Open Tools for Storage and Management of Quantitative Image Data

Joshua Moore, Chris Allan, Jean-Marie Burel, Brian Loranger, Donald MacDonald, Jonathan Monk, and Jason R. Swedlow

Division of Gene Regulation and Expression
College of Life Sciences
Wellcome Trust Biocentre
University of Dundee, Dundee
Scotland DD1 5EH
United Kingdom

Abstract

The explosion in quantitative imaging has driven the need to develop tools for storing, managing, analyzing, and viewing large sets of data. In this chapter, we discuss tools we have built for storing large data sets for the lifetime of a typical
research project. As part of the Open Microscopy Environment (OME) Consor-
tium, we have built a series of open-source tools that support the manipulation and
visualization of large sets of complex image data. Images from a number of
proprietary file formats can be imported and then accessed from a single server
running in a laboratory or imaging facility. We discuss the capabilities of the OME
Server, a Perl-based data management system that is designed for large-scale
analysis of image data using a web browser-based user interface. In addition, we
have recently released a lighter weight Java-based OME Remote Objects Server
that supports remote applications for managing and viewing image data. Together
these systems provide a suite of tools for large-scale quantitative imaging that is
now commonly used throughout cell and developmental biology.

I. Introduction

Quantitative digital imaging assays have now become universal in cell and
developmental biology and are one of the fundamental techniques for the growing
field of systems biology. As new imaging methods extend the resolution of spatial,
temporal, and spectral dimensions and applications expand into large-scale
genome-wide and chemical library screens, there is a growing requirement for
tools that support the storage, archiving, analysis, visualization, and querying of
large multidimensional sets of data. The new age of quantitative biology requires
the informatics tools to provide methods of accessing the quantitative data at its
foundation (Swedlow et al., 2006).

In some cases, the success of the genomics projects provides a good template for
the management of quantitative imaging data. Central repositories for shared data
sets using defined formats for data transfer have worked extremely well for projects
where a single reference set of data is to be shared by a large community (e.g.,
db=Genome). Indeed, similar image-based resources that are intended to be
community-wide resources are now being released (Christiansen et al., 2006; Lein
et al., 2007). However, experimental, hypothesis-driven cell and developmental
biology also make increasing use of multidimensional imaging, yet it is often not
clear initially which primary data represents a “reference” data set that will be used
for a publication or what combination of data is appropriate for sharing. To
complicate matters, the range and complexity of data collected for a single
project—the links between original and processed images, analytic results, experi-
mental descriptions, and final summarized results are only maintained manually by
experimental biologists (usually dedicated but itinerant Ph.D. and postdoctoral
students). The fact that such data is often physically stored on-line on CD or DVD
media (with limited life spans) only adds to the difficulty of the problem.
In this chapter, we discuss methods we have built and used for physically storing quantitative image data in a safe, secure, and accessible method, and then discuss the much harder problem of building software tools for managing, visualizing, and querying image.

II. Secure, Archived and Available Storage for Biological Image Data

Many of the debates about image data management center around standardized file formats and image databases. However, in most cases, even if these tools were in place, most laboratories and imaging facilities do not have the means to store the volume of data generated by their microscopes in a manageable and affordable way. The requirements for such a storage system are not simple to satisfy. Data must be stored securely and available for ~2–5 years, the lifetime of most projects. Image data files are often multiple gigabytes, so the system must be able to store and deliver these very large files. The entire system including the physical media, the data servers, networking, and data transfer facilities must be easily expandable to meet the needs of new experiments. In addition, most microscopy facilities will require nightly backups so that reasonable disaster recovery is possible.

At the Wellcome Trust Biocentre, University of Dundee, we have installed a Storage Area Network (SAN) based on a hierarchical file system composed of pools of high-performance disk arrays and redundant tape libraries for data archiving and backup. The layout of the system is shown in Fig. 1. The overall cost of the system is minimized by extensive use of Serial ATA drives within fiber channel enclosures. The disk arrays are connected by fiber channel to Linux x86 servers that provide access and manage all the data in the system, ensuring 25% of the available disk space is free at all times for storage of new data. Additional servers provide connections to two tape robots that keep redundant, physically distinct copies of the whole file system as backups. As new data are stored on the SAN system, older, less accessed data are moved through the storage hierarchy to the one of the tape libraries. This not only ensures that all data from a project is stored and available but also ensures that the most accessed data is kept available for user access. This hierarchical strategy is cost efficient and eliminates the need for users to archive their own data using CDs and DVDs. The SAN supports access from any computer in the center, regardless of location or operating system. In addition, the installation of a high-speed redundant network allows data to be recorded directly onto the SAN without any compromise in data acquisition performance. Perhaps most importantly, the installation of a truly enterprise solution ensures that expansion of the storage capacity of the system is easily accomplished by simply buying more physical storage media.

In our experience, this type of storage system that is normally found in commercial data centers is critical for multidimensional imaging and quantitative analysis.
Fig. 1 Simple architecture diagram for the SAN used for image storage at Dundee. A hierarchical file system, run using Tivoli Storage Manager installed on two IBM pSeries servers, manages all backup and archiving. All data is stored live either on the fiber channel or on the SATA disk systems, and then automatically migrated to tape in archive form if not accessed for a significant period. Data is accessed from user workstations and desktops via a high-performance network and the SAN is connected to a compute cluster for computationally demanding processing.
Moreover, once installed, this system can be used to support other data intensive activities in a research center like proteomics and bioinformatics.

### III. The Open Microscopy Environment: Data Management Tools for Biological Research

The Open Microscopy Environment (OME) is a multisite software development consortium ([http://openmicroscopy.org/about/](http://openmicroscopy.org/about/)) dedicated to building open, freely available software tools for biological image data management. Originally founded in 2000 by Peter Sorger, Jason Swedlow, and Ilya Goldberg, the project now has grown to include over 20 developers and designers in the United States and the United Kingdom ([http://openmicroscopy.org/about/](http://openmicroscopy.org/about/)). The project is attempting to provide a complete set of specifications and tools for biological imaging research, which is a significant task. As detailed on the project website ([http://openmicroscopy.org](http://openmicroscopy.org)), OME is releasing file format specifications, software tools to support these specifications, and software tools for data management. All tools are open and available and the project maintains mailing lists ([http://openmicroscopy.org/getting-involved/](http://openmicroscopy.org/getting-involved/)) to ensure that it can capture feedback from users and external developers.

In the past six years, the OME Consortium has released a series of software tools, specifications, and documents:

- A white paper summarizing the goals and the concepts of the project ([Swedlow et al., 2003](http://openmicroscopy.org/about/));
- A specification for a metadata format, OME-XML, that can be used for sharing image data and metadata between laboratories and collaborators ([Goldberg et al., 2005](http://openmicroscopy.org/about/));
- A software tool, Bio-Formats, that converts over 40 proprietary file formats into OME-TIFF ([http://www.loci.wisc.edu/ome/formats.html](http://www.loci.wisc.edu/ome/formats.html));
- A package of software tools, OME Server (versions 2.0–2.6), that provide data management and analysis tools for biological imaging ([Johnston et al., 2006; Schiffmann et al., 2006; Swedlow et al., 2003](http://www.loci.wisc.edu/ome/formats.html)). The OME Server (latest version is 2.6.0 as of this writing) is available at [http://cvs.openmicroscopy.org.uk](http://cvs.openmicroscopy.org.uk); and
- A new Java-based server that supports remote client applications providing a range of image visualization and management tools.
The interested reader can consult online and published resources for all aspects of the OME project. In this chapter, we focus briefly on the OME Server and then describe in depth the latest technology released by the OME Consortium, the OME Remote Objects (OMERO) Server, and its client applications.

A. The OME Server

The OME Server is a Perl-based application running on top of a PostgreSQL database system (http://www.postgresql.org) that supports upload of image data from a variety of proprietary file formats; organization of image data into different types of hierarchies; annotation of individual images, and visualization of all image data and metadata in a web browser-based user interface (http://openmicroscopy.org/getting-started/web-client.html (Johnston et al., 2006; Schifffmann et al., 2006; Swedlow et al., 2003). Most importantly for the purposes of the discussion that follows, the OME Server includes facilities for running and managing image analysis modules. Analysis modules written outside of OME and without any awareness of the OME Server can be accessed by the OME Server using customized code (technically, this is called “wrapping” the application code). The OME Server contains facilities for updating its data model to store new data types, so new analyses can use the OME database to store any generated results. Most recently, this OME Server has been used to define and measure components of the human kinetochore (Schifffmann et al., 2006) and has been modified to support the commercial data analysis tool MATLAB, allowing user-created analysis tools to be easily integrated into the OME Server (http://www.openmicroscopy.org/howto/quantitative-image-analysis-MATLAB.html).

Integration of analysis modules into an OME Server provides two major advantages. For analyses that require a series of sequential steps, the data and parameter inputs and outputs for each step are stored in an OME database that facilitates the provision of output from one module as input for the next. In addition, the results from any analysis are available in a relational database and can be queried, either directly using SQL or using database querying tools like Microsoft Excel or Spotfire. In fact, an Excel-based file for querying and downloading analysis data from an OME Server is available (OME-Excel; http://cvs.openmicroscopy.org.uk/horde/chora/browse.php?f=OME%2Fsrc%2FExcel%2FOME-Excel.xls). Combining image data import, a relational database for metadata and analytics, and analysis module management system and an external querying tool like OME-Excel completes a standard workflow for image visualization and quantitative analysis. A step-by-step guide for this approach is available (http://www.openmicroscopy.org/howto/FindSpots-v2.pdf). Together, these tools provide a complete solution for management and automated analysis for large sets of image data.
A. Rationale for a New Server

Developing new tools for new applications is always a challenge; however, the OME Consortium is developing its tools to enhance its own work and simultaneously releasing tools to support the world-wide community of imaging laboratories. Many laboratories generate software that they use themselves but releasing software for a large community requires much more robust testing, documentation, and consideration of diverse sets of requirements. It is also critical that released software is updated to adapt to changes in operating systems and the software libraries upon which it depends. In the case of the OME Server, there are a number of custom-built functions that are both large and have many external library dependencies. An example is our custom-built object-relational mapping (ORM) code (“DBObject”) that provides communication between the underlying database with the OME Server. Over the last two years, as new Linux and Perl versions were released, DBObject became incompatible with new versions of various Perl modules. This problem is by no means insurmountable, and DBObject was appropriately updated, but this is a significant code maintenance burden. Although DBObject worked well, there are simply other development teams who specialize in ORM and, for instance, have dedicated their efforts to providing a tool that can easily switch between different relational database management systems. One of the most successful is Hibernate (http://www.hibernate.org) that is an open-source library written in Java. Using Hibernate would significantly reduce our code maintenance burden and also provide substantial new flexibility for choosing different database systems including MySQL (http://www.mysql.com) and Oracle (http://www.oracle.com). This combination of flexibility and reduced code maintenance is critical for a small, open-source effort—code maintenance is a critical activity to ensure that users can continue to use released software. As the burden of maintenance grows, more and more precious developer time is spent on this activity, and less is spent on developing new tools for users.

A major goal of OME is the support and provision of remote client applications, allowing users to manage, view, and analyze images in a location and platform-independent manner. The OME Server, written in Perl, required an interface called OME-JAVA, written and maintained by the OME Consortium. OME-JAVA supports the Java-based Shoola client application, communicating to an OME Server. This communication occurs via a data transport mechanism (XML-RPC) that matches a common standard in the community, but unfortunately causes severe performance problems when large data graphs are passed from the OME Server to the Shoola client. Moreover, OME-JAVA provides an interface for Java clients but provides no support for other languages—C++, .NET, Python, etc., that were required by the community (Fig. 2).
Why are remote clients important? Given the complexity of multidimensional image data and imaging experiments, visualization, management, annotation, and analysis all require complex software tools to support user’s requirements. There are many powerful existing software packages that OME’s tools should at least try to support. Flexible support for a variety of application environments with good performance for complex sets of data is thus a critical requirement for an image data management system.

B. Design Criteria for OMERO

For all of these reasons, we decided to design and build a new server, the OMERO Server that would port much of the functionality in the Perl-based OME Server to Java. In taking on this project, it was clear that the code
maintenance burden must be substantially reduced, the system must be simple to install, and the performance of the remoting system must be significantly improved. A major design goal was the reduction of self-written code through the reuse of existing middleware and ORM tools where possible. Numerous successful middleware projects have grown up since the first years of OME and it only made sense to use them. Rather than writing and maintaining our ORM or remoting framework, we can integrate off-the-shelf components and, where necessary and/or possible, contribute back to those projects. After extensive research, it became clear that a number of facilities were available for Java that matched well with the requirements of a server for image data and metadata. Moreover, no other main stream programming language has the same level of widely used specifications and high-performance libraries available for the tasks at hand.

We chose the JavaEE5 specification because of its simplified programming model (and therefore increased maintainability) despite the fact that it was still in the specification stage. After the relative success of JavaEE4, we expected that JavaEE5 would become equally widespread. In addition, the choice of Java for the server was fairly straightforward. No other main stream programming language had the same level of widely used specifications and high-performance libraries available for the tasks at hand. While Java is not the fastest environment, performance of the database system, disk access, and network bandwidth will limit the speed of a Java-based image data server. Given the amount of work in JavaEE middleware, if a bottleneck is discovered in one of the Java components, a host of other solutions are available, simply because of the standardized specifications in the JavaEE environment.

Our goal in building a new server was not to simply replicate the functionality in the existing OME Server, but instead to generate a parallel project that would allow complementary technology. This strategy would certainly create potential difficulties in the short term for users, as desired functionality was split across the two server systems. For example, one of the major features of the OME Server is the use of a dynamic data model, with associated software that supports on-the-fly updates of the database, the application programming interface (API), and the web browser-based user interface included in the OME Server. This feature allows individual scientists to customize the OME Server system to their own needs. This type of functionality is much harder to support in a system designed to support remote applications that could include different applications written in different programming environments. For ease of entry into a system that nevertheless provides significant data management and visualization, the loss of dynamicity and flexibility, which are the hallmarks of OME Server, is an acceptable compromise, at least for the short term. Loss of this facility makes the OMERO Server simpler, and many of the currently missing features, like running chains of analysis modules, can be added back in a controlled way using any one of a wide variety of workflow tools available in Java (http://java-source.net/open-source/workflow-engines). This strategy allows the stepwise development of a new
application that can eventually be integrated with the functionality in the OME Server.

Perhaps most importantly from the user’s perspective, the installation and use of the OMERO system must be relatively easy. Users are now accustomed to automatic installation systems, where all dependencies are included and installation proceeds reliably, without fail or perturbation of the user’s computer. However, installation of a database system, application server, server applications and clients is somewhat more complex than installing a single standalone application. With the growing maturation of Unix-based operating systems (e.g., OS X and Linux), providing the required level of functionality is relatively easy, and we have taken advantage of many of these facilities in building the OMERO distributions (see http://trac.openmicroscopy.org.uk/omero/MilestoneDownloads). As a result, the major burdens for the system administrator are those of installing a relational database like PostgreSQL (http://www.postgresql.org), creating directories, and dealing with user administration.

C. Technical Details of the OMERO Server

Having described the rationale and design of the OMERO Server, the following sections outline the technical aspects of the different components of the OMERO Server.

1. OMERO Metadata Services

The OMERO Server consists of services implemented as EJB3 session beans (http://java.sun.com/products/ejb) that can operate in any JavaEE5 application server. These services make use of Hibernate (http://www.hibernate.org), a high-performance ORM solution, to retrieve metadata from the database. The OMERO services form a layered interface for controlled interaction with the database. The most basic layer consists of ome.api.IQuery and ome.api.IUpdate that provide object-oriented versions of the established SQL verbs SELECT, INSERT, UPDATE, and DELETE with additional logic for security and auditing (http://cvs.openmicroscopy.org.uk/tiki/tiki-index.php?page=Omero API). IUpdate accepts arbitrary graphs of entities defined by the OME specification and guarantees transitive persistence of the entire graphs. All new entities associated with the graph as well as all changes to existing entities are detected by the backend and saved to the database if allowed. The security checks take place on a row basis and function similarly to the Unix file system. Each entity has an owner, a group, and a permissions settings, which defaults to RWR-R- (“read/write for user, read for group, read for world”). Once authorized, all changes to the database are logged in Event table.

The central feature of IQuery is the storage of Hibernate Query Language (HQL) queries on the server that can then be referenced by name, permitting code reuse, and sharing. SELECT security throws an exception if an unreadable
entity is directly requested, and otherwise inserts statements into the SQL for collections to filter unreadable objects. Reads are currently not logged for performance reasons. Except for a few methods in the administration interface, all other interactions with the OMERO Server are built from these simple premises, but can become arbitrarily complex. See ome.api.IPOjos (http://trac.openmicroscopy.org.uk/omero/browser/trunk/components/common/src/ome/api/IPOjos.java) for an example of a well-rounded API that guarantees that the entity graphs returned do not contain nulls. A unique benefit of the OMERO design is that these metadata services sit in the same process as the binary data services, which can make use of preloaded metadata to efficiently parse data from disk.

2. OMERO Image Rendering Engine

A major requirement for any image data application is the ability to display images. In most applications, this is achieved by reading pixel data from a file system and then mapping the pixel data to the 256 gray level available on most computer display monitors. In fluorescence imaging, it is common to record and display multiple channels at once. Typically three, four, or even five separate images must be mapped, and then presented as a color image for painting on a monitor. Because these operations can require many thousands of operations and must be displayed rapidly to support the display of time-lapse movies, most image display software applications use a high-speed graphics CPU and dedicated hardware for image rendering and display. This requirement limits the deployment of these applications to high-powered workstations.

In both the OME and OMERO Servers, an image server—a software application that delivers rendered images to a client—is included that ensures that client applications can display image data. In both cases, this image server reads binary pixel data (sets of “Pixels”) from the server’s file system, renders them based on defined settings, and then transfers the rendered image, ready for display, to the client. Rendering is defined as the process of mapping the raw, multidimensional pixel data from a biological imaging system (e.g., a microscope) to a format that is suitable for visualization on a personal computer monitor or laptop display. The main goal of any visualization system is to provide data for display at speeds that meet or exceed real time.

The OMERO Rendering Engine (OMERO-RE) has been designed to minimize the amount of data transferred to the client and thus removes the requirement for specific graphics CPU, allowing high-performance image viewing on standard laptop computers. The OMERO-RE achieves this by limiting data transfer times because of being close to the data, using highly efficient network transfer protocols, utilizing modern multiprocessor and multicore machines and providing the data to clients in a format that is efficient to display as possible.

When displaying large sets of data, an image visualization system must ensure that an initial viewing of an image and a calculated thumbnail represents a reasonable or “pretty good” image. For this purpose, the OMERO-RE uses statistical
parameters that are calculated during image import and stored in the OMERO database as the basis for initial rendering settings of all images. The details of this statistic-based image rendering will be discussed in a separate publication. Using this strategy, RE and OMERO clients provide high-performance image visualization and thumbnailing in a remote client (Fig. 3C).

Fig. 3  Screen shots of the OMERO clients, connected to an OMERO Server. (A) The OMERO. Admin tool, for managing users, passwords, and groups. (B) The OMERO.Importer tool, shown during setup for an import of proprietary image files (DeltaVision format) into an OMERO Server. (C) The OMERO.Insight tool, shown with the HierarchyViewer and the ImageViewer. For more examples of this tools, see http://www.openmicroscopy.org/getting-started/shoola.html.
3. A Simple Interface for Image Data Management

As mentioned, ease of use was a central design goal of the OMERO system. This was of importance not only for server installation and administration but also for the access of data from a client. With all the metadata and rendering logic on the server, clients can begin working with an OMERO Server extremely quickly. With a Java client, getting started with a running server amounts to:

```java
ServiceFactory sf = new ServiceFactory(
    new Server("myhost"),
    new Login("name","password"));
sf.getQueryService().findAll(Image.class);
```

The use of the Internet Communication Engine (ICE) from http://zeroc.com extends this ease of use to other languages like C++, C#, Python, and Ruby. Interacting with data in an OMERO Server can be as simple as starting the Python interpreter and interacting with OMERO via the console. For example, for using OMERO with languages other than Java, see https://trac.openmicroscopy.org.uk/omero/wiki/OmeroBlitz.

D. A Flexible Server for Binary Data Applications

In general, OMERO can be of immediate benefit to anyone who has metadata information to attach to binary image data that falls within the boundaries of the OME data specification. However, another feature that makes OMERO unique is the use of a Model-driven architecture with extensive code generation from a domain-specific language. This allows not only for the simple addition of new types into the system (conceivably through the use of UML or other simple domain description languages) but also for the application of the entire system to other domains and data intensive applications.

Adapting OMERO for use with other data models (e.g., mass spectrometry) is straightforward. The OMERO developers would suggest some collaboration in order to support the most possible code sharing. Briefly, the files that describe the data model need to be rewritten for each new domain. Some types, however, such as user administration and image files can be shared between instances, and should then be moved to a common package. Although some OME services only make sense for the OME data model, most of the current services could be reused without change. Further services can be added by simply dropping the Java class file into the.ear application file. For more information, see http://trac.openmicroscopy.org.uk/omero/wiki/HowToCreateAService.

E. Clients for the OMERO Server: Leading by Design

The facilities built into the OMERO Server allow the development of new types of data management tools for quantitative analysis of biological data. Most laboratories use a number of different imaging platforms and thus require tools
to manage, visualize, and analyze heterogeneous sets of image data recorded in a range of file formats. Ideally, a single set of applications, running on a user’s laptop or workstation, could access all sets of data and provide easy-to-use access to this data.

We have designed and built three OMERO client applications for use with the OMERO Server. All are written in Java and require Java 1.5 to be installed on the user’s computer (automatically installed on up-to-date OS X and Windows XP systems). OMERO.admin is a simple tool that helps manage an OMERO Server, allowing management of users, groups, and passwords (Fig. 3A). OMERO.importer is a standalone application that allows a user to import proprietary image data files from a file system accessed from the user’s computer to a running OMERO Server. This tool uses a standard file browser to help the user find and specify files for import into the server and then uploads the files to an OMERO Server (Fig. 3B). OMERO.importer uses an open library known as Bio-Formats (http://www.loci.wisc.edu/ome/formats.html) that translates proprietary file formats in preparation for upload to an OMERO Server. Finally, OMERO.insight provides a number of tools for accessing and using data in an OMERO Server (Fig. 3C). OMERO.insight contains three tools for viewing and managing data:

- DataManager, a traditional tree-based view of the data hierarchies in an OMERO Server.
- ImageViewer, for visualization of 5D images (space, channel, time). The ImageViewer makes use of the OMERO-RE, and can provide high-performance viewing of multidimensional images on standard workstations, without requiring installation of high-powered graphics cards. Most importantly, image viewing at remote locations is now possible.
- HierarchyViewer, for viewing, annotating, and manipulating large sets of image data.

These documents will be described in detail in a separate manuscript. A tutorial that shows the use of these tools is available at http://openmicroscopy.org/getting-started/

A key aspect of these tools is the way they are designed, tested, and updated according to user feedback. Initially, we used users in our own laboratories and directly asked them to review our plans and software. Interestingly, this did not produce the most useful feedback—often users clearly did not like what they received, but getting clear statements of what specifically was required seemed very hard. In the last year, we have introduced a more professional process, using a team of design and human computer interaction experts to interview users, record, and analyze their reactions and provide detailed documentation of users’ reactions and requests. This project, known as Usable Image (http://www.usableimage.org), will be fully detailed in a future publication. Currently, most of our major design decisions and new software releases are presented to users by this team and all
reactions and comments recorded and acted upon before full public release. This approach ensures that developers get honest unfiltered feedback from users and provides a mechanism for fully fleshing out user suggestions and ideas before any requested changes or features before changes are made to the software codebase.

F. Current Capabilities

The OMERO Suite of Server and Clients will go through a series of beta releases in 2007, culminating with a “release” of a software package to support a “complete” workflow:

- Import acquired data from a microscope workstation from a variety of proprietary file formats
- thumbnail views of image in the system
- visualize imported data using a remote application
- annotate, organize, and share data in hierarchies
- export image data for publication (TIFF, JPEG)
- manually measure distances and image signal intensities in defined regions of interest
- export image data in OME-TIFF format (http://www.loci.wisc.edu/ome/ome-tiff.html)

The Beta1 release supports the first four of these items, with Beta2 (scheduled for May 2007) filling in some of the missing functionality. In our experience, users most value the flexibility of viewing images at remote locations (office, library, café, and home) and the provision of a data organization tool. A data management tool to users who have struggled to keep track of their data across the 3–5 years of a project appears to be valuable.

V. Future Directions

The OMERO suite of tools provides a strong foundation for users and developers of image management software. Through 2007, the project seeks to expand the functionality of the OMERO suite, to eventually provide complete support for a “standard” workflow—image import, annotation, simple processing and analysis, visualization, and export for publication. A major goal is to provide tools that handle sets of images as a single group to reflect the experimental relationships between them. To this end, OMERO development will focus on user interfaces and server functionality that allow visualization and analysis of complex sets of data, but that always use the power of a data and image server to reduce the data to a representation that is easy to view and comprehend.
Acknowledgments

The authors gratefully acknowledge helpful discussions with our academic and commercial partners (http://openmicroscopy.org/about/). Work in the authors’ laboratory on OME and OMERO is supported by grants from the Wellcome Trust (Ref. 068046, 077128, and 080087) and the BBSRC (BB/D00151X/1). J.R.S. is a Wellcome Trust Senior Research Fellow and declares a potential conflict as a founder of Glencoe Software, Inc., that develops image data management tools based on the OMERO Server and Clients.

References


