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

Contents

1. 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.

2. Hardware

2.1. 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.

2.2. 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.

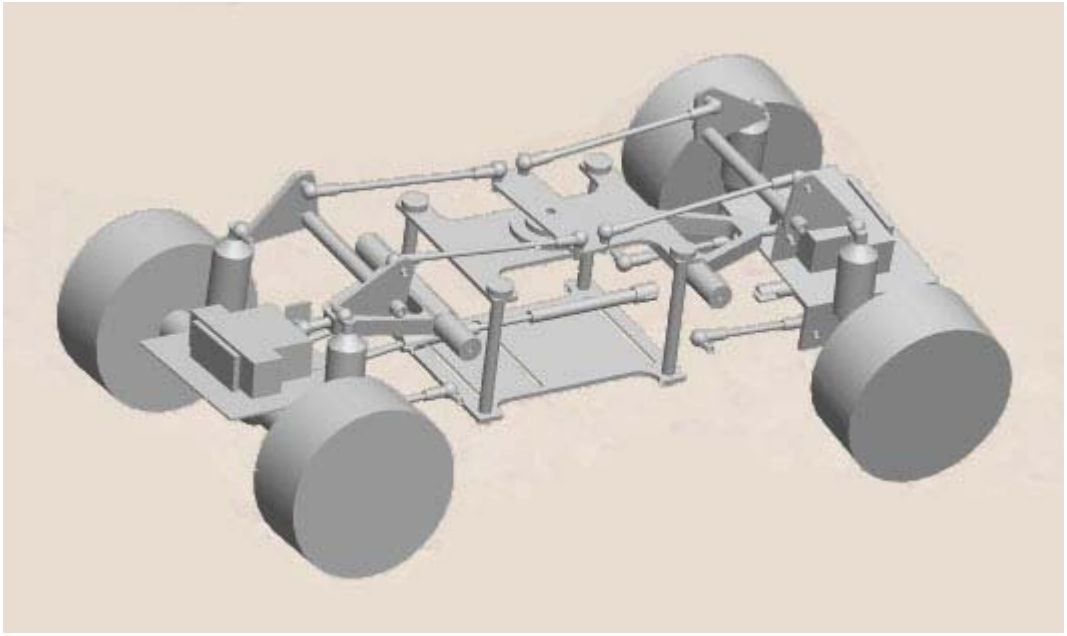

Figure 3 3D CAD model of the robot's mechanisms

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.

2.3. 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.

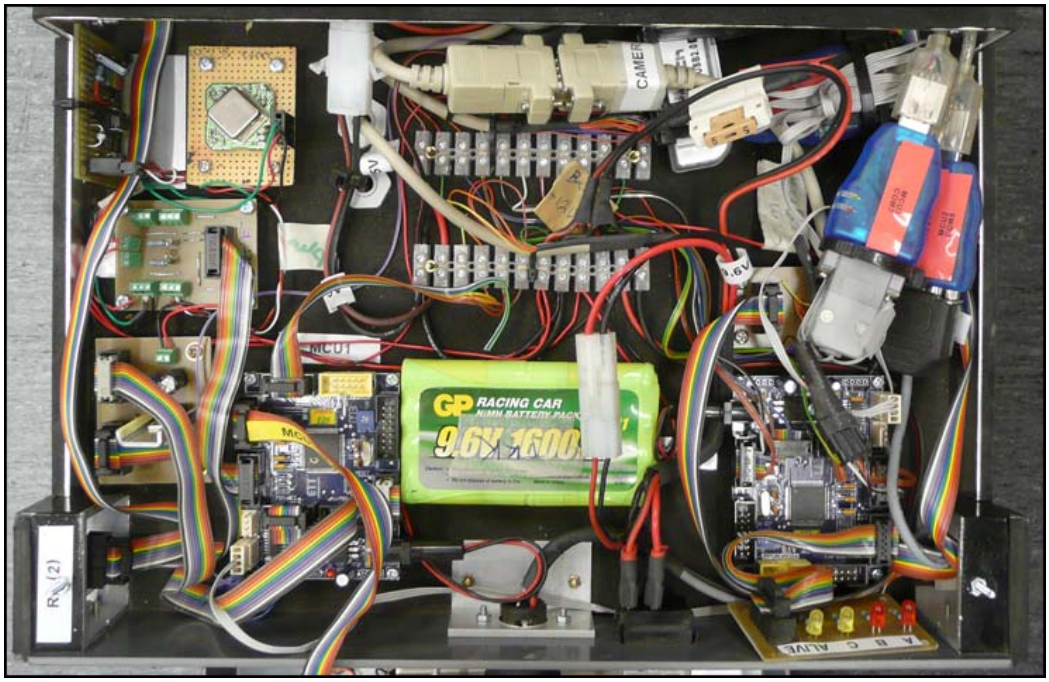

Figure 4 Electrical connections and electronics in the robot

2.3.1. 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.

2.3.2. 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.

2.3.3. 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.

2.3.4. 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.

3. 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].

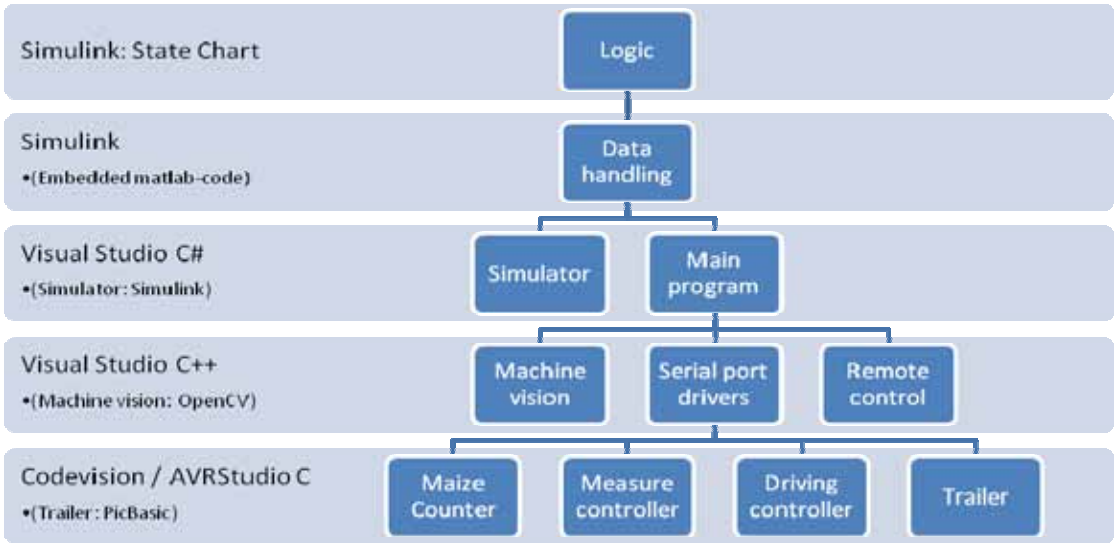

Figure 5 Software modules

3.1. 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].

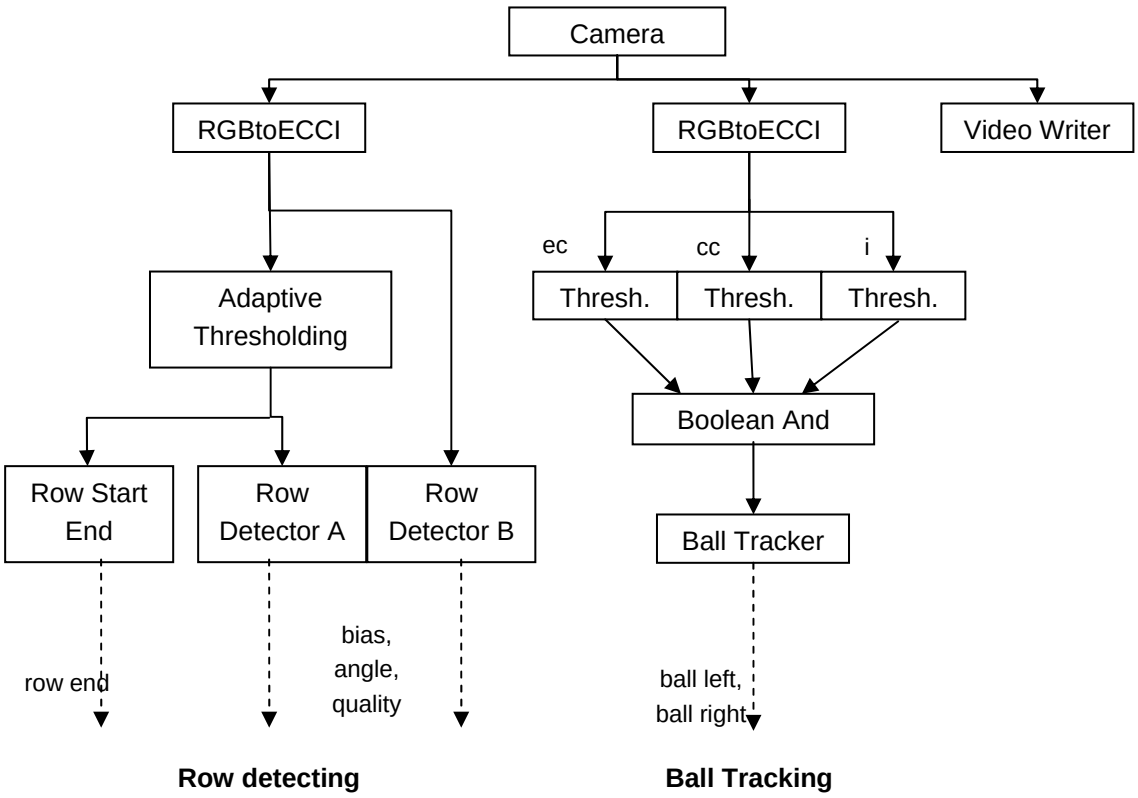

Figure 6 Simplified structure of the machine vision system

3.1.1. 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.

3.1.2. 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.

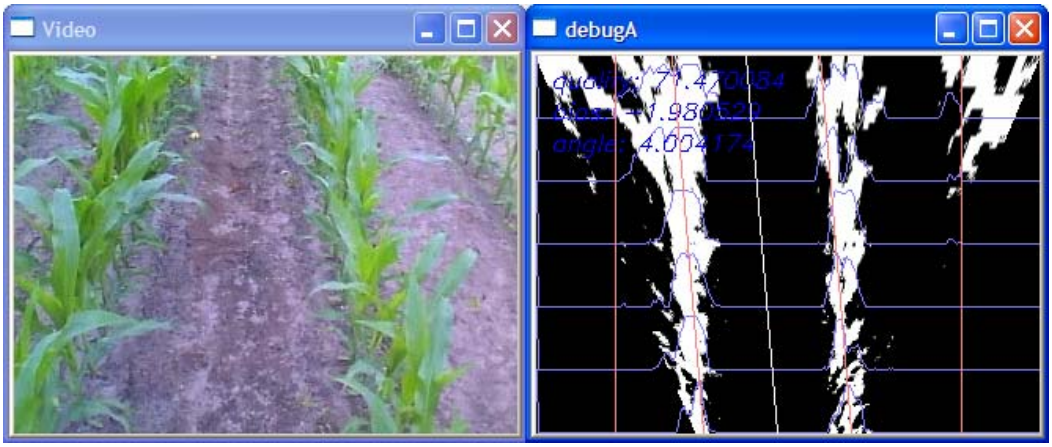

Figure 7 Original camera image and row detector A debug image. White line is the estimated center of the rows.

3.1.3. 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.

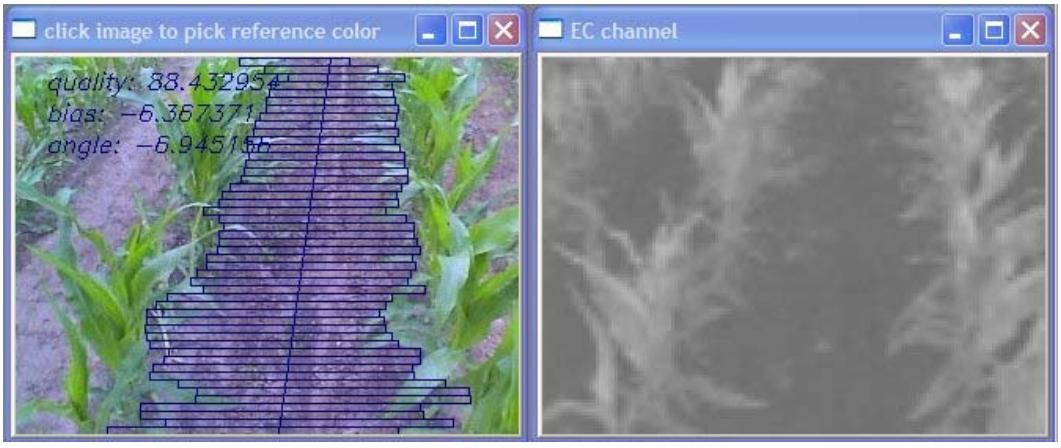

Figure 8 Row Detector B debug output is printed over original image. Right is shown the EC channel from which the ground is detected.

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.

3.1.4. 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.

3.1.5. 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.

3.2. 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.

3.2.1. 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.

3.2.2. 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.

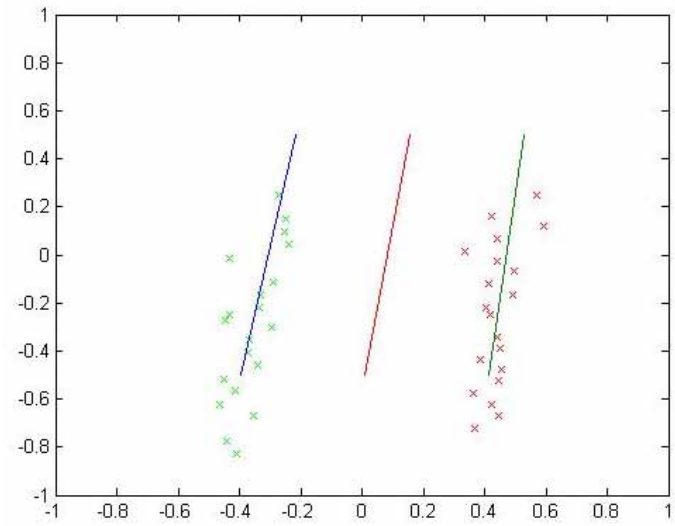

Figure 9 Maize row estimation using ultrasound measurement history

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.

3.2.3. 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.

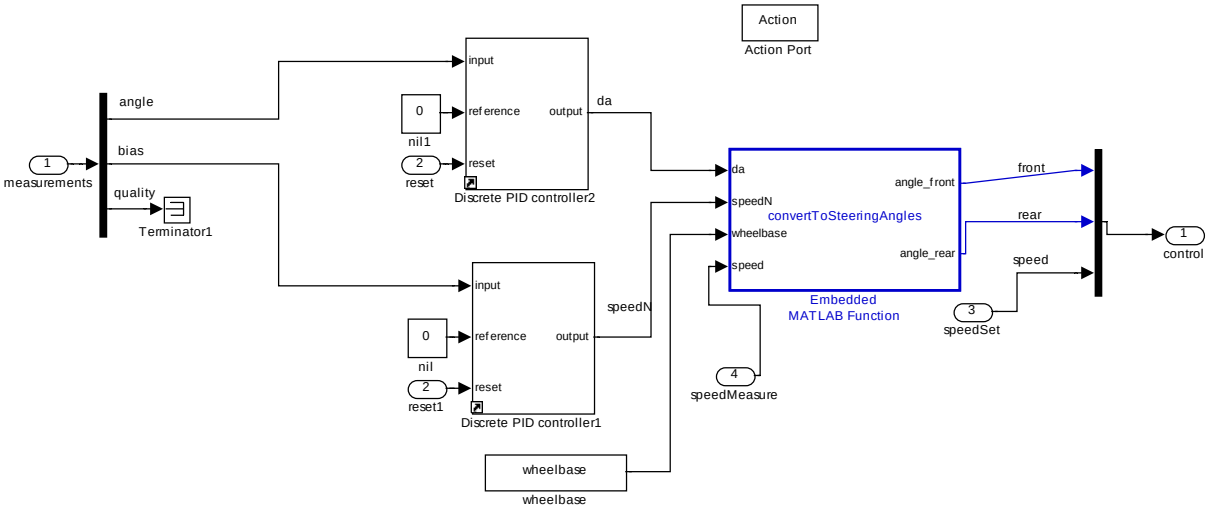

Figure 10 Block diagram of wheel turning controller

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.

3.2.4. 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.

3.2.5. 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.

3.2.6. 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.

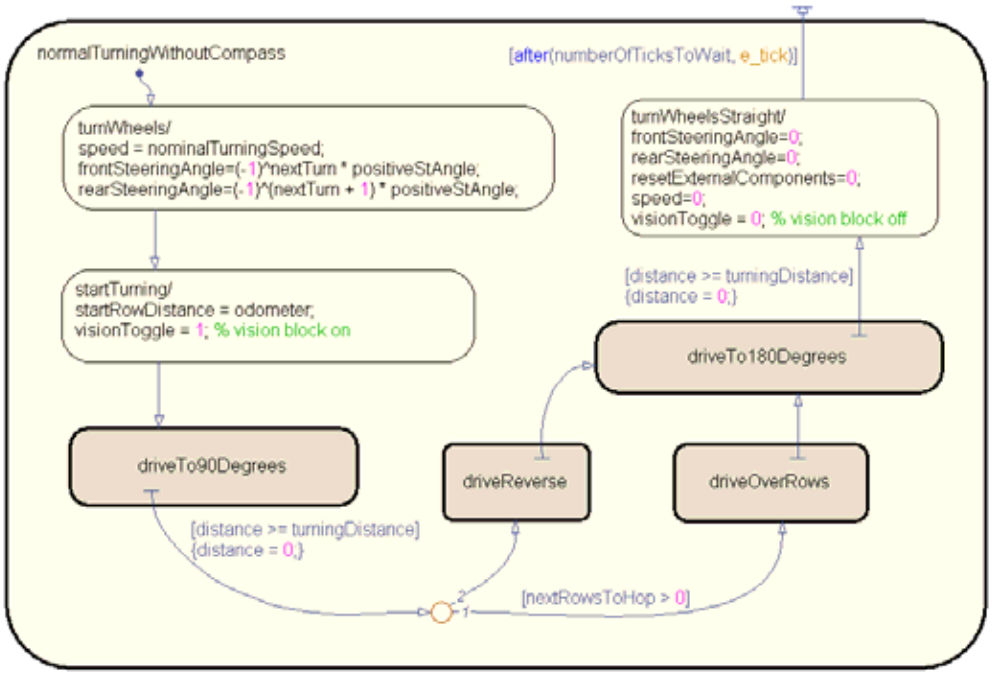

Figure 11 State chart of the normal turning method without compass usage

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.

3.3. 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).

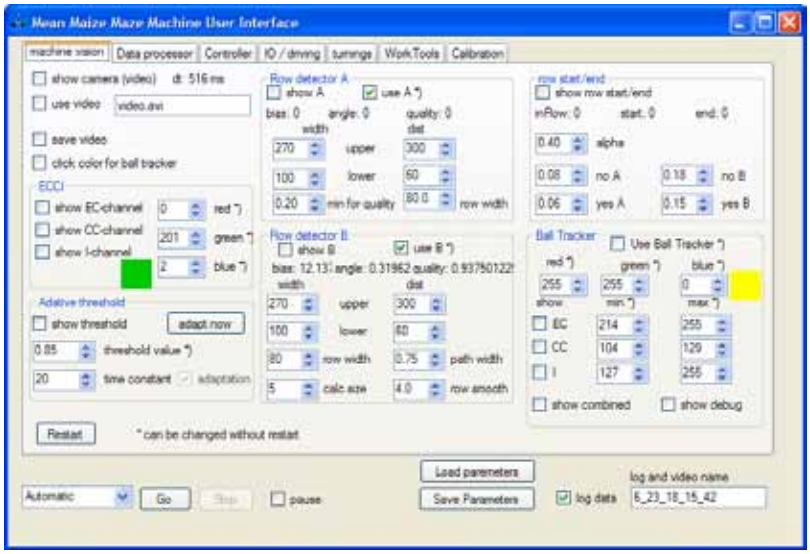

Figure 12 Basic view of the user interface

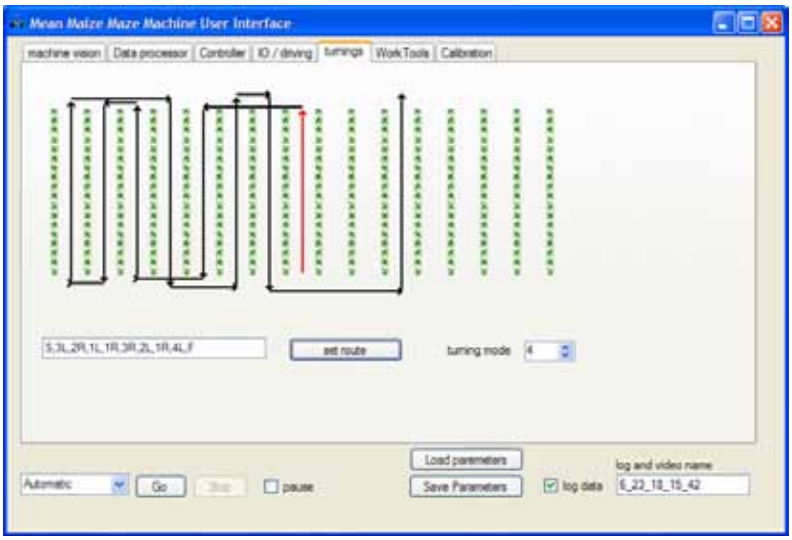

Figure 13 Robots route could be set from the main program

3.4. 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.

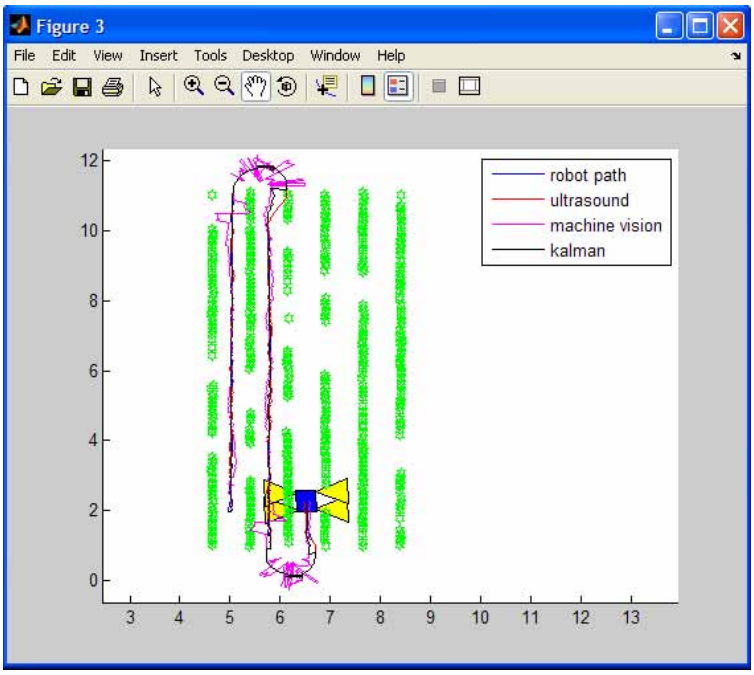

Figure 14 Simulated maize field with robot and some additional information

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.

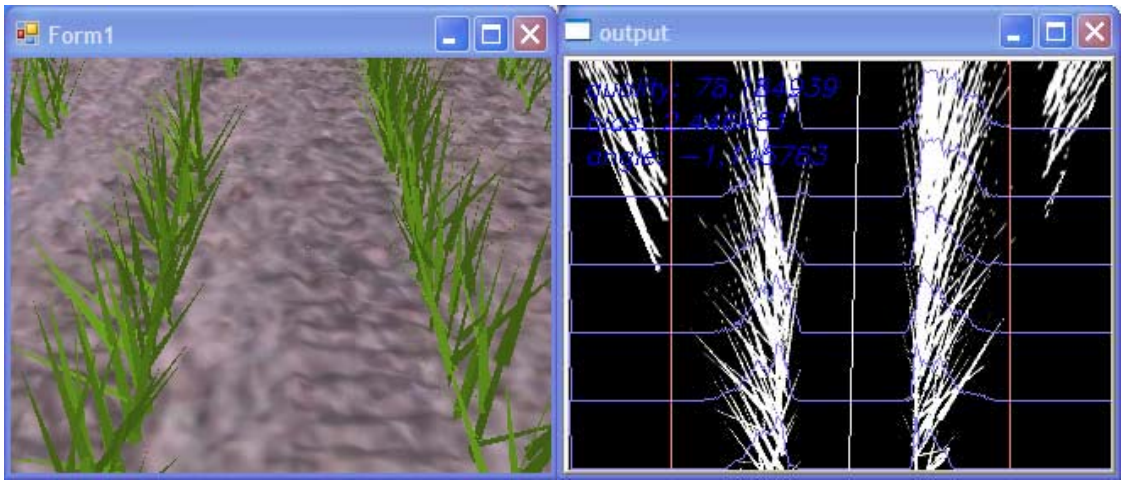

Figure 15 Simulators 3D rendered frame and corresponding row detector A debug image

4. 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.

4.1. 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.

4.2. 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.

4.3. 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

4.3.1. 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

4.3.2. 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.

4.3.3. 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

5. 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.

6. 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:

Ford

hp

koneviesti

Junkkari

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.