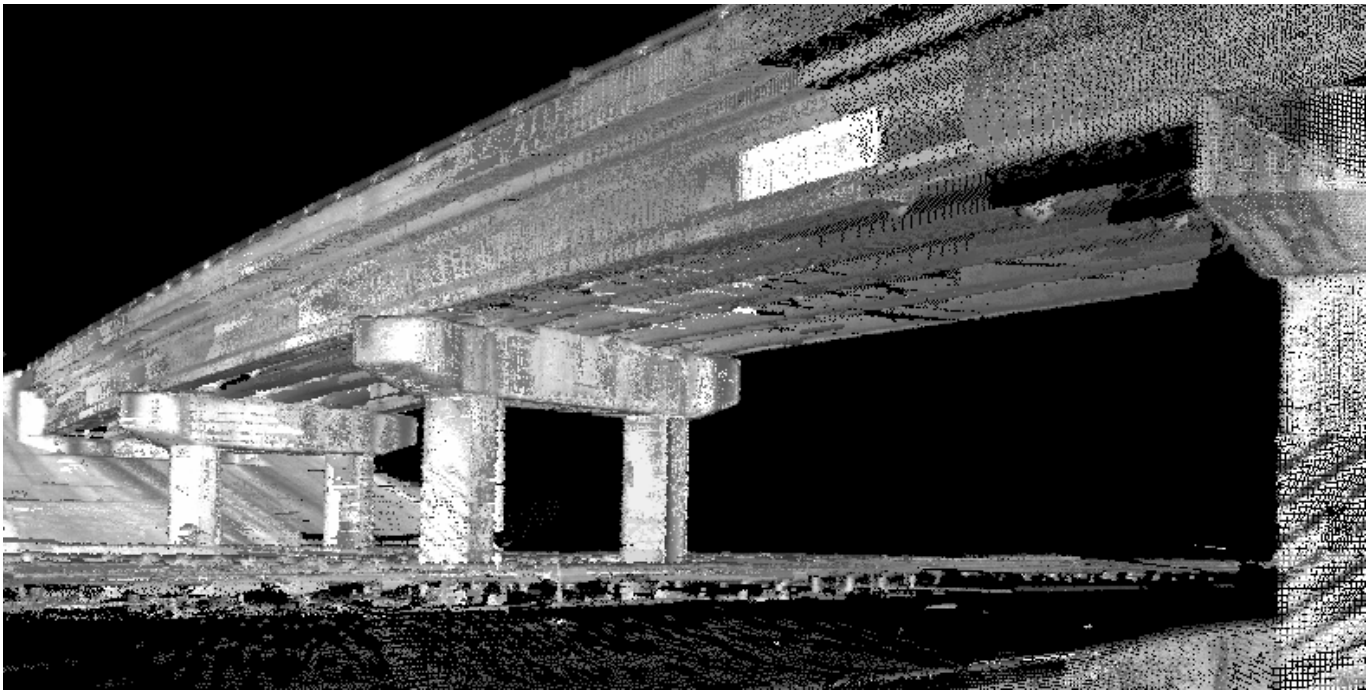
Final Report

Acquisition of Laser-Based



Data Measurement Systems

BC-354 #87

UF Project #00026874

David Bloomquist, PhD, PE
Department of Civil and Coastal Engineering
University of Florida
May, 2005

Disclaimer

The opinions, findings, and conclusions expressed in this publication are those of the author and not necessarily those of the State of Florida Department of Transportation.”

SI (MODERN METRIC) CONVERSION FACTORS (from FHWA)

APPROXIMATE CONVERSIONS TO SI UNITS

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
LENGTH				
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
AREA				
in^2	squareinches	645.2	square millimeters	mm^2
ft^2	squarefeet	0.093	square meters	m^2
yd^2	square yard	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
VOLUME				
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
NOTE: volumes greater than 1000 L shall be shown in m^3				

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
MASS				
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
TEMPERATURE (exact degrees)				
^oF	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	^oC

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
ILLUMINATION				
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m^2	cd/m^2

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
FORCE and PRESSURE or STRESS				
lbf	poundforce	4.45	newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	kPa

APPROXIMATE CONVERSIONS TO SI UNITS

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
LENGTH				
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
AREA				
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
VOLUME				
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.314	cubic feet	ft^3
m^3	cubic meters	1.307	cubic yards	yd^3

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
MASS				
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2000 lb)	T

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
TEMPERATURE (exact degrees)				
^oC	Celsius	1.8C+32	Fahrenheit	^oF

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
ILLUMINATION				
lx	lux	0.0929	foot-candles	fc
cd/m^2	candela/m^2	0.2919	foot-Lamberts	fl

SYMBOL	WHEN YOU KNOW	MULTIPLY BY	TO FIND	SYMBOL
FORCE and PRESSURE or STRESS				
N	newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

*SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380.
(Revised March 2003)

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle ACQUISITION OF LASER-BASED DATA MEASUMENTS SYSTEMS		5. Report Date May 2005	
		6. Performing Organization Code	
7. Author(s) David Bloomquist		8. Performing Organization Report No.	
9. Performing Organization Name and Address Department of Civil and Coastal Engineering 365 Weil Hall University of Florida Gainesville, Florida 32611		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. BC-354 #87	
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee, FL 32399		13. Type of Report and Period Covered Final Report 8/2003 – 5/2005	
		14. Sponsoring Agency Code	
15. Supplementary Notes			

16. Abstract

The objective of this project was to acquire two laser measurement devices. An OPTECH ILRIS 3D scanner and MINOLTA VIVID 910 imager. A secondary objective was to conduct preliminary tests using the devices on a variety of FDOT related projects.

Two bridges were scanned during load testing and the results indicated that the accuracy approaches traditional LVDT data without the concomitant cabling, traffic issues, etc. In addition, a haul truck was scanned and the volume of fill material computed. The results were within 1.5% of the calculated quantity.

The VIVID 910 was used to measure strain patterns on a concrete structural beam. Small spherical targets were affixed to the surface and their relative movements monitored. Software was written to plot the results, but did not perform as desired.

Finally, a novel target system was developed for bridges in order to monitor long-term movements of specific locations.

17. Key Word Laser, LIDAR, strain, infrastructure		18. Distribution Statement No restrictions.	
19. Security Classif. (of this report) Unclassified.	20. Security Classif. (of this page) Unclassified.	21. No. of Pages 45 + PowerPoint (60 pp.)	22. Price

Executive Summary

While 9/11 is slowly fading into history, the terrible events showed that damage assessment using traditional techniques was rapidly becoming obsolete. UF was fortunate to have participated in a research project that included using a scanning 3D imaging system to quantify some of the damage. A final report to the National Science Foundation was issued that recommended using this technology for pre/post event condition assessments.

Having collaborated on numerous research projects with our state department of transportation, FDOT was approached with the idea of purchasing such a unit and testing it for various applications, specifically critical infrastructural condition assessment. The device that was acquired is an OPTECH ILRIS 3D imager. At the time of purchase, it was the only manufacturer. However, as others have found out about its attributes, various firms are now selling similar products.

As a complementary piece of equipment, FDOT also approved the purchase of a MINOLTA Vivid 910 Laboratory laser scanner. The idea is to use this very accurate scanner to measure induced strains in concrete elements. While it was not designed for this purpose, the goal was to see if we could develop analytical software that will enable the real-time strain/damage data acquisition and analysis.

Objectives

The major objectives of this research include:

A. Acquiring the two instruments. (Note: the purchase of these two units represented over 80% of the total requested budget. The remaining funds were used to pay a student to evaluate the equipment, conduct preliminary tests and to begin to write the necessary software to analyze the data.)

B. Conduct preliminary tests using both systems. For the ILRIS, this involved two types of measurements. The first involved bridge deflections during loading, while the second was to look at the development of targets for rapid identification/monitoring. The MINOLTA also was to be evaluated as a NDT strain measurement system using micro spheres as targets.

Findings and Conclusions

Two tests were conducted to evaluate the capabilities of the ILRIS 3D system. Deflections were measured on two bridges during load application. It was found that the instrument was capable of measuring within 5 mm compared to an LVDT. In addition, a substantial amount of effort was expended to develop targets that could be placed on the bridge and used as reference locations. This would facilitate the determination of differential movements among the targets. A novel concept using hemispherical shaped domes (constructed from wood and Styrofoam plastic) was tested. This was one of the most important tasks and while successful, more effort on this topic is still ongoing. This Report also provides a PowerPoint presentation presented to an ASCE Branch meeting in September 2004. It was offered prior to additional work being performed; hence other uses of the devices are not included. It provides an overview of the technology, and several examples of how land-based or terrestrial scanners can be used. It bears mentioning that at a recent NSF Panel meeting (April 2005), the PI reviewed 4 proposals requesting this type of equipment. The proposers were going to use them to study rock slides, excavations and to verify as-built construction documents. Of even more interest is the fact that one of the proposals stated that they would need to design some type of target system, which was one of their major objectives of the proposal. Hence, the foresight of FDOT in acquiring this equipment is laudable.

A target system that was developed consists of half spheres placed along the surface of the structure. Then if point cloud data is acquired from target returns, any 3 or more points allows one to compute the location of the sphere's center. This procedure can then be performed to find the same center point regardless of the angle of the laser (Since it is assumed that the member being scanned displaces during a test, the relative angle between a particular target and the laser changes). For multiple targets, the relative movement between them can be determined by observing the movements of their center points.

In addition to developing the targets, research was conducted on developing an algorithm to rapidly locate the targets from the data set. Matlab mathematical software was used as the framework. In addition to custom software, the software packages that came with the two laser systems, Polyworks and Minolta Polygon Editor were also evaluated on their potential to perform the same function as the custom algorithm. The downside to using the commercial

applications is that the processing of the targets has to be done manually on a target by target basis, a task that is exceptionally labor and time intensive.

Benefits

The benefits of utilizing using these devices are enormous. Pre-scans of critical infrastructures (overpasses, bridges, intelligent highway signs, etc.) with the ILRIS can then be compared with post damage data. This will allow for rapid assessment of a structures' safety. Since the laser is setup remotely from the structure, there is no need to alter traffic during a test. Measuring the deflection of a bridge over a waterway is also now possible.

The laboratory MINOLTA laser will provide strain data over an area, as opposed to a single point on a loaded structure. In addition, it is capable of detecting microcracks – something traditional point location strain gages are not suited for.

Finally, FDOT will find numerous other applications for this equipment. Texture measurements of flexible highway pavements can signal a roadway's potential for hydroplaning. Rapidly computing the amount of soil in a haul truck can reduce the amount of money paid to contractors.

TABLE OF CONTENTS **Page**

Introduction

While 9/11 recedes into history, the terrible events showed that damage assessment using traditional techniques was rapidly becoming obsolete. UF was fortunate to have participated in a research project that included using a scanning 3D imaging system to quantify some of the damage. A final report to the National Science Foundation was issued that recommended using this technology for pre/post event condition assessments.

Having worked on numerous research projects with our state department of transportation, FDOT was approached with the idea of purchasing such a unit and trying it for various applications. The device that was acquired is an OPTECH ILRIS 3D imager. At the time of purchase, it was the only one available. However, as others have found out about its attributes, various manufacturers are now selling similar products.

As a complementary piece of equipment, FDOT also approved the purchase of a MINOLTA Vivid 910 Laboratory scanner. The idea is to use this very accurate scanner to measure induced strains in concrete elements. While it was not designed for this purpose, the goal is to see if we can develop analytical software that will enable the real-time strain data acquisition and analysis.

Hence the major objectives of this research were to:

A. Acquire the two instruments. (Note: the purchase of these two units represented over 75% of the total requested budget. The remaining funds were used to pay a student to evaluate the equipment, conduct preliminary tests and to begin to write the necessary software to analyze the data.) This purchasing task was completed.

B. Conduct preliminary tests using both systems. For the ILRIS, this involved two types of measurements. The first involved bridge deflections during loading. Two tests were conducted to evaluate the technique. In addition, a substantial amount of effort was expended to develop targets that could be placed on the bridge and used as reference locations. This would facilitate the determination of differential movements among the targets. A novel concept using hemispherical shaped domes (constructed from wood and Styrofoam plastic) was tested. This was one of the most important tasks and while the contract has since expired, effort on this topic is still ongoing. This Report also provides a PowerPoint presentation presented to an ASCE

Branch meeting in September 2004. It was offered prior to additional work being performed, hence other uses of the devices are not included. It provides an overview of the technology, and several examples of how land-based or terrestrial scanners can be used. It bears mentioning that at a recent NSF Panel meeting (April 2005), the PI reviewed 4 proposals requesting this type of equipment. The proposers were going to use them to study rock slides, excavations and to verify as-built construction documents. Of even more interest is the fact that one of the proposals stated that they would need to design some type of target system, which was one of their major objectives of the proposal. Hence, the foresight of FDOT in acquiring this equipment is laudable.

The remainder of this report presents some of the work performed to date. Again, the primary goal of this project was to acquire and test two new devices. While we have not yet created a viable software package that the FDOT requires, we plan to do this during the Summer 2005. A PhD student has been identified who will be responsible for this task.

Optech ILRIS 3D Laser System

As mentioned, two laser systems were purchased; one for scanning large objects at great distances and one for scanning smaller objects in greater detail. The Optech ILRIS 3D Laser System was acquired to perform long range scanning of structural elements. This system employs a self contained unit which can scan objects without the need for a laptop computer. The user controls and targets the laser through the use of a Palm PDA. The data from the scans are then stored on a flash memory card which is then removed and inserted in a computer for processing. The laser system comes with the InnovMetric Software package which can be used to analyze and model the data obtained with the laser scans.

ILRIS 3D Specifications:

Performance Range Data sample rate	 350 m (4% target) 800 m (20% target) 2,000 points/second	**Output**	Metafile consisting of XYZ, active laser photograph (intensity), digital photo data, operator setup parameters and notes (from Palm PDA).
Accuracy * Target registration accuracy Modeling accuracy Depth resolution	 4 mm 3 mm 3 mm	**Power (Input)** Power input Battery Power consumption	 24 VDC battery or AC converter 24 V rechargeable 75 W (typical)
Laser Spot Size Spot size	D = 0.17R+12, where D = diameter of spot (mm) R = range to target (m) 29 mm @ 100 m	**Size** Scanner (L × W × H) Scanner weight	 312 × 312 × 205 mm 12 kg
Minimum Spot Spacing Spot spacing	S = 0.026R, where S = spacing (mm) R = range to target (m) <2.6 mm @ 100 m	**Environmental** Operating temperature	0° C to +40° C (For extended range consult Optech)
		Storage temperature Environment	-20° C to +50° C NEMA 4X rated, water and dust-proof, IP65
Operator Interface Digital camera Viewfinder Control interface	 Colour 640 × 480 pixels 17 cm VGA Palm (or compatible) PDA	**Eye Safety**	Class I laser product IEC 60825-1, US FDA 21 CFR 1040 Eye-safe in all modes of operation
Standard Software	Data parser and reduction software	**Standard Accessories**	Palmtop computer ATA flash memory card Batteries and charger Carry case
Recommended Software	Palm OS control software Point cloud alignment, inspection and visualization software; Polyworks® software from InnovMetric	**Optional Accessories**	Extra flash cards Extra batteries AC adapter Solar power unit Charger accessories

Minolta VIVID 910

A Minolta VIVID 910 was purchased to perform the more detailed scans. This laser system consists of a self contained unit that, like the ILRIS, does not require the use of a computer. The scanner is controlled through the use of a control panel on the unit itself. Similar to the ILRIS laser, this unit stores scan data on a flash memory card which can be removed and later read and analyzed on a computer. The data from this scanner can be read by a variety of software packages, including InnovMetric Polyworks as well as an included Minolta Polygon imaging program.

VIVID 910 Specifications:

Resolution and Range of Digitized Volumes (X, Y, Z where x is the horizontal dimension of the focal plane, y is the vertical axis, z is distance from the sensor, units are millimeters), Field of View varies based on distance between VIVID and scanned object.

	Near field (@ 0.6 m)	Far field	Max resolution (depth)
Tele Lens:	111 x 84 x 40 mm	460 x 350 x 130 mm (@ 2.5 m)	0.039 mm (0.0016")
Mid Lens:	196 x 153 x 70 mm	830 x 622 x 220 mm (@ 2.5 m)	0.068 mm (0.0026")
Wide Lens:	355 x 266 x 92 mm	1200 x 903 x 400 mm (@ 2 m)	0.090 mm (0.0035")

Type	Non-contact 3-D LASER digitizer VIVID 910
Measuring Method	Triangulation light block method
Auto-Focus method	Image surface Auto-Focus (contrast method), active Auto-Focus
Light-Receiving Lens (Interchangeable)	TELE: Focal distance f=25mm MEDIUM: Focal distance f=14mm WIDE: Focal distance f=8mm
Scan Range	0.6 to 2.5m (2m for WIDE)
Optimal 3D measurement Range	0.6 to 1.2m
Laser class	Class 2 (IEC 60825-1), "Eye safe" Class 1 (FDA)
Laser Scan Method	Galvanometer-driven rotating mirror
X Direction Input Range (Varies with the distance)	111 to 463mm (TELE), 198 to 823 (MIDDLE), 359 to 1196mm (WIDE)
Y Direction Input Range (Varies with the distance)	83 to 347mm (TELE), 148 to 618 (MIDDLE), 269 to 897mm (WIDE)
Z Direction Input Range (Varies with the distance)	40 to 500mm (TELE), 70 to 800 (MIDDLE), 110 to 750mm (WIDE)
Precision (Z,Typ.)	+- 0.008mm (Condition: FINE mode, Konica Minolta's standard)
Accuracy	+- 0.008mm (Condition: FINE mode, Konica

	Minolta's standard)
TELE X: +- 0.22mm, Y: +- 0.16mm, Z: +- 0.10mm to the Z reference plane (Conditions: TELE/FINE mode, Konica Minolta's standard)	0.3 sec (FAST mode), 2.5 sec (FINE mode), 0.5 sec (COLOR)
Input Time	0.3 sec (FAST mode), 2.5 sec (FINE mode), 0.5 sec (COLOR)
Transfer Time to Host Computer	Approx. 1 sec (FAST mode), 1.5 sec (FINE mode)
Ambient Lighting Condition	Office Environment, 500 lux or less
Imaging Element	3-D data: 1/3-inch frame transfer CCD (340,000 pixels) Color data: 3-D data is shared (color separation by rotary filter).
Number of Output Pixels	3-D data: 307,000 (for FINE mode), 76,800 (for FAST mode) Color data: 640 x 480 x 24 bits color depth
Output Format	3-D data: Konica Minolta format & (STL, DXF, OBJ, AXCII points, VRML) (Converted to 3-D data by the Polygon Editing Software/standard accessory Color data: RGB 24-bit raster scan data)
Recording Medium	Compact Flash memory card (128MB)
Date File Size	Total 3-D and color data capacity: 1.6MB per data (for FAST mode), 3.6MB per data (for FINE mode)
Viewfinder	5.7-inch LCD (320 x 240 pixels)
Output Interface	SCSI II (DMA synchronous transfer)
Power	Commercial AC power 100 to 240V (50 to 60Hz), rated current 0.6A (when 100Vac is input)
Dimensions	213 (W) x 413 (H) x 271 (D) mm (8-3/8 (W) x 16-1/4 (H) x 10-11/16 (D) inches
Weight	Approx. 11kg (25 lbs)
Operating Environment	temperature: 10 to 40 degrees C (50 to 104 degrees F); relative humidity 65% or less with no condensation, Pollution degree: 2, Installation category: II
Storage Temperature & Humidity Range	-10 to 50 degrees C (14 to 122 degrees F); relative humidity 85% or less (at 35 degrees C/95 degrees F) with no condensation

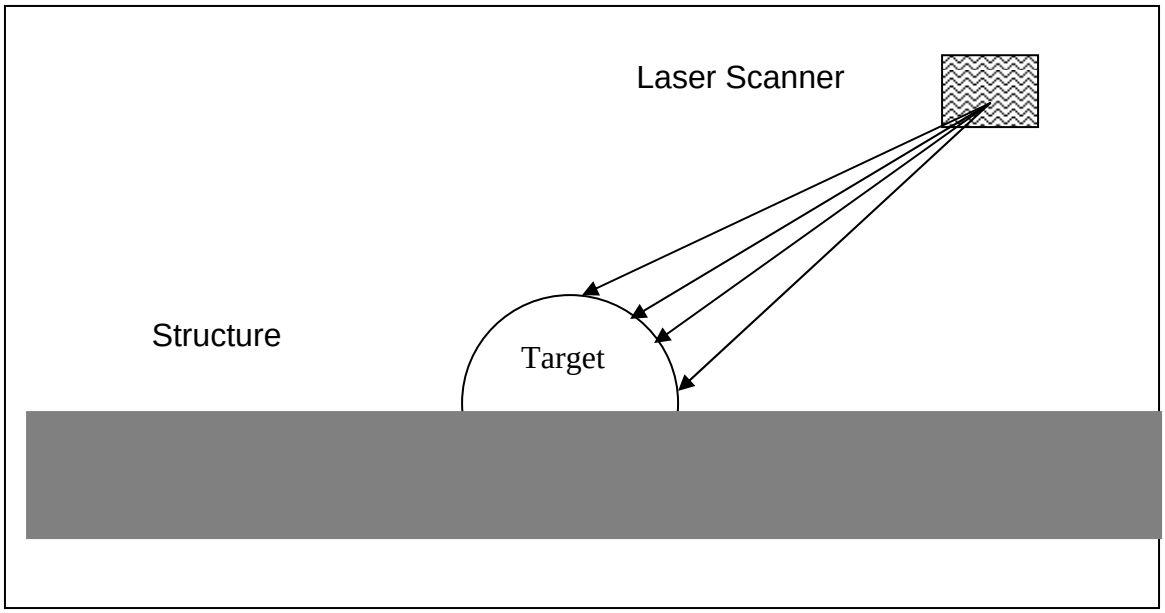
Laser Scan Data

The data obtained from the laser scans consist of digital information assembled into 3D point clouds. These data contain of a series of 3D (x,y,z) points which represent the timed reflections from emitted pulses that the laser detects. These reflections are returned in a coordinate system which is based on the relative position of the laser. The ILRIS-3D in addition to simply returning reflection locations also returns an intensity value of the return. This intensity value can be used to detect differences in material properties based on the surface of the object. The size of the ILRIS data files varies depending on the spot spacing selected. The finer the scan quality, the larger the size of the data files. The Minolta scans at two preset resolutions and produces two different file sizes.

Structure Targets

In order to properly analyze the laser scans it is necessary to determine an accurate and reliable means of locating single points along the scan area, and then find these exact same points through multiple scans in order to compute relative movements. A target system that was developed consists of half spheres placed along the surface of the structure. Then if point cloud data is acquired from target returns, any 3 or more points allows one to compute the location of the sphere's center. This procedure can then be performed to find the same center point regardless of the angle of the laser (Since it is assumed that the member being scanned displaces during a test, the relative angle between a particular target and the laser changes). For multiple targets, the relative movement between them can be determined by observing the movements of their center points.

For the lab testing, the above targets are too large; hence we attempted to use precision ball bearings. The methodology for locating the center of it and ultimately the point of contact was similar to the dome targets.

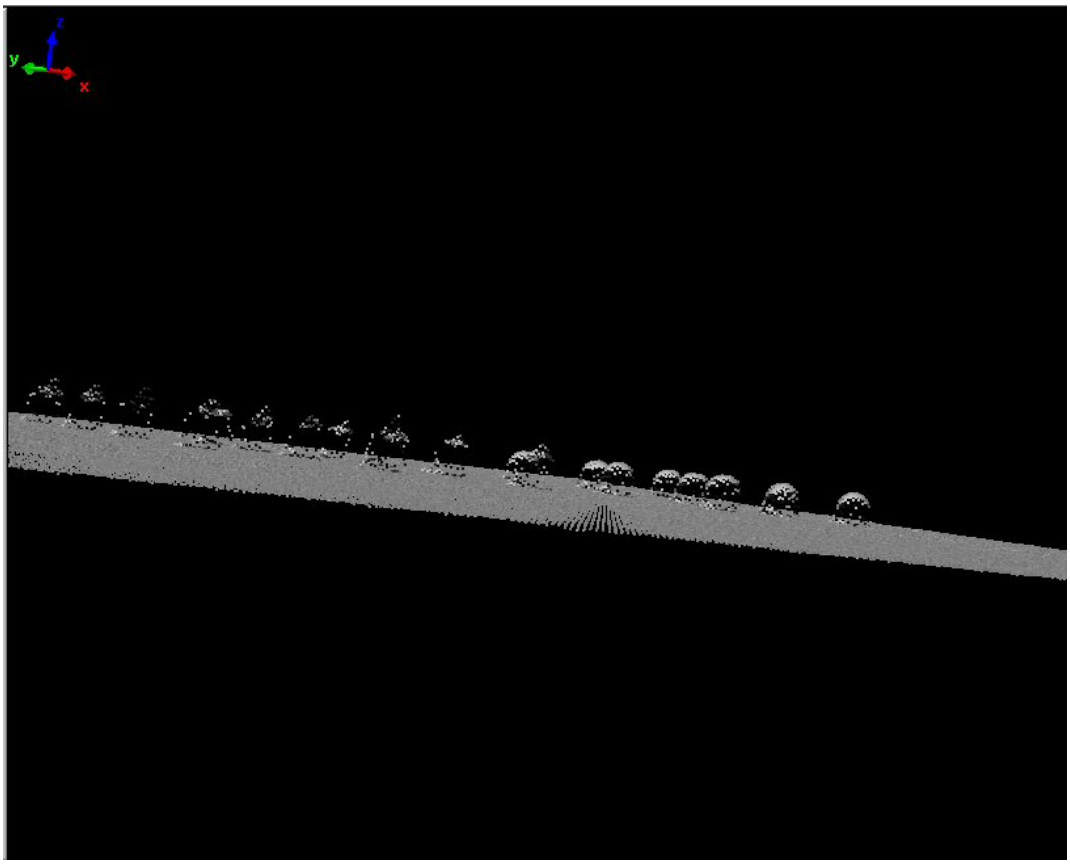


In addition to developing the targets, research was conducted on developing an algorithm to rapidly locate the targets from the data set. Matlab mathematical software was used as the framework. In addition to custom software, the software packages that came with the two laser systems, Polyworks and Minolta Polygon Editor were also evaluated on their potential to perform the same function as the custom algorithm. The downside to using the commercial applications is that the processing of the targets had to be done manually on a target by target basis, a task that is exceptionally labor and time intensive.

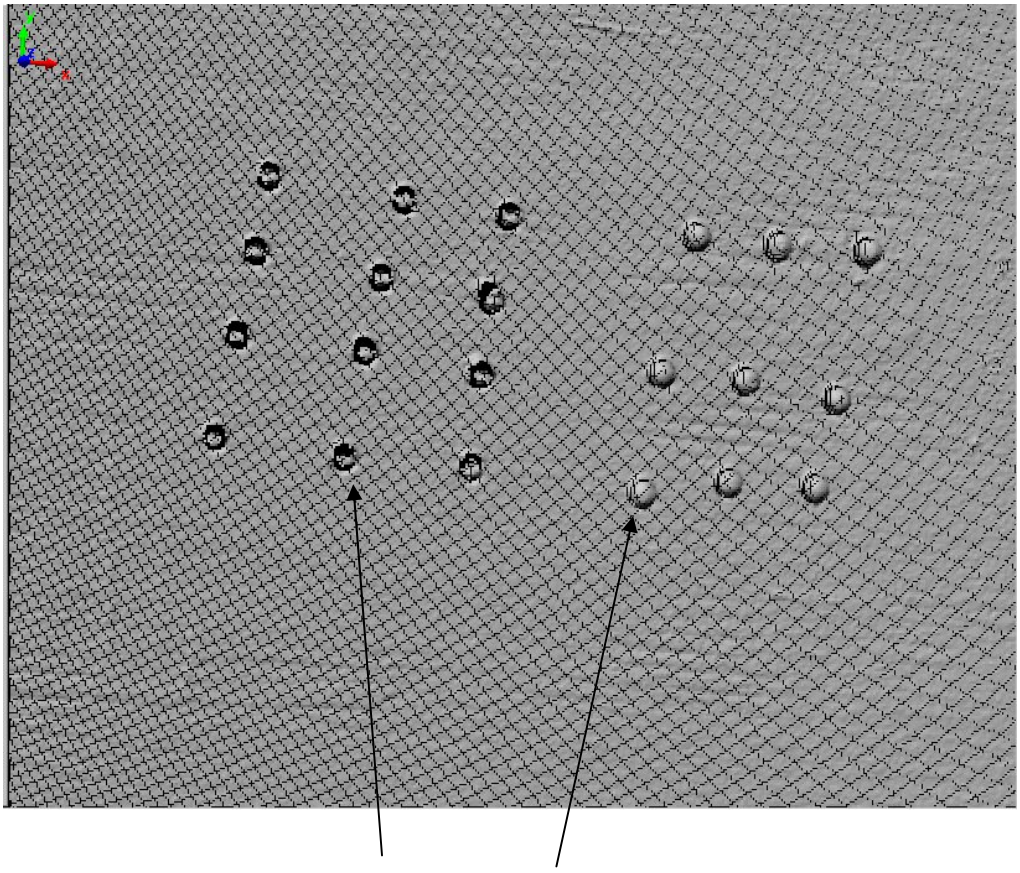
Target Detection

The following two images show scans of targets affixed to structural beams. Both show the effect of spraying the targets with a very thin developer coating. This creates a better reflective surface for the laser pulses to return from.

The Minolta Vivid 910 was the device used to scan the objects. The plan was to use the relative displacement of them to determine strain quantities. A substantial amount of effort was expended in this quest, and unfortunately, it was not entirely successful, due to software issues. However, much was learned about the technique and the difficulties encountered will be used this summer to create a new algorithm.

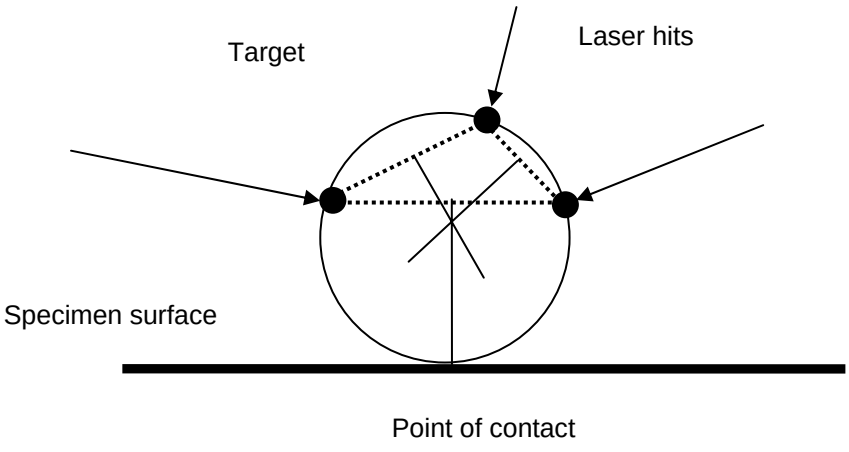


1 mm spherical targets affixed to beam. (Note, the ones on the left are not sprayed with developer, while the ones on the right are.



Uncoated versus coated targets

As seen in the sketch below, knowing the location of at least 3 points on the surface (of course more is better) of the sphere allows one to bisect connecting lines to locate the centroid. Then knowing the orientation of the beam's surface, one can project a vertical line down onto the specimen. This would be the point monitored for relative movement.



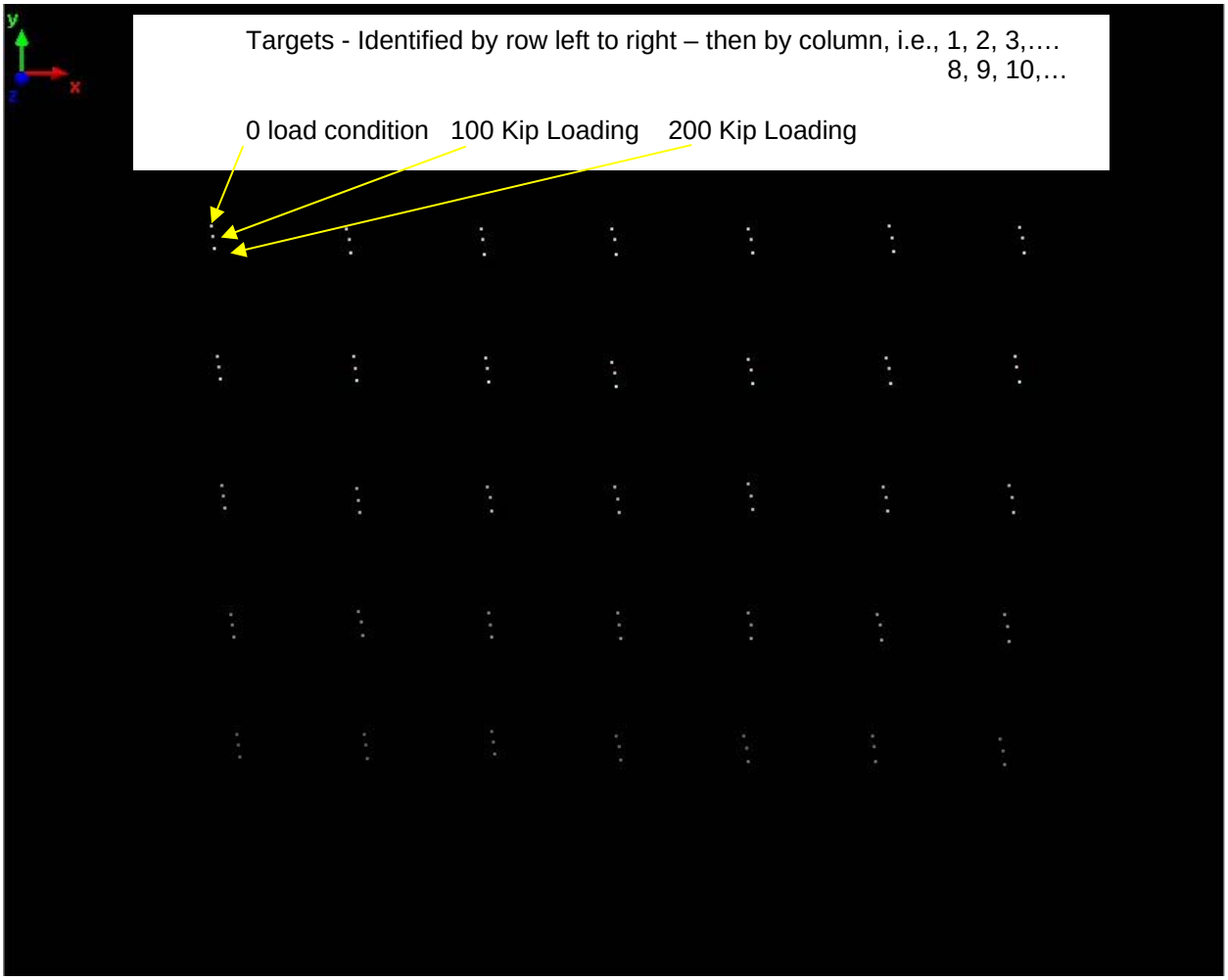
After numerous tests, it is felt that this methodology, i.e., using this sphere technique to locate points of contact was successful. We were able to locate points consistently – even as the angle of the laser pulse changes. This is a very important finding, since without a high degree of location repeatability, any subsequent strain computations will be suspect.

Tallahassee, FL Scan

In order to evaluate the system for use as a strain data plotter, the FDOT Structures Research Center in Tallahassee, FL was contacted for help. The idea was to use the Minolta VIVID 910 to scan a concrete beam's vertical face while being loaded to failure. The targets used were 0.1875 and 0.1233 inch diameter ball bearings (spherisity approaching 99.3%). These were glued onto the concrete surface in the vicinity of the loading point of loading and the expected failure zone. The idea was to use the ball bearings as targets (described previously) and once the midpoints of them determined the relative movements of the targets between scans used to calculate the strain induced in the beam.

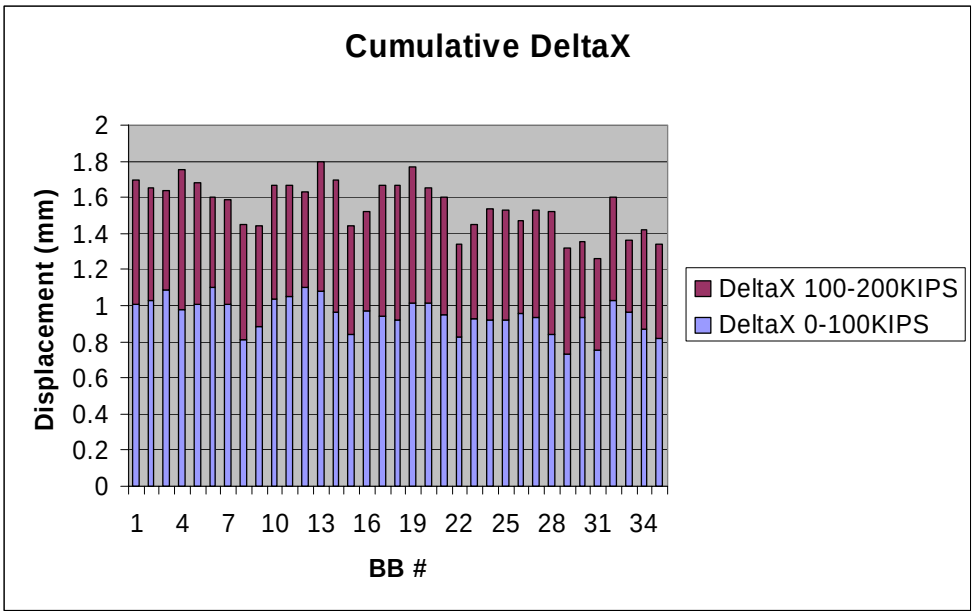
While the computation of relative movements was not achieved to the degree desired, a major accomplishment is the determination of the most appropriate types of targets. That is to say, the laser must return a signal to the CCD collector and hence without a distinct point of reference to use in the analysis, small displacements are not achievable. It bears mentioning that current commercial targets are simply reflective circular pads affixed to the specimen. However, this does not allow for the precision needed to monitor movement of them under loading.

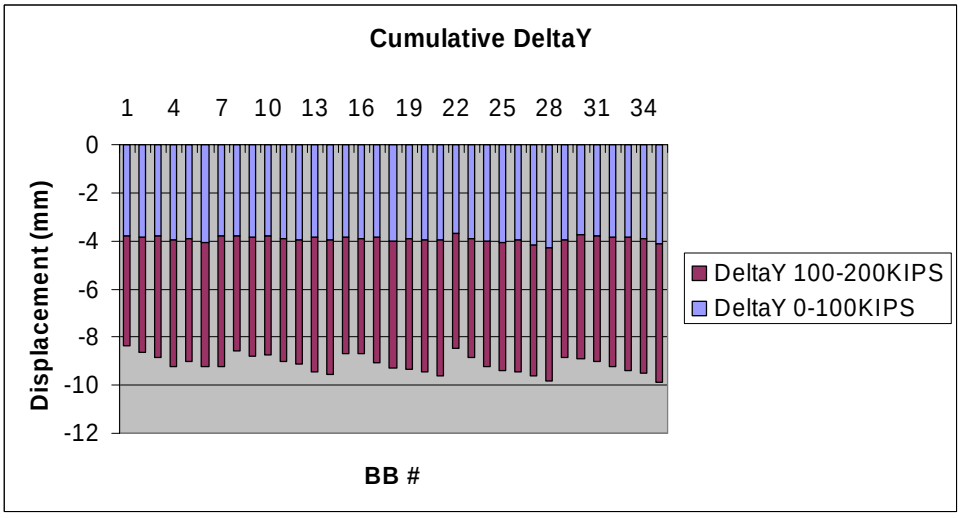
The following figures show the results of the test. The 35 targets were arranged in a grid and successive scans were taken for two loading conditions. The relative movement of each is apparent, showing that the technique looks promising.

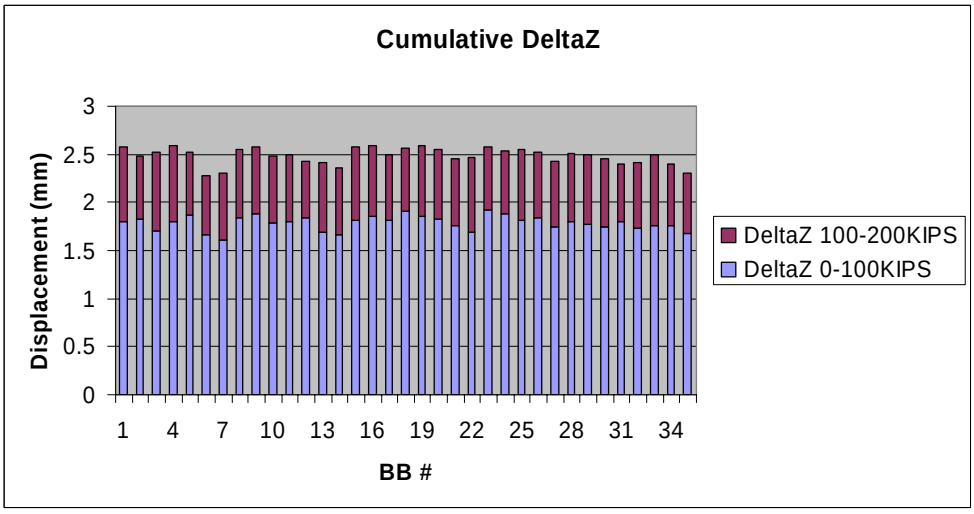


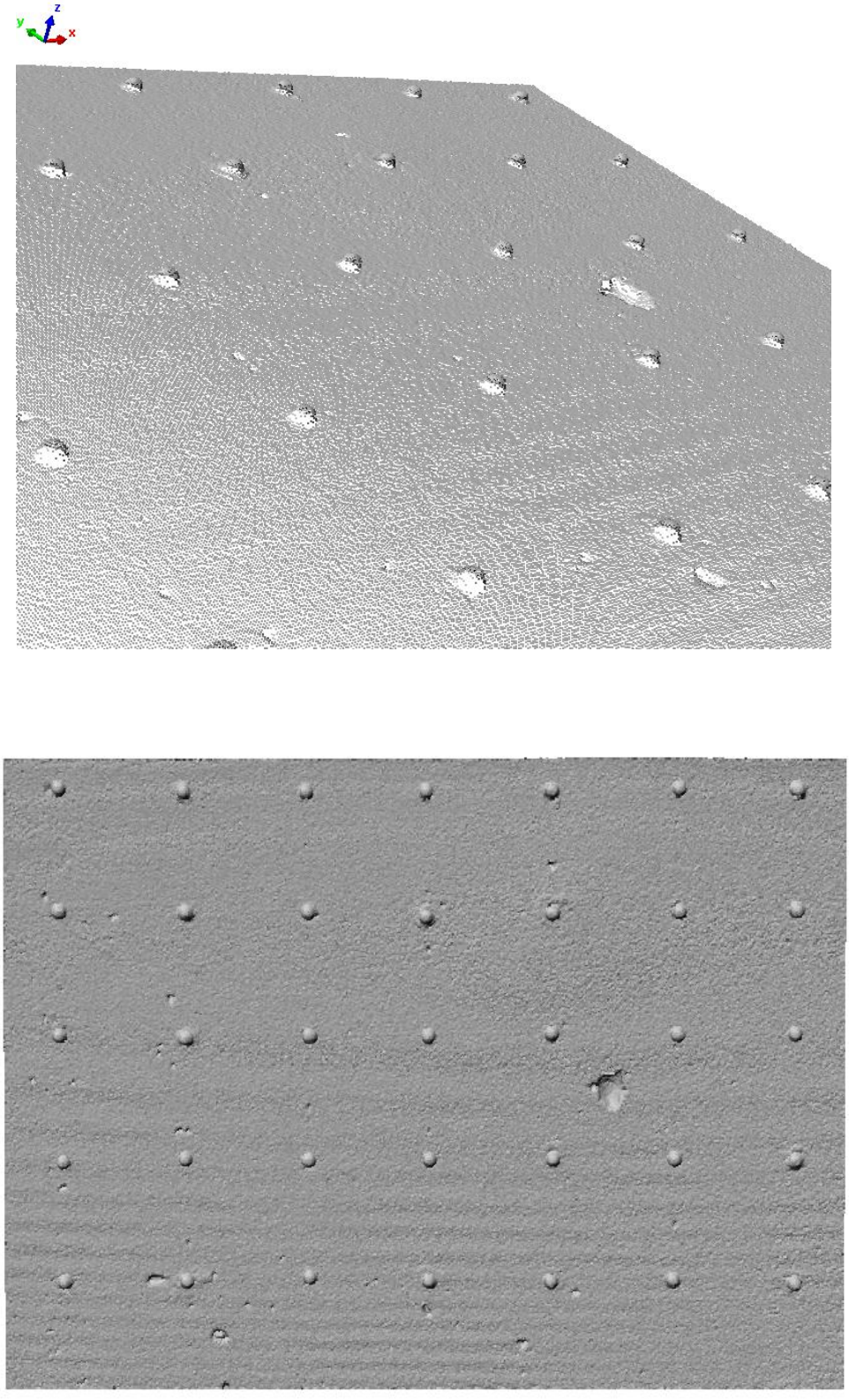
Relative target movements affixed to concrete beam face

These data were plotted in Excel and are shown on the following pages. The results look promising in the fact that the movement of targets furthest from the neutral axis exhibit the largest deflections (the Y and X axes). What is even more interesting is the Z axis data which shows that the entire beam is rotating out of plane a very consistent amount. This shows that the laser is superior to strain gages in this regard since they provide plane strain information only. Following are scans showing the no-load, and 100 and 200 kip loads. The cracking of the beam is readily apparent and the width easily computed.

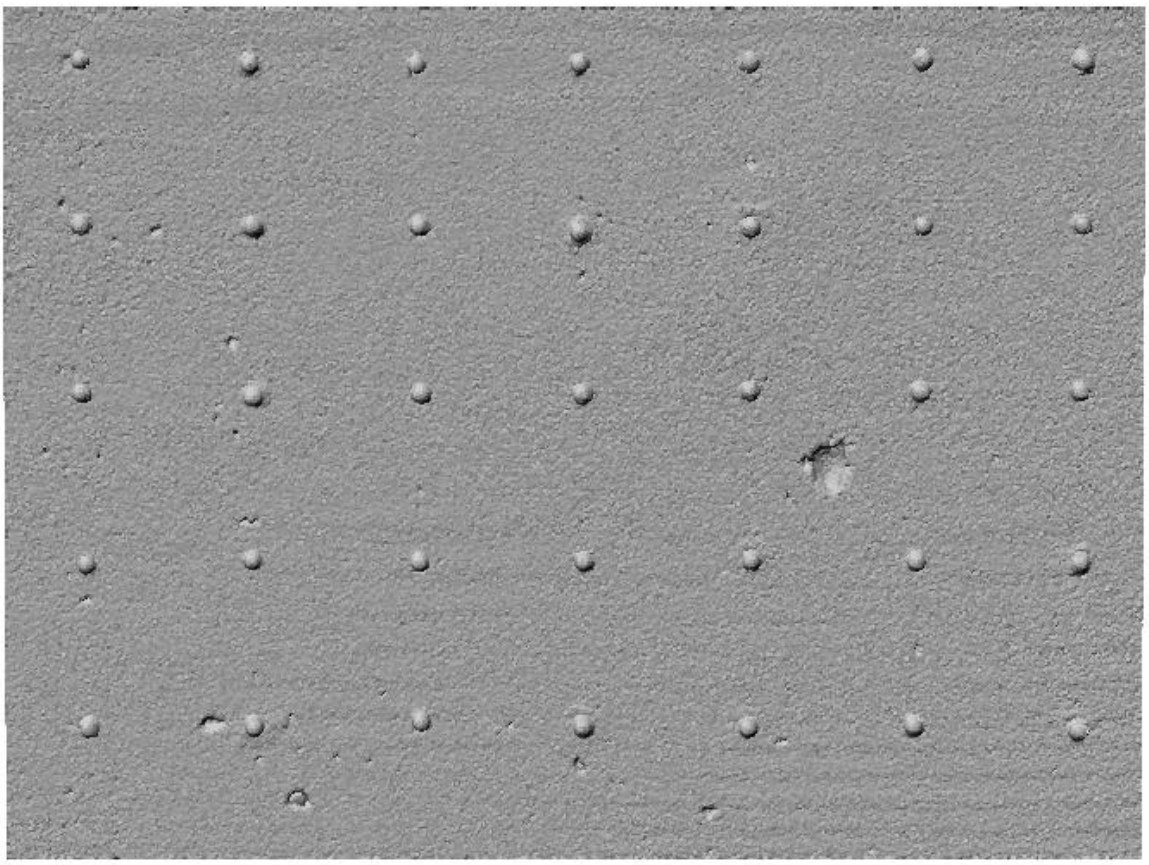

Cumulative DeltaX
Displacement (mm)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 4 7 10 13 16 19 22 25 28 31 34
BB #
DeltaX 100-200KIPS
DeltaX 0-100KIPS


Cumulative DeltaY
1 4 7 10 13 16 19 22 25 28 31 34
Displacement (mm)
0
-2
-4
-6
-8
-10
-12
BB #
DeltaY 100-200KIPS
DeltaY 0-100KIPS

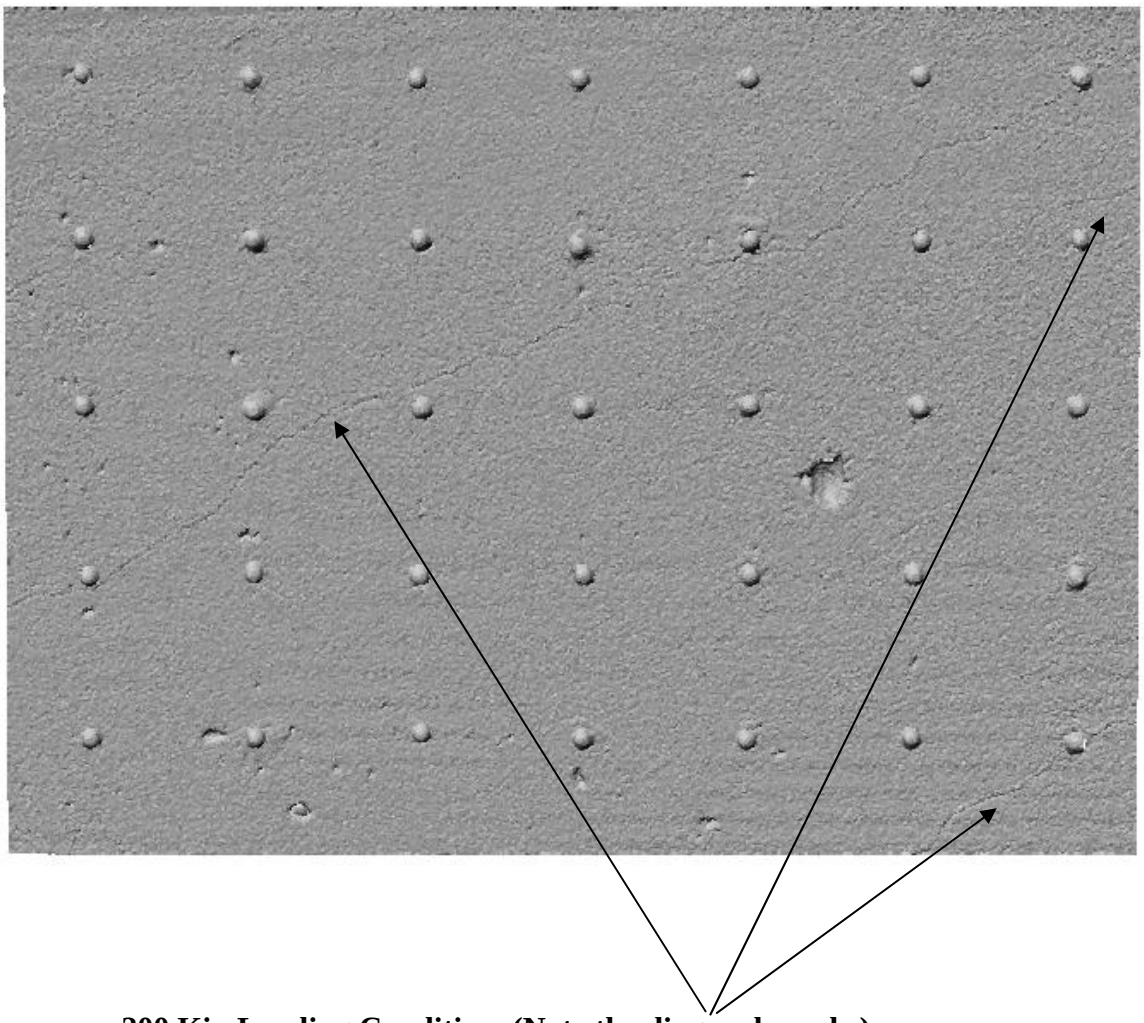

Cumulative DeltaZ
Displacement (mm)
3
2.5
2
1.5
1
0.5
0
1 4 7 10 13 16 19 22 25 28 31 34
BB #
DeltaZ 100-200KIPS
DeltaZ 0-100KIPS



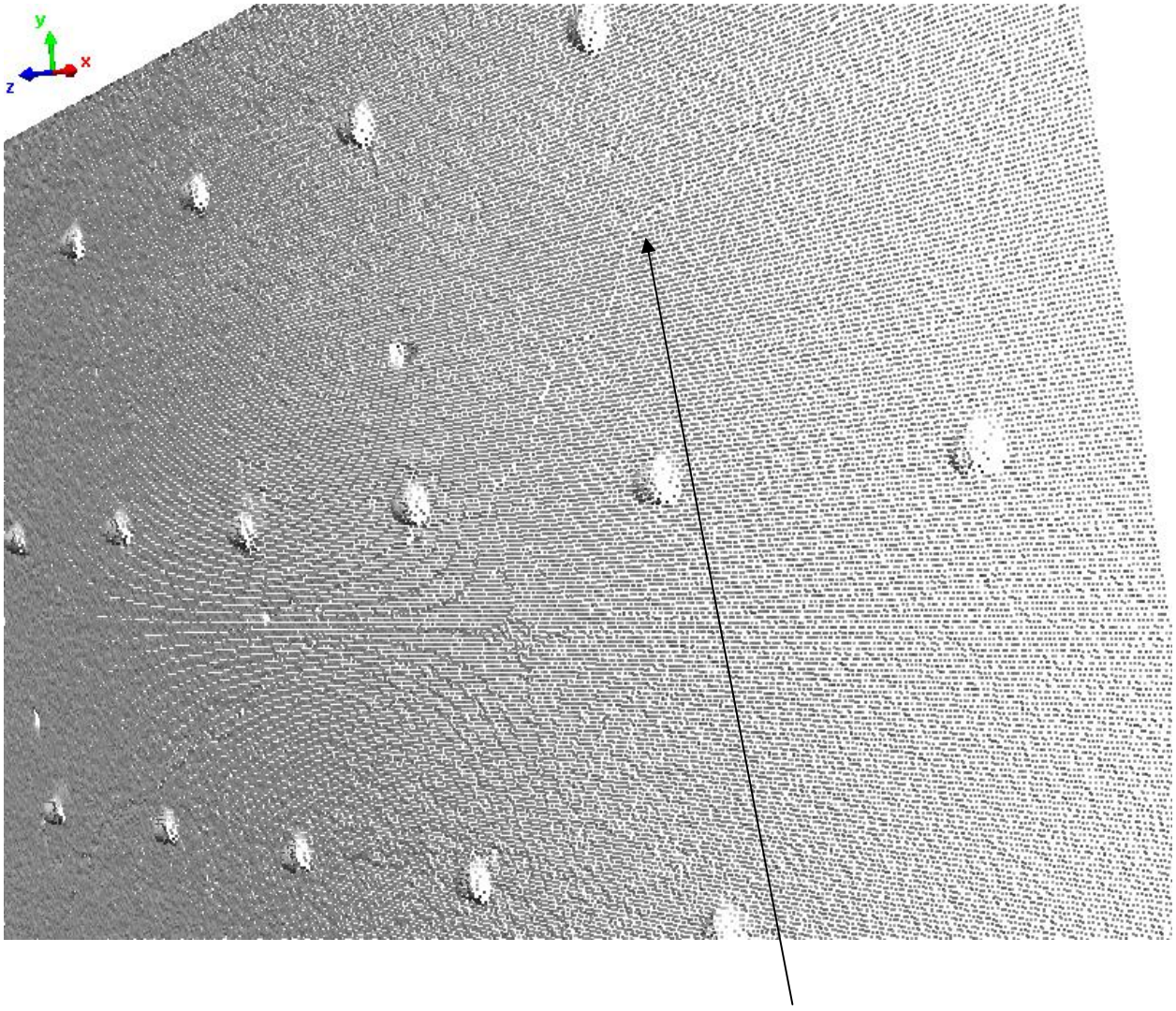
No load condition - (note the small concrete imperfections)



100 Kip Loading Condition
(it appears that the surface is becoming grainy compared to the no-load surface)



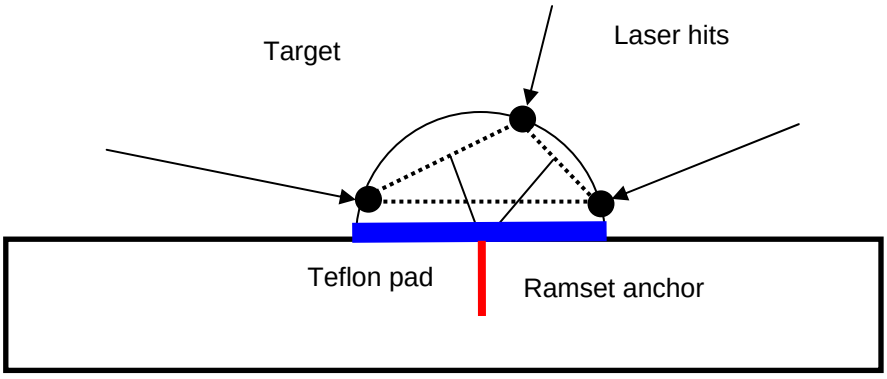
200 Kip Loading Condition. (Note the diagonal cracks).



200 Kip Loading Condition. Note crack.

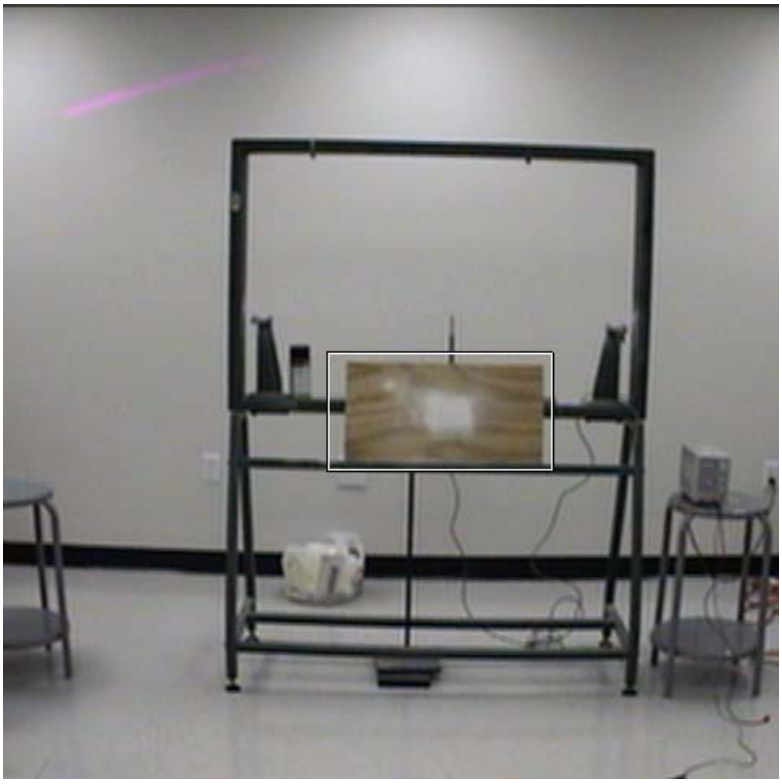
ILRIS Targets Discussion

For the bridge targets, it is more appropriate to use small dome shaped objects. First, they can be affixed to the element more stoutly and second, their size is not an overriding concern. This is because as the target size increases, the likelihood of multiple hits increases. And if the point of contact is small between the target and specimen, the size will not degrade the measurement. One idea is to insert a Ramset anchor into the concrete and then screw on the target. A Teflon pad at the bottom of the target prevents friction buildup between the target and the surface of the element and results in a very small point of contact. Below is a tentative design for a permanent target (say for a bridge girder)



Targets were constructed using both Styrofoam and wood and affixed to a wall in a UF lab. A photo of the setup is shown on the following page. Two targets (one styro and the other wood) were spaced exactly 24 inches apart and scanned. Very good target recognition was achieved, albeit the wood fared better. This is because the rough texture of the Styrofoam deflects some of the laser pulses reducing the number of return hits. It is interesting to note that a very smooth object also refracts the pulses reducing the amount of returns. So it appears that a textured surface (similar to a 400 – 600 grit sandpaper) is optimum for pulse acquisition.

Target acquisition. The faint target on the left is Styrofoam, on the right, wood



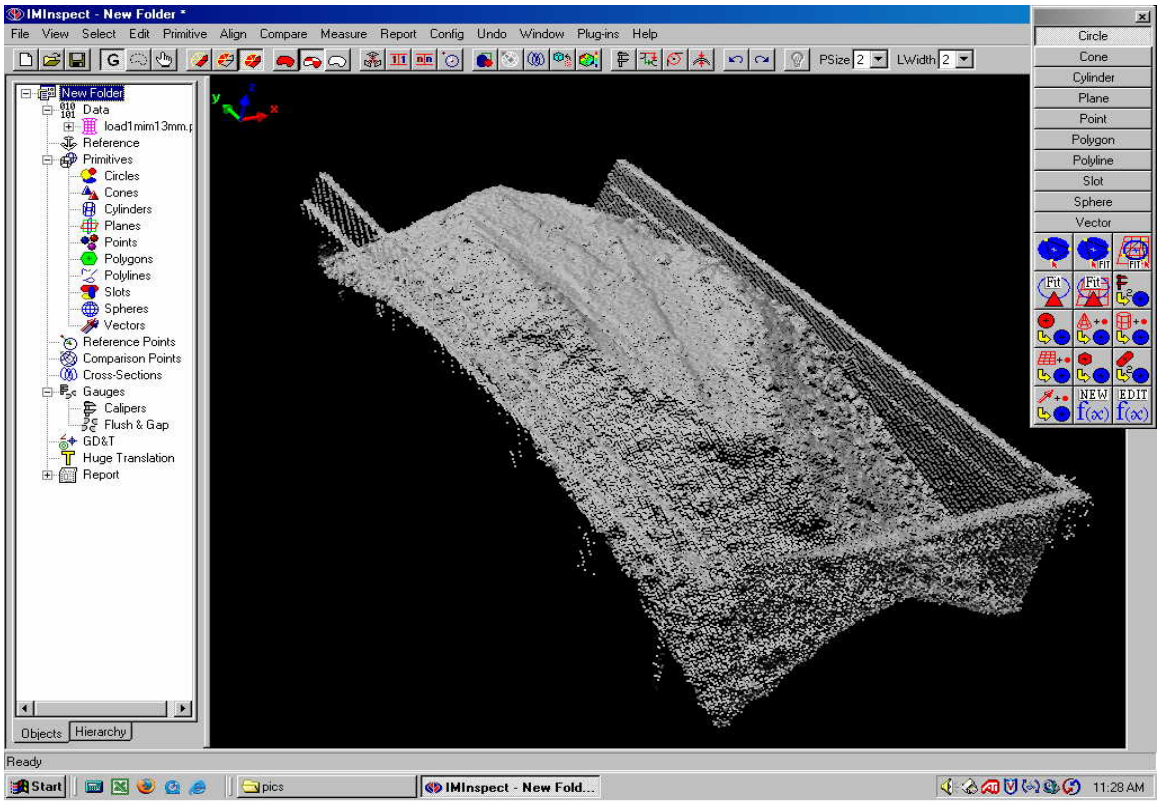
Test set-up to determine the amount of developer needed to obtain strong returns.

Truck Volume Determination

The examples of the ILRIS bridge scans are provided in the attached PowerPoint Presentation. However, another novel use of the device is to compute the volume of soil in trucks leaving borrow pits. FODT wanted to see if this technology could perform this function. Hence, the scanner was set up on an elevated platform and scanned an empty truck. Then, a full truck was scanned and Polyworks software used to compute the volume. This was compared to a physical measurement. As can be seen in the table below, the longer the truck is scanned, the better the correlation. This is reasonable since more data points are obtained with longer scan durations.

Scan Time	Empty Volume (ft^3)	Unused Volume (ft^3)	Volume of soil (ft^3)	Computed Empty Volume (ft^3)	Computed Unused Volume (ft^3)	Computed Volume of soil (ft^3)
1min	474.8	100.9	373.9	482.3	94.4	387.9
2min	474.8	100.9	373.9	480.9	95.1	385.8
3min	474.8	100.9	373.9	479.4	97.4	382.0
6min	474.8	100.9	373.9	477.5	98.5	379.0

Empty Haul Truck

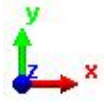


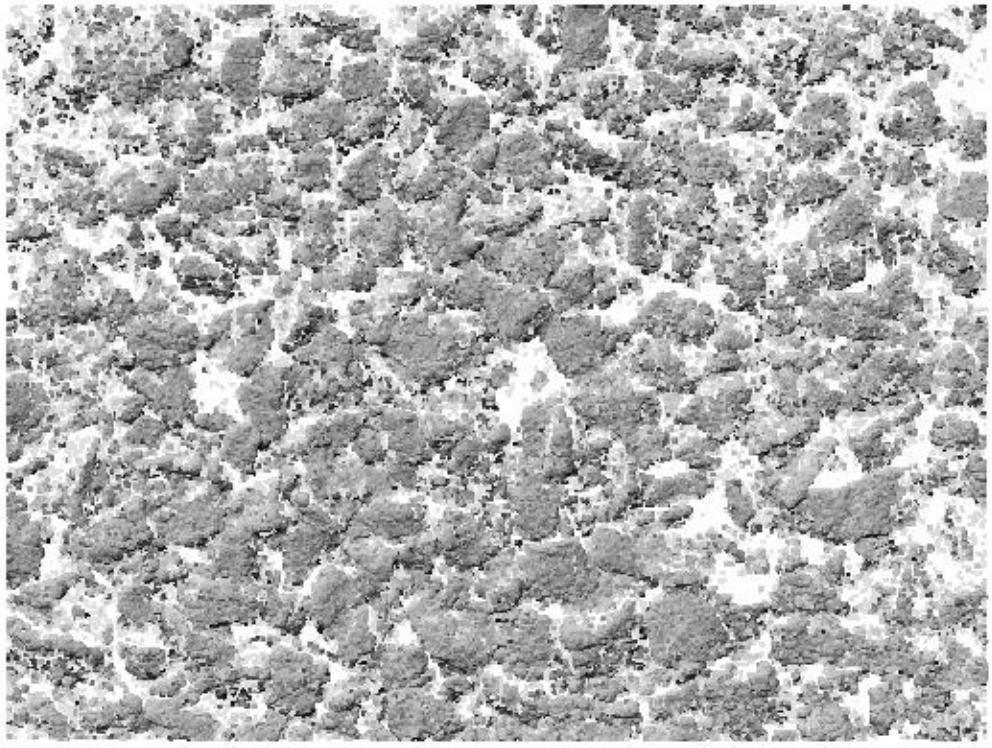
Scanned haul truck's surface load

Digital Photo of Loaded Haul Truck

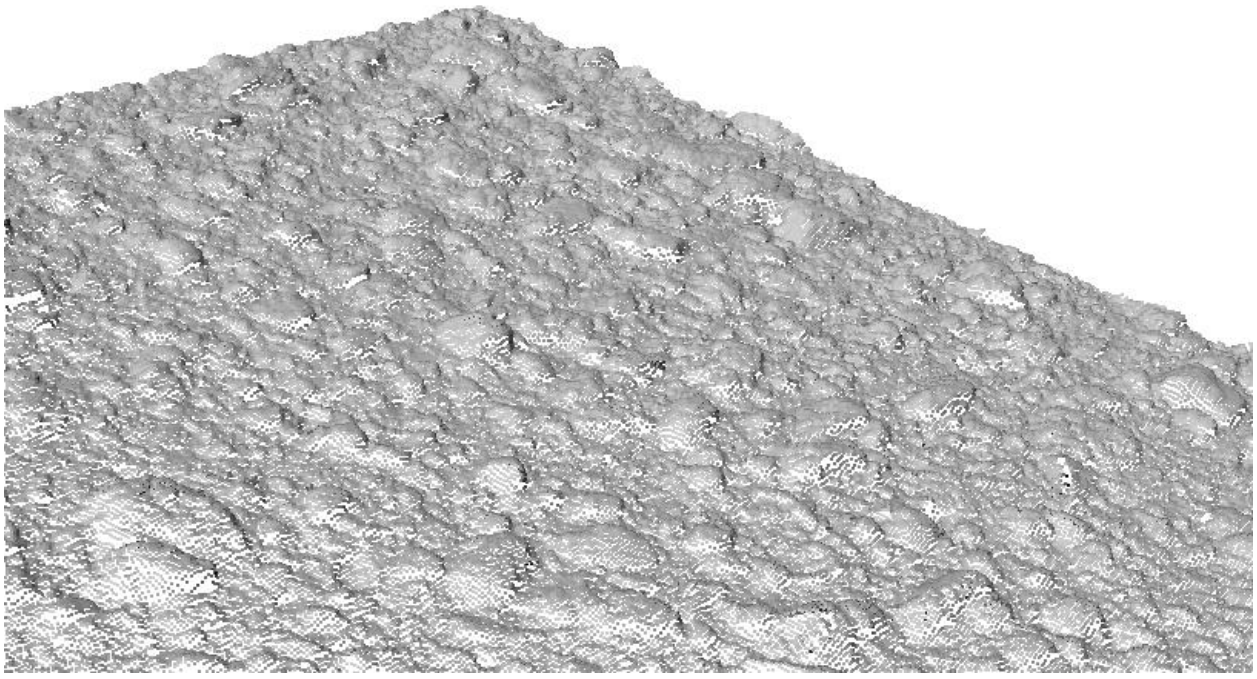
Pavement Texture Measurement

The State Materials Office requested that we attempt to scan pavement surfaces in order to determine the texture of the material. This is very important in accident analysis, since hydroplaning potential increases as the surface becomes smoother. Over 30 scans were performed. Some of the images are shown below. Based on the results, we have suggested that it is possible to compute the relative amount (based on volume) of material that extends above a certain datum. That is to say, a very rough surface would have a substantial amount of material raised above the reference plane, while a worn one would not. Hence, this device could be used to monitor (quantify) the surface quality and to indicate when re-surfacing is needed.

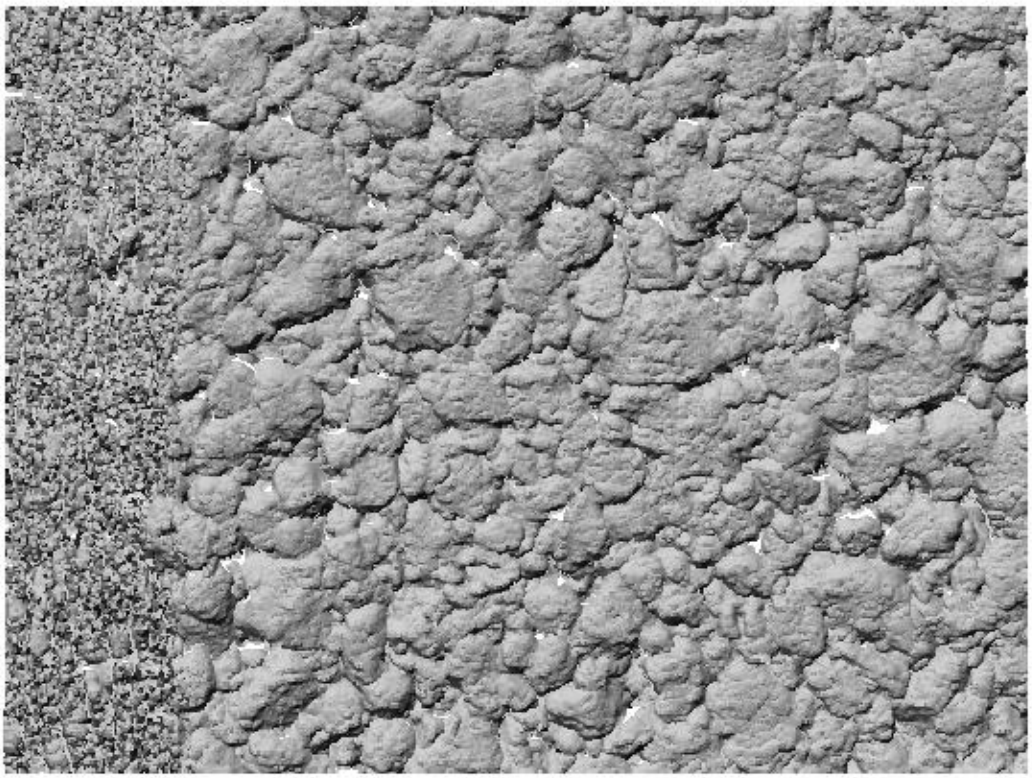




Asphalt Concrete Surface Scan.



Worn Asphalt Surface Scan.



Gradation difference in segregated asphalt concrete.

The plan would be to scan a freshly placed pavement section and compute the amount of solid particles at various elevations (e.g., 45% - 13 mm above datum, 21% - 9 mm, 14% above 3 mm, etc.). Then subsequent scans would reveal the amount of change that occurred due to surface wear. We plan to continue this effort during the summer, 2005.

Conclusions

The major objectives of this research have been met. The two devices are working properly, novel targets have been created and the instruments have been used on several projects to test their viability.

The only objective yet to be met is the development of real-time analysis software. The program that was created does work, (code is shown on the following sheets), but not to our satisfaction. Hence, we plan to continue development and produce a working model this summer.

Minolta Scanner Target Locator/Displacement Program

The following is the source code developed for the Minolta scanner. As mentioned previously, this was an attempt to create a program that would automatically track and analyze relative movement of a series of targets affixed to a structural element. While the program worked – albeit crudely, it did not perform as well as expected nor desired. Hence, we will revisit this issue this summer using another formulation.

```
/*"center.c" uses the target diameter 0.1875 inches and 0.1233 inches, assumes the center of the
 target not far from the average value of x,y,z of laser-hit points of a target, from here shifts
 up and down for y, further away for z, left and right for x, to find the location of the minimum
 error of ideal sphere surface from the laser points. To find the center of the target, "center.c"
 move the center in 0.005mm, and allow a bias of 2 square mm(Sure you can make it smaller, but
 considering the inexact nature of target, laser instrument, and the requirement of our purpose,
 I think this is enough), and see what error this center will lead. Because to predict a sphere,
 we need at least 4 laser points, for those targets with less than 4 laser-hit points the program
 got, I still keep the average as the center of the target. */
#include <stdio.h>              /* Input : inputdile1, inputfile2 */
#include <stdlib.h>             /* Output: outputfile             */
#define inputfile1 "epcn.txt"    /* inputfile1 is the geometry center of target guessed by num.c */
#define inputfile2 "edtnco.txt"  /* inputfile2 is laser points on a target, which is from num.c and processed by excel */
#define outputfile "ecent.txt"
main()
{
            FILE *dFile, *cFile, *oFile;
            int num,i,status;
            float cen_x,cen_y,cen_z,sum,n_sum,err,rad,radium2=2.296875,radium1=1.510425;
            float precision=0.005,threshhold=2;
   float sb_begin_x=-38.0,sb_begin_y=72.0,sb_end_x=52.0;   /* This line is the same as in Test.c */
            struct Point{
            float x;
                                                float y;
                                                float z;
            int ord;
                                               } point[80];                  /**** [*] should be changed for different file ****/
 cFile = fopen(inputfile1,"r");
 if (cFile==NULL)
 {
              printf("File open error. Please check your data file is under the right directory");
   abort(1);
            }
 dFile = fopen(inputfile2,"r");
 if (dFile==NULL)
 {
            printf("File open error. Please check your data file is under the right directory");
   abort(1);
            }
 oFile = fopen(outputfile,"w");
 if (oFile==NULL)
 {
            printf("File open error. Please check your data file is under the right directory");
   abort(1);
            }

 do{
              status=fscanf( cFile , "%f",&cen_x);
              if(status == EOF) break;
    fscanf( cFile , "%f",&cen_y );
    fscanf( cFile , "%f",&cen_z );
    fscanf( cFile , "%d",&num);
    for (i=0;i<num;i++)
             {
     fscanf( dFile , "%f",&point[i].x );
     fscanf( dFile , "%f",&point[i].y );
     fscanf( dFile , "%f",&point[i].z );
```

```
 fscanf( dFile , "%d",&point[i].ord);
}
if(num<4)
        {
                fprintf(oFile,"%f ",cen_x);
                fprintf(oFile,"%f ",cen_y);
                fprintf(oFile,"%f ",cen_z);
                fprintf(oFile,"%d \n",num);
 continue;
        }

if ((cen_x>sb_begin_x) && (cen_y<sb_begin_y) && (cen_x<sb_end_x)) rad=radium1;   /* This line definition is the same as in Test.c */
else rad=radium2;
sum=0;          /* get ready to find more exact center */
for (i=0;i<num;i++)
        {
        err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].x-cen_z)-rad*rad;
 if(err<0) err = -err;
        sum+=err;
         }

/* Different direction access to center will lead to different resuld, I chose from z down */
 n_sum=0;
do {
        cen_z-=precision;
    for (i=0;i<num;i++)
                                {
         err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].x-cen_z)-rad*rad;
     if(err<0) err = -err;
         n_sum+=err;
                                }
            if (n_sum<sum)
                                {
                sum=n_sum;
                n_sum=0;
                                }
            else if (n_sum>sum+threshhold) /* I changed threshhold from 1`20, found between 1`10 there is no chang for sum and center */
                                {
              cen_z+=precision;
              break;
                                }
                                }while (1);
n_sum=0;
do {
        cen_y-=precision;
    for (i=0;i<num;i++)
                                {
              err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].z-cen_z)-rad*rad;
     if(err<0) err = -err;
              n_sum+=err;
                                }
            if (n_sum<sum)
                                {
                        sum=n_sum;
                        n_sum=0;
                                }
            else if (n_sum>sum+threshhold) /* I changed threshhold from 1`20, found between 1`10 there is no chang for sum and center */
                                {
              cen_y+=precision;
              break;
                                        }
                                }while (1);
 n_sum=0;
 do {
            cen_y+=precision;
    for (i=0;i<num;i++)
                                {
              err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].z-cen_z)-rad*rad;
     if(err<0) err = -err;
         n_sum+=err;
                                }
```

```
                if (n_sum<sum)
                                {
                        sum=n_sum;
                        n_sum=0;
                                }
                 else if (n_sum>sum+threshhold) /* I changed threshhold from 1`20, found between 1`10 there is no chang for sum and center */
                                {
                   cen_y-=precision;
                                break;
                                }
                                }while (1);
        n_sum=0;
        do {
                        cen_x-=precision;
            for (i=0;i<num;i++)
                                {
                         err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].z-cen_z)-rad*rad;
             if(err<0) err = -err;
                         n_sum+=err;
                                }
                        if (n_sum<sum)
                                {
                                sum=n_sum;
                                n_sum=0;
                                }
                        else if (n_sum>sum+threshhold) /* I changed threshhold from 1`20, found between 1`10 there is no chang for sum and center */
                                {
                          cen_x+=precision;
                                break;
                                }
                                }while (1);
        n_sum=0;
        do {
                        cen_x+=precision;
            for (i=0;i<num;i++)
                                {
                         err=(point[i].x-cen_x)*(point[i].x-cen_x)+(point[i].y-cen_y)*(point[i].y-cen_y)+(point[i].z-cen_z)*(point[i].z-cen_z)-rad*rad;
             if(err<0) err = -err;
                         n_sum+=err;
                                }
                        if (n_sum<sum)
                                {
                                sum=n_sum;
                                n_sum=0;
                                }
                        else if (n_sum>sum+threshhold) /* I changed threshhold from 1`20, found between 1`10 there is no chang for sum and center */
                                {
                          cen_x-=precision;
                                break;
                                }
                           }while (1);
        fprintf(oFile,"%f ",cen_x);
                                fprintf(oFile,"%f ",cen_y);
                                fprintf(oFile,"%f ",cen_z);
                                fprintf(oFile,"%d \n",num);
   }while(1);
   fclose(cFile);
   fclose(dFile);
   fclose(oFile);
}
-------------------------------------------------------------------------------------------------------------------------
#include <stdio.h>
main()
{
                FILE *dFile1, *dFile2, *dFile3, *oFile;
                int i,j,p_num=219;
                struct Point{
                float x;
                                                        float y;
                                                        float z;
                                                        int num;
```

```
                                        } temp_a,temp_b,temp_c,temp;
   dFile1 = fopen("aaapoin.txt","r");
   if (dFile1==NULL)
              {
               printf("File open error. Please check your data file is under the right directory");
    abort(1);
               }
   dFile2 = fopen("aabpoin.txt","r");                    /**** "*" should be changed for different file ****/
   if (dFile2==NULL)
              {
               printf("File open error. Please check your data file is under the right directory");
    abort(1);
               }
   dFile3 = fopen("aacpoin.txt","r");                    /**** "*" should be changed for different file ****/
   if (dFile3==NULL)
              {
               printf("File open error. Please check your data file is under the right directory");
    abort(1);
               }
              oFile = fopen("abee.txt","w");          /* *bee for biggest common, *beeb for include all */
   if (oFile==NULL)
              {
               printf("File write error. Please check your disk space");
    abort(1);
               }
              for (i=0;i<p_num;i++)
              {
                        fscanf(dFile1,"%f",&temp_a.x);
                        fscanf(dFile1,"%f",&temp_a.y);
                        fscanf(dFile1,"%f",&temp_a.z);
                        fscanf(dFile1,"%d",&temp_a.num);
                        fscanf(dFile2,"%f",&temp_b.x);
                        fscanf(dFile2,"%f",&temp_b.y);
                        fscanf(dFile2,"%f",&temp_b.z);
                        fscanf(dFile2,"%d",&temp_b.num);
                        fscanf(dFile3,"%f",&temp_c.x);
                        fscanf(dFile3,"%f",&temp_c.y);
                        fscanf(dFile3,"%f",&temp_c.z);
                        fscanf(dFile3,"%d",&temp_c.num);
                        temp.num=temp_a.num+temp_b.num+temp_c.num;
                        if(temp_a.num&&temp_b.num&&temp_a.num)   /*if(temp.num!=0)  */
                        {
                        temp.x=(temp_a.x*temp_a.num+temp_b.x*temp_b.num+temp_c.x*temp_c.num)/temp.num;
                        temp.y=(temp_a.y*temp_a.num+temp_b.y*temp_b.num+temp_c.y*temp_c.num)/temp.num;
                        temp.z=(temp_a.z*temp_a.num+temp_b.z*temp_b.num+temp_c.z*temp_c.num)/temp.num;
                        }
                        else temp.x=temp.y=temp.z=0.0;
                        fprintf(oFile,"%f %f %f %d\n",temp.x,temp.y,temp.z,temp.num);
              }
              fclose(oFile);
              fclose(dFile1);
              fclose(dFile2);
              fclose(dFile3);
}
```

```
#include <stdio.h>              /* Function :cull out laser points which seems impossible */
#define inputfile1 "ap.txt"     /* Input : inputfile1, inputfile2 */
#define inputfile2 "adto.txt"   /* Output: outputfile             */
#define outputfile "adtono.txt" /* inutfile1 is used to get number of points the present target has */
                        /* inputfile2 has the target laser points information */
main()                      /* outputfile is the target data reasonable */
{
              FILE *fd, *fp, *ft;
              int i, j, k, maxp,l_points, sign,order;
              float zmax, flo;
              struct Point
                                                  {
              float x;
                                                    float y;
                                                    float z;
```

```
                                                int group;
                                               } point[70];                              /**** "*" should be changed according to different import data
file ****/
fp= fopen(inputfile,"r");
 if (fd==NULL)
  {
                printf("File open error. Please check your data file is under the right directory");
    abort(1);
                }
fd= fopen(inputfile2,"r");
 if (fd==NULL)
  {
                printf("File open error. Please check your data file is under the right directory");
    abort(1);
                }
ft = fopen(outputfile,"w");
 if (ft==NULL)
  {
                printf("File open error. Can't create file");
    abort(1);
                }
 order = 0;
 do {
     sign = fscanf( fp , "%f",&flo );  /* 3 fscanf to get rid of 3 unused float on the data file */
     if (sign == EOF) break;
     fscanf( fp , "%f",&flo );
     fscanf( fp , "%f",&flo );
     fscanf( fp , "%d",&l_points );

     for (i=0;i<l_points;i++)
                                {
                                        fscanf( fd , "%f",&point[i].x );
                                        fscanf( fd , "%f",&point[i].y );
                                        fscanf( fd , "%f",&point[i].z );
                                        fscanf( fd , "%d",&point[i].group );
                                }
     /* find the maximum point */
     zmax = point[0].z; maxp = 0;
     for (i=1;i<l_points;i++)
                               {
                                 if (point[i].z>zmax) {
                                        zmax = point[i].z;maxp=i;
                                 }
                               }
     for (i=0;i<l_points;i++)
                               {
                                 if(point[i].z>(zmax-3.5))
          fprintf(ft, "%f %f %f %d \n",point[i].x,point[i].y,point[i].z,order);
                               }
     order++;
 } while(sign != EOF);
 fclose(fp);
 fclose(fd);
 fclose(ft);
}
-------------------------------------------------------------------------------------------------------------------
#include <stdio.h>                    /* This program is for finding the points' number of each scan and the order number of the end of the scan    */
#include <stdlib.h>

void main()
{
  FILE *pFile, *dFile;
  int i,j,pos;                        /* pos keep the present point's order number */
                                     /*Position of the target center*/
  int intens,read_dn=1;
  float x,y,z,xf,yf,zf;
  pFile = fopen ("D:\\Hamilton\\b.asc","r");          /**** "*" should be changed for different file ****/
  if (pFile==NULL)
  {
                printf("File open error. Please check your data file is under the right directory");
    abort(1);
```

```
            }
 dFile = fopen("D:\\Laser Stuff\\blen.txt","w");        /**** "*" should be changed for different file ****/
 if (dFile==NULL)
  {
            printf("File create error!");
    abort(1);
            }
 i=0;
 xf=-5;yf=0;zf=0;
 pos=0;
 do{
    if(read_dn!=EOF) read_dn=fscanf(pFile,"%f",&x);
    if (read_dn==EOF) break;
    fscanf(pFile,"%f",&y);
    fscanf(pFile,"%f",&z);
    fscanf(pFile,"%d",&intens);
    if (x<xf-1.4)
            {   pos+=i;
                    fprintf(dFile,"%d %d\n",i,pos);
                    i=0;
            }
            xf=x;yf=y;zf=z;
            i++;
            }while(1);
   fclose(pFile);
   fclose(dFile);
}
```

```
---------------------------------------------------------------------------------------------------------------------- #include <stdio.h>                /*
This program is for figuring out the center of a target by proximity */
#define inputfile "D:\\Laser Data\\3-22-04\\baad.txt"    /* It groups data according to position (guessed target) */
#define outputfile1 "D:\\Laser Data\\3-22-04\\baao.txt"  /* Input :inputfile */
#define outputfile2 "D:\\Laser Data\\3-22-04\\baan.txt"    /* Output:outputfile1,outputfile2 */
main()                 /* outputfile1 keeps all the points information; outputfile2 keeps the center of the target */
{
 FILE *dFile,*ndFile,*pFile; /* "data_size" and struct "point" dimension is the number of lines of *.data.txt */
 int i,j,k,level,num,begin,next_begin,token,data_size=284;          /**** data_size should be changed for different import data file ****/
 float xavg, yavg,zavg;
            struct Point{
                    int d_row;
            float x;
                            float y;
                            float z;
                            int group;
                           } point[284];
 dFile = fopen(inputfile,"r");
 if (dFile==NULL)
  {
            printf("File open error. Please check your data file is under the right directory");
    abort(1);
            }
for (i=0;i<data_size;i++)
            {
      fscanf( dFile , "%f",&point[i].x );
      fscanf( dFile , "%f",&point[i].y );
      fscanf( dFile , "%f",&point[i].z );
   /*  fscanf( dFile , "%d",&point[i].d_row ); *//* check your data file to see if you need this line, or if this line should be moved to next up/down
3 lines */
            point[i].group= 0;
      }
level=1;
begin=0;
do{
   token = 0; /* Not find new beginning position yet */
   point[begin].group = level;
   i = begin+1;
   do{
      if ((point[i].x-point[begin].x)*(point[i].x-point[begin].x)+(point[i].y-point[begin].y)*(point[i].y-point[begin].y)<25)
         point[i].group = level;
       else if ((token==0) && (!point[i].group))
                    {
```

```
                        next_begin = i;
                         token = 1;
                                }
                       }while (++i<data_size);
            level++;
            if(begin<next_begin) begin=next_begin;
            else break;             /* No more data */
            }while(begin!=data_size);
fclose(dFile);
ndFile = fopen(outputfile1,"w");
 if (ndFile==NULL)
  {
            printf("File open error. Can't create file");
    abort(1);
            }
pFile = fopen(outputfile2,"w");
 if (pFile==NULL)
  {
            printf("File open error. Can't create file");
    abort(1);
            }

for (j=1;j<level;j++)
            {
      num=0;
  xavg=yavg=zavg=0;

              for (i=0;i<data_size;i++)
              {
                if(point[i].group == j)
                        {
                         fprintf(ndFile,"%f %f %f %d\n",point[i].x,point[i].y,point[i].z,point[i].group);
        xavg=xavg+point[i].x;
        yavg=yavg+point[i].y;
        zavg=zavg+point[i].z;
                         num++;
                        }
              }
              fprintf(pFile,"%f %f %f     %d\n",xavg/num,yavg/num,zavg/num,num);
            }
fclose(ndFile);
fclose(pFile);
}

---------- #include <stdio.h>                    /* This
program put the data somewhat orderly, presently, for excel manual process */
#define inputfile "ecent.txt"           /* Input : disorder inputfile */
#define outputfile "ecenod.txt"          /* Output: outputfile        */
main()
{
 FILE *pFile,*oFile;
 int i,j,k,point_num=219;              /**** data_size should be changed for different file ****/
            struct Point{
                        float x;
                                    float y;
                                    float z;
                                    int num;
                                   } point[219],chess;                      /**** [*] should be changed for different file ****/

pFile = fopen(inputfile,"r");
 if (pFile==NULL)
  {
            printf("File open error.");
    abort(1);
            }
for (i=0;i<point_num;i++)
             {
     fscanf( pFile , "%f",&point[i].x );
     fscanf( pFile , "%f",&point[i].y );
     fscanf( pFile , "%f",&point[i].z );
     fscanf( pFile , "%d",&point[i].num );
```

```
      }
for (i=0;i<point_num;i++)
                {
                 chess.y = point[i].y;chess.x = point[i].x; chess.z = point[i].z; chess.num = point[i].num;
    for (j=i+1;j<point_num;j++)
                        if((point[j].y>chess.y+8)||(((point[j].y<(chess.y+8)) && ((point[j].y+4)>chess.y)) && (point[j].x<chess.x)))
                        {       /* Here is very hard to understand, you must analysis the data very detailly */
                                chess.y = point[j].y; chess.x = point[j].x; chess.z = point[j].z; chess.num = point[j].num;
                                point[j].y = point[i].y;point[j].x = point[i].x;point[j].z = point[i].z;point[j].num = point[i].num;
                                point[i].y = chess.y; point[i].x = chess.x; point[i].z = chess.z; point[i].num = chess.num;
                        }
                }
oFile = fopen(outputfile,"w");
 if (oFile==NULL)
  {
                printf("File create error!");
    abort(1);
                }
for (i=0;i<point_num;i++)
    fprintf(oFile, "%f %f %f %d\n",point[i].x,point[i].y,point[i].z,point[i].num);
fclose(pFile);
fclose(oFile);
}
```

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include <time.h>
/*
  Demonstration code to illustrate and check the solution
  to finding the center and radius of a sphere given 4 points
  on the sphere.
  A sphere center and radius are chosen randomly, then four
  random points on that sphere are generated, the original
  center and radius are then computed from the 4 points.
*/

#define TWOPI          6.28318530717958647692528
#define PI             3.14159265358979323846264
#define PID2           1.57079632679489661923132

typedef struct {
  double x,y,z;
} XYZ;
double Determinant(double **,int);

int main(int argc,char **argv)
{
  int i;
  double r,theta,phi;
  XYZ c,p[4];
  double **a=NULL,m11,m12,m13,m14,m15;

  /*r = 3;
  c.x = 2;
  c.y = 4;
  c.z = 5;
  fprintf(stderr,"Sphere: (%g,%g,%g), %g\n",c.x,c.y,c.z,r);*/

  /* Create 4 random points on the surface */
  /*for (i=0;i<4;i++) {
    theta = 0.43*i* TWOPI;
    phi =0.19* i * PI - PID2;
    p[i].x = c.x + r * cos(phi) * sin(theta);
    p[i].y = c.y + r * cos(phi) * cos(theta);
    p[i].z = c.z + r * sin(phi);
  }*/
  p[0].x= 14.554000;
  p[0].y= 144.076996 ;
  p[0].z= -1248.864990 ;
```

```
  p[1].x= 13.323000 ;
  p[1].y=144.067993 ;
  p[1].z= -1254.442017;
  p[2].x= 15.265000 ;
  p[2].y= 144.057999 ;
  p[2].z= -1254.355957;
  p[3].x= 14.616000 ;
  p[3].y= 146.632004;
  p[3].z= -1254.197021;

/*7998 14.616000 146.632004 -1254.197021 2
8636 14.609000 145.921997 -1253.659058 2
9270 13.973000 145.393005 -1254.689941 2
9271 14.601000 145.190002 -1252.927979 2
9904 13.962000 144.630005 -1253.688965 2
9905 14.554000 144.076996 -1248.864990 2
9906 15.256000 144.613998 -1253.548950 2
10537 13.323000 144.067993 -1254.442017 2
10538 13.937000 143.723007 -1251.416016 2
10539 14.577000 143.656998 -1250.843018 2
10540 15.265000 144.057999 -1254.355957 2
11172 13.973000 143.442993 -1254.651001 2
11173 14.619000 143.425995 -1254.496948 2
  */

  /* Malloc the array for the minor arrays */
  a = malloc(4*sizeof(double *));
  for (i=0;i<4;i++)
    a[i] = malloc(4*sizeof(double));

  /* Find determinant M11 */
  for (i=0;i<4;i++) {
    a[i][0] = p[i].x;
    a[i][1] = p[i].y;
    a[i][2] = p[i].z;
    a[i][3] = 1;
  }
  m11 = Determinant(a,4);

  /* Find determinant M12 */
  for (i=0;i<4;i++) {
    a[i][0] = p[i].x*p[i].x + p[i].y*p[i].y + p[i].z*p[i].z;
    a[i][1] = p[i].y;
    a[i][2] = p[i].z;
    a[i][3] = 1;
  }
  m12 = Determinant(a,4);

  /* Find determinant M13 */
  for (i=0;i<4;i++) {
    a[i][0] = p[i].x;
    a[i][1] = p[i].x*p[i].x + p[i].y*p[i].y + p[i].z*p[i].z;
    a[i][2] = p[i].z;
    a[i][3] = 1;
  }
  m13 = Determinant(a,4);

  /* Find determinant M14 */
  for (i=0;i<4;i++) {
    a[i][0] = p[i].x;
    a[i][1] = p[i].y;
    a[i][2] = p[i].x*p[i].x + p[i].y*p[i].y + p[i].z*p[i].z;
    a[i][3] = 1;
  }
  m14 = Determinant(a,4);

  /* Find determinant M15 */
  for (i=0;i<4;i++) {
    a[i][0] = p[i].x*p[i].x + p[i].y*p[i].y + p[i].z*p[i].z;
    a[i][1] = p[i].x;
    a[i][2] = p[i].y;
```

```
    a[i][3] = p[i].z;
  }
  m15 = Determinant(a,4);

  fprintf(stderr,"Determinants: %g %g %g %g %g\n",m11,m12,m13,m14,m15);
  if (m11 == 0) {
    fprintf(stderr,"The points don't define a sphere!\n");
    exit(-1);
  }

  c.x = 0.5 * m12 / m11;
  c.y = 0.5 * m13 / m11;
  c.z = 0.5 * m14 / m11;
  r = sqrt(c.x*c.x + c.y*c.y + c.z*c.z - m15/m11);
  fprintf(stderr,"Sphere: (%g,%g,%g), %g\n",c.x,c.y,c.z,r);
}
/*
  Recursive definition of determinate using expansion by minors.
*/
double Determinant(double **a,int n)
{
  int i,j,j1,j2;
  double det = 0;
  double **m = NULL;

  if (n < 1) { /* Error */

  } else if (n == 1) { /* Shouldn't get used */
    det = a[0][0];
  } else if (n == 2) {
    det = a[0][0] * a[1][1] - a[1][0] * a[0][1];
  } else {
    det = 0;
    for (j1=0;j1<n;j1++) {
      m = malloc((n-1)*sizeof(double *));
      for (i=0;i<n-1;i++)
        m[i] = malloc((n-1)*sizeof(double));
      for (i=1;i<n;i++) {
        j2 = 0;
        for (j=0;j<n;j++) {
          if (j == j1)
            continue;
          m[i-1][j2] = a[i][j];
          j2++;
        }
      }
      det += pow(-1.0,1.0+j1+1.0) * a[0][j1] * Determinant(m,n-1);
      free(m);
    }
  }
  return(det);
}
```