

# Wheels of Corn tune



T. Maksimow<sup>1</sup>, J. Hölttä<sup>1</sup>, J. Junkkala<sup>2</sup>, P. Koskela<sup>1</sup>, E.J. Lämsä<sup>1</sup>, M. Posio<sup>2</sup>  
T. Oksanen<sup>1</sup> (advisor), J. Tiusanen<sup>2</sup> (advisor)  
*Helsinki University of Technology, Automation Technology Laboratory<sup>1</sup>*  
*University of Helsinki, Agrotechnology Department<sup>2</sup>*

## Abstract

Wheels of Corn tune was built as a student project for the Field Robot Event 2007 at Wageningen. Robot's main goal was to navigate autonomously between straight and curved rows of maize and turn at the headland to enter the next row. The robot's chassis is based on Demeter, a competitor at the Field Robot Event 2006, fitted with new sensors and improved electronics. The robot is controlled by an onboard laptop that is connected to two microcontrollers. The microcontrollers transfer sensor information to the laptop and control the servos and motors. The operation logic and control design is built with Simulink. In addition to the sensors the robot has a camera that is used in row and weed detection. In the competition the robot finished fifth. In this document the hardware and software implementations of the robot and technologies used are explained.

## 1. Introduction

The field robot Wheels of Corn tune is a result of a student project of four students from the Helsinki University of Technology and two from the University of Helsinki. The project began in October 2006 and we started from where the Demeter team left off after the Field Robot Event 2006. However we redesigned the software and made modifications to chassis. We also added new sensors and built a docking station for the freestyle session.

During the course we experienced a lot of electronic and mechanical problems that delayed the actual testing of the robot until the final stages. Lack of a proper testing environment also made things worse (*Figure 1-1*). Nevertheless we project was very educational in terms of technology and teamwork. In the following we will first discuss the hardware and secondly the software and finally we present some conclusions.

---



*Figure 1-1: Test field in Finland built from birch branches*

## **2. Hardware**

### **2.1. Robot's chassis**

The robot's chassis was built by the Demeter team for the Field Robot Event 2006. Their initial plan was to build a chassis that would be reliable, low cost, low power consumption and would have good off-road properties and good accuracy and ability to respond to instructions. They wanted to build and design the chassis from scratch because only then could they achieve the desired results. [3]

We improved the chassis by adding an aluminium framework for the new bumpers which also gave us more room for the electronic components in the middle of the robot. The bumpers were built for our freestyle session. We also repaired the suspension of the front axel. For the weeding task we built a harrowing system for the golf balls.

### **2.2. Electronics and sensors**

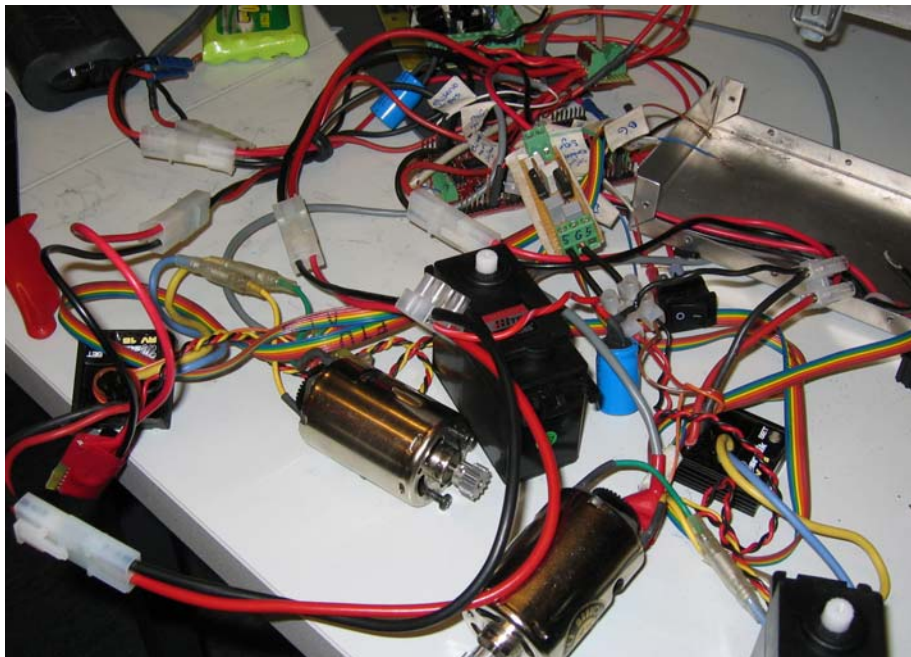
The robot has quite a lot of electrical devices most of which are from Demeter: for computing: onboard laptop and two microcontroller boards, sensors: webcam, five ultrasonic distance sensors, actuators: two DC motors, three servos, two for turning the front and rear wheels and one for turning the webcam.

We added two infrared distance sensors to the sides of the chassis, three infrared receivers to the front for our freestyle and two servos for lowering and lifting up the rakes of the weed-killer rake system.

The robot has two Mavric IIB microcontroller boards, which are equipped with AVR Atmega 128 microcontrollers. In Demeter robot there was one Mavric and one PIC, and as it was not possible to

extend the number of IO's in PIC, the old PIC was replaced with another Mavric and the whole program was reprogrammed. The first board is for collecting sensor data and the second is for controlling DC motors and servos. Both controllers are connected to the onboard laptop via serial or USB bus. The laptop does all the high level data processing and the microcontrollers are mainly IO devices.

We had to replace the old motor controllers and redo the wiring for the entire robot due to strange electrical problems. We had to completely disassemble the robot to locate the problem (*Figure 2-1*). The main culprit turned out to be the old RC-car motor controllers. We made two new H-bridge type motor controllers, which were connected to the other microcontroller and driven by PWM signal. Servos were connected directly to the PWM outputs of the controllers.



*Figure 2-1: Robot electronics attached one by one to find the problems*

Ultrasonic sensors and compass were connected to I2C bus and other sensors were connected directly to the I/O pins of the controller. The robot has a top part that can be detached from the chassis and most of the sensors and the camera and rake servos were mounted to the top part. For this reason we made also an adapter circuit board, which connects numerous I/O and PWM signals from microcontroller board to a single flat cable. The upper part has smaller adapter that connects the flat cable to the wires of sensors and servos.

### **2.3. Docking station**

As a part of the freestyle session we want to demonstrate docking to a station (*Figure 2-2*). After completing a task in the field robots need to recharge. It is easy to navigate to three meter proximity of the station with a cheap GPS, but more accurate local navigation needs to be done with other methods. We have used camera and infrared sensors. When the robot locates the hoods with the camera or sees the infrared beam, it starts to drive towards the station.

The station is made of Finnish birch plywood, second-hand drainpipe and two round lamp hoods (red and blue). There are two copper cables, with spring suspension, in the station to make a connection with the surfaces in the front bumper of the robot for charging the batteries.

Throughout the procedure the remote control interface communicates with the robot to synchronize and control the operation. The remote control interface drives a USB-port attached data acquisition device. The DAQ-device is set to measure current in the copper cables and therefore it can sense when the robot is attached to the docking station, the resistance of closed circuit is known. The interface drives the relay to output 12 V from the station's battery. The orange flashing light on the top of the robot starts to flash in order to demonstrate charging the batteries, power for that is coming from the station. When the charging is done the robot leaves the dock and the relay cuts down the power. After recharging the robot can return to the field with new orders.



*Figure 2-2: The Docking station*

### **3. Software**

The robot's software was built using several tools for different purposes. The machine vision was built using C++ and OpenCV [1] image processing libraries. Robot's operation logic and control design was done with Matlab/Simulink. The main program is written with C# and the Remote Graphical User Interface (RGUI) is implemented with National Instruments LabView 8.

#### **3.1. Machine vision**

A Logitech web camera was situated on top of the mast looking slightly downward. Image processing was done with EGRBI transform (Excessive Green, Red-Blue, Intensity) using OpenCV image processing functions. Also HSV transform was tried, but EGRBI was found to be better and easier to tune than HSV. At first algorithm takes a RGB image and splits it to 3 images: Red image, Green image and Blue image. Each image is used to make 3 new images: Excessive Green image, Red-Blue image and Intensity image.

$$\begin{bmatrix} EG \\ RB \\ I \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} *_{[3*3]} mask \quad (1)$$

$$EG = [2G - R - B]$$

$$I = \frac{1}{\sqrt{3}} [R \ G \ B]$$

$$RB = EG \times I$$

We extended the algorithm a bit, we modified the basic equations (1) to support the detection of any color by clicking the desired pixel in the camera image. This way any color can be changed by equation 2 to Excessive Color (EC).

$$EC = \frac{1}{2 * (R^2 - (B+G) * R + B^2 - B * G + G^2)} \begin{bmatrix} 2R - B - G \\ 2G - R - B \\ 2B - R - G \end{bmatrix}, \quad (2)$$

where I is same as on EGRBI transform and the third component will be cross product,  $EC \times I$ . Let's call this ECCI transform later on. After the transform 3 binary images are made from these 3 new images with thresholds and these 3 binary images are merged to one binary image.

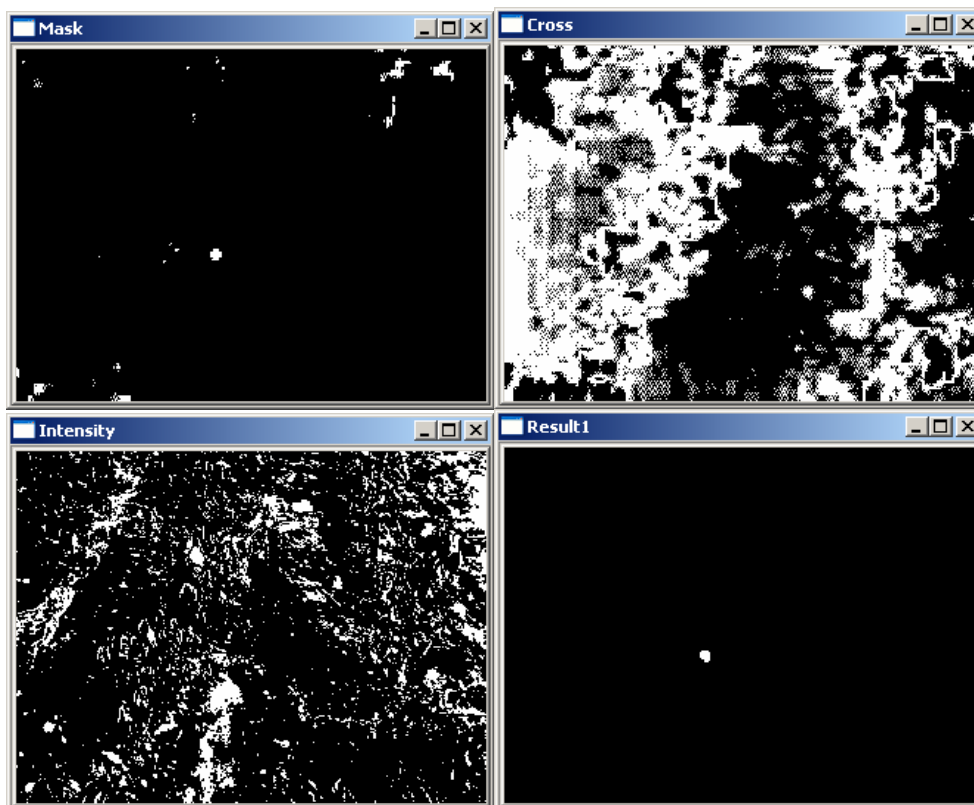


Figure 3-1: Images created by the ECCI transform and the result image for weed detection

### 3.1.1. Row detection

We used two methods for row detection, Hough transform and Histogram row detector, but in the end we found the latter to more reliable.

In the first method rows can be search with Hough transform based on the binary image. Place of robot can be estimated from rows with simple trigonometric calculations (*Figure 3-2*). This is the same method that Demeter team used. [3]

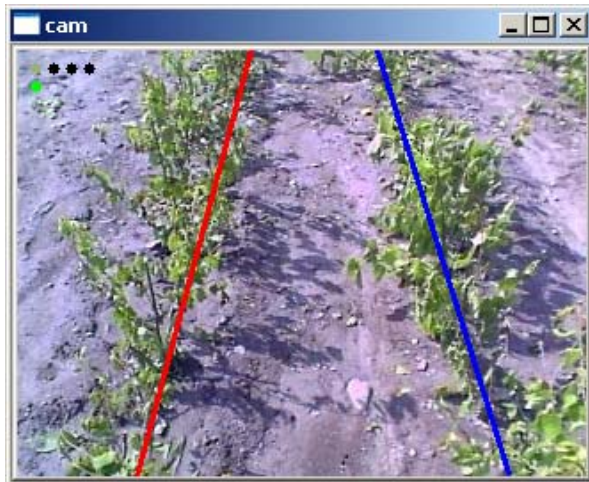


Figure 3-2: Hough transform for detecting lines

The second method is based on an ASAE Publication *Automatic Guidance System With Crop Row Sensor* by H. Okamoto et al [2]. First it does a perspective transform to the binary image and then it slices the picture to six slices and calculates their histograms. From these slices peaks are detected, shifted and combined several times. From the combined histogram the highest peaks determine the necessary shift. Using trigonometry the robot's lateral displacement and angular error can be resolved (*Figure 3-3*).

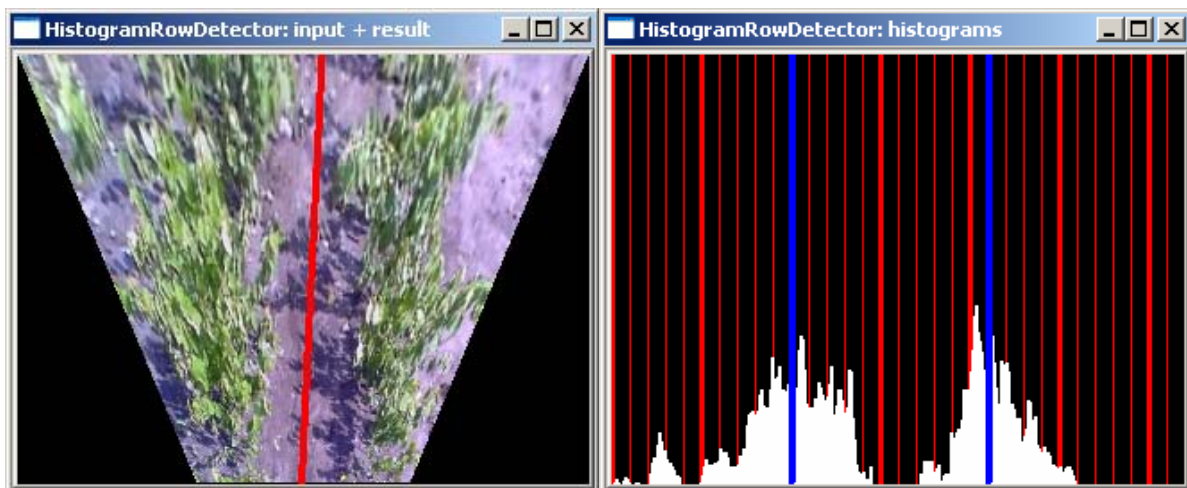


Figure 3-3: Histogram row detector in action: result image, combined histograms and detected peaks

### 3.1.2. Weed detection

Weeds were not only detected from the binary image (*Figure 3-1*), but they were also tracked. This means that separate objects that are visible in the result image are detected one by one and then marked into a list where their coordinates are maintained. As the balls move the list is updated and new balls are being searched for. The detection of new balls was implemented with OpenCV function `GoodFeaturesToTrack` and the updating of their coordinates was implemented with OpenCV function

CalcOpticalFlowPyrLK. When a tracked balls y-coordinate is above a certain threshold, it is removed from tracking and an event to lower the harrow is given. This implementation worked well in our tests but for some reason in the competition the harrows did not lower at all. Perhaps the balls were so close to the stalks that they were not visible in camera because of the leaves.

### **3.1.3. Dock detection**

Dock detection was made by detecting red circle and blue circle from images. Firstly two binary images are created, one with for the red balls and another for the blue. To avoid detection of wrong shapes and sizes, we used OpenCv function HoughCircles to detect the mast of the station. From both of the images circles were searched separately. All of the detected circles were matched with each other to see if there is circle in both images that are of the right size and close enough to each other. If the dock was seen, the direction and distance of dock can be estimated with simple algorithms. This method worked pretty well but was a bit slow and was not very robust.

## **3.2. Sensor Fusion and navigation**

In addition to the camera, the robot has 4 ultrasonic and 2 IR-sensors for navigating between maize rows, a gyroscope and compass for turning at the headland and three infrared receivers at the front for docking.

### **3.2.1. Navigating between rows**

The primary sensors used for navigation are the four ultrasonic sensors located on the upper hull at each corner of the robot. Two IR-sensors were added this year to the lower hull to increase the accuracy of driving in the middle of the row.

The Demeter team used two PID-controllers for controlling the lateral position and angle of the robot [3]. We decided to use a bit more direct approach and control the front and rear of the robot directly with PID-controllers. This gave us the possibility to move more smoothly between the rows by limiting the steering of the rear end of the robot and using the front steering servo primarily for controlling the direction of the robot.

We have two different methods for using the ultrasonic and IR-sensors for navigating between the rows. The first approach is similar than used by the Demeter team, the position of the robot is simulated backwards for a while using robot's kinematic model and the measurements are projected into cartesian space and lines are fitted to them using least square method on both sides separately. With simulator this worked well but at first we couldn't make it work with the robot and the reason was that the odometry measurements were not correctly calibrated. After calibration this method seemed to work well. One thing that we didn't have time to do, was a dynamic modeling for steering servo behaviour, wheels do not turn instantly, and the real steering angle should be estimated.

The other method to estimate position in a row was pure static geometric calculation, the ultrasonic sensor data was used directly to measure the front and rear lateral errors. With this method the robot started to drive logically but ended up being a more or less stable oscillator. With the limited time we had to test the robot in an actual maize field we were unable to tune either ultrasonic method to perfection but we noticed that they seemed to work quite well together.

The fusion of all the sensors for driving between the rows is done with a simple weighted average of the two ultrasonic sensor methods (37.5% + 37.5%) and the camera (25%).

### 3.2.2. Row end detection

Row end detection is made to be very robust and reliable, but still quite fast at the real row end. The ultrasonic sensors seemed to be very reliable from the start, they worked 100% of the time indoors with solid walls and even flowerpot rows, but when we moved outdoors to our makeshift birch branch testing field, we noticed that the ultrasonic sensors weren't reliable enough alone because we got some random row end detections in the middle of the rows.

We tried adding the IR-sensors to the row end logic and managed to get it more reliable results while on the rows but it had a negative effect on the actual row end detection. The IR-sensors often give false signals even when driving on an empty field so we decided not to use them. The camera information turned out to be the ideal pair for the ultrasonic sensors since the actual row end detection was extremely reliable with the ultrasonic sensors alone and with the camera data we could remove the false row end detections when the robot is actually still between the rows.

### 3.2.3. Headland turn

The headland turn is based on odometry and measured angle. The compass and gyroscope are both quite reliable alone, but neither is perfect alone. The compass gives some random spikes and isn't accurate if the robot is tilted and the integrated gyroscope angle drifts even though the bias is calibrated when the robot is started. A Kalman filter was implemented to combine these two. The filter was heavily simplified because we only needed to estimate one scalar variable, the angle.

*Prediction:*

$$X_{pri} = X_{pos} + \omega * dt$$

$$P = P + Q * dt$$

*where  $\omega$  and  $Q$  are the gyro measurement and its variance*

*Update:*

$$e = z - X_{pri}$$

$$S = P + R$$

$$K = P/S$$

$$P = P - K * S * K$$

$$X_{pos} = X_{pri} + K * e$$

*where  $z$  and  $R$  are the compass measurement and its variance.*

These equations are the same as used in the Personal Navigation System by Saarinen et al. [5]

Unfortunately the last two nights of coding and testing somehow broke the filter so it was abandoned and the compass data was used directly and we found it to be quite accurate outdoors when the compass is properly calibrated.

After the competition, a thorough analysis of the logs collected during the competition indicated that the gyro may have been malfunctioning because its bias drifts so heavily in many of the logs that the gyroscope data would've been unusable anyway. The gyroscope was however in earlier tests quite reliable and it could be used for smoothing random spikes from the compass data because of its good short term accuracy.

#### **3.2.4. Navigating to the docking station**

The robot has an array of three IR-receivers on top of the front buffer. The receivers are mounted so that the angle between them is 45 degrees (left, front, right). This allowed us to approximate the direction of the dock with just one IR-transmitter by calculating the vector sum of the voltages given by the receivers (length being 1/voltage). We also used this to roughly estimate the distance to the dock. Since the receivers were mounted at the front of the robot it could only find the dock with them if it approached so that it crossed the cone of the IR-transmitter at some point.

By using the camera and its turning servo we managed to increase the robots field of view and the dock could be found from a much bigger area around it.

The main problem with the camera was the lack of robustness. The Hough circles were sometimes lost even when the robot wasn't moving and it could clearly still see both of the colors. The situation gets much worse when the robot starts moving because the camera is mounted on the mast so high. Even when moving on a flat floor the robot often wobbles enough to cause the camera to swing so much to the side that it loses sight of the dock.

The IR-receivers worked well indoors when we were testing during the winter but they became completely useless when we started testing outside, even when they were not directly exposed to sunlight. The camera however worked well enough alone for the robot to drive to the dock, but not very smoothly.

### **3.3. Controller & operation logic**

The control algorithms of the robot were implemented in Matlab/Simulink. Controllers were developed using Simulink's blocks and embedded functions. We used Matlab's Real time workshop to generate C-code from Simulink models. [4] The generated source code was then used in the main program, which runs in the onboard laptop. In this way the need of writing code by hand was greatly decreased.

The controllers used while driving between the rows are two PID-controllers with anti-windup and filtered derivative. The front and rear errors given by the sensor fusion block is used to control front and rear wheels separately.

Operation logic of the robot was developed using Simulink's Stateflow extension (*Figure 3-4*). It can be used to create state machines and use them in connection with normal Simulink models. Our state machine consisted of states like "between rows", "at the end of row", "turning" etc. Turnings were implemented completely in Stateflow.

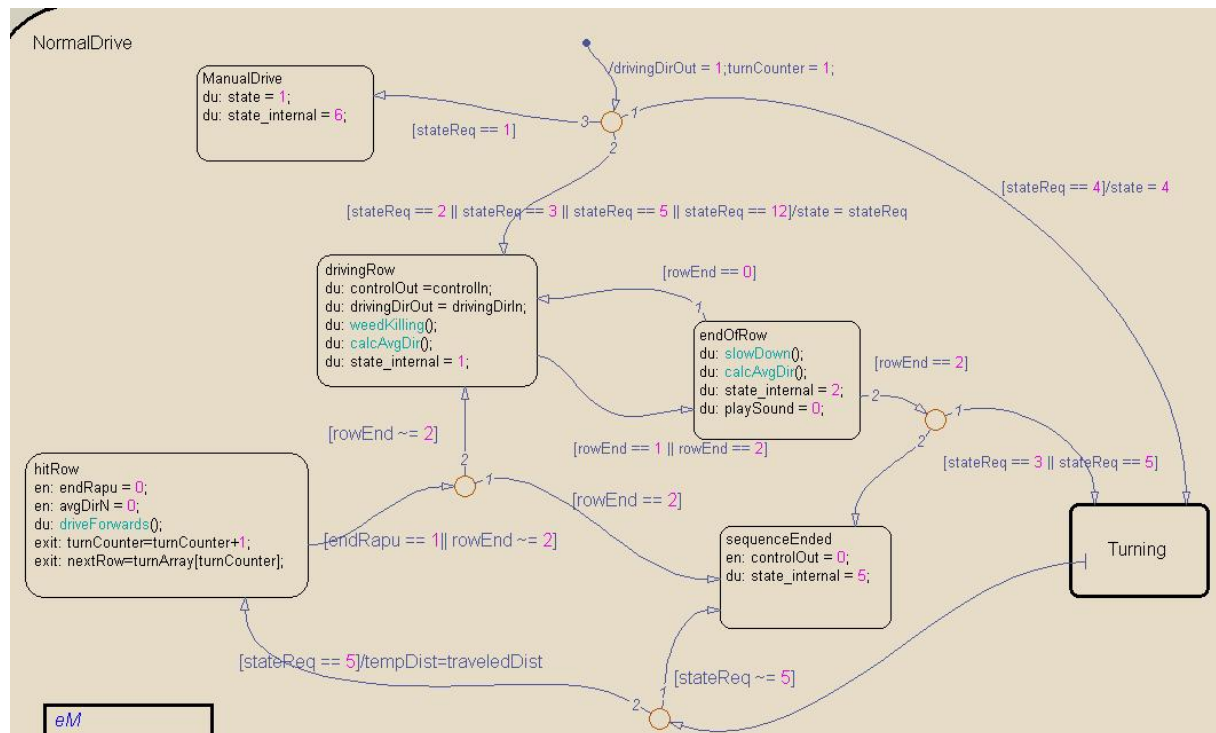


Figure 3-4: Sample of robot's state machine chart

Another advantage of using Simulink's code generation was that the models could be simulated using Matlab. We had a self made simulator which basically simulates the ultrasonic sensors and robot's movement in test tracks (Figure 3-5). The simulator was of great help especially in testing stateflow logic and overall functionality of the whole model. We did also some PID parameter approximation with it, but fine tuning of course had to be done in a field. The body of simulator code (laying maizes, kinematic model and simulating ultrasonic sensors) was got from last years team and we extended it to support docking, infrared sensors and plenty of small thing were improved.

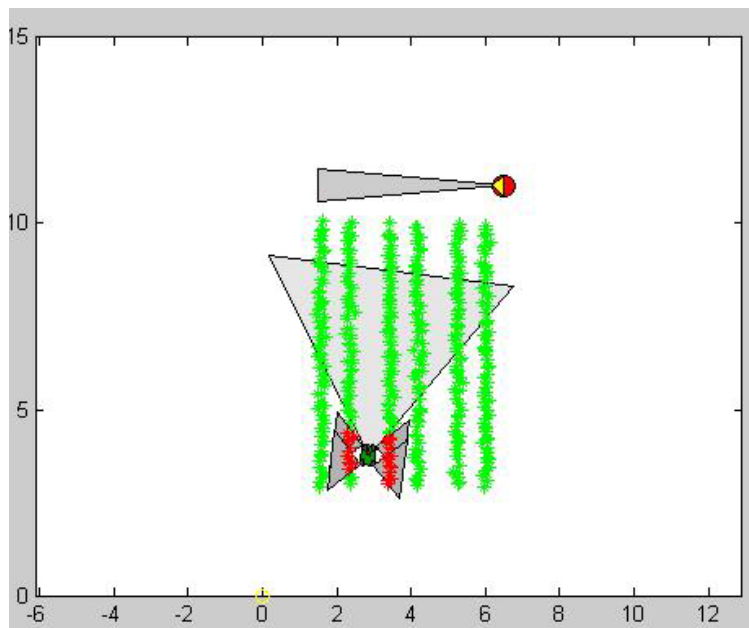


Figure 3-5: The simulator in action

### 3.4. Remote GUI

The robot has a remote graphical user interface on another laptop (*Figure 3-6*). The laptop uses its own WLAN in order to communicate with robot's laptop. The laptops communicate by sending each other data strings in UDP packages. The interface is made with National Instruments LabView 8. The main purpose is to be able set the robot to do specific tasks, drive manually with the joystick or to stop the robot in an emergency. Robots sensor data can also be viewed from the laptop and different variables can be adjusted.

The remote GUI communicates with robot by sending commands as strings. The strings are orders to change the state of the robot, for example "Drive automatically" or "Stop". These strings are then parsed by the robot and passed to the state machine of the robot. The robot also sends all information as strings to the robot. Different values are separated with a semicolon for splitting the string.

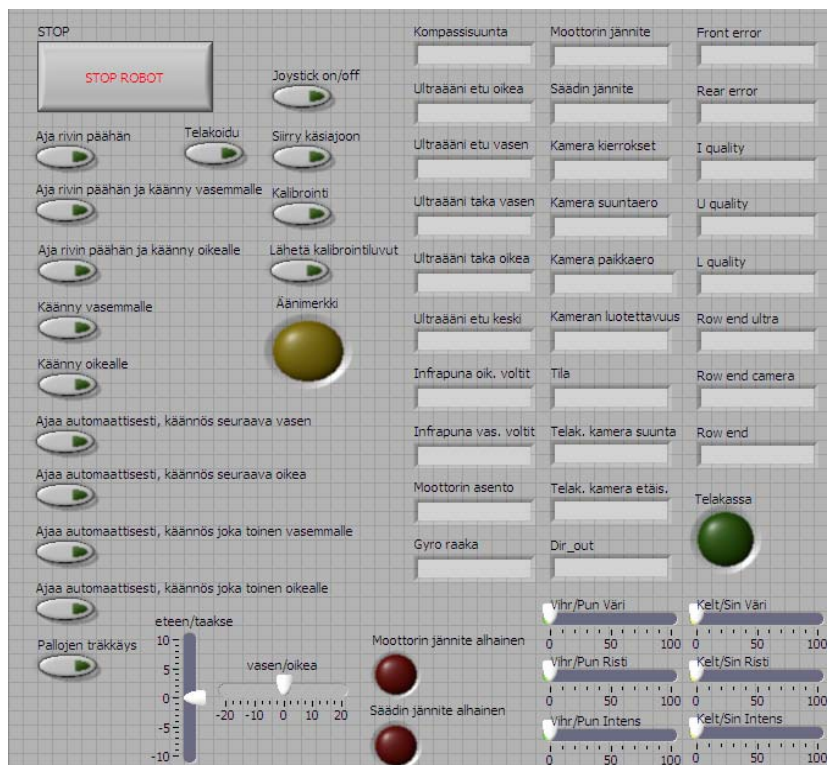


Figure 3-6: Remote Graphical User Interface

### 3.5. Main program

The main program is written with C# and is a multithreaded program. Although it synchronizes the operation of the camera, the robot controller and the Remote GUI interface threads, it mostly acts as a junction between the other components. It passes sensor information to the controller and pushes the controls to the robots microcontrollers.

In the main program it is possible to load and save parameters that are used. For instance the camera and odometry can be precalibrated and then the calibration values can be saved. Parameters are stored in an XML serializable class.

The main program also writes a log of every input and output the microcontrollers and the robot controller have. The log can then be analyzed later back in the lab.

#### 4. Conclusions

Starting from Demeter's chassis and other components that seemed to work somewhat was a mistake for us. We should have built the robot from scratch. Most of the problems we had during this project were due to the old hardware.

Concerning the software, the user unfriendly interface of Simulink also caused some grief, especially when the models started to get complicated and contained several self-made library blocks. Matlab itself is a great tool for data analysis but it is not well suited for a long term software development project. The simulator built with Simulink was a nice tool for validating the basic logic of the controller, but we relied too much on it and did not start testing with the robot early enough. Also the simulator's model needs to be very accurate in order to be useful in real testing. Especially if model based control is used, it is very important that the model used is as accurate as possible.

The camera worked well after we decided to change the Hough transformation to the histogram row detector. The Hough transform is not robust enough for the environment. For detection of the weeds the camera should have been lower. This we did not realise before seeing the real corn field as our test field was not that good a representation (*Figure 1-1*).

In long term software development projects proper version management is very important. This is something we learned in the final coding sessions before the competition.

On the whole the performance of the robot was still decent even though many things went wrong.

#### Acknowledgments

Thanks to our sponsors:



**koneviesti**

#### References

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

[2] 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

[3] Telama, M., Turtiainen, J., Viinainen, P., Kostamo, J., Mussalo, V., Virtanen, T., Oksanen, T. and Tiusanen, J. (2005) Field Robot Event 2006 Proceedings,

[http://automation.tkk.fi/FieldRobot2006/FieldRobot2006?action=download&upname=FRE2006\\_demeter.pdf](http://automation.tkk.fi/FieldRobot2006/FieldRobot2006?action=download&upname=FRE2006_demeter.pdf), (accessed 30.6.2007)

[4] The MathWorks, Inc, Real-Time workshop <http://www.mathworks.com/products/rtw/>

[5] J.Saarinen, J.Suomela, S.Heikkilä, M.Elomaa and A. Halme, "Personal Navigation System", Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan