FieldRobotEvent

PrototypeContest.Workshops.Party

June 12-14, 2008
Osnabrück, Germany
www.fieldrobotevent.de

Fachhochschule Osnabrück
University of Applied Sciences

Germany
Land of Ideas
Selected Landmark 2008

Proceedings

Proceedings of the 6th Field Robot Event 2008

Osnabrück/Germany June 12 -14, 2008

ISBN: 978-3-00-027341-4

Date : March 2009

Publisher : University of Applied Sciences Osnabrück

Contact : Albrechtstr. 30
49076 Osnabrück
Germany
Phone: +49 541-969-2090/2057
Arno Ruckelshausen, Christian Newton
fieldrobotevent@fhos.de
www.fieldrobotevent.de

Sponsors Field Robot Event 2008

AMAZONE

Germany
Land of Ideas
Selected Landmark 2008

WIGOS
Wirtschaftsförderungsgesellschaft
Osnabrücker Land mbH

LANDKREIS
OSNABRÜCK

Bundesministerium für
Ernährung, Landwirtschaft
und Verbraucherschutz

Bundesanstalt für
Landwirtschaft und Ernährung

CLAAS STIFTUNG

KRONE

Niedersächsisches Ministerium
für Wissenschaft und Kultur

Deutsche Bank

WALLENHORST
die Gemeinde

BOSCH

bertrandt

SIEMENS

MÜLLER
Elektronik
... wir regeln das!

Stadtwerke Osnabrück
Immer für Sie da.


WAGENINGEN UNIVERSITY
WAGENINGEN UR


INSP
IRE&
INNO
VATE


VME-STIFTUNG
Osnabrück - Emsland


Fachhochschule Osnabrück
University of Applied Sciences


WABE-Zentrum
Klaus-Bahlsen-Haus


Versuchsbetrieb
Waldhof


COALA®
nature · technology · food


UNIVERSITÄT OSNABRÜCK
Fachhochschule Osnabrück
University of Applied Sciences
Technologie-Kontaktstelle


enterprise
europe
network
Wir stehen Unternehmen zur Seite

Preface

Agricultural engineering has become a high-tech field with highest worldwide relevance with respect to food, energy as well as landscape conservation. There is a strong need for innovations and new ideas to create solutions with ecological, economical and social relevance. The future applications of autonomous small size robots and robot swarms in agriculture will be a revolution in this field. However, whenever technology meets nature, considerable technical and non-technical challenges have to be solved. The International Field Robot Event was founded by the Wageningen University (The Netherlands) in 2003 to inspire upcoming student generations. The event has developed into an international platform for students, experts and the public as well to exchange interdisciplinary knowledge and experiences on field robots.

Completely autonomous navigation in maize fields and application examples – such as weed control – are major tasks. Moreover, the freestyle and a challenge task with extreme field conditions demonstrate the agricultural relevance of the competition.

This event was accompanied by an exhibition of technology from companies and institutions and the “Field Robot Junior” competition with more than 30 school teams. Moreover, it has been awarded as a selected landmark 2008 within the initiative “Germany – Land of Ideas”.

In a workshop and fair the teams have presented their robots. The teams had also the option to write a paper describing the concept, the hard- and software of their robot. These papers are collected in the “Proceedings of the 6th Field Robot Event”, for most of the students it is the first scientific publication in their career.

Osnabrück/Germany, March 2009

Arno Ruckelshausen
Chairman 6th Field Robot Event

Index

4M - Mean Maize Maze Machine

Juha Backman[1], Heikki Hyyti[1], Jouko Kalmari[1], Jouko Kinnari[1], Antti Hakala[2], Vesa Poutiainen[2], Petro Tamminen[3], Heikki Väätäinen[3]
Timo Oksanen[1] (advisor), Jari Kostamo[2] (advisor), Johannes Tiusanen[3] (advisor)

[1]Helsinki University of Technology, Department of Automation and Systems Technology
[2]Helsinki University of Technology, Department of Mechanical Engineering
[3]University of Helsinki, Department of Agrotechnology

http://automation.tkk.fi/FieldRobot2008

Abstract

4M – Mean Maize Maze Machine was a joint project work of students from three departments from two universities in Helsinki, Finland. Planning of the robot was started in September 2007. The actual building of the robot's final version of mechanics started in January 2008 and the robot was mechanically and electrically finished in May 2008. The software of the robot was developed with help of a simulator at the same time when mechanics and electronics were designed and constructed. Rest of the time were testing and repairing.

The robot was designed to be fully autonomous and it could be supplied with implements. The robot also had a modular design so that every mechanical and electrical part was easy to repair. The same modular idea was also used in software design. This design helped to keep things in order and made clear boundaries between responsibility areas.

The budget of the robot was low. Parts cost below 2000€ and no special industry parts were used. Most of the mechanical parts were handmade and some parts were taken from RC cars. Most of the electric parts were standard consumer electrics

including the webcam and laptop PC. The software was completely self-made. OpenCV-library was used for basic operations of the machine vision. Data processing algorithms were developed in Matlab Simulink and the C code was automatically generated from that.

This was the fourth time when the joint student team from Helsinki University of Technology and University of Helsinki participated to the Field Robot Event. In the past years the concept has been similar, but this time everything worked and the result was winning of the competition.

Keywords: robot, agriculture, machine vision, sensor fusion, estimation, navigation, suspension systems, patch seeding

Introduction

4M was a project work of six students from Helsinki University of Technology (TKK) and two students from University of Helsinki. The goal was to make a modular, reliable, robust and low cost robot to the Field Robot Event 2008 competition. The goals were achieved in most of the parts and results were satisfying. The final version of the robot is in figure 1 in its natural environment – maize field.

Figure 1 Robot navigating towards maize rows

Students from the Department of Automation and Systems Technology were responsible of making the robot's software while students from the Department of Mechanical Engineering made robot's mechanics. The main responsibility of freestyle task and manufacturing the Seed Drill was held by students from University of Helsinki. Jouko Kinnari started as a captain of the team in January but after it was realized that Jouko cannot participate to the competition (due to his working duties), Juha Backman acted as a captain for the rest of the time.

The student members of the team had no prior expertise in making robots and the whole process was learning new things and techniques. This paper describes the most essential things that the team learned and discovered during this very educational year.

Hardware

Robot chassis

The robot design was started from the main ideas of a simple, strong and easy to maintain chassis. Conclusion was to build a modular structure for the robot. 4M consisted of four modular units: two modules for steering and axles, a module for motor and advanced support force divisor mechanism, a module for the main frame (bended aluminum plate) of the chassis and the top level module for electronics, sensors and machine vision.

A normal ”rock crawler” twisting torso design was implemented to the robot with a novel shock absorbing method to balance supporting forces and therefore maximize the traction on the field. In the beginning it was demanded the robot’s wheels should be capable of ascending 10cm obstacle but in the end the performance was better. The heaviest parts, such as the motor unit and lead batteries, were located as low as possible to obtain a low center of gravity.

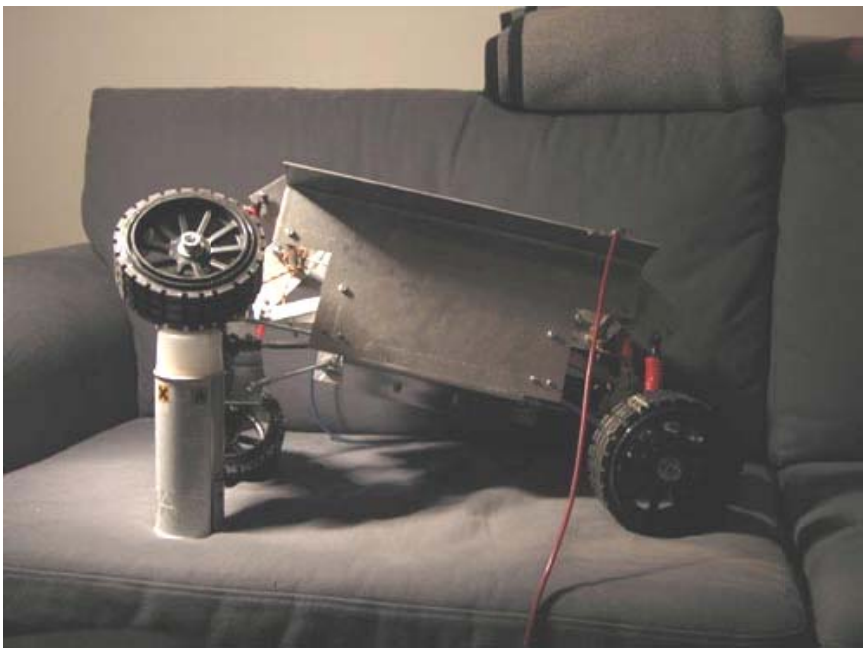


Figure 2 Torsional twist system allows robot to climb over large obstacles. It balances tilting of the chassis to keep machine vision sight on the field.

Development process

The design of the mechanical part of the robot was started by manufacturing a harsh plywood model of the mechanical structure, which was used to prove the functionality of the suspension system. The mechanism was also tested with MSC Adams simulation software [1] and A simple dynamical analysis of 4M was carried out. The

spring constants of the shock absorbers of 4M were analyzed to reduce the movements of the camera. After some iteration a relatively good value for the spring constants was found. However, at the field tests with 4M it turned out that the value found by simulating had nothing to do with reality because of the lack of parameters and physical properties of the modelled parts. It was concluded Adams was too complex software to analyze the functionality of the chassis mechanism. When dynamical analysis of a system is needed, Adams can be used, but a reliable analysis requires a lot of data concerning the mechanical parts and the parameters affecting the functionality of the mechanism. For next year's participants it is highly recommended to concentrate only on 3D –modelling. The next phase was to build 3D CAD model of the final robot with Pro/ENGINEER [7]. The parts were then manufactured based on the CAD model.

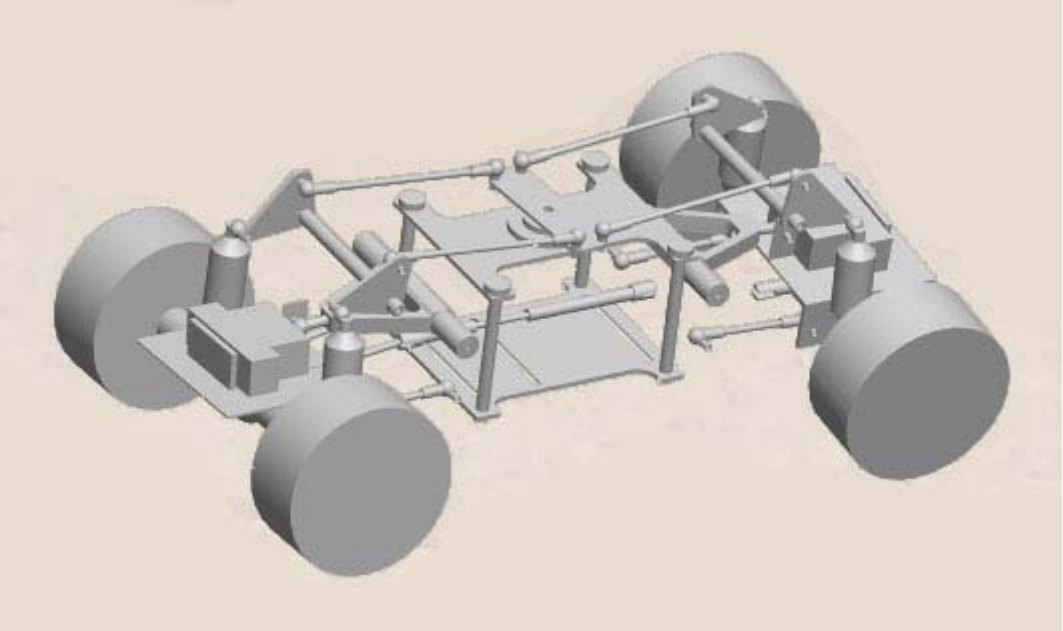


Figure 3 3D CAD model of the robot's mechanisms

The plywood version was the first prototype, the second prototype was more like the final result, but the accuracy of the parts' manufacturing was not satisfying. So, a third prototype of the chassis structure was built, now machined more accurately. After the proper aluminum frame was ready for the robot the workload was distributed between the mechatronics team members so that the other would make the axle modules and the other one would start manufacturing the power transmission and the mechanism which balanced the support forces of the wheels. When these stages were completed the mechanics was ready for simple testing.

First tests showed a lot more capability for the shock absorbers was needed to carry the full load of the robot. Based on the test result the chassis of the robot was improved and some parts of the robot were repaired. Strengthening the weak points and adding features as foamed rubber to make the system water proof increased the reliability of the system.

Previous robots from TKK had problems with too much friction in steering due to wide tires. The tires used previously were 100mm wide, low diameter and filled with very soft foamed plastic or air. This time two different types of tires for the robot vehicle were manufactured while keeping the previous flaws in mind. Now the diameter of the wheels was 140mm (hard ground) and 165 mm (soft ground) and they were only 50mm wide which increased the surface pressure and therefore enhanced the traction. The material of the tires was a lot stiffer than before so it wouldn't take too much power from the servos to turn the wheels. Original springs and the oil of the shock absorbers were replaced by higher viscosity shock oil and heavy springs to carry out the 19kg weight.

Most of the mechanical parts of the robot were built at Helsinki University of Technology and the only commercial parts of the chassis are the front and rear axis which were taken from a RC car. Main tools used for the manufacturing of the robot were hydraulic plate cutter, NC mill, lathe, circular- and band saw.

Although some alternative solutions were considered for the gear system, such as belt drive, a solution using strong pinion gears was finally used. Luckily the gears were specially made and heavy duty pinions, hardened steel for 1:5 class RC buggies because during the testing some minor was detected in the motor support structure which allowed the high torque to move the motor and this made noise from the gears while driving reverse. Although it obviously harmed the pinions a little, there wasn't any real problem with performance and durability.

The robot was totally ready two weeks before the competition. Even though the robot was ready for testing already in March the test showed there was still more work to do before it would be ready for the competition. Final tests were conducted in Osnabrück test field because the whole system required calibration in the right circumstances.

Electronics and sensors

The electronics and the rest of the robot were constructed from cheap materials. Standard and low cost connectors, cables, controllers and sensor chips were only used. Most of the electronics was build from scratch and only microcontroller boards and some sensors were bought as manufactured. Electronic assembly was made using central input output terminal strip to ensure that all the wires were properly connected and all the signals were always available for measurements as you can see from the figure 4.

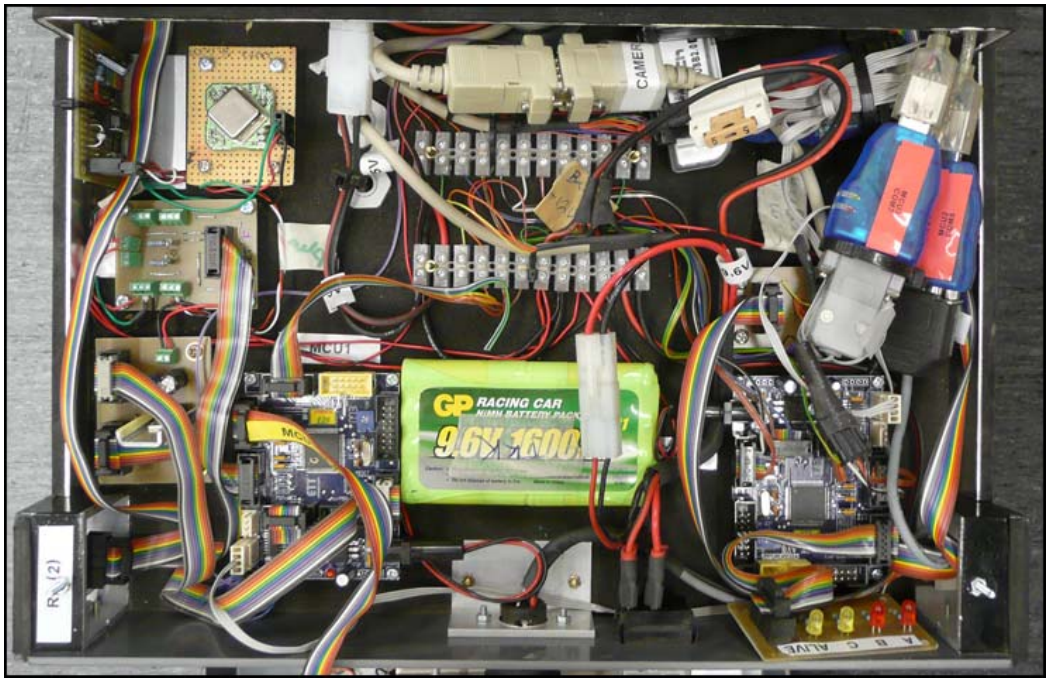


Figure 4 Electrical connections and electronics in the robot

Sensors

The maize sensing system of the robot consisted of a cheap USB web camera (Logitech QuickCam Pro 5000) for machine vision and four ultrasonic rangers (SRF08) on every corner of the robot for detecting maize plants in a row. There was a motor speed encoder in the robot (Sharp GP1A30RJ000F photo interrupter & wheel from old Logitech mouse) for getting the travel speed and the traveled distance and 10kOhm linear potentiometers for positioning rear and front wheel angles. Commonly used compass CMPS03 chip was used to determine the angle of the robot on headland, but it was found to be too inaccurate. Compass error was related to its tilting angle and because our compass was on the camera pole, it was constantly moving and turning. A cheap gyroscope (Murata ENC03A) was used to improve the estimation of the heading angle. The problem was also tried to be fixed

by using two VTI SCA610 accelerometers as inclinometers compensating the magnetometer error in the compass but it was too late and time run out.

Actuators

The actuators of the robot consisted of two solenoids for pressing spray cans, two Hitec HS-805BB+ servos to steer the front and the rear wheels and a Futaba S3003 servo for turning the camera. Latter was used with a self made transmission to get the camera turn 360 degrees with a cheap servo which turns only 180 degrees. The drive motor and the planetary gear was taken from a Bosch PSR 12 cordless drill. AEI security alarm was used as a horn to indicate the detected golf balls.

Power electronics

The drive motor was controlled with a PWM circuit that runs on a frequency of 16 kHz. PWM module has an H-Bridge connection to allow the driving motor to run in both directions. Input voltage of all three servo motors was 5 volts which was generated with three fixed 5V regulators (LM78T05). Solenoids and horn were controlled by N-type MOSFET transistors (IRF1310N) functioning as a relay system. The energy source of the robot was divided in to two different battery systems. Controllers and small power electronics had their own 9,6V battery and power electronics had a 12V battery system that consists of two 7Ah 6V lead-acid gel batteries.

Controllers

There were two Futurlec ATMega Controller boards in the robot both of which included an Atmel ATMega128 Microcontroller chip [2]. One of those was used to control the motor and wheel servos in 100Hz control loop. It took the measurements of wheel angle potentiometers and the signals of the motor encoder and controlled engine speed using PI controller. Servos which steered wheels were not accurate and powerful enough, so the controller was build to control wheel angles with cascade PI controller using the measured wheel angles. Measurement was done with the potentiometers. Controller was used to ensure that the set angles of the wheels and the speed of the robot were accurate. The other controller board serviced as input-output system for ultra sound sensors, compass, gyroscopes and accelerometers. It was also used to control the position of the servo turning the camera. Both of the ATMega controllers, maize counter and trailer were connected to PC laptop using USB RS232 adapters.

Software

The modular design was also applied in software design. First, the machine vision module was made because it could be tested with the help of previous year's videos from the maize field in Wageningen 2007. Later this machine vision module was tested with simulated maize field together with the Simulink part. All the modules are depicted in figure 5. Software tools used were Microsoft Visual Studio (C++ with machine vision and C# with main program) and Matlab Simulink [9][11]. Simulink Stateflow was used to program upper lever logic such as headland turnings and task related stuff. Matlab's Real-Time Workshop toolbox was used to generate C++ -code from Simulink models [8].

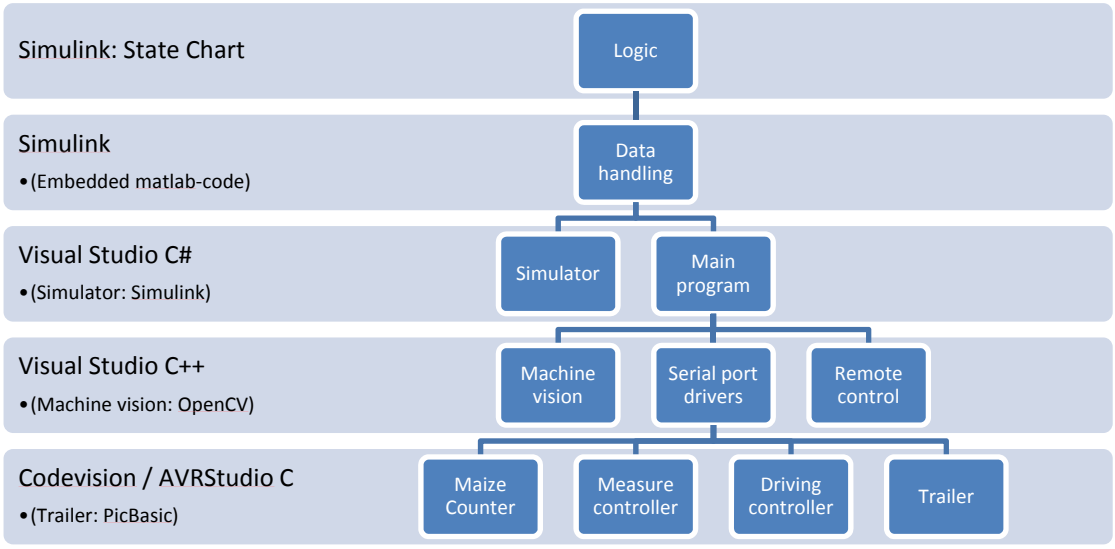


Figure 5 Software modules

Machine vision

Machine vision system was divided into blocks that all did their specific tasks. Blocks doing the machine vision processing shared the data in so called storage classes that usually just hold the processed images. This way all the blocks could be individually tested with test programs and it was easier to construct the whole machine vision simply by connecting the right classes to each other. Machine vision was programmed with C++ using OpenCV library when it was convenient [6].

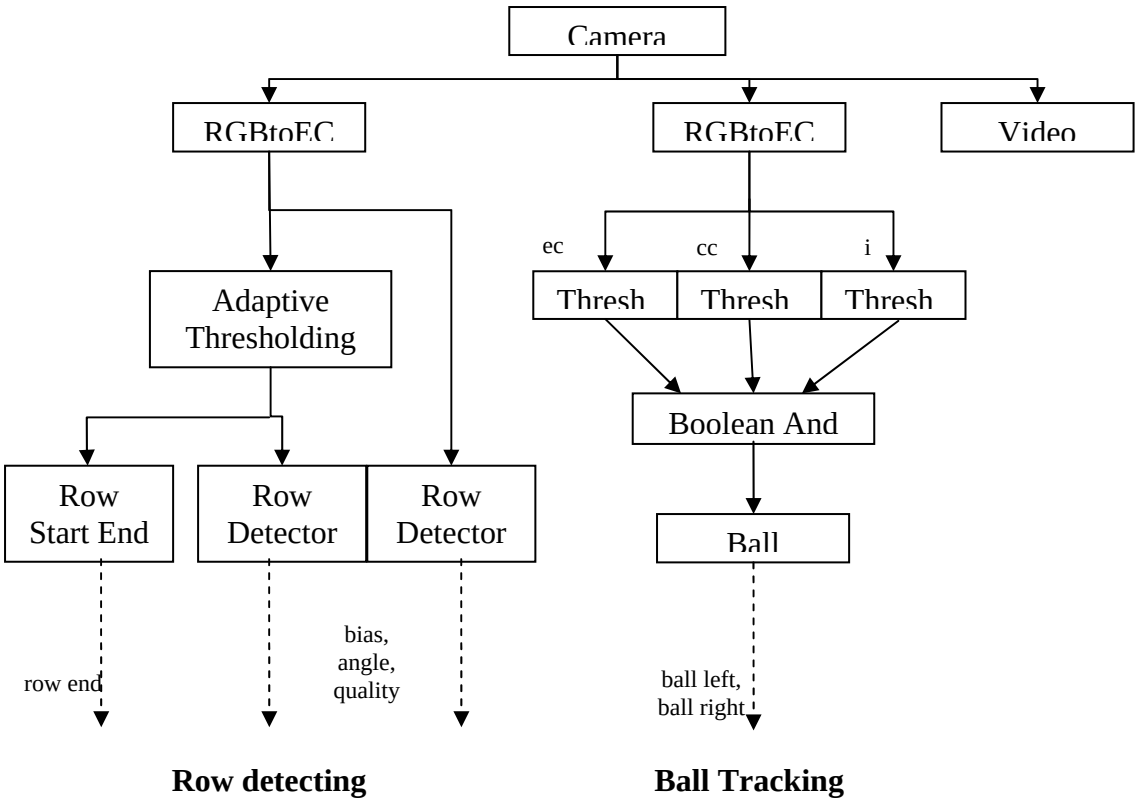


Figure 6 Simplified structure of the machine vision system

Color transformations: RGBtoECCI

A key issue in the robot's computer vision system is finding certain colors in the video since the recognition of dandelions and maize leafs is based on the recognizing their color. To be able to interpret what the robot actually saw with its camera, a measure for how much the color of each pixel in the video differed from a reference color was needed. The reference color could be arbitrarily picked.

The video information from the camera of the robot was RGB data which was transformed into ECCI color space. Transformation from RGB to ECCI was described in the proceedings of the last year [10]. ECCI transformation is a generalized version of the so-called EGRBI transform.

ECCI gives three components for each pixel:
a measure for how much of the reference color the pixel includes compared to the other colors, called the EC channel, this component is intensity-independent
a measure for how much the color of the pixel differs from the reference color, called the CC channel. For example in case of green target color this describes if the color

is more red or more blue (this is the case in EGRBI). For more information on this, see [10]
a measure for the intensity of a pixel, called the I channel

From the transformed color space, interpretations on whether the color of the transformed pixel is close to the target color can be done by e.g. thresholding the three component pictures and combining the threshold values with an AND operation.

The result from this color transformation is the ability to filter out very light pixels (high I value), very dark pixels (low I value) and those pixels that don't contain enough of the target color (low EC value). Also those pixels that seem to be too much off from the target color (as interpreted with the CC channel) can be filtered out.

Other color transformations, e.g. HSV and L*a*b, were tested but ECCI was found to be the most suitable in a comparison in which a video data from previous year maize field was used.

Row Detector A

Row Detector A was the main method used to determine the position of the robot on the row (bias) and the angle relative to row that the robot was on. The idea for the method was based on ASAE Publication *Automatic Guidance System With Crop Row Sensor* by H. Okamoto et al [5]. A similar algorithm was tried also in the last year's robot [10].

The input for the algorithm was a threshold image showing the maize white in color and everything else in black. First a perspective transformation was applied to the image. After this image looked like the picture was taken directly from above and all maize rows were parallel. Then the image was cut into six slices and a histogram was calculated from each slice (see figure 7). Histograms should have had a periodical signal. These histograms were shifted and combined several times to correspond with some possible angles of the robot. The angle having the most distinct combined histogram was then selected. The location of the peak in the combined histogram showed the bias, in other words the position of the robot on the row.

This algorithm worked quite well even in more difficult cases. The algorithm was able to handle the situations with missing plants or something green on the path. It worked even if the other maize row was completely missing.

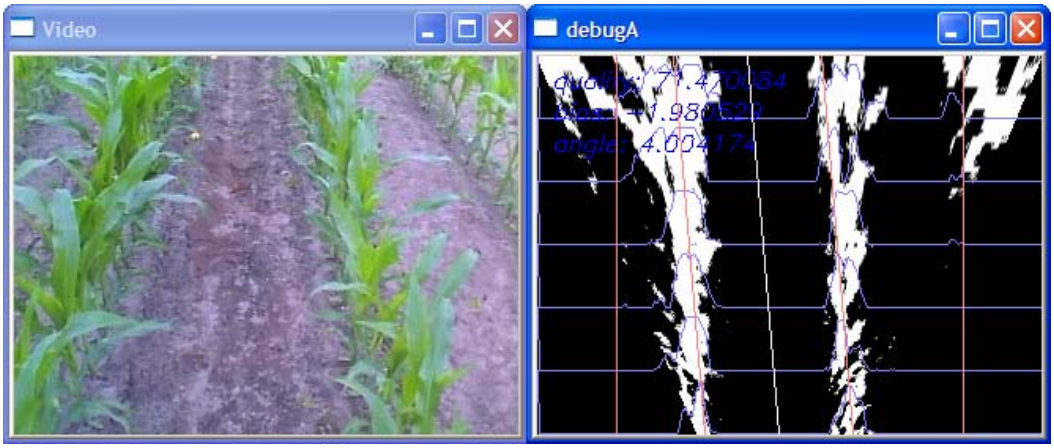


Figure 7 Original camera image and row detector A debug image. White line is the estimated center of the rows.

Row Detector B

There also was another algorithm to detect maize rows from the image. The main idea of this algorithm was to detect the ground between the rows. So the maize rows are detected indirectly. An advantage of this algorithm was that it didn't need the threshold image as the Row Detector A did.

First, the image is vertically divided into parts. Then the ground is separately detected in every part by calculating the correlation between the EC channel and the supposed ground value in the area which correspond to the size of the row. Basically the ground is the darkest area of the EC image. There are also some heuristics helping the algorithm to follow the right ground area and preventing big jumps between the overlapping parts. After the ground is detected in every part the center line is calculated by least squares method. It is also possible that the ground is not detected in every frame. In that case the quality factor of the detection is decreased and that part is leaved out of the centre line calculation. Also the fitness of the center line for the parts contributes to quality value.

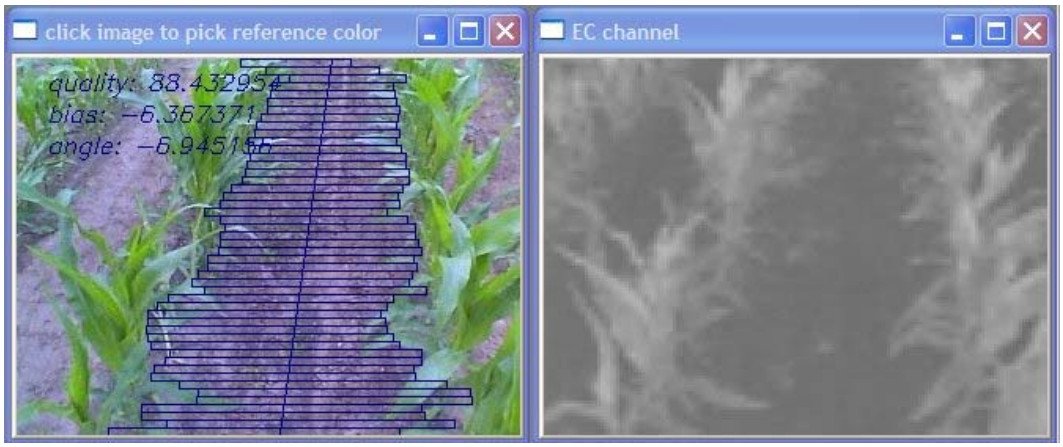


Figure 8 Row Detector B debug output is printed over original image. Right is shown the EC channel from which the ground is detected.

The algorithm was quite promising in the preliminary tests when the old video clips from maize field were used. But for some reason the algorithm did not work as supposed in the field tests it was decided to leave it out and only run the Row Detector A. Perhaps this was only a matter of parameter tuning but because there was no need to use two separate machine vision algorithms it was decided to go on with only one.

Weed detection

The dandelions (or the yellow balls) were found from the robot's camera image using ECCI transformation and thresholding. Once a ball is found in the picture, it is added to a list of objects being tracked.

Besides adding new balls to tracking, the optical flow between consecutive frames in the camera's video images is calculated with the Lucas-Kanade optical flow calculation algorithm. OpenCV implementation of Lucas-Kanade filter was used [6].

As the robot moves ahead and the balls move towards the bottom of the camera image, the balls seen in the previous frame are assumed to be found in the positions suggested by the optical flow calculation algorithm. However, in case the balls are not found in the locations where they are assumed be, and this happens often enough, the corresponding record is removed from the list of tracked objects. The weed control sprays are triggered once a successfully tracked ball hits the bottom of the camera image.

The benefits of the system, described above, include successful tracking of the balls that are not seen in every consecutive frame (the balls are covered by maize leafs every now and then) and filtering out random non-dandelion observations.

Row end detection

Besides using the readings from ultrasound distance measurements, the end of the row is recognized from the picture of the camera. The algorithm for row end detection with computer vision gets an image as an input containing a binary representation of the pixels that are considered or not considered to be maize. Based on the input image the algorithm calculates one fuzzy logic output value. This output value tells whether the robot seems to be in the middle of the row,.

The interpretation of being or not being in the middle of a row is done simply by first dividing the input image vertically into three equally sized areas. For each area, the number of pixels that are interpreted as maize is compared to the total number of pixels in that area. Those relative values are then converted into fuzzy logic values ("Are there lots of pixels in the upper/middle/bottom part of the picture?") via certain membership functions. The robot is considered to be in the middle of the row in case at least two of the three sections contain a lot of maize-colored pixels.

Sensor Fusion and navigation

Robot's basic function was to navigate between maize rows. To be able to do this, it was necessary to estimate robot's position between the maize rows. The machine vision algorithms used for this purpose were earlier described in chapter 3.1.1. Besides of these algorithms there were also two alternative ultrasound based estimation algorithms (there were 4 ultrasonic rangers in the each corner of the robot). Outputs of these all algorithms are combined with extended Kalman Filter and the result is used in wheel controller [12].

The robot had to detect also the end of the row in order to make a turn in the headland. There were also several alternative algorithms to do this and their outputs were combined. This time basic probability theory was applied to fuse data. There were alternative ways to make turning in headland. Based on certain rules, the robot decided which way to use in headland.

The data processing algorithms only do not make the robot clever. There was also a need for higher lever logic. This logic concludes the current situation of the robot and

what is supposed to be done. All these things together – data processing algorithms, controllers and higher lever logic – makes the robot clever and gives it an ability to perform the task autonomously.

Operation logic

The intelligence of the robot was programmed with Matlab Simulink and Stateflow and it controlled the current state of the robot's program. In this UML-like statechart programming style there are different states and transitions. When right input signal is gained the program goes from one state to another. The program had own states for every function, like row driving, different turning methods and automated calibration. This state machine had a lot of hierarchical and parallel states to control all the tasks and tricks the robot did. Some of the basic features are discussed briefly in the end of this chapter and more important algorithms are presented in the following chapters.

The speed of the robot was in general controlled by using input speed as a set value for motor controller. In detail the speed value was modulated with the quality of different sensor data to ensure that the robot was able to keep between maize lines. In addition, the speed was rate limited to prevent slip on the ground and to spare the driving motor at stops.

The robot was able to drive forward and backwards with same capabilities. This was crucial while using crab style turning to navigate to the next row. The simulink model had a circuit to switch all the front signals to back and all the left signals to right and vice versa. Speed and camera angle was also inverted. That allows the robot to switch direction by changing only one Boolean value in the program.

The solenoids of the spray cans and the horn were operated with a separate system. It delayed the activation of spray cans based on the driving distance depending on the direction in which the machine was driving. The spraying system was trigged by the machine vision system when it detected a yellow ball. A sound from the horn sound was played after a yellow ball was detected and while spray cans were active.

Position estimation of the robot

Robot had four ultrasound sensors and a camera to sense maize rows. Because ultrasound measures included lots of noise and false measurements, it was not feasible to blindly rely on instantaneous values. First, ultrasound measures were

filtered and then processed further and fused together. This way a quite reliable estimation of the maize rows was obtained.

A special filter to filter out the outliers and other unsuccessful measurements from the ultrasound data was used. The filter kept track in which range current valid measures were located. Then if the latest measurement was not in valid range, a mean value of the recent measurements were used as an output. The range of valid measurements was obtained from the mean value of five latest unfiltered measurements. In addition to this the filtered measurements being smaller than mean value were used to determine the valid range. This prevents filter to drift out.

After ultrasound measurements were filtered, more advanced algorithms were used to determine the position of the maize rows. Two alternative algorithms were implemented to estimate the position of the robot based on the ultrasound sensors. One of the algorithms was completely independent and the other one was included in extended Kalman filter which also merged the outputs of all algorithms.

The independent algorithm was so called “history algorithm”. It used odometry to estimate the robot’s movement. Ultrasound measurements were placed in a space where robot’s current position was at the origin. See figure 9 in which this space is visualized. Then in every step this space is translated and rotated according to robot’s current movement and the new measurements are placed in the same space. Old measurements are removed when a certain amount of time has passed. This way clear maize rows are formed from sequential measurements. Rows are estimated from that data by weighted least-squares method in which recent measurements have more weight and the value of the weight factor is decreased exponentially as the measurements get older.

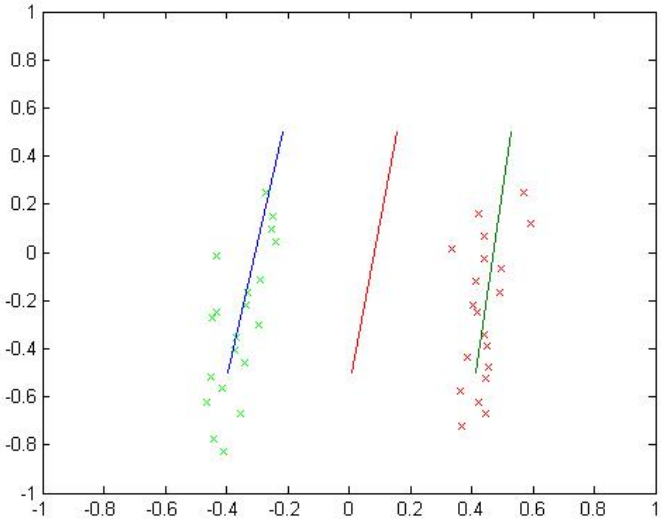


Figure 9 Maize row estimation using ultrasound measurement history

The other algorithm is built-in in extended Kalman filter. extended Kalman filter estimates robot's angular and vertical position between the maize rows (so called angle and bias error). This algorithm also uses odometry to estimate robot's movement. So, angle and bias errors are used as state variables, robot's movement is used as control variable and the filtered ultrasound measurements together with other algorithms outputs are used as measurement values. Because camera is directed towards the rows in headland, bias error output from machine vision cannot be used in headlands. Therefore there are two Kalman filters: one for navigating between rows and one for navigating in the headland. Headland's Kalman filter uses also compass to estimate robot's angular error. These Kalman filters give the final estimation of robot's position.

Wheel turning controller

When the robot is driving in the middle of the maize row the current heading and position of the robot are compared to the estimated center position of the maize row. The error in heading and position are minimized with a controller system that contains two PID controllers, as depicted in Figure 10.

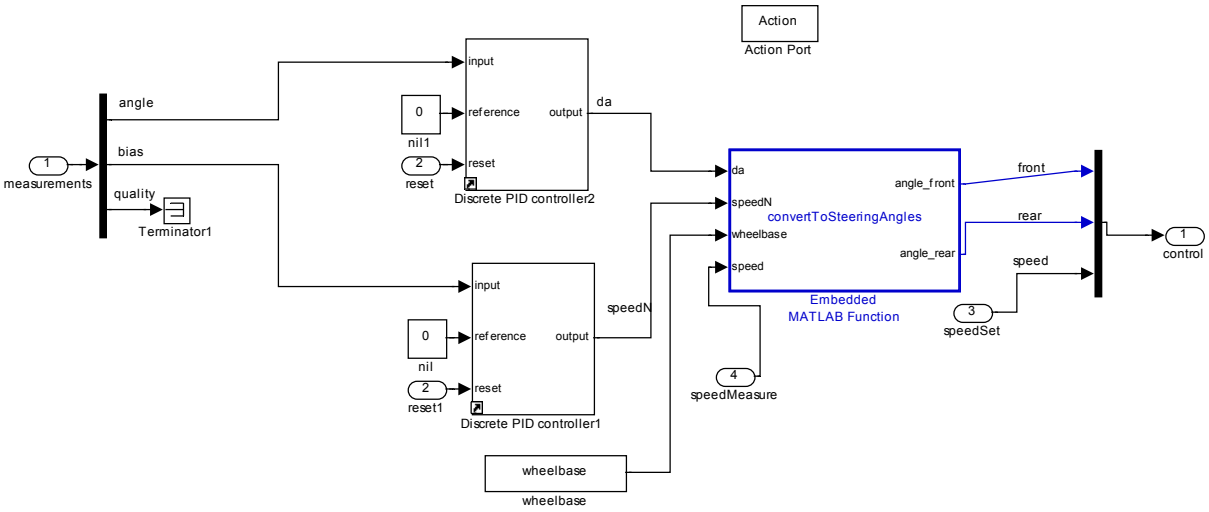


Figure 10 Block diagram of wheel turning controller

From the angle and sidewise position difference, a target angular velocity and sidewise velocity are generated with PID controllers. From those values, the front and rear steering angles can be solved when the kinematics of the robot are known.

Row end detection

Row end detection algorithm is based on probabilities. Besides of machine vision, ultrasound sensors and row length measured by odometer are used to detect the end of the row. Each of these algorithms is independent of each other and each of them gives out a probability to be in the row. Then these probabilities are combined using a given weight or a prior probabilities. The decision to be in the end of a row is determined by a given threshold value. The formula of the row end detection is:

$$InRow = \frac{P(InRow \mid MV)P(MV) + P(InRow \mid US)P(US) + P(InRow \mid Odo)P(Odo)}{P(MV) + P(US) + P(Odo)} \geq Threshold$$

, where $P(InRow \mid X)$ means the probability to be in a row given by algorithm X and $P(X)$ means a prior probability of algorithm X. A prior probabilities and the threshold value are parameters which must be tuned. Matlab scripts were used to determine these values. Weights were selected so that posterior probabilities of the machine vision and the ultrasound algorithms were equal in the middle of row. Then the

threshold value was selected to be higher than the inferred probability in the end of the row and lower than teh posterior probability in the middle of the row of each algorithm. A prior probability of the odometer is selected to be slightly higher than the threshold value. By selecting parameters like this, the row end detection algorithm can work even if some of the component algorithms detects a false row end.

Machine vision algorithm was described earlier in chapter 3.1.3 in context of machine vision. Therefore it is not revisited here again.

Ultrasound algorithm was based on quality values of the ultrasound-filter. The ultrasound-filter filters the measurements that are not in valid range. More information about this filter is in chapter 3.2.1 Navigating between rows. When the ultrasound sensors do not detect any maize, the quality values go to zero. The ultrasound algorithm counts these quality values of front ultrasound sensors and gives the probability by simply dividing the number of successful measurements by the total number of measurements in the decision horizon. The formula of ultrasound row end detection is:

$$P(InRow \mid US) = \frac{\sum_{n=1}^{N} \max(Q(US_{FrontLeft,n}, Q(US_{FrontRight,n}))}{N}$$

, where $Q(US_{FrontLeft,n})$ means the quality value of the front left ultrasound sensor and $Q(US_{FrontRight,n})$ respectively means the quality value of the front right ultrasound sensor.

The odometer algorithm is the simplest of component algorithms. This algorithm includes an assumption that adjacent maize rows have equal lengths. The odometer is set to zero when the robot starts to drive in a new row. Estimated true row length is updated every time when the row end is detected to be average of the previously measured row lengths. In the first row the algorithm doesn't know the true value of the row length and the output is zero all the time. After the first row the algorithm starts working. The output probability is higher in the middle of the row and goes to zero when row length is approaching the true row length. The formula of probability is:

$$P(InRow \mid Odo) = \max(\min(\frac{RowLenght - TrueRowLenght}{Variation},1),0)$$

, where *RowLenght* is the length of currently driven row, *TrueRowLenght* is the average of all row lengths and *Variation* is a parameter to be tuned.

Compass Kalman Filter

Compass and gyro data were combined with a Kalman filter. The filter was developed for two reasons. During the previous years it was discovered that the compass measurement had some noise that should be filtered. Other reason was that gyroscopes were not stabile and gyroscopes' zero points were shifting. All the measurements had some faulty measurements and a simple preliminary filtering method was developed too.

The linear system that was the basis of the Kalman filter had two states: angle x_1 and angular velocity (gyro) bias x_2. Input for the system was angular velocity u and measurement z was the angle from compass. Angle was calculated by integrating the angular velocity with the bias removed and bias should have stayed about the same. Hence the state model was quite simple:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t\left(u(k) - x_2(k)\right) \\ x_2(k+1) = x_2(k) \end{cases}$$
$$z(k) = x_1(k)$$

From this model the equations for the Kalman filter were derived and the filter was implemented in Matlab as a .m-file.

In tests it was found that the filtered compass angle was not accurate enough for headland turns and odometer based turnings were used instead. The problem was that when the robot tilted, compass gave false results and Kalman Filter didn't compensate this well enough. A conclusion was that a three axis compass would be much more reliable.

Headland turning

The robot could use two different turning methods. First turning method had two alternative realizations. The only difference was the angle estimation method. The first alternative used the angle from compass and gyroscopes estimated with Kalman filter and the second one used only raw odometer and wheel angle data. The Kalman filter based angle estimate proved to be more inaccurate than the simplest angle estimate from the odometer and the wheel angles. The angle from the compass chip seemed to have too much dependence on the robot's angle towards axis of gravity. That is why the compass based alternative was not used in the contest. The first turning method was named as normal turning.

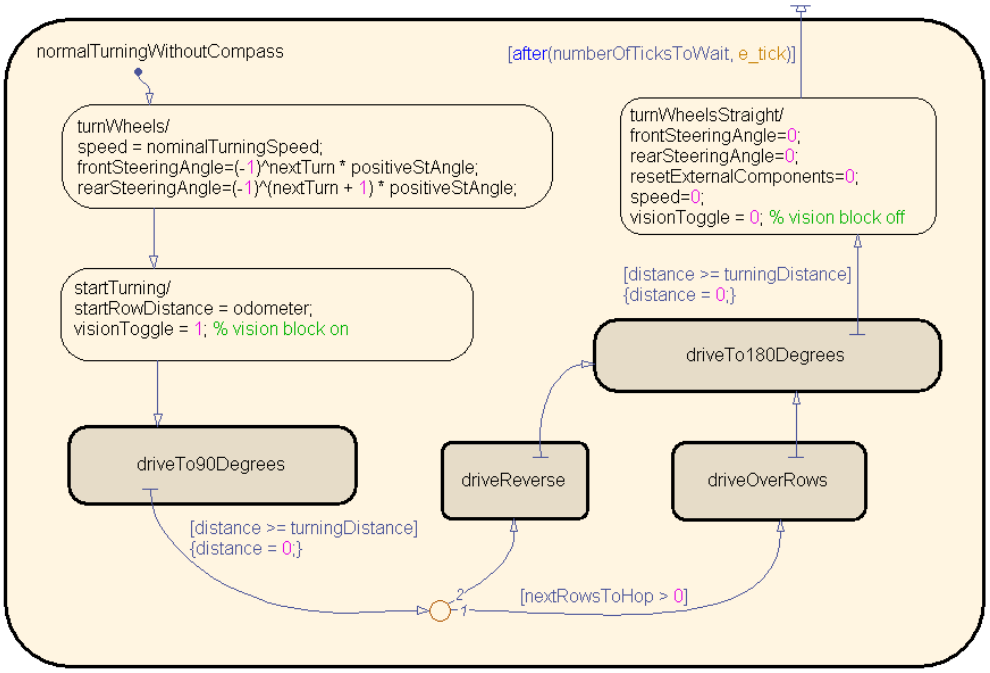


Figure 11 State chart of the normal turning method without compass usage

Normal turning was implemented on the state machine so that the wheels were turned to right angle first and then a 90 degree turn was made. Then the camera was turned to point towards the rows and then required transition in headland was driven and finally another 90 degree turn was made. In detail there were numerous adjustments in the angle and in the place using camera and row detector signals. Normal turning is demonstrated in the figure 11. Headland straight driving was done using 90 degrees rotated camera and same row driving controllers. The algorithm of the controllers kept the robot parallel to the maize lines and while camera was turned 90 degrees towards the field, the robot intended to drive perpendicular to the maize lines.

Headland driving had to be adjusted to the maize lines seen on side because odometer was not accurate enough to drive over multiple rows. This was done with camera and machine vision. Row places were extracted from the data of the row detector. The information of seen row places was added to the odometer distance to navigate more accurately to the right row. After these adjustments the robot could reliable drive over dozen of rows to the next desired row.

The robot was built to be symmetric at both ends and the crab-like all wheel turning was calculated to be the fastest way of turning to the next row. That is why the third turning method: crab style turning was implemented. We made the choice between

different turning styles automatically based on contest task and the number of rows to be skipped. Crab steering needed more space on the headland and could be used when neither 1.5 m space limit applied nor skipping of the rows was required.

Crab style turning was much simpler to realize than normal turning method. It first turned all wheels to the same angle, pointing to the direction of next row. Then robot drove a beforehand calibrated distance. After that above-mentioned direction switching circuit was used to swap front to rear end. Then wheels were turned to point to the direction of next row and the same calibrated distance was driven again.

Main program

Main program had two main functions; it worked as a user interface for testing and controlling the robot and it connected all the different components that were needed to get the robot running. Usually the main program was used from a remote desktop when it was actually running on a computer on board the robot. All of the important parameters could be changed from the main program. User was able to change the mode in which the robot was, pause it and override all controls. User could also drive the robot with a joystick connected to a remote computer. Main program logged all the important data that was collected by robot's sensors or generated by the different algorithms. This data was later used for debugging and parameter tuning.

User interface was divided into different pages. For example all parameters regarding the controller of the robot were on one page. It was also possible to set a route for the robot and see a graphical representation of it on one page. (Figure 13).

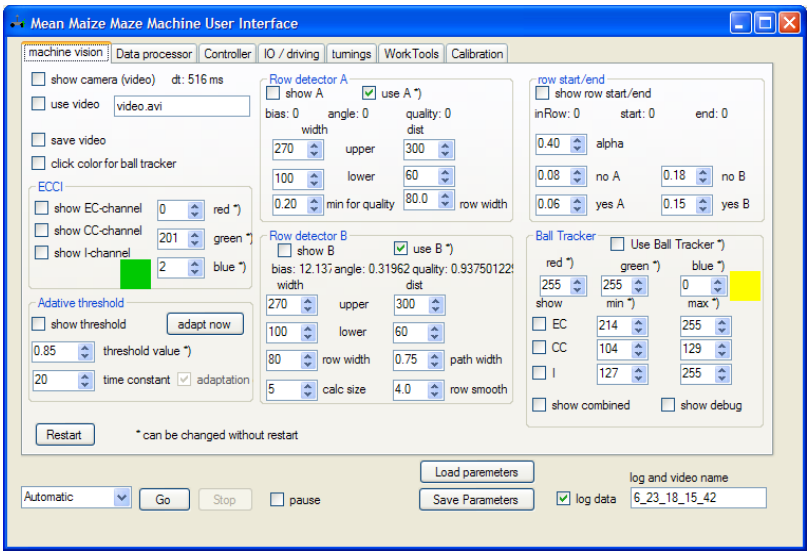


Figure 12 Basic view of the user interface

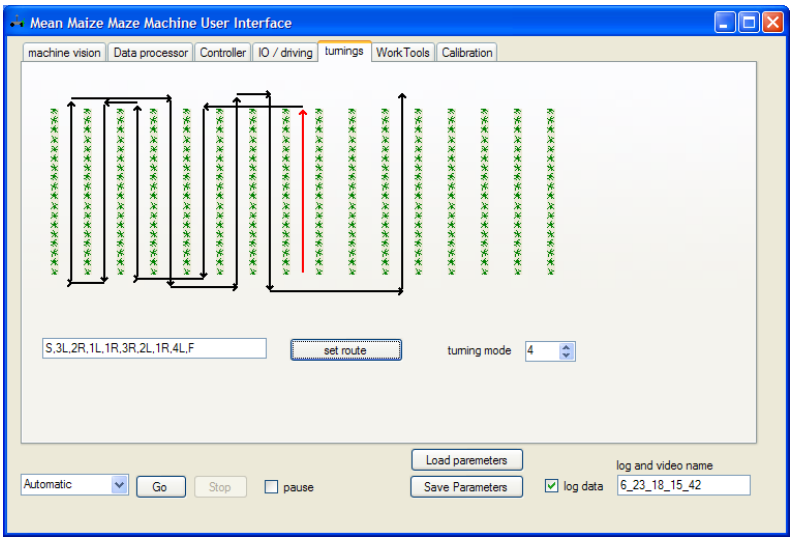


Figure 13 Robots route could be set from the main program

Simulator

As our robot was mechanically and electrically ready only some two weeks before the competition it was vital to be able to test our quite complex control algorithms beforehand. For this reason a simulator was created. The simulator consisted of two main parts: a Simulink part and a machine vision part. There were different maize fields generated for the simulator, some of which were harder and some easier for the robot.

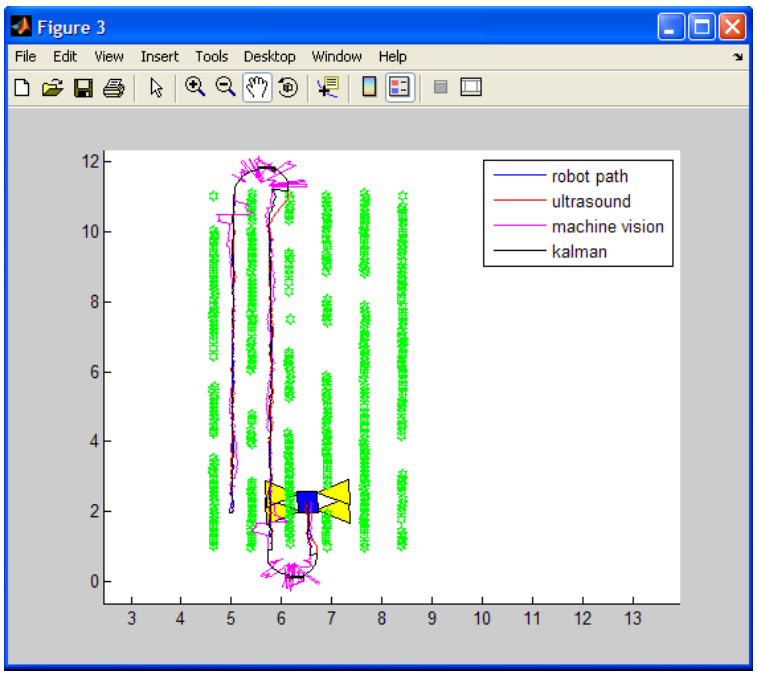


Figure 14 Simulated maize field with robot and some additional information

Simulink part simulated the movement of the robot on a maize field and generated the ultrasound measurements. These simulated inputs were then fed to the same

Simulink model that was used to control the robot. Machine vision part rendered similar images seen by the robot in a real field. Image was then processed by the machine vision system. Rendering was done with C++/OpenGL and the machine vision program and the Simulink simulator were connected via UDP link.

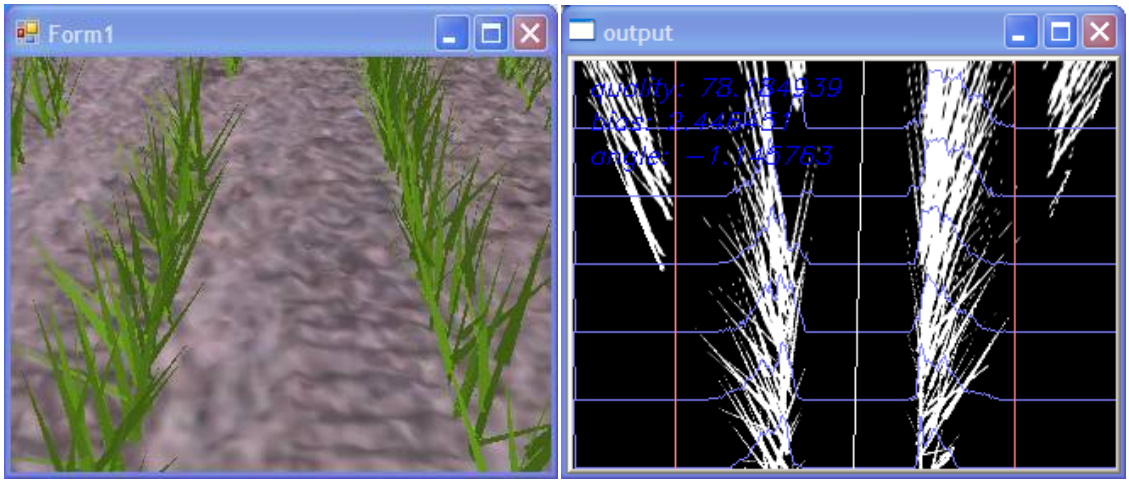


Figure 15 Simulators 3D rendered frame and corresponding row detector A debug image

Implements to robot

The robot was not only designed to fulfill the tasks that were in competition. It could be supplied with implements. Implements could be mounted in the rear axle suspension or in the robot chassis. Both of these options were used in competition.

Electric communication was basic RS232 serial communication line and protocol design follows ISOBUS like ideology [3]. However the protocol was synchronic and didn't follow completely the format of a CAN message. In our format there was a first identifier that tells which implement is connected and the rest of the message consist of data bytes followed by zero byte. Basic message format was equal in both directions and data bytes of each implement were agreed beforehand. The ISOBUS like ideology comes from a fact that implements can command the robot and also use all the sensors that it needs. The data processing was done in implement side.

Implements that were used in competition were Sprayer, Maize Counter and Seed Drill.

Sprayer

In task 3 weeds should be destroyed. A sprayer was used to do this because it is the most reliable way to interact weeds. Weed detection requires much intelligence so it

wasn't practical to do a stand-alone implement for this purpose. Instead the components required for this action were implemented in robot software and only actuation information was delivered to sprayer. Also this information is only digital output, so this implement didn't follow at all the general idea of what is described earlier. One could imagine that spraying was basic function of robot and not implement at all.

Weeds were detected with the machine vision system. This detection algorithm was described in chapter 3.1.4. After detection, this information had to be delayed so that spraying hits the weed. Some of this logic is described in chapter 3.2.1. The actual sprayer was simple: there was just one solenoid in both sides. These solenoids affected directly the hair sprays mounted below the solenoids.

Maize Counter

There was a need to count the maize plants in one task. A special implement was made for this purpose. It had four infrared distance-sensors (Sharp GP2D15) for detecting the plants and an Atmel ATmage168 microcontroller for counting and communication between the robot and the implement. Actually Maize Counter counted the empty spaces between the plants. It didn't recognize the species of the plant plant, so mistake detections could be occurred.

There were two infrared-sensors in both sides. Sensors were mounted close to each other so that there couldn't be two plants between the sensors. Because of this the maize plant trigged sensors in specific order. If the order changed then the algorithm concluded that either of the sensors had missed the plant or either of the sensors had detected false signal. To prevent miscounting there were also parameter for minimum distance between two plants. Parameter was not the same for both sensors, so occasional dense areas were counted too with rear sensors. Basic idea was to measure the distance of two plants with the front sensor and the rear sensor was for back up. When these parameters were set to be close to the real distance between two plants, leafs were not counted. The functionality of this system was proved with artificial maizes used in indoor tests during the winter, with birch stick field and also this worked very well in the competition.

Towed Seed Drill for Patch Seeding

Seed Drill's task is to find the gaps from the maize rows and place new seeds there. It uses the data from the main robot's ultrasonic sensors and makes the decisions independently

Mechanics

Mechanical construction is simple. Body and seeding arm are made of aluminum plate. The drawbar turns 45 degrees to both sides. There is a Hitec HS-815 Mega Sail Arm –servo turning it. Angle between main robot and drawbar is measured with a potentiometer. It is used only in the headlands, where it helps the trailer to follow the robot's trails. Seeding system includes a seed tank, a feeding solenoid, a seed pipe and a coulter. The solenoid is very cheap but strong enough to pull the spring in and release it rapidly, so it could give out one seed at time.

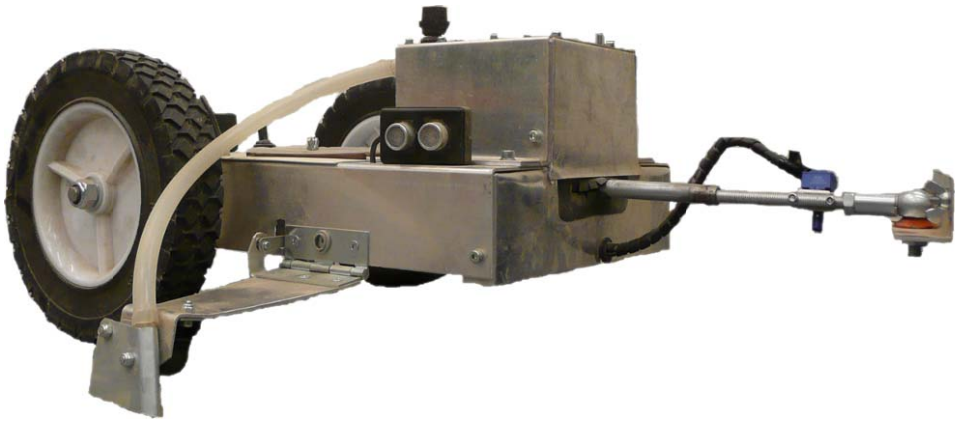


Figure 16 Towed Seed Drill

Electronics

The electronics is based on an Olimex PIC-P28 proto board with a PIC16f883 microcontroller. It has an external clock and it runs on 20 MHz. PIC generates PWM for both servos, makes a/d-conversion for the potentiometer, gives pulse for the solenoid and includes the logic. The most important task for microcontroller is to communicate with the main robot through a serial port. The Olimex board is already able to communicate with a serial bus, so it is easy to handle. Olimex board gets power from a standard 9 V battery. The servos and the solenoid are supplied with 7.2 V chargeable RC-car battery. Voltage is regulated to 5 volts with two regulators, one for big servo and another for smaller servo and solenoid. Microcontroller controls transistor switch which controls the relay controlling the solenoid. On the trailer there are also three ultrasonic sensors. Their data is sent to robot.

The trailer is programmed with PicBasic-language for PIC16F883. Whole program takes about 1400 bytes space. PicBasic was found not to be very efficient language. In order to gain a better control over interrupts, a C-compiler could be better than Basic.

The interrupt routines are the main core of the program. Pulse generation, A/D conversion and data transfer are all done in the interrupt routines. The rest of the program is driven in side. Controller's hardware PWM generator was unable to control servos so the PWM generation had to be made manually with timer and interrupts. Ultrasound sensors where used manually polling as well.

Program logic

The robot sent the data whether there was a gap in the maize row. This was done because the trailer's microcontroller had enough to do without analyzing ultra sonic data from the robot. When the robot sent the information about a gap in the row, the trailer started measuring the distance from robot's data and on the right position turned itself near to the row, lowered the seeding arm and released a seed. The trailer released seeds every 10 cm and when the robot told that the gap ends, the trailer lifted the seeding arm and turned itself straight at the right position. If there would not have been so many problems with programming the microcontroller, it would have possible to do much more intelligence in the trailer.

The main idea in programming was that the trailer acts as a master giving orders to the robot and the robot is a slave that gives the power to move. In movement the trailer commands the velocity of the robot (and the whole system), but steering of the wheels should be made in the robot side based on robot's row estimation.

Trailer's own ultrasonic sensors on both sides were meant to be used to control the beginning and the ending of the gap in a maize row and to keep the coulter on right line, but time ran out too soon. At first it seemed that the trailer was an easy part of the project to complete, but very soon limited resources of microcontroller and Basic programming language were met. This made completing the task much more difficult

A simple educational field robot, "Bambino"

As there was a considerable difference in knowledge of robotics, electronics and software within the group in the beginning of this robot project, it was decided that the

students at the University of Helsinki should get acquainted with basic robotics first. The requirement was that a 1:10 RC monster truck (HPI Wheely King) should be converted to a simple field robot beeing able to do the basic task of the competition indoors (driving between rows and making turnings to the next row). The idea was to make this with only two ultrasonic rangers, one steering servo, one speed feedback sensor and a PIC microcontroller.

Driving between the rows was done with a PD-controller. Ultrasonic sensors measured the distances from both sides. Results were saved to ten digits long vector, where the first number was always the newest result. Other vector included information about measuring thee results properly; number was either 1 or 0. This vector was also used to find the end of the row. If sum of the vector was 0, row had ended.

The preliminary project succeeded surprisingly well. Bambino was able to drive between the rows with a speed of about 1 m/s. Curves in test field were not a problem to Bambino on a test field made of folder covers. Also a test field done with chair foots was tested and successfully completed.

The headland area was the hardest task for Bambino. Driving with only odometer and ultrasound data was difficult. When the row ended bambino drove 75 cm straight forward to be sure that the row really ended. After that it drove back to the end of the row and turned wheels and drove half circle to the next row. About every second of the headland turns succeeded. Little bit more work for headland turnings would have been needed. Either a compass or a gyroscope would have been a great help in headland operation. In order to develop Bambino it seems to be necessary to have additional sensor to measure its position. In real field area this comes even more important because the ground may be very varying. It makes the odometer based driving unreliable.

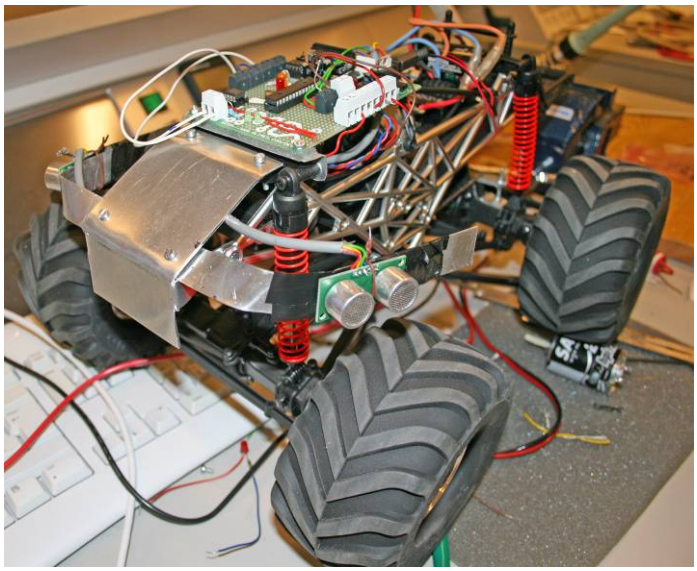


Figure 17 "Bambino" robot

However, even if this robot worked promisingly well indoors, further development was terminated after the learning period, since the whole group started to concentrate on the main robot and its implements. When looking back after the competition, it seems that with reasonable amount of work the Bambino would have become one good competitor in the event, at least in the first task.

Conclusions

It is not an easy task to make an autonomous robot to work in an unconstructed environment. Although the area where the robot is supposed to move is known, there are still huge variations in the size and form of maize plants. Also, there might be variation in row width, some plants can be missing and abnormal situations can occur. Weather conditions may change while the robot is moving. All these factors are things that must be taken into account when one is designing a robot to such areas.

Our solution was to make as much alternative algorithms and sensors that were possible and reasonable. Adaptation to current situations was also used. Downside in this approach a was huge amount of parameters to be tuned. Solution to this tuning problem was use of Matlab and some clever scripts that gather information from the test drives and estimated the parameters. Still there was a need for fine tuning the parameter manually. So what can be said is that testing is the key of success.

The mechanical structure of the robot is quite complex compared to usual structures used in this kind of machines. It was a real challenge to implement all the set requirements: equal support force in each wheel, balanced pose based on both axles, and suspension (spring+damper); to achieve a good traction and also to minimize swinging of camera on top. The novel suspension mechanism seems to be promising in this kind of agricultural field machines.

Most of the program of the robot was built by using advanced programming tools like Visual Studio and Matlab Simulink and Stateflow. Only minor parts of the program were done as raw coding. The Matlab Simulink model was generated automatically to C++ code using Matlab's Real-Time Workshop. It eased the programming and made our huge program and state machine easier to build and to get it work properly. It would be nearly impossible to get software as large as ours working without bugs in so little time without those software developing tools. The graphical user interface in the main program was built using Visual Studio and its capabilities to generate all visual elements of the main program automatically.

The intelligence of the robot was programmed with Simulink so that we could test it with our simulator. It expedited the development process of the program because we could test all the changes in a few seconds in our simulator and see if everything worked without going out to the field to test basic logics.

The whole robot is quite complex system with all the subsystems and algorithms. The huge number of tunable variables in every stage of the robot is a real challenge for testing. The tuning phase on the field requires a controlled practice as all the parameters cannot be tuned at the same time. Therefore during the development of the algorithms the tuning procedure for the parameters was already considered. It was known that there is very limited time to make the final tuning during the competition warm-up day. The well planned testing and the identification of parameter relations was one of the key factors to make a well-performing robot, especially with limited testing possibilities.

Finally it is emphasized that even if there are clear similarities in some algorithms and software framework to the previous robots built by the students from the same universities, no source code or functional models were used. Instead of that, the requirements for certain algorithms were given in literal format and the team has

implemented them in a novel way. Building the robot from scratch was seen as an important educational perspective.

Acknowledgments

Thanks to our sponsors:

References

[1] Adams, Motion simulation software package, MSC Software Corporation, http://www.mscsoftware.com/products/adams.cfm

[2] AVR Microcontroller, Atmel, http://www.atmel.com/products/AVR/

[3] ISOBUS, ISO 11783, http://www.isobus.net/

[4] Matlab, The Language of Technical Computing, The MathWorks, Inc, http://www.mathworks.com/products/matlab/

[5] Okamoto, H., Hamada, K., Kataoka, T., Terawaki, M. and Hata, S. (2002). "Automatic Guidance System With Crop Row Sensor" Automation Technology for Off-Road Equipment, Proceedings of the, July 26-27, 2002 Conference (Chicago, Illinois, USA), ASAE Publication Number 701P0502, Pp. 307- 316

[6] OpenCV, Open Source Computer Vision Library, Intel Corporation, http://opencvlibrary.sourceforge.net/

[7] Pro/ENGINEER, 3D Product Design, Parametric Technology Corporation, http://www.ptc.com/products/proengineer/

[8] Real-Time workshop, Generate C code from Simulink models,The MathWorks, Inc, http://www.mathworks.com/products/rtw/

[9] Simulink, Simulation and Model-Based Design, The MathWorks, Inc, http://www.mathworks.com/products/simulink/

[10] T. Maksimow, J. Hölttä, J. Junkkala, P. Koskela, E.J. Lämsä, M. Posio, T. Oksanen (advisor), J. Tiusanen (advisor), "Wheels of Corntune", Proceedings of the 5th Field Robot Event 2007, Wageningen, June 14,15 & 16 2007, ISBN 978-90-8585-168-4, http://www.fieldrobot.nl/html/Proceedings_FRE2007.pdf (accessed 24.6.2008), Pp. 75-88

[11] Visual Studio, Suite of Software development tools, Microsoft, Inc, http://msdn.microsoft.com/fi-fi/vstudio/default(en-us).aspx

[12] Y. Bar-Shalom, L. X. Rong, K. Thiagalingam. “Estimation with applications to tracking and navigation: theory, algorithms and software”, New York, Wiley, 2001, ISBN 0-471-41655-X Pp. 558.

Team AGRONAUT
Field Robot Event 2008

AGRO naut

Fachhochschule Osnabrück
University of Applied Sciences

AGRONAUT – Autonomous Field Robot

Timo Brenningmeyer, Axel Bruns, Christian Conforti, Roman Deters, Matthias Dünninghaus, Marcel Elberich, Tobias Flothkötter, Roman Greb, Julian Heitjan, Michael Hohaus, Stefan Lake, Lukas Pietruschka, Markus Ripke, Jens Thoben, Hendrik Voss

Abstract

The AGRONAUT was developed for the Field Robot Event 2008 by students of the University of Applied Sciences, Osnabrueck. The system is based on a modified model-scale chassis with all-wheel drivetrain and –steering. For navigation in the maize row infrared sensors mounted in the front bumper (4), at the sides (2 each) and at the bail (2) are used. They are supported by two ultrasonic sensors also mounted in the front bumper. For the weed-detection a CMUcam3 has been added to the robot. The weed-killing is done by two streamer-spray modules which can be attached to the back of the robot.

Keywords: motor-control, ultrasonic, infrared, sensor, CMUcam3, CAN-Bus, I²C-Bus

Introduction

The Field Robot Event 2008 was organized by the Fachhochschule Osnabrueck and took place in Haste on June 12th -14th. The tasks were mainly the same as in Wageningen in 2007. There were competitions in five tasks but only the first three tasks formed the main competition. There were two independent awards for the Freestyle and the Challenge Task.

The tasks were:

Robust Navigation

The robots had to navigate in a maize field with curved rows. At the end of each row a turn into the next row had to be handled. The maximum time for this task was three minutes, as the result the distance covered by the robots was measured.

Advanced Navigation

In a maize field with straight rows and plants missing the robots had to navigate and follow a predefined sequence of turns. Again the distance covered within three minutes was measured.

Weed Control

Yellow golf-balls were arranged in the maize field standing for weed. The robots had to detect the weed, show that by a clear signal and perform a weed-killing action. The number of detected golf-balls was registered.

Freestyle

All teams participating in this task had to perform a special action with an agricultural background.

Challenge Task

The robots had to navigate in a maize field with curved rows and slight slopes. Additionally several plants were missing and plenty of weed grew between the maize plants. Like in the other tasks the distance covered in three minutes was measured. Furthermore the number of plants along the travelled distance should be counted.

Chassis & Housing

As a mechanic base the Tamiya TXT-1 RC-model (Fig 1) was chosen which the two previous teams already used in the last years. The suspensions were not built up because the overall weight of the robot would make the springs being useless and the robot's driving behaviour becoming insecure.

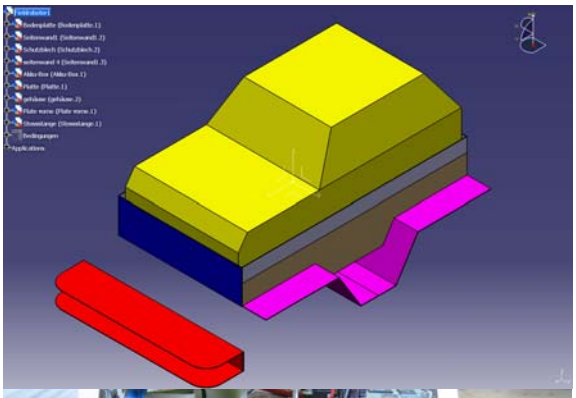
Fig 1: Tamiya TXT-1

Regarding the high stress on the steering system caused by the robot's weight and the ground characteristics, the strongest servo-motors fitting in the chassis were used.

Furthermore the differentials were filled up with differential-oil which has a high viscosity to get a particular differential lock in case the robot loses grip under a single tire. Moreover the steering mechanism was modified to achieve a turning radius of 75cm in maximum so that the head land turning could be done off the reel. Unfortunately the turning radius did not reach 75cm, so that a *Z-Turn* (compare [1, Amaizeing]) had to be performed to turn into the next maize-row.

The tires were filled with a combination of foam and thermomat stripes to generate the right hardness whereas the thermomat was put in an inner and the foam in an outer circle. To prevent overlapping of ends of the stripes they were stuck together so that they result in rings.

The design of the housing was done with the help of CATIA (Fig 2a). The main criteria for the design were easy access to the electronics on top of the robot and a modular assembly. So the Plexiglas-housing got a sliding mechanism using drawer slides from a DIY-store and the robot was split into two major parts: the lower nearly



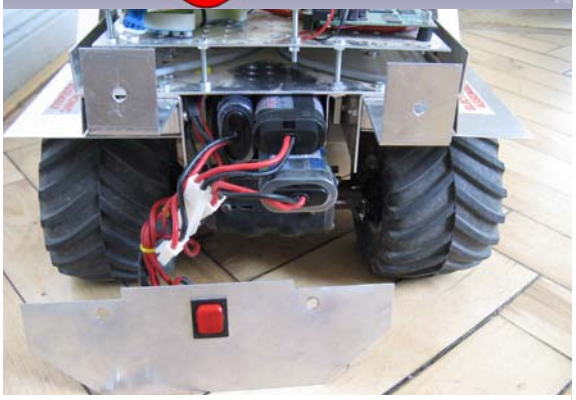


Fig 2a: CATIA chart of the aluminum and Plexiglas chassis
Fig 2b: Battery case

exclusively mechanical Tamiya chassis and the upper almost electrical part within the new designed aluminium chassis. Both parts can be separated with only 4 screws. So in case of defects every part of the robot can be accessed very quickly.
Furthermore the position of the rechargeable batteries was set to the lowest possible level to guarantee a better centre of gravity. The possibility of accessing the battery-box by removing a plate also enables a fast change of batteries (Fig 2b).

System Diagram

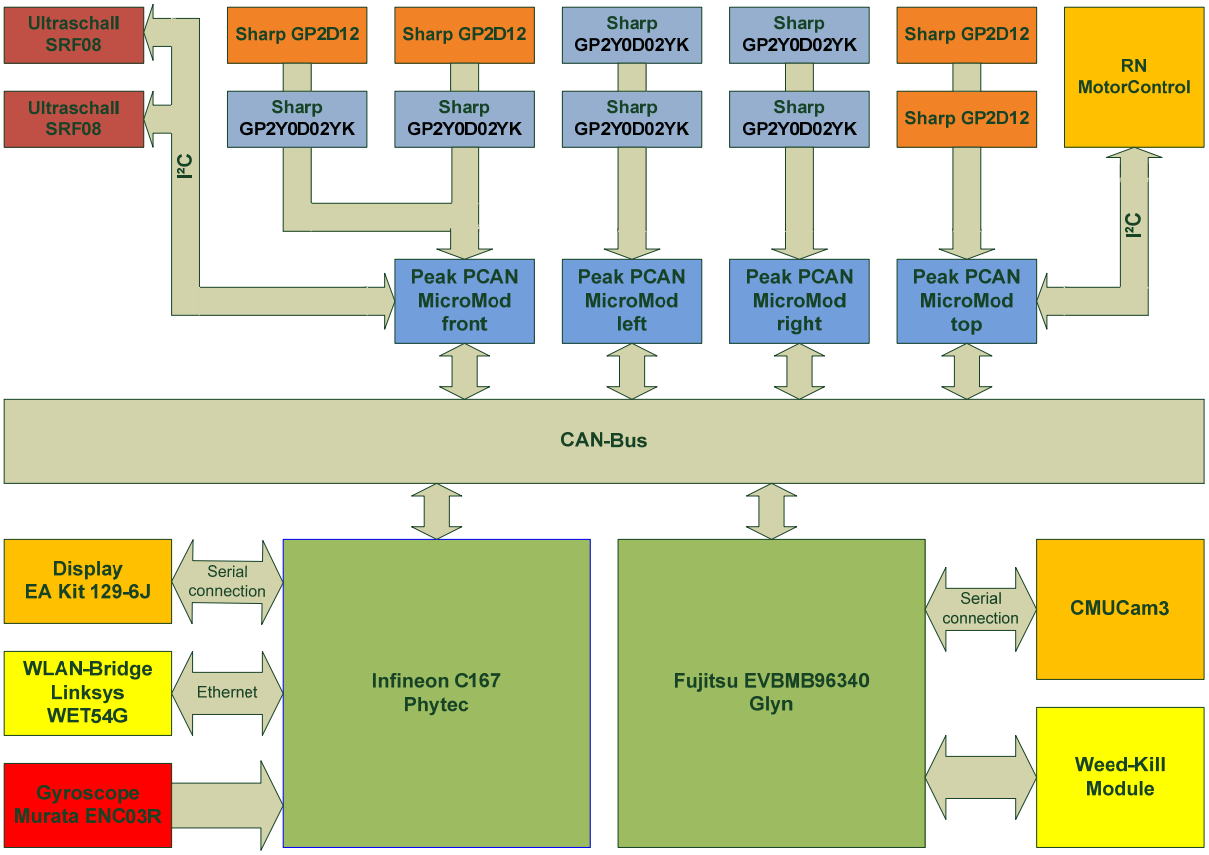


Fig 3: System Diagram

In the diagram above (Fig 3) you can see the setup of the electrical components of the AGRONAUT. It is easy to see that the main communication between the components is based on the CAN-Bus. The importance of the PEAK PCAN MicroMod modules as signal pre-processors and interface to the CAN-Bus can also be seen.

CAN-Bus

The CAN-Bus is the main communication channel on the robot. It connects the Phytec board with the Fujitsu board and the PCAN MicroMods. The communication is cut into several messages. The most important message gets the lowest ID and the less important message gets the highest ID. An overview for the communication is shown in figure 4.

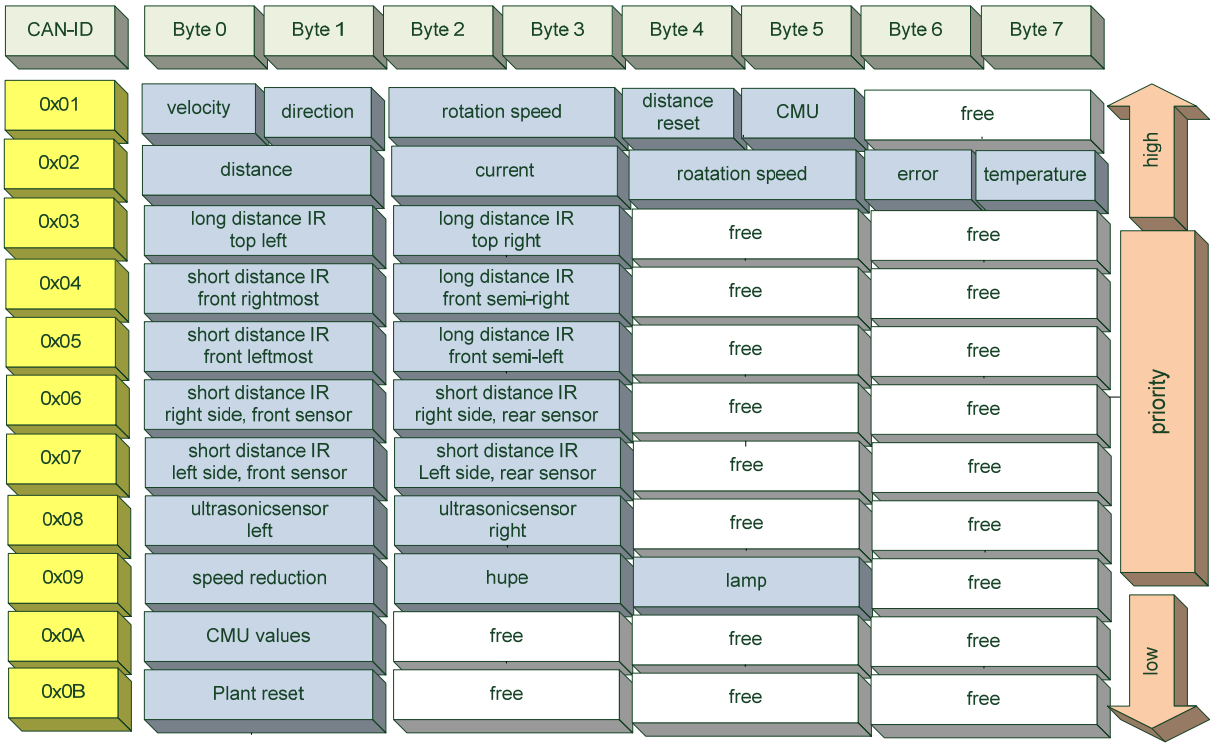


Fig 4: CAN-Bus diagram

Components

Phytec

The main processor board is a phyCore 167 HSE that contains an Infineon C167CS microcontroller. This microcontroller processes the sensor signals, the touch display and the gyroscope information and the motor control signals. The sensors and the motor-control are connected via CAN-Bus. Based on the sensor signals, the gyroscope signal and the distance information of the motor-control the algorithm computes the best way through the maize row and initiates and performs the headland turn.

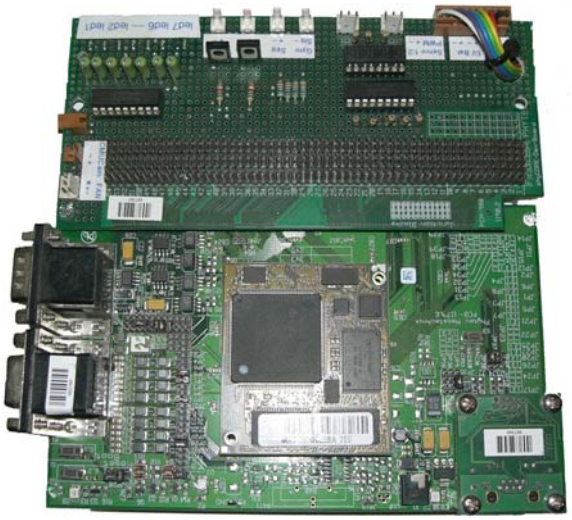


Fig 5: Phytec phyCore 167 HSE

The touch-display is connected by the serial RS232 interface.

To ensure the communication between the phyCore and the AGRONAUT-GUI the processor board is connected with a Wi-Fi.

The additional circuit board contains some status LEDs, the connectors for the flash light, the horn and the steering-servos. To separate the several energy supplies of the phyCore board and the steering-servos an optocoupler was implemented.

Infrared Sensors

Infrared triangulation sensors are used to measure the distance between the robot and the maize row. There are 10 infrared sensors on the AGRONAUT. Four long distance sensors with a scope from 200mm to 1500mm are used. Two of them are mounted in the bumper; the other two are located on the carrying handle of the robot. Further there are two short distance sensors on each side of the robot and two additional in the bumper. The sensors deliver an analogue output voltage which is non linear to the distance. The PCAN MicroMods were used to convert the voltage to a “cm value” on the CAN-Bus.

Fig 6: Sharp infrared sensor

Ultrasonic Sensors

Two ultrasonic sensors SRF08 were also mounted in the front bumper. The robot uses them to detect the end of the maize field and avoid a turning caused by missing plants. They were connected by I^2C to the front MicroMod module. Tests have shown that the mounting in the bumper was a failure. The sensors detected bigger stones on the field and were blinded by the dust.

Fig 7: SRF08 ultrasonic sensor

PEAK PCAN MicroMod

The robot uses 4 CAN-Bus controllers called MicroMod by Peak to convert analog signals for the CAN-Bus and to offer an I²C interface. These modules work with a F2MC-16LX microcontroller from Fujitsu.

For a flexible use of the MicroMod modules every module is plugged in a self developed clip board. All sensors, buses and power supplies are attached to theses clips. The triangulation sensors are connected to the analog input ports of the Peak modules. The two front ultrasonic sensors have an I²C connection to the MicroMod module. The top Peak module has also a connection to the motor controller via I²C. All modules are connected to the CAN bus and the 5V voltage supply.

Fig 8: Peak PCAN MicroMod & clip board

Peak offers a useful tool to configure these modules without any programming action just by sending some configuration data via CAN-Bus. For processing four sensor signals like in the MicroMod on the front bumper more resources than offered by the configuration software were needed. So the possibility of designing an own firmware was used. The modules can be programmed in C. For programming and flashing the software tools “Softune Workbench” and “Fujitsu flash mcu programmer” were used.

Motor-Control

Fig 9: RN Motor-Control

The motor-control is needed because of the slopes in the Challenge-Task.

For the AGRONAUT the RN-Motor-Control built by Robotik-Hardware was used.

This controller regulates the velocity of the two motors. To get the actual engine speed, a rotary encoder is mounted into the drive train. So the motor-control can compare the actual and the required speed. The motor-control is connected via I²C-Bus with one of the Peak MicroMod modules. The navigation-algorithm computes the correct velocity and places these information on the CAN-Bus. The MicroMod converts the CAN-Bus signal into an I²C-Bus signal and the motor-control collects this information from the I²C-Bus.

To control the power for the motors the controller board is connected with the RN-VNH2 driver board. This board appropriates the needed power to hold the required velocity. To handle the heating of the drivers a CPU heat sink and a cooling fan is mounted on the driver chips.

The motor-control also allows reading the actual motor current, the temperature of the driver board, the actual velocity and the driven way in centimeters. It places these information on the I²C-Bus.

Weed Detection & -killing

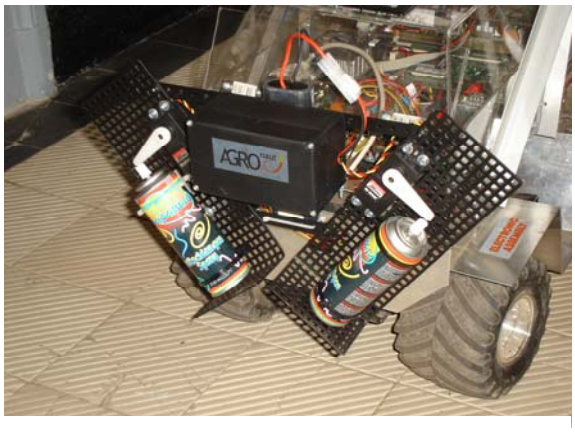


Fig 10: Weed Killing Module

The AGRONAUT is equipped with different actuators for weed killing action. To signal the detection of weed a horn and a flash light are used. The AGRONAUT has a removable weed killing device. It contains two streamer spray cans pushed by electrical servos and his own power supply with a 7,2V accumulator. The wired Glyn microcontroller has to decide whether the weed killing action should be performed on the left or on the right side. If there is an order to kill weed the servo pushes the can and the streamer marks the weed.

The AGRONAUT uses the CMUCam3 for the detection of weed which has a frame buffer and a small microcontroller. To detect the golf-balls the color tracking function of the camera is used in combination with the virtual window function. A virtual window is set rotationally to the left and to the right side of the screen before a color is tracked. For Color-Tracking the YCrCb-Colorspace is used which delivers better results than RGB because it is less sensitive to variations of brightness. To make it more resistant to noise the camera uses an objective with an infrared filter. Because of the light conditions are massively influencing the results of the color tracking, the used color minimums and maximums were set only a few minutes before the challenge was started. To determine the correct colors the CMUcam2 GUI was used.

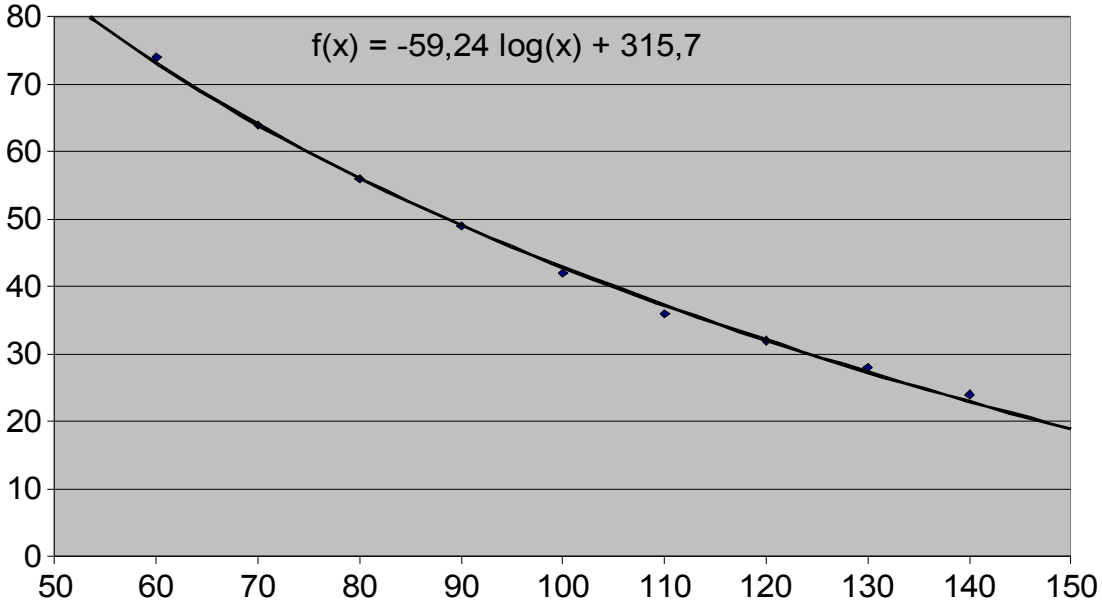


Fig 11: characteristic diagram of the CMUCam 3

The preprocessed information (e. g. x- and y-coordinates of the detected ball) are send to the Fujitsu microcontroller by a serial connection. Having this information the algorithm is able to calculate the distance of a ball with the aid of the formula and chart above (Fig 11).
The algorithm also has to ensure that noise and short-time covered balls don't cause (re-) detection. Therefore the microcontroller calculates the average over the distance of the last detected ball and compares a new distance with it. If the new distance is in the tolerance area, the value is used to calculate a new mean. In case of a longer distance (-tolerance area) than the previous it could be a new detected ball. This is verified by a new mean calculation.
When the AGRONAUT reaches a ball the weedkill-action is performed. Additionally a signal is send over the CAN-Bus to set the detection signal with the horn and the flash-light.

Glyn

The Glyn evaluation board has a F²MC-16FX MB96340 microcontroller by Fujitsu. The board is used for weed detection and initiation of the weed killing action. The self developed additional circuit board contains eight status LEDs and a PWM (pulse width modulation) electronic for the weed killing action. Furthermore it has a CAN driver chip which realizes the CAN connection. There are two optional useable digital outputs on the board.

CMUcam 3

The CMUcam3 is used to detect weed. The firmware emulates the CMUcam2. Since there is the need to decide whether the weed is left or right a virtual window splits the picture in two halfs. So the virtual window alternates between the left and the right side and it is only essential if there is weed detected or not. So you don't have to compute the coordinates. You only need to know on which side the virtual window is and if the camera has detected the corresponding color.
The camera is connected via a serial RS232 interface to the Glyn processor board.
The camera has a wide-angle lens with an infrared filter.

Conclusion

The first improvement we would make is to design another chassis. The criteria would be more stability and a smaller turning radius to achieve a direct turn form one row into the next. The chassis should also include a suspension to filter the uneven ground and make the sensors deliver more reliable data.

Further the ultrasonic sensors must not be mounted in the front bumper; there is a danger of detecting stones on the ground instead of plants. An additional danger is that the sensors are damaged by dust. A higher mounting position would lower those risks.

Sources

[1] Proceedings of the 5th Field Robot Event 2007, Wageningen, June 14, 15 & 16 2007

SunnyMaizing

-Team AMR-

Team members: Jan Peter Vornhülz
Jan-Hendrik Arling
Maximilian Terveer
Thomas Schierbaum (tutor)

Affiliation: Gymnasium Carolinum Osnabrück
Department: JugendForscht (Project AMR)
Große Domsfreiheit 1
49074 Osnabrück/Germany

Abstract

The robot SunnyMaizing is our first self-made robot within our JugendForscht-project. Besides, we participate in the Filed Robot Event (FRE) 2008 to gain experience for the next years. Our team consists of three pupils (9th and 10th grades) from the Gymnasium Carolinum Osnabrück and our tutor. The concept was to create a robot to make some experiments, to gain experience and to fulfil the tasks of the FRE successfully.

Fig. 1: Field robot SunnyMaizing

The plans of the design of the chassis, of the mechanic, electric, sensoric and controlling concept were created and developed by ourselves.

1 Introduction

This year the Field Robot Event took place at the University of Applied Sciences Osnabrück from 12th to 14th June 2008. We participated for the first time and so we did not expect too much. The robot was built parallel to our normal school activity all over the last year.

The tasks were similar to those last year:

- **Navigation:** The robot has to navigate through curved maize rows.

- **Advanced Navigation:** The robot has to navigate through straight maize rows in a pre-defined way. The special challenge of this task were the incomplete maize rows.

- **Weed Control:** The robot has to navigate through straight maize rows and detect some weed in the form of yellow golf balls.

- **Challenge Task:** The robot has to navigate through a special field and has to count the maize plants.

- **Free Style:** The team can play with their whole creativity, but an agricultural use should be there.

The aim is to cover as much distance as possible within 3 minutes. On the other hand the amount of touches on the robot by the team decrease the rating.

We participated in Weed Control and Freestyle, but we wanted to participate in Navigation and Advanced Navigation, too. But unfortunately we did not have enough time, because we had some problems on the day (and in the night) before the FRE.

2 Conception

This year was our first participation, so we had no experience from the years before, unlike some of the other teams.
We constructed our robot very simple and not really professional, but special in our own way. The reason for this was that we are pupils and not students; and we did not have so much time and not much money, either.

SunnyMaizing is based on the self-made and self-planned chassis with our own drive concept. The robot navigates with a special sensor concept and so it can manage the FRE tasks.
The single parts of the robot are explained in the following documentation.

3 Mechanics

3.1 Chassis

SunnyMaizing is based on an aluminium chassis with the following measurements:
Length: 42 centimetres
Width: 46 centimetres (chassis: 30 centimetres; both wheels: 16 centimetres)
Height: 60 centimetres (chassis and wheels with the camera tower)
Weight: about 6 kilograms
The chassis is constructed in such a way that you can open it and a level is integrated between the base and the top where the controller board is put on.

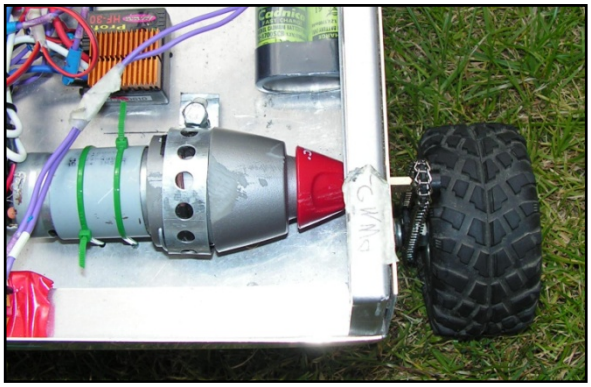
3.2 Drive System

The first question we had to answer was what kind of drive we would use.
We decided to use a chain drive, because we thought it would be helpful if our robot could rotate on the spot. Our drive system based on two cordless screwdriver engines, two waves, drive wheels and a linkage.
The linkage between the chains and the drive wheels was our most difficult problem, because the chain often breaks free from the drive wheel and we needed much time to construct a chain adjuster. Another big problem was that the chain often blocked or was blocked by the drive wheel.
Only 12 hours before the event we nearly had to capitulate, because one of the driving chains broke and was destroyed. We still decided to take part in the event despite all these problems, because we wanted to achieve our aims, but we could not start with our robot and so we had to build a new undercarriage/drive system (while the German football-match was running!).
As we did not have much time and money, we wanted to use as much of the “old” drive system as possible for the new system.
Two things had to be improved: The ground clearance and the celerity.
Finally we bought two model-wheels and small parts for the attachment and a cupboard wheel.
All in all, our new robot is much better and faster than the old one.
The new drive system is nearly the same principle but we changed the chains for three wheels (two wheels to drive and one wheel to stabilize the robot).

It was difficult to change the robot from chains over to wheels and we had a sleepless night but the result was and is great considering the short time we had.
Moreover, the engines are controlled by two engine controllers which are controlled by the controller.

Fig. 3: Bird´s eye view of our drive system

Fig. 2: New undercarriage based on wheel system



4 Electronics

4.1 Microcontroller

For controlling SunnyMaizing we use the microcontroller board *phyCore-167 HS/E* equipped with an Infineon C167CS microcontroller. The reason for choosing this board was that we informed ourselves about the last year robot “Amaizeing” and so we saw that this controller can be used. Besides, the company Phytec became one of our sponsors and so we got the board plus all additional equipment for free.
The controller is the brain of our robot: here all information (e.g. results of distance check, camera results) comes together and orders (e.g. motor orders, signal orders) are given.

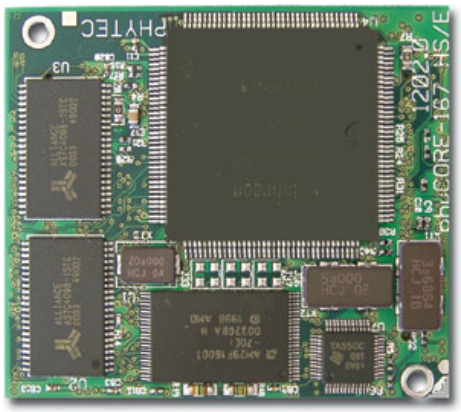


Fig. 4: phyCore-167 HS/E

4.2 Sensors

4.2.1 Distance sensor:

We use Sharp GP2D120 sensors which are analogue infrared distance sensors and measure with a triangulation principle. Six of these sensors are put up on SunnyMaizing and measure the distance to the maize rows continuously.
They are mounted in an angle of 45° in driving direction and the measured results are converted into the distance to the row by cosinus in the program.

Fig. 5: Sharp GP2D120

4.2.2 Camera:

We use the camera CmuCam2, a little, fast and nice camera, to detect the yellow golf balls in the “Weed Control”-task. The camera gives information about the colour of the pixels in the picture and does not transmit the whole picture. The specific colour information is given to the controller by RS-232 connection.
The reason for buying the camera was that we wanted to participate in the “Weed Control”-task and this camera is perfect for this job.

Fig. 6: CmuCam2-module

Others:

Besides, we bought two whiskers (FLEX sensor) to prevent a crash with maize plants and a compass module (CMPS03) to navigate in the row perfectly. But unfortunately we did not have enough time to integrate these sensors.

Power Supply

Our power supply consists of two independent parts:

One accumulator is responsible for the engines and one for the electronic, that means the sensors and the controller board. Both accumulators have 7.2 volt.

The engines need between 6 and 9 volt, and so 7.2 volt is suitable. So this is the first electrical circuit. The camera needs about 7 volt, but the sensors and the controller board 5 volt. So the second circuit is separated: on the one hand 7.2 volt for the camera, on the other hand 5 volt, which are regulated from 7.2 volt, for the sensors and the controller.

We think this is a good strategy to supply our robot with power.

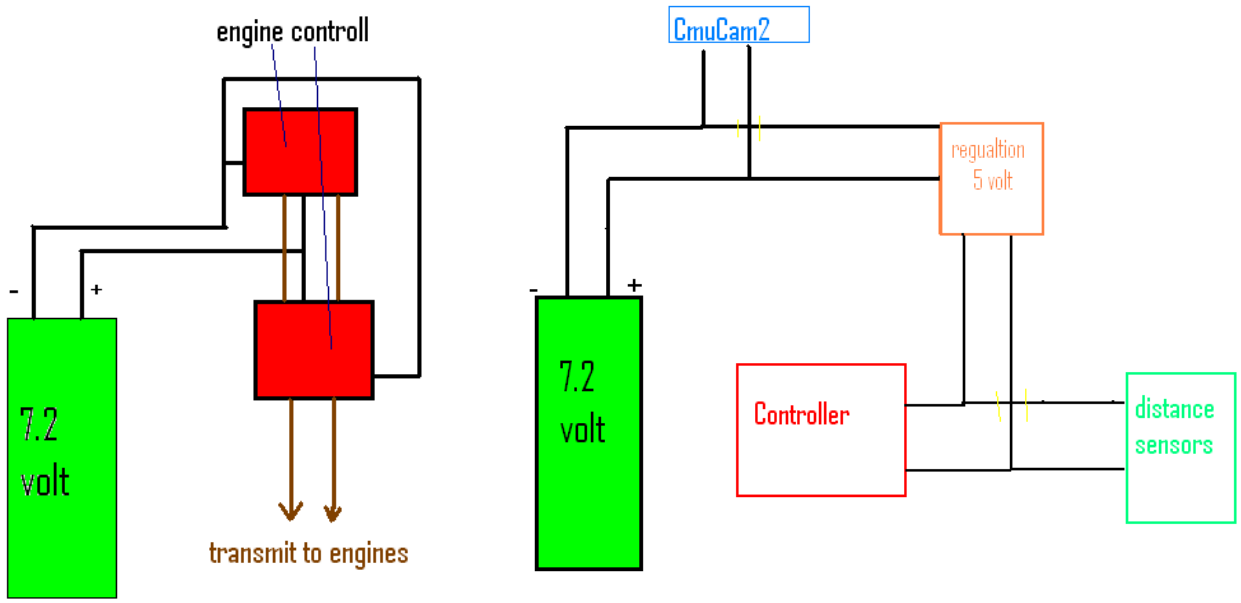


Fig. 7: Our power supply concept

Fig. 8: The accumulator of 7.2 volt which we used

5 Software

We programmed our microcontroller board with the suitable programming software from Keil, a Software company and one of our sponsors.
The name of this program is µVision3 (PK166) and it can compile, debug and create the HEX-data which is transmitted to the controller.
The programming language is C.
Additionally we used the Windows Hyperterminal to see what the program is doing at a particular moment or to look at the sensor results.

Our program is self-made. The main aspect of all tasks (except Freestyle) is to navigate through the rows. So the distance measurement is important and we developed our own strategy to navigate through the rows:

We measure with all six sensors and every sensors fifty times, then we calculate the average of every single sensor, because theses distance sensors are not very exact and the values can be improved by more measurements. The results are converted into centimetres-results and impossible results (e.g. 0) are “erased”. The three results of the left side are averaged in one value for the left side. The same method is used with the results of the three right sensors.
Now the program has the values of measurement and has to react.
If the two results for left and right are nearly equal (with a little tolerance, because to find the ideal drive way is nearly impossible), the speed can be raised for a short time.
If the left side has a higher result than the right side, the robot changes direction towards the left side. If the left side has a lower result than the right side, the robot changes direction towards the other (right) side.
So the robot can navigate through the maize rows with this simple method.

Our engines are controlled by PWM-signals from the microcontroller board. In the program you can set which engine is controlled and the speed of this engine can be diversified in percent (Between -100% and 100%).

6 Realisation of the tasks

Our solutions to the tasks:

Navigation:

We drive through the maize rows with the help of infrared sensors and the above described method.
If the robot does not see any maize plants he knows that he is at the end of the row and he decides to turn around.

Advanced Navigation:

It is similar to the task “Navigation” but in this task the robot orientates itself on only one side. He does not need both sides to drive through the rows. We count the rows if we have to drive a special, pre-defined route at the end of the rows.

‘Weed’ Control

We drive through the rows similarly to the task “Navigation”. We search the yellow golf balls with the CmuCam2, which is pretty good because we program the camera that she has to search only a special colour code. That is the reason for a fast signal (in form of a beep) of detected yellow golf balls.

Freestyle

We use a special device in front of the robot to play football. But the concept was “fun over use”.
The motives to do this were the European Football Championship 2008 and the fact that our team consists of football fans.

Fig. 9: SunnyMaizing in „Freestyle/Football“-mode

7 Results and Conclusion

All in all we are very happy about the results, because we had many problems before the FRE.
The concrete results are:
we participated in two tasks (Weed Control and Freestyle)
we won a special price (we and another team were the youngest competitors)
we gained much experience and knowledge

We benefited by acquiring new knowledge about the robot and its advantages and disadvantages.
For example, we know now that the drive system should be planned very well and that the ground clearance and celerity of the robot are important.
The electronic and sensor concept and the program worked very well, but it can surely be improved, too.

Now we want to plan a new robot with our new knowledge and want to participate with this robot in some competitions (JugendForscht, CAROLO, FRE) next year.

So we are looking forward to FieldRobotEvent in Wageningen 2009.

Ending

List of pictures sources

Fig. 1: Arling, Jan-Hendrik (private source)
Fig. 2: Arling, Jan-Hendrik (private source)
Fig. 3: Arling, Jan-Hendrik (private source)
Fig. 4: Phytec Technologie Holding AG (free internet picture)
Fig. 5: Roboter-Teile.de (free internet picture)
Fig. 6: Roboter-Teile.de (free internet picture)
Fig. 7: Arling, Jan-Hendrik (made with Windows Paint)
Fig. 8: (free internet picture)
Fig. 9: Arling, Jan-Hendrik (private source)

Acknowledgement

Our team would like to thank our sponsors, because without their help our project would not have been successful.

- Sparkasse Osnabrück, Germany
- Phytec Technologie Holding AG, Mainz, Germany
- Roboter-Teile.de, Dresden, Germany
- Keil, Grasbrunn, Germany
- Toom Baumarkt Belm, Germany

Besides, we would like to thank Ralph Klose and Andreas Linz from the University of Applied Sciences Osnabrück. They helped us in difficult situations and gave us some good information and advice.

Contact

Jan-Hendrik Arling
+49 151 56920860
jan-hendrik.arling@freenet.de

Jan Peter Vornhülz
+49 151 55506091
jan@vornhuelz.com

Team mail address: amr.robo.jufo@googlemail.com
Team homepage: not available yet

Please contact us for more information, questions or suggestions.

Michael Meinecke, Markus Robert, Jan Roesler, Lennart Roos, Jan Schattenberg, Martin Schwerter, Thomas Göres

Hard- and Software concept of the autonomous Field-Robot Helios

FREDT-members 2008

TU Braunschweig

Technische Universität Braunschweig
Institut of Agricultural Machinery and Fluid Power
Prof. Dr.-Ing. Dr. h. c. Hans-Heinrich Harms
Langer Kamp 19a, D- 38106 Braunschweig, Germany
www.tu-braunschweig.de/ilf

FREDT
TU BRAUNSCHWEIG

1 Electronics

Figure 1.1 shows the structure of the bus-concept, the interfaces and the different sensors and actuators.

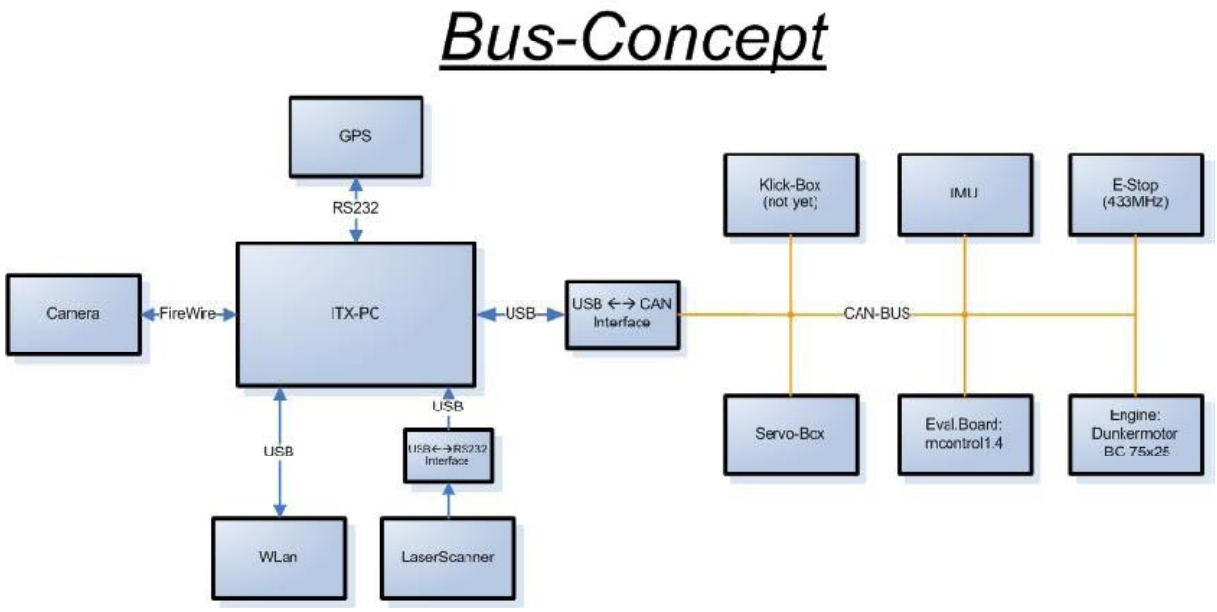


Figure 1.1: Bus-Concept

On the basis of last years experiences, concerning the I2C bus the whole bus structure had to be reengineered. By the grown sensor concept, unfortunately the I2C bus showed fast its fault liability. In case of failure of only one sensor the total failure of the bus was the result. The low flexibility to add or to remove single nodes has been another disadvantage.

The CAN bus, which is commonly used in automotive electronics, was chosen. The high fault tolerant protocol, the multi-master-system, bit rates up to 1 Mbit/s and the excellent support makes it the ideal choice. Furthermore it is very easy to listen on the bus, which is helpful for debugging.

The last years central BlackBox system has been removed. The BlackBox was responsible to connect the ultrasonic sensors with the I2C bus, to generate the servo signals and to transmit and receive data via RS232 to and from the ITX-PC. The BlackBox was the single point of failure and the RS232 interface the bottleneck.

Instead a decentralized system was developed. The main advantages are:

- no RS232 bottleneck between the pc and the peripherals
- faster exchange of defective parts
- simple troubleshooting
- the single modules can be set to sleep mode to reduce power consumption
- easy to add additional capacity or remove non-used parts
- high flexibility of arrangement (module next to its usage site)
- smaller harness

The following chapters give a general idea about the different modules.

1.1 PC-CAN-Dongle

For the PC-CAN interface a TINY CAN I module (figure 1.2) from MHS-electronics is used. The implementation of the dlls in the C-Code was quite easy. The module simulates a serial interface via the USB port. It can achieve bit rates up to 1 Mbit/s and has several internal transmit and receive buffers.

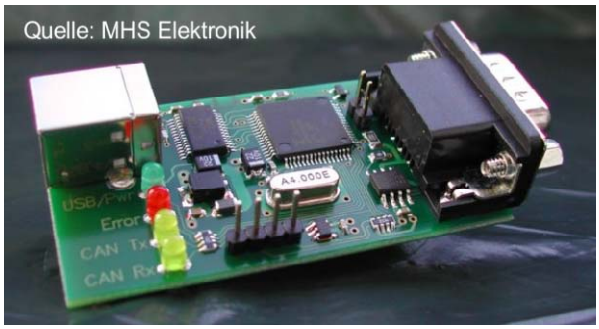


Figure 1.2: PC-CAN-Dongle

1.2 E-Stop

An „Emergency-Stop“ stops the robot in the event of a malfunction. As soon as the error has been fixed, the robot can be restarted. The remote control works on 433MHz and sends a coded signal to ensure no unwanted behaviour of the E-Stop.

1.3 Evaluation board

The evaluation board rncontrol 1.4 from robotikhardware.de with an ATMega32 microcontroller serves for quick testing of sensor and actuator systems. The board had been expanded with a CAN interface, a small buzzer and an LCD display to visualize the data of the various CAN messages formatted. Thus it is an easy and fast debug interface.

1.4 IMU

An inertial measurement unit permanently measures the yaw rate and the acceleration in the X and Y direction of the vehicle. As soon as the vehicle stops for a

few seconds, the sensors are automatically recalibrated and temperature compensated. Via CAN the data is sent to the PC to realize a better intra row and row-end navigation.

1.5 ServoBox

Figure 1.3: ServoBox

The ServoBox, which is shown in figure 1.3, creates the servo signals and the servo power. A small switching regulator LTC1374 (efficiency > 90 %) and a linear regulator LT1121 transform the 24 V battery voltage to 5 V logic voltage. For the servo power a LTC1775 switching regulator is used with a maximum output current of 10 Ampere and an efficiency of more than 95 %. The servo signals are generated by an ATMega8-microcontroller. To reduce power consumption during programming phases, the ServoBox is able to shut down the servo power and to enter sleep mode if no CAN message is received for a few seconds. Figure 1.4 shows the schematic structure of the ServoBox.

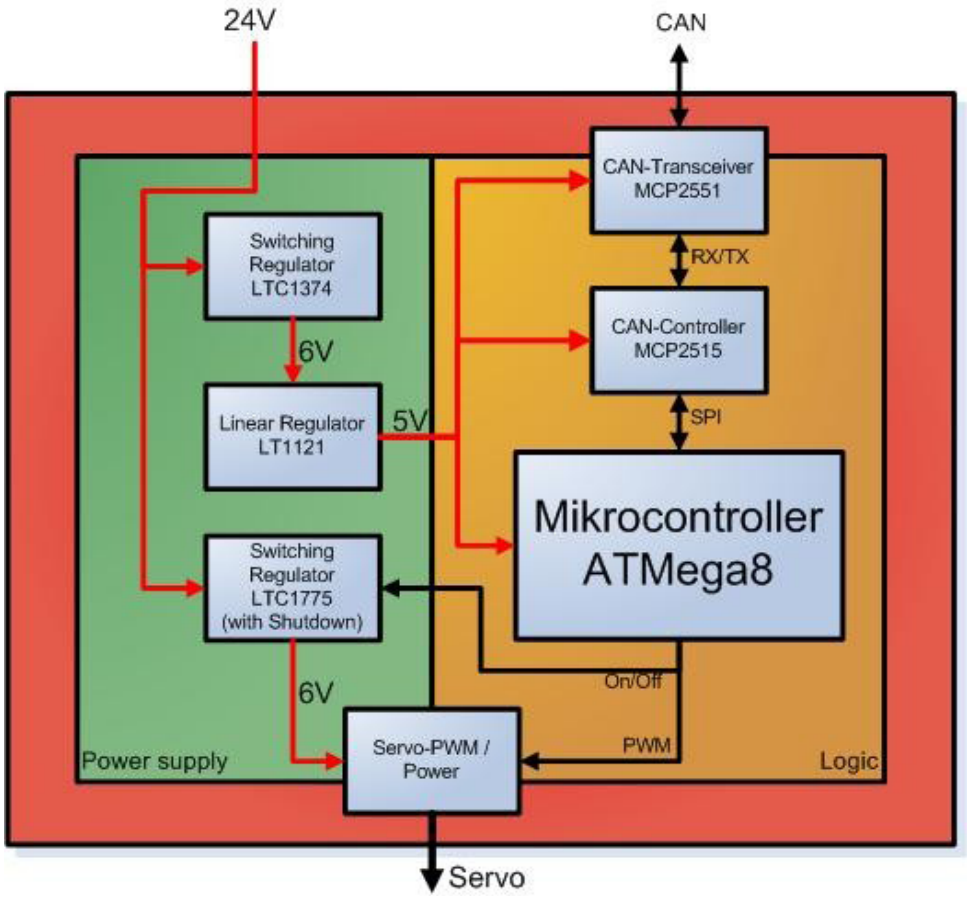


Figure 1.4: ServoBox schematic

2 Software-Concept

The software for the field robots from type “Helios” is developed in C++ with Microsoft Visual Studio. The idea was to develop a very modular software which allows easy integration of new sensors, actors and applications. Further requirements were to have wireless and also non-wireless communication with the HMI (iPAQ with the self-written operator interface), which was already used last year [source: Proceedings 2007, ISBN: 9789085851684 Date: December 2007 Publisher: Farm TechnologyGroup p. 35 ff] and to integrate a can-bus interface for connecting actors and sensors. Figure 2.1 shows a simplified figure of the software-concept.

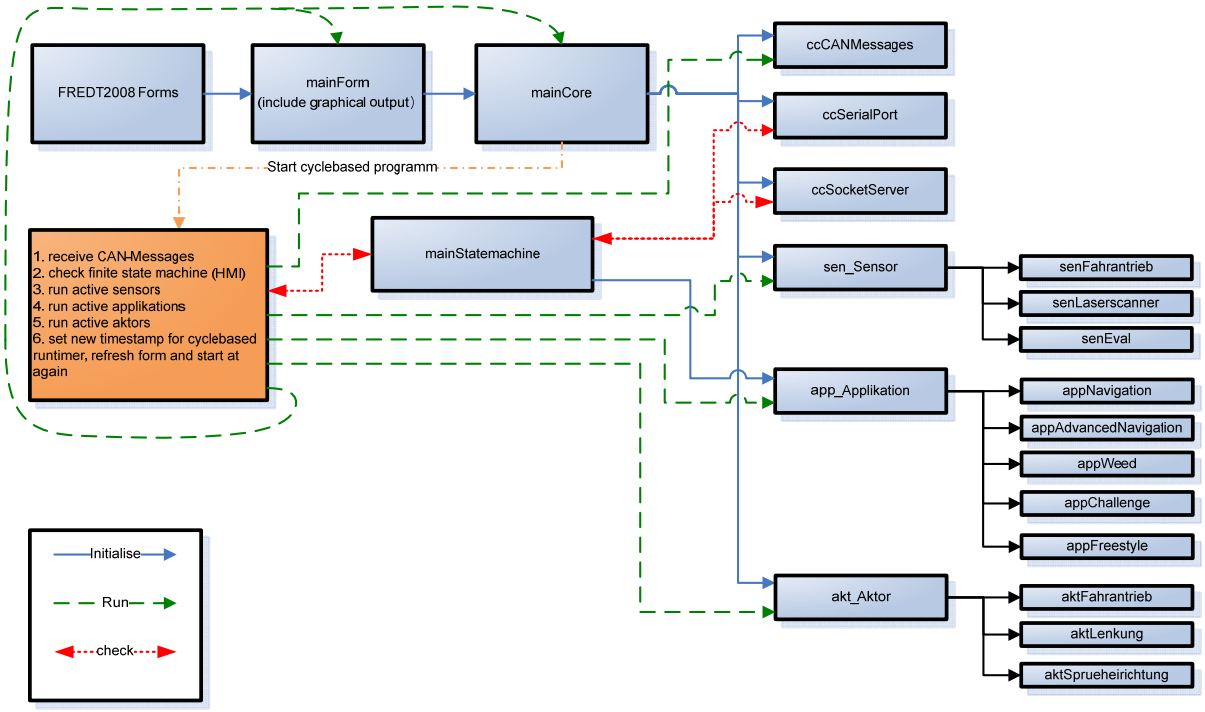


Figure 2.1: software-concept

The three blocks in the upper left corner of the figure form the main part of the software, while “FREDT2008 Forms” includes the typical main-function of a software project. The part „mainForm“ is the part of the software which generates the simple graphic user interface and also starts „mainCore“, the software part which performs cycle based the task to run the whole program. At the start of the program the „mainCore“ also initialise all the program parts for communication and for the hardware ports. These are the blocks “ccCANMessages”, „ccSerialPort“ and „ccSocketServer”.

The program execution is triggered by a timer so that there is a fix repeat rate of about 40 milliseconds, as long as the processing time for the program code is shorter than this 40 milliseconds. Otherwise the program runs as often as it can.

The sequence during one cycle (shown in the bigger box on the left hand side in figure 2.1) is first to receive the CAN-messages and than to check the state machine. This state machine checks the communication devices for information from the HMI and "decides" which sensors, applications and actors will be needed, initialises them and finally runs them.
In the applications itself, which will be explained in the next chapters, the tasks for the event are programmed.

3 The Navigation Task

3.1 Basic conditions

For the navigation through the rows only the laser-scanner is used, which is mounted in the front of the robot; and of course the electric powered engine generating enough thrust as well as the 4-wheel-steering for accuracy. The scanner delivers the distance of an object and the angle belonging to it; like polar coordinates. The information about the travelled distance since starting is transmitted from the engine.

3.2 Strategy/Algorithm

The basic principle is: Always driving at maximum speed, no unnecessary reducing of speed, except in front and within headland turns.

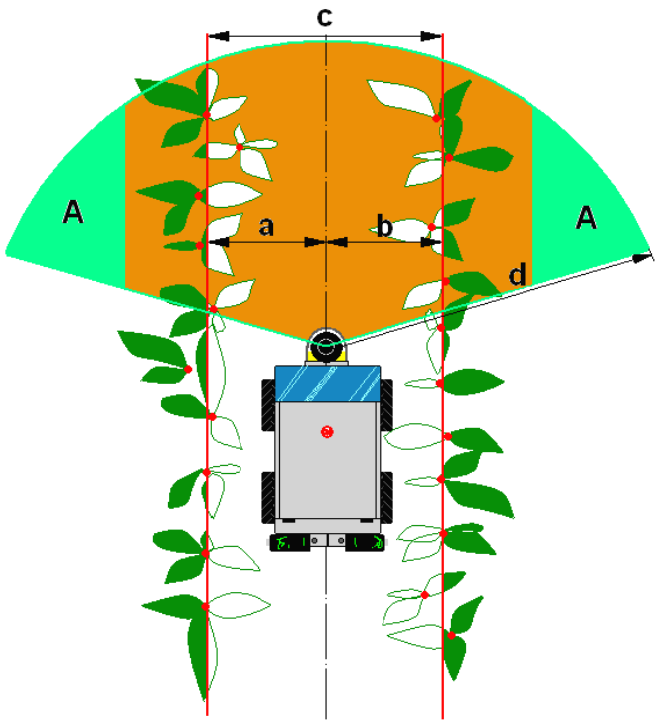


Figure 3.1: Helios finding his way through a row of maize

By an arithmetic averaging of both distances to the rows (see the red lines in figure 3.1) the robot detects the space to both sides (a and b). A determination of the difference is used to generate the required steering-value so that the vehicle recenters itself.

As shown in the drawing, the foresight (d) is strictly limited because otherwise the calculation of the distance-average would loose accuracy concerning the curved rows in the navigation task. The two areas (A) mark the spaces of which the data of the laser-scanner is not considered. In these spaces might exist some plants, leafs, etc. of other rows that must not be considered.

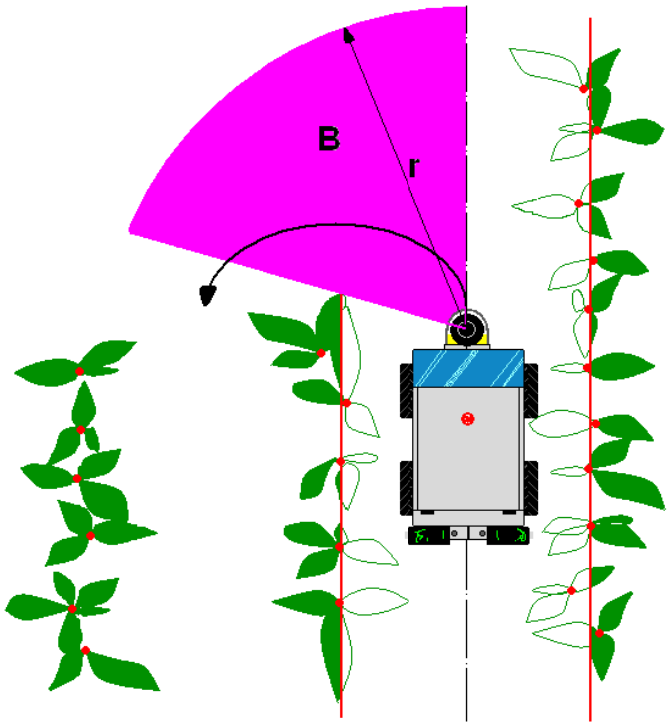


Figure 3.2: Helios detecting the end of the row for turning left

The end of a row is detected if there is no object within a specific scan angle in the close-up range. If this term is true, speed will be reduced continuously until the steering-system starts the headland turn. The system differentiates whether a right- or a left-hand turn has to be made because of the winding end of the rows. The robot finds the way back into the rows if a defined distance was travelled since end-detecting and/or if plants are again recognized in a specific distance in the inward curve. If these conditions are true the vehicle accelerates slowly to make sure the robot has the opportunity of finding the centre-position in any case. The top-speed results from the stability of the closed loop control of the system.

3.3 Problems

In the beginning there had been serious problems with a first algorithm. The idea was searching for gaps in the front. By using certain enquiries the system detected the largest gap, which had to be situated in a defined area in front of the vehicle. To head for the middle of the gap was quite simple and that is how the robot found its way. To make maximum speed a long foresight which unfortunately leads to cutting corners (and maize!) was chosen. Additionally a check-up of the close-up range to avoid the cutting of corners has been implemented. That was complex and somehow not satisfying at all.

An even bigger problem was the missing robustness of the algorithm, when across standing or missing plants had bad influence on the gap's geometry. Very often Helios stopped because of one single leaf detected as an obstacle in the way. The system was still not intelligent enough. This was the main reason to discard the first algorithm.

4 Golf ball detection

4.1 Hardware

The golf balls were detected by a camera. The used camera is the AVT Guppy F-033C, a firewire camera with a resolution of 640x480 px. This camera is mounted behind the windshield. Its line of sight is directed to a mirror, which divides the range of vision in such a way, that the left and right side of the lane can be observed simultaneously.

4.2 Strategy/Algorithm:

The used algorithms of the image processing are quite simple: it only watches for yellow color. The shape of the balls is unconsidered. As programming language C++ and the image processing library OpenCV were used.

The camera generates a color image in the RGB color space. Consequently each pixel contains three color values, one for the red (R), one for the green (G) and one for the blue color (B).

After its generation the image matrix is vertical split into left and right accordingly to the mirror system. Then both image matrices were scanned line by line for the color of the balls. Therefore a range of tolerance is defined for each of the three color values. The number of pixels, which matches with the range, is counted. When this number rises above a predefined value, probably a ball lays in the lane. In fact, the

word “probably” can not be faded out while discussing about this image processing. So this leads to the main problems. The light conditions have a huge influence to the system; sunny weather lets the balls appear brighter, a cloudy sky or the shadow of the plants blacks the balls out. That takes effect on the color values so that at least no ball is detected or rather the processing always detects a ball.
To improve this, the image has to be adapted to the light conditions by the shutter adjustment of the camera. But if the exposure time is to long, because of the motion of the robot the ball will not be imaged sharp, so that the color will be distorted too. So a compromise is needed between exposure time and sharpness of the image.
Summarized there are three parameters (range of the color values, number of pixels with the color value, exposure time), which had to be adjusted by hand in front of each run of the robot. There more automation has to find its way into the processing algorithm. So for the future there is already in planning to consider the shape of the ball and release the implementation of the color searching, because it is difficult to enhance this searching.

5 Plant counting

A mechanically counting system with some whiskers was made from long plastic bars. The whiskers are connected on both sides of the robot with hinge-joints.
A potentiometer was fixed on the end of the hinge shaft. After connecting the resistant to a constant voltage, the delivered voltage over the wiper and one fixed contact is proportional to the bar angle. This signal is connected to an AD-converter of an Atmel microcontroller. The evaluation of the signal is only made on the microcontroller and then the result is written to the CAN Bus to show it on a display.
A great advantage of this mechanical counting system is the possibility to count plants through weed, because the whiskers could press the weed down to detect only the plants.

Field Robot Event 2008

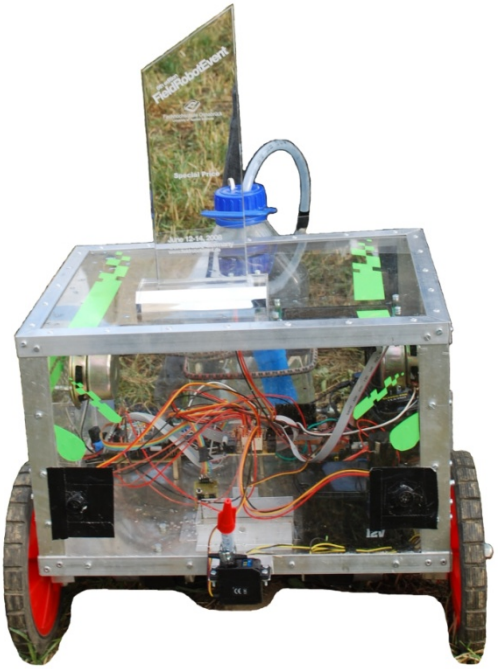
Team Kopi's Farmer

- Documentation -

Team Members: Dominik Lahmann*, Alexander Nedoluschko, Tassilo Tobollik, Malte Vaßholz, Daniel Wilbers

Kopernikus-Gymnasium Rheine, Kopernikusstr. 61, 48429 Rheine, Germany

June 2008



Kopi's Farmer is an autonomous robot, which is designed and developed by a group of five senior secondary school students to entry the Field Robot Event 2008. It is an extreme low-cost robot with an overall project budget of less than 300€. Furthermore planning and building the robot started just one month before the competition.
This paper shows, what can be done with little time and money by giving an overview of the design and control of the robot.

* E-Mail: dominik.lahmann@web.de

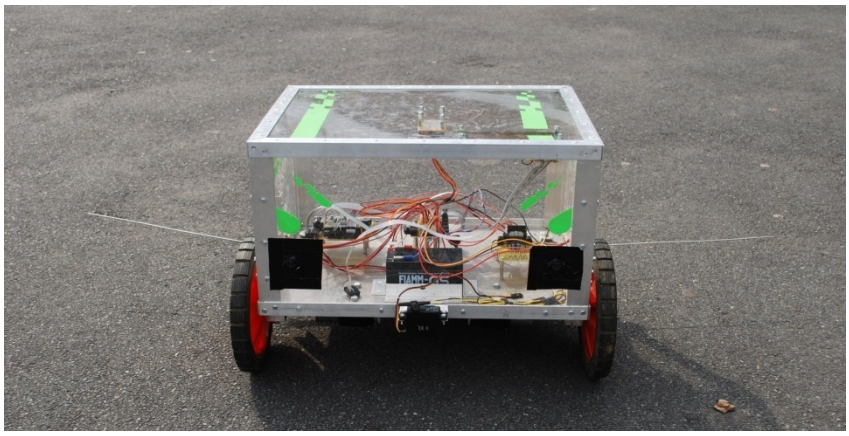
1 Introduction

As a group of five senior secondary school students we saw the Field Robot Event as a possibility to have fun by building a robot for an international competition. Moreover to compete against university teams with expert skills and higher budgets was a special attraction for us. The FRE gave us a great opportunity to pool our knowledge and work on interesting problems.

2 Datasheet

The following datasheet gives a short overview on the robot:

Chassis	W x L x H (in cm)	40 x 30 x 23
	Ground	clearance approx. 6 cm
	Weight	<10 kg
	Model/make	self-made
Drivetrain	Engine	2x 12V DC geared motor
	Conception	electric engine
	Power	12V * 0.7A max.
	Speed	approx. $4^{m}/_{s}$ max.
Control	Microcontroller/PC	Atmel Atmega16
	Interface	Switches, LCD, Leds, RS232, ISP
	Software	mainly C
Sensors	2x SRF02 (Devantech Ltd.)	Ultrasonic Sensor
	CMPS03 (Devantech Ltd.)	Compass Module
	2x GP2D120C	Infrared Sensor
	2x Whiskers	self-made
Total costs		<250€

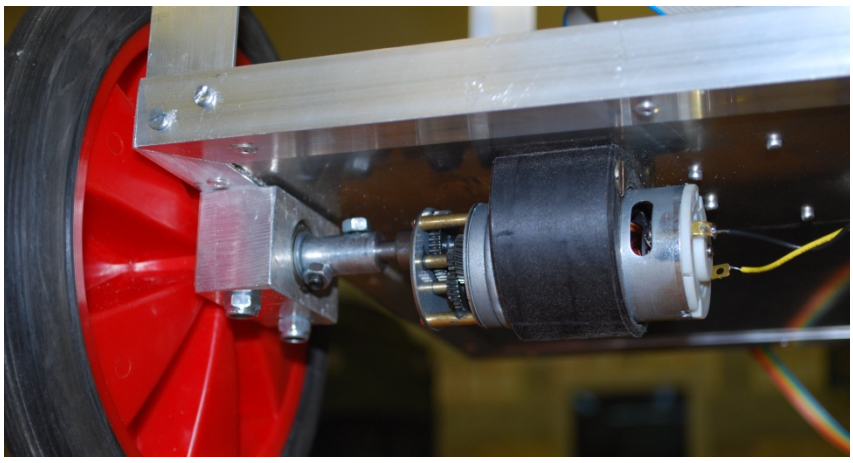


3 Hardware

3.1 Chassis

The chassis had to be a low-cost chassis, which is easy to build and functional. It is composed of a frame construction with aluminum profiles, an aluminum ground plate and plexiglass plates. Thereby the chassis offers enough space and a good flexibility for using different sensors and other mechanical parts. Thus the robot platform can be used for other projects as well.

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



Motor mounted on the robot's ground plate. The wheel is connected to the motor by a ball bearing axis.

3.2 Electronics

Because of the low budget, all PCBs have been layouted and produced by our own. On the one hand, the great advantage of this procedure was that we were very flexible to meet our demands. On the other hand our technical possibilities were

limited (e.g. we could only produce single layer PCBs) and the procedure was extremely time intensive.

3.2.1 Current supply

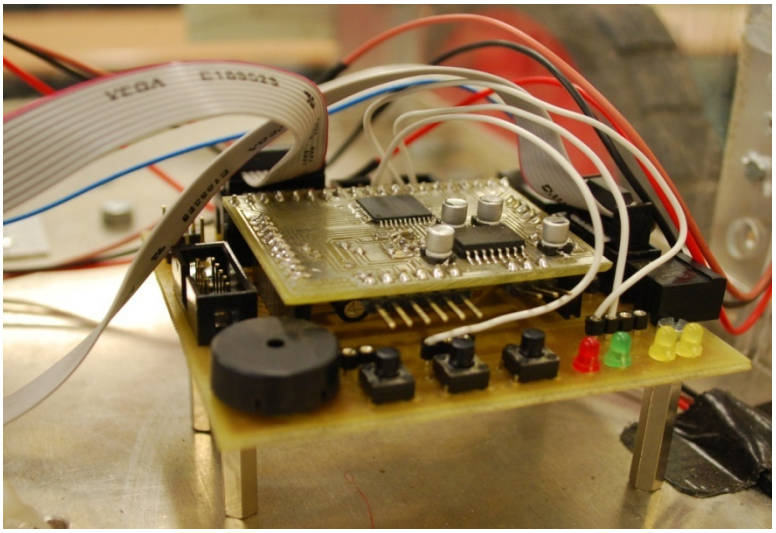
The electric circuits for logic elements and for actuating elements are completely galvanically decoupled. For the supply of electronics a 7.2V Ni-MH accumulator with a capacity of 1100mAh is used. The 12V for the actuating elements is supplied by a 1.5Ah lead acid accumulator.

The accumulators are quite cheap, but the capacities are too low for longer stationary test phases and long drives. Thus in future an external power should be attached.

3.2.2 Microcontroller

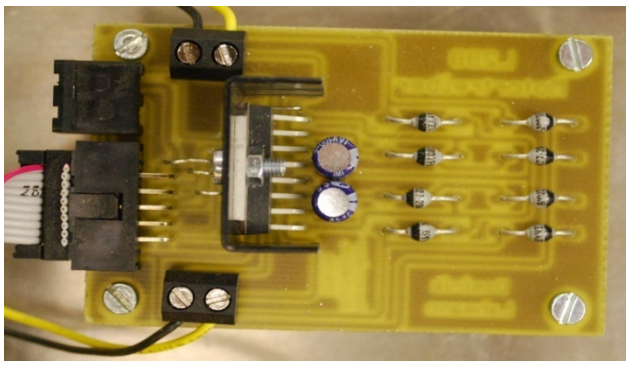
Because of the experiences in our team we decided to use an Atmel controller. Due to the needs an ATmega16 is used, which controls the whole robot by providing two PWM signals for controlling the DC motors together with some logic motor control channels, ADCs for converting sensor data and some channels for user communication (e.g. RS232). Furthermore the microcontroller acts as a bus master on a 400kHZ clocked I²C-bus to communicate with some sensors in order to get sensor data.

The microcontroller is surface mounted on a small PCB, which is directly connected to the main board. The main board provides a DC converter for power supply and connectors to other boards and components.



Controller board with Atmega16 and Max232 level converter slotted in the main board with voltage converter (7805), interface devices and connectors.

3.2.3 Motordriver



L298 motordriver board.

For controlling the 12V DC motors (with 0.7A max.) in speed and rotating direction we designed a motor controller motor with a L293 dual full-bridge motor driver.

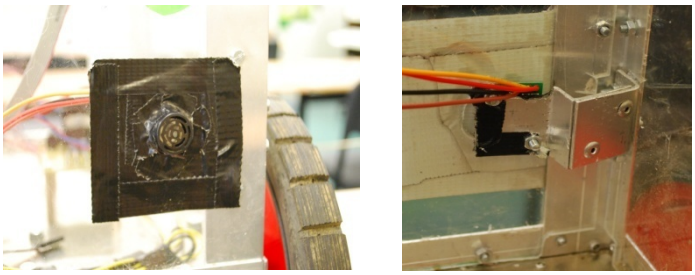
3.2.4 Display

To allow some output by the robot, a 16x2 character display is inserted as a user interface for monitoring different sensor data and information about the running program. It is mainly used for debugging.

3.3 Sensors

3.3.1 Ultrasonic Sensors

At the robot's front two SRF02 ultrasonic sensors from Devantech Ltd. measure the distance to the maize plants and if there still are maize plants or if the end of the row has already been reached. We decided to use that model because of the low price.
US sensors have got the advantage of a big measuring cone and spot size. So they are a good choice to overview the robot's situation.



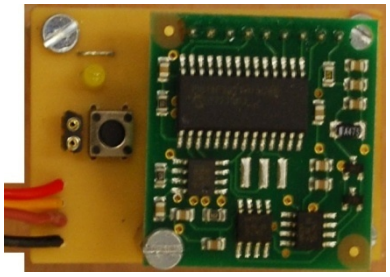
The ultrasonic sensors are mounted with an variable angle.

3.3.2 Infrared Sensors

Infrared sensors are mounted on both sides of the robot. We used the model GP2D120C from Sharp. The analog output signal is converted by a microcontroller's

ADC. The infrared sensors make a point-exact measuring of the distance to the maize rows on both, the right and left, sides of the robot possible.

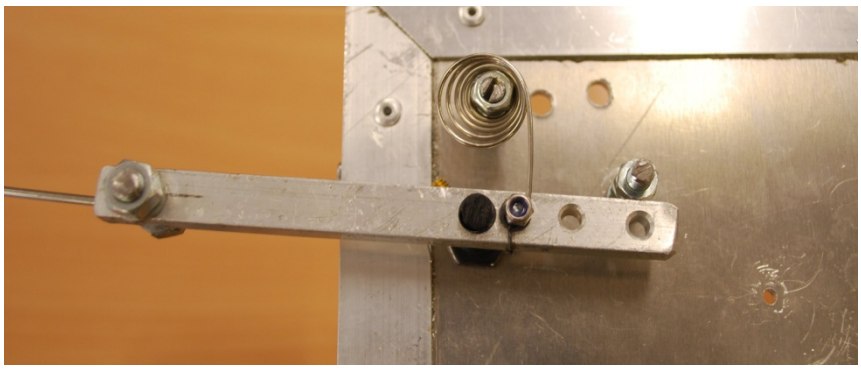
3.3.3 Compass-Modul



On the top of the robot a CMPS03 compass-module from Devantech Ltd. with two KMZ51 sensors measures the direction of the robot. The module is connected to the I²C-Bus. The sensor data varies due to movements around the horizontal plane because of the crumby and bumpy ground conditions. Thus, tolerant thresholds and averaging of the sensor data is necessary.

3.3.4 Whiskers

Whiskers are mounted on both sides of the robot to measure the rough position between the rows of maize plants and to protect it from colliding with them. They consist of metal feelers, which are connected to potentiometers and return springs. Different positions of the whiskers can be digitally measured in form of different resistances of the potentiometers by connecting a voltage and converting the potential drop by the microcontroller's ADC.



Mounting of the whiskers on the robot's ground plate.

3.4 Special Hardware

Because of missing optical sensors, the robot cannot detect golf balls, which represent weed, yet. However we already installed a system of a tank and a pump to spray water (herbicides) as a 'weed-killing' operation. Since the spraying-water mechanism cannot be used in the 'weed'-control task, it is used in the freestyle task to spray fertilizer.

Additionally an audio system consisting of two speakers and a control unit is installed in order to play music to the maize plants and to entertain human beings (e.g. the

farmer). There are some doubtful studies, that analysed plant's reactions on music. So maybe music helps the maize plants to grow faster and to increase yields. Furthermore the integrated sound system should make the robot more practical and lower the inhibition threshold for using robots in agriculture.

4 Software

Most parts of the program code are written in C. The compiler was avr-gcc. Other smaller parts are written in Assembler.

4.1 Program Concept

The general idea of the software concept is to use as many data as possible all the time. Well, overall, just a few sensor data are available at all, which is the data from both IR sensors, both US sensors, both whiskers and the compass module.

The compass value is used for headland turns. Although it is inaccurate when driving on crumby grounds, together with the motor data and by averaging it is possible to navigate with the compass module and the whiskers. Depending on the sense of rotation one of the whiskers still contacts the last maize plant in a row. So referencing to the whisker's angle and the direction given by the compass module it is possible to navigate. Of course, it is a problem when plants are missing at the end of a row. In this case only the compass module and a program with empirically determined parameters is used for a U-turn.
The Infrared sensors are the most important one for driving in a row between maize plants. The last five to ten measured distances within an empirically defined measuring range are saved. Together with the last maximum whisker's angle they give good information about the robot's position in the row.
The US sensors are used in the program to get information about if the robot has already reched the end of a row. If so, a U-turn follows. Also the whiskers are used to see if the robot has reached the row's end. If both whiskers did not get in touch with a maize plant for a longer distance (20cm to 30cm) and the US sensors do not see any plants in the right front or left front of the robot, a U-turn starts.

5 Conclusion

Our concept was not bad and we worked very fast and hard. But the lack of time, which lead to incomplete programs, was a real problem.

For the future there is much potential when having more time and a higher budget. The general concepts, especially the electronic concept, worked well, but the time for testing was too short. Furthermore some more sensor data is needed for a satisfactory sensor concept. Moreover better accumulators, wheels and motors are needed.

Robin

Maik Poggendorf, Patrick Pucholt, Frank Kühnemund, Jakob Jung, Dr.-Ing. Matthias Grimsel
TU Dresden
Department of Mechanical Engineering
Agricultural Systems Engineering
01062 Dresden

Introduction

Robin is our first prototype of an autonomous field robot which features a serial hybrid power train. The aim was to replace the battery-based power supply by a combustion engine that drives an alternator. Even if the contest revealed a couple of weaknesses and problems, the principle itself seems promising and will be pursued in terms of further developments. Another distinct feature of Robin are the four independently steerable wheels. This enables elegant driving maneuvers like turning on the spot.

Mechanics

Wheel Unit

The wheel units of the fieldrobot „Robin“ are special, because the motor, the gear and the sensor for measuring the rotational speed are integrated into each wheel’s rim, as depicted in figure (1).

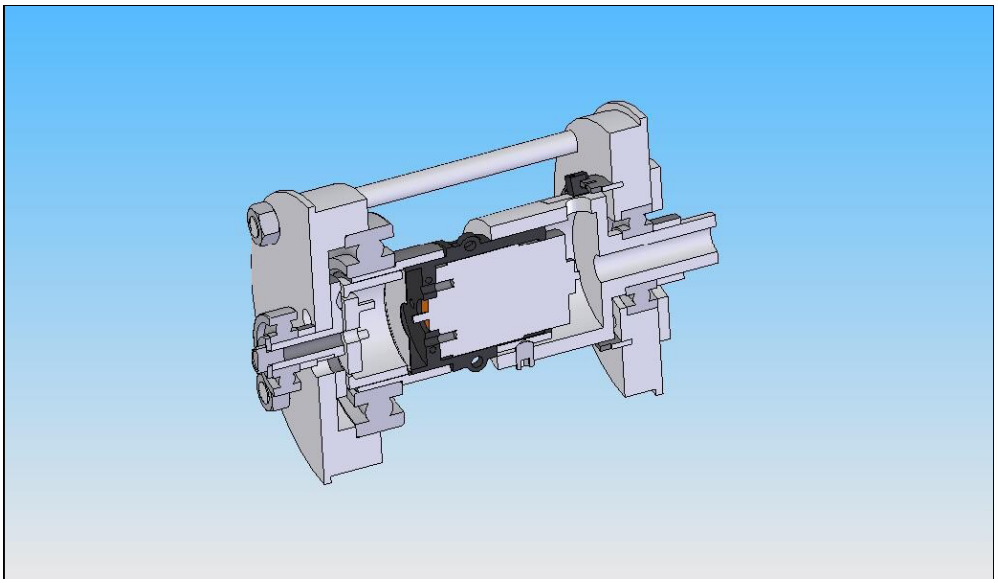


Fig. 1: Section view of the wheel unit

The drive train is actually taken from the cordless screwdriver „Einhell E-ASS 4,8V“ and includes a two step planetary gear with a total gear ratio of 81:1. Since the standard motor of the screwdriver had inappropriate electrical characteristics with respect to the power source we used, we replaced it with the motor „Igarashi N2738-51G-5P“, which runs at a rated voltage of 18 volts. For measuring the wheels' rotary speed we employed the subminiature photointerrupter „Sharp GP1S23“ which is attached at a selfmade alloy wheel. This wheel's circumference is equidistantly grooved by 36 slots of 0,5mm width each. Thus one turn of the wheel produces 36 periods of a square wave. The case of the motor is clamped on a steady hollow axle which takes up the counter torque. This axle is directly connected to the wheel's suspension. On the other side of the wheel housing, the motor axle is connected to the planetary gear whose output stage propels the wheel. The wheel housing consists of a piece of drainpipe, which serves as the rim, and two alloy plates at the sides. The outer plate holds a flange that links the output state of the planetary gear to the housing. The whole housing is waterproof and highly resistant to soil and dust.

Steering Unit

The functioning of a single steering unit resembles the steering of a bicycle. Each unit is situated above its related wheel which is hold by a fork rake. This fork rake is fastened to an axle that is driven by a geared motor. The maximum steering angle is more than 90° to both directions. This decentralised way of working enables various driving maneuvers as indicated in figure (2).

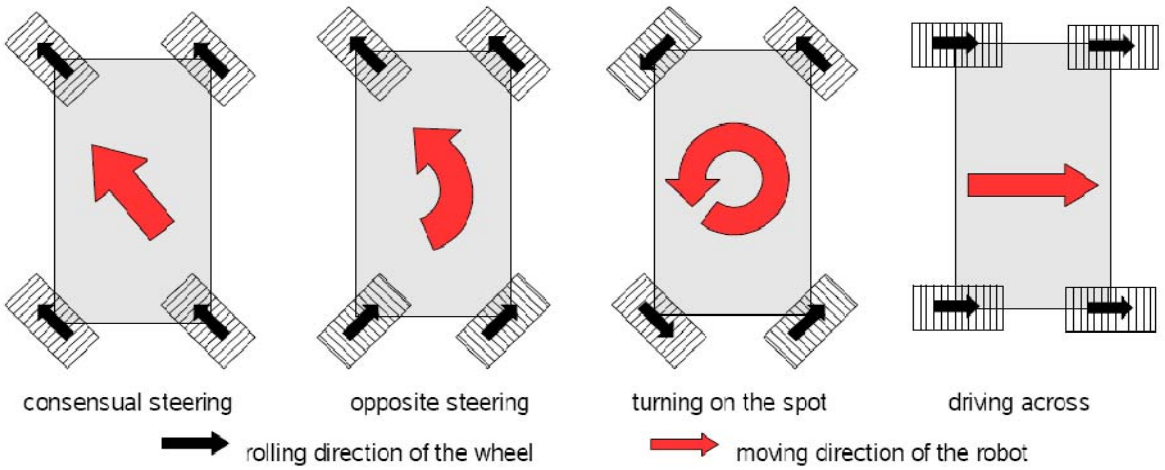


Fig. 2: Some possible steering maneuvers

The steering drive is actually intended for shutters in car air conditioning systems. The original motor was not powerful enough to meet our requirements, thus we installed the „Igarashi N2738-125“ motor whereby the housing had to be slightly modified. The resulting drive has a gear ratio of 532:1 and provides a slow rotational

speed of about 5rpm and a high torque of 2Nm at a voltage of 12V. The gear also includes a worm that holds the wheel in position when there is no steering action. This solution also has a significant downside. When hitting heavy obstacles there is no flexible part that could cushion the strokes. Moreover, the gears are made of plastic. That is why we lost three gears during test period. The measurement of the steering angle is performed by a potentiometer that is directly mounted on the steering axle. In case of a processor breakdown or software error the steering process is terminated by end switches. These switches are triggered whenever the steering angle exceeds a certain value and interrupt the electric circuit of the related steering motor.

Chassis

The chassis is made of alloy profile rails and plexiglas. The central box is divided into two sections, as shown in figure (3). The front section provides space for the electronics while the rear section contains the generator unit.

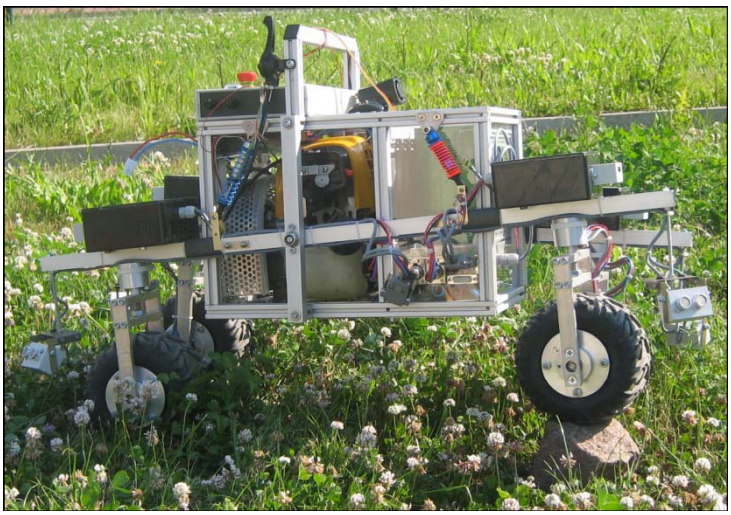


Fig. 3: The central box borne by two swing arms

The generator consists of a combustion engine that turns the armature of an alternator. This process causes noise and vibrations at a frequency corresponding to the motor speed of rotation. In order to reduce the resonance the generator unit is suspended on rubber buffers. The central box is linked to two swing arms at its centre of gravity. This connection brings a rotational degree of freedom in order to provide kinematic flexibility and is stabilised by springs and dampers. The assembly of the single components forms a mass-spring-damper system with a certain inertia, damping ratio and stiffness. The swing arms serve as a connection line between the central box and the wheels and steering units. Since the lever to the front side is longer than the one to the back side the front spring at each swing arm is softer than the back spring.

Generator Unit and Power Management

The generator unit comprises a 4-cycle combustion engine, a motorcycle alternator and a power controller. Figure (4) shows the engine and the alternator within a test set-up.

Fig.4: Testing the generator

The engine with the product name „Subaru Robin EH025 Micro“ is taken from a „Subaru PTV101“ water pump. It is capable of supplying a continuous output power of 550 watt (0,75HP) and a maximum output power of 810 watt (1,1HP). The alternator and the power controller (SH579A-12) are parts of the motorcycle „Kawasaki ZX-9R“. The alternator has a three phase output and is permanently excited. Its output voltage is a function of the armature speed of rotation and the current that is drawn by the load and can reach values up to 250 volts. Obviously, this raw output is not usable for subsequent loads. It compels the usage of a power controller. It took us two weeks to find an appropriate controller and to connect it properly to the alternator. When charged with a continuous load up to 180 watt the applied controller puts out 14,6 volt. At a charge of 200 watt the voltage diminishes to 12 volt, at 250 watt it is about 5 volt. For testing in closed rooms or at night (the noise of the combustion engine is considerable) it is possible to drive Robin without generator. The power management features an interface to an external power supply unit. An important role in this chain of power conversion plays the capacitance that stabilises the output voltage of the controller. It should be as high as possible and, simultaneously, must be able to endure high currents. It was the dominating weak point with respect to Robins performance during the contest. When accelerating from standstill, the current drawn by the electrical drives overcharged the controller. But also the fact that eight electrical drives have to be supplied by one power source turned out to be a problem, expecially when Robin had to drive uphill. Consequently, the controller output voltage broke down for several times. Eventually, we used

batteries in parallel in order to alleviate the power shortage. Thus, the first aim to achieve with regard to next year's competition is the design of a more powerful generator unit.

Electronics

Hardware Layout

Our main goal concerning the hardware layout of our electronics was to have an plug-and-play environment that allows the team members to develop their task independently. To achieve this, we mainly used standard components for 19" racks, such as the connectors, module and guide rails and mounted it inside a frame of aluminium profiles that made part of our chassis (Fig. 5).

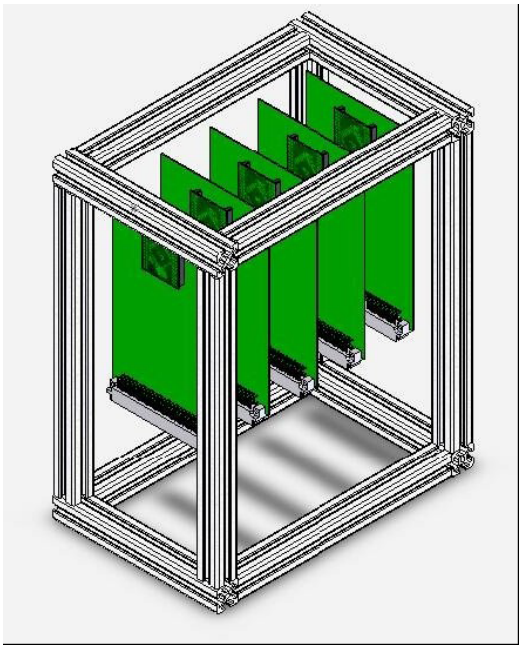


Fig. 5: Basic sketch of the electronic box

Heart of our wire maze is the main backplane, that distributes all the signals from the outside world as well as from the power electronics inside the box to the connectors of the plug-in boards. This means, that neither the backplane, nor any of our wires nor the power electronic – once mounted – needs to be touched anymore. The power electronics itself is mainly mounted on a second layer beneath the main backplane. Only those parts concerning logic and intelligence are located on the plug-in boards. There they are easily accessible, can be taken out for programming, can even be tested "out of the box" and there is only one connector for each board, so that it is nearly failsafe. In total we have room for five plug-in boards, however this year we only used three of them:

- Drivecontrol: ATmega128 dealing with the signals of the in-wheel incremental encoders and controlling the power electronics for the drives.
- Steercontrol: 4 relays for the steering motors together with the required electronics.
- Sensor & Masterboard: 2 x ATmega16 for processing ultrasonic as well as IR input

Principles of Measurement

Velocity

For the measurement of the rotational speed of the wheels we used in wheel incremental encoders based on a slotted aluminium ring and a photointerruptor.

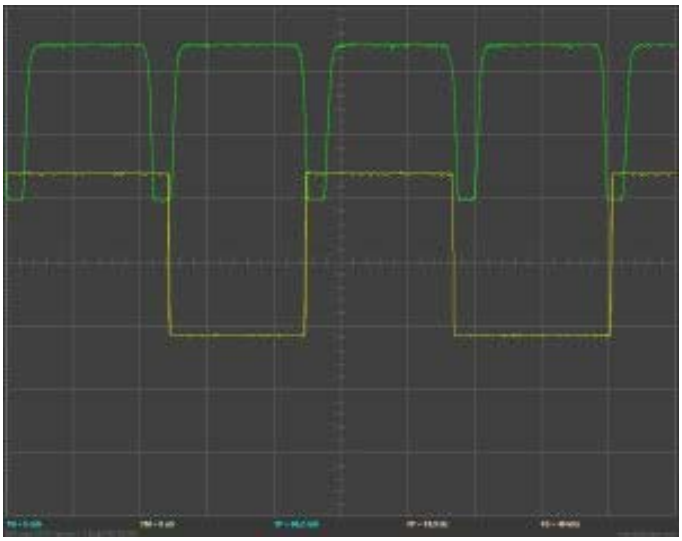


Fig. 6: Signal of the photointerruptor, edges detected by µC

Fig. 6 shows the signal, we obtained from the photointerruptors (green), as well as the edge detection done by the µC (toggle of the yellow signal). The expired time between two edges was measured and gradually averaged over 3 measurements.

Steering Angle

For the measurement of the wheel angles we used potentiometers. They were connected to the ADC
of the µC and powered by our 5 V main voltage reference.

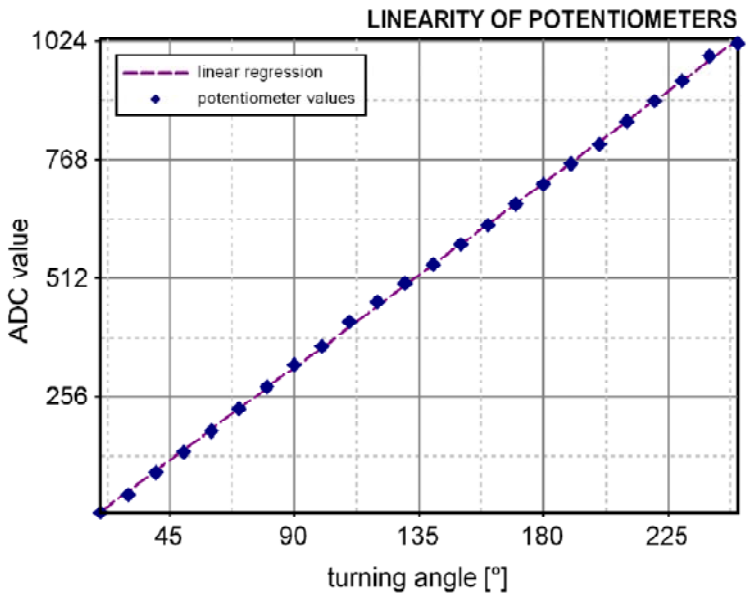


Fig. 7: Linearity of the used potentiometers

Bus & communication

Once we had discovered that we will not get our AT32AP7000 based master board running as a central controlling unit allowing us to access all the other boards via RS-485, we had to find a more simple yet just as beautiful solution. Finally we decided to give the tasks to one of the ATmega16 on the “Sensor & Control” board. This one, already talking I²C with the ultrasonic sensors, could also assume all the other communication via I²C. Even if not that powerful, the I²C bus was sufficient for all the communication needed for basic strategies. It worked very stable and because of our modular design it will be the decision of next years team whether to continue working with I²C or to use any other bus.

Strategies

Steering within the Row

According to our wish to keep everything as robust and simple as possible we eventually agreed on the following steering strategy:

Independently from each other the front as well as the back ultrasonic sensors compare the distance to the right and the left. Any difference of these distances will immediately cause a steering action of the corresponding wheels, so that both, the front and the back axis always try to keep in the middle of the row by bringing the difference of distances to zero. Thus we were able to control the orientation of the robot in the row without even using any real information about its orientation in reference to its surroundings. In case of missing plants, i.e. when the sensors did not

“see” anything within a defined distance, we replaced the sensor input by an “artificial wall” in a distance that made our row following algorithm remain stable until we detected the next plant. Difficulties we faced were mainly connected with finding the right parameters for the amplification (how big a steering angle α shall be caused by a difference of distances x) and the absolute maximum steering angle, so that the system remained stable. One problem, we were not able to solve, was the relatively slow turning velocity of the steering, which made it necessary to limit the maximum driving velocity. Next years team should consider faster or stronger steering motors.

End of Row Detection

In case that neither of the front ultrasonic sensor did detect plants anymore, the IR came into play, zeroing a distance measurement every time they detected an object in short distance. The distance measurement reaching a value of 30 cm or more indicated that the end of the row is reached.

Headland Turn

Since we had no compass, gyroscope or similar devices for detecting the turning angle of our robot, we had to rely totally on the odometry. Therefore we placed value on having reliable information about our wheel units at all times. Owed to the wide range of the wheels’ steering angle Robin was able to follow an circular arc with a small radius. In conjunction with the measured distance we could determine the driven angle on that arc. So far we did the headland turn in three steps: First we stopped, after having detected the end of the row and positioned our wheels in the correct angle, then we drove the defined distance $r\pi$, stopped again and turned the wheels back before driving into the next row. This worked out very well as long, as we did not have to deal with ground slip. We could massively increase the reliability of the incremental encoders, when using all four of them, not only for speed control (which we did), but also for odometry in those moments when high accuracy is needed.

Weed Detection

For weed detection we used the CMUCam2 with an sx52 microcontroller and the Omnivision OV7620 CMOS camera. The Advantage of this compact module is the build-in one pass tracking algorithm.

Via RS-232 it is possible to receive the coordinates of a bounding box which is supposed to be an area of the designed tracking color.

The control of the SX52 and the interpretation of the received T-Packets (tracking information) is carried out by an ATmega128 featuring RS-485 and RS-232

interfaces. As mentioned above, we hit problems implementing a RS-485 bus-system based on the AT32AP7000 master board. Therefore
the camera system worked as a standalone on the Atmega128. In order to observe the communication between Camera and Atmega and to control the camera without the ATmega we bypassed the rs-232 and connected a Bluetooth RS-232 Adapter (Pico Plug) to it.

You have to deal with some limitations of the camera and the embedded tracking algorithm.

First of all, if you want to track a range of RGB values, you have to limit these ranges to avoid too much interference by objects with nearly the same color. As you can see, if the range is too low you don't detect all yellow golf balls (figure 8).

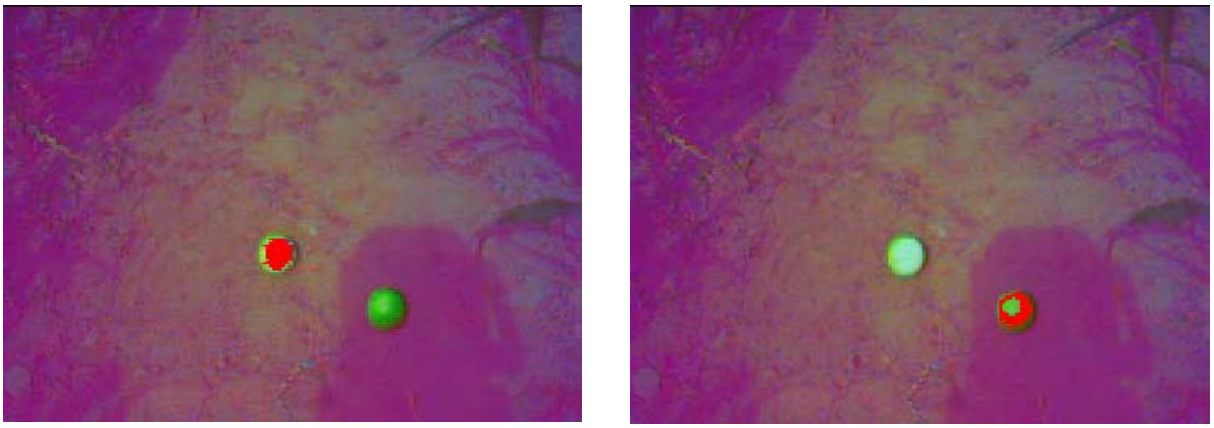


Fig.8 : Only the red-marked ball is detected

If the range is too high you get a lot of interference (figure 9).

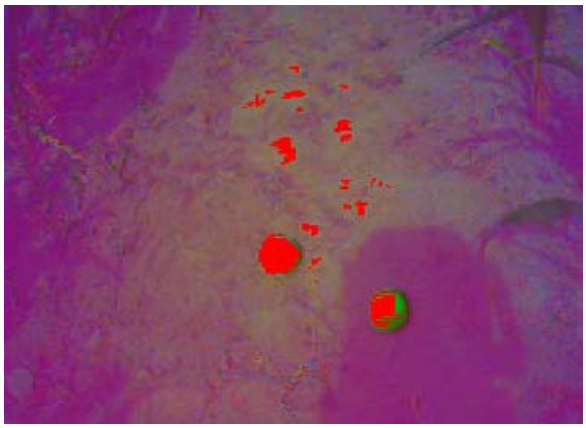


Fig.9: Interference due to high range values

To receive usable tracking information it is necessary to avoid changing brightness condition and use strong contrast objects to track. Secondly, the camera has two

resolution modes. We decided to use the high resolution mode because of the more accurate tracking information. As a result the framerate drops to 11 fps, which is a problem that will be discussed later. To get the optimum results we adjusted the camera to

- YBrCr mode
- Autocontrast
- Autobrightness
- high resolution mode

Thirdly, the camera tracking algorithm is able to track just one object. If a second comes in the field of view the bounding box is scaled to both (figure 10). The only way to separate the big bounding box is to use a camera embedded function called “virtual window”.

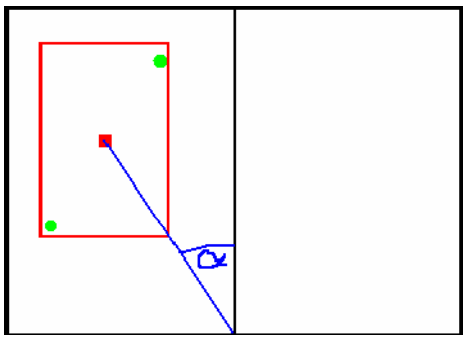


Fig.10: Bounding box including two objects

With the command VW x1 y1 x2 y2 you are able to track a colour in an sub-window. To search in another area you have to set the vw again. As I told, the frame or the number of tracking information per second is limited in High Resolution mode to eleven fps. So imagine a system with two sub-frames dividing the full area of view. A sequence of
VW -> ACK -> TC -> T-packet -> ACK
needs 160ms. The consequent is that the original 11fps are reduced by the transition overhead and time the camera need for mode-switching to 6fps.

To reveal, if just 3fps are received per side, it is not possible to smooth the values, compensating abrasion etc. .The positive thing is, that it is not necessary. If the robot moves 1m/s then the distance the roboter has moved between two received t-packets is 33cm. Depending on the camera position the backfill of the golf ball is far too big. As a conclusion, it makes no sense to use more than two sub-frames.
For the special on-field application between two maize rows it is practically to use a fixed two sub- frame system divided in the middle of view (figure 11). This causes no

problems as long as the angle of the camera is not too high, that means that the camera should not look so far to the front.

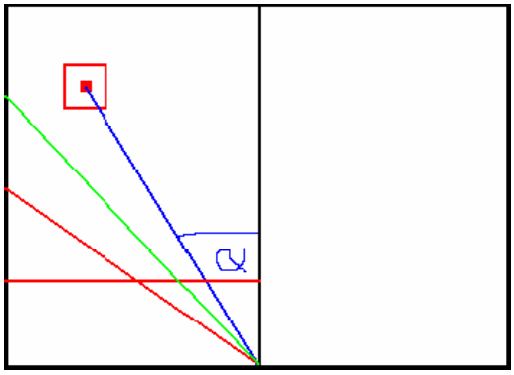


Fig. 11: Fixed two sub-frame system

To decide precisely whether the robot passes a golf ball, a decision algorithm is implemented in the ATmega128. The basic principal is based on the angle to the middle of the mass of the detected area. If this angle is bigger than the threshold (red) or the y-coordinate is bigger then a threshold (red in figure 11), an event is initiated. To avoid that an already detected golf ball is recognized twice you have to block these sub-frame, that means that the golf ball angle bigger than the threshold can't initiate the event. The release is caused by the next golf ball that comes into the field of the sub-frame from above, passes the green angle, and for the case of failure by a delay. In case that two golfballs are far away in the field of the sub-frame the t-package give the centre of masses back. So if the lower golf ball leaves the field of view, the centre of masses jumps to the higher golf ball and above the release threshold. In case that two or more golf balls are close together, the t-package gives the number of pixels back. This value is an indicator for more than one ball. Logically one ball has less pixels of the tracked colour than two. So if the bounding box has too many pixels there more than one golf ball. As I told, if a golf ball is detected an event is initiated and buffered. For every buffered event the ATmega128 gives a 50ms 5V high peak via a logic port out to the ATmega16, which does the navigation and has the incremental distance information to start the "weed demolishment". With this algorithm it's possible detecting multiple golf balls on both side under restriction of the CMUCam2.

Considerations on the Steer Control

This section investigates some aspects related to the design of an appropriate steer control and is especially dedicated to people who plan to participate the FieldRobotEvent in near future.

The basic task the robot has to perform is generally following a center line that bisects two limitating curves. That problem encloses a couple of issues that should be considered painstakenly. First of all, the information the robot needs in order for it to be able to navigate between the curves needs to be decided. Ideally, the robot drives along the center line. That includes firstly the same space either side of the robot, and secondly the robot being orientated tangential to the center line. When leaving this ideal pose, it has to be detected as to which kind of deviation has happened. Apparently, both measuring the position and orientation can be carried out by using distance sensors. The next question is how to arrange those sensors in a reasonable manner. A sensing of orientation angles requires at least two sensors per side. In general, since the overall space between the limitating curves is unknown, it is advantageous to place sensors on both sides of the robot. Altogether it seems plausible to attach four distance sensors, one to each edge. On the one hand, additional sensors provide redundancy. On the other hand, especially when using ultra-sonic sensors, they might cause interference problems. Figure (12) shows the four-edge sensor arrangement. Based on this the governing equations of the steering control path will be derived.

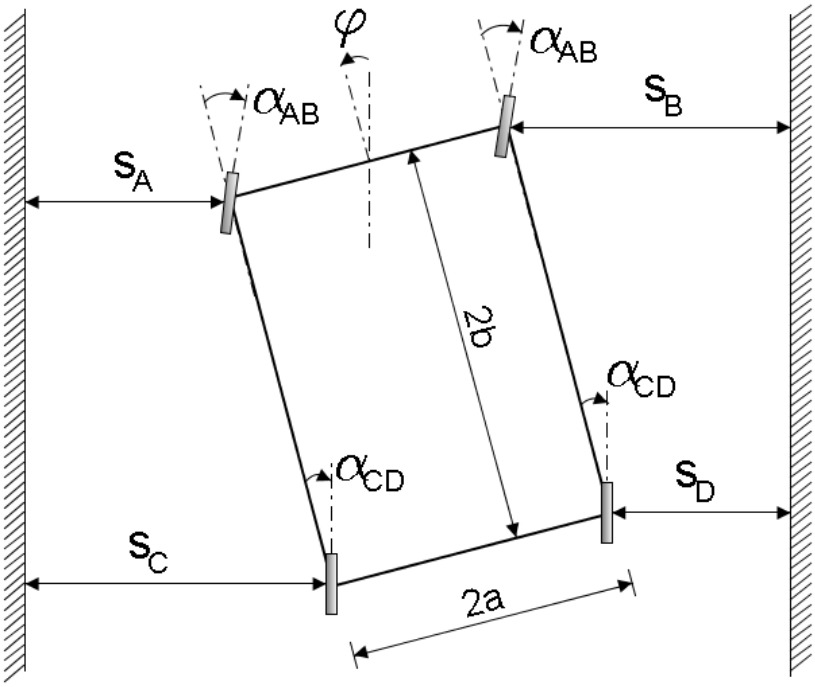


Fig. 12: Robot in between limitating curves

We propose that the robot will have four wheels which are driven at the same angular speed ω. That simplicity is tolerable if the robot's path is only slightly curved, as it is provided by the course of the maize rows. Note that turning or passing sharp curves requires the Ackermann-steering, which is described in section (7). The wheels'

radius R combined with ω leads to the circumferential speed $v = R\omega$. The angles α_{AB} and α_{CD} represent the wheels' orientation relative to the robot's lengthwise axis. The distance between back and front wheels is $2b$, the robot's breadth is $2a$. By means of the four range values $[s_A, s_B, s_C, s_D]$ both the robot's position and orientation with respect to the center line can be figured out. Finding the right path turns out to be a two-dimensional control problem. In order to cancel out the effect of a varying overall space between the limitating curves we define the two differences

$$\Delta s_{AB} = s_A - s_B$$

$$\Delta s_{CD} = s_C - s_D$$

as state variables. Each wheel propels into a direction which is perpendicular to the wheel's rotation axis. That basic constraint leads to the equation system

$$\Delta \dot{s}_{AB} = -2v\sin(\varphi - \alpha_{AB})$$
$$\Delta \dot{s}_{CD} = -2v\sin(\varphi - \alpha_{CD})$$
$$\sin(\varphi) = \frac{\Delta s_{CD} - \Delta s_{AB}}{4b}.$$

By making use of linearization we can derive the state space representation of the control problem which takes on the form

$$\begin{pmatrix} \Delta \dot{s}_{AB} \\ \Delta \dot{s}_{CD} \end{pmatrix} = \frac{v}{2b}\begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \Delta s_{AB} \\ \Delta s_{CD} \end{pmatrix} + 2v\begin{pmatrix} \alpha_{AB} \\ \alpha_{CD} \end{pmatrix}. \qquad (6.1)$$

We see that both steering angles α_{AB} and α_{CD} in (6.1) appear as the actuating variables. An appropriate method of closing the control loop is by implementing a full state feedback. Thereby the system becomes autonomously and will constantly try to zero its states as long as all roots of the system's characteristic polynomial have a negative real part.

If we employ a p-algorithm by assigning

$$\begin{aligned} \alpha_{AB} &= -k_\alpha \Delta s_{AB} \\ \alpha_{CD} &= -k_\alpha \Delta s_{CD} \end{aligned} \qquad (6.2)$$

and substituting into the state space equations above, the stability of the control system can be easily proved by determining the roots of the characteristic polynomial which is actually given by

$$\det\begin{pmatrix} s-\frac{v}{2b}+2vk_\alpha & \frac{v}{2b} \\ -\frac{v}{2b} & s+\frac{v}{2b}+2vk_\alpha \end{pmatrix} = s^2+4vk_\alpha s+\left(2vk_\alpha\right)^2 \qquad (6.3)$$

It follows that there is a double root

$$s_{1,2} = -2vk_\alpha$$

which indicates the system's stability. Note that in this context s stands for the complex variable in frequency domain. Unfortunately, in practice we cannot reach such an ideal control behaviour as it is expressed by equations (6.2). First of all, we have to regard that calculations are processed by a digital control unit which needs a certain time for computation. In general this delay is negligible compared to the delay associated with the gathering of distance information. When using ultra-sonic sensors or a laser-scanner, each of the signals Δs_{AB} and Δs_{CD} appears as a series of numbers which is updated at a rate of a couple of milliseconds. During two samples the controller has no information on the distances and consequently runs at idle. Another turn of the screw is the massive noise which is superimposed to the distance signals. This is because the sensors see some of the 8cm wide gaps between the maize plants. Additionally, there are leafs hanging in front of the plants pretending a shorter distance to the maize row. For that reason it is essential that any filter algorithm is applied to the measurements before those signals are passed to the controller. An effective way of filtering is to calculate the arithmetic average progressively. Figure (13) depicts the basic steps this method consists of.

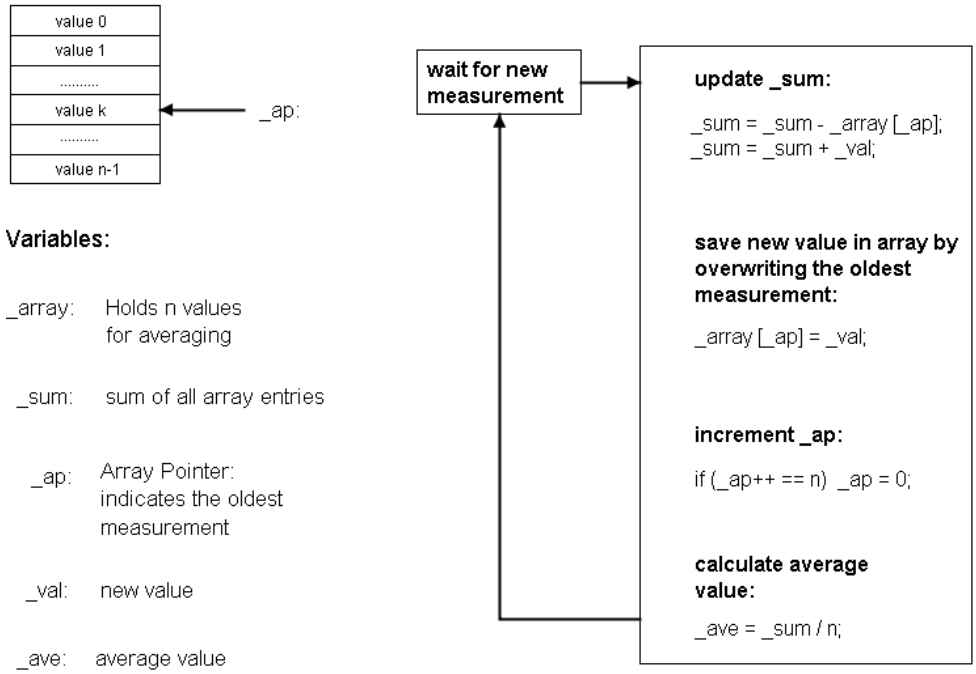


Fig.13: Progressive averaging

As one can infer from figure (14), the process of filtering also has a downside since it takes the controller more time to react on changes in distances. Therefore you have to make a sensible compromise. On the one hand, the distance signals become the smoother the more values you take into account for averaging. On the other hand, a higher number of averaged measurements makes the signal become more idle.

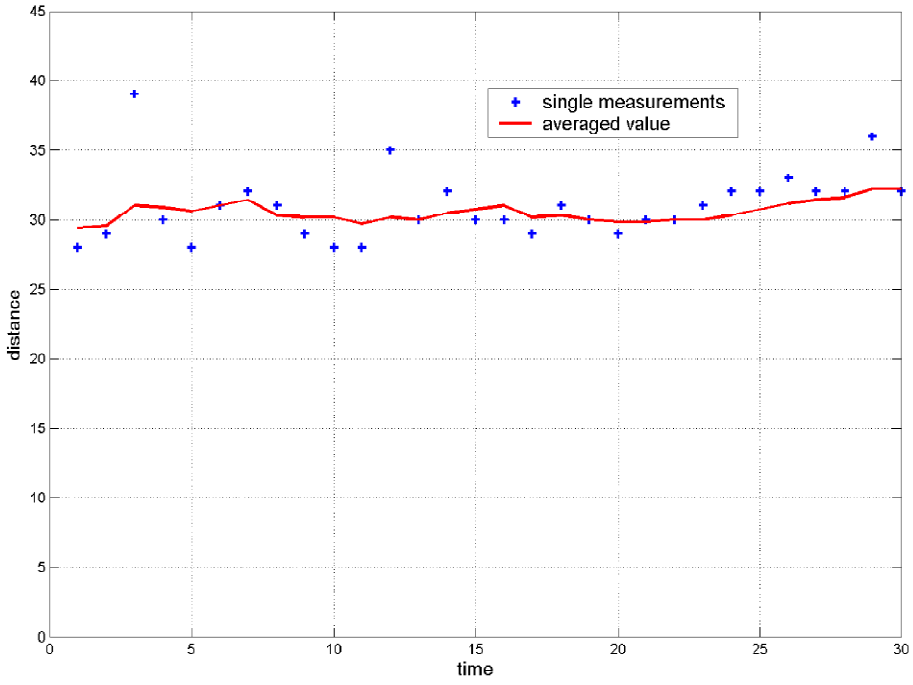


Fig.14: Effect of averaging over five measurements

After the distance is recognised and the computations completed the steering hardware is on duty to put the desired angles α_{AB} and α_{CD} into practice. A coarse terrain combined with the robot's weight can lead to appreciable friction torques on the shaft of the steering drive. The hardware has to be capable of overcoming these additional loads while still generating a high steering velocity which actually represents the time derivative of the steering angle. Consequently, also the steering process itself is delayed. In contrast to equations (6.2), which describe the control algorithm, we try to model the system's behaviour as to the closed control loop regarding the delays. To comprehend all the mentioned delay sources in a compact way we do the redefinitions

$$\alpha_{AB} = -k_\alpha \frac{1}{1+T_\alpha s} e^{-sT_t} \Delta s_{AB}$$
$$\alpha_{CD} = -k_\alpha \frac{1}{1+T_\alpha s} e^{-sT_t} \Delta s_{CD}$$

including a dead time T_t and a delay time T_α by using transfer functions. Partially, T_α corresponds to the time the wheels need to turn before reaching a certain steering angle. Moreover, it is effected by the low-pass behaviour of the filter applied to the measurements. Similarly, the dead time T_t can be traced to the sensor measurement period. The resulting equation system is fed into a real-time simulator for demonstrating the effect of the distinct parameters k_α, T_α and T_t. Note that the steering behaviour is adjusted by changing k_α.

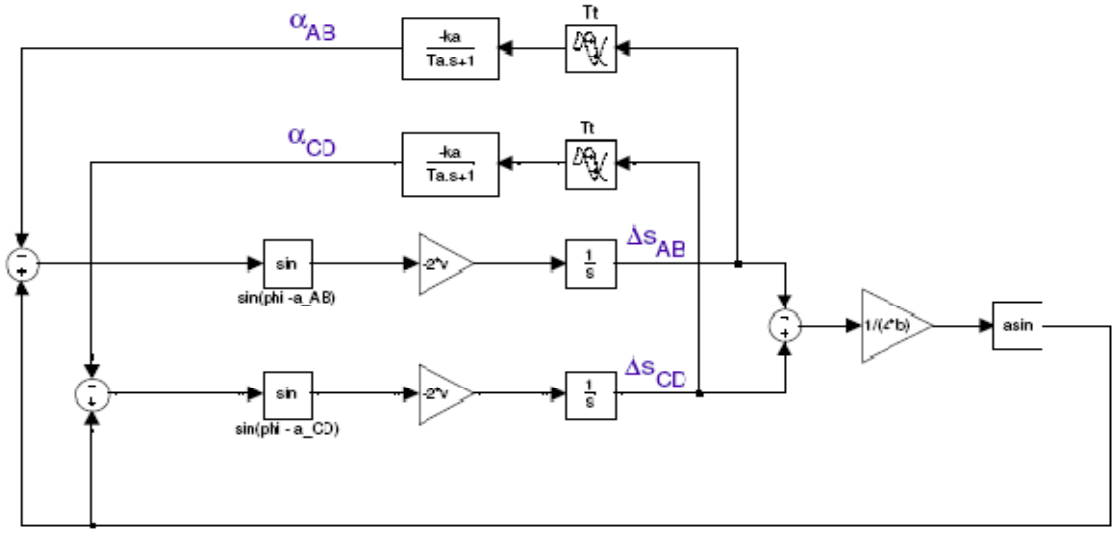


Fig.15: Simulink model

We fix the parameters

$[a \quad b \quad v] = [0.25m \quad 0.4m \quad 0.5m/s]$

$[T_t \quad T_\alpha] = [0.2s \quad 1s]$

and run the simulation for different gains k_α assuming an initial displacement of

$\Delta s_{AB0} = -0.2m$

$\Delta s_{CD0} = 0.$

The controller will try to sort out this displacement by manipulating the steering angles.

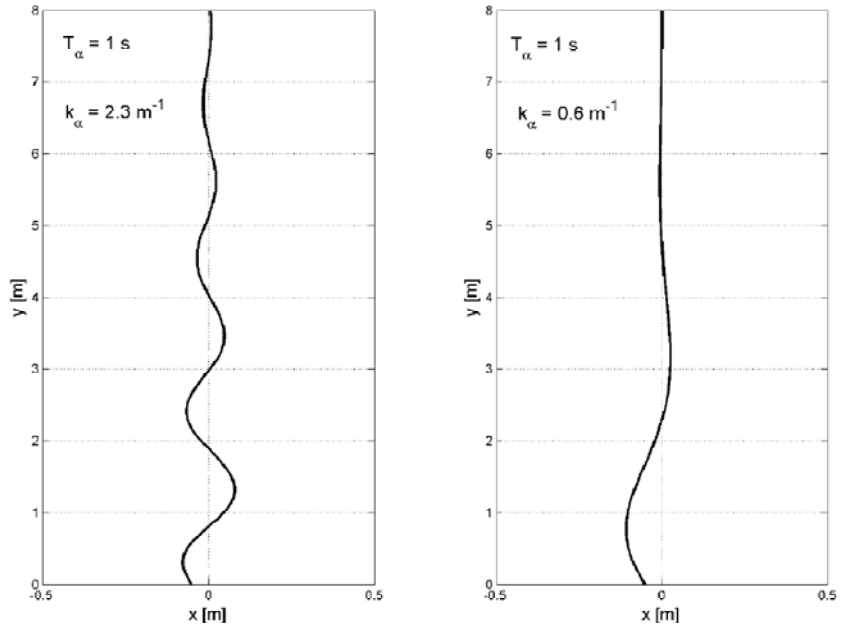


Fig.16: Path of the robot's center point for different k_α

Apparently, a higher gain k_α makes the robot react faster but causes more overshoot. The usefulness of the model depends on how accurate you match the parameters. Nevertheless, the order of magnitude of k_α is indeed revealed. Next we compare two different steering drives which differ from each other in their delay time. The simulation is run with the gain $k_\alpha = 0.6m^{-1}$.

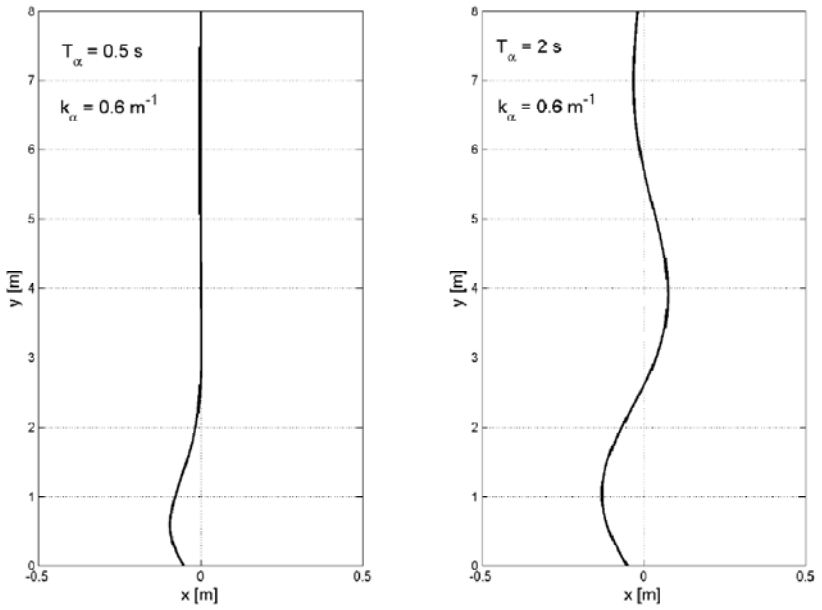


Fig.17: Comparing a fast steering drive with a slow one

At last, figure (18) reveals the great advantage of using four wheels for steering rather than two.

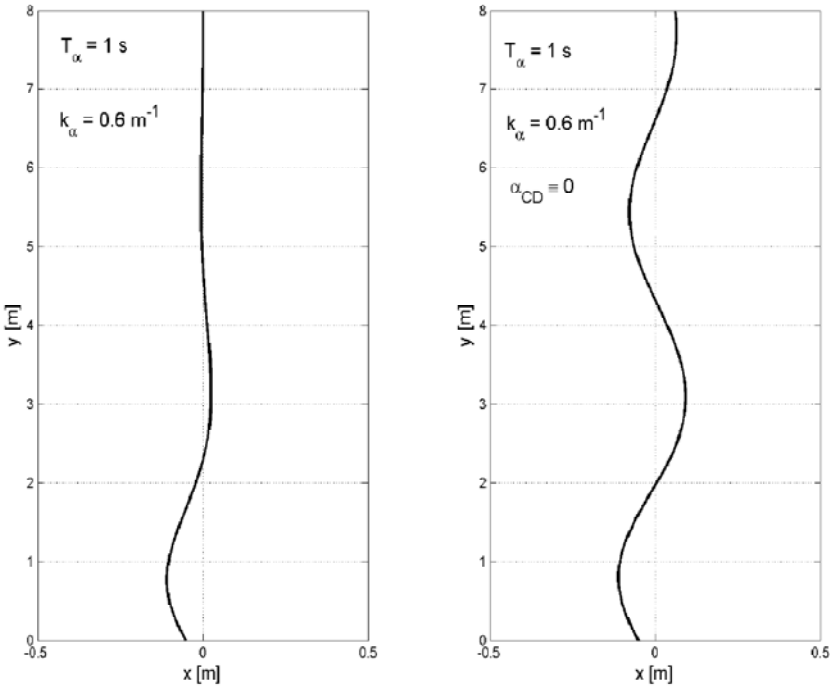


Fig.18: Effect of setting the rear steering angle α_{CD} to zero

Ackermann Steering

This sheet provides the basic formulas related to Ackermann Steering.

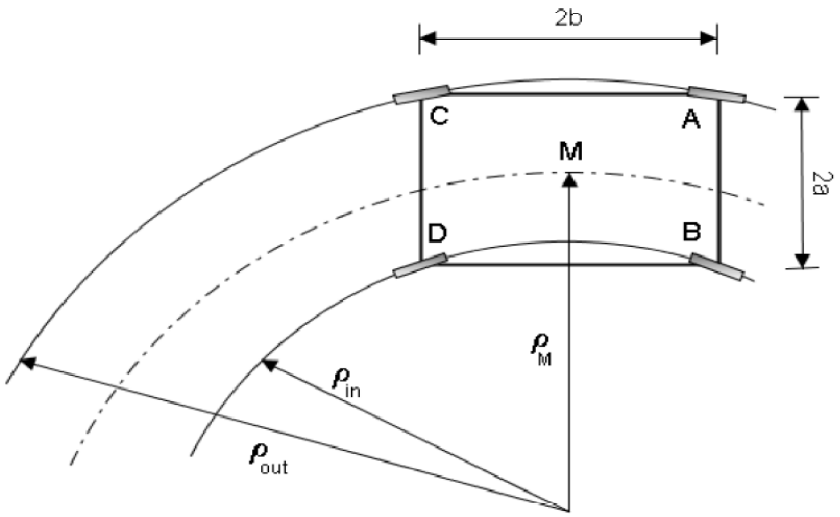


Fig.19: Robot following a circular arc

Inner and outer radius as a function ρ_M . M is the robot's center point:

$$\rho_{in} = \sqrt{(\rho_M - a)^2 + b^2}$$
$$\rho_{out} = \sqrt{(\rho_M + a)^2 + b^2}$$

Steering angles with respect to the robot's lengthwise axis, defined in clockwise direction (see fig. 12):

$$\alpha_A = \arctan\left(\frac{b}{\rho_M + a}\right)$$
$$\alpha_B = \arctan\left(\frac{b}{\rho_M - a}\right)$$
$$\alpha_C = -\arctan\left(\frac{b}{\rho_M + a}\right)$$
$$\alpha_D = -\arctan\left(\frac{b}{\rho_M - a}\right)$$

Inner and outer velocity as a function of v_M :

$$v_{in} = v_M \frac{\rho_{in}}{\rho_M}$$
$$v_{out} = v_M \frac{\rho_{out}}{\rho_M}$$

Distance covered by the inner and outer wheels when driving a section $\Delta\varphi$ of a circular arc:

$$d_{in} = \Delta\varphi\sqrt{(\rho_M - a)^2 + b^2}$$
$$d_{out} = \Delta\varphi\sqrt{(\rho_M + a)^2 + b^2}$$

Acknowledgments

The whole team likes to thanks it's sponsors:

1. GWT – TUD GmbH

2. Robin Europe GmbH

Moreover the team thanks Prof. Thomas Herlitzius, Prof. Gerd Bernhardt, the staff of the agricultural
technology workshop at TU-Dresden, Willibald Kühnemund and especially Dr. Matthias Grimsel for their support.