

The Nanomanipulator: A Virtual-Reality Interface to a Scanning Tunneling Microscope

TR94-030

1994

Russell M. Taylor II

Department of Computer Science
The University of North Carolina
Chapel Hill, NC 27599-3175



This work was supported by the following grants: NSF IRI-9202424, ARPA DABT 63-92-C-0048, NIH Division of Research Resources RR02170, and ARPA & NSF cooperative agreement ASC-8920219.

UNC is an Equal Opportunity/Affirmative Action Institution.

THE NANOMANIPULATOR: A VIRTUAL-REALITY INTERFACE TO A
SCANNING TUNNELING MICROSCOPE

by

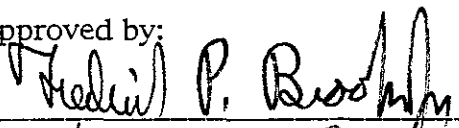
Russell M. Taylor II

A Dissertation submitted to the faculty of the University of North Carolina at
Chapel Hill in partial fulfillment of the requirements for the degree of Doctor
of Philosophy in the Department of Computer Science.

Chapel Hill

1994

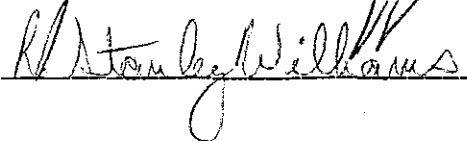
Approved by:



Fred P. Brooks Advisor



James M. Cargill Reader



Robert G. Williams Reader

© 1994
Russell M. Taylor II
ALL RIGHTS RESERVED

RUSSELL M. TAYLOR II. The Nanomanipulator: A Virtual-Reality Interface to a Scanning Tunneling Microscope (Under the direction of Frederick P. Brooks, Jr.)

ABSTRACT

We have developed a virtual-reality interface to a scanning tunneling microscope (STM); the resulting system is called the *Nanomanipulator*. The user interface comprises a stereoscopic color head-mounted display, a force-feedback remote manipulator master station, and a high-performance graphics computer. It provides the illusion of a surface floating in space in front of the user. The user's hand gestures are translated into commands that are sent to the STM in real time; the returned video and haptic signals allow the user see and to feel the surface topography and to control the timing and location of voltage pulses applied between the tip of the STM probe and the sample under study.

My thesis is that a virtual-reality interface is a powerful and effective user interface to an STM — allowing qualitatively different types of experiments to be performed. The success of our investigations using this system demonstrates the validity of the thesis.

We have used the Nanomanipulator to examine various surfaces and to perform surface modification experiments. This investigation has led to new insight into the meaning of certain surface features and into the mechanisms by which voltage pulses change the tip and sample. These insights were the direct results of the real-time visualization and the more interactive nature of our system compared to standard methods.

The key to the success of the Nanomanipulator system is that it provides an intuitive two-way interface to the instrument. Raw data from an STM is not in a format easily understood by a scientist, and the Etch-a-Sketch type of controls required for positioning an STM tip are neither natural nor familiar to a user. The Nanomanipulator system acts as a translator between the instrument and the scientist, allowing the scientist to concentrate on interacting with the surface under study rather than on the computer interface or the STM itself. This system seeks to put the scientists **on** the surface, **in** control, **while** the experiment is happening — thus turning the STM from a remote, batch surface modifier into a real-time, user-guided surface modifier.

ACKNOWLEDGMENT

I wish to thank Warren Robinett for getting me involved in this project and for sustaining ideas throughout. I also thank my advisor, Frederick P. Brooks, Jr., for suggestions, advice, and endless readings of this text.

I thank R. Stanley Williams for the loan of an STM, creation of surfaces to test, and advice and help throughout the project. I thank Erik Snyder for setting up the STM and for help and advice. I thank Sean Washburn for using the system and providing useful feedback, as well as advice and surface characterizations. I thank Vernon L. Chi for his expert work designing and characterizing electronics for all parts of system operation, for his advice, and for loans of equipment for testing the system and performing experiments. I thank the faculty members William V. Wright and Gary Bishop for guidance as project members, and James Coggins as my committee member.

I would also like to thank the student members of the group — Roberto Melo, Joseph Fletcher, Mark Finch, and Jonathan Halper — for design, evaluation, and building of system components. Further thanks are due Mark for his work on all aspects of the project.

Additionally, I would like to thank those who helped in various ways — Ron Azuma, Brad Bennett, Steve Brumback, David Ellsworth, Niall Emmart, Erik Erikson, David Harrison, John Hughes, Fred Jordan, Kurtis Keller, David Kuzminski, Anselmo Lastra, William Mark, Steve Ornat, John Poulton, John Thomas, Greg Turk, Norm Vogel, Brian White, and all other members of the Pixel-Planes, Tracker and Head-Mounted-Display projects.

I wish also to thank my wife, Shelly, for putting up with me as I worked on this dissertation, and for moral support throughout.

This work was supported under the following grants: NSF IRI-9202424 (STM), ARPA DABT 63-92-C-0048 (HMD), NIH Division of Research Resources RR02170 (GRIP), ARPA and NSF cooperative agreement number ASC-8920219 (STC). The UCLA portion of this research was supported by a grant from the Surface and Solid State Chemistry Division of the Office of Naval Research. I also acknowledge the support of the UNC Department of Physics.

TABLE OF CONTENTS

LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xii
LIST OF SYMBOLS	xiv
I. INTRODUCTION AND SUMMARY.....	1
The problem: Intuitive control of an STM	1
Thesis statement and summary of results	1
Introduction to the STM	3
STM theory of operation	5
Nanomanipulator System Overview	8
Overview of following chapters	10
II. PREVIOUS WORK	11
Survey of visualizations for STM data	11
Survey of surface modification	13
III. RESULTS AND CONTRIBUTIONS	15
Results	15
Learn something that was unknown before	15
Order of magnitude decrease in experiment time	17
Introduction	18
Experiment	19
Results	21
Build an intentional structure on a surface - not yet	26
Contributions	26
Lessons learned	27
IV. USER'S VIEW OF THE SYSTEM	29
Immersive interface (head-mounted display/ARM)	30
Surface representations	30
Modes of operation	33
Virtual-reality modes	33
STM Control modes.....	34
Menu system	36
Control panels	37

Pulse parameter control	38
Sampling control	39
V. SYSTEM HARDWARE	41
User interface hardware	42
Trackers	42
Polhemus tracker	43
ARM	43
Buttons	44
ARM	44
Python-3	44
Image generation	45
Host.....	45
Pixel-Planes	45
HMD	47
Projector screen	47
Force display	48
Sound.....	48
STM control	48
Personal computer controller	49
Scanning control and height measurement	50
Bias/pulse generation and delivery	50
Post-pulse tunnel current/position	51
Vibrational, thermal, and electrical isolation	51
Sample-advance motor	53
Control of the tip-sample gap	56
Characterizing system components	57
Crystal resonances	57
Pre-amplifier	61
Compensating for tunneling nonlinearity	61
Performance of the new feedback circuit	62
Noise in the system	62
VI. SYSTEM SOFTWARE	66
Surface sampling and reconstruction	66
Sampling	68
Reconstruction	72
Software architectural overview	75

Image generation	75
Vlib	76
PPHIGS	77
GP callbacks	78
User interface	80
Startup	80
Main loop	81
User Interaction	82
Control panels	85
Menus	86
STM report handling	87
Shutdown	88
Virtual-reality devices	88
Trackers	89
Buttons	90
Sounds	90
STM server	91
Setting and reading parameters	92
Controlling the STM	94
Compiling and linking the software	96
Running the software	97
VII. FUTURE WORK	102
Additional capabilities	102
Distance measurement	102
Side-by-side comparison	102
Line cuts through the surface	102
Hardware improvements	102
Additional microscopes	103
Increased scan rate	103
Study of incremental improvement	103
Availability	103
OpenGL	103
Commercially-available Pixel-Planes systems	104
Commercially-available force-feedback arms	104
ANNOTATED BIBLIOGRAPHY	105
APPENDIX A — SDI LIBRARY	114

LIST OF FIGURES

1.1 A scanning tunneling microscope	4
1.2 Classical treatment of a free electron in a material	6
1.3 Quantum-mechanical treatment of an electron	6
1.4 Tip/sample gap	7
1.5 User interface for the Nanomanipulator system	9
2.1 Gray-scale image of surface	11
2.2 Example of publication-quality visualization	12
3.1 Graphite planes	16
3.2 Mesa formed by voltage pulse	17
3.3 Circuitry converting tunneling current to voltage.	20
3.4 Flat and exponential events.	21
3.5 Flat events for different values of V_{bias}	22
3.6 Tip offset by I_t result type.	24
3.7 Tip offset after flat events.	25
4.1 User interacting with the system	29
4.2 Surface coloring	31
4.3 Menu system with File submenu selected	36
4.4 Pulse control panel	38
4.5 Sampling control panel	39
5.1 System hardware architecture	41
5.2 Architecture of Pixel-Planes 5	46
5.3 Architecture of PC STM controller subsystem	49
5.4 Vibration isolation system for our STM	52
5.5 Hand-held controller for sample advance motor	54
5.6 Drive electronics for sample advance motor.	55
5.7 Tunnel-current feedback controller diagram	56
5.8 Piezoelectric crystal	57
5.9 Piezo/preamp test setup	59
5.10 Piezo transfer function	60

Types of connections	114
Types of servers	115
Mapping device names to connections	116
Tricks to increase performance	117
Performance studies	118
Suitability of Ethernet connections	119
APPENDIX B — NOISE IN STM INPUTS.....	121
open:	122
terminated:	123
cable:	124
cable_terminated:	125
battery:	126
hp8116a:.....	127
mixer:	128
z_adjust:.....	129
x_dac_still:.....	130
y_dac_still:	131
y_dac_still_20kHz:.....	132
y_dac_long:	133
y_dac_medium:.....	134
y_dac_medium_5kHz:.....	135
y_dac_short:.....	136
y_dac_zoom:.....	137
APPENDIX C — POETIC VIEWS OF THE SYSTEM.....	138
Palabra Oscura	138
Sonnet VI	139
APPENDIX D — SOURCE CODE	140

5.11 Bias noise reduction	63
6.1 C0 continuity	66
6.2 Point sampling	67
6.3 Triangle tessellation	68
6.4 Under-sampling	68
6.5 Aliasing	70
6.6 Surfaces with discontinuities in position and slope	71
6.7 Linear reconstruction error	72
6.8 Phong shading	73
6.9 Surface normal calculation	74
6.10 Software architecture	75
6.11 Virtual world structure	76
6.12 Triangles affected by changing a point	79
6.13 Actions taken during user interface program initialization	80
6.14 Actions taken by main loop of user interface program	82
6.15 Actions taken to handle user interaction	83
6.16 Distribution of incoming STM messages	88
6.17 Patterns used when scanning the grid	92
6.18 Meaning of the pulse parameters	93
6.19 Libraries used by the user interface code	97
A.1 Histogram of round-trip latencies.....	119

LIST OF ABBREVIATIONS

ADC, A/D	Analog to digital converter, conversion
ARM	Argonne-III remote manipulator
DAC, D/A	Digital to analog converter, conversion
dB	Decibels
FIFO	First-in-first-out (like the line at a grocery store)
FFT	Fast Fourier transform
GP	Intel i860-based Graphics Processor on Pixel-Planes 5
HIF	Host interface board connecting Pixel-Planes 5 to its Sun-4 host
HMD	Head-mounted display
HOPG	Highly oriented pyrolytic graphite
LAN	Local area network (in our case, a 10Mbit/sec Ethernet)
LCD	Liquid crystal display
MFlops	Million floating-point operations per second (Very rough measure of computer processing power)
mV	Millivolt
μ s	Microsecond
nA	Nanoampere: 10^{-9} Ampere
nm	Nanometer: 10^{-9} meter.
PC	Personal computer (486DX2-50 PC/AT)
SDI	Standard device interface (network connection library)
STM	Scanning tunneling microscope
TCP	Transmission control protocol (network stream connection)

UNC	University of North Carolina (at Chapel Hill)
V	Volts
Vlib	Virtual-worlds library (designed by HMD team at UNC)
VR	Virtual reality

LIST OF SYMBOLS

Å	Ångstrom: 10^{-10} meter; approximately the radius of an atom.
Au	Gold
ϕ	Work function (energy required for electron removal)
Pt	Platinum
PtSi	Platinum Silicide
Si	Silicon
W	Tungsten

I. INTRODUCTION AND SUMMARY

The problem: Intuitive control of an STM

The problem we have undertaken to solve is providing an ideal user interface to a Scanning Tunneling Microscope (STM). Such an interface is required because the raw data presented by the STM is not in a form that is readily understandable by a scientist. Also, the parameters controlling the scanning and surface modification capabilities of an STM are not intuitive.

We have provided a virtual-reality interface that translates data from the STM into a form that is readily and intuitively understood by the scientist and to translate commands that are natural to the scientist (pointing at an area of interest and pressing a trigger) into commands that cause the STM to respond. We call the entire system, including the STM, the *Nanomanipulator*.

Thesis statement and summary of results

The processes by which voltage pulses change the surface under an STM are not well understood and will require extensive experimentation to be understood. A system that improves the facility, quality and speed of experimentation will accelerate the process of discovering the science of surface modification. Improved quality will reveal itself as new insights into the processes of change.

In the standard method of controlling an STM, the scientist lays out a plan for the experiment — where the tip is to scan and where and when pulses are to occur. Once the data has been collected, it is converted into a graphical image from a given point of view with specified lighting parameters; the experimenter studies the resulting image and uses it to provide insights and plan future experiments. This method is analogous to the batch-mode programming once standard in computer use.

We have provided a system that allows one to exert real-time control of the viewing parameters and lighting through the natural method of moving one's head. The data is continuously available for viewing, with new data overwriting the old as it arrives. This allows the user to build a mental model of the surface that is based not on a series of unrelated snapshots, but rather on a sequence of observations of a surface that are obtained in response to natural motions of his head and body. Additionally, the system allows the user to request voltage pulses at any time during the experiment by hand-placing a cursor at the desired location and pressing a trigger. This method is analogous to the time-shared, interactive programming that is standard today.

Reason suggests that experiments done with this system will be qualitatively different from prior experiments: while observing the image, the scientist can continuously adjust the viewpoint and lighting parameters until interesting features are revealed — features that the scientist does not know he is looking for until he sees them. Also, the rate of performing experiments is accelerated so that each takes place within the attention span of the scientist, thus removing the mental context switches between conceiving of an experiment and seeing its results.

Using this system, the experimenter can directly, immediately, and naturally control the parameters of the experiment and can directly and immediately observe the results. This allows a mode of experimentation consisting of a sequence of mini-experiments with immediate feedback. The scientist can direct the exploration continuously and make impromptu changes to the viewing parameters and experimental plan between each mini-experiment.

My thesis is that a virtual-reality interface is a powerful and effective user interface to an STM — allowing qualitatively different types of experiments to be performed. This difference reveals itself by providing insights into surfaces that are otherwise missed and by enabling new kinds of experiments that provide insights into the processes of surface modification.

Indeed, the system has provided insights into surface features that prior methods had missed: early in the work the system was used to view a data set

that had been previously studied for months using existing methods. When the same data was viewed with continuously-guided viewing and lighting parameters (steered by a user who was naive in materials science), scientists immediately recognized a feature that had confused them before as being a series of graphite sheets that were tilted up out of the surface. [Taylor93]

The system also enabled a series of experiments that led to the discovery of a process by which voltage pulses modify the surface: at times the tip will weld itself to the sample after a pulse and then be drawn back until breaking free. [Taylor94] Around 300 mini-experiments were performed and analyzed in four 5-hour blocks to determine the range of results possible; the time taken for this many mini-experiments using conventional methods would have been prohibitive.

The Nanomanipulator system has produced research results in computer graphics [RobinetCourse92] [Taylor93] and surface science [Taylor94], thus validating its usefulness to each discipline as a collaborative effort. It has also provided an example of a useful, working virtual reality system to a wider audience. [Taylor93VIR] [Taylor94IMG] [Taylor94LFW]

Introduction to the STM

The Scanning Tunneling Microscope (STM) was conceived in 1978 by Binnig and Rohrer at the IBM Zurich Research Laboratory and first demonstrated in 1981. Its invention was honored by the Nobel prize in Physics for 1986. It was originally designed to aid in understanding the growth, structures, and electrical properties of very thin oxide layers. [Binnig82] [Binnig87]

An STM consists of piezoelectric positioning elements, a conducting (usually metal) tip and a conducting sample (the surface under study). In our instrument, built by E.A. Eklund at UCLA, the piezoelectric crystal elements are arranged as three orthogonal bars, each of which controls motion along one axis (see Figure 1.1). As voltages are applied across the crystals, they change their lengths. Since the tip is rigidly attached to the crystals, they can be used to position the tip relative to the sample. Our STM can scan areas up to 200 nanometers (nm) on a side.

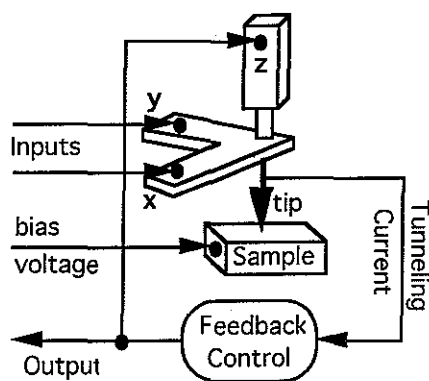


Figure 1.1 A scanning tunneling microscope. The feedback control maintains the tip at a constant distance above the surface.

A bias voltage is applied to the sample with respect to the tip. At very close range (on the order of a few tenths of a nanometer), a tunneling current flows between the tip and the surface. This current decreases exponentially with increasing distance between the tip and the sample. In our configuration, the Z crystal is controlled by a feedback circuit that attempts to maintain a fixed tunneling current and thus a constant distance between the tip and sample. The X and Y piezoelectric crystals are used to raster the tip back and forth across the surface. A scan of the surface proceeds by repeatedly moving the tip in X and Y and then reading the voltage on the Z crystal to determine surface height.

The STM is capable of resolving individual atoms in a sample. The radii of atoms range from about 0.10 to 0.27 nm, or approximately 1 billionth the size of common objects such as golf balls or basketballs. Typical chemical bonds range from 0.15 to 0.25 nm. For comparison, a typical feature on one of today's integrated circuits might be 1 micron (1,000 nm) across. Optical microscopes are limited in resolving power to approximately the wavelength of the radiation used in imaging, which is 400-700 nm.

The STM uses a very sharp physical probe to gather information about the sample surface, rather than analyzing reflected photons or electrons. The key to the resolution of the STM is that the length of the piezoelectric positioners can be accurately controlled to 0.01 nm, and that the exponentially varying tunneling current is extremely sensitive to tip-to-sample separation

(moving the tip 0.1 nm closer to the surface increases the tunneling current by a factor of 10).

Unlike other microscopes, the STM provides its information as an elevation map rather than a projected image. The scientist wants to understand the geometry of the three-dimensional surface, so these values must be interpreted. The most natural method of interpretation is to reconstruct the surface from the sampled height information, a common process in computer graphics.

In addition to its ability to map the surface, the tip of the STM can be used as a local probe to modify the surface. [Becker87] This makes the STM useful for nanofabrication. There are at least two ways this can be accomplished. The first is to physically contact the surface with the tip, which causes large and unpredictable modifications to both the tip and the surface. The second, more controlled method is to apply a voltage pulse between the tip and the surface. Since the distance between the two is so small, even moderate voltages produce strong electric fields.

STM theory of operation

This treatment closely follows [Cohen-Tannoudji77] and [Stoll91].

Classically, electrons are particles that have a specific location and velocity at any moment in time. Also, electrons do not normally travel outside the material of which they are a part, due to the energy barrier at the edge of the material. Sufficient energy (such as a strong electric field) can overcome this barrier and draw the electrons out of the surface. The amount of energy required to move an electron from the surface to a point at infinity is called the *work function* of the surface; the value of the work function varies for different materials (see Figure 1.2).

A quantum-mechanical treatment of the electron treats it not as a particle, but rather a probability density; the electron exists in a set of possible locations until it is actually observed, when it “rolls the dice” and chooses one of the possible locations. The probability density is continuous and decays exponentially in regions where a classical electron would not go (see Figure 1.3).

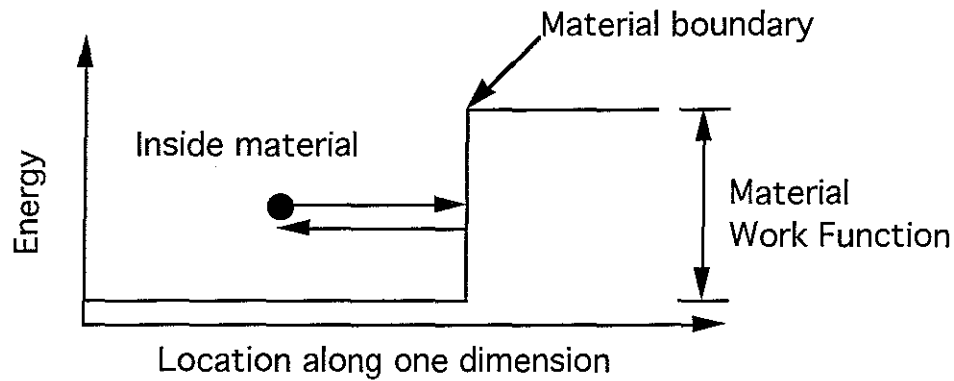


Figure 1.2 Classical treatment of a free electron in a material. The electron does not have enough energy to overcome the work function of the material, so it reflects off the material boundary.

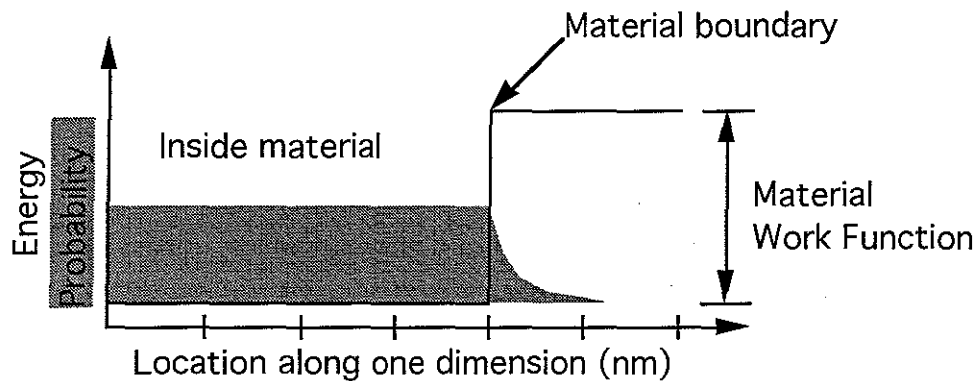


Figure 1.3 Quantum-mechanical treatment of an electron. Magnitude of electron's probability distribution plotted on same graph as energy. The electron is found outside the material with some probability. The probability falls exponentially to near zero within a few nanometers from the material surface.

As the tip and the sample are brought close to each other (on the order of 1 nanometer), electrons can tunnel from one to the other. This is a quantum effect that results from electrons having a non-zero probability of being found just outside the surface of a conductor. This means that there is some probability that an electron from one conductor will cross over into a nearby conductor. When there is a bias between the two conductors, this can influence the direction of tunneling and produce a net current flow from one conductor to the other, even though the two are separated by a non-conducting gap.

Although the exact equation governing electron tunneling between two conductors that have the shapes of the tip and sample is not known, it can be approximated by the following equation: [Stoll91]

$$i = \frac{e^2}{h} \frac{\sqrt{\frac{2m_e\phi}{h^2}}}{4\pi^2 D} V A_{eff} \exp\left(-2\sqrt{\frac{2m_e\phi}{h^2}} D\right)$$

In this equation, i , D , and V are the tunneling current, distance, and applied voltage as shown in Figure 1.4. ϕ is the average work function of the material in the tip and sample $(\phi_1 + \phi_2)/2$. A_{eff} is the effective area over which electrons tunnel. The other parameters, which do not vary from experiment to experiment, are the mass of the electron (m_e), Plank's constant (h), and the electron charge (e).

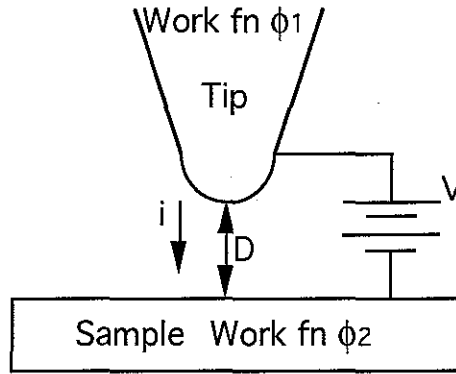


Figure 1.4 Tip/sample gap. D is the distance between the tip and sample. V is the applied voltage potential. i is the tunneling current. ϕ_1 and ϕ_2 are the work functions of the tip and sample.

For a given tip, sample, and bias we can express the relationship between the tunneling distance D and the tunneling current i as $i = \frac{K_1}{D} \exp(-K_2 D)$, where K_1 and K_2 are positive constants that depend on the experimental setup. Typical tunneling conditions have an applied voltage of a few hundred millivolts, a tunneling current of a few nanoamperes, and a tunneling distance of about a nanometer. The (at least) exponential drop-off of tunneling current with distance provides extremely high vertical resolution for the STM (on the order of 1% of one atomic diameter).

The horizontal resolution of an STM is limited by the radius of curvature of the tip used for scanning. This is because electrons can tunnel from any of the atoms at the end of the tip to any of the atoms on the surface. The effective area of tunneling increases with the radius of curvature of the tip. For this reason, sharp tips are critical in order to achieve high horizontal resolution when scanning.

Nanomanipulator System Overview

We are just beginning to have fast enough graphics engines and acceptable trackers to allow us to provide scientists with real-time immersive virtual-world interfaces to their instruments. We have brought this power to bear on the visualization of data from and control of a Scanning Tunneling Microscope with the UNC/UCLA Nanomanipulator system. The virtual-world interface demonstrably contributes to the power of the instrument.

In January 1992, an STM built at UCLA was brought to North Carolina and interfaced to the existing hardware and software of UNC's Head-Mounted Display project and the GROPE force display project. [RobinettCourse92] We named the system the "Nanomanipulator" because it allows the user to see, feel, and manipulate matter at the nanometer scale.

The STM functions as both the imager and effector in this atomic-scale teleoperator system. The system operates in three modes. In *raster-scan mode*, the STM tip moves back and forth, continually streaming in new surface height data on a user-specified grid. Our system uses this data to update the reconstructed surface model in real time. Independently and asynchronously, the viewer may fly about the surface, or hold it at arm's length and tilt it so that the directional illumination reveals and highlights surface detail. The surface model serves as the buffer mediating between the back-and-forth slow scanning of the STM and the TV-like fast scanning of the display system. At the same time, experimental parameters and instrument data are stored for later replay or off-line analysis.

In *feel mode*, the scientist uses the manipulator arm to move the STM tip directly (as the crow flies) over the surface, feeling the contours, and perceiving particular point heights, as the STM visual cursor traverses the surface image.

In *pulse mode*, the user also moves the tip directly over the surface, and, with a finger trigger, may select locations to fire voltage pulses, modifying the surface.

The user interface through which the human user perceives the microscopic world consists of a stereoscopic head-mounted display and a force-feedback handgrip. Motions of the user's head and control gestures of the user's hand are scaled down approximately one billion times by the Nanomanipulator to control the viewpoint from which the microscopic world is seen by the user and to control surface modifications effected by the STM.

Figure 1.5 shows the user interface for the Nanomanipulator.

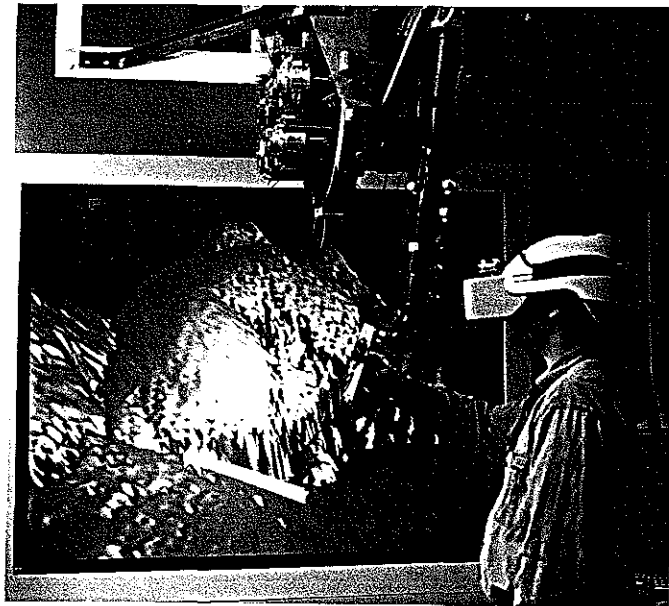


Figure 1.5 User interface for the Nanomanipulator system. The user can control the action of the STM tip with the force-feedback ARM, feel surface contours, and specify bias pulses by pressing the finger trigger. The user sees the sample surface through the head-mounted display. A copy of the user's view is also projected on a wall screen for onlookers.

The graphics system that generates the stereoscopic images for the HMD provides highly detailed shaded 3D color images in real time. The HMD and head-tracker allow the graphics to be generated in coordination with the user's voluntary head motions, so that users perceive themselves to be surrounded by the microscopic environment.

Overview of following chapters

Chapter 2 presents prior work, both in STM visualization methods and in surface modification using STMs.

Chapter 3 presents the results obtained with the Nanomanipulator system and lists the contributions made by this project. It also lists some of the lessons learned in the course of investigation.

Chapter 4 presents the system from the user's point of view. It describes the interface and the facilities that are available to the user of the system to aid in performing experiments.

Chapter 5 describes the system hardware, including the user interface hardware and the hardware that is used in control of the STM.

Chapter 6 describes the system software. This includes the methods used to reconstruct the data from the samples, the user interface software, and the STM control software. This chapter also provides information on compiling and running the software in its current environment.

Chapter 7 describes future directions for the project.

Appendix A describes the Standard Device Interface (SDI) library that is used by the system to connect its various components.

Appendix B provides information on the amount of electrical noise present on various inputs and outputs to the STM.

Appendix C contains two poems that were written about the system and presented to us by their authors.

Appendix D contains pointers to the source code for the system.

II. PREVIOUS WORK

Survey of visualizations for STM data

Standard practice for STM data visualization during data collection is viewing gray-scale images of the surface as seen from above, where dot brightness corresponds to surface height at each point in the image. An example of this sort of image is seen in Figure 2.1. For later offline viewing or publication, most visualization is done using various graphing routines on personal computers. These packages draw a connected line for each scanline the tip has made, doing hidden line removal on areas that would be occluded. Color is used effectively to show either surface height or other surface properties. For presentation, shaded surface images are often computed off-line, after the experiment is complete. [Edstrom91] [Komuro87] [Stoll91] [Snyder91] [Magonov91]

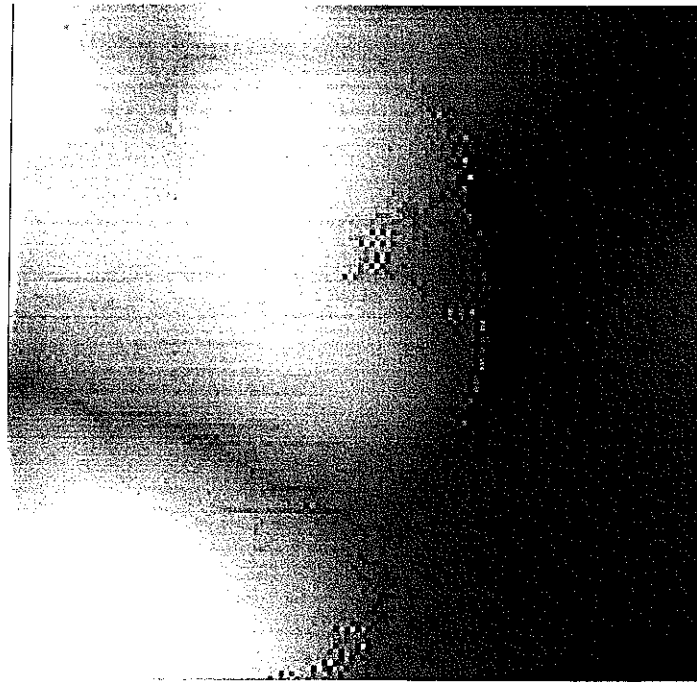


Figure 2.1 Gray-scale image of surface. The brightness of the image at each location depends on the height of the surface; higher areas on the surface are brighter.

Figure 2.2 shows an example of a publication-quality image. This picture shows the interference patterns of electrons that are captured in a “corral” of 48 iron atoms adsorbed onto a copper surface. [Crommie93] The fact that scientists use images of shaded surfaces colored according to properties of the data when publishing their findings indicates that they consider such presentations to be a useful way to convey information about a surface. By providing such presentations in real time, we will bring the benefits of publication-quality visualization to bear during the course of an experiment.

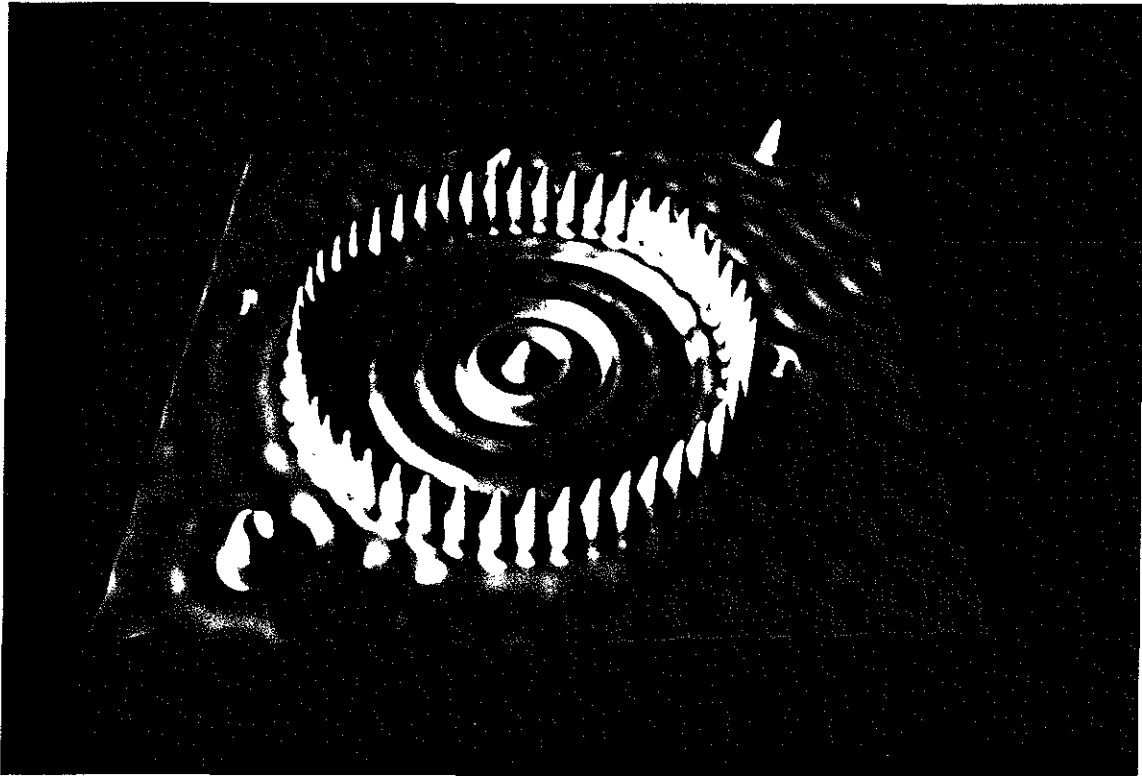


Figure 2.2 Example of publication-quality visualization. Image made at the IBM Almaden Research Center showing a spatial image of the interference pattern caused by electrons scattering from a ring of iron atoms on a copper surface. [Crommie93]

Recent work by Lyding at the University of Illinois in collaboration with the Beckman Institute has resulted in an STM interface module for the Application Visualization System (AVS) that gives data from the STM scans to other AVS modules for high-quality interactive rendering of surfaces using Gouraud shading. This results in shaded images from a given point of view at rates on the order of one image per minute. Lyding found that his interactive viewing system guides experiments while the data is still being collected.

[personal communication] We have found the same effect with our interactive system.

Besenbacher et al. at the University of Aarhus, Denmark, have taken another approach. They still draw the images on a personal computer, but they make images from successive scans and put them onto videotape for later viewing. This allows them to view surface dynamics that happened during the experiment. The dynamics cannot be viewed while the experiment is in progress, but nonetheless they have found the motion display to be very useful. Referring to his videotaped images of the scanned surface, Besenbacher writes "...one can record *STM movies* and thereby visualize in real time and space dynamical processes on metal and semiconductor surfaces. Such information, which cannot be obtained by any other means, is very decisive for a full understanding of both the growth mode of reconstructed phases and the resulting static structure." [Besenbacher91] We also have found real-time viewing of the data to be valuable.

Survey of surface modification

There are many groups around the world performing surface modification using STMs. I present a sampling here, including recent advanced work.

Work by Mamin et. al. at IBM Almaden Research Center studied modifications to gold surfaces using gold tips. After examining the results of many pulses, they came to the conclusion that field evaporation was the cause of mound formation on the surface after a pulse. The mounds formed were from 10-50 nm across, were stable, and could be made quite reliably. They used their system to produce a map of the world that was about $1\mu\text{m}$ across. [Mamin91]

Another research group at IBM Almaden Research Center, led by Eigler, has positioned individual atoms to form structures. Working in ultra-high vacuum and at ultra-low temperature (about 4° Kelvin), the group positioned individual xenon atoms on a nickel surface to form the letters "I B M." [Eigler90] They later went on to create the electron corral depicted above.

Both Lyo and Avouris, and Kobayashi et al., have shown that it is possible to alter the surface of a silicon crystal by carefully controlling the tip height and the bias between the tip and sample. The former authors have successfully removed clusters and even individual Si atoms from a surface by applying voltage pulses of +3V to the sample under study (with the tip grounded). The amount of material transferred in each pulse depended on the distance from the bottom of the tip to the sample surface (the smaller the distance, the larger the field and thus the more material transferred). They were also able to transfer atoms from the tip back onto the surface by applying voltage pulses of -3V to the sample. Kobayashi et al. have been able to form trenches only a few nanometers wide. They scanned the tip over the surface at a speed of 50 nm/s while holding the sample at a constant voltage of either polarity in the range from 4-10 volts. They used tip-sample separations significantly larger than those used by Lyo and Avouris. These studies have demonstrated the feasibility of altering the structure of a surface literally one atom at a time. What they lacked was the ability to interactively view the surface while it was being modified (they set the experiment, let it run, and then viewed the data afterwards). [Lyo91] [Kobayashi93]

Further work by the Aono group in Japan (which includes Kobayashi, Grey, and others) extended the work by Kobayashi and has been able to remove single silicon atoms from specific lattice sites and place single atoms at specific lattice sites. This spectacular work was done in ultra-high vacuum using tips that were meticulously prepared using etching, electron bombardment, and pulsing. [Grey93] [Aono93]

III. RESULTS AND CONTRIBUTIONS

Results

“If we perceive our role aright, we then see more clearly the proper criterion for success: a tool-maker succeeds as, and only as, the *users* of his tool succeed with his aid. However shining the blade, however jeweled the hilt, however perfect the heft, a sword is tested only by *cutting*. That swordsmith is successful whose clients die of old age.” [Brooks88]

Learn something that was unknown before

In building the Nanomanipulator, we have created a user interface for an STM. To study the effectiveness of this interface, we must study the insights into surface science that have been obtained using it.

While in the preliminary stages of this project, we received a data file from UCLA that contained data for a Highly-Oriented Pyrolytic Graphite (HOPG) sample that had been bombarded with 5 keV argon ions. [Eklund91] This data had previously been examined for several months in order to determine the effects of the radiation. We tessellated the data with triangles and used a standard viewing program to generate a videotaped fly-through of the data set and sent the tape back to UCLA. The fly-through was guided by an operator who was skilled in computer graphics, but a novice to surface science. The footage used in the tape was displayed in real-time on Pixel-Planes 5, which is the same graphics computer used by the Nanomanipulator system; the images were comparable to those produced by the Nanomanipulator.

While viewing the videotape (real-time color shading, but no stereo), the scientists at UCLA discovered that some features on the surface they had thought to be noise were actually graphite sheets that had been tilted up out of the surface (see Figure 3.1). The specular highlighting of the surface and particular viewpoint brought the features to the scientists' attention; the fact

that these sheets were regular and that they were not aligned with the scan direction allowed the determination of their origin. There are about seven samples per undulation of the sheet, so the undulations are not an artifact of the surface tessellation; they are about 1 nm in height, so they are not an artifact of under-sampling the crystal lattice (such artifacts would be only about 0.1 nm high). The discovery of the planes showed us that a visualization using real-time shading and motion is superior to either static gray-scale images or static line-contour images, both of which had been used on the data before, and gave us the impetus to relocate an STM from the Department of Chemistry at UCLA to the Department of Computer Science at UNC.



Figure 3.1 Graphite planes. This is a rendered image of an ion-bombarded highly-oriented pyrolytic graphite (HOPG) sample. The large scrapes in the center are visible even on a gray-scale image; the more subtle diagonal striping in the upper right corner is accentuated by highlighting and indicates graphite sheets that have pushed up out of the surface.

Order of magnitude decrease in experiment time

As part of our investigations, we performed a series of experiments to study the mechanisms of material transfer between an Au surface and tip during voltage pulses (see Figure 3.2). An important part of these experiments involved firing a large number of voltage pulses and observing their effects on the surface. In order to separate the changes to the tip from the changes to the surface, it was necessary to re-scan the surface after each pulse and watch how the surface evolved over a couple of scans.

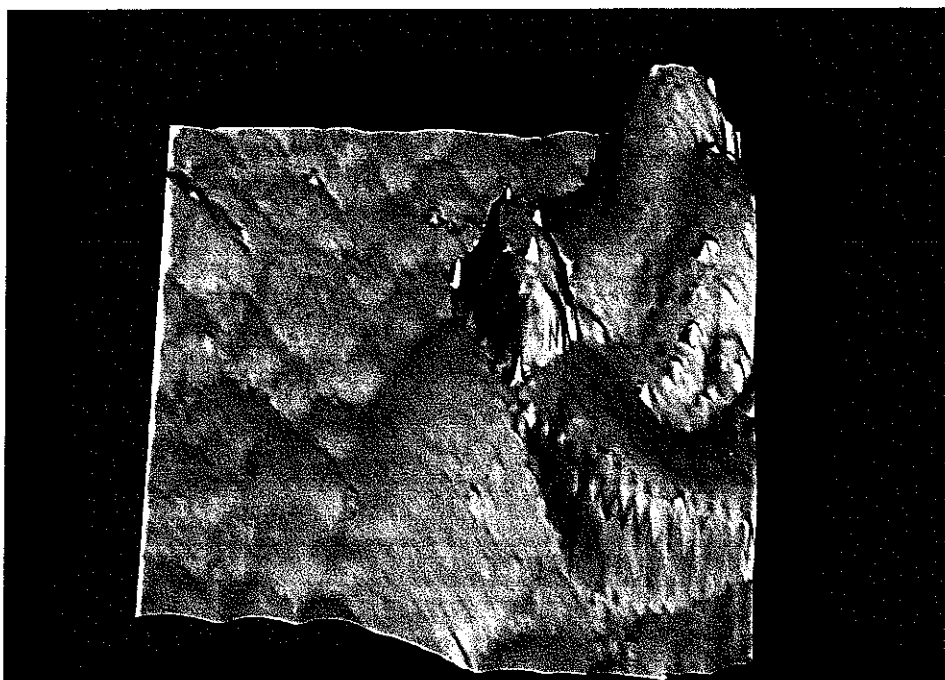


Figure 3.2 Mesa formed by voltage pulse. This mesa was formed by the STM tip welding itself to the sample and then being pulled loose by the feedback circuit after a voltage pulse. The interactive nature of the system allowed us to perform the experiments that discovered this effect.

The interactive control over pulse locations, combined with the immediate shaded display of data acquired from the STM, allowed us to do over 300 pulse experiments (including the determination of the tip-length change, surface modifications, and form of tunneling current after each pulse) in four 5-hour blocks. Without such control, the amount of effort required to determine the pulse effects and correlate them to the pulse parameters for their individual pulses would have prohibited performing the experiment.

Seeing the data arrive in real time allowed us to easily understand which surface transients were caused by brief tip lengthening after the pulses, as opposed to tip changes that occurred spontaneously when no pulse was fired; it also allowed us to determine which changes to the surface were temporary and which persisted for several scans.

This sequence of experiments revealed that it was often the case that the tip became welded to the sample after a pulse and would then be drawn back until breaking, leaving a mound on the surface and increasing the length of the tip. [Taylor94SSL]

Thus, interactive control of the pulse locations combined with interactive viewing of shaded display of the scan data enables new types of experiments that were not feasible before, experiments that reveal new insights into the mechanisms of surface modification. A further description of the experiment (closely following the presentation in [Taylor94SSL]) follows:

Introduction

There have been many reports of surface modification caused by voltage applied to the tunneling tip in a scanning tunneling microscope (STM). [Ringger85] [McCord86] [Stroscio91] Some of the work has been quite elegant in that single atoms have been added [Becker90] or removed [Avouris92] [Lyo91] [Kobayashi93] [Grey93] [Aono93] to create artificial *atomic-scale* structures. The technological importance of this capability cannot be overemphasized. The atomic-scale structures rely on fortuitous materials parameters (such as weak binding energies of certain atoms on certain surfaces [Avouris92] [Lyo91]) that cannot be duplicated in all materials systems.

Tunneling tips made from Au have been used routinely to modify the surface morphology of Au films. [Mamin91] The Au surface modifications have been at much larger than atomic scale. The smallest features are several nanometers in dimension, and tens of nanometers are more common. Instead of single atoms, mounds of material are transported from the tip to the surface and back depending on surface morphology and tip-surface voltage bias. Apparently a voltage bias of 3-5 V (corresponding to electric fields of 10^7 V/cm) is sufficient to cause field evaporation from the tip to the surface. By

using short voltage pulses (a few hundred nsec), Mamin and coworkers have been able to form uniform permanent mounds of Au on Au surfaces. Other materials have been deposited from other tips onto Au and onto semiconductor surfaces, but the most extensive work was done for Au/Au. [Mamin91]

The question of what in detail occurs during a surface modification event has not been settled. In atomic-scale modifications, the local surface chemistry plays an important role, which is apparent because many modifications occur at particular lattice sites. [Ringger85] [McCord86] [Stroscio91] [Becker90] Voltage pulses of approximately the local binding energy were used to reversibly place and remove atoms on semiconductor surfaces. [Becker90] In the Au/Au modifications, the local surface chemistry does not vary substantially over the Au surface, so another model must be employed. It was suggested that field evaporation from the tip to the surface was the main cause of the observed changes. [Avouris92] [Lyo91] [Kobayashi93] [Grey93] [Aono93] Observations that the minimum voltage bias to achieve surface changes was dependent on the tip-to-surface separation were consistent with the model that a certain electric field is required to move material from the tip to the sample or back.

Alternative models included hydrostatically-induced flow between the tip and surface [Landman90], and local melting of the tip and the surface, perhaps due to heating a small region with the rather high tunnel current density ($\approx 10^3 \text{ A/cm}^2$). Comparing the pulse power to the thermal time constants in the tips, one finds that the heating model does not seem plausible, since the heat dissipated in the tip (the smallest constriction) is only $\approx 10^{-16} \text{ W}$ under tunneling conditions and 10^{-11} W when shorted.

We have studied the temporal response of the tip feedback voltage and tunneling current just after short voltage pulses in order to determine in detail what happens to the tip and the surface during the modification. Our results indicate that the simple field evaporation explains most of the voltage pulses, but that frequently a filament forms between the tip and sample.

Experiment

Our experiment employed a standard piezo tripod. [Eklund91] The tips were either etched W, cut Au wire (0.25 mm in diameter) or cut PtRh wire

friction-loaded into a hollow tube on the piezo tripod. All experiments were conducted in air inside an electrically-shielded enclosure. The samples were electroplated Au films on the ends of the center conductors of specially configured coaxial connectors. The tunnel current I_t was sensed with a current preamplifier comprising a $10\text{ M}\Omega$ shunt resistor R_{sh} and a gain-of-10 buffer (see Figure 3.3).

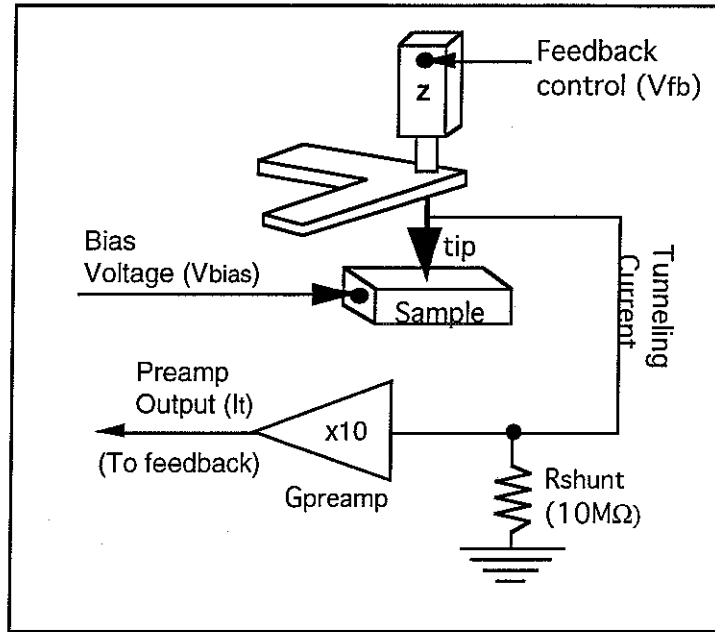


Figure 3.3 Circuitry converting tunneling current to voltage. The tunneling current flows through a $10\text{ M}\Omega$ shunt resistor, converting it to a voltage of 10 mV/nA . The gain-of-ten preamplifier converts this to a 100 mV/nA signal I_t that drives the signal to the feedback electronics. The feedback electronics servo the tip position by adjusting the feedback voltage V_{fb} .

The bias voltages V_{bias} were varied between 0.2V and 1V in both polarities, and the pulse heights V_{pulse} were several volts ($5\text{--}12$) at the same polarity as V_{bias} . In the experiments reported here, there was no significant difference between experiments at positive and negative polarity. The pulse lengths t_{pulse} were not critical above a threshold of a few tens of nanoseconds. Cut and etched tips behaved about the same. Neither was able to make surface modifications as reproducibly as has been claimed in experiments done under ultra-high vacuum (e.g. [Mamin91]). Both kinds of tip made moderate-sized features ($>20\text{ nm}$) up to very large features ($>200\text{ nm}$). In the experiments

with W tips, it seems likely that the tips were coated with gold after a few pulses, which could explain the similar behavior between Au and W tips.

We used a Tektronix DSA 602 Digital Signal Acquisition unit to measure I_t and the feedback voltage V_{fb} transients during and after the bias pulses. The rise time of the tunnel/preamp circuit is too long to observe much reaction during the $< 2 \mu\text{sec}$ pulses, but V_{fb} and I_t continued to react for several msec after the pulse. We began scanning again several tens of milliseconds after the pulse was completed, which is after the end of the transients shown.

Results

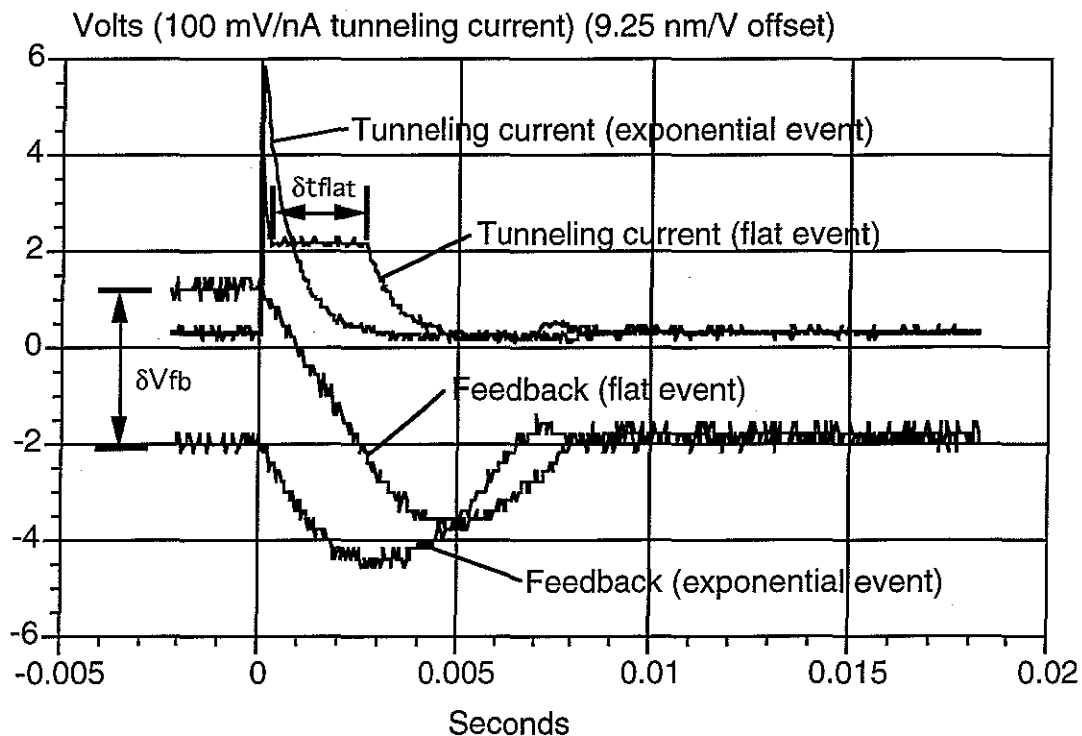


Figure 3.4 Flat and exponential events. This figure shows the tunneling current (I_t) and feedback voltage (V_{fb}) response for a flat event and an exponential event.

We have found that several different kinds of events can occur when a pulse is fired between the tip and surface. Mounds can be formed on or removed from the surface, and the tip can be shortened or lengthened. The tip shape can change as well in subtle or (more rarely) dramatic ways. All of these events can be correlated with transient recordings of I_t and V_{fb} . Most of the events can be labeled as flat or exponential based on the response in I_t .

Similar results have been reported by others. [Avouris92] [Lyo91] [Kobayashi93] [Grey93] [Aono93] [Landman90]

A flat event is characterized by a transient in I_t as shown in Figure 3.4. I_t falls rapidly from the value at the end of the initial pulse at $t=0$ and settles to the value $V_{bias}G_{preamp}$. This indicates that the tip and surface are in direct electrical contact. In nearly all cases the flat event leaves a mound on the surface. The mesa-like mound pictured in Figure 3.2 is the result of one such event. The mound dimensions are 28 nm by 54 nm by 36 nm high, an aspect ratio of height to width which is greater than unity. While I_t is constant, V_{fb} decreases linearly at the rate fixed by the time constant in the feedback integrator until, after a time δt_{flat} (see Figure 3.4), I_t begins to drop exponentially again. This correlation has been confirmed in many such events and is illustrated in Figure 3.5, by several curves which differ in V_{bias} . All of the curves scale such that the resulting plateau is $I_t = V_{bias}G_{preamp}$. Similar data were obtained with PtRh tips and Pt surfaces.

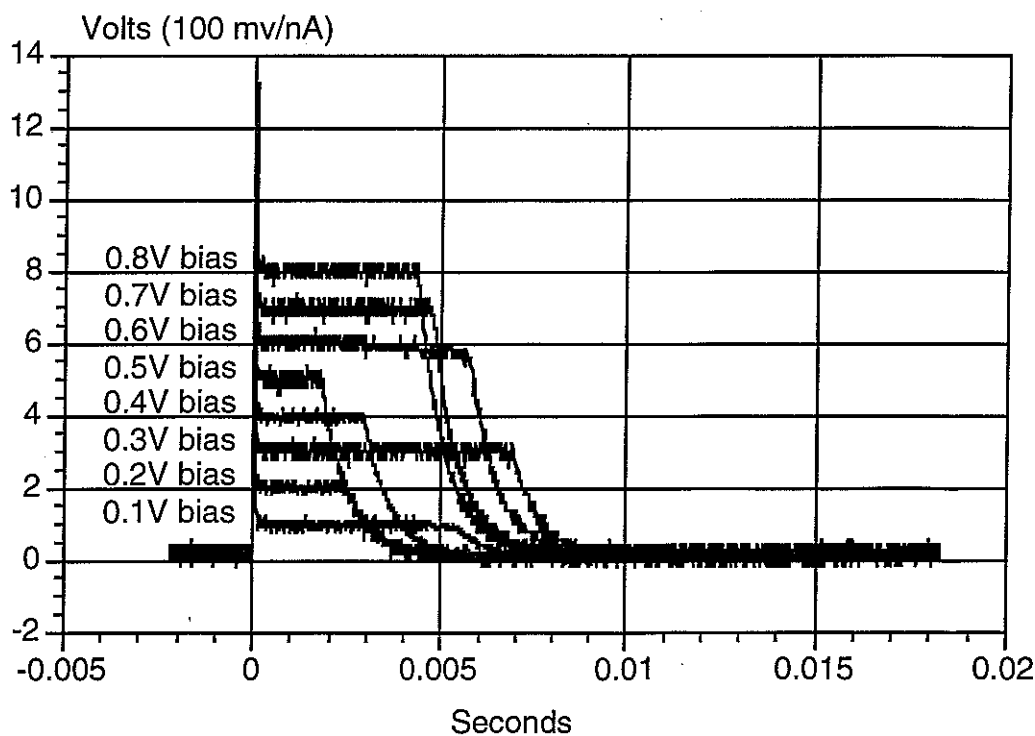


Figure 3.5 Flat events for different values of V_{bias} . The flat events always leveled off at the value $V_{bias}G_{preamp}$, indicating a low-resistance path from the tip to the sample for the duration of the event.

To explain the flat events, we suppose that the contact remains until the feedback circuit biases the z piezo crystal to apply enough stress to the break a filament formed between the tip and sample. Flat events always make the tip appear longer in the transient recordings, apparent in Figure 3.4 as a shift δV_{fb} down from the equilibrium V_{fb} by about 3 V or 28 nm; this shift represents the net difference between additions to the surface and changes to the tip length. Usually ($\approx 90\%$ of the events) this tip lengthening is confirmed by subsequent study of the topographical image in regions where there were no modifications. In no case was the tip found to be shorter after depositing a mound with a flat event. The lengthening and frequent sharpening of the tip after depositing material on the surface are consistent with the model that the tip bonds to the surface and then must be torn loose by breaking a filament, which presumably stretches (typically 30-40% in Au [AIP72]) before breaking.

The exponential events are characterized by a monotonic decay of the tunnel current after the pulse or by a ringing oscillation if the decay time-constant is long enough. This kind of event is also shown in Figure 3.4. The transient recordings from exponential events often indicate some shortening of the tip, which is confirmed in topography scan only about 50% of the time. The remainder indicate no change in tip length, although many of them indicate that the tip *shape* has changed. (Usually the tip gets slightly rounder as judged by a general loss of detail in the surface features.) We assume that these exponential events correspond to the field-evaporation events [Mamin91], *i.e.* loss of tip material will naturally occur where the field is strongest, where the tip is sharpest and where the topography image current is concentrated. About 25% of the exponential events deposit mounds and a few percent remove small mounds, but more than 67% of the simple exponential events make no detectable change on the surface. The same fraction of ringing or multiple exponential events make no change, with the remaining third randomly causing mounds to appear or disappear or causing pits to be formed by removing surface material from a smooth landscape.

Our experiments indicate that the kind of tip and surface modifications can be inferred from the type of transient recording that we observe. Nearly all of the flat events cause mounds to appear (many of them flat-topped), while only about one-third of the exponential or multiple exponential events modify the surface. To illustrate the correlation between tip change and transient event, we display a plot δV_{fb} taken from a large set of experiments in Figure 3.6. The lowest index (0) events are characterized by no apparent event in I_t . The next set (1 and 2) are simple exponential decays of I_t following the

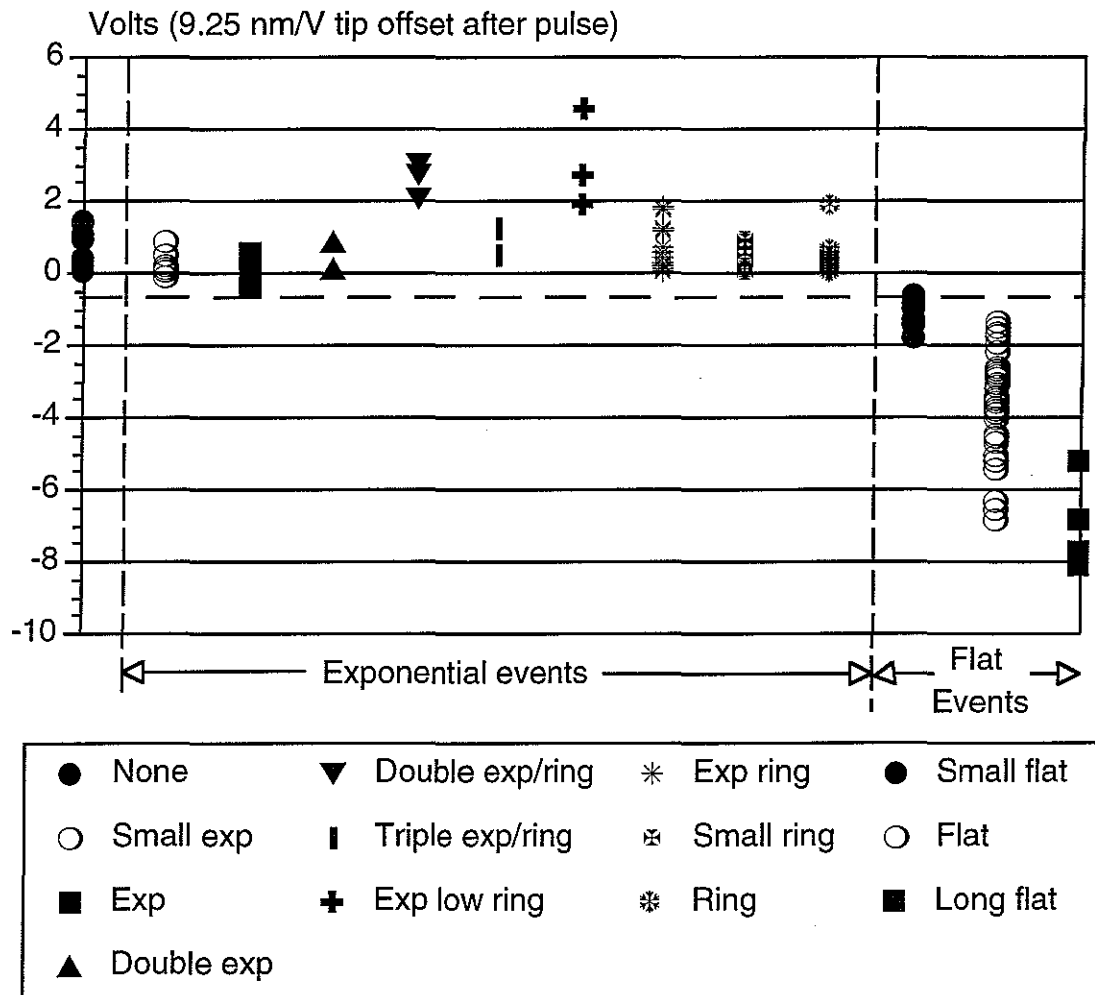


Figure 3.6 Tip offset by I_t result type. This graph shows that most exponential events had a net positive (towards the surface) offset after a pulse. All flat events had net negative offsets, with longer events having larger offsets. In particular, note that none of the exponential events had as large a negative offset as even the shortest flat events.

pulse, 3 - 9 are all ringing exponential events and 10 - 12 are flat events. There is a clear correlation between flat events and a net negative change in δV_{fb} and exponential (especially ringing) events and small positive changes in δV_{fb} . We have noticed (for single values of feedback gain and time constant) a linear correlation between δt_{flat} of the flat region in the tunnel current in flat events and δV_{fb} (see Figure 3.7). This correlation is sensible, since the breaking strength of the Au filaments is nearly a constant for all of the filaments.

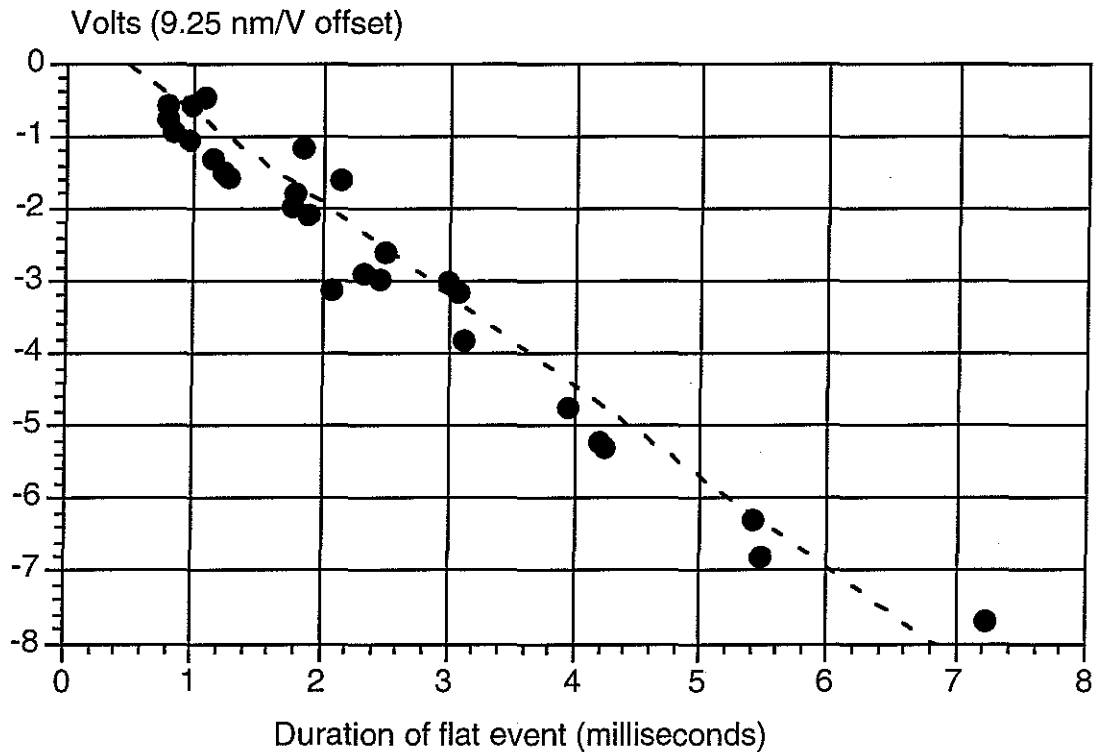


Figure 3.7 Tip offset after flat events. There was a linear relationship between the duration of the flat event and the distance the tip was offset after the event.

Our experiments indicate that the simple model of field evaporation of material from asperities on the STM tip cannot explain all events observed in our surface modification experiments. Our data imply that there are fairly frequent events in which the tip and sample surface are in electrical as well as intimate mechanical contact, and that the tip in fact bonds itself onto the sample surface only to be broken free by stress applied by the feedback.

Build an intentional structure on a surface — not yet

One of the initial goals for this project was to be able to make intentional structures with feature sizes of around 10 nm on a surface. Although we have made progress towards understanding the types of changes that can occur during surface modification attempts, we have not been able to control the type of change that occurs. While the user can adjust change attempts based on the results of earlier modifications, the changes are so random and varied as to make it impossible to create an intentional structure of this small size.

There have been times during experiments when conditions were favorable, and it was possible to draw a line or arc on the surface using repeated pulses, each of which removed a small amount of material. During these experiments, it was useful to be able to choose the location for each pulse based on the results of previous pulses.

Meanwhile, researchers at other laboratories have had spectacular success with surface modification — Eigler's group at IBM Almaden Research Center has written "IBM" and built an electron corral using individual atoms, Mamin's group at IBM Almaden has drawn a map of the world using gold mounds, and the Aono Atomcraft Project in Japan has succeeded in removing and replacing individual silicon atoms on a surface. [Crommie93] [Mamin91] [Aono93] These groups each worked with an STM that worked at ultra-low temperature, had a much wider scan range, or worked in ultra-high vacuum. Note that a virtual-reality interface could be added to any of the above instruments and would presumably augment their effectiveness just as it has augmented the effectiveness of the STM used by our project.

Contributions

The primary contribution of this dissertation to computer science has been the creation of the Nanomanipulator system, which presents a new paradigm for interacting with a scanning tunneling microscope. This paradigm can clearly be extended to other scanning probe microscopes, such as the atomic force microscope (AFM); in principle, it extends to control of any instrument whose interface is either indirect or unnatural for the user but that deals with objects (such as surfaces) that are familiar to the user. This

dissertation has shown that the computer can act as interpreter between user and instrument, and that as a result the effectiveness of instrument use can be improved.

The primary contribution of this dissertation to surface science has been the use of the Nanomanipulator system in the discovery of the graphite planes on HOPG and the discovery of a new mechanism for surface modification, nano-welding.

A final contribution of this dissertation has been the demonstration of a useful, working virtual reality system, both to the scientific community and to wider audiences. The system has used existing technology to successfully attack current problems in research, thus providing a real example of the types of benefits to be expected from such an interface. While we have shown that adding a virtual-reality interface to an STM is sufficient to provide a more powerful instrument; we have not shown that all of the elements of VR (in particular, immersion) are necessary. Though we believe that there are benefits to immersion (more intuitive navigation and pointing), it is possible that a fully-interactive, real-time, head-tracked, stereo, through-the-window system could have provided the same insights.

Lessons learned

During the course of this project, we have created several different ways to scan and view the surface. A question that we would like to ask the scientist is, "Which of these ways is best." The answer is often, "All of them." When we added additional scan patterns to the instrument, the scientists used all of the new ones and the old ones and then asked for more; the more ways to probe the same surface, the better! We provided both stereo HMD viewing and viewing on a projection screen — both are used. We allow pop-up menu selection of mode changes or verbal commands to a human assistant at a keyboard — both are used. The lesson here is that (especially for complicated interactions) providing multiple ways to perform the same action is usually better than any one of the choices.

It is easy to lose sight of the fact that scientists want **numbers**. Although improved shading and visualization can give good qualitative insights, serious study of a phenomenon requires quantitative measures of its

important characteristics. The place that this project has been the most lax has probably been our failure to provide quantitative measurement tools to the scientists — tools which they have cried out for many times over. The lesson here is to provide quantitative measurement tools from the beginning in any application designed to be used by a scientist. Examples of the kinds of numbers surface scientist are interested in can be found in [Eklund93].

IV. USER'S VIEW OF THE SYSTEM

An important part of the Nanomanipulator system is the view it presents to the user. The driving goal here was to provide an interface that allowed the scientist to interact with the surface itself, rather than with the computer. In a perfect system, the user would not realize at all that the computer was mediating for him.

The Nanomanipulator is designed to act as a translator, converting between the nanometer-scale actions of the STM and the meter-scale actions of the human operator. As shown in Figure 4.1, the user is presented with a three-dimensional image of the surface that is floating in space. A hand-held device allows the user to move, feel, or modify the surface. A pull-down menu system allows selection of the current mode of operation. Control panels, which float in the air about the user, allow adjustment of system parameters, including pulse parameters and surface representation.

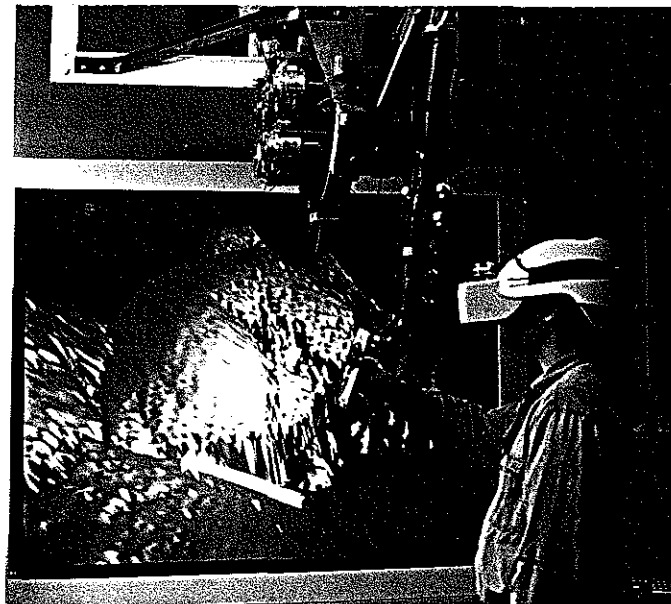


Figure 4.1 User interacting with the system.

Immersive interface (head-mounted display/ARM)

In order to see the immersive virtual world, the user wears a head-mounted display (HMD) and grasps the handgrip of an Argonne-III Remote Manipulator (ARM). (It is also possible to use the system by viewing the projection-screen display visible in Figure 4.1, but this is not described here.) The HMD displays two slightly different images to the user's two eyes, providing a stereo image that gives the illusion of looking at a three-dimensional space. While wearing the HMD, the user cannot see the world around him, but rather feels immersed in a computer-generated environment. Since the helmet is tracked by the computer, the display responds to the user's motion. This allows the user to move around in the world and have the objects respond to his motion. To him, it appears that he is standing in empty space, while nearby are his hand, a surface, and some control panels.

Headphones in the HMD allow the user to hear "auditory icons" presented by the computer. These icons indicate the mode of operation, voltage pulse firing, and the success or failure of actions taken by the user.

The user's hand is tracked by the ARM, and the computer draws an icon at the hand location that follows his motions. The user can position the icon by moving his hand normally. The particular hand icon changes to indicate what mode of operation the user is in. Buttons on the handgrip allow the user to effect actions in the world, such as moving things around or modifying the surface. The ARM is capable of providing force-feedback to the user (pushing back), so the user can probe the surface contours, his hand being pulled to the surface height.

Surface representations

The most important thing the user sees in the environment is the surface under study. When the user enters the space, the surface first appears as a flat plane floating in space. There is a directional light source attached to the user's head orientation, and the resulting specular highlights make it look as if the surface is made of shiny plastic. As the user tilts his head, the highlights change to match.

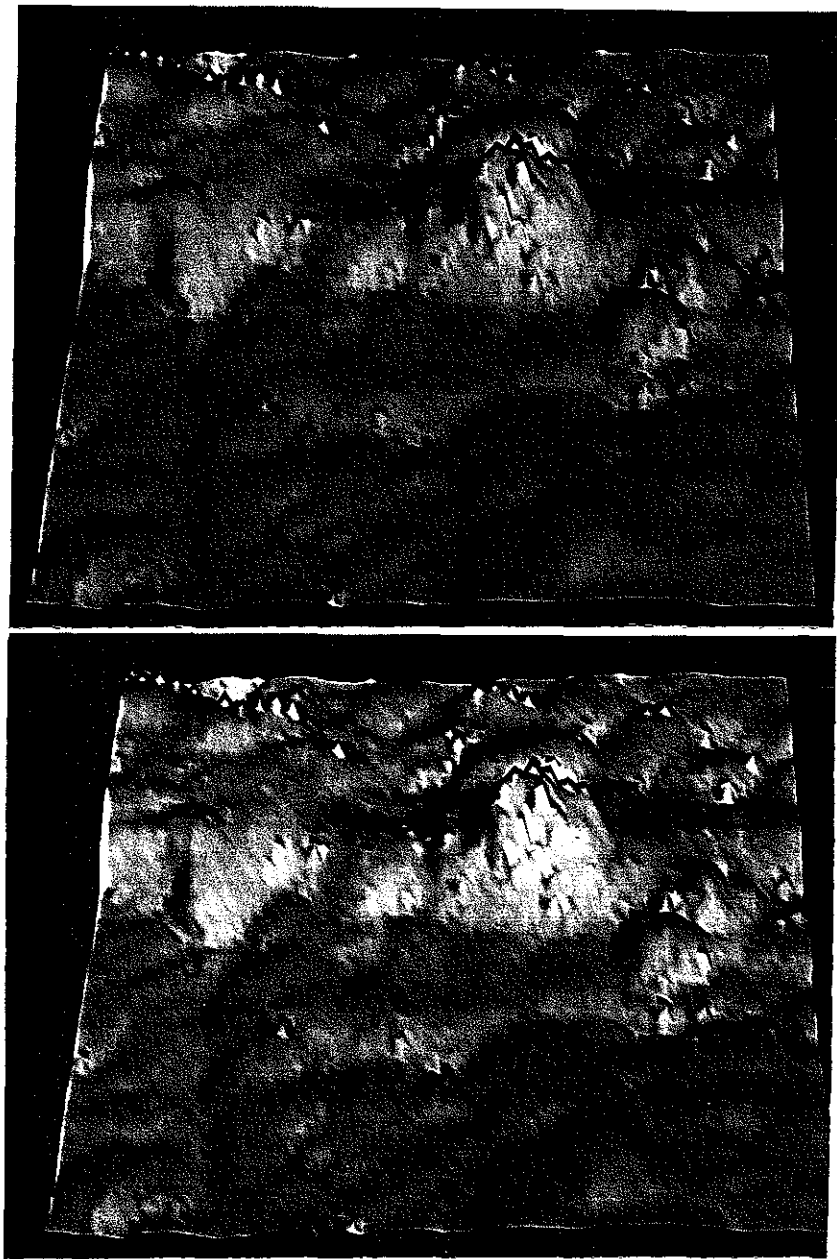


Figure 4.2 Surface coloring. The same surface colored first according to height and then according to standard deviation.

As the STM begins to scan and to send data points to the user interface, the surface starts to change. A scan begins at one end of the surface and sweeps across, leaving topography behind. The color of the surface is no longer uniform, but changes to show features of the incoming data. In about thirty seconds, the entire surface has been scanned and the user is viewing what appears to be a second-stage (positive) plastic mold taken of the surface. As the user moves to inspect the surface, shading and specular highlights

show clearly the contours on the surface, allowing him to understand better its shape.

Figure 4.2 shows an example of a data set using two difference coloring schemes. In the simplest case, the surface is colored according to height. In this representation, lower areas on the surface are bluer and higher areas redder. This color information is redundant, since the surface itself rises up in the high areas, and the shading also indicate slopes.

Another coloring scheme that is not redundant with topography is coloring by standard deviation. In this mode, several height samples are taken at each location on the surface with the tip held still in x and y during the samples. The surface is drawn passing through the mean of the samples and the color of the surface represents the standard deviation of the samples. In this mode, the standard deviation of the samples is calculated and then linearly scaled by a user-selected constant. The resulting value is clipped to the range $[0,1]$, and then the color is selected by linearly interpolating in RGB (Red/Green/Blue) color space from pure blue at 0 to pure red at 1.

Ideally, all of the samples would be at the same height, but since the feedback circuit that controls the height of the STM tip is not perfect it sometimes causes oscillations or jumps in the tip height. These jumps do not represent surface features, and they cause rapid vertical motion of the tip. The rapid tip motion causes a high standard deviation in the samples, so they show up as red areas on the surface. The red areas therefore indicate poor feedback tracking and alert the user that the data from those areas is suspect.

Two other data representations besides a shaded surface are available to the user of the system: spheres and dots. We find that these modes are not often used.

Sphere representation puts a large sphere at each data point such that the spheres from neighboring points overlap. This can avoid artifacts that are present in the polygonal representation and shows where each sample point is located while still presenting a closed surface. Of course, it adds many artifacts at the sphere intersections.

Dot representation displays each sample point as a small sphere. Spheres for neighboring points do not overlap. This provides a surface representation that is free of aliasing artifacts caused by interpolating between data points, and also provides a surface in which nearer parts do not occlude further parts.

Modes of operation

Since there are seven operating modes and the hand-held control only has two buttons on it, it is necessary to have some sort of modal system. The system uses the thumb trigger to access a menu system that allows changing of the mode and the finger trigger to cause action in whatever the current mode of operation is. There is a different hand icon for each mode of operation, so the user can always tell quickly what action will be taken by the finger trigger. The modes are broken down into two sets: those that affect only the display of the virtual world and those that send commands to the STM.

Virtual-reality modes

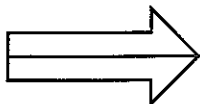
There are several modes that affect the translation, orientation, and scale of the user with respect to the surface; these are **scale up**, **scale down**, **fly**, and **grab**. None of these modes affect the operation of the STM; each of them moves the user about within the world, or the world about the user. These modes are all common actions to take in a virtual world. [Robinett92]



In **scale up** mode, the hand is represented by two green arrows pointing away from each other. In

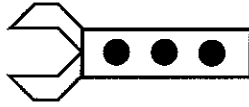


scale down mode, the icon becomes two red arrows pointing towards each other. In either mode, the size of the surface with respect to the user changes by 10% per viewing frame (50 msec) as long as the finger trigger is held down. This allows the user to select the apparent size of the surface — adjusting it from matchbox size up to several meters across. This is usually the first mode chosen by a user, as it has a great effect on how much of the surface can be reached and what kind of interactions are feasible.



In **fly** mode, the hand is represented by a white arrow that rotates to match the orientation of the ARM's handgrip.

When the user holds down the finger trigger, he is translated in the direction that the arrow is pointing. This mode allows the user to fly above (or below) the surface and get views from various locations. It is most useful when the displayed surface is scaled to be large with respect to the user.

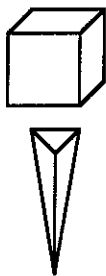


In **grab** mode, the hand is represented by a robot hand. While the finger trigger is depressed, the surface representation moves about as if it were being held by the user. That is, it is translated and rotated so that its position relative to the hand remains constant. The icon does not have to be in contact with the surface in order to grab - the interface acts as if there is an invisible link (like a glass rod) between the hand and the surface. This allows the user to move the surface to a convenient location or to rotate the surface so that the lighting best reveals surface features. It is most useful when the surface is scaled to be about the same size as the user or smaller.

In order to show an indication of scale, a 2-D grid of lines marks off the surface in 10 nm squares. The height of this grid relative to the surface can be adjusted by the user, and it can be used to get a rough idea of the size of features on the surface. There is another user mode, similar to grab mode, that allows the user to adjust the height of this measurement grid; this allows the user to move the grid to line up with a surface feature to be measured.

STM Control modes

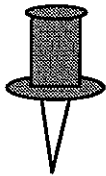
There are three modes of operation that control the operation of the STM itself; these are **touch**, **select scan**, and **pulse**. Each of these modes causes commands to be sent to the STM when the user takes action.



In **touch** mode, the hand is represented as a small white cube. Also visible in the scene is a thin white triangle that represents the tip location on the surface. When the user holds down the finger trigger, a force is applied to pull his hand (indicated by the cube icon) to the top of the tip (indicated by the triangle icon). This allows the user to feel the height of the surface at a given location. To feel other locations, the user moves his hand laterally across the surface.

What actually happens when the trigger is held down is that the STM is asked to take a sample of the surface height at the (x,y) location of the user's hand. Once the height of the surface at that location is known, the tip icon is moved just above the surface there. At this point, a simulated spring force is used to pull the user's hand to the end of the tip. The force pulls up if the user is below the surface and down if he is above.

There are several notable things about this mode. The first is that the sample location need not lie on the grid of samples that are taken during standard STM scanning, so that this mode can be used to supersample the surface. The second is that the user at no time controls the z position (height) of the tip above the surface - this is controlled by the electronic feedback circuit. Rather, the user feels a force that brings his hand into compliance with the tip height. The last is that the simulated spring force is fairly weak, producing a somewhat "soft" surface. This is due to the difficulty of presenting hard surface forces when the force information is updated at only 20 frames per second. [Kilpatrick76]



In **select scan** mode, the user's hand icon is a pushpin. This mode allows the user to specify that a particular subsection of the grid is to be sampled and the rest of the grid should remain unchanged. This mode is useful because it takes the STM about 30 seconds to scan the entire surface. When making changes, it is sometimes desirable to look at a small section of the surface more frequently. In this mode, the user presses the finger trigger at one corner of the region, drags his hand to the other corner, and then releases the trigger. From then on, the STM updates only that rectangular subset of the entire surface, and the rest of the surface is drawn as it was before the scan selection.



In **pulse** mode, the user's hand icon is a lightning bolt. This is the mode that allows the user to modify the surface. The user moves his hand to the location over the surface where a voltage pulse is to be fired and then presses the finger trigger. At this time, the STM tip is moved to the indicated location, the voltage pulse is fired, and the system goes back to scanning the surface. While the user is positioning his hand, a green targeting line continuously indicates the location on the surface at which the pulse would be fired. When the pulse

is fired, a yellow line appears that pierces the surface at the pulse location to indicate exactly where the pulse was fired. The user can fire repeated pulses by pressing the finger trigger once for each pulse.

Menu system

A pop-up menu system is provided to allow mode changes and other operations. Whenever the user presses the thumb trigger on the handgrip, the main level of the 2-D pull-down menu system appears in front of him, within his reachable volume. The user can move his hand around to select entries, which causes submenus to appear. The user makes his selection by placing the handgrip in the box with the desired function and releasing the thumb trigger. The box that the user is currently in is highlighted so the user always knows what choice he will be making. Figure 4.3 shows the menu system with the file submenu chosen. The available main-level menus are File, VR modes, STM modes, and Display.

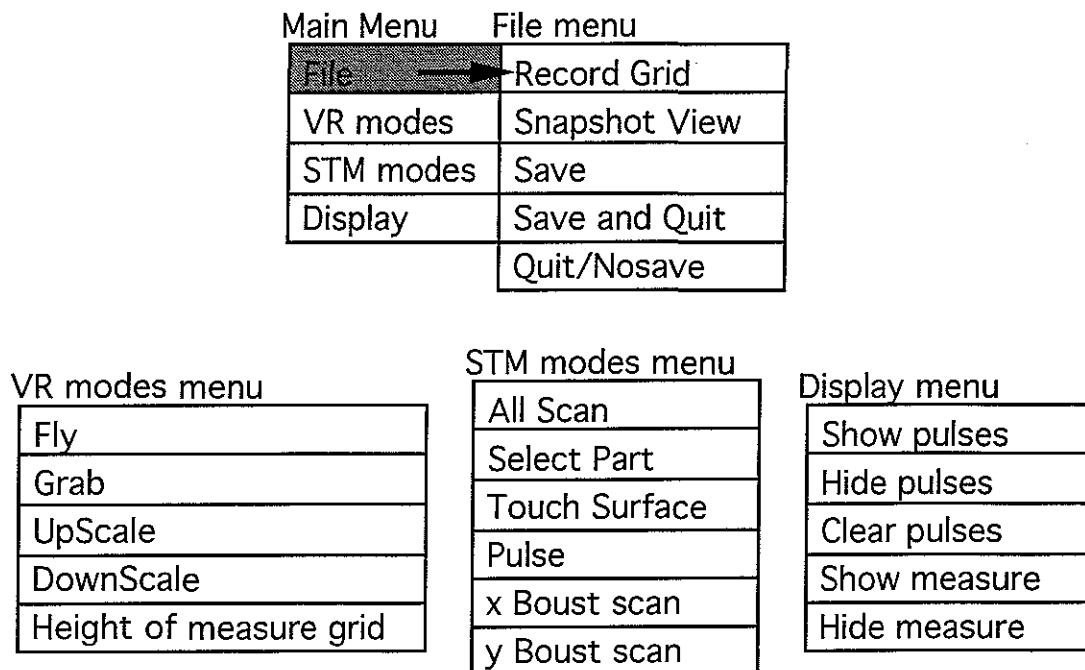


Figure 4.3 Menu system with File submenu selected. The other three submenus are shown below it.

The **File** submenu deals with saving information and quitting the program. The *Record Grid* entry stores the current grid of STM data in a

memory buffer that can be later saved to disk. The *Snapshot View* entry stores a copy of the image on the screen into a disk file. The *Save* entry stores any buffered snapshots to disk one at a time, in FIFO order. This is a separate command because storing the files to disk interrupts the smooth motion of the user through the world while it takes place. The *Save and Quit* entry saves any memory buffers and then exits the program. The *Quit/Nosave* command exits the program without saving any memory buffers.

The **VR modes** submenu allows the user to select among the various virtual world display change modes described above. It includes *Fly*, *Grab*, *UpScale*, *DownScale*, and *Height of measure grid*. The last mode allows the user to adjust the location of the measurement grid to allow coarse measurement of surface features.

The **STM modes** submenu allows the user to select among the various STM modes described above. These include *Touch Surface*, *Pulse*, and *Select Part*. In addition, this submenu allows the user to select the primary scan direction for the STM (either x or y) and has a shortcut entry, *All Scan*, which tells the STM to select the entire grid for scanning. The *x Boust scan* and *y Boust scan* entries cause the system to scan in either the X or Y dimension most rapidly. *Boust* is an abbreviation for boustrophedonic, which means “as the ox plows.”

The **Display** submenu allows the user to select which parts of the system are to be visible. The *Hide Pulses/Show Pulses* entries control the visibility of the markers that are displayed when a pulse is fired. There is another entry, *Clear Pulses*, that erases all of the past markers that have been displayed but that still allows future pulse markers to be visible (useful for clearing out the markers for a series of pulses when a new series is to begin). The *Show Measure/Hide Measure* entries control the visibility of the measurement grid described above. Hiding this grid removes screen clutter when absolute measurements are not important to the task at hand.

Control panels

There are two control panels floating in space within reach of the user. These panels maintain their location in the physical space around the user and are unaffected by scaling or flying. These control panels become active

whenever the user puts his hand icon near one of them. The various controls in the panels respond to the user pressing the finger trigger with the hand icon inside the control. The control panel responds when the hand is inside it no matter what the current mode of operation is. This means that the controls can be adjusted at any time.

Pulse parameter control

The first control panel, shown in Figure 4.4, is used to control the STM bias voltage and pulse parameters. The three dials on the left side of the panel allow the user to adjust the bias voltage, pulse duration, and pulse voltage value. The user can press the finger trigger while the handgrip is in one of the dials and rotate his hand to adjust the setting.

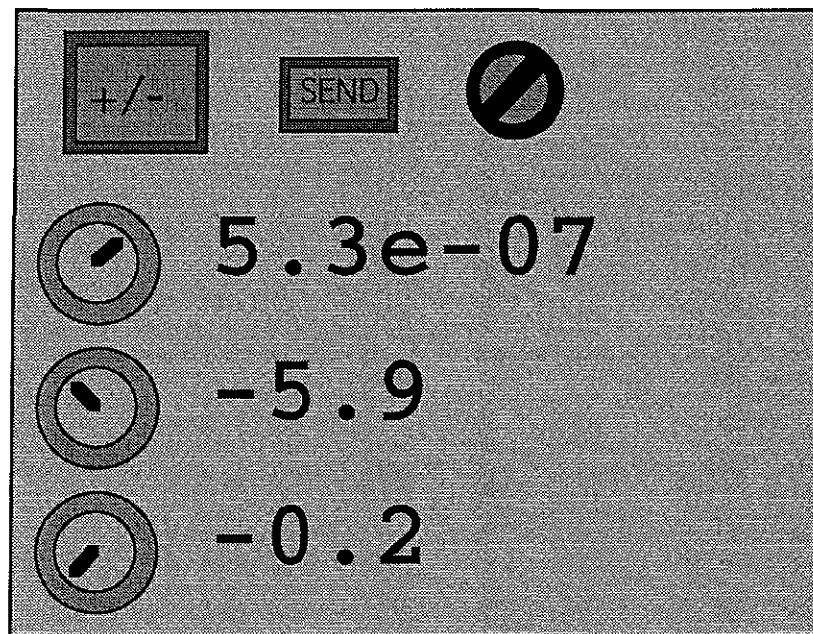


Figure 4.4 Pulse control panel. These settings indicate pulses of 530 ns duration, jumping from a bias value of -0.2 volts up to a peak value of -5.9 volts.

There are also three buttons on this control panel. The +/- button is used to switch the polarity of the bias and the pulse peak. The **send** button causes the current (adjusted) settings of the control panel to be sent to the STM controller. None of the aforementioned changes to bias, polarity, and so on take effect until this button is pressed. This also causes a message to be printed

to the console indicating the new settings, and records the new settings in the data file associated with the current program run. The **cancel** button causes changes in the control panel settings since the last **send** to be ignored. The control panel dials revert to the settings that were last sent to the STM.

Sampling control

The second control panel, shown in Figure 4.5, is used to control the parameters described in the **Surface representations** section. This control panel has three dials and two buttons.

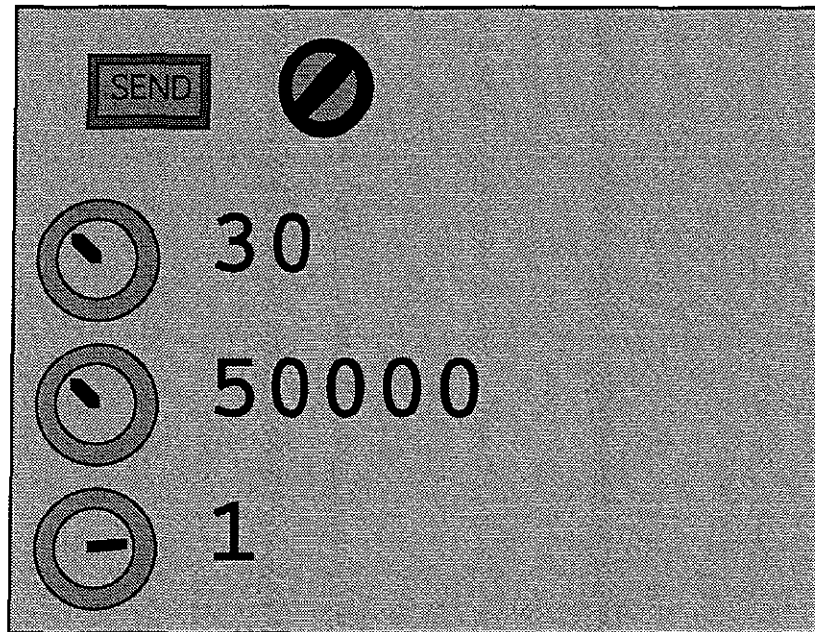


Figure 4.5 Sampling control panel. These settings indicate 30 samples taken at a rate of 50 kHz with a scale of 1 on the standard deviation.

The three dials correspond to the number of samples to be taken at each data point, the frequency at which they should be taken, and the scale to apply to the standard deviation when representing it by color. Setting the number of samples to one causes the surface to be colored according to the height of that sample. Setting the number of samples to more than one causes the sample to be colored according to the standard deviation of the samples. The scale factor multiplies the standard deviation of each set of samples; the result is clipped to the range $[0,1]$ and specifies the surface color. Blue areas on the surface

represent low standard deviation, red surface areas indicate high standard deviation, and purple areas indicate intermediate standard deviation.

The **send** and **cancel** buttons behave like those on the pulse control panel.

V. SYSTEM HARDWARE

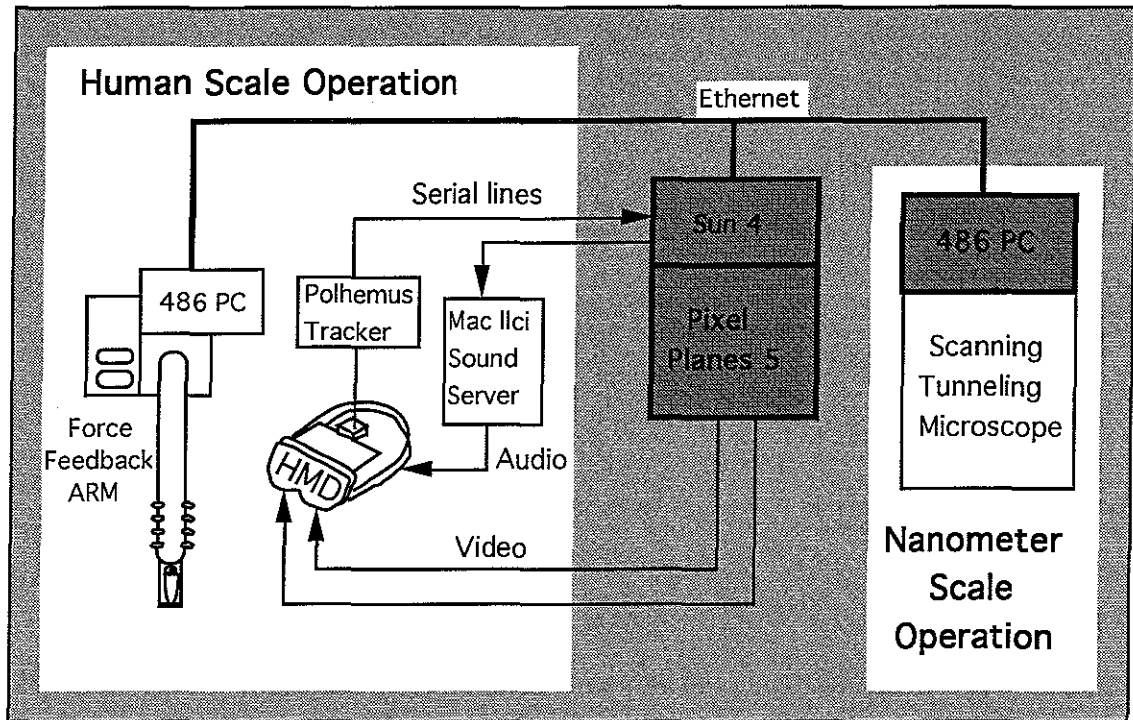


Figure 5.1 System hardware architecture. The VR interface running on the Pixel-Planes host mediates between the nanometer-scale operations of the STM and the meter-scale operations of the user. This interface communicates with the STM and ARM controllers using TCP ethernet connections, and with the Polhemus head tracker and Macintosh-based sound server via serial lines.

The Nanomanipulator system consists of several subsystems, each running on its own dedicated processor. This is done in order to meet all of the different types of constraints on the system. For example, the part of the system that controls the STM must have rapid Analog-to-Digital (A/D) and Digital-to-Analog (D/A) conversion and precise control over voltage pulses, whereas the part of the system that displays images to the user must be able to perform extremely rapid polygon rendering operations to maintain the images seen by the user. Rather than attempting to meet these and other goals

using one complex system, we have chosen to partition the problem into parts that need only a small amount of communication between them and to run the separate parts on different machines, each appropriate to the task it performs.

The system can be loosely divided into STM control, user interface, and glue. The user interface and STM control are described below. These two parts communicate with each other using a TCP (stream) connection over a 10Mbit/sec ethernet LAN (see Figure 5.1).

The part of the system that handles this and all other communication between subsystems runs on a Sun 4/280 processor. The main job of the Sun is to transfer information from one subsystem to another, sometimes translating the information as it does so. The Sun communicates with the PC that controls the STM (hereafter pc_stm0) and the PC that controls the ARM (hereafter pc_arm0) using TCP ethernet connections. It communicates with Pixel-Planes 5 using a specially designed host interface that connects the Sun's bus to Pixel-Plane's ring network. It communicates with the Polhemus tracker and the Macintosh-based sound server (not shown in the figure) using serial connections.

User interface hardware

User interface hardware includes the equipment in the lower two divisions of Figure 5.1, as well as some devices that are not shown in that figure. There are five major divisions of the hardware:

- devices used to track the user's position
- devices used to read user button presses
- devices used to generate the images seen by the user
- devices to present force output to the user
- devices used to present sound output to the user

Trackers

The purpose of the tracker subsystem is to report the positions of the user's head and hand. The head in our system is tracked using a Polhemus 3Space tracker, and the hand is tracked using either the same tracker or the ARM.

Polhemus tracker

The 3Space tracker is one of several magnetic trackers produced by Polhemus. These trackers work by having a source transducer at a known position that radiates a time-varying magnetic field. There are one or more sensors (wire coils) in which currents are induced by the varying fields. By measuring the effects of the fields on the coils, it is possible to calculate the position and orientation of the sensors relative to the source transducer.

We attach one sensor to the HMD worn by the user and (optionally) one to a hand-held controller. Knowing the position and orientation of the source and the relative positions of the sensors to the source allows us to determine the position and orientation of the user's head in the room. This allows us to generate views in the virtual world that correspond to the user's motion in the real world — thus making it seem as if the user is present in the virtual world.

The 3Space tracker includes an onboard microprocessor to convert the magnetic field information into position and orientation. This information is sent to the Sun 4 host processor over an RS-232 serial communications line. The 3Space is capable of providing this information about 60 times per second for one sensor and about 30 times per second for two sensors. We operate the tracker in continuous mode, in which it asynchronously sends reports as soon as they are available. The Sun 4 reads all of the reports and uses the most recent as the current position.

ARM

The ARM is a six-degree-of-freedom mechanical device whose joint angles are measured using rotary potentiometers. This is the same device described in [Taylor90], modified to use potentiometers rather than synchro transformers. The A/D and D/A processing required to control the ARM is performed by cards installed in pc_arm0. The values used by these cards are transmitted to and from the Sun 4 using a TCP ethernet connection.

The working volume for the ARM (before it hits mechanical constraints) is about a one meter cube. This volume is sufficient for most work, but its limitations are an inconvenience to the user.

Buttons

The buttons in the system are used to provide selection and command capability for the user. The method of reading the button values depends on the device that the buttons are on. As described in the software section, there is a library that hides these differences from user code. The two types of buttons used in the Nanomanipulator system are on the ARM and on the Python-3 hand-held controller.

ARM

There are two buttons on the ARM: a finger trigger and a thumb switch. Each switch acts as a momentary contact, reporting one value when pressed and another when released. The buttons on the ARM are connected via a resistor network to one of the A/D lines on the PC controller. Thus, their values are passed to the Sun 4 along with the values that determine the ARM position.

Python-3

In addition to the ARM, we also have another hand-held controller. To make this device, we mounted a 3Space sensor in the handgrip of a Python-3 Nintendo joystick controller. This controller resembles a pistol grip with five buttons arranged on it. The only buttons we use in the Nanomanipulator system are the finger trigger and thumb button (in order to keep the system the same when using the ARM or the Python controller).

The buttons in the Python controller are read using a single-board computer produced by Z-World Engineering. This controller has a built-in serial port that it uses to report the values. Due to an insufficient number of serial lines on the Sun 4, we connect this single-board computer to a serial port on another machine and use a TCP ethernet connection between this machine and the Sun 4 to send the values. This two-hop method does introduce some lag into the button events, but it is not noticed by the user.

There is no force-feedback capability when using the Python controller, so it is not possible to feel the surface when using this device. The

range of motion of the device is much greater than that of the ARM, however, so it is sometimes more convenient to use.

Image generation

Image generation is done on Pixel-Planes 5, which is a high-performance graphics engine developed at UNC under the direction of Henry Fuchs and John Poulton. [Fuchs89] The system is hosted from a Sun 4 processor that handles creating and updating the display list. The NTSC video from Pixel-Planes 5 is sent to the HMD and a projection screen via video cables.

Host

The host for Pixel-Planes 5 is a Sun 4/280. The connection from the Sun to Pixel-Planes is via a *Host InterFace* card (HIF) that plugs into the VME bus on the Sun. Ribbon cables connect this card to another card that is on the ring communication channel for Pixel-Planes.

Pixel-Planes

Pixel-Planes 5 consists of several Intel i860-based *Graphics Processors* (GPs), highly parallel *renderer* boards, Frame Buffers, and HIFs connected together on an eight-channel ring network (see Figure 5.2).

Each GP board in the system contains two i860s, each with 6 MB of local RAM. The primitives (triangles and spheres) associated with each object to be displayed are distributed among the GPs, so that each GP handles a part of every object in the scene. The GPs handle traversing the display list, applying transformations to all the primitives, and determining which parts of the screen a given primitive covers. They then form command lists for the renderer in each region of the screen to draw the primitives. A typical system contains about 20 Graphics Processors.

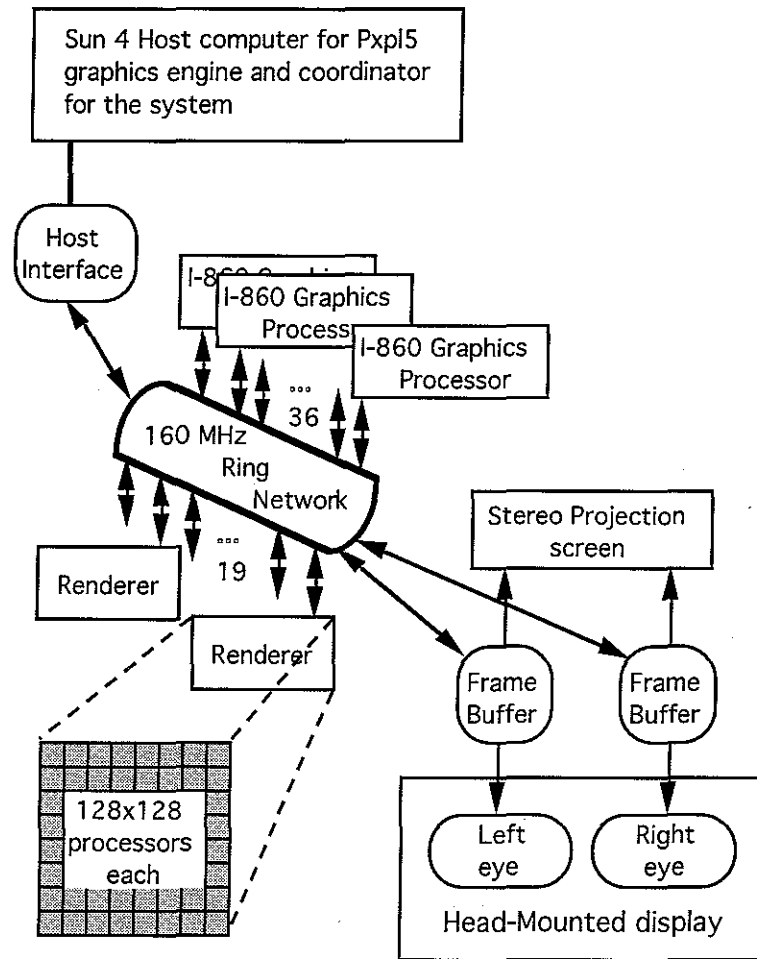


Figure 5.2 Architecture of Pixel-Planes 5.

The renderer boards are made up of processor-enhanced memories and quadratic expression evaluators. Each renderer board contains 128x128 processors, each of which handles one or more virtual pixels. As expressions pass through the expression evaluator on the way to a given processor, the parameters for that processor's (x,y) location in screen space are filled in. In this way, each processor "knows" its location. To draw a triangle, commands are sent to enable only those processors that lie on the correct side of the three lines making up the triangle, and then those processors color their pixels according to a quadratic expression. This allows Phong-shaded triangles to be drawn. The renderers perform Z-buffering on incoming primitives, so that once all the primitives have been processed, the closest one at each pixel is visible. Once a region is finished being rendered, it is sent to the frame buffer. A typical system contains about 16 renderer boards.

The frame buffers accumulate the regions as they arrive from the renderer boards and then scan the data out in an appropriate video format. There are currently three types of frame buffers available: 1280x1024 high-resolution RGB frame buffers, 640x512 NTSC frame buffers, and 640x480 field-sequential color boards. Video lines lead from the frame buffers out to screens where the image is displayed. Stereo is implemented using two frame buffers; the picture for each is computed in a separate pass through the display list with different viewpoints for the two passes.

HMD

There are several different head-mounted displays in use in the department. They each have two displays, one per eye, that show the contents of one of the frame buffers. They also contain optics to present the images as if they are located some distance from the user's eyes, so that they are in focus. The unit most used for the Nanomanipulator project is the Flight Helmet, made by Virtual Research, Inc. This HMD uses two Liquid Crystal Display (LCD) screens with resolutions of about 200x140 pixels spread over about a 90 degree field of view.[Holloway93] A diffusive surface is placed on the LCDs so that the image is blurry rather than a collection of separate dots. The optics used in this display produce significant distortions of the screen. These distortions do not seem to present a problem when trying to view a scene, but presumably have effects on the user's perception of the scene; these effects have not been characterized.

Projector screen

The video signal that is sent to the left eye of the HMD is also sent to a Sony rear-projection monitor whose display is in front of the user. This display has a resolution of about 640x512 pixels, which is much higher than the image seen in the HMD. This allows others to see what the user is seeing during an experiment, and also allows the user to remove the helmet and look at the screen to discern features that require higher display resolution.

It would be possible to display head-tracked stereo images on the screen rather than in the HMD (using a high-resolution frame buffer and a CrystalEyes stereo display system), but this has not been implemented. As the

display resolution of future HMDs will increase, this will become a less important option.

Force display

A motor is mounted on each joint angle on the ARM, allowing forces and torques to be applied to the end effector. Thus, the ARM is capable of both measuring the position and orientation of the handgrip and applying forces and torques to the user.

The maximum force we allow the ARM to produce is about 20 Newtons (about the weight of a gallon of milk). There is a foot pedal that provides a hardware fail-safe; when the user is not stepping on the pedal, the current to the motors is shut off to prevent the ARM from generating any forces.

Sound

Sound in the system is provided by a dedicated Macintosh computer. The current system is a Macintosh IIfx with 8MB of RAM and an 80MB hard drive. Sounds to be played are pre-recorded and stored by name in files on the hard drive. The Macintosh communicates with the user interface through a serial line. Commands sent from the user interface cause specific sounds to be played by the Macintosh. The sounds are played in speakers on the HMD. The sounds are currently monaural. They are basically auditory icons that tell when certain events occur.

STM control

The STM in place at UNC was developed at UCLA and shipped here. This section describes the instrument we got from UCLA, for which we developed some additional electronics.

There are two main tasks in controlling an STM. The first is to get the appropriate control signals to the STM and sense the outputs. The second is to keep unwanted signals and other effects away from the STM. An IBM PC/AT controller with associated hardware is used to provide the signals, and various hardware is in place to shield the system from unwanted effects.

Personal computer controller

Control and sensing of the STM is controlled by a Northgate PC/AT computer with an Intel 486-DX2/50 processor. The chassis is a tower with a 300W power supply and several 16-bit expansion slots. There is a local hard disk for storing the STM server program and a 3Com 3C507 ethernet card for communication over the building ethernet LAN. The console of the PC is used only for program development. During normal use, the display is turned off to reduce electrical interference, and the server program communicates with the user interface over the ethernet. See Figure 5.3 for a diagram of the system.

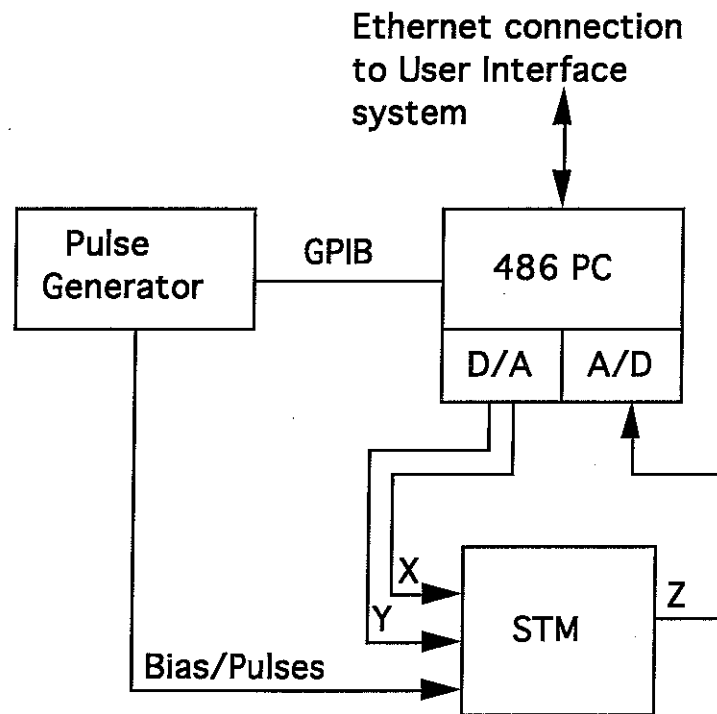


Figure 5.3 Architecture of PC STM controller subsystem.

The system includes several additional expansion cards and peripheral devices to allow it to perform its control and sensing operations. The basic operation are scanning the STM tip, sensing the height of the tip at each point, generating pulses to the STM, and observing the results of the pulses on the tunnel current and tip height. Each of these is discussed further below.

Scanning control and height measurement

The analog inputs and outputs needed to control the scanning and sensing of the STM tip (as described in chapter 1) are provided by a Data Translation DT2838 board. This board offers up to 8 16-bit differential inputs, which can be scanned at a maximum rate of 160K samples/second. It also provides two 16-bit outputs, which can be updated at up to 100K samples per second each. See the following section on noise in the system for a description of the noise present on the outputs while switching. Another disadvantageous property of this particular card is the fact that it takes 1 ms to start any A/D or D/A operation. Since we operate the system one scan step at a time, this is a severe limit on the number of steps per second our system can scan. Single-stepping is done because during some modes it is not possible to know ahead of time which direction the next step is to take.

The two analog outputs are used to control the (X,Y) position of the tip. The output of each DAC is fed into a high-voltage amplifier, which linearly amplifies the +/- 10V signal up to +/-100V to drive the piezo crystals. See the section on noise for a description of the filtering done on the signals before they are fed into the amplifiers. In order to avoid presenting sharp step changes to the crystals, the DAC takes many small rapid steps between one scan position and the next. The Direct Memory Access (DMA) mode of the DT2838 is used to send these at 100 kHz.

The voltage that the electronic feedback circuit in the STM presents to the Z piezo is sensed using one of the differential inputs on the DT2838. This tells us the height of the tip. When multiple samples are to be taken at a single scan position, DMA transfer is used to acquire the samples at the rate requested by the user interface.

Bias/pulse generation and delivery

Controlled voltage pulses to be added to the bias in the STM (which allow surface modifications as described in chapter 1) are provided by a Hewlett-Packard 8116A pulse/function generator. The 8116A is capable of producing pulses with durations of from 10 ns up to 1 second. The pulses can be of either polarity and up to a maximum of 16 volts. (Normally, the instrument drives up to 8 volt pulses into 50 ohm loads. Since we are driving essentially an open

circuit, these voltages are doubled.) See the section on noise for a description of how the output from this instrument is coupled into the bias voltage for the sample.

The PC controls the 8116A through its IEEE-488 (sometimes called GPIB or HP-IB) parallel interface bus. An expansion card provides the PC with an IEEE-488 connection that is used to control the 8116A.

Post-pulse tunnel current/position

The response of the tunnel current and feedback voltage immediately after a pulse is captured using a Tektronix DSA 602 Digital Signal Acquisition unit. The pulse waveforms are stored on a local PC-format floppy disk on the instrument itself and are later transferred to the rest of the system through the PC. There is an IEEE-488 port on the DSA 602, so it is planned that the instrument will some day be controlled by the PC and send its information directly through this bus.

Vibrational, thermal, and electrical isolation

Since an STM is sensitive to distance changes on the order of Ångströms and currents on the order of nA, it is crucial that the instrument be isolated from outside effects that would show up as noise on the signal. Most of the isolation in our system is a duplicate of the setup in place at UCLA. Figure 5.4 shows the vibration-isolation system.

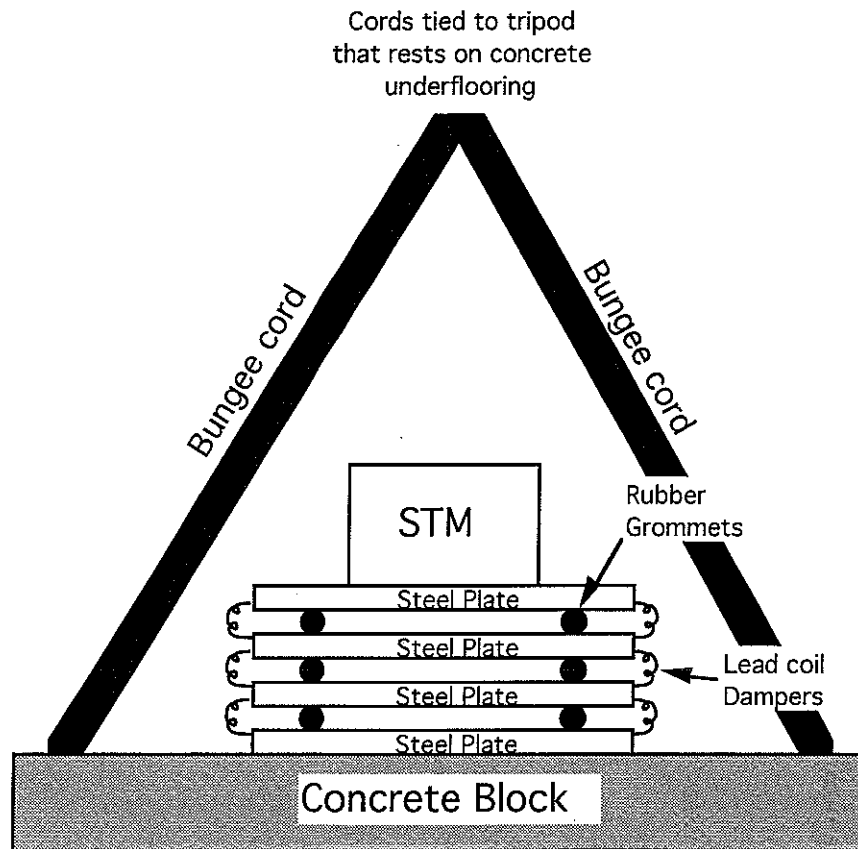


Figure 5.4 Vibration isolation system for our STM.

In order to provide vibration isolation, the STM is built on top of a mass-spring system that consists of a stack of steel plates with rubber supports between the sheets. These plates are also connected to each other with lead coils, which act to damp vibrations. All wires that connect to the instrument are then glued to these plates so that they do not couple vibrations to the scan head. The unit sits on a concrete block that is suspended from a tripod by bungee cords. There are lead bricks on the concrete to provide additional mass. This mass-spring system has a resonant frequency of about 4 Hz. The tripod sits on the concrete under-floor in our building and is located right next to one of the main support columns. The vibration isolation seems very good at high frequencies; pounding on the floor with a heavy hammer produces no noticeable change in the tunneling current. However, vibrations at about 4 Hz couple to the scan head; a person walking nearby produces visible ripples on very flat surfaces.

To provide thermal isolation, the entire assembly above is contained in a plywood box. Foam insulation has been placed around the tripod legs to prevent air from coming in through the bottom. We find very little changes in scanning due to thermal drift once the box has been closed for about an hour.

Electrical isolation is achieved by using shielded wires to connect to the various parts of the STM. This helps reduce crosstalk between the high voltage drive lines and the sensitive tunnel current and bias lines. To prevent coupling external electrical noise into the system, copper mesh has been placed around the inside of the box to form a Faraday cage. The cage is grounded to the same ground used by the feedback electronics and the control and measurement electronics, including the PC controller.

Sample-advance motor

When the STM was moved from UCLA, it had a manual advance micrometer head that was used to bring the sample into close contact with the tip. Successfully bringing the sample into range without crashing it into the tip required about half an hour of very delicate taps on the micrometer head, and it also required that the box holding the STM remain open. This resulted in many crashed tips and a large temperature change in the system when the sample was brought in. As the temperature drifted within the box, the box had to be reopened periodically to reposition the sample.

In order to overcome these difficulties, we purchased a "Motor Mike" motorized micrometer head from Ariel that had the same dimensions and range of motion as the original manual micrometer advance. We then designed and built an amplifier box to provide the control signals to the motor along with a hand-held switch box to allow the operator to provide command input.

The hand-held controller is used to specify the desired speed of the Motor Mike. It provides a voltage output that is proportional to the desired speed of the motor. The box is essentially a voltage divider. It includes a potentiometer to adjust speed and two buttons to specify forward and reverse. When one of the buttons is pressed, it switches voltage (+15 V for forward, -15 V for reverse) to the control potentiometer. The other end of the potentiometer is grounded. The control signal is picked off the wiper of the potentiometer, allowing adjustment from 0 V to nearly full range.

Micrometer motor manual control

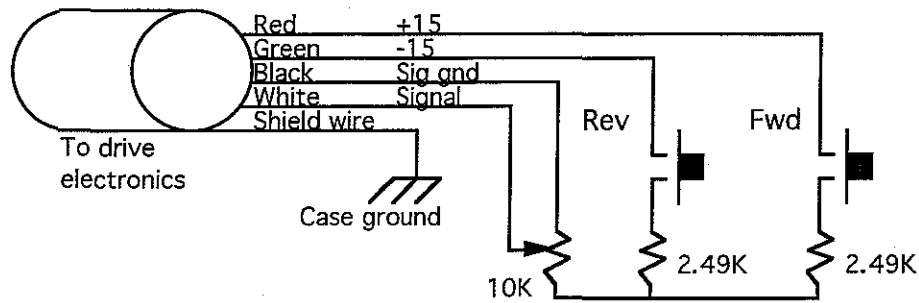


Figure 5.5 Hand-held controller for sample advance motor. The cable connects to the drive electronics shown in Figure 5.6.

The amplifier/control box drives the Motor Mike at a velocity that is linearly related to the voltage input from the hand-held control unit. The controller comprises three basic components. The first is an analog multiplexer that can allow multiple inputs (hand-held controller, computer, etc.). This is to allow the easy addition of computer control at a later time. The second component is a pair of amplifiers that limit the output of the multiplexer to maximum/minimum values. These values are adjustable with trim pots on the board and are used to set the maximum forward and backward speeds of the motor. The third component is a driver amplifier, which provides drive current to the motor. It also includes a compensation network that modifies the output signal to provide quick starts and stops of the motor.

See Figure 5.6 for a diagram of the drive electronics and Figure 5.5 for a diagram of the hand-held control unit.

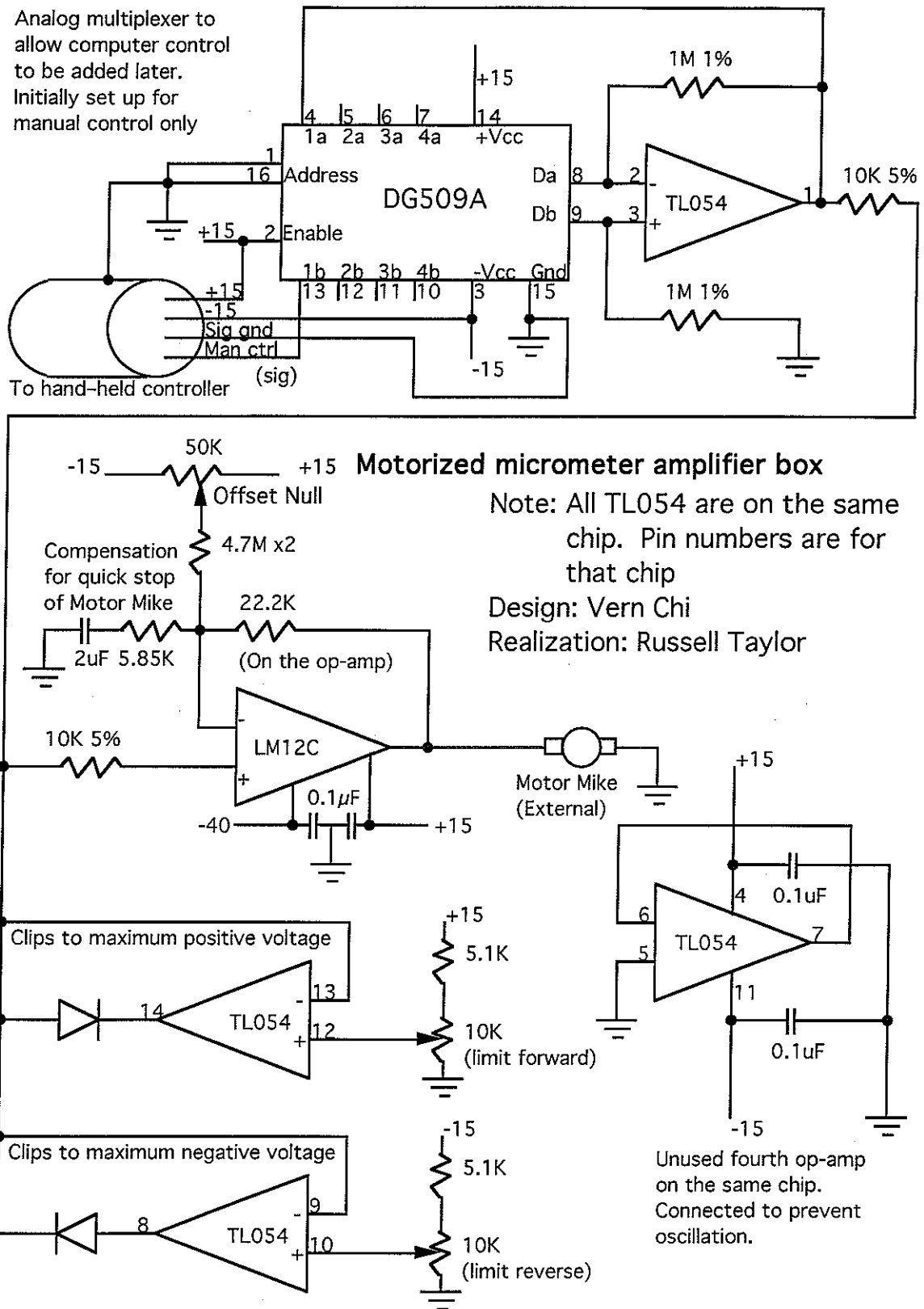


Figure 5.6 Drive electronics for sample advance motor.

Control of the tip-sample gap

An electronic feedback circuit is used to maintain the tip-sample separation that will provide a constant tunneling current as the tip is scanned in x and y. The voltage that the feedback circuit applies to the z piezo can be monitored to determine the tip z position, and thus the surface height (or more precisely the work function isosurface height). Figure 5.7 shows a diagram of the major components of this feedback system.

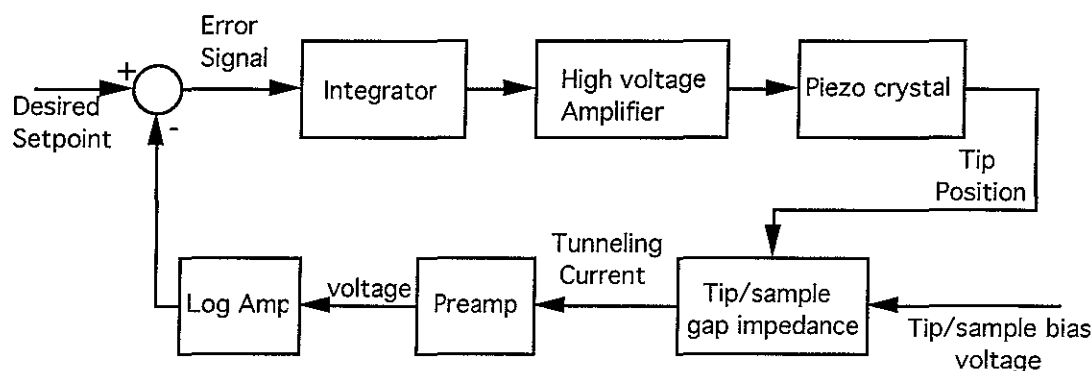


Figure 5.7 Tunnel-current feedback controller diagram.

The Integrator is present to make this a type 1 control system, which is able to follow a constant input with zero error.[D'Azzo88] Driving a voltage on the piezoelectric crystal causes motion of the tip towards the surface; it stretches about 1 nm per volt applied to it. The high-voltage amplifier provides a driving voltage up to 200 V to cause crystal elongation of up to 200 nm. The tip-sample gap is modeled as described in chapter 1; to enable use of linear feedback control system techniques, a logarithmic amplifier (log amp) compensates for the exponential behavior of the gap. The pre-amplifier (preamp) transduces the tip current into a voltage of sufficient level for transmission to the rest of the electronics.

During the course of investigation, we made modifications to our STM. The initial system had been developed and built at UCLA and was shipped to UNC complete with electronic feedback control. In order to improve performance, we developed another feedback control loop for this STM, under the direction of Vern Chi.

In order to design a stable feedback control system for the Z-axis piezo, one must know the transfer functions for all the components in the feedback loop. We used the circuit simulator HSPICE™ to model the behavior of the feedback loop to help design a stable system. [HSPICE92] This meant that we needed HSPICE™ models for each of the system components.

Characterizing system components

The manufacturers supplied models for the components used in the integrator, log amplifier, and high-voltage amplifier. We used the equation for tunneling current given in chapter 1 to describe the tunneling junction. We used an HP4195A network analyzer to measure the transfer functions for the piezo crystals and preamp circuit and used these measurements to make HSPICE™ models of these components.

Crystal resonances

Figure 5.8 shows a diagram of the piezoelectric crystal that controls tip height. We control the length of the crystal by controlling the voltage between the electrodes. Positive voltages cause extension of the crystal and negative voltages cause contraction.

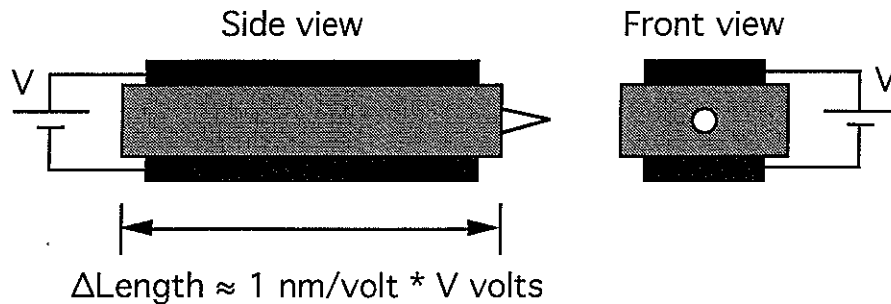


Figure 5.8 Piezoelectric crystal. The piezoelectric crystal that moves the tip has electrodes on two sides. As the voltage between the electrodes changes, the length of the crystal changes.

Just as a crystal dinner glass rings with a characteristic pitch when struck, each piezoelectric crystal has certain frequencies at which it resonates. There are two effects from these resonances. The first is that the circuit driving the crystals sees a different load as the frequency of the driving signal moves near a region of resonance. The second is that the gain

of 1 nm per volt changes with frequency, especially near resonance (See Figure 5.10). These effects cause distortions in a driving signal that has frequency components near a crystal resonance frequency, since these components of the signal are exaggerated. In order to avoid driving the crystals near resonances or to compensate for the resonances, it is necessary to know how the crystal behaves across frequencies.

An HP4195A network analyzer was used to characterize the load presented by the crystal to the driving circuitry. We measured the coefficient of reflection from the Z-axis piezo while sweeping the driving frequency from close to DC through 300 kHz. We found a fairly flat response (real part of coefficient close to 1.0) up to about 160 kHz, then a sharp drop (real part of coefficient down to 0.35) centered around 165 kHz. By 180 kHz, the response had returned to 1.0. There were smaller dips at higher frequencies. Since all these natural crystal frequencies were well above other resonances in the feedback loop, we have chosen not to incorporate them into our models.

There are other system vibrational resonances that can be measured when the system is in tunneling conditions. These resonances are at lower frequencies than the native crystal resonances, presumably because they involve the mass of the crystal mounts and possibly other parts of the system. The effect of these vibrational resonances is much more critical to the system, since they are at lower frequencies. These resonances depend not only upon the crystal geometry, but also on the mount used to hold the crystal.

To measure the vibrational resonance on the STM, it was necessary for us to treat the entire piezo, tunnel junction, and preamp unit as a black box; this was due to the difficulty of making measurements at intermediate points. The setup we used to measure the system is shown in Figure 5.9.

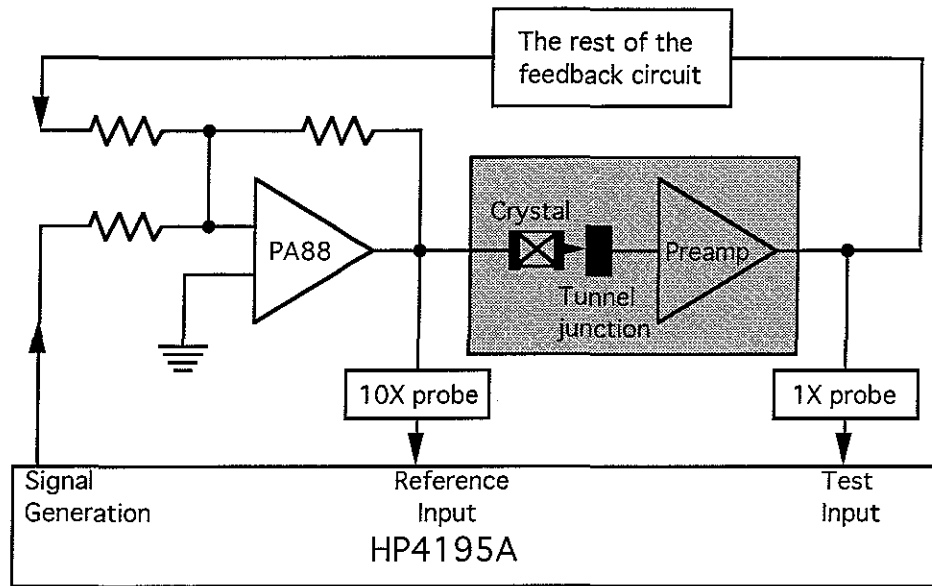
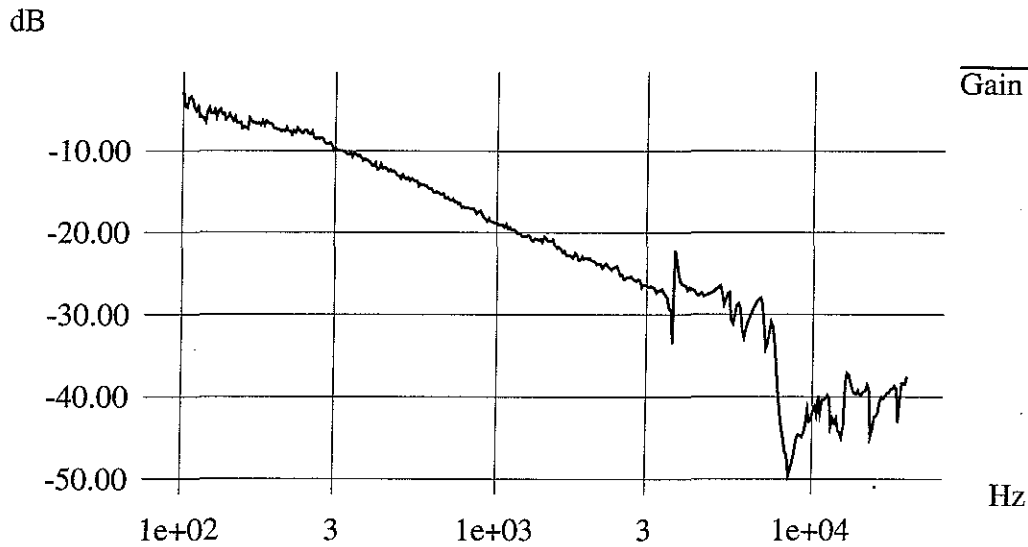


Figure 5.9 Piezo/preamp test setup. Test setup used to measure resonances of piezo and preamp circuits in the STM. The network analyzer sweeps the frequency of the signal generator and measures the difference between the reference input and the test input, thus producing the transfer function for the crystal, tunnel junction, and preamp.

Since we cannot separate the effects of the piezo resonances from those of the tunnel junction and preamp, we are forced to make some assumptions about the latter. Our first assumption (based on the specifications for the amplifiers used) is that the preamp does not affect the response for frequencies lower than about 100 kHz, which is well above the range of interest. The second assumption was that the tunnel gap transfer characteristic is independent of frequency and is approximately linear for small signals. We checked the linearity assumption by performing multiple tests at various settings for the feedback system parameters. During these tests, linear changes in feedback parameters produced linear changes in system response — indicating that tunneling conditions did not distort them.

The gain and phase for the transfer function for the Z piezo/preamp combination are plotted in Figure 5.10. Note that there is a linear rolloff out to about 3.6 kHz, at which point there is a resonance. At higher frequencies, the response gets highly irregular, so the graph is not shown above 20 kHz. This

Transfer function of piezo-preamp



Transfer function of piezo-preamp

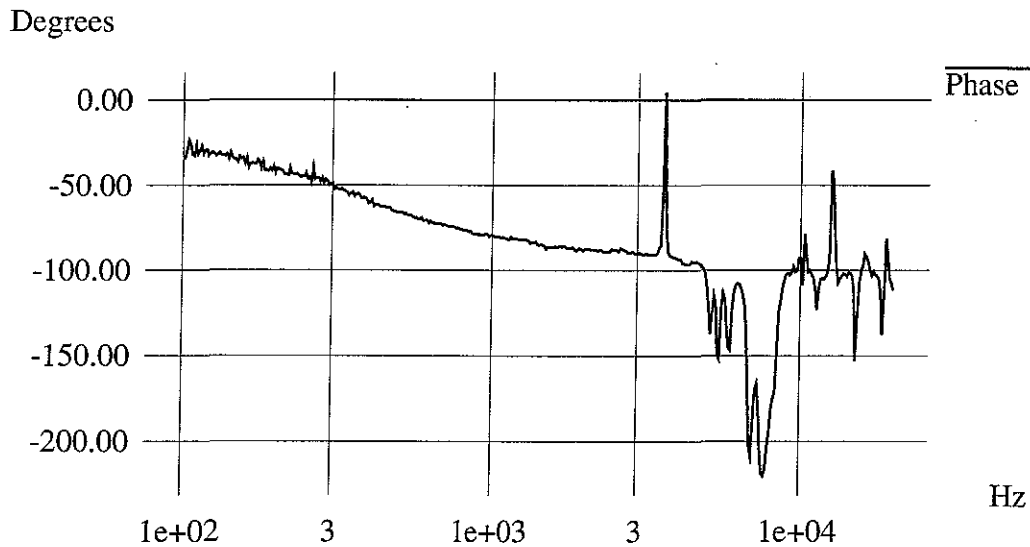


Figure 5.10 Piezo transfer function. Transfer function of the piezoelectric crystals and tunneling current preamplifier on the STM.

graph shows that it is not feasible to drive the feedback circuit with frequencies approaching 10 kHz, and indeed not above 3.6 kHz without compensating for that resonance.

Pre-amplifier

As noted above, a small-signal measurement of the transfer function of the pre-amplifier circuit was included in the testing for the crystal transfer function. The pre-amplifier is a transimpedance amplifier that provides 100 mV output per 1 nA of input current.

Compensating for tunneling nonlinearity

Since the tunneling junction is in the feedback loop and its effects on the tunneling current are highly nonlinear, compensation must be added to let us design the whole system as a linear feedback system. This is the function of the logarithmic amplifier in the circuit.

Since the approximate equation for tunneling current i based on tip-sample distance D is $i = (1/D) * \exp(D)$ in its basic form (see chapter 1), to achieve linearity would require compensating both the $1/D$ portion and the $\exp(D)$ portion of the equation. We have compensated the $\exp(D)$ portion using the logarithmic amplifier in the circuit, but have not dealt with the $1/D$ portion.

Ignoring the $1/D$ dependency makes the system diverge from linear behavior as D becomes very small (as the tip and sample come into extremely close proximity). This condition only occurs when the feedback is not tracking properly, as the logarithmic amplifier output will already have clipped at its maximum value due to the increase in tunneling current.

The exact equation for tunneling current based on a complete understanding of the tunneling effect is not known, so even a circuit that perfectly compensated the above equation might not necessarily match actual tunneling conditions.

HSPICE™ simulations allow adjusting the feedback loop parameters (including gain) until the system is stable under conditions that match the expected tunneling gap conditions during actual use.

Performance of the new feedback circuit

Testing the performance of the original and new feedback circuits on different surfaces indicates that the new circuit causes the feedback system to track with fewer occurrences of oscillation. This allows the controlled scanning of all surface that the original circuit could scan as well as additional types surfaces that caused instabilities in the original circuit.

Noise in the system

A Tektronics DSA 602 digital processing oscilloscope was used to measure the noise present on various signals in the system. This instrument produced both a digitized version of the signal of interest and its Fast Fourier Transform (FFT). The FFT was useful because it indicated the frequency components of the noise. Since the piezo crystals resonate at certain characteristic frequencies, the FFT indicated how much of the signal was likely to couple into our signals of interest. For example, noise signals near the 3.6 kHz resonance would be amplified by 100 dB more than signals at 3 kHz by the crystal resonance (see Figure 5.10). The input was DC-coupled, high impedance, for most of the tests. The tests of the Y DAC while the STM was scanning were AC-coupled with high input impedance. The FFT data had DC suppression (the DC component of the signal is set to 0) and the signal was averaged over a number of scans, with the FFT taken on the averaged signal. Only the magnitude of the FFT data was examined. The DSA 602 was plugged into the same ground and power connections as all other circuits being tested.

See appendix B for the graphs of system noise and its FFT data. The values on the FFT graphs are in dBV (dB relative to 1V). The noise is described quantitatively below.

In order to determine the baseline noise from the DSA 602 itself, data was first taken with its input connector open, and then with a coaxial cable with its far end open attached to the input (the same cable was used to connect to the various components being tested). Two more tests, with a 75 ohm terminator attached to the input and then to the far end of the cable, were also run. These tests revealed a baseline noise in the system of about 0.6 mV peak-to-peak. This was wideband white noise for the most part, although the measurements taken with the open input had spikes of noise at 35 kHz and

harmonics. The test with the open-ended cable attached showed signals of around 3 mV peak-to-peak when the cable was moved or struck. This was presumably due to piezoelectric effects of the cable's insulation. This noise did not appear when the cable was terminated, and so presumably would not appear with the low-impedance sources that were tested.

The next tests were of the battery and Hewlett-Packard HP8116a pulse generator that are used to provide bias voltage and pulses to the STM. The battery had about 0.9 mV peak-to-peak noise, which is only slightly above the baseline noise of the DSA 602 and cable. The noise was nearly white with slight spikes at 35 kHz and harmonics (presumably due to the baseline peaks on the DSA 602). The HP8116a produced about 75 mV peak-to-peak white noise. Since noise on the bias couples directly into the tunneling current, it would show up on the output as an offset to the surface height. Since the bias voltage is a linear term in the tunneling current equation, the error in tunneling current scales linearly with the bias voltage error.

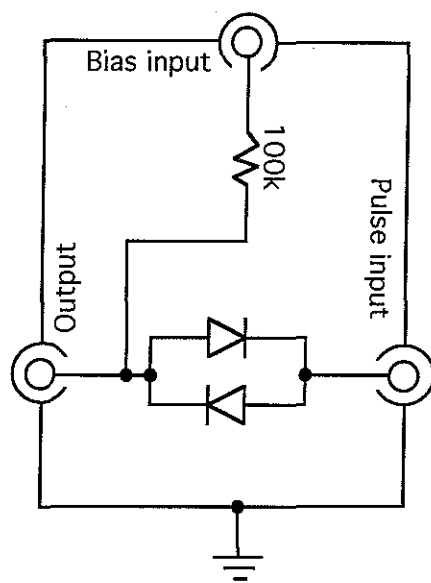


Figure 5.11 Bias noise reduction. This is the schematic of the bias/pulse mixer circuit to reduce the amount of noise from the pulse generator getting into the bias.

As an expedient to reduce the coupling of noise from the pulse generator into the bias, a battery reference voltage was used for the bias and a switching combiner circuit built to couple the two voltages (see Figure 5.11).

The diodes isolate the output from the pulse input unless it differs from the bias input by more than about 0.6 volts. If the difference is more than 0.6 volts, one of the diodes is turned on and passes the pulse (including any noise) through to the output. The tunnel gap resistance is much greater than 100 K Ω , so when the diodes are off, the 100K resistor has little impact on the voltage at the output, since most of it is dropped across the tunnel gap. This circuit reduced the noise on the output from 75 mV to about 5 mV peak-to-peak (a factor of 15 improvement). This was white noise.

The next error source studied was the Z-axis coarse adjustment. The output from this circuit was summed with the feedback signal and sent to the Z-axis high-voltage amplifier. The error measured was about 16 mV peak-to-peak. This signal had a strong frequency component at 585 Hz, which showed up clearly in the time-domain signal (it accounted for about 7 mV of the total signal). The FFT also showed smaller spikes at 20 kHz, 42 kHz, and 50 kHz. Since the gain through the amplifier is 10 and the piezo crystal moves about 9.8 Ångströms per volt applied, this corresponded to about an 1.5 Ångstrom uncertainty in the Z piezo location. Since the Z piezo crystal has fairly linear response up to about 3600 Hz, the response to the component at 585 Hz was probably adding a 0.7 Ångstrom modulation to the tip position. Since we no longer make use of the coarse Z adjustment, we removed this noise by disconnecting it from the circuit.

The third error source studied was the X/Y position outputs from the DACs on the PC. We use a Data Translation DT2838 board to provide both 16-bit A/D and D/A. There were both static and dynamic errors from this system. When the DACs were not changing, but simply presenting a constant voltage, there was about 50 mV peak-to-peak noise present in X and 40 mV noise in Y. For a scan area of 2000 by 2000 Ångströms, this corresponds to a position uncertainty of 6 Ångströms. Since we sample the grid with at most 128 samples on a side (typically only 80), a 50 mV error would not take the tip further than half the distance to the next scan point (it moves only 6 Ångströms out of about 15). There was an 8-10 dB spike in the FFT for the x DAC at 3300 Hz and in the y DAC at 6800 Hz. Since the piezo assembly has resonances at 3600 Hz and higher, these couple strongly to the STM tip position and could cause larger than 6 Ångstrom oscillations in the tip position when the sample is moving.

Since most of the noise appeared to be concentrated in narrow spikes, we attempted to reduce it by low-pass filtering. Using a 20 kHz single-pole low-pass filter removed the spikes from the data and left noise of only 3 mV peak-to-peak. This noise had a strong component at 180 Hz and was synchronized with the power line. The filter pole was still several decades above the frequencies of interest when scanning, since we scan at most ten lines per second. For this reason, it should not affect the signals of interest.

The most surprising noise we found appeared when switching the DACs. Since the STM scans in x fastest, this most clearly appeared when we examined the Y signal while the X DAC was changing. The DAC was changed by making 128 “moves” per second. Each move consisted of about 50 small steps, taken at 50 kHz. This was intended to allow the voltage to vary smoothly, so that the tip was not moved in one rapid jump from one location to the next. In order to allow motion in an arbitrary direction, the Y DAC is sent values at the same time the X DAC is. In this case the original value was always sent, so in theory the Y DAC should have maintained its value. Thus, the two possible sources of the noise were crosstalk between the two DACs and switching noise on the Y DAC when it was repeatedly given the same value.

This switching noise is detailed in the last several graphs in appendix B. The noise appeared to be made up of clusters of short-duration spikes that were each of up to 100 mV height and of either polarity. In order to remove the noise, we applied a 5 kHz single-pole low-pass filter to the DAC output before using it to drive the system. This filtering reduced the amount of noise to about 10 mV peak-to-peak. The resulting noise looked like square waves of about 10 mV amplitude. This was an improvement of about a factor of 10.

VI. SYSTEM SOFTWARE

Surface sampling and reconstruction

This section provides an introduction to some issues involved in discrete sampling of a continuous surface and in reconstructing a continuous surface from the samples. Note: the word *continuous* here means that the surface has a height value at every point (X,Y) within the plane.

The term *C0 continuity* refers to the surface value at a given point being infinitely close to the values at neighboring points (see Figure 6.1). [Bartle82]

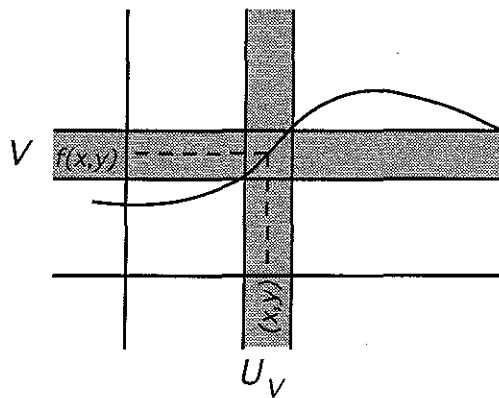


Figure 6.1 C0 continuity. Let $A \subseteq \mathbf{R}^2$, let $f: A \rightarrow \mathbf{R}$, and let $(x, y) \in A$. We say that f has **C0 continuity** at (x, y) if, given any neighborhood V of $f(x, y)$ there exists a neighborhood U_V of (x, y) such that if $(x, y) \in A \cap U_V$, then $f(x, y)$ belongs to V .

The term *C1 continuity* refers to a continuity of surface slope, which will also be referred to as the *smoothness* of the surface; C1 continuity of the surface is equivalent to C0 continuity of the surface derivative. Note that the surfaces under study need not, in general, have either C1 continuity or C0 continuity (a cliff on the surface has neither type of continuity).

The STM provides a collection of regularly-spaced samples of surface height. Each sample is taken (for an ideal tip) at a single point, rather than being an average over some area. This means that our information about the actual surface is limited to knowing its height at isolated locations, as shown in Figure 6.2. The height samples along one line through the surface are also shown. The height of the surface between the point samples is not directly measured and must be inferred from the height at the sample locations.

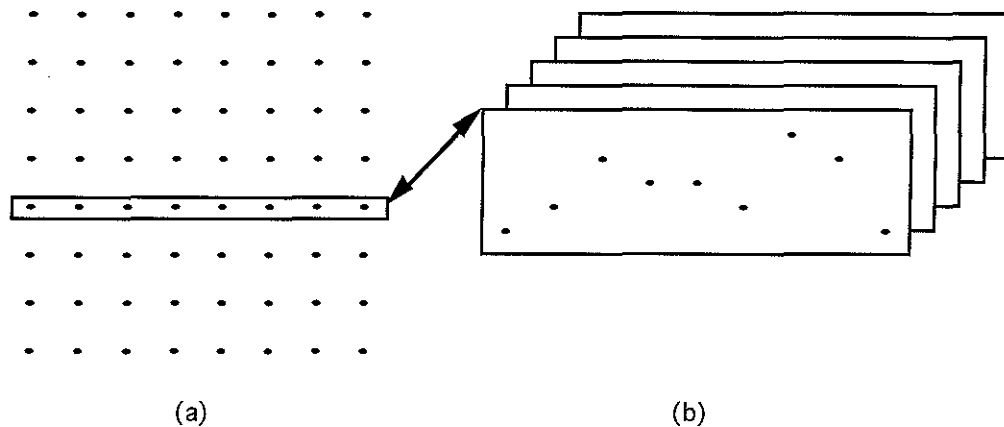


Figure 6.2 Point sampling. The point samples of height information received from the STM. (a) shows the locations of the samples on the surface, as seen from above. (b) shows one line of samples from the grid, viewed from the side.

To reconstruct a continuous surface from the point samples, we draw triangles between adjacent samples as shown in Figure 6.3. This is equivalent to doing linear interpolation to find the heights of points between the samples. The figure also shows the reconstruction for a slice through the surface. This reconstruction provides us with an estimate of the surface height at all points on the surface, rather than only at the points we have sampled. This allows us to draw a complete image of a surface on the screen, rather than a collection of isolated dots. We have chosen this method of reconstruction because the graphics system we use is capable of drawing the triangles quickly enough to draw this reconstruction in real time; it is not able to produce more sophisticated surface representations (such as splines) for data sets as large as ours in real time. The effect is to approximate the point-sampled surface with a fitted surface that has $C0$ continuity, but is not smooth.

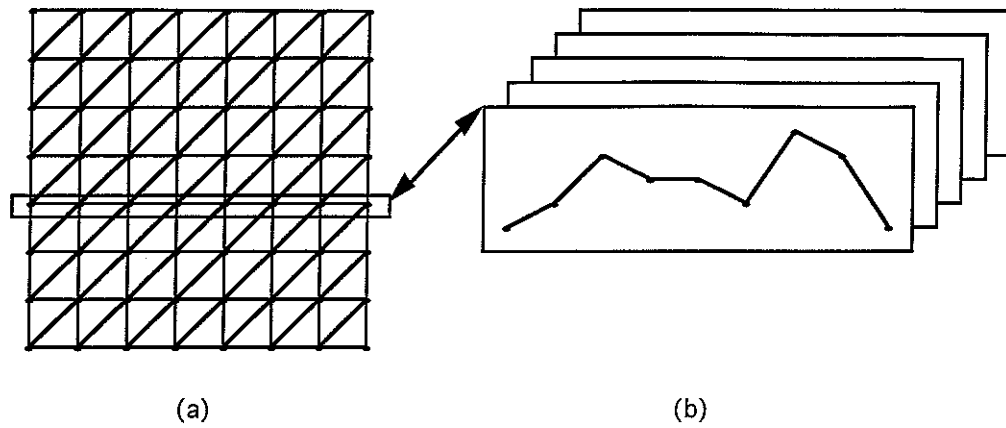


Figure 6.3 Triangle tessellation. Reconstructing the surface from the samples using linear interpolation. (a) shows the triangles used to reconstruct the surface. (b) shows the effect of this interpolation on a slice of samples.

The fact that the Pixel-Planes 5 graphics library is display-list oriented led us to use triangles that all cut the squares in the same direction. A more sophisticated reconstruction might select triangle orientation (diagonal slanted to the left or right) within each square based on the characteristics of the samples at the corners or based on the viewing direction.

Sampling

When doing reconstruction of any kind from point-sampled data, the assumption is made that the underlying surface was sampled finely enough to catch all important details. If this is not the case, then it is impossible to know which surface to reconstruct from the given samples. Figure 6.4 shows an example of this ambiguity, where two different surfaces produce the same set of height samples at the same sample locations.



Figure 6.4 Under-sampling. Two different surfaces that produce the same set of sampled height values when sampled at the same locations.

An examination of the sampling rate required to capture all features of a surface is most easily done in the frequency domain.¹ In order to capture all features in a signal, the sampling rate must capture information about all of the signal's components in the frequency domain (Fourier transform) representation. In particular, it must capture information about the highest-frequency component of the signal; the sampling theorem states that a signal can be correctly (and uniquely) reconstructed from a set of point samples whose period of sampling is less than half the period of the highest-frequency component present in the sampled signal. [Oppenheim83] Thus, the frequency of sampling must be greater than twice the frequency of any component making up the signal (this frequency is known as the *Nyquist rate*).

If the signal is sampled with a sample frequency that is below the Nyquist rate, the signal will be distorted by the sampling. In particular, it will be impossible to correctly reconstruct the high-frequency components of the sample that lie above half the sampling rate. The primary effect of this will be loss of detail in the reconstructed image; sharp edges will become rounded and small features will be overlooked by the sampling.

If loss of detail were the only result of under-sampling, this would be tolerable; the larger features are probably of more interest in large scans anyway. The problem is that the sampling can introduce *aliasing*; in this case a high-frequency signal appears to be a lower-frequency signal when sampled. The high frequency is said to have *aliased* to the lower frequency. An example of this is seen in Figure 6.5, where the reconstructed surface has fewer peaks per unit length than the original, sampled surface. In signal processing, it is customary to filter the signal to remove the frequency

¹ A signal can be broken down into a set of sine waves of various frequencies, amplitudes and shifts using the Fourier transform. [Oppenheim83] The sum of these sine waves forms the signal. The Fourier transform of the signal contains the coefficients (magnitude and phase) of the sine waves making up the signal; this collection of coefficients is called the *frequency-domain* representation of the signal. The signal itself, varying with time, is said to be the *time-domain* representation.

components above the Nyquist rate, so that they are not present in the sampled signal; this prevents aliasing. Since the surface is presented to us only as a collection of point samples, this method is not available to us. When doing point sampling, the only way to avoid aliasing is to sample the surface above the Nyquist rate.

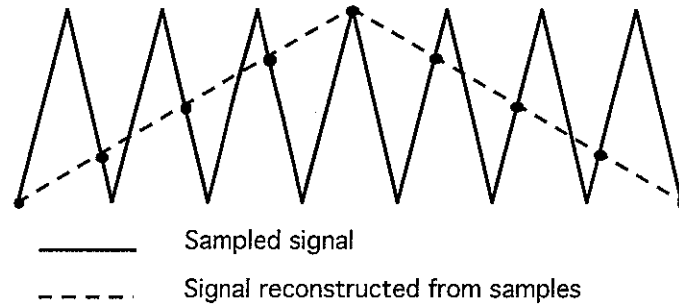


Figure 6.5 Aliasing. Higher-frequency signal aliases to low-frequency signal due to insufficiently-fine sampling.

The three types of features that are most likely to be under-sampled by our system are atomic lattices, areas with sharp slope discontinuities, and noise. Periodic surfaces (such as the one shown in Figure 6.5) alias to a lower frequency when they are regularly sampled, and the crystal lattices making up many of our samples are periodic surfaces. This means that any time we scan a sample with sample period larger than about 0.05 nm, we are going to see aliasing of the underlying crystal structure. Since we normally scan areas that are 200 nm on a side using only 80 samples (giving a sampling period of 2.5 nm), we can expect to see such aliasing in a large portion of our data sets. However, the magnitude of this aliased signal will be on the order of 0.1 nm or less (since that is how far the atoms protrude from the surface), which is only a 0.05% change in the data over a range of 200 nm, and only a 1% change for surfaces that vary by 10 nm. Such aliasing should not interfere with the study of large-scale features, but would present small ripples on flat surfaces.

Areas with surface or slope discontinuities (such as the surfaces shown in Figure 6.6) have very high frequency components in their Fourier transform. The sharper the edge, the higher the frequencies required to reproduce it. The sharpness of actual surfaces is limited to the radius of an atom, since the surfaces are constructed of atoms; this corresponds to a period of about 0.1 nm. Thus, sampling with a period of less than 0.05 nm would be

able to reproduce such features. However, our normal sampling period of 2.5 nm is insufficient to sample these features. The type of visible artifact this produces depends on the type of reconstruction done, and is described below in the reconstruction section.

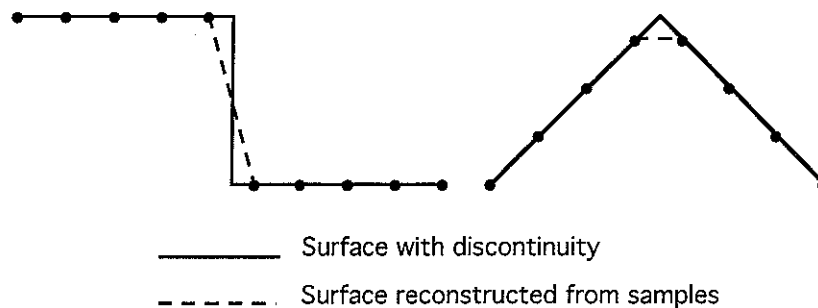


Figure 6.6 Surfaces with discontinuities in position and slope. Surfaces such as these must be sampled with infinitely high spatial frequencies (infinitely dense spacing).

Noise (either electronic noise in the sampling circuit or oscillations in the tip feedback circuit) can cause features in the data that are narrower than our sampling can capture and of sufficient height to mislead the scientist about surface features. This is one reason that scientists viewing STM data are often especially skeptical about surface features that lie parallel to the scan direction.

The Nanomanipulator system has three ways of mitigating the effects of noise. The first is that the surface is repeatedly scanned during experiments. This means that noise that is not correlated with surface position will appear and then vanish, while true surface features will remain constant with scanning (this is not true in all cases!) The second is the standard deviation coloring method described in chapter 4, where the system takes multiple samples at each location and colors spots red to the degree that the samples differ. Since surface height at a given location tends to remain constant, whereas noise varies with time, this can indicate areas where there is noise in the data. The third is that the system allows the surface to be scanned in either of two orthogonal directions; comparing scans taken in different directions can reveal noise that is caused by transient feedback responses to surface features.

Reconstruction

If the sampling frequency is above the Nyquist rate for the surface, it is possible to exactly reconstruct the original surface from the samples. This is done by using an ideal low-pass filter (multiplication by a pulse in the frequency domain, convolution with a sinc function in the time domain) on the samples. While possible, such reconstruction is very computationally expensive (requiring examination of all sampled data points for each point on the reconstructed surface) and not possible in real time on Pixel-Planes 5. Pixel-Planes does provide Phong-shaded triangles in real time.

Doing linearly interpolated reconstruction (triangle-tessellation) in place of ideal reconstruction introduces artifacts in the reconstructed surface that were not present in the sampled surface. This is true even when the sampling frequency is above the Nyquist rate for the surface. This is easily seen in Figure 6.7, which shows the difference between the original surface and the linearly-interpolated surface. This difference is because linear interpolation does not provide continuous derivatives (C1 continuity), although it does provide a continuous surface with C0 continuity.

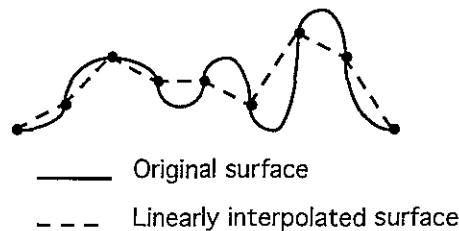


Figure 6.7 Linear reconstruction error. Original surface, samples, and linearly interpolated reconstructed surface.

The most striking artifact is that while the original surface is smooth and rounded, the linearly interpolated reconstruction is composed of flat triangles; the linear interpolation has introduced false edges. This effect is accentuated when the reconstructed surface is lit using a specular lighting model, since a flat triangle reflects light from the same direction over its entire area. While reflections off an ideally-reconstructed surface would show slow gradation of brightness with surface position, reflections off a triangle-tessellated surface show constant brightness across the triangles with sharp breaks (edges) at the triangle boundaries.

The triangles reflect similarly across their entire surface because the normal to the surface of a triangle is everywhere the same; the surface normal on a rounded surface changes smoothly across the surface. In order to reduce this difference in lighting, we apply Phong shading to the reconstructed surface (see Figure 6.8). Phong shading assumes that the triangular surface is a reconstruction of a surface that has smooth changes in derivative; it finds the surface normal at each vertex that matches the normal of the original surface and then linearly interpolates the normal across each triangle, rather than using a constant normal across each triangle. The reconstructed surface geometry is still a triangle, but the specular reflections are those of a surface with smooth changes in the normal. Note that in this method the normals match the sampled surface normals only at the sample points; they are interpolated between sample points. Phong shading does provide a surface with smoothly changing derivatives (C1 continuity), an improvement over the C0-continuity surfaces provided by linear interpolation alone.

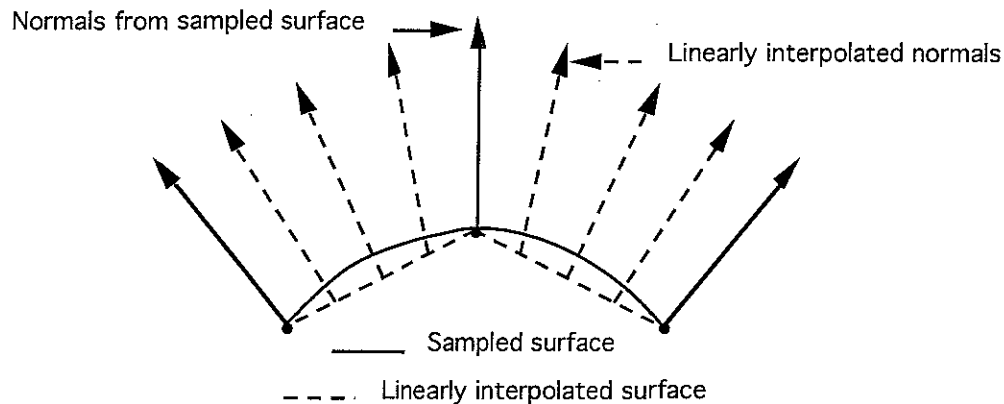


Figure 6.8 Phong shading. In Phong shading, the surface is drawn along the linear interpolation of surface data points, while the surface normal at each point is the linear interpolation from the normal at each data point.

Since we do not have a direct measurement of the surface normals at each data point, we must approximate. We find the approximated normal at a data point by taking the average of the perpendiculars to the line segments joining the data point with its four neighbors. The one-dimensional case is shown in Figure 6.9. Our case is equivalent to performing the one-

dimensional averaging in the X and Y directions independently and then averaging the result.

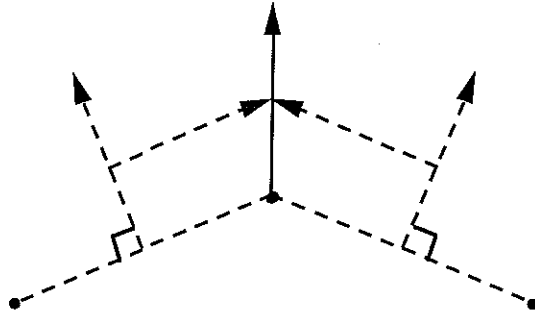


Figure 6.9 Surface normal calculation. The normal for a data point is found by averaging the perpendiculars to the line segments joining the data point to its neighbors.

The fact that Phong shading provides a C1-smooth surface means that the system produces incorrect shading at real slope discontinuities on the tessellated surface. Since the tendency of linear interpolation is to round off such discontinuities by assigning part of the change in slope to each of the surrounding samples (see Figure 6.6), the result is that the linear interpolation of geometry and the Phong shading combine to make the reconstructed surface appear smoother than is correct at places of real slope discontinuity on the sampled surface.

Software architectural overview

The Nanomanipulator system is conceptually a group of modules that communicate with each other via messages (see Figure 6.10). The user interface module handles coordinating the activities of the other modules in response to messages it receives from them. It sends information to the user via the image generation, tracker and sound modules and receives information from the user via the tracker and button modules. The STM module controls the STM, carrying out high-level actions (scan, pulse) and returning the results. The tracker module gives the position and orientation of the user's head and hand and provides forces to the hand. The image generation module displays all objects in the world. The button module provides events indicating when the user presses and releases buttons. The sound module provides short pre-digitized sounds to the user upon command.

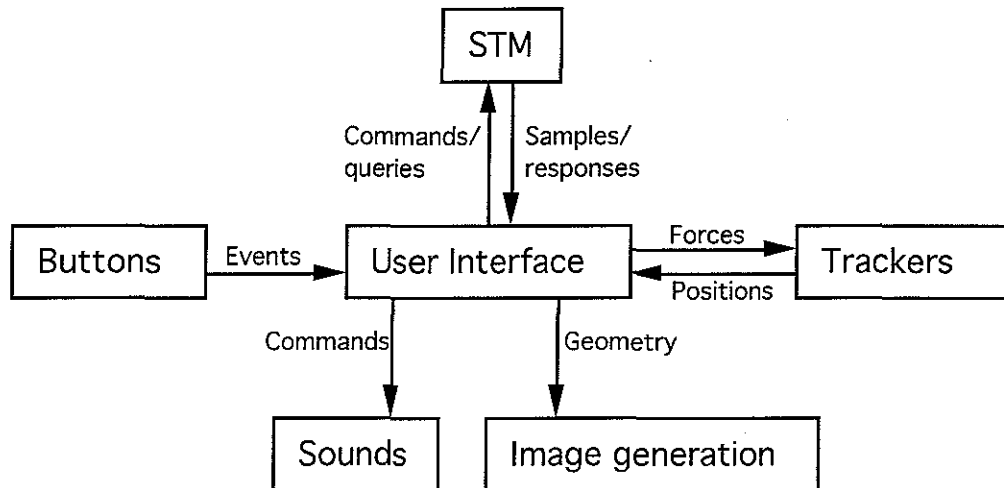


Figure 6.10 Software architecture. Basic software modules in the Nanomanipulator system. Boxes indicate software modules that deal with the various system components. Arcs indicate information flow (they do not always correspond to data paths across ethernet or other channels).

Image generation

The image generation system comprises several parts. At the highest level, the Virtual-Worlds library (Vlib) handles maintaining the tree of

Vlib concatenates the inverses of the transformations going up the tree from the eye coordinate system to the root node and then concatenates the world-to-object transform for each object to be displayed.

This tree of transformations allows the placement of objects and tools in reasonable coordinate systems. For example, the control panels in the user interface are placed in the **room** coordinate system, so that they are always near the user in both space and scale, even when the user flies about in the world. [Robinett92] describes this coordinate system, and manipulating objects in it, in detail. For this presentation, it is sufficient to know that the world-to-room transformation is adjusted to provide flying, scaling, and grabbing of the world. The individual world-to-object transformations are adjusted to move objects about relative to each other (such as the movement of the measuring grid). The tracker-to-hand and tracker-to-head transformations are updated based on position reports from the tracking system. Other transformations are fixed at run-time depending on the geometry of the trackers and HMD in use.

PPHIGS

PPHIGS is a display-list-oriented system where the entire scene is described in terms of *transformations* (matrices describing translation, perspective, rotation, and/or scale) and *primitives* (triangles, spheres). Larger objects can be constructed out of lists of primitives and transformations and given names. The objects so constructed can be referred to by name, and executed as sub-structures in larger objects. Pointers to the individual transformations and primitives can be maintained and used to modify an object after its construction by overwriting the old primitive or transformation with one of the same type. This is used to move the various objects about in the world.

The description of the objects in the world is done by routines in the **pobjects.c** file at startup time, as described in the section on the user interface module. Pointers to the parts of these objects that are going to be changed are obtained at their creation time and stored for later use.

This method of moving objects in the world is sufficient when there are a small number of changes to be made each video frame. Since the changes are all performed on the host computer (a sun 4/280) and then must be sent to

each GP in Pixel-Planes over a VME bus connection, this method is too slow for large numbers (several hundreds) of changes within a frame. Thus, for the routine virtual-world transformations, such as moving the user's head and hand, standard display-list editing is performed. For the more substantial task of adjusting the surface based on incoming STM samples, GP callbacks are used (since the STM scans a grid of 80x80 samples in about 30 seconds, approximately 10 surface points change position during each video frame).

GP callbacks

The Graphics Processors (GPs) on Pixel-Planes 5 are Intel i860 microprocessors with local memory. Normally, the list of primitives for a given object are spread among the processors so that each GP transforms a part of each object. It is also possible to write routines to run on the GPs during the display-list traversal, using GP callbacks.

Traditionally, a callback routine is a user-provided function that is called by operating system code when some event occurs. A *GP callback* is a user-provided routine that executes on the graphics processors when a certain object is found in the display list; parameters to be passed from the program running on the host to the callback routine are included in the object that causes the callback. The callback routine runs on each GP; they each receive the same parameters. Callback routines running on the GPs have access to many of the same graphics functions available on the host processor, but they have the benefit of running on all GPs in parallel. This brings to bear the compute power of about 30 i860's (13 MFlops each) as compared to one Sparc processor (2.6 MFlops), yielding a significant increase in processing power.

The GP callback routines in the Nanomanipulator project are used to describe and modify the surface drawn based on incoming STM samples points. Each GP takes a subset of the surface data and builds its own local display list describing that part of the surface. In addition to the local display list, each GP also constructs a list for each data point showing which triangles it belongs to (usually, there are six - see Figure 6.12) so that the data point can be recopied into the triangles each time it changes. This avoids re-generating the display list whenever a point changes.

As new data points arrive from the STM, the user interface copies them to the GPs as a parameter to another GP callback routine. This routine adjusts the triangles affected by each changed point; the four points surrounding a changed point have their normals adjusted. Each of these points is part of the six triangles that surround it. Since many of these triangles overlap, a single point change ends up affecting only the 20 surrounding triangles — see Figure 6.12.

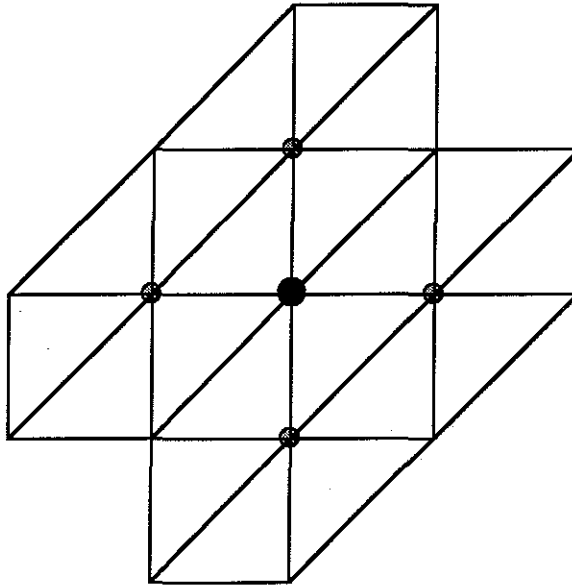


Figure 6.12 Triangles affected by changing a point. A change in position of the center data point affects the surface normal calculation at the four surrounding points. These points are part of a total of 20 triangles, each of which must be adjusted.

User interface

Startup

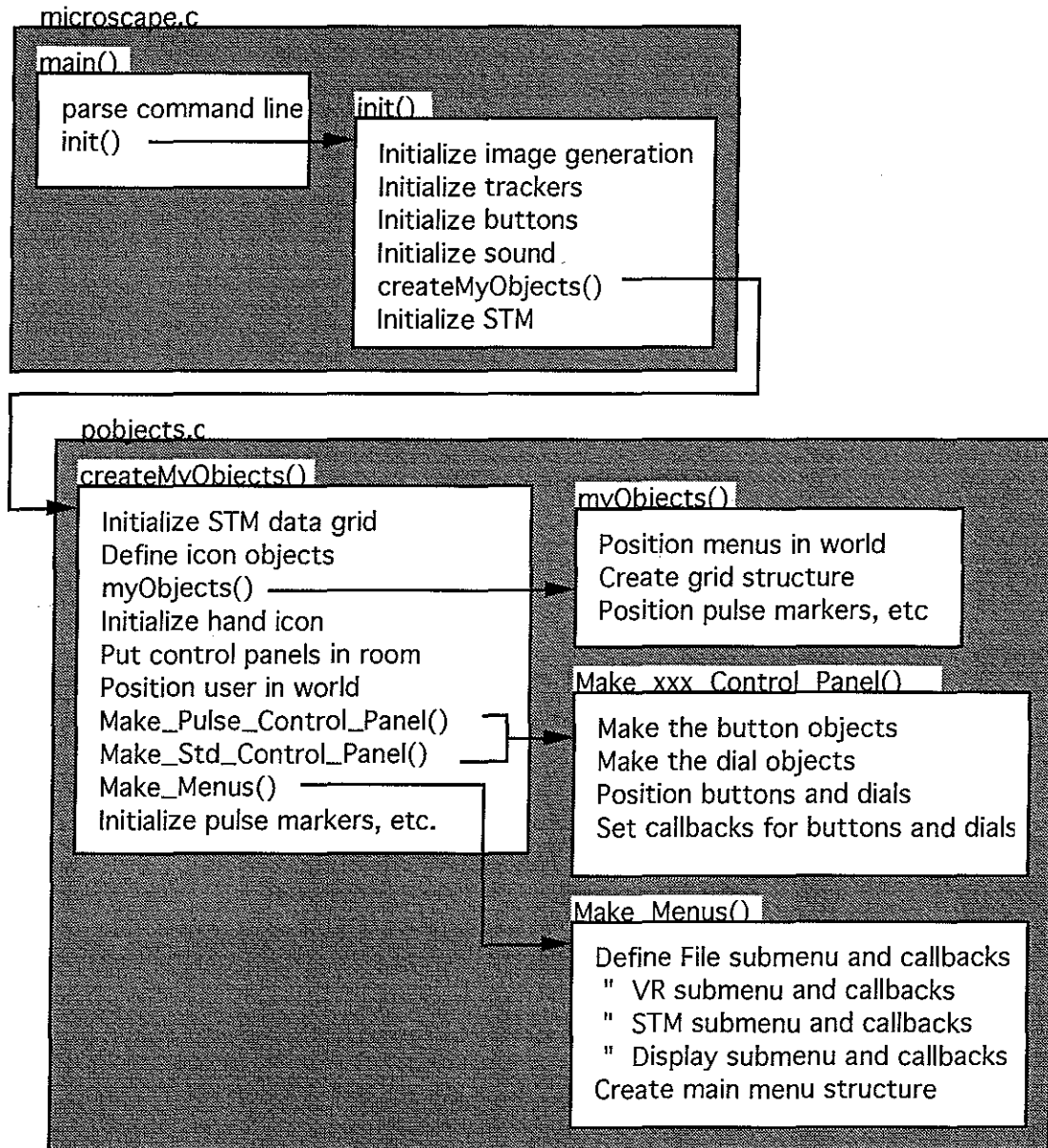


Figure 6.13 Actions taken during user interface program initialization. Gray blocks indicate source code files. White blocks indicate procedures. Arrows indicate procedure calls.

Figure 6.13 shows the actions taken as the user interface process starts executing. These include opening connections to the other modules (handled in the `init()` routine) and creating the objects in the virtual world (handled by routines in the `pobjects.c` file).

Note that most objects are created in the *world*, but that the control panels are located in the *room*. These terms refer to the display hierarchy used in the HMD systems at UNC. [Robinett92] The control panels are placed in room space so that the user can always access them. The menus adjust size and location whenever they are displayed in order to simulate being in room space (they should logically be defined in room space, but the library was built with them in world space).

The control panels and menus work by defining *control panel callback* or *menu callback* routines. Control panel callbacks are functions that are called when the user changes a dial setting or selects a button in a control panel. Menu callbacks are functions that are called when the user selects an item from a menu. The control panel and menu callback routines are also located in the `pobjects.c` file and will be further discussed in the sections on menus and control panels.

Main loop

Figure 6.14 shows the actions taken during each iteration of the user interface main loop. The basic strategy is to update the head and hand locations based on new tracker data, respond to user interactions, handle reports from the STM module, update the display, and handle keyboard commands. The user interaction (`interaction.c`) and STM report handling (`animate.c`) are described more fully in following sections.

One path through the interaction module is shown - the selection of a menu item. One action taken by the `interaction()` routine is calling the menu-handling routines in the menu library. If these routines detect that a menu item has been chosen by the user, the callback routine defined for the menu (during startup in `pobjects.c`) is invoked. The callback routines forward a character command to the user interface section. The keyboard command handler in the user interface section also forwards key presses to the same character command handler. This provides keyboard shortcuts for all

menu selections and allows a human assistant seated at the keyboard to type commands issued orally by the user.

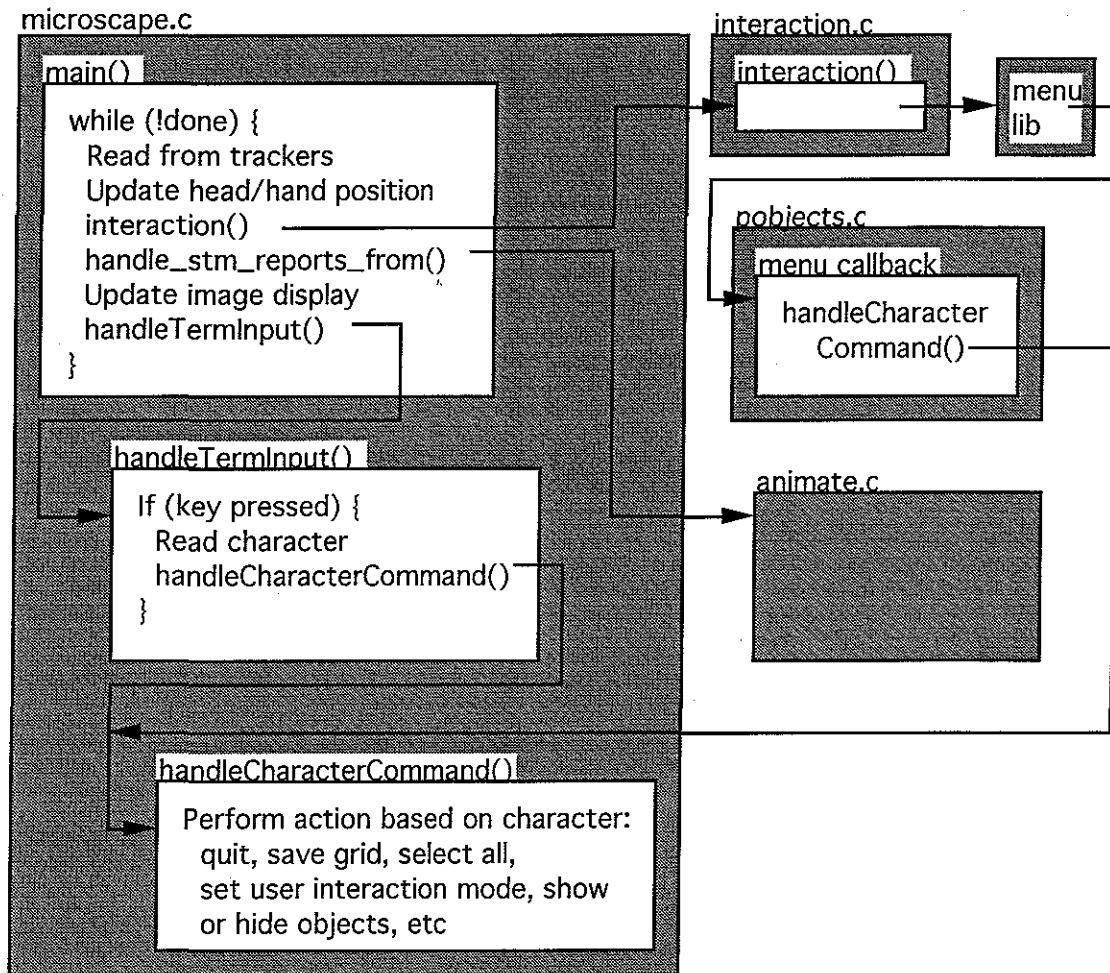


Figure 6.14 Actions taken by main loop of user interface program. Gray boxes indicate source code files. White boxes indicate routines within those files. The “menu lib” box indicates a library of routines that handle user interaction with the menus. “menu callback” is a generic name referring to the callback routine for any particular menu.

User Interaction

The user interacts with the world via button presses and hand motions read by the ARM. These correspond to menu picks in the world, control panel actions, and cursor actions in the world, as shown in Figure 6.15. The `interaction()` routine responds to the user’s button presses and hand

motions in the world. The control panel and menu libraries handle actions taken on these structures as described in following sections. The particular action taken in response to user input events depends on the interaction mode. Their actions are listed below by type:

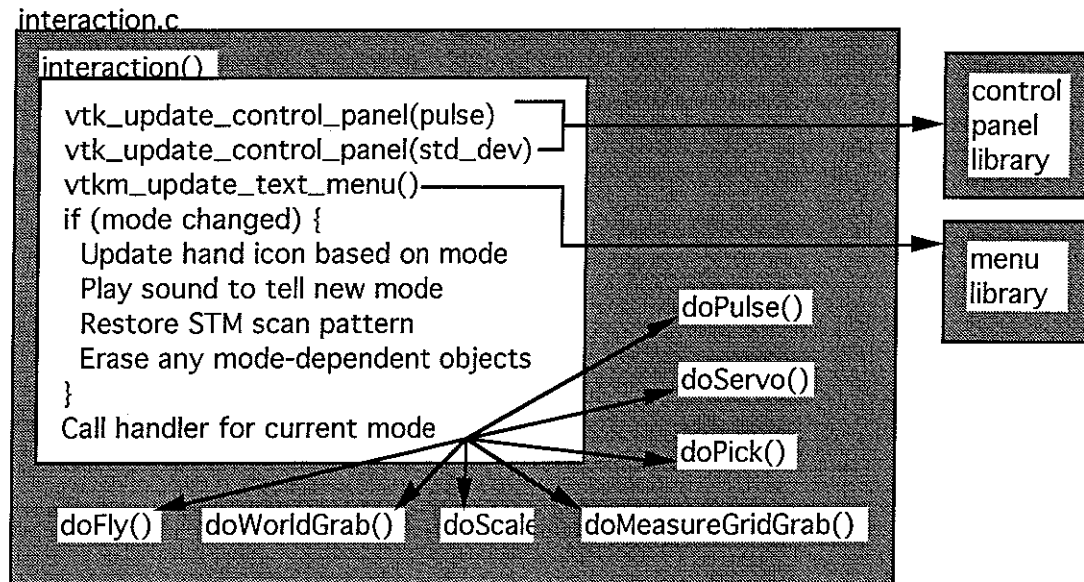


Figure 6.15 Actions taken to handle user interaction. The control panel and menu libraries invoke callback routines when the user takes action (possibly changing the mode of operation). Only one of the arrows in the 7-way branching is taken, depending on the current mode of operation.

The `doFly()`, `doScale()`, and `doWorldGrab()` routines handle the VR modes in the interface, allowing the user to fly, scale the world, and grab the world. Flying and scaling (both up and down) are implemented by merely calling an existing routine in the virtual-world library (Vlib) whenever and while the user holds the finger trigger down. Grabbing the world is handled by adjusting the position and/or orientation of the user's room coordinate system in the world coordinate system. The effect is the same as adjusting the world from the point of view of the user. All of these actions are described fully in [Robinet92].

The `doMeasureGridGrab()` routine allows the user to adjust the position of the measurement grid that is superimposed on the surface. The measurement grid is constrained to stay oriented the same way as the surface

and to cover the same area in X and Y. When the user holds the finger trigger down, this routine finds the position of the user's hand in the world (using a Vlib library call), picks out the Z component, and adjusts the Z component of the measurement grid object to match.

The **doPulse()** routine handles the pulse interaction mode. While the user is moving his hand around with the finger trigger unpressed, this routine causes a green *aiming line* to track the X and Y location of the hand and show the location on the surface where a pulse will be fired when the finger trigger is pressed. When the user presses the finger trigger, the routine sends a pulse request to the STM server asking for a pulse at the present location. It then sets a flag indicating that a pulse has been ordered.

The flag is reset after the pulse is actually fired by the routine that handles incoming messages from the STM. A yellow line is drawn piercing the surface at the location of the pulse. Once this is done, the **doPulse()** routine sends another message to the STM server asking it to resume its previous scan pattern. Note that the **interaction()** routine will handle sending this request if pulse mode is exited before the flag is cleared.

The **doPick()** routine handles *touch surface* mode, where the user can feel force feedback based on the surface height. Since the STM has a maximum scan rate, it may not be possible to move the tip from its current location to a requested location in one video frame time. In order to prevent the user's view from freezing during this time, the touch mode has two different states: attempting grab and lockstep.

When the user presses and holds the finger trigger in this mode, this routine sets a flag indicating a grab is being attempted and begins sending requests to the STM server every frame asking for a point sample at the current (X,Y) position of the user's hand. The responses from these requests are handled by the routine that handles incoming STM messages, which records the latest response (X,Y,Z) from the STM. Once the responses catch up to the requests, the system goes into lockstep mode.

Once the system has locked the tip onto the user's hand location, it goes into synchronous operation until the user releases the finger trigger. In this state, it sends a request for a point scan every frame and then calls the routine

to handle incoming packets repeatedly until the sample height is returned. The system then applies a force to the handgrip that depends linearly on the Z distance between the actual surface location and handgrip position. This force simulates a spring that pulls the handgrip towards the surface. Note that at no time does the handgrip position affect the Z height of the tip, which is maintained by electronic feedback.

When the user releases the finger trigger, a command is sent to the STM server to resume its normal scan. Note that the `interaction()` routine will send this request if touch mode is left before the finger trigger is released.

The `doServo()` routine handles the selection of a subset of the grid for scanning. When the user presses the finger trigger, this routine stores the (X,Y) location of the nearest sample gridpoint to the handgrip location. When the user releases the finger trigger, another (X,Y) grid location is recorded. These two locations form opposite corners of a rectangle. The `doServo()` routine sends a request to the STM server to limit its scan to within this rectangle.

Control panels

The control panel subsystem consists of three parts: setup, interaction, and callback. The setup and callbacks are defined in `pobjects.c` during program setup. The interaction is handled by the control panel library when the `interaction()` routine calls `vtk_update_control_panel()`. It results in a callback routine being called if the user modifies a setting in the panel.

The control panels are set up in the `Make_Pulse_Control_Panel()` and `Make_Std_Control_Panel()` routines during program initialization. These routines use control panel menu calls to set up the dials, digital displays, and buttons and bind the dials and buttons with appropriate callback functions. See figures 4.4 and 4.5 for the control panel layout. There is one callback routine for all of the dials and one for all of the buttons in each control panel. A string is passed to the callback routine to indicate which dial or button was adjusted. The dials that are set up have minimum and maximum values, and return floating-point values between the extremes. The exception is the “number of samples” dial in the standard deviation control panel, which returns integers from 1 to 100. The digital displays show the value set.

The interaction between the user and the control panels is handled by the control panel library during the call to `vtk_update_control_panel()` each frame in `interaction()`. When the user selects a new dial value, the dial callback is invoked with the new value. When the user presses a button, the button callback is invoked.

The dial callback routines modify the internal value for the given dial and also update the corresponding digital display. They do not modify the actual working conditions of the system. The button callbacks do modify the state of the system when they are pressed. The **send** button on each control panel causes a message to be sent to the STM server requesting a change in the values followed by a request to return the new values. The **cancel** button sends a request to the STM server to tell the current values. Neither button callback updates the displays or dial settings at this time.

Once the STM server responds to the request, the routine that handles incoming STM data sends updates to the dial callbacks on the menus by calling them directly. This causes the values in the dials to match that of the STM, since the STM has a limited precision in its settings.

Menus

The menu subsystem consists of three parts: setup, interaction, and callback. The setup and callbacks are defined in `pobjects.c` during program startup. The interaction is handled by the menu library when the `interaction()` routine calls `vtkm_update_text_menu()` and results in a callback routine being executed if the user selects from a menu.

The menus are set up in the `Make_Menus()` routine in `pobjects.c` during program initialization. The main menu is set up to have submenus for each of its entries (File, VR modes, STM modes, Display). Each of these submenus is set up to have a callback routine executed when one of its entries is selected. The callback routine is also passed a string that describes which menu item was chosen.

The menu display and selection operations are all handled by the menu library when the `vtkm_update_text_menu()` routine is called from `interaction()`. The library will show the proper submenu when a main-

menu topic is highlighted and will call the appropriate callback routine when a submenu item is selected.

The callback routines are specified at setup time and called by the menu library when an item is selected. The callback routine is passed a string to indicate which menu item was selected, so that one callback routine can handle more than one menu item. In the Nanomanipulator system, each callback takes action by sending one or more character commands to be handled by the `handleCharacterCommand()` function in `microscope.c` (each sends one, except for “Save and quit”, which sends one for save followed by one for quit).

STM report handling

Although commands are sent to the STM server from many locations within the user interface code, all responses come back in a single message stream. Also, when the system is reading data from an input stream rather than an actual STM, all messages arrive from the file. Each message is tagged with a type, which can be used to determine the proper action to take for each message.

All incoming messages are handled by the `stm_handle_packet()` routine, regardless of their source. A packet is a collection of one or more messages. Figure 6.16 shows the actions taken for each type of message. First, all messages are copied into the outgoing stream (if one is in use). After that, each message is handled according to type. The sections dealing with each individual module describe the actions taken in detail – this diagram is provided as an overview of message handling.

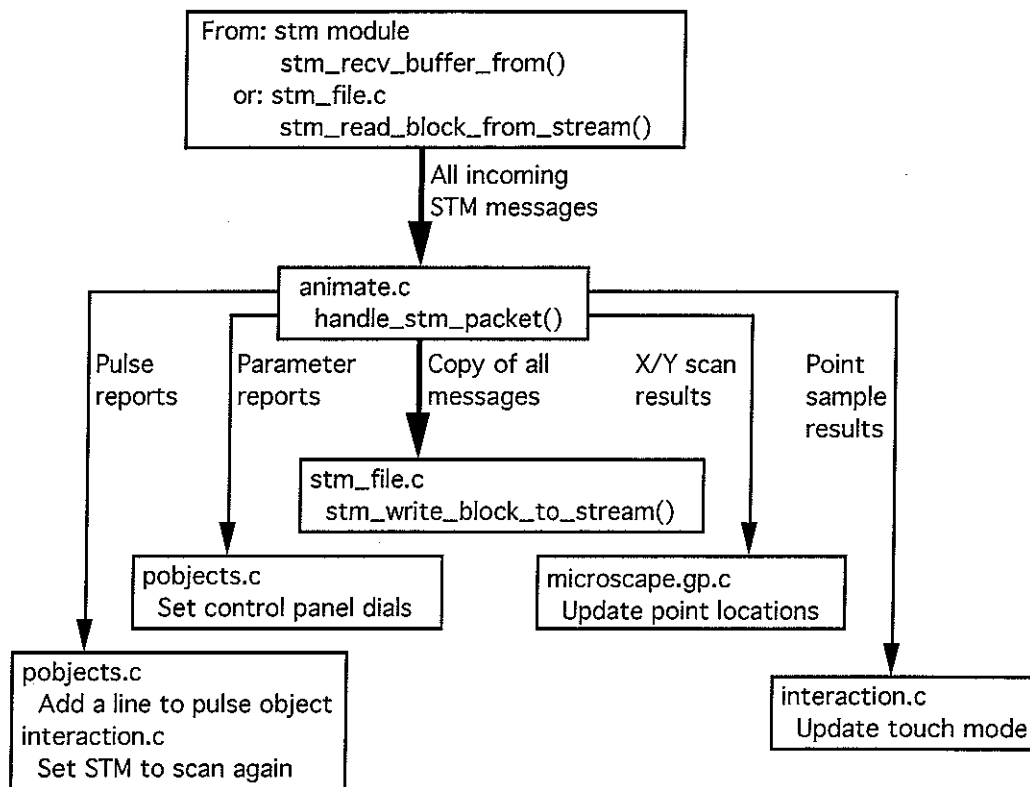


Figure 6.16 Distribution of incoming STM messages. All messages are copied to the output stream file if it is being used. Messages are also sent to be handled according to their type. Boxes hold the name of the file where the routines that handle the data are. Arcs indicate logical streams of messages.

Shutdown

Program shutdown mainly involves closing the connections to the various modules (STM, tracker, image generation, and button). Also, timing information is printed and the output STM data stream file is written to disk (the data stream accumulates in a main memory buffer to avoid disk I/O delays; up to 20 MB of data can be stored in this way).

Virtual-reality devices

This system makes use of three kinds of devices that are used in general virtual-reality (VR) systems: trackers, buttons, and sounds. As a result, existing modules (in the form of libraries) were already available for use at the time the Nanomanipulator was built. Since for each logical device there are a

number of supported physical devices, one job of the libraries is to hide the details of communicating with particular external devices from the user interface code and to present a uniform interface to all devices. An example of a logical device is the *hand-held controller*, which can be either the force-feedback ARM, a Polhemus-tracked button ball, or a Polhemus-tracked joystick.

At run-time, the libraries read information from the user's environment in order to determine which devices are to be used. The libraries establish any necessary connections to external devices and translate commands and responses between a standard format and the format of the particular device in use. One of the functions that is often provided is connection to a process on a remote machine via an ethernet connection. This is handled by the Standard Device Interface (SDI) library, which is further detailed in appendix A.

Trackers

The user's hand is tracked by the ARM. The head is tracked by either a mechanical tracker or one of several magnetic trackers made by Polhemus. Both the mechanical tracker and the Polhemus trackers communicate over a serial line to the user interface process. The ARM server process running on a PC communicates over a TCP (stream) ethernet connection.

There were no procedures for dealing with the ARM as part of the tracker library at the start of this project, so it was necessary to write the low-level interface routines and integrate them with the tracker library. This was made easier through the use of an existing library of ARM control routines.[Taylor90] In addition, the low-level drivers open the appropriate device to track the head so that the user code sees the tracking device as having both head and hand tracking on the same device.

Since the tracker library mediates between the user code and the devices, it is possible to run the program from stations other than the ARM. There are three other HMD stations set up around the lab, each with its own tracker for head and hand. The other stations cannot provide force-feedback to the user, so these requests are ignored by other stations.

The tracker library provides the position and orientation of the user's head and hand about 60 times per second. Since the display system is only updated between 15 and 30 times per second, some of these reports are ignored: the library always returns the latest report when a request is made.

The tracker library provides a routine for generating forces and torques at the user's hand position. There are also useful routines in the ARM library for converting forces and torques from one frame of reference to another. This makes it possible to convert force and torque from world space to user space (needed since world space may be scaled, translated, and rotated arbitrarily with respect to user space).

Buttons

The part of the module that reads the raw data values from the buttons runs on the PC that controls the ARM. The buttons are sampled via an A/D converter installed in the PC. The information on button state is sent to the user interface via a TCP (stream) ethernet connection.

The part of the button module dealing with converting raw button sample information into streams of button events runs on the same processor as the user interface. At each poll from the user interface, the button module checks the current and previous state of each button and converts this into one of four possible events: the button has just been pressed, the button is being held closed, the button has just been released, or the button is being kept open. These events are sent to the menu code, control panel code, or interaction code depending on the current system state and the location of the user's hand.

Sounds

The sound module runs on an Apple Macintosh computer. The module communicates via a serial connection with a DEC station 5000. A server on the DEC station shuttles the information to the user interface via a TCP (stream) ethernet connection. All sounds to be played are stored in local disk files on the Macintosh, which are read at program initialization time. Commands sent from the user interface cause particular sounds to be played. The sounds are single-channel pre-recorded sounds that we call "auditory icons." These

sounds are used to indicate the current operating mode, pulse completions, and the success or failure of attempted operations. The user interface has control over the volume of the played sounds.

STM server

The STM connection is handled using the *client-server* model. A server process runs constantly on the PC that controls the STM, waiting for connections from *client* applications, such as the user interface program. A client connects to the server by making a TCP connection to port 5559 on pc_stm0, which is the PC that controls the STM. When the user interface is started on the remote host, it opens a connection to this port. At this time, the server process on the PC opens all of the devices needed to control the STM and waits for commands from the user interface, which acts as the client.

The two devices used to control the STM are the analog-to-digital (A/D) board that it used to sense and set voltage levels and the pulse generator that is used to control the bias levels and generate voltage pulses. The A/D board is plugged into an expansion slot on the PC and is always available. The pulse generator is connected via an IEEE-488 parallel communication bus and can be turned on or off independently from the computer. When the pulse generator is not available, the server process will send error messages to the client whenever operations attempt to control the pulse generator, but other actions behave normally.

Once the devices have been opened, the server enters a loop in which it reads any incoming messages from the client, takes action based on its current mode of operation, and sends any responses generated by the current mode back to the client. Since the byte ordering of the computers on which the client and server run are not the same (i486 vs. Sparc), byte swapping must be performed by the server on incoming and outgoing data.

Outgoing messages are usually buffered and sent out 30 times per second; this is to prevent a bottleneck in transmission that occurs if the server attempts to send packets too close together (see the appendix describing the SDI library for more information on this). When the server is in **point scan** mode (described below), the information is sent as soon as it is available so that

real-time performance is possible. In this mode, the flow is controlled because the server must wait for the client to request each reading and vice-versa.

Setting and reading parameters

Many of the commands coming from the client process read and set parameters on the server. These are listed below, along with a description of their parameters and functions:

STM_SET_REGION_SIZE: This command is used to set the size and location of the scan region on the surface. The maximum range of the STM that is being controlled is considered to be $[0,1]$ in both X and Y. This command selects any rectangular subregion as the area to be scanned. Its floating-point parameters are the (X,Y) coordinates of two opposing corners of the rectangle to be scanned. This command sets only the boundaries of the region, not the number of samples taken. Any attempts to seek the tip beyond this region (either through scanning or user-directed motion) are considered an error.

STM_SET_GRID_SIZE: This command is used to set the number of samples to be taken within the scan region to make the grid of samples that are sent back to the client. Its integer parameters specify the number of samples in X and Y. Locations in the grid are indexed from 0 to the number of samples in each dimension minus 1. This command also sets the maximum step size of the tip in any scan mode to be the total scan length divided by the number of samples per line.

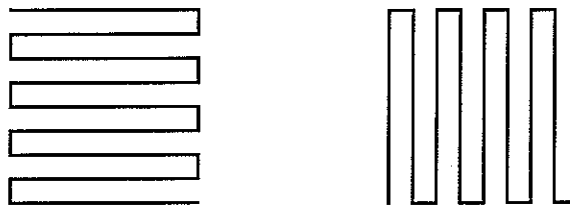


Figure 6.17 Patterns used when scanning the grid. The two patterns differ in the which direction is scanned faster. In each case, the scan proceeds one way and then reverses its path. Note that adjacent lines are scanned in opposite directions.

STM_SET_SCAN_STYLE: The integer parameter with this command specifies the scan style to use when scanning the grid. There are currently

two scan styles. The first style scans fastest along X and slowest along Y in a boustrophedonic pattern (see Figure 6.17). The second pattern is the same, but scans fastest in Y.

STM_SET_RATE: The floating-point parameter for this command specifies the rate of scanning in lines per second. This controls the maximum velocity that the server will attempt to move the tip across the surface in any mode of operation by setting the minimum time between steps to be the time to scan one line divided by the number of samples per line.

STM_SET_BIAS, STM_SET_PULSE_PEAK, STM_SET_PULSE_DURATION: These commands set the parameters of the voltage pulses that are used to modify the surface. Each has a floating-point parameter that specifies the value (in volts or seconds). The **bias** is the voltage between the tip and sample in normal scanning. The **pulse peak** is the voltage maintained during the pulse. The **pulse duration** is the time over which the peak value is maintained. (See Figure 6.18.)

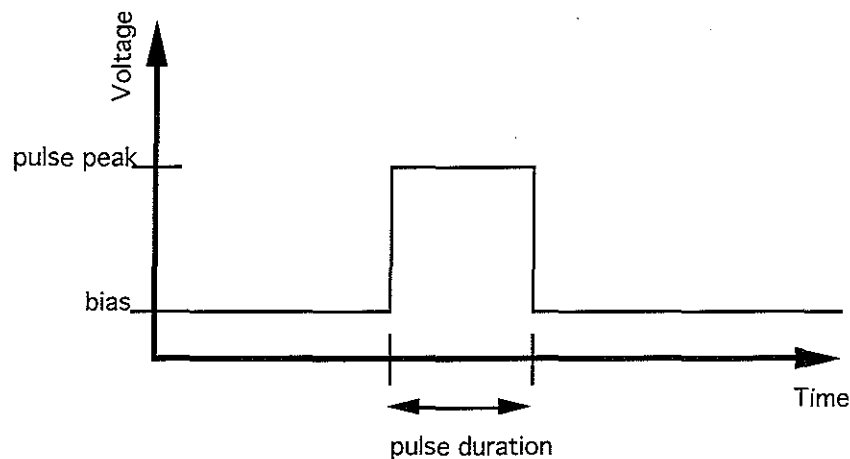


Figure 6.18 Meaning of the pulse parameters.

STM_QUERY_PULSE_PARAMS: This command has no parameters. It is used at startup to determine the initial state of the pulse generator. This command causes the server to query the pulse generator for its current settings and to send an **STM_PULSE_PARAMETERS** response to the client. This response has three floating-point parameters that tell the bias, pulse peak, and pulse duration, and an integer parameter that is 1 if the pulse generator is enabled and 0 if it is not.

STM_SET_STD_DEV_PARAMS: This command has an integer parameter that sets the number of samples to be taken at each grid location per pass and a floating-point parameter that specifies the rate in Hz at which these samples are to be taken. These parameters are used in the **window scan** mode to control the sample set for which the mean and standard deviation are computed. See the description of **window scan** mode below for more information on the meanings of these parameters.

STM_QUERY_STD_DEV_PARAMS: This command has no arguments. It causes the server to send a STM_STD_DEV_PARAMETERS response to the client. This response has an integer parameter telling the number of samples taken at each grid location per pass and a floating-point parameter that specifies the rate in Hz at which these samples are taken.

Controlling the STM

Other commands cause actions to be taken by the STM. The server enters different modes in order to perform these actions with the STM. These include scanning an area (in one of two patterns), sampling a point, generating pulses at given locations, and shutting down the STM. The messages dealing with these are listed below, along with a description of their parameters and functions:

STM_SCAN_WINDOW, STM_RESUME_WINDOW_SCAN: These commands cause the server to enter or re-enter **window scan** mode. The STM_SCAN_WINDOW command selects a subset of the entire grid that is to be scanned. Its four integer parameters specify the lower (X,Y) coordinates and the upper (X,Y) coordinates of the window within the grid to scan. When the scan is constrained to a small portion of the total grid, the data in that area is refreshed more frequently, although the tip moves at the same rate. This is used when the user wants to concentrate on a small area of the surface that might be changing (such as where pulses are being fired). The STM_RESUME_WINDOW_SCAN command causes the scan to resume in a previously-specified window after being interrupted by another command (such as touch or pulse).

Window scan is the mode that is used to acquire the grid of data points that are used to draw the surface in the user interface. While in this mode, the

STM scans the tip across the surface and acquires data at each grid location within the current scan window. The tip is stepped from one grid point to the next in the current scan pattern. At each grid point, the number of samples specified by the SET_STD_DEV_PARAMS command are taken at the rate specified in that command. At this point, the mean and standard deviation of the samples are found and a response is sent to the client.

The type of response depends on the number of samples taken. If there is only one sample per grid point, then an STM_WINDOW_SCAN_RESULT response is returned, in which case only the mean (the value itself) and not the standard deviation is sent to the client. If there is more than one sample at the point, then an STM_STD_DEV_SCAN_RESULT response is sent, which includes both the mean and standard deviation. Both commands send the integer (X,Y) coordinates of the grid location that was scanned and a pair of long integers that specify the time that the scan was taken in microseconds past midnight on Jan. 1, 1970 (same format returned by gettimeofday() in Unix). The time is read using a custom clock routine that is accurate to within 1 ms. The time could be used to compensate for a known thermal drift in the sample with time. In both cases, the window scan proceeds asynchronously from the operation of the client, sending scan results as they are found. This continues until another command is received from the client.

STM_SCAN_POINT: This command is used to sample a single point somewhere within the region. The two floating-point parameters specify the location of the point in the range [0,1]. The point need not lie on a grid point. This is the mode that corresponds to **touch mode** in the user interface. When this command is received, the tip is moved at the maximum step size and rate towards the point of interest. On arriving, the height is sampled once and a STM_POINT_SCAN_RESULT result is sent. The floating-point (X,Y) location and value are returned, along with the time at which the point was sampled. Once the point has been scanned, the STM goes to **idle mode** until another command is received from the client.

STM_PULSE_POINT: The two floating-point parameters for this mode specify the (X,Y) position at which a bias pulse is to be fired. The location is in the range [0,1] and need not lie on a grid point. The tip is moved using maximum-sized steps at the maximum rate (as set by STM_SET_SCAN_RATE and

STM_SET_GRID_SIZE) to the desired location, at which point the pulse generator is sent a command to cause it to generate a pulse. If the command succeeds, an STM_PULSE_COMPLETED response is sent to the client. If it fails, an STM_PULSE_FAILURE command is sent. Both commands return the (X,Y) location of the attempted pulse. Once the pulse is completed or fails to complete (if the pulse generator is turned off), the server goes into **idle mode** until another command is received from the client.

STM_IDLE: This command, which has no parameters, causes the server to enter **idle mode**. When in this mode, the server leaves the tip stationary and does not perform any operations on the STM. This mode is usually reached after a pulse or point scan and not as a result of a command from the client.

STM_SHUTDOWN: When this command is received the server enters **shutdown mode**. In this mode, the server shuts down the STM and closes the connection. The tip of the STM is moved to the center location of the region and then all devices are closed. The server then goes back to waiting for another connection. This mode is also entered when an error is encountered or the TCP connection is closed.

Compiling and linking the software

There are three parts to the software. The first part is the STM server that runs on the PC. The second is the GP callback code that runs on the Graphics Processors (GPs) on Pixel-Planes 5. The third is the user interface code, which is called Microscope and runs on the Sparc host for Pixel-Planes 5.

The source code for the PC server lives on **pc_stm0** on the **C:** drive in the **\src\stm** directory. There is a Borland C++ project file in that directory called **stm.prj** that contains the information needed to compile and link the server. To compile it, start the compiler by typing *bc stm.prj* at the command line and then select *compile/build all* from the menus.

The source code for the user interface and the GP callback routines lives on the Unix system in the **/afs/cs.unc.edu/project/stm/src/server_microscope** directory. The GP callback code is linked into the user interface code as part of the Microscope application. In order to make the object file to link with, type

hmdgo make -f microscope.gp.c in that directory (this is done automatically if you make the Microscope program with the make file).

In order to make the user interface program (called Microscope), type *make Microscope* while you are in that directory. This will build all of the object files and link them into an executable, which will be placed in the **sparc_sunos** subdirectory. Once the program has been compiled and tested, it should be copied into the executable directory for the STM project that holds Sun 4 executables, which is `/afs/cs.unc.edu/project/stm/bin/sparc_sunos`.

Many libraries are used by the Microscope program to control underlying devices (such as display hardware, trackers, and buttons). Figure 6.19 shows the relationships of these libraries.

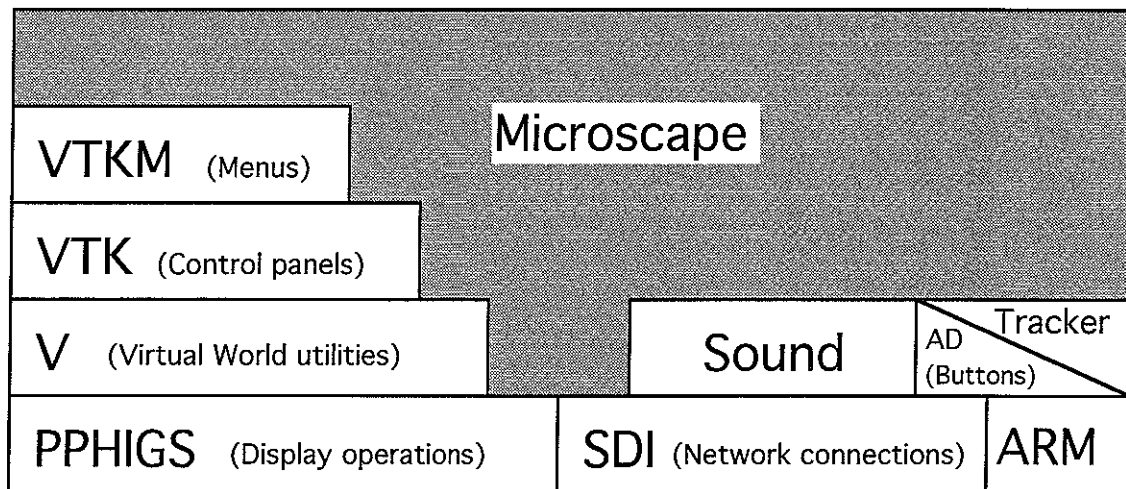


Figure 6.19 Libraries used by the user interface code. Boxes indicate libraries. A box sitting on top of another indicates that its library makes use of the lower library. The gray area indicates the Microscope program, which makes direct use of all but the ARM library.

Running the software

There are three parts to running the software: starting the PC server, setting environment variables and executing the user interface client. The PC server runs on the machine named **pc_stm0**. The server program lives on the **C:** drive in the `\src\stm` subdirectory. The program **server.bat** should be run from within that subdirectory. This program must be run before the

client application is started, since it must listen on a socket for incoming connection requests. The server is normally left running all the time, since this PC is dedicated to running the STM.

The Unix environment variables are needed to select among the various input devices and displays that can be used. There are separate variables for the display system, button inputs, trackers, and HMD in use at the time. Fortunately, all of the correct settings for various stations have been set up in files that are maintained by the HMD project, so that in order to set up the environment, all that is required is sourcing the correct station file. There are several station files, including **glab**, **hmdlab**, and **ARM**. In order to set up the environment to run at the ARM station, one would execute the following command:

```
source /afs/cs.unc.edu/project/hmd/bin/ARM
```

The **glab** and **hmdlab** setup files are in the same directory with the names **glab** and **hmdlab**.

One additional part of the environment that needs to be checked is the particular pixel-planes 5 host to run on. Each host can only run on some subset of the HMDs, so it is necessary to select the proper host. The only way to know this in general is to ask someone on the HMD team. Another possible solution is to check the `/etc/motd` files on the various hosts to see which one has its displays going to the proper station. At this time, the hosts to check are **jason**, **hugin**, and **audrey**.

The executable program for starting the Nanomanipulator is `/afs/cs.unc.edu/project/stm/bin/sparc_sunos/Microscape`. If it is run with the argument **-help**, it will display a list of command line arguments and default system parameters. A list of the arguments follows, with a description of each:

```
Usage: Microscape [-sphere] [-poly] [-dot] [-d device] [-f infile]
      [-z scale] [-r rate] [-call/nocall] [-grid x y] [-pots x y z]
      [-silent] [-perf] [-i streamfile rate][-o streamfile]
      [-std num rate scale] [-region low high] [-measure dist]
      [-nomenus] [-nocpanels]
```

- poly: Display polygonal surface (default)
- sphere: Display as spheres
- dot: Display as dots

These parameters describe the type of geometry to use for surface representation. The default surface representation is polygons, which corresponds to tessellating the sample points completely with triangles. When using polygon mode, callback mode (see below) should always be selected. If not, the display update rate falls to below 1 frame per second. If sphere or dot mode is selected, callback mode should be disabled.

- d: Use given **device** (default sdi_stm0)

This parameter tells which STM device to read from. The STM that was built at UCLA is sdi_stm0.

- f: Read from given **infile**, not device

It is possible to view a grid of data that was stored, rather than data coming live from an STM. If this parameter is specified, then one frame of data will be read from the file whose name is specified. Make sure that the grid size is the same as that in the file to be loaded using the -grid parameter (described below).

- z: **Scale** of z relative to x/y (default 1)

This parameter allows the scaling of Z to be changed with respect to X and Y. With the UCLA STM, this parameter is required in order to match the Z height depending on the settings of the gain knobs for the X,Y, and Z axes. When the instrument is set to scan a 2000x2000 Ångstrom region with maximum Z throw, this parameter should be set to 0.9245. See Russell Taylor's lab manual entry on 4/3/92 for a description of how to compute this value.

- r: **Rate** of scanning (default 1 line/sec)

This parameter tells the requested speed of STM scanning in lines per second. Each instrument has a maximum scan rate that it can achieve, so it may not be possible to go as quickly as requested. For example, the UCLA STM with its current controller is limited to scanning at about 4 lines per second.

-call: Use callbacks (default)

-nocall: Do not use callbacks

These arguments specify whether Graphics-Processor (GP) callbacks are to be used. These callbacks should be used for polygon mode and should not be used for sphere or dot modes. See the section on GP callbacks for more information.

-grid: Take *x* by *y* samples for the grid

This parameter sets the number of scans taken in the X and Y directions. These two will normally be the same, in order to produce a square grid. This size should be as large as possible while still maintaining an acceptable update rate. For a 2-rack pixel-planes system, a grid size of 80x80 is good.

Note that this parameter controls the resolution of scanning, but not the size of the area scanned. See the -region flag for setting the size of the region.

-pots: Pot settings (*x,y* gain, *z* axis sensitivity)

This parameter was added in an attempt to make the scaling between X, Y, and Z easier on the UCLA STM. In this case, the values set on the three gain pots would be entered and the scaling computed. This parameter currently does not work and should be avoided.

-silent: No sound made in world

If this parameter is used, the Nanomanipulator will not establish a connection to the sound server or play sounds in the virtual world. This parameter was added because at the time there was one sound server for several stations and this flag made it possible to avoid multiple programs trying to use it at once.

-perf: Turn on the perfmeter

The PPHIGS library has a flag that allows a performance meter to be turned on. This meter displays the current polygons per second and frame rate of the system in one corner of the screen. When this flag is given, the performance meter is turned on.

- i: Read data from **streamfile** at rate blocks/frame
- o: Write the STM data stream to **streamfile**

It is possible to save all the incoming data from the STM device into a stream file. Later, the file can be used to replay the entire experiment, including pulses and all scans. These parameters allow saving or loading a stream file. Only one should be specified at a time. The -i argument requires the rate at which to read from the file, which allows adjustment of the speed at which the data is replayed. This speed is in packets per frame, which corresponds to the actual data messages sent by the STM server. This is loosely related to the number of updates per frame, but not identical.

- std: Take **num** samples at rate per point. Multiply by **scale** to get [0,1]

This parameter enables coloring by standard deviation (see the section on surface coloring for more information). It allows the selection of the number of samples per data point, the rate at which these samples are taken, and the scaling given to the resulting standard deviation before using it to color the surface.

- region: Scan from **low** to **high** in x and y (Default 0 to 1)

This argument allows specifying a subset of the maximum scan area of the STM to concentrate on. The current maximum range of the instrument is normalized to (0,0) through (1,1) in (X,Y). The low and high values select a scan region from (low,low) to (high,high). This allows a zoom of the scan to a smaller subset of the area. Note that this only allows selection of square areas along the diagonal to scan.

- measure: Put a measure line every **dist** nm (default 10)

This argument allows the selection of the inter-line distance on the measurement grid (see the section on the system from the user's point of view for more information on the measurement grid). This selects the distance between lines in both X and Y.

- nomenus: Do not create or use menus

- nocpanels: Do not create or use control panels

VII. FUTURE WORK

Additional capabilities

While the Nanomanipulator has sufficient capabilities to allow experiments to be run, it still lacks features that should be present.

Distance measurement

The most important omission is the display of distance markers on the surface shown by the system. Any estimate of size or distance on the surface must now be made by judging their size relative to the whole surface shown, and the scientist must know the scan size that was specified when the system was run. Experience has shown that scientists want to get numbers out of their visualizations, not only qualitative information; we plan to add a pair of movable markers that slide over the surface. The markers will report the lateral (in the X-Y plane) and height (Z) difference between themselves.

Side-by-side comparison

We plan to add the ability to capture snapshots of the surface at two points in time and show them side-by-side for comparison. This will facilitate studying pulse effects by allowing before and after comparison.

Line cuts through the surface

One feature that we have been repeatedly asked for is the ability to see the graph of surface height along a line on the surface. This will allow the scientist to see a cutaway view through some feature of interest.

Hardware improvements

Work should also proceed on increasing the capabilities of the system hardware.

Additional microscopes

Work is underway to allow the system to work with an atomic force microscope (AFM), and with an STM that has a wider scan range. These instruments will enable additional types of experiments to be performed with the system.

Increased scan rate

The scan rate of the system is limited to about 4 lines per second by the update rate of the D/A converters on the PC controller. Modifying the algorithm used to drive the D/A converters to sweep a whole line at a time rather than a single point at a time, or replacing the converters with faster ones, will allow the surface to be scanned more rapidly.

Study of incremental improvement

While we have shown that a virtual-reality interface to an STM is sufficient to produce an instrument that is capable of new insights, we have not shown that all of the components of VR are necessary to achieve this goal. For example, we have no results showing that a fully-immersive system provides additional insights over a head-tracked stereo, through-the-window interface. It would be useful to know which features of VR are necessary to achieve what kinds of insights.

Availability

Finally, in order for the system to be broadly available it must be ported to graphics and user interface hardware that is more portable than Pixel-Planes 5 with the Argonne-III force-feedback ARM. Most, if not all, of the benefits of the system should remain in a through-the-screen implementation that includes head-tracked stereo display and real-time control over the lighting parameters.

OpenGL

Porting the system to use OpenGL as its display language would allow it to be run on Silicon Graphics Reality Engines, which are the most common commercially-available high-end graphics systems.

Commercially-available Pixel-Planes systems

In the near future, it will be possible to purchase Pixel-Planes 5 systems from at least two vendors. Porting the system software to run on hardware from Division and IVEX would allow purchasers of these systems to have the full immersive power of the Nanomanipulator to work with.

Commercially-available force-feedback arms

There are force-feedback devices available commercially. Two less-expensive ones are the Phantom (built at MIT by Tom Massey), and the PER-Force Handcontroller (built by Cybernet in Ann-Arbor, Michigan). Porting the system to be able to use one of these devices would allow the force-feedback to be used at other locations.

ANNOTATED BIBLIOGRAPHY

[AIP72] Billing, B.H. et. al., *AIP Handbook*, McGraw Hill, NY, 1972.

[Aono93] Aono, M., A. Kobayashi, F. Grey, H. Uchida, D.H. Huang, "Tip-Sample Interactions in the Scanning Tunneling Microscope for Atomic-Scale Structure Fabrication," *Jpn. J. Appl. Phys.* Vol 32 (1993), Part 1, No 3B. pp. 1470-1477.

Ibaraki, Japan. Pick and place single Si atoms from Si(111)7x7 and Si(001)2x1 using W tip in UHV. Separation of about 1 nm when scanning and pulsing. Can routinely remove single atoms from specific locations. For Si(001), usually two atoms forming a dimer are removed. Sample cleaned by flash heatings. Tip **etched, electron-bombarded**, and then **pulsed** to clean.

Can remove single atoms with pulses or cut grooves by scanning with raised bias. Whole QID pattern made in a 50 nm area (no metal, just trenches in Si). Looks like Si ions are being field evaporated from the surface. Polarity does not dominate, but has some effect due to the difference in critical field for + and - Si ions. Field strength, not current, causes change (they varied tunnel current without significant effects). They give the equations of field evaporation.

The center adatoms in Si(111) were more easily removed. Fraction of atom re-deposition during extraction varies from negligible to about 19% based on pulse height. It is possible to move a deposited atom around using other nearby pulses. The deposited/moved atoms always land at the center of a triangle formed by a corner adatom and two adjacent center adatoms.

[Avouris92] Avouris, P. and L.-W. Lyo, *Applied Surface Science*, 60/61, 1992. pp. 426.

[Bartle82] Bartle, Robert G. and Donald R. Sherbert, *Introduction to Real Analysis*, John Wilen & Sons, Inc., 1982.

[Becker87] Becker, R.S., J.A. Golovchenko and B.S. Swartzentruber, "Atomic-Scale Surface Modifications Using a Tunneling Microscope", *Nature*, 325, 1987. pp 419-421.

Could cause an atomic-scale bump on a germanium surface using a bias jump. Tungsten tip, germanium (111) sample. Tip bias at -1V. Tunnel current 20pA. Jump to -4V and let feedback settle to write. Feature size is ~8Å across. Could not do this on silicon. Success rate varied from >90% to <10%, seemed better after tip was "recharged" by touching it to the surface.

[Besenbacher91] Besenbacher, F., F. Jensen, E. Lægsgaard, K. Mortensen, and I. Stensgaard, "Visualization of the dynamics in surface reconstructions," *Journal of Vacuum Science and Technology*. B 9 (2), Mar/Apr 1991, pp. 874-877.

Makes clear the importance of watching surface evolution over time with an STM. Talks about making STM movies offline that show how things evolve on the surface with time.

[Binnig82] Binnig, G. and H. Rohrer, "Scanning Tunnelling Microscopy", *Helvetica Physica Acta*. Vol 55 (1982), pp 726-735.

The original STM paper. Basic introduction to using tunneling for surface study. Talks about designing their instrument. Talks about resonant tunneling. Comparison with other microscopes as for scale and resolution.

[Binnig87] Binnig, G. and H. Rohrer, "Scanning Tunneling Microscopy — from birth to adolescence," *Reviews of Modern Physics*, 59(3) July 1987, pp. 615-625.

[Brooks88] Brooks, F. P., Jr., "The computer scientist as a toolsmith: Studies in interactive computer graphics." *Information Processing* 77, 625-634. B. Gilchrist, ed. North-Holland: Amsterdam, 1977.

Brooks Jr F.P.(1988) "Grasping Reality Through Illusion — Interactive Graphics Serving Science", *Proceedings CHI* 88, May 15-19 1988, pp 1-11.

Read it once a year or more. Call for visualization, in-flight watching of simulation, and steering of computation. Call for rules of thumb and observations to be reported, as well as verified results. LOTS of observations given.

Brooks, F. P., Jr., M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick. "Project GROPE — Haptic displays for scientific visualization." *Computer Graphics: Proceedings of SIGGRAPH '90*, Volume 24, Number 4, August 1990. pp. 177-185.

Covers several experimental stages leading up to drug docking. Force addition gives up to a factor of two improvement over graphics only. Serious users report radically improved situation awareness. "Our users explore in virtual worlds holding a handgrip, an experience somewhat like exploring a dark engine compartment with a screwdriver." GROPE-I (Batter's 2-D experiments). Motivated students improved with force feedback studying force fields. GROPE-II (Kilpatrick's 3-D blocks experiment). Shadows helped most, then force, then stereo and variable viewpoint. Users decomposed 3-D tasks into 2-D and 1-D parts. Question of effects of nonlinearity of kinesthetic sense and what effect this has when superimposed with linear visual sense — visual seems to dominate. GROPE IIIA (Ming's 6D rod and spring experiment). Force was 2x faster and 2x lower energy, although the speed may result from decomposition of the visual task into 2 subtasks (force and torque). GROPE IIIB (Ming's drug docker). Actual 6-D motion was about 1.75 times faster and 41% more efficient with force. The overall task completion time was not

much reduced. Perhaps the best gain cannot be measured — reported superior understanding of the solution space. Hard surfaces are hard to do. Click sounds can fool users into thinking a surface is hard. About 80 ups should be as much as the user can sense in the ARM system. Smaller, 4" on a side finger-hand manipulator might be best kind — less fatigue, uses fine finger motions, better match to display space. "A haptic display makes magnitudes vivid."

Chen, C.J., "In situ testing and calibration of tube piezoelectric scanners," *Ultramicroscopy*, 42-44, pp. 1653-1658, 1992.

"Crosstalk" between the 4 electrodes on a tube scanner can be used to calibrate it. Driving one or opposing electrodes with an AC signal results in a charge on the other. Current flow (order microamps) to ground reveals the piezoelectric constants ($\text{\AA}/\text{V}$) for the tube. Asymmetries in the tube can also be found.

[Cohen-Tannoudji77] Cohen-Tannoudji, Claude, Bernard Diu and Franck Laloe, *Quantum Mechanics, Volume one*, Wiley-Interscience, 1977.

[Crommie93] Crommie, M.F., C.P. Lutz and D.M. Eigler, "Confinement of Electrons to Quantum Corrals on a Metal Surface," *Science*, Vol. 262, 8 October 1993. pp. 218-220.

IBM Almaden. Good color image of the work on the cover of this issue. Shows the interference patterns of electrons trapped within a ring of iron atoms on a copper surface. Used tunneling spectroscopy to resolve the local density of states, revealing the pattern. Ultra-high vacuum and ultra-low temperature (4° Kelvin).

[D'Azzo88] D'Azzo, John J. and Constantine H. Houppis, *Linear Control System Analysis & Design (3rd edition)*, MacGraw-Hill Electrical Engineering Series, 1988.

[Edstrom91] Edstrom, Ronald D. and Maria A. Miller "Scanning tunnelling microscopy and atomic force microscopy visualization of the components of the skeletal muscle glycogenolytic complex," *Journal of Vacuum Science and Technology*, B 9 (2), Mar/Apr 1991, pp. 1248-1252.

Shows that still-picture shaded views of the data are used for presentation of results. Imaged the complex lying on the surface of HOPG without coating them, but rather drying them from dilute solution onto the surface.

Ehrichs, E.E., R.M. Silver and A.L. deLozanne, "Direct Writing with the Scanning Tunneling Microscope", *Journal of Vacuum Science and Technology*, A6 (2), 1988. pp 540-543.

STM tunneling current dissociates molecules from an organometallic gas, leaving a metal deposition on the sample. They think the high E-field or field-emitted electrons break down the gas and cause a microscopic plasma, which deposits metal atoms on the surface. Gives good references to previous surface modification work. UHV

(10^{-8} Torr). Tungsten tip, tip negative, sample grounded while scanning. Tip + while writing. Wrote on copper and silicon. Gas was DMCd. They think a halogen could be used for etching. Their smallest spot was 30 nm across, made on silicon(111). When scanning, the spots are noisy and the surrounding (untouched) surface is clearer.

Ehrichs, E.E., S. Yoon and A.L. de Lozanne, "Direct Writing of 10-nm Features with the Scanning Tunneling Microscope", *Applied Physics Letters*, 53 (23) 2287-2289, 1988.

Very elaborate sample prep. Vacuum. Flood chamber with organometallic gas that the pulses cause to become adsorbed on the surface. They could also form lines with no gas in the chamber. Sample biased at -1.7V, 2 nA current; ground Tungsten tips, Si111 sample. Pulses from 10-400ns each at 130 Hz to 13 kHz. Wrote grid of 10-nm wide lines with 4.5V pulses of 400ns each at 13kHz while scanning at 23 nm/sec. Lines were 4 nm high. There is a bunch of carbon in the lines that end up on the surface. They do not know why. The lines they lay down can be used like photoresist to stop etching. Critical voltage at 3V pulses, rapid increase in blob size as voltage is increased above 5V.

[Eklund91] Eklund, Elliot A., R. Bruinsma, J. Rudnick and R. S. Williams, "Submicron-scale surface roughening induced by ion bombardment," *Physics Review Letters*, vol 67, 1991. pp. 1759.

[Eklund93] Eklund, Elliot A., Erik J. Snyder and R. S. Williams, "Correlation from randomness: quantitative analysis of ion-etched graphite surfaces using the scanning tunneling microscope," *Surface Science*, vol. 285, 1993. pp. 157.

[Eigler90] Eigler, D.M. and E.K. Schweizer, "Positioning single atoms with a scanning tunnelling microscope," *Nature*, vol 344, 5 April 1990. pp. 524-526.

IBM Almaden. Positioned individual xenon atoms on a single-crystal nickel (110) surface to form "IBM." Use the tip to exert force on the atoms; less than required to pull the atom off the surface, but more than the lateral corrugation energy. Ultra-high vacuum system cooled to 4 K. They suspect Van der Waals attraction causes xenon atoms to be attracted to the tip.

[Fuchs89] Fuchs, H., J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel, "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," *Computer Graphics: Proceedings of SIGGRAPH '89*, Vol 23, No. 3, pp. 79-88.

[Grey93] Grey, F., D.H. Huang, A. Kobayashi, E.J. Snyder, H. Uchida, M. Aono, "Time-resolved atomic-scale modification of silicon with a scanning tunneling microscope," *Proceedings of the 1993 STM symposium (Beijing)*.

Ibaraki, Japan. Watched tip displacement and tunnel current during and after a pulse. W tip, Si(111) surface. UHV. Surface cleaned by repeated flash heating, tip etched and then electron-bombarded (bombardment was **critical** to achieve reproducible modifications). Scan at +2V bias, 0.6 nA current. Pulse to -5.5V for 30ms. About 10% of pulses caused change. No control over whether extraction or deposition occurs, but extraction is dominant occurrence. Typically 1-5 atoms moved. There were often several modification events after a single pulse. Some events occur up to 340 ms or more after the pulse. Thus, many changes occur when the electric field is not the highest. Modifications usually occur within 3 nm of the tip location, but not always dead on (this makes it hard to provide a direct correlation between modification size and tip displacement). Tip displacement does give good indication of whether a deposition or removal occurred.

They put a low-pass filter to round pulse ends, in order to reduce current transients at pulse onset and end. This produced no change in behavior. They also tried current pulses, but these only modified about 1/10 as often. These results let them to think that the electron current is not the dominant reason for the modifications. They think a higher field during pulse onset (while tip being pulled back) causes field evaporation of ions from the surface (most changes that happened at this time were extraction events). They are not sure about the ones that came after the pulses, but note that most are deposition events.

[Holloway93] Holloway, Richard and Anselmo Lastra, "Virtual Environments: A Survey of the Technology," Technical report TR93-039, Department of Computer Science, University of North Carolina at Chapel Hill. June, 1993.

Describes the equipment needed to create a virtual world. Gives listings of available hardware with prices. Gives performance measurements when available.

Holloway, Richard, Henry Fuchs, and Warren Robinett, "Virtual-Worlds Research at the University of North Carolina at Chapel Hill," *Proc. Computer Graphics '91*, London.

Talks about the Sutherland paper and work. Talks about Pixel-Planes 5 for image generation. Talks about tracking (ceiling). Talks about HMDs (see-through). Talks about the ARM. Talks about sound. Talks about bike/treadmill. Talks about software libraries. Talks about GRIP, Walkthrough, Radiation Treatment Planning, X-Ray vision, and the games we play. Good list of credits, acknowledgments (grants), and a great bibliography.

[HSPICE92] *HSPICE User's Manual*, Meta-Software, Inc, 1300 White Oaks Road, Campbell, CA 95008. 1992.

[Kilpatrick76] Kilpatrick, Paul Jerome, "The Use of a Kinesthetic Supplement in an Interactive Graphics System," Ph. D. Dissertation in the Department of Computer Science at the University of North Carolina, Chapel Hill. 1976.

[Kobayashi93] Kobayashi, A., F. Grey, R. S. Williams, and M. Aono, "Formation of Nanometer-Scale Grooves in Silicon with a Scanning Tunneling Microscope", *Science*, 259, 1993. pp 1724-1726.

Describes surface modification theory and practice. Notes that the physical mechanisms involved in nanolithography are not well understood. Reproducibly made trenches in silicon with DC bias and provides theory about how it works. Gives equations and parameters for the action. Gives normal tunneling distance (10 Å) and current area (1-10 Å²). Stresses the importance of tip preparation. Hints at single-atom changes.

[Komuro87] Komuro, M., S. Okayama, O. Kitamura, W. Mizutani, H. Tokumoto, and K. Kajimura, "Nanometer Structure Fabricated by FIB and its Observation by STM," *Microelectronic Engineering* 6 (1987), pp. 343-348.

Shows that hidden-line removal contour methods are used for presentation of results.

Lyding, J. W. , S. Skala, J. S. Hubacek, R. Brockenbrough, and G. Gammie, "Variable-temperature scanning tunneling microscope," *Rev. Sci. Instrum.* 59 (9), September 1988, pp. 1897-1902.

Describes the construction of their low noise and low thermal drift STM design. Also provides some juicy tidbits about things to look out for in designing an STM control system and the STM itself.

[Landman90] Landman, U., W.D. Luedtke, N.A. Burnham and R.J. Colton, *Science*, Vol 248, 1990. pp. 454.

[Lyo91] Lyo, I.W. and Ph. Avouris. "Field-Induced Nanometer to Atomic Scale Manipulation of Si Surfaces with the Scanning Tunneling Microscope," *Science* 253 (1991), 173.

[Maganov91] Magonov, S. N., G. Bar, E. Keller, E. B. Yagubskii, and H. J. Cantow, "Atomic scale surface studies of conductive organic compounds," *Synthetic Metals*, 40 (1991), pp. 247-256.

Tries to fit models of molecules to the STM data. Shows that grayscale bitmap images are used in presentation of data.

[Mamin91] Mamin, H. J., S. Chiang, H. Birj, P. H. Guethner, and D. Rugar, "Gold deposition from a scanning tunnelling microscope tip", *Journal of Vacuum Science and Technology*. B 9 (2), Mar/Apr 1991, pp 1398-1402.

IBM Almaden. Etched gold tips used with pulses to deposit nanometer-size blobs of gold onto gold or platinum. Fast and reliable. They think it is due to field evaporation. Terminated coax cable to the tip to preserve pulses. Give specs on pulse width and height. Talked about both putting down and removing material. Static discharge in the room can cause surface/tip modifications! Blobs are on the order of 100Å-2000Å. Less reliable in UHV than in air. Blobs were stable. Tips were stable even after many pulses.

[McCord86] McCord, M.E. and R.F.W. Pease, *Journal of Vacuum Science Technology*, B4, 1986. pp. 86.

[Mine93] Mine, Mark, "Characterization of End-to-End Delays in Head-Mounted Display Systems," Technical report TR93-001, Department of Computer Science, University of North Carolina at Chapel Hill. March, 1993.

Presents the results of measurements of tracker, communication, and image generation latency in our system. Includes measurements of several types of trackers.

[Oppenheim83] Oppenheim, Alan V., Alan S. Willsky, Ian T. Young, *Signals and Systems*, Prentice-Hall Signal Processing Series, 1983.

Ouh-young, Ming, "Force Display In Molecular Docking," Ph. D. Dissertation, University of North Carolina, Chapel Hill, TR90-004, February, 1990. Ph. D. Dissertation.

[Ringger85] Ringger, M., H.R. Hidber, R. Schlögl, P. Oelhafen, H.J. Güntherodt, "Nanometer Lithography with the STM", *Applied Physics Letters*, 46 (9) 832, 1985.

[Robinett92] Robinett, W., and R. Holloway, "Implementation of Flying, Scaling, and Grabbing in Virtual Worlds," *ACM Symposium on Interactive 3D Graphics*, Cambridge, MA (1992).

[RobinettCourse92] Robinett, W., R. Taylor, V. Chi, W. V. Wright, F. P. Brooks Jr., R. S. Williams, and E. J. Snyder. The Nanomanipulator: An Atomic-Scale Teleoperator. *SIGGRAPH '92 course notes* for "Implementation of Immersive Virtual Environments."

[Snyder91] Snyder, Eric J., Mark S. Anderson, William M. Tong, R. Stanley Williams, Samir J. Anz, Marcos M. Alvarez, Yves Rubin, François N. Diederich, and Robert L. Whetten, "Atomic Force Microscope Studies of Fullerene Films: Highly Stable C60 fcc (311) Free Surfaces," *Science*, Volume 253, 12 July 1991, pp. 171-173.

Study of macroscopic packing of C60 using AFM and X-Ray diffractometry. They were not closely packed, as monolayer films tend to be.

[Stoll91] Stoll, E. P., "Picture processing and three-dimensional visualization of data from scanning tunneling and atomic force microscopy," *IBM Journal of Research and Development*, Vol. 35, No. 1/2. January/March 1991, pp. 67-77.

Describes the basic properties of tunnelling, the sources of noise in an STM feedback control system, and the standard methods of visualizing the data. The language is close to my level and provides a good introduction to several aspects of STM data processing and visualization.

[Stroscio91] Stroscio, J.A. and D.M. Eigler, *Science*, 254, 1991. pp. 1319.

Sutherland I.E. (1965) "The Ultimate Display", *Proceedings IFIP Congress 1965*.

The first published vision of V.R. We are still trying to achieve it.

[Taylor90] Taylor, Russell M., "Argonne-III Remote Manipulator (ARM) User's Manual and Programming Guide," University of North Carolina at Chapel Hill (internal document). 1990.

Describes the ARM library in use at UNC. Talks about the theory behind the code used in the library. Soon to be outdated when the ARM library is redone to include the new SARCOS ARM.

[Taylor93] Taylor, Russell M., Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Erik J. Snyder, "The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope," *Computer Graphics: Proceedings of SIGGRAPH '93*, August 1993. pp. 127-134.

This paper describes an early version of the Nanomanipulator system. It gives an introduction to the STM, points out some previous visualizations systems, describes the system from the user's point of view, describes the system hardware and software, and presents early results (seeing up-tilted graphite planes, replication of Mamin's gold pulse results). Describes why what we think will be the big wins of such systems (**on** the surface, **in** control, **while** things are happening).

[Taylor93VIR] Taylor, Russell M., Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Erik J. Snyder, "I dominatori dell'atomo", *VIRTUAL magazine*, Number 2 October 1993, pp. 18-21.

Translation of the SIGGRAPH '93 paper into Italian; condensed more technical parts, since it was for a more general audience.

[Taylor94SSL] Taylor, R., R.S. Williams, V.L. Chi, G. Bishop, J. Fletcher, W. Robinett and S. Washburn, "Nanowelding: Tip response during STM modification of Au surfaces," *Surface Science Letters*, vol 306 (Numbers 1 and 2), March 1994, pp. 534-538.

[Taylor94IMG] Taylor, Russell M., Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Erik J. Snyder, "The Nanomanipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope," *Proceedings of Imagina '94*, February 1994. pp. 69-76.

Reprint of SIGGRAPH '93 paper.

[Taylor94LFW] Taylor, Russell M., Vernon L. Chi, "Real-time Visualization of Data from Scanning Probe Microscopes," *Laser Focus World*, to appear in May 1994 issue.

This paper presents the Nanomanipulator system for an audience that is familiar with the concepts of scanning probe microscopy, but perhaps not familiar with the jargon of computer graphics. It provides a

description of the system's capabilities and discusses the benefits of using a virtual reality interface over the classical interfaces.

APPENDIX A — SDI LIBRARY

There are several head-mounted display (HMD) stations in use in the graphics lab; each uses a tracker, a hand-held controller, and a sound server. Connections to most trackers, most controllers, and the sound server are made through serial lines. Many stations can be used by more than one Pixel-Planes host. In addition, we have serial-line or other connections to VCRs, video switches, joystick boxes, and speech recognition servers. Some of these connections go through our building telephone switch, while some are directly connected to workstations on our Ethernet LAN. Two joysticks, located at Brown University and the California Institute of Technology, have been used to control programs running on a computer at the University of North Carolina at Chapel Hill.

In order to organize connections to devices through such a wide variety of communication schemes, I have developed the standard device interface (SDI) library. This library provides application programs with a uniform method of connection to a device regardless of the actual means of establishing the connection. The application requests a connection to a device by name, and the library returns a file descriptor that the application can use to communicate with the device.

Types of connections

The SDI library currently supports three methods of connecting to a device: phone connections, connecting to a well-known port, and starting a remote server.

Our department has an Intecom IBX_{TM} phone switch to route calls within the building. We can use asynchronous data interface (ADI) boxes to connect serial devices to each other through the switch. The computer placing the outgoing call gives the phone number of the device it wants to connect with to the ADI, which establishes connection through the IBX_{TM} switch. This

connection scheme has the advantage of only running one serial line from each device and each computer, while allowing any computer to connect to any device.

Some devices have server programs that run constantly and accept connections from remote client applications. These servers listen at a specific Internet protocol (IP) port on the computer that controls the device. Connections to these devices are made by establishing a transmission control protocol (TCP) connection to the given port on the server. This is a stream connection that is guaranteed to pass characters reliably from end to end. This connection method allows a computer to connect to a device that is attached to another computer on the Internet. Examples include the force-feedback ARM and Scanning Tunneling Microscope (STM), each of which is controlled by a dedicated Intel i486-based personal computer.

At times it is not practical to constantly leave a server program for a device running. In these cases, the SDI library starts a server program on the computer attached to a device when an application requests a connection to the device. This is often used to access a device connected to a serial port on a remote workstation. The library runs a remote shell command to start the server on the computer that the device is connected to; the server establishes a TCP connection back to the application. The Macintosh sound server is an example: it is connected to a DECstation 5000 via a serial port, but Pixel-Planes is connected to a Sun-4. When the Pixel-Planes application starts, it runs a serial server on the DECstation, telling it which port on the Sun to connect to. Once the server has started, it establishes the connection and then passes characters between its serial port and the Sun.

Types of servers

The most frequently used type of server is the *serial* server. There are two types, one that runs constantly waiting for connections and one that is started each time it is needed. Each type provides a connection between a serial line on the computer running the server and a TCP stream connection to the client application running on the remote host. Any data coming in the serial line is sent over the TCP connection, and any data coming in the TCP connection is sent over the serial line. This allows client applications to access

a serial device connected to a remote computer as if it were local. This server is used to connect to the hand-held button controllers, joystick boxes, and other devices.

A second type of server, called a *bounce* server, is used to test network latency from one machine to another. There are two types of bounce servers, one that runs constantly and accepts connections and one that is started up only when needed. Both kinds establish the connection to the client program and then echo characters received back to the client as soon as they arrive. A test program has been written to use bounce servers to measure the round-trip latency of messages (see below).

The SDI library also provides several routines to aid in the creation of special-purpose device servers. Several projects have made use of these to create servers for:

- a scanning tunneling microscope (STM)
- an optical tracker
- the button box on a Silicon Graphics IRIS workstation.

Routines are provided to connect to a client and set up the communication channel, and the same name-mapping mechanism that is used for standard servers (as described below) is used to start these servers.

Mapping device names to connections

The mapping from device name to connection path is stored in a text file named *.sdi_devices*. There is a system-wide version of this file whose location is compiled into the library. If a file with this name is in the user's home directory, any definitions in that file override those in the system-wide file. The *.sdi_devices* file contains lines of text, each of which maps from a device name to the procedure used to access that device. Each line is composed of five fields: DEVICE, MAP, MACH, PROGRAM, and ADDITIONAL ARGS.

The DEVICE field gives the name of the device that is being described. Names should all be unique — only the first occurrence of a name is used.

The MAP field is an integer field that will enable a single server to handle more than one device. To date, no such servers have been written.

The MACH entry tells the Internet name of the machine that the device is connected to or can be reached from. If the device is reached using the department phone switch, this field is ADI and the PROGRAM field is IBX.

The PROGRAM line gives the full path name of the program to run on the machine to get the server going. If the program field has PORT in it, then a connection will be made to the port listed in the ARGS field but no program will be run beforehand. This is to allow connections to an already-running server.

The ADDTNL ARGS are all passed to the server program, along with other information that is described in the manual pages about SDI. In the case of a PORT connection, the port number in decimal ASCII should be listed here. In the case of an ADI IBX connection, the serial port to connect to the IBX on, baud rate, and phone number are here.

Tricks to increase performance

The initial versions of the SDI library would start a server on the remote machine, wait a while, and then attempt to connect to that server at a well-known port. The server would attempt to start up and listen on the port before the connection request came from the client. Due to delays starting up, the wait time would either be too long or not long enough. In addition, the server had to be able to get the specified port — which was sometimes unavailable because an earlier failed server process still had it open.

To overcome these difficulties, the connection scheme was modified. Now, the client application opens any available port and listens for connections from the server; it starts the server, passing the address of the port it has opened as a parameter. The server opens any available port on its computer and connects back to the client at the specified port as soon as it starts. This method insures both that the server will have an available port (since it can accept any port) and that the connection will be established as soon as the server process has started, thus reducing the time for connections.

To reduce latency in messages for some devices, it was necessary to modify the behavior of the TCP connection. To ensure that all data sent over a TCP connection arrives, an acknowledgment is sent for each packet received,

at which time the sender deletes the information. In order to use the transmission medium efficiently, each end usually waits before sending the acknowledgment in the hope that it can piggy-back the acknowledgment on data that is sent from the local host back. This works well for connections that have data flowing continuously from each end; but on lines where one end sends and the other does not, the sender can be held up waiting for the acknowledgment of earlier data. To avoid this problem, the SDI library sets the TCP connection to not delay its acknowledgments waiting for data to return. It does this using the `TCP_NODELAY` input/output control (`ioctl`). This optimizes the connections for low-latency transmission of data, especially data that flows in only one direction. Since many of our devices send data in only one direction, this increases our performance.

The last trick is one we learned empirically, and we do not have an explanation for why it works. It turns out that if one attempts to write multiple times to a TCP connection immediately after one another, the transfer takes a large amount of time (on the order of seconds); if one pauses for 1/60th of a second or so between writes, the messages arrive about 1/60th of a second apart. To avoid long delays, the serial server and other servers throttle the sending of packets; waiting at least of 1/60th of a second between writes. Reading from the socket multiple times does not cause this behavior.

Performance studies

Tests of the latency when reading from a tracker that was connected to a Sun-4 local serial port vs. one that was connected via a TCP connection to another Sun-4 using SDI indicated that the TCP connection had an average of 4-8 milliseconds more latency than the local connection (for a Polhemus 3Space tracker, the tracker latency went from about 30 ms to about 34 ms). [Mine93] Later tests that examined a tracker connected to a DECstation 5000 indicated large gaps in the data transfer (on the order of a second). These were later traced to the serial port device driver on the DECstation (local connections also had these large gaps). These tests indicate that the local serial port driver on the machine the device is connected to has a much larger effect than the ethernet transmission times on latency.

To get a better idea of the types of delays to be expected when using an ethernet to transmit real-time data, a program was run that used a bounce server to echo back short messages. The two-way transmission time for each message was noted and a histogram made showing the distribution of time delays (see Figure A.1). For a run of 3000 messages between Sun-4's on our Ethernet, all but 6 took less than 0.01 second for the two-way transmission, five more were less than 0.02 seconds, and one was less than 0.1 second but more than 0.05 second. This data seems to match our experience when using such connections over a large number of trials. There are times when the delays are much worse, but during these rare times nothing else on the network works reliably either.

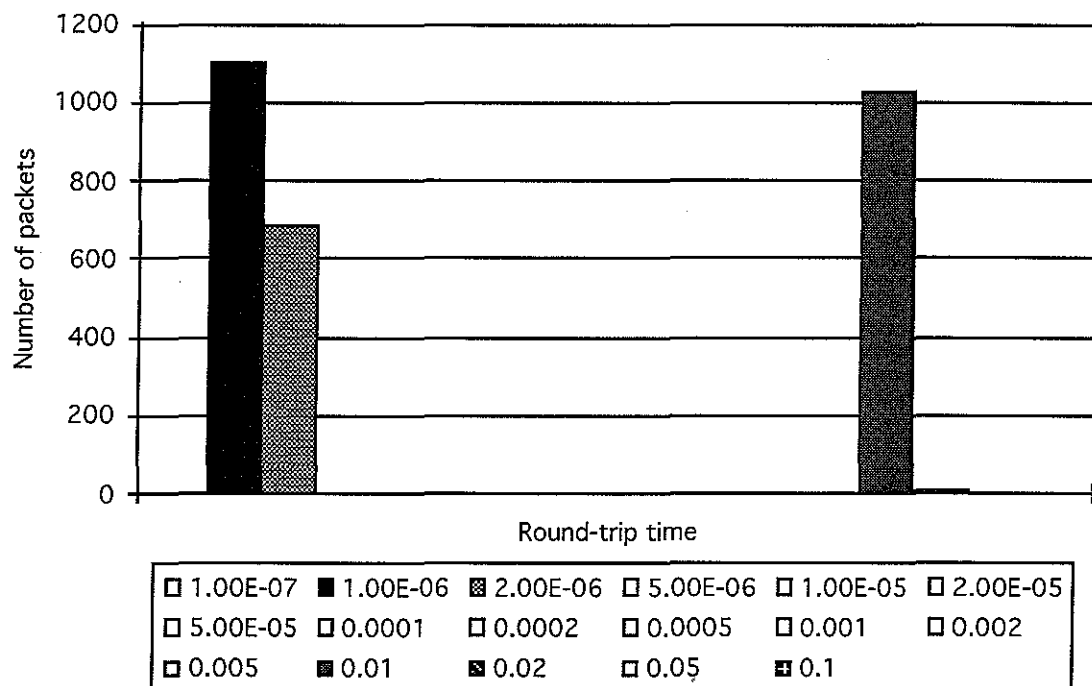


Figure A.1 Histogram of round-trip latencies. The latencies form two groups, one under 2 microseconds and another at 10 ms.

Suitability of Ethernet connections

We find that ethernet TCP connections work very well for handling button presses from hand-held input devices in a virtual world, for joystick control of object rotation, and for sending sound commands to our sound server. No user has been able to tell the difference between a local connection

and an SDI connection to these devices. The buttons and sound server seem to work well no matter what type of host they are connected to; the joysticks have been tested and work well on Suns and HP 700-class machines.

Additionally, the PCs controlling the force-feedback ARM and Scanning Tunneling Microscope (STM) in our department are connected to the user application via ethernet and work well. In the Nanomanipulator project, the user is part of a feedback loop that requires reading from the ARM, writing and reading from the STM, and then writing to the ARM again; the system response is still acceptable (the 20-Hz display update rate is not reduced when using the ARM in this mode).

Our experience with using trackers on remote hosts indicates that the host serial line performance impacts the performance more than the ethernet transmission time. Trackers work acceptably when connected to Sun-4's running a serial server, but not when connected to DECstation's. We have not investigated ways of speeding performance on the DECstations, nor have we tested other architectures. We now establish connections to trackers via telephone lines in order to avoid the 4-8 ms ethernet latency.

APPENDIX B — NOISE IN STM INPUTS

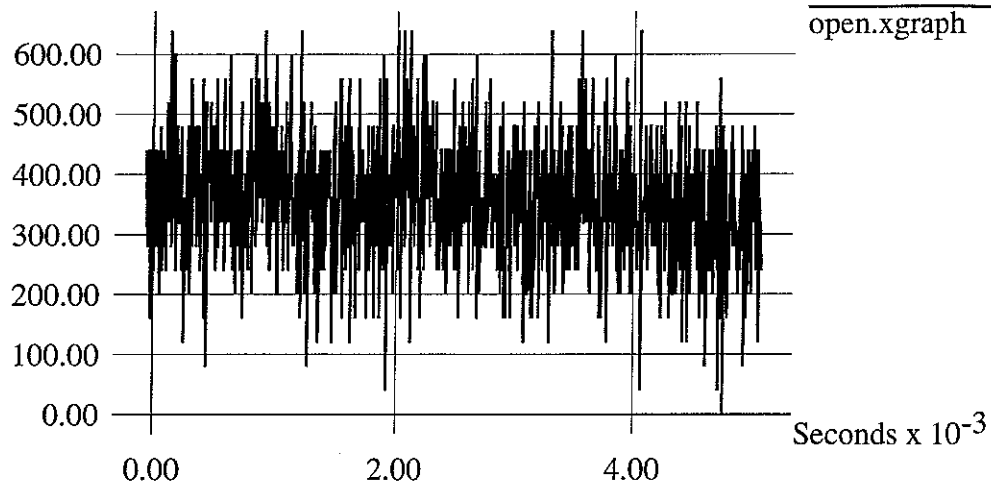
The graphs shown here are of noise tests of the various inputs to the STM that was first installed at UNC (the one built by Elliot Eklund at UCLA). More information can be found in Russ Taylor's lab notebook on 8/20/93 and 8/24/93, and a summary can be found in chapter 5. The files were taken from a Tektronics DSA 602 digital signal analyzer, and they come in pairs. The first of each pair measures the time-domain signal in volts. The second of each pair (labelled `_fft`) contains the FFT of the data in decibels. There was DC suppression and signal averaging on the FFT data. The input was DC-coupled into a high-impedance input on all but the `y_dac_long`, `y_dac_short`, `y_dac_medium`, and `y_dac_zoom` runs. There, it was AC-coupled into a high-impedance input.

The first set of tests (`open`, `terminated`, `cable`, `cable_terminated`) was made to get an idea of the minimum noise detectable by the DSA 602 and the cable used for testing. They show that the noise floor is about 0.6 mV and that the frequency spectrum is fairly flat across the region tested.

Note that there was a line-synchronous ground noise of about 2 mV p-p that shows up during the testing of electronics (particularly noticeable on the `y_dac_still_20kHz` graph). Also, the scope itself seemed to have a frequency component at about 35 kHz (seen on `open` and `battery` tests).

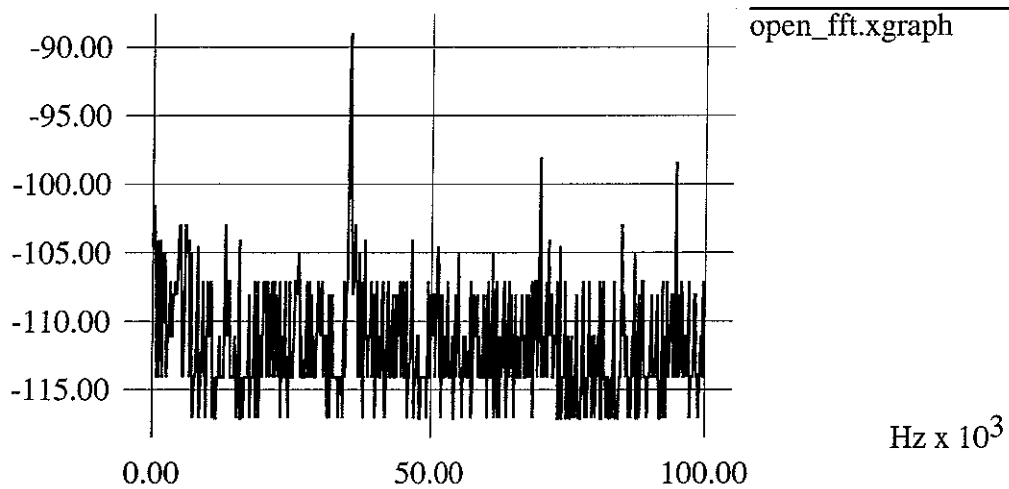
X Graph

Volts x 10^{-6}



X Graph

Decibels

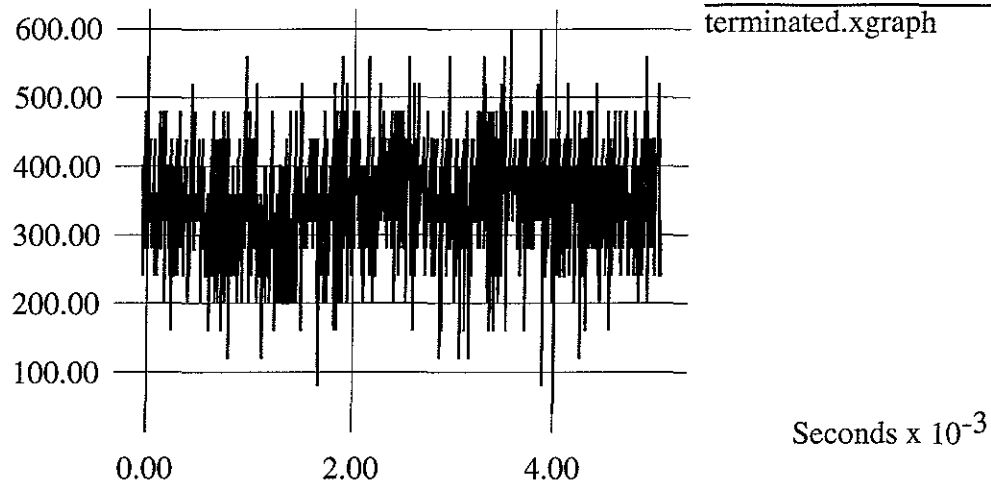


open:

This is measurement taken with the input connector open on the DSA 602. This shows noise about 0.65 mV noise peak-to-peak with spikes FFT at around 35 kHz and harmonics.

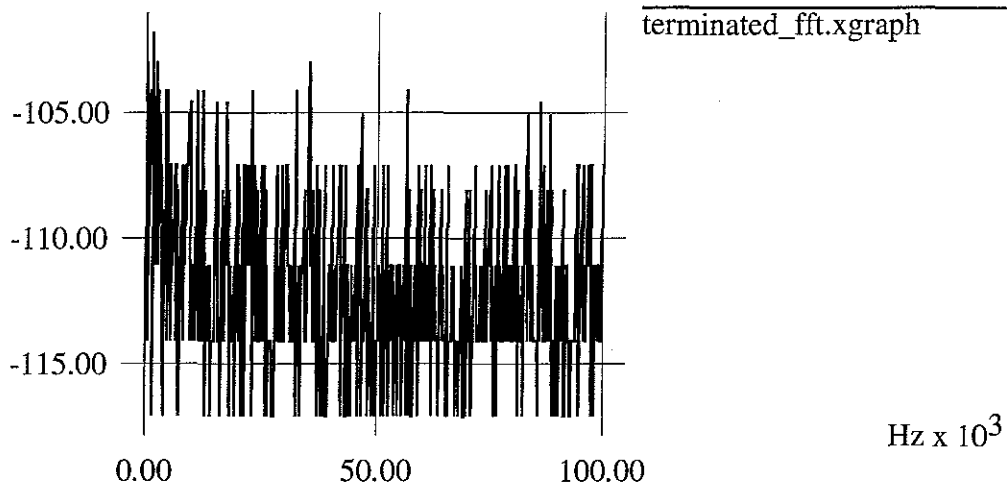
X Graph

Volts x 10^{-6}



X Graph

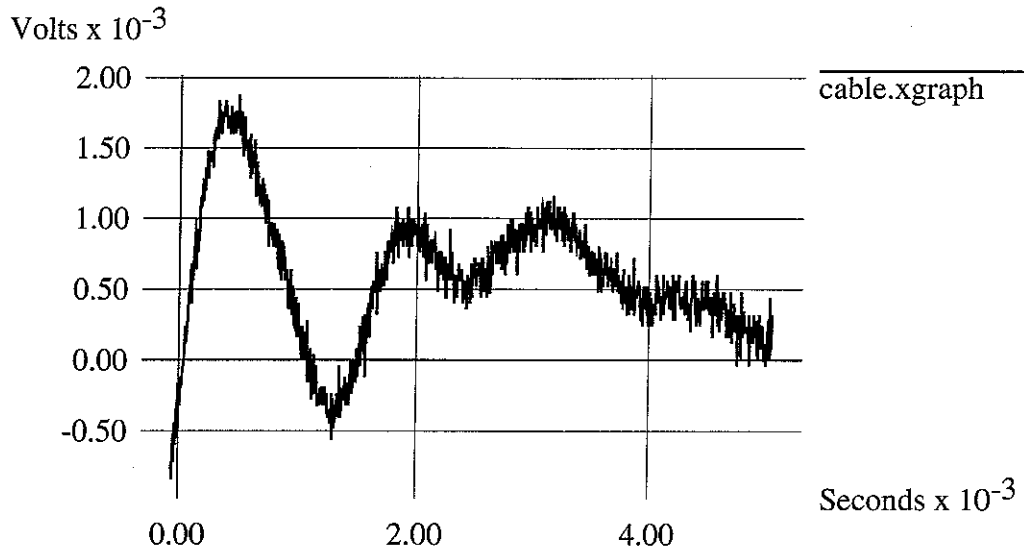
Decibels



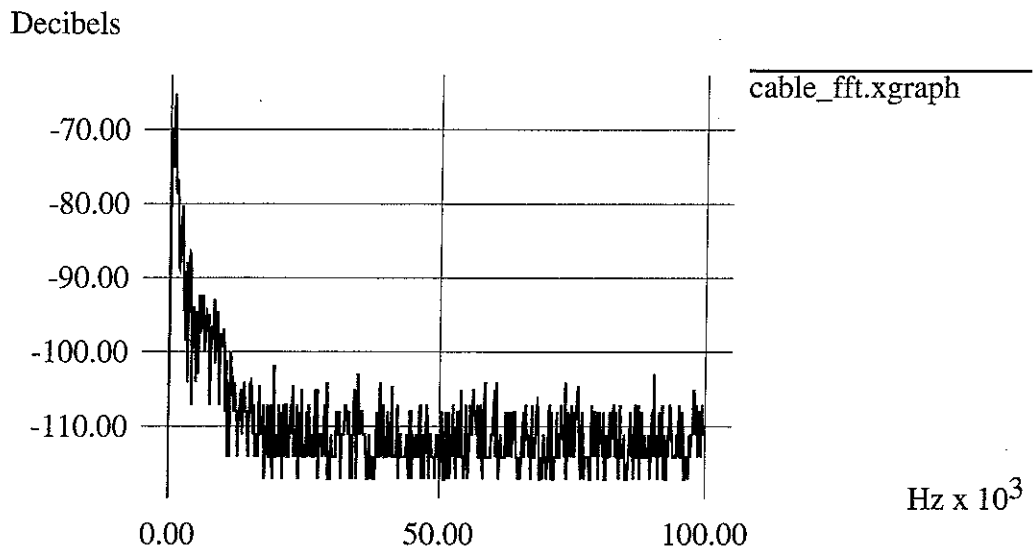
terminated:

This is measurement from when a 75-ohm terminator was plugged directly onto the DSA input. It shows about 0.6 mV noise peak- to-peak with no appreciable peaks in the FFT.

X Graph



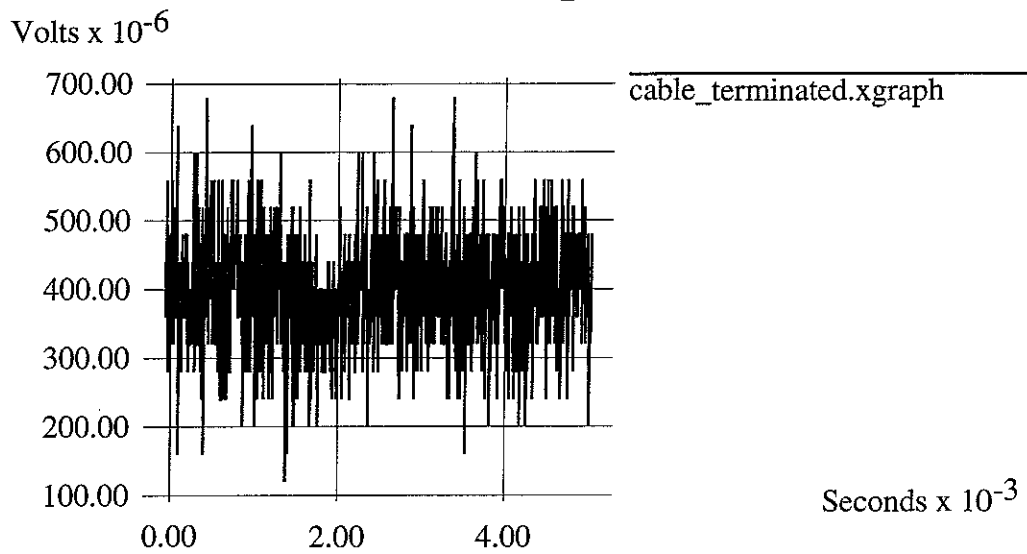
X Graph



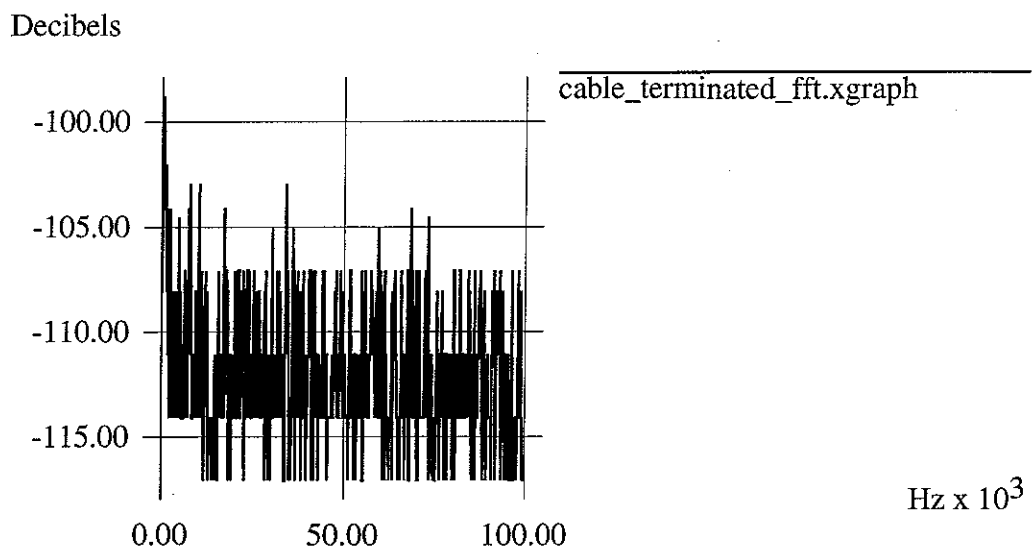
cable:

This is measurement of the open-ended coax cable. This is the cable used in the rest of the tests. I was striking the cable against a hard surface, which produced signals of at least 3 mV peak-to-peak that looks like 1/f noise in the FFT. Note that when the cable is terminated, this noise goes away (see below).

X Graph



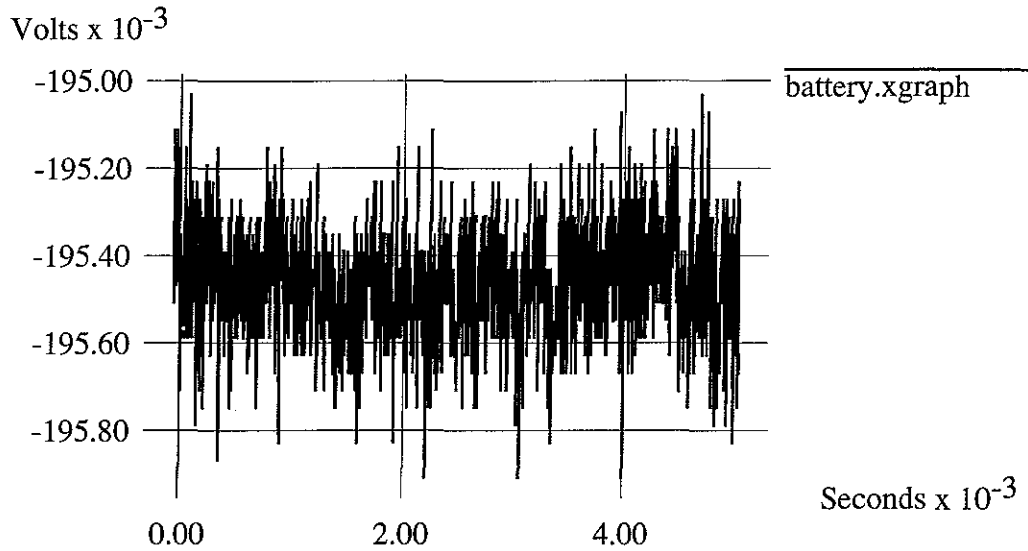
X Graph



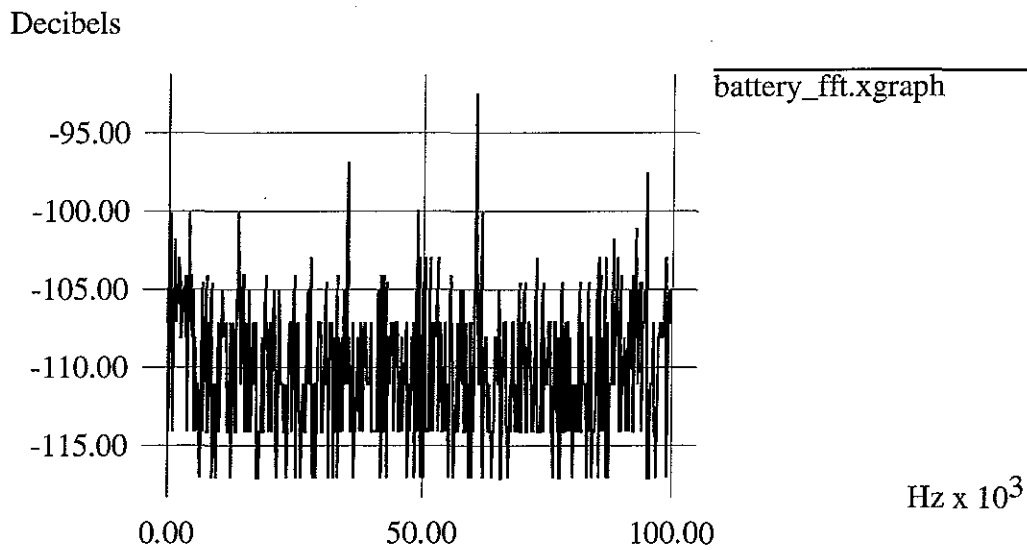
cable_terminated:

This is measurement of the cable with the far end terminated using a 75-ohm terminator. It shows about 0.6 mV peak-to-peak noise with a peak in the FFT at about 500 Hz.

X Graph



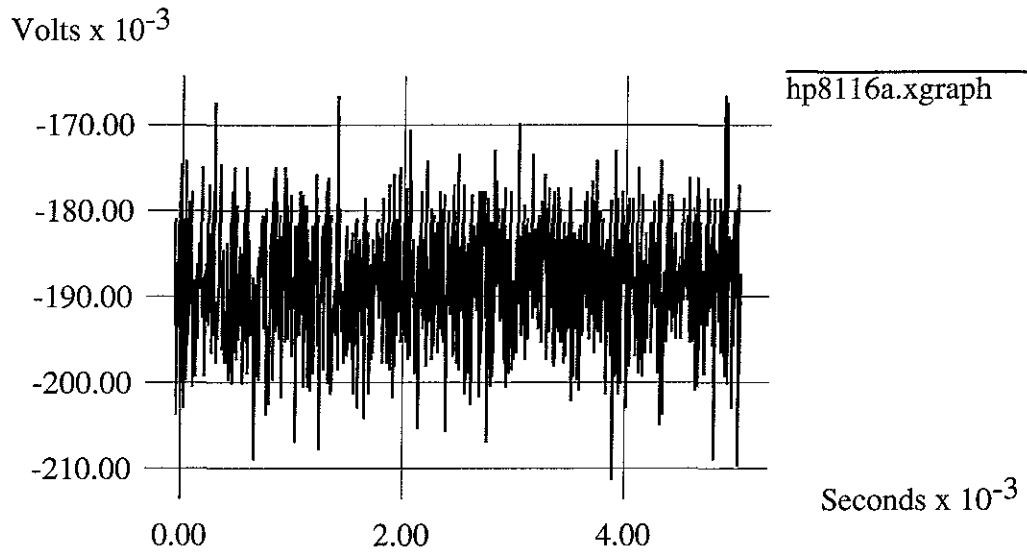
X Graph



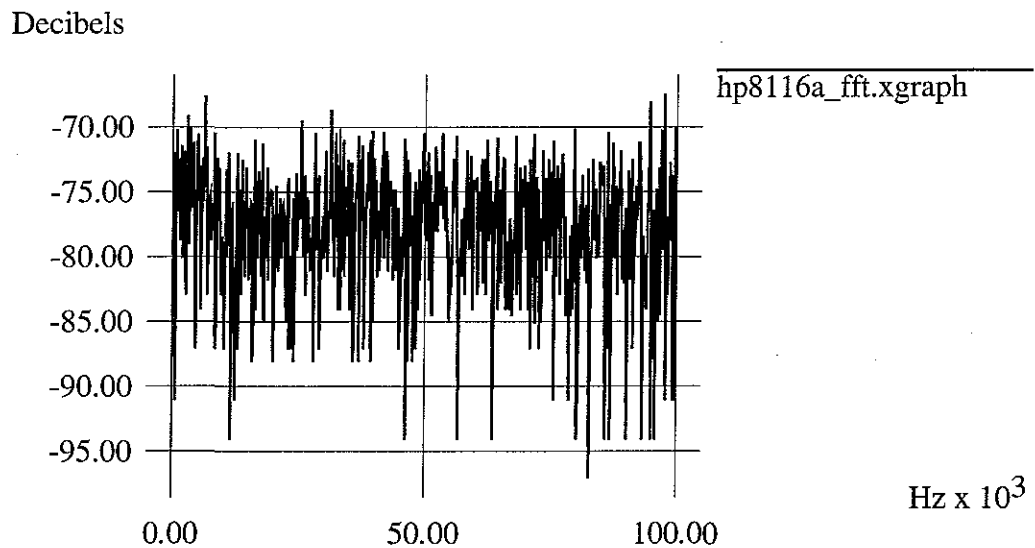
battery:

This measurement is of the battery used to provide bias voltage to the STM. It shows about 0.9 mV peak-to-peak noise with spikes at about 30-35 kHz and harmonics. The spikes are presumably due to the same source that produced them during the open-circuit tests, rather than to something in the battery. The battery case is lifted off ground.

X Graph



X Graph

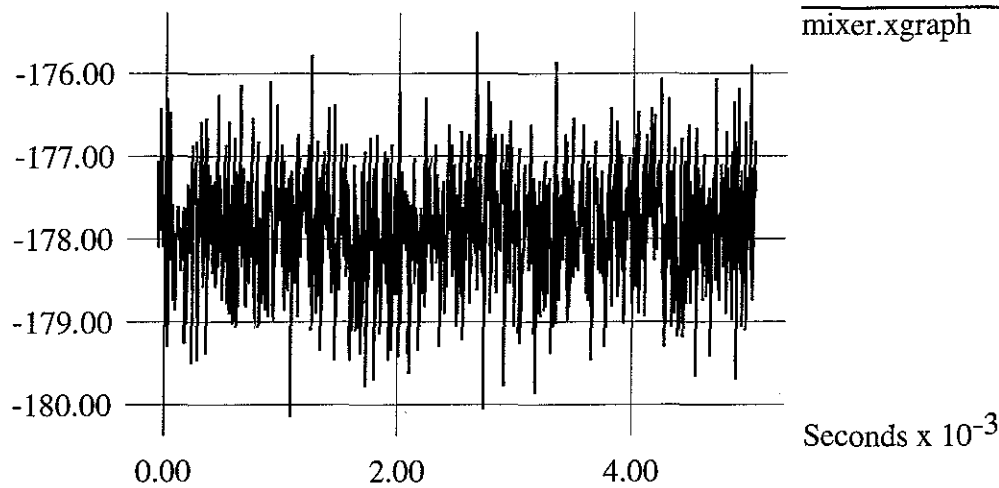


hp8116a:

This measurement is of the pulse generator that provides the controlled pulses to the STM. It used to provide bias until we found out how bad the noise on it was. It shows about 50 mV peak-to-peak noise (earlier measurements showed 75) with no noticeable peaks in the frequency spectrum (from 1 to 100 kHz).

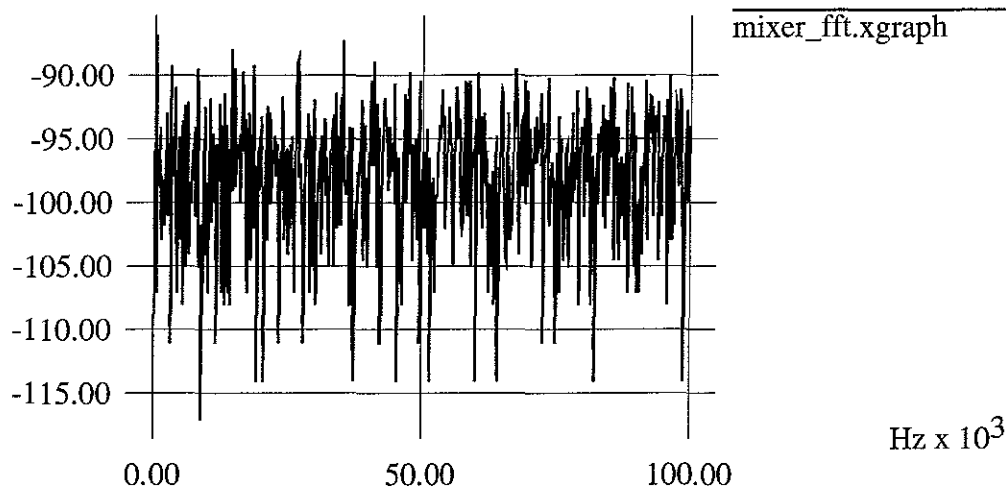
X Graph

Volts $\times 10^{-3}$



X Graph

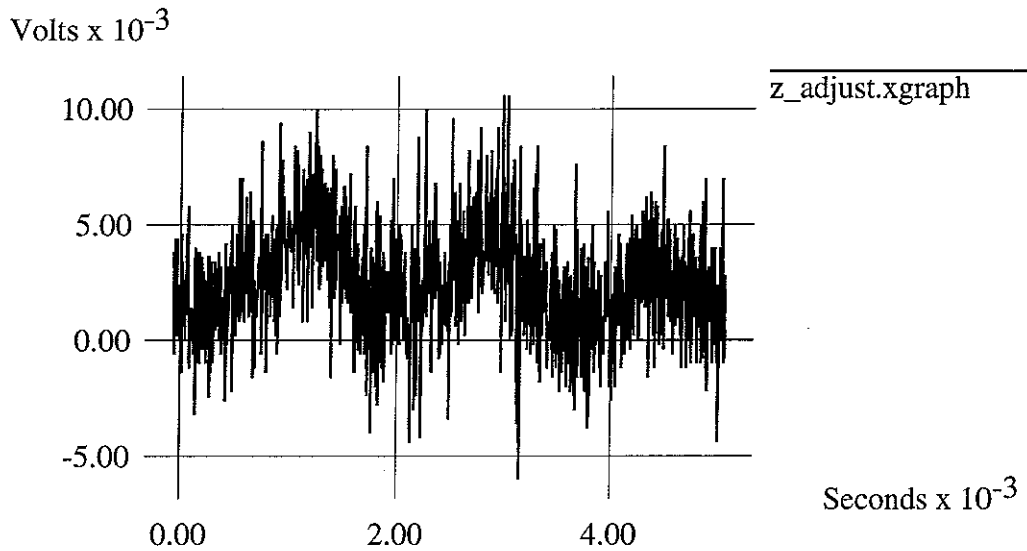
Decibels



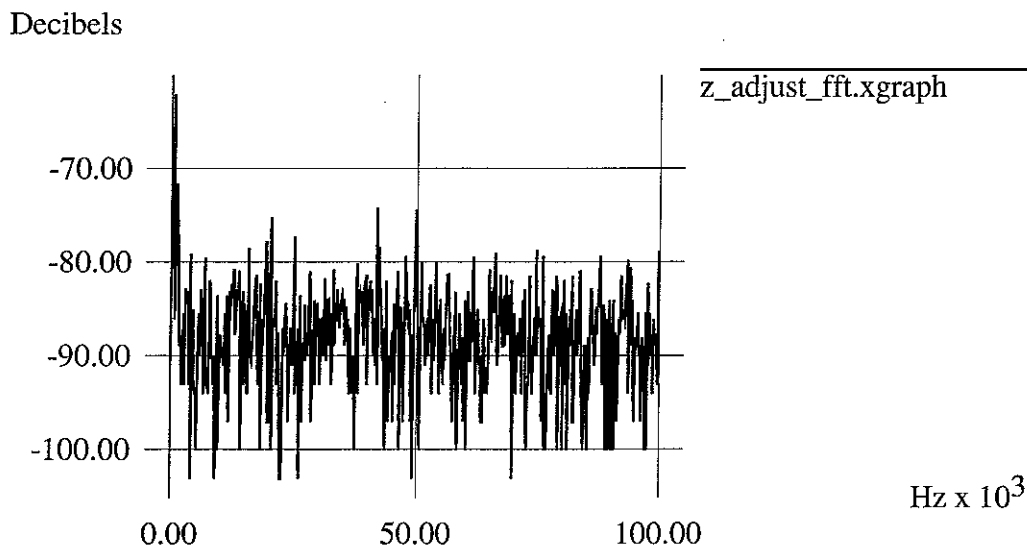
mixer:

This measurement is after the bias/pulse mixer circuit with both the battery and the hp8116a plugged into it. This shows about 5 mV peak-to-peak noise with no obvious spikes in the FFT.

X Graph



X Graph

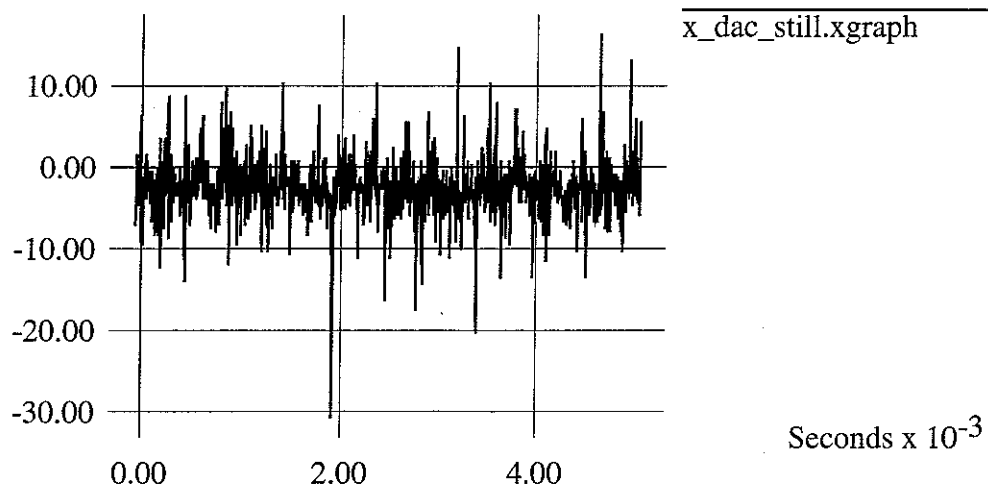


z_adjust:

This shows the Z coarse adjustment noise. This noise has a strong component at about 585 Hz, clearly visible in the time-domain signal. The total noise is about 16 mV p-p, with the 585Hz signal oscillating about 7 mV p-p. The FFT also shows the 585 Hz spike, as well as several others (notable ones at about 20 kHz, 42 kHz and 50 kHz).

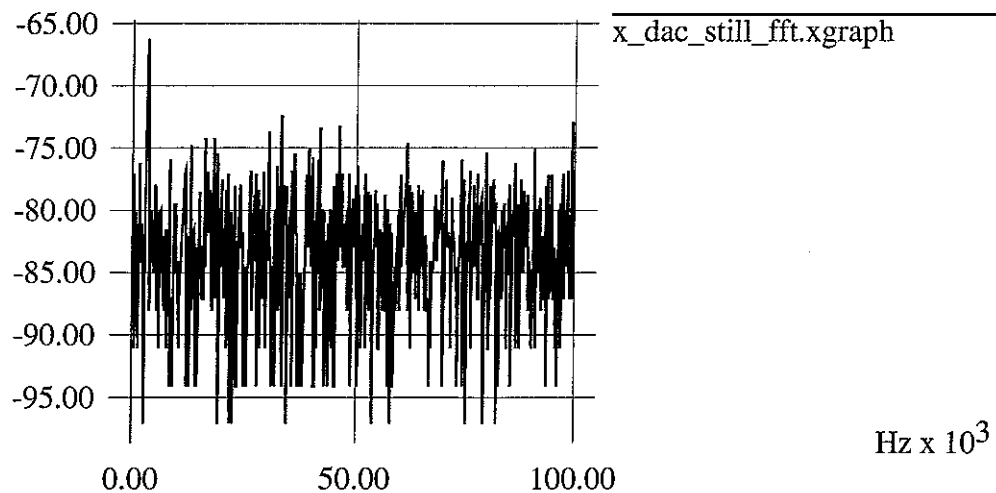
X Graph

Volts x 10^{-3}



X Graph

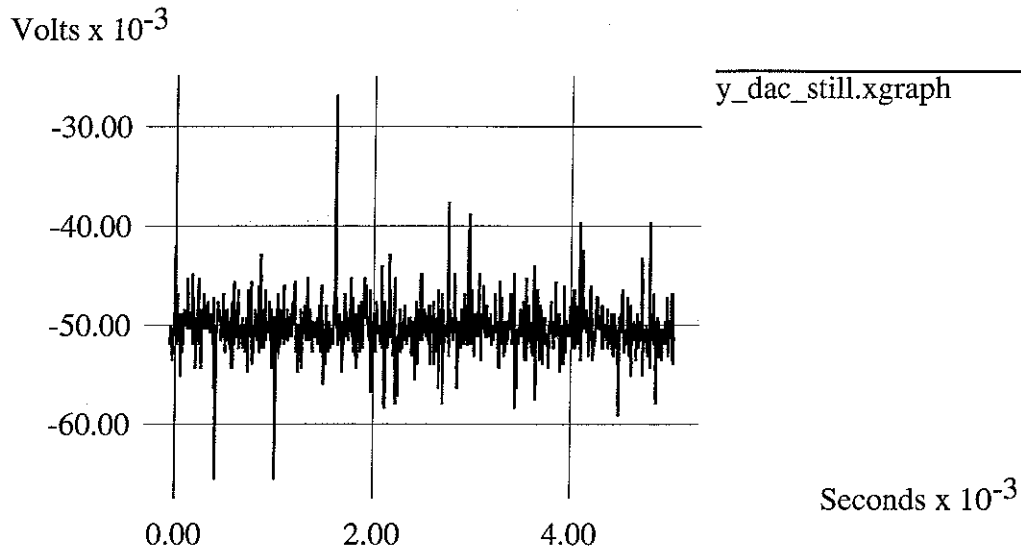
Decibels



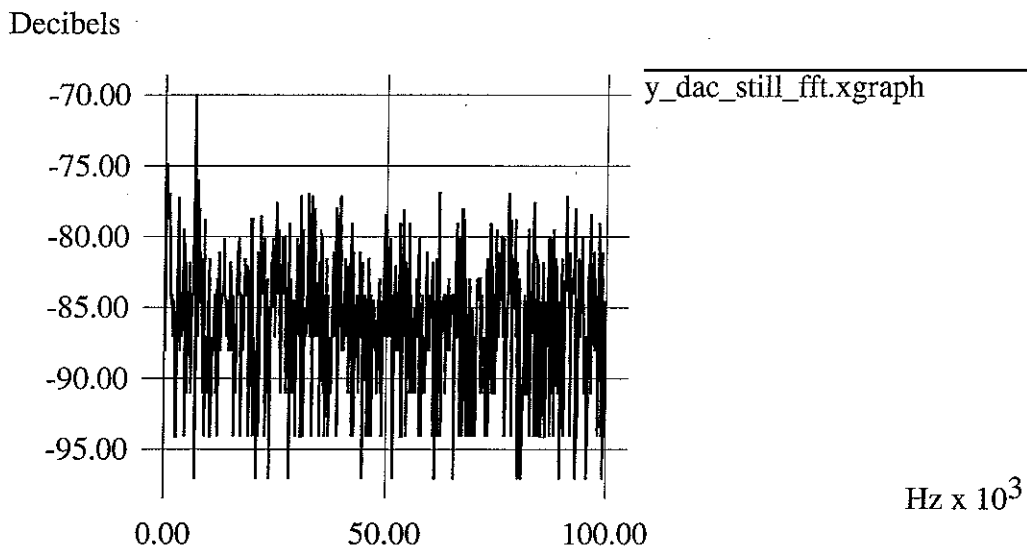
x_dac_still:

This measures the noise on the X DAC (DT2838 card, DAC 0) when the STM is not scanning. It shows about 50 mV peak-to-peak noise with a spike in the FFT at about 3.3 kHz. Since this is close to the resonant frequency of this STM (3.6 kHz) it is a cause for some concern.

X Graph



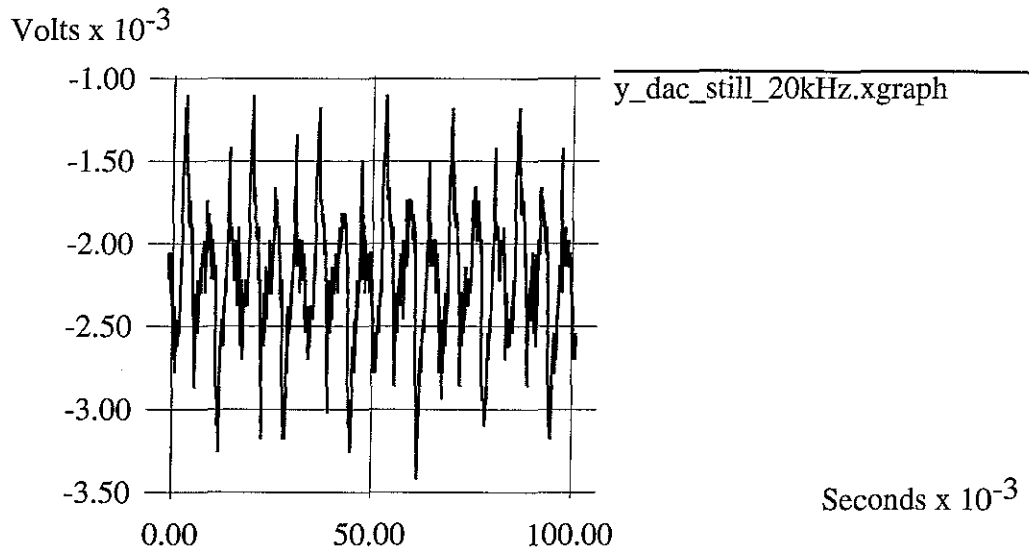
X Graph



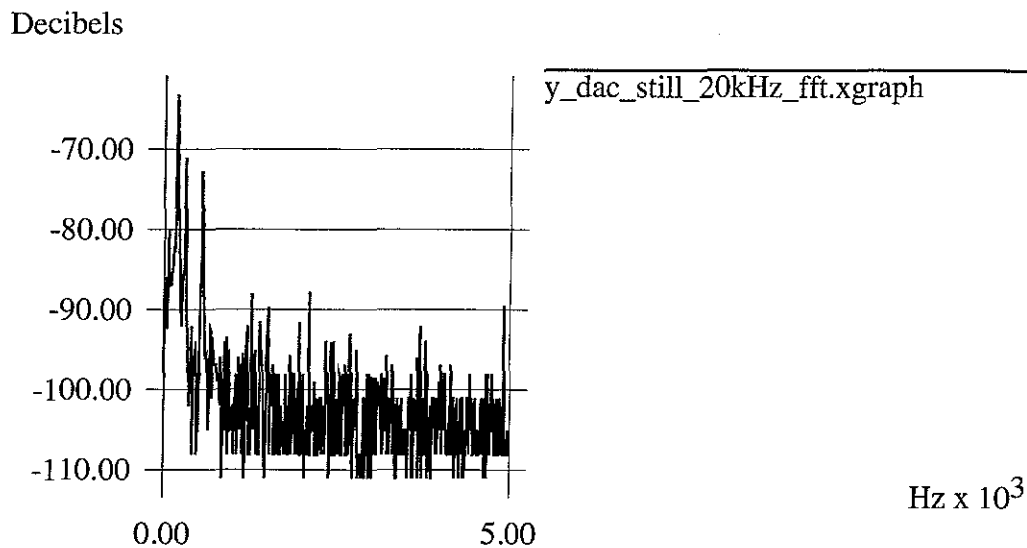
y_dac_still:

this measures the noise on the Y DAC (DT2838 card, DAC 1) when the STM is not scanning. It shows about 40 mV peak- to-peak noise with a spike in the FFT at about 6.8 kHz. Since this frequency is in the resonant zone of this STM, this is a cause for some concern. Later investigation showed that the spikes above 4 mV were removed if the signal was band-limited to 1 MHz, so filtering can easily take care of most of this noise.

X Graph



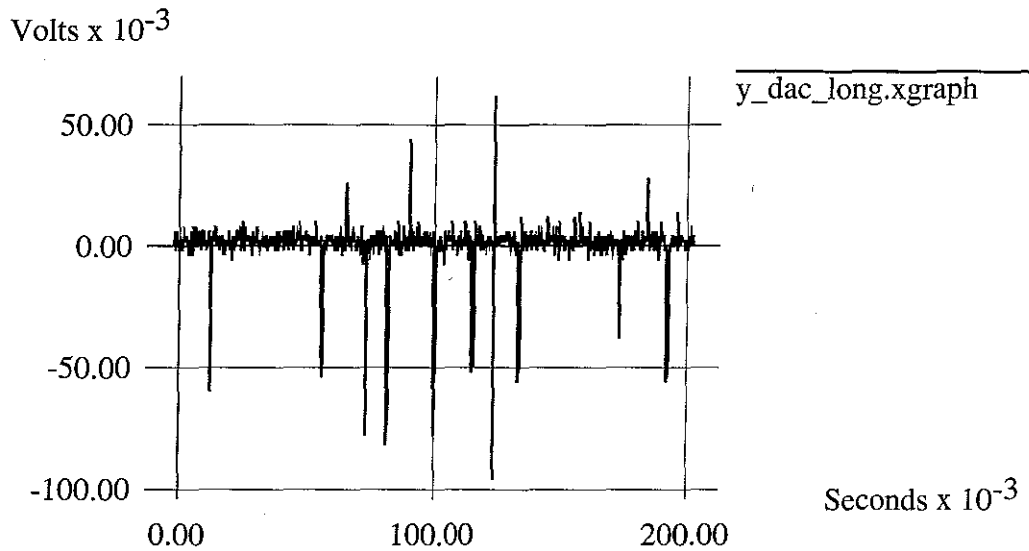
X Graph



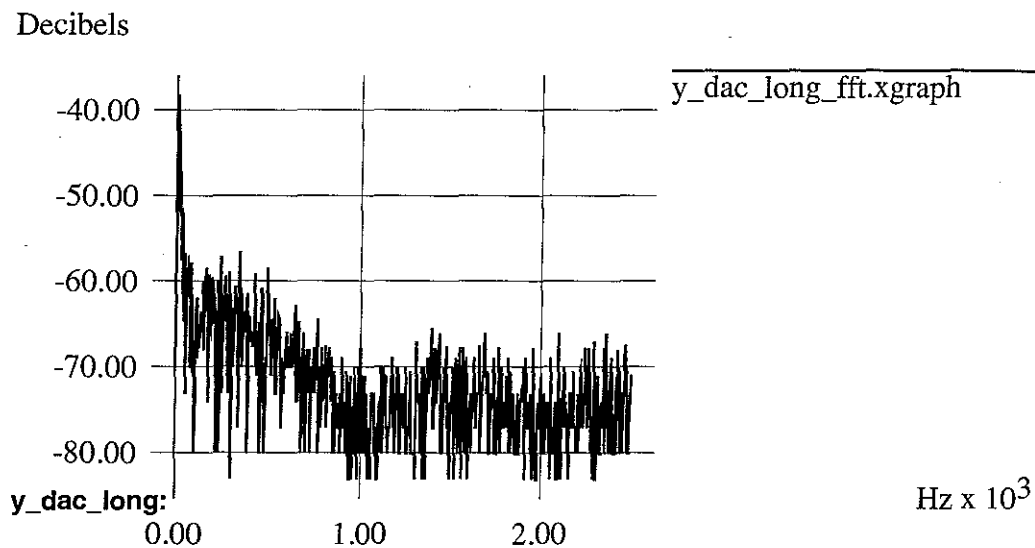
y_dac_still_20kHz:

This shows the same noise above, but with a 20kHz single-pole low-pass filter in place. This bandlimits the noise and it is reduced to about 3 mV peak-to-peak. The FFT shows a spike at 180 Hz (turns out this is synchronous with the power line) with additional peaks at 300 Hz and 540 Hz.

X Graph

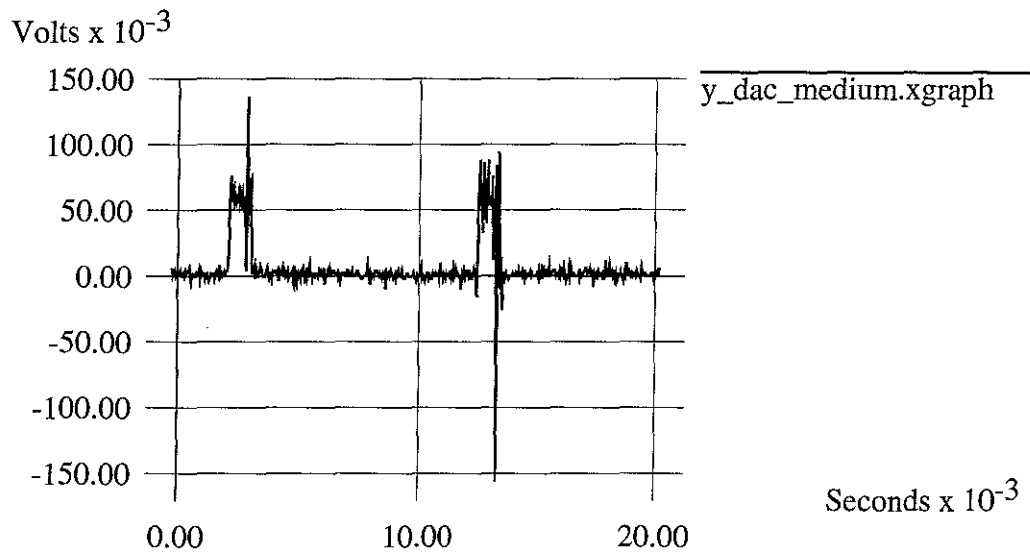


X Graph

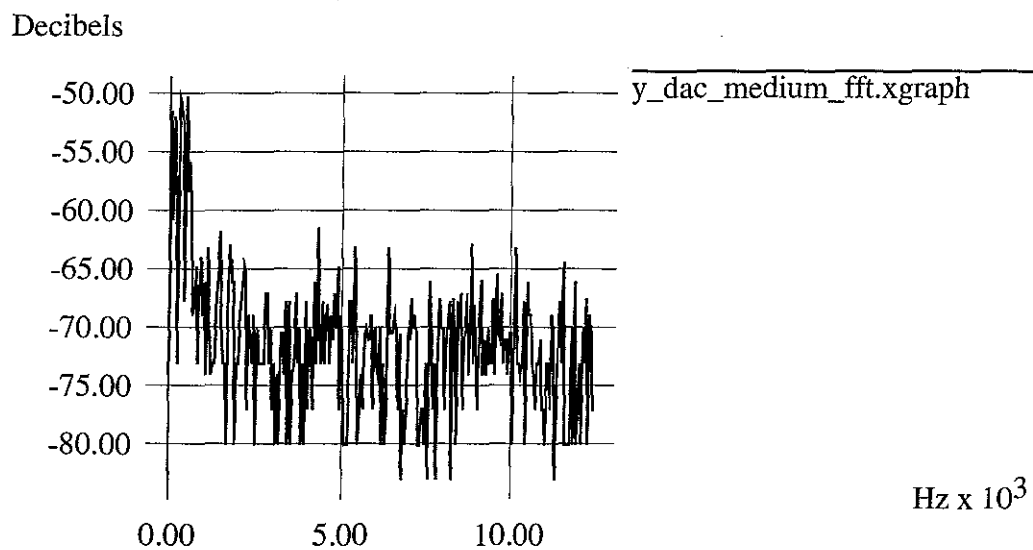


This shows a long scan of the Y DAC while the X DAC is stepping. Not captured in the trace is the fact that the y DAC is stepping about once per second. Note that the signal is AC coupled now. This shows glitches that are up to about 100 mV that occur in either + or - (200mV p-p total). These appear to be a series of sharp spikes at regular intervals (later shots will zoom in on these). The FFT shows a sharp spike at 10 Hz and a slow rolloff up to about 1 kHz. The distance between spikes in the time-domain signal seems to be about 8 ms. This corresponds to about the 128 samples/second being performed by the X DAC, leading to the conclusion that they are the cause.

X Graph



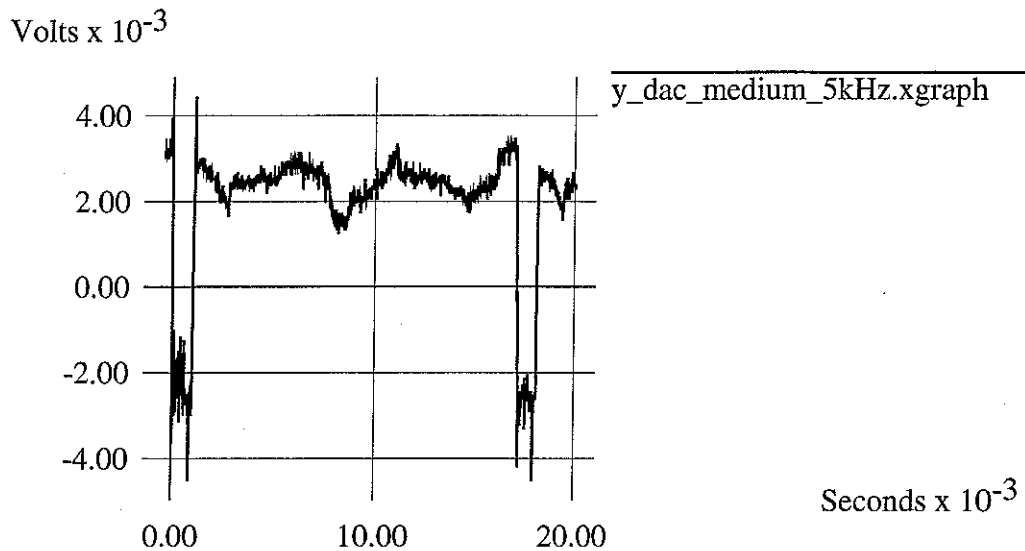
X Graph



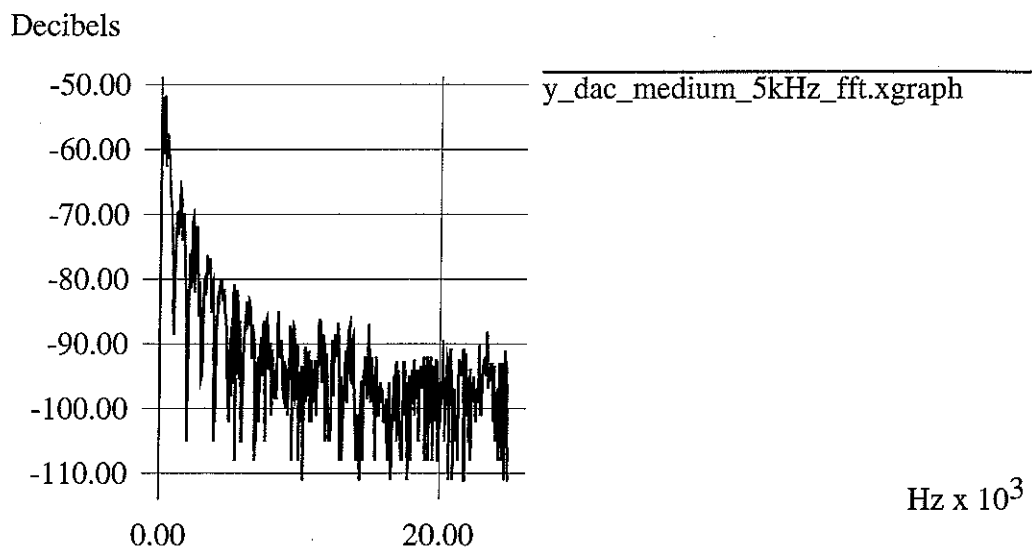
y_dac_medium:

This shows a zoomed-in view of `y_dac_long`. It captures two of the spikes, showing some of their internal structure. One of these spikes does have both positive and negative components in it. These spikes are 140mV and 260mV p-p, respectively. FFT shows spikes at around 300 and 500 Hz.

X Graph



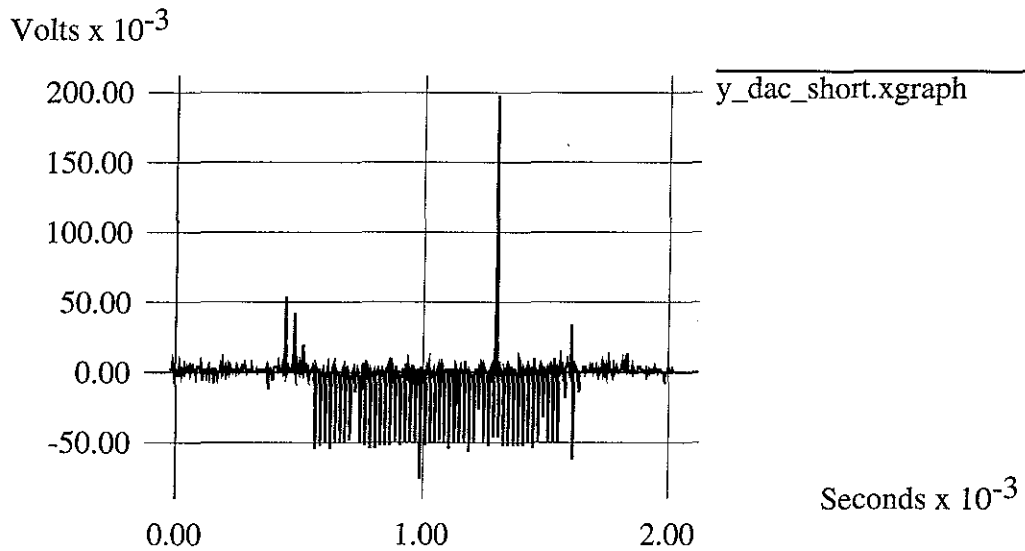
X Graph



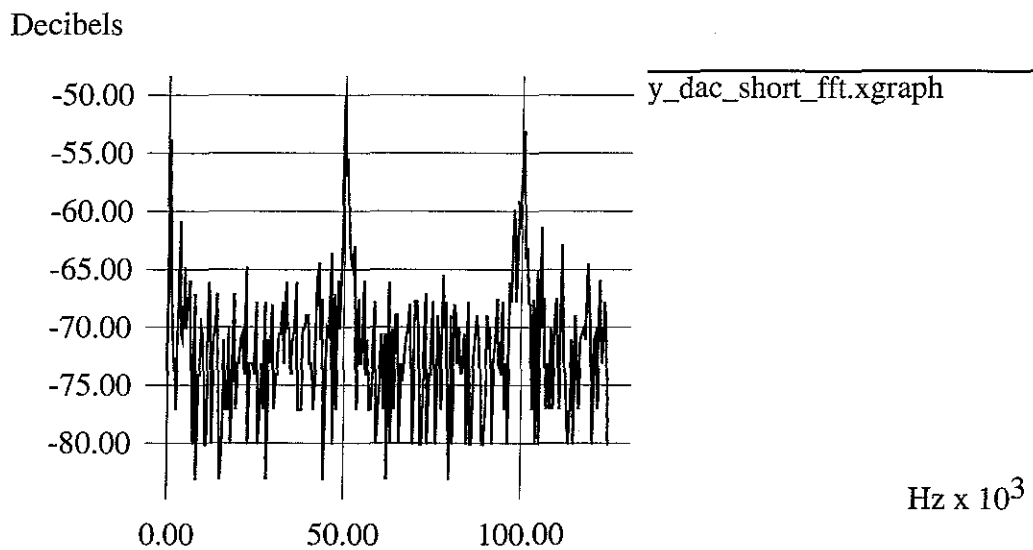
y_dac_medium_5kHz:

This shows the y_dac signal after a 5 kHz single-pole low-pass filter is applied to it. All of the structure present in the faster y signals has been removed, and the pulses are now down to about 10 mV peak-to-peak. The pulses are close to being square waves now, and the FFT reflects this. This shows that filtering is very effective in reducing the noise present on the lines, and this has been applied to the X and Y lines.

X Graph



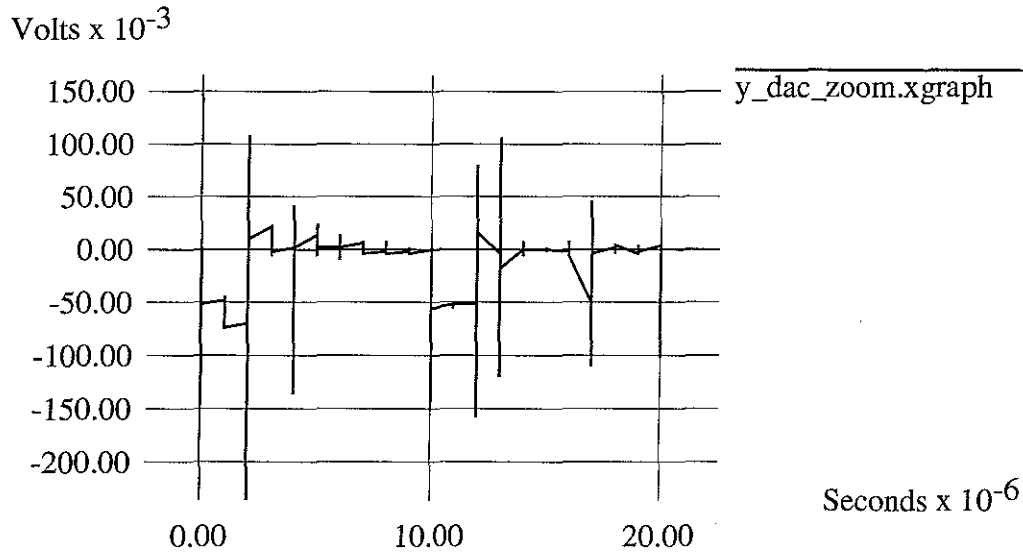
X Graph



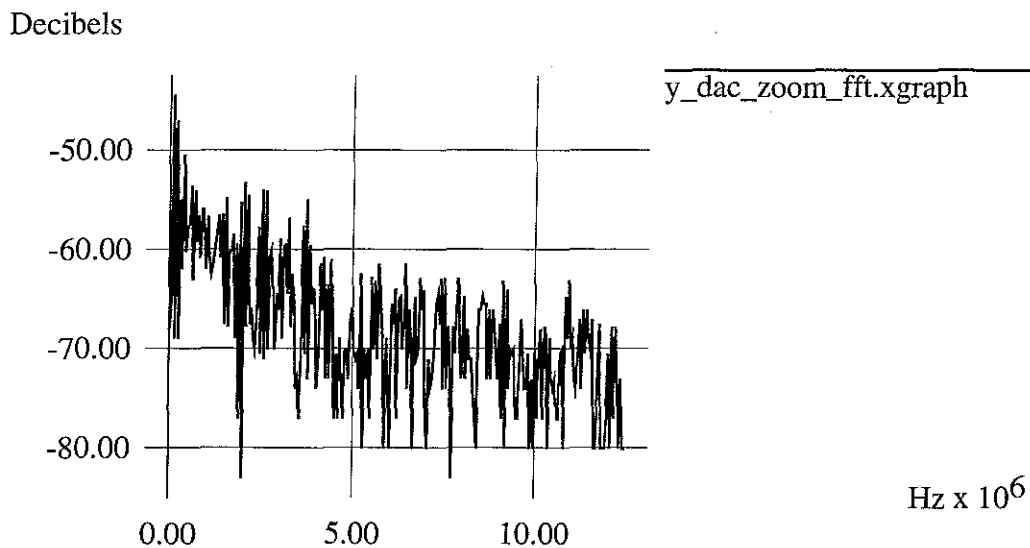
y_dac_short:

This view captures one glitch, showing yet more internal structure. This shows that the glitch is itself made up of many smaller glitches that are about 20 usec apart (freq of about 50 kHz, which matches the DAC switching rate set in the STM server). FFT shows clearly the 50 kHz spike and also a 100 kHz spike (harmonic?). It also shows a 500 Hz spike.

X Graph



X Graph



y_dac_zoom:

This view shows a zoom into two of the small glitches that make up the previous view. It shows that the spikes are complex and go both positive and negative. The short features in these spikes still go from ± 150 mV each. The changes are actually smooth, but the xgraph program truncates to microseconds, so they appear to be sharp jumps.

APPENDIX C — POETIC VIEWS OF THE SYSTEM

During the course of this project, we received two poems that were inspired by the system. The first came via electronic mail, and was inspired by the SIGGRAPH paper. The second was given to Warren Robinett after he gave a talk describing the system. We present them here:

Palabra Oscura

Three days after sinus surgery, first
night home alone, an imagefront blows through
every hour or two, asleep or not.

Four or five a.m. the word "boustrophedonic"
takes the stage with all the muscular assurance of a dream.
Literally, "ox-turn fashion", the way a furrow sweeps a field.

The "bous-", like Bossy, Spanish buey, bullish.
"-strophe-", near dawn, how the great shoulder
leans into the turn, the axial tilt of the opiated morning.

Meaning tracing back and forth in contrast to raster-scan,
rake-style, TV, musical staves, lines all begun at one margin --
a flash of shadows sketching afternoon across Arastradero Road.

A scanning tunneling electron microscope maps boustrophedonically,
its probe tilling atomic surfaces' fertile likelihoods; that fact
first read last summer holding places in a ticket line.

An honest track, not quite seismography, EEG,
EKG, but like that, no carriage return. With vital signs
it matters that pen never leave paper.

Awake behind gaping facial bone,
no one spoke, no one wrote, the word itself
turned out to have interiors, revealed its own poles,

some number of chambers, inner walls
adorned with pictures woven from unbroken thread,
light drawn through pinholes from lightsources.

Greg Keith

11/28/93

(printed with permission)

Sonnet VI

Atomic beach balls tossed beyond the bumps
Of solid golden strips will now design
A strange imaginary landscape. Clumps
Of particles appear to now entwine.

Electrons guided with some magic wands
Create a message to my covered eyes.
These nanonistic, small molec'lar bonds
Reveal to me all earthly matter ties.

I track and hover through the quarks of chance
Enthralled by how our measurements can rule
The future of all technical advance;
A strange discovery: a student's tool.

By thought and reason great ones had their turn.
Now, with my hands, some new things I shall learn.

Tammy Gordin

Copyright 1993
(printed with permission)

APPENDIX D — SOURCE CODE

The source code for this system is available on CD-ROM. Please contact the Department of Computer Science at the University of North Carolina at Chapel Hill for information on obtaining a copy. It is also available for anonymous FTP from `ftp.cs.unc.edu` in the `/pub/nanomanip/src` directory. It can also be found in AFS in the directory:

`/afs/cs.unc.edu/home/ftp/pub/nanomanip/src.`