Home Alone: Autonomous Extension and Correction of Spatial Representations

Nick Hawes, Marc Hanheide, Jack Hargreaves, Ben Page, Hendrik Zender, Patric Jensfelt

***Abstract*— In this paper we present an account of the problems faced by a mobile robot given an incomplete tour of an unknown environment, and introduce a collection of techniques which can generate successful behaviour even in the presence of such problems. Underlying our approach is the principle that an autonomous system must be motivated to act to gather new knowledge, and to validate and correct existing knowledge. This principle is embodied in Dora, a mobile robot which features the aforementioned techniques: shared representations, non-monotonic reasoning, and goal generation and management. To demonstrate how well this collection of techniques work in real-world situations we present a comprehensive analysis of the Dora system's performance over multiple tours in an indoor environment. In this analysis Dora successfully completed 18 of 21 attempted runs, with all but 3 of these successes requiring one or more of the integrated techniques to recover from problems.**

I. Introduction

Service robots working in human environments will require a lot of knowledge about their surroundings in order for them to discharge their duties successfully (e.g. maps, names for rooms etc.). In existing work this knowledge is assumed to be provided to the system during a *tour* in which a human guides the robot through the environment indicating important physical features and also providing associated descriptions. For a human, a tour is a rather natural way to convey information to a robot. However, this process is fraught with problems from a robot's point of view.

Fig. 1: Dora in the kitchen; a room of the flat the experiments were conducted in.

Nick Hawes, Marc Hanheide, Jack Hargreaves, and Ben Page are with the School of Computer Science at University of Birmingham; email: n.a.hawes@cs.bham.ac.uk. Hendrik Zender is with the German Research Center for Artificial Intelligence (DFKI), Saarbrücken. Patric Jensfelt is with the Royal Institute of Technology (KTH), Stockholm. The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

These problems include those familiar to roboticists (sensor noise, incorrect results generated by components) and those which are based more on a typical user's lack of knowledge about a robot's requirements. In this paper we present an analysis of how an intelligent mobile robot can be designed to overcome some of the problems that it will face following a home tour by a naïve user. From this analysis we developed a system with a range of behaviour and reasoning capabilities, including functionality related to the explicit representation of *gaps in its knowledge* and the ability to reason about how to act to fill these gaps. Our analysis focuses on how these abilities allow the system to recover from common errors that occur during tours through an environment for the first time. It is supported by evidence gathered from multiple runs of the robot in a real flat.

Our system features two main advances beyond previous work: *a drive to validate and refine spatial knowledge* (to detect and correct gaps and errors); and *non-monotonic reasoning about spatial knowledge* (necessary to support the revision of the robot's representations). These are presented in Section III. Sections IV and V present quantitative and qualitative analyses of our system over 18 successfully extended tours. Before this, Section II presents some background and existing approaches to dealing with typical tour-related problems.

II. Background & Related Work

A tour as a mechanism for providing a robot with information is a common meme in autonomous and interactive robotics (e.g. [1], [2]). Whilst previous work has identified problems that occur during tours due to the guide's incorrect expectations about a robot's abilities [3], to date no work has provided an account of the problems that occur from a robot's point of view, and what mechanisms can cope with them. When referring to "problems" we don't mean the ability for a robot to solve a particular fixed problem. Instead we refer to cases where either the robot or the human have (in some sense) functioned incorrectly.

From the robot's point of view we can distinguish *two general types of problems*: those caused by the robot not functioning correctly in a designer-anticipated context (e.g. the failure of a perceptual routine resulting in an object or feature going unobserved), and those caused by the robot's environment changing to become out of sync with the robot's current representation of it (e.g. a human closing a door which was open during the tour). Given the limited knowledge of current robots (including our own), these different types of problems are actually indistinguishable at a system level. Both result in the robot having a representation of the environment which does not reflect the true state of the world. Thus we propose that both problem types can be addressed by ensuring that robots feature a motivating drive *to ensure that their representations are as complete and as correct as possible*. Given that a tour only provides a robot with part of the knowledge it needs, any service robot in this situation must therefore be able to *autonomously explore* its environment (after the tour) in order to fill any knowledge gaps (following the principle of *discovery* [4]). And given that the aforementioned problems can occur both during the tour and during autonomous exploration, it is also important that a robot never assumes that its knowledge is ever entirely complete or correct. Instead the aforementioned drive should give rise to behaviours which attempt to *validate and refine* its knowledge as appropriate, in addition to extending it. This behaviour is an essential part of any system that must simultaneously learn about, and perform tasks in, the real world. Such systems do not have the luxury of separate training and test phases, and instead must learn online whilst acting.

It could be argued that all the problems we expect a robot to encounter are actually the result of a badly designed or engineered system, or a poorly trained user. However, given the current state-of-the-art it is unrealistic to expect that complex intelligent robots deployed into human-populated environments will not encounter these problems, or similar ones. The wide range of contexts these systems must cope with, and the behaviours they must be capable of generating, mean that it may prove just too difficult to fully debug a whole system before it is deployed in its target environment. Thus a robot must be equipped with systems that mitigate the effects of run-time problems. It is also unrealistic to expect any non-expert user to be able to reliably provide a current robot system with all of the information it actually needs during a tour. For example, localisation and navigation systems typically require detailed maps of an environment in order to function correctly, and it is clear that a busy user may only show a robot some subset of the environment before moving on to another task.

Existing work typically relies on one of two different approaches to coping with problems at run-time: *autonomy* or *interaction*. Autonomous approaches to coping with localisation errors typically leverage the probabilistic machinery they are implemented with. For example, most existing systems are capable of characterising their mapping and localisation uncertainty, some are capable of explicitly acting to reduce this uncertainty [5], and others can explicitly represent possibilities for extending their metric spatial knowledge beyond the frontiers of their maps [6]. Interactive approaches require that the robot asks the human to provide input if it encounters a problem. For example, systems have been developed which ask a human whether it has just passed a door [2], and which use uncertain categorisations as the basis for dialogue about room categories [1] and object features [7].

In this paper we focus on the autonomous resolution of some prototypical instances of perceptual problems: the possible failure of a door detector to identify doors in the robot's environment, and problems in a spatial model resulting in either failure to identify unvisited areas of space or failure to move the robot correctly.

III. System Architecture

We have developed an architecture for an intelligent mobile robot which embodies the principles described in the previous section. This architecture is deployed as part of *Dora*, a robot developed in the CogX project[1] (see Fig. 1). Dora has been designed to be able to autonomously explore an unknown environment and patrol a known one. It is also able to autonomously determine the functional category of rooms (to support future human-robot interaction and service tasks). Dora explicitly represents gaps in its spatial model (i.e. its map and its knowledge of room categories) and is able to take action to fill these. When displaying nominal behaviour, this model is in alignment with its environment; it contains representations of all visited rooms and their categories. In case of misalignment resulting from problems, Dora has mechanisms for updating these representations.

The Dora architecture is an instantiation of the CoSy Architecture Schema, and is implemented using the associated software toolkit, CAST [8].

[1] `http://cogx.eu`

This schema is based on a shared working memory model and is designed to support flexible, parallel, information sharing in heterogenous, yet integrated, systems. This support is an essential requirement for robots such as Dora which must share a lot of explicit knowledge between its subsystems and where this knowledge must be both used and refined concurrently.

In the following subsections we present two of the subsystems which allow Dora to act to make its representation as complete and correct as possible when faced with the complexity and dynamics of the real world: its multi-layered spatial model and its goal management framework. These subsystems are jointly revising representations and mutually notify each other about the modifications they make through the CAST framework. For a more detailed account of the rest of the Dora architecture, see previous work, e.g. [9], [10].

A. Reasoning with changing spatial knowledge

As a mobile robot, Dora requires a map-based spatial representation for localisation and navigation. We take the multi-layer hybrid approach to map representation described in [10]. Particularly relevant for this paper are the map's *place* and *conceptual* layers. Fig. 2 shows visualisations of maps generated by Dora. The colouring of the place nodes represents the information about the containment of places within rooms as provided by the conceptual layer.

The place layer is responsible for discretising the continuous representation of space generated by the layers below it (using a metric SLAM approach) into a graph-based representation of free space. The nodes in the graph are called places, and they are created at $1m$ intervals along the robot's trajectory. Graph-edges indicate immediate adjacency of places and the possibility of moving between them. Gateways play an important role for clustering places into larger coherent areas. In buildings, doorways are typical gateways. A *door detection* component inspects laser scans to find width changes that look like doorways and triggers the place layer to mark the corresponding place as a *gateway place*. The place layer can also generate *placeholders*. A placeholder represents an unexplored direction that the robot might move in, thus explicitly representing the possibility of creating additional places beyond the limits of the current place graph. The place layer is connected to a navigation module that can move the robot to places and placeholders.

It is important that autonomous robots which are supposed to operate in domestic environments have a notion of spatial units that are also meaningful for humans. In our system, the conceptual layer (an extended and improved variant of the conceptual map described in [11]), is responsible for linking the other layers of the spatial model (including the place layer) to concepts which humans can relate to. The conceptual layer represents map knowledge in an OWL-DL ontology consisting of a taxonomy of spatial concepts (*TBox*), and how they can be characterised in terms of human-compatible categories, as well as the knowledge about individuals in the domain (*ABox*). It also makes use of a combined rule and OWL-DL reasoner based on the Jena Framework to perform different reasoning tasks on the ontology. The taxonomy defines spatial regions called *areas* as the basic units corresponding to a human-compatible segmentation of space. We distinguish between two basic kinds of areas. *Rooms* are spatial areas whose primary purpose is defined by the kinds of actions they afford (e.g., WordNet defines a kitchen as "a room equipped for preparing meals"). Here, we make the simplified assumption that the presence of certain objects (e.g. kettle, toothpaste) determines the respective subcategory of a room (e.g. kitchen, bathroom). *Passages* (e.g. corridors) are areas whose primary purpose is to link rooms and provide access to other areas.

A prerequisite for reasoning about room categories is to have a notion of rooms. Based on the information about the connectivity of places and whether they constitute gateways or not, the conceptual layer forms areas (rooms or corridors) by clustering places that are transitively interconnected without passing a doorway. Since both door and object detection can malfunction, room formation and categorisation must be *non-monotonic* processes in order to support the potential for knowledge revision. Room formation and maintenance is handled by a general purpose rule engine, while the OWL-DL reasoner is used to infer which categories can be applied to known rooms. Both are able to make non-monotonic inferences in the ABox: whenever a previously true condition turns false, the conclusions drawn from it are retracted. This means that the conceptual layer is capable of correcting both the assignment of places to rooms and the assignment of categories to rooms when additional (counter-)evidence becomes available.

In order to perform room categorisation, Dora must populate the conceptual layer's ABox with knowledge about objects in the environment. To

do this Dora has an *active visual search* behaviour. This moves Dora around the current room running an object detector using a set of pre-trained models. Our visual search implementation is a derivation of the randomised art gallery algorithm which only looks at points in the room which are likely to contain objects [12]. These points are currently linked to obstacles in Dora's metric map.

B. Behaviour generation

Dora's behaviour is generated and coordinated by a continual planning and execution system which is controlled by a goal generation and management framework. This framework is responsible for generating new goals (i.e. descriptions of desired future states) for Dora from the outputs of sensors and other processing, selecting which goals should be followed, triggering the planner to create a plan to achieve the selected goals, then managing the execution of subsequent plans. Dora has *goal generators* which aim to satisfy its overall drive to have accurate knowledge about its environment (as discussed in Section II) by creating individual goals to *explore* each generated placeholder (yielding a more complete place layer) and to *categorise* each generated room (yielding a more complete conceptual layer). In addition to this, Dora also generates a goal to *patrol* each previously generated place (and consequently each area too). This allows it to revisit known space, thus validating previously generated knowledge. If a goal is no longer valid (if it has been achieved or it is not longer appropriate) the goal generator which created it removes it from the framework.

Dora's goal management framework is responsible for selecting which of the generated goals should be forwarded to planning and execution. A goal which has been thus selected is referred to as *activated*. Before activation it must pass through two additional stages. After generation, goals are first *unsurfaced*. They must then pass through a bank of *filters* to become *surfaced* before they can be considered for activation. The management processes move goals between these stages based on a variety of conditions. In Dora, goals are surfaced based on their type (all the goal types described above are automatically surfaced) and only unsurfaced if the system fails to achieve them more than a set number of times (currently 10).

Goals are activated based on a combination of features. The current implementation ranks all goals based on the number of previous attempts to achieve them, then by priority, then by a calculation of information gain versus estimated cost. Only the top ranked goal is then selected for activation. In Dora the information gain for exploring a placeholder is related to the amount of free space it covers, the information gain for categorising a room is related to the number of places it contains, and for a patrolling a place it is related to the last time the place was visited. Cost estimates are based on the distance Dora would have to travel to achieve the goal. These values are associated with a goal by its generator. The generator is also responsible for maintaining these values as state changes occur (e.g. the robot moves), and also for removing goals that are no longer justified by the robot's knowledge (e.g. when a placeholder is visited or a room is categorised). A goal is assigned a priority when it is generated. Currently these priorities are inherited from priorities manually assigned to their generators. Our current scheme assigns explore and categorise goals the same priority, with patrol goals having a lower priority. This reflects the need to generate knowledge before it can be validated or revised. The effects of this decision are seen in the case studies discussed below. Ranking goals first by the number of previous attempts to achieve them allows goals to be *postponed* when Dora fails to achieve them (as they are effectively relegated to the end of the ranked activation list). This is designed to prevent the system from repeatedly trying and failing to achieve one goal when others still exist. Currently Dora detects some failures by waiting for the execution system to timeout. The effects of this design are also discussed below.

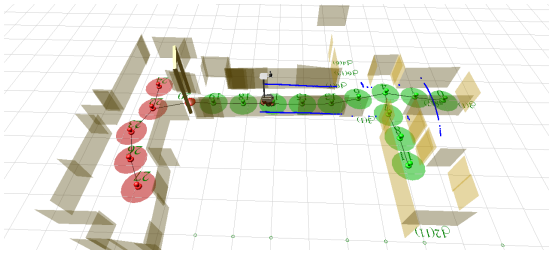
The use of a continual planner is essential for a robot in a dynamic environment. Changes in the environment and sensor noise mean that plans often need to be revised on-the-fly as they either become impossible or more information becomes available, allowing other plans to become possible. The continual planner we use is specifically designed to detect such changes during plan execution, and replan as appropriate [13]. This allows Dora to cope with open world problems (e.g. not knowing how many rooms exist in advance) in a deliberative way.

IV. Experiments

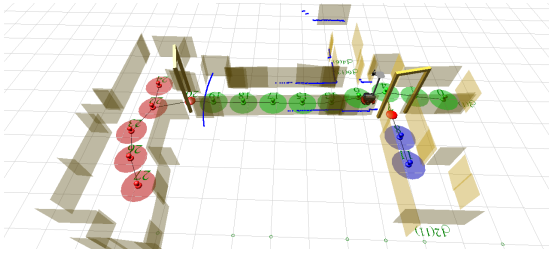
To evaluate how the Dora system copes with the problems that occur in realistic settings we took the robot out of the lab and into a flat in Birmingham, UK[2]. Here we ran Dora multiple times and gathered data about its performance[3]. Dora (displayed

[2]Mason Hall, `http://j.mp/9tLEUx`

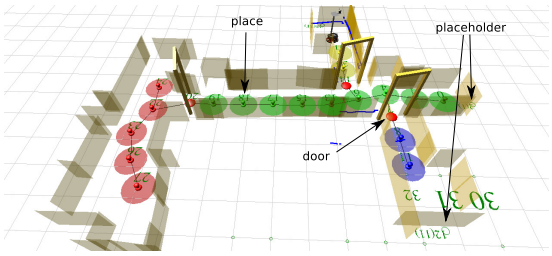
[3]Video available at `http://cogx.eu/results/dora`



(a) The door to the lower-right room has not been detected.



(b) Door detection triggers a revision of the representation.



(c) All rooms are now found.

Fig. 2: Snapshots from the evolution of the spatial representation during one run of Dora. Placeholder are depicted as small unfilled green circles, places are solid disks. This representation is entirely created from scratch each run as part of the tour and the autonomous exploration.

in Fig. 1) is based on a MobileRobots P3DX platform equipped with a Hokuyo laser scanner and a pan-tilt unit holding two Flea2 cameras. The software system runs on a single dual-core laptop. This section describes the methodology we followed when running the system and presents some descriptive statistics generated from this experiment.

A. Procedure

The environment we operated Dora in was composed of a kitchen, a corridor, and two bedrooms. Dora was initially given a short tour, then required to build a complete map of this environment and categorise all the rooms except the corridor. Each bedroom contained 3 or 4 objects that Dora could recognise and use for room categorisation. The kitchen, being larger, contained 6 objects. The objects were placed in relatively normal positions where they were visible to the robot. Fig. 2(c) shows a map of the environment as acquired by Dora. The room to the left is the kitchen, which is connected by the corridor to the bedrooms. Dora was always started at the right end of the corridor, then given a tour along the corridor, into the first bedroom (lower right in Fig. 2(c)), and then into the kitchen (ignoring the other bedroom), and no room was assigned a category. After the tour, the filter bank in the goal management framework was reconfigured to let goals surface, switching Dora to autonomous behaviour.

We conducted a systemic analysis of all the runs, adopting the SInA methodology originally developed for human-robot interaction [14]. To support this, Dora was instrumented with logging mechanisms to record a range of state changes. The acquired logs were used to generate annotations to support qualitative analysis using visual tools [15], and quantitative analysis via descriptive statistics [16]. In the following we take a closer look at the actual behaviour represented by the creation and activation of goals in the system.

TABLE I: Descriptive statistics from all 18 successful runs. Standard deviation indicated by $\pm$.

Type	(re-)activ.	duration	dur. per run
categorize room	4.1 (1.1)	216 ± 117s	901s (63%)
explore place	16.6 (0.94)	22.5 ± 15.6s	375s (27%)
patrol places	4.4 (0.0)	9.3 ± 6.0s	41.4s (3%)

B. Descriptive statistics

Dora met our criteria for success (i.e. explored and categorised all rooms) in 18 out of 21 attempted runs (85.7%). In the remaining three runs Dora failed to explore the bedroom which was omitted during the tour. This was because no placeholder was generated in it at any point during these runs. On average, Dora took 26.25 ± 8.64 (std. dev.) minutes to accomplish the full task in the 18 successful runs; of which the system spent 11.45 seconds planning for goals selected by the goal management scheme. An average of 23.52 ± 8.93 minutes was dedicated to autonomous behaviour, the remainder to the tour. The average total distance Dora travelled was 104 metres.

Tab. I presents some per-run statistics on active goals (those that are planned for and executed by the robot, thus generating behaviour). The "(re-)activation" column shows the average number of activations of goals of each type. In all runs, there were 3 rooms to categorise. The average number of goal activations for room categorisation was greater than three due to goal postponements. This also explains why we observe 1.1 re-activations in the case of categorise-room goals. The action execution timeout was set to 6 minutes (360 seconds) which was occasionally

not long enough for a categorisation process to complete successfully.

Tab. I also shows that 63% of the autonomous part of each run was spent categorising rooms, i.e., looking for objects. The exploration of new places accounted for 27% and only 3% were dedicated to patrolling previously-visited places. This is explained by the setup of the experiment: a run was deemed be finished when all rooms were explored and categorised. Patrol goals were assigned a lower priority than exploration and categorisation, so they were only activated in five runs when there were no other surfaced goals, but not all rooms had already been found and categorised. The remaining time in each run was taken by Dora deciding what to do next (activity planning accounted for roughly 14% of the remaining time) and general communication overhead between components.

V. Case Studies

The following sections present a detailed analysis of the different types of system runs we observed during our experimentation. We start with the case in which no problems occur and follow this with cases that cover a number of problems that occurred during the experiments.

Case A: Ideal case

In the ideal case Dora successfully detects all doors when first passing through them, puts placeholders in the bedroom which was not entered during the tour (thus allowing it to autonomously explore it), and achieves all goals on the first attempt. It should be noted that the ideal case requires the majority of Dora's capabilities, including autonomous extension and handling of incomplete knowledge. It doesn't require any non-monotonic reasoning or active validation of acquired knowledge because the knowledge is correct when initially obtained.

Fig. 3 shows the progress of a run of Dora. The run illustrated in the figure is *not* an ideal one in total, it is included to serve all the case studies. However, if we look only at the first 10 minutes (600 seconds) it presents the prototypical behaviour of Dora. In this behaviour placeholders, uncategorised rooms, and existing places give rise to goals to explore, categorise, or patrol them, respectively. In the ideal case patrol goals never get activated, as the experiment is finished once all placeholders have been explored or discarded and all rooms have been categorised. Patrol goals are assigned a lower priority, so they are only pursued if no other goals exist. In the figure we can see the two phases of the experiment. During the tour, all the goals were unsurfaced (represented as green bars). But it can be seen that during the tour (unsurfaced) goals were created whenever a placeholder was assigned to an area of open space that had not yet been visited, or when a new (and thus uncategorised) room entity was created due to the detection of a door. During the tour, Dora frequently created placeholders directly in front of itself which were immediately visited as part of the tour and therefore removed immediately. Explore goals 3, 9, 12-16, 21, and 23 were such cases. Also, categorisation goals for the first bedroom and the kitchen (corresponding to goals "CategorizeRoom_1" and "CategorizeRoom_2") were generated during the tour because all doors were correctly detected. Fig. 2(b) illustrates the spatial representation of an equivalent case, where the two rooms were detected.

After roughly 130 seconds of touring the autonomous behaviour was activated. The goal management framework took control, choosing which goal to pursue next by trading predicted information gain for costs. Goals were activated and then corresponding plans executed. Dora's behaviour caused new goals to be created. In Fig. 3, "Explore_25" was the first goal to be created while the robot was pursuing another goal. Shortly after creation it was activated and the associated plan executed. In this case, the exploration goal was removed during plan execution because the spatial layer concluded that the placeholder was too close to an already existing place, so the creation of a new place was not possible ("activated and merged with an existing place").

The "active" line in Fig. 3 highlights the class of the currently activated goal. It shows that Dora started by exploring 4 placeholders before deciding to categorise room 2 (the kitchen). This decision was justified by the cost and gain associated with that plan at that point: Dora was in room 2 already (low cost) and it has a large area (high gain).

In the ideal case similar behaviour is demonstrated throughout the run, with Dora switching between successful exploration and categorisation until the task is complete. However, due to the nature of the real world and robot perception, this is not guaranteed. Hence, we now look at cases which deviate from the ideal behaviour.

Case B: Missed placeholder

The first problem case we look at is one in which Dora does not immediately discover the open space associated with the bedroom ignored

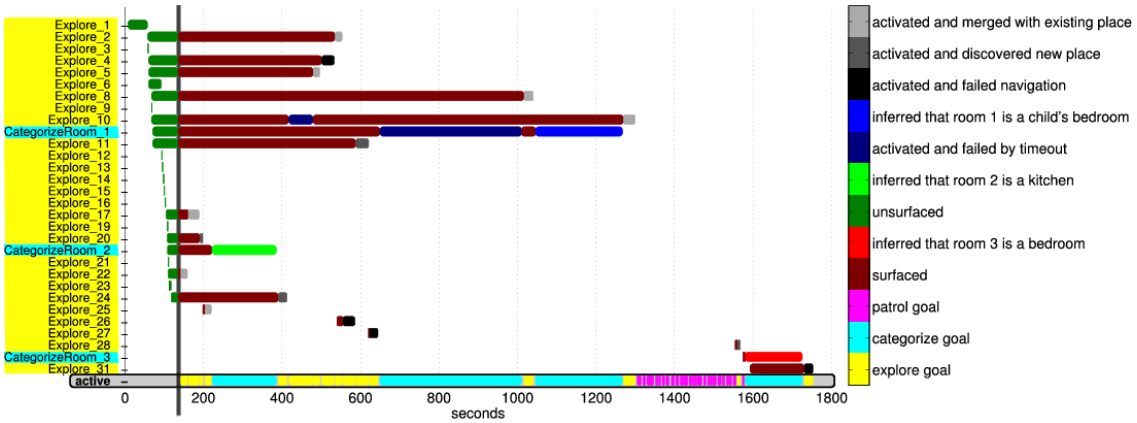


Fig. 3: A visualisation of the progress of a single Dora run. The x-axis shows the time from the start of the run. The y-axis lists all the goals generated in the run. The line titled "active" shows the type of the currently activated goal (in yellow, cyan, and magenta). The other lines show the life-time of each goal, colour-coded according to the respective status (unsurfaced, surfaced or active). Individual patrol goals are omitted for brevity. The vertical line at second 130 marks the end of the tour when the robot starts autonomous exploration.

during the tour. This occurs when no placeholder is generated inside the room and consequently no associated goal to explore is created either. Without the drive to continuously update and validate its representations, Dora would not be able to recover from this problem by revising and extending its knowledge.

This particular problem occurred in 5 of the 18 successful runs. One instance is depicted in Fig. 3. We can see that by 1300s only two rooms had been discovered and there are neither explore nor categorise goals left to pursue. At this point, Dora believes she has explored and categorised everything. She still has the drive to validate her representation so she starts patrolling, motivated by the lower priority patrol goals. Though these goals are not individually visualised we can see a series of (pink) patrol goal activations in the "active" line. During this period Dora patrolled from place to place, eventually reaching the place in front of the bedroom door. This time, the open space *was* detected and a placeholder generated, triggering the creation of the "Explore_28" goal. Because explore goals are assigned a higher priority than patrol goals, Dora immediately stopped patrolling and started exploring into that open space. While exploring, the robot detected the door and concluded that this is a new, yet unknown, room. A corresponding categorisation goal ("CategorizeRoom_3") was created and subsequently activated. After finding two objects, Dora concluded that this room was in fact a bedroom. It can be noted that during the active visual search in that room the robot also detected more open space (cf. the "Explore_31" goal). So, the system is capable of handling side effects of actions. To end this case, Dora then also explored this last placeholder, finally accomplishing the overall task successfully.

Case C: Undetected door

Dora is designed to detect doors when passing through them. Detection is performed by inspecting the laser scan for changes in width which might indicate the narrower passage of a door frame. This process is not completely reliable due to sensor noise and changes in robot orientation when moving. This means occasionally (on 1 from the 18 successful runs) Dora does not detect a door when one is present in the environment. Fig. 2(a) visualises how this occurred during our experiments: the door between the corridor and the first bedroom (lower right) was not detected during the tour, so the bedroom was initially considered to be an extension of the corridor. Later, when Dora was exploring a placeholder close to the door, she finally detected it (see Fig. 2(b)). This caused the conceptual layer to revise its representation, producing two rooms from the previously existing one (and yielding the correct map shown in Fig. 2(c)). This case underlines the necessity for non-monotonic reasoning and dynamic goal generation. It was only after revising its representation in the light of new information that Dora could determine that although it had already visited the newly distinguished room, it still lacked a categorisation result for it.

Case D: Timeouts and re-activations

A problem that occurs for the Dora system once the initial tour is over is that a planned action occasionally results in actual robot behaviour that both fails to generate the intended effects, and fails to

terminate. For example, a navigation command to move the robot to a place or placeholder may rarely encounter a problem which gets Dora stuck somewhere, and occasionally the active visual search will fail to find any objects at all. This problem is due to the planning actions being an opaque abstraction over the implementing techniques. This abstraction prevents components performing detailed reasoning about the conditions in which the actions can safely be executed, and the full range of their possible effects. In other words the world model Dora uses for reasoning does not capture the full range of (uncertain) knowledge it should. We address this problem by taking the, admittedly simplistic, approach of defining timeouts for the achievement of each type of goal (as mentioned in Sec. III-B). For instance, we set the timeout for achieving room categorisation goals to 360 seconds. The aim of this approach is to allow Dora to postpone goals that can't be achieved due to problems caused by the aforementioned incomplete world model. In our experiments, there was more than one such occasions per run in average, with only 5 runs featuring no such case.

An example of this behaviour is apparent in Fig. 3. The goal "CategorizeRoom_1" was active from 650s–1010s, but Dora failed to categorise the room. Instead of indefinitely pursuing that single goal, Dora postponed it and instead continued exploring, re-activating the still surfaced categorisation goal later on. The "Explore_10" goal from the same run shows a similar life cycle.

VI. Conclusion

We started this paper by describing our general view of the problems faced by a robot when placed in an unknown environment. From this view we derived a principle that such a robot should be driven to ensure that its knowledge of its environment is complete and correct. This led us to conclude that robots must be motivated to extend, validate and refine their knowledge in order to provide a representation to support action. We then described Dora, an intelligent mobile robot which instantiates these principles using *shared representations*, *non-monotonic reasoning*, and a *goal generation and management framework*, with a particular focus on spatial knowledge. To support our claims we presented a comprehensive analysis of Dora's performance when given incomplete tours of environment by a human, and then left to act autonomously. The contents of Dora's architecture allowed it to successfully complete 18 out of 21 of these runs, even in the presence of errors and unreliable perception. Only 3 of the 18 successes were ideal cases, with the remaining 15 requiring either goal management or non-monotonic reasoning to recover from one or more problems.

The analysis of the data produced by our experiments has yielded further insights into the limitations and constraints of our approach. In future work we will use these insights to support the introduction of non-monotonic processing into other Dora subsystems. We will also work towards closing the loop with humans (both during and after the tour) in order to exploit them as an additional source of knowledge.

References

[1] J. Peltason, F. H. K. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide, and E. A. Topp, "Mixed-initiative in human augmented mapping," in *Proc. Int. Conf. on Robotics and Automation*, May 2009, pp. 2146–2153.

[2] G.-J. Kruijff, H. Zender, P. Jensfelt, and H. I. Christensen, "Clarification dialogues in human-augmented mapping," in *Proc. Conf. on Human-Robot Interaction*, March 2006, pp. 282–288.

[3] K. Fischer and M. Lohse, "Shaping naive users' models of robots' situation awareness," in *Proc. Int. Symp. on Robot and Human Interactive Communication*, August 2007, pp. 534–539.

[4] D. Maio and S. Rizzi, "Clustering by discovery on maps," *Pattern Recognition Letters*, vol. 13, no. 2, pp. 89–94, 1992.

[5] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 684–699, 2010.

[6] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. Int. Conf. on Autonomous Agents*, May 1998, pp. 47–53.

[7] N. Hawes, J. Wyatt, M. Sridharan, M. Kopicki, S. Hongeng, I. Calvert, A. Sloman, G.-J. Kruijff, H. Jacobsson, M. Brenner, D. Skočaj, A. Vrečko, N. Majer, and M. Zillich, "The PlayMate system," in *Cognitive Systems*. Springer, 2010, pp. 367–393.

[8] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with CAST," *Advanced Engineering Informatics*, vol. 24, no. 1, pp. 27–39, January 2010.

[9] M. Hanheide, N. Hawes, J. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, and G.-J. Kruijff, "A framework for goal generation and management," in *Proc. AAAI Workshop on Goal-Directed Autonomy*, July 2010.

[10] J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G.-J. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, A. Vrečko, H. Zender, and M. Zillich, "Self-understanding and self-extension: A systems and representational approach," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 282–303, December 2010.

[11] H. Zender, P. Jensfelt, O. M. Mozos, G.-J. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, June 2008.

[12] A. Aydemir, K. Sjöö, and P. Jensfelt, "Object search on a mobile robot using relational spatial information," in *Proc. Int. Conf. on Intelligent Autonomous Systems*, August 2010, pp. 111–120.

[13] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Journal of Autonomous Agents and Multiagent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[14] M. Lohse, M. Hanheide, K. Rohlfing, and G. Sagerer, "Systemic interaction analysis (SInA) in HRI," in *Proc. Int. Conf. on Human-Robot Interaction*, March 2009, pp. 93–100.

[15] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes, "Elan: a professional framework for multimodality research," in *Proc. Language Resources and Evaluation Conference (LREC)*, May 2006.

[16] M. Hanheide, M. Lohse, and A. Dierker, "SALEM - Statistical AnaLysis of Elan files in Matlab," in *Proc. LREC Workshop on Multimodal Corpora: Advances in Capturing, Coding and Analyzing Multimodality*, May 2010.