Autonomous Navigation Tool for Real & Virtual Field Robots

*Ali ZOGHEIB**
Mid Sweden University, Sundsvall, Sweden

Abstract
Due to ecological and economical demands in agriculture as well as technological improvements, the development of autonomous field robots has gained importance within the last years. A common task for all field robots, whatever their main function is, is to Autonomously Navigate between row cultures. As a first step, toward a full featured Simulator tool dedicated to agriculture applications, an autonomous navigation tool based on sensor fusion and fuzzy logic, has been realised. The data of various sensors are transferred to control variables thereby opening the options for including further sensors without changing the model. Different aspects ranging from missing plants to path planning (such as skipping rows) and field mapping are included. A wireless connection with the physical robot is implemented, currently it is used for robot navigation control, data sending and receiving. The first experiments for the navigation between rows showed that simulation and real world were in phase.

Keywords
Field Robots, Simulator, Autonomous Navigation, Sensors Fusion, Agriculture Applications, Fuzzy Logic, Navigation Control.

1 INTRODUCTION

The development of autonomous field robots has gained importance within the last years, due to ecological and economical demands in agriculture, as well as to technological improvements. Many experimental field robots were developed with a wide range of applications: Seeding (Trebinski G. et al., 2004), Weed Control (Van Evert. F.K. et al., 2006), Fruit Picking (Katupitiya J. et al., 2005), Data Collecting (Nagasaka Y., 2004), Harvesting (Edan Y., 2000), ...). In all these applications, the field robots are required to successfully achieve a common task: *Autonomously Navigating between rows culture*, making it one of the most important tasks, in this field, as can be observed in the place this task has in specialised field robots contests (i.e. *Field Robot Event Contest*). In this paper, we will present an *Autonomous Navigation Tool*, dedicated for *real* and *virtual* field robots' in-between rows navigation. In this tool, data from different sensors types (i.e. distance sensors, mounted camera, remote sensing images, ...) can be fusioned into one coherent environment's model. From this model a set of five control variables are derived, reducing the complexity of the navigation control to a constant one, whatever the number of fusioned sensors is. A *Fuzzy Logic Controller* (Zadeh L., 2002), taking the control variables as its inputs, is designed to derive the move commands (i.e. *velocity* and *steering angle*). A graphical user interface was developed allowing the design and adjustment of navigation strategies, by simple interactions with the tool. With this interface, the navigation designer is not required to know the underlying theory of the implemented fusion method nor of the navigation control's one, allowing him/her to concentrate on the specialized tasks his/her field robot is required to do.

In the next section, the tool's overall architecture will be presented, as well as its major components' descriptions. The tool's features are detailed in section 3. Future works, and Conclusions are presented in section 4 and 5 respectively.

2 AUTONOMOUS NAVIGATION TOOL ARCITECTURE

Four components constitute the core of the *Autonomous Navigation Tool*: the *Environment Modelling*, the *Control Variables Computation*, the *Navigation Control* and the *Navigation Control Design Interface*. A typical use scenario is two folds: a *Design Phase* and a *Simulation/Real world Run Phase*. The *Design Phase* starts by specifying the robot's 2D dimensions and the position and the orientation of each sensor the robot's has, in a *Configuration File*, be it a virtual or a real robot. Designing the navigation strategy, using the dedicated interactive GUI (Graphical User Interface), should follow. At the start of the *Simulation/Real world Run Phase*, the tool reads the robot's specifications from the *Robot's Configuration File*. Once the tool configured for a particular robot, it starts to request the robot for its sensed data. In response to this invocation, the robot sends these data. The tool uses them, with the robot's sensors configuration, to derive a model of the robot's current surrounding environment. From whom the control variables are computed and then fed to the *Navigation Control* that decides what move decisions (i.e. *move velocity* and *steering angle*) are adequate for the current situation. The tool then sends these commands to the robot. The communication between the tool and the robot goes through a *Generic Communication* component that exchanges the data and commands with the real robot through a wireless connection, in case of real world experiment, and directly with the virtual robot, in case of a simulation. In this later case the screen, displaying the field/environment and the robot's current position, is updated.

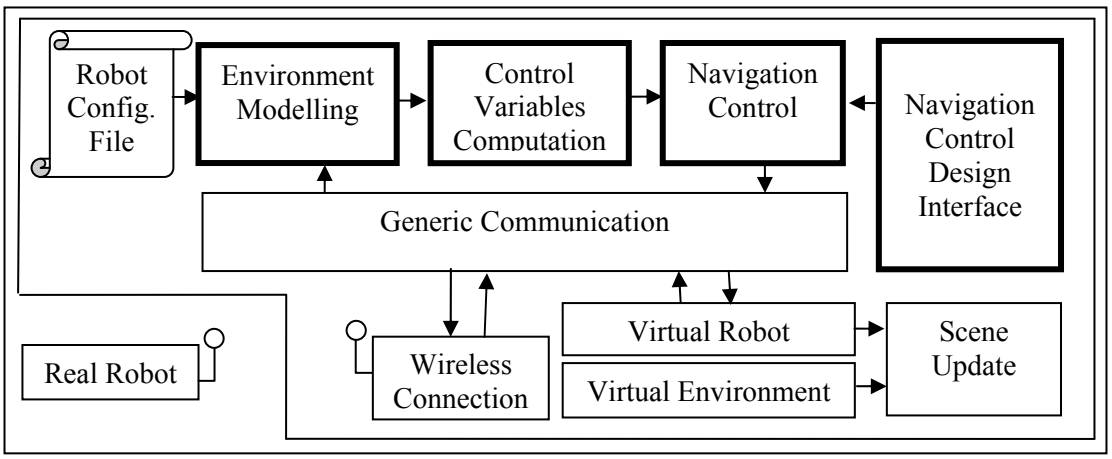

Figure 1: Field Robot's Autonomous Navigation Tool's Architecture

The environment for the virtual robot can be defined as *Sinusoidal*, *Triangular*, *Straight* or *Hand Drawn* discrete lines. In this paper we will focus on the tool's core components, those of interest to the navigation task. A detailed description of the remaining components can be found in (Zogheib, A., 2008).

2.1 Environment's Modelling

Field robots' inter-rows autonomous navigation exploits the field's natural environment, by the mean of sensors. The most exploited information is the plants' distribution over the rows, as patterns having different color and/or texture from the soil, or as borders to be avoided. For the former cases Camera sensors are used (Klose et al., 2005, …), whereas distance sensors (Nagasaka Y., 2004, …) are used for the latter cases. Remote sensing, Satellite Images (Sørensen C. G. et al., 2004; Bochtis D., 2007, …), are also used as well as GPS (Pedersen T. S. et al., 2002, …) for navigation as well as data collecting. Our aim from developing a tool for field robots' autonomous navigation was to provide a framework within which data of such sensors can be fusioned coherently in one and unique model. The modelling approach we implemented allows such fusion. Starting from the fact that the typical environment of a field robot is a field of quasi-parallel rows culture, we modelled the robot's current surrounding environment with lines, relatively positioned to the robot's body. It has the advantage of having all sensors types' data fusioned in one model. In its current version, the fusion is applied to distance sensors; handling other types of sensors is envisaged in the future implementations.

From the distance sensors' positions and orientations (relative to their associated sides) expressed in the *Robot Configuration File*, and their sensed data, the tool computes the environment model's lines using the *Least Square Error Method*. This method allows to find the parameters of a line, that fits the best to the set of noisy sensors' data points. The more sensors a robot has, the more accurate is the approximation the robot constructs for the environment's borders (i.e. Figure 4). Figure 3 illustrates the current surrounding environment of the robot in Figure 2.

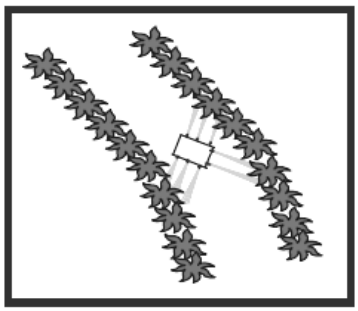

Figure 2: Robot in its typical Environment

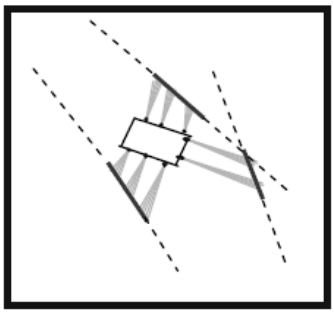

Figure 3: The Robot's Current Surrounding Environment

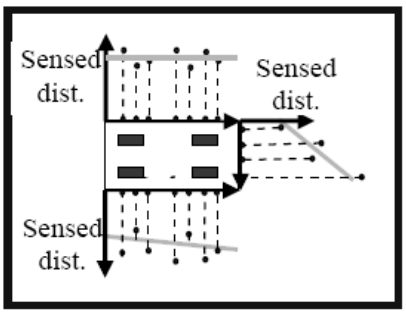

Figure 4: The Environment Model's Borders Computation[1]

With this modelling method, the quality of the environment's model will rely on that of the sensors. Even with low quality sensors the robot is able to produce a model, correct enough to navigate without collision, as proved the experiments we conducted.

By modelling the environment with lines, the robot will be able to navigate, between rows culture even with absence/missing of plants at one of the two rows. At any instant, it is enough for the robot to have at least two of its distance sensors, on at least one side (left/right), detecting the plants.

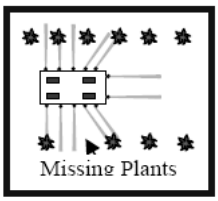

Figure 5: Missing Plants Situation

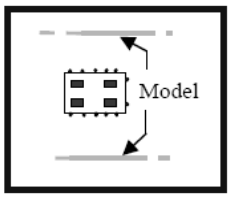

Figure 6: Correct Estimation of borders

2.2 Control Variables Selection

Fusionning the sensors' data in a model, arises the question of what control variables the Navigation Controller should base its move decisions on. A way to answer this question was to return to the definition of the desired navigation behaviour: *A robot should, if not sensing any thing with all its sensors, move straightforward with no steering* (no change of move's direction). *If sensing at one side* (i.e. left/right) *then it should try to be parallel to that side's sensed environment's border, in order to avoid any possible collision. It should avoid sensing any thing with its front's sensors, if possible, in order to have a free way in front of him, so it can move forward. Finally, it should navigate between the left & right rows, while trying to be at the middle of the path between the rows*.

[1] For this robot, there should be three sensors on each left and right sides and two on the front. Many distance sensors are displayed in this drawing, to accentuate the fact that the more sensors we have the more is accurate the environment's borders computation.

This desired behaviour can be reformulate, with a pseudo-mathematical form and/or constraints, as so:

- If the robot's body has an angle +θ with an environment's model line, it should steer -θ to be parallel, and in perfect orientation relative to that side (i.e. Figure 7 and Figure 8).
- If the robot is more near to one border than to the other, it should move away in a way that put itself, nearly, at equal distances to each border's sides (i.e. Figure 9). If the difference D_{diff} between the distances, d_{Left} and d_{Right}, of the robot to the left and right sides of the environment's model is negative the robot should steer to the right and vice-versa.
- The robot should move forward if it has enough distance in front of itself (i.e. Figure 10). We express this *enough distance* as the minimum distance between the left & right distances the robot has in front of itself, to the environment's border.

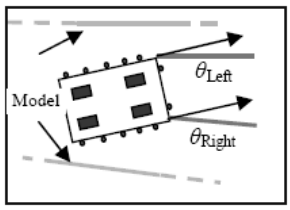

Figure7: Left & Right sides' Angles constraints

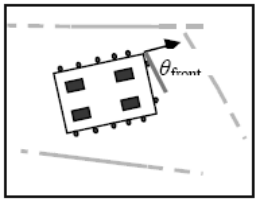

Figure8: Front's Angle constraint

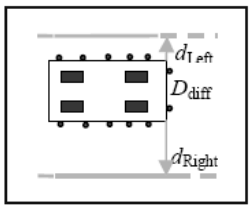

Figure9: Middle Line Constraint

Ddiff = dLeft – dRight

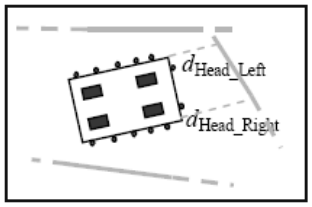

Figure10: Enough Distance Constraint

D_{Enough} = min (D_{Head_Left}, D_{Head_Right})

From this reformulation of the robot's ideal navigation behaviour, it becomes clear what control variables should be selected, to describe the current robot situation, relative to the surrounding environment. Table 1 summarizes these variables with their associated definitions.

Table 1: Navigation Control's Variables

θ_{Left}	Defines the angle that has the robot's left side, with the environment's border.
θ_{Right}	Defines the angle that has the robot's right side, with the environment's border.
θ_{Front}	Defines the angle that has the robot's front, with the environment's border.
D_{diff}	Defines the difference between the distances that have the robot on its left and right sides, with the environment's borders.
D_{Enough}	Defines the minimum distance the robot has in front of itself.

A robot should have at least two sensors, at each of the three sides (i.e. minimum points' count to have a line); which means a total number of 6 sensors. The number of control variables we need to design the control is 5 (see Table 1). We can observe that we obtained a reduction in the navigation control's complexity from 6 to 5 in this [worst] case. With the physical robot we experimented with, the complexity was reduced from 10 to 5. Whatever the number of sensors, the robot has, the navigation control will need only 5 control variables. This feature wasn't to be possible without our approach to sensor fusion, expressed by constructing a model of the surrounding environment. The next section will present the derivation method of the move's components.

2.3 Navigation Controller

The defined control variables are the most un/loosely-correlated variables, and each of which influences one output variable (all input variables, except D_{Enough}, influence the steering angle $\theta_{Steering}$; Output variable, *Velocity*, is uniquely influenced by the input variable D_{Enough}). Thus, each variable can be viewed separately from the others and its influence on the move decision (output variables) can be expressed by rules, without involving the other input variables. Being a rule-based method, *Fuzzy Logic Control* (Zadeh L., 2002) was the preferred method to implement the navigation control with.

We used the MATLAB's stand alone *Fuzzy Interpreter Engine* (i.e. *<MATLAB>\toolbox\fuzzy\fuzzy*) to interpret the rules set, defined in a configuration file and generated automatically by the

Autonomous Navigation Tool, after the 5 relations (θ_{Front}, $\theta_{steering}$), (θ_{Left}, $\theta_{steering}$), (θ_{Right}, $\theta_{steering}$), (θ_{Diff}, $\theta_{steering}$) and (D_{Enough}, *velocity*) being specified interactively in the dedicated graphical interface.

2.4 Control Design Interface

An interactive graphical interface was designed to express the influence a control variable has over the corresponding output variable, as a 2D curve. The specification of main points on the curve can be specified interactively by a simple click on the background. For a more precise specification of these points, we added the possibility of dividing each variable's support into equal intervals (with Major Lines) and if needed dividing each interval into equal sub intervals (with Minor Lines). Once the relation is drawn, the designer has the possibility to store it in a configuration file.

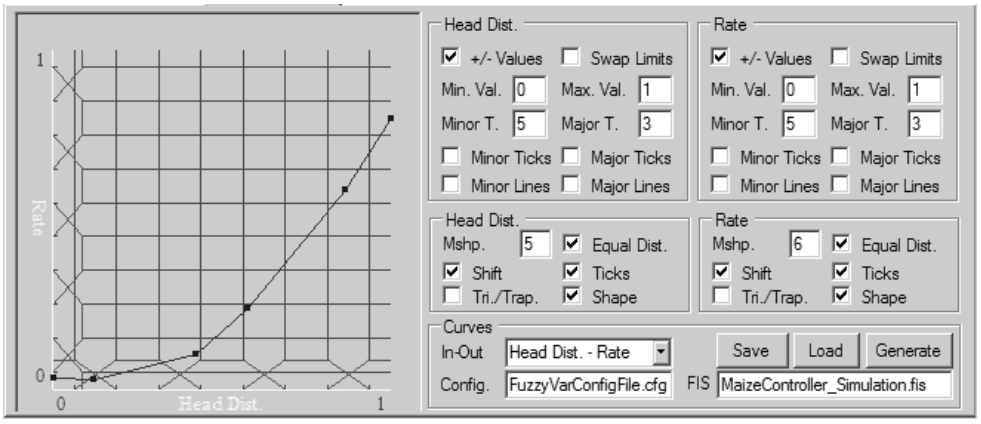

Figure 11: Navigation Control Design Interface

Figure 11, gives an example of the relation between the robot's velocity/rate and the free distance the robot has in front of itself. In this example, the robot is to be immobile while the normalized distance is less than 0.25 after that the robots increases its velocity/rate with the same proportion with which the distance increased. Other laws can also be used like Sigma law where the robot moves with two constant rates for two different intervals of the distance.

Once the input-output variables' relations specified, the support of these variable should be fuzzified. We mean by fuzzification, the definition of the fuzzy membership functions' shapes and numbers, over each variable's support. The most used functions are of a *Triangular* & *Trapezoidal* shape. The designer may split the support into any number of membership functions as well as changing the used shape. More membership functions leads to more fuzzy rules and thus lower the control's speed. The control designer should design the best control with the minimum number of fuzzy rules, expressed by the minimum number of fuzzy memberships for all the control's input and output variables.

3 TOOL'S FEATURES

The tool we developed has many important features, those that relate to the robot's autonomous navigation directly, are: *Missing Plants*, *in-between rows Navigation*, *Navigation in the middle*, *Rows Skipping*, *Field Detection*, and *Map Drawing*. They will be presented in the following sections. A detailed description of the whole tool's features is provided in (Zogheib 2008).

3.1 Missing Plants

Theoretically, a field is seen as a set of quasi-parallel rows culture, where plants are evenly distributed over the rows. Or, in practice plants may be missing. A navigation method should manage such exceptions, in a way that the robot navigates correctly.

The navigation method we implemented solved, in the most natural way, such exceptions. For a correct behaviour, it is enough to have at least two distance sensors sensing plants at, at least, one side.

The sequence, of Figure 13, shows how the robot was able to navigate correctly even in the absence of a consecutive set of plants.

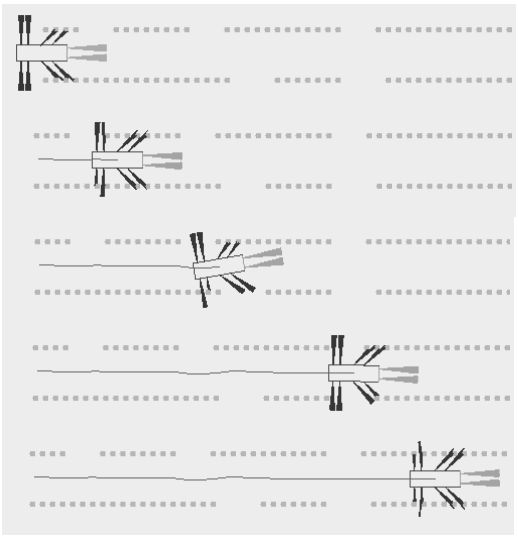

Figure 12: Navigation with Missing Plants

3.2 Navigation

The main task of a robot is to navigate in a field, between two rows. We added the ability for a robot to navigate on the field's extreme borders, along one row only, which can be defined as plant missing over a whole row. Being a development tool, the navigation tool allows simulating how the real robot will perform with the designed navigation strategy, in such environment.

The following two figures represent the sequences of pictures, giving an illustration of the robot's behavior for the two cases: *along one row* (Figure 14), and *in-between rows* (Figure 15).

Figure 13: Navigation Along One Row

Figure 14: Navigation Between Rows

3.3 Navigation in the Middle

The robot can take its way, between two rows, by being near to one border or taking a *zigzag* trajectory, …. Or, an ideal navigating robot should try, its best, to be in the middle between the two rows. With our set of control variables, this behavior is granted by design. The control variable D_{Diff} is mainly defined for this purpose only. Its role in the navigation control is to make the distance to the left row equals to that to the right one. The relation ($\theta_{Steering}$, D_{Diff}) defines how fast the robot will correct its position to be at the middle.

The following sequence of pictures represents how a robot, initially close to one row, will adjust its position to be in the middle, with progressive change in its orientation.

Figure 15: Navigation in the Middle

3.4 Field Detection

With this feature, it is enough to orient the robot in the direction of the field, wherever the person wants. Figure 17 shows how the robot moves straightforward when it does not sense anything, and how it adapts to the field's border, when it starts to sense a row in the field.

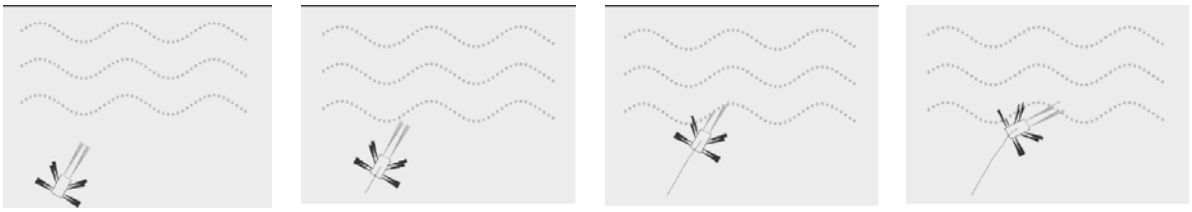

Figure 16: Field Detection

3.5 Rows Skipping

A feature that is of interest, for the robot's trip specification, is to order the robot to navigate starting at a specific row, and after navigation, to skip again another number of rows, then navigate and so on. This will give the possibility to define a trip as complex as one wants.

The following sequence of pictures shows, how a robot behaves when placed out of the field, and ordered to skip rows till detecting the third row, then turning to be in front of the path between two rows.

Figure 17: Rows Skipping

3.6 Field's Map Drawing

An interesting feature that derived, from modelling the robot's current surrounding environment with lines and from the deduction of its relative orientation to these lines, was the ability to draw the map of the traversed path, when the effective traversed distance and steered angles can be measured. The map is constructed by continuously connecting the environment model at time *t*-1 with the model at time *t*. The following sequence shows the map drawn by the tool, while the robot is traversing the path between two rows.

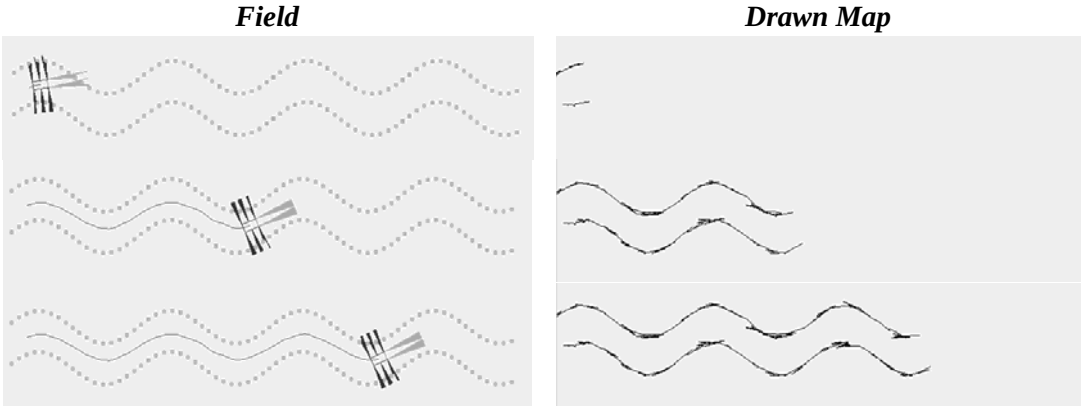

Figure 18: Map Drawing

3.7 Real Robot Controlling

For real world experiments, we used the robot OptoMaizer (Klose et al., 2005), developed at *Intelligent Sensors Systems* group at FH-Osnabrueck, Germany, for *Field Robot Event* 2005 competition. Due to weather conditions the experiments were limited to indoor space. The sequence of figures (Figue 21) shows how the robot navigated correctly in its typical environment, *in-between rows*. Features like *Map Drawing*, weren't possible to evaluate, with OptoMaizer, due to a hardware lack for a mean to measure the effective traversed distance as well as the rotated angle. It is expected that an upgraded robot will be ready, beginning of this summer, for outdoor experiments. A series of experiments are scheduled were all the tool's features will be tested and evaluated, on the upgraded robot.

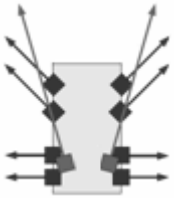

Figure 19: OptoMaizer's Sensor distribution[2]

In this experiment, the robot navigates between two rows of artificial maize plants. Periodically, it senses its environment, with its 10 distance sensors (4 at each left and right sides and 2 at the front). The sensed data are sent to the remote PC running the navigation tool. The model of the current environment is constructed; the control variables' values deduced, and the move decisions, returned by the *Fuzzy Controller*, are communicated back to the robot.

[2] Courtesy from (Klose et al., 2005)

Figure 20: Real World Experiment

4 FUTURE WORKS

In the current version of this work, distance sensors' data were fusioned and integrated in the environment's model construction. The integration of other types of sensors (i.e. Camera, remote sensing, …) in the computation of the environment model is possible, without any change, neither in the concept nor in the navigation control variables. Images can be processed to detect the rows surrounding, using Image Processing Techniques (Filtering , segmentation and Edge detection). The detected edges represent the future surrounding environment's borders and can be delayed then integrated with distance sensors' data to construct the environment. Remote sensing data, combined with GPS, can construct an approximation of the surrounding environment and the robot's relative position. The integration of all these sensors' types increases the model precision, and its immunity against wrong decision in case of missing plants, or any other source of confusion that may make the constructed environment model away from representing the real one.

A feature of interest, that the system allows, is to give the tool the possibility to predict the next surrounding environment's shape, based on previous recoded rows' map. Such functionality increases the robustness of the robot's navigation. The prediction of the whole field map's shape, based on the shape of the previously traversed row(s), and its integration in the environment model, is expected to be in the future version of the tool.

5 CONCLUSIONS

In this paper, we presented an Autonomous Navigation tool, for virtual and real robots; with which the design of the robot's navigation strategy is done interactively using a graphical interface. Five relations, between a defined set of control variables and the move commands, define the robot navigation's behaviour. In its current version, the tool integrates distance sensors data; the integration of other types of sensors does not require changes in the modelling nor in the defined control variables. Many features of the tool were presented, like Field Map Drawing, Missing Plants, Rows Skipping, Navigation Strategies' Design, …. Experiments conducted with a real robot, were in phase with the simulation results.

ACKNOWLEDGEMENTS

The author would like to thank Pr. Bengt Oelmann and Pr. Arno Ruckelshausen for their encouragement during his research done in their respective groups *Things That Think* and *Intelligent Sensors Systems* at Mid Sweden University and the University of Applied Sciences (FH-Osnabrueck, Germany). Many thanks go too, to Eng. Ralph Klose for making available OptoMaizer for the experiments with a real robot.

REFERENCES

Trebinski G., Bozsik B.: *Autonomous unit for one row seeding*. Conf. Proc. of 5th Int. Conference *Microprocessor Systems in Agriculture*. PW Plock, p: 225–234, 2004.

Van Evert F. K., Van Der Heijden G. W., Lotz L. A.P., Polder G., Lamaker A., De Jong A., Kuyper M. C., Groendijk E. J.K., Neeteson J. J., and Van Der Zalm T.: *A Mobile Field Robot with Vision-Based Detection of Volunteer Potato Plants in a Corn Crop*. Weed Science Society of America, Weed Technology Journal, Vol. 20, Issue 4, pp. 853–861, 2006.

Katupitiya J., Eaton R., Rodnay G., Cole A., Meyer C.: *Automation of an Agricultural Tractor for Fruit Picking*. IEEE Int. Conf. on Robotics and Automation, ICRA 2005, April 18-22, Barcelona, Spain, 2005.

Edan, Y.; Rogozin, D.; Flash, T.; Miles, G.E.: Robotic melon harvesting. IEEE Transactions on Robotics and Automation, Volume 16, Issue 6, p:831 – 835, Dec 2000.

Nagasaka Y., Zhang Q., Kanetani Y., Umeda N.: *Design of an autonomous field watching-dog for information collection*. 5th Int. Conf. Microprocessor Systems in Agriculture. PW Plock, p:120–130, 2004.

Pedersen T. S., Nielsen K. M., Andersen P., Bak T., Nielsen J. D.: *Development of an Autonomous Vehicle for Weed and Crop Registration*. Proceedings of the Int. Conf. on Agricultural Engineering, AgEng-2002, Budapest, Hungary 2002, 2002.

Sørensen C. G., Bak T., Jørgensen R.N. *Mission Planner for Agricultural Robotics*. Proc. AgEng 2004 Leuven, Belgium Int. Conf. on Agricultural Engineering. Abstract in Book of Abstracts, p.894-895, 2004.

Bochtis D., Vougioukas S., Tsatsarelis C. and Ampatzidis Y.: *Field Operation Planning for Agricultural Vehicles: A Hierarchical Modeling Framework*. Agricultural Engineering International: the CIGR Ejournal. Manuscript PM 06 021. Vol. IX. February, 2007.

Nagasaka Y., Grift T. E., Zhang Q., Knetani Y., Umeda N., Kokuryu T.: *Development of AGBO, A Laser Scanner Guided Autonomous Crop Scouting Robot*. Proc. of the 2nd Field Robot Event 2004, ISBN: 90-6754-818-9, p: 1-9, 2004.

Zadeh L.: *From computing with numbers to computing with words — from manipulation of measurements to manipulation of perceptions*, Int. Journal of Applied Math and Computer Science, pp. 307-324, vol. 12, no. 3, 2002.

Zogheib A.: *Field Robot Autonomous Navigation – Sensors Fusion-based*. Mid Sweden University, Master thesis 2008.

Klose R. and Meier M.: *Entwicklung eines autonomen Roboters auf Basis eines Multisensorkonzeptes* diploma thesis University of Applied Sciences, 2005

Field Robot Event Contest: www.fieldrobot.de, last accessed 25/3/2008.

* Ali ZOGHEIB, Mid Sweden University, Department of Information Technology and Media, Building S, SE-851 70 Sundsvall, Sweden.