1988

Automatic guidance system for farm tractor

Chang Hyun Choi

Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Agriculture Commons, and the Bioresource and Agricultural Engineering Commons

Recommended Citation

https://lib.dr.iastate.edu/rtd/8836

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Automatic guidance system for farm tractor

Choi, Chang Hyun, Ph.D.
Iowa State University, 1988
Automatic guidance system for farm tractor

by

Chang Hyun Choi

A Dissertation Submitted to the Graduate Faculty in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major: Agricultural Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1988
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>OBJECTIVES</td>
<td>4</td>
</tr>
<tr>
<td>LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>Automatic Guidance Systems</td>
<td>5</td>
</tr>
<tr>
<td>Marker Following System</td>
<td>7</td>
</tr>
<tr>
<td>Machine Vision System</td>
<td>12</td>
</tr>
<tr>
<td>Navigational System</td>
<td>14</td>
</tr>
<tr>
<td>Steering Control</td>
<td>18</td>
</tr>
<tr>
<td>MATERIALS AND METHODS</td>
<td>27</td>
</tr>
<tr>
<td>Description of Automatic Guidance System</td>
<td>27</td>
</tr>
<tr>
<td>Spatial Position Sensing System</td>
<td>27</td>
</tr>
<tr>
<td>Front Wheel Angle</td>
<td>29</td>
</tr>
<tr>
<td>Tractor Travel Speed</td>
<td>31</td>
</tr>
<tr>
<td>Interface System</td>
<td>32</td>
</tr>
<tr>
<td>Development of Steering Control Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>Kinematic Behavior of Tractor Movement</td>
<td>36</td>
</tr>
<tr>
<td>Computation of Steering Angle</td>
<td>39</td>
</tr>
<tr>
<td>Computer Simulation</td>
<td>47</td>
</tr>
<tr>
<td>Field Experiments</td>
<td>51</td>
</tr>
<tr>
<td>RESULTS AND DISCUSSION</td>
<td>56</td>
</tr>
<tr>
<td>Computer Simulation</td>
<td>56</td>
</tr>
<tr>
<td>Field Experiments</td>
<td>66</td>
</tr>
<tr>
<td>Guidance System with two AGNAV units</td>
<td>66</td>
</tr>
<tr>
<td>Guidance System with one AGNAV unit</td>
<td>69</td>
</tr>
<tr>
<td>SUMMARY AND CONCLUSIONS</td>
<td>79</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>83</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>88</td>
</tr>
<tr>
<td>Appendix</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>A</td>
<td>Main Program for Simulation</td>
</tr>
<tr>
<td>B</td>
<td>Main Program for Field Test</td>
</tr>
<tr>
<td>C</td>
<td>Included Files, Subroutines, and Functions</td>
</tr>
</tbody>
</table>
INTRODUCTION

Conservation tillage can be defined as any planting and tillage system that retains at least 30 percent crop residue cover on the soil surface after planting (Little, 1987). These systems include no-till or slot planting, ridge-till, strip-till, mulch-till, and reduced-till. In 1986, the total acres of cropland under conservation tillage was 97.6 million - that was 32.9% of the total planted cropland acres in the United States (CTIC, 1987). The reasons for using conservation tillage systems are cost and time saving as well as soil and water conservation (Magleby et al., 1985).

Use of the conservation tillage system known as ridge-till has steadily increased for corn (Zea mays L.) and soybean [Glycine max (L.) Merr.] production. The ridge-till system realized 45% growth rate in 1985 and more than 75% of that increase occurred within the Cornbelt and Northern Plains Regions (CTIC, 1986). The main reasons for adopting ridge-till are to reduce production costs and to increase profitability (Jolly et al., 1983). As farmers continue to look for ways to save more money on crop growing costs, use of the ridge-till system will increase (Lessiter, 1987). This tillage system also creates favorable soil conditions for root growth, gives good water management, and effectively controls soil erosion without sacrificing crop yield (Erbach, 1982; Bauder et al., 1985). It is important to plant rows at the same location and to rebuild the ridges in the same location every year. The efficiency of a ridge-till system depends upon the ability of a system to maintain the ridges in the same location. Without a system to guide
equipment along the previous ridge, "guess" rows become wide in places and narrow in others. This causes plants to be damaged during cultivation and yield to be lost during harvest.

Compaction of agricultural soils is due to several factors, but mainly due to wheel traffic by agricultural vehicles and equipment. The reduction of vehicle travel in the field could reduce soil compaction. Controlled traffic has been used to manage soil compaction and to increase tractive efficiency of machines (Dumas et al., 1973). The idea of controlled traffic is to establish permanent traffic lanes that are not tilled and are used for wheel traffic paths year after year. Wheel traffic zones and crop production zones are created and permanently separated. This is done because optimum conditions for vehicle wheel traction are different from those for crop production. The permanent traffic lanes become compacted and improve tractive efficiency, flotation, and timeliness of operation (Taylor, 1982). The untrafficked crop zones maintain better physical soil conditions which allow deeper root development and better water movement. Controlled tractor and machinery traffic can have a significant effect on soil compaction, plant growth, moisture availability and crop yield (Dumas et al., 1973).

By accurately controlling the traffic paths of agricultural vehicles, the width of the traffic zone and soil compaction in the plant root zone can be minimized, crops can be planted in the same rows, and ridges can be maintained at the same location every year. A practical field guidance system is required to keep field equipment operating only on the predetermined paths. Ridge-till and controlled traffic can be
fully implemented with an automatic guidance system for agricultural vehicles.

A guidance system that is able to generate a field map may be a useful tool for automatic control. This would require a general spatial position sensing system that can pinpoint the position of the machine at any time. With adequate sensor technology and a position sensing system, a computer can generate digitized maps of soil and crop variation. The system is essential for controlling the precise application to achieve the desired tillage results (Schafer et al., 1981). Automatic controls would use such information to steer the tractor to follow the same tracks repeatedly. This system could also make it possible to optimize the treatment applied to each field area rather than treating the entire field the same. This mapping system could be used to fertilize and plant a crop in each area of the field according to its yield potential, and to spray each area with a prescribed mixture and rate of herbicide or insecticide. The accuracy required from a position sensing system to develop the field maps for these operations may not be as critical as that needed for automatic guidance.
OBJECTIVES

The purpose of this study was to develop a guidance system that can be used for crop production practices such as ridge tillage or controlled traffic.

Specific objectives of this research were:

1. To develop a feasible automatic guidance system based upon a spatial positioning system.
2. To develop a tractor steering control algorithm that compares current tractor position with a predetermined path to determine proper front wheel angle.
3. To evaluate the steering control algorithm through use of computer simulation.
4. To test the automatic tractor guidance system in field conditions.
LITERATURE REVIEW

The need for automatic control of agricultural machinery systems to increase efficiency of field operation has been recognized for a long time (Moncaster and Harries, 1984). An increasing research effort in recent years has been devoted to automatic guidance for agricultural vehicles. An automatic guidance system for agricultural vehicles should be reliable, flexible, versatile, easy to maintain, simple to operate and inexpensive (Grovum and Zoerb, 1970). Although several types of guidance systems have been designed for mowers, tractors, combines, and apple harvesters, only some of them have fulfilled those requirements.

Automatic Guidance Systems

The main elements in automatic guidance systems are sensors for collecting data from reference positions, a signal conditioner for converting the signals to proper form, a controller for processing data to guide the vehicle, and actuators for steering. With explosive technological developments in electronics, microcomputers have been widely used as the controllers of automatic guidance systems. Tennes and Murphy (1984) used a microcomputer to detect lateral position errors from sensors mounted on the tractor, to determine the desired steering angle, and to steer the tractor using the hydraulic steering system.

The major part of an automatic guidance system is the sensor to obtain positioning information. Sensors with a controller can successfully guide agricultural vehicles along predetermined paths. Position sensors for automatic guidance can be categorized as contact
type with mechanical arms or linkages and microswitches and non-contact type that use either ultrasonic, photoelectronic, radio, microwave, light, or visual signals. Automatic guidance systems can be divided into marker following systems, machine vision systems, and navigational systems. Table 1 shows the characteristics of automatic guidance systems.

**TABLE 1. Characteristics of automatic guidance systems**

<table>
<thead>
<tr>
<th>System</th>
<th>Cost</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marker Following System</td>
<td>Cheap</td>
<td>Accurate</td>
<td>Loss of directrix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reliable</td>
<td>Not flexible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Easy to use</td>
<td>Not versatile</td>
</tr>
<tr>
<td>Machine Vision System</td>
<td>Very expensive</td>
<td>Flexible</td>
<td>Poor resolution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Versatile</td>
<td>Slow process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Complex units</td>
</tr>
<tr>
<td>Navigational System</td>
<td>Expensive</td>
<td>Flexible</td>
<td>Less accuracy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Versatile</td>
<td>Need algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reliable</td>
<td></td>
</tr>
</tbody>
</table>
**Marker Following System**

The most popular and simplest guidance systems for tractors employ marker followers that use some form of ground installed or presented devices, such as buried cables, furrows, or standing crops. Brooke (1968) and Rushing (1971) used leader cables buried a few inches below the surface of the ground as the guidance marker. The system followed a series of buried wires excited by a low current and low frequency signal. The electromagnetic field around the wire was used to steer a tractor. The steering sensor consisted of two cored coils connected in series so that phase differences produced voltages that were used to sense tractor position. If the sensor is placed directly over the buried wire, equal voltage was induced in each of the coils and output remained zero. If the sensor is to the left of the buried wire, the right coil had a higher voltage. The steering servo was actuated until the output returned to zero. Steering accuracy and repeatability of ± 2.5 cm were observed at speeds up to 9.5 km/h. The cost of preparing the field was $15 per acre and the cost of the equipment installed on the tractor was less than $3,000 (Rushing, 1971).

Schafer and Young (1979) used three pairs of ferrite core, resonant-circuit antennas to sense the location of an energized buried wire. One of these pairs provided a reference signal to the controller. The horizontal pair and the vertical pair were sensitive to angular orientation and lateral position over the energized wire, respectively. The antenna system was mechanically coupled to tractor's steering system such that the angular rotation of the antenna system was equal to and in
the same direction as the steering angle of the front wheels. With no implement load on the tractor and for straight-line operation at speeds to 10 km/h, the deviation from the buried wire was less than 5 cm.

Goering et al. (1972) studied the economic feasibility of the automatic guidance system with buried wires. Four farms from 250 to 633 acres in size were tested to calculate cost savings resulting from automatic control as compared to manual control. The cost of installed wire was about $20 per acre and the estimated price of control equipment was $4,000. They found that automatic control was profitable for multicrop farms greater than 770 acres. The system was most profitable on corn, soybeans and sorghum and unprofitable on small grain, silage and hay crops. For economic reasons, the life of the leader cable should be at least 10 years so that might be the minimum length of time the farmer need be restricted to a fixed field layout.

The cable following system has several advantages. The equipment is inexpensive, simple, and easy to maintain. This system has superior accuracy and stability over the other marker following systems. However, the main disadvantage of the cable following system is the lack of versatility. The system can be used only in the field where the cables are installed and the method can not be easily adapted to a variety of tool widths.

With the furrow following system, a furrow is left with each pass of the tractor and implement. The furrow is used to guide the tractor for the next run. Various kinds of mechanical followers, such as a skid (Hilton and Chestney, 1973), a 2.1-m supporting arm (Kirk et al., 1976),
or a set of disks (Pool et al., 1984), were moved in contact with the
previous furrows or ridges and sensed displacement errors. The
controller activated the steering system of the tractor in proportion to
the horizontal positioning error measured by the furrow follower.

An ultrasonic pulse-echo-ranging method was used to develop furrow-
following guidance systems for agricultural tractors (Warner and Harries,
1972). The systems used transversely scanned, downward looking
ultrasonic sensors to detect the furrow. The range discriminator
determined whether a received pulse had been reflected from the unplowed
land or the furrow bottom and passed only the latter to the control
tracking logic. They reported that this system had good stability at
speeds up to 11.2 km/h on test tracks consisting of a row of concrete
blocks laid on tarmac but could not be evaluated in field conditions
because of inadequately reflected ultrasonic signal from the soil.

Harries and Ambler (1981) developed a furrow following sensor and a
range meter for an automatic guidance system. The sensor included a
projector that formed a band of light on the ground about 0.5 m ahead of
the front axle of the tractor and a receiver with two arrays of
photodetectors. Deviation from the furrow resulted in an increase in the
number of illuminated elements in one array of photodetectors. They
found some limitations of this sensor. The receiver must be set to the
appropriate angle to view the furrow correctly. The vertical distance
measurement from the tractor depends upon the systems ability to scan a
field of view on the ground or in the image plane. Scanning may be
affected mechanically by moving the sensor, or electronically in a multi-

element array.

The range meter was used to control turning at the headland. The range

meter measures the time delay in terms of phase difference between

a projected continuously modulated light signal and the reflection

received from a reflecting post at field boundary. The projected light

was produced by a solid state infrared emitting diode (IRED) square-wave

energized at 5 MHz. Results of calibration showed that the error would

be less than 0.1 m over the working range (4-22 m). The bearing of

boundary post relative to the tractor was measured simultaneously with

the range measurement by a precision potentiometer linked to the rotating

mirror of the range meter. A rate gyroscope was used to get a continuous

record of the heading of the tractor relative to that at the start of the

turning sequence. The position and heading data were input to a

microprocessor which initiated and controlled a modified U-turn at the

headland.

Systems to guide from standing crops were used to steer tractors,
mowers, windrows, forage harvesters, apple harvesters, and combines.
Contact type sensors, such as feelers (Richey, 1959; Liljedahl and
Strait, 1962) and sensing arms (Suggs et al., 1972; Busse et al., 1977),
were used to sense the position of a crop row and to control actuators in
proportion to the amount of machine offset from the row.

The automatic mower used two grass edge sensors - one detected cut

grass and the other detected uncut grass. Each sensor was positioned

perpendicular to the travel direction. Microswitches (Surbrook et al.,
1982) and photoconductors (Nine and Griffin, 1982) were used as grass edge sensors. Kirk and Krause (1976) designed a swather edge guide steering control system using an infrared photoelectric sensor to detect the edge of a standing crop. The sensor was capable of detecting the crop edge up to 35 cm away and operated by reflecting pulsed infrared light of the crop edge, with the intensity of the reflected light giving an indication of the distance of the crop edge from the sensing head. The sensor was mounted on an arm extending in front of the swather.

Iseki Agricultural Machinery Mfg. Co., Ltd. introduced a completely automatic combine harvester with an automatic directional mechanism (Kanetoh, 1977). This combine harvester had automatic control systems for direction, reaping height, stalk feed, and bag packing. The direction sensor passed along the edge of the uncut plants to detect the machine position. If no plant exists during straight motion, the machine keeps on moving for some distance to determine whether there is a missing plant or row has ended.

A small, relatively inexpensive ultrasonic system has been developed and used for robotic applications. One ultrasonic ranging system was originally intended for use as a camera rangefinder for automatic focusing. The ultrasonic sensor is becoming more popular because it generates a beam that has sharper boundaries than other sonar sensors and has a small angle (5.5 degrees) of divergence (Gross, 1978). Ciarcia (1980) evaluated the Polaroid ultrasonic ranging system designer's kit which consisted of a circuit board and ultrasonic sensor. The cost of this kit was $125. This system was capable of detecting the presence and
distance of objects within range of 0.3 m to 10.6 m with resolution of ± 3 cm.

A steering control system using ultrasonic sensors was designed for an apple harvester so that the harvester would automatically drive over a tree row (McMahon et al., 1983). Five ultrasonic sensors and a timing circuit were used to measure the position of apple tree trunk. The controller based on a microprocessor directed the harvester’s wheel to turn to an angle which was proportional to the harvester’s lateral position error from an apple tree trunk. Test of sonar tree sensing indicated that the sonar system was accurate within ± 4 cm for measurements in the range of 0.3 m to 15 m.

The furrow or standing crop following systems are more flexible than the cable following system, but have less accuracy and stability. The first furrow should be provided manually for the next run’s guidance. Because there are no permanently placed or fixed reference points, the first furrow may not be at the same location as last year’s and the error may be accumulative from each pass. The ultrasonic sensor is an inexpensive and reliable non-contact type sensor. Short range would be a limiting factor for use in other than marker following systems. A furrow or crop following system based upon a non-contact sensor with additional information from fixed reference points seems to be desirable.

**Machine Vision System**

A powerful new technology for control applications is the machine vision system that can deduce three dimensional visual information present in the environment and provide meaningful information to a
computer. The main applications of machine vision systems can be found in robot guidance and control, part sorting, inspection, gaging, and process control. Two main tasks of machine vision systems are electro-optical imaging, converting optical radiation to an appropriate electronic signal for input to the computer, and image processing, extracting useful information from an electronic image provided by the sensor (Pinson, 1986).

Gerrish and Surbrook (1984) studied computer vision as sensory input for a guidance system for farm tractors and the steering control requirements for vision based guidance system. They found that the camera, mounted 1.5 m above the ground with a 30 degree tilt, had a good looking-ahead capability that may enable efficient open-loop steering correction.

Image processing techniques were investigated to find a path-line in agricultural field operation. Reid et al. (1985) developed a binary thresholding strategy for determining the guidance directrix data in row crop images and analyzed the effectiveness of the technique on field images of cotton. Selection of a threshold value was based upon the light intensity distributions for the two basic classes of information; plant canopy and soil background. Gerrish et al. (1986) captured the image by use of a 256 by 256 resolution camera. To extract a line from this picture, several methods were tried. They mentioned that color filters, infrared filters, varying resolution and voting-type line finders would help to get a clear binary image from field crops.
Machine vision systems are still in their developmental stage but successfully tested systems have proven the feasibility of the machine vision technique. Greater picture or image resolution and faster processing speed are problems to be solved and there is a great need for an easily usable and inexpensive small-scale machine vision system.

Navigational System

Marker systems follow cables permanently placed in position or the marker left from the previous pass. These system do not provide the geometrical location or heading error of the tractor. An ideal automatic guidance system should determine the tractor position and automatically correct the steering to the travel direction desired. For guiding an agricultural vehicle along a row or work-edge, position accuracy is more important than heading accuracy.

A dead-reckoning system was designed to guide a tractor (Gilmour, 1960). A tractor was started from a known point and operated by measuring heading and distance traveled, independently of any external reference. The inputs required to operate this system were heading angle, length of a field, implement width, and desired direction. Bearing information was obtained from a magnetic compass, later changed to a magnetically stabilized gyrocompass. Distance information was obtained from a towed track behind the tractor. Because there were no external references, no provision could be made to correct direction due to side slip and turning at the headland.

A planter guidance system was developed that was able to use more than one position sensing system, to evaluate the data and to provide
steering information to the planter operator (Patterson et al., 1985). The electronic guidance system involved the interaction of four component systems: an ultrasonic main guidance system sensing a physical directrix, a backup dead reckoning system referencing a perceived directrix, a microcomputer control system managing all functions of the electronic guidance system, and a steering directional display to indicate guidance information to the planter operator.

Tagami et al. (1983) developed an inertial navigation system for automobiles that could monitor the driving progress and present location of the car on a CRT (Cathode Ray Tube) where a transparent map sheet was set. The system, requiring no external information, was able to detect the direction and distance traveled by use of a precision gas rate gyro sensor and a distance sensor and to instantaneously integrate changes by use of a microcomputer. The distance traveled was measured by a pulse sensing system that recorded tire revolutions. The directional information was obtained from a gas gyro that detected the temperature difference between two hot wires due to an injected helium gas flow during turning of the car. The system determined the present location of the car based upon direction and travel distance from an initial starting point, and produced a picture for display of the traveled course by processing the stored two dimensional coordinate data.

Many researchers have proposed an alternate automatic guidance system by field mapping. A radar or laser unit used as a position sensor could provide angular and linear displacement signals to a microcomputer that would calculate tractor location. The microcomputer would compare
this location to a predetermined course (programmed into the microcomputer at the beginning of the field operation) and cause the transmitter to transmit the required commands for correction of course or actuation of steering devices on the tractor. MacHardy (1967) tried to determine the position of a tractor by triangulation. He planned to measure the angle from the two infrared detectors, to calculate the tractor position, and to compare the difference between the actual position and programmed position. He found that the tractor could not be satisfactorily controlled without information on the distance off-course and the attitude of the tractor.

LORAN (LOng Range Aid to Navigation) can find the absolute position of the target and previous errors do not greatly affect the current reading. LORAN-C has a series of fixed transmitters which emit pulses in a precisely timed sequence and find the current position of receiver based on the elapsed time between the receipt of signals. Wagner and Schrock (1986) evaluated LORAN-C in highway and field vehicles and found that the resolution was one-hundredth of a minute of longitude and latitude, approximately 20 m. A self-propelled plowing robot, designed by Bonicelli and Monod (1987), equipped with an absolute positioning system that used triangulation between tetrahedric passive radar beacons on the field border and a rotating radar on the moving vehicle. They reported results of 1 m accuracy for minimum distance of 15 m and maximum distance of 300 m. This machine was not sufficiently reliable to work continuously.
The availability of solid state continuous wave laser diodes, position sensitive photodetectors, and easily interfaceable microcomputers would make it possible to design an accurate position sensing system for tractor guidance and field mapping. Sakai (1978) developed equipment automatically to measure the position of an automobile by triangulation using a laser beam. The system included two laser beam units, each equipped with a projector, a receiver, and a rotary mirror. There were also a cardinal point mirror on reference line, a reflecting pole on test vehicle roof, a reference frequency oscillator for driving rotary mirrors, a pulse reshaper and digital counter for measuring pulse intervals, and a data recorder.

A laser guidance system was designed to measure the location and orientation of the Multi-Jointed Mobile Truss (Mizrach et al., 1987). The system included a laser rotating transmitter located in a fixed position, four receivers mounted on a rigid rectangular frame of a moving vehicle, and a IBM PC with an analog and digital I/O board. Three angles were calculated from elapsed time and a known constant angular velocity of the laser transmitter. The positioning was achieved from three calculated angles and dimensions of four rectangular frame corners. Data for distance and angle measurements were not accurate enough. A higher rate pulse laser and a precise quartz lock control motor for a rotating laser beam are required to improve resolution.

A similar concept was used to design a field machinery guidance system consisting of a single laser source, single detection unit and two scanning mirrors (Shmulevich et al., 1987). The system can find the
relative x-y coordinates of field machinery by geometric triangulation. Test results showed that use of an angle encoder with a higher resolution would be capable of measurement with wanted accuracy. This technology could be applied with minor changes to measure with an error of less than 15 cm in a 400-m by 400-m field.

Steering Control

Steering control characteristics of a tractor are concerned with its response to steering commands and to environmental inputs affecting the direction of motion. Two basic problems are the control of the tractor to follow a desired path and the stabilization of the direction of the motion against external disturbances. For most vehicles, steering is normally done by changing the angle of the front wheels.

The prime consideration in the design of the steering geometry is minimum tire scrub during cornering. The Ackermann steering geometry shows the relationship between the steer angle of the inside front wheel \( \delta_{in} \) and that of the outside front wheel \( \delta_{out} \) shown in Equation 1.

\[
\cot \delta_{out} - \cot \delta_{in} = \frac{B}{L}
\]  

(1)

where \( B \) and \( L \) are tread width and wheel base length of the tractor, respectively. Agricultural tractors have tread width to wheel base length ratios \( (B/L) \) in the range of 0.55 to 0.68. When the steering angle of the outside front wheel \( \delta_{out} \) equals 90 degrees with most common ratio \( (B/L) \) being 0.60, the radius of turning is 1.17 times the wheel base length (Schilling, 1960).
The steady-state response of a vehicle is the final condition of motion that occurs at some finite time after the start of a maneuver. When a vehicle is negotiating a turn at moderate or higher speeds, the four tires will develop significant slip angles. Wong (1978) showed the fundamental equation governing the steady-state handling behavior of a road vehicle from the geometric relationship shown in Figure 1.

\[
\delta_f = - \frac{L}{R} + \tau_f - \tau_r
\]

\[
= - \frac{L}{R} + \left(\frac{W_f}{C_f} - \frac{W_r}{C_r}\right) \frac{v^2}{gR}
\]

\[
= - \frac{L}{R} + \frac{W_f}{C_f} \frac{v^2}{gR}
\]

where

- \( \delta_f \): steering angle required to turn (rad)
- \( L \): wheel base length (m)
- \( R \): radius of curvature (m)
- \( \tau_f \): slip angle of front tire (rad)
- \( \tau_r \): slip angle of rear tire (rad)
- \( W_f \): weight distribution of front wheel (kN)
- \( W_r \): weight distribution of rear wheel (kN)
- \( C_f \): cornering stiffness of front tire (kN/rad)
- \( C_r \): cornering stiffness of rear tire (kN/rad)
- \( V \): forward velocity (m/s)
- \( g \): acceleration due to gravity (9.806 m/s\(^2\))
- \( K_{us} \): understeer coefficient (rad)
FIGURE 1. Simplified steady-state turning response of vehicle. Steering angle ($\delta_f$) depends upon wheel base ($L$), radius ($R$), and slip angles of front tire ($\tau_f$) and rear tire ($\tau_r$).
This equation indicates that the steering angle required to negotiate a given curve depends upon the wheel base, weight distribution, forward speed, and tire cornering stiffness. Dependent upon the value of the understeer coefficient or the relationship between the slip angle of the front and rear tires, the steady-state handling characteristics can be classified into three categories: neutral steer \((K_{us}=0)\), understeer \((K_{us}>0)\), and oversteer \((K_{us}<0)\).

Yaw velocity gain \((G_{yaw})\) is an often used parameter for comparing the steering response of road vehicles. It is defined as the ratio of the steady-state yaw velocity to the steer angle. When a vehicle operates at low speeds, there is a simple relation between the direction of motion of the vehicle and the steering wheel angle (Ellis, 1969). In the steady-state turn, it is assumed that the yaw and side slip accelerations become zero. He showed the steady-state yaw response to steering input of an automobile, without suspension system, can be expressed as:

\[
G_{yaw} = \frac{\Omega}{\delta} = \frac{V}{L}
\]  

\( (3) \)

where

- \(G_{yaw}\) = yaw velocity gain (s\(^{-1}\))
- \(\Omega\) = yaw velocity (rad/s)
- \(\delta\) = mean steering angle (rad)
- \(V\) = forward speed (m/s)
- \(L\) = wheel base length (m)
The equation shows that the yaw velocity gain is dependent upon the forward speed and the geometry of the vehicle. The yaw velocity gain increases linearly with the increase of the forward speed.

A continuous measurement of the tractor displacement and transverse velocity with respect to the preset guide path contains enough information for automatic tractor guidance (Grovum and Zoerb, 1970). Field tests showed that the best system performance was found when forward tractor speed was 6.7 km/h with the displacement sensor mounted in line with the tractor front wheels. They concluded that the parameters affecting the stability of the marker-following guidance system were tractor forward speed, displacement sensor location and characteristic, servomechanism response characteristic, displacement gain, and velocity gain.

Mathematical descriptions of vehicle-guidance systems were primarily geometric and kinematic models restricted to plane motion. The mathematical models can be combined into a computer simulation program. The computer simulation can be particularly useful to predict the effect of varying certain vehicle parameters that could not be varied easily on the prototype. It is important in the system design process to be able to predict and understand the vehicle handling behavior under certain conditions. Kelly and Rehkugler (1980) identified 21 variables that affected the stability of tractors. These variables were classified as static properties, dynamic properties, initial conditions, driver-controlled parameters, and surface conditions. All models include several assumptions based upon their particular interests because some
factors may not be consistent or fully defined. It is important to determine the most dominant factors and use them to develop the mathematical models. When a model with appropriate assumptions was developed, the accuracy of prediction should be measured to establish a level of confidence in the model. Verification is made by comparing model prediction values with prototype experimental results under same conditions.

A mathematical model was developed to investigate the relationship between tractor parameters and automatic steering accuracy for a marker following guidance system (Shukla et al., 1970). The model was simplified by assuming operation on a plane surface at low speed so that inertia and soil reaction forces could be neglected. The steering response of the vehicle was considered to be proportional to the front tracking error. The factors used for computer simulation were wheel base length, wheel tread width, steering type, type of drive, sensor length and location, travel speed, and type of path. The following conclusions were found concerning vehicle design for minimum tracking error: (1) Minimum wheel base length produced the best tracking accuracy; (2) Least overall tracking error was achieved when the front and rear tread widths were approximately equal; (3) Four wheel steering gave the best accuracy; (4) Rear wheel steering from a front sensor was less accurate than front wheel steering; (5) At least one sensor had to be near or ahead of the front axle to achieve stable operation; (6) The optimum sensor length was relatively independent of the type of path.
Parish and Goering (1971) developed a kinematic mathematical model for an automatic guidance system based on contact type crop sensor. The basic crop sensor consisted of two microswitches operated by the upward pressure of the hay. There was an allowable dead band between the switches. At low speeds there was relatively good agreement between computer simulation and prototype field test. As vehicle speed was increased, inertia effects and wheel slippage become more pronounced. They found that tread width, header width, and distance from drive axles to sickle had very little effect on RMS (Root Mean Square) error. RMS error was increased by making the steering corrections more abruptly and by increasing the vehicle speed. Theoretically, best accuracy was attained by keeping the two sensor switches as close together as possible.

Upchurch et al. (1983) developed a microprocessor based steering controller for an over-the-row apple harvester that would allow the drive to select any one of five steering modes: automatic, front only, four-way, crab, and rear only. Rear only was not tested because the rear only and crab steering modes used the same steering algorithm. The controller successfully maneuvered the rear wheels during the three manual steering modes (front only, four-way, and crab). The rear wheels were positioned very close to the desired position, which was dependent upon the selected steering mode and the front wheel position. During automatic steering, the controller kept the machine’s center within 7 cm of the tree stands with minimal oscillations in steering corrections. The appearance of
oscillations in the results were caused by the independent corrections by the front and rear wheels.

Young et al. (1983) evaluated a vehicle guidance controller by operating a tractor over three different paths (a straight line, a step function, and a sine wave) at various speeds and measuring the resulting guidance response and accuracy. The controller used an antenna mounted on the tractor to sense the location of an electrically excited wire buried beneath the ground. A linear relationship was assumed between tractor speed and the rate of increase in the steering rate required to correct steering errors. A NSD (Not Speed Dependent) algorithm and a SD (Speed Dependent) algorithm were evaluated. The major reason for investigating a SD algorithm was to minimize the effect of tractor speed on the steering response for all paths. However, they found that the tractor forward speed was the major factor affecting the steering accuracy for both algorithms. The frequency response of the hydraulic steering system may have limited the performance at the higher speeds when the SD algorithm was used. The steering control accuracy for those algorithms are very dependent upon error sampling rate and tractor forward speed. To control the steering wheel accurately, the error sampling rate should be increased as the forward speed increased.

Control algorithms for automatic guidance of three tractor-implement combinations were developed and evaluated by computer simulation techniques (Smith et al., 1985). Machine combinations considered were a front-steered tractor with a towed implement, a rear-steered vehicle with a front-mounted implement, and an articulated tractor with a towed
implement. The analyses were based upon the concept of computing the steering angle required to guide the vehicle such that the implement followed a predetermined path stored in the memory of a control computer. A spatial position sensing system was assumed to provide position coordinates of two points on the machine to initialize each iteration of the computational process. Simulation results indicated that guidance stability was highly dependent upon the magnitude of the steering gain factor. Maximum absolute implement errors increased as the steering gain factor was reduced and decreased as the distance traveled per update interval was reduced. They concluded that the constant-turn geometry techniques were usable for machine guidance.
MATERIALS AND METHODS

Description of Automatic Guidance System

A microcomputer based guidance system was designed which would be able to determine the actual location of the tractor in the field and to take corrective action based upon the difference between the actual tractor position and a predetermined path. Figure 2 shows the block diagram of the automatic guidance system. The automatic guidance system consists of two spatial position sensing units to locate positions of the tractor front wheel and the implement, a microcomputer to determine proper steering angle by analyzing the error, a stepping motor to steer the tractor front wheel, a potentiometer to measure front wheel angle, a wheel speed transducer to measure tractor forward speed, and an I/O (input/output) interface system. A 1.2-kw portable gasoline powered generator was used to supply electricity to the microcomputer, the stepping motor, and the I/O interface system.

Spatial Position Sensing System

AGNAV units, manufactured by D & N Micro Products Inc., were used to determine the locations of the center of the tractor front wheels and of the implement. The AGNAV unit is easy to operate, inexpensive ($6,500), and versatile for field operations. The AGNAV unit provides position data, in the form of x-y coordinates, of moving tractor and can be easily interfaced to a microcomputer. The AGNAV unit consists of a

1Trade and company names used in this dissertation are solely for providing specific information. Their mention does not imply recommendation or endorsement over others not mentioned.
FIGURE 2. Block diagram of automatic guidance system
computer-transceiver, a control panel, a display, a main antenna, a pair of repeaters (repeater-A and repeater-B), and three 12-volt batteries. The batteries power the computer-transceiver and the repeaters. The control panel provides switches to select, and displays to indicate, system modes, row number, swath width, battery status, and main antenna status. The display also indicates the error from the row and repeater signal status.

The computer-transceiver generates and transmits VHF radio signals (154.565 MHz - 154.605 MHz) to the repeaters where the signals are delayed and returned to the computer-transceiver. Returned signals from repeaters are used to determine location of a tractor. The x-y coordinates of the main antenna position are determined based upon repeater-A and repeater-B locations. The origin of the x-y coordinate system is the location of repeater-B. The AGNAV unit can locate the position of moving tractor up to 1.6 km from the repeaters. The error of tractor positioning depends upon the location of the main antenna relative to the repeaters. The accuracy of the x-y coordinates was about 25 cm with optimum location of the repeaters. Each AGNAV unit used a different frequency (154.57 MHz for the position of the center of the front wheels and 154.60 MHz for the position of the implement) to prevent interference between units.

Front Wheel Angle

A stepping motor (Bodine Electric Co., Model 2007) with a controller (THD-1810C 8-Amp Translator) was used with belt reduction to turn the steering wheel at the driver's seat (Figure 3). The typical torque
FIGURE 3. A stepping motor mounted on steering wheel post, with belt reduction, for steering tractor
required to turn the steering wheel is about 5 Nm. The motor has 5.3 Nm (750 oz-in) of dynamic torque at 100 steps per second and rotates 1.8° per step. The controller provides bi-directional stepping control and stepping rates up to 5,000 steps per second. The dynamic torque of the stepping motor decreased as the stepping rate increased. The controller was set to steer the front wheel at 4.5° per second. A safety switch was provided to rapidly disconnect the stepping motor from the power source so the steering wheel could be turned manually.

A potentiometer (5 k-ohms) was used to measure the angular position of the tractor front wheel. The potentiometer was attached to a vertical shaft which turned as the front wheels were steered. The output voltage of the potentiometer was proportional to the front wheel angle. The output of the potentiometer increased as the wheel steered left and decreased as the wheel steered right. The potentiometer was calibrated so the output of the potentiometer was zero degree when the tractor front wheels were pointing straight ahead.

**Tractor Travel Speed**

A magnetic pickup (American Philips Control Co., Airpax Model 087-304-0044) and 65-tooth gear were attached inside of the front wheel to measure ground speed. The magnetic pickup is compatible with the microcomputer and has a sensing frequency range from zero to 20,000 cycles per second.

The number of the teeth per gear was determined by Bedri (1982) so that every output pulse per second represents 0.16 km/h (0.1 mph). The conversion value was based upon the front tire circumferential distance
of 2.906 m. The circumferential distance of the front wheel can change with tire and soil surface conditions. The average circumferential distance of 2.91 m was calibrated and the number of pulses per second was counted by the microcomputer during field tests. The number of pulses per second and circumferential distance were variables to calculate the tractor travel speed and can be described as:

\[ V = \frac{D_f N_f}{65} \]  

where

- \( V \) = tractor travel speed (m/s)
- \( D_f \) = circumferential distance of tractor front wheel (m)
- \( N_f \) = number of pulses per second from transducer (s\(^{-1}\))

**Interface system**

A microcomputer (Zenith Model 159) was used to calculate the proper angle to which the front wheels should be turned for the tractor to follow the predetermined path. The microcomputer read the predetermined path data from floppy disk and wrote the actual tractor location onto a floppy disk. The microcomputer had two floppy disk drives, 640 kbytes of RAM (Random Access Memory), 8087-2 math co-processor, two RS-232 serial ports, and one printer port. Each AGNAV unit was connected to a RS-232 serial port of the microcomputer.

The microcomputer received measurement signals from the transducers and sent control signals to the stepping motor through a PC-Acquisitor (Dianchart Model PCA-48) that was connected to the printer port of the microcomputer. The interface system can handle up to 48 input channels
in the range 0.3 mV to 10 V with 16-bit resolution, 10 binary inputs, pulse counter, and 10 digital outputs for control. The system has a 16 bit A/D (analog to digital) converter, sampling rate up to 500 channels per second, and a signal conditioner to linearize most transducers and to compensate voltages for strain gages, RTDs (platinum resistance thermometers), and thermocouples.

Development of Steering Control Algorithm

Most guidance systems use an on-off steering control concept. Whenever the systems detect position errors, they simply steer the tractor in the direction to reduce the lateral position error. Some systems control the steering wheel angle in proportion to the lateral position error.

Use of field mapping data, in the form of x-y coordinates of the desired tractor path, would allow for predicting the curvature to follow. A position sensing system is needed to determine the location of the tractor in the field. A microcomputer can be used to calculate the tractor yaw angle, the lateral position errors by comparing the present location to desired position of the tractor, and the radius of the curvature from the field mapping data. Based upon the current tractor yaw, the current front wheel angle, and the predetermined path, the control algorithm can determine the steering angle that can reduce the lateral position errors at next measurement position. To develop the steering control algorithm, it is necessary to derive the relationships
to compute steering angles required to guide the tractor along the predetermined path.

The main requirement of an automatic guidance system is to control the tractor front wheels so that the implement follows a predetermined path. Unless the tractor is operating in sharp turns, the rear wheels will approximately follow the front wheel path. It has been shown that less than 5-cm fluctuations from a predetermined course, at the front wheel, produced negligible deviations at the rear, so that trailed implements follow a true course (Julian, 1971). On this basis a satisfactory control system may be designed around the requirement to maintain the front wheel within ± 5 cm of the desired course at a given tractor forward speed.

Farm tractors are operated on a variety of surface conditions. The frictional characteristics between soil and tire are not uniform or consistent (Ellis, 1969). It is impossible for a pneumatic tire to develop cornering forces without side slip even on a rigid road. There will be further slip in the field due to the deformation of the soil beneath the tire. The slip angles are dependent upon the side forces acting on the tires and their cornering stiffness. When the tractor operates at low speed, side forces are extremely small and slip angles will be negligible (Wong, 1978). Therefore, kinematic equations could accurately predict tractor turning behavior when the tractor is operated at low speeds. The computation to determine the proper steering angle was based upon the kinematic behavior of the tractor.
The kinematic equations for the tractor motion can be developed graphically or analytically. The bicycle model (Ge, 1987), in which wheels were considered lumped on the longitudinal center of the tractor, was used to develop the kinematic equations. The following assumptions were made in development of the steering control algorithm.

1. The tractor operated on the horizontal plane surface.
2. The forward speed was constant and less than 12 km/h.
3. The wheels of the tractor operated with zero slip angles.
4. The turning response of the tractor was mainly dependent upon the tractor kinematic behavior.
5. The rear mounted implement was fixed rigidly to the tractor.
6. The angular velocity to steer the front wheel was constant.
7. The maximum front wheel angle was 20° in either direction.
8. All angles that turn to counter-clockwise were considered positive.

The accuracy of steering control with the algorithm depends upon accuracy of position measurement, error sampling rate, and tractor forward speed. The tractor forward speed, the angular velocity to steer the front wheel and the error sampling interval were assumed to be constant. The magnitude of the front wheel turning ratio and distance interval between position measurements were found as follows:

\[ \gamma = \frac{\psi}{V} \]  \hspace{1cm} (5)

\[ s = V \tau \]  \hspace{1cm} (6)
where
\[ \gamma \] = magnitude of front wheel turning ratio (rad/m)
\[ s \] = distance interval for each sampling (m)
\[ \psi \] = angular velocity to steer front wheel (rad/s)
\[ V \] = tractor forward speed (m/s)
\[ t \] = sampling interval (s)

**Kinematic Behavior of Tractor Movement**

The front wheel angle is a main input variable for the tractor movement. The front wheel angle is the angle of the front wheel with respect to the longitudinal axis of the tractor. The front wheel turning ratio (\( \alpha \)) is \( \pm \gamma \) or \(-\gamma \) when the tractor turns left or right, respectively. The front wheel angle change depends upon the front wheel turning ratio (\( \alpha \)) and distance steered (\( z \)), and can be described as:

\[ \delta_{i+1} = \alpha z + \delta_i \]  

(7)

where
\[ \delta \] = front wheel angle before (i) and after (i+1) steering (rad)
\[ \alpha \] = front wheel turning ratio (rad/m)
\[ z \] = travel distance while front wheel is steered (m)

The kinematic equations for the tractor movement can be developed based on tractor geometry with assumption of no slip of the tractor wheels. From the tractor geometry as shown in Figure 4, the front wheel moved from the position \( F_i \) to \( F_{i+1} \) as the front wheel angle changed. The x-y coordinates for these points were defined with respect to the same
FIGURE 4. Front wheel movement, from $F_i$ to $F_{i+1}$, depends upon front wheel angle ($\delta$), steering angle ($\alpha z$), and tractor yaw angle ($\theta$). Front wheel moves $\Delta F_p$ in P direction and $\Delta F_q$ in Q direction.
coordinates used for the position sensing system. The displacement of the front wheel depends upon the tractor yaw angle ($\theta_i$), the current front wheel angle, and the steering angle. The tractor yaw angle was defined as the angle of the longitudinal axis of the tractor with respect to the positive x-coordinate axis and computed from the positions of the front wheel and the implement. The following equations were found to describe the movement of the front wheel:

$$\tan (\theta_i) = \frac{(F_y - I_y)}{(F_x - I_x)}$$  \hspace{1cm} (8)
$$F_{x_{i+1}} = z \cos(z \alpha/2 + \delta_i + \theta_i) + F_{x_i}$$  \hspace{1cm} (9)
$$F_{y_{i+1}} = z \sin(z \alpha/2 + \delta_i + \theta_i) + F_{y_i}$$  \hspace{1cm} (10)

Owen (1982) pointed out that a two-wheel drive tractor can be treated as a simple vehicle without suspension system under low travel speed conditions (up to about 12.6 km/h). When the tractor operated at low travel speeds, the curvature response was governed by the geometry of the tractor and the tractor yaw response was forward speed dependent. The tractor yaw increased linearly with the increase of forward speed under low travel speed conditions. Equation 3 was modified to calculate the tractor yaw angle as the front wheel angle changed. The x-y coordinates of implement position on the tractor centerline were found from the tractor geometry. The following equations were used to compute the tractor yaw angle and the implement position:

$$\theta_{i+1} = z (z \alpha/2 + \delta_i)/L + \theta_i$$  \hspace{1cm} (11)
$$I_{x_{i+1}} = F_{x_{i+1}} - H \cos(\theta_{i+1})$$  \hspace{1cm} (12)
$$I_{y_{i+1}} = F_{y_{i+1}} - H \sin(\theta_{i+1})$$  \hspace{1cm} (13)
where

\[ \theta = \text{tractor yaw angle (rad)} \]
\[ \delta = \text{front wheel angle (rad)} \]
\[ \alpha = \text{front wheel turning ratio (rad/m)} \]
\[ z = \text{travel distance while front wheel is steered (m)} \]
\[ L = \text{effective wheel base length (m)} \]
\[ H = \text{length between front wheel and implement (m)} \]

Computation of Steering Angle

The computation of the desired steering angle was initiated with measurement of the positions of the front wheel \((F_x, F_y)\) and the implement \((I_x, I_y)\) by use of the position sensing systems. It should be noted that the p-q coordinate system was aligned with the longitudinal axis of the tractor and was rotated by the tractor yaw angle \(\theta_i\) from the general x-y coordinate system. The positions of the front wheel and the implement in the x-y coordinate system were converted to those in the p-q coordinate system, \((F_p, F_q)\) and \((I_p, I_q)\). The general forward direction of the tractor movement was in the positive p-coordinate direction. From the geometry as illustrated in Figure 4, the displacement of the front wheel, \(\Delta F_p\) in p direction and \(\Delta F_q\) in q direction, can be described as follows:

\[ \Delta F_p = z \cos(z \alpha/2 + \delta_i) \]  \hspace{1cm} (14)
\[ \Delta F_q = z \sin(z \alpha/2 + \delta_i) \]  \hspace{1cm} (15)
The front wheel angle was assumed to be less than 20° in either direction. There is 2% difference between \( \sin(20^\circ) \) and 20° in radian. Therefore, Equation 15 can be simplified as:

\[
\Delta F_{q_i} = z^2 \frac{\alpha}{2} + z \delta_i
\]  

(16)

The lateral position error was defined as the deviation from the predetermined path in the \( q \)-coordinate direction. As shown in Figure 5, the desired positions of the front wheel (P1) and the implement (P0) were determined at the equivalent \( p \)-coordinates of the front wheel (Fp) and the implement (Ip), respectively. From the predetermined path stored in the computer memory, the \( q \)-coordinates of the points P0 and P1, P0q and Plq, were computed, respectively. The position errors of the front wheel (\( e_F \)) and the implement (\( e_I \)) can be expressed as:

\[
e_F = P_{lq} - F_q
\]  

(17)

\[
e_I = P_{0q} - I_q
\]  

(18)

The tractor would travel the distance interval (s) in the positive \( p \)-coordinate direction until the next measurement. Equation 14 shows that the front wheel movement in the \( p \)-coordinate direction depends upon the current front wheel angle, the direction of turn, and the distance steered. When the tractor travels without front wheel angle change, the "looking-ahead" next two measurement positions, P2 and P3, can be found from the path data as shown in Figure 5. By modifying Equation 14, the "looking-ahead" next two positions, P2 and P3, were:
FIGURE 5. Lateral position errors ($e_I$, $e_P$, and $e_{P_2}$) at implement, front wheel, and looking-ahead position, respectively. Slope difference ($\phi_i$) can be found between tractor yaw angle ($\theta$) and slope from $P_1$ to $P_3$. 
The purpose for steering the front wheel was to eliminate the lateral position error of the front wheel at the next measurement position, $P_2$, not at the current position, $P_1$. The position error ($e_{p2}$) to determine the steering angle was the deviation from the point $P_2$ in the $q$-coordinate direction and found as:

$$e_{p2} = P_2q - F_q$$

(21)

The lateral position error ($e_{p2}$) of the front wheel at the point $P_2$ was not the only factor used to determine the proper steering angle. The difference between the tractor yaw angle and the curvature of the predetermined path should be considered. Although there were no position errors, the angles of the front wheel might not always be zero. The slope difference ($\phi_1$) between the tractor yaw angle and the slope from point $P_1$ to $P_3$ was shown in Figure 5 and found as:

$$\phi_1 = \delta_1 - \tan^{-1}\left(\frac{P_3y-P_1y}{P_3x-P_1x}\right)$$

(22)

Three points ($P_1$, $P_2$, and $P_3$) were used to compute the angle of the curvature of the path. From these points, it was determined whether the predetermined path was straight or curved. When the path was considered as the straight line, the angle of the curvature of the path was zero. When the path was considered as the curve, the radius of the segment was computed and the angle of the curvature was calculated.
The slip angles of the tractor wheels were assumed to be zero. Therefore, the tractor, operating at low speed, can be considered as a neutral steered \(K_{us} = 0\) vehicle in Equation 2. The front wheel angle \(\delta_{cv}\) of the curvature of the predetermined path can be simplified as:

\[
\delta_{cv} = \frac{L}{R - WS/2}
\]

(23)

where

- \(L\) = effective wheel base length (m)
- \(R\) = radius of path segment (m)
- \(WS\) = tread width of tractor (m)

The position error \(e_{p2}\) of the front wheel at the point P2, the current front wheel angle, and the angle difference between the tractor yaw and the slope of the curvature were considered to determine the steering angle. The current front wheel angle was the combination of the angle due to position error \(\delta_i - \delta_{cv}\) and the angle of curvature of the predetermined path \(\delta_{cv}\). The proper front wheel angle should be the same as the angle of curvature of the predetermined path \(\delta_{cv}\) without any position errors. By modifying Equation 16, the expected position error of the front wheel at point P2 can be predicted after the tractor has been steered. Equation 11 and Equation 22 shows that the slope difference between the tractor yaw and the predetermined path changed as the front wheel steered. The following modified equations were used to determine the steering angle:

\[
e_{new} = e_{p2} - z (x/2 + \delta_i)
\]

(24)
\[
\phi_{\text{new}} = \phi_i - \frac{z}{L} (\frac{\alpha}{2} + \delta_i)
\]

\[
\delta_{\text{new}} = \delta_{cv} - (z \alpha + \delta_i)
\]

The proper front wheel angle was determined to fulfill the following conditions at the point P2 after the front wheel was steered: (1) The position error of the front wheel equals zero \((e_{\text{new}} = 0)\); (2) The tractor yaw angle is the same as the slope of predetermined path \((\phi_{\text{new}} = 0)\); (3) The angle of the front wheel is the same as angle required to follow curve of predetermined path \((\delta_{\text{new}} = 0)\). All conditions described could not be satisfied simultaneously. The primary condition to determine steering angle was to reduce position error. The value of the wheel turning ratio \((\alpha)\) could be \(+\gamma, -\gamma, \text{or } 0\) when the front wheel steered left, right, or straight, respectively.

To reduce the new position error \((e_{\text{new}})\) in Equation 24, the sign of the front wheel turning ratio \((\alpha)\) might be assigned the sign of the position error \((e_{p2})\). Although the sign of the front wheel turning ratio \((\alpha)\) was given the opposite sign of the position error, the new position error \((e_{\text{new}})\) could be decreased when the sign of current front wheel angle was the same as that of position error \((e_{p2})\). Therefore, the sign of the front wheel turning ratio \((\alpha)\) was given the sign of the position error when the current front wheel angle had the opposite sign of the position error as shown in Figure 6.

When the current front wheel angle had the opposite sign of the position error, the new position error \((e_{\text{new}})\) could be decreased without front wheel change. However, the tractor yaw deviated from the slope of the path-to-follow when the signs of the current front wheel angle and
FIGURE 6. Determination of turning direction based upon position error ($e_{p2}$), front wheel angle ($\phi$), and slope difference ($\delta$). Error will be increased without steering tractor when signs of position error and slope difference are different.
the slope difference ($\phi_1$) were different and was close to that when the
signs of the current front wheel angle and the slope difference ($\phi_1$) were
same as shown in Figure 6. The appropriate value for variable $z$ in
Equation 24 was used to determine the turn direction when the tractor
traveled straight ($\alpha=0$).

When signs of the position error ($e_{p2}$) and slope difference ($\phi_1$)
were the same, the greater value of the distance interval and the
distance required to return to the angle due to curvature ($\delta_{\text{new}}=0$) in
Equation 26 was used. Otherwise, the summation of the distance required
to return to the angle due to curvature ($\delta_{\text{new}}=0$) in Equation 26 and the
distance required to eliminate the slope difference ($\phi_{\text{new}}=0$) in Equation
25 was used. By substituting this value for variable $z$ in Equation 24
with $\alpha=0$, the sign of the front wheel turning ratio ($\alpha$) was given the
sign of the position error ($e_{p2}$) when the new position error ($e_{\text{new}}$) had
the same sign as the position error ($e_{p2}$).

The proper front wheel turning ratio was substituted in Equation 24
to find the distance required to eliminate the position errors ($e_{\text{new}}=0$).
The angle to steer the front wheel was found by multiplying this distance
by the front wheel turning ratio. The maximum steering angle for each
sampling interval was the product of the front wheel turning ratio and
the distance interval ($\alpha s$). The front wheel angle could not exceed 20°
in either direction.
Computer Simulation

The steering control algorithm was evaluated through use of computer simulation. A Zenith-159 microcomputer was used. A simulation program, written in C Language, was based upon the kinematic equations for the steering control algorithm. The program consists of several user defined functions and subroutines to read desired path data, to calculate errors, to determine amount of front wheel angle to steer, to find new tractor position and front wheel angle, and to print the results. The main program for computer simulation is listed in Appendix A.

The program simulates tractor movement and position measurements. Finite distance intervals were used rather than time increments. A flowchart of the simulation program is shown in Figure 7. The program read the x-y coordinates of the path, the front wheel turning ratio, and the distance interval as input variables. From the given locations of the front wheel and the implement, the control algorithm was used to calculate the errors and the proper steering angle. The program simulated the response of the tractor as the front wheel was moved. A function, generating uniform random numbers, was used to simulate the measurement error of the position sensing system. The new locations of the front wheel and the implement, the tractor yaw angle, and the front wheel angle were found at the end of each interval increment. The process was repeated until the end of path data was reached.

The path predicted from computer simulation deviated little from the desired path. The absolute maximum error, the RMS (Root Mean Squared) error, and the percentage of calculated points where absolute error was
FIGURE 7. Flowchart of the computer simulation program
greater than 5 cm were used as indices to evaluate the success in following the desired paths. The position errors of the front wheel and the implement were found at each calculated point. These errors were recorded, counted when the absolute error was greater than 5 cm, squared, and then accumulated. At the end of the run, the RMS error and the percentage of calculated points where absolute error was greater than 5 cm for the front wheel and for the implement were printed. The RMS error was calculated from the following equation:

$$e_{\text{rms}} = \sqrt{\frac{\sum e^2}{N}}$$  \hspace{1cm} (27)

where \(N\) is the number of points measured during test and \(e\) is the position error.

The algorithm was evaluated at different front wheel turning ratios (5, 10, 15, and 20 °/m) and position-measurement distance intervals (0.2, 0.4, 0.6, and 0.8 m). Two sinusoidal paths, two step paths, three arc paths (a quarter of a circle), and a 70-m straight-line path, as shown in Figure 8, were considered. The step paths begin 10 m before and end 40 m after the step functions with 1-m and 5-m magnitude. A 10-m amplitude and 100-m period and a 5-m amplitude and 50-m period sine wave functions were used as the sinusoidal paths. The arc paths had radii of 100 m, 50 m, and 25 m. Each sinusoidal and arc path had a 10m straight line before and after the given curve. Path data, in the form of x-y coordinates, were used instead of the equations of the desired paths.
FIGURE 8. Paths considered for computer simulation
Field Experiments

The steering control algorithm was developed based upon the kinematic behavior of the tractor movement. The kinematic equations were found from the tractor geometric relationships. Tests were conducted to verify this algorithm under field conditions. The system was tested on a 92-m by 106-m clean tilled loam field at the Iowa State University Agricultural Engineering Research Center. The average slope of this plot was less than one percent. A John Deere 4430 tractor was equipped with the automatic guidance system. Instruments were contained in a three-point-hitch mounted housing.

The steering control algorithm required, as inputs, the tractor position and the tractor yaw angle. The guidance system was designed based upon two AGNAV units which were used to find positions of the front wheel and the implement. The tractor yaw angle was computed from two positions measured by the AGNAV units. The main antennas of the AGNAV units were mounted at the center of the tractor body, 0.58 m ahead of the center of the front axle and at the end of the three point hitch, 1.96 m behind the center of the rear axle. The tractor has a wheel base length of 2.69 m and a tread width of 1.65 m. The repeaters of each AGNAV unit had to be placed 100 m apart to reduce the signal interference between the units. Figure 9 shows the layout of the main antennas and repeaters in the field. The unit-1 was used for position measurement of the front wheel and unit-2 for implement.
FIGURE 9. Layout of the main antennas and repeaters in the field
The origin of the general coordinate for the path data was the position of the unit-1 repeater-B. Each AGNAV unit found x-y coordinates relative to its own repeater-B location. The measured coordinates from the unit-2 were converted to the general x-y coordinates which were defined relative to the position of repeater-B of unit-1. The tractor started at a known point where the general x-y coordinate was found by tape measurements. The position readings from unit-1 and unit-2 were adjusted to the measured coordinate at the starting point.

The field tests were prepared to verify the guidance system with two AGNAV units, however, signal interference between two AGNAV units was found. For this reason, a guidance system with one AGNAV was tested. With one AGNAV, the position of the front wheel was measured and the tractor yaw angle was computed based upon the position change of the front wheel from the beginning of the test. Because there is a position-measurement error with the AGNAV unit, the accumulated error of computed tractor yaw angle may be large. The error of the computed tractor yaw angle may not be as critical as the measurement error of the AGNAV unit because the test path was only 70 m long.

The guidance system was tested for three different paths, a 70-m straight-line path, a sinusoidal path with a 5-m amplitude and 50-m period, and a step path with a 5-m magnitude of the step function. The step path had 25-m straight line before and after the step function and the sinusoidal path had 10-m straight line before and after the curve. The straight-line path is typical of many farm operations and evaluated the system accuracy. The sinusoidal path evaluated the ability to follow
a curve in which the slope was changing continuously. The step path evaluated the stability of the control system after an abrupt movement.

The paths were flagged at 2.5 m intervals in the middle of the test plot. A metal rod attached at the end of the instrument housing made marks on the soil surface where the tractor traveled. The positions of the front wheel and the implement were recorded on the microcomputer disk during the test and the errors of the implement were measured manually at the flagged points after each test. A total of four replications for each path were run.

The field test program, Figure 10, was very similar to the simulation program. The simulation program found the new tractor position and the yaw angle based upon the kinematic equations. The field test program found the tractor positions from the AGNAV units. It was necessary to initialize the guidance system before each test. The field test program has a menu to change file names of the desired path data and output results, to set initial variables such as the steering velocity and sampling time, to test the front wheel angle measurement, to test the tractor speed measurement, to test stepping motor control, to test the position measurements of the AGNAV units, to compensate the x-y coordinates of the AGNAV measurements, and to make the field test run. Subroutines to calculate errors and to determine the steering angle were the same as in the simulation program. The main program for the field test is listed in Appendix B and the included files, the subroutines, and the user defined functions are listed in Appendix C.
FIGURE 10. Flowchart of the field test program
RESULTS AND DISCUSSION

Computer Simulation

The maximum absolute position error, the RMS error, and the percentage of the measured points where the absolute error was greater than 5 cm showed essentially the same response to increases in turning ratio and distance interval between measurements. The RMS error was a good indication of success in following the desired path. Figure 11 shows that the RMS error of the front wheel increased as the distance interval increased. The steering angle was computed at the beginning of each interval and the front wheel was turned and held constant at that angle for the distance interval. Therefore, the position error would be increased as the distance interval increased. As the radius of the arc path increased, the RMS error increased when the distance interval was less than 0.6 m. This response was not surprising because the algorithm needed more travel distance to approach the path but the travel distance was reduced as the radius of the path decreased. The simulated travel distance for the arc was proportional to the radius of the arc path.

The RMS error of the implement showed the same response as that of the front wheel (Figure 12). The RMS error of the implement was always greater than that of the front wheel because more travel distance was required for the implement, than the front wheel, to reach the desired path. The error increment with the radius decrease was not as great as that with the distance interval increase. To better follow the desired path, the distance interval should be reduced as curvature increases.
FIGURE 11. RMS error of front wheel as affected by measurement distance interval when following arc paths with radii (R) of 100 m, 50 m, and 25 m. Each line represents combinations of four turning ratios (5, 10, 15, and 20 deg/m).
FIGURE 12. RMS error of implement as affected by measurement distance interval when following arc paths with radii (R) of 100 m, 50 m, and 25 m. Each line represents combinations of four turning ratios (5, 10, 15, and 20 deg/m)
Figure 13 shows the RMS error of the implement as affected by the turning ratio of the front wheel. In the path with 25-m radius, the RMS error reduced as the turning ratio increased. When the turning ratio increased, the control system would easily follow the path with small radius. As the turning ratio increased, the distance required to steer the front wheel was reduced. If the turning ratio was too large for a given distance interval, the tractor would oscillate on the path by over-steering.

The simulation results showed almost the same response for arc paths and sinusoidal paths. Figure 14 shows the RMS error of the implement when a sinusoidal path with a 10-m amplitude and a 100-m period was used. When the distance interval was large, the RMS error increased as the turning ratio increased. With the 0.2-m distance interval, the implement error decreased, but not significantly, as the turning ratio increased. The results indicated that reducing distance interval between measurements should be a primary consideration for improving the accuracy of the guidance system. As the tractor travel speed increased, the distance interval increased and the turning ratio of the front wheel decreased. To control steering wheel successfully, the allowable tractor travel speed should be determined based upon the error sampling time interval and the steering rate of the guidance system.

Figure 15 shows the percentage of the measured points where the absolute error of the front wheel was greater than 5 cm as affected by the distance interval. When the distance interval was greater, the front wheel oscillated frequently on the path. The front wheel would stay on
FIGURE 13. RMS error of implement as affected by turning ratio when following arc paths with radii (R) of 100 m, 50 m, and 25 m. Each line represents combinations of four distance intervals (0.2, 0.4, 0.6, and 0.8 m)
FIGURE 14. RMS error of implement when following a sinusoidal path with 10-m amplitude and 100-m period
FIGURE 15. Percentage of points where the absolute error of the front wheel was greater than 5 cm as affected by measurement distance interval when following sinusoidal paths with periods of 100 m and 50 m. Each line represents combinations of four turning ratios (5, 10, 15, and 20 deg/m).
the desired path when the distance interval was small. Figure 16 shows
the maximum absolute error of the implement as affected by the distance
interval. The front wheel followed the straight line and the implement
was still on the curved path when the tractor left the curve. The
situation was reversed when the tractor approached the curve. The
maximum absolute error was found when the tractor returned to the
straight line from the curve.

The simulation results for the step paths showed the stability of
the algorithm after an abrupt movement. The maximum absolute error of
the front wheel, the steady-state RMS error of the implement, and the
travel distance for the errors of the front wheel and the implement to
stabilize at less than 5 cm were found after step functions with 1-m and
5-m amplitudes as shown in Table 2. The tractor was not stable after the
step functions when either the distance interval or the turning ratio was
large. The tractor position error stabilized at less than 5 cm and the
travel distance required for the errors to stabilize was less than 25 m
when the distance interval between measurements was less than 0.6 m. The
steady-state RMS error was found when the tractor traveled 25 m after the
step function. The steady-state RMS error of the implement increased as
the distance interval increased. The maximum absolute error was found
after the front wheel passed the reference line. Because the tractor
steered to the maximum angle of the front wheel to follow the step path
with large amplitude, it took more distance to return to the reference
line and the maximum absolute error of the front wheel increased as the
amplitude of step function increased.
FIGURE 16. Maximum absolute error of implement as affected by measurement distance interval when following sinusoidal paths with periods of 100 m and 50 m. Each line represents combinations of four turning ratios (5, 10, 15, and 20 deg/m)
TABLE 2. Absolute maximum error of the front wheel, steady-state RMS error of the implement, and required-travel-distance when both position errors were less than 5 cm after step functions with 1-m and 5-m amplitude

<table>
<thead>
<tr>
<th>Turn Ratio</th>
<th>Required Travel Distance</th>
<th>Absolute Max. Error</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-m</td>
<td>5-m</td>
<td>1-m</td>
</tr>
<tr>
<td>deg/m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>5</td>
<td>0.2</td>
<td>10.2</td>
<td>24.3</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>10.6</td>
<td>24.2</td>
</tr>
<tr>
<td>5</td>
<td>0.6</td>
<td>31.7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>11.2</td>
<td>21.0</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>11.3</td>
<td>20.7</td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.2</td>
<td>11.1</td>
<td>21.1</td>
</tr>
<tr>
<td>15</td>
<td>0.4</td>
<td>10.9</td>
<td>21.4</td>
</tr>
<tr>
<td>15</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.2</td>
<td>10.9</td>
<td>22.1</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>12.1</td>
<td>21.8</td>
</tr>
<tr>
<td>20</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*aTractor position error did not stabilize at less than 5 cm.*
The results of the computer simulation indicated that the distance interval between position measurements of the guidance system was the major factor affecting guidance error. When the system tried to follow the sharp curved path, the turning ratio of the front wheel also increased. When the turning ratio increased without the distance interval reduction, the tractor oscillated frequently about the desired path.

Field Experiments

Guidance System with two AGNAV units

The guidance system with two AGNAV units took about 0.8 second to take corrective action. The angular velocity to steer the front wheel was set at 4.5 °/s. The system was tested at the low tractor forward speed of 1.8 km/h. From Equations 5 and 6, it was found that the guidance system with two AGNAV units had the magnitude of the front wheel turning ratio of 9 °/m and the distance interval of 0.4 m during the tests.

Figures 17 and 18 show the tractor movements, with two AGNAV units, when following the straight-line path and the sinusoidal path with 5-m amplitude and 50-m period, respectively. The position error increased as the tractor traveled along the path. It was found that position measurements with the AGNAV unit had errors of up to the 50 cm and the errors were not constant.

Errors in position measurement with the AGNAV unit were larger when the main antenna was close to its own repeater-B. When the tractor traveled forward, the main antenna of the unit-1 was moved away from own
FIGURE 17. Tractor movement, with two AGNAV units, when following a straight-line path.
FIGURE 18. Tractor movement, with two AGNAV units, when following a sinusoidal path with a 5-m amplitude and a 50-m period.
repeater-B and the main antenna of the unit-2 came closer to own repeater-B. Thus, the measurement error of the unit-1 decreased and that of the unit-2 increased during the test. This tendency was not uniform due to the signal interference between units and it was difficult to compensate for the AGNAV measurement errors.

Guidance System with one AGNAV unit

The sampling interval of one AGNAV unit was about 0.5 second. The tractor traveled at an average speed of 1.7 km/h during tests. A front wheel turning ratio of 9.5 °/m and a measurement distance interval of 0.24 m was used. The position error of the front wheel recorded by the microcomputer included the measurement error of the AGNAV unit. The position error of the implement recorded by the microcomputer included the accumulated computation error of the tractor yaw angle in addition to the measurement error of the AGNAV unit. Measurement with the tape showed the actual movement of the implement. The microcomputer measured positions and calculated errors at more than 270 points during each test and errors were measured with a tape at 27 flagged points after each test.

Figure 19 shows the tractor movement, with one AGNAV, when following a straight-line path. The straight-line path was parallel to the x-coordinate and the tractor traveled in the positive x-direction during tests. The position errors in the x-direction did not affect to determine the steering angles. The tractor oscillated on the path due to the position-measurement error of the AGNAV unit. More than 80 percent
FIGURE 19. Tractor movement, with one AGNAV unit, when following a straight-line path.
of the absolute errors measured on the straight-line path were less than 50 cm.

Table 3 shows the maximum absolute error and the RMS error found by use of tape measurements, the AGNAV measurements recorded with computer, and the computer simulation. The computer simulation with simulated position-measurement error of 50 cm had an RMS error similar in magnitude to that measured with AGNAV unit. There was less than 3 cm difference between the measured RMS error and the RMS error from computer readings. The computer simulation had less variability, smaller standard deviations, among runs than did the tape measurements or the computer readings.

Figure 20 shows the tractor movement, with one AGNAV unit, when following the sinusoidal path with 5-m amplitude and 50-m period. The control algorithm determined the proper steering angle from the error of the "looking-ahead" next position. The "looking-ahead" next segment of the curve was the only factor considered about the path. If the curvature changes suddenly, more distance would be required to reach the desired path. A large error was found when the tractor approached or left the curve. More than 92 percent of the measured errors of the front wheel and 75 percent of the measured errors of the implement were within 50 cm of the sinusoidal path.

The position errors when following the sinusoidal path are shown in Table 4. The RMS error with tape measurement was similar to that with the computer simulation. The maximum absolute error and the RMS error on the sinusoidal path were not greater than those on the straight-line
TABLE 3. Mean and standard deviation of absolute maximum error and RMS error when following a straight-line path. Values represented are the average of four replications.

<table>
<thead>
<tr>
<th>Method</th>
<th>Implement</th>
<th>Front Wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max cm</td>
<td>RMS cm</td>
</tr>
<tr>
<td>Tape Measurement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>99.0</td>
<td>40.9</td>
</tr>
<tr>
<td>Std.</td>
<td>25.0</td>
<td>4.8</td>
</tr>
<tr>
<td>Computer Reading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>101.3</td>
<td>38.3</td>
</tr>
<tr>
<td>Std.</td>
<td>28.4</td>
<td>9.2</td>
</tr>
<tr>
<td>Computer Simulation^</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>79.6</td>
<td>34.0</td>
</tr>
<tr>
<td>Std.</td>
<td>2.6</td>
<td>0.7</td>
</tr>
</tbody>
</table>

^Program simulated position-measurement error of 50 cm.
FIGURE 20. Tractor movement, with one AGNAV unit, when following a sinusoidal path with a 5-m amplitude and a 50-m period
TABLE 4. Mean and standard deviation of absolute maximum error and RMS error when following a sinusoidal path with 5-m amplitude and 50-m period. Values represented are the average of four replications.

<table>
<thead>
<tr>
<th>Method</th>
<th>Implement Mean</th>
<th>Implement Std.</th>
<th>Front Wheel Mean</th>
<th>Front Wheel Std.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cm</td>
<td>cm</td>
<td>cm</td>
<td>cm</td>
</tr>
<tr>
<td>Tape Measurement</td>
<td>78.3</td>
<td>34.8</td>
<td>10.2</td>
<td>6.1</td>
</tr>
<tr>
<td>Computer Reading</td>
<td>90.2</td>
<td>40.8</td>
<td>84.6</td>
<td>27.6</td>
</tr>
<tr>
<td>Computer Simulation^</td>
<td>84.9</td>
<td>32.7</td>
<td>90.9</td>
<td>33.5</td>
</tr>
</tbody>
</table>

^Program simulated position-measurement error of 50 cm.
path. Generally, the position error increased as the curvature increased. The results of the field tests indicated that the position-measurement error of the AGNAV unit was more critical than the difference in the curvatures.

The results of the computer simulation with simulated position-measurement error showed that the RMS error was about half of the maximum absolute error and more than 70 percent of the absolute errors were less than position-measurement error. The RMS errors, with simulated position-measurement error of 5 cm, were 3.8 cm at the front wheel and 3.6 cm at the implement when following the straight-line path, and 4.3 cm at the front wheel and 5.7 cm at the implement when following the sinusoidal path. To reduce the RMS error less than 5 cm, the measurement error of the position sensing system should be less than 5 cm when following the desired curve.

Figure 21 and Figure 22 shows the front wheel and the implement movements, respectively, with one AGNAV unit when following the path with a 5-m amplitude step function. As shown in Figure 21, the front wheel passed the 5-m reference line of the step path earlier in the field test than in the simulation. It was found that the front wheel angle was already changed to the left due to the position error before the step function. Thus the front wheel could be steered to maximum angle within short tractor travel distance.

The tractor yaw angle was computed based upon the position change of the front wheel. The kinematic equation, used for tractor yaw angle computation, was simplified based upon small steering angle. The tractor
FIGURE 21. Tractor front wheel movement, with one AGNAV unit, when following a 5-m step path.
FIGURE 22. Implement movement, with one AGNAV unit, when following a 5-m step path
yaw angle changed to more than 45° after the step function. The front wheel was steered to the maximum front wheel angle when following the step path. The computation of the tractor yaw angle, from the beginning of the test, may accumulate more error than small steering angle. The "looking-ahead" next position was determined based upon the tractor yaw angle and the measurement distance interval. The incorrect tractor yaw angle would affect determination of the proper steering angle. The forward direction of the microcomputer computation may not be the same as the actual forward direction in the field. It was observed that the implement oscillated on the path after the step function during the field tests, however, the implement positions of the computer readings did not reach to the 5-m reference line as shown in Figure 22. This error may come from the computation error of the tractor yaw angle during the field tests. The guidance system with one AGNAV unit would not be reliable for following a step path. The tractor yaw angle should be measured instead computed.
SUMMARY AND CONCLUSIONS

An automatic guidance system was designed that is able to steer a tractor along a predetermined path. The automatic guidance system consists of two position sensing systems to locate the tractor front wheel and the implement, a microcomputer to determine proper steering angle by analyzing tractor position error, a stepping motor to steer the tractor front wheel, a potentiometer to measure front wheel angle, a wheel speed transducer to measure tractor forward speed, and an I/O (input/output) interface system. The position sensing systems determined the location of the tractor in the field. The microcomputer calculated the tractor yaw angle, the lateral position error by comparing the present location to desired tractor position, and controlled the stepping motor to steer the front wheel. A control algorithm, based upon the kinematic behavior of the tractor movement, was developed to determine the steering angle that will reduce the lateral position error at next measurement position. The position error of the front wheel at the "looking-ahead" next measurement point, the current front wheel angle, and the angle difference between the tractor yaw and the slope of the desired path were considered to determine the steering angle.

The control algorithm was evaluated through use of computer simulation. The simulation program, written in C language, simulated the tractor movement. The absolute maximum error, the RMS (Root Mean Squared) error, and the percentage of calculated points where absolute error was greater than 5 cm were used as indices to evaluate the success in following the desired paths. The algorithm was evaluated at different
front wheel turning ratios (5, 10, 15, and 20 °/m) and position-measurement distance intervals (0.2, 0.4, 0.6, and 0.8 m). Paths considered for the computer simulation were straight-line, step, sinusoidal, and arc paths. Path data, in the form of x-y coordinates, were used.

Field tests were conducted to verify the algorithm under field conditions. A John Deere 4430 tractor was equipped with the automatic guidance system. Instruments were contained in a three-point-hitch mounted housing. The guidance system used two AGNAV units to find positions of the front wheel and the implement. However, signal interference between the two units was occurred during the field experiments. A guidance system with one AGNAV unit was tested. The AGNAV unit was used to locate the position of the front wheel and the tractor yaw angle was computed based upon the position change of the front wheel from the beginning of the test.

The guidance system was tested for three different paths, a 70-m straight line, a sinusoid with 5-m amplitude and 50-m period, and a step with 5-m magnitude. The guidance system took about 0.8 second with two AGNAV units and 0.5 second with one AGNAV unit to take corrective action. The angular velocity to steer the front wheel was set at 4.5 °/s. The tractor traveled at the average speed of 1.7 km/h during test. The guidance system with one AGNAV unit had the magnitude of the front wheel turning ratio of 9.5 °/m and the distance interval of 0.24 m. The maximum absolute position error, the RMS error, and the percentage of the
measured points where the absolute error was greater than 50 cm were analyzed.

Following conclusions were drawn from this study:

1. The guidance system based upon AGNAV units was designed to record where the tractor traveled and to guide the tractor along the desired path.

2. The control algorithm was written to determine the proper steering angle. The algorithm was based upon constant travel speed, constant steering rate, and zero slip angles of the tractor wheels.

3. The simulation results showed that the distance interval between position measurements was an important factor to design the guidance system. The RMS (Root Mean Squared) error was less than 5 cm when the distance interval was 20 cm. To more closely follow path with sharp curvature, the turning ratio of the front wheel should be increased. If the turning ratio is too large for a given interval distance, the tractor would oscillate on the path by over-steering.

4. Position measurements with the AGNAV unit had errors of up to the 50 cm. More than 75 percent of the absolute errors measured on the desired path were less than 50 cm. The guidance system with one AGNAV unit took about 0.5 second to take corrective action. As the tractor travel speed increased, the sampling time should be reduced to maintain the proper distance interval.
To guide the tractor with acceptable precision for field operations, more accurate position and tractor yaw angle measurement, and faster error processing are required. An algorithm based upon a variable steering rate might improve the stability of the guidance system. The tractor yaw angle should be measured instead computed. The measurement error of the position sensing system should be reduced to less than 5 cm. To reduce the sampling time interval, the microcomputer must simultaneously measure the tractor positions, steer the front wheel, and compute the steering angle. The multi-tasking operation of the microcomputer or the main control microcomputer with several independently operated single-board microcomputers should improve the accuracy of the guidance system in field conditions.
REFERENCES


ACKNOWLEDGEMENTS

I wish to thank the United States Department of Agriculture and the Agricultural Engineering Department for providing facilities and resources for this research.

I wish to express my gratitude and appreciation to the following people:

Dr. Donald C. Erbach, major professor, I am grateful for his guidance, support, encouragement and patience throughout the entire course of the program.

Dr. Loren W. Zachary, for serving as the representative on the program committee for the minor in Engineering Science and Mechanics.

Dr. Richard J. Smith, for his helpful ideas, counsel, and for serving as a graduate committee member.

Dr. Stephen J. Marley and Dr. Robert Horton for help in their respective areas of expertise, and for serving as graduate committee members.

Scott Farris, Research Assistant, for his help in the field experiments and making figures with his favorite MAC computer.

Dr. Thomas Colvin, for his interest and suggestion to setup the AGNAV position sensing system.

Harold Mesenbrink, Richard VanDepol and Jeff Erb, for their help in the fabrication of research equipment.

Terry Walker, President of the Protocomp Inc., for his suggestion and advice to modify the PCA-48 control programs.
Han-Keun Cho, Brian Catus, and other graduate students in the department for their help, encouragement, and friendship.

I am indebted to my parents for their unlimited support, encouragement, and blessings.

And most importantly to my wife Kyung-Ae for always being there for me with her patience, support, and love and for taking care of my daughters, Sue-Hong and Jea-Hong, when I was not able to do my part.
APPENDIX A: MAIN PROGRAM FOR SIMULATION

/* Include files for TURBO-C Functions */
#include <conio.h>
#include <ctype.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* Include file shown in Appendix C */
#include <d:menu.h>

FILE *fopen(), *out, *infile;
char input_dat[15];
char output_dat[15] = "output.dat";
float agnav_err = 0., limit = 5.;

main()
{
    /* Initial variables */
    ix = iy = ft_ang = yaw = 0.;
    fax = fx = LENGTH;
    fay = fy = 0.;
    ft_max = ip_max = 0.;
    fe1 = fe2 = ie1 = ie2 = 0.;
    impt = err_num = ft_num = ip_num = 0;
    maxang = deg_to_rad(20.);
    randomize();

    /* Get turning ratio, distance interval from keyboard,
       Read X & Y data from the input data file,
       and open the output data file */
    ask();

    do {
        /* Find errors at front wheel & implement (ft_err, ip_err)
           and slope for next curvature from path data set */
        find_error();
        error_calc();
        /* Determine proper front wheel angle to steer */
        due_to_error();
    }
    while (0);
/* Simulate tractor based upon desired angles and
Find new positions, front wheel angle, and yaw angle
*/
tractor_move();

) while ((xy_p(fx,fy)+2.*step*cos(ft_ang)) <
        xy_p(x[dat_num],y[dat_num])); /* Check End of path */

find_result();
fclose(out);
APPENDIX B: MAIN PROGRAM FOR FIELD TEST

/* Include files for TURBO-C Functions */
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <dos.h>
#include <mem.h>
#include <conio.h>

/* Include files shown in Appendix C */
#include <d:menu,h>
#include <d:pca48,h>

/* Global Variables */
FILE *fopen(), *out1, *out2, *infile;
char input_name[15] = "input.in";
char outl_name[15] = "output.out";
char out2_name[15] = "test.dat";

main ()
{
    int choice, crow=4, ccol=10;
    unsigned old_cursor;

    /* Initialize system
       init_pca48() - Data Acquisition system
       cominit1() - AGNAV at front wheel
       cominit2() - AGNAV at implement
    */
    init_pca48(&acqdat);
    digoutput[0] = 0;
    digout();
    cominit1();
    cominit2();

    textmode(C80);           /* Set text mode */
    old_cursor = cursor(TCURSOROFF); /* Turn off cursor */
    for (;;) {
        choice = menu(crow, ccol, mnuMain); /* Select menu choices */
        switch (choice) {
            case QUIT :
                fcloseall();               /* Close all open files */
                cursor(old_cursor);       /* Set previous cursor */
                window(1,1,80,25);
                clrscr();                 /* Clear whole screen */
                textmode(LASTMODE);       /* Reset text mode */
                exit(0);                  /* Return to DOS system */
            } 
        } 
    } 
}
case RUN:
    control(); break;  /* Call control */
case CHANGE:
    get_data(); break;  /* Call get_data */
case INVAR:
    get_input(); break;  /* Call get_input */
case MOTOR:
    test_motor(); break;  /* Call test_motor */
case ANGLE:
    test_angle(); break;  /* Call test_angle */
case SPEED:
    test_speed(); break;  /* Call test_speed */
case ADJUST:
    adjust(); break;  /* Call adjust */
case AGNAV:
    test_agnav(); break;  /* Call test_agnav */
case AGNUM:
    get_agnum(); break;  /* Call get_agnum */
APPENDIX C: INCLUDE FILES, SUBROUTINES, AND FUNCTIONS

/**************************************************************/
#include <menu.h>
Variables and functions supporting for Main Programs
 **************************************************************/

/* Constants */
#define WIDTH 1.65 /* Tread Width 65 inch */
#define BASE 2.69 /* Wheelbase length 106 inch */
#define LENGTH 4.67 /* Length from front axle to implement */
#define IP_LEN 1.98 /* Length from rear axle to implement */
#define FT_LEN 0.56 /* Length from front axle to front ant */
#define MAXM 1000 /* Max. no. of x & y position data */
#define NULL 0
#define TCURSOROFF 0x2020

/* Key scan codes */
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define ENTER 13
#define ESC 27
#define SPACE 32

/* Functions */
int far keyrd();
int far keychk();
void far get_time();
void far cominit1();
void far cominit2();
void far comrd1();
void far comrd2();
void comrecv1();
void comrecv2();

int menu(int, int, char* []); /* Menus selection */
unsigned cursor(unsigned); /* Cursor Control */
double xy_p(double, double); /* Convert x-y cod. to p-q cod. */
double xy_q(double, double);
double curve(int, double); /* Find desired location from data */
double find_z(double); /* Find distance by solving eq. */
int point(int, double); /* Find location of input data set */

#define deg_to_rad(W) (3.142592654*(W)/180.)
#define rad_to_deg(W) (180.*(W)/3.142592654)
/* Array and enum for main menu */

char *mnuMain[] = {
    "Quit",
    "Run System",
    "Change Filename",
    "Set Input Var.",
    "Test Step Motor",
    "Test Front Angle",
    "Test Tractor Speed",
    "Test AGNAV",
    "Adjust AGNAV Data",
    "Set AGNAV Number",
    NULL
};

denum {QUIT, RUN, CHANGE, INVAR, MOTOR, ANGLE, SPEED, AGNAV, ADJUST, AGNUM};

/* Structure for menu attributes (variables for color and monochrome) */

struct mnuAtr {
    int fgOld, fgNormal, fgSelect, fgBorder;
    int bgOld, bgNormal, bgSelect, bgBorder;
    char nw[2], ne[2], se[2], sw[2], ns[2], ew[2];
} menus = {
    0, 3, 10, 8,
    7, 0, 4, 2,
    \xda, \xbf, \xd9, \xc0, \xb3, \xc4
};

char mess1[] = {
    "Move to menu selection with cursor keys, press ENTER to select"
};
char mess2[] = {
    "Press ESC key to quit"
};
char mess3[] = {
    "Press ESC key to do Nothing and then return to Main Menu"
};
char mess4[] = {
    "Use ARROW key to turn, SPACE key to stop, and ESC key to quit"
};

float x[MAXM], y[MAXM]; /* Predetermined path data set */
int hour, minute; /* Current Time */
float second, old_sec;
int agrav_num=2; /* Number of AGNAV connection*/
int dat_num=0; /* Total number of path data */
int impt=0.; /* Data set index for rear wheel */
double error=0.; /* Error for angle computation */
double ft_err=0.; /* Error at front wheel position */
double ip_err=0.; /* Error at implement position */
double ix=0., iy=0.; /* Implement position */
double rx=0., ry=0.; /* Rear axle position */
double fx=0., fy=0.;       /* Front axle position */
double fax=0., fay=0.;      /* Front antenna position */
double ft_ang=0.;           /* Present front wheel angle */
double old_ang=0.;          /* Previous front wheel angle */
double cv_ang=0.;            /* Front wheel angle due to curvature */
double st_ang;              /* Desired front wheel angle */
double maxang=deg_to_rad(20); /* Maximum front wheel angle */
double yaw=0.;              /* Present tractor yaw angle */
double dslop=0.;            /* Difference between yaw and path slope */
double ratio;               /* Front wheel turning ratio */
double step;                /* Distance interval for each reading */
float fx_adj=0., fy_adj=0.;  /* Adjust for front antenna */
float ix_adj=0., iy_adj=0.;  /* Adjust for Implement position */
float speed;                /* Tractor travel speed */
float turn=deg_to_rad(4.5);  /* Front wheel turning rate */
float sample=0.5;           /* Data sampling time interval */
float ang_zero=2.4735;       /* Set front angle zero angle */
float ang_amp=37.06;         /* Front angle conversion factor */
float speed_factor=0.0447;   /* Speed conversion factor */

/* Variables for Statistical Calculation */

double ft_max, ft_mean, ft_rms, ft_over;
double ip_max, ip_mean, ip_rms, ip_over;
double fel=0., fe2=0., iel=0., ie2=0.;
int err_num=0, ft_num=0, ip_num=0;

#include <pca48.h>
Variables and functions required to access
PC-ACQUISITOR from DIANACHART Inc.

#include <pca48.h>
Variables and functions required to access
PC-ACQUISITOR from DIANACHART Inc.

#include <pca48.h>
Variables and functions required to access
PC-ACQUISITOR from DIANACHART Inc.

/* Constants */
#define DACLOW 254
#define DACHIGH 253
#define SELECT 247
#define GAINSTB 251
#define ALATCH 2
#define BLATCH 6
#define CLATCH 0
#define PRINTER 4

/* Functions */
void far zeromes ();
void fast digout ();
void fast digin ();
void far counter ();
void far cntoff ();
void far init_pca48 ();
void far fulladf ();
void far oneadf ();
void far wordadf();
void far byteadf ();
void far sparel ();
void far datastbf ();
void far dataoutf ();
void dignear ();
void fullad ();
void onead ();
void wordad ();
void bytead ();
void datastb ();
void dataout ();
void parexitO;

struct addrptr {
    char *anaaddr;
    char *addr;
    char *bdata;
    char *result;
    char *maxchan;
    char *cdat;
    char *chan;
    char *gain;
    char *par;
    char *seldat;
    char *binin;
    char *binout;
    char *curchan;
    char *ampz. ro;
    char *spare;
};

static struct addrptr datptr;

    /* Initial Variables */
int anaddress = 10;
int address = 10;
int bdata = 1;
int result = 1;
int maxchan = 48;
int cdat[50];
int chan[50] = {15,15,15,15,15,15,15,15,15,15,
                15,15,15,15,15,15,15,15,15,15,
                15,15,15,15,15,15,15,15,15,15,
                15,15,15,15,15,15,15,15,15,15,
                15,15,15,15,15,15,15,15,15,15};

int gain[50];
int par = 888;
```c
int seldat48[50] = {0,1,17,33,49,65,81,97,113,
  2,18,34,50,66,82,98,114,
  3,19,35,51,67,83,99,115,
  4,20,36,52,68,84,100,116,
  5,21,37,53,69,85,101,117,
  6,22,38,54,70,86,102,118,7};

int bininput = 0;
int digoutput[8];
int cchan = 1;
int ampzero[22];
int spare = 0;

static int compdelay = 10;
static int ampdelay = 300;
static int parout = 0x888;
static int parin = 0x889;
static int parstat = 0x890;
static char bsave = 0;
static char gconvert[] = {0x00,0x10,0x20,0x30};

float scale[50];
float zero[50];
float treg[50] = {0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
  0.,0.,0.,0.,0.,0.,0.,0.,0.,0.};

struct addrptr acqdat = {
  (char *) &anaddress, (char *) &address,
  (char *) &bdata, (char *) &result, (char *) &maxchan,
  (char *) &cdat[0], (char *) &chan[0], (char *) &gain[0],
  (char *) &par, (char *) &seldat48[0], (char *) &bininput,
  (char *) &digoutput[0], (char *) &cchan,
  (char *) &ampzero[0], (char *) &spare
};
```
This is steering control program for field test. This program is similar to the simulation program.

```c
control()
{
    /* Initialize system and variables */
    ix = iy = ft_err = ip_err = 0.;
    ft_ang = cv_ang = dslop = yaw = 0.;
    impt = 0;

digoutput[0] = 0; /* Stop stepper motor turning */
digout();
    if (dat_num == 0)
        get_data(); /* Get desired path data */
    while (agnav_num >= 3) /* Set No. of AGNAV system */
        get_agnum();
    if ((agnav_num == 0) || (dat_num == 0))
        return;
    result_box(); /* Draw border for screen display */
    counter(); /* Reset counter for speed measurement*/
    get_time();
    old_sec = second; /* Measure front wheel angle */
    old_ang = ft_ang;

    while (keychk() != ESC) { /* Type ESC return to main menu */
        find_position(); /* Get tractor position from AGNAV */
        get_time(); /* Get current time */
        measure_speed(); /* Measure tractor speed */

        if (speed > 0.2) {
            ratio = turn/speed; /* Find turning ratio */
            step = sample * speed; /* Find distance interval */
            measure_angle(); /* Measure front wheel angle */
            find_error(); /* Find Position error */
            due_to_error(); /* Find steering Angle */
            tractor_steer(); /* Control stepper motor */
        }
        else {
            digoutput[0] = 0; /* If tractor is not moving */
            digout(); /* Stop stepper motor turning */
        }
        show_result(); /* Display result on screen */
    }
    digoutput[0] = 0; /* Stop stepper motor turning */
digout();
}
```
/******************************************************
  Find tractor position from AGNAV system
  With two AGNAVs - read AGNAV positions and
  calculate tractor yaw angle
  With one AGNAV - read front AGNAV position and find tractor
  yaw angle based upon front wheel movement
*******************************************************/

find_position()
{
  double oldx, oldy, px, py, temp=0.;

  if (agnav_num == 1) { /* Two AGNAV systems */
    comrd1();
    comrd2();
    yaw = atan2(fay-iy,ifax-ix);
    fx = fax - FT_LEN*cos(yaw);
    fy = fay - FT_LEN*sin(yaw);
  } else { /* One AGNAV system */
    oldx = fx;
    oldy = fy;
    comrd1();
    px = fax - FT_LEN*cos(yaw);
    py = fay - FT_LEN*sin(yaw);
    temp = xy_q(px-oldx, py-oldy);
    if (fabs(temp) >= step*maxang)
      temp = (temp>0.) ? step*maxang : -step*maxang;
    yaw += temp/BASE;
    fx = fax - FT_LEN*cos(yaw);
    fy = fay - FT_LEN*sin(yaw);
    ix = fx - LENGTH*cos(yaw);
    iy = fy - LENGTH*sin(yaw);
  }
}

/**************************************************************
  Find RMS error, absolute mean error and percentage of
  points where error is greater than "limit"
  Print summary of error computation
/***************************************************************/

find_result()
{
  ft_mean = fel/err_num;
  ft_rms = sqrt(fe2/err_num);
  ft_over = 100.*ft_num/err_num;
ip_mean = iel/err_num;
ip_rms  = sqrt(ie2/err_num);
ip_over = 100.*ip_num/err_num;

fprintf (out, "\n Result for Error Analysis \n");
fprintf (out, "\nMaximum Error (cm)\t Front - %8.3f\t Imp - %8.3f\n", 
              ft_max, ip_max);
fprintf (out, "Abs. Mean Error (cm)\t Front - %8.3f\t Imp - %8.3f\n", 
              ft_mean, ip_mean);
fprintf (out, "RMS Error (cm)\t Front - %8.3f\t Imp - %8.3f\n", 
              ft_rms, ip_rms);
fprintf (out, "Total No. of points for Error Computation = 
%4d\n", err_num);
fprintf (out, "Total No. of points more than error limit (%3.0f 
        cm)\n", limit);
fprintf (out, "\tFront = %4d (%6.2f \%)\t Imp = %4d (%6.2f \%)\n", 
              ft_num, ft_over, ip_num, ip_over);

printf ("\n Result for Error Analysis \n");
printf ("\nMaximum Error (cm)\t Front - %8.3f\t Imp = %8.3f\n", 
              ft_max);
printf ("Abs. Mean Error (cm)\t Front - %8.3f\t Imp = %8.3f\n", 
              ft_mean);
printf ("RMS Error (cm)\t Front - %8.3f\t Imp = %8.3f\n", ft_rms, 
              ip_rms);
printf ("Total No. of points for Error Computation = %4d\n", err_num);
printf ("Total No. of points more than error limit (%3.0f 
        cm)\n", limit);
printf ("\tFront = %4d (%6.2f \%)\t Imp = %4d (%6.2f \%)\n", 
              ft_num, ft_over, ip_num, ip_over);

/**************************************************************/
find_error()
{
    int   tk, pi, p2, p3;
double xp, f xp, ixp;

    /* Find errors at front wheel axle and implement position */
        ixp = xy_p(ix,iy);
impt = max(point(impt,ixp),0);

/**************************************************************/

fxp = ip + LENGTH;
tk = impt + LENGTH/xy_p(x[impt+1]-x[impt], y[impt+1]-y[impt]);
pl = point(tk,fxp);

ft_err = curve(pl,fxp) - xy_q(fx,fy);
ip_err = curve(impt,ip) - xy_q(ix,iy);

/* Find the next two positions from path data set, 
error at next expected position, 
the difference between tractor yaw angle and slope of curvature, 
and angle due to constant turn for curvature */
xp = step*cos(ft_ang);
fxp = xp + xp;
p2 = max(point(pl,fxp),pl+l);
p3 = max(point(p2,fxp+xp),p2+l);

error = curve(p2,fxp) - xy_q(fx,fy);
dslop = atan2(y[p3]-y[pl], x[p3]-x[pl]) - yaw;
due_to_curve (pl, p2, p3);

/**************************
Error was squared and accumulated for RMS error calculation
Absolute error was accumulated for mean error calculation
Find absolute maximum error and No. of point
when absolute error is greater than "limit"
***************************************************************************/

error_calc()
{
double ef, ei;

/* Print front, rear wheel & implement position and error */
printf("%7.3f %6.3f %5.1f %7.3f %6.3f %5.1f\n", fx, fy,
    100.*ft_err, ix, iy, 100.*ip_err);

    ef = 100.*fabs(ft_err);
    ft_max = max(ef,ft_max);
    fe1 = ef;
    fe2 = ef * ef;
    ei = 100.*fabs(ip_err);
    ip_max = max(ei,ip_max);
    ie1 = ei;
    ie2 = ei * ei;
    err_num++;
    if (ef > limit)
        ft_num++;
    if (ei > limit)
ip_num++;
}

).* Convert x-y coordinate to p-q coordinate
xy_p() : find value for p-coordinate
xy_q() : find value for q-coordinate
*******************************************************************************/

double xy_p(px, py)
double px, py;
{
    double z;
    z = px*cos(yaw) + py*sin(yaw);
    return (z);
}
double xy_q(px, py)
double px, py;
{
    double z;
    z = py*cos(yaw) - px*sin(yaw);
    return (z);
}

****************************************************************************
Find q-coordinate value at "fp" p-coordinate
Integer "tp" is segment location of path data set
*******************************************************************************/
double curve(tp, fp)
int tp;
double fp;
{
    double px, py, sp;
    if (fabs(x[tp+1]-x[tp]) > fabs(y[tp+1]-y[tp])) {
        sp = (y[tp+1] - y[tp])/(x[tp+1]-x[tp]);
        px = (fp - (y[tp]-sp*x[tp])*sin(yaw)) / (cos(yaw)+sp*sin(yaw));
        py = y[tp] + sp*(px-x[tp]);
    }
    else {
        sp = (x[tp+1] - x[tp])/(y[tp+1]-y[tp]);
        py = (fp - (x[tp]-sp*y[tp])*cos(yaw)) / (sin(yaw)+sp*cos(yaw));
        px = x[tp] + sp*(py-y[tp]);
    }
return (xy_q(px,py));
}

/************************************************************
   Find the segment from path data where "px" is located
   ************************************************************/
int point(pi,px)
int   pi;
double px;
{
    int    pk, ti, tm, pmin=pi, pmax=pi;
    pk = step/xy_p(x[pi+1]-x[pi], y[pi+1]-y[pi]);
    ti = max(pk,1);
    while (xy_p(x[pmin],y[pmin]) > px)
        pmin = max((pmin-ti),0);
    while (px >= xy_p(x[pmax],y[pmax]))
        pmax = min((pmax+ti),dat_num);
    while ( pmax != pmin+1) {
        tm = (pmin+pmax)/2;
        if (px >= xy_p(x[tm],y[tm]))
            pmin = tm;
        else
            pmax = tm;
    }
    return(pmin);
}

/************************************************************
   Find angle due to constant turn for curvature
   from the looking-ahead positions (pi, p2, & p3)
   ************************************************************/
due_to_curve (pi, p2, p3)
int   pi, p2, p3;
{
    double sp1, sp2, xm=0., ym=0., radius=0., cang;
    /* Find slop1 & slop2 for next segments */
    sp1 = sp2 = cv_ang = 0.;
    if (fabs(x[p2]-x[pi]) > 0.0001)
        sp1 = (y[p2] - y[pi])/(x[p2]-x[pi]);
    if (fabs(x[p3]-x[p2]) > 0.0001)
        sp2 = (y[p3] - y[p2])/(x[p3]-x[p2]);
/* If difference of slope is greater than 0.0001, it is considered as a curve path. Calculate wheel angle due to curvature (cv_ang) from the center of circle (xm,ym) and radius. */
if ((fabs(spl-sp2) > -0.0001) & (fabs(spl-sp2) < 100.)) {
    xm = 0.5 * ( spl*(x[p2]+x[p3]) - sp2*(x[p1]+x[p2]) - sp2*(y[p1]-y[p3]) ) / (spl-sp2);
    ym = 0.5 * ( x[p1] - x[p3] + spl*(y[p1]+y[p2]) - sp2*(y[p2]+y[p3]) ) / (spl-sp2);
    radius = hypot(x[p2]-xm, y[p2]-ym);
    cang = BASE / (radius-0.5*WIDTH);
    if (fabs(cang) < maxang)
        cvang = (sp2 > spl) ? cang : -cang;
}

/**********************************************************
Find steering angle to reduce error at next position
**********************************************************/

due_to_error ()
{
double a=0., er_ang, zx;
float sg;

dslop = (fabs(dslop) >- 0.005) ? dslop : 0. ;
sg = (error > 0.0) ? +1. : -1. ;
a = sg*ratio;
er_ang = ft_ang - cv_ang;
if (sg*dslop >= 0.)
    zx = max(fabs(er_ang)/ratio, step);
else
    zx = max((fabs(er_ang) + 2.*fabs(dslop*BASE))/ratio, step);
if (sg*(er_ang-dslop) < -0.001)
    st_ang = ft_ang + a * find_z(0.5*a);
else {
    if (sg * (error - zx*ft_ang) >= 0.)
        st_ang = ft_ang + a * find_z(0.5*a);
    else
        st_ang = cv_ang;
}

/**********************************************************
Find distance required to eliminate "looking-ahead" error when the turning ratio is "a"
**Equation 24 is used**

```c
double find_z(a)
double a;
{
  double  z=0., t1, t2, temp, b, c;

  b = ft_ang;
  c = -error;
  temp = b*b - 4.*a*c;
  if ((a != 0.) && (temp >= 0.0)) {
    t1 = max (-0.5*(b - sqrt(temp))/a, 0.);
    t2 = max (-0.5*(b + sqrt(temp))/a, 0.);
    if ((t1 > 0.) && (t2 > 0.))
      z = min (t1, t2);
    else
      z = max (t1, t2);
  } return(z);
}
```

//******************************************************************************
Simulate tractor movement based upon desired angles
******************************************************************************

```c
tractor_move()
{
  double  ta=0., zx, tang;

  if (fabs(st_ang) > maxang)
    st_ang = (st_ang>0.) ? maxang : -maxang;

  tang = st_ang - ft_ang;
  zx = fabs(tang/ratio);

  if (zx < 0.01*step)
    wheel_position(0.,step);
  else {
    ta = (tang > 0.) ? ratio : -ratio;
    if (zx > 0.99*step)
      wheel_position(ta,step);
    else {
      wheel_position(ta,zx);
      wheel_position(0.,step-zx);
    }
  }
}
```
/*******************************************************************************
Find new positions, front wheel angle, and yaw angle
Turning ratio : a
Distance steered : xt
*******************************************************************************

wheel_position(a, xt)
double a, xt;
{
  double temp = 0.;
  temp = 0.5*a*xt + ft_ang;
  fax = xt * cos(temp + yaw);
  fay = xt * sin(temp + yaw);
  yaw = temp * xt / BASE;
  ft_ang = a*xt;
  fx = fax + (random(10)-5)*agnav_error/5.;
  fy = fay + (random(10)-5)*agnav_error/5.;
  ix = fx - LENGTH*cos(yaw);
  iy = fy - LENGTH*sin(yaw);
  ix = max(ix, 0.);
  fx = max(fx, 0.);
}

/*******************************************************************************
Control stepper motor to steer front wheel
Front wheel angle is less than "maxangle" and
steering angle is not greater than "a*step"
*******************************************************************************

tractor_steer()
{
  double zx, tang;
  if (fabs(st_ang) > maxang) 
    st_ang = (st_ang>0.) ? maxang : -maxang;
  measure_angle();
  old_ang = ft_ang;
  tang = st_ang - ft_ang;
  zx = fabs(tang/ratio);
  if (zx <= 0.05*step) {
    digoutput[0] = 0;
    digout();
  }
else {
    digoutput[0] = (tang > 0.) ? 3 : 2;
    digout();
    if (zx < 0.95*step)
        tractor_steer2(tang);
}

/*********************************************/
When front wheel angle is same as the steering angle
stop steering, otherwise keep turning
*********************************************/
tractor_steer2(a)
double a;
{
    measure_angle();
    if (a > 0.) {
        while (ft_ang < st_ang)
            measure_angle();
    } else {
        while (ft_ang > st_ang)
            measure_angle();
    }
    digoutput[0] = 0;
    digout();
}

/*********************************************/
Read X & Y data from the input data file
and open the output data file
Get turning ratio, distance interval from keyboard,
*********************************************/
ask()
{
    int ch, ans;
    float pa, pb;
    /* Find input data file-name and open it */
    do {
        printf ("\nType input data file name = ");
        scanf ("%s", input_dat);
        printf("\nInput Data File is %s", input_dat);
        printf("\nType output file name = ");
        scanf ("%s", output_dat);
        printf("\nOutput File is %s", output_dat);
        printf("\nType turning ratio = ");
        scanf ("%f", &pa);
        printf("\nTurning Ratio = %.2f", pa);
        printf("\nType distance interval = ");
        scanf ("%f", &pb);
        printf("\nDistance Interval = %.2f", pb);
        printf("\nFile ok? (y/n) = ");
        scanf ("%s", &ok);
        printf("\nFile name = %s", input_dat);
        if (lowercase(ok) != 'y')
            printf("\nFile name ok? (y/n) = ");
    } while (lowercase(ok) != 'y');
}

/*********************************************/
printf("\nIs this correct? (y/n) ");
do { ch=getche();
    } while (isspace(ch));
ans = isupper(ch) ? tolower(ch) : ch;
printf("\n");
if ((infile = fopen(input_dat,"r")) == NULL) {
    printf("\nERROR: Can not open file %s\n", input_dat);
    ans = 'n';
}
while (ans != 'y');

/* Read X & Y data from the input data file */
dat_num = 0;
while (fscanf(infile,"%f%f", &pa, &pb) != EOF) {
    x[dat_num] = pa;
    y[dat_num] = pb;
    dat_num++;
}
fclose(infile);
dat_num--;
out = fopen(output_dat,"a");
do {
    printf("\nType Wheel turning ratio (deg/m) = ");
    scanf("%f", &turn);
    printf("\nDistance Interval for each reading (m) = ");
    scanf("%f", &sample);
    printf("\nPosition Measurement Error (m) = ");
    scanf("%f", &agnav_err);
    printf("\nWheel Turning Ratio -%5.1f (deg/m)\n",turn);
    printf("\nDistance Interval - %5.2f (m)\n", sample);
    printf("\nPosition Measurement Error - %4.2f (m)\n
", agnav_err);
    printf("\nIs this correct? (y/n) ");
    do ch=getche();
        while (isspace(ch));
    ans = isupper(ch) ? tolower(ch) : ch;
    printf("\n");
} while (ans != 'y');

ratio = deg_to_rad(turn);
step = sample;
limit = max(limit, 100.*agnav_err);

fprintf(out, "\n Input Data\n");
fprintf (out,"\nWheel Turning Ratio =%6.4f (deg/m)\n",turn);
fprintf (out,"Distance Interval = %5.2f (m)\n", sample);
fprintf (out,"Position Measurement Error = %4.2f (m)\n\n", agnav_err);
Change input and output data file names
Read path data and open output files

*******************************************************************************/

get_data()
{
float pa, pb;
int rowl=10, coll=20, row2=19, col2=75;
int ans = 0, len, wid, row, col;
char temp[80];

/* Draw menu to change file names */
len = row2 - rowl - 1;
wid = col2 - coll - 1;
window(coll,rowl,col2,row2);
textrbackground(menus.bgNormal);
crscr();
msgline(mess3); /* Write instruction at 25th line */
box(rowl,coll,len,wid);
row = rowl + 2;
col = coll + 2;
gotoxy(col,row);
cputs("0. Read data file and return to Main Menu");
gotoxy(col,row+1);
cputs("1. Input data file name");
gotoxy(col,row+2);
cputs("2. Output data file name");
gotoxy(col,row+3);
cputs("3. Test run result file name");
gotoxy(col,row+5);
cputs(" Enter No. to change data file name ");
memset(temp,' ', wid-1);
temp[wid-1] = NULL;
col = coll + 37;
row++;
cfcloseall();

/* Get file names and check to open */
Read X & Y data from the input data file
Open output files */
do {
    textcolor(menus.fgSelect);
gotoxy(col,row);
cprintf("%s",input_name);
gotoxy(col,row+1);
cprintf("%s",outl_name);
gotoxy(col,row+2);
cprintf("%s",out2_name);
if ( (ans >= '0') && (ans <= '3') ) {
    if (ans == '0') {
        dat_num = 0;
        clrscr();
        while (fscanf(infile, "%f%f", &pa, &pb) != EOF) {
            x[dat_num] = pa;
            y[dat_num] = pb;
            dat_num++;
        }
        dat_num--;  
        fclose(infile);
        ans = ESC;
    } else {
        gotoxy(col+row+6); cputs(temp); 
        gotoxy(col+row+6); cputs(" Type new data file name and RETURN ");
        gotoxy(col+row+ans-'1');
        cputs( " ");
        gotoxy(col+row+ans-'1');

        switch (ans) {
            case '1':
                scanf ("%s", input_name);
                if ((infile = fopen(input_name,"r")) == NULL) {
                    gotoxy(col+row+6); textcolor(menus.fgSelect | BLINK);
                    cputs(" ERROR : Can not open input data file");
                    msgline("Press ANY key to continue");
                    keyrd();
                    msgline(mess3);
                    gotoxy(col+row+6); cputs(temp);
                    break;
            case '2':
                scanf ("%s", outl_name);
                outl = fopen(outl_name,"a");
                break;
            case '3':
                scanf ("%s", out2_name);
                out2 = fopen(out2_name,"a");
                break;
        }
    }
}
/* ans - 0 : Read path data 
   ans - 1 : Change input file name and Open path data file 
   ans - 2,3: Change output file names and Open files */

test_motor()
{
  int rowl=10, col1=25, row2=16, col2=75;
  int ans = 0, len, wid, row, col;
  float pturn, onesecond;
  /* Draw box on screen */
  len = row2 - rowl - 1;
  wid = col2 - col1 - 1;
  window(col1,rowl,col2,row2);
  textbackground(menus.bgNormal);
  clrscr();
  msgline(mess4);
  box(rowl,col1,len,wid);
  row = rowl + 2;
  col = col1 + 2;
  gotoxy(col,row);
  cputs("Current Time");
  gotoxy(col,row+2);
  cputs("Front Wheel Angle (deg)";
  col = col1 + 30;
  textcolor(menus.fgSelect);
  get_time();
  old_sec = second;
  measure_angle();
  old_ang = ft_ang;

  /* Turn stepper motor based upon arrow key input
   Fill the result in the box
   Type ESC to return to main menu */
  while (ans != ESC) {
    break;
  }
}

/************************************************************
 Test stepper motor control with arrow key from keyboard
 Data shown in box on the screen are time, front wheel angle, and angular velocity
 *************************************************************/
get_time();
gotoxy(col,row);
cprintf("%2d : %2d : %5.2f", hour,minute,second);
measure_angle();
one sec = second - old sec;
if (one sec < 0.)
one sec += 60.;
if (one sec >= 1.0) {
    pturn = rad_to_deg(fabs(ft_ang-old_ang)) / one sec;
    old_ang = ft_ang;
    old sec = second;
}
gotoxy(col,row+2);
cprintf("%5.1f, %8.4f", rad_to_deg(ft_ang), pturn);

ans = keychk();
if ((ans != 0) && (ans != ESC)) {
gotoxy(col,row+3);
switch (ans) {
case RIGHT:
cputs("turn right");
digoutput[0] = 2;
b reak;
case LEFT:
cputs("turn left ");
digoutput[0] = 3;
b reak;
case SPACE:
cputs(" stop ");
digoutput[0] = 0;
b reak;
default:
b reak;
};
digout();
}
digoutput[0] = 0;
digout();

/***************************************************************************/
/* Test front wheel angle measurement and display on screen */
with time and output voltage from PCA-48 */
/***************************************************************************/
test_angle() {
    int row1=10, col1=25, row2=16, col2=75;
int len, wid, row, col;

/* Draw box on screen */
len = row2 - row1 - 1;
wid = col2 - coll - 1;
window(coll,row1,col2,row2);
textbackground(menus.bgNormal);
crscr();
msgline(mess2);
box(row1,coll,len,wid);
row = row1 + 2;
col = coll + 2;
gotoxy(col,row);
cputs ("Current Time");
gotoxy(col,row+2);
cputs ("Front Wheel Angle (deg)");
col = coll + 30;
textcolor(menus.fgSelect);

/* Measure front wheel angle through PCA-48 
 Fill the result in the box 
 Type ESC to return to main menu */

while (keychk() != ESC) {
    get_time();
gotoxy(col,row);
cprintf ("%2d : %2d : %5.2f", hour,minute,second);
measure_angle();
gotoxy(col,row+2);
cprintf ("%5.1f, %8.4f", rad_to_deg(ft_ang), treg[l]);
}

/**************************@
Test tractor speed measurement and display with time 
***************************/
test_speed()
{
    int row1=10, coll=25, row2=16, col2=75;
    int len, wid, row, col;

    /* Draw box on screen */
    len = row2 - row1 - 1;
    wid = col2 - coll - 1;
    window(coll,row1,col2,row2);
textbackground(menus.bgNormal);
crscr();
msgline(mess2);
box(row1, coll, len, wid);
row = row1 + 2;
col = coll + 2;
gotoxy(col, row);
cputs ("Current Time");
gotoxy(col, row+2);
cputs ("Travel Speed (km/h)"),
col = coll + 30;
textcolor(menus.fgSelect);
counter();
get_time();
old_sec = second;

/* Measure tractor speed and display in the box
Type ESC to return to main menu */
while (keychk() != ESC) {
    get_time();
    measure_speed();
gotoxy(col, row);
cprintf ("%d : %d : %5.2f", hour, minute, second);
gotoxy(col, row+2);
cprintf ("%6.2f", 3.6*speed);
}

/************************************************************
Compensate x-y coordinates measured by AGNAV system
Adjusted Values are found from the difference between
actual measurements and AGNAV readings
************************************************************/
adjust()
{
    int row1=10, coll=20, row2=19, col2=75;
    int ans = 0, len, wid, row, col;
    char temp[80];

    /* Draw box on screen */
    len = row2 - row1 - 1;
    wid = col2 - coll - 1;
    window(coll, row1, col2, row2);
textbackground(menus.bgNormal);
crsr();
msgline(mess2);
box(row1, coll, len, wid);
row = row1 + 2;
col = coll + 2;
gotoxy(col, row);
cputs("1. Value to adjust front-X ");
gotoxy(col,row+1);
cputs("2. Value to adjust front-Y ");
gotoxy(col,row+2);
cputs("3. Value to adjust Implement-X ");
gotoxy(col,row+3);
cputs("4. Value to adjust Implement-Y ");
gotoxy(col,row+5);
cputs(" Enter No. to change value to adjust position ");
memset(temp,' wid-1);
temp[wid-l] = NULL;
col = coll + 37;
textcolor(menus.fgSelect);

/* Get values for adjust and display in the box
Type ESC to return to main menu */
do {
    gotoxy(col,row);
cprintf("%-7.2f",fx_adj);
gotoxy(col,row+1);
cprintf("%-7.2f",fy_adj);
gotoxy(col,row+2);
cprintf("%-7.2f",ix_adj);
gotoxy(col,row+3);
cprintf("%-7.2f",iy_adj);
ans = keyrd();
    if ( (ans >= '1') && (ans <= '4') ) {
        gotoxy(coll+2,row+5);
cputs(temp);
gotoxy(col,row+ans-'1');
cputs(" ");
cputs(" ");
gotoxy(col,row+ans-'1');
    }
    switch (ans) {
    case '1':
        scanf("%f", &fx_adj); break;
    case '2':
        scanf("%f", &fy_adj); break;
    case '3':
        scanf("%f", &ix_adj); break;
    case '4':
        scanf("%f", &iy_adj); break;
    }
gotoxy(coll+2,row+5);
cputs(temp);
gotoxy(coll+2,row+5);
cputs(" Enter No. to change value to adjust position ");
}
}
while (ans !- ESC);
}

/*******************************************************************************/
Set input variables
*******************************************************************************/
get_input()
{
int rowl=9, col1=25, row2=20, col2=70;
int ans = 0, len, wid, row, col;
char temp[80];
float pang=20., pturn=4.5 ;

/* Draw box on screen */
    len = row2 - rowl - 1;
    wid = col2 - col1 - 1;
    window(col1,row1,col2,row2);
    textbackground(menus.bgNormal);
    clrscr();
    msgline(mess2);
    box(rowl, coll, len, wid);
    row = rowl + 2;
    col = col1 + 2;
    gotoxy(col,row);
    cputs("1. Max. front angle (deg) ");
    gotoxy(col,row+1);
    cputs("2. Turning rate (deg/s) ");
    gotoxy(col,row+2);
    cputs("3. Sampling interval (s) ");
    gotoxy(col,row+3);
    cputs("4. Set zero front angle (V) ");
    gotoxy(col,row+4);
    cputs("5. Front angle conversion ");
    gotoxy(col,row+5);
    cputs("6. Speed conversion ");
    gotoxy(col,row+7);
    cputs(" Enter No. to change value ");
    memset(temp,' ', wid-1);
    temp[wid-1] = NULL;
    col = coll + 32;
    textcolor(menus.fgSelect);

/* Select variable to change
Display constants of variables in the box
Type ESC to return to main menu
do {
    gotoxy(col,row);
cprintf("%7.2f",pang);
gotoxy(col,row+1);
cprintf("%7.2f",pturn);
gotoxy(col,row+2);
cprintf("%7.2f",sample);
gotoxy(col,row+3);
cprintf("%7.4f",ang_zero);
gotoxy(col,row+4);
cprintf("%7.3f",ang_amp);
gotoxy(col,row+5);
cprintf("%7.5f",speed_factor);
ans = keyrd();

if ( (ans >= '1') && (ans <= '6') ) {
    gotoxy(col+2,row+7);
cputs(temp);
gotoxy(col+2,row+7);
cputs(" Type new value and enter RETURN ");

gotoxy(col,row+ans-'1');
cputs(" ");
gotoxy(col,row+ans-'1');

switch (ans) {
    case '1':
        scanf("%f", &pang); break;
    case '2':
        scanf("%f", &pturn); break;
    case '3':
        scanf("%f", &sample); break;
    case '4':
        scanf("%f", &ang_zero); break;
    case '5':
        scanf("%f", &ang_amp); break;
    case '6':
        scanf("%f", &speed_factor); break;
}
gotoxy(col+2,row+7);
cputs(temp);
gotoxy(col+2,row+7);
cputs(" Enter No. to change value ");
}
} while (ans != ESC);
maxang = deg_to_rad(pang);
turn = deg_to_rad(pturn);
/*******************************************************************************
  Test position measurement of AGNAV systems
  Display on screen with time
*******************************************************************************

test_agnav()
{
  int row1=10, col1=25, row2=17, col2=75;
  int len, wid, row, col;

  fax = fay = ix = iy = 0.;

  /* Draw box on screen */
  len = row2 - row1 - 1;
  wid = col2 - col1 - 1;
  window(col1,row1,col2,row2);
  textbackground(menus.bgNormal);
  clrscr();
  msgline(mess2);
  box(row1,col1,len,wid);
  row = row1 + 2;
  col = col1 + 2;
  gotoxy(col,row);
  cputs ("Current Time");
  gotoxy(col,row+2);
  cputs ("Front Wheel Position (m)");
  gotoxy(col,row+3);
  cputs ("Implement Position (m)");
  col = col + 30;
  textcolor(menus,fgSelect);

  /* Measure tractor positions from AGNAV systems */
  comrd1 : front wheel reading
  comrd2 : implement reading
  Show the measurement in the box
  Type ESC to return to main menu
  */
  while (keychk() != ESC) {
    get_time();
    if (((agnav_num == 1) || (agnav_num == 2))
      comrd1();
    if (((agnav_num == 1) || (agnav_num == 3))
      comrd2();
    gotoxy(col,row);
    cprintf ("%2d : %2d : %5.2f", hour,minute,second);
    gotoxy(col,row+2);
    cprintf ("%7.2f : %7.2f", fax,fay);
    gotoxy(col,row+3);
    cprintf ("%7.2f : %7.2f", ix,iy);
fprintf(out2, "%2d:%2d:%5.2f %7.2f %7.2f %7.2f %7.2f\n",
    hour, minute, second, fax, fay, ix, iy);
}

/******************************************************************************/
Determine No. of AGNAV system used
/******************************************************************************/

get_agnum()
{
    int row1=10, col1=25, row2=19, col2=60;
    int ans = 0, len, wid, row, col;
    char temp[80];

    /* Draw box on screen */
    len = row2 - row1 - 1;
    wid = col2 - col1 - 1;
    window(col1,row1,col2,row2);
    textbackground(menus.bgNormal);
    clrscr();
    msgline("Type No. to change or ESC key to quit");
    box(row1,col1,len,wid);
    row = row1 + 2;
    col = col1 + 2;
    gotoxy(col,row);
    cputs ("0. No AGNAV connected ");
    gotoxy(col,row+1);
    cputs ("1. Both AGNAV connected ");
    gotoxy(col,row+2);
    cputs ("2. Front AGNAV connected ");
    gotoxy(col,row+3);
    cputs ("3. Implement AGNAV connected ");
    textcolor(menus.fgSelect);

    /* Chose current configuration and show in the box
       Type ESC to return to main menu */
    do {
        gotoxy(col+2,row+5);
        printf(" Current configuration is %2d ", agnav_num);
        ans = keyrd();
        if ( (ans >= '0') && (ans <= '3') )
            agnav_num = ans - '0';
    } while (ans != ESC);
}
Measure front wheel angle
channel No. 1 of PAC-48 is used
ang_zero : Output voltage when front wheel is straight
ang_amp : Conversion constant from voltage to degree

measure_angle()
{
    int i=1;

    cchan=1;
    oneadf();
    zeromes();

    treg[i] = (float) (cdat[i] - ampzero[2*gain[i]]) / 
            (3276.7 * (float) powl0(gain[i]));
    ft_ang = deg_to_rad(ang_amp*(treg[i]-ang_zero));
}

Measure tractor speed (km/h) every second
counter of PAC-48 counts No. of pulse from speed transducer
speed_factor : Conversion constant from counted No. to speed

measure_speed()
{
    float onesec=0.;

    onesec = second - old_sec;
    if (onesec < 0.)
        onesec += 60.;
    if (onesec >= 1.0) {
        counter();
        speed = (float) (bininput & 2047) * speed_factor / onesec;
        old_sec = second;
    }
}

Put menu on screen.
Starting <row> and <column>.
Array of menu <items> strings.
Global structure variable <menus> determines:
    Colors of border, normal items, and selected item.
    Border characters.
Returns number of item selected.

int menu(row, col, items)
int row, col;
char *items[];
{
int i, mn_num, ln_max = 2, prev, curr = 0, choice;
int litem[25];

msgline(messl);
textbackground(menus.bgNormal);
crsr();

/* Count items, find longest, and put length of each in array */

for (mn_num = 0; items[mn_num]; mn_num++) {
litem[mn_num] = strlen(items[mn_num]);
    ln_max = (litem[mn_num] > ln_max) ? litem[mn_num] : ln_max;
}
ln_max += 2;
box(row++,col++,mn_num,ln_max); /* Draw menu box */

for (i = 0; i < mn_num; ++i) { /* Put items in menu */
if (i == curr) {
textcolor(menus.fgSelect);
textbackground(menus.bgSelect);
} else {
textcolor(menus.fgNormal);
textbackground(menus.bgNormal);
}
itemize(row+i,col,items[i],ln_max - litem[i]);
}

for (;;) {
switch (keyrd()) { /* Get selection from keyboard */
case UP :
    prev = curr;
    curr = (curr > 0) ? (--curr % mn_num) : mn_num-1;
    break;
case DOWN :
    prev = curr;
    curr = (curr < mn_num) ? (++curr % mn_num) : 0;
    break;
case ENTER :
    textbackground(menus.bgNormal);
    return(curr);
default :
    continue;
}
textcolor(menus.fgSelect);
textbackground(menus.bgSelect);
itemize(row+curr,col,items[curr],ln_max - litem[curr]);
textcolor(menus.fgNormal);
textbackground(menus.bgNormal);
itemize(row+prev,col,items[prev],ln_max - litem[prev]);
}

/************************************************************
Draw menu box.
<row> and <col> are upper left of box.
<hi> and <wid> are height and width.
*************************************************************/

box(row, col, hi, wid)
int row, col, hi, wid;
{
    int i;
char temp[80];

textcolor(menus.fgBorder);
textbackground(menus.bgBorder);
gotoxy(col,row);
    temp[0] = *menus.nw;
    memset(temp+1,*menus.ew,wid);
    temp[wid+1] = *menus.ne;
    temp[wid+2] = NULL;
cputs(temp);

    for (i = 1; i <= hi; ++i) {
        gotoxy(col,row+i);
        cputs(menus.ns);
        gotoxy(col+wid+1,row+i);
        cputs(menus.ns);
    }
    gotoxy(col,row+hi+1);
    temp[0] = *menus.sw;
    memset(temp+1,*menus.ew,wid);
    temp[wid+1] = *menus.se;
    temp[wid+2] = NULL;
cputs(temp);
    textcolor(menus.fgNormal);
textbackground(menus.bgNormal);
}

/************************************************************
Put an item in menu.
<row> and <col> are left position.
************************************************************/
<str> is the string item.
<len> is the number of blanks to fill.
**************************************************************************

itemize(row,col,str,len)
int row, col, len;
char str[];
{
char temp[80];

gotoxy(col,row);
cputs(" ");
cputs(str);
memset(temp,' ',len--);
temp[len] = NULL;
cputs(temp);
}

/***************************************************************************/
Put display items on screen with border
***************************************************************************/

result_box()
{
int row=7, col=25, row2=21, col2=75;
int len, wid;

len = row2 - row - 1;
wid = col2 - col - 1;
window(col,row,col2,row2);
textbackground(menus.bgNormal);
crscr();
msgline(mess2);
box(row,col,len,wid);
gotoxy(col+15,row);
cputs(" MEASUREMENT ");
row += 2;
col += 2;
gotoxy(col,row);
cputs ("Current Time");
gotoxy(col,row+2);
cputs ("Front Wheel Angle (deg) ");
gotoxy(col,row+4);
cputs ("Travel Speed (km/h) ");
gotoxy(col,row+6);
cputs ("Front Wheel Position (m) ");
gotoxy(col,row+7);
cputs ("Front Wheel Error (cm) ");
gotoxy(col,row+9);
cputs ("Implement Position (m)"");
gotoxy(col,row+10);
cputs ("Implement Error (cm)" );
}

/**********************************************************
* Display measured and calculated results on screen with time
* Type ESC key to return to main menu
**********************************************************/

show_result()
{
    int i, row=9, col=55;

textcolor(menus.fgSelect);
gotoxy(col,row);
cprintf("%2d : %2d : %5.2f", hour,minute,second);
gotoxy(col,row+2);
cprintf("%5.1f",rad_to_deg(ft_ang));
gotoxy(col,row+4);
cprintf("%6.2f",3.6*speed);
gotoxy(col,row+6);
cprintf("%7.2f : %7.2f",fx,fy);
gotoxy(col,row+7);
cprintf("%7.1f",100.*ft_err);
gotoxy(col,row+9);
cprintf("%7.2f : %7.2f",ix,iy);
gotoxy(col,row+10);
cprintf("%7.1f",100.*ip_err);
}

/**********************************************************
* Write string message on the 25th line of screen
**********************************************************/

msgline(str)
char str[];
{
    char temp[80];

    textcolor(menus.fgOld);
    textbackground(menus.bgOld);
    window(1,25,80,25);
    clrscr();
gotoxy(1,1);
cputs(" ");
cputs(str);
    window(1,1,80,24);
unsigned cursor(value)
unsigned value;
{
union REGS inregs, outregs;
int ret;

inregs.h.ah = 3; /* Get old cursor */
inregs.h.bh = 0;
int86(0x10,&inregs,&outregs);
ret = outregs.x.cx;

inregs.h.ah = 1; /* Set new cursor */
inregs.x.cx = value;
int86(0x10,&inregs,&outregs);

return(ret);
}

int far keyrd()
{
asm mov ah,7 /* DOS Function number */
asm int 21h /* DOS Function call interrupt */
asm xor ah,a /* Clear high byte */
return(_AX);
}

int far keychk()
{  
asm mov dl,offh /* Move FF in DL */
asm mov ah,6 /* DOS Function number */
asm int 21h /* DOS Function call interrupt */
asm xor ah,ah /* Clear high byte */
asm jz keyloop
asm xor ax,ax

keyloop:
  return(_AX);
}

/*******************************************************************************/
DOS function call to obtain and return the hours (0-23),
minutes (0-59), seconds (0-59), and
hundredths of a second (0-99)
*******************************************************************************/

void far get_time()
{
  int isec, iths;
  
asm mov ah,2ch /* DOS Function number */
asm int 21h /* DOS Function call interrupt */
asm mov al,ch /* Move hours into AL */
asm xor ah,ah /* Clear high byte */
asm mov hour,ax
asm mov al,cl /* Move minutes into AL */
asm mov minute,ax
asm mov al,dh /* Move seconds into AL */
asm mov isec,ax
asm mov al,dl /* Move 1/100ths second into AL */
asm mov iths,ax
second = isec + 0.01*iths;
}

/*******************************************************************************/
Initialize the 8250 UART (COM1) for
4800 baud, no parity, 2 stop bits, and 8 data bits
*******************************************************************************/

void far cominitl()
{
  asm mov dx,3fbh /* Address of line control register */
asm mov al,80h /* Set 7th bit to initialize baud rate */
asm out dx,al /* To address baud rate divisor registers */
asm mov dx,3f8h /* Address of baud rate divisor (LSB) */
asm mov al,18h /* LSB value for 4800 baud */
asm out dx,al
asm mov dx,3f9h /* Address of baud rate divisor (MSB) */
asm mov al,0 ; /* MSB value for 4800 baud */
asm out dx,al
asm mov dx,3fbh ; /* Initialize line control register */
asm mov al,07h ; /* No parity, 2 stop bits, 8 data bits */
asm out dx,al
asm mov dx,3fch ; /* Initialize modem control register */
asm mov al,03h ; /* Set for data-terminal-ready */
asm out dx,al ; /* and request-to-send */
asm mov dx,3f9h ; /* Address of interrupt enable register */
asm mov al,0 ; /* Disable all four classes of interrupts */
asm out dx,al

/************************************************************
 Front wheel position measurement - COM-1 port used
 Change to float Number from character reading
 ************************************************************/

void far comrdl() {
  char ch=0, buff[20]; /* Temporary buffer */
  int i=0;
  float temp=0.;

  while (ch !- 'X') { /* Wait Until read 'X' */
    comrecvl();
    ch = _AL;
  }
  for(i=0;i<=12;i++) { /* Read 13 characters */
    comrecvl();
    buff[i] = _AL;
  }

  temp = 1000.*(buff[1]-48) + 100.*(buff[2]-48) + 10.*(buff[3]-48)
  + (buff[4]-48) + 0.1*(buff[5]-48);
  fax = (buff[0] == '-') ? -0.3048*temp : 0.3048*temp;
    temp = 1000.*(buff[8]-48) + 100.*(buff[9]-48) + 10.*(buff[10]-48)
    + (buff[11]-48) + 0.1*(buff[12]-48);
  else
    temp = 1000.*(buff[7]-48) + 100.*(buff[8]-48) + 10.*(buff[9]-48)
    + (buff[10]-48) + 0.1*(buff[11]-48);
  fax = (buff[7] == '-') ? -0.3048*temp : 0.3048*temp;

  /* Compensate front wheel position measurements */
  fax += fx_adj;
  fay += fy_adj;
}

/* Miscellaneous */
Get character input from COM1 port
Low 7-bits are valid

```c
void comrecv1()
{
    wait1:
        asm mov dx,3fdh ; /* Address of line status register */
        asm in al,dx ; /* Line status in AL */
        asm test al,1
        asm jz wait1
        asm mov dx,3f8h ; /* Address of data receiver register */
        asm in al,dx ; /* Received data in AL */
        asm and al,7fh ; /* 7 bits of data - High bit is not valid */
}
```

Initialize the 8250 UART (COM2) for
4800 baud, no parity, 2 stop bits, and 8 data bits

```c
void far cominit2()
{
    asm mov dx,2fbh ; /* Address of line control register */
    asm mov al,80h ; /* Set 7th bit to initialize baud rate */
    asm out dx,al
    asm mov dx,2f8h ; /* Address of baud rate divisor registers */
    asm mov al,18h ; /* LSB value for 4800 baud */
    asm out dx,al
    asm mov dx,2f9h ; /* Address of baud rate divisor (MSB) */
    asm mov al,0 ; /* MSB value for 4800 baud */
    asm out dx,al
    asm mov dx,2fbh ; /* Initialize line control register */
    asm mov al,07h ; /* No parity, 2 stop bits, 8 data bits */
    asm out dx,al
    asm mov dx,2fch ; /* Initialize modem control register */
    asm mov al,03h ; /* Set for data-terminal-ready */
    asm out dx,al
    asm mov dx,2f9h ; /* Address of interrupt enable register */
    asm mov al,0 ; /* Disable all four classes of interrupts */
    asm out dx,al
}
```

Implement position measurement - COM-2 port used
Change to float Number from character reading

```c
void far comrd2()
```
{  
  char ch=0, buff[20]; /* Tempary buffer */  
  int i=0;  
  float temp=0.;  

  while (ch != 'X') { /* Wait Until read 'X' */  
    comrecv2();  
    ch = _AL;  
  }  
  for(i=0;i<12;i++) { /* Read 13 characters */  
    comrecv2();  
    buff[i] = _AL;  
  }  

  temp = 1000.*(buff[1]-48) + 100.*(buff[2]-48) + 10.*(buff[3]-48)  
       + (buff[4]-48) + 0.1*(buff[5]-48);  
  ix = (buff[0] == '-') ? -0.3048*temp : 0.3048*temp;  

    temp = 1000.*(buff[8]-48) + 100.*(buff[9]-48) + 10.*(buff[10]-48)  
       + (buff[11]-48) + 0.1*(buff[12]-48);  
  else  
    temp = 1000.*(buff[7]-48) + 100.*(buff[8]-48) + 10.*(buff[9]-48)  
       + (buff[10]-48) + 0.1*(buff[11]-48);  
  iy = (buff[7] == '-') ? -0.3048*temp : 0.3048*temp;  

  /* Compensate implement position measurements and  
     change to general x-y coordinate */  
  ix = ix_adj + 91.44 - ix;  
  iy = iy_adj + 101.5 - iy;  
}

/**********************************************************
Get character input from COM2 port
Low 7-bits are valid
**********************************************************/

void comrecv2() {  
  wait2:  
    asm mov dx,2fdh ; /* Address of line status register */  
    asm in al,dx ; /* Line status in AL */  
    asm test al,1  
    asm jz wait2  
    asm mov dx,2f8h ; /* Address of data receiver register */  
    asm in al,dx ; /* Received data in AL */  
    asm and al,7fh ; /* 7 bits of data - High bit is not valid */  
  }

/* This is an assembly-language subroutine called from TURBO-C to perform Analog-Digital conversion and Digital Counter for the PARALLEL interface version of PC-ACQUISITOR from DIANACHART Inc.

The Microsoft MACRO Assembler is required.

Written by DIANACHART Inc
Edited by Terry Walker and Chang Choi
*/

//******************************/
// Exit procedure
// FAR return after resetting the addresses to point to the printer
//******************************/

void parexitO
{
    asm mov
    asm mov
    asm mov
    al,PRINTER
    dx.parstat
    dl,al

    asm mov
    asm mov
   asm mov
    ax, 04
    asm push
    asm mov
    bx.datptr.par
    zloop:
    asm mov
    ax,[bx]
    asm pop
    dx
    asm dec
    dx
    asm jns zcont
    parexit();
    return;
    zcont:
    asm push
    dx
    asm mov
    di,dx
    asm mov
    al,[gconvert+di]
    asm mov
    cl,GAINSTB
datastb();
    wordad();
dataout();
    asm mov
    bx,datptr.ampzero
    asm mov
    al,0ffh
    asm mov
    ah,ALATCH
dataout();
    asm mov
    ax,0ffh
    asm mov
    bsave,al
    asm mov
    ah,BLATCH
dataout();
    asm mov
    al,0ffh
    asm mov
    ah,CLATCH
dataout();
    parexit();

}
DIGOUT sends the contents of BINOUT, integer variable to the Binary Outputs. Data taken from BINOUT Returns no data Calls PAREXIT DATAOUT

void far digout ()
{
    asm mov bx,datptr.binout
    asm mov dx,[bx]
    asm not dx
    asm mov cx,08h
reverse:
    asm rcr dx,1
    asm rcl al,1
    asm loop reverse
    asm mov ah,ALATCH
dataout();
    asm mov dx,[bx]
    asm not dx
    asm mov dl,dh
    asm rcr dh,1
    asm rcr dl,1
    asm rcr al,1
    asm and al,0cOh
    asm mov dl,low bsave
    asm and dl,03fh
    asm or al,dl
    asm mov low bsave,al
    asm mov ah,BLATCH
dataout();
    asm inc bx
    asm inc bx
    asm mov di,0
nextdac:
    asm mov cx,[bx+di]
    asm mov al,cl
boardout:
    asm not al
    asm mov ah,CLATCH
dataout();
    asm mov ax,di
    asm or al,0e0h
    asm and al,low bsave
    asm mov ah,BLATCH
dataout();
    asm mov ah,BLATCH
dataout();
    asm inc di
    asm mov al,chl
    asm mov dx,di
    asm and dx,01h
    asm jnz boardout
    asm mov dx,di
    asm cmp dx,8
    asm jl nextdac
    parexit();
}

DIGIN Reads Binary inputs & returns result in BININ
RETURNS result in BININ
Calls PAREXIT DIGNEAR

void far digin ()
{
    dignear();
    asm mov bx,datptr.binin
    asm mov [bx],cx
    parexit();
}

COUNTER Reads Binary Inputs, then resets & enables counter. Latest count is returned in BININ as integer
Calls PAREXIT DIGNEAR DATAOUT.

void far counter ()
{
    cnt1:
    dignear();
asm mov bx,cx
asm add bx,07
asm cmp bx,cx
asm jl cntl
asm mov al,low bsave
asm or al,020h
asm mov ah,BLATCH
dataout();
asm and low bsave,0dfh
asm mov al,low bsave
dataout();
asm mov bx,datptr.binin
asm mov [bx],cx
parexit();
)

void far cntoff()
{
  asm mov al,low bsave
  asm or al,020h
  asm mov low bsave,al
  asm mov ah,BLATCH
dataout();
parexit();
}

#include<

void far fulladf()
{
  fullad();
parexit();
}
Perform full A/D conversion on
channel in CURCHAN
RESULT returned in CDAT array
for appropriate channel
Calls PAREXIT ONEAD

_asm mov bx,datptr.curchan
_asm mov cx,[bx]
onead();
parexit();

This routine performs a 16-bit
A/D returning the result in
RESULT
The channel & gain should have
been previously set up.
Calls PAREXIT WORDAD

_wordad();
_asm mov bx,datptr.result
_asm mov [bx],cx
_parexit();

/** Not implemented at present */

void far byteadf()
{
}

void far sparel()
{
}

_Send the 8-bit data in BDATA
to the latch on the isolated
A/D section addressed by
ANAADDR
GAIN  Sets amplifier gain
SELECT  Sets Input multiplexers
DACHIGH 242 D/A Most
  significant Byte
DACLOW 241 D/A Least
  significant Byte
Calls PAREXIT DATASTB

_asm mov bx,datptr.bdata
_asm mov al,[bx]
_asm mov bx,datptr.anaaddr
_asm mov cl,[bx]
datastb();
parexit();

_Send the 8-bit data in BDATA to
the latch pointed to by ADDR
ALATCH=2 (Binary Output)
BLATCH=0 (Binary Output &
  Analog section strobes)
CLATCH=6 (8-bit data bus to
  Analog section)
PRINTER=4
Calls PAREXIT DATAOUT

_asm mov bx,datptr.bdata
_asm mov al,[bx]
_asm mov bx,datptr.addr
_asm mov ah,[bx]
_asm cmp ah,0
_asm jnz fdol
_asm mov bsav,ah

fdol:
dataout();
parexit();

_asm mov bx,datptr.bdata
FULL A/D All active channels up to MAXCHAN (Bit 3 set in CHAN array)
Returns result in CDAT array
Calls ONEAD

void fullad()
{
    asm mov bx,datptr.maxchan
    asm mov cx,[bx]
    asm inc cx
    asm push cx
    adloop:
    asm pop cx
    asm dec cx
    asm jnz adcont
    asm push cx
    return;
    adcont:
    asm mov bx,datptr.chan
    asm add bx,cx
    asm add bx,cx
    asm mov ax,[bx]
    asm and al,08h
    asm jz adloop
    onead();
    asm jmp adloop
}

Single-channel A/D with gain adjustment
Needs channel to be measured
Returns result in CDAT array
Calls DATASTB WORDAD

void onead()
{
    asm mov bx,datptr.seldat
    asm add bx,cx
    asm mov ax,[bx]
    asm mov cl,SELECT
    datastb();
    asm push cx

asm jmp donead
upgain:
asm pop dx
asm dec dx
asm jz donead
asm pop ax
asm push ax
asm push dx
asm mov bx, datptr.gain
asm add bx, ax
asm add bx, ax
asm inc byte ptr [bx]
asm cmp [bx], byte ptr 03
asm jg limit
asm mov cx, ax
asm jmp retry
limit:
asm mov [bx], byte ptr 03
asm pop ax
asm jmp donead

void bytead()
{
    adloop:
        asm mov al, bh
datastb();
        asm mov al, bl
        asm or bl, bl
        asm jz bytexit
        asm shr bl, 1
        asm or bh, bl
        asm xor al, bh
        asm mov di, ax
        asm mov ax, compdelay
        asm mov bx, ax

    bloop:
        asm dec al
        asm jnz bloop
        asm mov al, ALATCH
        asm out dx, al
        asm mov dx, parin
        asm in al, dx
        asm and al, 80h
        asm jnz adloop
        asm mov dx, di
        asm mov bh, dl
        asm jmp adloop
bytexit:
    return;
}

void wordad()
{
    adloop:
        asm mov cl, DACLOW
datastb();
        asm mov al, 00h
datastb();
        asm mov cl, DACHIGH
        asm mov bx, 8080h
        bytead();
        asm xor bh, 080h
        asm mov ch, bh
        asm mov cl, DACLOW
        asm mov bx, 8080h
        bytead();
        asm mov cl, bh
}

void datastb()
{
    asm mov ah, CLATCH

/****************************
8-Bit A/D Subroutine.
Gain/Channel already selected
Expects CL to contain DACHI or
DACLLO mask as needed
BH and BL to contain the test
masks (8080h-8 Bit)
Returns result in BH
Call DATASTB
****************************/

/****************************
16-BIT A/D on currently
selected gain & channel
RESULT returned in CX as signed
16-bit number
Calls DATASTB WORDAD
****************************/

/****************************
Sends data in AL to floating
latch strobed by mask in CL
Returns Nothing
Call DATAOUT
****************************/

/****************************
Sends data in AL to floating
latch strobed by mask in CL
Returns Nothing
Call DATAOUT
****************************/

dataout();
asm mov al,low bsave
asm or al,0fh
asm mov ah,BLATCH
asm mov di,ax
asm and al,cl
dataout();
asm mov ax,di
dataout();
asm mov ah,CLATCH
asm mov al,0ffh
dataout();
}

/********************************
Sends data in AL to latch
addressed by AH. Strobes low
control bit
Returns Nothing
********************************/

void dataout()
{
asm mov dx,parout
asm out dx,al
asm mov al,ah
asm mov dx,parstat
asm out dx,al
asm inc al
asm out dx,al
asm dec al
asm out dx,al
}