TrakEM2: an ImageJ-based program for morphological data mining and 3D modeling

Albert Cardona[a,b]

[a]Molecular Cell and Developmental Biology, UCLA, 621 Charles E. Young Dr. South, Los Angeles 90015, USA;
[b]Institute of Neuroinformatics, University of Zurich/ETH, 190 Winterthurerstrasse, Zurich, Switzerland

ABSTRACT

The modern morphologist relies extensively on computer-aided image acquisition and analysis. While most of the desired functionality exists in separate software packages, the latter don't cross-talk well, and further, given the high volume of data a lab can generate in a very short time interval, keeping track of data and metadata becomes nearly impossible. Here we present our design of an ImageJ-based program aiming at providing an integral solution to morphological data mining. In particular, our program aims at storing, displaying and analyzing digital images, while aiding in extracting both three-dimensional and relational models of the sample. Furthermore, the program frees the researcher from tedious data-management tasks, and provides the means to deploy the data remotely over the internet. We have explored possible user interfaces with a prototype program, and found that a design directing the researcher from a self-provided abstract object ontology not only results in the most consistent data acquisition and valuable aid in modeling, but also self-constructs a rapid, light-weight means of accessing data through hierarchical trees and graphs.

Keywords: imaging, morphological data mining, three-dimensional modeling, data abstraction

1. INTRODUCTION

What have data storage, image adjusting, data extraction and 3D visualization in common? They are different aspects of imaging with which the modern morphologist deals on a daily basis. While many software packages exist that deal with a subset of the problem, there is no software package that provides an integral solution.

The current situation is far from convenient, and a better software environment for modeling is possible. First of all, existing applications are limited in scope, and do not cross-talk easily with each other. Second, images are unlinked to the data from which they are taken. Third, the different transformations applied to a particular image are unrecorded; the individual images lack a history listing that could explain to a third person how the image came to have its current appearance and why. Fourth, there is no standardized way of extracting data from the image -measurements, highlights, annotations- and storing such data along the image that supports it. Fifth, besides confocal stacks, which are seriated in nature and provide registration for free, images belonging to different regions in the 3D space of the sample are ignorant of the multiple relationships with each other in terms of the data they hold.

While the above problems are almost negligible and minor inconveniences when dealing with very small datasets, nowadays labs -and our lab in particular- are generating hundreds of gigabytes in a weekly basis. The inexistence of a software package that addresses the problems outlined above does not only result in a waste of time, but also precludes us from getting the most from imaging techniques.

Despite, we researchers are committed to meet our goals regardless. In particular, here at the Institute of Neuroinformatics (University of Zurich / ETH), we pursue an understanding of mammalian brain cortex and aim at emulating its computational abilities. In our strive to understand function by studying and modelling cortical structure, we are undertaking detailed anatomical analysis of the brain. The particular needs of the project span

Further author information: (Send correspondence to A.C)
A.C.: E-mail: acardona@ini.phys.ethz.ch

from acquiring a vast amount of images in the electron microscope, which have to be montaged in 2D and 3D, to identifying features such as cell membranes and synapses in the images, leading to the extraction of both 3D models and neuronal network connectivity maps, and of course to classify, quantify and measure synapses, vesicles, neurite branches; in short all the relevant neuronal elements. An extra leap in complexity is added by samples processed through electron microscopy being selected first from light microscopy sections. Currently, the project is tackled with a combination of custom and commercial software packages including Amira (Amiravis), Blender (Blender Foundation), ImageJ,[1] Matlab (Math Works), Neurolucida (MBF Bioscience), Photoshop (Adobe), Traka[2] and TrakEM.[3]

Our goal is to create a software package that not only provides all the desired functionality, but also remains extensible and thus applicable to any kind of microscopy-approachable study. We find that ImageJ, as a public domain java-oriented imaging application that can be easily deployed over the internet and has a built-in plug in structure, constitutes the perfect core of the program.

2. THE PROGRAM: WHAT WE WANT

What we want is the Moon, as a spoiled child would boldly point out, but short of that, we want the most convenient morphological data mining environment possible. Our program has to be enable the researcher to handle primary digital data, i.e. images, in an organized manner, so that no data loss occurs and no painful and time-consuming tasks are required, that could be otherwise automated. Further, the extracted data must be exportable and compliant with current standards for 3D modeling (X3D, DXF) and neuronal network simulations (NeuroML).

2.1. Data storage and retrieval

Image files are currently stored in the local hard drive, where imaging applications access them directly. While such a setup is convenient and easy when dealing with a small collection of images, the system does not scale up properly. Hard drives do not have an infinite amount of space available. Backing up large collections of images is tedious and time-consuming, although necessary. Both issues, storing large collections and backing them up, are solved when one considers a remote file storage. All backup and space matters are delegated to server maintainers, whose sole purpose is to excel in the task.

While many strategies exist, we believe that a database-oriented strategy is best; in particular, the open source, PostgreSQL relational database. In the first place, PostgreSQL is mature, reliable, and fine-tuned for heavy-duty usage. Second, connectors exist already for several languages, including java. Third, communications with the database are done over TCP/IP, meaning, accessibility becomes a problem of bandwidth, and with proper settings, any computer plugged to the internet may access the database. Bandwidth is certainly not a problem in a local, properly setup network. Furthermore, not all data needs to be retrieved at once from the database; on the contrary, the opposite is true: there is no need to load what the user is not interested in. For example, there is no need to load a 6 gigabyte, multilayer Photoshop image file to edit or analyze but only a tiny portion of it. Photoshop can't do any better than to require the file to be loaded in its entirety. To overcome the latter limitation is certainly within our goals.

Finally, a remote storage enables a straightforward remote deployment of data: one can not only work remotely, and perhaps simultaneously, on the same project, but also present it -perhaps in a read-only mode- to other researchers through the internet. That all the delivery is handled by the database itself frees us from the task of building up a server-side program.

2.2. A multiscale, visual representation of the sample space

The central point of digital imaging is to visualize a sample in all possible detail. Theoretically speaking, a multi-dimensional space enclosing the sample can be defined. Such space includes the two dimensions of a sectioning plane -be it ultrathin section, optical section, et cetera-; the third dimension in stacking up the different planes of section; the temporal axis if present. Furthermore, a subset of any section may have been further processed; for example, a ROI in a light microscopy section may be scooped out, and ultrathin sections obtained for further imaging in the electron microscope, most likely with a different orientation of the plane of section.

All the images obtained from the sample, no matter what level, should be accessible in a orderly and relational way. We can envisage a scale-independent display that enables both the introduction of differently scaled images and their relational navigation. For example, when browsing a stack of light microscopy images, one may want to zoom into a particular section that contains, in a different orientation, a further electron microscope set of images that show further detail of a neurite. It is a requirement -i.e. greatly facilities data extraction- that all scales of detail are accessed within the same display, and that lower resolution images have flags at particular areas indicating a further, higher resolution set of images.

2.3. Links between data

Images by themselves are ignorant of each other. Unless any two images share a partial overlap, the positioning of one image relative to another is completely up to the observer, who, with a priori knowledge of the sample at stake, is able to identify structures that continue from one image to the other. Thus, in drawing, be it manually or by semiautomated means, a ROI or outline that spans over several images, the researcher is making several statements. First, the images involved support the abstract data; second, the images share a relationship that has to be maintained. Thus any future transformation -scaling, rotation, z-positioning- on any of the images or the abstract data, has to be propagated to all the involved data elements to preserve consistency.

Furthermore, any abstract data such as the outline of a neuron implies a statement towards the existence of such a neuron. Thus, any further outlines in nearby sections have to be properly linked together, in order to define, as a whole, the soma, neurite and every other structure of the neuron. By consistently directing the user into properly classifying all pieces of abstract data, restricting the line of action, the whole tissue is reconstructed. Needless to say, any transformations applied to such higher level data need to be propagated downward, ultimately to the images themselves.

2.4. Navigating the abstracted data: trees and graphs

Casting a look over a few dozen images is not such a difficult task. When the data base spans to hundreds and thousands, the task is completely different, and under the obvious time constraints, the researcher needs to focus on the 'why' question that leads to browsing the 'what'. Further, the purpose of browsing several images has more to do with the metadata than the images themselves: which images are not processed yet? Which abstract objects are half finished? Which images support a particular abstract object, such as a three-dimensional model of a neuronal cell? For the purpose, a hierarchical display of the different involved elements may prove most useful.

Within java, trees have been supported since very early versions. As an extension, JGraph has been designed on top of java trees, and provide the means to represent hierarchical and nested data on the screen. Rapid browsing of tones of data becomes a reasonable task by visualizing only what is relevant at each level of abstraction, digging down the layers until reaching the images. Thus, a user navigating the data may ask for a three-dimensional representation of a neuron, one of its synapses, or the whole neuronal tissue, by querying the nodes that represent the object of interest. In a similar fashion, the images that support a given node, or object, and any related abstract data, can be swiftly displayed without any unnecessary overhead.

2.5. Interacting with a three-dimensional model

Serial section reconstruction leads to the generation of three-dimensional models. Such models provide information on the relative position of all the elements in space. For the amount of work involved, current modeling applications provide little more than pretty pictures. We believe three-dimensional models need not be static, if only they could be integrated into the data-aware application that generated them. A display presenting objects floating in space in relative positions one to another not only results in an intuitive interface to data exploration, but can also be used for interaction. Thus, measurement of volumes, lengths, average positions of boutons over dendrites or the images that support any particular volume, may be queried from within the three-dimensional model.

The three-dimensional models need not be the ultimate ray-traced scene. For the latter, exporting options to standard formats such as X3D are planned.

2.6. Where ImageJ comes in

ImageJ's design enables its usage as an extremely powerful library toolbox, which we intend to take advantage of to the fullest extent.

Input/Output. ImageJ enables the importing and exporting of almost every image data format out there. When confronting rare image formats, plugins can be and have been written to handle them.

Image processing and analysis. Images need not be displayed within ImageJ's GUI to be adjusted, processed or visualized. An all-purpose java.awt.Image can be wrapped in, and extracted from, an ImagePlus instance. Such an ImagePlus can, in turn, be set as the active image in ImageJ's window manager. In this fashion, any existing ImageJ plugin can be applied to any image, and the results collected and returned to the main application. Processing any image by any of the built-in filters and analysis tools, or those in plugins, is thus very straightforward.

Prevent reinvention of the wheel. Several scientific-grade, freely available plugins exist already which can, and will, be included in the project. From input/output to automated image stitching (the TurboReg plugin,[4]), and particle analysis, color quantification, edge detection, image segmentation, et cetera.

Convenient plugin development. The plugin architecture provides a convenient development work flow. There's no need to recompile and relink the whole project, but only the plugins that extend ImageJ's functionality, from within a running ImageJ's instance.

Remote deployment and visualization. The built-in networking functionality of the Java library, plus the ability to run ImageJ as an applet, provide the means to consistently present image data in remote locations through any web browser.

2.7. Open source as a design strategy

The ultimate extendability is not only provided by a proper API and object structure, but also by the ability to read the code and pass it on. Here at the Institute of Neuroinformatics we are committed to open source development, in the understanding that such a strategy empowers our projects through sharing and interaction with the whole scientific community.

The project presented here, although biased towards neuroscience, aims at providing a configurable setup for the handling of any ontology. We design this application to fall within the scope of every field involving imaging. Its open source condition will facilitate its extension to related imaging applications.

3. A FIRST PROTOTYPE

We recognize the project as a challenging endeavor that, like Rome, will not be build up in one day. For testing purposes, we explored ImageJ upside down, and also tried out different user models for abstract data generation.

3.1. Data storage

As explained above, data storage is most conveniently handled remotely. Using internal ImageJ commands and the JDBC connector to a PostgreSQL database, zipped TIFF images can be stored in the database along their metadata and abstracted data. Storing the images with lossless compression reduces the size and thus the transfer time, while preserving all other attributes.

3.2. A tentative display for the sample space: extending the *ImageCanvas*

The very first functionality tested was the ability to display multiple images within a single window, while being able to interact with and transform each one individually. The core ImageJ's GUI classes excel in the purpose. In particular the *ij.gui.ImageCanvas* class provides a convenient drawing surface with built-in zooming and panning abilities. On top of them, it becomes almost trivial to display any number of images, ROIs, outlines and labels, without ever worrying about implementation details relative to display navigation. With such a tool, a layer representing a section in the tissue is displayed, and within it, any selected image is amenable to adjustment, processing and analysis.

A more challenging problem was posed by the stacking of several layers. While ImageJ defaults to evenly spaced slices in stacks, this solution lacks the desired flexibility. For the purpose, we built a Z axis representation by means of drawing thick horizontal lines on an image, where each line represented a layer which can lay on any Z axis coordinate.

While the representation of the sample space in stacked two-dimensional layers comes most natural when browsing and processing images, and follows up logically from the image acquistion techniques, we are exploring the possibilities of freeing the design from such a constraint. The convenience and usability of displaying images and abstract objects in a multi-dimensional space (primarily three-dimensional plus time) has yet to be explored.

3.3. An object ontology as a restrictive guide to modeling

As opposed to let the user create profiles (outlines of sectioned objects) at will, and then relay on his full attention to order them properly, we find the opposite approach most convenient and effective. First, a hierarchical schema of the possible objects to be found in the sample is predefined in an XML file. Such file is given to the project as a master ontology that defines the object templates, from which all project objects are created. Then, constrained by the given ontology, the user must create first the high-level objects, which do not hold any profiles per se, and then proceed to create its children, until reaching the level at which objects can contain a list of profiles. Finally, the profile itself is created and embeded into the particular's project hierarchy of objects, and the user is directed to the display to draw it, either manually or by semi-automated means.

In the fashion described above, the relationships within abstracted objects in the sample is seamlessly generated along with the abstract objects themselves. Now, any of such objects can be queried for its properties and the relationship with one another, either within the same hierarchical level or as parent-child. For example, a neuronal network contains neurons and synapses. Each neuron consists of a soma, a dendritic tree and an axonal tree. The soma itself is defined by a list of profiles that delimit the membrane, while at the same time the soma contains a nucleus, vesicles, and other objects, which in turn are defined by a list of profiles. As a result, the three-dimensional volume is obtained almost for free and can be rendered or exported, while generating an intuitive tree of objects to be used as an interface for data navigation, inspection and analysis. Furthermore, the relationship between all objects can be extracted with any desired level of detail, enabling for example the exporting of a NeuroML file containing the relationships, through synapses, between all neurons in a network. With proper extensions, any sort of quantitative question can be asked to the objects through a contextual menu.

3.4. What we learned from the prototype

Relieving the user from the hassle of taking proper care and keeping track of the raw data, the images, not only saves time but also ensures its backup and it's proper retrieval long periods of time afterwards. The digital lifespan of the images is thus expanded. On the other hand, a dedicated facility for system administration is an obvious requirement.

The preservation of the original images becomes a must. While storing any derived image, the original image must be preserved for restoring purposes, case the derivative becomes unuseful, or simply to check on whether any artifacts have been introduced.

The appropriate balance between freedom to edit and total restriction of movements must be achieved. The realization that the links between data elements are fundamental, and that they are mostly created as a secondary effect of user actions, provided us with an explorable usability framework that so far is proving its effectivity. By designing the program to direct the user into fulfilling the task, while propagating any changes between any related elements of the project, we believe a reasonable balance is achieved.

4. CONCLUSION

There are several unresolved problems in imaging. While the related problems of segmentation, edge detection and image stitching are getting most of the attention, the problems of data organization, storage and display are barely tackled at all. With the program presented here, we would like to deliver a solution to the latter set of imaging problems, while providing the best known solutions to the former.

ACKNOWLEDGMENTS

The author thanks Rodney Douglas and Kevin Martin for their seed idea, support and encouragement, and Volker Hartenstein for his vision and expertise in Drosophila biology, and of course for a fantastic postdoctoral research position in Drosophila neurobiology. The author is indebted to Stephan Nufer for his suggestions and proof-reading of the manuscript, and looking forward to team up and develop fantastic tools for microscopy.

REFERENCES

1. W. Rasband, "ImageJ. http://rsb.info.nih.gov/ij/," 1997-2006.
2. D. Botha, R. Douglas, and K. Martin, "Traka: a microcomputer assisted system for digitizing the three-dimensional structure of neurones," 1987.
3. B. Ahmed, J. Anderson, K. Martin, and J. Nelson, "Map of the synapes onto layer 4 basket cells of the primary visual cortex of the cat," 1997.
4. P. Thévenaz, U. Ruttimann, and M. Unser, "A pyramid approach to subpixel registration based on intensity," *IEEE Transactions on Image Processing* **7**, pp. 27–41, January 1998.