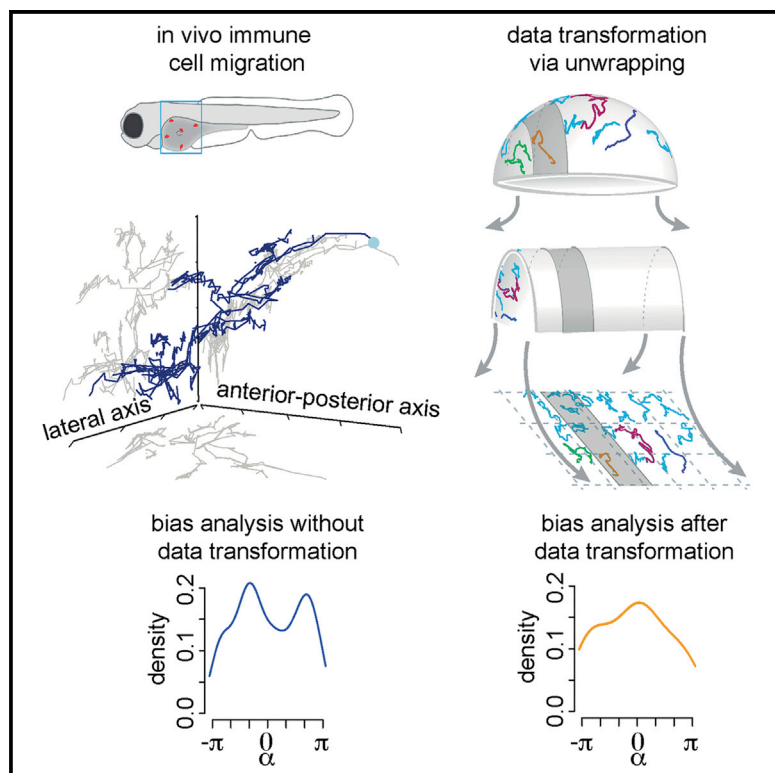


Cell Systems

Accurate Reconstruction of Cell and Particle Tracks from 3D Live Imaging Data

Graphical Abstract



Authors

Juliane Liepe, Aaron Sim,
Helen Weavers, Laura Ward,
Paul Martin, Michael P.H. Stumpf

Correspondence

m.stumpf@Imperial.ac.uk

In Brief

Liepe et al. present a set of tools that enable us to account for the spatial constraints acting on the motion of cells and particles in live imaging studies, and show how these allow us to accurately interpret these data.

Highlights

- Cell movement is often constrained, e.g., to surfaces of cellular structures
- We develop approaches to detect such constraints from in vivo live imaging data
- Accounting for these structures is necessary for correct analysis of cell tracks



Accurate Reconstruction of Cell and Particle Tracks from 3D Live Imaging Data

Juliane Liepe,^{1,2,6} Aaron Sim,^{1,2,6} Helen Weavers,³ Laura Ward,⁴ Paul Martin,^{3,4,5} and Michael P.H. Stumpf^{1,2,*}

¹Department of Life Sciences, Imperial College London, London, SW7 2AZ, UK

²Centre for Integrative Systems Biology and Bioinformatics, Imperial College London, SW72AZ, UK

³School of Biochemistry, Biomedical Sciences, University of Bristol, Bristol, BS8 1TD, UK

⁴School of Physiology, Pharmacology and Neuroscience, Biomedical Sciences, University of Bristol, Bristol, BS8 1TD, UK

⁵School of Medicine, University of Cardiff, BS8 1TD, UK

⁶Co-first author

*Correspondence: m.stumpf@imperial.ac.uk

<http://dx.doi.org/10.1016/j.cels.2016.06.002>

SUMMARY

Spatial structures often constrain the 3D movement of cells or particles *in vivo*, yet this information is obscured when microscopy data are analyzed using standard approaches. Here, we present methods, called unwrapping and Riemannian manifold learning, for mapping particle-tracking data along unseen and irregularly curved surfaces onto appropriate 2D representations. This is conceptually similar to the problem of reconstructing accurate geography from conventional Mercator maps, but our methods do not require prior knowledge of the environments' physical structure. Unwrapping and Riemannian manifold learning accurately recover the underlying 2D geometry from 3D imaging data without the need for fiducial marks. They outperform standard x-y projections, and unlike standard dimensionality reduction techniques, they also successfully detect both bias and persistence in cell migration modes. We demonstrate these features on simulated data and zebrafish and *Drosophila* *in vivo* immune cell trajectory datasets. Software packages that implement unwrapping and Riemannian manifold learning are provided.

INTRODUCTION

The ability to image the often complex behavior of biological systems is indispensable to much of modern biological research. Developments such as fluorescence, high-resolution, and live-imaging techniques are now firmly established technologies in cellular and molecular biology (Megason and Fraser, 2007). The major advances in imaging include the move from 2D to 3D data acquisition, the transition from static images toward time-lapse movies and the ability to image objects *in vivo* in living animals rather than *ex vivo* studies of smaller systems (Arranz et al., 2014; Weigert et al., 2013). The study of cell migration is one notable beneficiary of these methodological developments. Together with new statistical and computational

tools (Barbier de Reuille et al., 2015; Holmes et al., 2012; Jones et al., 2015), recent studies have already provided useful insights into many fundamental processes in immunology and developmental biology (Masopust and Schenkel, 2013; Phoon, 2006).

Movements captured in 3D are, however, rarely unconstrained 3D motions. They often take place in 1D (along e.g., blood vessels, microtubules, or actin filaments) or on 2D surfaces (e.g., curved cell walls or the interstitial medium in layered tissues such as the epithelium). Ignoring these structures during analysis can produce results that are skewed and erroneous (Figure 1). Even when acknowledged, these lower-dimensional spaces can be highly curved and irregularly shaped. For example when a cell or molecule moves along a curved surface (Figure 1E, top), standard 2D projections, including e.g., principal component analysis (PCA), can introduce curvature into its track where there is none (Figure 1E, bottom left) or artificially smooth a track (Figure 1E, bottom right).

It is therefore important to acknowledge underlying lower-dimensional structures when analyzing random walks; however, these underlying structures are rarely an ideal curved surface. These lower-dimensional spaces can be highly curved and irregularly shaped. In such cases, the commonly used linear dimensional reduction methods such as principal component analysis (PCA) are no longer appropriate for either data visualization or data analysis. Here, we present two methods for identifying a 2D coordinate representation of a given 3D point-cloud dataset that preserves the geometrical information of its hidden embedded surfaces. Both methods are non-linear generalizations of both linear projections onto pre-defined 2D planes and PCA (Jolliffe, 2013). As shown below for migrating cells on curved surfaces, the methods are able to detect the bias and persistence modes of biological random walks models. The first approach, which we refer to as “unwrapping,” is an intuitive two-step process that is particularly applicable to scenarios where the underlying 2D surfaces have relatively simple structures—specifically, convex surfaces with zero or small intrinsic curvatures (e.g., local patches on cylindrical or ellipsoid-like manifolds). This prior knowledge of the surface geometry allows the method to be effective even when the data are relatively sparse. The second method, “Riemannian manifold learning,” is an adaptation of an existing method in machine learning that, while somewhat

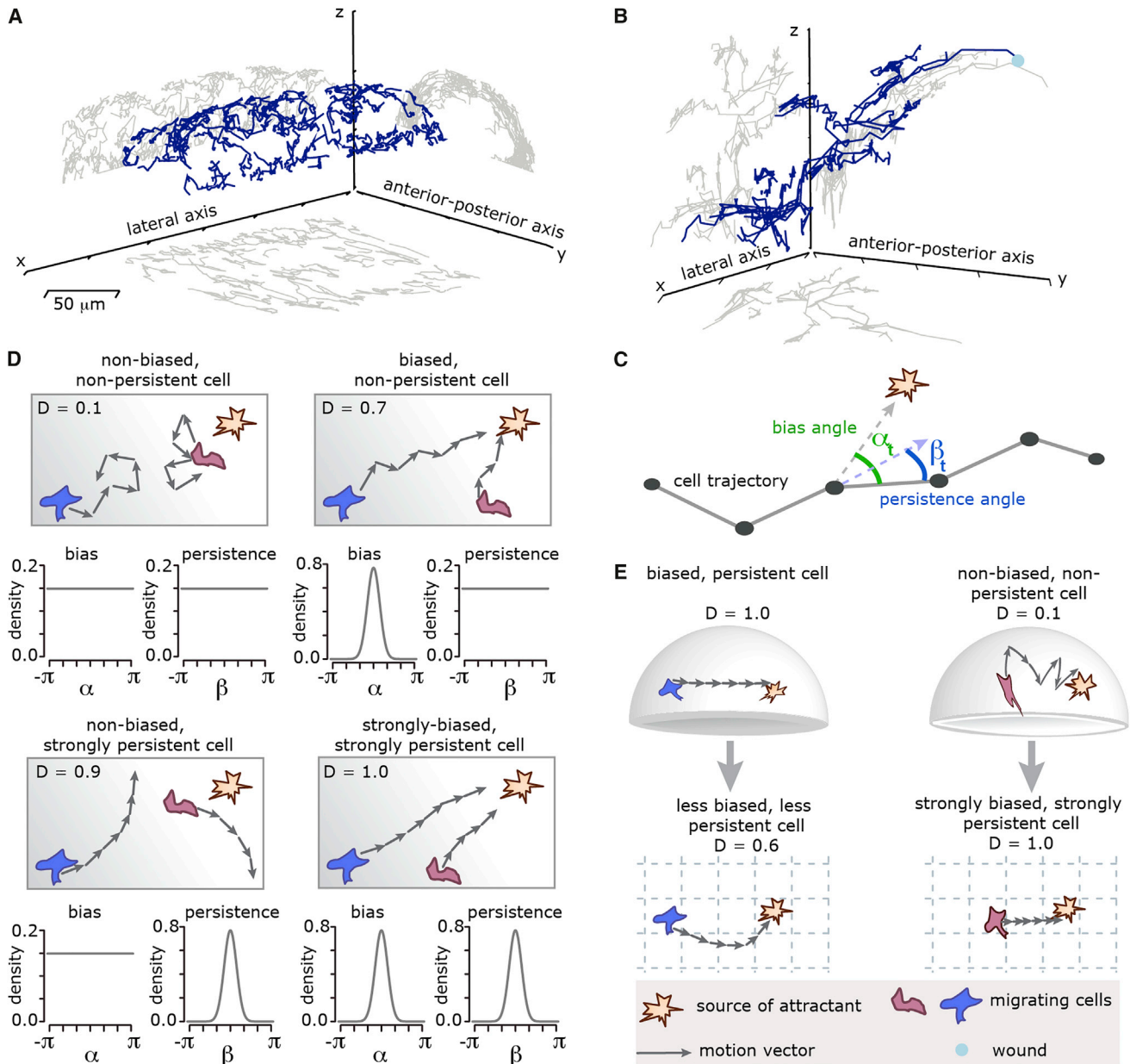


Figure 1. Directional Statistics of Cells Migrating on Curved Surfaces

(A) 3D representation of haemocyte cell tracks extracted from *Drosophila* embryo (blue) with the xy-, xz- and yz-projections (gray).

(B) 3D representation of neutrophil cell tracks extracted from laser wounded epidermis of the yolk syncytium of a zebrafish (blue) with the xy-, xz- and yz-projections (gray). Both, the datasets shown in (A and B) have a curvature, which is strong enough to induce analysis artifacts, but the same time weak enough to be analyzed using our proposed unwrapping method.

(C) From each cell trajectory the indicated bias and persistence angles are measured for each time step. The bias angle describes the angle between a motion vector (a step of the cell) and the direction pointing toward the attractant. The persistence angle describes the angle between two consecutive motion vectors. All the measured bias and persistence angles of each cell track built the bias and persistence distributions, from which the strength of bias and persistence can be estimated.

(D) Four types of random walks are sketched as cartoons, visualizing bias and persistence. For these random walk the expected bias and persistence distributions can be obtained mathematically and are here plotted as an example. The straightness index (D) is noted as a reference (see supplemental information for the definition of the straightness index).

(E) Artifacts that appear when random walks happen on curved surfaces but are analyzed in the 2D projections.

more abstract, can be applied to surfaces that are irregularly shaped, non-convex, or highly curved without prior knowledge of underlying structures. We provide a set of example code for

both methods in supplemental information, including pure R scripts as well as Jupyter notebooks coded in R and Python ([Data S1](#) and [Data S2](#)).

RESULTS

We provide a brief description of the two methods and demonstrate their application by extracting quantitative information from a set of simulated and experimental data of cell migration on complex surfaces.

Method 1: Unwrapping

In a standard xy projection, information about the cell's position (illustrated here along a convex, curved 2D surface) in the z dimension is simply ignored. By contrast, our unwrapping method maps the points on the curved 2D surface to appropriate coordinates by fitting a set of ellipses to a succession of data slices (indicated in gray in Figure 2B), and then “unrolls” these 1D strips onto straight lines. The resulting flattened representation of the data can then be analyzed using conventional tools for cell migration analysis. The method proceeds in two steps. First, unwrapping projects data that lie on part of a convex surface onto a cylindrical-like surface, i.e., flat in one direction (Figure 2B, see also Supplemental Information). In the second step, the process is repeated in the orthogonal dimension and the data are mapped from the intrinsically flat cylindrical surface onto a flat plane.

Method 2: Riemannian Manifold Learning

All 2D coordinate maps of a given curved surface will misrepresent the latter's geometry to some extent (consider, for example, the inflated sizes of countries near the polar regions in the Mercator projection of the world map). “Riemannian manifold learning” (or equivalently, metric manifold learning) is a method that allows one to quantify this misrepresentation for any given 2D coordinate map, and to then use this geometric information in any subsequent analysis of the data (Figure 2C). The method was first introduced as the LEARNMETRIC algorithm in [Perrault-Joncas and Meila \(2013\)](#) and is a straightforward adaptation of the non-linear dimensional reduction techniques common in the field of statistical machine learning.

The working principle behind this method is that the geometry of any embedded 2D surface is entirely encoded in a position-dependent 2×2 matrix known as the “metric tensor.” This metric can be consistently inferred, without using prior knowledge or making assumptions about the geometry, from the set of 3D data points and its corresponding set of 2D coordinate maps. In all the examples in this paper, we obtain the 2D coordinates using the locally linear embedding method. Nevertheless, as discussed in [Perrault-Joncas and Meila \(2013\)](#), the method is applicable to any other smooth, invertible map such as ISOMAP, Laplacian Eigenmaps, or even the unwrapping method introduced above. We have included a brief introduction of the relevant mathematical details in the supplemental information.

Given this 2D coordinate representation of the data, one then incorporates the geometrical information from the metric when calculating the usual statistics of interest that describe cell biological data, such as turning angles, step-lengths, and cell velocities. We note that despite being generally applicable to any open surface, this method requires more detailed input by the user than the unwrapping method. Specifically, this is the so-called bandwidth parameter intrinsic to manifold-learning algorithms; this is effectively the extent to which the neighborhood of a point

can be considered to be a flat surface (see Supplemental Information for details).

Unwrapping Recovers Random Walk Characteristics

To test and characterize our approach, we validate the two methods on a set of simulated (in silico) datasets, before applying it to data obtained by fluorescent time-lapse microscopy imaging.

To start, we simulated cell tracks based on a Brownian motion type (non-biased and non-persistent, as described in Figures 1C and 1D) random walk model on several surfaces of varying curvatures, ranging from a thinly-stretched ellipsoid to a sphere (details of the random walk models are described in the Supplemental Experimental Procedures). This type of random walk necessarily produces flat angular distribution (Figure 2D, “true distribution”), which we compare to the computed angular distributions based on the simple the xy-projection, the unwrapping method, and the manifold learning method (with and without incorporating geometrical information) (Figure 2D). We observe the largest deviation from the true angular distribution for the oft-employed xy-projection, highlighting the need for data transformation methods, especially for estimates of the bias distribution on more highly curved surfaces. Both the unwrapping and the metric manifold learning methods manage to recover the true distribution with only small deviations. The more commonly employed manifold learning approach that omits the metric (i.e., “Euclidian” manifold learning) performs significantly worse than the metric manifold learning method, especially on very narrow ellipsoids.

To demonstrate that unwrapping and metric manifold learning methods are generalizable, we tested them on in total six different geometries and simulated data obtained from three different random walk models (Figures S1A–S1C, related to Figure 2). The unwrapping method recovers all bias angle distributions and shows improvements for the persistence angle distributions compared to conventional xy-projections. The performance of the metric manifold learning method is slightly better still than the unwrapping method. To quantify the performance of the different methods, we computed the deviation distance of the angle distributions obtained through each of the methods from the true angle distributions (Figure S2, related to Figure 2). The unwrapping method and the metric manifold learning method perform better than the simple xy-projection on all tested surfaces. This analysis demonstrates that, in principle, both the unwrapping and the metric manifold learning algorithm are well-suited methods for the analysis of cell migration on curved surfaces. Given suitable high-resolution data they can also be applied to study intra-cellular movement of e.g., proteins on cellular structures such as the endoplasmic reticulum or the mitochondria.

Unwrapping Detects Biased-Persistent Immune Cell Migration

Next, we analyzed bias and persistence in the migratory behavior of immune cells in vivo. Specifically, we observed haemocyte migration in the embryo of the fruit fly *Drosophila* and neutrophil migration in the epidermis overlying the yolk syncytium of a zebrafish in response to wounding. We extracted the data from 3D time-lapse fluorescent movies and track the cells over time

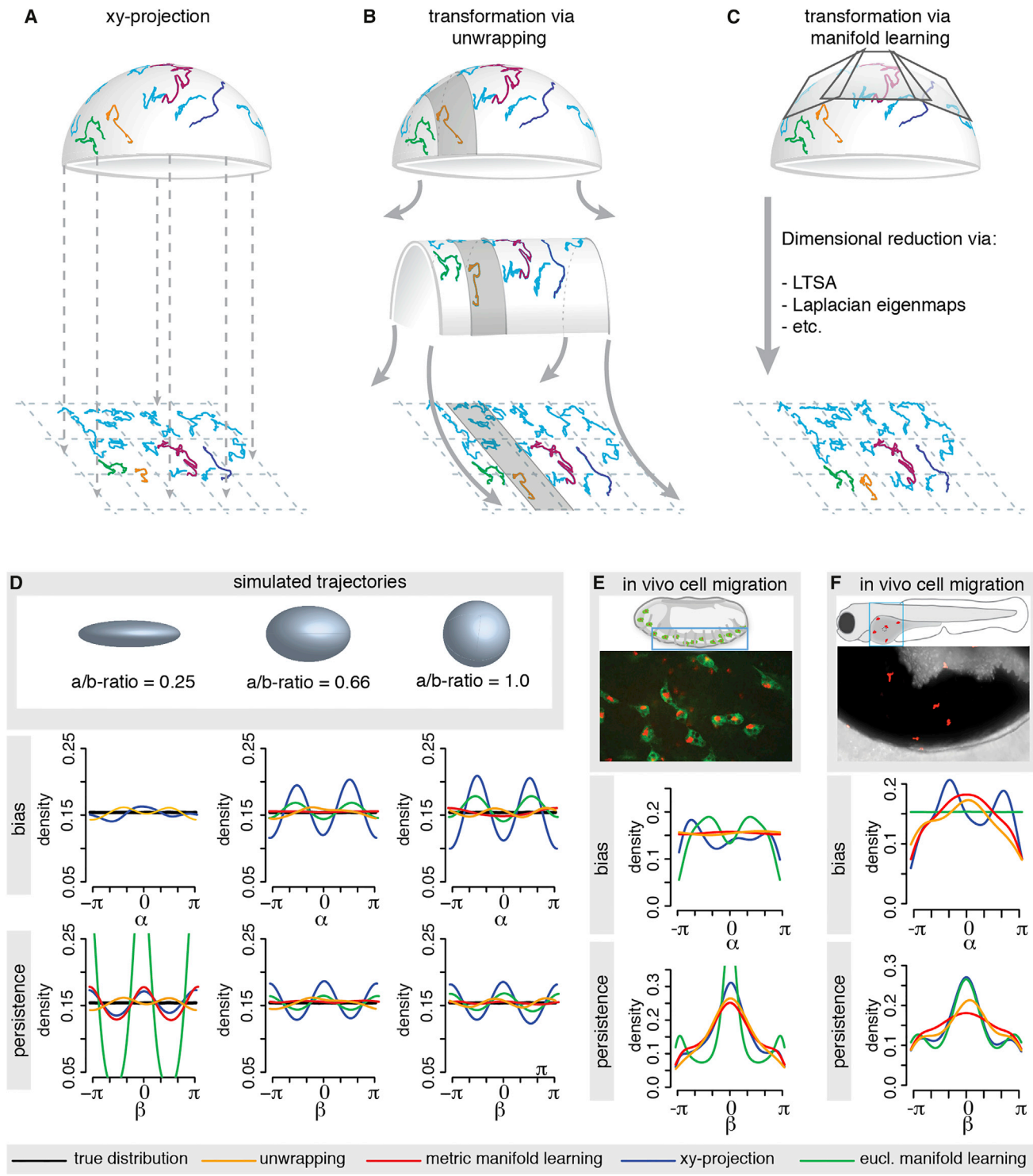


Figure 2. Methods for Manifold Learning and Applications

(A–C) Shown are example trajectories on a hemi-sphere and their transformation via one of the discussed methods. The 3D cell tracks are simply projected onto the xy-plane (A). The 3D cell tracks are transformed via unwrapping (B) or via well-known manifold learning methods (e.g., LTSA) (C).

(D) Random walk trajectories (in absence of any bias or persistence) are simulated on the displayed curved surfaces and then transformed with xy-projection, unwrapping, Euclidian manifold learning and Riemannian, or metric, manifold learning, respectively. The resulting bias and persistence distributions are compared with the respective true distribution (black), which are uniform for this random walk model. For the ellipsoid with the most extreme aspect ratio (i.e., the “thinnest” shape), the manifold learning approach was unsuccessful as it incorrectly interpreted the data as belonging to a 1D line. In this case, there was insufficient data to correctly reveal the spatial extent of one dimension.

(legend continued on next page)

(for details see [Supplemental Information](#)). The haemocytes migrated in a constrained, pseudo-2D region beneath the surface of the embryo and did not enter deeper tissue layers at this developmental stage ([Figure 1A](#), [Movie S1](#), and [Movie S3](#)). This is consistent with previous observations, which report that haemocytes have no spatial bias toward any particular point and move in non-biased, non-persistent manner ([Davis et al., 2012](#)). Accordingly, we observe a non-uniform persistence distribution when haemocyte motion is analyzed with the unwrapping or metric manifold learning methods ([Figure 2E](#)). Analysis of the xy-projection results in artifacts, which indicates a bias toward an arbitrary point and overestimates the strength of the persistence. Additional transformations of the data highlight the deviations between the unwrapping method, the metric manifold learning method, and the xy-projection ([Figure S3](#), related to [Figure 2](#)).

We also analyzed the response of neutrophils to a wound ([Figure S2F](#), related to [Figure 2](#)). From previous studies ([Holmes et al., 2012](#); [Taylor et al., 2013](#)), we know that neutrophils constitute the first line of defense and directly migrate persistently toward wounds, i.e., they show biased persistent motion. The cells in this example migrate on a curved surface constrained by the epidermis overlying the yolk syncytium ([Figure 1B](#), [Movie S2](#), and [Movie S4](#)). Unwrapping the data and analyzing the resulting distributions shows a clear bias of the neutrophils toward the wound with some level of persistence, both of which are confirmed by the metric manifold learning method. By contrast, analysis of the xy-projection results in strong artifacts for bias and persistence, missing the bias entirely, which is not biologically reasonable given the neutrophils' function. Further, when we use PCA to reduce the dimensionality, as is common in many applications where one would like to visualize a 2D representation of a higher-dimensional dataset, the resulting angular distributions show even more pronounced artifacts than the simple xy-projection ([Figures S1F](#) and [S1G](#)). These results highlight the need for manifold learning techniques that go beyond simple linear projections; but they also show that our rather intuitive and data-driven unwrapping approach can provide an adequate representation of the experimental data.

DISCUSSION

It is well-known that image processing techniques can introduce artifacts into cell migration analysis ([Beltman et al., 2009](#)). The problem highlighted and tackled here is more closely related to finding the right representation of data (and has obvious parallels with cartographic projections). In the two examples of in vivo data analysis, we have shown that an appropriate metric manifold learning method is required to detect a well-established biological behavior: the bias of the zebrafish neutrophils toward a wound. Without these methods, we would have wrongly concluded that *Drosophila* haemocytes migrate with a bias in absence of an obvious attractant source based only on the

xy-projection. Erroneous or incorrect analysis has implications beyond wrong conclusions ([Sim et al., 2015](#)): for example, poor analysis can render valuable patient and animal data useless.

Given the importance of constrained cellular and molecular movement throughout cellular and developmental biology and medicine, unwrapping and metric manifold learning methods will be broadly applicable. The development of quasi-conformal mapping methods ([Appleboim et al., 2006](#)) has been driven largely by the needs of the medical imaging community for 2D image representations of human organs with minimal geometric distortion ([Schwartz and Merker, 1986](#)). These methods, however, presume the ability to either capture a high-resolution image of the surface or to construct a triangulated mesh covering. In many contemporary biological applications these surfaces are rarely imaged directly with their existence only inferred indirectly from the migration tracks of the imaged objects. Obtaining an image of the surface would often require additional in vivo staining or generation of suitable tissue markers, both of which carries the risk of interfering with the image acquisition of the actual target cells. Both unwrapping and metric manifold learning relieve this need, as they require no fiducial marks or characterized in situ spatial constraints.

EXPERIMENTAL PROCEDURES

Data Acquisition

Drosophila were maintained on cornmeal agar fly food, supplemented with dried yeast, and handled according to standard protocols ([Greenspan, 2004](#)). Stage 15 embryos were collected from overnight apple juice plates at 25°C (ubi-EcadherinGFP, serpent-Gal4 > UAS-GFP;UAS-redstinger), carefully dechorionated in 50% bleach, washed thoroughly with distilled water and mounted on a glass slide in a drop of 10S voltalef oil (VWR). Movies were collected at 30 s/frame on a PerkinElmer UltraView spinning disc microscope using a ×40 oil immersion lens.

5-days-post-fertilization Tg(Lyz:dsRed)nz zebrafish larvae ([Hall et al., 2007](#)) were mounted laterally in 1.5% low-melting agarose (Sigma) in a glass-bottomed petri dish containing Danieau's solution and 0.01 mg/ml MS-222 (Sigma). The epidermis overlying the yolk syncytium was wounded using a UV-nitrogen laser (Coumarin 440 nm dye cell) coupled to a Zeiss Axioplan 2 microscope (Micropoint Laser System, Photonic Instruments) with a 40× water immersion objective. Movies were collected at 1 min/frame using a Leica SP5-II AOBS confocal laser scanning microscope attached to a Leica DM I6000 inverted microscope with a ×20 glycerol lens.

Further methods and any associated references are available in the [Supplemental Information](#).

SUPPLEMENTAL INFORMATION

Supplemental Information includes Supplemental Experimental Procedures, three figures, four movies, and two data sets and can be found with this article online at <http://dx.doi.org/10.1016/j.cels.2016.06.002>.

ACKNOWLEDGMENTS

The project was in part granted by National Centre for the Replacement Refinement and Reduction of Animals in Research (NC3Rs) through a David

(E) Application of the unwrapping method and manifold learning methods to haemocyte cell tracks extracted from a *D. melanogaster* embryo and their comparison to the xy-projection. Shown is a schematic of the embryo and a snapshot from the video microscopy imaging. Haemocytes (green) were tracked via their nucleus (red).

(F) Application of the unwrapping method and manifold learning methods to neutrophil cell tracks extracted from the epidermis overlying the yolk syncytium of a zebrafish and their comparison to the xy-projection. The epidermis was wounded with a laser before image acquisition. Shown is a schematic of the zebrafish with the imaged area and a snapshot from the video microscopy imaging with the neutrophils in red.

Sainsbury Fellowship to J.L., by the BBSRC, The Leverhulme Trust and the Royal Society through a Wolfson Research Merit Award to M.P.H.S. A.S. is supported by the Human Frontiers Science Program project grant RGP0043/2013. L.W. and P.M. are supported by BBSRC and Cancer Research UK programme grants. P.M. is furthermore supported by a Wellcome Trust Senior Investigator Award. H.W. is funded by M.R.C. We acknowledge technical support from the Wolfson Bioimaging facility at the University of Bristol.

Received: September 9, 2015

Revised: January 28, 2016

Accepted: June 3, 2016

Published: July 21, 2016

REFERENCES

- Appleboim, E., Saucan, E., Zeevi, Y.Y., and Zeitoun, O. (2006). Quasi-isometric and Quasi-conformal Development of Triangulated Surfaces for Computerized Tomography, in: *Combinatorial Image Analysis, Lecture Notes in Computer Science* (Springer), pp. 361–374.
- Arranz, A., Dong, D., Zhu, S., Savakis, C., Tian, J., and Ripoll, J. (2014). In-vivo optical tomography of small scattering specimens: time-lapse 3D imaging of the head eversion process in *Drosophila melanogaster*. *Sci. Rep.* *4*, 7325.
- Barbier de Reuille, P., Routier-Kierzkowska, A.-L., Kierzkowski, D., Bassel, G.W., Schüpbach, T., Tauriello, G., Bajpai, N., Strauss, S., Weber, A., Kiss, A., et al. (2015). MorphoGraphX: A platform for quantifying morphogenesis in 4D. *eLife* *4*, 05864.
- Beltman, J.B., Marée, A.F.M., and de Boer, R.J. (2009). Analysing immune cell migration. *Nat. Rev. Immunol.* *9*, 789–798.
- Davis, J.R., Huang, C.-Y., Zanet, J., Harrison, S., Rosten, E., Cox, S., Soong, D.Y., Dunn, G.A., and Stramer, B.M. (2012). Emergence of embryonic pattern through contact inhibition of locomotion. *Development* *139*, 4555–4560.
- Greenspan, R.J. (2004). *Fly Pushing* (CSHL Press).
- Hall, C., Flores, M.V., Storm, T., Crosier, K., and Crosier, P. (2007). The zebrafish lysozyme C promoter drives myeloid-specific expression in transgenic fish. *BMC Dev. Biol.* *7*, 42.
- Holmes, G.R., Dixon, G., Anderson, S.R., Reyes-Aldasoro, C.C., Elks, P.M., Billings, S.A., Whyte, M.K.B., Kadirkamanathan, V., and Renshaw, S.A. (2012). Drift-Diffusion Analysis of Neutrophil Migration during Inflammation Resolution in a Zebrafish Model. *Adv. Hematol.* *2012*, 792163–792168.
- Jolliffe, I.T. (2013). *Principal Component Analysis* (Springer Science & Business Media).
- Jones, P.J.M., Sim, A., Taylor, H.B., Bugeon, L., Dallman, M.J., Pereira, B., Stumpf, M.P.H., and Liepe, J. (2015). Inference of random walk models to describe leukocyte migration. *Phys. Biol.* *12*, 066001.
- Masopust, D., and Schenkel, J.M. (2013). The integration of T cell migration, differentiation and function. *Nat. Rev. Immunol.* *13*, 309–320.
- Megason, S.G., and Fraser, S.E. (2007). Imaging in systems biology. *Cell* *130*, 784–795.
- Perrault-Joncas, D., Meila, M., 2013. Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. *arXiv*.
- Phoon, C.K.L. (2006). Imaging tools for the developmental biologist: ultrasound biomicroscopy of mouse embryonic development. *Pediatr. Res.* *60*, 14–21.
- Schwartz, E., and Merker, B. (1986). Computer-Aided Neuroanatomy: Differential Geometry of Cortical Surfaces and an Optimal Flattening Algorithm. *IEEE Comput. Graph. Appl.* *6*, 36–44.
- Sim, A., Liepe, J., and Stumpf, M.P.H. (2015). Goldstein-Kac telegraph processes with random speeds: Path probabilities, likelihoods, and reported Lévy flights. *Phys. Rev. E.* *91*, 042115.
- Taylor, H.B., Liepe, J., Barthen, C., Bugeon, L., Huvet, M., Kirk, P.D.W., Brown, S.B., Lamb, J.R., Stumpf, M.P.H., and Dallman, M.J. (2013). P38 and JNK have opposing effects on persistence of in vivo leukocyte migration in zebrafish. *Immunol. Cell Biol.* *91*, 60–69.
- Weigert, R., Porat-Shliom, N., and Amornphimoltham, P. (2013). Imaging cell biology in live animals: ready for prime time. *J. Cell Biol.* *201*, 969–979.

Cell Systems, Volume 3

Supplemental Information

**Accurate Reconstruction of Cell
and Particle Tracks from 3D Live Imaging Data**

Juliane Liepe, Aaron Sim, Helen Weavers, Laura Ward, Paul Martin, and Michael P.H. Stumpf

Accurate reconstruction of cell and particle tracks from 3D live imaging data

Juliane Liepe^{*1;2}, Aaron Sim^{*1;2}, Helen Weavers³, Laura Ward⁴, Paul Martin^{3;4;5}, Michael PH Stumpf^{1;2}

¹Department of Life Sciences, Imperial College London, London, UK, SW7 2AZ

²Centre for Integrative Systems Biology and Bioinformatics, Imperial College London, UK, SW72AZ

³Department of Biochemistry, Medical Sciences, University of Bristol, Bristol, UK, BS8 1TD

⁴School of Physiology and Pharmacology, University of Bristol, UK, BS8 1TD

⁵School of Medicine, University of Cardiff, UK, BS8 1TD

* These authors equally contributed to the work.

1. Supplemental Experimental Procedures

1.1. Tutorial: How to unwrap cell trajectory data using Jupyter

In this section we provide a brief example how to unwrap trajectory data that lie on a curved surface. The required code is provided in Suppl. Materials as an html file as well as a Jupyter notebook. You can follow step by step the next sections in parallel with the Jupyter notebook.

Prerequisites.

The code is written in R. To run the Jupyter notebook you need: (i) an installation of the Jupyter notebook (<http://Jupyter.org/>), (ii) an installation of the R statistical environment (<http://www.r-project.org/>) and (iii) the R Kernel for the Jupyter notebook, which can be installed from <https://github.com/IRkernel/IRkernel>. For the latter the instructions found at <http://www.michaelpacer.com/maths/r-kernel-for-ipython-notebook> are good for installing this under OSX. Visualizing the output in R requires the *rgl* package, which can be installed using your

R environment. Providing that these packages are in place the code in this notebook should run without any further problems.

Installation.

The Jupyter notebook (previously *IPython notebook*) requires a working installation of Python in the first place; most Python distributions aimed at scientific computing contain the relevant files and packages. The *Anaconda* (<https://www.continuum.io/downloads>) is, in our experience, particularly straightforward to install, use, and maintain. Installation can be done via the provided installers (for Windows, OSX and Linux), or from the command line (the website <https://www.continuum.io/downloads> contains instructions for the various versions).

Maintaining and upgrading the distribution's packages is done using the *conda* package-manager. To upgrade the jupyter notebook, for example, at the command line write

```
> conda upgrade jupyter
```

This installs all the files required for using *Jupyter* in conjunction with Python. Other kernels can be installed as described on the relevant webpages, which are linked to at <https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages>. For the R kernel the *conda* distribution offers a convenient way of installing the relevant packages (assuming that a recent R installation is present),

```
> conda install -c r r-essentials
```

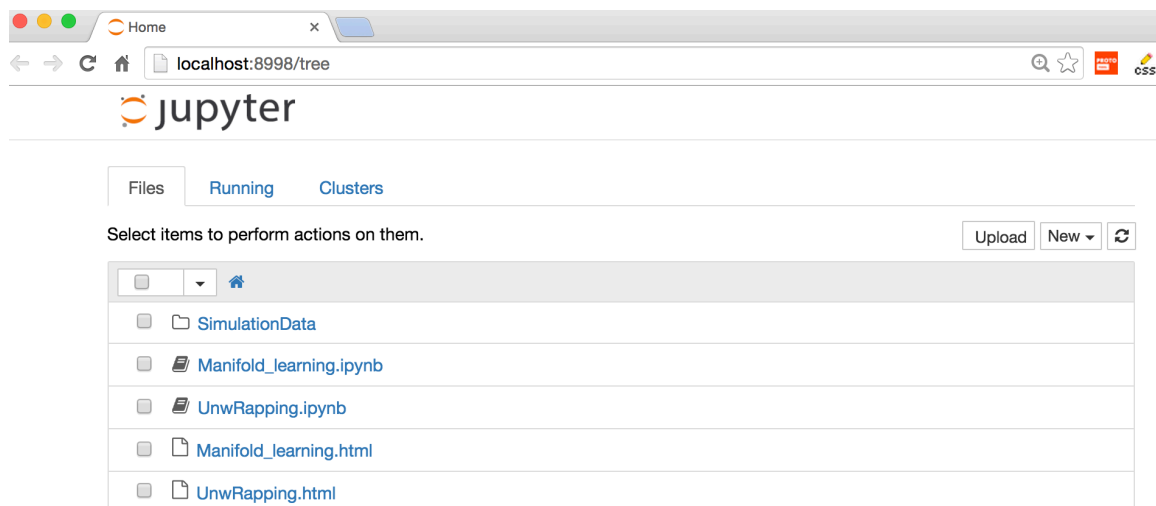
(see <https://www.continuum.io/blog/developer/jupyter-and-conda-r> for further details).

How to execute the Jupyter notebook.

The Jupyter notebooks are available in the folder *Jupyter*. A step-by-step guide is presented as a web page called *Unwrapping.html (Data S1)*. To execute the notebook, at the command line enter (in the Jupyter folder Data S1)

```
> jupyter notebook
```

This will start the default browser with and the loads the contents of the directory,



Clicking on the relevant Jupyter notebook (the files with an extension “.ipynb”) will then start the relevant Jupyter notebook.

Running *Manifold_learning.ipynb* will be straightforward with any recent Python installation; running the *Unwrapping.ipynb* notebook will require the installation of the R kernel.

Routines for Unwrapping Data. We first define a set of necessary routines. *getAngleBias()* lets us define the angle to the wound (target). *getAnglePersistence()* determines the angle between successive steps and hence measures the persistence. *transformData()* and *unwrapData3D()* are the routines that transform the data and unwrap them into a flat space. The code is provided in the Jupyter notebook.

Data preparation. We provide an example data set in Suppl. Materials (*exampleDataBPRW.csv*) in the folder *SimulatedData*. This data set describes simulated cell trajectories based on a biased persistent random walk on an ellipsoid surface. The data are saved as *csv* format, which can be opened in any text editor or Excel. The data file has to be provided in a specific layout: It should contain 4 columns, where the first column is the cell track ID (*id*), and the second, third and fourth columns are the *x*-, *y*- and *z*-coordinates of the cell tracks, respectively. The rows are then the individual time points for all cell tracks. The user can replace this data file with own data files. We use the same data format for all methods provided.

The target of the biased cells (wound) is at position (-7; 0; 0), which needs to be defined in the first step. Next the data are imported and reformatted (point 6 in the notebook).

Plot Data in 3D. We begin by plotting the trajectory data in 3D (using *rgl*). This allows us to get an idea of the geometry of the data (see notebook point 6). We can then clearly see that the trajectory data lie on a surface of an ellipsoid with radii 6; 6 and 18.

Analysis of random walk statistics in the 2D projection. Before unwrapping the data we calculate the bias and persistence of the random walk data in the conventional projection down to 2D, *i.e.* using the x - and y -coordinates only. We call the routines `getAngleBias()` and `getAnglePersistence()` by passing the relevant coordinates.

Unwrapping of the data onto a flat manifold. We next transform the data onto a flat space using the unwrapping method by calling the routine `unwrapData()`. If a simple representation of the manifold is available (such as a cylinder or ellipsoid) then we can unwrap the data by mapping the correct positions on the manifold (akin to cartographic projections) in a way that maintains the angles correctly. The routine will create a 3D graphical presentation of the progress of the unwrapping procedure. Firstly, the original data are plotted in 3D and then shifted to be suitable for the unwrapping using the routine `transformData()`. The original data are then clustered along the x -axis and plotted in 3D, where each cluster is shown in a different color. Next, an ellipse is fitted to each cluster. The data are then unrolled onto a new space based on the characteristics of the fitted ellipse (for exact details see section 4). This first step is an approximation to manifold learning techniques. In the following the data obtained from the first step are further unrolled by fitting an ellipse onto a flat space. The routine will plot the transformed data in grey.

Analysis of random walk statistics in the manifold projection. We can now calculate the bias and persistence of the random walk data in the 2D manifold projection. Again, this is done via the routines `getAngleBias()` and `getAnglePersistence()` by passing the relevant coordinates.

Comparison of inferred biased and persistence behavior in the 2D x - y and unwrapped (manifold) projection. Finally we can compare the computed statistics for the xy -projection with the statistics computed from the unwrapped data. This is for example done via plotting the histograms and densities for the bias and persistent distributions. Doing so, we observe strong artifacts in the obtained distributions based on the xy -projection. On the contrary, unwrapping manages to recover the expected bias and persistence distributions.

1.2. Tutorial: How to unwrap cell trajectory data using R

Additionally to the Jupyter notebook we also provide the plain R code for unwrapping data that lie on a curved surface. The equivalent routine for the above-described example of a biased-persistent random walk can be found in the folder `exampleCode_Unwrapping_1_BPRW` (part of Data S2). Furthermore we provide the same routine for a purely persistent random walk (without bias) in the folder `exampleCode_Unwrapping_2_PRW` (part of Data S2) and for an *in vivo* data set extracted from a fly embryo in the folder `exampleCode_Unwrapping_3_inVivo` (part of Data S2). In the latter data set the fly was wounded with a laser. In this example we find that without unwrapping it is possible to detect a weak bias towards the wound. However, after unwrapping the data it becomes apparent that the cells are strongly biased towards the wound but also into the opposite direction, *i.e.* away from the wound.

Installation and Prerequisites.

You need to install the R statistical environment (<http://www.r-project.org/>). You can download the precompiled binary file for installation for most computer platforms (<http://cran.ma.imperial.ac.uk/>). Simply download the binary file suitable for your platform, double click it to start installation. More advanced user might chose to install R from source code.

For visualization purpose you need the R library 'rgl'. Again, you can download the binary file for installation from <https://cran.r-project.org/web/packages/rgl/index.html>. Alternatively, open a terminal, start R by typing

R
followed by enter. Then type

```
install.packages('rgl')
```

which will also initiate the installation of the library.

How to use the R scripts.

First of all open a terminal. On most Macs you can find the terminal in the folder 'Applications/Utilities'. When the terminal is started, it usually links to your home directory. Type 'pwd' to know in which directory you currently are. Change the directory to one of the 3 example code folders by typing in the terminal for example:

```
cd WorkFolder/exampleCode_Unwrapping/ exampleCode_Unwrapping_1_BPRW
```

If you are new to terminal and the related commands, please refer to <http://ss64.com/osx/>

Then start R by typing in the terminal

```
R
```

followed by enter.

To run the example script type

```
source("runAnalysis.r")
```

The script will first read in the data 'exampleDataBPRW.csv' located in the folder 'simulatedData'. The data are saved as csv format, which can be opened in any text editor or Excel. The data file has to be provided in a specific layout: It should contain 4 columns, where the first column is the cell track ID (id), and the second, third and fourth columns are the x-, y- and z-coordinates of the cell tracks, respectively. The rows are then the individual time points for all cell tracks. The user can replace this data file with own data files.

After reading the data, a window pops-up, which shows the data plotted in 3D. The data are then transformed via unwrapping. In the same pop-up window the procedure of the algorithm can be followed, i.e. the data are shifted, grouped, unwrapped in the first dimension (rainbow colored, still curved surface), followed by unwrapping in the second dimension resulting in transformed data points that lie in on a flat 2D surface (grey plotted points).

After the data transformation took place, the R script analyses the initial data transformed via xy-projection as well as the data transformed via unwrapping. For both data sets the bias and persistence angles are computed and plotted as histograms, overlaid with the estimated density of the resulting distributions (red lines).

If you run the *in vivo* example in the folder 'exampleCode_Unwrapping_3_inVivo' an additional pop-up window will open, where the trajectories are plotted in the xy-projection and after unwrapping for comparison. In this example the analysis output of bias and persistence distribution differs slightly. Here, we plot additionally to the bias and persistence distributions, the transformed bias distributions. As explained further below, the bias angles can take values between $-\pi$ and π . In principle, this distribution should be plotted on a circle, because angles generate circular distributions. We refrain from doing so, since the circular representation is harder to read. However, one should keep in mind that $-\pi$ is equivalent to $+\pi$. We aim to highlight this by shifting our obtained bias distribution by $-\pi$. In this way it becomes clear that the bias distribution obtained from the unwrapped data indicates two maxima (one at 0 and one at $-\pi$), which shows that cells are biased towards the wound (0), but also in the opposite direction, i.e. away from the wound ($-\pi$).

Finally, we provide for comparison the Jupyter notebook for the presented manifold learning technique on the example of the persistent random walk based on the same data set as the R routine for unwrapping *exampleCode_Unwrapping_2_PRW* (see previous section).

1.3. Tutorial: How to do manifold learning with Jupyter

Installation and Prerequisites.

The method is implemented in Python and makes use of the following packages: *NumPy*, *Pandas*, *scikit-learn*, *Matplotlib* and *Seaborn*

The Jupyter notebook can be installed following the instructions in the unwrapping examples above, except there is no need for the *R* kernels here. Note that the code can be executed in both Python 2.7 and 3.x.

The Jupyter notebook document is titled *Manifold_learning.ipnb* (part of Data S1) and can be found in the folder *Jupyter (Data S1)*. A detailed step-by-step installation guide is included within. For reference, an HTML version of the document *Manifold_learning.html*, which can be accessed using any web browser, is also provided in the same folder.

1.4. Timing

All provided methods are able to handle large data sets. The provided examples run in a couple of seconds. The Unwrapping was tested on a dataset with 3000 data points. This takes depending on the computer used (here: Mac OS 10.8, 2.7 Ghz Intel Core i7, 16 GB Memory) several seconds. The manifold learning methods are slightly slower, again depending on data set size and computer used. The provided example in the Jupyter notebook analyses a dataset with a broadly realistic size of 3000 data points. The example can be run under a minute on a standard workstation. The manifold learning computation is dominated by the calculation of the similarity matrix, which in turn scales as $O(N^2)$ where N is the number of data points. Therefore a dataset with 20-25K data points could be analyzed within approximately one hour. In practice, the limitations are defined by the computer equipment (e.g. processor, memory, disk space).

1.5. Methods in Brief

1.5.1. Random walks in Biology

There are different types of random walks that are commonly described in Biology. We can classify them into random walks that describe the step length distribution and random walks that describe the angular distributions. The definition of random walks via step length distribution is somewhat more frequently used. However, to investigate if a cell or a molecule is targeted in its movement, it is easier to look at angular distributions.

The most prominent random walk is **Brownian motion**. The angular distribution is isotropic, meaning that at each step a cell or molecule has equal probability to move in any direction. If we measure the angles between a motion vector (cell step) and a reference direction, we will find that the resulting angular distribution is flat (uniformly distributed). If, on the contrary, a cell has a specific target direction, then the cell has higher probability to move towards that target direction compared to all remaining directions. In this case we speak about a **biased random walk**. The expected angular distribution will have a peak at the angle which points towards the target direction. The remaining characteristics of the angular distribution of such biased random walk depend on the details of the exhibited walk, which are usually unknown. However, a commonly used description of the angular distribution is a wrapped normal distribution (a normal distribution wrapped around a circle to describe circular variables such as angles). The mean of the wrapped normal distribution indicates the bias direction and the variance indicates the strength of the bias. The lower the variance, the narrower is the distribution and the stronger is the exhibited bias. A further type of random walk frequently used to describe animal movement and cell migration is a **persistent random walk**. A cell exhibiting this type of walk has higher probability of moving in the same direction as in the previous step compared to changing its

direction. If we measure the angles between consecutive motion vectors (consecutive cell steps) we will observe a peak at 0, i.e. no change of direction. As for the biased random walk, the persistent random walk can also be described using a wrapped normal distribution with 0 mean and a variance which indicates the strength of the persistence (the lower the variance the stronger the persistence).

All three types of walks have been described for migration of immune and other cells, migration of animals, and movement of molecules inside the cell. Often a mix of these three types is observed.

1.5.2. Analyzing cell migration data

Cell migration trajectories are often extracted from confocal time-lapse microscopy imaging data. Recent advances allow researchers to collect such data even *in vivo* in living animals. Examples include imaging of macrophage, neutrophils and cancer cell migration in zebrafish tail fin, flanks, gills or yolk; imaging of stem cells and hematopoietic cells in mouse bone marrow; imaging of haemocytes in various stages and organs of drosophila; imaging of migrating neutrophils on the surface of the heart and many more.

While these data contain potentially a huge amount of new information about the underlying biological processes *in vivo*, their correct analysis buries a vast range of challenges and one of them we highlighted in this study: the movement of cells on curved surfaces.

In order to extract information about bias and persistence from observed cell trajectories, we have to compute two types of angles: (i) the angle α between a fixed reference direction and the cell motion vector and (ii) the angle β between two consecutive motion vectors. While the first angle α helps us to detect potential bias direction, the later angle β helps us to measure the strength of persistence (as described in the previous section). If the movement of the cell is restricted to a curved surface, then directly measuring the angles α and β based on the original (untransformed) data will provide us with artifacts, which in some cases could mimic a target bias where in reality there is none. In order to still be able to extract bias and persistence information from such data, we need to either transform the trajectory data in such way that they lie on a flat surface (and then apply the standard analysis tools), or use some methods to learn the exact surface (manifold) and compute the angles on such manifold. Either way, the aim is to remove any artifacts that appear through curved surfaces from the analysis. The first solution can be obtained via unwrapping; the second brings us to the field of manifold learning.

1.5.3. Unwrapping trajectory data

As mentioned in the previous section, unwrapping trajectory data aims to transform data from a curved surface so that they lie on a flat surface.

More specifically, the Unwrapping method is fitting several ellipses to the observed data points. These ellipses can then be unrolled onto a 2D surface. The basic idea behind this method is rather simple and intuitive: Imagine our cells are migrating on the peel of an orange, which is clearly a curved surface describing a sphere or an ellipsoid. The aim is now to peel the orange in such way that we can lay the peel on the flat table and still conserve the characteristics of the cell trajectories. We are here interested to conserve directional characteristics, more than distances. The resulting transformed cell trajectories can now be analyzed with the commonly used tools.

Unwrapping is best suited for 3D objects that have a rather small intrinsic and convex curvature. This means before this method is applied, we already have an idea of the true geometry.

1.5.4. Manifold learning

Manifold learning refers to a diverse suite of methods that aim to generalize well-known *linear* dimensionality reduction methods – Principal Component Analysis (PCA), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA) – to account for non-linear features in the data.

The key assumption underlying dimensional reduction methods – linear and non-linear – is that the ‘true’ number of degrees of freedom is lower than the apparent dimensionality of the data. The problem addressed in this paper gives the simplest and perhaps the most explicit illustration: we have point-cloud data in three dimensions constrained to lie on two-dimensional surfaces, which may or may not be flat.

The study of smooth curved spaces belongs to the mathematical field of *differential geometry*. There is an intuitive idea underlying this field: at small enough scales, every local patch of a surface can be approximated by a flat surface; a curved manifold is then simply an overlapping patchwork of (small) flat spaces. This is the approach adopted by most manifold learning algorithms – that is, to identify local neighborhoods of data points, treat these as linear spaces, and then find some ‘optimal’ method of joining these together into a flat global representation. A necessary requirement for these methods to work well, therefore, is that the density of the data points is high enough to allow one to consider small linear patches.

If one starts to consider distances between points and angles between vectors, then we augment this description of the manifold with a *metric*, which is a mathematical object that, loosely speaking, provides a local specification of lengths and angles. A manifold with a metric is known as a *Riemannian manifold* and is the object of study in *Riemannian geometry*.

Manifold learning algorithms do not explicitly preserve the metric information. However one can augment these methods by extracting the metric at each of the data points. This turns out to be essential for obtaining accurate directional statistics, as we show in this paper.

1.6. Image processing and cell tracking.

Imaging resulted in image stacks with dark background and fluorescent cells. The image processing was done in R using the package EBImage [1]. The information of the cells was extracted automatically from the images using an edge detection method. A manually set threshold of the light intensity was used per image stack. Each detected cell was described as an object with the coordinates of its geometrical center indicating the cell location and the time the cell was observed. The cells were tracked and reconstructed over the z-stack using a surface algorithm. The surface algorithm was then applied to track reconstructed cells over time, which is based on the shortest distance between cells from two consecutive images. We excluded all cell trajectories that included time points in which the cell was located at the edge of the image. Extracted cell tracks were reoriented, so that the center of the imaged object (embryo or yolk syncytium) was positioned in the center of the coordinate system ($x = y = z = 0$) for further processing.

1.7. Dimensional reduction: from linear methods to Riemannian manifold learning.

In this section we provide the theoretical background to the methods employed in the paper, including the unwrapping method outlined in Section 1. The challenge of describing and visualizing the geometry of embedded curved surfaces is commonly encountered in physics [2], computer vision, and machine learning [3] tasks. The techniques used are those from differential geometry or, more specifically, Riemannian geometry. For the sake of completeness and consistency, we adopt this more formal mathematical description. We provide a brief introduction to the essential topics; for more details, we refer the reader to [2]. Let \mathcal{M} be a smooth m -dimensional manifold and g the Riemannian metric defined for every point $p \in \mathcal{M}$. For a smooth manifold \mathcal{N} with $\dim(\mathcal{N}) \equiv n \geq m$, let $f: \mathcal{M} \rightarrow \mathcal{N}$ be an isometric embedding, *i.e.* for all $p \in \mathcal{M}$ and tangent vectors $u, v \in T_p\mathcal{M}$

$$\langle u, v \rangle_{g_p} = \langle df_p(u), df_p(v) \rangle_{h_{f(p)}},$$

here $\langle \cdot, \cdot \rangle_{g_p}$ is the inner product on the tangent space $T_p \mathcal{M}$, $g_p \equiv g(p)$, and h the metric defined for every point $q \in \mathcal{N}$. $df_p: T_p \mathcal{M} \rightarrow T_{f(p)} \mathcal{N}$ is the Jacobian of f at p .

For a dataset $D = \{q_1, \dots, q_N\}$ of points in \mathcal{N} , dimensional reduction is the task of inferring the inverse map $f^{-1}: \mathcal{N} \rightarrow \mathcal{M}$. For many purposes it is often sufficient to infer the corresponding images $\{x(p_1), \dots, x(p_N)\}$ for $q_i = f(p_i)$ and in some coordinate chart $x: \mathcal{M} \rightarrow \mathbb{R}^m$. Throughout this paper, we restrict ourselves to $\mathcal{N} \subset \mathbb{R}^3$ (i.e. 3D imaging data). If $\mathcal{M} \subset \mathbb{R}^{1,2}$ then we can use linear dimensional reduction methods. Here we consider two linear and three non-linear methods.

Projection into the XY –plane. A trivial and linear dimensional reduction method is the simple projection onto some pre-defined set of coordinate axes. We assume, without loss of generality, that these are the first m coordinates; in two dimensions, these are the X - and Y -axes, hence the name. Then for $i = 1, \dots, N$, we simply have

$$[x(p_i)]_a = [q_i]_a, \quad a = 1, \dots, m,$$

where the a subscript is the coordinate index.

Principal component analysis. Instead of pre-specifying the axes, we can perform linear dimensional reduction via principal component analysis (PCA). Let $C = \frac{1}{N-1} \sum_{i=1}^N (q_i - \bar{q})(q_i - \bar{q})^T$ be the sample covariance matrix, with the mean $\bar{q} = \frac{1}{N} \sum_{i=1}^N q_i$. Then for the rotation matrix $R = (e_1 | e_2 | e_3) \in O(N)$, e_1, e_2, e_3 the eigenvectors of C in decreasing order of their respective eigenvalues, we have

$$[x(p_i)]_a = [Rq_i]_a, \quad a = 1, \dots, m.$$

Unwrapping. We introduce a method to map data points on a 2D convex surface onto a subspace of \mathbb{E}^2 , the 2D Euclidean space, which we call the Unwrapping method. This method is a particularly effective approximation for surfaces of small intrinsic curvature (e.g. a thin cigar-shaped surface, a small patch on a large curved surface, etc). The unwrapping happens in two steps, both of which involves fitting a series of 1D ellipses to the data. The first step is a transformation, which removes the intrinsic curvature of the surface while seeking to maintain the geometrical relationships between the points (i.e. distances, angles, etc). The second step simply unwraps the transformed surface onto a flat 2D surface.

Let (x_i, y_i, z_i) represent the 3D coordinates of the data point q_i . If we approximate the dataset as points on a subspace of an ellipsoid, we choose to align our coordinate system such that the largest radius of the ellipsoid is described by the x -axis and the second largest radius is the y -axis. Next the data points are clustered into n equal-sized bins along the x -axis. For each cluster c we fit an ellipse E_c as the locus of the equation

$$\frac{y^2}{r_y^2} + \frac{(z-m_z)^2}{r_z^2} = 1,$$

for radii r_y, r_z and the z -coordinate m_z of the midpoint $(\bar{x}_c, 0, m_z)$, with \bar{x}_c the mean x -coordinate of the cluster c . The ellipse is then unrolled onto a straight line parallel to the y -axis with $x = \bar{x}_c$ and $z = z_c^{max}$ the maximum z -coordinate value of the points in cluster c . This then guides the first transformation of the points $(x_i, y_i, z_i) \rightarrow (x'_i, y'_i, z'_i)$ as follows. We keep the x -coordinate fixed, i.e. $x'_i = x_i$. As for the z -coordinate, because, in general, the data points do not lie on the ellipse (i.e. $\notin E_c$), we let z'_i be equal to the difference in distances to the center of E_c from q_i and the point on E_c along the line joining q_i and the centre. It can be shown that

$$z'_i = |q_i - q_i^E| \equiv d_i,$$

where q_i^E is a point with components

$$\begin{aligned} [q_i^E]_1 &= x_i \\ [q_i^E]_2 &= \frac{y_i r_z^2}{2(z_i - m_z) r_y^2} - \sqrt{\frac{y_i^2 r_z^4}{4(z_i - m_z)^2 r_y^4} + \frac{y_i r_z^2}{z_i - m_z}} \\ [q_i^E]_3 &= \sqrt{\frac{r_y^2 r_z^4 - r_z^2 ([q_i^E]_2)^2}{r_y^2}} \end{aligned}$$

To determine y'_i , we define a point-specific ellipse E_i with the same centre $(\bar{x}_c, 0, m_z)$ as E_c but with radii $r_{z,i} = r_z + d_i$ and $r_{y,i} = r_{z,i} \frac{r_y}{r_z}$. Then if $q_{i,z}$ is the intersection of the E_i with the xy -plane, y'_i is then the arc length of E_i between $q_{i,z}$ and q_i . Repeating this transformation for all n clusters results in the first unwrapping of the data points.

For the second unwrapping the same procedure is repeated on the transformed data set giving $q'_i \rightarrow q''_i$, but with the variable swap $x \leftrightarrow y$. Note, if all data points $\{q_i\}_{i=1}^N$ lie strictly on an ellipsoidal surface then $z'_i = 0$ for all $i = 1, \dots, N$. An example tutorial is provided in suppl. material.

Manifold learning. *Manifold learning* refers to a class of non-linear dimensional reduction methods that seek to recover the geometry of the low-dimensional manifold. These include ISOMAP [4], Locally Linear Embedding (LLE) [5], and Laplacian Eigenmaps [6], amongst several others. In every case, the metric on M is a global Euclidean metric, *i.e.* $g_{ab} = \delta_{ab}$, where δ the kronecker delta or identity matrix and a, b the coordinate indices. We refer to these approaches as *Euclidean manifold learning*.

In this paper we have employed LLE in our simulation and analysis of real data. LLE is based on the expectation that given a sufficiently large data set, each data point and its closest neighbors lie on a locally linear patch of the surface. The algorithm has two steps: 1. Expressing each higher dimensional data point as a linear combination of its neighbors, and 2. Obtaining a set of lower dimensional coordinates given relations above. In both steps, we proceed by minimizing the mean square errors of the data points from their linear reconstructions.

Our motivation for adopting LLE comes from both its intuitive approach to dimensionality reduction (locally linear patches) and also its effectiveness in providing an isometric reduction for surfaces with no intrinsic curvature, *e.g.* the surface of a cylinder or data points on a cylindrical ‘swiss-roll’. Furthermore it is also widely used in the machine learning community. In this paper we have used the implementation of LLE in the Scikit-learn Python machine learning package.

Riemannian manifold learning. Riemannian manifold learning aims to augment the set of coordinates $\{x(p_i)\}_{i=1}^N$ with the corresponding set of local metric values, *i.e.*

$$\{(x(p_i), \delta_{ab})\}_{i=1}^N \rightarrow \{(x(p_i), [g(p_i)]_{ab})\}_{i=1}^N,$$

where $a, b = 1, \dots, m$ label the metric components. By definition, g is symmetric and positive semidefinite. In this setup, one can recover the precise geometrical information of the embedded manifold. In this paper, we have adapted the LEARNMETRIC5 [3] algorithm to recover the 2D coordinates and the corresponding metric components from 3D data points.

For certain applications, such as computing the geodesics (see below), there is a need to derive metric values for out-of-sample points on \mathcal{M} . The metric $g(x)$ for $x \notin \{x(p_i)\}_{i=1}^N$ are approximated in two steps. First we perform a regression for each of the $m(m+1)/2$ unique components of g . In this paper we used the

implementation of Gaussian Processes in the Scikitlearn *Python* machine learning package. Second we satisfy the positive semidefinite constraint by replacing the matrix $g(x)$ with the nearest positive semidefinite matrix as measured by the Frobenius Norm. We implement this using the approximation method of Higham (2002) [7].

1.8. Extraction of geometrical information

All the relevant geometrical information of interest can be extracted from the metric. The angle between the two vectors u, v is given by

$$\theta = \cos^{-1} \frac{\langle u, v \rangle_{gp}}{\sqrt{\langle u, u \rangle_{gp} \langle v, v \rangle_{gp}}}.$$

The geodesic $\gamma: \mathbb{R} \rightarrow \mathcal{M}$, is the path of extremal length and is the solution to the set of Hamiltonian equations. With slight abuse of notation writing $x(\gamma(t)) \equiv x(t)$, these equations are

$$\begin{aligned} \dot{x}^a &= \frac{\partial H}{\partial r_a} = g^{ab} r_b, \\ \dot{r}^a &= -\frac{1}{2} \frac{\partial g^{bc}}{\partial x_a} r_b r_c, \end{aligned}$$

where r_a is the conjugate momenta to x^a , g^{ab} the components of the inverse metric, $\dot{x} \equiv dx/dt$, and the Hamiltonian

$$H = \frac{1}{2} g^{ab} r_b r_a$$

The bias direction from cell at a given time to a point source of attraction is given by the tangent vector u' to the geodesic connecting the two points. Therefore we solve the geodesic equations (9) for $x(t)$ under the constraints

$$x(t=0) = x(p_1), \quad x(t=1) = x(p_2).$$

One approach is to simulate these geodesics from p_1 via, say, the simple Euler method [8] and find the initial vector that generates the points on the geodesic that intersects p_2 . However this seemingly straightforward process is highly sensitive to errors in the out-of-sample metric approximations – the errors compound and one often ends up with unstable trajectories. In this paper, we implement a more stable and efficient discrete approximation as follows.

We first overlay a grid over the learned manifold. Here we fix the grid dimensions to 200 x 200. Next, using the Gaussian Process regression method described above, we obtain the metric values at every grid vertex and consequently the lengths of the sides of the cells across the grid. Then we approximate our geodesics between the point of interest and the bias point by the grid path that minimizes the total length. We use Dijkstra's algorithm to accomplish this step. Finally, we approximate the initial vector by performing a simple linear regression on the first few vertices in our discretized geodesic path.

1.9. Directional Statistics.

The straightness index, D , is commonly used to investigate cell migration strategies and it is defined as $D = \frac{|x_0, x_T|}{l}$, where x_0 is the position of the cell at time 0, x_T is the position of the cell at time T, $|x_0, x_T|$ indicates the shortest distance between x_0 and x_T and l is the actual length of the path the cell took from x_0 to x_T . For most applications the shortest path between start and end point is simply the Euclidean distance, however, for curved surfaces the metric needs to be learned.

Another way of describing cell migration tracks is by determining their bias towards a specific target (source of

attractant, like wounds or other cell types) and their persistence. We define the persistence of a cell as the probability of the cell moving at time t in the same direction as at time $t - 1$. Therefore, we need to compute the angles (β_t) between two motion vectors, which will result in a characteristic distribution. The wrapped normal distribution has been successfully used to describe persistent movement of cells. The probability density function is defined as

$$N_w(\beta_t|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\beta_t - \mu + 2\pi k)^2}{2\sigma^2}\right),$$

where μ is the mean and σ is the standard deviation. In the case of persistence we have $\mu = \beta_{t-1}$. We can then define the strength of the persistence p as $\sigma = -2\log(p)$. A cell that is highly persistent has a p close to 1, while a cell that is not persistent at all has a p of 0, in which case the wrapped normal distribution becomes a wrapped uniform distribution. The bias of a cell is also described by an angular distribution. Here, we compute the angle (α) between a motion vector of a cell and the vector that points from the cell towards the target. Again, we apply the wrapped normal distribution with the bias parameter b (instead of p) describing the strength of the bias and $\mu = 0$.

1.10. Simulation of random walks on ellipsoids

The ellipsoid is defined as the set of points satisfying

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1,$$

with x, y, z the 3D cartesian coordinates, and a, b, c the three shape parameters. We consider several different ellipsoid shapes with a/c ratios from the set $\{1, 0.66, 0.5, 0.4, 0.33, 0.1\}$ where $a = b$ throughout. We use the 2D parameterization

$$\begin{aligned} x &= a \cos\mu \sin\nu, \\ y &= b \sin\mu \sin\nu, \\ z &= c \cos\nu, \end{aligned}$$

with metric components

$$\begin{aligned} g_{\mu\mu} &= (\sin^2\nu)(a^2\sin^2\mu + b^2\cos^2\mu), \\ g_{\mu\nu} &= g_{\nu\mu} = \sin\mu \cos\mu \sin\nu \cos\nu, \\ g_{\nu\nu} &= (\cos^2\nu)(a^2\cos^2\mu + b^2\sin^2\mu) + c^2\sin^2\nu. \end{aligned}$$

We simulate 400 random walks on each of the ellipsoids as follows: starting from the initial point $(\mu_0, \nu_0) = \left(\pi, \frac{\pi}{2}\right)$, we randomly select an initial angle of motion $\theta_0 \sim B(\theta, \theta'_0)$, where B is the bias angle distribution with θ'_0 the direction to the bias source at time-step index t ; θ'_0 is determined following the minimization procedure described above and, in turn, defines an initial tangent vector. The particle moves along the geodesic generated by this vector with random step length taken from a χ^2 -distribution with mean $k = 2$; the absolute lengths are scaled with a constant factor in the range 0.015 – 0.25 to ensure that the trajectories cover the ellipsoid. At each subsequent time step $t > 1$, the particle changes direction and takes an angle according to a weighted distribution

$$\theta_t = wB(\theta, \theta'_{t-1}) + (1 - w)P(\theta - \theta_t^{old}),$$

where P is the persistence angle distribution and θ_t^{old} is angle of motion prior to changing directions at time step t . Both P and B are either uniform distributions in the range $(0, 2\pi]$ or wrapped normal distributions with parameter $\sigma = 1.2$, depending on the type of random walk being simulated. For the bias persistent random walk, we fix the weight parameter to be $w = 0.5$. We repeat each path for 20 time steps, giving 21 trajectory points per path.

2. Supplemental figure, movie and data legends

Figure S1. Related to Figure 2. Performance comparison of xy-projection, the unwrapping method and manifold learning methods. Shown are the exact distributions (black) that describe bias and persistence for 3 types of random walks: (A) Brownian motion, (B) biased random walk and (C) persistent random walk. Cell trajectories were simulated on the surface of ellipsoids with different ratios of its radii (a/c ratio = 0.1, 0.25, 0.33, 0.5, 0.66 and 1.0). For each scenario we compute the distributions based on the xy-projection, the unwrapping method and the two manifold learning methods. The two manifold learning methods are not able to deal with the two lowest a/c ratios and were left out. The reason is that these ellipsoids were too elongated, so that the manifold learning methods treated the data as if they were located in a slim plane. (E) Shown are the exact persistence angle distribution (black), the persistence angle distribution computed from the 3D vectors (green) and the persistence angle distribution resulting from unwrapping the data into a non-curved surface (red). The underlying data are persistent random walk trajectories simulated on an ellipsoid with radii ratio $a/c = 0.33$. (F-G) Application of the unwrapping method (orange) to neutrophil cell tracks extracted from the epidermis overlying the yolk syncytium of a zebrafish and its comparison to the xy-projection (blue) and principle component analysis (green). The epidermis was wounded with a laser before image acquisition. (Corresponds to Figure 2F)

Figure S2. Related to Figure 2. Performance measurements of the different methods. We computed the Kolmogorov-Smirnov distance (ks-distance) between the true angular distributions and the extracted angular distributions using xy-projection, unwrapping, Euclidian manifold learning and metric manifold learning. We considered the same data generated for the random walk models described in Supplemental Figure 1A-C. The smaller the ks-distance is, the better is the performance of the methods.

Figure S3. Related to Figure 2. Shown are the 2D projections of cell tracks extracted from *Drosophila* embryos (colored tracks) after applying (A) xy-projection, (B) unwrapping and (C) manifold learning. The green, orange, dark blue and red tracks are highlighted for easy comparison between the methods. The grey lines in (B) and (C) are for comparison with the xy-projection as they are the same tracks using the xy-projection.

Figure S4. Related to Figure 2. Manifold learning with more complex data. (A) We simulated random walk cell trajectory data on a complex surface. Shown is the 3D representation of these data (blue) with the xy-, xz- and yz-projections (grey). This example is too complex to be successfully transformed via unwrapping, but can be dealt with manifold learning techniques. The trajectories display a Brownian motion type random walk, where the bias and persistence distributions are expected to be flat. (B) We analyze the data in the xy-projection and compare this to the Euclidian manifold learning algorithm and the metric manifold learning algorithm. As can be seen in supplemental figure 4B, the xy-projection induces extreme artifacts. The bias distribution is well extracted from both manifold learning methods, while the persistence distribution is only correctly extracted using the metric manifold learning algorithm, highlighting the need to accurately learn the metric of the data

Movie S1. Related to Figures 1 and 2. Shown are example raw data for the unwounded *Drosophila* embryo data set analyzed in figures 1A, 2E and Supplemental Figure 1. Time-lapse movie of the dynamic behavior of *Drosophila* immune cells (haemocytes) in unwounded tissue. Epithelial cells are labeled using E-cadherin-GFP (green cell outlines), immune cell nuclei are labeled using nuclear Red-Stinger (red) and immune cell cytoplasm using cytoplasmic GFP (green) both driven by *srp-Gal4*.

Movie S2. Related to Figures 1 and 2. Shown are example raw data for the wounded zebrafish data set analyzed in figures 1B and 2F. Time-lapse movie of the dynamic behavior of zebrafish immune cells (neutrophils) in laser-induced wounded tissue. Immune cells are labeled using cytoplasmic dsRed (red) driven by the lysozyme C (*lyz*) promoter.

Movie S3. Related to Figures 1 and 2. Shown are the haemocyte cell tracks extracted from *Drosophila* embryos in 3D. The different rotation angles show the curvature of the space the haemocytes are migrating in.

Movie S4. Related to Figures 1 and 2. Shown are the neutrophil cell tracks extracted from zebrafish embryo.

The epidermis overlying the yolk syncytium was wounded. The location of the wound is indicated by a light blue dot.

Data S1. Related to Experimental Procedures. This folder contains the two Jupyter notebooks for Unwrapping and Manifold learning. Furthermore it includes the two websites for both methods. Example data are stored in the folder 'SimulationData'.

Data S2. Related to Experimental Procedures. This folder contains all described R scripts and the provided example data in order to perform Unwrapping on simulated and *in vivo* data.

3. Supplemental figures.

Figure S1. Related to Figure 2.

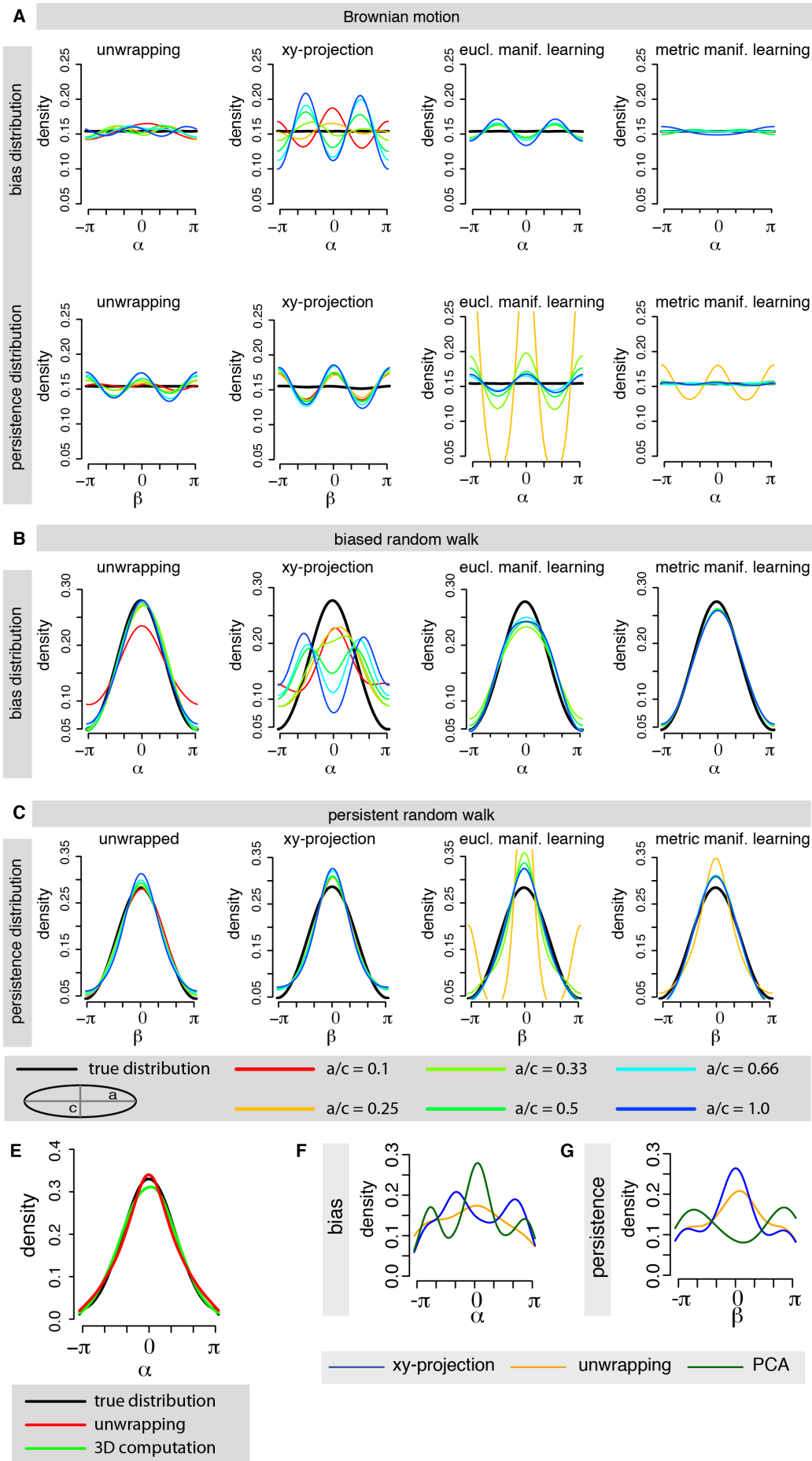


Figure S2. Related to Figure 2.

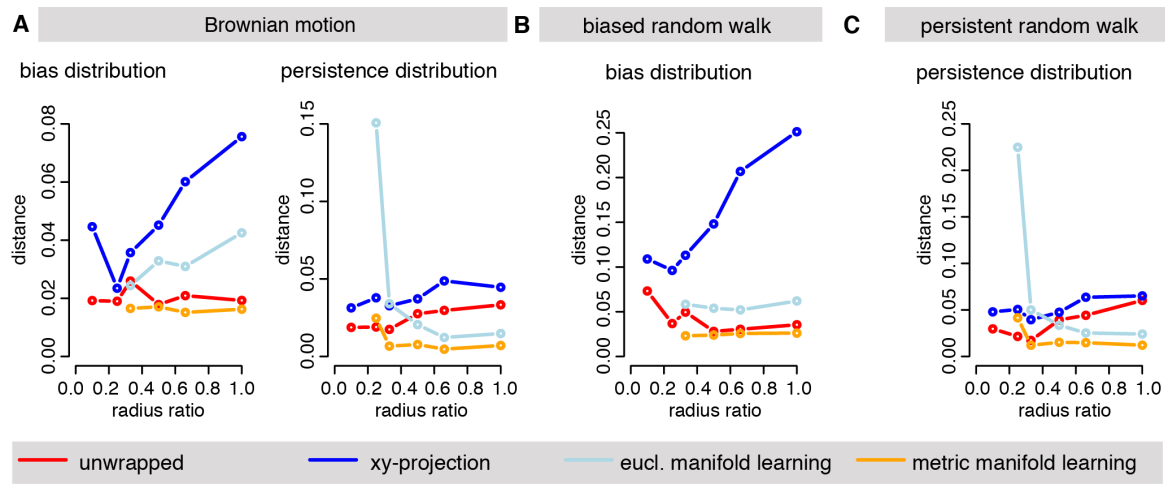


Figure S3. Related to Figure 2.

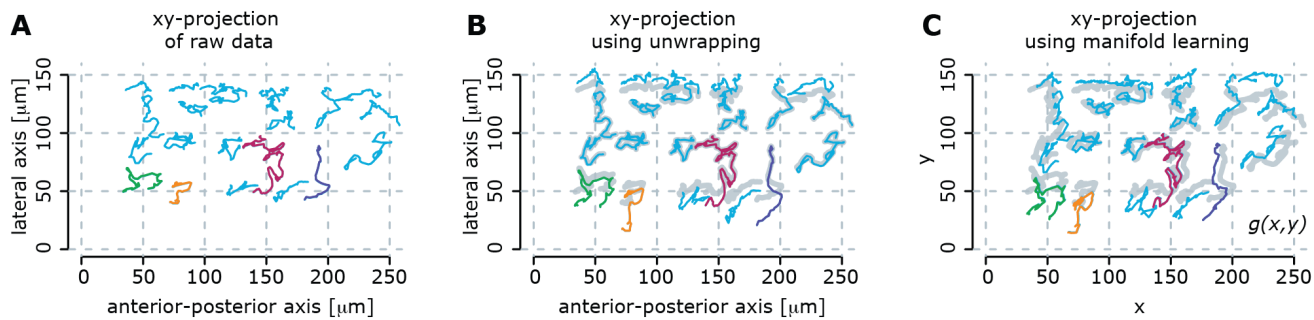
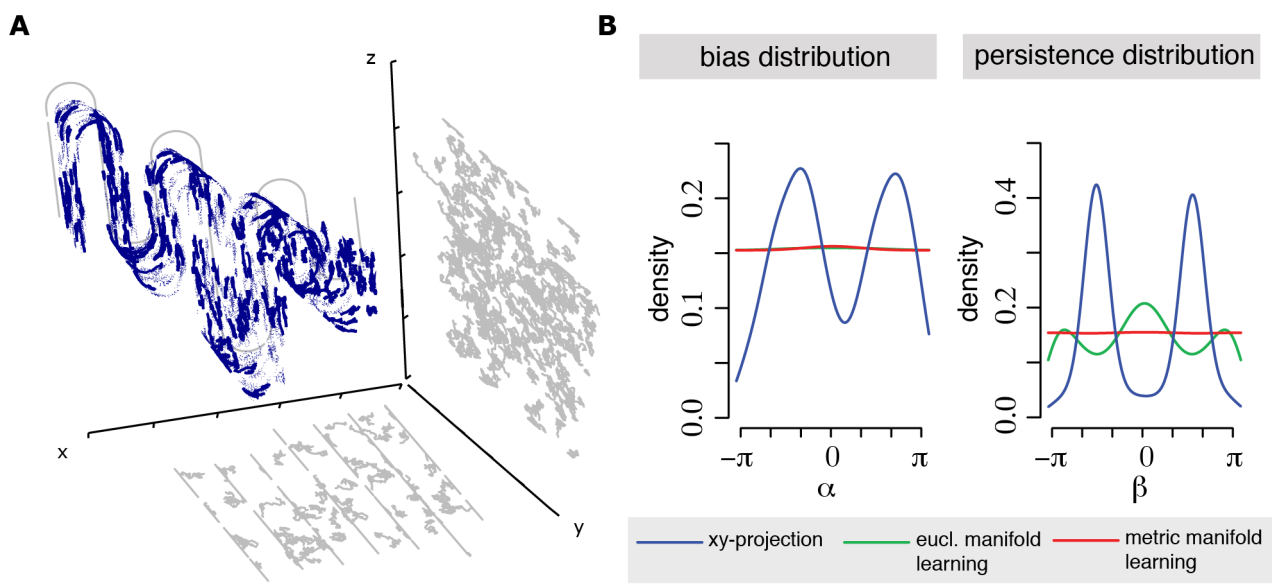


Figure S4. Related to Figure 2.



4. Supplemental References.

1. Pau, G., Fuchs, F., Sklyar, O., Boutros, M. & Huber, W. Ebimage - an r package for image processing with applications to cellular phenotypes. *Bioinformatics* 26, 979–81 (2010).
2. Nakahara, M. *Geometry, Topology and Physics, Second Edition* (CRC Press, 2003).
3. Perraul-Joncas, D. & Meila, M. Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. *arXiv.org* (2013). 1305.7255v1.
4. Tenenbaum, J. B., de Silva, V. & Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N.Y.)* 290, 2319–+ (2000).
5. Roweis, S. T. & Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)* 290, 2323–+ (2000).
6. Belkin, M. & Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15, 1373–1396 (2003).
7. Higham, N. J. Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis* 22, 329–343 (2002).
8. Leimkuhler, B. & Reich, S. *Simulating Hamiltonian Dynamics* (Cambridge University Press, 2004).