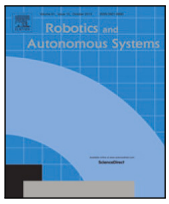
Robotics and Autonomous Systems 196 (2026) 105246



Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot





Mobile robot defensive wayfinding for incomplete and ambiguous route instructions

Hao Liang Chen [a,*], Elena Torta [a], Herman Bruyninckx [a,b,c], René van de Molengraft [a]

[a] Department of Mechanical Engineering, Eindhoven University of Technology, De Zaale, 5612 AZ, Eindhoven, The Netherlands
[b] Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300, 3001, Leuven, Belgium
[c] Flanders Make, Gaston Geenslaan 8, 3001, Leuven, Belgium

ARTICLE INFO

Keywords:
Mobile robot
Hypothesis tree
Defensive wayfinding
Ambiguous route instructions
Incomplete route instructions

ABSTRACT

Humans typically convey *route instructions* as a sequence of *locations* where *actions* need to be executed, e.g., turn left at the second crossing. It is not uncommon that humans generalize or forget parts of the environment or route instructions. Consequently, route instructions can be *incomplete* as decision points, in the form of action-location descriptions, are omitted. In addition, omitted location descriptions may be similar to one already present in the route instructions, thus making it *ambiguous* to the wayfinder at which specific location to execute the action. *Defensive wayfinding* then characterizes the procedure to deal with such uncertainties in the route instructions. The state of art is not capable of performing defensive wayfinding for mobile robots with route instructions that are incomplete and ambiguous regarding the specific location for action execution. This work tackles this problem by taking inspiration from practices of human wayfinding literature and incorporating it into the robotics context. More in particular by adding three types of knowledge to the route instructions: (1) the types of locations that can be encountered by the robot, (2) the action models to leave those locations, and (3) the temporal and spatial constraints on the (topological) sequence of encountered locations. (In the context of human defensive wayfinding, this additional information is "background knowledge", or it is provided in the form of a rough sketch.) Our defensive wayfinding approach relies on a hypothesis tree that associates (parts of) the executed robot path to the route instructed path. We experimentally validate defensive wayfinding in simulation with a mobile robot equipped with a 2D laser range finder in a corridor-junction environment that can be topologically inconsistent with the provided route instructions.

1. Introduction

Wayfinding is a term that originates from cognitive psychology and is defined in [1, p. 6] as *"the process of determining and following a path or route between an origin and destination"*. *Defensive wayfinding* elaborates on this term by taking into account *uncertainties* in the route instructions and is defined in [2] as *"a form of goal-orientated wayfinding occurring under adverse conditions, where the wayfinder must exert excessive mental effort to align, correct, supplement or find alternative interpretations of the information acquired from route instructions or signage because of a mismatch with the perceived structure of the environment or with their own expectations about the environment"*. This paper investigates the application of defensive wayfinding on a mobile robot. More specifically, for route instructions that have insufficient (uniquely described) *decision points* to *unambiguously* follow the route, i.e., locations at which the wayfinder can take actions to continue the route in different directions, e.g., at intersections [3]. To the best of the authors' knowledge, no methods exist yet that address error recovery for such route instructions in a mobile robot context.

Fig. 1 illustrates an example of an inconsistency between the route instruction and the ground truth. That inconsistency is not rare and can be explained in several ways. On the one hand, humans forget or simplify parts of the environment [2,4,5]. On the other hand, humans tend to generalize or chunk parts of the route instructions to remove "unnecessary" detail (from the human's perspective) [6,7]. However, a robot may not have the underlying knowledge to understand *why* some

* Corresponding author.
E-mail addresses: h.l.chen@tue.nl (H.L. Chen), e.torta@tue.nl (E. Torta), herman.bruyninckx@kuleuven.be (H. Bruyninckx), m.j.g.v.d.Molengraft@tue.nl (R. van de Molengraft).

https://doi.org/10.1016/j.robot.2025.105246
Received 22 March 2024; Received in revised form 13 July 2025; Accepted 31 October 2025
Available online 4 November 2025
0921-8890/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

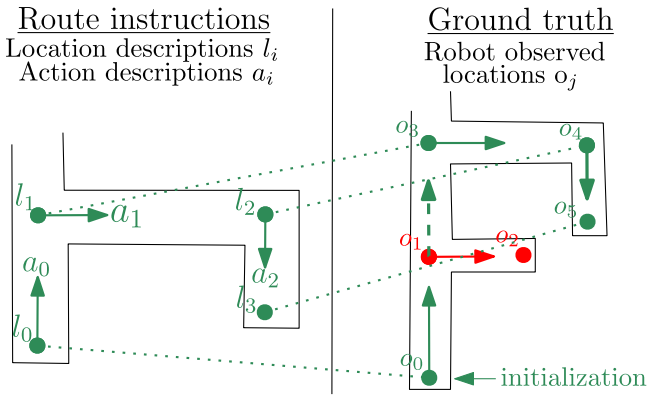


Fig. 1. Left: map on which the decision points d_i are visualized as a location description l_i (green dots) and an action description a_i (green arrow). Right: map produced by the robot with *observed* locations o_j (green/red dots) and *executed* actions (green/red arrows). Green dots o_j are associated with l_i via green dotted lines, red dots indicate omitted location descriptions, red arrows represent wrongly executed actions at red dots, and green dashed arrows represent the intended actions at red dots. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

detail is removed, e.g., that it may be common sense to humans to omit instructions of *not* turning into a dead-end street.

A mobile robot cannot expect a human to provide environmental knowledge that metrically aligns with the ground truth, e.g., in the form of metric accurate grid maps [8]. Instead, they convey route instructions either textually, e.g., Turn-by-Turn instructions [9], or pictorially, e.g., paths in a sketch map [10]. We hypothesize that textual route instructions can be derived from the richer pictorial route instructions [4], similar to how humans verbally convey a route by interpreting a map [7]. Based on this hypothesis, this paper focuses on the less rich textual route instructions. We convey these textual route instructions as a sequence of decision points $d = (a, l)$[1] [2,11], i.e., "$(a_0, l_0) \rightarrow (a_1, l_1) \rightarrow \cdots \rightarrow (a_{n-1}, l_{n-1})$" where $n \in \mathbb{N}$ represents the number of instructions, "a" is a description of the action, and "l" is a description of the location. Each l contains detailed information such as the type of the location, e.g., whether it represents a T-junction or a roundabout. Conventional robot wayfinding is successful if each of these location descriptions are uniquely described and/or the sequence of location descriptions matches the sequence of robot observed locations "o" in the ground truth, i.e., the map topology remains consistent [12–15].

In this paper, we focus on two specific types of route instruction uncertainties, inspired by the work of [2], that lead to greater ambiguity in performing correct wayfinding:

1. **Incomplete route instructions**: the route instructor omits a part of the route instruction. For example, the robot observed location o_1 on the right of Fig. 1 has no associated instruction.
2. **Ambiguity of location execution**: Multiple robot observed locations locally satisfy one particular location description. For example, both o_1 and o_3 in Fig. 1 resemble the T-junction at l_1.

The second type of uncertainty is reasonable as robot perceptual capabilities are not as advanced as these of humans, nor do humans specify l with metric accuracy. Conventional robot wayfinding would lead the robot to wrongly traverse $o_0 \rightarrow o_1 \rightarrow o_2$ in Fig. 1 without a way to recover.

The specific contributions of this paper are:

1. **Provide additional knowledge to resolve incompleteness and location ambiguity in route instructions** (Section 3): the route instructions are revised to allow the specification of (un)certain parts. Additionally, environmental knowledge is provided of the types of locations that can be encountered. The combination provides the wayfinder with a mechanism to generate alternative interpretations to the route instructions.
2. **Robot defensive wayfinding algorithm** (Section 4): The revised route instructions of Section 3 are used in combination with a hypothesis tree to keep track of, reject, and add interpretations of the *best fitting path*, i.e., the likely route instructed path.

The methods proposed are experimentally validated in Section 5. We show the feasibility of connecting the defensive wayfinding algorithm of Section 4 to the continuous robot motion and perceptual capabilities in a simulation. The validation via simulation does not reduce the applicability of the method, as the concept of defensive wayfinding focuses on uncertainties in the route instructions and not on uncertainties of the motion and perceptual capabilities of the wayfinder. The mobile robot is equipped with a 2D laser range finder and moves in a corridor-junction environment with incomplete and ambiguous route instructions.

2. Related work

2.1. Complete route instructions for robot wayfinding

Literature formulates route instructions for robot wayfinding assuming topological consistency with the ground truth, and thus that sufficient *decision points* are present to let a robot unambiguously follow the route.

One part of literature focuses on pictorial route instructions by localizing and following a path in a sketch map [14,16]. These sketch maps contain unreliable quantitative spatial relations, in contrast to traditional grid maps [8,17]. In [14], the authors solve an optimization problem which preserves the sketched landmark configurations along the drawn path. In [16], the authors assume diffeomorphism between the sketch map and the ground truth, which allows the sketch map to be aligned through local scaling parameter estimations. In both approaches, the authors assume that the necessary decision points are drawn at which the robot needs to change its direction.

Another part of literature focuses on textual route instructions [12,13,15,18–20]. In [19], the authors translate a path in a sketch map with objects into a textual sequence of locations with corresponding actions. In [18], the authors perform room-to-room navigation under the assumption that the sequence of encountered rooms is correct. In [15], the authors perform wayfinding by using positional information stored in markers, e.g., its relative position to other markers. Wayfinding can then be performed by navigating from marker to marker. In [13], the authors formulate a backtracking procedure in case a decision point is missed due to an error by the robot, e.g., not recognizing the location due to occlusion of perceptual features. In [12], the authors similarly allow error recovery in case the robot performed the wrong action, e.g., due to wheel slip. A Partially Observable Markov Decision Process (POMDP) is used to recover from erroneous state transitions. In all of the above works, the authors assume that no uncertainties are present in the route instructions.

To the best of the authors' knowledge, only [20] considers some form of uncertainty in the route instructions. They focus on *action ambiguity* for which the route instructed action to leave a location is incorrectly specified. For example, at l_1 in Fig. 1 the route instruction would describe an action similar to a_0. That type of route instruction uncertainty is complementary to the uncertainties considered in this paper. In case of incomplete route instructions, the authors of [20] suggest a brute-force approach that analyzes all the possible ways to traverse the environment. That, however, disregards the time consumption necessary for a mobile robot to explore a specific part of the environment.

[1] list of symbols and definitions in Appendix A

2.2. Solving location ambiguity

2.2.1. Robots assume correct location association

A closely related problem to this paper is *localization*, i.e., estimating the robot's position in a map, in maps that have unreliable quantitative spatial relations. Examples are semi-metric maps [21], sketch maps [16], and floor plans [22]. In these papers, some mathematical operation, such as (non-uniform) scaling, is performed in order to align the geometrical features of these maps with the ground truth. Human provided pictorial instructions, however, cannot be assumed to contain all the (necessary) geometrical features to successfully perform such alignment operations [5–7]. In [5], the authors mention that sketched objects may vary greatly in shape or size with respect to the ground truth. In [7], the authors mention the concept of *generalization*. For example, a collection of geometric buildings could be amalgamated into one sketched area. In [6], the authors mention the concept of *chunking*. For example, by omitting intersections in a map at which no directional change is required by the wayfinder.

Instead of aligning the quantitative spatial relations, an alternative approach is to preserve the *qualitative* spatial relations between features in the map. For example, that feature A is "left-above" a feature B. This abstraction allows for greater tolerances to metric disturbances, such as changes to scaling. However, it also introduces a greater risk of errors due to incorrect feature association whenever features are present that look similar. In robot wayfinding literature, these errors do not occur as they assume that the landmark association problem is solved [13–15]. Consequently, the robot can immediately identify the correct location at which an action needs to be performed. In [14], the authors configure the robot perception such that similar-looking cones can be uniquely identified. In [13,15], the authors introduce markers to uniquely identify landmarks or locations.

To distinguish non-unique landmarks, some "higher-order" pattern need to be used [12,18,19]. In [12], the authors add unique triangle patterns between non-unique circle features to recognize the locations. In [19], occupied cells are clustered forming a pattern to recognize the location. The order in which the route instructed patterns are encountered corresponds to the ground truth. In [18], the authors distinguish corner features via qualitative spatial relations related to a convex room pattern. Navigation succeeds in case the spatial relation between the rooms are indeed *sequential*. In all of the above works, it is assumed that the robot can unambiguously recognize the route-instructed location at its first encounter. In contrast, the robot in this paper is not supposed to be able to do this.

2.2.2. Location disambiguation via topological relations

Route instructions can be represented as a graph, with vertices indicating the locations, and the edges indicating the actions to navigate to the connected node. In essence, the above robot wayfinding literature distinguishes vertices by comparing the next reachable vertex properties to the local map that is obtained from a robot's perception.

In case individual vertices are indistinguishable, literature in online graph matching (e.g., SLAM) solves this by focusing on multi-vertex patterns. In [23,24], the authors verify vertices by letting a robot traverse a path through a known sequence of vertices. Essentially, a route instruction is online formulated and checked for traversability. In [25], the authors distinguish vertices by verifying prior sub-graph patterns that also consider directly neighboring vertices. In [26–28], the authors consider graph patterns that satisfy graph planarity constraints. Additionally, quantitative or qualitative spatial constraints were imposed on the graph pattern, e.g., "exact edge distance values" or relative directional information as "vertex A is positioned in sector 1 of vertex B". However, these approaches do not sufficiently deal with incomplete route instructions or its equivalent of a prior that is topologically inconsistent

Inspiration can be gained from the offline map matching literature (in the domain of geographic information systems) in dealing with incomplete route instructions. These approaches typically consider a large search space where every vertex-to-vertex correspondence is analyzed. In [29], the authors define force histograms between objects both for the ground truth and the sketch map. A search procedure is initialized to obtain optimal object correspondences based on these force histograms. In [30], the authors extract topological graphs from the sketch map and the ground truth. Vertices are associated via topological properties and a hypothesis tree is maintained to evaluate the associations. In [31–33], the authors extract multiple graphs each focusing on a different qualitative spatial relation, e.g., cardinal directions between vertices. A hypothesis tree of vertex associations is maintained in which the scoring is based on the number of matches and on the number of admissible qualitative graphs. A graph is admissible if the qualitative spatial relations can be preserved with the associated vertices. In [33], the authors explicitly focus on topologically inconsistent sketches whilst using spatial invariant relations obtained from [5]. In [10], the authors use a genetic algorithm to match routes of topologically inconsistent sketches. The objective function is based on qualitative spatial criteria such as the number of intersections along the route, or on the type of intersections. A set of route-to-route matches is found, where further pruning is obtained by applying a second genetic algorithm over the set of route-to-route matches. All these works show that topological patterns over multiple vertices can be used to make correct vertex-to-vertex associations. However, these approaches do not consider the online wayfinding aspect for which that topological knowledge is not immediately available.

3. Modified route instructions to tackle incompleteness and location ambiguity

A Route Instruction (RI) specifies a sequence of decision points. According to [2], these points consists of a combination of an action description and a location description, resulting in

$$RI = [d_0, d_1, \dots, d_n] = [(a_0, l_0), (a_1, l_1), \dots, (a_{n-1}, l_{n-1})].$$

This paper describes these actions and locations symbolically, and assumes that the "lower-level" motion control and perceptual capabilities to execute and recognize them are available. For example, a constraint-based controller as in [34] and location recognition via landmark patterns as in [12,19]. We distinguish locations via types "t" as l^t, e.g., T-junctions and L-junctions, and actions via features "f" to exit these locations $a^{t,f}$, e.g., a specific exit direction of a T-junction. For readability, these superscripts are omitted when not needed. To deal with *incomplete* and *ambiguous* route instructions, we extend the RI with additional environmental knowledge as

$$RI = \{\mathcal{W}, [(a_0^{t,f}, l_0^t, \delta_0), \dots, (a_{n-1}^{t,f}, l_{n-1}^t, \delta_{n-1})]\}, \tag{1}$$

of which the symbols will be explained further throughout this section.

Throughout this paper, the following assumptions are formulated to increase the applicability of the route instruction:

Assumption 1 (*Solved Initialization Wayfinder*)**.** The wayfinder starts in an observed location o_0 in the ground truth that is associated to location description l_0 with an initial heading in the direction of action description a_0.

Assumption 2 (*In-sequence Instructions*)**.** the ordered sequence of route instructed location descriptions l are preserved in the ground truth, as that relative order is maintained within the sequence of encountered observed locations o when executing the intended route. The association between an observed location o_j and location description l_i, denoted as $o_j \sim l_i$, can thus only be made if $(l_0, \dots, l_{i-1})$ can all be associated to (a subset of) $(o_0, \dots, o_{j-1})$.

Assumption 3 (*Correct Action Specifications*)**.** at the correct location associations $o_j \sim l_i$, the wayfinder should execute the route instructed action a_i.

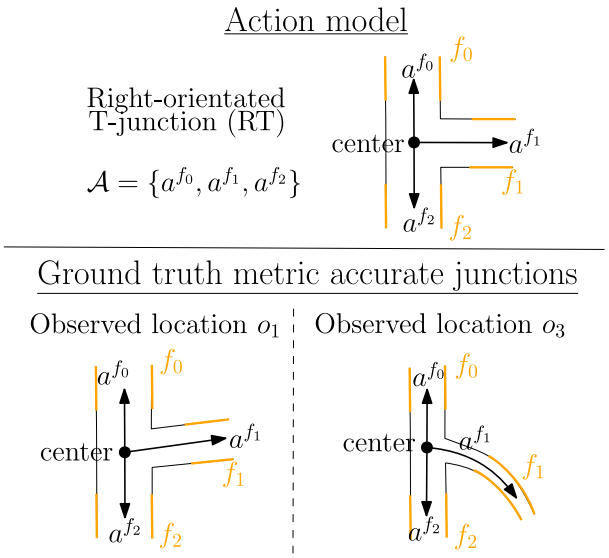


Fig. 2. Visualization of $\mathcal{A}$ for a Right-orientated T-junction (RT) on top where actions $a^{RT,f}$ (black arrows) are related to the corridor feature components (yellow equidistant curves). The bottom visualizes how the components of $\mathcal{A}$ can be applied on geometrically differing locations o_j. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Resolution of incomplete instructions

Due to incomplete route instructions, not all o_j can be associated to a route instructed location l_i. This paper's approach associates o_j that are not associated to l_i to non-instructed locations $\lambda_{i,k}$; that is, they do not appear in the route instructions. The subscript i indicates that λ is present between l_i and l_{i+1}, and the subscript $k \in \mathbb{N}$ represents an occurrence counter. For example, we would make the association $o_1 \sim \lambda_{0,0}$ in Fig. 1 as the first non-instructed location between l_0 and l_1.

3.1.1. Action knowledge for error recovery

In a route instruction, it is common to only specify one action to exit a location. We suggest that all possible actions need to be specified to a robot, to allow it to hypothesize about alternative route instruction interpretations. For example, by explicitly specifying in Fig. 1 that the green dashed arrow at o_1 is another viable action to continue wayfinding. Hence, each location type that can be encountered needs an action model $\mathcal{A}^t$ of all the possible ways to exit it. That action model is stored within $\mathcal{W}$ representing the *world model* [34].

Fig. 2 (top) visualizes an example of $\mathcal{A}^{RT}$ for a so-called *Right-orientated T-junction* (RT). The junction consists of three corridor features f_0, f_1, and f_2 (yellow equidistant curves), and corresponding actions a^{RT,f_0}, a^{RT,f_1}, and a^{RT,f_2} (black arrows). Relating actions to features is necessary, as that allows the robot to deal with geometrical mismatches between the specified location description l and the ground truth. This holds in particular for non-instructed locations λ, that the route instructor is not even aware of. An example is observed in the bottom of Fig. 2, which shows two possible versions of metrically accurate representations of o_1 and o_3 in Fig. 1. In this case, the direction a^{RT,f_0} may be determined online by defining a vector that is equidistant to the corridor edges, i.e., f_0, outgoing from the location center.

This approach assumes an environment where the features f that are linked to the actions $a^{t,f}$ will be (partially) perceivable until the next location is reached, similar to navigation strategies in [35]. For example, corridors remain visible as two equidistant curves and a discontinuity in the perceived curves is detected whenever a new junction is reached.

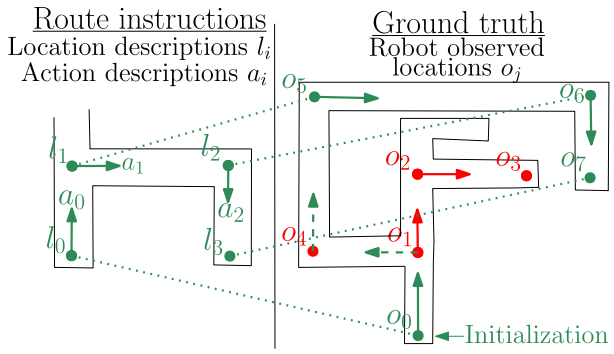


Fig. 3. Left: a map with a route instruction using locations l_i (green dots) and actions a_i (green arrows). Right: a different map with respect to Fig. 1 produced by the robot with observed locations o_j (green/red dots) and executed actions (green/red arrows). The intended location associations $o_j \sim l_i$ are visualized via green dotted lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1.2. Perceptual knowledge for error recovery

A robot needs to be aware of all the possible location types in the environment to configure its perceptual system to properly recognize any decision point. Not specifying these location types may lead to the robot to not recognize o_1 in Fig. 3, as that type of junction is not specified in the route instructions on the left. Consequently, the robot is not aware that it can backtrack to o_1 and explore in the direction of the green dashed arrow. This paper assumes that the route instructor and wayfinder are both aware of all the possible location types in the environment. Human wayfinding literature as [36] uses this assumption to recognize prematurely errors in wayfinding via communication of observed semantics from wayfinder to instructor.

Fig. 4 visualizes the location types that the robot can encounter while traversing forward. We have chosen them as they represent areas (i.e., decision points) where (1) the robot can change its direction of movement with respect to the direction it entered that area, or (2) the robot cannot continue its motion in the direction it entered the area.

We represent the world model $\mathcal{W}$ as a collection of location types and corresponding action models, e.g.,

$$\mathcal{W} = [\{\text{location_type} : l^{BT}, \text{action_model} : \mathcal{A}^{BT}\},$$
$$\{\text{location_type} : l^{RT}, \text{action_model} : \mathcal{A}^{RT}\}, \ldots],$$

which we represent for compactness throughout this paper as

$$\mathcal{W} = [l^{BT}, l^{RT}, \ldots].$$

A different environment requires a change in $\mathcal{W}$.

3.2. Knowledge to solve location ambiguity

A *first mechanism* with which a robot can recognize errors in its wayfinding, as a consequence of incorrect location associations, is the detection that it is *physically incapable* of further exploring its intended direction. An example is when the robot reaches o_3 in Fig. 3. A *second mechanism* to detect errors in wayfinding, works by imposing constraints on the topological map/route; detecting the violation of such constraints means the robot is *symbolically incapable* of further exploration. This second mechanism is necessary, as the presence of $\lambda_{i,k}$ in a large corridor-junction environment results in an "infinitely long" exploration into a specific direction. We guide the robot additionally by introducing three types of topological constraints combined in the constraint data structure $\delta_i = \{\sigma_i, X_i, \mathcal{T}_i\}$:

1. σ_i (Section 3.2.1): (minimum/maximum) number of non-instructed locations $\lambda_{i,k}$ between locations l_i and l_{i+1}.

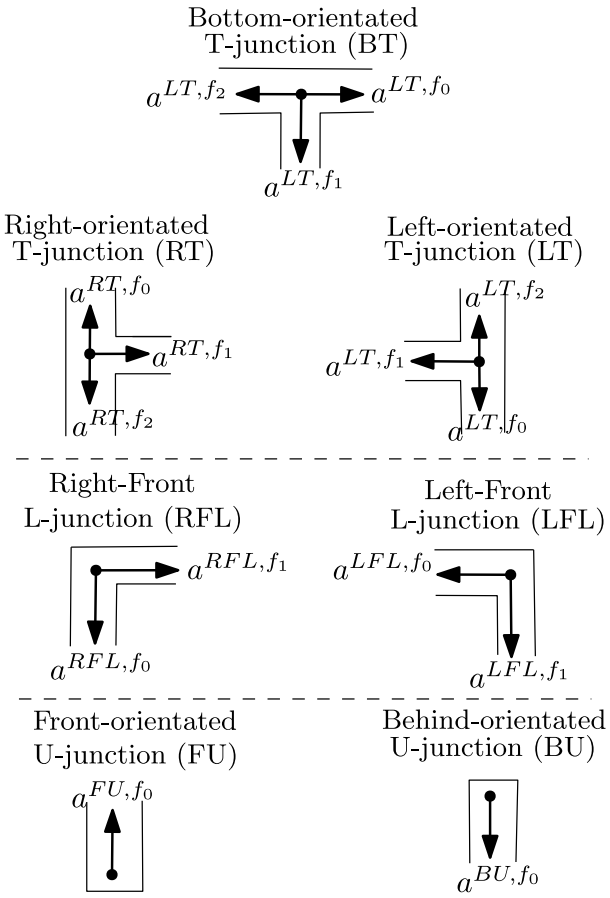


Fig. 4. A visualization of the location types from the robots perspective and corresponding $\mathcal{A}$ that are considered in this work. It assumes that the robot is positioned in the center (black dot) and is orientated with a heading similar to a^{RT,f_0} (left-top).

2. X_i (Section 3.2.2): location types l^t that cannot be encountered between l_i and l_{i+1}.
3. $\mathcal{T}_i$ (Section 3.2.3): geometric patterns over the locations l_i.

In other words, our approach relies on the following assumption:

Assumption 4 (*Correct Topological Constraints* δ_i)**.** associations between $o_j \sim l_i$ satisfy all specified constraints in $\delta_i \; \forall \; i$.

3.2.1. Number of non-instructed locations

A route instructor may not always know the exact *type* of locations l^t, but may be aware of the *number* of locations that the wayfinder should encounter. For example, an instruction such as "between decision points d_0 and d_1 you will observe at most two non-instructed locations $\lambda_{0,k}$ of which I do not know which l^t is applied". We represent this knowledge in σ_i which consists of two components: (1) a numeric value (e.g., "One", "Two", or "Unknown", related to the number k of $\lambda_{i,k}$), and (2) an operator ("minimum", "equal", or "maximum"). The former instruction can then be represented as

$$\sigma_0 = \{\text{Two, maximum}\} \tag{2}$$

suggesting that we should at most encounter $\lambda_{0,0}$ and $\lambda_{0,1}$.

Fig. 5 shows a route instruction (left) and the ground truth (right). Considering the route topology constraint σ_0 above, we have three valid route interpretations:

1. $o_0 \sim l_0 \rightarrow o_1 \sim l_1$.
2. $o_0 \sim l_0 \rightarrow o_1 \sim \lambda_{0,0} \rightarrow o_6 \sim l_1$.

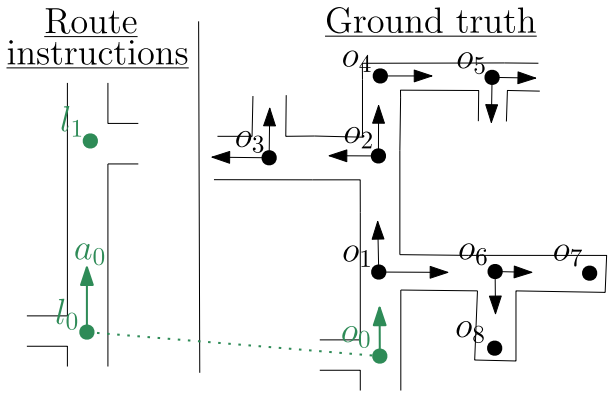


Fig. 5. Left: route instructed topology with green dots and arrows representing the locations l_i and actions a_i. Right: the ground truth where the robot starts at $o_0 \sim l_0$ with the green dot representing a correct association and the green arrow a correct action execution. The black dots and arrows represent all other non-explored locations o_j and corresponding actions that direct the robot into non-explored areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3. $o_0 \sim l_0 \rightarrow o_1 \sim \lambda_{0,0} \rightarrow o_2 \sim \lambda_{0,1} \rightarrow o_3 \sim l_1$.

Note that the route $o_0 \sim l_0 \rightarrow o_1 \sim \lambda_{0,0} \rightarrow o_2 \sim \lambda_{0,1} \rightarrow o_4 \sim \lambda_{0,2} \rightarrow o_5 \sim l_1$ violates σ_0; the exploration in this direction would stop whenever the robot reaches o_4, thus showing that σ_0 can be used to stop exploration into a specific direction even though it is physically possible.

3.2.2. Restricted location types

The second type of constraint preempts a robot's exploration into a specific direction by instructing location types that the robot should *not* encounter along its way; i.e., *true negatives* are provided explicitly to the robot. For example an instruction: "if you observe a location of type Left-orientated T-junction (LT) on the way to l_1 from l_0, then you have gone too far". We incorporate this knowledge into X_i, where the former instruction can be represented as

$$X_0 = [l^{LT}]. \tag{3}$$

If we consider Eqs. (2) and (3) for the example in Fig. 5, then the third route interpretation in the enumeration above is invalidated as o_2 is of type LT.

3.2.3. Shapes in the topological map

In case a topological graph can be maintained with quantitative values in the edges, i.e., a topo-metric graph [24], then one can add additional constraints on the shape of a topo-metric graph. We will consider the route instructions on the left of Fig. 3 with no shape constraints on the topo-metric graph, i.e., $\mathcal{T}_i = [\,]$, where we specify the route instruction according to (1) as:

$$\begin{aligned} RI = \{&[l^{BT}, l^{LT}, l^{RT}, l^{RFL}, l^{LFL}, l^{FU}, l^{BU}], \\ &[(a_0^{FU,f_0}, l_0^{FU}, \{\{\text{Unknown, equal}\}, [], []\}), \\ &(a_1^{RT,f_1}, l_1^{RT}, \{\{\text{Zero, equal}\}, [], []\}), \\ &(a_2^{RFL,f_1}, l_2^{RFL}, \{\{\text{Zero, equal}\}, [], []\}), \\ &(\text{none}, l_3^{BU}, \{\{\text{Zero, equal}\}, [], []\})]\}. \end{aligned} \tag{4}$$

where the action a_3 at the goal is not applicable and hence defaults to "none".

Fig. 6 visualizes the ground truth on the left and on the right two solutions, diverging at o_1, that satisfy the route instructions of Eq. (4). Constraints on the topo-metric graph needs to be introduced via $\mathcal{T}$ to obtain the route instructor's intended solution. $\mathcal{T}$ consists of topo-metric patterns $\tau_c = \{\mathcal{E}_c, \mathcal{R}_c\}$ where the subscript c is a counter. The

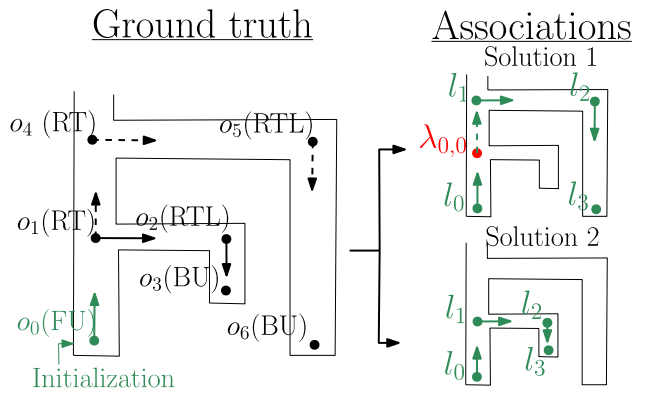


Fig. 6. Left: Ground truth with two routes visualized that satisfy the route instructions of Eq. (4). Solution 1 follows the dashed black arrows, and Solution 2 follows the solid black arrows. Right: the associations visualized of the two solutions that satisfy the route instructions.

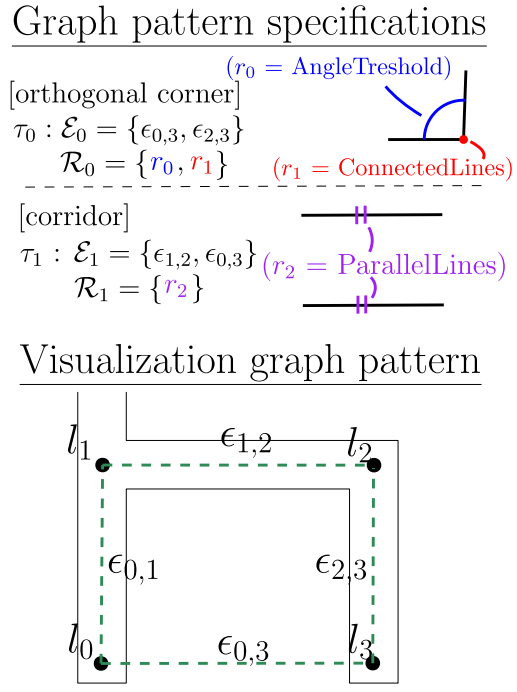


Fig. 7. Top: the topo-metric graph pattern τ_c specifications with a visualization of the individual pattern and the involved relations with its color indicating the related parts of the pattern. Bottom: visualization of a topo-metric graph that satisfies the above constraints τ_0 and τ_1.

topo-metric pattern is a collection of topological edges $\mathcal{E}$ and edge-edge (spatial) relations $\mathcal{R}$. We describe an edge between two route instructed locations l_a and l_b as $\epsilon_{a,b}$ with $a \neq b$. Fig. 7 shows at the top two examples of topo-metric patterns that can be introduced onto the route instructions above. The virtual corner pattern is introduced as τ_0 and a virtual corridor pattern as τ_1. We formalize which edges $\epsilon_{a,b}$ participate in those pattern descriptions and also which relations r needs to be checked. The topo-metric patterns τ_c that contain an edge $\epsilon_{a,b}$ with $a == i$ or $b == i$ get included into $\mathcal{T}_i$. In this case, $\mathcal{T}_0 = \mathcal{T}_2 = \mathcal{T}_3 = [\tau_0, \tau_1]$ and $\mathcal{T}_1 = [\tau_1]$. Adding those topo-metric constraints into the route instructions of Eq. (4) leads to the conclusion that only the first solution in Fig. 6 is correct. The design of those relations $r \in \mathcal{R}$ is out-of-scope in this work.

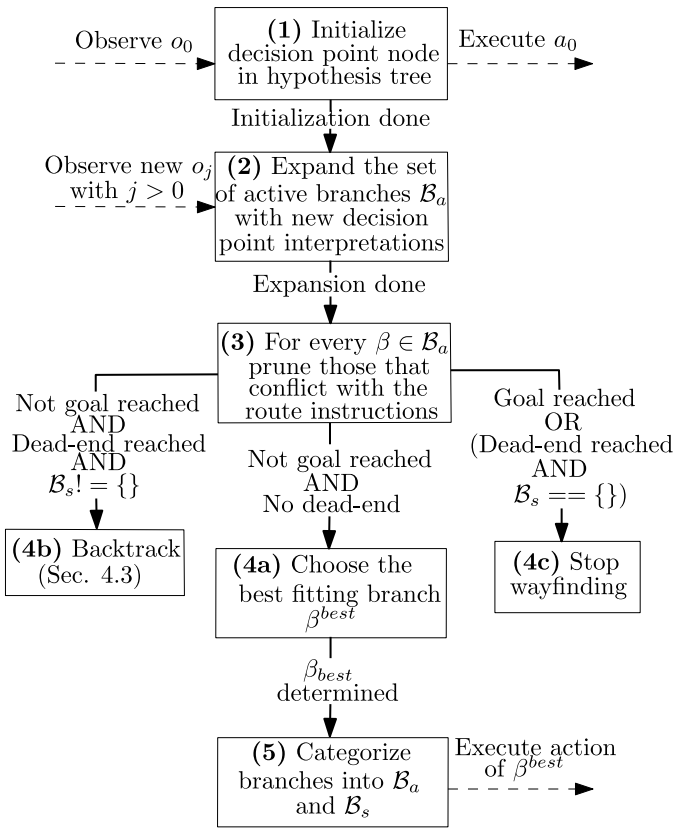


Fig. 8. A behavior tree visualizing the core logic of the defensive wayfinding procedure. The dashed ingoing arrows represent input from robot perception, and dashed outgoing arrows the output to robot actuation. Each block represents a robot command that needs to be executed with a number in brackets indicating its step in the sequence. The solid arrows indicate the transitions between commands. $\mathcal{B}_a$ represents the set of active complete path hypotheses, and $\mathcal{B}_s$ the set of standby complete path hypotheses.

4. Defensive wayfinding procedure

This Section explains the defensive wayfinding procedure with the route instructions of (1) as input. The robot successfully accomplishes defensive wayfinding if it can determine a complete path:

Definition 1 (*Complete Path*). A sequence of executed actions "α" and observed locations "o" that would have been realized if the route instructor did not omit any decision points, i.e., the intended path. It does not involve the executed actions and observed locations that are recorded during backtracking/erroneous wayfinding.

Any time the robot encounters a new decision point, a decision point hypothesis η is initiated as a node in the multiple hypothesis tree. That node stores (1) a hypothesized location association $o_j \sim l_i$ or $o_j \sim \lambda_{i,k}$, and (2) a suggested action $\alpha_j = a^{i,f}$ to execute in order to follow the route instruction. A complete path hypothesis β is then represented as one particular sequence of such nodes. Fig. 9 visualizes a multiple hypothesis tree with three branches on the right each representing a complete path hypothesis. On the left, the online wayfinding is visualized where an update is provided each time the robot makes a new observation o_j.

This paper implements *defensive wayfinding* as the procedure by which the robot physically explores and updates the best fitting complete path hypothesis β^{best}, until it reaches its destination, or the path becomes inconsistent with the route instruction. In the latter situation, the robot will take the decision to consider a different non-conflicting

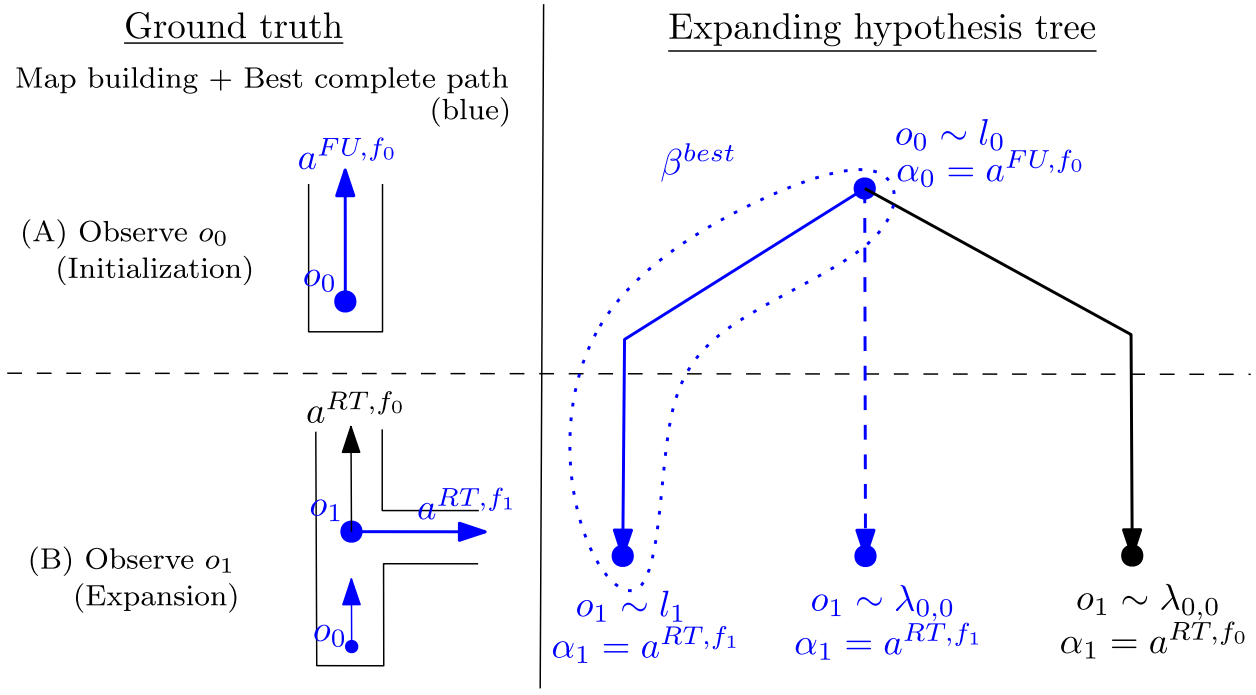


Fig. 9. Left: map building during defensive wayfinding where the robot starts at o_0 of the ground truth in Fig. 1. the blue dots and arrows represent the location association and action execution according to the hypothesis branch β^{best} visualized on the right. The black arrow represents an unexplored action. Right: the hypothesis tree where each node is the set of location association $o_j \sim l_i$ or $o_j \sim \lambda_{i,k}$ and the executed action α. The black branches are put on "standby" and the blue branches are those actively tracked and expanded where the solid arrow represents β^{best}. The hypothesis tree expands whenever a new location is encountered on the left. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

complete path hypothesis as β^{best} and iterates the procedure. Assumptions 2, 3, and 4 limit the number of hypotheses. The contribution of this section is a proof of concept of a hypothesis tree containing complete paths, and of the structured defensive wayfinding procedure that is realized with this hypothesis tree of which the core decision making logic is visualized in the behavior tree in Fig. 8; Appendix B provides more detailed pseudo-code for implementation insight. Throughout this section, we will explain the behavior tree procedure with the route instructions Eq. (4) and ground truth in Fig. 1.

4.1. Initialization and expansion of the hypothesis tree (steps 1–3)

Fig. 8 step (1) initializes a node in the multiple hypothesis tree. According to Ass. 1 and Ass. 3, that node should be formulated as $\eta_0 = \{o_0 \sim l_0^{FU}, \alpha_0 = a^{FU,f_0}\}$ for the robot starting in Fig. 9 (A) on the top left.

The robot executes that suggested action α_0 until it observes o_1 in Fig. 9 (B) on the bottom left. Fig. 8 step (2) expands the hypothesis tree with all possible decision point hypotheses. For now, it is not necessary to understand $\mathcal{B}_a$; a detailed explanation is provided in Section 4.3. One hypothesis is that l_1 is encountered as o_1 satisfies l_1^{RT}, and the node $\{o_1 \sim l_1, \alpha_1 = a^{RT,f_1}\}$ is created. Other hypotheses are that a non-instructed location $\lambda_{0,0}$ is encountered at which two actions can be performed, thus creating nodes $\{o_1 \sim \lambda_{0,0}, \alpha_1 = a^{RT,f_0}\}$ and $\{o_1 \sim \lambda_{0,0}, \alpha_1 = a^{RT,f_1}\}$. Fig. 8 step (3) then checks whether the branches are valid by verifying that they are non-conflicting with the route instruction. In this particular example, all three branches are valid interpretations of the route.

4.2. Best fitting complete path (step 4a)

The procedure continues with Fig. 8 step (4b) as the goal has not been reached and further exploration is possible. The robot has two options to continue its route after o_1, namely to continue in the direction of a^{RT,f_0} or a^{RT,f_1}. Our strategy is to select one of the complete path hypotheses as the best fitting branch β^{best} and to execute the action of its leaf node. The selection criteria for β^{best} is as follows:

- choose β that has the most route instructed location associations. This may suggest a quicker task completion.
- in case multiple β's exist that have the most route instructed location associations, and all β's associate the last observed o_j to λ, then continue in the direction of the previous route instructed action a_i. We assume that the route instructor did not specify the decision point as no directional change was required [6,7].
- in case multiple β's exist that have the most route instructed location associations, and at least one β associates the last observed o_j to l_i, then choose β with (1) the earliest route instructed location association l_i, and (2) an action choice in the direction of a_i.

This reasoning is sufficient to illustrate our wayfinding procedure, and can be extended in the future with other types of reasoning using richer semantics than the strictly topological interpretations in this paper; e.g., computer vision perception of "landmarks" of a particular type such as traffic signs or buildings. In this case, β^{best} is assigned to the left-most branch of Fig. 9 as it has the most route instructed location associations. Examples of the result of the two other selection criteria are treated in Appendix D.

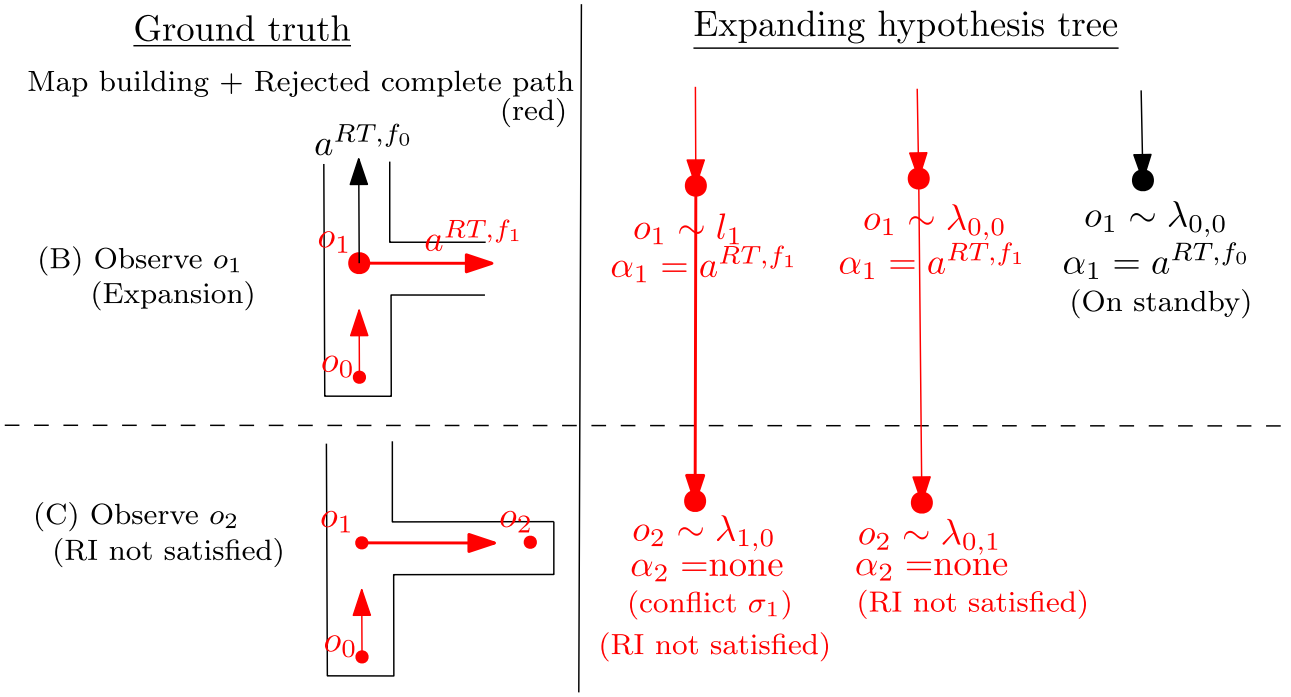


Fig. 10. Left: map building during wayfinding procedure where the robot continues by turning right at o_1 of Fig. 9. The arrows and nodes that are colored red represents a wrong complete path hypothesis. The black arrow represents an unexplored action. Right: the hypothesis tree where the red branches represent dead branches that either conflict with δ_i or do not fully satisfy the route instructions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3. Categorization of branches (step 5)

Fig. 8 step (5) then performs bookkeeping to avoid expanding branches that require an exploration direction that do not align with β^{best}. A distinction is made between the set of active branches $\mathcal{B}_a$ and the set of standby branches $\mathcal{B}_s$. The middle branch of Fig. 9 with the blue dashed arrow is an example of an active branch, while the right-most branch is an example of a standby branch. Active branches can be used to backtrack *virtually*, i.e., on the *map* of the world, in case β^{best} is rejected, whereas standby branches require *physical* backtracking.

Definition 2 (*Active Branches* $\mathcal{B}_a$)**.** set of complete path hypotheses that require a similar exploration direction as β^{best}, i.e., the same action choice when leaving the last observed location.

Definition 3 (*Standby Branches* $\mathcal{B}_s$)**.** set of complete path hypotheses that require a different exploration direction than β^{best}.

The robot will then execute the suggested action of β^{best}, and does so until it encounters a new o_j to resume at step (2) in Fig. 8. That iteration continues until a complete path candidate for the route instruction is found, i.e., the goal is reached, or no further exploration is deemed feasible, ending the algorithm at step (4c). An additional example of our wayfinding procedure is treated in Appendix D; It shows that the number of β branches can grow extensively and that an unique solution may not exist in case δ_i is unspecified, even for simple environments.

4.4. Rejecting β^{best} and physical backtracking (step 4b)

Fig. 10 (C) on the bottom left visualizes the situation where the robot encounters a dead-end at o_2. We thus reject on the right of Fig. 10 the previously active tracked branches, now highlighted in red representing so-called dead-branches $\mathcal{B}^d$.

Definition 4 (*Dead Branches* $\mathcal{B}_d$)**.** set of complete path hypotheses that either conflict with δ_i or are fully explored and do not satisfy the route instructions.

Step (4b) of Fig. 8 is called as $\mathcal{B}_s$ is not empty. In this particular case, the right-most standby branch remains to backtrack to o_1 and to execute action a^{RT,f_0}. In case multiple standby branches would be present, then we instruct the robot to backtrack to the "closest" one, i.e., the branch β^{most} with the highest number of o_j. The physical backtracking procedure is explained in more detail in Appendix C. After the robot has backtracked successfully and encountered a new observed location o_j, the wayfinding procedure resumes at step (2) in Fig. 8.

5. Experimental validation

This section shows the feasibility of expanding the hypothesis tree of Section 4 in an online fashion by connecting it to continuous robot motion and perception components. Additionally, a connection is made to a SLAM algorithm for map visualization purposes. The experiments are performed in a gazebo[2] simulation.

[2] http://gazebosim.org/

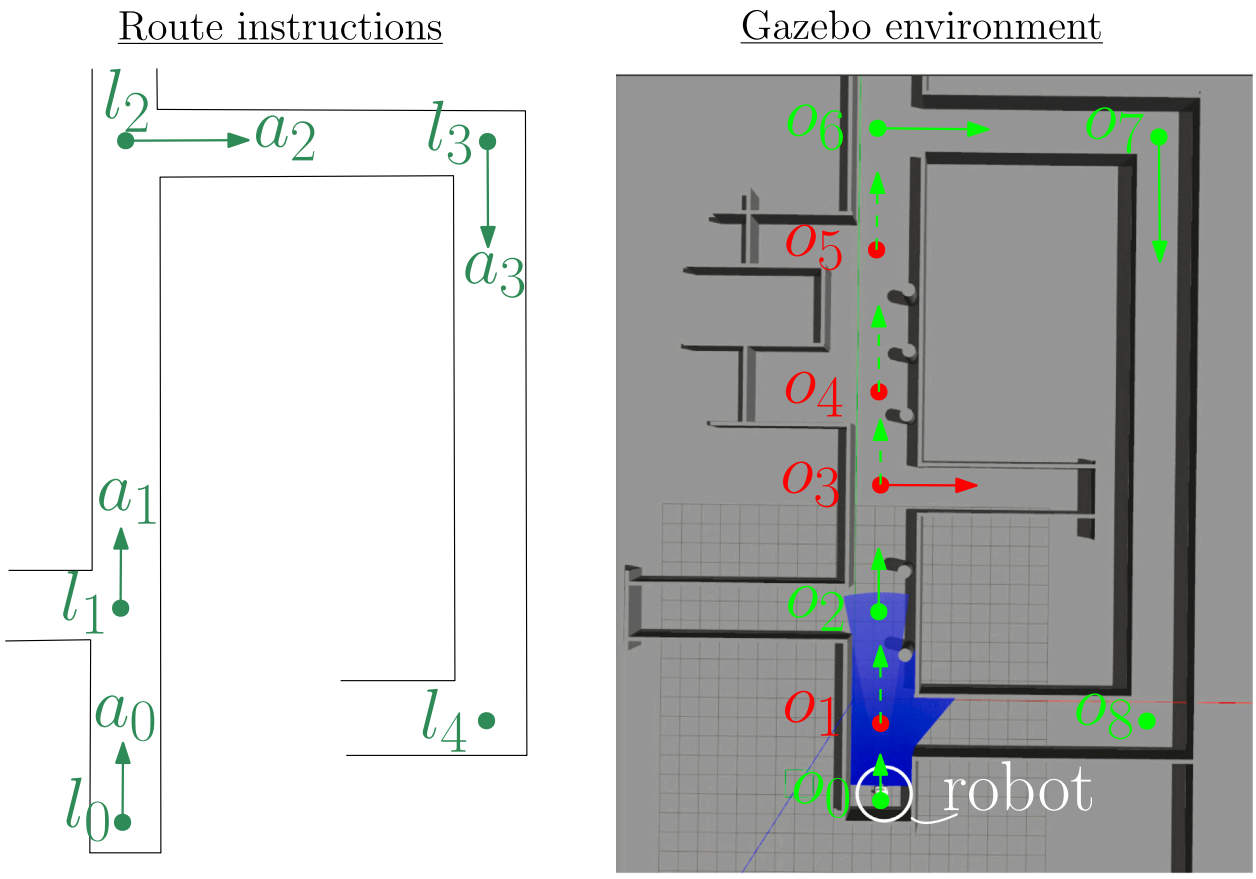


Fig. 11. Left: the route instructions with the green dots representing l_i and the green arrows a_i. Right: the gazebo environment with the white circle representing the robot and the blue area representing the laserscans from the 2D laser range finder. The complete path is visualized with the green dots representing the associatable locations and the red dots the non-instructed locations. The green arrows represent the intended action executions where the dashed arrows represent a non-instructed action. The red arrow at o_3 represents a wrong action execution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

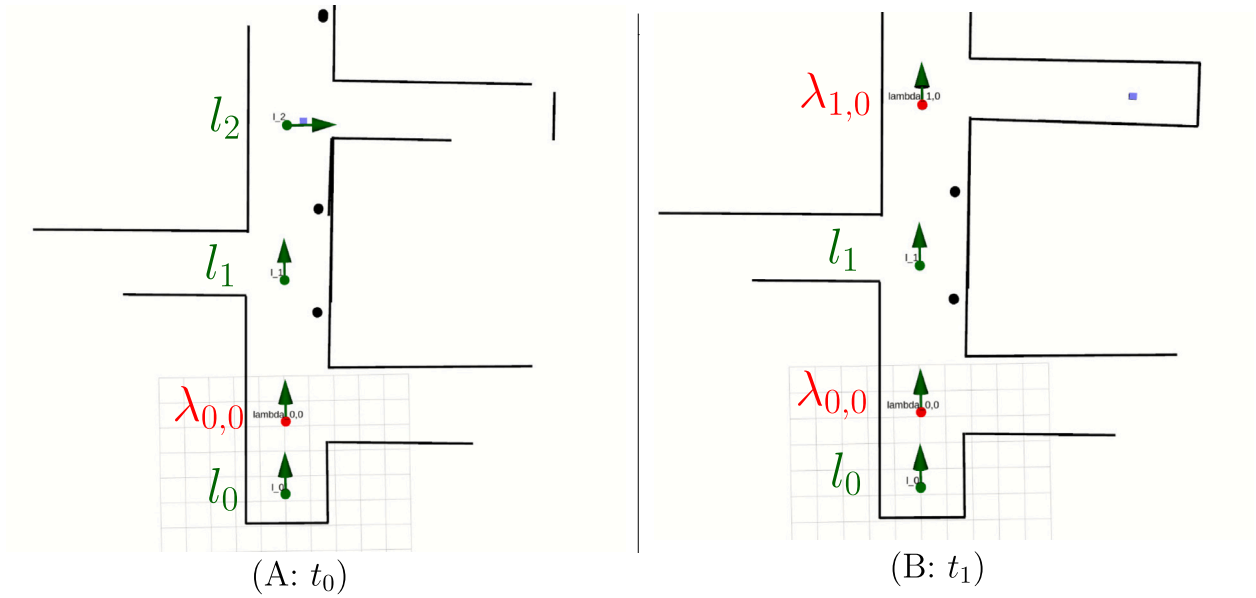


Fig. 12. Left: RVIZ map created at t_0 when the robot has traversed $o_0 \rightarrow o_1 \rightarrow o_2 \rightarrow o_3$ and made the association $o_3 \sim l_2$. Right: RVIZ map at t_1 ($> t_0$) where the robot concluded that o_3 is not l_2 and needs to backtrack to o_3.

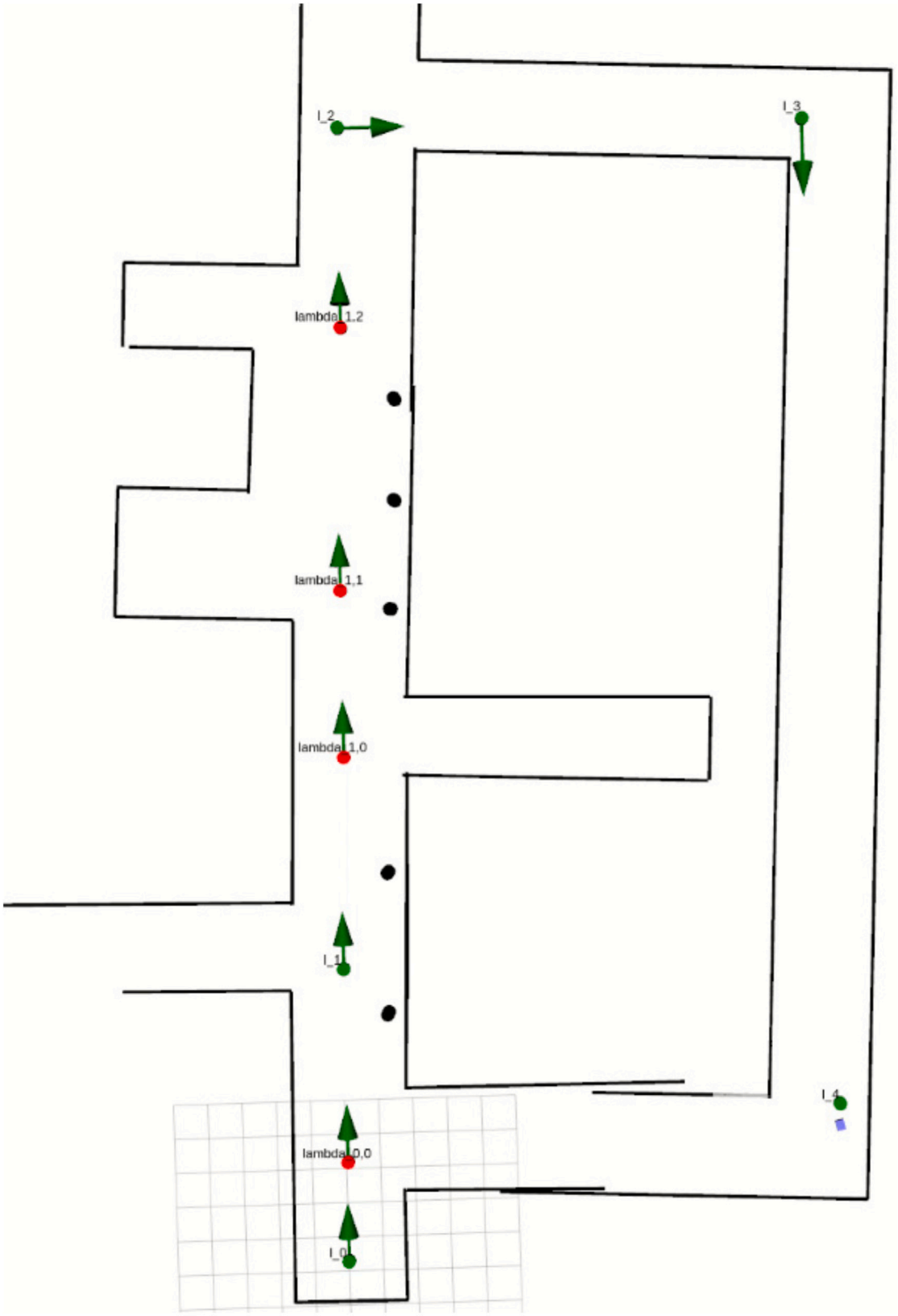


Fig. 13. A visualization of the real-time created map in RVIZ after obtaining the complete path. The black lines represent the walls and the circles the pillars. Each dot is labeled with an instructed location l_i or a non-instructed location $\lambda_{i,k}$. The green arrows are the executed actions. The purple square at l_4 represents the robot footprint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1. Experimental setting

Fig. 11 depicts the Gazebo environment on the right and the route instructed map on the left. The route instructions are formally represented as:

$$\begin{aligned} RI = \{[&l^{BT}, l^{LT}, l^{RT}, l^{RFL}, l^{LFL}, l^{FU}, l^{BU}], \\ &[(a_0^{FU,f_0}, l_0^{FU}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(a_1^{LT,f_2}, l_1^{LT}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(a_2^{RT,f_1}, l_2^{RT}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(a_3^{RFL,f_1}, l_3^{RFL}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(\text{none}, l_4^{RFL}, \{\{\text{Unknown}, \text{equal}\}, [], []\})]\} \end{aligned} \tag{5}$$

The robot is initially positioned at the bottom of the gazebo environment indicated with the white circle. Its orientation is "upward" (green arrow). A holonomic robot is considered with a 2D laser range finder sensing with a cone of 180 degrees in front, visualized as the blue beams in the gazebo environment. The complete path is visualized on top of the gazebo environment where the green dots represent observed locations o_j that are correctly associated to l_i, and the red dots those that are associated to non-instructed locations $\lambda_{i,k}$. The used software components are: *Robot Operating System (ROS) Melodic*,[3] the package of [37][4] to create a map with SLAM, and RVIZ[5] to visualize the created map. The package of [37] also extracts lines and circles which are used to verify the junction patterns of the location types in Fig. 4. That allows us to extract corridor features to generate online the corresponding action directions $a^{t,f}$. A simple feedforward motion controller is designed that aligns the robot to the action direction that needs to be executed until it encounters a new junction.

5.2. Results

To show the applicability of our approach in an online fashion, we visualize the map[6] obtained from SLAM in RVIZ on which β^{best} is superimposed with the location associations l_i or $\lambda_{i,k}$. The end-result with β^{best} is visualized in Fig. 13, where the black lines and circles represent the walls and pillars in the gazebo environment on the right of Fig. 11. During wayfinding, β^{best} changed as the robot made a wrong association at o_3, visualized on the left in Fig. 12. The robot turns right at o_3 according to the red arrow in Fig. 11. Only when it has reached the dead-end, the robot concludes that the association $o_3 \sim l_2$ is incorrect, and changes β^{best} online to the one visualized on the right of Fig. 12. The robot then backtracks to o_3 and follows the instructed arrow to go "upwards".

A structured exploration strategy is devised such that the robot does not need to explore the whole map to realize a complete path, as visualized by the lack of unexplored areas to the "left" of l_1 and "upwards" of l_2 in Fig. 13. Additionally the map can be used as feedback to revise the instructions, e.g., by observing a possible connection between $\lambda_{0,0}$ and l_4.

6. Discussion and conclusions

6.1. Reasoning about omitted decision points

Robot defensive wayfinding benefits from explicit knowledge about *why* mismatches occur between the route instruction and the ground truth. In this paper, a generic defensive wayfinding algorithm was incorporated that can deal with omitted decision points. We have not provided the robot explicit knowledge of all the possible types of reasoning behind that omission. That knowledge may allow the robot to expand and prune the hypothesis tree in a manner consistent with human reasoning, in turn allowing human route instructors to provide even more concise route instructions.

For example, the presence of an unique spatial structure in the environment, i.e., *structure chunking* [6], allows the route instructor to specify instructions as "turn left at unique feature". Another example, is that humans understand the semantics in the world and their implications on restricting the wayfinder's decision making at a decision point. The traffic system [38] is a great example, where semantics can be present for coordination, e.g., no-enter street signs or bus lanes, but also to infer the likely destination, e.g., following a sign towards Paris would not make sense if the destination is in Germany; That does not have to be made explicit by the route instructor, but should be inferred by the robot itself. That reasoning can then also help in dealing with erroneous actions [20] or overspecified location descriptions [2], e.g., a four-way crossing is represented as a five-way crossing. Future work will extend on testing such semantic reasoning for more semantically rich environments.

We think that following human route instructions is difficult for robots, because the robot lacks the knowledge to understand *why* the abstractions are made in a route instruction. Making that knowledge explicit is not trivial, as each human may reason and understand the world in a different manner, thus leading to different abstractions. It is also why efforts to make that abstraction explicit as in [7] from a non-robotic domain benefit robot wayfinding.

6.2. Less artificial environments

The environments in this paper are visualized in their most abstract representation using a Manhattan model, in which corridors are aligned with orthogonal axes. We hypothesize that less artificial environments, e.g., in which corridors are not aligned to orthogonal axes, do not reduce the applicability of our defensive wayfinding procedure as it mainly focuses on the topological structure. Instead, these environments demand more advanced robot motion control and perception capabilities by specifying more detailed action models (Section 3.1.1).

One of the challenges in formulating these action models is that the perceptual features in the real world may not persist over time, in contrast to the contours of the corridors treated in this paper. Consequently, it becomes ambiguous to how "long" that action needs to be executed. In [7], the authors conclude that sketch maps often draw perceptual features just to "show that they are present". For example, a path may be drawn that passes two houses, while in reality there could be twenty houses. Some reasoning needs to be present to indicate that the "robot is still progressing correctly to the next decision point", even though the two sketched houses have been passed. One could reason even one step further that the omission of "houses" for a reasonable time could indicate that the robot has gone too far, or that drawing many houses indicates that a possible decision point after the first house should be ignored. Future work will extend on such robot navigation with sparse instructed features in-between decision points.

6.3. Algorithm improvement

We have not addressed the occurrence of loop closures. Suppose the robot would have turned right at o_1 immediately towards o_8 in Fig. 11. A situation may occur where the robot would execute a path via the sequence $o_1 \rightarrow o_8 \rightarrow o_7 \rightarrow o_6 \rightarrow o_5 \rightarrow \cdots \rightarrow o_1$, thus revisiting location o_1. The current algorithm does not tackle this case, that is to be resolved by adding an additional step in Alg. 1: create additional hypothesis

[3] http://wiki.ros.org/melodic
[4] https://gitlab.tue.nl/s135700/semantic_graph_localization
[5] http://wiki.ros.org/rviz
[6] see supplementary material 'video_0' to see online update

Table A.1
Table of symbols and definitions.

Symbol	Definition
RI	Route Instruction, formulated as $\{\mathcal{W}, [(a_i^{t,f}, l_i^t, \delta_i), \dots, (a_{n-1}^{t,f}, l_{n-1}^t, \delta_{n-1})]\}$ with n the total number of instructions.
i	The subscript index representing the i'th decision point in the route instruction.
t	Type
f	Geometric feature
d	Decision point
l^t	Location description corresponding to a decision point. Adding a subscript i indicates that the description belongs to d_i.
$\mathcal{A}^t$	All the possible motion control actions to exit a location type.
$a^{t,f}$	Action specification corresponding to a decision point. The motion control to exit a location type via its feature. A subscript i indicates that the description belongs to d_i.
$\mathcal{W}$	The world model, specifying all the encounterable location types l^t and corresponding action models $\mathcal{A}^t$.
δ_i	Formulated as $\{\sigma_i, X_i, \mathcal{T}_i\}$, representing the topological constraints that are imposed on the topological (or topo-metric) graph obtained during online robot wayfinding. It incorporates those constraints that are related to l_i or l_{i+1}.
σ_i	Formulated as $\{\#, \text{operator}\}$, indicating the number k of non-instructed locations $\lambda_{i,k}$ between l_i and l_{i+1}.
X_i	Formulated as a collection of l^t that cannot be encountered between d_i and d_{i+1}.
$\mathcal{T}_i$	Formulated as a collection of topo-metric graph patterns τ_c which involve d_i.
τ_c	Formulated as $\{\mathcal{E}_c, \mathcal{R}_c\}$ representing, respectively, the participating topo-metric graph edges and the relations that needs to be verified between those edges with $c \in \mathbb{N}$.
$\mathcal{E}_c$	A collection of topo-metric graph edges.
$\epsilon_{a,b}$	The edge in the topo-metric graph from location descriptions l_a and l_b with $a, b \in \mathbb{N}$ and $a \neq b$.
$\mathcal{R}_c$	A set of relations r involving two or more top-metric graph edges.
r	A description of a constraint that needs to be checked between two topo-metric graph edges, e.g., that they should be parallel to each other.
j	The subscript index representing the j'th observed location from online robot wayfinding.
o_j	The observation of the j'th location that is recognized during online wayfinding.
α_j	The $a^{t,f}$ that is executed during online wayfinding to exit o_j.
$\lambda_{i,k}$	A non-instructed location between instructed locations l_i and l_{i+1} where $k \in \mathbb{N}$ is a counter of its occurrence between those instructed locations.
η	The hypothesized decision point node $\{o_j \sim l_i, \alpha_j = a^{t,f}\}$ with, respectively, the location association and the executed action at o_j.
β	A complete path hypothesis formulated as a sequence of decision point hypotheses nodes.
β^{best}	The best fitting complete path hypothesis with regards to the route instructions.
β^{most}	The complete path hypothesis with the most decision point nodes.
$\mathcal{B}_a$	The collection of β's that require a similar exploration direction as β^{best}.
$\mathcal{B}_s$	The collection of β's that require a different exploration direction than β^{best}.
$\mathcal{B}_d$	The collection of β's that either conflicts with the topological constraints δ_i of are fully explored and do not satisfy the route instructions.

branches from the loop-closed location while imagining that the robot took a different action. For example, a branch could be maintained from o_1 where it is imagined that it went straightforward in the direction of o_2 while incorporating the locations encountered in the loop closure. A necessity is the ability of the robot to identify loop-closures. But as loop-closures can be uncertain, this would similarly increase the number of hypotheses that needs to be maintained. Future work will extend on treating environments for which uncertain route instructions will lead to loop-closures.

Additionally, the hypothesis tree expansion occurs in a deterministic fashion. A probabilistic approach may be taken in case sufficient data is gathered about the likeliness of omitting a decision point in some specific scenario. That data can be obtained from human sketch map literature, such as [6,7], or by designing an experiment where human participants provide route instructions to a mobile robot. An interesting case is to observe whether the human would provide route instructions at the same level of abstraction as they would for another human. More specifically, to determine the relation between "route instruction abstraction" and "knowledge about robot capabilities".

CRediT authorship contribution statement

Hao Liang Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Elena Torta:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Herman Bruyninckx:** Writing – review & editing, Supervision, Methodology, Conceptualization. **René van de Molengraft:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Hao Liang Chen reports financial support was provided by TKI HTSM (NL). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work received co-funding from the TKI program of the Dutch government, The Netherlands and was performed under the Frontiers in Autonomous Systems Technology (FAST) project. The authors want to thank the reviewers for providing new insights and concise, yet constructive, feedback that has led to a significant improvement of the paper.

Appendix A. Table of symbols and definitions

See Table A.1.

Appendix B. Pseudo-code for implementation

Algorithm 1 Pseudo-code of Fig. 8

```
1:  procedure DEFENSIVE WAYFINDING(RI)
2:      B_a =[ ], B_s = [], B_d = []
3:      o_0 = DetectInitialLocation(RI)
4:      β^best = InitializeTree(o_0, RI)
5:      ExecuteAction(β^best)
6:      B_a ← β^best
7:      while CompletePathNotFound do
8:          if DetectNewLocation(W) then
9:              o_j = DetectNewLocation(W)
10:             B_a^expanded = ExpandTree(o_j, RI, B_a)
11:             for β ∈ B_a^expanded do
12:                 if (Conflict with δ_i) or (dead-end and RI not satisfied)
    then
13:                     B_d ← β
14:                     Remove β from B_a^expanded
15:                 end if
16:             end for
17:             if B_a^expanded ≠ [ ] then
18:                 if β ∈ B_a^expanded satisfies RI then
19:                     Stop procedure; plausible solution found
20:                 else
21:                     β^best = DetermineBestFit(B_a^expanded)
22:                     ExecuteAction(β^best)
23:                     (B_a, B_s) = SplitBranches(β^best, B_a^expanded, B_s)
24:                 end if
25:             else
26:                 if B_s ≠ [] then
27:                     Choose most explored β^most from B_s
28:                     Remove β^most from B_s
29:                     Add β^most to B_a
30:                     Backtrack to last recorded o_j of β^most
31:                     ExecuteAction(β^most)
32:                 else
33:                     Stop procedure; no solution found
34:                 end if
35:             end if
36:         end if
37:     end while
38: end procedure
```

Step (1) in Fig. 8 occurs within lines 3–6 of Alg. 1. A perception component `DetectInitialLocation(RI)` verifies that the location type of l_0^t is indeed perceived (line 3). Subsequently, the hypothesis tree is initialized via `InitializeTree()` (line 4). Currently, there is only one complete path hypothesis consisting of the initialized node, which represents β^{best}. The robot then executes the action specified in the leaf node of β^{best} with `ExecuteAction()` (line 5). That β^{best} is then added to the set of active branches $\mathcal{B}_a$ (line 6). The algorithm runs iteratively within a while-loop as long as (1) the complete path is not found (`Complete PathNotFound` line 7) or, (2) when either $\mathcal{B}_a$ or $\mathcal{B}_s$ is non-empty (lines 17 and 26).

Step (2) in Fig. 8 occurs within lines 8–10. The robot queries whether a location is observed according to one of the types specified in $\mathcal{W}$ via `DetectNewLocation()` (line 8). If so, then a new observed location o_j is initiated (line 9). We then expand the $\mathcal{B}_a$ into $\mathcal{B}_a^{expanded}$ by adding new decision point hypotheses η (line 10).

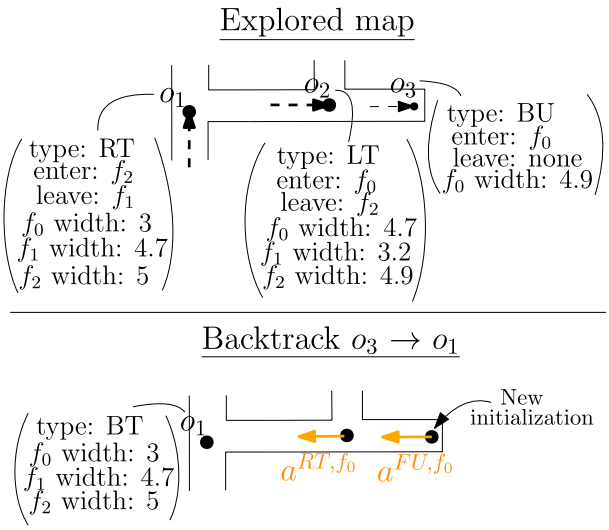


Fig. 14. Top: a robot explored from o_1 to o_3 where the backtracking related properties are mentioned in brackets. The dashed black arrow represent the direction in which the junction is entered. Bottom: the backtrack route instructions where the orange arrows represent the backtrack actions and each backtrack location contains a new perception type with signature properties, e.g., o_1 will be recognized as type BT instead of RT. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

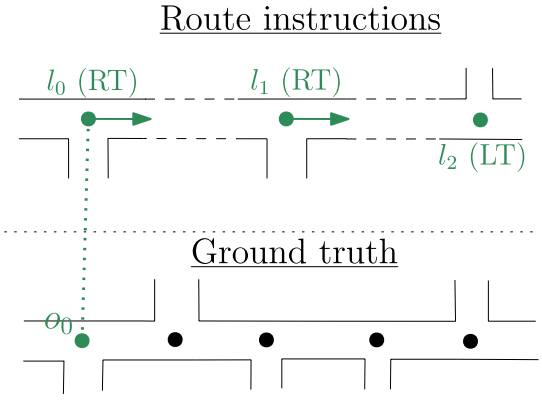


Fig. 15. Top: the route instructions where the dashed corridors between locations represent an uncertainty of the number of non-instructed locations λ. Bottom: the ground truth where the green dot is correctly associated to the route instructed location and the black dots represent the unexplored locations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

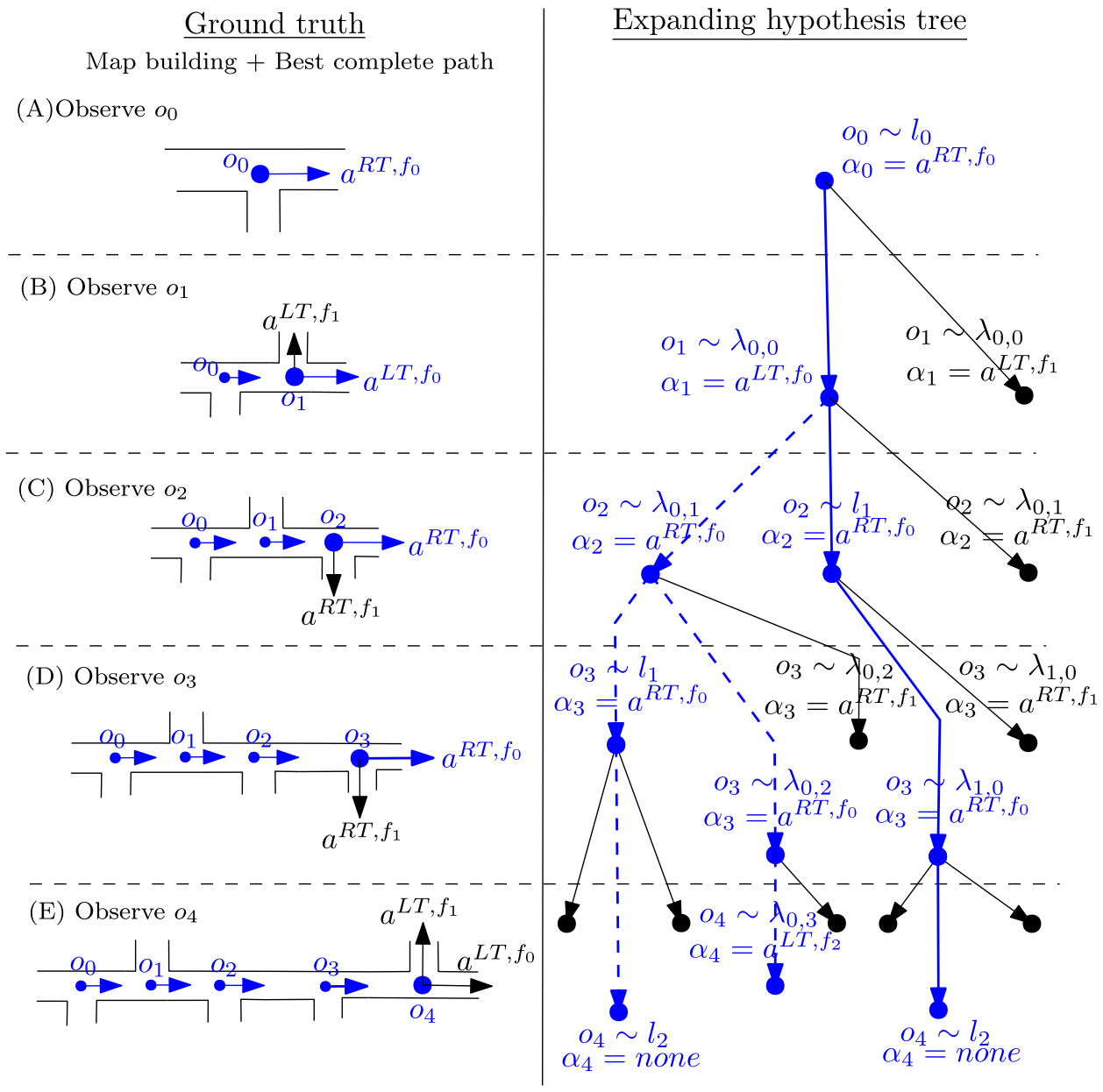


Fig. 16. Left: map building during wayfinding procedure where the robot starts at o_0 of Fig. 15 and follows the route instruction in Eq. (6). Right: the hypothesis tree showing all the possible active (blue) and standby (black) hypotheses. The solid blue line represents β^{best} and the dashed line an alternative explanation that requires a similar direction exploration as β^{best}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Step (3) in Fig. 8 occurs within lines 11–16. For every complete path hypothesis in $\mathcal{B}_a^{expanded}$ (line 11), we check whether it conflicts with the route instructions and/or a dead-end is reached (line 12). If so, then we remove the analyzed complete path hypothesis from $\mathcal{B}_a^{expanded}$ and store it into $\mathcal{B}_d$ (lines 13–14).

Step (4a) in Fig. 8 occurs within line 21 where `Determine Best-Fit()` is specified further in Alg. 2.

- (lines 2–3): choose β that has the most route instructed location l_i associations.
- (lines 5–6): in case multiple β's have the most route instructed location associations, and all β's associate o_j to λ, then continue in the direction of the previous route instructed action a_i.
- (lines 7–8): choose the β with the earliest route instructed location associations.

Step (5) in Fig. 8 occurs within lines 22–23. We re-categorize the β's in $\mathcal{B}_a^{expanded}$ into $\mathcal{B}_a$ and $\mathcal{B}_s$ via `Split Branches()` (line 23), and we execute the suggested action in the leaf node of β^{best} (line 22).

Step (4b) in Fig. 8 occurs within lines 26–31. We need to verify that there exists a location to backtrack (line 26). We search for the "closest" branch with the most observed locations β^{most} in $\mathcal{B}_s$ (line 27). That branch needs to be removed from $\mathcal{B}_s$ and added to $\mathcal{B}_a$ (lines 28–29). We then backtrack to the last observed location o_j in β^{most} and execute the suggested action of it (lines 30–31).

Step (4c) in Fig. 8 occurs at lines 19 and lines 33. The former indicates that a complete path hypothesis exists that satisfies the route instructions. The latter indicates that nothing is left to explore as both $\mathcal{B}_a$ and $\mathcal{B}_s$ are empty.

Algorithm 2 Pseudo-code to determine β^{best}

1: **procedure** DETERMINEBESTFIT($\mathcal{B}$)
2: **if** one $\beta \in \mathcal{B}$ has the most l_i associations **then**
3: Output that specific β
4: **else if** Multiple $\beta \in \mathcal{B}$ has the most l_i associations **then**
5: **if** All latest o_j is a $\lambda_{i,k}$ **then**
6: Output β with 1) α most aligned with a_i, and 2) earliest l_i assignment.
7: **else**
8: Output β with 1) α most aligned with a_i, and 2) earliest l_i assignment.
9: **end if**
10: **end if**
11: **end procedure**

Appendix C. Backtracking procedure

Backtracking is realized by keeping track of the signature of the junction and the corridor feature for which the robot entered and left a junction. That signature is used to verify perceptually that the robot has indeed revisited the location. An example of tracked properties is seen at the top of Fig. 14 which contains the knowledge from perception and motion when the robot has moved from $o_1 \rightarrow o_2 \rightarrow o_3$. The corridor features in Fig. 4 are used to fill in the "enter" and "leave" values. The width of the corridor features is extracted online from the robot perception component.

The bottom of Fig. 14 shows the knowledge related to backtracking. Essentially, the robot creates and follows a new route instruction until the backtracked location is reached. We consider a new initialization where the robot is 180 degrees rotated with respect to the heading where it finished, i.e., the robot starts at o_3 where it will now perceive the junction as type FU with a heading in the direction of a^{FU,f_0}. During backtracking the robot will observe the junctions o_2 and o_1 from a different perspective, and these are simply deduced via prior determined rules. For example, if the robot leaves an RT junction via corridor f_1, then it will observe it as a BT junction on the way back. The backtrack action can simply be deduced by checking the corridor that is used to enter the junction pre-backtracking. For example, an LT junction that is entered via feature f_0 pre-backtracking, needs to be exited during backtracking via action a^{RT,f_0}.

Appendix D. Elaborate example

This appendix shows that the number of β branches can grow extensively and non-unique solutions exist in case δ_i is unspecified, even for simple environments. Fig. 15 visualizes a corridor-junction environment where the robot needs to stop at a certain junction. The route instruction is formulated as:

$$\begin{aligned} RI = [&l^{BT}, l^{LT}, l^{RT}, l^{RFL}, l^{LFL}, l^{FU}, l^{BU}], \\ &[(a_0^{RT,f_0}, l_0^{RT}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(a_1^{RT,f_0}, l_1^{RT}, \{\{\text{Unknown}, \text{equal}\}, [], []\}), \\ &(a_2^{LT,f_1}, l_2^{LT}, \{\{\text{Unknown}, \text{equal}\}, [], []\})] \end{aligned} \tag{6}$$

The result of the hypothesis tree that has branched five times is visualized in Fig. 16. At (A), the robot initializes the hypothesis tree. At (B), o_1 is observed of type LT which does not satisfy l_1^{RT}. The only conclusion is that a non-instructed location $\lambda_{0,0}$ is reached. According to lines 5–6 of Alg. 2, β^{best} is the one that advocates an action that aligns (best) with a_0, i.e., a^{LT,f_0}. The option to turn left via a^{LT,f_1} is stored as a standby branch for possible later exploration. At (C), o_2 is observed, and because it satisfies l_1^{RT} an association l_1 is created. The alternative explanation of $\lambda_{0,1}$ is also possible with two actions. The branch β^{best} is chosen as the one with the most l_i associations. At (D), observation o_3 looks similar to o_2. Two branches exist with an equal number of l_i associations. According to lines 7–8 of Alg. 2, β^{best} is the one that has concluded the earliest l_i associations, i.e., the one with $o_2 \sim l_1$. At (E), o_4 is observed that satisfies l_2^{LT}. Two complete path hypotheses (blue colored lines) are present that can both explain the route instruction, and therefore the algorithm is completed.

Appendix E. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2025.105246.

Data availability

No data was used for the research described in the article.

References

[1] R.G. Golledge, Wayfinding Behavior: Cognitive mApping and Other Spatial Processes, JHU Press, 1999.

[2] M. Tomko, K.-F. Richter, Defensive wayfinding: Incongruent information in route following, in: S.I. Fabrikant, M. Raubal, M. Bertolotto, C. Davies, S. Freundschuh, S. Bell (Eds.), Spatial Information Theory, Springer International Publishing, Cham, 2015, pp. 426–446, http://dx.doi.org/10.1007/978-3-319-23374-1_20.

[3] A. Klippel, H. Tappe, L. Kulik, P.U. Lee, Wayfinding choremes—a language for modeling conceptual route knowledge, J. Vis. Lang. Comput. 16 (4) (2005) 311–329, http://dx.doi.org/10.1016/j.jvlc.2004.11.004, Perception and ontologies in visual, virtual and geographic space.URL https://www.sciencedirect.com/science/article/pii/S1045926X0400059X.

[4] B. Tversky, P.U. Lee, Pictorial and verbal tools for conveying routes, in: C. Freksa, D.M. Mark (Eds.), Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 51–64, http://dx.doi.org/10.1007/3-540-48384-5_4.

[5] J. Wang, A. Schwering, Invariant spatial information in sketch maps - a study of survey sketch maps of urban areas, J. Spat. Inf. Sci. 11 (2015) 31–52, http://dx.doi.org/10.5311/JOSIS.2015.11.225.

[6] A. Klippel, H. Tappe, C. Habel, Pictorial representations of routes: Chunking route segments during comprehension, in: C. Freksa, W. Brauer, C. Habel, K.F. Wender (Eds.), Spatial Cognition III, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 11–33, http://dx.doi.org/10.1007/3-540-45004-1_2.

[7] C. Manivannan, J. Krukar, A. Schwering, Spatial generalization in sketch maps: A systematic classification, J. Environ. Psychol. 83 (2022) 101851, http://dx.doi.org/10.1016/j.jenvp.2022.101851, URL https://www.sciencedirect.com/science/article/pii/S0272494422000962.

[8] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer 22 (6) (1989) 46–57, http://dx.doi.org/10.1109/2.30720.

[9] E. Ben-Elia, An exploratory real-world wayfinding experiment: A comparison of drivers' spatial learning with a paper map vs. turn-by-turn audiovisual route guidance, Transp. Res. Interdiscip. Perspect. 9 (2021) 100280, http://dx.doi.org/10.1016/j.trip.2020.100280.

[10] A. Zare Zardiny, F. Hakimpour, Route Matching in Sketch and Metric Maps, J. Geogr. Syst. 23 (3) (2021) 381–405, http://dx.doi.org/10.1007/s10109-020-00343-1.

[11] P.-E. Michon, M. Denis, When and why are visual landmarks used in giving directions? in: D.R. Montello (Ed.), Spatial Information Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 292–305, http://dx.doi.org/10.1007/3-540-45424-1_20.

[12] J. Padgett, M. Campbell, Q-link: A general planning architecture for navigation with qualitative relational information, Robot. Auton. Syst. 108 (2018) 51–65, http://dx.doi.org/10.1016/j.robot.2018.07.001.

[13] B. Krieg-Brückner, T. Röfer, H.-O. Carmesin, R. Müller, A taxonomy of spatial knowledge for navigation and its application to the bremen autonomous wheelchair, in: C. Freksa, C. Habel, K.F. Wender (Eds.), Spatial Cognition: An Interdisciplinary Approach To Representing and Processing Spatial Knowledge, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 373–397, http://dx.doi.org/10.1007/3-540-69342-4_18.

[14] D.C. Shah, M.E. Campbell, A qualitative path planner for robot navigation using human-provided maps, Int. J. Robot. Res. 32 (13) (2013) 1517–1535, http://dx.doi.org/10.1177/0278364913496485.

[15] B. Talbot, F. Dayoub, P. Corke, G. Wyeth, Robot navigation in unseen spaces using an abstract map, IEEE Trans. Cogn. Dev. Syst. 13 (2021) 791–805, http://dx.doi.org/10.1109/TCDS.2020.2993855.

[16] F. Boniardi, A. Valada, W. Burgard, G.D. Tipaldi, Autonomous indoor robot navigation using a sketch interface for drawing maps and routes, Proc. IEEE Int. Conf. Robot. Autom. 2016-June (2016) 2896–2901, http://dx.doi.org/10.1109/ICRA.2016.7487453.

[17] H. Moravec, A. Elfes, High resolution maps from wide angle sonar, in: Proceedings. 1985 IEEE International Conference on Robotics and Automation, 2, 1985, pp. 116–121, http://dx.doi.org/10.1109/ROBOT.1985.1087316.
[18] C. Freksa, R. Moratz, T. Barkowsky, Schematic maps for robot navigation, in: C. Freksa, C. Habel, W. Brauer, K.F. Wender (Eds.), Spatial Cognition II: Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 100–114, http://dx.doi.org/10.1007/3-540-45460-8_8.
[19] G. Chronis, M. Skubic, Robot navigation using qualitative landmark states from sketched route maps, Proc. IEEE Int. Conf. Robot. Autom. 2004 (2004) 1530–1535, http://dx.doi.org/10.1109/robot.2004.1308041.
[20] C. Jian, H. Shi, Resolving conceptual mode confusion with qualitative spatial knowledge in human-robot interaction, in: T. Tenbrink, J. Stell, A. Galton, Z. Wood (Eds.), Spatial Information Theory, Springer International Publishing, Cham, 2013, pp. 91–108, http://dx.doi.org/10.1007/978-3-319-01790-7_6.
[21] M. Mielle, M. Magnusson, A.J. Lilienthal, The auto-complete graph: Merging and mutual correction of sensor and prior maps for SLAM, Robotics 8 (2019) http://dx.doi.org/10.3390/ROBOTICS8020040.
[22] M. Luperto, M. Antonazzi, F. Amigoni, N.A. Borghese, Robot exploration of indoor environments using incomplete and inaccurate prior knowledge, Robot. Auton. Syst. 133 (2020) http://dx.doi.org/10.1016/j.robot.2020.103622.
[23] B. Kuipers, The Map-Learning Critter, Tech. rep., University of Texas, Austin, 1985.
[24] A. Soragna, M. Baldini, D. Joho, R. Kümmerle, G. Grisetti, Active SLAM using connectivity graphs as priors, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2019, pp. 340–346, http://dx.doi.org/10.1109/IROS40897.2019.8968613.
[25] G. Dudek, P. Freedman, S. Hadjres, Using local information in a non-local way for mapping graph-like worlds, in: IJCAI, Citeseer, 1993, pp. 1639–1647.
[26] F. Savelli, B. Kuipers, Loop-closing and planarity in topological map-building, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2 (2004) 1511–1517, http://dx.doi.org/10.1109/iros.2004.1389610.
[27] J.O. Wallgrün, Exploiting qualitative spatial constraints for multi-hypothesis topological map learning, in: K.S. Hornsby, C. Claramunt, M. Denis, G. Ligozat (Eds.), Spatial Information Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 141–158, http://dx.doi.org/10.1007/978-3-642-03832-7_9.
[28] S. Tully, G. Kantor, H. Choset, F. Werner, A multi-hypothesis topological slam approach for loop closing on edge-ordered graphs, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, 2009, pp. 4943–4948, http://dx.doi.org/10.1109/IROS.2009.5354255.
[29] G. Parekh, M. Skubic, O. Sjahputera, J.M. Keller, Scene matching between a map and a hand drawn sketch using spatial relations, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 4007–4012, http://dx.doi.org/10.1109/ROBOT.2007.364094.
[30] M. Mielle, M. Magnusson, A.J. Lilienthal, Using sketch-maps for robot navigation: Interpretation and matching, in: 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR, 2016, pp. 252–257, http://dx.doi.org/10.1109/SSRR.2016.7784307.
[31] J.O. Wallgrün, D. Wolter, K.F. Richter, Qualitative matching of spatial information, GIS: Proc. ACM Int. Symp. Adv. Geogr. Inf. Syst. (2010) 300–309, http://dx.doi.org/10.1145/1869790.1869833.
[32] A. Schwering, J. Wang, M. Chipofya, S. Jan, R. Li, K. Broelemann, SketchMapia: Qualitative representations for the alignment of sketch and metric maps, Spat. Cogn. Comput. 14 (2014) 220–254, http://dx.doi.org/10.1080/13875868.2014.917378.
[33] S. Jan, A. Schwering, C. Schultz, M.C. Chipofya, Cognitively plausible representations for the alignment of sketch and geo-referenced maps, J. Spat. Inf. Sci. 2017 (14) (2017) 31–59, http://dx.doi.org/10.5311/JOSIS.2017.14.294.
[34] H.L. Chen, B. Hendrikx, E. Torta, H. Bruyninckx, R. van de Molengraft, Behavior adaptation for mobile robots via semantic map compositions of constraint-based controllers, Front. Robot. AI 10 (2023) 917637, http://dx.doi.org/10.3389/frobt.2023.917637.
[35] B. Kuipers, The spatial semantic hierarchy, Artificial Intelligence 119 (1) (2000) 191–233, http://dx.doi.org/10.1016/S0004-3702(00)00017-5.
[36] H. Hua, P. Zhang, J. Renz, Qualitative place maps for landmark-based localization and navigation in GPS-denied environments, in: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 23–32, http://dx.doi.org/10.1145/3347146.3359107.
[37] R.W.M. Hendrikx, H. Bruyninckx, J. Elfring, M.J.G. Van De Molengraft, Local-to-global hypotheses for robust robot localization, Front. Robot. AI 9 (2022) http://dx.doi.org/10.3389/frobt.2022.887261.
[38] DoT, Manual on Uniform Traffic Control Devices: for Streets and Highways, U.S. Dept. of Transportation, Federal Highway Administration, 2009.



Hao Liang Chen obtained his M.Sc. degree (2018) in Mechanical Engineering and his Ph.D. degree (2025) from Eindhoven University of Technology. His Ph.D. research focused on modeling robot behavior using semantic world model knowledge. He is currently a postdoctoral researcher at the Eindhoven University of Technology in the Robotics group, where his current research is focused on the design of digital twin systems for robotic education



Elena Torta is an associate professor in Robotics at the Mechanical Engineering Department of the Eindhoven University of Technology. Her research addresses control and world modeling problems in human–robot and robot–robot collaboration. She holds a Ph.D. cum laude from the Eindhoven University of Technology and is experienced and fascinated by research problems that cross disciplinary boundaries. She held a position as software architect at ASML B.V. for five years. She has performed several human–robot interaction studies that lead to computational models actionable by robotics control algorithms.



Herman Bruyninckx Dr. Bruyninckx obtained the Masters degrees in Mathematics (Licentiate, 1984), Computer Science (Burgerlijk Ingenieur, 1987) and Mechatronics (1988), all from the KU Leuven, Belgium. In 1995 he obtained his Doctoral Degree in Engineering from the same university. He is full-time Full Professor at the KU Leuven, and part-time at the Eindhoven University of Technology. The research focus in both places is on the composability of the most advanced, knowledge driven algorithms for the dynamics of motion control of complex robotics applications, with distributed sensor processing and resource monitoring. The complementary objectives are to realize such systems with the least amount of resource costs, with "good enough" quality, and with full "explainability". René van de Molengraft



René van de Molengraft received the M.Sc. degree (cum laude) in mechanical engineering and the Ph.D. degree from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1986 and 1990, respectively. In 1991, he returned to the control group at Mechanical Engineering, after having fulfilled his military service. He is currently an Associate Professor and the chair of the Robotics Section, Eindhoven. He founded the Tech United RoboCup Team, and gained seven world champion titles in the MSL robot soccer, and two world champion titles in @Home carerobot league.