validation. Additionally, other information can be encoded, such as positional information or time information, which can be used in the control loops of the robot itself. Then, the actual instruction for the robot to move toward the next step is encoded in the edge of the graph, which encodes both the data in the current node being validated and the action or actions that will enable the robot to decode the next node. An illustration of this process is shown in Fig. 4. In the figure, we show how, at each step, instructions can be validated simply by being able to reproduce the corresponding hash. Moreover, as actions accumulate leading to new reproducible states (defined through chained hashes in the encoded instruction graph), we are able to iteratively validate the actions confirming their output with an expected outcome.

3) Multirobot Simultaneous and Mutual Validation: As mentioned earlier, one important scenario to which this validation framework can be applied is multirobot cooperative missions. Complementing the ideas proposed in [7], we are now also able to break down a mission into two parts that can be given to two different robots. An example of this is shown in Fig. 5, which illustrates a collaborative inspection process. In the example, encoded instructions are defined by combining a set of different signals or parameters. First, a set of sensor data is set to trigger an action from features extracted by the robot from. Now that multiple robots share the same environment, we can also differentiate between hashes obtained from sensing the environment and those obtained from sensing the behavior of other robots. Second, the robots can also exchange messages in order to trigger each other's actions. These messages do not hold any valuable data to the sender but only to the receiver as part of a hash decoding process. The messages can be predefined and based on the robot state, or generated as a function of the features sensed in the environment.

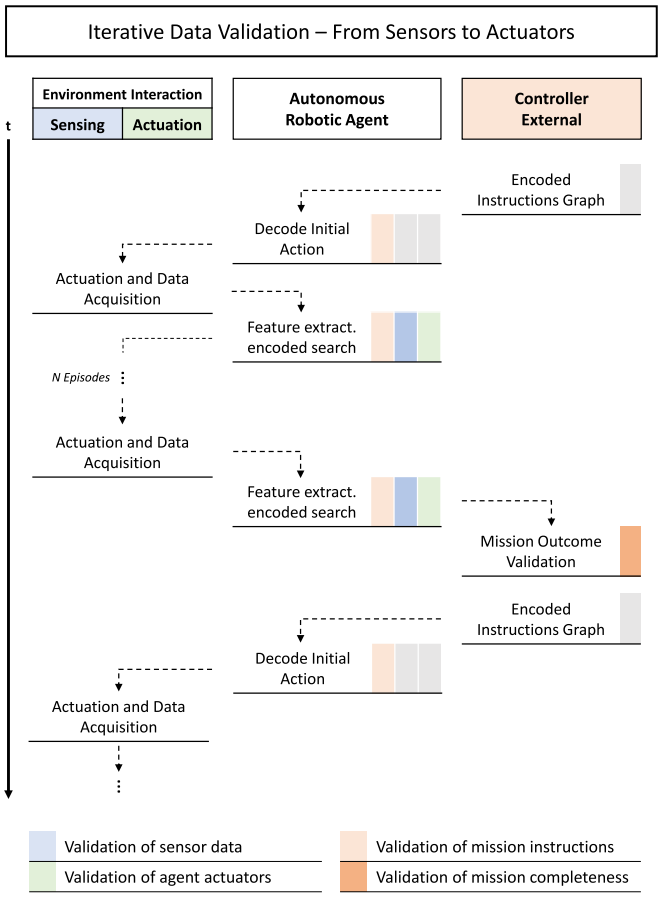


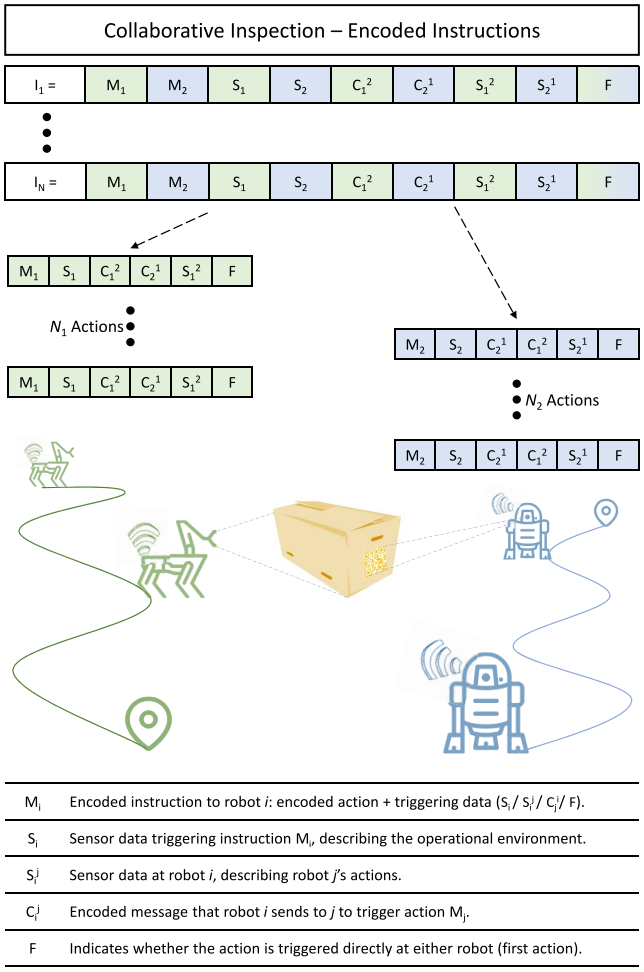
Fig. 4. Illustration of an iterative validation process.

IV. Use Case Encoded Navigation Graph

One of the most fundamental ways in which a robot interacts with its environment is through navigation. Maps have long been utilized for autonomous navigation and exploration in mobile robots to increase the robustness of the long-term autonomous operation [22], [23]. Maps and landmarks provide robot means for localization in a known reference frame, while enabling the calibration and adjustment of on-board odometry and localization algorithms.

Landmark-based navigation has been successfully implemented in various mobile robots with quick response (QR) codes [24]–[26] or other identifiable images [27]–[29], wireless sensor networks [30], ultrawideband (UWB) markers [31]–[33], IMU fusion [26], or topological maps for infrastructure-free navigation [34]. When utilizing landmarks that are already encoding certain information, such as QR codes or other text representations, additional information can be embedded into the landmarks. In an industrial scenario, this can be utilized to provide further instructions for robots [35].

In order to analyze the viability of the proposed research, we decided to investigate the problem of robot navigation since



M_i	Encoded instruction to robot *i*: encoded action + triggering data (S_i/ S_i^j / C_j^i/ F).
S_i	Sensor data triggering instruction M_i, describing the operational environment.
S_i^j	Sensor data at robot *i*, describing robot *j*'s actions.
C_i^j	Encoded message that robot *i* sends to *j* to trigger action M_j.
F	Indicates whether the action is triggered directly at either robot (first action).

Fig. 5. Illustration of a collaborative inspection process where robots only have partial instructions.

it is an easy way to make the robot interact with its environment. Along those lines, we focus the research questions to more concrete considerations regarding the navigation of autonomous robots.

1) Is it possible to provide a description of the environment (e.g., a map or a set of landmarks and how to travel between them) to an autonomous robot in such a way that the robot is unable to understand the map until it starts navigating, and such that it can only decode the information in that map if a series of conditions on how it sees its environment are met?
2) Is there a way of defining navigation instructions for an autonomous robot such that any modification of those instructions automatically renders them unusable ensuring that if wrong sensor data are fed to the robot's controller, the instructions cannot be followed?

A. Encoded Graph Definition

Rather than modeling a map of the exploration area and utilizing it for navigation, we utilize a landmark-based navigation graph that encodes the position of the different landmarks and the navigable directions between them. In this graph, each vertex represents one encoded position in the map, and each edge represents a straight or unique path between two positions. By unique we mean a path that might not be straight but such that the robot can realistically follow. A sample map and the corresponding encoded navigation graph are illustrated in Fig. 6. In this and the latter sections, we utilize the following notation: a graph is an ordered pair $\mathcal{G} = (V, E)$, where V represents a set of vertices, and E represents a set of edges associated with two distinct vertices, i.e., a set of tuples $\{(V_i, V_j)|V_i, V_j \in V\}$. We consider a directed graph, where the order of these tuples matters.

The most straightforward approach to landmark encoding is to define the hash of a position given its coordinates $\vec{r} \in \mathbb{R}^3$. Thus, we would define $H_i = H(\vec{r}_i)$. In order to ensure that hashes will be reproducible, the coordinates need to be given in a coarse grid with a resolution that is dependent on the accuracy of the robots' onboard odometry.

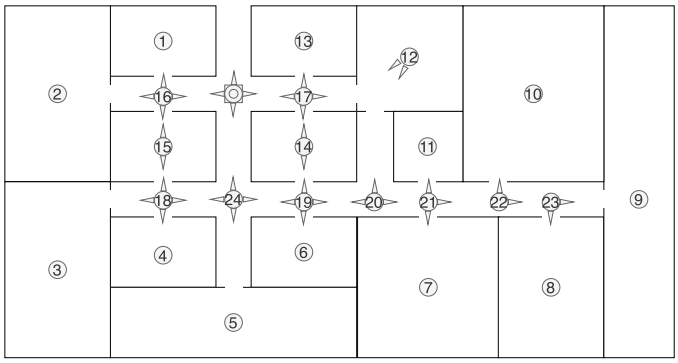
If the environment is accessible *a priori*, elements, such as QR codes or Bluetooth/UWB beacons can be placed to facilitate the localization of robots when they are nearby. The QR codes contain hashed data and can encode additional information, for example, instructions for a robot to operate in a given room or area. An alternative approach is to utilize the environment geometry and topology. The coordinates of the features can still be utilized to define their hash without using a predefined grid. Rather than having a robot utilizing its own or near position to calculate the hash, it can calculate it based on the coordinates of a position that depends on the robot's current local environment.

B. Deployment and Navigation

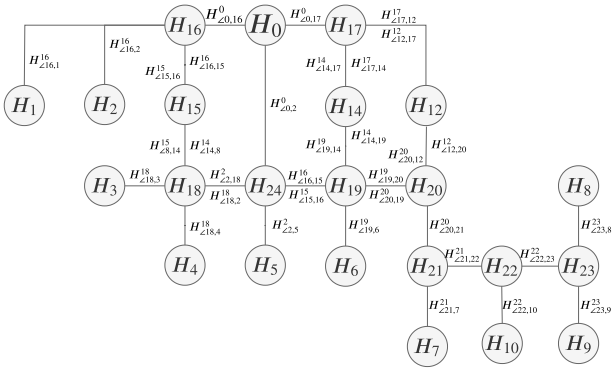
We assume that the initial location where a robot is deployed is either known in an absolute reference frame or utilized as a common reference in the robots' local coordinate system. If only local references are utilized, these must have a common orientation. This initial location is encoded with a hash but is also known to robots.

In the encoded navigation graph, each edge in the graph is given two hashes, as the robots might reach these from different directions. Therefore, the adjacency matrix containing the edge hashes shown in Fig. 6 is not symmetric. Only minimal information about the local environment required for navigation purposes needs to be stored at the robots. Odometry-only navigation, when possible, is preferred to map-based navigation to minimize the amount of raw information that robots store.

Global locations and relative positions between features as well as directions between them are encoded in a way that can be matched by robots on the basis of trying multiple possible directions until finding one that produces the corresponding hash. The edge hashes are calculated on a trial-and-error basis and, thus, they can be defined with an arbitrary division of the $[0, 2\pi]$ interval. However, the granularity of such division must consider the inherent computational overhead. Furthermore, not all the navigable directions are necessarily selected and, therefore, the real topology of the environment can be, to some extent, hidden. In addition, multiple features can be selected within a single room or small area, but even if



(a)



(b)

ID	H_N (256 bits)
0	H_0 = 5341dfa945ca9e52334 ... 8446048
1	H_0 = a6340c2ed22ff55a475 ... c3e92b4
⋮	
24	H_{24} = 2522cfaa21faaa45a ... d9de50a

(c)

$$\begin{pmatrix} 0 & 0 & 0 & \dots & 0 & H^0_{\angle 0,16} & H^0_{\angle 0,17} & \dots & H^0_{\angle 0,24} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ H^{16}_{\angle 16,0} & H^{16}_{\angle 16,1} & H^{16}_{\angle 16,2} & \dots & H^{15}_{\angle 16,15} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & 0 & \dots & H^{24}_{\angle 24,0} \end{pmatrix}$$

(d)

Fig. 6. Encoded navigation graph construction process. (a) Sample floorplan. (b) Corresponding navigation graph. (c) and (d) contain the information given to robots, a list of hashes, and an adjacency matrix with the edge hashes to aid navigation between landmarks, respectively. The landmark hashes are calculated based on the position (x, y, z) and the landmark type $(LT) : H_i = H(LT_i + x_i + y_i + z_i)$, where H is the hashing function and $+$ means concatenation. The edge hashes are $H^i_{\angle i,j} = H(LT_i + x_i + y_i + z_i + \angle i, j)$, where $\angle i, j$ represents the navigation direction from i to j.

all are detectable at the same time, a fully connected subgraph does not need to be generated within the navigation graph. In general terms, there is a tradeoff between the number of paths in the navigation graph between two features and the robustness of the navigation. In general, the more paths between two events, the higher the chance that the robot is able to reproduce a certain subset of hashes.

C. Landmark-Based Localization

The accuracy of the feature's position directly affects the error tolerance for the odometry method utilized when no landmarks are detected. To cope with this issue, the odometry error can be estimated and then taken into account to calculate the hashes from the position of landmarks. This is achieved by following a trial-and-error approach within a certain spatial area around the landmark. The number of trials that a robot needs to perform depends on the accuracy of the odometry method as well as the granularity of the grid utilized to define the position of the landmarks. Additionally, the possibility of the robot identifying a wrong landmark that is nearby must be taken into account.

V. Navigation Graphs: Methodology

To test the feasibility of the encoded navigation approach presented in this article, we conducted a series of simulations and real-robot experiments. In these, we analyzed the computational overhead as well as the performance impact of utilizing an encoded navigation graph. In all cases, we made the assumption that the environment is known to the mission controller. We devised two types of application scenarios in which we test the proposed framework.

First, we considered an environment where only robots are present. In this case, we simulated the interior of a building with empty rooms. For this environment, we encoded geometric features in the navigation graph, such as doorways, corners, and rooms. In our simulations, we considered an environment with no dynamic obstacles and a known geometry. This environment is comparable to modern logistic warehouses where only autonomous robots operate or to large industrial spaces where autonomous cleaning machines operate at night. This scenario represents well any environment that does not change significantly over time. In our simulations, the robot relies on a 2-D laser scanner for feature detection.

Second, we considered a real office scenario with a dynamically changing environment, people moving in it, and a wide variety of objects populating the different rooms. The experiments were carried out using visual markers that can be placed in multiple fixed locations. For this second scenario, real experiments were carried out in a real office environment with people and a variety of furniture across multiple rooms. Because of the large amount of desks, chairs, and other equipment, detecting geometric features from the environment would create multiple situations in which features cannot be detected due to either objects or people blocking the field of view of the sensors. To tackle this issue, we have utilized QR codes as markers to encode the landmark positional information.

A. Simulation Environment

The proposed encoding approach has been implemented within the ROS in Python. ROS is the current de facto standard for production-ready robot development [36], [37]. The

Algorithm 1: Feature Extraction and Hash Calculation

```
Callback:
    Calculate:
        F = getF(data);              // Orientation-ordered F set
        F_cv = getCv(F) ⊆ F;         // Set of concave features
        F_cc = getCx(F) ⊆ F;         // Set of convex features
    Define:
        H = [];                      // List of hashes
    foreach fp_i, fp_j ∈ F_cv do
        if ||fp_i − fp_j|| < δ_dw then
            H.append(doorwayHash(fp_i, fp_j));
    foreach fp_i ∈ F do
        if fp_{i+j} ∈ F_cx ∀j ∈ {−1, 0, 1} then
            H.append(roomHash(fp_{i-1}, fp_i, fp_{i+1}));
    foreach fp_i ∈ F_cv do
        if isCorner(fp_i) && notDoor(fp_i) then
            H.append(cornerHash(fp_i));
    // Utilize any matching hashes to update the robot's
    // position with respect to the global reference frame
    if ∃ h ∈ H | h ∈ NavGraph then
        updateAbsolutePosition(H));  // Use matching hashes
```

simulations were carried out within the ROS/Stage environment. A TurtleBot 3 is simulated with a 2-D lidar and wheel odometry. The robot was set to explore an indoor environment with a floorplan illustrated in Fig. 9. The environment is 40×40 m^2, and the robot has a circular shape with a diameter of 0.35 m. The simulated environment contains nine rooms with a single entrance and six more spaces with corridors between them. The starting exploration position of the robots is near the main door, in the bottom-left. The 2-D lidar had a field of view of 270° and produced 1080 samples (0.25° resolution) in each scan, with a scan rate of 10 Hz. In this experiment, we did not study the effect that different odometry methods have in the exploratory mission. Instead, we used wheel odometry and varied its error to study the impact that the corresponding computational overhead had due to a larger number of hashes being calculated.

Feature Extraction

In the simulation experiments, we utilized three types of features to localize the robot and navigate the environment: 1) doorways; 2) concave corners (CCs); and 3) rooms. These are defined from the same set of F feature points which we denote as Features of Interest $FoI = \{fp_1, \ldots, fp_F\}$, where $fp_i \in \mathbb{R}^3$. The feature extraction process is outlined in Algorithm 1. The *NavGraph* variable stores a list of hashed positions as well as an adjacency matrix with the edge hashes. A sample of this is shown in Fig. 6(c) and (d). The function *search*() calculates a certain number of hashes over a predefined area around the identified feature until it either finds a matching hash from *NavGraph* or ends the search unsuccessfully. This function ensures that the hashes are reproducible even if odometry error accumulates over the interlandmark navigation. The search area is defined based on the expected odometry error as well as the granularity of the grid utilized to define the hashes. Finally, the function *updateAbsolutePosition*() takes the matching hashes as arguments, calculates the relative position of the robot with respect to the landmarks that have been identified, and utilizes the known position of the landmarks (which is encoded in the hashes) to recalculate its own position and restart the odometry estimation.

Doorways: We define doorways as any set of two concave feature points that are within two predefined distances ($\delta_{dw,\min}, \delta_{dw,\max}$) from each other. In our simulations and experiments, we set these distances to $\delta_{dw,\min} = 1.2$ m and $\delta_{dw,\max} = 2.5$ m. Note that these feature points might not be consecutive if we consider the ordered set of feature points by orientation. We define the corresponding waypoint to be encoded according to

$$H_{dw}(fp_i, fp_j) = H\left(\text{“doorway”}, \frac{fp_i + fp_j}{2}, \angle fp_i fp_j\right). \quad (1)$$

Corners: For each CC not in a doorway, we define its corresponding hash with

$$H_{cv}(fp_i) = H(\text{“corner”}, fp_i, \angle fp_i) \quad (2)$$

where $\angle$ now represents the orientation of the normal vector to the wall surface at the position of the corner.

Rooms: A room waypoint is defined as the centroid of any three consecutive convex points and calculated as the arithmetic mean of their positions. To reduce the probability of having a mismatch in rectangular rooms where two consecutive subsets of three convex corners are visible by a robot, we add the area Δ of the triangle that the points define

$$H_{dw}(fp_i, fp_{i+1}, fp_{i+2}) = H\left(\text{“room”}, \frac{fp_i + fp_{i+1} + fp_{i+2}}{3}, \Delta_{i,i+1,i+2}\right). \quad (3)$$

B. Real-Robot Experimental Settings

The experimental environment has a size of 30 m by 25 m. For the experiments, an EAIBOT DashGo D1 was used. We installed on the mobile robot a 16-Channel Leishen 3-D Lidar, an SC-AHRS-100D2 IMU, and a Logitech c270 USB camera. The DashGo platform also provides wheel odometry from its differential drive system. The 3-D lidar was used to accurately localize the landmarks and provide ground truth (GT) with map-based localization algorithms for 3-D point clouds introduced in [38]. The camera was used to detect the QR codes and extract the encoded information in them. Fig. 7 show the implementation diagram with different ROS nodes. The 3-D lidar odometry and mapping are adapted from the LeGo-LOAM-BOR package [39]. The QR code decoding node has been written in Python using OpenCV and the Zbar library. The hash-based localization node utilizes the QR codes for localization when available and the wheel and inertial odometry as an estimation between landmarks. The QR codes utilized during the experiment are of known size (12 cm by 12 cm), and

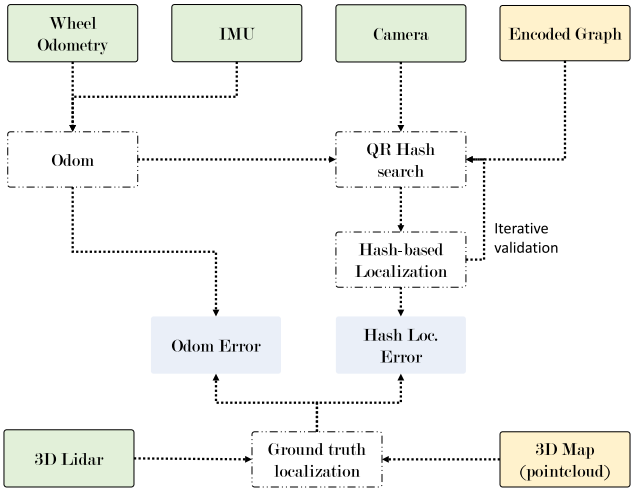


Fig. 7. Data flow in the experiments. Each box represents a ROS node which has been implemented either in C++ or Python. The outputs are the GT, odometry (odom) error, and hash-based localization error (loc. error).

the localization node was calibrated to map the size in pixels of a detected QR code in the camera to the distance to it. The localization also takes into account the relative orientation of the QR code.

C. Feature Hashing

We utilized SHA3-256 for hashing [40], which generates 32 byte hashes. It took an average of under 500 ns on an Intel Core i5-6200U CPU with the pysha3 implementation in Python. If additional security is required against offline attacks on exposed hashes, other hashing algorithms such as Bcrypt [41] can be utilized. Bcrypt needs around 300 ms to generate a hash with the same CPU. However, there is a trade-off between security and real-time operation as robots need to calculate multiple hashes per lidar scan. We believe that, in most applications, SHA3-256 is enough and can be utilized even in resource-constrained devices.

VI. Simulation and Experimental Results

We carried out a series of simulations with one and multiple robots to evaluate mainly the cost of utilizing hash matching for localization and navigation, but also the impact on the accuracy of the encoded landmarks.

A. Metrics

To evaluate the simulation results, we measured the absolute localization error of the robot with odometry only and hash matching. Furthermore, we analyzed the distribution of the computational load among the different tasks that the robots are carrying out: feature extraction, hash calculation, and hash matching. In the simulations, we also measured the effect of the odometry-based localization noise and the choice of spatial granularity for landmark positions.

B. Single-Robot Simulation Results

The aim of the simulations is to prove whether our encoded landmark localization and navigation scheme adds a significant computational overhead or not.

Fig. 8(a) shows the path recovered from odometry measurements and hash-based localization with doorway hashes only, and all three types of hashes, together with the GT. The data were recorded over 150 s; the translation odometry noise was set with $\sigma_t = 0.03$, and rotation noise with $\sigma_r = 0.05$. The error distribution for the two methods is given in Fig. 8(c).

In the simulated environment, we predefined the position of landmarks with an accuracy of 0.1 m. Therefore, when analyzing the errors in Fig. 8(b), any values below 0.15 m represent virtually zero error. Fig. 8(b) shows that the localization method is robust even when the odometry error increases significantly ($\sigma_t = 0.05$). However, there is a limit, around $\sigma_t = 0.06$, for which the size of the environment is big enough so that the robot is unable to match the correspondent landmark hashes due to the accumulated odometry drift. In order to calculate these hashes, we assumed an error tolerance with respect to its estimated position of ± 0.5 m, independently of the size of the grid utilized to locate the landmarks and generate the hashes. This limit defines the computational time required for the hash search together with the grid size.

Regarding the computational overhead necessary to calculate the hashes, estimate the robot's position, and perform path planning accordingly, Fig. 8(c) shows the distribution of computational time utilized to extract the set of features, or points of interest, from the raw lidar data and the distribution of computational time utilized in calculating and matching hashes. For an error tolerance of ± 0.5 m, the graphic shows situations in which the robot tests up to 9, 25, 121, 441, and 1681 grid positions, respectively. The search for a hash match is gradually done in a spiral manner around the estimated position and within the aforementioned error tolerance. These results show that even with a fine-grained grid search, in average the time required to localize the robot based on hashes is two orders of magnitude smaller than the time required to extract features from lidar data. In the worst case scenario, the time required can be comparable, with an equivalent order of magnitude for both hash matching and feature extraction.

C. Multirobot Exploration Simulation Results

In terms of cooperative exploration, we provide a qualitative analysis of four-robot cooperation. Fig. 9 shows the paths of four robots exploring different areas of the same simulation environment. By utilizing encoded landmarks, these robots can share their progress upon meeting in the center of the maze without revealing the raw data they have acquired. If the mission's nature is to perform surveillance or detect a series of items, robots do not need to store raw map data at all. Nonetheless, even if they did, the knowledge of the objective environment remains divided, as shown in Table I. In this case, robots acquire in average raw data form only 30% of the objective exploration environment, and 41% at most.

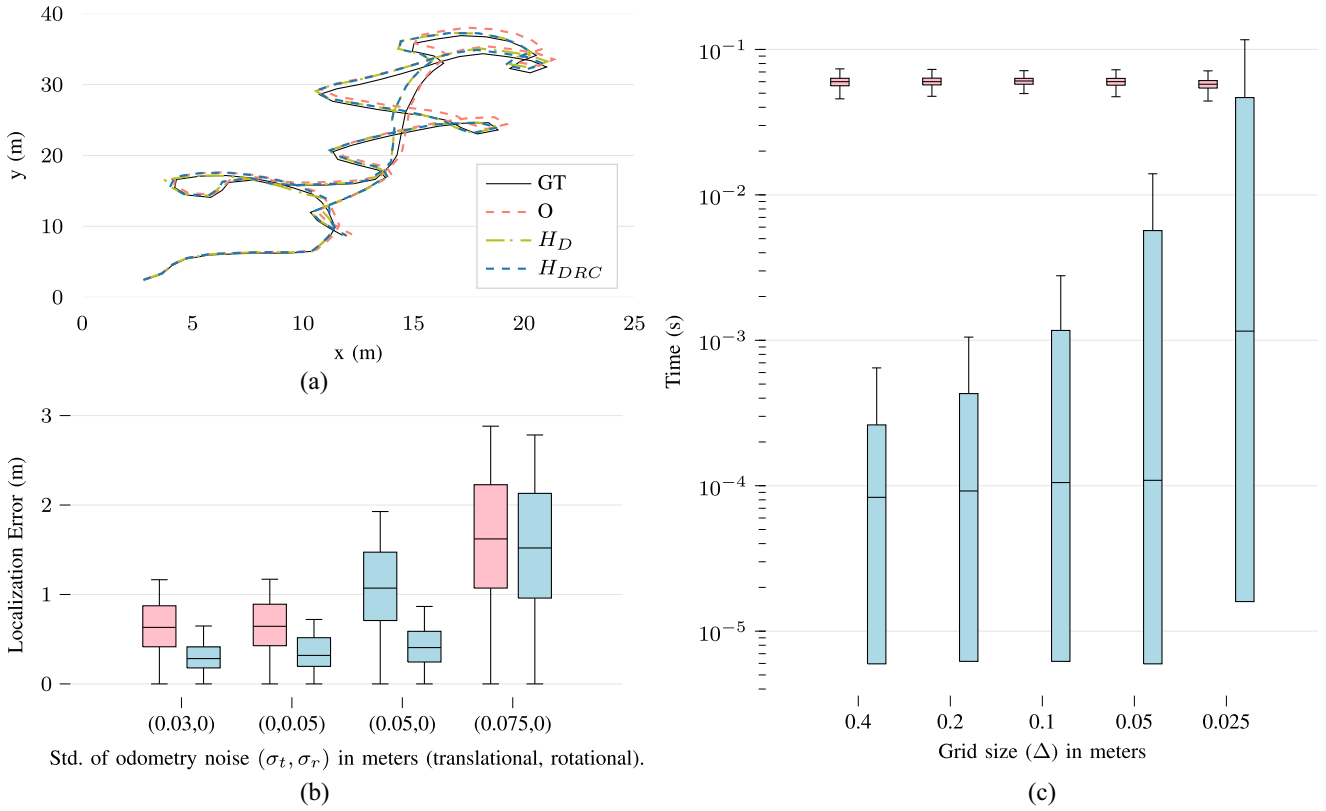


Fig. 8. Simulation results. (a) Reconstructed path with GT wheel and inertial odometry (O), only doorway hashes (D), and all features: doors(D), rooms (R), and CCs. (b) Odometry and hash-based localization errors for different odometry noise levels. (c) Execution time distribution for the feature extraction (red) and hash matching (blue) processes, where the grid size represents the search space when trying to find a hash match.

TABLE I
ENVIRONMENT KNOWLEDGE DISTRIBUTION DURING THE COLLABORATIVE EXPLORATION SIMULATION

	Hashes found	**Area**
Robot$_1$ (purple)	12/32	23%
Robot$_2$ (orange)	16/32	29%
Robot$_3$ (red)	15/32	41%
Robot$_4$ (cian)	11/32	27%

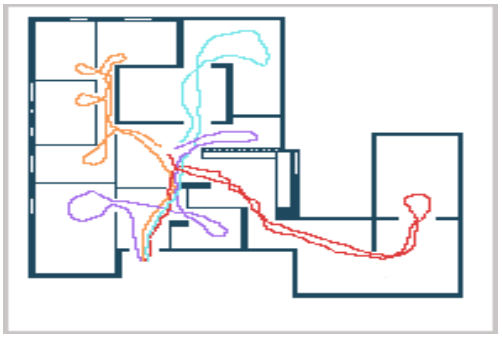


Fig. 9. Illustration of the paths followed by four robots during the multirobot collaborative exploration simulation.

D. Experimental Results

Fig. 10(a) shows the path recovered from odometry measurements and hash-based localization (QR codes). The error in the odometry is significantly higher than in the simulation experiments due to a drift in the yaw measurements. However, the translational odometry error is much smaller. The hash-based localization is able to correct this orientation whenever a QR code is within the field of view and, therefore, it does not suffer from the yaw drift. The maximum hash-based localization error that we observed was of 41.3 cm (between observations of landmarks and owing to the accumulated odometry drift). This allowed the utilization of a fine grid of 2 cm for calculating the landmark hashes. We set, experimentally, a 1 m^2 hash search area around the estimated location.

A total of 23 QR codes were installed in the office environment, and the tests were done with a small number of persons in their offices. Out of those, 17 QR codes were utilized by the robot during its navigation. The mission times at which at least one code was in sight, the error from each observation and the global error distribution are shown in Fig. 10(b). The execution time of the hash matching algorithm was on average over one order of magnitude smaller than the time required to extract the QR codes from camera images. Thus, the overhead was mostly negligible. Only in a reduced number of occasions was the latency of these two processes comparable, as Fig. 10(c) shows.

E. Viability and Usability

We have seen that the computational overhead added when encoding landmarks is mostly negligible. Thus, our approach

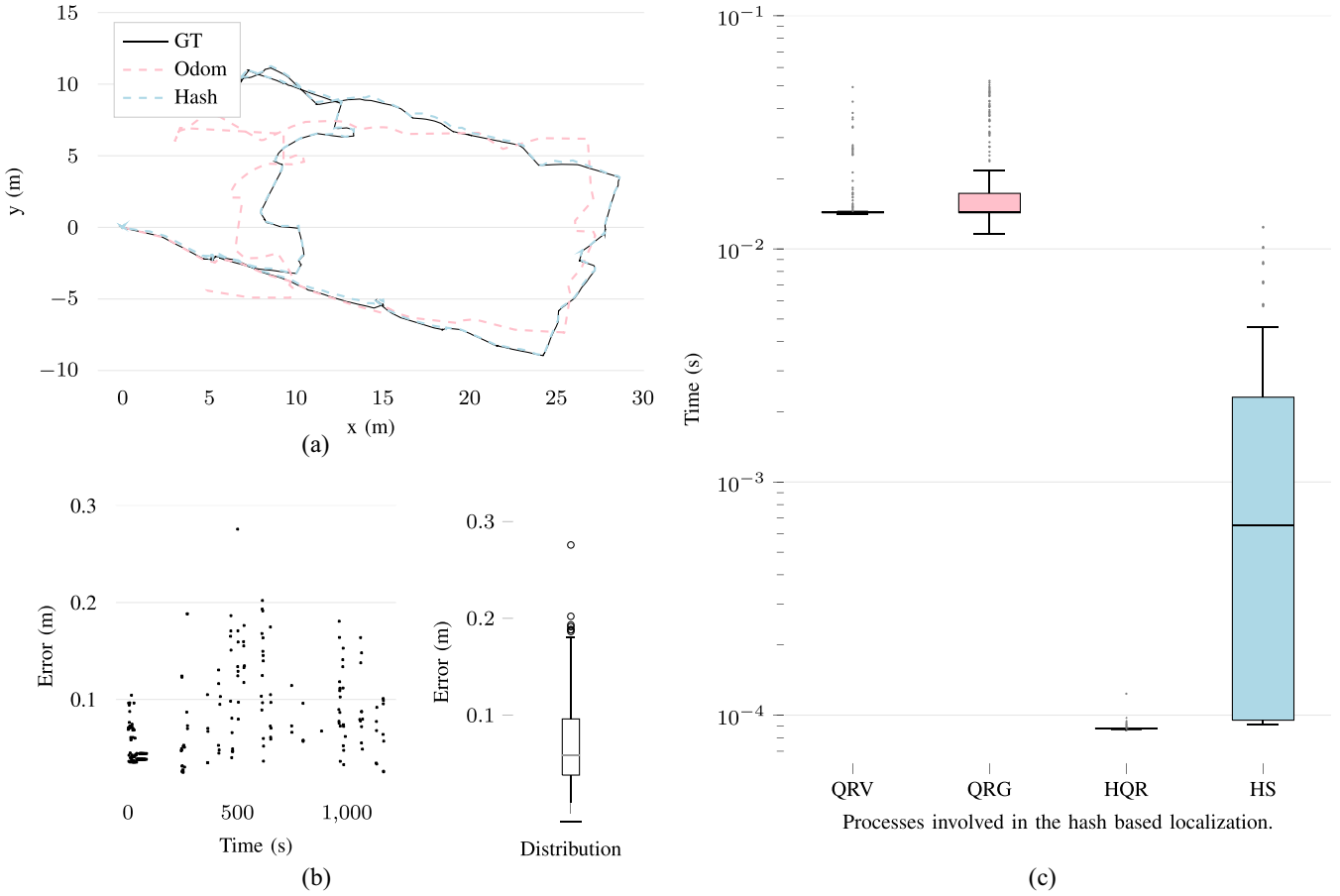


Fig. 10. Experiment results. (a) Reconstructed path with GT, wheel and inertial odometry (Odom), and hash-based localization (Hash). (b) Scatter plot with the localization error every time that a QR code is within the field of view of the camera, and a boxplot with the global error distribution. (c) Execution time for the different processes involved in the localization, with the hash search and matching being one to two orders of magnitude below the image processing processes that would still be in use in a traditional approach.

could be incorporated on top of many existing navigation and localization schemes, whether they are landmark based or not, to increase the level of security if the error tolerance allows.

This approach has additional uses when more than one robot is taken into consideration. In multirobot cooperation, different robots can share their plans, progress, or position (based on the navigation graph only) with others by utilizing the same hashes or parts of them. This would reduce the possibility of raw data being exposed but also virtually eliminate the options for attackers or byzantine agents to affect the mission, as has been shown in [7]. Moreover, as we have shown with the multirobot simulations, the framework allows for multirobot exploration or other collaborative missions, reducing the fraction of raw data or environment information that has to be made available to each individual unit.

F. Validation of Sensor Data

In the navigation experiments conducted, we were also able to leverage the encoded landmarks for validating the sensor data leading to odometry estimations. In these experiments, the robot was able to estimate the error (i.e., drift) in the odometry as a function of the computational time required to decode the landmark position (i.e., the number of hashes that need to be tested against the encoded map information). If the odometry error is too large, or a sensor malfunction, then the hash cannot be decoded unless the hash search radius and computational timeout are extended consequently. In both the simulations and real-robot experiments, we utilized only IMU and wheel odometry data (e.g., versus visual inertial or lidar odometry) to evaluate the performance of the proposed methods in more adversarial conditions where the odometry error between landmarks may increase significantly.

G. Tradeoff and Security Considerations

We have shown through simulation and real-world experiments that the proposed framework has a negligible impact in terms of usage of computational resources. In the simulations, performing a hash search in a grid map with a resolution of 2.5 cm requires on average over an order of magnitude less computational resources than the lidar feature extraction that is inherent to any localization process. Nonetheless, for each specific application in which an encoded instruction graph is used, there will be a certain size of the hash search space where the instruction decoding is no longer negligible.

Another tradeoff occurs between real-time operation and security. The larger the hash search space, the more computational resources are needed to perform a brute-force attack on the encoded information. Nonetheless, performing such

an attack requires information on the encoding algorithm, therefore requiring physical access to the robot or access to the control algorithms and data processing stack. The specific threshold on the hash search granularity will be defined, among other factors, by the hashing algorithm, and the ways in which the hashes are generated (e.g., involving time or unknown external inputs from the controller can, in many situations, render the brute-force attacks unfeasible).

In terms of the resilience of the proposed framework against adversarial attacks, the main security vulnerability that we have detected is the ability of an attacker to reproduce the encoded commands even without decoding them, potentially triggering the robot into repeating actions. If data are spoofed when transmitted to the robot, the robot's behavior could be studied under different encoded commands, and then an attacker could trigger a known mission. While this cannot be completely mitigated within our proposed approach, we have introduced time-checked actions and event-triggered actions. If a one-time action is required and either the start of the mission or its timing is known, then it is feasible to include the time component into the encoded instructions to avoid repeated actions even if the encoded data are spoofed. Moreover, other generic strategies designed against data spoofing could be introduced on top of our framework.

VII. Conclusion and Future Work

Security and safety in robotics are crucial aspects to consider given the current surge of autonomous systems. In this direction, further research is needed on the validation of data at the different layers of robotic systems and, in particular, the validation of the interaction of a robot with its environment. This interaction often starts with navigation, which has been studied in this article.

Navigation and localization in autonomous robots require large amounts of raw data for a long-term operation, either given *a priori* by a mission controller or acquired by robots while performing their missions. In addition, validating the integrity of both mission instructions and sensor data without any external feedback is an open problem. In this article, we have presented a framework that enables robots to validate both the correct operation of their onboard hardware and sensors and the integrity of information received from an external controller.

In particular, to the best of our knowledge, this article introduces and evaluates the first framework which allows robots to effectively perform their missions while also performing end-to-end validation of information, demonstrated on navigation and localization with encoded landmarks. We have shown that utilizing an encoded navigation graph adds only a negligible computational overhead even when high-accuracy positioning is required.

The end-to-end validation scheme demonstrated in this work for navigation tasks can be naturally extended to cover virtually all domains of robotic operation. In future work, we will focus our research efforts on experimentation in more realistic environments and in particular industrial settings. We will aim at extending this approach to other forms of interaction between a robot and its environment from multirobot collaborative assembly to human–robot interaction and control.

References

[1] L. A. Kirschgens, I. Z. Ugarte, E. G. Uriarte, A. M. Rosas, and V. M. Vilches, "Robot hazards: From safety to security," 2018, *arXiv:1806.06681*.

[2] S. Bragança, E. Costa, I. Castellucci, and P. M. Arezes, "A brief overview of the use of collaborative robots in industry 4.0: Human role and safety," in *Occupational Environmental Safety Health*. Cham, Switzerland: Springer, 2019.

[3] J. Huang *et al.*, "ROSRV: Runtime verification for robots," in *Proc. Int. Conf. Runtime Verification*, 2014, pp. 247–254.

[4] G. W. Clark, M. V. Doran, and T. R. Andel, "Cybersecurity issues in robotics," in *Proc. CogSIMA*, 2017, pp. 1–5.

[5] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, Apr. 2015.

[6] S. Rivera, S. Lagraa, C. Nita-Rotaru, S. Becker, and R. State, "ROS-defender: SDN-based security policy enforcement for robotic applications," in *Proc. Security Privacy Workshops*, 2019, pp. 114–119.

[7] E. C. Ferrer, T. Hardjono, A. Pentland, and M. Dorigo, "Secure and secret cooperation in robot swarms," *Sci. Robot.*, vol. 6, no. 56, 2021, Art. no. eabf1538. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abf1538

[8] J. N. K. Liu, M. Wang, and B. Feng, "iBotGuard: An Internet-based intelligent robot security system using invariant face recognition against intruder," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 1, pp. 97–105, Feb. 2005.

[9] N. M. Rodday, "Exploring security vulnerabilities of unmanned aerial vehicles," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2016, pp. 993–994.

[10] J. Wan, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[11] E. C. Ferrer, E. Jiménez, J. L. Lopez-Presa, and J. Martín-Rueda, "Following leaders in Byzantine multirobot systems by using blockchain technology," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1101–1117, Apr. 2022.

[12] J. Miller, A. B. Williams, and D. Perouli, "A case study on the cybersecurity of social robots," in *Proc. ACM/IEEE HRI*, 2018, pp. 195–196.

[13] F. Py and F. Ingrand, "Dependable execution control for autonomous robots," in *Proc. IEEE/RSJ IROS*, vol. 2, 2004, pp. 1136–1141.

[14] Y. Tang, W. Yang, D. W. C. Ho, D. Zhang, and B. Wang, "Event-based tracking control of mobile robot with denial-of-service attacks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 9, pp. 3300–3310, Sep. 2020.

[15] S. Tiku and S. Pasricha, "Overcoming security vulnerabilities in deep learning–based indoor localization frameworks on mobile devices," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, p. 114, Nov. 2019.

[16] A. H. A. Rahman, R. Sulaiman, N. S. Sani, A. Adam, and R. Amini, "Evaluation of peer robot communications using cryptoros," *Evaluation*, vol. 10, no. 7, pp. 658–663, 2019.

[17] L. V. Legashev, T. V. Letuta, P. N. Polezhaev, A. E. Shukhman, and Y. A. Ushakov, "Monitoring, certification and verification of autonomous robots and intelligent systems: Technical and legal approaches," *Procedia Comput. Sci.*, vol. 150, pp. 544–551, Jan. 2019.

[18] K. M. A. Yousef, A. AlMajali, S. A. Ghalyon, W. Dweik, and B. J. Mohd, "Analyzing cyber-physical threats on robotic platforms," *Sensors*, vol. 18, no. 5, p. 1643, 2018.

[19] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the Internet for ROS: A view of security in robotics research," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 8514–8521.

[20] R. Amini, R. Sulaiman, and A. H. A. R. Kurais, "CryptoROS: A secure communication architecture for ROS-based applications," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 10, pp. 189–194, 2018.

[21] J. Kim, J. M. Smereka, C. Cheung, S. Nepal, and M. Grobler, "Security and performance considerations in ROS 2: A balancing act," 2018, *arXiv:1809.09566*.

[22] B. R. Hilnbrand and P. Robert, "Automated vehicle map localization based on observed geometries of roadways," U.S Patent 10 289 115, May 14, 2019.

[23] H. Sobreira *et al.*, "Map-matching algorithms for robot self-localization: A comparison between perfect match, iterative closest point and normal distributions transform," *J. Intell. Robot. Syst.*, vol. 93, no. 2, pp. 533–546, 2019.
[24] R. S. Andersen, J. S. Damgaard, O. Madsen, and T. B. Moeslund, "Fast calibration of industrial mobile robots to workstations using QR codes," in *Proc. IEEE ISR*, Oct. 2013, pp. 1–6.
[25] H. Zhang, C. Zhang, W. Yang, and C.-Y. Chen, "Localization and navigation using QR code for mobile robot in indoor environment," in *Proc. IEEE ROBIO*, Dec. 2015, pp. 2501–2506.
[26] P. Nazemzadeh, D. Macii, D. Fontanelli, and L. Palopoli, "Indoor localization of mobile robots through QR code detection and dead reckoning data fusion," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 6, pp. 2588–2599, Dec. 2017.
[27] A. R. Zamir and M. Shah, "Accurate image localization based on google maps street view," in *Proc. ECCV*, 2010, pp. 255–268.
[28] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1744–1756, Sep. 2017.
[29] J. Thoma, D. Paudel, A. Chhatkuli, T. Probst, and L. Gool, "Mapping, localization and path planning for image-based navigation using visual features and map," in *Proc. IEEE CVPR*, 2019, pp. 7375–7383.
[30] L. Cheng, C.-D. Wu, and Y.-Z. Zhang, "Indoor robot localization based on wireless sensor networks," *IEEE Trans. Consum. Electron.*, vol. 57, no. 3, pp. 1099–1104, Aug. 2011.
[31] C. M. Almansa, W. Shule, J. P. Queralta, and T. Westerlund, "Autocalibration of a mobile UWB localization system for ad-hoc multi-robot deployments in GNSS-denied environments," 2020, *arXiv:2004.06762*.
[32] W. Shule, C. M. Almansa, J. P. Queralta, Z. Zou, and T. Westerlund, "UWB-based localization for multi-UAV systems and collaborative heterogeneous multi-robot systems: A survey," 2020, *arXiv:2004.08174*.
[33] Y. Song, M. Guan, W. P. Tay, C. Law, and C. Wen, "UWB/LiDAR fusion for cooperative range-only SLAM," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 6568–6574.
[34] M. Gadd and P. Newman, "A framework for infrastructure-free warehouse navigation," in *Proc. ICRA*, 2015, pp. 3271–3278.
[35] Q. Qin, D. Zhu, Z. Tu, and J. Hong, "Sorting system of robot based on vision detection," in *Proc. IWAMA Workshop*, 2017, pp. 591–597.
[36] M. Quigley, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, 2009, pp. 1–6.
[37] A. Koubâa, *Robot Operating System (ROS)*. Cham, Switzerland: Springer, 2017.
[38] Q. Li, J. P. Queralta, T. N. Gia, Z. Zou, and T. Westerlund, "Multi sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments," in *Proc. 9th IEEE CIS-RAM*, 2019, pp. 1–7.
[39] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IROS*, 2018, pp. 4758–4765.
[40] J. Czajkowski, L. G. Bruinderink, A. Hülsing, and C. Schaffner, "Quantum preimage, 2nd-preimage, and collision resistance of SHA3," IACR, Lyon, France, Rep. 2017/302, 2017.
[41] P. Sriramya and R. A. Karthika, "Providing password security by salted password hashing using Bcrypt algorithm," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 13, pp. 5551–5556, 2015.

Jorge Peña Queralta (Graduate Student Member, IEEE) received the B.S. degree in mathematics and physics engineering from UPC BarcelonaTech, Barcelona, Spain, in 2016, the M.Sc. (Tech.) degree in information and communication science and technology from the University of Turku, Turku, Finland, in 2018, and the M.Eng. degree in electronics and communication engineering from Fudan University, Shanghai, China, in 2018. He is currently pursuing the Doctoral degree with the Turku Intelligent Embedded and Robotic Systems (TIERS) Group, University of Turku.

Since 2018, he has been a Researcher with the TIERS Group, University of Turku. His research interests include multirobot systems, collaborative autonomy, distributed perception, and edge computing.

Qingqing Li (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and the M.Sc. degree in electronics and communication engineering from Fudan University, Shanghai, China, in 2016 and 2018, respectively, and the M.Sc. degree in information and communication science and technology from the University of Turku, Turku, Finland.

Since 2019, he has been a Researcher with the Turku Intelligent Embedded and Robotic Systems Group, University of Turku. His research interests include sensor fusion for autonomous robots and vehicles, 3-D point cloud data analysis, and multirobot collaboration.

Eduardo Castelló Ferrer received the B.Sc. degree (Hons.) in intelligent systems from the University of Portsmouth, Portsmouth, U.K., in 2007, and the M.Eng. and Ph.D. degrees in robotics engineering from Osaka University, Suita, Japan, in 2011 and 2016, respectively.

His experience and interests comprise robotics, blockchain technology, and complex systems. He is a Marie Curie Fellow with the MIT Connection Science and MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA, where he explores the combination of robotic systems and blockchain technology. His work focuses on implementing new security, behavior, and business models for distributed robotics by using novel cryptographic methods.

Tomi Westerlund (Senior Member, IEEE) received the Ph.D. degree from the University of Turku, Turku, Finland, in 2018.

He is an Associate Professor of Autonomous Systems and Robotics with the University of Turku, Turku, Finland, and a Research Professor with Wuxi Institute of Fudan University, Wuxi, China. He leads the Turku Intelligent Embedded and Robotic Systems Research Group, University of Turku. His current research interest is in the areas of Industrial IoT, smart cities, and autonomous vehicles (aerial, ground, and surface) as well as (co-)robots. In all these application areas, the core research interests are in multirobot systems, collaborative sensing, interoperability, fog and edge computing, and edge AI.