AFM Raw Data File Processing

CS 110-Intermediate Programming Java Project

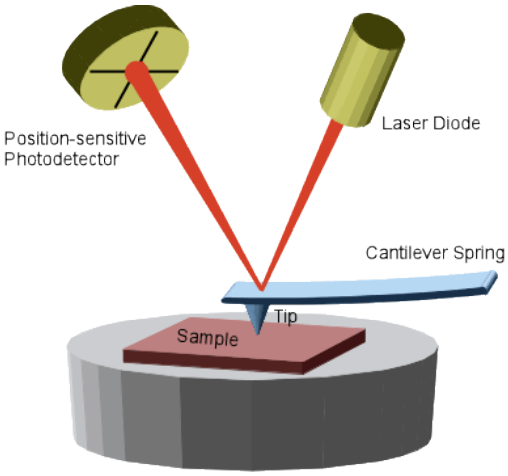

Figure 1 AFM Sketch

- ***What is AFM(Atomic Force Microscope)?***

AFM is a very high-resolution type of scanning probe microscopy, with demonstrated resolution on the order of fractions of a nanometer, more than 1000 times better than the optical diffraction limit.

AFM is typically used to scan the specimen surface. We use it as a tool to measure the force between the sample and the tip. As the tip approaches the sample close enough, the so called Van der Waals force takes effect, which will bend the cantilever a little bit, causing the optical length changes, so then we can get the force by measuring the optical length.

- ***What are the files like?***

They are generated by the AFM with "*.000", "*.001", "*.002" as their extensions (not the actual extension), and they can't be opened by any known software. If you try to open them using notepad or Office Word or whatever, there are always full of garbled characters. They're only readable by one software that is installed on an obsolete computer originally connected to the AFM, which runs DOS! And because it is a really crude software, we can only do simple operations on the data, one by one, manually.

- ***Problems to solve:***

1. Because there are tens of thousands of AFM files to read and process, we don't want to open them by hand one by one, we want to do batch-processing automatically.
2. How to extract raw data from the files.
3. Merging the data in an excel file.
4. Detecting the significant data points.

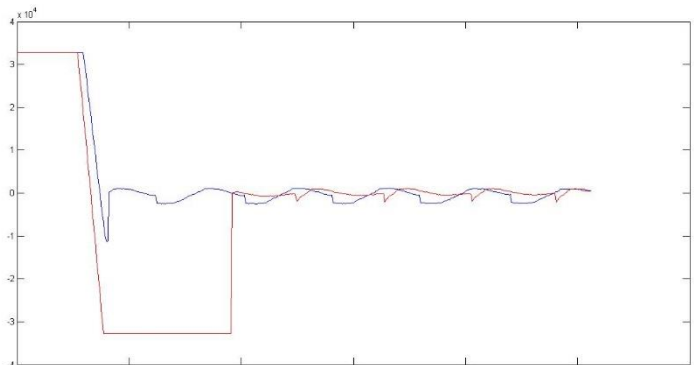

Figure 2 data plot

- ***Ways to deal with the problems:***

1. Use a for loop to detect valid file names and read in ten files as a group.
2. I used a binary viewer to open the file and found out that the file is made up of two parts:

The first part is the description of the data, e.g. date and time, scan size, scan rate etc. This part is encoded using ASCII.

The second part is the actual numeric data, which is encoded using 8-bit unsigned integer, Little Endian.

There are a bunch of zeroes between the two parts to make the file exactly 10 KB (10240 Bytes) .

So I read in the file as binary stream, skipped the ASCII and zeroes part, then transfer the UInt-8 hexadecimal number to decimal number, which is the data we want.

3. I used an API called POI, a convenient API providing all kinds of operations on Microsoft Office documents. I used POI to save the *.xls files, label certain cells and filling the background with colors.
4. It took me a long time to refine the algorithm, because the data is a little noisy and not so certain, my algorithm has to be robust enough to detect the right position. The final algorithm I used is first find the position with the largest differentiation, then look for the local extreme value around it. By this we can successfully detect the desired locations, we have tested dozens of groups of files and no error has been reported.