TissUUmaps 3: Interactive visualization and quality assessment of large-scale spatial omics data

Nicolas Pielawski*, Axel Andersson*, Christophe Avenel, Andrea Behanova, Eduard Chelebian, Anna Klemm, Fredrik Nysjö, Leslie Solorzano and Carolina Wählby*

Department of Information Technology and SciLifeLab BioImage Informatics Facility, Uppsala University, Uppsala, 752 37 Sweden.

January 28, 2022

Abstract: TissUUmaps is a browser-based tool for GPU-accelerated visualization and interactive exploration of 10^7+ datapoints overlaying tissue samples. Users can visualize markers and regions, explore spatial statistics and quantitative analyses of tissue morphology, and assess the quality of decoding in situ transcriptomics data. TissUUmaps provides instant multi-resolution image viewing, can be customized, shared, and also integrated in Jupyter Notebooks. We envision TissUUmaps to contribute to broader dissemination and flexible sharing of large-scale spatial omics data.

Multimodal omics data is typically subject to two different types of investigation: unspecific high level investigation, where we visually interpret the information, see patterns, construct hypotheses and draw conclusions; and low level hypothesis testing, where we build a representation of the unknown by asking, and seeking answers to, questions that can provide statistical certainty measures to test our hypotheses. In both cases, visual exploration and inspection of data quality are crucial for constructing and validating our hypothesis. However, the complexity and scale of data often makes visual exploration fundamentally challenging. For example, a typical in situ transcriptomics (IST, [1–6]) experiment results in millions of spatial markers for hundreds of different RNA species, where the micrometer-scale defines cell types while the centimeter-scale is needed for identifying tissue-level structures.

Furthermore, sharing of data with the scientific community is difficult due to slow and costly data transfer. Here, web-based applications that are hosted on a web-server make interactive visualization a matter of sharing a simple web-link that displays the contents instantly.

There are several powerful viewers available for interactive visualization of spatial omics data. One of the most popular viewers is Napari [7], which is an excellent tool for general visualization and annotation tasks, including the displaying of 3D content. It is however limited by the processing power and memory of the user's computer, and does not provide functionalities for interactive web-based visualization. The Giotto [8] viewer provides a flexible interface that allows the user to simultaneously explore gene expression markers in physical space and a low-dimensional expression space. It is web-based, allowing interactive visualizations to be shared over the Internet, but it is strictly limited by the size of the image data, and cannot visualize millions of markers, due to the lack of GPU-based rendering. Other viewers are available, such as STViewer [9] and Cytomine [10], but they are typically tailored for specific

tasks and not suitable for combining very large images and high-content marker data.

We present TissUUmaps 3, developed with the aim of making visual exploration and sharing of image data and markers easy and accessible to anyone. It overcomes hardware memory limitations by using a pyramidal image scheme only displaying the parts of a large multi-layered image that are needed at the instance of interaction (i.e. for a specific canvas location, size and zoom level). Furthermore, tens of millions of markers and regions can overlay the image data in real-time thanks to TissUUmaps' GPU-accelerated visualization, see Fig. 1a. Tedious data transfer is avoided by hosting TissUUmaps on a server and providing interactive visualizations by simply sharing a web-link. This makes TissUUmaps suitable as a tool for disseminating research data, as exemplified in the TissUUmaps gallery: `https://tissuumaps.github.io/gallery`. However, not every user possesses resources and expertise for setting up a server, and in order to make TissUUmaps easily accessible, it can also be installed and run as a regular desktop application (Fig. 1b).

Large-scale spatial omics experiments typically aim to identify cell types and explore the interaction of cells in a tissue sample. Popular tools for such analysis and spatial statistics are SquidPy [11] and Giotto [8], and thanks to a simple and flexible data structure, both markers and regions can be exported from these tools to TissUUmaps. We demonstrate this through a short video tutorial (Giotto/Squidpy video) with additional details in SI 1.

Furthermore, outputs from probabilistic cell typing methods such as pciSeq [12] can be presented as pie-charts within TissUUmaps, concisely displaying local cell-type probabilities (Fig. 1c). TissUUmaps is not limited to spatial omics data: any spatially resolved data that can be presented as regions or data points with coordinates in space may be displayed. For example, diverse data such as per-cell features extracted by CellProfiler [13], and projected UMAP [14] output can be interactively viewed, and points in a region can be manually selected, or 'gated', via TissUUmaps. The selected points can thereafter be overlaid on top of the corresponding tissue sample, providing a link between feature space and tissue space.

Apart from marker data, the image data in itself contains a wealth of information that may be related to spatial omics data. Here, we have optimized TissUUmaps for exploring such relations by focusing on Python compatibility via Jupyter Notebooks. Jupyter Notebooks structure code in cells, and TissUUmaps can be run within such a cell, as depicted in Fig. 1b, making it easy to integrate TissUUmaps into image and omics analysis pipelines. We provide video examples showing, e.g., how a pre-trained deep learning framework can be used to extract descriptors of tissue morphology and display them using TissUUmaps (conv. neural network video), and how graph neural networks [15] can be used to embed spatial gene expression patterns (Spage2Vec video). Additional details for implementing the examples shown in the videos are found in SI 2 and SI 3.

Moreover, manually drawn or automatically generated region- and cell outlines created in QuPath [16] and StarDist [17] can be directly loaded into TissUUmaps thanks to compatibility via the GeoJSON file format (see SI 4 for details).

Spatial omics techniques such as [1–6] use multiplex labeling strategies and rely on computationally expensive image analysis pipelines for extracting information such as the location of millions of different RNA species. Quality control in the form of visual inspection is crucial for validating such pipelines, but also difficult, as one must search centimeter sized spaces for micrometer sized fluorescent spots that are either present or absent in different imaging rounds and channels. Here, TissUUmaps' multi-resolution and multi-layer rendering capabilities become invaluable, as they allow the user to interactively explore images across all resolutions, while simultaneously displaying content from multiple rounds and channels using different colormaps. By extending TissUUmaps with the `Spot Inspector` plugin, the user can also click anywhere in TissUUmaps to open a gridded figure displaying content from all rounds and channels as well as traces indicating detected multiplexed labels (exemplified for in situ sequencing data in Fig. 1c). We demonstrate this through a short video (spot inspector video) with details in SI 5.

Cooperating and sharing data with others at each stage of research — from prototyping to dissemi-

nation — is important. TissUUmaps can export projects as a folder that can be easily shared with others. It is also possible to export a static web page that the user can upload to any web server and share the URL linking to the page. Anyone with the link can visualize the image data and overlay existing markers and regions as well as their own post-processed data, as shown in Fig. 1d. To conclude, TissUUmaps 3 is a visualization tool compatible with a wide range of large-scale raw and processed multi-omics data. Its unique ability to broadly and interactively display data via a web-browser without requirements on installing software opens up for unique quality control capabilities and FAIR [18] data sharing in a completely new way.

Online content

TissUUmaps is available at `https://tissuumaps.github.io` together with video tutorials, downloading instructions and project demos. The source code for TissUUmaps can be found at `https://github.com/TissUUmaps/TissUUmaps`, shared under a permissive free software license: BSD 3.

References

1. Ji, N. & Van Oudenaarden, A. Single molecule fluorescent in situ hybridization (smFISH) of C. elegans worms and embryos (2012).
2. Ke, R. *et al.* In situ sequencing for RNA analysis in preserved tissue and cells. *Nature methods* **10,** 857–860 (2013).
3. Lee, J. H. *et al.* Fluorescent in situ sequencing (FISSEQ) of RNA for gene expression profiling in intact cells and tissues. *Nature protocols* **10,** 442–458 (2015).
4. Wang, G., Moffitt, J. R. & Zhuang, X. Multiplexed imaging of high-density libraries of RNAs with MERFISH and expansion microscopy. *Scientific reports* **8,** 1–13 (2018).
5. Wang, X. *et al.* Three-dimensional intact-tissue sequencing of single-cell transcriptional states. *Science* **361** (2018).
6. Eng, C.-H. L. *et al.* Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature* **568,** 235–239 (2019).
7. Sofroniew, N. *et al. napari/napari: 0.4.12rc2* version v0.4.12rc2. Oct. 2021. `https://doi.org/10.5281/zenodo.5587893`.
8. Dries, R. *et al.* Giotto: a toolbox for integrative analysis and visualization of spatial expression data. *Genome biology* **22,** 1–31 (2021).
9. Navarro, J. F., Lundeberg, J. & Ståhl, P. L. ST viewer: a tool for analysis and visualization of spatial transcriptomics datasets. *Bioinformatics* **35** (ed Berger, B.) 1058–1060. `https://doi.org/10.1093/bioinformatics/bty714` (Aug. 2018).
10. Marée, R. *et al.* Collaborative analysis of multi-gigapixel imaging data using Cytomine. *Bioinformatics* **32,** 1395–1401 (2016).
11. Palla, G. *et al.* Squidpy: a scalable framework for spatial single cell analysis. *bioRxiv* (2021).
12. Qian, X. *et al.* Probabilistic cell typing enables fine mapping of closely related cell types in situ. *Nature methods* **17,** 101–106 (2020).
13. Carpenter, A. E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology* **7,** 1–11 (2006).
14. Becht, E. *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* **37,** 38–44 (2019).
15. Partel, G. & Wählby, C. Spage2vec: Unsupervised representation of localized spatial gene expression signatures. *The FEBS Journal* **288,** 1859–1870. `https://doi.org/10.1111/febs.15572` (Oct. 2020).
16. Bankhead, P. *et al.* QuPath: Open source software for digital pathology image analysis. *Scientific reports* **7,** 1–7 (2017).

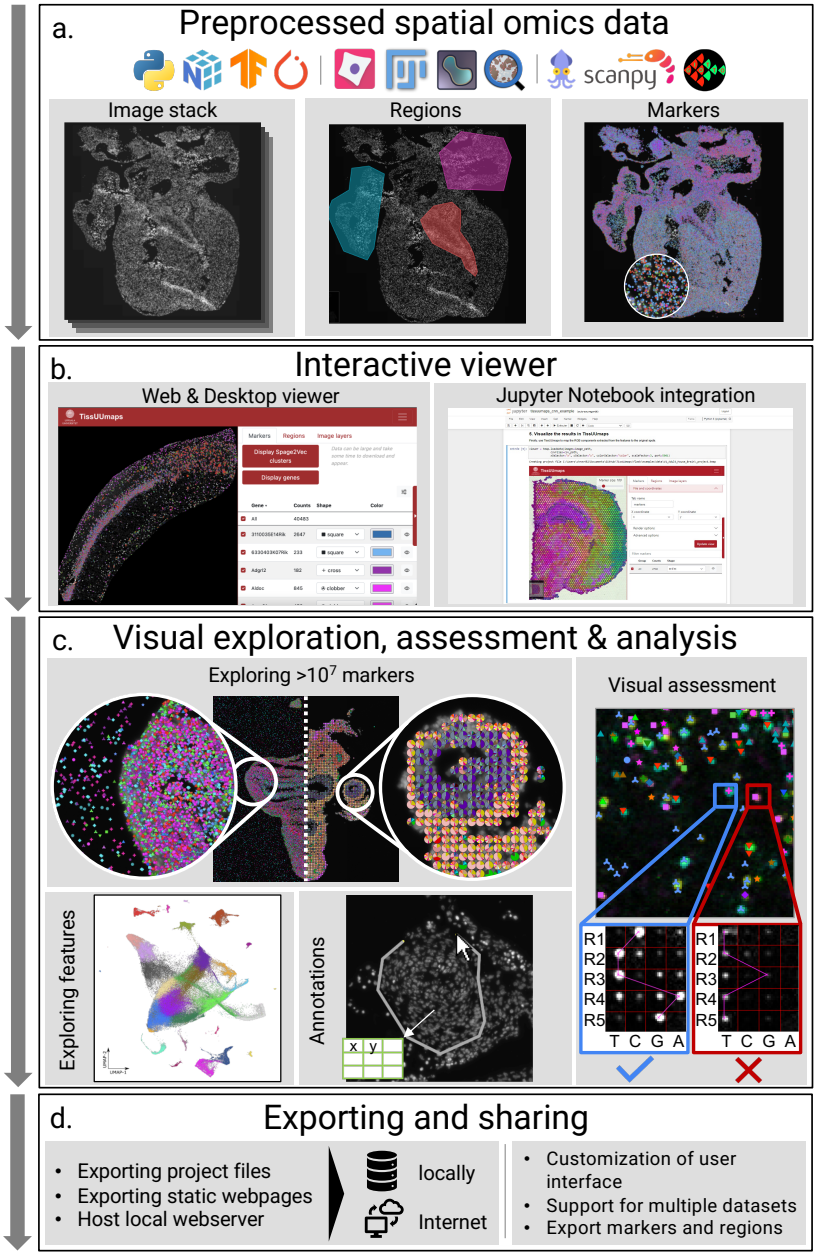

Figure 1: TissUUmaps is a viewer for large scale spatial omics data. **a** Data, in the form of regions and markers, extracted by image analysis and/or created by a range of different spatial statistics tools can be visualized on top of gigapixel sized image stacks. **b** Accelerated GPU-based rendering makes image and data interaction fast, via a web or desktop viewer, or as part of a Jupyter Notebook. **c** Data can be presented and explored as markers in space or summarized in localized pie-charts. Plugins for quality control enable visual assessment of raw data associated with decoded IST signals. Non-spatial data, such as UMAP projections, can also be interactively explored and selected or 'gated' signals can be overlaid on top of the original tissue sample. **d** The data owner can export custom project files or static web pages to be run locally or over the Internet, and external users can interactively explore, manually select, and download data from regions of interest.

17. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. *Cell detection with star-convex polygons* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018), 265–273.

18. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **3,** 1–9 (2016).

Methods

Software implementation. Loading large images can be prohibitively expensive due to hardware limitations such as the RAM. Most existing software load the entire data in memory unlike TissUUmaps which only displays the sub-parts of the images that are needed. Using a pyramidal scheme, looking at the full image displays a low-resolution version of the data and when zooming will display higher resolution patches of the data in the area of interest. Loading dynamically parts of the image and releasing the parts that are not displayed anymore removes entirely the limitation of the hardware. However, the images need to be converted to a pyramidal format before displaying them, which requires additional hard drive space. Images not in a pyramidal format are automatically converted using VIPS [1] when loaded into TissUUmaps.

In its core, TissUUmaps is implemented using regular web-development tools such as HTML, JavaScript and SCSS. It is completely open-source (available at `https://github.com/TissUUmaps`). As explained in the previous version of TissUUmaps [2], the software only depends on a small number of external libraries, making it easy to set-up and maintain. Since its previous version [2], TissUUmaps has been subjected to a range of upgrades, such as: Automatic conversion to pyramidal format; On-the-fly of millions of markers using WebGL; Interactive visualization and adjustments of multiple image layers; Desktop installation and usage; Jupyter Notebook compatibility; plugins for inspection In Situ Transcriptomics (IST) data and import/export to Napari.

Jupyter Notebook Integration. TissUUmaps can conveniently be imported into a Jupyter Notebook using the `tissuumaps.jupyter` module to interactively open a viewer. This allows the user to integrate TissUUmaps in their analysis pipelines. When a user has performed their analysis, they can call the `loaddata(images=, csvFiles=)` function to create a TissUUmaps project with the desired configuration. Additionally, TissUUmaps supports visualization on coordinate systems other than the image, allowing to visualize dimensionally reduced spaces such as UMAP [3]. These low dimensional features can then be mapped back to the spatial coordinates of the tissue to perform gating as a way of identifying clusters for quality assessment. We showcase TissUUmaps Jupyter Notebook integration through two short video examples. In the first example (Spage2Vec video) we make use of a graph neural network [4] to explore the spatial heterogeneity of gene expression data and visualize in TissUUmaps. In the second video (conv. neural network video), we make use of a deep learning frameworks to extract morphological features, and visualize the results using TissUUmaps. Implementation details for respective video are found in SI 3 and SI 2.

Visualizing spatial omics analysis outputs. Modern spatial omics analysis software such as Squidpy [5] and Giotto [6] allow in-depth analyses of omics data. Output from such software can easily be visualized in TissUUmaps by first converting the software's native data structures into regular CSV-formats. The CSV file can then be loaded into TissUUmaps along with image data. This allows the user to overlay and explore spatial analyses results on-top of full resolution gigapixel image data. We provide a video tutorial on how to visualize data analyzed using SquidPy [5] and Giotto [6] (Giotto/Squidpy video). Additional implementation details for the video are found in SI 1.

Spot Inspector Plugin. The Spot Inspector plugin is implemented using Python with a Javascript interface for interacting with the main module. The plugin allows the user to interactively visualize content from multiple staining rounds and channels by clicking on the currently displayed image in TissUUmaps. The plugin works for TissUUmaps Windows version and is demonstrated through a short video (spot inspector video). We refer to SI 5 for details on installation and usage.

Interfacing with Common Image Processing Tools. There are many powerfull tools for scientific image analysis of medical image data. Among these

are QuPath [7], Napari [8], CellProfiler [9, 10] and Fiji [11]. Results produced by such softwares can be uploaded to a server and shared as interactive visualizations using TissUUmaps. However, results produced by different tools and plugins are often of different file formats. We therefore provide few examples on how to interface such softwares.

Visualizing Napari Projects. A Napari-TissUUmaps plugin is available and allows the user to export a Napari project to be loaded in TissUUmaps. Images, annotations or labeling as masks and vector graphics, as well as point clouds are supported. The plugin is available on the Napari Hub and can thus be installed very quickly in the Napari plugin manager, alternatively using the Python package manager pip. Harnessing the power of available plugins and Python allows users to work on Napari and export their project to TissUUmaps and conveniently share their result and analysis online. A short video tutorial (Napari video), with additional details in SI 6, is provided as an example on how to export Napari projects and visualize in TissUUmaps.

Visualizing CellProfiler Outputs. CellProfiler [9, 10] is a popular tool for performing high-throughput analyses of medical image data. CellProfiler usually outputs results in the form of CSV-file(s). These results can be visualized directly in TissUUmaps by loading the CSV file into the software. However, tissue samples are often divided into a set of smaller tiles before being processed sequentially in a CellProfiler pipeline. In such situations, artefact may appear near the borders of individual tiles, resulting in undesired patterns called "tile-effects". Tile-effects become especially apparent when results from individual tiles are put together in a global context. Tile-effects should ideally be mitigated before visualizing in TissUUmaps. This can be done by using slightly overlapping tiles together with merging heuristics as explained in SI 7.

Visualizing QuPath outputs. QuPath [7] is an open-source tool for digital pathology. It offers a wide range of utilities for quantification of tissue morphology SI 4. For instance, QuPath supports classification, annotation, and segmentation of different cells and tissue regions. Such results can be visualized as regions in TissUUmaps by exporting the QuPath results to a GeoJSON file which in turn can be read into TissUUmaps.

Visualizing Fiji outputs. Another popular tool for image analysis is Fiji. Fiji bundles a lot of plugins that facilitate scientific image processing. We provide an example script that showcase registration of fluorescent image data using Fiji plugins, and exports the data into formats that can be read by TissUUmaps, see SI 8.

References

1. Martinez, K. & Cupitt, J. *VIPS-a highly tuned image processing software architecture* in *IEEE International Conference on Image Processing 2005* **2** (2005), II–574.
2. Solorzano, L., Partel, G. & Wählby, C. TissUUmaps: interactive visualization of large-scale spatial gene expression and tissue morphology data. *Bioinformatics* **36,** 4363–4365. ISSN: 1367-4803. eprint: `https://academic.oup.com/bioinformatics/article-pdf/36/15/4363/33796634/btaa541.pdf`. `https://doi.org/10.1093/bioinformatics/btaa541` (May 2020).
3. Becht, E. *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* **37,** 38–44 (2019).
4. Partel, G. & Wählby, C. Spage2vec: Unsupervised representation of localized spatial gene expression signatures. *The FEBS Journal* **288,** 1859–1870. `https://doi.org/10.1111/febs.15572` (Oct. 2020).
5. Palla, G. *et al.* Squidpy: a scalable framework for spatial single cell analysis. *bioRxiv* (2021).
6. Dries, R. *et al.* Giotto: a toolbox for integrative analysis and visualization of spatial expression data. *Genome biology* **22,** 1–31 (2021).

7. Bankhead, P. *et al.* QuPath: Open source software for digital pathology image analysis. *Scientific reports* **7,** 1–7 (2017).

8. Sofroniew, N. *et al. napari/napari: 0.4.12rc2* version v0.4.12rc2. Oct. 2021. `https://doi.org/10.5281/zenodo.5587893`.

9. Carpenter, A. E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology* **7,** 1–11 (2006).

10. McQuin, C. *et al.* CellProfiler 3.0: Next-generation image processing for biology. *PLoS biology* **16,** e2005970 (2018).

11. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nature methods* **9,** 676–682 (2012).

Supplementary Information

SI 1 Visualizing Results from Giotto and Squidpy

Popular software for IST analysis, including Squidpy [1] and Giotto [2], for clustering or cell type identification often rely on complex data structures to perform their integrated analyses. This can render the interactive visualization of the outputs of such software difficult or limit it to the chosen development environment.

TissUUmaps can be used for visualization of data structures in Squidpy (built in Python) and Giotto objects in Giotto (built in R) by extracting the necessary information and converting it to CSV files. By using Python wrapper scripts, the user can easily do so and proceed with the usual TissUUmaps visualization workflow.

The user can then use different TissUUmaps tabs to visualize (1) the analysis outputs together overlaying on the multiresolution images and (2) the corresponding dimensionally reduced space. As both visualizations share the same markers, specific subgroups -such as clusters- can be selected in the feature space and visualized on the original image.

This is showcased for both software in the following tutorial: `https://tissuumaps.github.io/tutorials/#giotto_squidpy`. The example wrapper scripts for Squidpy and Giotto can be found in `https://github.com/TissUUmaps/TissUUmaps/blob/master/examples/squidpy2tmap.ipynb` and `https://github.com/TissUUmaps/TissUUmaps/blob/master/examples/giotto2tmap.py`, respectively.

SI 2 Visualization of Deep Learning Features

Thanks to the Jupyter Notebook compatibility, TissUUmaps can be used to interactively visualize morphological features extracted from convolutional neural networks implemented in TensorFlow [3] as in [4]. For the complete implementation, please visit `https://github.com/TissUUmaps/TissUUmaps/blob/master/examples/tissuumaps_cnn_example.ipynb`.

In this example, we extracted morphological features from a Visium 10X spatial transcriptomics dataset of a mouse brain using an InceptionV3 model pretrained on ImageNet. The implementation is provided such that it can be applied to any other dataset using any other model.

1. First, the necessary libraries are imported. Apart from the common Python data handling libraries, scanpy [5], openslide [6], tensorflow [3], umap [7] and tissuumaps must be installed.

2. In order to make the example as reproducible as possible, the Visium 10X dataset is loaded using scanpy. This will automatically download a folder containing the .tiff image and the .csv barcodes files for the sample.

3. For reproducibility purposes, tensorflow is used to load the pretrained network that will be used for feature extraction. Note that the model is not used for classification and thus the final layer is discarded. Instead, a pooling layer is added in the end to aggregate the last convolutional output into a (2048-sized, in this case) vector.

4. The gigapixel whole slide image needs to be divided into smaller patches to extract local morphological features. It is convenient to use the spatial transcriptomics spots as the centre of such patches in order

to potentially map the morphological features around that spot to the existing gene expression. It is of great importance, then, to choose a patch size large enough to capture spatial variations but not so much that the blurring effect is excessive.

The patches are automatically extracted and inputted to the feature extraction network which yields 2048 features per spot, in this case. The features can be separately visualized but it is common practice to reduce the dimensionality using methods such as umap.

5. Once the dimensionality is reduced from 2048 per spot to three UMAP components, these can be interpreted as RGB color dimensions and shown in TissUUmaps as a morphological gradient overlaid on top of the whole slide image.

   Furthermore, using the same approach, we used the TissUUmaps viewer to visualize the UMAP space. As shown in the example, the user can interactively change the marker types, sizes and colors grouping by different criteria.

   As we are using the same data, the embedded spots can be mapped back to the original image coordinates to perform gating as a way of identifying clusters for quality assessment.

The reader should note that, the chosen model was pretrained on ImageNet, a natural image dataset. We recommend using models (pre-)trained on biological data when possible. Additionally, default parameters were used for all the demonstration. The results may improve when using different parameters (e.g. patch sizes or dimensionality reduction techniques).

A tutorial on how to use the TissUUmaps viewer in a Jupyter Notebook to visualize morphological features extracted by a convolutional neural network can be found in `https://tissuumaps.github.io/tutorials/#cnn`.

SI 3 Visualizing Spage2vec Clusters

TissUUmaps viewer can be integrated in Jupyter Notebooks. We provide a Python library such that the viewer can be loaded in a Jupyter Notebook cell and interactively display images along with markers. For this example, we use in situ sequencing data of mouse hippocampus. These data spots were generated by method ISTDECO [8], which is a decoding approach combining spectral and spatial deconvolution into a single algorithm. On this dataset, we applied the method spage2vec [9] — an unsupervised segmentation-free approach for decrypting the spatial transcriptomic heterogeneity of complex tissues at subcellular resolution.

The steps to achieve the integration of TissUUmaps viewer and spage2vec in this notebook are the following:

1. First, the necessary libraries are imported. Apart from the common Python data handling libraries, scipy, tensorflow, stellargraph, sklearn and TissUUmaps are required.

2. Next, the input data for spage2vec is loaded. The dataset needs to be a file containing three columns: X coordinate, Y coordinate, and the gene identity. Each row in the file represents an individual spot.

3. Then the spage2vec algorithm is run. The first step builds the graph by connecting the neighboring spots based on their spatial distances. The graph is used as input to the neural network. The network uses a graph representation learning technique based on a graph neural network. During training, the graph neural network learns the topological structure of each gene's local neighborhood. It does not require labeled training data but learns to find re-occurring patterns by comparing to a randomization of the

data. The network predicts a node embedding vector for each spot of the graph representing high-order spatial relationships with its local neighborhood. The network's architecture consists of two identical GraphSAGE [10] encoder networks sharing weights, taking as input a pair of nodes together with the graph structure and producing as output a pair of node embeddings. Thereafter, a binary classification layer with a sigmoid activation function learns to predict how likely it is that a pair will occur at a random position in the graph. Model parameters are optimized by minimizing a binary cross-entropy loss function between the predicted node pair labels and the true labels without supervision.

4. The next step is to apply clustering to the node embedding. Leiden clustering algorithm [11] is performed with an additional post-merging step, where we merge clusters with a gene expression correlation greater than 95%. The clusters represent combinations of different genes that can be identified as specific cellular substructures, cell types, or tissue domains.

5. Finally, the clustering results can be readily displayed in TissUUmaps.

This workflow can be applied to any code in Jupyter Notebooks. With TissUUmaps, one can visualize the input data, clustering results or the output from analyses. A video tutorial for how to integrate a code from Jupyter Notebook with TissUUmaps can be found here: `https://tissuumaps.github.io/tutorials/#spage2vec`.

SI 4 Exporting QuPath Regions to TissUUmaps

QuPath is a widely-used open-source software tool for digital pathology [12]. It supports the detection of cells and annotation of tissue regions in large image data files. Cells are segmented via nuclei detection and subsequent expansion of the nuclei area ("Cell detection"), or alternatively using the pre-trained models of StarDist [13], via the StarDist extension. Detected cells can be classified by presence/absence of a single marker ("Create single measurement classifier"). These classifiers are defined by manually setting a threshold on a marker measurement, e.g. mean marker intensity per cell, and can also be combined to classify the cells ("Create composite classifier"). QuPath also offers Machine Learning approaches – Random trees, Artificial Neural Network, K Nearest Neighbor – for cell classification ("Train object classifier"). The user can annotate tissue areas, e.g. tumor areas, by manually drawing annotations, by using a threshold on a single channel ("Create thresholder"), or by using pixel classification ("Train pixel classifier"). The outlines of cells and tissue regions can be exported into a GeoJSON file ("Export as GeoJSON"). TissUUmaps supports the loading of GeoJSON files and with that the loading of the outlines generated in QuPath. The user can then display these regions and extract information, e.g. counts per cell of a certain class.

SI 5 Spot Inspector Plugin

The In Situ Transcriptomic (IST) quality control plugin Spot Inspection allows the user to interactively click anywhere on an image displayed in TissUUmaps. This interaction opens a gridded figure displaying image content from multiple image rounds and channels as well as traces indicating the barcoded labels. The plugin enables an effective and interactive way for visualizing and interpreting errors introduced in IST pipelines. The Spot Inspector plugin is available for the TissUUmaps desktop client. It is implemented using Python with a JavaScript interface for communicating with the main TissUUmaps module.

Installation and Usage The plugin consists of the files spot_inspection.py and spot_inspection.js and is installed by placing the respective files in the "Plugin" folder (default path is `Program Files (x86)\TissUUmaps\Plugins`). The plugin is started in TissUUmaps by selecting the *Spot Inspection* option under the Plugin drop-down menu. Once the plugin is started, a new tab in TissUUmaps will appear that enables simultaneous loading of multi-round and multi-channel image data. The image data that is to be loaded must be named using the convention: {`Text1`}{`Round`}_{`Text2`}{`Channel`}`.tif` where {`Round`} and {`Channel`} are numeric values indicating which round and channel the corresponding tif file belong to, and {`Text1`} and {`Text2`} can be any string of characters. An image belonging to the first round and first channel could for instance be named `R1_CH1.tif`. Once the images are loaded into TissUUmaps, the user can click anywhere on the currently displayed image data to inspect the content from all image rounds and channels using a small gridded image. The barcoded labels of detected RNA species can also be visualized in the gridded image as small traces connecting different rounds and channels. This feature is only possible if the CSV file containing the marker data contains the additional columns *rounds* and *channels* that specifies which rounds and channels the barcoded labels are expected to be present in. Each entry in the round and channel column must be specified as comma-separated strings. For instance a barcode label present in rounds $\{0, 1, 2, 3\}$ and the channels $\{0, 0, 2, 2\}$ will have its corresponding round and channel attributes in the CSV file set to "0;1;2;3" and "0;0;2;2".

The source code of the plugin is available under `https://github.com/TissUUmaps/TissUUmaps/tree/master/tissuumaps/plugins_available`.

SI 6 Exporting Napari Projects to TissUUmaps

Napari [14] features an important hub containing 118 plugins at the time of writing, many of them expanding further the capabilities of Napari when dealing with biomedical imaging. We thus created our own plugin to allow users to work in Napari, benefit from the tools, scripting and existing plugins, and easily visualize and share the output of their research through TissUUmaps.

The plugin is available on Napari Hub which makes the installation trivial: from the Napari install/uninstall plugins menu, the "napari-tissuumaps" appears in the list and can be installed with a single click. Alternatively, the plugin can be installed with the Python package manager: "pip install napari-tissuumaps".

The plugin can export all standard Napari layers, such as images, labels, points, and shapes and preserves the metadata (opacity, visibility), but also the objects parameters (e.g.: label colors, marker colors and symbols, etc...). To export a TissUUmaps project, care must be taken to save all layers of interest and type in a name with the extension ".tmap", e.g.: "myProject.tmap". This is important for Napari to delegate the saving of the files to the plugin. A folder is created and contains all the necessary files and can be loaded in the TissUUmaps server, software, Jupyter Notebook, or shared with the community.

The project folders generated by the plugin contain the metadata in a "main.tmap" file, along with folders for each Napari layer types: images, labels, points and regions. Images and labels are saved as plain tif images, points are saved as CSV files, and shapes are saved as GeoJSON. We hope that the use of a simple structure and widespread file formats can simplify the modifying and updating of the TissUUmaps project when prototyping with e.g. Jupyter Notebooks. The source code is available at `https://github.com/TissUUmaps/napari-tissuumaps` under the permissive MIT license. A demonstration of the Cellpose plugin of Napari being exported to the TissUUmaps web viewer is available at: `https://tissuumaps.github.io/tutorials/#napari`.

SI 7 Exporting CellProfiler Data and Mitigating Tile Effects

Broad Institute's CellProfiler has long been used to perform high-throughput analyses [15, 16]. It allows an analysis pipeline to be run sequentially over a series of tiles for which large tissue samples are usually divided into during the imaging process in the microscope. Such division introduces a distinct pattern that receives the name "tile effect". It consists of clear line artifacts that become apparent once the analysis results are visualized on top of the original stitched image. This effect introduces errors in subsequent analysis.

To mitigate the tile effect, all the tiles can be analyzed on overlapping borders and duplicated cells (with their associated features) can be merged. Once all duplicated cells are merged, a unique CSV file can be produced and used to display features on top of the whole sample image.

For the analysis on the tile borders, the coordinates of each tile and its spatial location relative to the original image must be known, the coordinates have to be Euclidean and in a rectangular grid. The cell segmentation has to be deterministic and the same for all the tiles. The size of the border has to be known as well. If all of this is true then the tiles can be processed to give cells a global identifier within the tissue sample and remove tile effects.

CellProfiler output usually consists of a group of CSVs or a database per tile. These databases contain primary objects (in this case cells), secondary objects (like nuclei), local X and Y position and information on the tile and experiment they belong to. When all of this information is given, cells in the borders of tiles can be analyzed, matched and duplicates will be marked for deletion. They are only initially marked so that they can be verified visually first before actually removing them.

On a second step all the cells marked for removing duplicates will be merged. The result is a set of cells (represented as primary objects) where any object that was cut (due to being in an image border) or any duplicate object will be merged and solved.

SI 8 Interfacing Fiji plugins

We developed an example Python script for Fiji which shows how to interface Fiji plugins with TissUUmaps. The script showcases the registration of data that was captured through multiple rounds with DAPI and fluorescent staining. We consider that the fluorescent channels contain spots that need to be extracted from an e.g. in situ sequencing experiment. The DAPI channel is used to align the images together.

The example performs the following operations:

- Registration of rounds by using a single, common channel, usually a DAPI stain.
- Rigid transformation of all the fluorescent images such they are all aligned.
- Displaying the aligned images in Fiji
- Finding the spots in the fluorescent images and potentially export them as markers.
- Creating a TissUUmaps project file containing the aligned images along with the overlapping markers.

When running the script from Fiji, a user interface appears, allowing users to change directories, use different filenames and change the parameters of the alignment and dot detection.

The script is available at: `https://github.com/TissUUmaps/TissUUmaps/tree/master/examples`.

References

1. Palla, G. *et al.* Squidpy: a scalable framework for spatial single cell analysis. *bioRxiv* (2021).
2. Dries, R. *et al.* Giotto: a toolbox for integrative analysis and visualization of spatial expression data. *Genome biology* **22,** 1–31 (2021).
3. Abadi, M. *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. `https://www.tensorflow.org/`.
4. Chelebian, E. *et al.* Morphological Features Extracted by AI Associated with Spatial Transcriptomics in Prostate Cancer. *Cancers* **13,** 4837 (2021).
5. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome biology* **19,** 1–5 (2018).
6. Goode, A., Gilbert, B., Harkes, J., Jukic, D. & Satyanarayanan, M. OpenSlide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics* **4** (2013).
7. Becht, E. *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* **37,** 38–44 (2019).
8. Andersson, A., Diego, F., Hamprecht, F. A. & Wählby, C. ISTDECO: In Situ Transcriptomics Decoding by Deconvolution. *bioRxiv* (2021).
9. Partel, G. & Wählby, C. Spage2vec: Unsupervised representation of localized spatial gene expression signatures. *The FEBS Journal* **288,** 1859–1870. `https://doi.org/10.1111/febs.15572` (Oct. 2020).
10. Hamilton, W. L., Ying, R. & Leskovec, J. *Inductive Representation Learning on Large Graphs* 2018. arXiv: `1706.02216 [cs.SI]`.
11. Traag, V. A., Waltman, L. & van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* **9.** `https://doi.org/10.1038/s41598-019-41695-z` (Mar. 2019).
12. Bankhead, P. *et al.* QuPath: Open source software for digital pathology image analysis. *Scientific reports* **7,** 1–7 (2017).
13. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. *Cell detection with star-convex polygons* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018), 265–273.
14. Sofroniew, N. *et al. napari/napari: 0.4.12rc2* version v0.4.12rc2. Oct. 2021. `https://doi.org/10.5281/zenodo.5587893`.
15. Carpenter, A. E. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology* **7,** 1–11 (2006).
16. McQuin, C. *et al.* CellProfiler 3.0: Next-generation image processing for biology. *PLoS biology* **16,** e2005970 (2018).