Genome-wide analysis of the dynamic and biophysical properties of chromatin and nuclear proteins in living cells with Hi-D

Marwan Abdellah[1†], Cesar Augusto Valades-Cruz[2,3†], Roman Barth[4], Haitham A. Shaban[5,6,7*]

[1]Independent Researcher, Chemin des Mines 9, Geneve 1202. [2]SERPICO Project Team, Inria Centre Rennes-Bretagne Atlantique Rennes, France. [3]SERPICO Project Team, UMR144 CNRS Institut Curie, PSL Research University Paris, France. [4]Department of Bionanoscience, Delft University of Technology Delft, The Netherlands. [5]Spectroscopy Department, Institute of Physics Research National Research Centre, Cairo, Egypt. [6]Agora Cancer Research Center Lausanne, Switzerland. [7]Precision Oncology Center, Department of Oncology Lausanne University Hospital, Lausanne, Switzerland. *[†]M.A. and C.A.V-C. have equal contributions. * Corresponding author: haitham.shaban@chuv.ch*

Abstract

To understand the dynamic nature of the genome in real time, the localization and rearrangement of DNA and DNA-binding proteins must be analyzed across the entire nucleus of single living cells. Recently, we developed a new computational light microscopy technique, called high-resolution diffusion mapping (Hi-D)[1], that can accurately detect, classify, and map the types of diffusion and biophysical parameters at a high spatial resolution over the entire genome in living cells. Hi-D combines dense Optical Flow to detect and track local chromatin and protein motion, and Bayesian inference to characterize this local movement at nanoscale resolution. The initial implementation requires some experience using MATLAB (MathWorks) and computational resources, for instance in form of access to a computer cluster, to perform the Hi-D analysis. In addition, this implementation takes ~18-24 hours to analyze a typical image series. To avoid these limitations and emphasize high-performance implementation, we present a customized version called Hi-D-Py. The new open-source implementation is written in the open source Python programming language, and options for parallelizing the calculations have been added to the code pipeline. We introduce user-friendly python notebooks and automatically generated reports that summarize and tabulate the results of the analysis. Our efficient implementation reduces the analysis time to less than one hour using a multi-core CPU with a single compute node. We also present different applications of Hi-D for live-imaging of DNA, H2B, and RNA Pol II sequences acquired with spinning disk confocal and super-resolution Structured Illumination Microscopy.

Keywords : *Hi-D, live-cell imaging, whole-nucleus analysis, chromatic dynamics, nuclear proteins, genome organization.*

1 Introduction

The dynamic organization of chromatin structures is highly relevant to cell function, which may be reflected in the relative distribution of open euchromatin and dense heterochromatin regions[2,3]. According to structural models, the genome is folded into multiple hierarchical levels of chromatin structure, from domain folding and long-range looping to compartments and subcompartments[4]. It is becoming clear that the dynamic nature of these nuclear compartments are involved in the regulation of their organization and thus affect the regulation of nuclear processes[5]. For example, the nucleosome density must be rearranged at transitions within and between euchromatin and heterochromatin facilitate DNA processing[6]. Remarkably conversed structural elements typically spanning 100-200 nm, topologically associated domains (TADs), were readily identified from population-averaged cross-linking frequencies ("C" data) provided by static models[7] and have since then been shown to play important roles in chromatin organization[8]. Yet, direct real-time observation of these domains, which is necessary to capture their dynamic nature and behavior, has been unattainable due to the lack of tools to analyze bulk chromatin motion at high resolution[5]. The dynamic activity of chromatin and nuclear proteins in living cells has been studied at high spatiotemporal resolution using specifically labeled molecules that can be identified and tracked in real-time using various techniques[9,10]. Subsequently, labeled loci and single genes can be tracked over various length and time scales using single particle tracking methods[11,12]. However, to fully understand the physical properties of a long-fiber structure, genomic chromatin must be comprehensively studied across the entire nucleus. However, it has been challenging to localize individual loci in the context of global chromatin architecture or surrounded by functional proteins over the nucleus during genome activity. Therefore, we recently developed the high-resolution diffusion (Hi-D) method[1]. The Hi-D method combines computer vision (dense Optical Flow) to quantify the local motion of bulk structures with sub-pixel resolution. A Bayesian inference is used to precisely classify types of motion throughout and to assign biophysical parameters to the motion modes throughout the nucleus. Using Hi-D, we have investigated the dynamic roles of chromatin (both DNA and histones), heterochromatin protein 1, RNA polymerase II, and transcription factors in genome organization. Here we present a framework written in Python as an accessible and user-friendly tool, with an improvement in processing time. Using this framework, we present a step-by-step Hi-D analysis of DNA and nuclear proteins imaged with spinning disks confocal and super-resolution microscopy. Documentation and code are released as part of the Hi-D-Py project (https://github.com/marwan-abdellah/hidpy).

1.1 Applications of Hi-D

Hi-D allows monitoring of spatiotemporal dynamics within nuclei of living cells in action. Recently, we have shown the applicability of the Hi-D method in studying the dynamic properties and precisely mapping the biophysical parameters of chromatin, RNA Pol II, and transcription factors, proteins at nanoscale resolution in single living cells[1,13,14]. We believe that the Hi-D method sets a base for dynamic analysis of not only nuclear proteins but other biomolecules, tissues, and many imaging scenarios involving dense signals where well-established methods may reach their limits. Results from our analysis method open new perspectives toward a global understanding of chromatin organization, loop formation, potentially the kinetics of gene expression, and a deeper understanding of dynamic processes in many biological fields. Hi-D has moreover shown the capability to be combined with conventional and super-resolution light microscopy techniques, and its amenability to screening different cell types[15–17].

1.2 Key features and advantages

1. Hi-D is highly effective in analyzing the dynamics of abundant molecules and dense structures directly in single living cells without sacrificing active fluorophore density.

2. There is no need for prior knowledge of sophisticated (sparse) labeling preparations or experience in using advanced microscopy techniques.

3. Hi-D is capable of overcoming the limitations of sparse and ensemble approaches such as single particle tracking (SPT) and fluorescence correlation spectroscopy (FCS) methods.

4. Hi-D is adequate to analyze images acquired with both conventional and super-resolution microscopy techniques.

5. Hi-D can be applied separately to all kinds of different fluorophores, as long as, those can be spectrally demixed, even in the same cell. This property enables the processing of multi-channel images to spatiotemporally correlate different nuclear factors, as shown in[1].

1.3 Limitations

The principal limitations of the existing protocol of the Hi-D technique[1] can be summarized in the following points:

1. The protocol was implemented mainly in MATLAB, a commercial software application that requires specific and hefty licensing. The GMM analysis module is implemented using the pomegranate package in Python. The interaction between MATLAB and Python required the user to establish the connection between MATLAB and python manually, which may limit unexperienced researchers to run the protocol in their research environments.

2. The execution of the current protocol is time-consuming; processing a single video sequence might take around 24 hours or more on mid-range commodity hardware. This issue limits the analysis of lengthy time lapses with specific experimental conditions on personal computers and might require high-end computing clusters for large-scale processing.

3. The existing protocol does not have a flexible interface; it is merely a group of scripts that run independently. The user is required to visualize and compare results of different cell lines / biological treatment manually. The code was designed as a proof-of-concept to demonstrate the principal idea of the Hi-D method.

1.4 Adaptations

We present this protocol to overcome the limitations of the existing one by:

1. Re-designing and implementing the workflow in Python, a high level and non-commercial programming language that does not need any commercial licensing, contrary to MATLAB.

2. Reducing the execution time of the entire workflow by using more efficient yet accurate implementations of the Optical Flow algorithm and making use of parallel execution environments supported by Python including the multiprocessing and joblib packages by default. Even with a single-threaded execution, the Python implementation is faster than the MATLAB-based one.

3. Making the software more user-friendly and easier to run, either via Python notebooks which guide the user in a step-by-step fashion to analyze the results on the fly, or via a simple python command line interface (CLI) that can be launched relying on configuration files or command line arguments. The results are generated in HTML reports that can be viewed in any browser.

1.5 Overview of the procedure

The full protocol of Hi-D is divided into two complementary procedures. The first procedure is performed in the laboratory to acquire video sequences of living cells showing the dynamics of chromatin and nuclear proteins. Further details on this procedure are available[1,5,9,18]. The second procedure (Stages 1 and 2) analyzes these sequences relying on Hi-D-Py in a matter of minutes. The presented protocol is focused on the Python workflow implemented for analyzing the video sequences. Nevertheless, we briefly describe a few details of the first procedure for users aiming to replicate the conditions and setups that were used to generate the presented sequences (refer to the video sequences provided in the **Supplementary Material**).

The Python workflow is composed of two stages. In Stage 1, the input sequence is pre-processed to compute the Optical Flow fields, estimate the trajectories, and apply Bayesian inference. We then select the most suitable models for deconvolution in Stage 2.

1.5.1 Cell culturing, staining and live-cell imaging, and sequence generation

1. *Cell-culturing and staining.* For H2B imaging, a human osteosarcoma U2OS cell line stably expressing green fluorescent protein (GFP) tagged histones (H2B) was used. The same cell line U2OS that expresses RPB1 (subunit of RNA Pol II) fused with Dendra2 is used for RNA Pol II imaging. Using an incubator at 37°C with 5% CO2, cells were grown in Dulbecco's modified Eagle's medium containing phenol red-free, Glutamax with 50 g/ml gentamicin, 1 mM sodium pyruvate, 10% fetal bovine serum and G418 with 0.5 mg/ml. At a density of about 1×10^5 cells per petri dish (#1.5 coverslip-like bottom), cells were plated for one day before the imaging. For H2B imaging using super-resolution structured illumination microscopy, HeLa H2B-GFP cells were seeded using phenol red-free DMEM enhanced with 10 M Hepes buffer. For DNA staining of the cultured same cell line, the SiR-DNA (5-610CP-Hoechst) dye was used. On the day of imaging, SiR-DNA was introduced to the medium of the cultured cells at 37°C for an hour after at a final concentration of 1 M. The cells were then gently washed three times with prewarmed phosphate-buffered saline. Before the nuclei imaging, the medium was changed to Leibovitz's L-15 Medium adding all the medium supplements needed for live cell imaging. This step was performed as well for cells that express H2B-GFP, RNA Pol II-Dendra2, and DNA-SiR for live chromatin imaging.

2. *Live-H2B and RNA Pol II imaging using spinning disk confocal microscopy.* The microscope CSU22 (Yokogawa) was used to record a 150-frame sequence with an exposure time per frame of 200 ms. A diode-pumped solid-state laser with a single wavelength of 488 nm (25 mW) was used to image the H2B-GFP. For RNA Pol II (Rbp1- Dendra2) imaging, a laser source (5-10% light intensity) with a

wavelength of 488 nm was applied. Imaging was performed with a Plan Apo (100×/1.42 NA) oil immersion objective from Nikon and detected using a cooled electron-multiplying charge-coupled device camera (iXon Ultra 888), with a sample pixel size of 88 nm. CAUTION The imaging was performed in a humid chamber at 37°C.

3. *Live-DNA imaging using spinning disk confocal microscopy*. The microscope CSU-X1-M1N (Yokogawa) was used to record a 150-frame sequence with an exposure time per frame of 200 ms of DNA-SiR. For DNA-SiR molecules imaging, an excitation laser source with 140 mW power and a wavelength of 647 nm was used. The emitted light was collected with a Leica HCX-PL-APO (100×/1.4 NA) oil immersion objective lens. A CMOS camera (ORCA-Flash4.0 V2) with (1×1 binning) was used to record the imaged videos with a sample pixel size of 65 nm. CAUTION The imaging was performed in a humid chamber at 37°C.

4. Live-H2B imaging using super-resolution structured illumination microscopy (SIM). Human (HeLa) somatic cell lines stably expressing GFP tagged H2B was imaged using SIM. The microscope DeltaVision OMX V3 Blaze was used to record a 20-frames series with an interval between frames of 2 seconds with sample pixel size of 41 nm. CAUTION At 37°C and 5% CO2 supply, live-cell imaging was carried out utilizing an objective heater and stage top incubator.

1.5.2 Stage 1: Steps 1–14, pre-processing, Optical Flow calculation, trajectories estimation, MSD calculations, and Bayesian inference.

After the generation of the video sequence, we applied Stage 1 to pre-process the frames of the input sequence, compute the trajectories and their mean square displacements (MSDs) and use Bayesian inference to test the different models for any MSD curve (Stage 1 in Figure 1). The pre-processing stage includes zero-padding, thresholding, denoising, and edge detection operations which are applied to match the specific requirements of the Optical Flow algorithm. The Optical Flow maps are computed using an efficient implementation from the OpenCV library[19], which is based on the Farneback algorithm[20]. For a sequence with N frames, N-1 Optical Flow maps are produced. Those maps are then interpolated and linked to compute trajectories on a per-pixel basis. Therefore, we allow the user to define a thresholding value based on image intensity to determine if the value of the pixel is significant enough to compute a trajectory or not.

The resulting trajectories are then listed in a linear list and mapped to a three-dimensional array for visualization. After trajectory estimation, the MSD curves are calculated for every trajectory. The estimated MSD is a function of the initial position and the time lag (Supplementary Eq. 1). A nucleus mask is estimated during the pre-processing stage and used to select only nuclear localized trajectories. Local MSD estimation is described in the original Hi-D publication[1].

Bayesian inference is used to test different models for any given MSD curve[21]. Given the data $Y = \{Y_1, ..., Y_n\}$, K model candidates $M = \{M_1, ..., M_K\}$ and parameter set $\theta = \{\theta_1, ..., \theta_K\}$, the goal is to find the model $M_k(Y, \theta_k)$ such that the probability of $M_k(Y, \theta_k)$ is maximal for the selected set of models; therefore,the optimal parameters for every model are calculated. A general multivariate Gaussian function[22] is used to represent the probability that the data Y is observed for a Model M_k described by the function $M_k(x; \theta_k)$ for any parameter set θ_k as shown in Supplementary Eq. 4. After estimating the best parameter set for a model, the model, and its chosen parameters are selected to maximize their probability to describe the data in which $\hat{\theta}_{k,MLE} = \arg\max_{\theta_k} P(Y|\theta_k, M_k)$.

MSD analysis is performed locally[1], only the 3×3 neighborhood of a pixel is chosen to determine the MSD covariance matrix used for the Bayesian inference[23]. The choice of a window size of 3×3 is based on the selected filter size in the Optical Flow estimation. All the calculations are implemented in Python. Scipy[24] is used for numerical fitting.

The type of diffusion characterizing each trajectory motion was chosen in an unbiased manner using the Bayesian inference described previously from a set of five common models to fit each trajectory's MSD. These methods were detailed earlier[1], and a summary of them is presented in Table 1. A constant offset o is added to consider experimental noise.

Table 1: Overview of MSD models.

Model	**Equation**
Free Diffusion (*D*)	$MSD_D(\tau) = 4D\tau + o$
Drift (*V*)	$MSD_V(\tau) = v^2\tau^2 + o$
Free Diffusion + Drift (*DV*)	$MSD_{DV}(\tau) = 4D\tau + v^2\tau^2 + o$
Anomalous Diffusion (*DA*)	$MSD_{DA}(\tau) = 4D_a\tau^a + o$
Anomalous Diffusion + Drift (*DAV*)	$MSD_{DAV}(\tau) = 4D_a\tau^a + v^2\tau^2 + o$

1.5.3 Stage 2: Steps 15–18, deconvolution by a General Mixture Model.

The Bayesian inference of the trajectory MSD's results in maps of the diffusion constant (D), anomalous exponent (A) and drift velocity (V) for every cell (the best model parameters $\hat{\theta}_{k,\mathrm{MLE}}$). These parameter values are represented in histograms and subsequently deconvolved by a General Mixture Model (GMM) approach[25]. Before starting the trajectory computation and Bayesian inference (in Stage 1), the user selects the maximum number of sub-populations n_{pop} to be expected within the data set, as well as the type of distribution to be used by the GMM. Without prior knowledge, the number of sub-populations can be set to 3, and typically both normal and log-normal distributions can be considered. The GMM is then evaluated on all types of distributions for a number of sub-populations from 1 to n_{pop}. The goodness of fit for each of those combinations is evaluated using the Bayesian information criterion[26] (*BIC*) (**Supplementary Eq.5**). The result of the deconvolution is a table representing which fractions of cells in the analyzed data set are best described by a particular combination of distribution type and the number of sub-populations for every parameter (D, A, and V). The user should inspect this table, as well as the GMM deconvolution of the very cell's parameters to determine the optimal combination of distribution type and the number of sub-populations for each parameter within the data set (as shown in **Figure** 2). When a decision has been reached, the user supplies the choice to the configuration file and starts the final deconvolution stage with the chosen model(s) to obtain deconvolved histograms as well as a mapping of the respective distributions within the cell nucleus. If the Jupyter notebook is used, the parameters are updated the in the parameters panel directly.

1.6 Evaluation of the protocol implementation in Python

Dense Optical Flow fields are reconstructed with the Gunnar Farneback's algorithm[20] using an existing and fast implementation with the OpenCV library[19]. A manual screen across the input parameters for the Gunnar Farneback algorithm was performed, and the performance of the Gunnar Farneback algorithm was evaluated against the previously used MATLAB implementation[1,27]. The evaluation was performed on a range of synthetically generated images which we used previously to assess to compatibility of the Optical Flow algorithm for fluorescently labeled nuclear proteins[1]. Here, the diffusion constant of diffusing emitters varied between $0.01 - 1$ px^2/s. These values correspond to a range of 1.8 nm^2/s (for a pixel size of 13.5 nm, such as for super-resolution imaging[16]) and 12 000 nm^2/s for confocal imaging. All emitters were subject to normal diffusion with the respective diffusion constants.

To account for the observed spatial correlation of nuclear proteins[18,28–31], we introduced $1 - 100$ coherently moving domains. Emitters falling within a domain experience an additional displacement which

was randomly chosen according to a free diffusion with the same diffusion constant as single emitters. The fixed parameters to generate synthetic images were as follows: 100 frames of 150 × 150 pixels were generated and the interval between frames was chosen as 200 ms. Emitters were randomly seeded within the images at a density of 50 emitters/μm^2. The ground-truth trajectories of simulated emitters were recorded and compared to MATLAB and all trials of the Gunnar Farneback implementation. An objective evaluation is performed by computing the angular error (AE), endpoint error (EE), and relative error (EE_{rel}) (**Supplementary Equations 6, 7, and 8**).

We concluded the parameter screen of the Gunnar Farneback after 14 trial runs. The trial run parameters are summarized in **Supplementary Table S1**. Several parameter combinations yielded similar results and were thus ranked together. Exemplary heatmaps showing the AE, EE, and EE_{rel} of trials 2, 3, 7, 8, and 13 are shown in **Figure** 3. From the Gunnar Farnback implementation parameters, the most influential parameters according to first- and second-order correlations with the manually assigned ranking of trial runs (refer to **Supplementary Table S1**) are the number of levels, the window size (winsize), and the number of iterations, as displayed in **Supplementary Table S2**. **Figure** 4 shows a comparison between the results of the Bayesian analysis[1] using MATLAB and that resulting from our Python implementation. Similar results are shown for the least-square fitting from MATLAB Curve fitting toolbox and Python Scipy library.

1.7 Installation · TIMING a few minutes

The code is installed in a few easy steps as follows:

1. Go to the open-source GitHub repository at https://github.com/marwan-abdellah/hidpy.
2. If you have a GitHub account, clone the repository to download its contents into the file system. **CRITICAL STEP** If you wish to contribute to the code, you can fork the repository before cloning it.
3. If you do not have a GitHub account, you can download the project as a .Zip file.

1.8 Code execution · TIMING less than 1 hour

Hi-D-Py can be executed as Python notebooks located in the notebooks directory or from a command line interface (CLI) either using individual command line options or relying on preset configuration files (refer to **Figure** 1). Stages 1 (hidpy_stage1.py) and 2 (hidpy_stage2.py) are executed with the same configuration file. Setting the parameters of the configuration file demo.config, the workflow is executed by running the

scripts using the following commands: hidpy_stage1.py –-config demo.config and hidpy_stage2.py –-config demo.config. Depending on the output results of Stage 1, the configuration file is updated to run the python script of Stage 2 (hidpy_stage2.py). The CLI and configuration parameters are detailed in the **Procedure**. The results of each stage are stored in an HTML report that can be opened with any web browser. The code can be also executed in a step-by-step fashion using documented Jupyter notebook, where users visualize the results of each step in the notebook. Note that the code supports parallel execution using all available CPU cores in your computing node.

2 Materials

2.1 Biological materials

A list of all the biological materials including the cell lines, chemicals, and relevant data are summarized in Table 2.

2.2 Equipment

1. Spinning disk confocal microscopy system (CSU-X1-M1N, Yokogawa). This system is used to image live- DNA 150-frames sequences with an exposure time of 200 ms per frame.

2. A Nipkow-disk confocal microscopy system (CSU22, Yokogawa). This system is used to image live H2B and RNA Pol II 150-frames sequences with an exposure time of 200 ms per frame.

3. Super-resolution structured illumination microscopy (DeltaVision OMX V3 Blaze). This system is used to image 20 frames of live H2B sequences with a frame duration of 2 seconds. The system is equipped with a PlanApo 60/1.42 oil immersion objective, a 488 nm laser, and sCMOS cameras.

4. Standard personal computer. All protocols have been tested on three machines with different operating systems to ensure that our implementation is cross-platform. The reference machine runs Linux (Ubuntu 20.04) and has an Intel Corei9 CPU with 2.8 GHz and 128 GB of DDR4 memory. Hi-D-Py can be downloaded and run locally as a stand-alone application on Windows, Linux, and macOSX machines.

Table 2: Key resource table for all the biological materials used in this protocol.

Resource or reagent	Source	Identifier
Dulbecco's Modified Eagle Medium, high glucose, pyruvate	ThermoFisher	Cat#41966029
Fetal Bovine Serum	ThermoFisher	Cat#10270106
Sodium pyruvate solution	Sigma-Aldrich	Cat#113-24-6
Penicillin-Streptomycin	Bioconcept	Cat#4-1F00H
Puromycin	Thermofisher	Cat#A1113803
Glutamax	ThermoFisher	Cat#35050061
Trypsin-EDTA-Solution	Sigma-Aldrich	Cat#T4049
Doxycycline hyclate	Sigma-Aldrich	Cat#D9891
SiR-DNA (5-610CP-Hoechst)	Grazvydas Lukinavicius	N/A
L-15 Medium (Leibovitz)	ThermoFisher	Cat#11415064
HEPES buffer	Sigma-Aldrich	Cat#7365-45-9
Live-cell images of DNA, H2B, and RNA pol II data (Spinning disk confocal microscopy)	Shaban et al., Genome Biology, 2020	Reference [1]
Live-cell images of H2B data (Structured illumination microscopy)	Miron et al., Science Advances, 2020	Image Data Resource (IDR0086)
U2OS cell line	ATCC	N/A
U2OS cell line stably expressing H2B-GFP	ATCC. A gift from Sébastien Huet (Rennes)	N/A

2.3 Software

1. Scientific computing package: OpenCV (opencv.org) and its Python bindings.
2. Python's scientific computing package: Numpy (numpy.org).
3. Python's scientific computing package: Scipy (scipy.org).
4. Python's plotting package: Matplotlib (matplotlib.org).
5. Python's plotting package: Seaborn (seaborn.pydata.org).
6. Python's probabilistic models' package: Pomegranate (pomegranate.readthedocs.io).

7. Python's pipelining package: joblib (joblib.readthedocs.io).

8. Python's concurrency package: multiprocessing (docs.python.org/3/library/multiprocessing.html).

9. Image processing application: Fiji (https://imagej.net/software/fiji). This is an optional package.

10. Jupyter notebook with Visual Studio Code (https://code.visualstudio.com).

All the Python packages can be directly installed by relying on the pip command. The code has been tested on Unix-based operating systems (Ubuntu 20.04 and macOSX 11) and Windows 10. It is recommended to use Visual Studio Code (code.visualstudio.com) to run the Python notebooks.

3 Procedure

3.1 Stage 1: Pre-processing, Optical Flow calculation, trajectories construction, MSD estimation, and Bayesian inference · TIMING minutes to one hour

1. CAUTION We recommend downloading the provided datasets (**Supplementary Data**) and using the Jupyter notebooks as a *step-by-step* guide to getting an understanding of the code with a validated example. A full example is shown in **Figure 5** for analysis of H2B in U2OS, data is available in (**Supplementary Data**).

2. Select and crop single nuclei and convert the input video sequence into an AVI or TIFF file format. TROUBLESHOOTING.

3. If you are using the CLI, fill in the parameters of the configuration file required to run Stage 1 successfully including (i) the absolute path to the input video sequence, (ii) the pixel size, (iii) the time step, (iv) the pixel threshold value required for trajectory estimation and (v) the output directory where the results will be generated. TROUBLESHOOTING.

4. If you are using the Python notebooks, the input parameters are set in the inputs panel of the notebook 01_hidypy_stage1.ipynb. TROUBLESHOOTING.

5. *Denoising*. Each frame of the entire sequence is denoised using non-iterative bilateral filtering to verify the difference in intensity values to preserve the edges of the structures. This step was essential in the MATLAB-based implementation, but it is made optional in the current protocol due to the stability of the Optical Flow implementation of the Farneback algorithm[32] in OpenCV. TROUBLESHOOTING If the input video sequence contains severe salt-and-pepper noise, users may use Fiji to apply a bilateral filter to reduce this noise. CAUTION Another option is a median

filter but this filter could alter pixel intensity values non-linearly.

6. *Photobleaching detection*. Compute the relative intensity between the first and last frames. CAUTION If this value is lower than 10 - 15 %, photobleaching can be considered high and anticipated results might be inaccurate. This also depends on the length of the sequence.

7. *Optical Flow computations*. Compute the optical flow fields of the entire sequence using the Farneback algorithm[32] implemented in the OpenCV library. For a video sequence of N frames, a list of N-1 Optical Flow fields is computed.

8. *Trajectory estimation*. Starting from the first frame, compute the trajectories of every pixel whose value is above a given intensity threshold defined by the user. Trajectory computations are implemented without taking into consideration the dimensions of the pixel. The pixel size is defined by the user in the input data before running the protocol based on the configuration of the microscope used to image the sequence. Therefore, we scale each estimated trajectory by multiplying the computed coordinates by the pixel size to convert from pixel units to micrometers. We then use the Scipy library to interpolate the trajectories.

9. *Verify the resulting trajectories*. Map the estimated trajectories on the first frame of the sequence to verify if the trajectories are located within the extent of the nucleoli or not. TROUBLESHOOTING

10. *Trajectory maps reconstruction*. Once the estimated trajectories are calculated and verified, they are converted to displacement maps (in microns). TROUBLESHOOTING.

11. *MSD estimation*. For each trajectory inside the nucleoli mask, the MSD is estimated.

12. *Bayesian inference*. Bayesian inference[1,21] is applied to trajectories and thereby the best-fitting model is estimated considering the model type and the corresponding parameters (D, A, V). CAUTION This step requires the time step (Δt) between the frames in the input sequence. This step should be set manually by the user depending on the imaging procedure and the configuration of the microscope. TROUBLESHOOTING.

13. *Model selection and parameters are mapped*. Examples of these mapping are shown in **Figure 5c,d.** TROUBLESHOOTING.

14. *Storing the Bayesian results*. The Bayesian results for all the parameters (D, A, and V) are stored in a pickle file for processing in the next stage.

3.2 Stage 2: Deconvolution by a general mixture model · TIMING a few seconds to a minute

15. *Setting deconvolution parameters.* Before GMM deconvolution is performed, users must choose the maximum number of distributions (n_{pop}), the model parameter to deconvolve and the type of distribution (normal or log-normal) to be used by the GMM. The input to the GMM step is the raw values resulting from the Bayesian inference stage **(Stage 1**). Model parameter values are represented in histograms before deconvolution. CAUTION The number of histogram bins is selected by the user for visualization purposes. If you are using notebooks, set the input parameters of the pickel files directory and models in the input parameters panel of the notebook 02_hidypy_stage2.ipynb.

16. *GMM evaluation.* GMM is evaluated on all types of distributions and for subpopulations (1 to n_{pop}). Evaluation is done using the Bayesian information criterion (*BIC*). The deconvolution results are summarized in a table representing the fraction of cells from the analyzed dataset. The results are best described by a combination of distribution type and the number of sub-populations for every model parameter.

17. *GMM table evaluation.* Users should inspect the GMM table and select the optimal combination of distribution type and the number of sub-populations for each parameter within the analyzed data set.

18. *Deconvolution.* GMM deconvolution with the chosen model(s) is performed and population distributions are mapped within the cell nucleus for each selected model parameter. Final results from GMM deconvolution in example workflow are shown as histograms (see **Figure 5e**) and maps (see **Figure 5f**) for each biophysical parameter. TROUBLESHOOTING.

4 Troubleshooting

Troubleshooting recommendations for anticipated issues are summarized in **Table** 3.

5 Timing

The timing reported in the current protocol is registered based on several video sequences with 150 frames. However, we expect a total running time – on average– between 15 minutes to 40 minutes to run the entire pipeline. This timing is computed based on the hardware described in the **Materials sections**. The processing time of the Optical Flow and Bayesian inference steps depends on the number of cores in the CPU if the multicore functionality is enabled. This assumes that the running machine has sufficient memory if lengthy video sequences are supplied. The timing is estimated under the assumption that the user has basic coding experience in Python and understands how to run Python notebooks either via Jupyter notebooks or relying on Visual Studio Code (Microsoft). Average timing estimates for the entire protocol based on the exemplar datasets are listed in **Table** 4.

Table 3: Troubleshooting table

Step	Problem	Possible Reason	Solution
2	Input video sequences are not loaded.	Unsupported file format.	Use Fiji to convert the sequence into an AVI file.
3 or 4	Results are not generated to the specified output directory.	The directory is not valid.	Verify the path of the output directory.
5	Noisy video sequences after bilateral denoising.	Imaging artifacts such as noise.	Use Fiji to denoise the sequence using median filters
9	Trajectories are appearing outside the locus of the nuclei.	Imaging artifacts such as noise.	Re-adjust the pixel threshold value and repeat the Steps 3 − 9 until a convenient mapping is reached.
9	Trajectories all move collectively in a certain direction.	Nucleus contains drift	Drift may be eliminated using Fiji or another image processing program. Care should be tacken to correct lateral drift in a smooth manner and only using interger pixel values to avoid interpolation of pixel's intensity which alters pixel intensity values and often blurs images.
10	Parameters are higher than expected.	Error in units.	Verify pixel size and time step in input configuration. The units for the pixel size are micrometers.
12	Execution failure in case of long or high resolution videos.	Memory limitations.	Use a single core for processing instead of the multiple cores.
12	Fitting failure in Bayesian inference.	Fitting parameters are not adapted.	Correct fitting parameters in the python file core/inference.py
13	Bayesian maps show a region outside of the cell.	Error in pixel-threshold.	Adjust the pixel intensity threshold value.
14	Error empty sequence in deconvolution stage.	Empty pickle folder.	Verify that at least one pickle file is in the folder given to the deconvolution stage.
18	Execution failure during GMM constrained.	Model fitting did not converge.	Try different number of population and/or distribution type.

The timing also depends on the following factors:

- *Video length*. While the protocol is applied to video sequences with 150 frames, it is assumed that it can run with input sequences with thousands of video frames as long as it has a high signal-to-noise ratio, negligible photobleaching, and corrected lateral drift.
- *Pixels per frame*. The computations of the optical flow fields depend on the number of pixels per frame.
- *Pixel threshold value*. Increasing the value of the pixel threshold will reduce the number of calculated

trajectories.

- *Selected models*. Bayesian inference implies a fitting process per trajectory for each selected model, population number, and type of statistic distribution. Execution time is highly dependent of the combination of these factors.

Table 4: Average timing estimates for the entire protocol.

Time estimates for the development and execution of protocol steps	
Process	Execution
Data Preprocessing	Several seconds to a few minutes.
Optical Flow computations	A few seconds.
Trajectory computations	A few minutes.
MSD-Bayes Analysis	A few to several minutes.
GMM Deconvolution	A few seconds.

6 Anticipated results

The Hi-D-Py software provides an open-source improved implementation of the former Hi-D algorithm. This method permits quantification and classification of maps of diffusion and biophysical parameters over the entire genome in living cells. Hi-D-Py takes advantage of the Python programming environment to improve the speed and user friendliness through Jupyter notebooks. Hi-D-Py has been applied to two different microscopy techniques showing versatility. First, it has been validated on structured illumination microscopy (SIM) for the analysis of H2B (see **Figure 6a-d**). It has also been tested on a spinning disk for the analysis of DNA (see **Figure 6e-h**) and RNA (see **Figure 6i-l**).

As a characteristic example, a full Hi-D-Py workflow was applied to a spinning disk confocal sequence of H2B-GFP-stained nuclei of U2OS cell (see **Figure 5a**). First, trajectories estimated from **Stage 1** are shown in **Figure** 5b. Then, Bayesian inference was applied. Hi-D-Py generates maps of models(see **Figure 5c**) and bio-physical parameters (see **Figure 5d**). Finally, deconvolution by a General Mixture Model was applied to the biophysical parameters (D, A, V). The result of this deconvolution stage is a particular combination of distribution type and the number of sub-populations for every parameter. Histograms and maps of populations are shown in **Figures 5e,f**, respectively.

The different mobility classes of the diffusion constant, anomalous exponent, and drift velocity reflect the cell's dynamics landscape. Outputs generated by Hi-D-Py may be used to determine the impact of a certain biological condition (a drug interfering with nuclear processes or external stimuli) on a cell to determine mobility differences between cell lines. The found number of populations is informative about e.g. the extent of chromatin-bound, chromatin-processing, and unbound fractions of a nuclear molecule. Similarly, chromatin mobility is heterogeneous across the nucleus and mobility (re-)arrangements may be correlated with nuclear processes such as gene regulation or local compaction of chromatin. The mean/median values of D, A, and V per sub-population allow to express mobility changes quantitavely. We thus encourage the chromatin organization community to complement the presented Hi-D method with external stimuli and additional markers of nuclear processes of interest, both locus-specific, as well as genome-wide. The incorporation of Hi-D-Py as a simple, fast and user-friendly analysis package for nuclear mobility greatly enhances the aspects of nuclear biolgy research beyond static and population-average methods as well as restrictive sparse loci methods.

6.1 Software availability

The code is open source and freely available online on GitHub at https://github.com/marwan-abdellah/Hidpy. The version of the code that was used to create the results demonstrated in tagged **v.1** on the GitHub repository. The code in this protocol has been peer-reviewed by all the co-authors.

Supplementary Data

The example datasets that are used in this protocol are available in the Supplementary Material.

Acknowledgments

H.A.S acknowledges the financial support of the ISREC foundation. C.A.V-C acknowledges the funding by Labex Cell(n)Scale (ANR-11-LABX-0038) as part of the Idex PSL (ANR-10-IDEX-0001-02). M.A. and R.B. contribute to this work independently.

Authors' contributions

M.A. and H.A.S. co-conceived the study. M.A implemented the core framework, the Optical Flow, and trajectory computations. H.A.S. performed the wet lab experiments and imaging protocols. R.B. implemented the validation modules. C.A.V-C. implemented the Bayesian and deconvolution modules. M.A., C.A.V-C., R.B., and H.A.S wrote the manuscript. M.A. and C.A.V-C. contributed equally to this work. H.A.S. supervised the project. All the authors approved the manuscript.

Competing financial interests

The authors declare no competing financial interests.

Additional information

Correspondence and requests for materials should be addressed to H.A.S.

References

1. Shaban, H. A., Barth, R., Recoules, L. & Bystricky, K. Hi-D: nanoscale mapping of nuclear dynamics in single living cells. *Genome biology* **21,** 1–21 (2020).
2. Misteli, T. The self-organizing genome: principles of genome architecture and function. *Cell* **183,** 28–45 (2020).
3. Agbleke, A. A. *et al.* Advances in chromatin and chromosome research: perspectives from multiple fields. *Molecular cell* **79,** 881–901 (2020).
4. Dekker, J. & Mirny, L. The 3D genome as moderator of chromosomal communication. *Cell* **164,** 1110–1121 (2016).
5. Shaban, H. A., Barth, R. & Bystricky, K. Navigating the crowd: visualizing coordination between genome dynamics, structure, and transcription. *Genome biology* **21,** 1–18 (2020).
6. Klemm, S. L., Shipony, Z. & Greenleaf, W. J. Chromatin accessibility and the regulatory epigenome. *Nature Reviews Genetics* **20,** 207–220 (2019).
7. Su, J.-H., Zheng, P., Kinrot, S. S., Bintu, B. & Zhuang, X. Genome-scale imaging of the 3D organization and transcriptional activity of chromatin. *Cell* **182,** 1641–1659 (2020).
8. Nora, E. P. *et al.* Spatial partitioning of the regulatory landscape of the X-inactivation centre. *Nature* **485,** 381–385 (2012).
9. Shaban, H. A. & Seeber, A. Monitoring global chromatin dynamics in response to DNA damage. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis* **821,** 111707 (2020).
10. Shaban, H. A. & Seeber, A. Monitoring global chromatin dynamics in response to DNA damage. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis* **821,** 111707 (2020).
11. Germier, T. *et al.* Real-time imaging of a single gene reveals transcription-initiated local confinement. *Biophysical journal* **113,** 1383–1394 (2017).
12. Gu, B. *et al.* Transcription-coupled changes in nuclear mobility of mammalian cis-regulatory elements. *Science* **359,** 1050–1055 (2018).
13. Barth, R. & Shaban, H. A. Spatially coherent diffusion of human RNA Pol II depends on transcriptional state rather than chromatin motion. *bio*R*xiv* (2022).
14. Shaban, H. A. & Suter, D. M. Individual activator and repressor transcription factors induce global changes in chromatin mobility. *bio*R*xiv* (2022).
15. Miron, E. *et al.* Chromatin arranges in chains of mesoscale domains with nanoscale functional topography independent of cohesin. *Science advances* **6,** eaba8811 (2020).
16. Barth, R., Bystricky, K. & Shaban, H. Coupling chromatin structure and dynamics by live super-resolution imaging. *Science advances* **6,** eaaz2196 (2020).
17. Barth, R., Fourel, G. & Shaban, H. A. Dynamics as a cause for the nanoscale organization of the genome. *Nucleus* **11,** 83–98 (2020).
18. Shaban, H. A., Barth, R. & Bystricky, K. Formation of correlated chromatin domains at nanoscale dynamic resolution during transcription. *Nucleic acids research* **46,** e77–e77 (2018).

19. Bradski, G. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer* **25,** 120–123 (2000).

20. Farnebäck, G. *Two-framemotion estimation based onpolynomialexpansion* in *Scandinavianconference on Imageanalysis* (2003), 363–370.

21. Monnier, N. *et al.* Bayesian approach to MSD-based analysis of particle motion in live cells. *Biophysical journal* **103,** 616–626 (2012).

22. Seber, G. & Wild, C. *Nonlinear regression* (John Wiley & Sons, 2003).

23. Rousseeuw, P. J. & Croux, C. Alternatives to the Median Absolute Deviation. *Journal of the American Statistical Association* **88,** 1273–1283 (1993).

24. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17,** 261–272 (2020).

25. Schreiber, J. M. & Noble, W. S. Finding the optimal Bayesian network given a constraint graph. *PeerJ Computer Science* **3,** 122 (2017).

26. Wit, E., Heuvel, E. v. d. & Romeijn, J.-W. 'All models are wrong...': an introduction to model uncertainty. *Statistica Neerlandica* **66,** 217–236 (2012).

27. Sun, D., Roth, S. & Black, M. J. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision* **106,** 115–137 (2014).

28. Barth, R. & Shaban, H. A. Spatially coherent diffusion of human RNA Pol II depends on transcriptional state rather than chromatin motion. *Nucleus* **13.** PMID: 35723020, 194–202 (2022).

29. Zidovska, A., Weitz, D. A. & Mitchison, T. J. Micron-scale coherence in interphase chromatin dynamics. *Proceedings of the National Academy of Sciences* **110,** 15555–15560 (2013).

30. Di Pierro, M., Potoyan, D. A., Wolynes, P. G. & Onuchic, J. N. Anomalous diffusion, spatial coherence, and viscoelasticity from the energy landscape of human chromosomes. *Proceedings of the National Academy of Sciences* **115,** 7753–7758 (2018).

31. Salari, H., Di Stefano, M. & Jost, D. Spatial organization of chromosomes leads to heterogeneous chromatin motion and drives the liquid-or gel-like dynamical behavior of chromatin. *Genome Research* **32,** 28–43 (2022).

32. Aminfar, A., Davoodzadeh, N., Aguilar, G. & Princevac, M. Application of optical flow algorithms to laser speckle imaging. *Microvascular research* **122,** 52–59 (2019).

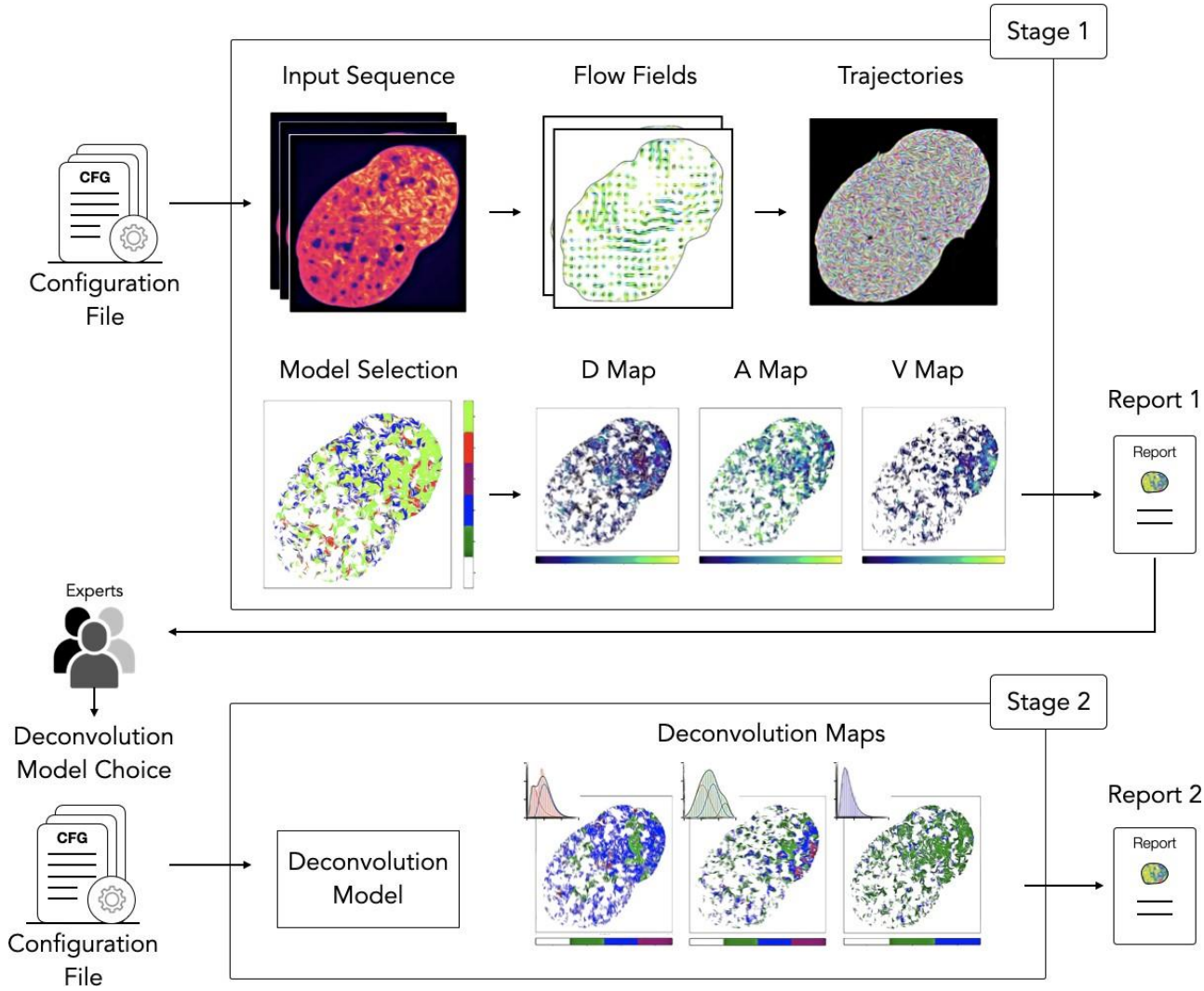
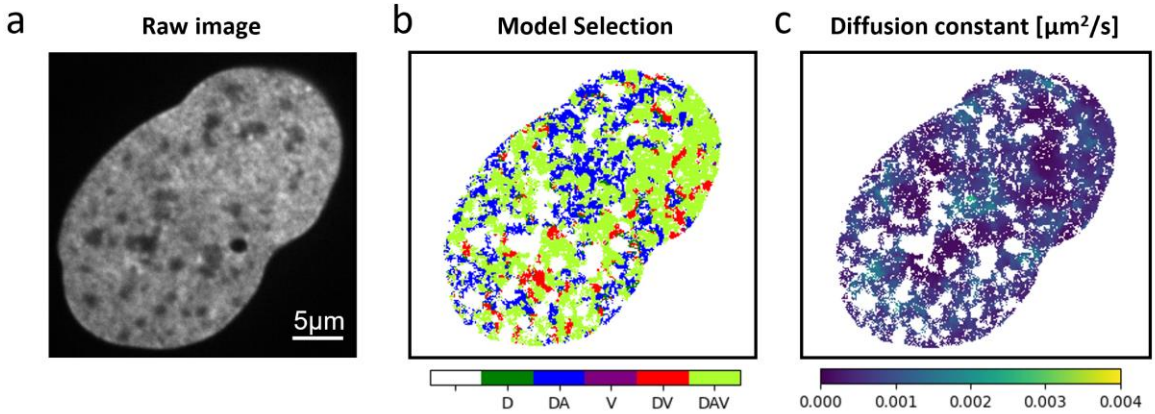
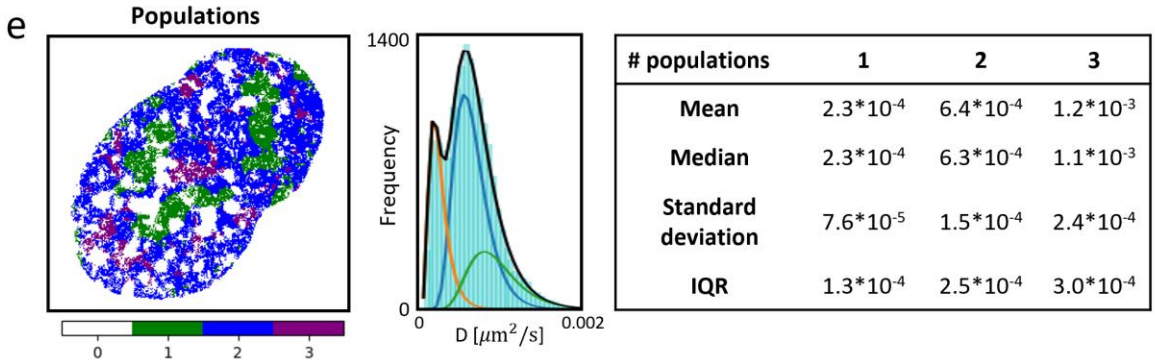


Figure 1 | Overview of the Hi-D-Py workflow. This workflow is composed of two stages. In Stage 1, we pre-process the input sequence, compute the flow field and trajectories and then apply Bayesian inference to compute the parameters needed for the decon- volution. The results are generated in an HTML report where experts can read and select the best-fitting models for deconvolution in Stage 2. The parameters of the workflow are set in input configuration files for simplicity, and the output results are generated in HTML reports.



d

	Normal distribution			Log-normal distribution		
# populations	1	2	3	1	2	3
D	0	0	0	0	0.333	**0.667**
A	0.333	**0.667**	0	0	0	0
V	0	0	0	0.333	**0.667**	0



# populations	1	2	3
Mean	$2.3*10^{-4}$	$6.4*10^{-4}$	$1.2*10^{-3}$
Median	$2.3*10^{-4}$	$6.3*10^{-4}$	$1.1*10^{-3}$
Standard deviation	$7.6*10^{-5}$	$1.5*10^{-4}$	$2.4*10^{-4}$
IQR	$1.3*10^{-4}$	$2.5*10^{-4}$	$3.0*10^{-4}$

Figure 2 | Deconvolution by a General Mixture Model (GMM) approach in Hi-D-Py. H2B-GFP-stained nuclei of U2OS cells imaged with Δt = 200 ms. Optical Flow estimation and Bayesian inference were done using Hi-D-Py. Representative maps are shown for one cell: raw (**a**), model selection (**b**), diffusion constant [μm^2 /s] (**c**). (**d**) Table with the distribution type and the number of sub-populations for each parameter (diffusion constant (D), anomalous exponent(A), and drift velocity (V)) within the data set (n=3 cells). The optimal combination is shown in bold numbers for every parameter. Estimated populations map (left **e**) and histograms (center **e**) were estimated using the optimal combination for parameter D. Diffusion coefficient statistics [μm^2 /s] for each population are shown in (right **e**).

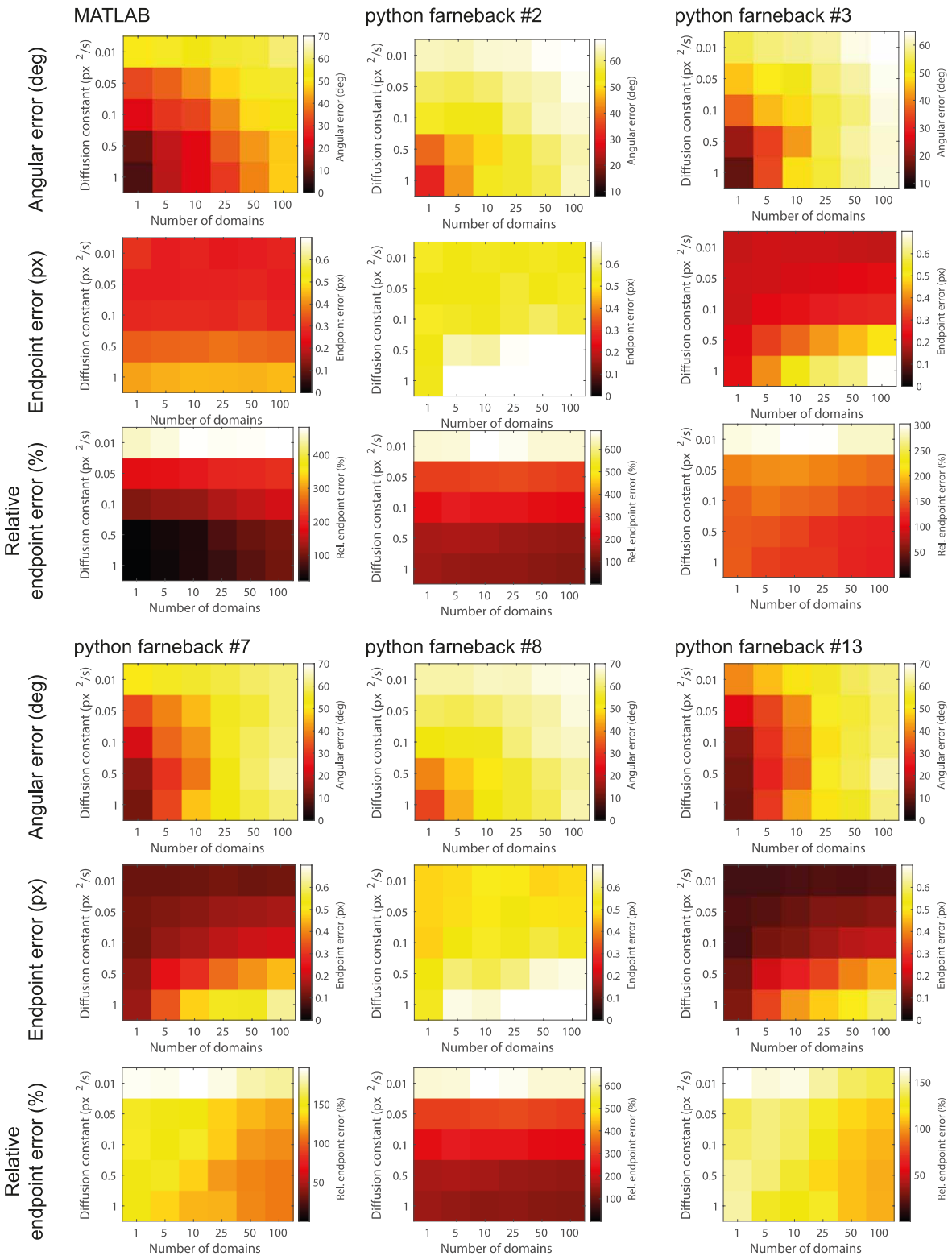


Figure 3 | Comparison between the used Optical Flow implementation in MATLAB and python. For the MATLAB as well as selected trial runs, heatmaps are shown which display the AE, EE and EErel over the varied diffusion constant and the number of simulated domains.

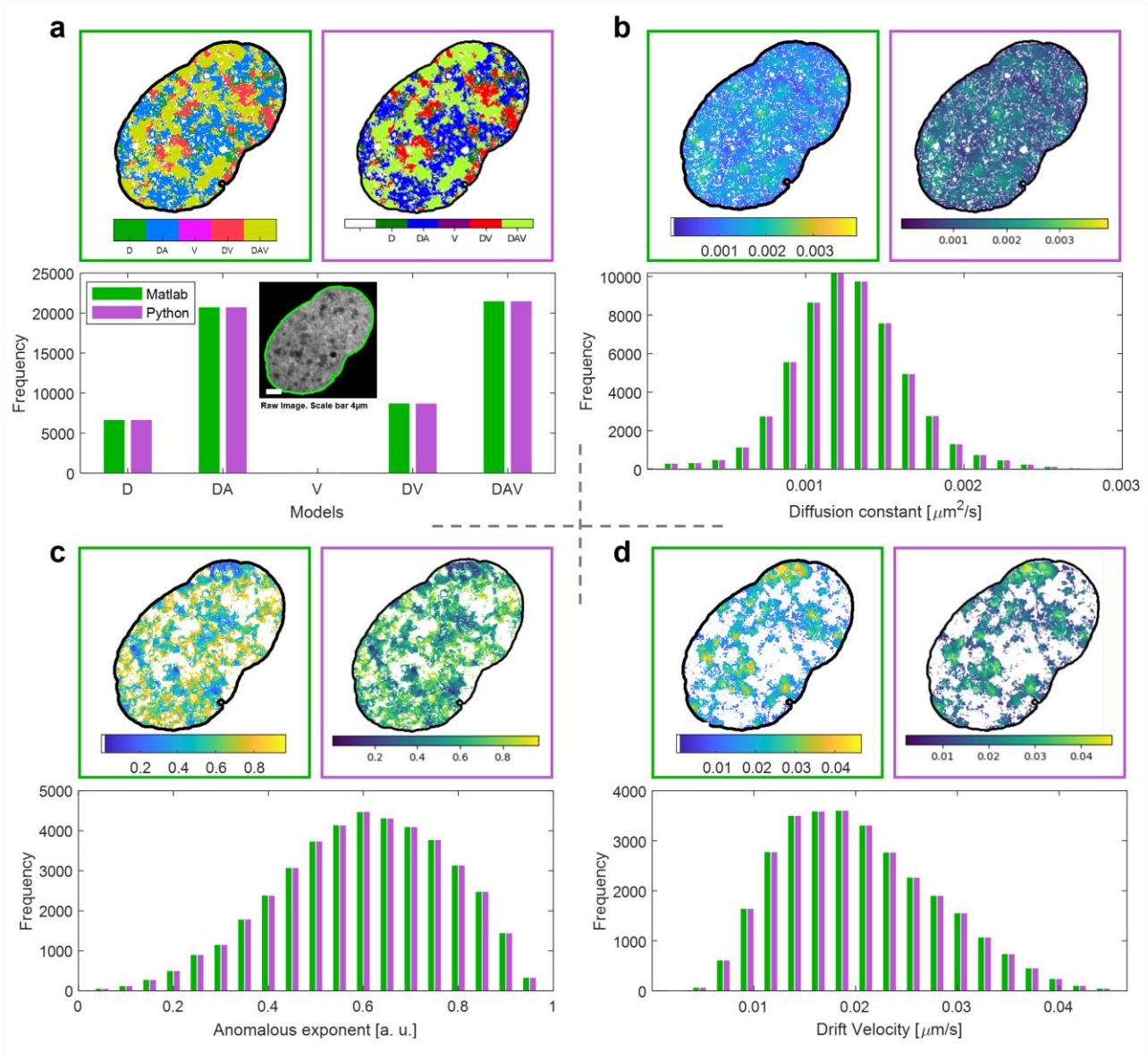


Figure 4 | Comparison between the Bayesian implementation in Hi-D(MATLAB) and Hi-D-Py(python). H2B-GFP-stained nuclei of U2OS cell (bottom inset **a**) were imaged with Δt = 200 ms. Trajectories estimated by Hi-D were used as input for the Bayesian modules of Hi-D and Hi-D-Py implementations. Bayesian inference of trajectory MSD's results is presented as maps of the selected models (upper **a**), diffusion constant (upper **b**), anomalous exponent (upper **c**), and drift velocity (upper **d**). Hi-D maps are inside the green squares, and Hi-D-Py maps are in magenta squares. Histograms of selected models(bottom **a**), diffusion constant(bottom **b**), anomalous exponent(bottom **c**) and drift velocity (bottom **d**) values are shown for both implementations.

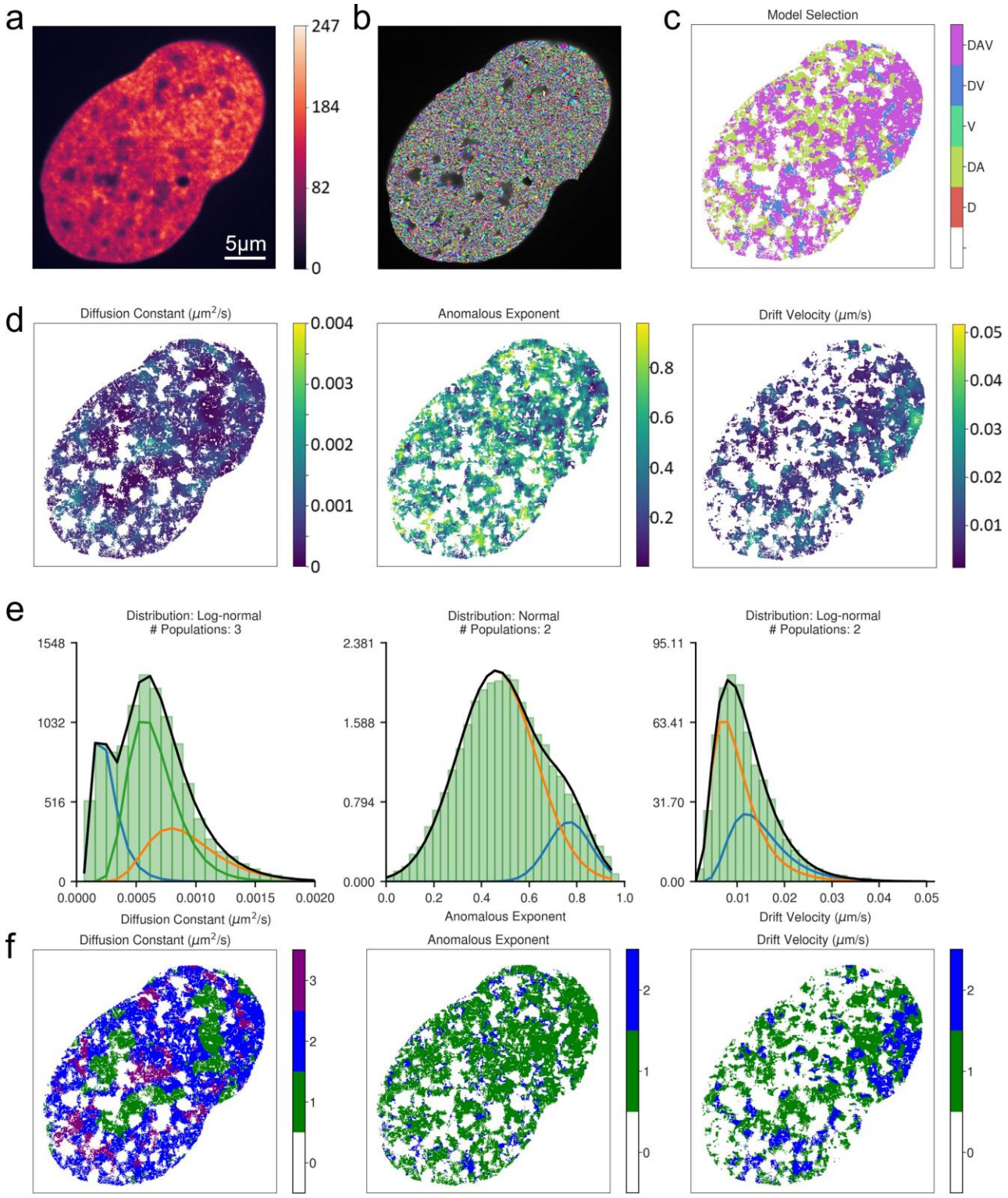


Figure 5 | A complete analysis example using the Hi-D-Py. The workflow was applied to a sequence of H2B (**a**) in U2OS. Trajectories are estimated (**b**). Bayesian inference mapped results for the models in(**c**) and biophysical parameters (**d**). Results from deconvolution by GMM. The histograms are shown in (**e**) and the maps of populations are shown in (**f**).

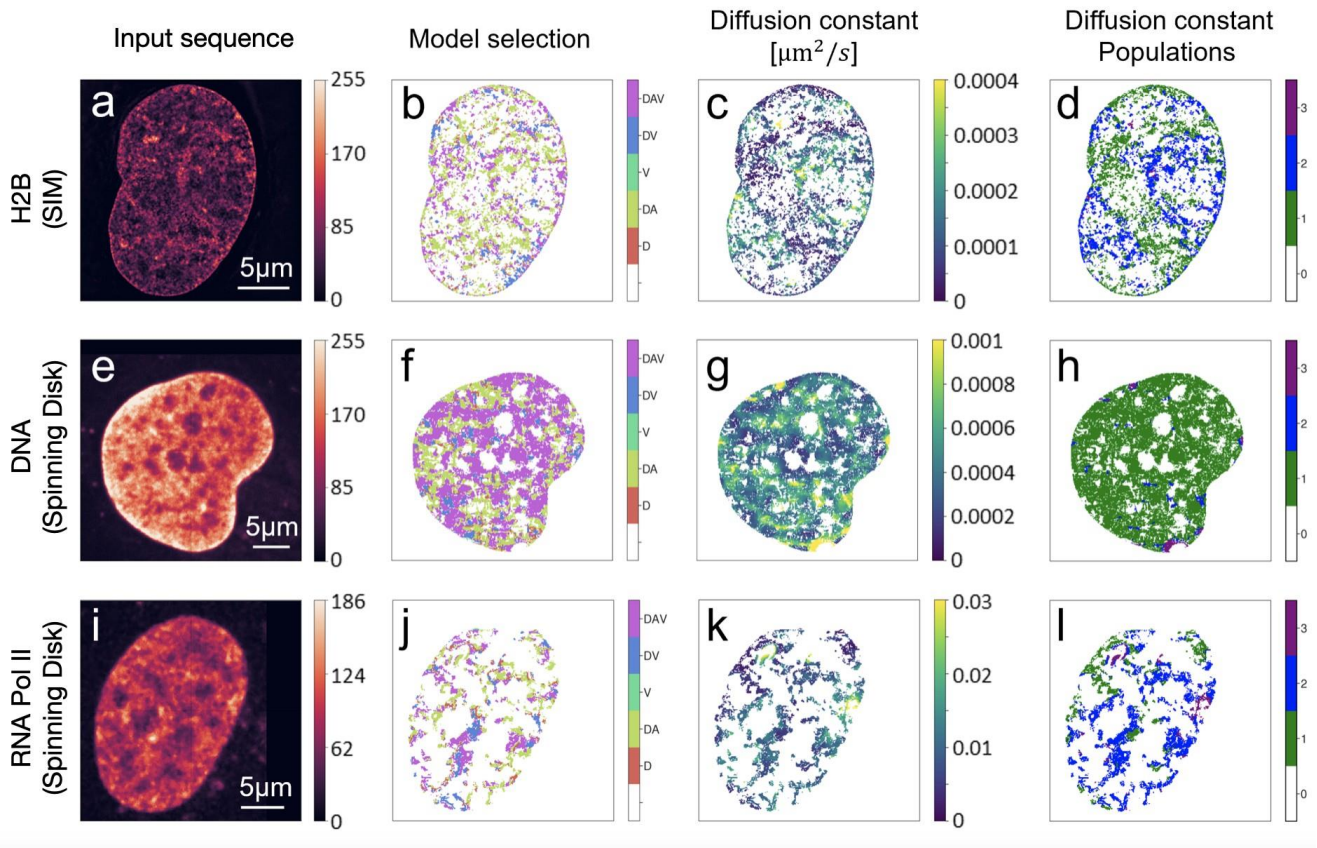


Figure 6 | Hi-D-Py was applied to analyze live cell imaging of chromatin and nuclear proteins using two microscopy modalities. (**a - d**) Live-nucleus imaging of Hela Cell expressing H2B-GFP imaged using SIM with Δt = 2s. Live cell imaging of SiR-DNA stained (**e - h**) and RNA Pol II (Rbp1-Dendra2) expression in U2Os cells **(i - l**) imaged using a spinning disk microscope with Δt = 0.2s. Representative maps are shown for each sample type: model selection (**b,f,j**), diffusion constant (**c,g,k**), and the estimated populations' map (**d,h,l**) from deconvolution by GMM are shown for the analyzed cells H2B, DNA, and RNA Pol II respectively.