Towards Context-Aware Navigation for Long-Term Autonomy in Agricultural Environments

Mark Höllmann*, Benjamin Kisliuk*, Jan Christoph Krause*, Christoph Tieben*, Alexander Mock†, Sebastian Pütz†, Felix Igelbrink†, Thomas Wiemann*†, Santiago Focke Martinez*, Stefan Stiene*, Joachim Hertzberg*†

*DFKI Niedersachsen Lab
Plan Based Robot Control Group
Osnabrück, Germany
firstname.lastname@dfki.de

†University Osnabrück
Knowledge-based Systems Group
Osnabrück, Germany
firstname.lastname@uni-osnabrueck.de

***Abstract*—Autonomous surveying systems for agricultural applications are becoming increasingly important. Currently, most systems are remote-controlled or relying on a single global map representation. Over the last years, several use-case-specific representations for path and action planning in different contexts have been proposed. However, solely relying on fixed representations and action schemes limits the flexibility of autonomous systems. Especially in agriculture, the surroundings in which autonomous systems are deployed, may change rapidly during vegetation periods, and the complexity of the environment may vary depending on farm size and season. In this paper, we propose a context-aware system implemented in ROS that allows to change the representation, planning strategy and execution logics based on a spatially grounded semantic context. Our vision is to build up an autonomous system called Autonomous Robotic Experimental Platform (AROX) that is able to generate crop maps over a whole vegetation period without any user interference. To this end, we built up the hardware infrastructure for storing and charging the robot as well as the needed software to realize context-awareness using available ROS packages.**

***Index Terms*—Autonomous systems, Context awareness, Navigation, Path Planning**

I. Introduction

To successfully deploy autonomous vehicles for long-term autonomy in dynamic environments like agriculture, it is necessary to take the current application context into account. In agriculture, unlike classic indoor scenarious, the shape of the environment may change rapidly. Hence, solely relying on established gridmap-based solutions is not possible. Depending on the time of year, the state of the crops, and the field's layout, more specialized solutions are needed. On the other hand, established methods for simpler use cases are well tested and understood, and might still be able to solve specific sub-problems. To fuse the usage of classic algorithms and novel, application-specific methods, we propose to take the geometric and semantic context into account. More specifically, we present an autonomous system called AROX for crop monitoring, that is deployed on a real farm. In addition to the mobile robot, the system consists of a storage and charging facility placed next to the monitored fields. The storage and charging station provides the shelter for the AROX robot (cf. Fig 1). In this scenario, the robot's objective is to periodically take scans of a maize field without user interaction. To achieve this, we built a semantically annotated geometric environment model that is divided into different navigation zones. Each zone exhibits different surface properties that require specific navigation algorithms or parametrization. For example, while driving and maneuvering within the container with the charg-

The DFKI Niedersachsen Lab (DFKI NI) is sponsored by the Ministry of Science and Culture of Lower Saxony and the VolkswagenStiftung. Work by Wiemann and Focke Martinez is supported by the Federal Ministry of Education and Research within the framework of the BonaRes programme (grant number: 031B0684D). Work by Höllmann and Tieben is supported by the Federal Ministry of Food and Agriculture within the experimental field Agro-Nordwest project (grant number: 28DE103E18)

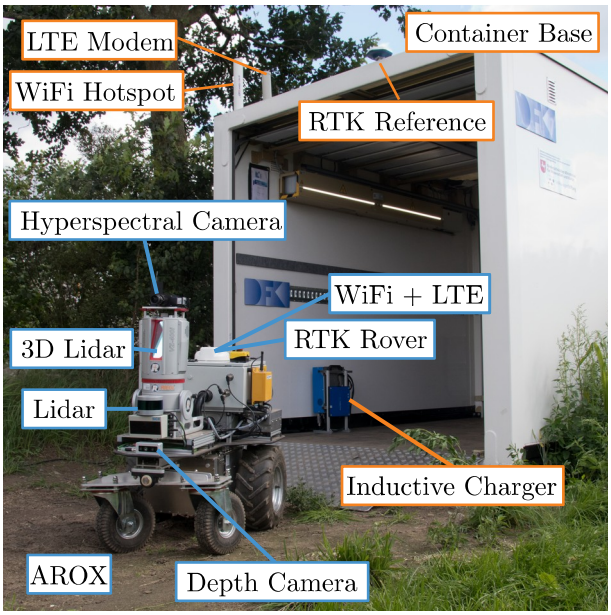

Fig. 1. Overview of the used hardware components on the AROX robot (blue) and storage container (orange) for autonomous long term crop monitoring.

ing station, the robot will use classic gridmap based navigation and action planning algorithms for docking. While driving on the field, it shall use polygonal 3D environment maps for path planning and execution, which include trafficability and roughness estimations derived from the recorded sensor data as described in [1]. The main idea of this work is to use the most appropriate specialized planning and navigation method based on the semantic context given in the geometric environment model. Although such an zone-based approach is not completely novel conceptually, for future development we plan to focus more on the integration of semantics for reasoning. In this paper, we contribute the low level implementation of the technical aspects of such a semantic system for robust navigation based on a semantic environment map. We present the system components of the used surveying system and the implementation of context-awareness in ROS, and give a preliminary report on the current state of the already realized components of the planned system as well as a proof of concept for the proposed context-aware navigation scheme.

The remainder of this paper is organized as follows: First, we present state-of-the-art algorithms and environment representations used in our context. Sec. III presents the hardware infrastructure implemented on the actual farm. Sec. IV presents the implementation of the context-aware planning and navigation system. Sec. V presents the current state of implementation by means of exemplary application scenarious. The final section discusses the achieved results and shows the planned extensions of the proposed system.

II. Related Work

Previous works like the one by Marder-Eppstein et al. [2] describe the software components needed for robust robot navigation. Many of them can be found in various implementations collected in the *ROS Navigation Stack*. The usual setup for such an autonomous system usually consists of an environment map, which is often some kind of occupancy gridmap [3], means for localization within such a map like *AMCL* [4] for indoor scenarios or the *robot_localization* package by *Charles River Analytics* in outdoor contexts to fuse *GNSS* position data with IMU and the odometry measurements. Path planning and execution are typically divided into global and local planning. Global planning typically involves the static information about distant parts of the environment encoded in a topological graph. Local navigation copes with the dynamic environment within the robots sensory horizon. For each of those components, various approaches and implementations already exist in ROS. The system presented in this paper is similar in that respect to these conventional approaches, but does not rely on fixed implementations of the single planning and execution components.

A. Flexible Navigation

An example for such a more or less static execution stack is *move_base*. However, more recently, Pütz et al. [5] presented a flexible navigation framework called *Move Base Flex* (MBF). It has been introduced specifically to address the highly dynamic and heterogeneous nature of navigation contexts, which may require highly different approaches to solve the aforementioned sub-problems. The main benefit of using *move_base_flex* is that it provides the possibility to easily switch between different navigation approaches. It uses the same software interfaces as `move_base`, but allows for online reconfiguration and replacement of planning and recovery movement strategies via a user implemented *SMACH* state machine [6]. In contrast to earlier approaches published by Conner et al. [7], it avoids multi-instantiation of core components and thus does not suffer from such computational overhead.

B. Semantic Context Mapping

Generally, maps for localization and navigation strictly rely on spatial information. Beyond that, semantic meaning was introduced early to enhance pure geometrical or topological representations. Semantic maps add semantic attributes to the geometric representation that allow to reason about the environment. First approaches proposed a layered architecture to fuse the different information domains [8], [9], resulting in various modern forms of introducing context information to maps by anchoring knowledge about objects, areas, and spaces to the geometrical robotic map representation. This context information is used to improve the performance in various robotic tasks such as object detection, object retrieval, task planning, navigation and more [8], [10]–[12].

However, these approaches regarding object localization are confined to *small-scale* spaces and are usually limited to small semantic domain models. In a more general and formal approach, Lang and Paulus define semantic maps as a form of hybrid map as defined by Buschka [13], combining geometrical representations of an environment with knowledge about the entities contained within the represented environment, their classes and attributes, and the relations between them in a way that allows inferencing [14].

Following this concept, we propose a way to handle heterogeneous navigation contexts derived from geometric layout and semantic labels. In our setup, we use that information to navigate on the farm and to switch between multiple, heterogeneous contexts requiring different localization methods, planning algorithms, local planners and recovery strategies by using a semantically annotated map and MBF.

III. (Hardware) Infrastructure

For long-term autonomy in agricultural environments, a base station is needed. In addition to the mobile robot platform, this station stores required supply equipment and external computation hardware. Furthermore, the overall hardware must be protected from unauthorized access and extreme weather conditions. In our setup, a $3.7\,\mathrm{m} \times 2.55\,\mathrm{m} \times 2.1\,\mathrm{m}$ container is used as base station. The so-called RYTLE HUB is mobile, so it is possible to transport all parts as a ready-to-operate system. Access through two electrical sliding gates is authorized with Bluetooth beacons. Via a mechanically fold-out ramp, the robot can reach the inductive charging station. Power supply is provided by a battery system or common line voltage. This

is required to operate the multi-band RTK GNSS reference station. For data exchange between base and robot, a high range WiFi network with 100 meter range is provided by the container. This is also used to transmit correction data from the RTK reference station to the rover. In addition to this WiFi network, both are equipped with a LTE modem to connect to the internet.

The AROX robot is a two-axes mobile platform based on an Innok Heros[1]. For pose estimation, sensor data from IMU, odometry and RTK GNSS are fused using the *robot_localization* ROS package. Besides these sensors for dead-reckoning, two laser scanners, located at the front and rear of the platform, are used for collision avoidance and localization. For crop monitoring and environment mapping, a high resolution 3D laser scanner with co-calibrated hyperspectral or RGB-camera is used. The entire equipment of the base station and the AROX robot is shown in Fig. 1.

IV. Context-aware navigation and planning

Our approach to deal with the rapidly changing and highly variable environment in agricultural environments is context-aware navigation and planning. Currently, we model the contextual data manually. The spatial dimension of each context is defined using geo-referenced polygons which can be associated with semantic labels. Additionally, we define specific waypoints, where the robot can switch from one context to another.

For example, one zone is the container. The corresponding semantic information is that the robot must navigate carefully and accurately due to the confined space. For this, the robot uses classic gridmap-based navigation and action planning. Another zone is the grass area in front of the container where the robot can navigate faster and less accurately. For this the robot uses a polygonal 3D environment map, that encodes the trafficability of the rough surfaces [1]. An example of such a waypoint is the gate that separates the container and the grass area as shown in Fig. 2.

A. Waypoint Server

The waypoint server takes the created zones and waypoints to build a topological graph. This graph includes all waypoints as vertices and inserts edges between all waypoints belonging to the same zone. Each waypoint is associated with both zones it connects. For each zone, the waypoint server stores the corresponding path planner, controller, recovery behaviors, and environment representation, and computes the locomotion costs for every edge in the graph. To query the waypoint server for the shortest path between pose A and pose B, both poses are first added as temporary vertices. Subsequently, edges are created to connect both of the new vertices to all other waypoints belonging to the same zones, and the locomotion costs for these edges are computed. If both poses belong to the same zone, a direct edge between them is also created. In this temporary extended graph, the shortest path from A to B is

[1]https://www.innok-robotics.de/en/products/heros

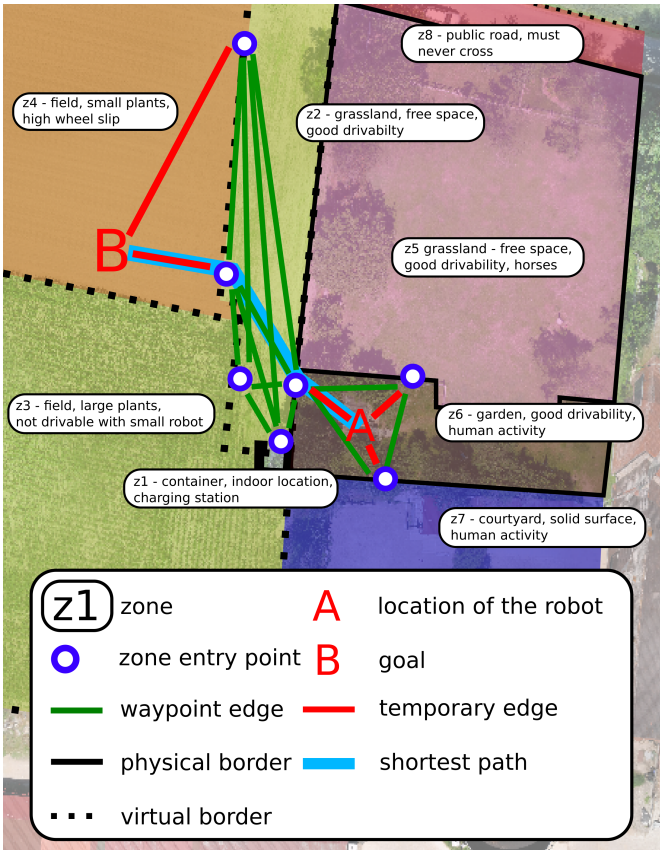

Fig. 2. A schematic map with waypoints and semantic associations.

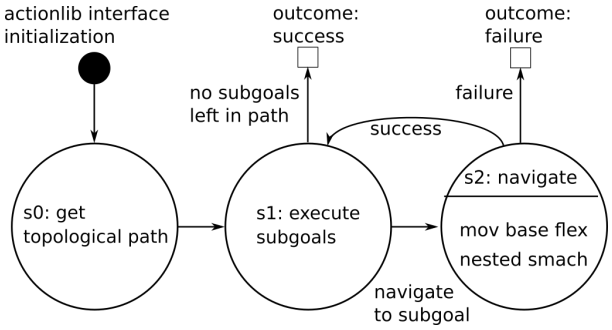

Fig. 3. The SMACH state machine for path decomposition.

then calculated using a shortest path algorithm. An exemplary result is displayed in Fig. 2. The result of the query are the edges between pose A to B including the contextual dependent information of the traversed zones .

B. Path Decomposition State Machine

To provide an abstract interface for this process, the Path Decomposition State Machine acts as an adapter and execution layer for the waypoint server and the navigation components. It is implemented as a *SMACH* hierarchical finite state machine within an *actionlib* wrapper to easily integrate into the ROS architecture of the robot as shown in Fig 3. Initially, it will be instantiated in *s0* by the action wrapper and given

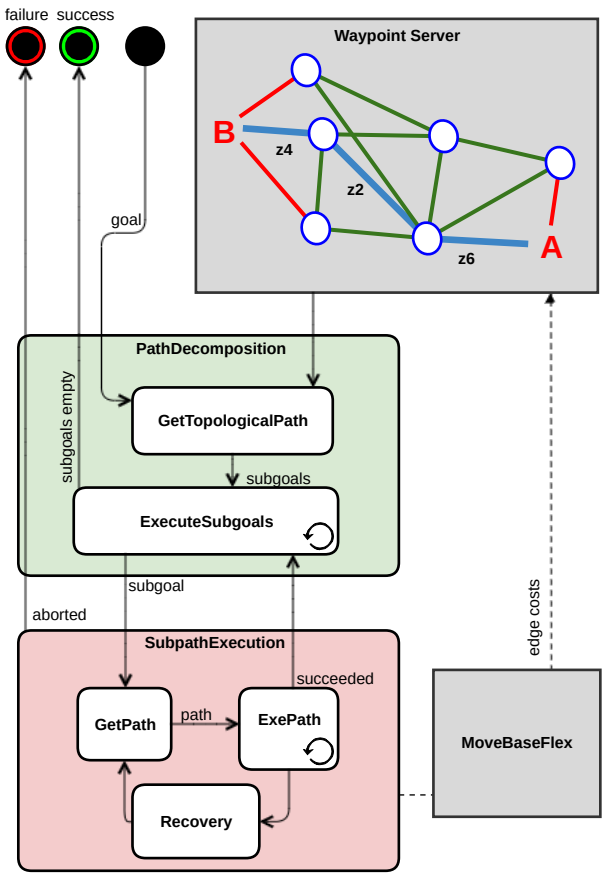

Fig. 4. Context-Aware Navigation state machine.

a goal pose which includes the geometric target pose and a robot coordinate frame known to the robot within its *tf-tree* coordinate frame. From there, it will request the current location of the robot, ask the waypoint server for the shortest path from this to the set goal, and transition to the execution state *s1*. The retrieved path consists of a list of sub-goals defined as tuples of geometric poses and parameters such as the local and global planner plugin to be used, and a list of recovery strategies. As long as the list of sub-goals is not empty, the first entry will be removed from the list and passed along the transition to the navigation state *s2*, which is a nested state machine interfacing directly to MBF. When this nested state machine terminates, the outcome will either be escalated to the action interface in case of failure, or it will transition back to the execution state *s1* where the next sub-goal will be passed or, in case the list of sub-goals is empty, terminate to a success outcome.

C. Sub-Path Execution

The actual movement of the robot to a sub-goal is executed by the sub-path execution state machine. For this purpose, the path-decomposition passes a sub-goal to the sub path execution, which is located in a zone together with the robot. This zone is associated with a path planner and controller names, as well as with a list of recovery behaviors as fallback strategies. In order to be able to flexibly execute and switch these sub-components of robot navigation, we use the MBF package *mbf_costmap_nav*, which provides a MBF navigation server based on the well known layered *costmap_2d* occupancy gridmap implementation. MBF is used as middle layer in between the higher level state machine and the low level navigation modules for planning, motion control, and recovering, which are implemented as plugins. It allows to load and run multiple plugins of the same type in a parallel fashion, e.g, with different configurations and different names. The *actionlib* interface of MBF allows to call path planners, controllers, and recovery behaviors with an associated name separately to support a high level of flexibility. According to the MBF interface, the sub-path execution consists of the three states *GetPath*, *ExePath* and *Recovery*. Using *GetPath*, MBF is called with the dynamically specified planner plugin name and the required start and goal pose parameters, as well as optional parameters such as the distance and angle tolerance to the goal pose. Based on this, MBF executes the corresponding path planning plugin and returns the computed path if the planning was successful, and a specified error code otherwise. The called path planner is using the underlying map representation to compute a path towards the given sub-target which has been defined as action goal for *GetPath*. This path is then used to call the controller within the *ExePath* state, in order to move the robot along this execution path. Using *ExePath*, MBF is called with a specified controller name and a list of poses defining the path to periodically execute the corresponding loaded controller / local planner plugin in order to move the robot towards the goal pose. The called navigation controllers manage the robot locomotion with different implemented strategies. This way, the controllers try to follow the given path while detecting dynamic obstacles. In case of error or failure, the MBF transitions to *aborted* and back-propagates the controller error code to the *SMACH* task level execution. Depending on the outcome, different strategies and recovery behaviors can be called in order to resolve the problem, e.g., by clearing the costmap, rotating to get an overview, waiting for an obstacle to pass, or moving backwards to escape from a possible collision or local dead end. However, when the robot reaches the sub-goal, the sub-path execution transitions to the terminal state *succeeded*. In cases of error, i.e., the controller failed, recovery behaviors which are domain-specifically associated with a certain zone, are executed by calling the MBF *Recovery* action with the corresponding recovery name. If the robot is able to free itself from a difficult situation with one of the recovery behaviors, the task level architecture resumes to the sub-path execution.

V. Applications

A. Demo Scenario

To show the capabilities of our planning system, we are currently building an outdoor field survey scenario. Within this test site, we plan to deploy the AROX robot autonomously in specified intervals to monitor the development of the crops

TABLE I
THE DIFFERENT ZONES, DOMAIN-SPECIFIC CONTROLLER AND RECOVERY CONFIGURATIONS, AND LOCALIZATION METHODS.

Zone	Type	Controller ID	Recovery ID	Localization
z1	**container**	docking	wait	AMCL
z2	**grassland**	dynamic_green	rotate	GNSS
z3	**field**	infield	wait	GNSS
z4	**field**	infield	wait	GNSS
z5	**grassland**	dynamic_green	rotate	GNSS
z6	**garden**	safety_ctrl	wait	GNSS
z7	**courtyard**	safety_ctrl	wait	GNSS
z8	**public road**	-	stop	GNSS

on the surrounding fields. For maize, we aim to acquire a series of 3D laser scans with the terrestrial laser scanner every week at specific poses to document the growth of the crops. After the data acquisition, the collected data shall be sent to a central server where it is processed automatically. Up to now, we installed the container with the power charging station, network connection, and GNSS RTK reference base. With this setup, we are currently able to safely drive to specified target locations using different controllers and representations to take 3D laser scans. An exemplary point cloud created from multiple 3D laser scans is shown in Fig. 5. A virtual fly-through of that scene video demonstrating the high quality the recorded colored 3D-point is available at our Youtube channel[2].

For the task at hand, we manually created a map of the working area using digital ground models (DGM) provided by the National Agency for Geoinformation and State Survey of Lower Saxony (LGLN). Within this geo-referenced map, we marked the respective zones in UTM coordinates to allow localization of AROX using the installed differential GNSS system. On top of the zone descriptions, we added a topological waypoint graph to model the transitions between the different areas as shown in Fig. 2. Each zone in this map is associated with a certain controller, a list of recovery behaviors and localization techniques. The different settings and zones configurations in the presented environment are shown in Tab. I.

The *Controller ID* column in Tab. I and Tab. II correspond to each other and specify the callable controller name used in *ExePath*. Further, Tab. II shows the used controller / local planner plugin with the specific configuration. For our setup, we use the *elastic-band-planner* (EBand) [15], the dynamic window approach (DWA) [16], and a custom local planner to dock the robot to the charging station.

B. Navigation Example

In the scenario shown in Fig. 2, the robot is located in the *garden* zone *z6* and has to navigate to a specific point inside the northern *field* zone *z4* before returning via *z2* to the *container z1* for charging. As listed in Tab. I, the robot starts in the *garden* zone (z6) with an *safety_ctrl* controller. It uses

[2]https://youtu.be/HyhkOWYah34

TABLE II
LIST OF CONTROLLER CONFIGURATIONS WITH PARAMETERS.

Controller ID	Controller Plugin	Basic Parameters
dynamic_green	EBand	vel_x: 0.1 - 3.3 m/s vel_rot: 0.05 - 0.8 rad/s xy_goal_tol: 1.0 m yaw_goal_tol: 0.8 rad
infield	EBand	vel_x: 0.1 - 2.2 m/s vel_rot: 0.05 - 0.4 rad/s xy_goal_tol: 0.15 m yaw_goal_tol: 0.25 rad
safety_ctrl	DWA	vel_x: 0.1 - 1.0 m/s vel_rot: 0.05 - 0.4 rad/s xy_goal_tol: 0.5 m yaw_goal_tol: 0.8 rad
docking	AI-trained custom	vel_x: 0.1 - 0.5 m/s vel_rot: 0.05 - 0.4 rad/s xy_goal_tol: 0.025 m yaw_goal_tol: 0.05 rad

the DWA local planner plugin with a configuration sketched in Tab. II to ensure an effective dynamic obstacles reaction.

When the robot passes the first waypoint into the *grassland* zone *z2*, the controller is switched to `dynamic_green`. To account for the available space and good driveability, the EBand controller plugin is configured with a higher maximum velocity and less strict pose tolerances. The last zone that the robot passes through in the direction of the goal position is the *field* zone *z4* using the *infield* controller. Currently this controller is based on a more restricted EBand parameterization as shown in Tab. II, with lower maximum speed and tolerances to the goal position, due to the expected high wheel slip in the loose soil.

To return to the container *z1* for battery charging, the robot has to traverse the *grassland* zone *z2* again as described before. The waypoint to enter the container is located just before the container's ramp.

By passing this point the system will have to switch from an outdoor multi-band RTK GNSS with fused odometry and IMU data as means of localization, to *AMCL* [4]. Currently, switching the corresponding ROS nodes have yet to be integrated to blend into the system as a whole. As further future steps we will integrate new path planners and controllers for the infield zone and additional maps for the outdoor scenario. Additionally, semantic annotation of the zones should be used not only to look up predefined configurations, but also to infer them using reasoning algorithms based on the well known *rete* algorithm [17]. Another step to accommodate for heterogeneous terrain navigation in addition to the 2D gridmap navigation, 3D polygonal representations will be provided for some areas, which can be used for 3D Mesh based navigation as described in previous work by Pütz et al. [1].

C. Towards long-term autonomy

With the current implementation, we realized a working prototype that is able to safely navigate within the modeled environment across different zones. To achieve self-sufficient long-term autonomy, besides robust navigation, an execution

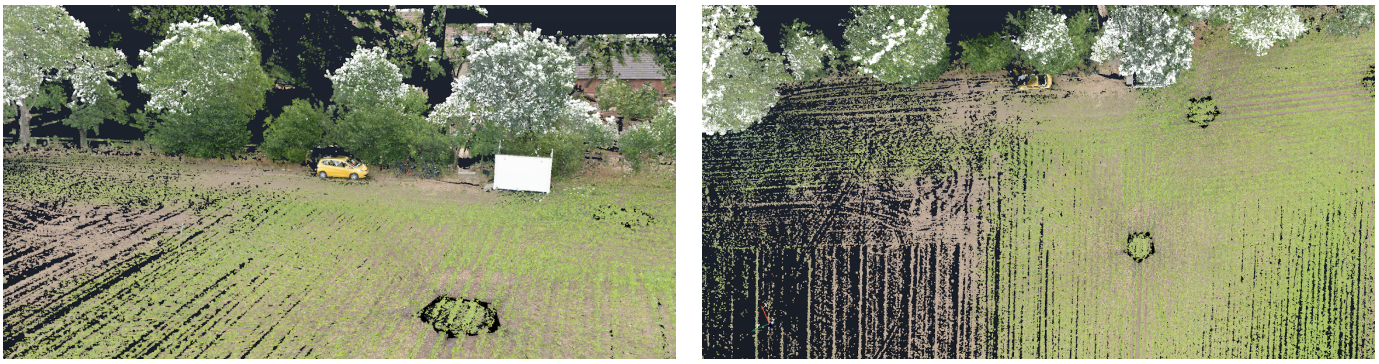

Fig. 5. Colored 3D point cloud of the data collected with the terrestrial laser scanner of the AROX robot. Color values are derived from the integrated RGB camery. Annotation with hyperspectral data is also possible.

monitoring system is required to detect unforeseen events and abnormal behavior of the system. For that, we plan to augment the current environment representation with a fully symbolic semantic model.

Having such a representation allows to use rule-based reasoning to detect process states. In previous work [18], we combined spatial representations like the one used in this scenario, to monitor machine states and generate process events in a maize harvesting campaign. For that, we used our SEMAP framework [11] that allows to deduct qualitative spatial relations with explicitly modeled semantic background knowledge about the involved machines, used facilities and process events.

The transfer of such a model into the currently developed system is straight-forward and would allow to monitor the current state of the autonomous surveying systems. In the given context, such a semantic monitoring system allows to automatically detect high level process states like ”charging”, ”driving” or ”scanning” based on the sensor data and semantic environment model. On the other hand, abnormal or illegal states could also be detected by defining simple rules like ”the AROX robot should never be outside the defined zones”, or ”the robot should never start a measurement campaign without having charged in the container before”. With an according spatio-semantic model, these and similar rules could be defined easily to improve the monitoring during the desired long-term deployment of the autonomous system.

VI. Discussion

In this paper we presented the first work towards a fully autonomous robotic system in an agricultural surveying scenario. The system is completely implemented in ROS allowing the use of proven standard methods as well as more application-specific algorithms and representations. The current state presented here serves as an extensible foundation to realize safe navigation in dynamic environments. In future work it will be combined with an actual semantic mapping framework to enable more complex planning and execution monitoring, and to increase the capabilities of the autonomous system.

References

[1] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, “3d navigation mesh generation for path planning in uneven terrain,” in *9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2016)*. IFAC, 2016.

[2] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, “The office marathon: Robust navigation in an indoor office environment,” in *ICRA 2010*. IEEE, 2010, pp. 300–307.

[3] H. Moravec and A. Elfes, “High resolution maps from wide angle sonar,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 116–121.

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte carlo localization: Efficient position estimation for mobile robots,” 01 1999, pp. 343–349.

[5] S. Pütz, J. Simón, and J. Hertzberg, “Move base flex: A highly flexible navigation framework for mobile robots,” in *IROS 2018*, 2018.

[6] J. Bohren and S. Cousins, “The smach high-level executive [ros news],” *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[7] D. C. Conner and J. Willis, “Flexible navigation: Finite state machine-based integrated navigation and control for ros enabled robots,” in *SoutheastCon 2017*, March 2017, pp. 1–8.

[8] D. V. Lu, D. Hershberger, and W. D. Smart, “Layered costmaps for context-sensitive navigation,” in *IROS 2014*. IEEE, 2014, pp. 709–715.

[9] B. Kuipers, “Spatial semantic hierarchy,” *Artificial Intelligence*, vol. 119, no. 1, pp. 191–233, 2000.

[10] M. Günther, J.-R. Ruiz-Sarmiento, C. Galindo, J. Gonzalez-Jimenez, and J. Hertzberg, “Context-aware 3d object anchoring for mobile robots,” *Robotics and Autonomous Systems (RAS)*, vol. 110, pp. 12–32, 12 2018.

[11] H. Deeken, T. Wiemann, and J. Hertzberg, “Grounding semantic maps in spatial databases,” *Robotics and Autonomous Systems*, vol. 105, pp. 146–165, 2018.

[12] A. Saffiotti, S. Coradeschi, P. Busch, and J. Gonza, “Multi-Hierarchical Semantic for Mobile Robotics,” in *IROS 2015*, 2005.

[13] P. Buschka, “An investigation of hybrid maps for mobile robots,” Ph.D. dissertation, Örebro universitetsbibliotek, 2005.

[14] D. Lang and D. Paulus, “Semantic Maps for Robotics,” *Robotics,Proc. of ICRA, ”Workshop on AI Robotics”*, 2014.

[15] S. Quinlan and O. Khatib, “Elastic bands: connecting path planning and control,” in *Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807 vol.2.

[16] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[17] C. L. Forgy, “Rete: A fast algorithm for the many pattern/many object pattern match problem,” in *Readings in Artificial Intelligence and Databases*. Elsevier, 1989, pp. 547–559.

[18] H. Deeken, T. Wiemann, and J. Hertzberg, “A spatio-semantic approach to reasoning about agricultural processes,” *Applied Intelligence*, vol. 2019, 2019.