Spring 2022

Quantifying Membrane Topology at the Nanoscale

Zachary Ryan Connerty-Marin
Yale University Graduate School of Arts and Sciences, zachcm@gmail.com

Abstract

Quantifying Membrane Topology at the Nanoscale

Zachary R. Connerty-Marin

2022

Changes in the shape of cellular membranes are linked with viral replication, Alzheimer's, heart disease and an abundance of other maladies. Some membranous organelles, such as the endoplasmic reticulum and the Golgi, are only ~ 50 nm in diameter. As such, membrane shape changes are conventionally studied with electron microscopy (EM), which preserves cellular ultrastructure and achieves a resolution of $2\ \mathrm{nm}$ or better. However, immunolabeling in EM is challenging, and often destroys the cell, making it difficult to study interactions between membranes and other proteins. Additionally, cells must be fixed in EM imaging, making it impossible to study mechanisms of disease. To address these problems, this thesis advances nanoscale imaging and analysis of membrane shape changes and their associated proteins using super-resolution single-molecule localization microscopy.

This thesis is divided into three parts. In the first, a novel correlative orientation-independent differential interference contrast (OI-DIC) and single-molecule localization microscopy (SMLM) instrument is designed to address challenges with live-cell imaging of membrane nanostructure. SMLM super-resolution fluorescence techniques image with $\sim 20\ \mathrm{nm}$ resolution, and are compatible with live-cell imaging. However, due to SMLM's slow imaging speeds, most cell movement is under-sampled. OI-DIC images fast, is gentle enough to be used with living cells and can image cellular structure without labelling, but is diffraction-limited. Combining SMLM with OI-DIC allows for imaging of cellular context that can supplement sparse super-resolution data in real time.

The second part of the thesis describes an open-source software package for visualizing and analyzing SMLM data. SMLM imaging yields localization point clouds, which requires non-standard visualization and analysis techniques. Existing techniques are described, and necessary new ones are implemented. These tools are designed to interpret data collected from the OI-DIC/SMLM microscope, as well as from other optical setups.

Finally, a tool for extracting membrane structure from SMLM point clouds is described. SMLM data is often noisy, containing multiple localizations per fluorophore and many non-specific localizations. SMLM's resolution reveals labelling discontinuities, which exacerbate sparsity of localizations. It is non-trivial to reconstruct the continuous shape of a membrane from a discrete set of points, and even more difficult in the presence of the noise profile characteristic of most SMLM point clouds. To address this, a surface reconstruction algorithm for extracting continuous surfaces from SMLM data is implemented. This method employs biophysical curvature constraints to improve the accuracy of the surface.

Quantifying Membrane Topology at the Nanoscale

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Zachary R. Connerty-Marin

Dissertation Directors: Joerg Bewersdorf & David Baddeley

May 2022

Acknowledgments

I am deeply indebted to many people in my life for making the research described in this document possible.

I was lucky enough to have two advisors, Dr. Joerg Bewersdorf and Dr. David Baddeley, who were exceptionally generous with their time, resources and expertise. Dr. Bewersdorf coached me through technical, political and personal problems, and helped me grow as both a researcher and as a person. Dr. Baddeley has reviewed and critiqued almost every line of code I wrote, and offered insights on every optical or experimental design I pitched for the past five years, which helped me improve my technical skills beyond what I thought was possible.

My committee members, Dr. Megan King and Dr. Michael Murrell, were enthusiastic about my projects, provided helpful feedback and made sure I was getting the support I needed throughout my Ph.D.

Dr. Michael Shribak regularly met with me to teach me concepts behind his OI-DIC system and how to perform the engineering and measurement tasks required to understand the activity of optical elements.

My lab mates over the years have been critical to my success, and I'm happy to call many of them my friends. In alphabetical order, thanks to Yujin Bao, Dr. Andrew E. S. Barentine, Dr. Arunima Chaudhuri, Dr. Kenny Kwok Hin Chung, Dr. Edward Courvan, Lukas Fuentes, Michael Graff, Dr., Xiang Hao, Dr. Kevin Hu, Phylicia Kidd, Dr. Dong-Ryoung Lee, Dr. Xiangzhou Lao, Mark Lessard, Dr. Yang Li, Shaocong Liu, Hannah-mariam Mekbib, Julia Neuwirth, Sonja Piehler, Dr. Ons M'Saad, Dr. Lena K. Schroeder, Dr. Florian Schüder, Dr. Lin Shao, Dr. Helen Mengyuan Sun, Tobias Stenz, Dr. Yuan Tian, Dr. Mary Grace Velasco and Dr. Yongdeng Zhang. Andrew, Michael, Mark, Phylicia, Ons, Lukas, Florian and Ed were as much fun (or more) outside the lab as in.

I will always be grateful I was able to go on this journey with Dave O'Connor and

Dr. Jenette Creso. Thanks to Anthony Alves, Dr. Robert Arthur, Robert Brouillard, John Cummons, Lila Lyons, Logan Gerchman, Lucas Higgins, Patrick McGrath, Evan McLean, Kyle Monahan, Dr. Andre Khalil, Anthony Panciocco, Julie Panciocco, Mackenzie Schlosser, Ian Shea, David Sherman, Melissa Stuart, Peter Strand, Eugene Wentworth, Dr. James Kent Wallace and Steven Wilkinson for their years of support and friendship. My apologies to anyone I've missed here, your absence is a result of my poor memory and not an indication of the effect you've had on my life.

It is impossible to show enough appreciation for my parents, Christine and David Connerty-Marin, and my sibling, Evey, for their love and support. Thanks to my Aunt Dr. Deb Thompson and Uncle Ken Thompson, Aunt Dr. Rachel Perla and Uncle Jeremy Marin, and my Aunt Lisa and Uncle Chuck for their humor and support. My grandparents, Larry and Teddi Marin and Eileen and Paul Connerty have given me unconditional love and encouragement for my entire life. I would not have made it through this program without them.

Finally, I'd like to thank my cats for letting me put my head on their fluffy little tummies whenever I was stressed.

For my mother and father

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A futon works as a bed when flat and as a couch when folded. Similarly, a cell's function is linked with its shape [1–3]. This is true all the way down to the nanoscale, where local differences that are 500 to 5000 times smaller than the width of human hair determine if a subcellular membrane is best suited for protein synthesis, transport, calcium intake, or another purpose [4–8]. These tiny changes in shape impact the health of a cell and consequently the health of its host organism. For example:

- Positive-strand RNA viruses, such as the brome mosaic virus, carve pockets, vesicles and other curved structures in and from the endoplasmic reticulum membrane. These structures create ideal environments for viral RNA synthesis and protect intermediate replication structures (e.g. double-stranded RNA) from host cell immune responses [6, 9].

- Parkinson's disease destabilizes mitochondria cristae junctions, which connect the inner and outer leaflets of the mitochondrial membrane. Disruption of these junctions causes the mitochondria to take on an onion shape, disrupting oxidative phosphorylation. Consequently, the mitochondria does not produce sufficient energy for its host cell [8, 10].

- In dilated cardiomyopathy (DCM), the left ventricle (LV) of a human heart stretches and thins, making it difficult to pump blood. This increases demand for natriuretic peptides (NPs) to regulate plasma volume and pressure homeostatis. To secrete NPs faster, Golgi increase in density, shrink, and become ellipsoidal. Thus, nanometer-scale changes in Golgi shape are at least correlated with increased LV size [11].

These discoveries—and many others concerning subcellular shape—were made using electron microscopy (EM), which is ideal for showing nanoscale morphological changes at fixed points in time, but not ideal for studying mechanisms of disease.

EM samples preserve three-dimensional cellular structure via freezing, and image with a resolution of 2 nm or better [12, 13]. While EM imaging can be performed on unlabelled samples, sometimes specific proteins are identified via immunolabelling with gold nanoparticles. However, immunolabelling requires fixation methods that often destroy cellular structures, and gold nanoparticle locations must be extracted from among other, unstained cellular features [13, 14]. Immunolabelled or not, structures of interest must be identified via manual annotation, which is never perfect, or via machine learning algorithms trained on example segmentation provided by manual annotators [15, 16]. Multiple annotators, human or computer, are required to achieve accurate segmentation of subcellular organelle structure in EM images.

Fluorescence microscopy, by contrast, uses fluorescent labels to target specific molecules. These labels provide colorful signals that can be isolated with filters, allowing structures to be identified by their label without annotation [17]. However, standard fluorescence techniques (e.g. widefield and confocal microscopy) have a minimum resolution of 250 nm, set by the diffraction limit (see Section 2.1.1) [18].

Super-resolution fluorescence techniques, stimulated emission depletion (STED) and single-molecule localization microscopy (SMLM), circumvent the diffraction limit by controlling the switching rates, from off (non-emitting) to on (emitting) and vice versa,

of fluorescent molecules [18–23]. STED microscopy can image at the same rate as a confocal microscope, but images live cells with $\sim 40\ \mathrm{nm}$ resolution[1] over a smaller field of view [24, 25]. SMLM is temporally limited, but can image live cells with $< 20\ \mathrm{nm}$ precision [26]. Neither method can simultaneously image more than 3 and 4 color channels due to overlapping spectra of usable dyes [24, 27].

Correlative light and electron microscopy (CLEM) combines EM and fluorescence microscopy to achieve both specific labelling and detailed imaging of subcellular structure [28]. CLEM imaging can involve live-cell and even super-resolution fluorescence imaging prior to fixation [28]. However, cellular context can only be imaged in EM post-fixation. Modern expansion microscopy techniques also provide correlative ultrastructure and fluorescence imaging, but samples must be fixed [29].

In order to visualize the mechanisms of disease in (+)-stranded RNA viruses, Parkinson's disease, dilated cardiomyopathy, and other membrane-associated illnesses, it is necessary to image subcellular membrane dynamics and membrane-associated protein locations at the nanoscale in live cells. In particular, to study the influence of proteins on changes in membrane shape, both membrane-associated proteins that may influence structure and the membrane itself must be imaged at the size-scale of the clusters of membrane-associated proteins. This means imaging live cells with $10 - 20\ \mathrm{nm}$ resolution, which requires SMLM.

As with any relatively new technique, there are still some engineering problems that need to be solved to do useful SMLM imaging in live cells. Due to SMLM's slow imaging speeds, most cellular movement is under-sampled, and faster imaging of cellular context is needed to supplement the super-resolution data [30]. SMLM yields point data, which requires non-standard visualization and analysis [31]. SMLM's resolution reveals labelling discontinuities, which make identifying the continuous shape of a membrane difficult [32].

[1] This number is up for debate in the field, see reference [24].

This dissertation describes the development of tools that address these problems. Improvements are made in parallel for both optics and data analysis capabilities. Chapter 2 expands upon the problems stated above in detail and provides the majority of background information needed to understand the remaining chapters. In chapter 3, I propose a live-cell compatible correlative SMLM and label-free quantitative phase contrast microscope, which is designed to maximize the available information about membrane dynamics at the nanoscale. Chapter 4 is a reproduction of my *Nature Methods* paper, which describes a software framework for visualizing and analyzing data collected from the microscope described in Chapter 3 and other SMLM setups. Chapter 5 is a draft of a manuscript describing a novel algorithm for extracting the nanoscale shape of subcellular membranes from SMLM data. Chapter 6 summarizes the work described in this dissertation, its significance, and offers suggestions and predictions for related future work.

Chapter 2

Background

2.1 Fluorescence microscopy

Fluorescence microscopy is a popular light microscopy technique for studying biological phenomena due to its ease of use, specificity, and compatibility with live-cell imaging. Molecules of interest in a cell are identified by endogenous expression of fluorescent proteins, or via an organic dye conjugated to DNA, a peptide strand, an antibody, or another molecule with an affinity for a specific DNA sequence, protein or lipid. Fluorescent molecules absorb incident light and then emit light at a lower energy than absorbed. By using spectral filtering, it is possible to separate the emitted light from the incident light, and to separate the emitted light from multiple, uniquely-colored labels simultaneously. This allows researchers to study the placement and interaction of multiple specific proteins in a single cell [17, 33].

This section covers concepts relevant to this thesis and is far from comprehensive. The interested reader is encouraged to consult [33] and [34] for more information.

2.1.1 Image formation and the diffraction limit

Look up into night sky at the right time of year and it is possible to see the Big Dipper. Like all constellations, the Dipper is made up of stars, each appearing as a point in the night sky. These points collectively form an image, in this case of a ladle.

It is useful to think of every image as a constellation: an image is a sum of light emanating from point sources. A point source is small enough that the behavior of an optical system does not vary significantly across its diameter, i.e. any optical aberrations present are consistent across the point source. The response of an optical system in the image plane to a point in the object plane is called the system's *point-spread function (PSF)*, and an example is shown in 2.1. Considering images as the sum of light emanating from point sources means image formation can be treated as a linear system [34].

Unlike a point source, which is effectively of infinitely small size, the PSF of an optical system spreads out in the far field. Two objects must be at least r_{xy} apart in the observation plane to be distinguished from one another, where

$$r_{xy} = \frac{0.61\lambda}{NA}, \tag{2.1}$$

λ is the wavelength of light emanating from the point source and NA is the effective numerical aperture of the optical system. Objects must be r_z apart along the optical axis to be distinguished from one another, where

$$r_z = \frac{2\lambda}{NA^2}. \tag{2.2}$$

r_{xy} and r_z define the lateral and axial resolution of an optical system, respectively [34]. Points closer to one another than r_{xy} laterally or r_z axially will appear as a single spot in an image. For a well-aligned optical system, the minimum distance between points that

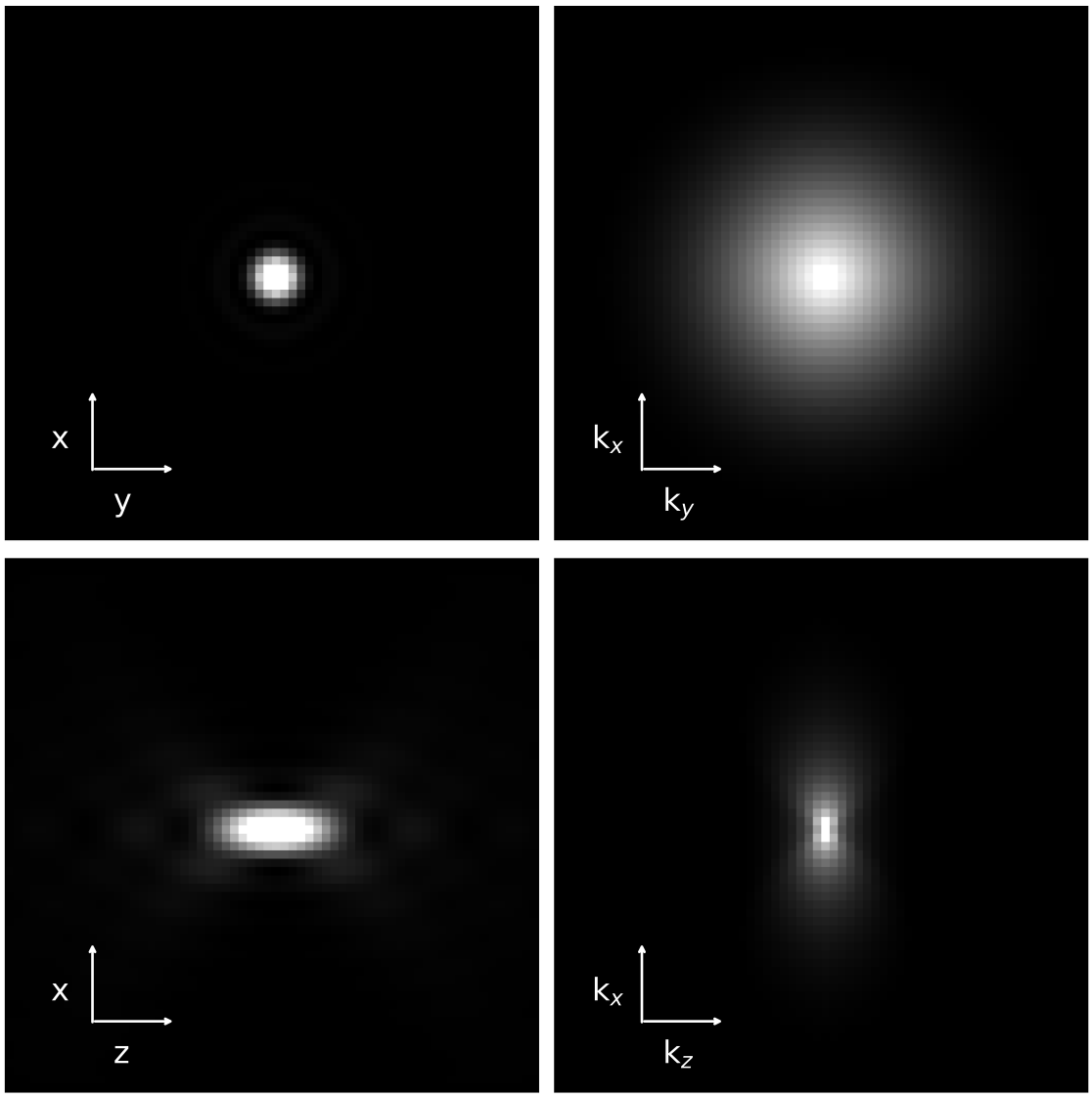


Figure 2.1: Left, A naive simulation of a widefield point-spread function, here for illustrative purposes only. **Right,** The point-spread function represented in frequency space is called the optical transfer function.

can be resolved is set by r_{xy} laterally and r_z axially. The smallest possible size of r_{xy}, 250 nm, is called the *diffraction limit* [18].

2.1.2 Single-molecule localization microscopy

In normal fluorescence microscopy, all fluorescent molecules hit with an exciting light beam turn on at the same time. In single-molecule localization microscopy (SMLM), each dye has a probability of turning on at any point in time. This stochastic parameter is tuned, with imaging buffer, exciting power and wavelength, so that it is rare that two fluorescent molecules that are within a diffraction limit of one another turn on at the same time. Well-separated fluorescent spots are fit to yield molecular position with high precision. By summing the positions, a super-resolved image—that is, an image with resolution smaller than the diffraction limit—is produced [35].

In the case of a well-aligned optical system, the PSF appears as an Airy pattern, and the central lobe of a PSF of a fluorescent molecule can be approximated with a Gaussian. It is possible to calculate the position of a fluorescent molecule as the mean $\hat{\mu}$ of the fit Gaussian, and the position error by the square root of the variance σ^2 of the Gaussian. For a PSF fit with a Gaussian, $\sigma_x \approx \sigma_y \approx r_{xy}/2.355$ and $\sigma_z \approx r_z/2.355$. The error of the mean term scales as

$$\hat{\mu}_e - \mu_e = \frac{\sigma_e}{\sqrt{N}} \tag{2.3}$$

where μ_e is the true point position in axis $e \in \{x, y, z\}$, and $N \in \mathbb{N}$ is the number of photons in the integral of the fitted Gaussian. In theory, it is possible to use this technique to distinguish molecules $< 1\ \mathrm{nm}$ apart [35]. In practice, because of molecular spatial fluctuations, camera noise, vibrations and because dyes emit a limited number of photons, it is usually possible to distinguish molecules $5 - 20\ \mathrm{nm}$ apart. This is an order of magnitude better than any conventional, diffraction-limited microscope can achieve, and more importantly is at the size scale of a cluster of proteins.

2.1.3 The difficulty with labeling density

In fluorescence microscopy, molecules of interest (proteins, DNA, lipids; $2 \sim 20$ nm) are labeled with fluorescent molecules (proteins, dyes; $1 \sim 5$ nm), each smaller than the resolution of the optical system, and a single one of these fluorescent molecule is considered a point source. The image produced shows a proxy for the distribution of the molecule of interest. Fluorescent molecular labeling is usually dense enough that all labels are closer to one another than the system resolution, and so fluorescent signal often appears as a continuous object as in Figure 2.2, left.

The appearance of continuity is an artifact of the imaging system. It is perfectly possible that the cellular object is continuous, but the proxy is almost certainly discontinuous. Each molecule of interest in the object gets an individual label (or multiple labels), and labels usually have different properties than their targeted molecules. Unless the labels also self-assemble in the same way into a continuous structure, the labels will be disconnected when their target molecules are connected. Furthermore, it is extremely difficult to get one-to-one, label-to-target labeling, and some molecules of interest are usually unlabeled [32]. Labeling efficiency can vary greatly and depends on several factors including sample fixation, antibody quality, dye quality, target protein, location of target protein, and more[1].

Labelling sparsity is quite visible in super-resolution microscopy techniques, where distinguishing individual labels is possible. An example of this is shown in Figure 2.2, right, where SMLM imaging of an endoplasmic reticulum (ER) membrane marker appears discontinuous, despite the membrane being a continuous structure.

[1]Producers of fluorescent labels have entire sections of their websites devoted to solving some of these labeling problems. For example, see `https://www.thermofisher.com/us/en/home/life-science/protein-biology/protein-labeling-crosslinking/fluorescent-protein-labeling.html` or `https://www.sigmaaldrich.com/US/en/applications/protein-biology/protein-labeling-and-modification`.

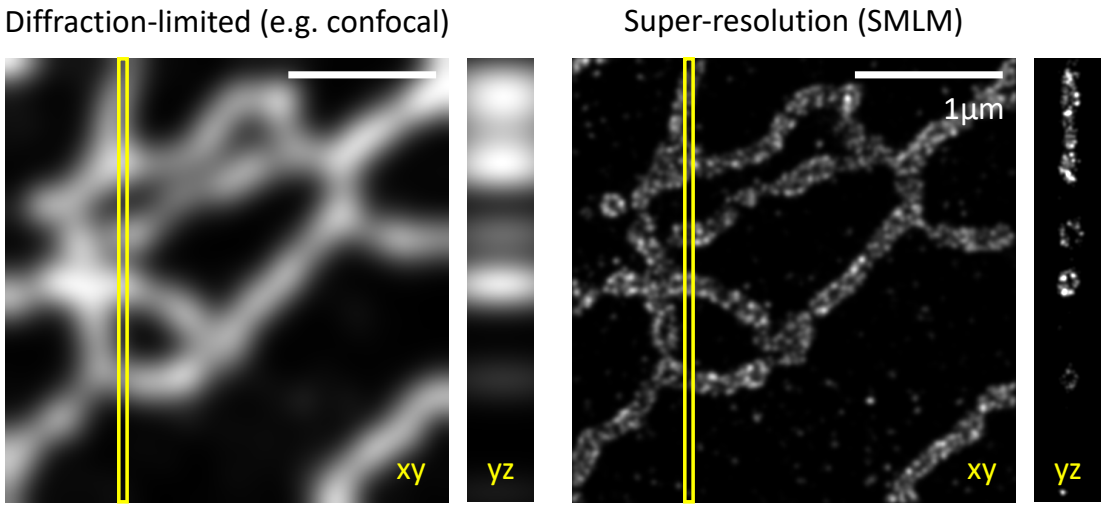


Figure 2.2: Left, The diffraction-limited image of the ER appears continuous. **Right,** Super-resolution imaging of the same structure reveals discontinuous labeling. This sample of a cell labelled with mEmerald-Sec61β, an ER transmembrane protein, was prepared by Dr. Lena Schroeder and imaged by Dr. Yongdeng Zhang on the 4Pi SMS microscope [36].

2.1.4 Live-cell challenges in SMLM

Turning molecules on and off has the drawback of extending imaging time. A sufficiently dense SMLM image of a large structure, such as the endoplasmic reticulum, can take minutes to hours to collect depending on imaging conditions. Any movement will appear as an artifact, blurring the summed image. Splitting the localizations by frame will provide access to only a few localizations at each moment in time. As such, the structure of the imaged object will be under-determined on a per-frame basis, exacerbating any labeling inefficiencies. It is possible to acquire an SMLM image of a live ER within 1 second, but the laser intensities required to do so are so harsh that they destroy the cell after this time, disallowing investigation of future movement [37].

Live-cell imaging is fundamentally limited by the available SMLM dyes. Most dyes are not cell permeable and many require toxic buffers [35]. Non-toxic, single color, live

cell SMLM acquisitions have run for as long as an hour, but it takes $15 - 30$ seconds to acquire enough localizations to determine structure [38]. Two-color live cell acquisitions of membranes has at best be done in 5 seconds intervals for no more than 15 seconds total before the sample bleaches, the cell dies, or both[2] [26]. As such, an additional trick is needed to image membrane dynamics along with membrane-associated protein locations at a native rate and for an acceptable duration.

As mentioned earlier, correlative EM imaging has had success in supplementing detailed structural information lacking in fluorescence imaging. Like EM, quantitative phase imaging (QPI) techniques provide cellular context without the need for specific labelling. Unlike EM, QPI methods are gentle enough to be used in live-cell imaging, making them excellent candidates for correlative live cell imaging.

2.2 Quantitative phase imaging

In fluorescence microscopy, light incident on a sample is separated from the light emitted by fluorescent molecules. In transmission microscopy techniques, such as quantitative phase imaging (QPI), we measure all of the light that passes through a sample [34, 39]. In this case, point sources are phase objects, which delay—but do not significantly change the intensity of—the transmitted light. Phase imaging techniques are designed to convert the relative delay of light passing through different parts of a sample into detectable changes in intensity on a camera [34, 40, 41].

QPI methods generate an optical path difference (OPD) map of a transparent sample via interference. The optical path length (OPL) of a beam traveling a distance d_s through

[2]Furthermore, I have been told that live-cell SMLM dyes, "just don't work," by colleagues who have tried them: Dr. Joerg Bewersdorf, Phylicia Kidd, Lukas Fuentes and Dr. Kevin Hu.

a sample of refractive index n_s is given by

$$\text{OPL} = n_s d_s.$$

To be quantitative, all light delay in a sample must be measured relative to a reference beam with known delay. As such, QPI methods measure the OPD between the distance traveled in the reference beam d_r that passes through a refractive index n_r and the distance traveled through the sample beam,

$$\text{OPD} = n_s d_s - n_r d_r.$$

In temporally-shifted QPI, widefield images are taken at two or more different phase delays, either at different points in time or different delays in a variable phase retarder, and these images are interfered with an on-axis reference beam and then combined to create an OPD map. In spatially-shifted phase imaging, an off-axis reference beam is interfered with the beam passing through the sample along the optical axis and an image of the interference pattern is taken. Spatially-shifted phase images often require computational post-processing. Some QPI methods can take temporally-shifted or spatially-shifted images of planes passing through a sample at multiple angles and/or z-positions, and combine these to create a 3D OPD map. These are called tomographic QPI techniques [39].

Many of these QPI techniques, such as those that require an off-axis beam or are tomographic, require specialized optical layouts that make combination with standard microscope layouts difficult. Among those that can easily be combined with standard layouts, spatial light interference microscopy is based on phase contrast microscopy, and therefore its signal is scattered by thick samples, and gradient light interference microscopy only provides an anisotropic OPD map of a sample [42, 43].

Orientation-independent differential interference contrast microscopy (OI-DIC), a vari-

ant of differential interference contrast (DIC) microscopy, is a QPI technique that can easily be combined with a standard fluorescence microscope path with no substantial modifications [44–46]. OI-DIC is a spatially-shifted, tomographic QPI technique that provides an isotropic optical path length map of a sample, and, because it is based on DIC, images through thick samples.

2.2.1 Orientation-independent differential interference contrast (OI-DIC) microscopy

DIC microscopy is a common interference microscopy method used to increase the contrast of transparent samples. In DIC, two beams displaced from one another along an axis, called the shear direction, by less than the width of the microscope PSF are interfered to create a shadow cast image that represents the derivative of the OPD along the shear direction [44]. Examples of DIC images taken at two different shear directions are shown in Figure 2.3 left and middle.

Whereas DIC measures a phase shift along a single direction, in OI-DIC, DIC images are taken at two orthogonal shear angles, e.g. shown in Figure 2.3, left and middle. This creates a basis of phase shift directions, allowing for reconstruction of isotropic phase shifts. To create a quantitative OPD map, multiple images are taken at each shear angle with varying, known phase delay (bias) $+\Gamma$, $-\Gamma$, and optionally 0 nm [45]. The images are combined to generate an OPD map. An example is shown in Figure 2.3, right [46]

Because OI-DIC is a widefield technique that uses the full NA of the system, it can achieve as low a resolution as the diffraction limit. The sensitivity of the OPD map is reported as ~ 0.5 nm [46]. In order to achieve this sensitivity at the desired size scale, the phase delta Γ can be used to adjust the OPD at which the greatest contrast is achieved

[3] https://www.fishersci.com/shop/products/fisher-chemical-permount-mounting-medium-2/p-121849

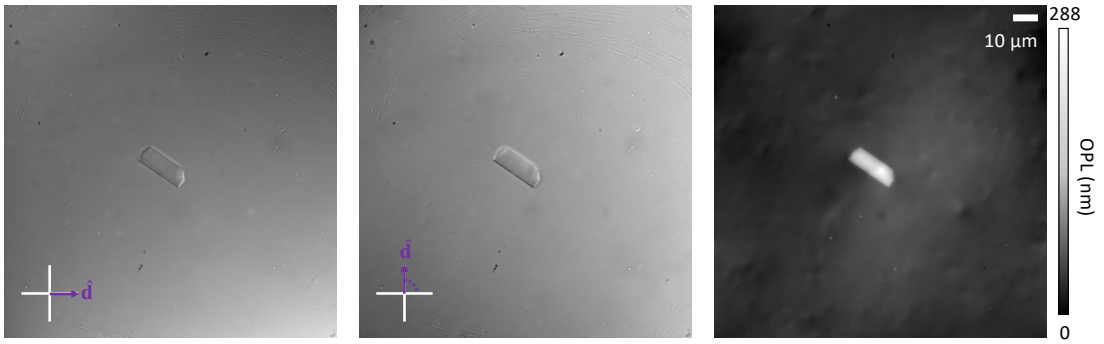


Figure 2.3: Left, DIC image of a 7 µm glass rod in Permount[3]**Middle,** DIC image of the same rod, taken at a shear angle orthogonal to the shear angle of the first DIC image. **Right,** An OI-DIC OPD reconstruction created from six DIC images, an image with delays $-\Gamma$, 0, and $+\Gamma$ nm at each of the orthogonal shear directions. This sample was prepared by Dr. Michael Shribak and imaged by Yujin Bao.

[44].

Current OI-DIC methods employ liquid crystals to shift between shear directions and biases, and imaging time is limited by the liquid crystal settling time. In theory, this can be as low as $5 - 20$ ms for bias change[4] and < 5 ms for changes in shear direction[5], yielding an imaging time of $35 - 125$ ms, which should be sufficiently fast for sampling subcellular movement. Microtubules, for example, remodel at a rate of $10 - 470$ nm $\cdot$ s^{-1}, and remodeling of the endoplasmic reticulum can be sampled with 250 ms integration time [47, 48].

[4] https://www.meadowlark.com/liquid-crystal-variable-retarder-p-94?mid=2

[5] https://www.meadowlark.com/binary-liquid-crystal-rotator-p-135?mid=2

2.3 Visualizing and extracting information from localization data

Most image data, including QPI images, are stored as an array of pixels or voxels, where a voxel is the 3D equivalent of a pixel[6]. SMLM data is stored as a point cloud that contains coordinates, coordinate uncertainties, photon counts from the fit that yielded each point, and other fitting data. A *point cloud* is a collection of coordinates, and associated with additional properties, in an affine space[7]. Unlike voxel-based data, point data has no physical size or regular spacing, and features of a point cloud must be identified by the spatial relationships between points. The common ways to extract this information follow.

- **Spatial statistics** convert the distances between points into descriptive metrics.
- **Rasterizing** the point cloud converts the data to a voxel-based image. From here, standard image analysis techniques may be used.
- **Surface reconstruction** algorithms connect or fit an approximate surface to the points to create a continuous representation of the object described by the point cloud. Curvature, packing parameters and other shape metrics can be computed from these representations.

These techniques must be implemented to make sense of localization data. There are many disparate implementations of these algorithms, and a few packages that bring multiple analyses together for use with SMLM (see supplement of [49] for a comprehensive list). To the author's knowledge, none of these packages are set up for correlative analysis

[6] A pixel represents a value on a regular grid in 2D space, and can be thought of as a square. A voxel is imagined as a cube.

[7] Here, we use Euclidean space.

with other imaging modalities, and only one, unmaintained package contains a surface reconstruction algorithm [50].

2.3.1 Efficiently visualizing and interacting with localization data

In image analysis and microscopy, it is always helpful to visualize the image collected to generate and answer hypotheses. In localization microscopy, this means not only looking at raw frames coming off of a camera, but displaying the resulting localization point cloud and allowing for interaction via translation, rotation and zoom.

Point clouds can be denoised and quantified via clustering and other applications of spatial statistics. Denoising and clustering often require users to manually set parameters based on their judgement [49, 51–53]. As such, point cloud visualization must quickly update after these processes for visual verification of the accuracy of the clustering process[8].

Once the point cloud is sufficiently filtered, spatial statistical methods can be used to construct metrics and partitions of space based on point cloud density [55, 56]. Depending on the metric, the point cloud can be re-colored to visually represent, for example, the distance from each point to a cell's nucleus [49].

2.3.2 Rasterizing

There are many established techniques for studying biological phenomena in voxel-based images [57–59]. As such, it can be helpful to convert point clouds into 2D and 3D image formats for further analysis. Established rendering techniques include binning the point cloud by a regular or adaptive grid and scaling pixel intensity by the number of points

[8]Rendering point clouds in 3D and manipulating them in real time is a well-established problem in computer graphics, and largely solved for point clouds smaller than 100 million points thanks to research on visualizing lidar and radar data [54]. Most single-molecule localization data sets contain fewer than 100 million points. By pushing the list of points to a computer's graphical processing unit (GPU), it is possible to render all localization microscopy point clouds in real time. This allows for interactive investigation of point cloud shape.

in each bin, rendering every point as a Gaussian, and rendering an image scaled on the area of the triangles (or tetrahedron) in a Delaunay triangulation [60] (or tessellation) of the point cloud [61]. It is important that the image intensity scales with local density for proper visual interpretation and further image processing.

2.3.3 Surface reconstruction from point clouds

A *surface* mesh is a discrete representation of a continuous object in 3D by 2D polygons, usually triangles or quadrilaterals. Each polygon is responsible for representing a plane that approximates the object at that location. For example, Fig. 2.4 shows a sphere rendered as a coarse triangular surface mesh.

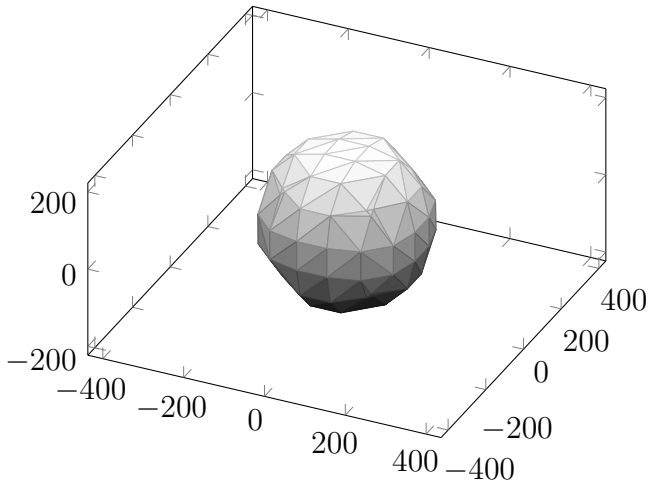


Figure 2.4: A sphere represented as a triangular surface mesh. Triangular meshes are common because three non-collinear points are the minimum needed to uniquely determine a plane in $\mathbb{R}^3$.

Surfaces are useful structures for representing complex objects, such as membranes. In most cases, operations on the surface of an object are equivalent to operations on its volume. With the exception of very simple objects (e.g. a cube), it takes less space to store a surface than a volume representation of the same object.

There are two main ways to extract a surface from point clouds:

- **Create a surface by directly triangulating points in the point cloud.** For example, the power crust method finds the Delaunay triangles that lie on the surface of a point cloud [62]. Some of these methods do not guarantee a manifold[9] surface [63].

- **Create a surface by binning the space occupied by the point cloud, assigning values to each bin via an indicator function, and then applying a voxel-based approach.** This method is more common. A regular grid or a K-d tree can be used to divide the 3D space containing points into discrete bins, with a indicator function value assigned to each bin based on point density within the bin [64]. More sophisticated techniques like screened Poisson reconstruction solve a smooth equation that fits a surface through the points [65]. An indicator (in this case, signed distance[10]) function measuring the displacement from the surface equation is constructed and evaluated at the desired resolution on a K-d tree. Once an indicator function is represented on a grid, marching cubes, dual marching cubes or another voxel-based algorithm can be applied on the indicator [66, 67]. The majority of these methods guarantee a manifold surface.

Once extracted, surfaces are sometimes further refined via fairing operations (see Section D.1) or by iteratively evolving the surface to a new shape, moving it subject to the gradient of a force (see Section D.3) [68]. This evolution can be used to achieve better estimates of subcellular shape (see Section D.2) [69, 70].

[9] A manifold surface has no singularities, i.e. every edge is connected to at most two triangles (see Appendix D for details).

[10] Signed distance functions (SDFs) represent an object as the distance from its surface, subject to the convention that the distance to a point below the surface is negative and above the surface is positive. For example, the SDF of a sphere of radius R is $f(\vec{\mathbf{p}}, R) = ||\vec{\mathbf{p}}|| - R$. SDFs are extremely compact representations of structures, and provide an easy way to determine interactions between surfaces and other objects.

Chapter 3

Combining label-free optical path length imaging with super-resolution fluorescence

This chapter describes the design and implementation of a novel correlative orientation independent differential interference contrast (OI-DIC) and single-molecule localization microscopy (SMLM) microscope, and proposes useful applications for such a system. Implementation details may be found in Appendix A. The latest iteration of this microscope is currently being built, and a manuscript is in preparation describing the microscope design and demonstrating its usefulness for identifying phase condensates.

3.1 Motivation

This microscope was designed for imaging subcellular membrane dynamics at the nanoscale in live cells. As discussed in Section 2.1.4, SMLM is temporally limited. It takes many localizations to form a good image of a membrane structure, and only a few localizations are available in each frame.

In live cells, the membrane structure may move quite fast. For example, it has been

shown that, in order to capture structural details, the endoplasmic reticulum should ideally be imaged at one frame every $250\ \mathrm{ms}$ or faster, although $1\ \mathrm{s}$ frames do provide a decent amount of structural information [48]. $100 - 250\ \mathrm{ms}$ are standard integration times for a single SMLM frame. As such, a simple sum of localizations from multiple frames will yield a blurry, inaccurate representation of a membrane.

However, the OI-DIC microscope we use, while diffraction-limited, can take a complete picture of a cell within $\sim 1\ \mathrm{s}$, and within $125\ \mathrm{ms}$ if the liquid crystals used are swapped out for faster versions, as discussed in Section 2.2.1.

Because of its relatively high sampling rate, as compared to SMLM, the optical path difference (OPD) map can be used to decide which localizations belong to a membrane in each frame, by checking if the localization appears in an area consistent with the OPD of lipids. A simulated example of this is shown in Figure 3.1.

By identifying which localizations are non-spurious, it is possible to create an uncertainty map of the position of a membrane by combining information from the SMLM and OI-DIC channels. This can be used to determine a surface representation of the membrane (see Chapter 5) in live-cell images, allowing investigation of dynamic curvature changes at the nanoscale, if only over small regions of the membrane at a time.

3.2 Optical design

Merging an OI-DIC microscope and a widefield SMLM microscope required two design steps: move the Nomarski recombination part of the OI-DIC path outside of the microscope body and insert a dichroic to divert the SMLM signal prior to recombination. This ensures the OI-DIC signal passes through both prism assemblies and the SMLM signal passes through neither, as shown in Figure 3.2.

The two main concerns about inserting a dichroic in the OI-DIC path were 1) the

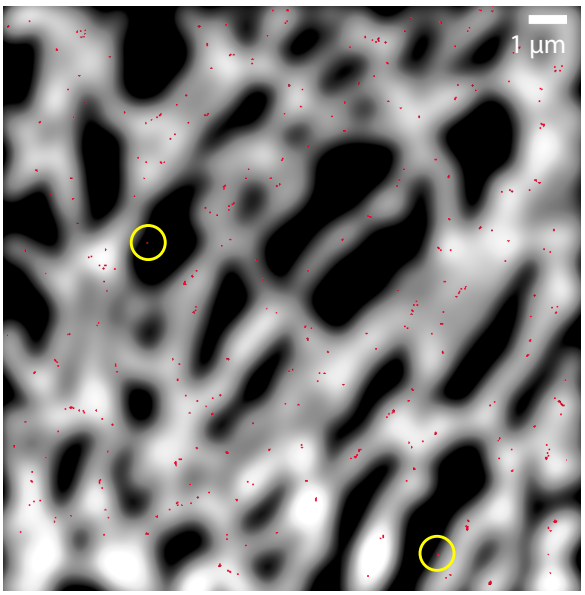


Figure 3.1: A simulation of single-molecule localization events overlaid on an OPD map. Real single-molecule localizations from frames 1-50 of an imaging session for mEmerald-Sec61β, an endoplasmic reticulum transmembrane protein, are overlaid on a simulated, diffraction-limited image of the endoplasmic reticulum created from the full set of frames. Spurious localizations are identified by yellow circles.

dichroic might be optically active 2) it might change the intensity of s- wave more than the p- wave or vice-versa. It was determined that the dichroic of choice, a Chroma ET546/22x notch filter, was not optically active and contrast was above 96 % percent if the filter is kept at an angle less than 20° off the optical axis (see Section A.2).

The OI-DIC setup images a field of view (FOV) of of 132×132 µm. Imaging time is limited by camera frame rate and liquid crystal settling time, with settling time the biggest factor in the current setup [46]. We use a partially coherent light source for the OI-DIC channel, which eliminates speckle, and the available intensity of our LED source is such that the system is not limited by shot noise.

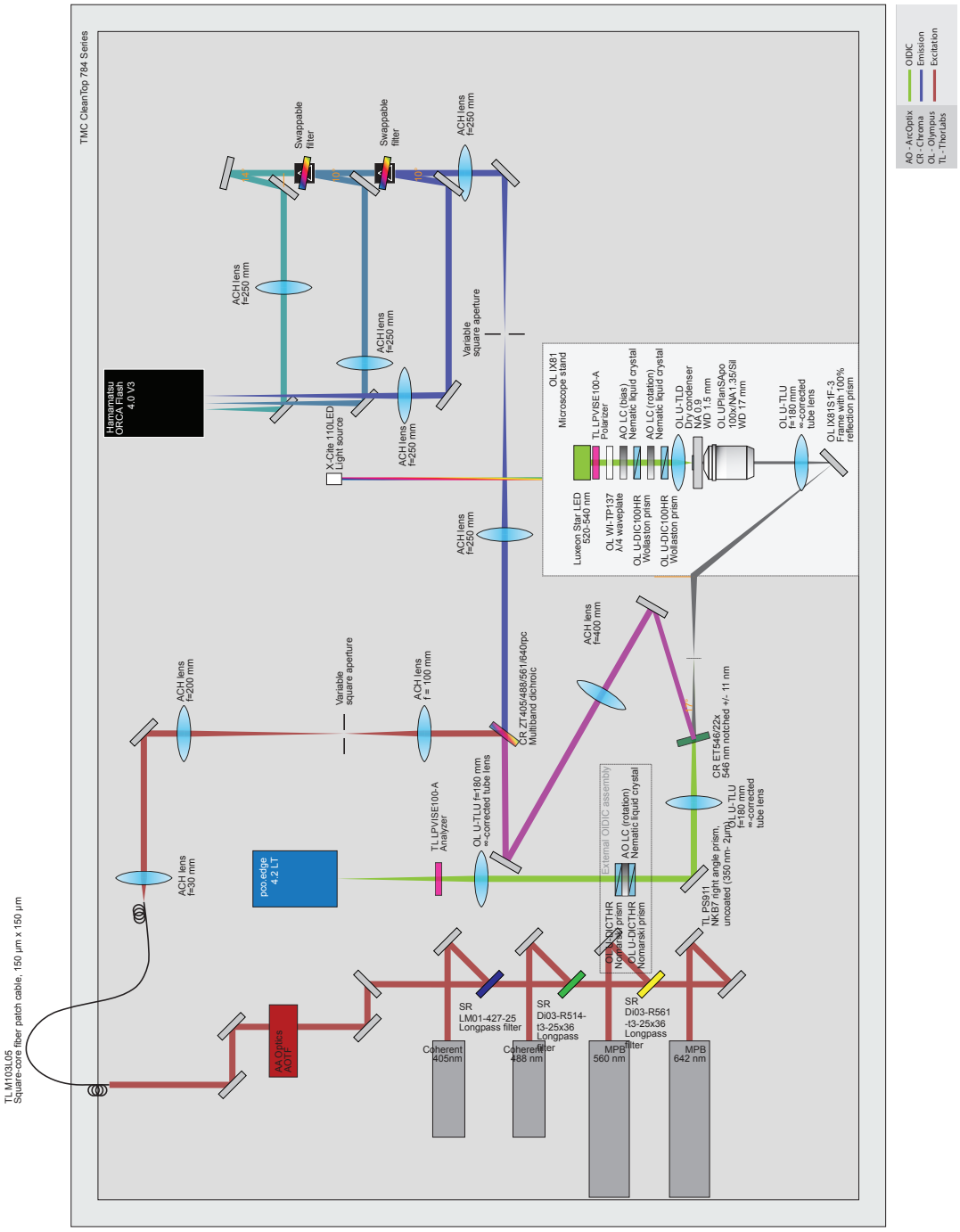


Figure 3.2: Optical layout of correlative OI-DIC and SMLM microscope.

In order to image multiple protein interactions, the widefield SMLM setup was designed to image up to 3 color channels at the same time. The ideal filters for splitting standard color channels of interest, namely to image dyes excited at $488\ \mathrm{nm}$, $560\ \mathrm{nm}$ and $642\ \mathrm{nm}$, were notch filters designed for normal incidence. As such, the optical design had to fit three mirrors at shallow ($< 15°$) angles. The SMLM path was slightly demagnified to support the use of up to 3 color channels on a 2048×2048 pixel chip. The SMLM detection path images up to 512×512 pixels or roughly a $53 \times 53\ \mu\mathrm{m}$ field of view without clipping (in fact, the optical path supports up to an $80 \times 80\ \mu\mathrm{m}$ FOV, but color channels would overlap on the camera). A variable aperture sets the detection FOV. The excitation path is critically illuminated to maximize power at the sample, and a fiber shaker smooths the speckle noise generated by the lasers [71].

This microscope features a motorized xy stage and a piezoelectric z stage to allow for automated, high throughput imaging [37].

3.3 Future applications

As described in 3.1, this microscope can be used to image subcellular membranes in live cells. Further development on the software side, most likely involving machine learning, is needed to enable this possibility.

The available cellular context is immediately useful in fixed-cell super-resolution imaging. For example, with only staining for an ER membrane marker and a structural protein, it is possible to determine if the ER is most likely displaced because of the structural protein, because a microtubule is pulling on it, or because there is a mitochondria physically blocking the ER from moving in a different direction.

This microscope may be beneficial for studying phase condensates. Microscopy, often fluorescence microscopy, is used to confirm a phase change in liquid-liquid phase separa-

tion and measure the local density of the phases. Some phase condensates exist below the diffraction limit, and super-resolution microscopy must be used. There was a recent call in *Cell* to use quantitative phase imaging to assess the material properties of phase condensates [72]. The OI-DIC/SMLM microscope will be capable of simultaneously imaging sub-diffraction phase condensates in SMLM while examining refractive index differences in the same area.

To assess the possibility of studying phase condensates, a cell stained for coilin was sequentially imaged in standard widefield fluorescence and in OI-DIC on an earlier build of the microscope. Coilin is a protein found in Cajal bodies that localizes to, but does not form, phase condensates [73]. The images, shown in Figure. 3.3 revealed good correlation between changes in the OPD reconstructed from OI-DIC and the fluorescence marker for coilin.

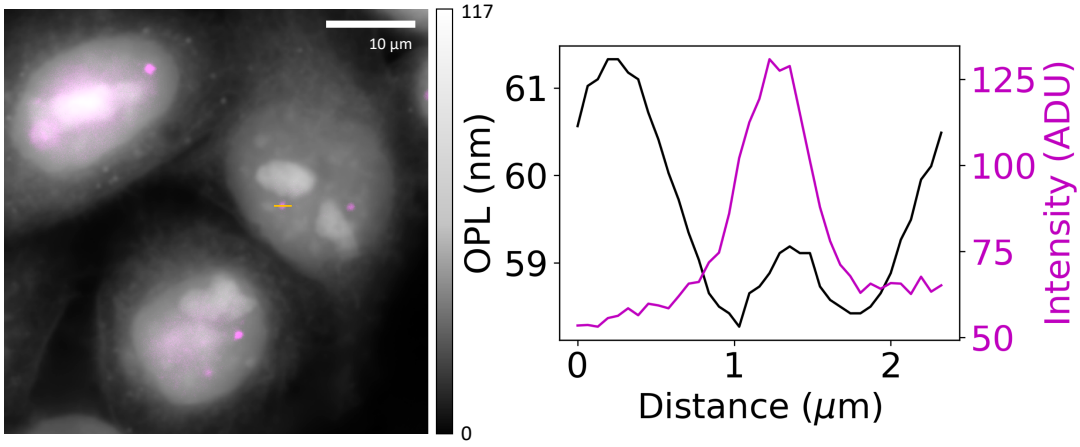


Figure 3.3: Left, An OPD map overlaid with fluorescence image of coilin in HeLa cells. **Right,** A sample line profile indicating the correlation between refractive index change and coilin aggregation.

It has been shown that OI-DIC achieves a sensitivity of ~ 0.5 nm in the OPD, but

this only works if the contrast of the object is sufficient [46]. To ensure it is possible to detect refractive index (RI) changes in the presence of condensates that are smaller than the diffraction limit, $100\ \mathrm{nm}$ fluorescent polystyrene beads (RI ~ 1.6) were imaged in water. The system bias was adjusted to $\Gamma \approx 100\ \mathrm{nm}$. Results, shown in Figure 3.4, indicate that objects smaller than the diffraction limit of light can produce shifts in the diffraction-limited OPD map. Further experiments must be performed on sub-diffraction samples with RI and surrounding RI closer to physiologically relevant values.

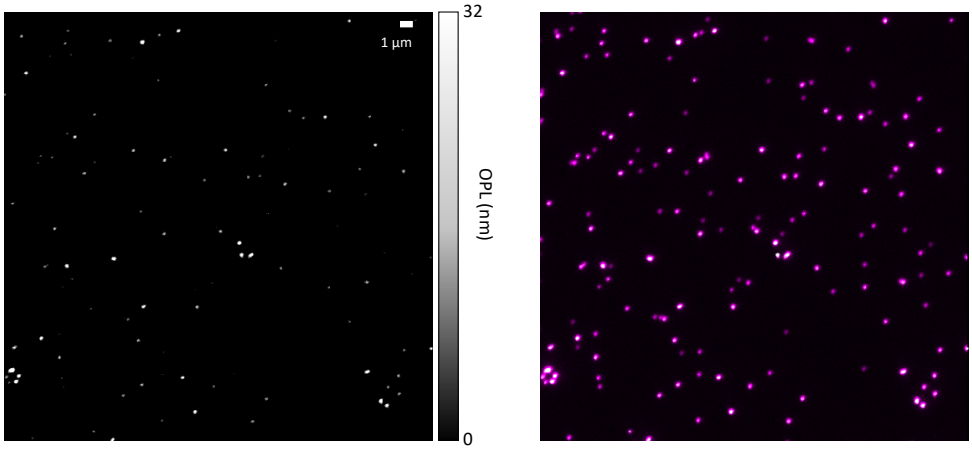


Figure 3.4: $100\ \mathrm{nm}$ fluorescent beads (ThermoFisher FluoSpheres^{TM} Carboxylate-Modified Microspheres, 0.2 µm, red fluorescent (580/605), 2% solids) imaged in OI-DIC to produce an OPD map (left) and simultaneously in a fluorescent channel (overlaid on the OPD image, right). Additional spots in the fluorescence channel may come from beads that did not produce sufficient phase contrast or from non-specific fluorescence in the sample.

Optical aberrations introduced by samples degrade SMLM images in thick cells and tissues to the point where adaptive optics corrections are a necessity [74, 75]. The OPD measured by OI-DIC is a map of the optical wave front aberrated by the sample [34]. Thus, the OPD image can be directly used as input for correction of aberrations in the SMLM channel.

OI-DIC is based on DIC, which is well-suited to imaging thick samples, and so the OI-DIC/SMLM microscope has the potential to be an excellent super-resolution tissue imaging scope, provided that sufficient corrections can be applied to the SMLM channel via a deformable mirror or similar adaptive optics device. No other tissue imaging systems offers simultaneous widefield/mesoscale imaging and nanoscale imaging. This could be particularly useful for searching for rare events in tissue and then imaging the sub-cellular structure of these rare events. This type of data could be used to generate novel hypothesis for mechanisms of disease.

The OI-DIC is able to image pan-ExM tissues stained with a refractive-index-shifting dye, an example is shown in Figure 3.5. Since these samples do not bleach, it is possible to extract all of the detail in the cellular ultrastructure. The high throughput capabilities of this microscope could be leveraged to image large numbers of pan-ExM images in a short period of time. The previously-mentioned aberration corrections could help produce better-quality images.

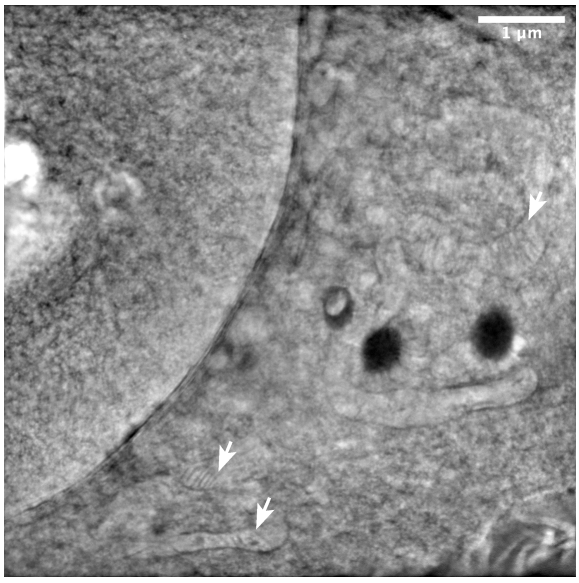


Figure 3.5: Riesz reconstruction of OI-DIC image of pan-ExM sample with refractive index stain showing cell ultrastructure. The nucleus occupies about a third of the image, starting in the upper left quadrant. The arrows indicate mitochondria cristae. Sample prepared by Ons M'Saad.

Chapter 4

Software for visualization and quantification of localization data sets

4.1 Introduction

Every modern Ph.D. student writes an analysis script or two, but it is rare to create a maintained, cohesive package of visualization and analysis techniques that can be used by both the student and other people. The software project described in this chapter, originally called PySMI, was started by my co-advisor Dr. David Baddeley during his doctoral research [76]. It solved, and continues to solve, the problem of handling the visualization and analysis of point clouds acquired during localization imaging. It is now perhaps the most comprehensive open-source localization analysis software package available (see Section B.3). My primary contribution to this project was developing mesh construction, quality control and visualization tools. I also repaired and optimized existing functionality, added additional established analysis methods (such as Ripley's K [77]), and helped improve some of the software's human-computer interaction.

4.2 PYMEVisualize: an open-source tool for exploring 3D super-resolution data

This section is reproduced from

Zach Marin, Michael Graff, Andrew E. S. Barentine, Christian Soeller, Kenny Kwok Hin Chung, Lukas A. Fuentes, and David Baddeley. PYMEVisualize: an open-source tool for exploring 3D super-resolution data. *Nature Methods*, 18(6):582–584, June 2021. ISSN 4159202101. doi: 10.1038/s41592-021-01165-9. URL `http://www.nature.com/articles/s41592-021-01165-9`.

Single-molecule localization microscopy techniques such as PALM, STORM, and PAINT are increasingly critical tools for biological discovery. These methods generate lists of single fluorophore positions that capture nanoscale structural details of subcellular organization, but to develop biological insight, we must postprocess and visualize these data in a meaningful way. Many algorithms have been developed for localization postprocessing [56, 78], transforming point data into representations that approximate traditional microscopy images [61, 78, 79], and performing specific quantitative analysis directly on points [56, 78, 80–82]. Available packages (Section B.3), however, typically implement a small subset of these algorithms, necessitating complex workflows involving multiple different software packages. Here we present PYMEVisualize, an open-source tool for the interactive exploration and analysis of three-dimensional (3D), multicolor, single-molecule localization data. PYMEVisualize brings together a broad range of the most commonly used postprocessing, density mapping and direct quantification tools in an easy-to-use and extensible package (Fig. 4.1). This software is one component of the Python Microscopy Environment (`http://python-microscopy.org`), an integrated application suite for light microscopy acquisition, data storage, visualization and analysis built on top of

the scientific Python environment [82].

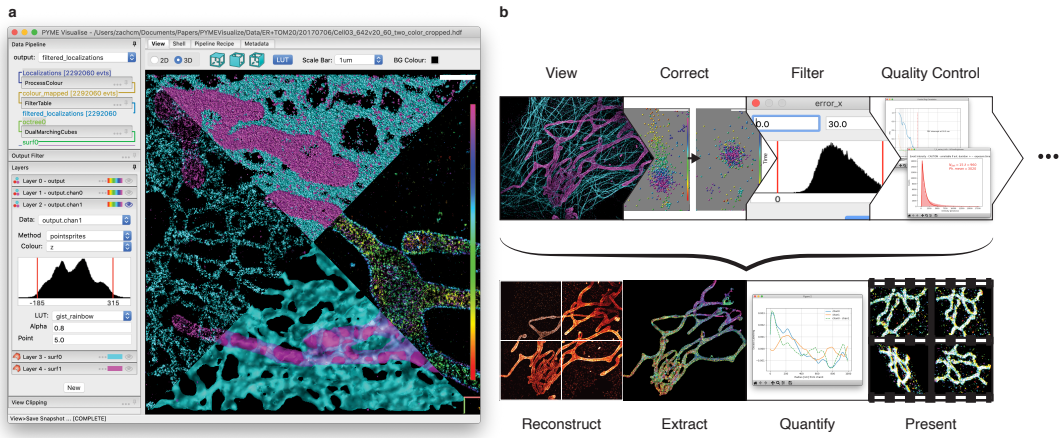


Figure 4.1: Overview of PYMEVisualize. a, The PYMEVisualize user interface. The viewer display is divided into four quadrants illustrating four different methods of visualizing the same dataset—a two-color dataset of the endoplasmic reticulum (cyan) and mitochondria (magenta). Clockwise from the left, the methods represent the data as the sum of Gaussians, solid spheres, a single channel (the mitochondria) colored by depth, and 3D surfaces extracted from the point clouds. **b,** Examples of exploration, visualization and quantification tools in PYMEVisualize. A workflow from raw events to quantification can consist of multiple steps—from an initial viewing of the localizations, through corrections (fiducial-based drift correction shown), filtering on localization precision or other parameters, and the calculation of quality control metrics such as Fourier ring correlation and event photon counts. Workflows can have a wide range of endpoints, such as the creation of density reconstructions (with a variety of supported algorithms), extraction of isosurfaces, quantification (analysis of pairwise distances between channels shown), and preparation of visuals and animations.

PYMEVisualize is the result of over 10 years of continuous development, evolving from a simple OpenGL-based point viewer into a complete platform for the analysis of localization microscopy point clouds, including modules for quality control, artifact correction, density image reconstruction and quantitative analysis. It has become an indispensable tool in our laboratories and those of several collaborators, facilitating widely varying workflows such as cluster analysis in multicolor ratiometric dSTORM data, correlative confocal and STORM imaging, PAINT imaging of DNA origami with fiducial correction, channel registration and z-mapping for multicolor biplanar astigmatism data, extraction

of organelle surfaces from 4Pi-SMS images of the endoplasmic reticulum and Golgi, and generation of 3D animations for use in presentations [83–85]. Its configurable processing pipeline allows for flexibility and repeatability. Loading data from a wide selection of localization packages is possible through our CSV and Matlab importers (Section B.2), and a flexible plug-in system makes it easy to extend.

We believe that PYMEVisualize has the potential to become the go-to tool for the analysis and visualization of localization point datasets, akin to the status ImageJ has for pixel-based microscopy images, and that we have a duty to share it with the broader imaging community. In contrast to packages based on Matlab, PYMEVisualize and all its dependencies are freely available and modifiable. An added advantage is the ability to leverage the entire scientific Python ecosystem, containing a large volume of mature, well-tested algorithms for writing high-quality plug-ins with relative ease. PYMEVisualize is in active development, and we welcome contributions from everyone—be it to the core code, through the creation of custom plug-ins, or simply by providing feedback about your experience. Guidelines on how best to contribute are provided at `https://python-microscopy.org/doc/Contributing.html`. The source code is available on GitHub (`https://github.com/python-microscopy/python-microscopy`), and queries, feedback or suggestions can be made directly, on the GitHub issue tracker, or using the "pyme" tag in the image.sc forum. We have included parts of the PYMEVisualize documentation— a brief tutorial and the user guide—as Sections B.1 and B.2, along with videos (Supplementary Videos 1 and 2) demonstrating visualization and quantification pipelines.

Chapter 5

Nanoscale surface topology from single-molecule localization microscopy point data

This chapter describes a novel algorithm for fitting surface representations of membranes to single-molecule localization microscopy data. It is implemented in a custom surface mesh Python library, described in Appendix D. This chapter is a draft of Z. Marin, L. A. Fuentes, J. Bewersdorf and D. Baddeley, "Nanoscale surface topology from single-molecule localization microscopy point data," in preparation.

5.1 Introduction

Changes in cellular membrane shape are linked with viral replication, Alzheimer's, heart disease, and an abundance of other maladies [1, 3, 5–8, 11]. In order to understand the mechanisms of these diseases, it is necessary to image both the locations of protein clusters causing structural changes in the membrane and the curvature of the membrane at this size scale. Some membranous organelles, such as the endoplasmic reticulum (ER) and the Golgi, are only ~ 50 nm in diameter, and a resolution of half this size scale or better is

needed to image structure.

Electron microscopy (EM) techniques have a resolution of 2 nm or better and are well-suited to imaging membranes at this size scale. Segmentation of membrane structures from EM images can be inconvenient, but is doable [15, 16]. However, it is difficult to label and identify proteins interacting with a membrane. Immunolabelling with gold nanoparticles in EM requires fixation methods that often destroy cellular structures, and gold nanoparticle locations must be manually extracted from among other, unstained cellular features [13, 14].

Fluorescence microscopy techniques can spectrally separate multiple fluorescent labels, allowing for easy identification of both membranes and membrane-interacting proteins. Conventional fluorescence microscopy techniques can achieve a resolution no better than 250 nm, and are therefore unable to visualize membrane changes at the necessary size scale. Single-molecule localization microscopy (SMLM) techniques, such as PALM, STORM, and PAINT, image the positions of proteins with ~ 20 nm resolution [86]. This size scale is sufficient for imaging membrane structural changes of interest, and for imaging protein clusters thought to cause these changes [87]. In contrast to EM imaging techniques, which show a continuous membrane, SMLM yields a sparse and noisy point cloud of protein locations, each with an uncertainty that depends on the brightness of the underlying blinking event. To visualize and quantify a membrane, it is necessary to interpolate a continuous surface from these positions.

In the fields of remote sensing [88] and 3D scanning [89], Screened Poisson Reconstruction (SPR) [65] is often used to extract surfaces from point clouds. This method has also been applied to SMLM [79], but requires extremely high quality data, a number of pre-processing steps and case-dependent subjective setting of hyper-parameters. In order to reconstruct an accurate cilia surface, Yoon et al. had to acquire a dense SMLM data set, filter out noise, and reflect the data set about a chosen cut point to generate a point

cloud that represented a theoretically manifold structure. SPR works well on high-quality SMLM data sets, but is time-consuming and difficult to generalize to sparse and/or noisy SMLM data.

SPR reconstruction is designed to follow point locations exactly, giving it high fidelity to a point cloud. Due to labelling inefficiencies and sampling, SMLM data often shows holes in large regions of a structure. Fluorescent labels often bind to not only molecules of interest, but to other, non-specific targets in the sample. This and sample auto-fluorescence can generate spurious background localizations [35]. The stochastic nature of SMLM imaging means each fluorescent molecule may blink multiple times, appearing as multiple localizations, each in a slightly different spot. Adhering strictly to all of these points does not necessarily generate a surface approximation of the underlying structure the point cloud represents.

In order to generate an accurate surface from SMLM data, it is necessary to account for the localization precision of each point. By weighting each point's influence on surface structure by its precision, a surface can fit to all local points in a given area. This also means a surface is allowed to float in areas of lower uncertainty, while still adhering to the data, making it possible to fit regions of space with few points, and to ignore contributions of poorly-localized spots arising from auto-fluorescence and non-specific binding. Zhao et al. developed theory for incorporating localization uncertainty to fit SMLM data sets, but did not create an implementation [70]. To our knowledge, no research group has demonstrated a general method for fitting surfaces to SMLM data that leverages information about localization uncertainties.

Here we present a novel algorithm that creates representations of a membrane from an SMLM point cloud that is generated by localizing membrane markers. This algorithm incorporates localization uncertainty into its fitting routine, works for any SMLM data set, and the resulting surface can be used to measure membrane curvature at the sampling rate

of the input data. Additional smoothing techniques are often used to extract a reasonable shape from ER images [68, 90], and a similar smoothing curvature force is applied in this algorithm to ensure the surface representation features only biophysically-allowed bending. This algorithm is implemented PYthon Microscopy Environment Visualize[1] for ease-of-use and integration with additional SMLM acquisition and analysis techniques [49].

5.2 Materials and Methods

The algorithm is sketched in Figure 5.1, and consists of an initial coarse estimation of a starting mesh, which is then refined under point fidelity and curvature constraints.

5.2.1 Initial / starting mesh

A set of single-molecule localizations, which are expected to come from a membrane marker, are placed in a sparse octree data structure [64]. The octree is truncated at a given minimum number of localizations per octree cell (equivalent to a minimum signal to noise ratio—see [61]). This has the effect of dividing the volume into cubic cells with a size that adapts to the local point density. Cells will be large in areas with few localizations, and small in areas which are localization dense. Cells containing fewer than the minimum number of points are not stored. The result is a volumetric data structure that contains the same information as a regularly sampled grid, but requires significantly less memory. The density of localizations in each cell are calculated. The Dual Marching Cubes algorithm is run on these cells with a given threshold density [67]. The result is a manifold triangular mesh that separates high from low density areas. Because of sparsity in SMLM labelling, the threshold density must be purposely set a bit low in order to create a single surface that

[1] `https://python-microscopy.org`

encapsulates the whole membrane.

5.2.2 Topology modification

The purposefully low threshold density has the occasional consequence of representing a disconnected portion of structure the point cloud represents as connected. To address this, holes and cuts in the mesh are generated where it is possible to pass a ball of a given size through the mesh without ever touching an input localization (see Supplementary Note). This allows for removal of false connections generated in the coarse isosurface, and testing for the presence of important cellular features such as nanoholes [91].

5.2.3 Mesh quality

This mesh is periodically remeshed to improve the numerical quality of subsequent calculations [92]. Remeshing consists of a number of operations - splitting long edges, collapsing short edges, "flipping" edges where too many are incident on a vertex, and moving vertices along the surface. Together these remeshing operations result in a well-formed mesh where the edge-lengths are roughly constant, the number of edges incident on a single vertex is roughly constant, and the triangles are roughly equilateral.

The minimum edge length of the mesh needs to sample all available data. This means properly sampling the size scale of our smallest available features, which are at the size scale of the minimum localization precision in the input point cloud. Starting from the edge length size of the input coarse isosurface, the mesh edge lengths decrease linearly with each remeshing step toward $\frac{\min_i \sigma_i}{2.5}$ (see Mesh optimisation)–that is, toward $0.4\times$ the minimum localization precision. This ensures the iterative fitting makes large adjustments early in the fitting and fits detailed features later on.

5.2.4 Mesh optimisation

The resulting mesh is refined to achieve high fidelity with the localisation input data under a curvature constraint, which ensures the result maintains the smoothness of a true biological membrane. This is expressed mathematically as a minimisation problem:

$$\arg\min_{\mathbf{v}} \left\| \frac{1}{\boldsymbol{\sigma}\left(\frac{\mathbf{p}-\mathbb{A}(\mathbf{v})}{2\boldsymbol{\sigma}}+1\right)} \left(\mathbf{p}-\mathbb{A}(\mathbf{v})\right) \right\|^2 + \lambda^2 \left\|\mathbb{B}(\mathbf{v})\right\|^2 \tag{5.1}$$

where $\mathbf{p}$ are the localisation positions, $\mathbf{v}$ the mesh vertex positions, the $\mathbf{p} - \mathbb{A}(\mathbf{v})$ term represents the distances between each localisation and the mesh, $\boldsymbol{\sigma}$ are localisation uncertainties, $\mathbb{B}(\mathbf{v})$ encodes a curvature penalty, and λ is a constant controlling the relative weighting of point fidelity and curvature terms. This is solved using a conjugate gradient descent method [93].

If, after refinement, the mesh has not changed shape significantly, or if the given maximum number of iterations has been reached, the algorithm terminates and the resulting surface is presented as an approximation of the membrane. Otherwise, the surface is passed back to the hole/cut generation step.

Point fidelity term

Notation:

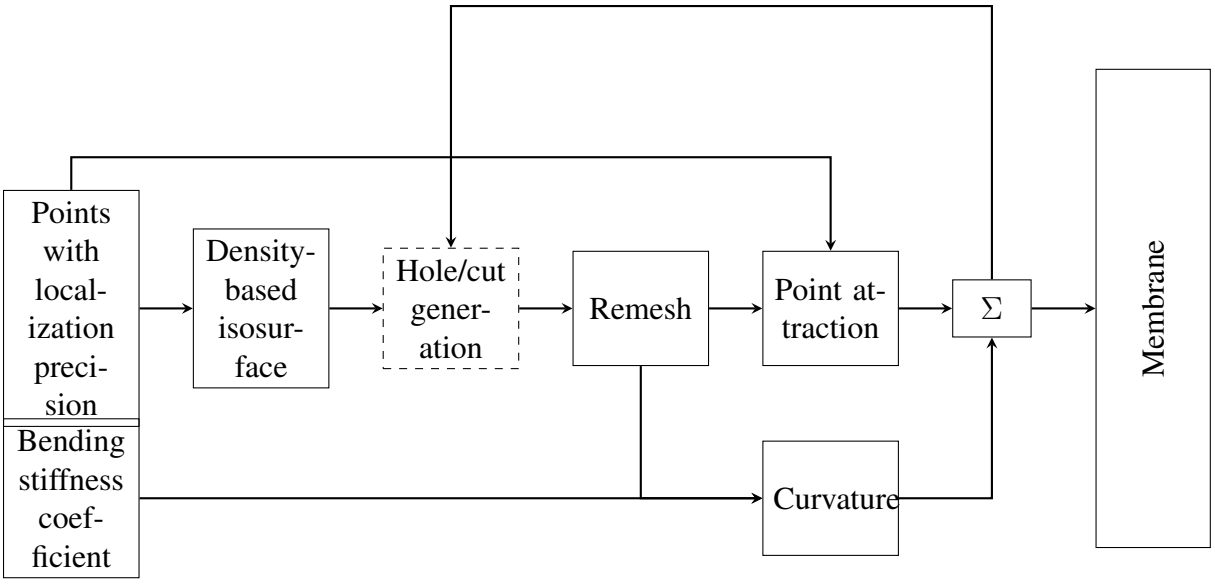


Figure 5.1: A flow diagram of the proposed algorithm. Localization data is first approximated by a coarse, density-based isosurface. This surface is then tested to see if any hole punching or cutting is needed, remeshed for improved numerical quality, and then moved toward the localizations subject to a curvature force constraint. The pipeline runs iteratively until stopping criteria are met.

$\mathbf{p}$	the $N \times 3$ array of fluorphore localisations
$\mathbf{p}_i = \vec{p_i}$	the ith localisation (1×3)
$\boldsymbol{\sigma}$	$N \times 3$ array of the uncertainty of the localisations
$\boldsymbol{\sigma}_i$	the uncertainty of the ith localisation (1×3)
$\mathbf{v}$	the $M \times 3$ array of mesh vertices
$\mathbf{v}_k = \vec{v_k}$	the kth mesh vertex (1×3)
$\vec{v_j}$	the jth vertex of a given mesh face
$\vec{v_l}$	the lth vertex neighbour of a given vertex

The point attraction term seeks to minimize the distances between each localisation and the surface. For each localization, we approximate its distance to the surface as the distance to a proxy point formed by a linear combination of the 3 vertex positions that define the closest surface face. The weightings used in the linear combination are computed as follows, based on the inverse distance from the localization $\vec{p_i}$ to each vertex $\vec{v_j}$ of the

nearest face.

$$w_{ij} = \frac{1}{||\vec{p}_i - \vec{v}_j(\vec{p}_i)||} \Bigg/ \sum_{j=0}^{2} \frac{1}{||\vec{p}_i - \vec{v}_j(\vec{p}_i)||}$$

The position of the proxy point is then calculated as

$$\mathbb{A}(\mathbf{v})_i = \sum_{j=0}^{2} \vec{v}_j(\vec{p}_i) w_{ij}$$

giving the following distance metric:

$$[\mathbf{p} - \mathbb{A}(\mathbf{v})]_i = \vec{p}_i - \sum_{j=0}^{2} \vec{v}_j(\vec{p}_i) w_{ij}.$$

This metric is asymptotically equal to the true distance to the surface at small distances, but tends to the distance between the localisation and the centre of the face (all vertices weighted equally) at large distances. This behaviour was deliberately chosen to ensure sensible updating of vertex positions - for a localisation close to the surface we want it to mostly pull on the closest vertex, whereas a point that is far away (compared to the face edge length) should pull equally on all vertices of the face.

This distance metric is weighted by

$$\frac{1}{\boldsymbol{\sigma}_i \left(\frac{[\mathbf{p} - \mathbb{A}(\mathbf{v})]_i}{2\boldsymbol{\sigma}_i} + 1 \right)}$$

as shown in Equation 5.1. The $\frac{1}{\boldsymbol{\sigma}_i}$ term ensures distances near a localization produce minimal cost. The remaining portion of the term de-weights localizations that are particularly far away, while still letting them exert influence on the surface. This ensures the surface can shift to accurately fit all points in a point cloud if necessary, but places priority on moving toward the centroid of its nearby points first. Since localizations are Gaussian-distributed, they should exercise most of their influence within $2\boldsymbol{\sigma}_i$ of their position.

Curvature term

The curvature force minimizes bending energy according to a variant of the Canham-Helfrich energy functional [94]. This force seeks to bound the surface curvature to biophysically-possible values.

We assume the input surface is manifold and use a Laplacian discretization of the mean curvature energy at a surface vertex $\vec{v}_k$,

$$E_{\text{bend}}(\vec{v}_k) = ||\mathbb{B}(\mathbf{v}_k)||^2 = \frac{\kappa}{2}\frac{1}{\Omega_k}\left[\sum_l \vec{v_k} - \vec{v_l}\right]^2$$

where $\Omega_k = \frac{\sqrt{3}}{4}\sum_l ||\vec{v}_k - \vec{v}_l||^2$, $\vec{v}_l$ is the lth neighbor of vertex $\vec{v}_k$, and κ is the stiffness coefficient for the lipid composition of the membrane. The stiffness coefficient is approximately $18 - 57k_bT$, where k_b is Boltzmann's constant and T is absolute temperature, and exact values can be calculated or looked up in the literature [95–97].

5.2.5 Simulation

SMLM point clouds were simulated from a theoretical figure eight, defined by a signed-distance function (see Supplementary Note). Simulations varied point cloud density, localization precision and number of background localizations.

5.2.6 Quality evaluation

To assess the accuracy of the method described in this paper as compared to SPR, surfaces were fit to simulated point clouds and then compared to the theoretical structure giving rise to these point clouds. When making this comparison we must consider two types of error 1) the distance from the true surface to the reconstructed surface and 2) the distance from the reconstructed surface to the true surface. Although these might seem redundant

at first glance, it is possible for a surface to appear good under metric 1 whilst having bad performance under metric 2. An example of this is a structure which mostly closely follows the true surface, but also has blebs or extrustrusions away from the true surface. Because the distance in 1) just considers the parts of the reconstruction which are closest to the true structure, the blebs or extrusions are unpenalised and metric 1 returns a small distance. Conversely a reconstruction which follows part of the ground truth correctly, but is truncated such that it does not extend into all areas of the ground truth, can score well on metric 2. To be a good reconstruction, both these metrics must be minimal. The distance between surfaces was calculated numerically as follows; a set of of noise-free verification points were simulated exactly on the surface of the theoretical structure, and a set of noise-free points were simulated on on the fit surface. The mean squared distance from the verification point set to its nearest neighbors in the mesh point set was computed as quality metric Q_1. The mean squared distance from the mesh point set to its nearest neighbors in the verification point set was computed as quality metric Q_2. Mesh quality was scored as a combination of the two error types:

$$Q = \sqrt{\frac{Q_1 + Q_2}{2}}$$

where Q is the root mean square error, representing the average distance from the mesh to the theoretical structure.

5.2.7 Sample preparation

The endoplasmic reticulum (ER)-microtubule (MT) sample was prepared by Lena Schroeder as detailed in [27]. The standalone ER sample was prepared by Phylicia Kidd. COS7 cells were fixed in 3% PFA 0.1% GA. The ER transmembrane marker Sec61β was attached to GFP. GFP was labeled with an Primary Antibody against GFP (host: Rabbit). 5$\times$R4

DNA-PAINT docking site was conjugated to a anti-Rabbit Secondary Nanobody. R4 was at a concentration of 1 nM in Buffer $1\times$PBS + 500 mM NaCl.

5.2.8 Experimental imaging

The ER-MT sample was imaged by Yongdeng Zhang with 4Pi SMLM to achieve isotropic resolution, as detailed in [27]. The standalone ER sample was imaged by Florian Schüder on an Andor Dragonfly confocal in total internal reflection mode with an angle of -70 degrees. A $60\times$ objective (NA=1.49) was used to achieve $108\ \mathrm{nm}$ pixel size. The 561 nm laser line was set at 100% resulting in a power of $82\mathrm{mW}$ and an intensity of $\sim 170\mathrm{Wcm}^{-2}$ at the sample plane. A $50\ \mathrm{ms}$ integration time was used and images were stored as 12 bit. Localizations were fit and analyzed in Picasso [98]. Axial location was determined via an astigmatic fit.

5.3 Results and Discussion

5.3.1 Validation on test structures

A three-dimensional figure eight (two toruses touching) was simulated with both low and high localization precision and background. For each condition, the parameters for both SPR and the method described in this paper were grid searched. The resulting meshes were scored as described in Quality evaluation. The results are shown in Figure 5.2. SPR performs as well or better than our method in the case of high point density, low noise and low localization precision. Our method outperforms SPR in lower-density, noisier point clouds that have localizations with larger precisions.

For each condition, the mesh with the lowest Q from each method was selected for comparison. Following recommendations in [65], for SPR we searched for estimating

normals from $10, 30, 50, 100$ nearest neighbors, smoothing parameter $\alpha = 0, 2, 4$, and samples-per-node$= 1.5, 5, 10, 30$. We used an octree depth of 8, 8 Gauss-Seidel relaxations, and a scale factor of 1.1. For the method in this paper we compared maximum number of iterations $= 9, 19, 39, 59, 79, 99$ and $\lambda = 5, 10, 15, 20, 25$. We remeshed every 5 iterations. The behaviour of each metric over these parameters is plotted in Figures C.1 and C.2. The runtime for our method is linear with number of iterations, as shown in Figure C.3.

5.3.2 Application to SMLM data

4Pi SMLM

An SMLM image of the ER was taken on a 4Pi SMLM microscope as described in Experimental Imaging. 4Pi imaging provides isotropic resolution in the x, y and z-directions. A mesh was generated and iteratively fit for 79 iterations, remeshing every 5 iterations, at $\lambda = 50$. This produces a reasonable surface fitting, as shown in Figure 5.3.

The ER shown in Figure 5.3 is part of a larger, two-color image that also shows microtubules (MTs). Although MTs are not membranes, they are hollow tubes and can be meshed and fit with the algorithm described in this paper. To verify the accuracy of these reconstructions, it is possible to compute the principal curvatures at each point on the mesh [99]. ER tubules are roughly $100\ \mathrm{nm}$ in diameter, suggesting they should have a curvature of roughly $\frac{1}{50}\ \mathrm{nm}^{-1}$. Microtubules are approximately $30\ \mathrm{nm}$ in diameter and should have a curvature around $\frac{1}{15}\ \mathrm{nm}^{-1}$. As shown in Figure 5.4, the ER mesh has a median maximal principle curvature around $\frac{1}{53}\ \mathrm{nm}^{-1}$, which is exceedingly close to what we expect. Microtubules are only for display here, as they were not wrapped for a sufficient number of iterations to be quantified.

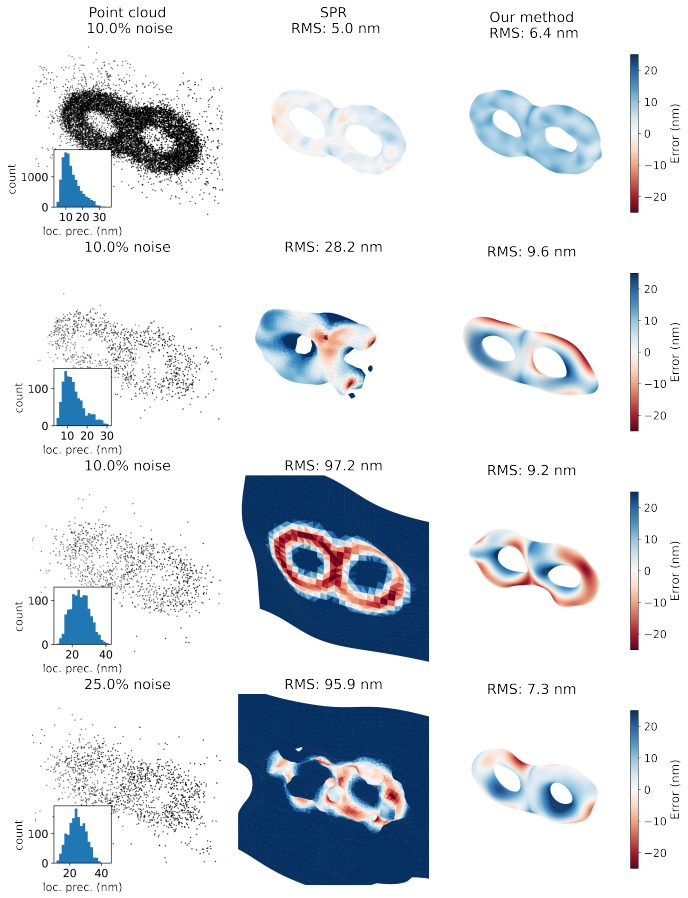


Figure 5.2: A comparison of SPR and our method on simulated point clouds of a 3D figure 8, which is approximately 500 nm in diameter and 60 nm thick. Each row contains a simulated point cloud, the SPR reconstruction and our method's reconstruction. The inset plots show the distribution of localization precision ($\sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$) for the simulated point cloud. The error bar shows the LUT for the distance from a mesh face to the theoretical structure, and is scaled the same for both meshes in each row. Both our method and SPR work well in the case of high density of points and low localization precision. As density decreases, SPR has a harder time approximating data (row 2). If localization precision also increases to realistic levels for SMLM (row 3), and/or background noise increases (row 4), SPR cannot reconstruct the underlying structure. Our method continues to work in all of these cases, with RMS errors (Q, see Quality evaluation) to the theoretical surface less than the localization precision of the point cloud.

Astigmatic SMLM

A SMLM image of the ER was taken with on a system using astigmatic fitting to acquire the z-position, yielding a z-resolution that is approximately $3\times$ worse than the xy-resolution. A mesh was generated and iteratively fit for 79 iterations, remeshing every 5

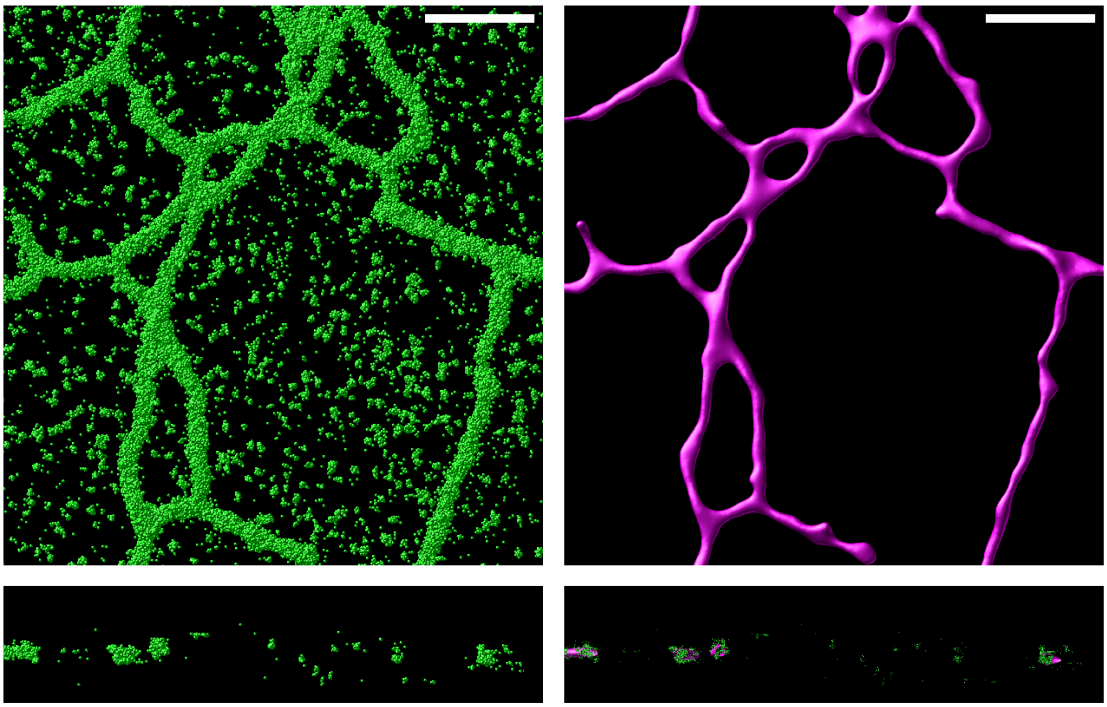


Figure 5.3: Left, A portion of the 4Pi ER data set shown in xy (top) and xz (bottom). Points are rendered as spheres of radius $30\ \mathrm{nm}$. **Right,** The mesh rendering of the ER data set. The xz slice shows a $100\ \mathrm{nm}$ cutaway of the center xy-plane. Points are rendered as spheres of radius $10\ \mathrm{nm}$ to show the tubules passing through the centroid of nearby points, indicating a good fit. Scale bars are $1\ \mu\mathrm{m}$.

iterations, at $\lambda = 50$. Our meshing routine is still able to produce a reasonable surface estimation, as shown in Figure 5.5, although there is some elongation in the z-direction. This is expected, as the procedure described in this paper will try to fit the point cloud shape. This data set provided no localization precision estimate for the z-direction, and so the error in the x-direction was used as a proxy. A better estimate for localization precision in the z-direction is likely to yield a more accurate surface.

5.4 Conclusion

Examining the interplay of membrane surfaces and proteins is critical to understanding cellular function [100]. The method described in this paper provides a new way for re-

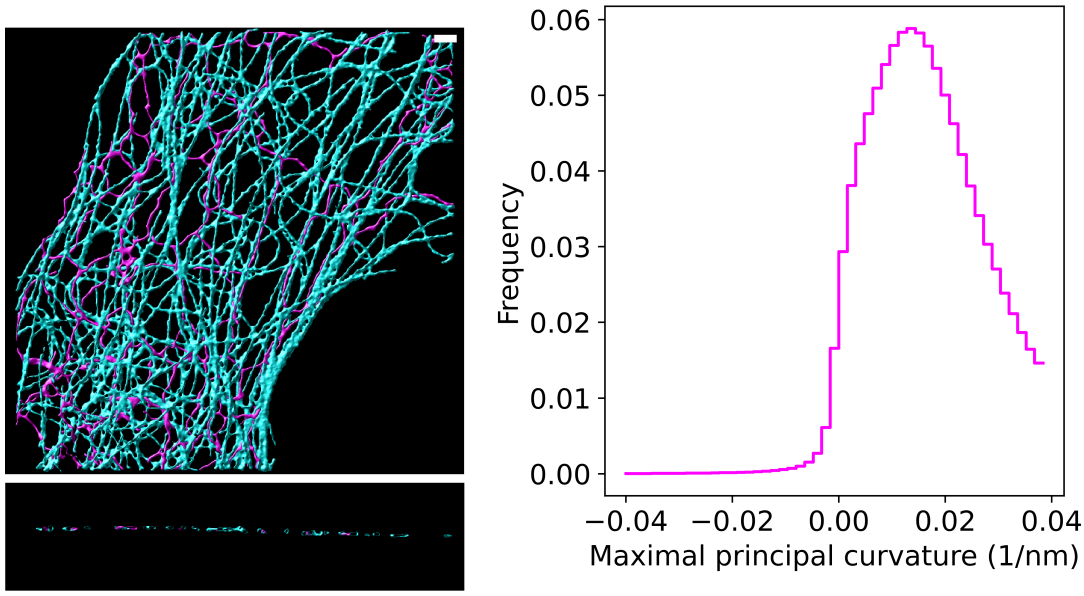


Figure 5.4: Two-color MT-ER sample, meshed. ER was fit as above, MT was fit for 19 iterations, remeshing every 5 iterations, at $\lambda = 50$. The left shows the xy and xz profile of the meshes, and the right shows a histogram of the ER's principal curvature values along the direction of maximal principal curvature.

searchers to quantify membrane surfaces from super-resolution localisation microscopy and to use this quantification to investigate the structure and biophysical properties of organelle and cellular membranes and their associated proteins.

Out method enables higher fidelity than previously demonstrated methods for acquiring surfaces from super resolution data, and functions across a wide range of localisation precision, density, and background. It is packaged as open-source software, with an interactive GUI, which lets it be rapidly used and adapted by others, and is sufficiently fast to allow use on standard computer hardware (e.g. laptops).

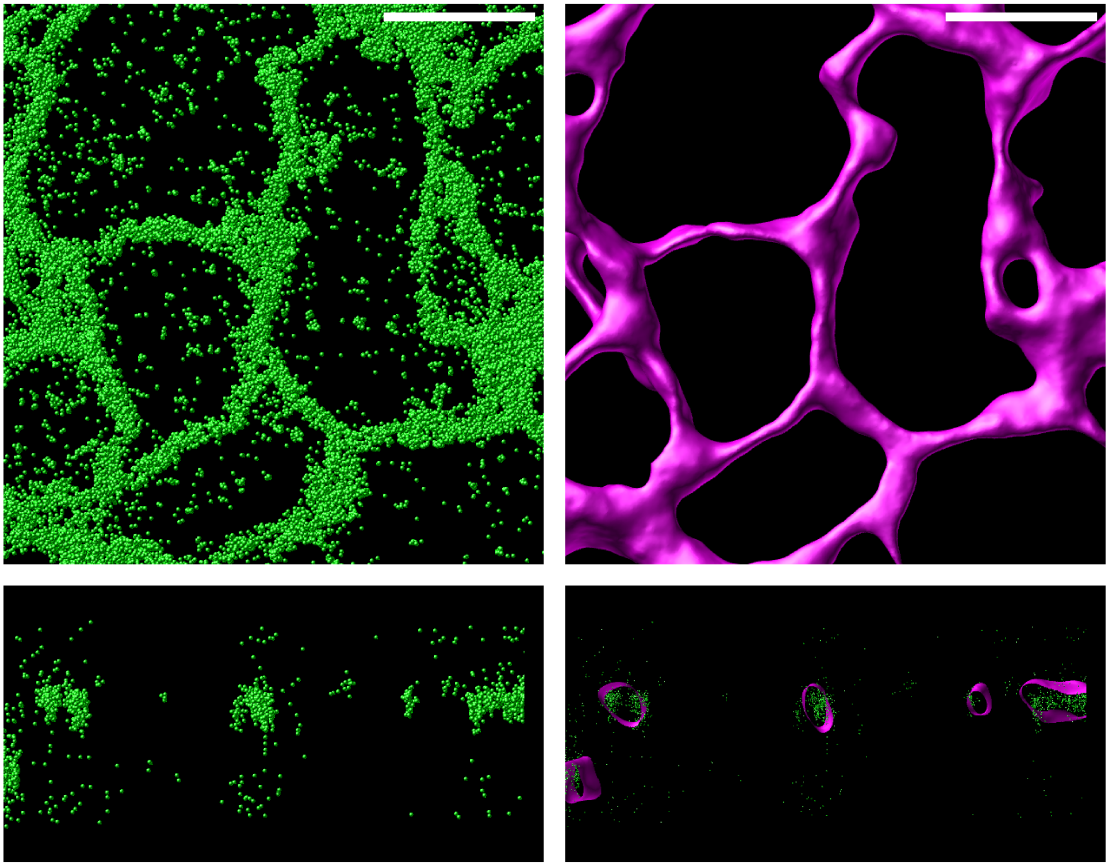


Figure 5.5: Left, A portion of the astigmatic ER data set shown in xy (top) and xz (bottom). Points are rendered as spheres of radius $30\ \mathrm{nm}$. **Right,** The mesh rendering of the ER data set. The xz slice shows a $100\ \mathrm{nm}$ cutaway of the center xy-plane. Points are rendered as spheres of radius $10\ \mathrm{nm}$ to show the tubules passing through the centroid of nearby points, indicating a decent fit but some elongation in the z-direction. Scale bars are $1\ \mu\mathrm{m}$.

Chapter 6

Conclusions and Outlook

6.1 Perspective on correlative OI-DIC and SMLM

The correlative OI-DIC and SMLM microscope provides quantitative cellular context for live-cell SMLM, something possible on no other existing microscope. Unfortunately, as discussed in Section 2.1.4, live-cell SMLM is still limited by available compatible dyes, imaging speed and excitation light intensity. The context provided by OI-DIC is beneficial, but requires some degree of luck in that the user has to hope localizations will pop up in areas where they are needed at the right time. The current speed of OI-DIC is a bit too slow to capture ER membrane movement. This is a technical rather than a theoretical problem, and could be solved either by incorporating faster liquid crystals (see Section 3.1) or by converting the OI-DIC to image multiple shear angles and biases simultaneously. Unfortunately, simultaneous imaging of multiple shear angles may require a specialized optical path, making it harder to combine the OI-DIC with SMLM. The faster liquid crystals are likely the better option, although further exploration is needed.

The fixed cell uses for this microscope, discussed in Chapter 3, hold more promise in the immediate future. In particular, the microscope could be quite useful for 3D tissue imaging. However, while the sensitivity of the OI-DIC to the OPD has been measured, the

depth of field of the OI-DIC is still not well characterized. Knowledge of the depth of field will be critical for teasing out overlapping structure in 3D OI-DIC images.

6.2 Perspective on open-source SMLM software

Microscopy has always required image analysis, and was therefore an early adopter of computational techniques. Researchers in the microscopy field analyzed images by eye[1] long before computers arrived on the scene, and carried the assumption that image analysis is a quick, final step in the research process into the digital age. This created incentives for junior researchers to quickly develop image analysis scripts on a per-project basis.

Microscopists now appreciate that it is much easier to identify interesting features in an image by eye than it is to write instructions telling a computer how to find these features. Writing high-quality image analysis software can take years, as many researchers will attest, and rigorous testing is needed. The average coder produces $15 - 50$ errors per 1000 lines of code [101]. As such, scripts written and used by a single researcher, especially if they are used infrequently, are likely to produce erroneous results. Regular testing helps identify and eliminate coding mistakes.

The first scientific software in microscopy to achieve general use, and consequently rigorous testing, were visualization packages. Many researchers still use ImageJ[2] to examine and analyze their microscopy data. As more scientists learned to code, they began plugging their custom routines into established packages, allowing for reuse, testing, and further development and extension by peers. This group development of software, termed open-source software development, has allowed myriad complex analysis routines to arise

[1] Aha! Waldo was hiding in the nucleus.

[2] `https://imagej.net/`

and stabilize after years of testing. Many scientists use ImageJ, SciPy[3], and established R packages to perform analysis routines daily. Analysis specific to a particular research question can borrow from established routines, which typically feature fewer bugs.

Modern scientific software has to balance accuracy of provided algorithms with ease of use. As programming becomes more common in the biomedical sciences, this trade off is true for both users accessing an algorithm via a graphical user interface and developers calling the algorithm as a function. While all packages have their strengths, no commonly used scientific visualization software balances all of these criteria perfectly.

PYMEVisualize, described in Chapter 4, emphasizes accuracy and provides a complete list of standard SMLM algorithms. These all have nice, self-contained interactions with the visualization window, which makes it possible to quickly look at and manipulate results. However, because of its comprehensiveness and limited documentation, it can be difficult to figure out how to use desired PYMEVisualize routines. From a developer perspective, even though there is a plugin framework, the program is sprawling, and entry points can be non-obvious. These issues may be solved by getting feedback from a large group of users.

PYMEVisualize is specifically designed for SMLM, which limits its potential user base. SMLM programs, such as PYMEVisualize, SMAP [102] and Picasso [98], tend to spread researcher-to-researcher, depending on which software one comes into contact with first. Users are loathe to switch away from software packages they are familiar with and know work for their data analysis pipeline.

ImageJ is aimed at all microscopists and therefore all imaging modalities, and focuses on usability. Despite its emphasis on voxel-based images, some SMLM plugins for ImageJ exist and are in use [103]. It is fairly easy to extend and develop, providing support for multiple languages (Java, Jython, IM Macro, etc.) and multiple obvious entry points,

[3] `https://scipy.org/`

depending on the level of control desired. However, because it is written in Java, the numerical libraries available are a bit tricky to use and provide access to only low-level operations (e.g. add two vectors). This makes writing complex analyses more time consuming in ImageJ than in some other packages, but this barrier has the advantage of self-limiting the overlap of available plugins.

The Python ecosystem has a large set of easy-to-use, high-level numerical and analysis libraries. napari is a new Python package that, like ImageJ, is aimed at all microscopists. Unlike ImageJ, and like PYMEVisualize, it features point and surface rendering layers in addition to image rendering layers, making it possible to display SMLM point clouds. napari is easy to use and to develop. Part of the reason for this is that it does not aim to provide any analysis algorithms, but only the framework to do analysis and visualize the results. This encourages the development of disparate plugin packages, leading to less rigorous testing, and indeed, despite the package being new, it is already possible to see overlap in available plugin functionality.

Ironically, the proliferation of open-source software has driven some microscopy labs back toward internal software development. Relying on packages that may be unmaintained, may have a long review process for a necessary code change, or that may spontaneously introduce breaking changes in a new release can be impractical.

Software choices for each research project must be made and the software developed well in advance of any publication in order to allow for proper testing and bug removal. Otherwise, there is a risk of producing an incorrect result after multiple years of painstaking scientific work. PYMEVisualize is the result of over 10 years of continuous development, contains the most comprehensive list of SMLM analysis techniques of any package out there, and is actively maintained, making it a solid choice for SMLM analysis.

6.3 Perspective on SMLM-specific surface fitting

The novel mesh generation technique for SMLM data described in this thesis can be used to quantify membrane-protein interactions at the subcellular level. The surfaces are useful not only for studying membrane curvature, but for studying the relationship of a membrane with other membranes and/or proteins. Studying curvature-sensing and curvature-inducing proteins *in vivo* can reveal a larger pattern of self-organization within the cell. As live-cell SMLM kinks get worked out, it may be possible to image proteins binding and unbinding from membranes and to watch the surface change shape. This could help scientists understand the mechanisms behind many diseases, including those mentioned in the introduction of this thesis, and to study the mechanisms of drugs acting to correct cellular behavior and restore health.

There are a few more things I'd like to do to prove this algorithm works well. I'd like to show protein locations along a membrane surface and quantify the membrane curvature near them, perhaps in a sample containing a labeled curvature-associated or curvature-inducing protein alongside a labeled membrane. The case for using this algorithm will also be stronger if the curvature values we get from our meshes are comparable to curvature values seen in EM segmentation of surfaces of membranes. I'd like to acquire a few good EM data sets to make this comparison.

The curvature force relies on an input bending stiffness, which influences the measured curvature values in the fit surface. Since it takes a single bending stiffness, the algorithm will not capture local fluctuations in liquid vs. gel phases of the membrane and instead will yield a range of curvature values that average the influence of these regions. With sufficient surface area, the statistical distribution of curvature should still converge to reasonable values. For a sufficiently dense SMLM point cloud, it may be possible to produce a reasonable surface without the curvature constraint, which would capture local

fluctuations in curvature between different lipid phases.

The curvature force scaling parameter λ appears to scale with point density, suggesting a need for normalization of the point attraction force by the number of localizations acting on a particular mesh face. The scaling parameter λ remains somewhat arbitrary, requiring fine tuning for each sample. Fortunately, it is not terribly difficult to make a good choice for λ. Undertuning causes the surface to appear rough, and overtuning pushes the surface through the points and forces it to collapse in on itself–this is likely due to the degenerate curvature solution: a surface of area 0 has minimal curvature. For a reasonable value, the surface appears to converge, barely changing for many iterations. The surface stays in the middle of all nearby localizations and appears smooth, suggesting an ideal λ will produce both likely curvature values for a Boltzmann-distributed energy and likely position values for a Gaussian-distributed location. This information might be used to automatically select λ.

There are many computational alterations and improvements that can be made to this technique. We developed a custom meshing library for implementation, but any graphics library can incorporate this algorithm with ease[4]. The algorithm is currently set up to run on a single processor core, and, while presently somewhat quick, could be sped up by orders of magnitude by pushing calculations to a graphical processing unit. Quad meshes are often more numerically stable than triangular meshes and may allow users to get even more accurate results from this routine.

Because this method desires to have good fidelity to SMLM data, high background is still a challenge. Coupling this technique with other filters could prove useful. Shape features and/or cluster-based denoising could be used to distinguish background localizations from real data prior to wrapping.

[4]In fact, it may have been quicker and more optimal to write this in CGAL (`https://www.cgal.org/`).

6.4 Toward investigating causality at the nanoscale

The goal of my work was to create tools that allow future researchers to visualize and describe the mechanisms of membrane-associated diseases. To do this, researchers must not only be able to image proteins interacting with membranes, but find a way to show that certain proteins are causing these membrane shape changes (or that membranes change shape and certain proteins bind to and stabilize these changes). This is termed causality analysis, and it is burgeoning area of research in cell biology [104, 105].

In particular, we can use Granger causality analysis to determine if what we see in one part of an image time series influences what happens next. A big challenge when using Granger analysis is that it can be difficult to decide where to split a time series. Using the tools described in this thesis, we can easily separate videos of cells into before and after membrane shape changes. To test the hypothesis that protein X modifies membrane shape, we can look for membrane displacements larger than the size-scale of protein X clusters, split the time series at this point, and then run Granger statistical tests.

Establishing causality may help researchers identify which proteins are responsible for membrane-associated maladies, leading to new drugs that render Alzheimer's, heart disease and other scourges as hazy demons of the past.

Appendix A

Appendix for Chapter 3

This section includes supplementary material for Chapter 3, should anyone wish to build the microscope described there.

A.1 Alignment

A.1.1 System

The alignment procedure for this microscope is standard as for all other widefield microscopes[1]. Use of an alignment laser is recommended. Be sure to Köhler illuminate if the transmitted light path is used for alignment.

Aligning the OI-DIC assemblies, polarizer and analyzer orientation is non-standard, and described below. The reader is referred to Figure 3.2 for identification of optical elements referenced here. The external OI-DIC assembly is found outside of the microscopy body, while the condenser OI-DIC assembly is found just above the microscope condenser.

1. Disconnect the liquid crystals so no voltage is applied.

[1] See, for example, `https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=12211&tabname=Alignment`

2. Take out the both OI-DIC assemblies.

3. Align the condenser polarizer parallel to the condenser quarter-wave plate. The output polarization should be linear and oriented vertically or horizontally.

4. Turn the external polarizer to get the complete extinction.

5. Turn the external polarizer by 90°. Image will be the brightest.

6. Place the external OI-DIC assembly into the beam. Rotate it to get the minimal intensity.

7. Turn the external polarizer back.

8. Turn the external OI-DIC assembly by 45°.

9. Put back the condenser OI-DIC assembly.

10. Move the position of the knob that shifts the prisms in the DIC assembly laterally to get the minimum intensity on the camera. The position should be about in the middle of the movement range. Move the position of the external DIC assembly axially to get an even intensity across the field of view of the camera.

11. Reconnect the liquid crystals so voltage can be applied.

This is a course alignment. The final tuning will be done during the per-sample calibration (see Section A.1.3).

A.1.2 Liquid crystal calibration

While the liquid crystal rotators must only move between 0° and 90° (low vs. high voltage), the delay produced by the liquid crystal retarder must be calibrated as a function of voltage. This can be done following the procedure in Appendix B of [45].

A.1.3 Per-sample calibration

The OI-DIC system must be calibrated on a per-sample basis to ensure a correct setting of the zero bias position. The procedure is outlined below. A description of the software implementation of steps 3-6 of this process can be found in Section A.3.

1. Place the sample in the microscope stage.
2. Köhler illuminate the sample.
3. Set the liquid crystal voltage so the liquid crystal retarder is in the middle of its linear response phase and the rotators are set to a shear direction of $-45°$.
4. Rotate the condenser polarizer until the image of the sample is at minimum intensity.
5. Switch the rotators to a shear direction of $+45°$.
6. Iterate through the linear voltage range of the retarder to find the voltage value that minimizes image intensity
7. If the retarder voltage that minimizes image intensity at $+45°$ is close to the voltage of the zero bias at $-45°$, and the minimum intensities achieved in both shear directions are also close, stop. Otherwise, adjust the bias of the external OI-DIC assembly slightly by rotating its knob and repeat from step 3.

A.2 Measuring the influence of the dichroic angle on contrast

The two main concerns about inserting a dichroic in the OI-DIC path were 1) the dichroic might be optically active 2) it might change the intensity of s- wave more than the p- wave or vice-versa.

Optical activity

A de Sénarmont compensator was used to investigate optical activity, as described in Section 2.2 of [44]. In a slight modification to the procedure, the DIC prism on a linear stage was swapped for the dichroic mirror on a rotation mount. The orientation of the analyzer at the minimum intensity value was investigated as a function of filter angle. No change was observed for angles $0 - 45°$, as shown in Figure A.1.

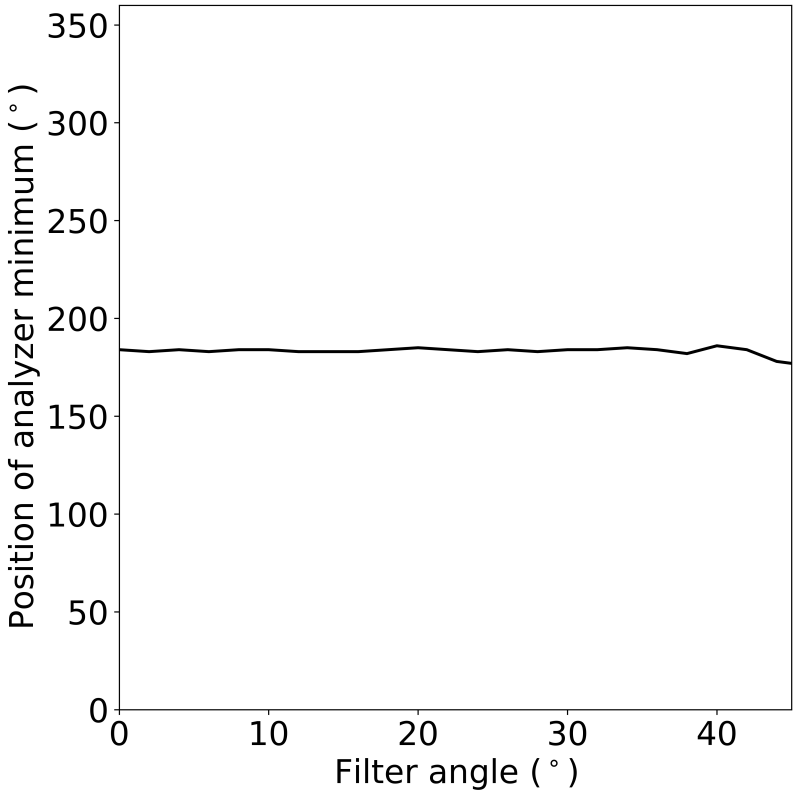


Figure A.1: Position of analyzer at minimum intensity as a function of ET546 angle of incidence.

s- vs. p- intensity

The setup to measure s- vs. p- transmission intensity as a function of angle is shown in Figure A.2. The calibration procedure for this setup is listed below.

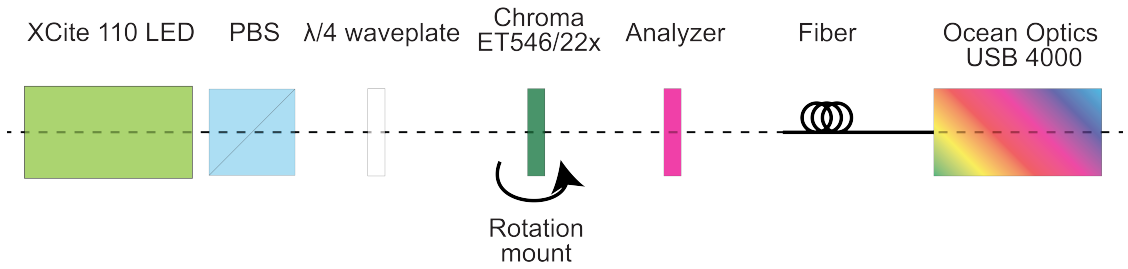


Figure A.2: Setup for measuring the full optical spectrum transmitting through an ET546 filter at various angles.

1. Place a PBS in front of the XCite source (at 5% intensity), passing horizontal light (parallel to the optical table).

2. Cross the analyzer with the PBS (this puts the analyzer at vertical orientation).

3. Swap the PBS for a polarizer and cross it with the analyzer.

4. Place the QWP in the path and rotate until transmission intensity on the fiber is at a maximum.

5. Iterate between vertical and horizontal ($-90°$) orientation of the analyzer and tweak the position of the QWP until the fiber reads roughly the same value at both orientations.

6. Place the ET546 in the path at normal incidence ($0°$).

7. Iterate between horizontal and vertical orientation of the analyzer again, adjusting the QWP until the intensity value at 546 nm is roughly the same and maximal.

8. With the analyzer in the vertical position (s-polarization), record the spectra passing through the ET546 at 0-45 degrees with 5 degrees of separation. Ocean Optics USB4000 Parameters: 15 ms integration time, 10 scan average.

9. Rotate the analyzer 90° (p-polarization) and record the spectra passing through the ET546 at 0-45 degrees with 5 degrees of separation.

The ambient light signal in the room (XCite off, T_{ambient}) and the signal in the absence of the ET546 filter (T_{air}) were also measured.

DIC image contrast is defined in [44] as

$$C = \frac{I_{\text{max}} - I_{\text{min}}}{I_{\text{max}} + I_{\text{min}}} \tag{A.1}$$

where I_{max} and I_{min} are the maximum and minimum of an image, respectively. Maximum intensity is achieved when the s- and p- waves are in phase (0°) and minimum intensity when they are out of phase (180°). Intensity in a DIC image can be approximated using the standard model for interference of optical waves

$$I(\theta_s - \theta_p) = I_s + I_p + 2\sqrt{I_s + I_p}\cos(\theta_s - \theta_p) \tag{A.2}$$

where I_s, θ_s and I_p, θ_p are the intensities and phases of the s- and p- waves, respectively [34]. This can be substituted into Equation A.1 where $I_{\text{max}} = I(0^\circ)$ and $I_{\text{min}} = I(180^\circ)$.

Using the collected spectra $T_{\text{ET546}}(\theta;\lambda)$, define $I_s = T_{\text{ET546}}(\theta;\lambda_{s,p}) - T_{\text{ambient, s}}(\lambda_{s,p})$ and $I_p = T_{\text{ET546}}(\theta;\lambda_{s,p}) - T_{\text{ambient, p}}(\lambda_{s,p})$ where $\lambda_{s,p}$ is the range of wavelengths where transmission is $> 50\%$ of the maximum value in both the s- and p- spectra, and $T_{\text{ambient, s}}$, $T_{\text{ambient, p}}$ are the ambient spectra s- and p- components, respectively. Contrast as defined by Equation A.1 and based on the collected spectra is plotted as a function of θ in Figure A.3. Notice that the contrast is $> 96\%$ from $0 - 20^\circ$ AOI.

To confirm it is fine to use this filter at 20°, normalized transmission spectra are plotted for this AOI in Figure A.4. Normalized transmission spectra $T_{\text{norm}}(\theta;\lambda)$ for the ET546 at

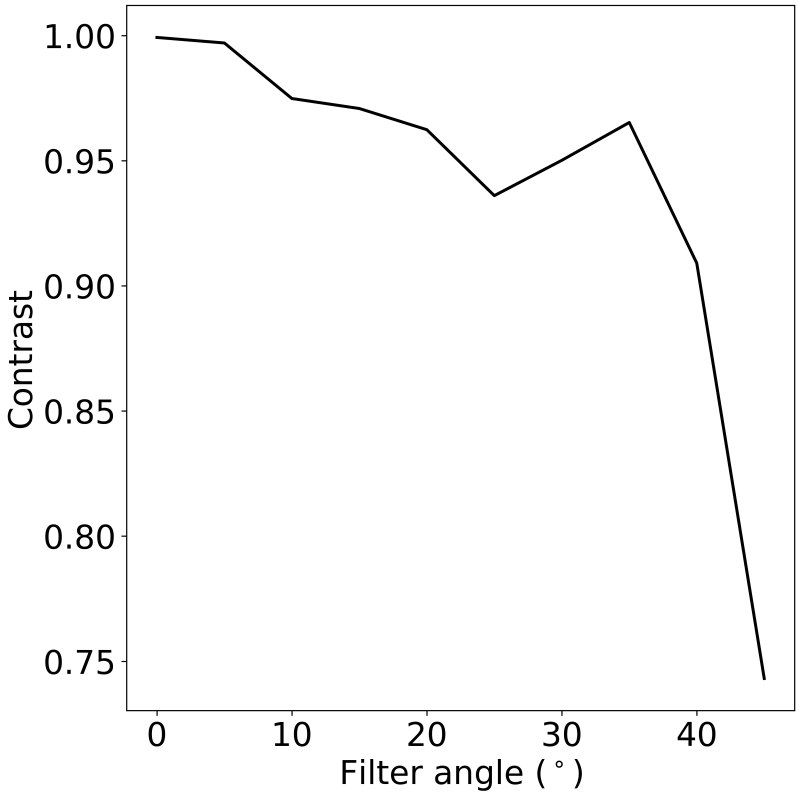


Figure A.3: Contrast as a function of angle incident on ET546.

various angles θ and wavelengths λ were calculated as

$$T_{\text{norm}}(\theta;\lambda) = \frac{T_{\text{ET546}}(\theta;\lambda) - T_{\text{ambient}}(\lambda)}{T_{\text{air}}(\lambda) - T_{\text{ambient}}(\lambda)}. \tag{A.3}$$

We can see that the signal loss is minimal over our range of interest (546 ± 11 nm), but the pass band shifts and narrows from $535 - 557$ nm to $528 - 545$ nm. The DIC prism is optimized for 546 nm, but this is sufficiently close to still work. Our LED light source in the microscope setup (see Chapter 3) emits $520 - 540$ nm, so more signal makes it through this filter at $20°$ than at $0°$.

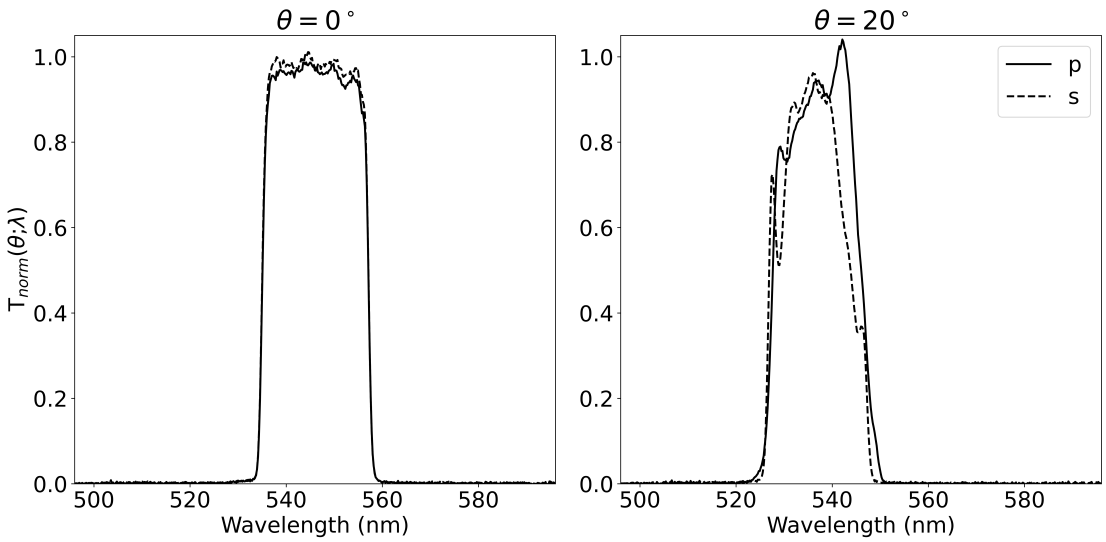


Figure A.4: Left Normalized spectra of s- and p-polarized transmitted light from 496-596 nm incident at $0°$ on an ET546 filter. **Right** Spectra incident at $20°$.

A.3 Acquisition and reconstruction software

Custom software was needed to perform acquisition and reconstruction of OI-DIC images. For acquisition, we needed to control the xy-stage, the z-piezo, the camera, the liquid crystal assemblies and the timing between them. Python interfaces were written for a pco. edge 4.2 LT sCMOS camera and an ARCoptix liquid crystal driver. Existing interfaces to the microscope stage and z-piezo were used. The graphical user interface, featuring controls and/or displays for all of these devices, is shown in Figure A.5.

When acquiring a DIC image, the following protocol must be used.

1. Apply voltages to the liquid crystals to set them to a shear angle and bias of choice.
2. Wait for the liquid crystal to settle, settling time is reported by the manufacturer.
3. If using a stage or z-piezo, wait for these to be on target.
4. Acquire an image.

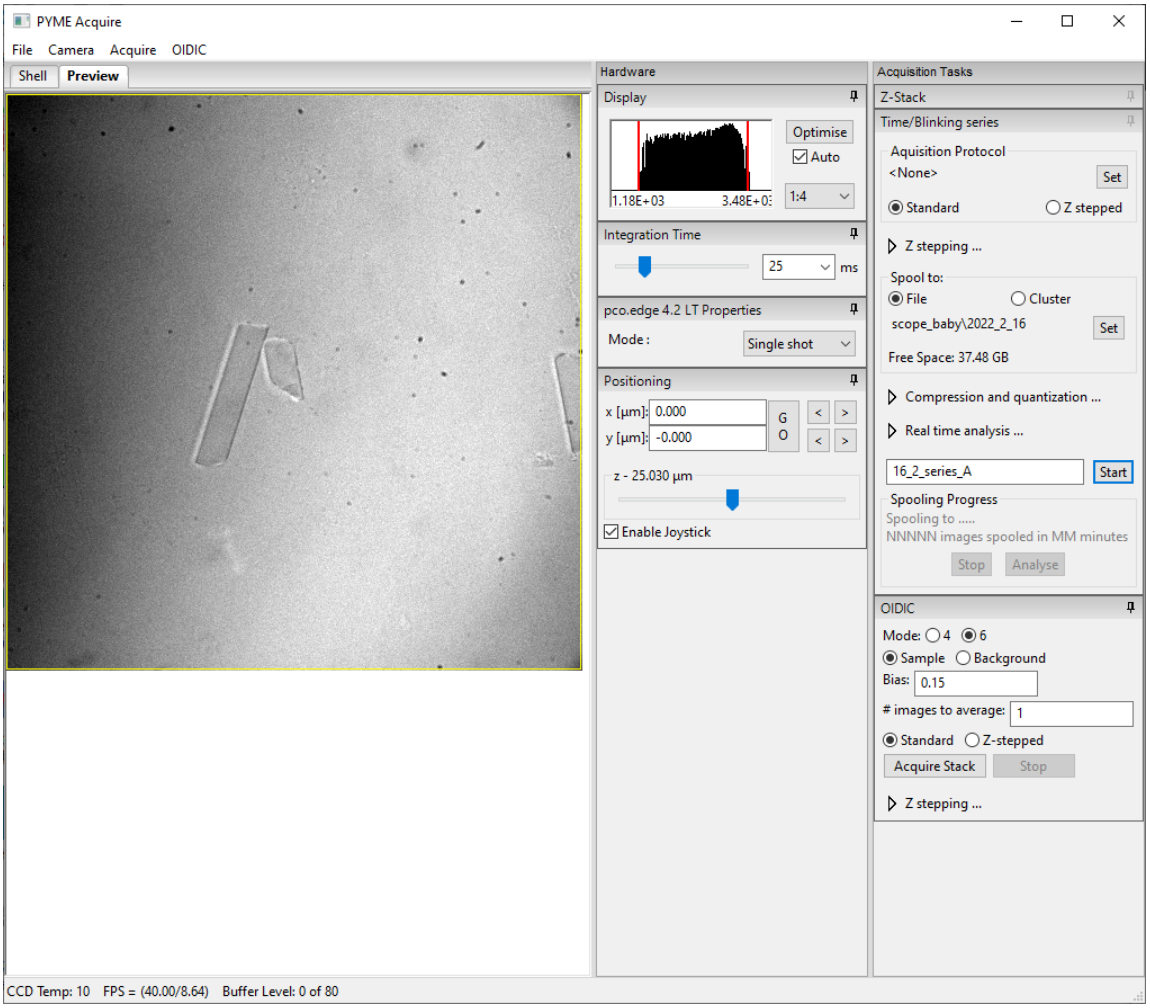


Figure A.5: OI-DIC GUI implemented in PYME. Includes a camera display and controls, x,y-stage and z-piezo controls, and single-shot and z-stepped OI-DIC imaging, featuring control over line averaging and sample bias.

Failure to wait for liquid crystals to settle will result in aberrant images, usually identifiable by strong striations in the azimuth image generated during reconstruction.

The zero bias is set, following the procedure described in A.1.3, with the help of a custom calibration window, shown in Figure A.6, and a programmed version of the calibration routine. The "Initialize" button corresponds to step 3 of this procedure and the "Run" button corresponds to steps 5-6.

Once set, the user selects a bias of choice Γ, based on the resolution of the object they are interested in imaging (see Section 2.2.1). The software automatically applies the above

DIC imaging procedure over a range of 3 biases ($-\Gamma$, 0, Γ) and two shear angles ($+45°$, $-45°$), for a total of six images.

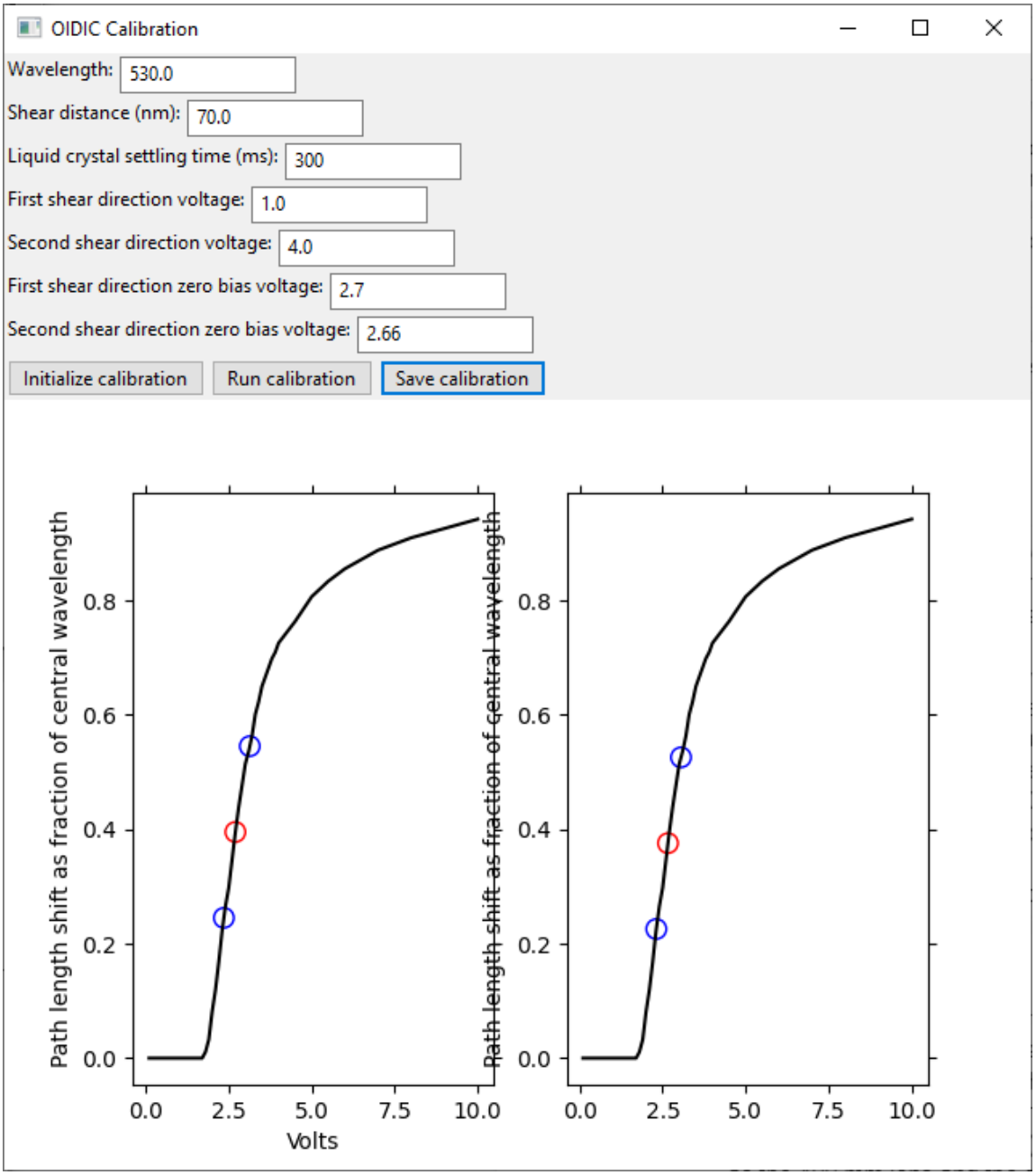


Figure A.6: Calibration GUI.

Once the DIC images are acquired at different biases and shear angles, they can be reconstructed to produce an OI-DIC image. The procedure described in [46] was imple-

mented as a Python routine and linked to the graphical user interface.

Controls and reconstruction routines were implemented as a plugin for the PYthon Microscopy Environment (`https://python-microscopy.org/`).

Appendix B

Appendix for Chapter 4

This chapter reproduces supplementary information from

Zach Marin, Michael Graff, Andrew E. S. Barentine, Christian Soeller, Kenny Kwok Hin Chung, Lukas A. Fuentes, and David Baddeley. PYMEVisualize: an open-source tool for exploring 3D super-resolution data. *Nature Methods*, 18(6):582–584, June 2021. ISSN 4159202101. doi: 10.1038/s41592-021-01165-9. URL `http://www.nature.com/articles/s41592-021-01165-9`.

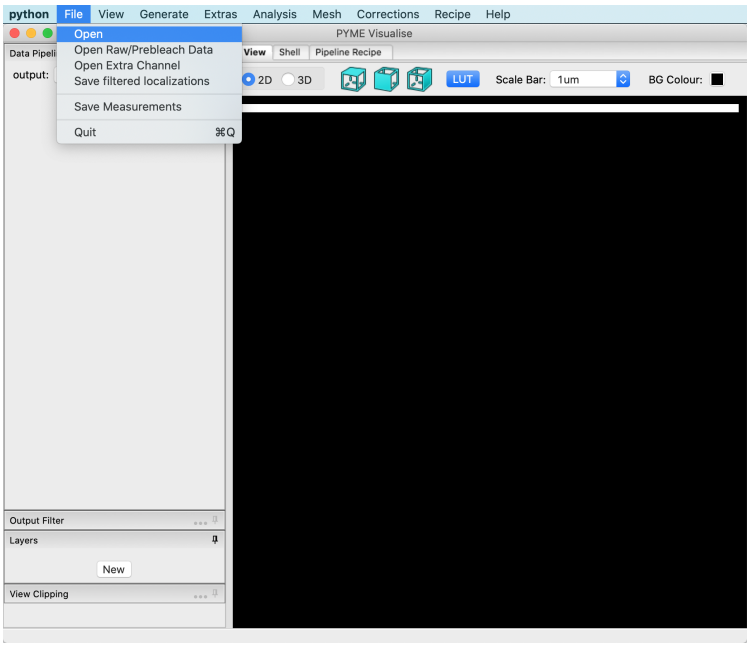
B.1 A quick tour of PYMEVisualize

In this tutorial, we'll touch on several aspects of the PYMEVisualize workflow, including opening a single-molecule localization microscopy (SMLM) data set, viewing it as points, filtering, and reconstructing density images, as well as extracting a 3D surface from the data set. The tutorial does not cover all (or even most) functionality, but rather aims to give a taste of what is possible. We assume PYMEVisualize is installed on your computer (see supplement or https://python-microscopy.org/doc/Installation/Installation.html), and that you have downloaded `test_er_data.zip` and unzipped this file to access

`test_er_data.hdf`, a supplementary SMLM data set of the endoplasmic reticulum provided with this paper.

Launch PYMEVisualize

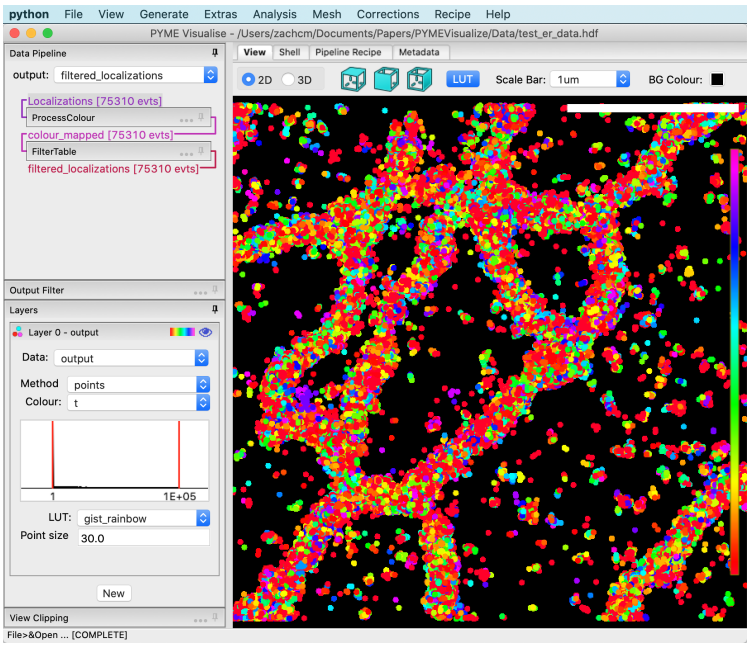
For the purposes of this tutorial, we'll launch PYMEVisualize from a command line. On windows, open **Anaconda Prompt**. On Mac, open **Terminal** [1] Once the command line is open, type PYMEVis and then press enter to launch the application [2].



You will be prompted with a file dialog asking you to *Choose a file to open*. Select `test_er_data.hdf`, which is provided with this publication. Your screen will appear as below.

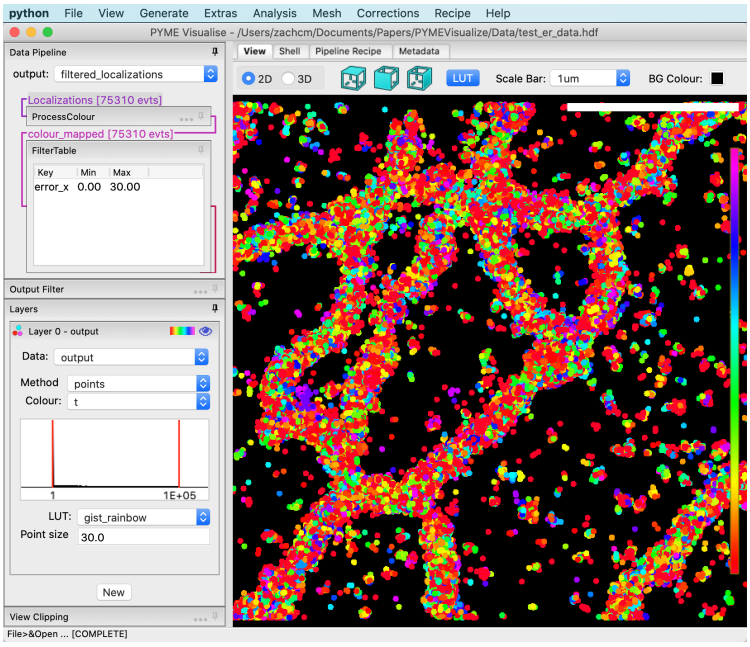
[1] If PYME is installed outside of the Anaconda base environment, type conda activate <`pyme_envronment`>. If you are unsure of where PYME is installed, assume you do not need to type this command.

[2] On Windows you should also be able to select **PYMEVisualise** from the start menu



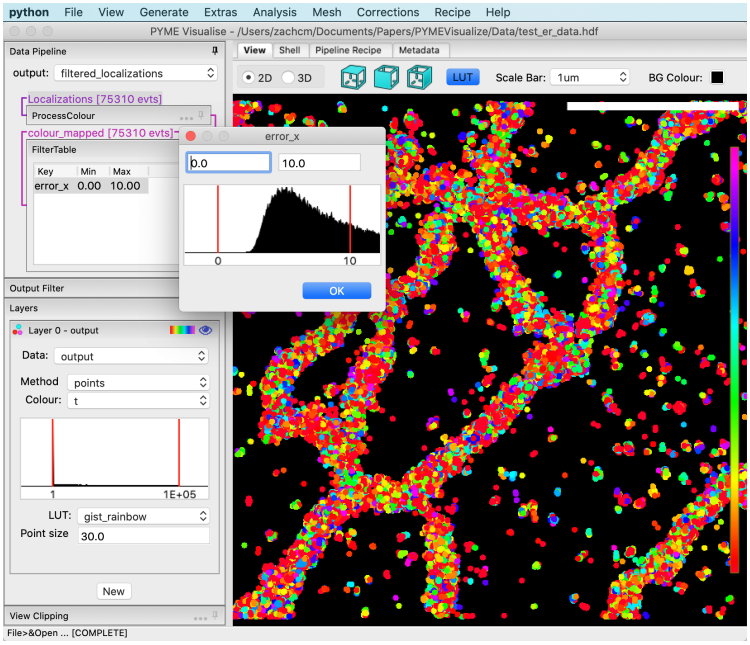
Filtering

We want to restrict our data to well localized events. Click the *FilterTable* box in the data pipeline view to expand, as shown below.



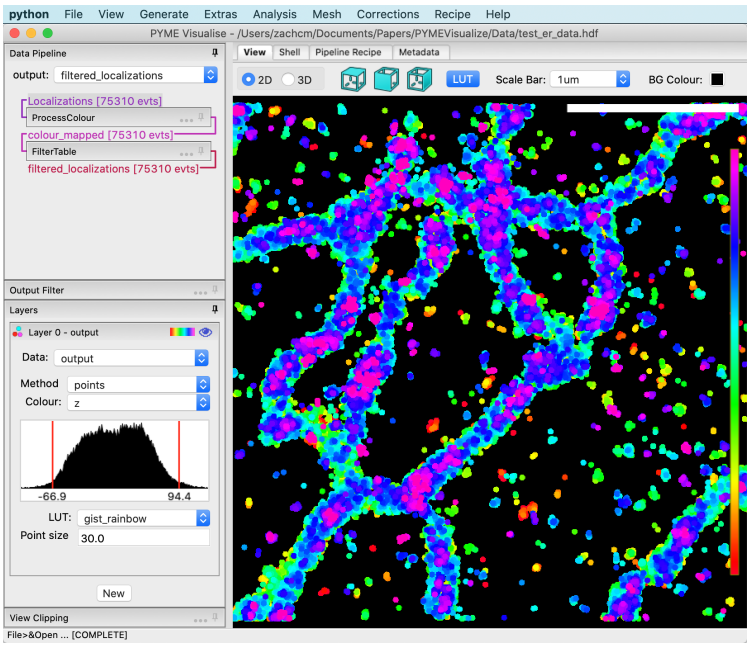
Double click on the entry for `error_x` to bring up the dialog as shown below. Drag

the right hand red bar in the histogram towards the left to restrict the localisation error to the range from 0 to 10 nm. Press OK.

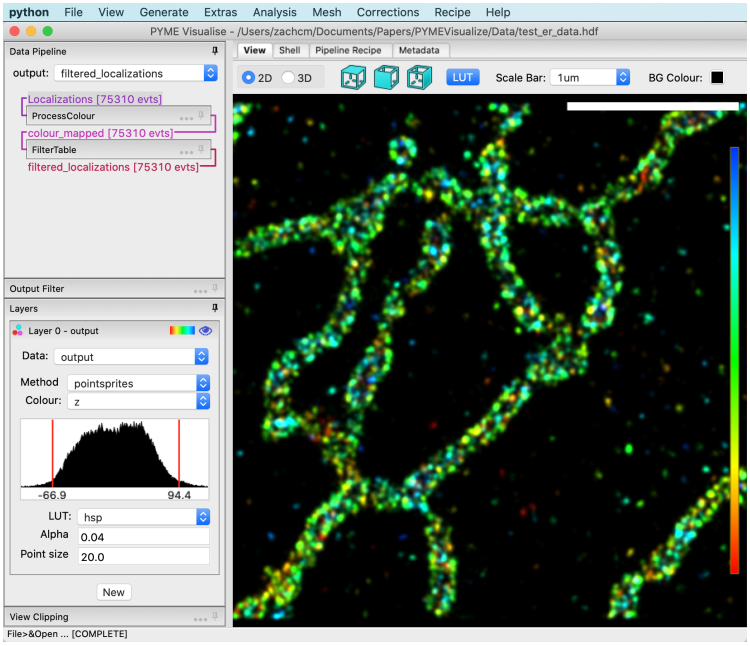


Adjust visualization parameters

By default, data opens with localisations coloured by time as this gives a quick visual indication if there is a problem with drift. The example dataset has negligible drift, so lets try some other options. Under *Layer 0 - output* Change the "Colour" to `z`, which is the axial position of the localisation, and pull the red bars on the histogram display under "Colour" in to adjust the colour scaling to exclude the outliers and make the depth changes in the structure more visible, as shown below. Alternatively, click in the histogram box and press **p** to ask PYMEVisualize to automatically set the histogram lower and upper bounds to 1st and 99th percentile of values, respectively.

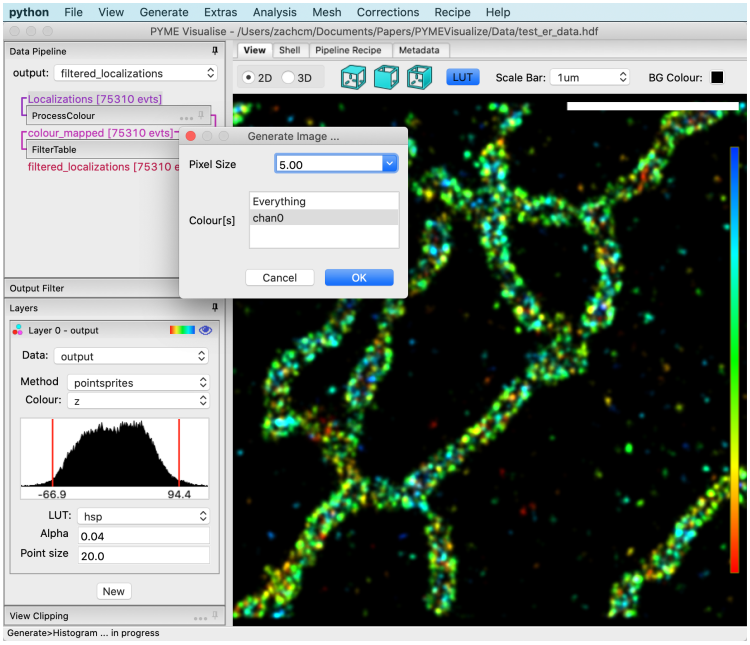


The simple `points` mode is very busy and tends to get swamped in areas of high point density. Switch "Method" to `pointsprites`, "Point size" to `10.0`, and "Alpha" to `0.2` to get a real-time approximation of the popular Gaussian reconstruction method. Switch "LUT" to hsp to get constant-intensity coloring.



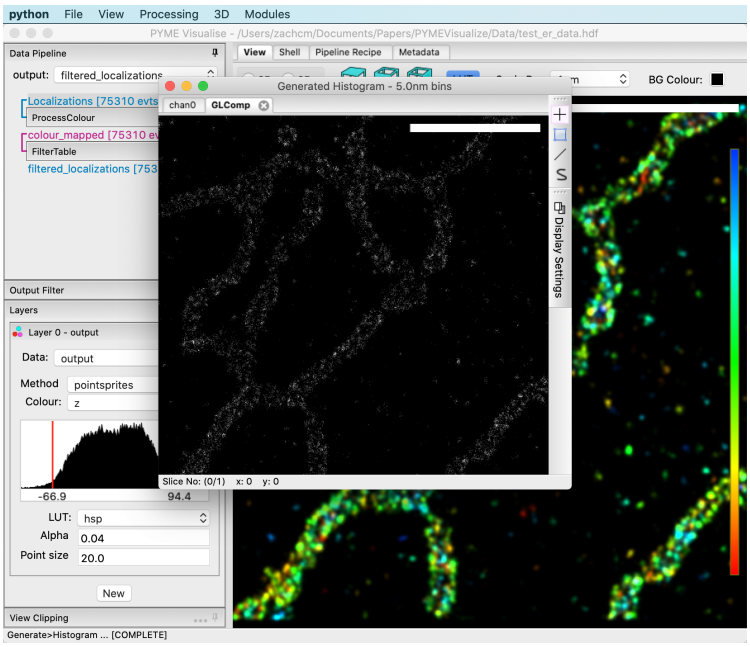
Generate a reconstruction

Let's create a 2D histogram reconstruction of our data set that we could use for pixel-based analysis. Navigate to *Generate* > *Histogram*. A dialog box will pop up as shown below.

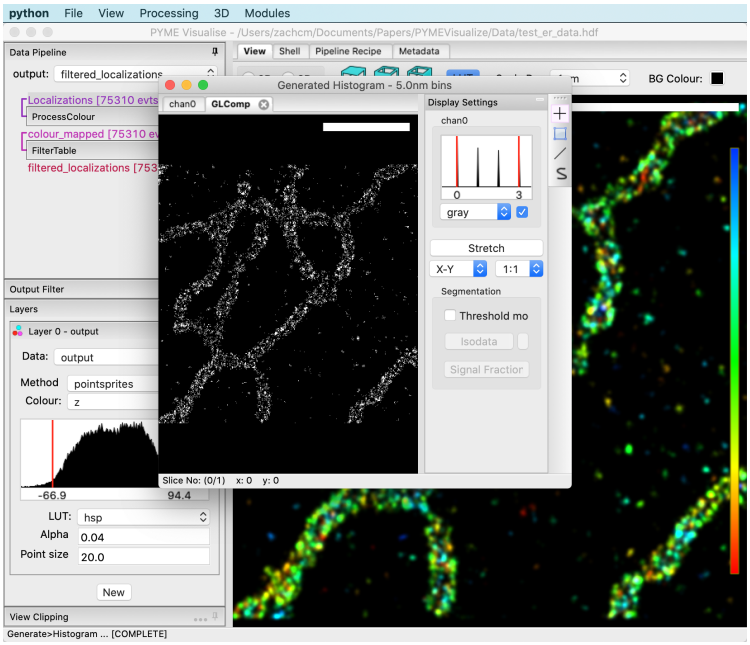


Notice that in the lower left corner of the window, it says "Generate > Histogram . . . in progress". This area of the program lets the user know what is currently running and if it is completed.

Leave the "Generate Image . . . " dialog options as default and press *OK*. A 2D histogram image will appear, as shown below.



To adjust the contrast of the histogram displayed in the "GLComp" tab, click *Display Settings* on the right of the histogram window. Pull the red bars on the histogram display under "chan0" to adjust contrast.

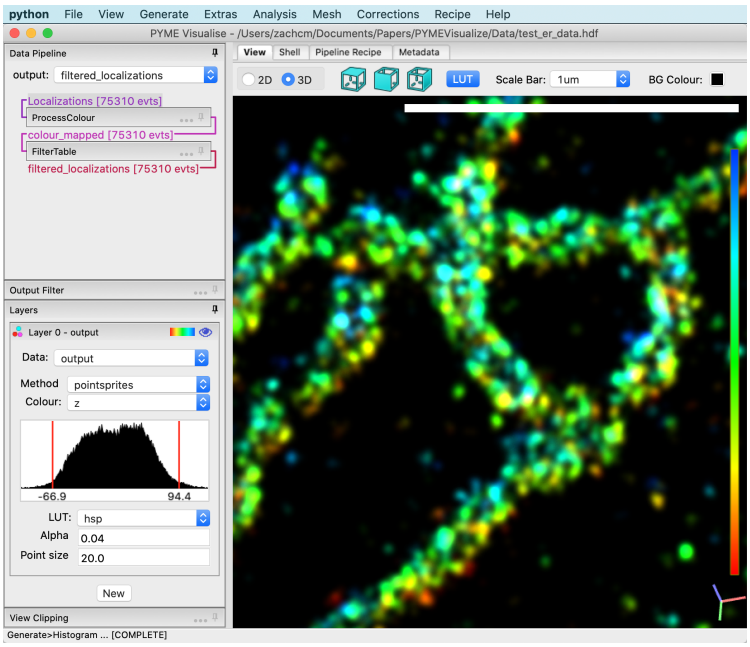


Navigate to *File* > *Save As* and name your file "histogram_rendering.tif". This is a

voxel-based image which can be opened and analysed using the same tools (e.g. ImageJ) that you might use for conventional microscopy images. The *Generate* menu also has options for a bunch of other 2D and 3D density reconstruction methods (see User Guide for details). Open in the image editor of your choice[3].

Explore data in 3D

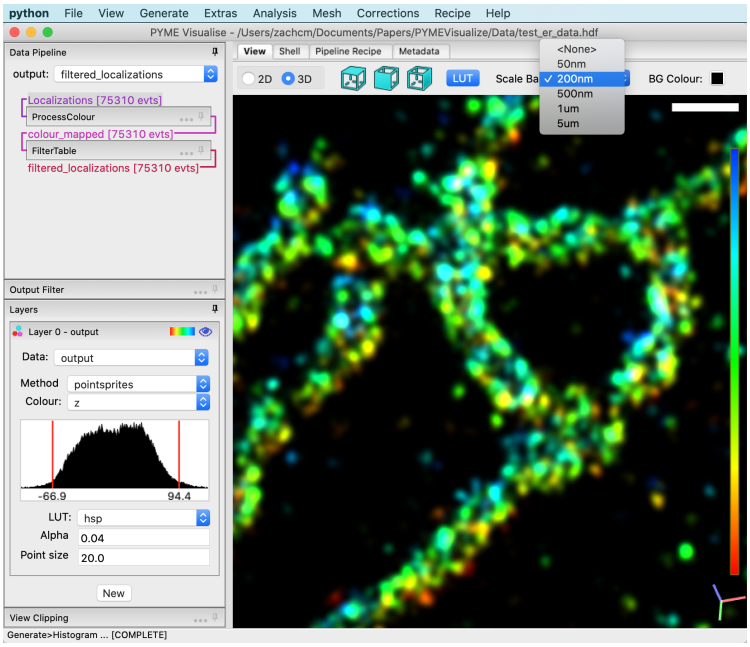
Return to the main PYMEVisualize window, optionally closing the Gaussian rendering window. Select the "3D" radio button under the "View" tab near the top of the screen. Click on the data and drag with your mouse to rotate the data view. Right-click on the data and drag to translate the data view. Rotate the scroll wheel to zoom in and out of parts of the data. Choose a rotation, translation, and zoom that you think looks nice. Ours is below.



Since we've zoomed in, the 1 micrometer scale bar is looking rather big. Change the

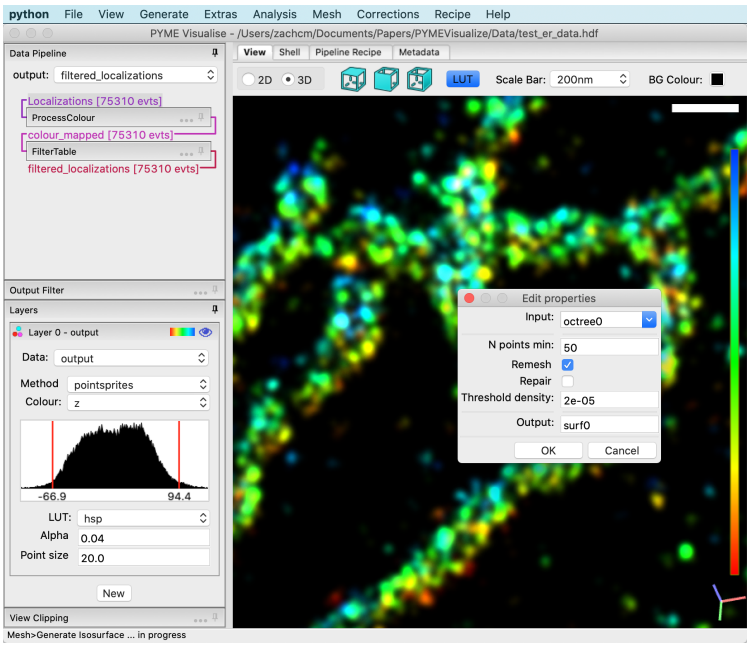
[3]Some versions of ImageJ/FIJI do not load floating point TIFF (and therefore our exported images) correctly, although the Bio-Formats importer does. If an exported .tif looks weird in ImageJ, try opening with the Bio-Formats importer.

scale bar size to “200nm”, as shown below.

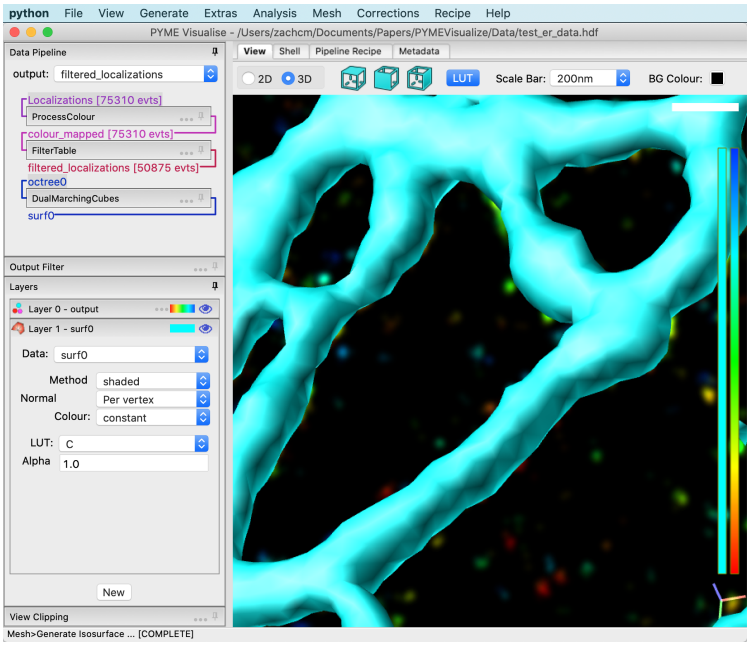


Generate an isosurface from point cloud data

This is a data set of membrane-bound proteins on the endoplasmic reticulum. As such, it is a good approximation of the membrane surface. We can generate an approximation of the ER’s surface from this data by navigating to *Mesh* > *Generate Isosurface*. A dialog box will appear. Change the properties to what is shown in the image below, then press *OK*.

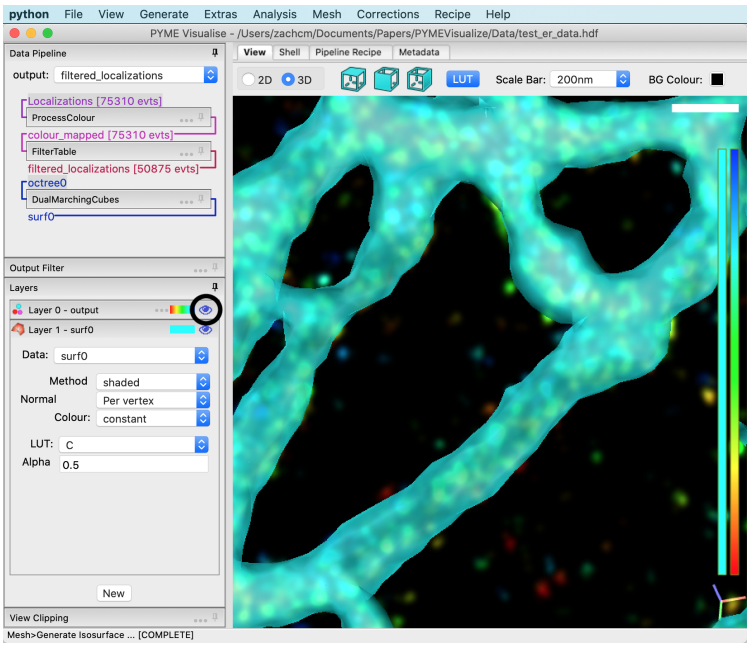


Give this 30 seconds to run. Once you see "Mesh > Generate Isosurface . . . [COMPLETE]" in the lower left corner, you are ready to proceed to the next step. The window should appear as below.



Notice that we now have two "Layers". The first is *Layer 0 - output*. We played with

this layer's parameters in the "Adjust visualization parameters" section earlier. *Layer 0 - output* is called a points layer, as indicated by its three colorful points. The second layer is called *Layer 1 - surf0*, and it is for displaying surfaces, as indicated by its red triangles. Layers stack on top of one another in the viewing area. As with the point layer, you can change how the surface layers appear - try changing "Alpha" in *Layer 1 - surf0* to `0.5`. You should see something similar to the image below.

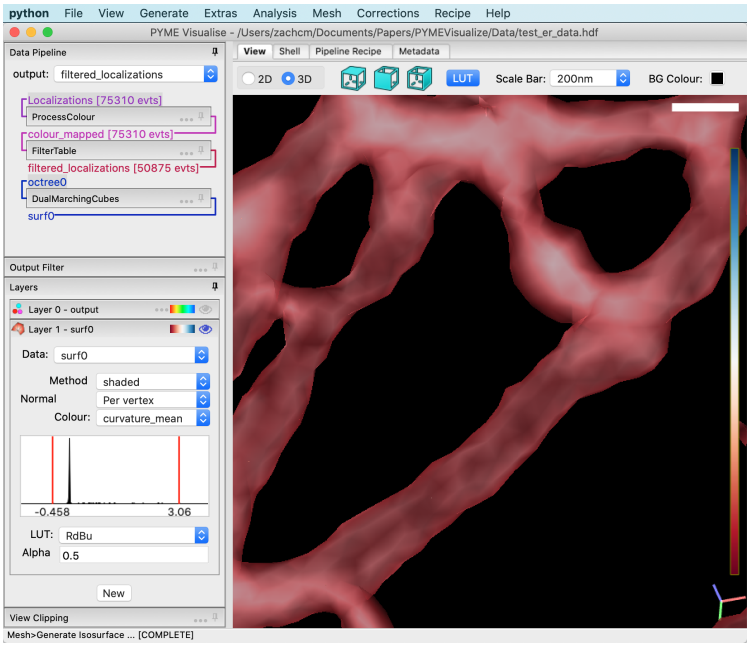


The visibility of individual layers can be toggled using the eye button associated with it. Press on the eye associated with *Layer 0 - output*, circled in black above.

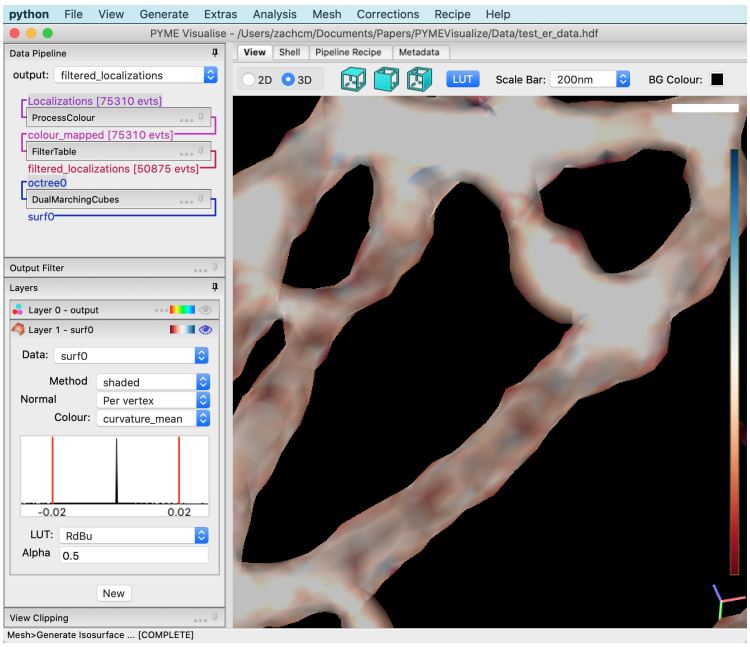
Color the isosurface by mean curvature

Like points, surfaces can be coloured by a number of different parameters. Let's color this isosurface by its mean curvature. Surfaces start off using solid colour lookup tables (one of 'C', 'Y', 'M', 'R', 'G', 'B') which display the same colour regardless of what the colour variable is, so the first thing we need to do is change the "LUT" to something which will show differences in our colour value. "RdBu" is a good choice in this case. Now, change

"Colour" to "curvature_mean", as shown below.



We don't see a lot of colour in the result, as we have a few outliers in our curvature estimates which broaden the distribution of values so that all the interesting curvature values map to one LUT point. We can change this using the histogram below "Colour". Right click in the middle of it to get a dialog box, and change the dialog box parameters to read `-0.02` for "Min" and `0.02` for "Max". Our mean curvature is expressed in units of 1/nanometer, making this range correspond to realistic physiological curvatures $\leq$ 1/50 nm.



The surface is now colored by mean curvatures between -0.02 nm^{-1} and 0.02 nm^{-1}. As expected, flat regions of the surface are white (close to 0 nm), curved surfaces are red (closer to 1/50 nm), and dips in the surface are blue (closer to -1/50 nm).

As before, we can rotate and zoom the view. Also try toggling the LUT using the toolbar button. Once you have a view you are happy with, export a snapshot using the *View* > *Save snapshot* menu item. This should give you a .png which can be viewed with standard image viewers or embedded in Word, PowerPoint, Illustrator, and other publication and presentation tools.

B.2 PYMEVisualise User Guide

B.2.1 Installation

PYMEVisualize is a part of PYthon Microscopy Environment (PYME) (`http://python-microscopy.org/`), an open-source package providing image acquisition and data analysis functionality for a number of microscopy applications, but with a particu-

lar emphasis on single molecule localisation microscopy (PALM/STORM/PAINT etc ...). There are multiple routes for installation, detailed at `https://www.python-microscopy.org/doc/Installation/Installation.html`. The simplest installation route uses a packaged installer on Windows or macOS (see Windows instructions below).

System requirements

PYMEVisualize runs on Windows, OSX, and Linux. It will run on relatively low spec machines (even a RaspberryPI), but for an enjoyable user experience we recommend:

- ~2GHz dual core CPU
- 4 GB RAM
- Hardware OpenGL support
- Wheel mouse for zooming in the interactive display

Installation on Windows using executable

1. Remove any existing PYME installs (see `https://python-microscopy.org/doc/Installation/Installation.html`.
2. Download the latest package from `http://python-microscopy.org/downloads/`.
3. Double-click `python-microscopy-XX.XX.XX-Windows-x86_64.exe`.
4. If prompted with *Windows protected your PC*, click *More info'* and then *Run anyway*, as shown in Fig. B.1 a.
5. Follow instructions in the installer, leaving all options at their default.

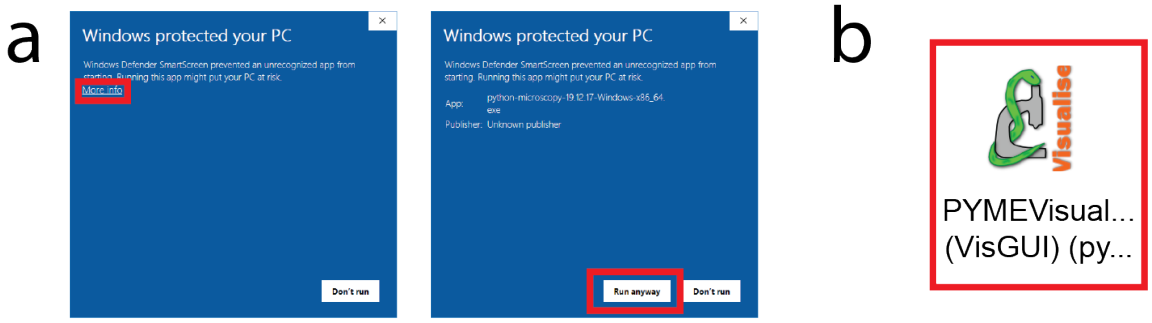


Figure B.1: Single-click installation and how to launch PYMEVisualize. (a) Potential Windows Defender error messages that can be safely ignored to install PYMEVisualize. (b) Double-click the **PYMEVisualize** logo on the desktop to start PYMEVisualize.

6. When the installation is finished, locate the **PYMEVisualise** shortcut on the desktop. Double click it to launch PYMEVisualize, as shown in Fig. B.1 b.

B.2.2 Data exploration

Importing data

PYMEVisualize is designed to work with localisation data in the formats .h5r/.hdf, .txt/.csv, and .mat. If a user has raw single molecule localisation microscopy frames, they should first process these images using one of PYME's localisation fitting routines or the program of their choice.

Localisation data can be opened using the *File* → *Open* menu command after launching PYMEVisualize, or by specifying the filename on the command line (e.g. `PYMEVis C :\\path\\to\\file.h5r`)[4].

Three data formats are currently supported for localisation data: HDF5 (**.h5r/.hdf**), delimited text **.txt/.csv**, and Matlab **.mat** files. In each case, the data should take the form of a table of values where each row corresponds to a detected single molecule event and

[4] You can also associate PYMEVisualise with a particular file type by using the "Open With" command in the windows explorer and locating the PYMEVis.exe (under `Scripts` in the directory you installed PYME to).

each column corresponds to a parameter. The .txt/.csv and .mat importers are flexible and support a variety of different column layouts, with the only hard requirements being that there are columns `x` and `y` for the position of a molecule and that all columns have equal lengths. Files may contain as many other columns as they like, and columns can be in any order. To take full advantage of PYMEVisualise, the following parameters should also be included: the time/frame number at which the event was detected, the event amplitude, event width, and the estimated localisation error, accessed through the column names `t`, `A`, `sig`, and `error_x`, respectively.

.h5r/.hdf formats The .h5r format is a custom format based on top of HDF5 and used by the analysis components of PYME to save localisation results and metadata. It has fixed table and column names and everything needed is read automatically from the file. It is significantly faster to read and has a smaller file size than .mat and .txt/.csv

The .hdf format is a slightly more generic HDF5 based format which has more freedom with how data is arranged within the HDF5 container. This is a good target for programs wishing to save data for use in PYME and avoid the performance issues inherent in saving as .txt.

Delimiter separated text (.txt/.csv) PYME supports both tab and comma delimited text files using the .txt and .csv extensions, respectively. In both cases, the column names are defined using an import dialog (Fig. B.2 a). It is possible to pre-populate the column names to speed up the process by adding a python style comment (signified by a leading `#`) to the first line of the file containing a list of delimiter separated column names. The dialog will still be shown for confirmation, but the correct column names should already be entered.

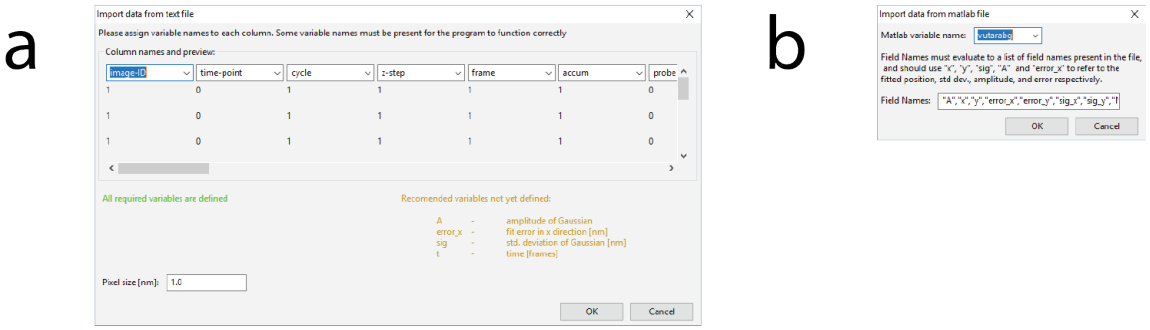


Figure B.2: Import dialog boxes for .txt/.csv and .mat file. (a) The dialog box that pops up when opening a text (.txt) or comma-separated value (.csv) file or a multi-column MATLAB (.mat) file. It lists a guess for each parameter name and the first ten values in that column. Columns can be renamed to match the recommended parameters not yet defined (yellow). The green text on the left indicates that required parameters (**x** and **y**) have been defined. (b) The dialog box that pops up when opening a single-array MATLAB (.mat) file. The name of the 2D MATLAB array containing localisation data is specified in the *Matlab variable name* box, and parameter names for each column within that array are specified by typing a comma-separated list of parameters into the *Field Names* box.

Matlab .mat files We support MATLAB files in two formats: each column stored as a separate variable in the .mat file, or all columns in a single variable (2D array). The first format is preferred. If the variable names in the .mat file correspond to the standard variable names (`x`, `y`, `z`), etc ...) described in Section B.2.2, .mat files will open automatically. Alternatively, an import dialog (Fig. B.2 a) will allow mapping of column names upon import, as described in Section B.2.2. If the .mat file contains a single array, the import dialog (Fig. B.2 b) is a little more primitive, but the same principle applies: each column needs to be given a name, and the parameters `x` and `y` must be defined. The names are specified by typing a comma separated list of parameters into the supplied box. Each of the parameters must be enclosed in double quotes, and there must be exactly the same number of parameters as there are columns in the 2D MATLAB array.

Metadata Acquisition metadata describing camera properties, localization routines, etc., can be important for quality control and analysis. Metadata is automatically loaded from .hdf/.h5r files, and improved metadata handling for other file formats is on our TODO

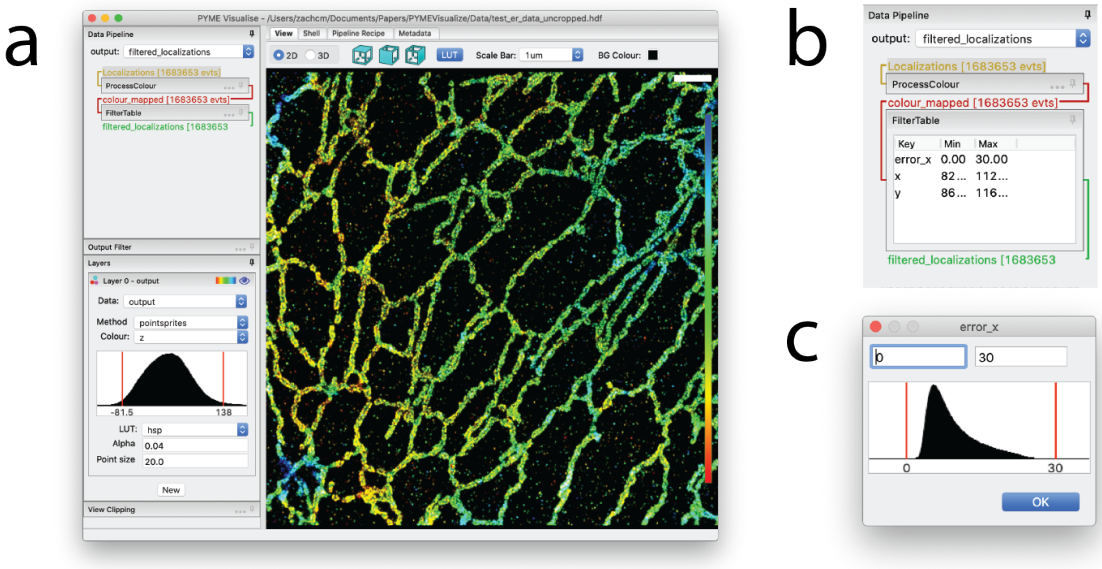


Figure B.3: The PYMEVisualize GUI with a loaded data set. (a) Interactive display of ∼1.7 million data points from a super-resolution image of the endoplasmic reticulum in a U2OS cell, courtesy of Yongdeng Zhang and Lena Schroeder. (b) The expanded filter for this image. (c) An example editing dialog for the `error_x` filter.

list. In the meantime, missing metadata can be supplied by the user in the *Shell* tab of PYMEVisualize (see B.2.11). For example, estimation of dye photophysics requires the **Camera.CycleTime** metadata entry (see B.2.3). To set `Camera.CycleTime` to 100 milliseconds, enter `pipeline.mdh['Camera.CycleTime'] = 0.100` into the shell. For more information on metadata, see `http://python-microscopy.org/doc/metadata.html`.

Having successfully loaded a dataset, the window should resemble Fig. B.3 a. If nothing is displayed, don't panic: the most common reason is that the filter (see B.2.2 section below) is throwing away all the data points.

The data pipeline

Data loaded into PYMEVisualise is processed using a configurable pipeline, accessed in the PYMEVisualize graphical user interface under the *Data Pipeline* tab (see Fig. B.3

b for an example). By default, the pipeline loads with two sections, `ProcessColour` which extract and process colour information, if present in the input, and `FilterTable` which filters on localization precision etc ... Expanding portions of the pipeline, such as *FilterTable* (see Fig. B.3 b, and the section below), allows for direct manipulation of their settings. Many of the additional manipulations accessible from the menus, such as drift correction and repeated localization chaining, will add steps to this pipeline. The parameters of these steps are then adjustable and will update the output in real-time. The entire pipeline can also be saved and re-loaded at a later date.

The filter The filter (Fig. B.3 b) restricts analysis and visualization to a subset of the data source. It allows specification of a valid range for each parameter, and points with parameters in these ranges are kept. The filter is used to discard erroneous events where, for example, the localization fit yielding the point picked up a noise spike or did not converge.

The filter is controlled from within the data pipeline in the sidebar, and can be expanded by clicking on *FilterTable*. Right clicking in the table gives you the option to add and, if a parameter is selected, edit or delete a parameter. Double-clicking on a parameter also enables editing. Editing parameters brings up a dialog, as shown for the `error_x` parameter in Fig. B.3 c. A histogram of the selected parameter is displayed and the current bounds are indicated by red vertical lines. These lines can be dragged with the mouse to change the filter bounds. The filter editor (and all other histogram editors) also understand the following keys if they have focus (i.e. if the user clicks on the histogram).

`m`	m sets the bounds to the minimum and maximum values of the variable
`p`	sets the bounds to the 1st and 99th percentiles
`l`	toggles log scaling on y-axis

The text editing boxes above the histogram can also be used to update parameter bounds. The filter will typically come with default bounds for `A` (the point amplitude), `sig` (the standard deviation of the fitted Gaussian), and `error_x` (the estimated error in

the x position). The default values in PYMEVisualize are for imaging at ∼647 nm excitation with a 1.47 NA objective, and quite likely need changing. Notably, `A` will need to be changed for different intensity calibrations, and `sig` will need to be changed when working at different wavelengths.

Colour channels

PYMEVisualise uses a probabilistic mechanism of channel assignment through which each fluorophore is given a probability of belonging to each of the colour channels present in the sample. Initially designed to support ratiometric imaging where colour assignments are not absolute, it is a flexible model which can also support simpler scenarios where channels are well separated or imaged sequentially. Colour assignment is performed by the `ProcessColour` pipeline module and three different methods of assigning colour probabilities are available: Bayesian channel assignment for ratiometric localisation data, temporal assignment for sequentially localised fluorophores, and pre-assignment using a `probe` column for imported data where channel assignment has already been performed. The method of colour assignment will be chosen automatically based on the file metadata and the presence of columns named either "probe" (pre-assigned), or `gFrac` [5] (ratiometric). Under the hood, these all feed into the probabilistic colour model resulting in special `p_<channel_name>` columns. If multiple color channels are detected, PYMEVisualize will automatically generate layers (see Section B.2.2) for each color channel when the file is loaded, in addition to the standard layer showing all points. See Section B.2.13 and Section B.2.13 for details on ratiometric colour processing and channel extraction for non-colour aware processing routines.

[5] Corresponding to the ratio of short channel to total intensity for a single event.

ROI selection / the "Output Filter"

The "Output Filter" [6] is located immediately below the data pipeline. It is similar to the filter within the pipeline, but operates after all other processing steps and immediately before display. Its primary use is for cropping the data to a smaller spatial ROI by adding filters on the `x` and `y` parameters. Rather than manually creating and setting these filters, a selection can be made by clicking and dragging with the left mouse button within the view tab (a yellow selection rectangle should be shown), and then clicking on *Clip to Selection* in the *Output Filter* pane (or pressing `F8`). The ROI can then be cleared by clicking the same button (or by pressing `F8` again).

Interactive display

The processing pipeline feeds into the interactive display (Fig. B.3 a). By default the display shows a single **"Points"** layer which renders the processed localisations as a point cloud. Points layers (see, e.g. Fig. B.3 a) support a number of different display modes, from simple dots, through shaded spheres, to transparent Gaussians (point sprites), which provide a real-time approximation to the popular Gaussian reconstruction mode. Points can be coloured by any of the fitted parameters (via the *Colour* dropdown), with a variety of different look up tables (*LUT*) and with adjustable size and transparency. Extra layers can be added to simultaneously visualise different steps in the processing pipeline, colour channels, or data types. In addition to the **Points** data type, there are layers for rendering triangular meshes/surfaces, octrees, single particle tracks and voxel-based image data.

The display can be zoomed in and out using the mouse wheel, and panned by dragging with the right mouse button. Choosing *View* $\rightarrow$ *Fit* from the menu will reset the display such that the whole data set fits within the display window. Pressing `C` recenters the data

[6] This name is historical, and refers to a time when this was the only filter in the workflow. It will probably be renamed to ROI at some point in the future.

bounding box on the current view. A scale bar and color lookup table are on the right of the display window.

B.2.3 Data correction and quality control

Chaining

A fluorophore that is on for multiple frames in the raw data will appear as a series of localizations at sequential times. To group localizations close in space and time into single events, run *Corrections* $\rightarrow$ *Chaining* $\rightarrow$ *Find consecutive appearances*. A dialog will appear allowing chaining options to be set.

`class FindClumps`

> **Clump radius** is the maximum spatial distance between chained localizations. The default is twice the localization's lateral fit error (a 2σ should correctly link 95% of localisations).
>
> **Time window** is the maximum temporal distance (in frames) allowed between chained localizations.

Pressing *OK* in the dialog will then identify which localizations are likely members of a chain, but will not replace the members of the chain with a single grouped/averaged localisation. This is done in a separate step, *Corrections* $\rightarrow$ *Chaining* $\rightarrow$ *Clump consecutive appearances*.

Drift correction

PYMEVisualise supports 3 forms of drift correction out of the box, with additional algorithms available as plugins. The builtin methods are as follows:

Fiducial based drift correction This uses fiducials localised along with the blinking molecules to correct drift, and assumes that the fiducial localisations are present in a different dataset to the single molecule localisations (as optimal detection, background subtraction, and fitting settings are likely to be different for fiducials and molecules). If the data was analysed using PYME, both these datasets should be in the same file, and running fiducial based correction should be as simple as selecting *Extras* $\rightarrow$ *Fiducials* $\rightarrow$ *Correct* from the menu (and potentially entering the fiducial diameter to permit filters to be set accordingly). If fiducial and single molecule datasets are not in the same file, you will need to load the localisations first and then run *Extras* $\rightarrow$ *Fiducials* $\rightarrow$ *Load fiducial fits from 2nd file* to load the fiducial fits. The algorithm extracts fiducial traces, and aligns and averages the traces from multiple fiducials (weighted by localisation precision). It tolerates small gaps in the fiducial traces as long as not all fiducials traces are broken in the same frame. After correction is complete, *Extras* $\rightarrow$ *Fiducials* $\rightarrow$ *Display Correction Residuals*, will show the residuals (error between each fiducial and the average correction) which gives an indication of correction quality.

Autocorrelation based drift correction Accessed as *Corrections* $\rightarrow$ *Autocorrelation based drift correction*, this is essentially an implementation of the algorithm described in the supplement of [106], dividing the localizations into overlapping time blocks, and performing autocorrelation between those blocks.

Transmitted light correction This relies on drift measurements made during image acquisition using a transmitted light channel (see [107] - our implementation does

not assume, however, that correction of x-y drift is necessarily performed in real-time, rather saving the recorded drift values along with the image data), and requires suitable drift *event* data in the input files.

Fourier Ring Correlation

Fourier ring correlation (FRC) is an established technique for estimating the resolution of localization-based images [108]. To use FRC to estimate resolution in PYMEVisualize, first select *Extras* → *Split by time blocks for FRC*. The will create 2 fake colour channels, `block0` and `block1`, based on dividing localisations in time with a temporal block size set using the dialog in Fig. B.4 a (for multicolour data it will split the existing color channels in 2 so you will get double the number of colour channels, e.g. `chan0_block0`, `chan0_block1`, `chan1_block0`, `chan1_block1`).

Render images by choosing *Generate* → *Histogram* from the menu and selecting the two blocks corresponding to the color channel of interest (the FRC module currently assumes a single color), as shown in Fig. B.4 b. Note that in principle any image generation method (see Section B.2.4) can be used, but histogram rendering is probably the best for a pure resolution assessment.

In the rendered image window, choose *Processing* → *FRC* and select the renderings of the two time blocks to compare as in Fig. B.4 c. An FRC plot like Fig. B.4 d will appear, quantifying resolution.

Photophysics

For a given image, it is possible to estimate the photophysics of the dye or fluorescent protein used in acquisition. To do this, first run through the clump detection part of the chaining procedure described in Section B.2.3. Then select *Analysis* → *Photophysics* → *Estimate decay lifetimes*. This will display three graphs, shown in Fig. B.5, indicating

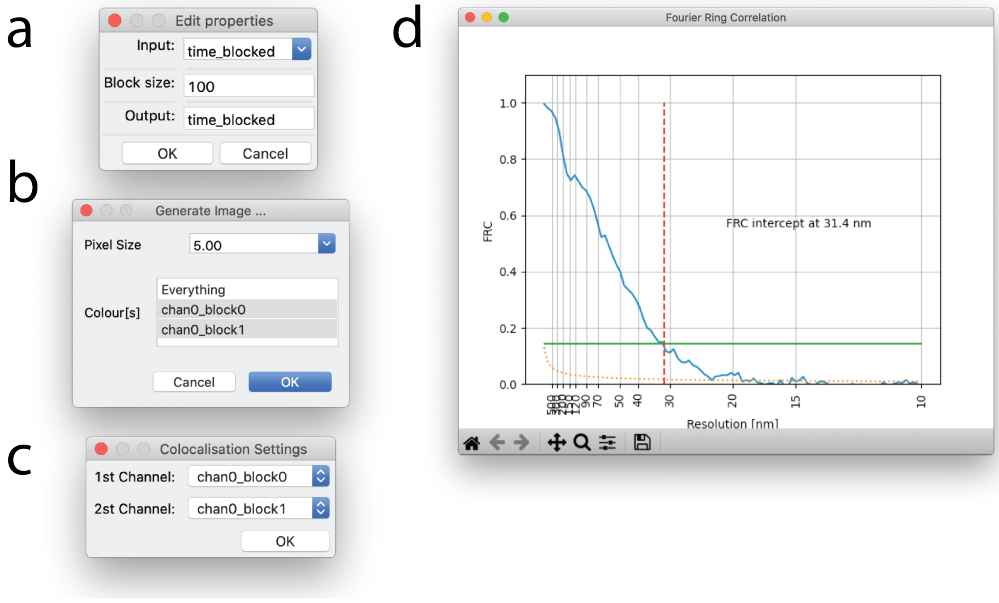


Figure B.4: Dialogs and plots in the Fourier ring correlation pipeline. (a) Dialog for *Extras* → *Split by time blocks for FRC*, used to set FRC time block size. (b) Histogram generation dialog window. Pixel size is set to 5 nm and FRC `block0` and `block1` are selected for rendering. (c) Dialog for *Processing* → *FRC*, indicating blocks to compare for FRC. (d) FRC plot for image shown in Fig. B.3 a.

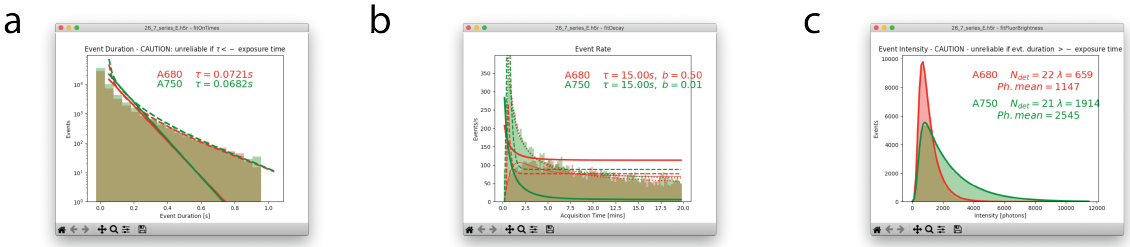


Figure B.5: Plots generated from running *Analysis* → *Photophysics* → *Estimate decay lifetimes* on data shown in Section B.2.13. (a) Estimation of fluorophore decay rate, indicated as τ in the upper right of the plot. (b) Estimation of mean number of fluorophores in an ON state per second throughout the duration of imaging, indicated as τ in the upper right of the plot. (c) Estimation of the mean mean number of photons per fluorophore in the ON state, indicated as *Ph. mean* in the upper right of the plot.

the fluorescence decay rate of the fluorophore, the mean number of fluorophores in an ON state per second throughout the duration of imaging, and the mean number of photons per frame.

Note that the metadata setting `Camera.CycleTime`, which is the integration time of the camera used to collect the raw localization data, must be present in order to analyze photophysics. See Section B.2.2 for details on how to ensure this metadata is present.

B.2.4 Image reconstruction

In many cases it is desirable to reconstruct a density image analogous to a more conventional voxel based dataset such as would be acquired by confocal microscopy. PYMEVisualise supports a number of different image reconstruction algorithms, which can be found under the *Generate* menu. The following methods are supported.

Histogram A histogram of localisation positions with a specified bin size. The simplest possible reconstruction technique.

Gaussian The popular reconstruction method which creates a density image by summing Gaussians at each localisation. By default, the estimated localisation error is

used to determine the width of the Gaussians (as introduced in [22]), but any of the fitted parameters can be used. Using the fitted event width (`sig`) instead is a simple way of generating synthetic diffraction limited images.

Jittered triangulation Described in [61], the jittered triangulation method performs a local density estimate based on a Delaunay-triangulation of the localisation data. When compared to Gaussian rendering it gives less weight to stochastic features resulting from only a few localisations and generally gives better quality segmentations when thresholded in subsequent processing. In the limit of high emitter density, it also gives better resolution (although practical emitter densities are seldom high enough for this to be relevant).

The variable which dictates the jitter magnitude can be selected, and defaults to a measure of the distance between a point and its neighbours. The number of samples to average defaults to 10.

In addition to jittering, it is also possible to smooth the triangulation by averaging several triangulations performed on Monte-Carlo subsets of the point positions. To try this out, set the multiplier for the jitter to 0 and set the MC subsampling probability to less than 1 (0.2 is probably a good start).

- **Quadtree** Also described in [61], the quadtree method renders a quadtree where the intensity of each leaf of the tree is proportional to the density of points within that leaf, dividing bins when the number of localisations contained is greater than the *Max leaf size* parameter. This leads to adaptive bin sizing where areas of the image that are localisation poor have large bins and localisation dense regions have small bins. A convenient way to think of the quadtree method is that it yields an approximately constant signal to noise (of $\sim \sqrt{\frac{max\_leaf\_size}{2}}$) across the image regardless of local point density. Like the jittered triangulation method, it helps avoid some

of the visual and segmentation artifacts obtained using the Gaussian method when sampling density is low.

3D versions of the histogram, Gaussian, and Jittered Tirangulation methods are also available. These generate a volumetric stack rather than a 2D image.

After an image is generated, it will pop up in a new window (Fig. B.6 b). Colour scaling in the viewer can be adjusted by expanding the *Display Settings* tab to the right of the image. When viewing multi-colour images, individual channels will appear in separate tabs, along with a composite tab in which the channels are overlaid. Generated images may be saved as raw values (*File* $\rightarrow$ *Save as*) suitable for quantitative analysis in downstream software, or exported as scaled and colour-mapped snapshots (*View* $\rightarrow$ *Export Current View*) for inclusion in presentations or publications. The default formats are floating point TIFF for raw data and PNG for snapshots.

Note Some versions of ImageJ/FIJI do not load floating point TIFF (and therefore our exported images) correctly, although the Bio-Formats importer does. If an exported .tif looks weird in ImageJ, try opening with the Bio-Formats importer.

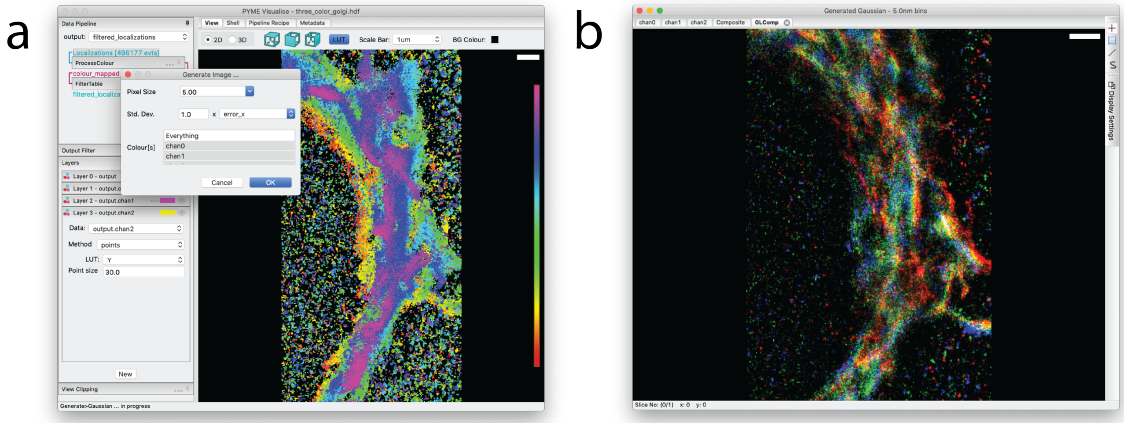


Figure B.6: 2D Gaussian rendering of a 3-color super-resolution image of *cis*, *medial*, and *trans*-Gogli. (a) A *Generate Image...* dialog specifying a pixel size of 5 nm, a standard deviation of `error_x` nm for each rendered Gaussian, and a request for renderings of all 3 colour channels. (b) An image viewer displaying a composite of the rendered colour channels. Individual channels are accessible via tabs (`chan0`, `chan1`, `chan2`) in the image viewer. Clicking on *Display Settings* will reveal a histogram that can be used to adjust colour channel contrast, among other display tools.

B.2.5 Surface extraction

Isosurfaces

Isosurfaces are a common tool for visualising volumetric voxel data sets such as those produced by confocal microscopy. The algorithms and software tools used to generate isosurfaces for confocal can be applied to super-resolution images after performing a 3D density reconstruction (see Section B.2.4). This indirect approach, however, has a number of disadvantages. To capture detail in the data sets generally requires the use of a small reconstruction voxel size, resulting in exceptionally large datasets. A 10x10x10 µm super-resolved volume with a 5 nm pixel size would give rise to an 8 gigavoxel (32 GB) reconstructed volume. This represents a major computational challenge, in practice limiting such reconstructions to small ROIs and often smoothed and downsampled data. A second limitation is the need to choose this voxel size in advance. Due to the stochastic nature of localisation microscopy, choosing an appropriate reconstruction voxel size is not

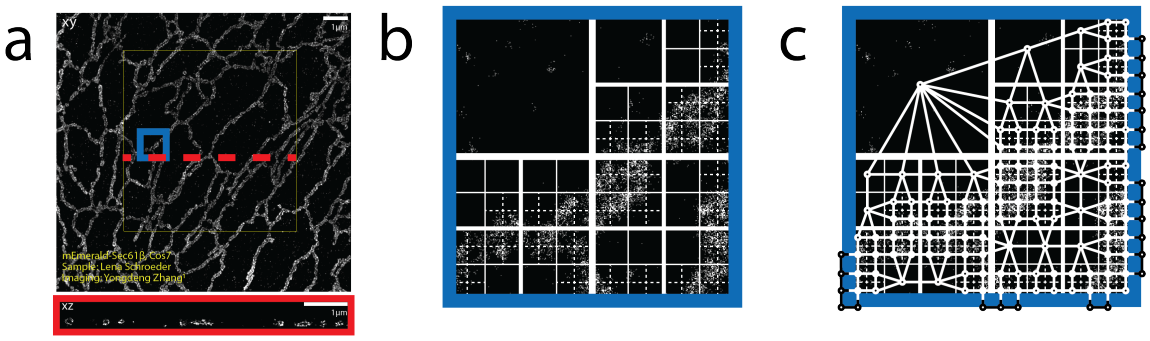


Figure B.7: Octree generation. (a) The 3D dataset from Fig. B.3 requires 900 megavoxels to sample at 5 nm intervals. To reduce memory use, we sample with a sparse octree. (b) Birds-eye view of an octree used to generate an isosurface, overlaid on a subregion of the dataset shown in Fig. B.3 a. (c) A birds-eye view of a dual grid used to generate an isosurface, overlaid on a subregion of the dataset shown in B.3 a. Each vertex of the dual grid the center of an octree leaf.

a trivial problem - different parts of the image could well have a different optimal voxel size.

In PYMEVisualise we have implemented algorithm which permits isosurfaces to be extracted much more efficiently from point datasets without a conventional image intermediate. Our algorithm initially places points into an octree data structure [64] (Fig. B.7 b). We then cut / truncate the octree at a given minimum number of localisations per octree cell (equivalent to a minimum signal to noise ratio (SNR) - see [61]). This has the effect of dividing the volume into cubic cells with a size which adapts to the local point density. Cells will be large in areas with few localisations, and small in areas which are localisation dense. The result is a volumetric data structure that contains the same information as a fully sampled reconstruction but with a lot less elements. We calculate a local density of localisations in each cell and then run the Dual Marching Cubes ([67]) algorithm on this with a given density threshold (Fig. B.7 c).

The algorithm for isosurface generation is accessible from the menu as *Mesh* $\rightarrow$ *Generate Isosurface*. This will construct the octree over which the isosurface is calculated and then display a dialog (Fig. B.8 a) allowing parameters of the isosurface generation to be

adjusted. The parameters are as follows:

`class DualMarchingCubes`

Input the octree name (do not modify)

NPointsMin the leaf size (number of localisations) at which we truncate the octree. A higher value increases the SNR at the expense of resolution.

ThresholdDensity the threshold on density (in localisations/nm^3) at which to construct the isosurface.

Remesh improves mesh quality by subdividing and merging triangles such that triangles are more regularly sized and the number of connections at each vertex is more consistent. This improves both appearance and the reliability of numerical calculations on the mesh (e.g. curvature and vertex normals). Disable when experimenting with thresholds to improve performance.

Repair will patch holes in the mesh (usually not needed).

Spherical harmonics

Another method of surface extraction from point data sets is to fit spherical harmonics (*Analysis* $\rightarrow$ *Surface Fitting* $\rightarrow$ *Spherical Harmonic Fitting* $\rightarrow$ *Fit Spherical Harmonic Shell*). In contrast to the isosurface method (which simply thresholds on density) spherical harmonic fitting assumes that points lie on a surface. Because it is model based it is much better constrained and can extract accurate surfaces from significantly sparser datasets. It is ideally suited to the extraction of the cell nuclear envelope based on a lamin or NPC staining, but is also applicable to other "blobby" structures which are shell-labelled [109]. When multiple objects are present in a field of view, these will need to be segmented first.

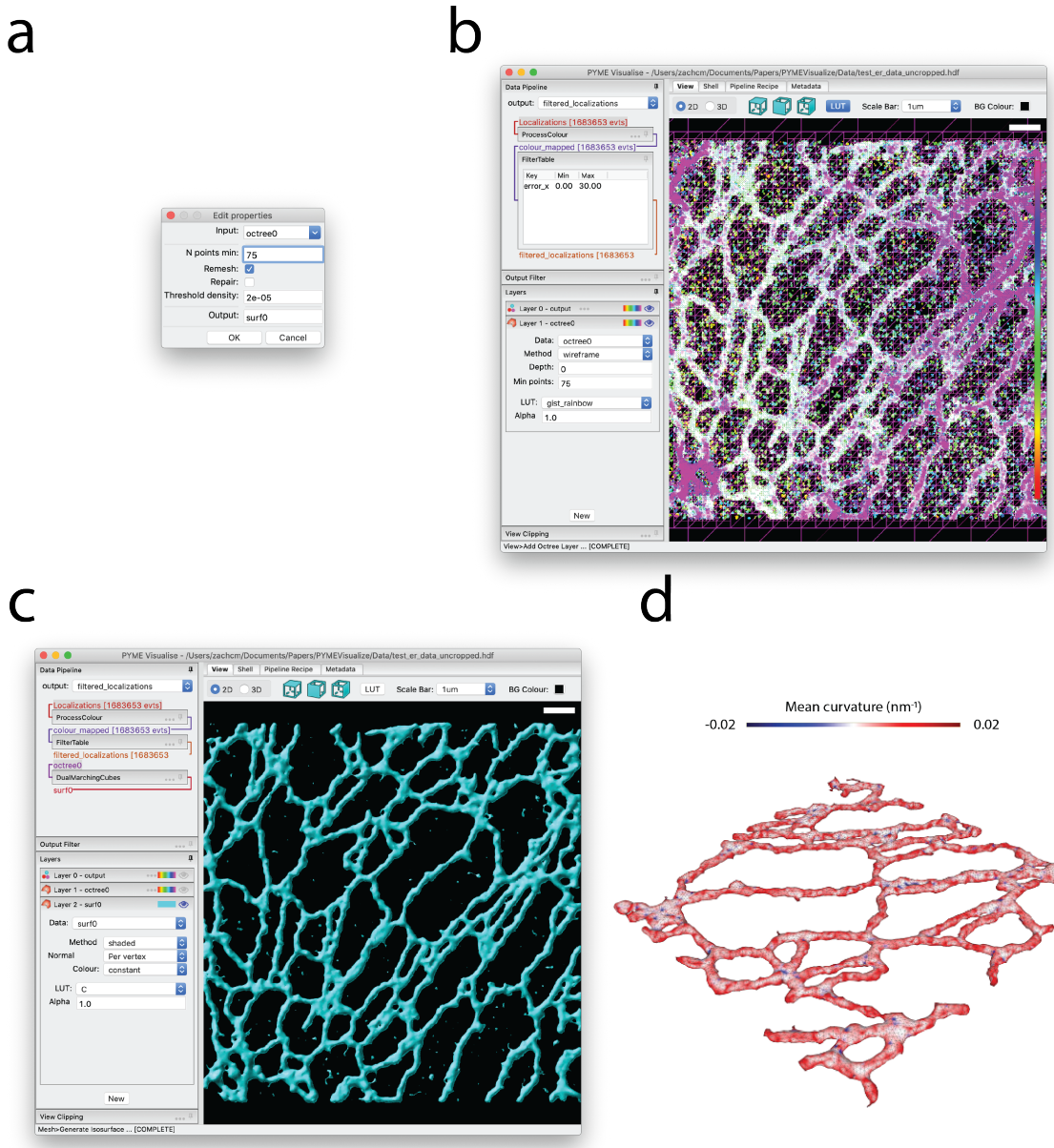


Figure B.8: Isosurface generation. (a) Dialog box for isosurface generation. (b) Birds-eye view of the octree layer used to generate isosurface, overlaid on the original dataset shown in Fig. B.3 a. (c) Birds-eye view of the generated isosurface. (d) Middle portion of the isosurface in c, colored by mean curvature and rotated for perspective.

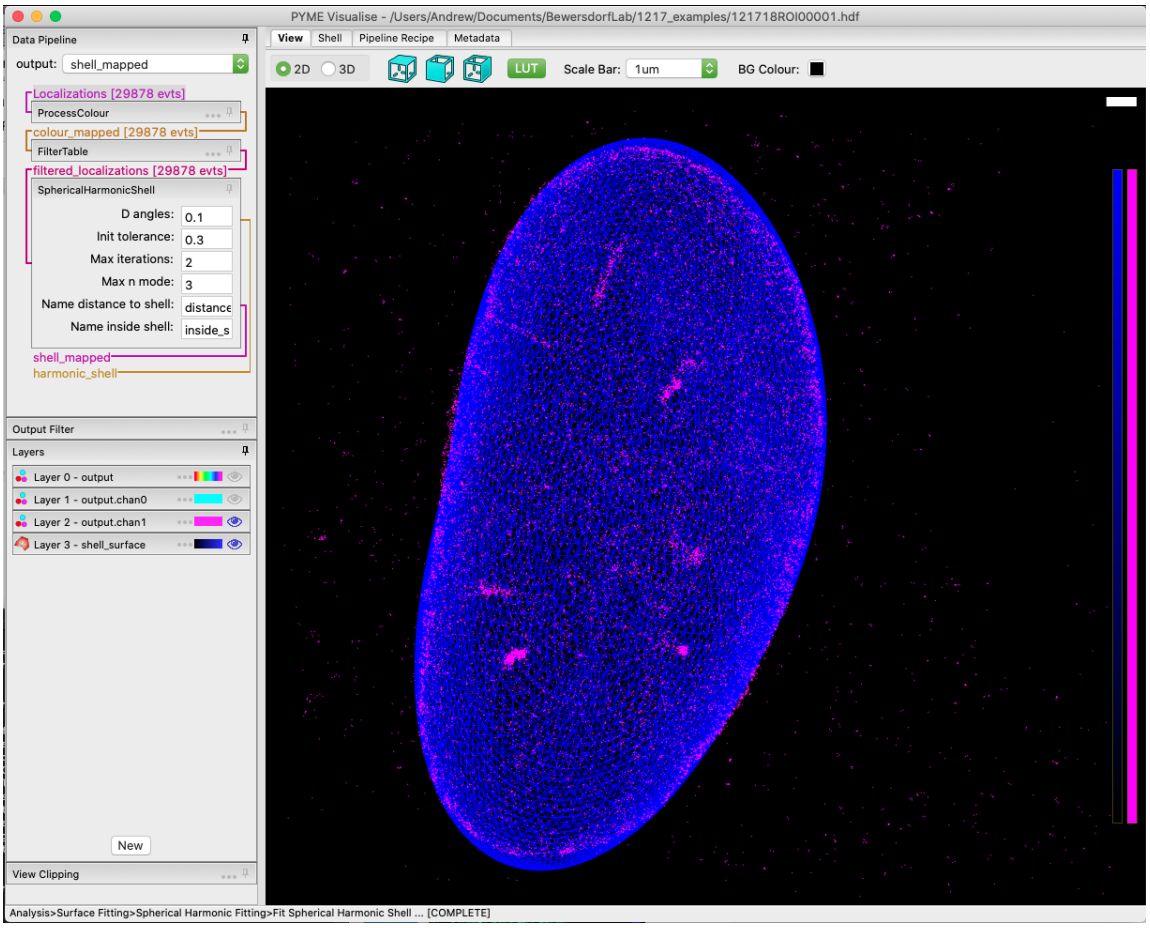


Figure B.9: Data from [37].

Mesh manipulation

A number of operations are possible on meshes generated using either isosurfaces or spherical harmonics. These meshes can be colored by variables, such as `x`, `y`, `z`, and curvature, as in Fig. B.8 d. Meshes can be exported to STL or PLY format, suitable for importing in other software or 3D-printing. There are also a growing number of analysis options (e.g. *Mesh* $\rightarrow$ *Analysis* $\rightarrow$ *Distance to mesh* which calculates the signed distance between localisations and the mesh) which operate on the meshes.

B.2.6 Quantification

PYMEVisualise includes several quantitative analysis routines, operating both directly on localisation data and on reconstructed images. An incomplete description of some of the most well used options is given below.

Pairwise distances

Pairwise distance histograms are a powerful metric calculated directly from the point data set without the need for reconstruction. They measure the distribution of distances from each point to every other point and can be used for cluster analysis, either through the raw pairwise distance histogram (*Analysis* $\rightarrow$ *Clustering* $\rightarrow$ *Pairwise Distance Histogram*) or using derived measures such as Ripleys K & L functions (*Analysis* $\rightarrow$ *Clustering* $\rightarrow$ *Ripley's K/L*) [56, 110]. When applied between two colour channels, they provide a co-localisation (or co-clustering) measure (*Analysis* $\rightarrow$ *Pointwise colocalisation*) [81]. It is also possible to infer some shape features from the pairwise distance histogram [55, 111].

To the best of our knowledge, the algorithms used for calculating pairwise distances histograms in PYMEVisualise are significantly more efficient than existing published options. This is achieved by calculating the histogram on the fly and never storing the full

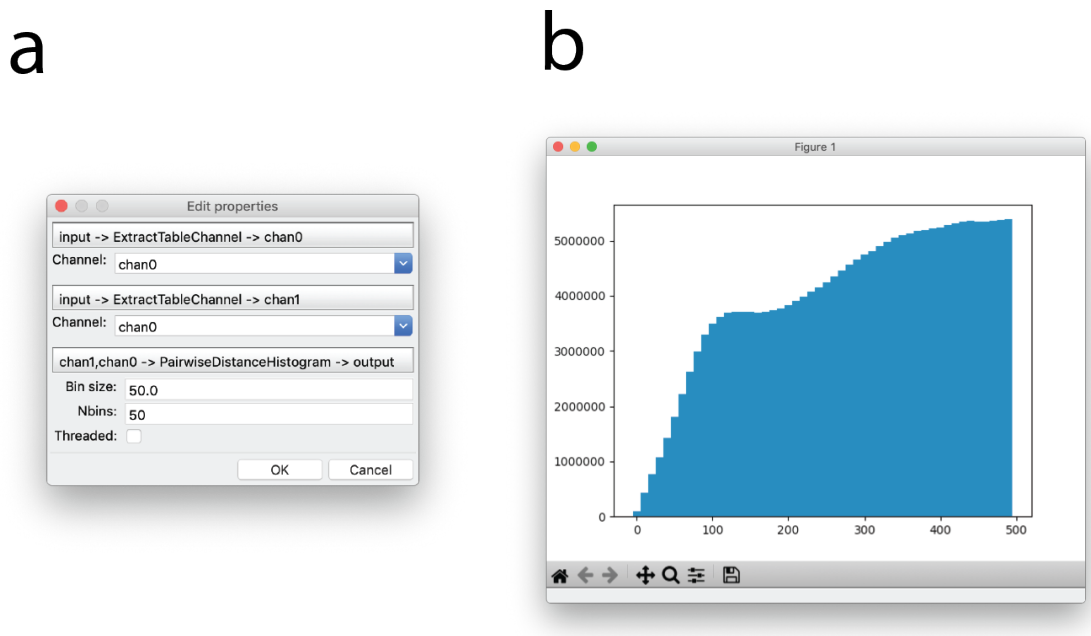


Figure B.10: Pairwise distance histograms. (a) Dialog for generating pairwise distance histograms. (b) Example pairwise distance histogram of sub-ROI of the image in Fig. B.3 a with a bin size of 10 and 50 bins. Distance in nanometers is plotted on the x-axis and counts are plotted on the y-axis.

pairwise distance matrix. To give a rough idea, we are 20x faster than the already optimised `numpy.histogram(scipy.spatial.pdist(...))` (or equivalent MATLAB calls). We also use several orders of magnitude less memory (memory demand scales as $O(N)$ rather than $O(N^2)$ as for `pdist()`) meaning that we can feasibly compute pairwise distance histograms from much larger datasets (100k points requires $\sim$1.6MB as opposed to $\sim$1GB).

For completeness, it is also possible to calculate a histogram of nearest neighbour distances (*Analysis* $\rightarrow$ *Clustering* $\rightarrow$ *Nearest Neighbor Distance Histogram*). Whilst somewhat easier to interpret, these tend to be much more susceptible to noise than the full pairwise distance histogram so it is preferable to use the latter where possible.

Single-molecule tracking

PYMEVisualise offers a couple of particle tracking algorithms. The simplest option, suitable for well separated particles, is accessed as follows. Open a dataset containing at least the variables `x`, `y`, and `t` (see Section B.2.2). Navigate in the menu to *Analysis* → *Tracking* → *Track single molecule trajectories*. A dialog box will pop up as shown in Fig. B.11 a. The maximum distance between two consecutive points which will still be connected as a track is given by `ClumpRadiusScale*ClumpRadiusVariable` where `ClumpRadiusVariable` can be any of the fitted parameters/columns or a constant. For particle tracking it makes sense to leave this at it's default of a constant 1 `nm` and just alter `ClumpRadiusScale`. `ClumpRadiusScale` should be set to a value that is greater than the maximum distance a particle is expected to move within one frame [7], but less than the distance between separate particles. `Min clump size` refers to the minimum number of points that need to be in a track. `Time window` is the maximum distance in time between any two consecutive points in a track. Once tracked, the trajectories will display in a tracks layer (see Section B.2.2) as in Fig. B.11 b. Tracks can be optionally colored and/or filtered by variables from the original dataset, by track unique identifier, by track length, and by instantaneous velocity. A slightly more sophisticated tracking algorithm, suitable for denser tracking scenarios, which can use the z-position along with additional features such as particle brightness and shape to improve linkages is available by manually adding the `TrackFeatures` module to the pipeline (see Section B.2.10).

Other algorithms

- **QPAINT** Algorithms for ploting and fitting off-time distributions for QPAINT ([112])

[7] Or twice the localisation precision if the molecules are super slow moving and their expected motion is less than this.

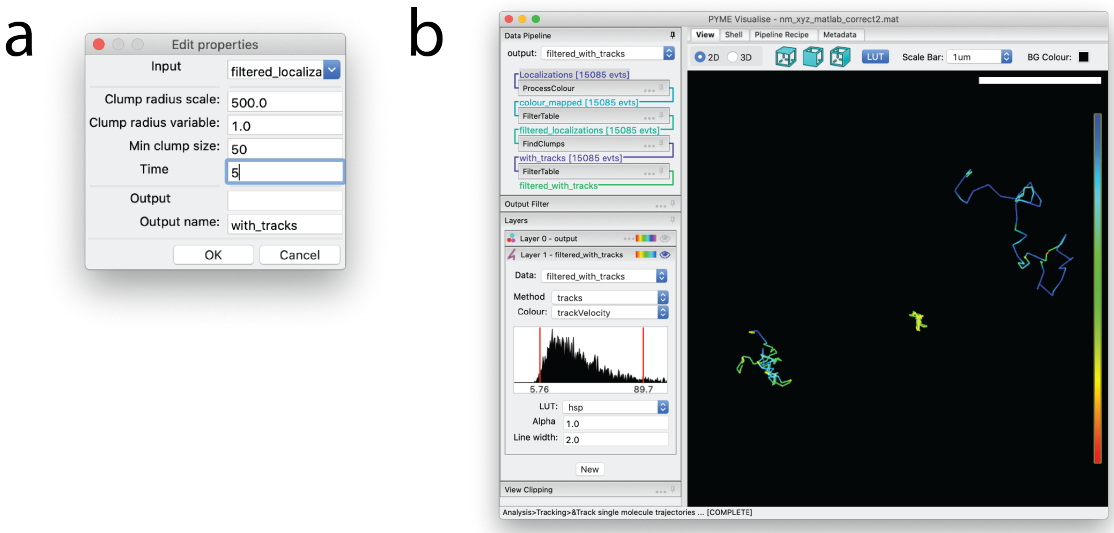


Figure B.11: Tracking Rtn4-SNAP in 2D. (a) Dialog window indicating settings for particle tracking. (b) The resulting trajectories, displayed with constant coloring. The particles giving rise to these trajectories are visible as points in Layer 0, which is hidden in this example, as indicated by the transparent eye.

- **Chromatic shift calibration** A couple of algorithms (*Corrections* → *Shiftmaps* → *XXX*) for calibrating chromatic shifts between colour channels from bead datasets. Mostly used with ratiometric localisation analysis.

- **Vibration characterisation** Accessed as *Extras* → *Diagnostics* → *Plot vibration spectra*, this looks for signatures of instrument vibration in a (rapidly acquired) bead localisation series.

B.2.7 Segmentation and clustering

Although the presence of clusters and average cluster sizes can be deduced directly from point distance distributions (see Section B.2.6), some form of segmentation is needed when quantifying the properties of individual clusters. PYMEVis offers a number of options here.

Voxel-based

PYMEVisualise's Section B.2.4 provides several density image representations that can be thresholded for density-based segmentation. One of the main advantages to this approach is that there is a very large body of existing work in the literature on how to set reasonable thresholds for voxel-based images (e.g. isodata, Otsu) and their limitations are well-characterized. In our experience image based cluster segmentation, when combined with an appropriate density reconstruction algorithm, performs as well as or better than more direct methods such as DBSCAN. It is also by far the simplest way to segment multi-channel data. Once thresholded, the segmentation assignments from the images can be propagated back to the point data.

To threshold a reconstructed image, expand *Display Settings*, select *Threshold mode* and press *Isodata* in the resulting window (Fig. B.12 a). For smoother blobs, it can help to run *Processing* $\rightarrow$ *Gaussian Filter* prior to entering threshold mode.

Once the voxel-based image is thresholded, select *Processing* $\rightarrow$ *Label* and enter the minimum number of pixels a blob must contain to qualify as a cluster in the dialog window that pops up. A new window named *Labeled Image* will appear in which each connected region is assigned a unique integer "label". Return to the main PYMEVisualise window and select *Analysis* $\rightarrow$ *Get segmented IDs from image* and choose the labeled image. Recolour by *objectID* to see the segmented points (Fig. B.12 b). The `labeled` LUT works quite well for displaying clusters.

Tessellation-based

An attractive method of density reconstruction which gives high-quality segmentations is to use a tessellation of the point data [61, 113]. Although one can segment such tessellations directly, our preferred option is to generate a voxel-based density image from

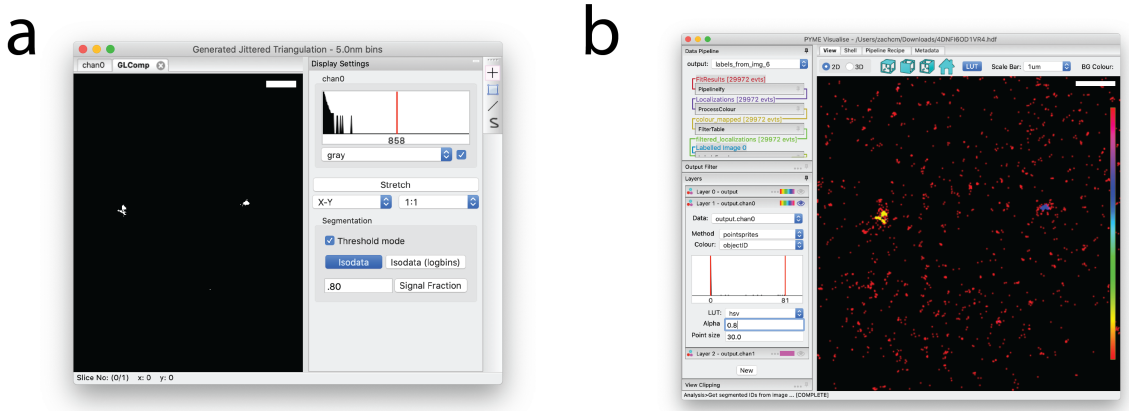


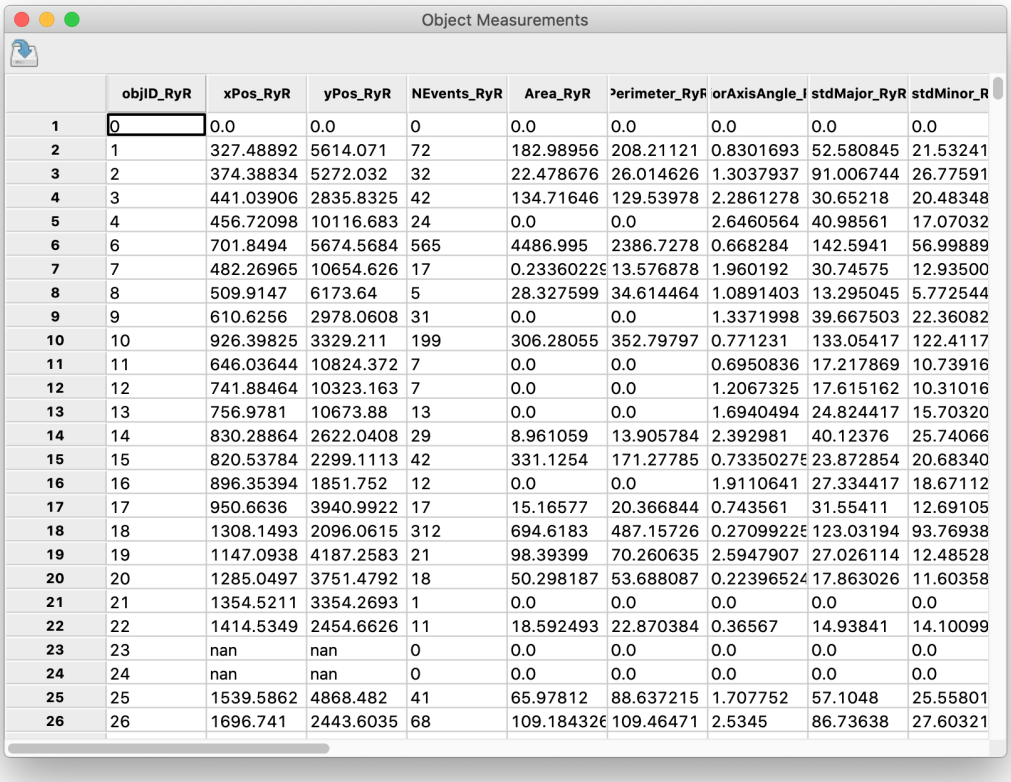
Figure B.12: Voxel-based segmentation in PYMEVisualize. (a) A variant of Fig. B.6 b with *Display Settings* expanded to allow for thresholding. (b) Original point data colored by *objectID*.

the tessellation and then segment this image (see Section B.2.7). This is mathematically equivalent, but a little more flexible than the direct method - particularly when it comes to the choice of thresholding algorithm and application to multi-channel data. In PYMEVis this is performed by running *Generate* → *Jittered Triangulation*, or *Generate* → *Jittered 3D Triangulation*, and then proceeding as described above.

DBSCAN

If a user prefers to cluster directly on points, an implementation of DBSCAN clustering [56, 114] is available by selecting *Analysis* → *Clustering* → *DBSCAN Clump* from the menu. To see the segmentation, recolour by *dbscanClumpID*.

Note DBSCAN relies on single linkages/edge length between points, which makes the algorithm sensitive to noise. It tends to perform less well than the voxel-based segmentation when there is high background.



	objID_RyR	xPos_RyR	yPos_RyR	NEvents_RyR	Area_RyR	Perimeter_RyR	orAxisAngle_I	stdMajor_RyR	stdMinor_R
1	0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0
2	1	327.48892	5614.071	72	182.98956	208.21121	0.8301693	52.580845	21.53241
3	2	374.38834	5272.032	32	22.478676	26.014626	1.3037937	91.006744	26.77591
4	3	441.03906	2835.8325	42	134.71646	129.53978	2.2861278	30.65218	20.48348
5	4	456.72098	10116.683	24	0.0	0.0	2.6460564	40.98561	17.07032
6	6	701.8494	5674.5684	565	4486.995	2386.7278	0.668284	142.5941	56.99889
7	7	482.26965	10654.626	17	0.23360229	13.576878	1.960192	30.74575	12.93500
8	8	509.9147	6173.64	5	28.327599	34.614464	1.0891403	13.295045	5.772544
9	9	610.6256	2978.0608	31	0.0	0.0	1.3371998	39.667503	22.36082
10	10	926.39825	3329.211	199	306.28055	352.79797	0.771231	133.05417	122.4117
11	11	646.03644	10824.372	7	0.0	0.0	0.6950836	17.217869	10.73916
12	12	741.88464	10323.163	7	0.0	0.0	1.2067325	17.615162	10.31016
13	13	756.9781	10673.88	13	0.0	0.0	1.6940494	24.824417	15.70320
14	14	830.28864	2622.0408	29	8.961059	13.905784	2.392981	40.12376	25.74066
15	15	820.53784	2299.1113	42	331.1254	171.27785	0.73350275	23.872854	20.68340
16	16	896.35394	1851.752	12	0.0	0.0	1.9110641	27.334417	18.67112
17	17	950.6636	3940.9922	17	15.16577	20.366844	0.743561	31.55411	12.69105
18	18	1308.1493	2096.0615	312	694.6183	487.15726	0.27099225	123.03194	93.76938
19	19	1147.0938	4187.2583	21	98.39399	70.260635	2.5947907	27.026114	12.48528
20	20	1285.0497	3751.4792	18	50.298187	53.688087	0.22396524	17.863026	11.60358
21	21	1354.5211	3354.2693	1	0.0	0.0	0.0	0.0	0.0
22	22	1414.5349	2454.6626	11	18.592493	22.870384	0.36567	14.93841	14.10099
23	23	nan	nan	0	0.0	0.0	0.0	0.0	0.0
24	24	nan	nan	0	0.0	0.0	0.0	0.0	0.0
25	25	1539.5862	4868.482	41	65.97812	88.637215	1.707752	57.1048	25.55801
26	26	1696.741	2443.6035	68	109.184326	109.46471	2.5345	86.73638	27.60321

Figure B.13: Tabular viewing window.

Exporting object measurements

It is possible to export object measurements to HDF or CSV file, which can then be further analyzed in a Jupyter notebook, Excel, etc. To do this, navigate to *Analysis* → *Measure objects*. A tabular viewing window will appear, as shown in Fig. B.13. Click on the disk drive icon in the upper left of the window to save the results to file.

B.2.8 Animation

Animations, such as point cloud fly-throughs, are generated from keyframes set by the user in the animation panel. The animation panel is accessed by selecting *Extras* → *Animation* from the PYMEVisualize menus.

A populated *Animation* pane is shown in Fig. B.14. Each row in the animation table represents a keyframe. A keyframe marks the end of a transition in the animation. A keyframe is set by rotating, translating, or zooming the data to have a desired look and then pressing *Add* in the *Animation* pane. Double-clicking on a row allows editing of the keyframe name, which can be used as a unique descriptor, and duration, which edits the length of the transition from the previous keyframe to this keyframe (note that the duration of the first keyframe is therefore a dummy variable). Selecting a row and pressing *Delete* removes the selected keyframe from the animation. Keyframes can be saved and loaded in a JSON format using *Save* and *Load*, respectively. All keyframes can be removed at once by pressing *Clear*.

When the *play* button is pressed, the animation will play in as a seamless transition through the keyframes in order. If *Capture* is pressed, a dialog will pop up and the user will choose a folder in which to save this animation as a series of image files. Expanding *Settings...* reveals a dropdown menu labeled *Export file type*, which allows the user to change the file type of the images exported to JPEG, PNG, or TIFF.

B.2.9 Synthetic data

For testing purposes, it can be helpful to generate synthetic data. PYMEVisualize lets users generate synthetic data using a simulated worm-like chain model (with the right settings this can produce reasonable analogues of a wide range of filamenteous structures from microtubules to tightly folded DNA), from an image, or from a text file containing a list of coordinates. First select *Extras* $\rightarrow$ *Synthetic Data* $\rightarrow$ *Configure* and choose the point generating source and set properties associated with the source, as shown in Fig. B.15 a. Then select *Extras* $\rightarrow$ *Synthetic Data* $\rightarrow$ *Generate fluorophore positions and events*, and a simulated set of localization events is generated, as shown in Fig. B.15 b. This data can then be analyzed in the same way that real data would be.

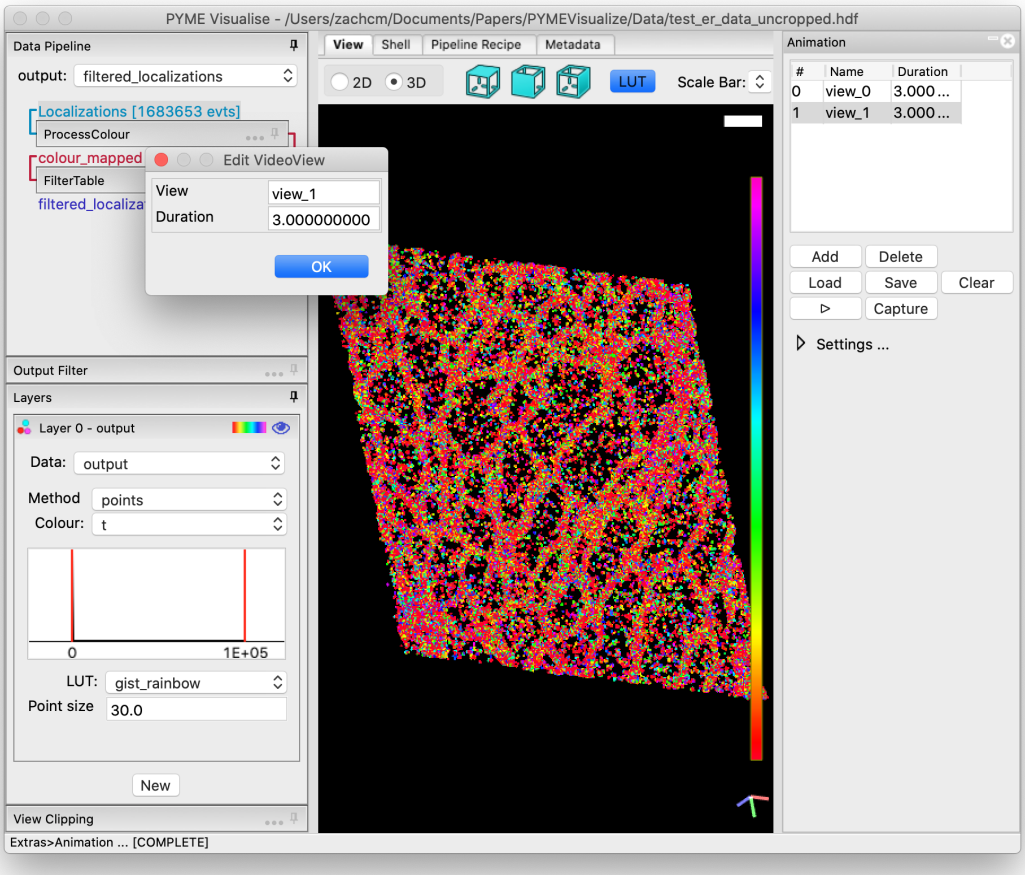


Figure B.14: Actively editing an animation keyframe in PYMEVisualize. This shows an animation with two keyframes added by rotating the data and pressing *Add* at each desired rotation. The second keyframe has been double-clicked, revealing an *Edit VideoView* window, which allows the user to change the name of the keyframe and the duration of the transition from the previous keyframe to this keyframe.

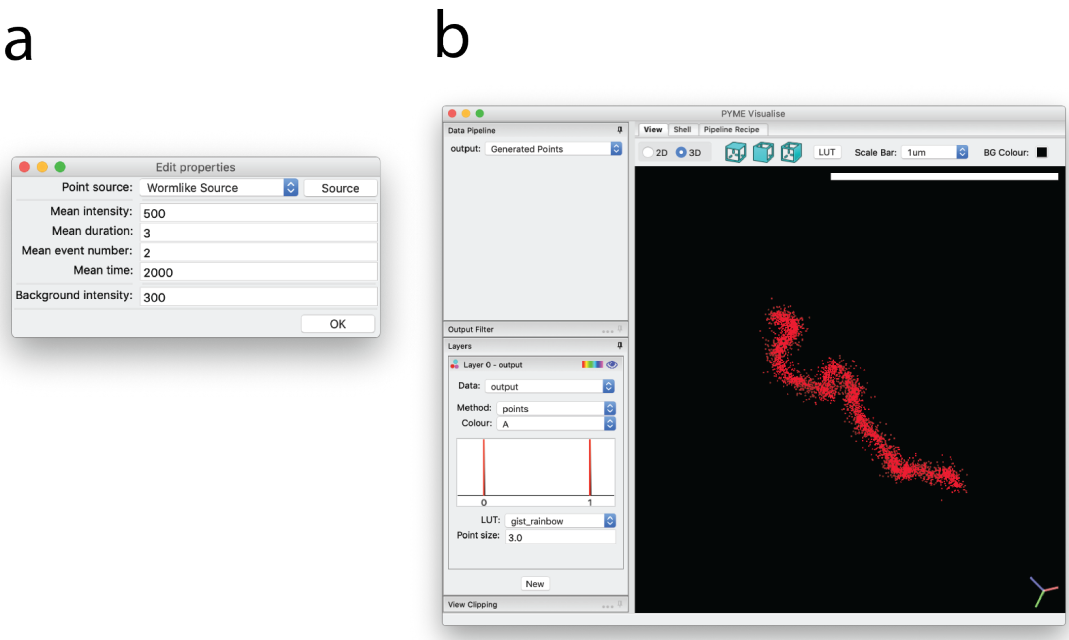


Figure B.15: Generation of synthetic data. (a) Dialog box that appears after selecting *Extras* → *Synthetic Data* → *Configure*. (b) Example synthetic worm-like chain created using parameters from dialog box in a.

To simulate additional localizations from the same set of points, but using different simulation parameters, edit the properties as in Fig. B.15 a and then select *Extras* → *Synthetic Data* → *Generate events*.

B.2.10 Editing the pipeline "recipe"

As mentioned in Section B.2.2, data flows through a configurable pipeline. When constructing a complex workflow (e.g. processing different colour channels in different ways - see Section B.2.13) it can be useful to edit this pipeline directly using the recipe editor (Fig. B.16), accessible via the *Pipeline Recipe* tab. This also lets you access additional functionality such as feature based tracking, which is not currently accessible through the menus. The pipeline is specified by a recipe (a .json textual description) which describes the various processing steps to be applied. The recipe can be modified, either graphically

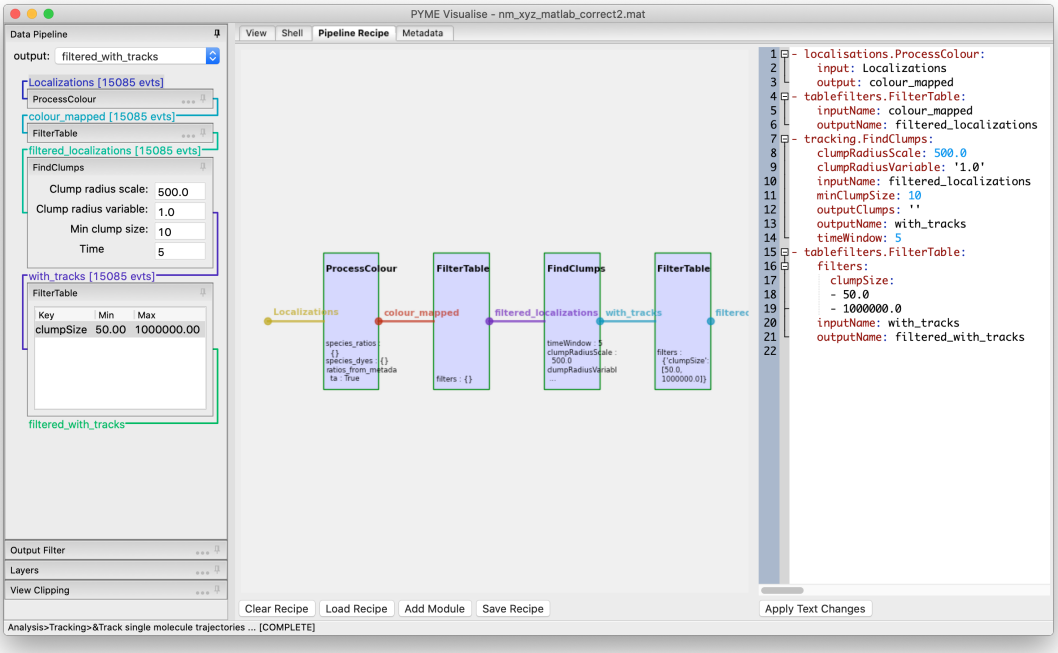


Figure B.16: The pipeline as seen in the recipe editor. Custom workflows can be created by manually adding processing modules.

or as text, saved and reloaded on a new dataset, or applied to multiple input files in a batch using the `bakeshop` utility.

B.2.11 Programmatic usage

Shell

The *Shell* tab is a functional Python command line embedded within the program. The pipeline can be accessed directly from the shell, and behaves like a dictionary keyed by variable names. Pylab is imported in the shell making a number of MATLAB-style plotting and basic numeric commands accessible (see the matplotlib webpage for more docs). One can, for example, plot a histogram of point amplitudes by executing `hist(pipeline['A'])`. Pipeline data sources can be accessed by entering `pipeline.dataSources[datasource_key]`. For a list of datasource keys, type `pipeline.dataSources`.

```
In [ ]: from PYME.LMVis import VisGUI

        %gui wx
```

```
In [ ]: pymevis = VisGUI.ipython_pymevisualize()
        pipeline = pymevis.pipeline
```

```
In [ ]: import numpy as np
        from PYME.IO import tabular

        points = np.random.randn(100,3)*100
        pipeline.addDataSource('points', tabular.mappingFilter({'x': points[:,0],
                                                                 'y': points[:,1],
                                                                 'z': points[:,2]}))
        pipeline.selectDataSource('points')
        pymevis.add_pointcloud_layer(ds_name='points')
```

Figure B.17: An example of generating a point cloud, passing it to a tabular data source, and visualizing the data source in PYMEVisualize from a Jupyter notebook.

Jupyter notebook

PYMEVisualize can be used directly from a Jupyter notebook. At the top of the notebook, enter `from PYME.LMVis import VisGUI`, `%gui wx`, and then `pymevis = VisGUI.ipython_pymevisualize()`. This makes it possible to access a PYMEVisualize instance from a notebook through the `pymevis` variable. Setting `pipeline= pymevis.pipeline` gives the user access to the PYMEVisualize pipeline in exactly the same way as described in Section B.2.11 section. An example of generating a point cloud, passing it to a tabular data source, and visualizing the data source is shown in Fig. B.17.

Please note that both PYME and the Jupyter kernel must be set up in the Framework build on a Mac (see `https://python-microscopy.org/doc/Installation/InstallationFromSource.html`) for this to work. To install the Jupyter kernel in the Framework build, activate `your_pyme_environment` and then type `PATH/TO/CONDA/ENVIRONMENT/python.app/Contents/MacOS/python -m ipykernel install --user --name your_pyme_environment`.

Plugins

Details on extending and writing plugins for PYMEVisualize are available at `http://python-microscopy.org/doc/hacking.html`. A template for extending PYMEVisualize can be found at `https://github.com/python-microscopy/pyme-plugin`.

B.2.12 Further reading

Documentation is kept up-to-date at http://python-microscopy.org/doc/. In addition to PYMEVisualise documentation, this includes information about other core components of PYME, which Windows users will see on their desktop: PYMEImage for voxel-based image visualisation and analysis, PYMEAcquire for microscopy hardware control, PYMEClusterOfOne for localization fitting, and Bakeshop for developing reusable analysis pipelines. Mac and Linux users can access these applications from the command line.

B.2.13 Appendix

Ratiometric colour settings

When processing ratiometric localisation data (having a `gFrac` column) the splitting ratios for each dye species can either be set in the series metadata, or by using the *Colour* tab (Fig. B.18). To add a labelling right click in the *Fluorophores* list box and select *Add*. Enter a channel name and splitting ratio in the dialog that opens. Alternatively click *Guess* to attempt to automatically detect the channels using the K-means algorithm. Once added, you can click in the name or ratio columns to edit. The plot above is a scatter plot showing a subset of all localisations and updates to show the resulting channel assignments.

Thresholds used in the Bayesian assignment process are adjustable in the *Channel Assignment* panel. A dye is assigned to a given channel if both it's probability of belonging

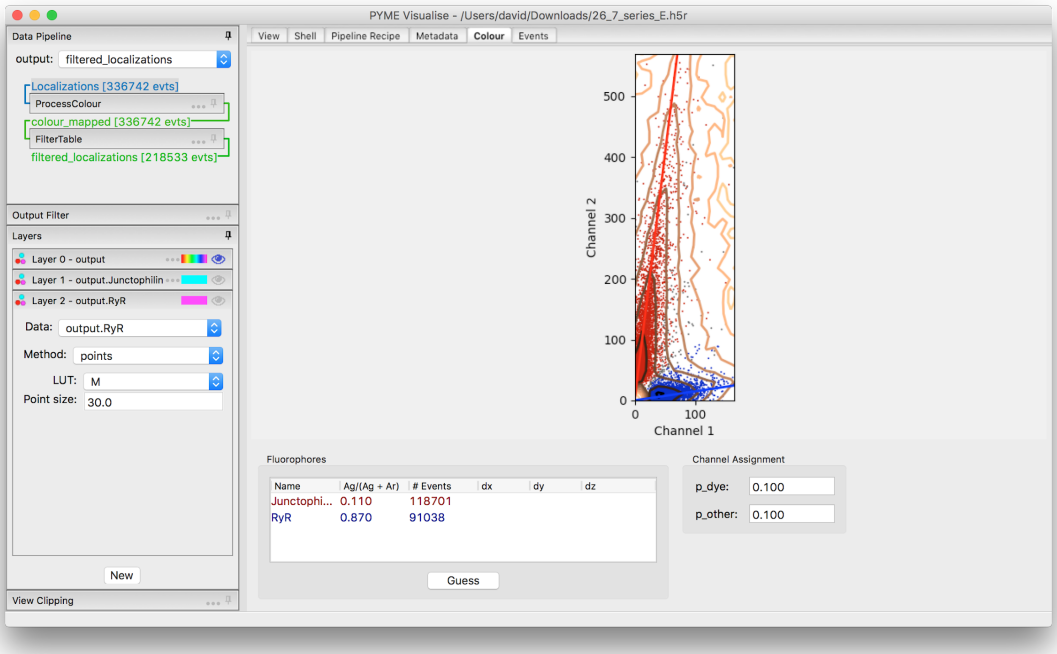
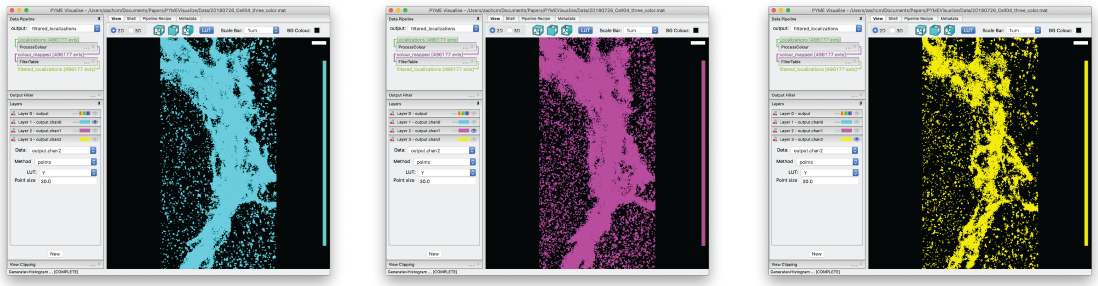


Figure B.18: Ratiometric splitting in the *Colour* tab.

to that channel is greater than `p_dye` **and** it's probability of belonging to any other channel is less than `p_other`. The defaults assign fluorophores to the most likely channel and ensure that the chance of a false assignment is less than 10%. We find they seldom need tweaking. If adjustment is necessary, `p_other`, which controls the rejection of potentially mis-assigned localisations, is most useful. It is tempting to think that `p_dye` should be higher (i.e. we should have a high certainty that a dye belongs to a given channel), but this would be a mistake - `p_dye = 0.1` will include 90% of the statistical spread of localisations belonging to that channel. `p_dye = 0.5` by comparison would only capture 50% of a dye's statistical spread and would needlessly discard a large fraction of the localisations.

a



b

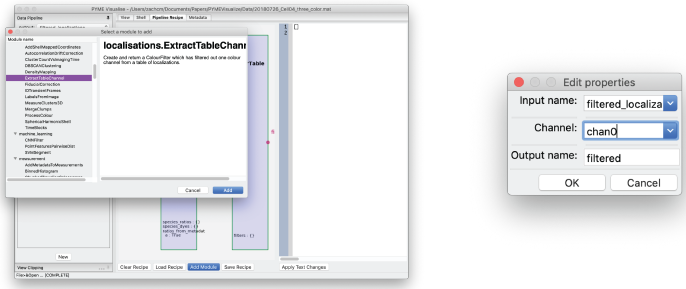


Figure B.19: Visualization and color channel selection of 3-color super-resolution image of *cis*, *medial*, and *trans*-Golgi. (a) *Top*. All three color channels visualized in a single layer. (b). Selection of the `ExtractTableChannel` recipe in the *Pipeline Recipe* tab and `ExtractTableChannel` dialog box, set to extract color channel `chan0` from the original data.

Isolating a single channel for processing

To apply processing steps to a single channel (rather than to all channels at once), it needs to be isolated in the pipeline. To do this, navigate to the *Pipeline Recipe* tab and select *Add Module*, as in Fig. B.19 b. Then select the `ExtractTableChannel` recipe from the **localisations** recipes and press *Add*. This will result in a dialog box as shown in Fig. B.19 c, where here the first color channel, `chan0`, is selected. Returning to the *View* tab and selecting `filtered` as the *output* in the upper-left portion of the window shows only the localizations present in the color channel `chan0` (Fig. B.19 a, bottom). Additional data processing will only operate on this color channel as long as `filtered` is selected as the *output*.

B.3 Comparison to other localization microscopy visualization packages

When compared with other packages for PALM/STORM visualization, we believe PYMEVis has many features that make it an attractive choice—both for researchers new to super-resolution field and to those who are running into the limits of existing analysis tools. These are:

- A "batteries-included" philosophy with the most comprehensive set of algorithms currently available in one package for localization point-cloud analysis.
- A plugin architecture which lets users leverage the vast number of high-quality computational packages already available within the scientific Python ecosystem, and to write new plugins in a fraction of the time that would be required in other languages.
- A GPU accelerated 3D viewer supporting multiple different data types (points, surfaces, images).
- A licensing model that does not require contributors and plugin authors to have expensive proprietary software (e.g. MATLAB).
- Algorithms designed to work efficiently with millions of points.
- Active, ongoing development.

This comparison is summarized in Table B.1.

Detailed feature comparison matrix

Table B.2 below gives a more detailed comparison of the individual algorithms available in each package, used to derive our 'quantification' score in the summary table. This is

	Programming Language	Open source	Plugin architecture	Last commit	Commits in the last year*	File support	Performance	Quantification
PYMEVis	Python/C	🟢	✔	2/02/21	2150 (510)	H T M S	G C L	★★★
VisP	C++	🔴		8/06/15	0	T	G	★
PALMsiever	MATLAB	🟡		9/03/17	0	T M	L	★★
SR-Tesseler	C++	🟢		10/02/20	2	T	G	★
SharpViSu	MATLAB	🟡		11/10/18	0	T M	L	★★
LAMA	Python	🟢		24/08/18	0	T	L	★★
3DClusterViSu	MATLAB/Python	🟡		14/01/21	2	T M	L	★
Grafeo	MATLAB	🟡		11/12/20	15	T M	L	★★
NanoJ	Java	🟢		10/09/20	9			★★
SMoLR	R	🟢		7/05/19	0	T	L	★★
SMAP	MATLAB	🟡	✔	13/01/21	333	H T M	L	★★★
Napari	Python	🟢	✔	2/02/21	603	T S	G C L	
Genuage	C#	🟢		14/12/20	32	T	G	★

Legend

🟢 Open source

🟡 Code openly available, but uses proprietary software/libraries for development

🔴 Closed source

H	HDF5	**G**	GPU accelerated visualization
T	Delimiter-separated text	**C**	Specific optimizations for CPU based parallisation
M	MATLAB	**L**	Uses 3rd party libraries with multithreading optimizations
S	STL/PLY		

Table B.1: Comparison of key features of available localization microscopy visualization packges. (*) For all repositories except PYMEVis, we report the total number of commits. As PYMEVis is only one component of the PYME repository, we also made a conservative estimate of the commits directly effecting PYMEVis (value in parentheses).

a best-estimate based on reading the primary publications, manuals, and feature lists (if available) for all listed packages. We did not go as far as a line-by-line examination of the code, however, and recognize that there may be omissions. We apologize for these. With actively developed packages, feature lists are also dynamic and will change over time.

Table B.2: Detailed comparison of available algorithms in single-molecule localization microscopy software packages.

Appendix C

Appendix for Chapter 5

This is supplementary material for the paper draft in Chapter 5.

C.0.1 Supplementary figures

C.0.2 Supplementary Note

This note describes additional details of the algorithm.

Hole/cut generation

Sometimes the density-based isosurface will connect regions of the localization point cloud that should be separate. For example, two tubes that run along side one another may form bridges in the density-based isosurface if they get closer than the minimum box size of the octree. To ensure recovery of true membrane structure, it is necessary to test for these regions and perforate or cut the mesh as necessary. The procedure is outlined in Algorithm 1.

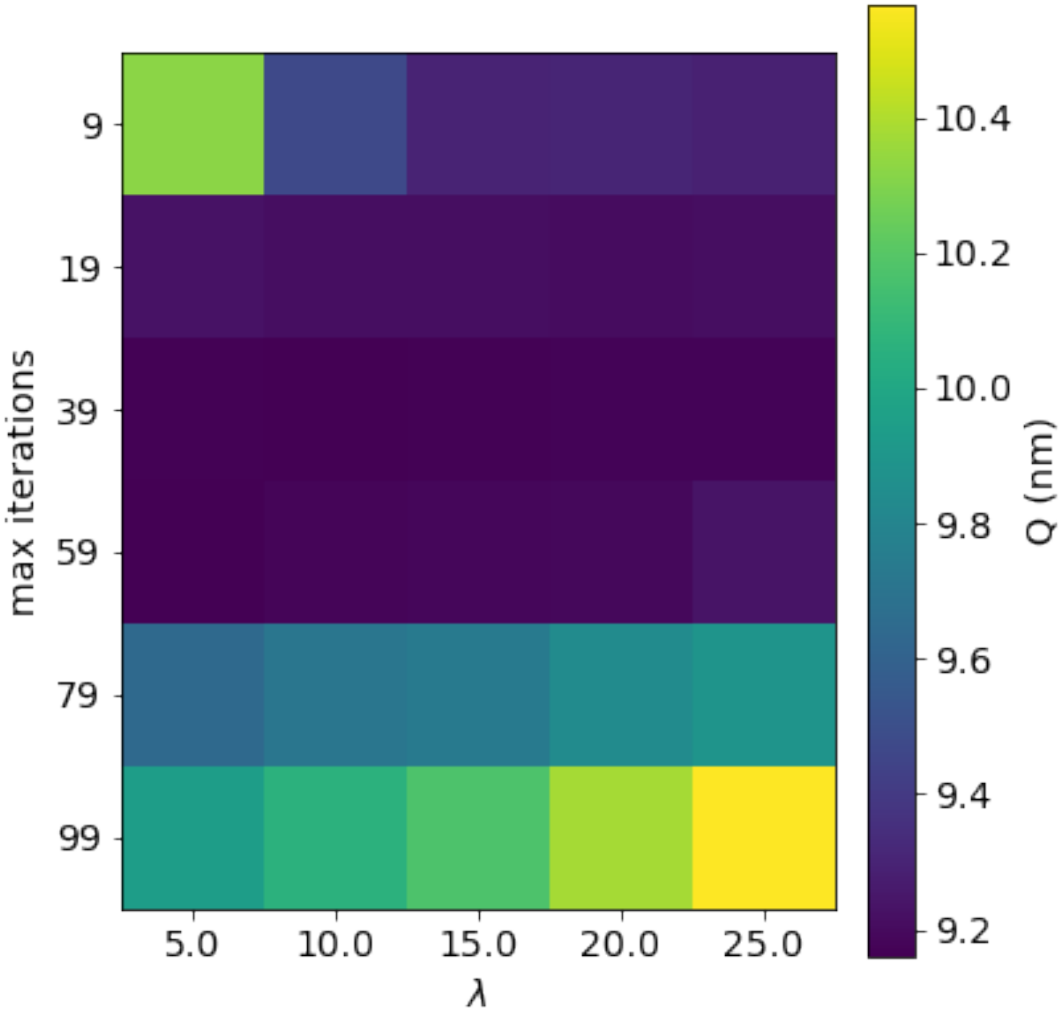


Figure C.1: Heatmap showing Q (RMS) values for meshes generated from search parameters for method described in this paper.

Point cloud simulation

In order to demonstrate the effectiveness of the algorithm, it was necessary to generate biologically-motivated test structures. To do this, we established constructive solid geometry library, where each primitive is represented as a signed distance function (SDF). For example, three-way junctions can be represented as a union of capsule SDFs.

Points were generated from a test structure using Monte-Carlo sampling on an octree.

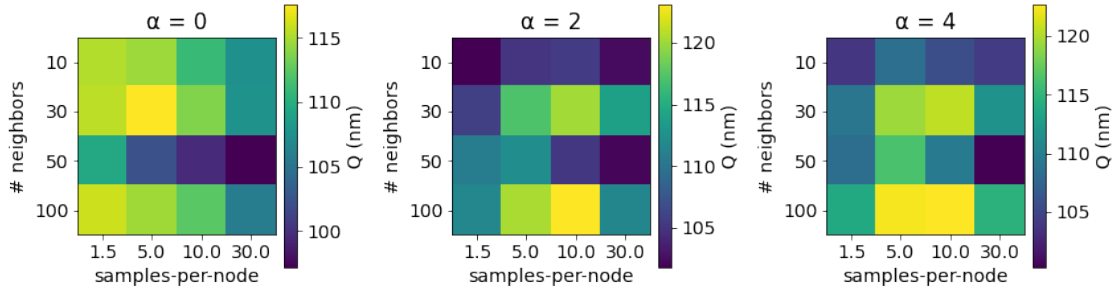


Figure C.2: Heatmap showing Q (RMS) values for meshes generated from search parameters for SPR. Note that the minimum at samples-per-node $= 30$ indicates a need to expand our grid search along this dimension at higher noise levels.

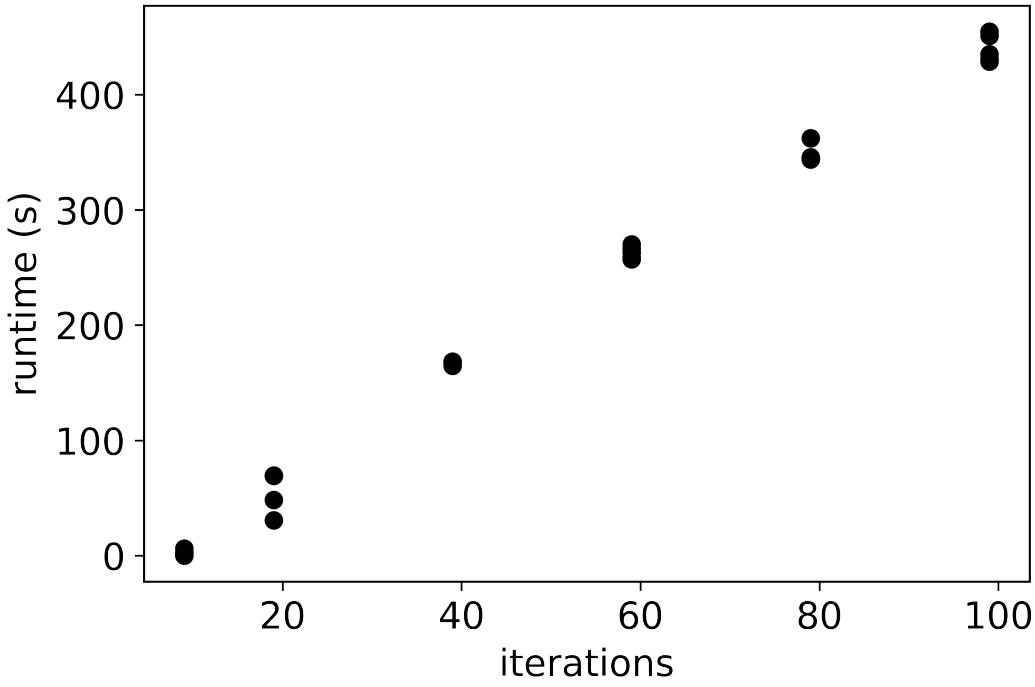


Figure C.3: Plot of runtime vs. number of iterations for meshes generated using the method described in this paper.

The octree subdivides space that straddles where the SDF of the test object goes to zero until a fixed sampling rate dx is reached. The center of each octree box with an absolute SDF value of less than dx is kept with a probability of p and rejected otherwise. This

Algorithm 1 Hole/cut generation in a manifold mesh

1. Find all faces that have no points within a distance ε.
2. Within this set, find opposing faces that are reachable by traveling along their negative normal.
3. For each pair, keep these faces if there are no points inside the prism formed by the opposing faces, or within a distance ε outside of this prism.
4. Compute the connected component labeling of the kept faces such that faces that share an edge are considered connected.
5. For each connected component...
   (a) If it has an Euler characteristic of 0, it is topologically a cylinder. Cut it with a plane.
   (b) If it has an Euler characteristic of 1, it is topologically a plane. If a face in this component is paired with a face in another component that has an Euler characteristic of 1, punch a hole in the mesh using this face pair. Remove both connected components from consideration for further hole/cut generation.

results in a point set of size S.

Next, non-specific localizations are simulated. For a given noise fraction n, $J = \frac{nS}{1-n}$ additional points are generated random uniform over the bounding box of the point set. This ensures $J/T = n$ for $T = J + S$ total points.

To simulate localization precision, each point i in the resulting point set is jittered according to a Gaussian noise model with μ_i equivalent to the location of point $i \in 0\,..\,T-1$, and uncertainty $\boldsymbol{\sigma}_{i,e}$ for $e \in x, y, z$. Up to N samples within the Gaussian are generated per point. The total point set is then clipped to size T via random uniform selection of points.

The uncertainty $\boldsymbol{\sigma}_{i,e}$ is calculated by simulating an exponential distribution with a mean value of ρ, corresponding to the average photon count of a theoretical dye that generated these points. This distribution is clipped to contain values greater than a background photon count b, simulating the noise floor for data collection. For the remaining counts,

the first S are selected and

$$\sigma_{i,e} = \frac{r_e}{2.355\sqrt{\rho_i}}$$

where r_e is the resolution of the simulated system along axis e.

Test structures

To simulate a range of biological features, a three-way junction, a collection of tubes and sheets, and two abutting toruses were simulated, as shown in Figure C.4. This spans the range of completely straight portions of a subcellular membrane network to holes perforating a large flat sheet. The 3D figure eight was selected as the test structure for grid-search comparison.

Figure C.4: Test structures used for simulation of theoretical membranes. From left to right, a three-way junction, a structure representing a small piece of the endoplasmic reticulum, and two abutting toruses. The scale bar in the upper right is $100\ \mathrm{nm}$ and is projected into the yz plane at the same angle as the structures.

Appendix D

Appendix: Mesh library

Rather than using CGAL [1], OpenMesh [115] or another existing graphics library, I opted to build a library for surface representation, extraction, and manipulation from scratch. The main reason for this was to have good interoperability with NumPy [116]. If this is not needed, I highly recommend using CGAL. This section includes many implementation details and some explanations of design choices in Python Microscopy's meshing library.

D.1 Structure

D.1.1 Storage and representation

The absolute easiest way to represent a mesh is with a set of vertices and a set of faces, which index these vertices. An example for a triangular mesh is shown in Table D.1.

This representation is cumbersome if we want to access a vertex's neighbors on a mesh. To find the *one-ring neighbors* of a vertex, which are the vertices accessible from this vertex via a single edge (i.e. the vertices of all the triangles incident on this vertex), we have to find all faces containing this vertex, grab the indices associated with these faces,

[1] `https://www.cgal.org/`

	Vertices			Faces		
0	110.58681	242.42331	204.87704	3553	3596	1389
1	116.72764	228.55923	154.79031	1100	2129	1079
2	189.80975	235.7897	291.6563	1328	780	1478
3	298.10123	206.28201	182.40652	2134	1939	3707
4	120.17539	254.84853	178.06435	861	1679	2089
		...			...	

Table D.1: Vertices and faces of a triangular mesh. The faces index the rows of vertices. A quadrilateral mesh would have four entries per face.

and remove any duplicates.

To make things easier on ourselves, we can store connectivity info as shown in Fig. D.1. Here we can move along edges to find all of the vertex neighbors. There are a number of ways to store connectivity, but a preferred method, which minimizes storage space while allowing constant time traversal, is the halfedge data structure used by OpenMesh[2] [115].

In a *halfedge* data structure, each vertex stores its position and an outgoing halfedge that points to some neighboring vertex. Each halfedge stores a pointer to the face it belongs to, a pointer to the next (and optionally to the previous) halfedge in its face, a pointer to its twin halfedge (if a halfedge points from $\vec{\mathbf{v}}_a$ to $\vec{\mathbf{v}}_b$, its twin points from $\vec{\mathbf{v}}_b$ to $\vec{\mathbf{v}}_a$), and a pointer to the terminating vertex of the halfedge. Each face stores a pointer to some halfedge in its face, and is made up of three halfedges (this halfedge, its next, and its previous halfedge).

Looking at Fig. D.1, we see that by using this data structure we can reach every vertex neighboring $\vec{\mathbf{v}}$ in linear time by starting with vertex pointed to by $\vec{\mathbf{v}}$'s halfedge and moving to the vertex that is pointed to by the twin-next halfedge (next halfedge after the twin halfedge of $\vec{\mathbf{v}}$'s halfedge) and so on.

[2] `https://www.graphics.rwth-aachen.de/software/openmesh/`

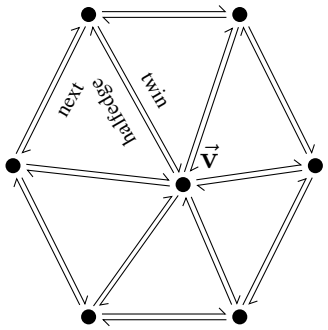


Figure D.1: One-ring neighborhood of a vertex $\vec{\mathbf{v}}$, shown with halfedges. The one-ring neighborhood contains all vertices that $\vec{\mathbf{v}}$ can connect to with a single halfedge. **halfedge** is the arbitrarily-assigned halfedge emanating from $\vec{\mathbf{v}}$. **twin** is the twin halfedge of halfedge. **next** is the next halfedge of halfedge. **prev** halfedge is not shown, but would be equivalent to the next halfedge of **next**.

D.1.2 Winding and normals

Winding is the order in which a face's vertices (and halfedges) are stored, and can be clockwise or counterclockwise, as shown in Fig. D.2. To ensure our mesh has consistent normal vectors, which is important for shading and many other calculations, each face must have the same winding.

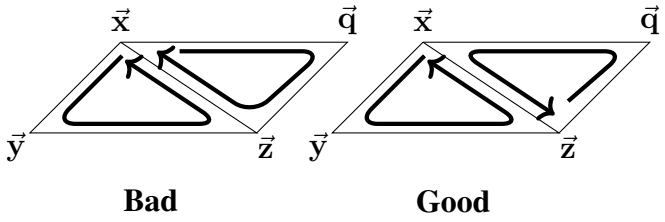


Figure D.2: Non-deal (left) and ideal (right) winding of contiguous faces. Since winding determines the normal, it is important that all faces are wound in the same direction (right) for accurate calculations.

A *normal* vector points out of and perpendicular to its face. Assuming the winding is correct, normals will conventionally point from inside to outside a meshed object. In the case of a triangular face described by three points in space $\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}$ or by the vectors $\vec{\mathbf{v}}_1 = \vec{\mathbf{x}} - \vec{\mathbf{y}}$ and $\vec{\mathbf{v}}_2 = \vec{\mathbf{z}} - \vec{\mathbf{y}}$, the normal is calculated as $\vec{\mathbf{n}} = \frac{\vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2}{||\vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2||}$.

Vertex normals can be approximated by weighting the contributions of the normals of all adjacent faces. A common weight is the face area divided by the area of all the adjacent faces.

D.1.3 Tangent planes and vectors

While normals are perpendicular to a mesh face, tangents are parallel. That is, they are in the plane defined by the mesh face.

Suppose we have a triangular face described by three points in space $\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}$ and let $\vec{\mathbf{v}} = \vec{\mathbf{y}} - \vec{\mathbf{x}}$ and $\vec{\mathbf{n}}$ be the face normal. A tangent vector can be constructed using the projection matrix $\mathbf{P} = \mathbf{I} - \vec{\mathbf{n}}\vec{\mathbf{n}}^T$ as $\vec{\mathbf{T}}_1 = \mathbf{P}\vec{\mathbf{v}} = (\mathbf{I} - \vec{\mathbf{n}}\vec{\mathbf{n}}^T)\vec{\mathbf{v}} = \mathbf{I}\vec{\mathbf{v}} - \vec{\mathbf{n}}\vec{\mathbf{n}}^T\vec{\mathbf{v}} = \vec{\mathbf{v}} - \vec{\mathbf{n}}(\vec{\mathbf{n}} \cdot \vec{\mathbf{v}})$. Thus, the tangent vector is portion of $\vec{\mathbf{v}}$ not traveling in the direction of $\vec{\mathbf{n}}$.

An orthogonal tangent vector can be constructed as $\vec{\mathbf{T}}_2 = -\vec{\mathbf{T}}_1 \times \vec{\mathbf{n}}$. $\vec{\mathbf{T}}_1$ and $\vec{\mathbf{T}}_2$ are tangent vectors emanating from the base of the normal $\vec{\mathbf{n}}$ and are tangents of the face.

We can construct tangent vectors for a vertex of this triangle in a similar manner. Suppose we want tangent vectors at the vertex $\vec{\mathbf{y}}$. Let $\vec{\mathbf{n}}_y$ be the vertex normal. A set of orthogonal tangent vectors are $\vec{\mathbf{T}}_1 = -\vec{\mathbf{v}} - \vec{\mathbf{n}}_y(\vec{\mathbf{n}}_y \cdot (-\vec{\mathbf{v}}))$ and $\vec{\mathbf{T}}_2 = -\vec{\mathbf{T}}_1 \times \vec{\mathbf{n}}_y$. If this vertex is adjacent to multiple triangles, it is clear that each triangle will produce a slightly different tangent vector at vertex $\vec{\mathbf{y}}$. All of these tangent vectors exist in the vertex's tangent plane. The weighted sum of the tangent vectors from each triangle can produce a single pair of tangent vectors for this vertex. The tangent vectors we are interested in varies by application, therefore the weighting varies as well.

D.1.4 Quality

Quality is a measure of how rapidly and accurately we can perform calculations on a mesh. Increased quality usually comes with a need for more computational power. If our faces are smaller, sampling our object more finely, the mesh quality increases but we now have to perform operations on many more edges and vertices.

There are some simple heuristics that improve the quality of a mesh. *Valence* is the number of edges emanating from a vertex (equivalently, the number of vertex neighbors).

In an ideal triangular mesh, all vertices have a valence of 6. Vertices on the *boundary*, a vertex connected to a halfedge that has no twin, of a triangular mesh should have a valence of 4. We often remodel meshes by flipping connecting edges in a way that minimizes valence. The edge flipping operation is shown in D.3.

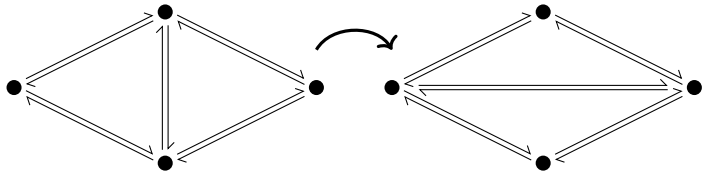


Figure D.3: Edge flip operation, with halfedges shown.

It is helpful to keep mesh edge lengths consistent. If edge lengths vary across a mesh, make sure they vary smoothly.

Skewness, which can be roughly thought of as the difference between the minimum and maximum angle of a mesh polygon, is another good indicator of mesh quality. It is helpful to minimize skewness.

We often need to remodel meshes to improve quality. This is usually done with a combination of the techniques described below that we collectively call remeshing [92].

Subdivision

We have to subdivide a mesh if we want to more finely sample an object. There is a temptation to do this by splitting faces, placing new vertices at face centers. This quickly leads to high-valence vertices. If you are unconvinced, sketch out a couple of iterations. A safer way to split a mesh is along its edges, shown for the halfedge data structure in Fig. D.4.

Ideally subdivision will upsample the underlying structure. This means we need to pay attention to things like local tangents and curvature. To avoid high valences, the proper way to subdivide a mesh is to split and then flip edges. Loop subdivision [117], shown

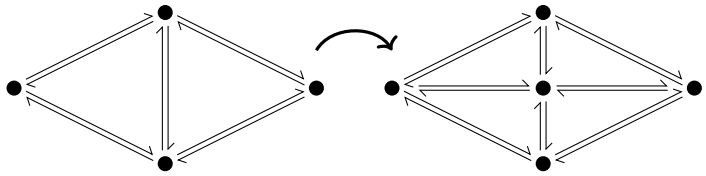


Figure D.4: Edge split operation, with halfedges shown.

in Fig. D.5 is an upsampling technique that preserves local tangents, curvature, and minimizes valence for triangle meshes (Catmull-Clark [118] is the corresponding algorithm for quadrilateral meshes). In Loop sampling, we split every edge in the mesh in half in random order and then flip any new edge that touches both an old vertex and a new vertex. Then we reposition each new vertex as the sum of $\frac{3}{8}\times$ the positions of the vertices of the edge along which we split, and $\frac{1}{8}\times$ the positions of the vertices of the **next** halfedge of each of these vertices. This repositioning is called fairing (see below) or relaxation, and is usually the step responsible for preserving curvature.

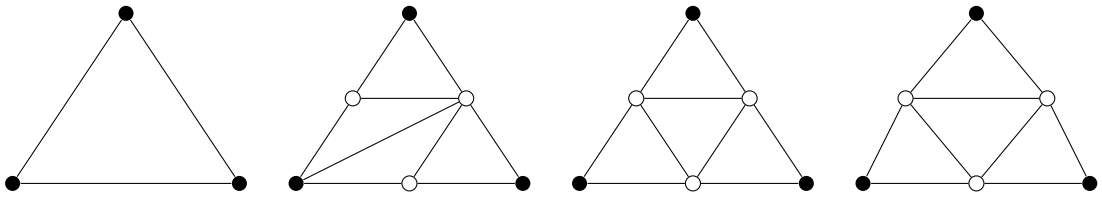


Figure D.5: The steps of Loop subdivision. From left to right: initial face, split face via splitting edges, flip edges that touch both an original and a new vertex, relax vertex positions to better represent shape of the underlying structure. Original vertices are filled in, new vertices are open circles.

All subdivision algorithms follow the same steps as the Loop algorithm: split, flip, fair (relax). Often we use algorithms that statistically drive a mesh toward high quality via the metrics mentioned above. For example, we may, in any order, split all edges longer than 20 percent of the mean edge length, flip random edges until all vertices have valence ≤ 6, and relax vertices to the mean position of their neighbors. The ideal subdivision algorithm will depend on your application.

Preserving structure during edge splits There is a temptation to simply split an edge in its middle and keep going. However, if we are trying to preserve the shape of the underlying structure, we can do better. Loop subdivision achieves this with its $\frac{3}{8}$-$\frac{1}{8}$ rule for splitting triangles.

We can develop a rule to split an edge defined by end points $\vec{\mathbf{x}}_0$ and $\vec{\mathbf{x}}_1$, sketched in Fig. D.6, in a way that preserves the curvature of the underlying structure by placing the split point at the center of a cubic fit through the end points of this edge. Because an edge exists in $\mathbb{R}^3$, it is necessary to select the plane on which the cubic is fit. The *x*-axis is naturally defined along the edge itself by $\hat{\vec{\mathbf{x}}} = \frac{\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||}$. A suitable choice of *y*-axis is the average of the two normals at the edge end points, $\hat{\vec{\mathbf{y}}} = \frac{\vec{\mathbf{n}}_0 + \vec{\mathbf{n}}_1}{||\vec{\mathbf{n}}_0 + \vec{\mathbf{n}}_1||}$.

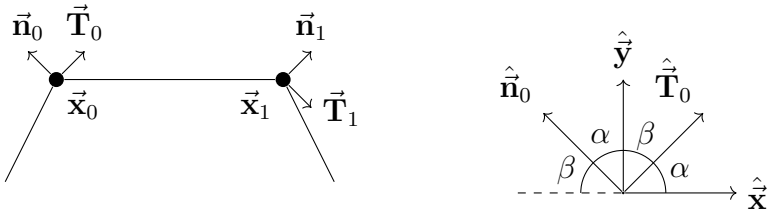


Figure D.6: Left: A mesh edge sketched in $\mathbb{R}^2$. **Right:** Angle relationships between the tangents and normals of the edge vertex $\vec{\mathbf{x}}_0$ in $\hat{\vec{\mathbf{x}}} \times \hat{\vec{\mathbf{y}}}$ space. $\hat{\vec{\mathbf{n}}}_0 = \vec{\mathbf{n}} \cdot \hat{\vec{\mathbf{y}}}$ and $\hat{\vec{\mathbf{T}}}_0 = \vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}}$.

Following Fig. D.6, let $x \in [0, 1]$ parameterize the distance from $\vec{\mathbf{x}}_0$ to $\vec{\mathbf{x}}_1$ in the $\hat{\vec{\mathbf{x}}}$ direction. The height y in the $\hat{\vec{\mathbf{y}}}$ direction is given by

$$y(x) = Ax^3 + Bx^2 + Cx + D$$
$$y'(x) = 3Ax^2 + 2Bx + C.$$

This equation is constrained by

$$\begin{aligned} y(0) &= 0 \\ y(1) &= 0 \\ y'(0) &= \vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} \\ y'(1) &= \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}}. \end{aligned}$$

Substituting these constraints into the equations above yields

$$\begin{aligned} y(0) = 0 &\implies D = 0 \\ y(1) = 0 &\implies B = -A - C \\ y'(0) = \vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} &\implies C = \vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} \\ y'(1) = \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} &\implies 3A + 2B + C = \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} \\ &\implies 3A + 2(-A - C) + C = \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} \\ &\implies A - C = \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} \\ &\implies A = \vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} + \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} \\ &\implies B = -2\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} - \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}}. \end{aligned}$$

Substituting $x = \frac{1}{2}$ into the original equation yields

$$\begin{aligned} y\left(\frac{1}{2}\right) &= \frac{\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} + \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}}}{8} - \frac{2\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} + \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}}}{4} + \frac{\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}}}{2} \\ &= \frac{\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} - \vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}}}{8}. \end{aligned}$$

By similar angles (see Fig. D.6, right), $\vec{\mathbf{T}}_0 \cdot \hat{\vec{\mathbf{y}}} = \vec{\mathbf{n}}_0 \cdot \hat{\vec{\mathbf{x}}}$ and $\vec{\mathbf{T}}_1 \cdot \hat{\vec{\mathbf{y}}} = \vec{\mathbf{n}}_1 \cdot \hat{\vec{\mathbf{x}}}$, yielding

$$y\left(\frac{1}{2}\right) = \frac{1}{8}(\vec{\mathbf{n}}_0 - \vec{\mathbf{n}}_1) \cdot \hat{\vec{\mathbf{x}}}.$$

Combining distances along $\hat{\vec{\mathbf{x}}}$ and $\hat{\vec{\mathbf{y}}}$, the position of the vertex splitting the edge is

$$\begin{aligned}
\vec{\mathbf{x}}_{\frac{1}{2}} &= \vec{\mathbf{x}}_0 + ||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0|| \left(\frac{1}{2}\hat{\vec{\mathbf{x}}} + y\left(\frac{1}{2}\right)\hat{\vec{\mathbf{y}}}\right) \\
&= \vec{\mathbf{x}}_0 + ||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0|| \left(\frac{1}{2}\hat{\vec{\mathbf{x}}} + \left(\frac{1}{8}(\vec{\mathbf{n}}_0 - \vec{\mathbf{n}}_1) \cdot \hat{\vec{\mathbf{x}}}\right)\hat{\vec{\mathbf{y}}}\right) \\
&= \vec{\mathbf{x}}_0 + ||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0|| \left(\frac{1}{2}\left(\frac{\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||}\right) + \left(\frac{1}{8}(\vec{\mathbf{n}}_0 - \vec{\mathbf{n}}_1) \cdot \left(\frac{\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||}\right)\right)\hat{\vec{\mathbf{y}}}\right) \\
&= \vec{\mathbf{x}}_0 + \frac{\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0}{2} + \left(\frac{(\vec{\mathbf{n}}_0 - \vec{\mathbf{n}}_1) \cdot (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0)}{8}\right)\hat{\vec{\mathbf{y}}} \\
&= \frac{\vec{\mathbf{x}}_0 + \vec{\mathbf{x}}_1}{2} + \left(\frac{(\vec{\mathbf{n}}_0 - \vec{\mathbf{n}}_1) \cdot (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0)}{8}\right)\hat{\vec{\mathbf{y}}}.
\end{aligned}$$

Downsampling

In cases where we want to reduce valence or increase local edge length/face size, we can improve mesh quality, or at least keep it the same, by downsampling. This has the advantage of reducing mesh storage space and computational time of further operations on the mesh.

Downsampling is usually achieved through edge collapse operations, which are the opposite of the edge split operation show in Fig. D.4. In edge collapse, the central vertex merges with the upper vertex in this figure (or another vertex, depending on which edge we collapse).

Choosing which edges get collapsed is a big part of downsampling. We may, for example, collapse any edge shorter than 20 percent of the mean edge length. We can go a step further and collapse edges according to which ones minimize the quadratic error between the downsampled approximation and the original mesh shape [119]. Often this

second option is unnecessary and comparatively expensive.

Fairing

Fairing operations smooth a mesh by shifting vertices in a way that minimizes a function on the surface [120]. Generally this is a curvature energy minimization. An easy approach is to iteratively shift each vertex toward the centroid of its neighbors. Over time this relaxes the surface to the point where curvature transitions smoothly. Lloyd relaxation is a popular Voronoi-based technique and a good starting point [121]. Fairing can also introduce regularization terms that can over-smooth the mesh based on metrics of choice.

Repair

All of the operations above work best on manifold meshes. A mesh is considered manifold if the one-ring neighborhood of every vertex has an Euler characteristic of 1. The Euler characteristic is defined as

$$\chi = V - E + F \tag{D.1}$$

where V is the number of vertices, E is the number of edges, and F is the number of faces in the object.

A mesh edge is considered singular if it is shared by > 2 faces. A vertex is considered singular if some of its incident edges are inaccessible by traversing its one-ring neighborhood[3]. Repair operations cut singularities out of the mesh and leave boundary edges [122]. However, meshes are still not manifold if they contain boundary edges (depending on who you ask, but I argue they are not in the halfedge data structure).

Connected cycles of boundary edges form holes in a mesh. Depending on the nature of

[3]Traversal involves moving in a twin-next fashion around the vertex, starting with its one emanating halfedge. If more than one one-ring neighborhood is joined to a single vertex, only one of these neighborhoods is traversable.

the cycle, holes can be closed via pinching [122] or by filling the hole with new triangles [123].

D.2 Curvature

Curvature is a measure of how much a curve deviates from a line. It is therefore natural to define curvature at a point on a function $f(s)$ as $\frac{1}{R}$ where R is the radius of a circle that touches $f(s)$ at that point and at least two other nearby points on $f(s)$, as shown in Fig. D.7.

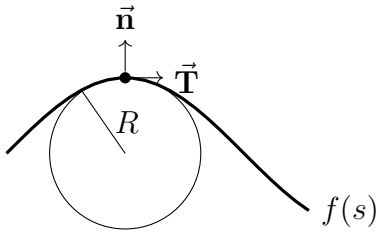


Figure D.7: A curve $f(s)$ with curvature measurement visualised at a point. $\vec{\mathbf{T}}$ and $\vec{\mathbf{n}}$ represent the tangent and normal vectors, respectively, at this point. A circle of radius R touches the point and at least two other points on the curve, so the curvature at this point is $\frac{1}{R}$.

If the radius of this circle is large, curvature is ≈ 0 and we can assume we have a line. As the radius shrinks, curvature increases.

Taking the intuitive definition a step further, curvature is how much the direction of $f(s)$ changes over a small distance. This can be algebraically defined as

$$\kappa = \left|\left|\frac{d\vec{\mathbf{T}}}{ds}\right|\right|$$

where $\vec{\mathbf{T}}$ is a tangent vector of a curve $f(s)$ at a point.

Suppose we choose that the unit normal $\vec{\mathbf{n}}_i$ at any point (x_i, y_i) on the circle points away from the circle center (x_c, y_c). Then, $\vec{\mathbf{n}}_i = \left[\frac{x_i - x_c}{\sqrt{(x_i-x_c)^2+(y_i-y_c)^2}}, \frac{y_i - y_c}{\sqrt{(x_i-x_c)^2+(y_i-y_c)^2}}\right] =$

$\left[\frac{x_i - x_c}{R}, \frac{y_i - y_c}{R}\right]$. Subsequently, $\vec{\mathbf{T}}_i = \left[-\frac{y_i - y_c}{R}, \frac{x_i - x_c}{R}\right]$. So, for two points on the circle $(x_0, y_0), (x_1, y_1)$,

$$\kappa = \frac{||d\vec{\mathbf{T}}||}{||ds||} = \frac{\frac{1}{R}\sqrt{(y_0 - y_1)^2 + (x_1 - x_0)^2}}{\sqrt{(y_1 - y_0)^2 + (x_1 - x_0)^2}} = \frac{1}{R}.$$

Thus, the definition of κ is in agreement with the circle-based definition of curvature.

At this point you may correctly surmise that curvature is only well-defined along a single curve. In 3D, each curve passing through a point may have a different curvature and it's up to us to decide which ones are worth paying attention to. Sometimes, in the case of a sphere, this is very easy as all curves on a sphere have the same curvature. In other cases, we must find the directions of principal curvature (section D.2.1) or select a similarly sensible plane on which we want to examine a curve (see the preserving structure discussion in D.1.4).

D.2.1 Principal curvature

Curvature is well-defined in $\mathbb{R}^2$, as described above. Curvature is defined in $\mathbb{R}^3$ only in $\mathbb{R}^2$ subspaces (planes). Thus, at every point in $\mathbb{R}^3$, there are infinitely many curvature values depending on the chose plane slicing through this point.

On a 3D mesh, we have two directions of principal curvature at each vertex, defined as the directions along the mesh of maximal, κ_1, and minimal, κ_2, curvature change at this vertex. For example, on a cylinder of radius r, the principal directions would be perpendicular to the cylinder, $\kappa_1 = \frac{1}{r}$, and along the cylinder, $\kappa_2 = 0$. $\vec{\mathbf{T}}_1$ and $\vec{\mathbf{T}}_2$ in the Darboux frame indicate these principal directions, which, along with the principal curvatures, are calculated from the curvature tensor at a vertex.

Darboux Frame

Each vertex on a mesh has its own orthonormal coordinate system comprised of the normal at that vertex $\vec{\mathbf{n}}$, and two tangents $\vec{\mathbf{T}}_1, \vec{\mathbf{T}}_2$, in the directions of principal curvature. This coordinate system, shown in Fig. D.8, is called the Darboux frame [124].

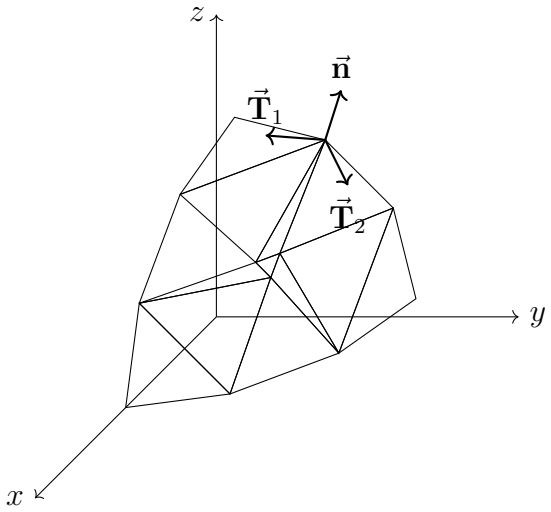


Figure D.8: Example Darboux frame for a single vertex $\vec{\mathbf{v}}$ on a mesh. The Darboux frame is $\vec{\mathbf{n}}, \vec{\mathbf{T}}_1, \vec{\mathbf{T}}_2$, the normal and the two tangents pointing along the principal directions of curvature. $\vec{\mathbf{T}}_1(\theta)$ and $\vec{\mathbf{T}}_2(\theta)$ are tangent vectors found at an angle θ to $\vec{\mathbf{T}}_1$ and $\vec{\mathbf{T}}_2$. $\vec{\mathbf{T}}_1(\theta)$ and $\vec{\mathbf{T}}_2(\theta)$ do not point along the principal directions of curvature, but may be used to reconstruct $\vec{\mathbf{T}}_1$ and $\vec{\mathbf{T}}_2$, as described in [99].

A way of calculating curvature at a mesh vertex

An easy way to calculate curvature at a vertex is to use the definition of curvature and note that by similar angles, in a 2D space, $||d\vec{\mathbf{T}}|| = ||d\vec{\mathbf{n}}||$. Then, $\kappa_{\vec{\mathbf{v}}}(\vec{\mathbf{T}}) = \frac{||\vec{\mathbf{n}}_2 - \vec{\mathbf{n}}_1||}{||\vec{\mathbf{v}}_2 - \vec{\mathbf{v}}_1||}$. However, as shown in Fig. D.9, this can mischaracterize a spike as having the same curvature as a circle. This problem does not arise if we use edge instead of vertex normals to calculate the curvature. To capture the accurate curvature of a vertex on a mesh, it is necessary to sum the contributions of all incident edges. To do this, we construct a curvature tensor.

The curvature tensor

The **curvature tensor** of a mesh S is a map which assigns each vertex $\vec{\mathbf{v}} \in S$ a directional curvature $\kappa_{\vec{\mathbf{v}}}(\vec{\mathbf{T}})$ where $\vec{\mathbf{T}}$ is a unit vector tangent to S at $\vec{\mathbf{v}}$. That is, $\kappa_{\vec{\mathbf{v}}}(\vec{\mathbf{T}})$ describes curvature in every direction at a point $\vec{\mathbf{v}}$. This means it is a linear combination of orthogonal

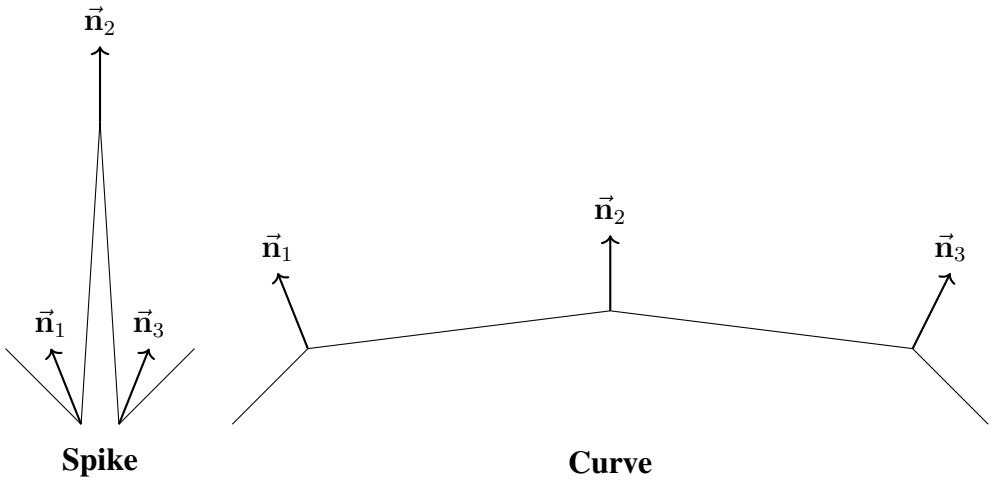


Figure D.9: Normals on and near a spike can be similar in direction and, therefore, won't give the spike the high curvature value it deserves. Notice that the spike and the curve both have roughly the same angular displacement between normals, which means dT will be roughly the same, and have the same ds. However, the spike has way higher curvature.

vectors at a point $\vec{\mathbf{v}}$, specifically,

$$\kappa_{\vec{\mathbf{v}}}(\vec{\mathbf{T}}) = \begin{bmatrix} n \\ t_1 \\ t_2 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \kappa_1^{\vec{\mathbf{v}}} & 0 \\ 0 & 0 & \kappa_2^{\vec{\mathbf{v}}} \end{bmatrix} \begin{bmatrix} n \\ t_1 \\ t_2 \end{bmatrix}$$

where $n, t_1, t_2 \in \mathbb{R}$ are coefficients such that $\vec{\mathbf{T}} = n\vec{\mathbf{n}}^{\vec{\mathbf{v}}} + t_1\vec{\mathbf{T}}_1^{\vec{\mathbf{v}}} + t_2\vec{\mathbf{T}}_2^{\vec{\mathbf{v}}}$, and $\vec{\mathbf{n}}^{\vec{\mathbf{v}}}$, $\vec{\mathbf{T}}_1^{\vec{\mathbf{v}}}$, and $\vec{\mathbf{T}}_2^{\vec{\mathbf{v}}}$ form the Darboux frame at $\vec{\mathbf{v}}$. Thus, the eigenvalues and eigenvectors of the curvature tensor restricted to the tangent plane orthogonal to $\vec{\mathbf{n}}^{\vec{\mathbf{v}}}$ at point $\vec{\mathbf{v}}$ are the principal curvatures and principal directions, respectively. Construction of this tensor evaluated at $\vec{\mathbf{v}}$ is easily done by summing the contributions of tangent planes formed by incident faces [99].

D.2.2 Mean and Gaussian curvature

Assume κ_1, κ_2 are principal curvatures at a vertex $\vec{\mathbf{v}}$. The mean curvature at $\vec{\mathbf{v}}$ is the average curvature of the surface and is defined as

$$H = \frac{1}{2}(\kappa_1 + \kappa_2).$$

Gaussian curvature at $\vec{\mathbf{v}}$ is defined as

$$K = \kappa_1 \kappa_2$$

and is used to find saddle points on the mesh. If at $\vec{\mathbf{v}}$ the principal curvature in one direction is positive and the other is negative, $K < 0$, and this vertex is a saddle point.

Mean and Gaussian curvature are important features in membrane biology. For example, many fission and fusion processes are thought to be catalyzed by proteins associated with negatively-curved necks in membranes [100].

D.2.3 Minimizing curvature on a mesh

Often we are interested in not only calculating, but minimizing curvature on a mesh. As mentioned in Chapter 1, curvature minimization can used to achieve better estimates of subcellular shape [69, 70].

Vertex curvature as a function of the distance from a vertex to the centroid of the vertex's neighbors is monotonic

It is therefore helpful to understand the relationship between curvature and a vertex's position relative to its neighbors. Here, using two neighbors, we demonstrate that the relationship between curvature at a vertex and the distance from the centroid of this vertex's

neighbors to the vertex is monotonic. What follows is a heuristic argument using two neighbors. The extension to $N \in \mathbb{N}$ neighbors is left as an exercise.

The following relationships are sketched in Fig. D.10.

$$\begin{aligned}
\vec{\mathbf{x}}_0 &= \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{p}}}{2} \\
\vec{\mathbf{x}}'_0 &= \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{p}} + \vec{\mathbf{dp}}}{2} \\
\vec{\mathbf{x}}_1 &= \frac{\vec{\mathbf{v}}_1 + \vec{\mathbf{p}}}{2} \\
\vec{\mathbf{x}}'_1 &= \frac{\vec{\mathbf{v}}_1 + \vec{\mathbf{p}} + \vec{\mathbf{dp}}}{2}
\end{aligned}$$

$$\begin{aligned}
\vec{\mathbf{T}}_0 &= \frac{\vec{\mathbf{p}} - \vec{\mathbf{v}}_0}{||\vec{\mathbf{p}} - \vec{\mathbf{v}}_0||} \\
\vec{\mathbf{T}}'_0 &= \frac{\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_0}{||\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_0||} \\
\vec{\mathbf{T}}_1 &= \frac{\vec{\mathbf{v}}_1 - \vec{\mathbf{p}}}{||\vec{\mathbf{v}}_1 - \vec{\mathbf{p}}||} \\
\vec{\mathbf{T}}'_1 &= \frac{\vec{\mathbf{v}}_1 - \vec{\mathbf{p}} - \vec{\mathbf{dp}}}{||\vec{\mathbf{v}}_1 - \vec{\mathbf{p}} - \vec{\mathbf{dp}}||}
\end{aligned}$$

Consider the curvature at vertex $\vec{\mathbf{p}}$ and at a vertex $\vec{\mathbf{p}} + \vec{\mathbf{dp}}$. To establish that curvature as a function of the distance between the centroid of its neighbors and the vertex is monotonic, we need to show that when

$$\left|\left|\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1}{2}\right|\right| \geq \left|\left|\vec{\mathbf{p}} - \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1}{2}\right|\right|, \tag{D.2}$$

then our curvatures satisfy the relationship

$$\frac{||\vec{\mathbf{T}}'_1 - \vec{\mathbf{T}}'_0||}{||\vec{\mathbf{x}}'_1 - \vec{\mathbf{x}}'_0||} \geq \frac{||\vec{\mathbf{T}}_1 - \vec{\mathbf{T}}_0||}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||}. \tag{D.3}$$

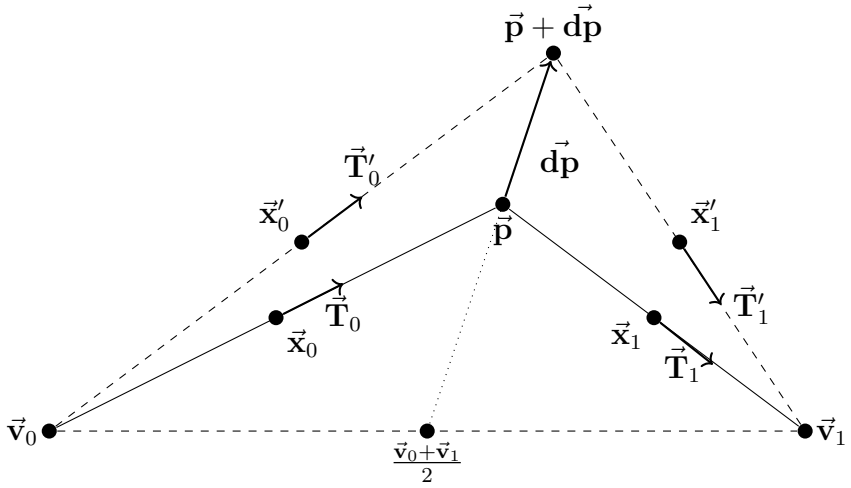


Figure D.10: Graphical relationship between a vertex $\vec{\mathbf{p}}$ displaced $\vec{\mathbf{dp}}$ from its original position and its neighbors and local tangents before and after displacement.

We compute the curvatures as follows.

$$\begin{aligned}
\frac{||\vec{\mathbf{T}}_1 - \vec{\mathbf{T}}_0||}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||} &= \frac{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{p}}}{||\vec{\mathbf{v}}_1-\vec{\mathbf{p}}||} - \frac{\vec{\mathbf{p}}-\vec{\mathbf{v}}_0}{||\vec{\mathbf{p}}-\vec{\mathbf{v}}_0||}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1+\vec{\mathbf{p}}}{2} - \frac{\vec{\mathbf{v}}_0+\vec{\mathbf{p}}}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)^2(\vec{\mathbf{v}}_1-\vec{\mathbf{p}})-(\vec{\mathbf{p}}-\vec{\mathbf{v}}1)^2(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)}{(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)^2(\vec{\mathbf{p}}-\vec{\mathbf{v}}1)^2}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{-(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)-(\vec{\mathbf{p}}-\vec{\mathbf{v}}1)}{(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)(\vec{\mathbf{p}}-\vec{\mathbf{v}}1)}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{\vec{\mathbf{v}}_0+\vec{\mathbf{v}}_1-2\vec{\mathbf{p}}}{(\vec{\mathbf{p}}-\vec{\mathbf{v}}_0)(\vec{\mathbf{p}}-\vec{\mathbf{v}}_1)}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{2||\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1 - 2\vec{\mathbf{p}}||}{||(\vec{\mathbf{p}} - \vec{\mathbf{v}}_0)(\vec{\mathbf{p}} - \vec{\mathbf{v}}_1)||||\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0||}
\end{aligned}$$

and

$$\begin{aligned}
\frac{||\vec{\mathbf{T}}_1' - \vec{\mathbf{T}}_0'||}{||\vec{\mathbf{x}}_1' - \vec{\mathbf{x}}_0'||} &= \frac{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{p}}-\vec{\mathbf{dp}}}{||\vec{\mathbf{v}}_1-\vec{\mathbf{p}}-\vec{\mathbf{dp}}||} - \frac{\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0}{||\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0||}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1+\vec{\mathbf{p}}+\vec{\mathbf{dp}}}{2} - \frac{\vec{\mathbf{v}}_0+\vec{\mathbf{p}}+\vec{\mathbf{dp}}}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)^2(\vec{\mathbf{v}}_1-\vec{\mathbf{p}}-\vec{\mathbf{dp}})-(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_1)^2(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)}{(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)^2(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_1)^2}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{-(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)-(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_1)}{(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_1)}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{\left|\left|\frac{\vec{\mathbf{v}}_0+\vec{\mathbf{v}}_1-2\vec{\mathbf{p}}-2\vec{\mathbf{dp}}}{(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_0)(\vec{\mathbf{p}}+\vec{\mathbf{dp}}-\vec{\mathbf{v}}_1)}\right|\right|}{\left|\left|\frac{\vec{\mathbf{v}}_1-\vec{\mathbf{v}}_0}{2}\right|\right|} \\
&= \frac{2||\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1 - 2\vec{\mathbf{p}} - 2\vec{\mathbf{dp}}||}{||(\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_0)(\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_1)||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)||}.
\end{aligned}$$

Let's assume (D.2). Examining the numerators of our curvatures, we see

$$\begin{aligned}
2||\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1 - 2\vec{\mathbf{p}} - 2\vec{\mathbf{dp}}|| &= 2||2\vec{\mathbf{p}} + 2\vec{\mathbf{dp}} - \vec{\mathbf{v}}_0 - \vec{\mathbf{v}}_1|| \\
&= 2\left|\left|\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1}{2}\right|\right| \\
&\geq 2\left|\left|\vec{\mathbf{p}} - \frac{\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1}{2}\right|\right| \\
&= 2||2\vec{\mathbf{p}} - \vec{\mathbf{v}}_0 - \vec{\mathbf{v}}_1|| \\
&= 2||\vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_1 - 2\vec{\mathbf{p}}||,
\end{aligned}$$

which is on the road to satisfying (D.3). In order to ensure (D.3) is true, the denominators

must also satisfy this relationship. Examining these, we see

$$
\begin{aligned}
&||(\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_0)(\vec{\mathbf{p}} + \vec{\mathbf{dp}} - \vec{\mathbf{v}}_1)||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)|| \\
&= || - ||\vec{\mathbf{p}}||^2 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_1 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_0 - \vec{\mathbf{v}}_0 \cdot \vec{\mathbf{v}}_1 + \vec{\mathbf{v}}_0 \cdot \vec{\mathbf{dp}} + \vec{\mathbf{v}}_1 \cdot \vec{\mathbf{dp}} - ||\vec{\mathbf{dp}}||^2 \\
&\qquad - 2\vec{\mathbf{p}} \cdot \vec{\mathbf{dp}}||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)|| \\
&\leq (|| - ||\vec{\mathbf{p}}||^2 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_1 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_0 - \vec{\mathbf{v}}_0 \cdot \vec{\mathbf{v}}_1|| + ||\vec{\mathbf{v}}_0 \cdot \vec{\mathbf{dp}} + \vec{\mathbf{v}}_1 \cdot \vec{\mathbf{dp}} - ||\vec{\mathbf{dp}}||^2 \\
&\qquad - 2\vec{\mathbf{p}} \cdot \vec{\mathbf{dp}}||)||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)|| \\
&\leq || - ||\vec{\mathbf{p}}||^2 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_1 + \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_0 - \vec{\mathbf{v}}_0 \cdot \vec{\mathbf{v}}_1||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)|| \\
&= ||||\vec{\mathbf{p}}||^2 - \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_1 - \vec{\mathbf{p}} \cdot \vec{\mathbf{v}}_0 + \vec{\mathbf{v}}_0 \cdot \vec{\mathbf{v}}_1||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)|| \\
&= ||(\vec{\mathbf{p}} - \vec{\mathbf{v}}_0)(\vec{\mathbf{p}} - \vec{\mathbf{v}}1)||||(\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_0)||
\end{aligned}
$$

where we used the triangle inequality in step 2.

Since the numerator of $\frac{||\vec{\mathbf{T}}'_1 - \vec{\mathbf{T}}'_0||}{||\vec{\mathbf{x}}'_1 - \vec{\mathbf{x}}'_0||}$ is greater than or equal to the numerator of $\frac{||\vec{\mathbf{T}}_1 - \vec{\mathbf{T}}_0||}{||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_0||}$ under the condition (D.2), and this relationship holds in general for the denominator, (D.3) is satisfied and curvature as a function of the distance between a vertex and the centroid of its neighboring vertices is monotonic.

This means we can locally minimize curvature on a mesh simply by moving a vertex toward the centroid of its neighbors.

D.2.4 The Canham-Helfrich energy functional and its discretizations

The Canham-Helfrich (C-H) energy functional is the canonical description of bending energy in lipid membranes [94, 96]. Minimizing C-H energy on a membrane surface minimizes the curvature subject to lipid stiffness coefficients κ, which have been measured and reported for different phospholipid groups. If a mesh is designed to represent a membrane, this yields a biophysically accurate curvature approximation.

The Laplace discretization of the C-H is given by

$$E_{\text{Lap}} = \frac{\kappa}{2} \sum_i \frac{1}{\Omega_i} \left[\sum_{j(i)} \vec{\mathbf{v}}_i - \vec{\mathbf{v}}_j \right]^2 ,$$

where Ω_i is the sum of the areas of surface triangles adjacent to vertex at index i, and $j(i)$ are the indexes of the vertices neighboring vertex i. Note that this is remarkably close to minimizing the difference between a vertex and the mean of its neighbors. Conveniently, it is also a preferred discretization of the C-H, yielding more accurate results over a broader range of structures than other discretizations [94, 125]. Cotangent weighting is sometimes used to improve the Laplace estimate of curvature on a mesh where triangles are not equilateral [126].

D.3 Optimization routines

The goal of surface reconstruction from point clouds, described in Section 2.3.3 and Chapter 5, is to fit a continuous plane to point data. Fitting a continuous function to points is a well-studied problem, applied in everything from drawing straight lines on a sheet of paper to modern deep learning techniques. A popular way of fitting data is to use an optimization routine, which minimizes the distance between a function and the data it is fitting.

This section briefly reviews gradient descent optimization routines used in the development of the algorithm described in Chapter 5. For more details and practical implementations of these and other fitting routines, please see [93, 127, 128].

D.3.1 Classic descent

Suppose a hiker is in some fog near the top of a mountain, as shown in Figure D.11. The fastest way to get down (although not necessarily alive) is to move in the direction of the

steepest drop they can see. This is the principle of gradient descent. In order to minimize a function $f(\vec{\mathbf{x}})$, it is possible to start with some initial guess $\vec{\mathbf{x}}_0$ and then move along the function in the direction of steepest descent.

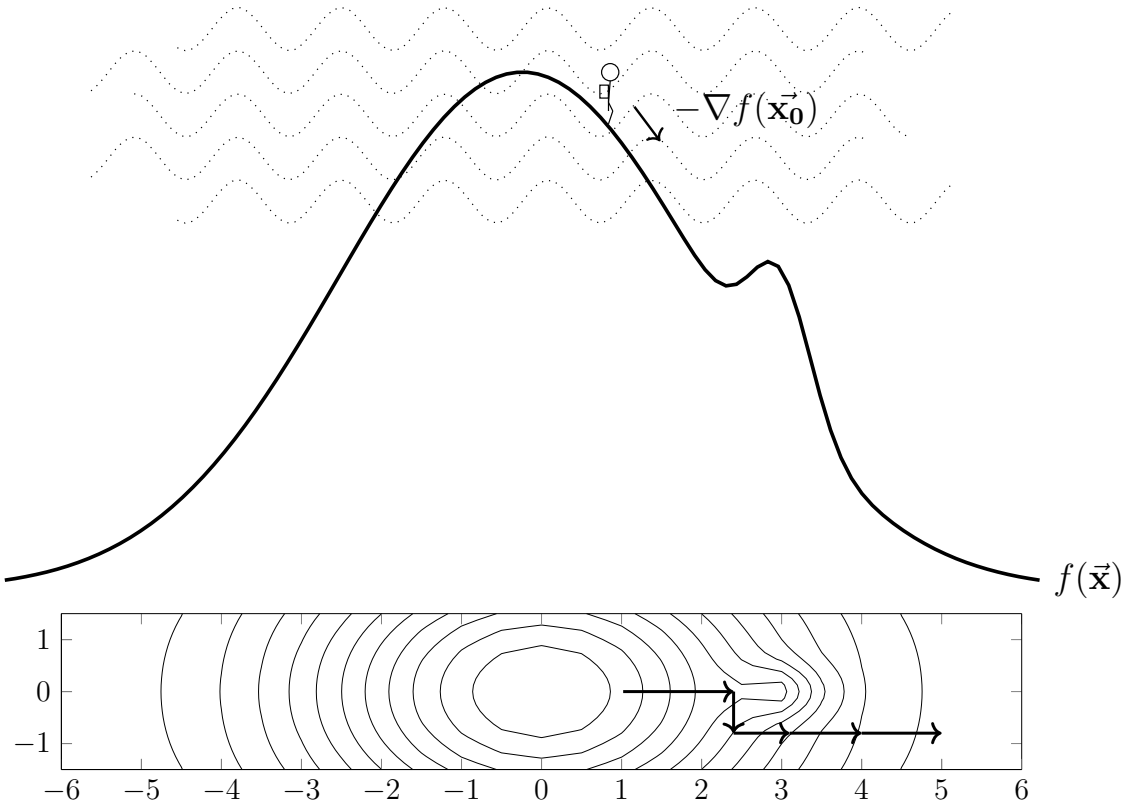


Figure D.11: A hiker is trapped in a fog on Function Mountain at position $\vec{\mathbf{x_0}} \in \mathbb{R}^{\mathbb{N}}$. *Top.* The problem imagined in one dimension. If the hiker moves in the direction of the negative gradient of the mountain at their location (gradient descent), they will eventually make it below the fog, but not over the second peak. *Bottom.* The problem imaged in two dimensions, from above. Here we can see that gradient descent causes us to move to the valley, and then slowly traverse to the bottom.

The direction of steepest descent is defined by the negative of the gradient of the function at $\vec{\mathbf{x}}_0$, $-\nabla f(\vec{\mathbf{x}}_0)$. The position updates

$$\vec{\mathbf{x}}_i = \vec{\mathbf{x}}_{i-1} - h\nabla f(\vec{\mathbf{x}}_{i-1}) \tag{D.4}$$

where $h \in \mathbb{R}$ is the step size until $||\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_{i-1}|| < \varepsilon$ where $\varepsilon \in \mathbb{R}$ is the precision of the

solution [93]. This can be generalized to apply descent on $N \in \mathbb{N}$ points at a time.

D.3.2 Regularization, adaptive learning and stochastic gradient descent

The hiker in Figure D.11 will get stuck in the valley part way up Function Mountain if they following the procedure in Section D.3.1. Adaptive learning methods introduce variability in the update step, which allows for escape from local minimums such as this. These are particularly useful when fitting complex landscapes, such as the parameter space for a deep learning algorithm or the shape space for a 3D structure.

The classic method for escape is not adaptive at all, but is to simply subtract a constant value with magnitude greater than the size of the smaller hill. This is called regularization, where we effectively smooth the bump out of the function.

Another easy method is to add random noise to the calculated gradient at each step. This often allows the optimization to hop over local minimums. The magnitude of the noise should decrease with the number of steps taken to allow convergence.

Momentum-based techniques keep track of previous gradient values and add them as a weighted sum to the current gradient value. If our hiker is running down the hill, momentum may send them careening up the smaller hill when they get to the valley and continue down to the bottom of the mountain [127, 128].

More sophisticated techniques, such as Adam, RMSProp and AdaGrad use complex combinations of the gradient and low-order function moments [128, 129]. These are particularly useful in stochastic gradient descent.

Stochastic gradient descent is used when trying to fit a large number of points or points in a large number of dimensions to a function. Rather than calculating the update for every point at every step, at each step in stochastic gradient descent the points are subsampled

and updates are made based only on this subsample. This allows for faster fitting of a large sample size and traverses noisy search space effectively [127–129]. Unfortunately, these methods also love to overfit data.

D.3.3 Conjugate gradient descent

Consider the example in the bottom of Figure D.11, which demonstrates a standard gradient descent approach for getting the hiker down the mountain in two dimensions. The hiker first moves in the direction of $\nabla f(\vec{x_0})$ until they reach the valley before the small peak, then move in a direction tangent to the constant height of valley until they reach the boundary of this height in that direction, and then move to the bottom of the mountain.

There is obviously a more efficient approach to simply go down the mountain in the direction that bypasses the peak. This can be done with *conjugate gradient descent*. Instead of moving in the direction of the gradient, we treat the x and y directions as conjugate search directions—that is, they are linearly independent of one another, so moving along one does not spoil optimization along the other direction—on the mountain. We can then weight the contribution of shifting in each direction and sum them as a linear combination. This sends us directly down the mountain, bypassing the smaller peak, which is faster.

At every stage of conjugate gradient descent, a new descent direction, conjugate to all previous directions (as much as possible), is chosen. By magic (math we won't get into here), for minimizing quadratics, at each step it turns out we only need to look in two search directions to get a result conjugate to all previous search directions [93]. This means we can search large spaces with only a few parameters, a.k.a. conjugate gradient descent lets us inexpensively optimize sparse systems of equations.

Bibliography

[1] Armin Haupt and Nicolas Minc. How cells sense their own shape – mechanisms to probe cell geometry and their implications in cellular organization and function. *Journal of Cell Science*, 131(6):jcs214015, March 2018. ISSN 1477-9137, 0021-9533. doi: 10.1242/jcs.214015. URL https://journals.biologists.com/jcs/article/131/6/jcs214015/57077/How-cells-sense-their-own-shape-mechanisms-to.

[2] Ashok Prasad and Elaheh Alizadeh. Cell Form and Function: Interpreting and Controlling the Shape of Adherent Cells. *Trends in Biotechnology*, 37(4):347–357, April 2019. ISSN 01677799. doi: 10.1016/j.tibtech.2018.09.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0167779918302609.

[3] Catarina Dias and Jesper Nylandsted. Plasma membrane integrity in health and disease: significance and therapeutic potential. *Cell Discovery*, 7(1):4, December 2021. ISSN 2056-5968. doi: 10.1038/s41421-020-00233-2. URL http://www.nature.com/articles/s41421-020-00233-2.

[4] Joseph Wong, David Baddeley, Eric A Bushong, Zeyun Yu, Mark H Ellisman, Masahiko Hoshijima, and Christian Soeller. Nanoscale Distribution of Ryanodine Receptors and Caveolin-3 in Mouse Ventricular Myocytes: Dilation of T-Tubules near Junctions. *Biophys. J.*, 104(11):L22–L24, 2013. doi: 10.1016/j.bpj.2013.02.059. URL http://dx.doi.org/10.1016/j.bpj.2013.02.059. Publisher: Biophysical Society.

[5] François Quemeneur, Jon K. Sigurdsson, Marianne Renner, Paul J. Atzberger, Patricia Bassereau, and David Lacoste. Shape matters in protein mobility within membranes. *Proceedings of the National Academy of Sciences of the United States of America*, 111(14):5083–5087, 2014. doi: 10.1073/pnas.1321054111.

[6] L.M. Westrate, J.E. Lee, W.A. Prinz, and G.K. Voeltz. Form Follows Function: The Importance of Endoplasmic Reticulum Shape. *Annual Review of Biochemistry*, 84 (1):791–811, 2015. doi: 10.1146/annurev-biochem-072711-163501.

[7] Marko Kaksonen and Aurélien Roux. Mechanisms of clathrin-mediated endocytosis. *Nature Reviews Molecular Cell Biology*, 19(5):313–326, May 2018. ISSN 1471-0072, 1471-0080. doi: 10.1038/nrm.2017.132. URL http://www.nature.com/articles/nrm.2017.132.

[8] L. Colina-Tenorio, P. Horten, N. Pfanner, and H. Rampelt. Shaping the mitochondrial inner membrane in health and disease. *Journal of Internal Medicine*, 287(6): 645–664, June 2020. ISSN 0954-6820, 1365-2796. doi: 10.1111/joim.13031. URL https://onlinelibrary.wiley.com/doi/10.1111/joim.13031.

[9] Van Nguyen-Dinh and Eva Herker. Ultrastructural Features of Membranous Replication Organelles Induced by Positive-Stranded RNA Viruses. *Cells*, 10(9):2407, September 2021. ISSN 2073-4409. doi: 10.3390/cells10092407. URL https://www.mdpi.com/2073-4409/10/9/2407.

[10] Pei-I Tsai, Chin-Hsien Lin, Chung-Han Hsieh, Amanda M. Papakyrikos, Min Joo Kim, Valerio Napolioni, Carmen Schoor, Julien Couthouis, Ruey-Meei Wu, Zbigniew K. Wszolek, Dominic Winter, Michael D. Greicius, Owen A. Ross, and Xinnan Wang. PINK1 Phosphorylates MIC60/Mitofilin to Control Structural Plasticity of Mitochondrial Crista Junctions. *Molecular Cell*, 69(5):744–756.e6, March 2018. ISSN 10972765. doi: 10.1016/j.molcel.2018.01.026. URL https://linkinghub.elsevier.com/retrieve/pii/S1097276518300558.

[11] Estefanía Tarazón, Esther Roselló-Lletí, Ana Ortega, Carolina Gil-Cayuela, José Ramón González-Juanatey, Francisca Lago, Luis Martínez-Dolz, Manuel Portolés, and Miguel Rivera. Changes in human Golgi apparatus reflect new left ventricular dimensions and function in dilated cardiomyopathy patients. *European Journal of Heart Failure*, 19(2):280–282, February 2017. ISSN 1388-9842, 1879-0844. doi: 10.1002/ejhf.671. URL https://onlinelibrary.wiley.com/doi/10.1002/ejhf.671.

[12] Lena Fitting Kourkoutis, Jürgen M. Plitzko, and Wolfgang Baumeister. Electron Microscopy of Biological Materials at the Nanometer Scale. *Annual Review of Materials Research*, 42(1):33–58, 2012. doi: 10.1146/annurev-matsci-070511-155004.

[13] Peter D. Dahlberg and W. E. Moerner. Cryogenic Super-Resolution Fluorescence and Electron Microscopy Correlated at the Nanoscale. *Annual Review of Physical Chemistry*, 72(1):253–278, 2021. doi: 10.1146/annurev-physchem-090319-051546. URL https://doi.org/10.1146/annurev-physchem-090319-051546. _eprint: https://doi.org/10.1146/annurev-physchem-090319-051546.

[14] Gareth Griffiths and John Milton Lucocq. Antibodies for immunolabeling by light and electron microscopy: not for the faint hearted. *Histochemistry and Cell Biology*, 142(4):347–360, October 2014. ISSN 0948-6143, 1432-119X. doi: 10.1007/s00418-014-1263-5. URL http://link.springer.com/10.1007/s00418-014-1263-5.

[15] Corey W Hecksel, Michele C Darrow, Wei Dai, Jesús G Galaz-Montoya, Jessica A Chin, Patrick G Mitchell, Shurui Chen, Jemba Jakana, Michael F Schmid, and Wah Chiu. Quantifying Variability of Manual Annotation in Cryo-Electron Tomograms. *Microscopy and Microanalysis*, page 21, 2016.

[16] Larissa Heinrich, Davis Bennett, David Ackerman, Woohyun Park, John Bogovic, Nils Eckstein, Alyson Petruncio, Jody Clements, Song Pang, C. Shan Xu, Jan Funke, Wyatt Korff, Harald F. Hess, Jennifer Lippincott-Schwartz, Stephan Saalfeld, Aubrey V. Weigel, COSEM Project Team, Riasat Ali, Rebecca Arruda, Rohit Bahtra, and Destiny Nguyen. Whole-cell organelle segmentation in volume electron microscopy. *Nature*, 599(7883):141–146, November 2021. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-021-03977-3. URL https://www.nature.com/articles/s41586-021-03977-3.

[17] Michael J. Sanderson, Ian Smith, Ian Parker, and Martin D. Bootman. Fluorescence Microscopy. *Cold Spring Harbor Protocols*, 2014(10):pdb.top071795, October 2014. ISSN 1940-3402, 1559-6095. doi: 10.1101/pdb.top071795. URL http://www.cshprotocols.org/lookup/doi/10.1101/pdb.top071795.

[18] E. Abbe. Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung. *Archiv für Mikroskopische Anatomie*, 9(1):413–468, December 1873. ISSN 0176-7364. doi: 10.1007/BF02956173. URL https://doi.org/10.1007/BF02956173.

[19] Stefan W. Hell and Jan Wichmann. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. *Optics Letters*, 19(11):780, June 1994. ISSN 0146-9592, 1539-4794. doi: 10.1364/OL.19.000780. URL https://www.osapublishing.org/abstract.cfm?URI=ol-19-11-780.

[20] Robert M. Dickson, Andrew B. Cubitt, Roger Y. Tsien, and W. E. Moerner. On/off blinking and switching behaviour of single molecules of green fluorescent protein. *Nature*, 388(6640):355–358, July 1997. ISSN 0028-0836, 1476-4687. doi: 10.1038/41048. URL http://www.nature.com/articles/41048.

[21] Samuel T. Hess, Thanu P.K. Girirajan, and Michael D. Mason. Ultra-high resolution imaging by fluorescence photoactivation localization microscopy. *Biophysical Journal*, 91(11):4258–4272, 2006. doi: 10.1529/biophysj.106.091116.

[22] Eric Betzig, George H. Patterson, Rachid Sougrat, O. Wolf Lindwasser, Scott Olenych, Juan S. Bonifacino, Michael W. Davidson, Jennifer Lippincott-Schwartz, and Harald F. Hess. Imaging intracellular fluorescent proteins at nanometer resolution. *Science*, 313(5793):1642–1645, 2006. doi: 10.1126/science.1127344.

[23] Michael J. Rust, Mark Bates, and Xiaowei Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM). *Nature Methods*, 3(10): 793–795, 2006. ISSN 1548-7091. doi: 10.1038/nmeth929.

[24] Giuseppe Vicidomini, Paolo Bianchini, and Alberto Diaspro. STED super-resolved microscopy. *Nature Methods*, 15(3):173–182, March 2018. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.4593. URL http://www.nature.com/articles/nmeth.4593.

[25] Francesca Bottanelli, Emil B. Kromann, Edward S. Allgeyer, Roman S. Erdmann, Stephanie Wood Baguley, George Sirinakis, Alanna Schepartz, David Baddeley, Derek K. Toomre, James E. Rothman, and Joerg Bewersdorf. Two-colour live-cell nanoscale imaging of intracellular targets. *Nature Communications*, 7(1):10778, April 2016. ISSN 2041-1723. doi: 10.1038/ncomms10778. URL http://www.nature.com/articles/ncomms10778.

[26] Jonathan Tyson, Kevin Hu, Shuai Zheng, Phylicia Kidd, Neville Dadina, Ling Chu, Derek Toomre, Joerg Bewersdorf, and Alanna Schepartz. Extremely Bright, Near-IR Emitting Spontaneously Blinking Fluorophores Enable Ratiometric Multicolor Nanoscopy in Live Cells. *ACS Central Science*, 7(8):1419–1426, August 2021. ISSN 2374-7943, 2374-7951. doi: 10.1021/acscentsci.1c00670. URL https://pubs.acs.org/doi/10.1021/acscentsci.1c00670.

[27] Yongdeng Zhang, Lena K. Schroeder, Mark D. Lessard, Phylicia Kidd, Jeeyun Chung, Yuanbin Song, Lorena Benedetti, Yiming Li, Jonas Ries, Jonathan B. Grimm, Luke D. Lavis, Pietro De Camilli, James E. Rothman, David Baddeley, and Joerg Bewersdorf. Nanoscale subcellular architecture revealed by multicolor three-dimensional salvaged fluorescence imaging. *Nature Methods*, 17(2):225–231, 2020. doi: 10.1038/s41592-019-0676-4. URL http://dx.doi.org/10.1038/s41592-019-0676-4. Publisher: Springer US.

[28] Pascal de Boer, Jacob P Hoogenboom, and Ben N G Giepmans. Correlated light and electron microscopy: ultrastructure lights up! *Nature Methods*, 12(6):503–

513, June 2015. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.3400. URL http://www.nature.com/articles/nmeth.3400.

[29] Ons M'Saad and Joerg Bewersdorf. Light microscopy of proteins in their ultrastructural context. *Nature Communications*, 11(1):3850, December 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17523-8. URL http://www.nature.com/articles/s41467-020-17523-8.

[30] Zhe Liu, Luke D. Lavis, and Eric Betzig. Imaging Live-Cell Dynamics and Structure at the Single-Molecule Level. *Molecular Cell*, 58(4):644–659, May 2015. ISSN 10972765. doi: 10.1016/j.molcel.2015.02.033. URL https://linkinghub.elsevier.com/retrieve/pii/S1097276515001653.

[31] David Baddeley and Joerg Bewersdorf. Biological Insight from Super-Resolution Microscopy: What We Can Learn from Localization-Based Images. *Annual Review of Biochemistry*, 87(1):965–989, June 2018. ISSN 0066-4154. doi: 10.1146/annurev-biochem-060815-014801. URL https://doi.org/10.1146/annurev-biochem-060815-014801. Publisher: Annual Reviews.

[32] Christopher P. Toseland. Fluorescent labeling and modification of proteins. *Journal of Chemical Biology*, 6(3):85–95, July 2013. ISSN 1864-6158, 1864-6166. doi: 10.1007/s12154-013-0094-5. URL http://link.springer.com/10.1007/s12154-013-0094-5.

[33] James B. Pawley. *Handbook of biological confocal microscopy*. Springer, New York, 2007. ISBN 0-387-25921-X 978-0-387-25921-5.

[34] M. Born and E. Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 1999. ISBN 978-0-521-64222-4.

[35] Mickaël Lelek, Melina T. Gyparaki, Gerti Beliu, Florian Schueder, Juliette Griffié, Suliana Manley, Ralf Jungmann, Markus Sauer, Melike Lakadamyali, and Christophe Zimmer. Single-molecule localization microscopy. *Nature Reviews Methods Primers*, 1(1):39, December 2021. ISSN 2662-8449. doi: 10.1038/s43586-021-00038-x. URL http://www.nature.com/articles/s43586-021-00038-x.

[36] Jingyu Wang, Edward S. Allgeyer, George Sirinakis, Yongdeng Zhang, Kevin Hu, Mark D. Lessard, Yiming Li, Robin Diekmann, Michael A. Phillips, Ian M. Dobbie, Jonas Ries, Martin J. Booth, and Joerg Bewersdorf. Implementation of a 4Pi-SMS super-resolution microscope. *Nature Protocols*, 16(2):677–727, February 2021. ISSN 1754-2189, 1750-2799. doi: 10.1038/s41596-020-00428-7. URL http://www.nature.com/articles/s41596-020-00428-7.

[37] Andrew E S Barentine, Yu Lin, Miao Liu, Phylicia Kidd, Leonhard Balduf, Michael R Grace, Siyuan Wang, Joerg Bewersdorf, and David Baddeley. 3D Multicolor Nanoscopy at 10,000 Cells a Day. *bioRxiv*, 2019.

[38] Shin-nosuke Uno, Mako Kamiya, Toshitada Yoshihara, Ko Sugawara, Kohki Okabe, Mehmet C. Tarhan, Hiroyuki Fujita, Takashi Funatsu, Yasushi Okada, Seiji Tobita, and Yasuteru Urano. A spontaneously blinking fluorophore based on intramolecular spirocyclization for live-cell super-resolution imaging. *Nature Chemistry*, 6(8):681–689, August 2014. ISSN 1755-4330, 1755-4349. doi: 10.1038/nchem.2002. URL http://www.nature.com/articles/nchem.2002.

[39] YongKeun Park, Christian Depeursinge, and Gabriel Popescu. Quantitative phase imaging in biomedicine. *Nature Photonics*, 12(10):578–589, October 2018. ISSN 1749-4885, 1749-4893. doi: 10.1038/s41566-018-0253-x. URL http://www.nature.com/articles/s41566-018-0253-x.

[40] F. Zernike. Phase contrast, a new method for the microscopic observation of transparent objects. *Physica*, 9(7):686–698, 1942. ISSN 0031-8914. doi: https://doi.org/10.1016/S0031-8914(42)80035-X. URL https://www.sciencedirect.com/science/article/pii/S003189144280035X.

[41] Natan Shaked, Zeev Zalevsky, and Lisa Satterwhite, editors. *Biomedical optical phase microscopy and nanoscopy*. Academic Press, Oxford, 1 edition, 2012. ISBN 978-0-12-415871-9.

[42] Zhuo Wang, Larry Millet, Mustafa Mir, Huafeng Ding, Sakulsuk Unarunotai, John Rogers, Martha U. Gillette, and Gabriel Popescu. Spatial light interference microscopy (slim). *Optics Express*, 19(2):1016–1026, Jan 2011. doi: 10.1364/OE.19.001016. URL http://opg.optica.org/oe/abstract.cfm?URI=oe-19-2-1016.

[43] Tan H. Nguyen, Mikhail E. Kandel, Marcello Rubessa, Matthew B. Wheeler, and Gabriel Popescu. Gradient light interference microscopy for 3D imaging of unlabeled specimens. *Nature Communications*, 8:210, Aug 2017. doi: 10.1038/s41467-017-00190-7. URL https://doi.org/10.1038/s41467-017-00190-7.

[44] Michael Shribak. Differential Interference Microscopy. In *Biomedical Optical Phase Microscopy and Nanoscopy*. Academic Press, 2012.

[45] Michael Shribak. Quantitative orientation-independent differential interference contrast microscope with fast switching shear direction and bias modulation. *Journal of the Optical Society of America A*, 30(4):769–769, 2013. doi: 10.1364/josaa.30.000769.

[46] Michael Shribak, Kieran G. Larkin, and David Biggs. Mapping optical path length and image enhancement using quantitative orientation-independent differential interference contrast microscopy. *Journal of Biomedical Optics*, 22(1): 016006–016006, 2017. doi: 10.1117/1.JBO.22.1.016006. URL http://biomedicaloptics.spiedigitallibrary.org/article.aspx?doi=10.1117/1.JBO.22.1.016006.

[47] Anastasiya Trushko, Erik Schäffer, and Jonathon Howard. The growth speed of microtubules with xmap215-coated beads coupled to their ends is increased by tensile force. *Proceedings of the National Academy of Sciences*, 110(36):14670–14675, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1218053110. URL https://www.pnas.org/content/110/36/14670.

[48] J. Nixon-Abell, C. J. Obara, A. V. Weigel, D. Li, W. R. Legant, C. S. Xu, H. A. Pasolli, K. Harvey, H. F. Hess, E. Betzig, C. Blackstone, and J. Lippincott-Schwartz. Increased spatiotemporal resolution reveals highly dynamic dense tubular matrices in the peripheral ER. *Science*, 354(6311):aaf3928–aaf3928, October 2016. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaf3928. URL https://www.sciencemag.org/lookup/doi/10.1126/science.aaf3928.

[49] Zach Marin, Michael Graff, Andrew E. S. Barentine, Christian Soeller, Kenny Kwok Hin Chung, Lukas A. Fuentes, and David Baddeley. PYMEVisualize: an open-source tool for exploring 3D super-resolution data. *Nature Methods*, 18(6): 582–584, June 2021. ISSN 4159202101. doi: 10.1038/s41592-021-01165-9. URL http://www.nature.com/articles/s41592-021-01165-9.

[50] Mohamed El Beheiry and Maxime Dahan. ViSP: Representing single-particle localizations in three dimensions. *Nature Methods*, 10(8):689–690, 2013. doi: 10.1038/nmeth.2566. Publisher: Nature Publishing Group.

[51] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[52] Christopher H. Bohrer, Xinxing Yang, Shreyasi Thakur, Xiaoli Weng, Brian Tenner, Ryan McQuillen, Brian Ross, Matthew Wooten, Xin Chen, Jin Zhang, Elijah Roberts, Melike Lakadamyali, and Jie Xiao. A pairwise distance distribution correction (ddc) algorithm to eliminate blinking-caused artifacts in smlm. *Nature Methods*, 18:669, Jun 2021. doi: 10.1038/s41592-021-01154-y. URL https://doi.org/10.1038/s41592-021-01154-y.

[53] Ismail M. Khater, Ivan Robert Nabi, and Ghassan Hamarneh. A review of super-resolution single-molecule localization microscopy cluster analysis and quantification methods. *Patterns*, 1(3):100038, 2020. ISSN 2666-3899. doi: https://doi.org/10.1016/j.patter.2020.100038. URL https://www.sciencedirect.com/science/article/pii/S266638992030043X.

[54] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Rendering Point Clouds with Compute Shaders and Vertex Order Optimization. *arXiv:2104.07526 [cs]*, April 2021. URL http://arxiv.org/abs/2104.07526. arXiv: 2104.07526.

[55] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002. doi: 10.1145/571647.571648.

[56] Philip R. Nicovich, Dylan M. Owen, and Katharina Gaus. Turning single-molecule localization microscopy into a quantitative bioanalytical tool. *Nature Protocols*, 12 (3):453–461, 2017. doi: 10.1038/nprot.2016.166.

[57] Jens Rittscher. Characterization of Biological Processes through Automated Image Analysis. *Annual Review of Biomedical Engineering*, 12(1):315–344, July 2010. ISSN 1523-9829, 1545-4274. doi: 10.1146/annurev-bioeng-070909-105235. URL https://www.annualreviews.org/doi/10.1146/annurev-bioeng-070909-105235.

[58] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, July 2012. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.2089. URL http://www.nature.com/articles/nmeth.2089.

[59] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. *Nature Methods*, 16 (12):1233–1246, December 2019. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-019-0403-1. URL http://www.nature.com/articles/s41592-019-0403-1.

[60] B. Delaunay. Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1 (6):793–800, 1934.

[61] David Baddeley, Mark B. Cannell, and Christian Soeller. Visualization of localization microscopy data. *Microscopy and Microanalysis*, 16(1):64–72, 2010. doi: 10.1017/S143192760999122X. Publisher: Yale University Library.

[62] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2-3):127–153, July 2001. ISSN 09257721. doi: 10.1016/S0925-7721(01)00017-7. URL https://linkinghub.elsevier.com/retrieve/pii/S0925772101000177.

[63] Frédéric Cazals and Joachim Giesen. Delaunay Triangulation Based Surface Reconstruction: Ideas and Algorithms. Technical Report RR-5393, INRIA, November 2004. URL https://hal.inria.fr/inria-00070610.

[64] D. Meagher. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. Technical Report Technical Report IPL-TR-80-111, Rensselaer Polytechnic Institute, 1980.

[65] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013. doi: 10.1145/2487228.2487237.

[66] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0-89791-227-6. doi: 10.1145/37401.37422. URL https://doi.org/10.1145/37401.37422.

[67] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. *Computer Graphics Forum*, 24(2):195–201, 2005. doi: 10.1111/j.1467-8659.2005.00843.x. Publisher: IEEE.

[68] Kenneth A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992. doi: 10.1080/10586458.1992.10504253.

[69] Yong Zhang, Xin Zhou, Jianjun Zhou, and Zhong-Can Ou-Yang. TRICONCAVE SOLUTION TO THE HELFRICH VARIATION PROBLEM FOR THE SHAPE OF LIPID BILAYER VESICLES IS FOUND BY "SURFACE EVOLVER". *International Journal of Modern Physics B*, 16(03):511–517, January 2002. ISSN 0217-9792, 1793-6578. doi: 10.1142/S0217979202009317. URL https://www.worldscientific.com/doi/abs/10.1142/S0217979202009317.

[70] Yao Zhao, Sarah M. Schreiner, Peter K. Koo, Paolo Colombi, Megan C. King, and Simon G.J. Mochrie. Improved Determination of Subnuclear Position Enabled by Three-Dimensional Membrane Reconstruction. *Biophysical Journal*, 111(1):19–24, 2016. doi: 10.1016/j.bpj.2016.05.036. URL http://dx.doi.org/10.1016/j.bpj.2016.05.036. Publisher: Biophysical Society.

[71] Daniel Schröder, Joran Deschamps, Anindita Dasgupta, Ulf Matti, and Jonas Ries. Cost-efficient open source laser engine for microscopy. *Biomed. Opt. Express*, 11 (2):609–623, Feb 2020. doi: 10.1364/BOE.380815. URL http://opg.optica.org/boe/abstract.cfm?URI=boe-11-2-609.

[72] Simon Alberti, Amy Gladfelter, and Tanja Mittag. Considerations and Challenges in Studying Liquid-Liquid Phase Separation and Biomolecular Condensates. *Cell*, 176(3):419–434, January 2019. ISSN 00928674. doi: 10.1016/j.cell.2018.12.035. URL https://linkinghub.elsevier.com/retrieve/pii/S0092867418316490.

[73] Edward M. Courchaine, Andrew E.S. Barentine, Korinna Straube, Dong-Ryoung Lee, Joerg Bewersdorf, and Karla M. Neugebauer. DMA-tudor interaction modules control the specificity of in vivo condensates. *Cell*, 184(14):3612–3625.e17, July 2021. ISSN 00928674. doi: 10.1016/j.cell.2021.05.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0092867421006255.

[74] Daniel Burke, Brian Patton, Fang Huang, Joerg Bewersdorf, and Martin J. Booth. Adaptive optics correction of specimen-induced aberrations in single-molecule switching microscopy. *Optica*, 2(2):177, February 2015. ISSN 2334-2536. doi: 10.1364/OPTICA.2.000177. URL https://opg.optica.org/abstract.cfm?URI=optica-2-2-177.

[75] Marijn E. Siemons, Naomi A. K. Hanemaaijer, Maarten H. P. Kole, and Lukas C. Kapitein. Robust adaptive optics for localization microscopy deep in complex tissue. *Nature Communications*, 12(1):3407, December 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23647-2. URL http://www.nature.com/articles/s41467-021-23647-2.

[76] David Baddeley. *Precision measurements with SMI and 4Pi Microscopy*. PhD thesis, University of Heidelberg, 2007. URL http://archiv.ub.uni-heidelberg.de/volltextserver/7960/1/thesis_baddeley.pdf. Issue: October.

[77] B. D. Ripley. The second-order analysis of stationary point processes. *Journal of Applied Probability*, 13(2):255–266, June 1976. ISSN 0021-9002, 1475-6072. doi: 10.2307/3212829. URL https://www.cambridge.org/core/product/identifier/S0021900200094328/type/journal_article.

[78] Mark Bates, Sara A. Jones, and Xiaowei Zhuang. Stochastic optical reconstruction microscopy (STORM): A method for superresolution fluorescence imaging. *Cold Spring Harbor Protocols*, 8(6):498–520, 2013. doi: 10.1101/pdb.top075143.

[79] Joshua Yoon, Colin J. Comerci, Lucien E. Weiss, Ljiljana Milenkovic, Tim Stearns, and W. E. Moerner. Revealing Nanoscale Morphology of the Primary Cilium Using Super-Resolution Fluorescence Microscopy. *Biophysical Journal*, 116(2):319–329, 2019. doi: 10.1016/j.bpj.2018.11.3136. URL https://doi.org/10.1016/j.bpj.2018.11.3136. Publisher: Biophysical Society.

[80] Geoffrey C. Rollins, Jae Yen Shin, Carlos Bustamante, and Steve Pressé. Stochastic approach to the molecular counting problem in superresolution microscopy. *Proceedings of the National Academy of Sciences*, 112(2):E110–E118, 2015. ISSN 1091-6490 (Electronic)\r0027-8424 (Linking). doi: 10.1073/pnas.1408071112. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1408071112.

[81] Carla Coltharp, Xinxing Yang, and Jie Xiao. Quantitative analysis of single-molecule superresolution images. *Current Opinion in Structural Biology*, 28(1): 112–121, 2014. doi: 10.1016/j.sbi.2014.08.008.

[82] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R.J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-

Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[83] David Baddeley, David Crossman, Sabrina Rossberger, Juliette E Cheyne, Johanna M Montgomery, Isuru D Jayasinghe, Christoph Cremer, Mark B Cannell, and Christian Soeller. 4D super-resolution microscopy with conventional fluorophores and single wavelength excitation in optically thick cells and tissues. *PLoS One*, 6(5), 2011. ISSN 1932-6203 (Electronic) 1932-6203 (Linking). doi: 10.1371/journal.pone.0020645.

[84] David J Crossman, Yufeng Hou, Isuru Jayasinghe, David Baddeley, and Christian Soeller. Combining confocal and single molecule localisation microscopy: A correlative approach to multi-scale tissue imaging. *Methods*, 88:98–108, 2015. doi: 10.1016/j.ymeth.2015.03.011. URL http://dx.doi.org/10.1016/j.ymeth.2015.03.011. Publisher: Elsevier Inc.

[85] Ruisheng Lin, Alexander H. Clowsley, Tobias Lutz, David Baddeley, and Christian Soeller. 3d super-resolution microscopy performance and quantitative analysis assessment using dna-paint and dna origami test samples. *Methods*, 174:56–71, 2020. ISSN 1046-2023. doi: https://doi.org/10.1016/j.ymeth.2019.05.018. URL https://www.sciencedirect.com/science/article/pii/S1046202318304262. Progress in Super-resolution Fluorescence Microscopy.

[86] Charles Bond, Adriana N. Santiago-Ruiz, Qing Tang, and Melike Lakadamyali. Technological advances in super-resolution microscopy to study cellular processes. *Molecular Cell*, 82(2):315–332, 2022. ISSN 1097-2765. doi: https://doi.org/10.1016/j.molcel.2021.12.022. URL https://www.sciencedirect.com/science/article/pii/S1097276521010844.

[87] Ron Milo and Rob Phillips. *Cell biology by the numbers*. Taylor & Francis Group, 2016. ISBN 978-0-8153-4537-4 0-8153-4537-2.

[88] Zhien Wang and Massimo Menenti. Challenges and opportunities in lidar remote sensing. *Frontiers in Remote Sensing*, 2, 2021. ISSN 2673-6187. doi: 10.3389/frsen.2021.641723. URL https://www.frontiersin.org/article/10.3389/frsen.2021.641723.

[89] Brian Curless. From range scans to 3d models. *SIGGRAPH Comput. Graph.*, 33 (4):38–41, nov 1999. ISSN 0097-8930. doi: 10.1145/345370.345399. URL https://doi.org/10.1145/345370.345399.

[90] Mark Terasaki, Tom Shemesh, Narayanan Kasthuri, Robin W. Klemm, Richard Schalek, Kenneth J. Hayworth, Arthur R. Hand, Maya Yankova, Greg Huber,

Jeff W. Lichtman, Tom A. Rapoport, and Michael M. Kozlov. Stacked Endoplasmic Reticulum Sheets Are Connected by Helicoidal Membrane Motifs. *Cell*, 154(2):285–296, July 2013. doi: 10.1016/j.cell.2013.06.031. URL http://dx.doi.org/10.1016/j.cell.2013.06.031. Publisher: Elsevier Inc.

[91] Lena K. Schroeder, Andrew E.S. Barentine, Holly Merta, Sarah Schweighofer, Yongdeng Zhang, David Baddeley, Joerg Bewersdorf, and Shirin Bahmanyar. Dynamic nanoscale morphology of the ER surveyed by STED microscopy. *The Journal of Cell Biology*, pages jcb.201809107–jcb.201809107, 2018. doi: 10.1083/jcb.201809107. URL http://www.jcb.org/lookup/doi/10.1083/jcb.201809107.

[92] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. *ACM International Conference Proceeding Series*, 71:185–192, 2004. doi: 10.1145/1057432.1057457.

[93] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007. ISBN 0521880688.

[94] D. Nelson, T. Piran, and S. Weinberg. *Statistical Mechanics of Membranes and Surfaces*. World Scientific, 2004. ISBN 981-238-760-9. doi: 10.1142/9789814541602. URL https://www.worldscientific.com/doi/abs/10.1142/9789814541602.

[95] T M Fischer. Bending stiffness of lipid bilayers. I. Bilayer couple or single-layer bending? *Biophysical journal*, 63(5):1328–1335, November 1992. ISSN 0006-3495. doi: 10.1016/S0006-3495(92)81710-1. URL https://pubmed.ncbi.nlm.nih.gov/1477282.

[96] Markus Deserno. Fluid lipid membranes - aprimer, 2007. URL https://www.cmu.edu/biolphys/deserno/pdf/membrane_theory.pdf.

[97] Laura Picas, Felix Rico, and Simon Scheuring. Direct measurement of the mechanical properties of lipid phases in supported bilayers. *Biophysical journal*, 102(1): L01–L3, January 2012. ISSN 1542-0086. doi: 10.1016/j.bpj.2011.11.4001. URL https://pubmed.ncbi.nlm.nih.gov/22225813. Edition: 2012/01/03 Publisher: The Biophysical Society.

[98] Joerg Schnitzbauer, Maximilian T Strauss, Thomas Schlichthaerle, Florian Schueder, and Ralf Jungmann. Super-resolution microscopy with dna-paint. *Nature Protocols*, 12:1198, Jun 2017. doi: 10.1038/nprot.2017.024. URL https://doi.org/10.1038/nprot.2017.024.

[99] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907, Cambridge, MA, USA, June 1995. IEEE. doi: 10.1109/ICCV.1995.466840.

[100] Patricia Bassereau, Rui Jin, Tobias Baumgart, Markus Deserno, Rumiana Dimova, Vadim A Frolov, Pavel V Bashkirov, Helmut Grubmüller, Reinhard Jahn, H Jelger Risselada, Ludger Johannes, Michael M Kozlov, Reinhard Lipowsky, Thomas J Pucadyil, Wade F Zeno, Jeanne C Stachowiak, Dimitrios Stamou, Artù Breuer, Line Lauritsen, Camille Simon, Cécile Sykes, Gregory A Voth, and Thomas R Weikl. The 2018 biomembrane curvature and remodeling roadmap. *Journal of Physics D: Applied Physics*, 51(34):343001, jul 2018. doi: 10.1088/1361-6463/aacb98. URL https://doi.org/10.1088/1361-6463/aacb98.

[101] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, USA, 2004. ISBN 0735619670.

[102] Jonas Ries. Smap: a modular super-resolution microscopy analysis platform for smlm data. *Nature Methods*, 17:870, Sep 2020. doi: 10.1038/s41592-020-0938-1. URL https://doi.org/10.1038/s41592-020-0938-1.

[103] Sebastian van de Linde. Single-molecule localization microscopy analysis with ImageJ. *Journal of Physics D: Applied Physics*, 52(20):203002, mar 2019. doi: 10.1088/1361-6463/ab092f. URL https://doi.org/10.1088/1361-6463/ab092f.

[104] Mariano Bizzarri, Douglas E. Brash, James Briscoe, Verônica A. Grieneisen, Claudio D. Stern, and Michael Levin. A call for a better understanding of causation in cell biology. *Nature Reviews Molecular Cell Biology*, 20(5):261–262, May 2019. ISSN 1471-0080. doi: 10.1038/s41580-019-0127-1. URL https://doi.org/10.1038/s41580-019-0127-1.

[105] Marco Vilela and Gaudenz Danuser. What's wrong with correlative experiments? *Nature Cell Biology*, 13(9):1011–1011, September 2011. ISSN 1476-4679. doi: 10.1038/ncb2325. URL https://doi.org/10.1038/ncb2325.

[106] Bo Huang, Wenqin Wang, Mark Bates, and Xiaowei Zhuang. 3D super-res imaging by STORM. *Science*, 319(5864):810–813, 2008. doi: 10.1126/science.1153529. Three-dimensional. URL http://www.sciencemag.org/content/306/5703/1895.short.

[107] Ryan McGorty, Daichi Kamiyama, and Bo Huang. Active microscope stabilization in three dimensions using image correlation. *Optical Nanoscopy*, 2(1):3–3, 2013.

ISSN 6176321972. doi: 10.1186/2192-2853-2-3. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3624763/pdf/nihms412728.pdf.

[108] Robert P.J. Nieuwenhuizen, Keith A. Lidke, Mark Bates, Daniela Leyton Puig, David Grünwald, Sjoerd Stallinga, and Bernd Rieger. Measuring image resolution in optical nanoscopy. *Nature Methods*, 10(6):557–562, 2013. ISSN 1548-7105 (Electronic)\r1548-7091 (Linking). doi: 10.1038/nmeth.2448.

[109] Shantanu Singh, Firdaus Janoos, Thierry Pécot, Enrico Caserta, Kun Huang, Jens Rittscher, Gustavo Leone, and Raghu Machiraju. Non-parametric Population Analysis of Cellular Phenotypes. *Med Image Comput Comput Assist Interv.*, 14(2):343–351, 2011.

[110] Maria A. Kiskowski, John F. Hancock, and Anne K. Kenworthy. On the use of Ripley's K-function and its derivatives to analyze domain size. *Biophysical Journal*, 97(4):1095–1103, 2009. doi: 10.1016/j.bpj.2009.05.039. URL http://dx.doi.org/10.1016/j.bpj.2009.05.039. Publisher: Biophysical Society.

[111] Alistair P. Curd, Joanna Leng, Ruth E. Hughes, Alexa J. Cleasby, Brendan Rogers, Chi H. Trinh, Michelle A. Baird, Yasuharu Takagi, Christian Tiede, Christian Sieben, Suliana Manley, Thomas Schlichthaerle, Ralf Jungmann, Jonas Ries, Hari Shroff, and Michelle Peckham. Nanoscale pattern extraction from relative positions of sparse 3d localizations. *Nano Letters*, 21(3):1213–1220, 2021. doi: 10.1021/acs.nanolett.0c03332. URL https://doi.org/10.1021/acs.nanolett.0c03332. PMID: 33253583.

[112] Ralf Jungmann, Maier S Avendaño, Mingjie Dai, Johannes B Woehrstein, Sarit S Agasti, Zachary Feiger, Avital Rodal, and Peng Yin. Quantitative super-resolution imaging with qPAINT. *Nature Methods*, 13(5):439–442, May 2016. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.3804. URL http://www.nature.com/articles/nmeth.3804.

[113] Florian Levet, Eric Hosy, Adel Kechkar, Corey Butler, Anne Beghin, Daniel Choquet, and Jean Baptiste Sibarita. SR-Tesseler: A method to segment and quantify localization-based super-resolution microscopy data. *Nature Methods*, 12(11): 1065–1071, 2015. doi: 10.1038/nmeth.3579.

[114] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996. ISSN 1-57735-004-9.

[115] Botsch Steinberg Bischoff, M Botsch, S Steinberg, S Bischoff, L Kobbelt, and Rwth Aachen. Openmesh–a generic and efficient polygon mesh data structure. In *In OpenSG Symposium*, 2002.

[116] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[117] Charles Loop. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, University of Utah, January 1987. URL https://www.microsoft.com/en-us/research/publication/smooth-subdivision-surfaces-based-on-triangles/.

[118] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, nov 1978. ISSN 00104485. doi: 10.1016/0010-4485(78)90110-0.

[119] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*, pages 209–216, 1997. doi: 10.1145/258734.258849.

[120] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages 105–114, Not Known, 1998. ACM Press. ISBN 978-0-89791-999-9. doi: 10.1145/280814.280831. URL http://portal.acm.org/citation.cfm?doid=280814.280831.

[121] Adrian Secord. Weighted voronoi stippling. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '02, page 37–43, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134940. doi: 10.1145/508530.508537. URL https://doi.org/10.1145/508530.508537.

[122] A. Gueziec, G. Taubin, F. Lazarus, and B. Hom. Cutting and stitching: converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and*

Computer Graphics, 7(2):136–151, June 2001. ISSN 10772626. doi: 10.1109/2945.928166. URL http://ieeexplore.ieee.org/document/928166/.

[123] Peter Liepa. Filling Holes in Meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 200–205, Goslar, DEU, 2003. Eurographics Association. ISBN 1-58113-687-0. event-place: Aachen, Germany.

[124] Gaston Darboux. *Lecons Sur La Theorie Generale Des Surfaces*. Gauthier Villars Et Fils, Paris, 1890.

[125] G. Gompper and D. M. Kroll. Random Surface Discretizations and the Renormalization of the Bending Rigidity. *Journal de Physique I*, 6(10):1305–1320, October 1996. ISSN 1155-4304, 1286-4862. doi: 10.1051/jp1:1996246. URL http://www.edpsciences.org/10.1051/jp1:1996246.

[126] Olga Sorkine. Laplacian Mesh Processing. In Yiorgos Chrysanthou and Marcus Magnor, editors, *Eurographics 2005 - State of the Art Reports*. The Eurographics Association, 2005. doi: 10.2312/egst.20051044.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[129] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.