IPOL: Reviewed publication and public testing of research software

Nicolas Limare, Laurent Oudre, Pascal Getreuer

IPOL: Reviewed Publication and Public Testing of Research Software

Nicolas Limare, Laurent Oudre, Pascal Getreuer
CMLA, ENS Cachan
61 Av. du Pdt Wilson, 94235 Cachan Cedex, France

***Abstract*—With the journal Image Processing On Line (IPOL), we propose to promote software to the status of regular research material and subject it to the same treatment as research papers: it must be reviewed, it must be reusable and verifiable by the research community, it must follow style and quality guidelines. In IPOL, algorithms are published with their implementation, codes are peer-reviewed, and a web-based test interface is attached to each of these articles. This results in more software released by the researchers, a better software quality achieved with the review process, and a large collection of test data gathered for each article. IPOL has been active since 2010, and has already published thirty articles.**

I. Introduction

For computational sciences, detailed algorithms and implementations are an integral part of the research results and should therefore be published extensively. This has been continuously pointed out in the literature about reproducible research, since the seminal papers by Claerbout [3] and Donoho [1]. Moreover, in research areas focused on the study and elaboration of computational methods rather than the results of these computations, like signal processing or econometrics, we don't produce, publish and exchange the outcome of a numerical experiment (which would need to be reproducible) but algorithms, methods and tools, which must be reusable. This reusability of computational algorithms supposes at least a completely detailed description, and can include the availability of an implementation.

This goal is far from being attained in the image and signal processing community, where in 2009 only 12% of published articles included the implementation details and 9% provided a source code [13]. Algorithm exchange in this community faces serious obstacles: multiple software developing environments have grown in each research group without interoperability. Source code is subject to portability issues and depends on local tools to process or exploit the result. Software maintenance is a costly issue. Additionally, many scientists have limited software engineering skills. Software maintenance and consolidation is therefore one of the main problems in this field [2].

In this context, research groups are unable to communicate efficiently by software and are therefore limited to paper journals and conferences. Software libraries and development environments are not a sufficient response, because in the absence of a common base, they reinforce the software fragmentation.[1] Existing software journals, when they publish codes [15], [16], lack the precise scientific evaluation and specification of the implemented algorithms and a discussion about their properties, qualities and limits. When an implementation is provided by the authors of a research article, it is usually available *as-is* on their personal research web pages,[2] with varying quality and documentation. Out of the peer-reviewed scientific publishing process, *there is no control that the source code as provided exactly implements the algorithm as described, there is no guarantee on the correctness of this implementation, or on its long-term availability*. Studying recent image processing algorithms and implementations provided by their authors, we have systematically observed that the proposed code differs from the paper publication, that the paper publication is not enough to characterize an algorithm, and that conversely the disclosed code contains parts that are not documented or explained in the paper.

Multiple recent initiatives to address the problems of conservation, exchange and validation of computational science software material (code, executable programs, data and results) show an increasing concern about the reliability of the computational science corpus.

Our initiative in this field was the creation in December 2010 of a research journal, Image Processing On Line (IPOL). Unlike classic academic journals, where the only required submission is the manuscript itself, an IPOL article has three components:

Documentation.
: Similar in its form to a classic research article but focused on the implementation, the *precise description of the algorithm* must be detailed enough to allow any specialist to implement it in their own programming language and environment.[3]

Implementation.
: The *source code* is published, after a validation by reviewers, who verify that the algorithm description

[1] The recent *IPOL 2012 Meeting on Image Processing Libraries* illustrates this fragmentation, with 13 different image processing libraries produced by the research community without any compatibility layer (http://www.ipol.im/news/20120627_image_processing_libraries/).

[2] See Xin Li's *Reproducible Research in Computational Science* (http://csee.wvu.edu/~xinl/source.html) for examples.

[3] This level of detail is similar to the requirement for a disclosure, *sufficiently clear and complete for it to be carried out by a person skilled in the art*, as required in patent law (European Patent Convention, art. 83).

is complete and detailed, and the implementation matches this description.

Demonstration.

A *demonstration web service* is attached to every article, allowing users to test the algorithms on their data with their choice of parameters. The full history of the experiments performed with this service is also publicly available.

These principles promote validation and comparison of the algorithms, reuse of implementations, and progressive compilation of a verifiable state-of-the-art based on trusted software.

Since the first component of an IPOL article (algorithm description) is somehow similar to a classic journal paper, we shall only focus on the last two components (implementation and demonstration) in this article. Section II describes the software peer review process and guidelines used for evaluating correctness, readability and usability of the software. Section III introduces the demonstration aspect of IPOL and the conceptual and technical structures of online demos. In Section IV, we shortly discuss legal and copyright issues inherent to software publication. Finally, Section V presents a preliminary study on results obtained since 2010, along with a discussion on the future of IPOL and on its possible extension to other domains.

II. Software

A. Software Peer Review

If software is to be considered as a first-class product of the research activity, we consider it should be submitted to the same procedure to maintain standards, improve quality and provide credibility as applied to regular research papers. This is achieved in academic publishing via the peer review process, so we also review the software before publication.

But contrary to classic academic journals, which only review and publish manuscripts, our review process must consider both the manuscript (which must contain precise mathematical description of the algorithm) and the software (which must match the description of the algorithm). Both are important because the source code ultimately specifies *how the program works*, and the higher-level description of the algorithm tells *what the program is supposed to do*.[4] The role of referees in IPOL is therefore different from (and complementary to) their role in a classic journal.

Publishing software in a scholarly journal implies that journal readers can expect some level of quality for the software. Such a software should be usable and have predictable effects across a reasonable diversity of present and future computing environments. The source code should be available, otherwise no one can say what the software is really doing. It must be readable and documented, analogous to how developments in a mathematics paper should be readable and explained.

[4]These dual points are made by Douglas Crockford ("*And anything less than [the software] doesn't really tell you anything about how it's ultimately going to behave.*") and Joe Armstrong ("*The code shows me what [a program] does. It doesn't show me what it's supposed to do.*) in *Coders at Work* [11].

In IPOL, the software review applies the following principles:

- the program must implement the algorithm exactly as described in the associated manuscript;
- the implementation must be understandable by other researchers, it is intended to be read, studied and verified;
- the code must be usable on a reasonable variety of computing environments, and reusable for future research works.

The reviewers, selected from the same research field, are asked to read the source code and to request explanations and corrections until the software satisfies these three principles. They do not necessarily need to be programming experts since software quality, in this context, is not its computational speed. It is the care taken to clearly convey its content to its recipients, the readers of the software journal interested in using and understanding it. As such, any researcher of the domain who has already done some programming and understands the software review principles should be able to fulfill the conditions to be a reviewer, as the review is more practical than purely technical. There are as many potential reviewers for a research software as there are many potential authors to publish such software.

The interactions we could observe in IPOL between authors and reviewers make clear that the implementation is improved by our reviews: software is tested outside of its development environment, bugs are found and fixed, alternative implementation designs are proposed, and the implementation choices are better documented since the reviewer approaches the code from their external point of view.

B. Software Guidelines

Guidelines were developed for IPOL as an attempt to guide the authors and reviewers. Their goal is to increase the readability and usability of the programs published in the journal with a set of requirements and recommendations for an article to be accepted. These guidelines are our first tentative to define how good research software should be distributed. They establish a clear and shared understanding of what is expected from the authors. Their application depends on how they can be understood and verified. To help authors, editors and reviewers, a set of simple algorithms implemented by following these guidelines are provided. These examples are completed by a tool[5] authors and reviewers can use in order to test the software against some of the guidelines and then focus on high-level analysis of the code.

During the review process, reviewers are encouraged to use these guidelines as a reference to ask for modifications to the source code, but ultimately they are free to be more or less flexible, and the editor in charge of a submission has the final word on its acceptance. The software guidelines are not exact rules enforced during the review, their aim is to express clearly the editorial board understanding of a "good

[5]Software Guidelines Validation, https://tools.ipol.im/swg_check/.

implementation worth being published, reliable, and useful to other researchers."

Such software guidelines are a compromise between desirable software engineering criteria and the effort one can expect from authors. Coding style guides[6] have been written and are used to enforce a unified visual style and improve code quality. Other reference books provide advice on coding style, program structure, or the additional matter of security risks in software [4], [5], [9]. But with the current guidelines, we only focus on our priority: correct, readable and usable programs.[7]

These guidelines may seem obvious for a seasoned programmer. Compared to the stringent rules enforced in some branches of the software industry, these requirements are minimal. But these are not the profile and work environment of computational science researchers, and even the simple expectation of being able to use a research code is rarely fulfilled [10]. Our goal with the software guidelines is to establish a basic quality criterion for research software, and we show by the experience of publishing IPOL that this quality can be attained if required as a journal policy.

We believe that these guidelines are relevant for other computational science research communities, after some adaptation of the domain-specific items like data file formats, common software libraries, or essential programming languages. The full text is available online,[8] and we only outline the essential items hereafter.

1) Packaging and Content: The first set of guidelines defines how a program is distributed and what it contains. Their goal is to ensure that the program can be easily identified, transmitted and evaluated, and that it can be manipulated by everyone.

> *The software must be distributed as a ZIP or TAR/GZIP compressed archive, named after the name and version number of the software. This archive must only contain files useful to build, use or study the program: no by-products of the development tools, no temporary versions of the code.*

2) Implementation: The second group of guidelines address portability of the software. No provision can guarantee that a program will be usable on all present and future computing systems, but known causes of incompatibilities can be avoided.

> *The complete implementation of the published algorithm must be provided in source code. This code must be written in one of the programming languages allowed, and tested with strict compiler options. It must not require a special hardware platform or operating system. The implementation can only depend on a list of allowed essential libraries chosen by the editorial board for their quality, portability and stability. This software must be compiled by an automated build system, and produce a program usable from the command-line and with standard and well-known file formats.*[9]

3) Copyright, License and Patents: The third category of guidelines ensures that the rights of the authors, contributors, inventors, readers and users are clearly mentioned and respected. These are the legal guidelines.

> *For every source code file, authors of every source code file must be mentioned with the distribution license. If a patent might be linked to this file, a patent warning must be inserted after the copyright attribution.*

4) Documentation: The last set of guidelines covers the documentation, and has provisions to guarantee that reviewers and readers of published implementations will have all the information they would need about the software. This includes generic information such as references to the article and usage instructions. But the source code itself is considered a published material and is expected to be read, so some rules are added to guarantee that the implementation is readable and understandable.

> *All the essential information about the software must be mentioned in a README file: name and brief description, reference to the article, authors, version and future releases, copyright, patent and license, build and usage instructions. The code must be formatted consistently with limited file and line length, and structured into separate files by abstraction level. Every function and every non-trivial part of the code must be explained in the comments, and linked to its counterpart in the document detailing the algorithm. Some example data is expected with the code.*

III. Demonstrations

A. Principle

Despite the availability of a reliable source code, an algorithm may not be immediately usable because its compilation, installation and use are not straightforward. Some researchers are reluctant to get into a compilation procedure to check an algorithm. Most would prefer a quick evaluation before they consider spending more time to study the article in depth. Our proposed solution is to provide an experimental environment directly accessible over the network for every algorithm published with its code.

This test interface uses the reviewed and published code to process, in real time, data freely submitted by the users. Every experiment performed with original data on this system

[6]Style guides since the original *Indian Hill Recommended C Style and Coding Standards* have been collected and archived by Chris Lott (http://www.maultech.com/chrislott/resources/cstyle/).

[7]These priorities are close to the "*six features of useful machine learning software*" as defined by the JMLR and MLOSS projects [16], [17], [12]: *[useful] software is usable, documented, robust, it has well-defined interfaces, it uses existing interfaces and standards, it has testing routines*.

[8]IPOL Software Guidelines, https://tools.ipol.im/wiki/ref/software_guidelines/.

[9]For IPOL, we chose to restrict the publications to languages clearly defined in standards, and to maintain some level of compatibility between all the codes published. This resulted in the restriction to the C89, C99 and C++98 programming languages. For other journals, this selection will depend on the research area and habits of the community involved.

is stored and publicly available. This experiment archive is an efficient mean to assess the performance of an algorithm over a large collection of input images.

This results in a large-scale testing of every published software, where the testers are the journal readers. They have different backgrounds but a common interest in image processing, they test the algorithms for unexpected yet interesting use, and a majority of these tests are performed with pertinent data and settings. On IPOL, this archive collects between 50 and 100 new tests per day as of July 2012, for a total of tens of thousands of experiments stored and freely browsable, more than any researcher would be able to perform. These tests are available from the preprint stage of the articles; they revealed software bugs, unexpected uses and limits of the algorithms, and they contribute to the quality of the software published in the journal.

Moreover, as noted by Li about image denoising, *the gap between mentally reproducible and experimentally reproducible research is often the biggest challenge [...] A good and solid theory is mentally reproducible but does not always lead to better experimental results* [6]. This is true for every field of image processing research, and probably for computational science as an experimental science: no matter how strong and innovative the theoretical background of an algorithm, one can not be convinced of its performance and usefulness without using the algorithm on real-world data.

B. Typical Demo

The base workflow of an IPOL demo is always the same:

1. Users select or upload input data. This data is converted to the format expected by the research software.
2. Users can edit the input, select some algorithm parameters, or both.
3. The input data is processed by the research software.
4. The result is shown.
5. If the input was submitted by the user, this experiment is archived. The user can go back to step 1 or 2.

Variations between demos are essentially changes in steps 2 (algorithm parameters), 3 (how the data is processed) and 4 (how the result is displayed).

C. Development and Deployment

Our demo system is a web interface to the software reviewed and published. The web model has the advantage of being accessible to almost any connected user. Despite varying support for advanced techniques across browsers, web base technologies are truly cross-platform. Our demos do not require any setting or tools beyond those needed to access the journal content: a network connection and a web browser.

In the IPOL journal, the image processing demo system was built on these priorities:

Simplicity.
: We avoid unnecessary technology layers and software dependencies;

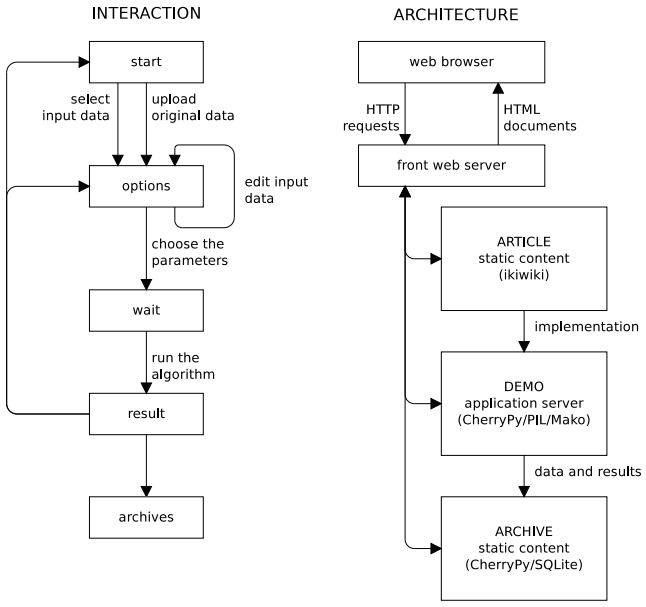

Fig. 1. Workflow and design of the web demo system.

Flexibility.
: We do not know yet which interface feature will be needed for future demos.

Openness.
: Authors and editors can independently create and test their own demos, which are integrated in the demo system.

Accessibility.
: No one should be barred for mere technical reasons if this can be avoided.

The user interface is made of classic HTML documents (see Figure 2). We use JavaScript to enrich some HTML forms, but this is essentially sugar coating, and so far JavaScript is not required to use IPOL. Flash and Java are not used either.

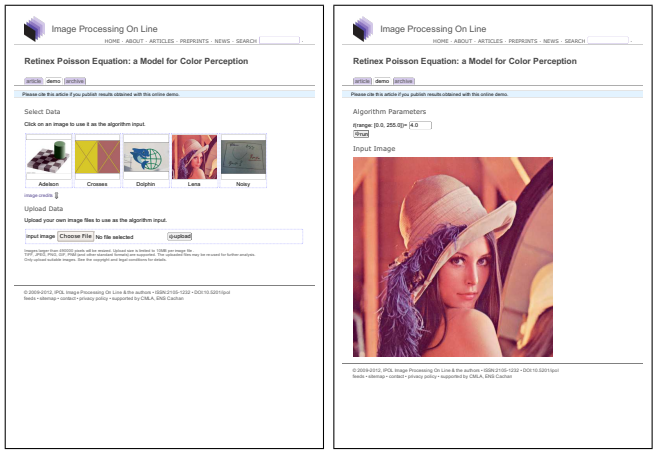

Fig. 2. Input selection, parameter choice and computation result pages for a typical IPOL demo.

The current version of the demo system is still a work in

progress; it is currently based on the CherryPy framework to implement the demo system as a Python web application. Until a stable state is reached and documented in future research literature, the current code is available online.[10]

IV. Other Aspects of Publishing Research Software

Other questions were investigated during the development of IPOL, such as the intellectual rights regulation (copyright and patents) applied to computer programs and research articles or the insertion of software material in the academic publishing ecosystem.

A. Copyright and Patent Policy

Intellectual rights regulation is heterogeneous, based on national jurisdictions, and software and manuscripts have different legal status, even if we would like to combined them as a single product of the research activity. In this context, we propose a copyright and license policy to facilitate dissemination and reuse of research works.[11]

Software reviewed and published by the journal must be released under a Free Software license, because the implementation of an algorithm should be as accessible and reusable as its description, and anyone can reuse the knowledge obtained from a research journal.

Possible patent issues only need to be mentioned with the software, but evaluation of their legal impact is left to the users because it depends on their local jurisdiction. Moreover, we believe software in a research journal is covered by the exceptions for research and experiments found in most patent regulations.

In addition, article manuscripts and datasets are distributed under a permissive Creative Commons license, and every material published by the journal is available freely, in Open Access. Finally, no copyright transfer is claimed by the editor. This policy was designed to maximize the exposition and usefulness of published works; we expect that it will benefit the authors and the research community as a whole.

B. Authorship Attribution

One aspect of software authorship attribution is already covered by the Free Software licensing terms: they usually require, based on copyright law, that any reuse or modification of the code mentions the origin and authors of this code.[12] A similar mechanism can be used for the descriptions of research programs by releasing them under a similar license, such as a Creative Commons CC-BY license, which stipulate proper attribution when these works are reused.

However, this copyright protection only covers situations when the content of a software is *reused*. We have more interest in obtaining citations for research software the same way research papers are referenced: credit given for important ideas, building blocks, and major influences on research. This cannot be obtained from a legal framework: ideas are free, no one can be forced to acknowledge the origins of their works. This is not different for software and manuscripts: proper credit relies on the integrity of the authors and peer-review control. It is a social norm with variations among research communities, not a rule.

Our strategy to obtain proper credit for software is to format and distribute this software as research articles: source code is downloaded from a journal website, with a manuscript formatted like any other research paper,[13] and clearly identified with full bibliographic metadata and digital identifier. So far, we observed some references to these software articles in regular research journals, but it is too early to conclude on the efficiency of this approach.

Finally, one must acknowledge that research literature and software usually have different ecosystems. Research papers are published in journals, indexed in bibliographic databases, and mentioned in the references section of other articles. A similar process happens for software in the open source community, but in different places: software is published in online source code repositories and indexed by software hosting facilities and directories.[14] When a software is reused in other projects, attribution to the initial authors is maintained in the code and documentation. As an experiment, we intend to complement the distribution of IPOL software via the journal articles with an insertion into open source repositories and directories, with references to the journal.

V. Discussion

A. Usage and Feedback

1) Usage: In less than two years, IPOL published thirty algorithms, precisely described, with a portable, usable and commented implementation. This first result shows that it is possible to require authors to provide an implementation for their algorithms, and to publish it with a review process based on quality guidelines. The authors currently come from a dozen universities worldwide, usually linked to the journal core team via current or previous research collaborations. IPOL and the SIAM Journal on Imaging Sciences (SIIMS) have an agreement to promote cross-publications: two articles can be published about the same algorithm, one in SIIMS focusing on the mathematical analysis and the other one in IPOL expanding on implementation, and these complementary articles include cross-references.

We do not have any impact indicator for the articles published in IPOL in terms of reference numbers and impact factor because the journal is too young to appear in bibliometric

[10] IPOL Demo Service, http://dev.ipol.im/git/?p=nil/ipol_demo.git.

[11] IPOL Copyright and License Agreement, https://tools.ipol.im/wiki/ref/copyright_license_agreement/.

[12] This has been enforced in courts for prominent cases. See for example the gpl-violations.org project, http://gpl-violations.org/. Free Software licenses usually add other conditions, not relevant in this article.

[13] IPOL recently switched from articles formatted as web pages to PDF documents, because it seems that only PDF files are considered a credible format for research articles, by other researchers and by bibliographic databases.

[14] SourceForge (http://sourceforge.net/), Google Code (http://code.google.com/) and GitHub (https://github.com/) are some of the largest software hosting services. Freecode (http://freecode.com/) and Ohloh (http://www.ohloh.net/) are two open source software directories.

studies. Meanwhile, our alternative is to observe the number of visits, source code downloads, and demo tests.

The algorithms published in IPOL were submitted to more than 100000 online tests as of August 2012. IPOL's impact on the image processing community is still difficult to assess since authors are not used to citing implementations and online articles as regular research articles, however, these numbers provide a good hint to the popularity of the journal. During the 2010–2011 academic year, we could observe visits coming from more than 850 scientific, research and educational institutions worldwide.[15]

As far as software is concerned, a study showed that on the period December 2010 to November 2011, 6% of visitors downloaded a source code or dataset attached to an article (while the others only read the article or played with the associated demo). This percentage corresponds to more than 5000 downloads in less than a year.

Interestingly, just like number of citations is an indicator of the popularity of an article, we think that the number of tests performed on a demo system can give a hint to the interest for an algorithm. Despite a strong possible bias resulting from variable exposure of these works, we can observe a ranking of the articles based on the popularity of their demo. This measure, together with the number of views of the article and the number of downloads of the software, suggests an estimation of usefulness of an algorithm for the general public, complementary to the measure of its influence in the research community as hinted (also with bias) by number of references in other research papers.

Detailed statistics were compiled in November 2011 and at this time, articles were viewed on average 300 times per month, but with large variations: the most popular article received more than 1500 visits, and the least exposed article was viewed 50 times. Similar numbers were observed for demos: the most popular demo was used 1150 times, and the least used one processed only 15 images. Different factors can explain these variations: some IPOL articles are very exposed on the Web via blog reviews and references in technical forums, and some other articles are more confidential and only found when browsing the IPOL index. The popularity of IPOL articles is also related to the trends of the discipline: some algorithms are connected to an active research domain, others are out of fashion. Finally, some algorithms are more accessible or useful for the general public, such as color correction and denoising for amateur photographers.

Around one third of the tests are done with original data uploaded by visitors. This has allowed the creation of 50000 new archives as of August 2012. Archives of the experiments conducted on the online web demonstration system also reveal some information for every article. One can observe unexpected uses of some algorithms deduced from the input data: building blueprints, microscope views of cells and satellite photography reveal interest from the architecture, biology and geography professional fields.

The complete experiment archives occupy 100GB of storage space as of August 2012. Given the decreasing cost of computer storage and the development of distributed storage solutions, archive size is not an issue. Despite being moderated a posteriori, we faced very few cases of abuse by insertion of offensive images in the archive, and these one or two cases per month did not justify implementation of a systematic validation of archive content.

Finally, with 200 demo executions per day, each taking less than 30s, the demo server is always available and IPOL does not face a pressure on computational power.

2) Feedback: Twenty-four authors of IPOL articles, published or in process, answered a survey in November 2011.[16] According to this study, the redaction of an IPOL article seems to require as much work as writing an article for a classic journal and adapting a software to the IPOL requirement seems to double on average the time required for development, but we can observe large deviations probably connected to the individual author proficiency in software development and the origin and characteristics of every software. All the authors declared they would cite an IPOL article in a research paper and suggest colleagues to read IPOL materials, and more than 90% of them would write another article in IPOL and suggest colleagues to do the same. When asked to detail the reasons for this satisfaction in a supplement survey, they cited the use of IPOL as a complete archive of all materials related to an algorithm. The web demonstration tools are said to be useful to get a better understanding of the algorithms, both for authors and for readers. Publication of the implementation is appreciated as a motivation to produce a better program.

B. The Future of IPOL

IPOL has been developed as an experiment in evaluating, sharing and publishing computational research materials. After two years of activity, the definition and goals of the journal have stabilized, new articles are continuously published, and the number of readers is constantly increasing. The next steps will be to handle the growth of the journal.

New authors must be convinced to contribute. We observe that researchers are usually convinced in IPOL as readers, but do not become authors spontaneously. The low profile of IPOL is probably a reason, and the progressive indexation of the journal in bibliographic databases may improve this aspect, but the author experience probably needs to be simplified and closer to the policies of other journals, and that is why the journal is currently switching to a publication format based on LaTeX and PDF files. Code review can also be facilitated by the adoption of automated testing procedures to pre-screen the software submitted to the journal.

IPOL's demo system is modular in the sense that it can easily accept new demos based on existing ones or with a largely different workflow. However, it is currently monolithic

[15] IPOL usage report for 2010/09 to 2011/08 (http://www.ipol.im/news/20110923_stats/).

[16] Author Survey 2011 (http://www.ipol.im/news/20111218_survey/) and Author Satisfaction Feedback (http://www.ipol.im/news/20111219_satisfaction/).

with a single back-end application and server to build the research programs, run them on user-submitted data, provide user interface, feed the archives and display archive content; everything is on a single server. The next step will probably be separating these functions into different services which could be managed by different machines and with the possibility to grow and accept more traffic by replicating the servers. This will be the opportunity to add functions missing in the current system: a procedure to build the algorithm programs in a controlled environment, monitoring the program execution and error reports, isolation of the programs via virtualization of the computing environment, and batch processing for large input or heavy algorithms.

Another issue is the sustainability of the software. For now, source codes present on IPOL are not maintained, but we think that our software guidelines constitute a good way to limit the dependencies only to the programming language. The one guarantee that we provide for the readers is that when the article was accepted and integrated to IPOL, the software was correct, readable and usable. If some major changes in the computing environment occur, the complete algorithm description provided in the article is also a precious tool which allows in itself to re-implement the algorithm. Consolidation of all the published codes into a unified, refined and maintained code base could be a continuation of the effort, but so far we postpone this task and it may be better handled outside of the journal. Restricting the authors to a few programming languages and software libraries is a way to tackle software obsolescence by reducing the dependencies. Another strategy would be to promote standard and stable programming interfaces, but unfortunately such APIs are still missing in the field of image processing.

C. Extension to Other Domains

Our journal was designed and tuned for image processing, and the complete system needs to be adapted for it to be usable for other research fields. In particular, image processing is well adapted to a web interface because images have always been integrated as a native component of the web technology stack, whose native interface is the flat visual computer display, a natural interface for images. Moreover, most image-related algorithms are faster than those of other computational science fields, such as fluid mechanics simulation or financial analysis. This is important because a web demo interface requires an interactive user experience. With longer computations, we would need a different interface, such as a batch processing service where tasks would be submitted (over the Web) and results received later, once their computation is completed.

The extension of IPOL to other domains is likely to be feasible but not straightforward. For instance, we are currently working on designing an equivalent journal for sound processing, and despite the fact that the themes of image and sound processing are rather close, computation time, data volume and visualization of the results can become a serious issue for the demos. The choice of programming language is also a tricky question, which strongly depends on the community.

Finally, we must take into account that some algorithms or some domains are not a good fit to the IPOL model as it is. One of the main limitations is that full code review is only feasible if it is possible to isolate an algorithm in a relatively small code base. Also, IPOL and similar journals are not the place for research about software programming techniques and computer science as such, because in IPOL the software is the support of the research, not its subject-matter: IPOL publishes algorithms, not code. The codes included in the articles provide a detailed description of the algorithms and a mean to test and reuse these algorithms.

VI. Conclusion

In this article, we have described and discussed IPOL (Image Processing On Line), a research journal which publishes algorithms with a complete software implementation. Contrary to classic journals, an IPOL article combines not only a thorough description of the algorithm, but also a rigorous and certified implementation and an online demo. By considering software a primary product of the research activity, and systematically submitting it to review and publishing process, we observe that more software is released by the researchers, with its quality verified by external reviewers. Of course, IPOL does not solve all the issues arising in research software, but still provides a unified framework in which the software is an integral part of any publication. Moreover, by collecting the tests performed online on each of these codes, we gather some testimony of its numerical performance and robustness, which in turn can help researchers to improve their algorithms.

Acknowledgments

This work was partially supported by the Office of Naval Research under grant N00014-97-1-0839, the European Research Council, advanced grant ERC-2009-AdG "Twelve Labours of Image Processing," and the National Science Foundation under Award No. DMS-1004694. The authors would like to thank the anonymous reviewers for their encouragements and suggestions.

References

[1] J. Buckheit and D. Donoho, "WaveLab and Reproducible Research," Technical report 474, Department of Statistics, Stanford University, 1995. http://www-stat.stanford.edu/~donoho/Reports/1995/wavelab.pdf

[2] C. Cannam *et al.*, "Sound Software: Towards Software Reuse in Audio and Music Research," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

[3] J. Claerbout and M. Karrenbach, "Electronic documents give reproducible research a new meaning," in *Proceedings of the 62nd Annual International Meeting of the Society of Exploration Geophysics*, 1992.

[4] B. Kernighan and P. Plauger, *The Elements of Programming Style*, McGraw-Hill, 2nd edition, 1978. ISBN:0070342075

[5] B. Kernighan and R. Pike, *The Practice of Programming*, Addison-Wesley, 1999. ISBN:020161586X

[6] X. Li, "Image Denoising, Past, Present and Future," in *Image Restoration: Fundamentals and Advances*, B. Gunturk and X. Li, Eds., CRC Press, 2012. ISBN:1439869553

[7] N. Limare and J.-M. Morel, "The IPOL Initiative: Publishing and Testing Algorithms on Line for Reproducible Research in Image Processing," in *Proceedings of the International Conference on Computational Science (ICCS)*, 2011. doi:10.1016/j.procs.2011.04.075

[8] N. Limare, “Reproducible Research, Software Quality, Online Interfaces and Publishing for Image Processing.” Ph.D. thesis, CMLA, ENS Cachan, 2012.
[9] S. McConnell, *Code Complete*, Microsoft Press, 2nd edition, 2004. ISBN:0735619670
[10] Z. Merali, “Computational science: . . . Error,” *Nature*, 2010. doi:10.1038/467775a
[11] P. Seibel, Ed., *Coders at Work: Reflections on the Craft of Programming*, Apress, 2009. ISBN:1430219483, http://www.codersatwork.com/
[12] S. Sonnenburg *et al.*, “The Need for Open Source Software in Machine Learning,” *Journal of Machine Learning Research*, 2007.
[13] P. Vandewalle *et al.*, “Reproducible research in signal processing — What, why and how?”, *IEEE Signal Processing Magazine*, 2009. doi:10.1109/MSP.2009.932122
[14] *Image Processing On Line (IPOL)*, ISSN:2105-1232, doi:10.5201/ipol http://www.ipol.im/
[15] *The Insight Journal*, http://www.insight-journal.org/
[16] *Journal of Machine Learning Research (JMLR)*, ISSN:1533-7928, http://jmlr.csail.mit.edu/
[17] *Machine Learning Open Source Software (MLOSS)*, http://mloss.org/
[18] *SIAM Journal on Imaging Sciences (SIIMS)*, ISSN:1936-4954, http://www.siam.org/journals/siims.php