Development Efforts for Reproducible Research: Platform, Library and Editorial Investment

Miguel Colom, José Armando Hernández, Bertrand Kerautret, Benjamin Perret

Development Efforts for Reproducible Research: Platform, Library and Editorial Investment*

Miguel Colom[1][0000−0003−2636−0656], José Armando Hernández[1][0000−0002−6692−8640], Bertrand Kerautret[2][0000−0001−8418−2558], and Benjamin Perret[3]

[1] Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, F-91190 Gif-sur-Yvette, France miguel.colom-barco@ens-paris-saclay.fr
[2] Univ Lyon, Univ Lyon 2, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69676 Bron, France bertrand.kerautret@univ-lyon2.fr
[3] LIGM, Univ Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallee, France benjamin.perret@esiee.fr

Abstract. Reproducible research in pattern recognition can be viewed from a number of angles, including code execution, platforms that promote reproducibility, code sharing, or the release of libraries providing access to relevant algorithms in the corresponding disciplines. In this work, after recalling the motivation and classic definitions of reproducible research, we propose an updated overview of the main platforms that might be used for reproducible research. We then review the different libraries that are commonly used by the pattern recognition, computer vision, imaging and geometry processing communities, and we share our experience of developing a research library. In the third part, new advanced editorial investments will be presented, such as the IPOL journal or other IPOL-inspired new initiatives like OVD-SaaS.

1 Introduction

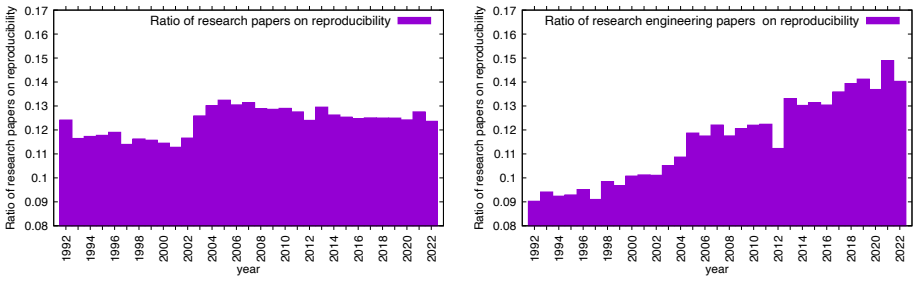
In general, research publications first highlight new theoretical or methodological advances related to scientific problems, while reproducibility is rather considered as a secondary point. While these academic distinctions and measures seem natural from an innovation perspective, the emphasis on reproducibility should also be a key point in avoiding the credibility crisis denounced by Donoho [1]. Reproducibility also has an impact on long-term research, such as simplifying comparisons to make research results more meaningful.

The Figure 1 shows the evolution of the ratio of publications mentioning the words “Reproducible” or “Reproducibility” in general and engineering topics (graphics (a) and (b) respectively). During the last 30 years, the mention of the

* This research was made possible by support from the French National Research Agency, in the framework of the projects WoodSeer, ANR-19-CE10-011, ULTRA-LEARN, ANR-20-CE23-0019, and by the SESAME's OVD-SaaS project from Région Île de France and BPI France, and Ministry of Science, Technology and Innovation of Colombia (Minciencias), call 885 of 2020.

reproducible research (RR) topic remains almost constant in the general domains while it steadily increased in the engineering domain. This increasing interest is also visible through the development of platforms for reproducible research [2]. For instance, since the publication date of this latter review, new major platform appeared like the *ReproducedPapers.org* platform [3] that allows researchers to share reproduction experience on papers especially in the machine learning field. Such a new platform reaches also educational purpose with, for instance, the integration of reproducibility into fairness, accountability, confidentiality and transparency in artificial intelligence [4]. Another example of new advance is the development of the *ReproServer* [5] that follows the previous ReproZip tools [6]. ReproServer is an open source web application allowing to reproduce experiments from a web browser and based on the ReproZip tool. The REusable ANAlyses system called REANA [7] is also another example started after the review on reproducible research platform [2]. This initiatives answer to the need of the reuse, re-validation, and re-interpretation of research works.

The platform evolution is not the only point that contributes to RR, since the development of libraries is also a meaningful ingredient if we consider the algorithm implementation point of view. It also contributes to the diffusion of scientific contents and also simplify code review in a editorial process where not only the classical paper is reviewed but also the associated source code. To put these efforts in perspective with reproducible research, the following section reviews new platforms that have been recently proposed. Then, the investment in libraries will be detailed by several examples taken from the pattern recognition domain and the experience of creating a new library will be shared by the Higra library creator (section 3). Finally, in section 4, the investment on advanced editorial initiative will be detailed with the recent progress around IPOL journal and a related new service oriented project OVDSaaS useful for the deployment of reproducible industrial applications.



(a) (b)

Fig. 1. Evolution of the Reproducible Research ratio of publications from the general (a) and engineering (b) topics. The data were extracted from ScienceDirect [5]database on August 29, 2022.

2 Reproducible Research Platform Updates

In the previous work [2], we have classified the major reproducible research platforms into three broad categories. The first group is the online execution platforms. It includes platforms like *RunMyCode*, *CodeOcean* or *Jupyter*, all of which allow to directly run code on a distant web server. The other group called dissemination platforms, is more focused on hosting source code archive package with a referencing system. In this category, we can mention *RunMyCode* which also proposes to host source code, *DataHub* or *ResearchCompendia*. The last category is the peer-reviewed journals, composed of three main journals: *IPOL*, *ReScience* and *Insight Tool Kit*.

Proof of the growing interest in reproducible research, new platforms have appeared since the previous publication and several of them such as *DagsHub*, *Paper with code*, *Replicate* or *Hugging Face Spaces* are now referenced from *arXiv* open access platforms.

ReproducedPapers.org [3] ***- Dissemination Platforms -***

This platform was introduced in the context of machine learning reproducibility and teaching activities at a master level degree. The main ideas of the platform are the following. First, a researcher suggests on the platform a paper to be reproduced. Then, from the proposed paper, different types of contributions can be submitted: (i) a replication including re-implementation from scratch, (ii) a reproduction where existing code is evaluated, (iii) hyper-parameters check including sensitivity of parameters, (iv) new data to test the result on other contexts, (v) new algorithm variant, (vi) new code variant including improved implementation, (vii) ablation study. A resulting pdf paper or web page describing the new contribution is then made freely available to the community, including source code (for (i) and (vi)) or new data (for (iv)). When the submission is proposed, a review is conducted to ensure that the main contributions match those that have been announced.

Figure 2 presents the evolution of the number of papers proposed for reproduction as well as the number of reproduction published since the platform was created. Both the research papers proposed for reproduction and the reproduction attempts are increasing. In average, there are around two reproductions per proposed article. This trend looks positive for the future of the platform.

Replicability.graphics [8] ***- Dissemination Platforms -***

The aim of this platform focused on computer graphics is to evaluate the replicability of the papers published at the SIGGRAPH conference. The evaluation is presented in the form of a light review describing the difficulty of the reproduction. This evaluation can result from the direct code execution, when available,

[5] `https://www.sciencedirect.com`

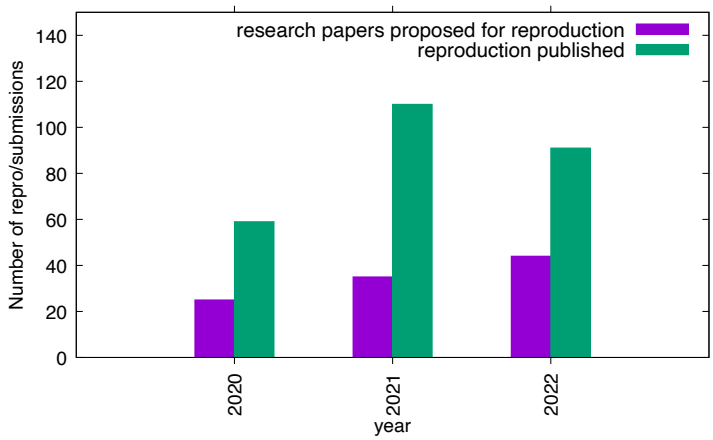


Fig. 2. Evolution of the activity of the `https://reproducedPapers.org` platform.

or from the re-implementation of the article. The ease of re-implementation is usually evaluated by the platform reviewers who have a strong experience in the computer graphics domain. In total, 454 papers were reviewed on the platform, covering SIGGRAPH events from 2014 to 2021, with 192 papers proposing code and 146 papers were evaluated as replicable. The remaining papers (25% with 116 papers) were evaluated difficult to replicate.

PapersWithCode [9] ***- Dissemination Platforms -***

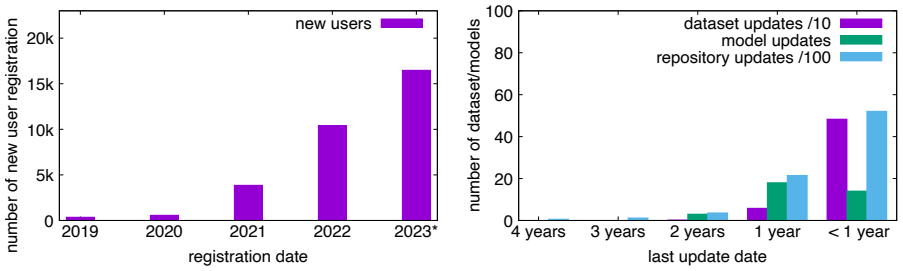
Introduced by *Meta* in 2018, this platform references papers associated with source code theoretically enabling the replication of the paper. It can be considered as a dissemination platform since only the paper and the source code are referenced. The platform can be useful to researchers interested to reuse the related proposed methods. However, unlike the previous two platforms, no details are provided on the potential difficulty of reproduction or conformity between source code and algorithms. With a total of more than 126 000 papers[6], it covers various domains with different portals like Machine Learning (76.2%), Computer Science (8.8%), Physics (6,0%), Astronomy (3.3%), Mathematics (3.3%), Statistics (2.4%). Papers can also be associated with datasets, methods and evaluation tables. Users can submit a paper and its implementation from a free user account.

Dagshub.com ***- Dissemination Platforms -***

This platform is specialized in the machine learning domain with an experience comparable to *GitHub* but integrating specialized tools for the visualization of models and machine learning pipelines. It allows users to manage experiments,retrieve the best data or parameters, and is therefore useful for optimizing

[6] Data extracted from `https://portal.paperswithcode.com` on 15 May 2023

results and sharing experiences with other users. It contributes to facilitate the reproducibility, in particular with an easy configuration of automation pipelines running on *GitHub actions* or other web-hooks. The platform is free for public and limited private repositories and chargeable for more intensive use with private cloud or enterprise access. In 2023, the platform was used by 23,000 data scientists and 400 organizations. The Figure 3 shows the evolution of user registration (graphic (a)) and update activities over the last four years with the focus on repositories, datasets or models (graphic (b)). These measures highlight the rapid success of the platform which is experiencing positive growth and are a promising sign of future user uptake.



(a) (b)

Fig. 3. Evolution of user registration (a) and update activities (b) of the dagshub.com platform. The graphic (b) displays update over time of datasets, models, or repositories. The data were obtained from the page listing the different ressources from the dagshub.com explore page. *For the 2023 year, the number of user registrations was interpolated by taking the double of registrations of the 6 first months of 2023.

***HuggingFace.co* - *Online Dissemination & Execution Platforms* -**

Originally launched in 2016 as a company developing a chat bot application for teenagers, the Hugging Face Hub can now be seen as straddling the line between dissemination and online execution platforms. Based on a *git* repository whose main structure is comparable to that of *GitHub*, it focuses on models and datasets, with a section called *Space* containing a playground allowing users to run the model directly online. The platform integrates various models with custom configurations and weights fine tuned on specific training sets. The dissemination part is advertised as forever free while the computational ressources is limited according to the subscription chosen by the user (free for two basic CPUs and chargeable for more advanced GPUs).

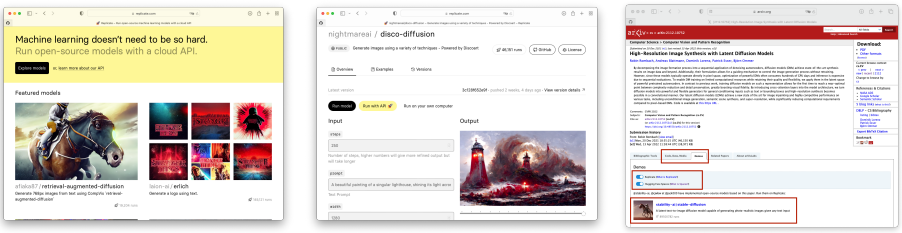
Reana [7] **- *Online Execution Platforms* -**

Oriented toward reproducible research and data analysis, this platform is designed to create workflows by using external sources (like *GitHub*, *GitLab*). This platform can play a main role in reproducible research, however it is actually

reserved to CERN (European Organization for Nuclear Research) users which limits its potential impact.

ReproZip [5] **- *Online Execution Platforms* -**

At the origin *ReproZip* was first an open source tool designed to bundle all the necessary contents to reproduce research results. Then a *ReproServer* was associated to *ReproZip* enabling to extract and run the resulting program. Actually as mentioned by the authors the full construction of the platform is in progress. The main idea of such a platform appears promising for the future. Unlike the previous *Reana* platform, the current project is more oriented towards open access and unrestricted use which could be an important factor for its future growth.



(a) (b) (c)

Fig. 4. Illustration of the Replicate.com: (a) overview, (b) parameter setting before online execution and (c) link appearing from the *arXiv* platform highlighted in red.

Replicate.com **- *Online Execution Platforms* -**

The main motivation of this platform is to apply machine learning to real world problem without complex code installation. It can be considered as mainly a machine learning code replication platform. User can construct and upload their package allowing the reproduction of their research results based on their own machine learning models. The platform proposes three ways to reproduce results: by using a direct online execution process, by running a platform API or by running the code directly on the user computer. The Figure 4 illustrates this platform from the front page (image (a)), the main page allowing to set the parameters of a particular demonstration (image (b)) and the *arXiv* referencing this platform (image (c)). Unfortunately this platform is not free (users can try it for free with a limited processor time allocation).

The Table 1 shows the recent platform comparison using the criteria defined in [2]. In order to complete the comparison context, the result of the platform best covering the different criteria is reported on the last line of the table with the IPOL platform. This platform still leads the ranking with eight validated

criteria and is now followed by *Reproduced Papers* and *Replicate.com*, which satisfied seven criteria.

3 Reproducible Research Through Libraries

In addition to RR platforms, efforts to contribute to reproducible research are also visible through the development of libraries offering implementations of algorithms and tools. To give an overview of the library development types, we first review different initiatives in the discrete geometry and mathematical morphology community before detailing an example of an author initiative with the development of the Higra library (Hierarchical Graph Analysis) which could help new researchers to understand and anticipate the important steps to create, publish and maintain a new library.

3.1 Library Experiences from Pattern Recognition, Image and Geometry Domains

The Table 2 lists some of the major libraries in the domains of pattern recognition, image and geometry processing. A first group gathers libraries initiated and financed by private companies. The ITK medical imaging library, created and still managed by Kitware, is an example. On another popular topic of computer vision, the OpenCV library presents a mean of 60 contributing authors by year. As for the PCL library, the private company Willow Garage supports the development. As mentioned in our previous work, insides the development of the ITK library the *Insight Journal* aims to contribute to reproducible research by publishing algorithm descriptions together with its implementation.

Another group of libraries is composed of academic initiatives, including also the open source community. The older referenced in Table 2 is the CGal library created with the help of European project funding. The economic model was then supported by the commercialization of the research results obtained through the company GeometryFactory thus ensuring the continuation of the development of the library while contributing to the promotion of academic research. Initiated from individual initiative, the libraries CImg (David Tschumperlé, GREYC), Geogram (Bruno Levy, LORIA), Olena (Thierry Geraud, LRDE), Tulip (David Auber, LaBRI) and Vigra (Ullrich Koethe, HCI) were also contributing in helping the diffusion and reproduction of research results. Like other libraries such as OpenMVG, TTK or Higra, the DGtal library was also created from a French initiative gathering five laboratories.

If we measure the development of certain libraries over time, we can easily highlight the libraries that have been supported by or associated with private companies, as shown by the CGal library with a continuous investment during 25 years (Figure 5 (a)). On the contrary, the OpenMVG library shows a slowdown since the moment the library started to be based on the open source development mode only (Figure 5 (a)). The impact of the development of such a library

Platform	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
Reproduced Papers	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	–
Replicability.graphics	✓	✓	✓	✗	✓	✗	✗	✗	–	✗	–
Papers With Code	✓	✓	✓	✗	✓	✓	✗	✗	–	✗	–
Dagshub.com	(A)	✓	✓	✗	✓	✗	✓	✓	✗	✗	–
HuggingFace.co	(A*)	✓	✓	✗	✓	✗	✓	✓	✗	✓	–
reana	✗	✗	✓	–	✗	✓	✓	✓	–	✗	–
ReproZip	–	–	✓	–	✓	✓	–	–	–	✗	–
Replicate.com	(A*)	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗
IPOL	(B)	(C)	✓	✓	✓	✓	✓	✓	✓	✓	✗

Criteria:
(1) Free to use
(2) No mandatory registration
(3) Several programming languages allowed
(4) Peer-reviewed code and data
(5) Easy to use by the non-expert
(6) General scope
(7) Possibility to upload user data
(8) Interaction through a web interface
(9) Access to a public and persistent archive of experiments
(10) Design of automatic demonstration from textual description or visual tool
(11) Allow to modify the source code before execution

Legend:
– Not Applicable.
(A) Partially free (paid for non entreprise or restricted private projects).
(A*) Limited time free demonstration or basic CPU (then paid by time usage of CPU/GPU usage).
(B) True for demonstrations using a sample learning dataset. To use the platform as a service, the user needs to be connected with a role authorizing this usage.
(C) The demonstrations are free to use up to some limits (say, size of the data or computation time), but industrial use of demonstrations and applications requires payment.

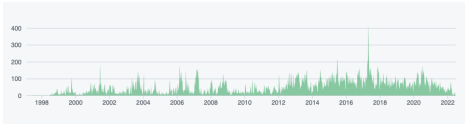
Table 1. Recent platform comparisons using the eleven criteria introduced in previous work [2].

Library	ref	domain	langage	version	#auth.	date	funding
OpenCV	[10]	Comp. Vision	C++	4.5.5	1,383	1999	Willow Garage
ITK	[11]	Image Processing	C++/Pyt.	5.2.1	265	2000	Kitware
PCL	[12]	Point clouds	C++	1.12.1	464	2010	Willow Garage
CGal	[13]	Geometry proc.	C++	5.4	123	1996	Acad./GeometryFactory
CImg	[14]	Image processing	C++	3.1.2	72	1999	Acad.
Geogram	[15]	Geometric algorith.	C++	1.7.8	7	1998	Acad./INRIA/ERC
Olena	[16]	Image processing	C++/Pyt.	2.1	50	2001	Acad. / Project
Tulip	[17]	huge graph visualiz	C++/Pyt.	5.6.2	9	2001	Acad./private
Vigra	[18]	Comp. Vision	C++	1.11	50	2008	Acad.
DGtal	[19]	Digital geometry	C++/Pyt.	1.2	27	2011	Acad. / Project
OpenMVG	[20]	Mult. View Geom.	C++	2.0	86	2013	Acad./Mikros/Foxel
TTK	[21]	Topology ToolKit	C++	1.0	36	2017	Acad. / Project
Higra	[22]	Graph analysis	C++/Pyt.	0.6.5	4	2018	Acad. / Project

Table 2. Example of librairies related to the pattern recognition, image and geometry domains. The three first libraries are mainly supported from private company including open-source community (highlighted in gray inside horizontal lines and light gray above the dashed horizontal line for mixt of private/academic).

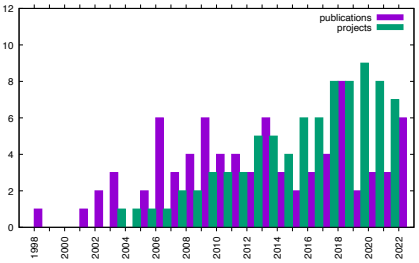
on the research world can be measured by citations in research publications or the number of research projects that rely on the library. Graph (c) in Figure 5 shows one such example measured on the Geogram library. The number of research papers referencing the library appears to be stable over time, while the number of project mentioning is increasing over the years.

This rapid overview shows that outside the benefit for reproducibility, the library development has impact on sharing results both on research publications and projects. In order to give the main steps of library development, we show an experience feedback on the main steps of the Higra library.



(a) Commits measure from 25 years of the CGal library. (source GitHub accessed in 28 June 2022)

(b) Commits measure from 10 years of the OpenMVG library.



(c) Example of publications number mentionning Geogram library.

Fig. 5. Example of developer investissement from two libraries (a,b) and evolution of publications and project exploiting the Geogram library (c).

3.2 Higra Library Development Feedback

Higra – Hierarchical Graph Analysis – is a C++/Python library for efficient sparse graph analysis with a special focus on hierarchical methods: construction of hierarchical representations (agglomerative clustering, mathematical morphology hierarchies, etc.), the analysis and processing of such representations (filtering, clustering, characterization, etc.), and their assessment. The development of Higra started in 2018 and is still a quite young project of moderate size. It is now extensively used for research and teaching purpose in our laboratory and is also used by researchers and practitioners in several other places in the world. The purpose of this section is to explain the different steps of this project in order to help researchers interested in creating their own libraries to enhance and facilitate the reproducibility of their work.

Deciding and starting the project

Project prequel Before starting the Higra project, our researches on hierarchical graph processing were essentially supported by two mostly internal libraries: 1) a pure C library called SM (*Saliency Map*) which was fast but lacked flexibility; and 2) a pure Python library called HiPy (*Hierachies in Python*) which was flexible but slow. Although both libraries are available online, neither was really designed for distribution to an external audience: there was mostly no documentation, no tests, and, in a general way, no project management. This approach also started to be detrimental for the research activity of our own group: at some point, we had about a dozen of forked versions of the SM library on our computation server, developed by various researchers of our team, without any documentation on the modifications made. In practical terms, this resulted in lost changes, wasted effort, and overall reduced reusability and reproducibility of our work.

Motivations The aim of the Higra project was to overcome the limitations and solve the problems described in the previous section. The idea was to have a unified place for our developments that would enable us to better integrate all the contributions of the team and to attract external users and developers. This goal can only be achieved with a good level of project management which includes notably using a source version control system with a versioning scheme, writing tests to ensure correctness and non regression, and writing an extensive documentation. Another goal of the new library was to be easy to install and to naturally integrate into the rich Python ecosystem for machine learning and computer vision, which essentially meant supporting Numpy arrays [23] and the ability to be installed by the standard package-management system for python *pip*.

All these would enable us to achieve a more efficient use of our resources, a better dissemination of our contributions, and ease the reproducibility of our research.

Getting started The start of such project can be intimidating and requires some planning as several choices have to be made regarding 1) the project organization (code and documentation hosting, build-system, CI/CD pipelines) and 2) the technical development aspects (languages, external dependencies, software architecture). Many researchers are not familiar with all these aspects that are not part of their main area of expertise. Fortunately, today, we can rely on the many tools developed by the open source community and the many examples available online to make wise choices and implement them. We recommend to not neglect this initial planning and configuration phase, even if it takes a little time, because it will save a lot more time in the following phases of the project and help to deliver features quickly.

In the following, we describe the most important choices made for the library Higra and the motivations behind them. Of course, there are no absolute best choices and, in the end, this depends on many factors such as the objective of the projects and the developer preferences.

Technical choices of the Library Structures

Languages, architecture and dependencies We opted for a relatively classical organization for a data analysis project in Python with a front-end, comprising essentially high-level functions, written in Python and a core, with data-structures and critical algorithms, written in C++. We decided that the C++ core should also be usable without Python which would allow us to use these functions in another context, for example to build web demos using Emscripten[7] which allows compiling C/C++ code to javascript. This is quite different from many Python packages such as Numpy, Scikit-learn or Scipy whose back-ends rely heavily on the C-Python API. One other possibility would have been to write a pure Python package with Numba [24] which can perform just in time compilation of a subset of the Python language to improve the execution time. Numba manages to obtain excellent performances, but it complicates debugging and was, at the time we started the project, quite an experimental project.

The bridge between C++ and Python is made thanks to the Pybind11 library [25] which is a header only C++ library which enables to easily create Python module from C++ and export C++ functions in this module. Pybind11 is a very popular library used in many large projects. In order to integrate with Numpy arrays we chose to rely on the XTensor[8] C++ header only library which aims at providing a C++ substitution of Numpy with a seamless integration with Numpy arrays (C++ tensors can be converted seamlessly into Numpy arrays and conversely).

Build tools, unit testing The main build system of the project is CMake which is frequently used for C++ projects. It is available on most systems and can

[7] `https://emscripten.org/`
[8] `https://github.com/xtensor-stack/xtensor`

be installed through many package managers, including *pip* (even if it is not a Python package). Such tools become nearly mandatory when one wants to compile cross-platform projects in a reliable and reproducible way: they help to deal with platform specific options and with dependency management. We have never heard about a pleasant build system and CMake is no exception, but it gets the job done and as it is used in many projects it is easy to find examples covering many use cases. A secondary build system, *setuptools*, is used to generate Python wheels (python packages which can be installed with *pip*). This consists of a single Python script which is merely a wrapper around the CMake project.

The unit tests are written in C++ using the library Catch2[9] and in Python with the standard *unittest* package. Using a library such as Catch2 for unit tests provides many benefit such as easy test generations for template functions and types, easy logging, selective test execution and debugging, and are supported by advanced IDEs (execute/debug specific tests directly from the GUI).

Hosting and CI/CD pipelines We naturally opted for *Git* as source version control which is very well adapted for open source development and decided to use *Github* for hosting. The documentation is hosted by *Read the Docs*[10] which is commonly used by python packages but can also be used with C++. Python wheels are hosted by PyPi[11] which is the standard python package repository used by *pip*.

Continuous integration (CI) pipelines are managed by Github. The idea is that any pull request made on the repository first goes through automated testing on multiple platforms before being merged into the main branch. Our CI pipelines cover several versions of Python and 3 platforms: Linux, MacOS, and Windows. Each pipeline validates that the project can be built and that unit tests pass in a fresh new environment. Setting the CI pipelines is quite difficult at first if the user is not familiar with them, but it is really worth it by drastically reducing the risk of regressions in a cross-platform environment. The CI pipeline also includes a coverage test which measures how much of the code is really covered by the unit tests, the results are hosted by Codecov[12] and an error will be raised if the overall coverage is decreased by a pull request.

The CI pipelines are complemented by Continuous Delivery (CD) pipelines, which automatically build new Python wheels and send them to PyPI when a new version tag is pushed on the repository. CD pipelines are similar to CI pipelines, but are more constrained: they must provide release/optimized code which can be distributed. This is in particular a constraint for Linux environment where PyPi requires using a specific Linux Image, called *manylinux*, with

[9] `https://github.com/catchorg/Catch2`
[10] `https://readthedocs.org/`
[11] `https://pypi.org/`
[12] `https://about.codecov.io/`

a limited set of libraries that are expected to be found in any Linux environment. Finally, the documentation host is also part of the CD pipeline as it will automatically rebuild and publish a new version of the documentation when a new version tag is pushed.

Strategy and Roadmap From the previous technical choices describing the project, several key strategies were put in place.

Adding new features Our strategy was to ship new features as fast as possible to quickly get users feedback. In practice, coding represents only about 25% of the time required to propose a new feature. The rest of the work corresponds to an investment of about 35% for writing tests and debugging, 30% for documentation and examples (notebooks), and 10% for the management of the project (updating libraries, updating CI/CD...).

Getting users Initially, the library project was shared with a few close users, which had the advantage of quick feedback, numerous bug fixes, and fairly frequent breaking changes. After this start-up period, we started to use the library extensively in courses, tutorial and research projects. We also took the habit to systematically publish companion notebooks based on the library to advertise and enhance the reproducibility of our research papers. Today the library is used in several labs, by researchers from the community and from other communities, essentially for some image processing methods that do not exist in more standard packages. In the future, we hope that some of these new users will contribute to the development of the library itself.

Global feedback The creation and the diffusion of this library is a very positive experience. In the same way as reported in the previous section, the library greatly eased several research projects within and outside our laboratory, as it saved a lot of time for interns and PhD students. It also greatly facilitates the dissemination of code and the reproducibility of our research papers.

4 Advanced Editorial Efforts

In this section, we discuss two RR platforms, which are managed by some of the authors of this article, showing the last editorial efforts to improve reproducible publications. These platforms do not consider the article itself as the sole output of the research work, but also include source code, rich interactive interfaces for demonstration, or even advanced system which go beyond the concept of simple and isolated demonstrators.

The two platforms we review are: a mature journal, IPOL, and a starting project which can be considered as its spin-off for industrial applications on machine learning (ML) algorithms, OVD-SaaS.

4.1 Improvements in the IPOL journal

IPOL is a research journal on reproducible algorithms, focusing on their mathematical details [26]. It started as an image-processing journal, but soon it added other data types, such as video or audio, among others, as well as other applications, including remote sensing [27] or even biomedical [28], among others.

IPOL has continued to advance in making the system more adapted to the needs of authors and demo editors. Here we focus on three aspects and present what was done to improve the journal: the possibility to use a git repository to develop the code and see the results immediately in the online demonstration website, the use of containers for better reproducibility and maintenance, and new datatypes such as interactive maps for remote sensing applications.

Fast development after integrating git One request made to IPOL by authors was the possibility of fast code edition. Indeed, the IPOL editor or advanced users usually had to package their sources in a single file and upload it to some server; the IPOL's demo system would then download it, compile it, and run the updated version. The editors and authors felt that this was too time-consuming and preferred a solution more suited to their rapid development needs. Ideally, they wanted to use their own git repositories (for example, hosted in Github or Gitlab) and that the system fetched the last changes before each execution. This is now implemented in the system.

This brings up the question of reproducibility, given that, as authors are the owners of their repositories, they can change the published code at any moment. The solution adopted by IPOL is to let authors use their own repository, and at the moment of the publication, their repository is forked in a repository owned by IPOL. The code is then frozen at the particular revision which passed the peer review.

This is an example on how the source code is now specified in the demo description:

```
"build": {
    "url": "git@github.com:mlbriefs/88.git",
    "rev": "42252c0a84771c9abb141d0eacb6e9d54f44e9e5",
    "dockerfile": ".ipol/Dockerfile"
}
```

It specifies the git repository, a particular revision, and a Dockerfile. The use of Docker container in the demos is explained in the following section.

Execution in Docker containers When IPOL was started in 2009, C and C++ were some of the few accepted languages, along MATLAB. Authors were given strict coding guidelines and were limited to use a small subset of libraries known for their stability and backward compatibility. Indeed, most of the demos of that time are still running without major problems.

However, it was complicated to run MATLAB with a unique version of the framework, especially when authors actually used different versions, not necessarily the one accepted by IPOL. At the same time, Python gained a wide popularity in scientific research, and with the rise of machine learning and associated frameworks and libraries (PyTorch, TensorFlow, and others) it became the *de facto* standard.

Each MATLAB program could be designed to run on its own MATLAB version, and Python's virtual environments were not enough to ensure long term reproducibility. Indeed, it could happen that after a major update of the Debian distribution of the IPOL servers, some packages in PyPI were not yet available, among other causes that prevented reconstructing the same environment.

The solution that we adopted was to run the demos inside Docker containers. To make them reproducible, we maintain the complete instructions to re-create the container. Now all new IPOL demos are containerized, and we actively port the old ones to the new execution environment, now well-defined. We also expect that the use of containers will make it easier to run demos on the cloud in the future, combined with Infrastructure as Code techniques.

Interactive map GeoJSON demos The opening of IPOL to more diverse research fields required to improve its underlying infrastructure to support new data types, especially in the web interface. In particular, IPOL has already published several articles on remote sensing, along with the associated demos [27]. One request from users and authors was the possibility to draw one or more polygonal regions on a map and to save them in the standard GeoJSON format (standardized by the IETF as RFC 7946).

A good library to render maps on websites was started by the Mapbox company in 2010, initially under a BDS license. On December 2021, the BSD license used in Mapbox GL JS was changed to a proprietary one. The community forked the project and started a free-software alternative, Maplibre, under BSD license again. Both projects have a very similar API. In 2023 we implemented the interactive map feature in IPOL, with Maplibre as the preferred option, to keep the project free software, although Mapbox is technically viable. Figure 6 shows a detail of an IPOL demo with this new feature.

The latitude and longitude of the polygon vertices is encoded in the GeoJSON response without further processing needed from the IPOL's side. It is also simple to obtain both the pixel locations and their corresponding longitude and latitude.

Perpetual archiving Software as a fundamental artifact of the research must be perpetually preserved. Especially in the case of IPOL publication, where we consider the article, associated data, and source code as a whole.

Until 2020, IPOL simply stored in their own infrastructure the source code package which was accepted by the editors. The compromise of IPOL is to make the

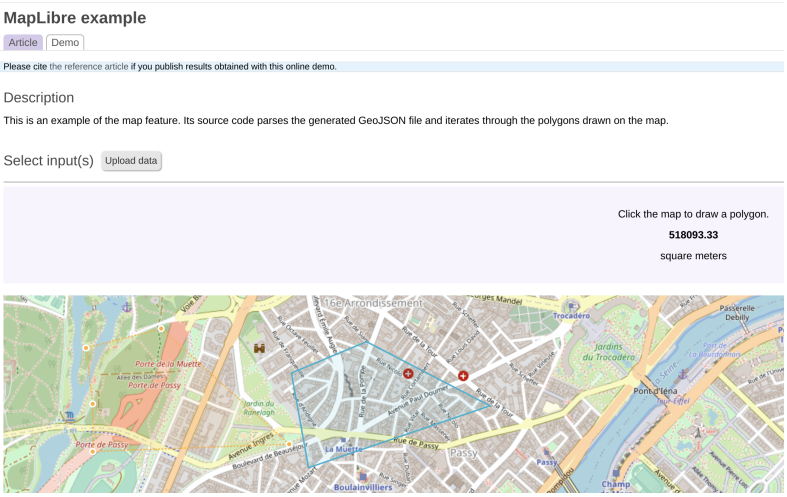


Fig. 6. A detail of an IPOL demo with a new feature: interactive map. In this example, one region has been already completed, and the second is being drawn. The feature is based on Maplibre and it allows to draw one or more regions as polygons on the map, and return them encoded in GeoJSON format.

sources available forever, or at least for all the lifetime of the journal. However, the role of IPOL is not being a perpetual archive of source code. Note that, even large source code repositories such as Github, should be considered development tools, without that perpetual archiving commitment.

Fortunately, the Software Heritage project [29], supported by UNESCO, is devoted to the long-term preservation of software artifacts. Moreover, Software Heritage is able to point to the artifacts at different levels, such as package, git project, a particular commit, or even a specific line of the source code, thanks to the SWHID (SoftWare Hash IDentifier). The SWHID are intrinsic, persistent, and decentralized identifiers.

Since June 2020, IPOL systematically submits the source code of all published articles to Software Heritage[13], in an alliance in favor for Open Science and reproducibility.

4.2 OVD-SaaS, a Spin-off of IPOL for Industrial Applications

OVD-SaaS (Online Verifiable Datascience Software as a Service) is a new project at Centre Borelli which can be considered as the extension of IPOL to industrial applications. This project takes advantage of the 12 years of experience with the current computing infrastructure of IPOL to develop a new general platform for artificial intelligence (AI), including machine learning (ML) applications.

[13] `https://www.softwareheritage.org/2020/06/11/ipol-and-swh/?lang=es`

OVD-SaaS aims at integrating diverse domains such as academic research, scientific publishers, and industry through the seamless end-to-end deployment of scalable, secure, reproducible industrial applications. This is expected to highlight the value of ML/AI research applied to different scientific fields, as for example finance, health, transportation, or commerce.

The most important features are:

- Easy comparison of algorithms on user data to establish the state of the art on diverse applications and datasets with a high level of confidentiality.
- A step to certification (badges) or at least granting a label of quality to algorithms, thus allowing for a better recognition and reputation for authors.
- High degree of specialization and standardization in peer-reviews of code for ML/AI scientific journals and conferences.
- Use of advanced cloud computing platform for end-to-end deployment of agile ML-Ops applications, from their publication in the scientific journal to their implementation at the operational level.
- Code provenance traceable, reproducible, citable (using SWHIDs) functionality by chaining algorithms to develop complete pipelines that provide support solutions to business problems across application domains.

The two main possible business models are:

- Sharing the responsibilities in reproducibility of scientific results, facilitating contracts between research laboratories, faculties and companies. This allows to develop specific algorithms which are relevant to their business or to benefit from the experience of the laboratory when jointly developing a complete pipeline.
- Spin-off projects (startups or incubated within larger companies) which would design tailor-made software solutions for SMEs based on particular chains of algorithms optimized for specific businesses. It can be understood as an accelerator for the development of business ideas in startups by facilitating agile cycles providing proofs of concept, sandbox, and minimum viable products (MVP).

OVD-SaaS Applications The OVD-SaaS application differs from the IPOL demo system in that it allows for applications whose lifetime is much larger compared to the short execution of an online demo. In the case of an online demonstration, the input data is loaded, the algorithm is run, the results are displayed, and the execution of the demonstration finishes. However, in the case of OVD-SaaS applications, they run continuously, listening for events to wake up and processes more data if needed. We shall generically call these new long-lived processes *Applications*. The architecture of the OVD-SaaS system has not been written from scratch, but new modules are added to the IPOL's architecture to build OVD-SaaS.

The main advantages of OVD as a SaaS system are the scalability of the application (permitting to spawn effortlessly new processing instances as needed) and the modularity of the resulting pipeline (permitting to replace any block with ease). The developer of an algorithm does not license its use to a client, instead it sells the service of processing the data, or a final application, as in our proposal.

5 Conclusion

Reproducible research is necessary for the advancement of science and the validation of research results. It requires efforts covering various fields including platforms to demonstrate and reproduce results, libraries to directly exploit research artifacts, and editorial work, publishing not only source code, but even complete services. We have surveyed different platforms which can be helpful in reproducible research, recent image-processing libraries such as Higra, and even journals such as IPOL. The IPOL was one of the first requesting that the article, the software and the associated data artifacts should be part of the same publication. IPOL is constantly adapting to the needs of its users with the inclusion of new data types and interactive visualizations. Inspired by IPOL, OVD-SaaS is an extension which intends to be a nexus between academy, industry, and publishers to provide an answer to the reproducibility problems in the context of industrial applications.

Strong efforts are needed from the different actors involved in the research community, such as authors, editors, and publishers. As we showed in this short survey, fortunately several tools, libraries, and services are available to help perform reproducible research.

Acknowledgement

The authors would like to thank Burak Yildiz from Delft University of Technology for providing statistics on reproducedpapers.org platform and Dean Pleban from the Dagshub platform for helping and orienting the authors to measure user activity. They also thank the reviewers for their valuable comments and corrections.

References

1. David L. Donoho, Arian Maleki, Inam Ur Rahman, Morteza Shahram, and Victoria Stodden. Reproducible research in computational harmonic analysis. *Computing in Science & Engineering*, 11(1):8–18, 2009.
2. M. Colom, B. Kerautret, and A. Krähenbühl. An Overview of Platforms for Reproducible Research and Augmented Publications. In *RRPR*, volume 11455 of *LNCS*, pages 25–39. Springer, 2018.

3. B. Yildiz, H. Hung, J. H Krijthe, C. Liem, M. Loog, G. Migut, F. A Oliehoek, A. Panichella, P. Pawełczak, S. Picek, et al. Reproducedpapers. org: Openly teaching and structuring machine learning reproducibility. In *International Workshop on Reproducible Research in Pattern Recognition*, pages 3–11. Springer, 2021.
4. Ana Lucic, Maurits Bleeker, Sami Jullien, Samarth Bhargav, and Maarten de Rijke. Reproducibility as a Mechanism for Teaching Fairness, Accountability, Confidentiality, and Transparency in Artificial Intelligence. 2022.
5. Remi Rampin, Fernando Chirigati, Vicky Steeves, and Juliana Freire. *ReproServer: Making Reproducibility Easier and Less Intensive*. 2018. _eprint: 1808.01406.
6. R. Rampin, F. Chirigati, D. Shasha, J. Freire, and V. Steeves. ReproZip: The Reproducibility Packer. *Journal of Open Source Software*, 1(8):107, 2016.
7. Tibor Šimko, Lukas Heinrich, Harri Hirvonsalo, Dinos Kousidis, and Diego Rodríguez. REANA: A system for reusable research data analyses. In *EPJ web of conferences*, volume 214, page 06034. EDP Sciences, 2019.
8. Nicolas Bonneel, David Coeurjolly, Julie Digne, and Nicolas Mellado. Code Replicability in Computer Graphics. *ACM Transactions on Graphics*, 39(4), July 2020.
9. R. Stojnic and R. Taylor. Papers with code—a facebook AI project, 2018. `https://paperswithcode.com` last accessed: August. 30, 2022.
10. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
11. Matthew McCormick, Xiaoxiao Liu, Julien Jomier, Charles Marion, and Luis Ibanez. ITK: enabling reproducible research and open science. *Frontiers in neuroinformatics*, 8:13, 2014.
12. Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE Int. Conference on Robotics and Automation (ICRA)*. IEEE, 2011.
13. The CGAL Project. *CGAL User and Reference Manual*. 5.4.1 edition, 2022.
14. David Tschumperlé. The CIMG library. In *IPOL 2012 Meeting on Image Processing Libraries*, pages 4–pp, 2012.
15. Geogram: A programming library with geometric algorithms. `https://github.com/BrunoLevy/geogram`.
16. Michaël Roynard, Edwin Carlinet, and Thierry Géraud. An image processing library in modern c++: Getting simplicity and efficiency with generic programming. In *RRPR*, pages 121–137. Springer, 2018.
17. David Auber. Tulip—a huge graph visualization framework. In *Graph drawing software*, pages 105–126. Springer, 2004.
18. Vigra: Vision with generic algorithms. `https://ukoethe.github.io/vigra` last access on May 2022.
19. Dgtal: Digital geometry tools and algorithms library. `http://dgtal.org`.
20. P. Moulon, P. Monasse, R. Perrot, and R. Marlet. OpenMVG: Open multiple view geometry. In *RRPR*, pages 60–74. Springer, 2017.
21. Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE Trans. on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017. `https://topology-tool-kit.github.io/`.
22. B. Perret, G. Chierchia, J. Cousty, S.J. F. Guimarães, Y. Kenmochi, and L. Najman. Higra: Hierarchical graph analysis. *SoftwareX*, 10:1–6, 2019. `https://github.com/higra/Higra`.
23. Charles R. Harris *et al.*. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
24. Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

25. Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 — seamless operability between c++11 and python, 2016. https://github.com/pybind/pybind11.
26. A. Nicolaï, Q. Bammey, M. Gardella, T. Nikoukhah, O. Boulant, I. Bargiotas, N. Monzón, C. Truong, B. Kerautret, P. Monasse, and M. Colom. The approach to reproducible research of the image processing on line (ipol) journal. *Informatio*, 27(1):76–112, 2022.
27. M. Colom, T. Dagobert, C. d. Franchis, R. G. v. Gioi, C. Hessel, and J. M. Morel. Using the ipol journal for online reproducible research in remote sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:6384–6390, 2020.
28. Anne-Flore Baron, Olivier Boulant, Ivan Panico, and Nicolas Vayatis. A Compartmental Epidemiological Model Applied to the Covid-19 Epidemic. *Image Processing On Line*, 11:105–119, 2021. `https://doi.org/10.5201/ipol.2021.323`.
29. Roberto Di Cosmo and Stefano Zacchiroli. Software heritage: Why and how to preserve software source code. In *iPRES 2017-14th International Conference on Digital Preservation*, pages 1–10, 2017.