

DEMETER – Autonomous Field Robot



M. Telama¹, J. Turtiainen¹, P. Viinanen¹, J. Kostamo², V. Mussalo², T. Virtanen²
T. Oksanen (advisor)¹ J. Tiusanen (advisor)²

¹Automation Technology Laboratory
Department of Automation and Systems Technology
Helsinki University of Technology

²Department of Agrotechnology
University of Helsinki

Abstract

An autonomous robot Demeter was developed for the Field Robot Event 2006. The four-wheel driven robot was built on a modified RC-platform. Spring suspension has been replaced with a middle joint. The robot has been build symmetric, and having all the wheels turning, is equally capable of driving into either direction. The robot has been made out of aluminum and is battery driven. The robot is controlled by two microcontrollers and a laptop that all onboard. The basic tasks for the robot are; driving towards a flag, driving between rows and hole detection in grass. The robot archives this by using machine vision and ultrasonic sensors. An electronic compass was included as an additional sensor. Machine vision uses a webcam attached to a servo that rotates the camera to point the robot's driving direction. The robot comes with a trailer that measures soil hardness and moisture. The work was done by a group of students from two universities during the semester 2005-2006. This document describes the development and technology overview of the robot.

Keywords

Field robot, Machine vision, autonomous navigation

1. Introduction

In August 2005 a group of six students began planning the building of a robot that would take part of the Fieldrobot competition 2006 in Hohenheim. A similar team from the same universities had taken part in Fieldrobot competition in 2005 under the name Smartwheels. After sharing information with this team, it was concluded that building the robot should start from a clean table. This time, a considerable amount of effort was put into planning and designing the robot. Number of different design possibilities were evaluated, before choosing the current design. Great care was put into building the mechanics platform to perform as well as possible. This delayed the project and it was not until April that the first tests could begin. That and various unexpected problems kept the team occupied to the last minute.

2. Materials and methods

2.1 Hardware

2.1.1 Chassis

The main task of the chassis of the robot is simple: It should be able to carry the measurement devices and provide the robot the ability to operate in field conditions. There are numerous different ways to achieve this goal and the selection of the technical solution depends on many factors. At least following properties of the chassis are desirable:

- Reliability
- Low cost
- Good off-road properties
- Good accuracy and ability to respond to instructions
- Low power consumption

In practice the low budget of student project forces to some trade offs and all of these properties can not often be fulfilled in the desired way. However, it should be kept in mind that the chassis of the robot is the base on which the all the sensor and computer systems are built. It is obviously essential requirement that the chassis is able to drive all the equipment to the position defined by the controlling computer.

The position of the robot and the position defined by the computer are not the same due to measurement errors and perturbations. There are many sources of perturbations when the robot operates in the field. Error to the dead reckoning position measurement can be caused by e.g. inaccuracies in wheel angles, slip of the wheels, errors in rotation measurements of the wheels and so on. These errors cumulate when driving the robot for a longer time in the field and therefore additional position measurement and navigation systems are needed. Despite of the additional navigation systems the off road properties of the chassis and its ability to follow control signal have a significant impact on the performance of the robot. It is not reasonable to compensate the large errors caused by the chassis with a sophisticated computer control program. If the errors caused by the chassis can be kept as small as possible, the reliability of the navigation and accuracy can be improved. The key point in developing the motion system of the robot is to ensure good controllability in all conditions that can be normally expected in the field.

The chassis for the robot is based on a Kyosho Twin Force R/C monster truck. Building began with the original chassis which needed some modifications, but eventually a completely new chassis was build. Only the tires, axles and some parts of the power transmission were kept in the final version of the robot.

The chassis is newly built with self-made parts with no previous drawings, though based loosely to the original Twin Force parts. Aluminium was chosen because

of its lightness. The engine rooms have been separated from the rest of the chassis and the microcontrollers have been placed into a separate slot to allow easy access. Chassis is four-wheel-driven and each one of the engines powers one axle. All of the springs have been removed, and instead there is a large joint in the middle of the chassis providing the robot manoeuvrability somewhat similar to farm tractors.

The vehicle is electrically driven with two Mabuchi 540 motors. The motors are driven by two PWM amplifiers.

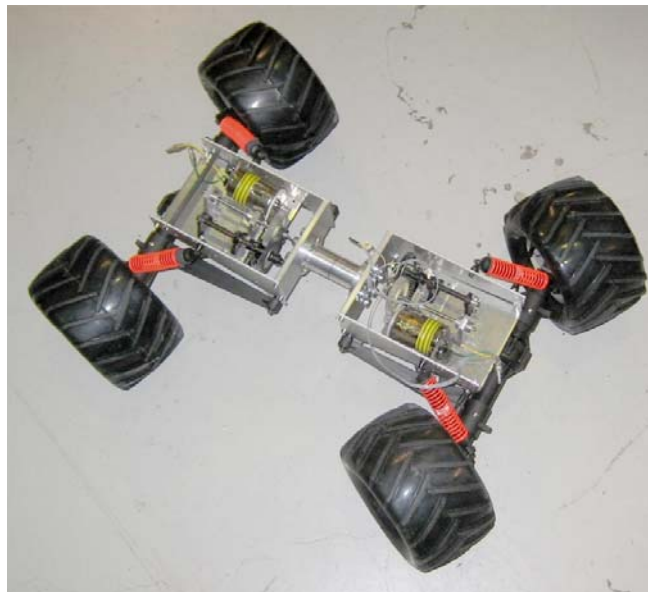


Figure 1 Motor and steering unit

The Motor unit of the Twin Force monster truck was modified to meet the requirements of the field robot. Two separate motor units were used for both front and rear axis. The motor units were identical and they were operated independently from each other. The use of two motor units enables the independent velocity control and independent steering of both front and rear axis. The main modifications done to the motor unit were:

- The maximum speed of the Monster truck was reduced to one quarter by adding an additional pair of gears.
- An optical encoder was attached directly to the shaft of the Mabuchi drive motors.
- A steering servo was attached to the motor unit

The chassis didn't originally have four-wheel-steering, so it had to be build with optional parts provided by Kyosho. Four-wheel-steering was necessary to give the robot enough turning radius. Two HS-805BB+ MEGA servos were used for steering and they seemed to provide enough steering power for the robot.

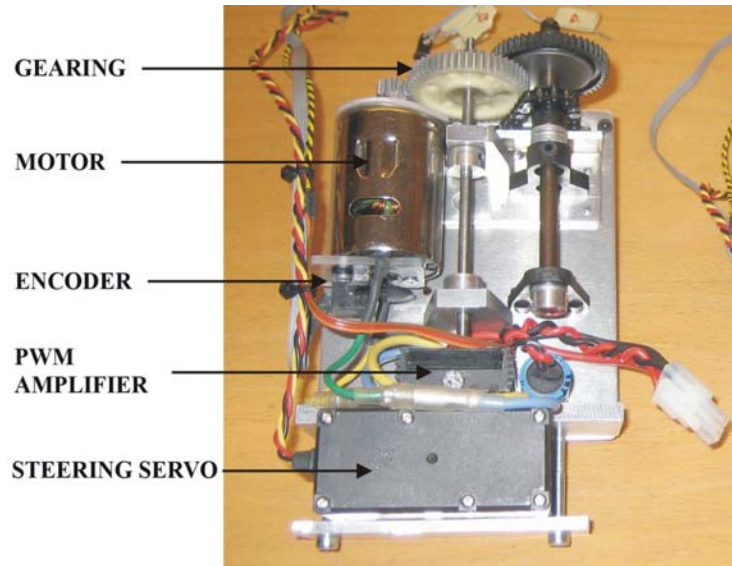


Figure 2 Power supply

Eight (8) pieces of 1.2V rechargeable Ni-Mh batteries soldered in series were used as the power supply of the robot. The capacity of each battery ranged from 7000 mAh to 9000 mAh depending on the battery package that was in use. The power from the batteries was proven to be adequate.

2.1.2 Sensors and machine vision

Four Devantech SRF08 ultrasonic sensors were used together with a Devantech CMPS03 I²C compatible compass. The sensors were connected to I²C sensor bus. For machine vision the robot has a Logitech QuickCam Pro 5000 camera attached to a carbon fibre pole with a servo motor to enable a turn of 180 degrees. The compass was placed to the camera pole, high enough to avoid magnetic fields that would create interference with it.

2.2 Processing equipment

2.2.1 Microcontrollers

The robot has two microcontrollers: ATMEGA 128 and a PIC 18F2220. The first one was used to control the Msonic power controllers for the DC-motors, steering servos, and the camera-servo. The second one was used to collect sensor input from compass, ultrasonic sensors, and to connect to trailer's inputs and outputs.

2.2.2 Laptop

A P4 laptop was used to process all necessary tasks along with the microcontrollers.

2.3 Hardware for the freestyle

2.3.1 Trailer

The trailer was built from scratch. The platform had to be made big enough to fit all of the equipment and with enough ground clearance to give the linear motor enough room to function. Old Tamiya Clod Buster wheels were used together with a new spindle that was made to increase manoeuvrability.

The main equipment used in the trailer is a linear motor provided by Linak. All the electronics have been fitted into two boxes. The linear motor is operated by two relays. The primary idea for the freestyle was to measure the penetration resistance of the soil (cone-index) and simultaneously measure the moisture of soil by measuring the electrical conductivity of the soil.

The linear motor was used to thrust two spikes into soil. Soil penetration resistance could then be measured from the amount of current consumption changes in the linear motor. Simultaneously the electrical conductivity of the soil was measured with an input voltage of 5V directly from the microcontroller of the robot. This was an experiment and it has to be remembered that the soil conductivity measurements rely heavily on other properties of the soil, especially on the amount of different salts there are in the soil.

The linear motor operates with a 12 V lead-acid battery, which, although being quite heavy, doesn't give the trailer enough mass to thrust into hard soil. However the trailer cannot be too heavy for it has to be pulled by the robot. The weight of approximately 9 kilograms was rather easily pulled.

3. Camera & Machine vision

3.1 Camera

Logitech QuickCam Pro 5000 was used for machine-vision. It's a higher end webcam, which is still quite cheap. Camera is capable of 640*480 resolution, but 320*240 resolution was used to improve performance and this doesn't require so much data transfer. Camera sends about 25 images per second to laptop. Laptop processes about 20 images per second and calculates parameters and image conversions for each frame.

3.2 Machine vision

Machine vision software was developed on Microsoft Visual C++ 2005 and Open Source Vision Library (OpenCV [1]). OpenCV provides libraries and tools for processing captured images.

3.3 Preprocessing

Image processing was done by EGRBI [2] color transformation (Excess Green, Red-Blue, Intensity). At the beginning of algorithm, image is split into 3 spaces: red, green and blue. In EGRBI transformation a new image-space is created. Components are Green, Intensity and Red-Blue (cross product of green and intensity). Intensity is $1/3(R+G+B)$ value).

EGRBI is calculated by matrix product.

$$\begin{matrix} \text{Image} & * & \text{mask} & = & \text{result} \\ [320 \times 240 \times 3] & & [3 \times 3] & & [320 \times 240 \times 3] \end{matrix}$$

By changing mask Excess Red can be calculated easily.

The main idea of using this method was to detect green maize rows from dark soil. Green pixels could be detected from camera image by adding more weight on green and by 'punishing' red and blue.

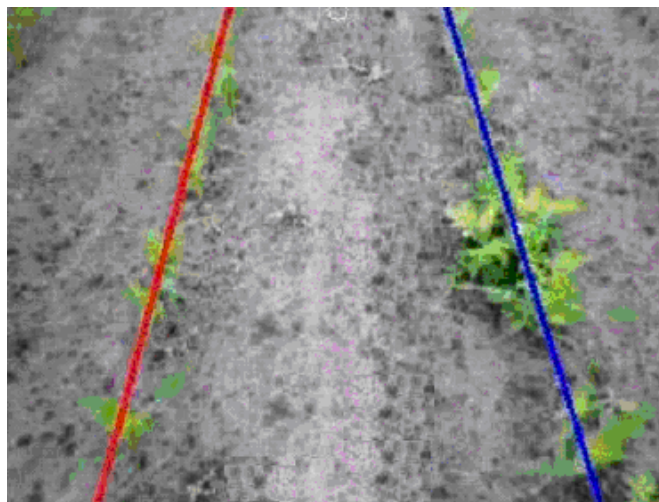


Figure 3 Calculated Hough lines from binary picture.

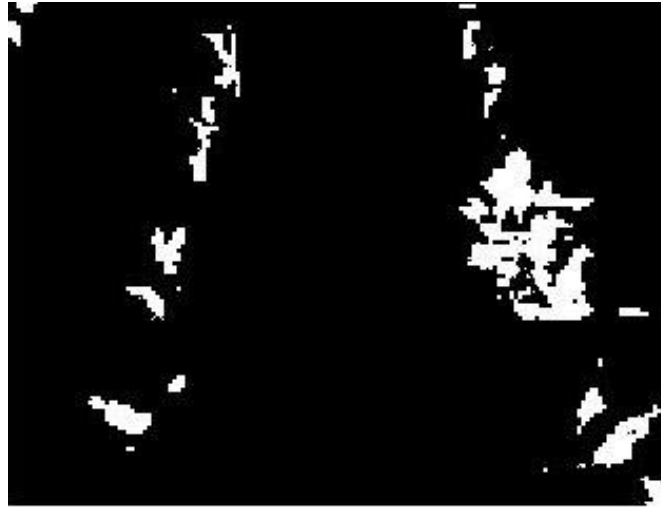


Figure 4 Binary picture, used EGRBI transformation.

Estimate of robot's positioning and angle between maize rows was made by Hough transform. Hough transform is quite heavy to calculate. For each pixel (in binary image) 2 parameter values (angle, distance) are calculated. So the pixels in binary image have to be kept as low as possible. Hough transform returns many lines that fit with the pixels. 10 best lines from each side are taken and mean value of the best lines is calculated. As a result left and right lines are gotten. From these the positioning and angle errors are calculated. The information is send to controller which then calculates correct steering controls.

3.4 Dandelion detection

While driving between maize rows, dandelions must be detected and counted. EGRBI with a yellow mask was used to detect yellow. This was made by finding proper intensity levels and different weighting in R/G/B-values. After binarization each dandelion became a contour. Each contour has position and area. Each dandelion should be detected only once so the positions of contours between image frames were compared.

3.4.1 Hole detection in grass

This section was performed by using inverse green detection. 10cm x 10cm hole covers a well-known proportional area in a picture, so hole can be detected in binary-image with correctly calibrated threshold value.

3.4.2 Driving towards flag

In this section, red flag was detected, in a way similar to detecting the green maize, except that this time a red mask was used instead. Center of the flag was calculated as the center of mass from the threshold binary image. Error was sent to controller.

4. Software

The microcontrollers only function as I/O devices and do very little processing to the data. One of them has the speed controller but apart from that all the intelligence is found in the main program which runs on a laptop PC. The main program was implemented in C++ .NET with MS Visual Studio. Camera- and machine vision functions were done using OpenCV-library.

Controllers are designed with Simulink and compiled into DLLs which are loaded into the main program.

The program's architecture is seen in the picture below. All the blue classes represent different threads and run in 50ms loops.

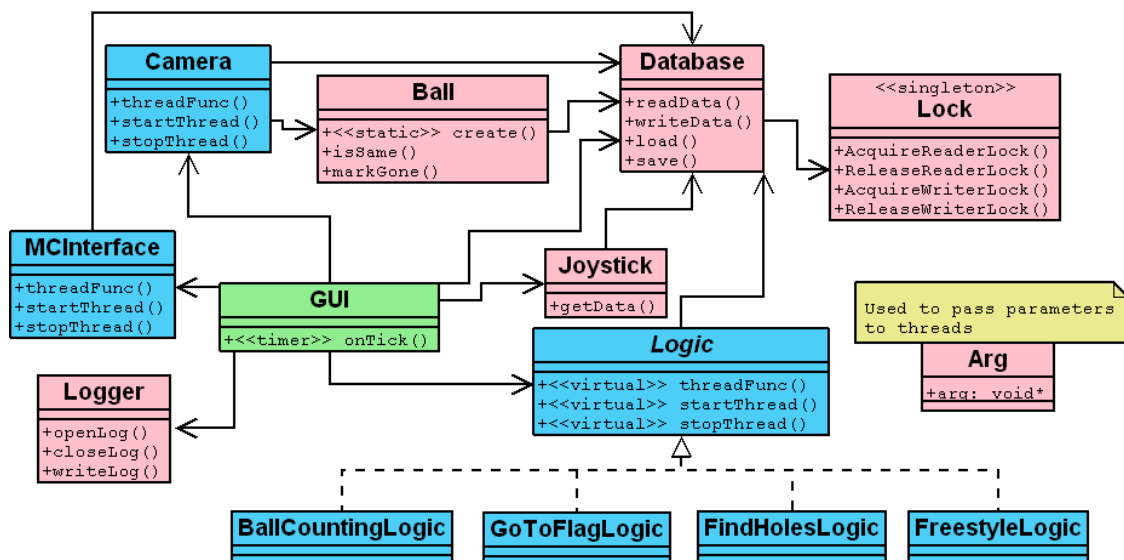


Figure 5 UML class diagram of the main program

GUI: Handles communication between user and the program. Has a 100ms loop.

Database: Contains all the data and takes care of synchronization. Have also methods to construct input arrays to different controllers.

Lock: Used by Database to prevent access to same data by different threads simultaneously. Implemented using Singleton pattern, only one instance allowed.

Joystick: Reads joystick using DirectX-interface.

Logger: Writes all input- and controldata to a log-file.

Camera: Gets picture from camera and processes it. Runs in it's own thread.

MCInterface: Communicates with both microcontrollers. Runs in it's own thread.

Ball: Used to keep track of seen balls, so that each ball is only counted once. No public constructor. Only way to create new instances is to use createBall() method, which checks whether there already exists a ball at the given coordinates.

Arg: A managed class used to pass unmanaged object to Threads, which require managed parameters.

Logic: Reads the DLL for the controller. The base class for all the logics. This is an abstract class and all the real logics are derived from this. The derived classes have different routines to do different tasks.

5. Simulator

Simulator was implemented to better design the controllers. Implementation was done using Matlab and Simulink. It can be used to simulate the use of ultrasonic sensors. The simulator has an editor that can be used to create test tracks. The tracks also contain a correct route that is compared to simulated route to calculate error values. These values can be used to measure how good the controller is.

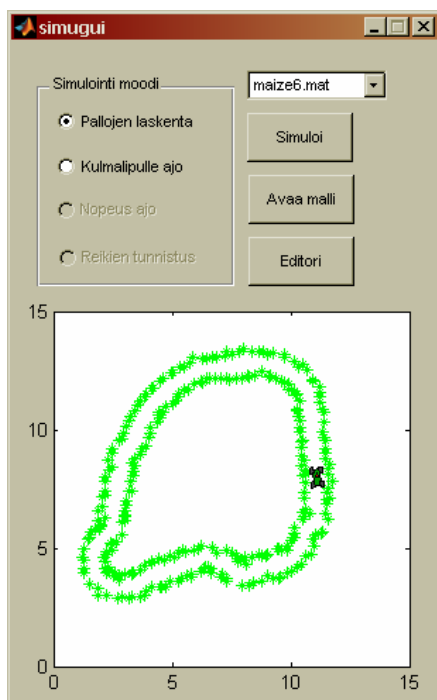


Figure 6 UI of the simulator

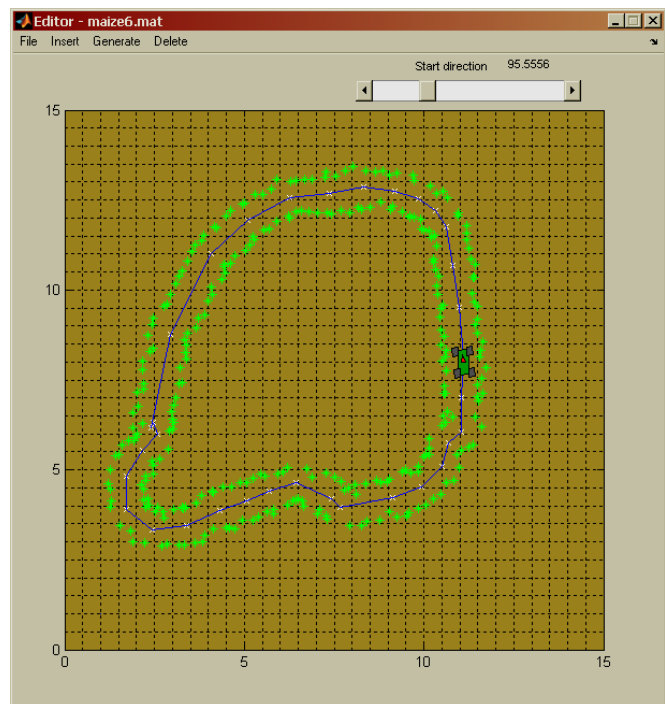


Figure 7 Editor for the simulator

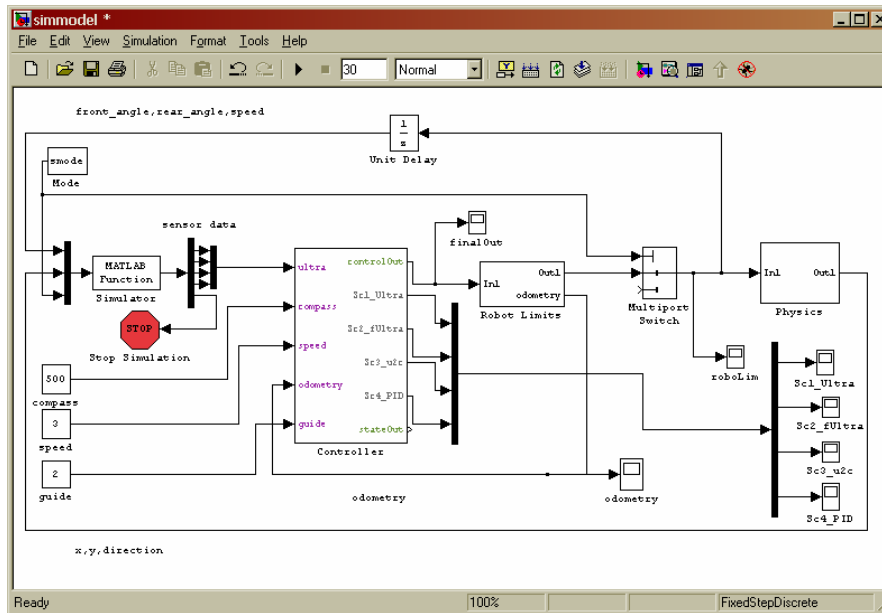


Figure 8 Simulink model of the simulator

6. Control logics

Control logics were developed in Simulink / Matlab. Each task was given an unique controller. Controllers were developed as a set of Simulink library blocks that could simultaneously be used in simulator and controller-export models. C code was generated from the Simulink model using real-time workshop. The final product from the simulink model was a Dynamic Link Library file. Reason behind the idea was to make the imports to the main program easier. The dll came with a function that was called from main program every 50 milliseconds. The function was given all the sensor information (ultrasonic, camera, odometry), as well as some parameters as an input. The output returned steering angles for front and rear wheels and speed, together with some other controls and debugging information depending on the task.

6.1 Driving between the rows

Of the four different controllers, Rowcontroller was the one used for driving between rows. The controller had four basic parts: filtering and processing of the sensor information, sensor fusion, control, and state flow, shown in figure 9. Filtering and processing was mainly done for the ultrasonic sensors and compass as the image processing had already been done in the camera class. Two methods were used to get position and direction errors from readings of the ultrasonic sensors. One method used direct filtering, that calculated the position and directional error from past 5 readings, filtering off any miss readings. The second method used backward simulation of the robot movement. Axes were drawn from the robots current coordinates and the robot movement was simulated backwards for a short distance while the sensor readings were plotted. After that a least square roots line was fitted over the plots to estimate the rows

Figure 10 Simulation of driving in a maize field, right side shows the result of a backward simulation with least squares lines drawn.

6.2 Driving towards the flag

For driving towards the flag, there was a controller that only had a single PID controller. This time there was only the directional error to deal with as the position error could not be measured. Thus whenever the robot was turning, both wheels were turned the same amount to opposite directions. The directional error was gotten from the camera class. Flagcontroller also had a state flow block with couple stopping states. State changes were triggered based on information gotten from an ultrasonic sensor in front of the robot. Some additional rate and other simulink blocks were used to smooth the robot movement.

6.3 Searching for holes

Our strategy for hole-seeking was to cover the whole area by driving straight and moving little sideways in both ends. As the robot had a 180 degrees turning camera and was equally capable of driving in to either direction the idea was to keep the robot heading direction same all the time. This was mainly done by just giving same steering commands to both front and rear wheels, but also with a help of a PID controller that used filtered compass reading to prevent robot from totally losing its heading. The controller came with two state flows, one for driving, and one for actions when a hole was seen.

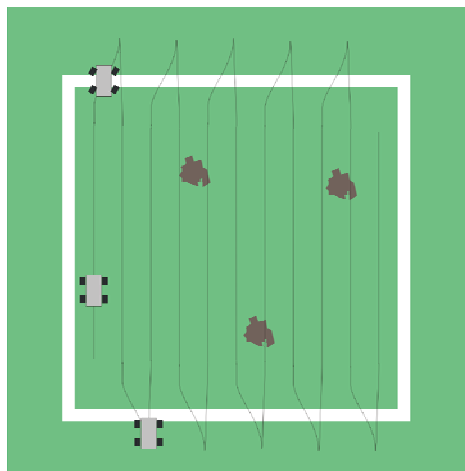


Figure 11 Robot is facing the same direction at all the time in hole-seeking while camera is rotated 180 degrees to point into the current driving direction. Robot navigates through the 5m * 5m area, and checks whether there are any holes in the grass.

7. Conclusion

Quite many algorithms and other possibilities were researched and constructed, yet never used in the final product. They have been a bit of wasted time in making the robot, but might not be such a waste as far as the learning goes.

While there were little problems with the laptops, they might still be considerable choice for this kind of projects. Laptops provide enough processing power for machine vision, and can also be utilized in the development part. It was however noticed that the UI should have been on a remote computer. The machine vision should be made more adaptive to reduce any need for manual calibration, to really get use of this kind of application. The use of Matlab and Simulink turned out to be very useful in the development of control algorithms, especially in the initial testing and debugging. Being able to first test the controller with a simulator and the directly export it to the main program was a great help. The use of a stateflow block was also found useful as it made the understanding of the process easier for other team members, and it made the debugging faster. The mechanics performed well, especially having the middle joint instead of spring suspension has seemed to be a good choice.

8. References

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

[2] Nieuwenhuizen, A.; van den Oever, H.; Tang, L.; Hofstee, J.W.; Müller, J. 2005. Color-Based In-Field Volunteer Potato Detection Using A Bayesian Classifier And An Adaptive Neural Network. In: ASAE Annual International Meeting, Tampa, Florida, 17-20 July 2005.

Koolen A.J. & Kuipers H. Agricultural Soil Mechanics. Springer-Verlag 1983. 241 p.

Srivasta A.K., Goering C.E. & Rohrbach R.P. Engineering Principles of Agricultural Machines. ASAE Textbook Number 6, Revised printing 1995

Rial, W. S., Han, Y. J. Assessing Soilwater Content Using Complex Permittivity. Transactions of the ASAE. VOL. 43(6): 1979-1985

9. Acknowledgements

This work has been made possible by the following sponsors:

Henry Ford foundation Finland
HP
Linak
Koneviesti
OEM Finland