ICY: A USER-FRIENDLY ENVIRONMENT FOR ALGORITHM DEVELOPMENT AND DEPLOYMENT

Fabrice de Chaumont, Stéphane Dallongeville, Thomas Provoost, Timothée Lecomte, Alexandre Dufour, Jean-Christophe Olivo-Marin

Institut Pasteur, Quantitative Image Analysis Unit, CNRS URA 2582, 25 rue du Dr Roux 75015 Paris

ABSTRACT

Bioimage informatics has emerged as a new interdisciplinary research endeavor for bringing the power of computational and mathematical sciences into the biological imaging arena. We describe an open-source software platform, Icy, that proposes a comprehensive framework for easy algorithm development and deployment fostering community-oriented efforts. Icy offers a platform to share and publish collaborative algorithm developments, while promoting re-usability and code sharing to ease the development of new algorithms, and simplifying user's feedback and support through a community web site.

Index Terms**—** Bioimage informatics, open source, automated image acquisition, Java, visual programming, script, reproducible research

1. INTRODUCTION

With the advent of modern microscopy, an entire research field dedicated to developing specific image processing and analysis algorithms to answer biological questions has emerged. Yet, software availability and the technical level of accessibility of these algorithms remain two major issues. This is mostly due to the wide portfolio of techniques and methodologies required and developed in fields as diverse as computer vision, signal and image processing, computational modeling, optics, biophysics and computer science [1-6].
Alongside the request for more accessible resources, reproducible research [7] has emerged as a new initiative that urges researchers to make full protocols and software publicly available [8]. The underlying methods thus become easier to apprehend, reproduce and adapt to a similar problem [9]. In such an inter-disciplinary context, adopting and fostering the reproducible research philosophy raises a number of issues: 1) how to decompose complex imaging protocols into small individual tasks (from acquisition to analysis) such that they become easily accessible and adaptable to similar quantification problems? 2) how to create a platform where experts from all backgrounds can collaborate on protocol development, share and publish them online? 3) how to promote re-usability and code sharing to simplify the development of new algorithms? 4) how to encourage user feedback and provide proper support via regular bug corrections and methodological improvements?
Here we describe Icy (icy.bioimageanalysis.org), a community software platform that addresses these issues by proposing a comprehensive framework that strives to bridge the gap between the life science, bio-imaging and image processing communities [10]. Other available software like ImageJ [11], Fiji [12] or BioImageXD [13] also propose solutions to this problem, but in a less coherent manner. Icy is a fully integrated framework designed from the ground-up using modern concepts in programming and ergonomics, taking end-users and developers into equal consideration through a community-oriented paradigm. This is achieved via two key concepts: 1) a plug-in-oriented Java-based software that is adapted to the needs of biological imaging and simplifies the development of new plug-ins; 2) a community web-site to publish, share and manage algorithms and protocols in a straightforward manner, fostering interaction and collaborations between researchers from all backgrounds. Taking advantage of this framework, we and others have created a collection of plug-ins available to the community, implementing state-of-the-art algorithms for biological image

processing and analysis. We illustrate Icy on two specific bioimage applications: automated image acquisition and protocols.

2. CORE COMPONENTS

2.1 Kernel & Plug-ins

Icy structure is made of two layers: a kernel providing core functionalities and a plugin management engine that ensures inter-operability of plug-ins. The kernel of Icy has been designed to provide as much freedom as possible to the conception of plug-ins. The development of plug-ins takes advantage of a rich Application Programming Interface (API) based on 15 cutting edge open source libraries that facilitates the development of algorithms, visualization tools, database access, etc. Icy's image API provides direct access to sequences and enables the management of image data through more than 100 different functions. Those functions start from simple access, like raw (x,y,c,z,t) pixel set in any data format, which is the simpler but slowest access mode, to direct memory access to linear buffers without copy, which drastically increase the computation speed. All calls to the API are thread-safe, encouraging parallel programing. Plug-ins can be of different types. Depending on their nature, plug-ins are integrated directly in the proper section of the GUI: File importer/exporter plug-ins appear directly in the application menu, LUT/Histogram plug-ins in the inspector. Canvas or Viewer plug-ins affect the way data are viewed and appear directly in the menu of the same category. Thanks to dependencies, it is possible to define new types of plugins that can be further fetched by other plug-ins. This inter-operability (Figure 1) encourages the development of small, easy to maintain task-focused plug-ins and maximizes their reusability, thus improving reliability by reducing potential sources of errors.

2.2 Graphical User Interface

The general interface provides consistent window decorations (unified across all operating systems), customization facilities such as color themes (also called «skins»), and the ability to work in single- or multiple-window mode. This design integrates a number of popular GUI designs with which biologists are already familiar. The GUI design essentially relies on a ribbon toolbar, where all plug-ins can be easily accessed and organized in tabs, groups and menus. To facilitate the search of functionalities, we added* a search field on top of the toolbar. This search field is always present to receive queries. Each query is processed both locally to access commands and local plug-ins, and remotely to parse the online documentation of the whole website. We also designed a contextual window that gives access to advanced contextual information and visualization components such as a small image navigator, a Look-Up Table manager component that implements contrast enhancement, histogram interaction and color-map editing for each channel independently. Visualization of 3D data is performed via VTK (http://vtk.org), enabling ray-cast volume rendering of biological sequences. Other views (2D, montage, logarithmic, orthographic projection and magnifier) are also available.

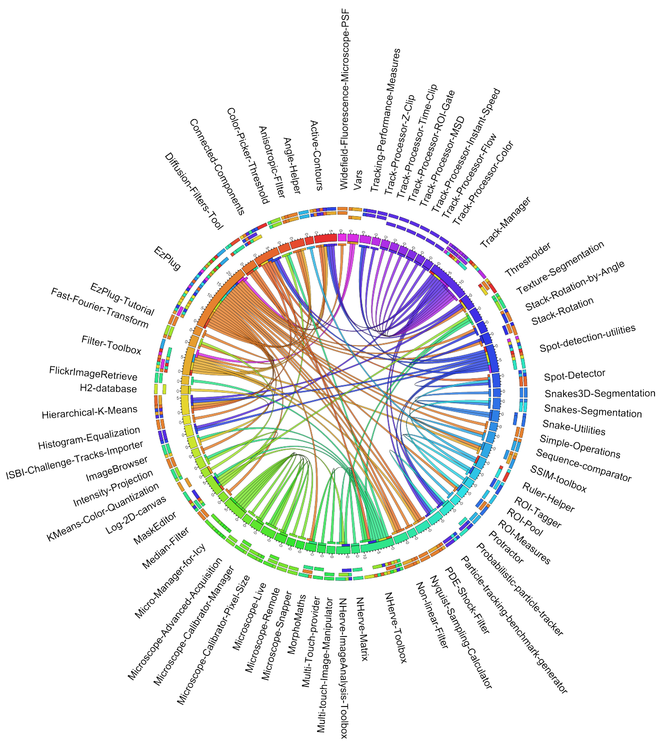
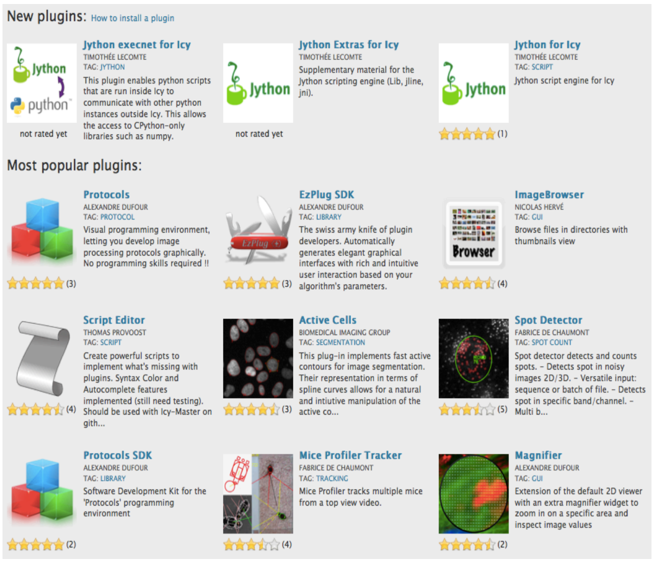


Figure 1: Dependency map of plugin inter-operability. Each arc connects two plugins that use each other's functionalities (30% of plug-ins are represented).

2.3 Website

All the necessary resources and information on Icy are made available through a single, comprehensive website, thus providing the community with a central communication hub. This hub is composed of a number of sections such as documentation, support, blog, tutorial, FAQ. The website centralizes all contributed material such as published plug-ins, scripts and protocols (see below) which are publicly and permanently available. They can be browsed online and also directly from within the application and installed on demand (Figure 2). To facilitate navigation, the website enables cross-links between plugin pages using tags, thus helping people discovering other plug-ins. The website contains all relationship information about plugins, therefore during installation, Icy checks dependencies and ensures that all required plug-ins and protocols are automatically downloaded. The website also gives high visibility to plug-ins by providing a public page for each plugin that is maintained by its creator through the content management system described below. Furthermore, Icy is the only platform where one can rate plug-ins directly from the website[**], thus creating a selection of the best and most popular plug-ins available.


New plugins: How to install a plugin

Jython execnet for Icy
TIMOTHÉE LECOMTE
TAG: JYTHON
This plugin enables python scripts that are run inside Icy to communicate with other python instances outside Icy. This allows the access to CPython-only libraries such as numpy.
not rated yet

Jython Extras for Icy
TIMOTHÉE LECOMTE
Supplementary material for the Jython scripting engine (Lib, jline, jni).
not rated yet

Jython for Icy
TIMOTHÉE LECOMTE
TAG: SCRIPT
Jython script engine for Icy
(1)

Most popular plugins:

Protocols
ALEXANDRE DUFOUR
TAG: PROTOCOL
Visual programming environment, letting you develop image processing protocols graphically. No programming skills required !!
(3)

EzPlug SDK
ALEXANDRE DUFOUR
TAG: LIBRARY
The swiss army knife of plugin developers. Automatically generates elegant graphical interfaces with rich and intuitive user interaction based on your algorithm's parameters.
(3)

ImageBrowser
NICOLAS HERVÉ
TAG: GUI
Browse files in directories with thumbnails view
(4)

Script Editor
THOMAS PROVOOST
TAG: SCRIPT
Create powerful scripts to implement what's missing with plugins. Syntax Color and Autocomplete features implemented (still need testing). Should be used with Icy-Master on gith...
(4)

Active Cells
BIOMEDICAL IMAGING GROUP
TAG: SEGMENTATION
This plug-in implements fast active contours for image segmentation. Their representation in terms of spline curves allows for a natural and intiutive manipulation of the active co...
(3)

Spot Detector
FABRICE DE CHAUMONT
TAG: SPOT COUNT
Spot detector detects and counts spots. - Detects spot in noisy images 2D/3D. - Versatile input: sequence or batch of file. - Detects spot in specific band/channel. - Multi b...
(5)

Protocols SDK
ALEXANDRE DUFOUR
TAG: LIBRARY
Software Development Kit for the 'Protocols' programming environment
(2)

Mice Profiler Tracker
FABRICE DE CHAUMONT
TAG: TRACKING
Mice Profiler tracks multiple mice from a top view video.
(4)

Magnifier
ALEXANDRE DUFOUR
TAG: GUI
Extension of the default 2D viewer with an extra magnifier widget to zoom in on a specific area and inspect image values
(2)

Figure 2: Plugins page sorted by rating and popularity on the Icy website.

2.4 Content management

The content management system assists the developer in deploying, monitoring and maintaining plug-ins. The developer can enrich this page with a number of information such as images, video, tags, links to publication or website, iconography, splash-screens, abstracts, slideshows and documentation which will then be crawled by search engines. Icy also provides per-plug-in automatic audit for developers to increase their visibility.
The CMS handles the versioning and distribution of plug-ins, which are automatically updated on the desktop applications. It also allows the developer to roll back to a previous version in case of major problem (broken compatibility, missing feature, broken package integrity, etc.). An automatic bug report feature is provided to allow the user to inform the developer a problem has occurred while using the plug-in. Corrected plugins uploaded on the CMS are automatically deployed. This mechanism increases reliability of plug-ins. Non-public collaborations are also supported by Icy: all CMS features are available independently of the main website through private repositories.

2.5 Augmented visualization

Icy provides two key components to manipulate and annotate imaging data in context (Figure 3): 1) a flexible annotation mechanism, enabling users to create unlimited regions of interest on the image, which are permanently saved for further editing, and can be further exploited for targeted analysis. The annotation mechanism also allows plug-ins to incorporate custom annotations (ROI, text, analysis results, etc.) and provide user-interaction. Once the information is properly tuned and informative, a screenshot feature allows capturing the fully annotated data, producing high-resolution images suitable for communication, scientific reports and publications; 2) a multiple synchronized viewers system, allowing to view the information in multiple ways, while annotations and data are synchronized in real-time across all viewers.

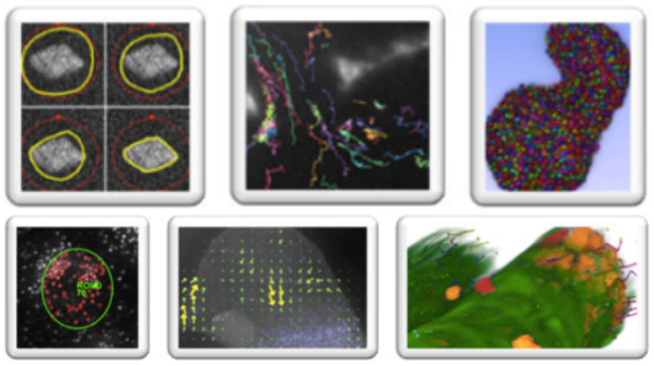


Figure 3: Different examples of plugins and their image data visualization.

3. DEVELOPING WITH ICY

3.1 Rapid application development

Icy is fully written in Java. It comes with an internal library called “EzPlug” that simplifies two redundant and tedious aspects of plug-in development, writing the graphical interface, loading and saving user parameters and allowing the user to interrupt a running process. EzPlug allows creating standardized user interfaces without requiring any knowledge in graphical programming. For each parameter declared by the developer, a graphical component is automatically generated, and is tailored to receive user input depending to the parameter type (e.g. drop-down lists, bounded values, check boxes, etc.). The final generated interface lays out all graphical components in an ordered manner, and adds a control panel with action buttons to run or interrupt the plug-in, and to import or export user parameters.

3.2 Protocols

Protocols are a graphical front-end that implement software development by enabling end-users to design image processing pipelines (termed ‘protocols’) in a graphical manner, without any programming knowledge. A protocol is constructed by assembling blocks representing elementary image processing tasks, and linking input and output via connecting lines (Figure 3). Protocols provide facilities such as switches, loops, automatic parameter checking and scheduling to ensure consistent execution. A protocol and its parameters may be saved and submitted to the Icy website for immediate sharing and/or perma-linking in a publication. Any such protocol may then be downloaded and seamlessly installed (all required plug-ins are automatically installed). Any user can hence re-run a given protocol using the original parameters, and further edit the pipeline by adjusting parameters or adding and removing blocks.

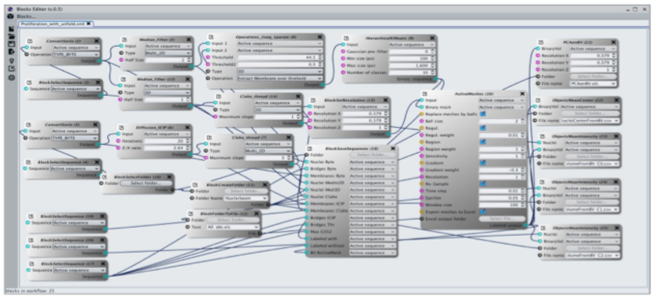


Figure 3: Protocol development through Protocol Editor.

3.3 Scripting in JavaScript and Python

To complete functionalities that may not exist as an already developed block for protocols, and to enable the creation of code directly within Icy, we released[***] a full scripting suite directly accessible under Icy. To help using the API, we created a specific editor enabling auto-completion of code, and parsing the documentation directly inside the source code of the existing objects. This editor also indicates with symbols where exactly errors specified by the compiler are located. As for plug-ins and protocols, script resources are centralized online on the website. Fifty scripts of different complexities are shared online[****]. They can be browsed and examined online with the proper syntax highlighting.

4. EXAMPLES OF EXISTING TOOLS

4.1 Computerized control of image acquisition hardware

New challenges in bioimaging impact the microscope acquisition process. Biologists need automation tools managing the observation of samples over long period of time, and assisting in the detection of biological events. Optimizing image interpretation calls for real time analysis tools coupled with augmented quantitative information over the images. These new requirements imply a direct link between the control of the microscope and the image analysis platform: this link can be implemented via software such as the open-source package Micro-Manager (www.micro-manager.org*)*. We have associated Micro-Manager and provide Icy bundled with a patched version of Micro-Manager. This patched version removes all the redundancies between Micro-Manager and Icy’s functionalities, combining its powerful hardware control abilities with the advanced visualization tools provided natively by Icy (rich look up tables in 2D/3D and 3D ray-casting thanks to VTK). The Icy visualization system also allows observing a same sequence within different viewports with different settings to literally set up cameras to watch structure of samples. A real time 3D acquisition example using multi-viewport can be found at http://youtu.be/7j7xFwF_PlU. Icy also provides a framework that allows several plugins to access Micro-Manager. This enables user-customizable combination of automatic tools. For instance a first plugin tracks a cell and drives the x-y stage, a second displays the acquisition in 3D and a third one simultaneously performs an analysis over the image for real time quantification (Figure 4).

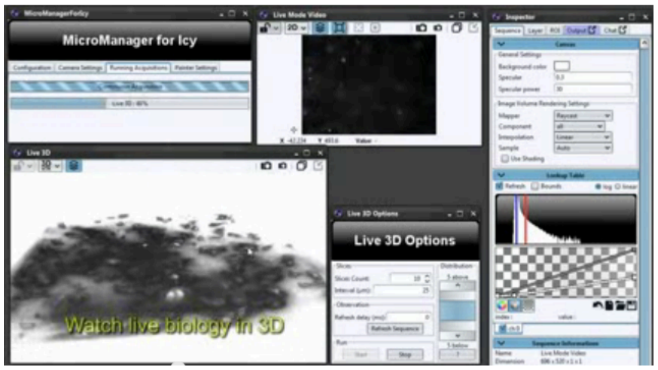


Figure 4: Micro-Manager for Icy in action, demonstrating live acquisition combined with 4D visualization.

4.2 Intelligent image acquisition and analysis

One of the main advantages of Icy is to enable the creation of very complex tools in a simple manner. Thanks to the communication between all modules of Icy, one can create an execution pipe either with scripts or protocols and obtain a live chain of multi-threaded analysis able to respond in real time. Then, the different displays of Icy help augmenting and enriching a scene by adding overlays on top of the live acquisition display provided by Micro-Manager. Those views act as helpers that perform analysis directly over a live stream. All the algorithms present in Icy can be used in such schemes and help the end-user (in our case a biologist) examine if the context of the sample corresponds to the one of the acquisition. Thanks to this augmented visualization, it then becomes possible to monitor in live the information content and the quality of a scene through examination of parameters such as cell density, colocalization, counts, shape and inter-distances.

5. REFERENCES

1. Swedlow J.R., Goldberg I., Brauner E. and Sorger P.K., “Informatics and quantitative analysis in biological imaging”, *Science*, 300, no. 5616, pp. 100-2 (2003)
2. Peng, H., Ruan, Z., Long, F., Simpson, J.H., and Myers, E.W. “V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets”, *Nature Biotechnology*, 28, no. 4, pp. 348-353 (2010). DOI: 10.1038/nbt.1612.
3. Kalinka, A.T., Varga, K.M., Gerrard, D.T., Preibisch, S.W., Corcoran, D.L., Jarrels, J., Ohler, U., Bergman, C.M. and Tomancák, P., “Gene expression divergence recapitulates the developmental hourglass model”, *Nature*, 468, no. 7325, pp. 811-814 (2010)
4. Meijering E., Jacob M., Sarria J. C. F., Steiner P., Hirling H., Unser M. “Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images”, *Cytometry Part A*, vol. 58, no. 2, pp. 167-176 (2004)
5. Zimmer C., Zhang B., Dufour A., Thébaud A., Berlemont S., Meas-Yedid V., Olivo-Marin J.-C., “On the digital trail of mobile cells”, *IEEE Signal Processing Magazine*, 23, no. 5, pp. 54-62 (2006).
6. Chenouard N., Dufour A. and Olivo-Marin, J.-C., “Tracking algorithms chase down pathogens”, *Biotechnology Journal*, 4, no. 6, pp. 838-45 (2009)
7. Fomel S. and Claerbout J., "Guest Editors' Introduction: Reproducible Research", *Computing in Science and Engineering*, 11, no. 1, pp. 5–7 (2009). DOI:10.1109/MCSE.2009.14
8. Ince, D.C., Hatton, L. and Graham-Cumming J. “The case for open computer programs”, *Nature*, 482, 485–488 (2012). DOI:10.1038/nature10836
9. Vandewalle, P., Kovacevic, J. and Vetterli, M. “Reproducible Research in Signal Processing - What, why, and how”, *IEEE Signal Processing Magazine*, 26, no. 3, pp. 37-47 (2009)
10. de Chaumont F, Dallongeville S, Chenouard N, Hervé N, Pop S, Provoost T, Meas-Yedid V, Pankajakshan P, Lecomte T, Le Montagner Y, Lagache T, Dufour A, and Olivo-Marin, J.-C. “Icy: an open bioimage informatics platform for extended reproducible research”, *Nature Methods*, 9, 7, pp. 690-6 (2012)
11. Rasband, W.S., ImageJ, U.S. National Institutes of Health, Bethesda, Maryland, USA, http://imagej.nih.gov/ij/, 1997-2011.
12. Fiji : http://fiji.sc
13. BioImageXD : http://www.bioimagexd.net

[*,**,***] Features added respectively on January, February and March 2013.
[****] Count as of June 2013.