COMPUTATION AND ANALYSIS OF CELLULAR TRAJECTORIES USING LIGHT-SHEET FLUORESCENT MICROSCOPY

A Thesis

Presented to the Faculty of the Graduate School

of the University of Chicago

in Partial Fulfillment of the Requirements for the degree of

Master of Science

Ashwin Chetty

August 2020

Abstract

Light-sheet Fluorescent Microscopy (LSFM) is an emerging technology in embryology which allows for the tracking of cells in entire organisms as they develop—at larger sizes, greater resolution and with less photodamage than when compared with other methods. LSFMs produce large high-resolution 3D+t datasets; and when they are used to image organisms transfused with histone-GFP fusion protein, it is possible to obtain information about the migration of cells in the developing organism. However, any 3D microscopy technique suffers from reduced resolution due to scattering, especially at higher depths within the image—which confounds cell identification and tracking. We discuss methods to recover cell positions from greater z-depths of the microscope image; present a simple multiscale blob-based tracking strategy; and then discuss statistics that can be performed on the space of trajectories once they are computed.

Section 1: Introduction

Light-Sheet Fluorescence Microscopy

Light-sheet Fluorescent Microscopy (LSFM) is a powerful technique in developmental biology which allows for the 3D+t imaging of developing organisms at larger scales and with finer resolution than can be obtained with other methods, such as two-photon microscopy or confocal laser scanning microscopy (CLSM) [1]. In LSFM, a planar sheet of light illuminates successive horizontal slices of a transparent organism, and fluorescent responses within the organism (for example, from expressed GFP or RFP) are captured by a camera perpendicular to the sheets of light. A 3D volume is constructed after all horizontal slices are illuminated; and a 3D+t dataset is constructed when the procedure is repeated at various timepoints. Because only one sheet of the organism is illuminated at a time, photobleaching and photodamage to the organism are reduced; and because an entire sheet of the organism can be imaged at once, LSFM is able to capture images more rapidly than is possible with other methods (such as CLSM) [2].

LSFM is able to produce high-quality images at subcellular resolutions [3], which makes it an ideal modality for studying the migration of cells within developing embryos [4]. In particular, to study the migration of developing neurons, transparent Zebrafish (Danio rerio) embryos have been transfused with fluorescent histone-GFP (green fluorescent protein) fusion protein, which causes cell nuclei to fluoresce. Within the light-sheet microscope, the laser sheet successively illuminates slices of the Zebrafish, and when excited the GFP-histone molecule re-emits green light that is captured by the

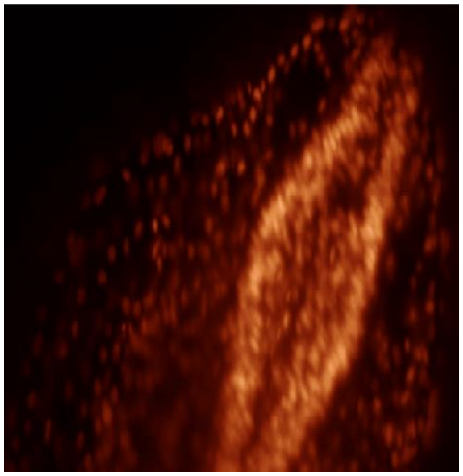


Figure 1 - 2D slice of a developing Zebrafish embryo imaged with LSFM

camera. GFP captures light at a peak wavelength of 400nm and re-emits at 500nm, which fluoresces as a visible green.

At each timepoint, the light-sheet microscope produces a series of 2D images (one image at each z-depth), which are then collated into a 3D volume. A 3D+t dataset is constructed when 3D image-stacks are collected at multiple timepoints. This dataset can then be used to track the migration of cells over time [5].

Cell Tracking

Although fully-automatic cell tracking has not yet been realized (most algorithms require careful parameter tuning or post-processing to work well), cell-tracking has been the subject of a substantial body of recent work, and is relevant enough that it has even spawned a Cell-Tracking Challenge, a review of the submissions of which is described by Ulman et al. Cell-tracking algorithms are myriad and it is difficult to fully taxonomize them; however, they can be broadly classified into two categories—"tracking by contour evolution" methods which segment cells in the first frame of the dataset and then evolve their contours to follow them in subsequent frames; and "tracking by detection" methods which first segment the image into cells; and then only later establish temporal associations between subsequent frames [6]. Tracking algorithms generally operate in two steps: *segmentation* and *tracking* (in "tracking-by-contour-evolution" methods these steps are performed simultaneously). First, the location of cells within the image is determined and cells' constituent voxels are identified. Next, cells' migrations are tracked over time.

Amat et al. [7] have developed a software suite, Tracking-using-Gaussian-Mixture-Models (TGMM) that has achieved 3D full-scale cell segmentation and tracking with up to 97% accuracy on the datasets on which it was tested. The algorithm has seen success in several domains, including in zebrafish, fruit flies and mouse embryos and is the tracking algorithm that is used as a benchmark in this paper.

Optical Dynamics

In order to track cells, it is first necessary to find them within the image. It is a known problem that in 3D microscopy the resolution of the depth-axis of the microscope is substantially worse than those of the lateral axes. At lower depths, light must travel further through the specimen and immersion medium and undergoes greater scattering before it reaches the camera. Various research has been conducted in finding cells by forming exact physical models of the optical dynamics of microscopes [8]—however, these methods require precise understandings of the optical characteristics of both the microscope as well as the specimen and immersion medium (for example, refractive indices, working distances, etc.) and are often cumbersome to use.

Under ideal circumstances, a fluorophore acts as a point-source of light and when imaged under a microscope produces a diffraction pattern in the form of an Airy disk, which can be approximated by a Gaussian profile. For many purposes a Gaussian approximation for the diffraction point-spread function is simply *simpler* to use, and in fact provides a very good approximation in a number of applications [9]. In this paper, we model cell nuclei as having roughly Gaussian profiles.

Trajectory Modeling

Trajectories themselves are the subject of investigation in various domains. Tracking of people may be conducted in surveillance applications; tracking of objects may be conducted in robotics applications; and the tracking of animals may be conducted in the study of ecosystems [10]. As a result, a sizable amount of work exists on the subject of modeling trajectories in 2D spaces; less work has been conducted in modeling trajectories in 3D space; and many applications involve reducing 3D trajectories to 2D subspaces.

Fazli et al. have conducted work in trajectory clustering in 3D datasets of T. gondii cells [11] and have found a method to reparametrize cellular trajectories in a way that is invariant to absolute spatial coordinates. Moradi et al. have found success in performing statistics on trajectories by first modeling them as point patterns [12] and then conducting statistics between collections of points. Various other authors have written on trajectory clustering through several other means—for example by clustering trajectories based on common sub-trajectories in the TRACLUS algorithm. Various distance metrics can be applied for use on trajectories—for example the Hausdorff distance, Bhattacharyya distance, Frechet Distance, Longest-common-subsequence (LCSS) distance, and dynamic-time-warping (DTW) Distance.

Section 2: Cell Segmentation

Before cells can be tracked, they must first be found within the image. Cells in more dorsal regions of the embryo are clearly delineated (Figure 2) and can be identified by simply filtering the image (with a Gaussian blur, for example) and finding peaks. However, at greater z-depths cells become decreasingly differentiated from one another as light scatters more when it travels through the organism. Figure 1 shows the extent to which the spatial resolution of the image decreases with z-depth: at lower depths, cells are barely distinguishable. We propose a variant of the popular Laplacian of the Gaussian filter to correctly differentiate blurred cells.

Under ideal conditions, cell nuclei will appear as a bright spot of light convolved with a Gaussian kernel (simulating scattering and diffraction) in a pattern similar to that of a blurred Airy disk. As z-depth increases however, light emitted from fluorophores must travel longer distances through the medium before reaching the sensor and, due to variance in the optical characteristics of the specimen, it becomes difficult to predict the profile of the cell nucleus exactly. Still, we can assume that cell nuclei will appear as a bright spot of light convolved with some anisotropic kernel which *resembles* a Gaussian. The salient

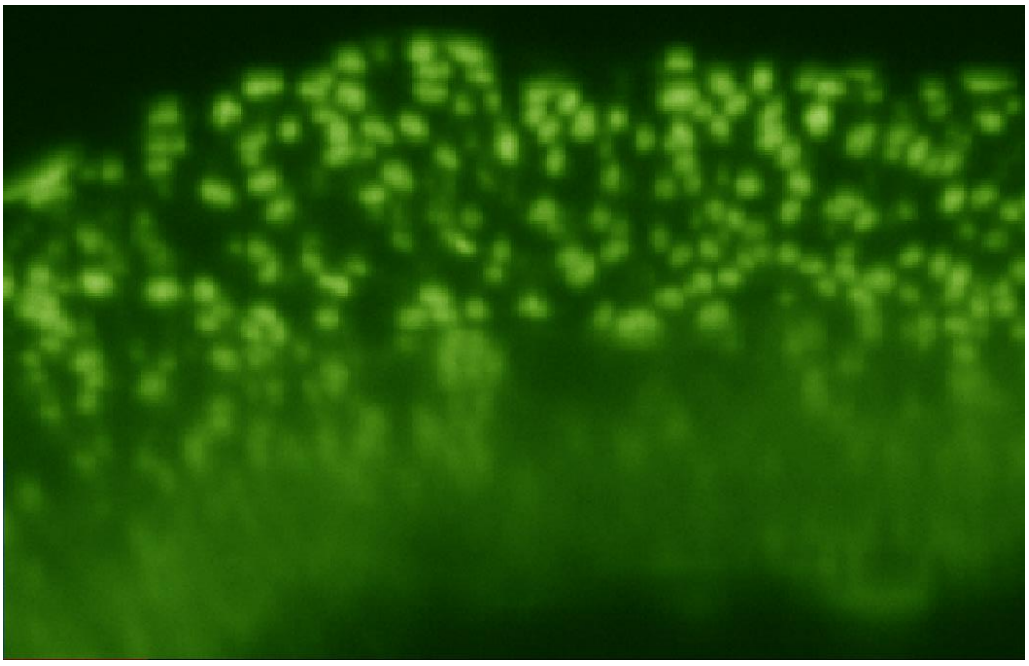


Figure 2 Slice along x-z axis of Zebrafish embyro, 18 hours-post-fertilization

feature of cell nuclei is that they should be local maxima of the image. That is to say, the *convexity* of the center of a blurred cell nucleus is negative, and hence we can use second-order methods to find cell nuclei. We propose finding cell nuclei using a variant of the Laplacian of the Gaussian.

The Laplacian of an image at a point p is the divergence of the gradient of the image at p, or equivalently the trace of the Hessian at p (or the sum of the *unmixed* second partial derivatives):

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} + \frac{\partial^2 I}{\partial z^2}$$

It is coordinate-invariant; and for a smooth image is high at areas of positive curvature, zero at inflection points, and low at areas of negative curvature. Intuitively, the Laplacian measures the speed with which, when considered as a fluid, the signal at a point will diffuse into surrounding regions of the image. For a discrete image, the Laplacian can be approximated as convolution with the 3D kernel

$$\Delta_{\text{discrete}} I = I * L, \text{ with } L[0] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, L[1] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, L[2] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The Laplacian can be found at larger scales by first convolving with a Gaussian G_σ:

$$\mathrm{L} \circ \mathrm{G}[\mathrm{I}] = I * L * G_\sigma = I * (G_\sigma * L)$$

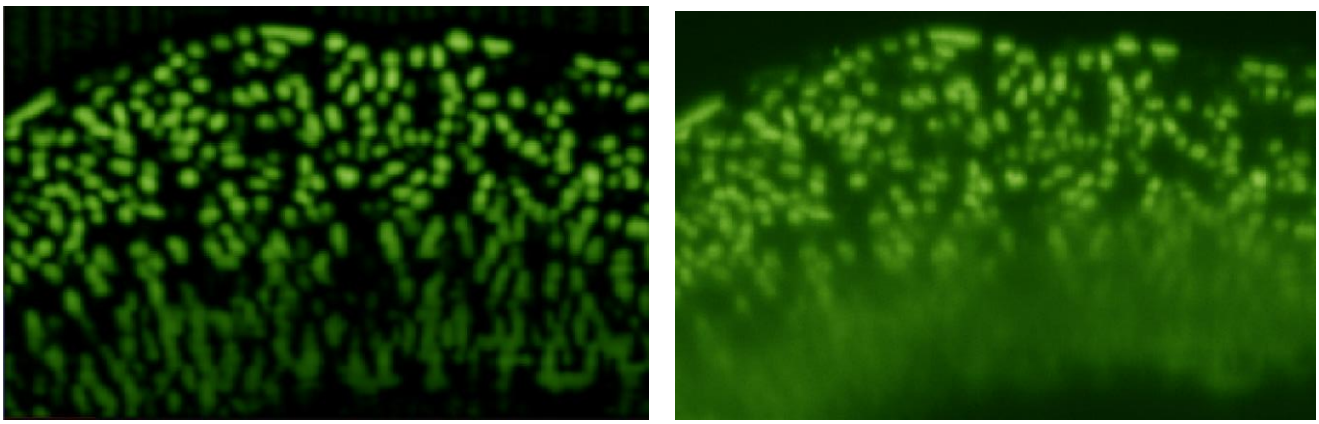


Figure 3 ClippedLaplacian applied on an image. Noise is removed and most cells are segmented from one another and separated by the 0-isocontour of the Laplacian.

The Laplacian filter is commonly used for detecting edges and blobs within an image. Zero-crossings of the filtered image correspond to edges; and troughs of the filtered image correspond to the centers of blobs. For the purposes of detecting cells, we are interested in both: a cell will be a bright spot surrounded by a fuzzy edge. We consider the *clipped Laplacian*, a filtering operation conducted on the image:

```
ClippedLaplacian[I] :=
  Clip[
    NegativeLaplacian[
      Gaussian(σ1)[
        Median(1)[I]
      ]
    ], 0.0, ∞
  ]
```

First, we take a median filter with radius 1 over *I* in order to remove speckled noise and to correct for artefacts in the data. Next, we compute the Laplacian of the Gaussian, find its negative, and discard negative values. This operation preserves only the hills of the image and discards valleys. σ_1 should be roughly the radius of a cell.

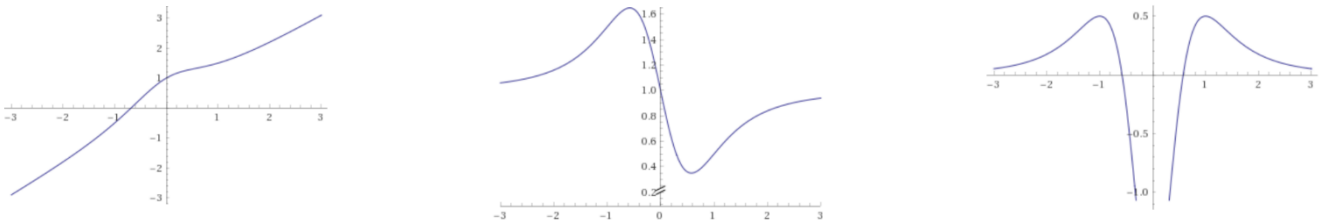


Figure 4 – 0th, 1st, and 2nd derivatives of the function $f(x) = \frac{1}{x^2+1} + x$. *f is always increasing, yet the second-derivative allows us to capture the shape of the inflection point centered around x = 0. This property is useful when finding blobs in regions of increasing noise.*

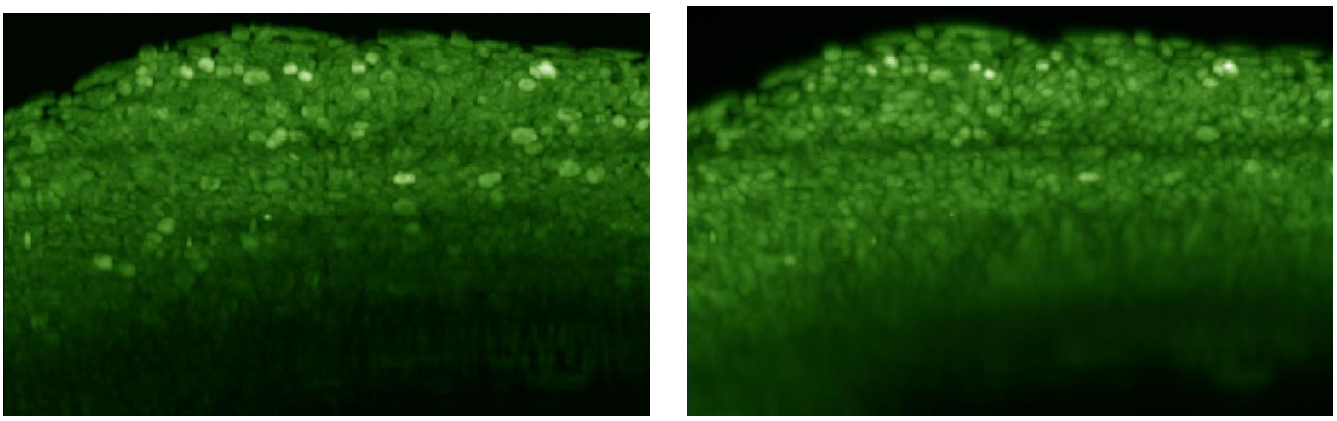


Figure 5 – Maximum-Intensity-Projection of (left) the filtered image and (right) the unfiltered image.

The result of this filter can be seen in Figure 2. If we find the local maxima of this image, we recover information about the centers of blobs in the image. Cells are also conveniently differentiated from one another by the zero-isocontour of the Laplacian. Because cells are separated from one another, the task of differentiating cells can be conducted with a simple watershed algorithm: starting at peaks and filling pixels until a minimum is reached. In regions where local maxima are connected, peaks of the Laplacian will define blobs.

This filter allows us to differentiate cells even in regions of low resolution (Figure 4). Figure 5 shows a maximum-intensity-projection (MIP) of the unfiltered and filtered images. Cells are segmented from one

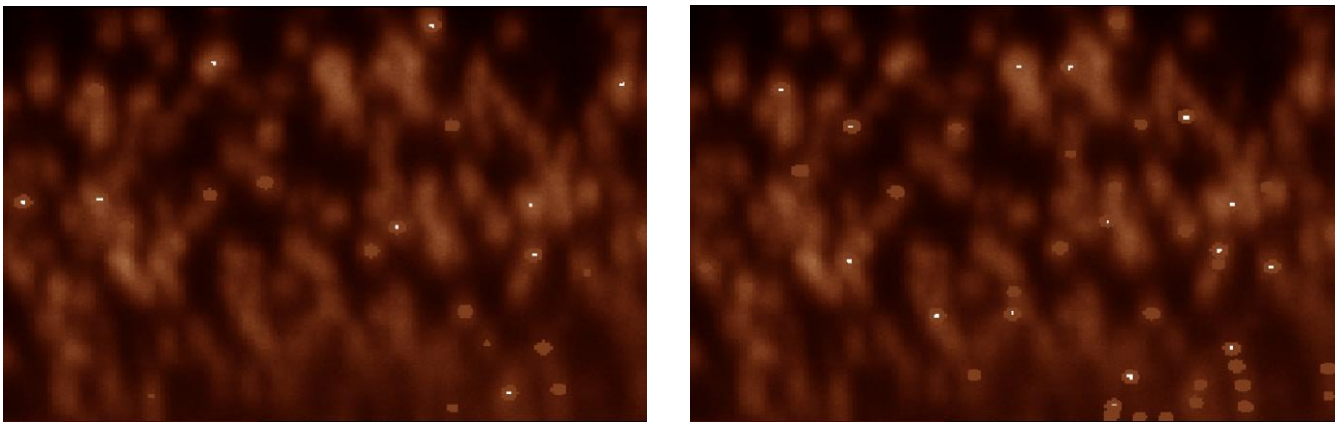


Figure 6- Cell nuclei captured with TGMM on a small 2D x-z slice of the data. (Left) when TGMM is run on a filtered image; (right) when TGMM is run on the unfiltered image. Recognized cell nuclei are marked with grey spots. Notice that when run on the unfiltered image TGMM has more false-positive detections in deeper regions.

another in the image and separated by their inflection points. For a given cell, the region within its inflection isocontour will be positive and the region outside of it will be zero.

Empirically, this filter is also an effective pre-processing step which can be run before a more complex cell-segmentation algorithm. By running this filter over the data prior to running TGMM (which uses a persistence-based clustering approach to identify cells), we were able to reduce the number of redundant cell detections especially in deeper regions of the image (Figure 6). Prior application of this filter also improved TGMM speed by a factor of 7, and reduced the amount of memory used during the segmentation task.

An alternate approach might involve filtering the image by finding the minimum eigenvalue of the Hessian, and setting the value at each pixel to be $I[p] \coloneqq \max\left[\min_i(\lambda_p)_i\right]$. Such an approach would provide information about the convexity of each pixel of the image which would be useful in shape-estimation.

Choosing a scale for the ClippedLaplacian

The clipped Laplacian is a useful preprocessing step that can be performed before running a more complex segmentation- or tracking algorithm. However, it is not immediately obvious how to find a scale at which to run this filter prior to analysis. For a given image and average cell radius, we find that there is a large window of acceptable values σ with which the filter can be run (Appendix B), which simplifies the task greatly. We propose a simple method to automatically find σ.

Consider the function $\mathbf{find\_blobs}(I, \theta),\ \theta \in \mathbb{Z}$, which finds blobs in the image I and then discards blobs which contain fewer θ voxels. This is useful for enforcing a minimum blob size and for discarding noise in the image. Let $\mathbf{\Lambda_\sigma} = |\mathbf{find\_blobs}(\text{ClippedLaplacian}(I, \sigma), \theta)|$ denote the count of the number of blobs in the filtered image I with volume greater than θ voxels. Here, θ is a parameter which enforces

the minimum volume of a blob in voxels. A reasonable value of θ might be $5^3 = 125$, which is larger than the size of a single voxel but smaller than the size of most blobs in the image.

When the clipped Laplacian is performed at a very low scale σ, a large number of small blobs will be found, most of which will have volumes smaller than θ, and $\mathbf{\Lambda_\sigma}$ will be quite small. For very large σ, most blobs will cease to exist and $\mathbf{\Lambda_\sigma}$ will again be small. In order to maximize the number of blobs with volume greater than θ, we thus let

$$\sigma_{max} = \underset{\sigma}{\operatorname{argmax}}\, \mathbf{\Lambda_\sigma} = \underset{\sigma}{\operatorname{argmax}}\ |\text{find\_blobs}[\text{ClippedLaplacian}(I, \sigma), \theta]|$$

The maximum can be found with a simple linear search over σ, and this parameter σ_{max} should be constant for all images in the dataset.

Minimum eigenvalues of the Hessian

Cell nuclei are differentiated based on their convexity. In 3D, a function f is convex at a point p if, when the Hessian of f is computed at p, its eigenvalues are all nonnegative (f is strictly convex at p if all of its eigenvalues are positive). To identify regions of the image which are part of cell nuclei, we identify pixels of the image with negative convexity:

$$I'[p] \coloneqq \max\left[0, \lambda_{1_p}\right]$$

Where λ_{1_p} is the minimum eigenvalue of the Hessian matrix of I at point p.

The discrete image Hessian can be computed as:

$$\mathcal{H}_p = \begin{pmatrix} \frac{1}{2}\left(2I_{p_{13}} - I_{p_{12}} - I_{p_{14}}\right) & & \\ \frac{1}{4}\left(I_{p_{15}} - I_{p_9} - I_{p_{17}} + I_{p_{11}}\right) & \frac{1}{2}\left(2I_{p_{13}} - I_{p_{10}} - I_{p_{16}}\right) & \\ \frac{1}{4}\left(I_{p_3} - I_{p_5} - I_{p_{21}} + I_{p_{23}}\right) & \frac{1}{4}\left(I_{p_1} - I_{p_7} - I_{p_{19}} + I_{p_{25}}\right) & \frac{1}{2}\left(2I_{p_{13}} - I_{p_4} - I_{p_{22}}\right) \end{pmatrix}$$

denoting p_n as the nth neighbor of p in raster order:

$$p_0 = p + (-1 \quad -1 \quad -1)$$

$$p_1 = p + (-1 \quad -1 \quad 0)$$

$$p_2 = p + (-1 \quad -1 \quad 1)$$

$$p_3 = p + (-1 \quad 0 \quad -1)$$

$$p_4 = p + (-1 \quad 0 \quad 0)$$

$$\vdots$$

$$p_{13} = p + (0 \quad 0 \quad 0)$$

and so on.

Although not discussed in detail in this paper, the Hessian image provides insight into the shapes and positions of blobs, and admits stronger guarantees about convexity than does the Laplacian. Figure 7 depicts a single 2D slice of the data. Note that the Hessian image is tighter than the Laplacian image, and not all blobs are large enough to be seen in the slice shown (though they are visible in deeper slices).

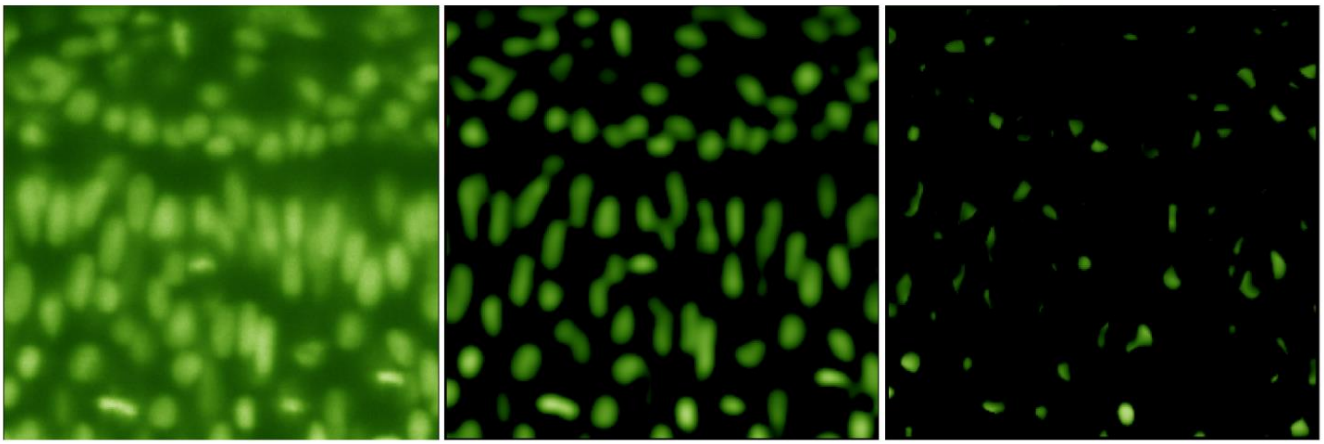


Figure 7 - (left) original image; (middle) Laplacian of Gaussian filter; (right) minimum-eigenvalue-of-Hessian filter

Section 3: Modeling Data with Multivariate Gaussians

Raster data requires a large amount of memory to store and is inherently difficult to perform computation on. It is useful to be able to model image volumes in terms other than as raw pixel values. Further, knowing the locations of cell nuclei is more physically significant than knowing the values of particular voxels in the image.

Data from the light-sheet microscope can be assumed to be comprised of a collection of cell nuclei surrounded by noise (ie. unusable data) and then blurred. We propose a method to approximate the volume of cells in the image as a sum of Gaussian profiles, which is an idea that arises naturally from the segmentation provided by the clipped Laplacian defined previously. We propose a scheme by which we view the data as being a sum of scaled multivariate Gaussians, which is a model which provides insight into the locations, shapes and orientations of cell nuclei.

The Multivariate Gaussian Distribution is a well-known probability distribution in $\mathbb{R}^p$ with probability-density-function

$$p(x|\mu, M, k) = (2\pi)^{\frac{k}{2}} \det(M)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T M^{-1}(x-\mu)\right)$$

for $x \in \mathbb{R}^p$, where $M \in \mathbb{R}^{p\times p}$ is the shape of the distribution, μ is its mean, and k is the dimension of the data.

Letting $k = 3$ and introducing a variable scaling parameter $\kappa \in \mathbb{R}^+$ before the exponent, we define the profile of a Multivariate Generalized Gaussian as

$$g(x|\mu, M, \kappa) = \kappa \exp\left(-\tfrac{1}{2}(x-\mu)^T M^{-1}(x-\mu)\right).$$

We attempt to find the set of best-fit Multivariate Gaussian profiles for a given 3D volume of the data.

Figure 8 - result of climb[climb[⋯ climb[p]]]. We assign a point p_i to its natural source when following the discrete gradient direction in the image. All points within this blob will belong to the same locus X_i

The data is assumed to be a collection of cell nuclei surrounded by i.i.d noise (ie. essentially unusable signal). Within the algorithm, cell nuclei will ultimately be modeled by multivariate Gaussian profiles. However, our model of the image has a bit more complexity:

We assume that the boundaries of histone molecules are precise, and that each cell nucleus admits an intensity function $\phi: P \to \{1,0\}$ indicating whether GFP is emitted from a particular location, where P is a location in the image. We assume that nuclei do not intersect and that GFP-emitting regions are ellipsoidal: ie. $\phi^{-1}(1)$ satisfies $\left(A(x-\mu)\right)^2 < 1$ for some $A \in \mathbb{R}^{3x3}, \mu \in I$, and all $x \in \phi^{-1}(1)$.

We assume then that each nucleus is subject to a symmetric Gaussian blur (due to optical scattering), and then subject to noise (from the sensor). Thus, the image will resemble:

$$I = \sum_i \kappa \cdot g_i * \mathrm{Im}(\phi_i) \; + n$$

Where $\mathrm{Im}(\phi_i)$ is the image which contains the intensity response of ϕ_i, ie. $\mathrm{Im}(\phi_i)[p] = \phi_i(p)$, $g_i * (\cdot)$ represents a symmetric Gaussian blur convolution kernel with some σ, and n is an i.i.d noise term with $|n| < \delta$ for some small δ. Experimentally, we find that the proposed algorithm works well with $n < 0.05 \cdot I_{max}$, where I_{max} is the maximum value in the image.

In practice, the shapes of cell nuclei rarely perfectly resemble a Gaussian (cell nuclei usually have flatter peaks and irregular shapes)—however, Gaussians provide a computationally-simple way to model cells.

We assume that at each region, there is some σ such that convolution of the region with σ will result in a local maximum of the Laplacian within the region. We estimate the positions of nuclei based on connected regions of the image with negative Laplacian.

Consider the image I. Compute the clipped Laplacian $L = \text{ClippedLaplacian[I]}$. Find the local maxima of L and store it in X,

$$X = \{X_1, \dots, X_n\} = \text{local\_maxima}(L)$$

$X_1, \dots, X_n$ thus list all of the local maxima of L, or equivalently all local minima of the Laplacian of I, or the natural sinks of the Laplacian of I. These X_i enumerate the peaks of blobs.

We next assign each nonzero point in L to a blob in X. For each $X_i = (x_i, y_i, z_i)$, let B_i be the set of points that are assigned to the blob centered at X_i. For each X_i, $B_i = \{p_{i_1}, p_{i_2}, p_{i_3}, \dots, p_{i_{m_i}}\}$.

For a voxel p within the image I, Define $\text{climb[p]} = p + \nabla_{\text{discrete}} I|_p$, where $\nabla_{\text{discrete}} I|_p$ is the discrete gradient of I evaluated at p. climb[p] assigns p to the adjacent voxel of greatest value; or, if p is already a local maximum, then $\text{climb}[p] = p$. We assign a voxel p to a blob B_i if $\text{climb}\big[\text{climb}\big[\cdots \text{climb[p]}\big]\big] = X_i$—that is, p reaches B_i through repeated hill-climbing. Thus, X_i contains the centers of blobs and B_i contains all of the voxels which are contained within blob X_i. The position of blob i is given by its locus X_i.

Our next task is to estimate the shape M_i and center μ_i of blob i. A blob i is defined by the image's fluorescent responses, which are greater in regions of higher fluorophore density and lesser in regions of lower fluorophore density (the relationship between fluorophore density and image voxel value will be

linear). The center μ_i of blob i can be found by finding the center of the fluorescent response, which is given by the weighted mean of the pixels within the blob:

$\mu_i = \frac{\sum_{p \in B_i} I[p] \cdot p}{\sum_{p \in B_i} I[p]}$.

Similarly, the shape M_i of blob i is given by the weighted covariance of the fluorescent responses, $Cov[B_i, I_{B_i}]$, which is the weighted covariance of B_i with weights given by image values I_{B_i}:

$$M_{i_{kl}} = \frac{1}{\sum_{p \in B_i} I[p]} \sum_{p \in B_i} (p_k - \mu_{i_k})^T (p_l - \mu_{i_l}) I[p]$$

We can then compute the scale parameter κ:

$$\kappa = \frac{\sum_{p \in B_i} \exp\left[-\frac{1}{2} p_i^T M p\right]}{\sum_i I[p]}$$

Future work may involve finding an appropriate scaling factor to transform shapes from Laplacian-space to image-space.

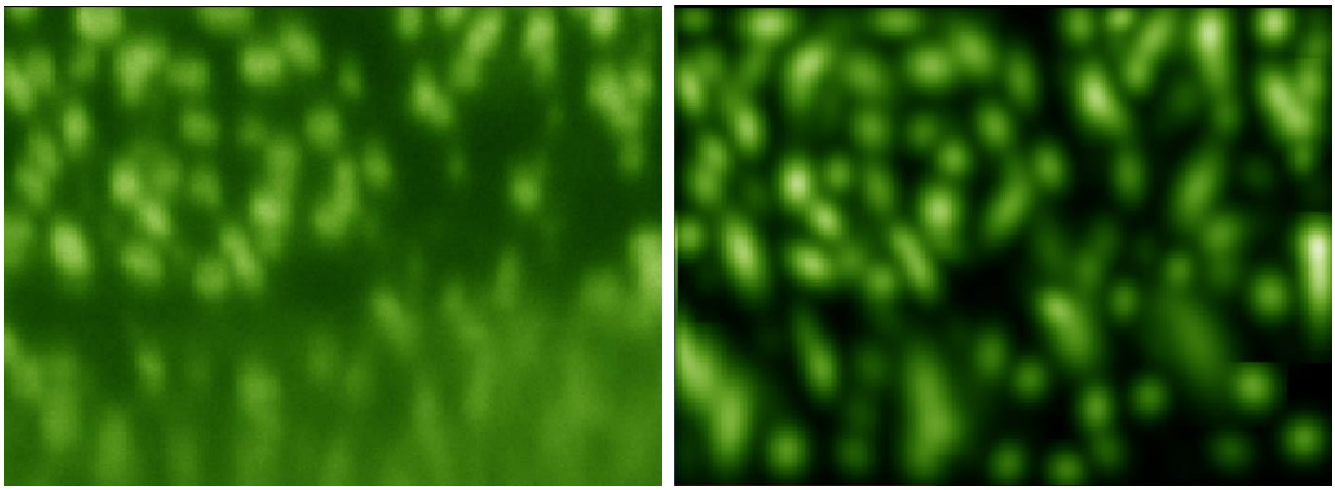


Figure 9- XZ- slice of (left) original data (right) data modelled by multivariate Gaussians.

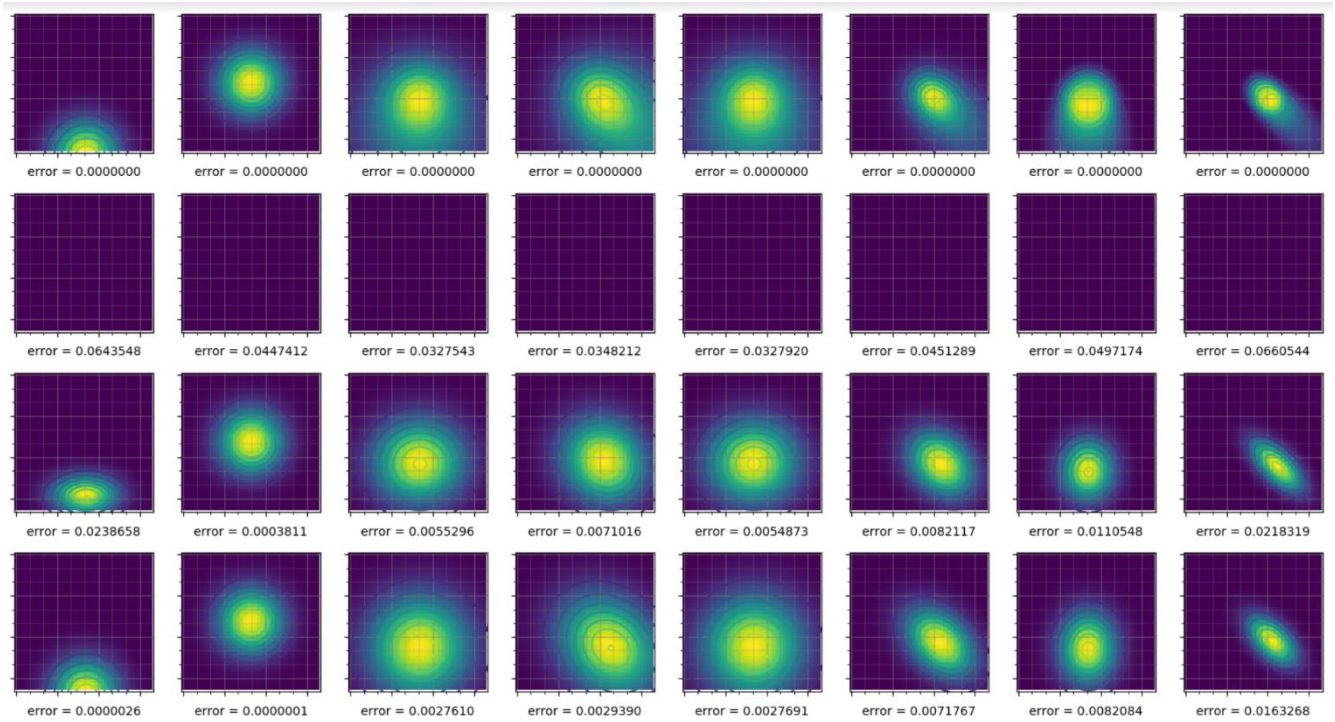


Figure 10 - Centroids and shapes of cells modelled in 2D using BFGS and our analytic approximation.

Our strategy for computing μ and M provides good results given the data. For data that is a perfect Gaussian, our weighted mean and covariances model the data exactly. Figure 9 compares this approach to a standard optimization technique in finding a multivariate-Gaussian-of-best-fit. From left to right, the models shown are (1) a symmetric Gaussian centered at an edge of the window; (2) a symmetric Gaussian centered in the center of the image; (3-5) various incarnations of a geometric skew-normal distribution [13],

$$\sum_{k=1}^{N} \frac{p(1-p)^{k-1}}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}k^{\frac{3}{2}}} \exp\left[-\frac{1}{2}k(x-k\mu)^{T}\Sigma^{-1}(x-k\mu)\right]$$

(6-8) various incarnations of a geometric skew-normal distribution with an added exponential parameter:

$$\sum_{k}^{N} \frac{p(1-p)^{k-1}}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}k^{\frac{3}{2}}} \exp\left[\left(-\frac{1}{2}k(x-k\mu)^{T}\Sigma^{-1}(x-k\mu)\right)^{\beta}\right]$$

From top to bottom, the images depict (1) the synthetic data constructed with the model; (2) the least-squares error attained when modeling the data with the zero-image; (3) the error attained when modeling the data with a multivariate Gaussian with parameters given by our analytic approach; and (4) the error attained when modeling the data with a multivariate Gaussian with parameters computed using BFGS optimization, seeded with the model derived from our analytic approach.

We find that the analytic approach closely models the data in all cases when most of the cell nucleus is contained within the window. With the exception of the first image, errors are not substantially different between the analytic approach and the iterative (BFGS) approach. Neither approach perfectly models the last image (8), which has both a skew and exponential parameter. In practice, cell shapes are rarely perfectly modeled by a Gaussian. By using our approaches to model data which is inherently skewed (eg. produced by a multivariate skew-normal distribution), we can gain an understanding of how modeling approaches handle data that is not perfectly Gaussian.

Section 4: A Simple Tracking Strategy

This model of cells as multivariate Gaussians gives rise to a natural tracking strategy.

Cells can have varying radii, and there is not necessarily a single scale at which one must compute the Laplacian in order to correctly identify all of the cells in an image. To find cells at varying scales, we compute a reconstruction of the data at multiple scales with varying values σ, and construct a scale-tree relation between successive scales. A blob found at scale s_i may in fact be contained within another larger blob found at scale s_{i+1}. It may itself contain multiple child blob, each found at scale s_{i-1}. Thus we represent a single *frame* of the dataset as a hierarchy of Gaussian blobs.

Let S_i contain all blobs at scale i. Then, a blob $B \in S_i$ is the child of blob $B' \in S_{i+1}$ if there is no other blob in S_{i+1} that is closer to B than B', using the Euclidean distance metric with respect to the centers of blobs.

Next, we draw a graph connecting blobs between adjacent frames in the 3D+t imageset. Blobs in successive frames are connected if their covariance ellipsoids (one standard deviation from the mean) intersect, regardless of scale. Now, we can use a graph traversal strategy to track cells between frames.

A source of uncertainty exists in choosing the particular scale at which a blob pertains to a particular nucleus. We note that a single cell nucleus will be associated with a single blob at a particular scale *as well as* all of the blob's descendents, which uniquely belong to the same cell. A cell C_i is therefore a collection $\{B_i\} \cup \{B_j : B_j \text{ is a descendent of } B_i\}$. This simplifies the tracking task, as tracking cell C_i becomes equivalent to tracking any of its constituent blobs $b \in C_i$.

An entire dataset consists of frames $F_1, \dots, F_T$. Each frame F_i consists of a collection of cells C_i, each cell consists of a series of scales $s_1, \dots, s_{\hat{s}}$, and each scale consists of a collection of blobs found at that scale, B_i.

A cell is a tuple (C_{pred}, C_{succ}, B), consisting of a predecessor, a set of successors, and a set of blobs. Our goal is to find the set of cells which constitute the entire experiment.

Maintain a set of *valid* blobs for each frame, initially consisting of all blobs in each frame. Denote the set of valid blobs in frame t as $valid(\widehat{F_t})$. Also set a maximum distance that cells can travel between subsequent frames. Call this M. This is a parameter that varies based on image resolution and the particular sample being imaged. In the data used in this experiment, M was set to 30 voxels.

We initialize the set of cells at frame 0 to consist of the set of blobs at the minimal scale. At each subsequent frame t, we proceed by finding successors for each cell in frame $t-1$ within distance M. Until each cell in F_t has a successor OR no such pairing can be found, find the closest pairing between cell C in frame F_{t-1} and blob B in frame $\widehat{F_t}$. $C, B = \underset{C \in F_{t-1}, B \in valid(\widehat{F_t})}{\operatorname{argmin}} \operatorname{distance}(C, B)$. If $\operatorname{distance}(C, B) > M$, then no other cell in frame F_{t-1} has a successor within distance M, and therefore has no successor in the succeeding frame, and we end the loop. If instead there is a closest pairing, then we mark the successor of C as B, and then invalidate all descendants of blob B as well as all blobs for which B is a descendant.

Finally, if there are any blobs which are not yet accounted for, then we mark these blobs as being born, and we add them to the subsequent frame. If there is a cell in the previous frame within M distance, then we mark this cell as being the predecessor of the cell, and mark this as being a possible cell division.

As a sanity check, this algorithm produces tracks that seem biologically plausible. More work should be done to verify correctness.

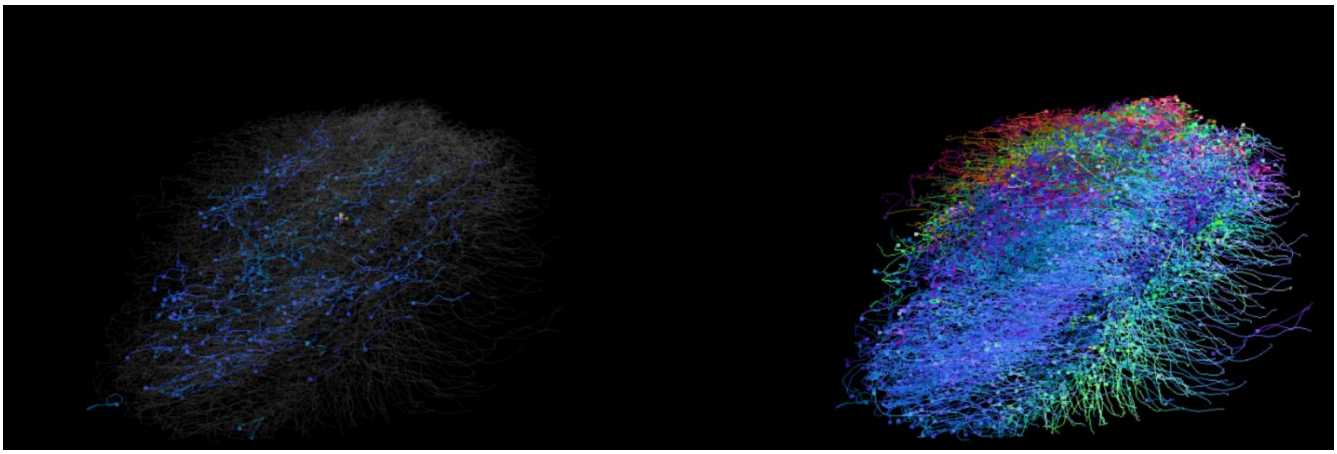


Figure 11 (left) a subset of axially-migrating tracks (right) set of tracks computed using this algorithm—the whole embryo, between 11 and 17 hours post-fertilization.

Section 5: Statistics on Tracks

Tracks are embeddable in an n-dimensional Euclidean space [14]

The output of a tracking algorithm is a set of tracks T, each following a single cell through the duration of a single run of the experiment. Because time-resolution is finite, each track T can be thought of as a list of discrete points in space over time, $T_i = \{p_{i1}, p_{i2}, \dots, p_{in_i}\}$. We notice that we can expand each point and in fact treat a track T_i as a point in an $n \times 3$ -dimensional vector space:

$$T_i \cong \begin{pmatrix} p_{i1x} & p_{i2x} & \cdots & p_{in_ix} \\ p_{i1y} & p_{i2y} & \cdots & p_{in_iy} \\ p_{i1z} & p_{i2z} & \cdots & p_{in_iz} \end{pmatrix} \cong \left(p_{i1x}, p_{i1y}, p_{i1z}, \dots, p_{in_ix}, p_{in_iy}, p_{in_iz}\right).$$

Immediately, we can see that this transformation forms a metric space with the Euclidean norm and the induced metric $d(x, y) = \|x - y\|_2$.

Define the naïve transformation

$$\phi_{naïve}: T_i \to \left(p_{i1x}, p_{i1y}, p_{i1z}, \dots, p_{in_ix}, p_{in_iy}, p_{in_iz}\right).$$

The naïve norm here gives us a rough indication of how closely a trajectory remains near to the origin $(0,0,0)$. However, because trajectories exist throughout the volume of the organism, and because the space-origin has no special significance, we attempt to construct another transformation ϕ_T which provides a translation-invariant norm.

Define

$$\phi_T: T_i \to \left(p_{i1} - p_{i1}, p_{i2} - p_{i1}, \dots, p_{in_i} - p_{i1}\right)$$

$$= \left(p_{i1x} - p_{i1x},\ p_{i1y} - p_{i1y},\ p_{i1z} - p_{i1z},\ \dots,\ p_{in_ix} - p_{i1x},\ p_{in_iy} - p_{i1y},\ p_{in_iz} - p_{i1z}\right).$$

For each point, we subtract the first point; and the construction is the same as with ϕ_{naive}. Now, the norm gives us an indication of the rough *distance away from the start* that the cell takes on its trajectory. For example, a cell which goes in a circle will have non-zero norm. The induced metric then becomes translation-invariant, but not scale- or rotation- invariant (non-rotation-invariance is desirable, because we want to recover information about the direction of cell migration). The canonical inner product

$$< T_1, T_2 > = \sum T_{1i} \cdot T_{2i}$$

gives us information about the orthogonality of trajectories: trajectories that coincide perfectly will have a high inner product; and trajectories that move in perpendicular directions will have low inner product. Thus, this construction is physically meaningful and trajectories form a Hilbert space.

Before performing computation, some strategy must be used to ensure that trajectories have the same length. For a fixed trajectory length ℓ, we simply resample each trajectory at equal intervals ℓ times, linearly interpolating between points. This method allows for easy computation, but information about the start- and end- timepoints of trajectories is lost.

Dimensionality reduction and clustering of tracks

Treating the space of trajectories as a Euclidean space, it is now possible to perform statistical methodology in the space of trajectories such as clustering and dimensionality reduction. We consider a set of tracks over 100 timepoints obtained with the algorithm described above.

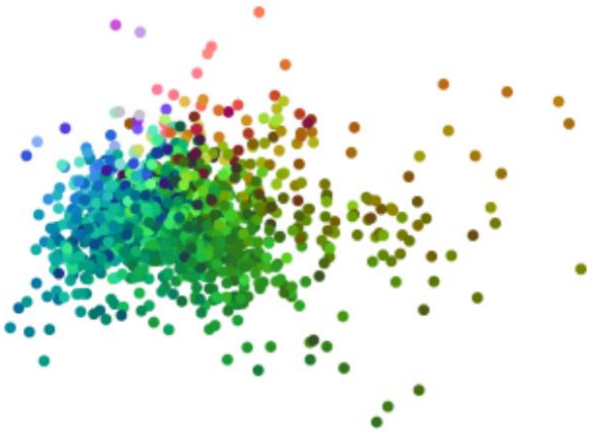


Figure 12 – Top two PCA-components on a set of tracks. PCA roughly separates tracks by color, ie. by overall direction.

Standard dimensionality reduction using PCA separates trajectories roughly by their overall direction of travel, which is denoted by color:

```
Color := track[n-1] – track[0]
Color := Color / norm(Color)
Color := (1.0 + Color)/2.0
```

Trajectories which move in the x+ direction are light red; in the x- direction are dark red; in the y+ direction are light green; in the y- direction are dark green; in the z+ direction are light blue; and in the z- direction are dark blue.

Locally-Linear-Embedding is also able to recover some structure about the manifold on which tracks lay, and in fact also separates tracks by their overall direction of travel. Figure 13 shows a set of trajectories when reduced to two dimensions using LLE. It produces an ‘L’ shape which seems to separate points by their direction of travel. Figure 13 depicts the LLE embedding in 2D as well as the 3D trajectories that each portion of the embedding corresponds to.

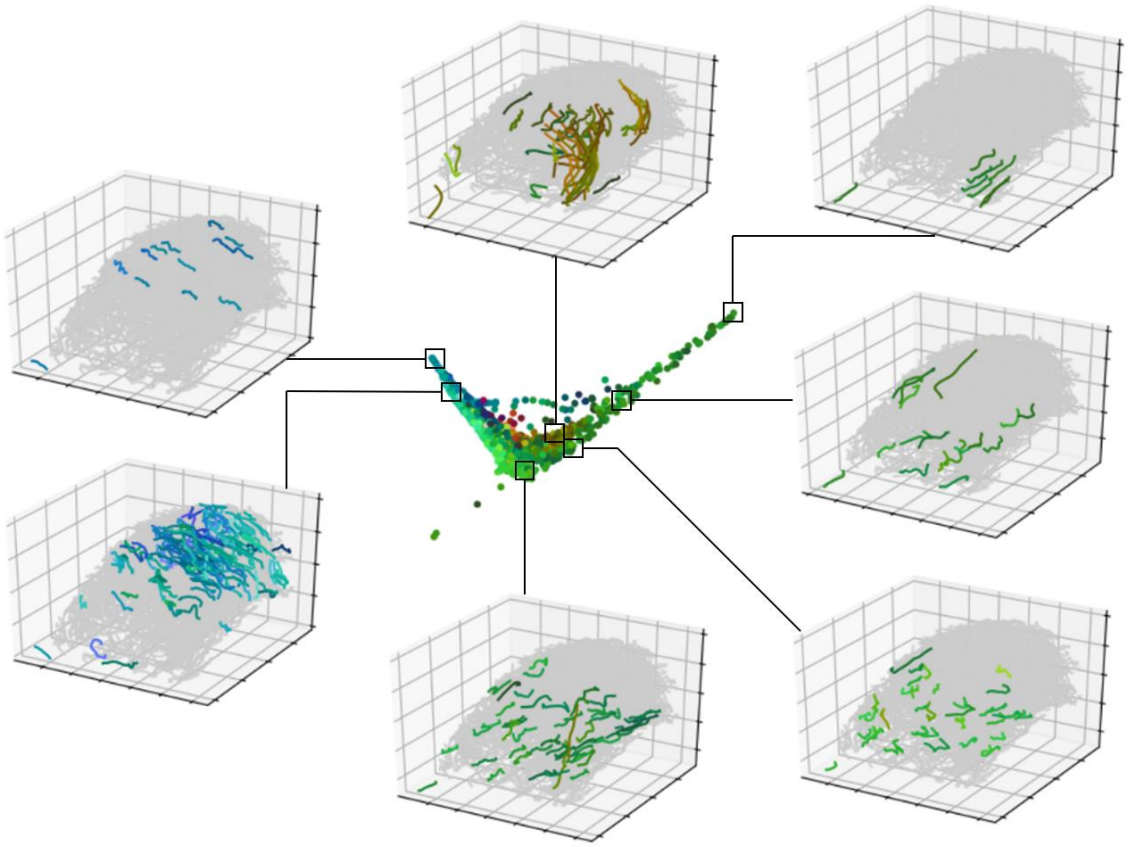


Figure 13 - Locally-Linear-Embedding also arranges tracks with relation to their direction of travel.

t-Distributed Stochastic Nearest Neighbor Embedding (t-SNE) is a dimensionality-reduction technique which attempts to embed a higher-dimensional space into a lower-dimensional one by preserving similarities between points as closely as possible, where the similarity between two points x_i and x_j in the embedding space is given by $p_{ij} = \frac{\left(1+\|y_i-y_j\|^2\right)^{-1}}{\sum_{k\neq l}(1+\|y_k-y_l\|^2)^{-1}}$, and the similarity of two points in the original space is given by the more complicated (but symmetric) relation $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2n}, p_{i|j} = \frac{\exp\left(-\|x_i-x_j\|^2\right)}{\sum_{k\neq l}\exp\left(-\|x_k-x_l\|^2\right)}$. t-SNE minimizes the KL-divergence between these distributions. t-SNE is better than other dimensionality-reduction techniques at visualizing data that lies on several different lower-dimensional manifolds [15], and is an ideal dimensionality-reduction technique for visualizing bundles of intertangled trajectories. Figure 14 visualizes MDS, Isomap and t-SNE when

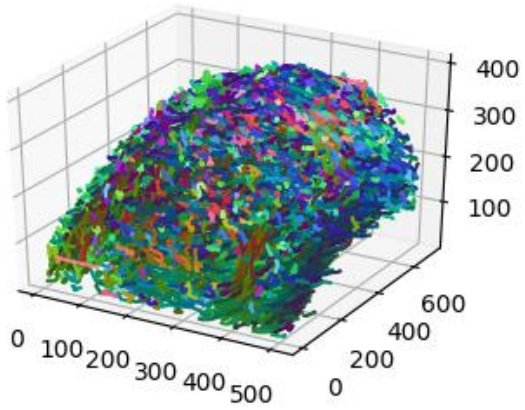


Figure 15 – Rendering of all trajectories, colored by direction. Trajectories can be seen over the dorsal surface of the Zebrafish, but this image is unenlightening.

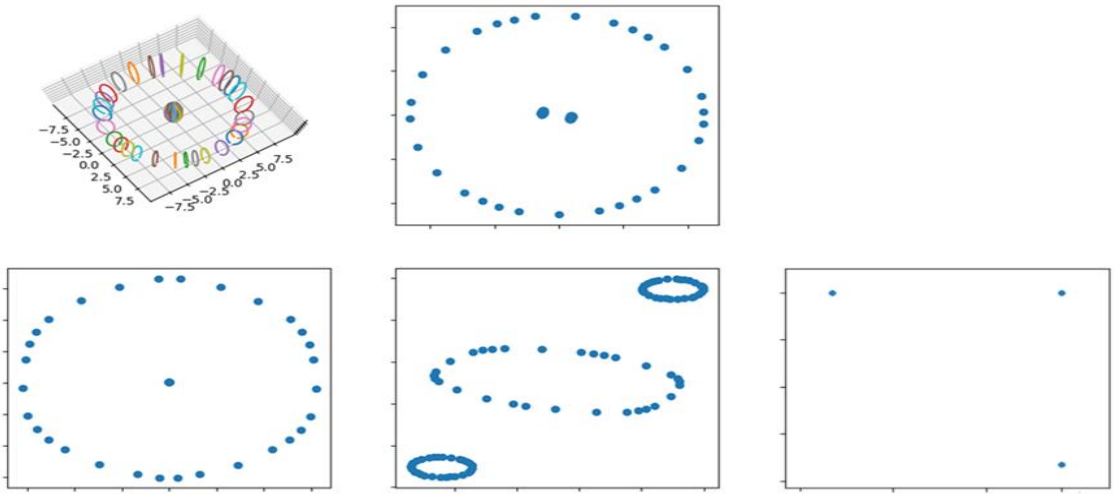


Figure 14 - (top left) Example trajectories consisting of a series of longitude lines on a sphere (half going N-S, and half going S-N), surrounded by circular orbits arranged in a torus, and their dimensionality-reduced models as produced by (top right) MDS; (bottom left) Isomap; (bottom middle) t-SNE; (bottom right) LLE.

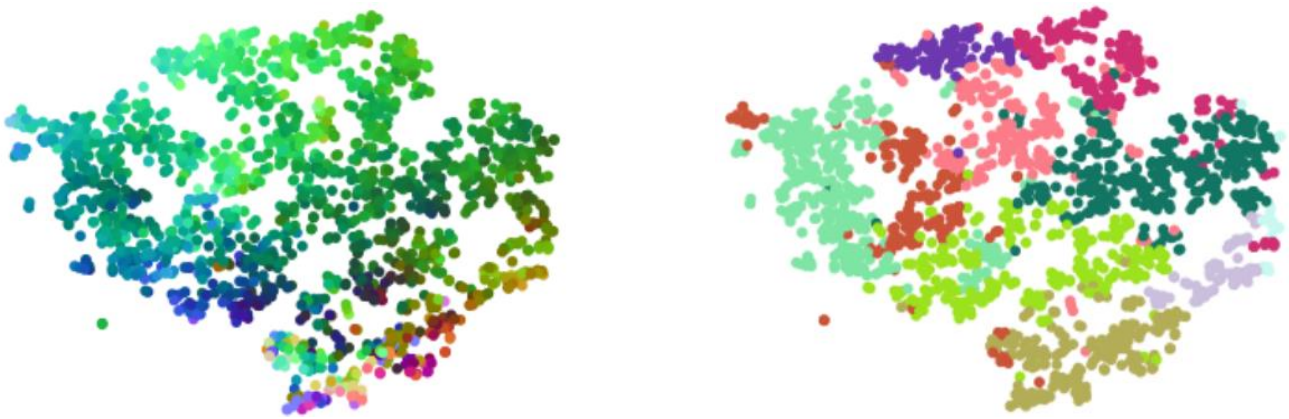


Figure 16 – A sample of the data visualized with t-SNE (left) and colored based on hierarchical clustering in the latent space of trajectories (right)

applied to an example dataset consisting of three types of trajectories. t-SNE does not preserve the geometry of the data, but does effectively cluster the types of trajectories present.

t-SNE produces visible clusters when applied to a dataset consisting cell trajectories. We can then identify these clusters using a clustering algorithm. In this paper, we use a hierarchical agglomerative clustering approach. Initially, we mark each point as being in its own singleton cluster. We then iteratively find the two clusters which, when merged, have the minimum intra-cluster variance, and then merge these clusters. We terminate the algorithm when N clusters remain, for some N. In the clustering performed in Figure 16, there are 10 clusters. We capture these clusters and then display a sample of the resulting

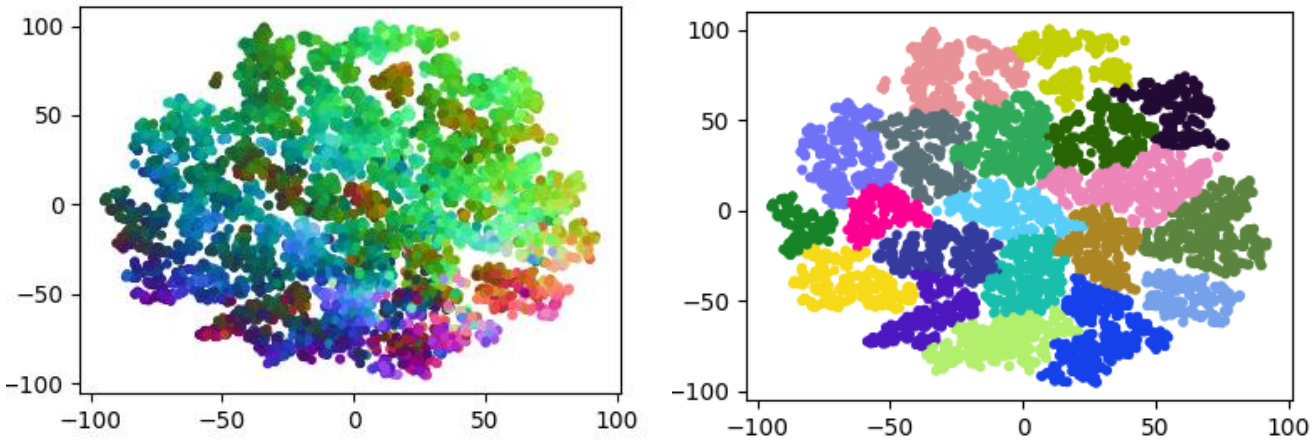


Figure 17 - Results of visualizing the data with t-SNE (left), and then clustering within the embedding space (right).

clusters in 3D. Figure 15 is a rendering of all cell trajectories in the developing embryo. This is rather unintuitive to understand; but when we view individual clusters, we can see that cell paths follow clear patterns (Figure 19). We can see that one group of cells migrates axially, while another group of cells migrates tangentially.

We may perform agglomerative clustering either in the latent space of trajectories or in the embedding space output by t-SNE (Figure 17). Conveniently, we find that clusters identified within the latent space closely resemble clusters generated by t-SNE (Figure 16). We may understand these clusters as trajectory phenotypes of various cells within the developing Zebrafish. Figure 18 visualizes 10 clusters of trajectories computed by performing clustering within the latent space. Clear similarities between trajectories within each cluster can be observed. These trajectories are taken from a small region around the center-of-mass of the zebrafish between 13- and 15.5- hours post fertilization.

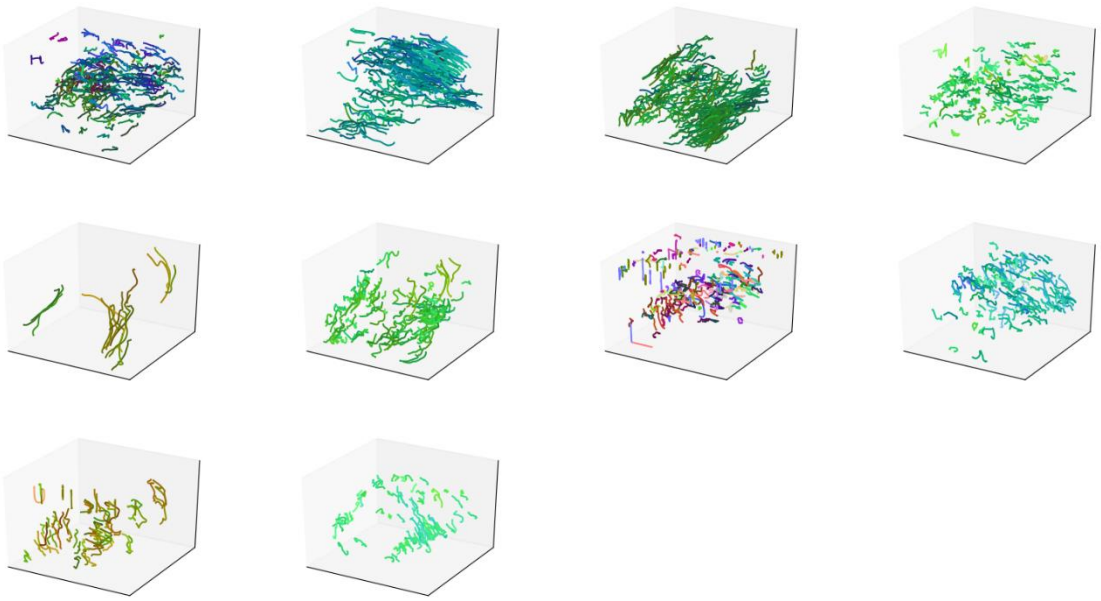


Figure 18 - Rendering of 10 trajectory phenotypes. Images 1 and 7 seem to contain mostly noisy trajectories.

Dimensionality reduction using splines

We notice that subsequent timesteps of a trajectory are highly correlated. To aid analysis, we perform another transformation on datapoints ϕ_{s_N} which transforms trajectories into cubic splines with N knots—which provides a compact representation of trajectories with minimal loss of information about the shapes of trajectories.

A 3D spline with N knots can be described with $(3)(4)N = 12N$ coefficients. On experimental data, we find the following relationship between N and the average approximation error of splines

$$E \coloneqq \frac{1}{|\text{tracks}|} \sum_{T \in \text{tracks}} \frac{1}{|T|} \sum_{p \in T} \left\| \text{spline}_{T,N}(p) - p \right\|$$

where tracks represents the set of tracks, $|\text{tracks}|$ is the total number of tracks, $|T| \text{for } T \in \text{tracks}$ is the number of points contained within the original track T obtained with the cell-tracking algorithm, $p \in T$ is a point along the track T, and $\text{spline}_{\text{T,N}}(p)$ is the equivalent point's approximation using a spline-approximation for the trajectory T with N knots.

To be precise, let the track T have a spline approximation $S_{T,N}(t)$ for $t \in [0,1]$, and call T_i the i^{th} point along track T, for $i \in 0, \ldots, |T|$. Then,

$$\text{spline}_{\text{T,N}}(T_i) \coloneqq S_{T,N}\left(\frac{\sum_{j=1}^{i} \left\| T_j - T_{j-1} \right\|}{\sum_{j=1}^{|T|} \left\| T_j - T_{j-1} \right\|} \right)$$

N (# of knots)	E (voxel error)
2	27.38
3	17.37
4	12.78
5	10.11
6	8.77
7	7.42
8	6.68

Acceptable error values are attained with $N > 4$, with 48 coefficients, where the voxel error is less than the diameter of a cell nucleus. Clustering conducted on splines produces results that are similar to those obtained with the previous method.

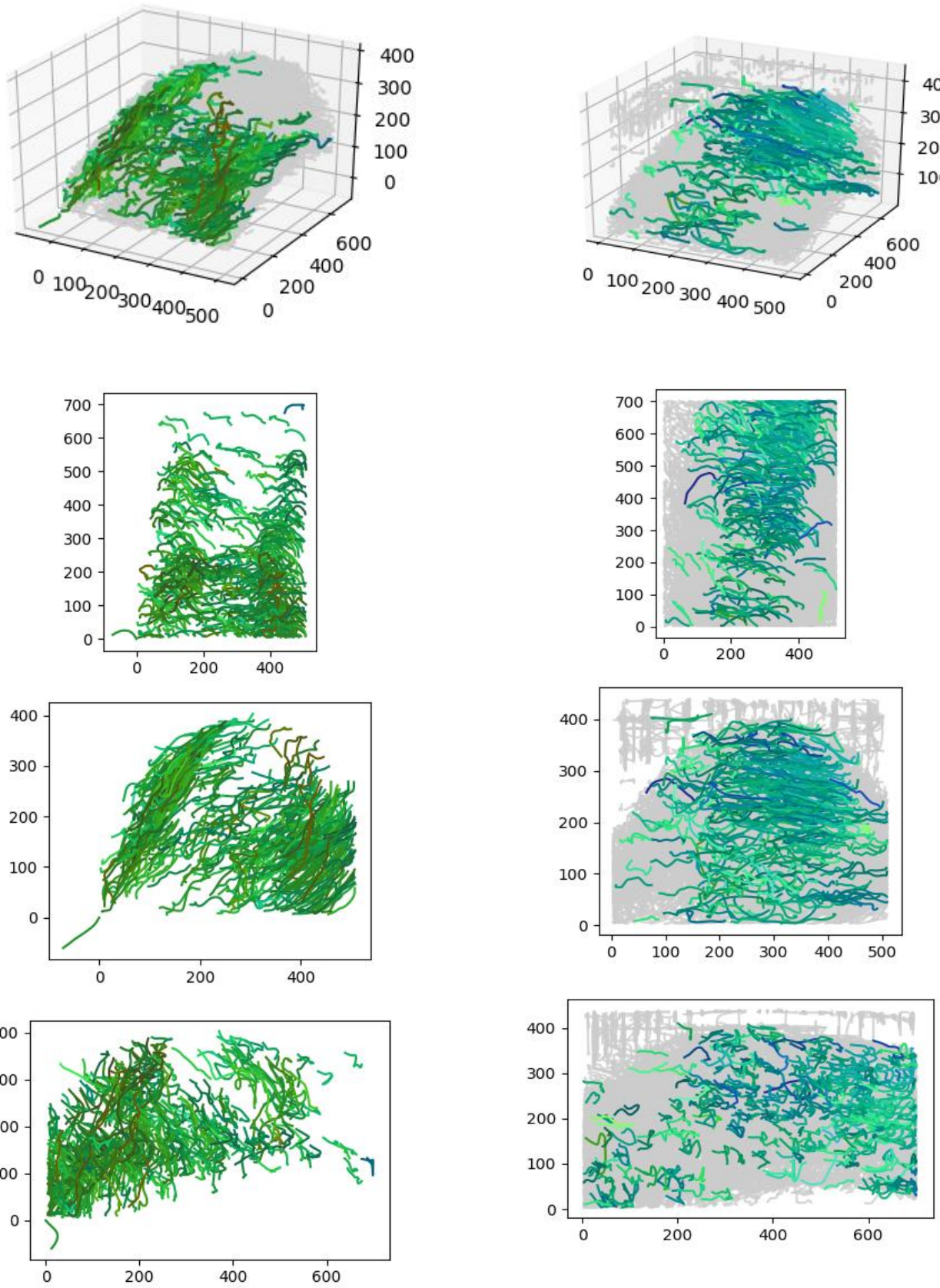


Figure 19 - Two clusters of trajectories captured with t-SNE and then agglomerative clustering, and their projections on the x-y, x-z, and y-z planes.

Conclusion and Future Work

In this paper, we have described two second-derivative filters which can act as preprocessing steps to denoise and segment blobs in a microscopy image. The Laplacian filter is effective in improving the accuracy of other cell-segmentation-and-tracking algorithms. By finding peaks and basins of attraction of this Laplacian filter, we presented a method to efficiently recover the shapes of cells and model images using multivariate Gaussian profiles. A simple tracking algorithm can also be constructed after modeling the data at different scales.

After tracks are found, it is possible to analyze them using regular statistical techniques such as clustering and dimensionality reduction, and we find that the clustered trajectories reveal salient patterns through embryological development. Future work involves assessing these methods for correctness (through manual comparison with human-tracked data) and interpreting trajectory data for physical significance.

It is likely that future directions involve machine learning approaches for cell classification and tracking. Even within a particular region of a particular organism, the shapes of cells are myriad, and a simple identification of cells by convexity produces a high volume of erroneous detections (ie. blobs detected at various scales, which detect various transient features of the cell rather than the cell itself). In particular, our segmentation algorithm fails when cells are too close to one another and the blobs from one merge with the blobs of another. The problem of cell-segmentation can easily be understood as one of classification, and a machine-learning model (for example, a neural network) can be trained to identify cells at the correct scale respective to a certain setting of the resolution of the microscope. Cell-tracking can similarly be described in a way that is solvable with machine learning methods. The limiting factor is the collection or generation of sufficient quantities of labeled training data.

Appendix A

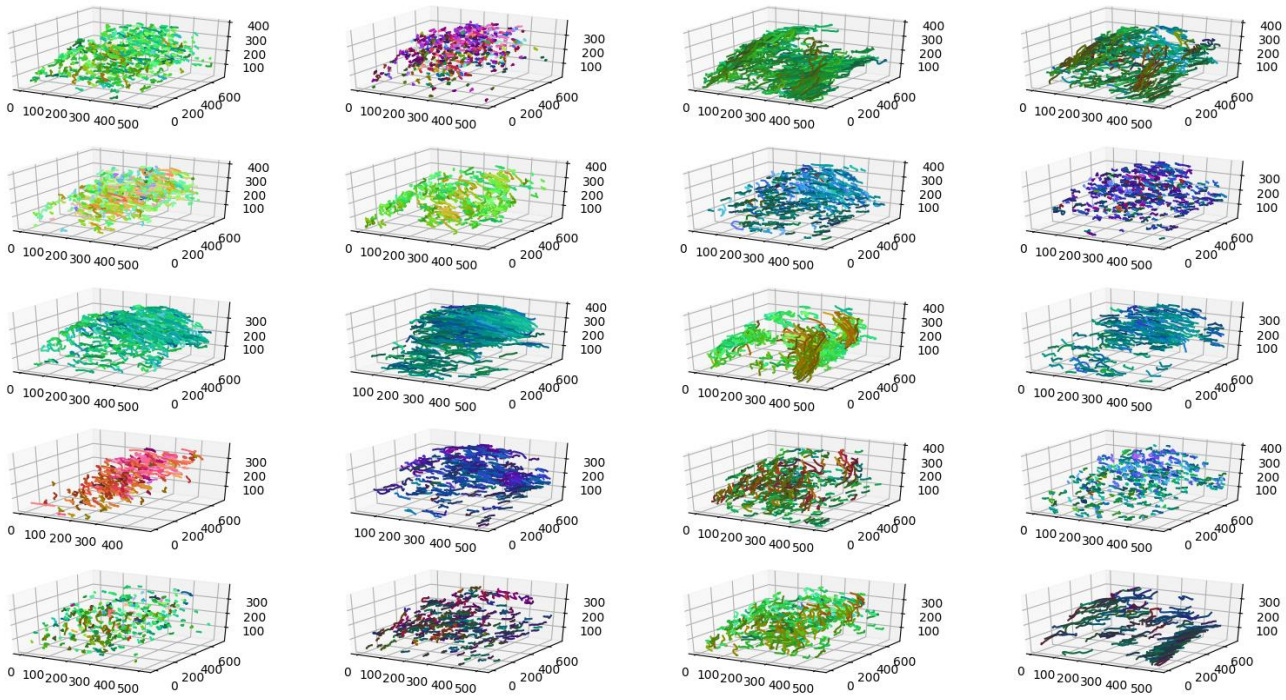


Figure 20 – Renderings of 20 trajectory phenotypes, corresponding to clusters identified in Figure 17.

Appendix B

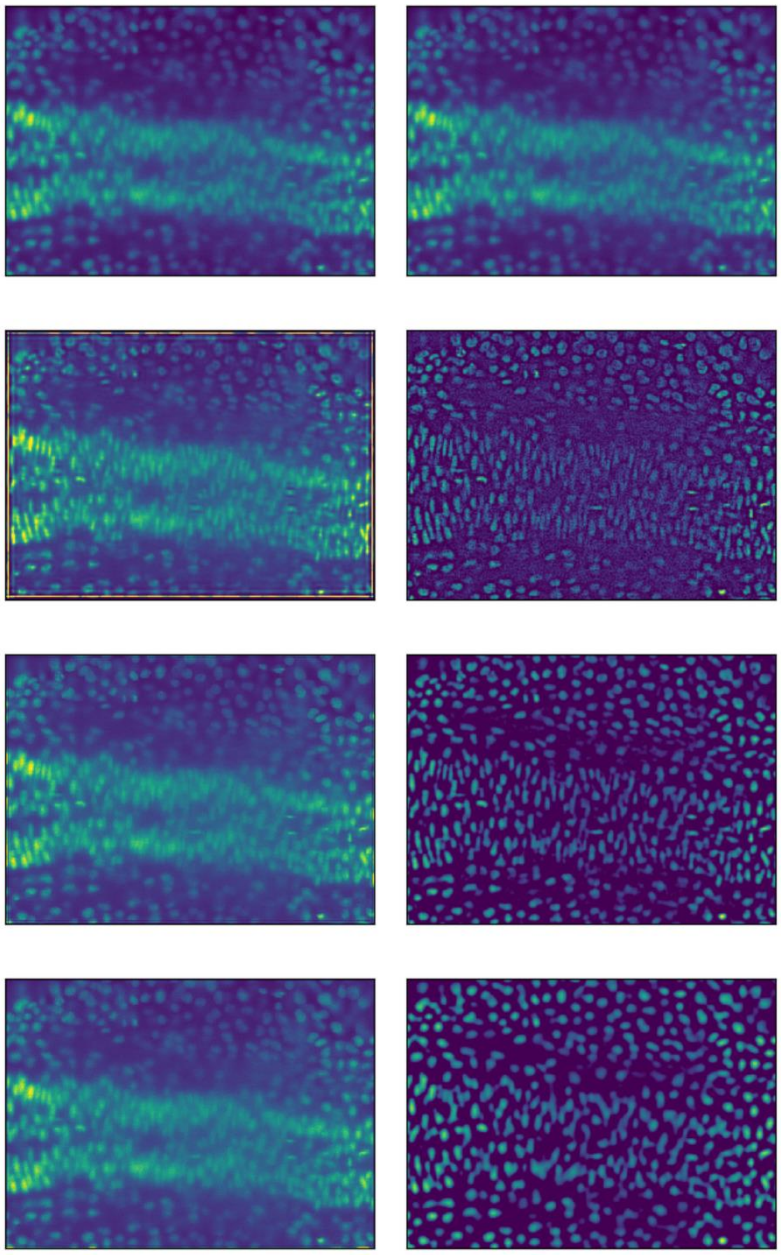


Figure 21 - A comparison of the results of deconvolution with a Gaussian kernel (left) and the clipped Laplacian operator (right) with varying sigma = {1, 3, 5}. The original image is atop each column. Deconvolution is more sensitive to the scale of the Gaussian kernel. The Laplacian offers cell centers reliably at various scales.

For finding cell nuclei, the Laplacian is also preferable to deconvolution because it can be performed at a single scale over the entire image. Deconvolution must be performed with different point-spread-functions because light-scattering is nonconstant within the image.

Appendix C

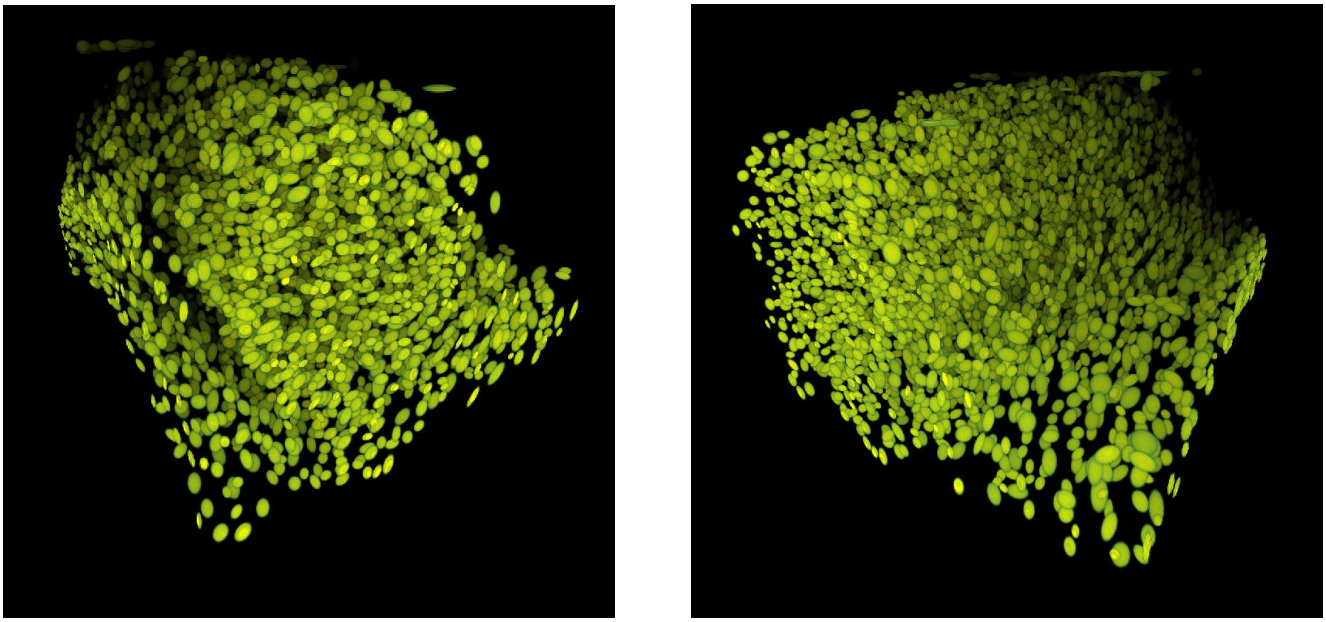
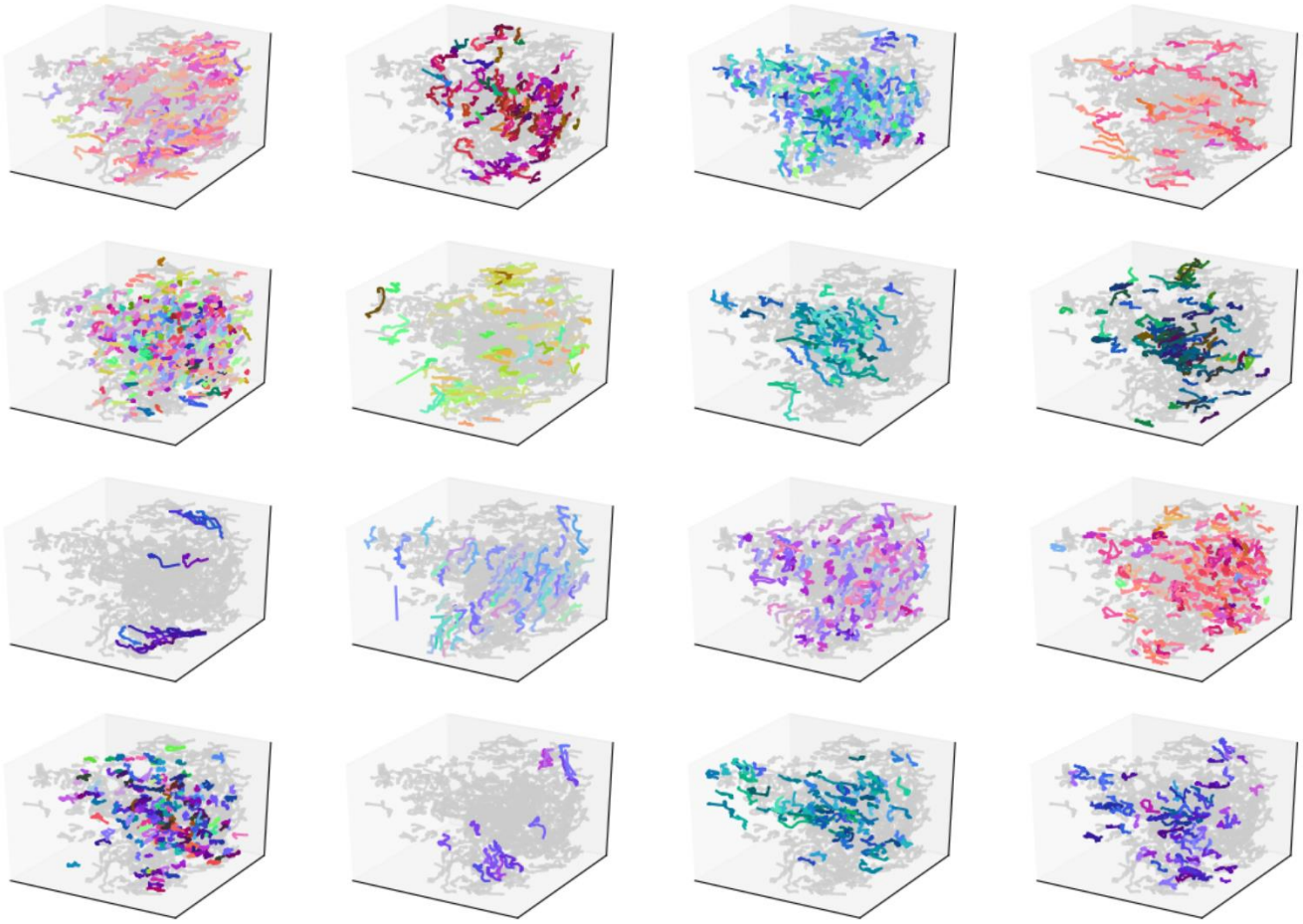


Figure 22: 3D Rendering of blobs found in the Zebrafish embryo 13 hours post fertilization, when imaged in LSFM and modeled using our algorithm. The Gaussians are replaced with ellipsoids in this rendering.

Appendix D



Visualization of trajectories over 100 timepoints on another dataset of Zebrafish trajectories taken between 13 and 15.5 hours post-fertilization, near the midline of the embryo. The general shape of the Zebrafish can be seen from the grey background. The dorsal end faces the camera, and the caudal end of the Zebrafish is to the left. Image 9 (left-to-right, 1-16) identifies two symmetric groups of tracks that migrate cranially.

Acknowledgements

It was originally Dr. Gordon Kindlmann who introduced me to the ideas that turned into this thesis project, and for this I offer him the utmost thanks. I would like to thank Dr. Yali Amit for his advice, guidance and mentorship through the completion of my project; Dr. Victoria Prince for many insightful conversations and suggestions regarding the biological significance of my work; and Dr. Eric Jonas for his advice and perspectives on the computational biology task I had at hand. Of course, none of the work in this paper could have been possible without the research conducted by the Prince Lab, and in particular the research conducted by Dr. Ana Beiriger in her studies of neuronal migration in Zebrafish. All of the microscopy data used in this paper were collected by her. I hope that the ideas in this paper may be useful to others interested in trajectory visualization, cell-tracking, and embryology.

References

1. R. S. Fischer, Y. Wu, P. Kanchanawong, H. Shroff, and C. M. Waterman, “Microscopy in 3D: a biologists toolbox,” *Trends in Cell Biology*, vol. 21, no. 12, pp. 682–691, 2011.
2. E. G. Reynaud, U. Kržič, K. Greger, and E. H. K. Stelzer, “Light sheet-based fluorescence microscopy: More dimensions, more photons, and less photodamage,” *HFSP Journal*, vol. 2, no. 5, pp. 266–275, 2008.
3. P. A. Santi, “Light sheet fluorescence microscopy,” *Journal of Histochemistry and Cytochemistry*, pp. 129–138, Feb. 2011.
4. R. Tomer, K. Khairy, and P. J. Keller, “Light sheet microscopy in cell biology.,” *Cell Imaging Techniques*, pp. 123–137, Aug. 2012.
5. C Wolff, J Tinevez, T. Pietzsch, E. Stamataki, B. Harich, L. Guignard, S. Preibisch, S. Shorte, P. Keller, P. Tomancak, A. Pavlopoulos, “Multi-view light-sheet imaging and tracking with the MaMuT software reveals the cell lineage of a direct developing arthropod limb,” *Elife*, Mar. 2018.
6. Ulman et al., “Analysis Published: 30 October 2017 An objective comparison of cell-tracking algorithms,” Nature Methods, vol. 14, pp. 1141–1152, 2017.
7. F. Amat, W. Lemon, D. P. Mossing, K. Mcdole, Y. Wan, K. Branson, E. W. Myers, and P. J. Keller, “Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data,” Nature Methods, vol. 11, no. 9, pp. 951–958, 2014.
8. J. Li, F. Xue, and T. Blu, “Fast and accurate 3D PSF computation for fluorescence microscopy,” Journal of the Optical Society of America, May 2017.
9. B. Zhang, J. Zerubia, and J.-C. Olivo-Marin, “Gaussian approximations of fluorescence microscope point-spread function models,” *Applied Optics*, vol. 46, no. 10, p. 1819, 2007.
10. M. Schwager, D. M. Anderson, Z. Butler, and D. Rus, “Robust classification of animal tracking data,” *Computers and Electronics in Agriculture*, vol. 56, no. 1, pp. 46–59, 2007.

11. M. S. Fazli, R. V. Stadler, B. Alaila, S. A. Vella, S. N. J. Moreno, G. E. Ward, and S. Quinn, “Lightweight and Scalable Particle Tracking and Motion Clustering of 3D Cell Trajectories,” 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2019.
12. M. Moradi, E. Pebesma, and J. Mateu, “trajectories: Classes and Methods for Trajectory Data,” *Journal of Statistical Software*, vol. VV, no. II.
13. D. Kundu, “Multivariate geometric skew-normal distribution,” Statistics, vol. 51, no. 6, pp. 1377–1397, 2017.
14. J.-G. Lee, J. Han, and K.-Y. Whang, “Trajectory clustering,” Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD 07, 2007.
15. F. Pascal, L. Bombrun, J.-Y. Tourneret, and Y. Berthoumieu, “Parameter Estimation For Multivariate Generalized Gaussian Distributions,” IEEE Transactions on Signal Processing, vol. 61, no. 23, pp. 5960–5971, 2013.
16. P. Duggirala and D. Sheehy, “When Can We Treat Trajectories as Points?,” Canadian Conference on Computational Geometry, Aug. 2018.
17. L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, Nov. 2008.
18. M. Guo, Y. Li, Y. Su, T. Lambert, D. D. Nogare, M. W. Moyle, L. H. Duncan, R. Ikegami, A. Santella, I. Rey-Suarez, D. Green, A. Beiriger, J. Chen, H. Vishwasrao, S. Ganesan, V. Prince, J. C. Waters, C. M. Annunziata, M. Hafner, W. A. Mohler, A. B. Chitnis, A. Upadhyaya, T. B. Usdin, Z. Bao, D. Colón-Ramos, P. L. Riviere, H. Liu, Y. Wu, and H. Shroff, “Rapid image deconvolution and multiview fusion for optical microscopy,” Nature Biotechnology, 2020.
19. A. Beiriger, "Origins And Migration Of The Octavolateral Efferent And Facial Branchiomotor Neurons In Zebrafish," Ph.D dissertation, Biological Sciences Divison, The University of Chicago, 2020.