NISTIR 6457

NIST Construction Automation Program Report No. 4: Non-Intrusive Scanning Technology for Construction Status Determination

Building and Fire Research Laboratory
Gaithersburg, MD 20899

United States Department of Commerce
Technology Administration
National Institute of Standards and Technology

NISTIR 6457

NIST Construction Automation Program Report No. 4: Non-Intrusive Scanning Technology for Construction Status Determination

Geraldine S. Cheok
Robert R. Lipman
Christoph Witzgall
Javier Bernal
William C. Stone

January 2000
Building and Fire Research Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

U. S. Department of Commerce
William M. Daley, *Secretary*
Technology Administration
Dr. Cheryl L. Shavers, *Under Secretary of Commerce for Technology*
National Institute of Standards and Technology

Raymond G. Kammer, *Director*

CONTENTS

1.0 INTRODUCTION

Recent studies by the Construction Industry Institute[1] have indicated that for a typical $100 million construction project, between $500 000 and $1 million are spent just keeping track of where things are on the site -- typically tens of thousands of items -- and monitoring the status of construction activity. Additional expenses are directed to the establishment of the state of the infrastructure following the actual construction work. Approximately 2 % of all construction work must be devoted to manually intensive quality control and tracking of work package completion, including operations involving earthmoving and bulk materials handling. Any technology that can reduce this burden and decrease time to delivery will offer a significant competitive edge. It should be further emphasized that any technology that delivers automated, rapidly available information relating to project status and the position of components at the site would also leverage further cost savings by supplying that information to automated and semi-automated systems performing work at the site.

One of the more difficult things to track at a construction site is the geometry of construction materials which are not neatly classified as "components." The ability to capture such "amorphous" data becomes important if one is to achieve true automation. Amorphous data include such things as the state of excavation of terrain, the presence of raw materials (e.g., sand, gravel) depots; the location and extent of construction material piles; progress of a concrete casting; highway alignment; paving operations; etc. To obtain this information, the current state-of-practice is to conduct surveys. These surveys involve: 1) Field work and data acquisition - making measurements and recording data in the field 2) Computation and data processing - calculations based on recorded data to determine locations, areas, volumes, etc. 3) Mapping or data representation - plotting of measurements to produce a map, chart, or plat, and, for earth moving, the placement of grade stakes on the site -- a practice that even small construction companies recognize as inefficient.

Equally important is the need to automatically capture the "as-built" condition of an existing structure, or to capture and clarify a complex construction operation as it occurs and to provide real-time feedback to those conducting the operations. All of these are complex situations where traditional metrology techniques are not effective, due to the massive quantities of data needed to describe the environment. The research discussed herein focuses on the use of new fast laser ranging technologies and three-dimensional analysis to automatically and non-intrusively scan a construction site and to mine useful information from that data for project planning and documentation purposes.

The National Institute of Standards and Technology's (NIST) project in *Non-Intrusive Scanning Technologies for Construction Status Assessment* builds on metrology, wireless communications, and simulation research conducted as part of the National Automated Manufacturing

[1] An industry consortium comprising some 100 of the USA's largest contractors, AE design firms, and building owners. See, e.g., "1997 Strategic Plan & Governance Plan," **Construction Industry Institute**, Austin, TX, February 1997 and the in-progress report of Committee 151, on RFID Technologies.

Testbed (NAMT) collaboration [4]. The objectives of the project are to utilize new scanning technologies to improve critical construction status assessment needs by making these measurements faster and cheaper than traditional methods and to develop, in conjunction with industry, standard means for transmission and interpretation of such data.

Collaborations are being established with construction industry partners to ensure that (1) the technology developed is responsive to industry needs, and (2) the technology developed is usable in a construction context.

1.1 Scope and Objectives of Project

The research project was initiated in October 1998. It focuses on the development of an integrated software and wireless remote sensing system that will accept input from a variety of high speed automated ranging sensors and create a 3-D model of the present state of a portion of a construction site. A demonstration test using a pile of sand was given on June 24, 1999. Further details of the demonstration are described in Chapter 2. The next step, full-scale practical demonstration, site topography during the construction of the Building 205 Emission Control System at NIST will be tracked. This step will help identify problems that arise from conducting the scans in the field as opposed to the controlled environment in a laboratory.

Emphasis is on the use of non-intrusive scanning systems that can acquire site geometry, yet do not require instrumentation to be installed on earthmoving machinery. This is important for initial introduction of the technology to the construction industry as most contractors are reluctant to test new technologies and methods if it means an increase in their costs or delays and interruptions of the work progress. The models generated in this fashion (from the actual Building 205 construction site) will be compared against those acquired by means of an all-terrain vehicle equipped with a real-time, kinematic (RTK) global positioning and attitude determination system. Derivative quantities such as remaining cut-and-fill volumes, overall progress rate, and projected completion date will be displayed by means of a graphical user interface at the construction site office; ultimately, the same information will be able to be delivered directly to the operator of the appropriate earth moving machinery.

The work focuses on (1) scene update rate requirements and methods for improving scene update rates; (2) development of standardized means for transmitting and interpreting scan-data packets; and (3) development of practical post-processing routines that automatically operate on the 3-D data to produce useful derivative quantities identified in collaboration with the construction industry. This is a base-technology development project combined with practical, full-scale demonstrations to industry. Its various elements will be subsequently used as building blocks to address more complex 3-D as-built assessment problems for construction industry.

The excavation tracking technology developed during this project will be tested first at the Building 205 construction project on the NIST, Gaithersburg campus, scheduled to begin in late 1999. The procedures developed during this project will be extended to the much larger Advanced Measurement Laboratory construction project during the 2000-2002 time frame. Long-term research is being directed to the use of high speed, precision laser radars (LADARs) as an automated means of as-built discrete component identification and placement assessment.

To achieve these objectives, several tasks were identified:

1. Develop a laser ranging system that can image a construction area in "real-time".
2. Develop the ability to wirelessly transfer range data from the field to a remote office.
3. Link the rangefinder to positions using global positioning systems (GPS) and to attitude measurement systems so that the range data can be registered to a known reference frame.
4. Develop a user interface:
   a. To automatically operate the scanner.
   b. To display the 3-D data.
   c. To determine cut/fill requirements.

1.2 Report Outline

This report details the initial activities in the project and documents procedures used to acquire and display data obtained from a scanner and to perform volume calculations. Chapter 2 describes the laser scanner and the input parameters required to perform a scan, as well as the actual operation of the scanner during a demonstration project. Chapter 3 describes a program written in-house for visualizing and manually registering point cloud data from several scans, in their raw form or as filtered surfaces, and for calculating volumes. The program was developed using Data Explorer, an IBM product but now open source software. Chapter 4 describes a method and a FORTRAN 77 program used to represent the scan data by a triangulated mesh and provide a second source of volume calculations. Chapter 5 describes the process of tetrahedralization on which to base yet another method – also implemented in FORTRAN77 - for volume calculation, extending the capabilities of the method considered in Chapter 4. All three methods of volume calculation are demonstrated for scan data gathered during the June 24 demonstration. A summary and future research needs are given in Chapter 6. The source codes for the Data Explorer and FORTRAN programs are given in the attached disk.

2.0 SCANNING LASER

2.1 Introduction

Two principal categories of scene imagers are currently under consideration at research laboratories and in the commercial sector. The first bases its operation on measuring the time-of-flight of a coherent light pulse. The use of a pulsed diode laser will currently enable a range measurement to be made over several scores of meters in daylight to "non-cooperative" targets (i.e. natural targets without the use of retroreflectors) with a maximum measurement rate of around 25 kHz. In order to develop a scene image, this type of unit must be swept in raster mode to create a field of radially spaced rows or columns of range measurements. The unit output is typically in the form of scan lines, with each scan line progressively stepping across the scene within the field of view of the instrument.

A variant on the time-of-flight design is the "flash" LADAR or laser rangefinder (Sackos et al., 1996) in which a relatively high energy, diffuse light pulse "illuminates" the scene. The returned photons are then optically guided to a photomultiplier array which permits individual time-of-flight range measurements to be made simultaneously over a matrix representing the scene. In principle this approach offers the most direct path to "real-time" performance, yet it remains a research tool for two reasons. The “flash” LADARs that have been developed have limited range and/or field-of-view, and the most advanced publicly known flash unit employs a 32 x 32 pixel matrix chip.

The second category involves the use of a continuous wave (cw) laser interferometer that determines relative distances based on phase differences. This type of unit is typically operated in a raster scan mode, similar to the pulsed diode LADAR

An obvious shortcoming of all these rangefinders is that they cannot “see” through objects and thus scans from different locations need to be acquired and registered to obtain a composite, unobstructed view.

The majority of the rangefinders used in surveying employ lasers with wavelengths in the infrared region and are eyesafe. The maximum range of these rangefinders varies from a few meters to greater than 50 km with the penalty of reduced accuracy at the higher ranges.

The accuracy of the measurement is influenced by many factors with the main ones being the reflectivity of the object and the existing environmental conditions. Bright sunlight, rain, snow, fog, smoke, and dust will adversely affect the accuracy of the measurement. Accuracy may be increased by increasing the measuring time, the signal power, and by taking multiple readings.

2.2 Description

There are a few commercially available LADARs. The criteria used in selecting a scanning LADAR for this project were 1) speed in acquiring the range data, 2) accuracy, 3) maximum range, 4) angular resolution, 5) cost, and 6) commercial availability.

The objective of any site metrology system is to permit real-time machine control and obstacle avoidance. This requires an update rate of approximately 10 Hz, i.e., a full geometric site model being updated ten times per second so that there is no perceived lag time or delay between the display and live activities. None of the commercially available LADAR systems can achieve this rate. Thus, real-time machinery control based on LADAR range data is still not yet achievable. However, there is still a great deal that can currently be done, even with slow instruments.

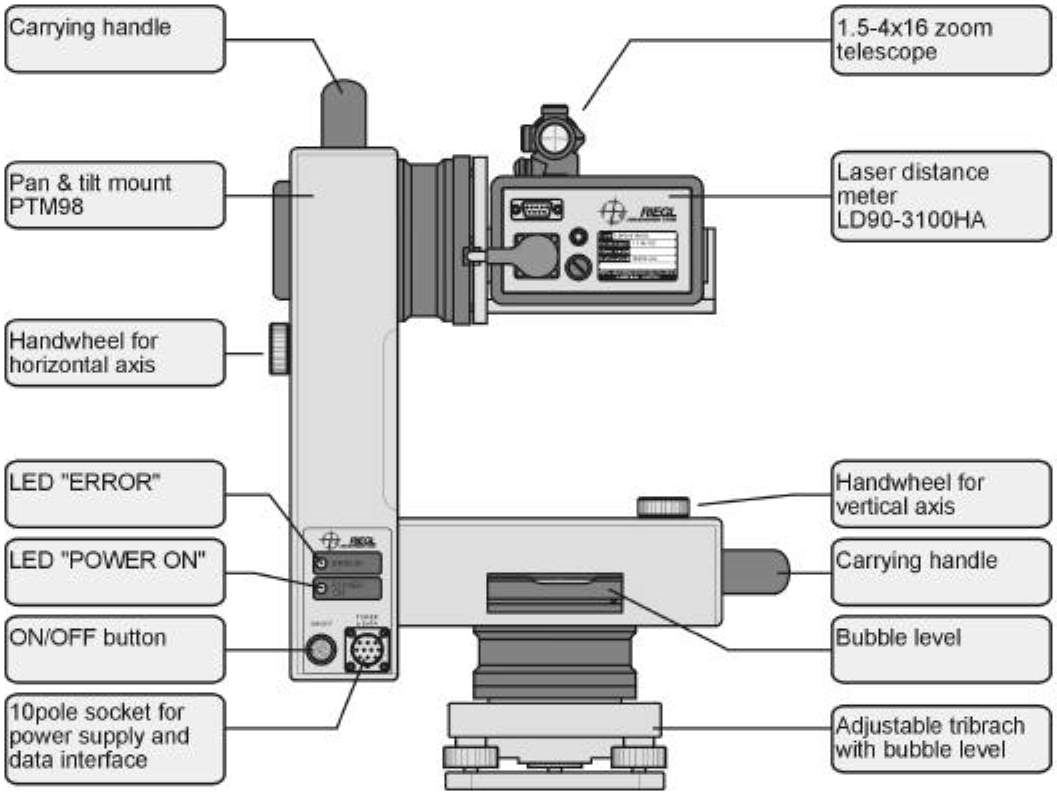
For purposes of obtaining an as-built model, positioning, alignment, etc., the required accuracy had to be on the order of millimeters to meet construction tolerance requirements and for accurate placement/positioning of elements. The minimum range required to scan a construction site is about 100 m to 200 m.

The project focus is on exploring the utility of non-intrusive scanning techniques, and the use of a slower model does not yet adversely affect the development of basic post-processing tools. Fundamental research efforts both within and outside NIST are underway to develop the next generation of LADAR which will seek to improve scene update rate, range, and accuracy.

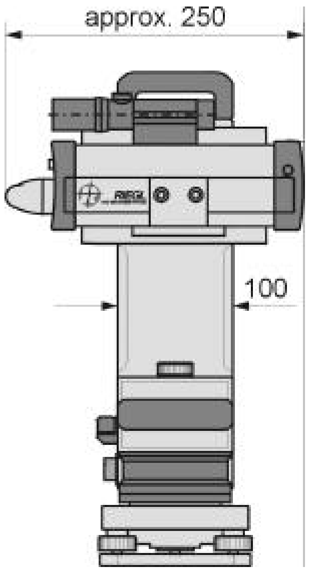
Based on the above discussion, a laser scanner manufactured by Riegl, LPM98-VHS, was selected. The laser profile measuring system consists of a high precision pan/ tilt mount, laser rangefinder, and laptop control computer. This system will henceforth be referred to as "the laser scanner."

The specifications for the laser scanner given by the manufacturer are as follows. The laser scanner is classified as a Class 1 (eyesafe) system that emits an infrared laser pulse with a wavelength of 905 nm ± 5 nm. The measurement rate is 1 000 Hz. The scanning field-of-view (FOV) is 360º horizontally and ±150° vertically which permits a large area to be imaged. Two stepper motors are used for positioning the laser scanner. The range of the laser scanner is up to 150 m for objects with reflection coefficients, ρ, greater than 80 % and 50 m for objects with ρ greater than 10 %. Distance measurements have a typical standard uncertainty of ±20 mm and ±50 mm in the worst case (due to dust and atmospheric effects). The resolution of the laser scanner is 2 cm. The beam size as it leaves the exit aperture is 42 mm by 25 mm. Beam divergence is 3 mrad by 3 mrad. The scan rate is 36°/s both in the horizontal and in the vertical directions. The accuracy of the angular positioning of the pan/tilt mount is ±0.009°. The voltage requirement is a minimum of 11 V DC to a maximum of 18 V DC and when both motors are in operation, the power requirement is 3.5 A at 12 V. The average power output is 1 mW with a pulse width of 17 ns.

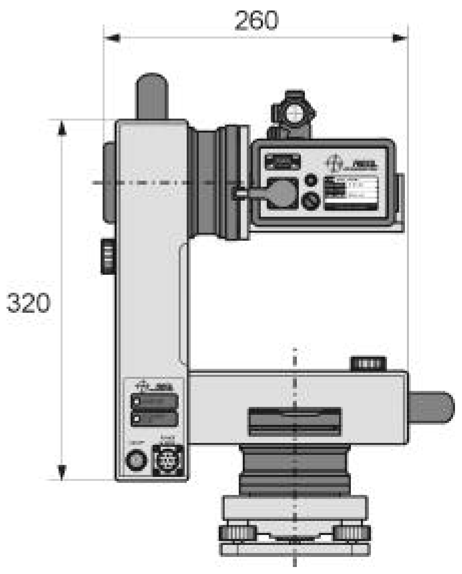
The main dimensions, L x W x H, of the laser scanner are (230 x 260 x 320) mm. It weighs about 8 kg. A schematic of the laser scanner is shown in Fig. 2.1.


Carrying handle
1.5-4x16 zoom telescope
Pan & tilt mount PTM98
Laser distance meter LD90-3100HA
Handwheel for horizontal axis
LED "ERROR"
Handwheel for vertical axis
LED "POWER ON"
Carrying handle
ON/OFF button
Bubble level
10pole socket for power supply and data interface
Adjustable tribrach with bubble level

(a) Various Elements of Laser Scanner



(b) Side View



(c) Back View

Figure 2.1. Schematic of Laser Scanner (taken from Riegl, 1999).

Communication with the laser scanner is through an RS232 serial interface. The communication settings are 115.2 kBaud, 1 start bit, 8 data bits, no parity, and 1 stop bit. The commands are sent to the laser scanner through a Microsoft© Windows-based program, LPMScan, or by direct communication with the laser scanner. The actual commands will be presented in Section 2.3.

2.3 Laser Setup and Scan Acquisition

2.3.1 GENERAL

A brief description of the scanner functions is given in this section. When the scanner is turned on, it moves to its initial zero position as shown in Fig. 2.2. Because of physical constraints, the area beneath the laser cannot be scanned and the vertical range of the scanner is 0° (0 gon) to approximately 150° (170 gon) and from approximately 200° (230 gon) to 360° (400 gon). The angular measurement unit of the laser scanner is gons where 400 gon equals 360° or 1 gon equals 0.9°.

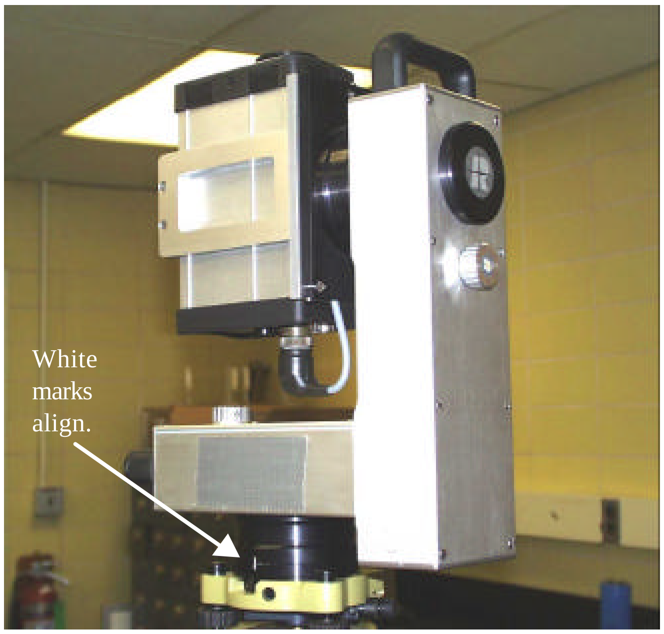


Figure 2.2. Scanner in its Initial Zero Position - Laser Points Straight up.

Scanning is accomplished with a series of vertical up and down movements followed by a counter-clockwise rotation to begin the next scan line. The sequence always begins with an up scan followed by a down scan as shown in Fig. 2.3.

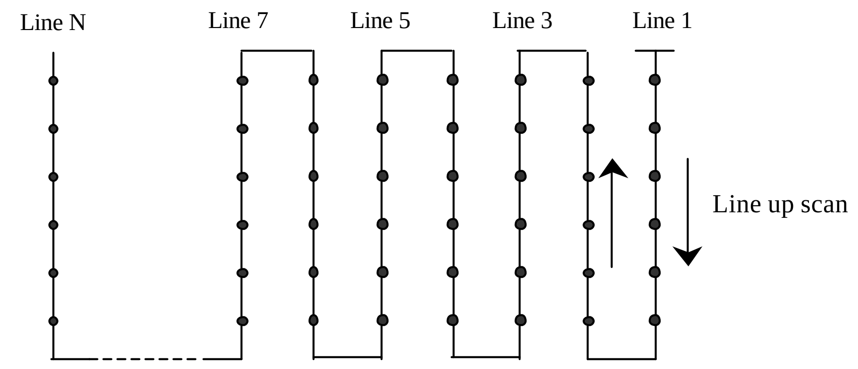
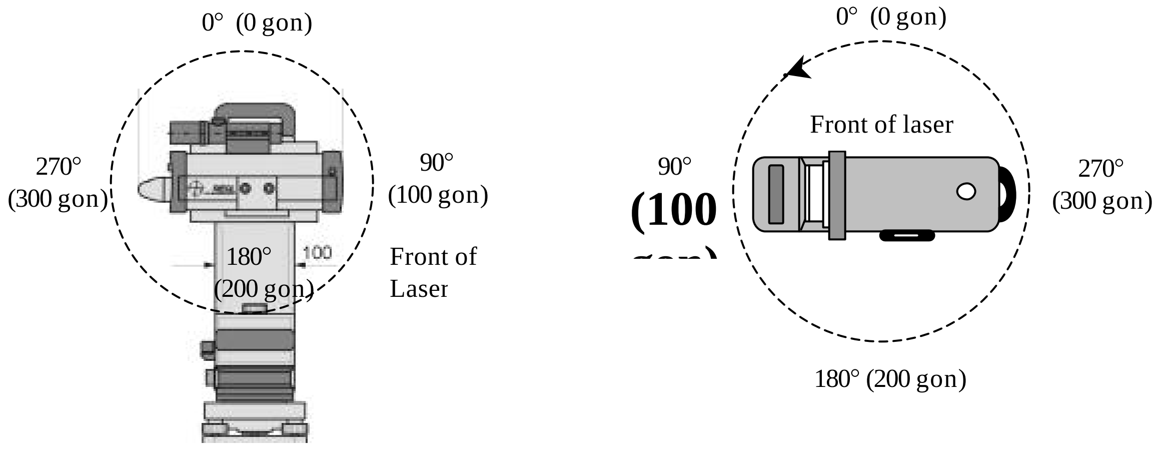


Figure 2.3. Scanning Sequence (Riegl, 1999).

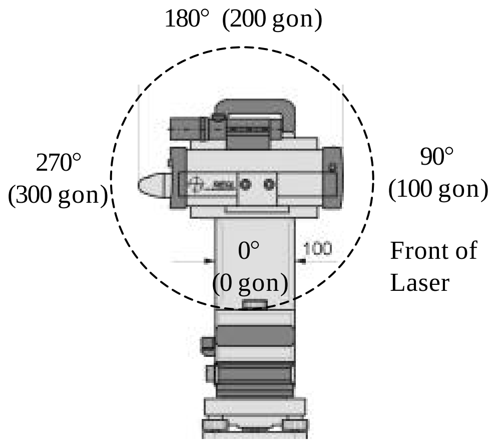
The scanner has two modes of operation: a positioning mode and a scanning mode. Prior to discussing the laser setup and data acquisition, the convention for rotating the scanner has to be first described. The rotation angle convention in scanning mode differs slightly than in positioning mode. The latter case is seldom used or needed as the specified angles for scanning are measured from the laser's initial zero position and not from its current position. The convention is shown in Fig. 2.4. The angles shown in Fig. 2.4b, positioning mode, indicate the angles to which the laser will move to when given an absolute move command.



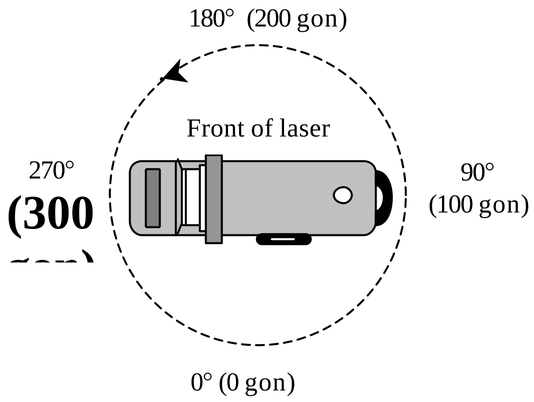
Vertical Angle

Horizontal Angle

a. Scanning Mode.



Vertical Angle



Horizontal Angle

b. Positioning Mode

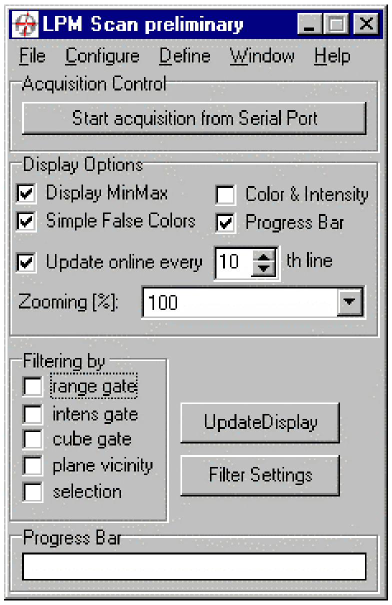
Figure 2.4. Scanner Rotation Convention.

2.3.2 LASER SETUP AND DATA ACQUISITION

To initiate the scan of an area, the basic input commands to the laser scanner are: *start horizontal angle*, *start vertical angle*, *incremental angle* in both horizontal and vertical directions, number of data points in a scan line, and the number of scan lines. As mentioned earlier, there are two ways to communicate with the laser scanner and these two methods are described in the following two sections.

2.3.2.1 LPMScan

The first method of communication with the laser is by using a Windows-based program called *LPMScan* developed by Riegl. To run the program, double click on the executable icon. The main window, LPM Scan Preliminary, appears as shown in Fig. 2.5.



✓ Display MinMax
☐ Color & Intensity
✓ Simple False Colors
✓ Progress Bar
✓ Update online every 10 th line
Zooming [%]: 100
☐ range gate
☐ intens gate
☐ cube gate
☐ plane vicinity
☐ selection

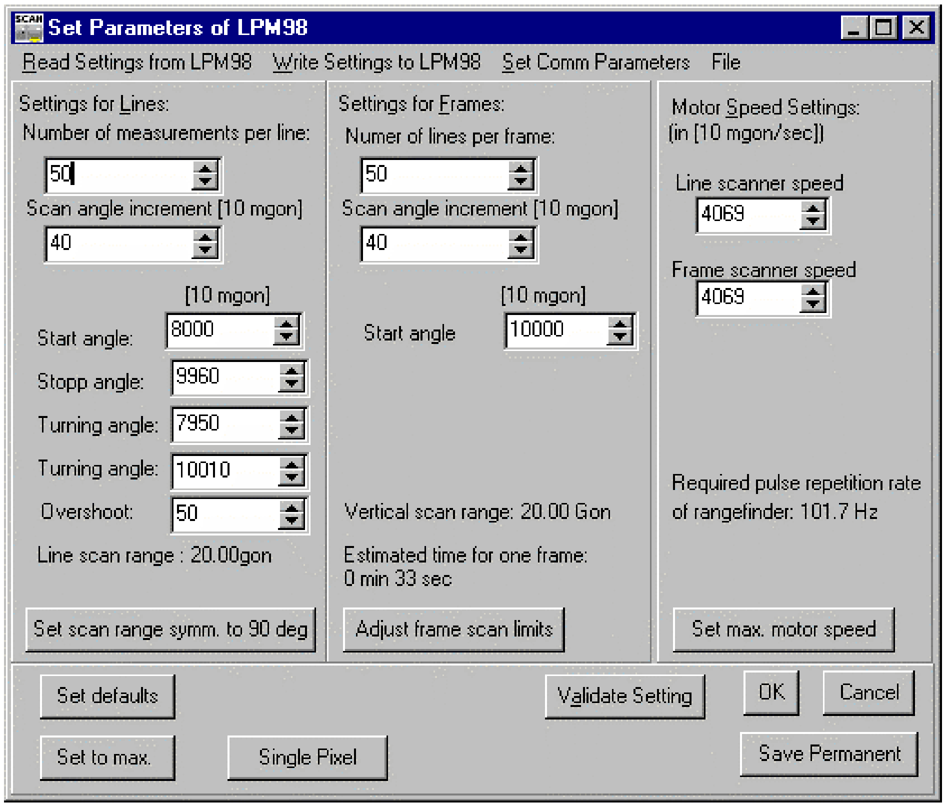
Figure 2.5. LPMScan Start-up Window.

The steps to set-up and acquire a scan are given in Table 2.1.

Table 2.1 Input to LPMScan to Acquire Scan.

Step	Procedure	Comment
1	Under Configure a. Select Data source b. Select Serial Port	Select how the communications will be achieved. In this case, communications is done through the serial port.
2	Under Configure a. Select ComPort, Set Parameters. b. A Com Port Options window appears. 1) In ComPorts box, select to Direct to Com 1 2) In Baud Rates box, select 115200. 3) In Parity box, select None 4) In Data Bits box, select "8" 5) In Stop Bits box, select "1" Leave the other parameters as default values.	Define communication parameters.

3	Under Define, select Scanner Parameters. The Set Parameters of LPM98 windows (Fig. 2.6) will come up. **NOTE:** The values entered for angles and angle increments are in gons and are <u>100 times</u> the actual values - this was done so that only integers are used. In Scanner Parameters window, 1. In Settings for Lines box: a. Enter number of measurements per line. b. Enter vertical scan angle increment - 0.36° (40 gon) is minimum increment. c. Enter start vertical angle. 2. In Settings for Frames box: a. Enter number of lines per frame. b. Enter horizontal scan angle increment - 0.36° (40 gon) is minimum increment. c. Enter start horizontal angle. 3. Click OK button. The rest of the parameters are automatically updated or are left as the default values. HDELAY and VDELAY (explained in Section 2.3.2.2) are internally set to 160.	Define scan parameters.
4	In the main window, click on the Start Acquisition from Serial Port button.	Start scan.
5	Under File, select Export 3-D Data as 1. Under Save as type, select ASCII text file from the pull down menu. The first few lines of a sample scan is shown in Fig. 2.7.	Saving data. The data can be saved as a/an VRML 2.0 file, ASCII text file, DXF file, 3DD file, or vtk file. In this case, ASCII text format is selected.



Settings for Lines:
Number of measurements per line: 50
Scan angle increment [10 mgon]: 40
[10 mgon]
Start angle: 8000
Stopp angle: 9960
Turning angle: 7950
Turning angle: 10010
Overshoot: 50
Line scan range : 20.00gon

Settings for Frames:
Numer of lines per frame: 50
Scan angle increment [10 mgon]: 40
[10 mgon]
Start angle: 10000
Vertical scan range: 20.00 Gon
Estimated time for one frame:
0 min 33 sec

Motor Speed Settings:
(in [10 mgon/sec])
Line scanner speed: 4069
Frame scanner speed: 4069
Required pulse repetition rate of rangefinder: 101.7 Hz

Figure 2.6. Scanner Parameters Window.

```
#ASCII Export File
#Automatically generated from - RLMSSCAN
#  coordinates x y z [a] in units of meter
#
6.635 -1.739 13.113 74
6.632 -1.832 13.108 74
6.629 -1.925 13.102 72
6.639 -2.023 13.127 72
6.625 -2.112 13.098 70
6.576 -2.190 12.996 76
6.408 -2.225 12.666 52
```

Figure 2.7. Sample ASCII Data File.

The frame of reference for the data values shown in Fig. 2.7 is as follows: the positive x-axis points in front of the laser, the positive y-axis points to the left of the laser, and the positive z-axis points above the laser. The origin of the reference frame is the intersection of the two rotation axes of the laser scanner. The last column of data contains the intensity of the returned signal and the data range from 0 to 255.

After the scan is completed, the scanned area is displayed in two windows: 1) false image of the scene in color with respect to range and 2) false image in grayscale with respect to intensity.

2.3.2.2   Direct Communication

The second method of communicating with the laser using a PC is direct serial communication using terminal emulation software. Table 2.2 lists the steps needed to conduct a scan. Commands may be typed in lowercase or uppercase. Once the command is entered, the laser echoes the command with an asterisk at the beginning. In Table 2.2, a lowercase alphabet represents an integer value defined by the user. Similar to the LPMScan program, angles and incremental angles are entered in units of gons and are multiplied by 100.

Table 2.2. Procedures to Obtain a Scan Using Direct Serial Communication.

Step	Procedure	Comment
1	ctrl-P	Begin programming mode. The reply is an asterisk (*).
2	RF_NUMBER_L=a	Enter the number of data points per line
3	RF_DELTA_L=b	Enter the vertical angle increment.
4	RF_START_L[0]=c	Enter the starting value for the vertical angle.

5	SC_START_L[0]=d	Enter the scanner start position for the up line scan. The recommended value is 0.45° to 0.9° (0.5 gon to 1 gon) less than RF_START_L[0].
		d = RF_START_L[0] - 50 = c - 50
		or
		d = RF_START_L[0] - 100 = c - 100

6	RF_START_L[1]=e	Enter starting point of down scan. e = RF_START_L[0] + (RF_NUMBER_L - 1) RF_DELTA_L = c + (a-1) b

7	SC_START_L[1]=f	Enter the scanner start position for the down line scan. The recommended value is 0.45° to 0.9° (0.5 gon to 1 gon) less than SC_START_L[0].
		f = SC_START_L[0] - 50 = d - 50
		or
		f = SC_START_L[0] - 100 = d - 100

8	SC_START_F = g	Enter number of lines per scan.
9	SC_DELTA_F = h	Enter the horizontal angle increment.
10	SC_START_F = j	Enter starting value for the horizontal angle ± 4.5° (5 gon). j = Desired start angle - 50
11	HDELAY = 160	Enter horizontal delay between motor steps. Enter 160 for maximum scanning rate.
12	VDELAY = 160	Enter vertical delay between motor steps. Enter 160 for maximum scanning rate.
13	Q	Quit programming mode

14	Ctrl-x	Start scanning

The commands to obtain a 100 x 200 scan, 100 measurements per line by 200 lines, are listed in Table 2.3. The start horizontal angle is 0° (0 gon) and the start vertical angle is 45° (50 gon). The scan overshoot is 0.9° (1 gon) and the angle increment of 0.09° (0.1 gon) in both the horizontal and vertical directions.

Table 2.3. Commands for Direct Communication with Laser.

Command	Reply	Comment
Ctrl-P	*	Start programming mode
RF_NUMBER_L=100	*RF_NUMBER_L=100	No. of points per line
RF_DELTA_L=10	*RF_DELTA_L=10	Vertical angle increment
RF_START_L[0]=10000	*RF_START_L[0]=10000	Start vertical angle
SC_START_L[0]=9900	*SC_START_L[0]=9900	
RF_START_L[1]=10990	*RF_START_L[1]=10990	
SC_START_L[1]=11000	*SC_START_L[1]=11000	
SC_NUMBER_F=200	*SC_NUMBER_F=200	No. of lines per frame
SC_DELTA_F=10	*SC_DELTA_F=10	Horiz. angle increment
SC_START_F=39950	*SC_START_F=39950	Start horizontal angle
HDELAY=160	*HDELAY=160	
VDELAY=160	*VDELAY=160	
Q	*Q b399.99;i000.01	Quit programming mode. The current angular location of the laser is echoed.

When using the direct communication method, the user will have to be able to capture the binary data that is sent back by the laser and to convert the data to another format such as ASCII. The format of the binary data can be found in Riegl (1999).

2.4 Sample Scans

2.4.1 Single Scan to Create 2-1/2-D image

A sample scan of the NIST Tri-Directional Test Facility was obtained and is displayed in the form of a point cloud in Fig. 2.8. The 2-1/2-D image consists of approximately 40 000 data points - 200 points per scan line and 200 scan lines. The actual number of data points is somewhat less than 40 000 because in some cases no signal was returned – typically, diffuse black objects make poor targets. The horizontal and vertical angles were set to 90° (100 gon) and angle increments were set to 0.45° (0.5

gon). The scan acquisition time was approximately 7 min. The range data contained in this very coarse image are still sufficient to easily pick out many components including ladders, access doors, scaffolding, test fixtures, and an overhead crane. Dense data, registered with similar data taken from other points of view, will provide the raw information upon which component identification algorithms operate.

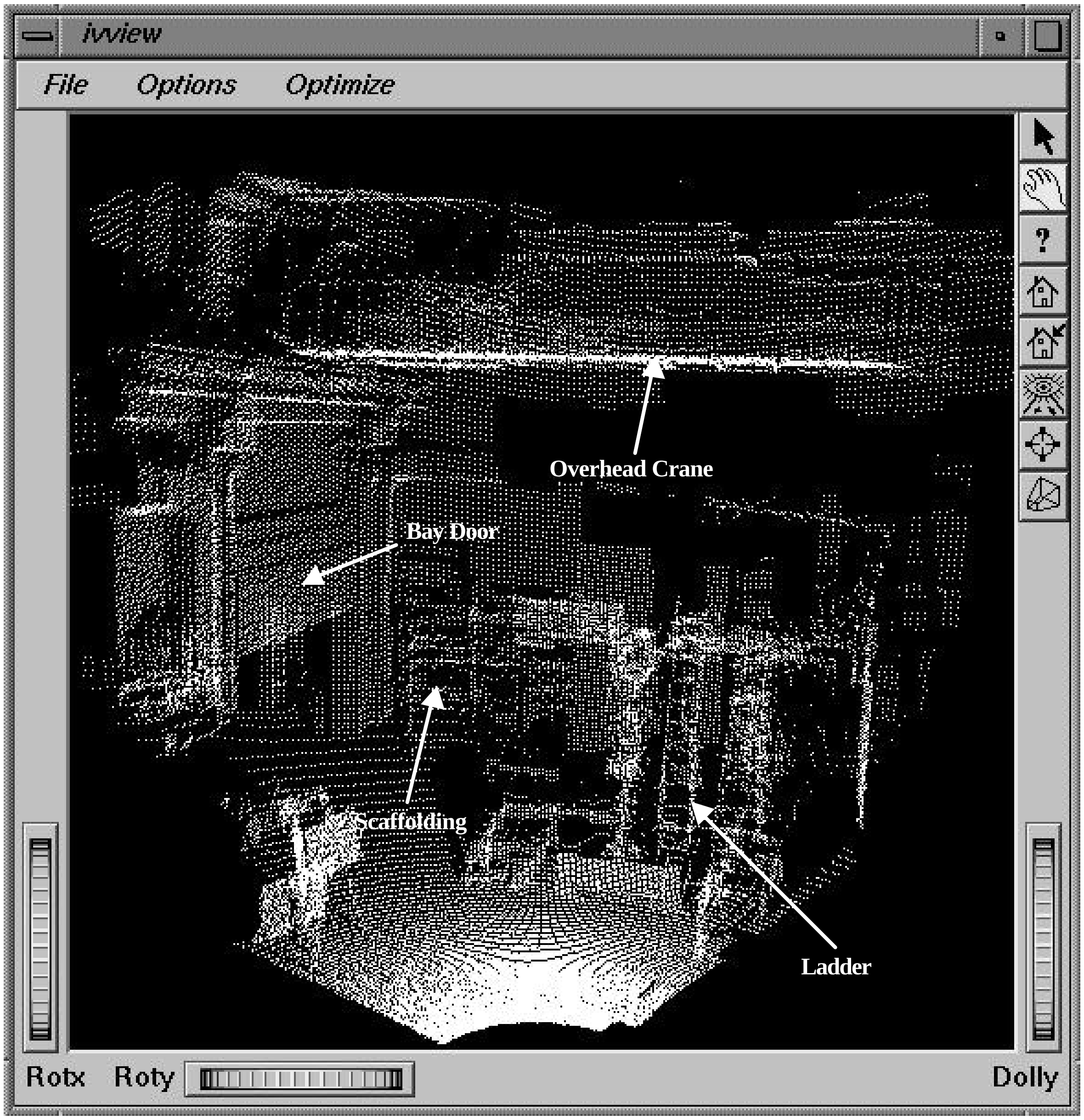


Figure 2.8. Point Cloud of NIST TTF - 40 000 data points.

2.4.2 Multi-Scans to Create 3-D Images

A live demonstration, given on June 24, 1999 at NIST, was developed to illustrate the viability of using a laser scanner for terrain status assessment and the ability to wirelessly transfer data from one location to another to achieve real time progress updates at a construction site. In addition, the demonstration was used to determine the necessary post-processing tools needed to display and calculate the cut/fill requirements.

For purposes of the demonstration, a small sand pile was used to simulate terrain. The sand pile was located in NIST Large-Scale Testing Laboratory (Building 202) representing the construction site and the audience was located in one of the auditoriums in the Administration Building (Building 101), representing the offsite engineering office.

Before the demonstration, a graphical representation of the sand pile (initial state of terrain) was created by combining the point clouds obtained from four locations around the sand pile. Figure 2.9 shows the sand pile and the four point clouds obtained by the laser scanner from around the sand pile. LPMScan was used to control the laser scanner, and the data is visualized using *demoscan*, a Data Explorer program described in Chapter 3.

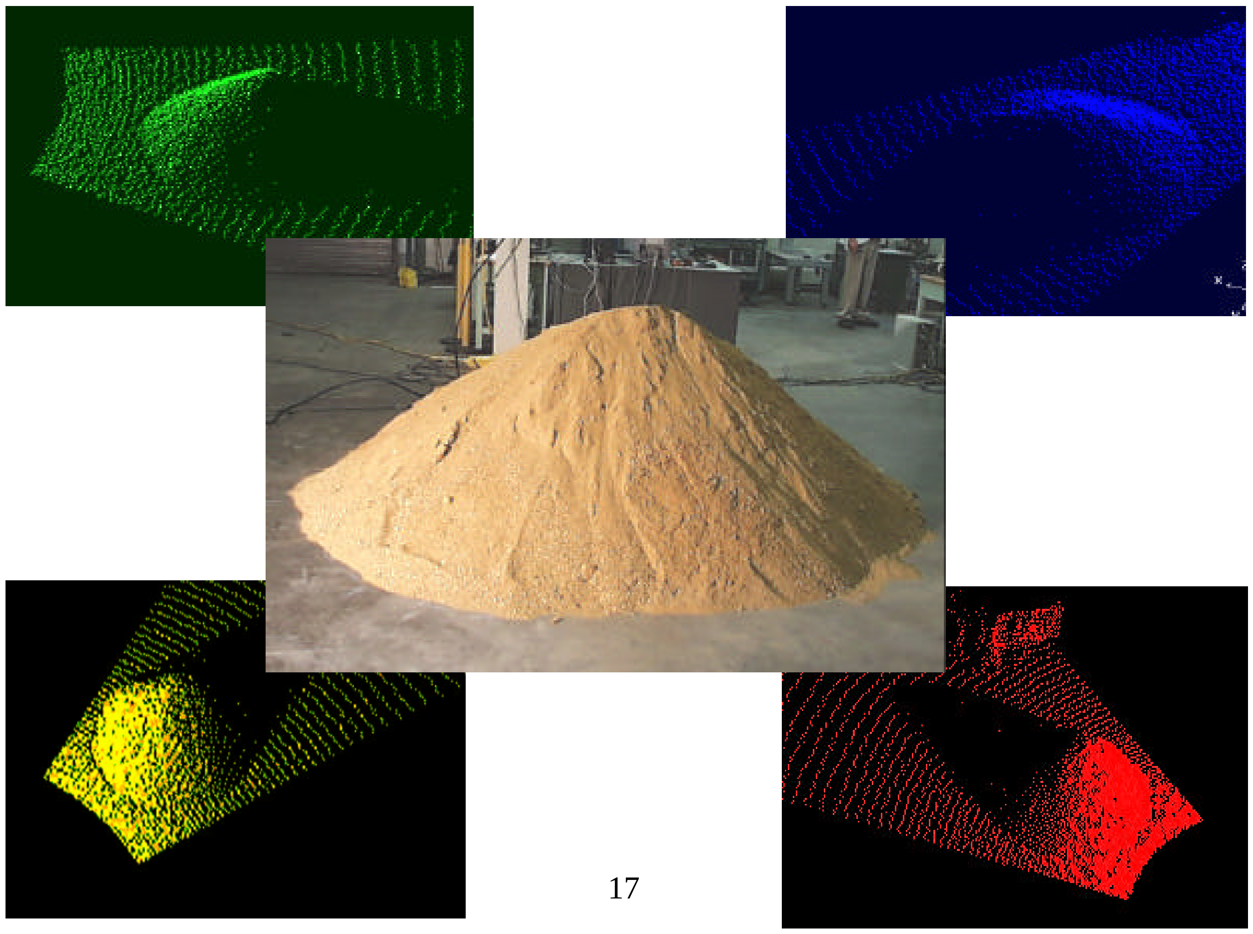
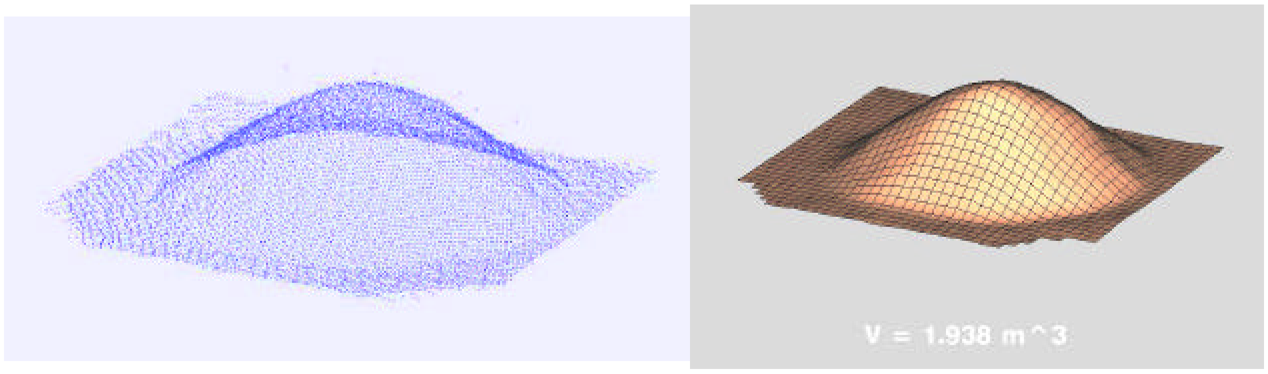


Figure 2.9. Sand Pile and 4 Point Clouds from Around the Sand Pile.

By obtaining views from different locations, a true 3-D representation of the sand pile is possible as any occluded features from one location are eliminated by the data obtained from the other locations. Prior to combining the point clouds, the data from the four locations has to be registered to a common reference frame. For the demonstration, one of the four points, Point C, was selected as the reference point (x, y, z = 0, 0, 0) and all the other points were referenced to this point. The points were previously surveyed using a total station and their relative coordinates were therefore known. The laser scanner was placed above each of the points and the angles to the two adjacent points were measured using the angle readout from the laser and the scope on the laser to sight to the adjacent points. In the field, GPS equipment will be used to determine the location and attitude of the laser scanner. Algorithms will have to be developed to convert the scanned positions to a common frame of reference using the information obtained from the GPS equipment. The coordinate transformations may either be performed on the laptop computer in the field or may be performed on the offsite computer that receives the data from the field. For this demonstration, all transformations (translations and rotations) were done semi-manually.

Even though the locations of the points were known to within ±2 mm^2, final registration of the point clouds was still required and was manually performed by visual means using an interactive graphics program (*demoscan*). The need for further manual registration was due mainly to the inaccuracies associated with the angular measurements. These measurements were obtained using the scope on the laser to sight from one point to each of the other two adjacent points. The sources of measurement errors result from the scanner not being perfectly level - when obtaining the scan data and when reading the angular measurement, the line-of-sight of the scope not being exactly parallel to the laser, human limitations, and from errors in the laser's ability to measure angular moves.

Figure 2.10 shows the point cloud after registration of the four scans and the 3-D surface of the sand pile. The procedures describing how the surface was created and the volume calculations are described in detail in Chapters 3 and 4.



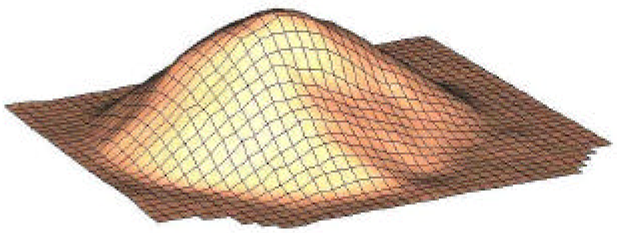
a. Combined Point Cloud from 4 Scans b. 3-D Surface

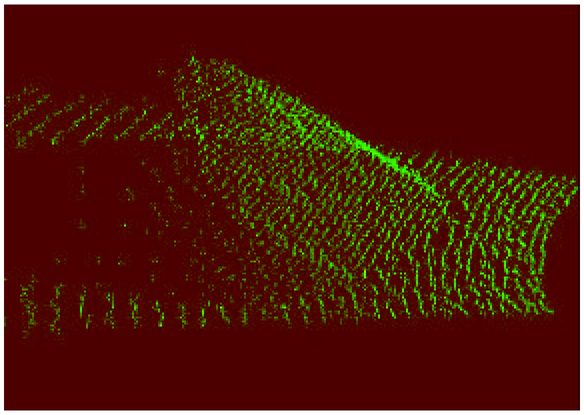
Figure 2.10. 3-D Surface of Sand Pile.

To simulate a change in the terrain, some sand was removed from the sand pile and the volume removed was automatically calculated as shown in Fig. 2.11. The sand pile was re-scanned from only one location, in the interest of time, as the demonstration was live. In this case, the last point from which a scan was obtained was Point C, the reference point, and the scanner was left at this location. By selecting the reference point for the second scan, no transformations of the data were required. If all changes to the sand pile cannot be captured from one scan location, the sand pile would have to be rescanned from two or more locations and the point clouds would have to be registered.

a. Sand removal.

b. "Changed Terrain.





V = 1.770: removed 0.168 (m^3)

c. Rescan from one location, Point C. d. Combined new scan with the three previous scans to regenerate the 3-D face.

Figure 2.11 Sand Removal Simulating "Changed" Terrain.

Once the data was acquired, the data file was sent via file transfer protocol (FTP) to the National Advanced Manufacturing Testbed (NAMT) computer lab where the volume calculations were performed and the new surface was regenerated and displayed. The data transmission, volume calculation, and surface regeneration were accomplished in a matter of seconds. The volume of sand removed was estimated at 0.15 $m^3 \pm 0.1$ m^3[3]. The cut volume may be computed using the Data Explorer script as described in Chapter 3 or using the Fortran program described in Chapter 4. The computed cut volume using both procedures was 0.168 m^3. Major sources of difference between the estimated value and the computed values are compaction of the sand during removal from the sand pile and during placement of the sand into buckets. The estimated volume of sand removed was obtained by shoveling the sand into a bucket and calculating the volume of the bucket. Compaction of the sand would result in a lower estimated value of sand removed.

The data transmission from the laptop was achieved by using a RangeLaAN2 card in the laptop and a Proxim base station to establish a wireless Ethernet connection to the subnet. The transmission of the field data, volume calculations, and scene update will have to be fully automated and will be further developed.

[3] Type B evaluation of the standard uncertainty. Fill level of the buckets was within ± 13 mm of the top of the bucket.

The sand pile demonstration showed that the laser scanner is a potentially useful tool in providing real-time construction updates. The next step is to develop a mobile platform for the laser scanner. This will be accomplished by mounting the laser scanner on a heavily instrumented all-terrain vehicle (ATV) which automatically registers the scanner data to the job site using precision GPS position and attitude reporting and a geometry transformation to adjust for instrumentation that are not co-located. Registration of the vehicle-based data will involve more elaborate calculations, owing to the additional sources of error from the vehicle position and attitude sensors. The advantage of this approach is significant as it opens the route to fully automated terrain status sensing at a job site using small, unmanned vehicles.

2.5 References

Riegl (1999), LPM98-VHS Laser Profile Measuring System 98 - User's Manual, Austria, 58 pp.

Sackos, J. T, Bradley, B. D., Diegert, C. F., Ma, P. W., Gary, C. (1996), "Scannerless Terrain Mapper," *Proceedings of the SPIE – The International Society for Optical Engineering Space Sciencecraft Control and Tracking in the New Millennium*, Aug. 6, Vol. 2810, pp. 144-153.

3.0 DATA VISUALIZATION

3.1 Introduction

The software package IBM Data Explorer was used to visualize the surface of the objects that were scanned by the laser scanner. Data Explorer (DX) is a general-purpose visual programming environment[4]. A visual program is created in DX by connecting various functional modules in a dataflow network. The modules include a wide variety of input, output, data manipulation, and graphic display functions. User interface widgets are also available to provide interaction with the dataflow network.

Laser scan data from two objects were obtained, a sand pile and a group of boxes. After the objects are scanned, they are changed by either removing sand or a box and rescanned once. The surfaces of the objects are visualized from the initial multiple scans from different locations and recomputed from the single updated scan. One general-purpose DX visual program was developed to handle scan data from either object. The visual program reads in the laser scanner data, registers (manually) the data, filters the data for noise, maps a regular 2D grid to the data, displays the surface defined by the grid, and computes the volume under the surface.

3.2 DX Visual Program

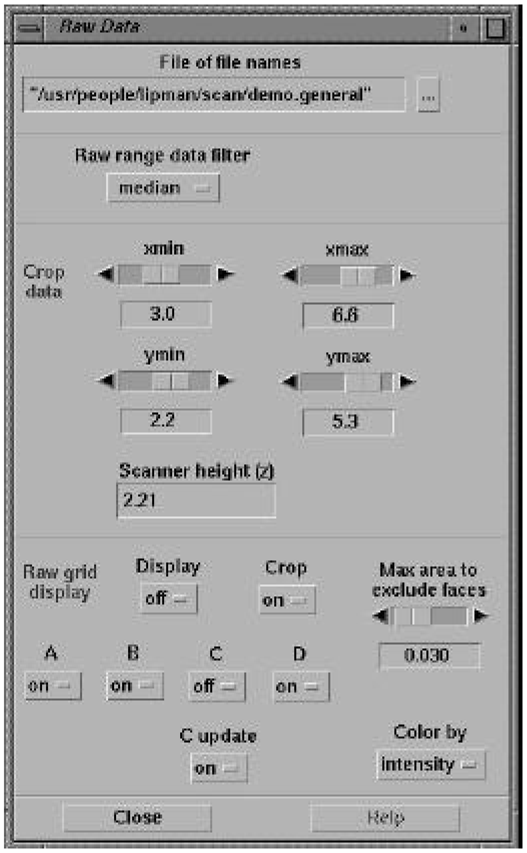
The DX visual program that was generated to visualize the two objects is called *demoscan*. A complete listing of the *demoscan* DX visual program and associated macros is found in Appendix A and is given in the included disk. The visual program is included for experienced DX users, although it will be referred to in the detailed description of the visualization process.

The *demoscan* DX visual program is contained in two files, demoscan.net and demoscan.cfg. There are also eleven DX macro files that are used with *demoscan*: construct.net, rangefilter.net, raw3dgrid.net, register.net, sum.net, surfdraw.net, switchxyz.net, vec2to3.net, xminmax.net, xycrop.net, and yminmax.net. To ensure that *demoscan* can find the macro files, the environment variable DXMACROS must point to the directory containing the macros.

When DX is started up with the *demoscan* visual program, the seven windows of the graphical user interface (GUI) will appear as shown in Fig. 3.1. Table 3.1 lists the sequence of steps necessary to visualize the laser scan data. A brief description of the function of each step is given along with a reference to the related GUI windows in Fig. 3.1, and the DX visual program listing in Appendix A.

[4] IBM Visualization Data Explorer User's Guide, (http://www.research.ibm.com/dx/, http://www.opendx.org/)

The resulting visualization is displayed by *demoscan* in three windows: *Scanned Object*, *Raw Grid*, and *Updated Scanned Object* as shown in Figs. 3.2, 3.3, and 3.4. A detailed explanation for some of


Raw Data

File of file names: "/usr/people/lipman/scan/demo.general"

Raw range data filter: median

Crop data: xmin 3.0, xmax 6.6, ymin 2.2, ymax 5.3

Scanner height (z): 2.21

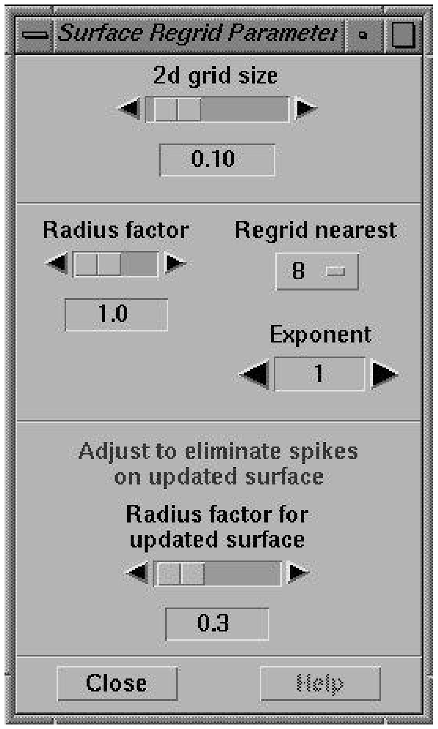
Raw grid display: Display off, Crop on

Max area to exclude faces: 0.030

A on, B on, C off, D on

C update: on

Color by: intensity


Surface Regrid Parameter

2d grid size: 0.10

Radius factor: 1.0

Regrid nearest: 8

Exponent: 1

Adjust to eliminate spikes on updated surface

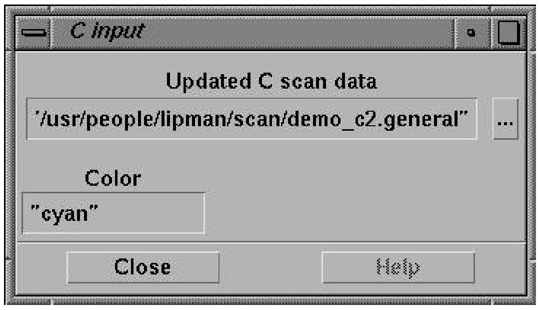
Radius factor for updated surface: 0.3

the steps is given in the following sections.

a. Raw Data Window b. Surface Regrid Parameters Window

c. Surface Appearance Window

Figure 3.1 GUI Windows of the DX *demoscan* Visual Program.


C input

Updated C scan data

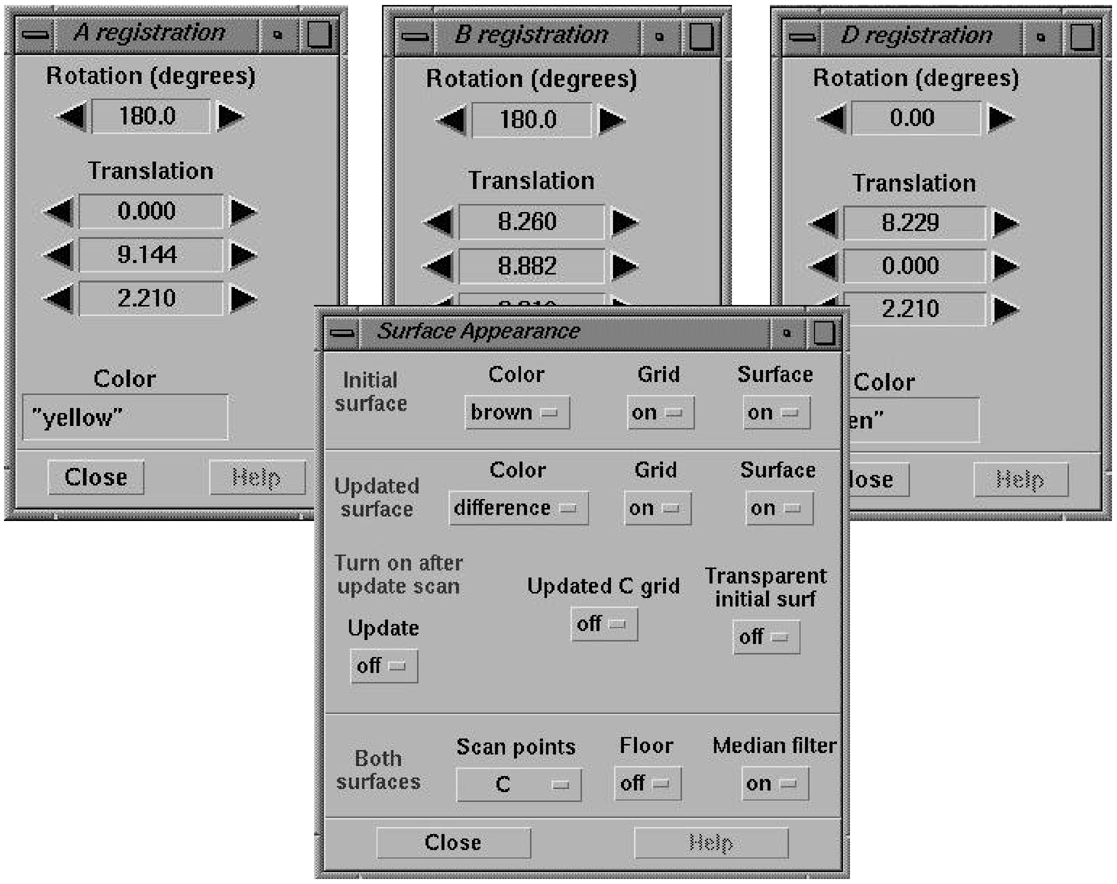
'/usr/people/lipman/scan/demo_c2.general"

Color

"cyan"

Close Help

d. Select Data File for Updated Scan (C).


A registration

Rotation (degrees) 180.0

Translation 0.000 9.144 2.210

Color "yellow"

B registration

Rotation (degrees) 180.0

Translation 8.260 8.882

D registration

Rotation (degrees) 0.00

Translation 8.229 0.000 2.210

Color

Surface Appearance

	Color	Grid	Surface
Initial surface	brown	on	on
Updated surface	difference	on	on

Turn on after update scan

Update: off

Updated C grid: off

Transparent initial surf: off

Both surfaces: Scan points: C; Floor: off; Median filter: on

Close Help

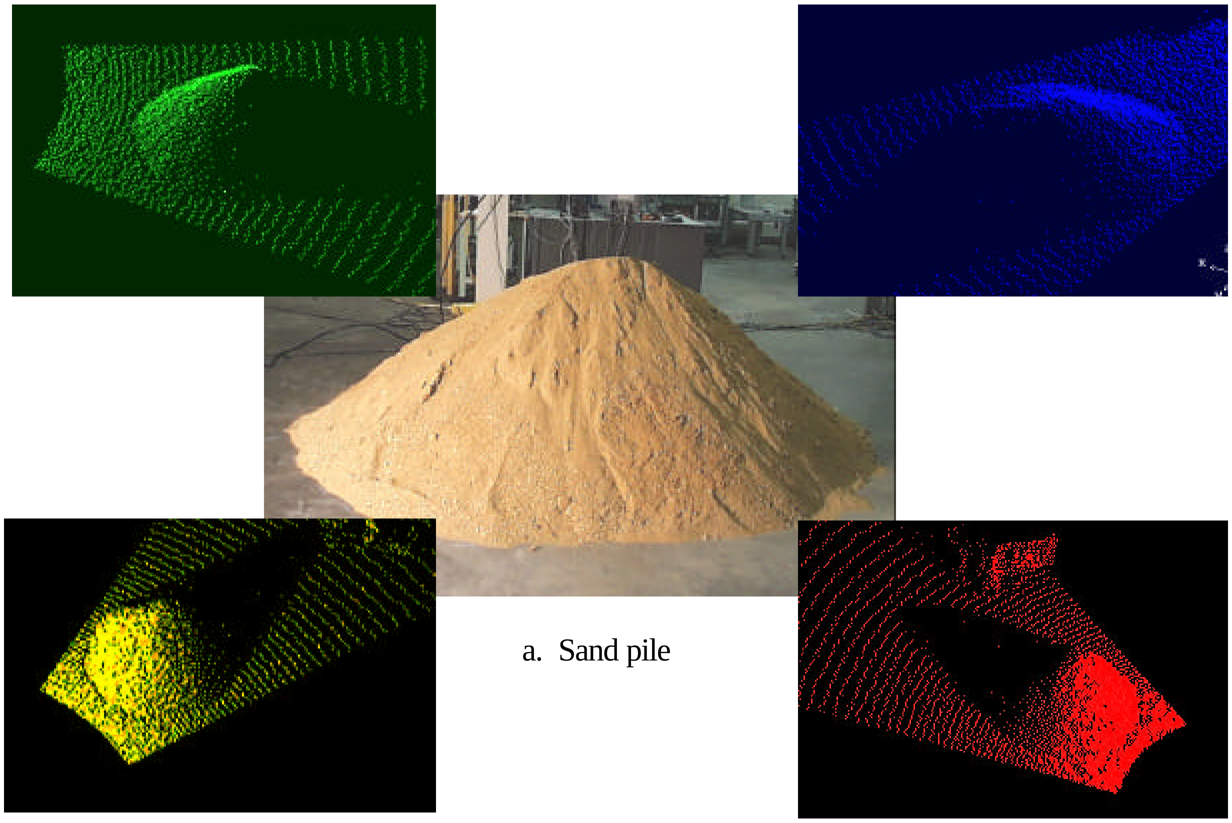
e. Parameter Windows for Interactive Registration of A,B,D Datasets.

Figure 3.1. GUI Windows of the DX *demoscan* Visual Program. (cont.)

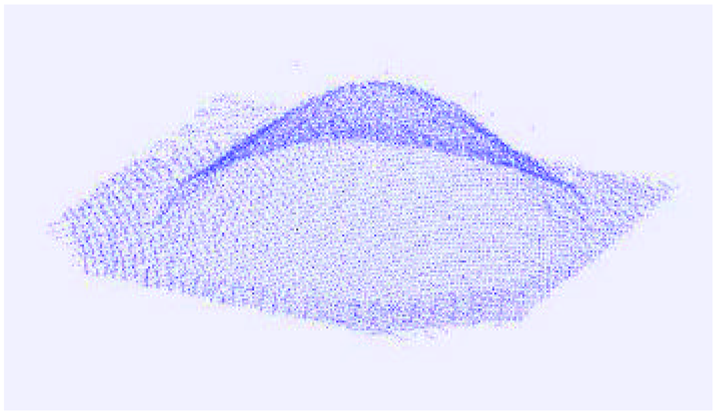
Table 3.1. DX Visualization Procedure and Description for *demoscan*.

Step	Brief Description	GUI Windows in Fig. 3.1	5. Corresponding DX Visual Program Pages in Appendix A
Data input	The files of XYZ data from the four initial laser scans for locations A, B, C, and D are read in. The XYZ data is also converted from the scanner coordinate system to the global coordinate system by switching the order of the Y and Z coordinates and the sign of the X coordinate.	Raw Data (File of file names) – Fig. 3.1a	filenames a_input, b_input c_input, d_input SwitchXYZ
Filter data	The range data is smoothed by optionally applying a Gaussian or median filter. The median filter is recommended because it preserves sharp discontinuities while eliminating noisy data.	Raw Data (Raw range data filter) – Fig. 3.1a	RangeFilter a_input, b_input c_input, d_input
Register data	The three laser scan datasets (A, B, D) are registered (aligned) to the reference dataset (C). The datasets are registered by applying a rotation and translation whose initial values are determined from the known location of the three datasets relative to the reference dataset.	A registration; B registration; D registration – Fig 3.1e; Raw Data (Scanner height) – Fig. 3.1a	Register a_input, b_input c_input, d_input
Point cloud display	To visually and interactively register the datasets, the XYZ scan data can be displayed as a point cloud or as a “raw grid”. Any or all of the four datasets can be visualized as a points cloud, each with its own color. The point cloud display comes up automatically in the *Scanned Object* window (Fig. 3.2a). Optionally, a “raw grid” can be drawn through the point cloud data to create a surface. The “raw grid” is useful to see the effects of filtering the range data and for detecting anomalies in the data. The “raw grid” is displayed in the *Raw Grid* window (Fig. 3.3).	Surface Appearance (Both surfaces, Scan points) – Fig. 3.1c; Raw Data (Raw grid display) – Fig. 3.1a	Raw3dGrid disp_rawgrid pointcloud a_input, b_input c_input, d_input
Crop data	The individual point cloud datasets are combined into one dataset and cropped to the area of interest before mapping a surface to it. Cropping eliminates unnecessary points in the foreground and background. The area of interest is set by minimum and maximum values of X and Y (Fig. 3.2b).	Raw Data (Crop data) – Fig. 3.1a	combined2d crop2d Xminmax Yminmax XYcrop
Generate 2D grid	A regular uniform 2D grid is generated in the area of interest of the combined and cropped point cloud data. The 2D grid is used to map the point cloud data to a surface. The size of the 2D grid is determined by the X and Y extents of the cropped point cloud data and the size of an individual grid.	Raw Data (Crop data) – Fig. 3.1a Surface Regrid Parameters (2d grid size) – Fig. 3.1b	surface3d Construct2d

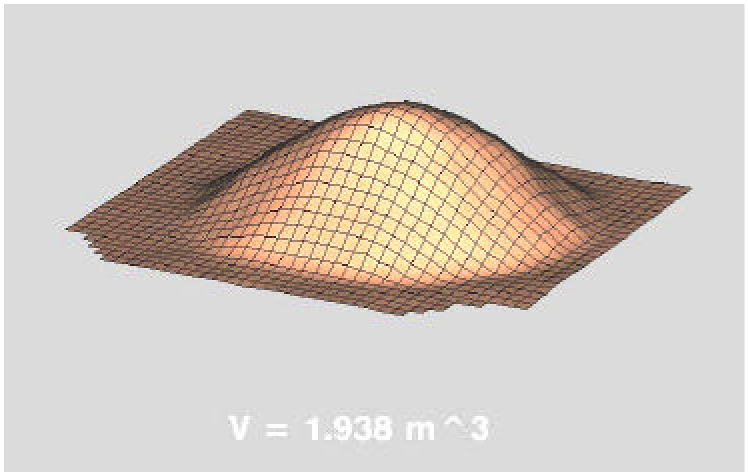
Step	**Brief Description**	**GUI Windows in Fig. 3.1**	5. Correspon ding **DX Visual Program Pages in Appendix A**
Generate surface	The point cloud data is mapped to the 2D grid to generate a surface. The height at a grid point is a weighted average of point cloud Z values. The weighting depends on a number of nearest neighbors within a radius around the grid point. The weighting, number of nearest neighbors, and radius are user-specified.	Surface Regrid Parameters (Radius factor, Regrid nearest, Exponent) – Fig. 3.1b	surface3d Construct2d
Surface display	The surface is displayed in the *Scanned Object* window along with the point cloud data. Several parameters can be set to change how the surface is drawn. A median filter can also be applied to the surface. The volume under the surface is also computed (Fig. 3.2c).	Surface Appearance (Initial surface, Both surfaces) – Fig. 3.1c	surf_orig SurfDraw
Input updated scan data	The file of XYZ data from the updated laser scan for location C is read in. It is filtered, cropped, displayed similar to the initial laser scan data. The same 2D grid, from the initial laser scans, is used to map the updated point cloud data to its own surface.	C input (Updated C scan data) – Fig. 3.1d	c_update
Generate updated surface	The mapping to the 2D grid of updated laser scan point cloud data is substituted into the mapping from the initial combined laser scan data. This ensures that the updated surface will reflect the updated scan from the initial scans. The updated surface is displayed in the *Updated Scanned Object* window (Fig. 3.4).	Surface Regrid Parameters (Radius factor for updated surface) – Fig. 3.1b Surface Appearance (Updated surface, Both surfaces) – Fig. 3.1c	surf_c_dig surf_dig SurfDraw



a. Sand pile

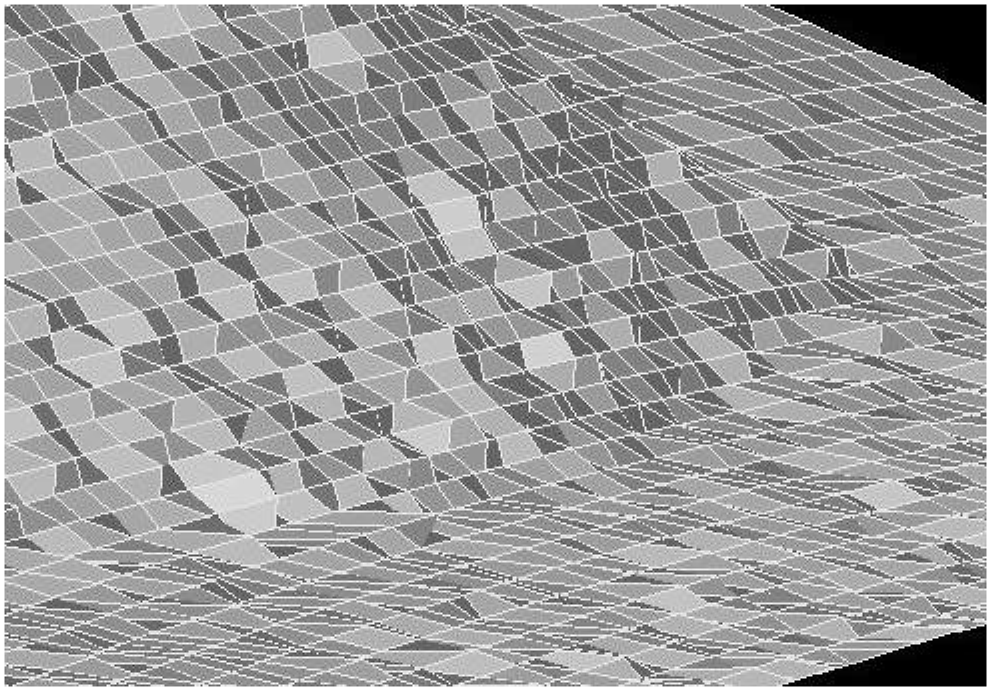


b. Combined point cloud

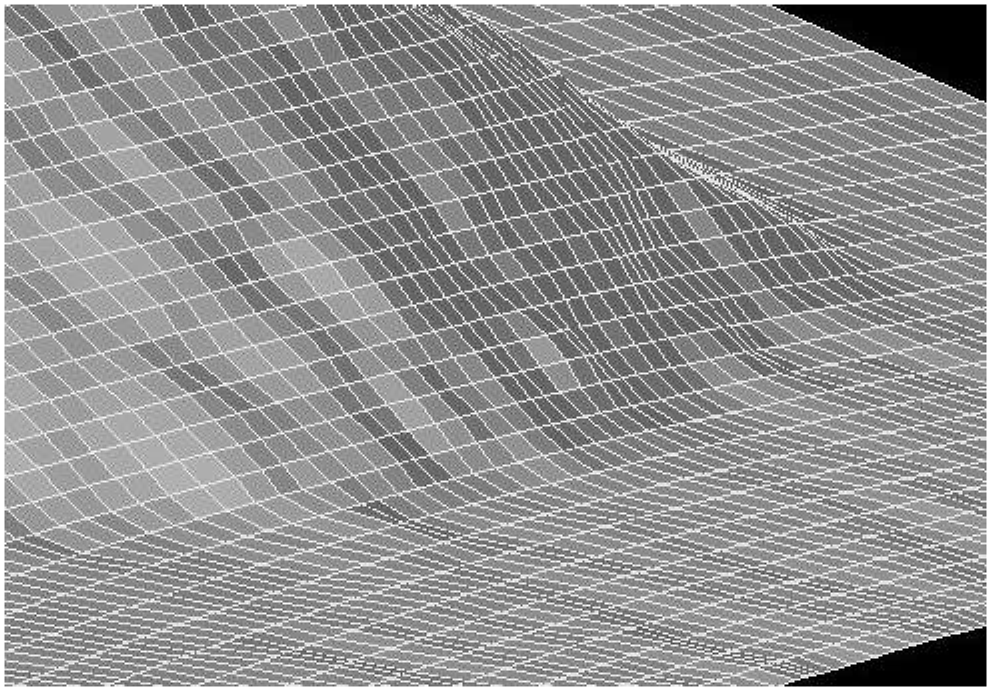


c. 3-D surface

Figure 3.2. Scanned Object.

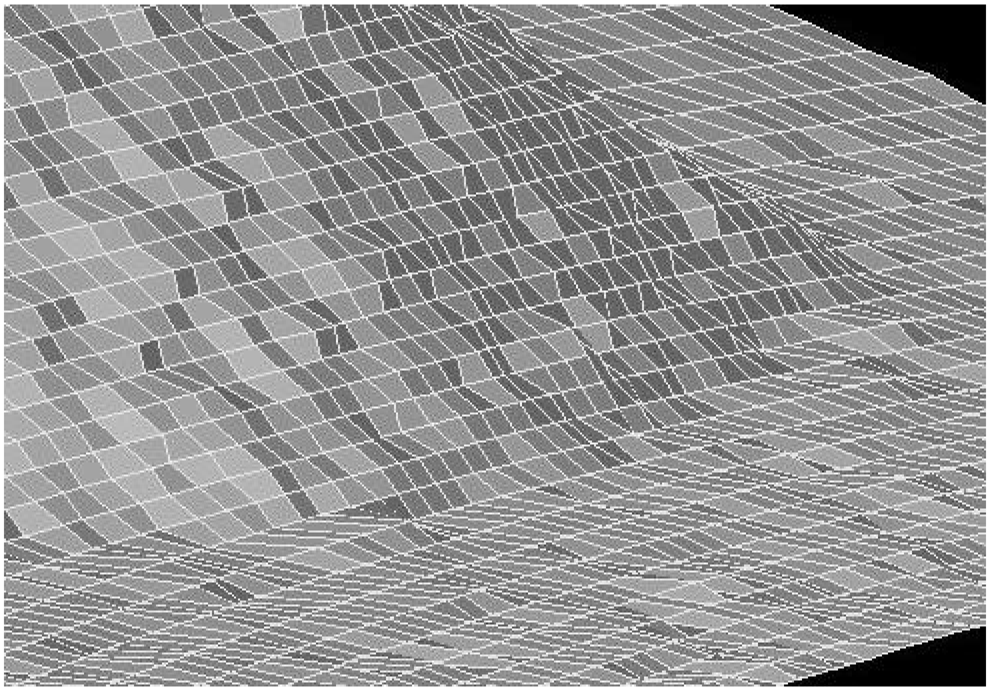


a. Surface Without Filtering.



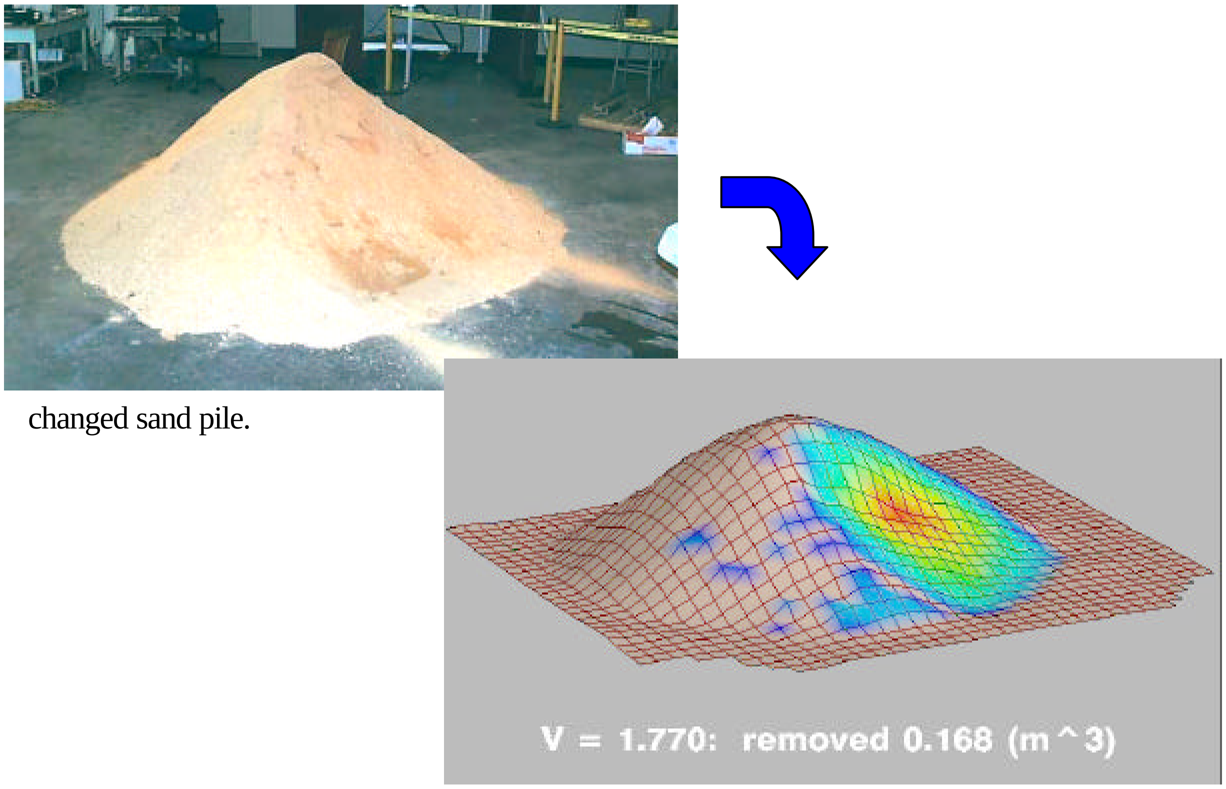
b. Surface using Gaussian Filter.

Figure 3.3. Raw Grid.



c. Surface Using Median Filter

Figure 3.3. Raw Grid. (cont.).

a. 

changed sand pile.

b. Regenerated surface incorporating new data set.

Figure 3.4. Updated Scanned Object.

3.3 Detailed Explanation

For each of the two objects, four initial scans were made using the laser scanner. The data is transmitted by the scanner as four files, one for each scan, of XYZ coordinates. Each line in the file has one XYZ coordinate and an intensity value. There are also several header lines delimited by a “#” sign. The first few lines of a typical file are shown in Fig. 3.5.

```
#ASCII Export File
#Automatically generated from - RLMSSCAN
#  coordinates x y z [a] in units of meter
#
6.635 -1.739 13.113 74
6.632 -1.832 13.108 74
6.629 -1.925 13.102 72
6.639 -2.023 13.127 72
6.625 -2.112 13.098 70
6.576 -2.190 12.996 76
```

Figure 3.5. Sample Output Data from the Laser Scanner.

A DX header file must be created to read in the XYZ laser scan data. It lists the name of the actual data file and several keywords and associated values that describe how the data should be read. A sample DX header file is shown in Fig. 3.6.

```
file = data/demo_a.txt
header = lines 4
grid = 50 x 90
format = text
interleaving = field
field = locations, data, intensity
structure = 2-vector, scalar, scalar
type = float, float, float
end
```

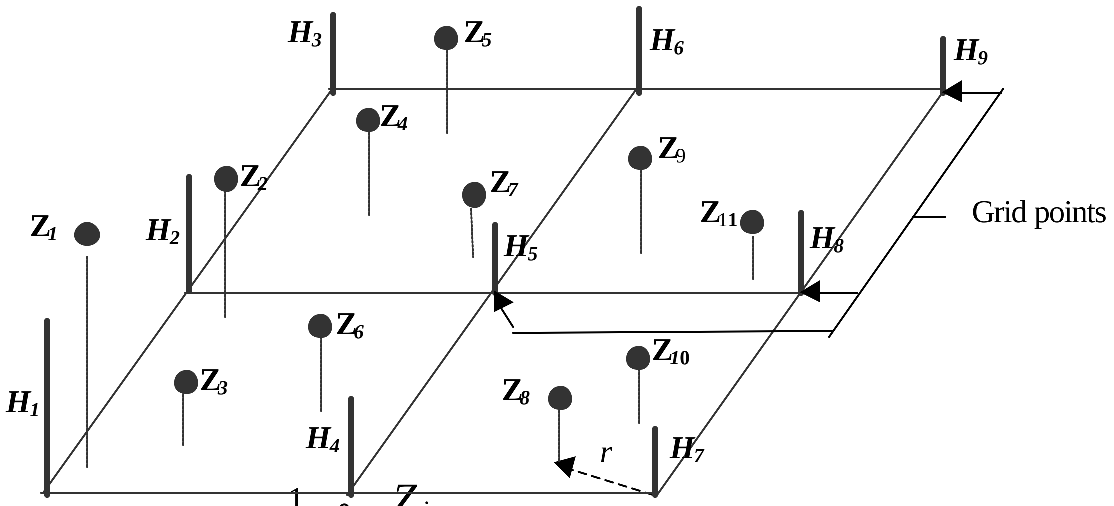
Figure 3.6. Sample DX Header File for Scan Data.

An explanation of the keywords can be found in the DX QuickStart Guide (IBM, 1997). The most important keyword that will vary between different laser scan datasets is "grid". This keyword corresponds to the horizontal and vertical size of the grid used when acquiring the scan data. A DX header file is created for each set of scan data. The names of the header files need to be input to DX. Rather than input the names of all four header files in the GUI individually, a single file that lists all four header files names is created. This file is input in *Raw Data* under "File of filenames".

To filter the range values of the laser scan data, the XYZ values are converted to a spherical coordinate system where the data are expressed as two angle values and one range value. The range value is filtered while the angle values remain constant. The filtered range value is converted back to XYZ values. Either a Gaussian filter or a median filter can be applied. The Gaussian filter computes a filtered range value by averaging itself with its neighboring values. The median filter computes a filtered range value by ranking the neighboring values and itself in numerical order and then uses the median value of the ranking as the new value. The median filter is much better at eliminating noisy, spurious data while preserving sharp changes in range values.

Figure 3.3a shows the raw grid for one set of laser scan data. Figure 3.3b show the affect of applying a Gaussian filter which has the affect of turning the interface between the sand pile and floor into a gentle slope. Figure 3.3c shows the affect of the median filter. Some of the noise has been reduced while maintaining the sharp change in range at the interface between the sand pile and the floor.

To map the XYZ cropped and combined point cloud data to the 2D grid, a weighted average of the Z values is taken (Fig. 3.7). The *Z* values that are considered are within a certain radius of a grid point. Within the radius, the number of *Z* values can also be limited. Typically, the radius is the same size as an individual grid and the number of points to consider within that radius is set to 8. The weighting is an exponent on the radius and is typically set to 1. Once the heights (H_i in Fig. 3.7) are determined, a surface can be generated through those points. All three parameters can be adjusted to give the most appropriate looking surface.



$$H_i = \frac{1}{n} \sum \frac{Z_i}{r_i^a}$$

$$n = \sum \frac{1}{r_i^a}$$

$$Z_i = \text{z - values of the point cloud data}$$

$$r_i = \text{distance of point cloud data point to grid point}$$

$$a = \text{user defined exponent}$$

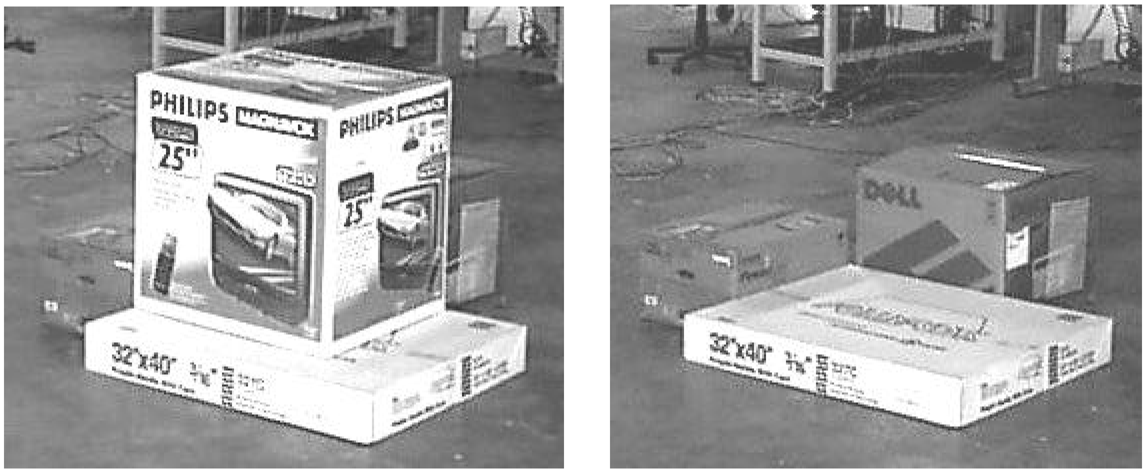
$$\text{Volume} = (\text{Total Grid Area}) \times \sum H_i$$

Figure 3.7. Weighting to map point cloud data to 2D regular grid.

The problem with generating a surface reflecting the updated laser scan data is that there is only one updated scan and it overlaps scan data from the other three scans that are not updated. To solve this problem, first the height values on the 2D grid are computed for just the updated scan data. This generates height values only in the region of the updated scan. Where updated scan values are not available to be used to generate height values on the grid, null or invalid height values are generated. Then, to combine the initial and updated height values, where the heights of the updated scan are null, the heights from the initial four scans are used, else the heights from the updated scan are used. The effect of the updated scan on the initial scan heights can be seen by turning on "Update C grid" in the *Surface Appearance* GUI window.

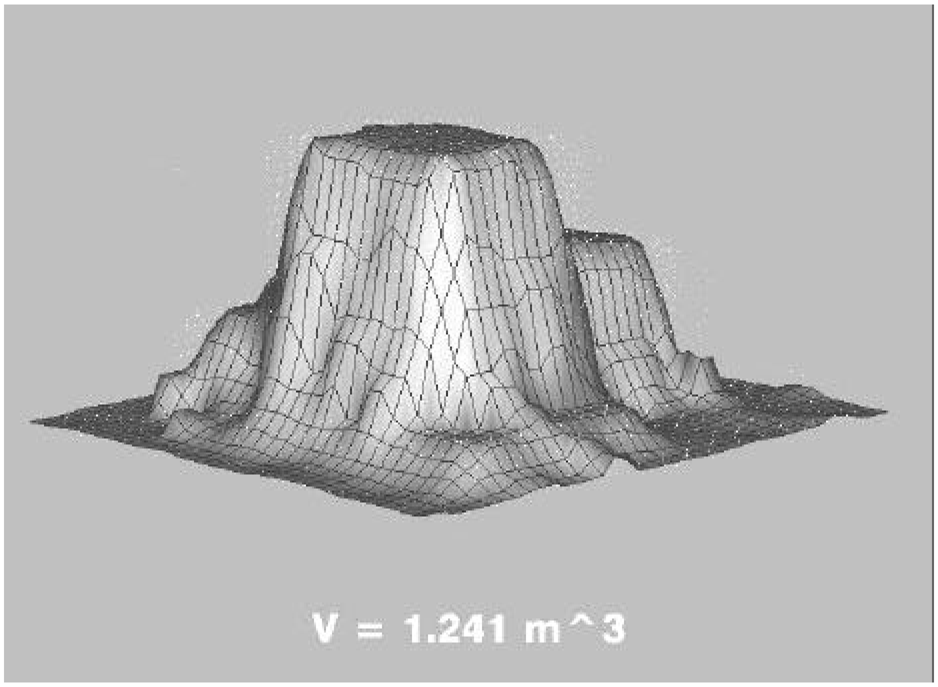
3.4 Box Dataset

Prior to the sand pile data, laser scans were taken of a set of boxes shown in Fig. 3.8. One of the boxes was removed and the scene was rescanned from one location. The resulting visualization is shown in Fig. 3.9. It is clear that the visualization cannot represent the vertical surfaces of the boxes. This is because the point cloud data is mapped to a horizontal 2D grid where there can only be one height value at a grid point. A vertical surface would require at least two height values at a grid point.

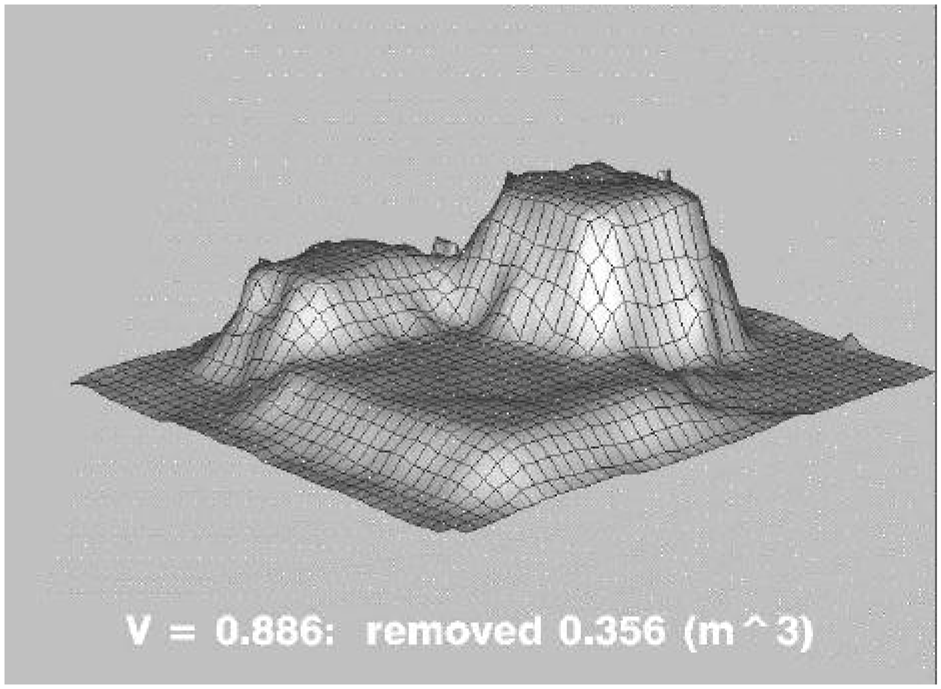


a. All Boxes b. One Box Removed

Figure 3.8. Boxes That Were Scanned.



a. Mesh for All Boxes.



b. Mesh for One Box Removed.

Figure 3.9. Visualization of Boxes.

3.5 Volume Calculations

The actual volume of the sand pile is not known, so no comparison between the computed volume and the actual volume can be made. However, the computed volume of sand removed was 0.168 m^3 which compares favorably with the actual volume of sand removed of 0.15 m^3.

The volume of the four boxes scanned is 0.73 m^3 while the computed volume of the boxes is 1.241 m^3. The discrepancy is because *demoscan* cannot accurately visualize the vertical surfaces of the boxes. The volume of the box of the box removed is 0.351 m^3 while the computed volume of that box is 0.356 m^3. It is unclear why determining the volume of the box removed is so accurate when the volume calculation of all the boxes is not.

More calculations with *demoscan* need to be done to determine the effect of the grid size and other parameters which map the point cloud data to a 2D grid on the volume. Other methods to accurately represent vertical surfaces need to be developed.

3.6 References

IBM (1997), IBM Visualization Data Explorer QuickStart Guide, Version 3, Release 1, Modification 4, May.

4.0 TIN Surfaces: Data Representation and Volumes

4.1 Introduction

The purpose of this chapter is to provide an alternative approach to the approach described in Chapter 3 for representing spatial data by a surface and for subsequently calculating volumes.

A surface in *x,y,z*-space consisting of triangles which are connected along their edges is called a

"triangulated surface".

A more precise definition of triangulated surfaces will be given in Chapter 5.

Triangulated surfaces, sometimes called "triangulated nets", have been established over the past decade as a widely accepted tool for 3-D shape representation. They accommodate variable density requirements by not being tied to a regular mesh of uniform density. They accommodate the inclusion of predetermined break lines as part of the edge pattern. They accommodate, as examined in Chapter 5, arbitrary shapes in 3-D.

The discussion in this chapter is restricted to a special kind of triangulated surfaces which will be called in this report

"elevated triangular surfaces".

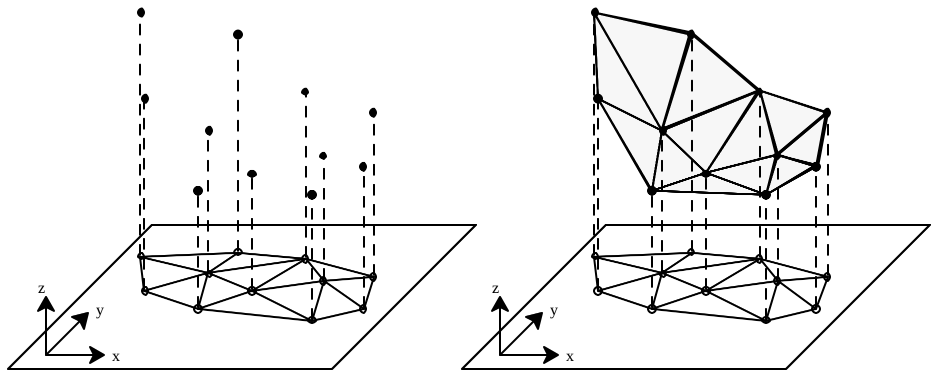


Figure 4.1 A triangulation in the *x,y*-plane, its elevated vertices and the corresponding elevated triangulation.

Each such surface rises above a

“triangulation” or “triangulated mesh”

in the footprint region, that is, the projection of the surface into the x,y-plane Fig. 4.1. Such a triangulation is a covering of the footprint region by footprint triangles t_k, $k = 1, \ldots, l$ with vertices $v_j = (\tilde{x}_j, \tilde{y}_j)$, $j = 1, \ldots, m$. Those triangles are not permitted to “overlap”, that is, two different triangles of a triangulation may meet only at a single vertex or an entire edge of both triangles.

Following common usage in computational geography, the term

“**triangulated irregular network (TIN)**”

will also be used to denote a triangulation. The associated elevated triangulated surface with vertices $V_j = (\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$ is called a

*TIN surface***.**

Each triangle T_k, $k = 1, \ldots, l$, of the TIN surface corresponds to a unique footprint triangle t_k in the triangulation. The footprint triangle is a vertical projection of its counterpart in x,y,z-space into the x,y-plane. Edges and vertices of the triangulation are referred to as footprints of corresponding edges and vertices of the TIN surface. In what follows, lower case letters are used for triangles, edges, and vertices of the triangulation in the footprint region, whereas capital letters denote such features of the TIN surface.

A TIN surface is fully determined by specifying an elevation $\tilde{z}_j$ at each vertex $(\tilde{x}_j, \tilde{y}_j)$ of its footprint triangulation.

TIN surfaces are typically used for either interpolating or approximating sets of data points $P_i = (x_i, y_i, z_i)$ with footprints $p_i = (x_i, y_i)$. Such data sets, which specify elevations above footprints, may, for instance, represent terrain surfaces, as long as overhangs, arches, and similar terrain features are ruled out. Construction surfaces may also fall into this category. However, if undercuts or tunnels are present, then more general triangulated surfaces than TIN surfaces need to be applied. Such surfaces will be discussed in Chapter 5.

Interpolation requires that the interpolating surface passes through every data point P_i. The task of approximation, on the other hand, is to minimize as much as possible the deviations from the data points according to some “measure-of-fit”.

4.2 Interpolating Data; Delaunay Triangulations

In order to interpolate data points $P_i = (x_i, y_i, z_i)$, $i = 1, ..., n$, it makes sense to choose those data points P_i as vertices V_j of the TIN surface and, consequently, their footprints $p_i = (x_i, y_i)$ as vertices v_j of the triangulation. Exceptions are data points which already happen to lie on a surface triangle spanned by neighboring data points. If different data points share the same footprint, only one of them can be interpolated. For those reasons, the vertices v_j of the TIN surface may not represent the full data set and are therefore differently indexed.

There are many ways of triangulating a set of planar points and, consequently, there are many interpolating TIN surfaces for any given set of data points. While all such interpolating surfaces are of zero error as far as replication of the data points is concerned, the quality of the representation depends very much on the choice of the triangulation. In particular, the occurrence of long edges and associated skinny triangles in the interior would clearly be undesirable, as it would distort the appearance of the surface.

For every set of planar points, there exists a triangulation, usually unique, which satisfies the following (Delaunay 1934)

(4.2.1) **Empty Circle Criterion**: *No triangulation vertex v_j lies in the interior of the circumcircle of any triangle t_k of the triangulation.*

Such a triangulation is known as

"Delaunay triangulation" or *"Delaunay TIN"*.

Figure 4.2 features two triangulations of the same set of points, one a Delaunay triangulation, the other violating the empty circle criterion (4.2.1).

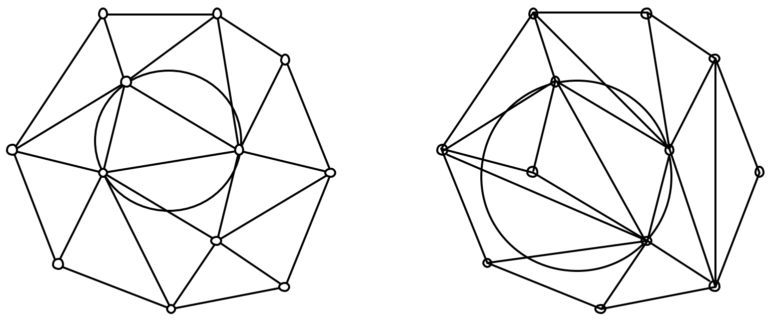


Figure 4.2. Two triangulations of the same set of points. The one on the left is Delaunay.

The Delaunay triangulation is uniquely determined if no circumcircle of any of its triangles contains additional data footprints on its periphery. In that exceptional case, the footprints p_i on the periphery determine a convex polygon which can be subdivided by diagonals in several ways into triangles, and each of these subdivisions is acceptable within a Delaunay triangulation.

The Delaunay triangulation is widely considered a standard because of its essential uniqueness. For this reason, and since it tends to avoid long skinny triangles when possible, it is the method of choice for many triangulations.

The concept of a Delaunay triangulation permits an important generalization (Bernal 1988). Given a set of

"constraint edges",

namely a set of straight line segments in the x,y-plane, a

"constrained Delaunay triangulation"

is required to have those pre-specified constraint edges as edges of the triangulation. Constraint edges may not cross each other. More precisely, if two different edges meet, they must meet at a single point which is an end point of both edges. Given such a set of constraint edges, any point p in the x,y-plane is considered

"visible"

from point q, if its view is not obstructed by one of the constraint edges. Constrained Delaunay triangulations are then characterized by the following

(4.2.2) **Constrained Empty Circle Criterion:** *No triangulation vertex v_j lies in the interior of the circumcircle of any triangle t_k of the triangulation and is also visible from the interior of that triangle.*

Constrained Delaunay triangulations are again essentially unique. The only exceptions occur if the periphery of the circumcircle of a triangle contains additional vertices that are visible from the interior of the triangle.

4.3 Approximating Data

Interpolation is clearly the most conservative approach to constructing TIN surfaces from given elevation data points. It must be kept in mind, however, that, in this approach, data points arising from artifacts or process-malfunctions are accommodated in the same way as all the other data points. In addition, measurements are subject to noise, that is, random statistical deviations from the "true" value. Interpolation is thus not necessarily the optimal approach unless it has been preceded by some screening or editing procedures, which strictly speaking, amount to approximations. Also, if there are many data points, the interpolating TIN surface structure may be too cumbersome to use, and it may be desirable to reduce the data set by selecting a subset of

"*critical points***".**

For the purposes to be discussed here, those critical points are then interpolated -- possibly after suitable elevation adjustments -- by a Delaunay surface. That clearly amounts to approximation, where the goal is to either reduce the size of a data set or to minimize some "measure-of-fit" by an approximating surface. The vertices $V_j = (\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$ of the TIN surface will, in general, not correspond to the given data points $P_i = (x_i, y_i, z_i)$.

Most triangulation based approaches which are sensitive to terrain variability fall into one of two general categories. The "top-down" approach selectively deletes data points P_i until the desired sample size is reached. The "bottom-up" approach starts essentially from scratch and builds up towards the desired sample size by successively adding data points P_i in some sequence. The respective merits of those approaches are as yet to be evaluated. The top-down approach may be most useful if the sample percentage is large. Our procedures fall into the "bottom-up" category. They use successive refinements of Delaunay surfaces based on insertion techniques.

A major question in the general context of surface representation then is how to assess the accuracy of an approximation. Ideally, the approximating surface should be compared to the "actual" surface. In most cases, the actual surface remains elusive, and the only information about it is the the full data set. Accuracy assessment is then limited to determining how well the approximating surface represents the data set. Residual-based measures-of-fit are commonly used to gauge the approximation to given data. The

$$\text{"}\mathit{residual}\text{"}\, r_i = r_i(x_i, y_i) = z_i - \hat{z}_i$$

indicates the shortfall -- positive or negative -- of the elevation $\hat{z}_i$, which the surface assumes at the data footprint (x_i, y_i), compared to the given elevation z_i at that point. The following measures-of-fit are most commonly used.

(4.3.1) **Maximum Deviation (MAX).** *The size (= absolute value) of the largest residual is used as measure-of-fit:*

$$\text{Max}_{i=1}^{n} = |\, r_i \,|$$

(4.3.2) **Root Mean Square (RMS).** *The square root of the average squared residual is used as measure-of-fit:*

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} r_i^2}$$

(4.3.3) **Average Size Deviations (ASD).** *The average of the absolute values of the residuals is used as measure-of-fit:*

$$\frac{1}{n}\sum_{i=1}^{n} |\, r_i \,|$$

The choice between measures-of-fit depends on the intended application. The familiar "least squares" approach minimizes the sum of the squares of the residuals, which is clearly equivalent to minimizing the RMS measure. In general, the RMS measure is less sensitive to outliers, that is, extreme elevation values of the data set than the MAX measure. The ASD measure is least sensitive to outliers, and may be used to screen out undesirable data.

4.4 Level Volumes of Triangulated Surfaces

Given a particular elevation level z_0, consider the spatial region bounded below by z_0, above by an elevated surface, and cut off vertically either along the boundary of the footprint region, or at the intersection of surface and elevation level. The volume of that spatial region is called the

"cut volume" $V_{cut}(z_0)$

for the elevation level z_0 . The corresponding

"fill volume" $V_{fill}(z_0)$

is defined as the volume of the region below the elevation level z_0 and bounded below by the surface. Lateral cut-off is again vertically along the boundary of the footprint region or at the intersection of surface and elevation level. The projections of cut volumes and fill volumes onto the x,y-plane cover precisely the footprint area, and meet only along the projections of the lines along which the surface intersects the elevation level. The following discussion is restricted to cut volumes since everything is analogous for fill volumes.

The cut volume $V_{cut}(z_0)$ is readily computed if bounded by a TIN surface. The enclosed space is the union of triangular prisms. As shown in Fig.4.3, such a prism is spanned by a usually tilted triangle on top and its footprint at elevation z_0 on the bottom. It is bounded at three sides by trapezoids spanned by edges of the top triangle and their respective footprints below.

Let h_1, h_2,, h_3 denote the lengths of the parallel sides of the trapezoids, that is, the elevations of the top vertices above z_0. Let a denote the area of the footprint triangle. Then the volume of the prism is given by

$$volume = \frac{h_1 + h_2 + h_3}{3} \times a$$

The decomposition of the volume by triangular prisms is readily identified. The straightforward

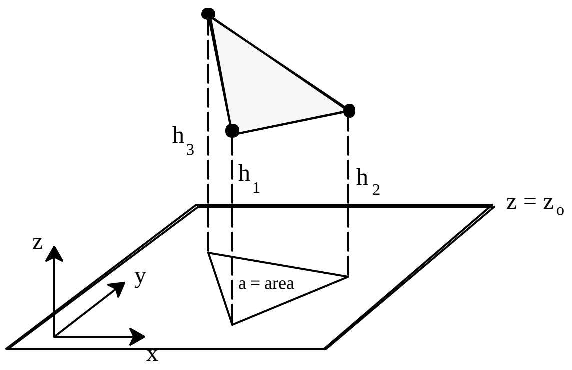


Figure 4.3. A Triangular Prism.

case is that of a triangle

$$T_k = (V_{k_1}, V_{k_2}, V_{k_3}), \quad V_{k_s} = (\tilde{x}_{j_s}, \tilde{y}_{j_s}, \tilde{z}_{j_s}), \quad s = 1, 2, 3,$$

of the TIN surface fully above the z_0 elevation level. In this case, it by itself along with its footprint defines one of those prisms. The heights h_1, h_2, h_3 are given by

$$h_s = \tilde{z}_{j_s} - z_0 \geq 0, \quad s = 1, 2, 3$$

If triangle T_k lies below the stipulated elevation level, then it does not contribute a prism. The are two intermediate cases. In the first case (Fig. 4.4a), two vertices of triangle T_k lie strictly above the

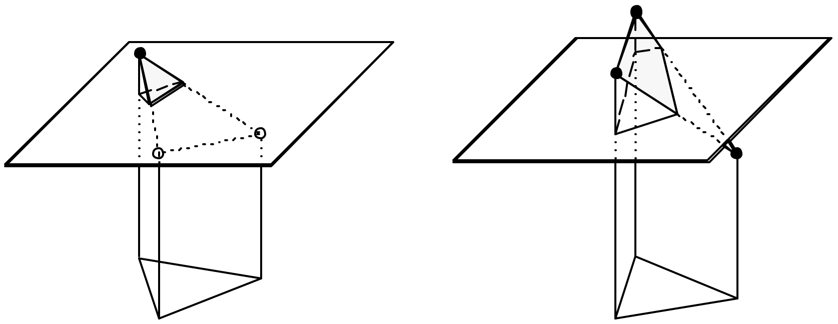


Figure 4.4: Portions of triangular prisms cut by a horizontal level plane.

elevation level z_0, whereas the remaining vertex lies strictly below. In this case, only a portion forming a quadrangular prism over a quadrangle footprint contributes to the overall volume. Two vertices of the footprint quadrangle are calculated as the points where those edges of triangle T_k which connect to the low vertex intersect elevation level z_0. Once the footprint quadrangle has been determined, any of its two decompositions into triangles will produce two triangular prisms with the obvious height adjustments. The remaining case (Fig. 4.4b), with one vertex strictly above, and the other vertices of triangle T_k either at or below elevation level z_0, leads to a single triangular prism, whose footprint is a triangular portion of the full footprint of triangle T_k.

4.5 The Basic Insertion Method

The task of interpolating a given set of data points $P_i = (x_i, y_i, z_i)$ by a Delaunay surface is addressed first. The so-called “insertion method” is frequently used for this purpose, and will now be outlined.

In the procedures discussed in this report, all data footprints are enclosed in a rectangular

“map” $\{(x, y)$: x_{min} £ x £ x_{max}, y_{min} £ y £ $y_{max}\}$,

and by including its four corners, the entire map will be covered by the triangulation. If there are no data points at those corners, arbitrary “no-data values" will be assigned. Only after a final triangulation of the map area has been determined, an actual footprint region of the data points will be established. It is obvious that the level volume of a TIN surface depends critically on the definition of that footprint region. The problem is to decide which data points are boundary points and which edges are boundary edges. This problem is particularly acute if an object casts a shadow when scanned. The most straightforward solution is to specify a

“maximum acceptable length”

for an edge in the triangulation, and to delete all triangles with an edge above that threshold, along with triangles meeting any of the four corners of the map.

The initial Delaunay triangulation of the map consists of just the two triangles created by splitting the map rectangle along one of its diagonals. Data footprints $p_i = (x_i, y_i)$ are then inserted successively and, after each such insertion, the resulting triangulation is readjusted to maintain the Delaunay property. The process stops after all data have been inserted. The sequence of the insertion will affect the computational effort. The end result, however, will be essentially independent of the insertion sequence. Differences may occur in the degenerate cases where the Delaunay triangulation is not unique.

In more detail, insertion of $p_i = (x_i, y_i)$ proceeds as follows. A triangle containing p_i is determined. If p_i lies in the interior of this triangle t_k, then a new triangulation is created by connecting p_i to the corners of t_k (Fig. 4.5) so that there are now three triangles where previously was just triangle t_k.

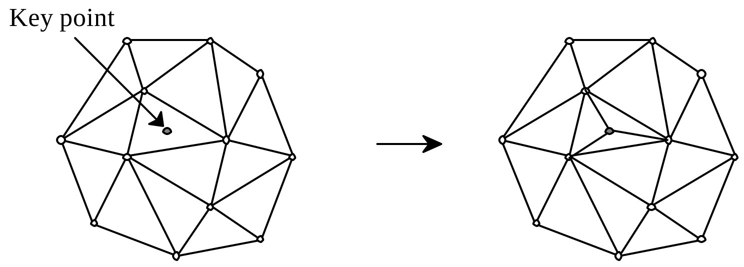


Figure 4.5. Triangulation Immediately After Insertion of Key Point.

In the unlikely case that p_i falls on an edge of triangle t_k, the initial insertion procedure has to be modified in obvious fashion.

The new triangulations generated by the above procedures are, in general, no longer Delaunay triangulations. They will have to be readjusted to meet the empty circle criterion. That adjustment takes the form of repeated

“diagonal interchanges or *flips”*.

Such a diagonal interchange (Fig. 4.6) may be carried out whenever two triangles of a triangulation are adjacent along an edge and together form a quadrangle whose two diagonals intersect in its interior. One of those diagonals is the common edge of the above triangles. By using the other diagonal to divide the quadrangle, two new triangles are created in the space of the two old ones and become part of an altered triangulation.

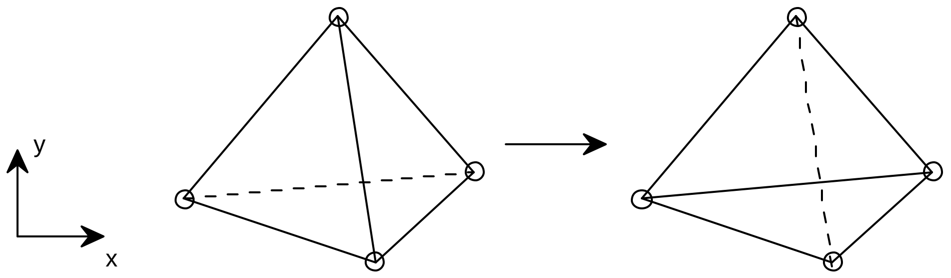


Figure 4.6. Diagonal Flips.

Unless an edge *e* of any triangle *t* in the triangulation lies on the map boundary, it will be shared by a unique adjacent triangle. Each such neighboring triangle, furthermore, has a unique vertex *v* which is not an end of edge *e*. Here such vertices will be called

"opposite vertices"

of the triangle. Lawson (1977) has shown that

(4.5.1) *if the empty circle criterion holds for the opposite vertices of each triangle, then it holds universally*

It is also easy to verify that

(4.5.2) *if an opposite vertex of a triangle t violates the empty circle criterion, i.e., if that vertex lies outside the circumcircle of t, then that vertex together with t defines a quadrangle which permits a diagonal interchange.*

That diagonal interchange will rectify that particular violation of the Delaunay condition.

The readjustment algorithm (Fig. 4.7) then is as follows: If the opposite vertices of the new triangles created by the insertion do not violate the empty circle criterion, then the new triangulation is already Delaunay by (4.5.1). Otherwise, select one opposite vertex which violates the empty circle criterion, and execute the associated (4.5.2) diagonal interchange. This creates new triangles whose opposite vertices may or may not violate the empty circle criterion. In the former case, a beneficial diagonal interchange is again triggered. In the latter case, attention shifts to another one of the new triangles.

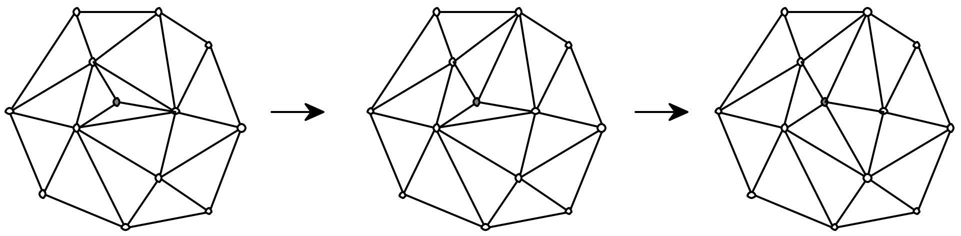


Figure 4.7. Adjustment to Achieve a Delaunay Triangulation.

Lawson (1977) has also proved that the adjustment procedure terminates with a Delaunay triangulation containing the newly inserted point.

If there are pre-specified constraint edges, the approach to achieve a constrained Delaunay triangulation involves again the two steps of insertion followed by readjusting the triangulation to satisfy the respective Delaunay criteria.

Inserting a point into a constrained Delaunay triangulation is identical to the procedure followed in the non-constrained case. Note that a point may not be inserted into a constraint edge. The readjustment process requires only a slight modification. If a diagonal interchange is considered where the current diagonal is a constraint edge, then that diagonal interchange is blocked.

The only new aspect is that in building the triangulation, not only points, but also edges -- the constraint edges -- need to be inserted, generally, between two existing vertices of the triangulation.

One approach, developed by Bernal, is to remove all edges crossed by edge *e*. This creates a polygon divided into two polygonal parts by the insertion of *e*. Those two portions are then triangulated by inserting diagonals. New triangles are examined for possible Delaunay violations and the corresponding diagonal interchanges are carried out until a constrained Delaunay triangulation has been achieved. In an alternate approach, edge crossings are removed by diagonal interchanges. The procedure is complicated because such diagonal interchanges may not immediately exist and have to be set up by a sequence of diagonal interchanges that keep the number of edge crossings constant.

4.6 Selective Insertion Techniques for Approximation

A bottom-up approach for selecting a subset of critical data points with the purpose of approximation can now be described, touching on various options and strategy choices which have been implemented.

The starting point is again a trivial triangulation of a rectangular map, consisting of just two triangles. If constraint edges have been specified, then those edges and their endpoints are inserted first. There may also be data points which are desired to be in the critical subset no matter what. Those

"pre-specified points"

are inserted next in any order.

The remaining data points are then examined with respect to the current (constrained) Delaunay triangulation. In each triangle t_k of the triangulation, a

"key point"

is selected among the data footprints p_i which are located in the triangle. For the purposes of this report, only one selection rule will be considered.

(4.6.1) **Key Point Selection:** *Select the point in the triangle for which the size of the residual*

$$| r_i | = | z_i - \hat{z}_i |$$

is largest.

The triangles and their key points are then ranked by a suitable criterion and maintained in a sorted heap accordingly. The highest ranking triangle is located at the top of that list. The keypoint of this triangle is first in line to be selected as the next critical point to be inserted, unless the process is terminated. The ranking thus determines the sequence in which key points are selected as critical points. Here two alternate criteria for ranking triangles are considered.

(4.6.2) **Ranking by Maximum Deviation:** *Rank triangles t_k according to the size of the residuals*

$$| r_i | = | z_i - \hat{z}_i |$$

of the respective key points of the triangles.

This ranking criterion is most commonly used. It is geared towards an aggressive reduction of the maximum residual error.

(4.6.3) **Ranking by Volume Deviation:** *Rank triangles* t_k *according to the product*

$$| z_i - \hat{z}_i | ´ \; area\,(t_k)$$

of the size of the residuals times the area of the triangle.

The product between size of the residual at the key point of the triangle and its area describes up to a factor of 1/3 the volume change that an insertion of the key point would cause, if the current Delaunay surface were considered as bounding a volume. For this reason, the term “ranking by volume deviation” was chosen. When using that criterion, the resulting triangulation tends to be more locally homogeneous as to triangle size, because a large triangle may reach top rank, -- and be broken up -- even if its associated residual is relatively small. Ongoing work by NIST for the Army Corps of Engineers appears to indicate, that the ranking criterion (4.6.3) yields better results than ranking criterion (4.6.2) if the given data are to support a volume computation.

The computational effort of the selection procedure to generate a Delaunay TIN is greatest in the early stages of the procedure when many data points need to be examined in order to determine key points in a triangle. For this reason it is not recommended to start with a very sparse initial triangulation. An additional reason is that a triangulation represented by only a few data points does not provide good guidance for the selection of key points and, subsequently, critical points. Of the many ways an initial selection of data points can be achieved, the following has been implemented.

(4.6.4) **Initial Binning:** *Divide map into an almost square grid of bins. In each bin of the grid select the data point* P_i *whose footprint* $p_i = (x_i, y_i)$ *lies closest to the center of the bin and interpolate this subset of data points by a Delaunay surface.*

4.7 Data Filtering

After a TIN surface has been constructed whose vertices are a sample of a larger set of data points P_i, $i = 1, ..., n$, various measures-of-fit can be improved by adjusting the elevation $\tilde{z}_j$ of the vertices $V_j = \left(\hat{x}_j, \hat{y}_j, \hat{z}_j\right)$, $j = 1, \ldots, m$, of the surface.

In order to describe some such techniques, the notion of the *“star”*

of a vertex v of the triangulation, namely, the union of all triangles t which have v as a vertex is introduced.

When varying the elevation of the associated surface vertex V, the surface triangles T above the star of v are raised or lowered, while their edges opposite to vertex V stay fixed. The picture that comes to mind is of raising or lowering a tent pole.

(4.7.1) **Local RMS Optimization**. *Adjust the elevation $\hat{z}_j$ at triangulation vertex v_j so that the RMS measure of the data points with footprints in the star of v_j is minimized.*

If a local RMS step yields an improvement, then the RMS measure of all data points is improved by this step. Conversely, it can be shown, that

(4.7.2) *if no local RMS step yields an improvement, then the surface is the surface of smallest RMS error among all TIN surfaces over the same footprint triangulation.*

These observations suggest the following RMS optimization procedure. Perform the local optimization step (4.7.1) for each vertex whose elevation has not been preset as part of a constraint edge. This constitutes one pass of the procedure. Passes are repeated until no local optimization step yields an elevation change above a specified tolerance.

The ASD measure-of-fit of a TIN surface with given footprint triangulation can be improved by an analogous procedure. Again, a local optimization step can be defined.

(4.7.3) **Local ASD Optimization.** *Adjust the elevation $\hat{z}_j$ at triangulation vertex v_j so that the ASD measure of the data points with footprints in the star of v_j is minimized.*

While it is still true that every local ASD improvement results in a global ASD improvement, the analog to the optimization criterion (4.7.2) no longer holds. Local ASD optimizations analogous to the RMS optimization procedure, that is, repeating passes through all local optimizations at eligible vertices, will converge, but not necessarily terminate with the optimal surface. For more accurate ASD optimization algorithms see Bernal and Witzgall (1999). That work also suggests, that while suboptimal, proceeding with local ASD optimizations will come reasonably close to optimality and will thus be sufficient for practical purposes.

The above elevation adjustments can be interpreted as filters where data reduction is combined with a smoothing process. In that sense, RMS optimization corresponds to a Gaussian filter whereas ASD optimization corresponds to a median filter. ASD optimization, in particular, can be used to identify and reduce “outliers” among the original data points.

The key algorithms are the ones addressing the local optimizations (4.7.1) and (4.7.3). The former amounts to setting up a quadratic polynomial in one variable -- the locally optimal elevation at the center vertex of the star in question -- and minimizing it, which, of course, can be done in closed form. As to the local ASD optimization, it requires the determination of a suitable weighted median of data elevations, again an elementary operation. For details see Bernal and Witzgall (1999).

4.8 TIN-Based Programs

TIN-based software developed by the Mathematical and Computational Sciences Division (MCSD) of NIST was applied to data gathered by scanning a sand pile during the demonstration of June 24, 1999. In particular, two major software modules were set up:

TINvolume for generating a TIN surface and calculating volumes

TINscreen for identifying and eliminating data outliers

Both routines are based on a collection of about 50 subroutines written in Fortran 77. They provide a stand-alone software capability for volume calculations and related tasks. The source code for these programs are given in the provided disk. Also included on the disk are programs needed for data preparation.

Table 4.1. *TINvolume routine.*

Step	**Description**	6. *Comments*
Input	Read input files	• *x*, *y*, *z* elevation data • map corners • pre-specified vertices (optional). These points are required by the user to among the TIN vertices.
	Specify limit on number of TIN vertices	
	Specify process options	• ranking strategy: max size/ max volume (see 4.6.2 and 4.6.3). • triangle relevance threshold (see Sect. 4.5)
	Specify RMS vertex adjustment	• number of passes • termination tolerance
	Specify output	• names of output files • output format: VRML, etc.
Initial triangulation	Split map into initial two triangles	
	Triangulate bin centers	• addition to map corners (optional) • using insertion method (see Fig. 4.5 and Fig. 4.7) • bin centers are data points closest to center of bin (see 4.6.4)
	Triangulate pre-specified points	• addition to previous vertices (optional) • using insertion method (see Fig. 4.5 and Fig. 4.7)
Linking	Link data points to triangles and identify their key points	Linked list connect data points located in the same triangle. Each data point is linked to a single triangle. Key points (see 4.6.1) are determined concurrently.
Ranking	Rank triangles by key points	Triangles heap-sorted by increasing (area weighted) residuals in accordance with selected ranking strategy (see 4.6.2 and 4.6.3).
Main iteration	Insert highest ranking key point	See Fig. 4.5.
- while number of vertices $\leq$ specified limit	Make triangulation Delaunay again	Use diagonal flips (see Fig. 4.6)

Step	**Description**	*6. Comments*
	Re-linking / re-ranking	• the linking structure is reset for the new triangles. • new key points are determined concurrently • the triangles are re-sorted in accordance with ranking strategy
Delineate footprint region	Determine "relevant" triangles to constitute the footprint region of the TIN surface	triangles with edges lengths above specified threshold tolerances are deleted from the footprint region (see Sect. 4.5).
Gaussian Filter	Adjust elevations at TIN points so as to minimize RMS.	In each pass, RMS minimization is carried out locally for the star of each TIN vertex (see 4.7.1).
Error Statistics	MAX, RMS, ASD errors	Comparing surface against data points (see 4.3.1, 4.3.2, and 4.3.3)
Volume - for each specified elevation level	Calculate cut/fill volumes.	Prismodial decomposition (see Sect. 4.4)

The *TINscreen* routine first interpolates all possible data points by a TIN surface. It then finds for each TIN vertex v the median of the elevations of the neighboring vertices, that is, the vertices which are connected to vertex v by an edge. If the elevation z at vertex v falls outside specified upper and lower tolerance bounds around the median value, then the data point is considered an outlier and is eliminated. The flow of the routine is exhibited in Table 4.2.

Table 4.2. *TINscreen* routine.

Step	**Description**	7. *Comments*
Input	Read input files	• x, y, z elevation data • map corners
	Specify upper/lower outlier tolerances	
	Specify output	• name of output file • output format: VRML, etc.
Triangulation	Enter data points in any order	using insertion method (see Fig. 4.5 and Fig. 4.7)
Identify outliers - for each TIN vertex	Find median of elevations of neighboring TIN vertices and determine whether TIN vertex is outlier	A TIN vertex is an outlier if it exceeds the specified upper tolerance above the median or the lower tolerance below the median.
Eliminate outliers	Create new x,y,z-file without the outliers	Otherwise the order of data points is preserved

4.9 Sand Pile Volume

The task considered here is to determine the volume of the sand pile from scan data collected from four different locations of the laser scanner. The four sets of scan data had been pre-registered. The general procedure is outlined in Table 4.3.

Table 4.3.

Step	**Brief Description**	**Software Routine Used**
8. Data input	The files from the four initial laser scans as registered to a common coordinate system.	

Step	Brief Description	Software Routine Used
Data combination	The four data scans are combined into a single set of *x, y, z*-data points. The combined data set is cropped to a 4 m to 4 m region containing the sand pile and excluding background features.	
Data screening	The data points are screened for outliers, which are deleted from the data set.	9. TINscreen
Surface generation	A specified number of data points are selected as vertices of a TIN surface.	10. TINvolume
Volume calculations	Volumes bound above the TIN surface and below a specified elevation are calculated for a sequence of potential floor levels at steps of 1 cm.	
Determine floor level and final volume	Second differences of the sequence of volumes are determined manually. A spike in the size of second differences indicates the floor elevation and thus the final volume. The volume at that elevation is selected as the final result.	

The *TINvolume* routine requires specification of an elevation level above which the volume is to be determined. In the case of the sand pile, this elevation level is at floor elevation. This floor elevation needed to be determined from noisy data. For this purpose, volumes were determined for a range of elevations in steps of 1 cm. For this sequence of volumes, first and second differences were determined. It was observed that at a particular elevation, -221 cm, the second difference showed a spike, indicating that -221 cm represented a good estimate of the floor elevation.

The *TINvolume* routine was used for two runs of 2000 points and 3000 points. “Ranking by volume deviation” (4.6.3) was specified. Visualization of the surface with 2000 vertices is provided in Fig. 4.8. The results of the runs, 2000 points and 3000 points, are displayed in Tables 4.4 and 4.5, respectively. The errors in representing the total data points by the two surfaces created with 3000 and 2000 vertices, respectively, are listed in Table 4.6 (see 4.3.1, 4.3.2, and 4.3.3).

Table 4.4. Determination of Floor Elevation with 2000 TIN Points.

Floor Elev. (cm)	**Volume (m^3)**	**First Difference (m^3)**	**Second Difference (m^3)**	**Comment**
-226	2.6356			
		0.1428		
-225	2.4928		0.0001	
		0.1427		
-224	2.3501		0.0000	
		0.1427		
-223	2.2074		0.0022	
		0.1405		
-222	2.0669		0.0196	
		0.1209		
-221	**1.9460**		**0.0392**	← Floor level
		0.0817		
-220	1.8643		0.0208	
		0.0609		
-219	1.8034		0.0048	
		0.0561		
-218	1.7473		0.0014	
		0.0547		
-217	1.6926			

Table 4.5. Determination of Floor Elevation with 3000 TIN Points.

Floor Elev. (cm)	**Volume (m^3)**	**First Difference (m^3)**	**Second Difference (m^3)**	**Comment**
-226	2.6428			
		0.1441		
-225	2.4987		0.0000	
		0.1441		
-224	2.3546		0.0001	
		0.1440		
-223	2.2105		0.0028	
		0.1412		
-222	2.0694		0.0204	
		0.1208		
-221	**1.9485**		**0.0375**	← Floor level
		0.0833		
-220	1.8652		0.0216	
		0.0617		
-219	1.8035		0.0055	
		0.0562		
-218	1.7473		0.0016	
		0.0546		
-217	1.6927			

Table 4.6. Residual Statistics.

No. of Vertices	**MAX (cm)**	**RMS (cm)**	**ASD (cm)**
3000	3.5490	0.7661	0.6118
2000	3.8411	0.8865	0.7166

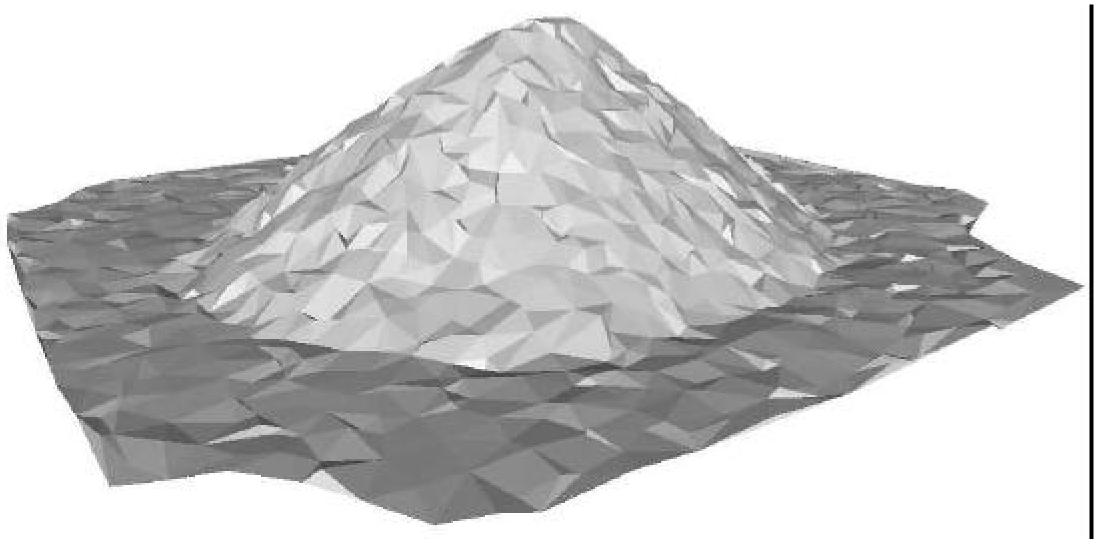


Figure 4.8. Visualization of Sand Pile Surface with 2000 Vertices.

These results highlight the high degree of sensitivity of the volume calculation to the assumed floor level. The constant increments at low elevations indicate that these elevations are clearly below floor level as those increments represent the volume of a 1 cm layer over the foot print region. The TINs for 2000 points and 1000 points have a slightly different footprint region and, therefore, slightly different increments at the lowest elevations displayed above. As the cut elevation rises from its lowest levels, the influence of noise becomes apparent.

The assumption that the actual floor level is indicated by the maximum size of the second difference between consecutive volumes makes intuitive sense but is not corroborated by a theoretical argument.

At the stipulated floor level, there are spurious volume contributions due to data noise. A separate volume computation, to be described in Chapter 5, permits screening for those contributions.

4.9 References

Bernal, J., On constructing Delaunay triangulations from sets constrained by line segments, National Institute of Standards and Technology, Technical Note 1252 (1988).

Bernal, J., and C. Witzgall, Triangulation-based L_1 -fitting of terrain surfaces, National Institute of Standards and Technology, Internal Report 6346 (1999).

Delaunay, B., Sur la sphere vide, Bull. Acad. Sci. USSR (VII), Classe Sci. Mat. (1934), 793-800.

Lawson, C.L., Software for C^1 surface interpolation, Mathematical Software III, J.R. Rice (Ed.), Academic Press, New York (1977), 161-194.

5.0 General Triangulated Surfaces and Tetrahedralization

In this chapter, triangulated surfaces are considered which are not elevated surfaces such as the TIN surfaces of the previous chapter. The concept, "triangulated surface", needs to be defined in more detail. Such a surface consists of flat triangles in space which are connected along their respective edges. If any two different triangles of the surface meet at all, then they meet either in a common edge or a common vertex. Even if these conditions are met, there are rare cases in which an edge may belong to more than two triangles. With this case in view, definition of a triangulated surface requires specification, for each triangle t of at most one neighbor triangle at each edge e of the triangle t. It then can be decided unambiguously, whether a triangulated surface is self-intersecting. For the purposes of this report, it is assumed that triangulated surfaces do not self-intersect.

If the specification of a triangulated surface assigns to each of its triangles a full complement of three neighbor triangles, the surface is

"closed".

A closed triangulated surface -- without self-intersection -- bounds a 3-dimensional polyhedral region. Such regions may be donut-shaped or intertwine as knots. TIN surfaces are not closed, and they do not bound polyhedral regions, although there are various obvious ways of extending a TIN surface into a closed surface. Implicitly, that had to be done, for instance, in order to define cut and fill volumes with respect to an elevation level.

The concept of a

"tetrahedralization"

of a polyhedral region is analogous to the concept of a triangulation of a 2-dimensional polygonal region. The tetrahedra cover the spatial polyhedral region exactly, and the intersection of any two tetrahedra of the tetrahedralization is either empty, or consists of a single vertex, edge or facet of both tetrahedra. It follows that a triangular facet in a tetrahedralization belongs to at most two tetrahedra. If it belongs to only one, it is a boundary triangle. The boundary triangles of a tetrahedralized (polyhedral) region form a closed triangulated surface. Conversely, every closed triangulated surface is the boundary surface of some tetrahedralized region. A tetrahedralized polyhedral region is

"connected"

if any two of its tetrahedra can be connected by a chain of tetrahedra so that each subsequent pair of tetrahedra has a triangular facet in common.

Tetrahedralization techniques provide a powerful tool for addressing fundamental tasks concerning triangulated surfaces. In the context of this report, the following tasks are of interest.

- Determine the Volume of a Polyhedral Region
- Determine the Intersection of Polyhedral Regions

The volume of a tetrahedralized polyhedral region is naturally the sum of the volumes of the tetrahedra in its tetrahedralization. Because closed triangulated surfaces need not be elevated, they may be used to calculate volumes of more general geometrical objects, such as terrain with overhangs or tunnels.

The task of determining "cut and fill volumes" is more involved in the case of general triangulated surfaces. Given here is an ordered pair of closed surfaces, each bounding a polyhedral region. The region described by the first surface is to be "cut" and "filled" so that the second surface results. Typical examples are a construction site where excavations and fill-ins are specified.

In its general formulation, the above task can be carried out by approximating the two bounding surfaces by closed triangulated surfaces, respectively, and then determining the tetrahedralized intersection of the two polyhedral regions corresponding to those triangulated surfaces. Subtracting that intersection from the first and second full polyhedral regions, respectively, will leave connected polyhedral regions, each of which essentially belongs to one of the original polyhedral regions but not the other. These polyhedral regions are referred to here as

"cut bodies" and *"fill bodies"*.

Their tetrahedralizations provide for easy determination of cut and fill volumes.

A polygonal region in the plane can always be triangulated in such a fashion that the vertices in the triangulation are all contained in the boundary. For a non-convex polygonal region the analog does not hold (Schonhardt 1928), and additional

"Steiner points"

in the interior are needed as vertices of a tetrahedralization. Current research is aimed at finding suitable Steiner points so that tetrahedralization can proceed. It is also desired to minimize the number of Steiner points.

The tetrahedralization method currently under development at NIST, solves the problem of identifying Steiner points by starting with an initial tetrahedralization of the vertices of a triangulated surface and then inserting its edges and facets subsequently. These insertion processes identify necessary additional points as intersection points of edges and facets of the triangulated surface and the edges and facets of the initial tetrahedralization.

Computing a tetrahedralization of a polyhedral region bounded by a given closed triangulated surface is thus accomplished in three steps:

- Compute a (Delaunay) tetrahedralization of the vertices of the surface.
- Insert into the tetrahedralization the missing edges of the surface.
- Insert into the tertrahedralization the missing triangles of the surface.

These steps will be outlined in the following sections.

The insertion processes listed above can also be used for finding intersections of two polyhedral regions bounded by closed triangulated surfaces, referred to here as "first" and "second" surface. To start, the corresponding first and second polyhedral regions are tetrahedralized. Then it will be necessary to determine the intersections of facets of the first surface with facets of the second, and vice versa. Finding the intersection of two triangulated surfaces is greatly facilitated by the data structures of their tetrahedralizations. In general, that intersection consists of straight line segments, which are to be inserted into both tetrahedralizations. The next steps involve inserting into the first tetrahedralization those edges of the second which intersect tetrahedra of the first tetrahedralization, and vice versa. Finally, the analogous insertions of facets will have to be accomplished.

As mentioned earlier, the determination of cut and fill volumes defined by two closed triangulated surfaces, essentially requires determining the intersection of two tetrahedralized regions.

5.1 Delaunay Tetrahedralizations

Just as there are many ways to triangulate a set of planar points, there exist many tetrahedralizations of a given set S of data points in space. In fact, if only set S has to be tetrahedralized, then this can be done in several ways even without additional Steiner points as vertices v_j. In analogy to Delaunay triangulations,

"Delaunay tetrahedralizations"

are defined by the following (Delaunay 1934):

(5.1.1) **Empty Sphere Criterion** *No tetrahedralization vertex v_j lies in the interior of the circumsphere of any tetrahedron t_k of the tetrahedralization.*

The tetrahedra of the Delaunay tetrahedralization of a set S precisely fill the convex hull of S. Delaunay tetrahedralizations are again uniquely determined if no circumsphere of any of their tetrahedra contains more than four vertices on its periphery.

As in the planar case, an insertion method is available for computing Delaunay tertrahedralizations point by point. Each time a point is inserted into the interior of a tetrahedron, that tetrahedron is divided into four tetrahedra, each having the new point as a vertex and sharing one of its facets with the original tetrahedron. Similar subdivisions are generated if the new point falls on a facet or an edge of the current tetrahedron. In any of these cases, a new tetrahedralization results which now includes the inserted point as vertex but may no longer satisfy the Delaunay criterion (5.1.1). An adjustment process for reestablishing the Delaunay property is outlined in what follows. It takes the form of repeated

"flips".

Such flips may be carried out in any of the three situations described below.

Suppose two tetrahedra t_1 and t_2 of the tetrahedralization, share a facet f with vertices v_3, v_4, v_5. Vertex v_1 belongs to t_1 but not to t_2, whereas vertex v_2 belongs to t_2 but not to t_1. Then the following condition is necessary and sufficient for admitting a flip.

(5.1.2) *The interior of the line segment l connecting the vertices v_1 and v_2 intersects the interior of the common facet f.*

Flipping in this case will produce three new tetrahedra in the space of the old tetrahedra t_1, t_2. The line segment l with endpoints v_1, v_2 is now an edge in the adjusted tetrahedralization and is shared by all three new tetrahedra. The facet f, which belonged to the previous tetrahedralization, is no longer represented in the new one, but the edges of f still are. Each of the new tetrahedra has an outward facet in common with the old tetrahedron t_1 as well as t_2, and each new tetrahedron contains a different single edge of the relinquished facet f (Fig. 5.1).

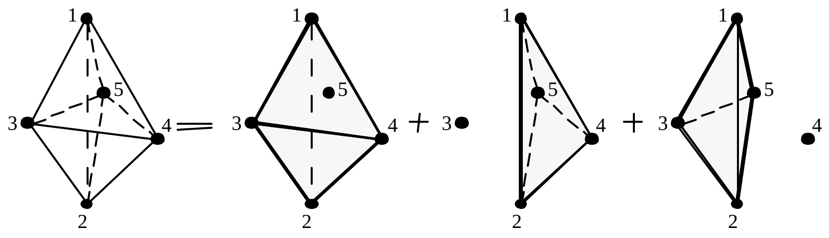


Figure 5.1. A Two-for-Three Flip.

For the second case, consider three tetrahedra t_1, t_2, t_3 which intersect in a common edge l connecting vertices v_1 and v_2, both of which belong to every one of the three tetrahedra. The remaining vertices v_3, v_4, v_5 in that configuration are shared by two of the tetrahedra, respectively. Then a flip is possible if and only if the following condition is satisfied.

(5.1.3) *The interior of the triangle f spanned by vertices v_3, v_4, v_5 intersects the interior of the edge l with endpoints v_1, v_2.*

Flipping in this case will produce two new tetrahedra in the space of the three old tetrahedra t_1, t_2, t_3. The triangle *f* with vertices v_3, v_4, v_5 is now a triangle in the adjusted tetrahedralization and is shared by the two new tetrahedra. The edge *l*, which belonged to the previous tetrahedralization, is no longer represented in the new one. Each of the two new tetrahedra has a different single facet in common with each of the three previous tetrahedra (Fig. 5.2). This flipping operation can be considered as the inverse of the one described above.

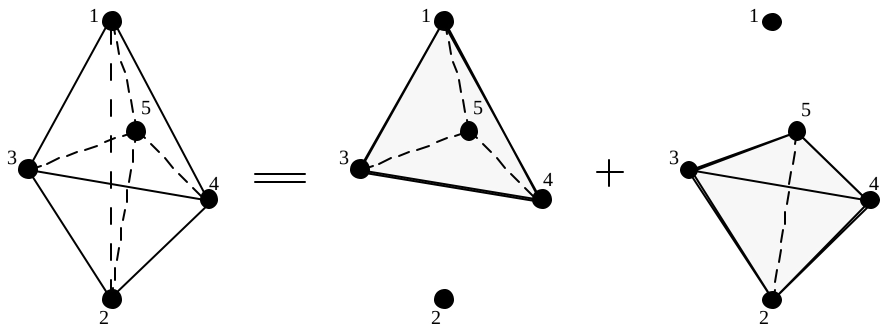


Figure 5.2. A Three-for-Two Flip.

Finally, consider two adjacent facets f_1 and f_2 with vertices v_3, v_4, v_5 and v_3, v_4, v_6, respectively, and four teratrahedra

$$t_1 = \{v_1, v_3, v_4, v_5\},\ t_2 = \{v_2, v_3, v_4, v_5\},\ t_3 = \{v_1, v_3, v_4, v_6\},\ t_4 = \{v_2, v_3, v_4, v_6\}$$

with t_1 and t_2 adjacent at facet f_1, and t_3 and t_4 ajacent at facet f_2. All four tetrahedra have the edge *e* with endpoints v_3 and v_4 in common. A flip is again possible under the following condition, which however will occur only in the unlikely case in which four or more data points are coplanar.

(5.1.4) *The facets f_1 and f_2 lie in a common plane so that the four vertices v_3, v_4, v_5, v_6 form a planar quadrangle, and the interior of its diagonal $d = \{v_5, v_6\}$ intersects the interior of the other diagonal $e = \{v_3, v_4\}$.*

Flipping in this case will amount to a diagonal interchange in quadrangle $q = \{v_3, v_4, v_5, v_6\}$ analogous to the triangulation adjustment in the plane. Four new tetrahedra replace the original four tetrahedra. The new tetrahedralization will contain diagonal *d* as a common edge of all four new tetrahedra, replacing edge *e* (Fig. 5.3).

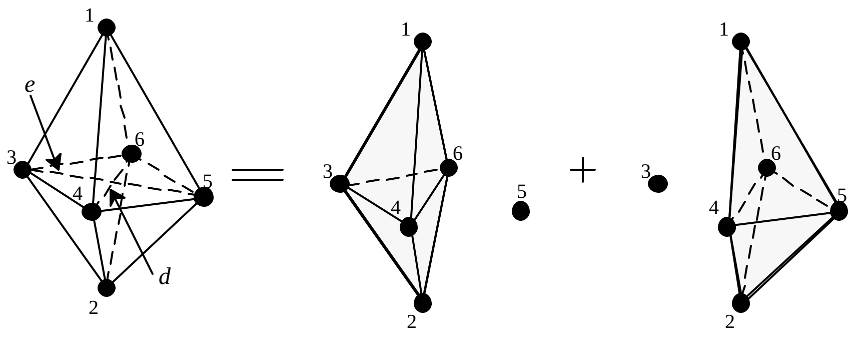


Figure 5.3. A Four-for-Four Flip.

The concept of opposite vertices of triangles in a triangulation extends naturally to that of opposite vertices of tetrahedra, and again according to Lawson (1986),

(5.1.5) *if the empty sphere criterion holds for opposite vertices, then it holds universally.*

For the case of data points in general position, that is, if no four points are coplanar, Edelsbrunner and Shah (1996) have shown that as new tetrahedra are created by insertion of a new point into a Delaunay tetrahedralization or a subsequent adjustment, and as this action leads to a violation of the empty sphere criterion, it is always possible to find tetrahedra satisfying conditions (5.1.2) and (5.1.3) with one of them being a recently introduced tetrahedron. The same authors also show that a finite number of corresponding flips will result in all opposite vertices satisfying the Delaunay property. These observations and (5.1.5) form the theoretical basis of the insertion method for Delaunay tetrahedralizations.

5.2 Insertion of Line Segments and Triangles

The approach to modifying a tetrahedralization, so that a given line segment is represented in it, will now be considered. A line segment is represented if, in the modified terahedralization, it is either a single edge or the union of several edges. It is assumed that the endpoints of the line segment are already vertices of the tetrahedralization, having been inserted into the Delaunay tetrahedralization of the set of data points.

Additional vertices are added whenever the line segment intersects, in a single point, the interior of a facet f or, in degenerate cases, the interior of an edge e of the tetrahedralization. In the first case, each of the two tetrahedra sharing facet f are split into three tetrahedra as follows. The triangular facet f is split into three triangles. Each of those triangles together with the vertex v opposite to facet f in the original tetrahedron determines one of the three new tetrahedra. The three new tetrahedra share a common edge whose endpoints are the vertex v and the inserted vertex $\tilde{v}$. In the second case, each tetrahedron containing edge e is split into two tetrahedra. Here, the edge e is split by the inserted vertex $\tilde{v}$ into two line segments, each of which becomes an edge of a new tetrahedron. The two new tetrahedra share a new facet spanned by the new vertex $\tilde{v}$ and the old edge $\bar{e}$ that was opposite edge e in the original tetrahedron. The resulting tetrahedralization is no longer expected to be Delaunay.

By executing suitable flips before actually inserting the line segment, it is possible to reduce the number of facets intersected, and thereby the number of vertices to be added. Such flips may take place whenever the two tetrahedra are adjacent to facet f and satisfy the flipping criteria described in the previous section. Such flips may be considered even if they do not lead to an immediate reduction of intersection points because they may create a favorable situation later on. This is particularly true in the case of "2 for 3" flips.

For tasks of modifying the tetrahedralization so that a given triangle is represented in it as a facet or a union of facets, it is assumed that the vertices and edges of the triangle are already contained in the given tetrahedralization since the corresponding points and lines have been inserted in the two earlier phases of the approach. If the triangle is the union of triangles in the tetrahedralization, then it is already represented.

An additional vertex will have to be added when the interior of an edge of the tetrahedralization intersects the interior of the triangle in a single point. Necessarily, such a point will have to become a vertex of the tetrahedralization. In particular, if a tetrahedron with its interior intersects the triangle, then up to four of its edges intersect the triangle, resulting in up to four new vertices, which will be inserted one by one. Each such insertion is an insertion of a point into an edge e of the prevailing tetrahedralization. As pointed out in the context of inserting line segments, this requires splitting each tetrahedron containing edge e into two tetrahedra.

Again carrying out suitable flips before insertion may reduce the number of additional points to be added as vertices. Such a flip may take place when two tetrahedra are adjacent along a facet whose interior intersects the triangle and the two tetrahedra satisfy the condition (5.1.2) or (5.1.3).

5.3 Tetrahedralization Codes

Two NIST programs, *vol3tet* and *vol3cmp*, have been developed for computing volumes by tetrahedralization. A third program *vol3tri*, also developed for volume computation, is limited to current applications where data points define elevated surfaces. All three programs are written in FORTRAN 77 and the source code for these programs are included in the provided disk. Program *vol3tet* computes a Delaunay tetrahedralization for the set of points in space that is the union of the set of vertices of three closed triangulated surfaces. Two of those surfaces represent an ordered pair of surfaces with respect to which cut and fill bodies are defined. The third surface encloses a specified region of interest. Only those portions of cut and fill bodies that fall into that region are assumed to contribute to the volumes to be calculated.

Program *vol3cmp* inserts the edges and triangles that align the three surfaces with the tetrahedralization obtained with program *vol3tet*, and identifies the cut and fill tetrahedra within the region of interest. That program also collects those tetrahedra into their separate connected cut and fill bodies.

Finally, program *vol3tri* is used to create a Delaunay TIN surface from given *x,y,z*-points. Program *vol3tri* is used especially for elevated surfaces, and is executed in this case after program *vol3tet* and before program *vol3cmp*.

When determining cut and fill volumes in the entire region of interest, the identification of the individual cut and fill bodies provides valuable additional information. When planning, for instance, to excavate a construction site, that identification permits determining advantageous ways to utilize removed material for filling. A different application in the context of the demonstration will be described in the following section.

5.4 Tetrahedralization for Sand Pile Volumes

The first application of the above programs in the context of the demonstration is to determining the volume of the undisturbed sand pile for a sequence of elevation levels. The outcome will be compared against the previous results from program *TINvolume.*

The data provided were the same 2000 *x,y,z*-points selected by the *TINvolume* routine within the cropped area. Also given was the boundary of the planar area of relevance as determined by that program. This will determine the spatial region of interest to which the volume calculation was to be confined. An ordered pair of triangulated surfaces has to be given. For this application, the first surface is the Delaunay TIN surface interpolating the 2000 data points. The second surface is a horizontal plane at the specified elevation level. The volumes of interest are the cut volumes.

The total cut volume for a specified elevation level should match the *TINvolume* result. However, this result is polluted by small cut bodies due to floor noise. The feature of program *vol3cmp* that identifies separate cut bodies now permits to single out the main cut body representing the sand pile. The actual volume given for the sand pile will thus be smaller than the total cut volume as shown in Table 5.1.

Table 5.1. Comparison of Sand Pile Volumes for 2000 TIN Points.

Elevation (cm)	Triangulation (m^3)	Tetrahedralization All Cut Bodies (m^3)	Tetrahedralization Main Cut Body (m^3)	Comments
-222	2.0669	2.0667	2.0666	

-221	**1.9460**	**1.9459**	**1.9429**	<= floor level
-220	1.8643	1.8643	1.8616	
-219	1.8034	1.8034	1.8032	
-218	1.7473	1.7473	1.7473	
-217	1.6926	1.6926	1.6926	

The first column in Table 5.1 lists the elevation above which the sand pile volume is being determined. The second column lists the results by program *TINvolume*, which is based on triangulation. The volumes in the third and fourth column have been computed using tetrahedralization. They represent total cut volume and the volume of the largest cut body, respectively.

The total cut volume, comprising the volumes of all cut bodies, is seen in perfect agreement with the triangulation-based volumes arrived at by using program *TINvolume*. The volume of the single cut body corresponding to the sand pile is perceptively smaller, and is considered here to be the more exact value.

In the second part of the sand pile demonstration, a portion of the sand pile was scooped up, and the removed volume had to be determined. Again scan data were the basis for this determination. In particular, data from a single scanner in identical position were collected before and after the sand removal. The scanner was positioned in such a fashion, that all changes in the surface of the sandpile were observable by that single scanner. This allowed the computation of the volume difference from before-and-after scan data from that single position. No registration adjustments were necessary because the scanner position had not changed.

The common area of relevance is the result of an option in program *TINvolume*. Again, three surfaces have to be specified. The area of relevance is utilized to define the closed surface which encloses the region of interest. The Delaunay TIN surface from the scan data for the undisturbed sand pile is similarly extended to yield the first closed triangular surface. The Delaunay TIN surface from the scan data for the disturbed sand pile, -- TIN is constrained to the boundary of the area of relevance -- produces the second surface. The amount of volume removed equals the difference between volumes of the two surfaces with respect to an arbitrary floor level:

$$\text{volume of scoop} = 0.170 \text{ m}^3.$$

5.5 References

Delaunay, B., Sur la sphere vide, Bull. Acad. Sci. USSR (VII), Classe Sci. Mat. (1934), 793-800.

Edelsbrunner, H., and N.R. Shah, Incremental topological flipping works for regular triangulations, Algorithmica 15 (3), 223-241 (1996).

Lawson, C.L., Properties of n-dimensional triangulations, Computer Aided Geometric Design 3, 231-246 (1986).

Schonhardt, E., Uber die Zerlegung von Dreieckspolyedern in Tetraeder, Mathematische Annalen 98, 309-312 (1928).

6.0 Summary, CONCLUSIONS and future research needs

6.1 Summary and Conclusions

This report summarizes the initial efforts at NIST in developing a non-intrusive scanning method for providing construction assessment and updates. The objectives of the project are to utilize new scanning technologies to improve critical construction status assessment needs by making these measurements faster and cheaper than traditional methods and to develop, in conjunction with industry, standard means for transmission and interpretation of such data.

It has been demonstrated that a scanning laser, described in Chapter 2, is a potentially useful tool to track changes of amorphous components such as terrain. The sand pile demonstration to a live audience illustrated that “real-time” construction progress updates were possible. Information about terrain changes in one location was sent via wireless Ethernet to another location for display and to perform cut/fill volume calculations. From these initial efforts, we anticipate the development of automated information transfer protocols for the uplink of both construction site terrain data as well as discrete component locations. This work is concurrently conducted by other NIST researchers [Pfeffer, 1999].

Post-processing tools for the data registration, display, and volume calculation were necessary and were developed in-house. One set of tools, described in Chapter 3, involves using a Data Explorer program that allows for the manual registration of several data sets, 3-D data display, and volume calculations. The data in the form of point cloud data is used to create a regular grid onto which a 3-D surface is mapped. The other set of tools, described in Chapters 4 and 5, involves using stand-alone Fortran programs based on Delaunay triangulation to compute the volume bounded by two surfaces.

6.2 Future Work and Research Needs

Based on the encouraging results of the past year’s effort, the scanning technology will be used to track the progress and to document the construction of the Emissions Control Facility for Building 205 at NIST which is slated to begin in the fall of 1999. This effort will be used to determine the viability of using a laser scanner to provide real-time construction progress and updates. As the mobile platform, described below, will not be developed in time for this project, it is anticipated that the laser scanner will be located at a stationary site and will be mounted on the roof of a building adjacent to the construction site. The selection of the location enables a good field of view of the activity and also does not interfere with the construction activity. A problem that may arise from the restricted mobility is that all changes to the terrain may not be visible from one location and scans from other locations may be needed to capture all the changes to the terrain. Also, material, equipment and vehicles that are left on the construction site when the area is scanned will be interpreted as being part of the terrain, and if they move or are moved from one day to the next, they will be interpreted as changes to the terrain.

Therefore, procedures will have to be developed to eliminate or remove this artificial terrain. These procedures fall under the field of object recognition, a field that is in its infancy. Procedures sophisticated enough to identify construction equipment and vehicles are currently unavailable and would be an area for further research. For the Building 205 project, human assisted removal of artificial terrain with the aid of photographs or video images is planned.

The next step is to develop a mobile platform for the laser scanner. This may be accomplished by mounting the LADAR unit on a heavily instrumented all-terrain vehicle (ATV) which automatically registers the LADAR data to the job site using precise GPS position and attitude reporting and a geometry transformation to adjust for non-collocated instrumentation. Registration of the vehicle-based data will be more difficult as exact positioning of the laser scanner and the relative positions of GPS antennas to the laser scanner have to be very accurate and have to be fixed. Means to rapidly verify the accuracy of the laser measurements in the field will also have to be developed.

A more robust cut/fill algorithm will also be developed that can accommodate terrain features such as undercuts and vertical surfaces and to perform automated registration of two or more point clouds. Alternative post-processing approaches to compute and display the cut/fill requirements, using commercially available software packages such as Interactive Data Language (IDL), and products from Intergraph, and AutoDesk, need to be investigated. The purpose of this process being to compare the accuracy of the results (e.g., volume calculations) of different methods and the ease of use of different methods. Verification of the existing methods of volume calculations will also have to be performed.

Additional verification of the scanner in various climatic and topographic environments is needed to determine its limitations. Climatic conditions such as bright sunshine, overcast skies, light rain, etc. Topographic conditions such as hilly terrain, relatively flat terrain, large and/or deep pits, trenches, tunnels, etc.

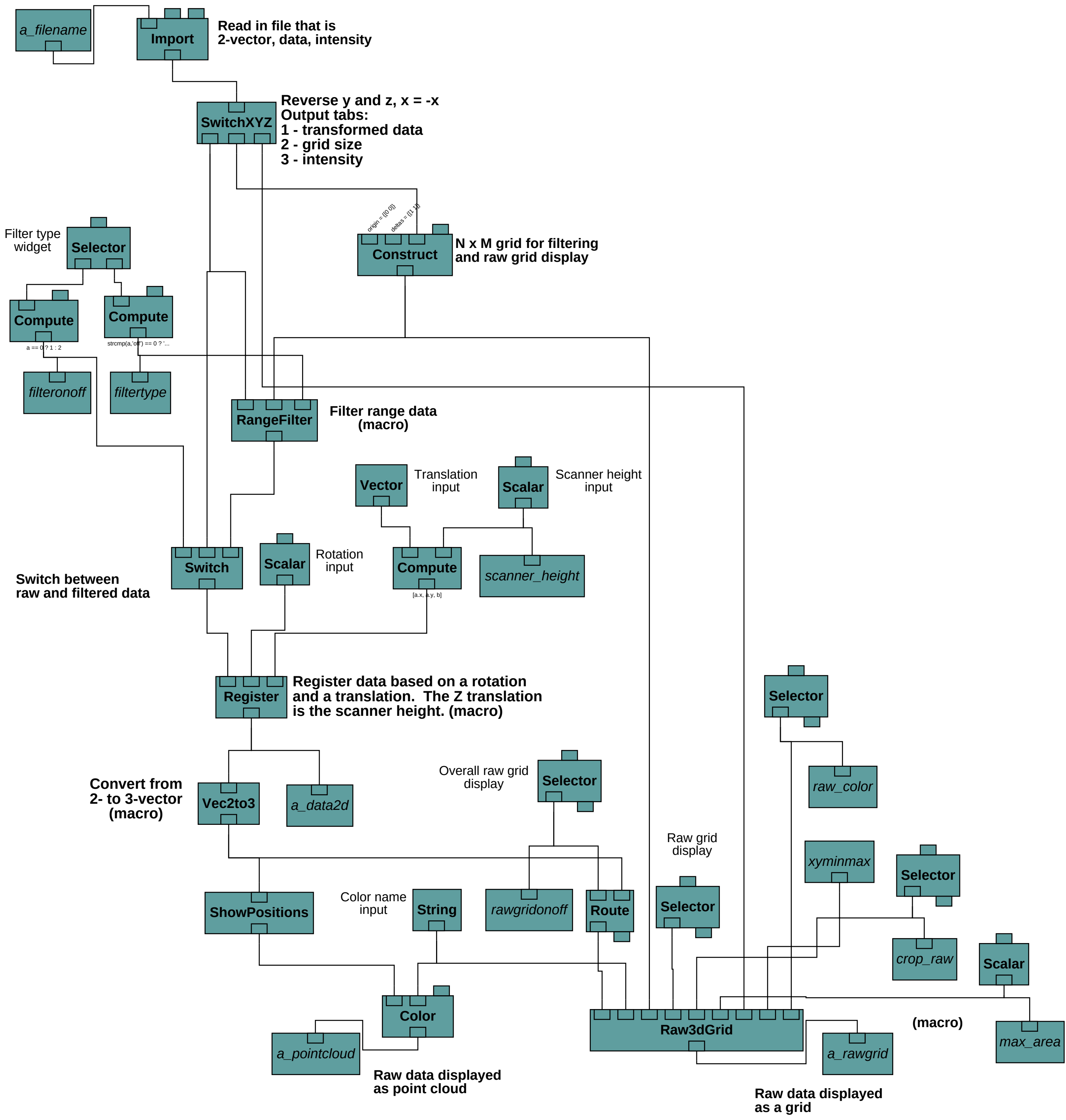
In the longer term, object recognition (human-assisted in the beginning with the end goal of full automation) will also be investigated. This will allow for the removal of unwanted objects as discussed above and for dense scanned data to be replaced by more compact 3-D model (e.g. VRML) representations. The replacement of point clouds with 3-D models will advance the goal of obtaining automated 3-D as-built models. Work in this field is being conducted at various universities, military research laboratories, private sector companies, and at NIST.

NIST is also pursuing the development of more advanced imaging devices through its "Advanced Scanner Initiative" that would allow higher scanning rates and better instrument accuracy and resolution (in the millimeter range for distances of 150 m to 500 m).

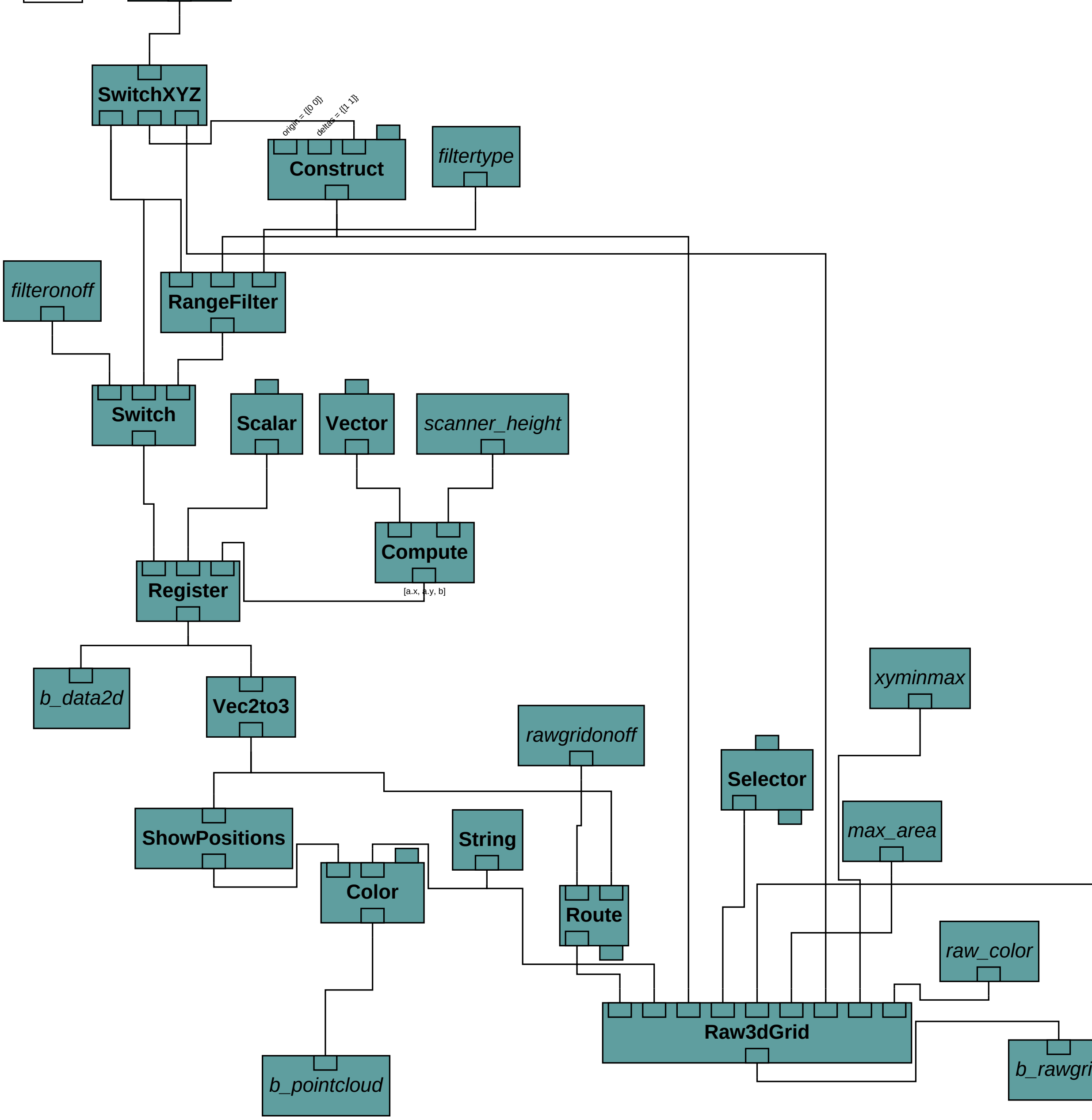
6.3 References

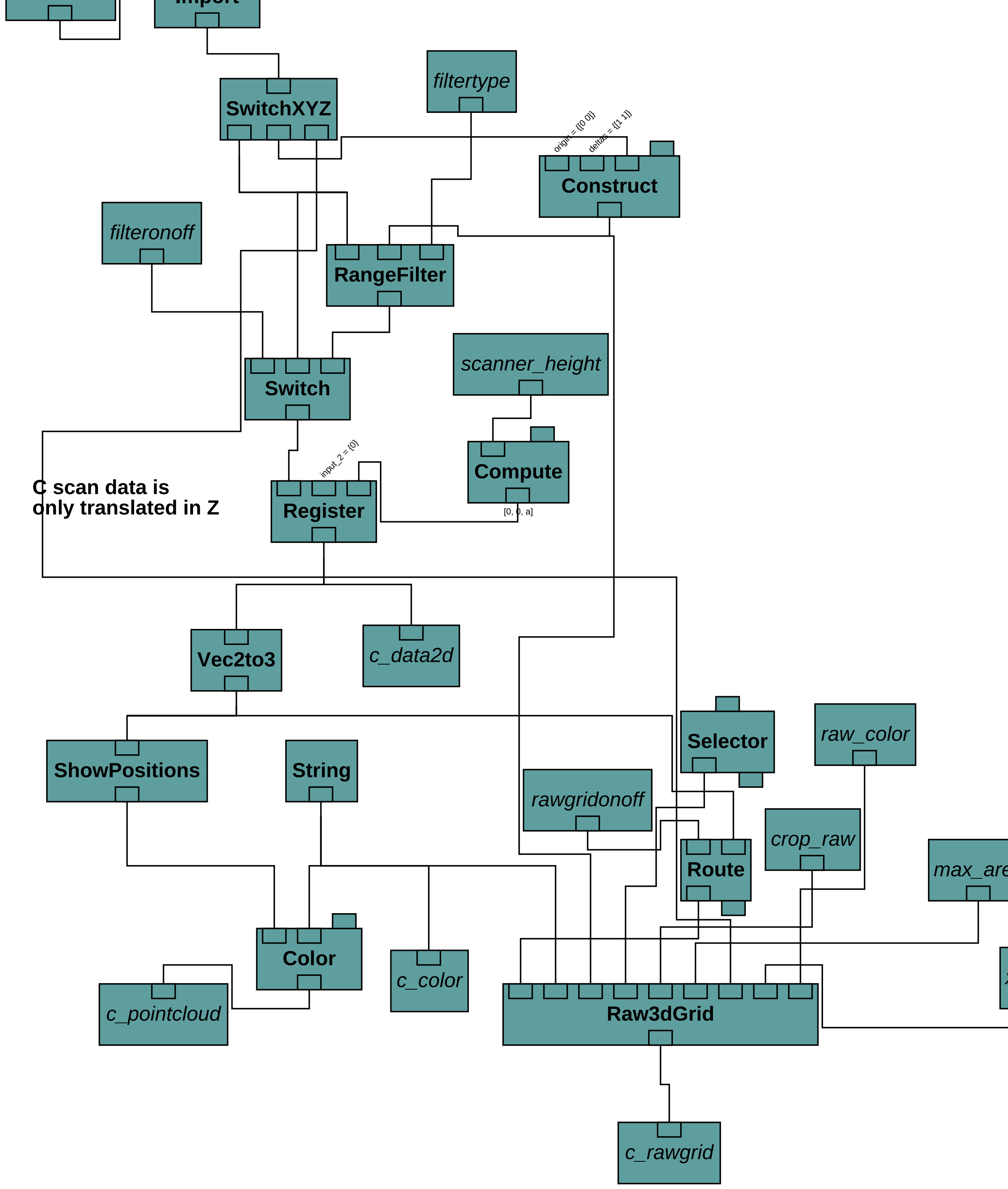
Pfeffer, L. (1999), "Mobile Sensor Platform for Construction Metrology and Automation: Design and Initial Results", *Proceedings of the 15th International Symposium on Automation and Robotics Conference* (ISARC 15), Madrid, Spain, September.

APPENDIX A: DATA EXPLORER VISUAL PROGRAM

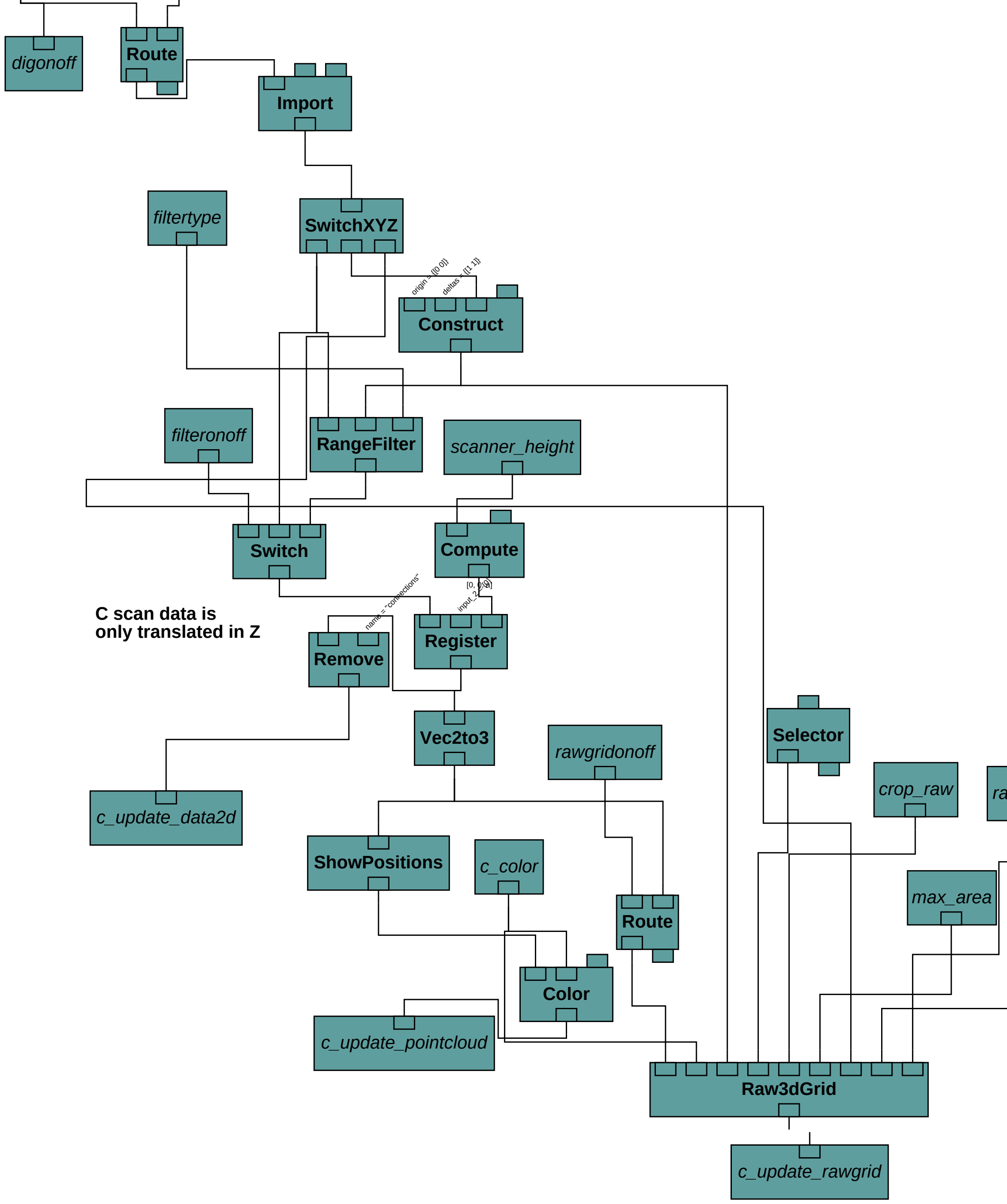


a_input - page 1 of 14


SwitchXYZ
origin = {[0 0]}
deltas = {[1 1]}
Construct
filtertype
filteronoff
RangeFilter
Switch
Scalar
Vector
scanner_height
Compute
[a.x, a.y, b]
Register
b_data2d
Vec2to3
rawgridonoff
xyminmax
Selector
max_area
ShowPositions
String
Color
Route
raw_color
Raw3dGrid
b_rawgri
b_pointcloud

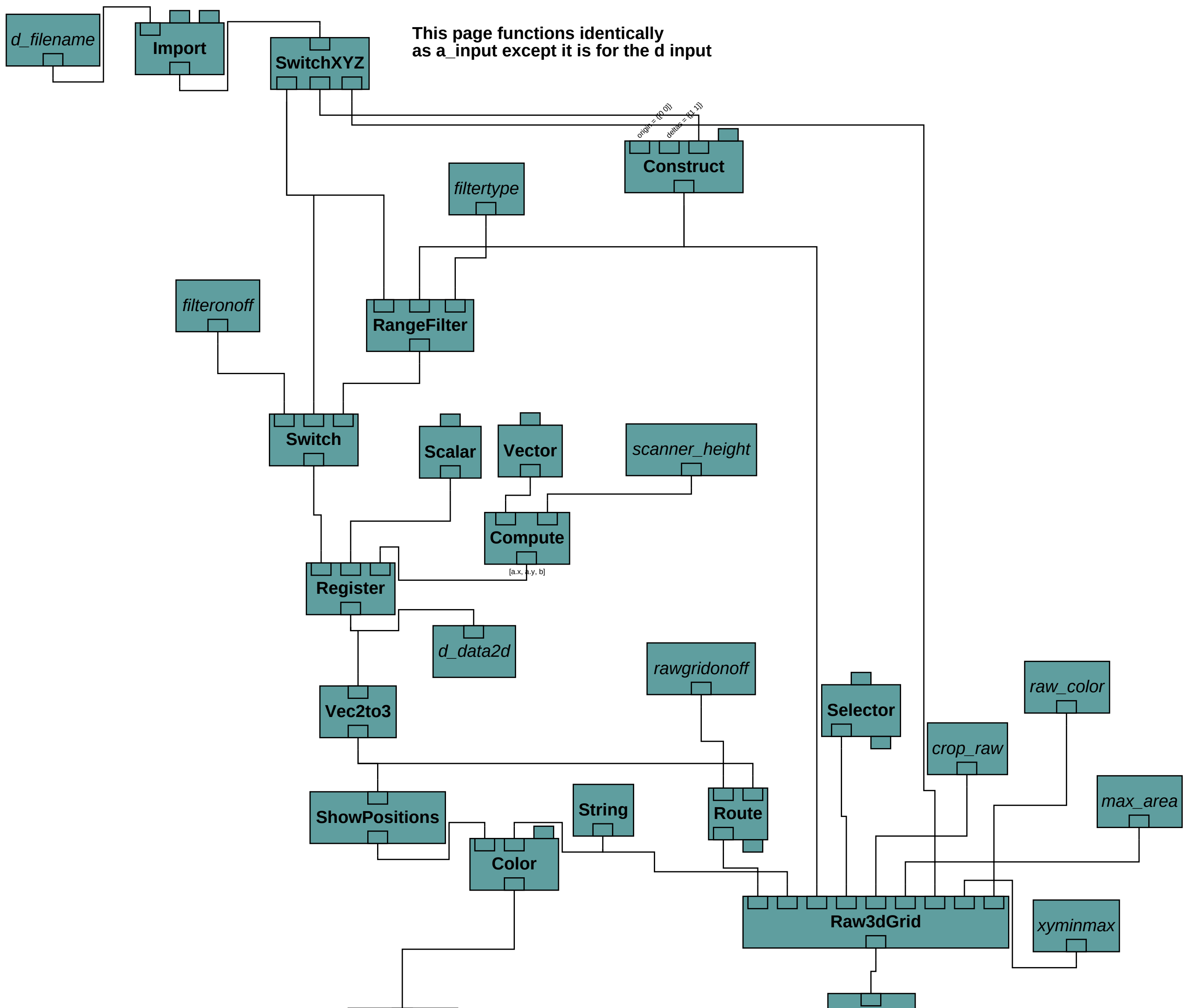


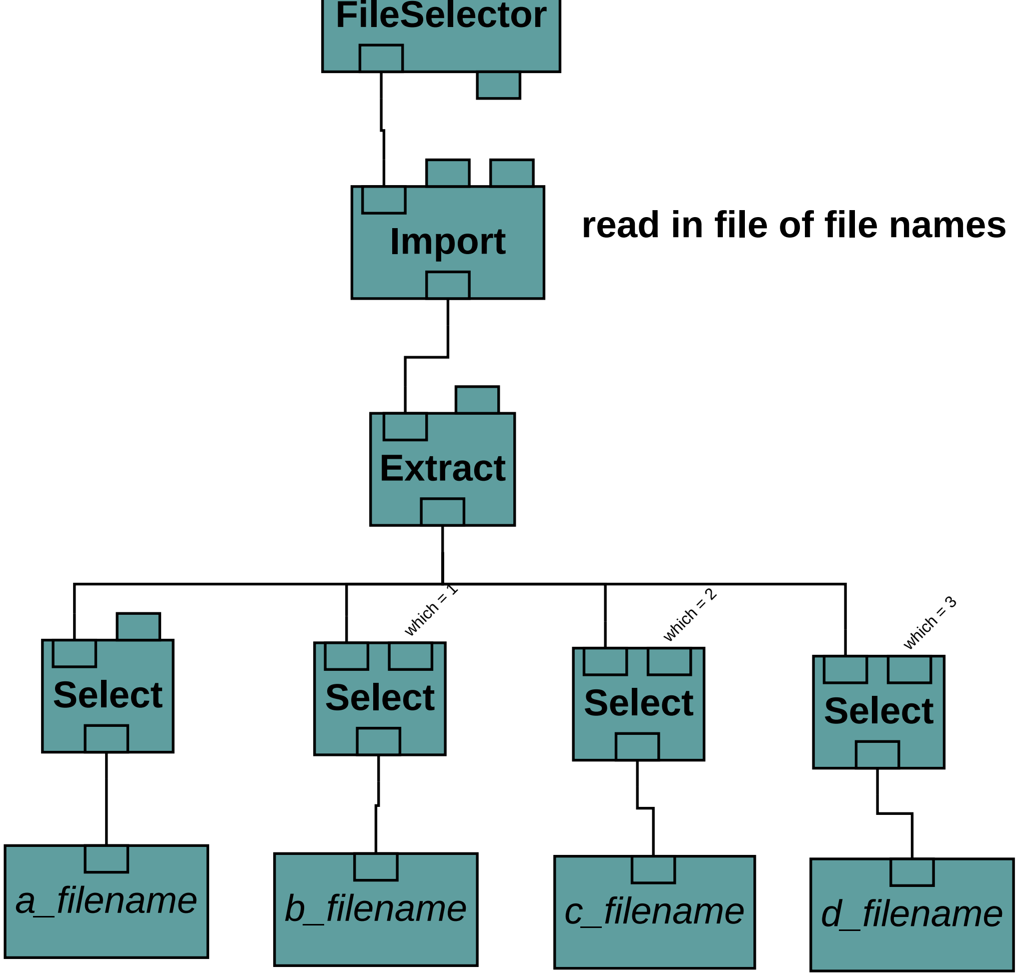
C scan data is
only translated in Z


digonoff
Route
Import
filtertype
SwitchXYZ
origin = {[0 0]}
deltas = {[1 1]}
Construct
filteronoff
RangeFilter
scanner_height
Switch
Compute
[0, 0, a]
input_2 = a
name = "connections"
C scan data is
only translated in Z
Remove
Register
Vec2to3
rawgridonoff
Selector
crop_raw
c_update_data2d
ShowPositions
c_color
Route
max_area
Color
c_update_pointcloud
Raw3dGrid
c_update_rawgrid

d_input - page 5 of 14

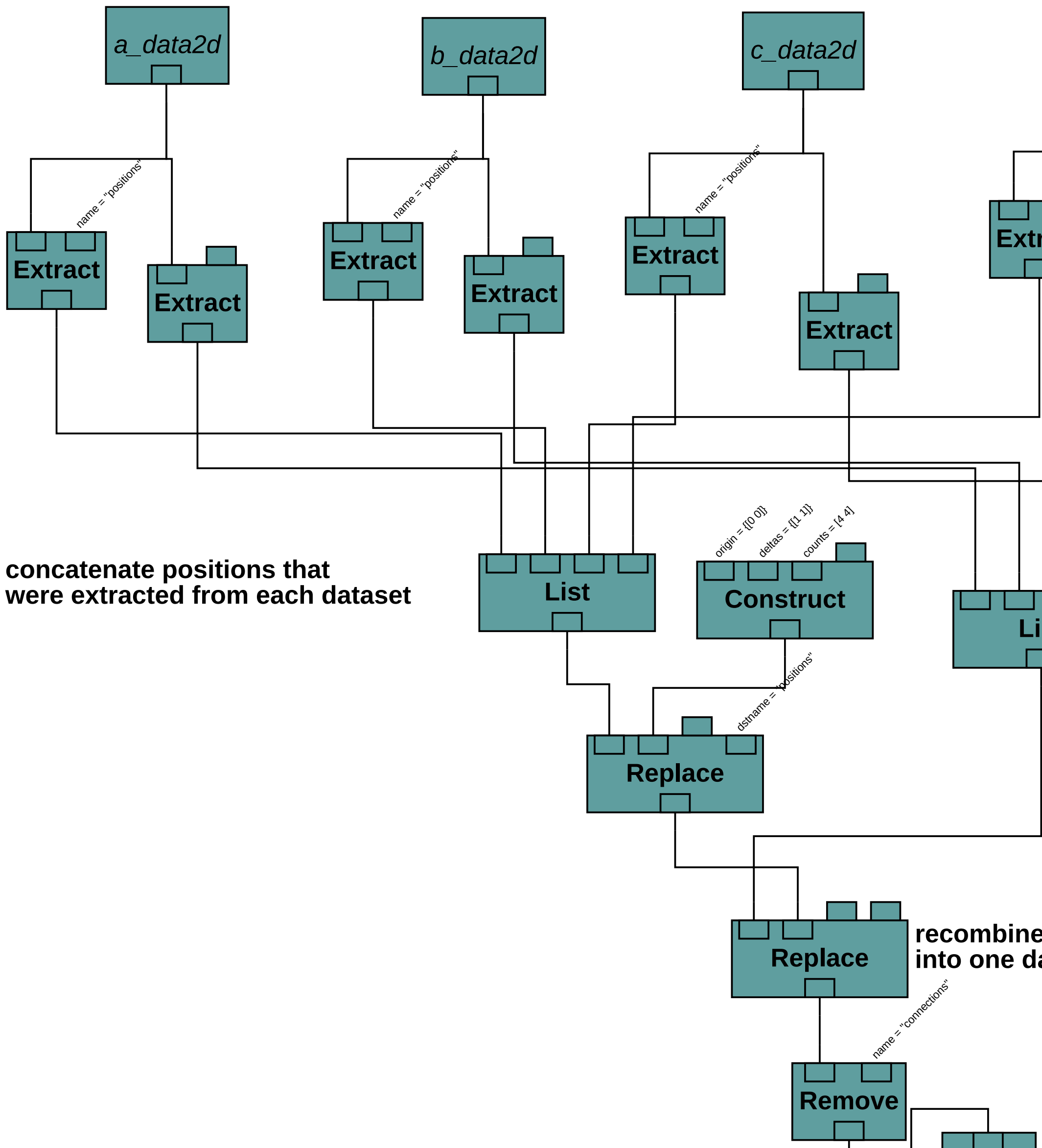
This page functions identically
as a_input except it is for the d input




FileSelector
Import
read in file of file names
Extract
which = 1
which = 2
which = 3
Select
Select
Select
Select
a_filename
b_filename
c_filename
d_filename

combine2d - page 7 of 14

combine 2d positions and data from all four datasets into one dataset



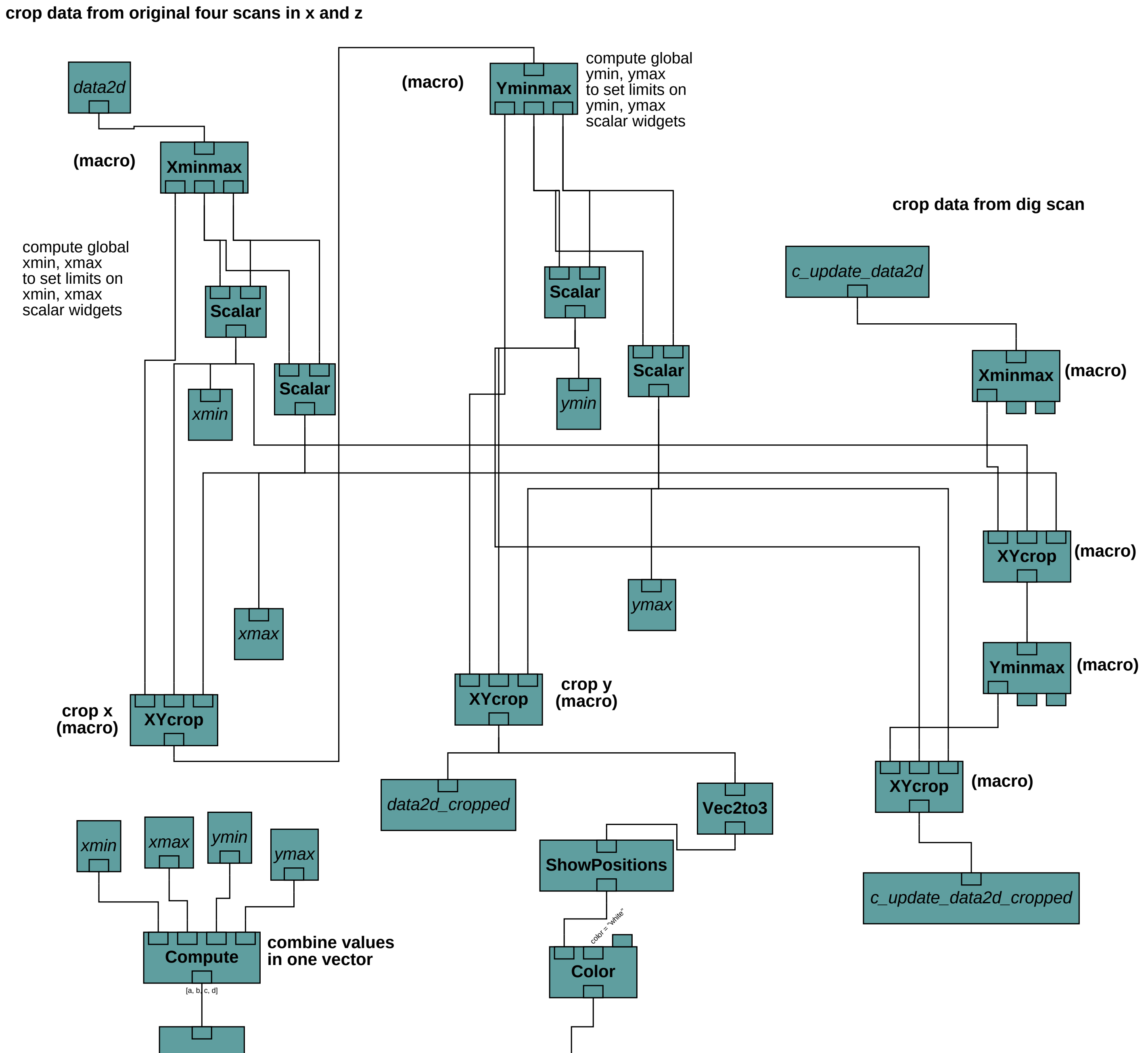
extract positions and data for each dataset

concatenate positions that were extracted from each dataset

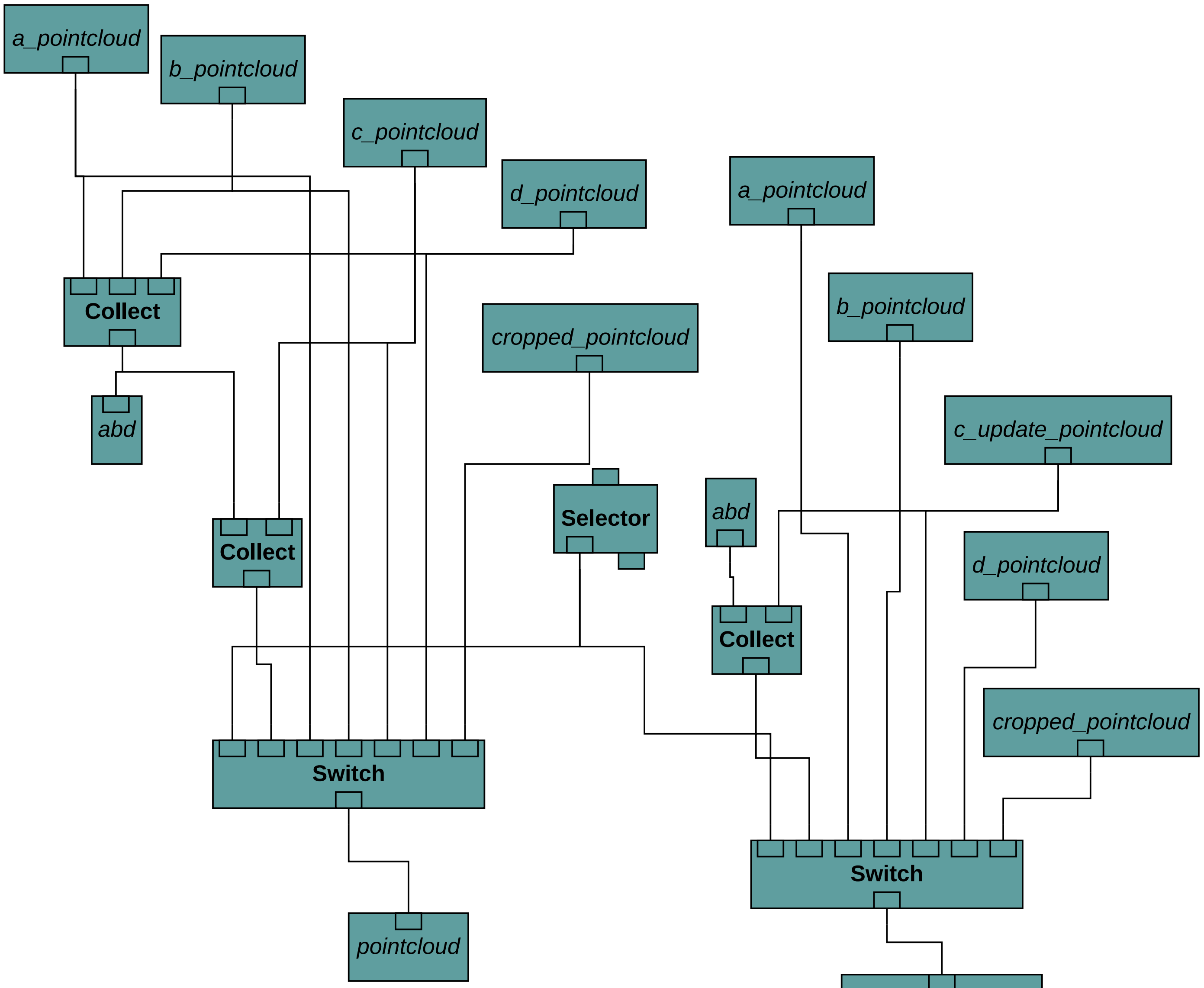
concatenate data that was extracted from each dataset

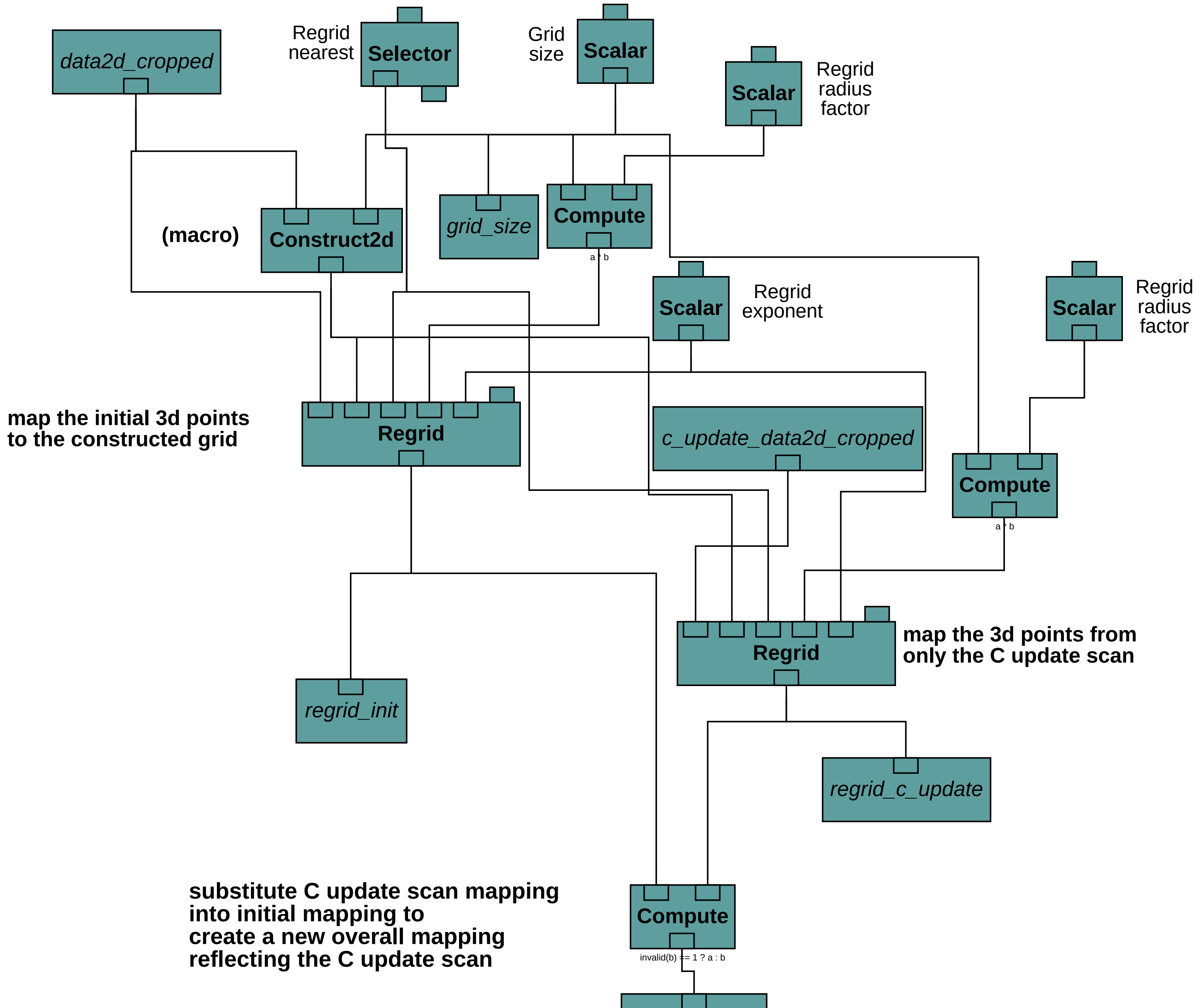
recombine positions and data into one dataset

crop2d - page 8 of 14



collect 3d point cloud data
in various combinations

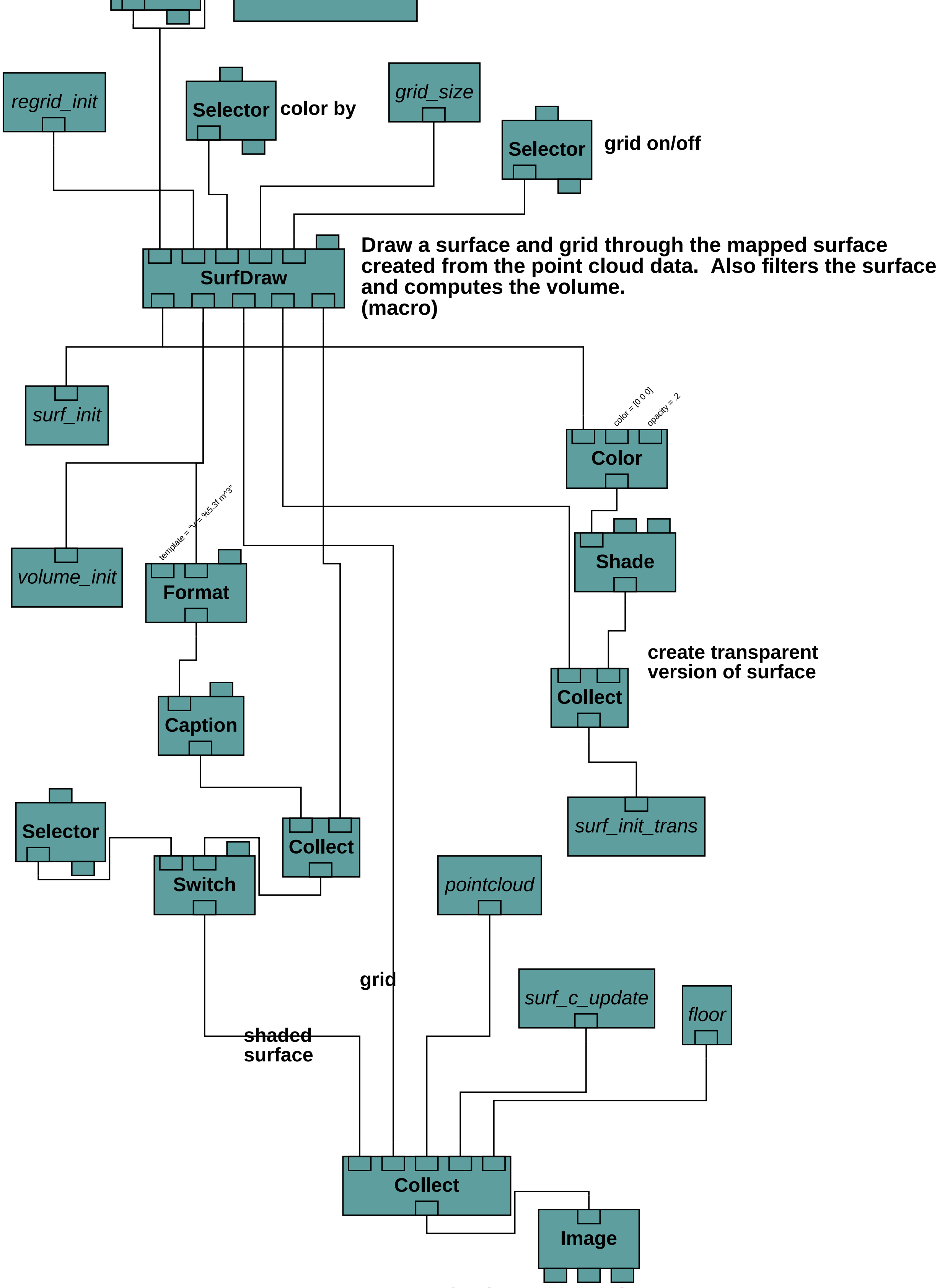


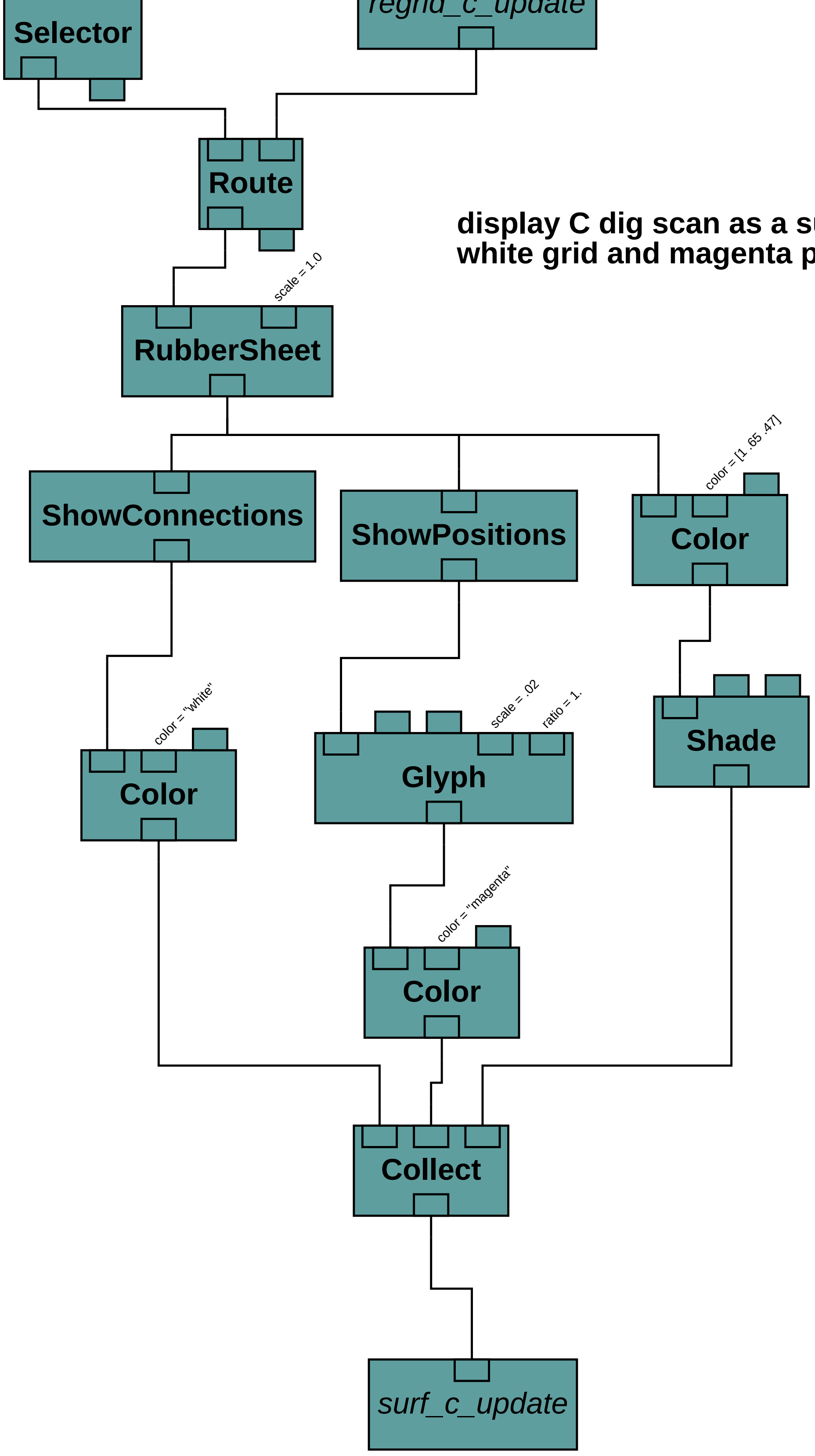


map the initial 3d points
to the constructed grid

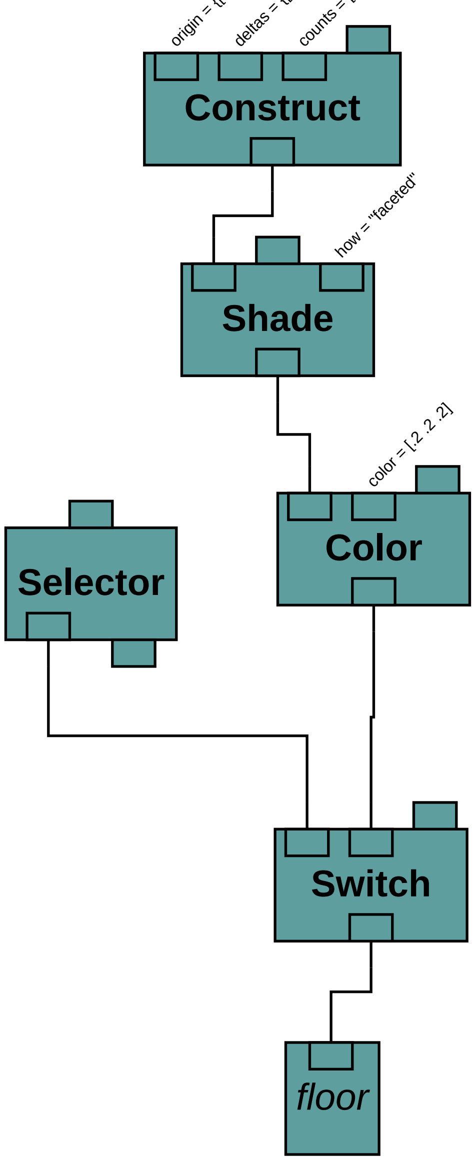
map the 3d points from
only the C update scan

substitute C update scan mapping
into initial mapping to
create a new overall mapping
reflecting the C update scan

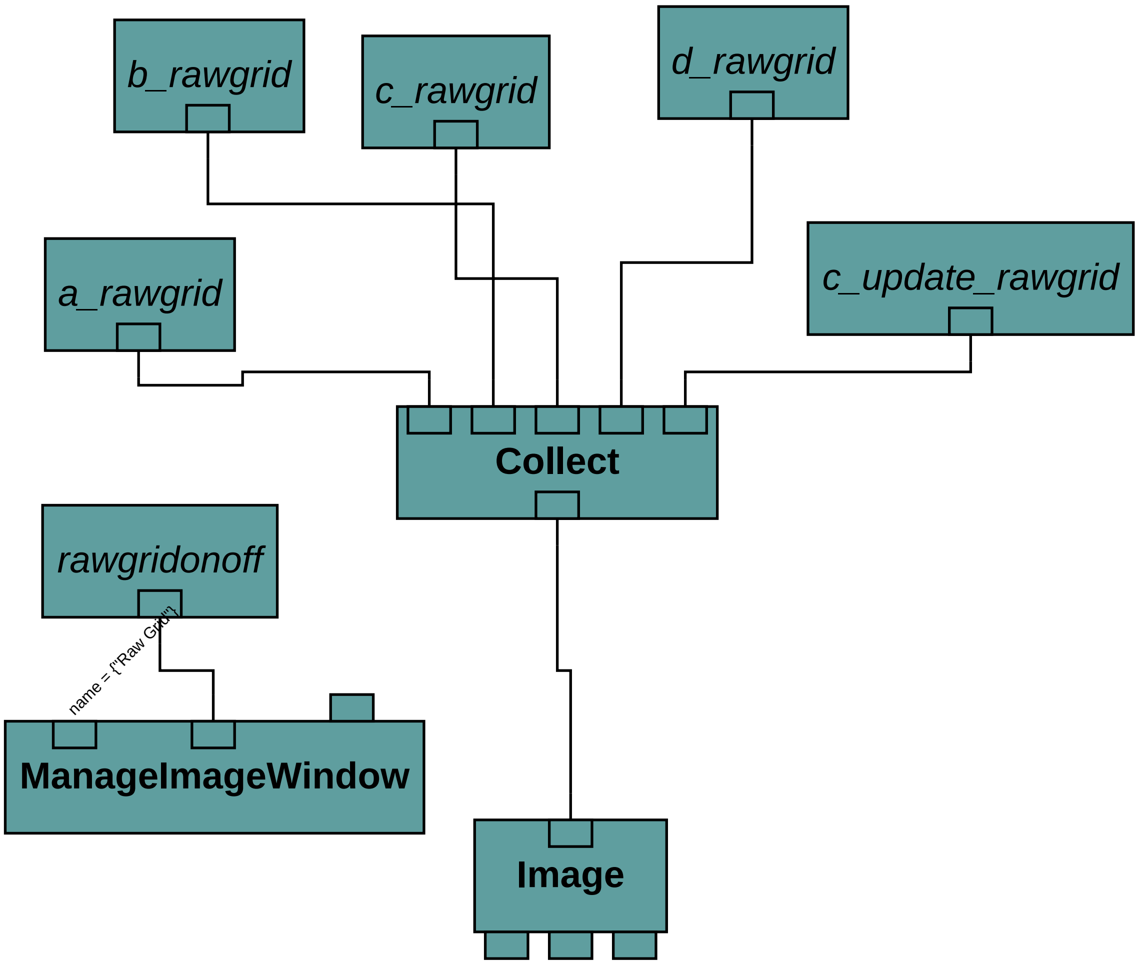

regrid_init
Selector
color by
grid_size
Selector
grid on/off
SurfDraw
Draw a surface and grid through the mapped surface created from the point cloud data. Also filters the surface and computes the volume.
(macro)
surf_init
color = [0 0 0]
opacity = .2
Color
Shade
volume_init
template = "V = %5.3f m^3"
Format
create transparent version of surface
Collect
Caption
surf_init_trans
Selector
Collect
Switch
pointcloud
grid
surf_c_update
floor
shaded surface
Collect
Image



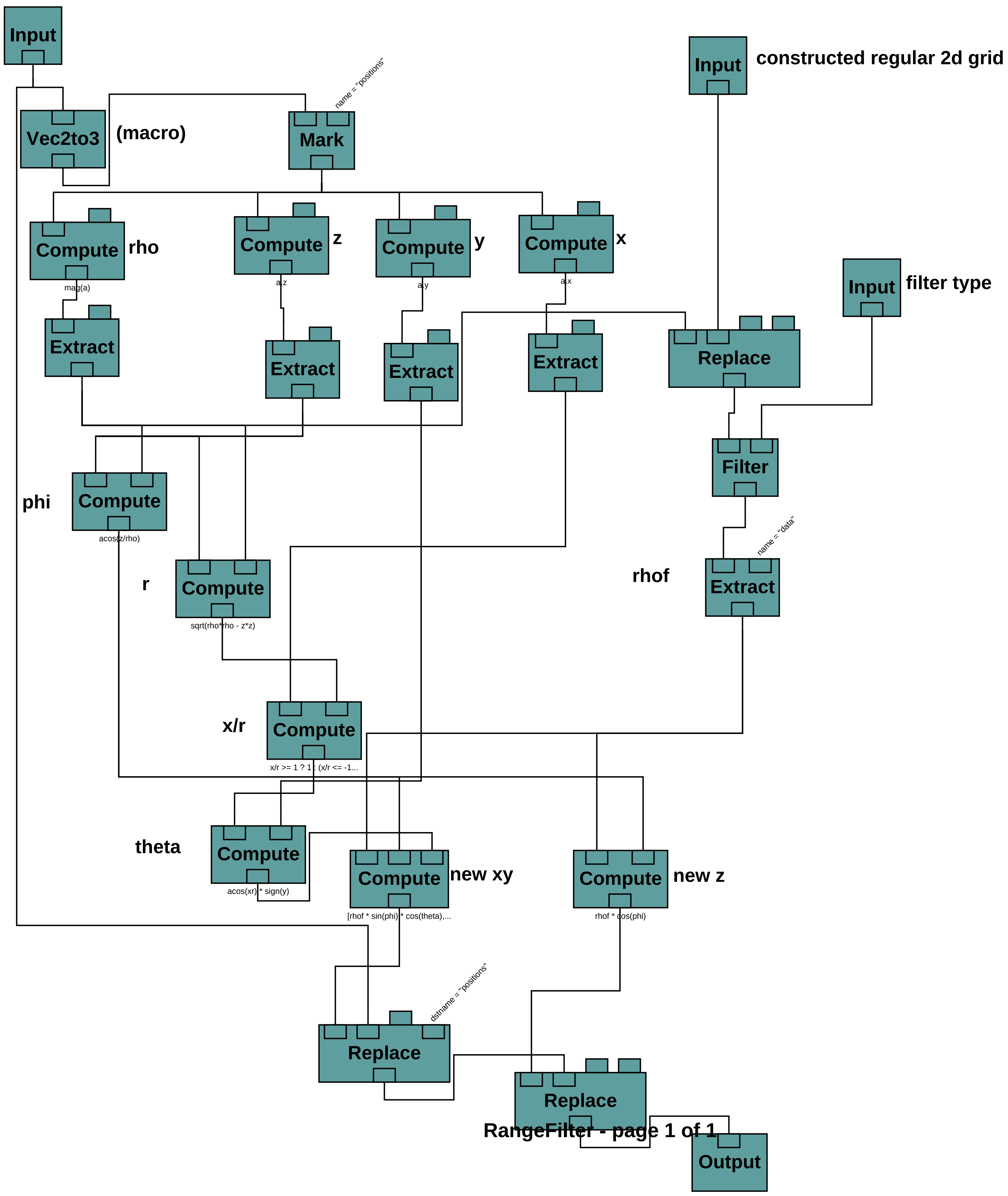
display C dig scan as a surface with a white grid and magenta points


origin = t
deltas = t
counts =
Construct
how = "faceted"
Shade
color = [.2 .2 .2]
Color
Selector
Switch
floor

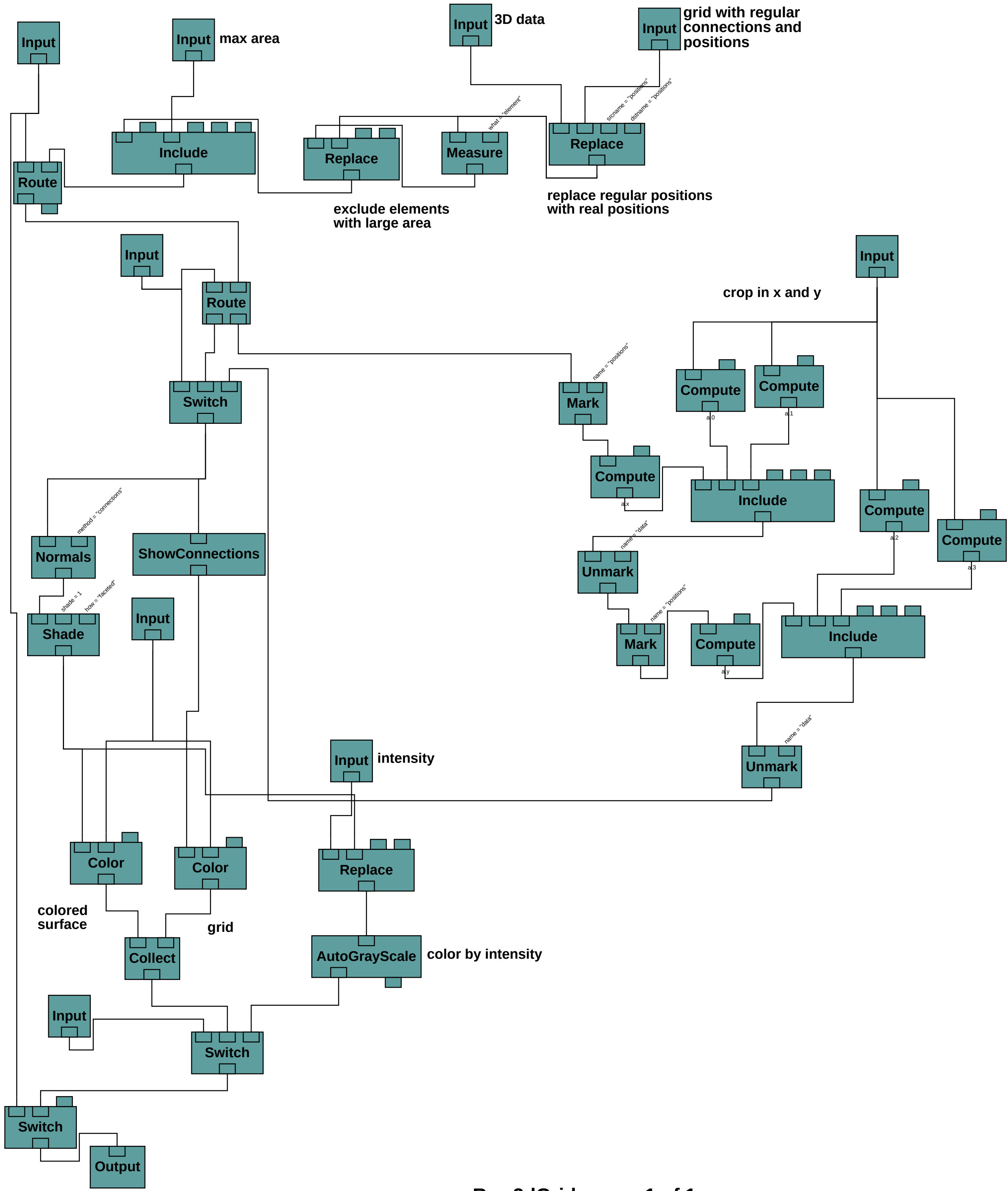
display raw grid



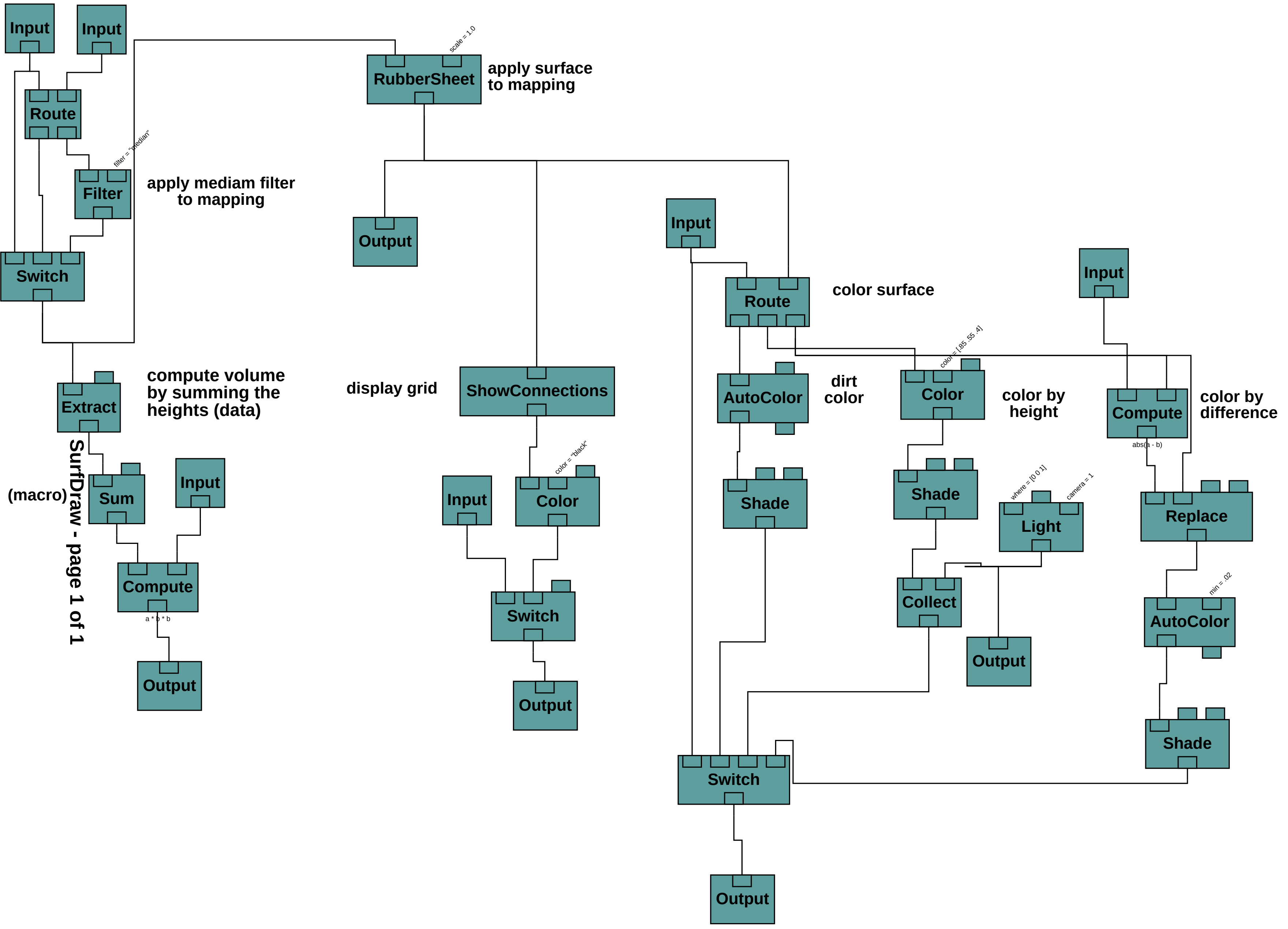
RangeFilter macro filters the range data. The point cloud data is in cartesian coordinates (XYZ) and has to be converted to spherical (theta,phi,range) so that the range can be filtered. The new data is converted back to cartesion coordinates.



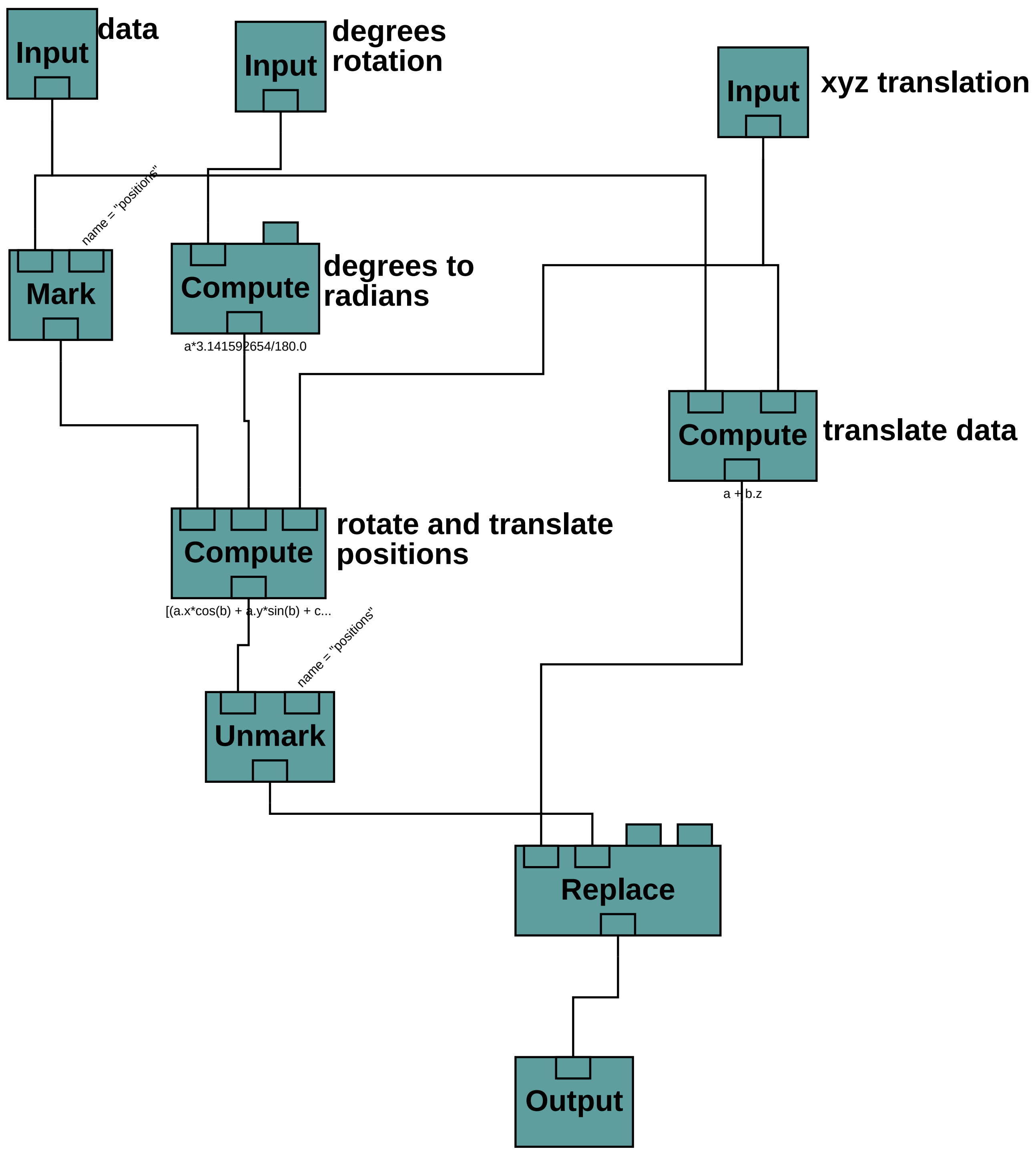
Raw3dGrid - Display the raw point cloud data as a 3D grid



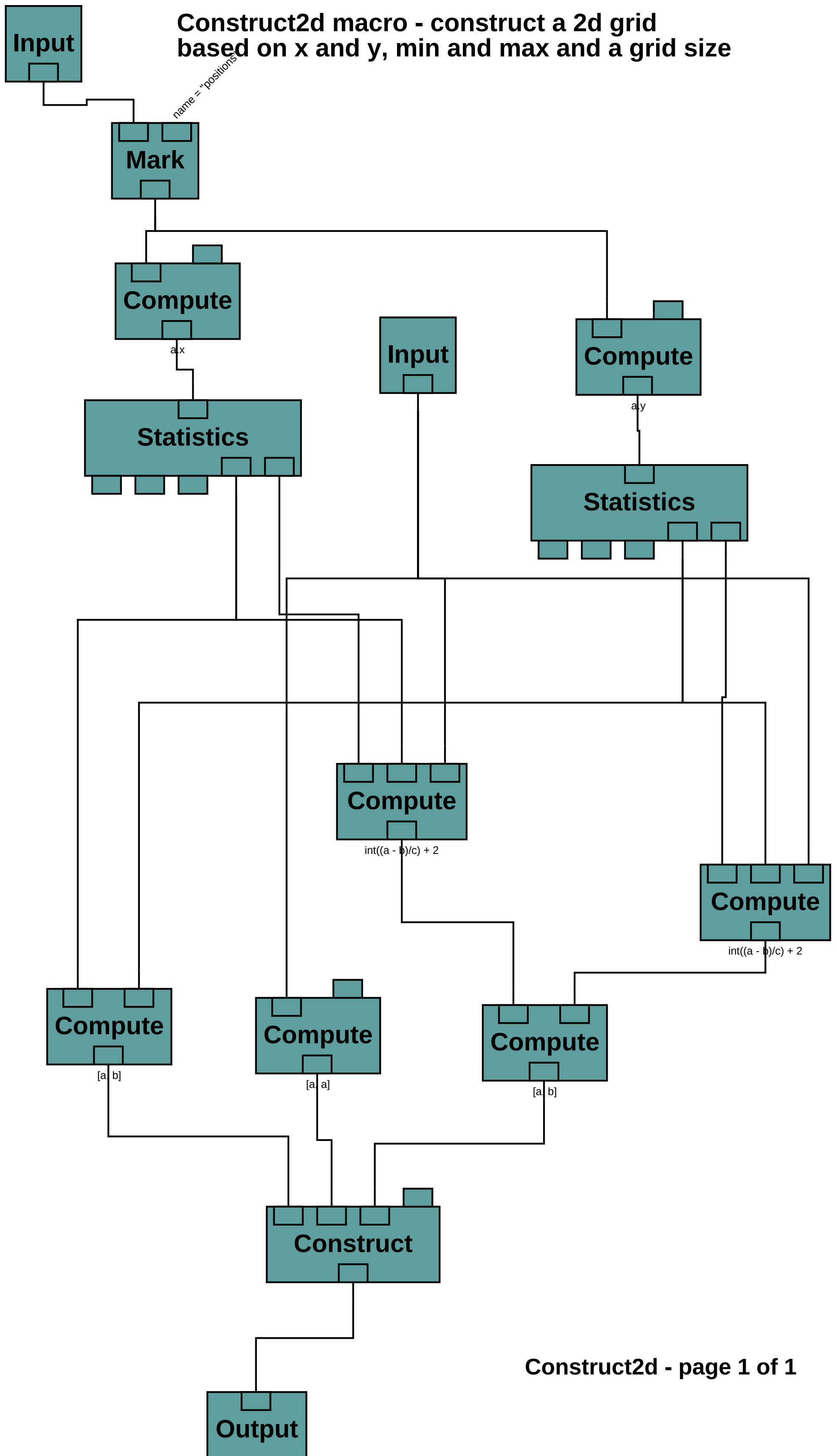
SurfDraw macro - generates surface and grid,
filters surface and computes volume



Register macro - Apply rotation and translation to point cloud data

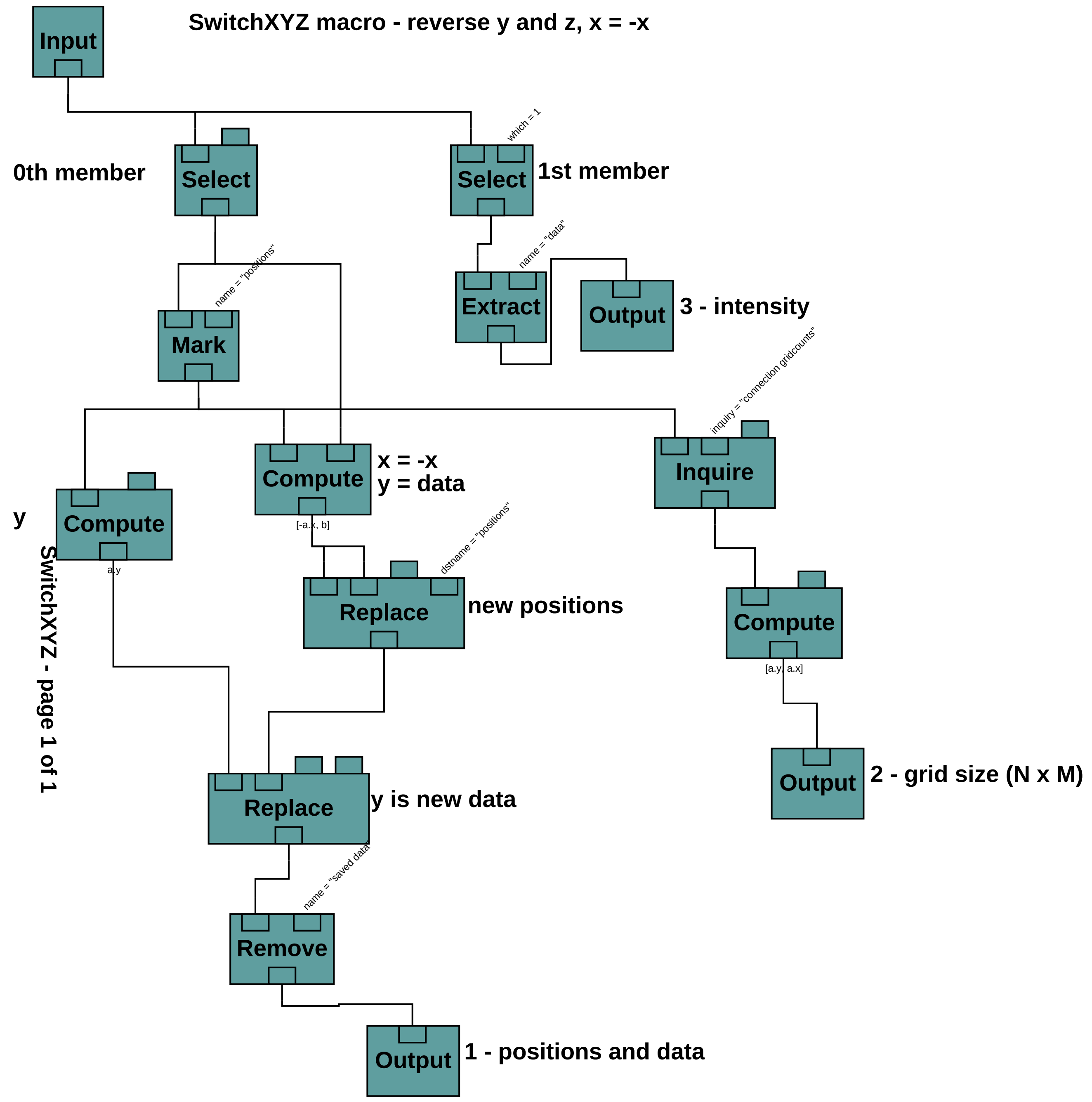


Construct2d macro - construct a 2d grid
based on x and y, min and max and a grid size

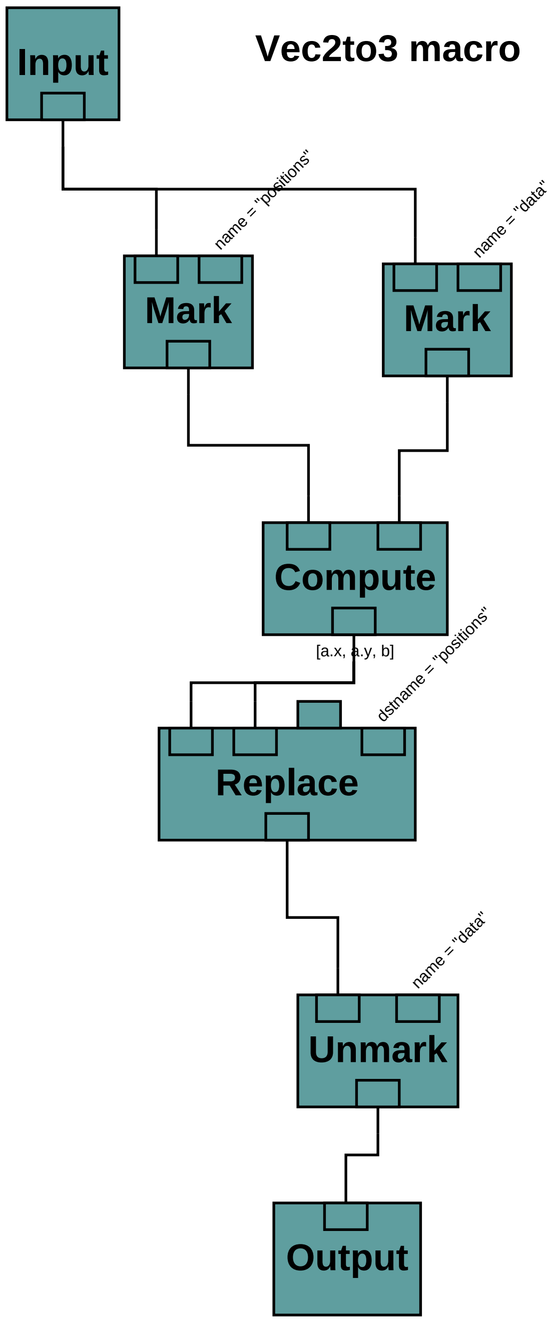


Construct2d - page 1 of 1

SwitchXYZ macro - reverse y and z, x = -x

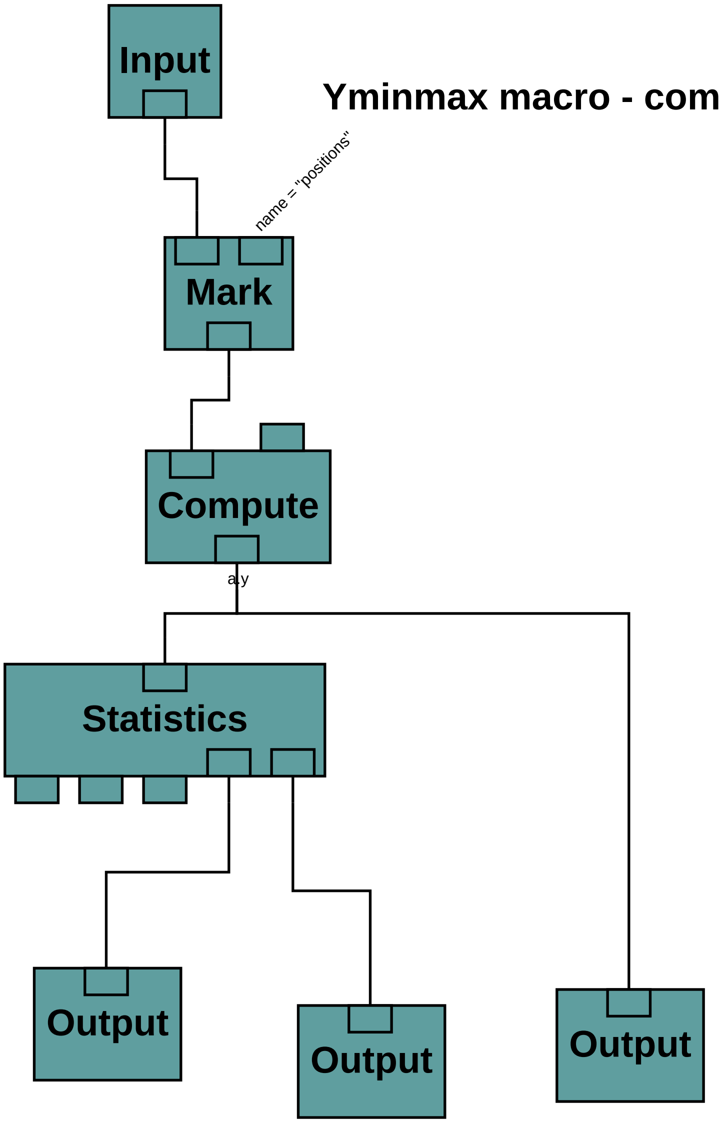


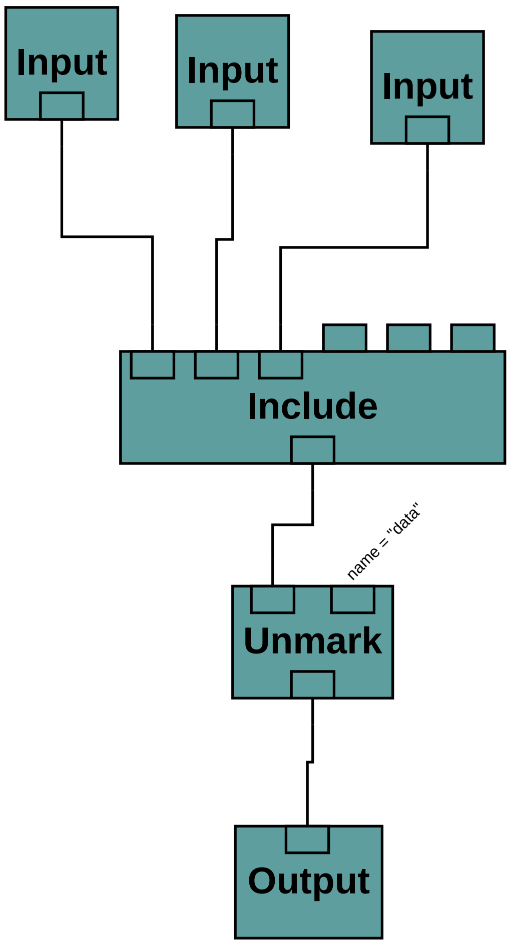
Vec2to3 macro



convert data from 2-vector, scalar to 3-vector

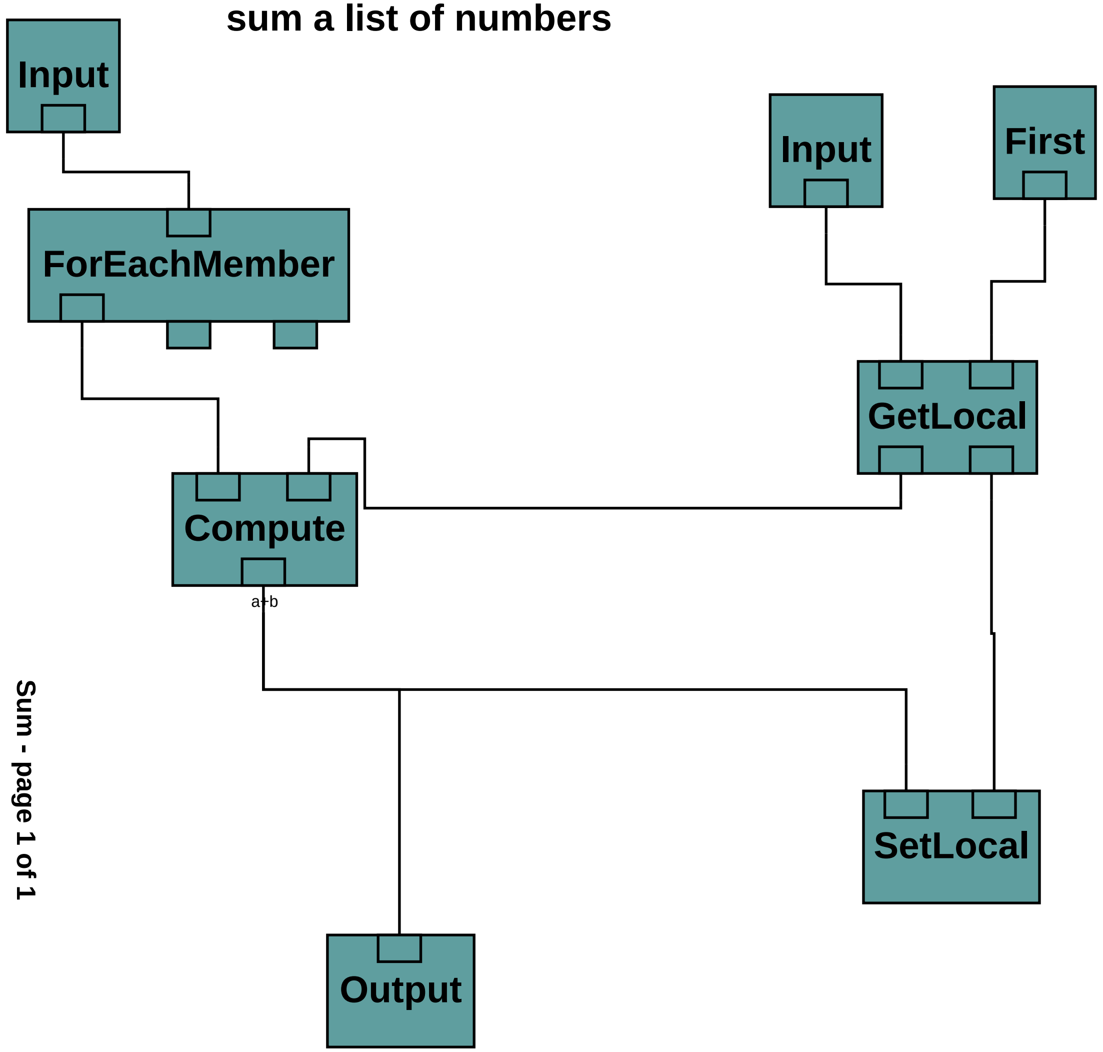
Yminmax macro - compute ymin, ymax



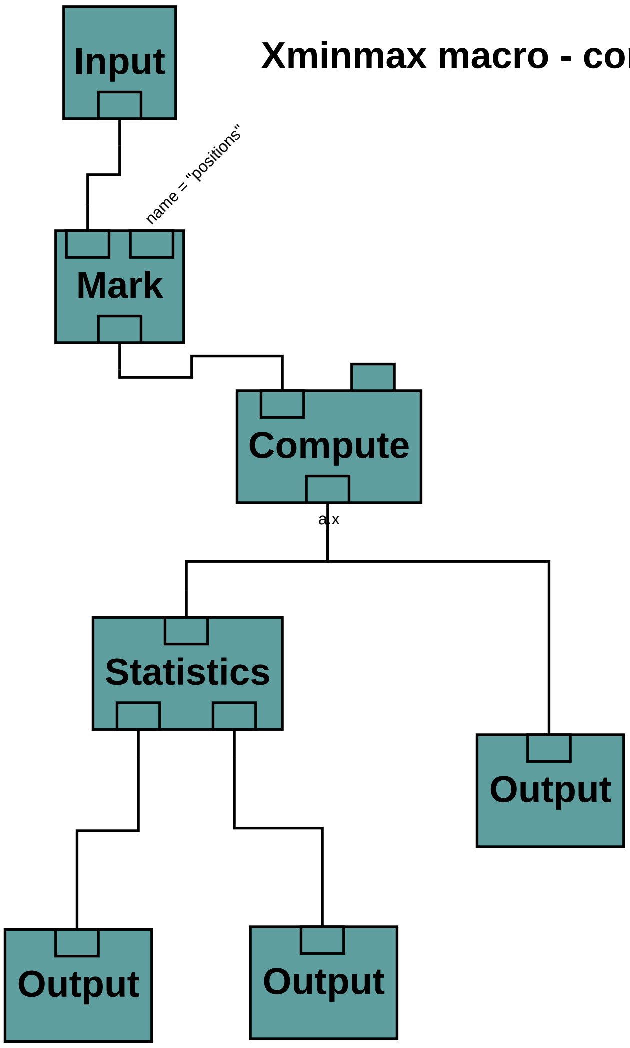


**XYcrop macro:
exclude data based on
x or y, min and max**

sum a list of numbers



Xminmax macro - compute xmin, xmax


Input
name = "positions"
Mark
Compute
a.x
Statistics
Output
Output
Output