Advancing OMR as a Community: Best Practices for Reproducible Research

Alexander Pacha
Institute of Visual Computing and Human-Centered Technology
TU Wien
Vienna, Austria
alexander.pacha@tuwien.ac.at

***Abstract*—Optical Music Recognition has been under investigation for over 60 years but remains an unsolved problem, because research happens distributedly, often without reusability in mind. As scientists, it should be one of the goals to share knowledge in a way, that it becomes accessible and useful for someone else to build on top. Without that, one's effort is often doomed to rot in a drawer. To oppose this development, not only the paper, but also the source code, datasets, and executables should be made publicly available for the community to finally advance beyond the state, where the wheel is re-invented every time a new researcher joins the field.**

***Index Terms*—optical music recognition, reproducible research**

I. Introduction

Optical Music Recognition (OMR) has seen decades of research with many questions remaining unsolved [1]. Solutions, algorithms, and systems were frequently developed independently, trying to solve particular (sub-)problems [2]–[7]. However, without the perspective of what happens to a project after the scientific publication has been accepted, the advances we have achieved remain scattered across the world. It would be great to share datasets, algorithms, results, and knowledge alike. This is slowly becoming the scientific norm, but to leverage the work of someone else, the author has to do more than just providing a link to the source-code: properly documenting the code base, providing automatic tests that can be executed on a continuous integration platform and ideally one-click solutions for running the application are necessary. These things are part of the standard repertoire used in industry projects and should be equally relevant in scientific projects from the field of computer science.

II. Publish reproducible results

There are three kinds of artifacts that researchers commonly share with the community: knowledge, primarily in the form of scientific articles, datasets, and source code. Papers usually undergo a scientific peer-review to ensure a certain quality level and are published in journals, or presented at conferences. Many of these publications now come with links to the source code that is hosted on Github or similar platforms [8]–[12]. Sharing access to the repository allows not only to see the final result, but also the steps that led to that result. However, the other two artifacts often lack a rigorous review or are not review at all. When reviewing another peer's work, demand the code as supplementary material to verify reproducibility. If understanding their code turns out to be too hard for another expert working in the same field, it is unlikely that anyone will use the reviewed work. If so, let the authors know by adding an appropriate remark in the review.

A. Source Code

When publishing source code, it should contain at least a README file, which summarizes the purpose of this repository as well as how to build and run the application. Keep in mind, that good names for methods, variables, and packages are preferred to extensive commenting in the code [13]. Sensible documentation that is generated directly from the source code[1] can help other users to quickly get started, ideally with a few examples on how to use your project. http://muscima.readthedocs.io/en/latest/Tutorial.html is an example of a good tutorial. To further promote free reuse, an appropriate license should be attached to the repository, e.g., the MIT license. https://tldrlegal.com/ contains a concise summary of many popular licenses. Note, that if no license is specified, default copyright applies that restricts the allowed usage heavily.

B. Executables

Use a continuous integration platform, such as Travis[2], to prevent unexpected regressions in the code. Many of these services are freely available for open-source projects. Some of them can also be used as a platform to provide executables for potential users. If an application has a complex environment and requires specific dependencies, consider using containerization, e.g., with Docker[3]. The additional benefit of setting up an application as a ready-to-use container is a possible integration with environments like the DIVAServices [14], that provide interactive demos as part of their website[4]. Such an interactive demo lets other researchers explore your work without the huge upfront effort, that is sometimes required to get an application to run locally.

[1] https://readthedocs.org/
[2] https://travis-ci.org/
[3] https://www.docker.com/
[4] http://divaservices.unifr.ch/spotlight

C. Datasets

A delicate matter concerns the creation of datasets. This sore point is still a major issue within the OMR community. Although there are a few large datasets that have recently been published or that will be published in the near future, these datasets are often not compatible with each other due to different notation formats, encodings, and granularity of the underlying information. WoRMS offers a great possibility to address this issue as a community. From a practical point of view, datasets will probably always be encoded somewhat differently, but if the authors claim, that their dataset can (in theory) be exported into a particular format, it is their responsibility to provide a respective converter. As a general guideline, prefer a simple, plain format (e.g., comma-separated-values) that can easily be converted into other data formats. It is also beneficial, if additional visualization functions are provided, that can help to inspect intermediate and display final results. Finally, to gain visibility, a dataset can be listed on the website of the OMR datasets project (http://apacha.github.io/OMR-Datasets/) by simply creating a pull-request or adding an issue.

D. Showcases

Two projects demonstrate the implementation of above-mentioned guidelines: The OMR datasets project[5] and the Music Object Detector[6] [15]. Their source code is freely available under an MIT license, and detailed README files explain how to use the source code, how to reproduce the results and how to test the final system with pre-trained models and demo-scripts. The Music Object Detector is also available via the DIVAServices Spotlight, where arbitrary music scores can be uploaded to test the object detection capabilities easily.

III. Do not re-invent the wheel

OMR has seen many attempts to "solve it" but it is still considered an unsolved problem for many scenarios. Often, users are disappointed when they try OMR on their dataset and see themselves confronted with its insufficiencies. This is caused by applications not being designed to handle arbitrary scores: hard-coded variables, assumptions on the layout or the limitation to a particular music notation system lead to programs that only work well for the dataset for which they were developed. This pitfall can be avoided by letting actual users test your system and give you honest feedback. When reflecting on the achievements as a community, many things were attempted but failed so far. Agreeing on an output format or a standard vocabulary are good examples. Another one is how to evaluate entire OMR systems. Despite many proposed metrics, you simply cannot compare two systems that were designed with entirely different purposes in mind. Scientific competitions can help to mitigate this problem because the task needs to be well defined, an appropriate dataset has to be provided, and all participants must follow the same evaluation protocol. The 2012 Music Scores Competition [16] is an example that led to a range of robust and generalizable solutions for staff removal. And even many years after the competition, the dataset is still being used intensively for a wide range of tasks, thanks to its liberal license. Finally, it has to be said, that we should avoid re-inventing the wheel over and over again and not only make sure that our research is usable by other researchers, but also that we actively leverage the work of others and build on top of their work, instead of always starting over from scratch.

[5] http://apacha.github.io/OMR-Datasets/

[6] http://github.com/apacha/MusicObjectDetector-TF

References

[1] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. Marcal, C. Guedes, and J. S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, 2012.

[2] J. Ashley, B. Laurent, P. Greg, and E. I. Fujinaga, "A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources," *ACM SIGSOFT Software Engineering Notes*, vol. 24, no. 1985, pp. 1994–1997, 2008.

[3] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga, "A Comparative Study of Staff Removal Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 753–766, 2008.

[4] A. Fornés, "Primitive Segmentation in Old Handwritten Music Scores," in *International Workshop on Graphics Recognition 2005*, 2005, pp. 279–290.

[5] P. Bellini, I. Bruno, and P. Nesi, "An off-line optical music sheet recognition," in *Visual Perception of Music Notation: On-Line and Off Line Recognition*. IGI Global, 2004, pp. 40–77.

[6] D. Blostein and H. S. Baird, *A Critical Survey of Music Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 405–434.

[7] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Early handwritten music recognition with hidden Markov models," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 319–324.

[8] A. Pacha and H. Eidenberger, "Towards a universal music symbol classifier," in *Proceedings of the 12th IAPR International Workshop on Graphics Recognition*, New York, USA, 2017, pp. 35–36.

[9] J. Calvo-Zaragoza and D. Rizo, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, no. 4, 2018.

[10] L. Tuggener, I. Elezi, J. Schmidhuber, and T. Stadelmann, "Deep watershed detector for music object recognition," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[11] J. Hajič jr. and M. Dorfer, "Prototyping full-pipeline optical music recognition with muscimarker," in *Extended abstracts for the Late-Breaking Demo Session of the 18th International Society for Music Information Retrieval Conference*, 2017.

[12] K. MacMillan, M. Droettboom, and I. Fujinaga, "Gamera: Optical music recognition in a new shell," *Proceedings of the International Computer Music Conference*, pp. 1–4, 2002.

[13] A. Shvets, G. Frey, and P. Marina, "Source making." [Online]. Available: http://sourcemaking.com/refactoring/smells/comments

[14] M. Würsch, M. Liwicki, and R. Ingold, "Web Services in Document Image Analysis - Recent Developments and the Importance of Building an Ecosystem," in *13th IAPR Workshop on Document Analysis Systems*, Vienna, Austria, 2018, pp. 334–339.

[15] A. Pacha, K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger, "Handwritten music object detection: Open issues and baseline results," in *2018 13th IAPR Workshop on Document Analysis Systems (DAS)*, 2018, pp. 163–168.

[16] A. Fornés, A. Dutta, A. Gordo, and J. Lladós, *The 2012 Music Scores Competitions: Staff Removal and Writer Identification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 173–186.