AFM/CLSM Data Visualization and Comparison Using An Open-Source Toolkit

BARTEK RAJWA,[1,2] HELEN A. MCNALLY,[3] PADMA VARADHARAJAN,[1] JENNIFER STURGIS,[1] AND J. PAUL ROBINSON[1,2]*

[1]*Purdue University Cytometry Laboratories, School of Veterinary Medicine, Purdue University, West Lafayette, Indiana 47907*
[2]*Department of Biomedical Engineering, College of Engineering, Purdue University, West Lafayette, Indiana 47907*
[3]*Center for Paralysis Research, School of Veterinary Medicine, Purdue University, West Lafayette, Indiana 47907*

KEY WORDS visualization; atomic force microscopy; confocal light scanning microscopy; neuronal cells; open-source software

ABSTRACT There is a vast difference in the traditional presentation of AFM data and confocal data. AFM data are presented as surface contours while confocal data are usually visualized using either surface- or volume-rendering techniques. Finding a common meaningful visualization platform is not an easy task. AFM and CLSM technologies are complementary and are more frequently being used to image common biological systems. In order to provide a presentation method that would assist us in evaluating cellular morphology, we propose a simple visualization strategy that is comparative, intuitive, and operates within an open-source environment of ImageJ, SurfaceJ, and VolumeJ applications. In order to find some common ground for AFM-CLSM image comparison, we have developed a plug-in for ImageJ, which allows us to import proprietary image data sets into this application. We propose to represent both AFM and CLSM image data sets as shaded elevation maps with color-coded height. This simple technique utilizes the open source VolumeJ and SurfaceJ plug-ins. To provide an example of this visualization technique, we evaluated the three-dimensional architecture of living chick dorsal root ganglia and sympathetic ganglia measured independently with AFM and CLSM. *Microsc. Res. Tech. 64:176–184, 2004.*

INTRODUCTION

The use of atomic force microscopy (AFM) to image living biological materials in their native environments with molecular or submolecular resolution is an area of great interest to the biological and medical communities (Morris et al., 2001). On the other hand, confocal microscopy (CLSM) has been used by biologists for many years to map biological pathways and understand intracellular mechanisms as well as to view the overall architectures of living cells (Pawley, 1995; Matsumoto, 2002). Some reports show implementations of combined AFM and confocal instruments for biological applications (Henderson and Sakaguchi, 1993; Putman et al., 1993; Schabert et al., 1994; Vesenka et al., 1995). For the purpose of our research on live neural cells, we have been utilizing stand-alone AFM and confocal systems separately but imaging cells under similar conditions. In order to confirm the AFM analysis of the living neurons, we had to combine the CLSM with AFM-based investigations, and visualize the results in a common format. AFM and CLSM instrument manufacturers offer some visualization capabilities in their standard software packages; however, dramatically different image collection modes of these technologies make any image comparison a very difficult task. AFM operating in the tapping mode measures topography by tapping the surface with an oscillating probe tip, so that the tip makes contact with the sample for only a short duration in each oscillation cycle. In contrast to atomic force microscopy, confocal microscopy is a method based on traditional far-field optics. Detectors in CLSM collect photons emitted by fluorescent labels introduced into the biological sample. The instrument utilizes the optical pathway of a regular optical microscope. Owing to the presence of a confocal aperture that stops the fluorescence signal from out-of-focus optical planes, this technique is capable of collecting three-dimensional (3-D) images. It is very important to recognize that the concept of 3-D imaging in confocal systems differs greatly from what is understood as 3-D in the AFM world. CLSM using fluorescence mode can collect emissions originating in the interior of a biological sample. This means that a confocal microscope can record a 3-D array of numbers representing an intensity of fluorescence from all the scanned voxels within the analyzed volume of the sample. In their simplest form, confocal data can be shown as a series of individual sections, but usually the data are presented using volumetric display techniques (Fig. 1). The structure of AFM data is simpler: an image is represented by a 2-D array of numbers, where values correspond to deflections of the AFM cantilever caused by the probe-sample interactions. Such a data structure is often referred to as 2.5-D. Visualization of AFM data is usually attained by the creation of simple surface displays or elevation maps (Fig. 1). Even though

*Correspondence to: J. Paul Robinson, Purdue University Cytometry Laboratories, Hansen Research Building B050, 201 S. University Street, West Lafayette, IN 47907. E-mail: jpr@flowcyt.cyto.purdue.edu

Received 15 April 2004; accepted in revised form 3 May 2004

Contract grant sponsor: CPR general funds; Contract grant sponsor: State of Indiana; Contract grant sponsor: Office of the Provost of Purdue University; Contract grant sponsor: Purdue University Cytometry Laboratories.

DOI 10.1002/jemt.20067

Published online in Wiley InterScience (www.interscience.wiley.com).

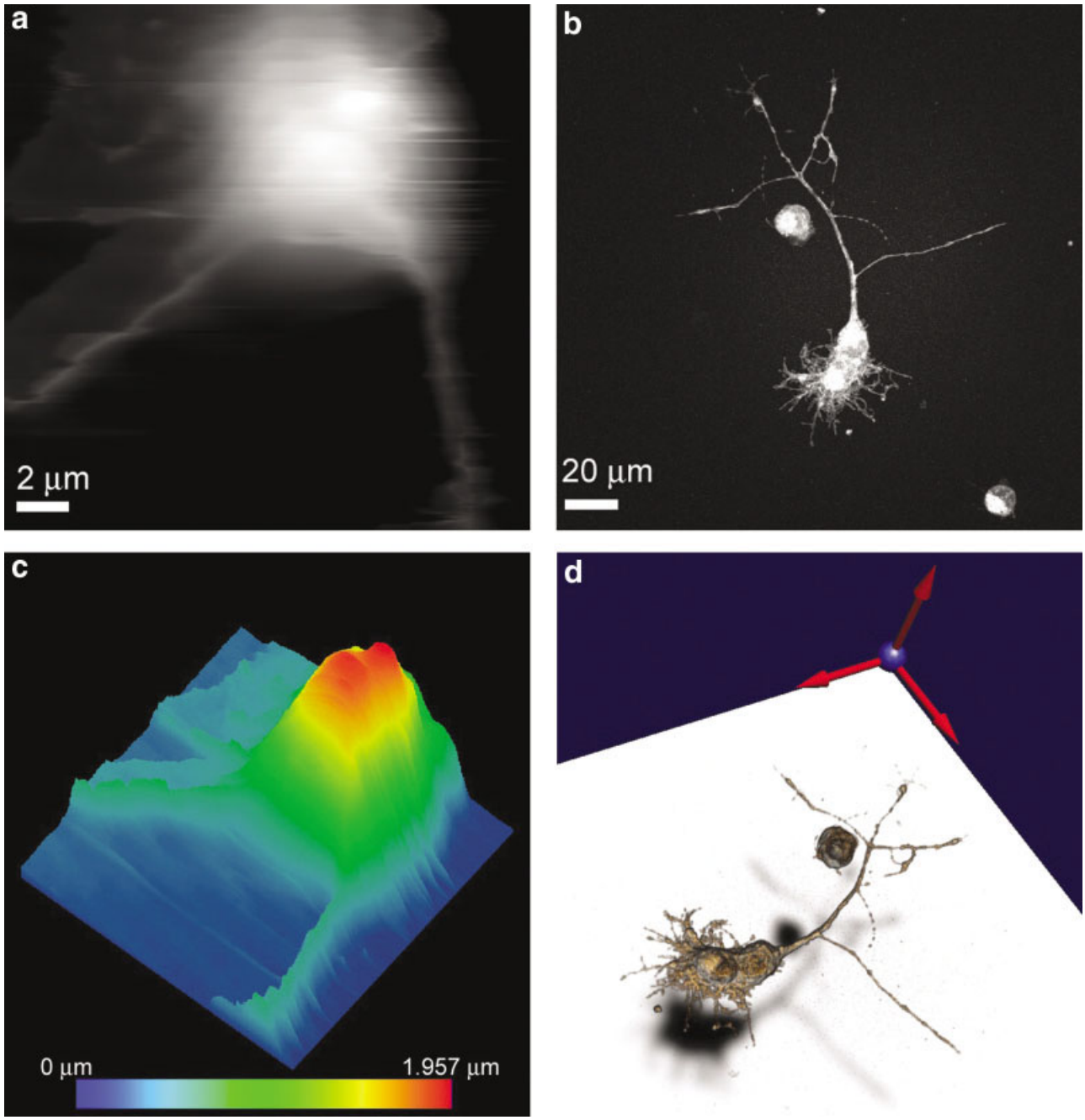

Fig. 1. Examples of visualization of AFM and CLSM data. **a:** AFM range image (gray scale proportional to the elevation); **b:** CLSM maximum intensity projection image; **c:** AFM pseudo-colored isometric view; **d:** CLSM volumetric imaging. The AFM images show the cell body of a live neuronal cell, while CLSM images show the whole live neuron stained with FM 1-43.

this visualization technique gives the impression of three-dimensionality, it is not a fully 3-D representation.

The purpose of this report is to describe a simple, routine method for AFM-CLSM image visualization and comparison, which represents both AFM and CLSM data in a similar and intuitive manner (Fig. 2). Our proposed system is based on an open-source software package, which includes ImageJ by W. Rasband (developed at the U.S. National Institutes of Health and available on the Internet at http://rsb.info.nih.gov/nih-image/), VolumeJ/SurfaceJ plug-ins by M. Abramoff (Abramoff and Viergever, 2002), and OpenAFM/AFM-FileInfo[1] plug-ins prepared by us. ImageJ is a general-purpose image-processing program written in Java and based on the

[1]The plug-ins can be downloaded from http://www.cyto.purdue.edu/afm.

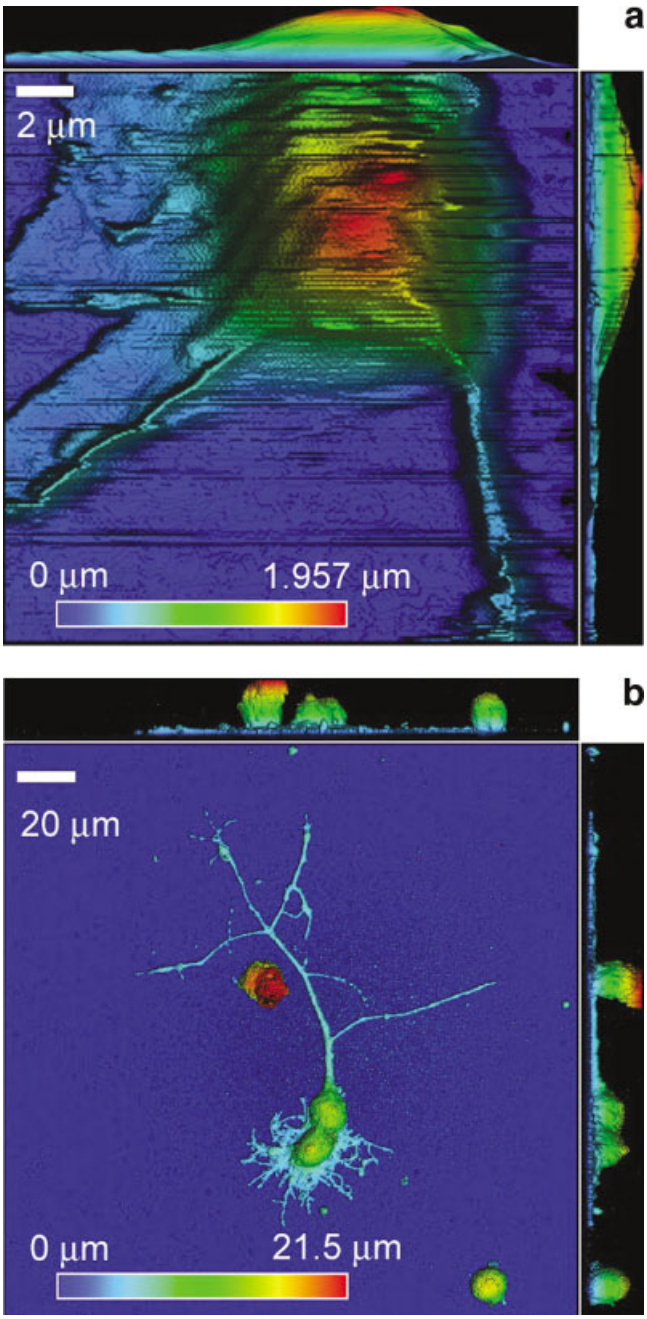

Fig. 2. Datasets from Figure 1 imported into the ImageJ environment and displayed as pseudo-colored height maps. The images are shaded to communicate elevations in an easy-to-interpret manner. **a:** AFM image; **b:** CLSM image.

well-known NIH Image software. ImageJ can run on any platform with a Java Virtual Machine (including Apple Macintosh, MS Windows, and various flavors of Unix), and it allows extensibility by addition of Java plug-ins.

MATERIALS AND METHODS

Sample Preparation

Dorsal root ganglia (DRG) and sympathetic ganglia were dissected from 7–8-day-old chick embryos by conventional methods (Mahanthappa and Patterson, 1998). Individual neurons were obtained from the tissue using tituration, enzymatic digestion (trypsin in Puck's medium), and differential centrifugation. Cell suspensions were moved to drilled 35-mm Petri dishes with a glass coverslip attached to the bottom, and cell density within these samples was monitored using a hemocytometer. For the purposes of this study, we attempted to obtain primary cultures on the order of 40,000 cells/35-mm Petri dish (1 cell/100 μm^2). The cells were plated on a substrate of polyornithine and laminin, and maintained at 37°C in 5% CO_2. Healthy DRG neurons on laminin, for example, will attach to the substrate and begin to form process within 24 hours. A conventional 2% neuron growth medium (Higgins and Banker, 1998) containing nerve growth factor (NGF), vitamin C, Insulin (Sigma Chemical Co., St. Louis, MO; no. I-6634), and penicillin/streptomycin (Sigma Chemical Co., no. P-0906) was used in these studies. The base medium was prepared from an F-12 nutrient mixture (Gibco, Gaithersburg, MD; no. 21700-075), supplemented with the other adjuncts including conalbumin (Sigma, no. C-0880) and horse serum (Gibco, no. 26050-088) to a final pH of ~7.4, and refrigerated until used.

Confocal Imaging

In order to visualize morphology of the cells, a fluorescent liphophilic tracer N-(3-triethylammoniumpropyl)-4-(4-(dibutylamino)styryl)pyridinium dibromide (Molecular Probes, Eugene, OR) was used to stain the cultures. This dye, known as FM 1-43, is water-soluble, is nontoxic to cells, and remains nonfluorescent in aqueous medium. It is believed to insert into the outer leaflet of the cell membrane, where it becomes intensely fluorescent (Ryan et al., 1997; Ryan, 2001).

Confocal laser scanning microscopy was performed with a Bio-Rad Radiance 2100 Rainbow instrument (Hemel Hempstead, UK) based on an inverted Nikon Eclipse TE2000 microscope (Nikon, Tokyo, Japan). The confocal system was equipped with a 60× PlanApo 1.4-NA oil-immersion objective lens, an air-cooled 100-mW argon laser, three fluorescence detection channels (photomultipliers), and a nonconfocal transmitted light detector. Blue laser light attenuated to 3.5% of the maximum power was introduced into the sample. One of the photomultipliers was used to collect fluorescence signals from the green and yellow regions of the fluorescence emission, and the nonconfocal transmitted light detector was used to collect brightfield images. Fluorescence signals from the FM 1-43 probe passed through a 560-nm dichroic long-pass filter, a 500-nm laser blocking filter, and a 570-nm long pass filter before being detected by a photomultiplier.

AFM Imaging

Temperature, humidity, and CO_2 level were not controlled during AFM imaging; thus, experiments never exceeded 4 hours, at which time the cells were still healthy and well attached to the substrate. Cells were imaged in tapping mode (Hansma, 1994) using a Digital Instruments Dimension 3100 AFM. The fluid cell (Digital Instruments, no. DTFML) was fitted with a silicon nitride tip (Digital Instruments, no. DNP –20). The V-shaped cantilever was 200 μm in length, yield-

ing a nominal spring constant of 0.32 N/m. The tip was square pyramidal in shape with a nominal tip radius of curvature of 20–60 nm. A course approach to the surface was performed until the fluid cell entered the culture medium. At that time, the alignment was adjusted and the resonance frequency determined (6–8 kHz) for maximum tip oscillation. A drive amplitude of 400–600 mV was applied to obtain a free amplitude of 0.5V RMS. The image fields ($\approx$10 μm^2) were obtained at 0.4–0.6 Hz, requiring 2–10 minutes to complete a raster scan of the entire sample. Thus, small area images comprising neurites or growth cone usually required 2 minutes prior to the next period of imaging, while large cells could take 5 to 10 minutes to complete. It is important to note that some features of the anatomy that may change within this time constant would not be a portion of the composite image. Both height and amplitude data were used to image the surface topography.

RESULTS

A plug-in for opening and processing proprietary AFM files was developed within the ImageJ environment. ImageJ was chosen for its platform-independence, openness, simplicity, and portability. ImageJ is also the fastest pure Java image-processing program currently available. The program has built-in command recorder, editor, and Java compiler; therefore, it is easily extensible through custom plug-ins. Initially inspired by NIH Image for Macintosh, it is now an advanced, multi-threaded application allowing parallel processing of time-consuming image-processing tasks.

In order to build a toolkit for AFM file processing, it is essential to understand the format of AFM Digital Instruments files. Every AFM file has an ASCII header, with parameters that follow the Digital Instruments CIAO (Control Input And Output) format. To find the actual topography of a measured object, a so-called soft-scale and hard-scale have to be extracted from the header. Raw data have to be read from the remaining part of the data file and multiplied by the "hard-scale" to provide so-called hard values. The prefix "hard-" is used to indicate that these numbers are typically defined by the hardware itself and are not changeable by the user.

The hard value is the analog representation of the measurement and is given in volts. The hard-scale is the conversion factor used to convert the raw data into the hard values. The soft values are the representation of height. The soft-scale is a user-defined calibration number used to convert a hard value to a soft value. To calculate the real height of the surface measured by AFM, data must be multiplied by both the hard-scale and the soft-scale:

$$\text{raw data} * \text{hard-scale [V]} = \text{hard value [V]}$$

$$\text{hard value [V]} * \text{soft-scale [nm/V]} = \text{soft value [nm]}$$

Our OpenAFM plug-in reads the raw data and calculates the actual elevation measured by the AFM instrument. The OpenAFM is a public class that extends the ImageJ interface, recognizing and importing AFM files. It functions as a rudimentary parser, analyzing the ASCII header for the required information. This is required owing to the variable length of the header and the absence of predefined allocation of certain bytes of the header for certain data. The OpenAFMFile method takes the directory and filename as input parameters and returns an ImageJ object. The relevant details are set in a FileInfo object, which is embedded in the returned ImageJ object. The raw data are composed of signed two-byte numbers. The method converts the raw data into heights by multiplying each number by the product of the hard-scale and soft-scale. OpenAFM also provides an option to rescale the image so that the minimum height is set to zero.

AFM-FileInfo plug-in implements PluginFilter, to capture relevant details from the header of the file and to display the header information to the user. The required data are stored as properties in the ImageJ object by OpenAFM when the initial parsing is complete. Parameters captured from the header include filename, filetype, height and width, header length, soft-scale, hard-scale, minimum, maximum, maximum after rescaling, bytes per pixel, and so on.

After the AFM data were imported, the SurfaceJ plug-in (by M. Abramoff) was used to create shaded elevation maps with color-coded height. By rotating the map 90° along the x and y axes, xz and yz elevation maps were also prepared. The result of AFM image visualization created with ImageJ, OpenAFM, and SurfaceJ plug-ins is shown in Figures 2a and 3a,c.

Confocal data stored in proprietary Bio-Rad PIC format were imported into ImageJ. We reversed the order of confocal data stacks and then visualized them using the VolumeJ plug-in (by M. Abramoff). The plug-in operated in ray-tracing mode using pre-defined look-up tables to code height. Using a method similar to the AFM case, two 90° rotations along the x and y axes were calculated to provide xz and yz views. The result is presented in Figures 2b and 3b,d.

DISCUSSION

After importing AFM and confocal data into the ImageJ application via a custom-made OpenAFM plug-in (see Appendix), we represented both AFM and CLSM datasets as elevation maps with color-coded height using VolumeJ and SurfaceJ plug-ins (Abramoff and Viergever, 2002). The VolumeJ and SurfaceJ plug-ins have a unique capability of visualizing both volume and surface data in an almost identical manner. This is achieved by internal conversion of surface data to an intermediate volume representation. By providing a simple visualization technique that is independent of data collection modality and is available via open-source software ImageJ/SurfaceJ/VolumeJ/OpenAFM combination, we have demonstrated an easy-to-use, simple toolkit for CLSM/AFM data comparison.

The resolution limits of the AFM and CLSM systems are very different. The resolution of far-field light microscopes is restricted by the diffraction limits of the microscope objective. AFM instruments can resolve sub-nanometer structures because their resolution is governed by different principles and is limited by the smallest resolvable vertical displacement of the tip.

Both confocal and AFM images revealed similar shapes and overall architectures of the soma and growth cones of live neurons. However, the compara-

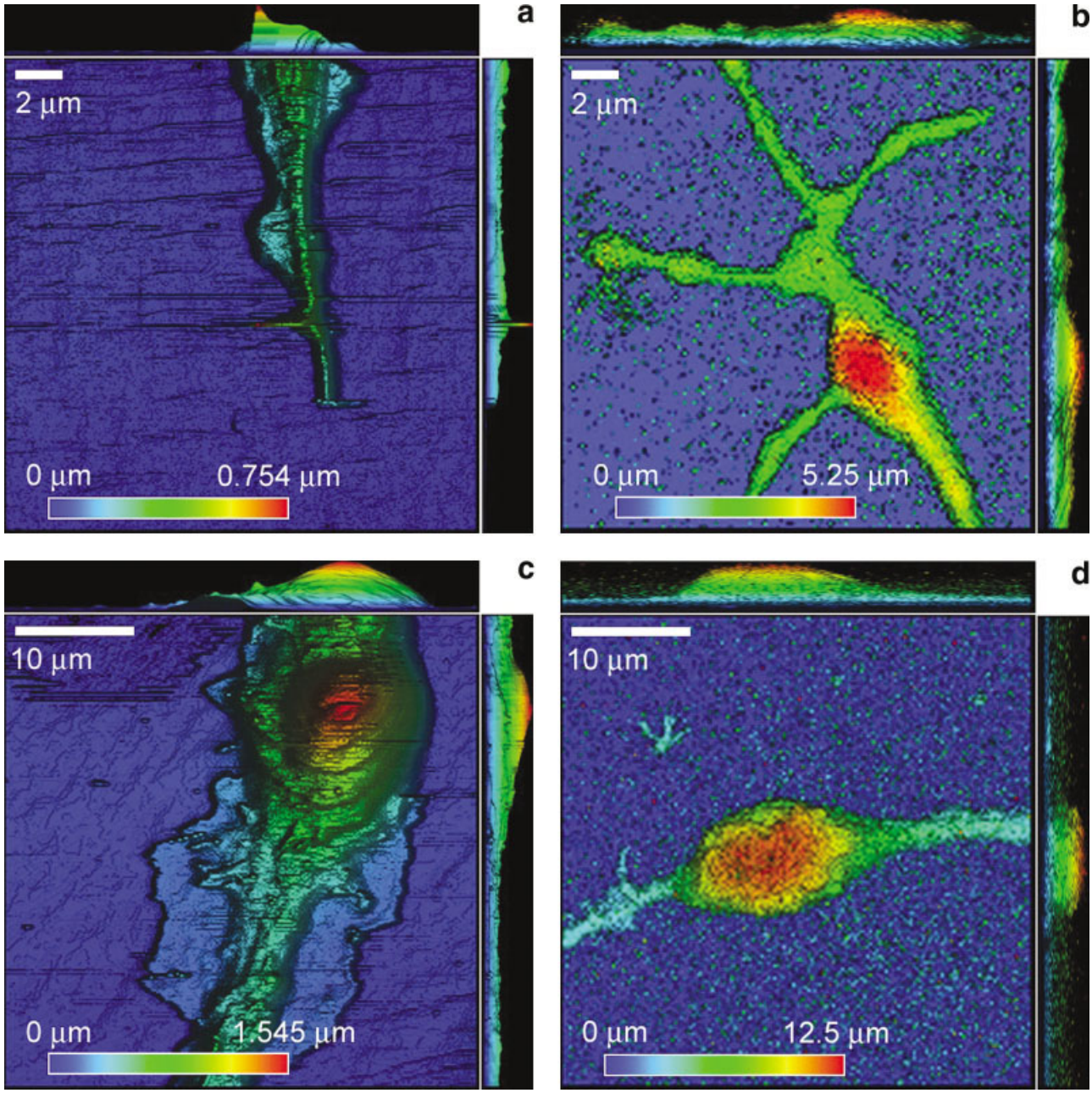

Fig. 3. **a:** A typical AFM height image of a DRG growth cone with its associated color bar designating height; **b:** CMLS image of a similar cell area; **c:** AFM height-map of a typical DRG cell body and extending neurites; **d:** CLSM visualization of a similar region in a cell. The confocal and AFM data are on two similar but not identical cells, as the imaging of the same cell by both instruments has not been possible in our system.

tively low z-resolution of a far-field confocal system does not permit accurate evaluation of the height, thus explaining the discrepancy between height data from the AFM and confocal systems. The z-resolution of CLSM for any given wavelength is always at least 2 times worse than the corresponding xy-resolution. Therefore, height values calculated from 3D images acquired by CLSM are much less precise than xy-measurements obtained from the same datasets. In contrast, in AFM instruments z-resolution is determined by detection of extremely small (less than a fraction of a nanometer) vertical movements of the tip.

AFM and CLSM technologies can be used to evaluate the three-dimensional architecture of living neurons. Shaded, color-coded orthogonal representations of AFM and CLSM data allow easier comparisons of the results obtained with these two techniques than the standard visualization methods typically used with these two different imaging modalities (Fig. 1). The fact that all the packages used for visualization are open

source enhances the value of this simple presentation method.

ACKNOWLEDGMENTS

We gratefully acknowledge the technical assistance of Ms. Judy Grimmer, cell culture lab, Center for Paralysis Research (CPR), Purdue University. We also thank Professor Michael Melloch, Purdue University, for access to the AFM used in this study. We are grateful to Peter Lombrozzo for detailed information about Digital Instruments AFM data format.

REFERENCES

Abramoff MD, Viergever MA. 2002. Computation and visualization of three-dimensional soft tissue motion in the orbit. IEEE Trans Med Imag 21:296–304.

Hansma PK. 1994. Tapping mode atomic force microscopy in liquids. Appl Phys Lett 64:1738–1740.

Henderson E, Sakaguchi DS. 1993. Imaging F-actin in fixed glial cells with a combined optical fluorescence/atomic force microscope. Neuroimage 1:145–150.

Higgins D, Banker G. 1998. Primary dissociated cell cultures. In: Banker G, Goslin K, editors. Culturing nerve cells. Cambridge, MA: The MIT Press. p 53–57.

Mahanthappa NK, Patterson PH. 1998. Culturing mammalian sympathoadrenal derivatives. In: Banker G, Goslin K, editors. Culturing nerve cells. Cambridge, MA: The MIT Press. p 289–307.

Matsumoto B. 2002. Cell biological applications of confocal microscopy. San Diego: Academic Press.

Morris VJ, Kirby AR, Gunning AP. 2001. Atomic force microscopy for biologist. London: Imperial College Press.

Pawley JB. 1995. Handbook of biological confocal microscopy. New York: Plenum Press.

Putman CAJ, van Leeuwen AM, De Grooth BG, Radosevic K, van der Werf K, Van Hulst NF, Grieve J. 1993. Atomic force microscopy combined with confocal laser scanning microscopy: a new look at cells. Bioimaging 1:63–70.

Ryan TA. 2001. Presynaptic imaging techniques. Curr Opin Neurobiol 11:544–549.

Ryan TA, Reuter H, Smith SJ. 1997. Optical detection of a quantal presynaptic membrane turnover. Nature 388:478–482.

Schabert F, Knapp H, Karrasch S, Haring R, Engel A. 1994. Confocal scanning laser-scanning probe hybrid microscope for biological applications. Ultramicroscopy 53:147–157.

Vesenka J, Mosher C, Schaus S, Ambrosio L, Henderson E. 1995. Combining optical and atomic force microscopy for life sciences research. BioTechniques 19:240–248, 849, 852.

APPENDIX

The Java code of OpenAFM plug-in allowing import of Digital Instruments AFM format into ImageJ application

```
/*
   Copyright © 2004 Purdue University Cytometry Laboratories

   This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published
   by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

   This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
*/
import ij.*;
import ij.io.*;
import ij.gui.*;
import ij.util.Tools;
import ij.process.*;
import ij.plugin.*;
import java.io.*;
import java.util.*;
import ij.measure.*;
import java.awt.image.*;
import java.lang.String;
import ij.plugin.filter.Info;

public class OpenAFM_implements Plugin {
   public static double hs=0.0,ss=0.0;
   public static int imageOffset = 0;
   public static String width=null,height=null;
   public static ImagePlus openAFMFile(String directory, String file) {
      ImagePlus imp = null;
      File f = null;
      BufferedReader fread = null;
      String temp = null;
      String hardScale = null;
      String hardValue = null;
      String softScale = null;
      String dataOffset = null;
      int count = 0;
      FileInfo fi = null;
      String pixelSize = null;
      try
      {
         f = new File(directory,file);
         fread = new BufferedReader(new FileReader(directory+"/"+file));

         temp=fread.readLine();
         temp = temp.substring(1);
      }
      catch(Exception ex)
      {
                                          IJ.showMessage("Open AFM. . .","Exception caused in block 1 − "+ex);
      }
      try
      {
         StringTokenizer info = new StringTokenizer(temp,"");
         String param = info.nextToken();
         param = param.substring(1);
         while(param.equals(""File"))
         {
            if(info.hasMoreTokens())
            {
               String nextToken = info.nextToken();
               //IJ.showMessage("Open AFM . . .","param is"+param+" and nextToken is "+nextToken);
               if(param.equalsIgnoreCase("@2:Z"))
               {
                  if(nextToken.equalsIgnoreCase("scale:"))
                  {
                     param = info.nextToken();
                     param = info.nextToken();
                     param = info.nextToken();
                     hardScale = info.nextToken();
                     hardScale = hardScale.substring(1);
                     param = info.nextToken();
                     hardValue = info.nextToken();
                     count++;
                     //IJ.showMessage("Open AFM. . .","Value of hardscale is"+hardScale);
```

APPENDIX. (continued)

```
            }
        }
        else
        if(param.equalsIgnoreCase("Bytes/Pixel:"))
        {
            pixelSize = nextToken;
        }
        else
        if(param.equalsIgnoreCase("@Sens."))
        {
            if(nextToken.equalsIgnoreCase("Zscan:"))
            {
                param = info.nextToken();
                softScale = info.nextToken();
                count++;
                //IJ.showMessage("Open AFM. . .","Value of softscale is "+softScale);
            }
        }
        else
        if(param.equalsIgnoreCase("Data"))
        {
            if(nextToken.equalsIgnoreCase("offset:"))
            {
                dataOffset = info.nextToken();
                count++;
                //IJ.showMessage("Open AFM. . .","Data offset is"+dataOffset);
            }
        }
        else
        if(param.equalsIgnoreCase("Samps/line:"))
        {
            width = nextToken;
            //IJ.showMessage("Open AFM. . .","Value of width is "+width);
        }
        else
        if(param.equalsIgnoreCase("Lines:"))
        {
            height = nextToken;
            //IJ.showMessage("Open AFM. . .","Value of height is"+height);
        }
    }
    temp = fread.readLine();
    info = new StringTokenizer(temp," ");
    param = info.nextToken();
    param = param.substring(1);
  }
}
catch(Exception e)
{
  IJ.showMessage("Open AFM. . .","Exception caused in block 2 - "+e);
}
try
{
  hs = (double)(new Double(hardScale)).doubleValue();
  ss = (double)(new Double(softScale)).doubleValue();
  imageOffset = Integer.parselnt(dataOffset);
}
catch(Exception exc)
{
  IJ.showMessage("Open AFM. . .","Exception caused in block 3"+exc);
}
try
{
  fi = new FileInfo();
  fi.directory = directory;
  fi.fileFormat = fi.RAW;
  fi.fileName = file;
  fi.fileType = FileInfo.GRAY16_SIGNED;
  fi.gapBetweenImages = 0;
  fi.height = Integer.parseint(height);
  fi.intelByteOrder = true;
  fi.offset = imageOffset;
  fi.width = Integer.parseint(width);
  fi.description = "Height: "+height+" Width: "+width;
  fi.info = "HardScale:"+hardScale+" SoftScale: "+softScale;
  fi.nImages = 1;

  FileOpener fo = new FileOpener(fi);
  imp = fo.open(false);
```

APPENDIX. (continued)

```
        imp.setFileInfo(fi);
        imp.setProperty("HardScale",hardScale);
        imp.setProperty("SoftScale",softScale);
        imp.setProperty("PixelSize",pixelSize);
      }
      catch(Exception exx)
      {
        IJ.showMessage("Open AFM. . .","Exception caused in block 5"+exx);
      }
      return imp;
  }
  public void run(String arg) (
      OpenDialog od = new OpenDialog("Open AFM. . .", arg);
      String file = od.getFileName();
      If (file == null) return;
      String directory = od.getDirectory();
      ImagePlus imp = openAFMFile(directory, file);
      if (imp = null ) {
        imp.show();
        //IJ.run("Raw. . .","open="+directory+"/"+file+"image='16-bit Signed' width="+width+"height= "+height+"
  offset="+imageOffset+" number=1 gap=0 little-
  endian"):
        IJ.run("Multiply. . .", "value="+(hs"ss));
        GenericDialog gd = new GenericDiaglog("Height Settings");
        gd.addCheckbox("Yes",true);
        gd.showDialog();
        boolean ans = gd.getNextBoolean();
        if(ans)
        {
            IJ.run("Measure");
            ij.measure.Results Table rt = ij.measure.Results Table.getResultsTable();
            double value = rt.getValue("Min",0);
            double value2 = rt.getValue("Max",0);
            imp.setProperty("Minimum",new Double(value));
            imp.setProperty("Maximum",new Double(value2));
            IJ.run("Subtract,. . .","value="+value);
            IJ.run("Clear Results");
        }
        //IJ.showMessage("Open AFM. . .","Value of min is"+value);
        IJ.showStatus("");
      }else{
        IJ.showMessage("Open AFM. . .","Failed.");
      }
  }
}
```