SOPHYSM: More Steps towards *Digital Twins* of Solid Tumours

Diego Cividini[1] and Marco Antoniotti[1,2*]

[1] Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy.
[2] Bicocca Bioinformatics Biostatistics and Bioimaging Centre - B4, University of Milano-Bicocca, Milan, Italy.

ORCID codes: DC 0009-0003-3955-0886; MA 0000-0002-2823-6838.
*corresponding author: marco.antoniotti@unimib.it

September, 1, 2026

Abstract

The notion of *digital twin* is well established in several engineering disciplines, but, due to the complexity of the field, it is still in the making when it comes to the Life Sciences. We are addressing this issue, while realizing that many different pieces must come together to progress toward the goal.

The SOPHYSM system is a Julia tool-set for simulating the evolution of *solid tumours*, while tracking genotyping information at the *Single-cell* level.

To be more useful for Life Science practitioners, SOPHYSM provides a *histopathology* image handling facility. The images are analysed to pinpoint cells in a tissue. The cell positions and their morphological boundaries are then fed into a *spatial simulator* as a basis to compute possible *solid tumour* evolutions. In this paper we describe the useful (and necessary) porting to Julia of one of the latest *image analysis* "algorithms" for cell identification: the Python-based Cellpose. This allowed us to simplify our development pipeline for identifying (i.e., *segment*) cells' positions from histopathology images.

Here we report on this effort, which required unpacking and reverse engineering what is a sophisticated AI tool, Cellpose, to port its *public weights* to our Julia implementation, by leveraging the ONNX libraries.

It should be noted that this is an example of the importance of providing the weights of a trained AI model, to be able to reuse them in different settings, like, in our case, SOPHYSM.

1 Introduction

A *Computational Biology* research group ends up spending quite some time setting up *simulation* datasets in order to evaluate the *Machine Learning* and the *A.I.* tools it creates. Our experience is not unusual in this respect, and there is a growning consensus that shared simulation tools, capable of consistently generating data for various scenarios, would be desirable.

The notion of *digital twin* is well established in several engineering disciplines, but it is still in the making when it comes to the Life Sciences. While the establishment of different digital twins would have far reaching consequences, at least in the pre-clinical setting (cf., [1]), the complexities of the Life Sciences have been quite an obstacle. We are addressing this issue, while realizing that many different pieces must come together to progress toward the goal.

Our tool **S**olid **O**ncology **PHY**logenetic **S**patial **M**odeller (SOPHYSM) includes the J-Space simulator [2], which addresses these issues for *solid tumor*, *single-cells*, *genotype lineage tracking* simulation. To the best of our knowledge, J-Space is the first tool with all these features. Alas, J-Space usability is limited by the less-than-friendly, manual simulation setup.

To overcome this issue, we decided to use *histopathology* images as input. Previous iterations of SOPHYSM relied on traditional computer vision baselines for image segmentation (e.g., watershed algorithms and adaptive thresholding). However, to improve robustness and incorporate recent AI methods into our workflow, we felt it was necessary to upgrade this pipeline. As a software engineering contribution, this paper details the native Julia porting of a state-of-the-art AI segmentation tool, establishing a cohesive, single-language toolchain without external runtime dependencies.

2 The Cellpose Algorithm and its Native Julia Implementation

The segmentation of histological and cytological images is a fundamental step in the quantitative analysis of biomedical images. The traditional approaches, such as those based on *adaptive thresholding* [3], *watershed algorithms* [4] and *morphological operations* [5], often require careful manual calibration and show significant limitations in the presence of complex tissue architectures, overlapping cells and variable staining conditions. The Cellpose algorithm, introduced in [6], represents a change of paradigm in this field by substituting the traditional direct prediction of instance masks with a flow-based representation. Instead of predicting the binary mask for each cell, Cellpose infers a vector field that encodes the direction and magnitude of pixel movement towards the center of each cell. This approach allows for a more robust segmentation, especially in challenging scenarios where cells

are densely packed or have irregular shapes. With this approach, a neural network is trained to predict, pixel-wise, two components of the spatial gradient of the predicted flow vector $\mathbf{dP} = (dP_y, dP_x)$, where dP_y and dP_x denote the vertical and horizontal components of the simulated diffusion gradient field, and a probability map $\mathbf{P}$ that indicates the likelihood of each pixel belonging to a cell. These gradients define a deterministic flow system: pixels belonging to cells are iteratively moved along the predicted vector field until they converge to local fixed points, which correspond to the centers of the cells. Pixels that share the same convergence point are aggregated in the same instance mask. This architecture, inspired by simulated heat diffusion and gradient dynamics, implicitly encodes topological connectivity and proves robust to variations in the cell shape, in the cell density and in the instrumentation noise, overcoming the limitations of traditional geometric-prior methods or post-separation heuristics.

2.1 Evolution to Cellpose-SAM

Recently, the Cellpose architecture has been extended by integrating pre-trained backbones on foundational datasets such as SAM [7]. The resulting model, Cellpose-SAM, combines the inductive biases acquired during pre-training on natural images with the flow-based mask reconstruction mechanism, specific for biological microscopy. This synergy allows the model to surpass the level of inter-human agreement in segmentation, approaching the theoretical limit of human consensus. Furthermore, the training was enriched with augmentations aimed at ensuring invariance of the order of channels, cell dimensions and degradations typical of microscopy images. Consequently, the model can be applied directly to images with arbitrary resolution or contrast without the need for manual resizing or auxiliary models to estimate the cell diameter, which simplifies the entire pipeline configuration and improves the generalization on unseen datasets.

2.2 Reasons for a Native Port to Julia

Despite the excellent algorithmic design and the impressive performance of Cellpose, its official implementation depends on Python and especially on the PyTorch deep learning framework. Integrating the original Cellpose implementation into Julia environments like SOPHYSM inevitably introduces interoperability overheads (e.g., cross language data serialization with PyCall), fragmenting memory management and causing non-deterministic behavior in multithreaded contexts. In order to guarantee a native seamless integration and numeric stability, we decided to port the Cellpose algorithm to Julia. Our implementation leverages models exported in the ONNX format [8], making it framework-agnostic and compatible with runtime inference engines optimized for CPU and GPU acceleration, such as ONNXRuntime, the official

high-performance cross-platform engine for executing ONNX models [9].

2.3 Architecture of Cellpose.jl

The module is structured as a modular pipeline, type-stable and optimized to have a column-major memory layout, necessary for efficient numerical computations in Julia. The main components of the pipeline include:

- **Preprocessing and normalization**: The image is normalized channel-wise mapping the 1^{st} and 99^{th} intensity percentiles in the range $[0, 1]$. This step mitigates the sensitivity to extreme outliers (e.g., saturation artifacts or some dust on the slide) preserving the contrast of the majority of the pixels.
- **Parallel tiling inference with weighted blending**: The image is partitioned into 256×256-pixel tiles with an overlap of 10% (corresponding to a stride of 230 pixels). This allows to process large images that do not fit in memory and to leverage the GPU acceleration for inference. The predictions on the overlapping regions are blended with a flat-top blending window, given by the product of truncated linear functions in order to suppress border discontinuities without amplifying the noise. The execution is parallelized via `Threads.@threads`, employing thread-safe accumulation through atomic locks, ensuring linear scalability on the number of available CPU cores.
- **Flows numeric integration and seeds recognition**: The routine `follow_flows` executes 200 iterations of the Euler integration method with bilinear interpolation. The implementation uses views (`@views`) and bounds-check elimination (`@inbounds`) to avoid unnecessary memory allocations and improve runtime performance. It maintains strict type-stability with the given input and it applies a scale factor to the predicted flows in order to align itself with the original loss function used during training, which is based on the simulated diffusion field of the cell masks. The seeds recognition is performed with a local peak detection algorithm, which identifies the pixels that are local maxima in the convergence point density map and that are also convergence points of the flow field. The convergence points are next smoothed with a 3×3 low-pass filter, thresholded and clustered with a breadth-first search to uniquely identify all the cell centers.
- **Masks reconstruction and adaptive quality check**: Each foreground pixel is assigned to the mask of the seed to which it converges. The post-processing step removes the masks that have a flow field with flow consistency error (i.e. likely false positives) (by using customizable Mean Squared Error (MSE) thresholding, typically set to 0.4) applying dynamic area-based filters (via `min_size` and `max_size` parameters

automatically computed based on the cell diameter) and reassigning the IDs consequently from 1 to the number of masks N. The use of `@inbounds`, the pre-allocation of queues and the substitution of the vector operations with indexed loops allows to achieve reduced memory footprint and deterministic behaviour even under multithreaded execution.

2.4 User Interface and Integration

The Cellpose.jl module is designed to be invoked directly from the SOPHYSM graphical interface, allowing users to segment histopathology images without writing code. Figure 1 shows the main segmentation panel, where users can load an image, select the ONNX model, adjust parameters (e.g., cell diameter, thresholds), and visualize the resulting masks overlaid on the original image. The interface also supports batch processing and exports results in both analytical (16-bit TIFF) and visual (PNG overlay) formats.

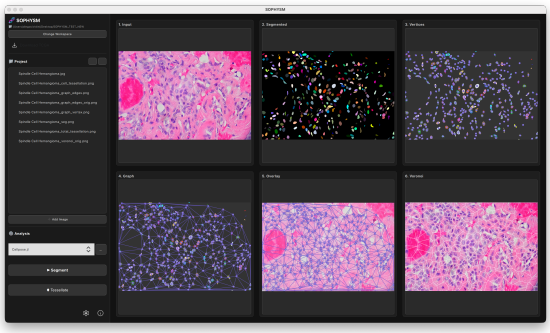

Figure 1: Graphical user interface of SOPHYSM with the Cellpose.jl segmentation module. The left panel contains project files, parameter settings, and buttons to *Segment* and *Tessellate* the input image. The center panel displays sequentially: the original image, the predicted masks, the adjacency graph (vertices and full structure), and its overlay with the Voronoi tessellation on the original image.
(Image by Ed Uthman, CC BY 2.0, via Wikimedia Commons).

This tight integration enables end-to-end workflows: from raw histopathology images to quantitative simulations, without intermediate file conversions or external dependencies.

2.5 Validation and Integration of the Pipeline

The ONNX format guarantees structural consistency with the reference Python implementation, as we use identical pre-trained weights, activation functions and computational graph. To quantitatively assess numerical consistency,

we performed a side-by-side benchmark on a representative histopathology image using identical preprocessing and post-processing parameters.

Tables 1 and 2 summarize key metrics. The cell count discrepancy of 2.3% (390 vs. 399 cells) is minimal and well within the expected range for out-of-distribution generalization benchmarks [10]. The binary Intersection-over-Union (IoU) of 90.5% and pixel accuracy of 98.5% indicate excellent agreement on foreground/background classification, surpassing typical inter-annotator agreement levels reported in the literature [10, 7].

Table 1: Cell-level metrics for Python reference and Julia native implementations on a test histopathology image.

Metric	**Python**	**Julia**
Cell count	390	399
Mean cell area (px^2)	718.2 ± 367.5	713.5 ± 376.2
Median cell area (px^2)	666	672

Table 2: Agreement and discrepancy metrics between Python and Julia implementations.

Metric	**Value**
Cell count difference	+9 (2.3%)
Pixel accuracy	98.54%
Binary IoU (foreground)	90.49%
False negative pixels	11,798
False positive pixels	16,390

The distribution of cell sizes shows that discrepancies, though minimal, are slightly more frequent in the smaller size bins (e.g., < 500 px^2), where threshold sensitivity has the largest impact—a pattern consistent with expected behavior under minor numerical perturbations [6].

To complement the quantitative metrics, Figure 2 provides a visual assessment of segmentation agreement based on cellular presence rather than instance IDs. The overwhelming prevalence of blue regions confirms that both implementations successfully detect the same cellular structures across the tissue architecture. Isolated red and green pixels occur primarily at cell boundaries, which is consistent with minor floating-point accumulation differences during the weighted stitching phase and varying sensitivity to boundary thresholds.

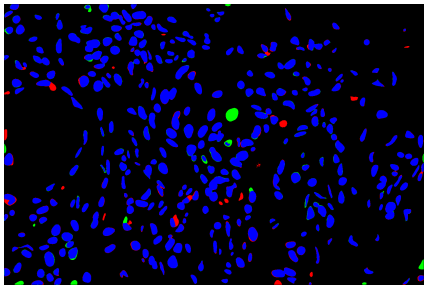

Figure 2: Pixel-wise difference map comparing the Python reference implementation and the Cellpose.jl native port. **Blue:** Structural agreement (both implementations detected a cell instance). **Green:** Pixels identified as cells only by the Python reference. **Red:** Pixels identified as cells only by the Julia port. The predominance of blue regions visually confirms that the Julia implementation preserves the cellular structure identified by the reference model, with discrepancies restricted to sparse boundary pixels.

To assess robustness across varying tissue complexities, we also evaluated the implementation on more challenging images featuring densely packed cells and lower contrast. In these scenarios, discrepancies naturally increase (cell count difference up to $\sim 5\%$, binary IoU $\sim 70\%$) but remain within acceptable bounds for downstream simulation tasks, consistent with the variability observed in out-of-distribution benchmarks [10]. This confirms that the Julia port maintains functional equivalence to the reference implementation across a range of histopathological conditions. A broader quantitative evaluation across multiple datasets and downstream performance metrics is currently under development for an extended version of this work.

For downstream applications such as tumor evolution modeling in SOPHYSM, this level of agreement is more than sufficient: population-level statistics (cell counts, positions, neighborhood relationships) are preserved with near-perfect accuracy, which drives simulation outcomes more than pixel-perfect boundaries.

The removal of the Python runtime dependency reduced memory overhead by approximately 15% in our benchmarks, enabled predictable native multi-threading behavior, and established a lighter, reproducible toolchain. The module is designed to be invoked directly during the SOPHYSM acquisition phase, returning `Int32` matrices where zero denotes background and $k \in \{1, \ldots, N\}$ identifies the k-th cell. These masks constitute direct input for tumor evolution models and cellular tracking, enabling end-to-end transition from raw data to quantitative simulations without intermediate conversions or external dependencies.

3 Conclusion

In this paper we have described a part of the SOPHYSM tool-set, devoted to the segmentation of histopathology images. The function of this part of the tool-set is to provide an input for the J-Space simulator [2]; while working on this part of the overall SOPHYSM pipeline, we realized that being able to leverage a full Julia implementation of the recent Cellpose procedures would be a plus. We thus proceeded to port the Cellpose procedure to Julia, making it available to the community at large and providing a solid engineering foundation for the tool.

The ultimate goal of SOPHYSM is to act as a fully-fledged *digital twin* framework. While this paper focuses on the necessary architectural and segmentation upgrades, our immediate future work will focus on providing a complete end-to-end evaluation. Specifically, we plan to quantitatively compare this new Cellpose-based pipeline against the previous traditional baselines (e.g., watershed) using downstream metrics, directly feeding the extracted cell counts, positions, and neighbourhood structures into an active J-Space simulation run.

Conflict of interests

The authors declare that they have no conflict of interest.

Availability of data and software code

The source code for the Cellpose.jl module is open-source and publicly available on GitHub at `https://github.com/diegocividini/Cellpose.jl`.

References

[1] Colleen E. Clancy and Bennet A. Landman. Towards digital twins for basic and preclinical research. *Nature Reviews Methods Primers*, 6(5), January 2026.

[2] Fabrizio Angaroni, Alessandro Guidi, Gianluca Ascolani, Alberto D'Onofrio, Marco Antoniotti, and Alex Graudenzi. J-SPACE: a Julia package for the simulation of spatial models of cancer evolution and of sequencing experiments. *BMC Bioinformatics*, 23(269), July 2022.

[3] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[4] Jos Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informatica*, 41:187–228, 01 2000.

[5] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 1999.

[6] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, pages 100–106, 2021.

[7] Marius Pachitariu, Michael Rariden, and Carsen Stringer. Cellpose-SAM: superhuman generalization for cellular segmentation. *bioRxiv*, May 2025.

[8] ONNX Community. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`, 2020. Accessed: 2024-05-15.

[9] Microsoft Corporation. ONNX Runtime: High performance inference engine. `https://onnxruntime.ai/`, 2023. Accessed: 2024-05-15.

[10] Haoran Chen, Robert F. Murphy, and Diane Lidke. Evaluation of cell segmentation methods without reference segmentations. *Molecular Biology of the Cell (MBoC)*, 34(6):ar50, May 2023.