Development of an Autonomous Forest Machine for Path Tracking

Thomas Hellström, Thomas Johansson, and Ola Ringdahl

Department of Computing Science
Umeå University
Sweden
{thomash,thomasj,ringdahl}@cs.umu.se

Summary. In many respects traditional automation in the forest-machine industry has reached an upper limit, since the driver already has to deal with an excess of information and take too many decisions at a very high pace. To further automation still, introduction of semi-autonomous and autonomous functions are expected and considered necessary. This paper describes an ongoing project along these ideas. We describe the development of the hardware and software of an unmanned shuttle that shifts timber from the area of felling to the main roads for further transportation. A new path-tracking algorithm is introduced, and demonstrated as being superior to standard techniques, such as Follow the Carrot and Pure Pursuit. To facilitate the research and development, a comprehensive software architecture for sensor and actuator interfacing is developed. Obstacle avoidance is accomplished by a new kind of radar, developed for and by the automotive industry. Localization is accomplished by combining data from a Real-Time Kinematic Differential GPS/GLONASS and odometry. Tests conducted on a simulator and a small-scale robot show promising results. Tests on the real forest machine are ongoing.

1 Background and Introduction

This paper describes an ongoing project of the design and development of an autonomous path-tracking forest machine. This kind of product is part of a long-term vision in the forest industry [4], of developing an unmanned shuttle that transports timber from the felling area to the main roads for further transportation. The main advantages with such a solution are lower labor costs and less ground damages and emissions due to the lower weight of an unmanned vehicle (the cabin alone weighs several tons). The general requirements and conditions for the development of this kind of product are not addressed in this paper. It focuses instead on one of the necessary components: autonomous navigation, which involves sensing and moving safely along a user-defined path in a dynamic forest environment.

Unmanned vehicles in off-road use have been for long an active area of research and development. The mining company LKAB has been using vehicles in underground mines for many years, with reflective markers to aid the laser-based navigation system. Due to safety reasons combined with high demands on availability, these systems are no longer in full commercial operation. Localization techniques of autonomous forest machines based on a combination of odometry and artificial visual landmarks are described in [9]. Requirements and system design for a robot performing selective cleaning in young forest stands is described in [14].

The initial design and ideas underlying the project reported in this paper are described in [5]. More technical detals are found in [3]. The resulting system has two modes of operation: *Path Recording*, in which the human operator drives or remote controls the vehicle along a selected path back and forth from the area of felling to the transportation road. In this phase, position, speed, heading and the operator's commands are recorded in the vehicle computer. When the vehicle has been loaded with timber (this subtask could also be done autonomously, but is not considered in this project) the operator activates *Path Tracking* mode, which means that the vehicle autonomously drives along the recorded path to the transportation road. In this paper, we describe the hardware and software developed and implemented for a pilot study on a Valmet 830 forest machine (forwarder) supplied by our industrial partner Komatsu Forest AB. The presented project started January 2003 and will end by December 2005. A continuation of the project is planned. Section 2 gives an overview of the developed hardware and software. The general design ideas behind a developed software system for sensor interfacing and actuator control are briefly described in Sect. 3. A novel algorithm for path tracking is described in Sect. 4. General experiences and directions for the future work finalizes the paper in Sect. 5.

2 System Overview

A block diagram of the developed system design is shown in Fig. 1. The major building blocks are described in this section. The system runs on two computers: the one placed on the vehicle is responsible for hardware interfacing and low-level data processing, such as data fusion in an occupancy grid. This computer communicates by a regular wireless local area network (WLAN) with the operator computer running the high level path tracking algorithms and the user interface. The two computers run with Windows XP at present, although it is designed so it can be moved to other OS:es, for instance UNIX or LINUX. This operating system independence comes from the choice of implementation language, Java and Matlab, as programs written in these languages are easily moved between different platforms. However, certain low-level drivers in C++ will have to be converted. The Java version used has varied from 1.1.8 up to 1.5, with little or no problems. The upgrad-

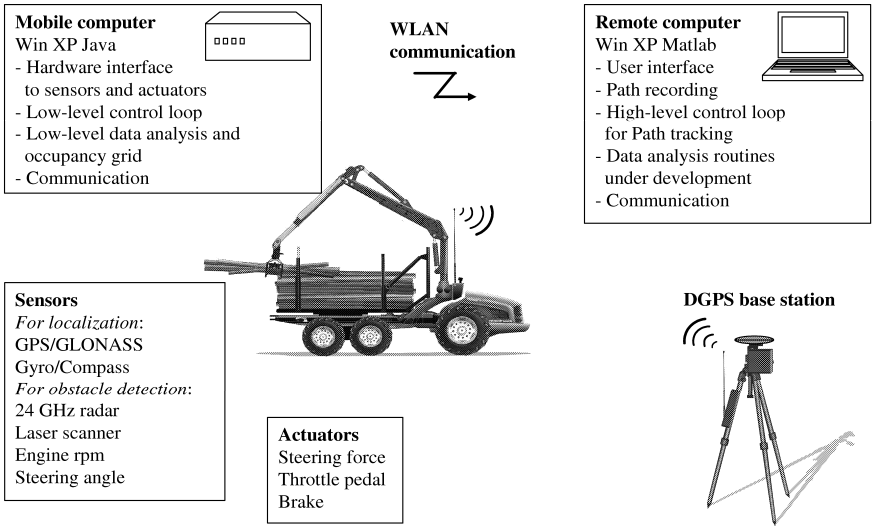


Fig. 1. Overview of the architecture of the developed system. The high- and low-level parts are split between two computers. The shown forest machine is a vision of what a future autonomous vehicle would look like.

ing was needed partly for incorporating new features in Java, and partly to overcome compatibility problems with Matlab. In general, low-level processing and communication is implemented in Java, and high-level processing in Matlab. However, during development the Matlab environment has been used also for typical low-level operations, such as Kalman filtering. With a top-level cycle time of around 100 ms, it has not presented any problems, and it simplifies the research work considerably. In a final productified version most processing should be implemented on the mobile computer, with the operator computer in charge of the user interface only.

2.1 Communication

As mentioned in the previous section, the modules of the system may reside on different computers. The communication routines take care of the data routing, and make the actual location of each module transparent to the other modules. Communication between separate computers is done by Ethernet network, either directly through a cable or via a Wireless Local Area Network (WLAN). The WLAN equipment is the standard 10-to-54 Mbps hardware used for offices and homes. The WLAN is used for controlling the vehicle, but since the communication handling is transparent to the system, debug and

in-office tests can be done by either a cable or direct communication within the computer.

The network communication uses datagrams (by the Internet UDP protocol), i.e. small packets of data transmitted with no control over their arrival, and therefore no acknowledge of received packets is obtained. The amount of data that travels through the network is very small, currently the largest packets are about 1200 bytes, and are transmitted every 200 ms. Most packets contain only a few bytes of payload, usually one reading of a sensor. A conservative estimate on the communication bandwidth needed is about 200 Kbps. (20 different types of packets, 100 bytes per packet, 100 ms. period). This amounts to 2% of the bandwidth of a 10 Mbps. link. The delay in the network is difficult to measure, since the two participating computers usually do not have synchronized clocks, but the estimate from preliminary tests is of less than 10 ms.

2.2 Sensors and Actuators

Sensors are primarily required for localization of the vehicle and obstacle detection. The performance of various sensor types and analysis algorithms is closely tied to the physical vehicle, on which the sensors are mounted. This means that the approach with multiple target machines during development (further described in Sect. 2.3) is of limited value for evaluation and development of sensor hardware/software. E.g., the range of sensors is normally fixed, and can not be scaled up in the same way as path tracking for example.

One major sensor type for obstacle detection on the forest machine is radar. The advantage of a radar, compared to sensors based on ultrasound or light, is the ability to view obstacles reliably during bad weather conditions, like snow, fog or rain. One of the radar types used in the project is a pair of the Sequential Lobing Radar C1, produced by Tyco - M/A-COM, USA (primarily for the automotive industry). This radar works with 24 GHz frequency and is based on the monopulse theory [10]. Target range and bearing can be obtained by transmitting and receiving pulses. The range is estimated by matching the received pulse to an internally time-delayed one. An estimation of target-bearing can be obtained by using a receiver antenna with switchable lobe characteristics.

The occupancy grid is 2-dimensional (100x100 cells) and move along with the vehicle. Although the system is working under a flat-ground assumption, the vehicle and range sensors are not assumed to be parallel to the flat ground. Their poses are fully 3-dimensional with 6 degrees of freedom, and the subsystem is designed to handle any pose set to a sensor. Although the pose of each range sensor has 6 degrees of freedom, a range sensor is approximated to have a 2-dimensional sensing plane, aligned with the xy-plane of its pose, where the x-axis equals the line-of-sight.

An RTK DGPS (Real-Time Kinematics Differential GPS) satellite navigation system, Javad Maxor, is the primary sensor for the vehicle's position. A

stationary GPS receiver is connected by a radio link to a mobile GPS receiver (see Fig. 1). Correction signals for timing, ionospheric and tropospheric errors are transmitted by radio from the stationary to the mobile GPS, resulting in a centimeter accuracy under ideal conditions. The Javad receiver is capable of receiving signals from both American GPS system and Russian GLONASS system. While providing a lower accuracy than the GPS, GLONASS provides important backup, especially at high latitudes (64 degrees north), at which the work has been conducted. A second satellite receiver with the same accuracy enables computation of heading, by estimating the direction of the line between the two receiving antennas. The heading accuracy is better than 0.5 degree when both receivers operate in full Real-Time Kinematics mode. Under worse conditions, the heading and position is computed by odometry through wheel encoders or speed and turning angle sensors. The fusion of GPS and odometry sensors are done in a binary fashion based on status information from the GPS system.

The forest machine is entirely controlled via an industrial communications bus, a CAN bus. The forest machine, a Valmet 830 as shown in Fig. 2, is equipped with an articulated joint for steering. We have implemented a simple proportional integrating (PI) controller that takes care of controlling the steering angle by controlling the joint.

2.3 Development Strategy

Testing algorithms on the full-size forest machine is both impractical and inefficient. Therefore, the work has been conducted on four different target machines, each with increased complexity. As shown in Fig 2, the same main program can control any of the four target machines through a software switch board. Likewise, sensor data passes from the target machine to the main program. In this way, high-level routines like path tracking are easily developed and implemented by the use of a simple simulator [11]. The simulator implements no sophisticated sensor models, and has a simplified kinematics model for propulsion, but serves very well its purpose for debugging and testing the path-tracking algorithm described in Sect. 4. The user interface is also easily developed using the simulator as the target machine. The infrastructure for sensors or actuators (see Sect. 3), and the modules for communication between the two main computers are most conveniently developed on the small-size Pioneer AT2 robot [12]. Various types of sensors are also evaluated on this target machine. The third target machine, the Smart Pioneer AT2, has a dynamics (software determined) that more closely mimics a real forest machine, and is used for more realistic tests of the routines for turning, speed setting, and path tracking. In the current phase of the project, the system is moved to the real forest machine, and the routines for vehicle control are fine-tuned and tested. Also, reliable sensor tests are only possible using this final target machine.

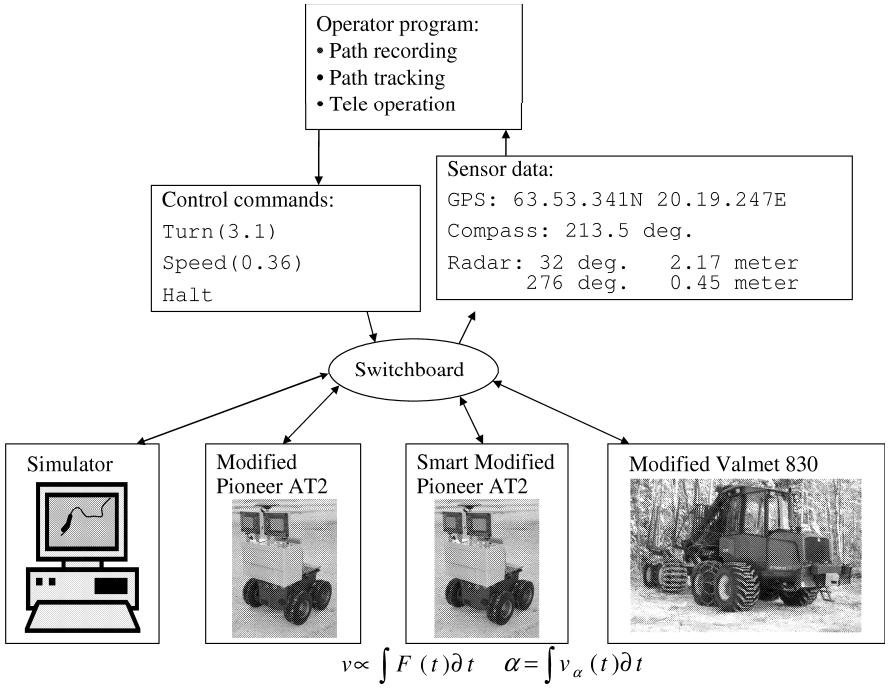


Fig. 2. The work has been conducted on four different target machines, each with increased complexity. This approach greatly simplifies the research and development of both hardware and software.

3 Infrastructure for Software Development

As part of the development work, a general framework for development of software for robots and autonomous vehicles has been constructed. A modular structure was designed with a set of requirements in mind:

- Not having to change any code in the system when replacing one sensor (e.g. a speed sensor that gets data from the wheel encoders) by another sensor (e.g. a speed sensor that gets its data from a GPS receiver).
- Not having to change any code when using different vehicles.
- Operating the system on one or more computers, connected by network.
- The analysis, design, and implementation should be object-oriented, to suit the modular design.

The system is made up of a number of software modules, some of which also represent real hardware devices. Each module can be independently loaded into the running system, and be logically connected to other modules. There are currently six major categories of modules:

Sensors (represent hardware units that deliver sensor data, e.g. speed, heading, position). Actuators - represent hardware units that control external equipment, e.g. throttle, steering angle, and brakes. Vehicle (several implementations of real and virtual vehicles. See Fig. 2). Control Panels (present data on the screen, or allow the vehicle to be manually controlled). Controllers (process sensor data and compute control signals for actuators). Proxies and Servers (facilitate transfer of sensor data and control commands over a network).

Each module is implemented as one or more classes. The modules are connected primarily by an event-driven system. There is no central control loop running. Instead the system reacts to changes in its environment. For example, a sensor signal arrives and sets up a series of method calls that ultimately leads to a change in the state of the system. Some sensor signals might just update an internal map of the surroundings, while others might stop the vehicle immediately. Other events are timed events, i.e. some action is performed repetitiously. Operator input for pure tele-operation of the vehicle is handled in the same way: a push of a button in the GUI (Graphical User Interface) or a real joystick sets up a chain of events. The Matlab program responsible for path tracking runs on another computer, and is not part of this event-driven system. Instead it polls the sensor system, in a traditional fashion, in its main control loop.

A module usually executes in a separate thread, i.e. all the modules run in parallel. Most sensing, control, user interface, and behavior are defined in modules. They communicate with other modules through connections, either by sending commands directly to another module or by listening on other modules. They also have other properties that depend on their actual type, for instance update interval for sensors, and network addresses for servers and proxies.

Every module has an associated configuration file with properties that control the module's behavior. For a sensor, this file may contain e.g. pose, network address, and filter constants. Most of these properties are set once and for all, while others are changed either by the user or by the module itself. The module can also save the changes so they take effect upon the next time the system is run. One example would be an experiment to find the most appropriate filter constant for a specific sensor; when a suitable value is found the sensor can store it in its configuration file. The next time the sensor is run, it automatically uses the saved value. The forest machine system contains more than 60 different initialization files, and to facilitate changes, as well as for providing an overview, a graphical configuration manager has been developed. The configuration manager gives the user a graphical picture of how every module is connected to other modules, and can also be used to modify individual properties for a module.

4 Path Tracking and Obstacle Avoidance

To navigate safely through the forest, a new path-tracking algorithm named *Follow-the-Past* has been developed. Traditional algorithms like Follow-the-Carrot [1] and Pure-Pursuit [2] use position information only, and sometimes run into problems that can be avoided by taking into account additional recorded information from a human driver. If the vehicle deviates from the recorded path, for example as a result of avoiding an obstacle, or because of noise in the positioning sensors, the Follow-the-Past algorithm steers like the driver, plus an additional angle, based on the distance to the path. The algorithm is described in the following section. More details and test results are found in [6] and [7].

4.1 Follow-the-Past Algorithm

While manually driving along the path, the orientation and steering angles are recorded together with the position at every moment. The recorded position (x', y'), the recorded orientation θ' and the recorded steering angle ϕ' are used by three independent behaviors:

ϕ_β: Turn towards the recorded orientation θ'
ϕ_γ: Mimic the recorded steering angle ϕ'
ϕ_α: Move towards the path

Each behavior suggests a steering angle and is reactive, i.e. operates on the current input values; orientation, steering angle, and shortest distance to the path. ϕ_α uses recorded positions (x', y') and actual position (x, y) as inputs. ϕ_β uses recorded orientation θ' and actual orientation θ as inputs. ϕ_γ uses the recorded steering angle ϕ' as input. The three behaviors are fused into one action, the commanded steering angle ϕ_t, as shown in Fig. 3.

The three independent behaviors ϕ_α, ϕ_β, and ϕ_γ operate in the following fashion:

ϕ_β: Turn towards the recorded orientation

The angle θ' is defined as the recorded orientation at the closest point on the recorded path. This point is called the path point. ϕ_β is computed as the difference between the current orientation θ and the recorded orientation θ':

$$\phi_\beta = \theta' - \theta. \tag{1}$$

ϕ_γ: Mimic the recorded steering angle

This behavior simply returns the recorded steering angle ϕ' at the path point:

$$\phi_\gamma = \phi'. \tag{2}$$

By using the recorded steering angle, the curvature of the path is automatically included in the final steering command. This is a great advantage compared to methods like Pure-Pursuit [2] and Follow-the-Carrot [1].

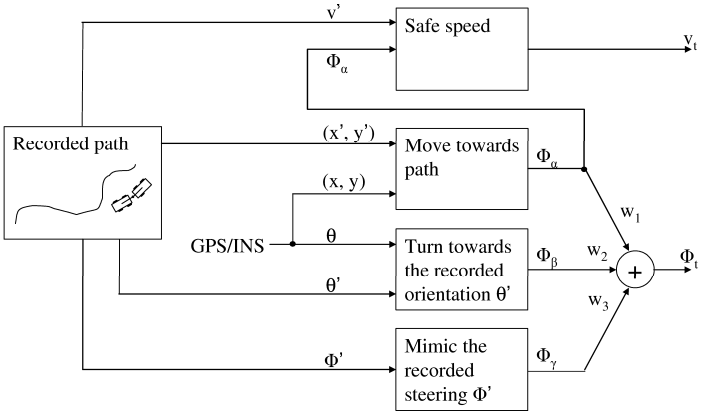


Fig. 3. Path tracking with reactive control of steering angle ϕ_t.

ϕ_α: Move towards the path

This behavior is responsible for bringing the vehicle back to the path, if the vehicle deviates for some reason from the path. Such a deviation can be caused by noise in the position signal, or by the obstacle-avoidance system. ϕ_α can be computed in many ways, e.g. by the following algorithm (refer to Fig. 4:

1. Determine the closest point on the recorded path (denoted Path Point).
2. Compute a Look Ahead Point at a Look Ahead Distance ℓ from the Path Point, in a direction δ, defined as the sum of the recorded orientation θ' and the recorded steering angle ϕ' at the Path Point, i.e.:$\delta = \phi' + \theta'$.
3. Calculate a Look Ahead Angle ψ, defined as the polar angular coordinate for the vector between the vehicle's current coordinates and the Look Ahead Point.
4. Compute ϕ_α as the difference between ψ and the angle δ., i.e.: $\phi_\alpha = \psi - \delta$.

An alternative method to compute ϕ_α can be found in [6].

Command Fusion

The three behaviors ϕ_α, ϕ_β, and ϕ_γ all return a suggested steering angle, aiming at fulfilling the goals of the respective behaviors. These three values are fused into one value ϕ_t by a weighted addition as shown in Fig. 3. In our tests, all weights have been set to 1. I.e.:

$$\phi_t = \phi_\beta + \phi_\gamma + \phi_\alpha. \tag{3}$$

The expression for the fused ϕ_t is:

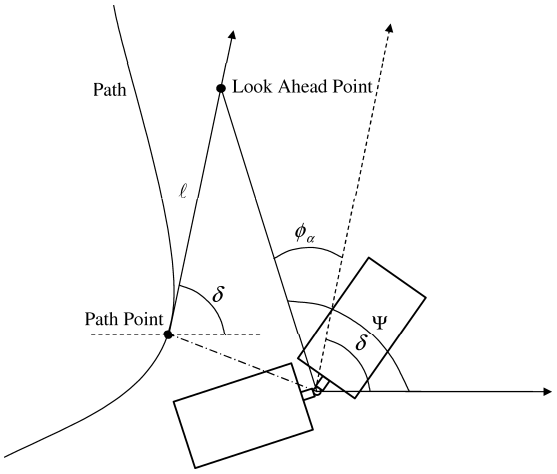


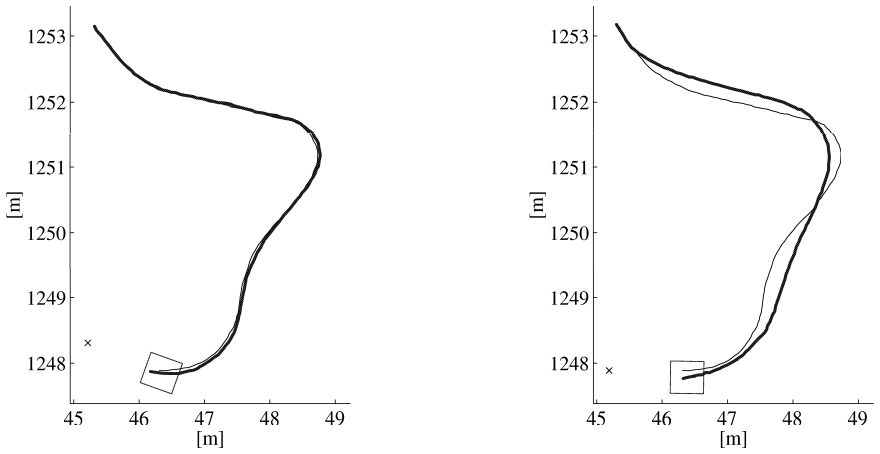
Fig. 4. Calculation of ϕ_α, which is responsible for keeping the vehicle on the track.

$$\begin{aligned}\phi_t &= \psi - \delta + \phi_\beta + \phi_\gamma \\ &= \psi - (\phi' + \theta') + (\theta' - \theta) + \phi' \\ &= \psi - \theta.\end{aligned} \tag{4}$$

Testing and Results

The developed algorithm has been tested both in a simulator for forest machines [11] and on a Pioneer robot. In this report we present only a test done with the Pioneer robot and compare the results to an implementation of the Pure-Pursuit [2] method. In this test, a Look-Ahead Distance $\ell = 1.2$ meter is used. Figure 5(a) shows results for path tracking with the Follow-the-Past method. The vehicle (thick line) is capable of following a recorded path (thin line) with almost no deviation from the path. As a reference, Fig. 5(b) shows how the vehicle behaves when using the Pure-Pursuit method under the same conditions. This path has some sharp turns, which makes it difficult for both Follow-the-Carrot and Pure-Pursuit, as they tend to "cut corners" instead of following a highly curved path. Under normal conditions, this problem is avoided by the Follow-the-Past algorithm.

To function in the forest machine application, the path-tracking behavior has been combined with VFH+ [13] for obstacle avoidance. The HIMM grid used in the original VFH+ algorithm is replaced by an occupancy grid with Bayesian updating.



(a) Follow-the-Past (b) Pure-Pursuit

Fig. 5. Comparison between a) the Follow-the-Past and b) the Pure-Pursuit path tracking algorithms. Follow-the-Past is able to follow the path almost perfectly, while Pure-Pursuit tends to "cut corners". The examples above are from tests with the Pioneer robot.

5 Status, Experiences and Future Work

The system has been successfully implemented on the simulator and on the Pioneer robot. The developed algorithm for path tracking performs very well, and the general tools for robot software architectures have been shown to be both powerful and flexible. Sensor tests and system adjustments for the forest machine are in progress and planned to be completed during 2005.

The continuation of the project will deal with the sensing problems specific for forest environments. Techniques to distinguish obstacles from ground in uneven and hilly environment have to be developed. The specific problem with detection of human beings is also of the highest priority for a future productification of an unmanned forest machine. The future work will also involve development of ways to achieve accurate localization and estimation of heading without expensive GPS equipment.

Acknowledgements

This work was financed by The Kempe Foundations, VINNOVA, Land Systems Hägglunds, Carl Tryggers stiftelse, LKAB and Komatsu Forest AB. We acknowledge their support gratefully. The authors would also like to thank Urban Sandström for his implementation of the occupancy grid and Kalle Prorok for his work with the radar sensors.

References

1. M. J. Barton. *Controller Development and Implementation for Path Planning and Following in an Autonomous Urban Vehicle*. Undergraduate thesis, University of Sydney, Nov. 2001.
2. R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1992.
3. F. Georgsson, T. Hellström, T. Johansson, K. Prorok, O. Ringdahl, and U. Sandström. Development of an autonomous path tracking forest machine - a status report -. Technical Report UMINF 05.08, Department of Computing Science, Umeå University, 2005.
4. U. Hallonborg. Förarlösa skogsmaskiner kan bli lönsamma. *Skogforsk RESULTAT*, (9), 2003.
5. T. Hellström. Autonomous navigation for forest machines. Technical Report UMINF 02.13, Department of Computing Science, Umeå University, 2002.
6. T. Hellström and O. Ringdahl. Follow the past - a path tracking algorithm for autonomous forest vehicles. Technical Report UMINF 04.11, Department of Computing Science, Umeå University, 2004.
7. T. Hellström and O. Ringdahl. Autonomous path tracking using recorded orientation and steering commands. In *Proceedings of Towards Autonomous Robotic Systems 2005*, Imperial College London, England, 2005.
8. R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.
9. H. Mäkelä and K. Koskinen. Navigation of outdoor mobile robots using dead reckoning and visually detected landmarks. In *ICAR'91 Fifth International Conference on Advanced Robotics*, pages 1151 – 1156, June 1991.
10. P. Z. Peebles Jr. *Radar Principles*. John Wiley & Sons, 1998.
11. O. Ringdahl. Path tracking and obstacle avoidance for forest machines. Master's Thesis UMNAD 454/03, Department of Computing Science, Umeå University, 2003.
12. A. Robotics. Robots, AGV's & robotic sensing. *http://www.activmedia.com/*, 27 Jan. 2005.
13. I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. *IEEE Int. Conf. on Robotics and Automation*, pages 1572–1577, May 1998.
14. K. Westlund and T. Hellström. Requirements and system design for a robot performing selective cleaning in young forest stands. *to be published in Journal of Terramechanics*, 2005.