Execution Monitoring for Long-Term Autonomous Mobile Robots in Outdoor Scenarios

Tim Bohne[1] and Benjamin Kisliuk[1]

Abstract— **This work attempts to reduce the barriers towards long-term autonomy of mobile outdoor robots by identifying and classifying key difficulties in such a context and developing a fully integrated monitoring and resolution framework capable of overcoming typical limitations for those systems. Experimental evaluation of the proposed framework in a simulation environment indicates a drastically improved resilience with respect to the identified challenges.**

I. Introduction

With robotic technology maturing, a focus is set on long-term autonomous (LTA) deployments in highly dynamic, partially observable real-world environments such as agriculture, where knowledge is incomplete. Goal-oriented acting requires intelligent planning, but even good plans do not always work out as expected. Thus, it is critical to ensure a certain level of robustness during plan execution, with problems ideally being solved by the robot as they arise. Consequently, execution monitoring techniques that address problematic situations by means of recovery mechanisms must be an essential component of robot architectures.

II. Long-Term Autonomous Mobile Robots

All three attributes - long-term, autonomous, and mobile - have the potential of dramatically increasing the complexity and the risk for failures of the system. To be able to test our approaches, we define *long-term* as a period of time, spanning multiple mission supply cycles, during which inherent dynamics of the environment and the system itself can be expected. [1] Likewise, *autonomy* is defined as levels on a spectrum by the necessity of human intervention with respect to the core functionalities of a system. To enable LTA, there is no single problem to solve, but rather a series of solutions being integrated and interacting. Two of those, namely autonomous energy supply and a dynamic battery monitoring systems, are described in the same work.

III. Related Work

A substantial majority of the research concerning LTA deals with indoor service scenarios ([2][3][4]). The examples available for outdoor scenarios tend to focus on very specific aspects of LTA, such as environmental change and place recognition ([5][6][7]). In addition, general problem handling capabilities are often tightly tethered to specific systems and environments and therefore hardly reusable. What is perhaps a bit underexposed in the literature is the big picture of LTA mobile robotic applications in outdoor scenarios, not focusing on specific aspects but approaching an integrated functioning system. More precisely, work that discusses common problems in LTA scenarios, regardless of the specific application, and how to address them with execution monitoring. The present work is based on the first author's Master's thesis [8], which provides a comprehensive and practically motivated development of such an execution monitoring system for a mobile robot, concretely applied in an agricultural plant inspection task. It serves as a condensed summary of the main findings, focusing on high-level failure monitoring and management, and is intended as a step towards a generic framework that is nonetheless concretely applicable in the previously proposed LTA scenario. [1]

IV. Challenges for Long-Term Autonomy

There are numerous potential hinderances for LTA systems that can cause a failure and prevent the system from continuing its task. The following three constraints precisely define the ones considered in this work:

1) *It can practically occur in outdoor scenarios, e.g. [1].*
2) *It can prevent the smooth functioning of an LTA system or affect the quality of its results.*
3) *It can be detected by monitoring methods and subsequently solved or communicated.*

This definition contains certain implicit assumptions. The first constraint, for instance, implies that these problems should be relatively likely to occur. Furthermore, the third constraint assumes that the system is still fundamentally functioning (recoverable in principle). Without claiming to be exhaustive, the following is a list of challenges that fulfill the above restrictions and are thus worthy of investigation:

- ***power management*** - *unexpected energy consumption*
- ***charging failure*** - *unsuccessful docking*
- ***extreme weather*** - *e.g. storm, heavy rain, extreme cold*
- ***natural dynamics*** - *e.g. day, night*
- ***sensor (perception) failure***
- ***perceptual aliasing issue***
- ***data management*** - *e.g. sensor data processing failure*
- ***lost connection*** - *WiFi, RTK-GNSS, internet etc.*
- ***navigation failure*** - *obstacles (static, dynamic)*
- ***sustained recovery*** - *no return to normal operation*
- ***inaccurate localization*** - *IMU, odometry, GNSS*
- ***mapping error***
- ***plan deployment failure***

The DFKI Niedersachsen is sponsored by the Ministry of Science and Culture of Lower Saxony and the Volkswagenstiftung. The paper describes work carried out in the context of the funded projects PORTAL (BMEL) and ZLA (NiMWK, Volkswagenstiftung).

[1] Plan-Based Robot Control, German Research Center for Artificial Intelligence, Osnabrück, Germany `firstname.lastname@dfki.de`

This list is the result of a tripartite analysis: Observations in real-world experiments, problems that are obvious and do not require justification, and challenges that have been encountered in LTA experiments in the literature. Generally, potential barriers for LTA can be classified into three categories of increasing negative impact on the system:

1) *Contingency: The robot recognizes a problem and is able to solve it.*
2) *Catastrophe: The robot recognizes a problem, is unable to solve it, and calls an operator for help.*
3) *The robot has a problem, does not recognize it and therefore cannot solve or communicate it.*

Type (1) is the ideal case and accordingly the ultimate goal of all efforts to implement LTA in practice. Type (2) is already a step forward, because problems are at least recognized and can be communicated, which is the minimum requirement to guarantee a certain robustness with respect to the problems. In a baseline scenario without any monitoring, the problems can be naturally classified as type (3). Part of the goal of this work is to shift the identified challenges to another category and thereby improve the utility of the system, i.e., to solve them (1), or at least to enable the robot to recognize them with execution monitoring approaches and request help (2).

V. Plan Execution and Monitoring

A key aspect of dealing with the introduced issues is that the robot will not be able to complete its missions without occasionally preempting the execution of the high-level task plan. As Harris et al. remark, even well-crafted plans may fail, essentially when the situations encountered do not match prior expectations. [9] Execution monitoring enables a robotic system to recognize and classify such situations caused by unexpected internal (robot) or external (environment) conditions [10] and provide recovery options. [11] If the behavior of a robotic system is plan-based, this is very meaningful for monitoring because the current state of plan execution provides context and thus certain expectations. [12] These properties make universal monitoring (introspection / extrospection) and resolution methods a highly relevant building block for robust LTA. [13][3] In case of a problem, the robot needs to be able to save the current state of the plan execution and continue precisely with this state after the reason for the interruption has been resolved, which is far from trivial since the original plan may no longer be applicable, e.g., due to preconditions of actions and unplanned resource consumption. However, if the interruption of the plan involves returning to the base station, such a recovery can always be accompanied by recharging. All other cases are covered by the integrated battery monitoring module described in [1].

A. Execution Monitoring State Machine Architecture

Plan execution, i.e., acting and monitoring, is modeled as a high-level hierarchically structured state machine (cf. Fig. 1) implemented using the SMACH[1] library. Unlike many approaches in the literature [12], the capability for a certain degree of fault tolerance is directly integrated into the control architecture. The state machine coordinates the entire robot operation and is composed of the states `NORMAL_OPERATION` (represented by an embedded state machine named `OPERATION`, visualized in Fig. 2, as well as the two parallel running monitoring states), `CONTINGENCY`, `CATASTROPHE`, and `SHUTDOWN`.

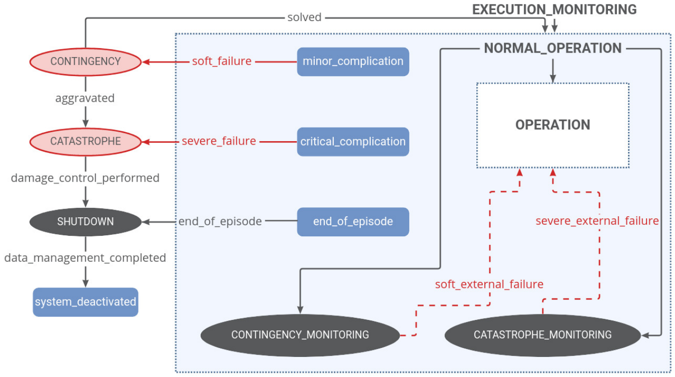

Fig. 1: Hierarchical State Machine (High-Level)

The parallel running monitoring states (connected to monitoring nodes for each of the challenges listed in section IV) are used to interrupt the robot's operation at any time when a problematic situation is detected (dashed arrows). In this case, depending on severity, the respective procedure interrupts `OPERATION` and triggers a transition to the appropriate handling state. For instance, the outcome `minor_complication` leads to a state transition to `CONTINGENCY` where the issue is addressed (cf. category (1) in sec. IV). If it is able to solve the issue, `NORMAL_OPERATION` resumes (cf. "solved"). In case of `critical_complication` or if the robot is not able to solve the problem (cf. "aggravated"), it ends up in the `CATASTROPHE` state and the human operator is notified (type (2)). The following example will illustrate the concepts. During runtime, the battery does not discharge precisely according to the plan (e.g. fluctuations due to temperature). The power management monitoring node would initiate a transition to `CONTINGENCY` when it detects that the battery is already too low to complete the plan until the next charge stop, but the robot is still able to recover, i.e., drive back to its base, recharge, and continue the plan execution. However, it would proceed to `CATASTROPHE` if it detects that the battery is so low that it is not possible to reach the base and recharge, even with an immediate return. This leads to an event message to `CATASTROPHE_MONITORING`, then to an external failure, which interrupts `OPERATION` with a `critical_complication` and causes the high-level state machine to transition to `CATASTROPHE`. The handling node would then shut down the robot after communicating the problem. Finally, there is also the case that first a contingency is launched as the problem still seems to be solvable based on current estimations and then it turns out that it in fact is not ("aggravated"). During the state `OPERATION`,

[1] ROS-independent Python library to build hierarchical state machines [14]

i.e., in the embedded state machine visualized in Fig. 2, the robot can be in one of two self-explanatory states: `IDLE` and `EXECUTE_PLAN`. In conclusion, there are two types of problems: Low-level problems that are detected directly, e.g., by simple error treatment, and problems that are detected by the monitoring nodes (cf. `external_problem`) with the consequence that execution is interrupted and the problem is solved superordinately. Essentially, in one long-term episode, the robot should stay in `NORMAL_OPERATION` as long as the episode is running and no problem has occurred.

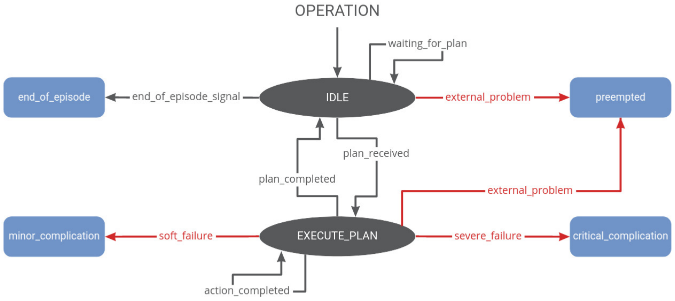

Fig. 2: Embedded State Machine (Low-Level)

B. Solutions for LTA Challenges

Our approach aims to have a resolver node for each of the identified challenges. While in the case of contingencies these nodes actually refer to problem solving, in catastrophe cases they are more for damage control. Since the focus of this work is on detection, the operational fallback is to request assistance from a human operator. [15] Nevertheless, within the list of identified issues, there is only one that explicitly leads to a catastrophe condition (power management failure). All other problems start with a contingency and therefore launch at least a simple heuristic that attempts to resolve the issue. Surprisingly, simple workarounds often lead to success, such as waiting for a short time, restarting a component, changing the position slightly or repeating the task. Despite deterministic repetition of the same operation, different results can be expected due to the presence of sufficient nondeterminism (dynamics) in the environment [15]. Detailed explanations of the simulation of the presented LTA challenges, the monitoring solutions developed for each, and available resolution procedures can be found in [8].

VI. Experiments and Evaluation

The scheme is essentially the same for all potential LTA challenges in the upcoming experiments: Fault simulation by publishing on the respective ROS topic, followed by a triggered monitoring procedure that should detect the problem, interrupt normal operation and initiate an appropriate remediation. Various metrics can be used to evaluate the performance of an LTA system: Hawes et al. [3] suggest the *autonomy percentage* to examine how actual autonomous acting relates to idle time. Steinberg et al. [16] introduce the metric of time between undesired human interventions, i.e., situations in which the robot should be able to recover, as opposed to situations in which human intervention is expected. Applied to this work, this refers to problems for which specific solutions have been proposed but have not been successful. This would manifest in simulated contingency situations ending in a catastrophe. Furthermore, they propose a metric for information sharing, i.e., the percentage of time the robot communicates meaningful information to the operator. Based on the developed architecture, this boils down to a question of reliability, which is evaluated in section VI-B. If a problem is properly identified, meaningful information will be provided with certainty, at least in the sense of information that led to the detection of the problem.

A. Classification of the Presented Challenges

Initially, the identified LTA challenges are classified according to their severity based on their evaluation in simulation together with a comparative contextualization of the practical case. More interesting in terms of classification is the present state of the physical robot system, i.e., no monitoring for the identified issues. In this case, catastrophe refers to battery failure or timeout and is thus equivalent to mission abort. Despite the evident potential for mission failure among some of the identified problems, it is systematically investigated how the system responds to these failure cases in simulation without the developed monitoring procedures. This is critical to verify the expected behavior of the simulated fault cases. The findings of the classification are summarized in Tab. I. Crucially, any problem category that has a "✓" or "(✓)" in the catastrophe column has the potential to abort the mission without monitoring solutions. Assuming persistent failures, "✓" means that mission abort is a certainty, while "(✓)" stands for the possibility. Another perspective is that "✓" refers to catastrophes caused directly by the problem, whereas "(✓)" implies indirect causation. Furthermore, the remaining problems do not interrupt the mission, but they do invalidate the results both in practice and simulation. There is only one exception: Drastic weather changes in the simulation. The actual occurrence of such events is not simulated, only the information of their presence. However, since the practical relevance is obvious, there is no need for further simulation. Thus, all of the issues identified have the potential to render LTA deployments worthless in real-world scenarios, highlighting the importance of addressing all of them.

problem	**contingency & catastrophe** *(with monitoring)*	**catastrophe** *(without monitoring)*		**potential to render mission worthless**	
	sim / prac	sim	prac	sim	prac
power_management	✓	✓	✓	✓	✓
charging_failure	✓	✓	✓	✓	✓
drastic_weather_change	✓	✗	(✓)	✗	✓
sensor_failure	✓	✗	✗	✓	✓
data_management	✓	✗	✗	✓	✓
lost_connection	✓	(✓)	(✓)	✓	✓
plan_deployment_failure	✓	✓	✓	✓	✓
navigation_failure	✓	(✓)	(✓)	✓	✓
incorrect_localization	✓	(✓)	(✓)	✓	✓

TABLE I: Classification in Terms of Severity

B. Evaluation of the Monitoring Framework

Now the natural question is to what extent the monitoring framework improves the situation. For a meaningful evaluation, we perform an LTA episode and randomly simulate the occurrence of issues from the set of identified problem categories. Since it is always known which reaction is expected after a certain simulation, the expected can be compared with the observed. The following results are based on a frequency of $1250s$, i.e., a random problematic situation occurs every $1250s$. However, this is only a lower bound because some of the simulations can only occur under certain circumstances and there are never two simultaneous fault simulations. The frequency was not chosen too high, so that the robot can still perform tasks (e.g. `drive_to`, `return_to_base`, `charge` and `scan`) and is not only occupied with error handling. Experimentally, it turned out that a time per run of 5 hours is sufficient to achieve a certain validity, since several mission and charge cycles occur and there is enough room for error simulation. Obviously, the number of missions that take place during this period depends on the plan that defines such a mission. The complete plan underlying the experiments, executed in a loop, can be found in [8]. The rationale behind it is that it satisfies the constraints for an LTA operation [1] and provides a variety of situations (plausible representation of real-world deployment). The 5-hour episode was repeated 10 times, on the one hand to endow the conclusions with some significance, but on the other hand also to be able to judge how many of these runs ran successfully to the end. Ideally, there should be no missions that end in a catastrophe, since only problems that are in principle solvable are simulated for this experiment. In order for the runs to be comparable, the random selection of failure simulation was initialized with the same seed.

Of 10 runs, 8 were completed successfully. The two aborted runs are the result of docking failures that could not be resolved. This is not a flaw of the monitoring framework, which correctly identified the issues, but rather a matter of robustness of the integrated docking solution presented in [1]. The first thing to consider is the expected response to error cases per run in percent. Ideally, this would be 100%, which would mean that the monitoring detects exactly all simulated problems and nothing beyond, which was the case in $5/10$ runs. This proportion is composed of correct contingency cases, i.e., when exactly the expected condition occurs, which means that the monitoring worked correctly, and correct non-contingency cases, where there is no contingency expected and none occurs. In general, across the problem categories, there are some conditions that are somewhat problematic but do not cross the boundary of contingency. Thus, the latter category is used to verify that the monitoring solutions are not configured too restrictively to ensure that the robot does not constantly try to solve imaginary problems. The remaining unexpected responses are composed of false positives, false negatives, and finally unexpected contingencies, i.e., cases where a contingency situation is expected due to a simulated failure but a different type of problem is detected. It is very crucial to note that false positives and unexpected contingencies are not necessarily a fault of the monitoring framework or its configuration. A significant majority of the cases is related to problems in the simulation that were not simulated as part of an experiment, but actually occur and are thus correctly detected. In order to actually evaluate the performance of the monitoring framework and not include other aspects such as the robustness of the docking solution, these correctly identified but not simulated problems are manually removed from the results. On average, the number of simulated problem cases is 13.75 per episode. The average proportion of expected responses to these error cases across all runs is 95.89%, which highlights a reasonable reliability of the system. The few unexpected responses consist of some false positives with respect to localization problems. Apparently, some of the monitoring approaches are configured to be too sensitive. In addition, there is a total of two false negatives for the same simulated problem, namely divergence between odometry and GNSS estimates of the total distance traveled. This is most likely due to the fact that the way this is simulated is not perfect for all circumstances and should be fairly easy to resolve. Moreover, on average, the robot completed 3 missions with a total of 88.75 tasks and required an average of 10.38 charge cycles. Per episode, the robot traveled an average total distance of 967.1 meters in 5.06 hours, estimated from odometry data. The autonomy percentage illustrates that the majority of the runtime is spent on actual autonomous tasks, on average 96.28%.

VII. Conclusion and Future Work

Practically relevant challenges to the LTA of mobile outdoor robots were identified, classified in terms of their potential impact and implemented in simulation. Based on the results, it can be concluded that the proposed framework drastically improves the resilience with respect to the identified challenges. The initial goal of assigning each of the identified problems to a different category (cf. section IV) was achieved, and the experiments demonstrate a certain level of reliability. Even if an episode fails, as in the two cases of experimental evaluation, catastrophe conditions are always communicated to the human operator, which is a huge improvement. The monitoring and resolution approaches are incorporated into a generic framework with as few assumptions about specific systems and scenarios as possible. While the context and many of the challenges studied are specific to outdoor scenarios, the framework itself is generally applicable to indoor scenarios. Since the system was developed within ROS, universality ends when considering other middlewares, although the concepts in this work should apply to non-ROS systems as well.

In total, there are 48 problem instances that can be explicitly simulated, many other cases that are explicitly addressed by monitoring procedures, and numerous potential problems that are implicitly covered. The overall architecture has proven to be suitable for this type of non-nominal plan execution. In particular, due to the parallel running monitoring nodes, the tight coupling of acting and monitoring in general, and the modularity that enables extensibility and reconfigurability. For a robotic system to use the monitoring framework, the embedded `OPERATION` state machine would have to be replaced by the system's own operational model with two basic assumptions. First, it must communicate predefined information about the mode of the system via a ROS topic. Second, all active targets should be interruptible via a specific

ROS message. Finally, new monitoring and resolution nodes can be easily added by customizing the framework's launch file and publishing to the predefined topics to communicate identified problems, severity and resolution progress.

Now that the presented framework performs well in simulation, a natural next step is to conduct field tests and demonstrate it on the physical robot platform. Further next steps include long-term data acquisition to learn from experience and more elaborate resolution techniques.

References

[1] T. Bohne, G. Parthasarathy, and B. Kisliuk, "A systematic approach to the development of long-term autonomous robotic systems for agriculture," in *43. GIL-Jahrestagung, Osnabrück, Germany*, vol. P-330, 2023, pp. 285–290.

[2] M. Hanheide, D. Hebesberger, and T. Krajník, "The When, Where, and How: An Adaptive Robotic Info-Terminal for Care Home Residents," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 341–349.

[3] N. Hawes, C. Burbridge, and F. J. et al., "The STRANDS Project: Long-Term Autonomy in Everyday Environments," *IEEE Robotics and Automation Magazine*, vol. 24, no. 3, pp. 146–156, 2017.

[4] F. D. Duchetto, A. Küçükyilmaz, L. Iocchi, and M. Hanheide, "Do Not Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation From Human Demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4084–4091, 2018.

[5] F. Han, S. E. Beleidy, H. Wang, C. Ye, and H. Zhang, "Learning of Holism-Landmark Graph Embedding for Place Recognition in Long-Term Autonomy," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3669–3676, 2018.

[6] J. M. Santos, T. Krajník, and T. Duckett, "Spatio-temporal exploration strategies for long-term autonomy of mobile robots," *Robotics and Autonomous Systems*, vol. 88, pp. 116–126, 2017.

[7] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[8] T. Bohne, "Execution monitoring for long-term autonomous plant observation with a mobile robot," *Osnabrück University*, 04 2022.

[9] C. A. Harris, N. Hawes, and R. Dearden, "Online plan modification in uncertain resource-constrained environments," *Robotics and Autonomous Systems*, vol. 140, p. 103726, 2021.

[10] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[11] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.

[12] E. Khalastchi and M. Kalech, "On Fault Detection and Diagnosis in Robotic Systems," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–24, 2018.

[13] P. T. Furgale, C. Pradalier, and T. D. Barfoot, "Editorial: Special Issue on Calibration for Field Robotics," *Journal of Field Robotics*, vol. 32, no. 5, pp. 629–631, 2015.

[14] J. Bohren and S. Cousins, "The SMACH high-level executive," *IEEE Robotics & Automation Magazine*, vol. 17, pp. 18–20, 2011.

[15] I. R. Nourbakhsh, C. Kunz, and T. Willeke, "The mobot museum robot installations: a five year experiment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 3636–3641.

[16] M. Steinberg, J. R. Stack, and T. Paluszkiewicz, "Long duration autonomy for maritime systems: challenges and opportunities," *Autonomous Robots*, vol. 40, no. 7, pp. 1119–1122, 2016.