Navigation System of the Autonomous Agricultural Robot "BoniRob"*

Peter Biber, Ulrich Weiss, Michael Dorna, Amos Albert

***Abstract*—In the context of the public funded project BoniRob we have developed an autonomous agricultural robot that can autonomously perform repeating phenotyping tasks for individual plants on different days.**

These tasks require a robust and reliable navigation and perception system. In this paper, we summarize our navigation approach and give details about the employed sensors, algorithms and the system integration.

The system has successfully been demonstrated at various public occasions and we are now, with our partners, investigating opportunities for a limited-lot production for research and evaluation purposes. An overview of the complete project is given in [1]. Furthermore, we are currently developing the next version of the robot that will allow in addition mechanical weed control.

I. Introduction

Autonomous agricultural robots are an alternative to the tractors found on fields today. Cultivation tasks like seeding, spraying, fertilizing and harvesting may be performed by fleets of autonomous agricultural robots in the future.

Independent of the actual design a serious agricultural robot will be a complex and expensive vehicle – the challenge is therefore to prove that it is competitive to traditional technology and may even bring a decisive lead. One critical factor here is the optimal utilization of the robot over the day and over the year. To reach a full utilization we need to support multiple applications as the tractor does with different tractor/implement combinations. Following that scheme the agricultural robot needs to be a vehicle with some basic capabilities and the possibility to support multiple applications. Among the basic capabilities we surely require a navigation system for safe and autonomous navigation.

*The authors are with:
Corporate Sector Research and Advance Engineering,
Robert Bosch GmbH, Postfach 300240, 70442 Stuttgart, Germany
Email:
(peter.biber|ulrich.weiss|michael.dorna|amos.albert)@de.bosch.com

The project *BoniRob* was and *RemoteFarming.1* is supported by funds of the Federal Ministry of Food, Agriculture and Consumer Protection (BMELV) based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support programme. Without our project partners the presented results would not have been possible. Many thanks to our colleagues at AMAZONE and HS Osnabrück.

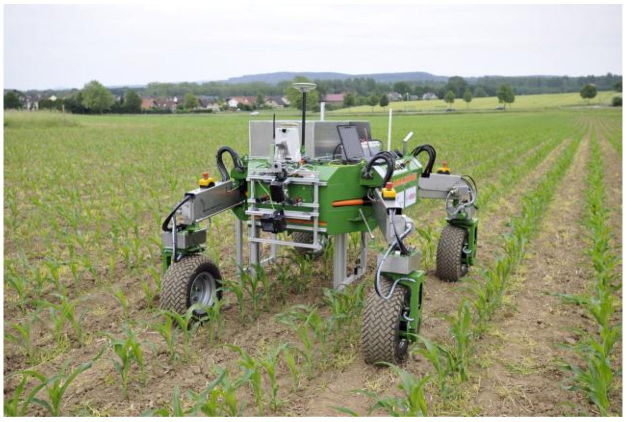


Figure 1: BoniRob on a maize field.

In the context of the public funded project "BoniRob" such a vehicle (the "BoniRob", Fig. 1) has been developed by our project partners and us. The vehicle comes with a navigation system that allows the autonomous navigation over row based cultures. It can be configured to work for different cultures, e.g. maize or wheat, different track widths, number of rows, structure of the field or sensors used. In addition applications are provided by the navigation system with the current robot position and the position of individual plants or the plant rows.

As a first interesting application for such a robot we have identified the "crop scout" that performs repeating phenotyping tasks for plant breeders. In a follow-up project we are developing a manual weeding module. Further modules are under investigation but to build a successful ecosystem such applications will have to be developed by third parties in the future.

The first public presentation of BoniRob was at Agritechnica 2009. At DLG Feldtage 2010 BoniRob was navigating continuously on a small maize field for three days. We are now, with our partners, investigating opportunities for a limited-lot production for research and evaluation purposes.

The robot platform itself was developed by Amazonen-Werke H. Dreyer GmbH & Co. KG (AMAZONE) [2]. The navigation system was developed at Bosch corporate research by the authors.

In this paper we describe the navigation system. To set the background we also give a brief overview of the other components of the BoniRob system. Parts of the navigation system have already been presented in previous publications. These are referenced in the respective places.

Here we give for the first time a complete description where we emphasize the requirements from potential user, the resulting system architecture and the interaction between the components. We also share some of the experiences we made in developing and testing the system.

II. BoniRob Overview

First we give a brief overview of the complete BoniRob system.

A. Vehicle and system architecture

The BoniRob vehicle is a flexible platform with 4 wheels that can be steered separately. The wheels are driven by wheel hub motors. By moving the arms BoniRob can adapt its track width from 0.75m to 2m and its chassis clearance from 0.4m to 0.8m. Overall the vehicle comprises 16 degrees of freedom, implemented by a mix of electro and hydraulic actuation. The BoniRob system is equipped with a number of dedicated control units: 4 Motor controllers, motor and hydraulic interface, navigation control unit and application control unit. The different control units are connected by Ethernet and communicate using TCP/IP (except for the motor controllers that are connected by a CAN bus).

B. Application

BoniRob follows an “App”-concept similar to the mobile phone world. We are providing the basic hardware and software for driving on fields and give third parties the possibility to integrate their own application specific modules. For that, the BoniRob corpus contains a shaft for inserting applications containing sensors or actuators.

Within the project a plant phenotyping application for the evaluation of field trials for plant breeders has been implemented by the University of Applied Sciences Osnabrück [3]. Plant breeders are interested in the evaluation of breeding processes and crop status. For that quantifiable measures are of highest importance. Today, such measures are taken manually by human experts based on random samples in the different test fields. With BoniRob, these measures can be taken automatically for each individual plant in the complete field trial. The measures of interest include for example number of plants, plant height, stem thickness and biomass as well as information about water supply and indications for possible plant diseases.
There is also the possibility to produce output for the Open Source GIS “OpenJump” with the PIROL plugin [4]. The phenotyping application has its own set of dedicated sensors connected to the phenotyping control unit.

III. Navigation system hardware

Fig. 2 shows the hardware of the navigation system. We employ the following sensors for the navigation: A 3D MEMS lidar (FX6 from Nippon Signal), an inertial sensor unit (XSens) and optionally a RTK-GPS. Furthermore we have a remote control (Logitech Rumblepad) and a WLAN access point with a special outdoor antenna. All navigation sensors are connected to a fan-less embedded PC (Intel Atom) designed to operate in the target environment conditions.

While most of the sensors are fairly standard it is worth to describe the 3D MEMS lidar in more detail. The FX6 measures the distance and reflectance intensity using an infrared pulsed laser light (Class 1) with up to 16 fps. It measures the time-of-flight at a smallest unit of about 30 picoseconds, which leads to a ranging precision of at most 1 cm. The laser beam is reflected by a mirror oscillating independently into two directions and thereby creating a full scan in a single iteration (Lissajous figure). The resolution of a full scan is 59x29 pixels covering a field of view of 50°x60°. The specified precision is 80mm (3σ). The FX6 has a low power consumption (6W) and a moderate weight (1kg).

Its main advantage is that it is not influenced by sunlight. Using the lidar sensor it is even possible to operate 24 hours a day and handle conditions like light fog and dust. For further details we refer to [5].

The outdoor antenna guarantees optimal connectivity to a remote user. Using WLAN, the remote user can activate a visualization of the robot operation or – via remote desktop – change the configuration of the robot.

IV. Navigation system layer model

The navigation software is organized in four layers (Fig. 3).

A. Layer 1: Drivers

The lowest layer 1 contains drivers and pre-processing for sensors and the interface to the BoniRob drive control. This includes preprocessing/postprocessing steps that can be performed within the driver and where no additional data from other modules are needed. Examples for that are calibration of the yaw rate sensor or conversion of motion commands from lateral and rotational speed to wheel angles and speeds for the individual motors.

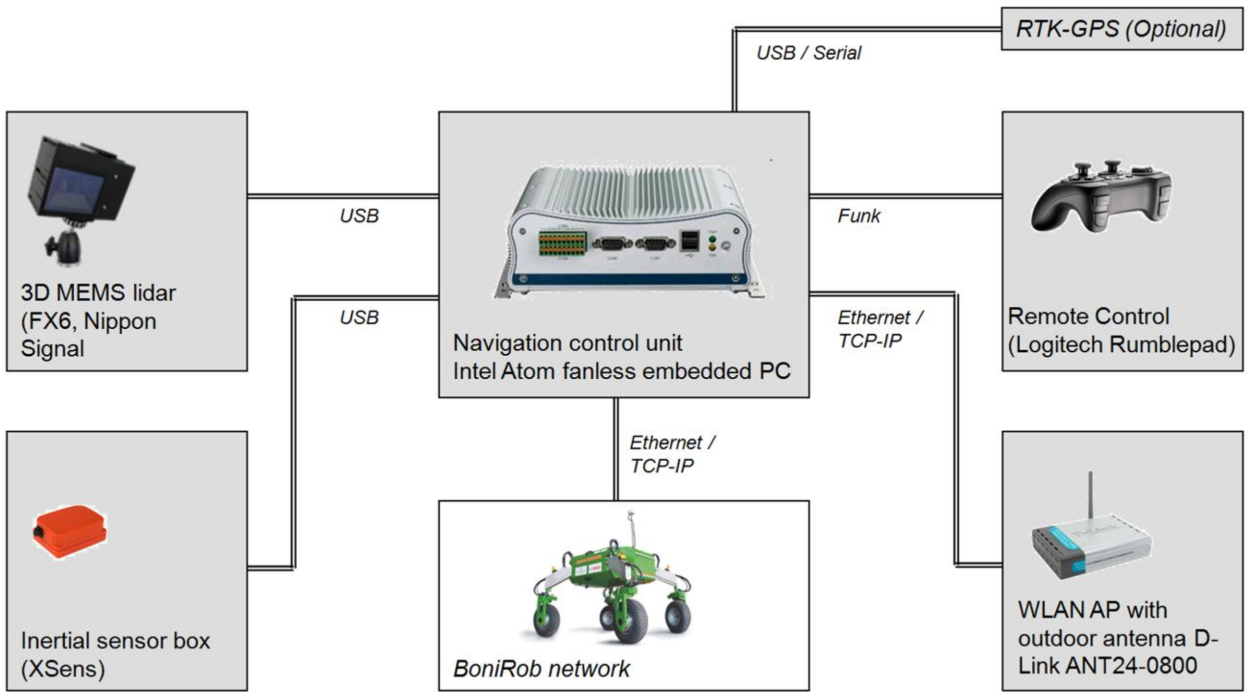

RTK-GPS (Optional)
USB / Serial
3D MEMS lidar (FX6, Nippon Signal
USB
USB
Navigation control unit Intel Atom fanless embedded PC
Funk
Ethernet / TCP-IP
Remote Control (Logitech Rumblepad)
Ethernet / TCP-IP
Inertial sensor box (XSens)
BoniRob network
WLAN AP with outdoor antenna D-Link ANT24-0800

Figure 2: Navigation system hardware

B. *Layer 2: Reactive layer*

Layer 2 is the reactive layer. This layer contains on the one hand modules that directly process sensor data without knowledge of high-level states. On the other hand it contains primitive driving skills receiving their input data from the sensor data processed on the reactive layer. On the sensor processing side on layer 2 subsequent modules have been implemented: Ground detection, Row detection, Basic Kalman Filter localization, global localization, and mapping. The most important driving skills are row following control and turning.

1) *Ground Detection*

The ground is detected from the 3D point cloud recorded by the FX6. For that we use the RANSAC algorithm to fit a Hessian plane equation to the data. The detected plane is then refined by a Least Squares fit.

Number of inliers and residuals of the Least Squares are then considered to detect failed ground detection. Also, a Kalman filter tracks the state of the plane over time. If the ground detection failed the state is just propagated.

Depending on the field the detected ground can correspond to soil or the canopy. Also the thresholds for plane detection have to be set according to the application.

As a side effect of ground detection we can then derive the height and the tilt angle of the scanner so we have to define only the x/y position of the scanner manually in the configuration.

2) *Row detection*

After ground detection we transform the point cloud into the ground plane coordinate frame. In that frame the z-coordinate directly indicates the height over ground of a 3d point. We then segment the data into three clusters: ground, row, and other. The row cluster is filled with those 3d points that have a z-coordinate within a specified interval. The interval again is application specific. For example for maize 3d points with z-coordinate (i.e. height over ground) greater than 10cm are selected.

Then, again, RANSAC and a least-square fit determine the state of a row model. The row model currently may consist of one or two rows. In the BoniRob standard scenario we drive over two rows of maize hence the two rows model is used then. Here, the detected rows correspond to the maize plants. In contrast for winter wheat the row cluster is below the ground and the detected rows correspond to the tracks.

3) *Localization/Mapping*

The localization module requires at least data from odometry and inertial sensors as input. We further support RTK-GPS data. A standard Kalman filter then combines the input to a pose estimate.

If the system is equipped with a RTK-GPS system we can provide high precision position data (~ 2cm accuracy). High precision is needed only if we want to map a field. As input for the control the pose estimate without RTK-GPS is sufficient. Mapping support is only for visualization purposes. We can map individual plants or plant rows.

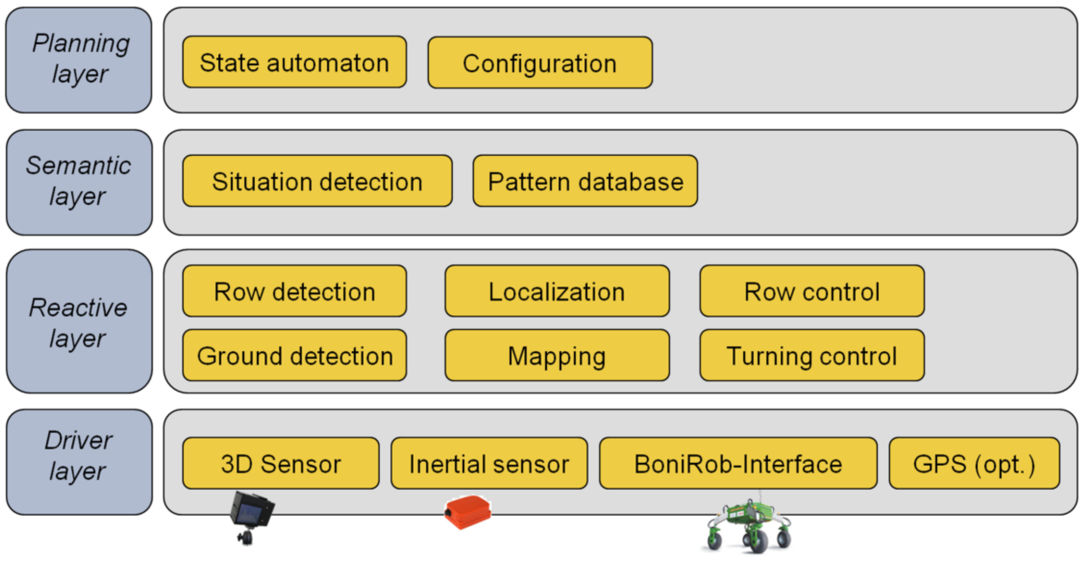


Figure 3: Navigation system software architecture. Modules are organized in four layers.

4) *Row control*

Row control takes data from row detection and localization as input. An optimal control is used to keep the robot centered above the row. Lateral error and angular error with respect to the row form the input to the optimal control. The outputs are lateral and angular speed corrections. If the requested angular speed exceeds a threshold the robot is stopped for safety reasons and the correction is applied while the robot is not moving forward. This minimizes the mechanical stress to the robot arms and their connection to the robot body.

5) *Turning*

Turning is triggered by upper layers. The following triggers can be configured:

- row detection fails to find a row for a certain distance (for flat headland, no preparation required)
- GPS borders (teach-in required)
- End markers (preparation of field required)

Turning control is based on localization data. Part of the turning maneuver is to drive along the field side until the next row is detected. The exact behavior can be configured in the state machine.

C. *Layer 3: Semantic layer*

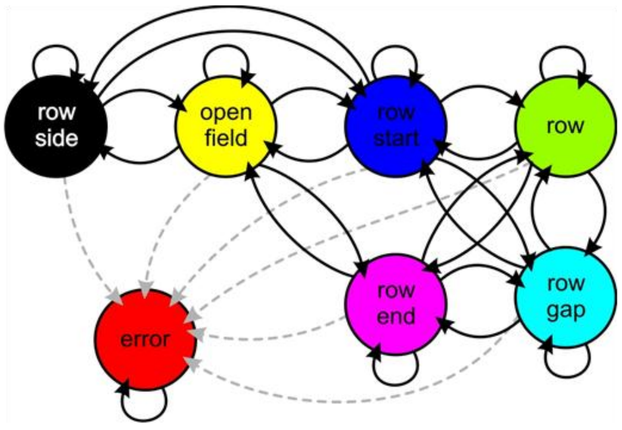
Semantic mapping and localization in layer 3 extends the metric mapping and localization of the lower layer 2. In the semantic layer we are interested in classifying the type of environment. Semantic mapping in general is currently an active research topic in the mobile robotics community, especially in an indoor environment where coarse features like walls, floors or doors can be recognized and used for map interpretation or place recognition ([6],[7]). It is however not possible to completely predict all types of features or objects that can be contained in such an environment.

In agricultural robotics we are actually facing a less complex environment than in indoor service robotics. Here we can have the realistic goal to arrive at a set of features that completely describes all allowed working conditions. If we cannot classify a concrete situation using this set of features we can then deduce that the robot’s environment is not a specified working condition.

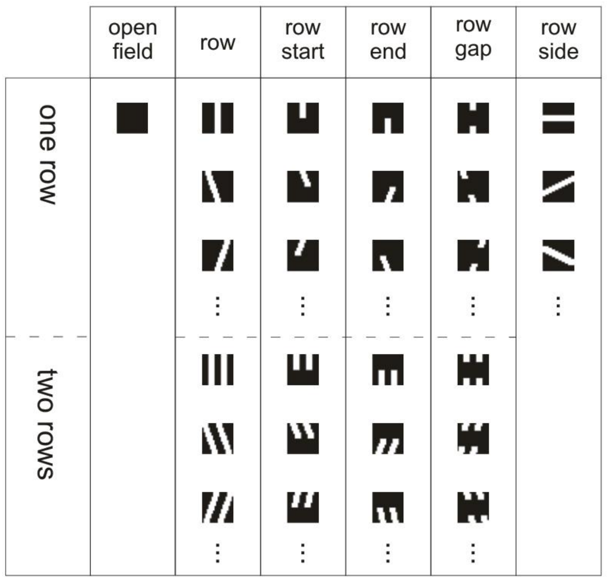
On a field we are expecting the following limited number of classes: crop row, open field, side of field, row gap, beginning and end of row. Everything else, e.g. an unexpected obstacle, is mapped to an error state. The states and the possible transitions are encoded using a probabilistic state automaton (see Fig. 4(a)). It consists only of transitions that are possible according to the defined field layout and reasonable for navigation. For example the robot should not drive from *row side* into *row*. Such circumstances and other not predicted situations are handled by the error state. If this state is reached the robot should stop immediately until the situation is cleared, possibly with human help.

The probabilistic state automaton maintains a probability distribution over states that is updated during operation using a particle filter. The needed state and transition probabilities were determined by a using a large number of example data recorded in simulation.

For the perceptual model needed for the particle filter we have developed a place classification algorithm using the 3D sensor and results from the ground detection in layer 2.The point cloud is first transformed into the ground plane coordinate frame. Then we place a square grid with 20x20 cells and a side length of 20cm on the plane. For each grid cell the maximum height of the points above the ground plane is calculated and a binary image using a fixed height



(a)



(b)

Figure 4: (a) Probabilistic state automaton.(b) Pattern database.

threshold is calculated. The binary image is then compared to a pattern database (Figure 4(b)). Each state is represented by a number of patterns in this database (except for the error state). The database is specific to a certain application. For example in the figure we see the expected binary images for maize fields in two variations: driving over one row only or over two rows. This approach allows us to adapt to other environments in a flexible way.

The existence of the semantic layer also allows simple implementations in the reactive layer: The semantic layer implements an independent monitoring of the high level state of the robot, e.g. if it is over a row or on the side of the field. So it can decide if there is a row at all in the current sensor image. For example, the row detection just has to find the best candidate for a row and does not need to implement complex verification steps. More details about the semantic localization can be found in [8].

D. *Layer 4: Planning layer*

The planning layer contains the application state machine and the application configuration. We provide a generic state machine creation and execution mechanism the can be configured by a xml file. The xml file contains the name of the states and the events that trigger the transitions between states. It is even possible to reread the configuration file during its execution. The state machine then coordinates the data flow between modules and activates/deactivates primitive driving skills in the reactive layer.

A navigation class is associated to each possible state in the state machine, e.g. *FollowRowNavigation*. Objects are created from a navigation class using the Factory pattern. This pattern allows flexible changes of navigation classes across possible applications and field types while leaving the state machine untouched. A navigation state itself starts and stops the primitives provided by layer 2. E.g., *FollowRowNavigation* uses the *FollowLaneCommand* and provides the primitive with parameters, in this case the lane vector that it gets from the row detection module in layer 2. To summarize here: we provide an architecture where we can flexibly manage the overall software in an application-dependent way: state machine, navigation states, data flow from sensors to actuators and primitives.

In the xml configuration we can additionally define system data like COM ports or IP addresses of SW components.

V. System Integration and Testing

A. *Simulation*

The robot, its sensors and the complete environment have been modelled in the simulation environment Gazebo [9]. This includes all degrees of freedom also for changing track width and height. The simulation provides 3D rendering and a physical simulation of robot and environment. New sensors like the FX6 were easy to add based on code for existing sensors. Fig. 5 shows a screenshot of the simulation.

The complete navigation system was developed using the simulation at a time where the robot was physically not yet available. It has been integrated and tested in this environment. We chose Gazebo as simulation based on a value benefit analysis and it turned out to be a good decision. When it came to system integration on the real robot the behavior was as simulated and no adaptions were needed. We conclude therefore that a simulation based on Gazebo provides a sufficient level of reality to develop and test agricultural robots like BoniRob.

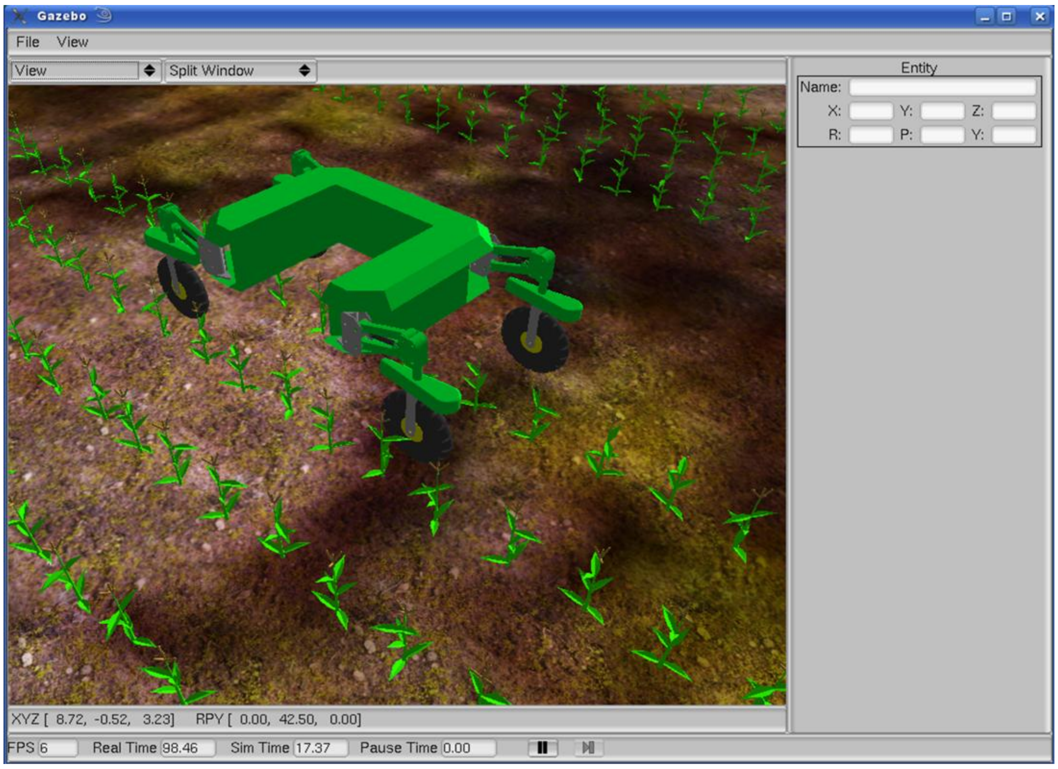

Gazebo
File View
View
Split Window
Entity
Name:
X: Y: Z:
R: P: Y:
XYZ [8.72, -0.52, 3.23] RPY [0.00, 42.50, 0.00]
FPS 6 Real Time 98.46 Sim Time 17.37 Pause Time 0.00

Figure 5: Screenshot of BoniRob simulation in Gazebo

B. Middleware

The navigation software uses a component based architecture which includes a middleware that is specially adapted to the requirements in robotics. This incorporates a component class and ports, specification of communication data, communication pattern (publish-subscribe, query-response), naming server and data serialization.

We employ the ACE library [10] as operation system abstraction. It provides a POSIX interface, a middleware and serialization framework. SmartSoft [11] is the component framework and contains the communication patterns we used. Microsoft Windows is used as underlying operating system. Several other open source libraries are used as part of the navigation software e.g. for mathematical operations.

C. Testing

Actual testing time in a multi partner project with distributed location is limited. Therefore it is of highest importance to use this time optimally and record as much data as possible. For that we have implemented an automatic logging system that saves all messages that are passed around in the system. The communication classes require the implementation of methods for saving and loading messages. This way complete logging is enforced by the software architecture.

We also support a replay mechanism. Here we can give the system the location of log files and it replays the data in the original order. For that each log file entry automatically contains a time stamp with both CPU time and GPS time. Matlab based visualizations are also available for the most important log files. These can be developed in advance and be used at testing site. This proved to be especially useful for testing of data integrity during system integration.

VI. FUTURE WORK

We will continue to develop the navigation system within a new project called “RemoteFarming.1” where the goal is to develop a weeding application for carrots.

There are four main targets for the next version of the navigation system:

- Providing support for more environments. Besides the carrots we are currently preparing grass and wheat which are of special importance to plant breeders.
- Separation of real time critical and uncritical parts of the navigation system. For the real time critical part a separate control unit using a real time OS will be employed.
- Change of communication framework from SmartSoft to ROS [12]. ROS is about to become the important standard in mobile robotics and is employed at Bosch also in other projects.
- Making the system more robust and reliable by means of system monitoring and diagnosis in order to allow thorough evaluation of the system by third parties.

The navigation system will also support the next version of the BoniRob currently developed by AMAZONE.

VII. References

[1] A. Ruckelshausen et al. Bonirob: an autonomous field robot platform for individual plant phenotyping. In Joint International Agricultural Conference, 2009.

[2] http://www.amazone.de

[3] Wunder, E., Kielhorn, A., Klose, R., Thiel, M., Ruckelshausen, A.: GIS- and sensor-based technologies for individual plant agriculture, Landtechnik 67 (2012), no. 1, pp. 37-41.

[4] http://www.pirol.hs-osnabrueck.de/pirol-openjump.html. (German only, 26.06.2012)

[5] Weiss, U., and Biber, P. Plant detection and mapping for agricultural robots using a 3D-LIDAR sensor. Robotics and Autonomous Systems 59, 265-273, 2011. Available at: http://dx.doi.org/10.1016/j.robot.2011.02.011.

[6] A. Nüchter and J. Herztberg. Towards semantic maps for mobile robots. Robotics and Autonomous Systems, 56(11), 915-926, 2008.

[7] O. Mozos. Semantic Place Labeling with mobile Robots. PhD thesis, Department of Computer Science, University of Freiburg, July 2008

[8] U. Weiss and P. Biber. Semantic place classification and mapping for autonomous agricultural robots. In IROS 2010 workshop: Semantic Mapping and Autonomous Knowledge Acquisition, 2010.

[9] http://gazebosim.org/ (16.07.2012)

[10] ACE: The ADAPTIVE communication environment. http://www.cs.wustl.edu/~schmidt/ACE.html (16.07.2012)

[11] SmartSoft. http://smart-robotics.sourceforge.net/ (16.07.2012)

[12] ROS (Robot Operating System): http://www.ros.org/wiki/ (16.07.2012)