Proceedings of the 8th

Prototype Contest

Field Robot Event

June 12th 2010

Technische Universität Braunschweig

Johann Heinrich von Thünen-Institut

Proceedings of the 8th FieldRobotEvent 2010

Braunschweig/Germany June 11 – 13, 2010

urn:nbn:de:gbv:084-11100508057

URL: http://www.digibib.tu-bs.de/?docid=00041345

Date:	Oktober 2011
Publisher:	Institute for Agricultural Machinery and Fluid Power
Contact:	Langer Kamp 19a
	38106 Braunschweig
	Germany
	Phone: +49 531-391-2672
Authors:	Prof. Dr.-Ing. Dr. h. c. Hans-H. Harms
	Prof. Dr.-Ing. Thorsten Lang
	Dipl.-Ing. Jan Schattenberg
	info@fieldrobotevent2010.de
	www.fieldrobotevent2010.de

Preface

The international Field Robot Event was launched in 2003 at the University of Wageningen (Netherlands) by Professor Müller, who now teaches at the University of Hohenheim. It was launched with the aim to inspire students for agricultural developments. Since then this event has developed to an international platform for students and experts to share knowledge across disciplines and to gain experience in the field of robotics.

After its establishment in 2003, the Field Robot Event was held again in 2004, 2005, 2007 and 2009 in Wageningen. In 2006 it was organised for the first time at a different venue, in this case at the University of Hohenheim and 2008 at the University of Applied Sciences Osnabrück. In 2007 the original competition in Wageningen was extended by a pupil competition, in order to introduce the younger generation to this topic. This was then taken up by Professor Ruckelshausen in Osnabrück in 2008 and since then it has established itself firmly. Last year both competitions were organised in Braunschweig by the Institute of Agricultural Machinery and Fluid Power at the Technische Universität Braunschweig.

I would like to thank the staff of the Johann Heinrich von Thünen-Institut (vTI) for giving us the opportunity to use their premises and assisting us with the organisation of the event.

Quite particularly I would like to thank the nearly 20 teams from six countries who took part in the event. There were more than 130 enthusiastic students, pupils and professionals. This was the highest number of participants in the history of the FieldRobotEvent and we were very glad to welcome them in Braunschweig.

[illegible]

Prof. Dr.-Ing. Dr. h. c. Hans-H. Harms

Partners

Special thanks to the partners of the event and especially to the vTI for making available the infrastructure. Without that support it would certainly not have been possible to align the event in Braunschweig. Thanks also to the many not directly referred supporters of the event, be it for creating the maize rows and maintenance of green areas, the installation of internet access and energy supply as well as support and advice in setting up the event.

vTI
Johann Heinrich
von Thünen-Institut

Technische
Universität
Braunschweig

Sponsors

Last but not least, have many thanks to all sponsors who supported the event financially. Without your support it wouldn´t have been possible to organize such a nice event:

Statements from the jurys point of view about fieldrobotics

Before the event, we have asked the jury to give us some statements from their ponit of view about fieldrobotics. Some of these questions and opinions were also taken up during the event and discussed. In the following these are now once again listed briefly.

Prof. Dr. Joachim Müller, Universität Hohenheim

1. **What are the origin and the initial objectives of the Field Robot Event?**

Origin of the FRE is Wageningen University, were as a freshly arrived professor I got the task from the Rector Magnificus to attract more students for Agricultural Engineering. A group of our Wageningen AgEng students won the ‘Award for Young Engineers’ at the CIGR-conference in Budapest with a remote controlled camera eyed robot. We decided to use this impetus and extend the activities to further universities and started the first FRE in 2003. Objective was hands-on education.

2. **Besides the FRE, what is “state of research” in Field Robotics? How much time will it take, until the first commercially distributed agricultural system will be established?**

First commercial systems are already established in form of milking robots. Those are stationary robots, working under controlled in-door conditions. Out-door conditions are much more challenging. Research is ongoing on robots for orchard spraying, weeding and fruit and vegetable harvesting. A commercial filed robot for plant rating was presented on Agritechnica 2009 by Amazonen-Werke.

3. **What are the main challenges at the current state of research? Why does it still last time, until systems will be accomplished competitive to manual solutions?**

Precise navigation in centimetre-precision under out-door conditions is still challenging. dGPS-navigation is still too expensive in relation to the small working width of field robots. Field robots are small (and presumably will stay small) because of safety reasons. Accident prevention of autonomous vehicles will remain a challenge. Here the multi-sensory system and brain of humans is unrivalled. However, certain tasks can be better done by robots than by humans, e.g. optical and sonic sensing tasks in infra- and ultra-spectral ranges.

4. **In further future, which kind of areas will be carried out under usage of these systems? Are there already agricultural disciplines/areas, where field robots are in use?**

Most promising areas are operations with following requirements and characteristics: small vehicles sufficient or even preferred (sub-canopy, low soil compaction), 24h-operation, repeated standards actions, harmful or tiring for humans, hyper-spectral sensing. Therefore, typical operations will be data collection, weeding, spraying, fruit harvesting.

5. **Will people become unemployed due to our work?**

Robots are doing that kind of work that humans don't like to do. For example for mechanical weeding it is difficult to find personell. New employment is created by development and production of robots, as well as by supervision of swarming robots during field operation.

Dr.-Ing. Josef Horstmann, Management Maschinenfabrik Bernard Krone GmbH

1. **From the industry's point of view, what future potential lies in the area of field robotics?**

From our point of view it is very important to develop robots for testing different jobs. There will come up new things, jobs and systems for robots in the future. Before there are solved a lot of issues (technical and security)

2. **What is industry's intention in keeping track of the Field Robot Event? Do you expect new ideas or new technological impulses out of the Field Robot Event?**

Yes we are expecting new technologies and sensor impulses out of these Events. Some of the new developments can be used later in existing machines f. e. to improve comfort or data management systems

3. **Are there current industry driven projects in the area of autonomous field robotics, or is this area limited to (universities') research?**

For the future this will not only be limited for the universities. The first autonomous tractors are already in the field in a test modus. But as in Q. 1 many things have to be solved first.

4. **How long will it take from your point of view, until the systems are ready for production? And what abilities are mandatory for the users to make these systems run properly?**

The safety issues are most important. So it is not possible today to say when the issues are solved. I think it will take another five years.

5. **Which kind of challenges have to be taken on, until autonomous systems will work in our fields?**

First we have to develop sensor systems f. e. cameras that are save enough to let an autonomous machine out into a field. Parallel do we need different or new laws to allow machines driving without the driver in the cab.

Dr.-Ing. Jens Möller, Management CLAAS Agrosystems GmbH & Co. KG

1. **From the industry's point of view, what future potential lies in the area of field robotics?**

Until now costumers have pushed for larger machines, to gain efficiency, but we are reaching the limits of this path. At CLAAS we believe autonomously operating vehicles and close integration with implements to be the next big step for higher work quality, efficiency and lower emissions, pushing agriculture towards a more sustainable and energy efficient production.

2. **What is industry's intention in keeping track of the Field Robot Event? Do you expect new ideas or new technological impulses out of the Field Robot Event?**

The Field Robot Event is an on field demonstration of the latest research within field robotics, and as such, it is a good way to keep track of the advances within this important area. Also, the event provides the opportunity to spot when it is the right time for the industry to adopt the research and initiate technology projects within our company. Also the many talented students participating, provides an opportunity for recrouting new employees.

3. **Are there current industry driven projects in the area of autonomous field robotics, or is this area limited to (universities') research?**

The technologies involved in autonomous field robotics are developing very fast. In order to explore the real potential of the different advanced technologies, seen from a commercial or industrial point of view, it is necessary for the industry to run its own projects. Within the last decade auto-steering systems have emerged as an important technology step, both easing operation of the large high capacity farm machines and also leading to increased productivity. Today's auto-steering systems are primarily based on GPS solutions like the CLAAS GPSPilot. Even though the GPS technology has been subject for extensive research, CLAAS and in general the whole industry, are constantly researching this area as part of both internally and externally funded research projects. Key driver in the field robotic projects are close "integration" of technologies; both within the vehicle / implement and to/from remote information systems.

4. **How long will it take from your point of view, until the systems are ready for production? And what abilities are mandatory for the users to make these systems run properly? Which kind of challenges have to be taken on, until autonomous systems will work in our fields?**

It is hard to say exactly how many years will elapse before we see fully autonomous field machines in production, as many issues still need research. To take the next step from auto-steering to more autonomous operation, will require new solutions to especially the issues of environment understanding and various safety aspects.
The requirement for a cognitive system, able to "see" and interpret the ever changing natural surroundings on the field and to produce a suitable intelligent reaction, is a major challenge ahead of us.
Another challenge will be the system reliability, as there is no one on-board to help if a break-down occurs or the system encounters operational problems that might cause damage to e.g. the crop. Supervisory systems, for reliable monitoring of the actual on board quality crop production, is today in a very early phase. Without such a reliability monitoring

systems, the market acceptance of an autonomously operated vehicle will likely be very low. Today's systems thus leave the driver to take care of unforeseen events e.g. safety aspects, and thus much is still to be developed before autonomously operating field vehicles can be broadly implemented.

Prof. Dr.-Ing. Walter Schumacher, Institute of Control Engineering, TU BRaunschweig

1. **What are the pro’s and con’s of the Field Robot Event from the educational perspective?**

Events like this one provide a strong motivation for students to solve real world problems by extending their professional knowledge. They immediately see the results of their efforts and get feedback from their design decisions. The problems to be solved are far from trivial, being only slightly simplified by the clear rules of the contest. Students usually invest much time in preparation of the contest, which might be in conflict with the progress of their studies. But the experience gained will prove (will be) a great benefit for their whole life.

2. **What do you think about the knowledge transfer from field robotics into other disciplines, such as yours? Can it be compared to the “Carolinchen”-Challenge (Carolo-Cup)?**

The tasks that have to be accomplished can be divided into several application areas, such as cognition of the environment, path planning, motion control of the vehicle. These are quite common in many scenarios: autonomous driving of large as well as small cars or navigation of automated guided vehicles on the factory floor. But these are only the narrower technical aspects. The students also learn a lot of soft skills like the approach of solving engineering problems and cooperation in a team.

3. **What potential does the Field Robot Event have for students / semi-professionals / professionals?**

The rapid development of electronics and computers makes especially students apply the latest technology in their projects. They are more likely to try new approaches and deviate from well trodden trails than long serving professionals, who did similar tasks before. So

there are strong benefits from contests like this one for the whole community, students and professionals alike.

4. **For Braunschweig, being one of the locations of the Automotive Research Centre of Lower Saxony (NFF), and for the University of Braunschweig: what kind of sustained success can be achieved in holding the FRE2010?**

This event is another contribution to promote Braunschweig and its Technische Universität and the Automotive Research Centre of Lower Saxony in public awareness as a key location for the research in vehicle technology. I see the FRE2010 well in line with a number of past and current projects as the project Auonomes Fahren 1996-99, the team Carolo taking part in the DARPA Urban Challenge in 2007 and the yearly Carolo Cup as a student contest.

Index

Contest information

Below you will find the rules of the Field Robot Event 2010 (as of June 7, 2010). Any subsequent changes are available on the website www.fieldrobotevent2010.de.

Task 1: "Basic"

Within three minutes the robot has to navigate through long curved rows of a maize field to cover as much distance as possible. On the headland it has to turn and return in the adjacent row. There will be no plants missing in the rows. This task is all about accuracy and smoothness of operation within the rows. The headland turning is not as important as in the last year's events.

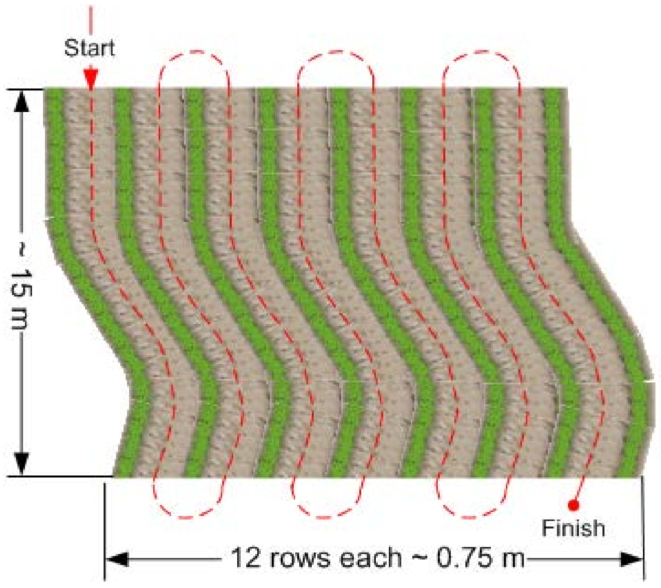


Figure 1: Task 1 "Basic"

Assessment:

1. The distance travelled in 3 minutes is measured. If the end of field is reached within this time, the total time counts. Distance and time are observed by officials.
2. Touching the robot *within the rows* results in a penalty of 5 metres (per touch). The number of touches is counted by the officials.
3. A manual intervention *at the end of a row* to help the robot entering the next row will be punished with a penalty of only 2 metres.
4. Destroying a plant (e.g. kinked maize stem) results in a penalty of 1 metre (per plant). The officials will decide whether a plant is broken or not.
5. Distance and time results in a team ranking.
6. The overall points for the Field Robot Event 2010 Champion will be given as follows (similar to Formula1 point system): First place in this task: 10 points - Second place: 8 points - Third place: 6 points - …5-4-3-2-1-1-1-1… points. Not participating in this task results in 0 points.

Task 2: “Advanced”

The robot should cover as much distance as possible within 3 minutes while navigating between straight rows of maize plants. It should be able to follow a certain pre-defined pattern over the field. At various places in the maize field, plants will be missing in either one or both rows over a length of maximally 1 metre. The headland border may not be perpendicular to the crop row orientation. The difference in length of two subsequent rows will be less than 1 metre. A headland of only 1.5 metres will be available for turning (see assessment 4).

Coding of the row-pattern through the maize field is done as follows. S means start, L means left-hand turn, R means right-hand turn and F means finish. The number before the L or R represents the row that has to be entered after the turn and the single number 0 means return in the same path. So, 2L means: enter the second row after a left-hand turn. 3R means: enter the third row after a right hand turn. The row shown in figure [coming soon] is coded as follows: S - 4L - 0 - 3L - 3R - 1R - 3L - 1R - F.

The code of the pattern is made available to the competitors 10 minutes before the start of the competition without having the opportunity to test it in the maize rows.

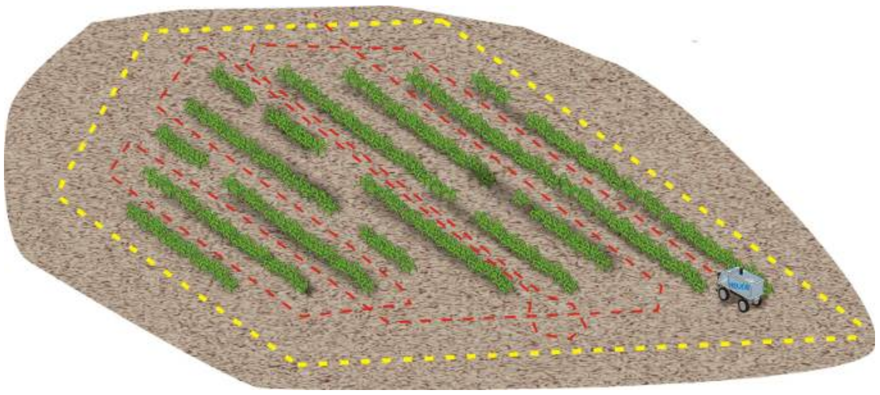


Figure 2: Task 2 "Advanced"

Assessment:

1. The distance travelled in 3 minutes is measured. If the end of field is reached within this time, the total time counts. Distance and time are observed by officials.
2. Touching the robot results in a penalty of 5 metres (per touch). The number of touches is counted by the officials.
3. If the robot enters the wrong row after the headland turning, it results in a penalty of 5m. The penalty for any interaction during the headland will be 8m. Anyway the vehicle must be set into the correct row by hand if the headland turning was not successful.

4. Crossing the headland boundary located at the end of the rows by a distance of more than 1.5 metres or twice the length of the robot results in a penalty of 5 metres per crossing; number of crossings are counted by officials.
5. Destroying a plant (e.g. kinked maize stem) results in a penalty of 1m (per plant).
6. Distance and time results in a team ranking. The following sequence for the overall points for the Field Robot Event 2010 is used: 10-8-6-5-4-3-2-1-1-1-1… Not participating in this task results in 0 points.

Task 3: “Professional”

The “Professional Task” consists of two subtasks. First the teams will have to demonstrate their weed handling device and explain to a jury and the audience how it works. Afterwards they have to demonstrate their weed detection system.

Subtask 1: “Weed-Handling-Device”-Demonstration

The teams have to present a weed handling device. Within a 5 metres long straight maize row they have to prove its functionality as well as its efficiency.

Every team can use its own type of weed with a self defined shape and colour. The weed will be placed by the jury in between the maize plants either on the left or the right side of the row. The jury will judge the function as well as the efficiency of the device. A realistic type of weed as well as an economic and ecological extinction device will be honoured by the jury.

Subtask 2: “Weed-Detection”-Demonstration

The robot should cover as much distance within 3 minutes while navigating through straight rows of maize. Between the maize plants randomly distributed artificial weeds have to be detected. Plastic flowers will be used for the weeds (details following soon). The successful detection has to be characterized by an audible or visual signal. Additionally it must be shown on which side of the row the weed has been detected. It is not required to ‘handle’ the weed if you do not want to. Anyway you will not get additional points for a ‘extinction action’.

At the headland the robot has to do a headland turn and return in the next row. In between the first 2 metres at the beginning of each row there can be an obstacle within the rows (e.g. an additional maize plant). In this case the robot has to leave the row and enter the next row.

In this part there will not be any jury points. Only the “hard facts” will be scored by the officials.

Update:

The ratio of correct detections - correct position as well as correct side - to false detections will be multiplied by the travelled distance (minus penalty meters) and added as bonus meters.Example: Ten plants counted but only five correct detections - Ratio: 0.5. The travelled distance (minus penalty meters) is 20m. This results in an overall distance of: 20m + 20m * 0.5 = 30m.

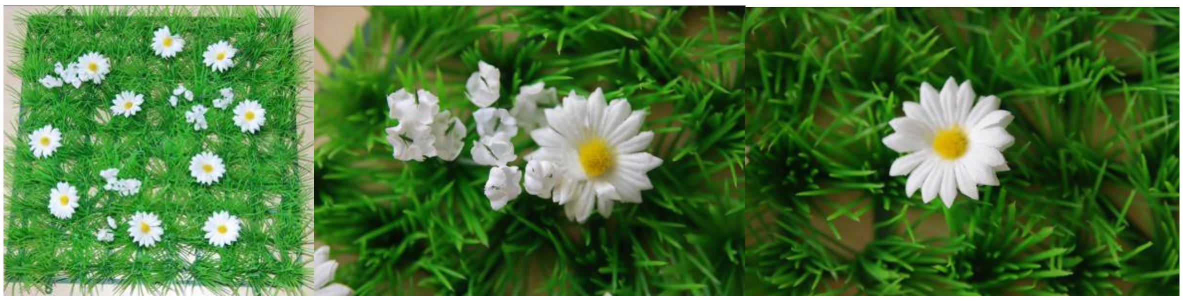


Figure 3: Taks 3 "Professional"

Assessment:

1. "Weed-Handling-Device" – Demonstration
    a. The jury ranks all robots at the end of the extinction device demonstration subtask. The points for this subtask are based on ranking number. The following sequence is used: 10-8-6-5-4-3-2-1-1-1-1…
    b. If the control device does not work during the first run, the team will get one additional chance after all the other robots.
    c. A realistic type of weed as well as an economic and ecological handling device will be honoured by the jury and result in a better ranking.

2. "Weed-Detection" – Demonstration
    a. The distance travelled in 3 minutes is measured. If the end of field is reached within this time, the total time counts. Distance and time are observed by officials.
    b. Touching the robot results in a penalty of 5 metres (per touch). The number of touches is counted by the officials.
    c. A manual intervention at the end of a row to help the robot entering the next row will be punished with a penalty of 8 metres.
    d. Destroying a plant (e.g. kinked maize stem) results in a penalty of 1 metre (per plant).
    e. Distance and time results in a team ranking. The following sequence is used: 10-8-6-5-4-3-2-1-1-1-1…

3. The points from both subtasks will be added. If two teams have the same number of points, the team with the better detection device (subtask 2) will be ranked higher.
4. The following sequence for the overall points for the Field Robot Event 2010 Champion is used: 10-8-6-5-4-3-2-1-1-1-1… Not participating in this task results in 0 points.

Task 4: "Cooperative Challenge"

Cooperation between one or more robots has to be demonstrated. There is no given task the robots have to fulfil. The robots can drive, fly or even swim (if it is raining cats and dogs). The application should have an agricultural background and has to be shown on the field.

To enforce the exchange between all participating teams, the jury points will be multiplied by a factor. If a single team or two teams from the same university / city show(s) their co-operating idea, the jury points will be multiplied by 1. Teams from different cities within the same country will get their points multiplied by 1.5. For cross-country cooperation the points will be multiplied by 2. No more than two teams should cooperate. The teams have to find cooperating partners on their own (e.g. teams from older events).

The teams have to submit a paper before the event starts (not more than one page) to inform the jury as well as the audience about their idea.

Assessment:

1. The jury ranks all the robots after the performance of all teams.
2. The idea and the quality of the demonstration are most important.
3. This task is optional and will be awarded separately. There will be no overall points for the Field Robot Event 2010.

Task 5: "Freestyle"

Robots are invited to perform a free-style operation on the field. Fun is important in this task as well as an application-oriented performance. One team member has to inform the jury and the audience about the idea.

Assessment:

1. The jury ranks all the robot performances at the end of the task.

This task is optional and will be awarded separately. There will be no overall points for the Field Robot Event 2010.

Additional Task Information:

- Unlike the last years, there won't be any jury points for the basic, advanced and professional[1] tasks. Only the "hard facts" will be considered by the officials.
- During the tasks the robots will have to wait in a Parc Fermé, so that no further testing or modification is possible. Between the tasks there will be a 10 minute break for the teams to prepare their robots for the next challenge (change batteries, etc.).
- From the moment a robot is given permission to start, it must start within one minute. If the robot doesn't start within this time, it has one more chance to start after all other teams. If it does not start within one minute for the second time, the robot is disqualified for that task.
- Large robots and/or robots with a probability of destroying the field will always start at the end of the task (after all second chances restarted again).
- There will be an award for the first three ranks of each task. The basic, advanced and professional tasks together will yield the overall winner of the Field Robot Event 2010.
- If two or more teams have the same number of points for the overall ranking, the team with the better placements during all three tasks will be ranked higher.

For the first three disciplines it is not allowed to use GPS (or rather GNSS). It is allowed only for the cooperating as well as the freestyle task.

Before the start every team has to explain to the officials, which kind of hardware they are using. If they are using simple hardware (e.g. infrared or ultrasonic distance sensors combined with cheap microcontrollers) instead of high end equipment (e.g. embedded PCs, laser range finder), they will get 6 additional points. For a medium complex solution 3 additional points will be given.

[1] Only the weed killing device will be judged by the jury (see task 3).

Robot Information

AbuRoBo Field Robot Team

University of Tehran

Sina Valizadeh, Seyed Vahid Mirnezami, Nastaran Rezaei

Supervisor: Akbar Arabhosseini

Agrotechnology Departement, College of Abouraihan, University of Tehran, Iran

[*] Corresponding author: College of Abouraihan, University of Tehran, P.O.Box 11365-4117 - Tehran – Iran Tel:+98 292 30 40 614; fax: +98 292 30 40 730; email: sinav558@yahoo.com

Abstract

The AbuRoBo team was established in 2008 by a group of four B.Sc students in the field of agricultural technology in College of Abouraihan, University of Tehran. The team's aim was to design and build a simple robot using low cost sensors and aluminum chassis and rubber track which is applicable in farms. The second version of the robot was built coming with fiberglass chassis and again rubber track in 2010 but the navigation system was based on image processing. The robot is able to follow the defined color and react to certain colors.

1. Mechanics

The chassis was made of fiberglass to provide low weight, high stiffness and adequate rigidity. Since a laptop is embedded on the robot, the mechanics has to provide sufficient rigidity and high stability, therefore an innovative suspension system was developed. The main advantage of this method is that the suspension stiffness can be adjusted by easily moving springs acting point. The track method had many difficulties which was hard to come along with the system so we also tested the wheels which were taken from RC car. The chain track and rubber track were studied to find out which one is more applicable and useful in this case. The traction of the rubber track was checked doing lots of tests. The final decision was made to use rubber track on the robot according to high traction using rubber track.

2. Vision

Two cameras were located on the front and the rear of the chassis in order to prepare relevant conditions for easy and efficient programming. The locations of the cams are very important factor because the basics of the programming are dependent upon the locations. Here we used ‘A4tech’ cam for both front and rear and both of them connected to portable computer via USB cable (Fig.1). Another important factor is the height of cams above the ground. Since the maize row widths explicit and determined degree of maize row about the horizon, by using some easy maths, the cam height was calculated to be 80cm.

Figure 1 the camera used for AbuRoBo

3. Navigation

The MATLAB and the Image Processing toolbox were used as navigation software to control the movement of the robot and keep moving between the rows. When the moving path is found by software, an electronic board receives the data from the laptop via LPT port and controls the rotational speed and direction of the motors. Although the MATLAB software is weak and slow in real time job, but we found it easy to use and ideal for this case. The MATLAB program filters the images taken by cams. This filter is divided into two main filtering. First they are filtered by the toolbox itself and then by the custom program.

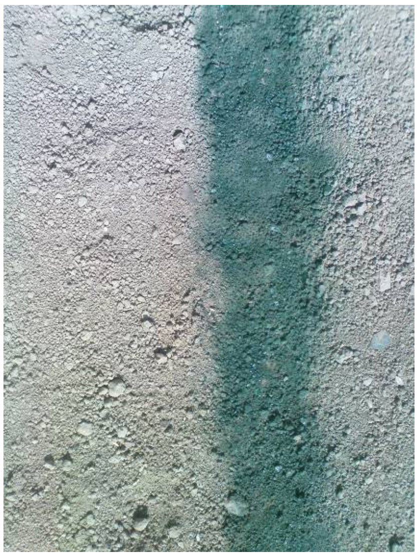


Figure 2

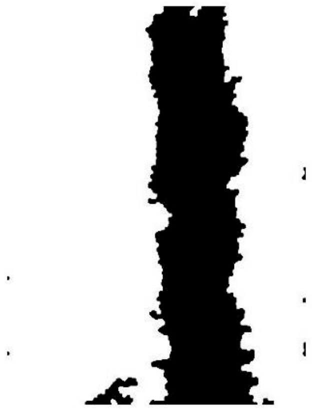


Figure 3

The program is written according to competition circumstances. The robot can be directed if there is only one row in the image, taken by the cam and this means that if there are two rows in the image then the robot recognizes to move within the rows and takes no action unless there is just one row in the image. Then the image is assumed as a 2D image and the row angle will be calculated. The slope must be within specific interval otherwise it means that the robot is getting too close to a row. When the robot has reached to the end of each row, it’s time for rear cam to do its job to keep the robot away from the last ending maize plant and guide it to enter the row B from row A.

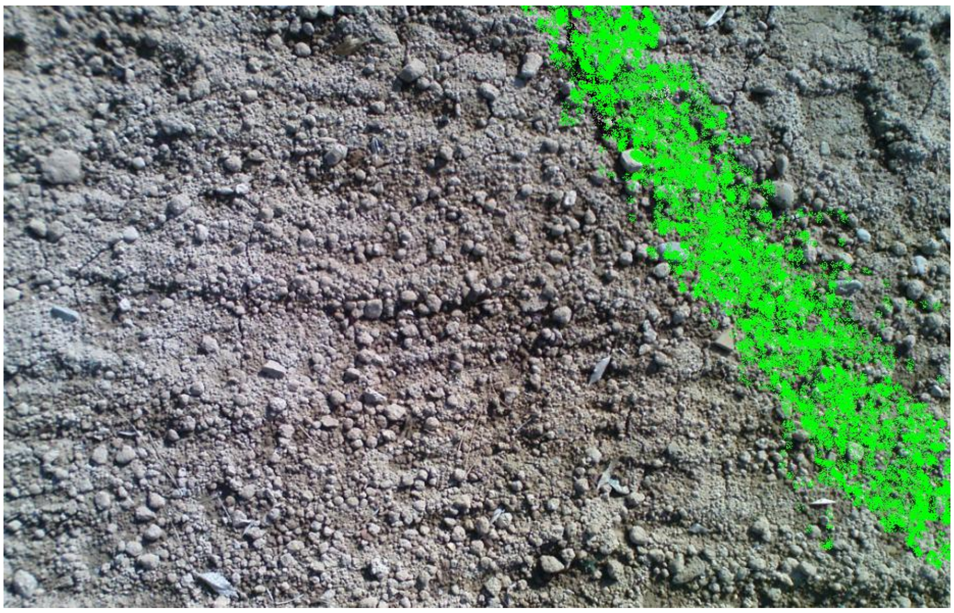


Figure 4

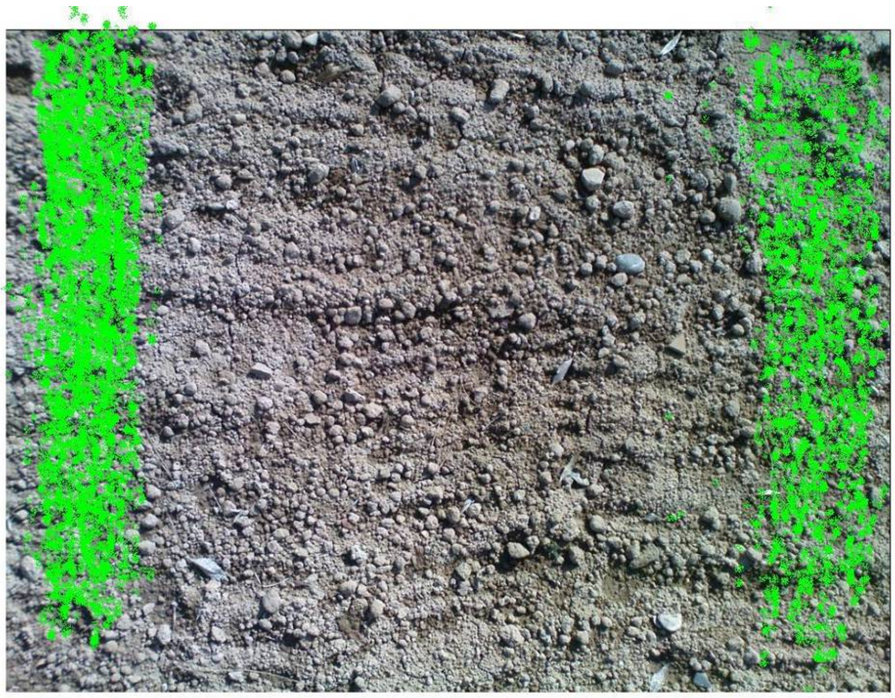


Figure 5

4. Steering

The robot should be able to make head turns from row to row at the end of the crop line. When the robot reaches to the end of each row and goes out of the row A, it steers till the first maize appears in the image and again by calculating the angle it finds its way normal to the rows till it gets to specified row and then it steers and enters to the row B.

5. Hardware

There will be laptop on the robot to which cameras are connected, the operating program is installed and the electronic board is attached. The electronic board is very simple and it’s been achieved and no microprocessor has been used to control the motors. The power will be supplied using a 12v lithium battery for supporting of two motors and the electronic board while the laptop uses its own battery.

6. Conclusion

Field robot is all about designing and building an autonomous robot that has to find its way through an unpredictable field. there may be some maize missing and the weather may change anytime during the competition, thus there can be too many ideas and innovative designs. Field robot is an approach to mechanization of today agricultural and its first step for students to get familiar with real engineering world and difficulties there may occur in actual world and manage their educational priorities. The robot was built on a low budget including two cams and rubber track system which is able to move in farm and follow the crop rows and it is also able to turn to the next row when it reaches to the end of each row.

7. Reference

Professors of our department were supported the team for all required information.

Ceres

Rik Boonen, Bart Bouten, Erik Hoedemaekers, Jeroen van de Mortel, Ben van Seggelen, Daan Verstegen, Frank van Gennip (Tutor)

University of Applied Sciences Fontys Venlo, Tegelseweg 255, 5912 BG Venlo, the Nederland

Contact: f.vangennip@fontys.nl

1. Introduction

The autonomous fieldrobot Ceres (Figure 2.1) has been built by a group of six students of the University of Applied Sciences Fontys Venlo to participate at the Field Robot Event 2010 in Brauwnsweig. The robot build in 5 months must navigate through a cornfield and pending on the given challenge enter a new row.

To accomplish this the Ceres Team has chosen an Vision camera with a curved, round mirror on top, so that the camera has a 360° field of view. The drive motors are placed in the separated motor modules and the “brain” of the robot is placed in the upper module. How these modules, electronics and software work is explained in the following chapters.

2. Hardware concept (e.g.)

2.1 Mechanics

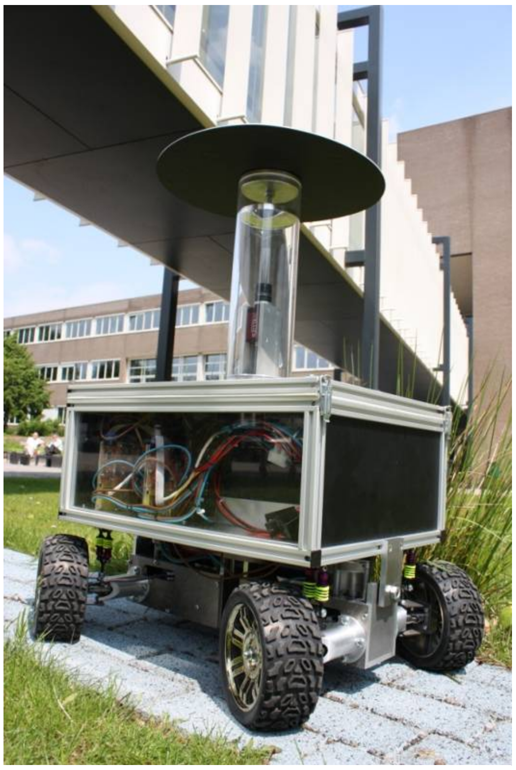


Figure 2.1 Fieldrobot "Ceres"

When looking at the mechanics, the robot can be split into three parts. The part for the camera (mirror), the chassis and the motor modules. In this chapter we will only explain the chassis and the motor modules.
The Chassis is made of an aluminum Bosch profile. This is strong and it's fairly easy to construct the box for all the electronics. The sides are closed with some Plexiglas.

The motor modules are two identical modules. The cad-drawing of this module is show in figure 2.2. The motors used are a Maxon RE-40 motor with gearbox(26:1) for driving and a wiper motor from a car is used for steering.

The motor module is able to steer the wheels to an angle of 60 degrees. This makes it possible to make very sharp turns and drive into the next row in one turn. At the beginning of the project we calculated we needed 55 degrees to make a nice u-turn to the next row. This made the design more complicated. Because of the 60 degree steering possibility there are double universal joints at every wheel. When steering, the axle will vary in length. To handle this varying axle length we used a spline axle.

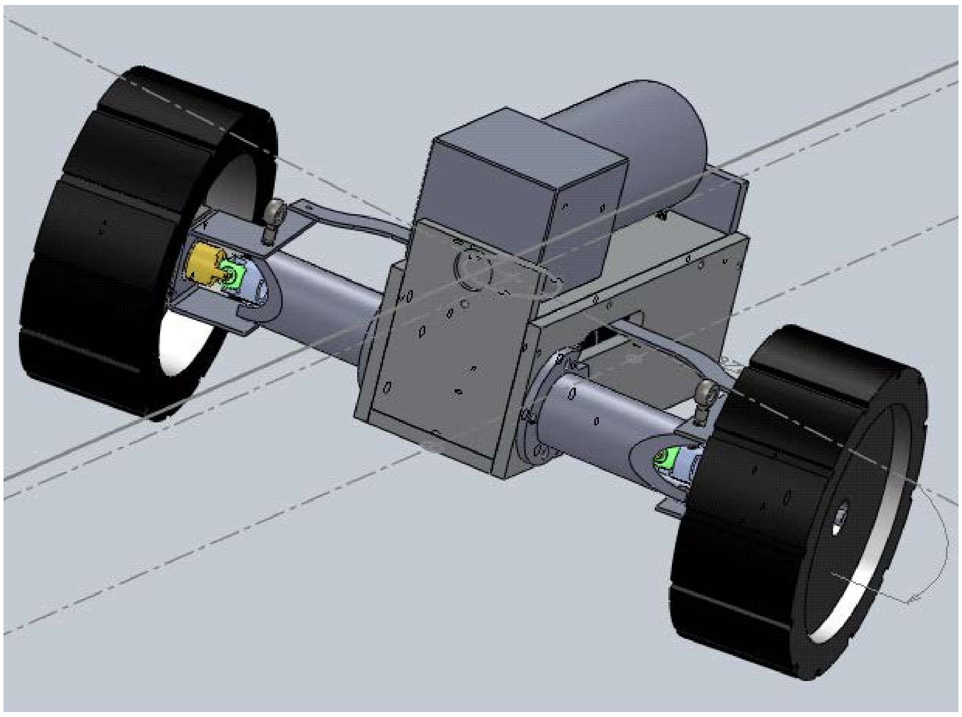


Figure 2.2 Motor module

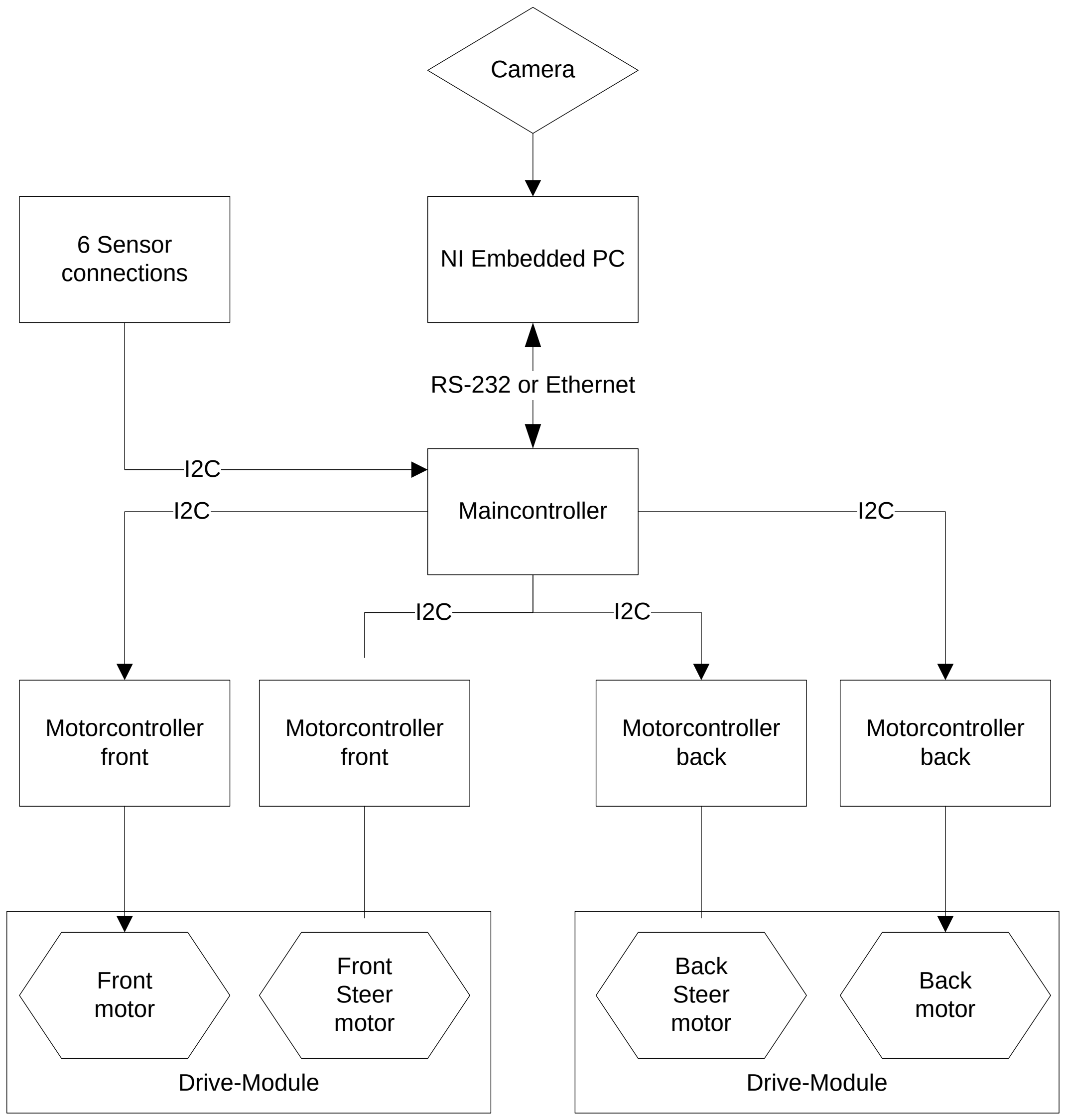
The biggest problem in the robot was steering the robot. At first we used two simple servo's to steer. But because of the wheels had so much friction with the underground the two servo's broke down internally. For this we needed more power to steer so we went looking for high torque dc-motors and came up with a wiper motor. Even the first wiper motor wasn't strong enough to steer we got an even stronger one, to steer more easily. Next year they should look for new wheels so they can implement smaller motors for steering, making the robot lighter.

3. Electronics

The electronics of the robot was completely new, and developed by ourselves. The idea of making all the electronics by ourselves, was that we weren't dependent of other suppliers. So if something would go wrong we were able to repair it without support of some company. Another advantage was that the design could be changed to our needs which makes it a lot easier to develop a robot.

3.1 Electronic modules

The electronics consist of four main modules the embedded PC (NI EVS-1464RT) from National Instruments, a main controller and two drive modules with each two motor controllers (one for the drive motor, one for the steer motor) (Figure 3.1). And there is one Wi-Fi router in the robot to generate a wireless connection with our laptops. Everything is powered by lead acid batteries. One 12V 12Ah battery for the embedded PC, main controller and the Wi-Fi router and two 12V 7.2Ah batteries in series for the four motors.



Figuur 3.1 Electronic schematic overview

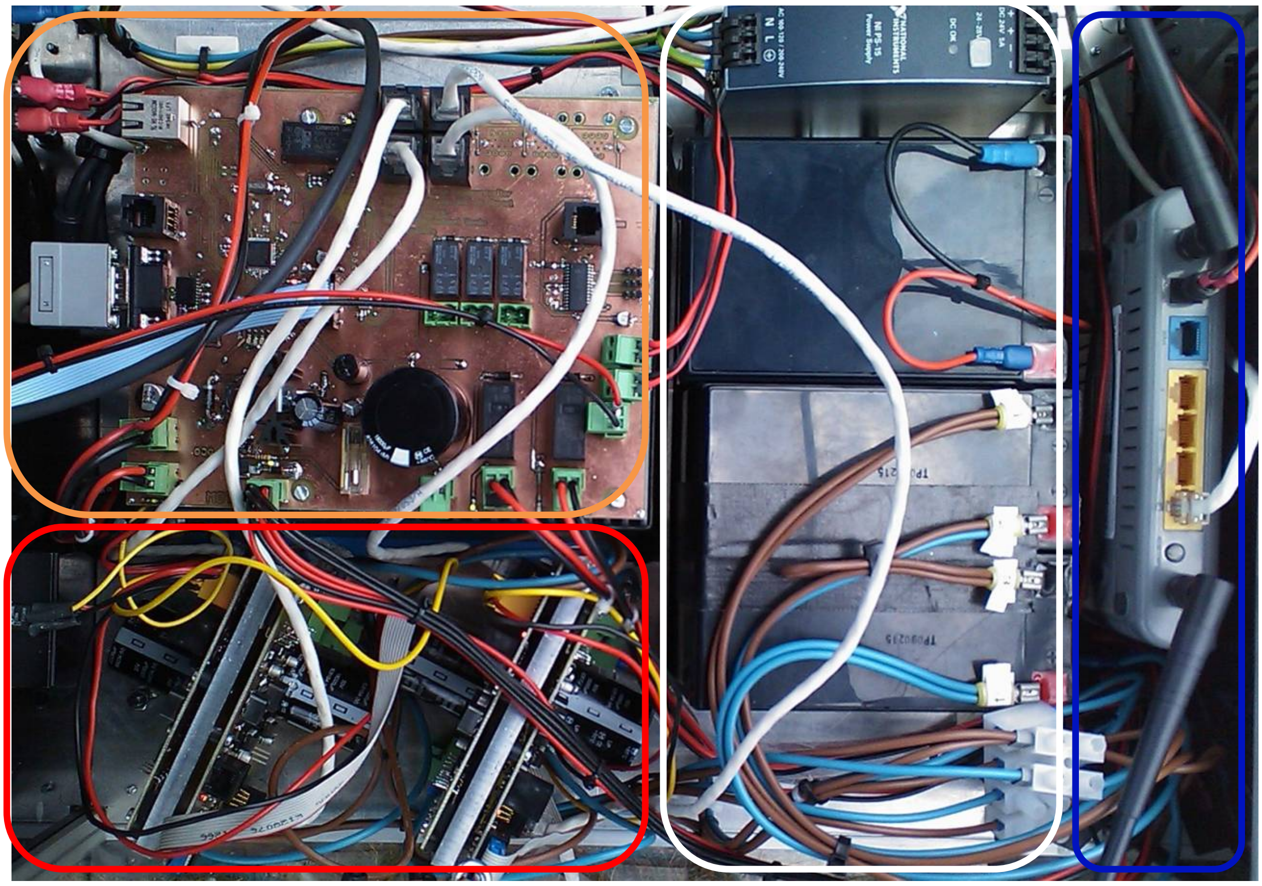


Figure 3.3 Orange-Main controller, Red-Motor controllers, White-Batteries and power supply, Blue-Wi-Fi router

3.2 Main controller

The main controller is mainly intended to converted the data that is coming from the embedded PC over RS-232 to I2C. But it does more than that, de controller has two power inputs, 12V from the battery and 24V from the industrial power supply. It automatically switches between 12 and 24V, it takes the 24V input if it is there. This is done so we can turn the robot on without batteries or changing the batteries without shutting down the electronics and the embedded PC. The main controllers also checks if the 12V battery comes below 10V, if this happens the main controller automatically shuts the motor controllers off and gives a message that the battery is empty. The main controller also has an Ethernet port onboard to communicate with the embedded PC, but due the time limit we didn't make the software for it.

3.3 Motor controllers

To control the motors we made 4 motor controllers one for each motor. The motor controllers are basically the same except for the sensor connection. Because the drive motors has optical encoders with RS-422 line drivers, and the steer motors have a potentiometer. The motor controller consist of a PIC18F2431 microcontroller, the microcontroller drives the LT1336 bridge drivers and they drive the four IRFP3206 MOSFETs which can handle 120A continuous and 890A pulsed. But the motor controller is limited to 15A continuous current because of the PCB trace width. The power electronics and the control electronics are fully optical isolated. The motor controllers get their setpoints from the main controller over I2C.

The drive motor controller controls the speed of the motor with a 500Hz PI control loop, the steer controllers have a 100Hz P control loop. All the code for the controllers are written in C. There is still a lot of improvement needed for the software. For example the steer controller needs some kind of integrator so that the error can be regulated to zero.

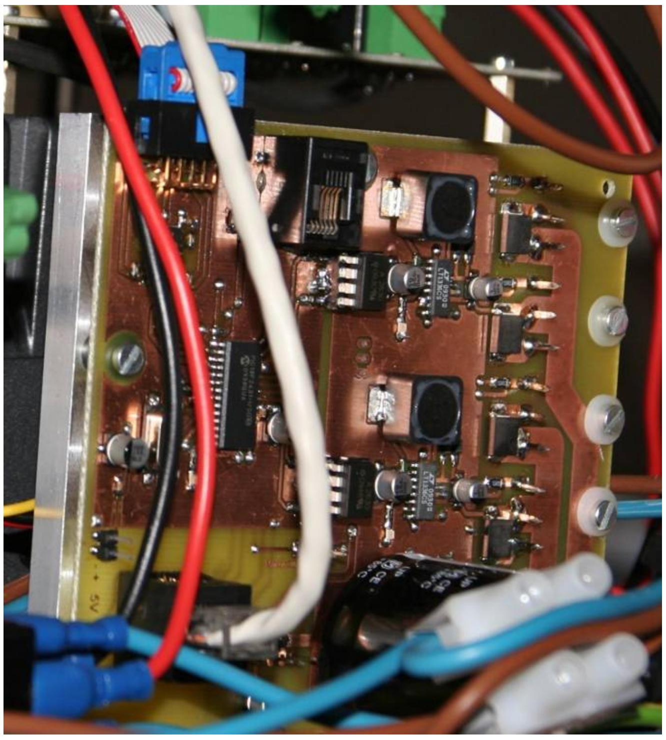


Figure 3.3 One of the 4 motor controllers

4. Software

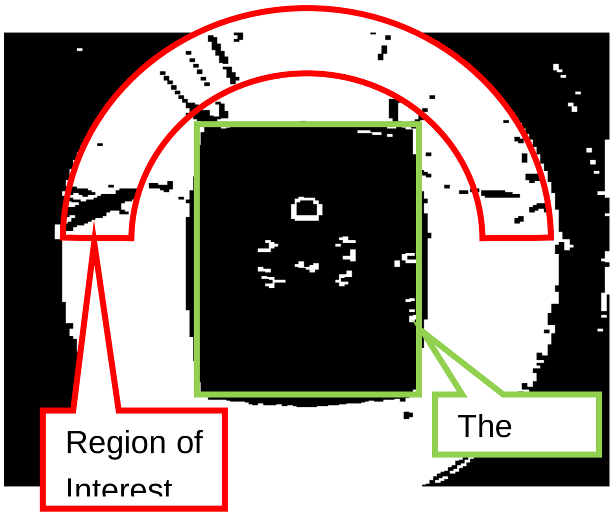
Besides the software in our microcontrollers all the software is writing in National Instruments' LabView, the largest part of this is done by graphical programming and small dedicated pieces are written in c-code which is imported into LabView. This chapter explains the main program and the most important pieces of lower-level programs, like the image filtering, navigation, end row detection, the end row navigation (containing the row count) and weed detection.

4.1 Main program

The main program contains the entire software to control our robot. This program is divided in different pieces, each with their own task. Some of these pieces are the UDP communication for controlling the robot by an external laptop, the serial communication to enable the PC to control the motors and an image acquisition piece.
As mentioned we use UDP for controlling the robot externally, therefore we wrote another program, remote. This program is capable of selecting the right challenge to do and even of controlling the robot manually with a gamepad.
The following paragraphs show the link between the image acquisition and the serial communication, which determines the direction and speed of the robot when it drives autonomously.

4.2 Image filtering

The only sensor we use for navigation through the maize field is a fire wire camera which has a 360° view around the robot. Before our algorithm can determine a direction from this image it has to be filtered an digitalized. The first thing we do is applying an excessive green algorithm to the image which multiplies the green pallet by 2 and then subtracts the red and blue pallet from it. This algorithm brings out the green in the image and directly converts it into a grayscale image.
After this filtering we use a simple threshold to make the image digital and suitable for the navigation software. Besides that we use some basic morphology to filter out small particles and fill small holes in this binary image (see figure 4.1 *Binary image*).



4.3 Navigation

As shown in figure 4.1 *Binary image*, our region of interest is a half round shape which looks in front of the robot as well as on both of its sides. The first thing we do with this image is unfolding it into a rectangle, after that we turn it into an array of heights. This means that we from the bottom (inner circle) to the top (outer circle) and determine how far from the inner circle the first object appears in the image. This distance is than saved into an array of heights, representing vectors.
This array is used to find the gap to go which is closest to the center of the robot. This is done by checking if there is enough free space (a certain number of vectors that are long enough) in front of the robot. The software does this by starting in the middle of the image and moving outwards to both the left and right side of the middle. If it doesn't find enough free space in front of itself it will know on which side the first object appears and will look to the other side for more free space.
After this part the software has found the middle of the gap closest to the middle of the image and steers the front wheels according to the distance between these two middles (with a certain factor of course).

4.4 End row detection

Similar to the navigation the end row detection searches for free space in front of the robot, the only differences are that it searches for a much wider area and it doesn't deviate from the middle. This means that the row has to end on both sides of the robot.

4.5 End row navigation

For challenge two it was also necessary to navigate at the end of the row and count the rows passed. For this navigation we used a similar region of interest as before, but now only the left or right half of it depending on the side of the field.
The way of determining the angle to steer is actually quite different from the standard navigation, this time we create an array of heights from the middle to the outside (horizontally instead of vertically) and determine the average distance of the first obstacle and then compare it to a fixed value. The deviations then directly relates to an angle to steer.

4.6 Weed detection

Finally for the third challenge (subtask 2) we used a second camera in front of the robot to detect the flower (representing weeds). For this we divided the camera image in to separate parts to make the difference between flowers left and right.
The image acquired gets thresholded, filtered with some basic morphology and then analyzed by a particle analysis module of LabView. When a particle matches the right color and size parameters it's identified as a flower and the software sends out a signal towards the robot.

5. Conclusion

We achieved more than expected, we came just to participate in challenge 1 and eventually we won it. So with our robot we became first in 2 challenges and eventually won the event, an unexpected achievement even by ourselves.

Our focus for next year will be in improving the wheels and software. The wheel have too much friction and the software is far from complete. The overall concept is good, but there is still a lot of improvement possible.

Cogito MART

Jiří Iša[1], Stanislav Petrásek[2]
[1]*Charles University in Prague, Department of Theoretical Computer Science and Mathematical Logic, Malostranské náměstí 25, 11800 Praha 1, Czech Republic*
[2]*Czech University of Life Sciences, Department of Agricultural Machines, Kamýcká 129, 16500 Praha 6, Czech Republic*

Abstract

In this paper, we describe a hardware and software implementation of the robot(s) used for the Field Robot Event 2010, their advantages and their drawbacks. A part of this paper is also dedicated to the team, which is a joint cooperation of Charles University in Prague and Czech University of Life Sciences.

Keywords: outdoor, robot, competition

1. Introduction

This paper describes a joint effort of Charles University in Prague and Czech University of Life Sciences – the Cogito MART team. Section 2 touches the inter-university cooperation. In Section 3, the hardware is described. Section 4 covers the software. We believe, that the future participants of the competition should be aware of problems we make public in Section 5.

2. Team

CHARLES UNIVERSITY PRAGUE
faculty of mathematics and physics

KZS
KATEDRA ZEMĚDĚLSKÝCH STROJŮ

In Field Robot Event 2009, Eduro Team from Czech Republic participated. The performance of the Eduro Explorer robot inspired researchers from Czech University of Life Sciences (CULS) to join the competition. This year, Eduro Team competes for CULS and a new team emerges in a cooperation of CULS and Charles University in Prague (CU): Cogito MART.
While CU is particularly strong in software engineering, CULS provides a much needed technical and agricultural expertise. To further support both teams (Cogito MART and Eduro Team), a testing ground has been seeded. This itself is a great improvement over the previous year, when the first testing opportunity of Eduro Team was at the competition ground.

3. Hardware

3.2. Platform

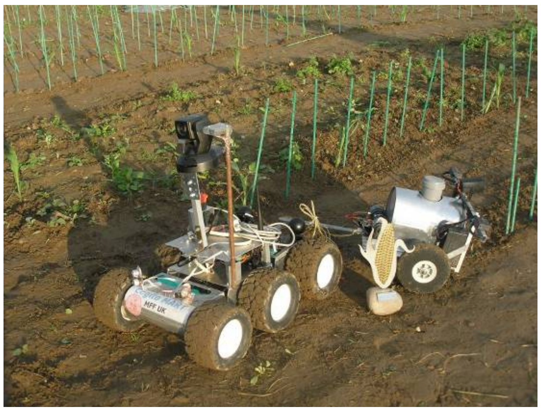


Figure 7: UTrooper platform from Wifobot with a weed extinction device attached.

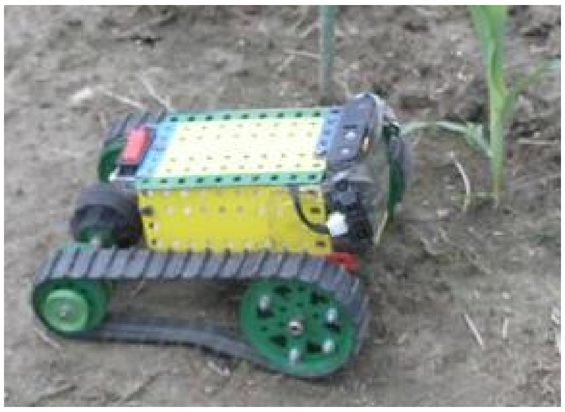


Figure 8: Fatima takes part in Freestyle.

Cogito MART competes with a commercially available Utrooper platform from Wifibot. Utrooper carries an on-board PC and a wi-fi router. Steering is provided with six DC motors. To minimize modifications of the platform and a risk induced by water in the weed extinction device (WED), the WED is designed as a standalone cart. This construction takes inspiration in real-world agricultural machines. The two wheels of the WED can be easily detached and the spraying device can be put on another robot with a transportation space.

For the Freestyle task, we have decided to use our robot Fatima. Fatima is an educational robot originally constructed for Eurobot 2005 competition. Fatima was built from Merkur (a Meccano-like construction set) and two common servo motors.

3.3. Sensors

A pan-tilt-zoom IP camera is mounted on the Utrooper platform by its manufacturer. In the end, we did not use it. For an obstacle (maize) avoidance we have added two ultra-sonic range finders (SRF08) to the robot. Further, two webcams are pointing towards the areas sprayed by the weed extinction device. A compass (CMPS03) measures rotation of the robot. Fatima uses two infra-red range sensors (Sharp 2Y0A21) to detect the plants.

4. Software

First, a straightforward maize avoidance using the sonars has been programed (Version 0 code)

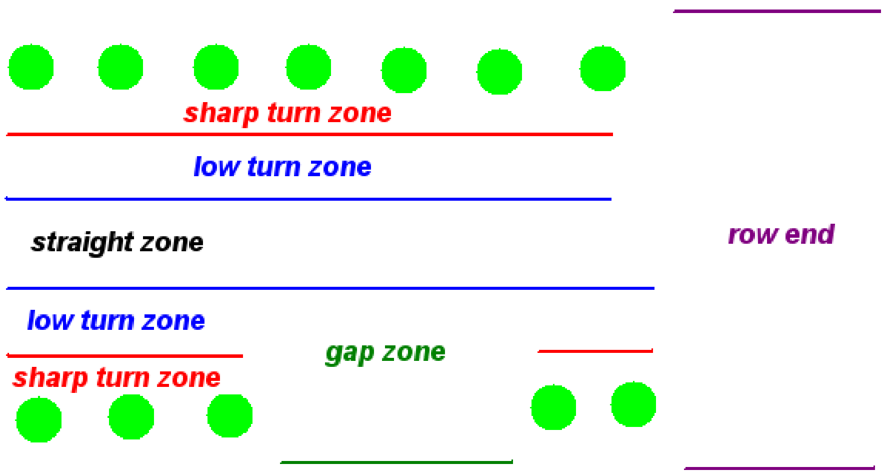


Figure 9: States identified by the Version 0 code.

The robot recognizes five important situations:

- Too close to a plant ("sharp turn"): In this case the robot slows down significantly and turns away from the plant almost on the spot.
- Going close to the maize ("low turn"): The robot slightly slows down and steers away from the maize.
- Being safe ("straight zone"): When sufficiently distanced from plants, the robot drives straight, full speed.
- Gap zone: When only one of the sonars detects the plants, a gap is recognized. The robot is steered to keep a predefined distance from the detectable side of the row.
- Row end: When both sonars measure a free space, a row end is detected and the robot turns as requested.

Next, a Version 1 code follows. It uses computer vision to detect a row and steers the robot in the "low turn" and "straight" zone.

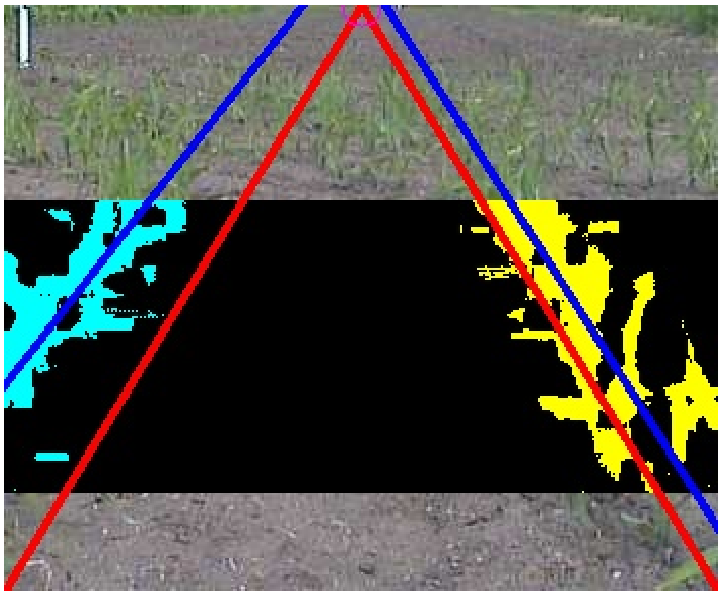


Figure 10: Red line shows an ideal row projection, blue line is a result of the vision-based row detection algorithm.

In the first step, green pixels are detected. In the second step, they are grouped into the “left” and “right” group. Using the Principal Component Analysis (PCA), major axes of both groups are computed. The intersection point of both lines is a vanishing point towards which the robot should steer.

5. Problems

5.2. Mother Nature

Because of an extraordinarily cold spring, the maize did not grow very well this year. Originally, we had a two-stage plan for our testing ground. The first part of the field was seeded a bit earlier to provide an opportunity for an early testing. A second seeding should have followed later to provide conditions similar to the competition ground. Because of the weather, the second part was omitted and the first plants grew just in time for the competition.

During the competition, the testing ground totally lacked plants (probably for the same reason). This made the last-minute testing nearly impossible.

5.3. Hardware

The Utrooper platform turns out to be a very problematic one:

- It has mechanical deficiencies: A fault in construction lead to a battery burn-out just before the competition. The wheel attachment is not strong enough to sustain the experienced mechanical forces, thus the wheels kept sliding of the axes during the FRE.
- The six-wheel platform turns out to be almost impossible to turn.

For the weed detection, two cheap Logitech webcams were used. As we have not tested them in advance, we did not realize that the captured images are saturated outdoors (the webcams are designed to work indoors). Luckily, there were heavy clouds during the Professional task.

5.4. Software

The software we have used evolved over the course of several robotic competitions this year. As such, it was well-tested and we are not aware of any problems we had during the FRE. Before FRE, we had to deal with non-deterministic garbage collection in Python and the pauses in run of the program it causes. Similar issues arise from the task planner in operating system.

Because of the lack of grown-up plants during the competition, we decided to use Version 0 software only. There was no way to test whether the vision-based algorithm works in actual conditions.

6. Conclusions

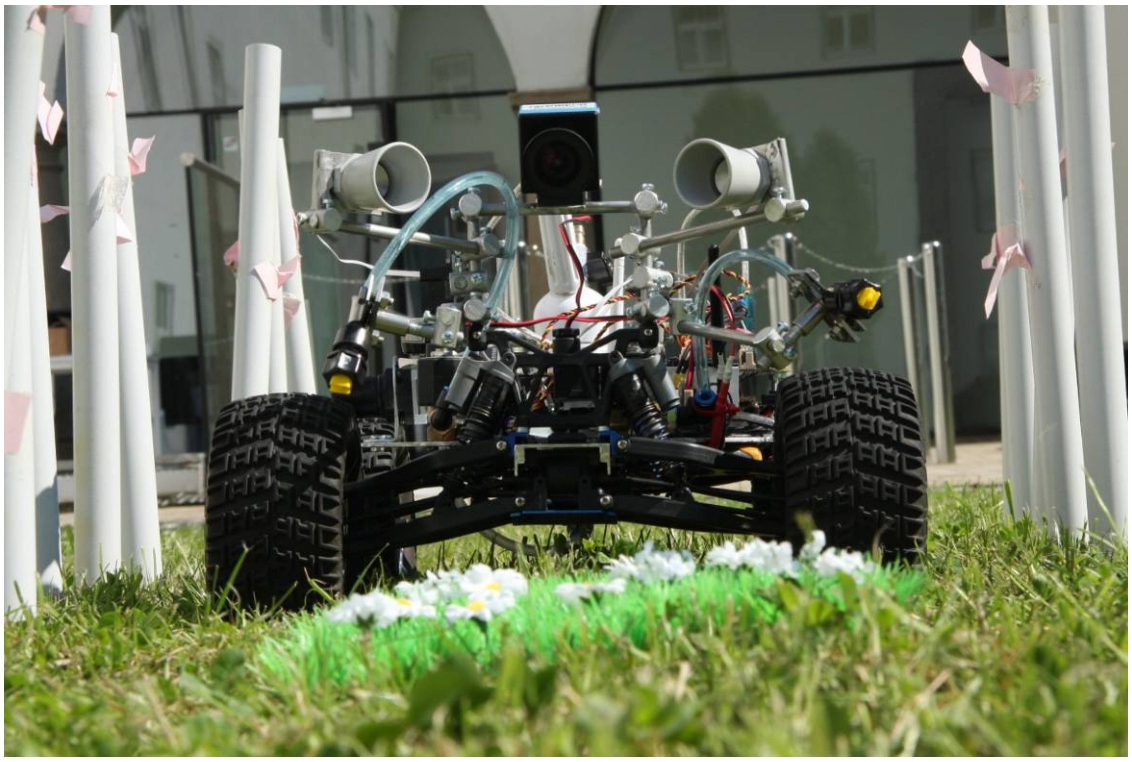
This paper introduced a joint coordination effort of Charles University in Prague and Czech University of Life Sciences – Cogito MART team. We have described both the hardware and the software we used for FRE. For the future participants, we also describe the problems we had and which should be taken into consideration.

Acknowledgments

This work was supported in part by the Grant Agency of the Czech Republic, grant number 202/10/0761.

CornStar

Peter Berk, Peter Lepej, Tomaž Paripović, Jurij Rakun, Miran Lakota
University of Maribor, Faculty of Agriculture and Life Sciences, Chair of Biosystems engineering, Pivola 10, 2311 Hoče, Slovenia



Abstract

We present a small automated self oriented robot, that is able to autonomously navigate on the corn field (trough the plant rows) and selectively apply pesticides. In order to achieve this goal we equipped the robot with ultrasonic sensors, a high resolution digital camera and an onboard embedded computer. The ultrasonic sensors are used to evaluate the distance from each corn row, enabling the robot to navigate through the maze, while the camera and embedded computer captures the video stream. The video stream is then transmitted via wireless IEEE 802.11n draft connection to an off-field workstation, that analyses it and responds with instructions to open or close one of the nozzles used to selectively apply pesticides.

Keywords: *field robot, embedded computer, digital image processing, colour segmentation*

1. Introduction

In the age of technological revolution agriculture is one of the disciplines that have a bright future. As big food producers rely on the use of heavy machinery, this is still not the case for mid- and small-sized farms, which can pose a potential food safety problem. If handled manually, food can transmit disease from person to person as well as serve as a growth medium for potentially harmful bacteria. Nevertheless, some work still demands manual labour that is time consuming, exhausting and expensive. The thought of introducing a small army of intelligent robots to do the job quicker and more accurate seems appealing, but we are not just there yet. For one, natural uncontrolled environment poses a challenge with its changing conditions. An overview on the subject showed that there are some potentially good solutions but the authors rely on specific conditions (like night time) or their solution is designed to work in controlled environments (green house) and some are simply too big or too heavy to be useful at this stage. In this paper we try to tackle the problem by introducing our own mobile agricultural platform.

In order to achieve our goal, we decided to put our efforts to build a small autonomous self oriented robot, that could for instance serve as a potential tool for selective pesticide spraying, fertilizer applicator or even as a device that could estimate the yield at the end of the harvest by simply taking digitalized snapshots of the tree canopies. In the following section we start by describing our solution in detail.

2. Description

The presented robot consists of four crucial components. The first is an onboard embedded computer with high speed digital camera and a wireless connection. The second is a four-wheel drive that enables the robot to move around on the rough terrain. The third are the ultrasonic sensors that help to keep track of the surroundings. And finally, the fourth part, the onboard reservoir and nozzles that spray the plants. Fig. 1 depicts the robot platform in detail along with all the components and their location.

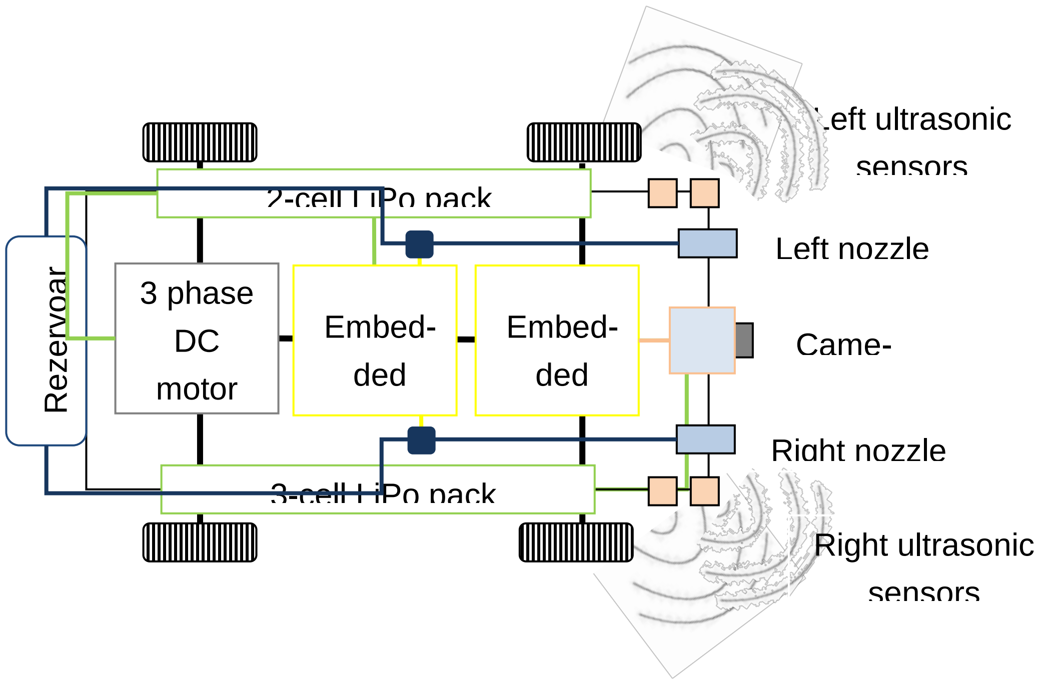


Fig. 1: Robot component layout.

2.1 Mechanical part

The base of our robot consists of a modified RC-Monster truck model. The base allows both axles to be turned individually by servo motors.

We made a CAD model with Catia software, shown in Fig. 2, and designed a new platform. This was done so additional parts could be designed more accurately. Last year's chain sprocket and chain were replaced by a belt and pulley system. This allowed a much smoother and more silent operation.

By having a digital model of the robot it was possible to optimize space consumption. Most parts were made of aluminium due to weight reduction. Unfortunately lack of time caused the chassis to be left without a body and the electronics sensitive to weather conditions.

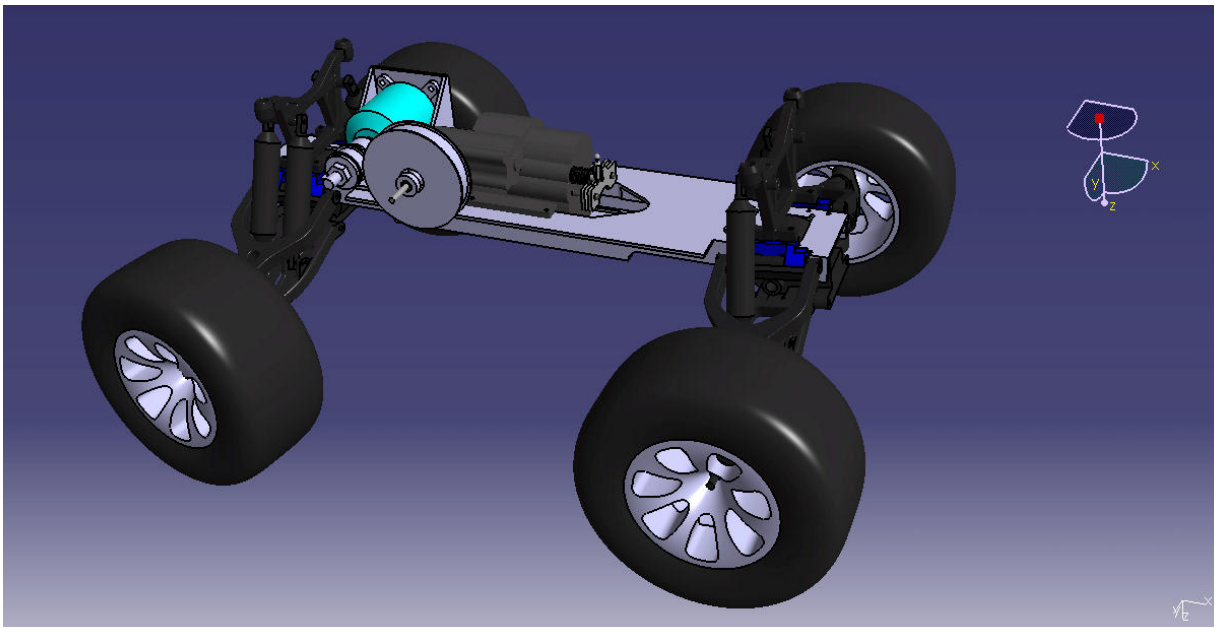


Fig. 2: CAD model of the chassis.

2.2 Steering

In order to drive the mobile platform, we have chosen a high-performance brushless motor (X-power eco A4130-06BL). Technical data for the selected brushless motor can be observed in Table 1. This is a three-phase motor so in order to use it, we have added a controller (x-power professional 70-3P BEC), which controls the coils of the motor. The specifics of the controller are depicted in Table 2. The controller automatically detects the number and type of batteries used. In our case we used Lithium batteries (LiPolice 5000mAh/2S 7,4V 75/100A) with two cells. The controller also controls the speed and rotation direction of the motor. Brushless motor speed is controlled with the help of pulse width modulation. At given (high enough) modulation the motor will run at constant speed consuming around 40 watts of power.

Table 1: Brushless motor - Technical data.

Cells Li-XX:	2 – 6
U/min/V (without gear):	510
Lenght (without shaft):	65 mm
Weight:	400 g

Table 2: Controller – Technical data.

Cell number Li:	2 – 3
Current, conti-nuous/burst:	70/85A
Dimension:	75x28x10 mm
Weight with cable:	54 g

2.3 Embedded circuitry

The system contains an 8-bit AVR microcontroller whose task is it to take sensor measurements and communicate with a remote computer and the onboard BeagleBoard (an onboard embedded computer). A low cost AtMega324p was chosen to do the job. It can do all the A/D conversion we need for ultrasonic sensor readings and has two separate USART ports. One of the ports is connected to the Beagle-board and allows the microcontroller to receive information about visually detected weed plants. The other port is used to communicate with the XBee wireless module.

The electronics board consists of the AVR microcontroller, Xbee wireless module and the peripherals needed for motor and sensor control. The microcontroller is driven by a 6.5536 MHz crystal, which is used to allow all of the PWM outputs to be used for motor control.

In order to navigate, the robot relies on the distance measurements captured by using ultrasonic sensors. In comparison to the last year's model of the robot, the SRF04 ultrasonic sensors were replaced by Maxbotix LV-WR1 that are waterproof but have a smaller detection cone. That is why four sensors were used and placed in the front to reduce the chance of missing a plant.

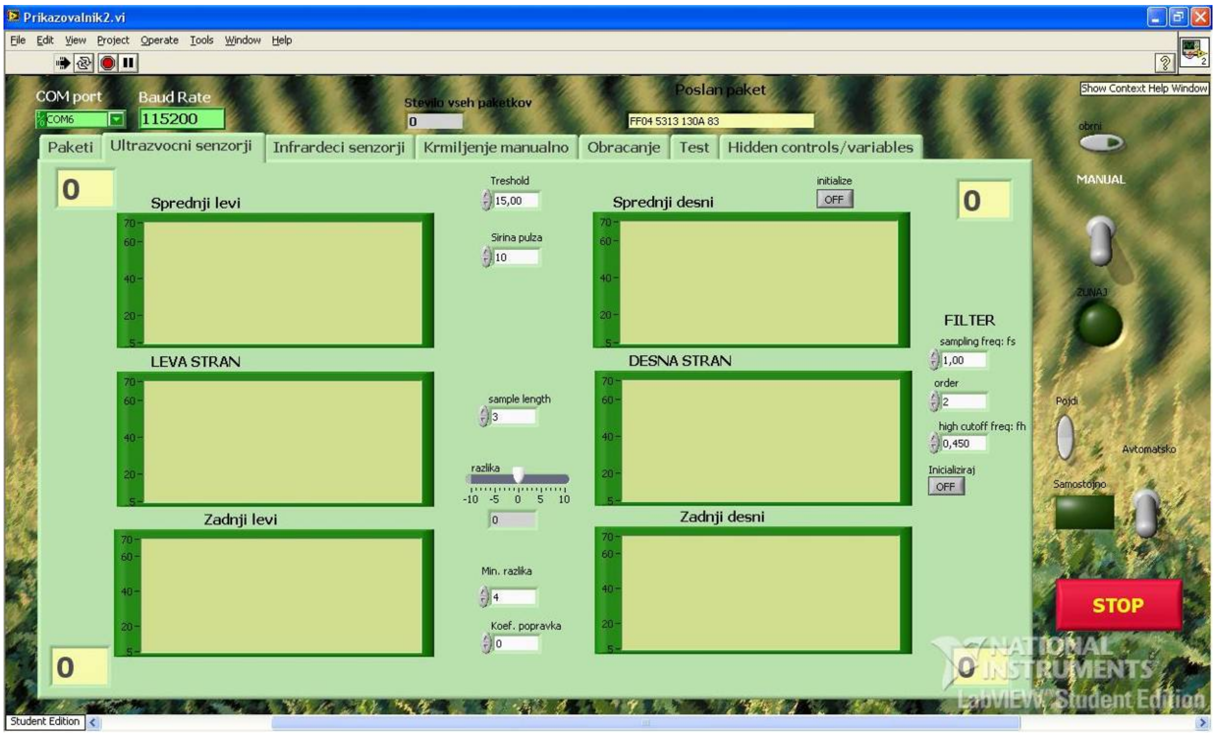
For orientation detection a compass module was integrated. Unfortunately it turned out the readings were influenced by the surrounding electronics and the gearbox. The output was therefore unusable and the sensor was left unconnected.

For the purpose of row counting two Sharp GP2Y0A02YK0F infrared proximity sensors were also placed on the sides of the robot.

2.4 Navigation

The basic task of embedded circuitry is to gather all relevant data from onboard sensors and transmit them to off-field workstation using an XBee wireless module. There the data is processed and a decision is made on how to control the platform (which direction to take, when the open the nozzles, speed adjustment, etc.). Of course most of the decision making algorithms could run on the embedded computer, but are still located on the off-field workstation. The purpose behind this scenario is easy and rapid development of new algorithms as well as almost instant control of the platform in cases of emergency.

In order to communicate a communication protocol was agreed upon. It consists of multiple 8-bit data bytes including start byte, ID bytes, data bytes and checksum. All of the transferred data is checked and any faulty packets of data are not processed. Counting the false packets also allows communication stability checking, error counting and calculating the speed of data processing.

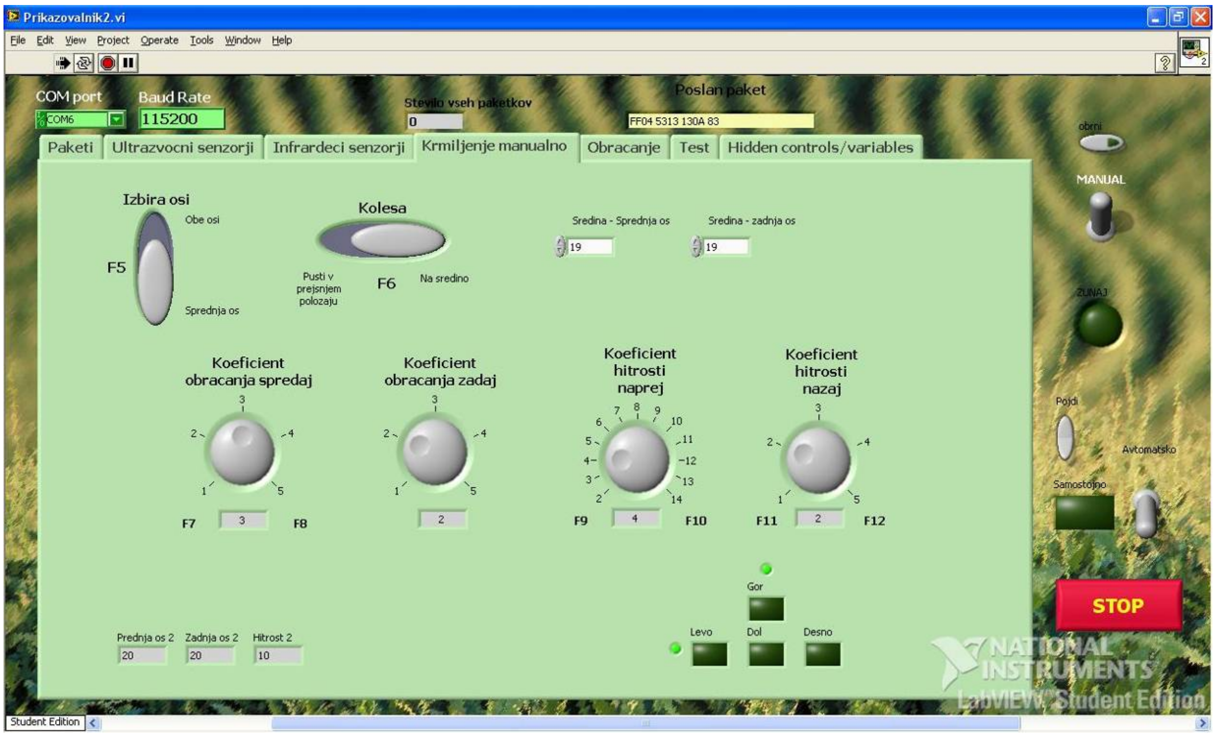


COM port: COM6
Baud Rate: 115200
Stevilo vseh paketkov: 0
Poslan paket: FF04 5313 130A 83
Treshold: 15,00
Sirina pulza: 10
initialize: OFF
sample length: 3
razlika: 0
Min. razlika: 4
Koef. popravka: 0
sampling freq: fs: 1,00
order: 2
high cutoff freq: fh: 0,450
Inicializiraj: OFF

Figure 3: Graphical user interface for sensory data display.

The data from the ultrasonic sensors is filtered using a low pass filter in order to reduce the number of false readings to a minimum. These occur as a sharp changes caused by false interpretations (less distinct areas of the natural scenes). All of the filter attributes are accessible from the control panel of the program and were set by empirical findings.

The measurements can be observed on the front panel of the program as shown in Fig. 3. The algorithm for steering between the rows is very simple and based on the difference measured on the left and right side of the robot. All parameters used in the algorithm are accessed on the front panel, including driving speed and turning of both axles (Fig. 4). This allows easy adjustments on the field.



COM port: COM6
Baud Rate: 115200
Stevilo vseh paketkov: 0
Poslan paket: FF04 5313 130A 83
Sredina - Sprednja os: 19
Sredina - zadnja os: 19
Koeficient obracanja spredaj: 3
Koeficient obracanja zadaj: 2
Koeficient hitrosti naprej: 4
Koeficient hitrosti nazaj: 2
Prednja os 2: 20
Zadnja os 2: 20
Hitrost 2: 10

Figure 4: Graphical user interface – a main control panel.

Manual driving for testing purposes is also supported by the program, as well as display of the packets sent by the microcontroller; useful for debugging purposes.

2.5 Image acquisition

The onboard embedded image acquisition part consists of three main components; a high resolution digital camera that captures video stream, a wireless interface that works according to the IEEE 802.11g standard and the final, third component, an embedded computer for which we selected the BeagleBoard ver. C4. This is a superscalar ARM Cortex A8 based embedded computer that consumes only around 2 W of power an offers 1200 Dhrystone MIPS performance. The embedded computer is depicted on Fig. 5.

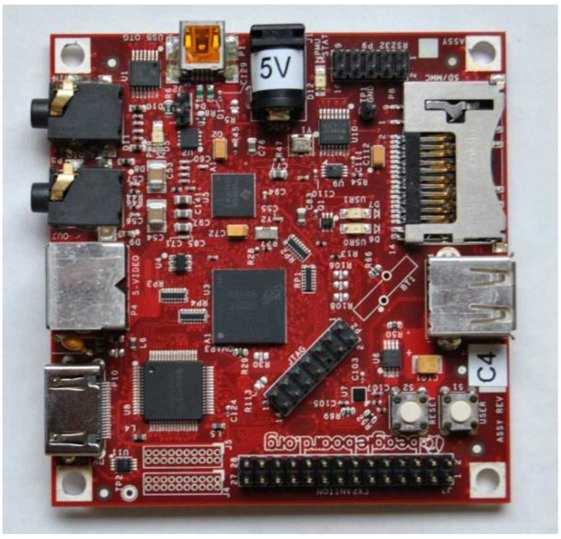


Fig. 5: The BeagleBoard – an embedded computer build around OMAP3530 processor.

The onboard embedded computer and the microcontroller are connected via serial (RS232) interface at a speed of 2400 bps, while the wireless connection operates at 150 Mbps to provide off-field workstation with a video stream, which demands a wideband wireless connection.

For embedded computer operating system we have chosen and customized a version of the Linux operating system - an Angstrom distribution. The first step we took was to install only the necessary software so the final version fitted on 1GB SD memory card. As the second step, we compiled a custom 2.6.32 kernel according to the hardware specifications and finally, as the third step, we tuned the settings for the TCP/UDP protocols to reserve more buffer space and to transmit small data chunks faster in effect minimizing the transmission delays.

2.6 Server algorithm

In comparison to the last year's model, the embedded computer is now used only for video acquisition and does not act as a mediator between the microcontroller and the off-field workstation. The purpose behind this decision was to minimize the communication lag that was in part caused by the transition of the video stream. Steering data is therefore transmitted on a separate communication channel, as described in subsection 2.4.

The main role of custom written server program running on embedded computer is to provide the off-field computer with video stream. It is able to capture live stream at the rate of 30 images per second with resolution of up to 1024 x 768 pixels and send it to off-field workstation in Bayer encoded format using different resolutions.

In order to be able to capture video stream and to transmit last captured image simultaneously two memory locations are reserved. The algorithm first chooses one of the locations and it stores new image. Then continues using the second memory location and stores another, newer image. This is repeated until the off-field workstation requests a new image. In this case, the location of last fully saved image is protected so it is not overwritten by the video stream capturing thread unit it is not fully transmitted. When the server thread locks the memory location, video capture thread stores new images only on one location that is available for writing. This way we provide up-to date data and also prevent the transmission of incomplete images. The mechanism for capturing and transmitting images is depicted on Fig. 6.

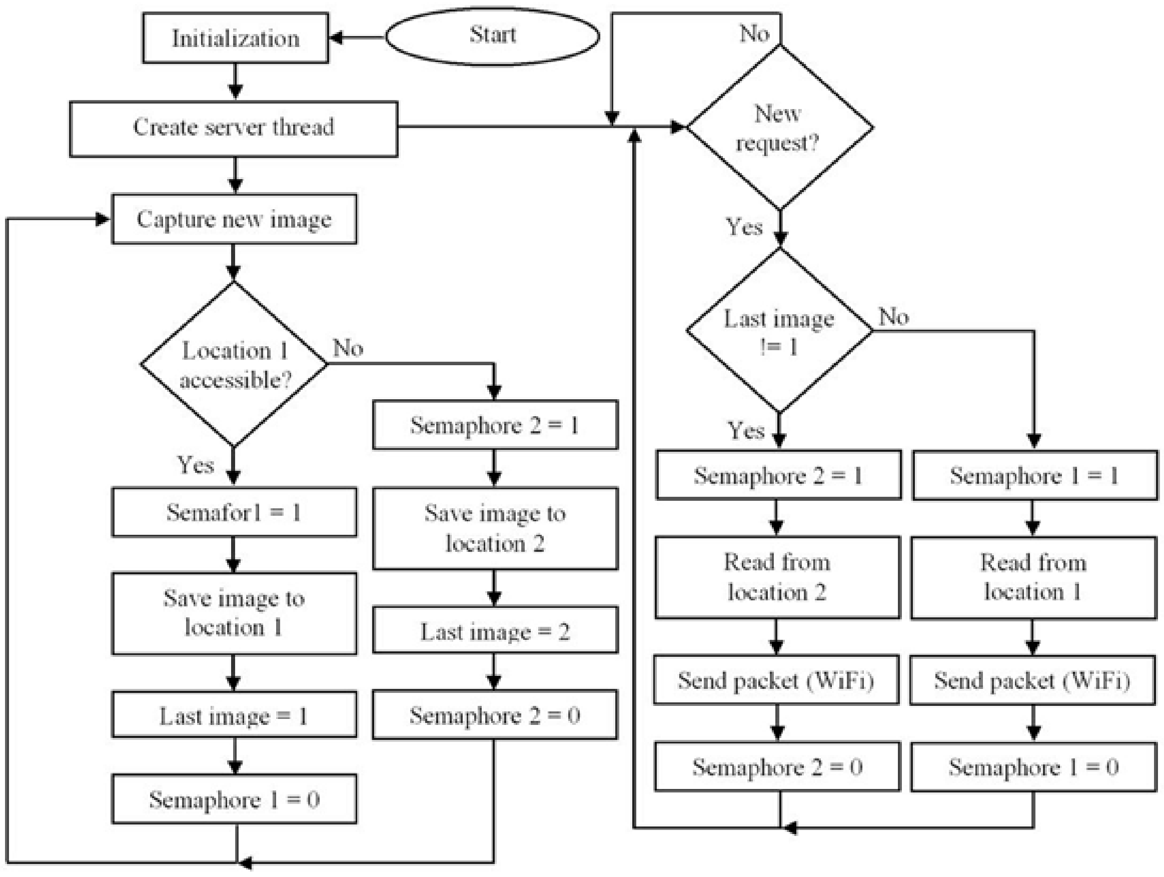
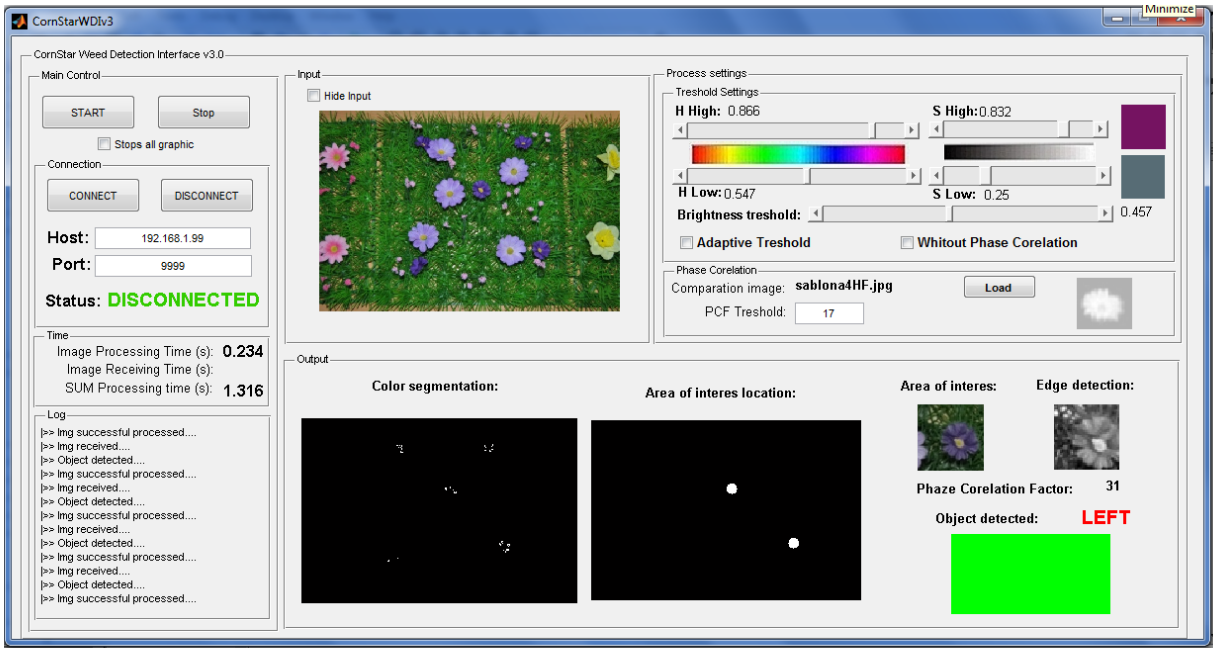


Figure 6: Server program - flow chart.

2.7 Image processing

In contrast to the previous version of the robot presented last year, we improved the object detection algorithm. Instead of focusing on one distinctive shade of colour (e.g., yellow for yellow flowers), we implemented an algorithm that can detect variable colour shades that represent different plants we are observing on snapshots taken with the help of a digital camera. So we created one universal graphical user interface for object detection that consists of 5 main areas: main control, time and log, input, processing settings and output. An example of the graphical user interface is depicted by Fig. 7 and is configured to search for blue flowers, with phase correlation factor threshold set to 17.



Host: 192.168.1.99
Port: 9999
Status: DISCONNECTED

Image Processing Time (s): 0.234
Image Receiving Time (s):
SUM Processing time (s): 1.316

```
|>> Img successful processed....
|>> Img received....
|>> Object detected....
|>> Img successful processed....
|>> Img received....
|>> Object detected....
|>> Img successful processed....
|>> Img received....
|>> Object detected....
|>> Img successful processed....
|>> Img received....
|>> Object detected....
|>> Img successful processed....
```

H High: 0.866
S High: 0.832
H Low: 0.547
S Low: 0.25
Brightness treshold: 0.457
Adaptive Treshold
Whitout Phase Corelation

Comparation image: sablona4HF.jpg
PCF Treshold: 17

Phaze Corelation Factor: 31
Object detected: LEFT

Figure 7: Image processing graphical user interface.

On the main control subsection of the interface window one can find basic commands to start and stop image processing. It also offers a possibility to enable or disable original or binarized representation of each video frame, which is suitable for debugging purposes. As it turned out, the step of refreshing all sub windows takes quite a lot of processing time (up to 0.4 sec), time that can be more wisely spent for image processing. The connection subsection can be used to connect to different hosts (by setting different IP addresses) and includes an indicator for the connection status. The time and log subsections are included to help keep track of processing status and accrued events. The input subsection shows an input image that was received from the onboard embedded computer, captured by onboard digital camera. The most important subsection is the process settings subsection. It starts with colour segmentation that is the key for further processing. By applying the HSL component threshold (e.g., setting the right threshold for hue, saturation and intensity values), we select areas that could represent objects we are trying to detect. Of course objects of the right colour may not be the objects we are trying to detect, so further processing is necessary. In order to prove or disprove detected areas, we use a template matching procedure for which we selected phase correlation approach. Input images used in later on phase correlation are first filtered using a Homomorphic filter that may eliminate any uneven lighting conditions. When using phase correlation step a good representative template must first be loaded and response threshold set. The output subsection depicts two large areas that show binarized images (left being the colour segmented version of an input images and the right its morphologically enhanced version), two partial images show a part of the scene that is of the right colour (original partial image on the right and its Homomorphic version on the left) and finally the result of phase correlation. If it produces high enough response (more or equal to response threshold) the area (i.e., the flower) in the image is identified as the one on the template image. Further on, the centre of flower is then calculated and a decision is made on which side the flower is located, a data needed to open the right nozzle.

The image processing algorithm that detects object of interest is summarized on Fig. 8 and consist of HSL tresholding, Homomorphic filtering, 2D median filtering, image dilating, edge detection and phase correlation functions. Combined, they help to detect objects that have the right colour and are of the right shapes to represents object we are trying to detect.

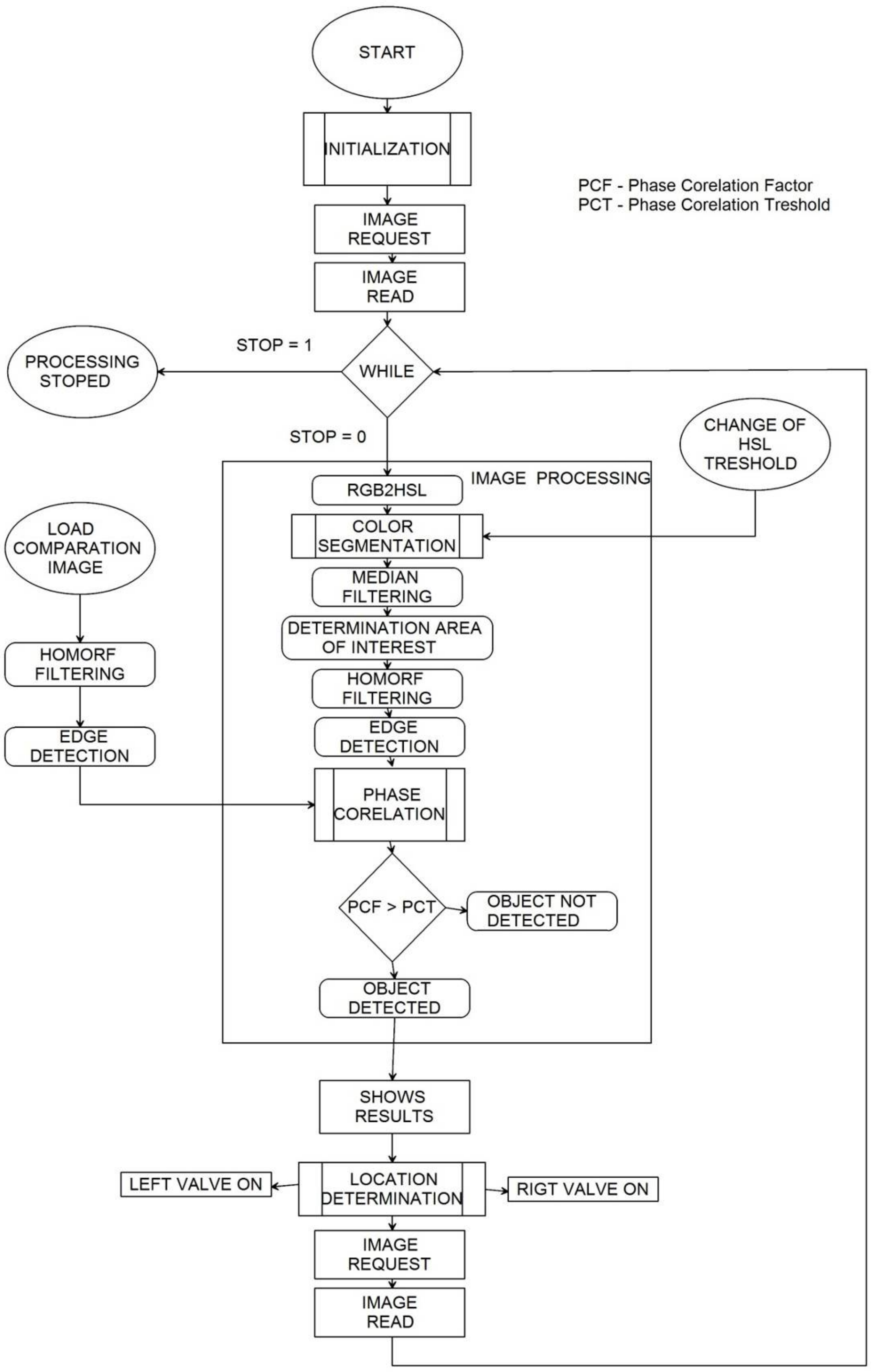


Figure 8: Image processing flow chart.

3. Conclusion

With the second generation of our Cornstar field robot we are a step closer to a prototype robot that could serve as a small agricultural tool for various tasks. Of course this is only the first step toward building a universal tool that could do the everyday work quicker, more precise and without human interaction. We are satisfied with the current results, but still have a way to go.

The main problem we faced this year were the distance measurements we got from ultra-sonic sensors used on the robot. All tests were conducted using real corn plants, but were forced to detect narrow sticks at the Fieldrobot event that were in most cases missed by the sensors or produced doubtful readings. Furthermore our robot was without the body and was left unprotected to the natural factors, a thing we regretted during the event when it rained cats and dogs and soaked our circuit board.

Along with building a new body to protect the robot and to try different distance (ultrasonic or laser) sensors for the next year's event, we can of course also focus on the developed algorithms. As we improved our algorithms for object detection based on colour and shape analysis, our goal for next year will be to transfer them to an onboard embedded computer, in effect eliminating the lag that occurs due to transmission delay, and increase the robot performance.

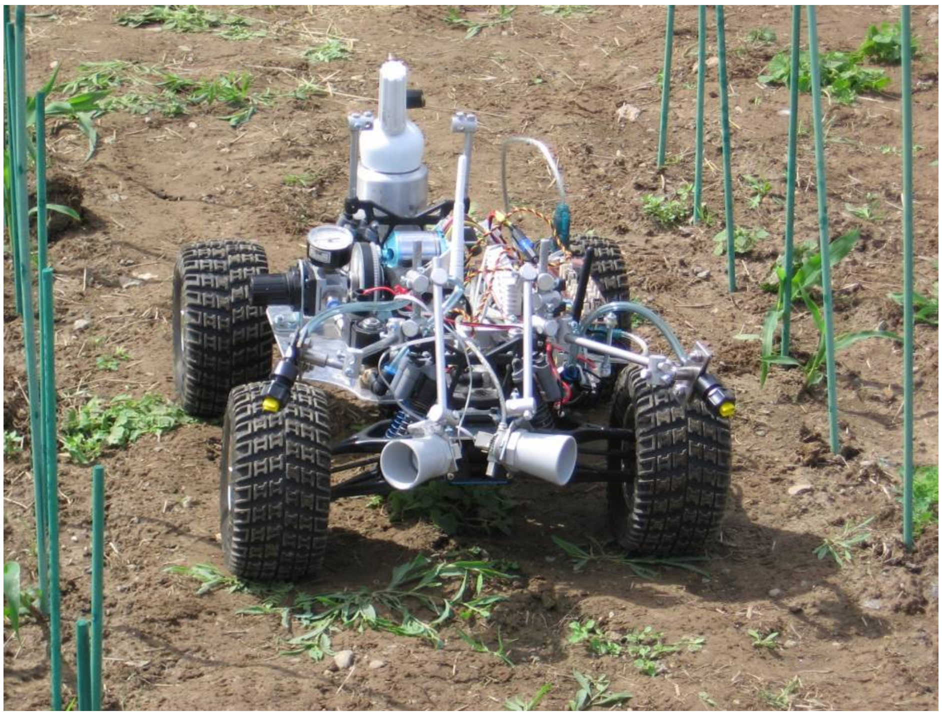


Figure 8: The Cornstar field robot at the Fieldrobot even 2010.

Acknowledgments

A special thank you goes to our sponsor ***The imaging source GmbH*** who provided us with a free camera we used on our robot.
The second acknowledgement goes to the ***Javni sklad Republike Slovenije za razvoj kadrov in štipendiranje*** that funded us to attend the Fieldrobot even 2010 in Brownschweig, Germany.
And the final thank you goes to the organization committee of the Fieldrobot 2010 for organizing this great event.

More information about the robot:

Email: *miran.lakota@uni-mb.si*
Website: *www.fk.uni-mb.si*

References

R. C. Gonzales, R. E. Woods, 2001. *Digital Image Processing*, 3rd edition, Upper Saddle River: Prentice Hall PTR.

J. F. Thompson, J. V. Stafford, P. C. H. Miller, *Potential for automatic weed detection and selective herbicide application*, Crop Protection, vol. 10, pp. 254-259, 1991.

L. Shapiro, G. Stockman, *Computer Vision*, Prentice-Hall, Inc. 2001.

Bulanon, D.M., T. Kataoka, Y. Ota, and T. Hiroma. *A Machine Vision System for the Apple Harvesting Robot*, Agricultural Engineering International: the CIGR Journal of Scientific Research and Development. Manuscript PM 01 006. Vol. III.

Guo Feng, Cao Qixin, Nagata Masateru, *Fruit Detachment and Classification Method for Strawberry Harvesting Robot*, International Journal of Advanced Robotic Systems, vol. 5, no. 1, 2008.

EasyWheels 2010

Timo Oksanen, Jari Kostamo
Aalto University, School of Science and Technology, Helsinki/Espoo, Finland.
Otaniementie 17, FIN-00076 Aalto, Finland. email: timo.oksanen@tkk.fi

Abstract

EasyWheels 2010 is an improved version of EasyWheels robot that took part into Field Robot Event 2009 in the Netherlands. EasyWheels 2010 robot is capable of driving between corn rows, doing the turns and capable of following another robot in constant distance. The robot is pretty symmetric and it is able to drive the rows in both directions, and therefore the turnings to a next row are quick to do. The robot uses ultrasonic and infrared rangers in each corner of the robot, turning camera head on top of the mast and tilt-compensated compass. The algorithms estimate position in a row using both ultrasonic and infrared rangers, backwards odometry and machine vision together. In Field Robot Event 2010 EasyWheels 2010 took part into the basic tasks, but also with Helios robot in two robot co-operative challenge. In two robots co-operative task, Helios was driving in front and "shooting harvest backwards" and EasyWheels 2010 was equipped with a hopper that collected the harvest – this demonstration was awarded with the first prize.

Keywords: *field robots, agriculture, co-operative robotics, field robot event*

1. Introduction

EasyWheels robot participated in Field Robot Event 2009 in Wageningen, the Netherlands. The robot was built by students from Helsinki University of Technology (nowadays part of new Aalto University) and from University of Helsinki. The robot won the second prize in the event 2009.

However, the robot suffered some technological problems, like bad turning servo for camera head and painful problems in programming C both in Windows and Windows CE cross-connection. The latter was considered to be too bad experience to be given any further and therefore some crucial change was needed, to give a bit better (programming) tool chain for next student team (that was to be Turtle Beetle). The problems in EasyWheels were reported in the Proceedings of 7th Field Robot Event 2009.

Pretty soon after 2009 competition we as teachers wanted to investigate possible fixes for the tool chain, as well as some other technological drawbacks, like DC motor drivers. The underlying tool chain both in EasyWheels and later on, has been Matlab + Simulink + code generation + Visual Studio + Windows CE, with remote debugging. To find proper fixes for the tool chain, we wanted to test it first with EasyWheels, by starting from the scratch (in software) and developing most of that with C#, not with C/C++. Only Simulink generated code requires some wrapping. This technology was utilized in Windows in “Wheels of Corntune” (Maksimow et al 2007) and victorious “Mean Maize Maze Machine – 4M” (Backman et al 2008). However, this was not possible to do in the same way in Windows CE Embedded. Also real-time capabilities of C# over Windows CE was a bit mystery. The main motivation was to *learn* how Windows CE + Simulink + CF.NET (C#) can be used together and to *teach* it for the student team.

So EasyWheels 2010 was born, and after discussions with FREDT / Helios team, we ended up developing a two-robot demonstration for Field Robot Event 2010 special co-operative challenge.

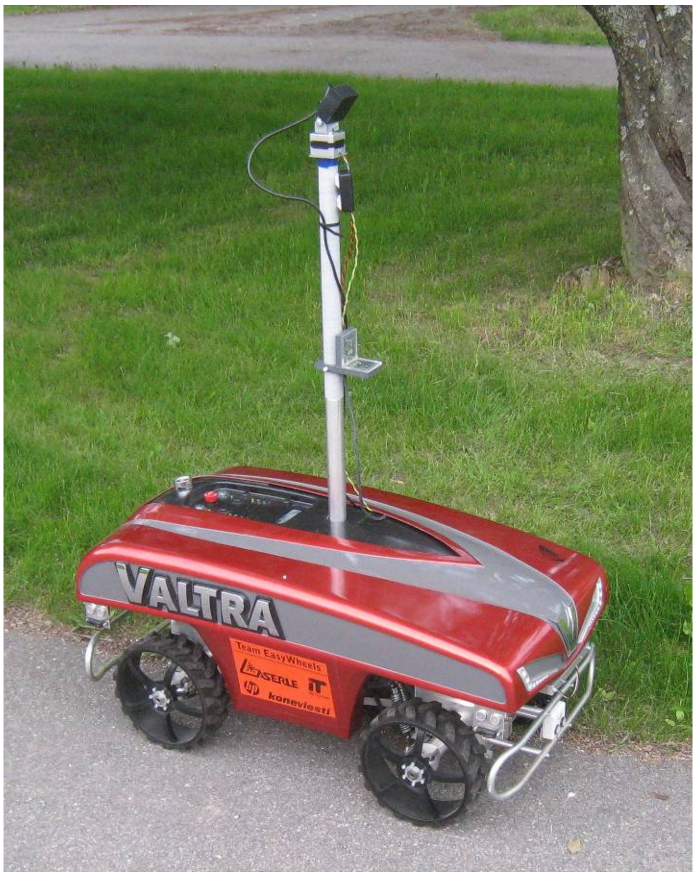


Figure 8: EasyWheels 2010

2. Mechanics and Mechatronics

2.1 Axle Module

For locomotion EasyWheels is equipped with two identical “axle modules” that encapsulate DC motor, reduction gears, differential, optical encoder in motor shaft, steering servo, feedback steering sensor (potentiometer), DC motor driver and microcontroller that does lower level servo control and offers a simple control interface for computer with RS-232 serial connection. Beside serial port the axle module requires only 12V power supply. Three identical axle modules were constructed for EasyWheels. These modules have proven out to be reliable, as only the DC motors have been changed to more powerful after last event. No further fixes were needed. (Kemppainen et al 2009).

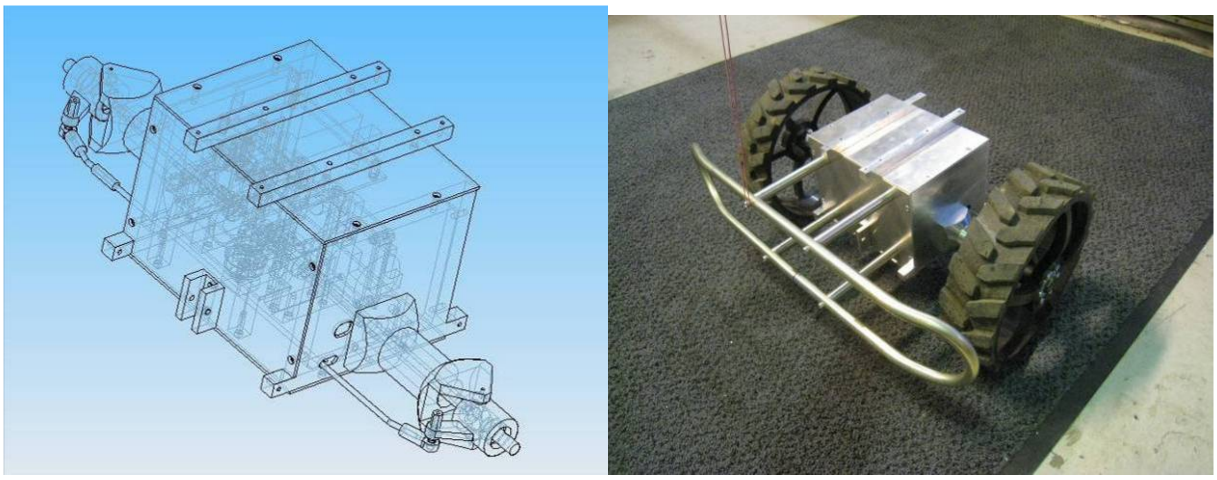


Figure 9: Axle module (Kemppainen et al 2009)

2.2 Suspension

In EasyWheels suspension combined spring-damper with force balancing mechanism was used. The similar system was used in Mean Maize Maze Machine - 4M (Backman et al 2008), and in EasyWheels a bit more compact way. The suspension works very well in driving uneven terrain, but still the problem is to get enough roll stability to frame. This is still the problem in the chassis, as the upper body tends to roll on turns.

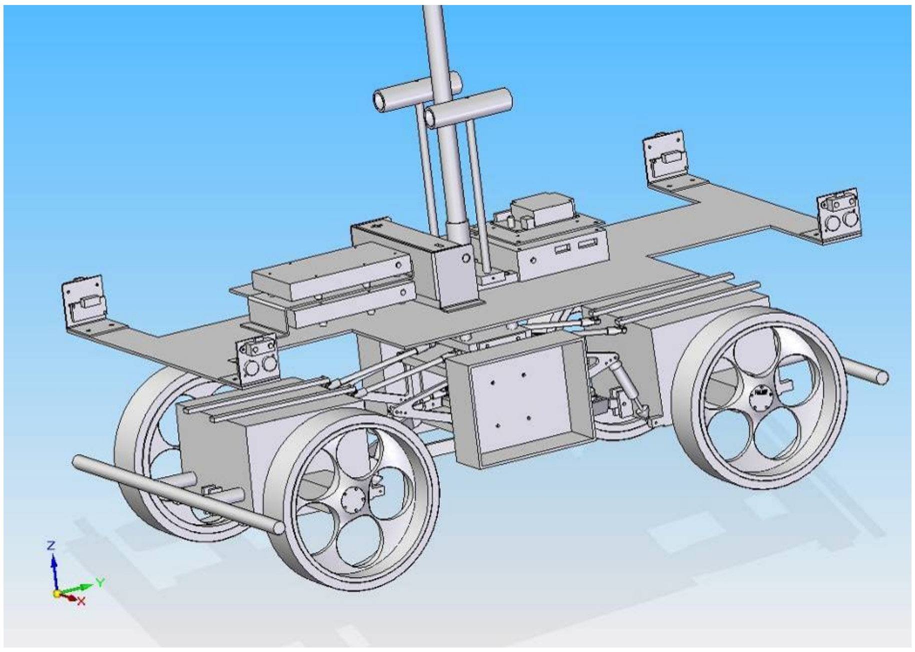


Figure 10:Suspension (Kemppainen et al 2009)

2.3 Camera head

One of the problems in original EasyWheels was the camera turning head. 360 degree operation was required for usage in two-way driving and headland operation. RC car servo with 360 degree rotation was selected. One of a few models capable of turning 360 degrees is GWS S125. However, the repeatability of this servo was rather bad (5 degree error) and therefore ordinary 180 degree servo was used in the competition 2009. (Kemppainen et al 2009)

For refurbished EasyWheels 2010 the RC servo was replaced with a stepper motor. A stepper motor is a bit heavier, but for this turning purpose good enough solution as there is no other counterforce than camera cables and vibration. A SparkFun stepper motor was selected for the purpose. The key numbers of the selected stepper motor are: step angle 1.8 degrees, 2 phase, rated voltage 12V, rated current 0.33A, holding torque 2.3kg*cm. The stepper motor was installed on top of mast, in the place where servo used to be, and the new parts to connect the camera to motor were made with milling machine.

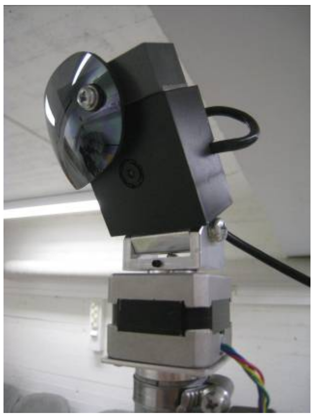


Figure 11:On the left: the stepper motor. On the right: the camera head with re-encapsulated Logitech 5000 with a sun glass.

3. Electrics and Electronics

3.1 System

The electric system of the robot has completely been refurbished. One of the reasons was that the original electric wiring system, grounding plates etc. was a mess and hard to maintain. All the wires were taken out, components repositioned, some components added (like beeper, more ultrasonic range sensor to front, turning servo for front ultras, stepper motor for camera head, LCD display to local user interface, some new user interface buttons...). Some new PCBs were milled to distribute power and to act as a star junction point for grounding (see Figure 12). I^{2}C is utilized more and more to connect devices to microcontroller: LCD is working over I^{2}C and all user interface LED's and buttons also through GPIO expanders.

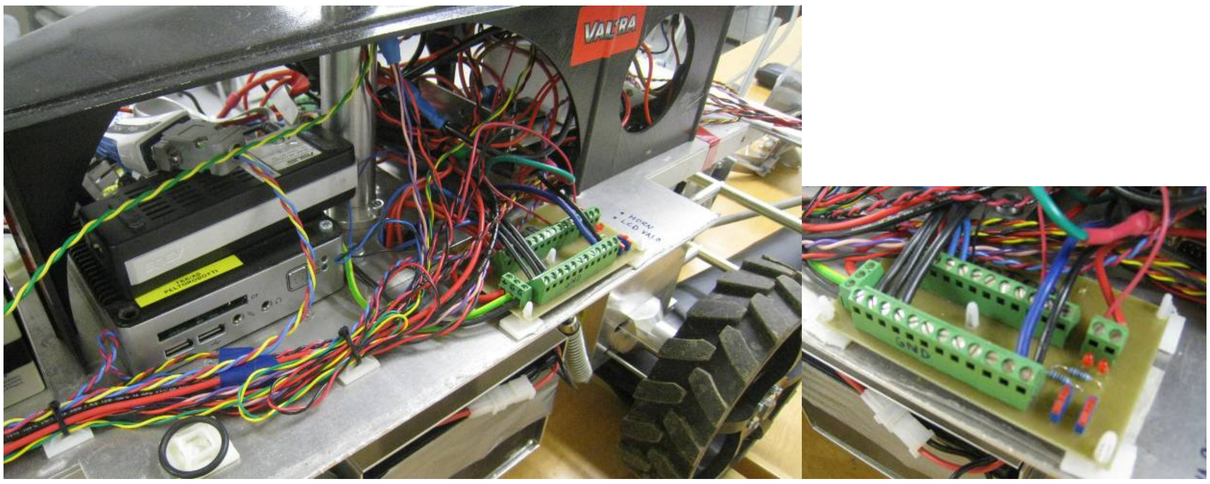


Figure 12:Wirings of the robot, one of the two junction boards on the right.

EasyWheels 2010 uses two parallel 12V lead acid battery systems to carry energy. One of the battery systems (2 times 6V/7Ah batteries in series) delivers power to DC motors and servos inside axle modules, and the other system (12V/2.1Ah + 12V/4.5Ah in parallel) supplies power to computers etc. The battery systems are split in two to eliminate risk of computer & controller reset during high current peaks when accelerating motors. The other battery was installed in parallel with an old 2.1Ah computer battery, as in FRE2009 the robot suffered battery problems. This resulted more weight, but was important for doing tests.

3.2 Microcontrollers

Futurlec Atmega128 prototype boards were used in axle modules to do servo control for drive and steering, and the third one was used in upper body to connect all the sensors, stepper motor, front servo and horn to the computer. Five SRF08 ultrasonic rangers, a SRF235 ranger, two CMPS3 compass modules, an LCD display (Batron LCD with Philips PCF2119 driver), and buttons and LEDs by using GPIO expander (Microchip MCP23008) in local user interface are all connected with I^{2}C bus (Figure 13).

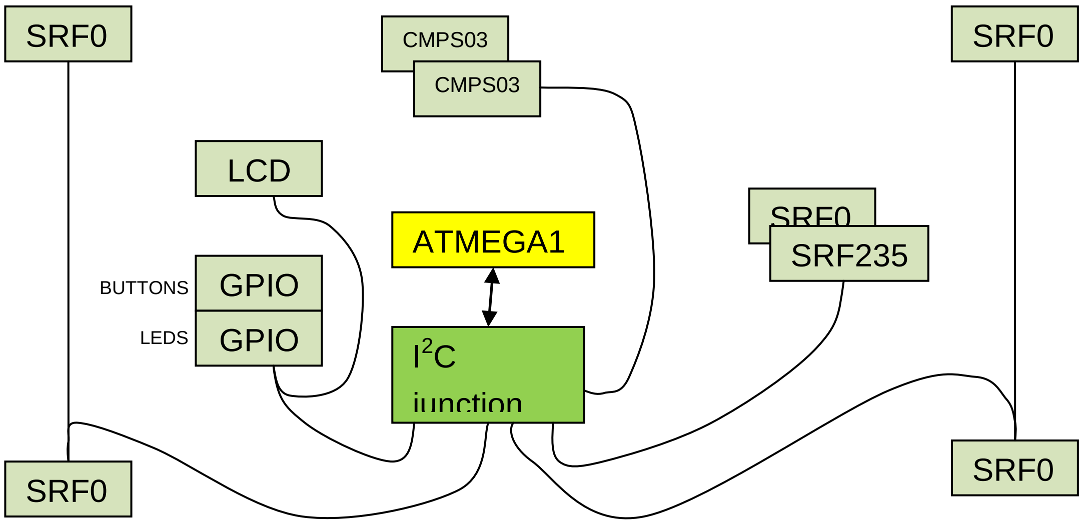


Figure 13:Devices connected with I^{2}C bus

3.3 Stepper motor control

The stepper motor could be controlled directly from the microcontroller by using only some transistors between them. However, specific controllers are available that provide benefits like current control and microstepping to reach better control accuracy. A SparkFun EasyDriver v4 was selected for this purpose, Figure 14. The key features are: A3967 microstepping driver chip, resolution from full steps to 1/8 steps, adjustable current control from 150mA/phase to 750mA/phase and power supply from 7V to 30V.

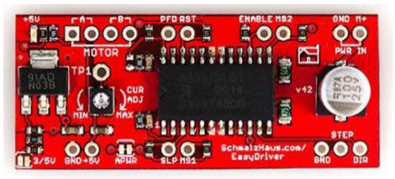


Figure 14:EasyDriver v4 – the stepper motor driver

The stepper motor control lacks feedback, so it is not possible to detect if the motor drops steps. In the power up sequence the motor is placed by hand to the zero position. This worked well in the short field test runs and also in the competition, but for longer working hours some feedback should be added.

3.4 Computers

The original EasyWheels was equipped with three Windows CE 6.0 running embedded computers, but in EasyWheels 2010 only two were used. The third one was used to detect weeds, and that was not objective this time. The other embedded computer is x86 miniPC and the other is XScale based.

EasyWheels 2010 uses embedded distributed computing instead of an on-board laptop. Earlier robots before EasyWheels couldn't reach strict real time, because of the non-real time operating system, Windows XP. EasyWheels' computers are running Microsoft Windows CE 6.0, which is a real time operating system. With a non-real time operating systems the problem usually was non-deterministic operating system's background processes that could freeze or interrupt critical navigation and machine vision algorithms. With Windows CE, this kind of behaviour should not happen. In addition to real time operating system EasyWheels' computers are embedded. The goal was to achieve a modular computing platform with strict real time capabilities. The drawback of Windows CE is lack of flexibility, for instance relating new hardware added to computer – the compilation of operating system works only with the hardware for which it has been compiled, and if some required drivers are not compiled in, it may require recompiling the whole operating system. However, in the robot, only some USB-RS232 adapters and specific camera are needed to connect besides normal hardware, so this is not a repeating problem, only when setting up the system or changing the hardware.

Based on tests, eBox-4300 is about 4 times quicker to compute floating points (single) than Colibri. However, on the other hand Colibri is about 4 times quicker than eBox when it comes to 32-bit integers. For this reason eBox-4300 is used in navigational computation with Simulink generated code and Colibri used to compute machine vision algorithms that are integer optimized.

3.4.1 ICOP eBox-4300

Figure 15:Ebox 4300 (EmbeddedPC.NET)

As a navigation computer eBox is very efficient. EBox uses conventional x86 PC hardware and therefore has a FPU. With VIA Eden ULV 500 MHz processor eBox can easily run complex navigation algorithms which can't be run on a Colibri. (Kemppainen et al 2009)

ICOP eBox-4300 requires an external 5V supply. The original 230V adapter is rated to 3A, even if the computer does not take more than 10W. To produce 5V @ 3A from 12V lead-acid battery system, a DC/DC converter was installed (Texas Instruments TPS5430). The DC/DC converter was shared also to WLAN access point.

3.4.2 Toradex Colibri

Colibri PXA320 has a very low power consumption with a great amount of computing power. Colibri has an ARM processor running at 806 MHz but it doesn't have a floating-point unit (FPU). It was selected for machine vision because of its computing power, the lack of a FPU was considered acceptable, as machine vision doesn't necessarily need floating-point computing as RGB images consist of 8bit integers. Naturally floating-point operations are possible, but through emulation they are very slow. (Kemppainen et al 2009)

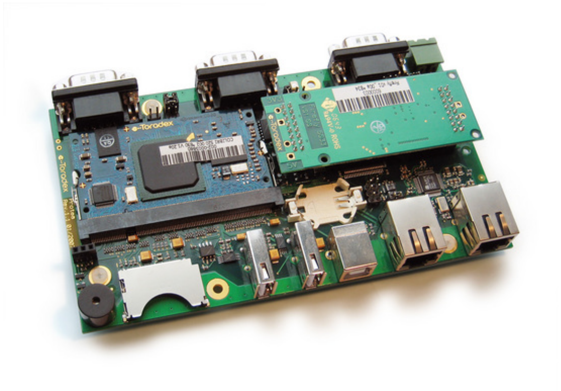


Figure 16:Toradex Protea carrier board with Colibri module (Toradex)

4. Sensors

4.1 Rangers

4.1.1 Ultrasonic

The ordinary sensors in low budget robots are ultrasonic rangers. In this robot two kind of ultrasonic sensors were used: 5 pieces of Devantech SRF08 which operate in 40kHz sound frequency and one Devantech SRF235 which has 235kHz frequency. Both of the sensors are used over I^2C bus. SRF08 is cheaper and has quite a wide beam, as opposite higher frequency produces narrow beam, but this sensor is more expensive. Four of the SRF08 were used in each corner of the robot, and the other two sensors were installed in front, in the middle of a bumper on top of a RC servo, and these sensors were used to measuring distance to the other robot running ahead. As it was not sure which of the sensors 40kHz or 235kHz is better for detecting the co-operative robot, both of them were installed.

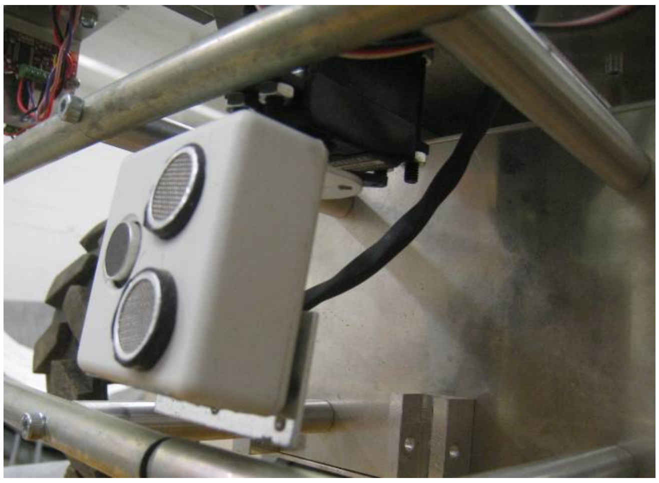


Figure 17:SRF08 and SRF235 installed in front of robot with servo head.

4.1.2 Infrared

Four Sharp GP2D120 sensors were used in each corner to measure the distance to the maize row. The noise in these sensors is higher compared to SRF08 when driving in a maize row, but by using sensor fusion, these sensors still have additional information. The claimed range of sensors is 4cm to 20cm, but the sensor works quite OK from 3cm to 30cm. Under 3cm the signal is similar than in valid range, so under 3cm measurements should be restricted mechanically. Over 30cm range the signal is decreasing outside of valid range and therefore usable, but the sensitivity is low.

4.2 Heading

The heading of the robot is estimated by using both magnetic field sensors and inclinometer. With an inclinometer the attitude (roll & pitch) are measured, and this information is used to a rotate magnetic field vector to horizontal level. The heading angle is computed from rotated vector x- and y- components by using arcus tangent.

4.2.1 Compass modules used as magnetic field sensor

Like in the original EasyWheels, two CMPS03 compass modules were used. However, the directly computed heading angle from the sensor is not used at all, but only the raw values that are readable from “internal test registers”, in 8-9 and 10-11, respectively. The registers contain raw measurements from the two Philips KMZ51 one axis magnetic field sensors. So practically CMPS03 was used only “as a demo board” for KMZ51 chips. From two CMPS03 modules together four raw measurements are recorded every 50ms.

4.2.2 Inclinometer

Two VTI SCA610 sensors (1-axis) were used to measuring roll and pitch. The maximum slope is 30 degrees. The inclinometer works well in static conditions, but not that well in dynamic movement as there is no gyroscope to stabilize the angle. However, an internal 2Hz filtering provides stable signal and with slow driving speed the signal is good enough for magnetic field attitude compensation purposes.

4.3 Camera

Logitech QuickCam 5000 was used. The camera was re-encapsulated to a new box already in the original EasyWheels. The camera has problems in outdoors, in summer sunshine the pixels may saturate, and the colours are twisted. To correct this, a sunglass added (see Figure 11). Actually, this was the other "sunglass" from the cheap sunglasses bought as a part for 4M (Backman et al 2008).

5. Software

5.1 The main tool chain

As described above, one of the main motivations was to learn a proper tool chain to connect Matlab, Simulink, Stateflow, Windows Embedded CE and .NET framework. Compared with EasyWheels (Kemppainen et al 2009), the main difference is to use C# to program runtime code for Windows Embedded CE. Runtime includes interoperation with all the sensors, doing communication with remote user interface, handling parameter storing and handling log files. .NET framework and C# provides more powerful tools for writing equally working code both for Windows Embedded and regular Windows. In C/C++ programming the biggest problem in cross-platform development are a number of very small hidden differences in bit level (like in WinSock) that causes a lot of headache. .NET framework resolves many of these problems. For mobile devices and Windows CE Microsoft provides a smaller version of ".NET framework" which is known as ".NET Compact Framework". To make a difference, the Windows version is hereafter called ".NET Full Framework". The basic namespaces, classes and methods in Compact Framework are just the same as in the Full Framework, but all the advanced features are not included. Also in some classes some methods may not be available. Generally it can be said, that if C# code works in Compact Framework, it will work also in Full Framework, and this has to be a way of development, to guarantee cross-platform usage.

The first problem was to find how to use C++ code and C# code together in .NET Compact Framework. C++ code comes from Simulink code generation (Real Time Workshop, Embedded Coder), and in order to use it efficiently with C# programming, a Multilanguage programming has to be used for integration. For Full framework this is straight forward as it supports so called "managed C++" code, run on .NET framework. This technology was utilized both in Wheels of Corntune (Maksimow et al 2007) and 4M (Backman et al 2008). However, Windows Embedded CE and .NET Compact Framework is not supporting running "managed C++", only regular C++ without .NET. Due this reason, for the original EasyWheels (Kemppainen et al 2009) the decision was to solely use C++ in programming – which was not a good solution as was concluded by Kemppainen et al (2009).

Luckily, a way to overcome the problem partially exists, it is known Platform Invoke (usually shortened as P/Invoke), which is a technology of .NET to make calls to native libraries (Windows internal libraries, or developed by a user). To make calls, a definition of library function has to be written in C# with some definite syntax, to make a connection. After definition, the function is called like any other function. Not to be too simple, .NET Compact Framework is a bit more limited in a sense of supporting the data types. Only simple basic data types (like integers and floating points) are supported, and fixes size structures (struct) – strings are not included for instance. Relating Simulink generated code, the biggest limitation is that (fixed size) arrays inside structures are not supported. This can be overcome by not using any other except scalar signals in Simulink model inputs and outputs (relevant only for C-interface). In other words, measurements from four ultrasonic sensors may not be defined as "US_cm[4]", but four variables "US_cm1", "US_cm2"..., for instance.

The other limitation in .NET Compact Framework that was found, was lack of binary serialization classes and methods. This was needed when sending data structures between remote user interface (laptop running Windows 7) and embedded computer (eBox). The serialization converts runtime presentation of data to common binary stream, that can be deserialized on the other end. Luckily, this was not the first time in the world this limitation was found, and a good solution for the problem exists: a library called "CompactFormatterPlus". This library does the same the .NET Full Framework would do, and more. A simple usage of the library for enough for this project.

So main cycle of developing runtime is presented in Figure 18.

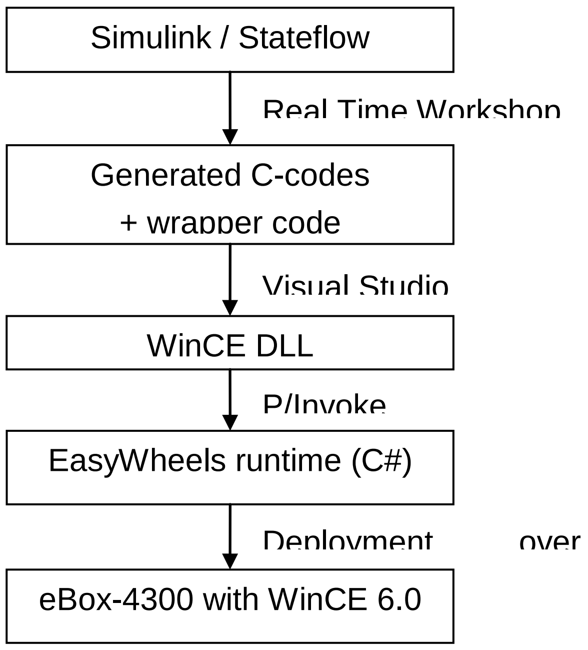


Figure 18:Main stream of software tool chain from Simulink to embedded computing.

One of the manual phases needed to take place is the definition of I/O Simulink interface to C# by hand. These structures in C# and C++ (in DLL) have to be exactly the same, in order to work. This is a pretty quick phase to do in case of changing the interfaces, but just have to be remembered. The objective was not to have anything like this manual phases, but no solution has been found so far.

Parameters of the Simulink model can be tuned from C# code, as long as in the code generation phase the parameters are marked to be tunable. If marked as tunable, the code generation places the variables in to a separate struct, where they are easily available for a programmer. In EasyWheels 2010 the tunable parameters are stored in XML files (by using .NET framework XML Serialization), which was used to store different parameter sets for different tasks, just like in 4M (Backman et al 2008).

One of the nicest features that help debugging, is that the tool chain also supports the remote debugging of code by using Simulink as interface. Simulink code generation supports “external mode”, which means that the code generated contains certain additional code with which the Simulink running on one computer can get and set data for the C-code running on the other computer. Simulink also offers stack over TCP/IP to do the job, so practically this does not need any custom code. However, again the support for Windows Embedded CE was not available, but it was possible to customize Simulink provided C-code stack a bit, to support also Windows CE builds (Winsock was the difference). The main usage of the external mode is to see the signal values (either by using numeric display or Simulink scope), and by watching in which state Stateflow programmed state machines are, it shows the active states by highlighting them.

The output of this learning process, the main tool chain, was used also in Turtle Beetle (Pentikäinen et al 2010). The technologies and skills required to use and develop the tool chain were educated for Turtle Beetle at the beginning of the project.

5.2 Microcontrollers

CodeVisionAVR was used to program microcontrollers. CodeVisionAVR is easy to use for beginners, as it helps user to select a proper register values with integrated wizard. For advanced user it does not make so much difference, but for beginners it saves time by not making it necessarily to go through datasheets and code examples.

5.3 Machine vision

The machine vision was programmed in Toradex Colibri, running Windows Embedded CE 6.0. The machine vision component incorporates OpenCV library, but actually only the image data formats are utilized, as almost all algorithms suffer lack of FPU. The vision algorithm was written with C++, and compiled to CE compatible native DLL. The runtime was programmed with C#, and it uses P/Invoke to call the DLL functions, to set parameters, and read the outputs. With C# it was much easier to handle network data.

6. Algorithms

6.1 Machine vision

6.1.1 Exposure control

A typical characteristic in cheap webcams is the optimization for indoor (human face) usage. The exposure control in Logitech QuickCam 5000 works so that it tries to minimize so called “gain” which increases brightness but amplifies noise, and therefore the internal exposure control tries to keep as long exposure as possible. However, this is bad for moving systems, as the images blur in low light conditions. In field robot, usually it does not make a difference if driving in clear sky sunlight, but has difference in cloudy conditions or driving on the evening. This problem was already tried to be solved in Wheels of Corntune (Maksimow et al 2007) and 4M (Backman et al 2008), but it was not possible to program your-own-algorithm, as Windows driver does not support control of exposure from the software. Luckily, the UVC driver for Windows Embedded CE supports direct control of these camera parameters, and now it was possible to do-it-yourself-exposure automation.

320x240 resolution was used. The automatic exposure works in three phases, listed from the brightest conditions to the darkest: 1) gain 0 & exposure > 1/500, 2) gain 0-255, exposure = 1/500, 3) gain 255, exposure < 1/500. The algorithm counts number of pixels and channels that are 255 (which is the maximum). This count number is regulated by a simple controller to 200 (of 320x240x3).

6.1.2 Colour transform

To detect maize plants, a generalized EGRBI transform was used. The original EGRBI transform converts a colour from RGB space to another 3D space: Excessive Green (EG) – Red-Blue (RB) and Intensity. The goal is to detect the green, and the Excessive Green channel excludes intensity changes, so it is easier to threshold the value than in raw G channel. Red-Blue channel makes difference if EG channel says that the pixel is “not green”, to tell “is it more reddish or bluish”. The Intensity channel corresponds to grey-scale. The problem of original EGRBI is the assumption that the colour in interest is exactly green-green. In the case of maize plants the colour is not green-green, but more yellow (red-green) than green-green or green-blue. One example of RGB colour of maize is 160-200-90, experienced by Logitech 5000.

In the Finnish series of field robots, EGRBI transform was tried in Demeter robot with a reasonably good experience. For Wheels of Corntune, the method was generalized the first time, and the generalized version was called as “ECCI” by Maksimow et al (2007). In the generalized EGRBI, or ECCI, the idea is that user may select any colour of interest, and the algorithm computes automatically the transformation to form “EG”, “RB”, and “I” channels. EC (Excessive Colour) channel marks how much the value is given in the direction of colour of interest (excluding intensity variation). Intensity is computed just like in the original version, and the CR (Cross) channel base vector is computed as a cross-vector of EC and I channels, to provide more information. If the colour of interest if 0-255-0 (pure green), the transform is exactly the same as EGRBI. The same method was used also in 4M (Backman et al 2008).

In EasyWheels 2010, the ECCI transform is used. The main difference in the previous editions is hidden in the reasoning: threshold is not used, but instead on each channel the probability is calculated by using on piecewise linear curves. For CR channel the symmetry is assumed, but on the other channels not, so the method provides 5 tuning parameters. The overall probability (of being maize) is computed as a product of the three channels. The parameters are easy to tune by using test images, by first clicking the colour if interest and then finding appropriate values for the corner points of the curves.

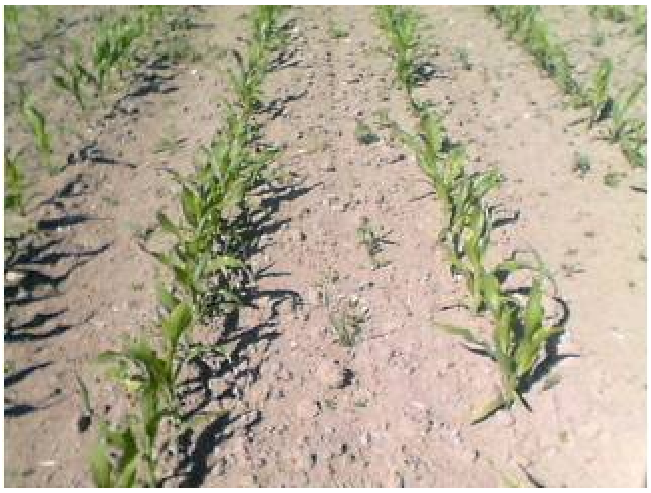


Figure 19: Camera picture from top of mast.

6.1.3 Projection correction

One of the challenges in the project was to deal with the lack of floating-point unit (FPU) in Colibri. Computation of ECCI, correcting the projection and transferring all the data to navigational computer (eBox) was out of the question. The solution was to use pre-computation to form "blocks", or regions-of-interest for the image that correspond certain metric area in the field. The angle of camera was set so that the top edge corresponds to 3.00 meters and in case the bottom edge corresponds about 0.70 meters. A "rasterization" of 12cm x 20cm was used in the preliminary tests, and this metric rasterization means that on top of image smaller number of pixels is used to form a block than in the bottom. By taking the height of camera and aspect ratio of image into consideration, the rasterization produces 268 blocks. For all of them ECCI transform and probability computation is done, based on average values in RGB. The resolution could have been even denser, as in Colibri this required only about 17% of processor power.

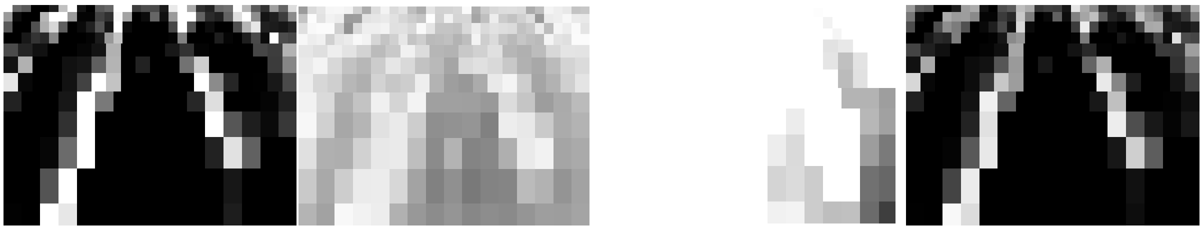


Figure 20: Multiresolution rasterization and ECCI transform, from the left: EC, CR, I, and total probability

6.1.4 Adaptive thresholding

This system also utilized adaptation to finding probabilities. Adaptive thresholding has also been used in Wheels of Corntune (Maksimow et al 2007) and 4M (Backman et al 2008). The assumption in adaptation is that when driving in rows, the average number of pixels which are interpreted as maize, stays (more or less) constant. For adaptation it is possible to set the assumption as a percentage, e.g. 30% has been used, even if this year 10% was a better guess. Here adaptive thresholding controlled upper corner point of EC channel piecewise curve. After the competition and playing with test data it can be said that this was not the best choice.

6.2 Position in row

The position estimation in row is separated from navigation, in order to tune one in time. The row position estimation means finding the two values: lateral position in row (error sideways) and angular position in row (error in heading). The third value is quality which indicates how well the sensor data fits a model. The row model is simply based on assumption on straight maize rows (in 3 meters ahead – 1 meter behind) and 0.75 meter row width (tunable parameter). As EasyWheels is pretty symmetric and the special feature is two-way driving, the natural selection for robot origin is the center point (where the mast is located).

Ultrasonic rangers, infrared rangers and information from machine vision are all projected to the same 2D metric plane. In case of ultrasonic and infrared sensors, the sensor reading is put to the correct geometric position in the robot coordinate system and perpendicular reflection is assumed. Furthermore, in case of ultrasonic and infrared rangers, also historic measurements from last 20 steps (corresponding 1000 ms) are projected, by using backwards odometry computation (backwards in time, historical steering angles, Euler backwards integration) – and historical projected measurements have a forgetting factor, so that the oldest measurement has only 0.2 weight of the newest. The same principle was used already in Wheels of Corntune and 4M robots (Maksimow et al 2007, Backman et al 2008).

Compared with earlier editions, this time also the detected maize probabilities are projected in the same 2D space with ultrasonic and infrared measurements. In this way all the information is processed together, with no intermediate rounding as would happen if the same position in row indices were computed separately. However, this worked in principle, but the problems arise as all the information is not in the same scale. Using only the probability and positive detections was not good enough – also positive probability of “not being maize” has to be counted. Due to lack of time and test data this was not possible for this edition, but it will be investigated in the future.

In line fitting two stages were used to reach computational efficiency. In the first stage a rough estimate is found with a multi-resolution search algorithm: at first, 10 degree resolution is used and then the resolution is decreased to one degree – in the angle in question all the data is projected in the same axis and variation (0.75m cyclic) is counted. The higher variation – the more probable heading angle. In the second stage the data is first rotated to the angle given by rough search (which includes camera head turning), and recursive least squares method with 0.75m modulus is applied – to find a more accurate estimate. A visualization of the method is shown in Figure 21.

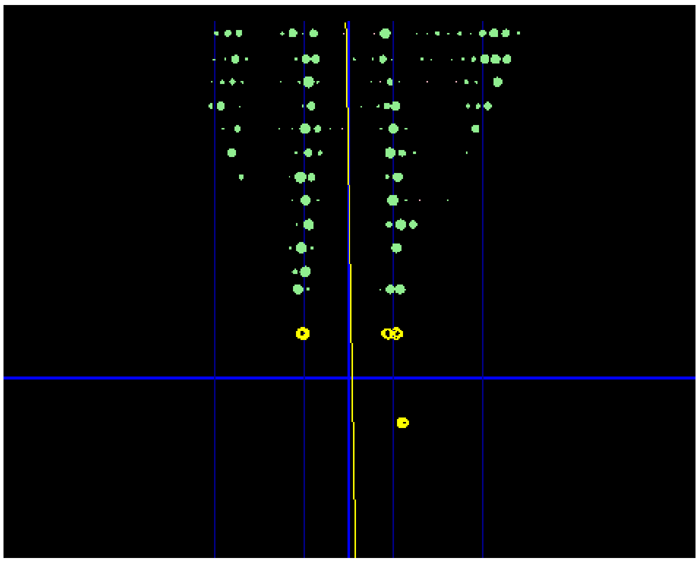


Figure 21: Line fitting diagram: green spots are probabilities of being maize from the image processing, and yellow circles are projected maize plants from ultrasonic sensors in corners of the robot. Robot center position is in the center of blue axis. A yellow line is the result of estimated center driving line.

6.3 Navigation in row

Input to navigation in row is coming from row estimation (lateral error, angular error, and quality) and the output is driving speed, steering angle of front wheels and steering angle of rear wheels. EasyWheels 2010 is using the same navigational principle as its predecessors SmartWheels, Demeter, Wheels of Corntune and Mean Maize Maze Machine (Honkanen et al 2005, Telama et al 2006, Maksimow et al 2007, Backman et al 2008). The principle is simple: two separate PID controllers are used, one of them is handling lateral error control and the other is handling angular error. The output of the first controller is lateral speed (normal to robot heading) and the output of the second is angular velocity.

The lateral speed requirement and angular velocity requirement together with set point for robot speed can be converted to steering angles, by using inverse kinematic model of the four wheel steered robot. In EasyWheels 2010 the angular error is also used as feed-forward to steering angles, which makes it easier to tune the controllers. The main idea of this navigation principle is to make the tuning easier: first the lateral control parameters can be tuned and afterwards the angular – 6 parameters to tune at the same time by hand is too much.

6.4 Navigation in the end of rows

For turning at the end of a row, two methods were implemented: The first utilized two-way driving and works like-a-crab. The other is more common; at first it makes a 90 degree turning, then backwards or forwards some distance and finally completes the last 90 degree turn. In both cased, after detecting the end of row, the robot stops, drives backwards a bit (not to exceed a limit of 1.5 meters in the field), and stops again. Stops between the turning phases make it easier to tune the parameters, and for competition they were minimized.

An intention was to use the camera as a main sensor while the robot was driving in headland, like 4M did, to count how many rows are passed, and on the other hand to keep the orientation right. The angle of the camera was kept always towards to the rows and the angle was adjusted by the measurements obtained from the compass and/or odometry. However, we didn't have enough time to implement camera based navigation at the end of the row, so the competition was done only by using odometry and compass heading.
For improved accuracy to drive a desired distance or a desired angle, a model based odometry was used. The kinematics and identified dynamics of the robot were used in the prediction model to estimate how much the robot would move after setting the driving speed to zero. In other words, the robot all the time thinks "where would I end up if stopping now?", and this predicted position is used as threshold value in the navigation state machine.

6.5 Adaptive cruise control

Each of the axle modules implements so called cruise control, or in the other words keeps the constant speed. For two-robot co-operative challenge, the other feature from the automobiles was to be implemented, so called "adaptive cruise control" or ACC. This system keeps the constant distance to the other car driving ahead, if the car is detected, and otherwise operates like normal cruise control.

In EasyWheels 2010 the adaptive cruise control is implemented by adding another PI control on top of normal cruise control, which adjusts the driving speed up or down from the desired speed. The other robot sent his speed to EasyWheels, and this was “the desired speed”. The output of the PI controller was limited to -0.2 to 0.2 m/s range, so the tuning up or down was done in a safe range.
Both SRF08 and SRF235 sensors were used. There was not enough time for the deep analysis of sensors in the competition, so the beforehand implemented “use minimum of the two sensors”. In the event demonstration the speed of 0.25 m/s was used and a set point for distance was 35cm.

7. Two robot co-operation (with Helios)

We were asked if we wanted to make a co-operative demo for a new Task in Field Robot Event 2010, with Helios. The first idea was to do it with Turtle Beetle, but quickly it was found out that time was running out of that project, and thus the authors ended up taking EasyWheels out of shelf and making the “EasyWheels 2010”. Helios has participated in Field Robot Events every time since 2007 (Knaupp et al 2007; Meinecke et al 2008) and they won the event 2007, were second 2008, and again winners in 2009; so the project was on solid ground – and it was decided to take as a challenge.
After a quick brainstorming, the representatives of the two teams agreed to have demonstration about “corn harvesting”, where one of the robots acts as a harvester and the other as a transporter. Helios team had some kind of “corn shooter” as a tool for Helios in their mind, so evidently Helios was to be the harvester and EasyWheels 2010 to be the transporter. The first idea was to do parallel driving so that one robot in one row and the other following in the next row. However, this was considered more challenging as there was no clear idea how long maize plants will be during the event, and which sensors could be used to detect the position of the other robot – therefore one driving behind another was considered a better choice.

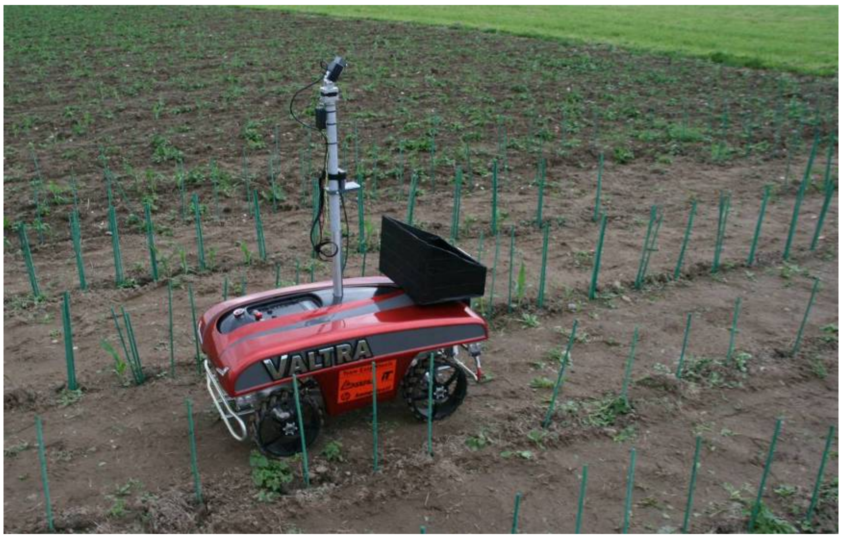


Figure 22: EasyWheels 2010 with a black hopper in Field Robot Event test field.

To demonstrate the job, and taking into consideration a fact that we have only about a day to test the co-operation, the two teams had three aids in use: 1) a wiki-based common workplace, 2) a layered strategy for building up the technology, and 3) simulators to test the communication between the robots.

The layered strategy incorporated from the simplest to complex:

1. one driving behind the other, with fixed turnings, no online communication, manifold acting when Helios sees a row, the second robots starting a bit later than the first
2. online communication: EasyWheels2010 informs when "in contact" e.g. set point in gap reached and Helios informs when the headland starts
3. both robots start from headland, Helios first navigates some rows further, informs EasyWheels2010 of how many rows were counted, Helios drives slowly in row, EasyWheels2010 starts and finds the correct row, and catches up Helios, and informs about the "contact", Helios turns the manifold backwards and simulates blowing
4. some real matter is transferred between robots
5. EasyWheels2010 would go after two rows somewhere to "empty" the hopper...
6. ....and come back...
7. programmable turnings for both robots

In the Event, everything was not going as expected. Helios had some problem in their mechatronics (a front wheel steering problem, or something), and unfortunately they were not able to participate in any other task than the two robot co-operative challenge. Both the robots were communicating with the simulators, but in real life there were some problems, as it took for a while to tune the WLAN access point to right frequency band where no 2.4 GHz noise appeared. The starting from the headland and waiting in turnings by Easy-Wheels was tested in test field, and was proven to be working.

Due to Helios' problems with steering, it was agreed with the jury that Helios will be driven by remote control (Nintendo Wii controller) and EasyWheels 2010 has to act autonomously. Helios had problems with steering, but with remote control this problem was solved, and fortunately we had something to show. A comparison to layered strategy presented above shows that 1) "following" and 4) "shooting a real matter" were working in the competition demonstration. 2) "communication" and 3) "starting position" were implemented and partially tested, but not demonstrated in the Event.

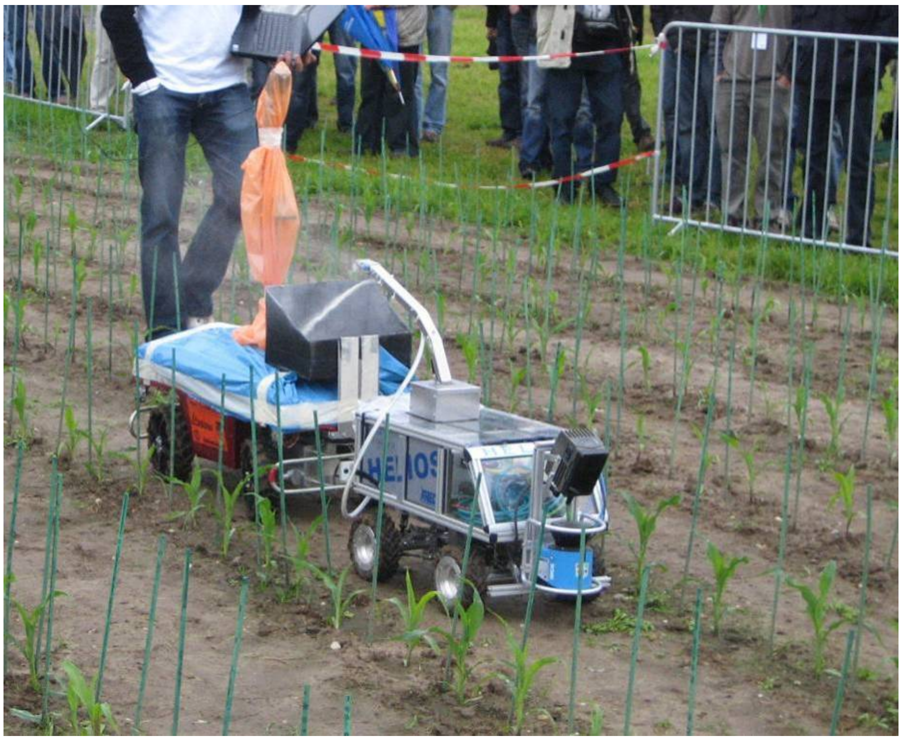


Figure 23: Helios and "well protected" EasyWheels doing the demonstration in FRE 2010.

An experience on two robot co-operative demonstration was positive, as we (the two teams) were able to do some real co-operation. The expected risks were encountered in Braunschweig, as a minor steering servo problem made the autonomous co-operation of the two robots impossible. It was again learned that a robot is a complex system and all the layers in the robot (from mechanics to most advanced algorithms) have to work perfectly to be operative.

8. Discussion

In the Event, the field was different from what was expected. The test field had almost no maize plants, as they were transplanted to the competition field. The only way to do navigation tests in the competition area was to put 8mm diameter gray-green sticks into the soil and try to detect them (see

Figure 22). These sticks were also put into the competition fields, to supplement 10-15 cm height maize plants. EasyWheels 2010 used ultrasonic rangers (4x), infrared rangers (4x) and machine vision to detect the rows (and the end of rows). Based on the first manual drive data in the test rows it came evident that infrared rangers were not able to detect any of the sticks (no use), with machine vision it was very hard to distinguish 8mm gray-green sticks from the ground (no use) and out of four ultrasonic rangers only front-left and rear-right sensors were in such condition that they were able to detect 8mm sticks. The other two ultrasonic sensors seemed to work only in range under 12 cm, as they had worn out somehow, maybe the dust from the field had gone inside the sensors. For future robot builders: be aware with SRF08 condition! So the conclusion was that only two of nine sensors were working, and especially when the most powerful sensor (machine vision) was out of the question, it seemed almost impossible to make it to work in the competition tasks.

However, the robot was capable of driving between stick rows surprisingly well, only by using front-left and rear-right ultrasonic sensors, thanks to sophisticated line detection algorithm. We had only one spare ultrasonic ranger for both EasyWheels and Turtle Beetle, and this was replaced in front-right. With that replacement also row end detection was improved a bit, but still the row navigation in backward driving was a bit too waving – which was seen in Task 1.

The next day after the competition it was possible to do test drives in the competition fields, and finally at that time it was possible to use machine vision as there was enough maize plant green. After short calibration EasyWheels was driving pretty well in the rows, and the driving speed could be increased.

Co-operative challenge proved to be a real challenge – as we co-operated with a team from another country and we had only about one day to do (all) tests before the competition. Luckily, we had prepared well for this task, by developing a clear layered strategy how to do the demo – objectives were set high, but still we had in mind that at least some movement had to be done in case of e.g. wireless communication is not working. Pretty standard WLAN communication between the robots was surprisingly one of the problems as in some phase the communication did not work anymore in channel 8 (mysterious noise?), but after an hour or two investigation it started to work again when changing the channel to 9 – a lot of test time was lost for solving this problem. Helios robot had some mechanical and other problems and they were not able to participate in the other tasks, but luckily we were still able to do the two robot co-operative challenge in a bit eased way: Helios was driven in manual control, and EasyWheels followed autonomously in a constant distance.

9. Conclusions

A good field robot has to be able to adapt to any kind of field, whether it is in good condition or not. To be good in row driving in any conditions, multiple sensors have to be used as one may work better in some conditions and some in the other. EasyWheels 2010 was equipped only with ultrasonic rangers, infrared rangers and machine vision with turning head – the best detection is achieved if all three types of sensors can detect the row. The test field was very challenging, and two of three sensors were not working, but still some row navigation was possible.

The two robot co-operative challenge was considered being challenging. One robot is a pretty complex system and it can be said that if one piece in the system is not working, the whole robot might not work. The risk in the two robot challenge is about on power three, as there are two robots that have to work by themselves, and also the communication and other interaction between them have to work. This risk realized in our two robot demonstration, but luckily a way to do the demo was found – after a lot of work this was better than nothing – and the jury appreciated it with the first prize in co-operative challenge task.

Acknowledgments

At first, we would like to thank all the sponsors (Valtra, Koneviesti, Laserle, HP Compaq, HP Infotech and Suomen Kulttuurirahasto) that supported EasyWheels project in 2009. Secondly we would like to thank all the EasyWheels team members in FRE 2009 – the robot was made with good quality and it provided a good starting point for learning new software technology. Especially the axle modules and the suspension system appeared to be bullet-proof technology. We would also like to thank all the Turtle Beetle team (FRE 2010) members for inspiring co-operation and giving a permission to test our robot in the "orange test field". Gratitude is also given to staff members of Department of Engineering

Design and Production for letting us to play with the robots in the facilities. Finally, we would like to thank all the 44 students participated on Finnish Field Robot Event teams during 2005-2010, to give us motivation to improve education and technology, and to learn more and more about field robots.

References

Backman, J., Hyyti, H., Kalmari, J., Kinnari, J., Hakala, A., Poutiainen, V., Tamminen, P., Väätäinen, H., Oksanen, T., Kostamo, J. and Tiusanen, J. 2008. **4M – Mean Maize Maze Machine.** In Proceedings of the 6th Field Robot Event 2008. ISBN 978-3-00-027341-4. pp. 9-41. http://www.fieldrobot.nl/downloads/Proceedings_FRE2008.pdf

Honkanen, M., Kannas, K., Suna, H., Syvänne, J., Oksanen, T., Gröhn, H., Hakojärvi, M., Kyrö, A., Selinheimo, M., Säteri, J. and Tiusanen, J. 2005. **The development of an Autonomous Robot for Outdoor Conditions.** In Proceedings of the 3rd Field Robot Event 2005. ISBN 90-6754-969-X. pp. 73-90. http://www.fieldrobot.nl/downloads/Proceedings_FRE2005.pdf

Kemppainen, T, Koski, T., Hirvelä, J., Lillhannus J., Turunen, T., Lehto J., Koivisto, V., Niskanen, M., Oksanen, T., Kostamo, J. and Tamminen, P. 2009. **Robot Brothers EasyWheels and ReD in Field Robot Event 2009.** In Proceedings of 7th Field Robot Event 2009, Wageningen, the Netherlands. ISBN 978-90-8585-744-0. pp. 37-64. http://www.fieldrobot.nl/downloads/Proceedings_FRE2009.pdf

Knaupp, J., Meyer, R., Meinecke, M., Robert, M., Schattenberg, J. and Schlott, J. 2007. **Autonomous Field Robot System "Helios".** In Proceedings of the 5th Field Robot Event 2007. pp. 35-44. http://www.fieldrobot.nl/downloads/Proceedings_FRE2007.pdf

Maksimow, T., Hölttä, J., Junkkala, J., Koskela, P., Lämsä, E.J., Posio, M., Oksanen, T. and Tiusanen, J. 2007. **Wheels of Corntune.** In Proceedings of the 5th Field Robot Event 2007. pp. 75-87. http://www.fieldrobot.nl/downloads/Proceedings_FRE2007.pdf

Meinecke, M., Robert, M., Roesler, J., Roos, L., Schattenberg, J., Schwerter, M. and Göres, T. 2008. **Hard- and Software concept of the autonomous Field-Robot Helios.** In Proceedings of the 6th Field Robot Event 2008. ISBN 978-3-00-027341-4. pp. 65-74. http://www.fieldrobot.nl/downloads/Proceedings_FRE2008.pdf

Pentikäinen, J., Pihlanko, M., Pihlanko, P., Helenius, J., Tuhkanen, E., Matilainen, M., Sairanen, M., Oksanen, T., Kostamo, J., and Tamminen P. 2010. **Turtle Beetle.** In Proceedings of the 8th Field Robot Event 2010. (in print).

Telama, M., Turtiainen, J., Viinanen, P., Kostamo, J., Mussalo, V., Virtanen, T., Oksanen, T. and Tiusanen, J. 2006. **DEMETER – Autonomous Field Robot.** In Proceedings of the 4th Field Robot Event 2006, ISBN 978-90-8585-480-7. pp. 27-38. http://www.fieldrobot.nl/downloads/Proceedings_FRE2006.pdf

BaneBots Robot Parts, http://www.banebots.com

EmbeddedPC.NET, http://www.embeddedpc.net

Futurlec, http://www.futurlec.com

ICOP, http://www.icoptech.com

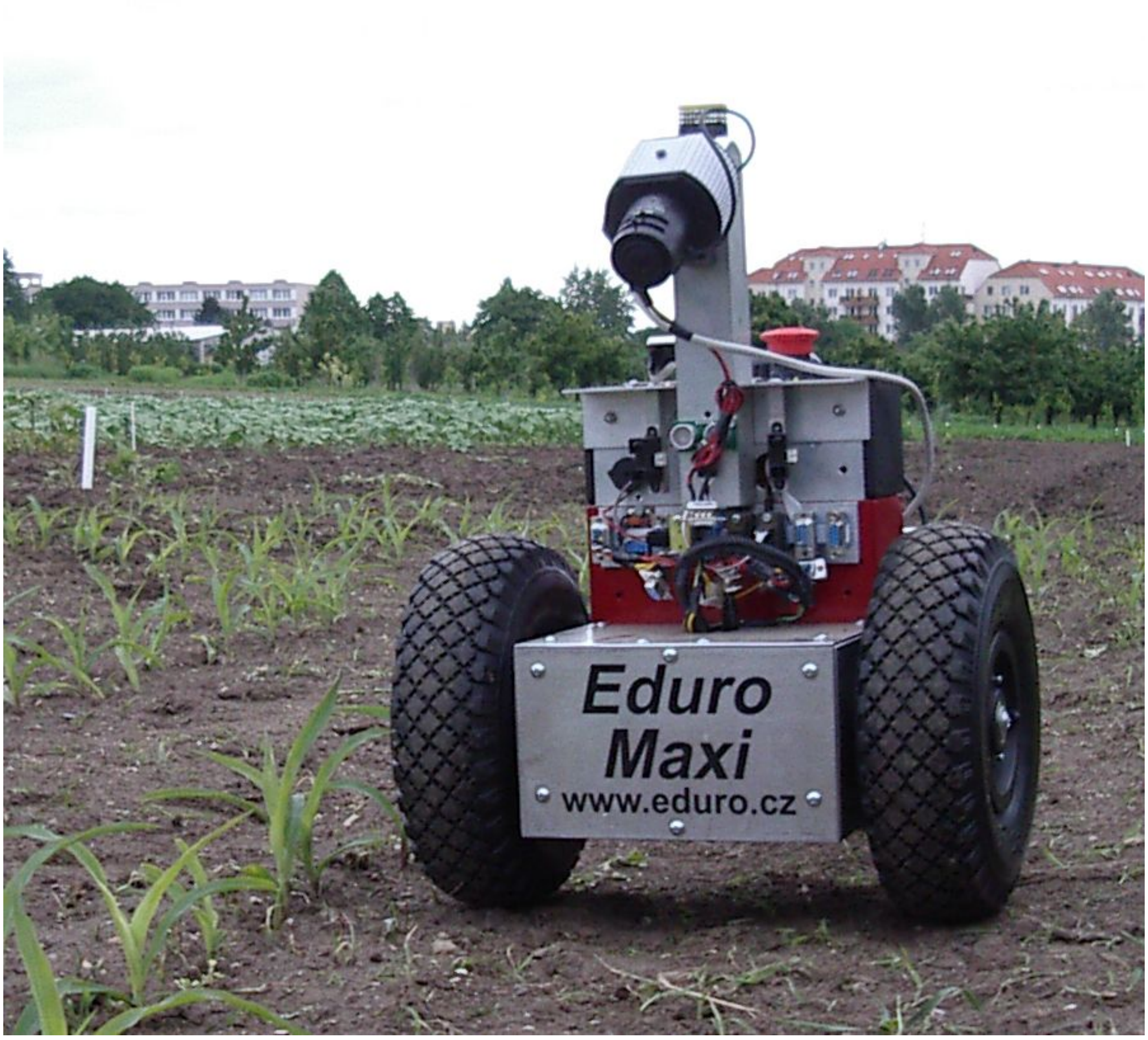
Laserle Oy, http://www.laserle.fi

OpenCV, Open Source Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/

Toradex,http://www.toradex.com

Eduro Maxi HD – Navigation in the maize

Milan Kroulík[1], Jan Roubíček, Tomáš Roubíček, Martin Dlouhý
[1] *Czech University of Life Sciences Prague, Department of Agricultural Machines*



Abstract

This article describes robot Eduro Maxi HD used by the Eduro Team for the Field Robot Event 2010. The goal of the team was to build and program the robot and demonstrate its usability in agricultural task.

Keywords: *EDUcational RObot, CAN bus*

1. Introduction

The Eduro Team for FRE2010 was formed in coordination with Czech University of Life Sciences Prague, Department of Agricultural Machines. Last year experience showed, that such partnership could be fruitful, and our expectations were fulfilled. We are bond together by a belief, that robots can and should be used for tasks humans cannot perform well or even at all. Our past experience with the robotic constructions and our previous participation on the robotic competitions (Eurobot, Robotour, RobotChallenge, ...) has shown us an applicability of our approach and of our research ideas to the robotics.
The past has also learned us that it is not a lack of ideas which stops the robots from performing well, but a lack of reliability. Because of this observation, we focus on the reliability instead of the complexity. Our approach is further reinforced by a necessary trade-off between computing power of the control board and its power consumption. There are many nice and important research ideas, which just cannot be implemented on a real robot yet. However, some can.

2. The Robot

2.1 Hardware

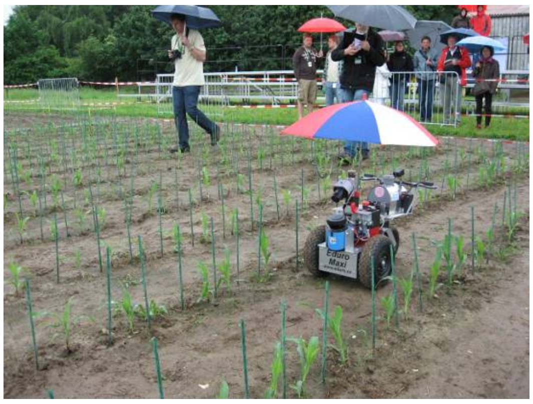


Figure 24:Eduro Maxi HD during the FieldRobotEvent 2010

Eduro Maxi HD is the prototype of three-wheel outdoor robot with a differential drive (Fig. 1). It is the modular robotic platform for education, research and contests. It weights about 15 kg with the dimensions of 38x60x56 cm. HD version uses SMAC (Stepper Motor – Adaptive Control) drives with belt transmission. The power supply is managed by two Pb batteries 12V/8Ah. The brain of the robot is single board computer with AMD Geode CPU, 256 MB RAM, compact flash card, wi-fi, 3 Ethernet, 1 RS232 and 2 USB ports. The RS232 port is used for connection to CAN bus via RS232-CAN converter. The most of sensors, actuators and other modules (display, beeper, power management etc.) are connected to CAN bus, which forms the backbone of the robot. The more data rate demanding sensors are connected directly to PC via ethernet interface. Two main sensors were used for the Field Robot Event. The weed detection was provided by an IP camera with fish-eye lens. For obstacle detection, the laser range finder SICK LMS100 was used. The robot is further equipped with sonar, GPS and compass, but these sensors were not used in main tasks during the Field Robot Event.

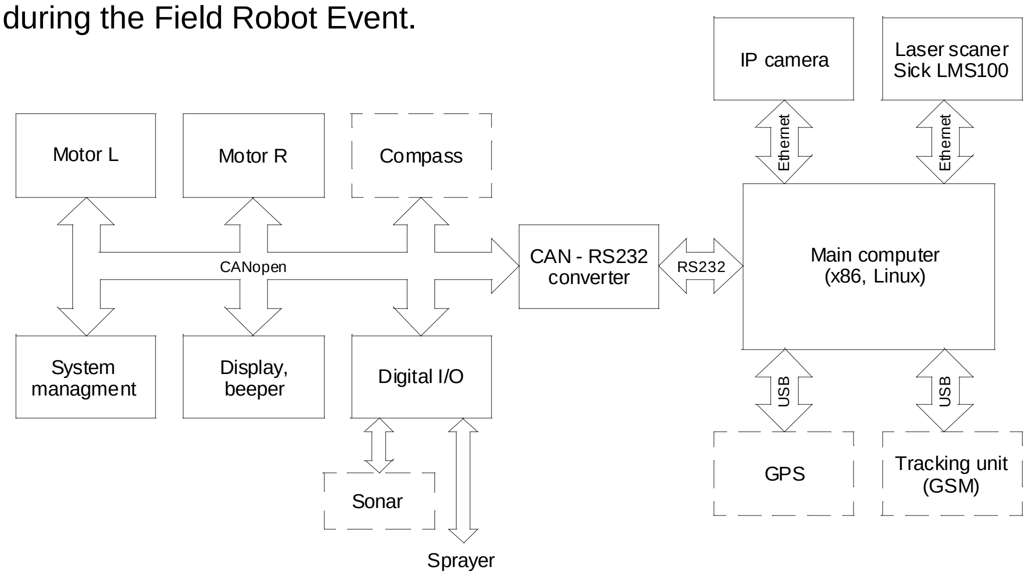


Figure 25: Hardware structure of the Eduro

The robot carried spraying system on the back. The spraying system had been designed as model of a real agriculture machine and was shared with team Cogito MART. The sprayer is independent module with own battery and simple interface (3pin connector - ground, left, right). Besides spraying it blinks and beeps.

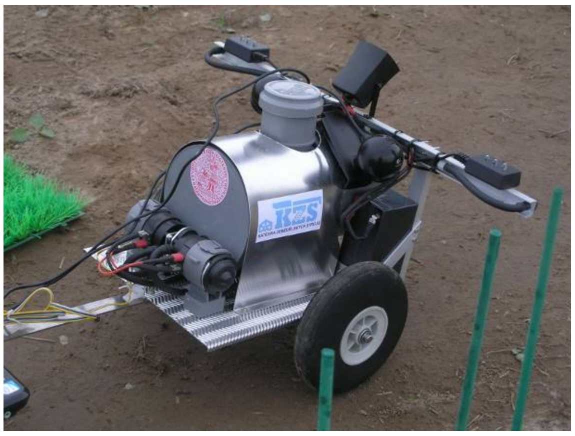


Figure 3: The sprayer

2.2 Software

The main program is running on x86 single board computer with the operating system Linux and Xenomai real-time extension. The Xenomai is used for decreasing of serial port interrupt latency and improvement of communication reliability.
The low-level motor control is provided by microcontrollers in motor modules. The main computer only sends desired speed to the units via CAN bus.

The highlevel software is written in Python. It is using routines for image processing written in C/C++ with help of OpenCV library. It turned out that all three tasks could be unified in one code:

```
contest = FieldRobot( robot, verbose )
contest.ver2([-1,1]*10)                  # Task1
contest.ver2([2,-1,2,-3,-4,0,2,4,-1]) # Task2
contest.ver2([1,-1]*10)          # Task3
```

The code is relatively compact and it has approx. 300 lines including logging, prints and sprayer specific routines.

The navigation in the row was originally handled with camera. The camera was much more successful on the grown maize test field in Prague when compared to laser scaner pointing down.

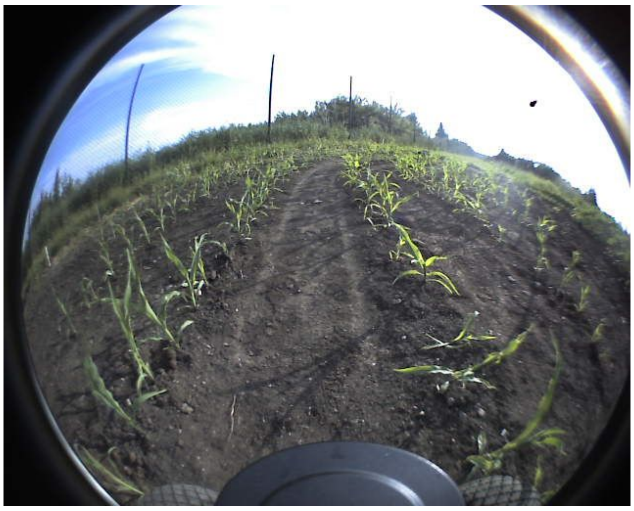


Figure 4: Robot view of test filed in Prague

The camera row navigation was based on color segmentation (green/white), where green pixel was considered if in RGB color space G > 1.05*R && G > 1.0*B.
The image was processed from dynamic center to left/right until sum of green pixels was above given threshold.

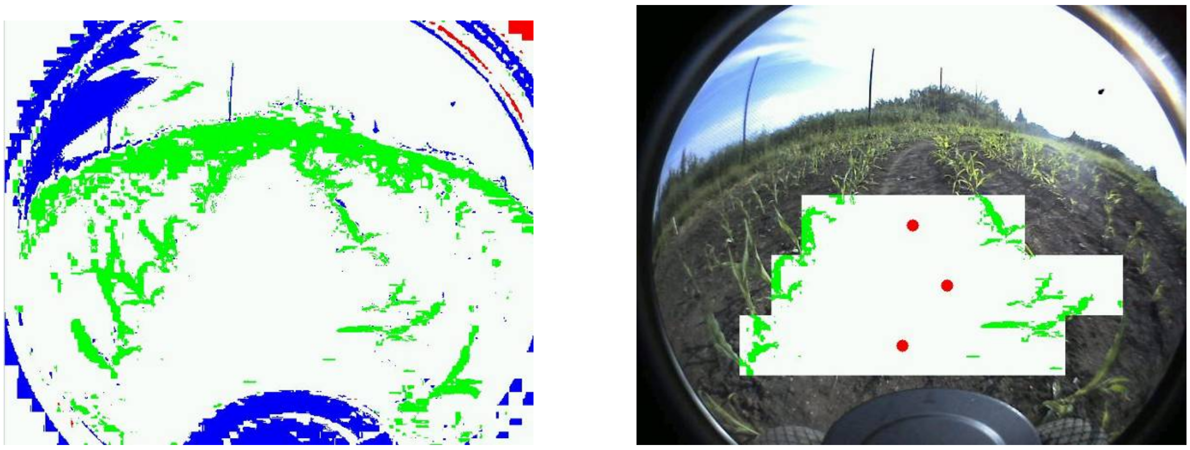


Figure 5: Color segmented image and located positions of rows

Similar algorithm was used for weed detection. The only difference was definition of bright green: (G > 1.3*R && G > 1.3*B). The weed was detected in front of the robot so the spraying had to be delayed by time when robot travels approximately 75cm. The implementation used extra task queue with time offset.

Laser row navigation
In Braunschweig we used laser navigation instead of camera. The sensor was still pointing 10deg down to eliminate uneven terrain. Input was 541 measurements with half degree step. A simple algorithm searched for center of row – as an obstacle was considered measurement shorter than 1m. Also for the end of row detection we used simple rule then the space opened wider than 85degrees on both sides.

3. Conclusions

The Field Robot Event 2010 was for Eduro Team quite successful. The team got 4 prices in total with gold in Professional Task, and ranked 2nd in overall evaluation.

Figure 6: Eduro Team for FRE2010

Acknowledgments

Thank you organizers for very nice competition and the Czech University of Life Sciences Prague, Department of Agricultural Machines for their support.

References

http://kzs.tf.czu.cz/

http://robotika.cz/

http://www.eduro.cz/

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

KATEDRA ZEMĚDĚLSKÝCH STROJŮ

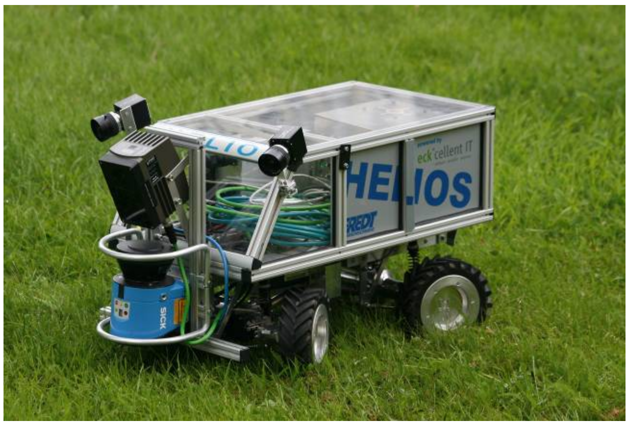
Fredt

Martin Schwerter, Niko Brasch, Philipp Koitsch, Kai Uwe Jesussek, Jan Roesler

TU Braunschweig, Institut für Landmaschinen und Fluidtechnik, Langer Kamp 19a, Braunschweig

1. Introduction

In previous events the basic mechanical construction of Helios emphasized as being robust and solid. Referring to the recent event in Braunschweig, the most important points to do were software-based and new mechanical parts for the tasks.
Therefore a new manipulator and a spout had to be constructed. Software sided the weed detection and the trajectory planning was redesigned.



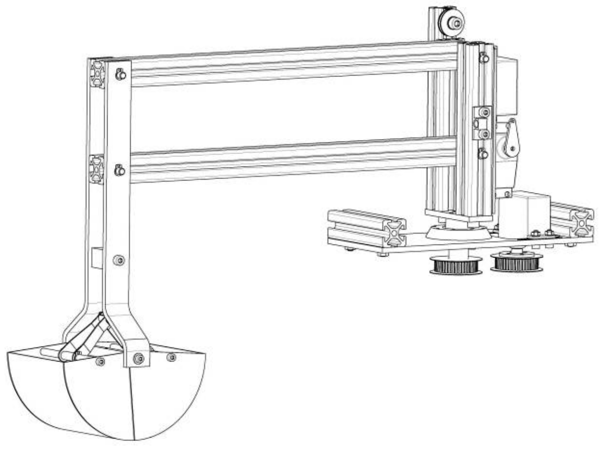
1.1. Basic System

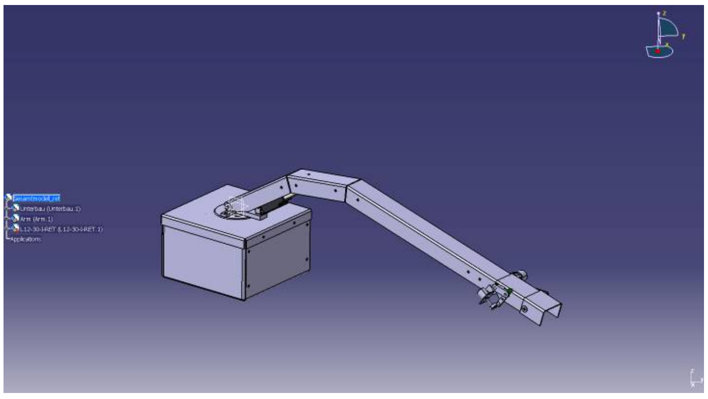
The basic system is similar to the one used in Wageningen and Osnabrück. An aluminium chassis is carrying the body made of ITEM aluminium profiles. Right under the base plate the powerful Dunkermotor engine is installed, which drives all four wheels over three differentials. Both axles are steerable using two servos. These servos had to be changed this year due to transmit the power via tooth belt.

1.2. Manipulator

In order to defeat the weed without using chemicals, we decided to build a manipulator. A parallelogram rotary bedded is able to spin about 180° to catch weed on both sides of Heilos, At the down end, a bucket (2 parted) picks the weed. As actuators servos were used, one of them had to be rebuild to an endless winding one to lift the bucket via a winch.

Many of the parts the manipulator is built with had to be milled, for the parallelogram ITEM profiles have been very useful.





1.3. Overburden machine

To demonstrate an overburden process in the cooperation task an overburden machine had to be constructed. The base of this machine is a box containing the closed loop control and one servo to spin the machine. To accelerate it, a linear motor is set between the arm and the pivot.

A second servo is mounted at the end of the arm for sighting.

To visualize the overloading an air pressure system shoots a small load of any fine grained substrate out of the end of the arm. The system uses two different tanks one for the compressed air and another one for the substrate. The overburden machine is controlled via CAN-Messages which transfer the position data.

2. Electronic systems

2.1. Main computer

In contrast to our first experiences years ago, we are not using microcontrollers for the main arithmetic system. Due to the huge amount of data from sensors like the laser scanner or cameras, we decided to use an Intel Atom dual core Mini ITX. It is able to run both the trajectory planning and camera analysis. The Mini ITX Board is equipped with a GPU which has a CUDA® capability. So it is possible to use the GPU for some calculations. In our case it's used for a faster image processing.

2.2. Bus system

For connecting the Klickbox and the servo boxes a fast and reliable bus - the CAN bus as used in automotive industry since several years - is most applicable. It can be easily analysed by computer so errors can be detected fastly.
The Sick laser scanner, the two AVT Prosilica cameras and the PMD-camera are equipped with Ethernet. Therefore Ethernet had to be installed as the second bus system.
With these two buses, we are able to add many other electronic parts in future.

2.3. Klickbox

The most components in Helios can be switched on/off using the so called Klickbox. It is connected to the ITX board via CAN. All channels can also be switched manually.
Another advantage of this Box is the possibility to plug in an external power supply without the loss of power on the robot during the connecting.

2.4. Servobox

In order to control the servos via CAN we are using servo boxes. The older ones can control standard servos known from the modelling scene (using PWM-signals), the newer boxes are able to control servos using RS232. This admits us to get back the steering angles from the servos, so a closed loop control could be designed.

3. Software concept

3.1. Basic concept

We split up the Software into smaller parts which work certain and can be used for varieties of different tasks. Due to the fact that many components of the hardware communicate via CAN bus and there are several programs who need to read and write on the bus, the CAN messages are redirected by the CAN Control program over TCP/IP to all other programs. The electric motor and the servo motors send the distance and steering angle which are important for the closed loop control. For indoor and primitive testing a program was written to test the basic functions of the path planning algorithms. It simulates the scanner, the motor and the servo motors. To manage all the parameters and to have a graphical feedback of the scanner data and trajectory generation process we have a very simple GUI which communicates over TCP/IP on the one side with CAN Control and on the other side with the path planning application to get the effective laser scanner data.

3.2. Trajectory planning

The data from the laser scanner is taken to calculate the inner edge of the driving lane. Therefore the algorithm starts at the robot and tries to find the nearest object which fits to the boundary conditions. Then the next object is calculated from the last object. This algorithm is used to find the left and the right side of the maize rows. The boundary conditions contain maximum and minimum distances to the robot and to the last objects. After that the trajectory is generated on the base of the driving lane. The trajectory is described by a cubic function. This cubic function is transformed into a global coordinate system and sent to the closed loop control application.

3.3. Turning around

Even though it was planned to turn into the next row with a self generated trajectory, at the moment the turn is driven like last year. To drive along the headland the above described trajectory generation is also used. The planed pattern is saved in an array.

3.4. Obstacle detection

In order to detect obstacles and dykes a PMD-3D-camera is used. An algorithm, which analyzes beginning at the robots front the ground surface, checks if the ground has bumps ore dykes ore something else blockades the way. The robot automatically stops if an obstacle was found.

3.5. Camera analysis

Technical Proceedings:

Weed Detection on the field Robot Helios is based on a difficult system and is done not only on CPU but also on GPU.
For a better reusability, there is a middleware between the Capture Devices and the image processor. This middleware called WebcamD and shares the image from a Capture Device through a HTTP web server.

The splitting of the processes has some good effects:

- It is possible to test with a cheap webcam or a dummy video, and the image processor doesn't see a difference
- It is possible to move this part to a different computer, to increase performance
- Debugging is quite easy, you only need a web browser

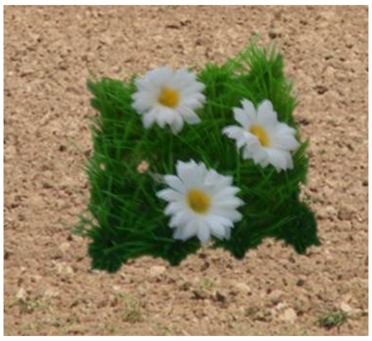
WebcamD is written in C++ and uses following libraries:

OpenCV	Image processing Dummy capture driver
PvApi	AVT GigE cameras
UniApi	AVT Firewire cameras
Libmicrohttp	HTTP server library

The Image Processing is done with a separate program.
This program is written in C++, too and uses following Libraries

libCURL	Fetch images through HTTP
OpenCV	General image processing
cvBlob	Find blobs in binarized Images
OpenCL	Split image into different Layers on the GPU
SQLite3	Store information about blobs in a database

The following steps were processed with every camera:

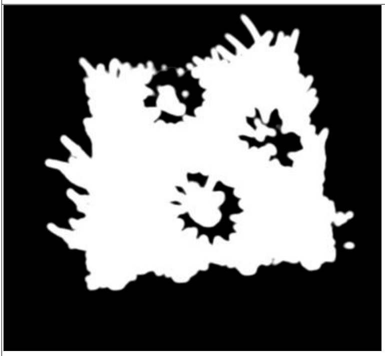
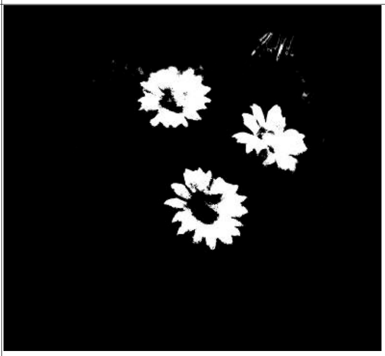
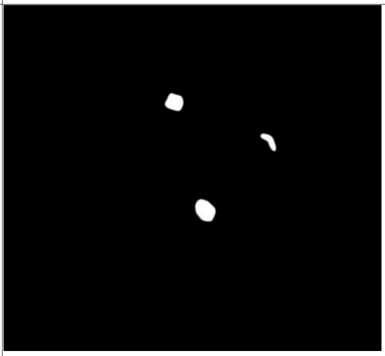


- Fetch Image trough HTTP

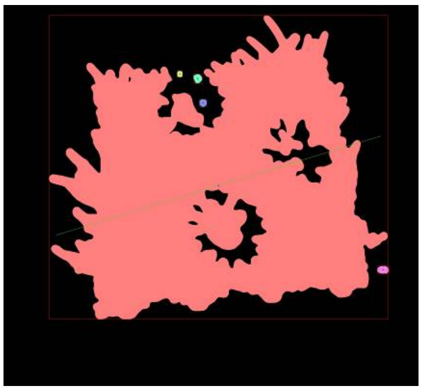
The image fetching is done through the HTTP Protocol in version 1.1 with the curl library.

- Split Image into different layers

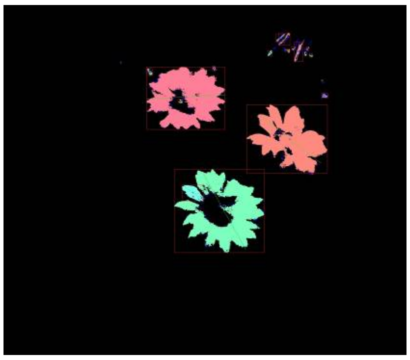
The splitting of an Image into different layers is made with the RGB Values of each Pixel in an image. This Part is done for each Pixel and is computed due to performance on the GPU.

Green	White	Yellow
		

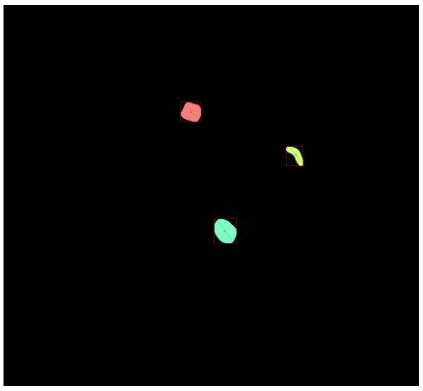
FieldRobotEvent 2010



- Find blob in green layer
  - Filter out blobs that were to small or too big
- Find blob inside of green layer in white layer
  - Filter out blobs that were too small
  - More blobs are better ranked



- Find blob in yellow layer inside in white layer
  - If blobs in this layer were found, it is really sure, that a flower was detected
- Store blob information's inside an in memory SQLite3 Database
- *Search inside of SQLite3 Database for old Blobs to check movement of this blob*
  - *Area size of database entry must be less or equal green blob area size*
  - *Centroid of database entry must be around centroid of green blob*



- *If blob is new, give it a new unique id / else give it the id of the found blob*
- *If a blob with flowers inside was not found in the last 3 frames, signalize it to the main program via TCP/IP*
  - *Main Program uses weed manipulator or acoustic signal to react on this*
- *Return to the first step*

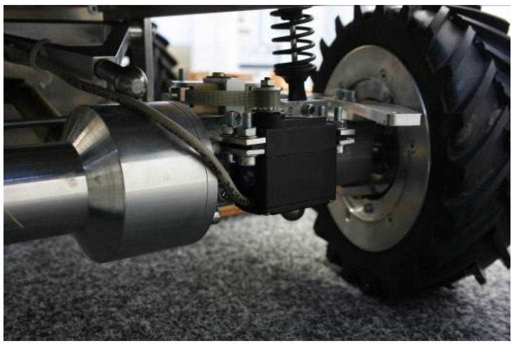
3.6. Cooperative Task

3.7.

For the cooperative task together with the Finnish team we decided to show an overburden process.
Therefore the above mentioned spout should be used. To send information from one robot to the other using the available wireless LAN adapters the UDP protocol fitted the best.
The data structure was discussed via email and a wiki board and the process was concretized.
The complexity of the process was increased during the work depending on the time and the progess.

4. Discussion

This time we concentrated too much on the more complex tasks like the manipulator the overburden process or the improvement of the driving lane detection and forgot about the importance of a stable running system for the basic tasks. And we experienced that a supposed improvement of a specific part can tie up the whole vehicle. In this case the overleaping of the steering belt lead to an unacceptable deviation in the single-track model.



Hugo

Frits K. van Evert[1], Martijn van der Bijl[2], Arjan Lamaker[3], Tom Stravers[2], Gerrit Polder[1], Gerie W.A.M. van der Heijden[1], Bastian Kroon[2], Jasper Knol[2], Maurice Dhaene[2], Ton van der Zalm[1], Tijmen Bakker[4], Lambertus A.P. Lotz[1]

[1]Wageningen UR, PO Box 616 Wageningen, The Netherlands. E-mail: (frits.vanevert, gerrit.polder, gerie.vanderheijden, ton.vanderzalm, bert.lotz)@wur.nl
[2]Kverneland Group, Nieuw Vennep, The Netherlands. E-mail: (martijn.vanderbijl, tom.stravers, bastian.kroon, jasper.knol, Maurice.dhaene)@kvernelandgroup.com
[3]MARIN, Wageningen, The Netherlands. E-mail: a.lamaker@marin.nl
[4]Tyker Technology, Wageningen, The Netherlands. E-mail: tijmen.bakker@tyker.com

Abstract

Our earlier robots have not solved the Field Robot Event's row-following problem with a sufficient degree of robustness. The objective of the work presented here was to build a robot that can detect rows consisting of small or large maize plants by using a camera system; and to provide this robot with robust localization and navigation by using probabilistic methods to process the data from the vision system in conjunction with data from other sensors. We employed a particle filter approach where information from the robot's wheel encoders and a gyroscope is used in the control step and where the filter is updated using information from a downward-looking camera and a laser scanner. For the weed detection and control tasks, the robot is equipped with a self-contained spray unit consisting of two CMUCAM3 camera's and four narrow-cone nozzles (two on each side of the robot) which allow for precision-treatment of small areas. At the Field Robot Event, the robot was able to follow rows and turn into the correct new row in all tasks. No manual intervention was necessary; the first objective was met. In the days and weeks leading up to the event, it was demonstrated that the robot can navigate even when the maize plants are very small. Thus, the second objective was also met. However, weed detection was less than perfect. It turned out to be more sensitive to the light conditions than we had realized. Also, the turf patches were placed almost between the maize plants instead of well inside the row, and were out of the camera's view. In conclusion, the robot is capable of a high degree of autonomy in the tasks of the Field Robot Event: it didn't once get lost and it damaged few plants.

Keywords: *camera, laser scanner, gyroscope, probabilistic framework*

1. Introduction

Our team has taken part in several previous editions of the Field Robot Event. The first of our robots was built in 2004 (Van Evert et al., 2004). The second robot detected volunteer potato plants in maize fields (Van Evert et al., 2006). The third robot detected broad-leaved weeds in grass fields (Polder et al., 2007). The weed detection capability of this robot was later implemented in a farm-sized prototype robot (Van Evert et al., 2009; Van Evert et al., 2010). These efforts were followed by a two-wheel balancing robot in 2007 (Van Evert et al., 2008) which reappeared as a three-wheeler in 2009 (not documented).

While there have thus been some results and an important spin-off, it does not seem that the Field Robot Event's row-following problem has been solved with a sufficient degree of robustness. The problem seems simple enough: follow rows of maize plants, detect the end of the row, and enter a row to the left or the right, possibly skipping one or more rows. At the time of the Field Robot Event, maize plants typically are 0.50 m tall. Thus, they are easily detected with a vision system or with proximity sensors (IR, ultrasonic, laser). However, our team interprets the challenge as a proxy for realistic agricultural problems, of which weed control is the most obvious one. Intervention for weed control is most beneficial if it is performed when the crop is still small, possibly only a few cm high. Thus, a row following solution that works with a variety of plant sizes is needed.

A second problem that has not been solved fully is completely autonomous operation of the robot under a variety of circumstances. The robots we have built (and indeed practically all other robots participating in the Field Robot Events of past years) occasionally lose track of where they are, either while following rows or while on the headland, or they miscount the number of rows crossed and enter into a wrong row. Clearly, the challenges posed to robots that attempt to navigate a farm field are many: an irregular surface with bumps, holes, and variable traction; plants with irregular shapes and sizes; dirt and dust spread by wind and rain; and extremely variable light conditions that will confuse camera's and IR sensors. An entry for the Field Robot Event should explicitly target these challenges. Probabilistic methods to integrate sensor data over time and over different sensors offer inherent robustness (Thrun et al., 2005). Earlier, we combined Hough transform-based row detection with a particle filter (Van Evert et al., 2008). Later, the Hough transform was omitted (Van der Heijden et al., 2008).

The objective of the work presented here was to build a robot that can detect rows consisting of small or large maize plants by using a camera system; and to provide this robot with robust localization and navigation by using probabilistic methods to process the data from the vision system in conjunction with data from other sensors.

1.1 Robot hardware

Robot “Hugo” is designed around the self-contained Kverneland wheel unit (described by Hofstee et al. (2007)). This unit consists of an actuated, steering, wheel. It is self-contained because it comprises a microprocessor which controls both speed and direction of the wheel. Communication with the outside world is effectuated through CAN-bus and consists of commands from high-level control to the wheel and status messages from the wheel to high-level control.
The robot is equipped with one fully functional wheel unit. This unit is located in the front. The two rear wheels are essentially wheel units without the ability to steer. Equipping the robot with three (or four) fully functional wheel units would have made it more manoeuvrable, but would also have increased the mass as well as raised its center, thereby causing instability (Fig. 1).
The wheel units are equipped with incremental encoders to measure rotational speed of each wheel. In addition, the front wheel unit is equipped with an absolute angle sensor to measure the steering angle.
With three driven wheels, one of which can be steered, the kinematic model depicted in Fig. 2 applies to the robot. When turning, each wheel is driven with a speed that depends on the target speed of the controlled point, the radius of the turning circle, and the location of the wheel relative to the controlled point.

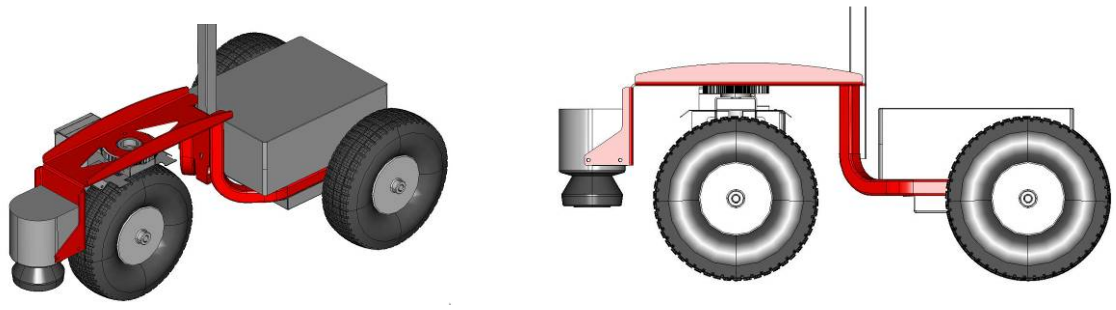


Figure 1. Drawings of the robot. Left: all components are shown, including laser scanner in front and electronics box between the rear wheels. Right: side view showing that the wheel unit requires a lot of space.

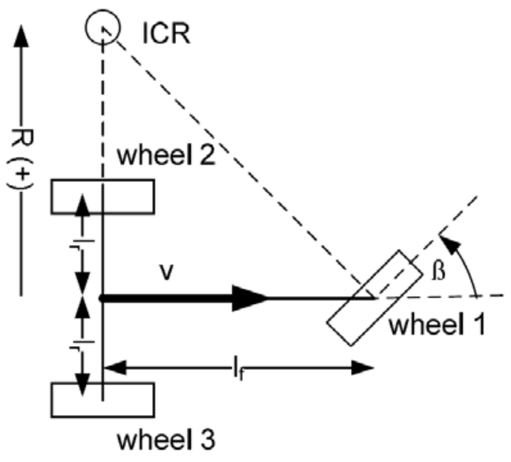


Figure 2. Kinematic model of the robot.

The robot is equipped with the following sensors to navigate. A downward-looking camera (uEye UI-1220 SE, IDS Imaging Development Systems GmbH, Obersulm, Germany) with a 2.4 mm, 186 degrees field-of-view lens CF2420 (Lensation GmbH, Karlsruhe, Germany) is mounted at a height of 1.65 m and provides a view of the robot and its vicinity. A laser scanner (LMS-111, Sick AG, Waldkirch, Germany) is mounted to the front of the robot and allows detection of obstacles in front of and to the sides of the robot. Finally, a gyroscope (Inertia-Link, Microstrain Inc., Williston VT, USA) provides information about the rotational speed of the robot.

The robot has a mini-ITX computer with a 2.4 GHz Intel Core2 Duo processor, running Windows XP. Also present is a WiFi-enabled LAN router.

Energy is provided by three 12 V NiMH racing packs: 1 for the front wheel unit, one for both rear wheel units, and one for the PC and router. The robot is pictured in Fig. 3.

For the weed detection and control tasks, the robot is equipped with a self-contained spray unit. The unit consists of two CMUCAM3 (one on each side of the robot), and four narrow-cone nozzles (two on each side of the robot). Also present are a small water tank, a pump, and four valves. The spray unit is powered by a 12 V NiMh accu pack. The nozzles on each side of the robot are adjusted so that it is possible to treat a small area (Fig. 4).

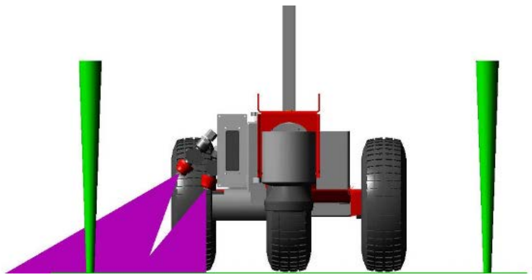


Figure 3.Schematic representation of the spray unit. Please note that the two nozzles on one side of the robot each treat a different area.

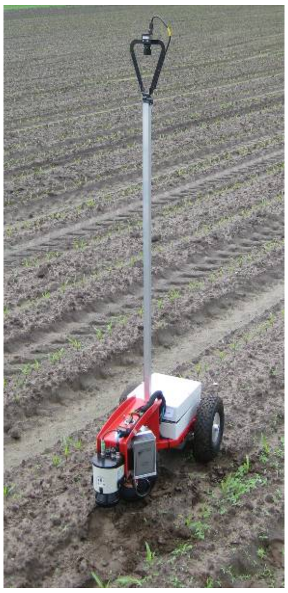


Figure 4. Robot Hugo pictured in a field with small maize plants.

2. Localization

The robot navigates in a world which consists of a row of maize plants to its left and a row of maize plants to its right. The rows extend beyond the field of view of the sensors opposite to the direction of travel, and they may or may not extend beyond the field of view in the direction of travel. The robot may find itself between the rows, or it may also find itself outside the rows (on the headland). The state of the world and the position of the robot in the world can be described by six state variables (Table 1 and Fig. 5).

Table 1. State variables for localization.

Symbol	**Description**
D	Distance between rows, m
W	Width of rows, m
H	Heading, radians
L	Lateral deviation of the robot from the center line between two rows of maize, m
E_L	Distance from the robot to the end of left-hand side row of maize, m
E_R	Distance from the robot to end of the right-hand side row of maize, m

Estimates for these state variables are necessary for the robot to navigate. At any given time, the current estimate of the state variables is called the “belief” of the robot. Here we adopt a probabilistic framework, meaning that a probability distribution is maintained for the belief, rather than a set of deterministic values. In particular, we employed a particle filter approach where the joint distribution of the state variables is estimated with a large number of so-called particles and where each particle holds one estimate for each state variable. A good introduction to particle filters in the context of robotics is given by Thrun et al. (2005).

The particles are initialized by drawing from a uniform distribution for each state variable that reflects the starting position of the robot (in the middle of, and parallel to, two rows of maize plants).

In the control step of the particle filter algorithm, the state variables are updated using information from the front wheel unit about the distance travelled, and information from the gyroscope about the turning speed of the robot. The particle transition function is given by:

$$
\begin{aligned}
H &= H + dH + N(0, \sigma_H) \\
L &= L + dx \sin(H) + N(0, \sigma_L) \\
W &= W + N(0, \sigma_W) \\
D &= D + N(0, \sigma_D) \\
E_L &= E_L - \cos(H)\, dx + N(0, \sigma_{EL}) \\
E_R &= E_R - \cos(H)\, dx + N(0, \sigma_{ER})
\end{aligned}
$$

where H, L, W, D, E_L , and E_R are the state variables as given earlier; N(0, σ) denotes the Normal distribution with mean 0 and standard deviation σ; σ_H , σ_L, σ_W, σ_D, σ_{EL}, σ_{ER} are the standard deviations associated with the transition for each state variable; dH is the change in the robot’s heading as measured by the gyroscope; and dx is the longitudinal displacement of the robot as measured by the wheel encoders.

In the update step of the particle filter algorithm, information from the camera and the laser scanner is used to assign a weight to each particle. Particles with low weights are less likely to be retained in the resampling step that concludes the particle filter algorithm.

The camera provides information that can be directly related to the robot’s world model (Fig. 6). From the belief of the robot an image can be constructed that corresponds to the camera’s field of view, and where the value of each pixel represents the strength with which we expect to find plant material at that location (Fig. 7).

Detecting the ends of the rows of maize requires a modification of the general particle filter algorithm as described above. The particle state transition function always decreases the value of the state variables that describe the distance to the end of the rows (E_L and E_R),

even if the robot is still very far away from the end of the row. After each update stop, E_L and E_R are re-initialized in a small percentage of the particles to values that are just outside the field of view of the camera. When the end of the row enters the camera's field of view, these (partially) re-initialized particles receive a much higher weight than most other particles and allow the particle filter to detect the end of the rows correctly.
The camera-based end-of-row detection works well when there is much green material in the rows, but fails when there are few plants or when the plants are small. In particular, we found that the end-of-row detection performed poorly with the non-standard maize plants during this year's Field Robot Event. The algorithm was therefore modified so that in the update step the state variables that describe the distance to the end of the rows (E_L and E_R) were re-initialized to the largest distance from which a return was received by the laser scanner.
An important element of navigation on the headland is counting how many rows have been crossed. This was implemented by maintaining a row counter which initially is set to zero. When the robot travels on the headland and crosses over from one row to the next, the (absolute) value of the "lateral deviation" state variable becomes larger than half the value of the "distance between rows" state variable. When this happens, the row counter is incremented (or decremented, depending on whether the rows are to the left or to the right of the robot), and the "lateral deviation" state variable of each particle is incremented (or decremented) by the value of "distance between rows".
When the robot has traversed the headland, made the turn into the new row, and is about to start following the new row, its belief is still expressed in terms of the previous row. For example, its heading is approximately 180 (or -180) degrees. Before the robot starts to follow the new row, the state variables are normalized so that heading is approximately 0 degrees.
In Task 3 an obstacle is placed in the first row. This obstacle is detected by considering the returns from the laser scanner in a narrow field of view in front of the robot. Only returns from distances between 0.10 and 0.50 m are considered. With a sampling frequency of 10 Hz, a counter of hits is kept. The counter is incremented when one or more laser beams hit an object, and it is decremented when no returns are received; the counter is not allowed to become lower than zero. An obstacle is considered to be present when the counter reaches a value of three. At this point the robot reverses out of the row, traverses the headland to the next row, and starts following the next row.

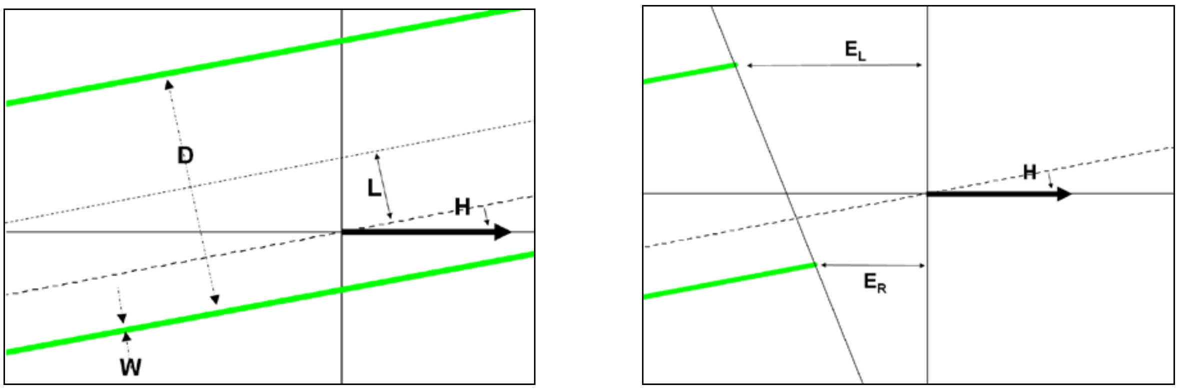


Figure 3. Model of the world in which the robot travels. Green lines represent rows of maize. Symbols are explained in Table 1. Left: the robot is between the rows. Right: the robot has entered the head-land.

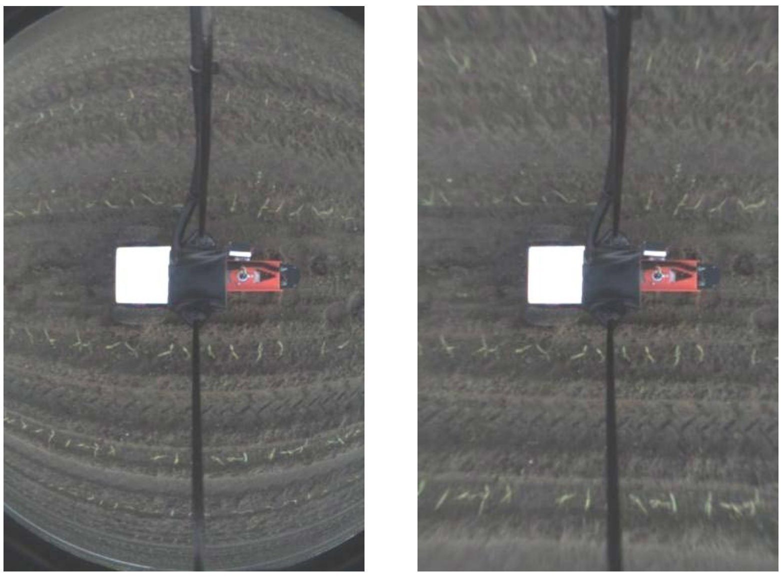


Figure 4. Image from the camera. Left: original image. Right: undistorted image.

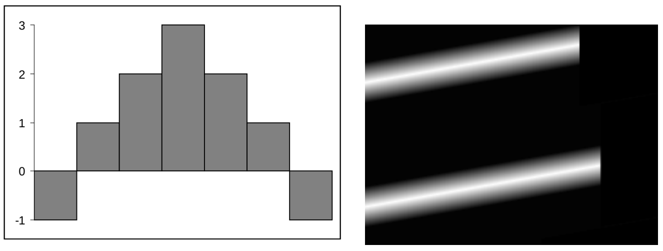


Figure 5. (A) Cross-section of a row, showing that pixels in the middle have a large weight, pixels toward the edges a lower weight, and pixels that are far from the middle have a negative weight. (B) Weight image resulting from the belief that H = 10 degrees, L = 0.1 m, D = 0.75 m, W = 0.25 m, and E_L = E_R = 0.5 m

3. Navigation

Row following: The robot needs to navigate in two distinct situations: between the rows, and on the headland (Fig. 8). While between the rows, the robot follows the line that lies in the middle of the path between two rows of maize. While on the headland, the robot follows a line that parallels the imaginary line that can be drawn between the ends of the maize rows and that lies at a given distance from the ends of the maize rows. Of course, these two situations reduce to one, namely following a straight line, where the variables to be controlled are the distance from the control point to the line that is being followed, and the angle between the robot's current path and the line that is being followed.
The controller used is:

```
C = -H + atan(-L)
```

where C = the steering angle setpoint of the front wheel.
Headland following: The robot continues following the rows until it has cleared the rows by a preset distance. Then it comes to a full stop, makes an on-the-spot turn to position the robot parallel to the new path, and follows the new path. When it has reached the middle of the new row, it comes again to a full stop, makes an on-the-spot turn, and starts following the rows again.
Obstacle: If an obstacle is detected during the first few meters of a new row, the robot backs out and returns to its last position outside the row, and then travels on the headland to the next row.

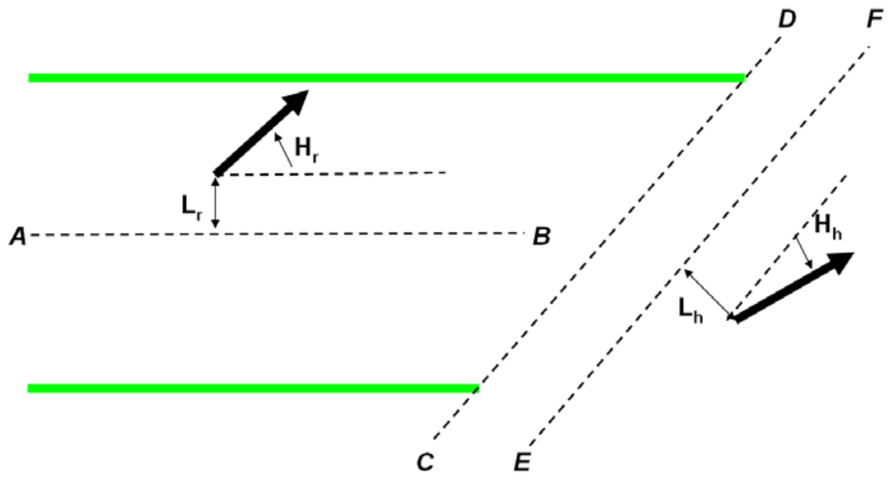


Figure 6.Schematic representation of row following between the rows and on the headland. See text.

4. Weed detection and spraying

The artificial weeds consist of patches of green plastic turf with small white flowers. Weed detection comprises two threshold steps. The first of these steps detects the green turf, the second step detects the white flowers. Whenever a white flower inside a green patch is detected, the sprayer is activated (Fig. 9). The position of the white flower determines whether the inside nozzle or the outside nozzle is activated.

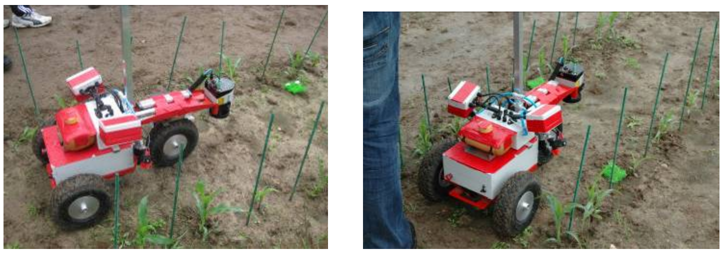


Figure 7. Close-up of spraying module.

5. Results, discussion and conclusion

At the Field Robot Event, the robot was able to follow rows and turn into the correct new row in all tasks. No manual intervention was necessary. This means that the most important objective was met.

The fact that no interventions were necessary was possible in part because the robot travelled rather slowly, only 0.7 m/s. In addition, robustness was increased by adding delays of several seconds at the end of the row and during turning. This had a negative impact on the total distance covered by the robot and on the overall performance during the Event.

Detection of the obstacle in task 3, backing out of the row, and continuing in the next row, was executed perfectly.

The flower detection turned out to be more sensitive to the light conditions than we had realized. The camera was calibrated early in the day; when the sky was much more overcast during the Event, detection did not work as well as intended. What also played a role that the turf patches were placed almost between the maize plants in stead of well inside the row, and were out of the camera's view.

At the Event, the maize plants were fairly large. But in the days and weeks leading up to the event, the robot was tested while the maize plants were much smaller. It was shown that navigation is perfectly possible with plants between 5 and 10 cm (see http://www.youtube.com/watch?v=GUgEHlZHIJ4). Thus, the second objective was met.

In conclusion, the robot is capable of a high degree of autonomy in the tasks of the Field Robot Event. It didn't once get lost and it damaged few plants. Navigation is possible with small plants and with large plants.

Acknowledgments

We are grateful to Sick Nederland B.V. for providing us with a laser scanner.

References

Hofstee J.W., Jansen R., Nieuwenhuizen A.T., Govers M.H.A.M., Blaauw S.K., Van Willigenburg L.G., Stigter J.D., Bakker T., van Asselt C.J., van Straten G., Speetjens S.L. (2007) WURking - An autonomous robot for agriculture applications, Proceedings of the 5th Field Robot Event 2007, Farm Technology Group, Wageningen UR, Wageningen. pp. 89-96.

Polder G., Van Evert F.K., Lamaker A., De Jong A., Van der Heijden G.W.A.M., Lotz L.A.P., Van der Zalm T., Kempenaar C. (2007) Weed detection using textural image analysis (Available online at http://library.wur.nl/file/wurpubs/wurpublikatie_i00363383_001.pdf), 6th Biennial Conference of the European Federation of IT in Agriculture (EFITA), Glasgow.

Thrun S., Burgard W., Fox D. (2005) Probabilistic robotics MIT Press.

Van der Heijden G.W.A.M., Van Evert F.K., Lamaker A. (2008) Probabilistic robotics in an autonomous field robot. In: The 7th French-Danish Workshop on Statistics and Image Analysis in Biology, Toulouse, France, 13 - 16 May, 2008.

Van Evert F., Lamaker A., Van Evert K., Van der Heijden G., De Jong A., Lamaker J., Lotz B., Neeteson J., Schut T. (2004) Een grote toekomst voor kleine robots in open teelten. Agro-Informatica 17:21-26.

Van Evert F.K., Polder G., Van der Heijden G.W.A.M., Kempenaar C., Lotz L.A.P. (2009) Real-time, vision-based detection of *Rumex obtusifolius* L. in grassland. Weed Research 49:164-174.

Van Evert F.K., Lamaker E.J.J., Polder G., De Jong A., Van der Heijden G.W.A.M., Groendijk E.J.K., Lotz L.A.P., Van der Zalm T. (2008) Hielke is a Bayesian, balancing field robot, Proceedings of the 5th Field Robot Event 2007 (Available online at http://library.wur.nl/way/bestanden/clc/1868835.pdf), Farm Technology Group, Wageningen. pp. 45-53.

Van Evert F.K., Van der Heijden G.W.A.M., Lotz L.A.P., Polder G., Lamaker A., De Jong A., Kuyper M.C., Groendijk E.J.K., Neeteson J., J. , Van der Zalm T. (2006) A mobile field robot with vision-based detection of volunteer potato plants in a corn crop. Weed Technology 20:853-861.

Van Evert F.K., Samsom J., Polder G., Vijn M.P., Van Dooren H.J.C., Lamaker A., Van der Heijden G.W.A.M., Kempenaar C., Van der Zalm T., Lotz L.A.P. (2010) Herkenning en bestrijding van ridderzuring met een robot. Gewasbescherming 41:15 - 16.

Turtle Beetle

Juho Pentikäinen (captain), Mikko Pihlanko, Pekka Pihlanko, Juha Helenius, Eero Tuhkanen, Mika Matilainen, Milla Sairanen, Timo Oksanen (instructor), Jari Kostamo (instructor), Petro Tamminen (instructor)

Aalto University, Department of Automation and Systems Technology,
Aalto University, Department of Engineering Design and Production,
University of Helsinki, Department of Agricultural Sciences

http://autsys.tkk.fi/en/FieldRobot2010
email: jpentika@cc.hut.fi
P.O. Box 11000
FI-00076 AALTO
FINLAND

Abstract

Turtle Beetle is a robot build by the students of Aalto University School of Science and Technology for the Field Robot Event 2010, in Braunschweig, Germany. The robot was designed to navigate between maize rows, identify weeds, handle them effectively and sow seeds in the gaps in the row. This paper describes the building process and used methods for controlling the device. This was the sixth time when Aalto University (the former Helsinki University of Technology) students participated in the contest. In the robot design a clear influence from earlier robots can be seen. However, everything is build and coded from a scratch. The original feature of the new robot design was a pneumatic system, which was used for active suspension, weed destroying and seeding. Also a machine vision as the main navigation method was replaced with a laser scanner. To increase modularity from previous years, most of the electronics was built on a separate plate (plexiglass) that was easy to install on the robot when the chassis was completed. Building the electronics on separate plate made it possible to start the testing of electronic assembly in parallel with mechanical construction.

The planning of the robot was started in September 2009, the actual building started in January 2010 and robot was mechanically and electrically finished in May 2010. The software was developed in test environment in the very same time. Test environment consisted of Matlab simulator and the plexiglass.

The robot was built by the team including design and machining. The completely self-made software was built with Matlab/Simulink, Visual Studio, CodeVisionAVR and LabVIEW -software.

The robot participated in Field Robot Event 2010 in June 2010. In overall competition, the Turtle Beetle robot gained the third prize, and in freestyle competition the second prize.

Keywords: Robot, Maize Field, Navigation, Machine Vision, Suspension System

1. Introduction

Turtle Beetle is a robot built for Field Robot Event 2010 by Aalto University School of Science and Technology students. Aalto University (Former Helsinki University of Technology) students have participated in the event for several years. However, the robot is build from a scratch every year and previous results are not used directly. Most of the ideas for this year's robot can be found in the previous robot "Easywheels" (Kemppainen et al, 2009), but for example all the program code redesigned by the students and basically all the machining and electrification is made by the students. However, building a robot and its program code is quite a complicated task and advice and support from the instructors have been essential for completing the task.

The robot's main task is to navigate between the maize rows, identify weeds, handle them effectively and resow seeds to gaps in the row. The robot's main navigation uses a laser scanner, ultrasonic and infrared distance sensors. There are two computers onboard, one for machine vision and the other for the main program. In addition to two computers, up to four microcontrollers are used onboard; two of them drive the motors, the third connects sensors, user interface and controls pneumatics, and the fourth computed inclination.

Also the mechanical system is quite a complicated. The vehicle is a four-wheel-drive system with four wheel steering. Suspension is made especially to keep all the wheels on the surface to avoid slipping and to keep chassis as vertical as possible. The mechanical suspension was improved by adding an active pneumatic roll compensation system.

Most of the differences between the concepts of the previous years and this year are: using a laser scanner for navigation, improved axle module design, improved electrification, semi-active suspension, pneumatically working weed destroying tool, pneumatically working patch seeding tool and improved overall design. Totally new ideas are the use of pneumatics for weed destruction, the seeder and the platform auto balancing system which keeps the robot leveled relative to ground.

Also LabVIEW, which was used for machine vision development, was introduced as a new programming language and tool to the project. The main program was build with C# in Visual Studio compared with C++ in previous year. C# was used in the earlier robot, but this was the first time to use C# together with Windows Embedded CE.

The planning of the robot started in September 2009, actual building started in January 2010 and robot was mechanically and electrically finished in May 2010. The software was developed in test environment in the very same time. Test environment consisted of Matlab simulator and "plexiglass".

Most of the robot was built by the team including design and machining. The completely self-made software was built with Matlab/Simulink, Visual Studio, Codevision and Lab-VIEW -software.

The main mechanical design goal was to construct the robot from modules. The modular structure contains the frame and suspension, axle modules, agricultural machinery, the platform for electronics, and the body.

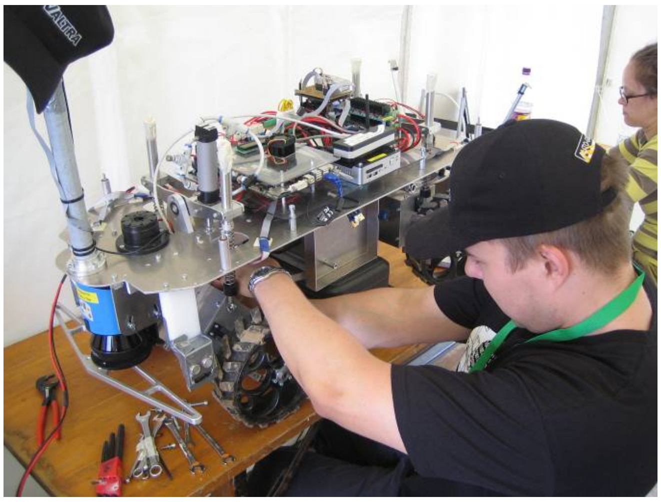


Figure 29: Building up a robot

The modular structure held several benefits. In the design phase the CAD-design of the modules was divided with the team members and also the assembly of the modules could be done independently. The modular structure is very practical in everyday use and servic-ing the robot, because if a module breaks down it can be easily replaced by a new one or fixed without disassembling the whole robot.

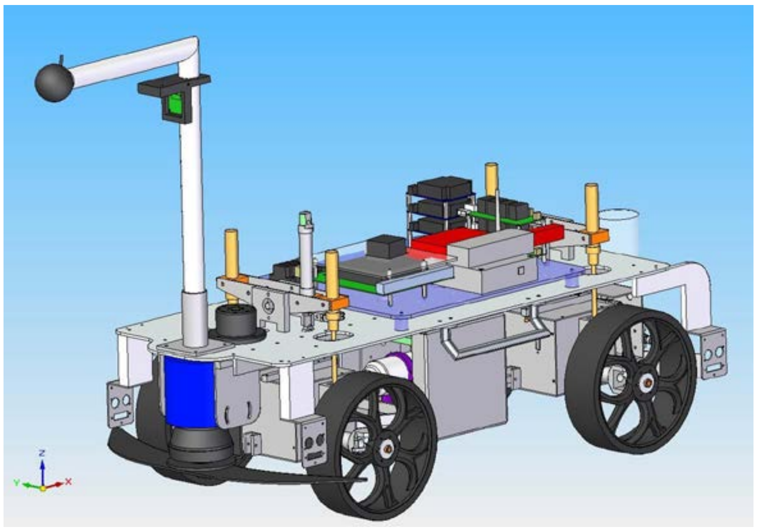
All the metal parts of the robot are made from aluminium, because of its light weight and cost quality ratio. The sheet metal parts were made by laser cutting (Laserle) and the other parts were machined by the team members.

2. Mechanical structure

Frame

All the mechanical parts were modelled with Solid Edge v20, besides the cover which was designed with Pro/ENGINEER.

The main design aim for suspension was to engineer a pneumatic anti roll bar. There is an air compressor installed inside the frame. The compressor keeps the pressure of the air reservoir in the range of 5...8 bars. The air compressor is controlled with a microcontroller, which regulates the pressure by using a pressure sensors installed in the air reservoir. A speciality of the suspension is that the upper ends of the shock absorbers are connected to moving swings. The front swing angle can be pneumatically adjusted in order to keep the robot's frame in a horizontal position. The front swing can be locked if the roll compensation is not needed.



The pneumatic equipment used in the robot is provided by the FESTO company. In Figure 30 all the pneumatic valves and the suspension absorbers are shown.

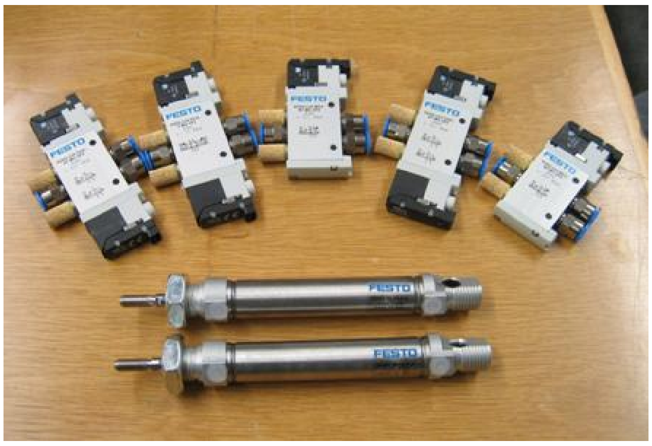


Figure 30:Pneumatic valves and suspension absorbers from one of our sponsors FESTO

Axle Module

In the preceding robot, “EasyWheels”, everything required to drive the robot was encapsulated to “axle modules”. For EasyWheels three identical axle modules were constructed, one to act as a front axle, the other as a rear axle and the third one was a spare. The idea was to pack all the required components for steering and driving to one module, and this was reported to be such a good solution that the same concept was selected for Turtle Beetle. The modular structure was found convenient for varying the construction of the EasyWheels robot with different tasks. The modular structure also enabled a fast recovery from a malfunction situation, because the whole axle module could be replaced with a spare one in a few minutes. Hence the EasyWheels robot could carry out its tasks on the field without long repair periods. (Kemppainen et al, 2009)

The drive train of the axle module is illustrated in Figure 31. The red ellipse encloses a Tamiya sport tuned DC-motor, which produces the torque for the wheels. Inside the blue ellipse is the planetary gear box (16:1 ratio) and in front of the gear box is a clutch bell (yellow ellipse). Differential is enclosed in a green ellipse and obviously the drive shafts

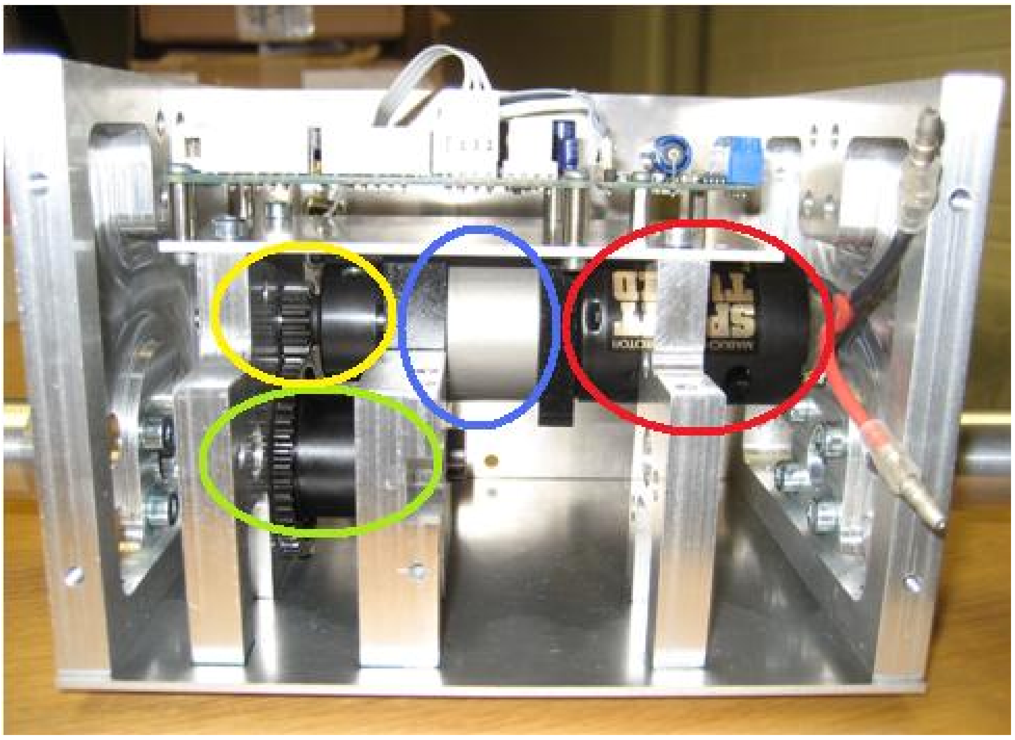


Figure 31:Drive train

are connected to it although they are not present in Figure 31. The clutch bell, differential and drive shafts come from Xray RC model car.

The power for the steering system is produced with a Hitec HS-7954SH RC-servo (enclosed by a red rectangle in Figure 32). The steering torque of the RC-servo is delivered to the steering blocks (enclosed by a blue rectangle in Figure 32) via a steel cable. Inside the yellow rectangle are a pulley and a potentiometer, which are attached to the RC-servo's output axis. The steel cable circulates the pulley and the ends of the steel cable are fastened to the steering blocks. The angle of the RC-servo is determined with the potentiometer, which is needed for the PI-control of the steering system. The DC-motor is also PI-controlled and the angular velocity data of the motor is provided for the micro controller by an optical rotary encoder.

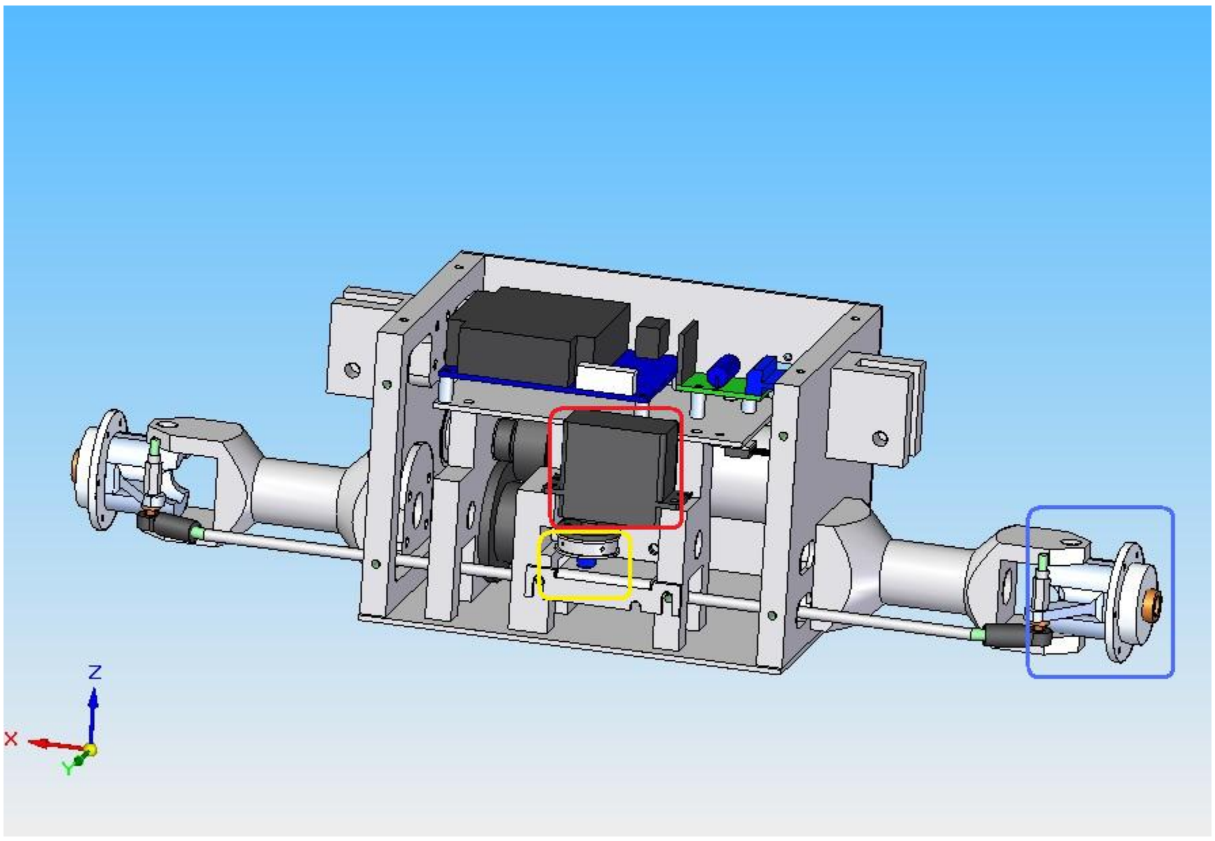


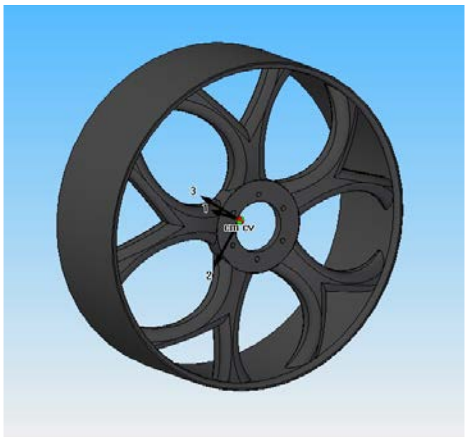
Figure 32:CAD image of axle module components

The objective with the axle module of the Turtle Beetle was to improve the issues that came up with the axle module of EasyWheels. The main improvement goal was to reduce the weight of the axle module and therefore reduce the unwanted non-suspended mass of the whole robot. The previous self-made motor driver was also changed to a much smaller commercial one. The steering blocks of EasyWheels came from a RC-vehicle and they already had a certain amount of camber angle. Thus the steering blocks had to be modified so that the camber angle was zero. Because of this problem, the steering blocks of Turtle Beetle were designed and made by the team members.

Wheels and Tires

The wheels were designed by the team and carved from a large cylinder of plastic (Figure 33).

The plastic cylinder has cut to eight equal pieces. The work pieces have been carved first inside. Then we put a cylinder made from steel inside to support the fixing for outside carving which has made in two parts. The cylindrical part has carved first. After flipping the piece and putting the support ring the face has carved with the same tool. After turning operations the rim was almost finished but we decided to mill some spool shapes to achieve good-looking and lighter rims. The piece was fastened in the chuck which was installed in the mill table. Then the shapes has milled by using three dimensional tool paths. The surface of the tire has cut from the big RC- truck car tire. Only 60 mm wide strip was needed. Strip has cut with the knife tool. The rubber was fixed on the wooden support block in the manual lathe. Then carefully turning the spindle by hand the cutting was possible. The strips has glued to the rim with the adhesive.



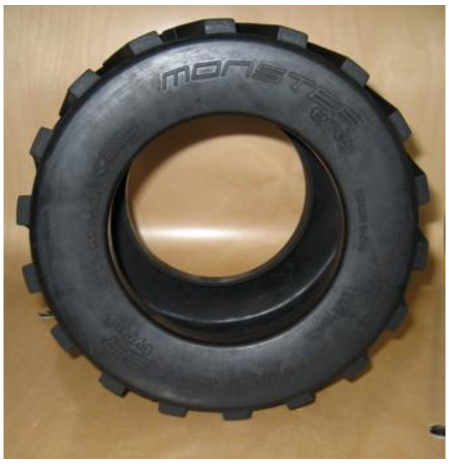


Figure 33:CAD image of axle module components

Agricultural Machinery

Two agricultural machines were designed for the robot in order to take part in weed destruction and freestyle tasks. The machines can easily be attached to the rear axle module with three bolts. They also have quick couplings for pneumatic and electrical connections. The compressed air comes directly from the robot's pneumatic system and valves and solenoids are driven by three Opto Output amplifiers, which are in turn controlled by a microcontroller on the plexiglass.

The seeder

The seeder was created for the freestyle task of the competition. The idea was that when a gap was detected in a maize row, the robot would stop and sow a seed in place of the missing plant. The idea was similar like in 4M (Backman et al, 2008). The seeds are actually airsoft gun pellets, which are put into vertical pipes (Figure 34, the red ellipse). Solenoids (yellow ellipse) are used to drop the seeds one by one into elastic tubes (not shown in Figure 34) which are connected to longer pipes (blue ellipse). When the seed finally drops into the longer pipe, the pneumatic cylinder (green ellipse) pushes the pipe downwards and the seed is blown out by compressed air. The seeder has three pneumatic valves (black ellipse): two 5/2 valves to control the cylinders and one 2x3/2 normally closed valve to control airflow into the long pipes.

It was expected that the weakest link of the seeder might be the solenoid system designed to drop the seeds and actual tests proved this to be true. The solenoids were not powerful enough to compress return springs that hold slide closed and the seeds won't drop. This problem can be corrected with more powerful solenoids. On the other hand, the pneumatics work well which is the most important thing in the freestyle task because moving cylinders and whistling valves give the judge and audience a feeling that something cool is happening.

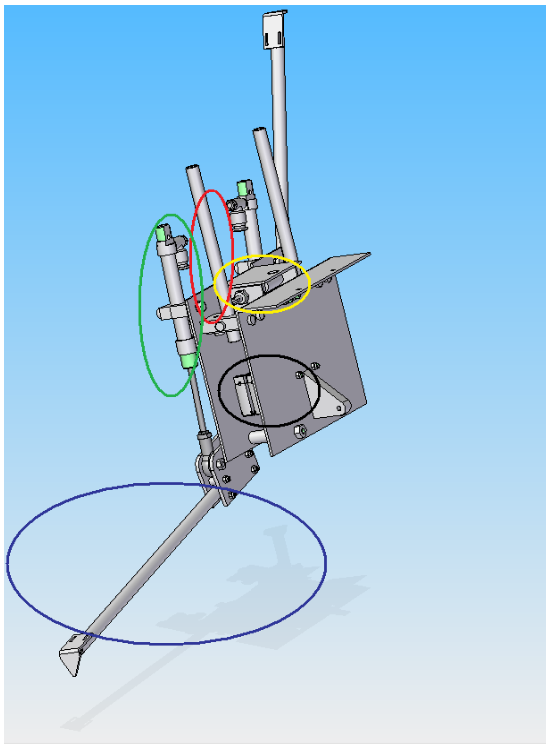


Figure 34: The seeder

The sprayer

The sprayer is to be the weeds' worst nightmare. When the robot detects a weed in the maize field with machine vision, it stops and sprays herbicide (water) to kill the weed. The operating principle of the sprayer machine is very simple. The bottle contains water and is connected to two vacuum generators (Figure 35, blue ellipse) with pneumatic tubes. One 2x3/2 normally closed pneumatic valve (red ellipse) is used to control airflow to the vacuum generators. When the valve is opened, vacuum sucks water from the bottle to the vacuum generator where it is mixed with air and finally sprayed to the side of the robot. The vacuum generator is not meant to be used for this purpose, but seems to work nicely. The amount of water can be adjusted by flow control valves which are shown in Figure 36.

Thanks to its simple structure, the sprayer performed well in tests. The only problem was that water flew spontaneously out through vacuum generators due to gravity. Fortunately, we were able to restrict the leak by adjusting the flow control valves.

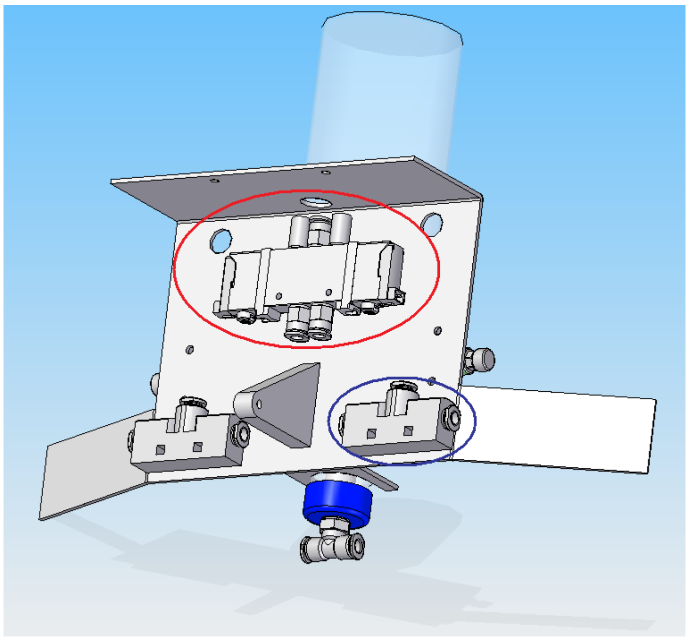


Figure 35: CAD image of the sprayer

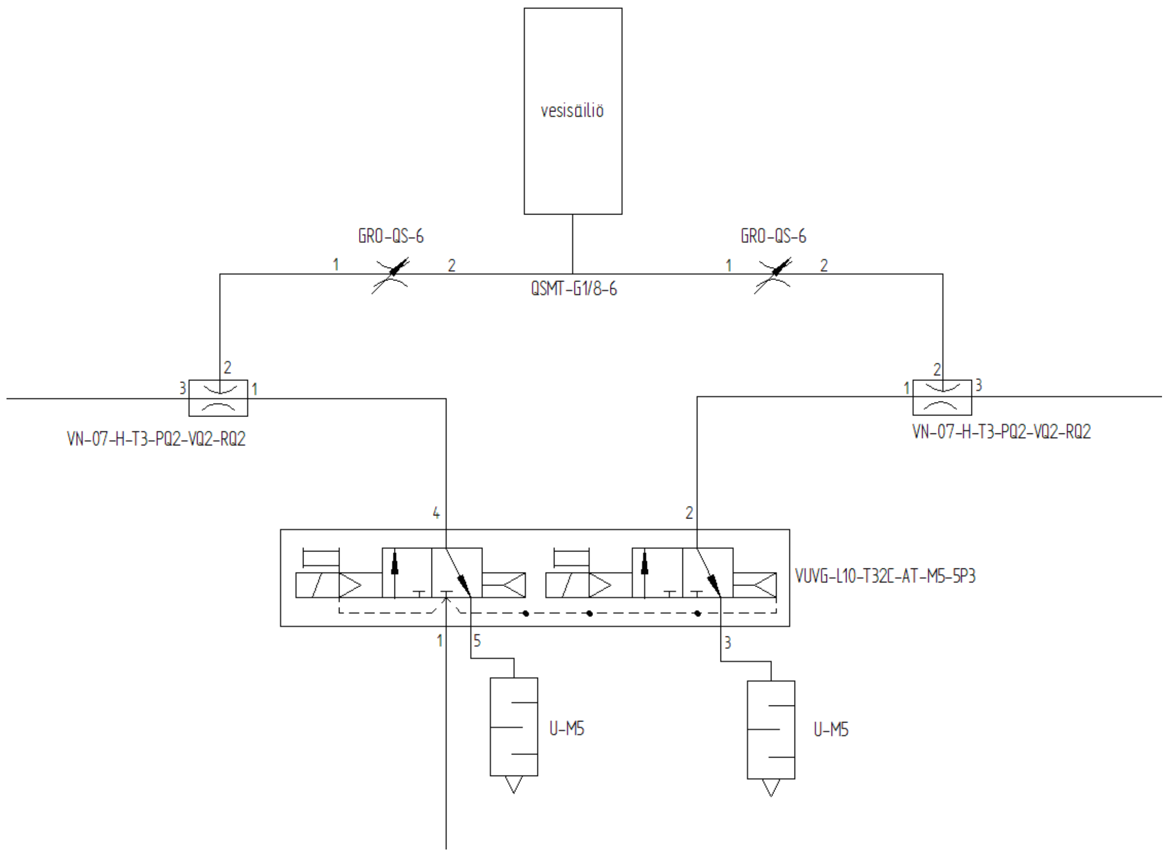


Figure 36:The sprayer's pneumatic diagram

Cover

The cover for preceding robot “EasyWheels” was designed and manufactured by a sponsor, and this was not the fact in case of Turtle Beetle. The idea to have nice looking, single piece cover was decided. Solid Edge was used use design all the robot mechanical parts etc. but the software does not suit for designing smooth surfaces, like Turtle Beetle has. Fortunately, an industrial designer Pekka Kumpula from S.E.O.S design Ltd, helped us by designing the cover outlook with Pro/ENGINEER. Industrial designer had an idea that usually robots have many bugs inside, so the design was to be a Big Bug (Beetle is one of many species of bugs).

The shape of the body has made with vacuum forming, which is a process whereby a sheet of plastic is heated to a forming temperature, stretched onto a single-surface mold, and held against the mold by applying vacuum between the mold surface and the sheet. The mold has milled to the SIKA polyurethane tooling board. The mold was milled by a team member, but the team got some help for vacuum forming from a local workshop. The sheet was 3mm thick VIVAK (Polyethylene terephthalate). Formed sheet has trimmed and the body has painted inside to make a shiny finish.

3 Electronics

Computers

The robot has two computers onboard (Figure 37). ICOP eBox-4300 minicomputer with 500 MHz VIA Eden processor and 512 MB DDR2 is for the main program (“the brains”) and Toradex Woodpecker with 1.6 GHz Atom processor (Z530) and 1024 MB of DDR2 is only for weed detection (“machine vision”).

The main program loop runs in eBox which is attached to three microcontrollers in total with serial ports and to a WLAN router with RJ45 cable. One of the serial ports is made with USB-serial converter. Router is used for communicating between eBox and Toradex Woodpecker.

The router is also used to connect a laptop to eBox or Toradex Woodpecker. This can be used for the remote desktop connection or the remote user interface (Remote UI) which was made for the testing and adjustment of the robot. Also updating the software to eBox goes over WLAN. However, the robot can move independently without any laptop. Communication via router was built with UDP-protocol for simplification reasons.

Woodpecker uses 12V voltage which is the same as used the robot's electrical system. However, eBox uses 5V voltage like the WLAN router so there is a DC/DC converter onboard (Texas Instruments TPS5430 in evaluation board).

eBox has Windows Embedded CE 6.0 installed as operating system and Toradex Woodpecker has a regular Windows XP. Both worked well for the competition and for learning purposes. eBox should have had somewhat more computing power, as it limited developing the algorithms partially. Perhaps structure and coding would have been more simplifier if both computers would have used the same operating system for example Windows XP Embedded.

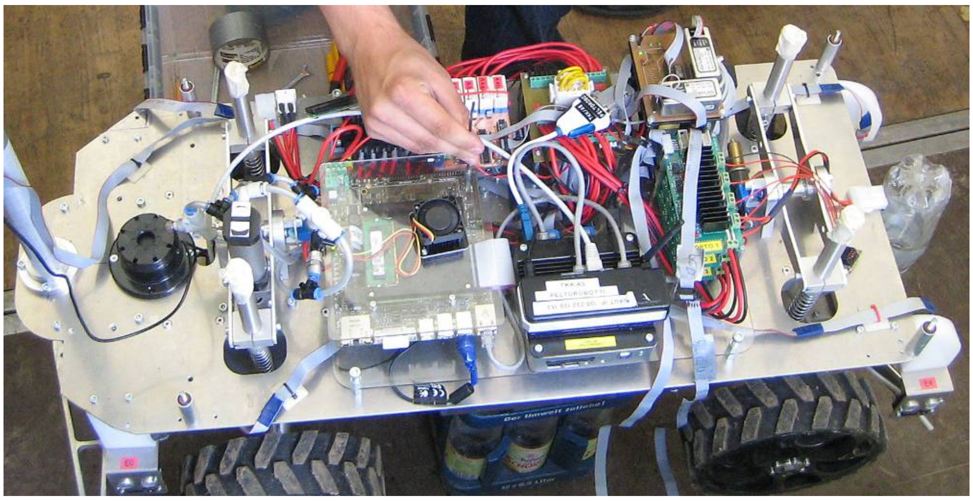


Figure 37:Robots main electronics

Controllers

Turtle Beetle has four Atmel ATmega128 microcontrollers on Futurlec ATMega proto boards (Figure 38), which were used also in 4M (Backman et al, 2008) and in EasyWheels (Kemppainen et al, 2009) and were reported being good. Front and rear axle modules have their own controllers for motor and servo control and there are also two controllers on the plexiglas. One of them calculates robot's inclination angles (roll and pitch) from the information about inclinometers and gyroscopes, the other is used to read sensors (ultra-sonic and infrared range finders, compasses), to control pneumatics and to read and write

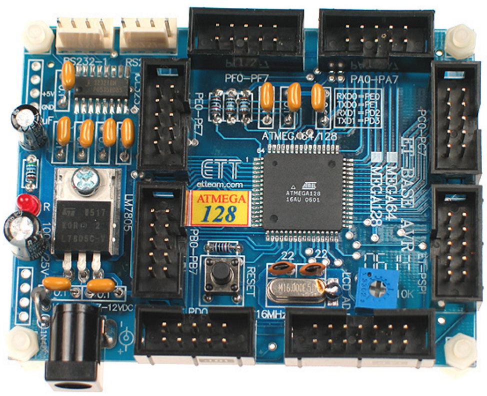


Figure 38: Futurlec ATMega proto board

local user interface I/O. Control signals for pneumatic valves, a buzzer and a compressor relay are amplified in three Opto Output amplifiers. User interface includes LCD display and eight LEDs and buttons, which are controlled by two Microchip MCP23S08 GPIO expanders in SPI bus.

Motor drivers

The motor is driven with PWM signal which is amplified in Pololu 18v15 motor driver (Figure 39). For the motor driver two inputs are required: the direction as a bit, and the PWM signal to be amplified. Controller measures speed with the Avago Technologies optical rotary encoder and uses the feedback to keep the desired speed if RPM value is set. The PWM output is then obtained by a PI controller. The servo position can also be set by PWM or desired angle. Angle feedback is given by potentiometer that is connected directly to servo’s axle.

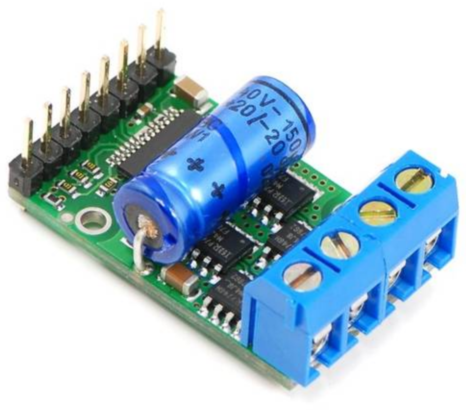


Figure 39: Pololu 18v15 motor driver

The very first field tests revealed a very serious problem with the motor drivers. The big blue capacitor (in Figure 39) easily overheated and blew up under normal driving conditions. The robot had then to be stopped immediately to avoid more damage. This happened a few times but we couldn’t find any good reason, since the motor drivers were used according to manufacturer’s instructions and specs. The solution was to replace the original capacitor with more heat resistant tantalum one and reduce the PWM frequency remarkably (from 16 kHz to 2 kHz), which prevented the heat from rising too high. Besides a tantalum capacitor, a much bigger electrolyte capacitor was installed in parallel. These actions significantly improved reliability, but actually we were able to blow up also one tantalum capacitor in the event test field, just a day before competition.

Sensor interfaces

Infrared range finders' output is analog voltage (0-5V) which is read by the controller and converted to distance by eBox. Communication with SRF08 ultrasonic range finders and CMPS03 compasses is via I^{2}C bus. Each sensor in I^{2}C bus has a unique address to separate it from other devices. The controller is the bus master that commands sensors and reads measurement data; sensors are slave devices. The I^{2}C bus is relatively easy to use, but it seemed to be a bit unrobust. For example, if one device stopped working because of a broken cable, the whole bus died.

Communication

Communication between controllers and eBox is via RS-232 serial bus (19200 bps, 8N1). A special protocol was developed for the communication. Data is always sent in one frame, which starts with 0xFF followed by a unique device ID. The next bytes are actual data. After data bytes comes a checksum byte, which is calculated from data bytes and used to prevent errors in communication. Controllers and eBox both calculate the checksum for each message and if there is a mismatch, the whole message is rejected. The frame ends with bytes 0xFF, 0xEE. All 0xFF bytes in the data are replaced with two bytes (0xFF, 0x00). All the measurement data is sent to eBox for further analysis; eBox sends back parameter and set point values, like PI controller parameters or desired speed.

Batteries

The main idea with the batteries was to use equal size batteries which are easily changeable. In addition there was a goal to make the robot lighter than earlier years so it was decided to use lithium polymer batteries (LI-PO). These batteries are quite light compared with lead acid batteries, but there are some differences. One of the most important thing is that battery voltage must be between 3.0 and 4.2 volts/cell. So it is needed to use a cutoff circuit to prevent discharging the battery voltage too low. The charger must also be "intelligent". The charger starts charging with constant current set by user and when the voltage reaches 4.2 volts per cell it starts charging with constant voltage. Then batteries are about 95-90% full. The last 10-15% takes time about the 1/3 of the full load charging time so it is often useful to end charging if time matters. LIPOs are also very vulnerable; there is only a very thin layer of plastic film over the battery core. Misuse of batteries can cause a fire.

Turtle Beetle uses four batteries (3 Cell 5000mAh, Figure 40) placed in the cases (two packs in each case). In total two sets of batteries (8 pieces) were bought, so it is possible to charge one set while running with the others. So it has the energy to make a one-hour

Figure 40: Batteryback

continuous trip on the field (or to make a huge LIPO fire). The nominal voltage of each battery was 11.1V while the legal usage voltage is from 9.0 to 12.6V.

Battery Mode Switch

The electrical devices of the robot were divided into different power circuits. The purpose of this was that we could get different battery switch modes which are Racing mode, Debug mode, and External power supply mode

The battery switching can be done with a single circuit board, which has input to batteries and outputs for the power circuits. The circuit board holds a 15-pole socket in the middle (Figure 41). All four batteries and the power circuits have their own connectors in this 15-pole socket. By chancing a differently wired plug to this socket we can assign the batteries to different power circuits.

In racing mode the front - and the rear axle modules have their own batteries for DC-motor and RC-servo power supply. In debug mode a single battery is supplying power for both axle motors and therefore more power can be supplied for the computers and micro controllers. In the external power supply mode we can use external power supply for running all of the electrical equipments.

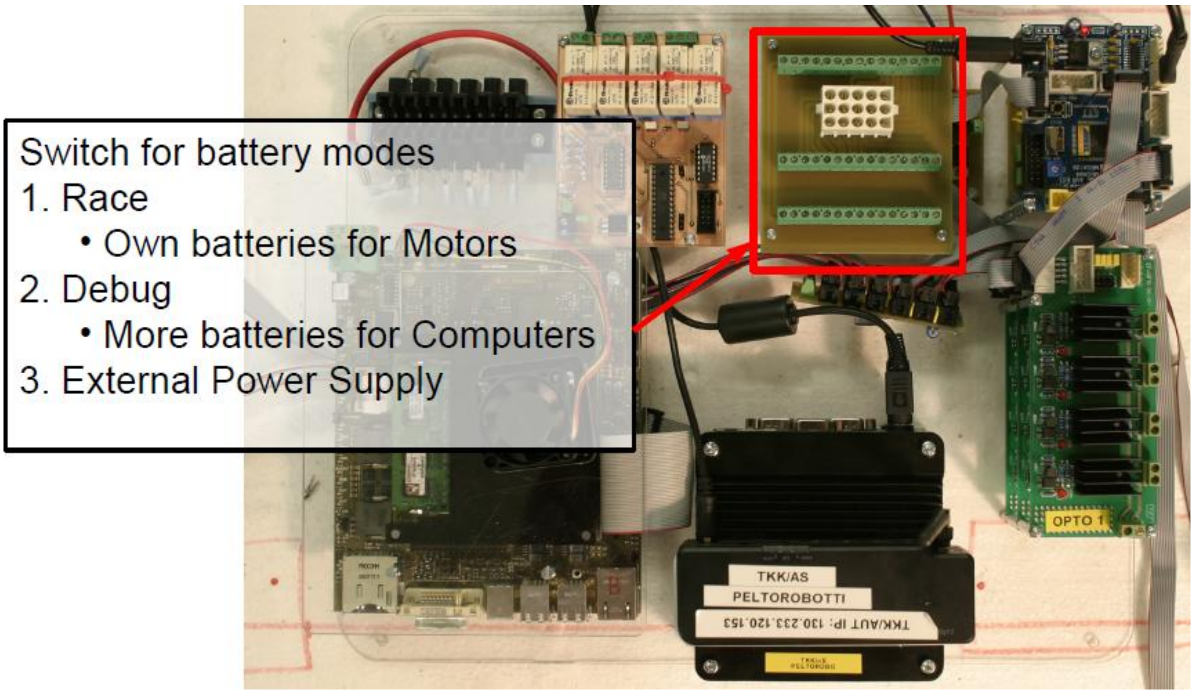


Figure 41: Battery switch modes

Sensors

The robot has four infrared sensors and four ultrasound sensors, a SICK laser scanner, two compasses installed at 90 degrees angle relative to each other for 3D information, combined gyroscope and 3-axis accelerometer, and an ordinary web camera for weed detection.

Inclinometer and compass

The robot has two 2D electronical compasses (CMPS03) and two gyroscope-accelerometers combos (VTI SC-1120) for taking the accurate measurements of the direction (Figure 42). The idea is that gyroscopes and accelerometers are used as an inclinometer, which gives the roll and pitch of the robot. Compasses are very sensitive to tilt because the measured component of the Earth's magnetic field is measured in the wrong angle. For this reason it was necessary to compensate the compass error with inclination.

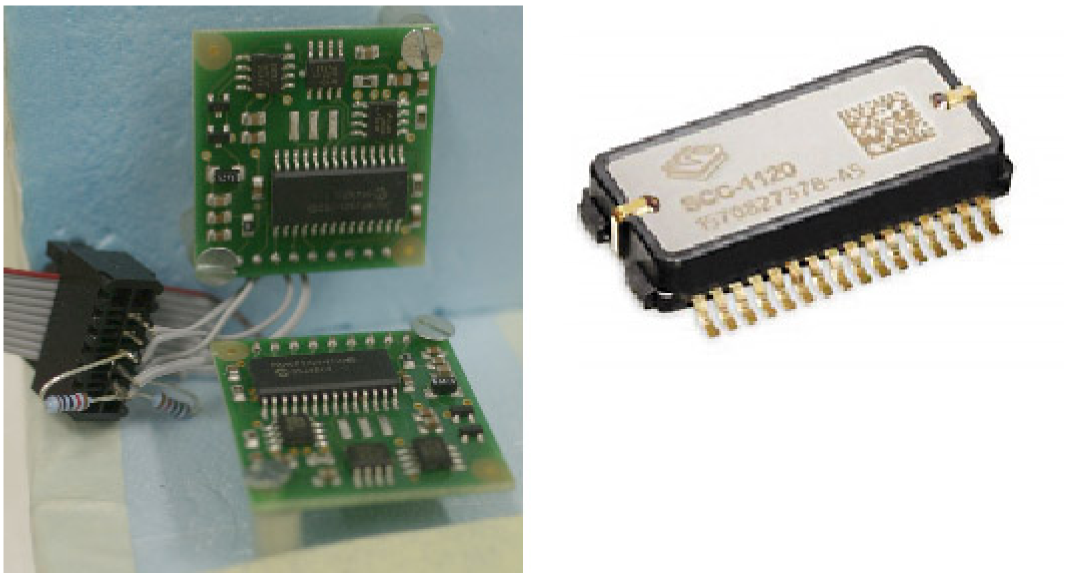


Figure 42: Compass and gyroscope

The compass had to be calibrated carefully. Together we had four compass measurements which were components of Earth’s magnetic field. The compass angle could be calculated directly as a tangential angle of the components. However, this measurement could lead to very odd compass angles without calibration, because measurements were not located around origin as presented in Figure 43. The easiest way for calibration was to roll the robot on the floor and draw measurements as Matlab graph. This had to be done in two steps to get the errors of the measurements in x, y and z direction. In the first step, the robot was standing on its wheels while turning it 360°. In the second step, the robot had to lie on its other side while turning it again 360°.

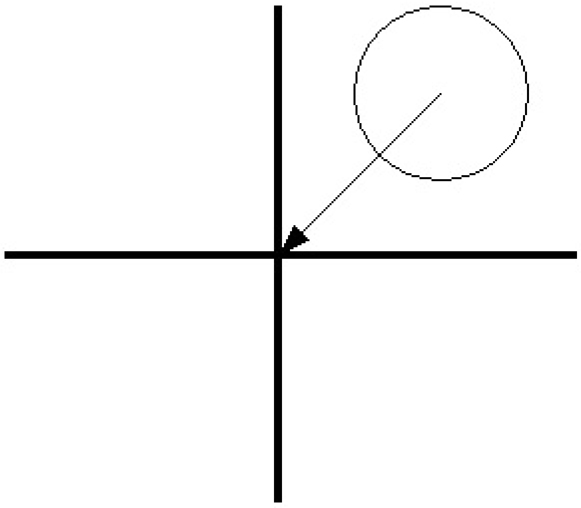


Figure 43: Compass calibration

Ultrasonic and infrared sensors

SRF08 ultrasonic range finders are located in each corner of the robot and measure distance from maize plants. The finders' maximum measuring distance was set to 0.60 m which is enough for operating in a maize field. Longer distance would give echoes from farther rows and from ground. Ultrasonic range finders are fired in every 50 ms and there is a 24 ms phase difference between front and rear sensors in order to prevent false echoes. Front sensors are fired and read before rear sensors. The SRF08 sensors give distance directly in centimeters.

In addition, the robot has four SHARP GP2D12 infrared range finders located below each SRF08 (Figure 44). These sensors give an analog voltage output which can be converted to distance. The maximum practical measuring distance is about 30 cm.

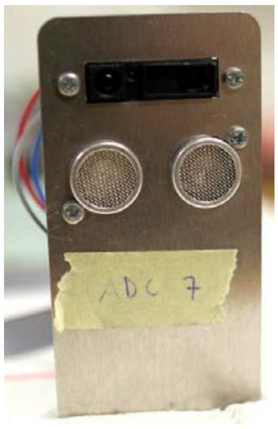


Figure 44: Ultrasonic and infrared module (upside-down)

Laser scanner

Sick LMS100-1000 laser scanner (Figure 45) is located in front of the robot. It is attached upside-down to make sure that laser beam will hit small plants. The scan angle of LMS100-1000 is 270º but only 200º was used because of the geometry of the robot. Updating frequency was set to 50Hz and angular resolution to 0.5º. The scanner is connected directly to the router and communication is based on TCP protocol.

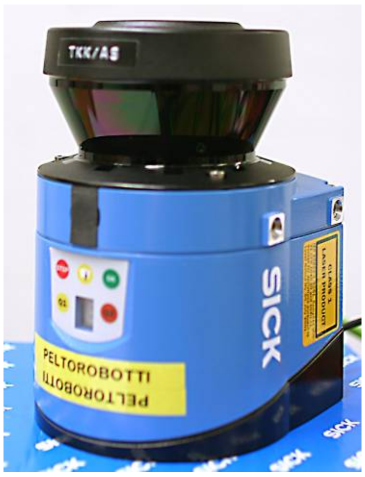


Figure 45: Sick LMS100-1000 laser scanner

The manufacturer has announced that distance measuring range is from 0,5m to 20m but we noticed that it is possible to measure distances as short as few centimeters. Maximum measuring range used was 2 meters.

Web camera

For the weed detection, a mid price web camera was used. It had been proven successful with previous robots (Backman, J. et al.), was relatively easy to obtain and use, and did not cost much. A web camera uses USB 2.0 for data transfer. Theoretically, it is faster than Firewire but in practice slower (FireWire). However, for this application a web camera using USB 2.0 was good enough as the main bottleneck was the processing power of the computer.

The chosen web camera was Logitech QuickCam Communicate Deluxe (product number 961465-0509) by Logitech Inc. It claimed to use a CCD image sensor but we were not so sure about it. Another important characteristic was the “extra wide” field of view, which was about 60 degrees. The wide field of view means that the camera can be positioned lower than otherwise. Logitech QuickCam Communicate Deluxe also uses software to automatically adjust to lighting conditions although that property was not tested for nor intentionally used.

Electronics modular assembly

Previously, electronics have been firmly attached to the robots body (Backman, J. et al. and Kemppainen, T. et al.). The problem with this structure is that robots body needs to be ready before electronics can be made. That is against of the idea of modular structure.

In Turtle Beetle, all the main electronic components have been attached to a piece of plexiglass. This way it was possible to build up a module for electronics and it was done simultaneous with building up the robot. The ready module can be attached to the robot very fast.

There was also second reason to build a module for electronics. With the module it was possible to start to use the electrical system although the mechanical structure was under construction. This was very important for testing electronics assembly, testing the protocols and software.

4. Algorithms and methods

The robot's main algorithms and methods comprise of sensing where the robot is (row detection), moving the robot (row navigation, end of rows turning) and detecting the weeds (machine vision).

One or more methods for each purpose were developed. All of these methods are described in more details in the following Chapters 0 -0. The main idea was to at first develop simple and easy to use methods. Later, more sophisticated methods were developed, but the both input and output interfaces of the algorithms had to remain the same, in order to provide easy changing from one method to the other. By doing this way, it was easier to find out what the method should do. It was learning by doing. We could also be sure that in every case we have at least one method that will do its work.

Row Detection

A few different methods for detecting the position of the robot were developed. All the methods use the same interface.

Inputs to the methods are measurements from the ultrasonic and infrared sensors and the laser scanner.

Outputs are the angle between robot's moving direction and the real direction of a maize row (in degrees, Figure 46), deviation from the center line of the row (in meters, Figure 47), probability that the robot is in the row and quality of row detection.

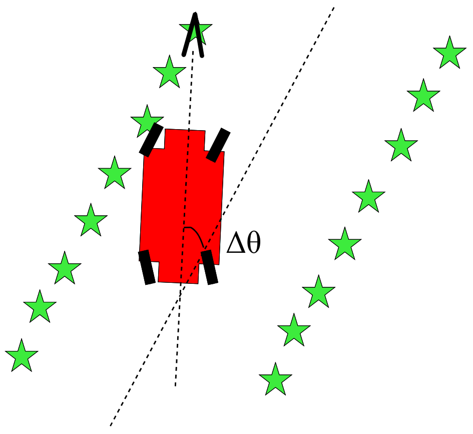


Figure 46: Angle error

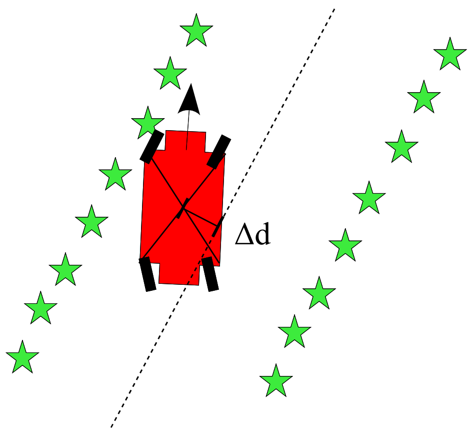


Figure 47: Distance from the center line

Detection using only ultrasonic sensors

The simplest method uses data only from the ultrasonic sensors. If some (or all) of the sensors give invalid measurement, the data will be rejected. The smallest and biggest acceptable measurements can be set simply by changing the value of the parameter.

Rejected measurement will be replaced by the data that is calculated from the last valid measurement of the current sensor and measurement of other sensor as an average. The first option is to use measurement from the same end but different side. The second option

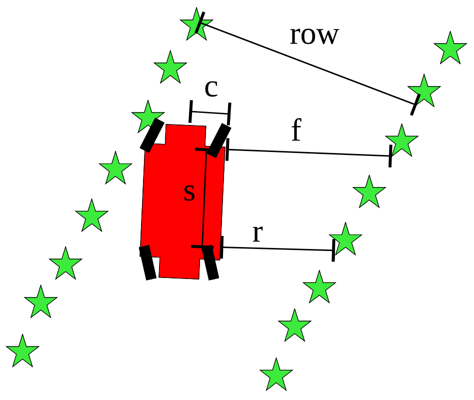


Figure 48: Calculating deviations

is the measurement from the other end and the same side.

Angle error can be calculated by using simple geometry, presented in Figure 48. Symbols *f* and *r* represent values that are obtained by using method described above.

$$\Delta\theta = \arctan\frac{f-r}{s} \tag{1}$$

Because sensors are symmetrically around the center point, deviation from center line can be calculated by using average of front and rear sensor readings

$$\Delta d = \frac{f+r}{2} + c * \cos(\Delta\theta) - \frac{row}{2} \tag{2}$$

On field tests it was noticed that this method works quite well. The steering angle tends to oscillate a bit.

Moving average, ultrasonic and infrared sensors

This method calculates the position of the robot first by using data from ultrasonic sensors and then from the infrared sensors. The calculations are made by using the same function for both sensor types. This function uses a few last valid measurements and calculates the (moving) average. If a measurement is not valid, then algorithm uses previous measurements but gives them less weight.

From the filtered values, the angle and distance deviations are obtained by using the same equations (1 and 2) than in the previous method.

Final values for deviations are obtained by calculating weighted average from the deviations of both ultrasonic and infrared sensors.

On field tests it was noticed that this method works quite well. The overall result is quite similar to the previous method.

Using the laser scanner and pre-defined areas

Data from ultrasonic and infrared sensors is used similarly as in the previous method. For laser scanner data, four areas ("boxes") around the robot have been defined. The average is calculated from the position of plants that fit inside the box (Figure 49). This way the method will get four points from the laser data. Each calculated point will get weight-value according to the number of plants found inside the box.

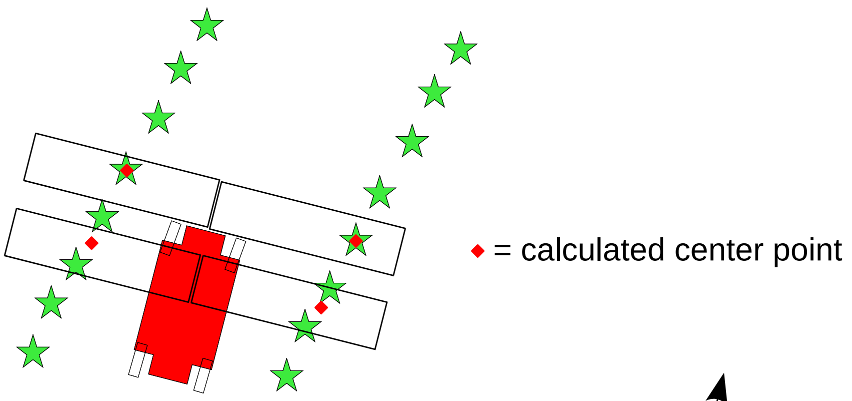
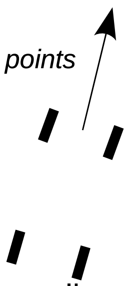


Figure 49: Pre-defined boxes and calculated center points



The next step is that measurements are shifted next to center line and then front end (and rear end) values are combined together using weighted average (Figure 50).

$$point = \frac{l * weightL + r * weightR}{weightL + weightR} \tag{3}$$

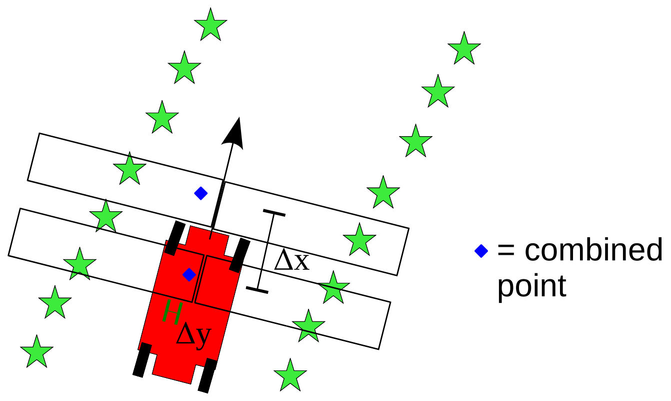


Figure 50: Combined points

Deviations can be calculated quite similar to previous methods.
Angle error

$$\Delta\theta = \arctan\frac{\Delta y}{\Delta x} \qquad (4)$$

where

Δx is distance between points in x direction and

Δy is distance between points in y direction

Deviation from center line

$$\Delta d = y_1 + \frac{\Delta y}{\Delta x} * x_1 \qquad (5)$$

where

x_1 is the x coordinate of front point and

y_1 is the y coordinate of front point

Laser for finding maize row

This method is experimental and it hasn't been completed. The idea of this method is

- if the measurement given by laser scanner is larger than threshold value, there is a gap
- laser scanner finds small gaps from both side of robot and the assumption is that there is equal amount of gaps on both sides
- the biggest "gap" is the row itself

Now it is easy to select the measurement that represents the center point of a row because it is the middlemost value. The angle of middle-most value is used as angle deviation.

Distance deviation is obtained by using ultrasonic and infrared sensor and methods described in Chapter 0.

Recursive least squares

The recursive least squares method uses all the data from the laser scanner (if the measurements aren't taken too far from the robot). This method fits the line to data by using recursive least squares algorithm. It is based on a function described in reference (Hyötyniemi H.). The current version is in Appendix 1.

We assume that distance between maize rows is constant that is why all the plants can be shifted to a single "mathematical" row by using modulus. It is possible to use this method only if the angle between the robot and rows is small. To be able to use this method in the general case, the data from the laser is rotated in small steps and each time least-squares is calculated. Finally, the rotation which gives the best correlation is selected. Simulated example figures are shown in

Appendix 2.

Summing up angles given by least-squares and rotation angle, the angle deviation is obtained.

$$\Delta\theta = \arctan(k) + rotationAngle \tag{6}$$

where

k is the slope given by least squares method

rotationAngle is the rotation angle which gives the best correlation.

The distance deviation is obtained

$$\Delta d = \frac{b}{\cos(rotationAngle)} \tag{7}$$

where

b is the intercept given by least squares.

While testing this method on the on-board computer, we noticed that it tokes too much CPU time. That is why it was decided not to use it on the field test.

Row Navigation

A few methods were developed for navigation in row. All the methods use the same interface. Inputs to the methods are the output values of the row detection method (the angle between the moving direction and the real direction, deviation from the center line of the row, probability that the robot is in the row and quality of the data). Outputs are the controls for driving and steering motors.

Independent PID-controllers for the front and rear end of robot

This method calculates the distance between the corners of the robot's frame and the maize row. Basically, this is reverse operation to defining angle and distance deviations in Chapter 0 because optimal ultrasonic sensors would give desired values. Now by using deviations, calculated distances are "filtered".

Left-side distance is given as a set-point to PID-controller and right-side as measurement because by doing this it does not need to use zero as a set-point. Front and rear axle steering are controlled independently.

On field tests it was noticed that this method works well.

Advanced PID-controllers for angle error and position deviation

In this method PID-controllers correct the angle error and the deviation in robot's position. At first it is calculated how far the center point of the front end (and rear end) is from the center line of the row.

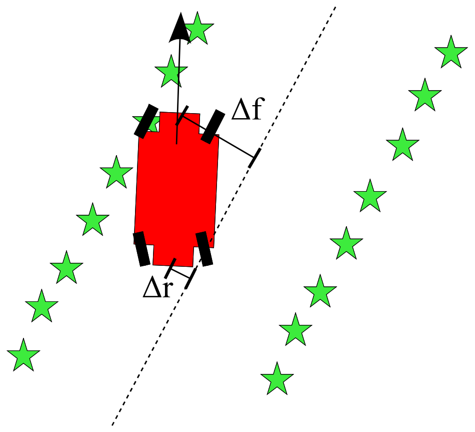


Figure 51: Front and rear end deviations

One PID-controller tries to correct the deviation of the front end by steering front wheels and the other controller the deviation of the rear end by steering rear wheels. The angle error is corrected by subtracting it from controllers' outputs. The result is that in situation presented in Figure 51 angle error makes front wheels turn to the right even more and rear wheels turning less to the right.

On field tests it was noticed that this method works quite well but the result was not as good as independent PID-controllers for the front and rear end of the robot.

Drive to target point

In this method a target point for the robot is calculated. Only front wheels are steering. This method is an experimental method and was not thoroughly tested. Method seems to work in some conditions, but probably a more sophisticated control would be better. Adding for example I and D terms with some limitations could be the right solution.

The idea is that there are two parts in the turning angle of front wheels. The first part is the angular deviation of the robot $\Delta\theta$. The second part is calculated from the distance deviation of the front end Δf and from one tunable parameter dX (Figure 52)

$$\Phi = -\Delta\theta - \arctan\frac{\Delta f}{dX} \qquad (8)$$

where

Φ is the turning angle of front wheels,
$\Delta\theta$ is the angular deviation of the robot,
Δf is the distance deviation of the front end and
dX is the tunable parameter.

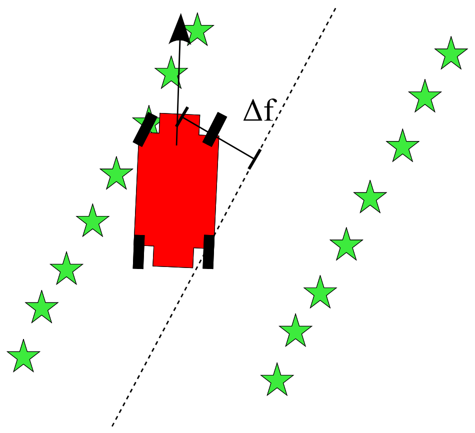


Figure 52: Driving when only front wheels are steering

Row End Detection

The main sensor for detecting the end of the row is a laser scanner. If the laser scanner does not find enough plants (more than 50) near the robot (closer than 0.8m), it is assumed that the row has ended (Figure 53). Back-up system uses ultrasonic sensors. If none of the sensors detect plants, inside a certain range (from 0.01m to 0.35m), during a certain time (nine program cycles), it is also assumed that the row has ended.

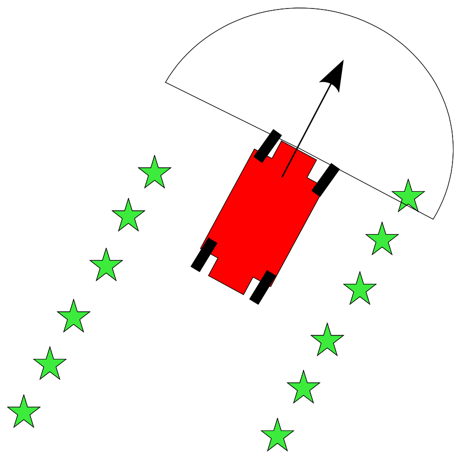


Figure 53: Row end detection by using laser scanner

On field tests it was noticed that this method works very well and normally ultrasonic sensors are not needed.

Turning Methods

A turning method is activated after the row end has been detected. All the turning logic methods have been implemented by using Simulink Stateflow tool.

Simple turning

The simple turning method uses only angle data from electronic compass and distance data from axle modules. At first robot makes a 90-degree turning to the desired direction. After the turn, the robot drives backwards or forwards depending on the position of the next row. The next step is another 90-degree angle turn. Finally, the robot drives forward until it detects the row. The Stateflow model is shown in Appendix 3.

On field tests it was noticed that this method works quite well. However, it is important to calibrate compass and turning parameters on site.

Wrong-way turning

Wrong-way turning method is very similar to the simple turning. The only difference is that instead of a 90 degree angle turn at start-up, a 270-degree angle turn is done. Because of that it seems that the robot starts to turn to the wrong way.

The idea for this method is that it makes possible to count all the rows while driving in the headland. Counting the rows is important to make sure that we turn back to right row. If rows are not counted, turning is made solely based on odometry.

This method has been tested only in the simulator. According to simulation it needs too much space at the end of the row so it was decided not to use it.

Advanced turning

The basis of advanced turning method is similar to the simple turning method. The advanced feature is that after the first 90 degree turn this method uses data from the laser scanner to keep the distance to the row endings at desired value. It is based on counting the number of plants inside the certain area that is shown in Figure 54.

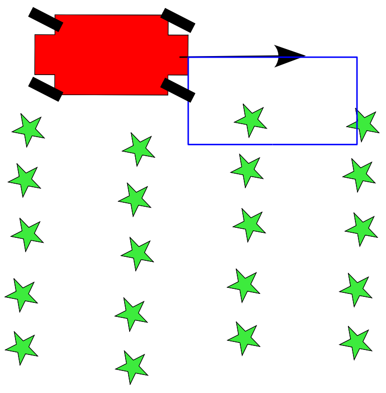


Figure 54: Keeping constant distance to row ends

Two different methods for actual steering were developed.

The first method is very simple.

- if there are not enough plants inside the box, turn both wheels to the maximum angle towards the plants
- if there is a correct amount of plants inside the box, keep wheels straight
- if there are too many plants inside the box, turn both wheels to the maximum angle away from the plants

Second method is more sophisticated and it uses P-controller to steer wheels. There is only one common controller for both front and rear wheels.

$$\Phi = K_p * err \tag{9}$$

where

Φ is the turning angle of wheels,

K_p is gain and

err is the error between set-point and the number of plants inside the box.

On field tests it was noticed that both methods work quite well. Because there were only a few plants inside a box, even a small variation in the number of plants will cause quite a big change in the steering angle while using P-controller. That is way both methods work quite similarly.

Weed Detection

The web camera is directed at a right angle to the ground so there is no need to geometrically correct the image. After the image acquisition, the luminance or color plane of the RGB image is extracted to get a gray level image. Because the objective is to look for bright white objects on a bright green background, the best option seems to be either red, blue or luminance plane extraction depending on the overall circumstances. The image look-up table needs to be reversed so that bright objects become dark, and the image data is then manually thresholded to a binary image. The threshold value is chosen close to zero. (Figure 55)

The result from the previous stage is a binary image with blobs of varying sizes and forms. To get the daisies, which are nearly uniform in size and shape being small circles, the blobs that are either too big or small are needed to be filtered out, and then from the re-

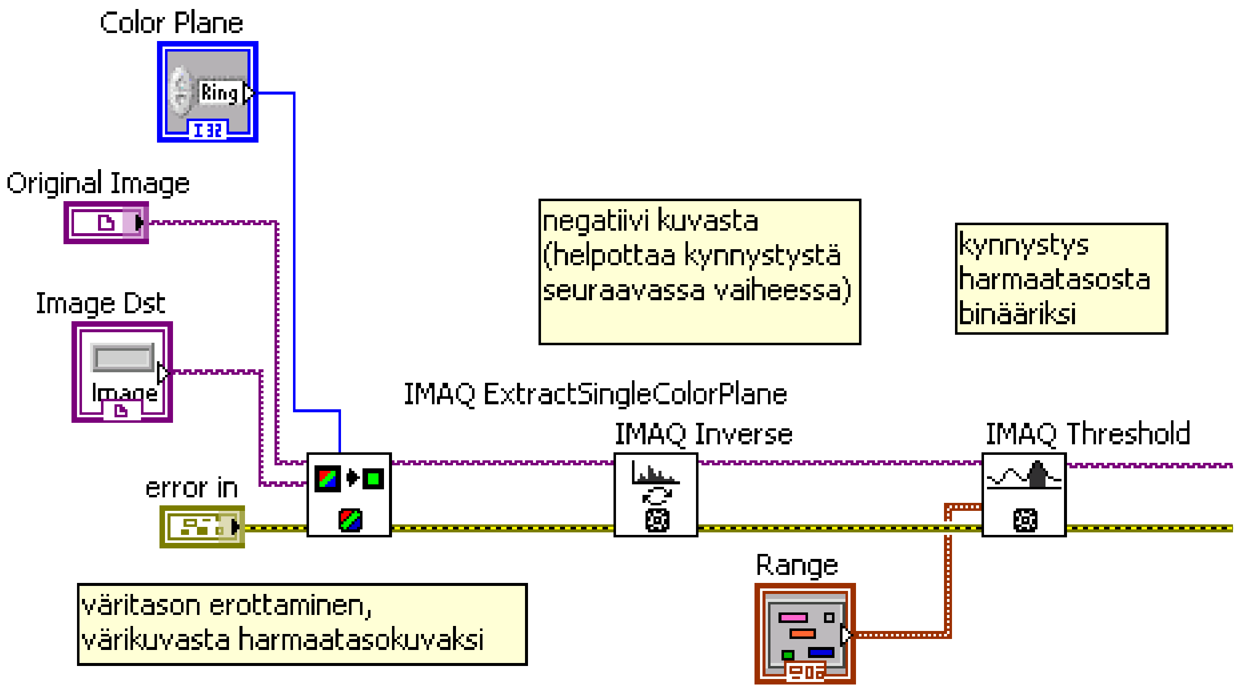

Color Plane
Ring
I32
Original Image
Image Dst
Image
error in
negatiivi kuvasta (helpottaa kynnystystä seuraavassa vaiheessa)
kynnystys harmaatasosta binääriksi
IMAQ ExtractSingleColorPlane
IMAQ Inverse
IMAQ Threshold
Range
väritason erottaminen, värikuvasta harmaatasokuvaksi

Figure 55: Pattern recognition for weed detection, first half. Color plane extraction, look-up table reverse and thresholding

maining ones look for circular blobs. (Figure 56)

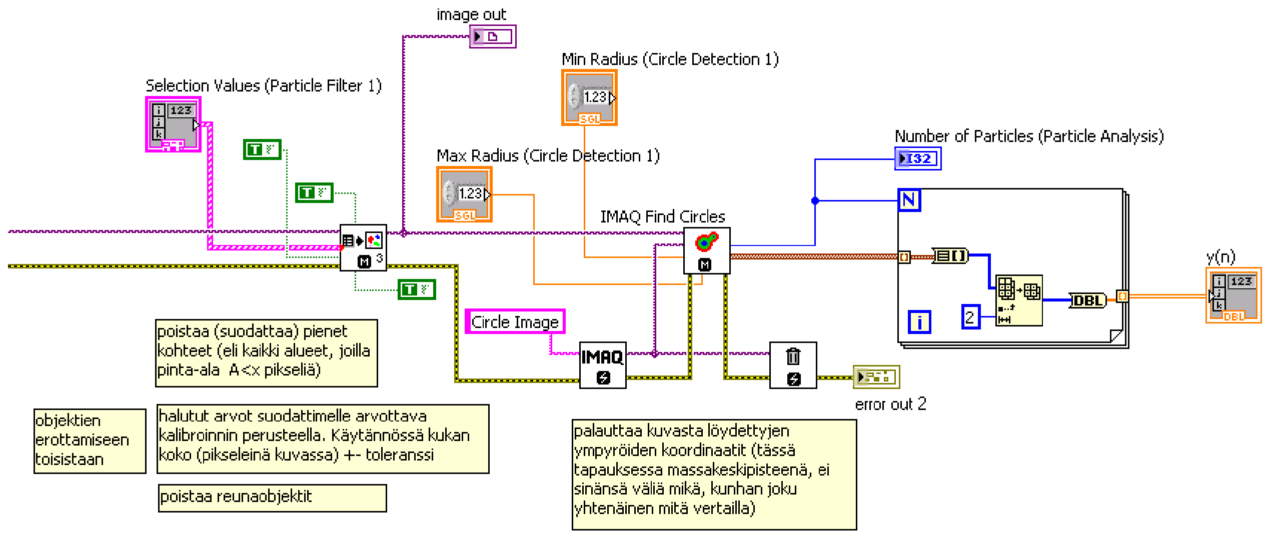


image out
Min Radius (Circle Detection 1)
Selection Values (Particle Filter 1)
Max Radius (Circle Detection 1)
Number of Particles (Particle Analysis)
IMAQ Find Circles
y(n)
Circle Image
IMAQ
error out 2
poistaa (suodattaa) pienet kohteet (eli kaikki alueet, joilla pinta-ala A<x pikseliä)
objektien erottamiseen toisistaan
halutut arvot suodattimelle arvottava kalibroinnin perusteella. Käytännössä kukan koko (pikseleinä kuvassa) +- toleranssi
poistaa reunaobjektit
palauttaa kuvasta löydettyjen ympyröiden koordinaatit (tässä tapauksessa massakeskipisteenä, ei sinänsä väliä mikä, kunhan joku yhtenäinen mitä vertailla)

Figure 56: Pattern recognition for weed detection, second half. Filtering small, big and border objects, detecting circles of certain radius and returning their mass centre points

The coordinates of the found blobs are sent to a tracker which keeps track of the blobs and returns data that can be used for the weed handling device. The tracking algorithm is very simple, just looking for blobs on the right or left side of the image and checking that the coordinates of the objects in the frame have changed (Figure 57).

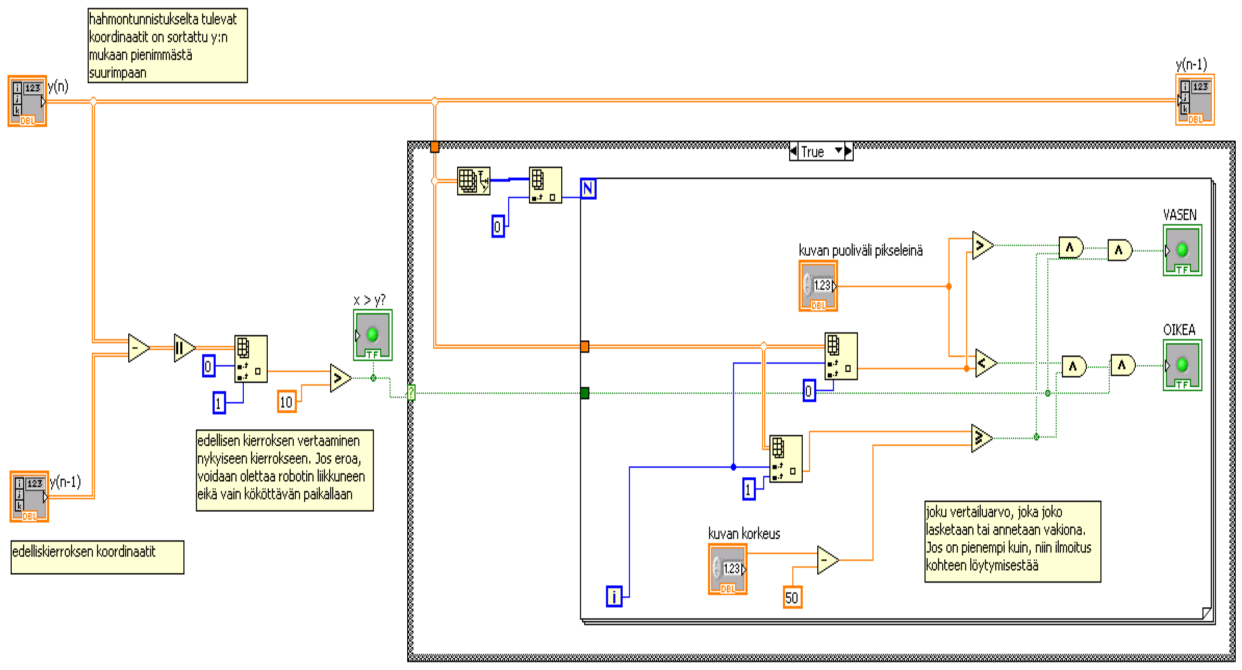

hahmontunnistukselta tulevat koordinaatit on sortattu y:n mukaan pienimmästä suurimpaan
y(n)
y(n-1)
True
VASEN
kuvan puoliväli pikseleinä
OIKEA
x > y?
edellisen kierroksen vertaaminen nykyiseen kierrokseen. Jos eroa, voidaan olettaa robotin liikkuneen eikä vain kököttävän paikallaan
y(n-1)
edelliskierroksen koordinaatit
kuvan korkeus
joku vertailuarvo, joka joko lasketaan tai annetaan vakiona. Jos on pienempi kuin, niin ilmoitus kohteen löytymisestää

Figure 57: Tracking of the objects (weeds)

5. Programming techniques and communications

Several programming languages and development environments were used. Graphical programming in the form of Simulink Stateflow charts and LabVIEW was used. Bulks of code were generated automatically, but a lot of coding had to be done by hand as well. Program structure is described in Figure 58.

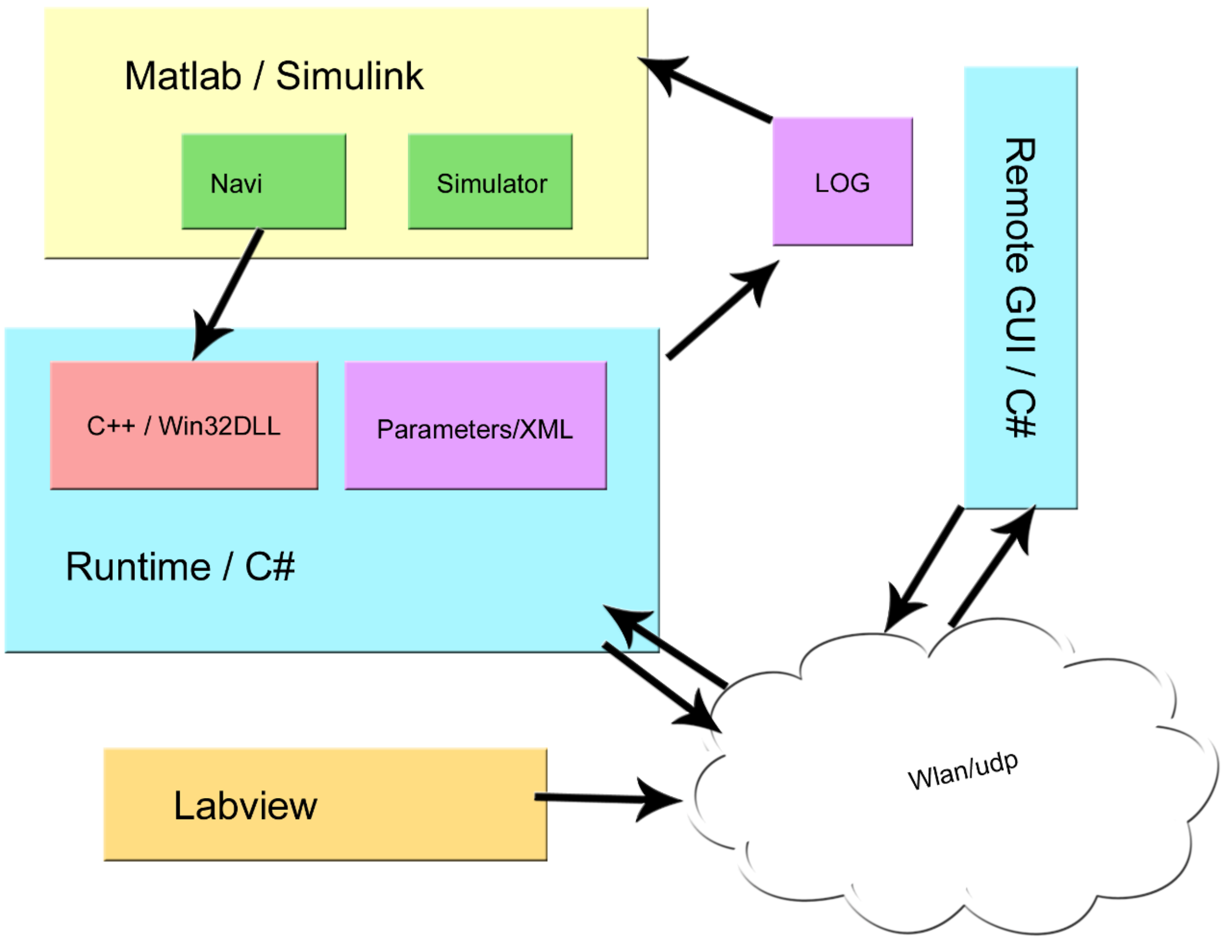


Figure 58: The program structure

Matlab & Simulink

The “brains” of the robot has been made using Simulink and also Matlab has been used for some functionality. The way to use Matlab and Simulink files in the robot is to generate them to C++ files. C++ files contain functions for running the program loop and definitions for inputs and outputs. Matlab version R2009b was used.

Matlab and Simulink have been also used for testing and simulation. The more about it is written in Chapter 0.

The Simulink model of the robots main program has been divided into functional groups, like row detection and row navigation. To make the model modular, Simulink libraries have been used for these functional groups. If someone makes a change to the library, the change will be updated to all models that use the same library. Of course the interface must be determined at first and it should not be changed. After that it is possible to develop different parts of the program at the same time by different people.

Libraries might contain other libraries or to be more precise, links to other libraries. We have used this possibility and whole navigation has been packed in one library block. The navigation and simulator blocks are shown in Figure 62.

Another important technique used is Simulink buses. Buses are basically signals that contain many signals. By using buses, it is possible to make interfaces to look simpler. Buses are also a good thing when compiling the model to C++. One bus is converted into one struct in C++ file, and all the signals in the bus are converted to fields into the struct.

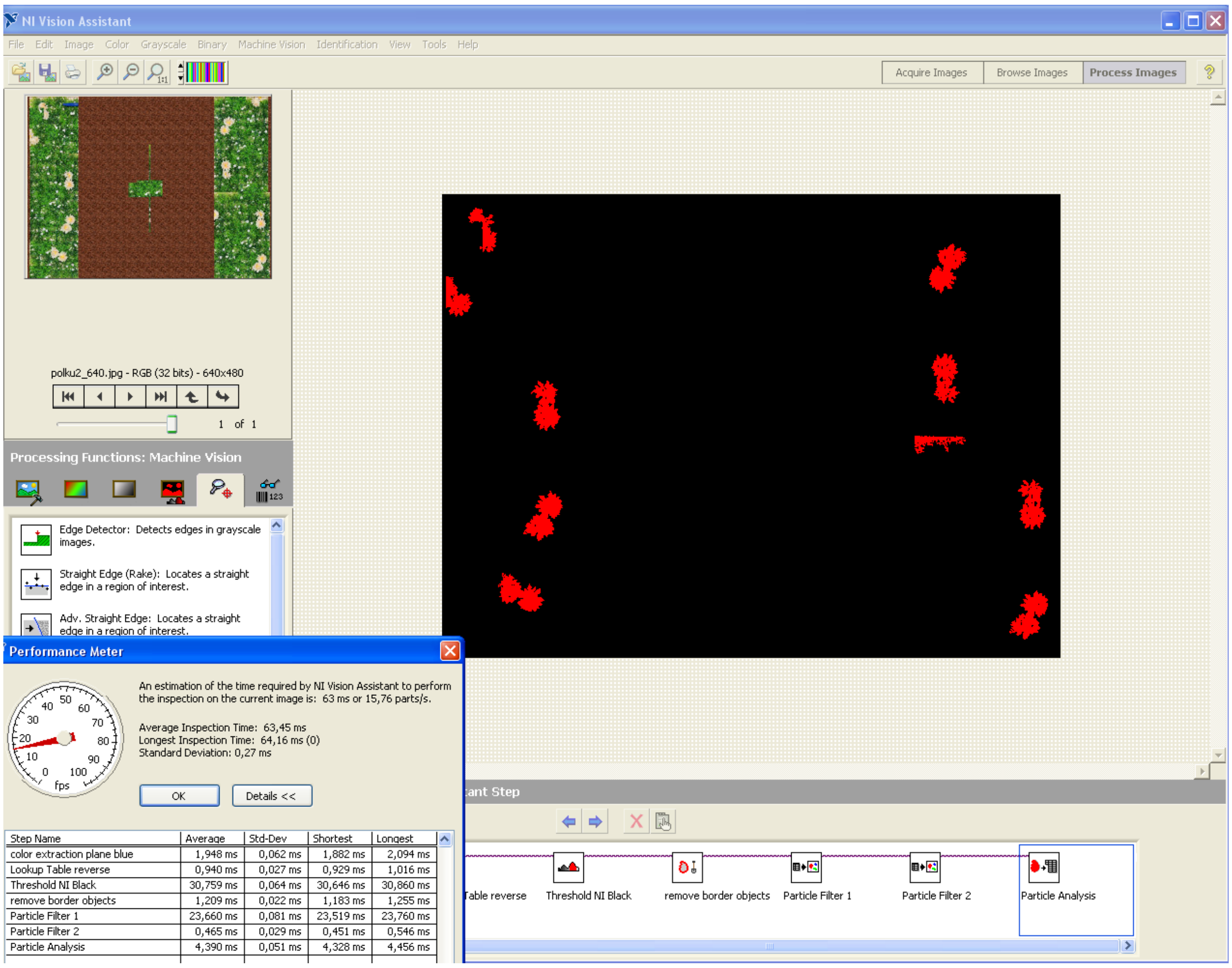
Most of the used blocks are standard Simulink blocks, but also Embedded MATLAB Function blocks have been used. Some of them contain embedded code and some functions that have been made using Matlab m-files. It is not possible to use all Matlab functions because only part of them can be generated into C++.

The turning methods and some other features of the robot have been made by using Simulink Stateflow, which is a tool for drawing illustrative state machines. Stateflow is a good tool for describing a sequential action. An example about Stateflow is shown in Appendix 3.

LabVIEW and NI Vision

LabVIEW is a graphical programming language designed for measuring and automation applications. NI Vision is a platform of both software and hardware designed for machine vision applications. (National Instruments) Both of them were used to build the weed detection application for the field robot.

There are at least three approaches to developing machine vision applications with LabVIEW and NI Vision. The first is the basic approach: wire LabVIEW elements together and use the Vision library. With this approach, one has to know what he/she is doing. The second method is to use the Vision Builder application to build more complicated applications. For this project, the resulting LabVIEW code was far too complicated. The third method is to use Vision Assistant, a separate program which comes with the Vision package. For a novice developer it is probably the easiest approach to find most suitable algorithms and it was used for this project as well (Figure 55 and Figure 59).


NI Vision Assistant
File Edit Image Color Grayscale Binary Machine Vision Identification View Tools Help
Acquire Images
Browse Images
Process Images
polku2_640.jpg - RGB (32 bits) - 640x480
1 of 1
Processing Functions: Machine Vision
Edge Detector: Detects edges in grayscale images.
Straight Edge (Rake): Locates a straight edge in a region of interest.
Adv. Straight Edge: Locates a straight edge in a region of interest.
Performance Meter
fps
OK
Details <<
ant Step
Table reverse
Threshold NI Black
remove border objects
Particle Filter 1
Particle Filter 2
Particle Analysis

An estimation of the time required by NI Vision Assistant to perform the inspection on the current image is: 63 ms or 15,76 parts/s.

Average Inspection Time: 63,45 ms
Longest Inspection Time: 64,16 ms (0)
Standard Deviation: 0,27 ms

Step Name	Average	Std-Dev	Shortest	Longest
color extraction plane blue	1,948 ms	0,062 ms	1,882 ms	2,094 ms
Lookup Table reverse	0,940 ms	0,027 ms	0,929 ms	1,016 ms
Threshold NI Black	30,759 ms	0,064 ms	30,646 ms	30,860 ms
remove border objects	1,209 ms	0,022 ms	1,183 ms	1,255 ms
Particle Filter 1	23,660 ms	0,081 ms	23,519 ms	23,760 ms
Particle Filter 2	0,465 ms	0,029 ms	0,451 ms	0,546 ms
Particle Analysis	4,390 ms	0,051 ms	4,328 ms	4,456 ms

Figure 59: NI Vision Assistant with performance meter

Algorithms (scripts) were easy to develop and test for performance with the performance meter with Vision Assistant. Final script was easy to make into a LabVIEW virtual instrument ready to be used. The main program is a cut and paste of LabVIEW / NI Vision examples (Figure 60 and Figure 61).

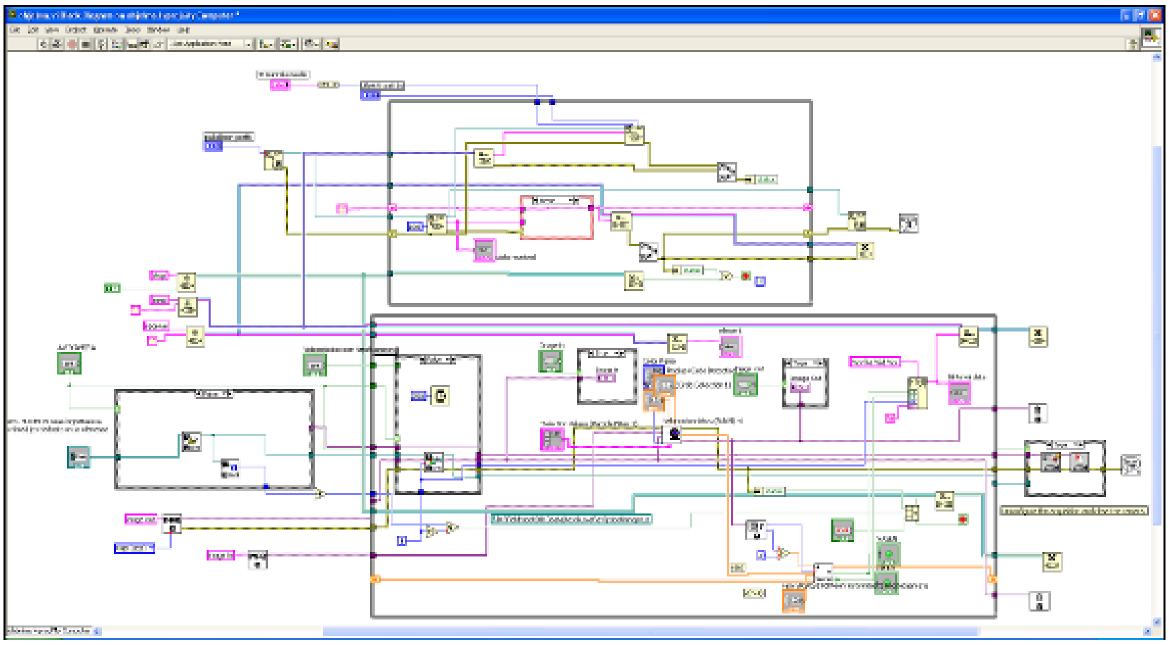
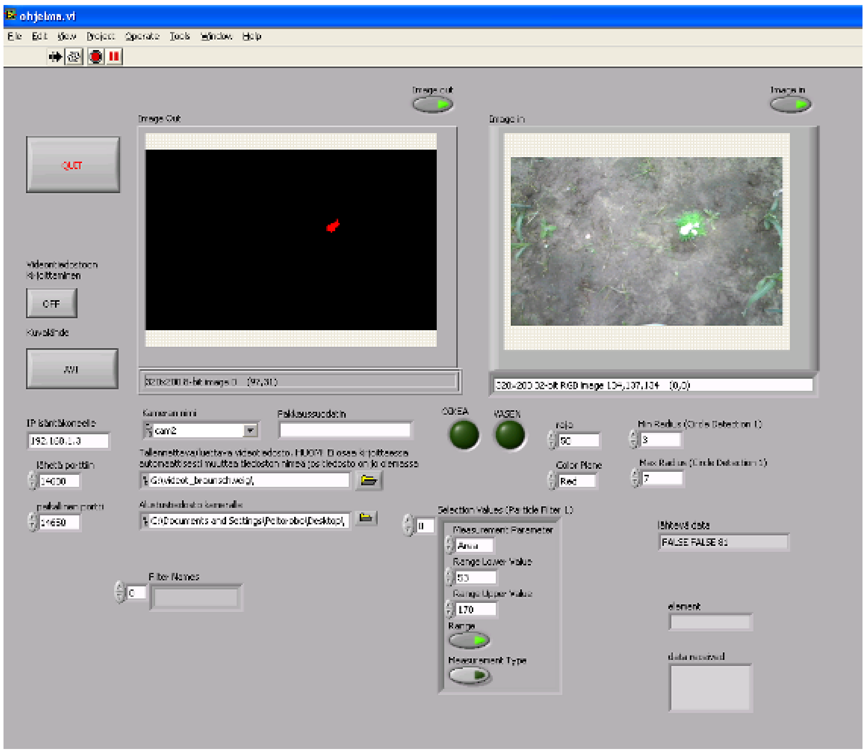


Figure 60: Example of program "code"



Videontiedoston kirjoittaminen: OFF
Kuvalähde: AVI
320x200 8-bit image 0 (97,31)
320x200 32-bit RGB image 104,107,104 (0,0)
IP lähtökoneelle: 192.168.1.3
lähetä porttiin: 14000
paikallinen portti: 14650
Kameran nimi: cam2
Pakkausuodatin:
Tallennettava/luettava videotiedosto. HUOM! Ei osaa kirjoittaessa automaattisesti muuttaa tiedoston nimeä jos tiedosto on jo olemassa
G:\videot_braunschweig\
Alustustiedosto kameralle
C:\Documents and Settings\Peltorobot\Desktop\
Filter Names: 0
raja: 50
Color Plane: Red
Min Radius (Circle Detection 1): 3
Max Radius (Circle Detection 1): 7
Selection Values (Particle Filter 1): 0
Measurement Parameter: Area
Range Lower Value: 50
Range Upper Value: 170
lähtevä data: FALSE FALSE 81
element
data received

Figure 61: User interface

Visual Studio

Visual Studio 2005 and C# was used to build the main program, the remote user interface running on another laptop and communication with microcontrollers and remote user interface.

The main loop is based on Simulink RTW generated C++ code. The problem between different programming languages was solved by using P/Invoke. Basically it means that C++ code is generated as Windows DLL library. Functions in the library can be defined to C# code and called then as normal C# functions.

The idea behind this program structure is that when Simulink code is generated, it works directly without changes in the main code.

Telecommunications

Communication between the laser scanner, the main program, machine vision and the remote user interface is based on regular WLAN and LAN technologies. The robot contains a wireless router which is used as a platform for a standard IP network.

Communication between the laser and the main program is done with TCP/IP, as this is the way SICK scanner provides. All the other communication routes use simple UDP. Basically, every machine has their own predefined IP address and every communications message has its own port for UDP. When a program receives a message, the type of the message is identified by the port used.

Messages are basic C# structs which are serialized with Compact Formatter Plus before sending. Compact Formatter is used for changing structs to bit streams for UDP connection.

Communication with microcontrollers uses basic serial port. For this purpose a special “field robot protocol” was defined so that the developers of the C# code and C code have a common specification.

Parameters

We divided parameters into four different categories. Categories are explained in Table 1.

Table 1 - Parameter categories with their explains.

The prefix of category	Meaning
p0_	Parameter is constant and its value can be set only in Matlab e.g. The wheel base of the robot
p1_	Parameter might need some online tuning. It is possible to tune it manually in XML-file e.g. Warning limit for battery voltage
p2_	Parameter needs tuning. It is possible to tune it in the remote user interface. These parameters are common for every task in the competition e.g. Compass bias
p3_	Parameter needs tuning. It is possible to tune it in the remote user interface. These parameters are task specific e.g. Driving speed

Parameters in categories p1-p3 can be tuned and these parameters are stored in one common XML-file. This XML-file is stored in the robots onboard computer (eBox). Parameter values can be loaded from this file to the remote user interface whenever they are needed.

Parameter Slots

One nice feature used were parameter slots. In the program code one parameter “set” was written as a struct. In the eBox there was a list of these structs and structs were named as “Task 1”, “Task 2” et cetera. The main idea was that there was own slot for each task and all the parameters could be saved before hands. In the competition all you have to do is select the correct parameter slot instead of changing many parameters at once. This list was very easy to serialize to XML as described in the previous section.
Parameter slots proved to be successful in the competition and the system worked very well. However in the testing phase different parameter slots caused also some annoyance.

6. Testing

Our testing consisted of three parts: a simulator running in Matlab Simulink, artificial test field and of course individual testing for each part.

Individual/unit testing

Basically all the parts made had to be tested before use. Our testing method wasn't very precise and planned but before assembling bigger systems we tested individual subsystems. Axle modules for example, were tested thoroughly before they were attached to body.

The same methods were used with software. After making a sub routine, it is very important to test it because finding problems in bigger systems are somewhere between magic and wizardry.

Simulator

For testing our methods without the actual robot, a simulator library block was built in Matlab Simulink. Basically, it consists of basic kinetics which calculates robot's new position in our imaginary maize field and blocks which generate measurements for navigation algorithms used.

The simulator block is connected to the navigation block. This same block is used in the robot for its navigation. Of course it has been compiled to C++ code before using it in the robot. This structure is shown in Figure 62.

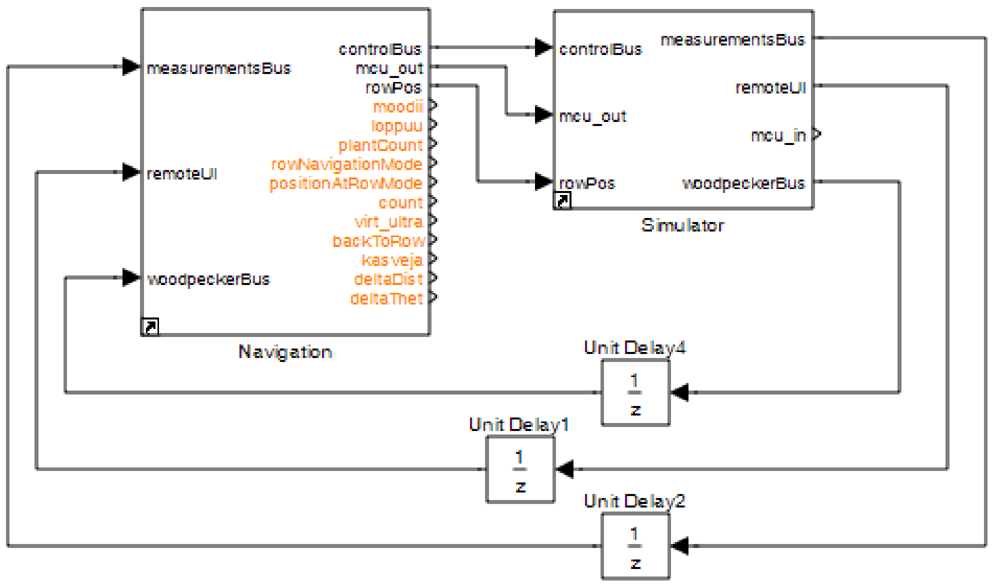


Figure 62: Connecting navigation library to simulator

At the beginning of simulation a field is created or loaded from a file. This field is plotted using basic Matlab *plot* command. After that robot is added to the same figure by using *fill* command. During the simulation Simulink model of the simulator (Figure 64) calculates a new position of the robot by using a kinematics model. The robot is moved to its new position by using figure handles and *set* command. It is computationally more efficient to “move” the robot than plot it again every time. The plotted user interface is shown in Figure 63.

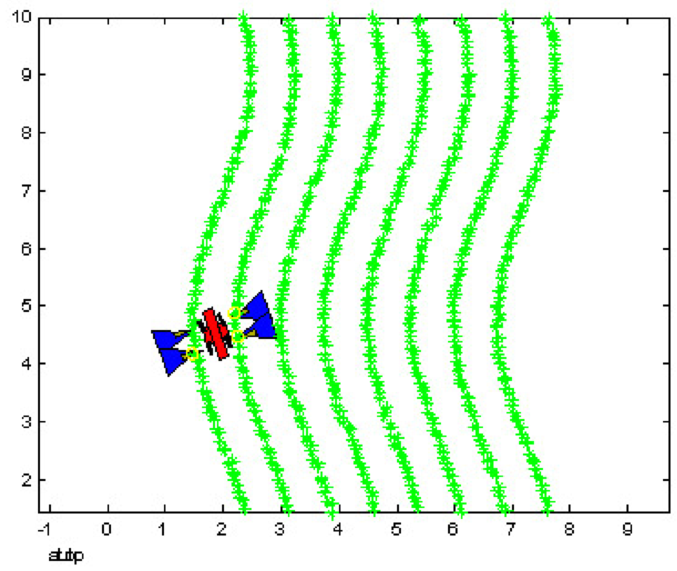


Figure 63: Simulator user interface

The simulator uses *find* and *inpolygon* functions to determine whether robot “sees” plants or not. The simulator block is not compiled to C++ so it is possible to use the wider range of Matlab functions than in the navigation block.

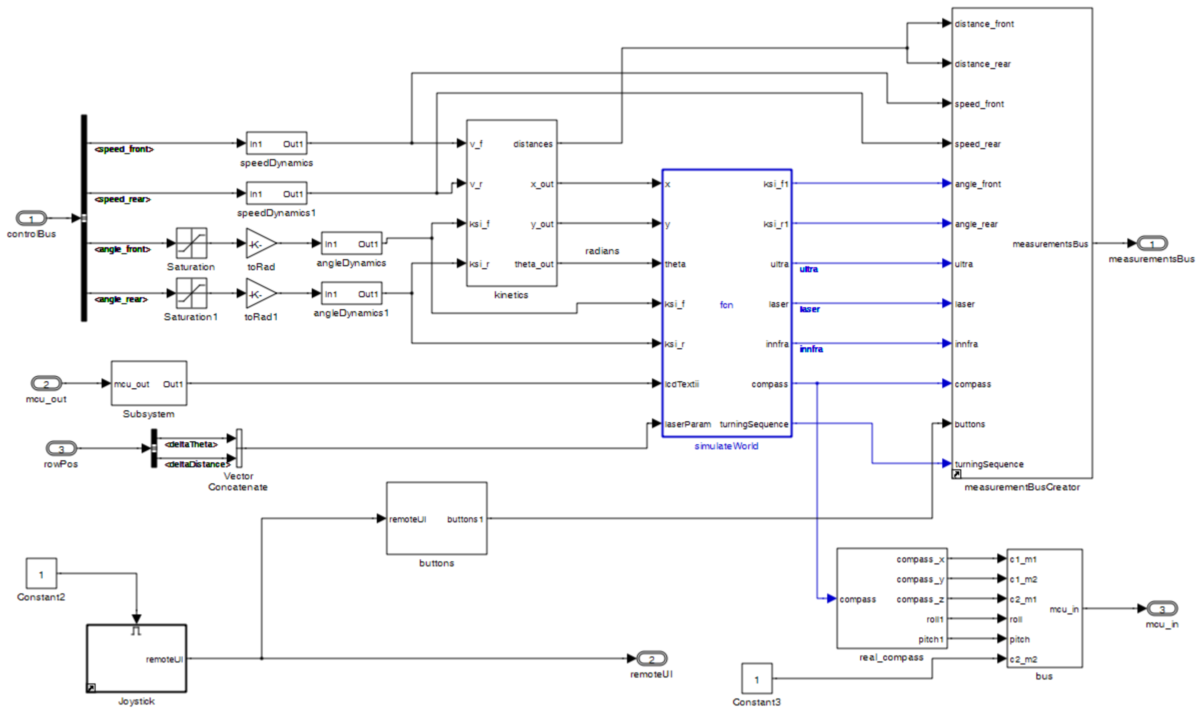


Figure 64: Simulink model of simulator block

Test Field

When the robot was constructed, we set up an artificial maize field at the back field of our university buildings (Figure 65). For the test field, a plastic construction fence was used and erected it with metal sticks. Construction was a very easy task and our hypothesis was that holes in the fence would generate enough noise for our measurements.

However, our artificial field was not that optimal. The plastic fence had some very nasty properties. Firstly if the robot hit the fence, it would suck the robot into it. A real maize row would have bent a bit. This was a very nasty feature because our sticks were made of steel so there were good chances to hit some iron with our €1000 laser scanner or ultra-sonic sensors.

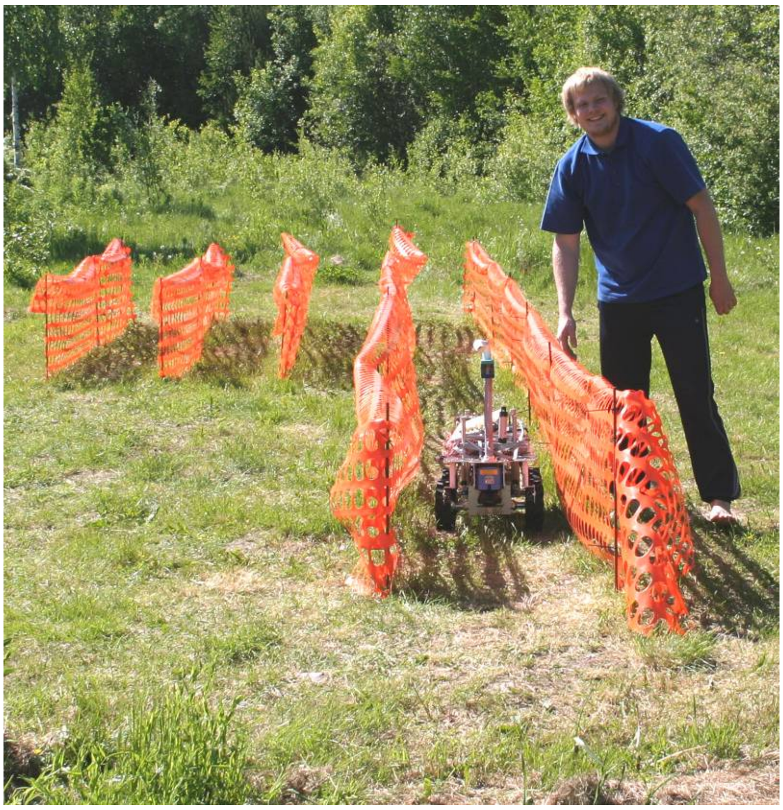


Figure 65: Test field in Finland

Second bad feature was vertical stripes and if the stripe was at the same level with sensors, there was not enough noisy signal.

Thirdly, the plastic fence easily makes waves horizontally and the amplitude of the waves can be 10 centimeters at the height of the robots sensors.

The biggest problem was the actual field in Germany (Figure 66). The spring was very cold and maizes were very tiny. In the test field, hardly any real maizes appeared and artificial sticks were put as a replacement. In the competition field, there were somewhat more real maizes but sticks were still used.

Anyway the tests were success in that sense that the robot performed pretty good when comparing with the other teams with less testing.

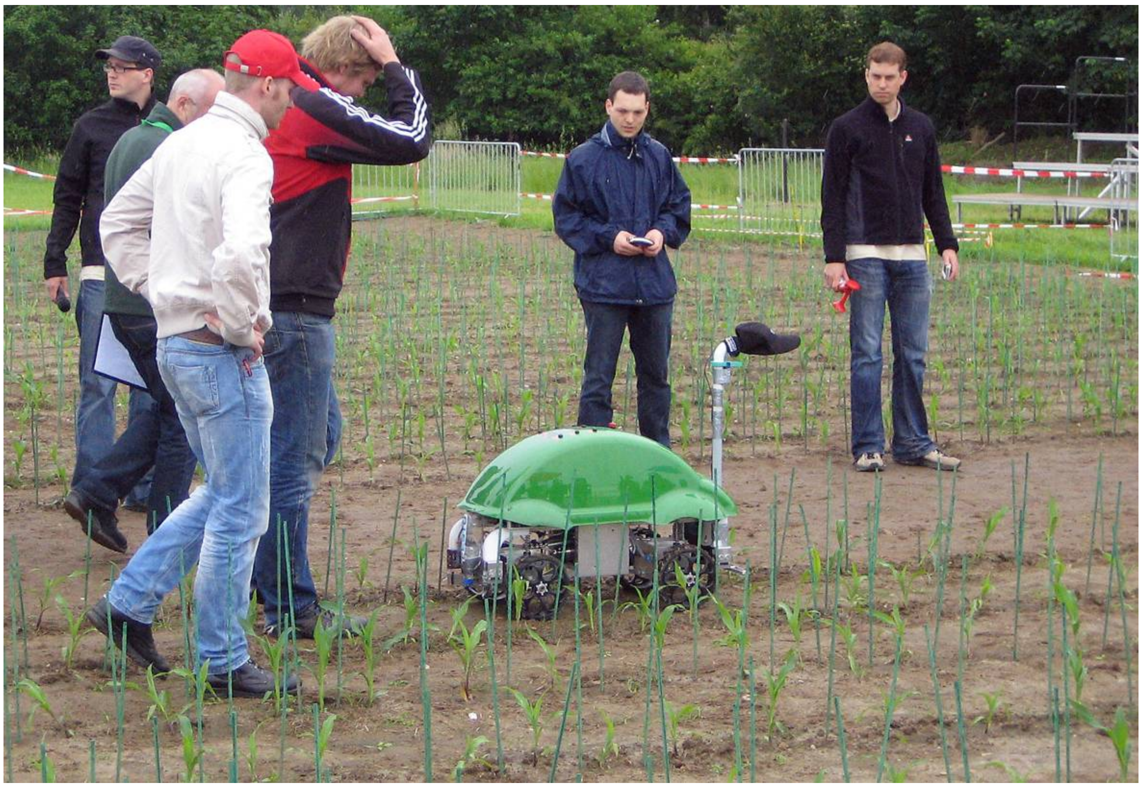


Figure 66: Competition field in Germany

7. Discussion

The robot participated in the Field Robot Event 2010 in June 2010 and was third in overall placing. The team had some major difficulties before the contest.

The schedule for building the robot stretched way over the set deadlines. A lot of work was done just weeks before the competition and even on the day before competition some development and debugging was done. It is difficult to say though if the robot had worked out better with more time or better time management.

When starting the project, it was difficult to tell what kind of things are needed and in which order they should be conducted. Many things were redone over and over again. Also some of the things were done to late because they were not recognized as 'bottlenecks'.

The robot’s structure is quite complicated. For learning purposes this is good, but for better performance in actual competition, simplified structure would probably perform better.

We also used pneumatics for active suspension and for agricultural machinery. Active suspension with used construction was relatively weak but the main concept was a success. Probably it could be possible to actively maintain robot straightness in slope fields with this kind of structure, but for the compensation of quick movements our system is too slow. With agricultural machinery pneumatics worked fine.

Machine vision was made as simple as possible. Luckily, the weather was not too sunny so there was no need to adjust the camera for bright light. Unfortunately, the camera was still too low so that the field of vision was not wide enough and many of the weeds were missed. Because of the lack of the robust tracking algorithm, many of the initially detected weeds were missed when signalling the robot for weed control.

Testing in Finland was quite successful but basically almost everything broke down in Germany before the contest. Our compressor for pneumatics blew up twice and the third compressor was factory-made broken. The fourth generation was a combination of broken compressors. The main difficulty was a mystical electrification problem which made robots LEDs tingle like a Christmas tree and buttons to be pressed randomly. After changing one of the controllers and many hours our team found that one wiring to compass was somewhat loose which caused problems with I2C-bus. Two hours before the competition our machine vision computer broke down. Also we had some broken down ultrasonic sensors and WLAN-difficulties when both computers used wireless.

8. Conclusions

Most of the concepts tried in this experimental project worked as expected. However with some extra tuning and testing the robot could perform better and more reliably in the field.

Probably concepts of this kind are the future of the farming in more advanced countries. There are still many issues to be solved before robot's can perform their tasks autonomously on the fields.

The used development tools for the main program, Simulink, code generation, and Visual Studio were easy to use after some serious magic tricks in the beginning.

It was relatively easy to make the logic by using Simulink because one only needs to understand the block diagrams. It is also relatively simple to make a simulator where the navigation part can be tested. Very useful features in Simulink are: Libraries, Buses and

Stateflow tool. Libraries make it possible to "write" modular and recoverable software. Busses are in the important role for making simple interfaces both in Simulink and in C#. Stateflow is very powerful tool for making software that runs in steps. Of course to be able to use these features in the robot, the code generation is needed. At the first time, some parameters needs to be set but after that the usage is very simple: just press one button.

Visual Studio and C# language provide a combination that is quite easy to use. Especially tools for creating graphical user interfaces are very easy to use. Nevertheless there are also some complicated "tricks" to do before it is possible to use external C++ files created by Simulink. User for example has to add some path variables, manually modify a few files and give some compiler parameters. All these are things that user just has to know. After setting these things, using C++ functions still needs p/invoke method. In fact p/invoke method is very easy to use, all it needs are some special keywords. It can be say that these development tools form a very powerful combination and these tools should be first choice for a project like this. The only thing that was a disappointment was Windows CE (and Compact Formatter Plus). CE operating system should use only if other systems like Windows XP Embedded cannot fulfil strict real-time requirements.

Weed detection was simple but effective enough. With the test runs, it worked fine but for some reason, probably due to a communication lag between machine vision and robot, it failed to perform in a desirable way in the competition.

Acknowledgments

We would like to thank all our sponsors that made possible for us to do the project, and make a trip to the competition to take third prize in the overall competition. The sponsors were Valtra, Koneviesti, Suomen Kulttuurirahasto, SICK, FESTO, Laserle, HP InfoTech, Mathwork, Toradex, VTI. We would also like to thank our hosting universities: Aalto University and University of Helsinki.

Special thanks to Pekka Kumpula for designing the cover.

References

Hyötyniemi H.: *Multivariate Regression - Techniques and Tools*. Lesson 4. Page 66. https://noppa.tkk.fi/noppa/kurssi/as-74.4191/materiaali/05._chapter_4.pdf

Backman, J., Hyyti, H., Kalmari, J., Kinnari, J., Hakala, A., Poutiainen, V., Tamminen, P., Väätäinen, H., Oksanen, T., Kostamo, J., Tiusanen, J.: 4M - Mean Maize Maze Machine. http://autsys.tkk.fi/en/FieldRobot2008.

Kemppainen, T., Koski, T., Hirvelä, J., Lillhannus, J., Turunen, T., Lehto, J., Koivisto, V., Niskanen, M., Oksanen, T., Kostamo, J., Tamminen, P. Robot Brothers Easy Wheels and ReD in Field Robot Event 2009. Proceedings of the 7th Field Robot Event 2009.

FireWire - Still the Performance King! http://www.cwol.com/firewire/firewire-vs-usb.htm - accessed June 30th 2010

Homepage of National Instruments. http://www.ni.com/ accessed June 30th 2010

Appendix 1

```
function [thetas, R2, y2] = rlsmodulus(Fii, Y, lambda, theta0, P0, p3_maizeRowSpacing)
%#eml

% system dimensions
[n,m] = size(Fii);

% initial values
P = P0;
thetas = theta0;
R_2 = 0;

y2 = zeros(n,1);
f2 = zeros(n,2);

Index = 1;
maizes = p3_maizeRowSpacing;

for k = 1:n
    f = (Fii(k,:));
    y = Y(k);

    dX = inf;
    dY = inf;

    % plants that are not close to robot
    if(f(1,1) < -dX || f(1,1) > dX || y(1,1) < -dY || y(1,1) > dY)
        y2(Index,1) = 0;
        f2(Index,:) = [0 0];
        Index = Index +1;
    else
        f2(Index,:) = f;

        % devider (scalar value)
        S = f * P * f' + lambda;
        W = P * f' / S;
        P = P - W * S * W';

        ys = (mod(y+maizes/2 , maizes) - maizes/2);

        %for R2
        y2(Index,1) = ys;

        %scalar
        modulus = ys - (f * thetas);

        % evaluated parameters
        thetas = thetas + W * modulus;

        Index = Index +1;
    end
end

SSe = (y2 - f2*thetas)' * (y2 - f2*thetas);  % eq 4.15
SSt = y2' * y2;         % eq 4.16

if(SSt ~= 0)
    R_2 = abs(1 - SSe / SSt);   % eq 4.14
else
    R_2 = 0;
end
```

```
% Check that 0<= R2 <= 1
if(isnan(R_2) || isinf(R_2) || R_2 > 1 || R_2 < 0)
    R_2 = 0;
end

R2 = R_2;
End
```

Appendix 2

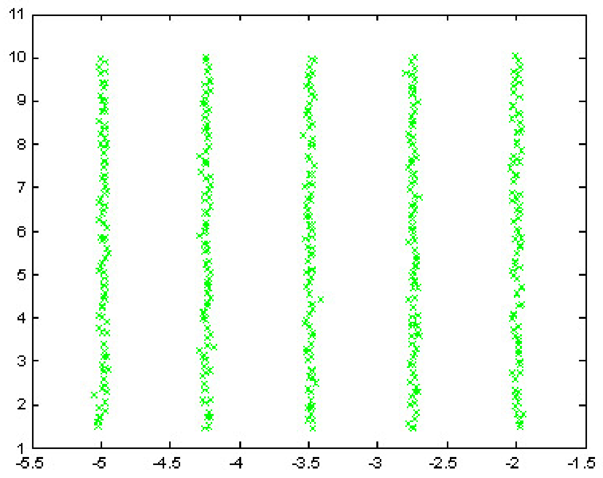


Figure 67: Simulated data from field

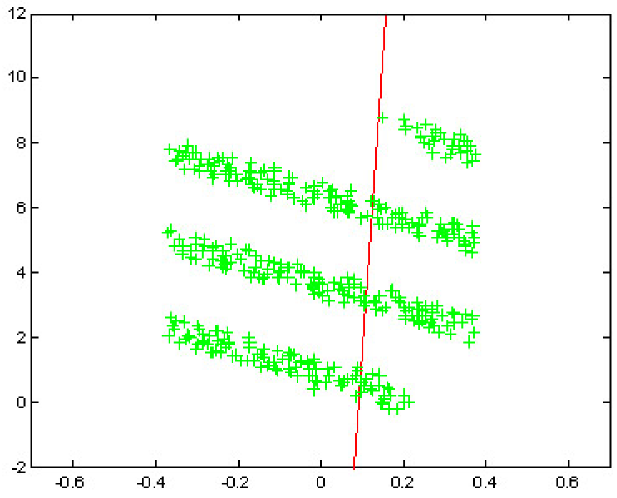


Figure 68: Angle between robot and rows is too big. The data folds and line fitting fails

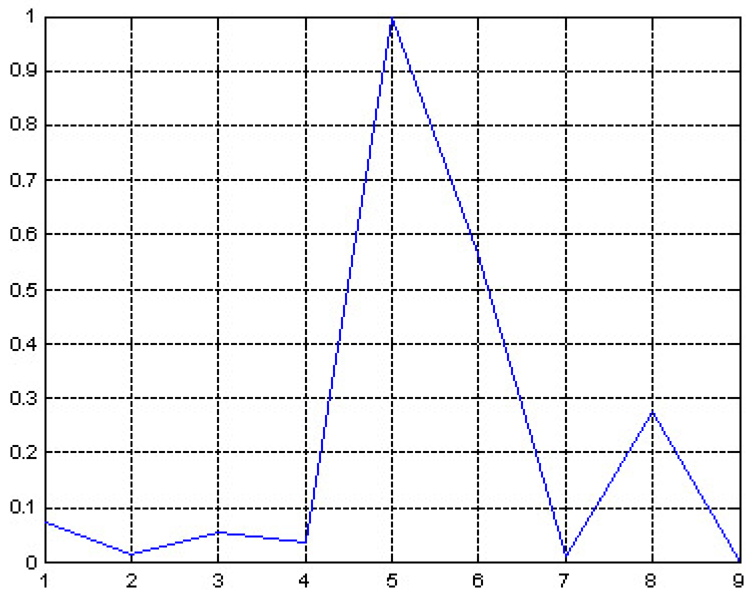


Figure 70: Correlation coefficient after different rotation angles. Number 5 represents situation in Figure 69 and 8 in Figure 68

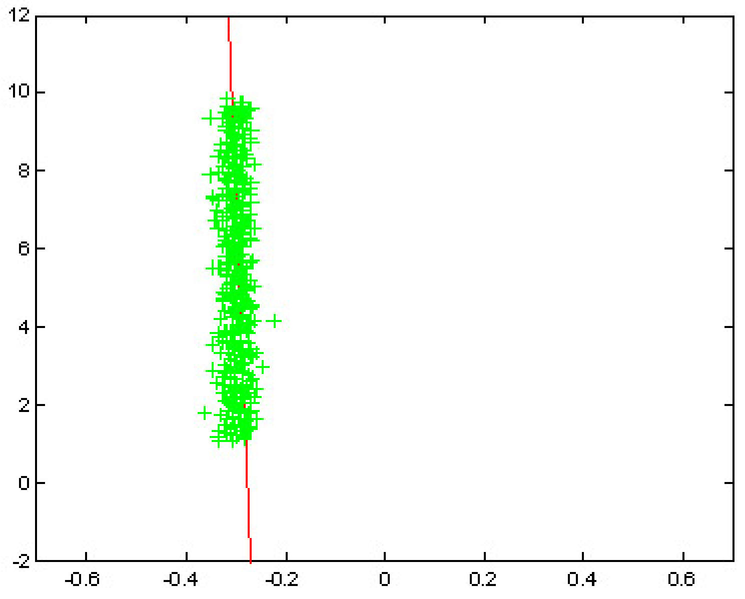


Figure 69: Optimal line fitting to data (modulus has packed data)

Appendix 3

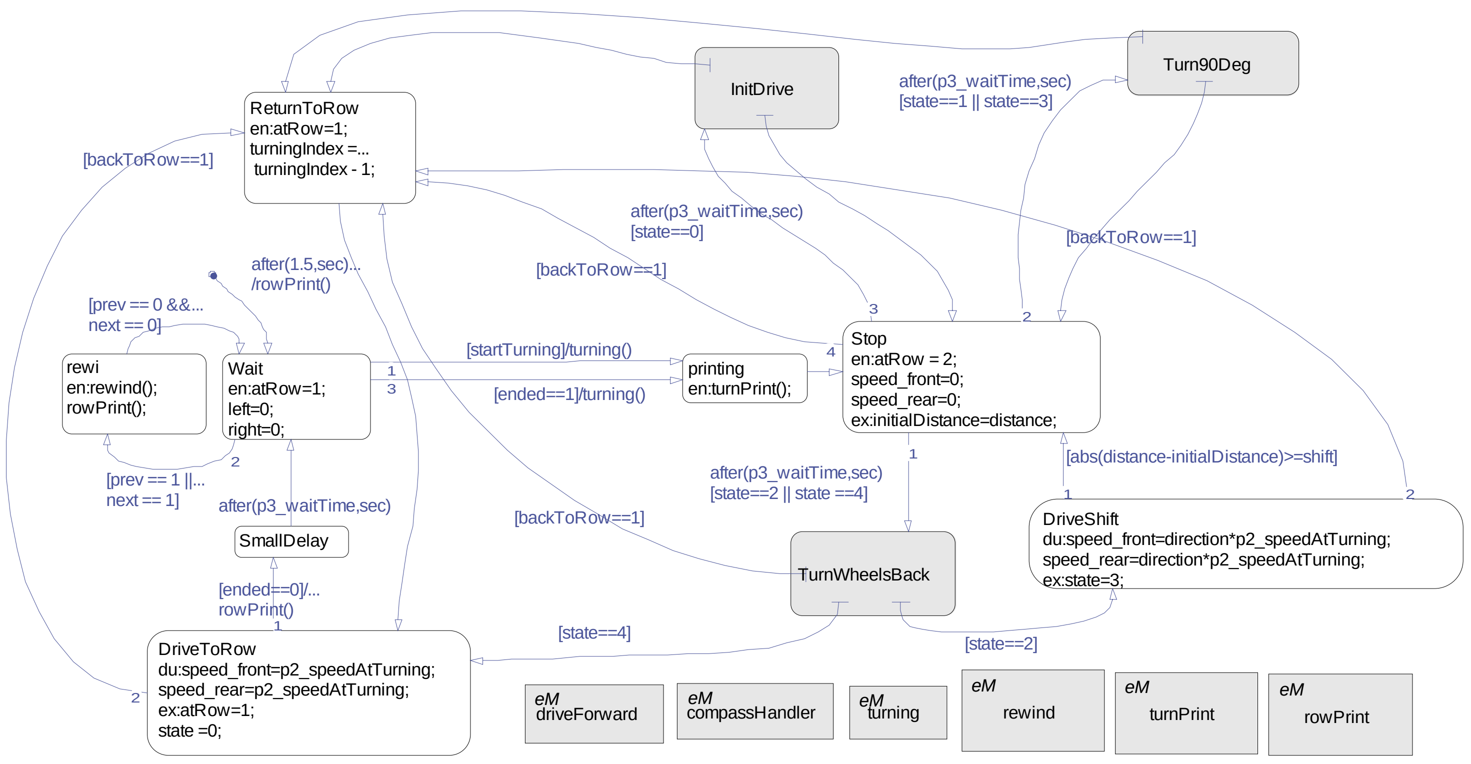


Figure 71: Stateflow model of turning logic

KaMaRo – an autonomous field robot

Thomas Hummel, Manuel Großkinsky, Patrick Rößler, Martin Hohberg, Marius Siegfarth, Marc Lang, Timo Hackel, Tim Najuch, Simon Pinter, Nicolai Hildebrandt

KaMaRo Engineering (e.V.)
c/o Lehrstuhl für mobile Arbeitsmaschinen
Fritz-Erler-Straße 1-3
76133 Karlsruhe

kamaro.engineering@googlemail.com
www.kamaro.kit.edu

Figure 69: Picture of the team in Braunschweig with the robot

Abstract

For the first time in history of the FieldRobotEvent a team of the Karlsruhe Institute of Technology is taking part in the FieldRobotEvent 2010 in Braunschweig.
The key feature of the robot KaMaRo 1 (Karlsruhe Maize Robot) is that it is able to navigate absolutely autonomous through the rough terrain of a maize field. For the several tasks of the FieldRobotEvent it uses some specific sensors to detect the surroundings and the weed, which has to be dealt with.
In this paper we describe our proceeding from the first idea to a working robot.

Keywords: field robot, laser scanner, webcam, autonomous navigation, weed detection, organisation

1. Introduction

We cannot refer to any robot we built before, because this is our first participation at the FieldRobotEvent. Therefore the first journey we did last year – after being introduced to this Event by Prof. Dr. Marcus Geimer – was visiting the FieldRobotEvent 2009 at the University of Wageningen to collect information about the challenge and the several tasks. Our team was then founded on the 9th of July in 2009.

The first step was organising the group of 19 students from different fields of study with no experience in developing robots. This took us some weeks (see next chapter). After these organisational tasks, collecting information and creating milestones were the main goals.
The next step was to search for support of the industry because you can't build a robot without parts and money. We also tried to become a registered society to speed things up and simplify the data flow, but we have been working on this for about three quarters of a year.

We decided to design the whole chassis by ourselves, because most of the team members are mechanical engineering students. We also decided to use a Laser Range Finder for navigational purposes and to use a camera for the weed detection.

In December 2009 we started with the software development. The assembly of the mechanical parts started around February.
The first testing of the code was done with a data generator and later with real sensor data from our small test field at the botanical institute.

2. Organisation

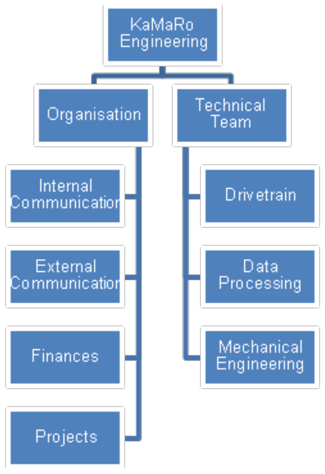


Figure 73: Structure of KaMaRo Engineering

2.1. Structure

Our team consists of 19 members which are mainly mechanical engineering students. There are also some electrical engineers and a few industrial engineers.
At first we formed smaller groups for the different tasks without strict separation.
Each team has a leader to enable communication in smaller groups at team leader meetings.

2.2. Internal Communication

To enable close collaboration we have a weekly non-obligatory meeting where results are presented and ideas are discussed. From time to time we have a full-team meeting where we discuss important issues which consider the whole team.

Figure 74: Weekly Meeting

Other ways of communication are e-mail, wiki and the cloud-service dropbox.com.

2.3. External Communication

External Communication is important for three different kinds of relations.
At first we have to present our team in public. This means: creating posters and flyers, maintain a website or writing press articles.
Then we have to communicate within the university: where do we get support, a room or a workshop. Here we have a strong binding to the Institute of Vehicle System Technology and especially the Chair of Mobile Machines which supports us in many ways.
Another task of the external communication is the sponsoring. A lot of helpful contacts were made resulting in free parts or exchange of know-how.

2.4. Projects

This sub team is responsible for one-time jobs. For example presentations, exhibitions and field trips.

Figure 75: Foundation Presentation

3. Mechanical Concept

3.1. Chassis

After many discussions we decided that our robot should be steered four-wheels and that all wheels should be driven. With that background we split up our construction team into five subgroups, each having a special specification. These five groups were: steering construction, axle and differential concept and construction, driving concept with power train, suspension and damper concept as well as housing concept. So each of us became a specialist in one topic.
In these groups we started with investigations about already existing concepts and parts we could use from model making cars. As a result of these investigations, we decided that our robot should have a live axle. Only the wheels should turn while steering. A suspen-

sion between the axle and the housing was chosen finally. So the concept for our robot was complete.
As a second step we started with the construction of our robot. As we found out, we could only buy the wheels and had to produce every other part. So we made the drawings with the CAD program Pro/Engineer and with the help Mobima workshop. Furthermore our supporting institute and the company Kurre produced the parts for us.
The last step was to create the housing. It had to contain the engine, the accumulators and the electronic components. We used aluminium profiles for the framework, which were covered with thin plastic panels. Also the electrical components were mounted on them.
During the construction we noticed that the wheels are not strong enough. The pressure in the wheels was too low. We tested different materials to fill the wheels and in the end we got the best results with rice.
The construction of our robot was finished one and a half weeks before the FRE event 2010.

3.2. Sprayer

The water reservoir is filled with water and air, which is compressed with a hand air pump. If weed is detected, the microcontroller sends a 12V signal to a magnet valve. The valve opens and the water is sprayed through the injector onto the weed.

4. Data processing concept

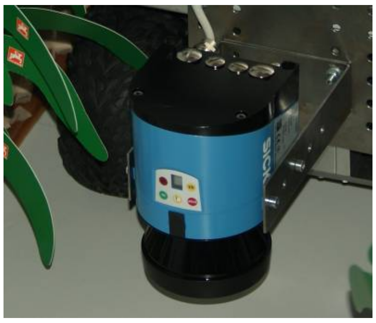


Figure 76: Sick LMS 100 Laser Range Finder

4.1. Sensor

A LIDAR unit (Sick LMS100) is used for the detection of maize plants. The functional principle of LIDAR (Light Detection and Ranging) is to send out a laser beam, pointed on a rotating mirror, and then measure the time it takes the reflected light to return.
The LIDAR has been mounted upside down to get the laser beam as close as possible to the ground and threby avoid recognizing leaves which might be hanging between the rows of plants.

When using a laser scanner for maize detection, mainly two restrictions must be considered:
Firstly, the impreciseness: Although the resolution of the measured distance is very high (<1mm), the error is up to +/-20mm. This means that the exact position of recognized objects may vary from one measuring to the next.
Secondly, the laser beam is actually a cone with an opening angle of 0,5°. As the light spot produced by the laser gets bigger with growing distance, the amount of light being reflected by a maize plant gets smaller. Therefore, plants can only be detected when they are near enough.

The laser scanner operates at a frequency of 25 Hz and a resolution of 0,25°, which means it is able to deliver a complete scan of the robot's environment every 40 milliseconds. As it has an opening angle of 270° it can see everything in front of and beside the robot.

The laser scanner communicates with the main program (written in Matlab) using the Ethernet port of the computer and the TCP/IP protocol. When the computer requests data, the laser scanner responds, sending a data package containing general information about the device, how to interpret the following data and then the latest set of measured data. When the data package has been received, the relevant data is extracted by the software and saved for further use (navigation). One set of data consists of 1082 data pairs (angle/distance).

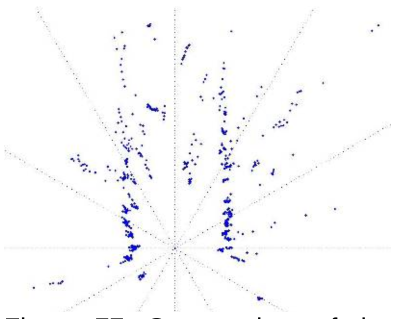


Figure 77: Sensor data of the test field

4.2. Mainboard and Microcontrollers

4.2.1. Hardware

The main controlling hardware unit is a miniITX board with a Core2Duo 3GHz CPU and Matlab as the commanding software. The hardware equipment provides the needed performance to run the greedy navigation and image recognition algorithms and has enough reserves for future development. On the software-side Matlab offers the potential to implement most of the needed features itself and the flexibility to include specialized external components.

Connected to the PC is an underlying hardware unit consisting of an Atmega32 microcontroller, which handles the LEDs for weed detection, the push-buttons for external access and the valves for weed control.

4.2.2. Power Supply

The power supply for the electronics is one 12V/7.2Ah lead-gel-battery. A DC/DC-converter is connected to ensure a stabilized voltage for the miniITX board and the Lidar. The microcontroller board is supplied directly by the battery, as well as the valves. Under normal conditions the power supply is able to run the electronics for circa 15 minutes.

4.3. Software

4.3.1. Weed Detection

Hardware:

For detecting weed, both the Logitech Quickcam Pro for notebooks and preprocessed data from the LIDAR are used.

Software:

The core of the algorithm is Opencv's Haar-detection, a cascade-detection-system which is usually used for face detection [4]. To achieve weed detection the following things have to be done:

1. Building a weed-cascade can be done by executing Opencv's “Haartraining”-algorithm, a machine learning algorithm, with approximately 1500 positive weed-samples and as much negatives. To detect differences between grass and flowers a second cascade for detecting flowers has also been created.
2. Optimizing the detection algorithm: For a reduction of detection time it is tried to detect weed only between the two maize rows on the robots left and right. Therefore the detection algorithm is turned from scale-invariant to scale-variant. This means that the weed is not detected in every theoretical possible size any more.

After detecting weed the weed gets tracked to reduce errors.

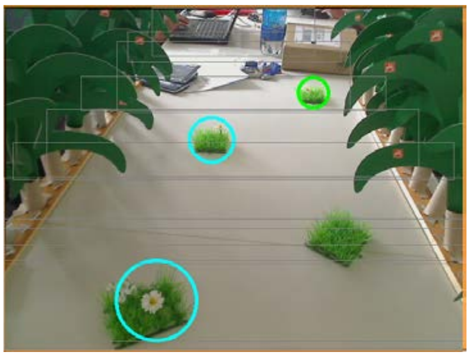


Figure 78: Weed detection with Opencv's Haar-detection

Figure 79: Detecting beer glasses for a sponsor

5. Navigation

Infield navigation

The navigation algorithms were programmed in Matlab. The data delivered by the laser-scanner is used to determine the necessary steering angle to keep the robot in the middle of the left-hand and right-hand row. The program runs through the following steps:

1. The data (distances and angles of obstacles) is received and transformed into Cartesian coordinates.
2. The relevant area is cut out so that only obstacles in a specific area in front of and beside the robot are considered during navigation.
3. Detected points that don't have other points within a specified radius are deleted. Single points are not considered as plants and therefore not used.
4. Two straight lines are determined using the RANSAC algorithm [3]. The lines represent the left-hand and right-hand row.
5. The slope of the straight lines is supposed to vary only within a limited span. If the slope is too big or too small, the calculated line is considered as defective. One suitable line is enough to navigate through the field and therefore missing plants on one side are no problem.
6. Now the steering angle can be calculated based on the offset between the robot and the estimated lines.

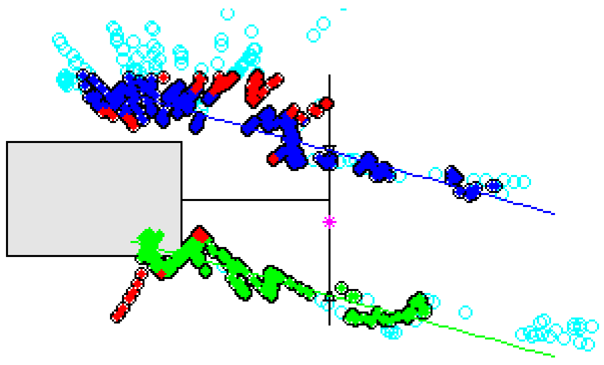


Figure 80: Infield navigation: grey: robot; blue and green: points used to calculate the lines; red: points considered as outliers by RANSAC algorithm; pink: center between both rows (used to calculate offset); cyan: unused points (single points or out

During tests with artificial and real maize, the navigation algorithm was optimized. Especially the dimensions of the area which is used to detect the rows turned out to be very important. Areas in the shape of rectangles, semi-circles and other variations were tested. The best results were achieved with an area consisting of a rectangle with a half ellipse added at the front side.

The fact that detected plants on just one side of the robot are enough to calculate the necessary steering angle makes this algorithm reliable, even if there are larger gaps within the plants. Problems can occur if there are a lot of leaves hanging into the free area between the rows. Especially leaves positioned near the laser-scanner can falsify the data which is used by the program significantly.

Headland navigation

The first attempt was to enable the robot to drive along the headland and to keep the same distance from the rows. Thereby it was counting the rows which are passed using the RANSAC algorithm. The goal was to detect lines in the received data from the laser-scanner. However, this method was soon considered to be not reliable enough. One reason for this decision was that the robot moves perpendicularly to the rows and therefore only the first few plants of a row are detected while the laser beams cannot reach the plants hidden behind them.

A second program was developed which doesn't make use of RANSAC. The robot has two imaginary areas in the shape of rectangles besides itself. While driving along the headland, the amount of points, which represent obstacles found by the laser-scanner, are checked permanently in the two areas. If the amount of points within the first area exceeds a defined threshold, the robot knows that it is positioned right next to a row. If the points in

the second area, which is located behind the first area, exceed the threshold, the robot assumes that it has passed the row and watches out for the next one.
In tests with artificial maize, this program worked accurate and was reliable. Even if the rows are not exactly perpendicular to the driving direction of the robot, the rows are detected and counted.

In both, infield and headland navigation, a new steering angle is determined about three or four times a second to enable the robot to move accurately through the maize field without damaging any plants.

Remote Control

The remote control is using a wireless LAN connection between the robot and a computer. TCP/IP is used as communication protocol. The communication is programmed with Java, because there are special packages for TCP/IP, so it is easy to create a communication between a client and a server.
Furthermore it is possible to import self-programmed Java classes in Matlab. Importing the self-programmed Java class for TCP/IP communication is essential, because our main program runs in Matlab.

So there is the robot, which is the client, and a computer, which acts as the server.
Commands are sent in form of integers from the computer to the robot.
Every command has a specific number. For example: “3“ is sent to accelerate the robot.

The only problem is, that Matlab does not support several threads. Two functions cannot run at the same time. For example one function which handles the TCP/IP communication on the one hand and another function for the navigation cannot be executed parallel.
This can be solved by requesting a message from the server after the main function was called by Matlab. Since the main function contains an infinite loop it is easy to combine it with the „network-function“, because the „network-function“ can be integrated into the infinite loop, so that there is always a request for a message, every time the main function is run.

In addition it is useful to know what the robot is actually doing if you control it by remote control. So there is a graphical interface, which displays the speed, the wheel angel and the different types of modes (autonomous driving, driving by remote control, idle).
This graphical interface was built with Swing, a graphical package for Java.

6 Drivetrain Concept

Since there was no back up driveline, it was finally decided not to develop the power electronics by oneself but to take tested parts, in order to keep the risk of driving motor problems low.

6.1. Motors

A "Dunkermotoren BG 75x25 SI" with 250 watts is used as the drive motor. A speed controller and all the power electronics are already part of it. Front and rear steering are built up identically: a "BG 31" servo motor, an "RE 30" encoder and a 128:1 planetary gear, all made by Dunkermotoren, are used in this robot.

6.2. Energy supply

The energy for the power electronics is provided by two serially connected 7,2 Ah / 12 V lead batteries. 12 V and 5 V supply for the signal electronics is taken from the ITX board supply.

6.3. Hardware

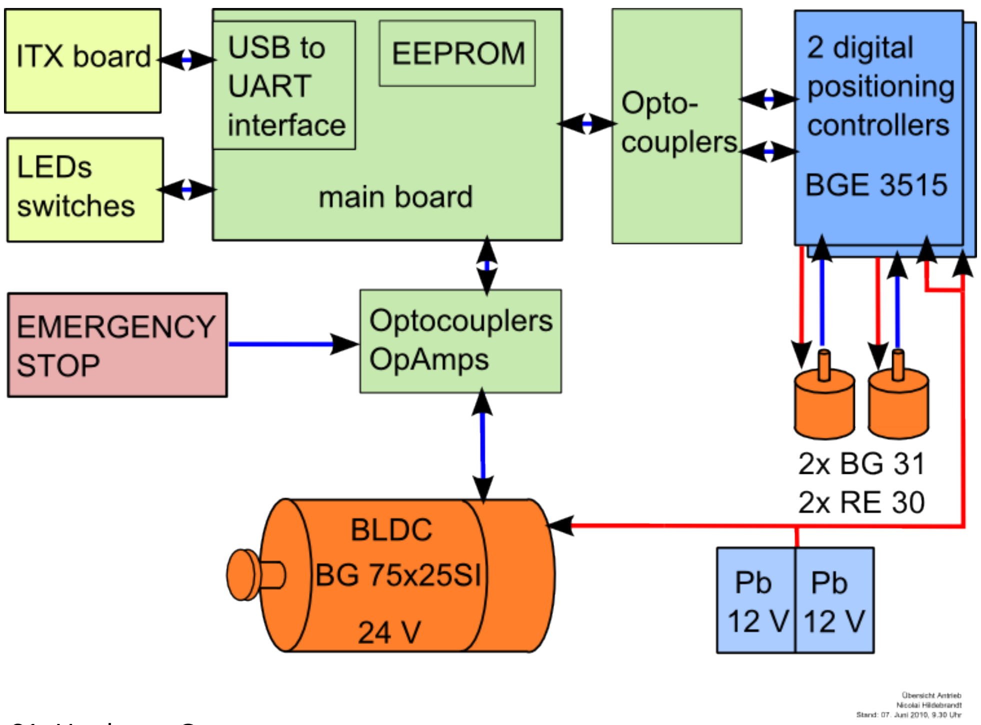


Figure 81: Hardware Concept

The hardware is built as modular as possible, so that most parts can be reused in further development or replaced in case of damage. Despite of the "Dunkermotoren BGE 3515" digital positioning controller, all electronics is self made

The microcontroller board is kept as simple as possible. The Atmel Atmega32 microcontroller is connected through an FT232R USB to UART interface chip to the ITX board. It proceeds the commands and motor error signals and controls the motor controllers. Extra boards with optocouplers are used to connect the microcontroller board to 24 V electronics. Further, an operational amplifier circuit is used to generate a 0 to 10 V signal for the drive motor speed controller on the drive motor optocoupler board. An external EEPROM is kept as a hardware option for further development needs.

An emergency stop button is directly connected to the drive motor optocoupler board in order to definitely put a stop signal on the motor controller input lines. Robot movement is then disabled irrespective of the signals from the ITX board or the driveline microcontroller and can be enabled as soon as the computational problems have been solved. Also, safe stop signals are provided by the optocoupler circuits in case the microcontroller board is disconnected.

All software is written in C.

7. Discussion

7.1. Mechanical

On the FRE our robot drove for the first time. So unfortunately we didn’t know whether all mechanical components work in the way they should. At the FRE, where we were resticted to react to potential problems.
The first problem we noticed was, that the wheels had to much steering-resist, because we had no differential in our robot. To correct this, we decided, that the front wheels should only be used to steer and the back wheels should drive. With this solution we got the robot ready to drive.
In the middle of the night, a connection between two driveshafts transmitted no more turning moment, because a setscrew had cut into the shaft and lost it transmitting function. So we had to repair the robot with our limited equipment. During the sunset we finished the repair and continued with the testing.
After the FRE some of our team drove to the DLG-Feldtagen, where the teams were invited to demonstrate their robots. During these days, we talked with many cultivators and with producers of agricultural technical equipment. In the conversations we learned a lot about potential operational areas in the agriculture and about possible tasks for the freestyle task. Especially the eco-farmers were very interested in the idea of mechanical weed control. Furthermore the discussion with the other teams improves our knowledge about special components of the robots.
So all in all we made many experience and we will use them to improve our robot and later to design a new one.

7.2. Drivetrain

Problems:
The drive motor worked well and was more powerfull than expected. The driveline control worked fine as well. The two steering servo controllers were damaged one week before the contest but gladly one spare controller could be found so the robot could run with one steering servo. The front wheel steering printed circuit board had a flaw but could easily be replaced by the spare board of the rear steering. The controller parametrisation took too long because of a missing password, but luckily Team FREDT from Braunschweig could help us. Yet it is not clear to us whether the steering servo was strong enough in combination with steering and four wheel drive system as it was planned but changed the morning before contest.
Possible improvements:

The drive speed control signal values could be made continuous by a smoothed pulse width modulator output. Further acceleration ramps could be programmed in the drive control microcontroller. Weight could be reduced by the use of high energy density batteries. Some more fuses on printed circuit boards.

8. Conclusions

Finally it was a great honour for everybody to take part in this event. Even if the robot didn’t work as good as expected the weekend in Braunschweig had been one of the most exciting and useful ones for the past year.

There was a great exchange of knowhow and experience on a very high level with all the participants and many difficulties were mastered due to the acute work on the robot and the help of other teams.

At least the event formed our team in a very special way so that we will be able to cooperate much better and take part at the next event with a competitive robot.

Acknowledgments

We are very glad to thank the employees of the mobima institute at the Karlsruhe Institute of Technology as well as their development association WVMA who supported us financially and with know-how. Special thanks go to Phillip Thiebes who is our main advisor. Also we got great support from the IFL, MRT and the Botanical Institute.

Without the help of our sponsors the realization of our robot would have not been possible. Therefore we thank the Sick AG for their laser range scanner, Dunkermotoren for their engine parts and Mädler for several chassis elements. Not to forget the W und A Kugellager GmbH with their ball bearings and the Adolf Pfeiffer GmbH which sponsored us our tool kit.

Our financial needs were covered by the CLAAS Foundation as well as Fendt (part the AGCO GmbH).

WVMA
WISSENSCHAFTLICHER VEREIN FÜR MOBILE ARBEITSMASCHINEN
SICK
Sensor Intelligence.
Pfeiffer
WIR ERZEUGEN BEWEGUNG
CLAAS Stiftung
W A
WÄLZLAGER- UND ANTRIEBSTECHNIK
MOBIMA
LEHRSTUHL FÜR MOBILE ARBEITSMASCHINEN
IFL
FENDT
mrt
Institut für Mess- und Regelungstechnik
dunkermotoren
advanced motion solutions
MÄDLER
GEGRÜNDET 1882
Botanische Garten des KIT

Figure 82: We want to thank our supporters and sponsors

We want to emphasize especially the great effort and help of Dunkermotoren. They gave advice on choosing the right parts, helped with the use of them and at least replaced some defect parts in not time during the last week before the contest.

Last but not least there were many companies which gave us discount on their products.

References

[1] Dropbox http://www.getdropbox.com (2010-05-25)

[2] Media Wiki http.//www.mediawiki.org (2010-06-07)

[3] http://vision.ece.ucsb.edu/~zuliani/Code/Code.html (2010-06-07)

[4] “Learning Opencv” from “machine vision” from Field Robot Event 2008 - Proceedings

Team Hohenheim / Osnabrück

Phillip Bernhardt, Klaus Meissner, Daniel Mentrup, Markus Pesch, Daniel Kinski

Universität Hohenheim, Faculty of Agriculture, 70593 Stuttgart, Germany, Phone +49 711 459-22491, Email: meissner@uni-hohenheim.de

Abstract

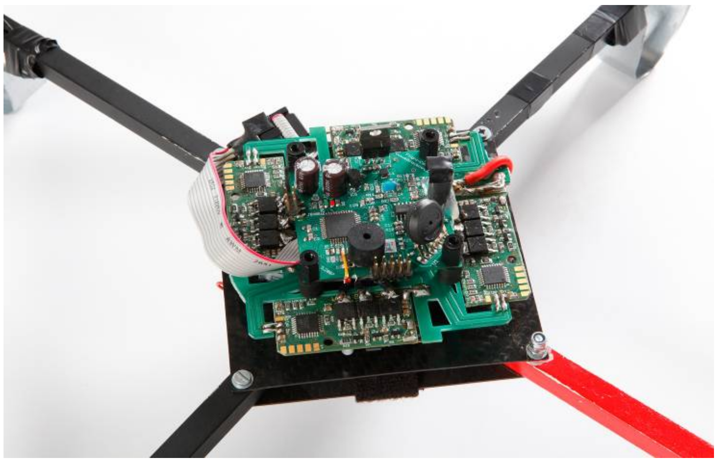
The Optikopter was developed for the Field Robot Event 2010 by students of the University Hohenheim and the University of Applied Sciences, Osnabrück. The system is based on an open source Quadrokopter, which was modified for the event. For navigation in the air the Optokopter uses GyroScope (3x), Acceleration Sensor, Compass, GPS-System. For the Cooperative Challenge a 2,4 GHz-System has been added.

Keywords: Quadrokopter, Field Robot, Cooperative Challenge, GPS,

1. Introduction

The **Optikopter** (general quadrocopter) counts, like a helicopter, to the vertical take-off and landing aircrafts (VTOL).

A quadrocopter has four propellers (two left-handed and two clockwise).
In this way the torque is balanced. So the quadrocopter can float stable in the air, is analyzed by the control electronics including position sensors (gyros).
These gyros measure the speed of rotation around each axis (roll, pitch, yaw).



The Field Robot Event 2010 was organized by the Institute of Agricultural Machinery and Fluid Power from the Technische Universität Braunschweig and took place in Braunschweig on June 11th -13th
There were competitions in five, the tasks were:

Basic
Within three minutes the robot has to navigate through long curved rows of a maize field to cover as much distance as possible. On the headland it has to turn and return in the adjacent row. There will be no plants missing in the rows.

Advanced
The robot should cover as much distance as possible within 3 minutes while navigating between straight rows of maize plants. It should be able to follow a certain pre-defined pattern over the field.

Professional
The Task consists of two subtasks. First the teams will have to demonstrate their weed handling device and explain to a jury and the audience how it works. Afterwards they have to demonstrate their weed detection system.

Cooperative Challenge
Cooperation between one or more robots has to be demonstrated. There is no given task the robots have to fulfil. The robots can drive, fly or even swim (if it is raining cats and dogs). The application should have an agricultural background and has to be shown on the field.

Freestyle
Robots are invited to perform a free-style operation on the field. Fun is important in this task as well as an application-oriented performance. One team member has to inform the jury and the audience about the idea.

2. Mechanics

2.1 Motors

In general, a Quadrokopter needs **four** brushless motors.
We use Outrunner Brushless Motor (1200Kv)

- Shaft diameter: 3mm
- Shaft Length: 46.5mm
- Dimension : 46.5mm x 31mm
- Recommended Propeller: 10x5, 10x4.7
- Weight: 34g
- Kv: 1200rpm/V
- Max Currect: 15A
- Max Trust: 745g

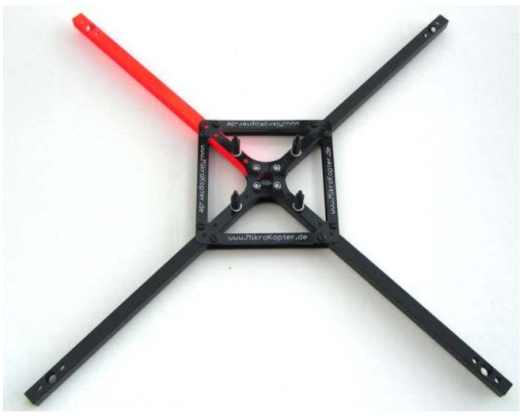
2.2. Propellors

Two Clockwise (CW) propellors and two CounterClockWise (CCW) propellers are needed.

- Front = Motor #1 and Back = Motor #2 clockwise
- Right = Motor #3 and Left = Motor #4 counterclockwise

2.3. Frame

The frame is a constructed of aluminum square profiles. The motor-motor is 40 cm .

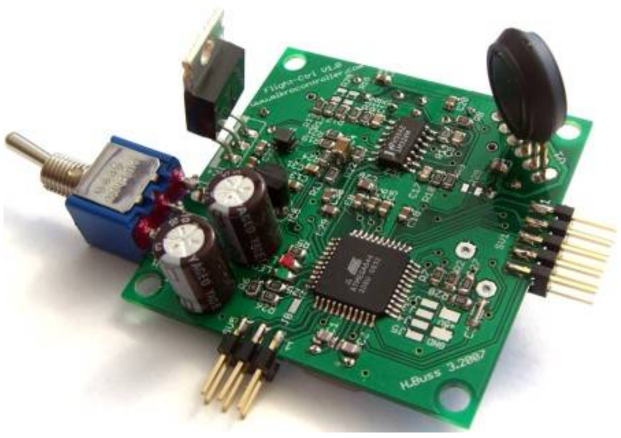


2.4. Battery

Usually Lithium-Polymer batteries or LiPos are used as the main power source.

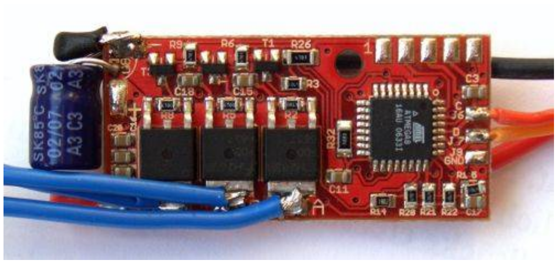
3. Hardware concept

The Flight Control



The Flight-Control (Flight-Ctrl) is the main board of the MikroKopter. It contains the main processor and all the sensors that are necessary for a stable flight.
Of all sensors, the rotation speed sensors are most important. The software uses them to determine the position in the air and to compensate for external influences. For every axis (x,y and z) a rotation speed sensor is needed, so three sensors all together. Usually these rotation speed sensors are called Gyroscopes or Gyros and they measure changes in degrees per second.
Another sensor is the acceleration sensor. It senses the acceleration in all of the three axis. The vertical acceleration sensor is also able to measure the angle of the mikrokopter towards the earth. Usually they are referred to as Accellerometers or ACC. You can fly without them, but with these sensors you are able to automatically get the MikroKopter back to level flight. This way you can let go of the joysticks and the MikroKopter will stay at its position. Without these sensors, the Mikrokopter will keep on flying in a prescribed angle. This kind of flying is called "Heading Hold".
A height sensor can be placed on the Flight-Ctrl as an option. This gives you the ability to keep flying at the same height all the time.
When hooked up to a PC, the Flight-Ctrl can be read and set with the MikroKopter-Tool.

The Brushless Controller



In total there are four Brushless Controllers (BL-Ctrl) needed for the MikroKopter. Each controller controls one brushless motor. Brushless motors don't use brushes for energy-distribution to the rotor. In contrary to the motors with brushes, the magnets are rotating while the coils do not. That's why you can't use a DC current, AC current with precise pulses is needed to drive the motors. This AC current is provided by the BL-Ctrl.

A standard Brushless Controller will not work in a MikroKopter. The BL-Controllers are connected to the main board with a bus system (I2C) and each controller is given a unique address that is used by the main processor to communicate with the BL-controller. It is possible to control the BL-controller with a standard RC-receiver. PPM signal input is available (but not for the use in a Mikrokopter).

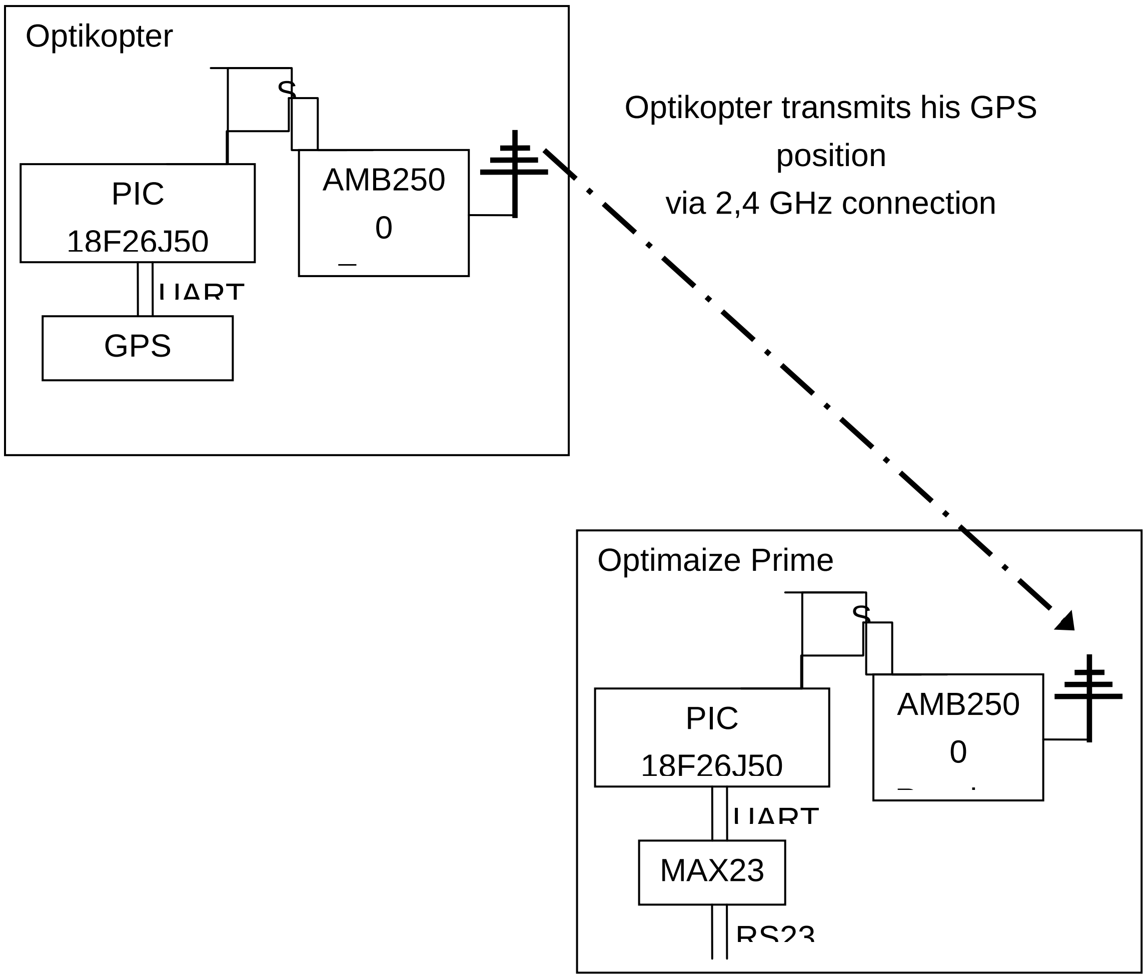
Features:

- Controller: AVR ATMEGA8 of Atmel
- Placed with six 60 Amps MosFets
- integrated current measurement
- Current limitation at the DC current-side
- Designed for approx.110W at 11.1V or 150W at 14,8V (10A contineous)
- Peak current approx.220W at 11.1V or 300W at 14,8V (20A peak)
- two LEDs (Okay and Error)
- Battery voltage sensing with low-voltage detection
- Software is totally in C
- As setting point either the rpm can be controlled or set (per PWM)
- several inputs for setting point
- Size (L x H): 43mm x 21mm

The actual version 1.1 is available as emty PCB or with all components in place. The older version 1.0 isn't available anymore. The old version needs a separate firmware for each controller with separate addresses. Since version 1.1 the addresses are adjustable via jumpers (shortcircuits).

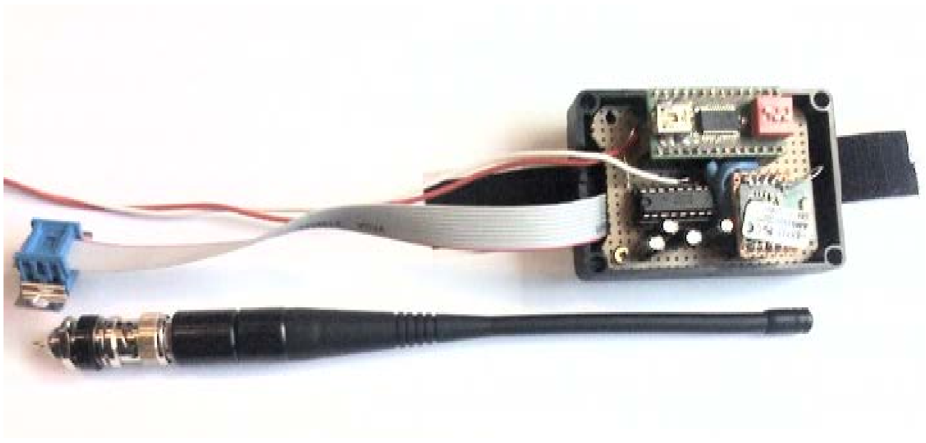
3.1. Communication Unit

In order to establish a radio communication between Optikopter and Optimaize Prime, two different communication units were developed. The communication unit of the Optikopters receives the data of its current position and sends the raw data to the communication unit of the Optimaize prime. Over a flag communicates the Optikopter the Optimaize prime, when it is to move toward the Optikopters GPS position. The figure below is showing the schematic of the communication with a PIC 18F26J50 as the MCU where the program is running and an AMB2500 as the receiver and transmitter. The onboard communication was realized over serial communication busses like UART and SPI. The Optimaize Prime communication unit transmits the data over the RS232 interface to its onboard PC.


Optikopter
PIC
18F26J50
AMB250
0
UART
GPS
Optikopter transmits his GPS
position
via 2,4 GHz connection
Optimaize Prime
PIC
18F26J50
AMB250
0
UART
MAX23
RS23

3.2.

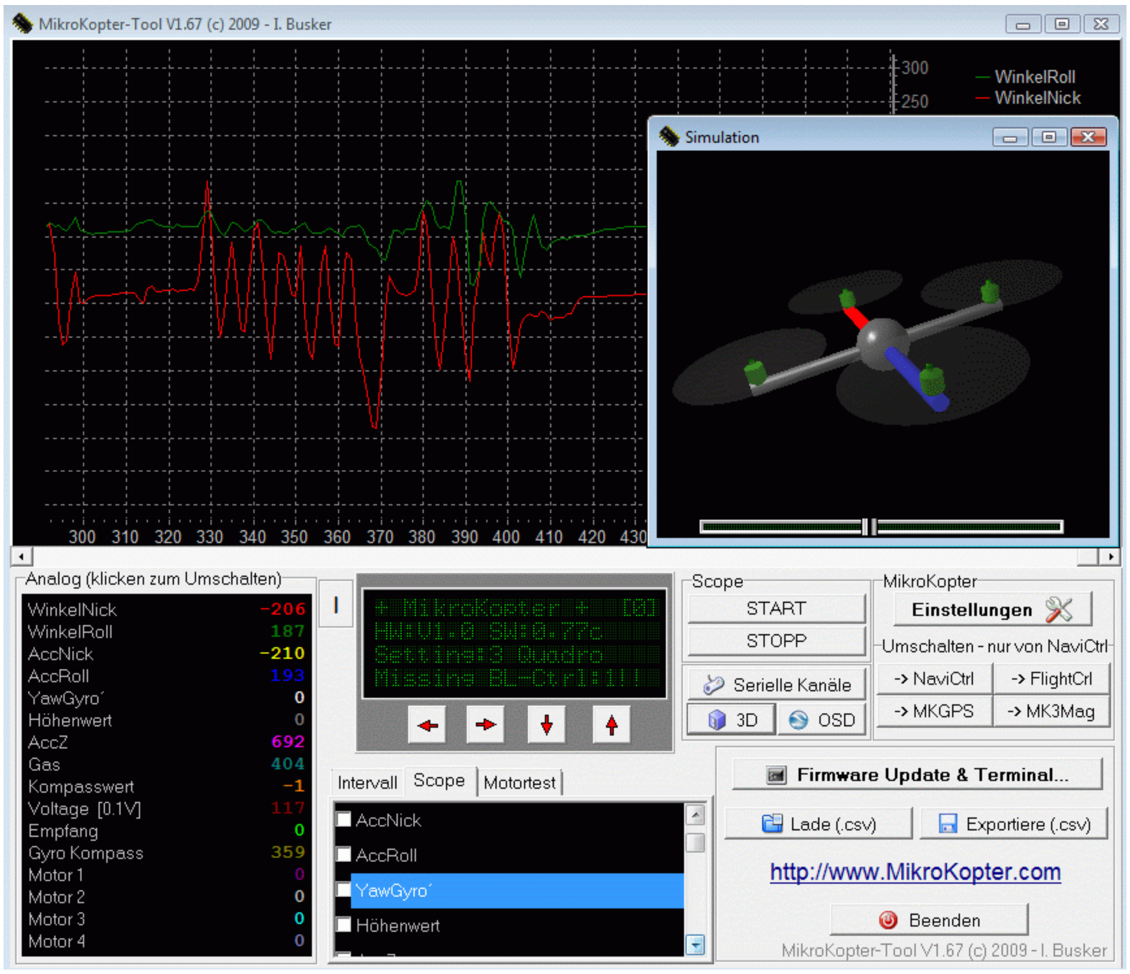
The next figure shows the communication unit of the Optimaize Prime with the PIC MCU, the AMB2500 and the MAX232.



4. Software concept

Computing and processing is accomplished by a Atmel ATMEGA644 @ 20MHz. It was connected to four Motor-Control-Boards (Brushless-CTRL) to operate with the Motors.
We connect the Motor-Control to the I²C- bus, which carries the command sequences. The Flight-Ctrl needs the special brushless ESC, to ensure fast communication via the I2C Bus. The I2C Bus has a clock (SCL) and data (SDA) line. The bus connects all SCL and SDA lines together.

MikroKopter Tool
We use the MikroKopter Tool, a GUI, to receive and transmit data from the OptiKopter and to program the Micro-Controllers on the circuit-boards . It allows us to write different settings to the Optikopter and set Waypoints for the GPS-Module.



Analog (klicken zum Umschalten)

WinkelNick	-206
WinkelRoll	187
AccNick	-210
AccRoll	193
YawGyro´	0
Höhenwert	0
AccZ	692
Gas	404
Kompasswert	-1
Voltage [0.1V]	117
Empfang	0
Gyro Kompass	359
Motor 1	0
Motor 2	0
Motor 3	0
Motor 4	0

```
+ MikroKopter +    [0]
HW:V1.0 SW:0.77c
Setting:3 Quadro
Missing BL-Ctrl:1!!
```

Communication Software

To realize the communication with the communication units there was a program developed for the Optimaize Prime communication unit and another program for the Optikopter communication unit. The Optikopter's program receives the GPS-Data and sends it in a FIFO-Queue to the Optimaize prime. The Optimaize Prime's program calculates the angle with the GPS-Data to turn around and drive to the Optikopter's position.

5. Acknowledgments

Moreover the Team would like to thank Arno Ruckelshausen for making the participation at the field robot event possible. Special thanks go to Andreas Linz and Ralph Klose for their advice and assistance in realization of this project.

6. References

[1] http://www.mikrokopter.de, state June 2010

[2] Proceeding 6th Filed Robot Event 2008 / Team Aggronaut, state June 2010

[3] http://www.cmucam.org, state June 2010

[4] http://www.ifm.com, state June 2010

Team Optimaizer

Nils Feldkämper, Hendrik Hufendiek, Hannes Jahn, Christoph Kampmeyer, Simon Kerssen, Christoph Lemke, Dominik Mosler, Philipp Rave, Silvia Simon, Mihaela Tilneac, Jens Westerhoff, Erik Wunder

University of Applied Sciences Osnabrück, Faculty of Engineering and Computer Science/ Interdisciplinary Research Center Intelligent Sensor Systems (ISYS), Albrechtstr. 30, 49076 Osnabrück/Germany, Phone +49541 969 2090, Email: a.ruckelshausen@fhos.de

Field Robot Team
Fachhochschule Osnabrück
University of Applied Sciences

Fachhochschule Osnabrück
University of Applied Sciences

1. Abstract

The Optimaize Prime was developed for the Field Robot Event 2010 by students of the University of Applied Sciences Osnabrück. Thereby the development follows a new concept to create a robust platform with high modularity which will be further enhanced for further events. The Robot is based on the Volksbot-system developed by the Fraunhofer institute and was modified to fulfil the needs for this event. For navigation a Sick laser scanner (LMS 100) and two 3D cameras (IFM Effector) are used. The weed detection is realized by two CMOS cameras (CMUCam3) combined with a weed killing device which sprays the detected weed with a liquid. The main control-system of the robot with enough computing power is an ITX PC-Board with an Intel Core i5 Processor, handling the steering of the robot and data analysation of the sensors.

Keywords: Autonomous, Field Robot, Maizerowdetection, Sick LMS 100, CMUCam3, IFM Effector 3D, Volksbot

2. Introduction

The University of Applied Sciences Osnabrück participates in the Field Robot Event 2010 with the Optimaizer Team which consists of twelve people from different departments. Most members of the team are bachelor-students of electrical engineering, technical informatics, master students of mechatronics systems engineering. Additional support was provided by two exchange students from Rumania: Silvia Simon, master student, and Mihaela Tilneac, Ph.D. student, enrich our team. Good Advice and helpful support was given by our advisors Ralph Klose, Andreas Linz and Arno Ruckelshausen.

For most members of the team it is the first time they participate at the field robot event. Furthermore the designed robot is a completely new setup. It follows a concept of setting up a robust platform which will be continuously enhanced during the next years. Thereby it is achieved that in next years less chassis work is to be done and more time can be spent on the intelligence of the system (algorithms) to enhance navigation, weed detection and data analysation. In this way more development of future technology can be realized.

3. Mechanics

One of the most important parts that compose the robot is the chassis. The body is a frame construction made of aluminium threads and covered with sheet metal. The construction consists of a lower part and an upper part, which forms a fold-away housing. The lower part contains the motor controls, the dc-motors with chain-drive and the power supply. The laser scanner is mounted on the front end, a vertical thread which is also mounted on the front end holds two 3D cameras. Furthermore the PC is positioned on the lower part (see Figures 1 and 2). The operators of the robot can get access to the power supply via a flap at the rear end of the robot.*Figure 2.80. Section Cut of Optimaize Prime*

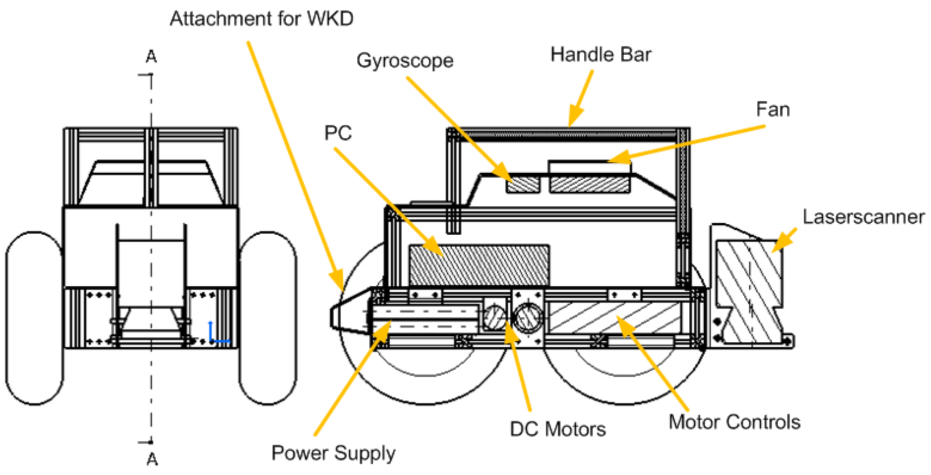
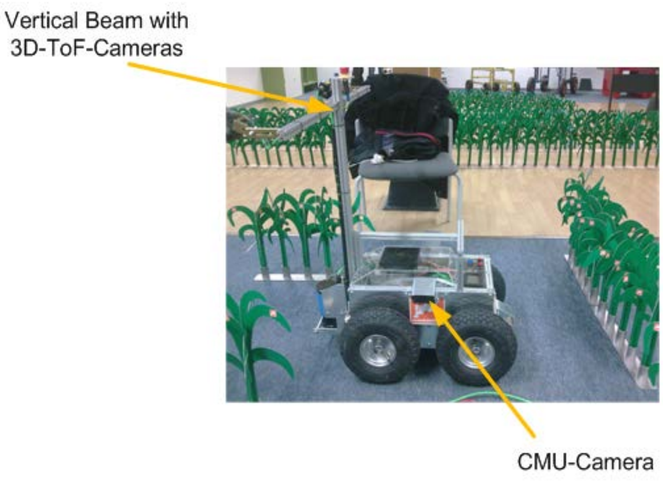


Figure 81.2. Position of the cameras



The upper part contains the fan and the gyroscope which are fixed in an acrylic glass housing on the top of the robot. A handle bar is also part of the fold away housing. Two CMU-Cameras are fixed at the side of the upper part.

The weed killing device is placed at the back with a special attachment that can be removed quickly.

4. Hardware concept

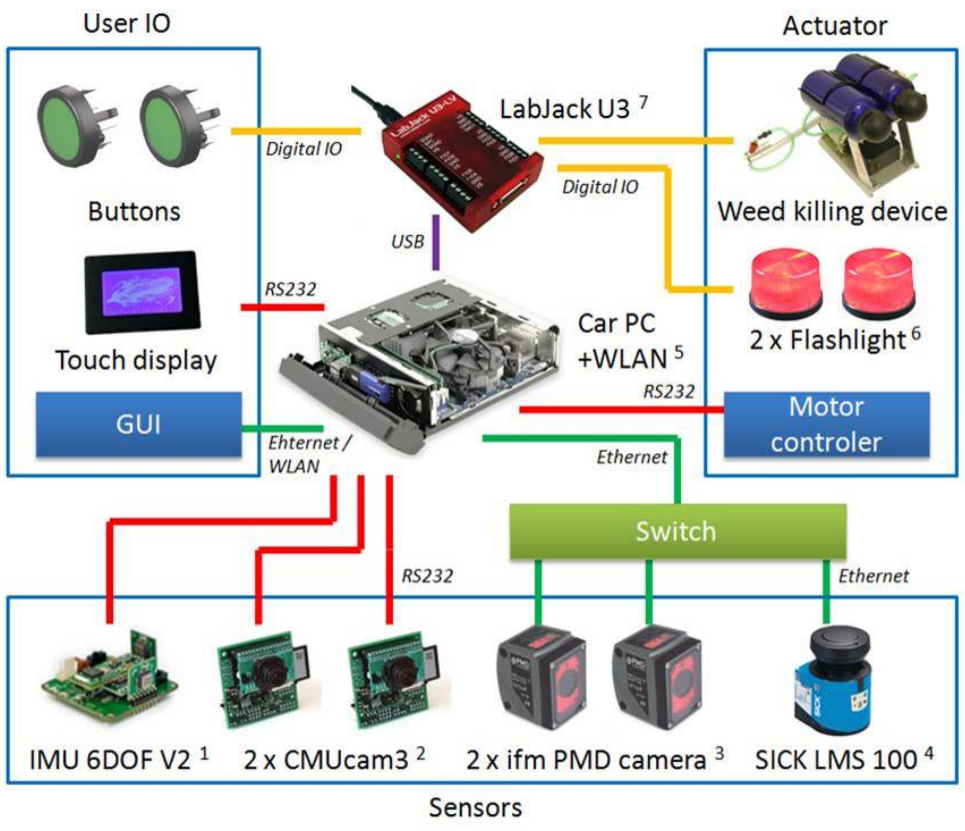


Figure 4.1. Hardware concept overview

4.1. Sensors

Gyroscope

To detect if a turn at the end of a row has reached the required angle the robot uses an analogue device, an ADXRS300 type gyroscope. This sensor is measuring the rotational velocity around its yaw-axis. Additionally the gyroscope offers a reference voltage and a measurement of the temperature, which can be used for compensation. A Microchip PIC 16F88 microcontroller is used for processing the velocity readings and also for communication with the main system.
The rotational velocity is integrated to get the angle. If the calculated angle reaches the target-angle, a signal is sent to the main system.

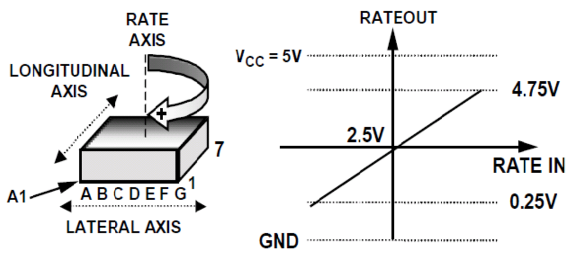


Figure 4.2. Functioning principle of the gyroscope [1]

Laser scanner

For navigation trough the maize rows the robot uses the information received from a laser scanner (Sick LMS100).

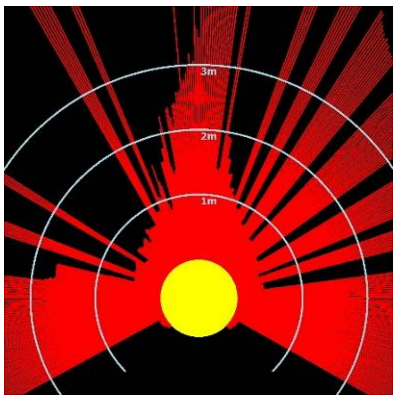
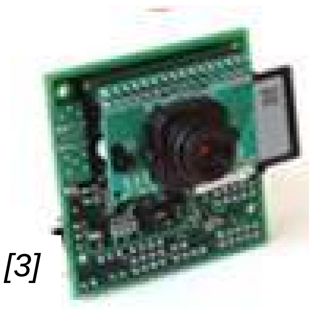


Figure 4.3. Laser scanner LMS100 and navigation principle of the laser scanner [2]

CMU Cameras



For the weed detection task, two CMU Cameras are used. The advantage of the CMU camera is that it's freely programmable and the image processing can be performed direct on the camera.

Figure 4.4. CMU Smart Cam [3]

Technical data:

- RGB image sensor OV6620
- Color areas: RGB, YCrCb, HSV

- Resolution: 352x288 pixel
- ARM7-micro controller LPC2106
- Interfaces: UART(RS-232), GPIO, Analog
- Power : 6V – 15V

For our purposes we use the RS-232 interface to transfer data to the Robot.

3D Time-of-Flight cameras

Two 3D Time-of-flight (ToF) cameras are used for robot navigation. The advantage of Time-of-Flight cameras is their ability to generate real-time images of all three dimensions. We used the IFM PMD 3D-camera with a frame rate of 25 Hz, a measurement range of 7.5 m, a resolution of 64x48 pixels and an angle of view of 40º / 30º.

Figure 4.5. 3D Time-of-flight camera [4]

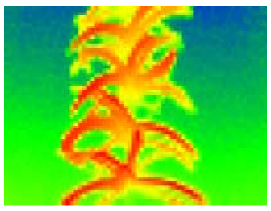


Figure 4.6. Crop row in a 3D-camera image

4.2. Power-switching and fuse-protection Board

This board is used to switch on and off the main-power supply by a high current relay, as well as the motor-power. The motor-power is switched by a second relay, thereby it is possible to only interrupt this power rail when the emergency-stop is hit. A complete power-interruption is unfavorable, because some parts of the equipment need several minutes to power up completely. Even more important is that an uncontrolled shutdown of the installed computer could lead to file-system corruption which could result in an inoperative system.

To energize the system, a button has to be pressed, which turns on the main relay. Thereby the computer starts up and the USB-ports provide power. This power is then used to bypass the button and keep the relay switched on.

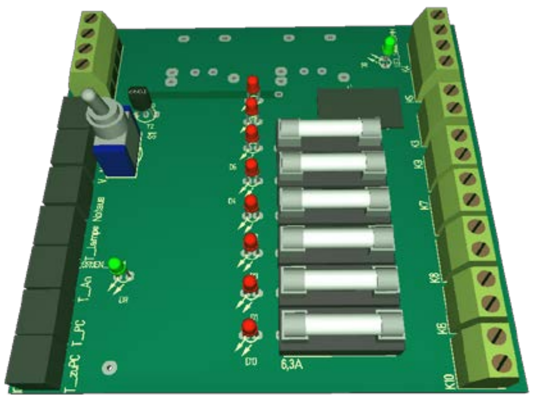


Figure 4.7. 3D Model of the board

If now the computer is shut down, the USB-power is automatically turned off after controlled shutting down the system. So the relay will be turned off when the computer has finished shutting down and the system has been safely turned off.
The motor-power rail is switched by the main relay but can be interrupted by the emergency stop button.
There are several fuses placed on this board. The two automotive-fuses are used as main fuses. The smaller fuses are to protect the connected sensors and other peripheral parts. If a fuse is burned out, the appropriate LED is lit up. Thus an easy error indication is possible.

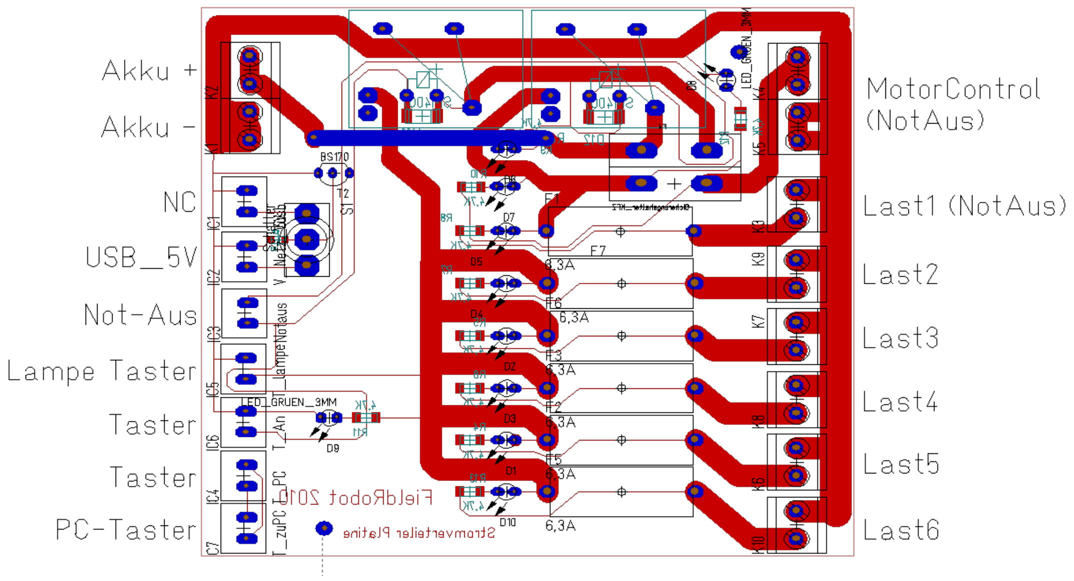


Figure 4.8. Power-switching and fuse-protection Board

Labjack Expansionbox

To switch high power loads with a LabJack box, a switching expansion is needed. The LabJack box itself only can switch low power loads such as LEDs. So a transistor is connected to an output of the box which again is used to turn on and off a relay. This relay than again can switch the high-power loads that are connected.
The LabJack is used to control the weed-killing device.

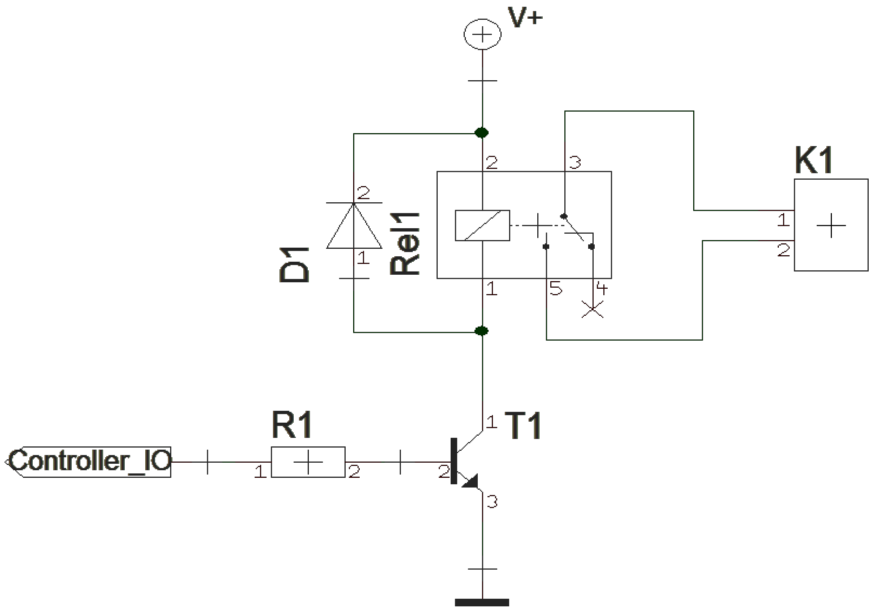


Figure 4.9. Electrical diagram of the Labjack Expansionbox

4.3. Weed killing device

The weed killing device consists of a sprayer, which together with the CMUcams performs the weed killing task. The CMU cameras detect the weed and send a signal to the robot which triggers the output ports of the I/O device- LabJack U3. These logic-level signals trigger a relays box which operates the pumps of the weed killing device.
For the visual signalization two flash lights are used.The flash lights are connected in parallel with the two pumps.

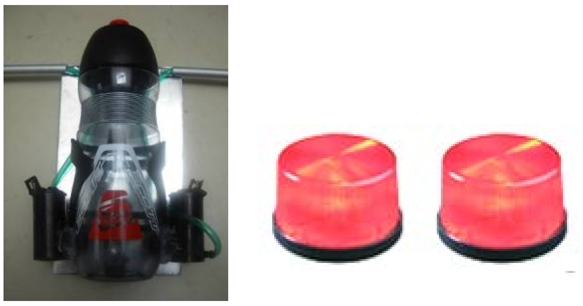


Figure 4.10. Sprayer and flashlights

5. . Software concept

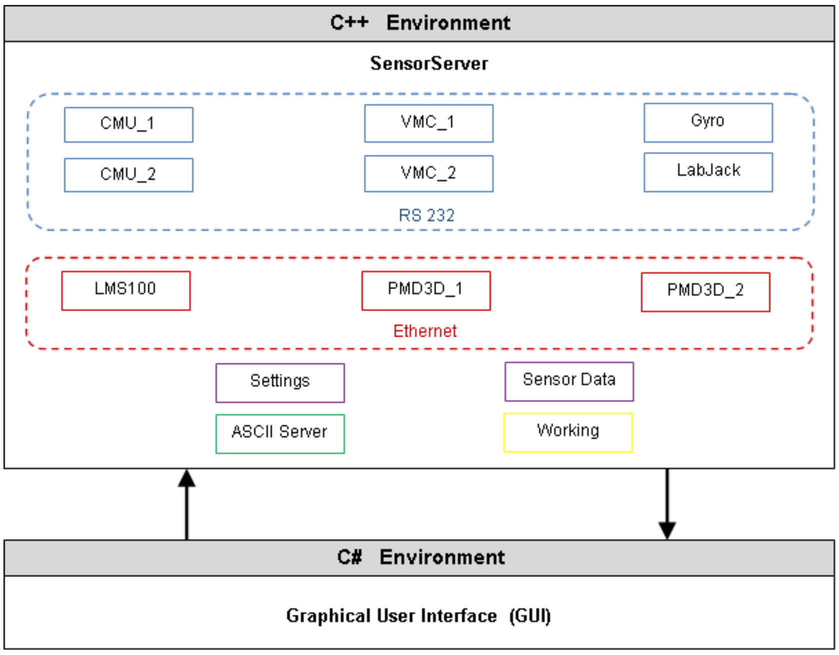


Figure 5.1. Software concept overview

5.1. Robot control software

The Robot control software bases on a C++ server which is executed on a Windows 7 platform. The GUI is written in C# and communicates with the server program on the robot.

The entire Server uses thread programming to collect data from the different sensors. Also the task behavior runs in an own thread and controls the operational sequences inside the particular tasks.
For communication with the laser scanner and the "PMD 3D Cameras" via Ethernet the server uses socket connections, which transfer data to the server.
The communication between the CMU Cams, the gyroscope and the server is realized with RS-232 USB converters, because the robot computer just has USB ports and no onboard RS232 interface.

Figure 5.2. Robots controller

The "Working Thread"

In the "Working Thread" all actions concerning the robot behaviour are performed. The delay cycle of the working thread is about ten milliseconds. The different tasks are called by the run function in the Working Thread. The GUI sends a variable which is used for the task selection. Every task has its own function where the action behavior is implemented. For example the first and second task are subdivided in five steps.

00 => ready to start
10 => cruise through maize row
20 => rotate 90° out of row
30 => select next row
40 => rotate 90° in row

The server waits until all sensors and controllers are initialized and the pattern is received from the GUI. After that the robot is ready for start. Now, the robot is in the first step. It drives through the first maize row till the end of maize row is reached. Then it changes to the second step and realizes the 90° rotation out of the row.
The selection of the next maize row is implemented in the third step. If the robot finds the row, where it wants to go, it selects the last step in this cycle. The robot turns to the same side it as in step two. After this procedure the cycle begins again at step one.

5.2. Laser scanner software

The navigation algorithm of the robot is using the laser scanner and is composed of many little algorithms who work together:

1. Conversion of the scanned data in cartesian format.
2. Restricting the scan area to 180°
3. Reducing the scan area [Length: 1400mm, Width: 1500mm]. **(Picture 1)**
4. Dividing the scan area in a square, x Direction (10mm), y Direction (10mm)
5. Dividing the scan area in the middle so that two equal surfaces result, the left one covers (0-90)° , and the right one (91-180)° **(Picture 2)**
6. Horizontal mirroring of the left scan area so that the algorithms for one scan area can be extrapolated for the second scan area. **(Picture 3)**

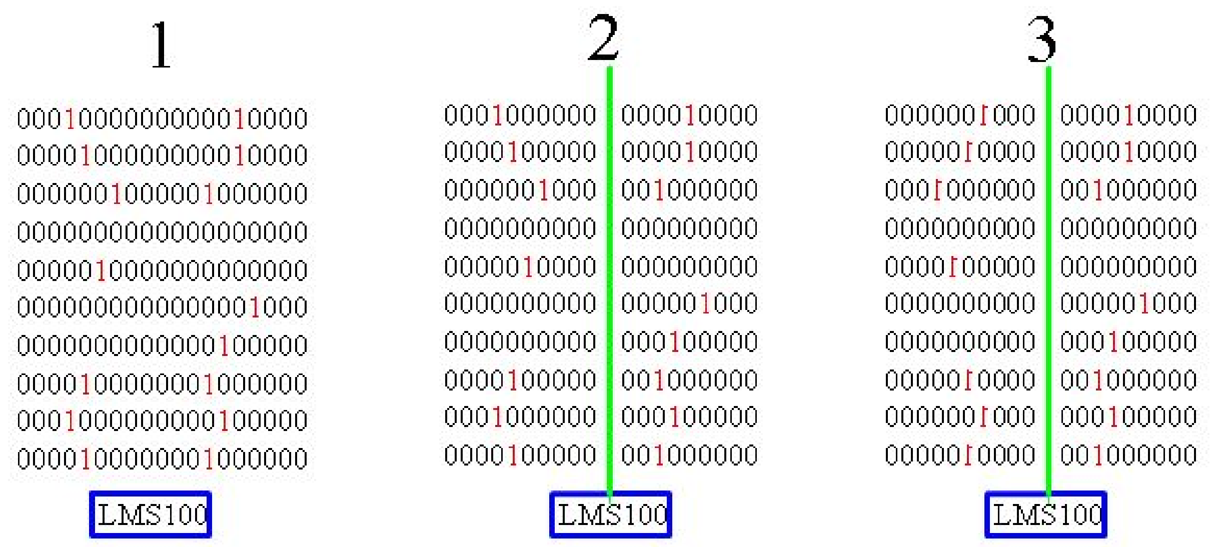


Figure 5.3. Binary representation of the field

7. Measurement of the distance from the right and the left side. The distance to the left side is represented with negative values.

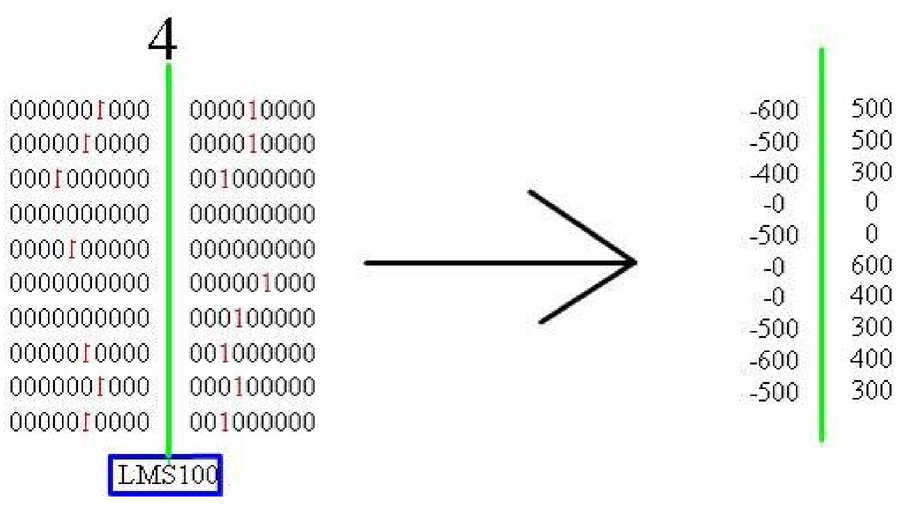


Figure 5.4. Algorithm for measurement of the left and right distance

8. Closing the horizontal gaps: when on the right side there is and 0 detected and on the left side an 1, and vice-versa, then is the value from the side where there was an 0 detected estimated with the minimal distance.

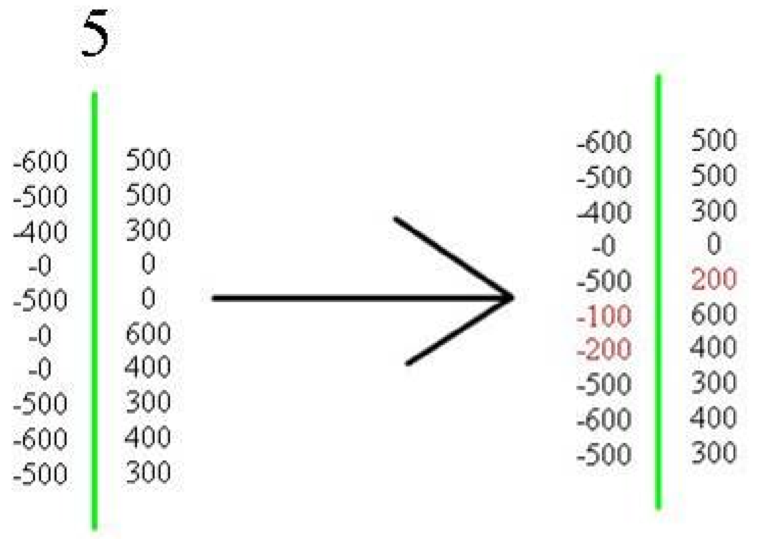


Figure 5.5. Binary representation of algorithm 8

9. Closing of the Vertical gaps: when on the both sides an 0 appears then the vertical value is calculated using the last value and the next value that not 0 is.
10. Calculating the distance middle value.
11. Defining the curve radius.
12. Collision recognizing.
13. Recognizing the end of the row.
14. Counting the rows.
15. Approximation of the speed.

5.3. CMU Cameras software

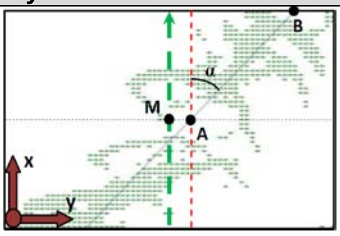
The software running on the CMU cameras uses color recognition to detect the desired objects. To accomplish object detection the colors spaces of the desired objects are stored. During test with different lightning conditions optimal color spaces for weed detection were determined. Further criteria to recognize the Objects are the recognized pixels and the density of these pixels. If an Object is discovered the camera sends a signal to the robot.

5.4. 3D Cameras algorithms

The 3D-camera returns a matrix which contains the distance information of each pixel. The maize row is identified by linear regression using the (x,y) coordinates that corresponds to distances in a range from Hmin to Hmax. The robot follows the row and tries to keep it in the middle of the image parallel to x axis. If there are no deviations (no offset and no angle), the robot will drive straight ahead. When deviations occur, the robot must turn right or left, aimed to correct the deviations that are calculated with the following algorithms:

$\mathit{calibration\ rapport} = \dfrac{H - 1.68}{-0.008};$	$H \quad [m] \quad - \mathit{distance\ between\ 3D - camera\ and\ soil}$
$d = y_M - y_A;$	$d \quad [pixel] \quad - \mathit{deviations\ offset\ in\ the\ image}$
$D = \dfrac{y_M - y_A}{\mathit{calibration\ rapport}};$	$D \quad [m] \quad - \mathit{real\ deviations\ offset}$
$\alpha = arctg\left(\dfrac{y_B - y_A}{x_B - x_A}\right);$	$a \quad [°] \quad - \mathit{deviation\ angle}$

left 3D-camera	
robot trajectory deviations	**trajectory corrections**

	if $\alpha > 0$ // turn right
	if $\alpha < 0$ // turn left
	if $d > 0$ // offset left
	if $d < 0$ // offset right

right 3D-camera	
robot trajectory deviations	**trajectory corrections**

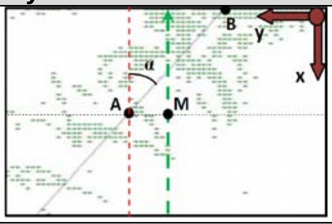
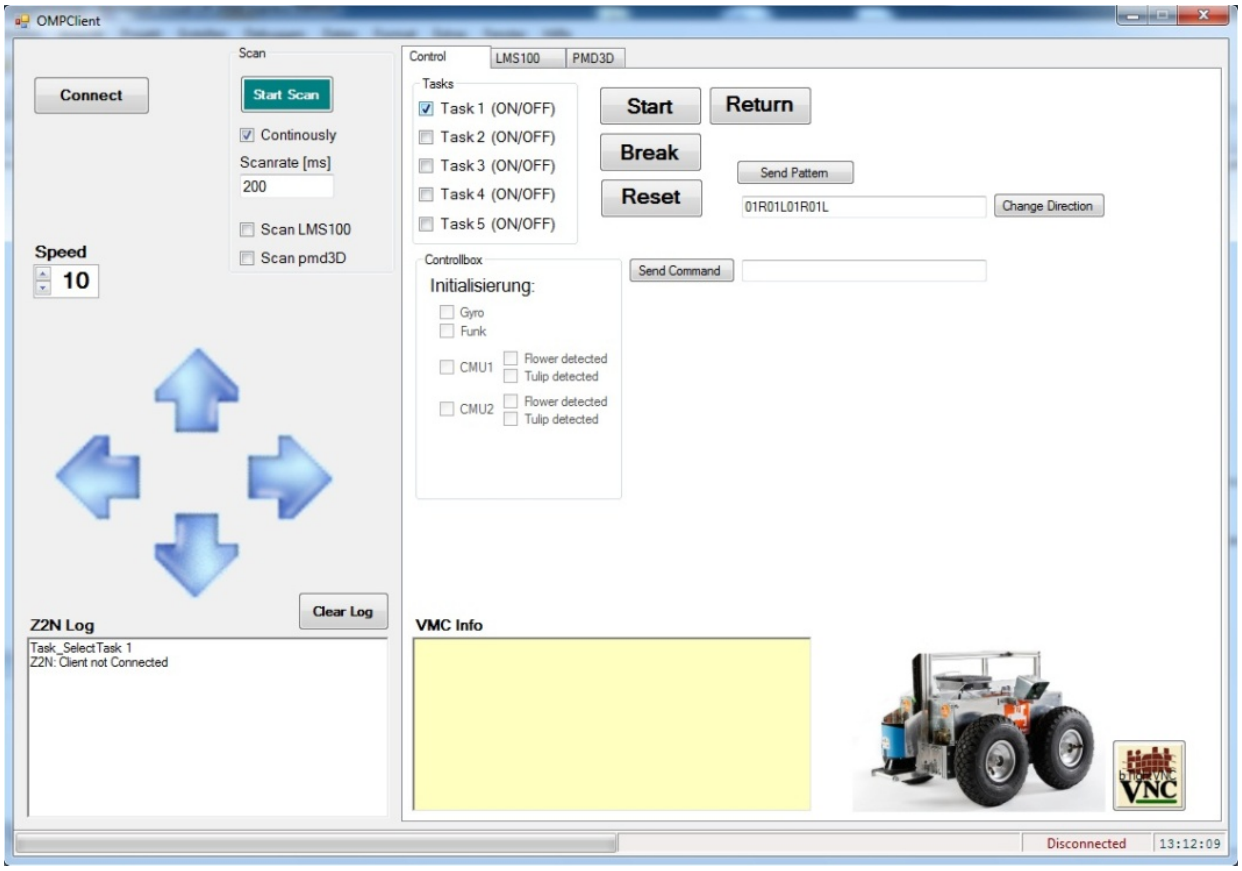
	if $\alpha > 0$ // turn right
	if $\alpha < 0$ // turn left
	if $d > 0$ // offset right
	if $d < 0$ // offset left

Figure 5.6. Image analysis

5.5. Graphical User Interface (GUI) for the OptimaizePrimeControl (OPM)

The Optimaize Prime Software consists of the server on the robot and the client on the control computer. The client creates a graphic user interface (GUI). The interface shows the software control on the right site. With the “Connect-Button” the client connects to the server via Ethernet. The IP-address and the port-number are stored in an ini-file (see fig-

ure 4.3.4). In the middle on the left are the buttons for manual control and speed adjustment. The lower text field is showing status messages.
With the tabs on the right side, “Control”, “LMS 100” and “PMD3D” parameters for the tasks and the sensors are set. In the “Task-Box” you can choose the different tasks. The “Start-Button” starts the particular task, and with the “Break-Button” the program can be paused. “Reset” stops the program and with “Return” you can re-choose the latest command. The “Send Pattern-Button” sends a drive pattern for the tasks. The pattern code is shown in the text-field below the button. The “Change Direction” button inverts the directions of the pattern.
The “Send Command-Button” sends the direct command from the text field beside the button. The “Control-Box” shows the initialization status of the sensors. Beside the “CMU-Cams” the signals show the detection of the weed. In the “VMC Info-Box” all important motor information are displayed.



Scanrate [ms]: 200

[✓ Continously]

[✓ Task 1 (ON/OFF)]

Speed: 10

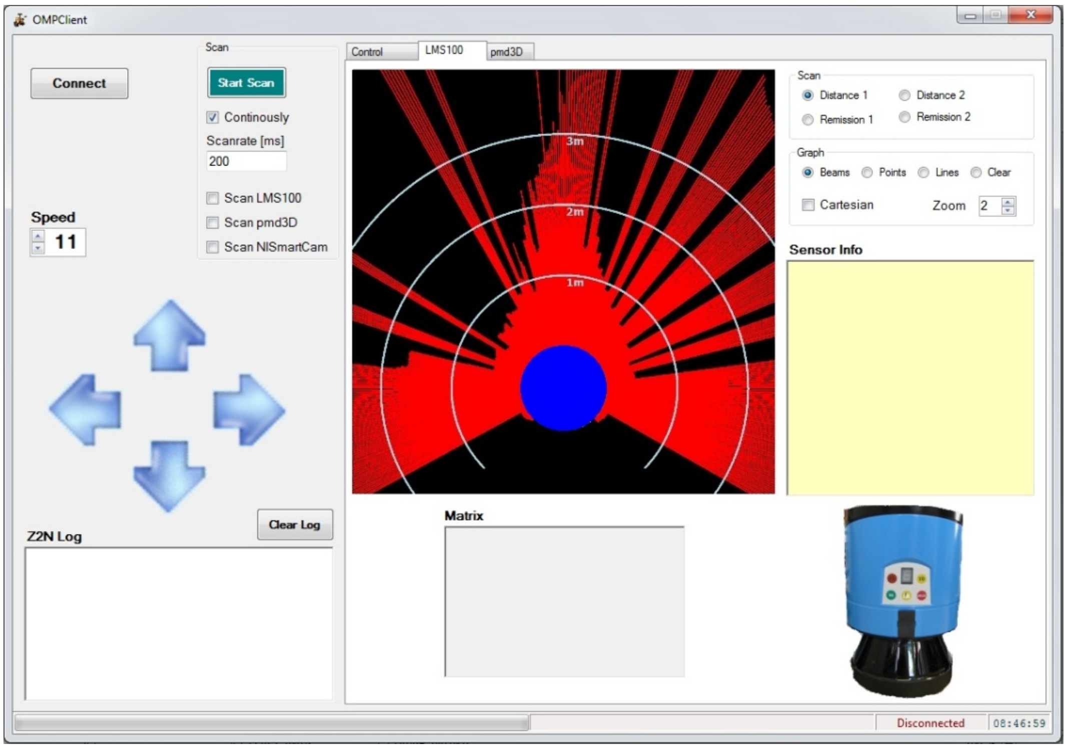
01R01L01R01L

Z2N Log:
```
Task_SelectTask 1
Z2N: Client not Connected
```

Disconnected 13:12:09

Figure 5.7. Overview of the GUI

The second tab is the control panel for the laser scanner LMS100. In the middle, you can see a live picture of the on-going measurement. In the “Scan-Box” you can change the algorithm for the scanner. The box “Graph” changes the display of the live picture. You can change between beams, points and lines values. In the “Sensor Info-Box” all important information are shown.



[✓] Continously
Scanrate [ms] 200
Speed 11
[ ] Scan LMS100
[ ] Scan pmd3D
[ ] Scan NISmartCam
Scan: [✓ Distance 1] Distance 2 Remission 1 Remission 2
Graph: [✓ Beams] Points Lines Clear
[ ] Cartesian Zoom 2

Figure 5.8. Laser scanner control panel

The last tab “PMD3D” is the interface for the 3D-cameras. The two live pictures from the 3D-cams are shown at the top of. The the two “Sensor Info-Boxes” show the information from the cameras.

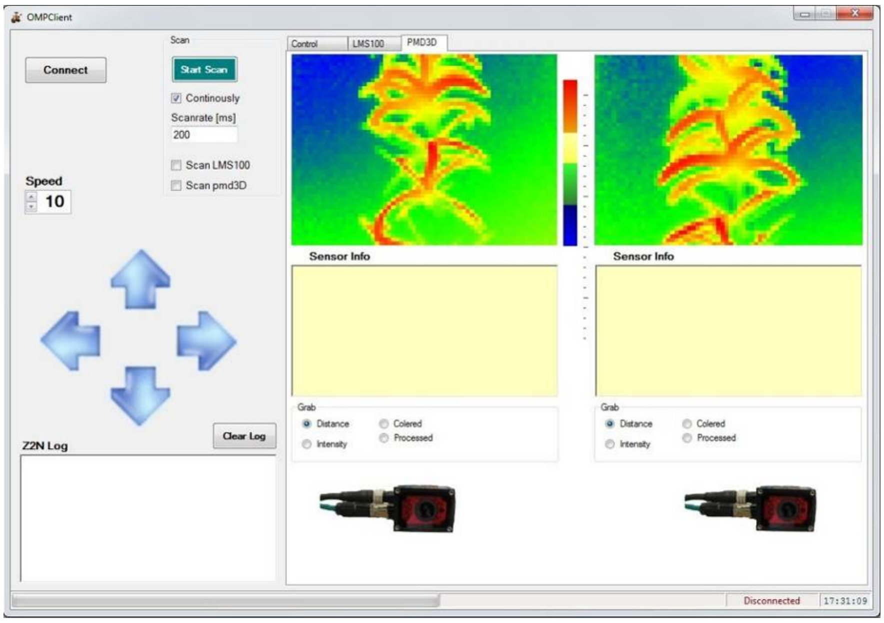
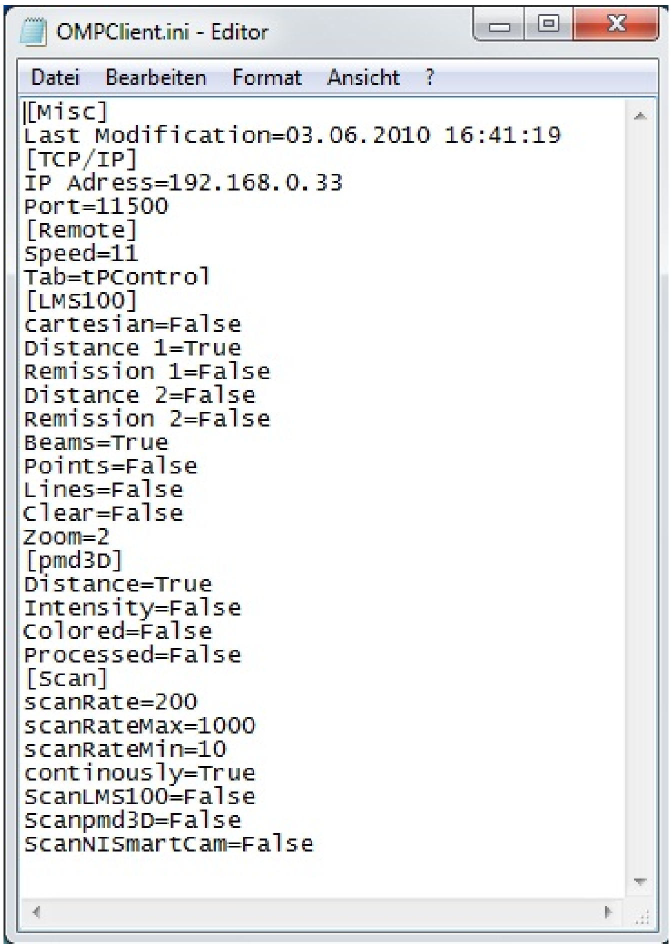


Figure 5.9. Sensors Info-Boxes

Most of the parameters can not be changed in the GUI, but in the OMPClient.ini.



```
[Misc]
Last Modification=03.06.2010 16:41:19
[TCP/IP]
IP Adress=192.168.0.33
Port=11500
[Remote]
Speed=11
Tab=tPControl
[LMS100]
cartesian=False
Distance 1=True
Remission 1=False
Distance 2=False
Remission 2=False
Beams=True
Points=False
Lines=False
Clear=False
Zoom=2
[pmd3D]
Distance=True
Intensity=False
Colored=False
Processed=False
[Scan]
scanRate=200
scanRateMax=1000
scanRateMin=10
continously=True
ScanLMS100=False
Scanpmd3D=False
ScanNISmartCam=False
```

Figure 5.10. OPMClient.ini

6. Conclusions

The new platform created for this competition seemed to be very robust and is nearly ideal for the prevailing conditions. But it was very difficult to put all components in the chassis because of marginal storage space. Other difficulties were caused by the applied algorithms. Especially driving the predefined pattern was problematical because of the very little maize plants and the too thin green sticks which were added to the maize plants. In weed-detection maybe another strategy should be persecuted than colour tracking because this approach is very addicted to light conditions. Altogether there are a few things that should be enhanced to achieve better results in next competitions; especially algorithms have to be improved.

7. Acknowledgments

The Field Robot Team of the University of Applied Sciences Osnabrück would like to thank its generous sponsors for their support.

AMAZONE-Werke H.Dreyer, Hasbergen-Gaste, Germany
INOEX GmbH, Bad Oeynhausen, Germany
IFM Electronic GmbH, Essen, Germany

8.

Moreover the Team would like to thank Arno Ruckelshausen for making the participation at the field robot event possible. Special thanks go to Andreas Linz and Ralph Klose for their advice and assistance in realization of this project.

9. References

[1] http://www.analog.com/static/imported-files/data_sheets/adxrs300.pdf, Page 4, state June 2010

[2] http://www.hizook.com/blog/2009/03/03/new-sick-laser-rangefinder-lms-100-designed-compete-hokuyo-utm-30lx, state June 2010

[3] http://www.cmucam.org, state June 2010

[4] http://www.ifm.com, state June 2010

Rusticus

Team Members: Alexander Engeln, Heinz Dornseifer, Frank Engeln
Bursibantstrasse 3, 48429 Rheine, Germany

Abstract

Rusticus is an autonomous robot, which is designed and developed by a private team of 3 generations. The robot was built from a Tetrix® kit and is to show the Lego Mindstorms is not just a toy, but can also be used for high-quality work.

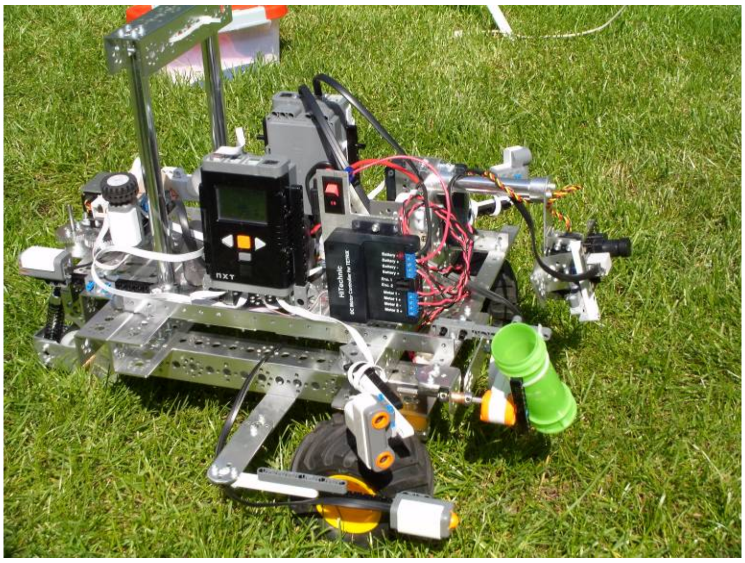
1. Introduction

The Field Robot Event is a good opportunity for the strengths or weaknesses of Lego Mindstorms and Tetrix® test. We want to show that robotic is not only interesting for young people. I am a computer science teacher for lower classes, my son is student of the 8th class and my father is electronic in retirement.

2. Datasheet

Chassis	W x L x H (without whiskers)	*290x390x 400*
	W x L x H (with whiskers)	*500x600x 400*
	Ground clearance	*85 mm*
	Weight	*kg*
	Drive concept 2 front wheels	
	Steering system 1 wheel steering system	
	Special characteristics	
Energy	1x 12 V 3 Ah	
	2x 7,4 V 1,5 Ah	
Motors	2x DC RB35	
	1x Servo, 1 NXT Motor	

Controller	2x Mindstorms NXT, 1x Hitechnic Servo Controller, 1x Hitechnic DC Controller
Software	Robot C, NXC
Sensors	2x Ultrasonic, 2x Touch Sensors, 1x Soundsensor, 1x Encoder
	A complete Set of Tetrix Robotic

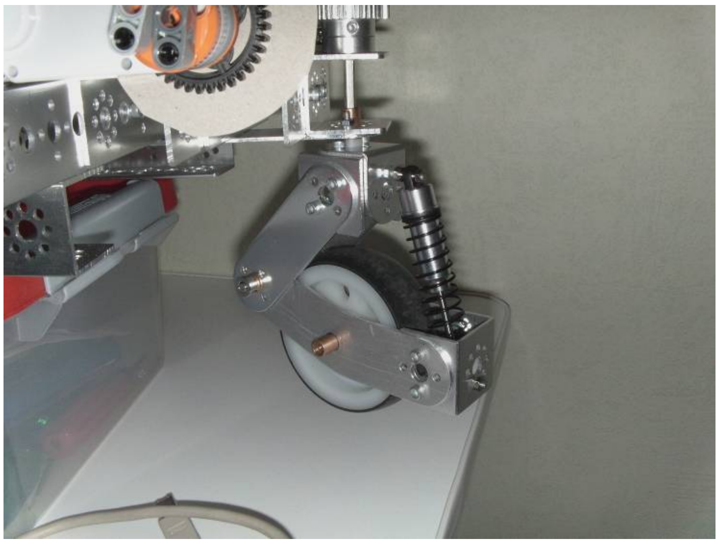


Hardware

Chassis

The chassis is a build from a complete design system called Tetrix®. Tetrix® provides the ideal platform for flexible and creative robot design and building. Whether students want to build a basic square chassis robot base with a remote control or incorporate other electronics (not included) to design a highly specific autonomous robot, this system is unlimited.

The robot has got three wheels: Two driven wheels at the front sides and a non-driven pivoted wheel.



Self-built wheel with „shock absorber“.

Current supply

For supply of actuating elements, a 12 V Ni-MH accumulator with a capacity of 3 Ah is used.
For both NXT and Sensors, two 7,4 V accumulator with a capacity of 1,5 Ah are used.

Microcontroller

2x Lego Mindstorms NXT

• 32-bit ARM7 microprocessor

• 256kByte FLASH and RAM memory 64kByte

• Unlimited number of presets

• Bluetooth technology enables wireless communication between the NXT´s, computers, PDAs and mobile phones

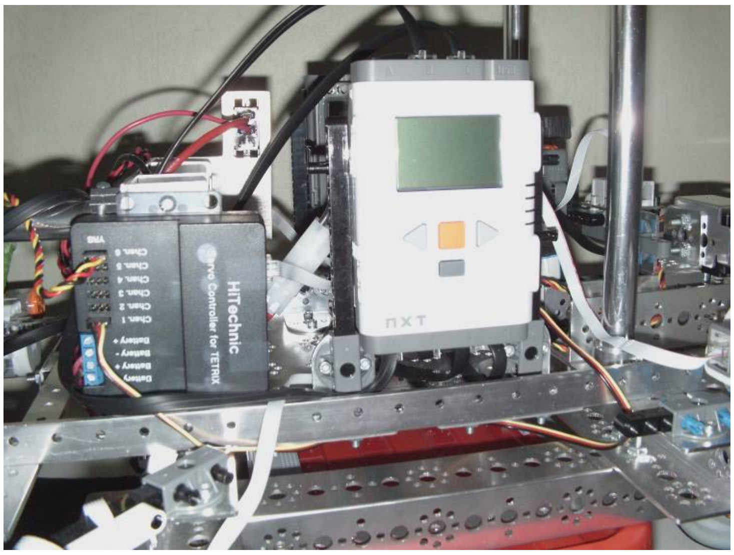
• USB 2.0 port

• 4 inputs for sensors

• 3 outputs for motors and lights

• Programmable, graphic display (60 x 100 pixels)

• Speakers



Motordrivers

The HiTechnic DC Motor Controller for TETRIX™ connects to an NXT sensor port and will enable you to control powerful DC gear motors for use with TETRIX robots. The controller has two H-bridge outputs to control the speed and direction of two DC gear motors and is designed to connect to the TETRIX whole pattern. The HiTechnic Servo Motor Controller for TETRIX™ will enable you to control up to six R/C-type servo motors for use with TETRIX robots, enabling you to add proportional positioning to your robotic creations.

Sensors

We decided to use two ultrasonic sensors. The ultrasonic sensor can measure the distance from the sensor to something that it is facing, and detect movement. It shows us the distance in cm. The maximum distance it can measure is 233 cm with a precision of 3 cm. The ultrasonic sensor works by sending out ultrasonic sound waves that bounce off an object ahead of it and then back. It senses the time it took for that to happen.
The touch sensor in front of the robot detects whether it is currently pressed, has been bumped, or released.
The Encoder enables our robot to move a fixed distance, rotate to a specific position or move at a constant speed.

NXTCam

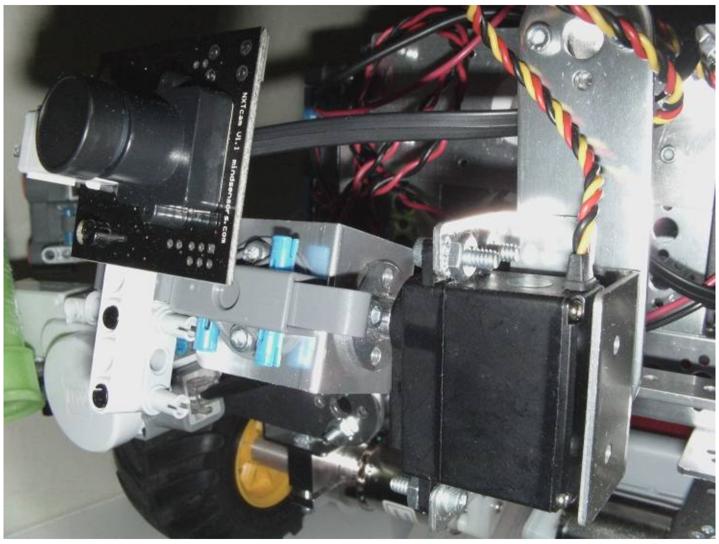
The NXTCam is a real-time image processing engine. Think of it as a vision sub-system with on-board processor and a protocol interface that is accessible through a standard NXT sensor port. This interface provides high-level, post-processed information of the image NXTCam sees. The processed information contains the bounding box coordinates of the objects of interest in view of NXTCam, in line tracking mode, this information contains coordinates of line segments.

• Tracked image resolution of 88 x 144 pixels at 30 frames/second

• Perform full-resolution (176 x 144) pixels color image dumps to PC via USB port.

• Maximum power consumption (42 mA at 4.7 V)

• Uses NXT compatible I2C protocol for communications.



NXTCam in front of the robot

Software

Some parts of the program code (for driving) are written in Robot C other in NXC (not exactly C).

Program Concept

The general idea is to use the encoder for the way through the plants. The US sensors are used to get information about if the robot drives in the middle of the plants. The encoder counts the meters and the length of the distance covered.

Conclusion

We need three more multiplexers to connect more ultrasonic sensors. The concept is o.k. but we need more time to word. The Tetrix system is new in Germany (March 2010). So, it is our first experiment with this system.