

MISA++: A standardized interface for automated bioimage analysis

R. Gerst, A. Medyukhina, M.T. Figge

28/10/2021

LEIBNIZ-HKI

MISA++: A standardized interface for automated bioimage analysis

Roman Gerst^{1,2}, Anna Medyukhina¹, and Marc Thilo Figge^{1,3}

¹Applied Systems Biology, Leibniz Institute for Natural Product Research and Infection Biology - Hans-Knoell-Institute, Jena, Germany
²Faculty of Biological Sciences, Friedrich-Schiller-University Jena, Germany
³Institute of Microbiology, Faculty of Biological Sciences, Friedrich-Schiller-University Jena, Germany

1. Analysis of big volume image data

Modern imaging techniques, such as lightsheet fluorescence microscopy (LSFM), allow the capture of whole organs in three spatial dimensions. This includes the evaluation of structural changes such as a reduced number of glomeruli in kidneys [2] or the formation of bronchus-associated lymphoid tissue (BALT) caused by lung inflammation [3]. The analysis of these big volume image data requires a combination of user-friendly and highly efficient tools.

High-performance tools e.g. C++ / OpenCV	User-friendly tools e.g. ImageJ
<ul style="list-style-type: none">Fast runtimeFull control over hardwareComplex code syntaxHard to debugNeed compilationHard to integrate	<ul style="list-style-type: none">Easy to program and debugGraphical user interfacesEasy to integrateSlow runtimeLimited optimizationLimited server support

Idea: Make high-performance tools easy to integrate into user-friendly tools via our framework [1].

- High performance
- Easy to use for non-developers

2. Framework

MISA++
Modular Image Stack Analysis for C++

Available on **GitHub** <https://github.com/applied-systems-biology/misa-framework>

Modular Image Stack Analysis for C++ (MISA++) is a platform-independent open source framework that simplifies developing efficient image analysis tools. It provides standardized data and parameter handling, parallelization, documentation, and integration into user-friendly software via an ImageJ plugin.

High-performance tools

- OpenCV
- OME
- CPU and memory-efficient
- Runs on servers

Our framework

- Automated parallelization
- Memory management
- Standardized parameters
- Standardized input/output
- Standardized documentation

User friendly tools

- Easy data analysis
- Graphical interfaces

3. Glomeruli segmentation

Glomeruli are functional structures within the renal cortex that are damaged by diseases and toxins [1]. Our algorithm consists of five steps that first segments the tissue and then extracts the glomeruli.

- 2D tissue segmentation**
Thresholding
- Tissue quantification**
Number of pixels, volume
- 2D glomeruli segmentation**
Optimal background removal, Otsu thresholding
- 3D glomeruli reconstruction**
3D constrained connected components
- Glomeruli quantification**
Number of glomeruli, volume, diameter

The original implementation is written in the Python language. By creating an advanced MISA++-based implementation, we could reduce the analysis runtime for the same data set on the same hardware by two orders of magnitude.

MISA++ 41 min
MISA++ 841 min
Python 3906 min
ImageJ 5510 min

Runtime (min)

Compared to the implementation published by Klingberg et al. [1] (Python), MISA++ calculates the same work up to 95 times faster if all images are available at the same time. If the analysis is done one after another kidney (*), the calculation is still approximately 4 times faster. MISA++ was 134 times faster than a implementation in ImageJ. Parallelization using 30 threads.

4. ImageJ integration

We developed a plugin for ImageJ that integrates tools created with MISA++ into ImageJ. Users can setup and monitor analyses, and display results within a easy-to-use graphical user interface.

Features

- Manage MISA++ applications
- Import input data
- Change algorithm parameters
- Run analyses
- Display analysis results
- Create plots & tables
- Export analysis tasks
- Detailed runtime statistics
- Create pipelines

All features are available for any MISA++ application, made possible by the machine-readable documentation. All user interfaces are automatically generated.

In future, the ImageJ integration will be enhanced by upgrading it to JIPipe (<https://www.jipipe.org/>).

JIPipe

roman.gerst@leibniz-hki.de

www.leibniz-hki.de

References

[1] Gerst et al. 2020. SoftwareX 11:100405.
[2] Klingberg et al. 2017. ASN 2020: 452-59.
[3] Mohr et al. 2018. Cell Mol Immunol. 15: 875-887

ILRS **UNIVERSITÄT JENA**