FIRST LINE OF TITLE
SECOND LINE OF TITLE

Thèse n. 1234 2011
présenté le 12 Mars 2011
à la Faculté des Sciences de Base
laboratoire SuperScience
programme doctoral en SuperScience
École Polytechnique Fédérale de Lausanne

pour l'obtention du grade de Docteur ès Sciences
par

Paolino Paperino

acceptée sur proposition du jury:

Prof Name Surname, président du jury
Prof Name Surname, directeur de thèse
Prof Name Surname, rapporteur
Prof Name Surname, rapporteur
Prof Name Surname, rapporteur

Lausanne, EPFL, 2011

Wings are a constraint that makes
it possible to fly.
— Robert Bringhurst

To my parents…

Acknowledgements

Lausanne, 12 Mars 2011 D. K.

Preface

A preface is not mandatory. It would typically be written by some other person (eg your thesis director).
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lausanne, 12 Mars 2011 T. D.

Abstract

Contents

List of Figures

List of Tables

1 Introduction

Many methods exist to observe objects too small for the naked eye. Optical microscopy was one of the first developed technique (17th century) to see visible lights with a system of magnifying lenses. One of the advantage of optical microscopy is that you can record images at the video rate, but it still has a low resolution. Electron microscope can acquire data faster because of the speed of the electron beam, but the raster scan pattern is a serious hindrance(speed). Atomic Force Microscope are slower than electron microscopes mainly because of its bulky mechanics (inertia). Toshi Ando /INSERT REF developed small fast piezos that can reach /INSERT VALUE. All of those techniques display data on a grid. We will see how we can use non-raster scan pattern to improve the bandwidth on the XY-plane. Also, we have developed image processing algorithms to render non-gridded data. Finally, we will investigate new ways to go beyond the limits on z with tilt corrections and fast feedback on z.

Limitations

Microscopy: - optical -> video rate but crappy resolution - EM -> raster scan can slow electron beam fast - AFM -> slow mostly because of bulky mechanics: intertia (mass of the system), toshi ando small piezos (small range) but fast

Using spiral has been shown to be better(demands less BW x,y) but need display of non gridded data (lower res. freq. system)

z is still limiting to go on large scales (x,y)

2 Visualization of non-gridded data

The most common way to render gridded data is using raster scan patterns. This technique steers the tip of the AFM on specific points of the grid. Past AFM research has been focused on improving those position controllers. Indeed, the AFM precision depends from the quality of the closed loop feedback on XY. These improvements, however, didn't solve fundamental problems with raster scanning. Most of the data is thrown away (border) and the actual position on the XY plane is inaccurate - and directly correlated with the efficiency of the position controller.

First, we discuss using sensor data instead of theoretical one and its implications in non-raster scanning. Then, we investigate how to render images from sparse data with Delaunay triangulation and inpainting.

2.1 Sensor data

Current AFMs run in closed loop: position controllers on the XY axis are needed to steer the tip at the right position.

Instead of using this method, we will work in open loop and register the data of the position sensors. One of the advantage of using sensor data is that we don't need accurate position controllers: it has no impact on our data. Instead of position the tip on an exact position, we're embracing its inaccuracy. The precision of our system is limited by the sensors and not the feedback on XY. Our current setup (Asylum Research MFP3D) uses capacity sensors.

If we use sensor data, we don't acquire it on specific points on a grid. Therefore we need image processing algorithms to render missing parts of our scan. The easiest way is to have a linear gradient between the closest points. We will see in the next section how to find these points. We will link these points together with triangles. Another way is to use image processing algorithms.

2.2 Image rendering techniques

Current AFMs give discrete data about the cantilever's position; therefore, we'll need to use image processing algorithms to generate images.

2.2.1 Inpainting algorithms

Reconstructing missing parts of images was first developed for restoring photographs and paintings or remove undesirable data like text and publicity. The art of restoration was performed manually. Nowadays, tools like Photoshop or Gimp are widely used in the media. It can be used to produce special effects [Richard and Chang(2001)].

This process is called inpainting. The principle behind it is to fill a patch with its surroundings. Mathematicians have developed wide range of algorithms to solve that kind of problems. We will investigate a special case of partial differential equations (PDE): heat equations. The heat equation is a PDE that represents the distribution of heat in a region over time.

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0 \tag{2.1}$$

α is the thermal diffusivity - that is interpreted as a "thermal inertia modulus" - and u is the temperature over space and time (i.e. $u(x, y, z, t)$). A high thermal diffusivity implies that the heat moves rapidly.

The algorithm has been implemented by Travis Meyer on MATLAB. This algorithm will spread out the information of each point on missing parts of our grid.

[Aubert and Kornprobst(2006)] shows that heat equations are powerful to fill out these patches of missing data, but it smooths data on sharp edges(high frequency data). One of the effect is that edges are blurred out by the algorithm.

/INSERT inpaing example

2.2.2 OpenGL

In this section, we see how to render images with OpenGL(Open Graphics Library). It is an API (Application Programming Interface) developed by Silicon Graphics to hide the complexities of interfacing with different 3D accelerators and mainly used for 3D modeling in video games and simulations. OpenGL leverages the fact that GPUs are optimized to render triangle. Before drawing our data, the GPU will enter in the tessellation process. It will optimize the rendering by using the repetition of a geometric shape (triangle). The more complex the shape the harder it will be for the GPU to process it. If we already pre-process the data into triangles, we will minimize processing costs. [Abobe(2013)]

Triangulation with Delaunay

The first problems we'll face when trying to plot our cloud of points is to know how to generate triangles. Indeed, OpenGL can only render triangles from a triplet of points. An unordered list of points will not be ordered by OpenGL. We'll need algorithms like Delaunay triangulation to obtain those. The principle behind the Delaunay triangulation is to generate triangles from triplets of point.

The algorithm minimizes the angles of each triangle. The triangulation is successful if no vertex (i.e. 3-dimensional point) is in the interior of a triangle.

Triangle.c is a library developed in C by Jonathan Shewchuk [Shewchuk(1996)] to generate Delaunay triangulations. The Figure 2.1 shows the effect of the program on a spiral scan. We will discuss later about spiral scanning. The input data has 20'000 points and the program generates 38'784 triangles.

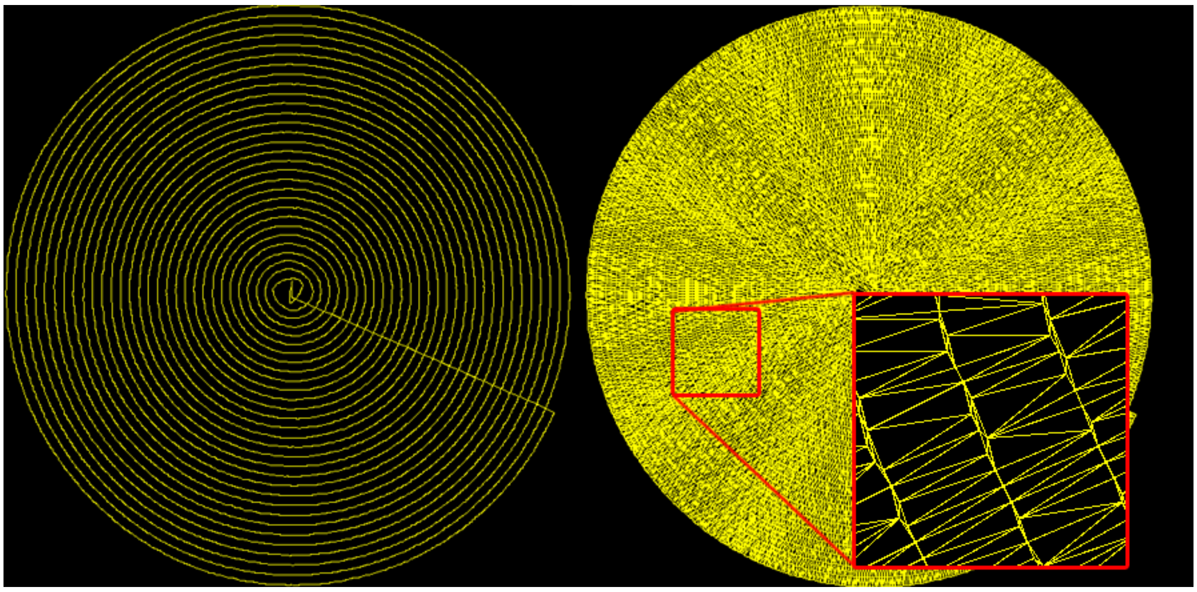
Figure 2.1: Delaunay triangulation on spiral scan and closeup

With this method we can connect the points on a 2D plane. In the figure 2.2 We use the z-axis data to compute the height of each of our point and render 3D models of our scans. To add colors to our data, OpenGL will create a linear gradient between each of the data points.

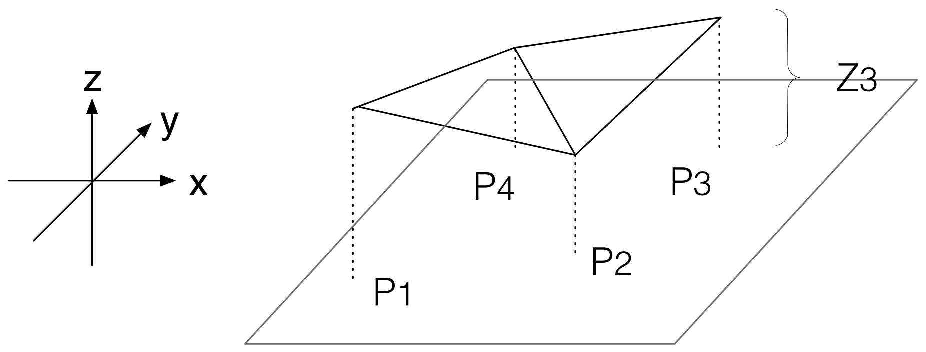

Figure 2.2: Delaunay triangulation: From 2D to 3D

Immediate mode vs VBO

We'll investigate two ways to render our surface with OpenGL: the immediate mode and vertex buffer objects.

The immediate mode is the simplest implementation of OpenGL. Indeed, we render every frame. If we rotate the our 3D model, we'll have to regenerate the latter. The power of the immediate mode is its simple implementation (no initialization and extra code). Moreover, it is easier to debug. For a small number of vertices (< 10'000) the immediate mode is appropriate. [Kilgard(2008)] states that the immediate mode is more convenient and less overhead than other implementations (Vertex Buffer Objects)

The following code is an example of the immediate mode for a simple triangle. The display function is called when GLUT(OpenGL Utility Toolkit) determines that the windows needs to be redisplayed. Action like rotation, translation or resizing of the model will trigger the display event. Each time the display function will be called, the program will upload the vertices to the GPU.

This implementation is easy to understand and debug. Unfortunately, it is not the most efficient one.

If we try to display a significant number of triangles (> 10'000 vertices), the CPU will be the bottle neck. The GPU doesn't start rendering data before glEnd. Thus, the CPU is spoon-feeding the GPU by transferring the data triangle by triangle. Moreover, the number of API calls is proportional to the number of triangles. I.e. if you have 10 triangles you will make (10*(2+3+3)) 80 API calls [OpenGL(2012)]. In conclusion, if you want to render less than 10'000 vertices, code a quick implementation and are not planning on making a lot of changes in your rendering, the immediate mode is the way to go.

One of the problem we have encountered with the immediate mode is the transfer from the system memory to the GPU. We've seen there is a bottle neck in the transfer. With 10'000 points we can only have 3 frames per seconds. It means that our computer takes 300ms to upload our data to the GPU.

Instead of transferring the data from the memory to the GPU, the GPU could read the memory of the program. Buffer objects have been created to allow the GPU to have access to the memory. The process of reading the memory from the GPU is called Direct Memory Access (DMA). A buffer object is a contiguous untyped memory which the CPU and the GPU have access to.

We can't just upload our data into the memory without any structure. We need to map the data and make it readable for the GPU. We store our data in a vertex array object.

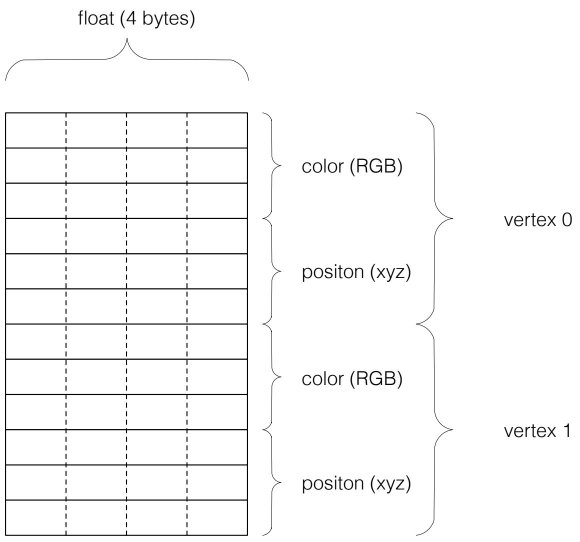

Figure 2.3: Memory for the VAO

After having allocated and created this chunk of data, we need to map it to make it readable to the GPU. OpenGL has API calls for that application.

The advantage of this implementation is that you directly pull your data to a shared memory between the CPU and the GPU. Your CPU will spend less cycles making API calls thus improving the performances of the program. The power of the VBOs is that you just need to upload your data and your display function will just bind the VBO. Our performances have improved from 3FPS to 130FPS for 100'000 data points. Having a high FPS makes the animations smoother.

Table 2.1 show the non-linearity of our implementation. We see that Delaunay triangulation doesn't scale well for 1'000'000 points. In AFM scans we will rarely sample 1'000'000 datapoints. The limits of our AFM is 100'000 kHz. If we take 10 seconds scans at the limit rate, we observe that the computation time is still way below the scanning time.

Table 2.1: Rendering results[ms]

Nb of points	Delaunay	VBO
1000	2.9	23.9
10000	8.1	27
100000	66.9	181
1000000	640.7	267

3 Techniques for fast z feedback

3.1 Double PID

3.2 Tilt compensation

If the probe/sample angle is not perpendicular, we observe a tilt on the surface. This tilt is problematic when it becomes larger than the features. Flattening algorithms restore the image and put the data on the same level. This technique works if the range of the tip is large enough. We have decided to take another approach and to dynamically compensate for the tilt.

First, we scan the surface of the sample with a circle pattern. It gives us informations about the general topography of the surface. Then, we compute the plane equation of the surface by applying a fit in Igor Pro.

$$z = a_1 x + a_2 y + a_3 \tag{3.1}$$

Igor Pro finds the coefficients by minimizing the values of Chi-Square. We generate the waves to send to the controller the following way.

$$wavetosend = a_1 wavex + a_2 wavey \tag{3.2}$$

3.2.1 Experiment

We took a calibration sample to test the efficiency of our method.

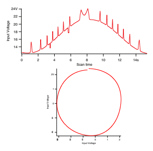

Figure 3.1: Output of the tilt pattern

When we scanned the surface on our plane we had the following coefficients for the plane.

Table 3.1: Planefit coefficients

a_1	a_2	a_3
-0.14615	-0.031882	29.537

The size of the scan is 30 um and the spiral has 80 loops. The scan pattern we are going to send to the controller is generated with the previously computed planefit coefficients.

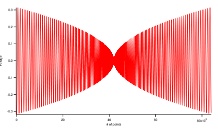

Figure 3.2: Input of the tilt compensation

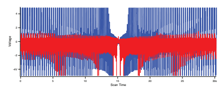

Figure 3.3: Output of the fast piezo

The tilt compensation will take a load off the small fast piezoelectrical ceramics. The Figure 3.3 shows the efficiency of our method. Indeed, the fast piezo was previously saturating. The piezos was trying to reach features that are larger than his range. If we use the tilt correction, we see that our piezo has no problem reaching those features.

4 Results

4.1 Calcite experiment

In this experiment we've observed the dissolution of calcite ($CaCO_3$) with water. This experiment is interesting

4.2 Soft sample

A An appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Bibliography

[Abobe(2013)] Abobe. Tips for optimizing gpu rendering performance. 2013. URL http://help.adobe.com/en_US/as3/mobile/WS5d37564e2b3bb78e5247b9e212ea639b4d7-8000.html.

[Aubert and Kornprobst(2006)] Gilles Aubert and Pierre Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer, 2006.

[Kilgard(2008)] Mark J. Kilgard. Modern opengl usage: Using vertex buffer objects well. Technical report, 2008.

[OpenGL(2012)] OpenGL, 2012. URL http://www.opengl.org/wiki/Legacy_OpenGL.

[Richard and Chang(2001)] Manuel M Oliveira Brian Bowen Richard and McKenna Yu-Sung Chang. Fast digital image inpainting. In *Appeared in the Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain*, 2001.

[Shewchuk(1996)] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.