# A VEHICLE-BASED LASER SYSTEM FOR GENERATING HIGH-RESOLUTION DIGITAL ELEVATION MODELS

by

PENG LI

B.A., China Agricultural University, China, 2002
M.S., China Agricultural University, China, 2005

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Biological and Agricultural Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

# Abstract

Soil surface roughness is a major factor influencing soil erosion by wind and water. Studying surface roughness requires accurate Digital Elevation Model (DEM) data. A vehicle-based laser measurement system was developed to generate high-resolution DEM data. The system consisted of five units: a laser line scanner to measure the surface elevation, a gyroscope sensor to monitor the attitude of the vehicle, a real-time kinematic GPS to provide the geographic positioning, a frame-rail mechanism to support the sensors, and a data-acquisition and control unit. A user interface program was developed to control the laser system and to collect the sensors data through a field laptop.

Laboratory experiments were conducted to evaluate the performance of the laser sensor on different type of targets. The results indicated that the laser measurement on a white paper had the least variability than that on other targets. The laser distance measurement was calibrated using the data acquired on the white paper.

Static accuracy tests of the gyroscope sensor on a platform that allowed two-axis rotations showed that angle measurement errors observed in combined pitch/roll rotations were larger than those in single rotations. Within ±30° of single rotations, the measurement errors for pitch and roll angles were within 0.8° and 0.4°, respectively. A model to study the effect of attitude measurement error on elevation measurement was also developed.

DEM models were created by interpolating the raw laser data using a two-dimensional, three-nearest neighbor, distance-weighted algorithm. The DEM models can be used to identify shapes of different objects.

The accuracy of the laser system in elevation measurement was evaluated by comparing the DEM data generated by the laser system for an unknown surface with that generated by a more accurate laser system for the same surface. Within four replications,

the highest correlation coefficient between the measured and reference DEMs was 0.9371. The correlation coefficients among the four replications were greater than 0.948. After a median threshold filter and a median filter were applied to the raw laser data before and after the interpolation, respectively, the correlation coefficient between the measured and reference DEMs was improved to 0.954. Correlation coefficients of greater than 0.988 were achieved among the four replications. Grayscale images, which were created from the intensity data provided by the laser scanner, showed the potential to identify crop residues on soil surfaces.

Results of an ambient light test indicated that neither sunlight nor fluorescent light affected the elevation measurement of the laser system. A rail vibration test showed that the linear rail slightly titled towards the laser scanner, which caused small variations in the pitch angle.

A preliminary test on a bare soil surface was conducted to evaluate the capability of the laser system in measuring the DEM of geo-referenced surfaces. A cross-validation algorithm was developed to remove outliers. The results indicated that the system was capable of providing geo-referenced DEM data.

# A VEHICLE-BASED LASER SYSTEM FOR GENERATING HIGH-RESOLUTION DIGITAL ELEVATION MODELS

by

PENG LI

B.S., China Agricultural University, China, 2002
M.S., China Agricultural University, China, 2005

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Biological and Agricultural Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Dr. Naiqian Zhang

# Abstract

Soil surface roughness is a major factor influencing soil erosion by wind and water. Studying surface roughness requires accurate Digital Elevation Model (DEM) data. A vehicle-based laser measurement system was developed to generate high-resolution DEM data. The system consisted of five units: a laser line scanner to measure the surface elevation, a gyroscope sensor to monitor the attitude of the vehicle, a real-time kinematic GPS to provide the geographic positioning, a frame-rail mechanism to support the sensors, and a data-acquisition and control unit. A user interface program was developed to control the laser system and to collect the sensors data through a field laptop.

Laboratory experiments were conducted to evaluate the performance of the laser sensor on different type of targets. The results indicated that the laser measurement on a white paper had the least variability than that on other targets. The laser distance measurement was calibrated using the data acquired on the white paper.

Static accuracy tests of the gyroscope sensor on a platform that allowed two-axis rotations showed that angle measurement errors observed in combined pitch/roll rotations were larger than those in single rotations. Within ±30° of single rotations, the measurement errors for pitch and roll angles were within 0.8° and 0.4°, respectively. A model to study the effect of attitude measurement error on elevation measurement was also developed.

DEM models were created by interpolating the raw laser data using a two-dimensional, three-nearest neighbor, distance-weighted algorithm. The DEM models can be used to identify shapes of different objects.

The accuracy of the laser system in elevation measurement was evaluated by comparing the DEM data generated by the laser system for an unknown surface with that generated by a more accurate laser system for the same surface. Within four replications,

the highest correlation coefficient between the measured and reference DEMs was 0.9371. The correlation coefficients among the four replications were greater than 0.948. After a median threshold filter and a median filter were applied to the raw laser data before and after the interpolation, respectively, the correlation coefficient between the measured and reference DEMs was improved to 0.954. Correlation coefficients of greater than 0.988 were achieved among the four replications. Grayscale images, which were created from the intensity data provided by the laser scanner, showed the potential to identify crop residues on soil surfaces.

Results of an ambient light test indicated that neither sunlight nor fluorescent light affected the elevation measurement of the laser system. A rail vibration test showed that the linear rail slightly titled towards the laser scanner, which caused small variations in the pitch angle.

A preliminary test on a bare soil surface was conducted to evaluate the capability of the laser system in measuring the DEM of geo-referenced surfaces. A cross-validation algorithm was developed to remove outliers. The results indicated that the system was capable of providing geo-referenced DEM data.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgements

I would like to convey my heartfelt gratitude to my major professor, Dr. Naiqian Zhang, for his academic guidance, continuous encouragement, and financial support through my Ph.D. program. He is a mentor and a friend to me. His infectious enthusiasm for teaching and research, his open-minded attitude towards novelties, his rigorous scholarship and scientific thought will be guidelines for my future professional life.

I would like to express my sincere gratitude to my co-major professor, Dr. Larry E. Wagner, for his invaluable suggestions, continuous encouragement, and financial support through my Ph.D. research.

My sincere gratitude is extended to my supervisory committee members, Dr. Paul Nelson, and Dr. Dale Schinstock, and to my outside chair, Dr. Sanjoy Das. Thanks to Dr. Nelson for his patience and kindness, for spending his precious time on answering my statistics questions, for giving me many suggestions on outlier removal and factorial treatment designs. Thanks to Dr. Schinstock for his valuable suggestions on dynamic experiments design. Thanks to Dr. Sanjoy for conducting my final examination.

I would like to thank many experts who helped me overcome the obstacles in this research. Thanks to Dr. Fred Fox for his many significant suggestions on solving many problems at each stage of system development, for his continuous encouragement. Thanks to Mr. Darrell Oard for his assistance in constructing the aluminum frame and guiding me how to use the mechanical machines in the workshop. Thanks to Mr. Hubert

Lagae for his assistance in building artificial surface and mechanical structure, and for his support on the operation of the RTK GPS. I thank many technical support calls to Schmitt Measurement Systems, Inc. and Topcon Positioning System, Inc., for answering my questions on the laser line scanner and Topcon RTK GPS.

Profound thanks and appreciation also go to Dr. Gary A. Clark, former department head, Dr. Joe P. Harner, current department head, Dr. James M. Steichen, Dr. Donghai Wang, Dr. Wenqiao Yuan, and all other faculty members and staff in the department. Many thanks to Mrs. Barb Moore and Mr. Randy Erickson for helping me in all issues related to my study and research in the department.

I also would like to extend my gratitude to all the fellow students in the Instrumentation and Control Lab: Dr. Yali Zhang, Mr. Wei Han, Mr. Ning Tang, Ms. Ling Xue, Dr. Jizhong Deng, Dr. Suxia Wang, Ms. Sarah Shultz, Ms. Huiquan Zhang, Mr. Xu Wang, and Mr. Joseph Dvorak for their help on my research and profound friendship among us. My gratitude also goes to all my friends in the BAE department and others at KSU.

Finally, my deep appreciation is given to my family for their patience, continuous support and encouragement, which have made me finish my study smoothly. Special thanks are given to my wife, Ning Tang, for her academic suggestions and help on my research, for her support and encouragement when I met difficulties, for her patience and unconditional love in life, which have helped me complete my studies.

# CHAPTER 1 - Introduction

Soil surface roughness, or microrelief, which describes the micro variation in surface elevation, is a crucial factor that influences soil erosion by water and wind (Moreno et al., 2008a). Soil erosion is related to surface processes, such as infiltration, runoff, gas exchange, sediment transport, and deposition. Surface roughness is a dynamic soil property that controls these physical processes (Darboux and Huang, 2001). Therefore, studying these processes and their spatial variations requires accurate descriptions of soil surface roughness.

Soil roughness has been classified to four categories : 1) roughness due to individual particles (aggregates of 0-2 mm in diameter); 2) random roughness (aggregates of 100 mm in diameter); 3) oriented (ridge) roughness due to tillage implements (aggregates of 100-300 mm in diameter); 4) higher order roughness due to field topography (Romkens and Wang, 1986). Random roughness and oriented roughness are of the most interest to wind erosion studies, because they change rapidly due to weathering and tillage (Wagner and Hagen, 1991). Tillage ridges containing large aggregates prevent soil loss from surface more effectively than unridged surface. Thus, soil tillage ridges are often used to aid in control of wind erosion. Aerodynamic surface roughness, which is converted from random and oriented roughness based upon wind direction, is the ultimate goal of all the roughness measurement (Hagen and Armbrust, 1992).

Wind erosion is the detachment, transportation, and deposition of soil particles by the wind (MCI, 2003) The most obvious result of wind erosion is loss of fine particles and organic materials in soil, which reduces a soil’s quality and its ability to produce crops. Small soil particles may suspend in the atmosphere which pollutes the air. Agricultural engineers seek ways to control and prevent wind erosion. Hagen (1996) found that managing surface residues is a key factor to successfully control wind erosion. Crop residues standing above the soil surface are much more effective in preventing wind erosion than the same quantity of residues lying flat on the soil surface. Studying and managing surface residues is a very challenging task, because it requires time and effort to count and measure standing residues (Fox and Wagner, 2001). Thus, a method to accurately assess the standing residue on a soil surface is needed.

Surface roughness is also related to improvement of water quality in the riparian zone. A riparian buffer is a vegetated area next to a water resource that helps shade and partially protect a water source from the impact of adjacent land uses (WFI, 2008). It includes stream banks, floodplain, and wetlands, as well as sub-irrigated sites forming a transitional zone between upland and aquatic habitat (Lowrance et al., 1997). Riparian buffer not only provides valuable habitat for wildlife, it also helps improve water quality by intercepting sediment, nutrients, pesticides, and other materials and pollutants in surface runoff or subsurface flow before they reach the water resources. Riparian buffers are also important in reducing erosion by maintaining stable stream banks. To maintain and design the riparian buffer, a study of the physical hydrology processes in the buffer is needed. This study is highly empirical because of the difficulty in tracking the water movement through or over the buffer. Thus, a method to accurately describe the physical processes of a riparian buffer is necessary.

The overall objective of this research was to develop a real-time, field portable measurement system that is capable of measuring geo-referenced surface elevations with a sub-inch accuracy.

The specific objectives were:

1) to develop a system that integrates a precision laser scanner, a gyroscope, and a RTK GPS with a portable computer;

2) to construct a flexible mounting frame with a linear rail on a vehicle that supports the system components and controls the movement of the laser;

3) to develop a computer program to control the system and to record the measurement data;

4) to develop data processing algorithms to create three-dimensional digital elevation model (DEM);

5) to conduct laboratory tests to evaluate the performances of individual components of the system;

6) to conduct laboratory tests to evaluate the accuracy and repeatability of the system in measuring elevation under different environments;

7) to conduct preliminary field tests to examine the capabilities of the system in measuring micro-relief of soil surfaces.

# CHAPTER 2 - Background and Literature Review

## 2.1 Study of Soil Surface Roughness

Soil erosion by wind and water are natural, complex dynamic processes. Soil surface roughness is a scientific index to help understand these physical processes. Many researchers have studied the mechanisms of these processes. To assist these studies, various techniques to measure surface roughness were developed over the past years.

Surface roughness measuring techniques can be classified by spatial dimensions and sensor types (Jester and Klik, 2005). Two-dimensional measurements used simple tools, such as a pin meter and roll chain, to acquire limited surface characterization. Three-dimensional measurements gave a more realistic surface representation and allowed more analysis of surface parameters. Concerning the sensor type, measurement techniques can also be divided into contact and noncontact categories. The contact methods disturbed soil surface with pins or chain set for height measurement, whereas noncontact methods measured the distance from soil surface to a reference plane without touching the soil surface (Robichaud and Molnau, 1990). The noncontact methods gave an opportunity to monitor changes in soil properties over time. These methods fell into six categories: infrared, ultrasonic, photogrammetry, laser, LiDAR, and 3D laser. Each method had its advantages and disadvantages. For field studies and applications, the system used to measure soil surface roughness should be portable, accurate, and less time-consuming.

## 2.2 Techniques for Measuring Soil Surface Roughness

### *2.2.1 Pin Meter*

The pin method was the dominant technique for measuring soil surface roughness before 1990s. A pin meter, or "profile meter", consisted of a single pin or a row of evenly spaced pins (Figure 2.1). When measuring the elevation, the pins were lowered manually or automatically onto the soil surface until contacts were made. Surface elevations were then registered manually, electronically, or photographically from the relative heights of the pins, and later digitized (Podmore and Huggins, 1981; Radke et al., 1981; Wagner and Yu, 1991).

**Figure 2.1 Photographic Pin Meter (from ARS Photo, 2007)**

Moreno et al. (2008b) developed a method based on a pin-meter prototype and compared surface roughness of different types of soils (sandy clay loam and sandy loam) tilled with different tillage tools (roller, chisel, and tiller) in central Spain. A digital camera mounted on a tripod on the pin frame was used to digitalize the micro-topography identified by each pin position.

The pin meter was a simple, low cost method to obtain data in the field. However, this method required a long measurement time. Furthermore, this method gave limited resolution, which was determined by the physical pin spacing. Being a contact type measurement, this method also disturbed and damaged the original soil surface profile, especially for soft surfaces.

### *2.2.2 Chain Method*

Roller chain was another contact method for measuring soil surface roughness. The chain method used chain-like devices placed across a surface. Soil roughness was mechanically measured by calculating the horizontal length reduction of the chain (Saleh, 1993; Merrill et al., 2001). This method consisted of a roller chain, or a number of chain sets, and caliper rulers with a telescopic pointer (Figure 2.2). The chain method was an easy, fast, and inexpensive technique. However, it also deformed the original soil surface, especially on loose or wet soils. Furthermore, the chain technique was subject to scale indeterminacy. This was because a chain could yield the same roughness result for a surface with a large number of small rough elements and a surface with a small number of large rough elements (Skidmore, 1997). No digital elevation model can be created from the chain method.

**Figure 2.2 Chain Sets (from Merrill et al., 2001)**

### *2.2.3 Infrared Sensor*

An early non-contact technique to measure soil roughness was infrared radiation. Romkens and Wang (1986) developed an infrared profiler meter. The meter included an optical probe which detected the soil surface at a known tracking height in predetermined transects. The horizontal and vertical movement of the probe was accomplished through ball bearing screws driven by DC motors. The position of the probe was recorded by encoders. The profiler frame was equipped with utility vehicles to allow measurements of different transects. The covered area was 1 m by 1.15 m. A vertical accuracy of 3 mm was reported. However, due to the difference in reflectivities of different surfaces, the profiler meter was only used on surfaces with a uniform albedo.

### *2.2.4 Ultrasonic Profiler*

Ultrasonic technology was widely used in distance measurement. The use of ultrasonic waves in distance measurement was also called SONAR (sound navigation and ranging). A non-contact ultrasonic profiler was developed by Robichaud and Molnau

(1990) to measure soil roughness. This profiler used an ultrasonic transducer to emit an ultrasonic signal to the soil surface and then received the reflected signal. The time between transmission and reception of the signal was proportional to the distance between the sensor and the soil surface. A 1.5 m by 1.5 m aluminum frame was used as the base to provide a reference plane. The movement of the ultrasonic sensor on the frame was controlled by a stepper motor. A vertical accuracy of 3 mm and a horizontal resolution of 30 mm were achieved by this profiler.

### *2.2.5 Photogrammetry*

Photogrammetry was the first remote sensing technology developed to determine the geometric properties of objects from photographic images (WFI, 2009). A more sophisticated technique, called stereophotogrammetry, could estimate the three-dimensional coordinates of points on an object. These coordinates were measured in two or more photographic images taken from different positions and angles with common reference points in each image (Welch et al., 1984; Warner, 1995; Taconet and Ciarletti, 2007). A measurement accuracy of 1 mm has been reported.

The photogrammetric technique was capable of reducing the data acquisition time for images covering large areas. However, interpretations of the photogrammetric data were complicated (Jester and Klik, 2005).

### *2.2.6 Laser*

Laser was an acronym for Light Amplification by Stimulated Emission of Radiation. The light it referred to was an electromagnetic radiation of a certain wavelength, including visible, infrared, and ultraviolet lights, as well as electromagnetic waves in other wave bands. A laser light was usually a narrow, low-divergence beam in a narrow wavelength band.

Due to its ability to obtain accurate measurement, laser has been used in a wide variety of scientific, military, medical, and commercial applications since its invention in

1958. One of the applications was range measurement. Most laser rangefinders projected a visible or infrared laser beam onto a target or a surface to which the distance was to be measured. A light-detector then received the beam reflected from the target or surface. The distance between the target or surface and the laser sensor was determined based on the triangulation or the time-of-flight principles. “Triangulation” sensors calculated the distance by determining where the reflected beam fell on the detector. “Time-of-flight” sensors derived the distance from the time the light took to travel from the sensor to the target or surface and return.

Laser systems for automated, non-contacted measurements of surface elevation commonly included three major components: an optical transducer with a receiver to detect the distance, a computer-controlled, motor-driven, two-dimensional traversing frame, and a set of interface circuitry and a PC to control the motion of the camera-laser carriage and to register the elevation data (Bertuzzi et al., 1990; Huang and Bradford, 1990; Flanagan et al., 1995; Wilson et al., 2001; Darboux and Huang, 2003).

With the fast development of laser technology, many commercial laser scanners became available in the market. Arvidsson and Bolenius (2006) reported a commercial laser sensor, SICK DME3000-211, with a three-leg, two-direction mounting frame, in studying the effects of soil water content during primary tillage in Sweden. Lee and Ehsani (2008) compared characteristics and performances of two laser scanners, SICK LMS200 and Hokuyo URG-04LX.

The laser method was not very convenient for frequent field experiments because it required transportation of a heavy traversing frame. Moreover, hidden objects in the reflection path could cause missing data for the “triangulation” lasers.

### *2.2.7 LiDAR*

LiDAR (Light Detection and Ranging), also known as Airborne Laser Swath Mapping or ALSM, was one of the most recent remote sensing technologies used to detect range and acquire other information of a distant target. The range to an object was determined by calculating the time taken between transmission of a laser pulse and detection of the reflected signal. LiDAR had a wide range of applications in mapping and surveying.

Typically, LiDAR was the combination of three different data collection equipment - a laser scanner mounted on an aircraft or a helicopter to determine the distance to the object, a Global Positioning System (GPS) to provide the sensor position, and an Inertial Navigation System (INS) to acquire the orientation characteristics (Brovelli et al., 2004; Habib et al., 2005; Hollaus et al., 2005; Reutebuch et al., 2005; Webster et al., 2006; Pfeifer and Briese, 2007; Liu, 2008). Other important extensions of the LiDAR systems were an integration of a high-resolution digital camera or a digital video camera (Ackermann, 1999; Ahlberg et al., 2004). Commercial LiDAR systems could achieve a root mean square error (RMSE) of 15 cm vertically and a sub-meter RMSE horizontally (CARMS, 2006). However, a number of researchers have examined the vertical accuracy of LiDAR data with varying results from 5cm to 1m (Woolard and Colby, 2002).

A relatively new technique of LiDAR is the ground-based LiDAR system, which was mainly used for forestry applications. Accurate canopy structure with individual tree parameters has been successfully estimated using ground-based LiDAR systems (Watt and Donoghue, 2005; Henning and Radtke, 2006). It has also been used to study the effect of system geometric set-up on the accuracy of trees structure measurement (Van der Zande et al., 2006). Loudermilk (2009) introduced another promising approach of ground-based LiDAR to efficiently capture fine-scale characteristics of shrubs, specifically heights and volumes.

The LiDAR system was one of the fastest and most effective means of collecting topographic data and it has become the primary choice for large-area forestry applications. However, effectively processing raw LIDAR data to avoid errors and distortion was a big challenge. Development in this area was still in progress (Liu, 2008). Due to the limitation in accuracy, LiDAR has not been used to characterize soil surfaces in small scales.

### *2.2.8 3D Laser Scanner*

A 3D scanner was an instrument that analyzed a physical object or environment by capturing dense point cloud data on its surface. The collected data could be further processed to build virtual 3D models. A 3D scanner was similar to a camera. However, it collected distance information, rather than color information, as a camera would do. By combining the distance information with the orientation of the scanner, three-dimensional positions of the points could be created in local coordinate of the scanner. Many different technologies could be used to construct 3D scanning devices, although laser scanner was the most commonly used one.

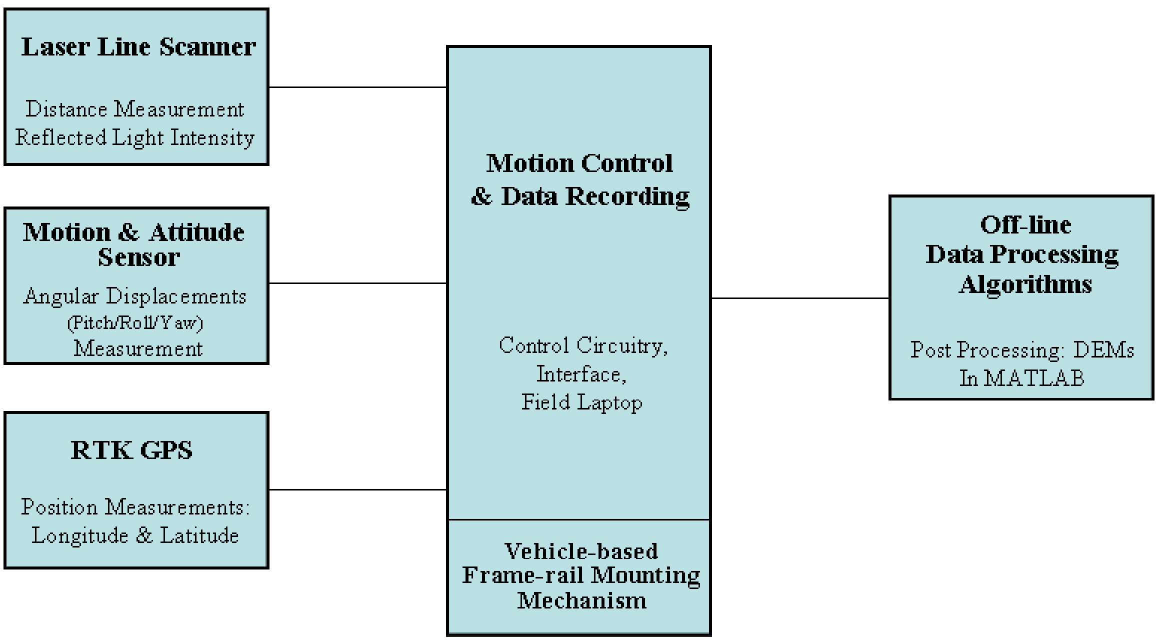
In a 3D laser scanner, a laser probe projected a laser beam to a surface while cameras continuously triangulated the changing distance and the shape of the laser line as it swept along, and digitized the object in three dimensions (LDI, 2008). The 3D laser scanning technique was an effective method to create a complete, three-dimensional documentation for the spatial geometries of an object. The recorded data, point clouds, could not be used directly. They had to be processed using special software to convert into detailed, useable 3D models (ArcTron, 2005; STC, 2008). Milan (2007) reported an application of a 3D laser scanner (LMS-Z210) in studying erosion and deposition volumes in a proglacial river. A vertical precision of 2 cm and an average spacing of 2 cm were achieved. Some of the commercial 3D laser scanners could provide an accuracy of 0.5 mm and a vertical resolution of 0.1 mm (CFI, 2008).

The 3D laser scanner was a new technology to build 3D models fast, accurately, but expensively. This technology has become widely used in collecting high-density 3D geo-spatial data for buildings, factories, landscapes, and other large architectures. Due to its high accuracy and resolution, the 3D laser scanner would have a great potential in measuring micro-topography of soil surfaces in the future.

# CHAPTER 3 - Design and Construction of the Laser System

A vehicle-based laser measurement system was constructed. The system consisted of five major components: 1) a distance-measuring unit, 2) a frame-rail unit, 3) a frame angular-position measuring unit, 4) a geo-referencing unit, and 5) a data-acquisition and control unit. Algorithms for data processing were developed to create the DEMs in MATLAB (MATLAB, 2006). A block diagram of the system components is shown in Figure 3.1. Detail descriptions of hardware and software are reported in the following sections.

**Figure 3.1 System Components**



## 3.1 Hardware Design

### *3.1.1 Distance-measuring Unit: Laser Line Scanner*

The distance-measuring unit was used to measure the surface elevation. An AccuRange Line Scanner, AccuRange 4000-LIR (Acuity Research Inc., 2000), was used (Figure 3.2). This scanner consisted of an elliptical, rotating mirror driven by a DC motor with an optical encoder, and mounting hardware for use with an Acuity's 4000 laser rangefinder. The rotating mirror swept a laser beam in 360° rotations. The rangefinder measured distances of up to 15.24 m with a 2.54 mm accuracy.

**Figure 3.2 AccuRange Laser Line Scanner**


The measurement principle of the laser line scanner is demonstrated in Figure 3.3. A collimated, infrared laser beam of 780nm wavelength was emitted by the laser diode. The beam was deflected 90 degree toward the target by the rotating mirror. The target then reflected the laser beam back to the sensor through the same mirror. By knowing the traveling time of the laser beam, the distance between the sensor and the target can be determined. The elliptical mirror was situated at a 45 degree angle from the incident laser beam and was rotated continuously by a 12V DC motor. The mirror was engineered to the highest optical standards with 96% reflectance. An optical encoder with 4096 counts per revolution was attached on the motor to record the mirror's angular position. Additional information on the laser line scanner is listed in Table 3.1.

**Figure 3.3 Measurement Principle of the Laser Line Scanner**

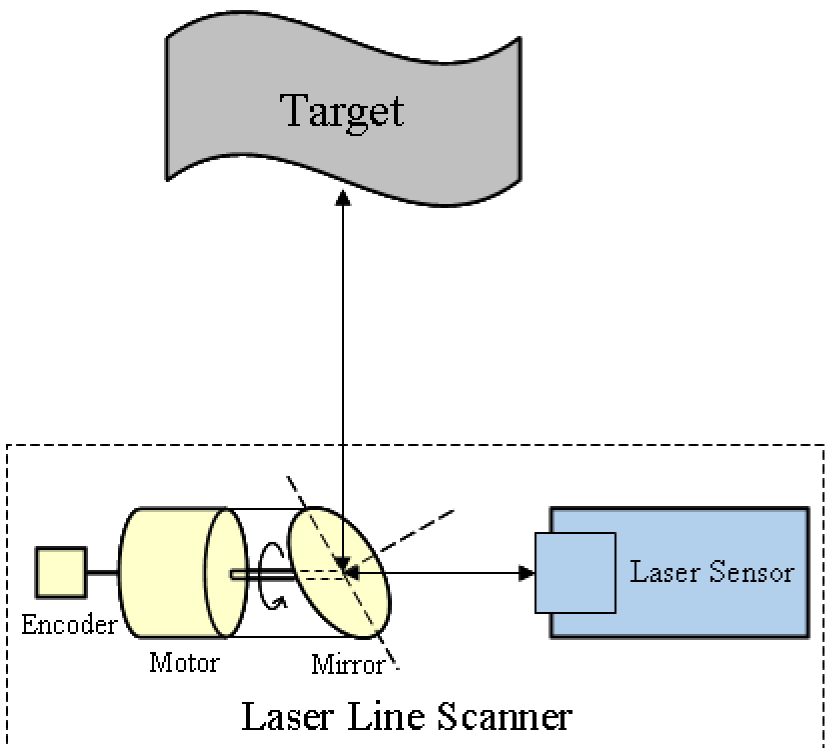



**Table 3.1 Technical Specifications of the Laser Line Scanner**

| | |
|---|---|
| Laser Model | AccuRange 4000-LIR |
| Laser Type | 780 nm IR Laser Diode |
| Laser Class | IIIb (Avoid direct exposure to the beam) |
| Laser Power | 8 milliwatts maximum, +5 volts |
| Effective Range | 0 to 15.24 meters for most surfaces |
| Accuracy | 2.54mm (0.1in), 0.5mm (0.02in) short-term repeatability |
| Sampling Rate | Up to 50 KHz (programmable) |
| Output Interface | RS-232, Pulse Width Output, optional 4-20mA Current loop |
| Motor Speed | Up to 2600rpm @ 12V (programmable) |
| Encoder | 4096 counts/revolution |

The AccuRange 4000-LIR had two interface cables. One was a 9-pin connector for standard RS-232 serial port communications. The port was configured for 8 data bits with no parity, 1 stop bit, and in either the ASCII or the binary format. Output data can be in either an internally calibrated distance format, or an uncalibrated format which included information on uncompensated range, signal strength, background light, and sensor temperature. The sensor can be configured via commands sent over the serial port. Configuration information was stored in a nonvolatile EEPROM. Table 3.2 provides a quick reference of commands.

**Table 3.2 Commands for AccuRange Laser Scanner**

| **Command Name** | **Length** | **Command Code** |
|---|---|---|
| Set Sample Interval | 3-8 bytes | ASCII Code: S<Interval> (20<=Interval<=9999999) |
| Set Maximum Range | 1-6 bytes | ASCII Code: F[<MaxRange>] (0<=MaxRange<=99999) |
| Laser Power On | 1 byte | ASCII Code: H |
| Laser Power Off | 1 byte | ASCII Code: L |
| Enable Serial Data Output | 2 bytes | ASCII Code: A<Mode> (Mode: 1=English, 2=low level, 3=flowctl, 4=Metric(mm)) |
| Disable Serial Data Output | 2 bytes | ASCII Code: T<Mode> (Mode: 1=calibrated, 2=low level, 3=flowctl, 4=Metric(mm)) |
| Set Baud Rate | 2 bytes | ASCII Code: B<Baud Rate Code> |
| Set Serial Output to ASCII | 1 byte | ASCII Code: D |
| Set Serial Output to Binary | 1 byte | ASCII Code: N |
| Set Analog Output Mode | 2 bytes | ASCII Code: X[<Mode>] (Mode:1=calibrated, 2=uncalibrated, 3=off) |
| Read Configuration Data From EEPROM | 1 byte | ASCII Code: R |
| Write Configuration Data To EEPROM | 1 byte | ASCII Code: W1234 |
| Reset Configuration to Factory Defaults | 1 byte | ASCII Code: I |
| Show Version Number | 4 bytes | ASCII Code: V1234 |

The other cable was an 8-pin power/signal cable. Output signals transmitted through this cable were analog data formatted as either the Pulse Width Output or Optional 4-20mA current Loop. Descriptions of the wiring for the power/signal cable are given in Table 3.3. All wires were connected straight through, except that the power line passed through an interlock box before powering the laser. The interlock box included a keyswitch and an interlock jack required for CFR certification for Class IIIb lasers (Figure 3.4).

**Table 3.3 Power and Signal Cable Wiring (from Acuity Research Inc., 2000)**

| Wire Color | Function | Direction |
|---|---|---|
| Red | Sensor Power, +5V (5-6V) | In |
| Black | Ground | |
| Orange | Heater Power, +5V (4.5-7V) | In |
| Brown | Heater Power Return | |
| Yellow | Internal Sensor Temperature, 0 to 5 volts | Out |
| Blue | Pulse Width Range Signal, square wave | Out |
| Green | Ambient Light, 0 to 5 volts | Out |
| Purple | Amplitude Light, 0 to 5 volts | Out |
| Shield | Ground at Supply End | |

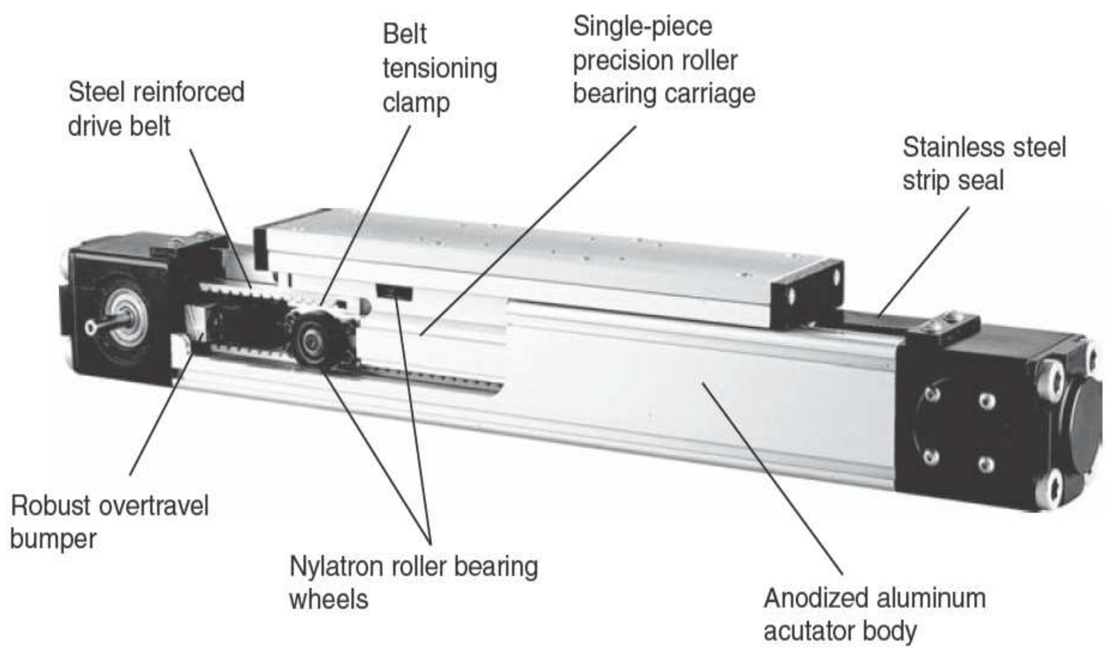
**Figure 3.4 Interlock Box**

### *3.1.2 Frame-rail Unit*

Field applications required a portable system. Factors to be considered for a portable system included flexible mounting to allow adjustment of laser height and scan area, frame integrity to allow easy transportation, and easiness for on-site assembly. The frame-rail mechanism was designed based on these considerations.

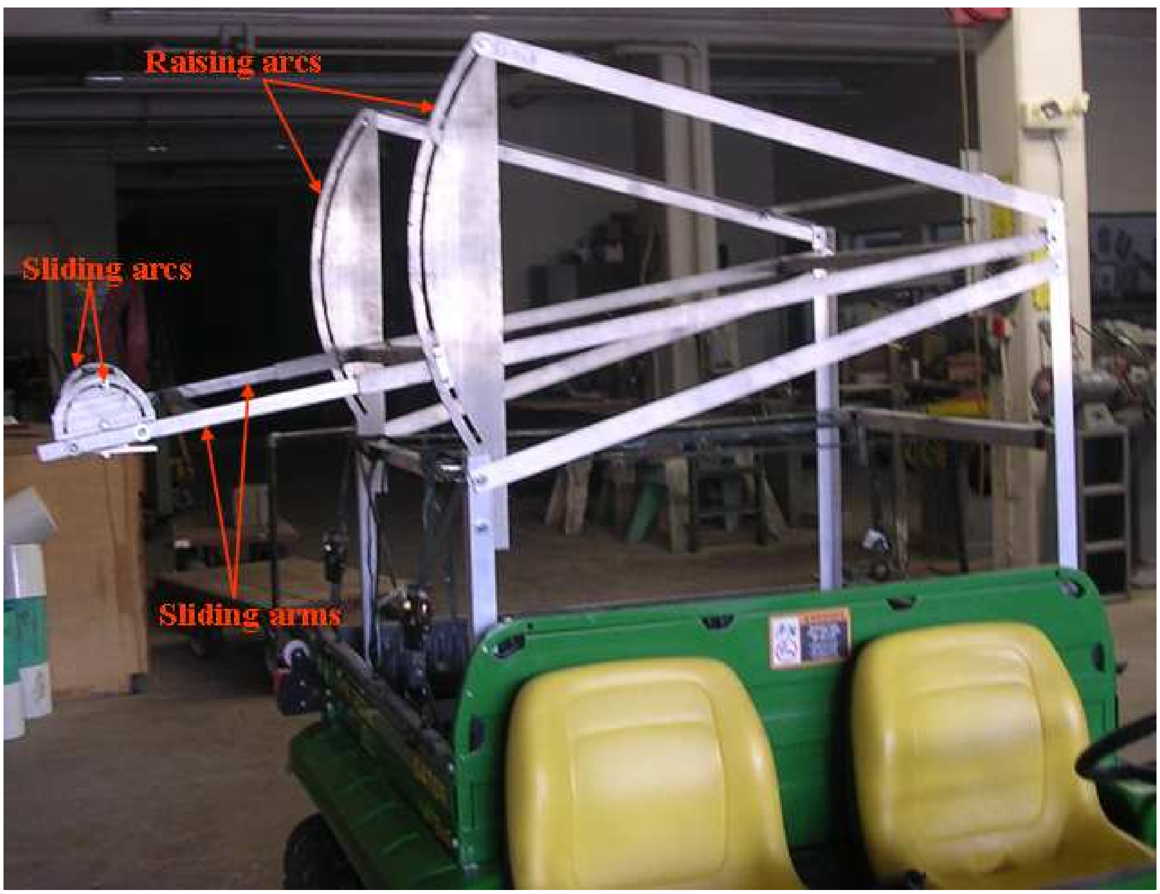
Since the laser line scanner only measured distances along a straight line, a rail was needed to move the sensor in the direction perpendicular to its scan plane for surface measurement. An ER belt-driven actuator (Parker Hannifin Corp., 2004) (Figure 3.5) was selected to move the laser scanner along the rail. The actuator moved the laser scanner through a carriage. A 12V DC, geared motor was mounted at one end of the rail to control the translational motion of the laser scanner. An incremental optical encoder with 200 pulses per rotation (BEI Industrial Encoders, model H20) was attached to the motor to provide feedback signals on the position of the laser scanner. A gyroscope sensor and an RTK GPS were also mounted on the rail.

**Figure 3.5 ER Belt-driven Actuator (from Parker Hannifin Corp., 2004)**



A portable frame was designed to support the rail. Figure 3.6 shows the construction of the aluminum frame mounted on the bed of a John Deere Gator. This frame had a sliding structure that allowed two sliding arms to extend up to 1 meter. The sliding arms could also be raised for more than 1m with the help of the raising arcs. By adjusting the length and height of the sliding arms, the scanning area can be adjusted. At the end of the sliding arms, a 1/4 inch thick, flat plate was mounted on two small sliding arcs. The linear rail was mounted on the flat plate with four toe clamps. This tilting mounting made the field of view of the laser scanner adjustable. The rectangular base of the frame could easily fit to the bed of a small utility vehicle, such as John Deere Gator and Golf Club Car.

**Figure 3.6 Adjustable Aluminum Frame to Support the Rail**



### *3.1.3 Frame Angular-position Measuring Unit: Gyroscope Sensor*

A gyroscope sensor, Crossbow AHRS400CD-100 Motion and Attitude Sensing Unit (Crossbow Technology Inc., 2005a), was selected to measure the pitch, roll, and yaw angular displacements of the frame under static and dynamic conditions. The "AHRS", short for "Attitude and Heading Reference System", was a nine-axis measurement system that integrated linear accelerometers, rotational rate sensors, and magnetometers. It used a 3-axis accelerometer and 3-axis rate sensor to accomplish complete measurements of the system dynamics. The addition of a 3-axis magnetometer also allowed the sensor to make a true measurement of magnetic heading. More detailed specifications of the sensor are listed in Table 3.4.

**Table 3.4 Specifications of the AHRS400CD-100 Gyroscope Sensor (from Crossbow Technology Inc., 2005b)**

| Specification | AHRS400CD-100 | Remarks |
|---|---|---|
| Performance | | |
| Update Rata (Hz) | >50 | Continuous update mode |
| Start-up Time Valid Data (sec) | <1 | |
| Fully Stabilized Data (sec) | <60 | Under static conditions |
| Attitude | | |
| Range: Roll, Pitch (°) | ±180, ±90 | |
| Static Accuracy (°) | <±0.75 | |
| Dynamic Accuracy (°) | ±2.0 | |
| Resolution (°) | < 0.1 | |
| Heading | | |
| Range (°) | ±180 | |
| Static Accuracy (°) | <±1.5 | |
| Dynamic Accuracy (° rms) | ±3 | |
| Resolution (° rms) | < 0.1 | |
| Angular Rate | | |
| Range: Roll, Pitch, Yaw (°/sec) | ±100 | |
| Bias: Roll, Pitch, Yaw (°/sec) | <±1.0 | |
| Bias: Roll, Pitch, Yaw (°/sec) | <±0.05 | |
| Scale Factor Accuracy (%) | < 1.0 | |
| Non-Linearity (% FS) | < 0.3 | |
| Resolution (°/sec) | < 0.025 | |
| Bandwidth (Hz) | > 25 | -3 dB point |
| Random Walk (°/hr1/2) | < 2.25 | |
| Acceleration | | |
| Input Range: X/Y/Z (g) | ±4 | |
| Bias: X/Y/Z (mg) | <±12 | |
| Scale Factor Accuracy (%) | < 1 | |
| Non-Linearity (% FS) | < 1 | |
| Resolution (mg rms) | < 0.5 | |
| Bandwidth (Hz) | > 10 | -3 dB point |
| Random Walk (m/s/hr1/2) | < 0.1 | |
| Environment | | |
| Operating Temperature (°C) | -40 to +71 | |
| Non-Operating Temperature (°C) | -55 to +85 | |
| Non-Operating Vibration(g rms) | 6 | |
| Non-Operation Shock (g) | 1000 | |
| Electrical | | |
| Input Voltage (VDC) | 9 to 30 | |
| Input Current (mA) | < 300 | |
| Power Consumption (W) | < 4 | |
| Digital Output Format | RS-232 | |
| Analog Range (VDC) | ±4.096 | |
| Physical | | |
| Size (cm) | 7.62*9.52*10.41 | Incl. mounting flanges |
| Weight (kg) | < 0.77 | |
| Connector | 15 pin sub-miniature | |

A Cartesian coordinate system the AHRS400 gyroscope sensor used was illustrated on the front face of the sensor (Figure 3.7). The pitch, roll, and yaw angles were defined as the angular displacements about the Y, X, and Z axes, respectively. Directions of these angles were defined using the "right-hand rule". The AHRS400 sensor utilized a sophisticated Kalman filter algorithm to track orientation accurately through dynamic maneuvers. The Kalman filter automatically adjusted for changing dynamic conditions without user inputs.

**Figure 3.7 The AHRS400 Gyroscope with its Coordinate System Illustrated on the Front Face**


The AHRS400 gyroscope sensor had a male DB-15 connector (Figure 3.7). Table 3.5 lists the pin assignments. The sensor provided both analog and RS-232 outputs. Data can be requested via the serial link as a single polled measurement or continuous stream measurements. The analog outputs were fully conditioned and can be connected directly to an analog data acquisition device. The serial interface was standard RS-232 configured for 38,400 baud rate, 8 data bits, 1 start bit, 1 stop list, no parity, and no flow control.

**Table 3.5 Pin Assignments of the AHRS400 Gyroscope Connector (from Crossbow Technology Inc., 2005a)**

| Pin | Signal |
|---|---|
| 1 | RS-232 Transmit Data |
| 2 | RS-232 Receive Data |
| 3 | Positive Power Input |
| 4 | Ground |
| 5 | X-axis accelerometer Analog voltage |
| 6 | Y-axis accelerometer Analog voltage |
| 7 | Z-axis accelerometer Analog voltage |
| 8 | Roll rate analog voltage |
| 9 | Pitch rate analog voltage |
| 10 | Yaw rate analog voltage |
| 11 | NC – factory use only |
| 12 | Roll angle/X-axis magnetometer scaled analog voltage |
| 13 | Pitch angle/Y-axis magnetometer scaled analog voltage |
| 14 | Yaw angle/Z-axis magnetometer scaled analog voltage |
| 15 | NC – factory use only |

There were three measurement modes in AHRS400: voltage mode, scaled sensor mode, and angle mode. In the voltage mode, only the accelerometer analog outputs were available and they were converted to unsigned 12-bit digital data with 1 mV resolution. In the scaled sensor mode, the analog signals were sampled and converted to the digital form. The sampled data were temperature compensated, corrected for misalignment, and scaled to the engineering units. The data was sent as singed, 16-bit, 2's complement integers. In the angle mode, the gyroscope sensor acted like a complete attitude and heading reference system to output stabilized pitch, roll and yaw angles along with their angular rate, angular acceleration, and magnetic field information. The Kalman filter operated only in the angle mode.

The AHRS400 gyroscope sensor had a simple command structure. The one-byte commands can be sent to the sensor over the RS-232 interface. A list of the commands is shown in Table 3.6.

**Table 3.6 AHRS400 Gyroscope Command Quick Reference (from Crossbow Technology Inc., 2005a)**

| Command (ASCII) | Response | Description |
|---|---|---|
| R | H | Ping: Pings AHRS to verify communications |
| r | R | Change to Voltage Mode |
| c | C | Change to Scaled Sensor Mode |
| a | A | Change to Angle Mode (VG Mode) |
| P | None | Change to polled mode. Data packets sent when a G is received by the DMU. |
| C | None | Change to continuous data transmit mode. Data packets streamed continuously. Packet rate is dependent on operating mode. Sending “G” stops data transmission. |
| G | Data Packet | Get Date. Requests a packet of data from the DMU. Data format depends on operating mode. |
| S | ASCII String | Query DMU serial number. Returns serial number as 32-bit binary number. |
| v | ASCII String | Query DMU version ID string. Return ASCII string |
| b | Change baud rate | Autobaud detection. Send “b”; DMU will respond “B”; change baud rate; send “a”; DMU will send “A” when new baud rate is detected. |
| s | S | Start Hard/Soft iron calibration. DMU should be rotated through at least one complete turn (360° of rotation) with the system basically level. |
| u | U | End Hard/Soft iron calibration. Calibration is saved in EEPROM. |
| h | H | Clear Hard iron calibration |
| t | T | Clear Soft Iron calibration |

Ideally, the gyroscope sensor used its magnetic sensors to only measure Earth’s weak magnetic field to determine the heading (yaw angle). In the real world, however, residual magnetism in the sensor and the surrounding environment would add to the measured magnetic field. These extra magnetic fields were called “hard iron magnetic fields”. In addition, extra magnetic material can change the direction of measured magnetic field. This effect was called “soft iron effect”. Both hard iron and soft iron effects may create errors in the heading measurement. As a result, even small amounts of moving magnetic material near the gyroscope can change the heading measurement (Crossbow Technology Inc., 2005a).

The gyroscope can correct for the magnetic fields through hard and soft iron calibration. The calibration was done by making a series of measurements with the extra magnetic field and then using these measurements to model the hard iron and soft iron environment in the system. However, the calibration would not help for time-varying magnetic fields, or fields created by large moving parts within 60.96 cm distance from the sensor.

The hard and soft iron calibration can be performed using the GyroView software provided by Crossbow Technology, Inc (2005a). The calibration required the following steps:

1). Power on the gyroscope and start GyroView.

2). Click on the “CLEAR CAL” button when the sensor was still and leveled. This cleared the old hard and soft iron calibration.

3). Click on “START CAL” button to command the sensor to enter the calibration process.

4). Rotate the system through a complete circle. The system did not have to be perfectly leveled as the algorithm would compensate for any angle offsets.

5). Click on “STOP CAL” when the turn was done.

6). Wait at least 1 minute. The algorithm would initialize if the system was motionless.

The quality of hard and soft iron calibration can be evaluated and tested by observing the magnetometer outputs. The following procedure was followed:

1). Power up the gyroscope and start GyroView.

2). Click on the “START LOG” to start logging the data

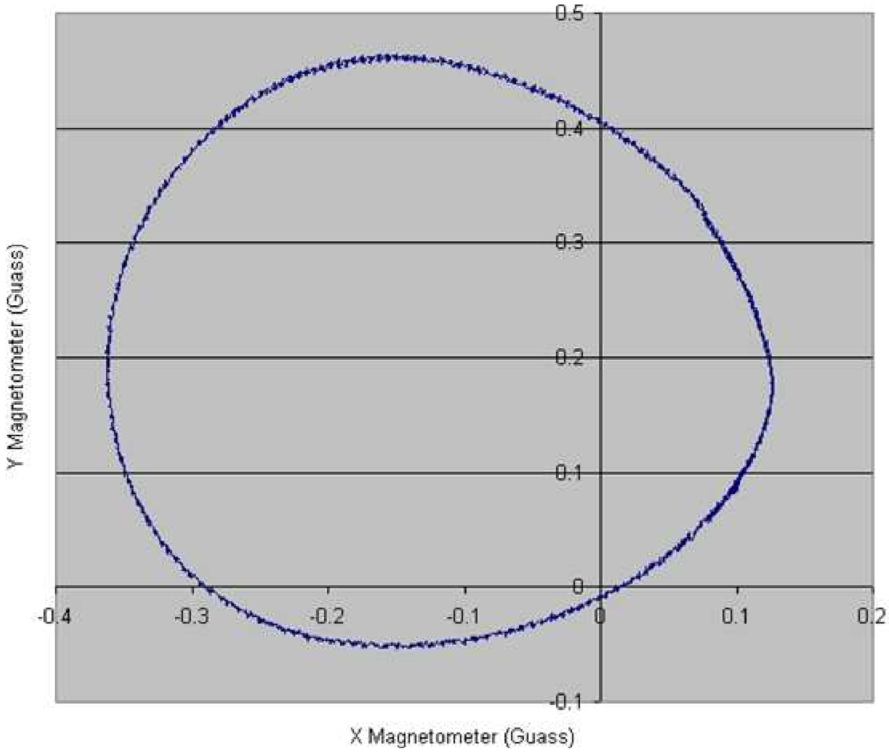
3). Rotate the gyroscope slowly about the z axis through one complete turn.

4). Click on the “STOP LOG” to stop logging

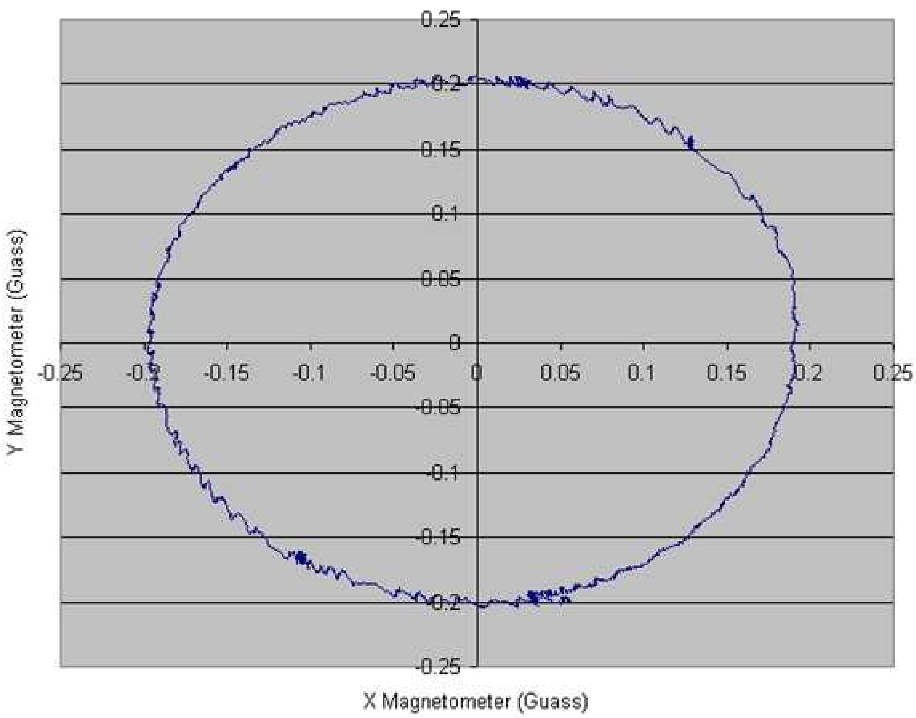
5). Plot the X-mag vs Y-mag from the data file.

A successful calibration would give a perfect circle centered around the origin (Figure 3.8). If the circle was not circular or the center of the circle had an offset, repeat the calibration procedure without clearing the existing calibration. Usually a successful calibration can not be achieved after the first turn. Multiple calibration routines were often necessary.

**Figure 3.8 Evaluation of the Hard and Soft Iron Calibration: (a) Before Hard and Soft Iron Calibration, and (b) After Hard and Soft Iron Calibration**




**(a)**




**(b)**

### *3.1.4 Geo-reference Unit: RTK GPS*

The geo-reference unit was for providing the global position information of the measured surface and converting the measurement results from a local coordinate system to the global coordinate system. A Real-time Kinematic (RTK) GPS, Topcon HiPer Lite+ (Topcon Positioning System Inc., 2006), was used to help register the measured surface into a geographic coordinate system.

The Topcon HiPer Lite+ RTK System (Figure 3.9) consisted of a HiPer Lite+ Base, a standard HiPer Lite+ Rover, and a FC-100 data collector with TopSURV data collection software. The Topcon Hiper Lite+ was a robust, fast set-up, completely cable-free system with an advanced internal radio technology that provided interference-free data links of up to 1.5 miles radius. The HiPer Lite+ gave static accuracies of 3 mm horizontal and 5 mm vertical. The kinematic accuracies were 10 mm horizontal and 15 mm vertical. More detailed specifications are given in Table 3.7.

**Figure 3.9 Topcon HiPer Lite+ GPS System with FC-100 Controller**

**Table 3.7 Specifications of the HiPer Liter+ RTK System (from Topcon Positioning System Inc., 2006)**

| | |
|---|---|
| **Tracking Specifications** | |
| Tracking Channels, standard | 40 L1 GPS |
| Signals Tracked | L1/L2 C/A and P Code & Carrier and GLONASS |
| **Performance Specifications** | |
| Static, Rapid Static | H:3mm+0.5ppm, V:5mm+0.5ppm |
| RTK | H:10mm+1ppm, V:15mm+1ppm |
| **Power Specifications** | |
| Battery | Internal Lithium-Ion batteries for up to 14+ hours of operation |
| External power input | 6 to 28 volts DC |
| Power consumption | Less than 4.2 watts |
| **GPS + Antenna Specifications** | |
| GPS/GLONASS Antenna | Integrated |
| Antenna Type | Center-mount Spread Spectrum Antenna |
| Ground Plane | Antenna on a flat ground plane |
| **Radio Specifications** | |
| 915 MHz SpSp Modem | Internal Tx/Rx |
| Power Output | 1.0W/0.25W (selectable) |
| **Wireless communication** | |
| Communication | Bluetooth™ version 1.1 comp. |
| **I/O** | |
| Communication Ports | 2 serial (RS232), USB |
| Other I/O Signals | 1pps, Even Marker (optional) |
| Status Indicator | 4×3- color LED's, two-function keys (MINTER) |
| Control & Display Unit | External Field Controller |
| **Memory & Recording** | |
| Internal Memory | Up to 128 MB |
| Data Update Rate | Up to 20 Hz |
| Data Type | Code and Carrier from L1 and L2, GPS and GLONASS |
| **Data Input/Output** | |
| Real time data outputs | RTCM SC104 version 2.1,2.2,2.2,3.0,CMR, CMR+ |
| ASCII Output | NMEA 0183 version 3.0 |
| Other Outputs | TPS format |
| Output Rate | Up to 20 Hz |
| **Environmental Specifications** | |
| Enclosure | Aluminum extrusion, waterproof |
| Operation temperature | -30 °C to 55 °C |
| Dimensions | W:159×H:172×D:88 mm/ 6.25×6.75×3.5in |
| Weight | 1.65 kg/3.64 lbs |

The Topcon's FC-100 was a field controller to communicate with the HiPer Lite+ and store the data. It incorporated a graphical, Windows CE operating system on an ultra-bright, sunlight-active, thin film transistor (TFT), color touch-screen display. The FC-100 was equipped with a mini USB data port and an RS-232C serial data port. Bluetooth was also available to wirelessly connect to the HiPer Lite+. With the installed TopSURV software, FC-100 can set up the configurations for both the Base station and the Rover, and store data into the secure digital (SD) cards or internal memory.

The HiPer Lite+ GPS had two serial ports, a USB port, and a power connector (Figure 3.10).  By setting the serial port D through a set of commands, the standard GPS data can output directly at a predefined update rate.

**Figure 3.10 Ports on the HiPer Lite+ GPS**

Both the Base and the Rover can be configured using the TopSURV program in the FC-100 field controller. The TopSURV software provided the following functions to set up the RTK GPS system:

1). Job creation: to save all configurations of the system;

2). Survey configurations: to declare the communication settings of the system;

3). Projection definition: to select a geographic coordinate system;

4). Start the Base: to broadcast RTK correction signals;

5). Fix the Rover: to listen to the Base radio;

6). Status indicator: to check the radio link;

7). Data management.

The coordinates were saved in a text file in the controller's memory after the RTK GPS system had been configured successfully. Data file then can be uploaded to the field computer for further analysis. However, the laser system required the NEMA GGA string output from the Rover. A special command had to be sent to the Rover to output the GGA string from its serial port D. The command was:

```
%%em,/dev/ser/d,/msg/nmea/GGA:1
```

The output rate can be set by changing the last field of the command. For instance, using ":0.2" for 5 Hz. This script can be sent using the "Manual Mode" option under the "file" menu in CE-CDU (controller software) or PC-CDU (office software).

### *3.1.5 Data-acquisition and Control Unit*

The laser line scanner was configured for use with a High Speed Interface (HSIF) to increase the sample rate. The power/signal cable provided the pulse width output. A standard PCI HSIF card was employed to measure the duration of the pulse width output from the laser sensor (Figure 3.11). This HSIF card also had two pulse-width modulated motor control channels, two 32-bit quadrature decoders to read motor encoders, and three general purpose inputs. A motor control channel with a quadrature decoder was used to monitor the angular displacement of the rotating mirror. Another motor channel and encoder reader were wired to the DC motor on the linear rail. The input and output connectors on the PCI HSIF card (P1 and P2) are illustrated in Table 3.8 and 3.9, respectively. P1 supplied power and received signals from the laser sensor. P2 controlled the power levels of two motors and read the two encoders.

**Figure 3.11 AccuRange High Speed Interface Card**

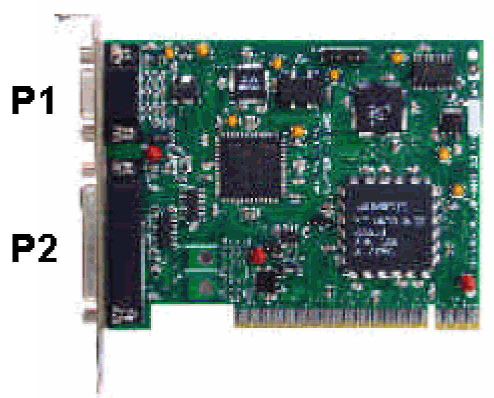



**Table 3.8 Pin Assignments of the Power and Signal Connector (P1) (from Acuity Research Inc., 2000)**

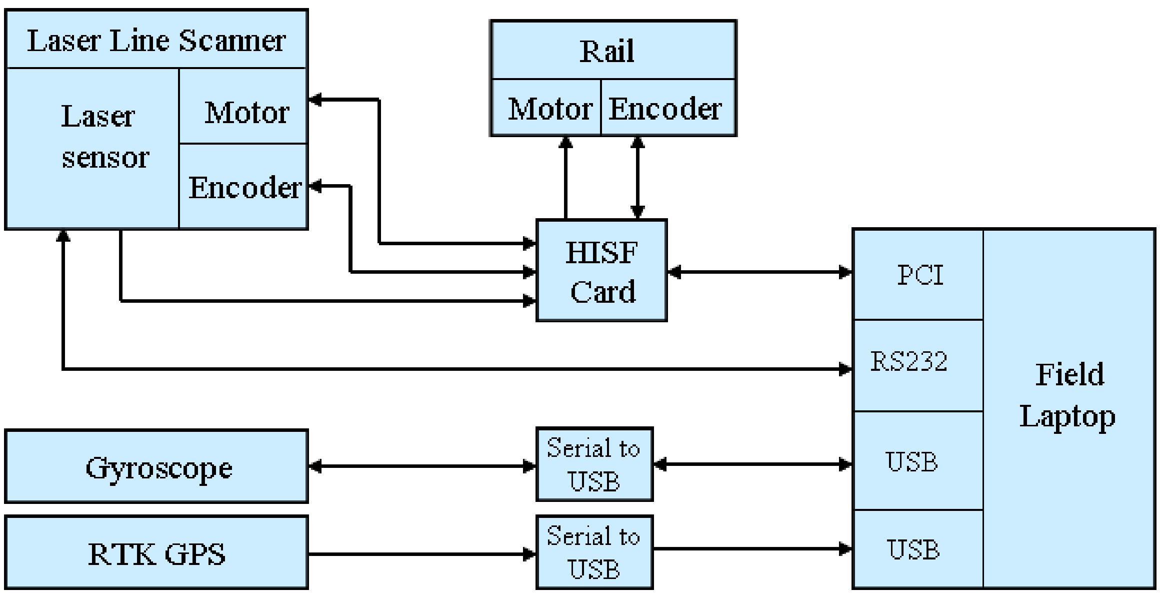
| Pin | AR4000 wire | Function | Direction |
|---|---|---|---|
| 1 | Red | Power, +5V | out |
| 2 | Black | Ground | |
| 3 | Orange | Heater Power, +5V (4.5~7V) | out |
| 4 | Brown | Heater Power Return | |
| 5 | Yellow | Temp, 0~5V | in |
| 6 | Blue | Pulse Width Range Signal | in |
| 7 | Green | Ambient Light Signal, 0~5V | in |
| 8 | Purple | Amplitude Signal, 0~5V | in |
| 9 | Not used | Laser Control, 0~5V | out |

**Table 3.9 Pin Assignments of the I/O Connector (P2) (from Acuity Research Inc., 2000)**

| Pin | Function | Direction |
|---|---|---|
| 1 | Motor 2 Control | out |
| 2 | Motor 2 Return | Out |
| 3 | Motor Power Supply | In |
| 4 | Ground | |
| 5 | +5V Power, 100mA | Out |
| 6 | Ground, Motor 2 encoder Ch A- (dif) | in |
| 7 | Ground, Motor 2 encoder Ch B- (dif) | in |
| 8 | Ground, Motor 1 encoder Ch A- (dif) | in |
| 10 | Ground, Motor 1 encoder Ch B- (dif) | in |
| 11 | Gen. Purpose Input 2/ Encoder 2 index pulse – (dif) | in |
| 12 | Start/Stop sample control | in |
| 13 | Gen. Purpose Input 2/ Encoder 2 index pulse + | in |
| 14 | Motor 1 Control | out |
| 15 | Motor Power Ground | |
| 16 | Motor 1 Return | out |
| 17 | Laser Control | out |
| 18 | +5V Power, 100 mA | out |
| 19 | Motor 2 encoder Ch A+ | in |
| 20 | Motor 2 encoder Ch B+ | in |
| 21 | Motor 2 encoder Ch A+ | in |
| 22 | Motor 1 encoder Ch B+ | in |
| 23 | Gen. Purpose Input 1/ Encoder 1 index pulse - | in |
| 24 | Gen. Purpose Input 1/ Encoder 1 index pulse + (dif) | in |
| 25 | Gen. Purpose Input 3 | in |

The gyroscope sensor had a 15-pin to 9-pin converter cable to connect the sensor with a standard serial port. The rover of RTK GPS was configured as a standard serial interface through the receiver's port D. A ruggedized field laptop computer (GETAC Inc., model GETAC A770) was used to acquire data from different components. Figure 3.12 displays the connections among the system components. Two RS232 ports were converted to USB ports using serial-to-USB converters due to the limited number of RS232 serial ports available on the field laptop.

**Figure 3.12 The Data Acquisition System**

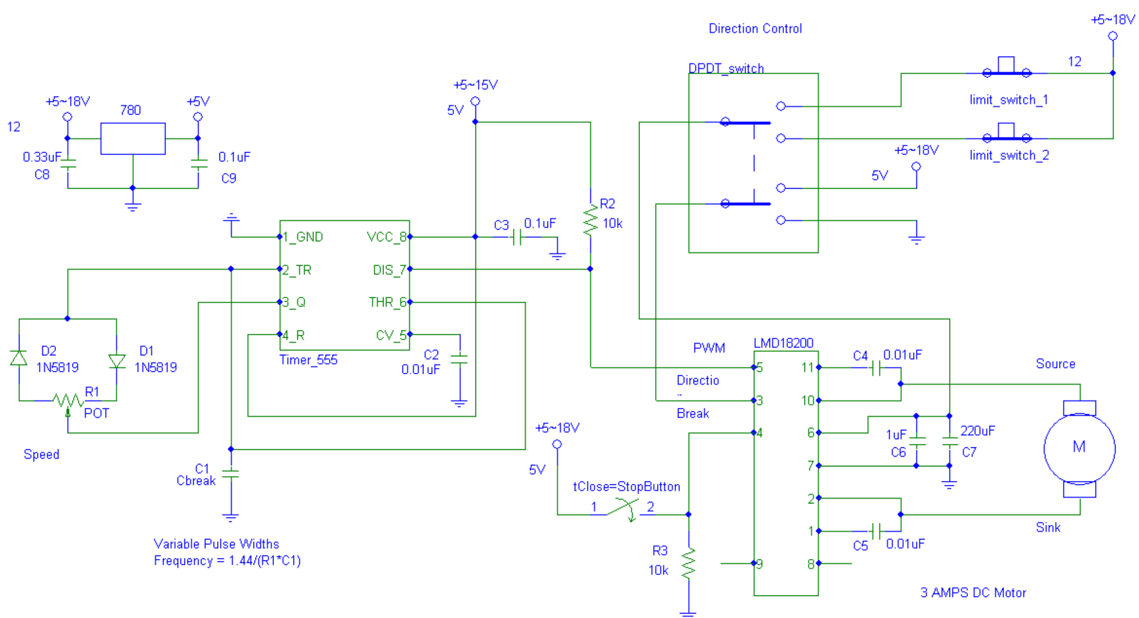



### *3.1.6 Power and Control Circuitry*

For any field applications, the power supply is always an issue. The laser system was originally designed to operate under field conditions. The laser line scanner with a heater powered on required +5 volts at 4A. The input voltage of the gyroscope sensor ranged from 9 to 30 volts at 275 mA. Power consumption of the GPS receiver was less than 4.5 watts. The laser mirror was driven by a DC motor that consumed 12 volts at 110 mA. The maximum power required by the DC motor driving the carriage on the rail was 17 watts. Thus, the total power consumption of the system was approximately 47 watts. The system can run for approximately 21 hours at 4A on the vehicle battery, which was rated at 85 Ah. Another 12 V deep cycle battery of 200 Ah was also provided as a backup power source.

The movement speed of the laser-carriage on the linear rail was one of the major factors affecting the spatial resolution of the measurements. To control the speed, a circuit was designed to drive the DC motor on the rail. Figure 3.13 shows the schematic of the DC motor controller. This controller controlled the motor through pulse width modulation (PWM), which changed the duty cycle of the power at a fixed frequency. A 555 timer was configured as a stable oscillator in the circuitry to generate the required PWM signal at a frequency of 2 KHz. A commercially available LMD18200 H-Bridge was used to drive the DC motor with direction control and stop function. The circuit provided a near linear relationship between the PWM duty cycle and motor speed. A 1K multi-turn potentiometer, R1, was used to control the duty cycle. To avoid the laser from moving out of the linear rail, a limit switch was attached at each end of the rail. Once a limit switch was tripped by the carriage, the power would be cut off.

**Figure 3.13 Schematic of the DC Motor Controller**



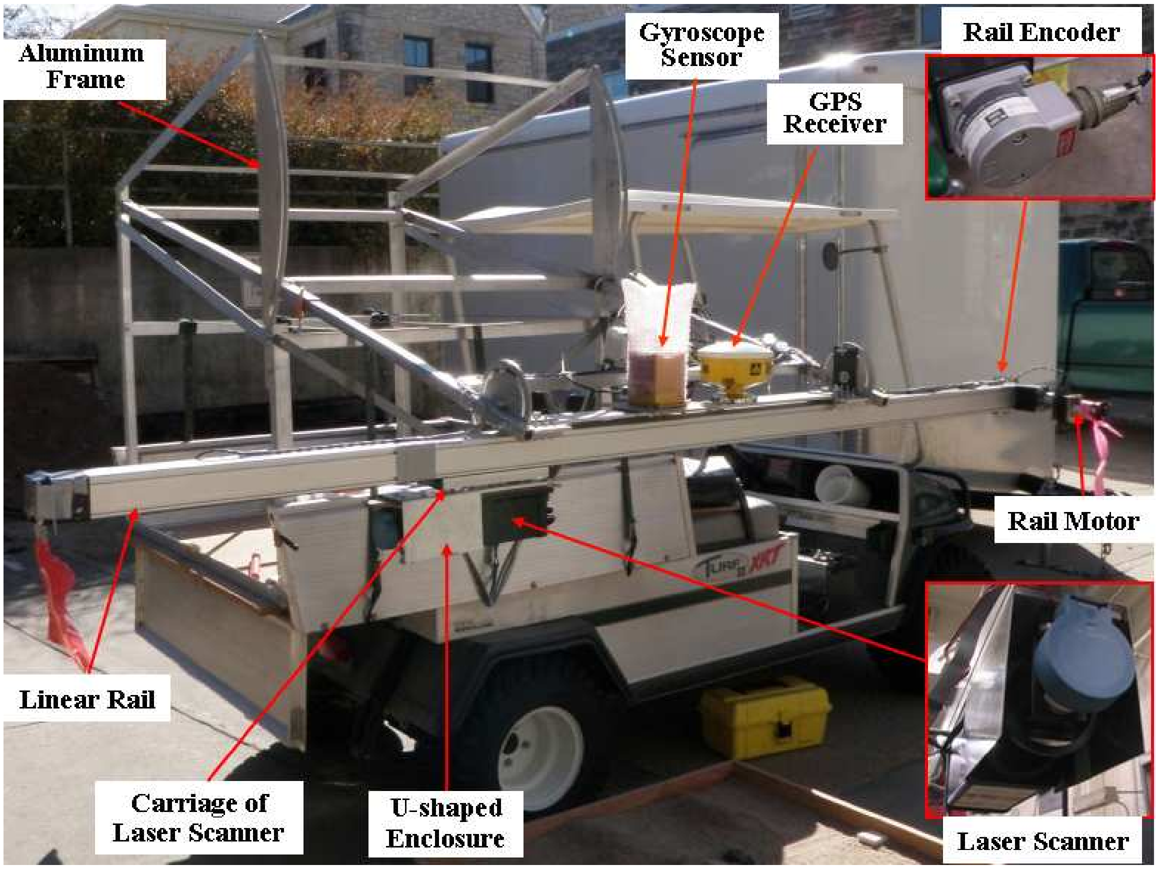
The motor control circuit was secured in a black control box (Figure 3.14). The control box also provided power connections between the battery and all the components.

**Figure 3.14 Black Control Box with a Cigarette Lighter Adepter**


### *3.1.7 System Assembly*

Figure 3.15 shows all the components of the system installed on a frame that was tied on the bed of a Club car. The laser line scanner was mounted on a carriage that moved along the linear rail and the rotating mirror was oriented towards the ground. For safety, the laser beam was covered by a U-shaped enclosure to limit the field of view of the laser to 100 degree. The gyroscope sensor was tightened on an aluminum frame with nonmagnetic copper screws. The GPS receiver was installed on the same frame.

**Figure 3.15 The Laser System Assembled on the Club Car**




## 3.2 Software Design

### *3.2.1 AR4000 PCI High Speed Interface*

The AR4000 High Speed Interface was a PCI board that took readings from the laser line scanner. It sampled laser data at a sampling frequency of up to 50,000 Hz and buffered the data in the onboard memory for reading by the host computer. Data collected by the PCI HISF was not scaled or calibrated. For maximum accuracy, the actual distances were calculated by the host computer using calibration tables supplied with the interface.

The HSIF card application program interface (API) employed routines accessible through the Acuity HSIF dynamic link libraries (DLL's) and windows driver. Descriptions of the basic API's are listed in Table 3.10.

**Table 3.10 Summary of Frequently Used Functions in the HSIF's API**

| **Function Name** | **Description** |
|---|---|
| HsifDllInit | Initializes the library before using any of its function |
| HsifOpen | Opens communications with the PIC HSIF card and returns its handle. |
| HsifClose | Closes the application's access to the card |
| HsifSamplingModeInit | Sets HSIF card to sampling mode |
| HsifResetBoard | Performs a hardware reset of the card |
| HsifGetBufferedSamples | Gets buffered samples |
| HsifSetSampledPeriod | Sets the sample period and maximum range |
| HsifProcessSamples | Calculates calibrated range measurements |
| HsifLoadCalibrationData | Loads calibration file to generate true distance measurements |
| HsifSamplingEnable | Enables sampling |
| HsifClearSampleBuffer | Clears sample buffer |
| HsifSetMotorPower | Sets power for motors 1 and 2 |
| HsifClearEncoder | Calibrates the encoders |
| HsifCalibrate | Calibrates HSIF card |
| OpenPort | Opens a com port to communicate with the laser sensor |
| ClosePort | Closes com port |

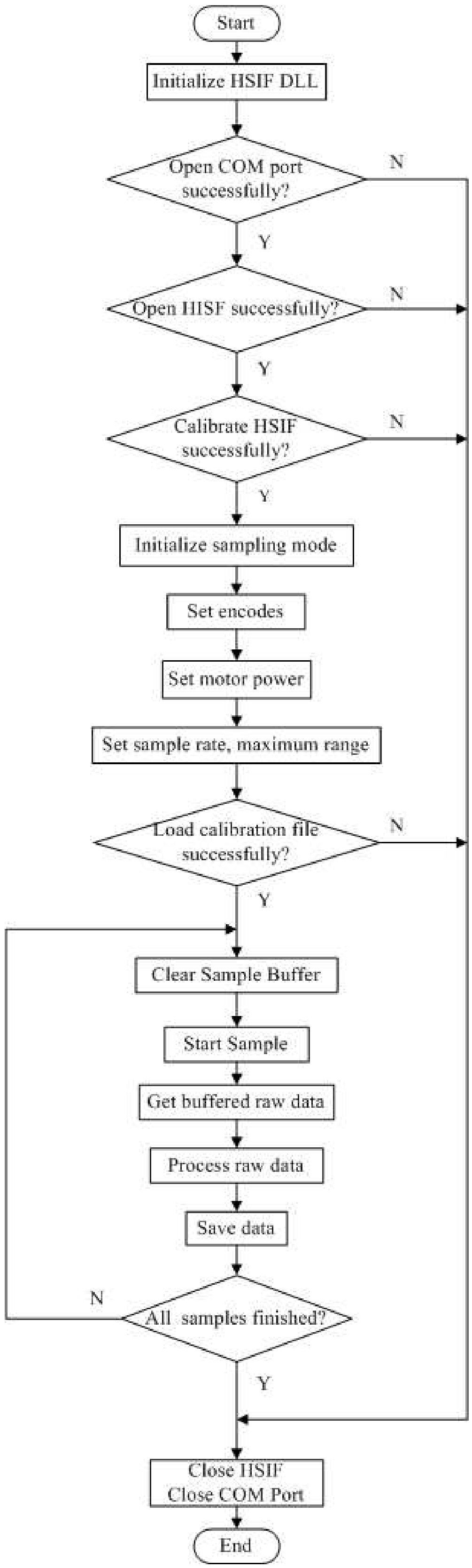
The PCI HSIF collected 16 bytes/sample in a sequential data stream which were read with function HsifGetBufferedSamples. Each sample included a 32-bit range reading, two 32-bit encoder readings, 1 byte for signal strength, 1 byte for ambient light, 1 byte for sensor internal temperature, two general purpose input bits, and others. The sampled data format is shown in Table 3.11.

**Table 3.11 Sampled Raw Data Format (from Acuity Research Inc., 2000)**

| Word | Bit # | Contents |
|---|---|---|
| **0** | 0 | buffer overflow indicator |
| | 1 | Input 3 |
| | 2 | Motor 1 encoder index / Input 1 |
| | 3 | Motor 2 encoder index / Input 2 |
| | 4-7 | sample count, 0-15 repeating |
| | 8-15 | 8-bit Sensor internal temperature |
| | 16-23 | 8-bit ambient light sample |
| | 24-31 | 8-bit amplitude sample |
| **1** | 0-15 | low word of 32-bit Motor 1 encoder position |
| | 16-31 | high word of 32-bit Motor 1 encoder position |
| **2** | 0-15 | low word of 32-bit Motor 2 encoder position |
| | 16-31 | high word of 32-bit Motor 2 encoder position |
| **3** | 0-15 | low word of 32-bit range |
| | 16-31 | high word of 32-bit range |

A C program was written to sample the laser data (Appendix A). Figure 3.16 shows the flowchart of the program. To generate the true distance measurement from the raw range data, a calibration file named LOOKUPHS was included. This program was compiled with Microsoft Visual C++ 6.0 under Windows XP.

**Figure 3.16 Flowchart of the Laser Sampling Program**



### *3.2.2 Serial Communications in Microsoft Win32*

In a data packet of the gyroscope sensor, each measured variable was sent in two bytes, starting with the MSB. Each data packet began with a header byte 255 (0xFF) and ended with a checksum. These digital data can be converted to actual measurements using conversion formulas in three different modes, as shown in Table 3.12.

**Table 3.12 Data Conversion Formulas**

| Measurement Mode | Formula | Note |
|---|---|---|
| Voltage Mode | voltage = data * (5 V)/$2^{12}$ | "voltage" is the voltage measured by the sensor; "data" is measurement data in the data packet, in unsigned 16-bit integer format. |
| Scaled Sensor Mode | accel = data*(GR*1.5)/$2^{15}$<br>rate = data*(AR*1.5)/$2^{15}$<br>mag = data*(MR*1.5)/$2^{15}$ | "accel" is the measured acceleration in G's, "GR" is the G range; "rate" is the measured angular rate in °/sec; "AR" is the angular rate range; "mag" is the measured magnetic field in Gauss; "MR" is the magnectif field range; "data" is the measurement data in the data packet. |
| Angle Mode | angle = data*(SCALE)/ $2^{15}$ | "angle" is the measured angle in degrees; "data" is the measurement data in the data packet, in signed integer format. |

The GPS receiver outputted the NMEA-0183 standard messages. NMEA-0183 messages began with a dollar sign ($), followed by a talker ID code (GP) and a message ID code (GGA), and ended with a carriage return and a line feed. One of the available messages, the GGA (GPS fix data) message contained the information on latitude and longitude. Table 3.13 describes the fields included in the message.

**Table 3.13 GGA Message Fields**

| Field | Description |
|---|---|
| 1 | UTC of position fix in HHMMSS.SS format |
| 2 | Latitude in DD MM,MMMM format (0-7 decimal places) |
| 3 | Direction of latitude<br>N: North; S: South |
| 4 | Longitude in DDD MM,MMMM format (0-7 decimal places) |
| 5 | Direction of longitude<br>E: East; W: West |
| 6 | GPS Quality indicator<br>0: fix not valid; 1: GPS fix; 2: DGPS fix<br>4: Real-time kinematic, fixed integers<br>5: Real-time kinematic, float integers |
| 7 | Number of SVs in use, 00-12 |
| 8 | HDOP |
| 9 | Antenna height, MSL reference |
| 10 | “M” indicates that the altitude is in meters |
| 11 | Geoidal separation |
| 12 | “M” indicates that the geoidal separation is in meters |
| 13 | Correction age of GPS data record, Type 1; Null when DGPS not used |
| 14 | Base station ID, 0000-1023 |

Both data packets from the gyroscope sensor and the GPS receiver were transmitted to the field laptop via standard serial interfaces. The operating system on the laptop was the Microsoft 32-bit Windows XP professional.

In Microsoft Win32, the serial data communication can be implemented through two techniques: Application Programming Interfaces (APIs) and ActiveX controls. Programs using the ActiveX were simple, clear, but not very flexible. On the contrary, using APIs made programming a bit more difficult but more flexible. Thus, the APIs were used in the program. In Microsoft Windows, a serial port can be treated as a file. Functions in APIs for reading and writing files can be applied to a serial port. To send and receive data through a serial port of a computer, the following procedure was followed:

1). Open a communication port;

2). Get the configuration of the port from Device Control Block;

3). Modify the configuration by setting the baud rate, parity, number of data bits, etc;

4). Store the configuration;

5). Set timeout value for communication;

6). Read or write data to the port;

7). Close the port.

A universal serial communication program was written in C based on these steps (Appendix B). Figure 3.17 shows the flowchart of the program. In this program, a serial port was opened by the CreateFile function in the "overlapped method" (Table 3.11), which was not as straightforward as the "nonoverlapped" I/O, but it allowed more flexibility and higher efficiency. An event was created by the SetCommMask (Table 3.14) function to manage the serial port. Any event, such as connect, disconnect, error, and receiving data, that occurred on the serial port would be notified. The program was compiled with Microsoft Visual C++ 6.0 under Windows XP. This program can be very easily integrated into a windows application with a user interface.

**Figure 3.17 Flowchart of the Program for Universal Serial Communication**

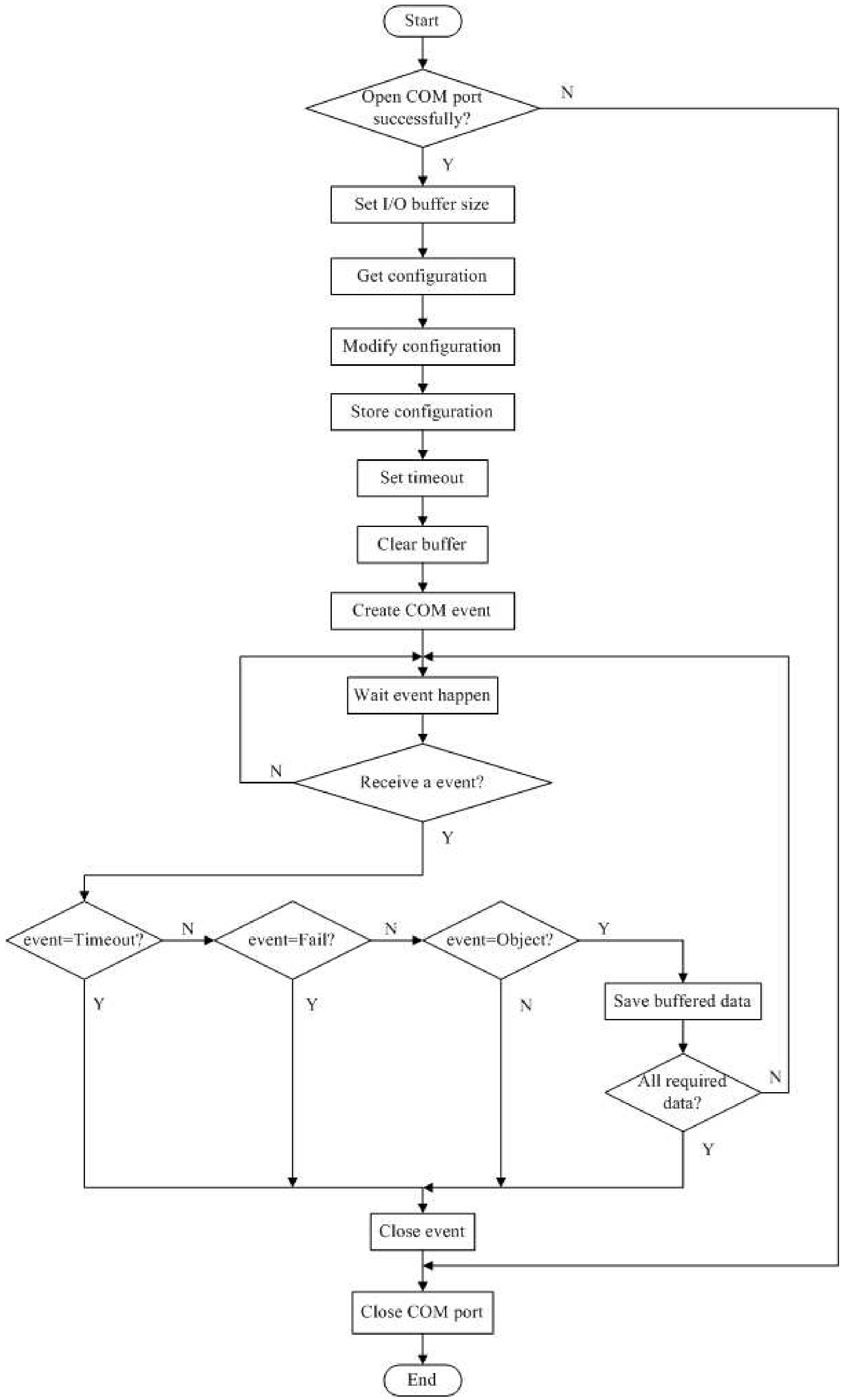


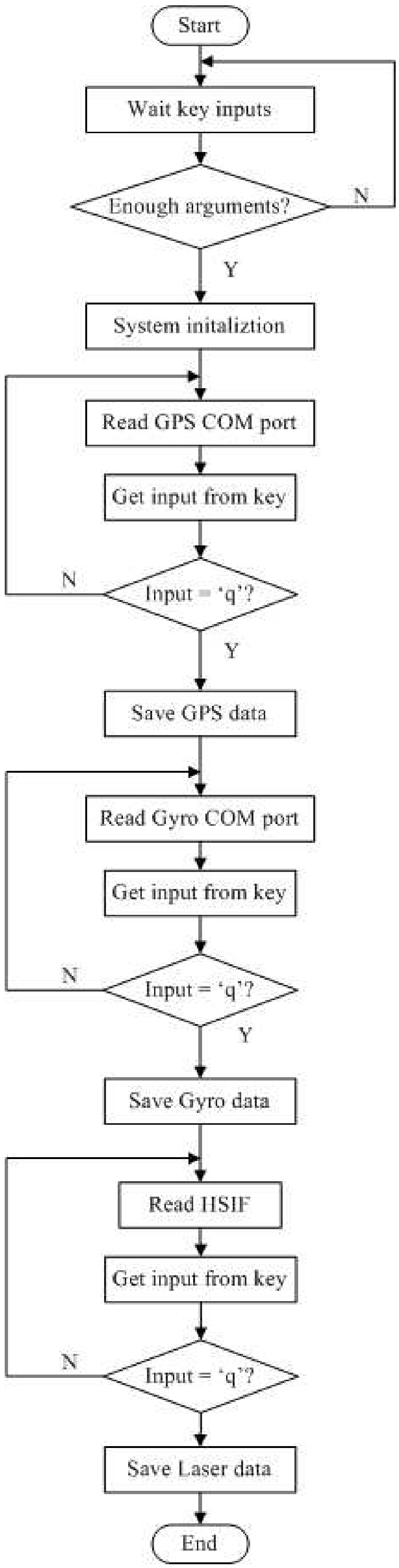
**Table 3.14 Serial Communication Functions in APIs**

| Function | Description |
|---|---|
| CreateFile | Open a communications port and create a handle |
| GetCommState | Restore a current configuration from the serial port |
| SetCommState | Store a configuration into the serial port |
| GetCommTimeouts | Get a current timeout |
| SetCommTimeouts | Set a time-out communication |
| ReadFile | Receive data |
| WriteFile | Send data |
| CloseHandle | Close a serial port |
| SetCommMark | Set the desired events that cause a notification |
| WaitCommEvent | Detect the occurrence of the events |
| ClearCommError | Detect errors and clear the error condition |
| PurgeComm | Clear all buffer |

### *3.2.3 Main Program*

A user interface program was developed in C to control the laser system and to acquire the sensors data through the field laptop (Appendix C). Figure 3.18 shows the flowchat of the program. This main program was a command-line program. It took 18 command line arguments: three data files name, the serial port number for the laser, the HISF card number, the laser sample period in microsecond, the data buffer size, the maximum laser measurement range in inch, the number of motors, the motor power level, the measurement mode of the gyroscope sensor, the serial port number for the gyroscope sensor, the baud rate of the serial port of the gyroscope sensor, the serial port number for the GPS, the baud rate of the serial port of the GPS, the number of bits of the GPS com port, the parity control of the GPS com port, and the number of the stop bit of the GPS com port. There were three raw data files created. One was for a sequential laser samples. Another was for a sequence of the gyroscope data packets. The last one logged the GGA messages from the GPS. The program was compiled with Microsoft Visual C++ 6.0 under Windows XP.

**Figure 3.18 Flowchart of the Main Program**



# CHAPTER 4 - System Components Tests

## 4.1 Laser Line Scanner

### *4.1.1 Rotating Mirror Control*

The laser line scanner swept laser beam at a selected speed as the mirror rotated by a DC motor. At a given sampling rate, a constant mirror rotational speed gave the laser line scanner a constant number of measurements per revolution. If the speed varied, the number of measurements per revolution would change, resulting in changes in horizontal sample spacing. Thus, the rotational speed of the DC motor had a great influence on the measurement resolution of the laser system. The basic requirement for a DC motor was to rotate at the desired speed with minimum steady-state errors. The other requirement was that the motor should accelerate to its steady-state speed within a relatively short period of time after it was turned on.

A test was designed to evaluate the performance of the DC motor on the laser line scanner in controlling the rotating mirror.

The DC motor speed was controlled by the output voltage level of the motor power control on the HSIF card, which could be varied as commanded. The power control resolution was 1 part in 256 and the power could be set from 0 (off) to 255 (full power). The resolution of the optical encoder mounted on the motor shaft was 4096 pulse counts per revolution. The angular displacement of the motor could be monitored by counting the number of pulse detected within a period of time. Therefore, the motor speed (RPM) can be derived from Equation 4.1.

$$S_M = \frac{60000 \times N_{pulse}}{4096 \times T} \qquad (4.1)$$

where $S_M$ is the speed of motor (RPM),

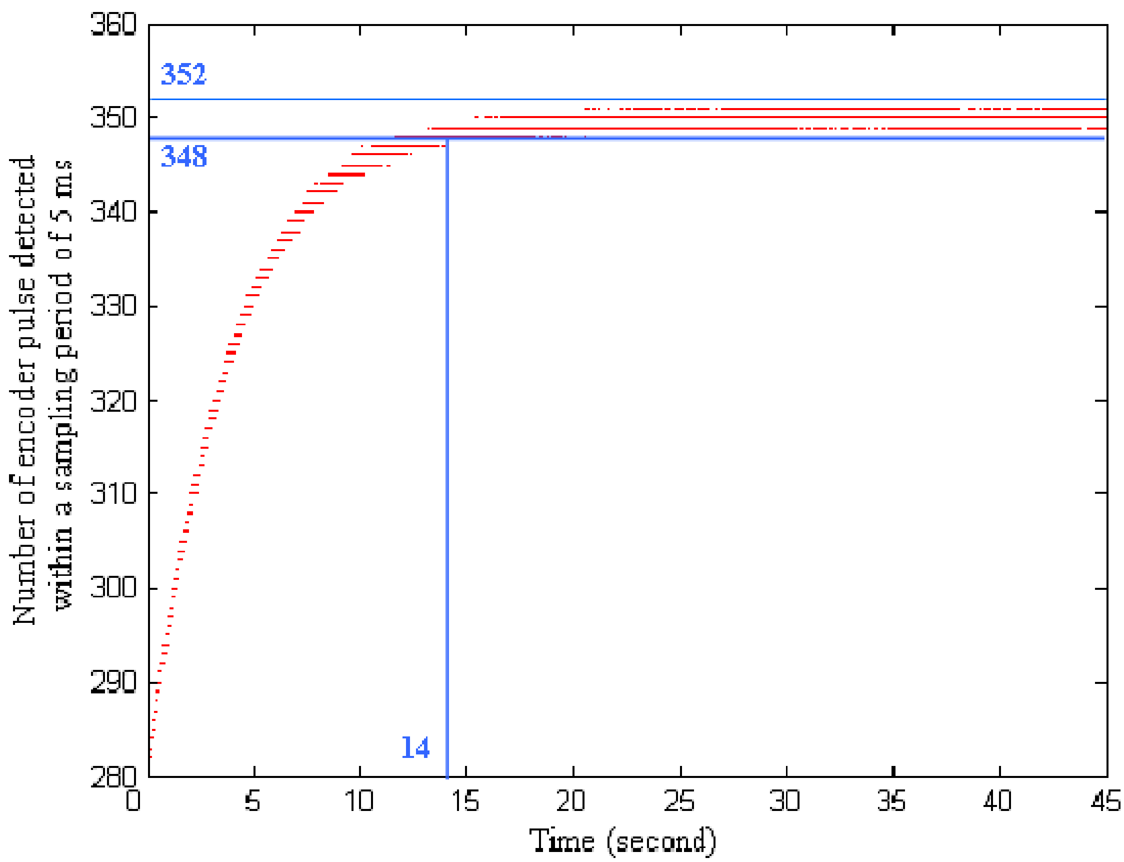
$N_{pulse}$ is the number of encoder pulse detected within a sampling period, and

T is the sampling period (ms).

The settling time is the duration for the $N_{pulse}$ to reach a narrow envelope of ±2 pulses/sampling-period surrounding the steady-state value.

In this test, the laser sampling period was set at 5 ms, corresponding to a sampling rate of 200 Hz. The DC motor was running at full power. The $N_{pulse}$ readings of the encoder were recorded for 45 seconds after the DC motor was powered on. The $N_{pulse}$ recorded during the first 45 seconds is illustrated in Figure 4.1.

**Figure 4.1 Number of Encoder Pulses Detected within a Sampling Period after the Motor was Powered on**




This figure shows that $N_{pulse}$ reached within the narrow envelope of 350±2 pulses/sampling-period 14 seconds after the motor was powered on. After the steady-state speed was reached, $N_{pulse}$ was maintained at 350±1 pulses/sampling-period. Thus, the laser scanner should not start measurement until the transient period was completed. In this test, a delay time of 20 seconds was applied to all data collection programs.

### *4.1.2 Distance Calibration*

The AR4000 rangefinder on the laser line scanner required sufficient light reflection from the target surface. The amount of reflected light, expressed as a percentage of the incident light, was defined as reflectance, which depended on target color and composition, and the frequency of the light being reflected (SMSI, 2009).

Target reflectance was an important factor affecting the accuracy of distance measurement. There were two types of targets: cooperative targets and uncooperative targets. A cooperative target was designed to reflect a major portion of the incident light. Light colored materials, such as wood, paper, and white paint, were the best cooperative targets. They may be measured over a wide range of incident angles. In some applications, mirrors may be used as cooperative targets. Uncooperative targets included surfaces or materials not specifically designed to reflect light, such as shiny metal, painted surfaces, liquids, and loose or granular solid substances. The AR4000 rangefinder was designed to work with both cooperative and uncooperative targets. In addition to the amount of light a surface reflects, the way in which the light hit the target surface could affect the sensor's performance. Temperature and ambient light may also affect the accuracy of the laser sensor (SMSI, 2009).

In the laboratory, a test was conducted to evaluate the performance of the laser sensor on difference targets and to calibrate the distance measurement. The targets tested were white paper, black paper, and sand. During the calibration, the target was positioned on a lifting platform, at seven different distances away from the center of the rotating mirror (Figure 4.2). The laser sensor took distance measurements at each position for 15-18 seconds. For each distance, an average number of 60,000 ~ 100,000 measurements was taken and the average readings were compared with the actual distances (Table 4.1).

**Figure 4.2 The Platform for the Laser Scanner and the Targets**

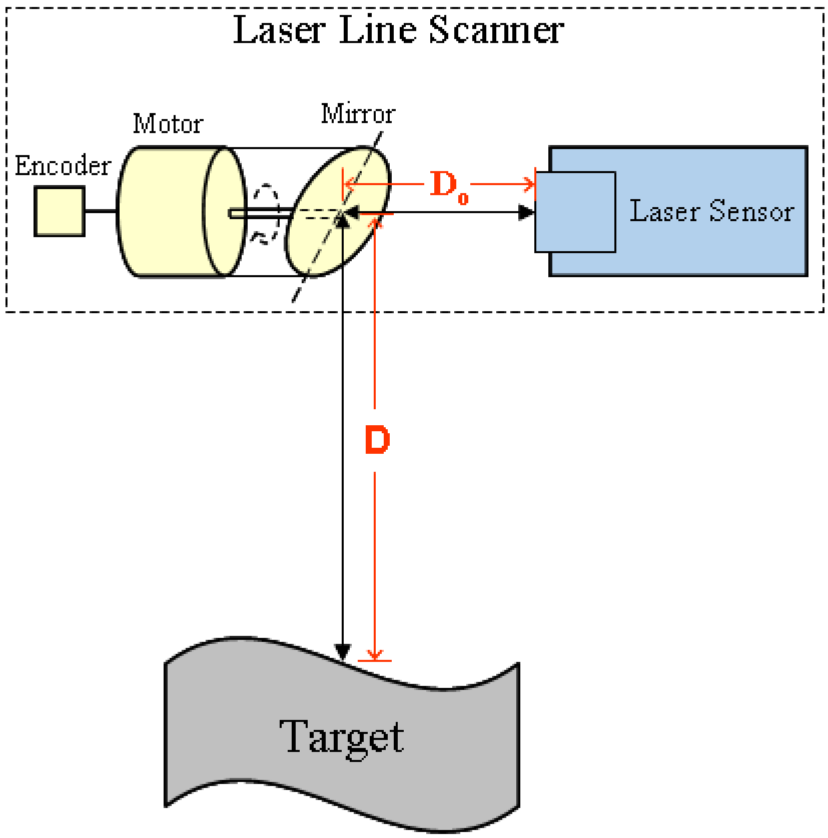
**Table 4.1 Calibration of the Laser Distance Sensor**

| **Actual distance** (cm) | **Measured distance** (cm) | | |
|---|---|---|---|
| | **white paper** | **black paper** | **sand (‡)** |
| 16.19 | 19.31 | 21.06 | 19.23 |
| 23.81 | 27.69 | 29.17 | 27.08 |
| 39.05 | 42.72 | 43.19 | 42.77 |
| 54.29 | 58.09 | 58.18 | 57.21 |
| 69.53 | 73.84 | 77.88 | 73.40 |
| 84.77 | 87.92 | 87.61 | 87.25 |
| 100.01 | 103.94 | 104.65 | 103.06 |

‡: 2.7 cm was added to the measured distances because sand surface in the box was 2.7 cm higher than other targets.

The actual distance (D) was measured from the center of the rotating mirror to the target, whereas the distance measured by the laser sensor was the actual distance plus $D_o$ (Figure 4.3). $D_o$ was the distance from the faceplate of the laser sensor to the center of the rotating mirror, which was not provided by the manufacturer.

**Figure 4.3 The Actual and Measured Distances**




The relationships between the actual distance and the measured distance for three targets are showed in Figures 4.4 through 4.6.

**Figure 4.4 Laser Distance Measurement with White Paper as the Target**

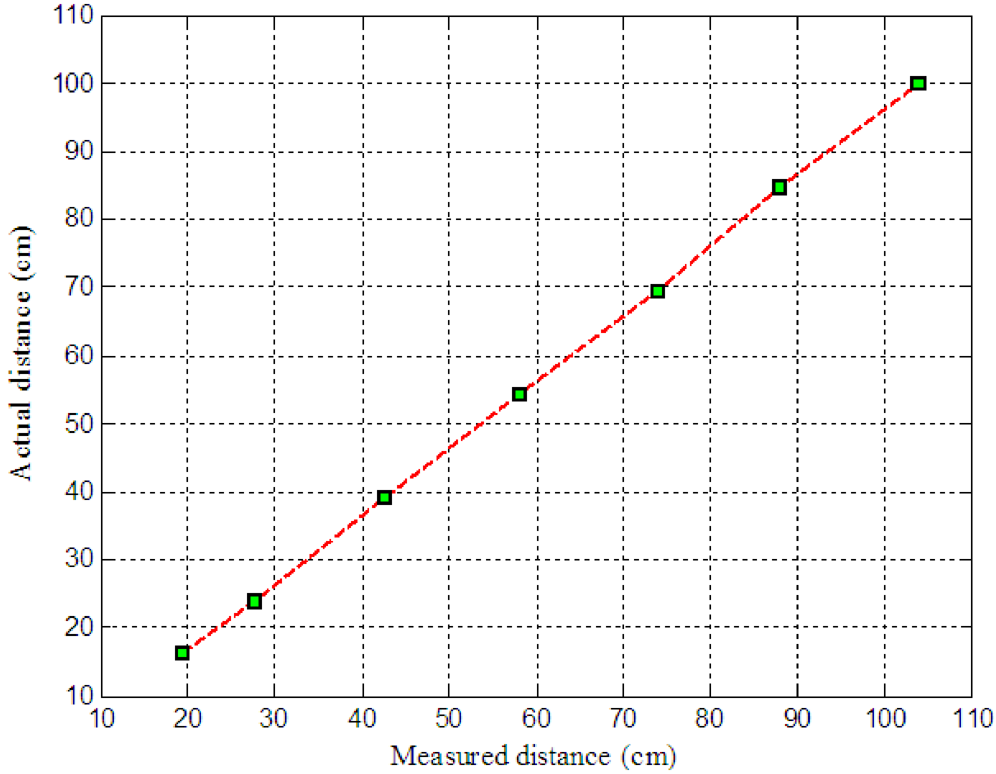



**Figure 4.5 Laser Distance Measurement with Black Paper as the Target**

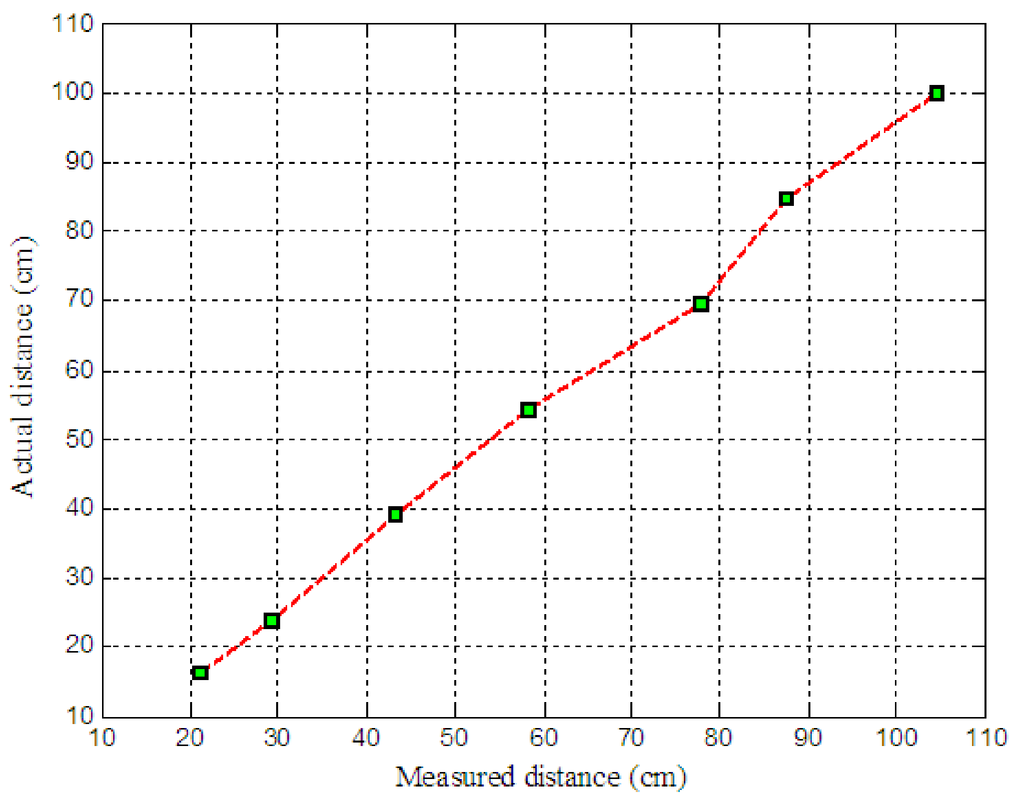
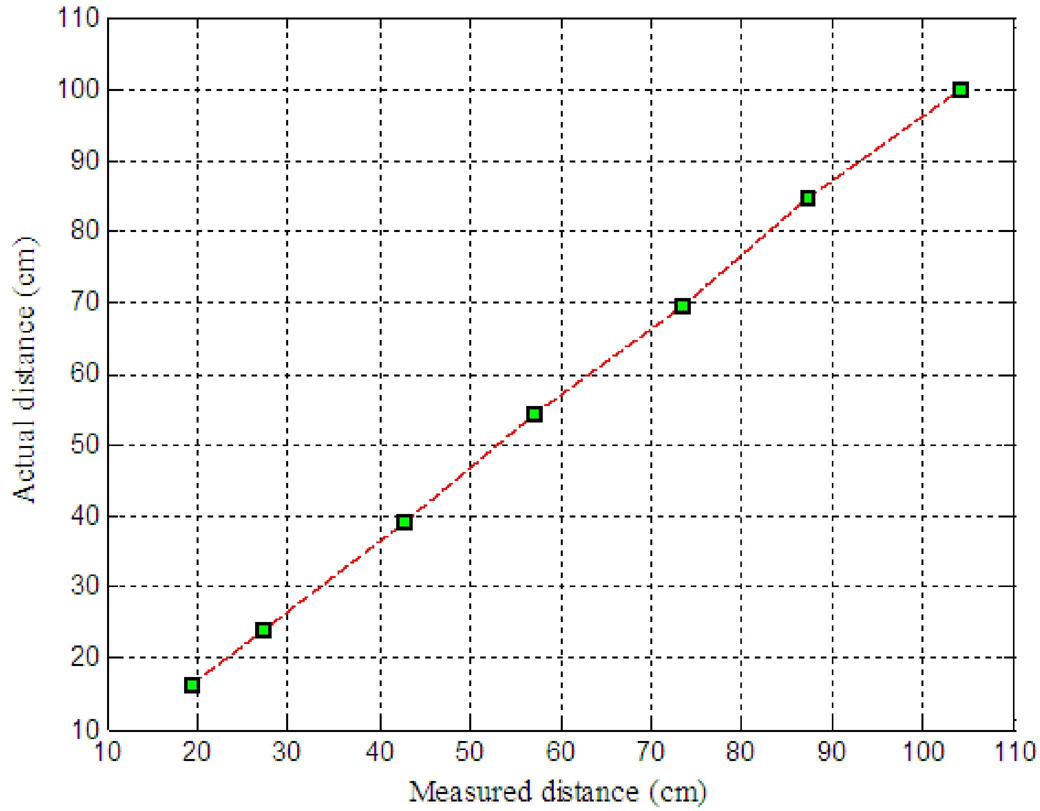


**Figure 4.6 Laser Distance Measurement with Sand as the Target**




To compare the accuracies in distance measurement on these targets, standard deviations of 60,000 ~ 100,000 repeatedly measured distances at each position for each target are plotted in Figures 4.7 through 4.9. It is obvious that the laser measurement on the white paper had the least variability. To determine the value of $D_o$, the actual distance were calibrated against the laser readings through a linear regression (Equation 4.2) using the data acquired on the white paper. Apparently, the intercept value (3.4763) in the regression equation represented $D_o$. Figure 4.10 shows the regression curve and the residuals. In the remaining sections of the dissertation, the calibrated laser distance reading, $D_C$, will be referred to as "raw laser data".

$$D_C = 0.99633 \times D_M - 3.4763 \qquad (4.2)$$

where $D_C$ is the measured distance (cm), and

$D_M$ is the distance reading of the laser sensor (cm).

**Figure 4.7 Standard Deviations of 60,000 ~ 100,000 Distance Measurements Taken at Seven Distances. The Target Material was White Paper.**

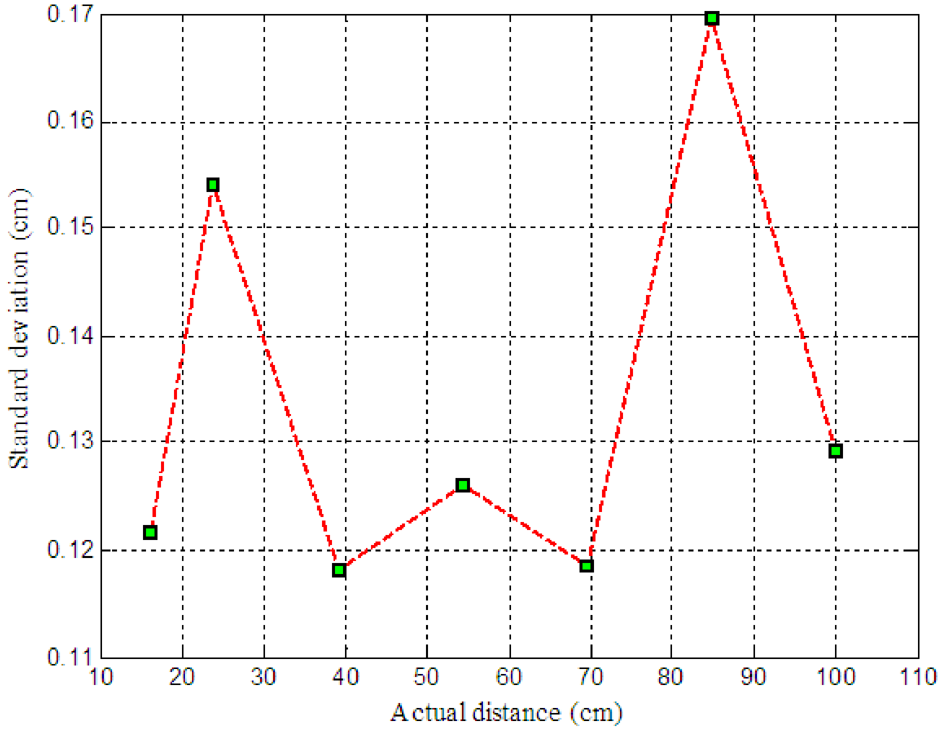



**Figure 4.8 Standard Deviations of 60,000 ~ 100,000 Distance Measurements Taken at Seven Distances. The Target Material was Black Paper.**

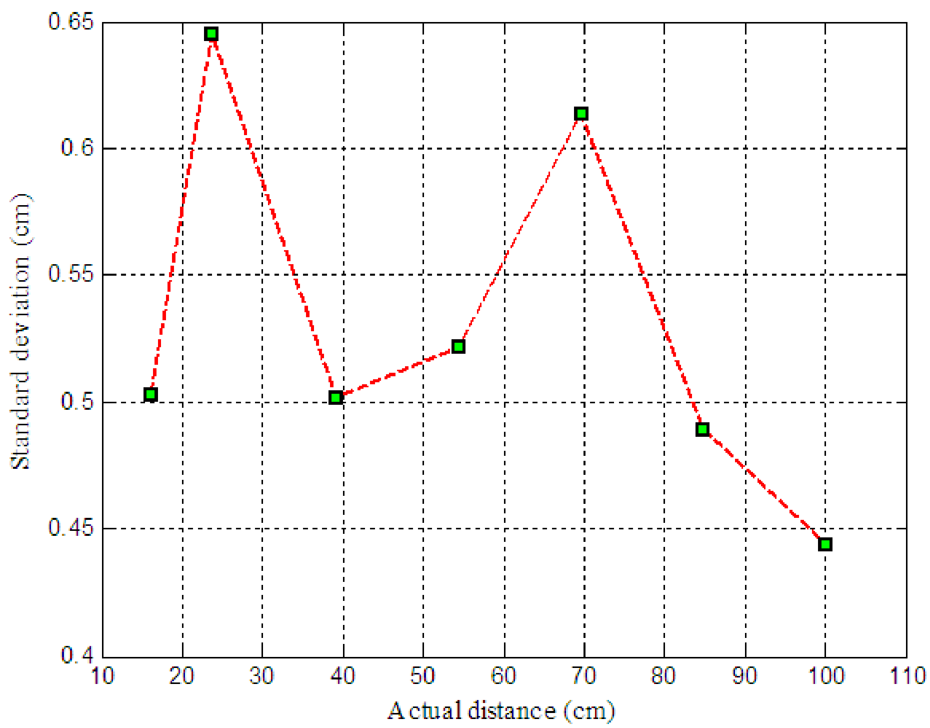


**Figure 4.9 Standard Deviations of 60,000 ~ 100,000 Distance Measurements Taken at Seven Distances. The Target Material was Sand Surface**

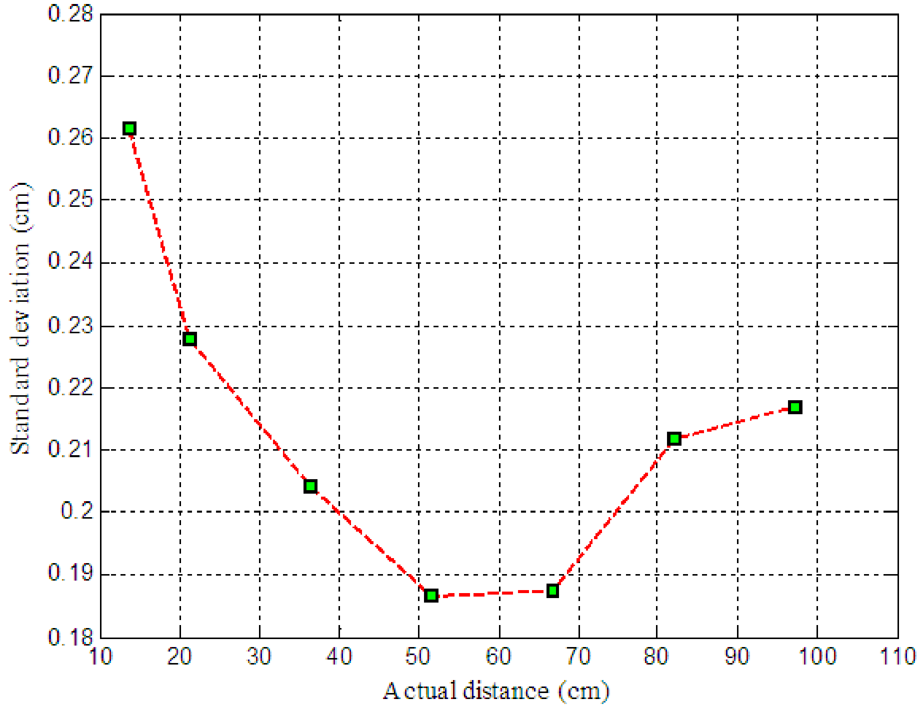
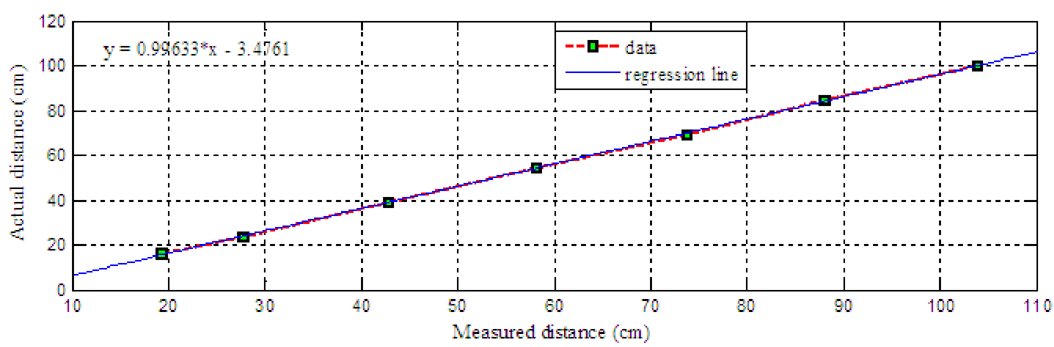
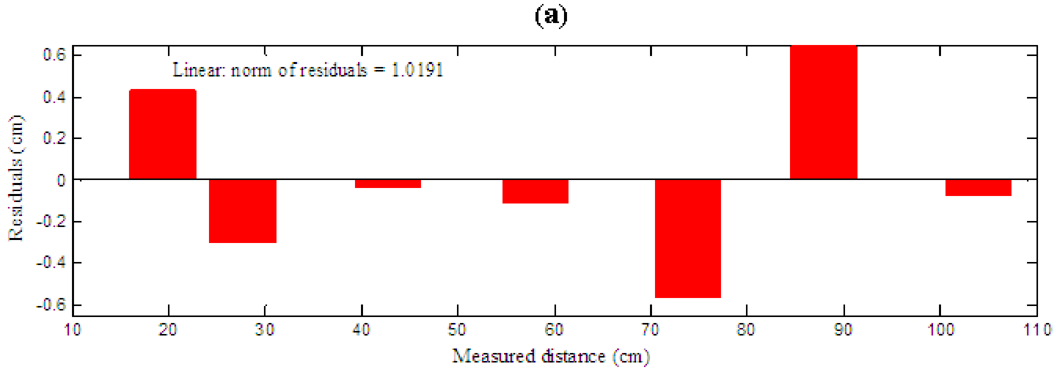



**Figure 4.10 Calibration for Distance Measurement: (a) Regression Line, and (b) Residual Plot**




(a)




(b)

### *4.1.3 Measurement Resolutions of the Laser System*

The resolution of the laser system was defined in two parts: the resolution in elevation measurement (Z) and the resolutions in sample grids (X and Y).

The resolution in elevation measurement was highly dependent on the resolution of the laser line scanner, which was limited by the sample rate selected and the maximum range to be measured. Using the HSIF, the resolution can be 33% finer than that using the serial port (Acuity Research Inc., 2000). Table 4.2 shows sampling rates required at different combinations of maximum distance settings and attainable resolutions.

**Table 4.2 Sampling Rates (sample/second) required for Different Combinations of Maximum Distance Settings and Attainable Resolutions (from Acuity Research Inc., 2000)**

| **Resolution** (cm) | **Maximum distance** (m) | | |
|---|---|---|---|
| | **1.83** | **9.14** | **15.24** |
| 0.0119 | 2304 | 677 | 390 |
| 0.0239 | 4609 | 1355 | 781 |
| 0.0478 | 9218 | 2711 | 1562 |
| 0.0952 | 18346 | 5422 | 3125 |
| 0.1905 | 36873 | 10845 | 6250 |
| 0.3810 | 50000 | 21691 | 12500 |
| 0.7620 | 50000 | 43382 | 25000 |
| 1.5240 | 50000 | 50000 | 50000 |

The horizontal spatial resolution was affected by the rotational speed of the mirror, the sampling rate, the movement speed of the laser-carriage, the height of the laser line scanner from the ground, and the size of the scan area. Assuming that, X axis was in the direction along the laser scan line, Y axis was in the direction along the rail, the size of the scan area was D by W, and the height of the laser line scanner was H (Figure 4.11), the resolutions in the X and Y directions, $\Delta x$ and $\Delta y$, can be derived using Equations 4.3 and 4.5, respectively.

$$\Delta x = \frac{W}{2} - tan(tan^{-1}(\frac{W/2}{H}) - \alpha) \times H \qquad (4.3)$$

where $\alpha$ = angle travelled by the laser beam during a sampling period (degree), and

$$\alpha = \frac{S_M}{f_s} \times 360^\circ, \qquad (4.4)$$

$S_M$ = the rotational speed of the mirror (RPM),

$f_s$= the sampling rate (Hz),

W = the width of the scan area (meter), and

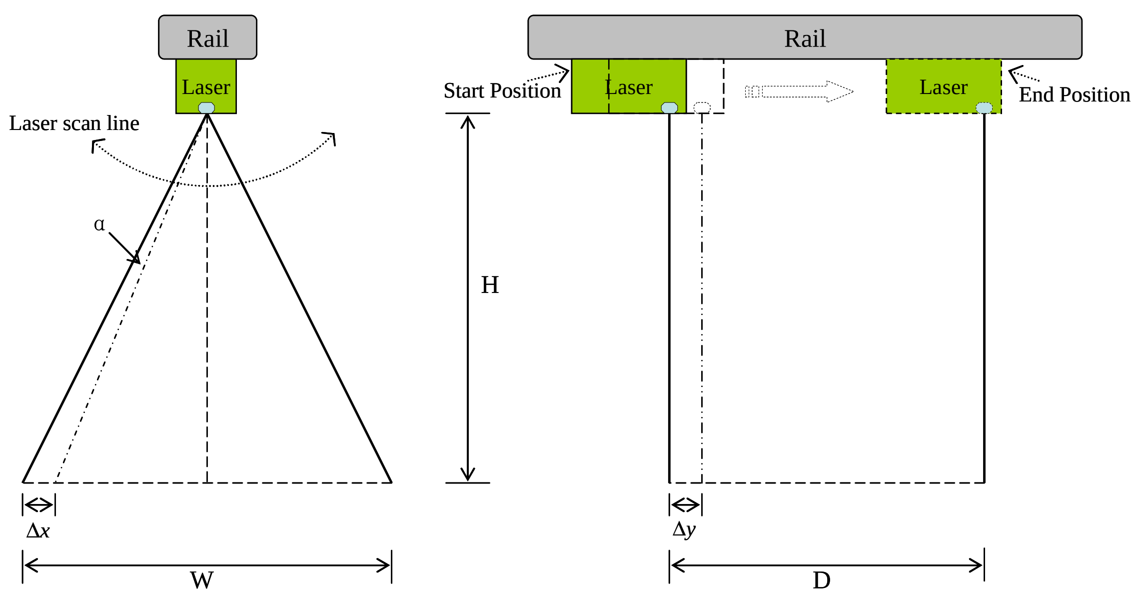
H= the height of the laser line scanner from the surface (meter).

$$\Delta y = \frac{S_C}{S_M} \qquad (4.5)$$

where $S_C$ = the movement speed of the laser-carriage (m/s).

Obviously, $\Delta x$ was not constant along the X axis. The largest $\Delta x$ occurring near the edge of the scan area would dominate the resolution of X axis. The resolutions of sample grids can be changed by adjusting variables in Equations 4.3, 4.4, and 4.5.

**Figure 4.11 Horizontal Spatial Resolutions**



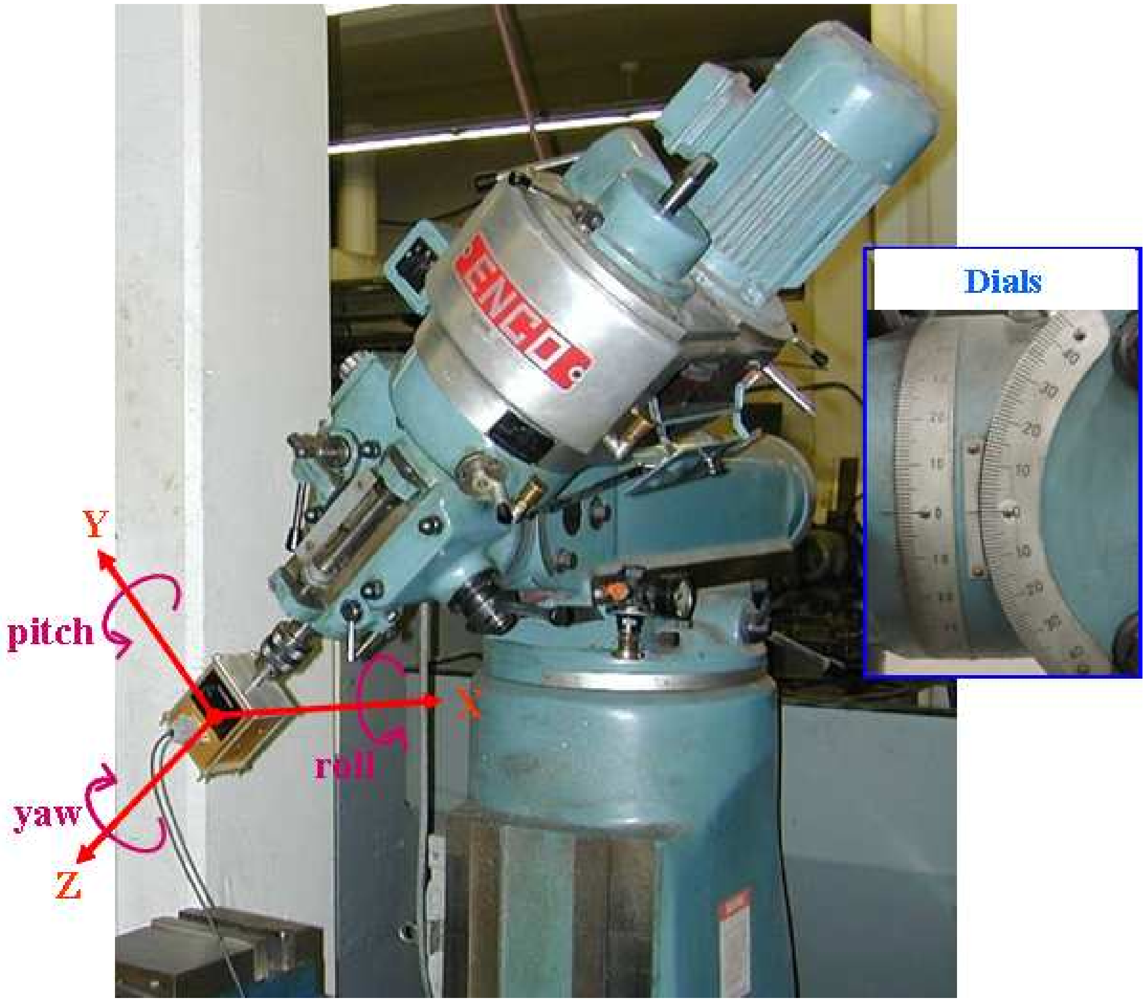
## 4.2 Gyroscope Sensor

### *4.2.1 Static Accuracy on a Two-dimensional Platform*

The attitude information of the linear rail detected by the gyroscope sensor was used to correct the elevation data. The gyroscope sensor should provide reliable and accurate angle measurements, especially for roll and pitch angles. A series of tests was conducted to examine the measurement errors of the gyroscope sensor in measuring pitch, roll, and combined pitch/roll rotations.

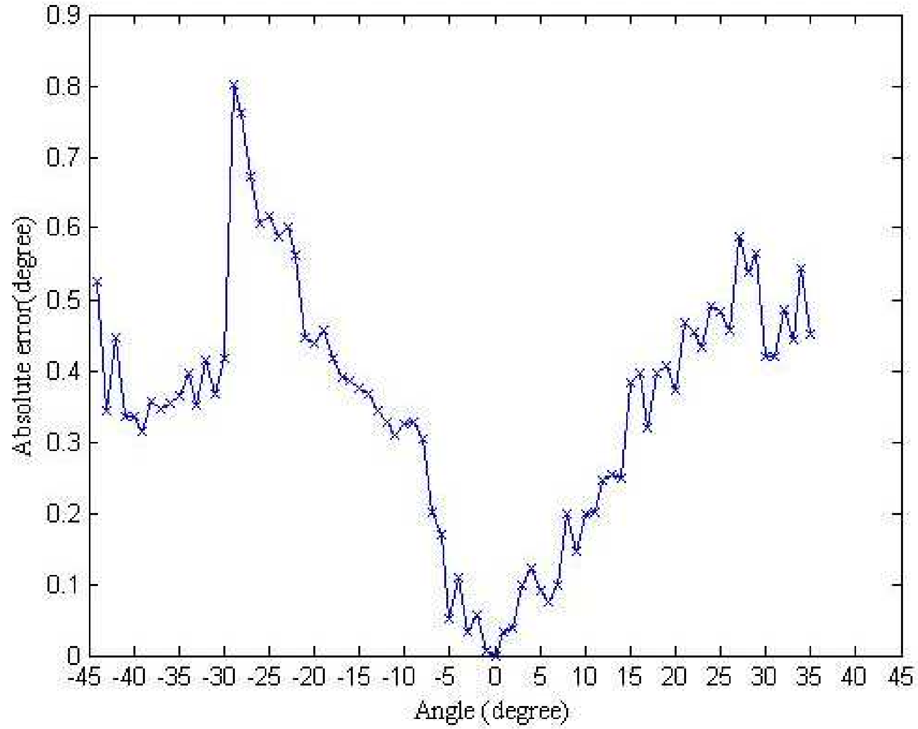
A milling machine (Enco Manufacturing Co.) was used as a two-dimensional testing platform. With the gyroscope sensor fastened on the drill head, the machine can provide three-dimensional rotations to the gyroscope sensor (Figure 4.12). The pitch and roll angles were defined as the angular displacements around the Y and X axes, respectively. The milling machine could provide maximum roll and pitch angles of ±90°and ±45° with a 1° increment, respectively. For the static test, the gyroscope was tested with pitch rotation only, roll rotation only, and combined pitch and roll rotations. Considering the application of the gyroscope sensor on the laser system, pith and roll rotations of up to ±45° were tested. The gyroscope sensor was also calibrated for hard and soft iron compensation after being mounted on the milling machine. At each angle, the average value of 4,000 measurements was compared with the actual angle read from the dials of the milling machine. The absolute error between the average measurement value and the actual angle was used to evaluate the accuracy of the measurement.

**Figure 4.12 Gyroscope Sensor Installed on the Milling Machine**

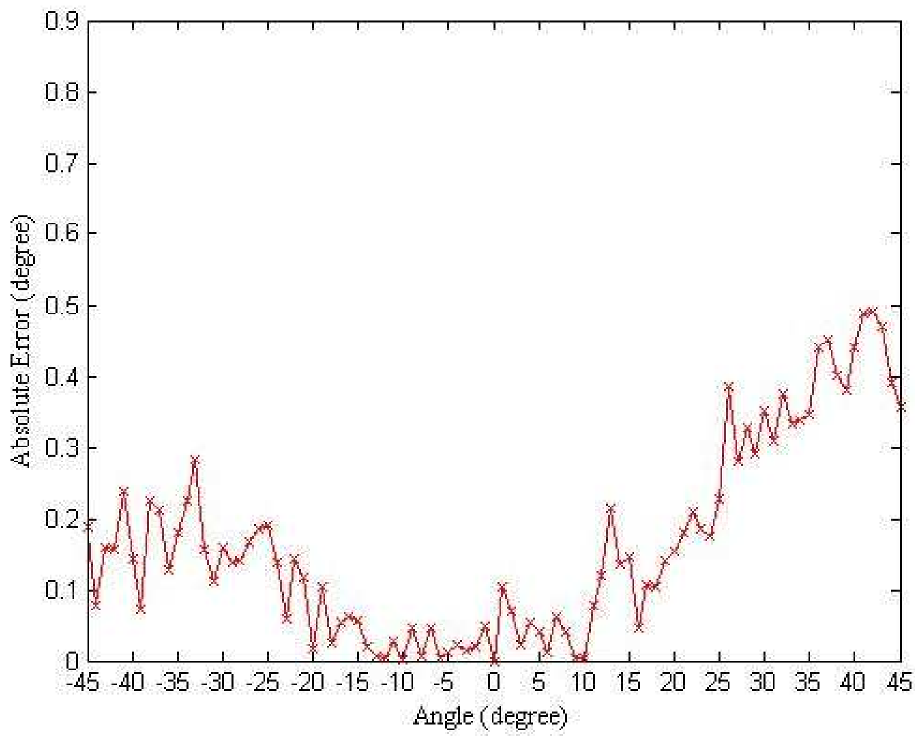



The pitch and roll angle measurement errors are shown in Figure 4.13. For both pitch and roll angles, the measurement errors generally increased as the angles increased. The maximum absolute error for pitch was 0.80° when the pitch angle was -28°. The maximum absolute error for roll was 0.49° when the roll angle was 42°. Within the ±30° range, the measurement errors of pitch and roll angles were within 0.8° and 0.4°, respectively.

**Figure 4.13 Absolute Errors for Pitch and Roll Measurement Observed in Single Rotations: (a) Pitch Rotation, and (b) Roll Rotation**



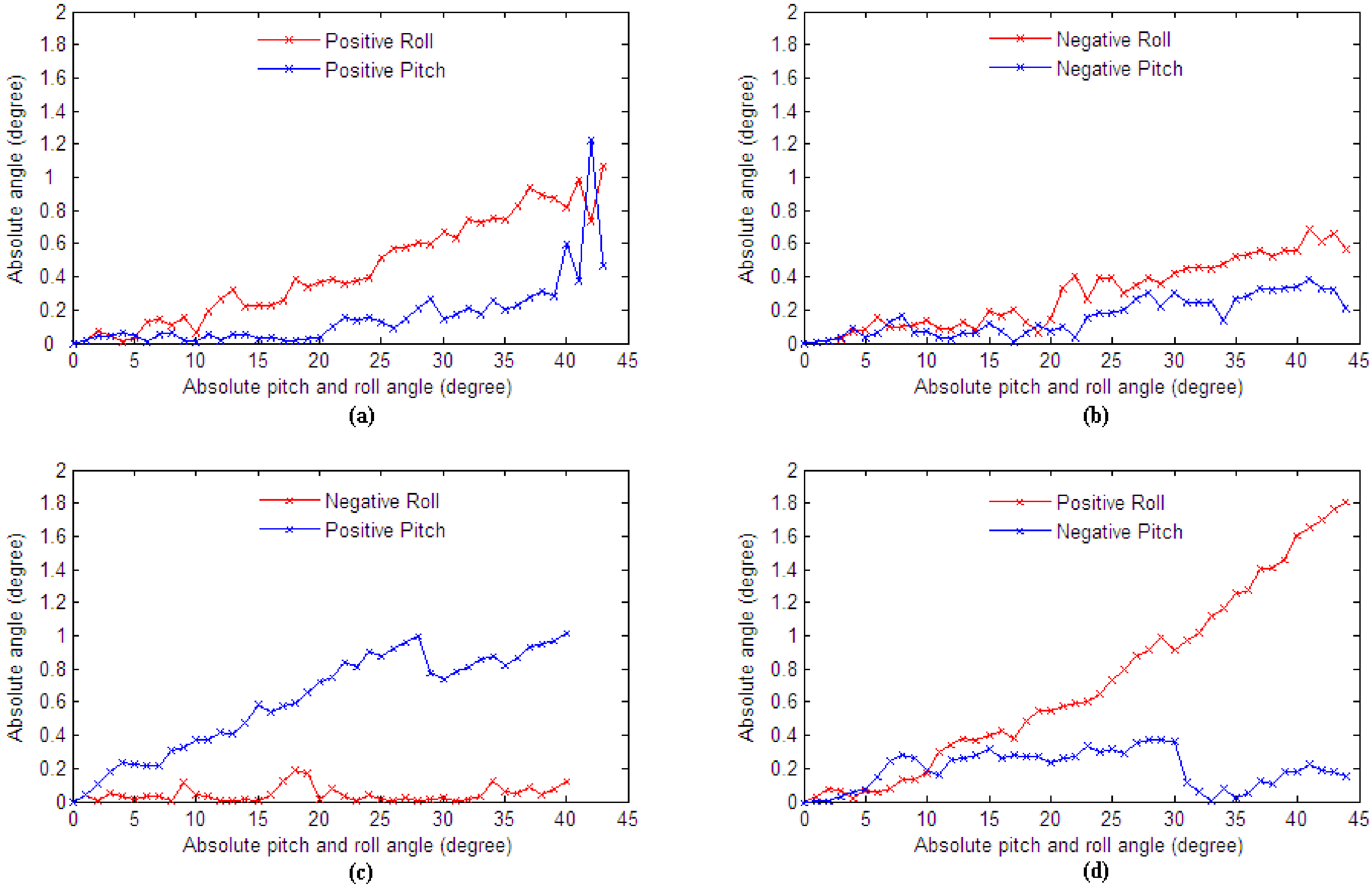

**(a)**




**(b)**

Figure 4.14 shows the measurement errors for the pitch and roll angles when both rotations existed. Four cases were considered: (1) positive pitch and roll, (2) negative pitch and roll, (3) positive pitch with negative roll, and (4) negative pitch with positive roll. For case 1, maximum absolute errors of 1.23° and 1.07° occurred at 42° and 43° for pitch and roll, respectively (Figure 4.14a). The maximum absolute errors observed for case 2 were 0.38° at pitch angle of -41° and 0.68° at roll angle of -41° (Figure 4.14b). For case 3, the maximum absolute errors were 0.99° at pitch angle of 28° and 0.19° at roll angle of -18° (Figure 4.14c). The maximum absolute errors observed for case 4 were 0.37° at pitch angle of -28° and 1.81° at roll angle of 44° (Figure 4.14d). In summary, when pitch or roll angle varied within ±30°, the measurement errors for both pitch and roll angles were maintained within 1°.

**Figure 4.14 Absolute Errors in Pitch and Roll Measurements Observed in Four Cases: (a) Positive Pitch and Roll, (b) Negative Pitch and Roll, (c) Positive Pitch and Negative Roll, and (d) Negative Pitch and Positive Roll**



It can be noted that some of the errors observed in combined pitch/roll rotations were larger than those in single rotations. This may have been caused by the difference in magnetic effect of the steel milling machine under these conditions. Magnetic effect and angle measurement were always inter-linked in the gyroscope sensor (CTI, 2006) .

### *4.2.2 Effect of Attitude Measurement Error on Elevation Measurement*

The effect of pitch and roll measurement errors on elevation measurement was analyzed for a typical application of the laser system on a 1m × 1m soil surface with the laser line scanner mounted at 1m height from the ground. Four cases were analyzed to study the maximum allowable errors in roll and pitch angle measurements that would limit the elevation measurement error to 1 cm.

For this analysis, constants and variables were defined as follows (Figures 4.15, 4.19, 4.23, and 4.25):

$e$ - elevation error, which was limited to ±1 cm,

$\alpha$ - roll angle reading from the gyroscope sensor (degree), which was in the range of ±30°,

$\alpha_r$ - measurement error of roll angle (degree),

$\beta$ - pitch angle reading from the gyroscope sensor (degree), which was in the range of ±30°,

$\beta_r$ - measurement error of pitch angle (degree),

$h$ - elevation to be measured (meter),

$H_0$ - height of the laser line scanner from the reference ground when the rail was leveled (meter), which was equal to 1m,

$H$ - height of the laser line scanner from the ground when the rail was tilted (meter),

$L$ - distance from the laser line scanner to the surface to be measured (meter), which was in the range of 0 to 1m,

$W_R$ - distance between the laser beam and the outer edge of the vehicle's right wheel, which was equal to 0.5 m,

$W_L$ - distance between the laser beam and the outer edge of the vehicle's left wheel, which was equal to 1.5 m,

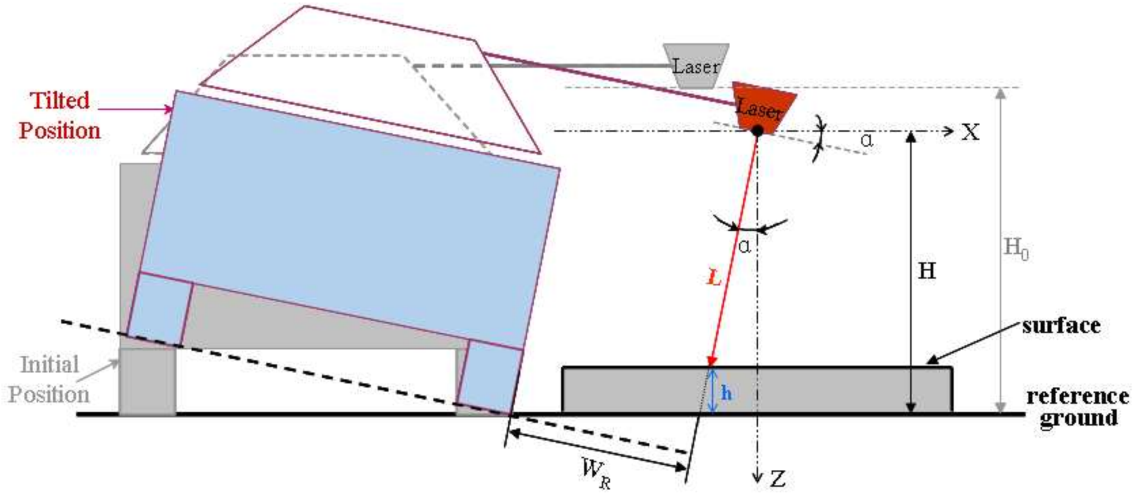
$D_R$ - distance between the laser beam and the center of the vehicle's rear wheel, which was equal to 1 m, and

$D_F$ – distance between the laser beam and the center of the vehicle's front wheel, which was equal to 1 m.

This analysis was implemented in MATLAB (Appendix D).

Case A: The vehicle is tilted to the right (Figure 4.15). A positive roll angle would be measured by the gyroscope sensor.

**Figure 4.15 Elevation Measurement When the Vehicle is Rolled Right**




Based on the geometrical relationship, the height (*H*) and the elevation (*h*) can be derived from Equations 4.6 and 4.7.

$$H = (H_0 - W_R \cdot \tan\alpha) \cdot \cos\alpha \qquad (4.6)$$

$$h = H - L \cdot \cos\alpha \qquad (4.7)$$

With the measurement error ($a_r$) in the roll angle measurement, the height of the laser line scanner from the ground would become:

$$H_r = [H_0 - W_R \cdot \tan(\alpha + \alpha_r)] \cdot \cos(\alpha + \alpha_r) \qquad (4.8)$$

and the measured elevation would become:

$$h_r = H_r - L \cdot \cos(\alpha + a_r) \qquad (4.9)$$

Thus, the error in the elevation measurement caused by the measurement error of roll angle can be calculated as:

$$e = \left| h_r - h \right| \tag{4.10}$$

And it can be calculated as:

$$e = \left| (H_0 - L) \cdot \cos(\alpha + \alpha_r) - W_R \cdot \sin(\alpha + \alpha_r) - (H_0 - L) \cdot \cos\alpha + W_R \cdot \sin\alpha \right| \tag{4.11}$$

For the maximum allowable measurement error in elevation, 1cm, the maximum allowable errors in the roll angle measurement, $a_r$, became a function of $L$ and $a$ , and can be analyzed in two conditions.

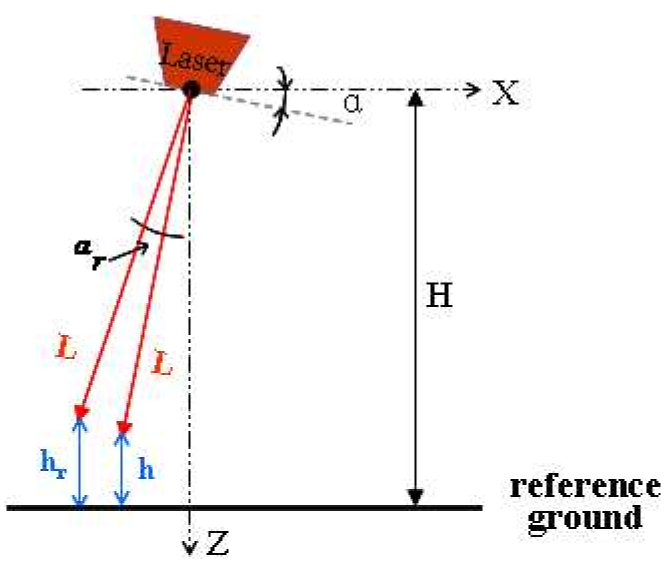
Condition 1 (Figure 4.16): when $e = 0.01$,

$$\alpha_r = - \arcsin\left(\frac{0.01}{\sqrt{(H_0 - L)^2 + W_R^2}} + \sin(\theta - \alpha)\right) + \theta - \alpha \tag{4.12}$$

where $\theta = \arccos\left(\frac{W_R}{\sqrt{(H_0 - L)^2 + W_R^2}}\right)$ (4.13)

Obviously, when $|\alpha_r| > |\alpha|$ , the actual roll angle became negative. Under this condition, the above analysis became invalid.

**Figure 4.16 Maximum allowable errors in positive roll angle measurement when the maximum allowable measurement error in elevation was 1cm**

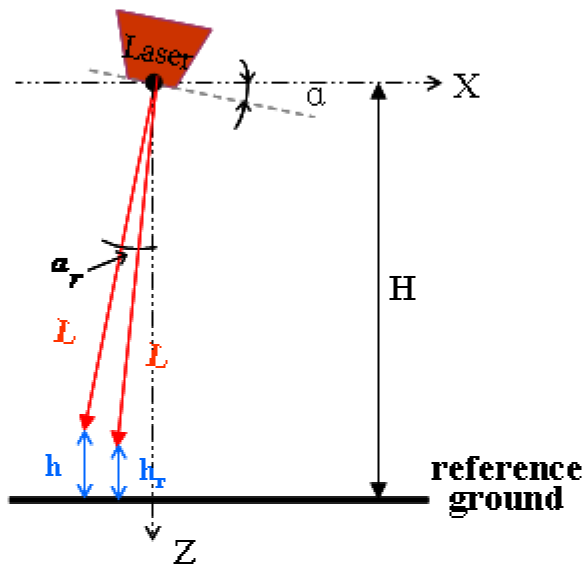


Condition 2 (Figure 4.17): when $e$ = -0.01,

$$\alpha_r = -\arcsin\left(\frac{-0.01}{\sqrt{(H_0 - L)^2 + W_R^2}} + \sin(\theta - \alpha)\right) + \theta - \alpha \qquad (4.14)$$

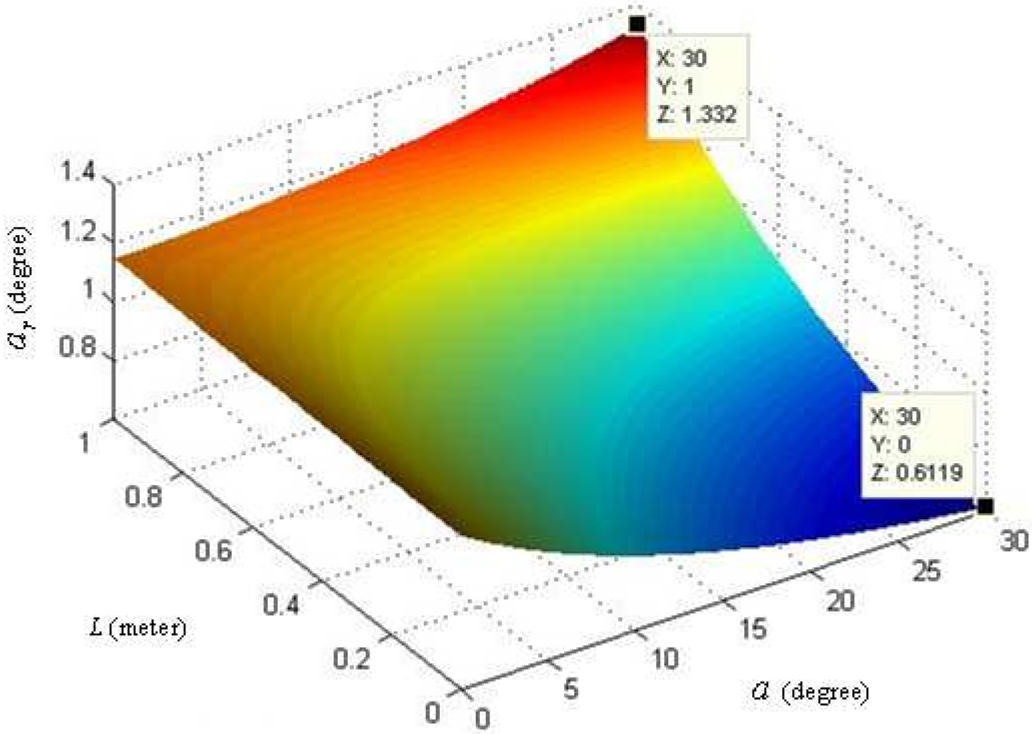
where $$\theta = \arccos\left(\frac{W_R}{\sqrt{(H_0 - L)^2 + W_R^2}}\right) \qquad (4.15)$$

**Figure 4.17 Maximum allowable errors in positive roll angle measurement when the maximum allowable measurement error in elevation was -1cm**




Among the two conditions, condition 2 gave a smaller $|\alpha_r|$ value. Thus, condition 2 was considered more critical. A three-dimensional model for $a_r$ established for condition 2 is shown in Figure 4.18.

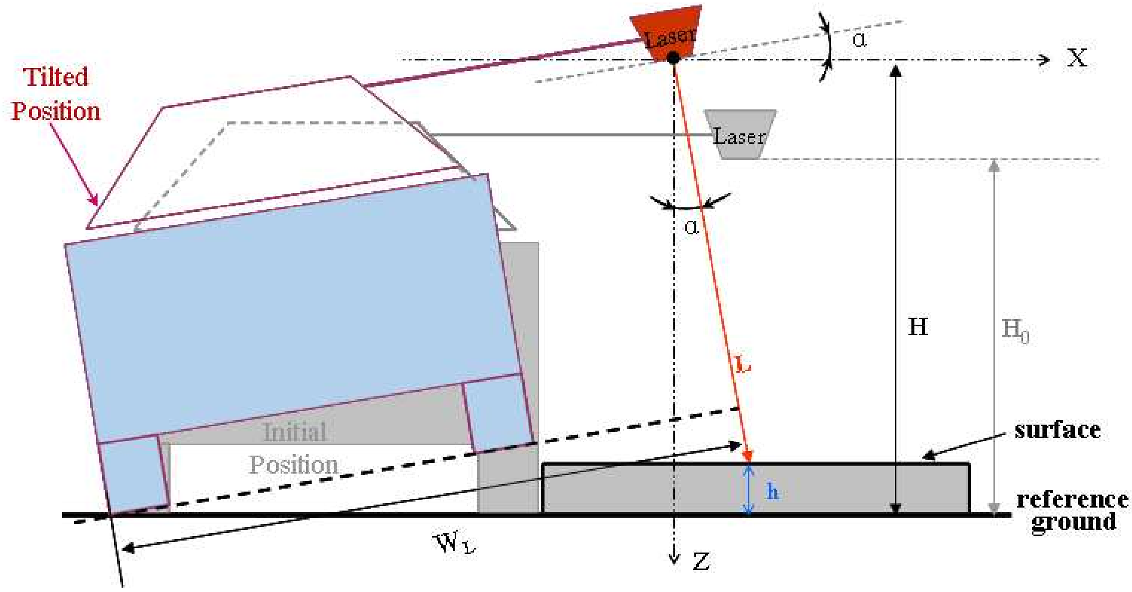
**Figure 4.18 The Maximum Allowable Error for Roll Angle Measurement to Limit the Elevation Measurement Error within ±1 cm when the Vehicle was Rolled Right**




The maximum allowable error of the roll angle measurement reached its maximum value, 1.332°, when the measured distance (*L*) was 1m and the vehicle was rolled to 30°. It reached the minimum value, 0.6119°, when the measured distance (*L*) was 0 m and the vehicle was rolled to 30° (Figure 4.18). Recalling that the measurement error of the gyroscope sensor in roll angle measurement was less than 0.4° when the roll angle was within 30° (Figure 4.13b), it can be concluded that the measurement error for roll angle would not be a significant effect on the elevation measurement in case A.

Case B: The vehicle is tilted to the left (Figure 4.19). A negative roll angle would be measured by the gyroscope sensor.

**Figure 4.19 Elevation Measurement When the Vehicle is Rolled Left**




From Figure 4.19, *H* and *h* can be derived from Equations 4.16 and 4.17.

$$H = (H_0 + W_L \cdot \tan\alpha) \cdot \cos\alpha \quad (4.16)$$

$$h = H - L \cdot \cos\alpha \quad (4.17)$$

Taking the same process as in case A, *e* can be expressed by Equation 4.18.

$$e = \left|(H_0 - L) \cdot \cos(\alpha + \alpha_r) + W_L \cdot \sin(\alpha + \alpha_r) - (H_0 - L) \cdot \cos\alpha - W_L \cdot \sin\alpha\right| \quad (4.18)$$

The maximum allowable errors, $a_r$, can be analyzed in two conditions.

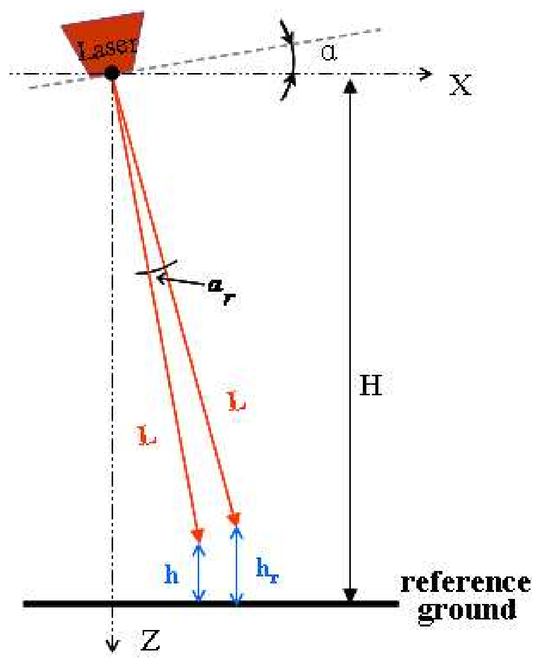
Condition 1 (Figure 4.20): when $e = 0.01$,

$$\alpha_r = arcsin(\frac{0.01}{\sqrt{(H_0 - L)^2 + W_L^2}} + sin(\theta + \alpha)) - \theta - \alpha \quad (4.19)$$

$$\text{where } \theta = arccos(\frac{W_L}{\sqrt{(H_0 - L)^2 + W_L^2}}) \quad (4.20)$$

Obviously, when $|\alpha_r| > |\alpha|$ , the actual roll angle became positive. Under this condition, the above analysis became invalid.

**Figure 4.20 Maximum allowable errors in negative roll angle measurement when the maximum allowable measurement error in elevation was 1cm**

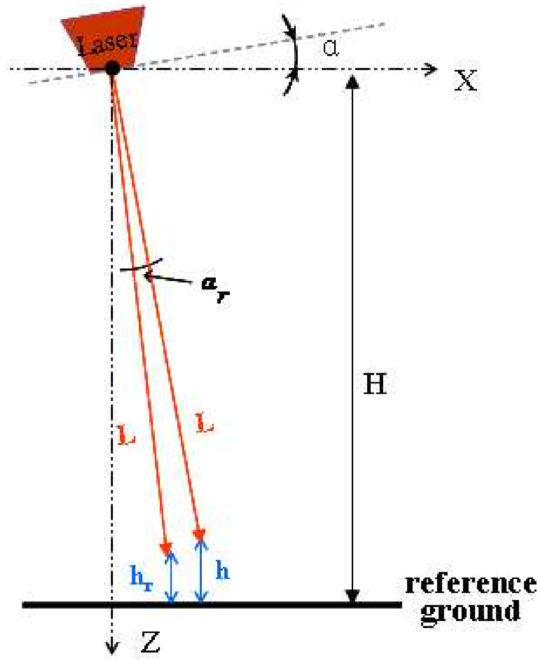



Condition 2 (Figure 4.21): when $e$ = -0.01,

$$\alpha_r = arcsin(\frac{-0.01}{\sqrt{(H_0 - L)^2 + W_L^2}} + sin(\theta + \alpha)) - \theta - \alpha \quad (4.21)$$

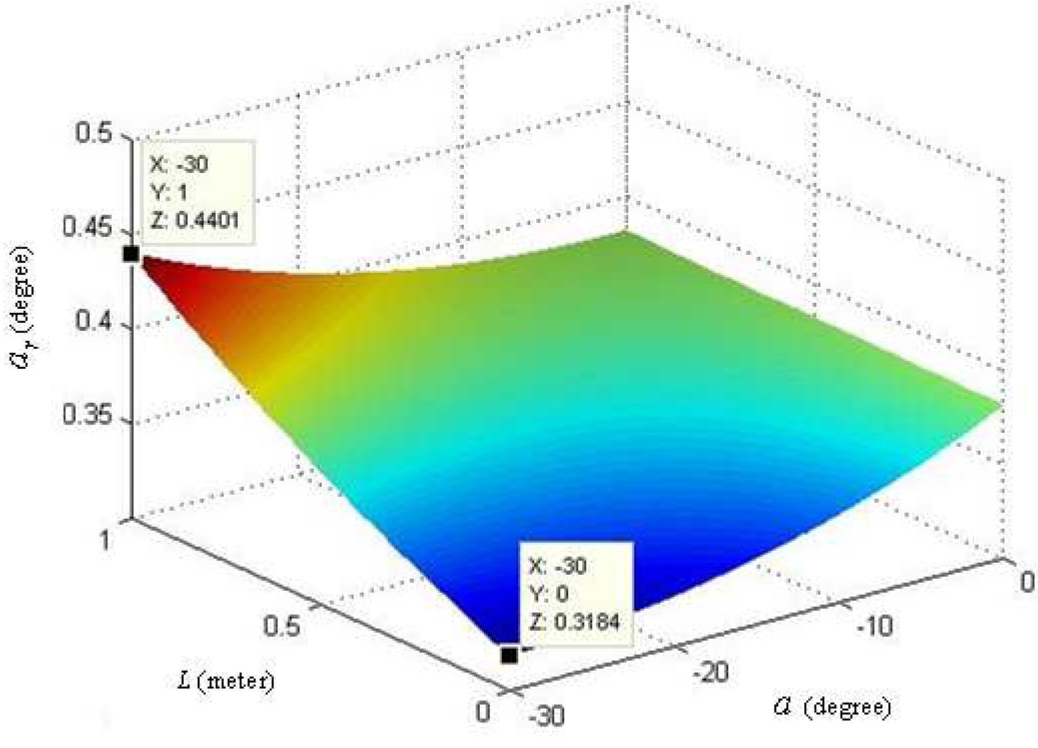
$$\text{where } \theta = arccos(\frac{W_L}{\sqrt{(H_0 - L)^2 + W_L^2}}) \quad (4.22)$$

**Figure 4.21 Maximum allowable errors in negative roll angle measurement when the maximum allowable measurement error in elevation was -1cm**




The values of $a_r$ in condition 2 were smaller then those in condition 1. Thus, condition 2 was considered more critical. A three-dimensional model for $a_r$ established for condition 2 is shown in Figure 4.22.

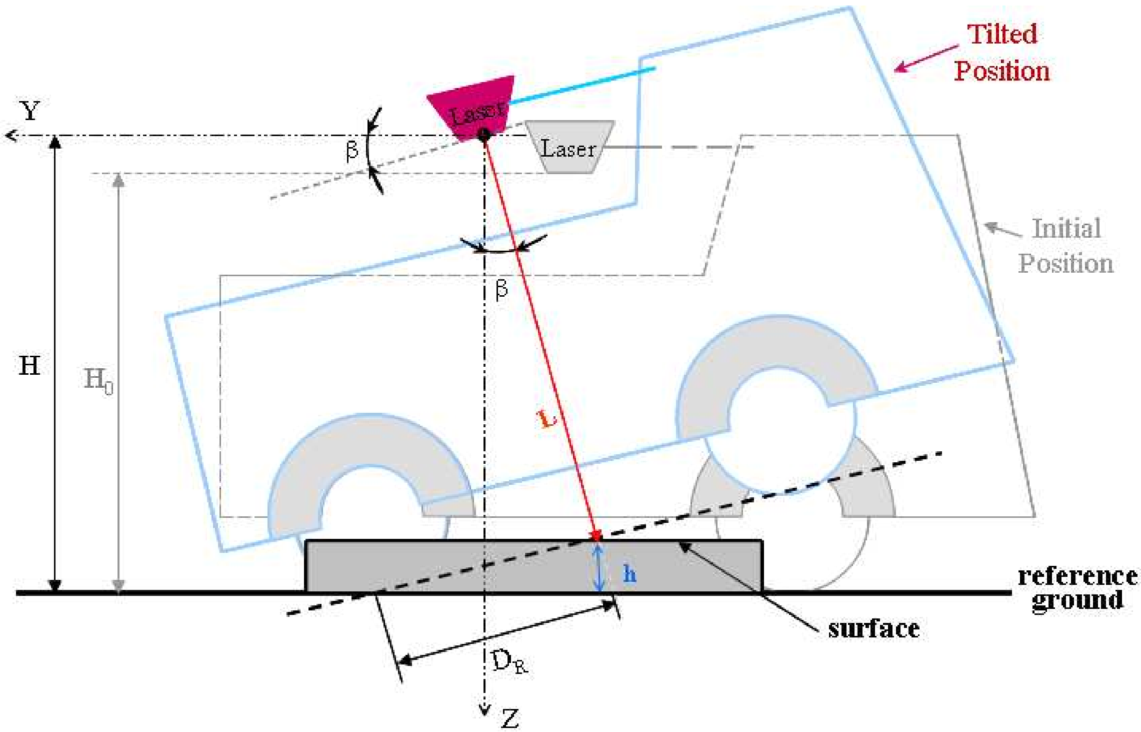
**Figure 4.22 The Maximum Allowable Error for Roll Angle Measurement to Limit the Elevation Measurement Error within ±1 cm when the Vehicle is Rolled Left**




The maximum allowable error for roll angle measurement reached its maximum value, 0.4401°, when the measured distance ($L$) was 1m and the vehicle was rolled to -30°. It reached the minimum value, 0.3184°, when the measured distance was 0 m and the vehicle was rolled to -30° (Figure 4.22). Recalling that the measurement error of the gyroscope sensor in roll angle measurement was less than 0.3° when the roll angle was within -30° (Figure 4.13b), it can be concluded that the measurement error for roll angle would not be a significant effect on the elevation measurement in case B.

Case C: The front end of the vehicle is tilted up (Figure 4.23). A positive pitch angle would be measured by the gyroscope sensor.

**Figure 4.23 Elevation Measurement When the Vehicle is Pitched Up**




From Figure 4.23, the height ($H$) and the elevation ($h$) can be derived from Equations 4.23 and 4.24.

$$H = (H_0 + D_R \cdot \tan\beta) \cdot \cos\beta \quad (4.23)$$

$$h = H - L \cdot \cos\beta \quad (4.24)$$

Taking the same process as in case A, *e* can be expressed by Equation 4.25

$$e = \left|(H_0 - L) \cdot \cos(\beta + \beta_r) + D_R \cdot \sin(\beta + \beta_r) - (H_0 - L) \cdot \cos\beta - D_R \cdot \sin\beta\right| \quad (4.25)$$

The maximum allowable errors, $\beta_r$, can be derived using Equations 4.26 and 4.28 under conditions 1 and 2, respectively. Because condition 2 gave smaller values, the results under condition 2 was chosen for further error analysis. A three-dimensional model for $\beta_r$ is shown in Figure 4.24.

Condition 1: when $e = 0.01$,

$$\beta_r = arcsin(\frac{0.01}{\sqrt{(H_0 - L)^2 + D_R^2}} + sin(\theta + \beta)) - \theta - \beta \quad (4.26)$$

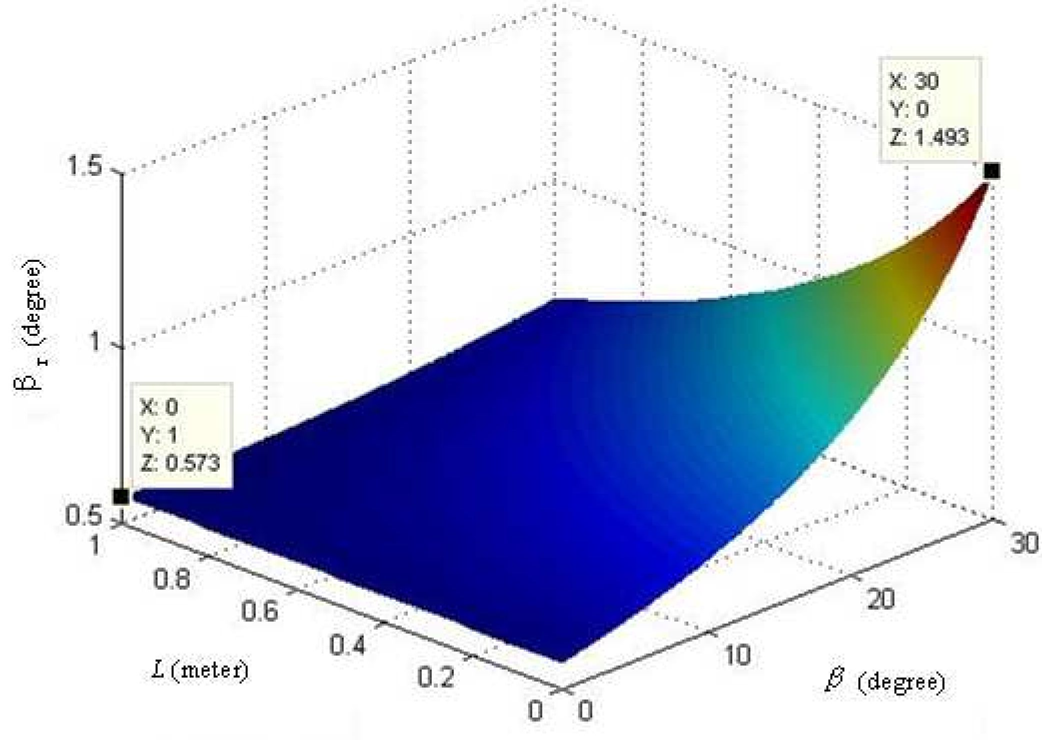
where $\theta = arccos(\frac{D_R}{\sqrt{(H_0 - L)^2 + D_R^2}})$. (4.27)

Condition 2: when $e = -0.01$,

$$\beta_r = arcsin(\frac{-0.01}{\sqrt{(H_0 - L)^2 + D_R^2}} + sin(\theta + \beta)) - \theta - \beta \quad (4.28)$$

where $\theta = arccos(\frac{D_R}{\sqrt{(H_0 - L)^2 + D_R^2}})$. (4.29)

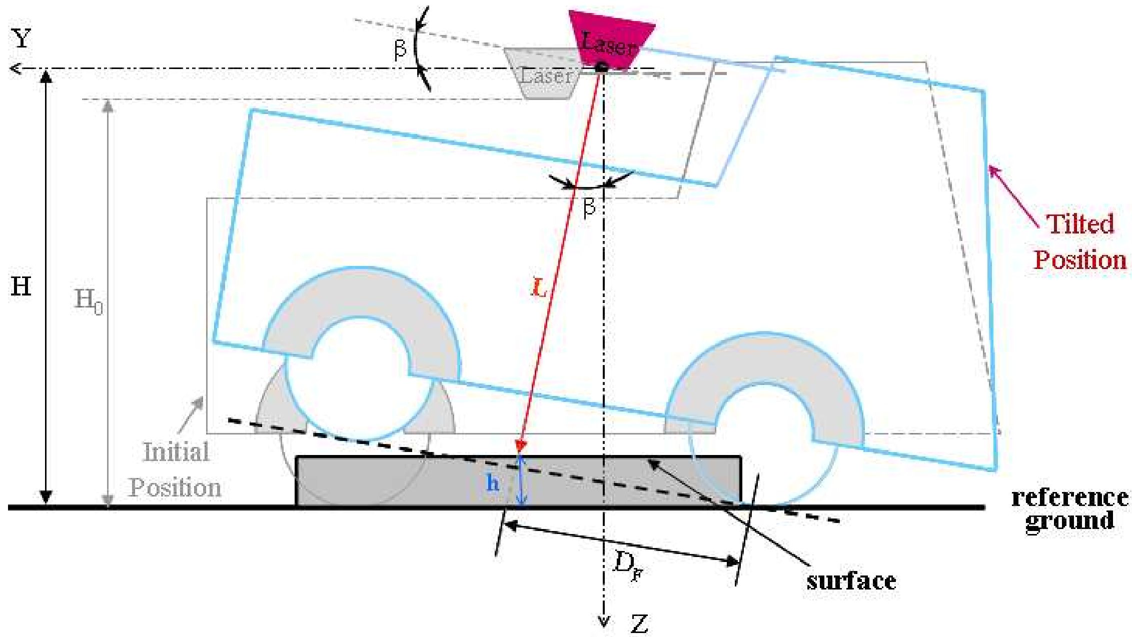
**Figure 4.24 The Maximum Allowable Error for Pitch Angle Measurement to Limit the Elevation Measurement Error within ±1 cm when the Vehicle is Pitched Up**




The maximum allowable error of the pitch angle measurement reached its maximum value, 1.493°, when the measured distance (*L*) was 0 m and the vehicle was pitched up to 30°. It reached the minimum value, 0.5730°, when the measured distance (*L*) was 1m and the vehicle was leveled (Figure 4.24). Recalling that, when the pitch angle was within ±30°, the maximum error of pitch measurement was 0.5871° (Figure 4.13a), it can be concluded that, for case C, the measurement error for pitch angle would have a significant effect on the elevation measurement, especially when the actural pitch angle approached 30°.

Case D: The rear end of the vehicle is tilted up (Figure 4.25). A negative pitch angle would be measured by the gyroscope sensor.

**Figure 4.25 Elevation Measurement When the Vehicle is Pitched Down**




From Figure 4.25, $H$ and $h$ can be derived from Equations 4.30 and 4.31.

$$H = (H_0 + D_F \cdot \tan\beta) \cdot \cos\beta \tag{4.30}$$

$$h = H - L \cdot \cos\beta \tag{4.31}$$

Taking the same process as in case A, $e$ can be expressed by Equation 4.32

$$e = \left|(H_0 - L) \cdot \cos(\beta + \beta_r) + D_F \cdot \sin(\beta + \beta_r) - (H_0 - L) \cdot \cos\beta - D_F \cdot \sin\beta\right| \tag{4.32}$$

The maximum allowable errors, $\beta_r$, can be derived using Equations 4.33 and 4.35 under conditions 1 and 2, respectively. Because condition 2 gave smaller values, the results under condition 2 was chosen for further error analysis. A three-dimensional model for $\beta_r$ is shown in Figure 4.26.

Condition 1: when $e = 0.01$,

$$\beta_r = arcsin(\frac{0.01}{\sqrt{(H_0 - L)^2 + D_F^2}} + sin(\theta + \beta)) - \theta - \beta \qquad (4.33)$$

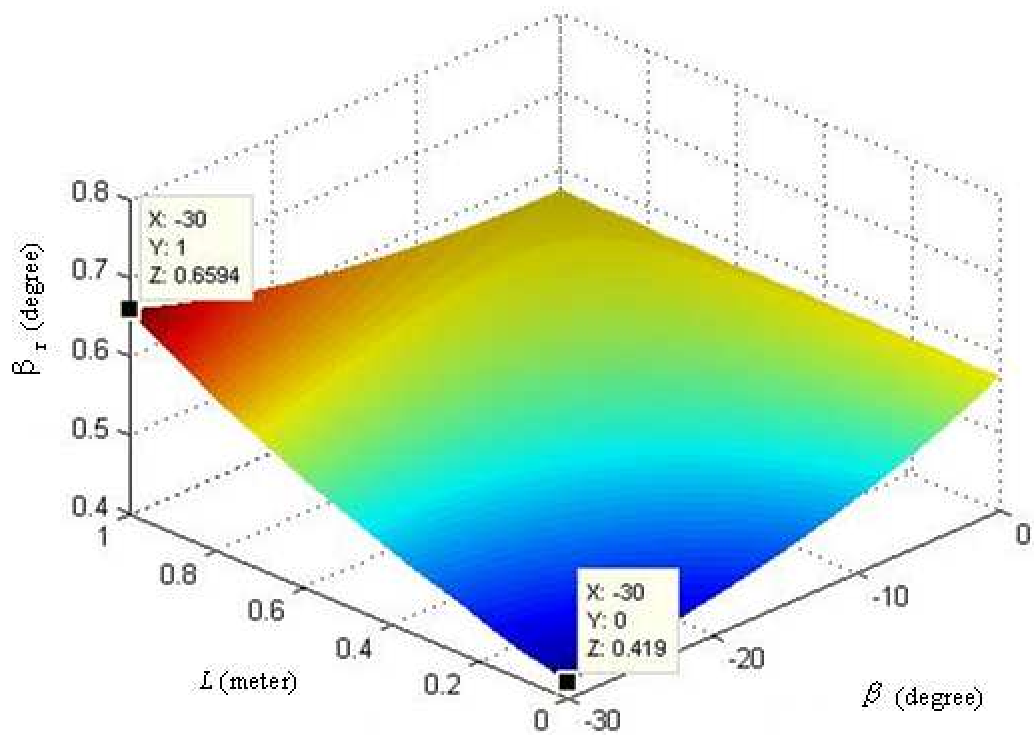
where $\theta = arccos(\frac{D_F}{\sqrt{(H_0 - L)^2 + D_F^2}})$. (4.34)

Condition 2: when $e = -0.01$,

$$\beta_r = arcsin(\frac{-0.01}{\sqrt{(H_0 - L)^2 + D_F^2}} + sin(\theta + \beta)) - \theta - \beta \qquad (4.35)$$

where $\theta = arccos(\frac{D_F}{\sqrt{(H_0 - L)^2 + D_F^2}})$. (4.36)

**Figure 4.26 The Maximum Allowable Error for Pitch Angle Measurement to Limit the Elevation Measurement Error within ±1 cm when the Vehicle is Pitched Down**



The maximum allowable error of the pitch angle measurement reached its maximum value, 0.6594°, when the measured distance (*L*) was 1 m and the vehicle was pitched down to -30°. It reached the minimum value, 0.4190°, when the measured distance (*L*) was 0m and the vehicle was pitched down to -30° (Figure 4.26). Recalling that the measurement error for the pitch angle was less than 0.416° when the pitch angle within -18°, and the error was greater than 0.67° when the pitch angle was between -30° to -27° (Figure 4.13a), the following conclusions can be drawn for case D: (1). The pitch measurement error would not significantly affect the elevation measurement when the pitch angle of the vehicle was within 0 to -18°, (2). The pitch measurement error would significantly affect the elevation measurement when the pitch angle was higher than -27°, (3). When the pitch angle was between -18° and -27°, the effect of pitch measurement error on the elevation measurement depended on the measured distance (*L*).

# CHAPTER 5 - System Tests

## 5.1 Accuracy in Elevation Measurement

### *5.1.1 Description*

Three-dimensional raw laser data was registered in a local coordinate system by combining measured distance with position information provided by two optical encoders. This data was the foundation to generate the three-dimensional digital elevation model (DEM). Inaccuracies in the 3D raw data would result in both low accuracy and low precision of the DEM data.

The DEM data was created by interpolating the 3D raw data into a regular, square grid. Adequate data processing algorithms were implemented before and after the interpolation to improve the accuracy. The accuracy of the system was evaluated by comparing the DEM data with a reference DEM data measured by a more accurate laser system.

In addition to distance measurement, the laser system also provided the intensity data of the reflected laser signals, which can be used to generate grayscale images of the measured surface. Additional information on the target surface, such as color and texture, can be obtained from these images.

### *5.1.2 Objectives*

The objectives of the accuracy tests were:

1). to develop algorithms for creating three-dimensional DEM data from the raw laser data;

2). to evaluate the repeatability of the laser system in DEM measurement;

3). to evaluate the accuracy of the laser system in DEM measurement by comparing with the DEM measured by a reference system;

4). to study the potential use of the gray-scale data.

### *5.1.3 Methodology*

#### *5.1.3.1 Test of Surface with Known Geometric Shapes*

##### *5.1.3.1.1 Surface Characteristics*

This test was conducted by scanning two surfaces with various objects. The first surface was a rectangular box placed on a white paper. The laser line scanner was positioned 116 cm above the paper (Figure 5.1). The box was 22.54 cm long, 13.33 cm wide, and 11.43 cm tall.  The second surface had four objects of different shapes (Figure 5.2).  Dimensions of the objects were given in Figure 5.3. For both surfaces, the laser line scanner used a sample rate of 40 KHz, a mirror rotation speed of 1400 RPM, and a max range setting of 183 cm. The carriage of the scanner moved on the rail at a speed of 6 cm/s. The scan tests were replicated five times on surface 1 and twice on surface 2. The measured and actual dimensions of the box were compared. The correlation coefficients between DEMs obtained by the laser system and digitized models of the surfaces were calculated.

**Figure 5.1 Test Surface 1 with a Box of a Known Geometric Shape**

**Figure 5.2 Test Surface 2 with Four Objects of Known Geometric Shapes**

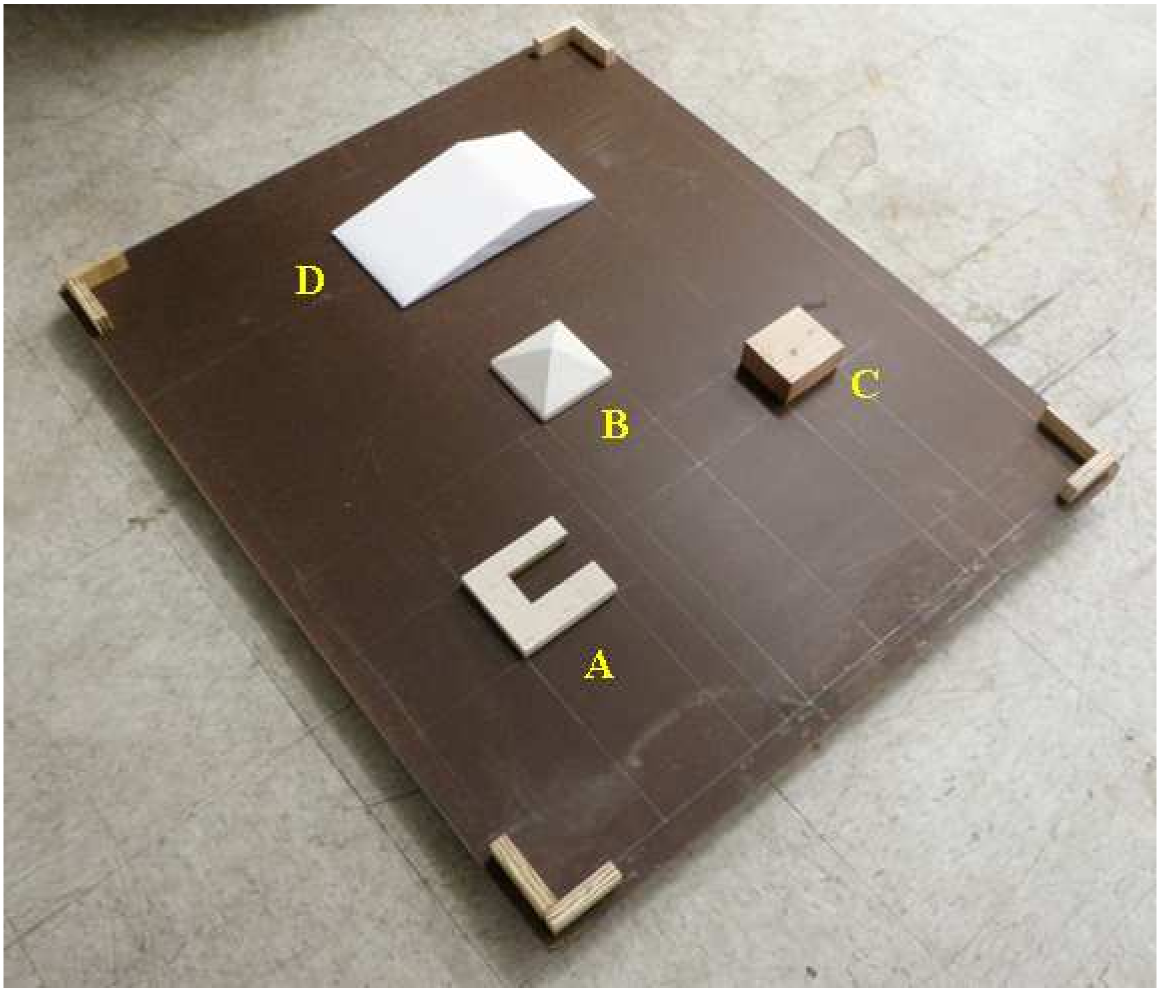
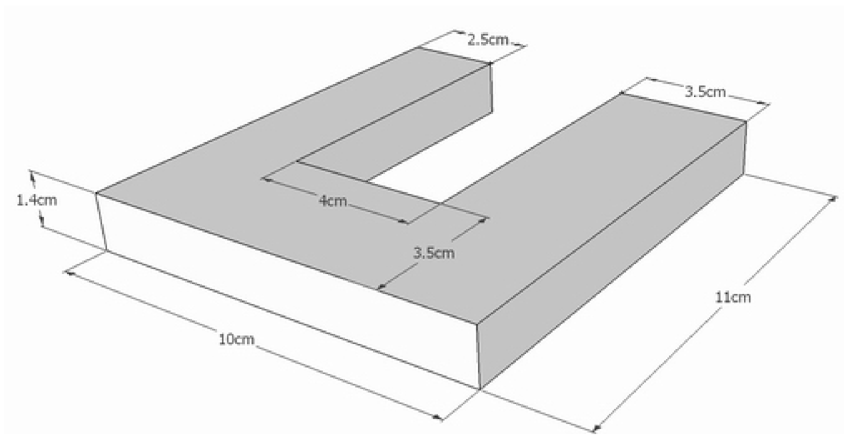
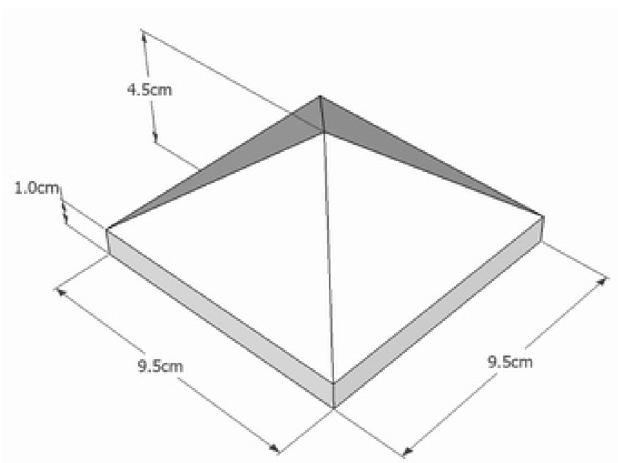
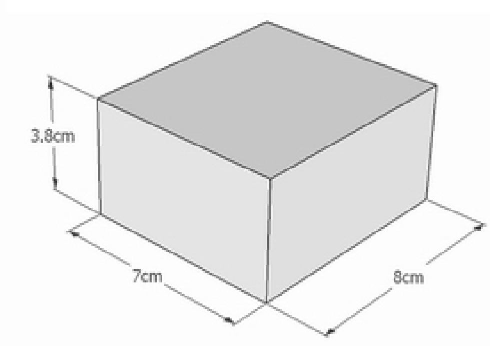


**Figure 5.3 Dimensions of the Four Objects Placed on Surface 2: (a) Object A, (b) Object B, (c) Object C, and (d) Object D**

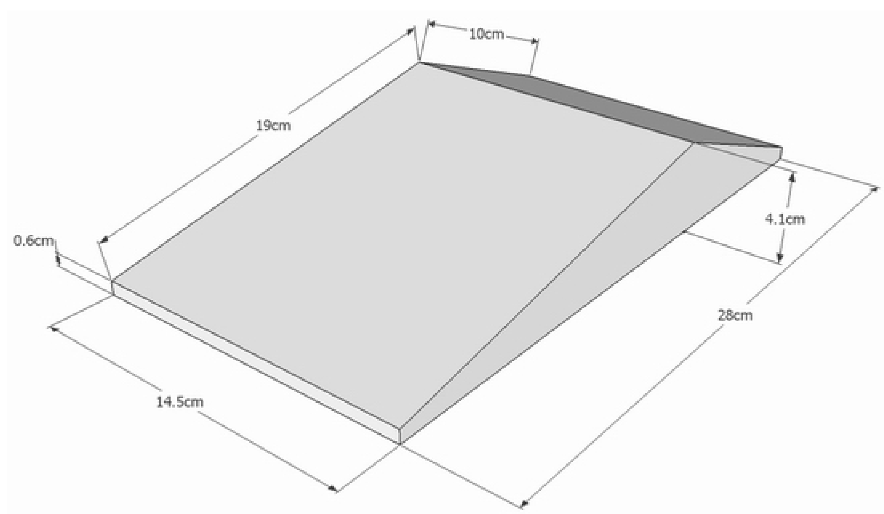



**(a)**




**(b)** **(c)**

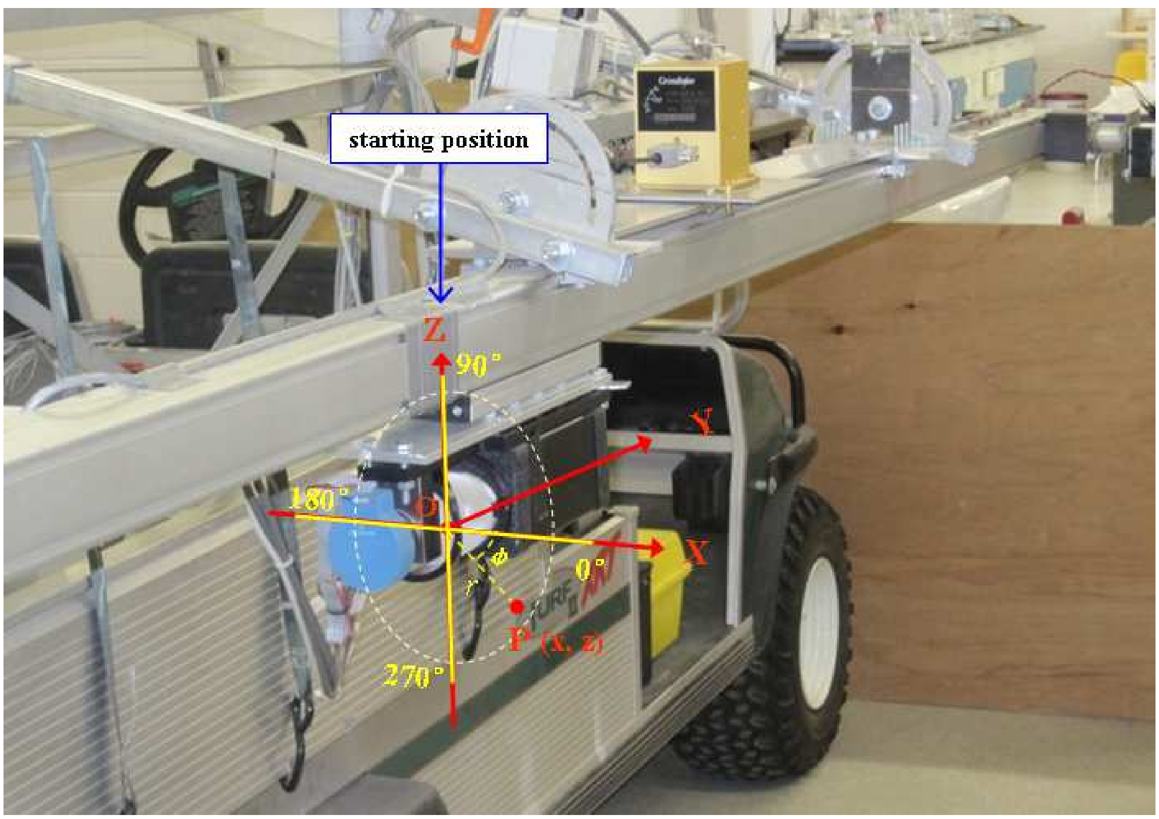



**(d)**

#### *5.1.3.1.2 Coordinate Conversions*

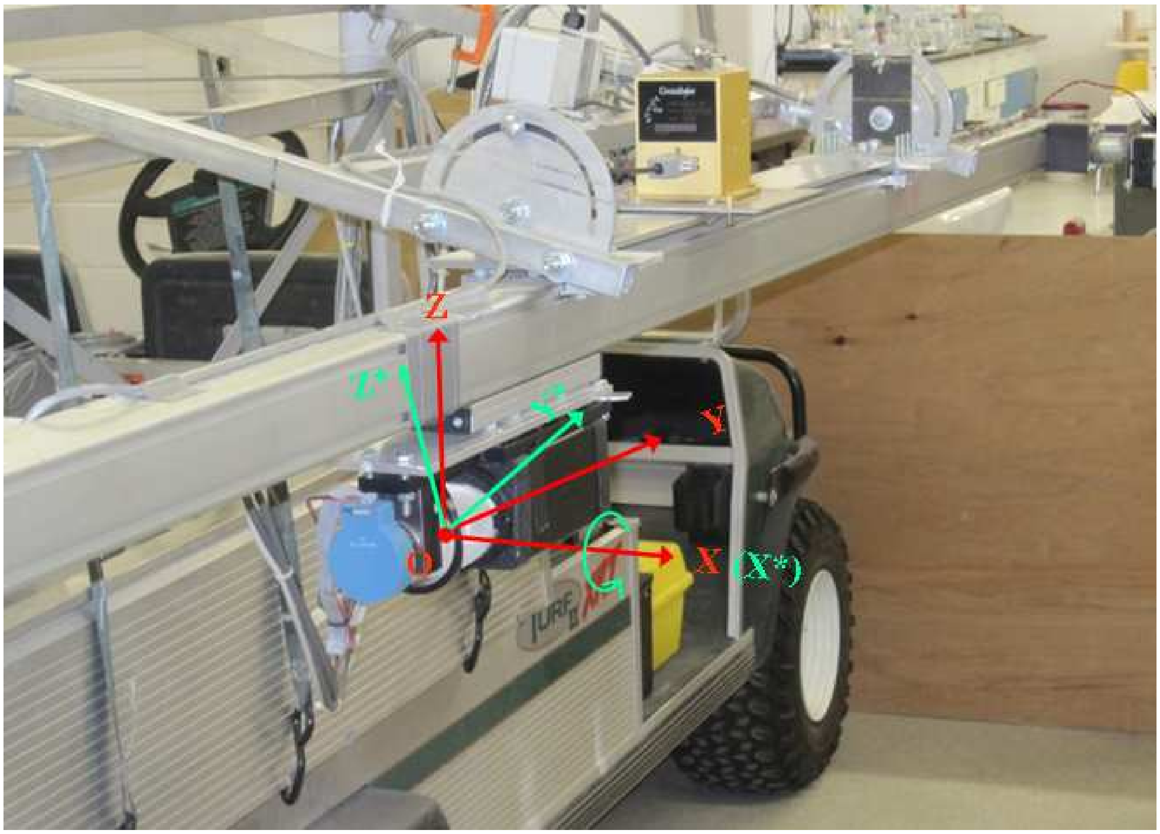
The distance ($r$) measured by the laser line scanner and the rotational angle ( $\phi$) of the laser beam described the measured surface in a polar coordinate system. The origin of the polar coordinate system was at the center of the rotating mirror when the laser-carriage was at the starting position on the rail, which was determined by a limit switch. The DEM of the measured surface, on the other hand, was best represented in a Cartesian coordinate system (Figure 5.4). The origin of the Cartesian coordinate system coincided with that of the polar coordinate system. The direction of the X-axis was aligned with the horizontal direction pointing to the right of the vehicle. The direction of the Z-axis pointed upwards. For both coordinate systems, the Y-axis pointed to the front direction of the vehicle.

**Figure 5.4 Conversion from Polar to Cartesian Coordinates**




When the linear rail was not leveled with the ground, as in the case of non-zero roll or pitch rotations, the laser measurements were registered in a new 3D Cartesian coordinates (X* ,Y* ,Z*). Figure 5.5 demonstrates a case with a positive pitch angle. In this case, a coordinate transformation from the (X* ,Y* ,Z*) system to the (X, Y, Z) system was needed.

**Figure 5.5 Rotation of the Coordinate System at a Positive Pitch Angle When the Linear Rail Is Tilted**




The coordinate conversion may include translation, scaling, and rotation, or their combinations (Gonzalez and Wintz, 1987). A generalized form of the conversion is shown in Equation 5.1.

$$V = R_{\alpha} \cdot R_{\beta} \cdot R_{\theta} \cdot S \cdot T \cdot V^{*} \tag{5.1}$$

where $V = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$ (5.2)

$$V^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} \quad (5.3)$$

and T is for translation,

$$T = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.4)$$

where $X_0$, $Y_0$, $Z_0$ are the coordinates of the origin of the new coordinate system (X, Y, Z) in the old coordinate system (X*, Y*, Z*).

The S is for scaling:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.5)$$

where $S$x, $S$y, $S$z are the scaling factors between the X, Y, and Z axes and the X*, Y*, and Z* axes, respectively.

$R_\alpha$, $R_\beta$, and $R_\theta$ are the rotation factors for the X*, Y*, Z* axes, respectively.

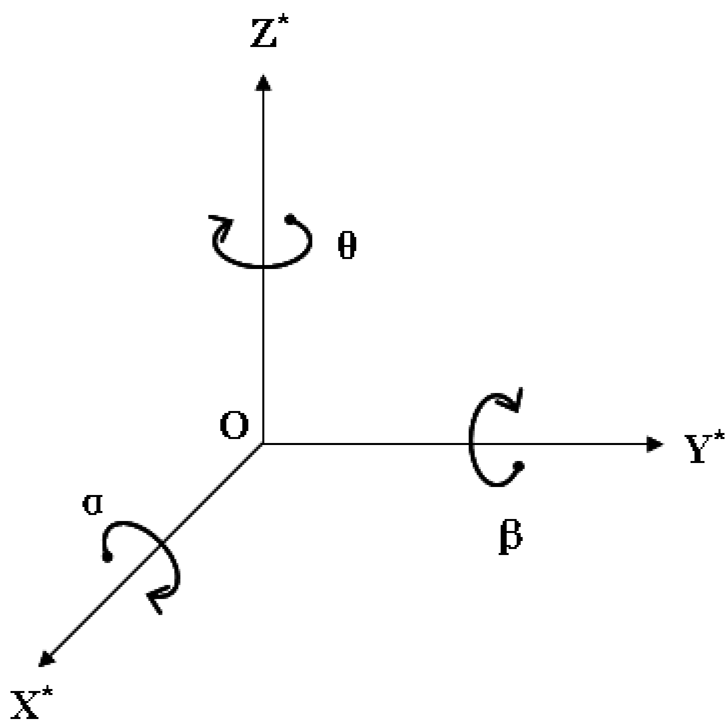
$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.6)$$

$$R_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

where $\alpha$, $\beta$, $\theta$ are the rotational angles of the Y-Z, Z-X, and X-Y planes about the X*, Y*, and Z* axes, respectively (Figure 5.6).

**Figure 5.6 Rotation Angles of the Y-Z, Z-X, and X-Y Planes about the X*, Y*, Z* Axes**




As a result, the general orthogonal transformation can be written as

$$V = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \tag{5.9}$$

$$\begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot V^*$$

For a simple pitch rotation, the coordinate transformation can be simplified as

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X^* \\ \cos\alpha \cdot Y^* + \sin\alpha \cdot Z^* \\ -\sin\alpha \cdot Y^* + \cos\alpha \cdot Z^* \\ 1 \end{bmatrix}. \tag{5.10}$$

It should be noted that the pitch angle measured by the gyroscope sensor had a direction that was opposite to the positive direction of angle *a* shown in Figure 5.6. This algorithm was implemented in MATLAB (Appendix E).

#### *5.1.3.1.3 Interpolation algorithm for DEM generation*

A digital elevation model (DEM) was a digital, or numerical, representation of ground surface topography. Regular grids, or square grids, were commonly used in DEM. The grid DEM used a matrix structure that implicitly recorded topological relations between data points (El-Sheimy et al., 2005). In a DEM, each grid cell had a value corresponding to its elevation, which was usually obtained by interpolation between adjacent sample data points.

The grid DEM was a simple and efficient approach in data storage and manipulation since this data structure was similar to the array structure in a computer (Ziadat, 2007). However, this approach may introduce errors because it discontinuously represented the surface, especially when coarser grid sizes were used (Ramirez, 2006). In this study, data acquired by the laser system had a high density, which allowed a fine grid size.

The interpolation was an approximation procedure in mathematics and an estimation issue in statistics (Li et al., 2004). It was a process of predicting a new value of a given variable in un-sampled locations within an area of known data points (Liu, 2008). For DEM data, interpolation was used to determine the elevation of a point by using known elevations of neighboring points. Two implicit assumptions were applied to the interpolation process: the surface was continuous and smooth, and there was a high correlation between the neighboring data points (Li et al., 2004).

The interpolation algorithm used in this study was a two-dimensional, n-nearest neighbor, distance-weighted interpolation (Zhang et al., 1989). The number of nearest neighboring data points was three. The algorithm employed weighting functions to obtain smooth surfaces.

The first step of the interpolation algorithm was to select three nearest neighboring data points surrounding each cell on the regular grids from the raw laser data on the irregular grids. Assuming data points A($x_1$, $y_1$), B($x_2$, $y_2$), C($x_3$, $y_3$) were the three nearest data points surrounding cell O(X, Y) (Figure 5.7), polynomial weighting functions can be calculated using the following equations:

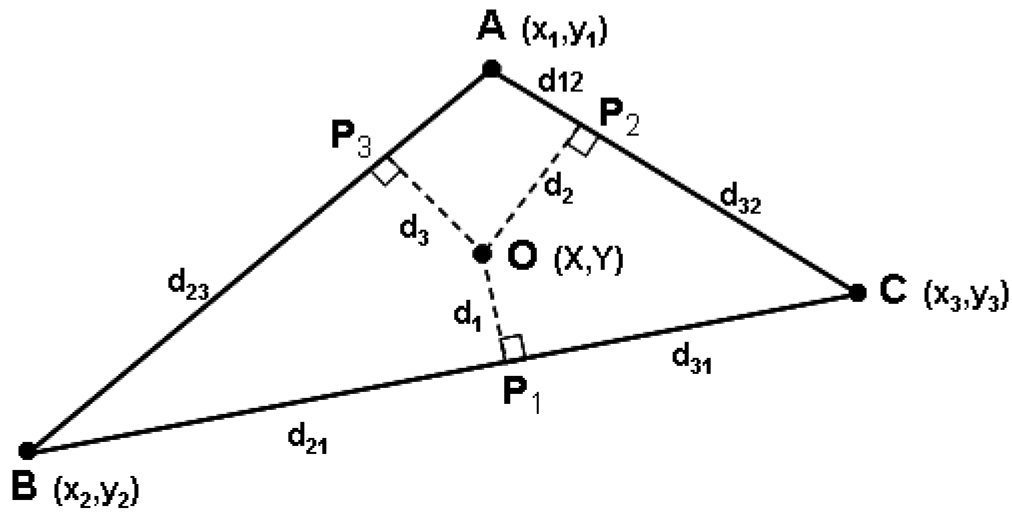
$$W_1(x_1, y_1) = d_1^2(d_2^2 d_{23}^2 + d_3^2 d_{32}^2) \tag{5.11}$$

$$W_2(x_2, y_2) = d_2^2(d_1^2 d_{13}^2 + d_3^2 d_{31}^2) \tag{5.12}$$

$$W_3(x_3, y_3) = d_3^2(d_1^2 d_{12}^2 + d_2^2 d_{21}^2) \quad (5.13)$$

where $d_1$, $d_2$, and $d_3$ were the distances between the center of cell O and the three data points.

**Figure 5.7 Distance-weighted, three-nearest neighbor Interpolation**




The elevation, Z, of cell O was then determined as the weighted average of the elevations of points A, B, and C:

$$Z = \frac{W_1(x_1, y_1)z_1(x_1, y_1) + W_2(x_2, y_2)z_2(x_2, y_2) + W_3(x_3, y_3)z_3(x_3, y_3)}{W_1(x_1, y_1) + W_2(x_2, y_2) + W_3(x_3, y_3)} \quad (5.14)$$

where $z_1$ to $z_3$ = the elevations at the three data points.

This algorithm was implemented in MATLAB (Appendix F).

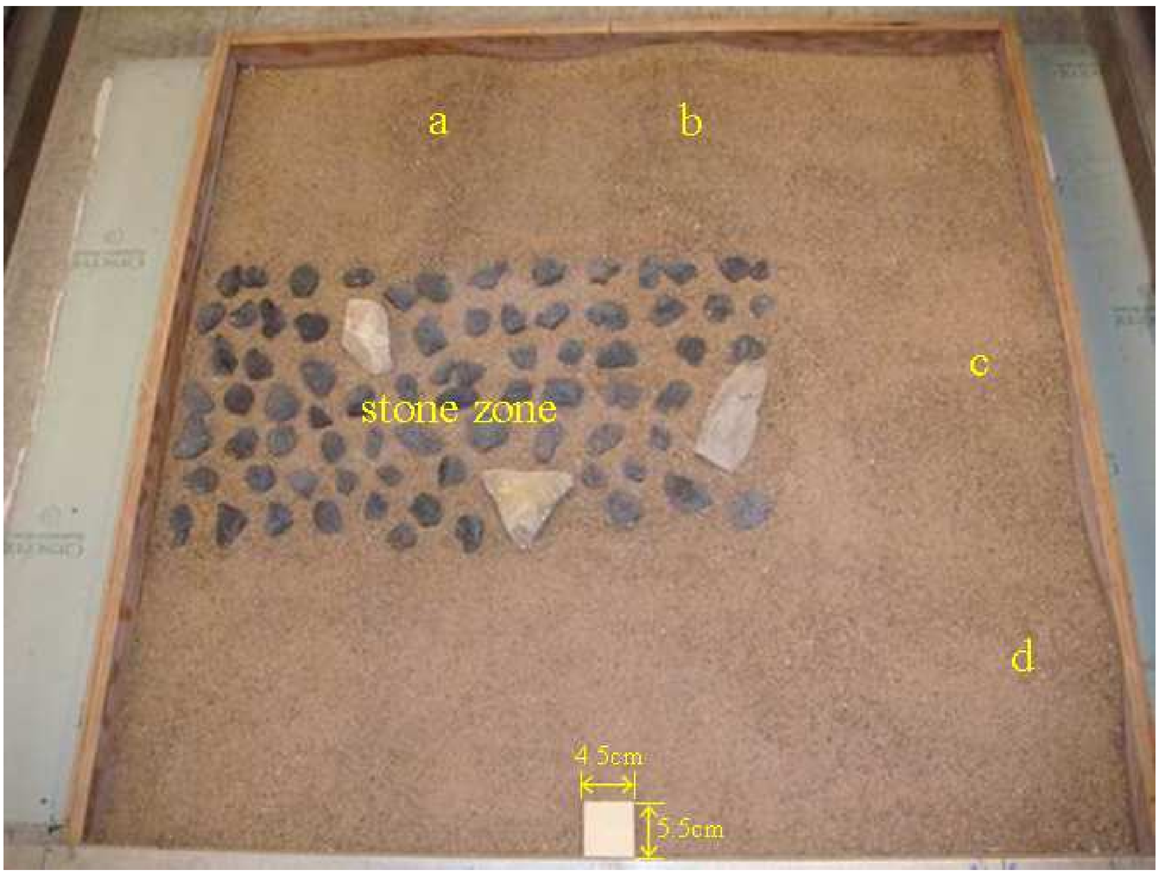
***5.1.3.2 Test of Unknown Surface***

DEM data of an unknown surface were measured using the laser system and a reference system. Accuracy of the measurement was evaluated by comparing the two DEM data.

*5.1.3.2.1 Surface Characteristics*

An artificial surface made of sands and stones was constructed as a testing surface (Figure 5.8). The surface was formed by four small regions, labeled a, b, c, and d, respectively, a stone zone, and a small wooden block. The surface was surrounded by a 1m by 1m wooden frame.

**Figure 5.8 A Test Surface**

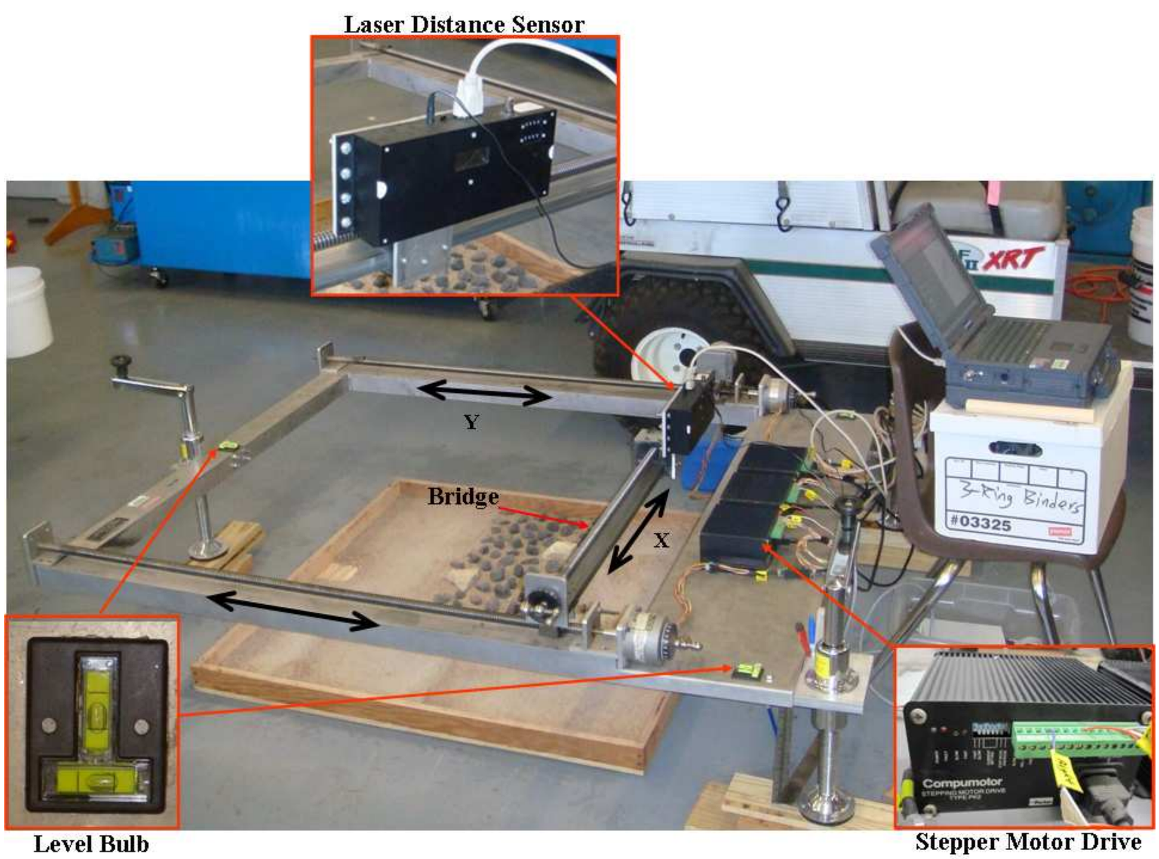



*5.1.3.2.2 The Reference System*

A laser-based profile meter was used as the reference system (Figure 5.9). This meter included three components: a laser distance sensor, NR-40-105 (Nova Range Inc., 2001), to measure surface elevation, a computed-controlled, motor-driven, two-dimensional traversing frame, and a data acquisition card, PC-CARD-DAS16/16AO (Measurement Computing Corp., 2007), plugged into a field laptop to control the motion of the laser-carriage and to register the elevation data.

The traversing frame had a 1.2 m by 1.5 m rectangular base constructed from aluminum tubing (Figure 5.9). A bridge carrying the laser distance sensor rode on the frame in the X-direction. The motion of the laser distance sensor along the bridge was accomplished by a screw-nut mechanism, in which the screw was positioned parallel to the bridge and was driven by a stepper motor, whereas the nut was mounted on the carriage of the laser distance sensor. A plate was attached on one side of the frame to serve as a platform for electronic components. Two ball screws, each 1.2 m long, were mounted on two sides of the frame, in the Y-direction. These ball screws were supported by bearings on both ends. A stepper motor was attached to each screw by a sleeve. When turned simultaneously, two ball screws can provide the bridge motions in Y-direction. The traversing frame as a reference plane was equipped with three leveling screw jacks and three dual-vial bulb levels to allow level adjustment before each scan.

**Figure 5.9 The Laser-based Profile Meter and the Traversing Frame**



The laser distance sensor was a CLASS IIIa product, which emitted a red laser beam of 670 nm wavelength, received reflected beam, and calculated the distance based on the triangulation principle. It provided an accuracy of 0.05cm within 60cm range. The output was either a 4 to 20 mA current loop or RS-232 signals. Configuration of the RS-232 interface was: 9600 baud rate, 8 data bits with no parity, 1 stop bit, and hardware flow control.

Three PK2 packaged stepper drives (Parker Hannifin Corp., 1997) were screwed on the frame platform to drive three stepper motors. This bipolar stepper drive had motor connections, control signal connections, and function switches to allow flexible control. Descriptions of the connections and switches are listed in Table 5.1.

**Table 5.1 Connection and Switches on PK2 Stepper Drive (from Parker Hannifin Corp. 1997)**

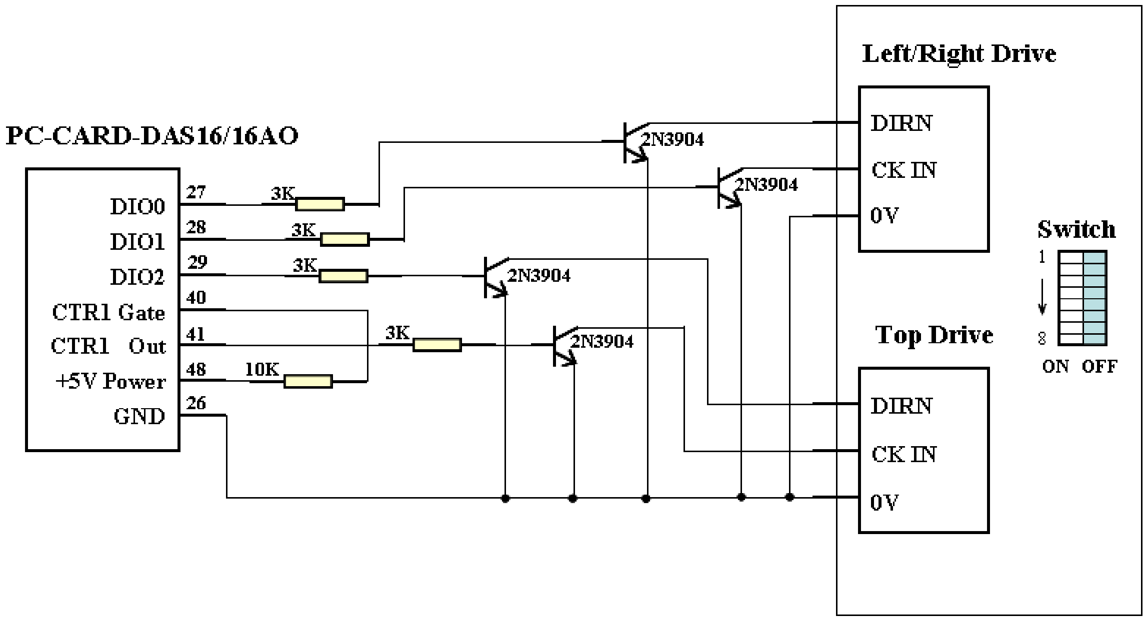
| Terminal | | Description |
|---|---|---|
| Motor Connections | 1A | Connect one phase of the motor |
| | 1B | |
| | 2A | Connect the other phase of the motor |
| | 2B | |
| Signal Connections | 0V | Common return point for control signals |
| | Fault output | Output high in the event of a drive fault |
| | Energize/Reset | High to enable motor to be de-energized, reset a fault condition |
| | Dirn | Direction of motor rotation, 0v to reverse |
| | Ck.in | Clock input to make motor step at low-going transition |
| | Slow | 0v to run the internal oscillator at low rate, 40~1000 steps/sec. |
| | Fast | 0v to run the internal oscillator at fast rate, 400~ 1000 steps/sec. |
| | Fast Adj | Connect to an external potentiometer (10K) between Adj.com to control the fast speed of the internal oscillator |
| | Adj.com | Common return connection for external speed controls |
| | Slow Adj | Connect to an external potentiometer (100K) between Adj.com to control the slow speed of the internal oscillator |
| Switch Settings | Switch1-Energ | Turn on to permanently energize the drive |
| | Switch2-Mode | Turn on to select Full step mode and off to Half step mode |
| | Switch3 | Motor current set: 2.0A all switches off; 1.8A only Switch6 on; 1.6A only Switch5 on; 1.4A only Switch4 on; 1.2A only Switch3 off; 1.0A only Switch3 on. |
| | Switch4 | |
| | Switch5 | |
| | Switch6 | |
| | Switch7-Slow | Turn on for internal slow speed adjustment and off for external speed control |
| | Switch8 - Fast | Turn on for internal fast speed adjustment and off for external speed control |

PC-CARD-DAS16/16AO was a data acquisition and control board for a PC computer with PCMCIA type II interface. It provided 16 single-ended or 8 differential analog inputs with 16-bit A/D resolution, two analog outputs, four digital I/O lines, and three 16-bit down counters. Four digital I/O lines and one counter were used to connect to the stepper drives to control the motors running in independent or continuous mode.

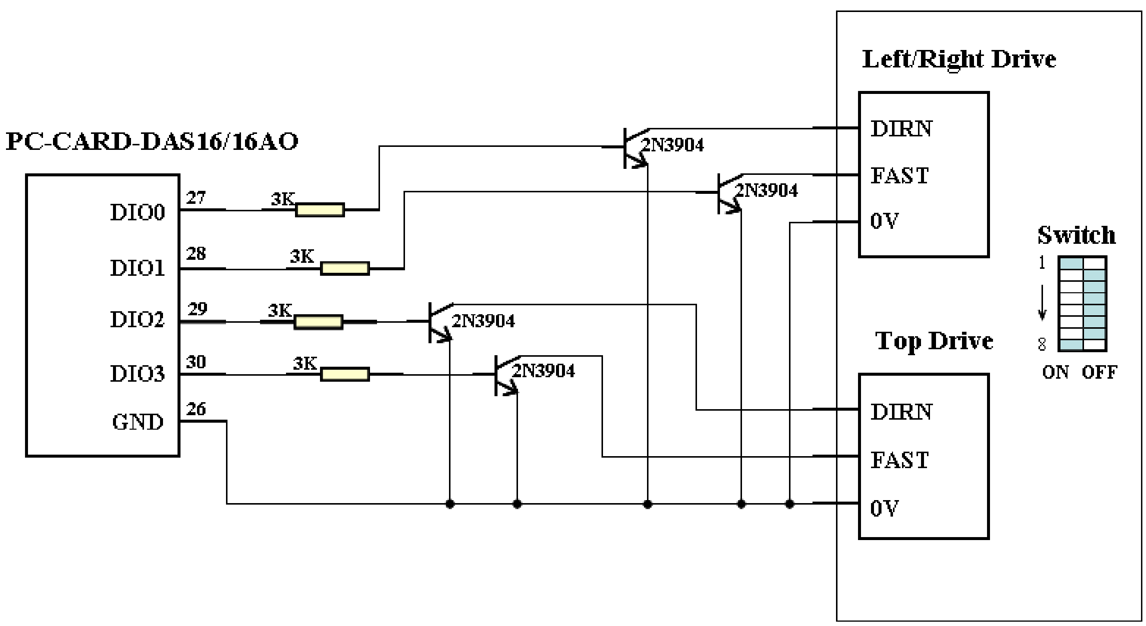
In the independent mode, the motor stepped at negative edges of the clock input of the PK2 stepper drive. This input was driven by an external clock source generated by a port of the PC-CARD-DAS16/16AO board. The continuous mode allowed the motor to continuously run at a predefined speed determined by a built-in potentiometer that controlled the frequency of an internal oscillator.

Figure 5.10 shows wire connections between the data acquisition board and the stepper drives for two modes. Two LabVIEW (LabVIEW, 2004) programs were developed to control the motors running in these modes, respectively (Appendix G). Sampled laser data was registered into a text file. In independent mode, the spatial sampling interval was limited to 0.5 cm and it took eight hours to scan a 1m by 1m surface. When the sampling interval increased to 0.2 cm in the continuous mode, time required to scan the surface was reduced to five hours.

**Figure 5.10 Connections between the PC-CARD-DAS16/16AO and PK2 Stepper Drives: (a) Independent Mode, and (b) Continuous Mode**




(a)




(b)

#### *5.1.3.2.3 Test Procedure*

Both the laser system and the reference system measured the sand-stone surface. The laser system scanned the surface with a sampling rate of 40 KHz, a mirror rotation speed of 1400 RPM, and a maximum range setting of 183 cm. The carriage of the scanner moved on the rail at a speed of 6 cm/s. For the reference system, motors were driven in the continuous mode. Because the position at which the first measurement was made on each scan line varied, the distance between this position and the edge of the wooden frame was recorded. During data processing, these distances were used to correct the coordinates of the data points in the X direction. The spatial resolutions of the measured data were 0.2 cm and 0.3 cm in the X and Y directions, respectively. The DEM data generated by both systems were plotted and compared.

#### *5.1.3.2.4 Matching by correlation*

The DEM data measured by the laser system and the reference system were compared using the image matching method (Gonzalez and Woods, 2002), which calculated the correlation coefficient between the two DEM data.

If the DEM data obtained by the reference system and the laser system were defined as $f(x, y)$ of size M*N and $w(x, y)$ of size J*K, respectively, the correlation function between $f$ and $w$ can be calculated as

$$c(x,y) = \sum_{s=0}^{J-1}\sum_{t=0}^{K-1} f(s,t)w(x+s, y+t) \tag{5.15}$$

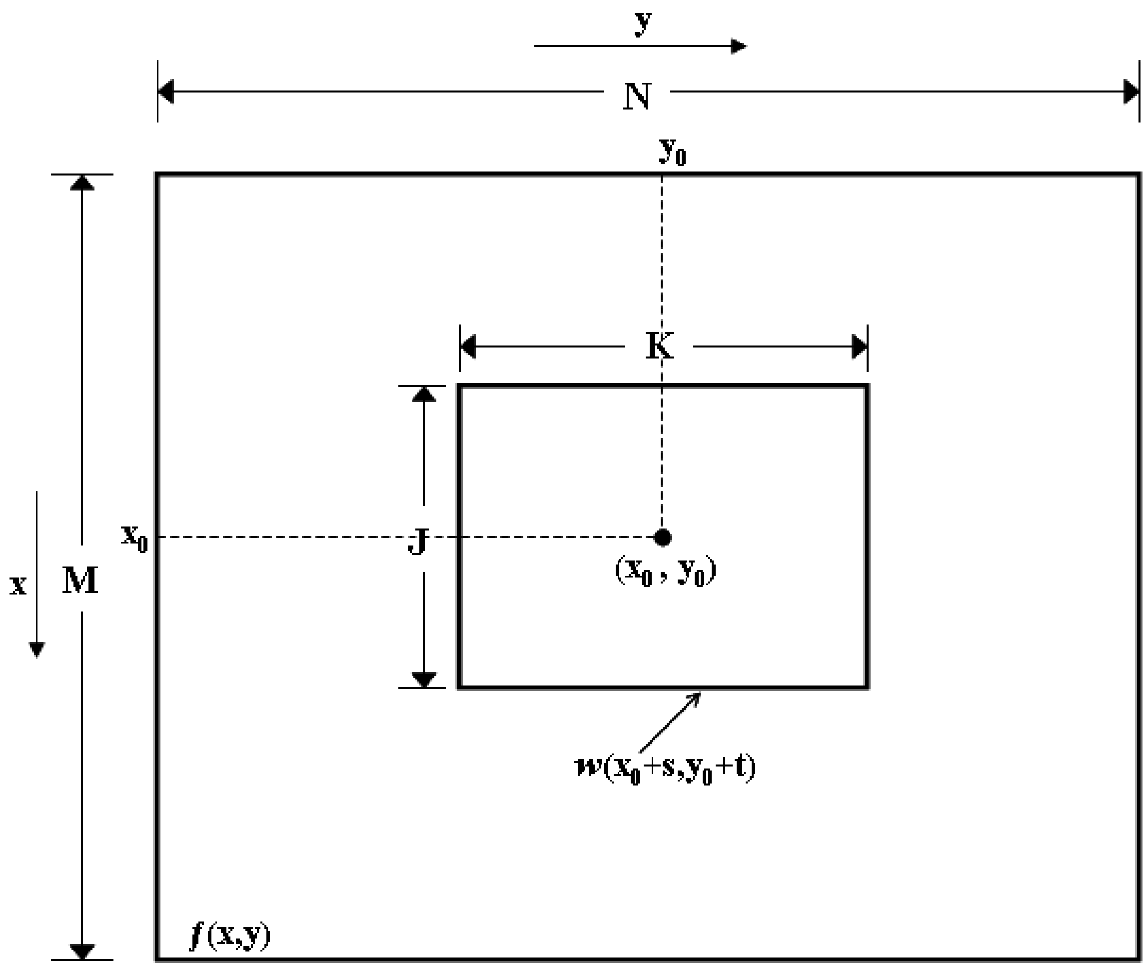
where x = 0, 1, 2 …M-1,

y = 0, 1, 2…N-1, and

the summation is taken over the data region where $w$ and $f$ overlap.

Figure 5.11 illustrates the procedure for computing $c(x, y)$. As x and y varied, $w$ moved within the area of $f$, giving the function $c(x, y)$ at each location. The maximum value of $c$ indicated the position where $w$ best matched $f$.

**Figure 5.11 Correlation Function between the Two Measured DEM Data (from Gonzalez and Woods, 2002)**




The correlation function given in Equation (5.15) had the disadvantage of being sensitive to changes in the amplitudes of $f$ and $w$. A frequently used approach to overcome this difficulty was to use the correlation coefficient, which was defined as:

$$r(x,y)=\frac{\sum_{s=0}^{J-1}\sum_{t=0}^{K-1}\left[f(s,t)-\bar{f}(s,t)\right]\left[w(x+s,y+t)-\bar{w}\right]}{\left\{\sum_{s=0}^{J-1}\sum_{t=0}^{K-1}\left[f(s,t)-\bar{f}(\mathrm{s},t)\right]^2\sum_{s=0}^{J-1}\sum_{t=0}^{K-1}\left[w(x+\mathrm{s},y+t)-\bar{w}\right]^2\right\}^{1/2}} \qquad (5.16)$$

where x = 0, 1, 2…, M-1,

y = 0, 1, 2…, N-1,

$\bar{w}$ is the average value of the elevations in *w* (computed only once),

$\bar{f}$ is the average value of $f$ in the region coincident with the current location of *w*, and

the summations are taken over the coordinates common to both $f$ and *w*.

The correlation coefficient *r*(x, y) was independent of scale changes in the amplitudes of $f$ and *w*. This algorithm was used to evaluate the accuracy and repeatability of DEM measurement. It was also used to analyze the effects of noise on DEM measurement. This algorithm was implemented in MATLAB (Appendix H).

#### *5.1.3.2.5 Spatial filters*

A spatial filter (also called spatial mask, kernel, template, or window) is an image enhancement approach. In this study, two spatial filters were used to smoothen the data measured by the laser scanner before or after the interpolation. The first filter applied was a median threshold filter. The second one was a standard median filter.

The median threshold filter was applied to the raw laser data. It included the following steps:

(1) Specify the size of a rectangular filter, and set the upper and lower limits,
(2) Move the filter through the raw laser data without overlapping,
(3) Sort the values of the elevation in the filter area at each position,
(4) Determine their median,
(5) Remove all values that are out of a range of [median - lower limit, median + upper limit].

In this study, the median filter was only applied to the laser data after interpolation. It consisted of the following steps:

(1) Specify the size of a rectangular filter (typically a 3 by 3 window),
(2) Move the window from pixel to pixel over the laser data,
(3) At each pixel, rank the values within the window,
(4) Determine their median,
(5) Replace the value of the pixel at the center of the window with the median.

It was critical for the median threshold filter to select an optimal combination of filter size and the limits, and for the median filter to determine the window size. These spatial filters may help remove errors in the laser data. They may, however, remove useful details of the data if the parameters were not appropriately selected.

#### *5.1.3.3 Use of Gray-scale Data*

The laser line scanner used in this study provided gray scale data, which depended on the color, texture composition, and orientation of the target.

Intensities of the reflected light were stored as 8-bit integers, allowing 256 gray levels. The two-dimensional, three-nearest neighbor, distance-weighted interpolation algorithm was used to create the grayscale intensity image. The typical pixel size was 0.5 cm by 0.5 cm.

### *5.1.4 Results and Discussion*

#### *5.1.4.1 Test of Surface with Known Geometric Shapes*

One of five measured raw laser data for test surface 1 is shown in Figure 5.12. The spatial resolution of the raw laser data was 0.27 cm by 0.28 cm. Two-dimensional views of the raw laser data of the box are displayed in Figure 5.13.

**Figure 5.12 A Three-dimensional View of the Box on Test Surface 1 using Measured Raw Laser Data**

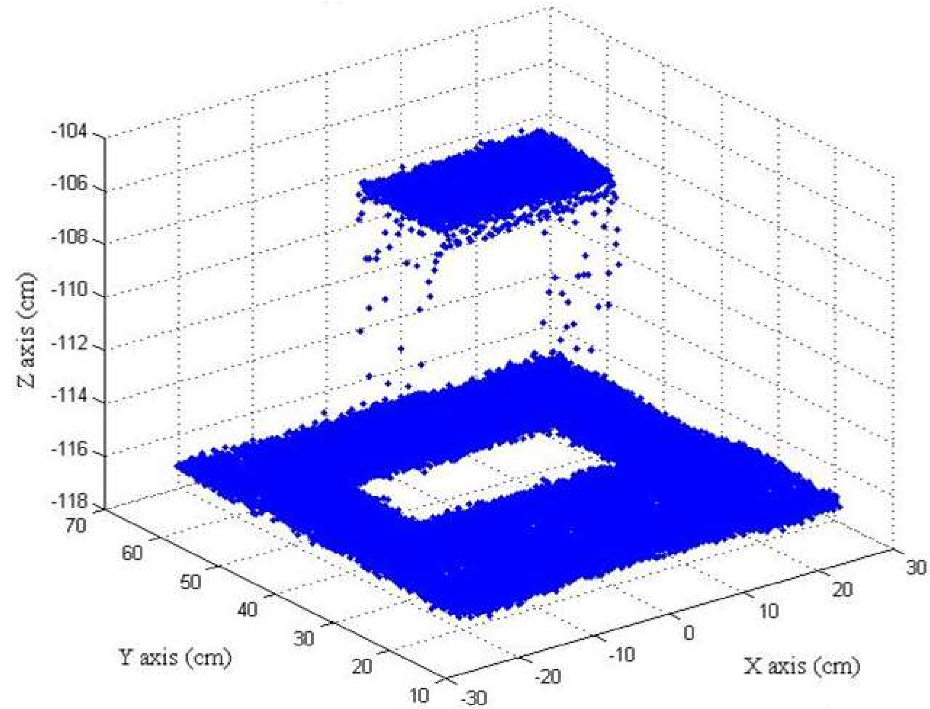
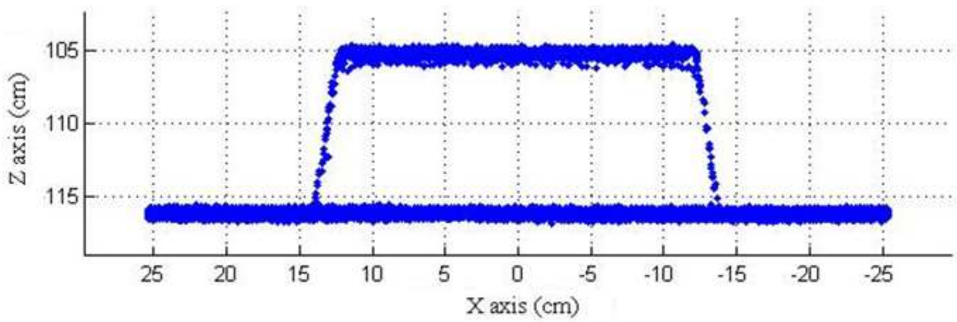
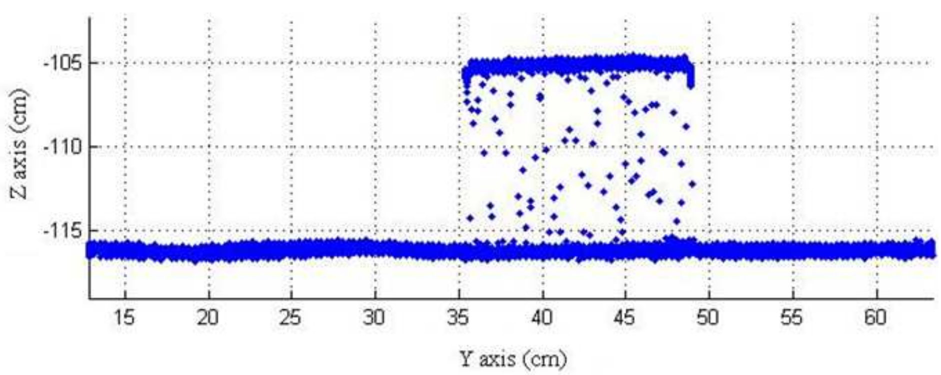



**Figure 5.13 Two-dimensional Views of the Box in (a) X-Z Plane, and (b) Y-Z Plane**




**(a)**




**(b)**

To compare the measured box with the actual box, linear regression analyses were applied to the edges of the box on the raw laser data. The regression lines are showed in Figure 5.14 in cyan color. The actual box (in red color) is positioned in Figure 5.14a based on its actual dimensions and its location in the 3D Cartesian coordinate system. It is postioned in Figure 5.14b, based on its actual dimensions and with its left edge aligned to the regression line. The area covered by the four regression lines was calculated and compared with the actual area of the box. The volume of the parallelepiped formed by the regression lines was also calculated and compared with the volume of the box. Relative errors of area and volume in five replications are listed in Table 5.2.

**Table 5.2 Comparisons of Areas and Volume Measured by the Laser System with the Actual Values of the Box on Test Surface 1**

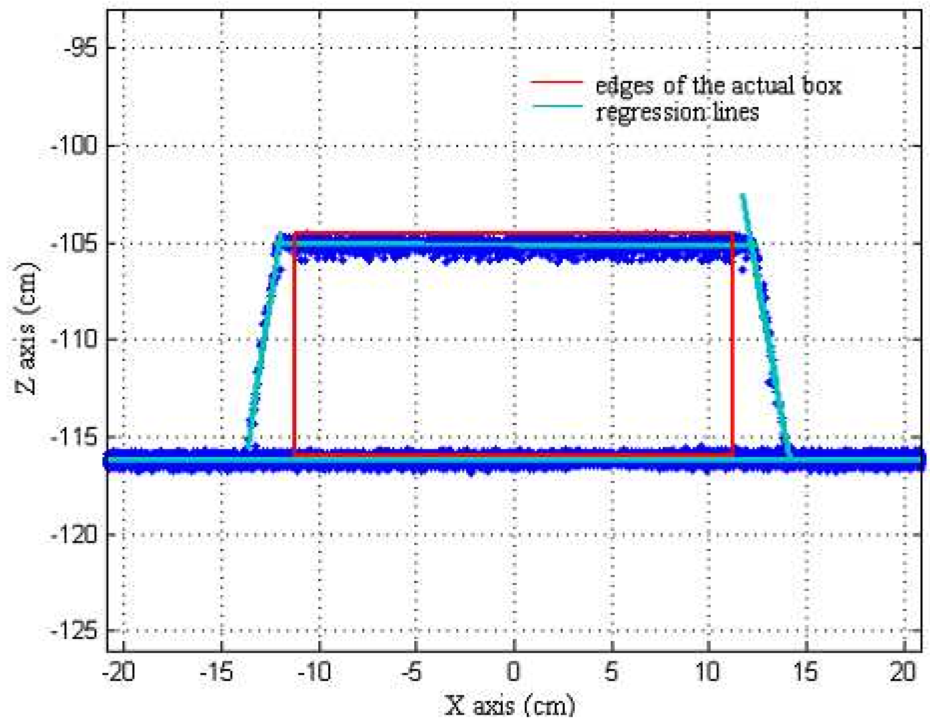
| Reference | Area | | | | Volume (Actual: 3434.24 $cm^3$) | |
|---|---|---|---|---|---|---|
| | X-Z plane (Actual: 257.63 $cm^2$) | | Y-Z plane (Actual: 152.36 $cm^2$) | | | |
| | Measured, $cm^2$ | Relative error(‡), % | Measured, $cm^2$ | Relative error,% | Measured, $cm^3$ | Relative error,% |
| **1** | 291.22 | 13.03 | 147.85 | 2.96 | 3896.56 | 13.46 |
| **2** | 285.59 | 10.85 | 146.85 | 3.62 | 3812.59 | 11.02 |
| **3** | 286.28 | 11.12 | 147.44 | 3.23 | 3854.69 | 12.24 |
| **4** | 289.62 | 12.42 | 148.28 | 2.68 | 3904.08 | 13.68 |
| **5** | 283.54 | 10.06 | 148.07 | 2.82 | 3799.48 | 10.64 |
| **Mean** | 287.25 | 11.49 | 147.70 | 3.06 | 3853.48 | 12.21 |
| **CV**(†) | 1.08% | | 0.38% | | 1.23% | |

‡: Relative error = |measured value – actual value| / (actual value) *100%

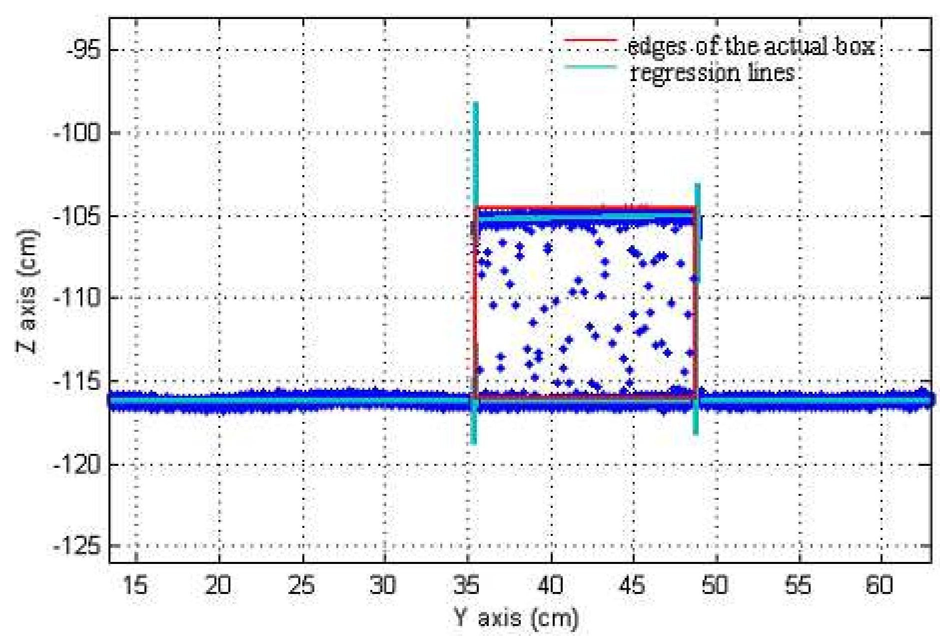
†: CV (coefficient of variation) = standard deviation/ mean * 100%

The results showed the laser system had a good repeatability in area and volume measurements. The area measured in the Y-Z plane was more accurate than that in the X-Z plane.

**Figure 5.14 Two-dimensional Views of the Measured Box with Linear Regression Lines for the Edges in (a) X-Z Plane, and (b) Y-Z Plane**




**(a)**




**(b)**

The DEM of the box was interpolated from the raw laser data. The grid size of the DEM was 0.5 cm by 0.5 cm. Figure 5.15 displays the DEM of the box and Figure 5.16 shows its two-dimensional views. The box gave a trapezoidal look in the X-Z plane (Figure 5.16a). This may have been caused by the fact that, for each scan line, the first measurement was taken at a slightly different angle of the laser beam, hence, slightly different distance between the intercept point of the laser beem and the box to the edge of the box, which may in turn vary the intensity of the reflected light, causing errors in distance measurements.

**Figure 5.15 Measured DEM for Test Surface 1**

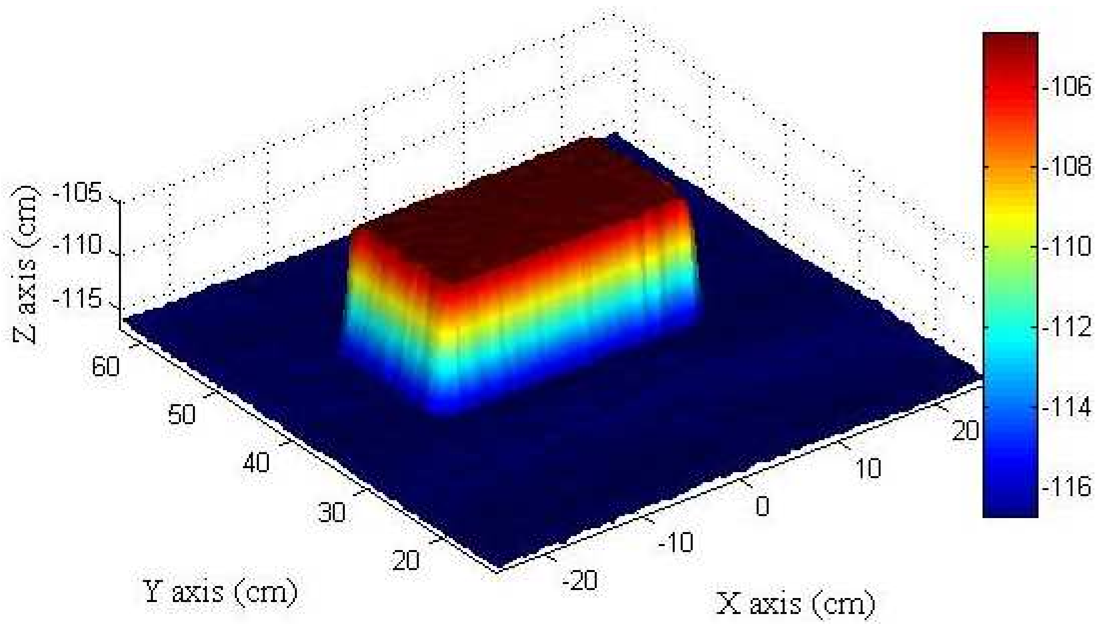
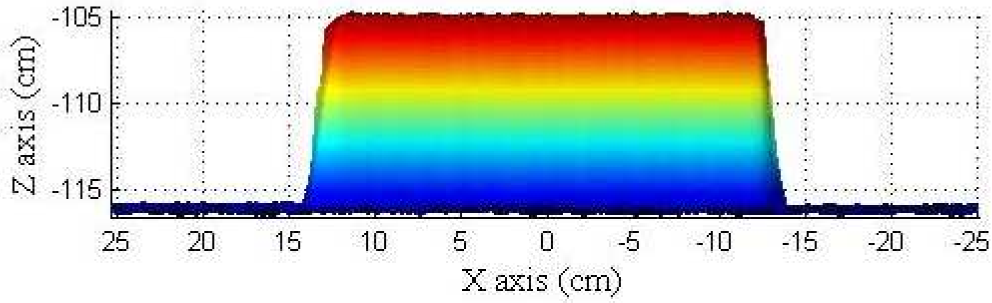
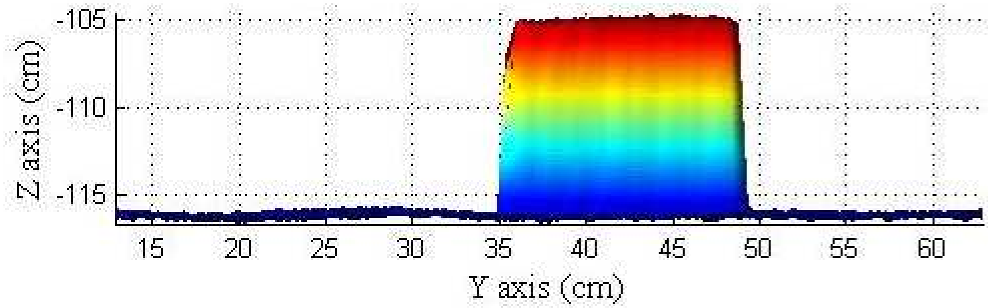


**Figure 5.16 Two-dimensional Views of the Measured DEM for Test Surface 1 in (a) X-Z Plane, and (b) Y-Z Plane**




**(a)**




**(b)**

The DEM of test surface 2 acquired by the laser system is shown in Figure 5.17. The grid size of the DEM was 0.5 cm by 0.5 cm. The four objects can be easily identified from the DEM. Figure 5.18 shows the digitized model of the original objects on a 0.5 cm by 0.5 cm grid. This DEM model was used as the reference model for comparison. Correlation coefficients between the reference DEM and two replications of the measured DEMs are listed in Table 5.3.

**Figure 5.17 DEM of Test Surface 2 Acquired by the Laser System**

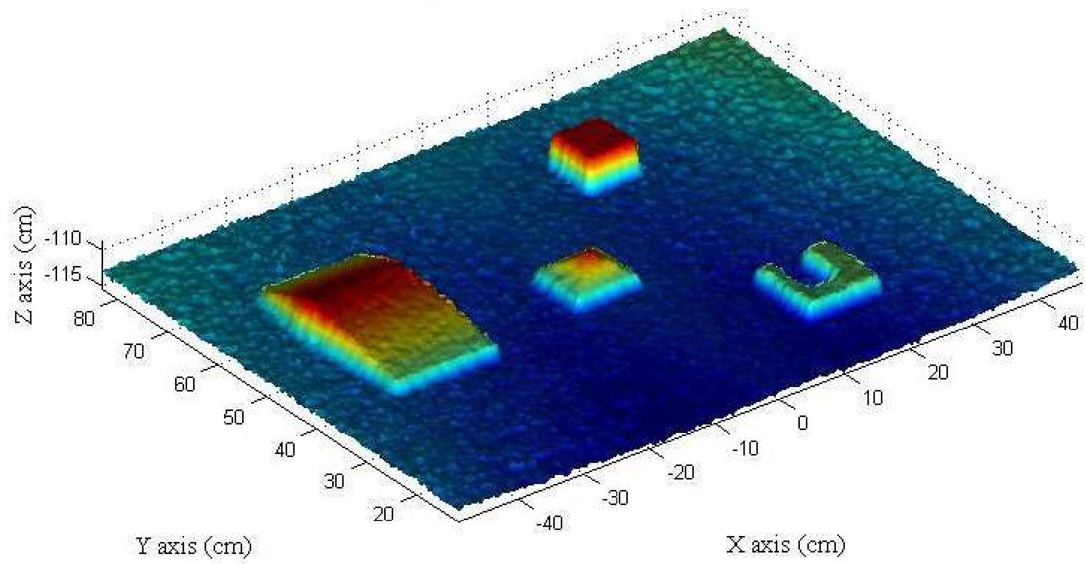



**Figure 5.18 Digitized DEM of Surface 2 using the Known Geometric Parameters**

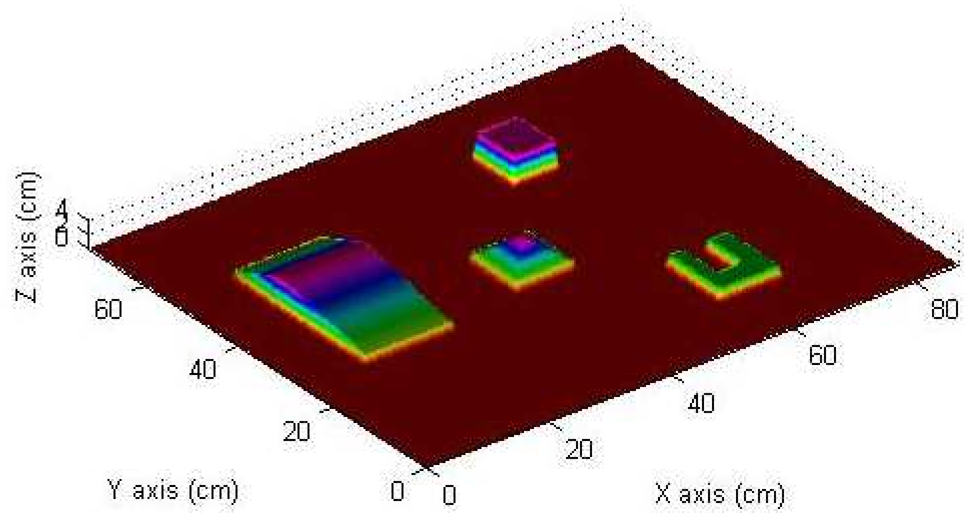
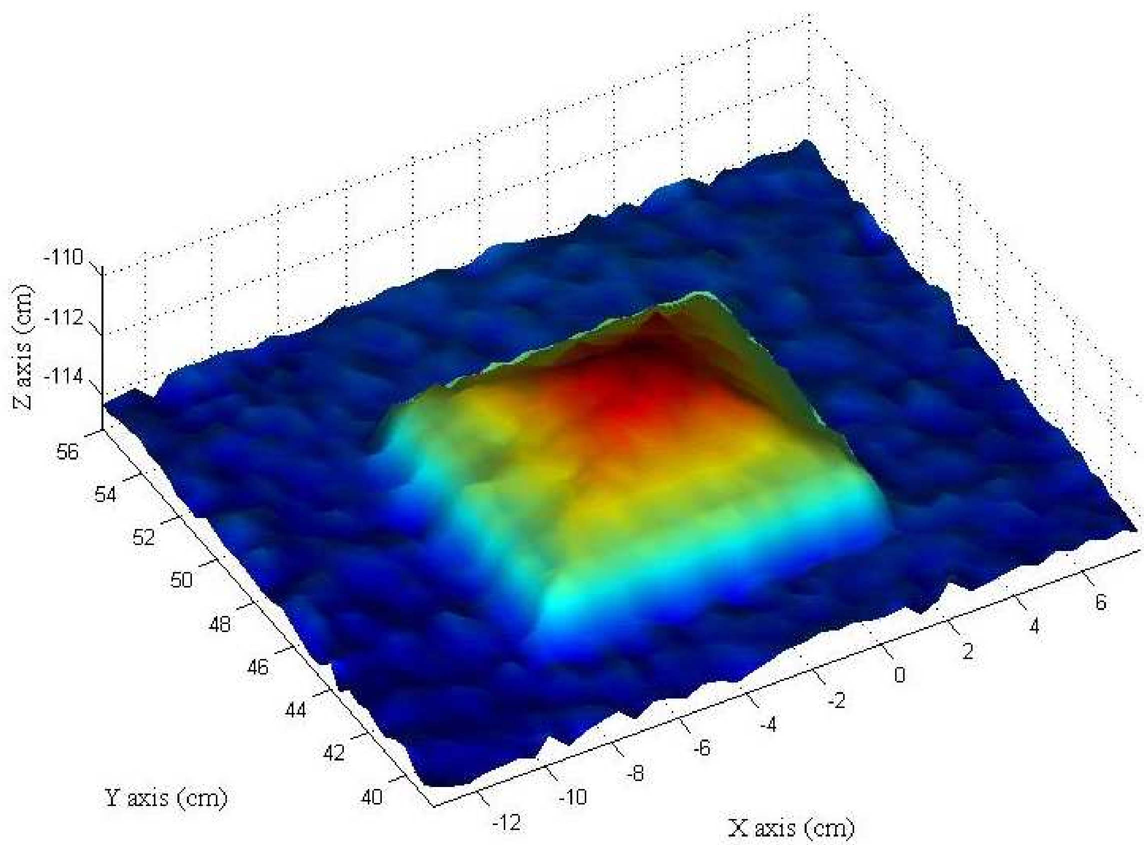



**Table 5.3 Correlation Coefficients between the Reference DEM and DEMs Acquired by the Laser System in Two Replications for Test Surface 2**

| | Reference | Replication 1 | Replication 2 |
|---|---|---|---|
| **Reference** | 1 | 0.7167 | 0.7199 |
| **Replication 1** | 0.7167 | 1 | 0.9585 |
| **Replication 2** | 0.7199 | 0.9585 | 1 |

R-values of about 0.72 were achieved between the reference DEM and the DEMs measured in two replications, indicating low accuracy of the laser system in elevation measurement. However, when comparing portions of the DEM that contained only individual objects, the R-values were much higher. Taking object B as an example, when only the portion of the DEM data that contained object B and surrounding area was compared (Figure 5.19), the R value reached 0.9679. R-values of 0.8633, 0.9493, and 0.9261 were achieved for other three objects, respectively. These results indicated that the laser system can describe individual objects more accurately than a flat surface.

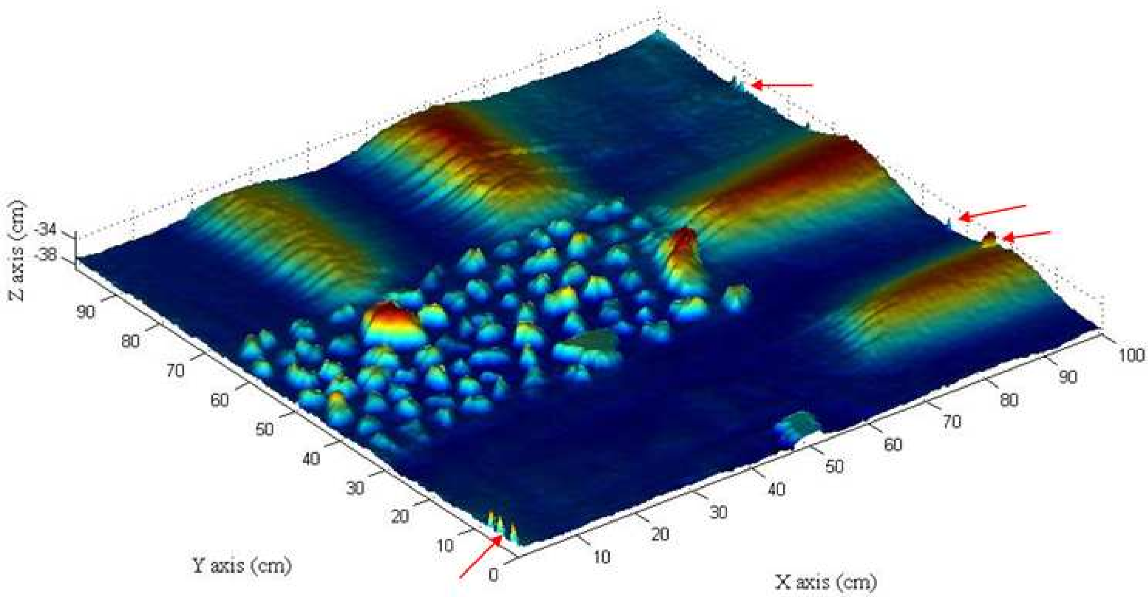
**Figure 5.19 DEM Derived for Object B on Test Surface 2**



#### *5.1.4.2 Comparison with the Reference System*

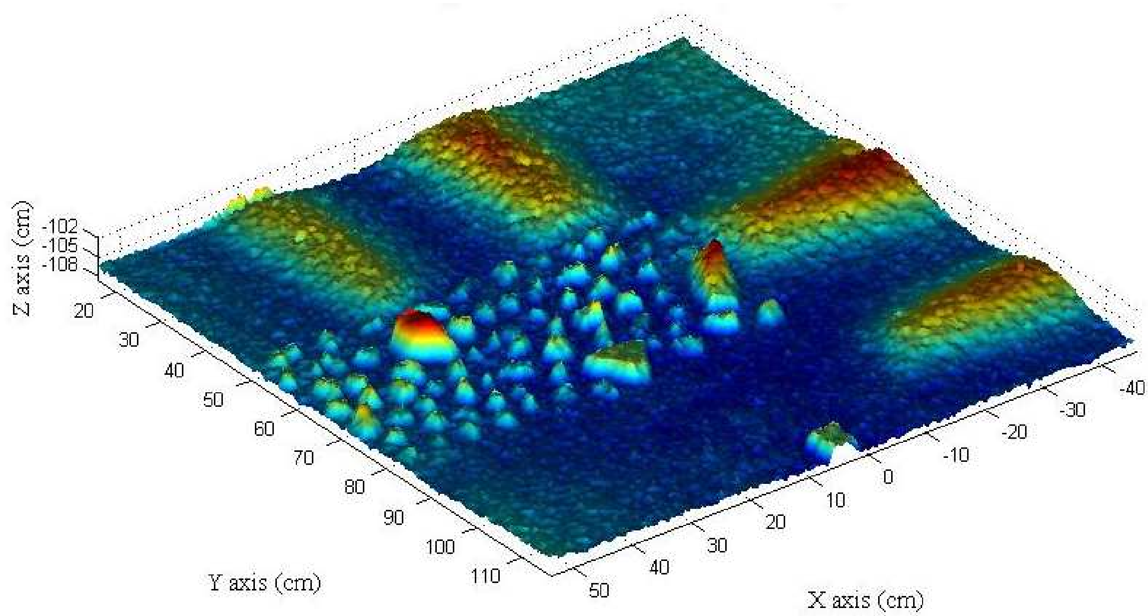
Figure 5.20 displays the sand-stone surface measured by the reference system. Measurement errors found along the boundaries of the surface, which are indicated by red arrows in Figure 5.20, were probably caused by the reflection of the wood frame.

**Figure 5.20 DEM of the Sand-stone Surface Measured by the Reference System**




The laser system spent approximately 17 seconds to scan the sand-stone surface. This derived DEM is shown in Figure 5.21.

**Figure 5.21 DEM of the Sand-stone Surface Measured by the Laser System**




Four small regions, a stones zone, and a small wooden block can be easily observed on both DEMs. To quantitatively evaluate the accuracy of the laser system, correlation coefficients between the reference and four replications of the measured DEMs was calculated (Table 5.4). The R-values between the reference and measured DEMs were greater than 0.925. The R-values among the four laser-measured DEMs were greater than 0.948.

**Table 5.4 Correlation Coefficients between DEMs Obtained by the Laser System in Four Replications and the Reference DEM**

| | Reference | Replication 1 | Replication 2 | Replication 3 | Replication 4 |
|---|---|---|---|---|---|
| **Reference** | 1 | 0.9314 | 0.9251 | 0.9274 | 0.9371 |
| **Replication 1** | 0.9314 | 1 | 0.9492 | 0.9481 | 0.9483 |
| **Replication 2** | 0.9251 | 0.9492 | 1 | 0.9614 | 0.9587 |
| **Replication 3** | 0.9274 | 0.9481 | 0.9614 | 1 | 0.9647 |
| **Replication 4** | 0.9371 | 0.9483 | 0.9587 | 0.9647 | 1 |

To further improve the measurement accuracy, the median threshold filter was applied to the raw laser data before interpolation (Figure 5.22). The filter's size was 1cm by 1cm. The upper and lower limits of the median value used in the filter were set to 1 cm and 0.5 cm, respectively. The filtered laser data was then interpolated to generate the DEM. Comparisons between the filtered DEMs and the reference DEM are given in Table 5.5. The R-values between the filtered DEMs and the reference DEM were greater than 0.93. The R-values among the four replications were greater than 0.955. In both cases, the comparisons were slightly improved.

**Figure 5.22 DEM of the Sand-stone Surface Filtered using the Median Threshold Filter**

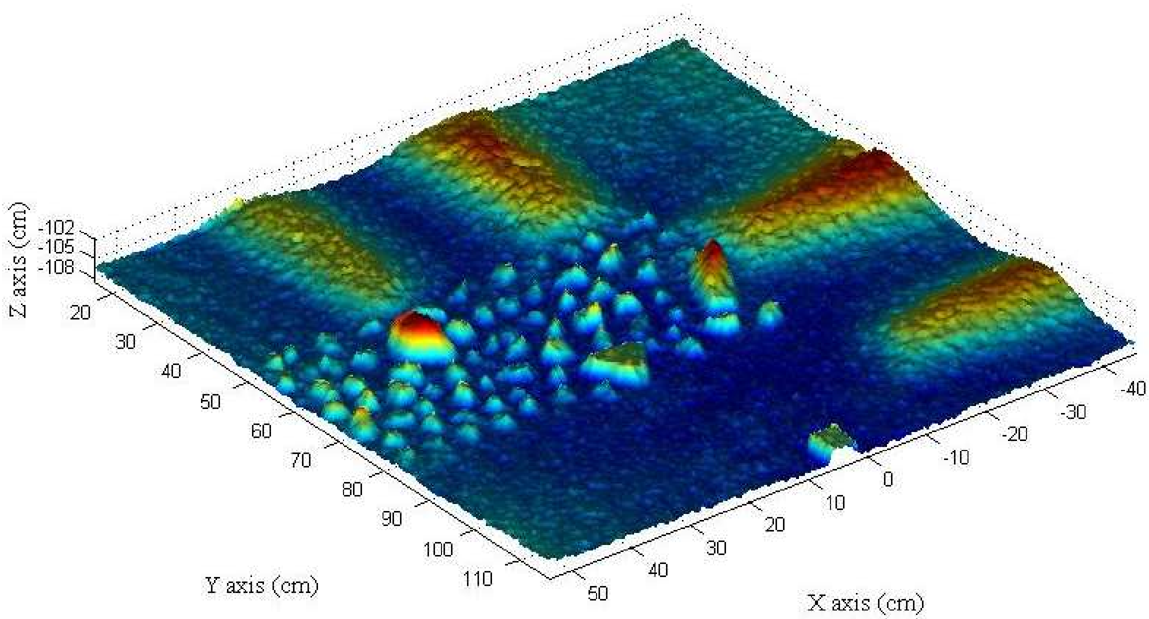



**Table 5.5 Correlation Coefficients between the DEMs filtered using the Median Threshold Filter and the Reference DEM**

| | **Reference** | **Replication 1** | **Replication 2** | **Replication 3** | **Replication 4** |
|---|---|---|---|---|---|
| **Reference** | 1 | 0.9412 | 0.9309 | 0.9318 | 0.9422 |
| **Replication 1** | 0.9412 | 1 | 0.9548 | 0.9552 | 0.9556 |
| **Replication 2** | 0.9309 | 0.9548 | 1 | 0.9648 | 0.9621 |
| **Replication 3** | 0.9318 | 0.9552 | 0.9648 | 1 | 0.9679 |
| **Replication 4** | 0.9422 | 0.9556 | 0.9621 | 0.9679 | 1 |

To further improve the result, the median filter was applied to the laser measured DEMs after interpolation. The filter was a 3-by-3 spatial filter. Figure 5.23 illustrated a DEM of the sand-stone surface measured by the laser system after the median filter was applied. The DEM in Figure 5.23 appeared more blurring than that shown in Figure 5.21. This was due to the removal of certain details by the filter. Table 5.6 shows the comparisons between the DEMs. The R-values between the DEMs obtained by the laser system and the reference DEM were greater than 0.935. The R-values among the four replications were greater than 0.988. The filter seemed to have slightly improved both accuracy and repeatability of the laser system, although its disadvantage in removing detailed information was fully awared of.

**Figure 5.23 DEM of the Sand-stone Surface Filtered using the Median Filter**

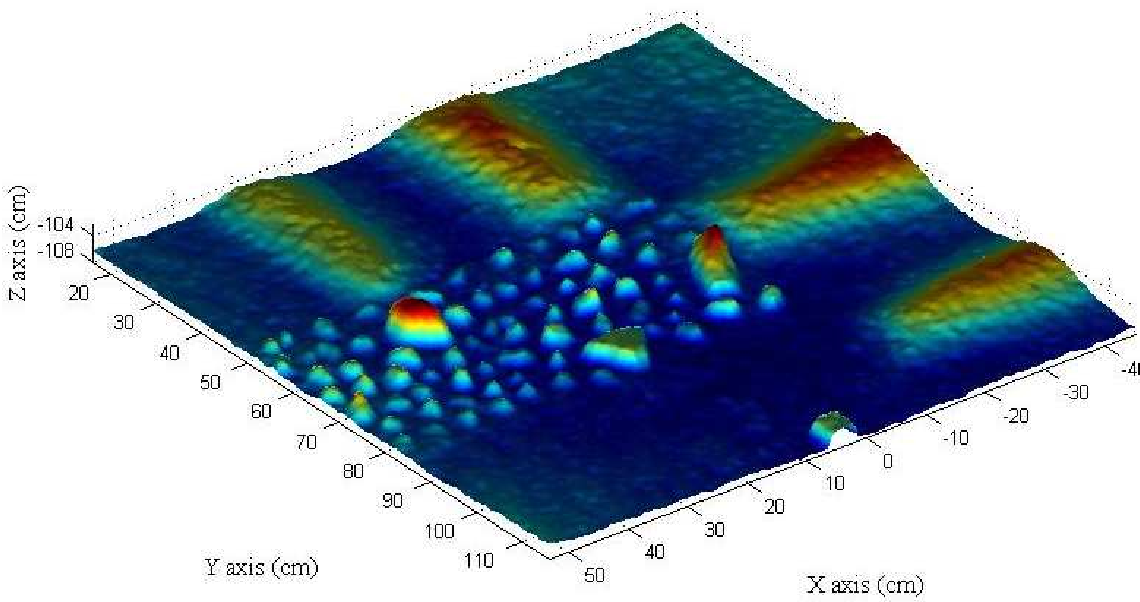



**Table 5.6 Correlation Coefficients between the DEMs Obtained by the Laser System and Processed using the Median Filter and the Reference DEM**

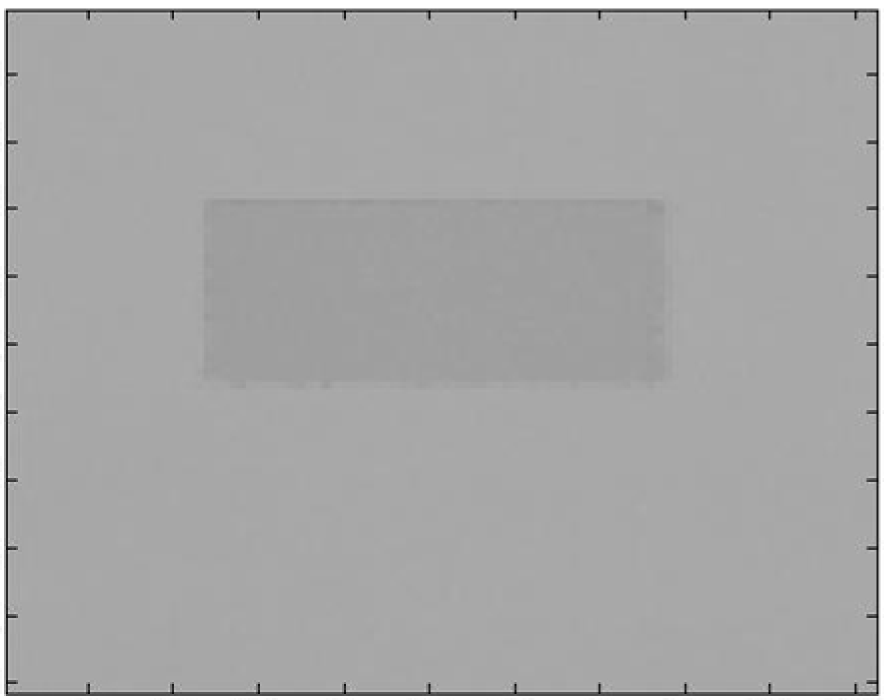
| | Reference | Replication 1 | Replication 2 | Replication 3 | Replication 4 |
|---|---|---|---|---|---|
| **Reference** | 1 | 0.9540 | 0.9393 | 0.9359 | 0.9459 |
| **Replication 1** | 0.9540 | 1 | 0.9891 | 0.9884 | 0.9885 |
| **Replication 2** | 0.9393 | 0.9891 | 1 | 0.9917 | 0.9886 |
| **Replication 3** | 0.9359 | 0.9884 | 0.9917 | 1 | 0.9916 |
| **Replication 4** | 0.9459 | 0.9885 | 0.9886 | 0.9916 | 1 |

In summary, the comparison between DEMs obtained by the laser system and the reference system showed that the laser system generated the DEM for a sand-stone surface with satisfactory accuracy and repeatability. The accuracy was further improved when a median threshold filter and a median filter were used to process the raw laser data before and after the interpolation, respectively.

#### *5.1.4.3 Use of gray-scale data*

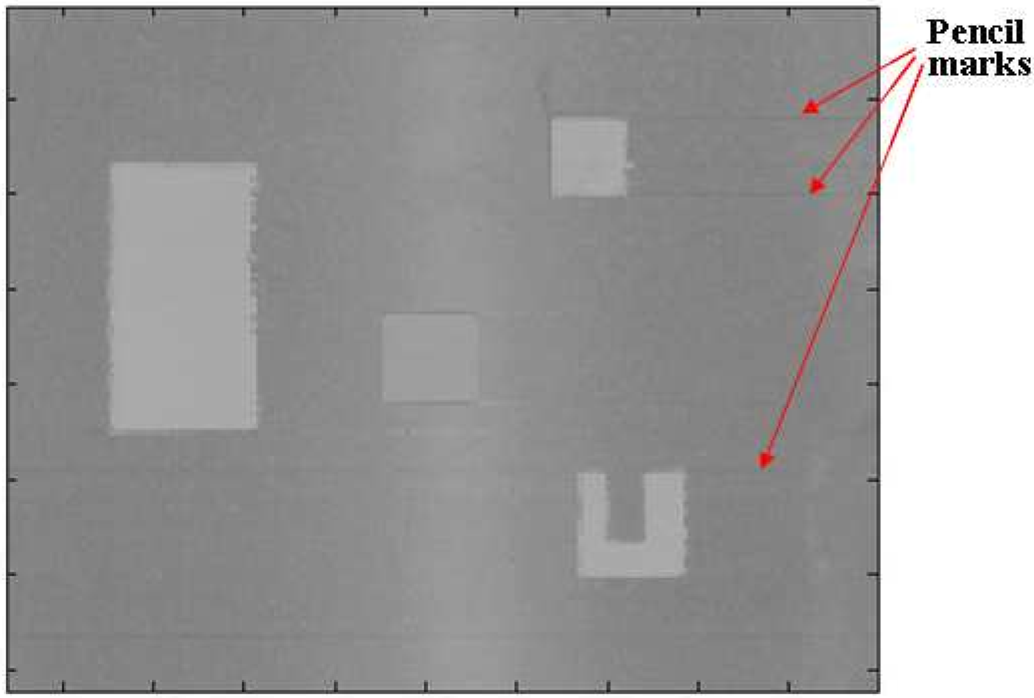
Grayscale image of test surface 1, test surface 2, and the sand-stone surface are displayed in Figures 5.24, 5.25, and 5.26, respectively. For the box, the gray levels of the paper background and the box were similar. This was because the background and the box were made of similar materials.

**Figure 5.24 Grayscale Image of Test Surface 1**



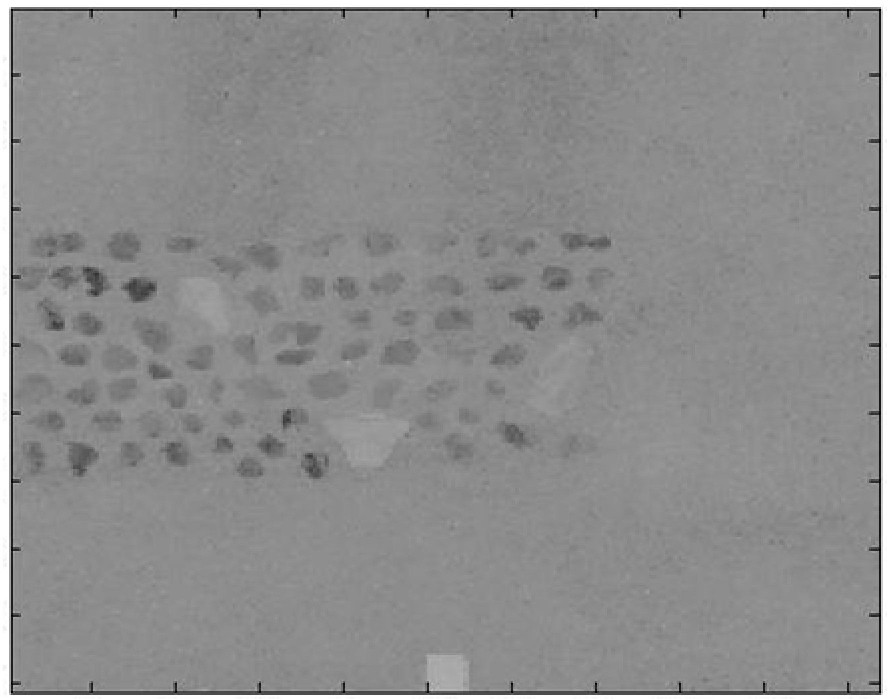
In Figure 5.25, four objects can be easily distinguished from the background. However, it was difficult to identify the shapes. A vertical strip was clearly shown in the image. It may have been caused by the shadow of the linear rail. Several horizontal pencil marks clearly appeared on the grayscale image, which were never shown on the DEM surfaces.

**Figure 5.25 Grayscale Image of Test Surface 2**




For the sand-stone surface, stones and wooden block can be easily observed from the grayscale image. Darker stones appeared more clearly in the image. However, the four sand regions were not distinguishable from the sand background.

**Figure 5.26 Grayscale Image of the Sand-Stone Surface**



In summary, differences in materials and colors affected the gray-scale image. This made it possible to identify crop residues from soil surface in the field.

## 5.2 Noise Tests

### *5.2.1 Description*

There were two major sources of noise that affected the laser measurement. The first was the ambient light. Optical sensors were sensitive to ambient light. For indoor experiments, fluorescent light was the major source that may affect the laser sensor. In the field, sunlight usually had the strongest effect on optical sensors. In some cases, the laser signal received from the surface may be overwhelmed by the strong daylight.

The second source of noise was the mechanical vibration of the laser sensor, which was caused by the laser-carriage movement and the laser mirror rotation. If there were roll or pitch angular changes on the rail during scanning, the measured elevation would be affected.

This test focused on the effects of these two noise sources on the elevation measurement. In the ambient light test, the DEMs of the sand-stone surface measured under different indoor and outdoor ambient light conditions were compared. Angular displacements recorded by the gyroscope sensor during scanning were analyzed to study the effect of mechanical vibration.

### *5.2.2 Objectives*

The objectives of the noise tests were:

1). to study the effect of ambient light on elevation measurement in both indoor and outdoor environments;

2). to investigate the potential effect of rail vibration on the angular displacement of the linear rail.

### *5.2.3 Methodology*

#### *5.2.3.1 Ambient Light Effect*

##### *5.2.3.1.1 Indoor Ambient Light*

For the indoor test, fluorescent lamps, Philips cool white F32T8/TL841/ALTO, were used as the light source. The sand-stone surface was measured by the laser system at four different times: 9:30AM, 1:30PM, 8:30PM, and 11:30PM. At each time, measurements were taken with and without the fluorescent light on, respectively. For all scans, the laser line scanner was configured at a sample rate of 40 KHz, a mirror rotation speed of 1400 RPM, and a max range setting of 183 cm. The laser-carriage moved at a speed of 6.31 cm/s. At this speed, each scan was completed within 17 seconds. DEMs data derived were observed at different lighting conditions. Correlation coefficients among the measured DEMs were compared.

#### *5.2.3.1.2 Sunlight*

In order to investigate the effect of sunlight on the measurement, an outdoor test was designed (Figure 5.27). A sand-stone surface was constructed (Figure 5.28) and was measured by the laser system at four different times - 10:30AM, 1:30PM, 3:30PM, and 6:00PM - in December, 2009. Two replications were taken at each time. For these measurements, the laser line scanner used a sample rate of 40 KHz, a mirror rotation speed of 1400 RPM, and a max range setting of 183 cm. The laser-carriage moved at a low speed of 6.31 cm/s. DEM data of the measured surface were plotted. Correlation coefficients among the DEM data were calculated.

**Figure 5.27 Outdoor Test on the Effect of Sunlight on Elevation Measurement**

**Figure 5.28 The Sand-stone Surface Used in the Outdoor Test**

### *5.2.3.2 Rail Vibration Effect*

To investigate the effect of rail vibration, roll, pitch, and yaw angles measured by the gyroscope sensor were recorded under the following conditions:

1). The laser-carriage moved at three different speeds on the rail while the laser mirror rotated at three different speeds;

2). The laser-carriage moved at three different speeds while the laser mirror did not rotate;

3). The laser mirror rotated when the laser-carriage was situated at the starting and ending positions on the rail, respectively;

4). The laser-carriage moved at three different speeds on the rail while the laser mirror rotated at three different speeds, when the rail was fixed at both ends.

The three speeds at which the laser-carriage moved were 6.31cm/s (“low”), 9.32cm/s (“medium”), and 14.63cm/s (“high”). The three rotational speeds of the laser mirror were 1400 RPM, 2000 RPM, and 2600 RPM. The “starting” position was located at 66 cm from the rear-end of the rail. The “ending” position was 120 cm away from the starting position, along the rail. The distance between the center of the gyroscope sensor and the starting position was 51 cm. Roll, pitch and yaw angles were measured by the gyroscope sensor during all the tests.

### *5.2.4 Results and Discussion*

#### *5.2.4.1 Ambient Light Effect*

For this experiment, correlation coefficients were calculated among DEMs of the sand-stone surface measured under various light conditions. The DEMs were generated through the following steps:

1). Create 3D raw data based on the 3D coordinate conversion algorithm (Section 5.1.3.1.2);

2). Apply the median threshold filter (Section 5.1.3.2.5) to the 3D raw data;

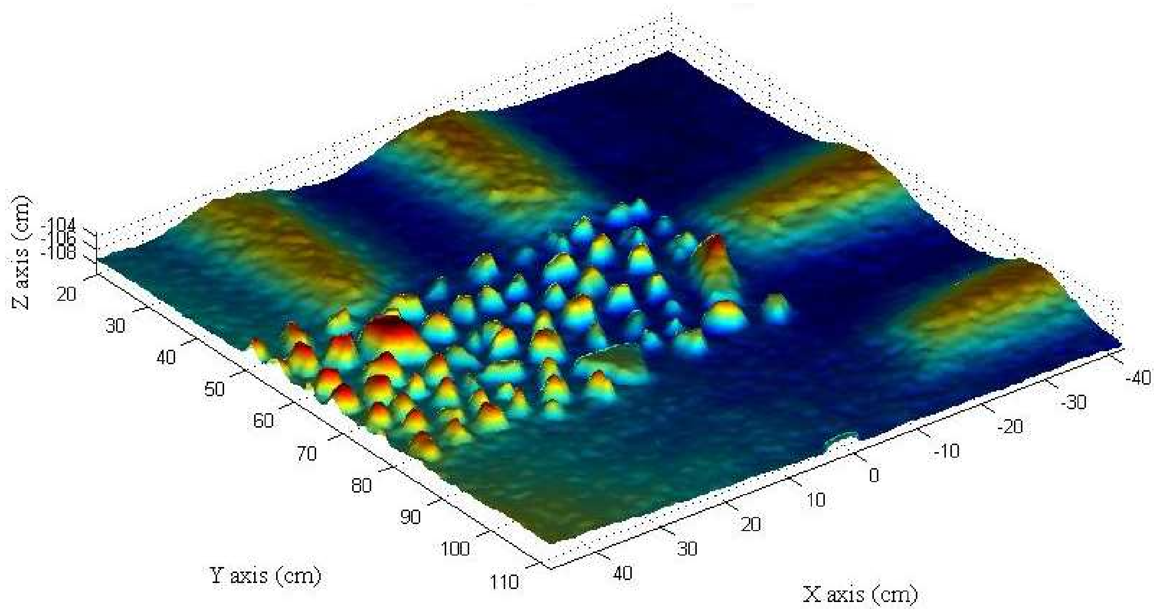
3). Interpolate the filtered 3D data using the interpolation algorithm (Section 5.1.3.1.3);

4). Apply the 3-by-3 median filter (Section 5.1.3.2.5) to derive the final DEM.

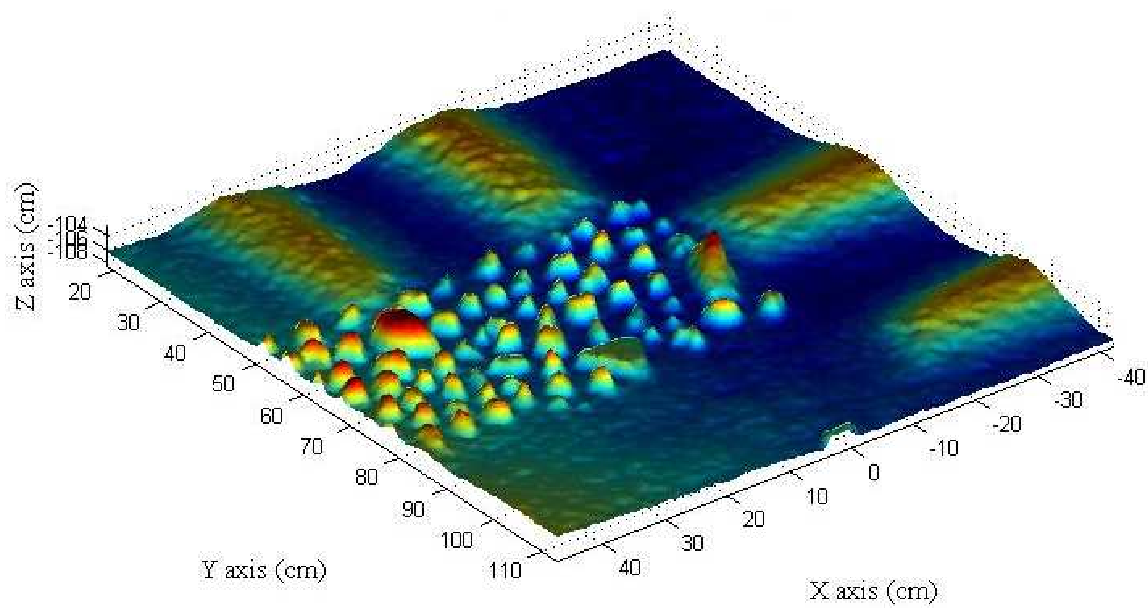
##### *5.2.4.1.1 Indoor Ambient Light*

The DEMs of the sand-stone surface derived under different light conditions are showed in Figures 5.29 through 5.32. R-values among these DEMs are listed in Table 5.7. It can be observed that all the R-values were greater than 0.98, indicating that the fluorescent light did not affect the elevation measurement of the laser system.

**Figure 5.29 DEMs of the Sand-stone Surface Measured under Different Indoor Lighting Conditions (a) 9:30AM with Fluorescent Light, and (b) 9:30AM without Fluoresecent Light**

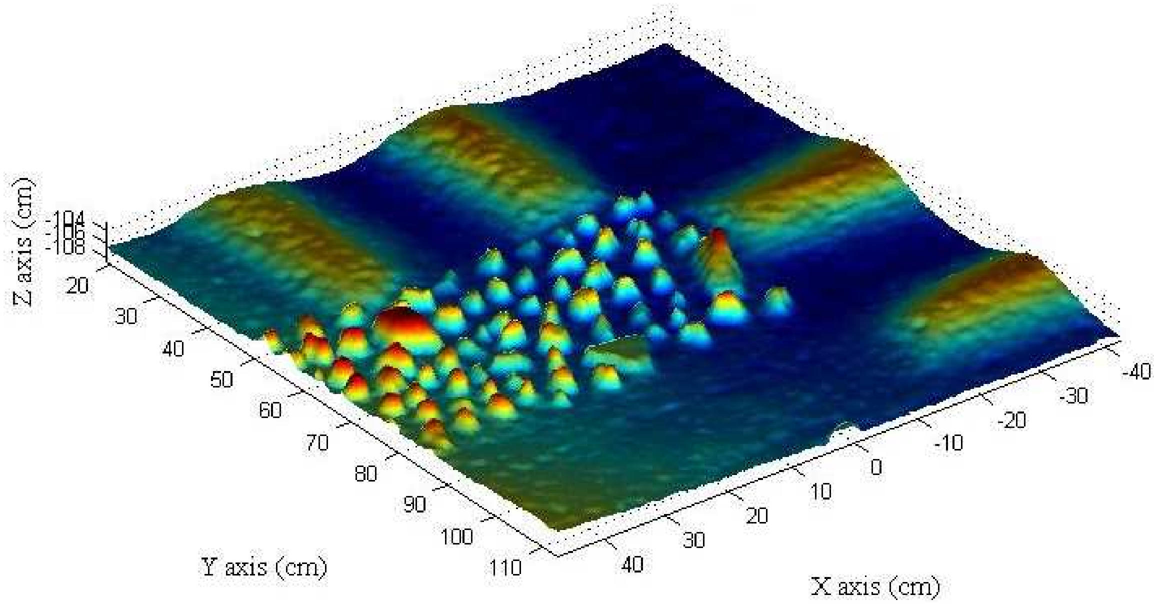



**(a)**

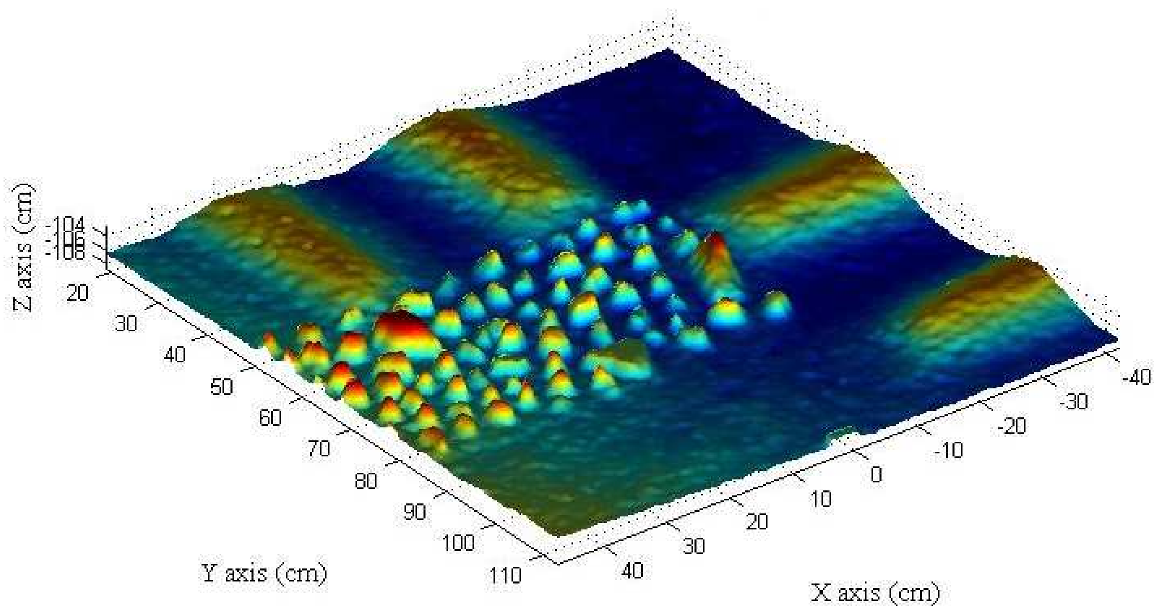



**(b)**

**Figure 5.30 DEMs of the Sand-stone Surface Measured under Different Indoor Lighting Conditions (a) 1:30PM with Fluorescent Light, and (b) 1:30PM without Fluorescent Light**

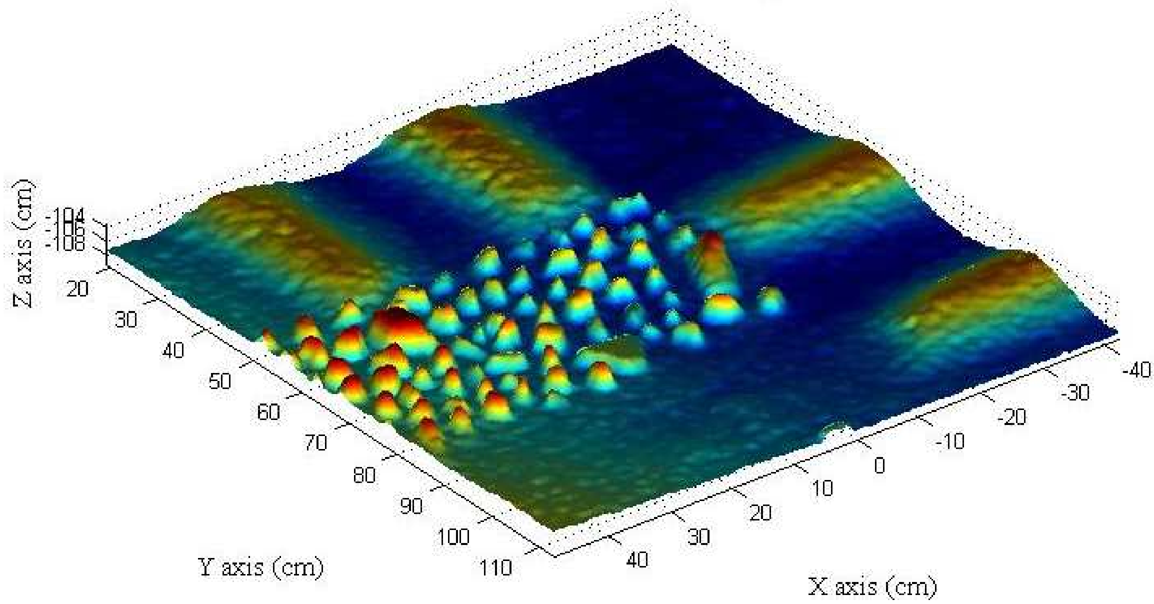



**(a)**

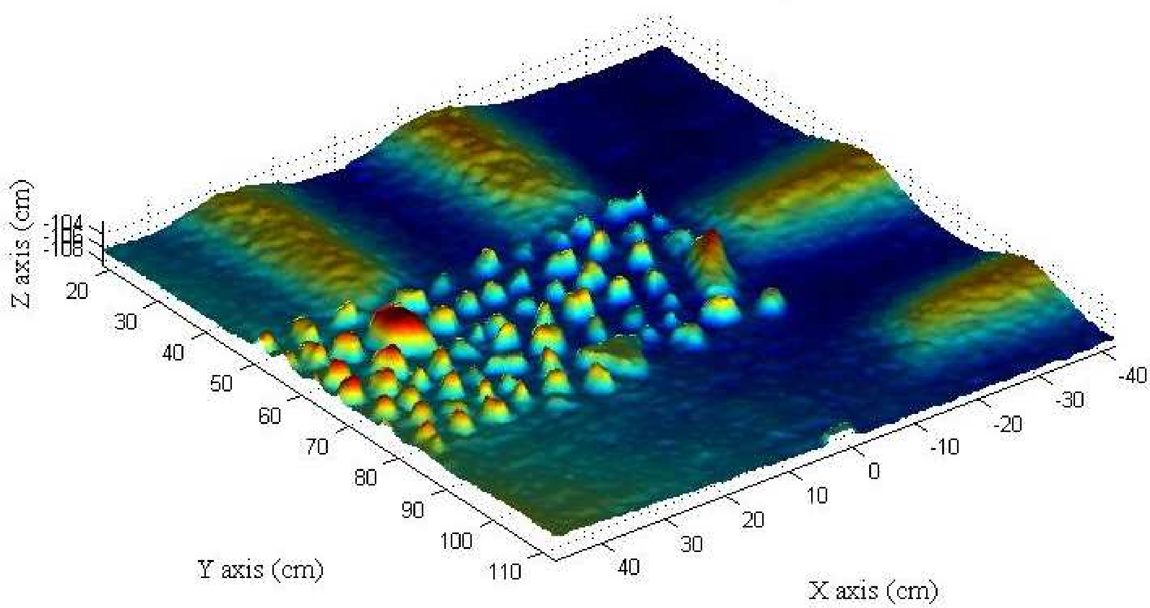



**(b)**

**Figure 5.31 DEMs of the Sand-stone Surface Measured under Different Indoor Lighting Conditions (a) 8:30PM with Fluorescent Light, and (b) 8:30PM without Fluorescent Light**

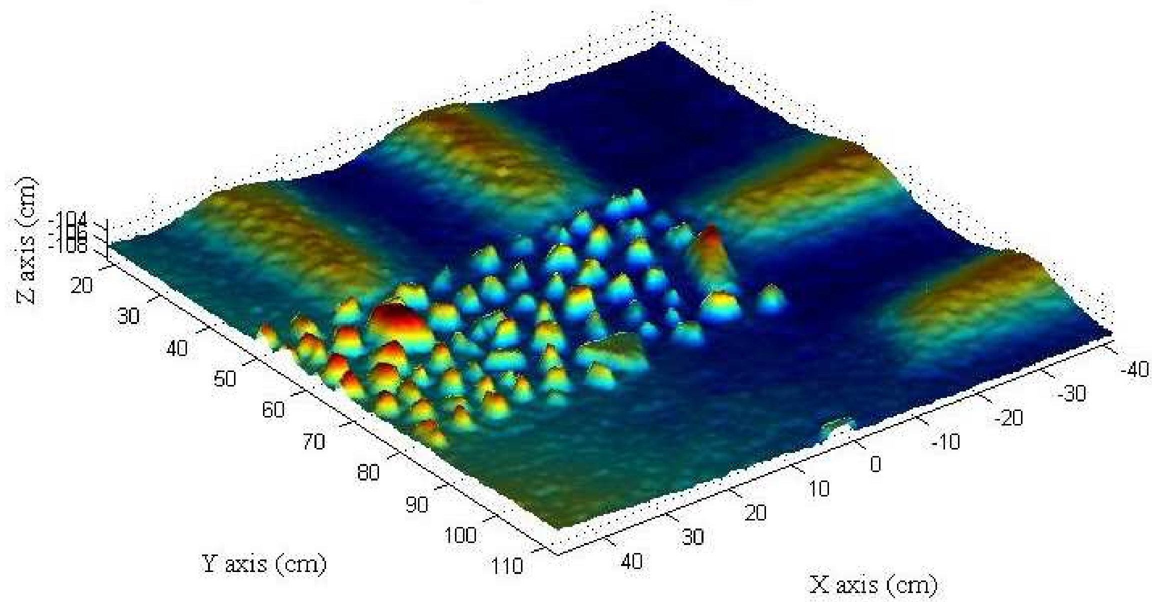



**(a)**

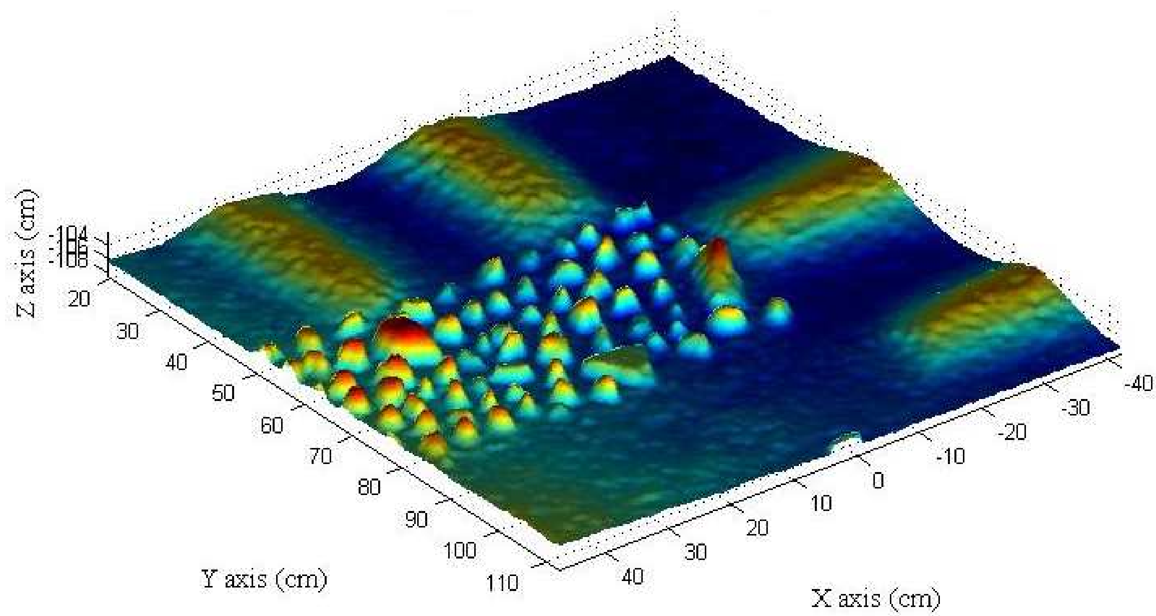



**(b)**

**Figure 5.32 DEMs of the Sand-stone Surface Measured under Different Indoor Lighting Conditions (a) 11:30PM with Fluorescent Light, and (b) 11:30PM without Fluorescent Light**




**(a)**




**(b)**

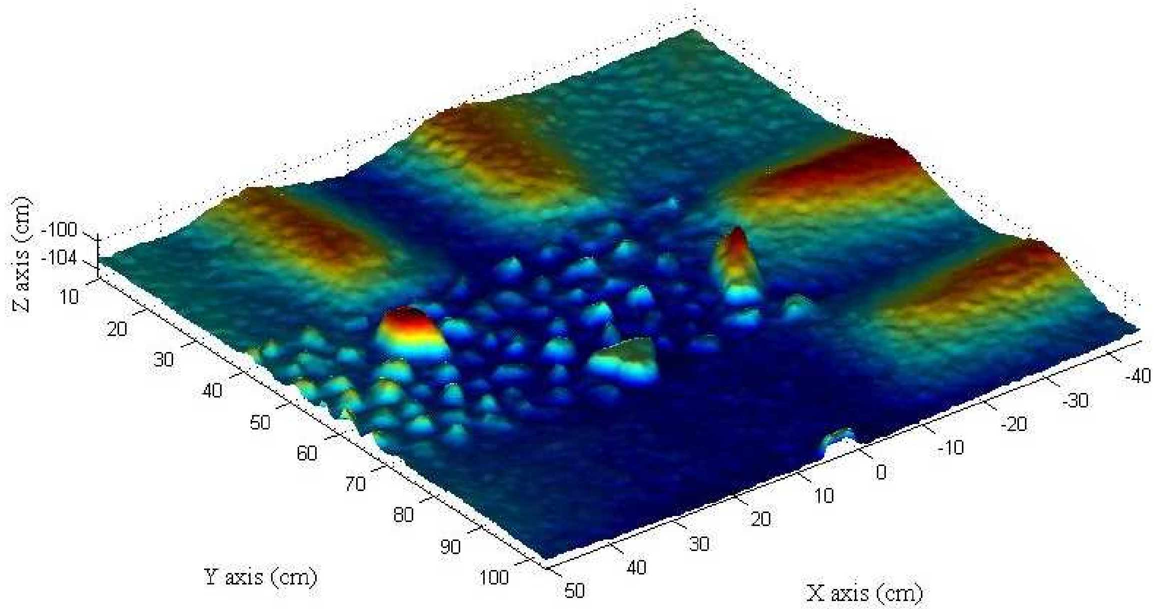
**Table 5.7 Correlation Coefficients among DEMs Derived during the Indoor Tests Conducted at Different Times in a Day and under Different Lighting Conditions**

| | | 9:30 AM | | 1:30 PM | | 8:30 PM | | 11:30 PM | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Light on** | **Light off** | **Light on** | **Light off** | **Light on** | **Light off** | **Light on** | **Light off** |
| **9:30 AM** | **Light on** | 1 | 0.9898 | 0.9880 | 0.9902 | 0.9886 | 0.9913 | 0.9877 | 0.9898 |
| | **Light off** | 0.9898 | 1 | 0.9903 | 0.9892 | 0.9888 | 0.9929 | 0.9910 | 0.9931 |
| **1:30 PM** | **Light on** | 0.9880 | 0.9903 | 1 | 0.9862 | 0.9825 | 0,9885 | 0.9927 | 0.9905 |
| | **Light off** | 0.9902 | 0.9892 | 0.9862 | 1 | 0.9874 | 0.9915 | 0.9885 | 0.9898 |
| **8:30 PM** | **Light on** | 0.9886 | 0.9888 | 0.9825 | 0.9874 | 1 | 0.9911 | 0.9890 | 0.9910 |
| | **Light off** | 0.9913 | 0.9929 | 0.9885 | 0.9915 | 0.9911 | 1 | 0.9892 | 0.9931 |
| **11:30 PM** | **Light on** | 0.9877 | 0.9910 | 0.9927 | 0.9885 | 0.9890 | 0.9892 | 1 | 0.9907 |
| | **Light off** | 0.9898 | 0.9931 | 0.9905 | 0.9898 | 0.9910 | 0.9931 | 0.9907 | 1 |

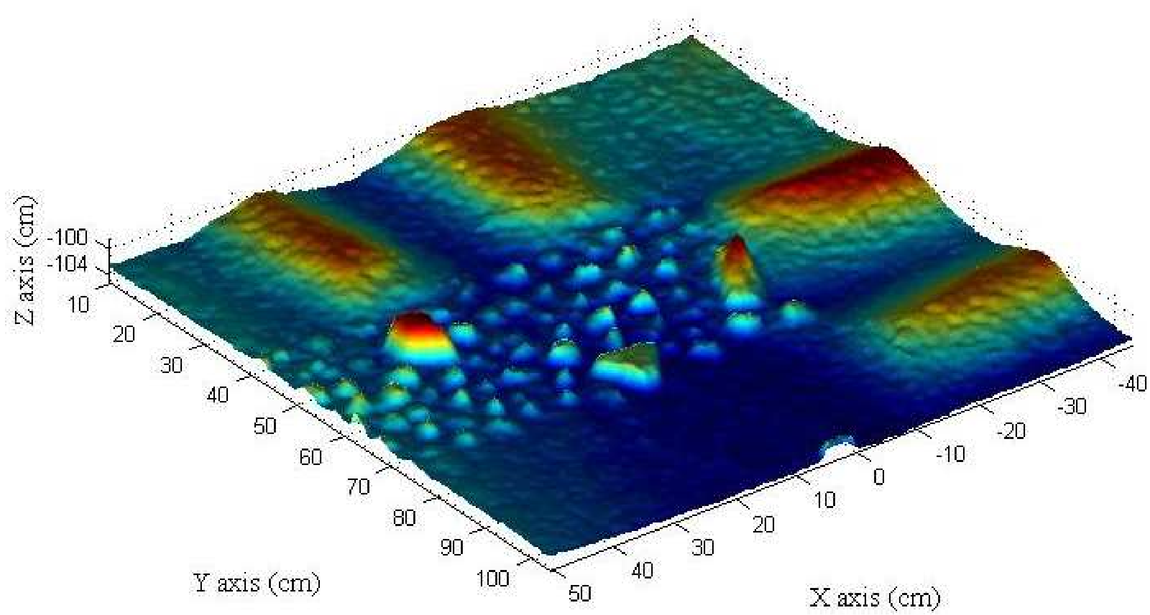
#### *5.2.4.1.2 Sunlight*

The DEMs of the sand-stone surface measured at different times of the day are displayed in Figures 5.33 and 5.34. This surface was slight different from the one used in the indoor test. R-values among these DEMs are listed in Table 5.8. It can be observed that all R-values were greater than 0.97, indicating that the sunlight did not affect the elevation measurement of the laser system.

**Figure 5.33 DEMs of the Sand-stone Surface Measured under Sunlight on December 17, 2009: (a) 10:30 AM, and (b) 1:30 PM**

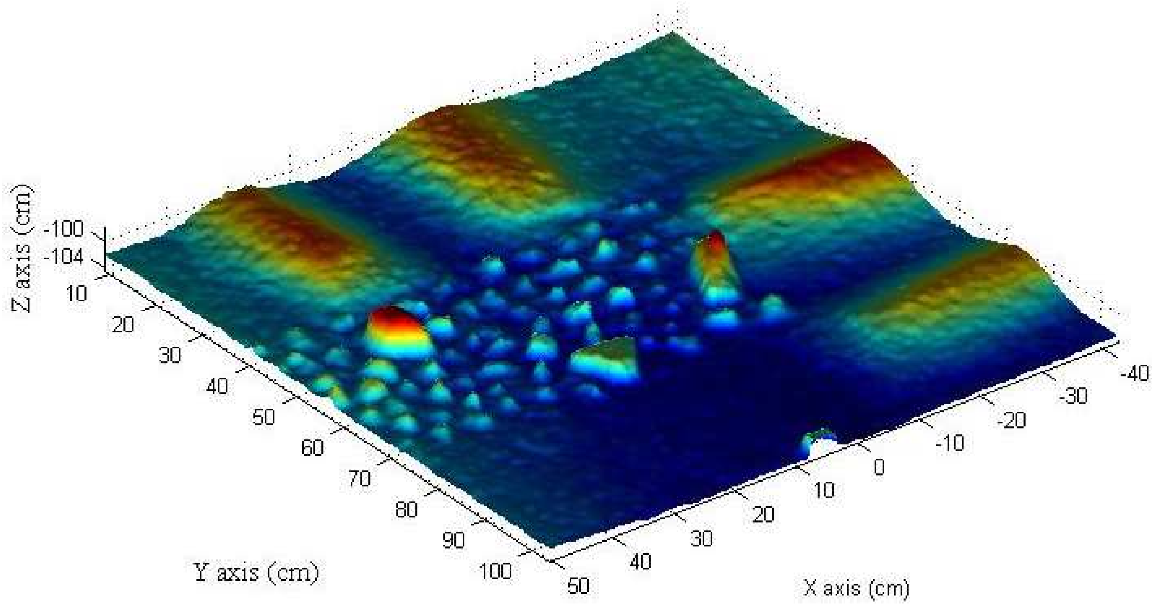



**(a)**

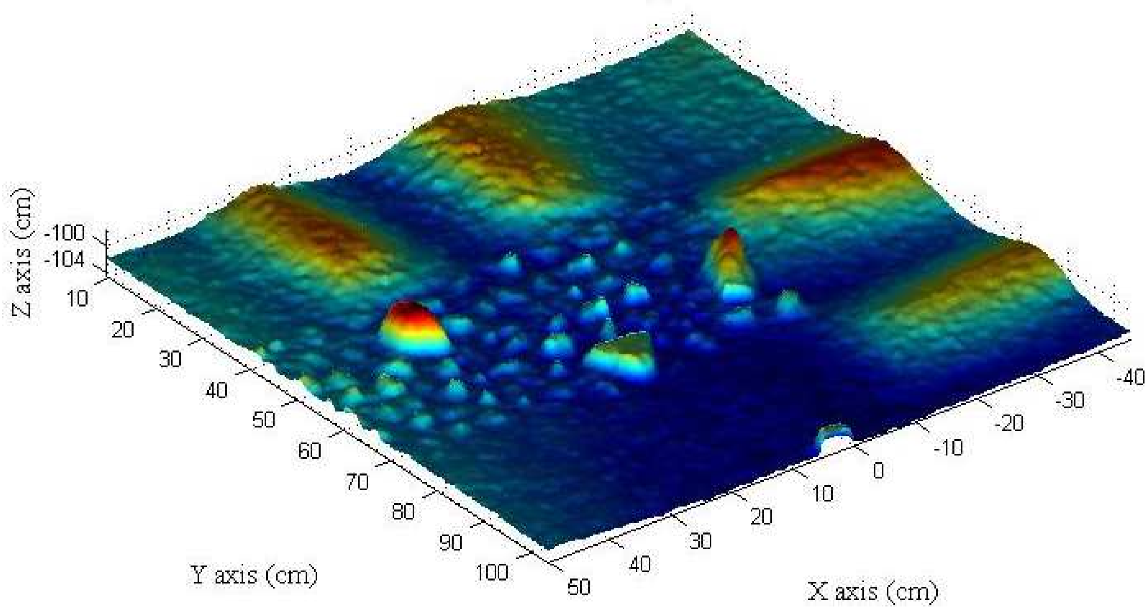



**(b)**

**Figure 5.34 DEMs of the Sand-stone Surface Measured under Sunlight on December 17, 2009: (a) 3:30 PM, and (b) 6:00 PM**




**(a)**




**(b)**

**Table 5.8 Correlation Coefficients among DEMs Measured at Different Times of the Day under Sunlight on December 17, 2009**

| | | 10:30 AM | | 1:30 PM | | 3:30 PM | | 6:00 PM | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Replication 1** | **Replication 2** | **Replication 1** | **Replication 2** | **Replication 1** | **Replication 2** | **Replication 1** | **Replication 2** |
| **10:30 AM** | **Replication 1** | 1 | 0.9790 | 0.9843 | 0.9791 | 0.9724 | 0.9739 | 0.9748 | 0.9746 |
| | **Replication 2** | 0.9790 | 1 | 0.9726 | 0.9778 | 0.9763 | 0.9732 | 0.9767 | 0.9766 |
| **1:30 PM** | **Replication 1** | 0.9843 | 0.9726 | 1 | 0.9718 | 0.9780 | 0.9872 | 0.9785 | 0.9876 |
| | **Replication 2** | 0.9791 | 0.9778 | 0.9718 | 1 | 0.9751 | 0.9883 | 0.9798 | 0.9829 |
| **3:30 PM** | **Replication 1** | 0.9724 | 0.9763 | 0.9780 | 0.9751 | 1 | 0.9810 | 0.9942 | 0.9854 |
| | **Replication 2** | 0.9739 | 0.9732 | 0.9872 | 0.9883 | 0.9810 | 1 | 0.9768 | 0.9920 |
| **6:00 PM** | **Replication 1** | 0.9748 | 0.9767 | 0.9785 | 0.9798 | 0.9942 | 0.9768 | 1 | 0.9823 |
| | **Replication 2** | 0.9746 | 0.9766 | 0.9876 | 0.9829 | 0.9854 | 0.9920 | 0.9823 | 1 |

In summary, results of the ambient light test indicated that neither sunlight nor fluorescent light affected the measurement of the laser system. The system provided consistent elevation measurements under both indoor and outdoor lighting conditions.

#### ***5.2.4.2 Rail Vibration Effect***

Angular displacements recorded by the gyroscope sensor were plotted and compared under four conditions.

*Condition A: The laser-carriage moved at three different speeds on the rail while the laser mirror rotated at three different speeds* (Figure 5.35 ~ Figure 5.37).

**Figure 5.35 Roll Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds**

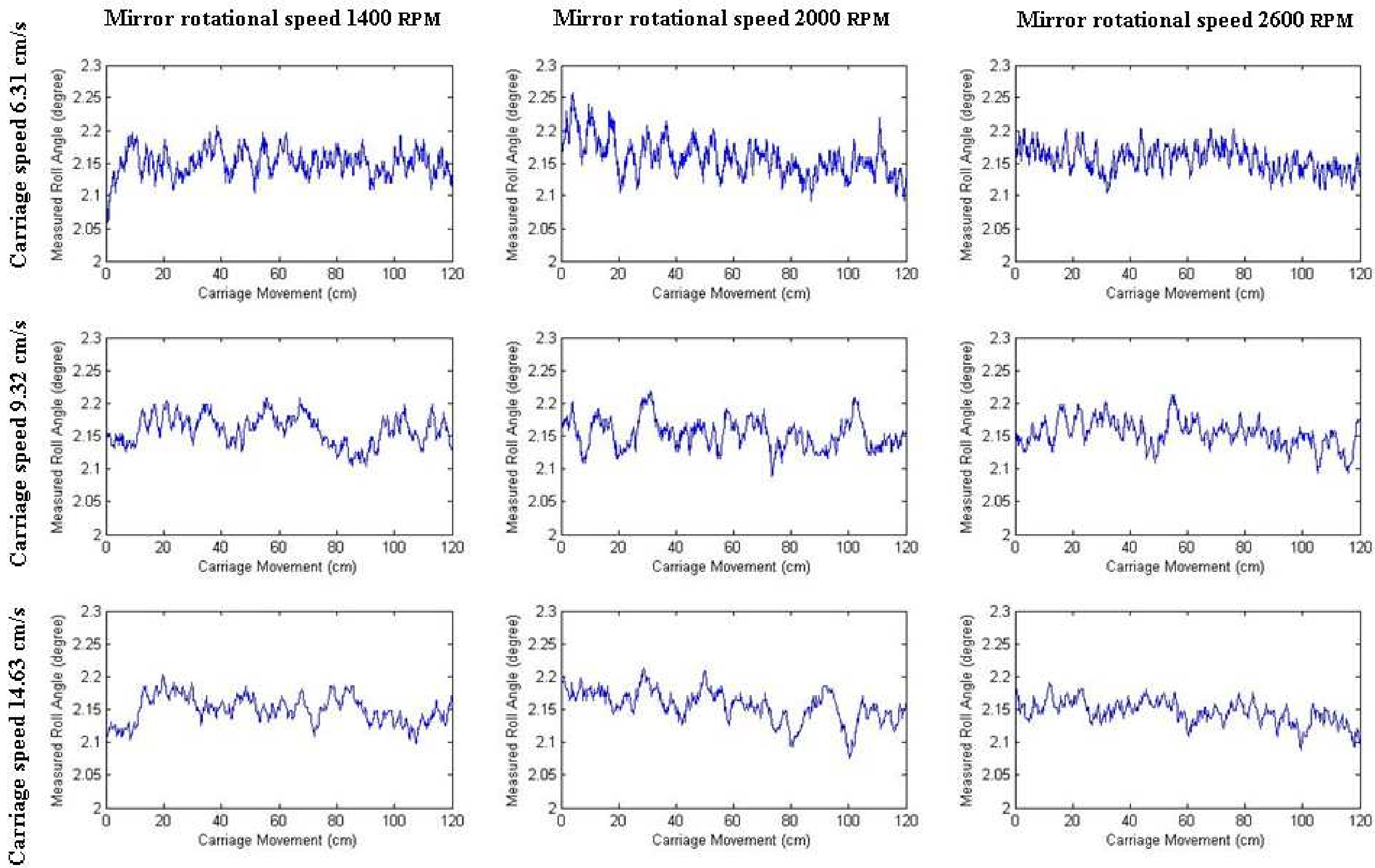


**Figure 5.36 Pitch Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds**

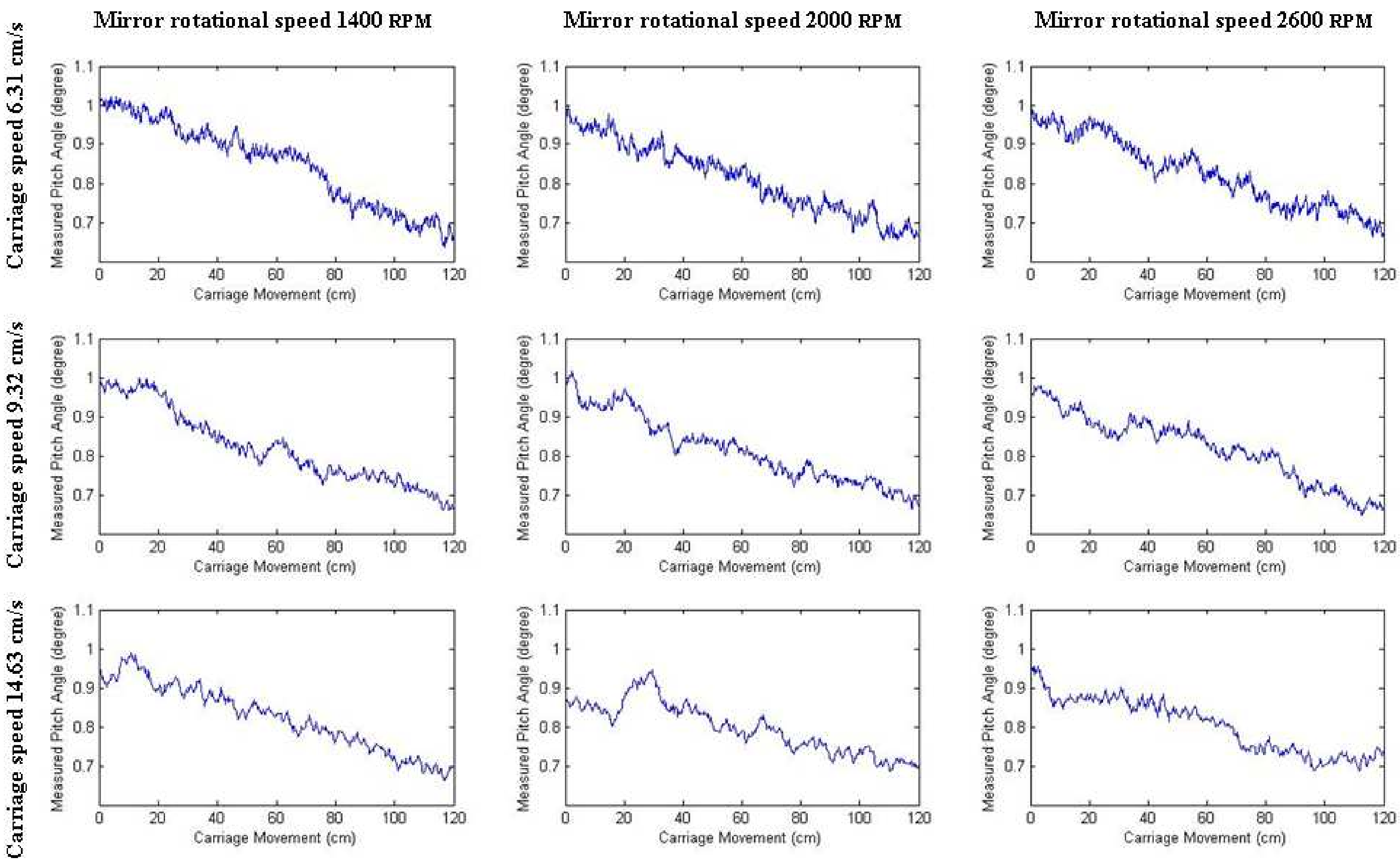
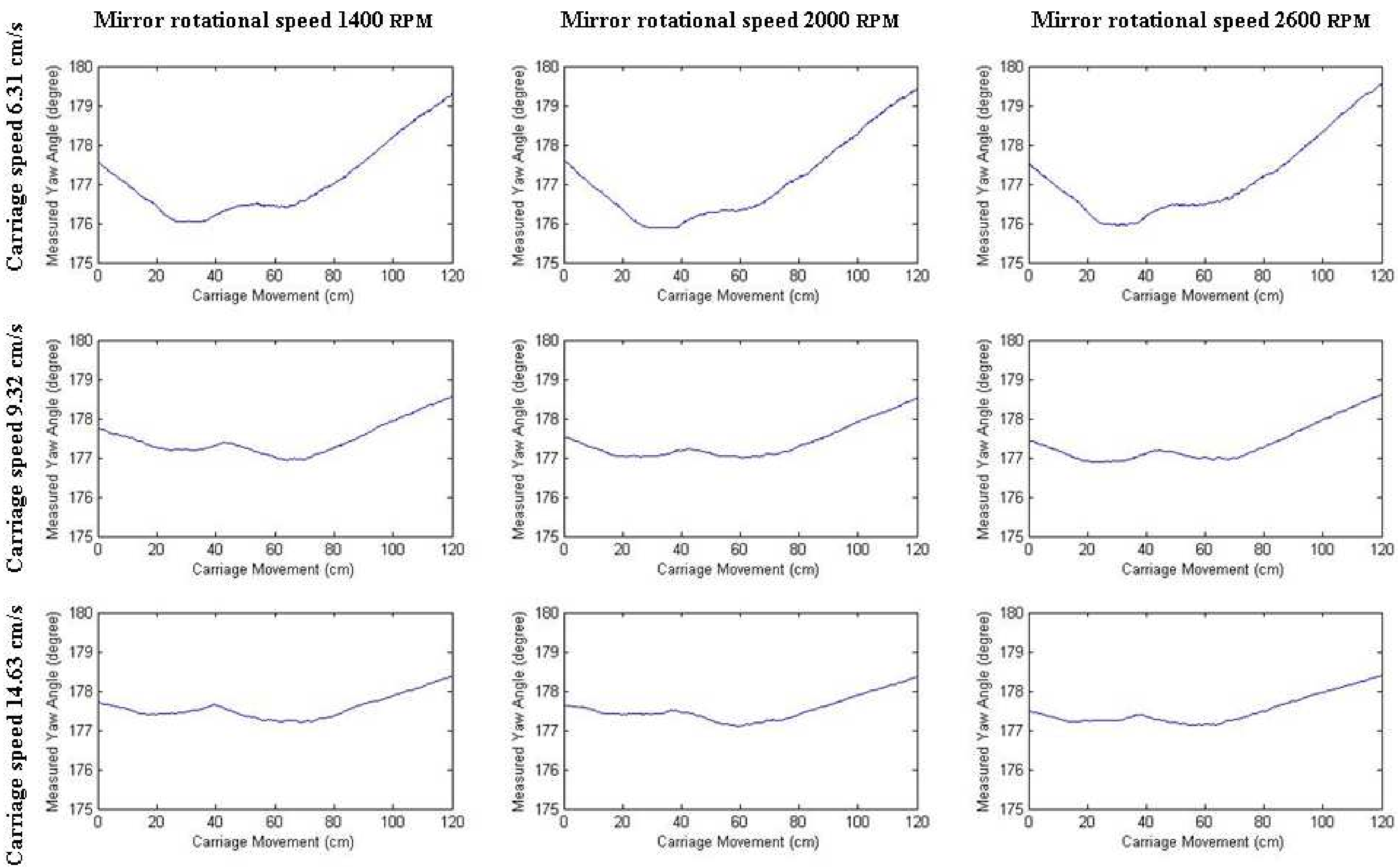


**Figure 5.37 Yaw Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds**



In order to study the impact of the mirror rotation, carriage movement, and their interaction on the roll angle measurement, the “mixed” procedure in SAS (SAS, 2002) (Appendix I) was used to analyze the roll angles measured under condition A, Carriage movement speed (CS), mirror rotation speed (MR), and interactions (CS*MR) at combinations of three mirror rotational speeds and three carriage movement speeds were considered as the fixed effects. The unequal numbers of replicates in the combinations were also considered as a fixed effect. A least squares analysis of a factorial design with unequal numbers of replicates at factor combinations was carried out. The results are summarized in Table 5.9.

**Table 5.9 Results of Statistics Analysis of the Measured Roll Angle under Condition A**

| **Effect** | **CS levels** | **MR levels** | **Estimate** (degree) | **Standard Error** (degree) | **Pr > \|t\|** |
|---|---|---|---|---|---|
| CS | 1 | | 2.1592 | 0.000424 | <0.0001 |
| CS | 2 | | 2.1527 | 0.000550 | <0.0001 |
| CS | 3 | | 2.1453 | 0.000658 | <0.0001 |
| MR | | 1 | 2.1520 | 0.000532 | <0.0001 |
| MR | | 2 | 2.1545 | 0.000525 | <0.0001 |
| MR | | 3 | 2.1506 | 0.000531 | <0.0001 |
| CS*MR | 1 | 1 | 2.1547 | 0.000706 | <0.0001 |
| CS*MR | 1 | 2 | 2.1627 | 0.000700 | <0.0001 |
| CS*MR | 1 | 3 | 2.1602 | 0.000700 | <0.0001 |
| CS*MR | 2 | 1 | 2.1565 | 0.000944 | <0.0001 |
| CS*MR | 2 | 2 | 2.1503 | 0.000931 | <0.0001 |
| CS*MR | 2 | 3 | 2.1512 | 0.000932 | <0.0001 |
| CS*MR | 3 | 1 | 2.1446 | 0.001096 | <0.0001 |
| CS*MR | 3 | 2 | 2.1506 | 0.001087 | <0.0001 |
| CS*MR | 3 | 3 | 2.1405 | 0.001114 | <0.0001 |

The carriage movement speed, mirror rotation speed, and their interactions were found to have statistically significant effects on the measured roll angles, $P<0.0001$, when the significance level $\alpha=0.05$. However, the standard errors at the combinations were less than 0.0012, indicating that these effects were not practical significant. In fact, at almost all the combinations of the carriage speeds and mirror rotational speeds, the measured roll angles were within the range of 2.09~2.26 degree.

From Figure 5.36, it is obvious that the pitch angle gradually decreased when the laser-carriage moved from the starting position towards the ending position. The reduction range was 0.38°. The yaw angle decreased when the laser-carriage moved from the starting position to 30cm and increased from approximate 70cm to the ending position. There was a variation on the yaw angles when the laser-carriage moved from 30cm towards 70cm (Figure 5.37).

*Condition B: The laser-carriage moved at three different speeds while the laser mirror did not rotate* (Figure 5.38 ~ Figure 5.40).

**Figure 5.38 Roll Angle Measured by the Gyroscope Sensor at Laser-carriage Movement Speeds of (a) 6.31cm/s, (b) 9.32 cm/s, and (c) 14.63 cm/s, while the Laser Mirror was not Rotating**

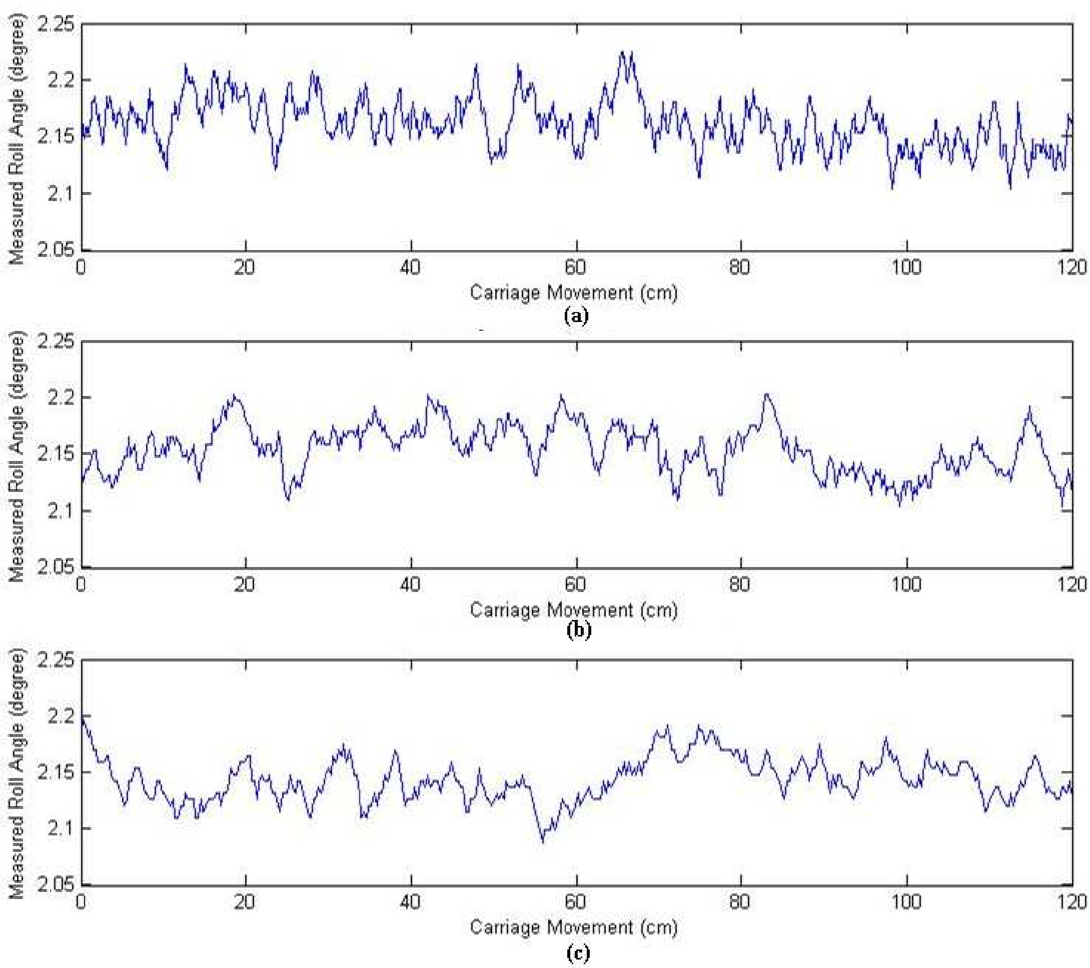


**Figure 5.39 Pitch Angle Measured by the Gyroscope Sensor at Laser-carriage Movement Speeds of (a) 6.31cm/s, (b) 9.32 cm/s, and (c) 14.63 cm/s, while the Laser Mirror was not Rotating**

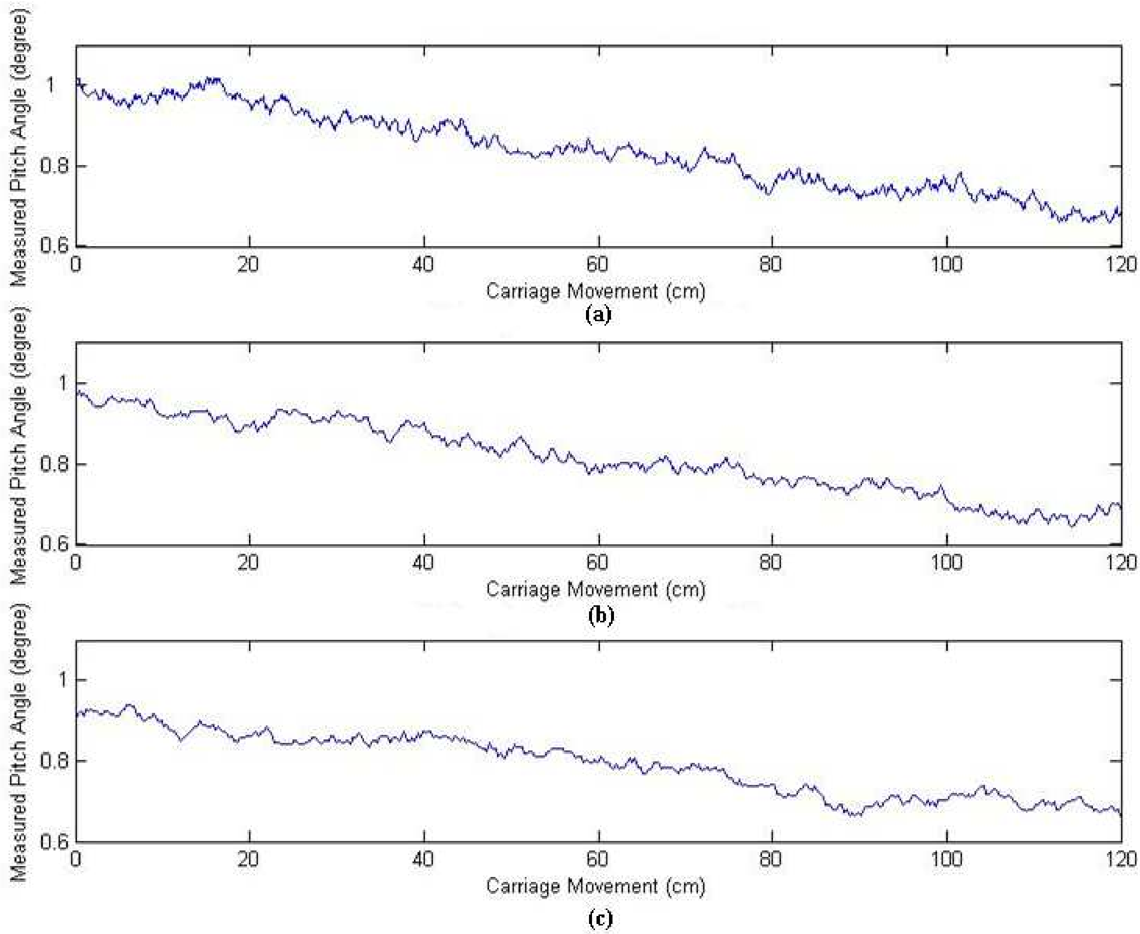
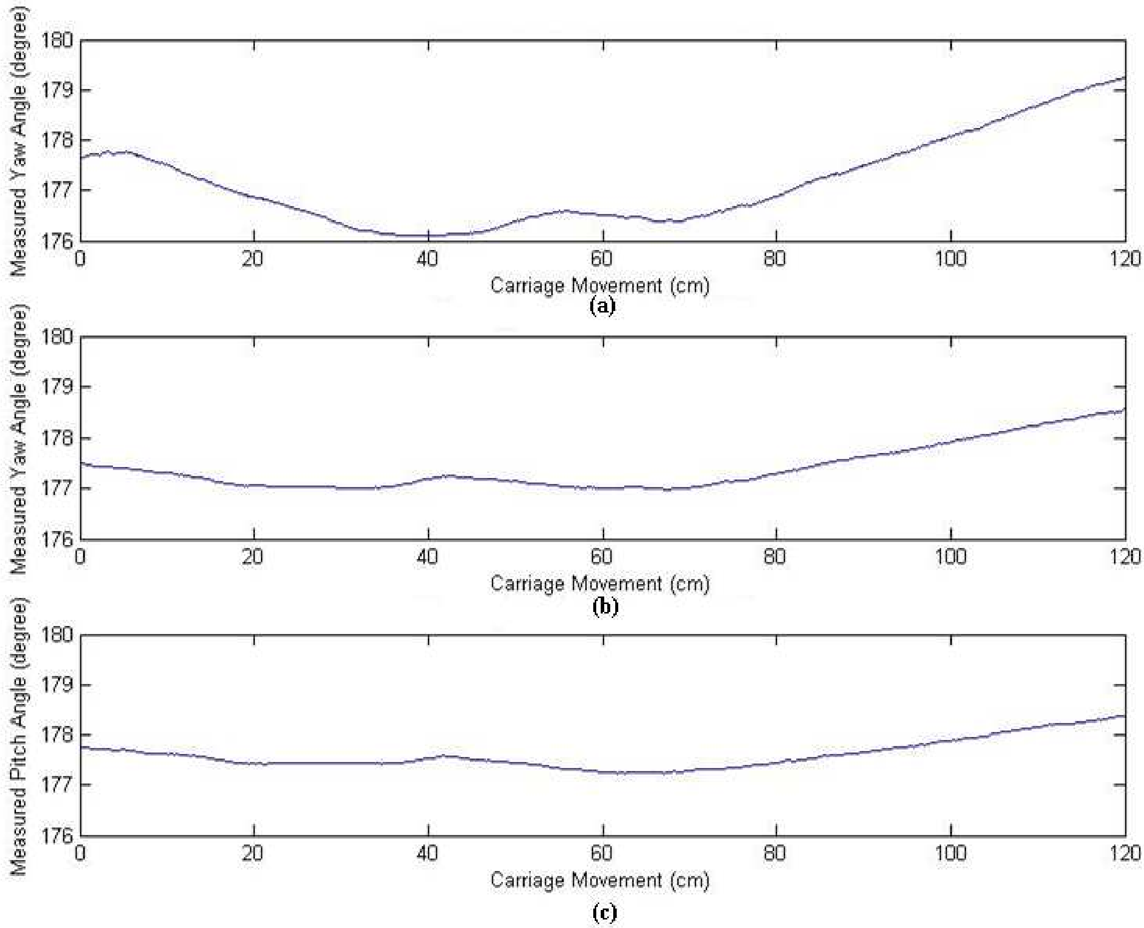


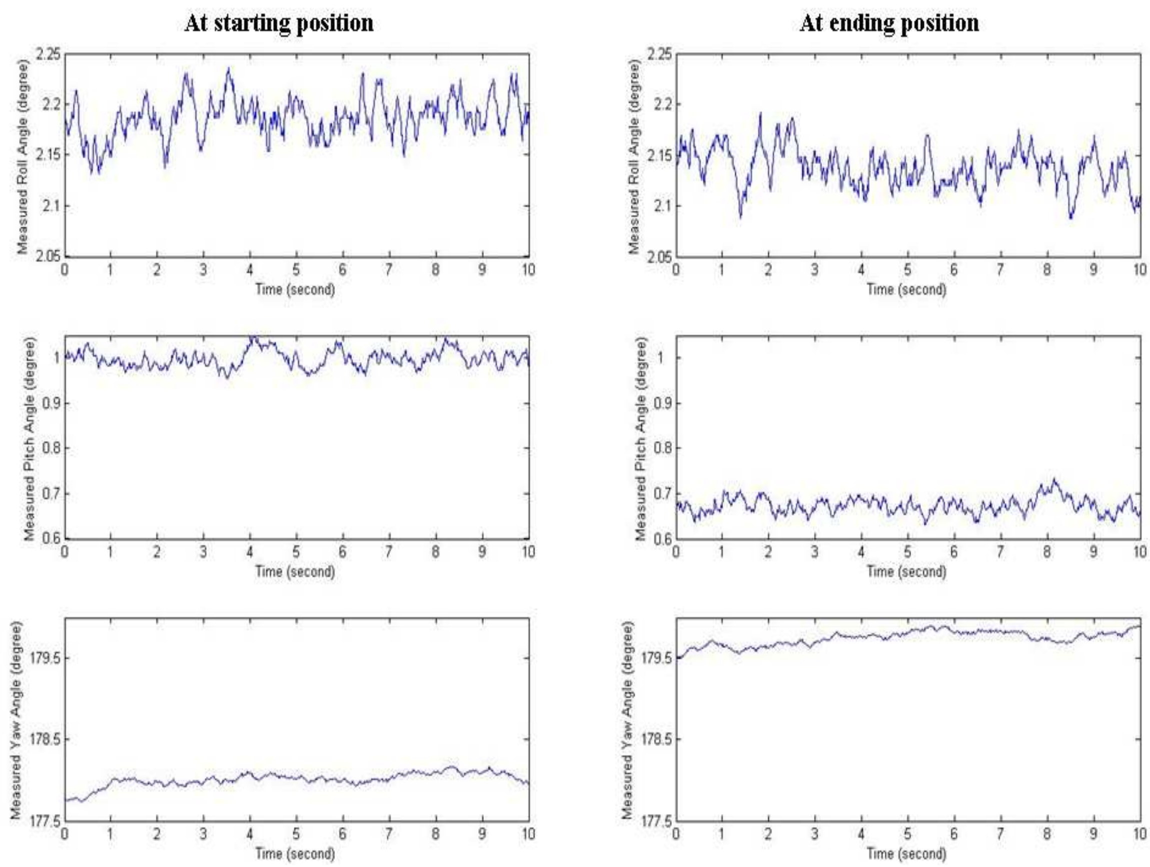
**Figure 5.40 Yaw Angle Measured by the Gyroscope Sensor at Laser-carriage Movement Speeds of (a) 6.31cm/s, (b) 9.32 cm/s, and (c) 14.63 cm/s, while the Laser Mirror was not Rotating**




When the laser mirror was not rotating, the average values of roll angles were 2.1617°, 2.1546°, and 2.1438° at carriage movement speeds of 6.31cm/s, 9.32 cm/s, and 14.63 cm/s, respectively. The variation ranges of roll angles at three difference movement speeds were 2.1039°~2.2248°, 2.1039°~2.2028°, and 2.0875°~2.1973°. Thus, the measured roll angle was maintained within a range of 0.12°, which was slightly reduced from 0.16°, when the laser mirror was in motion (condition A). Pitch angle also decreased when the laser-carriage moved from the starting position to the ending position on the linear rail. The reduction range was about 0.36°, which was slightly smaller than that observed under condition A (0.38°). Comparing these numbers under two conditions, it can be concluded that the mirror rotation did not significantly affect the roll and pitch measurements. The trends on yaw angles were similar to those derived for condition A.

*Condition C: The laser mirror rotated when the laser-carriage remained still at the starting and ending positions on the rail, respectively* (Figure 5.41).

**Figure 5.41 Roll, Pitch, and Yaw Angles Measured by the Gyroscope Sensor at Two positions on the Rail, while the Laser Mirror Rotated at 2600 RPM and the Laser Carriage Remained Still**




Comparison between the two positions clearly showed that the pitch and yaw angles measured at the two positions were different. The average pitch angle at the starting position (0.997°) was greater than that at the ending position (0.6718°), whereas the average yaw angle measured at the starting position (178.0148°) was less than that at the ending position (179.9615°). The roll angle almost kept constant. The average roll angles measured at two positions were 2.1857° and 2.139°.

Comparing these numbers with those obtained for condition A, one can find that the pitch and roll angles measured at the two positions were similar. Thus, it can be concluded that the movment of the laser-carriage did not significantly affect the pitch and roll measurements.

*Condition D: The laser-carriage moved at three different speeds on the rail while the laser mirror rotated at three different speeds, when the rail was fixed at both ends* (Figure 5.42 ~ Figure 5.44).

**Figure 5.42 Roll Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds, when the Rail was Fixed at Both End**

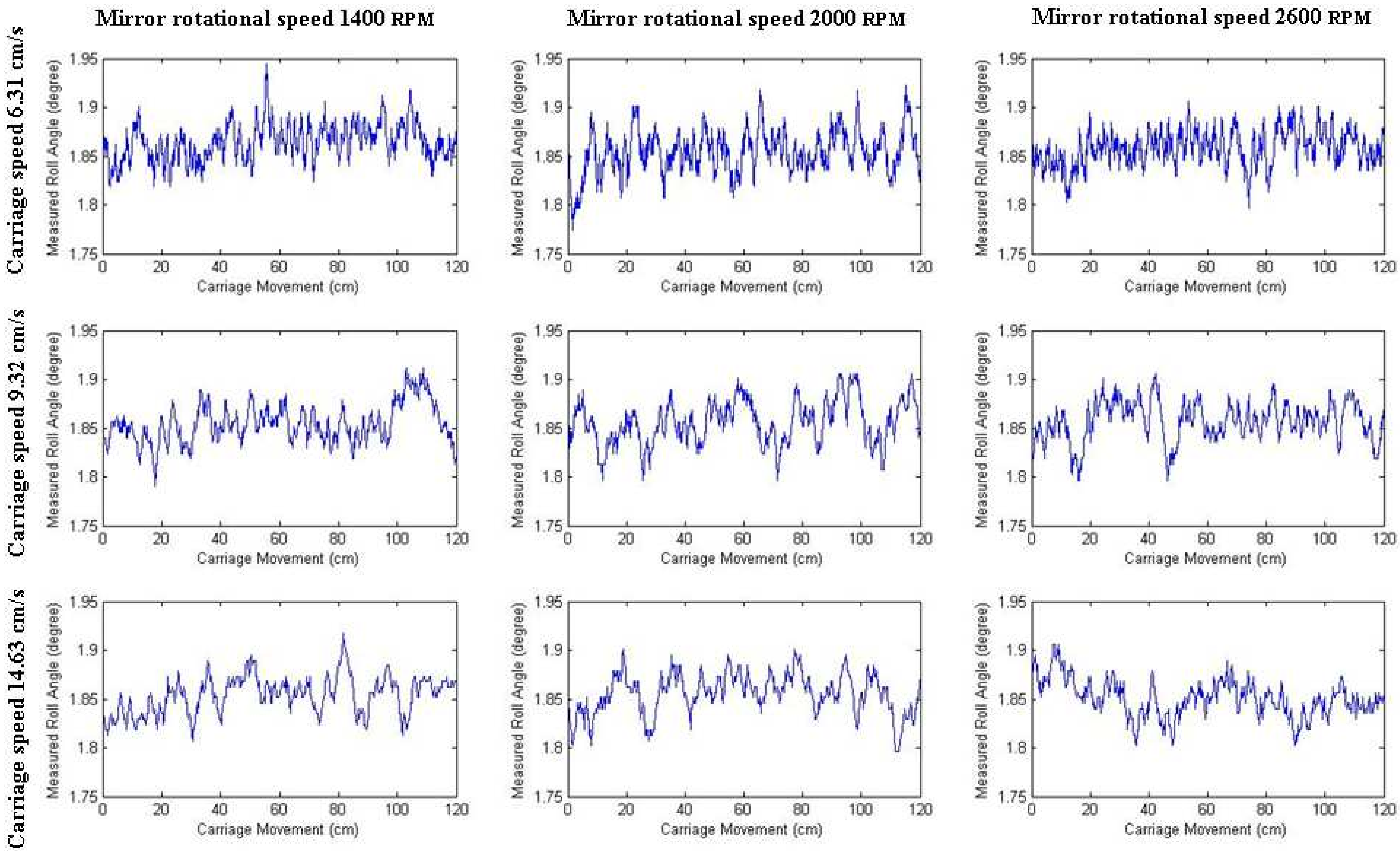


**Figure 5.43 Pitch Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds, when the Rail was Fixed at Both End**

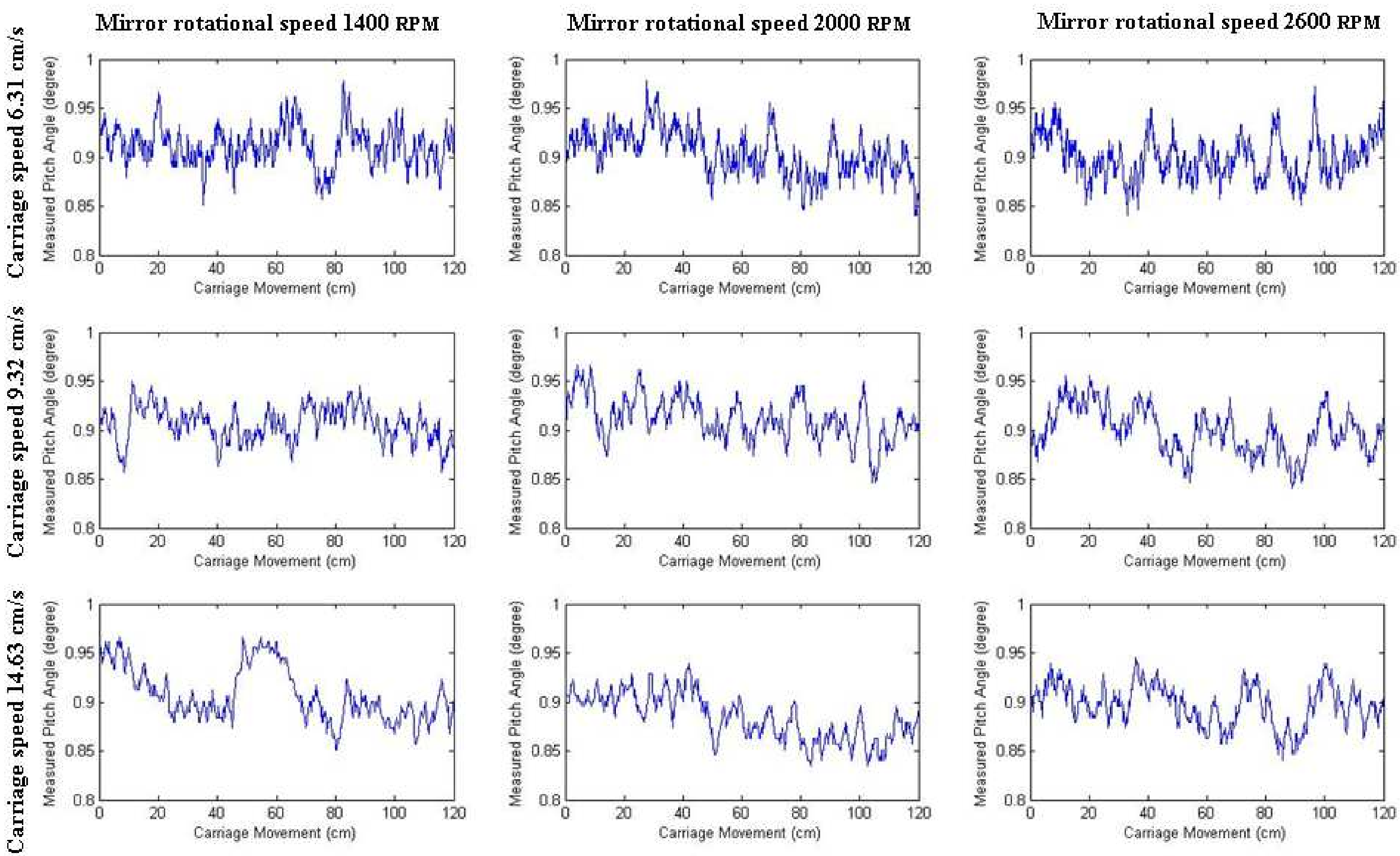
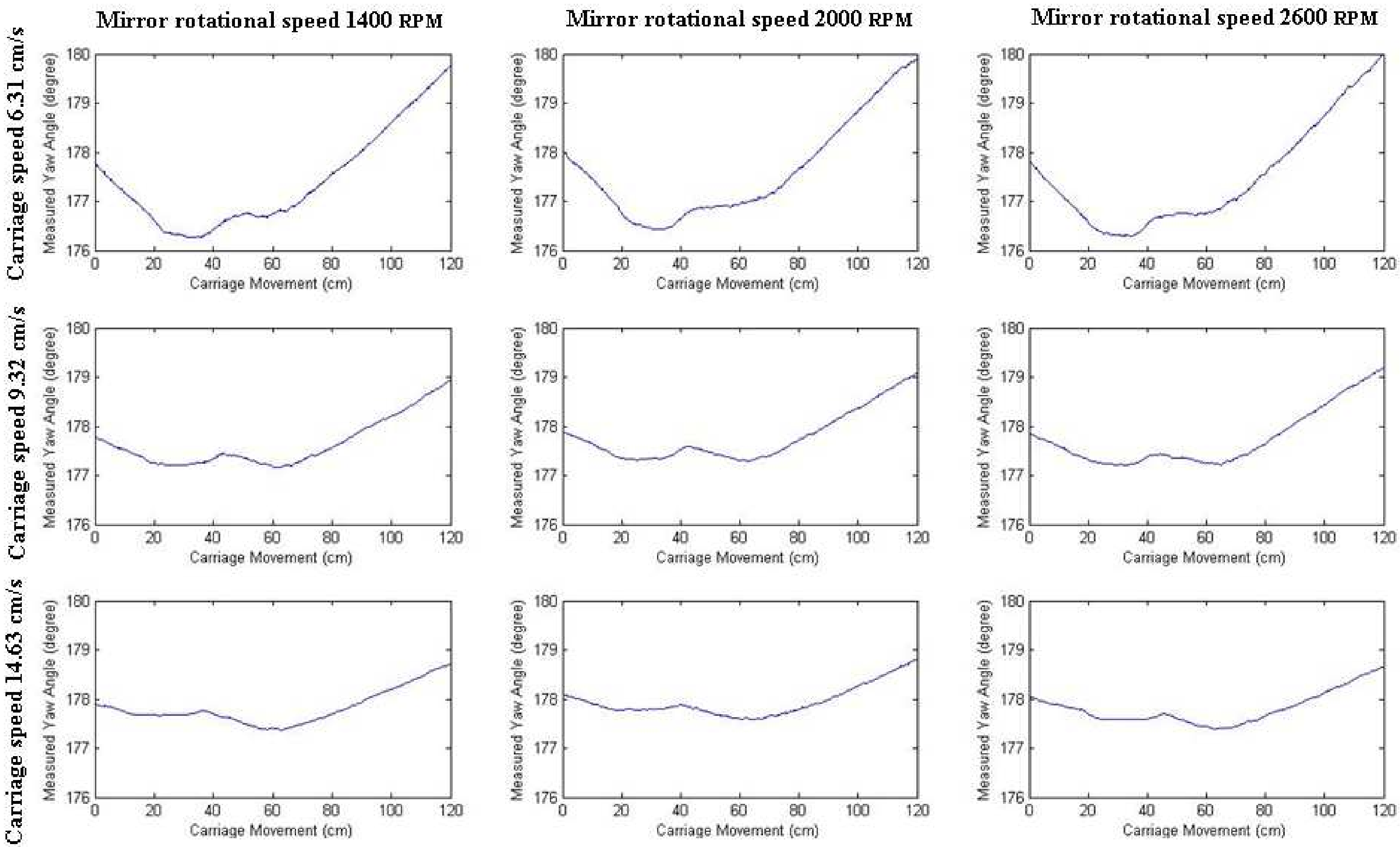


**Figure 5.44 Yaw Angles Measured by the Gyroscope Sensor at Combinations of Three Mirror Rotational Speeds and Three Carriage Movement Speeds, when the Rail was Fixed at Both End**



The “mixed” procedure used in condition A was applied to study the impact of the mirror rotation, carriage movement, and their interaction on the roll and pitch measurements under condition D, respectively (Appendix I). Carriage movement speed (CS), mirror rotation speed (MR), and interactions (CS*MR) between the rotational speeds and the carriage movement speeds were considered as the fixed effects. The unequal numbers of replicates in the combinations were also considered as a fixed effect. A least squares analysis of a factorial design with unequal numbers of replicates at factor combinations was carried out for roll and pitch measurements, respectively. The results are summarized in Tables 5.10 and Table 5.11, respectively.

**Table 5.10 Results of Statistics Analysis of the Measured Roll Angle under Condition D**

| **Effect** | **CS levels** | **MR levels** | **Estimate** (degree) | **Standard Error** (degree) | **Pr > \|t\|** |
|---|---|---|---|---|---|
| CS | 1 | | 1.8566 | 0.000399 | <0.0001 |
| CS | 2 | | 1.8575 | 0.000502 | <0.0001 |
| CS | 3 | | 1.8567 | 0.000603 | <0.0001 |
| MR | | 1 | 1.8586 | 0.000497 | <0.0001 |
| MR | | 2 | 1.8562 | 0.000491 | <0.0001 |
| MR | | 3 | 1.8559 | 0.000479 | <0.0001 |
| CS*MR | 1 | 1 | 1.8622 | 0.000666 | <0.0001 |
| CS*MR | 1 | 2 | 1.8520 | 0.000663 | <0.0001 |
| CS*MR | 1 | 3 | 1.8555 | 0.000660 | <0.0001 |
| CS*MR | 2 | 1 | 1.8561 | 0.000864 | <0.0001 |
| CS*MR | 2 | 2 | 1.8577 | 0.000859 | <0.0001 |
| CS*MR | 2 | 3 | 1.8586 | 0.000849 | <0.0001 |
| CS*MR | 3 | 1 | 1.8576 | 0.001035 | <0.0001 |
| CS*MR | 3 | 2 | 1.8588 | 0.001017 | <0.0001 |
| CS*MR | 3 | 3 | 1.8536 | 0.000975 | <0.0001 |

**Table 5.11 Results of Statistics Analysis of the Measured Pitch Angle under Condition D**

| **Effect** | **CS levels** | **MR levels** | **Estimate** (degree) | **Standard Error** (degree) | **Pr > \|t\|** |
|---|---|---|---|---|---|
| CS | 1 | | 0.9085 | 0.000426 | <0.0001 |
| CS | 2 | | 0.9035 | 0.000536 | <0.0001 |
| CS | 3 | | 0.8920 | 0.000644 | <0.0001 |
| MR | | 1 | 0.9073 | 0.000531 | <0.0001 |
| MR | | 2 | 0.8991 | 0.000524 | <0.0001 |
| MR | | 3 | 0.8976 | 0.000511 | <0.0001 |
| CS*MR | 1 | 1 | 0.9148 | 0.000711 | <0.0001 |
| CS*MR | 1 | 2 | 0.9078 | 0.000708 | <0.0001 |
| CS*MR | 1 | 3 | 0.9030 | 0.000705 | <0.0001 |
| CS*MR | 2 | 1 | 0.9039 | 0.000922 | <0.0001 |
| CS*MR | 2 | 2 | 0.9097 | 0.000918 | <0.0001 |
| CS*MR | 2 | 3 | 0.8968 | 0.000907 | <0.0001 |
| CS*MR | 3 | 1 | 0.9031 | 0.001105 | <0.0001 |
| CS*MR | 3 | 2 | 0.8797 | 0.001086 | <0.0001 |
| CS*MR | 3 | 3 | 0.8931 | 0.001041 | <0.0001 |

The carriage movement speed, mirror rotation speed, and their interactions were found to have statistically significant effects on the measured roll and pitch angles, $P<0.0001$, when the significance level $\alpha=0.05$. However, the standard errors at the combinations were less than 0.0011 for roll angle and 0.0012 for pitch angle, indicating that these effects were not practically significant.

The measured roll angle was maintained within a rangle of 0.13°, which was smaller than that observed under condition A (0.16°). The range of measured pitch angle was also 0.13°. This result was different from those obtained from conditions A and B, where a clear declining trend in pitch angle was observed when the laser-carriage moved forward. Thus, it can be concluded that the rigidity of the aluminum frame to support the rail was not sufficient. The results of yaw angle measurement were similar to those derived from conditions A and B. A similar trend on the yaw angles was observed when the rail was supported at both ends.

The trends on yaw angle readings may have been caused by changes in the magnetic field when the laser line scanner moved along the rail. However, displacements of the yaw angle should not have affected the laser measurements because the yaw angle was only used in coordinate system conversion. The roll angles measured under all conditions had small variation ranges (<0.16°), resulting in no practically significant effect on the elevation measurement. By comparing the measurements of pitch angle under conditions A and B with those under condition D, it was obvious that the rail always tended to tilt to the direction where the laser scanner was located. These changes were related to the change of weight balance on the linear rail. The coordinate system of the laser system was defined based on the orientation of the linear rail. If the linear rail was tilted, the positions of the laser measurements would change. Thus, changes of pitch angle would affect the elevation measurements. These measurement errors shoud be compensated.

## 5.3 Preliminary Test on a Bare Soil Surface

### *5.3.1 Description*

The laboratory experiments demonstrated that the laser system can provide accurate DEM data with high resolution. However, this vehicle-based system was aimed to measure soil surface roughness in the field. It was desirable to examine the functionality of the laser system under complex field conditions. Furthermore, the ability of the system to correctly register the measured DEM in the geographic coordinate system also needed to be examined.

The field test reported in this dissertation was only a preliminary study. It focused on evaluating the capability of the laser system in measuring geo-referenced DEM under field conditions.

### *5.3.2 Objectives*

The objectives of the preliminary test on a bare soil surface were:

1). to examine the capability of the laser system in measuring soil surface elevation;

2). to study possible sources of measurement errors and to design algorithms for error correction;

3). to develop algorithms to convert derived DEM data from a local coordinate system to the geographic coordinate system.

### *5.3.3 Methodology*

#### *5.3.3.1 Test Procedure*

The outdoor experiment was conducted on November 20th, 2007. The scanned bare soil surface was located near the east parking lot of the Bill Snyder Family stadium, Manhattan KS (Figure 5.45). The selected laser sampling rate was 10 kHz. The maximum measurement range was set at 304.8 cm. The laser mirror rotated at the maximum speed of 2600 RPM. The laser-carriage moved at the medium speed (9 cm/s). The vertical distance from the laser scanner to the bare soil surface was measured at approximately 117 cm. A Trimble AgGPS 132 DGPS receiver was used to provide the global positioning information.

**Figure 5.45 Outdoor Test on a Bare Soil Surface**

### *5.3.3.2 Cross-validation-based Outlier Removal Algorithm*

In statistics, an outlier was an observation that was numerically distant from the rest of the data (Barnett and Lewis, 1994). Because outliers were found in the data sets acquired by the laser system, an algorithm was developed to recognize and remove the outliers.

The algorithm to remove the outliers used a polynomial model to describe the elevation profile derived during each scan and a leave-one-out cross-validation method to select the model that best fitted the data. From the selected model, the predicted laser data and a 95% confidence interval can be obtained. Measured data that fell outside the confidence interval would be treated as outliers and removed from the data set. The algorithm consisted of the following steps:

(1) Break the data obtained during a scan into *k* partitions,

(2) Take one of the partitions out from the data,

(3) Use the remaining partitions to develop a polynomial regression equation with order *r* (Equation 5.17),

$$y = a_r x^r + a_{r-1} x^{r-1} + \ldots + a_2 x^2 + a_1 x + a_0 \quad (5.17)$$

where (x, y) are the data points within the remaining partitions, and

$a_0$, $a_1$,…, $a_r$ are the coefficients of the polynomial regression equation.

(4) Use the partition that was taken out in step 2 as the test set to test the polynomial model and to calculate the prediction error (Equation 5.18),

$$E_P = (Y - \hat{Y})^2 \quad (5.18)$$

where *Y* is the observed data of the test set, and

$\hat{Y}$ is the predicted data of the test set.

(5) Repeat steps (2) through (4) until all partitions have been used as a test set,

(6) Determine the average of the prediction errors,

$$E_r = \frac{1}{k}\sum_{i=1}^{k}(Y_i - \hat{Y}_i)^2 , \quad (5.19)$$

where *k* is the number of partitions.

(7) Increase order *r* at step (3) and repeat steps (2) through (6) until the maximum order, 16, is reached,

(8) Select the optimum polynomial order $r$ that gave the minimum error $E_r$ among models of all orders tested,

(9) Develop a polynomial regression equation at the optimum order using all data partitions,

(10) Determine the 95% confidence intervals of the regression model,

(11) Remove data points that fell outside the confidence interval.

The algorithm was implemented in MATLAB (Appendix J). Two functions, *polyfit* and *polyval*, were used. The *polyfit* function generated a 'best fit' polynomial (in the least squares sense) of a specified order for a given set of data. The *polyval* function calculated the predicted value of y at a given value of x using the best fit polynomial equation. Both functions had two return parameters. [$p$, S] = polyfit(x, y, n) returned the polynomial coefficients $p$ and a structure S for use with *polyval* to obtain predictions or error estimates. [$y$, δ] = polyval($p$, x, S) uses the optional output structure S generated by *polyfit* to generate prediction y and error estimates δ. Thus, $y \pm 2\delta$ contained 95% of the predictions and, hence, can be used as the 95% confidence interval.

### *5.3.3.3 Coordinate Conversion Algorithm*

The DEM data generated by the laser system was represented in its local coordinate system. In order to locate these data points on a geographic coordinate system, a coordinate conversion algorithm was required.

The most convenient way to specify points on the curved surface of the Earth was using latitude, longitude, and altitude in a coordinate system built on a geodetic datum. However, these points can also be specified on a flat surface or map in grid coordinates after projecting the curved surface onto a flat surface.

A geodetic datum was a tool to describe the size and shape of Earth surface, the origin and orientation of the coordinate system used in mapping Earth. Throughout time, there were hundreds of geodetic datums in use around the world. One of the most widely used datums was the World Geodetic System 1984 (WGS84) developed by the U.S. Department of Defense. It comprised a coherent set of global models and definitions: an

Earth-centered, Earth-fixed (ECEF) Cartesian coordinate frame, a reference ellipsoid of revolution as a geometric model of the shape of Earth, a characterization of Earth's gravity field, and a consistent set of fundamental constants (Misra and Enge, 2001). WGS84 was essential to the development of the Global Positioning System (GPS), because the position information from the GPS receiver was expressed in the WGS84 geodetic coordinate system.

The Universal Transverse Mercator (UTM) geographic coordinate system used a grid-based method to specify locations of points corresponding to their latitudes and longitudes (USGS, 2006). Based on a transverse Mercator projection, the UTM system divided the surface of Earth into sixty north-south zones, each of which was six degrees wide in longitude. The zones were numbered from west to east, beginning at the International Data Line (180°). Each zone was subdivided into the eastern and western halves by drawing the 'central meridian', which was a line perpendicular to the equator between the poles. In each zone, coordinates were measured North and East in meters. To avoid negative coordinates, an arbitrary false northing value of 10,000,000 meters was assigned to the equator and an easting value of 500,000 meters was assigned to the central meridian (Riesterer, 2008). The WGS84 ellipsoid was used as the underlying model of Earth in the UTM coordinate system.

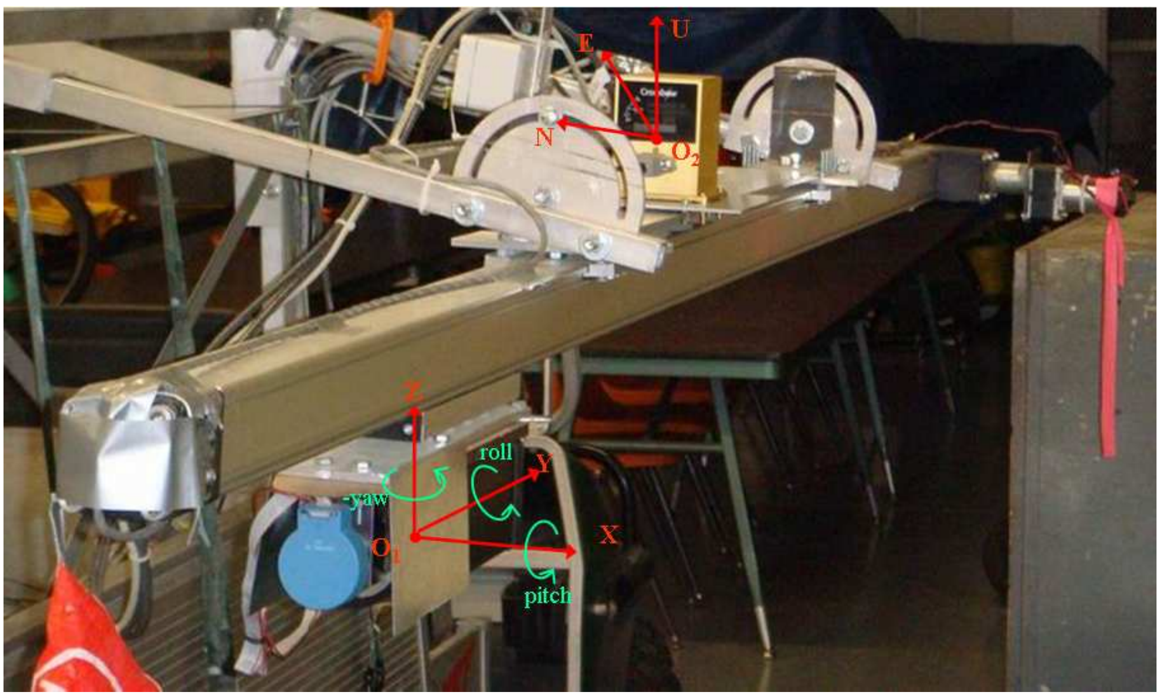
The coordinate conversion algorithm used in the laser system was to convert the DEM data from its local coordinate system to the UTM geographic coordinate system using the laser measurements, the yaw angle measured by the gyroscope, the latitude and longitude information provided by the GPS receiver, and the known positions of these sensors (Appendix K). Two local coordinate systems and one global coordinate system were defined during conversion. These coordinate systems are described as follows:

(1). The X-Y-Z coordinate system (Figure 5.46). This was a local coordinate system defined on the laser system with Y and X axes aligned with the forward and rightward and Z axis pointing upwards. The origin $O_1$ located at the center of the rotating mirror when the laser was at the starting position (Figure 5.4).

(2). The E-N-U coordinate system (Figure 5.46). This was a local coordinate system with the three axes aligned with the East, North and Upward directions, respectively. The origin $O_2$ located at the center of the gyroscope sensor. For the field test, the roll and pitch angles were measured only once before the laser-carriage started to move. Variations in the pitch and roll angles during the rotation of the mirror and the translation of the laser-carriage on the rail were ignored. Only the measured yaw angle was used in the transformation.

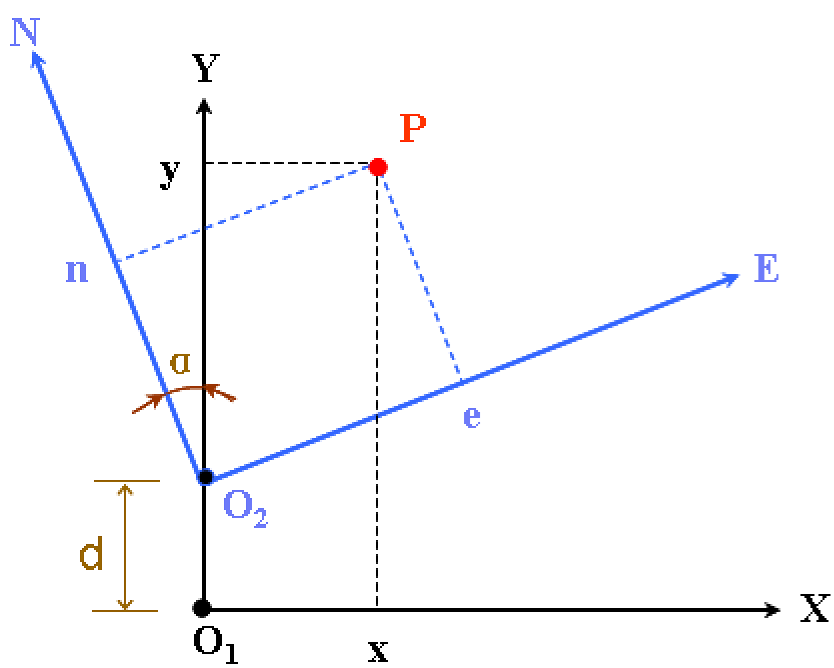
(3). The UTM geographic coordinate system. This is an Earth-fixed coordinate frame with the three axes aligned with the North, East and down (NED) directions, respectively. Obviously, only translations were involved in the transformation between the E-N-U and the UTM coordinate system.

**Figure 5.46 The Local Coordinate Systems**



The yaw angle measured by the gyroscope sensor was referenced to the magnetic north, pointing to Earth's magnetic north pole. On the other hand, the UTM coordinates used the true north (geographic North Pole) as its north. The true north did not coincide with the magnetic north. The angular difference between the true north and the magnetic north was called "declination" or "magnetic variation". By knowing the yaw angle and the magnetic declination, elevations measured in the X-Y-Z coordinate system can be converted into the E-N-U coordinate system by using Equations (5.20), (5.21) and (5.22). The mathematical model of the calculation is shown in Figure 5.47. The estimated value of magnetic declination can be found on the website of National Geophysical Data Center, which is provided as a public service by the U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite, Data and Information Service.

**Figure 5.47 Illustration of Coordinate Conversation between the X-Y-Z coordinate system and the E-N-U coordinate system**




$$e = \cos\alpha \cdot x + \sin\alpha \cdot (y - d) \tag{5.20}$$

$$n = -\sin\alpha \cdot x + \cos\alpha \cdot (y - d) \tag{5.21}$$

$$u = z + z_0 \qquad (5.22)$$

where (x, y) was the location of a data point P in the X-Y-Z coordinate system, in meter,

$z$ was the elevation of data point P in the X-Y-Z coordinate system, in meter,

($e$, $n$) was the location of the same data point P in the E-N-U coordinate system, in meter,

$u$ was the elevation of data point P in the E-N-U coordinate system, in meter,

$$a = \gamma + \delta \qquad (5.23)$$

where $\delta$ was the magnetic declination, in radian,

$\gamma$ was the yaw angle measured by the gyroscope sensor, in radian. Due to the fact that the Z-axis in the X-Y-Z coordinate system was in the opposite direction of the U-axis in the E-N-U coordinate system, a negative sign was added to $\gamma$.

d = 51cm, the horizontal distance between the origin $O_1$ of the X-Y-Z coordinate system and the origin $O_2$ of the E-N-U coordinate system,

$z_0$ = 24cm, the vertical distance between the origin $O_1$ of the X-Y-Z coordinate system and the origin $O_2$ of the E-N-U coordinate system.

The GPS antenna was mounted on top of the gyroscope sensor during field experiments. The vertical distance between the center of the antenna and the center of the gyroscope sensor was 11 cm. Coordiantes in the E-N-U coordinate system can be converted into the UTM (WGS84) coordinate system by using Equations (5.24), (5.25), and (5.26).

$$e' = e + e'_0 \qquad (5.24)$$

$$n' = n + n'_0 \qquad (5.25)$$

$$u' = u + u'_0 + u_a \qquad (5.26)$$

where (e’, n’) were easting, northing in the UTM geographic coordinate system, in meter,

u’ was the altitude, which was referenced to Mean Sea Level (MSL), in meter,

(e, n) were the coordiantes in the E-N-U coordinate system, in meter,

*u* was the elecations in E-N-U coordinate system, in meter,

($e'_0$, $n'_0$) was the coordinate of the origin $O_2$ of the E-N-U coordinate system in the UTM (WGS84) geographic coordinate system (Appendix L),

$u'_0$ was the elevation of the center of GPS antenna, referenced to the MSL, in meter,

$u_a$ = 11cm, was the vertical distance betweent the center of the GPS antenna and the origin $O_2$ of the E-N-U coordinate system.

### *5.3.4 Results and Discussion*

Outliers were observed in most three-dimensional raw datasets acquired by the laser system in the outdoor test. Figure 5.48 shows one of these data sets. Figure 5.49 is the same data set viewed in the X-Z plane. These plots clearly showed that most of the measured data points laid between -110 cm and -120 cm, whereas a few were great than -120 cm. These data points appeared to qualify for the “outliers” by human judgment.

**Figure 5.48 3D Plot of a Raw Laser Data Set Measured during the Field Test**

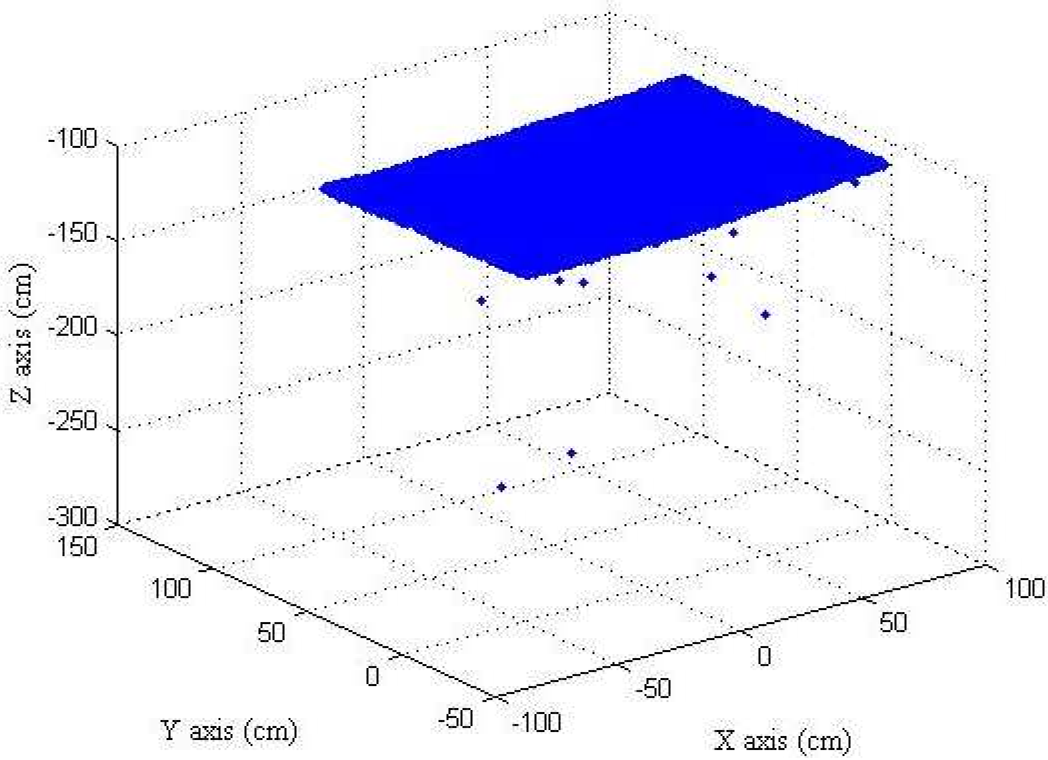
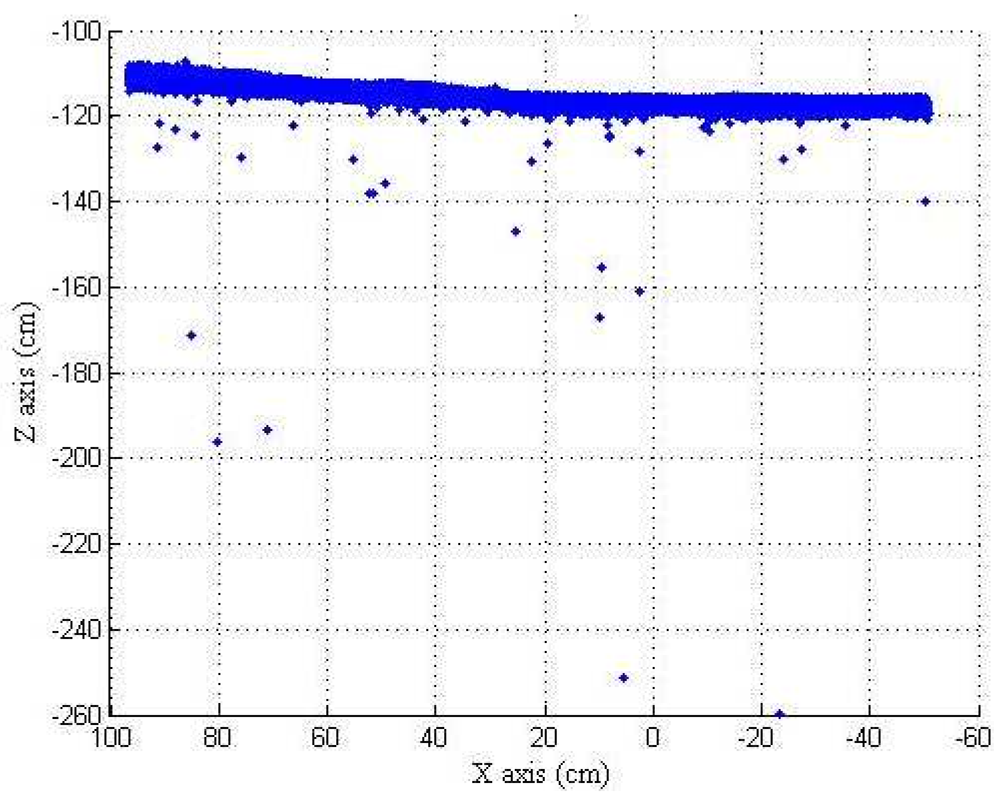



**Figure 5.49 Outliers Viewed in the X-Z Plane**



The leave-one-out cross-validation algorithm was applied to data shown in Figures 5.48 and 5.49. The result is shown in Figures 5.50 and 5.51. Obviously, most outliers were removed. The bare soil surface had an inclined slope.

**Figure 5.50 3D Plot of the Raw Data after the Outliers were removed**

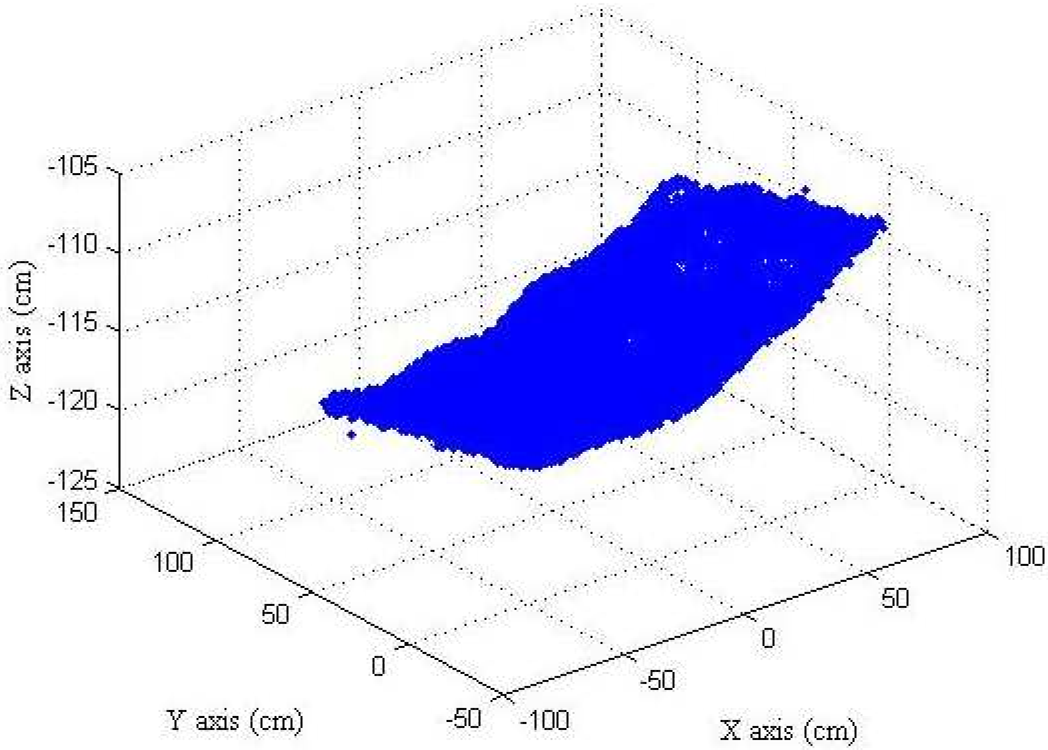
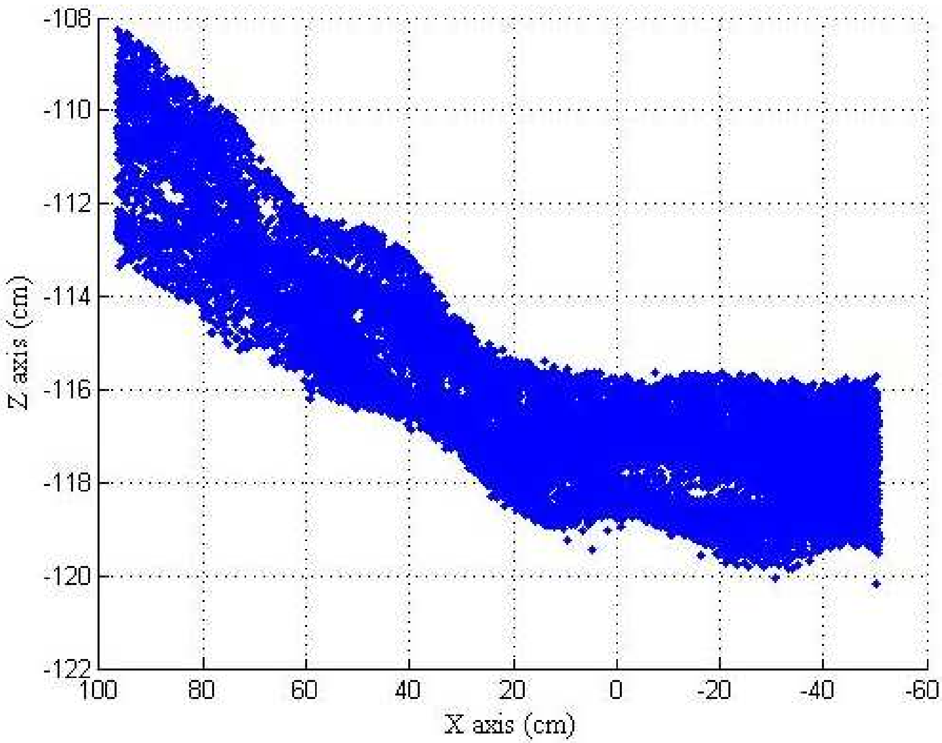
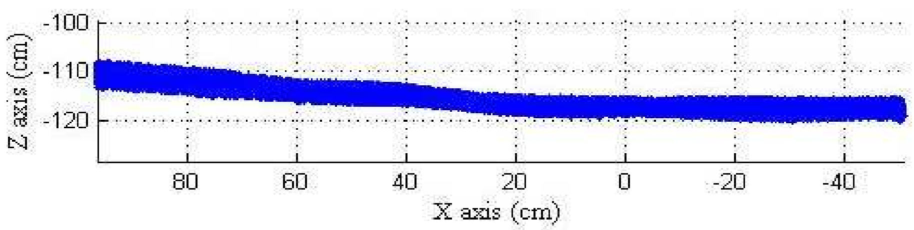


**Figure 5.51 (a) The Raw Data Viewed in the X-Z Plane after the Outliers were removed, (b) The Same Data Viewed in the X-Z Plane after the Scales for the X- and Z-axis were Unified**



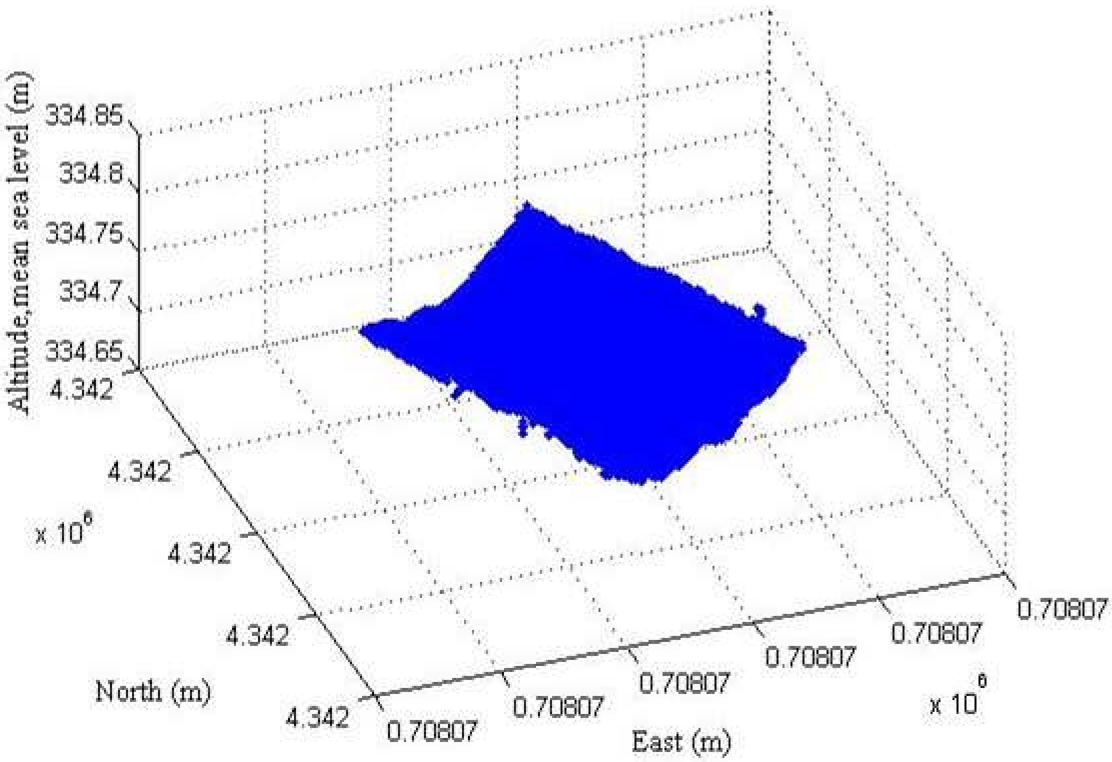

**(a)**



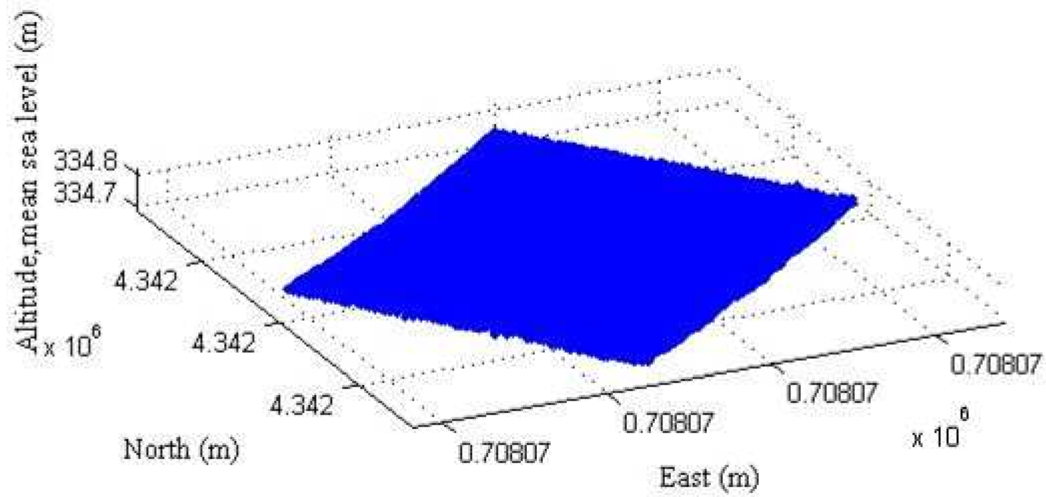

**(b)**

The coordinate conversion algorithm, along with the yaw angles measured by the gyroscope sensor and the latitudes, longitudes and altitudes provided by the GPS, was used to convert the measured data from the X-Y-Z local coordinate system to the geographic coordinate system, UTM-WGS84 (Figure 5.52). The recorded latitude was 39°12.161782’ north and the longitude was 96°35.424589’ west. The results showed that the soil surface was located at 4,342 kilometers north and 208.07 kilometers east in zone 14 in the UTM geographic system.

**Figure 5.52 (a) Measured DEM in the UTM (WGS84) Geographic Coordinate System, (b) The Same Data Plotted at Identical X-, Y- and Z-axis Scales**




(a)




(b)

# CHAPTER 6 - Conclusions

Conclusions drawn from each stage of this study are summarized as follows:

## 6.1 Laser System

A vehicle-based laser measurement system for generating high-resolution DEM data was developed. The system consisted of five units:

- a laser line scanner to measure the surface elevation,
- a flexible frame-rail mechanism to support the sensors on the bed of a small utility vehicle,
- a gyroscope sensor to measure the roll, pitch, and yaw angular displacements of the vehicle,
- a RTK GPS to provide the geographic position of the measured surface, and
- a data-acquisition and control unit.

A user interface program was developed to control the laser system and to collect the sensors data through a field laptop.

## 6.2 Components Tests

- A test on the rotating mirror showed that the DC motor required a transient period of about 20 seconds to reach its steady-state speed. The laser scanner should not start the measurement during the transient period.
- After testing the laser scanner on three types of objects – white paper, black paper, and sand surface, it was found that the white paper gave the most consistent distance measurement. The laser scanner was then calibrated on the whiter paper for distance measurement.
- The spatial resolution of the DEM models developed by the laser system was affected by the rotational speed of the mirror, the sampling rate, the movement speed of the laser-carriage, the height of the laser line scanner from the ground, and the size of the scan area.
- A test conducted on the gyroscope sensor showed that errors in pitch and roll measurement were larger in combined pitch/roll rotations than those in single pitch

or roll rotations. Within ±30° of single pitch or roll rotations, the measurement errors in pitch and roll angles were within 0.8° and 0.4°, respectively.

- A modeling study on the effects of pitch and roll measurement errors on elevation measurement demonstrated that: a) Errors in roll angle measurement did not significantly affect the elevation measurement; b) For positive pitch angles, especially those approaching 30°, errors in pitch angle measurement had a significant effect on the elevation measurement; c) For negative pitch angles in the ranges of 0~-18°, errors in pitch angle measurement did not significantly affect the elevation. For those in the ranges of -27~-30°, errors in pitch angle measurement had a significant effect on the elevation measurement. Outside these ranges, the effect depended on the measured distance.

## 6.3 Accuracy Tests

- DEM models were created by interpolating the raw laser data using a two-dimensional, three-nearest neighbor, distance-weighted algorithm.
- Test results of surfaces with known geometric shapes indicated that the laser system can describe individual objects more accurately than a flat surface. Shapes of different objects can be identified from the DEMs.
- The accuracy of the laser system in elevation measurement was evaluated by comparing the DEM models generated by the laser system for a sand-stone surface with a reference DEM model generated by a more accurate, laser-based profile meter for the same surface. The correlation coefficient between the DEMs was calculated. Within four replications, the highest correlation coefficient between the measured and reference DEMs was 0.9371. The correlation coefficients among the four replications were greater than 0.948.
- A median threshold filter and a median filter applied to the raw laser data before and after the interpolation slightly improved both measurement accuracy and repeatability. The correlation coefficient between the measured and reference DEMs was improved to 0.954. Correlation coefficients of greater than 0.988 were achieved among the four replications.

- Differences in materials and colors affected the grayscale images, which were created from the intensity data provided by the laser scanner.

## 6.4 Noise Tests

- Results of the ambient light test indicated that neither sunlight nor fluorescent light affected the elevation measurement of the laser system. The system provided consistent elevation measurements under both indoor and outdoor lighting conditions.
- A test on the linear rail when the laser-carriage moved with and without the mirror rotation indicated that the mirror rotation did not significantly affect the roll and pitch measurement.
- A test on the linear rail when the mirror rotated with and without the laser-carriage movement indicated that the laser-carriage movement did not significantly affect the roll and pitch measurement.
- A test on the linear rail when both ends were fixed indicated that the rigidity of the aluminum frame to support the rail was not sufficient.
- Tests on the linear rail showed that the rail slightly tilted to the direction where the laser scanner was located because of the change in weight balance. The variation range of the roll angle was small ($<0.16°$), which resulted in no practically significant effect on the elevation measurement. A consistent trend in yaw angle variation was observed when the laser scanner moved along the rail. It may have been caused by changes in the magnetic field.
- Errors in the elevation measurement caused by the variation of the pitch angle needed to be compensated.

## 6.5 Preliminary Test in the Field

- A leave-one-out cross-validation algorithm was used to remove the outliers observed in the raw laser data.
- The DEM data measured by the laser system was converted from a local coordinate system to the UTM geographic coordinate system by using the yaw angle measured

by the gyroscope, the known positions of the sensors, and the latitude, longitude, and altitude information provided by the GPS receiver.

- Results of the preliminary outdoor experiment indicated that the laser system was capable of providing geo-referenced DEM data.

# CHAPTER 7 - Future Work

Rail vibration test results showed that the linear rail tilted to the direction where the laser scanner was located when the laser-carriage moved on the rail. These angular changes of the rail would affect the elevation measurements. Two methods to solve this effect are recommended. One is to strengthen the rigidity of the aluminum frame. Another method is to correct the measurement errors during the generation of the three-dimensional data.

The comparison between DEMs obtained by the laser system and the reference system showed that the laser system can describe the measured surface with satisfactory accuracy and repeatability based on distance measurements. Using additional information such as intensity may have the potential for removing the measurement error and increasing the accuracy. Thus, future studies on the fusion of distance measurement and intensity information need to be investigated.

More field experiments need to be conducted for further evaluating the performance of the laser system in measuring the soil surface roughness.

Differences in materials and colors affected the graycale image. Thus, tests on the reflected light intensity need to be conducted to examine the potential capabilities of the laser system in measuring crop residues on soil surface. Methods in digital image processing could be introduced to enhance the grayscale image.

# References

Ackermann, F. 1999. Airborne laser scanning - present status and future expectations. *ISPRS Journal of Photogrammetry and Remote Sensing* 54(2-3): 64-67.

Acuity Research Inc. 2000. *AccuRange Line Scanner User's Manual*. Rev. 2.3. Menlo Park, CA: Acuity Research, Inc.

Ahlberg, S., U. Soderman, M. Elmqvist and A. Persson. 2004. On modeling and visualization of high resolution virtual environments using lidar data. In *Proc. of 12th International Conference on Geoinformatics Geospatial Information Research: Bridging the Pacific and Atlantic*. 299-306. University of Gavle, Sweden.

ArcTron. 2005. New technology revolutionises 3D surveying techniques. Regensburg, Germany: ArcTron 3D. Available at: http://www.arctron.com/3D_Surveying/3D_Laser_Scanning/index.php. Accessed 16 October 2008.

ARS Photo. 2007. Photographic pin meter. United Stated Department of Agriculture, Agriculture Research Service. USDA-ARS. Available at: www.weru.ksu.edu/new_weru/multimedia/research/research1.html. Accessed 04/01 2010.

Arvidsson, J. and E. Bolenius. 2006. Effects of soil water content during primary tillage - laser measurements of soil surface changes. *Soil tillage research* 90(1/2): 222-229.

Barnett, T. and T. Lewis. 1994. *Outliers in statistical data.* 3rd ed. New York: John Wiley & Sons.

Bertuzzi, P., J. Caussignac, P. Stengel, G. Morel, J. Lorendeau, and G. Pelloux. 1990. An automated, noncontact laser profile meter for measuring soil roughness in situ. *Soil Science* 149(3): 169-178.

Brovelli, M. A., M. Canata, and U. M. Longoni. 2004. LIDAR data filtering and DTM interpolation within GRASS. *Trans. GIS* 8(2): 155-174.

CARMS. 2006. LiDAR – overview of technology applications, market features and industry. University of Victoria, BC Canada: Centre for Applied Remote Sensing, Modeling and Simulation. Available at: http://carms.geog.uvic.ca/LiDAR%20Web%20Docs/LiDAR%20paper%20june%202006.pdf. Accessed 16 October 2008.

CFI. 2008. The REVscan laser scanner. Newark, DE: Creaform Inc. Available at: http://www.creaform3d.com/en/handyscan3d/products/revscan.aspx. Accessed 16 October 2008.

Crossbow Technology Inc. 2005a. *AHRS400 Series User's Manual*. Rev. A. San Jose, CA: Crossbow Technology, Inc.

Crossbow Technology Inc. 2005b. *AHRS400 data sheet*. Rev. A. San Jose, CA: Crossbow Technology, Inc.

CTI. 2006. Knowledge base. San Jose, CA: Crossbow Technology Inc. Available at: http://www.xbow.com/Support/iobjectDetail.aspx?id=50130000000JjN6AAK&type=Solution&page=0. Accessed 08 December 2009.

Darboux, F. and C. H. Huang. 2003. An instantaneous-profile laser scanner to measure soil surface microtopography. *Soil Science Society of America Journal* 67(1): 92-99.

Darboux, F. and C. H. Huang. 2001. Contrasting effects of surface roughness on erosion and runoff. *Soil Erosion Research for the 21st Century:* 143-146.

El-Sheimy, N., C. Valeo, and A. Habib. 2005. *Digital terrain modeling: acquisition manipulation and applications.* Portland, OR: Artech House.

Flanagan, D., C. Huang, L. Norton, and S. Parker. 1995. Laser scanner for erosion plot measurements. *Trans. ASAE* 38(3): 703-710.

Fox, F. A. and L. E. Wagner. 2001. A laser distance-based method for measuring standing residue. *Soil Erosion Research for the 21st Century:* 143-146.

Gonzalez, R. C. and R. E. Woods. 2002. Chapter 12.2.1: Matching. In *Digital Image Prcessing,* 2nd ed. 698-704. Upper Saddle River, NJ: Prentice Hall.

Gonzalez, R. C. and P. Wintz. 1987. Chapter 2.5: Imaging Geometry. In *Digital Image Processing*, 36-54. R. C. Gonzalez, ed., Menlo Park, CA: Addison-Wesley Publishing Company, Inc.

Habib, A., M. Ghanma, M. Morgan, and R. AlRuzouq. 2005. Photogrammetric and LiDAR data registration using linear features. *Photogrammetric Engineering and Remote Sensing* (71): 699-707.

Hagen, L.J. 1996. Crop residue effects on aerodynamic processes and wind erosion. *Theoretical and Applied Climatology* 54(1): 39-46.

Hagen, L. J. and D. V. Armbrust. 1992. Aerodynamic roughness and saltation trapping efficiency of tillage ridges. *Trans.ASAE* 35(4): 1179-1184.

Henning, J. G. and P. J. Radtke. 2006. Ground-based laser imaging for assessing three-dimensional forest canopy structure. *Photogrammetric Engineering and Remote Sensing* 72: 1349-1358.

Hollaus, M., W. Wagner, and K. Kraus. 2005. Airborne laser scanning and usefulness for hydrological models. *Advance in Geosciences*(5): 57-63.

Huang, C. and J. M. Bradford. 1990. Portable laser scanner for measuring soil surface roughness. *Soil Science Society of America Journal* 54(5): 1402-1406.

Jester, W. and A. Klik. 2005. Soil surface roughness measurement - methods, applicability, and surface representation. *Catena* 64(2-3): 174-192.

LabVIEW. 2004. *LabVIEW*. Ver. 7.1. Austin, TX: National Instruments Corp.

LDI. 2008. 3D laser scanning. Mineapolis, MN: Laser Design Inc. Available at: http://www.laserdesign.com/services/3d_laser_scanning/. Accessed 16 October 2008.

Lee, K. H. and R. Ehsani. 2008. Comparison of two 3D laser scanners for sensing object distances, shapes, and surface patterns. *Computers and electronics in agriculture* 60(2): 250-262.

Li, Z., Q. Zhu and C. Gold. 2004. *Digital Terrain Modeling: Principles and Methodology*. London: CRC Press.

Liu, X. 2008. Airborne LiDAR for DEM generation: some critical issues. *Progress in physical geography* 32(1): 31-49.

Loudermilk, E. L., J. K. Hiers, J. J. O'Brien, and R. J. Mitchell. 2009. Ground-based LIDAR: a novel approach to quantify fine-scale fuelbed characteristics. *International Journal of Wildland Fire* 18(8): 676-685.

Lowrance, R., S.A. Lee, J. D. Newbold, R.R. Schnabel, M.P. Groffman, M.D. Judith, etc. 1997. Water quality functions of riparian forest buffers in Chesapeake Bay watersheds. *Environmetal Management* 21(09/05): 687-712.

MATLAB. 2006. *MATLAB for Windows*. 2006b. Natick, MA: The MathWorks, Inc.

MCI. 2002. McGraw-Hill Dictionary of Scientific and Technical Terms. 6th ed. New York, NY: The McGraw-Hill Companies, Inc. Avaible at http://www.answers.com/topic/wind-erosion. Accessed 01 November 2009.

Measurement Computing Corp. 2007. *PC-CARD-DAS 16/16AO Analog I/O and Digital I/O Board User's Guide*. Rev. 1. Norton, MA: Measurement Computing Corp.

Merrill, S., C. Huang, T. Zobeck, and D. Tanaka. 2001. Use of the chain set for scale-sensitive and erosion-relevant measurement of soil surface roughness. In *Proc. of 10th international soil conservation organization meeting*. 594 - 600.

Milan, D. J., G. L. Heritage, and D. Hetherington. 2007. Application of a 3D laser scanner in the assessment of erosion and deposition volumes and channel change in a proglacial river. *Earth surface processes and landforms* 32(11): 1657.

Misra, P. and P. Enge. 2001. *Global Positioning System: Signals, Measurements and Performance*. 2nd ed. Lincoln, MA: Ganga-Jamuna Press.

Moreno, G. R., D.M.C. Alvarez, S.A. Requejo, and A. M. Tarquis. 2008a. Multifractal analysis of soil surface roughness. *Vadose zone journal* 7(2): 512-520.

Moreno, G.R., D.M.C. Alvarez, T.A. Alonso, S. Barrington, and S.A. Requejo. 2008b. Tillage and soil type effects on soil surface roughness at semiarid climatic conditions. *Soil and tillage research* 98(1): 35-44.

Nova Range Inc. 2001. *Laser Range User's Manual*. San Diego, CA: Nova Ranger, Inc.

Pfeifer, N. and Briese, C. 2007. Geometrical aspects of airborne laser scanning and terrestrial laser scanning. *International Archives of Photogrammery, Remote Sensing and Spatial Information Sciences* 36(3/W52): 311-319.

Parker Hannifin Corp. 1997. *PK2 Packaged Stepper Drive User Guide*. Rohnert Park, CA: Parker Hannifin Corp.

Parker Hannifin Corp. 2004. *ER Series Rodless Actuators*. Wadsworth, OH: Parker Hannifin Corp.

Podmore, T. and L. Huggins. 1981. An automated profile meter for surface roughness measurements [for tillage effectiveness]. *Trans. ASAE* 24(3):663-665,669.

Radke, J., M. Otterby, R. Young, and C. Onstad. 1981. A microprocessor automated rillmeter [Soil erosion]. *Trans. ASAE* 24(2): 401-404,408.

Ramirez, J. R. 2006. A new approach to relief representation. *Surveying and Land Information Science* 66(1): 19-25.

Reutebuch, S. E., H. E. Andersen, and R. J. McGaughey. 2005. Light detection and ranging (LIDAR): an emerging tool for multiple resource inventory. *Journal of Forestry* (103): 286-292.

Riesterer, J. 2008. UTM - universal transverse mercator geographic coordinate system. geospatial training and analysis cooperative. Pocatello, ID: Geospatial Training and Analysis Cooperative. Available at: http://geology.isu.edu/geostac/Field_Exercise/topomaps/utm.htm. Accessed 08 December 2009.

Robichaud, P. and M. Molnau. 1990. Measuring soil roughness changes with an ultrasonic profiler. *Trans.ASAE* 33(6): 1851-1858.

Romkens, M. and J. Wang. 1986. Effect of tillage on surface roughness. *Trans.ASAE* 29(2): 429-433.

Saleh, A. 1993. Soil roughness measurement: chain method. *Journal of soil and water conservation* 48(6): 527-529.

SAS. 2002. *SAS for Windows*. Ver. 9.1. Cary, NC: SAS Institute, Inc.

Skidmore, E. L. 1997. Comment on chain method for measuring soil roughness. *Soil Science Society of America Journal* 61(5): 1532-1533.

SMSI. 2009. Glossary of laser sensor terminology. Portland, OR: Schmitt Measurement Systems, Inc. Available at: http://www.acuitylaser.com/resources/glossary.shtml. Accessed 01 November 2008.

STC. 2008. 3D laser scanning (high definition scanning). Markham, ON Canada: Sammer Technologies Corp. Available at: http://www.sammertech.com/news/index.php. Accessed 24 October 2009.

Taconet, O. and V. Ciarletti. 2007. Estimating soil roughness indices on a ridge-and-furrow surface using stereo photogrammetry. *Soil tillage research* 93(1): 64-76.

Topcon Positioning System Inc. 2006. *Guick start guide of Topcon HiPer Lite+ with TopSURV on an FC-100*. Livemore, CA: Topcon Positioning Systems, Inc.

USGS. 2006. The Universal Transverse Mercator (UTM) grid. Reston, VA: U.S. Geological Survey. Available at: http://egsc.usgs.gov/isb/pubs/factsheets/fs07701.html. Accessed 08 December 2009.

Van der Zande, D., W. Hoet, I. Jonckheere, J. Van Aardt, and P. Coppin. 2006. Influence of measurement set-up of ground-based LiDAR for derivation of tree structure. *Agricultural and Forest Meteorology* 141(ScienceDirect): 147-160.

Wagner, L. E. and L. J. Hagen. 1991. Relationship between shelter angle surface roughness and cumulative sheltered storage depth. In International Wind Erosion Workshop of CIGR: Budapest, Hungary.

Wagner, L.E. and Y. Yu. 1991. Digitization of profile meter photographs. *Trans.ASAE* 34(2): 412-416.

Warner, W.S. 1995. Mapping a three-dimensional soil surface with hand-held 35 mm photography. *Soil & Tillage Research* 34(3): 187-197.

Watt, P. J. and N. M. Donoghue. 2005. Measuring forest structure with terrestrial laser scanning. *International Journal of Remote Sensing* 26(7): 1437-1446.

Webster, T. L., J. B. Murphy, J. G. Gosse, and I. Spooner. 2006. The application of lidar-derived elevation model analysis to geological mapping: an example from the Fundy Basin, Nova Scotia, Canada. *Can.J.Remote sensing* 32(2): 173-193.

Welch, R., T. Jordan, and A. Thomas. 1984. Photogrammetric technique for measuring soil erosion. *Journal of Soil and Water Conservation* 39(3): 191-194.

WFI. 2009. Photogrammetry. Wikimedia Foundation, Inc. Available at: http://en.wikipedia.org/w/index.php?title=Photogrammetry&oldid=315599416. Accessed 20 October 2009.

WFI. 2008. Riparian buffer. Wikimedia Foundation, Inc. Available at: http://en.wikipedia.org/w/index.php?title=Riparian_buffer&oldid=206483151. Accessed 10 October 2008.

Wilson, B. N., R. B. Leaf, and B. J. Hansen. 2001. Microrelief meter for field topography measurements. *Trans. ASABE* 44(2): 289-295.

Woolard, J. W. and J. D. Colby. 2002. Spatial characterization, resolution, and volumetric change of coastal dunes using airborne LIDAR: Cape Hatteras, North Carolina. *Geomorphology*(48): 269-287.

Zhang, N., V.O. Shanholtz, A. ReyanMcGlone, and C. Desai. 1989. Two-dimensional algorithm for resampling digital elevation model data. *Trans.ASAE* 32(4): 1319-1328.

Ziadat, F. M. 2007. Effect of contour intervals and grid cell size on the accuracy of DEMs and slope derivatives. *Trans. GIS* 11(1): 67-81.

# Appendix A - Laser Sampling Program

```
/************************************************************************
** Program:  LaserReading.cpp
** This is a data logger for Acuity - AccuRange 4000 through HSIF.
** Files:  LOOKUPHS - laser calibration file; hsiflib.lib - HISF static linked library
**       sio_util.h; sio_util.cpp; rangehs. h - Laser settings and HSIF routines
** Inputs: COM port number, HSIF Card number, Sampling Period, Data Buffer Size,Max Range,
**                      Motor Number, and Motor Power Lever. (Calibration File)
** Ouputs: *.txt: Distance, Raw range, Amplitude, Temperature, Angle1(Laser Motor),
**           Angle2(Rail Motor),Index1(Laser Encoder Index), Index2(Rail Encoder Index).
** Usage: laser_gyro_gps <Laser_file_name><COM> <Card> <SamplePeriod><Lines><Range>
**                 <Motor> <Power>
** Subroutines: Save_file();
************************************************************************/
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <process.h>
#include <windows.h>
#include <conio.h>

#include "..\hsiflib\rangehs.h"   //h.file for laser setting
#include "..\utilities\sio_util.h"  //h.file for laser setting

#define MAX_PACKET_SIZE 255
#define MAX_BUF_SIZE 255
#define DEFAULT_LASER_COMM 1  //default settings
#define DEFAULT_LASER_BOARD 1
#define DEFAULT_LASER_SAMPLE_PERIOD 20
#define DEFAULT_LASER_SAMPLE_LINES 200
#define DEFAULT_LASER_MOTOR_NUM 1
#define DEFAULT_LASER_MOTOR_POWER 255
#define EXIT_CHAR 'q'
```

```
#define NUM_SAMPLES 5000
#define HS_BUFSIZE (NUM_SAMPLES*sizeof(HSIF_SAMPLE))
HSIF_SAMPLE hsbuffer[HS_BUFSIZE/sizeof(HSIF_SAMPLE)];//Raw sample buffer
#define NUM_PROCESS 1000000
#define HPS_BUFSIZE (NUM_PROCESS*sizeof(HSIF_PROC_SAMPLE))
HSIF_PROC_SAMPLE hpSampleBuf[HPS_BUFSIZE/sizeof(HSIF_PROC_SAMPLE)];
HSIF_RESULT result;    // see manual, page26
COMMINFO Commport;   // in sio_util.h, "COMMINFO" in serial I/O routine for communication
COMMINFO * pcom = 0;

static void save_file(int lines, FILE* fp);  // save laser data
main(int argc, char *argv[])
{
        COMMINFO Commport;                          //laser data structure, in sio_util.h
        pcom = &Commport;
        HSIF_HANDLE hsif = HSIF_INVALID_HANDLE;      // handle of HSIF card
        int serialport = DEFAULT_LASER_COMM;           // serial port number
        int board = DEFAULT_LASER_BOARD;             // HSIF number
        long sample_period = DEFAULT_LASER_SAMPLE_PERIOD; // sampling period
        int lines = DEFAULT_LASER_SAMPLE_LINES;   // total lines of range data before exiting
        int max_range = DEFAULT_MAX_RANGE;            //650 inches
        int motor = DEFAULT_LASER_MOTOR_NUM;        //motor number
        int powerlevel = DEFAULT_LASER_MOTOR_POWER;   //motor power
        char *outLaserFileName;                          //point to data file
        char calibration[]="LOOKUPHS";                   //calibraion file: LOOKUPHS
        DWORD numRead;
        int hplines;
        FILE *fp;    //point to a file structuce.
        unsigned int userChar = '0';
        int currentArg;  //Argument number of Main()

        if (argc < 2) { /* not enough argument */
                printf("usage: laserreading<Laserfile><options>\n");
                printf("Available options are: \n");
                printf("-c <comm port number for Laser> where the default is 1.\n");
                printf("-b <HSIF board number> where the default is 1.\n");
                printf("-s <sampling period> where the default is 20us.\n");
                printf("-l <buffer size> where the default is 200 which means 1000000 samples \n");
                printf("-r <max range> where the default is 650 inches.\n");
                printf("-m <motor number> where the default is 1.\n");
```

```
        printf("-p <motor power level> where the default is 255, the highest speed\n");
        getchar(); // waiting key_point
        exit(1);
}

currentArg = 1;   //0 is the program name
outLaserFileName = argv[currentArg]; currentArg++;
while (argc > currentArg) {       //argument > 4 (begin)
        switch (argv[currentArg][1]) {
        case 'c': // serial port number
                currentArg++;
                serialport = atoi(argv[currentArg]); currentArg++;
                break;
        case 'b': // HSIF board number
                currentArg++;
                board = atoi(argv[currentArg]); currentArg++;
                break;
        case 's': // sampling period
                currentArg++;
                sample_period = atol(argv[currentArg]); currentArg++;
                break;
        case 'l': // data buffer size
                currentArg++;
                lines = atoi(argv[currentArg]); currentArg++;
                break;
        case 'r': // maximum measurement range
                currentArg++;
                max_range = atoi(argv[currentArg]); currentArg++;
                break;
        case 'm': // motor number
                currentArg++;
                motor = atoi(argv[currentArg]); currentArg++;
                break;
        case 'p': // motor power level
                currentArg++;
                powerlevel = atoi(argv[currentArg]); currentArg++;
                break;
        default:
                printf("unknown argument: -%c\n",argv[currentArg][1]);
                printf("type laser_gyro_gps without any parameters for help");
                break;} // end switch
} // end while
```

```
HsifDllInit();          // Initialize library, See manual, page 25
fp = fopen(outLaserFileName , "w");

if(!OpenPort(pcom, 9600, serialport)){ //Initialization: COM, HSIF Card ,in sio_util.h
        pcom = 0;
        fprintf(stderr, "Error: Serial Comm port open failed.\n");
        goto ERROR_EXIT;
}
sendstr(pcom,"I");// set to factory defaults. manually send the command to set the sensor through RS232.
Sleep(200);     //give 0.2 sec time to take effect.

fprintf(stderr, "Opened high speed interface.\n");
hsif = HsifOpen(board, &Commport); // Open HSIF, in rangehs.h
if(hsif == HSIF_INVALID_HANDLE) {
        fprintf(stderr, "Error: HsifOpen failed.\n");
        goto ERROR_EXIT;
}
if(HsifResetBoard(hsif) == HSIF_FAIL) { // Reset HSIF, in rangehs.h
        fprintf(stderr, "Error: HsifResetBoard failed.\n");
        goto ERROR_EXIT;
}
fprintf(stderr, "HISF Card Calibration\n");
HsifSetPollMode(hsif,TRUE);             // TRUE = polling allowed
if(HsifCalibrate(hsif) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Could not get calibration value\n");
        goto ERROR_EXIT;
}
fprintf(stderr,"SamplingModeInit\n");
if(HsifSamplingModeInit(hsif) != HSIF_SUCCESS){// set card to sampling mode
        fprintf(stderr,"Error: Could not initialize sampling mode\n");
        goto ERROR_EXIT;
}
if(HsifClearEncoder(hsif, ENCODER1|ENCODER2,TRUE)!= HSIF_SUCCESS){
        fprintf(stderr,"Error: Failed to clear encoder1\n");
        goto ERROR_EXIT;
} //True: take place when the index pulse occurs;  FALSE: immediately
if(HsifCalibrateEncoder(hsif, ENCODER1, 0, 4096) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Failed to set encoder1\n");
        goto ERROR_EXIT;
} // Set sample period and resolution for encoder1.
if(HsifCalibrateEncoder(hsif, ENCODER2, 0, 200) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Failed to set encoder2\n");
```

```
			goto ERROR_EXIT;
		} // Set sample period and resolution for encode2.
		if(HsifLoadCalibrationData(hsif, calibration) == HSIF_FAIL) {
			printf("ERROR: Could not read calibration file\n");
			goto ERROR_EXIT;
		}
		if(HsifSetSamplePeriod(hsif,0,max_range,sample_period)!= HSIF_SUCCESS){
			fprintf(stderr,"Error: Failed to set sample and range\n");
			goto ERROR_EXIT;
		}		// Set sample period and resolution
		HsifSetMotorPower(hsif, motor, powerlevel);
		fprintf(stderr,"Waiting (20 sec.) for mirror to move\n");
		Sleep(20000);
		fprintf(stderr, "Clear Sample Buffer\n");
		HsifClearSampleBuffer(hsif);// for clean buffer each time
		HsifSamplingEnable(hsif);			// need to skip ahead
		fprintf(stderr,"Sampling Enable\n");
		fprintf(stderr,"Begin sampling\n");
		hplines = lines;
		while((userChar != EXIT_CHAR) && (lines--)){
			if((HsifGetBufferedSamples(hsif,hsbuffer,NUM_SAMPLES,&numRead,TRUE))==HSIF_FAIL){
				printf("ERROR: Buffer overflow flag not getting set.\n");
				return(0);
			}
			if( DATA_LOST(hsbuffer[NUM_SAMPLES-1])){
				HsifClearSampleBuffer(hsif);
				HsifSamplingEnable(hsif);
			}
	result=HsifProcessSamples(hsif,hsbuffer,hpSampleBuf+(hplines-lines-1)*NUM_SAMPLES,NUM_SAMPLES);
			if(hpSampleBuf[(hplines-lines-1)*NUM_SAMPLES].timeout){
				printf("ERROR: Sample measurement timeout!\n");
				return(0);
			}
			if (kbhit()) {
				userChar=getch();
			}//kbhit()--checks the console for keyboard input, required header <conio.h>
}

		fprintf(stderr,"End sampling\n");
		fprintf(stderr,"Sampling Lines counter: %d\n", lines);
		HsifSetMotorPower(hsif, motor, 0);
		fprintf(stderr,"Saving Data,Please Waiting\n");
```

```
        save_file((hplines-lines-1), fp);
ERROR_EXIT:
        if(pcom)
                ClosePort(pcom);
        if(hsif != HSIF_INVALID_HANDLE)
                HsifClose(hsif);
        return(0);
}

/***********************************
//Function Name: save_file(int lines)
//Descriptions:Write all data from hpSampleBuf[] buffer to a .txt file (data.txt).
***********************************/
static void save_file(int lines, FILE *fp)
{
        int i;
        fprintf(fp, "Distance(inches)\t");
        fprintf(fp, "Raw Range(bits)\t");
        fprintf(fp, "Amplitude\t");
        fprintf(fp, "Ambient\t");
        fprintf(fp, "Temp(F)\t");
        fprintf(fp, "Angle1(radins)\t");
        fprintf(fp, "Angle2(radins\t");
        fprintf(fp, "Index1\t");
        fprintf(fp, "Index2\t");

  for( i=0; i < (lines*NUM_SAMPLES); i++){
                fprintf(fp,"\n");
                fprintf(fp, "%.2f\t", hpSampleBuf[i].distance);
                fprintf(fp, "%.0f\t", (double)hpSampleBuf[i].rawRange);
                fprintf(fp, "%.1f\t", hpSampleBuf[i].amplitude);
                fprintf(fp, "%.1f\t", hpSampleBuf[i].ambient);
                fprintf(fp, "%.1f\t", hpSampleBuf[i].caltemp);
                fprintf(fp, "%.4f\t", hpSampleBuf[i].angle1);
                fprintf(fp, "%.4f\t", hpSampleBuf[i].angle2);
                fprintf(fp, "%d\t", INPUT1(hpSampleBuf[i]) ? 1:0);
                fprintf(fp, "%d\t", INPUT2(hpSampleBuf[i]) ? 1:0);
        }
  if (fclose(fp))
                fprintf(stderr,"Error: Can't close the file\n");
}
```

# Appendix B - Universal Serial Communication Program

```
/************************************************************************
** Program:  Serial.cpp    (GPSreading.cpp)
** This is a universial serial communation program for RS-232.
** It initailize the port and collect data and save to txt file.
** Available options are:
**          -c <comm port> where the default is COM1
**          -b <baud rate> where the default is 9600
**          -n <number of bits in the bytes> where the default is 8
**          -p <parity > where the default is NOPARITY (0)
**          -s <the number of the stop bits> where the default is ONESTOPBIT (1)
** Subroutines: Output();
**********************************************************************/
#include <stdio.h>
#include <windows.h>
#include <conio.h>  // console function

#define MAX_BUF_SIZE 255
#define EXIT_CHAR 'q'
static void Output(unsigned char *comBuf, unsigned long bytesRead, FILE* out);// save data

void main (int argc, char *argv[])
{
        char *outFileName;                      //a pointer to output file name
        FILE *out;
        int currentArg;
        char *comPort="COM1";                   //Seiral port, default is COM1
        unsigned int baudRate = CBR_9600;     //Baud Rate, default is 9600, 'CBR'-Key words
        unsigned int byteSize = 8;                      // number of bits in the bytes transmitted and received
        unsigned int parity = 0;
        unsigned int stopBit = 0;
        unsigned char comBuf[MAX_BUF_SIZE] = "";  //Comm buffer
        unsigned int userChar= '0';
        unsigned long bytesRead = 0;
        DWORD dwCommEvtMask;
        DWORD dwErrorMask;
```

```
DWORD nToRead;
DWORD dwRes;
LPDCB serialCtrl= new DCB;              //DCB structure in MSDN,LPDCB
HANDLE serialPort;
COMMTIMEOUTS commTimeouts;
COMSTAT comstat;
OVERLAPPED ol = {0};

if (argc < 2) { /* not enough argument */
        printf("usage: gpsreading <file> <options>\n");
        printf("Available options are: \n");
        printf("-c <comm port> where the default is COM1\n");
        printf("-b <baud rate> where the default is 9600\n");
        printf("-n <number of bits in the bytes> where the default is 8\n");
        printf("-p <patiry > where the default is NOPARITY (0)\n");
        printf("-s <the number of the stop bits> where the default is ONESTOPBIT (1)\n");
        printf("  Parity values: 0 -- No parity; 1 -- Odd; 2 -- Even; 3 -- Mark; 4 -- Space.\n");
        printf("  StopBits values: 0 -- 1 stop bit; 1 -- 1.5 stop bits; 2 -- 2 stop bits.\n");
        getchar();
        exit(1);
}
currentArg = 1; // 0 is the program name
outFileName = argv[currentArg]; currentArg++;
while (argc > currentArg) { //argument > 2
        switch (argv[currentArg][1]) {
        case 'c':
                currentArg++;
                comPort = argv[currentArg];currentArg++;
                break;
        case 'b':
                currentArg++;
                baudRate = atoi(argv[currentArg]); currentArg++;
                break;
        case 'n':
                currentArg++; // atof() - convert strings to double
                byteSize =  atoi(argv[currentArg]); currentArg++;
                break;
```

```
		case 'p':
			currentArg++;
			parity=atoi(argv[currentArg]); currentArg++;
			break;
		case 's':
			currentArg++;
			stopBit =atoi(argv[currentArg]); currentArg++;
			break;
		default:
			printf("unknown argument: -%c\n",argv[currentArg][1]);
			printf("type gpsreading without any parameters for help");
			break;}
	}
	switch (parity){
		case 0: parity = NOPARITY; break;
		case 1: parity = ODDPARITY; break;
		case 2: parity = EVENPARITY; break;
		case 3: parity = MARKPARITY; break;
		case 4: parity = SPACEPARITY; break;
		default: parity = NOPARITY; break;
	}
	switch (stopBit){
		case 0: stopBit = ONESTOPBIT; break;
		case 1: stopBit = ONE5STOPBITS; break;
		case 2: stopBit = TWOSTOPBITS; break;
		default: stopBit = ONESTOPBIT; break;
	}

	out = fopen(outFileName, "w");
	serialPort=CreateFile(comPort,GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXISTIN
G,FILE_FLAG_OVERLAPPED,NULL);// OverLapped Mode
	if (serialPort == INVALID_HANDLE_VALUE) {
		printf("could not open serial port: %d\n", GetLastError());
		exit(1);
	}
	if (!(SetupComm(serialPort, 1024, 1024))) {  // Set input  and output buffer size:
		printf( "error setting up comm: %d\n", GetLastError());
```

```
        CloseHandle(serialPort);
        exit(1);
    }
    if (!(GetCommState(serialPort, serialCtrl))) {  //get serial port state
        printf("error getting comm: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
    }
    serialCtrl->BaudRate = baudRate;    //configure the serial port parameters.
    serialCtrl->ByteSize = byteSize;
    serialCtrl->Parity = parity;
    serialCtrl->StopBits = stopBit;
    if (!(SetCommState(serialPort, serialCtrl))) { //reset serial port state
        printf("error setting comm: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
    }
    if (!GetCommTimeouts(serialPort, &commTimeouts)){ // get the serial port timeout
        printf("error getting comm timeouts: %d\n",GetLastError());
        CloseHandle(serialPort);
        exit(1);
    }
    commTimeouts.ReadIntervalTimeout = MAXDWORD; //interval time between the two bytes.
    commTimeouts.ReadTotalTimeoutMultiplier =MAXDWORD;
    commTimeouts.ReadTotalTimeoutConstant = 100;
    if( !SetCommTimeouts(serialPort, &commTimeouts)){   //reset the serial port timeout
        printf("error setting comm timeouts: %d\n",GetLastError());
        CloseHandle(serialPort);
        exit(1);
    }
    if (!SetCommMask(serialPort, EV_RXCHAR)) {  // specify a event to monitor the serialPort.
        printf("could not set a Mask for serial port: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
    }
    if(!PurgeComm(serialPort,PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURG
E_RXCLEAR)){
```

```
            printf("error purge comm: %d\n",GetLastError());
            CloseHandle(serialPort);
            exit(1);
}// clear all buffer
ol.hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
if (ol.hEvent==NULL){
            printf("could not set event: %d\n", GetLastError());
            exit(1);
}
while(userChar != EXIT_CHAR) {
    if (WaitCommEvent(serialPort, &dwCommEvtMask, NULL)){ //waiting for moniting event
        ClearCommError(serialPort,&dwErrorMask,&comstat);  // get data length
        if ((dwCommEvtMask & EV_RXCHAR) && comstat.cbInQue){// receive
            if(comstat.cbInQue > 255)
                nToRead = 255;
            else
                nToRead = comstat.cbInQue;
            if (nToRead ==0)
                continue;
            if (ReadFile(serialPort,comBuf,nToRead,&bytesRead,&ol)==0){
                if(GetLastError()== ERROR_IO_PENDING){
                    dwRes = WaitForSingleObject(ol.hEvent,500);
                    switch (dwRes){
                        case WAIT_OBJECT_0:
                        if (!GetOverlappedResult(serialPort, &ol, &bytesRead,FALSE))
                            printf("Error getting Overlapped result: %d\n", GetLastError());
                        else{
                            if(bytesRead){ // no data print here
                                Output(comBuf,bytesRead,out);}
                        }
                        break;
                        case WAIT_TIMEOUT:
                            printf("TimeOut for waiting single object: %d\n", GetLastError());
                            break;
                        case WAIT_FAILED:
                            printf("error waiting for single object: %d\n", GetLastError());
                            break;
```

```
                                    default:
                                     break;
                                  }
                              }
                              else
                        printf("Reading of serial communication has problem: %d\n", GetLastError());
                  }
                  else{
                      if (bytesRead){  // data  print here
                           Output(comBuf,bytesRead,out);
                   }
             }
         }
     }
     else{
           if(GetLastError()== ERROR_IO_PENDING){
            continue;
            }
            else
                printf("Error for Waiting Comm Event: %d\n", GetLastError());
     }
     if (kbhit()) { //kbhit()--checks the console for keyboard input, required header <conio.h>
           userChar=getch();}  //getch()--get char for keyboard input, required header <conio.h>
     }
     if (CloseHandle(serialPort) == 0 )
               printf("Port  Closing isn't successed: %d\n", GetLastError());
     if (CloseHandle(ol.hEvent) == 0 )
               printf("Event Closing isn't successed: %d\n", GetLastError());
     if (fclose(out))
               fprintf(stderr,"Error: Can't close the file\n");
}

/************************
// Function Name: Output(unsigned char *comBuf, unsigned long bytesRead, FILE* out)
//Descriptions: Write all data from comBuf[] buffer to a .txt file .
************************/
static void Output(unsigned char *comBuf, unsigned long bytesRead, FILE* out)
```

```
{
	unsigned long i;
	for (i=0; i<bytesRead; i++){
		fprintf(out,"%c",comBuf[i]);
	}
}
```

# Appendix C - Main Program

```
/***********************************************************************
** Program:  Laser_gyro_gps.cpp
** Source Files: LOOKUPHS - laser calibration file; hsiflib.lib - HISF  static linked library
**           sio_util.h; sio_util.cpp; rangehs. h - Laser settings and HSIF routines
** Inputs: Laser file name, Gyro file name, GPS file name, COM port number for laser,
**       HISF card number, Sampling period, Buffer size, Max measurement range, Motor Number, **
Motor Power Lever; Operating Mode of gyro, COM port number for gyro ;
**                       Baud rate of COM for gyro, COM port number for GPS, baud rate of COM for GPS,
**      Bits number of COM for GPS, Parity control of COM for GPS, Stop bit of COM for GPS
** Ouputs: Three data file for Laser, Gyroscope, and GPS, respectively.
** Subroutines: Save_file(); restart_system(); check_packet_valid(); get_packet();output_packet();
**          output_GPS()
** Description: This is a data logger for Acuity laser scanner, Crossbow - AHRS400, and GPS systems.
***********************************************************************/
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <process.h>
#include <errno.h>
#include <windows.h>
#include <conio.h>
#include <sys/types.h>
#include <sys/timeb.h>

#include "..\hsiflib\rangehs.h"
#include "..\utilities\sio_util.h"
#include "laser_gyro_gps.h"

static void save_file(int lines, FILE* fp);     //save laser data
static bool check_packet_valid(char mode,unsigned char* packet, unsigned int packetSize);
static void get_packet(HANDLE fp, unsigned int packetSize, unsigned char *packet);
static void output_packet(char mode,double dt, unsigned int packetCnt,
unsigned char* packet, bool isDOSTIME, FILE* out);
```

```
static void restart_system(HANDLE serialPort,char mode, unsigned int *packetSize);
static void output_GPS(unsigned char *comBuf,unsigned long bytesRead,FILE* outGPS);

main(int argc, char *argv[])
{   // Laser parameters
        COMMINFO Commport;
    pcom = &Commport;
    HSIF_HANDLE hsif = HSIF_INVALID_HANDLE; // handle of HSIF card,
    int  serialport = DEFAULT_LASER_COMM;
    int  board = DEFAULT_LASER_BOARD;
    long sample_period = DEFAULT_LASER_SAMPLE_PERIOD;
    int  lines = DEFAULT_LASER_SAMPLE_LINES;
    int max_range = DEFAULT_MAX_RANGE;
    int motor = DEFAULT_LASER_MOTOR_NUM;
    int powerlevel = DEFAULT_LASER_MOTOR_POWER;
    char  *outLaserFileName;
    char calibration[]="LOOKUPHS";//calibraion file: LOOKUPHS
    DWORD    numRead;
    int hplines;

   //Gyro  parameters
    char *outGyroFileName;
    unsigned char packet[MAX_PACKET_SIZE]; //buffer the transfered and received data
    FILE *out, *fp;
    char *comPort = "COM2";
    unsigned int baudRate = CBR_38400;
    char mode = ANGLE_MODE; //Measurement Modes, defualt is Angle mode.
    unsigned long decimationRate=1;
    bool outputData=FALSE, badPacket=FALSE;
    bool isDOSTIME = FALSE;
    unsigned int packetSize=0;
    unsigned int packetCnt =0;
    unsigned int badpacketCnt =0;
    unsigned int i;
    double dt=-1;
    LPDCB serialCtrl= new DCB;    //DCB structure in MSDN,LPDCB
    HANDLE serialPort;
```

```
COMMTIMEOUTS commTimeouts;

//GPS parameters
char *outGPSFileName;
FILE *outGPS;
char *comGPSPort="COM3";
unsigned int baudRate_GPS = CBR_9600;
unsigned int byteSize = 8;
unsigned int parity = 0;
unsigned int stopBit = 0;
unsigned char comBuf[MAX_BUF_SIZE] = ""; //Comm buffer
unsigned long bytesRead = 0;
DWORD dwCommEvtMask;
DWORD dwErrorMask;
DWORD nToRead;
DWORD dwRes;
COMSTAT comstat;
OVERLAPPED ol = {0};
LPDCB GPSserialCtrl= new DCB;    //DCB structure in MSDN,LPDCB
HANDLE GPSserialPort;
COMMTIMEOUTS GPScommTimeouts;

//common
unsigned int userChar = '0';
int currentArg; //Argument number of Main()
printf(VERSION_STRING);

if (argc < 4) { // not enough argument
        printf("usage: laser_gyro_gps <Laserfile><Gyrofile><GPSfile> <options>\n");
        printf("Available options are: \n");
        printf("-c <comm port number for Laser> where the default is 1.\n");
        printf("-b <HSIF board number> where the default is 1.\n");
        printf("-s <sampling period> where the default is 20us.\n");
        printf("-l <buffer size> where the default is 200 which means 1000000 samples (15M).\n");
        printf("-r <max range> where the default is 650 inches.\n");
        printf("-m <motor number> where the default is 1.\n");
        printf("-p <motor power level> where the default is 255 which means the highest speed\n");
```

```
        printf("-M <mode for Gyro> where <mode> is a, r, or c. Default is angle mode (a).\n");
        printf("-C <comm port for Gyro> where the default is COM2 (2)\n");
        printf("-B <baud rate for Gyro> where the default is 38400 (38400)\n");
        printf("-S <comm port for GPS> where the default is COM3\n");
        printf("-R <baud rate for GPS> where the default is 9600\n");
        printf("-N <number of bits in the bytes> where the default is 8\n");
        printf("-P <patiry > where the default is NOPARITY (0)\n");
        printf("    Parity values: 0 -- No parity; 1 -- Odd; 2 -- Even; 3 -- Mark; 4 -- Space.\n");
        printf("-n <the number of the stop bits> where the default is ONESTOPBIT (1)\n");
        printf("    StopBits values: 0 -- 1 stop bit; 1 -- 1.5 stop bits; 2 -- 2 stop bits.\n");
        getchar(); // waiting key_point
        exit(1);
}

currentArg = 1; //0 is the program name
outLaserFileName = argv[currentArg]; currentArg++;
outGyroFileName = argv[currentArg]; currentArg++;
outGPSFileName = argv[currentArg]; currentArg++;
while (argc > currentArg) {      //argument > 4 (begin)
        switch (argv[currentArg][1]) {
        case 'c': // serial port number
                currentArg++;
                serialport = atoi(argv[currentArg]); currentArg++;
                break;
        case 'b': // board number
                currentArg++;
                board = atoi(argv[currentArg]); currentArg++;
                break;
        case 's': // sample period
                currentArg++;
                sample_period = atol(argv[currentArg]); currentArg++;
                break;
        case 'l': // sample lines
                currentArg++;
                lines = atoi(argv[currentArg]); currentArg++;
                break;
        case 'r': // max range
```

```
        currentArg++;
        max_range = atoi(argv[currentArg]); currentArg++;
        break;
case 'm': // motor number
        currentArg++;
        motor = atoi(argv[currentArg]); currentArg++;
        break;
case 'p': // motor power level
        currentArg++;
        powerlevel = atoi(argv[currentArg]); currentArg++;
        break;
case 'M': // mode
        currentArg++;
        mode = argv[currentArg][0]; currentArg++;
        break;
case 'C': // comm port
        currentArg++;
        comPort= argv[currentArg]; currentArg++;
        break;
case 'B': // baud rate
        currentArg++;
        baudRate = atoi(argv[currentArg]); currentArg++;
        break;
case 'S':
        currentArg++;
        comGPSPort= argv[currentArg]; currentArg++;
        break;
case 'R':
        currentArg++;
        baudRate_GPS = atoi(argv[currentArg]); currentArg++;
        break;
case 'N':
        currentArg++;
        byteSize =  atoi(argv[currentArg]); currentArg++;
        break;
case 'P':
        currentArg++;
```

```
                parity=atoi(argv[currentArg]); currentArg++;
                break;
            case 'n':
                currentArg++;
                stopBit =atoi(argv[currentArg]); currentArg++;
                break;
            default:
                printf("unknown argument: -%c\n",argv[currentArg][1]);
                printf("type laser_gyro_gps without any parameters for help");
                break;
            } // end switch
        } // end while
        //GPS
        switch (parity){
                case 0: parity = NOPARITY; break;
                case 1: parity = ODDPARITY; break;
                case 2: parity = EVENPARITY; break;
                case 3: parity = MARKPARITY; break;
                case 4: parity = SPACEPARITY; break;
                default: parity = NOPARITY; break;
        }
        switch (stopBit){
                case 0: stopBit = ONESTOPBIT; break;
                case 1: stopBit = ONE5STOPBITS; break;
                case 2: stopBit = TWOSTOPBITS; break;
                default: stopBit = ONESTOPBIT; break;
        }
        outGPS = fopen(outGPSFileName, "w");
        GPSserialPort=CreateFile(comGPSPort,GENERIC_READ|GENERIC_WRITE,0,NULL,OPEN_EXI
STING,FILE_FLAG_OVERLAPPED,NULL); // OverLapped Mode
        if (GPSserialPort == INVALID_HANDLE_VALUE) {
            printf("GPS - could not open serial port: %d\n", GetLastError());
            exit(1);
        }
        if (!(SetupComm(GPSserialPort, 1024, 1024))) {  // Set input and output buffer size:
            printf( "GPS - error setting up comm: %d\n", GetLastError());
            CloseHandle(GPSserialPort);
```

```
    exit(1);
}
if (!(GetCommState(GPSserialPort, GPSserialCtrl))) {  //get serial port state
    printf("GPS - error getting comm: %d\n", GetLastError());
    CloseHandle(GPSserialPort);
    exit(1);
}
GPSserialCtrl->BaudRate = baudRate_GPS;    //configure the serial port parameters.
GPSserialCtrl->ByteSize = byteSize;
GPSserialCtrl->Parity = parity;
GPSserialCtrl->StopBits = stopBit;
if (!(SetCommState(GPSserialPort, GPSserialCtrl))) { //reset serial port state
    printf("GPS - error setting comm: %d\n", GetLastError());
    CloseHandle(GPSserialPort);
    exit(1);
}
if (!GetCommTimeouts(GPSserialPort, &GPScommTimeouts)){ // get the serial port timeout
    printf("GPS - error getting comm timeouts: %d\n",GetLastError());
    CloseHandle(GPSserialPort);
    exit(1);
}
GPScommTimeouts.ReadIntervalTimeout = MAXDWORD;
GPScommTimeouts.ReadTotalTimeoutMultiplier =MAXDWORD;
GPScommTimeouts.ReadTotalTimeoutConstant = 100;
if( !SetCommTimeouts(GPSserialPort, &GPScommTimeouts)){ // reset the serial port timeout
    printf("GPS - error setting comm timeouts: %d\n",GetLastError());
    CloseHandle(GPSserialPort);
    exit(1);
}
if (!SetCommMask(GPSserialPort, EV_RXCHAR)) { // specify a event to monitor the serialPort.
    printf("GPS - could not set a Mask for serial port: %d\n", GetLastError());
    CloseHandle(GPSserialPort);
    exit(1);
}
If (!PurgeComm (GPSserialPort,
    PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR)){
        printf("GPS - error purge comm: %d\n",GetLastError());
```

```
        CloseHandle(GPSserialPort);
        exit(1);
} // clear all buffer
ol.hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
if (ol.hEvent==NULL){
   printf("GPS - could not set event: %d\n", GetLastError());
   exit(1);
}
printf("GPS record is starting now!\n");
printf("For stop, Press 'Q' !\n");
while(userChar != EXIT_CHAR) {
   if (WaitCommEvent(GPSserialPort,&dwCommEvtMask,NULL)){
   ClearCommError(GPSserialPort,&dwErrorMask,&comstat);// get data length
   if ((dwCommEvtMask & EV_RXCHAR) && comstat.cbInQue){
           if(comstat.cbInQue > 255)
              nToRead = 255;
           else
               nToRead = comstat.cbInQue;
           if (nToRead ==0)
                 continue;
           if (ReadFile(GPSserialPort,comBuf,nToRead,&bytesRead,&ol)==0){
                if(GetLastError()== ERROR_IO_PENDING){
                      dwRes = WaitForSingleObject(ol.hEvent,500);
                      switch (dwRes){
                      case WAIT_OBJECT_0:
                           if (!GetOverlappedResult(GPSserialPort,&ol,&bytesRead,FALSE))
                               printf("GPS - Error getting Overlapped result: %d\n",GetLastError());
                            else{
                               if(bytesRead){ // no data print here
                               output_GPS(comBuf,bytesRead,outGPS);
                               }
                           }
                           break;
                      case WAIT_TIMEOUT:
                           printf("GPS - TimeOut for waiting single object: %d\n",GetLastError());
                           break;
                      case WAIT_FAILED:
```

```
                        printf("GPS - error waiting for single object: %d\n",GetLastError());
                        break;
                    default:
                        break;
                    }
                }
                else
                printf("GPS - Reading of serial communication has problem: %d\n",GetLastError());
            }
            else{
                if (bytesRead){
                    output_GPS(comBuf,bytesRead,outGPS);
                }
        }
    }
}
else{
    if(GetLastError()== ERROR_IO_PENDING){
        continue;
    }
    else
        printf("GPS - Error for Waiting Comm Event: %d\n", GetLastError());
    }
    if (kbhit()) {
        userChar=getch();}
}
if (CloseHandle(GPSserialPort) == 0 )
    printf("GPS - Port Closing isn't successed: %d\n", GetLastError());
if (CloseHandle(ol.hEvent) == 0 )
    printf("GPS - Event Closing isn't successed: %d\n", GetLastError());
if (fclose(outGPS))
    fprintf(stderr,"GPS - Error: Can't close the file\n");
userChar = '0';

//Gyro
out = fopen(outGyroFileName, "wb");
serialPort = CreateFile(comPort,GENERIC_READ|GENERIC_WRITE,0,
```

```
                    NULL, OPEN_EXISTING, FILE_ATTRIBUTE_SYSTEM, NULL);
if (serialPort == INVALID_HANDLE_VALUE) {
        printf("Gyro - could not open serial port, %d\n", GetLastError());
        exit(1);
}
if (!(SetupComm(serialPort,0x1000, 0x1000))) {
        fprintf(stderr, "Gyro - error setting up comm: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
}//set internal buffer size, in bytes.
if (!(GetCommState(serialPort, serialCtrl))) {
        fprintf(stderr, "Gyro - error getting comm: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
}
serialCtrl->BaudRate = baudRate;
serialCtrl->Parity = NOPARITY;
serialCtrl->StopBits = ONESTOPBIT;
if (!(SetCommState(serialPort, serialCtrl))) {
        fprintf(stderr, "Gyro - error setting comm: %d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
}
if(!(GetCommTimeouts(serialPort, &commTimeouts))){
        fprintf(stderr, " Gyro - error getting comm timeouts:%d/n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
};
commTimeouts.ReadIntervalTimeout = MAXDWORD;
commTimeouts.ReadTotalTimeoutMultiplier = MAXDWORD;
commTimeouts.ReadTotalTimeoutConstant = 5;
if(!(SetCommTimeouts(serialPort, &commTimeouts))){
        fprintf(stderr, "Gyro - error setting comm timeouts:%d\n", GetLastError());
        CloseHandle(serialPort);
        exit(1);
}
printf("Gyro record is starting now!\n");
```

```
printf("For stop, Press 'Q' !\n");
if(mode == ANGLE_MODE) {
        printf(" ROLL PITCH YAW GYROX GYROY  GYROZ ACCELX ACCELY ACCELZ
MAGX MAGY MAGZ TEMP\n");
}
else {
        printf("GYROX GYROY GYROZ ACCELX ACCELY ACCELZ MAGX MAGY
MAGZ TEMP\n");
}
restart_system(serialPort, mode, &packetSize);
while(userChar != EXIT_CHAR) {
    for (i=0; i<decimationRate; i++){
        get_packet(serialPort, packetSize, packet);
        if (!check_packet_valid(mode, packet, packetSize)) {
            if(badpacketCnt == 5) restart_system(serialPort, mode, &packetSize);
            badpacketCnt++; /* at least one packet to restart system */
            badPacket = TRUE;
        }
        else {
            badPacket = FALSE;
            badpacketCnt = 0;
        }
        packetCnt++;  // number of packets
    }
    if (!badPacket) {
        output_packet(mode, dt, packetCnt, packet, isDOSTIME, out);
    }
    if (kbhit()) {
        userChar=getch();
    }
} // end while
if (CloseHandle(serialPort) == 0 )
        fprintf(stderr,"Gyro - Port  Closing isn't successed: %d\n", GetLastError());
if (fclose(out))
        fprintf(stderr,"Gyro - Error: Can't close the file\n");
userChar = '0';
```

```
//Laser
HsifDllInit();
fp = fopen(outLaserFileName , "w");
if(!OpenPort(pcom, 9600, serialport)){ //initialize Serialport. About Serial Port see sio_util.h
        pcom = 0;
        fprintf(stderr, "Error: Serial Comm port open failed.\n");
        goto ERROR_EXIT;
}
sendstr(pcom,"I");// set to factory defaults
Sleep(200);
fprintf(stderr, "Opened high speed interface.\n");
hsif = HsifOpen(board, &Commport);   // see rangehs.h about HsifOpen
if(hsif == HSIF_INVALID_HANDLE) {
        fprintf(stderr, "Error: HsifOpen failed.\n");
        goto ERROR_EXIT;
}
if(HsifResetBoard(hsif) == HSIF_FAIL) {
        fprintf(stderr, "Error: HsifResetBoard failed.\n");
        goto ERROR_EXIT;
}
fprintf(stderr, "HISF Card Calibration\n");
HsifSetPollMode(hsif,TRUE);
if(HsifCalibrate(hsif) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Could not get calibration value\n");
        goto ERROR_EXIT;
}
fprintf(stderr,"SamplingModeInit\n");
if(HsifSamplingModeInit(hsif) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Could not initialize sampling mode\n");
        goto ERROR_EXIT;
}
if(HsifClearEncoder(hsif, ENCODER1|ENCODER2 , TRUE)!= HSIF_SUCCESS){
        fprintf(stderr,"Error: Failed to clear encoder1\n");
        goto ERROR_EXIT;
}
if(HsifCalibrateEncoder(hsif, ENCODER1, 0, 4096) != HSIF_SUCCESS){
        fprintf(stderr,"Error: Failed to set encoder1\n");
```

```
            goto ERROR_EXIT;
}           // Set sample period and resolution for encoder1.
if(HsifCalibrateEncoder(hsif, ENCODER2, 0, 200) != HSIF_SUCCESS){
            fprintf(stderr,"Error: Failed to set encoder2\n");
            goto ERROR_EXIT;
}           // Set sample period and resolution for encode2.
if(HsifLoadCalibrationData(hsif, calibration) == HSIF_FAIL) {
            printf("ERROR: Could not read calibration file\n");
            goto ERROR_EXIT;
}
if(HsifSetSamplePeriod(hsif,0,max_range,sample_period)!= HSIF_SUCCESS){
            fprintf(stderr,"Error: Failed to set sample and range\n");
            goto ERROR_EXIT;
}           // Set sample period and resolution
HsifSetMotorPower(hsif, motor, powerlevel);
fprintf(stderr,"Waiting (20 sec.) for mirror to move\n");
Sleep(20000);
fprintf(stderr, "Clear Sample Buffer\n");
HsifClearSampleBuffer(hsif);        // for clean buffer each time
HsifSamplingEnable(hsif);           // need to skip ahead
fprintf(stderr,"Sampling Enable\n"); //Sleep(DEBUG_SLEEP_DELAY);
fprintf(stderr,"Begin sampling\n");
hplines = lines;
while((userChar != EXIT_CHAR) && (lines--)){
if((HsifGetBufferedSamples(hsif,hsbuffer,NUM_SAMPLES,&numRead,TRUE))==HSIF_FAIL){
                        printf("ERROR: Buffer overflow flag not getting set.\n");
                        return(0);
            }
        if( DATA_LOST(hsbuffer[NUM_SAMPLES-1])){
                        HsifClearSampleBuffer(hsif);
                        HsifSamplingEnable(hsif);
       }
       result =HsifProcessSamples(hsif,hsbuffer,hpSampleBuf+(hplines-lines-1)*NUM_SAMPLES,
                                                NUM_SAMPLES);
       if(hpSampleBuf[(hplines-lines-1)*NUM_SAMPLES].timeout){
            printf("ERROR: Sample measurement timeout!\n");
            return(0);
```

```
            }
            if (kbhit()) {//kbhit()--checks the console for keyboard input, required header <conio.h>
                userChar=getch();
        }
    }
        fprintf(stderr,"End sampling\n");
        fprintf(stderr,"Sampling Lines counter: %d\n", lines);
        HsifSetMotorPower(hsif, motor, 0);
        fprintf(stderr,"Saving Data,Please Waiting\n");
        save_file((hplines-lines-1), fp);

ERROR_EXIT:
        if(pcom)
                ClosePort(pcom);
        if(hsif != HSIF_INVALID_HANDLE)
                HsifClose(hsif);
        return(0);
}

/**********************************
//      Function Name: save_file(int lines)
//      Descriptions: Write all data from hpSampleBuf[] buffer to a .txt file (data.txt).
**********************************/
static void save_file(int lines, FILE *fp)
{
        int i;
        fprintf(fp, "Distance(inches)\t");
        fprintf(fp, "Raw Range(bits)\t");
        fprintf(fp, "Amplitude\t");
        fprintf(fp, "Temp(F)\t");
        fprintf(fp, "Angle1(radins)\t");
        fprintf(fp, "Angle2(radins\t");
        fprintf(fp, "Index1\t");
        fprintf(fp, "Index2\t");
        for( i=0; i < (lines*NUM_SAMPLES); i++){
                fprintf(fp,"\n");
                fprintf(fp, "%.2f\t", hpSampleBuf[i].distance);
```

```
            fprintf(fp, "%.0f\t", (double)hpSampleBuf[i].rawRange);
            fprintf(fp, "%.1f\t", hpSampleBuf[i].amplitude);
            fprintf(fp, "%.1f\t", hpSampleBuf[i].caltemp);
            fprintf(fp, "%.4f\t", hpSampleBuf[i].angle1);
            fprintf(fp, "%.4f\t", hpSampleBuf[i].angle2);
            fprintf(fp, "%d\t", INPUT1(hpSampleBuf[i]) ? 1:0);
            fprintf(fp, "%d\t", INPUT2(hpSampleBuf[i]) ? 1:0);
        }
  if (fclose(fp))
            fprintf(stderr,"Error: Can't close the file\n");
}

/***********************************
// Function Name: check_packet_valid()
// Descriptions: Check gyro data packect to start sampling.
***********************************/
static bool check_packet_valid(char mode, unsigned char* packet,unsigned int packetSize)
{
        unsigned int i;
        unsigned int checksum = 0;
        bool retVal = TRUE;  // check packet data valid
        switch (mode) {
        case RAW_MODE:  // 24
           if ( packetSize != AHRS_RAW_PACKET_SIZE ) {
                printf("packet size incorrect: %d should be %d for AHRS\n",
                                packetSize, AHRS_RAW_PACKET_SIZE);
                return FALSE;}
        break;
        case COOKED_MODE:  //24
           if (packetSize != AHRS_COOKED_PACKET_SIZE){
                printf("packet size incorrect: %d should be  %d for AHRS\n",
                                packetSize, AHRS_COOKED_PACKET_SIZE);
                return FALSE;}
        break;
        case ANGLE_MODE:  //30
            if (packetSize != AHRS_ANGLE_PACKET_SIZE){
                printf("packet size incorrect: %d should be %d for AHRS\n",
```

```
                    packetSize, AHRS_ANGLE_PACKET_SIZE);
        return FALSE;}
    break;
    default:
        printf("\nunrecognized packet type: %c", mode);
        return FALSE;
    } /* end switch */
    //check the packet header
    if (packet[0] != NORMAL_HEADER) {
        printf("Gyro \nunrecognized header byte: ");
        retVal = FALSE;
    }
    //check the packet checksum
    for (i=1; i<packetSize-1; i++) {
        checksum += packet[i];
    }// Sum all packet contents except header and checksum
    if ( packet[i] != (checksum & 0xFF)){
        printf("\nbad checksum:");
        retVal = FALSE;
    }
    if (retVal==FALSE) {
        for (i=0; i<packetSize; i++)
            printf("%x ", packet[i] &0xFF);
        printf("\n");
        return FALSE;
    }
    else {
        return TRUE;
    }
} // is packet valid end

/******************************
// Function Name: get_packet()
// Descriptions: Get one packet data from gyroscrope for each calling.
*******************************/
static void get_packet(HANDLE fp, unsigned int packetSize, unsigned char *recvBuf)
{
```

```
	unsigned int i;
	unsigned char tmpBuf[MAX_PACKET_SIZE];
	unsigned long bytesRead, byteCnt, bytesToRead;
	byteCnt = 0;
	while (byteCnt < packetSize) {
		bytesToRead = packetSize-byteCnt;
		ReadFile(fp, tmpBuf, bytesToRead, &bytesRead, NULL);
		for(i=0; i<bytesRead; i++)
			recvBuf[byteCnt + i] = tmpBuf[i];
		byteCnt += bytesRead;
	}
} /* end get_packet */

/*****************************
//	Function Name: output_packet()
//	Descriptions:	save gyro data into a file
*****************************/
static void output_packet(char mode, double dt, unsigned int packetCnt,
				unsigned char* packet, bool isDOSTIME, FILE* out)
{
	unsigned int i, j;
	static double initialTime, thisTime;
	static int firstTime = 1;
	static unsigned tmp[50][25];
	static int tmpCnt=0;
	double elapsedTime;
	double accel[3], gyro[3], temp, roll, pitch, mag[3], yaw;
	unsigned int hdxTime;
	unsigned short partNumber=0,bit=0;
	static int adjustTime =1;
	static unsigned long packetCntAdjust =0;
	static double lastOutputTime=0.0;

	if (dt < 0) {
		if (isDOSTIME) {
			thisTime = (double)clock() / (double)CLOCKS_PER_SEC;
			fprintf(out, "%d %f ", packetCnt, thisTime);
```

```
                fileOutputRate = 0.0;
            }
            else {
                fprintf(out, "%d ", packetCnt);
                fileOutputRate = 0.0;
            }
    }
    else {
            if (isDOSTIME) {
                thisTime = (double)clock() / (double)CLOCKS_PER_SEC;
                fprintf(out, "%d %f ", packetCnt, thisTime);
                fileOutputRate = 0.0;
            }
            else {
                thisTime = (double)clock() / (double)CLOCKS_PER_SEC;
                if (firstTime) {  // firstTime=1
                    initialTime = thisTime;
                    lastOutputTime = initialTime;
                    firstTime = 0;
                }
                elapsedTime = dt*(packetCnt-packetCntAdjust) + initialTime;
                if((elapsedTime-lastOutputTime) < fileOutputRate )
                    return;
                else
                    lastOutputTime = elapsedTime;
                if (elapsedTime+(dt*4) < thisTime) {
                    packetCntAdjust = packetCnt;
                    initialTime = thisTime;
                    elapsedTime = thisTime;
                    lastOutputTime = elapsedTime;
                }
                fprintf(out, "%d %f ", packetCnt, elapsedTime);
            }
    }
    switch (mode) {
    case RAW_MODE:{
i=1;
```

```
                for (j=0; j<3; j++) {
                        gyro[j]=((unsigned short)((packet[i]<<8)+packet[i+1]))*gSensorVoltConversion;
                         i+=2;}
                for (j=0; j<3; j++) {
                       accel[j]=((unsigned  short)((packet[i]<<8)+packet[i+1]))*gSensorVoltConversion;
                         i+=2;}
                for (j=0; j<3; j++) {
                        mag[j]=((unsigned  short)((packet[i]<<8)+packet[i+1]))*gSensorVoltConversion;
                         i+=2;}
                temp=((((packet[i]<<8)+packet[i+1])*5.0/4096.0)-1.375)*44.44;
                i+=2;
                hdxTime = (packet[i]<<8)+packet[i+1];
                fprintf(out, "%f %f %f ", gyro[0], gyro[1], gyro[2]);
                fprintf(out, "%f %f %f ", accel[0], accel[1], accel[2]);
                fprintf(out, "%f %f %f ", mag[0], mag[1], mag[2]);
                fprintf(out, "%f %d\n", temp, hdxTime);
                return;}
                break;
        case COOKED_MODE:{i=1;
                for (j=0; j<3; j++) {
            gyro[j]=((short)((packet[i]<<8)+packet[i+1]))*(gGyroRange[j]*1.5)/TWO_EXP_FIFTEEN;
                         i+=2;}
                for (j=0; j<3; j++) {
            accel[j]=((short)((packet[i]<<8)+packet[i+1]))*(gAccelRange[j]*1.5)/TWO_EXP_FIFTEEN;
                         i+=2;}
                for (j=0; j<3; j++) {
              mag[j]=((short)((packet[i]<<8)+packet[i+1]))*(gMagRange[j]*1.5)/TWO_EXP_FIFTEEN;
                         i+=2;}
                temp = ((((packet[i] << 8) + packet[i+1]) * 5.0/4096.0) - 1.375)*44.44;
                i+=2;
                hdxTime = (packet[i] <<8) + packet[i+1];
                fprintf(out, "%f %f %f ", gyro[0], gyro[1], gyro[2]);
                fprintf(out, "%f %f %f ", accel[0], accel[1], accel[2]);
                fprintf(out, "%f %f %f ", mag[0], mag[1], mag[2]);
                fprintf(out, "%f %d\n", temp, hdxTime);
                return;}
        case ANGLE_MODE:
```

```
            i=1;
            roll=((short)((packet[i]<<8)+packet[i+1]))*(180.0)/TWO_EXP_FIFTEEN;
            i+=2;
            pitch=((short)((packet[i]<<8)+packet[i+1]))*(180.0)/TWO_EXP_FIFTEEN;
            i+=2;
            yaw=((short)((packet[i]<<8)+packet[i+1]))*(180.0)/TWO_EXP_FIFTEEN;
            i+=2;
            for (j=0; j<3; j++) {
        gyro[j] = ((short)((packet[i] << 8) + packet[i+1])) *  (gGyroRange[j]*1.5)/TWO_EXP_FIFTEEN;
                i+=2;
            }
            for (j=0; j<3; j++) {
         accel[j] = ((short)((packet[i] << 8) + packet[i+1])) *(gAccelRange[j]*1.5)/TWO_EXP_FIFTEEN;
                i+=2;
            }
            for (j=0; j<3; j++) {
          mag[j] = ((short)((packet[i] << 8) + packet[i+1])) *(gMagRange[j]*1.5)/TWO_EXP_FIFTEEN;
                i+=2;
            }
            temp=((((packet[i]<<8)+packet[i+1])*5.0/4096.0)-1.375)*44.44;
            i+=2;
            hdxTime = (packet[i] <<8) + packet[i+1];
            fprintf(out, "%f %f %f ", roll, pitch, yaw);
            fprintf(out, "%f %f %f ", gyro[0], gyro[1], gyro[2]);
            fprintf(out, "%f %f %f ", accel[0], accel[1], accel[2]);
            fprintf(out, "%f %f %f ", mag[0], mag[1], mag[2]);
            fprintf(out, "%f %d\n", temp, hdxTime);
            break;
        default:
            printf("Gyro - unrecognized packet type in output_packet \n");
            break;
        } // end switch
}

/**********************************
//Function Name: restart_system()
//Descriptions:  restatr gyro sensor
```

```
***************************************/
static void restart_system(HANDLE serialPort, char mode, unsigned int *retPacketSize)
{
	unsigned char buf[MAX_BUF_SIZE];  //255 bytes, buffer the transfered and received data.
	unsigned long bytesWritten, bytesToWrite;
	unsigned int packetSize=0;   // record the packet size.
	LPDCB serialCtrl= new DCB;
	bytesWritten = 1;
	while (bytesWritten > 0) {
		buf[0] = 'P';  // AHRS command, change to polled mode, waiting 'G' to sent data
		bytesToWrite = 1;
		WriteFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
		Sleep(100);
		bytesToWrite = MAX_BUF_SIZE;
		ReadFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	}
	//set AHRS operating Mode
	buf[0] = 0;
	while  (buf[0] != (mode - 0x20)) {
		buf[0] = mode;
		bytesToWrite = 1;
		WriteFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
		Sleep(1);
		bytesToWrite = 1;
		ReadFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	}
	//stop continuous mode
	bytesWritten = 1;
	while (bytesWritten > 0) {
		buf[0] = 'P';
		bytesToWrite = 1;
		WriteFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
		Sleep(100);
		bytesToWrite = MAX_BUF_SIZE;
		ReadFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	}
	//send a 'G' to find the size of the packet
```

```
	buf[0] = 'G';
	bytesToWrite = 1;
	WriteFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	Sleep(100);
	bytesToWrite = MAX_BUF_SIZE;
	ReadFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	packetSize = bytesWritten;
	check_packet_valid(mode, buf, packetSize);
	buf[0] = 'C';
	bytesToWrite = 1;
	WriteFile(serialPort, buf, bytesToWrite, &bytesWritten, NULL);
	*retPacketSize = packetSize;
}

/*********************************
//	Function Name: output_GPS(int lines)
//	Descriptions: Write GPS data to a .txt file .
*********************************/
static void output_GPS(unsigned char *comBuf, unsigned long bytesRead, FILE* outGPS)
{
	unsigned long i;
	for (i=0; i<bytesRead; i++){
		fprintf(outGPS,"%c",comBuf[i]);
	}
}
```

# Appendix D - Modeling Study on the Effects of Pitch and Roll Measurement Errors on Elevation Measurement

```
%%***********************************************
%% CASE A: Roll Right
%%************************************************
%%Condition A
H0 = 1; %in meter
Wr = 0.5; %in meter
hh = 0.01; %in meter
aaa1 = zeros(length(0:0.05:1),length(0:0.05:30));
i = 1; j = 1;
for L=0:0.05:1 %in meter
  for a=0:0.05:30 %in degree
    b = acosd(Wr/sqrt((H0-L)^2+Wr^2)); %in degrees
    ar = b-a-asind(hh/sqrt((H0-L)^2+Wr^2)+sind(b-a));
    if(-ar>a)
      aaa1(i,j)=100;
    else
      aaa1(i,j)= ar;
    end
    j=j+1;
  end
  j=1;  i=i+1;
end

%%Condition B
H0 = 1; %in meter
Wr = 0.5; %in meter
hh = -0.01; %in meter
aaa3 = zeros(length(0:0.05:1),length(0:0.05:30));
i = 1; j = 1;
for L=0:0.05:1 %in meter
  for a=0:0.05:30 %in degree
    b = acosd(Wr/sqrt((H0-L)^2+Wr^2)); %in degrees
    ar = b-a-asind(hh/sqrt((H0-L)^2+Wr^2)+sind(b-a));
    aaa3(i,j)= ar;
```

```
      j=j+1;
   end
   j=1; i=i+1;
end

%%Comparison A and B
aaa = zeros(length(0:0.05:1),length(0:0.05:30));
[m,n]=size(aaa);
for i=1:m
   for j=1:n
      if(abs(aaa1(i,j))>abs(aaa3(i,j)))
         aaa(i,j)=abs(aaa3(i,j));
      else
         aaa(i,j)=abs(aaa1(i,j));
      end
   end
end

%%3D model plot
for ii = 1:length(0:0.05:1)
   x_matrix(ii,:) = (0:0.05:30);
end
for jj = 1:length(0:0.05:30)
   y_matrix(:,jj) = (0:0.05:1)';
end
figure;
handle_axe1=surf(x_matrix, y_matrix, aaa);
xlabel('X axis - a (degree)');
ylabel('Y axis - L (meter)');
zlabel('ar (degree)');
shading interp,
handle_light1 = camlight('headlight')
lighting phong, material dull;

%%**********************************************
%% CASE B: Roll Left
%%***********************************************
```

```
%%Condition A
H0 = 1; %in meter
Wl = 1.5; %in meter
hh = 0.01; %in meter
aaa1 = zeros(length(0:0.05:1),length(-30:0.05:0));
i = 1; j = 1;
for L=0:0.05:1 %in meter
    for a=-30:0.05:0  %in degree
        b = acosd(Wl/sqrt((H0-L)^2+Wl^2)); %in degrees
        ar = -b-a+asind(hh/sqrt((H0-L)^2+Wl^2)+sind(b+a));
        if(ar>-a)
          aaa1(i,j)=100;
        else
          aaa1(i,j)= ar;
        end
        j=j+1;
    end
    j=1; i=i+1;
end

%%Condition B
H0 = 1; %in meter
Wl = 1.5; %in meter
hh = -0.01; %in meter
aaa3 = zeros(length(0:0.05:1),length(-30:0.05:0));
i = 1; j = 1;
for L=0:0.05:1 %in meter
    for a=-30:0.05:0  %in degree
        b = acosd(Wl/sqrt((H0-L)^2+Wl^2)); %in degrees
        ar = -b-a+asind(hh/sqrt((H0-L)^2+Wl^2)+sind(b+a));
          aaa3(i,j)= ar;
        j=j+1;
    end
    j=1; i=i+1;
end

%%Comparison A and B
```

```
aaa = zeros(length(0:0.05:1),length(-30:0.05:0));
[m,n]=size(aaa);
for i=1:m
    for j=1:n
        if(abs(aaa1(i,j))>abs(aaa3(i,j)))
            aaa(i,j)=abs(aaa3(i,j));
        else
             aaa(i,j)=abs(aaa1(i,j));
        end
    end
end

%%3D model plot
for ii = 1:length(0:0.05:1)
    x_matrix(ii,:) = (-30:0.05:0);
end
for jj = 1:length(-30:0.05:0)
    y_matrix(:,jj) = (0:0.05:1)';
end
figure;
handle_axe1=surf(x_matrix, y_matrix, aaa);
xlabel('X axis - a (degree)');
ylabel('Y axis - L (meter)');
zlabel('ar (degree)');
shading interp,
handle_light1 = camlight('headlight')
lighting phong, material dull;


%%**********************************************
%% CASE C: Pitch Up
%%**********************************************
%%Condition A
H0 = 1; %in meter
Dr = 1; %in meter
hh = 0.01; %in meter
aaa1 = zeros(length(0:0.05:1),length(0:0.05:30));
```

```
i = 1; j = 1;
for L=0:0.05:1 %in meter
    for b=0:0.05:30 %in degree
        a = acos(Dr/sqrt((H0-L)^2+Dr^2)); %in radians
        br = -a*180/pi-b+asin(hh/sqrt((H0-L)^2+Dr^2)+sin(a+b*pi/180))*180/pi;
        aaa1(i,j)= br;
        j=j+1;
    end
    j=1; i=i+1;
end

%%Condition B
H0 = 1; %in meter
Dr = 1; %in meter
hh = -0.01; %in meter
aaa3 = zeros(length(0:0.05:1),length(0:0.05:30));
i = 1; j = 1;
for L=0:0.05:1 %in meter
    for b=0:0.05:30 %in degree
        a = acos(Dr/sqrt((H0-L)^2+Dr^2)); %in radians
        br = -a*180/pi-b+asin(hh/sqrt((H0-L)^2+Dr^2)+sin(a+b*pi/180))*180/pi;
        aaa3(i,j)= br;
        j=j+1;
    end
    j=1; i=i+1;
end

%%Comparison A and B
aaa = zeros(length(0:0.05:1),length(-30:0.05:0));
[m,n]=size(aaa);
for i=1:m
    for j=1:n
        if(abs(aaa1(i,j))>abs(aaa3(i,j)))
            aaa(i,j)=abs(aaa3(i,j));
        else
             aaa(i,j)=abs(aaa1(i,j));
        end
```

```
    end
end

%%3D model plot
for ii = 1:length(0:0.05:1)
    x_matrix(ii,:) = (0:0.05:30);
end
for jj = 1:length(0:0.05:30)
    y_matrix(:,jj) = (0:0.05:1)';
end
figure;
handle_axe1=surf(x_matrix, y_matrix, aaa);
xlabel('X axis - b (degree)');
ylabel('Y axis - L (meter)');
zlabel('br (degree)');
shading interp,
handle_light1 = camlight('headlight')
lighting phong, material dull;


%%*********************************************
%% CASE D: Pitch Down
%%**********************************************
%%Condition A
H0 = 1; %in meter
Df = 1; %in meter
hh = 0.01; %in meter
aaa1 = zeros(length(0:0.05:1),length(-30:0.05:0));
i = 1; j = 1;
for L=0:0.05:1 %in meter
    for b=-30:0.05:0 %in degree
        a = acos(Df/sqrt((H0-L)^2+Df^2)); %in radians
        br = -a*180/pi-b+asin(hh/sqrt((H0-L)^2+Df^2)+sin(a+b*pi/180))*180/pi;
        aaa1(i,j)= br;
        j=j+1;
    end
    j=1; i=i+1;
```

```
end

%%Condition B
H0 = 1; %in meter
Df = 1; %in meter
hh = -0.01; %in meter
aaa3 = zeros(length(0:0.05:1),length(-30:0.05:0));
i = 1; j = 1;
for L=0:0.05:1 %in meter
   for b=-30:0.05:0 %in degree
      a = acos(Df/sqrt((H0-L)^2+Df^2)); %in radians
      br = -a*180/pi-b+asin(hh/sqrt((H0-L)^2+Df^2)+sin(a+b*pi/180))*180/pi;
      aaa3(i,j)= br;
      j=j+1;
   end
   j=1; i=i+1;
end

%%Comparison A and B
aaa = zeros(length(0:0.05:1),length(-30:0.05:0));
[m,n]=size(aaa);
for i=1:m
   for j=1:n
      if(abs(aaa1(i,j))>abs(aaa3(i,j)))
         aaa(i,j)=abs(aaa3(i,j));
      else
         aaa(i,j)=abs(aaa1(i,j));
      end
   end
end

%%3D model plot
for ii = 1:length(0:0.05:1)
   x_matrix(ii,:) = (-30:0.05:0);
end
for jj = 1:length(-30:0.05:0)
   y_matrix(:,jj) = (0:0.05:1)';
```

```
end
figure;
handle_axe1=surf(x_matrix, y_matrix, aaa);
xlabel('X axis - b (degree)');
ylabel('Y axis - L (meter)');
zlabel('br (degree)');
shading interp,
handle_light1 = camlight('headlight')
lighting phong, material dull;
```

# Appendix E - Generation of 3D Raw Laser Data

```
%%**********************************************
%% MATLAB program
%% Description: 3D raw laser data generation
%% update date: 08/26/2009
%***********************************************

%% Open Laser.txt, Gyro.txt
clear; close all; clc;
fid = fopen('laser.txt','r');
C = textscan(fid, '%f %f %f %f %f %f %f %f %f', 1100000,'headerlines',1);
L = length(C{1})
fclose(fid);

fid = fopen('gyro.txt','r');
C1=textscan(fid,'%*f%f%f %f %f %f %f %f %f %f %f %f %f %*f %*f', 1100000,'headerlines',0);
L1 = length(C1{1})
fclose(fid);

phi = mean(C1{1}); % roll angle
theta = mean(C1{2}); % pitch angle
psi = mean(C1{3}); % yaw angle or heading, to magnetic north.

%% Only keep data foucing on the ground. RawData
Shield1 = 2.6353;  %default, U-shape enclosue
Shield2 = 4.3182;  %default
j = 0;
for i = 1:L
   mulripleC = floor(C{6}(i,1)/(2*pi));
   remC = C{6}(i,1)-mulripleC*2*pi;
   if ((remC>=Shield1) & (remC<= Shield2))
      j=j+1;
   end
end
Num_C1 = j  % Number of new Raw Data
RawData1 = zeros(Num_C1,10);
```

```
j = 1;
for i = 1:L
   mulripleC = floor(C{6}(i,1)/(2*pi));
   remC = C{6}(i,1)-mulripleC*2*pi;
   %if (rem(C{5}(i,1),2*pi)>=Shield1 & rem(C{5}(i,1),2*pi)<= Shield2)
   if ((remC>=Shield1) & (remC<=Shield2))
      RawData1(j,1)=C{1}(i,1);    % Distance in original file
      RawData1(j,2)=C{2}(i,1);    % Raw Range in original file
      RawData1(j,3)=C{3}(i,1);    % Amplitude in original file
      RawData1(j,4)=C{4}(i,1);    % Ambient in original file
      RawData1(j,5)=C{5}(i,1);    % Temp in original file
      RawData1(j,6)=C{6}(i,1);    % Angle1 in original file
      RawData1(j,7)=C{7}(i,1);    % Angle2 in original file
      RawData1(j,8)=C{8}(i,1);    % Index1 in original file
      RawData1(j,9)=C{9}(i,1);    % Index2 in original file
      RawData1(j,10)=i;           % Number in original file
      j=j+1;
   end
end

j = 0;
for i = 1:Num_C1
   if (RawData1(i,1)>0)
      j=j+1;
   end
end
Num_C = j  % Number of new Raw Data
RawData = zeros(Num_C,10);
j = 1;
for i = 1:Num_C1
   if (RawData1(i,1)>0)
      RawData(j,1)=RawData1(i,1);    % Distance in original file
      RawData(j,2)=RawData1(i,2);    % Raw Range in original file
      RawData(j,3)=RawData1(i,3);    % Amplitude in original file
      RawData(j,4)=RawData1(i,4);    % Ambient in original file
      RawData(j,5)=RawData1(i,5);    % Temp in original file
      RawData(j,6)=RawData1(i,6);    % Angle1 in original file
```

```
        RawData(j,7)=RawData1(i,7);     % Angle2 in original file
        RawData(j,8)=RawData1(i,8);     % Index1 in original file
        RawData(j,9)=RawData1(i,9);     % Index2 in original file
        RawData(j,10)=RawData1(i,10);   % Number in original file
        j=j+1;
    end
end

%% only calibrate the distance using data acquired on white paper
RawDatapre = RawData;  %Keep old data.
% calibration curve from Point_12inches\Points\CalibrationWhite.jpg
RawData(:,1) = RawDatapre(:,1)*0.99633 - 1.3685; %White paper

%% Polar coordinate to Cartesian coordinate conversion using AngleAdj to adjust the level.
gyro_angle = phi*pi/180;  % Gyro reading: roll angle
AngleAdj = 1.171 - gyro_angle;  % 1.171 is angle difference of z-axis between Polar and Cartesian coor.
[X,Z]=pol2cart(RawData(:,6)+AngleAdj,RawData(:,1));  %distance and angle

%% Building the Y axis based on rail encoder reading.
Angle = RawData(:,7)*180/pi;  % radian to degree

% Counting the number of rail angle change.
j= 0;
for i = 1:(Num_C-2)
    if (Angle(i)~=Angle(i+1) &Angle(i)==Angle(i-1) &Angle(i)~=Angle(i+2))
        j=j+1;
    end
end
Num = j  % Total Number of angle change.

%Create Diff_Angle[Num+1,2] to store "Number" and "Angle Value"
Diff_Angle = zeros(Num+1,2);
Diff_Angle (1,1) = 1;   % first value
Diff_Angle (1,2) = Angle(1);
k=2;
for i = 1:(Num_C-2)
    if (Angle(i)~=Angle(i+1) &Angle(i)==Angle(i-1) &Angle(i)~=Angle(i+2))
```

```
        Diff_Angle(k,1)=i+1;     % Number in RawData file (order)
        Diff_Angle(k,2)= Angle(i+1);  % Changed Angle in degree
        k=k+1;
    end
end

%Define the first and last valued number, and total valued number.
First_N = Diff_Angle(2,1);  % valued number starts
Last_N = Diff_Angle(Num+1,1) - 1; % valued number ends
Number_N = Last_N - First_N + 1; % number of valued number

%% Angle Difference array: 3 columns - "Number", "Angle Interval", "Angle in order"
%% Note: In this case the first 0 in Diff_Angle is useless.
%% So the length of Adj_Diff_Angle short than Diff_Angle, it shorts 1.
Adj_Diff_Angle = zeros(Num,3);
Adj_Diff_Angle(1,1) = Diff_Angle(2,1);
Adj_Diff_Angle(1,2) = 0;
Adj_Diff_Angle(1,3) = 0;
for i = 2 : Num
    Adj_Diff_Angle(i,1) = Diff_Angle(i+1,1);
    Adj_Diff_Angle(i,2) = abs(Diff_Angle(i+1,2) - Diff_Angle(i,2));         % Angle interal
    Adj_Diff_Angle(i,3) = abs(Diff_Angle(i+1,2) - Diff_Angle(2,2));         % incremental Angle
End

%% Reassemble X,Y,Z
R = 0.1570  % (inch) computed/empirical vaule.Define rail motor radius
% Building New_Y
Temp0 = zeros(Adj_Diff_Angle(2,1)-Adj_Diff_Angle(1,1),1);
Temp2 = Temp0;
for i = 2: Num-1
    Temp1 = zeros(Adj_Diff_Angle(i+1,1)-Adj_Diff_Angle(i,1),1);
    for j = 1 : Adj_Diff_Angle(i+1,1)-Adj_Diff_Angle(i,1)
      Temp1(j) = Adj_Diff_Angle(i,3);
    end
    Temp2 = vertcat(Temp2, Temp1);
end
New_Y = (Temp2*pi/180)*R;
```

```
% Building New_Number
Temp0 = zeros(Adj_Diff_Angle(2,1)-Adj_Diff_Angle(1,1),1);
for j = 1 : Adj_Diff_Angle(2,1)-Adj_Diff_Angle(1,1)
    Temp0(j) = Adj_Diff_Angle(1,1)+j-1;
end
Temp2 = Temp0;
for i = 2: Num-1
   Temp1 = zeros(Adj_Diff_Angle(i+1,1)-Adj_Diff_Angle(i,1),1);
   for j = 1 : Adj_Diff_Angle(i+1,1)-Adj_Diff_Angle(i,1)
     Temp1(j) = Adj_Diff_Angle(i,1)+j-1;
   end
   Temp2 = vertcat(Temp2, Temp1);
end
New_Number = Temp2;
% Building New_X
New_X = zeros(Number_N,1);
for i = 1 : Number_N
   New_X(i) = X(First_N+i-1);
end
% Building New_Z
New_Z = zeros(Number_N,1);
for i = 1 : Number_N
   New_Z(i) = Z(First_N+i-1);
end
% Building New_RawRange
New_RawRange = zeros(Number_N,1);
for i = 1 : Number_N
   New_RawRange(i) = RawData((First_N+i-1),2);
end
% Building New_Amp
New_Amp = zeros(Number_N,1);
for i = 1 : Number_N
   New_Amp(i) = RawData((First_N+i-1),3);
end
% Building New_Ambient
New_Ambi = zeros(Number_N,1);
for i = 1 : Number_N
```

```
    New_Ambi(i) = RawData((First_N+i-1),4);
end
% Building New_Temp
New_Temp = zeros(Number_N,1);
for i = 1 : Number_N
    New_Temp(i) = RawData((First_N+i-1),5);
end

% Pitch roations only (by 12/21/09), transoformation between two X-Y-Z Cartesian coordinates
gyro_pitch = -theta;
New_Y1 = New_Y;
New_Z1 = New_Z;
New_Y = (cos(gyro_pitch*pi/180))*New_Y1 + (sin(gyro_pitch*pi/180))*New_Z1;
New_Z = -(sin(gyro_pitch*pi/180))*New_Y1 + (cos(gyro_pitch*pi/180))*New_Z1;

%% Concatenate X,Y,Z matrices. [X,Y,Z,Raw Range,Amplitude,Temp,Number]
NewXYZ =
horzcat(New_X,New_Y,New_Z,New_RawRange,New_Amp,New_Ambi,New_Temp,New_Number);
save('raw_level.mat'); % save 3D raw laser data

%% Narrow X range.
data = NewXYZ;
X_upper = 20; % need to change
X_lower = -25; % need to change
data(data(:,1)<X_lower,:)=[];
data(data(:,1)>X_upper,:)=[];
%% Narrow Z range.
data = NewXYZ;
Z_upper = -33.11; % need to change
Z_lower = -45.23; % need to change
data(data(:,3)<Z_lower,:)=[];
data(data(:,3)>Z_upper,:)=[];
%% Narrow Y range.
data = NewXYZ;
Y_upper = ; % need to change
Y_lower = 1.44; % need to change
data(data(:,2)>Y_upper,:)=[];
```

```
data(data(:,2)<Y_lower,:)=[];

figure;
plot3(data(:,1),data(:,2),data(:,3),'marker','.','linestyle','none')
xlabel('X axis - roll (inches)');
ylabel('Y axis - pitch (inches)');
zlabel('Z axis - yaw (inches)');
title('Distance plot ');
grid on;

save('raw_level_zxy_pitch.mat');
```

# Appendix F - Interpolation Algorithm

```
%%****************************************
%%  MATLAB program
%%  Description:  Triangle-based linear/distance-weighted Interpolation
%%****************************************

load raw_level_zxy_.mat data;  %% MATLAB data file constaining laser raw data in a array - data
[m_data,n_data] = size(data);
nLdata(:,1:3)=data(:,1:3)*2.54; % inch to cm

fprintf('The minimum X is %.4f cm\n',min(nLdata(:,1)));
fprintf('The maximum X is %.4f cm\n',max(nLdata(:,1)));
fprintf('The minimum Y is %.4f cm\n',min(nLdata(:,2)));
fprintf('The maximum Y is %.4f cm\n',max(nLdata(:,2)));

xres = 0.5; % x resolution,cm,equal to the resolution of the reference frame.
yres = 0.5; % y resolution,cm,equal to the resolution of the reference frame.
xmin = min(nLdata(:,1)) + 3; % cm, starting point of x for the intoplation
xmax = max(nLdata(:,1)) - 3; % cm, ending point of x for the intoplation
ymin = min(nLdata(:,2)) + 1; % cm, starting point of y for the intoplation
ymax = max(nLdata(:,2)) - 1; % cm, ending point of y for the intoplation
xrange = 2; %search range for x
yrange = 2; %search range for y

yb = ymin;
zb_i = 1;
zb_k = 0;
[m_nLdata,n_nLdata] = size(nLdata);
while (yb <= ymax)
   zb_j = 1;
   xb = xmax;
   while (xb >= xmin)
          % narrow the X and Y within xrange and yrange of xb and yb, save them in nLdataxy.
      nLdatax = nLdata(find(nLdata(:,1)<=(xb+xrange)),1:3);
      nLdatax = nLdatax(find(nLdatax(:,1)>=(xb-xrange)),1:3);
```

```
    nLdatay = nLdatax(find(nLdatax(:,2)<=(yb+yrange)),1:3);
    nLdataxy = nLdatay(find(nLdatay(:,2)>=(yb-yrange)),1:3);
    % distance calculation
    dis = sqrt((nLdataxy(:,1)-xb).^2 + (nLdataxy(:,2)-yb).^2);
    nLdataxy_dis = horzcat(nLdataxy,dis);
    % ascend the data based on the ascending order of distance.
    [dis_ascend,dis_ind]= sort(nLdataxy_dis(:,4));
    nLdataxy_dis_ascent = nLdataxy_dis(dis_ind,1:4);
    if (nLdataxy_dis_ascent(1,1)==xb & nLdataxy_dis_ascent(2,1)==xb &
nLdataxy_dis_ascent(3,1)==xb )
      nLdata_ascentxx = nLdataxy_dis_ascent(3:end,:);
      nLdata_ascentx = nLdata_ascentxx(find(nLdata_ascentxx(:,1)~=xb),1:4);
      nLdataxy_dis_ascent(3,:)=nLdata_ascentx(1,:);
    end
    if (nLdataxy_dis_ascent(1,2)==yb & nLdataxy_dis_ascent(2,2)==yb &
nLdataxy_dis_ascent(3,2)==yb )
      nLdata_ascentyy = nLdataxy_dis_ascent(3:end,:);
      nLdata_ascenty = nLdata_ascentyy(find(nLdata_ascentyy(:,2)~=yb),1:4);
      nLdataxy_dis_ascent(3,:)=nLdata_ascenty(1,:);
    end
    if((nLdataxy_dis_ascent(1,1)==xb)& nLdataxy_dis_ascent(1,2)==yb)
      zb_matrix(zb_i,zb_j) =nLdataxy_dis_ascent(1,3) ;
      zb(zb_k+zb_j)= nLdataxy_dis_ascent(1,3);
    else
      %Pick the first three as the three vertices locations
      trix1 = nLdataxy_dis_ascent(1,1); triy1 = nLdataxy_dis_ascent(1,2);
      triz1 = nLdataxy_dis_ascent(1,3);
      trix2 = nLdataxy_dis_ascent(2,1); triy2 = nLdataxy_dis_ascent(2,2);
      triz2 = nLdataxy_dis_ascent(2,3);
      trix3 = nLdataxy_dis_ascent(3,1); triy3 = nLdataxy_dis_ascent(3,2);
      triz3 = nLdataxy_dis_ascent(3,3);
      %calcuation1
      tri2b=sqrt((trix2-xb)^2+(triy2-yb)^2);
      tri3b=sqrt((trix3-xb)^2+(triy3-yb)^2);
      tri23=sqrt((trix2-trix3)^2+(triy2-triy3)^2);
      d1 = sqrt(tri3b^2-((tri23^2+tri3b^2-tri2b^2)/(2*tri23))^2);
      d21 = sqrt(tri2b^2-d1^2);
```

```
            d31 = sqrt(tri3b^2-d1^2);
            %calcuation2
            tri1b=sqrt((trix1-xb)^2+(triy1-yb)^2);
            tri31=sqrt((trix3-trix1)^2+(triy3-triy1)^2);
            d2 = sqrt(tri1b^2-((tri31^2+tri1b^2-tri3b^2)/(2*tri31))^2);
            d32 = sqrt(tri3b^2-d2^2);
            d12 = sqrt(tri1b^2-d2^2);
            %calcuation3
            tri12=sqrt((trix1-trix2)^2+(triy1-triy2)^2);
            d3 = sqrt(tri2b^2-((tri12^2+tri2b^2-tri1b^2)/(2*tri12))^2);
            d23 = sqrt(tri2b^2-d3^2);
            d13 = sqrt(tri1b^2-d3^2);
            %Polynomial Weighting Functions W1, W2, and W3
            W1 = d1^2*(d2^2*d23^2+d3^2*d32^2);
            W2 = d2^2*(d1^2*d13^2+d3^2*d31^2);
            W3 = d3^2*(d1^2*d12^2+d2^2*d21^2);

            zb_matrix(zb_i,zb_j) = (W1*triz1 + W2*triz2 + W3*triz3)/(W1+W2+W3);
                % the matrix of the interpolated z-value: m*n=(number of y)*(number of x)
            zb(zb_k+zb_j)= (W1*triz1 + W2*triz2 + W3*triz3)/(W1+W2+W3);
        end
        zb_j=zb_j+1;
        xb = xb - xres;
    end
    zb_k=zb_k+zb_j-1;
    zb_i=zb_i+1;
    yb = yb +yres;
end

% generate X and Y matrix for 3D plots
[m_y,n_x] = size(zb_matrix);
xb_matrix=zeros(m_y,length(xmax:-xres:xmin));
for i=1:m_y
    xb_matrix(i,:)=xmax:-xres:xmin;
end
for i=1:n_x
    yb_matrix(:,i)=(ymin:yres:ymax)';
```

```
end

figure
handle_axe1 = surf(xb_matrix,yb_matrix,zb_matrix);
xlabel('X axis (cm)','fontsize',12);
ylabel('Y axis (cm)','fontsize',12);
zlabel('Z axis (cm)','fontsize',12);
title({'Surface Plot of the reference laser readings';'Distance-weighted Neareset Neighbor Interpolation'});
shading interp,colorbar
handle_light1 = camlight('headlight')
lighting phong, material dull;
```

# Appendix G - LabVIEW Programs

LabVIEW programs for the reference system to control top and left/right motors running in two modes and to log laser data.

## G.1 Independent Mode

**Figure G.1 Front Panel**

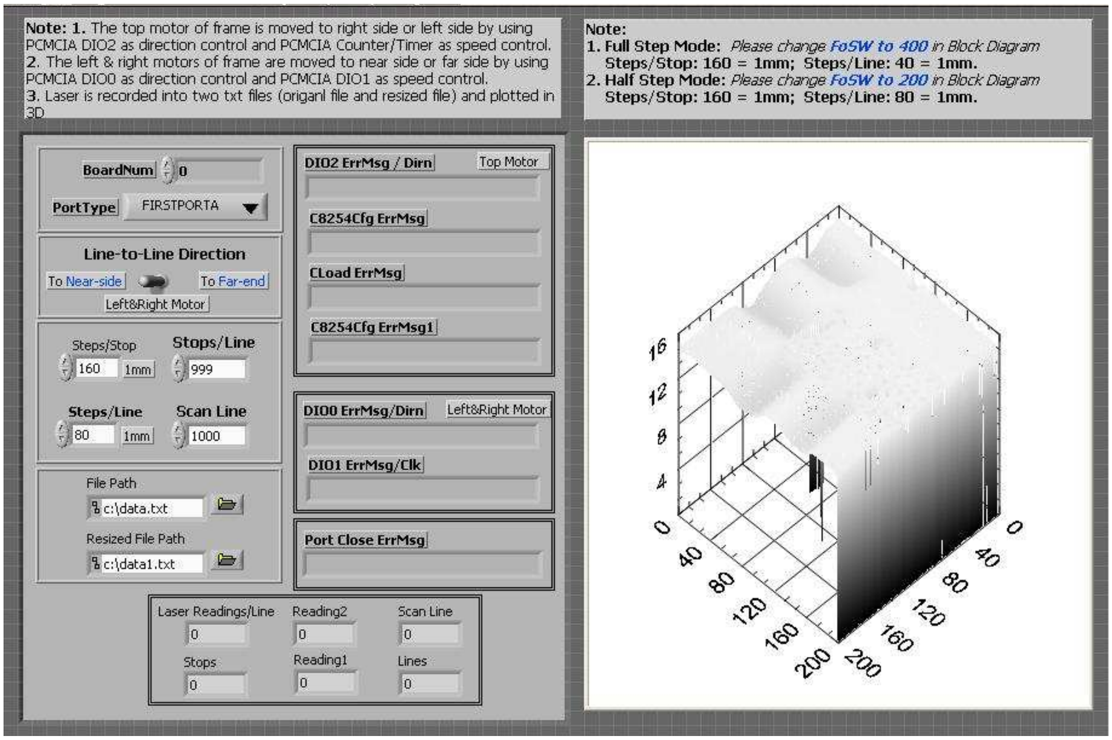


**Figure G.2 Block Diagram**

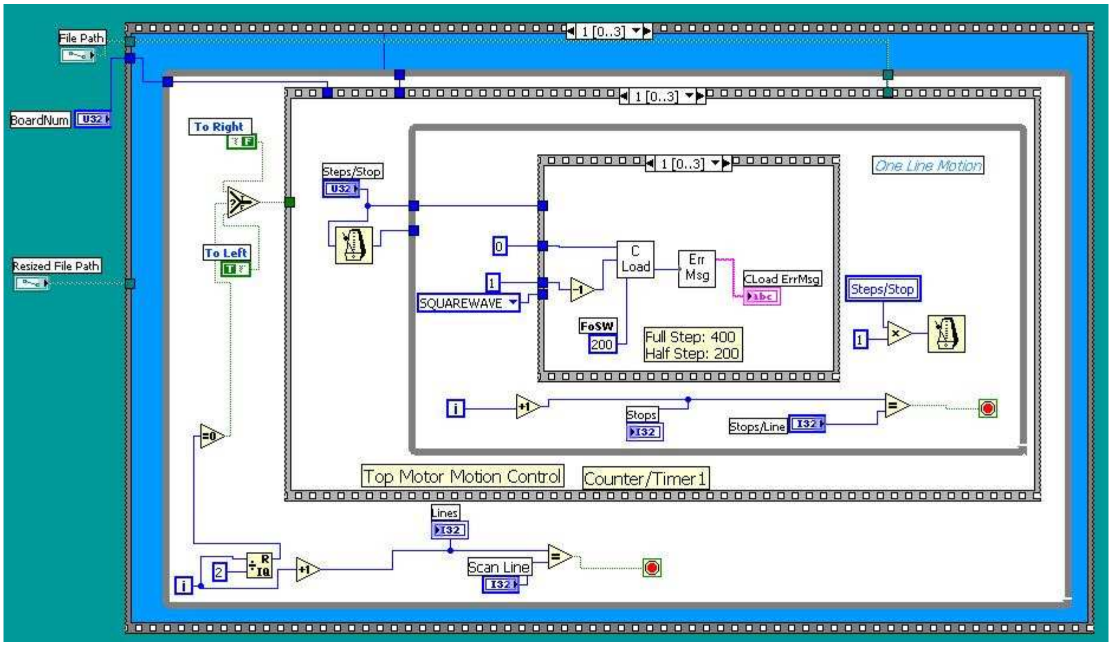
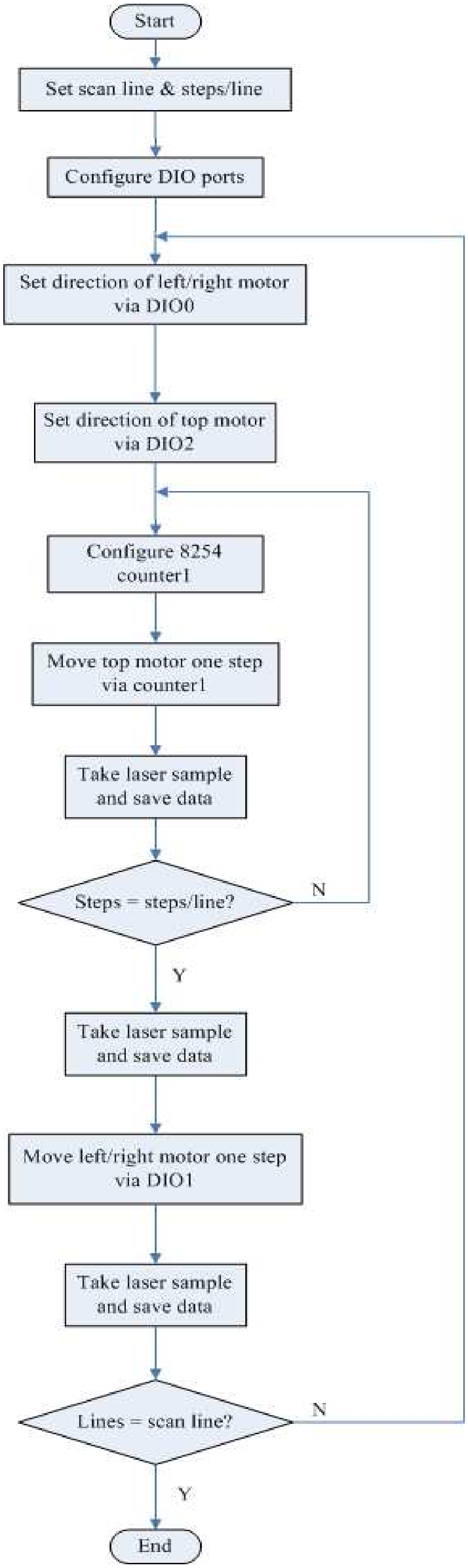
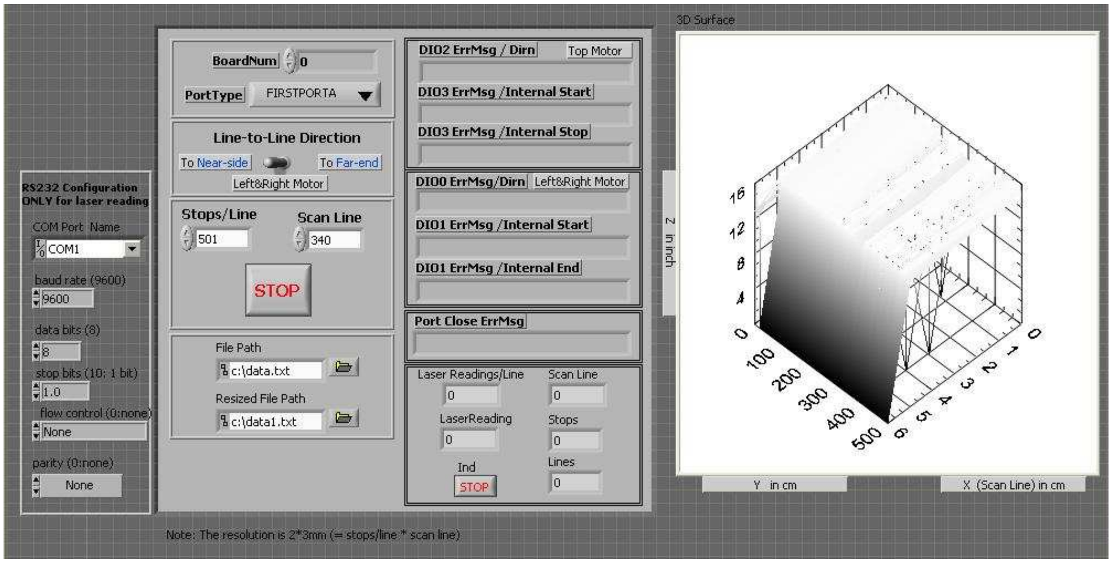


**Figure G.3 Flow Chart**



## G.2 Continuous Mode

### Figure G.4 Front Panel



### Figure G.5 Block Diagram

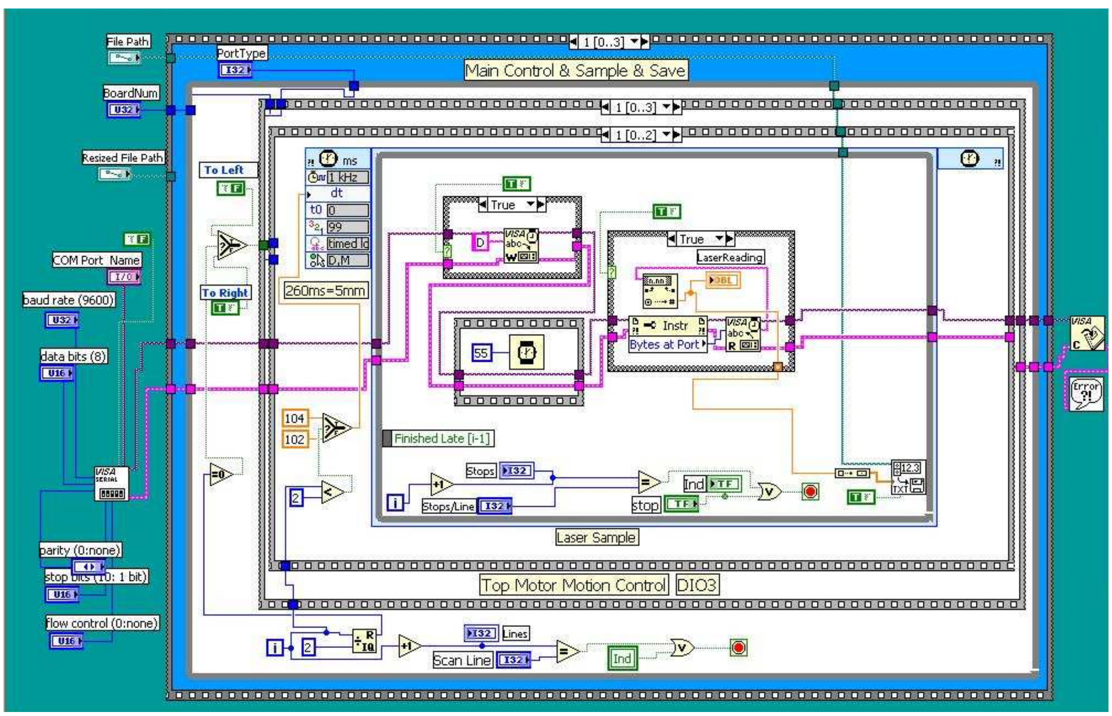
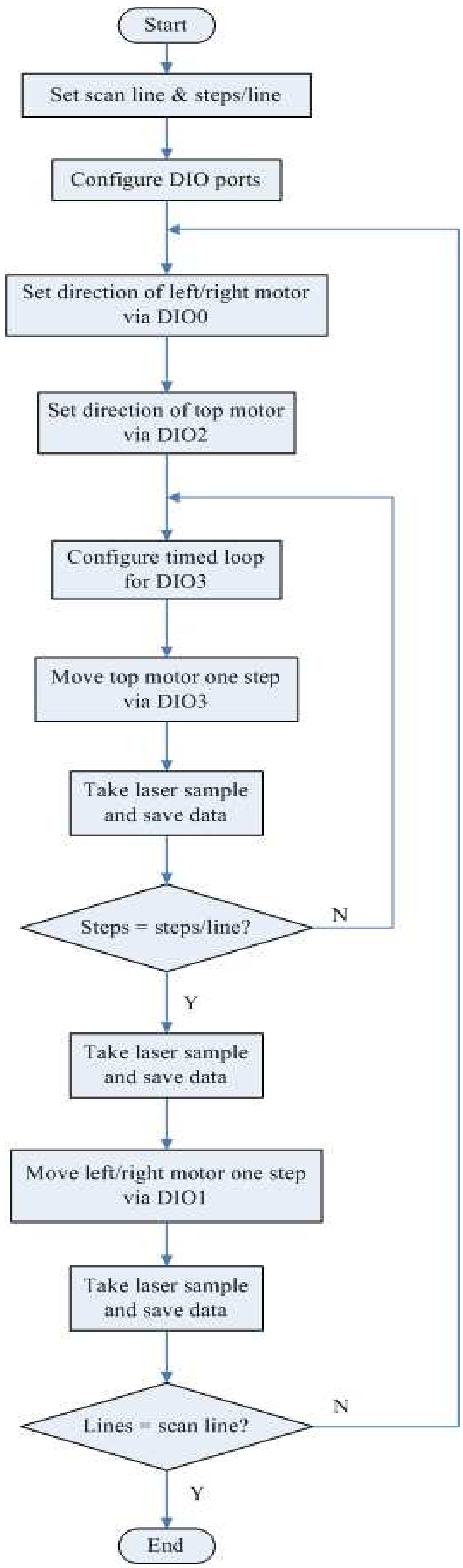


**Figure G.6 Flow Chart**



# Appendix H - Matching by Correlation

```
%%****************************************
%%  MATLAB program
%%  Description:  Matching by correlation
%%****************************************

[m_fxy, n_fxy] = size(Fxy);       % Fxy - Data array contains the reference data set
[j_wxy, k_wxy] = size(Wxy);      % Wxy - Data array contains the sub data set
if (rem(j_wxy,2) == 0 )
   j_Wori = floor(j_wxy/2);
else
   j_Wori = floor(j_wxy/2)+1;
end
if (rem(k_wxy,2) == 0 )
   k_Wori = floor(k_wxy/2);
else
   k_Wori = floor(k_wxy/2)+1;
end

W_average = mean(mean(Wxy));
Rxy = zeros(m_fxy,n_fxy);
for i=1:m_fxy
   if ((i >= j_Wori) & ((m_fxy - i) >= j_Wori)) %high/low parts of w(x,y) is inside of f(x,y)
      for j=1:n_fxy
         if ((j >= k_Wori) & ((n_fxy - j) >= k_Wori)) %left/right parts of w(x,y) inside of f(x,y)
            F_average_st = mean(mean(Fxy((i-j_Wori+1):(i+(j_wxy-j_Wori)),(j-k_Wori+1):(j+(k_wxy-
k_Wori)))));
            for ii=1:j_wxy
               for jj=1:k_wxy
                  if ((ii==1) & (jj==1))
                     st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
                     st2=Wxy(ii,jj)-W_average;
                     st1_square = st1*st1;
                     st2_square = st2*st2;
                     Rxy_numerator = st1*st2;
```

```
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
            st2=Wxy(ii,jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
            Rxy_denominator2 = Rxy_denominator2 + st2_square;
         end
      end
   end
   Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
elseif (j < k_Wori)             % left parts of w(x,y) is outside of f(x,y)
   F_average_st = mean(mean(Fxy((i-j_Wori+1):(i+(j_wxy-j_Wori)),1:(j+(k_wxy-k_Wori)))));
   for ii=1:j_wxy
      for jj=1:(j+k_wxy-k_Wori)
         if ((ii==1) & (jj==1)) %first data point
            st1=Fxy(i-j_Wori+ii,jj)-F_average_st;
            st2=Wxy(ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = st1*st2;
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(i-j_Wori+ii,jj)-F_average_st;
            st2=Wxy(ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
            Rxy_denominator2 = Rxy_denominator2 + st2_square;
         end
      end
   end
```

```
            Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        else          % (n_fxy - j)<k_Wori. right parts of w(x,y) is outside of f(x,y)
            F_average_st = mean(mean(Fxy((i-j_Wori+1):(i+(j_wxy-j_Wori)),((j-k_Wori)+1):end)));
            for ii=1:j_wxy
                for jj=1:(k_Wori+ n_fxy-j)
                    if ((ii==1) & (jj==1)) %first data point
                        st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
                        st2=Wxy(ii,jj)-W_average;
                        st1_square = st1*st1;
                        st2_square = st2*st2;
                        Rxy_numerator = st1*st2;
                        Rxy_denominator1 = st1_square;
                        Rxy_denominator2 = st2_square;
                    else
                        st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
                        st2=Wxy(ii,jj)-W_average;
                        st1_square = st1*st1;
                        st2_square = st2*st2;
                        Rxy_numerator = Rxy_numerator + st1*st2;
                        Rxy_denominator1 = Rxy_denominator1 + st1_square;
                        Rxy_denominator2 = Rxy_denominator2 + st2_square;
                    end
                end
            end
            Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        end
    end
elseif(i < j_Wori)              %high parts of w(x,y) is outside of f(x,y)
    for j=1:n_fxy
        if ((j >= k_Wori) & ((n_fxy - j) >= k_Wori)) %left/right parts of w(x,y) inside of f(x,y)
            F_average_st = mean(mean(Fxy(1:(i+(j_wxy-j_Wori)),(j-k_Wori+1):(j+(k_wxy-k_Wori)))));
            for ii=1:(i+j_wxy-j_Wori)
                for jj=1:k_wxy
                    if ((ii==1) & (jj==1))
                        st1=Fxy(ii,j-k_Wori+jj)-F_average_st;
                        st2=Wxy(j_Wori-i+ii,jj)-W_average;
                        st1_square = st1*st1;
```

```
            st2_square = st2*st2;
            Rxy_numerator = st1*st2;
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(ii,j-k_Wori+jj)-F_average_st;
            st2=Wxy(j_Wori-i+ii,jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
            Rxy_denominator2 = Rxy_denominator2 + st2_square;
         end
      end
   end
   Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
elseif (j < k_Wori)                  % left parts of w(x,y) is outside of f(x,y)
   F_average_st = mean(mean(Fxy(1:(i+(j_wxy-j_Wori)),1:(j+(k_wxy-k_Wori)))));
   for ii=1:(j_wxy-j_Wori+i)
      for jj=1:(k_wxy-k_Wori+j)
         if ((ii==1) & (jj==1)) %first data point
            st1=Fxy(ii,jj)-F_average_st;
            st2=Wxy(j_Wori-i+ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = st1*st2;
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(ii,jj)-F_average_st;
            st2=Wxy(j_Wori-i+ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
            Rxy_denominator2 = Rxy_denominator2 + st2_square;
         end
```

```
            end
          end
          Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        else              % (n_fxy - j)<k_Wori. right parts of w(x,y) is outside of f(x,y)
          F_average_st = mean(mean(Fxy(1:(i+(j_wxy-j_Wori)),((j-k_Wori)+1):end)));
          for ii=1:(j_wxy-j_Wori+i)
            for jj=1:(k_Wori+ n_fxy-j)
              if ((ii==1) & (jj==1)) %first data point
                st1=Fxy(ii,j-k_Wori+jj)-F_average_st;
                st2=Wxy(j_Wori-i+ii,jj)-W_average;
                st1_square = st1*st1;
                st2_square = st2*st2;
                Rxy_numerator = st1*st2;
                Rxy_denominator1 = st1_square;
                Rxy_denominator2 = st2_square;
              else
                st1=Fxy(ii,j-k_Wori+jj)-F_average_st;
                st2=Wxy(j_Wori-i+ii,jj)-W_average;
                st1_square = st1*st1;
                st2_square = st2*st2;
                Rxy_numerator = Rxy_numerator + st1*st2;
                Rxy_denominator1 = Rxy_denominator1 + st1_square;
                Rxy_denominator2 = Rxy_denominator2 + st2_square;
              end
            end
          end
          Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        end
      end
    else                   % (m_fxy - i) < j_Wori. Low parts of w(x,y) is outside of f(x,y)
      for j=1:n_fxy
        if ((j >= k_Wori) & ((n_fxy - j) >= k_Wori)) %left/right parts of w(x,y) inside of f(x,y)
          F_average_st = mean(mean(Fxy((i-j_Wori+1):end,(j-k_Wori+1):(j+(k_wxy-k_Wori)))));
          for ii=1:(j_Wori+m_fxy-i)
            for jj=1:k_wxy
              if ((ii==1) & (jj==1))
                st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
```

```
            st2=Wxy(ii,jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = st1*st2;
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
            st2=Wxy(ii,jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
            Rxy_denominator2 = Rxy_denominator2 + st2_square;
         end
      end
   end
   Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
elseif (j < k_Wori)           % left parts of w(x,y) is outside of f(x,y)
   F_average_st = mean(mean(Fxy((i-j_Wori+1):end,1:(j+(k_wxy-k_Wori)))));
   for ii=1:(j_Wori+m_fxy-i)
      for jj=1:(k_wxy-k_Wori+j)
         if ((ii==1) & (jj==1)) %first data point
            st1=Fxy(i-j_Wori+ii,jj)-F_average_st;
            st2=Wxy(ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = st1*st2;
            Rxy_denominator1 = st1_square;
            Rxy_denominator2 = st2_square;
         else
            st1=Fxy(i-j_Wori+ii,jj)-F_average_st;
            st2=Wxy(ii,k_Wori-j+jj)-W_average;
            st1_square = st1*st1;
            st2_square = st2*st2;
            Rxy_numerator = Rxy_numerator + st1*st2;
            Rxy_denominator1 = Rxy_denominator1 + st1_square;
```

```
                Rxy_denominator2 = Rxy_denominator2 + st2_square;
              end
            end
          end
          Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        else          % (n_fxy - j)< k_Wori. right parts of w(x,y) is outside of f(x,y)
          F_average_st = mean(mean(Fxy((i-j_Wori+1):end,(j-k_Wori+1):end)));
          for ii=1:(j_Wori+ m_fxy-i)
            for jj=1:(k_Wori+ n_fxy-j)
              if ((ii==1) & (jj==1)) %first data point
                st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
                st2=Wxy(ii,jj)-W_average;
                st1_square = st1*st1;
                st2_square = st2*st2;
                Rxy_numerator = st1*st2;
                Rxy_denominator1 = st1_square;
                Rxy_denominator2 = st2_square;
              else
                st1=Fxy(i-j_Wori+ii,j-k_Wori+jj)-F_average_st;
                st2=Wxy(ii,jj)-W_average;
                st1_square = st1*st1;
                st2_square = st2*st2;
                Rxy_numerator = Rxy_numerator + st1*st2;
                Rxy_denominator1 = Rxy_denominator1 + st1_square;
                Rxy_denominator2 = Rxy_denominator2 + st2_square;
              end
            end
          end
          Rxy(i,j) = Rxy_numerator/(sqrt(Rxy_denominator1*Rxy_denominator2));
        end
      end
    end
  end
R_max = max(max(Rxy));  % correlation coefficient
```

# Appendix I - SAS Code

```
%%*********************************************
%% SAS code
%% Note: Input different datalines, before running this code
%%********************************************

data;
input CS MR REP Y;
datalines;


proc mixed method= type3;
class CS MR;
model y= CS MR CS*MR REP /solution;
lsmeans cs mr cs*mr/diff cl slice = cs;
run;
proc sort;
by cs mr;
run;
proc plot;
plot y*rep;
by cs mr;
run;
proc reg;
model y = rep;
by cs mr;
run;
```

# Appendix J - LOOCV Based Algorithm

```
%%****************************************
%%  MATLAB program
%%  Description:  Leave- one-out  cross-volidation based data filter algorithm
%%     Note: May not good or necessary for all data sets
%%****************************************

NewXYZ1 = data;                 % data - Data array contains the laser raw data set
deltaY = min(all_deltaY(:,1));       % all_deltaY(:,1) - interval of each scanning line.
[m_NewXYZ1,n_NewXYZ1] = size(NewXYZ1);
data1 = ones(1,n_NewXYZ1);        %store the final data set.
Degree = 16;                 % the maximum polynoial degree based on previous anaysis of CV
ii = 1;
numLines = 0; % count number scanned lines.
for i=1:(m_NewXYZ1-1)
   if ( (NewXYZ1(i+1,2)-NewXYZ1(i,2))>=deltaY)
      j = i;
      numLines = numLines + 1;
      Ddata = NewXYZ1(ii:j,:); %Each line. part of NewXYZ
      sizeD = size(Ddata,1); % numbers of data in each line.
      nRegression = 1;  % for while loop
      pd(1,3) = 0;      % for while loop.
      numDdata = sizeD; % for while loop
      while (pd(nRegression,3)~=numDdata)
         sizeD = size(Ddata,1); % numbers of data in each line.
         Ddata_random = Ddata;
         fold = sizeD; %LOOCV
         numDdata = size(Ddata,1);
         for degree = 0: Degree % given possible polynomial degree
            sizeG2 = 0; % end number of each group
            for fold_Ddata =1:fold %k-fold method and leave-one-out method
                  sizeG1 = sizeG2;
                  sizeG2 = round(sizeD*fold_Ddata/fold);
                  sizeG = sizeG2-sizeG1; % each group size
                  xtest = Ddata_random(sizeG1+1:sizeG2,1);
                  ytest = Ddata_random(sizeG1+1:sizeG2,3);
```

```
                xtrain = Ddata_random(:,1);
                ytrain = Ddata_random(:,3);
                xtrain(sizeG1+1:sizeG2) = [];
                ytrain(sizeG1+1:sizeG2) = [];
                p_g = polyfit(xtrain,ytrain,degree);%polynomial coefficients
                r_g(degree+1,fold_Ddata) = sum((ytest - polyval(p_g,xtest)).^2);
            end
            pe(degree+1) = mean(r_g(degree+1,:)); %degree vs. 'MSE'
        end
        for pe_index = 1:Degree+1 % search for the minimum of 'MSE'
            if pe(pe_index)== min(pe)
                degreeCV = pe_index-1;
            end
        end
        pd(nRegression+1,1) = degreeCV; % regression time vs optimal polynomial degree
        x_std = (Ddata(:,1)-mean(Ddata(:,1)))./std(Ddata(:,1)); %Normalize X in Ddata.
        p_d = polyfit(x_std,Ddata(:,3),degreeCV); %polynomial coefficients
        pd(nRegression+1,2) = mse(Ddata(:,3) - polyval(p_d,x_std)); % RT vs mse
        [p,S]= polyfit(x_std,Ddata(:,3),degreeCV);
        [Dis,delta] = polyval(p,x_std,S);
        BB = double(abs(Ddata(:,3)-Dis)< 2*delta);  % condition
        Ddata(:,3) = Ddata(:,3).*BB;
        Ddata(Ddata(:,3)==0,:)=[];
        pd(nRegression+1,3) = size(Ddata,1);
        Lines(numLines,nRegression) = degreeCV;
        nRegression = nRegression +1;
    end
    data1 = vertcat(data1,Ddata);
    ii = j+1;
  end
end
% last one line
numLines = numLines + 1;
Ddata = NewXYZ1(ii:m_NewXYZ1,:); % parts of NewXYZ
sizeD = size(Ddata,1); % numbers of data in each line.
nRegression = 1;  % for while loop
pd(1,3) = 0;      % for while loop. need to delete in furture.
```

```
numDdata = sizeD; % for while loop
while (pd(nRegression,3)~= numDdata)
   sizeD = size(Ddata,1); % numbers of data in each line.
   Ddata_random = Ddata;
   fold = sizeD;
   numDdata = size(Ddata,1);
   for degree = 0: Degree % given possible polynomial degree
      sizeG2 = 0; % end number of each group
      for fold_Ddata =1:fold
            sizeG1 = sizeG2;
            sizeG2 = round(sizeD*fold_Ddata/fold);
            sizeG = sizeG2-sizeG1; % numbers in each group
            xtest = Ddata_random(sizeG1+1:sizeG2,1);
            ytest = Ddata_random(sizeG1+1:sizeG2,3);
            xtrain = Ddata_random(:,1);
            ytrain = Ddata_random(:,3);
            xtrain(sizeG1+1:sizeG2) = [];
            %xtrain = (xtrain-mean(xtrain))./std(xtrain);
            ytrain(sizeG1+1:sizeG2) = [];
            p_g = polyfit(xtrain,ytrain,degree);
            r_g(degree+1,fold_Ddata) = sum((ytest - polyval(p_g,xtest)).^2);
      end
      pe(degree+1) = mean(r_g(degree+1,:));
   end
   for pe_index = 1:Degree+1
      if pe(pe_index)== min(pe)
         degreeCV = pe_index-1;
      end
   end
   pd(nRegression+1,1) = degreeCV; % regression time vs optimal polynomial degree
   x_std = (Ddata(:,1)-mean(Ddata(:,1)))./std(Ddata(:,1)); %Normalize X in Ddata.
   p_d = polyfit(x_std,Ddata(:,3),degreeCV); %polynomial coefficients
   pd(nRegression+1,2) = mse(Ddata(:,3) - polyval(p_d,x_std)); % RT vs mse
   [p,S]= polyfit(x_std,Ddata(:,3),degreeCV);
   [Dis,delta] = polyval(p,x_std,S);
   BB = double(abs(Ddata(:,3)-Dis)< 2*delta);  % condition
   Ddata(:,3) = Ddata(:,3).*BB;
```

```
    Ddata(Ddata(:,3)==0,:)=[];
    pd(nRegression+1,3) = size(Ddata,1);
    Lines(numLines,nRegression) = degreeCV;
    nRegression = nRegression +1; % for while loop
end
data1 = vertcat(data1,Ddata);
data1(data1(1,1)==1,:)=[];

figure;
plot3(data1(:,1),data1(:,2),data1(:,3),'marker','.','linestyle','none')
xlabel('X axis - roll (inches)');
ylabel('Y axis - pitch (inches)');
zlabel('Z axis - yaw (inches)');
title('Distance plot ');
grid on;
```

# Appendix K - Conversion of the Local Coordinate System to the UTM Geographic Coordinate System

```
%%**************************************************
%% MATLAB program
%%  Description:: Local coordinate system to UTM geographic coordinate system.
%%  Nomenclature:
%%  {b} – X-Y-Z coordinate system
%%  {be} – E-N-U coordinate system
%%  {e} – UTM (WGS84) geographic coordinate system
%%**************************************************

%% open gyro.txt
fid = fopen('gyro.txt','r');
% import degree,rate sensor, accelerometer, magnetometer
C1=textscan(fid, '%*f%f%f%f%f %f %f %f %f %f %f %f %f %f %*f %*f', 1100000,'headerlines',0);
L1 = length(C1{1})
fclose(fid);

phi = mean(C1{1});    % roll angle
theta = mean(C1{2});   % pitch angle
psi = mean(C1{3});    % yaw angle or heading, to magnetic north.
dis_origin = 51/2.54;   % horizontal distance between the starting position of laser and Gyro.
GM_Angle = 4.5; %the angle between grid north and magnetic north,in 1997
psi = -psi + GM_Angle;
dis_origin_v = 24/2.54;  % vertical distance between the starting position of laser and Gyro.

data2 = data1;      %3D raw laser data in {b}
data2(:,1) = cos(psi*pi/180)*data1(:,1)+sin(psi*pi/180)*(data1(:,2)-dis_origin); %east in {be}
data2(:,2) = -sin(psi*pi/180)*data1(:,1)+cos(psi*pi/180)*(data1(:,2)-dis_origin); %north in {be}
data2(:,3) = data1(:,3)+ dis_origin_v;    %up in {be}

% unit conversion: inch to meter
data2(:,1) = data2(:,1)*0.0254;
data2(:,2) = data2(:,2)*0.0254;
data2(:,3) = data2(:,3)*0.0254;
```

```
%% open gps.txt
fid = fopen('gps.txt','r');
C2 = textscan(fid, '%s%f%f%c%f%c%n%n%f%f%c%f%c%c%s', -1,'headerlines',1, 'delimiter',',');
L2 = length(C2{1})
fclose(fid);

% Universal Transverse Mercator Projection Parameters % all angles are in radians,
% DD MM.MMMM to DD.ff
lat_mean = mean(C2{3});
long_mean = mean(C2{5});
lat = (39+(lat_mean-3900)/60)*pi/180;            % latitude
long = (-(96+(long_mean-9600)/60))*pi/180;    %longitude
long0 = (-99)*pi/180;        % longitude of the origin (the central meridian) of the projection
                              % need to change. refernce to UTM grid
delta_long = long-long0 ;     % difference of longitude from the central meridian
U_k0 = 0.9996;                %central scale factor, an arbitrary reduction applied to all
                              % geodetic lengthes to reduce the maximum scale distortion of the prejection
FN = 0;              % false northing
FE = 500000;     % false easting

% ellipsoid parameters (NAD83/WGS84)
E_a = 6378137;                        % meters, semi-major axis of the ellipsoid
E_b = 6356752.3142;                  % meters, semi-major axis of the ellipsoid
E_f = 1/298.25723563;                % flattening or ellipticity =(a-b)/a
E_e2 = (E_a^2-E_b^2)/E_a^2;    % the square of first eccentricity.%0.00669
E_ee2 = (E_a^2-E_b^2)/E_b^2;   % the square of second eccentricity.%0.007
E_n = (E_a-E_b)/(E_a+E_b);
E_p = E_a*(1-E_e2)/(1-E_e2*sin(lat)^2)^(3/2);     % raduis of curvature in the meridian
E_v = E_a/(1-E_e2*sin(lat)^2)^(1/2);                  %raduis of curature in the prime vertical;
                                    % also defined as the normal to the ellipsoid termination at the minor axis
E_A = E_a*(1-E_n+5/4*(E_n^2-E_n^3)+81/64*(E_n^4-E_n^5));
E_B = 3/2*E_a*(E_n-E_n^2+7/8*(E_n^3-E_n^4)+55/64*(E_n^5-E_n^6));
E_C = 15/16*E_a*(E_n^2-E_n^3+3/4*(E_n^4-E_n^5));
E_D = 35/48*E_a*(E_n^3-E_n^4+11/16*(E_n^5-E_n^6));
E_E = 315/512*E_a*(E_n^4-E_n^5);    % 0.03mm
E_S = E_A*lat-E_B*sin(2*lat)+E_C*sin(4*lat)-E_D*sin(6*lat)+E_E*sin(8*lat);
```

```
%Terms used to calculate general equations.
T1 = E_S*U_k0;
T2 = E_v*sin(lat)*cos(lat)*U_k0/2;
T3 = E_v*sin(lat)*cos(lat)^3*U_k0/24*(5 - tan(lat)^2 + 9*E_ee2*cos(lat)^2 +...
    4*E_ee2^2*cos(lat)^4);
T4 = E_v*sin(lat)*cos(lat)^5*U_k0/720*(61 - 58*tan(lat)^2 + tan(lat)^4 +...
    270*E_ee2*cos(lat)^2 - 330*tan(lat)^2*E_ee2*cos(lat)^2 +...
    445*E_ee2^2*cos(lat)^4 + 324*E_ee2^3*cos(lat)^6 -...
    680*tan(lat)^2*E_ee2^2*cos(lat)^4 + 88*E_ee2^4*cos(lat)^8 -...
    600*tan(lat)^2*E_ee2^3*cos(lat)*6 - 192*tan(lat)^2*E_ee2^4*cos(lat)^8);
T5 = E_v*sin(lat)*cos(lat)^7*U_k0/40320*(1385 - 3111*tan(lat)^2 +...
    543*tan(lat)^4 - tan(lat)^6);
T6 = E_v*cos(lat)*U_k0;
T7 = E_v*cos(lat)^3*U_k0/6*(1 - tan(lat)^2 + E_ee2*cos(lat)^2);
T8 = E_v*cos(lat)^5*U_k0/120*(5 - 18*tan(lat)^2 + tan(lat)^4 +...
    14*E_ee2*cos(lat)^2 - 58*tan(lat)^2*E_ee2*cos(lat)^2 +...
    13*E_ee2^2*cos(lat)^4 + 4*E_ee2*3*cos(lat)^6 -...
    64*tan(lat)^2*E_ee2^2*cos(lat)^4 - 24*tan(lat)^2*E_ee2^3*cos(lat)^6);
T9 = E_v*cos(lat)^7*U_k0/5040*(61 - 479*tan(lat)^2 + 179*tan(lat)^4 - tan(lat)^6);

%% latitude and longitude of  the origin of {be} in {e}
North = FN+(T1+delta_long^2*T2+delta_long^4*T3+delta_long^6*T4+delta_long^8*T5);
East = FE+(delta_long*T6+delta_long^3*T7+delta_long^5*T8+delta_long^7*T9);

% {be} to {e}: UTM coordinate system - North and East and Down.
data2(:,1) = data2(:,1) + East;
data2(:,2) = data2(:,2) + North;
figure;
plot3(data2(:,1),data2(:,2),data2(:,3),'marker','.','linestyle','none')
xlabel('East (meter)');
ylabel('North (meter)');
zlabel('Distance (meter)');
title('Earth fixed coordinate frame - UTM-WGS84');
grid on;

%add 3/19/2010
data3 = data2;
```

```
MSL = mean(C2{10});
dis_v = 11/100; % vertical distance between the GPS and Gyro.
data3(:,3)= data2(:,3)+MSL+dis_v;

figure;
plot3(data3(:,1),data3(:,2),data3(:,3),'marker','.','linestyle','none')
xlabel('East (meter)');
ylabel('North (meter)');
zlabel('Altitude,mean sea level (meter)');
title('Earth fixed coordinate frame - UTM-WGS84');
grid on;

save('UTM.mat')  % data save.
```

# Appendix L - Official Formulas for the Conversion of Geographic Coordinates to UTM Grid Coordiantes

## L.1 General Formulas and Terms

The National Geospatial-intelligence Agency issued the official formulas for the conversion of geographic coordinates to UTM grid coordinates in the technical document of the Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPC) in 1989. These general formulas for the computation of north and east in the UTM geographic system by knowing the latitude and longitude are given in equations H.1 and H.2. For the following definitions of terms, all lengths are in meters and all geographic coordinates are in radians unless specified otherwise. The sign notation is negative for the Southern and Western hemispheres.

$$N = FN + ( T1 + \Delta\lambda 2T2 + \Delta\lambda 4T3 + \Delta\lambda 6T4 + \Delta\lambda 8T5) \qquad (H.1)$$

$$E = FE + ( \Delta\lambda T6 + \Delta\lambda 3T7 + \Delta\lambda 5T8 + \Delta\lambda 7T9) \qquad (H.2)$$

where N = grid northing in {e},

E = grid easting in {e},

FN = False Northing (0 for the Northern hemisphere; 10000000 for the Southern Hemisphere),

FE = 500000, False Easting,

T1 = $Sk_0$,

$$T2 = \frac{v \sin\phi \cos\phi \, k_0}{2},$$

$$T3 = \frac{v \sin\phi \cos^3\phi \, k_0}{2}(5 - \tan^2\phi + 9e'^2\cos^2\phi + 4e'^4\cos^4\phi),$$

$$T4 = \frac{v \sin\phi \cos^5\phi k_0}{720}(61 - 58\tan^2\phi + \tan^4\phi + 270e'^2\cos^2\phi -$$
$$330\tan^2\phi e'^2\cos^2\phi + 445e'^4\cos^4\phi + 324e'^6\cos^6\phi -$$
$$680\tan^2\phi e'^4\cos^4\phi + 88e'^8\cos^8\phi - 600\tan^2 e'^6\cos^6\phi$$
$$-192\tan^2\phi e'^8\cos^8\phi)$$
,

$$T5 = \frac{v \sin\phi \cos^7\phi\, k_0}{40320}(1385 - 3111\tan^2\phi + 543\tan^4\phi - \tan^6\phi),$$

T6 = vcosΦ $k_0$,

$$T7 = \frac{v \cos^3\phi\, k_0}{6}(1 - \tan^2\phi + e'^2\cos^2\phi),$$

$$T8 = \frac{v \cos^5\phi k_0}{120}(5 - 18\tan^2\phi + \tan^4\phi + 14e'^2\cos^2\phi -$$
$$58\tan^2\phi e'^2\cos^2\phi + 13e'^4\cos^4\phi + 4e'^6\cos^6\phi -$$
$$64\tan^2\phi e'^4\cos^4\phi - 24\tan^2 e'^6\cos^6\phi)$$
, and

$$T9 = \frac{v \cos^7\phi\, k_0}{5040}(61 - 479\tan^2\phi + 179\tan^4\phi - \tan^6\phi).$$

## L.2 Universal Transverse Mercator Projection Parameters

Φ = latitude

λ = longitude

Φ' = latitude of the foot of the perpendicular from the point to the central meridian

λo = longitude of the origin (the central meridan) of the projection

Δ λ = λ - λo, difference of longitude from the central meridian (for genearl formula use, value is sing dependent; for use in tables, value always considered position)

Ko = 0.9996, central scale factor, an arbitrary reduction applied to all geodetic lengths to reduce the maximum scale distortion of the projection.

k = scale factor at the working point on the projection

## L.3 Ellipsoid Parameters:

$a = 6378137$, semi-major axis of the ellipsoid

$b = 6356752.3142$, semi-major axis of the ellipsoid

$f = (a-b)/a = 1/298.25723563$, flattening or ellipticity

$e^2 = (a^2 - b^2)/a^2 = f(2 - f) = (\text{first eccentricity})^2$

$e'^2 = (a^2 - b^2)/b^2 = f(2 - f)/(1 - f)^2 = e^2/(1 - e^2) = (\text{second eccentricity})^2$

$n = (a - b)/(a + b) = f/(2 - f)$

$\rho = a(1 - e^2)/(1 - e^2\sin^2\Phi)^{3/2}$, radius of curvature in the meridan

$v = a(1 - e^2)/(1 - e^2\sin^2\Phi)^{3/2} = \rho(1 + e'^2 \cos^2\Phi)$,

radius of curvature in the prime vertical;also defined as the normal to the ellipsoid terminating at the minor axis

$S = A'\,\Phi - B'\sin2\Phi + C'\sin4\Phi - D'\sin6\Phi + E'\sin8\Phi$ ,

meridional arc, the true meridional distance on the ellipsoid from the equator,

where $A' = a\,[\,1 - n + 5(n^2 - n^3)/4 + 81(n^4 - n^5)/64 + \ldots]$,

$B' = 3a/2\,[\,n - n^2 + 7(n^3 - n^4)/8 + 55n^5/64 + \ldots]$,

$C' = 15a/16\,[\,n^2 - n^3 + 3(n^3 - n^4)/4 + \ldots]$,

$D' = 35a/48\,[\,n^3 - n^4 + 11n^5/16 + \ldots]$, and

$E' = 315a/512\,[\,n^4 - n^5 + \ldots]$.