nature methods

Article

https://doi.org/10.1038/s41592-024-02404-5

Arkitekt: streaming analysis and real-time workflows for microscopy

Received: 12 October 2023

Accepted: 1 August 2024

Published online: 18 September 2024

Check for updates

Johannes Roos[1,2], Stéphane Bancelin[1], Tom Delaire[1], Alexander Wilhelmi[3], Florian Levet[1,4], Maren Engelhardt[2,5], Virgile Viasnoff[6], Rémi Galland[1], U. Valentin Nägerl[1] & Jean-Baptiste Sibarita[1]

Quantitative microscopy workflows have evolved dramatically over the past years, progressively becoming more complex with the emergence of deep learning. Long-standing challenges such as three-dimensional segmentation of complex microscopy data can finally be addressed, and new imaging modalities are breaking records in both resolution and acquisition speed, generating gigabytes if not terabytes of data per day. With this shift in bioimage workflows comes an increasing need for efficient orchestration and data management, necessitating multitool interoperability and the ability to span dedicated computing resources. However, existing solutions are still limited in their flexibility and scalability and are usually restricted to offline analysis. Here we introduce Arkitekt, an open-source middleman between users and bioimage apps that enables complex quantitative microscopy workflows in real time. It allows the orchestration of popular bioimage software locally or remotely in a reliable and efficient manner. It includes visualization and analysis modules, but also mechanisms to execute source code and pilot acquisition software, making 'smart microscopy' a reality.

Over the past 30 years, cellular microscopy has seen a complete transformation, passing from analog microscopy in the 1990s to the fully automatized, quantitative and 'intelligent' modalities of today[1]. We are now able to monitor live biological samples ranging from single cells to entire organisms, resolving their structural details with resolution down to the nanoscale. Progress in optics, electronics and computer technology paved the way for this, and the advent of deep learning tools is ringing in a new era in bioimage analysis[2,3].

In today's expanding and diversifying bioimage ecosystem, the acquisition and analysis software MicroManager[4,5] and ImageJ/FiJi[6,7] have remained gold standards for open-source image analysis. They come with a user-friendly graphical user interface (GUI) and can be customized and support new methods via an extensive plugin system. Both tools are under active development and see improvements in each new version.

Yet new, often Python-based, bioimage software such as Napari[8] and CellProfiler[9] have seen rapid adoption, in part because they can outperform ImageJ in certain advanced image processing and visualization tasks. Napari is tailored to efficiently work with data in external memory, enabling the visualization of large datasets (>terabyte) in three dimensions (3D), while CellProfiler offers reliable segmentation and analysis plugins for batch analysis. Both solutions integrate a plugin system, but even though the number of available analysis modules and plugins is rapidly growing, a lot of functionality of ImageJ/FiJi, especially legacy analysis scripts, is not available for these solutions.

[1]Interdisciplinary Institute for Neuroscience, University of Bordeaux, CNRS, Bordeaux, France. [2]Institute of Anatomy and Cell Biology, Medical Faculty, Johannes Kepler University, Linz, Austria. [3]Frankfurt Institute for Advanced Studies, Frankfurt, Germany. [4]Bordeaux Imaging Center, University of Bordeaux, CNRS, INSERM, Bordeaux, France. [5]Clinical Research Institute for Neurosciences, Johannes Kepler University, Linz, Austria. [6]Mechanobiology Institute, National University of Singapore, Singapore, Singapore. e-mail: jean-baptiste.sibarita@u-bordeaux.fr

Table 1 | Overview of the main differences between existing solutions and Arkitekt

	Workflow design	Application type	Data management	Virtualization	Concurrency	Multicomputer setup	Data exploration
JIPipe	Node based	Java application	File based	None	Thread-based parellelism	No	Yes (through tables)
Imjoy	Linear	Progressive web app	File based	WASM in browser	Asynchronous worker queues	Yes	No
BioimageIT	Linear	Python-based desktop app	File based	Container	Sequential execution	No	No
Arkitekt	Node based	Web server	Database and object storage	Container	Asynchronous worker queues	Yes	Yes

Additionally, many interesting new methods need to be manually run through command line interfaces or necessitate complex developmental and hardware setups. For example, the prolific field of deep learning still requires manual script writing for training and predictive analysis and access to large computing power that is typically not available on a regular desktop computer.

In this flourishing software and hardware environment, tool users unfortunately often have to face interoperability issues and thus need to laboriously bridge various analytical software through programmatic patchwork. This prevents many nonexpert users from taking advantage of the most appropriate methods for their bioimage workflows.

These technical limitations not only hinder the impact of these methods in biology but also do not meet FAIR data principles (findability, acessibility, interoperability and reusability), as complex workflows are hard to reproduce[10]. Additionally, as modern microscopy allows the acquisition of large amounts of multidimensional data, the question of data provenance, that is, knowing when and how a process affected the original data, becomes more and more important.

To facilitate the orchestration of diverse bioimage tools, researchers are adopting general workflow manager solutions such as Nextflow[11] or Galaxy[12]. These tools allow the creation of highly automated pipelines of data transformation, where multiple tools can be linked together to create a workflow that can run in a reproducible manner on dedicated hardware such as computer clusters. However, they usually can only be deployed in scenarios where the system and its workflows are maintained by experts. Crucially, these systems also lack essential features for modern bioimage analysis, such as dedicated data types for multidimensional image data or support for GUI applications to enable interactive workflows (for example, to mark regions of interests (ROIs) or adjust thresholds).

Consequently, more targeted approaches have recently been developed. Building on the same containerization technology employed in a variety of general workflow managers, BioImageIT[13] provides a user-friendly graphical desktop application that allows running analysis scripts locally in sandboxed environments and the creation of batch workflows, including deep learning, C/C++ code, Python or Java scripts. ImJoy[14], a progressive web application originally developed to simplify nonexpert access to deep learning algorithms, allows image processing and analysis software such as ImageJ to be run in the browser, bridging remote execution on dedicated hardware in the cloud with local plugins that can be written in various languages.

While these tools are paving the way to facilitating workflow creation from existing bioimage ecosystem components, they remain limited in several important aspects (Table 1 and Supplementary Table 1). Both tools are tailored to enable linear batch workflows, where data get transformed sequentially in an analytical pipeline after the acquisition. Consequently, they are usually not compatible with the emerging context of nonlinear or feedback workflows, often referred to as 'smart microscopy'.

In smart microscopy, automatic bioimage pipelines can trigger or modulate acquisition and signal processing during the ongoing experiment, depending on the imaging content. This allows for the optimization of both acquisition speed and data flow, reducing, for instance, the exposure of the sample to potentially toxic light. Several dedicated applications of smart microscopy were recently published[1,15,16], illustrating how event-driven acquisitions can greatly facilitate monitoring biological processes. However, all these applications were developed on dedicated microscopy and analysis systems only maintainable by expert users, calling for a solution that can make smart microscopy more accessible.

In parallel, as automated and high-throughput microscopy becomes more widespread, real-time analytical workflows, providing live quantifications and data quality assessment during ongoing experiments, become a necessity to ensure that fragile experimental conditions are well maintained. The capability to remotely visualize them, together with the data, would be beneficial but is unfortunately currently not easily achievable.

Existing solutions are not very user friendly and hardly allow nonexperts to design, run and monitor more involved analytical pipelines. In this context, graphical programming tools, as seen in LabView and the recent work of JIPipe[17], can vastly help nonexpert users design workflows in an intuitive manner without a single line of code. Unfortunately, both JIPipe and Labview are restricted to their own ecosystems and do not allow for workflows spanning multiple software.

Here, we introduce Arkitekt, a solution specifically developed to fill these gaps. Arkitekt offers nonexperts and developers alike a flexible platform to create sophisticated interactive and feedback bioimage workflows from popular acquisition, visualization and analysis tools.

Design and implementation

An early design decision for Arkitekt was to 'not reinvent the wheel' and utilize the features of the myriad of bioimage analysis tools that already exist. Therefore, Arkitekt was built as a backbone that allows to integrate these tools seamlessly into workflows, offloading computation to them and the hardware they were designed to run on.

Arkitekt acts as a middleman between the users and the bioimage applications (the apps) and utilizes network connections to connect them irrespective of their physical location. Arkitekt and its interface then represents an abstraction layer for the user to both schedule work on the apps and retrieve and store their data. Arkitekt operates therefore as a central app (the server) that can be installed on any computer, inside the laboratory or on cloud infrastructure, and that advertises itself on the network. The bioimage apps can then seamlessly connect and in turn advertise their functionality to Arkitekt (for example, display an image, deconvolve or segment it and so on) (Fig. 1a). Arkitekt will save a record of all connected apps and their functionalities (the nodes) and can play the role as the central storage for all the data and metadata, including images (raw and processed) as well as quantitative data and annotations (ROIs, labels and tables) (Fig. 1b). In addition, through its graphical or programmatic interface, Arkitekt enables the user to call these nodes directly and to orchestrate them altogether by drag-and-drop manipulation (the workflow) (Fig. 1c).

https://doi.org/10.1038/s41592-024-02404-5

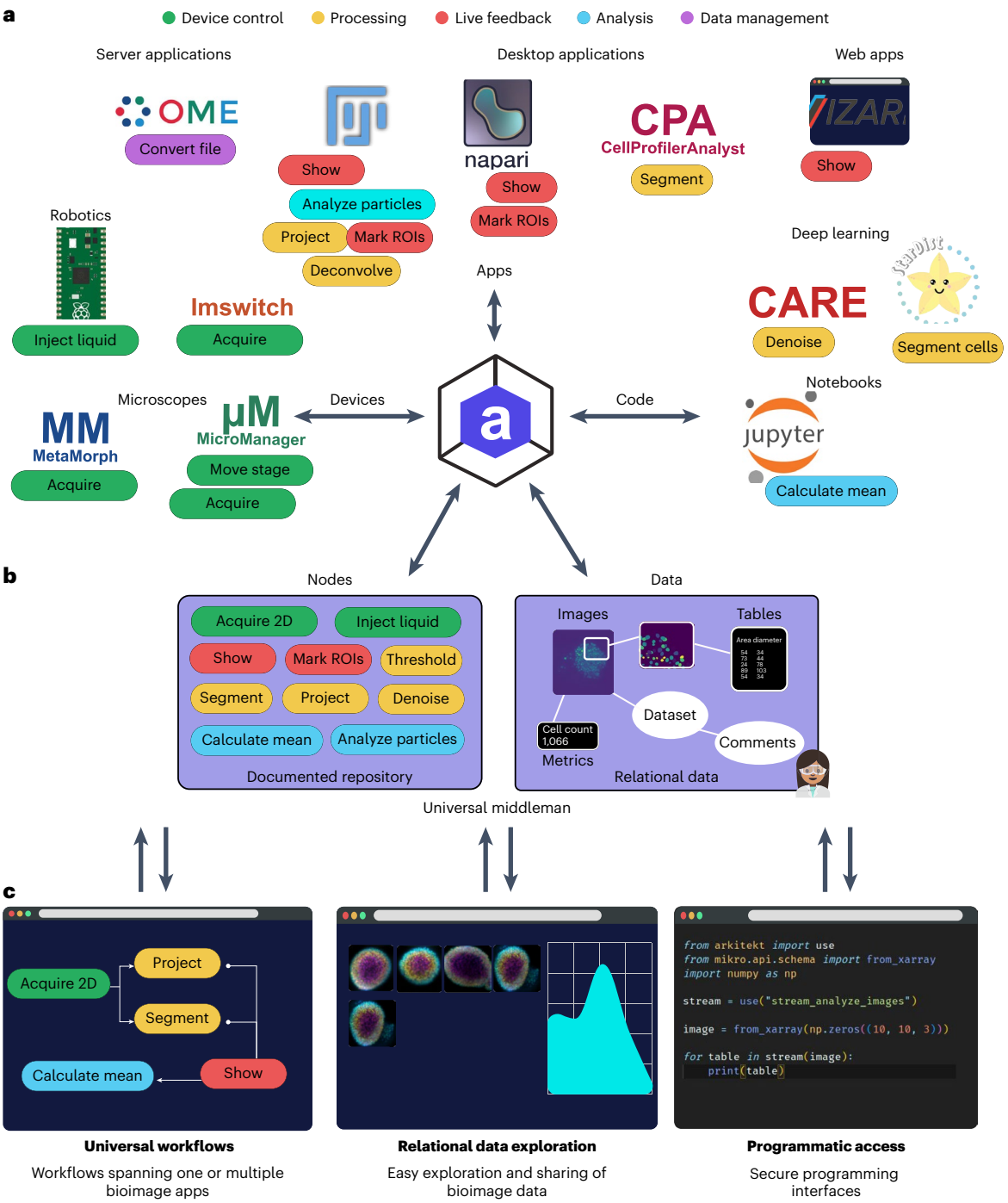


Fig. 1 | The core design of Arkitekt platform, serving as a middleman for real-time microscopy experiments and streaming analysis workflows. **a**, Arkitekt allows the interaction with various tools in different categories: devices, bioimage apps and custom code. These tools connect to Arkitekt platform (central a) and advertise their functionality as nodes. Nodes are abstract descriptions of the tool functionalities, including which bioimage data they can work with (for example, taking an image as an input and a ROI as an output). **b**, Arkitekt stands as the data hub for connected apps and makes their functionality accessible and inspectable for every app in the laboratory through remote calls. Arkitekt takes care of the communications and orchestration of all apps from data acquisition to postprocessing analysis, organizing and maintaining the data and metadata in a relational data graph. **c**, Arkitekt also provides an abstraction layer for users and applications alike to interface with the connected apps and data, both visually and programmatically. Workflows of connected functionality can be created through drag and drop. All of the data can be visualized directly in the web interface, while exploring associated metadata, or sent to one of the apps directly. Programmatic access is facilitated by easy-to-learn client libraries that require no advanced configuration and facilitate the easy inclusion and dispersion of new custom functionality as plugins. Logo credits: Fiji, © 2007 Free Software Foundation, Inc.; Napari, reproduced under a Creative Commons license CC0 1.0; Stardist, © 2018–2024, Uwe Schmidt, Martin Weigert; Imswitch, © 2007 Free Software Foundation, Inc.; Jupyter, reproduced under a Creative Commons license CC0 1.0; OME, OpenMicroscopy.org.

Core design concepts

Arkitekt has been designed around four core concepts (Fig. 1):

- Apps represent software that can connect to the Arkitekt framework, and whose functionalities can be integrated and executed in workflows. Apps can be stand-alone software such as Fiji, Napari or MicroManager, but also Python scripts, Jupyter Notebooks, websites, containerized apps (powered by Docker) and network-connected devices controlling hardware and robotics (the Internet of Things) (Fig. 1a). These apps negotiate access rights with the platform and when being used are systematically authenticated on a user and application basis, protecting users' data from potentially harmful external apps (for example, running in the cloud). Once authenticated, apps can advertise their functionalities through implementing nodes, which are then available to be used in workflows.
- Nodes represent a conceptual bioimage task, for example, 'Show an image', 'Acquire', 'Denoise', 'Threshold', 'Segment nuclei' and so on. They are defined through their data inputs and outputs, specific for each task, as well as by a description of this task (Figs. 1b and 2 and Supplementary Fig. 3). Apps themselves provide an implementation of this specific task. As multiple apps may provide the same functionality, they can potentially implement the same node. For example, both ImageJ and Napari can display images and mark ROIs. When performing analysis tasks, users can choose either to use one specific app (Supplementary Video 9), or to distribute the tasks amongst multiple apps in parallel (Fig. 3). Nodes can be orchestrated and combined in a user-friendly graphical manner to create workflows.
- Workflows are graphical pipelines composed of nodes that orchestrate bioimage tasks (Figs. 1c and 2). They can be linear (for example, Acquire → Deconvolve → Segment → Display), but also conditionally diverge (for example, 2D Acquire → Analyze → Filter if Condition = True → 3D Acquire) and loop back. Workflows can be used to both orchestrate local functionality within an application, serving a similar purpose to an ImageJ Macro, or to orchestrate multiple apps together in overarching data pipelines.

  When designing workflows, the user connects nodes and their respective inputs and outputs, manipulating and transforming the stream of data (Supplementary Video 1). This stream of data is conceptually central to Arkitekt workflows and allows data to be transformed in real time directly after creation. Arkitekt's interface not only guides users in correctly combining the nodes but also provides reactive 'helper nodes' that allow to deal with the unexpected pitfalls of real-time analysis (that is, to buffer or delay incoming data to accommodate a later processing).
- Structures. Arkitekt also takes care of storing and distributing analysis data among the apps through the concept of structures, which represent the inputs and outputs of nodes. Structures describe primitive data types, such as numbers and strings, but also dedicated bioimage data and metadata such as images and ROIs as well as comments or deep learning models. They can be read from and created on the platform, and are accessible from all apps through Arkitekt's application programming interface (API) and open data protocols (see 'Data management' section).

Once designed, users can deploy workflows through the web interface from anywhere. During the deployment, workflows get associated with a scheduling app that will take charge of the task orchestration during execution. Once deployed, a workflow becomes conceptually just another node that can be easily integrated in other workflows (Fig. 4).

When executing a workflow, Arkitekt automatically schedules the tasks to execute on the connected apps, using worker-based parallelism to speed up the analysis when needed. The progress of an ongoing workflow run can be monitored in real time on the same web interface by the user, visualizing all data transformations during execution (Supplementary Video 2). Arkitekt also keeps track of all events in the workflow and can provide the user with a 'replay' of the events in chronological order, enabling to debug an entire workflow and its data from start to end. As this information is associated with the created data, Arkitekt can ensure full data provenance at every workflow step.

By relying on apps, nodes and structures, workflows become universal, reproducible and implementation agnostic (Supplementary Fig. 1). Workflows can be shared between laboratories with different hardware and software (for example, the same workflow can work for a laboratory that uses MicroManager, Imswitch[18] or MetaMorph as microscope control software). As workflows are automatically versioned, they can be run on newer and older versions of the software if the implementation of the nodes (inputs, outputs and procedures) has not changed. Conversely, workflow runs keep a complete record of which implementation was used and establish a track record of data alteration that can be exported and inspected outside of the platform.

Core technology

Real-time performance. Arkitekt is designed to meet the demands of modern microscopy in data size and analysis speed. By default, workflow tasks are scheduled with low latency (100 ms range) via websocket-powered network connections from the Arkitekt server, and data are stored and retrieved from the central storage on demand. However, if network speed presents an issue or if there is no need to go through the network, workflows spanning in-app nodes (that is, consecutive nodes running inside the same app) can entirely run locally and manipulate data in memory.

Data management. Arkitekt takes care of the microscopy data flow through a central storage solution (Mikro), which can be hosted either locally (on a folder on the server) or remotely (that is, in the cloud). It allows for managing and discovering relationships within the raw data and metadata and is built on top of a relational database (PostgreSQL) for metadata and an open-source S3 Storage (MinIO) for binary data storage (Supplementary Fig. 2).

Traversing the complex relationships between images and elements, such as ROIs and higher-order features (for example, cell morphology parameters), is facilitated by a web interface. Advanced users can rely on standards such as structured query language (SQL) to explore the data, or can use the GraphQL API, which allows central and type-safe programmatic access to the relationship graph.

Arkitekt's data management is adapted to deal with gigabyte to terabyte datasets. Binary storage is based on the Zarr format, an efficient open-source lossless compressed format allowing for concurrent, distributed access to large multidimensional microscopy data. The data layer also allows real-time updates (through websocket subscriptions), which enables both data monitoring and Google Docs-like collaboration features such as real-time shared annotations (Supplementary Video 3). Once the specifications of the open microscopy environment next generation file format (OME-NGFF)[19] file format has matured, Arkitekt will support on-the-fly conversion to OME-Zarr files.

Arkitekt's data management is entirely opt in, and apps can implement nodes that process data from other popular bioimage data management solutions such as OMERO[20]. When enabling a dedicated optional Arkitekt service, data from OMERO can be directly managed in the desktop and web interface, and used and tracked in workflows. It should be noted, however, that due to the design of OMERO, lazy loading and fast and low-latency access to data are not feasible, and Mikro currently remains the data backbone for most apps developed for Arkitekt.

https://doi.org/10.1038/s41592-024-02404-5

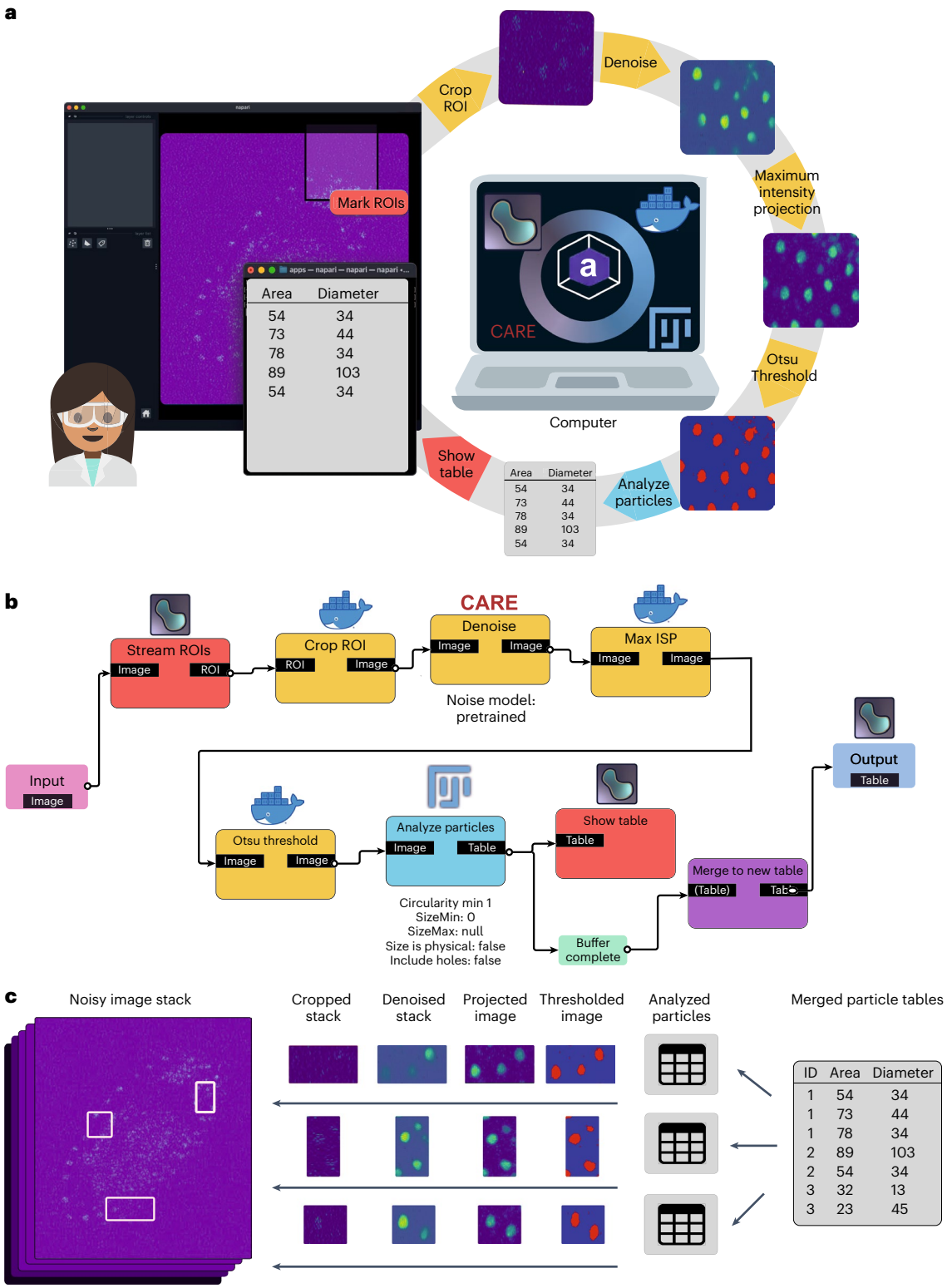


Fig. 2 | Arkitekt as a broker in interactive analysis workflows on a single computer. a, On invocation, this workflow will send the input image stack to a connected Napari instance, and ask the user to mark ROIs. These ROIs are then cropped, denoised via the CARE denoising deep-learning algorithm, maximum intensity projected (Max ISP) and thresholded via Otsu's method inside a virtualized Python plugin inside the Arkitekt Instance. The thresholded image is then send to FiJi instance, which will run the 'analyze particle' plugin, and finally display the result table inside the Napari instance (Supplementary Video 2). **b**, A one-to-one representation of the analysis workflow as constructed on the Arkitekt web UI. The initial given image is manipulated to a stream of ROIs that pass through the processing pipeline (yellow nodes). Each analyzed table is both sent to Napari and buffered, so that on ROIs marking termination, all tables are merged to one big data table. **c**, The automatically built data graph: as Arkitekt keeps track of the workflow steps, exploration of the data post hoc is facilitated through link navigation and can be easily queried programmatically through its type-safe API or directly through SQL queries of its database. Logo credits: Fiji, © 2007 Free Software Foundation, Inc.; Napari, reproduced under a Creative Commons license CC0 1.0; Docker, © 2013–2021 Docker, Inc.

https://doi.org/10.1038/s41592-024-02404-5

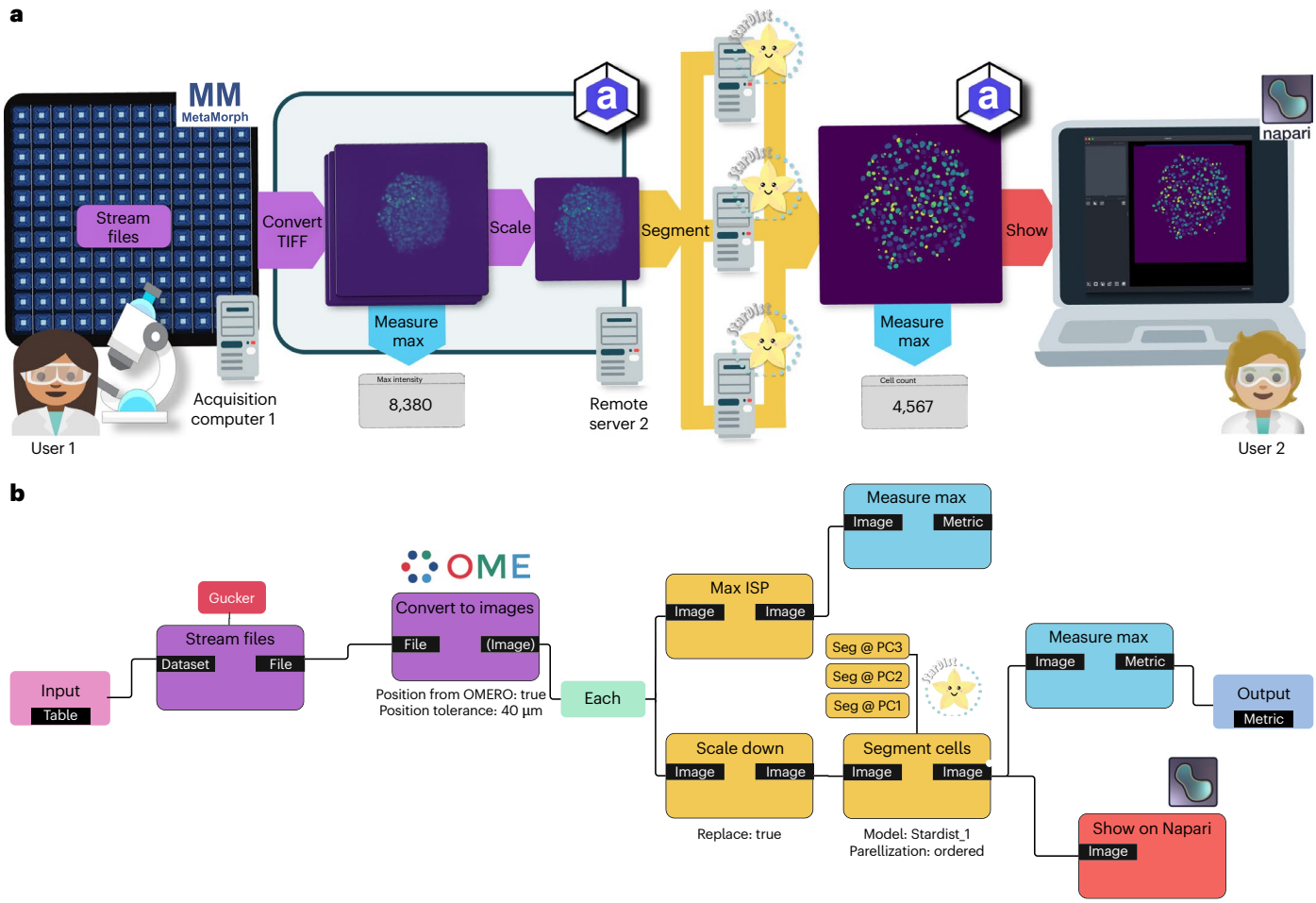


Fig. 3 | Arkitekt facilitates parallelizable workflows on multiple computers. a, In this streaming analysis workflow, live acquired light-sheet TIFF image stacks of liver organoids are pushed to the Arkitekt server, which converts both data and its OME metadata into elements in its relational data graph (recontextualizing position data in one unified 'stage' context representing the physical microscope stage). The converted 3D image stacks are then scaled down and segmented in round-robin parallelization on three connected computers (PC1, PC2 and PC3, each running the segmentation app (Seg)). The resulting segmentation is then measured for cell count and shown on a remote Napari instance for inspection. **b**, The 'stream files' nodes establish the initial data stream by watching a predefined microscopy folder. Each raw image file gets combined with a position and time context (every workflow 'run' will be put into a new 'stage') and converted. Workflow paths then split to both project and scale down the image in parallel. Here, the segmentation node was instructed to parallelize on three connected apps. **c**, The data can be explored on the web interface with live updates of the defined metric plotted in graphs, to enable easy monitoring of the data. **d**, A waterfall representation of the time (t, experimental timepoint) execution of the workflow. While the per item processing time does not change in the parallelized examples, the backpressure is alleviated by putting data items in separate queues. Every node is parallelizable by adding more connected apps. Logo credits: Napari, reproduced under a Creative Commons license CC0 1.0; Stardist, © 2018–2024, Uwe Schmidt, Martin Weigert; OME, OpenMicroscopy.org.

Flexible and powerful. Arkitekt was designed from its conception with the challenge to be user friendly for both end users and developers. Mouse and keyboard interactions facilitate data navigation and access to user options such as laboratory-wide search of functionality and data, as well as the creation of workflows (Supplementary Video 1).

Arkitekt ensures accessibility for non-programmers with little or no knowledge of programming. It comes with a drag-and-drop web interface that allows nonexpert users to create workflows by wiring nodes together. Arkitekt fully supports popular interactive applications such as Napari and its ability to load terabyte datasets directly

from the central server (plugin available through the Napari hub), as well as ImageJ and its ecosystem (through MikroJ app). Users can use their pre-existing installations and do not need specific configurations.

Arkitekt also provides a comprehensive plugin system that allows the discovery and one-click installation of containerized applications from GitHub repositories. These plugin apps can then be run sandboxed (with fine-grained access rights to data) on the Arkitekt server or on a dedicated computing resource such as a graphics processing unit (GPU) server, without the need for a complex developmental setup.

Powerful environment for developers. Arkitekt is extensible with different programming languages by adapting an 'API first' approach, exposing its functionality in a comprehensive GraphQL API. We chose GraphQL over representational state transfer (REST) to guarantee end-to-end type-safety and easy querying of complex relational data as well for its ability to provide an interactive API documentation right from the developer documentation. Additionally, Arkitekt comes with tested and documented software development kits (SDK) for both Python and Typescript. In practice, enabling a Python script as an app on the platform takes usually just a line of code (a decorator). The SDK will then inspect function parameters and documentation to enable it as a node (Supplementary Fig. 3). Through this process, Arkitekt can be used to generate a GUI for Python and Typescript scripts without necessitating a complex setup and the steep learning curve of GUI development, akin to popular tools such as MagicGUI (Supplementary Video 4). Since Arkitekt is built around open web protocols and a comprehensive API, it is easy to also bridge its functionalities to other programming languages.

Extensible and complementary. Arkitekt aims to be a platform for distributed concurrency and does not aim to replace dedicated distributed computing platforms such as SLURM or Dask that excel at parallelization. Conversely, it intends to integrate and complement them by providing more unified API and user interfaces. Thus, an experimental plugin Kluster allows users to visually create and provide (Dask) clusters to apps that can in turn schedule and distribute tasks on the cluster.

Installation and scaling. Arkitekt is primarily built to be hosted on computers that have fast low-latency network access to its connected apps. It is therefore preferably installed on site, handling most tasks on the local network. Only one computer with administrator rights is needed for the installation of Arkitekt. Any additional workstation does not require administrator rights, as long as it can access the server computer via network. To ensure an easy installation and updating process of the platform in local scenarios, Arkitekt comes with a graphical administration tool. However, as Arkitekt is based on secure web standards and uses cloud-native technologies, it can be readily connected to the wider internet and accessed from anywhere. It is fully adapted to be hosted on an institute level, with one server able to support hundreds of users and teams, each with their own data. The Arkitekt documentation comes with a few common examples illustrating different strategies for its deployment from testing environments to a laboratory-wide system.

Extensive documentation can be found at https://arkitekt.live/.

Results

We illustrated Arkitekt's core strengths and wide applicability through three common bioimage analysis scenarios. First, a classic offline interactive analysis, where already acquired data are analyzed by three different software (FiJi[7], CARE[21] and Napari[8]) on a single computer (Fig. 2). Second, a more advanced streaming analysis workflow, where acquired data (MetaMorph) are live monitored, processed and analyzed using different software (Stardist[22], Napari[8] and an OMERO-based file reader[23]). Here, the software are located on five different computers connected on a local network, one for acquisition, one for remote monitoring and three for parallel distributed computing (Fig. 3). Third, a closed-loop experiment, where real-time analytical output computed using two different software (Stardist[22] and the Arkitekt standard library plugin (stdlib)) is looped back to the ongoing microscopy experiment (MicroManager[4,5]) to perform specific actions (Fig. 4). All three workflows are described below and are available online to be used as templates (https://arkitekt.live/docs/showcases).

Single computer interactive analysis workflow

In this modern offline analysis workflow, we show how Arkitekt allows to combine different bioimage processing, analysis and visualization functionalities, including a deep-learning restoration algorithm (CARE)[21], all on a single computer. We demonstrate that tasks from different popular software platforms, here Fiji[7], containerized Python scripts and Napari[8], can be combined in the same workflow (Fig. 2). We also illustrate how to use the interactive capabilities of the GUI apps during the workflow execution to manually define ROIs.

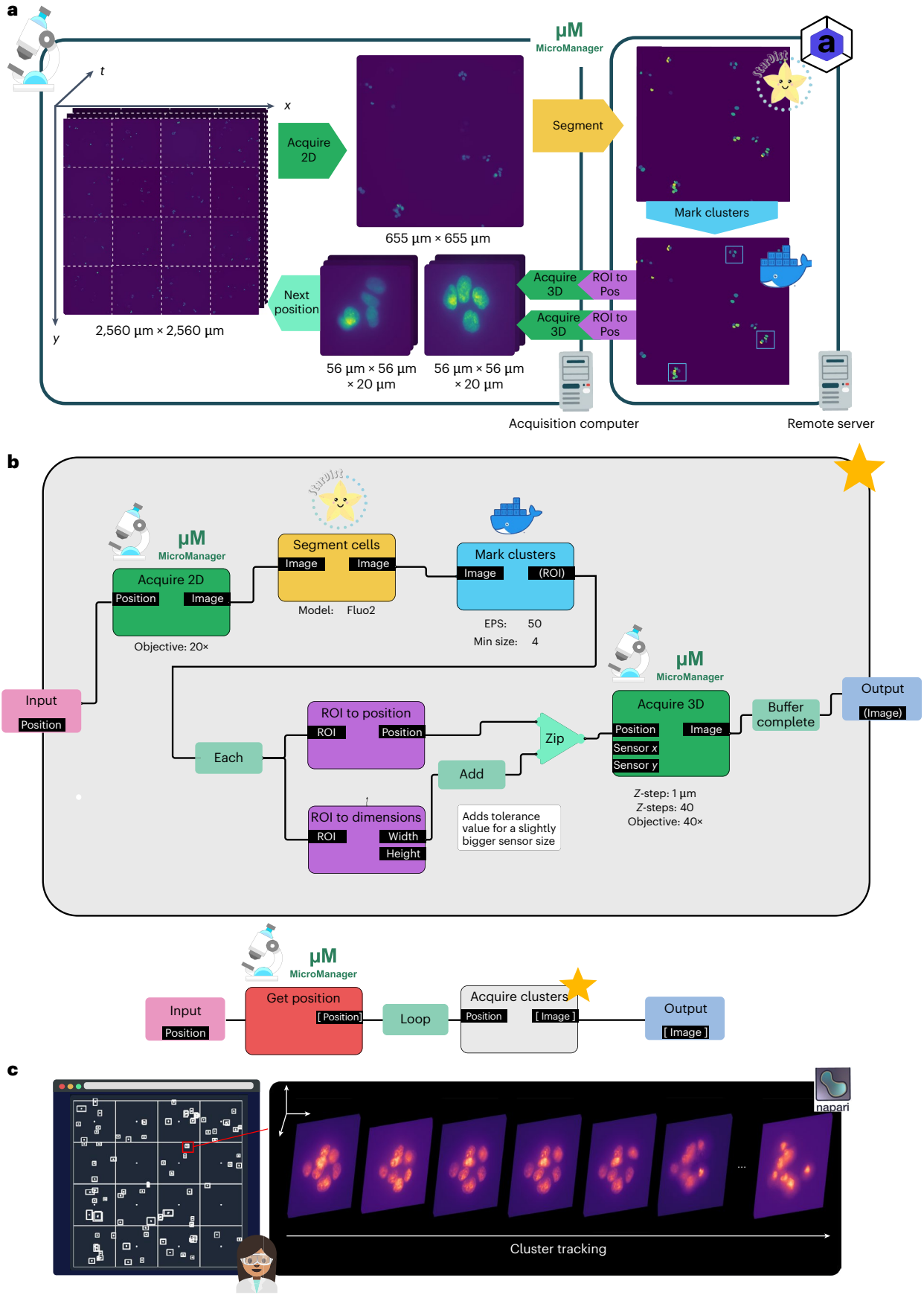
In this example, a noisy fluorescence image stack (*Tribolium*[21]) was visualized in Napari and a user was prompted to mark ROIs. Interactively and in parallel, marked ROIs were cropped from the original image and denoised through the CARE[21] algorithm, binarized using Otsu auto-thresholding[7] and analyzed through the particle analyzer in Fiji[7]. Finally, the resulting quantifications returned from the particle analyzer were visualized back to the user in Napari and on the web interface for inspection (Supplementary Video 2).

This workflow illustrates how Arkitekt allows aggregating functionalities from several popular software, passing the data and metadata from one platform to another via a GUI. Importantly, this workflow not only substantially relieves the user burden of having to use three different software one after the other, but it also avoids the potential pitfalls of data management. Indeed, Arkitekt keeps track of the relationship between original images, marked ROIs, cropped images and filtered analysis of the cropped images, as well as all associated quantifications. This relationship and provenance graph can then be explored on the Arkitekt frontend or exported for further statistical analysis.

In this use case, Arkitekt and the connecting bioimage apps Napari and FiJi were installed on the same machine. CARE was installed on the Arkitekt server as a plugin following its one-click installation from a GitHub repository and run within the Arkitekt setup through Docker virtualization. Setting up this workflow only required the additional installation of both the Arkitekt Napari plugin (through the Napari plugin manager) and the FiJi wrapper app (MikroJ, available as a GUI installable).

Fig. 4 | Arkitekt enables smart microscopy experiment. a, In a closed-loop workflow, a large FOV (2 × 2 mm^2) is acquired in an overarching multidimensional acquisition workflow with positions previously set interactively in MicroManager. Each acquired image is then offloaded to an Arkitekt instance running on a GPU-powered machine, that segments images using Stardist deep-learning segmentation and marks ROIs around clusters based on DBSCAN clustering analysis. Each ROI is then transformed back into stage coordinates, and necessary camera dimensions are calculated to fit the acquire cluster with a 40× objective. Once all the detected clusters are acquired at 40× magnification, the next multidimensional acquisition position (pos) is acquired. **b**, A subworkflow, marked with a yellow star, integrated into a larger multidimensional acquisition type workflow that loops over all active microscope positions every 20 min. Helper nodes are employed to add a dynamic tolerance and to synchronize separate data streams in time (buffering, zip). **c**, Cluster tracking thanks to the stored position saved in the metadata. As the camera is always centered on the center of mass of each detected cell cluster, clusters can be easily monitored through time without any specific tracking routine by lasso selecting a detected cluster position on the stage view in the web interface. These selections can then be easily monitored as a timeseries in Napari. Logo credits: Napari, reproduced under a Creative Commons license CC0 1.0; Stardist, © 2018–2024, Uwe Schmidt, Martin Weigert; Docker, © 2013–2021 Docker, Inc.

https://doi.org/10.1038/s41592-024-02404-5



Live monitoring and streaming analysis

Biological systems can be highly reactive, and closely monitoring them during long-run acquisitions is necessary for ensuring a stable environment in an ongoing experiment. This can range from just checking that the acquisition is running properly (for example, no loss of focus), to monitoring the evolution of live biological samples (for example, premature cell death). Classically, images are batch processed once the acquisition is complete, leaving no chance for real-time analysis and feedback. In this second workflow, we illustrate how Arkitekt enables to display real-time feedback on the running acquisition.

Here, we used Arkitekt to perform real-time quantitative monitoring of single objective selective plane illumination microscopy (soSPIM)[24,25] 3D microscopy data using the popular StarDist[26] deep-learning segmentation algorithm (Fig. 3). A user-defined directory located on the microscope's computer was monitored for new data, and all the analysis steps were performed remotely in parallel with the acquisition, minimizing the risks of slowing the acquisition down or causing its crash.

As the deep learning-based segmentation can be highly time consuming, they were run in parallel, distributed on three different computers. Visualization of both the 3D acquisition and the results were performed in Napari[8] on a remote and mobile computer, as well as through the web interface, where results (here the number of nuclei and the average volume) were plotted on a live dashboard (Supplementary Video 5).

Practically, multidimensional acquisition (3D, 20 positions, 30 time points every 20 min) of 3D cell cultures composed of HEP-G2 cells were acquired automatically using the soSPIM technology[24,25] and MetaMorph software. The 3D stacks were automatically saved to a folder, which was selected in the File Watcher app (Gucker), enabling the 'Stream Files' node in the Arkitekt workflow (Fig. 3a). Each time a new 3D stack was saved in this folder, it was uploaded to the remote central server hosting Arkitekt and converted to Arkitekt's distributed image format, integrating the OMERO metadata into the database. Images were then visualized by 3D maximum intensity projection, and the maximum intensity was computed as a quality metric and displayed. Simultaneously, incoming images were downscaled according to the metadata to match the pixel size that the Stardist[26] deep-learning model was trained on. The segmentation was then distributed among the Arkitekt server and two additional remote workstations, reducing the processing time and alleviating the bottleneck of Stardist segmentation (Fig. 3a,b). Segmentations were then piped directly back to a mobile Napari instance, providing immediate feedback to any remote computer for user inspection (Fig. 3a). As the acquisition proceeded, the past and current progress of the workflow could be monitored remotely and live on the web interface (Supplementary Video 6). The ability to parallelize the Stardist segmentation on three remote computers allowed the analysis to be performed in real time with the acquisition. A waterfall diagram representation allows to visually monitor the time evolution of the workflow (Fig. 3d and Supplementary Video 5).

Smart microscopy

Where live streaming analysis still deals with a unidirectional data flow away from the microscope, 'smart microscopy' allows analytical insights to be fed back to the microscope to adjust or optimize acquisition parameters. To illustrate Arkitekt relevance for smart microscopy, we created an illustrative example of a closed-loop analysis workflow, involving multidimensional acquisition and analytical feedback loop from a deep-learning-based quantitative analysis readout (Fig. 4). By offloading the analysis, the microscope software could remain dedicated to providing acquisition functionality only, establishing a clear separation of concern.

In this workflow, the user interactively sets up within the MicroManager[4,5] acquisition software a four-by-four view grid over live cancerous fluorescent cells placed under an inverted microscope, with a 20× objective, providing an acquisition field of view (FOV) of 2.4 × 2.4 mm^2 (Fig. 4a). When the workflow started, two-dimensional (2D) images of the recorded stage positions were acquired in time-lapse mode, every 30 min for 24 h (768 images). For every acquired image, nuclei were automatically segmented through the StarDist[26] deep-learning algorithm running within the Arkitekt remote server. Then, a DBSCAN[27] clustering plugin, also running within Arkitekt, allowed to identify cell clusters, here defined as groups of five or more aggregated cells. Establishing real-time feedback, when one or more cell clusters were detected, the stage coordinates corresponding to their centroids were sent back to MicroManager. The 3D stacks (25 optical sections) of ROIs tailored to the dimensions of each of these clusters were then collected at higher magnification (40×), which could be sent on demand for visual inspection to a connected Napari instance. After all the clusters were acquired at high magnification in 3D, the microscope returned to the 20× magnification to continue the large FOV 2D time-lapse acquisition.

Through this adaptive strategy, we were able to reduce the data load of the experiment from 2.5 TB (if the same field of view was acquired at 40× magnification in 3D) to 50 GB. In total, 21,186 nuclei were segmented, from which 2,027 clusters were identified and acquired in 3D. The acquisition time dropped from 29 min per time point to 7.30 min.

This exemplary workflow illustrates Arkitekt's ability to provide a no-code smart microscopy platform, utilizing modular building blocks (Fig. 4b). It is also possible to use the entire, or part of the workflow to create a node that can be used in a bigger workflow (Fig. 4b). Furthermore, Arkitekt offers the possibility for the user to navigate through the acquired data using the graphical database during or after the acquisition (Fig. 4c). This workflow also demonstrates Arkitekt's capacity to integrate computational methods, which are not available in the acquisition software, into the decision loop.

Finally, based on the strength of Arkitekt's reactive programming paradigm, it is possible to take full control over the acquisition process, orchestrating and synchronizing timed events without relying on any higher-level features of the microscopy platform such as multidimensional acquisition tools.

Discussion

We introduce the Arkitekt software platform as an open-source solution to address the mounting challenges of modern bioimage analysis workflows. Through three simple examples, we demonstrated Arkitekt's capability to combine multiple bioimage apps and scripts interactively, providing real-time distributed analysis in a multidimensional microscopy framework, and its applicability as a scheduler for smart microscopy. As compared with the young body of literature, Arkitekt stands out in terms of its interoperability via a computational and data backbone that seamlessly connects multiple bioimage apps and script, tailored to the scientist's specific needs. It also provides a unique set of features dedicated to the orchestration of these tools interactively and in real time, allowing advanced data flow and facilitating interactive analysis. These features are enabled through a set of abstractions that allow nonexperts to design parallelizable and universal workflows visually, without having to worry about the underlying hardware. In accordance with the principles of FAIR[28], it employs strategies to address the needs of modern and secure scientific data and metadata management, providing the user with the ability to explore their data graph programmatically and in usable interfaces. Arkitekt platform is fully operational and comes with a packaged set of supporting libraries and an emerging interactive online documentation (https://arkitekt.live/).

However, given its nature as an early built framework, some features are still in experimental status and require a few iterations of user and developer feedback cycles to achieve maturity. This includes more dedicated client libraries for its open web APIs in other programming languages, which are currently only developed for Python

and JavaScript/Typescript. Especially, first-class support for the Java ecosystem could help in transitioning more bioimage scripts beyond ImageJ into the platform.

Designed for orchestrating tasks on multiple desktop workstations, Arkitekt's apps and nodes are well placed to handle the orchestration of tasks on high-performance computing (HPC) resources. Here, an Arkitekt app could be imagined that would run (node) tasks not locally in memory but instead submit them via a HPC job scheduler such as SLURM that could on-demand allocate resources. End users could then rely on the same mechanisms for task assignment and visually build reactive workflows spanning HPC resources and desktop bioimage apps.

Arkitekt design emphasizes separation of concern, and the vast part of its developed ecosystem could readily extend to other modalities or experimental settings. One potential application could be the world of behavioral experiments that share a similar need for closed-loop experiments. Feeding analytical insights of the animal's behavior back into the ongoing experiment would open new avenues for reactive monitoring of neuronal activities during behavior-dependent stimulation. A broad software arsenal in this space, such as Bonsai[29] and more recently Autopilot[30], have already shown the merit of applying reactive and distributed workflows, and can also integrate deep-learning algorithms, such as shown in DeepLabCut[31]. Joining efforts in finding a unifiable solution to both worlds could spark new experimental ideas, such as combining behavior-driven (for example, through pose estimation) and neuronal event-driven (for example, through in vivo calcium imaging) closed-loop experiments.

Recent discourse highlights the potential benefits of integrating intelligent agents more closely within microscopes to enable sophisticated interactions via language[32]. However, despite their advanced capabilities, these models are not yet adept at understanding complex multilayer systems programming, as employed in microscope control software, but excel at integrating with high-level interfaces, such as NumPy or Pandas, and when automating routine tasks. Arkitekt nodes, with their well-documented interfaces, could serve as a crucial interaction layer for large language models to interface with hardware, providing an orthogonal additional way of automation to workflows and enabling 'conversational microscopy'.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-024-02404-5.

References

1. Mahecic, D. et al. Event-driven acquisition for content-enriched microscopy. *Nat. Methods* **19**, 1262–1267 (2022).
2. Belthangady, C. & Royer, L. A. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nat. Methods* **16**, 1215–1225 (2019).
3. Moen, E. et al. Deep learning for cellular image analysis. *Nat. Methods* **16**, 1233–1246 (2019).
4. Edelstein, A., Amodaj, N., Hoover, K., Vale, R. & Stuurman, N. Computer control of microscopes using MicroManager. *Curr. Protoc. Mol. Biol.* **14**, 14.20 (2010).
5. Edelstein, A. D. et al. Advanced methods of microscope control using muManager software. *J. Biol. Methods* **1**, e10 (2014).
6. Rueden, C. T. et al. ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics* **18**, 529 (2017).
7. Schindelin, J. et al. Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).
8. Sofroniew, N. et al. Napari: a multi-dimensional image viewer for Python. *Zenodo* https://doi.org/10.5281/zenodo.3555620 (2022).
9. Stirling, D. R. et al. CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics* **22**, 433 (2021).
10. Sheffield, N. C. et al. From biomedical cloud platforms to microservices: next steps in FAIR data and analysis. *Sci. Data* **9**, 553 (2022).
11. Di Tommaso, P. et al. Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
12. Galaxy, C. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. *Nucleic Acids Res.* **50**, W345–W351 (2022).
13. Prigent, S. et al. BioImageIT: open-source framework for integration of image data management with analysis. *Nat. Methods* **19**, 1328–1330 (2022).
14. Ouyang, W., Mueller, F., Hjelmare, M., Lundberg, E. & Zimmer, C. ImJoy: an open-source computational platform for the deep learning era. *Nat. Methods* **16**, 1199–1200 (2019).
15. Alvelid, J., Damenti, M., Sgattoni, C. & Testa, I. Event-triggered STED imaging. *Nat. Methods* **19**, 1268–1275 (2022).
16. Beghin, A. et al. Localization-based super-resolution imaging meets high-content screening. *Nat. Methods* **14**, 1184–1190 (2017).
17. Gerst, R., Cseresnyes, Z. & Figge, M. T. JIPipe: visual batch processing for ImageJ. *Nat. Methods* **20**, 168–169 (2023).
18. Casas Moreno, X., Al-Kadhimi, S., Alvelid, J., Bodén, A. & Testa, I. ImSwitch: generalizing microscope control in Python. *J. Open Source Softw.* https://doi.org/10.21105/joss.03394 (2021).
19. Moore, J. et al. OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies. *Nat. Methods* **18**, 1496–1498 (2021).
20. Allan, C. et al. OMERO: flexible, model-driven data management for experimental biology. *Nat. Methods* **9**, 245–253 (2012).
21. Weigert, M. et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nat. Methods* **15**, 1090–1097 (2018).
22. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. AnchorCell detection with star-convex polygons. *In Medical Image Computing and Computer Assisted Intervention MICCAI* 265–273 (2018).
23. Besson, S. et al. Bringing open data to whole slide imaging. *Digit Pathol.* **2019**, 3–10 (2019).
24. Galland, R. et al. 3D high- and super-resolution imaging using single-objective SPIM. *Nat. Methods* **12**, 641–644 (2015).
25. Beghin, A. et al. Automated high-speed 3D imaging of organoid cultures with multi-scale phenotypic quantification. *Nat. Methods* **19**, 881–892 (2022).
26. von Chamier, L. et al. Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat. Commun.* **12**, 2276 (2021).
27. Ester, M., Kriegel, H., Sander, J. & Xu, X. *Proc. 2nd International Conference on Knowledge Discovery and Data Mining* 226–231 (1996).
28. Wilkinson, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data* **3**, 160018 (2016).
29. Lopes, G. et al. Bonsai: an event-based framework for processing and controlling data streams. *Front. Neuroinform.* **9**, 7 (2015).
30. Saunders, J. L. & Wehr, M. Mice can learn phonetic categories. *J. Acoust. Soc. Am.* **145**, 1168 (2019).
31. Mathis, A. et al. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat. Neurosci.* **21**, 1281–1289 (2018).
32. Carpenter, A. E., Cimini, B. A. & Eliceiri, K. W. Smart microscopes of the future. *Nat. Methods* **20**, 962–964 (2023).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2024, corrected publication 2024

Methods

Design principles

Arkitekt is designed as a modular server platform that encapsulates various (micro-)services to separate the concerns of the platform and can be distributed separately and stand alone. Core components are as follows: Lok, the authentication and authorization backend and configuration service; Rekuest, the task scheduler that manages task assignment to connected apps; Fluss, the workflow design service; and Port, a service that allows for the management of virtualized plugin apps in Docker containers. Mikro is the only service of the Arkitekt platform that focuses specifically on the management of microscopy data and metadata. Supplementary Fig. 1 provides an overview of their purpose and interplay for a simple workflow example.

Through Konstruktor, Arkitekt's installer, the user can decide which services to install and how to manage database access, as well as whether to install other popular open-source services such as OMERO[20].

Arkitekt follows an API first design, which leads to a unified, versioned and documented programming interface based on common web protocols such as GraphQL, S3 and websockets. Every software that connects to Arkitekt connects to this same API (even its own web interface). With this unified API, a contract is provided between connecting apps and the Arkitekt backend. Given that every software project respects this contract, they can develop completely independently, yielding smaller maintainable software pieces.

As Arkitekt can be deployed to run anywhere from one computer to being self-hosted on an institute server and even in the cloud, the platform ensures adequate security and protection of the data from third parties. Built around OpenID and Oauth2, Arkitekt allows for fine-grained permission and access policies on both the user and application level.

Implementation details

Tech stack. In its current version, Arkitekt services are primarily built with Python and TypeScript/JavaScript, using Django on the backend and React on the frontend. Both technologies were chosen for their maturity, general popularity and availability of documentation, to make the onboarding of new developers as easy as possible. Performance-critical parts of Arkitekt (such as task scheduling) interface with other open-source software solutions such as RabbitMQ and Redis. These two solutions were chosen because of their low-latency task assignment and ease of maintenance (as opposed to distributed message brokers such as Apache Kafka). Arkitekt supports a variety of SQL databases, but by default comes with PostgreSQL. For binary data storage (for example, images or big data tables), Arkitekt uses the S3 standard, which in its default configuration is backed by MinIO, an open-source self-hosted S3 alternative. All Arkitekt services are running as containerized web applications (see Supplementary Fig. 1 for the description of services interplay).

Performance. Arkitekt is built around Zarr (for image data) and Apache Parquet (for tabular data) to enable fast and lazy loaded data access for datasets spanning from megabytes to terabytes. Arkitekt provides only little abstraction (no web server roundtrip) around these technologies and securely wraps them for performant access. Tools such as Napari can take advantage of this tech stack to completely virtualize image loading.

Task scheduling. Arkitekt can distribute analysis workflows and tasks through either a distributed or a local execution. In the distributed environment, Arkitekt uses an asynchronous message queue (with RabbitMQ by default) to schedule work on apps, allowing for concurrent execution of assigned tasks. Optimizing for low-latency task scheduling, Arkitekt does never send binary data to the app directly, but passes references to datasets that can then be loaded lazily from the central server. Data that are not needed does not need to be loaded. All tasks are cancellable through programmatic and graphical interfaces, enabling the termination of complex workflows, acquisitions or training sessions. Errors are propagated and inspected for the assigning app or user, and automatic retries in case of failure are available. Logs of the task execution can be inspected on the web interface. For local execution, apps (only Python apps supported at this point) can become their own scheduler which enables zero-latency local assignment of tasks, only resorting to network calls when functionality is not locally present.

Client libraries. Any software and programming language can interact with Arkitekt through its web API. For Python and TypeScript (react based), we provide a comprehensive client that enables users to interact with Arkitekt through native function calls or expose part of their software as a node on Arkitekt by simply adding a decorator. The Python library fully supports Jupyter notebooks, and GUI applications and comes with a set of example apps that illustrate the ease of integration. The library can also be used to package and deploy Python apps as Docker containers and make them installable with one click on any Arkitekt instance. Both client SDKs (and their dependencies) are typed, tested and documented online.

Workflow methods

Workflow I. *Installation*. Docker Desktop V20.04 was installed on a desktop computer (Windows 10, Nvidia GPU 2080Ti, Intel i9 computer processing unit (CPU), 1 TB solid state drive (SSD)). The Arkitekt installer (Konstruktor) was download from its GitHub Repository (https://github.com/arkitektio/konstruktor) and started. The platform was installed following its guided installer, choosing a single user setup. The Arkitekt service was started and Konstruktor instructed to advertise the Arkitekt installation on the network.

Workflow specific installation. After installation, both the bioformats reader-based OMERO conversion plugin (https://github.com/arkitektio-apps/omero) and the standard processing plugin (https://github.com/arkitektio-apps/stdlib), as well as the CARE plugin (https://github.com/arkitektio-apps/kare) and the remote workflow scheduling plugin Reaktor (https://github.com/arkitektio-apps/reaktor) were installed via the 'Plugin Pane' from their respective GitHub repositories and started as virtualized internal containers managed by the platform. The MikroJ app was downloaded from its repository (https://github.com/arkitektio-apps/mikroj), installed and configured to point to a previously installed instance of Fiji/ImageJ[5]. Napari[6] 4.17 was installed on the system and the Mikro-Napari plugin installed through the interactive code terminal inside the Napari environment, via a subprocess call (necessary step as the plugin was not yet available on the Napari plugin hub page). All apps were connected and the user authenticated with the platform.

Data preparation. The original CARE dataset *Tribolium*[21] was downloaded from its data repository (https://publications.mpi-cbg.de/publications-sites/7207/) and unzipped. A new dataset was created through the Orkestrator user interface (UI), and the images contained in the dataset were uploaded through drag and drop. To convert the uploaded raw image data (TIFF files) to the distributed Zarr-based format on the mikro server, a conversion node 'Convert File' was searched and reserved on the web UI, directly linking it to the OMERO Conversion plugin. All image files in the dataset were selected using multiselect and converted through drop drown and Convert File (Batch) assignment.

Model preparation. The converted images of the original dataset were inspected through the Arkitekt web interface, and three corresponding images of low and high signal-to-noise ratio were associated by dragging one over the other, labeling them as 'ground-truth' in same context ('CARE training set') inside the pop-over Relate Dialog. The CARE app providing the node train CARE model was reserved and run directly from the drop-down menu of the newly created context, specifying

the 'ground-truth' relation as the training relation. The progress of the training was inspected on the web UI (Supplementary Video 8).

Workflow design. The workflow, as described below, was designed on the Arkitekt design pane, exported and deployed through the web interface on the Reaktor scheduling app, specifying Tribolium Analysis as the new node title.

Workflow run. The deployed workflow '*Tribolium* Analysis' node was reserved through the web interface. MikroJ and Napari were started as the state changes of the workflow were observed on the web interface. The workflow was then started on the web interface, by right clicking on a low signal-to-noise ratio image in the dataset and selecting the Tribolium Analysis assignment. During execution, ROIs were subsequentially marked on the opened ROI layer in Napari, intentionally selecting perceived areas of varying cell density and the loop backed results table inspected in Napari.

Workflow II

Installation. Docker Desktop V20.04 was installed on a desktop computer (Ubuntu 22.04, Nvidia GPU 2080Ti, Intel i9, 1 TB SSD). The Arkitekt installer (Konstruktor) was download from its GitHub repository (https://github.com/arkitektio/konstruktor) and started. The platform was installed following its guided installer, choosing a single user setup with Docker virtualization. The Arkitekt service was started, and Konstruktor instructed to advertise the Arkitekt installation on the network. The Tailscale virtual private network client was installed on the same machine, added to a shared private virtual network according to their documentation.

Sample preparation. *JeWell preparation*. JeWell, which are culturing devices with truncated pyramidal shape structure enabling soSPIM imaging, were fabricated and passivated to prevent cell adhesion as thoroughly described in Beghin et al.[25]. JeWells with a square top opening of 120 µm and a height of 100 µm were used for this experiment.

Spheroid culture. HEP-G2 cells stably expressing H2B–eGFP fusion protein were maintained in Dulbecco's modified Eagle medium (DMEM) (11965092, Invitrogen) supplemented with 10% fetal bovine serum (FBS) (10082147, Invitrogen), 1% GlutaMAX (35050061, Invitrogen), 1% penicillin–streptomycin (15070063, Invitrogen) and 1% sodium pyruvate (11360070, Invitrogen) at 37 °C and 5% CO_2. After being trypsinized, the cells were suspended in complete DMEM supplemented with 20% FBS, 1% GlutaMAX, 1% penicillin–streptomycin and 1% sodium pyruvate. The cell suspension was adjusted to 0.5×10^6 cells ml^{-1} and 200 µl of cell suspension was poured onto the JeWell plate for 10 min to allow for the cells to fall within the JeWells with approximately 80 cells per JeWell. Excess of cells were then removed by gently washing the plate with complete culture medium. After the wash, 2 ml of complete DMEM was added to the plate and the cells were cultured for 3 days at 37 °C and 5% CO_2, allowing them to form spheroids of approximately 150 µm in diameter.

Fixation and labeling. On the third day, the plate was rinsed two times with sterile PBS, and the 3D cell cultures were fixed for 20 min in 4% paraformaldehyde (28906, Thermo Fisher Scientific) at room temperature. The 3D cultures were then permeabilized for 30 min in 1% Triton-X-100 (T9284, Sigma-Aldrich) solution in sterile PBS at room temperature, followed by 24 h of incubation in blocking buffer (2% bovine serum albumin (37525, Thermo Fisher Scientific) and 1% Triton-X-100 in sterile PBS) at 4 °C on an orbital shaker. Samples were then incubated with 0.5 µg ml^{-1} 4,6-diamidino-2-phenylindole (62248, Thermo Fisher Scientific) at 4 °C for 24 h on an orbital shaker followed by three rinsing steps with blocking buffer and three rinsing steps with sterile PBS before being stored at 4 °C until imaging.

Acquisition preparation. The fixed samples were mounted under a Nikon Ti2 inverted microscope (Eclipse Ti2 series, Nikon), equipped with a 60× objective (water immersion (WI) 1.27 numerical aperture (NA), Nikon). The microscope, the motorized stages and the multidimensional acquisition process were controlled by MetaMorph software (Molecular Devices). Dedicated home-made plugins were integrated into MetaMorph to allow for the control of the soSPIM beam steering unit and sample illumination. They also allowed to precisely reposition each JeWell in the camera field of view, ensuring optimal 3D multiposition soSPIM as previously described[24,25]. MetaMorph was configured to run a multi-time point (30 time points every 20 min), multiposition (20 positions) 3D acquisition (60 *z*-steps, 1 µm step size) of the samples. The multi-time point acquisition was chosen to simulate a time-lapse experiment on live samples while allowing for the assessment of the reproducibility of the analytical pipeline. MetaMorph was instructed to put the resulting TIFF image stacks into a specific folder on the acquisition computer.

Workflow specific installation. After installation, the OMERO conversion plugin (https://github.com/arkitektio-apps/omero), the Standard Processing plugin (https://github.com/arkitektio-apps/stdlib), the Stardist segmentation plugin (https://github.com/arkitektio-apps/segmentor) and the remote workflow scheduling plugin Reaktor (https://github.com/arkitektio-apps/reaktor) were installed via the 'Plugin Pane' from their respective GitHub repositories and started as virtualized internal containers managed by the platform. Gucker, the Arkitekt enabled file watcher app, was installed through its installer (https://github.com/arkitektio-apps/gucker) on the microscope computer (Windows 10, Intel-i5) and pointed to the output directory of MetaMorph. Gucker was connected to Arkitekt through its GUI and authenticated with the platform.

Napari was installed on a portable desktop computer (MacBook Air M2, 2023) and the Mikro-Napari plugin installed as described in the previous workflow. The Notebook was configured to share the same private virtual network as the desktop and microscope computer utilizing Tailscale.

Docker Desktop was installed on two additional GPU-powered computers (both Windows 10, i7 Nvidia GPU 2080TI, 500 GB SSD) and two instances of the Stardist app were run through the terminal call of 'docker run --t jhnnsrs/segmentor arkitekt run easy', which downloaded and started the plugin, the Fakts connection link in the terminal was followed and the app was authorized on the platform.

Data preparation. A previously trained Stardist3D model[25] was zipped and uploaded through drag and drop on Arkitekt's web interface.

Workflow preparation. The workflow was created through the workflow interface, setting the uploaded 'Stardist soSPIM' model as default for the Stardist prediction node. The 'convert OMERO node' was given a position tolerance of 40 µm to merge motion-corrected positions together. The workflow was deployed and reserved.

Workflow run. The workflow was run, specifying a newly created dataset as the input. The MetaMorph acquisition was started on the microscope computer. During the acquisition, the workflow was monitored both on the Arkitekt web dashboard, back on the acquisition computer, as well as through inspecting the streamed segmentations on Napari, from the remote laptop. Napari was occasionally switched on and off, and moved to different networks, under assessment of the workflow still running.

Workflow III

Docker Desktop V20.04 was installed on a desktop computer (Ubuntu 22.04, Nvidia GPU 2080Ti, Intel i9, 1 TB SSD). The Arkitekt installer

Konstruktor was download from its GitHub repository (https://github.com/arkitektio/konstruktor) and started. The platform was installed following its guided installer, choosing a single user setup with Docker virtualization. The Arkitekt service was started, and Konstruktor instructed to advertise the Arkitekt installation on the network.

Workflow specific installation. After installation, the Standard Processing plugin (https://github.com/arkitektio-apps/stdlib), the Stardist segmentation plugin (https://github.com/arkitektio-apps/segmentor) and the remote workflow scheduling plugin Reaktor (https://github.com/arkitektio-apps/reaktor) were installed via the 'Plugin Pane' from their respective GitHub repositories and started as virtualized internal containers managed by the platform. Mikro-Manager was installed on the microscope computer through the installer hosted on https://github.com/arkitektio-apps/mikro-manager and pointed to a previously installed instance of MicroManager[5] (nightly version of 30.06.2023) (https://download.micro-manager.org/nightly/2.0/Windows/MMSetup_64bit_2.0.1_20230630.exe). All apps were connected and authenticated with the platform.

Sample preparation. *Cell culture*. HEP-G2 cells stably expressing H2B–eGFP fusion protein were maintained in DMEM (11965092, Invitrogen) supplemented with 10% FBS (10082147, Invitrogen), 1% GlutaMAX (35050061, Invitrogen), 1% penicillin–streptomycin (15070063, Invitrogen) and 1% sodium pyruvate (11360070, Invitrogen) at 37 °C and 5% CO_2.

Preparation. Cells were seeded at 30,000 cells ml^{-1} concentration on 18 mm round coverslips the day before the experiment and maintained under culture medium at 37 °C and 5% CO_2. On the experimental day, the existing medium on the 18 mm round coverslips with the HEP-G2 cells was replaced with a fresh batch of Flourobrite DMEM (A1896701, Gibco) supplemented with 10% FBS and 1% GlutaMAX. During imaging, cells were maintained at a constant temperature of 37 °C and CO_2 levels were maintained at 5%.

Microscope preparation. A Nikon Ti2 inverted microscope equipped with a tailored thermal chamber and CO_2 airflow (Life Imaging Services GmbH) was used. The MicroManager installation was instructed to load the corresponding device drivers The separately installed Mikro-Manager was configured to point to the correct configuration groups, mapping objectives (20× and 40×) as well as the main stages (x, y and z) and auto-focus stage.

On the day of the experiment, both 40× (Plan apochromatic (APO) 40×/0.95 NA, Nikon) and 20× (Plan APO 20×/0.5 NA, Nikon) objectives were installed. Automated pixel size calibrations were performed utilizing sparse cells on the mounted coverslips as guiding stars, through the integrated MicroManager plugin. In total, 16 positions (4 × 4) on the coverslip were selected with the help of the MicroManager Grid-Position manager to define the 2.5 × 2.5 mm^2 FOV to be probed at 20× magnification. When the sample was mounted, offsets of the two 20× and 40× objectives with the Perfect Focus System offset ('T-PFS Perfect Focus Unit', Nikon) were noted to allow precise z-repositioning when switching objectives.

Workflow preparation. The starred workflow was created on the Arkitekt web interface. It represents the logic of the acquisition of a single position and consequent cluster monitoring on this position. The workflow was deployed on the 'Reaktor' scheduling app, naming the new node 'Image Clusters on Position'.

A second workflow, now representing the wider multiposition acquisition was created, utilizing the just-created primary workflow as a node. the workflow was deployed, setting 'Stream Multi Position Clusters' as the node name.

Workflow run. Mikro-Manager was started, connected and authenticated with the Arkitekt platform. The Mikro-Manager Set Objective PSF-Offset node was reserved and run two times, setting each objective's previously noted offsets. The workflow node stream multi position clusters was reserved and started on the Arkitekt web interface, providing the following parameters as input: stage: newly created stage; acquire 3D objective: 40×; acquired 2D objective: 20×; epsilon: 100 pixels; minsize: 3; iterations: 24; iterationsleep: 20 min. The streamed output was monitored on the Arkitekt web interface and on the Stage Detail Pane, which automatically displayed new positions and images.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All data for the listed workflows is publicly available. The original data for workflow I is a direct copy of the *Tribolium* dataset (https://publications.mpi-cbg.de/publications-sites/7207/). A subset of this dataset is also made available on the documentation (https://arkitekt.live/docs/showcases) for demo purposes. An export of the analyzed data is available via Zenodo at https://doi.org/10.5281/zenodo.10031633 (ref. 33). Due to size restriction, only a limited export of the datasets for workflow II is available via Zenodo at https://doi.org/10.5281/zenodo.10031787 (ref. 34). Due to size restriction, only a limited export of the datasets for workflow III is available via Zenodo at https://doi.org/10.5281/zenodo.10031807 (ref. 35). The whole dataset is available on request.

Code availability

All software developed and used in this publication is freely available for access and use and have been deposited in dedicated GitHub repository mentioned alongside the methods, which can be accessed from https://github.com/arkitektio-apps. All software is licensed under the MIT License. In addition to the repositories, an online documentation (https://arkitekt.live) is provided to assist users in navigating and utilizing the dedicated code associated with this publication. This documentation offers insights into the structure, functions and applications of the software. Workflows are available as documented templates with system requirements and documentation on (https://arkitekt.live/docs/showcases/paper).

References

33. Roos, J. Workflow I – Interactive analysis – Three analysed ROIS. *Zenodo*https://doi.org/10.5281/zenodo.10031633 (2023).
34. Roos, J. Workflow II – Streaming analysis – Multi-position, multi timepoint acquisition. *Zenodo* https://doi.org/10.5281/zenodo.10031787 (2023).
35. Roos, J. Workflow III – Smart Microscopy – Adaptive monitoring of cell clusters. *Zenodo* https://doi.org/10.5281/zenodo.10031807 (2023).

Acknowledgements

We acknowledge F. Saltel and N. Allain for their kind gift of a stable fluorescent cell line. This work was supported by the Ministère de l'Enseignement Supérieur et de la Recherche (ANR-10-INBS-04 FranceBioImaging, ANR-22-CE42 DEEPHEPATOSCREEN, the University of Bordeaux's IdEx Investments for the Future program/GPR BRAIN_2030). It received support from MBI seed funding to V.V. and from the Calipso program (NRF2019-THE002-0007 to J.-B.S. and V.V.). This project received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie action (grant no. 794492), the Fonds AXA pour la Recherche to S.B., the Doctoral School for Health and Life Sciences of the University

of Bordeaux to J.R. and the European Research Council (ERC-SyG ENSEMBLE) (grant no. 951294) to U.V.N.

Author contributions

J.R. performed the architectural design and implementation of all relevant code of the platform backend and SDKs. J.-B.S. supervised this work. R.G. performed the microscopy experiments. T.D. contributed to the development of the acquisition platform of workflow 2. F.L. performed the development and implementation of segmentation plugins and guidance in bioimage focus. V.V., R.G. and J.-B.S. contributed to the design of the workflows and supervised the deployment and execution of the platform on different sites. M.E. contributed to the initial supervision of a precursor to the platform. S.B. helped in guidance and design input for metadata models. A.W. performed testing and additional plugin apps development. U.V.N. provided scientific environment and student supervision. All the authors contributed to the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-024-02404-5.

Correspondence and requests for materials should be addressed to Jean-Baptiste Sibarita.

Peer review information *Nature Methods* thanks Peter Bajcsy, Jean-Karim Hériché and Nico Stuurman for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

nature portfolio

Corresponding author(s): Jean-Baptiste Sibarita

Last updated by author(s): Apr 28, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☒	☐	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Workflow I: This workflow used the publicly available dataset: Tribolium (https://publications.mpi-cbg.de/publications-sites/7207/) , which was used both for training as well as in the analysis pipeline. Workflow II: Data was acquired using the MetaMorph (version 7.10.5.476) commercial software (Molecular Devices). Workflow III: Data was acquired through the MikroManager App provided with the platform (https://github.com/jhnnsrs/mikro-manager) which was orchestrated by the described workflow. Micro-Manager 2.0.1 20230523 (MMcore version 10.4.0, Device API version 71, Module API version 10). All the source code related to Arkitekt can be found at https://github.com/arkitektio-apps Additional technical information can be found at https://arkitekt.live Workflows can be found at https://arkitekt.live/docs/showcases/paper
Data analysis	Data analysis and used software are described in detail in the corresponding method section.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

All data for the listed workflows is publicly available. The original data for Workflow I is a direct copy of the Tribolium dataset (https://publications.mpi-cbg.de/publications-sites/7207/). A subset of this dataset is also made available on the documentation (https://arkitekt.live/docs/showcases) for demo purposes. An export of the analyzed data is available as a Zenodo archive (https://doi.org/10.5281/zenodo.10031633). Due to size restriction only a limited export of the datasets for Workflow II (https://doi.org/10.5281/zenodo.10031787) and Workflow III (https://doi.org/10.5281/zenodo.10031807) are available as Zenodo archives. The whole dataset is available on request.

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	As the biological question was secondary to the introduced methodology, not sample size calculations were performed. In Workflow 2 (live quantitative monitoring), we automatically collected and analysed 20 different organoids in 3D, in time-lapse mode (30 times points every 20 minutes), representing a total of 600 3D stacks In Workflow 3 (smart microscopy), we acquired 16 positions in time-lapse (24 time-points) representing 384 images, from which 21,186 nuclei were segmented and 2,027 clusters were detected and acquired in 3D.
Data exclusions	None
Replication	All depicted Workflows have been executed several times in testing scenarios spanning multiple configurations of platform (Linux and Windows) and underlying hardware.
Randomization	No randomization was performed as it was not relevant for this work, which consisted in demonstrating software acquisition and analysis pipelines. Not relevant quantitifications were performed.
Blinding	No blinding analysis was performed as it was not relevant to this work, which consisted in demonstrating software acquisition and analysis pipelines. Not relevant quantitifications were performed

Behavioural & social sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	*Briefly describe the study type including whether data are quantitative, qualitative, or mixed-methods (e.g. qualitative cross-sectional, quantitative experimental, mixed-methods case study).*
Research sample	*State the research sample (e.g. Harvard university undergraduates, villagers in rural India) and provide relevant demographic information (e.g. age, sex) and indicate whether the sample is representative. Provide a rationale for the study sample chosen. For studies involving existing datasets, please describe the dataset and source.*
Sampling strategy	*Describe the sampling procedure (e.g. random, snowball, stratified, convenience). Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient. For qualitative data, please indicate whether data saturation was considered, and what criteria were used to decide that no further sampling was needed.*
Data collection	*Provide details about the data collection procedure, including the instruments or devices used to record the data (e.g. pen and paper, computer, eye tracker, video or audio equipment) whether anyone was present besides the participant(s) and the researcher, and whether the researcher was blind to experimental condition and/or the study hypothesis during data collection.*
Timing	*Indicate the start and stop dates of data collection. If there is a gap between collection periods, state the dates for each sample cohort.*
Data exclusions	*If no data were excluded from the analyses, state so OR if data were excluded, provide the exact number of exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.*
Non-participation	*State how many participants dropped out/declined participation and the reason(s) given OR provide response rate OR state that no participants dropped out/declined participation.*
Randomization	*If participants were not allocated into experimental groups, state so OR describe how participants were allocated to groups, and if allocation was not random, describe how covariates were controlled.*

Ecological, evolutionary & environmental sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	*Briefly describe the study. For quantitative data include treatment factors and interactions, design structure (e.g. factorial, nested, hierarchical), nature and number of experimental units and replicates.*
Research sample	*Describe the research sample (e.g. a group of tagged Passer domesticus, all Stenocereus thurberi within Organ Pipe Cactus National Monument), and provide a rationale for the sample choice. When relevant, describe the organism taxa, source, sex, age range and any manipulations. State what population the sample is meant to represent when applicable. For studies involving existing datasets, describe the data and its source.*
Sampling strategy	*Note the sampling procedure. Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient.*
Data collection	*Describe the data collection procedure, including who recorded the data and how.*
Timing and spatial scale	*Indicate the start and stop dates of data collection, noting the frequency and periodicity of sampling and providing a rationale for these choices. If there is a gap between collection periods, state the dates for each sample cohort. Specify the spatial scale from which the data are taken*
Data exclusions	*If no data were excluded from the analyses, state so OR if data were excluded, describe the exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.*
Reproducibility	*Describe the measures taken to verify the reproducibility of experimental findings. For each experiment, note whether any attempts to repeat the experiment failed OR state that all attempts to repeat the experiment were successful.*
Randomization	*Describe how samples/organisms/participants were allocated into groups. If allocation was not random, describe how covariates were controlled. If this is not relevant to your study, explain why.*
Blinding	*Describe the extent of blinding used during data acquisition and analysis. If blinding was not possible, describe why OR explain why blinding was not relevant to your study.*

Did the study involve field work? ☐ Yes ☒ No

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☐	☒ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Eukaryotic cell lines

Policy information about cell lines and Sex and Gender in Research

Cell line source(s)	The hepatocellular carcinoma HepG2 source cell line are commercial cell lines from ATCC with the reference: [HEPG2] (ATCC® HB-8065™) The cell line HepG2 stably expressing the proteins H2B-GFP were given by the Group of Frederic Saltel. They were produced by lentivirus transduction using the LV H2B-GFP lentivirus produced by the group of Frederic Saltel.
Authentication	The hepatocellular carcinoma HepG2 source cell line used have been authentificated by the ATCC with the number HB-8065 by DNA barcoding and morphology."
Mycoplasma contamination	The cell line HepG2 stably expressing the proteins H2B-GFP were tested negative for mycoplasma.
Commonly misidentified lines (See ICLAC register)	No commonly misidentified cell lines were used in the study.

Palaeontology and Archaeology

Specimen provenance	*Provide provenance information for specimens and describe permits that were obtained for the work (including the name of the issuing authority, the date of issue, and any identifying information). Permits should encompass collection and, where applicable, export.*
Specimen deposition	*Indicate where the specimens have been deposited to permit free access by other researchers.*
Dating methods	*If new dates are provided, describe how they were obtained (e.g. collection, storage, sample pretreatment and measurement), where they were obtained (i.e. lab name), the calibration program and the protocol for quality assurance OR state that no new dates are provided.*

☐ Tick this box to confirm that the raw and calibrated dates are available in the paper or in Supplementary Information.

Ethics oversight	*Identify the organization(s) that approved or provided guidance on the study protocol, OR state that no ethical approval or guidance was required and explain why not.*

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Animals and other research organisms

Policy information about studies involving animals; ARRIVE guidelines recommended for reporting animal research, and Sex and Gender in Research

Laboratory animals	*For laboratory animals, report species, strain and age OR state that the study did not involve laboratory animals.*
Wild animals	*Provide details on animals observed in or captured in the field; report species and age where possible. Describe how animals were caught and transported and what happened to captive animals after the study (if killed, explain why and describe method; if released, say where and when) OR state that the study did not involve wild animals.*
Reporting on sex	*Indicate if findings apply to only one sex; describe whether sex was considered in study design, methods used for assigning sex. Provide data disaggregated for sex where this information has been collected in the source data as appropriate; provide overall*

numbers in this Reporting Summary. Please state if this information has not been collected. Report sex-based analyses where performed, justify reasons for lack of sex-based analysis.

Field-collected samples	For laboratory work with field-collected samples, describe all relevant parameters such as housing, maintenance, temperature, photoperiod and end-of-experiment protocol OR state that the study did not involve samples collected from the field.
Ethics oversight	Identify the organization(s) that approved or provided guidance on the study protocol, OR state that no ethical approval or guidance was required and explain why not.

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Clinical data

Policy information about clinical studies

All manuscripts should comply with the ICMJE guidelines for publication of clinical research and a completed CONSORT checklist must be included with all submissions.

Clinical trial registration	Provide the trial registration number from ClinicalTrials.gov or an equivalent agency.
Study protocol	Note where the full trial protocol can be accessed OR if not available, explain why.
Data collection	Describe the settings and locales of data collection, noting the time periods of recruitment and data collection.
Outcomes	Describe how you pre-defined primary and secondary outcome measures and how you assessed these measures.

Dual use research of concern

Policy information about dual use research of concern

Hazards

Could the accidental, deliberate or reckless misuse of agents or technologies generated in the work, or the application of information presented in the manuscript, pose a threat to:

No	Yes	
☒	☐	Public health
☒	☐	National security
☒	☐	Crops and/or livestock
☒	☐	Ecosystems
☒	☐	Any other significant area

Experiments of concern

Does the work involve any of these experiments of concern:

No	Yes	
☒	☐	Demonstrate how to render a vaccine ineffective
☒	☐	Confer resistance to therapeutically useful antibiotics or antiviral agents
☒	☐	Enhance the virulence of a pathogen or render a nonpathogen virulent
☒	☐	Increase transmissibility of a pathogen
☒	☐	Alter the host range of a pathogen
☒	☐	Enable evasion of diagnostic/detection modalities
☒	☐	Enable the weaponization of a biological agent or toxin
☒	☐	Any other potentially harmful combination of experiments and agents

ChIP-seq

Data deposition

☐ Confirm that both raw and final processed data have been deposited in a public database such as GEO.

☐ Confirm that you have deposited or provided access to graph files (e.g. BED files) for the called peaks.

Data access links *May remain private before publication.*	For "Initial submission" or "Revised version" documents, provide reviewer access links. For your "Final submission" document, provide a link to the deposited data.
Files in database submission	Provide a list of all files available in the database submission.

Genome browser session (e.g. UCSC)	*Provide a link to an anonymized genome browser session for "Initial submission" and "Revised version" documents only, to enable peer review. Write "no longer applicable" for "Final submission" documents.*

Methodology

Replicates	*Describe the experimental replicates, specifying number, type and replicate agreement.*
Sequencing depth	*Describe the sequencing depth for each experiment, providing the total number of reads, uniquely mapped reads, length of reads and whether they were paired- or single-end.*
Antibodies	*Describe the antibodies used for the ChIP-seq experiments; as applicable, provide supplier name, catalog number, clone name, and lot number.*
Peak calling parameters	*Specify the command line program and parameters used for read mapping and peak calling, including the ChIP, control and index files used.*
Data quality	*Describe the methods used to ensure data quality in full detail, including how many peaks are at FDR 5% and above 5-fold enrichment.*
Software	*Describe the software used to collect and analyze the ChIP-seq data. For custom code that has been deposited into a community repository, provide accession details.*

Flow Cytometry

Plots

Confirm that:

☐ The axis labels state the marker and fluorochrome used (e.g. CD4-FITC).

☐ The axis scales are clearly visible. Include numbers along axes only for bottom left plot of group (a 'group' is an analysis of identical markers).

☐ All plots are contour plots with outliers or pseudocolor plots.

☐ A numerical value for number of cells or percentage (with statistics) is provided.

Methodology

Sample preparation	*Describe the sample preparation, detailing the biological source of the cells and any tissue processing steps used.*
Instrument	*Identify the instrument used for data collection, specifying make and model number.*
Software	*Describe the software used to collect and analyze the flow cytometry data. For custom code that has been deposited into a community repository, provide accession details.*
Cell population abundance	*Describe the abundance of the relevant cell populations within post-sort fractions, providing details on the purity of the samples and how it was determined.*
Gating strategy	*Describe the gating strategy used for all relevant experiments, specifying the preliminary FSC/SSC gates of the starting cell population, indicating where boundaries between "positive" and "negative" staining cell populations are defined.*

☐ Tick this box to confirm that a figure exemplifying the gating strategy is provided in the Supplementary Information.

Magnetic resonance imaging

Experimental design

Design type	*Indicate task or resting state; event-related or block design.*
Design specifications	*Specify the number of blocks, trials or experimental units per session and/or subject, and specify the length of each trial or block (if trials are blocked) and interval between trials.*
Behavioral performance measures	*State number and/or type of variables recorded (e.g. correct button press, response time) and what statistics were used to establish that the subjects were performing the task as expected (e.g. mean, range, and/or standard deviation across subjects).*

Acquisition

Imaging type(s) *Specify: functional, structural, diffusion, perfusion.*

Field strength *Specify in Tesla*

Sequence & imaging parameters *Specify the pulse sequence type (gradient echo, spin echo, etc.), imaging type (EPI, spiral, etc.), field of view, matrix size, slice thickness, orientation and TE/TR/flip angle.*

Area of acquisition *State whether a whole brain scan was used OR define the area of acquisition, describing how the region was determined.*

Diffusion MRI ☐ Used ☐ Not used

Preprocessing

Preprocessing software *Provide detail on software version and revision number and on specific parameters (model/functions, brain extraction, segmentation, smoothing kernel size, etc.).*

Normalization *If data were normalized/standardized, describe the approach(es): specify linear or non-linear and define image types used for transformation OR indicate that data were not normalized and explain rationale for lack of normalization.*

Normalization template *Describe the template used for normalization/transformation, specifying subject space or group standardized space (e.g. original Talairach, MNI305, ICBM152) OR indicate that the data were not normalized.*

Noise and artifact removal *Describe your procedure(s) for artifact and structured noise removal, specifying motion parameters, tissue signals and physiological signals (heart rate, respiration).*

Volume censoring *Define your software and/or method and criteria for volume censoring, and state the extent of such censoring.*

Statistical modeling & inference

Model type and settings *Specify type (mass univariate, multivariate, RSA, predictive, etc.) and describe essential details of the model at the first and second levels (e.g. fixed, random or mixed effects; drift or auto-correlation).*

Effect(s) tested *Define precise effect in terms of the task or stimulus conditions instead of psychological concepts and indicate whether ANOVA or factorial designs were used.*

Specify type of analysis: ☐ Whole brain ☐ ROI-based ☐ Both

Statistic type for inference (See Eklund et al. 2016) *Specify voxel-wise or cluster-wise and report all relevant parameters for cluster-wise methods.*

Correction *Describe the type of correction and how it is obtained for multiple comparisons (e.g. FWE, FDR, permutation or Monte Carlo).*

Models & analysis

n/a	Involved in the study
☐	☐ Functional and/or effective connectivity
☐	☐ Graph analysis
☐	☐ Multivariate modeling or predictive analysis

Functional and/or effective connectivity *Report the measures of dependence used and the model details (e.g. Pearson correlation, partial correlation, mutual information).*

Graph analysis *Report the dependent variable and connectivity measure, specifying weighted graph or binarized graph, subject- or group-level, and the global and/or node summaries used (e.g. clustering coefficient, efficiency, etc.).*

Multivariate modeling and predictive analysis *Specify independent variables, features extraction and dimension reduction, model, training and evaluation metrics.*