Oct 20, 2025 Version 1

SpaHDmap Analysis Workflow for 10X Visium H&E Imaging Data V.1

Nature Cell Biology

DOI
https://dx.doi.org/10.17504/protocols.io.n2bvjebqxgk5/v1

Junjie Tang[1], Zihao Chen[1], Kun Qian[1]

[1]School of Mathematical Sciences and Center for Statistical Science, Peking University

SpaHDmap

kun qian

DOI: **https://dx.doi.org/10.17504/protocols.io.n2bvjebqxgk5/v1**

External link: **https://doi.org/10.1038/s41556-025-01838-z**

Protocol Citation: Junjie Tang, Zihao Chen, Kun Qian 2025. SpaHDmap Analysis Workflow for 10X Visium H&E Imaging Data. **protocols.io** **https://dx.doi.org/10.17504/protocols.io.n2bvjebqxgk5/v1**

Manuscript citation:
Tang, J., Chen, Z., Qian, K. *et al.* The interpretable multimodal dimension reduction framework SpaHDmap enhances resolution in spatial transcriptomics. *Nat Cell Biol* (2026). https://doi.org/10.1038/s41556-025-01838-z

Protocol status: Working
We use this protocol and it's working

Created: October 18, 2025

Last Modified: October 21, 2025

Protocol Integer ID: 230139

Keywords: spatial transcriptomics, multi-modal, dimension reduction, spahdmap analysis workflow, 10x visium st dataset, spahdmap, running spahdmap, 10x genomic, adult mouse brain posterior section, spot expression data, adult mouse brain, example data, spot expression data as input

Abstract

This tutorial will guide you through the process of running SpaHDmap on your data. Our example data is the 10X Visium ST dataset MPBS-01 sequenced from an adult mouse brain posterior section comprising a H&E image, we will take the H&E image and the spot expression data as input to run SpaHDmap. This data could be downloaded from **10X Genomics** or **Google Drive**.

Troubleshooting

Import necessary libraries

1 Please make sure you have installed SpaHDmap.

```
!pip install SpaHDmap
```

2 Then run

```
import torch
import numpy as np
import scanpy as sc

import SpaHDmap as hdmap
```

Set the parameters and paths

3 In this section, we will set the parameters and paths for the SpaHDmap model, including:
Parameters:

- rank: the rank / number of components of SpaHDmap model
- seed: the random seed
- verbose: whether to print the log information

Paths:

- root_path: the root path of the experiment
- project: the name of the project
- results_path: the path to save the results

3.1 **Parameters settings**

By default, we set the 'rank' to 20, the 'seed' to 123, and the 'verbose to True. You can modify the parameters according to your data.

```
rank = 20
seed = 123
verbose = True

np.random.seed(seed)
torch.manual_seed(seed)
```

3.2 **Paths settings**

These paths are set with respect to the current directory. You can modify the paths according to your data.

```
root_path = '../experiments/'
project = 'MPBS01'

results_path = f'{root_path}/{project}/Results_Rank{rank}/'
```

Load the data and pre-process

4 The data used in this tutorial is a 10X Visium ST dataset MPBS-01 sequenced from an adult mouse brain sagittal section, could be downloaded from **10X Genomics** or **Google Drive**.
You are able to utilize the function prepare_stdata for data loading and pre-precessing. This function supports loading data in two ways: directly loading an Anndata object or loading from file paths. There are several essential parameters, as follows:

- section_name: The name designating the section.
- image_path: The location path of the image.
- scale_rate: The scaling ratio applied to the input images. The default value is 1.
- create_mask: Whether to create a mask for the image. The default value is True.
- swap_coord: A parameter indicating whether to swap the row and column coordinates. The default value is True.

For more details of this function, please refer to the **API document**!

4.1 **Load data with an Anndata object**

We recommend to download / load the **Anndata object** using the api 'sc.datasets.visium_sge' from scanpy, and set 'include_hires_tiff=True' to download the hires image.

In this case, this data will be saved locally in the folder 'data/id_of_the_section', and next time you can load the data from the local folder.

```
section_id = 'V1_Mouse_Brain_Sagittal_Posterior'

# Download the data from the 10X website (set
include_hires_tiff=True to download the hires image)
adata = sc.datasets.visium_sge(section_id,
include_hires_tiff=True)
image_path = adata.uns["spatial"][section_id]["metadata"]
["source_image_path"]

# or load the data from a local folder
# adata = sc.read_visium(f'data/{section_id}',
count_file='filtered_feature_bc_matrix.h5')
# image_path = f'data/{section_id}/image.tif'
```

Then we are able to load and pre-process the data using function 'prepare_stdata', which will normalize both the gene expression and the image. Note that it will be fine if you provide an adata object whose expression data has already been normalized (but should not be scaled, the expression data has to be non-negative), this function will recognize the normalized expression data automatically.

Subsequently, we will utilize the function 'select_svgs' to select spatial variable genes (SVGs). By default, it will identify 3,000 SVGs in accordance with the Moran's I index. Alternatively, you may set method='sparkx' or method='bsp' to identify SVGs based on the SPARK-X or BSP methods respectively.

```
# Load the data from the 10X Visium folder
mouse_posterior = hdmap.prepare_stdata(adata=adata,

section_name='mouse_posterior',
                                       image_path=image_path,
                                       scale_rate=1)

hdmap.select_svgs(mouse_posterior, n_top_genes=3000,
method='moran')
```

'prepare_stdata' returns a STData object, which contains several key attributes for the analysis:

- .adata: An AnnData object holding the normalized gene expression data and spot metadata.
- .image: A NumPy array of the high-resolution tissue image.

- .spot_coord: The spatial coordinates for each spot.
- .radius and .scale_rate: Metadata defining the spot size and image scaling.
- .scores: An empty dictionary that will be populated with analysis results (e.g., NMF, SpaHDmap scores)."

4.2 **Load data from file paths**

Alternatively, you can load the data by providing the paths of spot coordinates, gene expression and image. In this case, you have to provide the 'radius', which is the radius of each spot measured in pixels on the original high-resolution image.

These data can be downloaded from the 10X Genomics website in:

- spot coordinates: 'Output and supplemental files → Spatial imaging data → tissue_positions_list.csv', please make sure that the first column is the spot names and the last two columns are the coordinates.
- gene expression: 'Output and supplemental files → Feature / barcode matrix (both filtered and raw are fine)', a hdf5 file or a csv file with the first row as the gene names and the first column as the spot names.
- image: 'Input files → Image (TIFF)'.

Then you can load and pre-process the data using the following code:

```
# Load the data from file paths
mouse_posterior =
hdmap.prepare_stdata(section_name='mouse_posterior',

image_path='data/V1_Mouse_Brain_Sagittal_Posterior/image.tif',

spot_coord_path='data/V1_Mouse_Brain_Sagittal_Posterior/spatial/ti
ssue_positions_list.csv',

spot_exp_path='data/V1_Mouse_Brain_Sagittal_Posterior/filtered_fea
ture_bc_matrix.h5',
                                       scale_rate=1,
                                       radius=45) # Has to be
provided if load from file paths

hdmap.select_svgs(mouse_posterior, n_top_genes=3000,
method='moran')
```

Run SpaHDmap

5 To run SpaHDmap, we need to initialize the 'Mapper' object, which contains the following attributes:

- 'section: the section object
- 'results_path': the path to save the results
- 'rank': the rank / number of components of the NMF model, default is 20
- 'reference': dictionary to assign the reference section for each query section, default is None (only for multi-sections)
- 'ratio_pseudo_spots': ratio of pseudo spots to sequenced spots, default is 5
- 'verbose': whether to print the log information, default is True

```
# Initialize the SpaHDmap runner
mapper = hdmap.Mapper(mouse_posterior, results_path=results_path,
rank=rank, verbose=verbose)
```

6 We can run all steps in one function `run_SpaHDmap` after obtaining the Mapper object.

```
# Run all steps in one function
mapper.run_SpaHDmap(save_score=False, save_model=True,
load_model=True, visualize=True, format='png')

# If you want to lower the GPU memory usage, you can set
`batch_size` to a smaller number
# mapper.args.batch_size = 16 (default 32)
# mapper.run_SpaHDmap(save_score=False, save_model=True,
visualize=False)
```