Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Electronics in Agriculture 44 (2004) 1–19
Computers and electronics in agriculture
www.elsevier.com/locate/compag

Development of a master–slave robot system for farm operations

Noboru Noguchi [a,*], Jeff Will [b], John Reid [c], Qin Zhang [d]

[a] *Agricultural Vehicle System Engineering, Graduate School of Agriculture, Hokkaido University, Kita-9, Nishi-9, Kita-ku, Sapporo 060-8589, Japan*
[b] *Electrical and Computer Engineering, Valparaiso University, Valparaiso, IN 46383-6493, USA*
[c] *John Deere Technology Center-Moline, One John Deere Place, Moline, IL 61265-8098, USA*
[d] *Department of Agricultural Engineering, University of Illinois, Urbana, IL 61801, USA*

Received 29 July 2003; received in revised form 17 December 2003; accepted 29 January 2004

Abstract

The primary objective of this study was to develop a control system for autonomous mobile robots used in farm operations. To accomplish this objective, it was necessary to develop mobile robots having minimal centralized control. This paper focuses on the development of two basic motion control algorithms, namely a GOTO algorithm and a FOLLOW algorithm, for use in a master–slave multi-robot system. These two robot motion control algorithms would have wide applicability in farm operations. The GOTO algorithm can be applied when the master wants the slave to go to a specific place, a certain distance from the current operational position. Safety is one important issue in controlling the master–slave system because the master and the slave move independently. In this GOTO algorithm, the slave was set to slow-down to allow the master pass the slave safely in case there was a potential collision due to path overlap in the field. The slave was also able to change its path to avoid a crash based on the collaborative GOTO algorithm. The FOLLOW algorithm allows for a more cooperative way to guide the slave to follow the master at a predetermined relative distance and angle, regardless of the traveling speed and direction. This FOLLOW algorithm incorporated a nonlinear sliding mode controller to provide a robust control for the slave. The validation tests indicated that the sliding mode controller could provide a better performance in terms of both lateral offset and spacing controls compared than using a conventional PD controller.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Agricultural mobile robot; Multiple robot system; Obstacle avoidance; Robust control

* Corresponding author. Tel.: +81-11-706-2568; fax: +81-11-706-2568.
E-mail address: noguchi@bpe.agr.hokudai.ac.jp (N. Noguchi).

0168-1699/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.compag.2004.01.006

1. Introduction

Increases in agricultural productivity have contributed to a decline in the number of people in the skilled machine operator work force, particularly in Japan. The age of the overall agricultural workforce is increasing, indicating that this profession is not being adopted by younger generations. Therefore, the development of automated or autonomous agricultural equipment is considered to be of the utmost commercial and societal importance (Noguchi and Reid, 2000a). In fact, numerous studies have already dealt with hardware techniques (e.g., spatial position-sensing systems and steering control systems) for following a predetermined path (Choi et al., 1990; Erbach et al., 1991; Smith et al., 1985, 1987; Tillett, 1991) towards the development of autonomous machinery. Some researchers used the GPS signal for positioning for vehicle guidance. Researchers at Stanford University (O'Connor et al., 1995, 1996; Bell, 1999) have successfully developed a four-antenna carrier-phase GPS system for guiding a John Deere 7800 tractor on prescribed straight row courses with headland turns. Researchers at the University of Illinois (Stombaugh et al., 1998) utilized a real-time kinematic (RTK) GPS for guidance of a 2WD Case 7720 tractor. In order to eliminate lag in the system responses the GPS was mounted in front of the front wheels on a mast extending to a height above the cab. Vehicle responses show that the lateral position error at 4.5 m/s was within 16 cm. Benson et al. (1998) utilized a geomagnetic direction sensor (GDS) and a GPS for vehicle guidance along with straight direction courses. They were able to combine the GDS with a medium accuracy GPS system (20 cm) and track a straight line with an average error less than 1 cm. Noguchi et al. (2001) have completed a field robot which uses a RTK GPS and an inertial measurement unit to provide navigation information. The robot has successfully carried out the tillage, planting, cultivating, and spraying of a soybean field. In addition, the robot has demonstrated autonomous navigation from a shed to a field. The root mean square (RMS) lateral error of the robot's operation was less than 5 cm, which is an improvement over skilled-human operation. During curved crop-row operations, the robot was able to navigate without causing plant damage.

However, when a robot is used in an open site (which is the case in most practical operations) some type of monitoring system is required to ensure safety (Reid et al., 2000). Therefore, at least one human manager is required to supervise the robot during operation. This means that for every human, one robot can work in a field. This is equal in efficiency to a single human driving a tractor (Noguchi et al., 2002). In order to improve efficiency, the next step for agricultural robot development is to increase the number of robots in simultaneous operation by developing a multiple robot system.

The objective of this study is to develop a control system for autonomous mobile robots that perform farm operations. To accomplish this objective, it is necessary to develop mobile robots with minimal centralized control. The solution to this decentralization problem is to give more autonomy to the robots to allow them to cope with unexpected events and obstacles, such as other vehicles, having an inaccurate environment model, and similar challenges. Such a high level of autonomy can be achieved using advanced sensor-based capabilities for localization and obstacle detection, as well as local communication and coordination for joint planning and deliberation between the robots.

In this study, two basic motion control algorithms, a GOTO and a FOLLOW algorithm, were developed for use in a master–slave structure within a multi-robot system. The GOTO

algorithm consists of a behavior where the slave follows a path from the current point to another, predetermined one, following instructions such as 'go to the refueling station'. The FOLLOW algorithm puts the slave into a state such that it mimics the navigation of the master, but with a predetermined backward and lateral offset. An example instruction would be 'follow me 5 m behind and 2 m to the left'.

In the GOTO algorithm, the issue of safety was one of the primary problems of the master–slave structure, because the master and the slave move independently. Thus, one of the robots may have to take action to avoid a collision with the other by either slowing down or by generating an intermediate pathway. The present paper focuses mainly on this issue of detection and avoidance. This paper also focuses on the development of steering and speed controllers for the FOLLOW algorithm of the slave. A sliding mode nonlinear controller that is a robust control technique is implemented in the algorithm. Hendrick et al. (1994) and Peng and Tomizuka (1993) studied a nonlinear control approach taken in the California PATH control program, which used a lateral offset and a longitudinal 'Platoon' control program for an automated highway system. The approach provided a smoothed form of the sliding mode control and showed satisfactory control performance.

2. Concept of a master–slave system

2.1. Multi-robot structure

There are many types of field operations that use two vehicles. When harvesting hay on grassland, it is customary for one dump truck and one tractor with a hayfork to be used. When harvesting corn, a combination of one wagon and one harvester is generally adopted. Therefore, a master–slave system, which uses two vehicles, can be very useful in actual field operations. Fig. 1 depicts the concept of the master–slave structure used in this study. The so-called master vehicle performs the functions of making decisions and sending commands to the slave vehicle. The slave vehicle follows the master and broadcasts its own status by sending information about its location, orientation, and operating conditions.

Two types of basic operations have been proposed in this study, as seen in Fig. 1. One is the GOTO algorithm and the other is the FOLLOW algorithm. The GOTO algorithm can be applied when the master wants the slave to go to a specific place, a certain distance from the current operational position. This type of cooperative work can be adopted, for example, during hay harvesting. If the master were a tractor with a hayfork and the slave were a truck, the GOTO algorithm could be used to make the slave transfer to an operational position to deliver the hay to a barn. On the other hand, the FOLLOW algorithm uses a more common, cooperative style of interaction. The slave follows the master at a given relative distance, d, and a given angle, γ. This cooperative work produces large benefits in terms of increased overall productivity, even when using two identical machines. At times, harvesting requires two vehicles, e.g., a harvester and a wagon. The FOLLOW algorithm is suitable for this type of field management. To use the FOLLOW algorithm, the master broadcasts an initial command to inform the slave to follow the master at a certain offset and a certain relative angle. Table 1 shows the information formats for communication between master and slave. In total, eight kinds of formatted strings are used in these com-

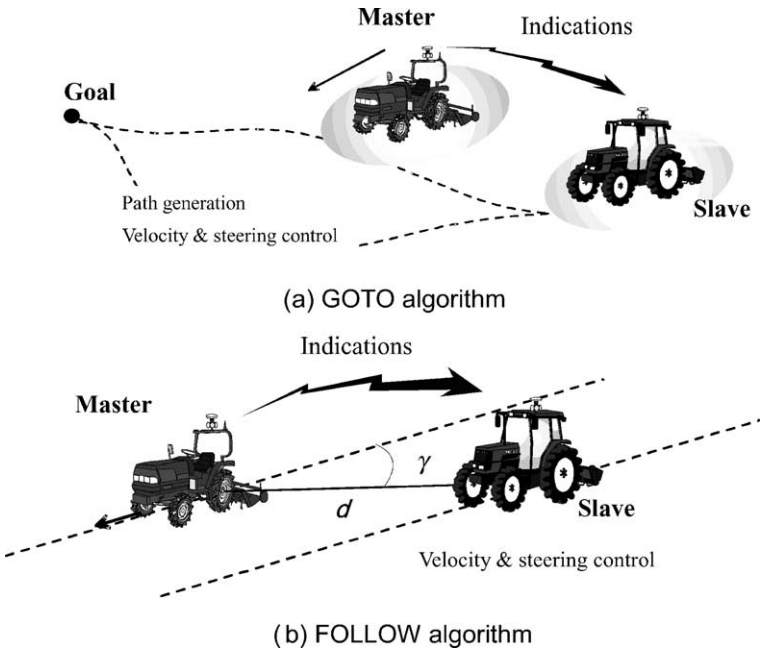

Fig. 1. GOTO and FOLLOW algorithms for a master–slave robot system. The GOTO algorithm can be adopted for harvesting hay. On the other hand, the FOLLOW algorithm results in a more common, cooperative style of work. The slave follows the master from a given relative distance, d, and relative angle, γ.

munications. A header string indicates the type of information, either Command, Request, or Status.

2.2. GOTO algorithm

A flowchart of the GOTO algorithm is shown in Fig. 2. As mentioned in the description of the master–slave structure, a control algorithm for the slave is essential when using the GOTO algorithm because the master sends only a single command to the slave. Fig. 2 shows the control algorithm of the slave. The proposed algorithm assumes that the master continuously broadcasts its state information, consisting of the current location, orientation, and operating condition. In the algorithm, the slave first obtains the GOTO command from the maser. Second, the slave generates the pathway from its present location to the target location based on a third-order spline function, which has already been developed by Noguchi et al. (2000b). The GOTO algorithm permits the slave to take almost any pathway to reach a destination except the pathway on which the master is located. After generating a satisfactory pathway, the slave starts to move in accordance with this path. While it travels on the path, the slave obtains its present state from its navigation sensor and sends the state update to the master at a frequency of at least 2 Hz. The master simultaneously broadcasts its state to the slave at the same update rate. It is essential for the slave being able to detect and avoid the master. If the slave senses any danger of colliding with the master, the slave

Table 1
Specifications and formats for communication in the master–slave system

Communication	Type	Header ID	Parameters	Direction
Goto	Command	CG	Time Latitude Longitude Heading	Master → slave
	Status	SG	Time Latitude Longitude Heading	Slave → master
Follow	Command	CF	Time Distance Offset angle	Master → slave
	Status	SF	Time Distance Offset angle	Slave → master
Stop	Command Status	CS SS	Time Time	Master → slave Slave → master
Situation	Command Status	R S	Time Time Latitude Longitude Heading	Master → slave Slave → master

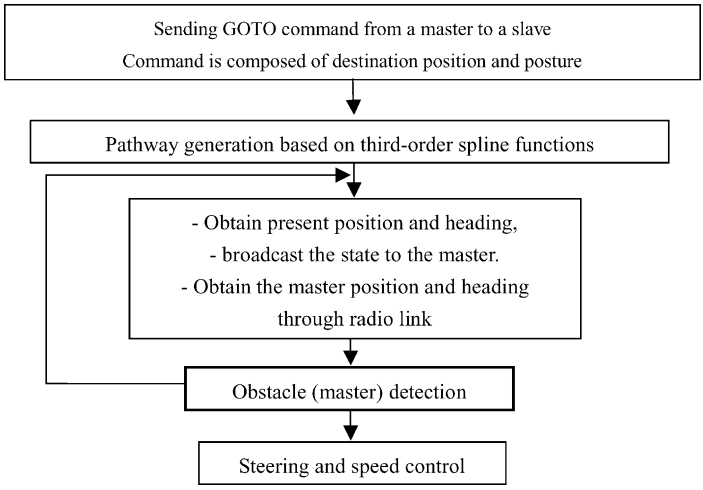

Fig. 2. Control flowchart of the slave following the GOTO algorithm. The slave can generate a path to a destination. The avoidance of the master by the slave is plugged into the control functions.

has to either generate a different pathway or slow down to avoid a crash. In this type of situation, only a master that is moving has to be considered, because static obstacles can be taken into account before the slave generates a pathway.

The master broadcasts commands and its own state to the slave, but does not need to be mindful of the slave's behavior. Therefore it is only the slave that has functions to recognize the master's behavior and to take an action to avoid an impending crash, and the primary avoidance algorithm is incorporated into the control functions of the slave as shown in Fig. 2. The central idea of this avoidance algorithm is that the slave ascertains the action of the master, using the master's updated state information (specifically, its location) in each control step.

The slave receives updates of the master's location and velocity at a frequency of 2 Hz and thus recognizes the state of the master at a sufficient temporal resolution. In this research, a wireless local area network (LAN) and a PC that were used with the robots limited the robots' communication and control capabilities with a frequency rate of 2 Hz for both measurement and control updates. Obviously, increasing the rate of updates would support more accurate and timelier decisions by the slave. In fact, the control loop for the slave may be run at a much higher frequency than 2 Hz. While the informational loop for collision avoidance by the master is updated at the slower 2 Hz rate, because it is limited by networking and other constraints, local 'intraslave' communication is free of such constraints.

The state broadcasted by the master is defined as (x_m, y_m, ϕ_m), as shown in Fig. 3. In addition, the state of the slave is also defined as (x_s, y_s, ϕ_s) in the X–Y world coordinate system. Due to the fact that the danger of a collision can be determined by the distance between the master and the slave, the level of such danger is defined according to the risk index. The risk index can be calculated by a function of $(x_m, y_m, \phi_m, x_s, y_s)$.

$$\text{Risk index} = f\,(x_m, y_m, \phi_m, x_s, y_s) \tag{1}$$

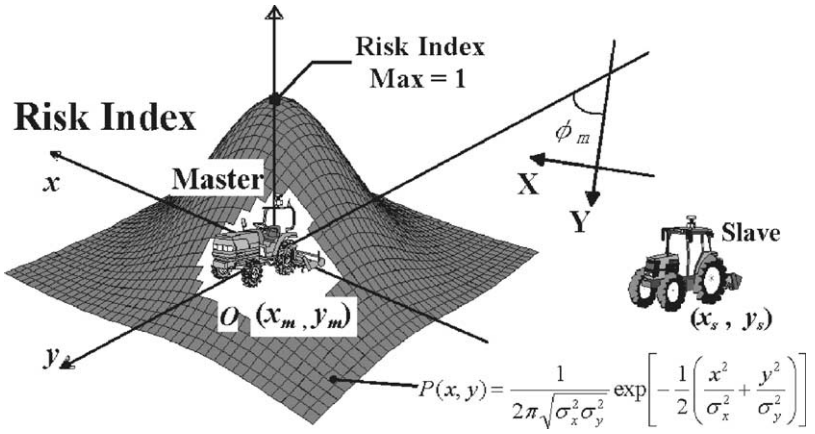

Fig. 3. Definition of risk index for crash. The two-dimensional normal distribution was adopted as the risk index function.

The position of the slave can be converted to the local coordinate system, or the origin of the master's present position, using the following equation:

$$\begin{bmatrix} X_s \\ Y_s \end{bmatrix} = \begin{bmatrix} \cos\phi_m & -\sin\phi_m \\ \sin\phi_m & \cos\phi_m \end{bmatrix} \begin{bmatrix} x_s - x_m \\ y_s - y_m \end{bmatrix} \tag{2}$$

In this paper, a two-dimensional normal distribution, shown in Eq. (3) was adopted as the risk index function in Eq. (4):

$$P(X_s, Y_s) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{1}{2}\left(\frac{X_s^2}{\sigma_x^2} + \frac{Y_s^2}{\sigma_y^2}\right)\right] \tag{3}$$

$$\text{Risk index} = \frac{P(X_s, Y_s)}{P(0, 0)} \tag{4}$$

The risk index is a function of X_s and Y_s, which is the slave's present position in the master's coordinate system. The slave itself can calculate the risk level of the collision with the master during each measurement step. The variables σ_x and σ_y are standard deviations for the two axes, latitude and longitude, of the master. These values are determined by the size of the master vehicle because the shape of the risk index is influenced by these two parameters. The slave uses this risk index to determine an appropriate action to avoid collision with the master.

Two actions are considered as preventative measures against collisions. One is speed reduction and the other is pathway correction. Which action must be taken depends on the type of field operation in progress. If the slave is conducting tillage or planting, pathway correction is generally not feasible. Therefore, the slave must choose to slow down to avoid the master. On the other hand, the slave can change pathways to reach a proper position for transporting hay on a meadow. In terms of work efficiency, this solution is much better than slowing down.

2.3. FOLLOW algorithm

The FOLLOW algorithm involves both lateral offset and spacing controls. The spacing control changes the engine speed or the position of the transmission, while a change in the steering angle is needed to control the lateral offset. This control algorithm adds following behavior to the functions of the slave. A sliding-mode control was employed for both the spacing control and the lateral offset control. Because the dynamics of an off-road vehicle are highly nonlinear and operate over a wide range, the sliding mode control approach was for developing the FOLLOW algorithm to stabilize the system while achieving insensitivity to external disturbances and parametric deviations. Fig. 4 shows a schematic diagram of the lateral control geometry. The controller was essentially based on a point-follower approach. The black dot in the figure indicates the point where the slave exists at the current time t. The quantities ε_s and ε_c were defined as a lateral deviation from the line that is parallel with the heading angle of the master and crosses the target point of the slave at time, t. The variable ε_c represents the lateral deviation from the center of gravity (COG) of the slave,

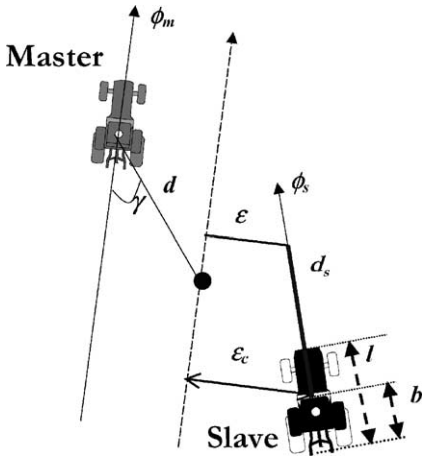

Fig. 4. Schematic diagram of the lateral controller geometry for the slave.

while ε_s represents the amount of deviation from point at a distance d_s ahead of the COG, expressed as:

$$\varepsilon_s = \varepsilon_c + d_s(\phi_c - \phi_m) \tag{5}$$

In Eq. (5), ϕ_s and ϕ_m are the angles in which the slave and the master are heading, respectively.

Next, we will define a function of the slave lateral error that we would like to make equal to zero as in the following equation:

$$S_\varepsilon = \dot{\varepsilon}_s + P_1\varepsilon_s + P_2\int_0^t \varepsilon_s dt \tag{6}$$

Differentiating Eq. (6) and presuming that we would like S_ε to approach zero so that ε_s will approach zero, we set the following:

$$\dot{S}_\varepsilon = -\lambda_1 S_\varepsilon \tag{7}$$

In addition, by assigning Eq. (7) to differentiated Eq. (6), the following equation is obtained:

$$\dot{\varepsilon}_s = \frac{-P_2\varepsilon_s - \lambda_1 S_\varepsilon}{P_1} \tag{8}$$

Finally, the steering angle δ can be expressed as,

$$\delta = \frac{l}{V_s(b + d_s)}\left(\frac{-P_2\varepsilon_s - \lambda_1 S_\varepsilon}{P_1}\right) \tag{9}$$

where l is the wheel-base of the slave, V_s is the slave speed, and b is the distance from the rear wheel to the COG. The variables P_1, P_2, and λ_1 must be tuned through computer simulation. This control algorithm requires obtaining the position and heading angle of the master. As mentioned above, this is accomplished via communication of status between the master and slave at a frequency of 2 Hz.

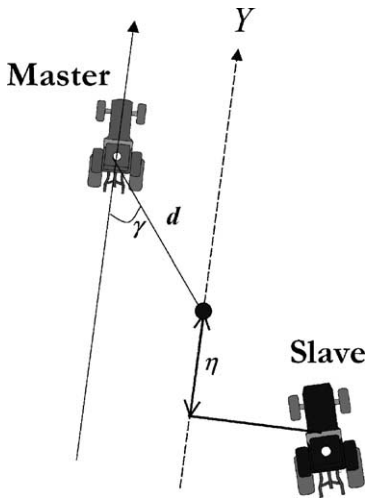

Fig. 5. Schematic diagram of the spacing controller geometry for the slave.

Fig. 5 shows a schematic diagram of the spacing control geometry. The slave controls the speed based on the spacing error of η. The control algorithm is the same as a lateral control algorithm using a sliding mode control. The sliding mode control for spacing can be defined as in Eq. (10) below:

$$S_\eta = \dot{\eta}_s + P_1\eta_s + P_2\int_0^t \eta_s dt \tag{10}$$

$$\dot{S}_\eta = -\lambda_2 S_\eta \tag{11}$$

By using Eqs. (10) and (11), the change in velocity ΔV_s is determined using Eq. (12) where Δt is the control interval of the slave speed.

$$\Delta V_s = (-q_1\dot{\eta} - q_2\eta - \lambda_2 S_\eta)\Delta t \tag{12}$$

Finally, the control algorithm can be expressed as follows:

$$V_s = V_m + \Delta V_s \tag{13}$$

In Eq. (13), V_m is the master speed coded in a status string from the master.

3. Simulation

3.1. Vehicle model

To verify the performance of the GOTO and FOLLOW algorithms that were developed, a computer simulation was conducted. A vehicle movement model was built for this sim-

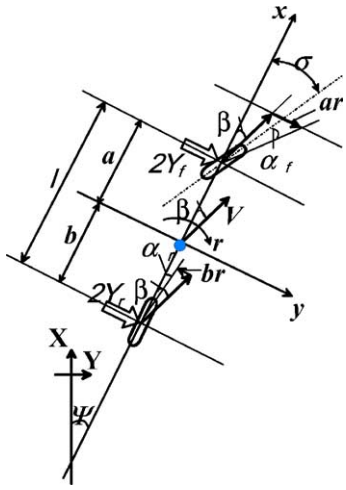

Fig. 6. Vehicle motion model for the computer simulation.

ulation. From Fig. 6, the basic vehicle movement can be explained using Eqs. (14) and (15).

$$I\frac{\mathrm{d}r}{\mathrm{d}t} = 2aY_{\mathrm{f}} - 2bY_r \tag{14}$$

$$mV(\frac{\mathrm{d}\beta}{\mathrm{d}t} + r) = 2Y_{\mathrm{f}} + 2Y_{\mathrm{r}} \tag{15}$$

In the equations above, the variables m and I are the mass and the inertial moment of the vehicle, respectively. The variable β is the lateral slip, the variable r is the yaw rate, the variable a is the distance from the front axis to the COG, and the variable b is the distance from the rear axis to the COG. The variables Y_{f} and Y_{r} are lateral forces on the front and rear tires.

$$Y_{\mathrm{f}} = -K_{\mathrm{f}}\alpha_{\mathrm{f}} \tag{16}$$

$$Y_{\mathrm{r}} = -K_{\mathrm{r}}\alpha_{\mathrm{r}} \tag{17}$$

In the equations above, K_{f} and K_{r} are cornering powers, and α_{f} and α_{r} are the slip angles of the front and the rear tires, which can be calculated using Eqs. (18) and (19).

$$\alpha_{\mathrm{f}} = \beta + \frac{ar}{V} - \delta \tag{18}$$

$$\alpha_{\mathrm{r}} = \beta - \frac{br}{V} \tag{19}$$

Therefore, the equations to be solved for the vehicle model are described as follows:

$$mV\frac{\mathrm{d}\beta}{\mathrm{d}t} + 2(K_{\mathrm{f}} + K_{\mathrm{r}})\beta + \left[mV + \frac{2}{V}(aK_{\mathrm{f}} - bK_{\mathrm{r}})\right]r = 2K_{\mathrm{f}}\delta \tag{20}$$

$$2(aK_f - bK_r)\beta + I\frac{dr}{dt} + \frac{2(aK_f - bK_r)}{V}r = 2aK_f\delta \quad (21)$$

To change the coordinate system of the trajectory to WGS84, Eqs. (22) and (23) were applied.

$$X_{k+1} = X_k + dt\, V \sin(\phi_k + \beta) \quad (22)$$

$$Y_{k+1} = Y_k + dt\, V \cos(\phi_k + \beta) \quad (23)$$

$$\phi_{k+1} = \phi_k + dt\, r \quad (24)$$

The differential equations from Eqs. (16) to (21) can be calculated using the fourth-order Runge–Kutta method. The steering controller for both the master and the slave in the GOTO algorithm use a proportional-differential (PD) method that factors in both heading error and offset (Noguchi and Reid, 2000a). In addition, the PD controller was used for controlling the master's steering angle in the FOLLOW algorithm as well. The desired steering angle, δ, was computed assuming a proportional controller for both a heading error and an offset as follows:

$$\delta = k_\phi \Delta\phi + k_p \varepsilon \quad (25)$$

In Eq. (25) above, the control gains k_ϕ and k_p were determined by a preliminary experiment.

3.2. Results and discussion

3.2.1. GOTO algorithm

Fig. 7 shows the setup for a computer simulation of confirming the validity of the simulation model built in the previous session, and recognizing a static obstacle. In addition, the slave's recognition of the master and calculation of the risk index were tested. As shown in Fig. 7, there was a target pathway for the slave, and the stationary master robot was located on the target pathway. The speed of the slave was set at 1.0 m/s. The robot guidance algorithm was based on a map-based guidance system developed by Noguchi and Reid (2000a). Fig. 8 shows the results of the simulation. The parameters σ_x and σ_y, which determined the shape of the risk index, were 10 and 20 m, respectively. In the simulation, the slave moved along the predetermined pathway, and the risk index increased as the slave moved closer to the master. Finally, when the distance between the master and the slave approached zero, the risk index reached 1.0. This indicated that the slave collided with the master. It was clear that the simulation model of the robot worked properly and also that the slave could obtain the risk index during travel. Two actions for the slave to take to avoid a crash with the master had been developed. First, the algorithm to decrease the slave's speed in response to the risk index was tested. The following rule for slowing down was chosen for the simulation:

$$V_s = -1.6 \times \text{Risk index} + 1 \quad (26)$$

Here, the order of Eq. (26) and two coefficients were determined by the simulation. More than 30 runs representing various situations, including changes in the positions and the velocities of both robots, were conducted to determine the speed of the slave. The rule to

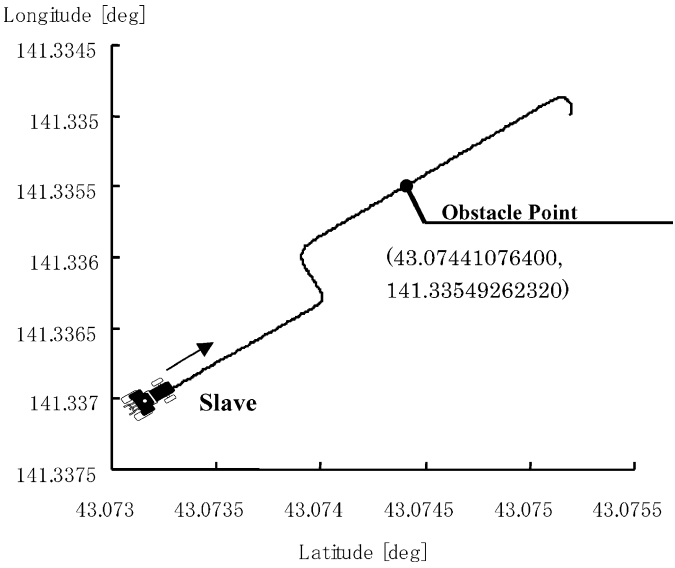

Fig. 7. Setup for a computer simulation of recognizing a static obstacle. There is a target pathway for the slave, and the master robot, located on the target pathway, is stationary.

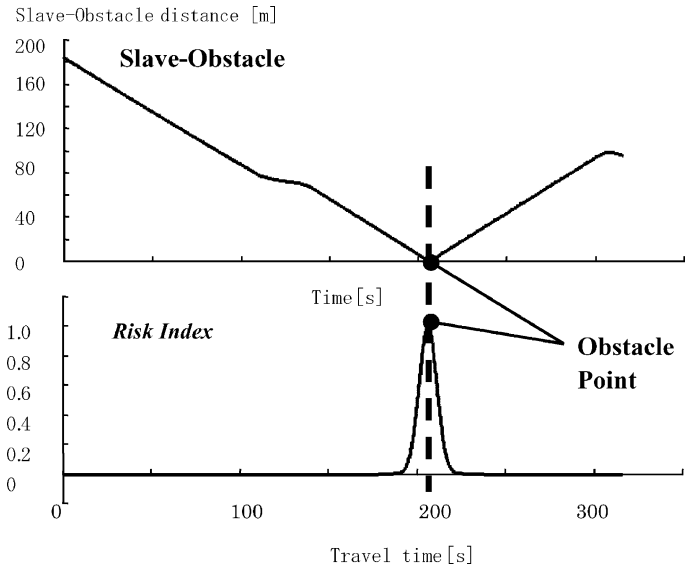

Fig. 8. Relative distance and risk index for a static obstacle. The slave traveled along the predetermined pathway and the risk index increased as it moved toward the master.

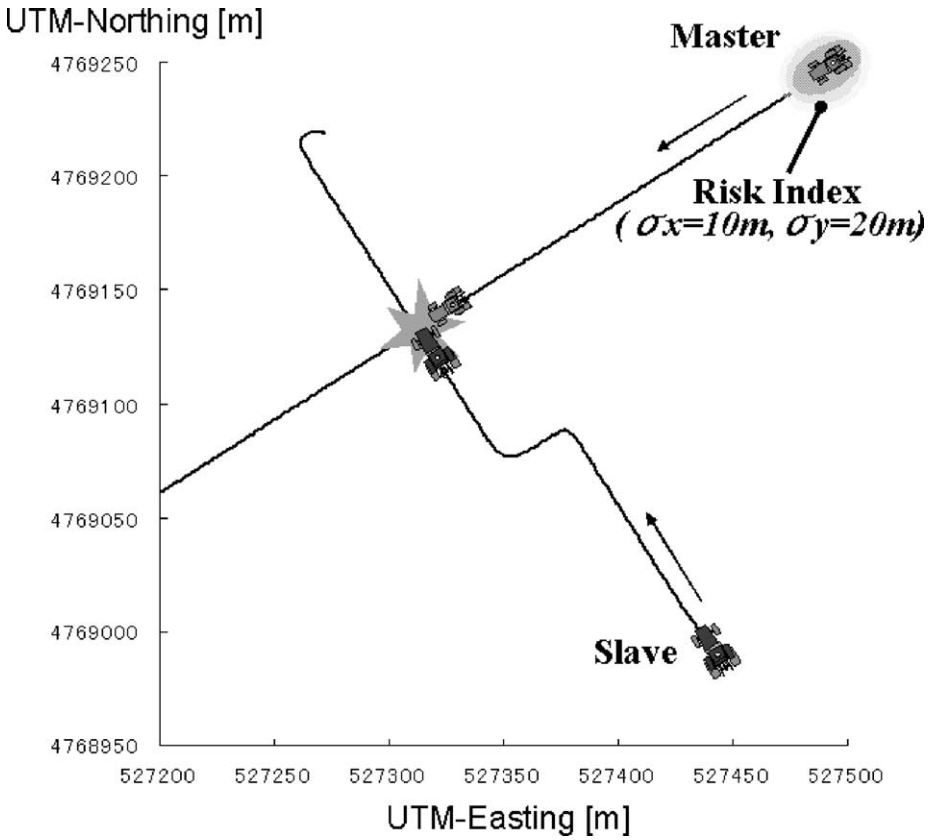

Fig. 9. Setup for a computer simulation of the slave avoiding the master. The master's path and speed were set so that it would crash with the slave at a certain location in a field.

decrease speed, expressed in Eq. (26), ensured the avoidance of a crash in every instance of the simulation.

Figs. 9 and 10 show simulation results for the case where both master and slave are moving on trajectories. The parameters σ_x and σ_y were 10 and 20 m, respectively. In addition, the pathway for the slave was the same as in the last simulation. The pathways and the speed of the master and the slave were set up so that they would crash at a certain position in a field if the slave did not take an appropriate measure. Fig. 10 indicates that, because the slave slowed down, the impending crash was averted. As seen in the results, the slave reduced its speed to approximately 0.05 m/s, in response to the risk index. The closest distance between the master and the slave was 12.5 m, but the master passed safely in front of the slave. At the master's and the slave's closest position to each other, the risk index was only 0.46.

Figs. 11 and 12 illustrate the alternative measure for avoiding a crash. This measure involved the slave changing pathways in addition to slowing down. As mentioned above, the slave calculates its own pathway to reach a destination. However, to avoid crashing with

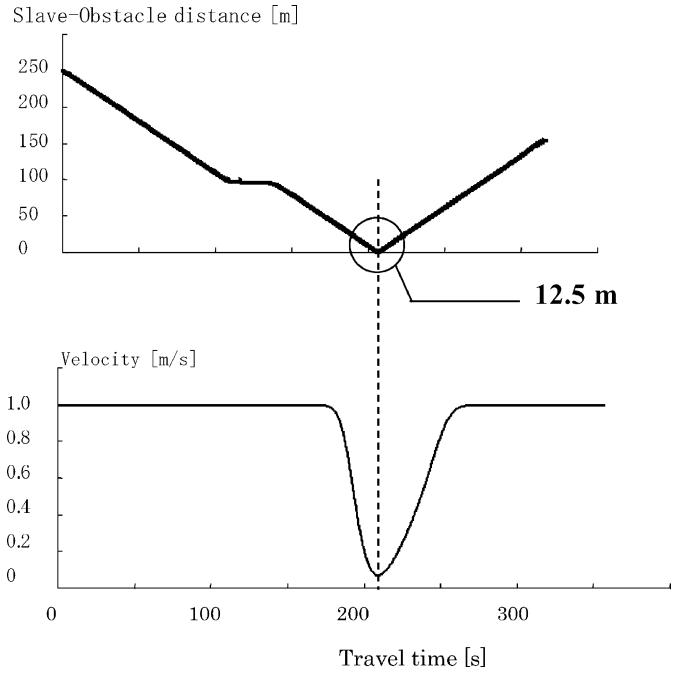

Fig. 10. The slave avoiding the master, averting a crash, by reducing its own speed.

the master, the slave can change pathways using Eqs. (27) and (28):

$$\delta^* = (1 - e)\delta + 40e \tag{27}$$

$$e = 1 - \exp\left(-\frac{\text{Risk index}}{0.04}\right) \tag{28}$$

As a point of fact, the algorithm does not make the slave generate an alternative pathway to the original one. Instead, the steering angle δ^* is modified based on the risk index. As seen in Eq. (27), the slave applied a combination of feedforward and feedback controllers. Basically, the feedforward controller prevented the crash, while the feedback controller continued to try to follow the original pathway. Steering output, which is variable δ in Eq. (27), was calculated using Eq. (25), which constituted the feedback controller. The results are shown in Fig. 12. As manifested in the trajectory, the slave changed its pathway to avoid a crash. The minimum distance between the master and the slave was 12.6 m, and the risk index at that point was 0.49. Interestingly, the slave returned to its original pathway after the risk index decreased. As seen in Eqs. (27) and (28), the output from the feedback controller and the output from the feedforward controller were always in competition. After it avoided the dangerous situation, the steering output from the feedback controller became predominant against the steering output determined by the risk index, resulting in a return to the original pathway.

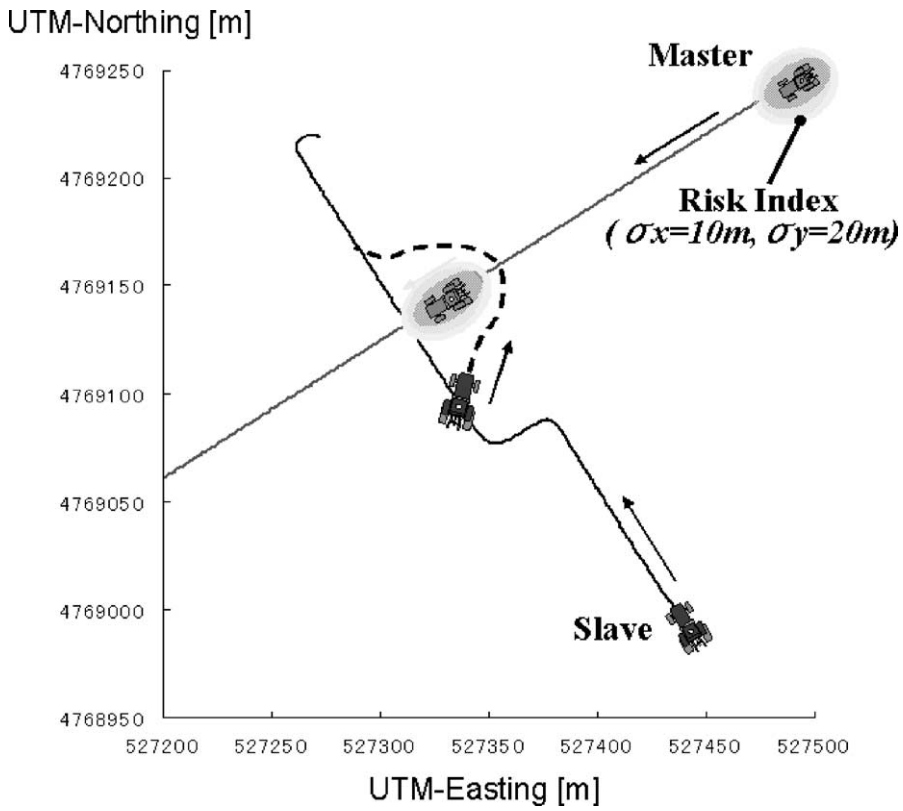

Fig. 11. Setup for a computer simulation of avoiding the master by changing pathways.

3.2.2. FOLLOW algorithm

In this section, we describe the results of the simulations using the FOLLOW algorithm that were performed with simulated robot tractors having two degrees-of-freedom of motion. The master traveled on a predetermined straight path, while changing speed according to a predetermined schedule. The speed of the master changed following a sinusoidal pattern ranging from 0.5 to 1.5 m/s over 200 s. The master and the slave communicated their status via a wireless LAN system at an interval of once every 0.5 s. The command from the master to the slave indicated that the spacing, d, should be 2 m and the relative angle, γ, should be 45 degrees. Initial errors of the slave were set at 2 m for its lateral offset direction and at 5° for its angle of direction. The initial speed of the slave was 0.5 m/s. Furthermore, because we had to include the latency of communication that would occur in an actual situation, 0.1 s of time delay was added during the simulation. Fig. 13 shows the results when the slave was steered by the sliding mode control algorithm developed for the simulations. The response of the robot when controlled by the sliding mode controller versus when controlled by a conventional PD controller are presented for comparison. The PD controller can be simply expressed in the following equations, based on Figs. 4 and 5:

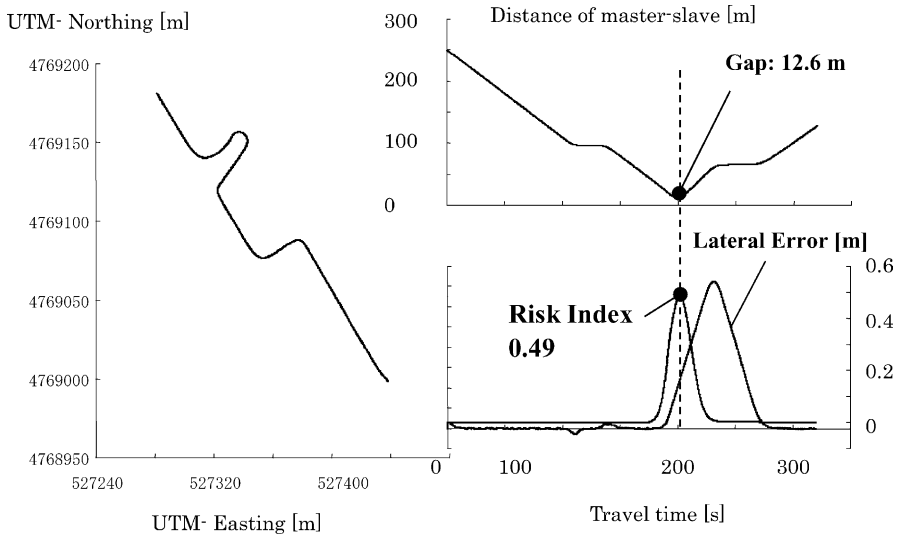

Fig. 12. Trajectory for avoiding the master by changing pathways. The slave changed pathways to avoid the crash. The minimum distance between the master and the slave was 12.6 m, and the risk index at that point was 0.49.

Lateral controller:

$$\delta = -\{\alpha_\delta(\phi_\mathrm{s} - \phi_\mathrm{m}) + \beta_\delta \varepsilon_\mathrm{c}\} \tag{29}$$

Spacing controller:

$$V_\mathrm{s} = V_\mathrm{m} + \alpha_v \dot{\eta} + \beta_v \eta \tag{30}$$

where, α_δ, β_δ, α_v, and β_v are control gains.

The slave, with an initial tracking error of 2 m, followed a straight path from $t = 0$ to 200 s. It can be seen that the performance of the sliding mode controller was better than that of the PD controller. For the lateral control in Fig. 13a, the sliding mode controller gave a relatively large overshoot compared to the PD controller. The RMS error of the lateral control over the 200 s implementation using the sliding mode controller was 0.134 m. As comparison, the PD controller gave an RMS error of 0.184 m in the same simulation conditions. Consequently, the lateral error of the sliding mode controller indicated that it performed better than the PD controller due to its rapid response during the initial time period. For the spacing control, a large difference in the performance between the sliding mode controller and the PD controller was seen in Fig. 13b. The sliding mode controller achieved highly accurate spacing control compared with the PD controller. The RMS error of the sliding mode controller and the PD controller were 0.106 and 0.131 m, respectively. In addition, since the speed control change was discrete, the PD controller oscillated more and was more unstable. In an actual situation, the robots can change the engine speed at increments of 100 rpm, using an eight-step transmission. Therefore, a look-up-table of the slave speed, composed of an engine speed set and corresponding shift positions, was prepared for this simulation. The nonlinearity of this motion model made the PD controller's

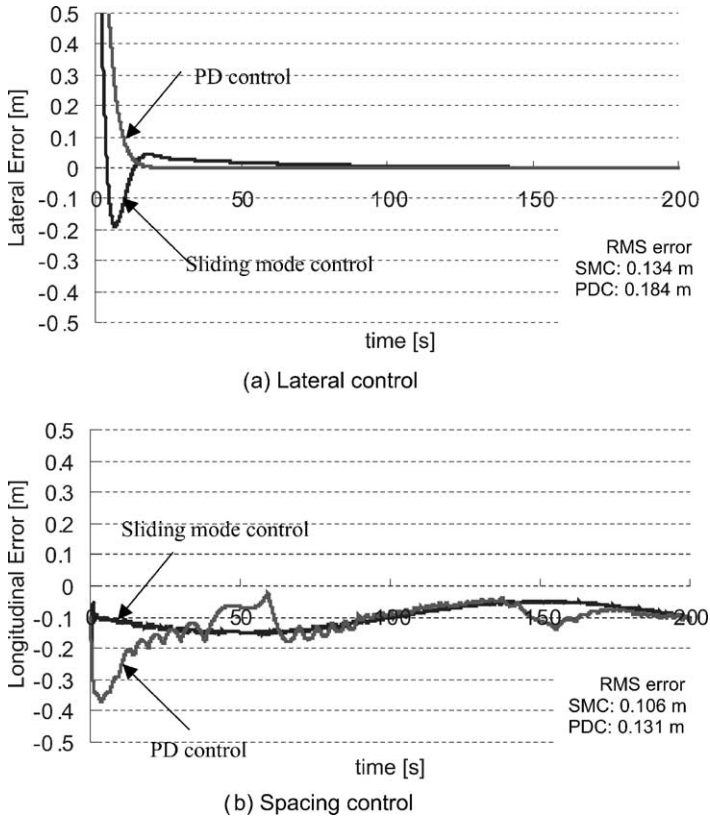

Fig. 13. Performance of the FOLLOW algorithm on lateral offset and spacing controls. The sliding mode controller was adopted for both spacing and lateral motions.

performance worse. Consequently, it was concluded that the newly developed sliding mode controller had better performance for both lateral offset and spacing controls.

4. Conclusions

This paper has dealt with cooperative work in a multi-robot structure. Two robot tractors were employed in this multi-robot structure, referred to as a master–slave system. In addition, two types of basic operations, a GOTO algorithm and a FOLLOW algorithm, were developed in this research. The GOTO algorithm can be applied when the master wants the slave to go to a specific place, a certain distance from the current operational position. This type of cooperative navigation method can be adopted for harvesting hay. The FOLLOW algorithm results in a common, cooperative style of work. The slave follows the master from a given relative distance and angle. This type of cooperative work provides a large benefit of increased productivity, even when using two identical machines. In this

paper, the master–slave system was first defined, then local communication formats were constructed.

Only the GOTO algorithm requires the detection of the master and the avoidance of a collision by the slave. Two measures for avoiding the master were developed by defining a risk index. The risk index, which was expressed using a two-dimensional normal distribution, described the level of danger of crashing. The measures to be taken by the slave were to 'slow down' and to 'change pathways'. Even though the pathway and speed of the master were set so that it would crash with the slave at a certain location in a field, the master was able to safely pass in front of the slave, due to the slave autonomously slowing down to approximately 0.05 m/s. The slave's speed was determined by the value of the risk index. Also, the avoidance measure of 'changing pathway' was developed by combining feedforward and feedback controllers. The slave could change pathways to avoid the crash in response to the risk index. Then, after avoiding the crash, the slave could return back to the original pathway by using the feedback controller.

For the FOLLOW algorithm, sliding mode controllers were adopted for both the spacing control and the lateral offset control. During the simulation, the master traveled on a predetermined path and changed speed according to a schedule. The slave controlled its own speed and angle of direction to follow the path of the master. When the slave was controlled by the sliding mode controllers, the following performance was better compared to the when it was controlled by a conventional PD controller because the sliding mode control was suitable for control on a highly nonlinear system. Also, the RMS error of the sliding mode controller was smaller for both lateral deviation and spacing error. In addition, since the speed control changed speed in discrete steps, the PD controller oscillated more and was more unstable. It was concluded that the sliding mode controller that was developed for this study had better performance for both lateral and spacing controls.

References

Bell, T. 1999. Automatic guidance using carrier-phase differential G.P.S. Computers and Electronics in Agriculture: Special Issues Navigating Agricultural Field Machinery.

Benson, E., Stombaugh, T., Noguchi, N., Will, J., Reid, J.F., 1998. An evaluation of a geomagnetic direction sensor for vehicle guidance in precision agriculture applications. ASAE Paper 983203.

Choi, C.H., Erbach, D.C., Smith, R.J., 1990. Navigational tractor guidance system. Transactions ASAE 33 (3), 699–706.

Erbach, D.C., Choi, C.H., Noh, K., 1991. Automated guidance for agricultural tractors. In: Proceedings of the ASAE Symposium on Automated Agriculture for the 2lst Century, 1991, pp. 182–191.

Hendrick, J.H., Tomizuka, M., Veraiya, P., 1994. Control issues automated highway systems. IEEE Control Systems, 21–27 December, 1994.

Noguchi, N., Reid, J.F., 2000a. Engineering challenges in agricultural mobile robot towards information agriculture. In: Proceedings of the XIV Memorial CIGR World Congress, pp. 147–154.

Noguchi, N., Kise, M., Ishii, K., Terao, H., 2000b. Dynamic path planning based on spline function for agricultural mobile robot. In: Proceedings of the XIV CIGR World Congress, pp. 943–946.

Noguchi, N., Reid, J.F., Zhang, Q., Will, J.D., Ishii, K., 2001. Development of robot tractor based on RTK-GPS and gyroscope. ASAE Paper 01–1195.

Noguchi, N., Kise, M., Ishii, K., Terao, H., 2002. Field automation using robot tractor. In: Proceedings of Automation Technology for Off-Road Equipment, pp. 239–245.

O'Connor, M., Elkaim, G., Parkinson, B., 1995. Kinematic GPS for closed-loop control of farm and construction vehicles, ION GPS-95. Palm Springs, CA, September, pp. 12–15.

O'Connor, M., Bell, T., Elkaim, G., Parkinson, B., 1996. Automatic steering of farm vehicles using GPS. In: Proceedings of the 3rd International Conference on Precision Agriculture, Minneapolis, MN, 23–26 June.

Peng, H., Tomizuka, M., 1993. Preview control for vehicle lateral guidance in highway automation. Journal of Dynamic Systems, Measurement, and Control 115, 679–686.

Reid, J.F., Zhang, Q., Noguchi, N., Dickson, M., 2000. Agricultural automatic guidance research in North America. Computers and Electronics in Agriculture 25, 155–167.

Smith, L.A., Schafer, R.L., Young, R.E., 1985. Control algorithms for tractor implement guidance. Transactions ASAE 28 (2), 415–419.

Smith, L.A., Schafer, R.L., Bailey, A.C., 1987. Verification of tractor guidance algorithms. Transactions ASAE 30 (2), 305–310.

Stombaugh, T., Benson, E., Hummel, J.W., 1998. Automatic guidance of agricultural vehicles at high field speeds. ASAE Paper 983110.

Tillett, N.D., 1991. Automation guidance sensors for agricultural field machines: a review. Journal of Agricultural Engineering Research 50, 167–187.