The STRANDS Project

Long-Term Autonomy in Everyday Environments

Thanks to the efforts of the robotics and autonomous systems community, the myriad applications and capacities of robots are ever increasing. There is increasing demand from end users for autonomous service robots that can operate in real environments for extended periods. In the Spatiotemporal Representations and Activities for Cognitive Control in Long-Term Scenarios (STRANDS) project (http://strands-project.eu), we are tackling this demand head-on by integrating state-of-the-art artificial intelligence and robotics research into mobile service robots and deploying these systems for long-term installations in security and care environments. Our robots have been operational for a combined duration of 104 days over four deployments, autonomously performing end-user-defined tasks and traversing 116 km in the process. In this article, we describe the approach we used to enable long-term autonomous operation in everyday environments and how our robots are able to use their long run times to improve their own performance.

Long-Term Autonomy in STRANDS

Autonomous robots come in myriad forms and can be used in a range of applications. With these differences, long-term autonomy (LTA) has a variety of meanings. For example, NASA's *Opportunity* rover has been autonomous for more than ten years on the surface of Mars; wave gliders can automatically monitor stretches of ocean for months at a time; and autonomous cars have completed journeys of thousands of kilometers. In this article, we restrict our contributions to mobile robots operating in everyday, indoor environments, such as offices and hospitals, and capable of performing a variety of

By Nick Hawes, Chris Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrová, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Körtner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Fäulhammer, Michael Zillich, Markus Vincze, Eris Chinellato, Muhannad Al-Omari, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomáš Krajník, João M. Santos, Tom Duckett, and Marc Hanheide

Digital Object Identifier 10.1109/MRA.2016.2636359
Date of publication: 8 June 2017

service tasks. Across the various robots described previously, there are commonalities in low-level, short-term control algorithms (e.g., closed-loop motor control). Beyond this, the algorithms used to provide long-term, task-specific autonomous capabilities—and the hardware these algorithms control—vary greatly, according to application and environmental requirements. The challenges that distinguish indoor service robots from other LTA robots relate to both their environment and their task capabilities. Indoor task environments are less physically risky than outdoor environments, but they have a comparatively higher degree of short- to medium-term physical variability, e.g., moving objects such as people, doors, and furniture. You might argue that traffic is highly variable, but roads are generally similar to each other and the movement of vehicles is generally more predictable than the movement of people. In terms of application requirements, multipurpose service robots must be capable of predictable scheduled behavior while also being retaskable on demand with high availability and must be able to navigate in relatively confined, dynamic environments. This is in contrast to the largely restricted-purpose systems mentioned previously, such as rovers and wave gliders. Taken together, the set of requirements for indoor service robots presents unique challenges, and, thus, LTA in this context warrants dedicated research.

Given the state of the art, we consider *long-term* for a mobile service robot to mean at least multiple weeks of continuous operation. In very general terms, such LTA operation requires a robot's hardware and software to be robust enough to overcome failure to enable such operation. Such robustness can be provided by both design-time and run-time approaches. It is essential that LTA systems actively manage consumable resources (e.g., a battery) and that any autonomy-supporting capabilities (e.g., localization) are not adversely affected by long run times. While this latter point is common sense and may be common practice in many other technologies (from operating systems to cars), it has only recently been considered in autonomous robotics.

One reason it is challenging to design a service robot to meet the requirements of LTA is the impossibility of anticipating all situations in which it may find itself. If we can enable robots to run for long periods of time, however, then they will have opportunities to learn about the structure and dynamics of such situations. By exploiting the results of such learning, the robots should be able to increase their robustness further, leading to a virtuous cycle of improved performance and greater autonomy. It is this latter point that motivates STRANDS: To go beyond robots that simply survive to those that can improve their performance in the long term. It is within this context that this article makes its main contribution: the STRANDS Core System, a robotic software architecture that was designed for LTA service robot applications and has been evaluated across four end-user deployments. The STRANDS Core System contains a mix of common sense and novel elements that have enabled it to support more than 100 days of autonomous operation. This is the first time all of these elements have been presented together, and this is the first presentation of metrics describing performance across deployments. Our approach is inspired by the work of Willow Garage [1] and the CoBot project [2], plus the pioneering work on the Rhino and Minerva systems (e.g., [3]). Our work is distinguished from previous work by the combination of multiple service capabilities in a single system capable of weeks-long continuous autonomous operation in dynamic indoor environments while using various forms of learning to improve system performance. Many other projects address one or two of these elements but not all four simultaneously.

Application Scenarios

To ensure our research meets the demands of end users, our work is evaluated in two application scenarios—security and care. Space does not permit a detailed explanation of the tasks in each scenario; instead, we cite other works that include further information on the tasks and technology from each scenario.

Our security plan was developed with G4S Technology. The aim of this system was to have a robot monitor an indoor office environment and generate alerts when it observed prohibited or unusual events. We completed two security deployments in which a mobile robot routinely created models of the environment's three-dimensional (3-D) structure [4], objects [5], and people [6]; modeled their changes over time; and used these models to detect anomalous situations and patterns. For example, we developed robot behaviors to detect when a human moves through the environment in an unusual manner [6], to build models of the arrangement of objects on desks [7], and to check whether fire exits have been left open. Long-term deployments are essential for these services to gather sufficient data to build appropriate models.

Our robots have been operational for a combined duration of 104 days over four deployments, autonomously performing end-user-defined tasks and traversing 116 km in the process.

Our care scenario was developed with the Akademie für Alterforschung at the Haus der Barmherzigkeit. In this arrangement, the robot supported staff and patients in a large elder-care facility. We completed two care deployments in which a mobile robot guided visitors, provided information to residents, and assisted in walking-based therapies. In the care scenario, the robot serves users more directly, and, therefore, long-term system robustness is crucial, as is adapting to the routines of the facility. More information on this plan is available in [8] and [9].

(a) (b)

Figure 1. Two of the STRANDS MetraLabs SCITOS A5 robots in their application environments. (a) The robot *Bob* at G4S's Challenge House in Tewkesbury, United Kingdom. (b) The robot *Henry* in the reception area of Haus der Barmherzigkeit, Vienna.

Robot Technology

The systems reported in this article are developed in Robot Operating System (ROS), available under open-source licenses and binary packaged for Ubuntu (http://strands-project.eu/software.html) [2]. While the majority of our work is platform-neutral, all of our deployed systems are based on the MetraLabs SCITOS A5 robot (Figure 1). This is an industry-standard mobile robot capable of long run times (12 h on one charge) and autonomous charging. Our robots each have SICK S300 lasers in their bases (for localization, leg detection, and so on) and two Asus Xtion PRO RGB-D cameras, one at chest height pointing downward (for obstacle avoidance) and the other on a pan-tilt unit (PTU) above the robot's head. The SCITOS has an embedded computer with an Intel Core i7 processor with 8 GB of random-access memory (RAM), to which we networked two additional computers, each with an Intel Core i7 processor and 16 GB of RAM.

We also take advantage of the robot's need to regularly dock with a charging station by resetting the robot's position to this known location while it is docked.

The STRANDS Core System

The STRANDS Core System (Figure 2) is an application-neutral architecture for LTA in mobile robots. It is a combination of widely used components and components designed specifically for LTA. As mentioned previously, hardware and software robustness is essential for LTA. Hardware robustness is beyond the scope of our research; thus, we assume our software is running on an appropriate robot and computational platform. We address software component robustness through a mix of strategies. During development, we encourage components to be designed in a way that makes the minimal assumptions about the existence of other components and services (e.g., by checking service existence before running). We also pay particular attention to error handling to ensure component-local errors and exceptions do not propagate unnecessarily. This allows components and whole subsystems to be brought up and down automatically. At run-time, we use built-in ROS functionality to automatically relaunch crashed components, and most subsystems run only when required, thus saving the energy and processing power and reducing opportunities for errors. We also use run-time topic monitoring to detect problems (e.g., low publish rates) and trigger component restarts. Finally, we run a continuous integration server that tests components and the whole system in isolation, on recorded data, and in simulation.

The overall performance of a mobile robot is constrained by its localization and navigation systems, so we use widely adopted ROS packages to provide state-of-the-art performance. At the start of a deployment, we build a fixed map from laser scans, localize in it with adaptive Monte Carlo localization, and navigate using the dynamic window approach (DWA) over 3-D obstacle information [3]. See http://wiki.ros.org/navigation for details on these techniques. While our use of a fixed map appears at odds with LTA in a dynamic environment, our environments are dominated by static features (e.g., walls), which prevent the robot's localization performance from degrading. We also take advantage of the robot's need to regularly dock with a charging station by resetting the robot's position to this known location while it is docked. This limits localization drift to that which can occur during time away from the dock.

We manually build a topological map on top of the fixed continuous map. We place topological nodes at key places in the environment for navigation (e.g., either side of a door) or for tasks (e.g., by a desk to observe). The topological map from our 2015 security deployment is shown in Figure 3. Edges in the

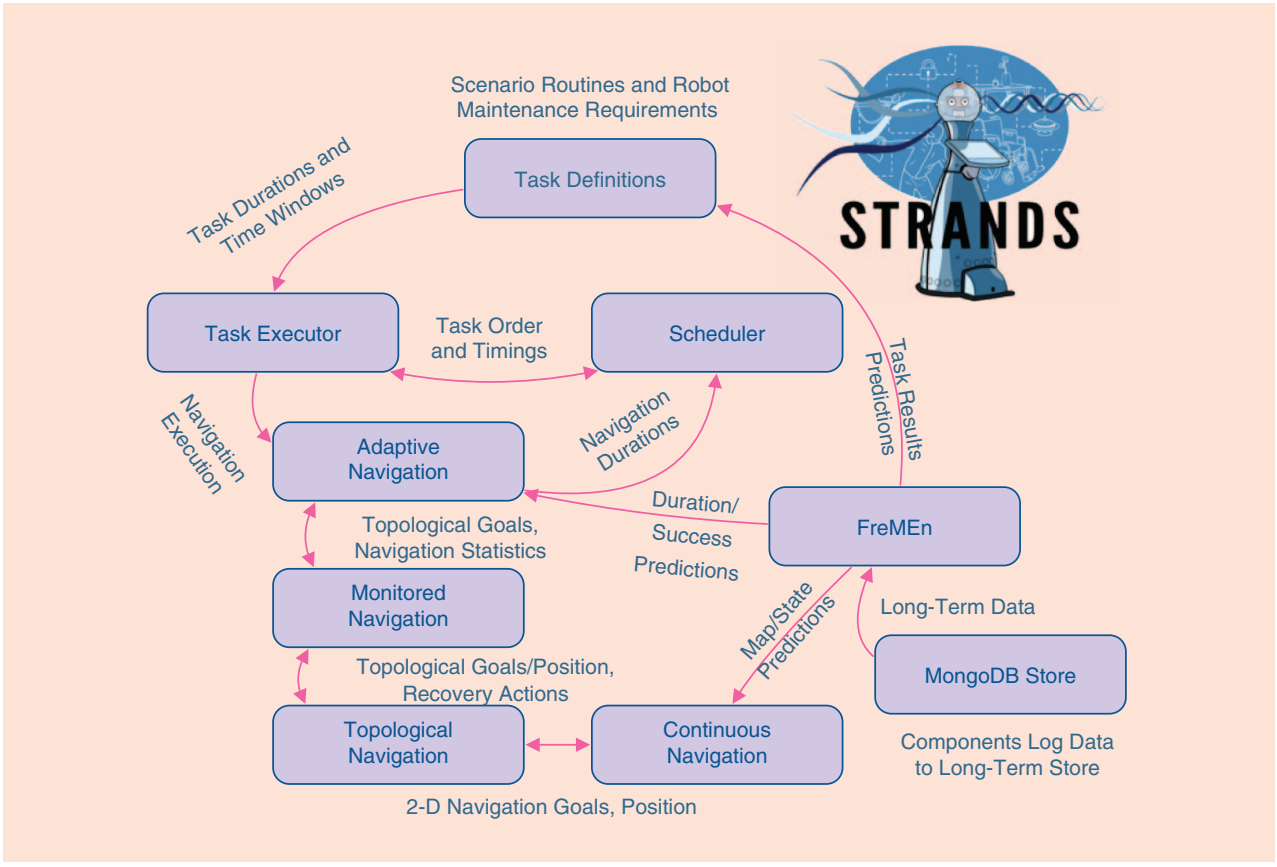


Figure 2. A schematic overview of the STRANDS Core System. 2-D: two-dimensional.

topological map are parametrized by the action required to move along them. In addition to DWA navigation, our system can perform door passing, docking with a charging station, and adaptive navigation near humans [10].

In our experience, navigation performance is a major determinant of the autonomous run time of a mobile robot. This is because navigation failures (e.g., getting stuck near obstacles) can result in the robot being unable to return to its charging station. The elements of the STRANDS Core System support LTA in the following ways: By constraining the robot's movements to the topological map, we are able to restrict navigation to known good areas of the environment. We additionally restrict movement by marking areas of the static map as "no go" zones that cannot be planned through. Despite these restrictions, navigation failures still occur due to environmental dynamics (e.g., people walking in front of the robot). Therefore, edge traversals in the topological map are executed by a monitored navigation layer that can perform a range of recovery actions in the event of failure (see the "Monitored Navigation" section). Topological route planning and execution is one area where our core system adapts to long-term experience, as described in the "Adaptive Topological Navigation" section.

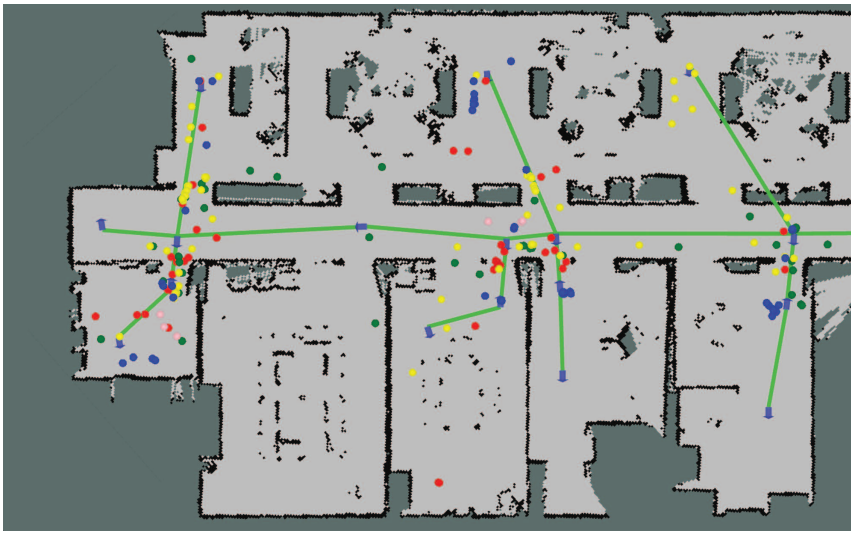


Figure 3. The map of the deployment area in Challenge House in Tewkesbury, United Kingdom, with the topological map superimposed. Also displayed are the locations where the robot successfully recovered from a navigation failure. Locations where the bumper was triggered are red, and green locations indicate nonbumper fails; the robot asked humans for help at these locations. Places where recoveries were performed by reversing along the previous path are marked in yellow, and recoveries performed by simply retrying are shown in blue.

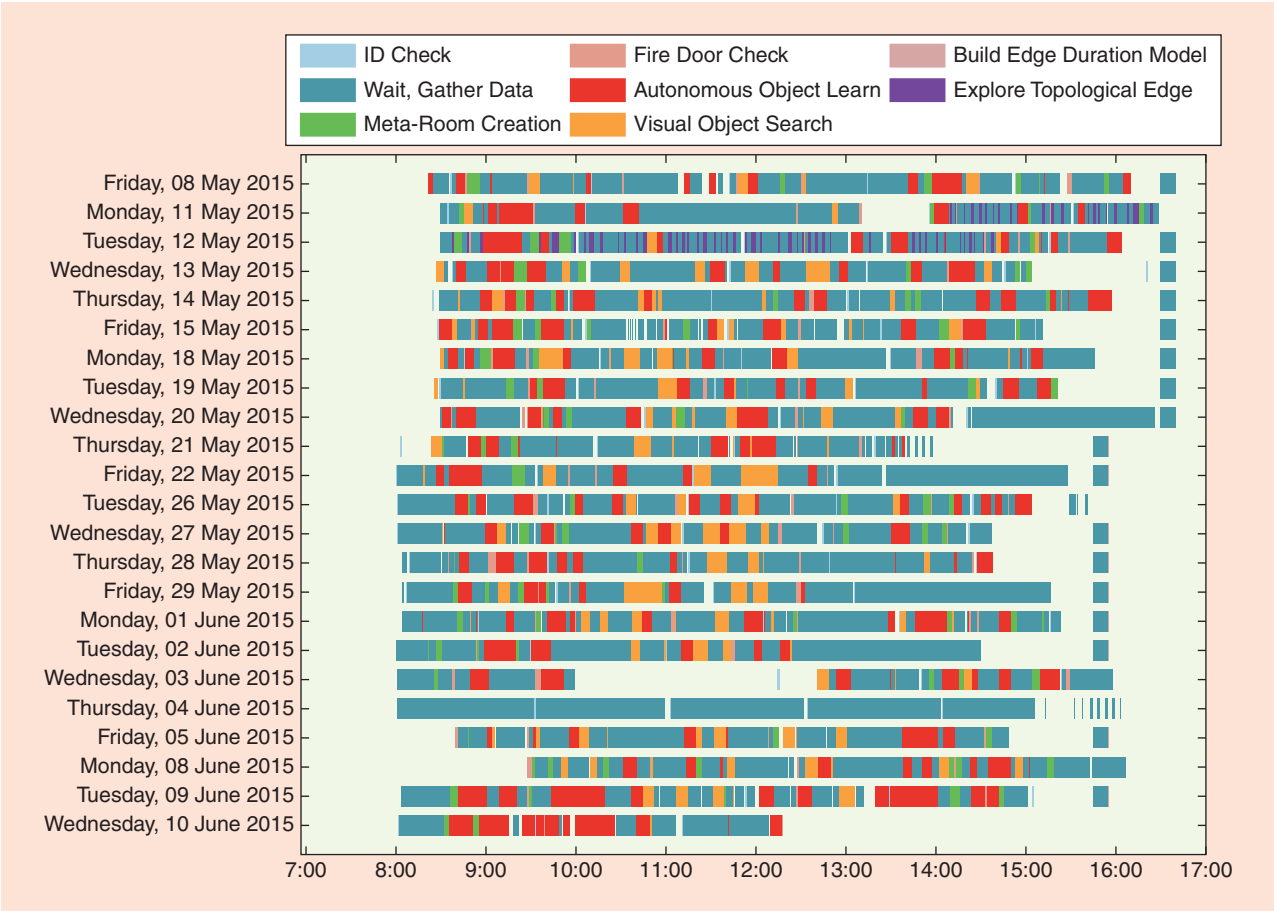


Figure 4. A plot of the tasks performed by the robot during the 2015 security deployment. White space represents times when the robot was not performing any tasks, which indicates that the robot was charging or a failure had occurred.

The main unit of behavior in our system is a task. Tasks represent something the robot can do (e.g., check whether a fire door is open, deliver information via a graphical user interface), and tasks have an associated topological location, a maximum duration, and a time window for execution. Our executive framework [11] schedules tasks to be executed within their time windows and manages task-directed navigation and execution. To prevent task failures from interfering with long-term operations, our framework detects task time-outs and failures, at which point it will stop or restart robot behaviors as necessary. Maintenance actions such as charging, batch learning, and database backups are all handled as tasks, allowing the executive framework to control most of the robot's behavior. This is essential for LTA as it enables the system to actively manage its limited resources. A plot of tasks from the 2015 security deployment can be seen in Figure 4.

The data observed and generated (e.g., as intercomponent communication) by an LTA system is crucial for both learning and for monitoring and debugging the system.

Our system relies on separate pipelines for perceiving different elements of its environment: real-time multiperson RGB-D detection and tracking [12], visual object instance and category modeling and recognition [13], and 3-D spatiotemporal mapping [4]. This article does not cover our work on perceptually challenging tasks. Instead, we refer readers to other sources where we have exploited these perception pipelines, e.g., [5], [7], and [10].

The data observed and generated (e.g., as intercomponent communication) by an LTA system is crucial for both learning and for monitoring and debugging the system. We therefore use tools based on MongoDB (http://wiki.ros.org/mongodb_store) to save ROS messages to a document-oriented database. Database contents (e.g., observations of doors being opened or closed) can then be interpreted by the Frequency Map Enhancement (FreMEn) component [14], which integrates sparse and irregular observations into spatiotemporal models representing (pseudo) periodic environment variations. These can be used to predict future environment states (see the "Adaptive Topological Navigation" section).

Table 1. LTA metrics from the first four STRANDS system deployments.

	Care 2014	Security 2014	Care 2015	Security 2015	Total
Total Distance Traveled (km)	27.94	20.64	23.41	44.25	116.24
Total Tasks Completed	1,985	963	865	4,631	8,444
Maximum TSL	7 d, 3 h	6 d, 19 h	15 d, 6 h	28 d, 0 h	
Cumulative TSL	20 d, 19 h	21 d, 0 h	27 d, 8 h	35 d, 3 h	104 d, 7 h
Individual Continuous Runs	18	18	5	2	43
A%	38.80%	18.27%	53.51%	51.10%	

Metrics

So far, we have performed two evaluation deployments for each of the security and care scenarios. For each deployment, we monitored overall system performance against two metrics: the total system lifetime (TSL) and the autonomy percentage (A%). The TSL measures how long the system is available for autonomous operation and is reset if the system experiences an unrecoverable failure or needs an unrequested expert intervention (i.e., something that cannot easily be done by an end user on site). The A% measures how long the system was actively performing tasks as a proportion of the time it was allowed to operate autonomously; in our deployments, this is typically restricted to office hours. The motivation of the A% is that it is of little value to achieve a long TSL if the system does nothing. Neither the TSL nor the A% measures the quality of the services being provided. As this article focuses on LTA, we restrict our presentation to task-neutral but LTA-specific metrics. End-user evaluations of task-specific performance are ongoing and will be published in the future (see [8] and [9] for early evaluations from the care scenario).

Table 1 presents our systems' LTA performance in 2014 and 2015. In 2014, we aimed for 15 days for the TSL; however, the longest run we achieved was seven days. Most of our system failures were caused by the lack of robustness of our initial software, leading to unrecoverable component behavior (crashes or deadlock states). This was fixed for our 2015 deployments by following the development approaches outlined in "The STRANDS Core System" section. In 2015, we targeted 30 days for the TSL, coming close with 28 days in the security deployment. This long run was terminated when the robot's motors failed to respond to commands, an issue that has since been fixed with a firmware update. In the 2015 deployments, most failures were due to computer-related issues beyond the direct contributions of the project (e.g., USB drivers, power cables, network problems, and so on). Of the seven runs in 2015, one run was ended due to user intervention (a decorator powered off the robot), two due to bugs in our software, and the remaining four due to faults in software or hardware beyond our components.

The variations across deployments in terms of the number of tasks completed and distance traveled are largely attributable to the different types of tasks performed by the robots and the different environments in which they were deployed. For example, information-serving tasks may take several minutes with very little travel, but door-checking tasks will be brief but will require the robot to travel both before and during the task.

Systems in the literature have delivered more autonomous time and distance cumulatively (i.e., accumulated across multiple robots and system runs), but we believe the 28-day run is the longest single continuous autonomous run of an indoor mobile service robot capable of multiple tasks. The most relevant comparison we can make is to the CoBots, which reported a total of 1,279.5 h of autonomy time, traversing 1,006.1 km [2]. This was achieved by four robots in 3,199 separate continuous autonomous runs over three years, at an average of 23 min and 0.31 km per run. They did not report the longest single continuous run (either in time or distance), but even an extremely long run for a CoBot would only be measured in hours, not days, because the CoBot does not have autonomous charging capabilities. In contrast, the STRANDS systems performed a total of 43 separate continuous runs, yielding a total of 2,545 h and 116 km over the four deployments, at an average of 2.7 km and 58 h 12 min per run. The varied durations of individual runs can be seen in Figure 5. Note that we use this data to provide a point of comparison.

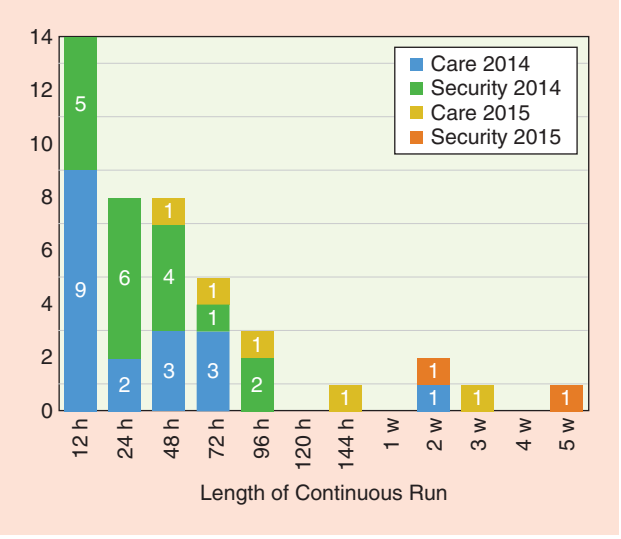


Figure 5. A histogram of individual continuous run lengths over the four STRANDS deployments.

Table 2. Classes of navigation failure, their associated recoveries, and the overall counts of successful and unsuccessful recoveries from these failures. Per-recovery counts are shown in Figure 6.

Failure	Recoveries	Successful	Unsuccessful	Total
Bumper pressed	Request help via screen and voice. Repeated request until recovered.	177	148	325
Navigation failure (no valid local or global path)	Sleep then retry; backtrack to last good position; request help via screen and voice. Repeated request until recovered.	707	993	1,700
Stuck on carpet	Increased velocities commanded to motors	16	247	263

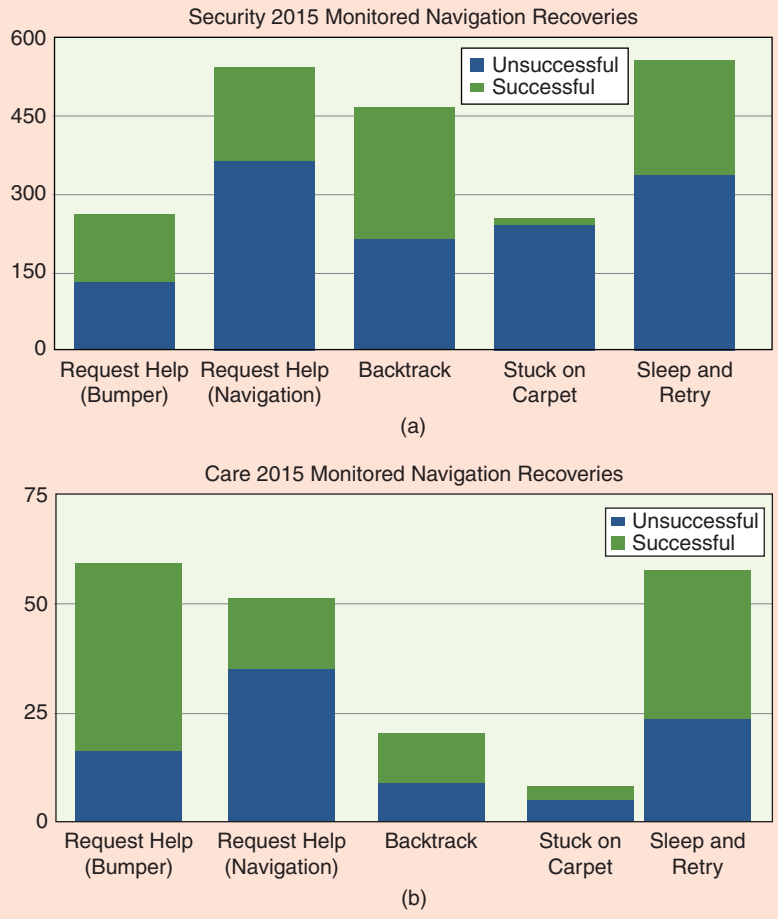


Figure 6. Per-recovery counts for the 2015 (a) security and (b) care deployments.

The two projects are targeting different metrics (i.e., total distance for CoBots and single-run duration for STRANDS); thus, the systems naturally have different performance characteristics.

Monitored Navigation

Given the huge variety of situations an LTA service robot will encounter, it is impossible to develop a navigation algorithm that will successfully account for all of them. We therefore developed a framework that executes topological navigation actions and monitors them for failure. If a failure is detected, then the framework steps through a list of recovery behaviors until either the navigation action completes successfully or the list is exhausted, in which case, failure is reported back to the calling component. Failure types can be mapped to specific lists of recoveries. When the robot's bumper is pressed, a hardware cut-off prevents it from moving forward, which means that the robot must ask to be pushed away from obstructions by nearby humans. If the local DWA planner fails to find a path, then simply clearing the navigation costmap to remove transient obstacles may suffice. We also developed a backtrack behavior that uses the PTU-mounted depth camera to sense backward while the robot reverses along the path it took to the failure location. This is triggered when navigation fails and clearing the costmap does not overcome the failure.

Table 2 and Figure 6 present the recovery behaviors used in our 2015 deployments. Successful recoveries are those that were not followed by another failure within one minute or 1 m of travel; otherwise, they are unsuccessful. A successful recovery may be preceded by any number of unsuccessful recoveries. A sequence of unsuccessful recoveries can come from the monitored navigation system as it attempts recoveries that then fail or from the task execution framework unsuccessfully trying to navigate the robot to another task after a previous failure. Figure 3 shows where all the successful recoveries from our 2015 security deployment occurred. They are largely clustered around areas where it

was difficult to navigate, such as near doors and close to desks. This novel approach contributed significantly to the LTA performance of our systems, as each recovered failure could have potentially caused the end of a continuous run.

Adaptive Topological Navigation

While monitored navigation helps the robot recover from navigation failures, it does not help it to avoid them. To avoid navigation failures in the future, the robot's navigation experience is aggregated into a Markov decision process (MDP) automatically built from the topological map [15]. Using the MDP allows the system to model uncertainty over the success of the robot traversing an edge in the map and its expected duration. By learning models for these success probabilities and durations online, the robot is able to continually adapt its behavior to the environment in which it is deployed. Every time the robot navigates an edge, the duration and success of the traversal is logged to the robot's database. These logs are processed by FreMEn to produce a temporal predictive model that allows the actions of the MDP to be assigned probabilities and travel durations appropriate for the time of execution [11]. This MDP is then solved for a target location to produce a policy for topological navigation that prefers low-duration edges with high success probabilities (see [15] for details). This improves the system's robustness by making it avoid areas where it previously encountered navigation failures. This is only possible in an LTA setting where the robot runs repeatedly in the same environment.

Predicting Human–Robot Interaction

In the Haus der Barmherzigkeit care facility, our robot acted as an information terminal, using its touch screen to present the weather, daily menu, news, and so on, to staff and residents with potentially severe dementia. This behavior was scheduled as a task at different topological nodes in the care home. As we did not know in advance the locations and times people would prefer to interact with the robot, we allowed it to adapt its routine based on long-term experience. To achieve this, each node in the topological map was associated with a FreMEn model that represented the probability of someone interacting with the robot's screen at a given time. This was built from logs of screen interactions stored in MongoDB. These FreMEn models were used to predict the likelihood of interactions at given times and locations. These predictions were used by the robot to schedule where and when it should provide information during the day.

The schedule must satisfy two contradicting objectives common to many online active-learning tasks: exploration (to create and maintain the spatiotemporal models) and exploitation (using the model to maximize the chance of interacting with people). Exploration requires the robot to visit locations at times when the chance of obtaining an interaction is uncertain. Exploitation requires scheduling visits to maximize the chance of obtaining interactions. To tackle this trade-off, the schedule was generated using Monte Carlo sampling

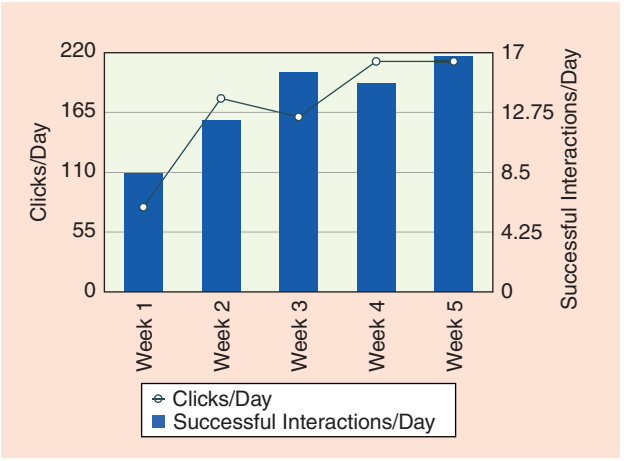


Figure 7. The results of the robot selecting interaction times and locations using FreMEn models learned during the 2015 care deployment.

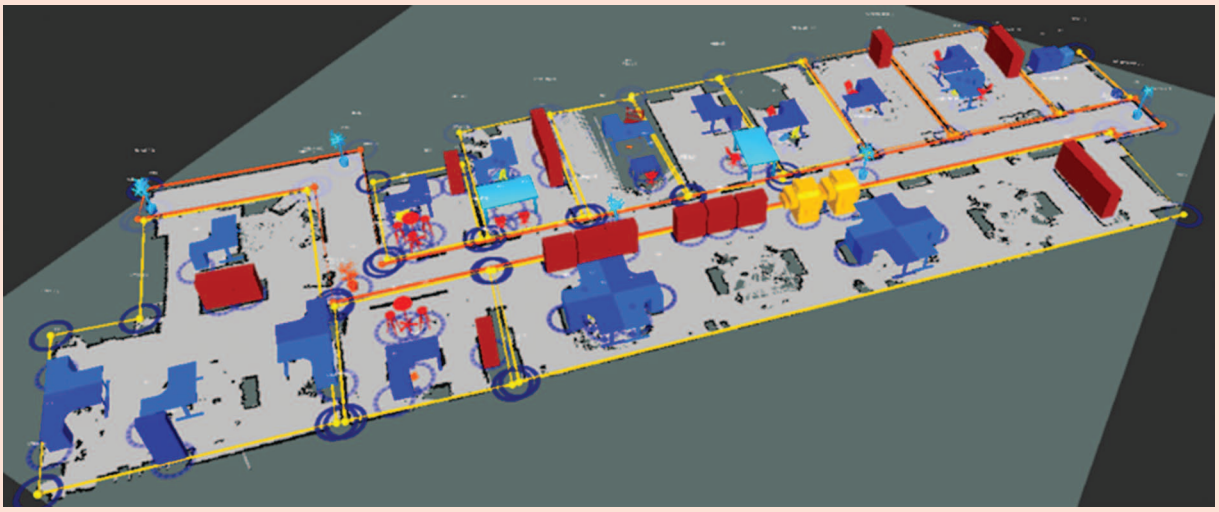
from the location/time pairs according their FreMEn-predicted interaction probability (exploitation) and entropy (exploration). For more details see [16].

Figure 7 shows that the robot was able to increase the number of successful interactions (i.e., when information was offered and someone interacted with the screen) on average, per day, over the course of its deployment. Although we have no control group to compare against, our on-site observations indicate that the robot's choices had a positive effect. This demonstrates the ability of the system to improve its application-specific behavior from long-term experience.

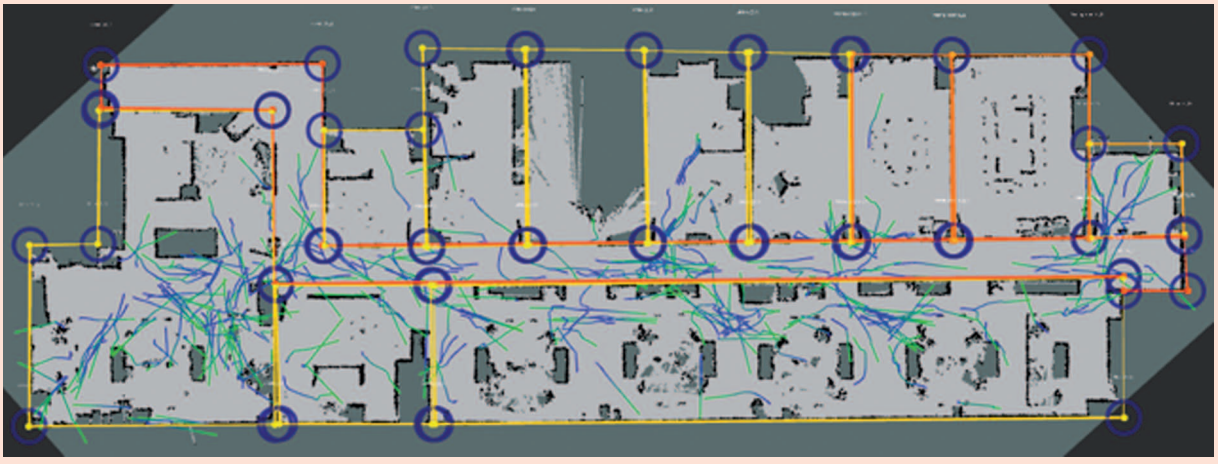
Activity Learning

In our security scenario, the robot had to learn models of normal human activity and then raise an alert if an observation deviated from this. We explored activity learning using walking trajectories [see Figure 8(b)]. Over the 2015 security deployment, the robot detected 42,850 individual trajectories. As described in [6], we used qualitative spatiotemporal activity graphs (QSTAGs) to generalize from individual trajectories to spatial and temporal relations between trajectories and landmarks in a semantic map [see Figure 8(a)]. QSTAGs ignore minor quantitative variations across trajectories but capture larger, qualitative changes. Every night, the robot created QSTAGs for a subset of all trajectories observed during the day (based on their displacement ratio). It then clustered these to create classes of movement activities. Some examples of the results can be seen in Figure 9.

During the day, an observation of a trajectory sufficiently far from any cluster center triggered a task to approach the tracked human and request confirmation of their identity using a card reader. To enable a fast response, it is important that the robot can accurately match the start of the trajectory to a cluster. Table 3 shows how the accuracy of predicting the cluster of a trajectory from an initial segment (20%) improves as more data is gathered over the robot's lifetime. This provides another example of how a robot can improve its application-specific performance once it can operate over long periods.



(a)



(b)

Figure 8. (a) The manually created semantic map from the 2015 security deployment. (b) Example human trajectories with lengths close to the average trajectory length of 2.44 m. Also pictured are the manually annotated room regions used for task planning, where the circles indicate the vertices of region polygons that were used to annotate the types of regions in the environment. The yellow regions indicate offices, the dark orange regions indicate meeting rooms, the light orange region is a kitchen, and the red region is a corridor. Blue to green lines indicate direction of movement.

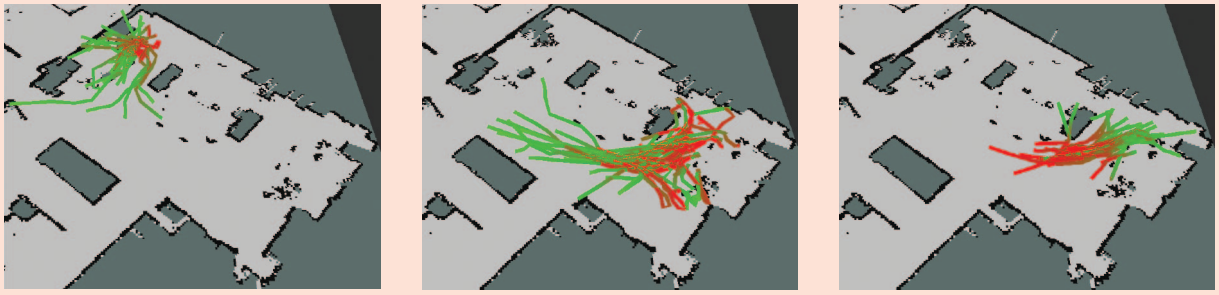


Figure 9. Trajectories belonging to three learned clusters in the region at the bottom left of Figure 8 (the direction of motion is red to green). These can be interpreted as two clusters of a desk-approaching activity and one cluster of a desk-leaving activity.

Table 3. Accuracy of activity cluster prediction on week 5 data from partial input trajectories.

Training Weeks (Number of Trajectories)	*Number of Clusters*	Recall	Precision	F-Score
Week 0 (342)	9	0.24	0.72	0.29
Weeks 0–1 (511)	12	0.43	0.54	0.44
Weeks 0–2 (707)	12	0.43	0.56	0.43
Weeks 0–3 (811)	10	0.43	0.71	0.49
Weeks 0–4 (1,016)	14	0.48	0.63	0.53

Conclusions and Future Work

The STRANDS Core System features a mix of design- and runtime approaches that allow it to deliver LTA in everyday environments. A key strategy for delivering long-term robustness is the monitoring of system behavior, from the individual component level up to navigation and task behaviors, as well as the ability to restart system elements on demand. This allows the system to cope with unexpected situations both internally and in the external environment. The system also uses the long-term experience of failures to learn to avoid these failures in the future. This approach improves navigation ability, and we hope to generalize this to other parts of the system. While these features provide a fundamental ability to operate autonomously for long durations in everyday environments, our robots currently have no way to manage failures that are more catastrophic, harder to predict, or both. For example, our systems have suffered from computer component failure and subtle networking issues. In the future, we would like to look at the use of redundancy and online reconfiguration, such as substituting a failing software or hardware component, coupled with more general failure detection approaches. Both of these topics have been extensively researched in robots and other systems.

Our robots are able to learn online from lengths of experiences from which no other robots to date have access. The discussed results demonstrate what we have always known from machine learning: More data improves performance. The novel element here is that a robot must be able to operate longer to gather additional data and must be able to make active choices about what data is gathered.

In the future, we will focus on the robot's ability to understand human activities, which are the major causes of environmental dynamics at most scales, and to actively close gaps in the knowledge it has already obtained from weeks of autonomous run time.

Acknowledgments

We wish to thank our project reviewers and project officers for their contributions to our research: Luc De Raedt, James Ferryman, Horst-Michael Gross, Olivier Da Costa, and Juha Heikkilä. The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement No. 600623, STRANDS.

References

[1] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robotics Automation*, Anchorage, AK, 2010, pp. 300–307.

[2] J. Biswas and M. Veloso, "The 1,000-km challenge: Insights and quantitative and qualitative results,"*IEEE Intelligent Syst*., vol. 31, no. 3, pp. 86–96, May 2016.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proc. IEEE Int. Conf. Robotics Automation*, Detroit, MI, 1999, pp. 1999–2005.

[4] R. Ambrus, J. Ekekrantz, J. Folkesson, and P. Jensfelt, "Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 5678–5685.

[5] T. Faeulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *IEEE Robot. Autom. Lett*., vol. 2, no. 1, pp. 26– 33, Jan. 2016.

[6] P. Duckworth, Y. Gatsoulis, F. Jovan, N. Hawes, D. C. Hogg, and A. G. Cohn, "Unsupervised learning of qualitative motion behaviours by a mobile robot," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, Singapore, 2016, pp. 1043–1051.

[7] L. Kunze, C. Burbridge, M. Alberti, A. Tippur, J. Folkesson, P. Jensfelt, and N. Hawes, "Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 2910–2915.

[8] D. Hebesberger, C. Dondrup, T. Körtner, C. Gisinger, and J. Pripfl, "Lessons learned from the deployment of a long-term autonomous robot as companion in physical therapy for older adults with dementia: A mixed methods study," in *Proc. IEEE Int. Conf. Human–Robot Interaction,* New Zealand, 2016, pp. 27–34.

[9] D. Hebesberger, T. Körtner, J. Pripfl, and M. Hanheide, "What do staff in eldercare want a robot for? An assessment of potential tasks and user requirements for a long-term deployment," in *Proc. Workshop on Bridging user needs to deployed applications of service robots,* Hamburg, Germany, 2015.

[10] C. Dondrup, N. Bellotto, M. Hanheide, K. Eder, and U. Leonards, "A computational model of human-robot spatial interactions based on a qualitative trajectory calculus," *Robotics*, vol. 4, no. 1, pp. 63–102, Mar. 2015.

[11] L. Mudrová, B. Lacerda, and N. Hawes, "An integrated control framework for long-term autonomy in mobile service robots," in *European Conf. Mobile Robots*, Lincoln, UK, 2015, pp. 1–6.

[12] O. H. Jaffari, D. Mitzel, and B. Leibe, "Real-Time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *Proc. IEEE Int. Conf. Robotics Automation*, Hong Kong, 2014, pp. 5636–5643.

[13] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "RGB-D object modelling for object recognition and tracking," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 96–103.

[14] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *Proc. IEEE Int. Conf. Robotics Automation*, Hong Kong, 2014, pp. 3706–3711.

[15] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 1511–1516.

[16] J. M. Santos, T. Krajnk, J. P. Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4-d spatio-temporal maps," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 684–691, July 2016.

Nick Hawes, School of Computer Science, University of Birmingham, United Kingdom. E-mail: n.a.hawes@cs.bham.ac.uk.

Chris Burbridge, School of Computer Science, University of Birmingham, United Kingdom. E-mail: c.j.c.burbridge@cs.bham.ac.uk.

Ferdian Jovan, School of Computer Science, University of Birmingham, United Kingdom. E-mail: fx345@cs.bham.ac.uk.

Lars Kunze, School of Computer Science, University of Birmingham, United Kingdom. E-mail: l.kunze@cs.bham.ac.uk.

Bruno Lacerda, School of Computer Science, University of Birmingham, United Kingdom. E-mail: b.lacerda@cs.bham.ac.uk.

Lenka Mudrová, School of Computer Science, University of Birmingham, United Kingdom. E-mail: lxm210@cs.bham.ac.uk.

Jay Young, School of Computer Science, University of Birmingham, United Kingdom. E-mail: j.young@cs.bham.ac.uk.

Jeremy Wyatt, School of Computer Science, University of Birmingham, United Kingdom. E-mail: jlw@cs.bham.ac.uk.

Denise Hebesberger, Akademie für Altersforschung am Haus der Barmherzigkeit, Austria, and Donau-Universitaet Krems, Austria. E-mail: Denise.Hebesberger@hausderbarmherzigkeit.at.

Tobias Körtner, Akademie für Altersforschung am Haus der Barmherzigkeit, Austria, and Donau-Universitaet Krems, Austria. E-mail: tobias.koertner@altersforschung.ac.at.

Rares Ambrus, KTH Royal Institute of Technology, Sweden. E-mail: rares.ambrus@gmail.com.

Nils Bore, KTH Royal Institute of Technology, Sweden. E-mail: nbore@kth.se.

John Folkesson, KTH Royal Institute of Technology, Sweden. E-mail: johnf@kth.se.

Patric Jensfelt, KTH Royal Institute of Technology, Sweden. E-mail: patric@kth.se.

Lucas Beyer, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: beyer@vision.rwth-aachen.de.

Alexander Hermans, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: hermans@vision.rwth-aachen.de.

Bastian Leibe, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: leibe@umic.rwth-aachen.de.

Aitor Aldoma, Technische Universität Wien, Austria. E-mail: aldoma@acin.tuwien.ac.at.

Thomas Fäulhammer, Technische Universität Wien, Austria. E-mail: faeulhammer@acin.tuwien.ac.at.

Michael Zillich, Technische Universität Wien, Austria. E-mail: zillich@acin.tuwien.ac.at.

Markus Vincze, Technische Universität Wien, Austria. E-mail: vincze@acin.tuwien.ac.at.

Eris Chinellato, Faculty of Science and Technology, Middlesex University London, United Kingdom. E-mail: e.chinellato@mdx.ac.uk.

Muhannad Al-Omari, University of Leeds, United Kingdom. E-mail: scmara@leeds.ac.uk.

Paul Duckworth, University of Leeds, United Kingdom. E-mail: scpd@leeds.ac.uk.

Yiannis Gatsoulis, University of Leeds, United Kingdom. E-mail: y.gatsoulis@leeds.ac.uk.

David C. Hogg, University of Leeds, United Kingdom. E-mail: d.c.hogg@leeds.ac.uk.

Anthony G. Cohn, University of Leeds, United Kingdom. E-mail: A.G.Cohn@leeds.ac.uk.

Christian Dondrup, University of Lincoln, United Kingdom. E-mail: cdondrup@gmail.com.

Jaime Pulido Fentanes, University of Lincoln, United Kingdom. E-mail: jpulidofentanes@lincoln.ac.uk.

Tomáš Krajník, University of Lincoln, United Kingdom. E-mail: tkrajnik@lincoln.ac.uk.

João M. Santos, University of Lincoln, United Kingdom. E-mail: jsantos@lincoln.ac.uk.

Tom Duckett, University of Lincoln, United Kingdom. E-mail: tduckett@lincoln.ac.uk.

Marc Hanheide, University of Lincoln, United Kingdom. E-mail: marc@hanheide.net.