STUDYING TRANSMEMBRANE PROTEIN TRANSPORT IN PRIMARY CILIA WITH SINGLE MOLECULE TRACKING

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfilment
of the Requirements for the Degree
DOCTOR OF PHILOSOPHY

---

by
Andrew Ruba
December 2019

Examining Committee Members:

Dr. Weidong Yang, Thesis Advisor, Department of Biology
Dr. Joel Sheffield, Advisory Committee Chair, Department of Biology
Dr. Raymond Habas, Department of Biology
Dr. Bojana Gligorijevic, Bioengineering Department

ABSTRACT

The primary cilium is an immotile, microtubule-based protrusion on the surface of many eukaryotic cells and contains a unique complement of proteins that function critically in cell motility and signaling. Critically, the transport of membrane and cytosolic proteins into the primary cilium is essential for its role in cellular signaling. Since cilia are incapable of synthesizing their own protein, nearly 200 unique ciliary proteins need to be trafficked between the cytosol and primary cilia. However, it is still a technical challenge to map three-dimensional (3D) locations of transport pathways for these proteins in live primary cilia due to the limitations of currently existing techniques. To conquer the challenge, this work employed a high-speed virtual 3D super-resolution microscopy, termed single-point edge-excitation sub-diffraction (SPEED) microscopy, to determine the 3D spatial location of transport pathways for both cytosolic and membrane proteins in primary cilia of live cells.

Using SPEED microscopy and single molecule tracking, we mapped the movement of membrane and soluble proteins at the base of the primary cilium. In addition to the well-known intraflagellar transport (IFT) route, we identified two new pathways within the lumen of the primary cilium - passive diffusional and vesicle transport routes - that are adopted by proteins for cytoplasmic-cilium transport in live cells. Independent of the IFT path, approximately half of IFT motors (KIF3A) and cargo (α-tubulin) take the passive diffusion route and more than half of membrane-embedded G protein coupled receptors (SSTR3 and HTR6) use RAB8A-regulated vesicles to transport into and inside cilia. Furthermore, ciliary lumen transport is the preferred route for

membrane proteins in the early stages of ciliogenesis and inhibition of SSTR3 vesicle transport completely blocks ciliogenesis. Furthermore, clathrin-mediated, signal-dependent internalization of SSTR3 also occurs through the ciliary lumen. These transport routes were also observed in *Chlamydomonas reinhardtii* flagella, suggesting their conserved roles in trafficking of ciliary proteins.

While the 3D transport pathways in this work are always replicated multiple times with a high degree of consistency, several experimental parameters directly affect the 3D transport routes' error, such as single molecule localization precision and the number of single molecule localizations. In fact, if these experimental parameters do not meet a minimum threshold, the resultant 3D transport pathways may not have significant resolution to determine any biological details. To estimate the 3D transport routes' error, this work will explain in detail the component of SPEED microscopy that estimates 3D sub-diffraction-limited structural or dynamic information in rotationally symmetric bio-structures, such as the primary cilium. This component is a post-localization analysis that transforms 2D super-resolution images or 2D single-molecule localization distributions into their corresponding 3D spatial probability distributions based on prior known structural knowledge. This analysis is ideal in cases where the ultrastructure of a cellular structure is known but the sub-structural localization of a particular protein is not. This work will demonstrate how the 2D-to-3D component of SPEED microscopy can be successfully applied to achieve 3D structural and functional sub-diffraction-limited information for 25-300 nm subcellular organelles that meet the rotational symmetry requirement, such as the primary cilium and microtubules. Furthermore, this work will provide comprehensive analyses of this method by using computational simulations which investigate the role of various experimental parameters on the 3D transport pathway error. Lastly, this work will demonstrate that this method can distinguish different types of 3D transport pathway distributions in addition to their locations.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ACRONYMS

2D	Two dimensional
3D	Three dimensional
AA	Amino acid
AF	Alexa Fluor
AP	Acceptor peptide
BirA	Biotin ligase
CCD	Charge coupled device
COPI	Coat protein I
COPII	Coat protein II
CTS	Ciliary targeting sequence
DMEM	Dulbecco's Modified Eagle Medium
DNA	Deoxyribonucleic acid
EM	Electron microscopy
GCA	Golgicide A
GDP	Guanosine diphosphate
GFP	Green fluorescent protein
GNC	Glass nano-capillary
GPCR	G-protein coupled receptor
GTP	Guanosine triphosphate
HEPES	(4-(2-hydroxyethyl)-1-piperazineethanesulfonic acid
HTR6	5-hydroxytryptamine receptor 6
IFT	Intraflagellar transport

JF	Janelia Fluor
mNG	mNeon green
mRNA	messenger ribonucleic acid
MSD	Mean squared displacement
NPC	Nuclear pore complex
NPHP4	Nephrocystin-4
PALM	Photoactivated localization microscopy
PKD2	Polycystin-2
PSF	Point spread function
R_I	Ideal radius
R_M	Mean radius
RNA	Ribonucleic acid
SAR	Sum of the absolute valued residuals
SPEED	Single point, edge excitation, sub-diffraction
SSTR3	Somatostatin receptor 3
STED	Stimulated emission depletion
STORM	Stochastic optical reconstruction microscopy
TM	Transmembrane
TPR	Tetratricopeptide repeat
TZ	Transition zone
WI	Gaussian width
σ_{LE}	Single molecule localization error

CHAPTER 1: INTRODUCTION

1.1 General overview of primary cilia

The primary cilium is an immotile, antenna-like projection on nearly all mammalian cells and is involved in a variety of signal transduction pathways ranging from planar cell polarity, Hedgehog signaling, and neuronal signaling to nutrient sensing, mechanosensation, olfaction, phototransduction, and cellular growth (Marshall and Nonaka, 2006, Nauli et al., 2003, Ross et al., 2005, Jones et al., 2008, Rohatgi et al., 2007, Corbit et al., 2005, Breunig et al., 2008, Gerdes and Katsanis, 2005, Scholey and Anderson, 2006, Craft et al., 2015, Wang and Dynlacht, 2018, Wheway et al., 2018, Anvarian et al., 2019, Nachury and Mick, 2019). Central to all these processes is a unique composition of structural, soluble, and transmembrane (TM) proteins in primary cilia (Ostrowski et al., 2002, Pazour et al., 2005, Ishikawa et al., 2012, Breslow et al., 2018). Structural proteins, like tubulins, form the backbone of primary cilia and facilitate mechanosensation (Singla and Reiter, 2006, Shida et al., 2010, Wloga et al., 2017), while TM proteins typically act as signal receptors (Bangs and Anderson, 2017, Hilgendorf et al., 2016, He et al, 2014, Singh et al., 2015, Barzi et al., 2011, Lerea et al., 1986, Anholt et al., 1987, Pazour et al., 2002, Yoder et al, 2002) and soluble proteins act as intracellular messengers (Nair et al., 2005, Rosenzweig et al., 2007, Jensen and Leroux, 2017, Brooks et al., 2018). Defects in ciliary structure and function can lead to a broad class of pathologies termed “ciliopathies”, which include Bardet-Biedl syndrome, Joubert syndrome, polycystic kidney disease, nephronophthisis, and retinal degeneration (Hildebrandt et al., 2011).

1.2 Primary cilia ultrastructure

From an ultrastructural perspective, primary cilia are composed of three main sub-regions. First, the basal body, which contains the 9 triplet microtubules of the mother centriole and their associated distal and subdistal appendages. Distal to the basal body is the transition zone (TZ, 300-1000 nm in length and 160-250 nm in diameter) where the 9 microtubule triplets transition to 9 microtubule doublets (termed 9+0) and is the proposed location for the components of the selectivity barrier (Kee et al., 2012, Yang et al., 2015, Czarnecki and Shah, 2012, Craige et al., 2010, Najafi et al., 2012). The TZ contains Y-shaped structures in electron micrographs that appear to tether the microtubules to the encompassing ciliary membrane. Third, the main ciliary shaft, composed of the 9+0 microtubule axoneme, is 200-250 nm in diameter and extends into the extracellular environment (Czarnecki and Shah, 2012), where it receives and transmits signals via TM receptors (Ye et al., 2018, Hilgendorf et al., 2016) and ectosomal release (Wang et al., 2014, Phua et al., 2017, Nager et al., 2017), respectively. Encompassing both the TZ and the ciliary shaft is a membrane with a unique lipid and protein composition (Nauli et al., 2003, Yang et al., 2015, Praetorius, 2014, Molla-Herman et al., 2010, Praetorius and Spring, 2003, Kaneshiro, 1987). While the primary cilium axoneme is generally modelled as a rigid 9+0 structure, various asymmetries may occur, such as a microtubule doublet turning into a singlet and/or collapsing toward the inner lumen, especially toward the distal end (Rogowski et al., 2013, O'Hagan et al., 2017, Sun et al., 2019).

The construction of primary cilia is called "ciliogenesis" and typically occurs during the G1 or G0 phase of the cell cycle. Oftentimes, primary cilia are then deconstructed just before mitosis. Ciliogenesis is initiated when a basal body (ring of 9 microtubule triplets) develops from an existing centriole in the cell or develops *de novo*. Subsequently, the basal body becomes associated with a membrane vesicle. This vesicle will form the ciliary membrane once the basal body/vesicle complex is transported to the actin-rich region of the cell where the primary cilium will eventually reside and buds into the extracellular matrix (Ishikawa and Wallace, 2011). Following this budding process, tubulin subunits show an increased level of synthesis and are transported to the primary cilium where they are incorporated into growing end of the axoneme which projects from the basal body (Wood and Rosenbaum, 2014). As primary cilia grow, it is hypothesized that calcium mechanosensors in the ciliary membrane experience variable levels of stretch based on the length and flexibility of the growing cilium. This provides negative feedback control affecting the level of intraciliary tubulin transport (Ishikawa and Wallace, 2011). Ultimately, primary cilia tend to grow to ~3-5 μm in length in mammalian cells.

1.3 Processes responsible for intraciliary transmembrane protein transport

Since primary cilia are dynamic signaling organelles, the efficient transport of transmembrane proteins is essential for their proper construction and function. Broadly, there are two types of transport mechanisms for intraciliary proteins. 1) Motor protein driven transport, termed intraflagellar transport (IFT), relies on the movement of kinesin (anterograde motor) and dynein (retrograde motor) along the microtubules of the ciliary axoneme to move cargo throughout the primary cilium. 2) Passive diffusion of several types of molecules occurs throughout the matrix and on the membrane of the primary cilium (Ishikawa and Wallace, 2011, Nachury et al., 2010). The following sections will provide an overview of the important components of these two transport processes.

1.3.1 Overview of intraflagellar transport

One transport mode for TM proteins is the directional, motor-driven, evolutionarily-conserved mechanism of intraflagellar transport (Ye et al., 2013, Qin et al., 2005). IFT proceeds bidirectionally along the ciliary shaft by molecular motors that "walk" along the axoneme. Kinesin motors are responsible for anterograde movement and dyneins for retrograde movement. Noncovalently bound to these molecular motors is an A/B complex of 15 proteins called the IFT complex that bridges the space between the axoneme and ciliary membrane to confer directional movement to ciliary TM proteins (Scholey and Anderson, 2006, Taschner et al., 2012). Mutations in the A subcomplex typically result in TM protein mislocalization while B subcomplex mutations usually prevent ciliary formation altogether (Eguether et al., 2014). Foundational transmission electron microscopy studies in *Chlamydomonas reinhardtii* have shown that this process occurs as "trains" of motors, IFT complex proteins, and cargo in the space between the axonemal microtubules and the ciliary membrane and proceed along the primary cilium at a rate of approximately 0.1-1 μm/s. These trains seem to have variable structure with lengths from 0.2-0.65 μm and a periodicity of 25-40 nm (Prevo et al., 2017). More recent studies using techniques such as single molecule tracking have shown that IFT trains assemble in a coordinated fashion near the base of primary cilia and flagella (Wingfield et al., 2017). Furthermore, since the microtubules terminate at the end of primary cilia, IFT trains dissociate from their kinesin motor following anterograde transport and associate with a dynein motor for retrograde transport in order to turn around (Wei et al., 2012).

1.3.2 Overview of the BBSome

Another important component of the intraciliary movement of transmembrane proteins is a ~500 kDa complex of eight proteins called the BBSome. It is necessary for proper TM protein localization and transports with the IFT complex (Jin et al., 2010, Berbari et al., 2008). In *C. elegans*, the BBSome is present in stoichiometric concentrations to the IFT complex and BBSome mutations decouple the directional velocities of both IFT subcomplexes suggesting that the BBSome bridges and stabilizes the IFT complex (Ou et al., 2005). However in *C. reinhardtii*, the BBSome is present in substoichiometric concentrations and a mutagenesis screen revealed a mutation that prevents the BBSome from binding to the IFT complex. This mutation did not decouple the directional velocities of the IFT subcomplexes but rather caused the accumulation of IFT subcomplex B in the tip and the BBSome in the base. It was concluded that the BBSome regulates the formation of the IFT complex but does not stabilize it during transit as previously described (Wei et al., 2012). In mammalian systems, the BBSome has been shown to directly interact with relatively conserved sequences on TM proteins called ciliary targeting sequences (CTS) (Jin et al., 2010, Berbari et al., 2008). Most integrative models generally suggest that the BBSome specifically recognizes TM proteins and tethers them to the IFT complex which is, in turn, attached to the molecular motors that drive the whole supercomplex called an IFT train.

1.3.3 Overview of passive diffusion

In addition to IFT, passive diffusion of many molecules occurs throughout the primary cilium. Tubulin, for example, enter primary cilia via IFT or passive diffusion en route to its integration onto the growing ciliary microtubules (Harris et al., 2018). On the other hand, the primary cilium has many G-protein coupled receptors (GPCR) on its surface which are integral to the primary cilium's role as a signaling organelle and explore the ciliary membrane space via thermodynamic diffusion (Ye et al., 2013, Hilgendorf et al., 2016). Part of the process of GPCR signaling involves the external binding of a specific ligand to the GPCR and internal release of diffusive G proteins which have various intracellular signaling functions. Previously in our lab, we found that the central lumen of the axoneme is a prominent transport route for several diffusive proteins: about half of the mobile α-tubulin and kinesin in primary cilia as well as free GFP (Luo et al., 2017). Others have found that passive diffusion accounts for the majority of transmembrane protein movement in primary cilia (Ye et al., 2013).

1.4 The transition zone of primary cilia as a selective gate

For primary cilia to properly function, it is essential to maintain the proper complement of ciliary proteins compartmentalized to the ciliary space. Previous work has shown that a diffusion barrier at the base of the primary cilium restricts the entry and exit of soluble and TM proteins. Many studies suggest that the cutoff ranges for this barrier of 40-100 kDa (Kee et al., 2012, Breslow et al., 2013, Calvert et al., 2010, Lin et al., 2013). Currently, it is not entirely clear what constitutes this barrier. The knockout of certain structural proteins in the transition zone can cause the mislocalization of non-ciliary proteins (Williams et al., 2011). Interestingly, nucleoporins, proteins that control the size exclusion barrier at the nuclear pore complex (Rout et al., 2003), have been found at the base of primary cilia (Kee et al., 2012). As previously mentioned, both active transport by microtubule-based motors and free diffusion can drive the entry of structural/soluble proteins into *Chlamydomonas reinhardtii* flagella (Hao et al., 2011, Craft et al., 2015). For TM proteins, two models have been put forth for their cytoplasm-cilium transport and entry into primary cilia: 1) vesicle fusion outside the primary cilium and subsequent transport of TM proteins into primary cilia after loading onto an intraflagellar transport (IFT) train; or 2) vesicle fusion at an unknown location inside primary cilia (Figure 1 A) (Jensen and Leroux, 2017, Chuang et al., 2015, Nachury et al., 2010, Hunnicutt et al., 1990, Pazour and Bloodgood, 2008, Vieira et al., 2006). While the precise site of extraciliary fusion is unclear and may vary between cell types, the first model is largely derived from early findings in frog photoreceptors which show that rhodopsin localizes to vesicles that appear to be fusing at the periciliary ridge complex, a unique membrane domain near the base of the photoreceptors' connecting cilium (Papermaster et al., 1985). The second model, which appears to accomplish the transport and enrichment of TM proteins in primary cilia in one step, arises from the occasional appearance of vesicle-like structures inside primary cilia and photoreceptors (Chuang et al., 2015, Reese, 1965, Poole et al., 1985, Jensen et al., 2004, Jana et al., 2018). The role of IFT in the import of TM proteins is less clear in this model. A common thread through all models is the necessity of certain factors that facilitate the proper fusion of vesicles and entry of TM proteins to the primary cilium. The small GTPase Rab8a is the representative of the Rab family of proteins that associates stably with vesicles targeted to cilia and promotes fusion (Grigoriev et al., 2011). Knockout prevents ciliary formation (Yoshimura et al., 2007) while overexpression of a GDP-locked, inactive variant causes the accumulation of vesicles in the cell body at the base primary cilia indicating, upon GTP loading, it promotes the fusion of vesicles and movement of its cargo into primary cilia (Nachury et al., 2007).

While much has been discovered, TM protein trafficking into primary cilia is an exciting field of research because of the variety of unanswered questions. While the prevailing models suggest TM protein transport vesicles fuse at the base of primary cilia, why do vesicle proteins, such as Rab8a, concentrate in primary cilia (Nachury et al., 2007)? Also, BBSome proteins have many WD40 and TPR repeats and appear to share evolutionary origins with traditional exomer, COPI, and COPII coat proteins but did not sculpt membranes into buds in vitro like traditional coat proteins do (Jin et al., 2010). The conditions may be different for the BBSome to form a vesicular coat but questions remain as to why the BBSome recognizes a TM protein's CTS, where this process takes place, and how the cargo/BBSome/IFT/motor supercomplex makes it through the crowded environment of the transition zone. Is there remodeling of the crowded transition zone or is there some specific recognition that allows passage of the IFT train? To answer at least some of these questions and ultimately further develop the model for ciliary TM protein transport, a super resolution microscopy-based approach will be explored.

1.5 Super-resolution microscopy to sub-diffraction-sized biological structures

Prior to super-resolution microscopy, fluorescently-labeled protein localization using epifluorescence or confocal microscopy was a staple of molecular and cellular biology research to study protein localization and dynamics in live cells. Both techniques rely on the wide-field illumination and detection of many fluorophores simultaneously to localize the labeled protein. Consequently, the emissions from all the illuminated fluorophores are summed to produce the whole cell localization of the protein. However, fluorophores that are spaced less than the diffraction limit (~200 nm) from each other are indistinguishable. Since stated by Ernst Abbe in 1873, the resolution of conventional light microscopy has been believed to be limited to approximately 200 nm due to light diffraction from the objective (Huang et al., 2009, Leung and Chou, 2011). Currently, super-resolution techniques break this limitation and allow the capture of dynamic images with sub-diffraction (<200 nm) resolution. The techniques generally fall into two broad categories: stimulated emission depletion (STED) microscopy based approaches, which generate sub-diffraction illumination volume due to nonlinear optical response of fluorophores in samples; and photoactivated light microscopy (PALM) and stochastic optical reconstruction microscopy (STORM)-based super-resolution techniques, which utilize mathematical functions to localize the centroids of fluorophores and then reconstitute these centroids to form super-resolution images (Huang et al., 2009, Leung and Chou, 2011, Hell and Wichmann, 1994, Betzig et al., 2006). Currently, due to the relatively uncomplicated optical setup, PALM and STORM are extensively employed by only activating a small subset of fluorophores in each frame of a long video of a biological preparation. This allows for a more accurate localization by 2D Gaussian fitting of the fluorescent spot, termed the point spread function or PSF, of fluorescently-labeled proteins in each frame of the video. The 2D location of each fluorescently-labeled molecule can then be superimposed on a single imaging plane to produce a super-resolution image of the biological preparation Huang et al., 2009, Leung and Chou, 2011). While these single molecule localization, super-resolution approaches to microscopy certainly revolutionized how imaging of biological samples was performed, there are still challenges to be overcome. For example, STORM and PALM can achieve their best spatial resolutions after fixation of biological samples and thus present a static representation of the fluorescently-labeled proteins, which is a similar limitation of electron microscopy. Additionally, to achieve high spatial resolution for each fluorescently-labeled protein in live cells, samples must be imaged at very long framerates for extended periods of time which are unable to capture protein dynamics.

1.6 Three dimensional super-resolution imaging

Fast, three-dimensional (3D) super-resolution imaging is critical for obtaining structural or dynamic information in live cells, which are inherently 3D objects. Moreover, many biological functions in bacteria (Lew et al., 2011, Lee et al., 2014, Oswald et al., 2016) and sub-cellular organelles are near or below the spatiotemporal resolution limit of current 3D super-resolution imaging techniques, such as nucleocytoplasmic transport through 50-nm nuclear pore channels or protein transport in the primary cilium, with millisecond transport times (Akey, 1989, Akey and Radermacher, 1993, Yang and Musser, 2006, Luo et al., 2017). Therefore, it was necessary to overcome these main technical hurdles.

Typically, 3D super-resolution imaging is more technically demanding than 2D super-resolution imaging. This is due to the fact that the point spread function (PSF) of the emitting fluorescent probe in the axial dimension is much larger than in the lateral dimension at the focal plane of the light microscopy objective (Huang et al., 2009). Several methods have been developed to improve axial resolution in fluorescence microscopy. One category is to alter the shape of the PSF of the fluorescent probe along the optical axial position and then determine the probe's axial information by referencing a predetermined relationship, established through control experiments, between the shape of the PSF and the corresponding locations in the z dimension (Pavani et al., 2009, Franke et al., 2017, Huang et al., 2008, Shechtman et al., 2014, Prabhat et al., 2004). The other is to use two objectives to improve the axial resolution after comparing fluorescent signals of the probes from these objectives with or without interference (Böhm et al., 2016). Typically, the above approaches involve expensive and complex optical implements.

1.7 SPEED microscopy

To obtain a high spatiotemporal resolution that is well-suited for detecting fast-moving proteins or RNAs in the live, 3D environments of cells, we have developed super-resolution SPEED microscopy in our laboratory (Figure 1 B) (Ma and Yang, 2010, Ma et al., 2012, Ma et al., 2013). Several major technical advances in SPEED microscopy have previously enabled us to successfully track nucleocytoplasmic transport of small molecules, proteins, mRNA and virus through native NPCs (Ma and Yang, 2010, Ma et al., 2012, Ma et al., 2013). Briefly, the following features of SPEED microscopy will be used to track fast-moving macromolecules through sub-micrometer rotationally symmetrical structures in live cells, such as nuclear pore complexes (NPCs) and primary cilia: (1) A vertical or an inclined illumination PSF enables the excitation of single molecules within a small diffraction-limit volume in the focal plane (Figure 1 B); (2) The high optical density (100–500 kW/cm^2) in the small illumination volume allows a high number of photons from the fluorophores be collected in a short time period. Typically, more than a thousand photons were obtained from a single GFP-labeled protein within a 2-ms detection time. If needed, an optical chopper was used to create an on/off operational mode with an off-time that is ten-fold longer than the on-time, which aims at reducing photobleaching and phototoxic effects; (3) The inclined illumination volume further avoids out-of-focus background fluorescence if necessary and auto-fluorescence of the objective, which enhances the signal to noise ratio for single-molecule localization. (4) The small illumination volume of SPEED microscopy allows the imaging of single molecules, as opposed to wide field fluorescence imaging which illuminates and images many fluorescent molecules at once, within a small pixel area of the CCD camera, resulting in a fast detection speed (up to 5000 frames per second). (5) The fast detection speed greatly reduces the single-molecule spatial localization error in determining the spatial trajectories of moving fluorescent molecules in live cells, enabling us to obtain a spatial localization resolution of <10 nm for moving protein. (6) In our lab, a well-established 2D to 3D conversion algorithms enables us to provide 3D probability density maps of transport routes for molecules in the NPC or the primary cilium. It is noteworthy that our conversion process between the Cartesian and the cylindrical coordination system is used to generate a virtual 3D probability density map rather than 3D single-molecule tracking. Previously, electron microscopy data have revealed that the NPC (Akey, 1989, Akey and Radermacher, 1993) and the primary cilium (Czarnecki and Shah, 2012) both have a rotationally symmetrical structure. In principle, randomly diffusing molecules moving through the NPC or primary cilium should also have spatially symmetrical distributions. As described in previous publications (Ma and Yang, 2012, Ruba et al., 2019), a high-number of randomly diffusing molecules inside the cylinder would generate rotationally symmetrical distributions at the cross-section view as that in the NPC, further resulting in an approximately uniform spatial distribution within each very small sub-region between two neighboring rings. This uniform distribution leads that the spatial distribution along ϴ dimension in the cylindrical system is constant. Then the 3D coordinates (R, X, ϴ) can be simplified to be the 2D coordinates (R, X, constant). Actually, our conversion process between the Cartesian and the cylindrical systems is from 2D (X, Y) to 2D (R, X, constant). The constant ϴ, refers to the spatial density **ρ** (Figure 2).

Ultimately, single molecule tracking has broad application in biological research and, thus, it is natural that a plethora of techniques will be developed to fill specific biological niches (Elf et al., 2010, Anzalone et al., 2014, Ritter et al., 2010). Such is the case with SPEED microscopy. Previously, when coupled with a 3D transformation algorithm, this technique was developed to resolve 3D transport routes of protein cargo through the NPCs, a sub-diffraction-sized and rotationally symmetric biological structure (Ma and Yang, 2010). In this thesis, primary cilia are shown to be excellent model organelles as well since they are cylindrical, antenna-like organelles (~125 nm radius) that project from the surface of most mammalian cells (Marshall and Nonaka, 2006, Yang et al., 2015, Scholey and Anderson 2006). As previously mentioned, they are responsible for receiving external signals and transmitting an intracellular response typically associated with growth and metabolism (Marshall and Nonaka, 2006, Scholey and Anderson 2006). Therefore, flux of structural proteins, recycling of transmembrane receptors, and transmission of intracellular messengers are vital responsibilities of primary cilia. Both the sub-diffraction limit size and the complex process of selective protein transport through the TZ make the primary cilium a prime target for this technique. In this thesis, we will demonstrate the tracking of a variety of ciliary proteins and focus on the transport routes and mechanisms of the transport of ciliary transmembrane proteins.

1.8 Specific aims of these studies

1. Determining transmembrane protein transport routes in the transition zone of primary cilia

    a. Empirically validate SPEED microscopy and the 2D-to-3D transformation algorithm for use in primary cilia

    b. Determine transport route for transmembrane proteins in primary cilia

    c. Probe relationship between transmembrane protein and vesicle transport routes

    d. Determine role of transmembrane protein transport routes during active receptor signaling

    e. Elucidate role of transmembrane protein transport routes during ciliogenesis

    f. Determine similarities between transport routes in both primary cilia and flagella

2. Simulating the localization error on transport routes determined by SPEED microscopy and the 2D-to-3D transformation algorithm

    a. Develop simulation method to model the process of single molecule localization collection and 2D-to-3D transformation

    b. Determine role of single molecule localization precision in transport route error

    c. Determine how number of single molecule localizations affects transport route error

    d. Determine effect of single molecule distribution type on transport route categorization

    e. Estimate role of symmetry compression and rotation on transport route error

    f. Determine how labeling ratio affects transport route error

CHAPTER 2: MATERIALS AND METHODS

2.1 Tissue culture and transfection

NIH-3T3 cells or NIH-3T3 cells stably expressing NPHP4-mCherry/IFT20-GFP were grown in DMEM, high glucose, GlutaMAX Supplement (Life Technologies), 10% fetal bovine serum (Fisher Scientific), and 1% penicillin-streptomycin (Thermo Fisher) and split every 2 days to 40-50% confluency. 24 hours prior to imaging, the cells were transferred to glass bottom dishes (MatTek) and grown in OPTIMEM (Life Technologies) to induce growth of primary cilia. Transfection was performed concurrently with induction of primary cilia growth using Transit-LT1 (Mirrus) according to the manufacturer's protocol. Prior to imaging, media was replaced with transport buffer (20 mM HEPES, 110 mM KOAc, 5 mM NaOAc, 2 mM MgOAc, and 1 mM EGTA, pH 7.3). For permeabilization, cells were permeabilized in glass-bottom dishes for 2 min with 30 μg/mL digitonin in transport buffer and washed again with transport buffer. Transport buffer was supplemented with 1.5% polyvinylpyrrolidone. For external membrane labeling, biotin ligase (BirA) and AP-SSTR3-GFP were co-transfected using Transit-LT1 according to the manufacturer's protocol except for the media, which was supplemented with 1 μM biotin. Prior to imaging, cells were incubated with 1 μM AlexaFluor647-streptavidin (Life Technologies) for 30 minutes to cause efficient binding to transiently expressed, biotinylated AP-SSTR3-GFP on the cell surface. Cells were then washed five times with PBS to efficiently wash away unbound AlexaFluor647-streptavidin before placing cells in transport buffer. For Golgicide A inhibition, cells were serum-starved as described above and the media was supplemented with the given concentration of Golgicide A. For somatostatin stimulation, cells were incubated with 10 μM somatostatin for 1 hour prior to imaging. For Pitstop 2 inhibition of somatostatin signaling, cells were incubated with the given concentration of Pitstop 2 for 1 hour prior to addition of somatostatin. For the growth status experiments, cells were serum-starved as described above and imaging was performed at the given time interval post-serum-starvation.

2.2 Plasmids and stable cell lines

The SSTR3-GFP plasmid was a gift from Kirk Mykytyn (The Ohio State University, College of Medicine). The GFP-IFT43 plasmid was a gift from Kazuhisa Nakayama (Kyoto University, Department of Physiological Chemistry). The IFT20-GFP plasmid was a gift from Gregory Pazour (University of Massachusetts Medical School). Arl13b-mCherry, KIF3A-GFP, SSTR3-mCherry, the IFT20-GFP stable cell line, and the NPHP4-mCherry stable cell line was a gift from Kristen Verhey (University of Michigan). AP-SSTR3-GFP was a gift from Maxence Nachury (Addgene plasmid # 49098), GFP-RAB8A was a gift from Maxence Nachury (Addgene plasmid # 24898), BirA was a gift from Alice Ting (Addgene plasmid #20856), α-tubulin-GFP was a gift from Patricia Wadsworth (Addgene plasmid #12298), and mCherry-Gamma-Tubulin-17 was a gift from Michael Davidson (Addgene plasmid # 55050). The *Chlamydomonas reinhardtii* β-tubulin-GFP and mNG-IFT54 cell lines were a gift from Karl F. Lechtreck (University of Georgia). The *Chlamydomonas reinhardtii* PKD2-GFP cell line was a gift from Kaiyao Huang (Chinese Academy of Sciences). The *Chlamydomonas reinhardtii* KAP-GFP cell line was a gift from Mary Porter (University of Minnesota) through the Chlamydomonas Resource Center (CC-4296).

2.3 Optical setup of SPEED microscopy

The SPEED microscopy setup includes an Olympus IX81 equipped with a 1.4-NA 100× oil-immersion apochromatic objective (UPLSAPO 100×, Olympus), a 35 mW 633 nm He-Ne laser (Melles Griot), 50 mW solid state 488-nm and 561-nm lasers (Coherent), an on-chip multiplication gain charge-coupled-device camera (Cascade 128+, Roper Scientific) and the Slidebook software package (Intelligent Imaging Innovations) for data acquisition and processing. For individual channel imaging, GFP, mCherry, and Alexa Fluor 647 were excited by 488 nm, 561 nm, and 633 nm lasers, respectively. The fluorescence emissions were collected by the same objective, filtered by a dichroic filter (Di01- R405/488/561/635-25x36, Semrock) and an emission filter (NF01-405/488/561/635-25x5.0, Semrock) and imaged with the above CCD camera operating at either 500 Hz when the 2D-to-3D transformation was performed and 100 Hz when SSTR3/RAB8A cotracking and SSTR3-GFP directionality tracking under somatostatin stimulation.

2.4 Spatial localization of primary cilia and the transition zone

To mark the transition zone, the centroid of the fluorescent spot for NPHP4-mCherry was used since NPHP4 localizes on the arms of the Y-shaped linkers, which have 9-fold symmetry and are organized in multiple layers. In situations where the tested fluorescently-labeled proteins did not strongly localize to primary cilia, Arl13b-mCherry was used as a ciliary marker. Since Arl13b-mCherry molecules are mobile in primary cilia, they can be quickly pre-photobleached by focusing a 561-nm laser on the ciliary shaft before localization of the NPHP4-mCherry was determined in the transition zone. Since the transition zone may range up to 1000 nm in length along the long axis of primary cilia, only single molecule data collected at ±300 nm from the NPHP4-mCherry centroid was used in the 2D to 3D transformation to ensure only single-molecules moving through the transition zone was collected. To mark the basal body, the above protocol was used except that γ-tubulin-mCherry, the type of tubulin that constitutes the basal body, was used instead of NPHP4-mCherry.

2.5 Single-molecule localization precision

Single molecule videos were analyzed using Glimpse (Gelles Lab) to determine the XY locations of each single molecule on the imaging plane using 2D Gaussian fitting. The Gaussian width parameter of each single molecule was filtered to ensure only single molecules within the dimensions of the primary cilium were retained for 2D to 3D transformation analysis while the photon count parameter of each single molecule was filtered to ensure adequate precision. The localization precision of immobile fluorescent molecules and moving fluorescent molecules was defined as how precisely the central point of each detected fluorescent diffraction-limited spot was determined. For immobile molecules, the fluorescent spots were fitted to a 2D symmetrical or an elliptical Gaussian function, respectively, and the localization precision was determined by the standard deviation of multiple measurements of the central point. However, for moving molecules, the influence of particle motion during image acquisition should be considered in the determination of localization precision. In detail, the localization precision for moving substrates (σ) was determined by the formula:

$$\sigma = \sqrt{F[\frac{16(s^2 + a^2/12)}{9N} + \frac{8\pi b^2(s^2 + a^2/12)^2}{a^2N^2}]}$$

Where F is equal to 2, N is the photon count, a is the pixel size of the CCD camera, b is the standard deviation of the background in photons per pixel, and s is:

$$s = \sqrt{s_0^2 + \frac{1}{3}D\Delta t}$$

Where s_0 is the standard deviation of Gaussian function of the mobile molecules at the focal plane and D is the diffusion coefficient of the mobile molecule (Mortensen et al., 2010, Quan et al., 2010, Robbins and Hadwen, 2003, Deschout et al., 2012). Additionally, s_0 and N of single transiting fluorescent molecules were used as selective criteria to ensure that only single molecules with high localization precision within the dimensions of the transition zone were selected. For the single-molecule data used in 2D to 3D transformation, the average single-molecule localization precision for every experiment was summarized in Table 1.

2.6 2D-to-3D density transform algorithm

The transformation process used to compute the 3D spatial probability density maps of particles transiting through the TZ as has been described in detail in our previous publications (Ma and Yang, 2010, Ruba et al., 2018, Ruba et al., 2019). In short, the 3D spatial locations of molecules transiting through the NPC can be considered in either Cartesian (x, y, z) or cylindrical (x, r, θ) coordinates. In microscopic imaging, the observed 2D spatial distribution of particle localizations is a projection of its actual 3D spatial locations onto the xy plane. Since primary cilia are symmetrical about their long central axis, the 2D spatial distribution is averaged about that axis. The underlying 3D spatial distributions can then be recovered by projection of the measured Cartesian (x, y) coordinates back onto the simplified cylindrical (x, r, constant) coordinates, on the basis of the expected cylindrically symmetrical distribution along the θ direction of the primary cilium.

2.7 Estimation of spatial probability percent for transiting molecules traveling in each transport route

To calculate the spatial probability of single molecules for a protein with two transport routes, the integrated area of density histogram for each transport route was calculated and expressed as a percentage of the total integrated area of all transport routes to estimate the probability of detecting a single molecule in that given transport route.

2.8 SSTR3 and RAB8A co-movement analysis

To maximize the chance of detecting of any co-movement between SSTR3 and RAB8A above control levels in live cells, we co-transfected plasmids containing

SSTR3-mCherry and GFP-RAB8A into NIH-3T3 cells and used a dual channel filter set to split the individual fluorescence illuminated by simultaneous 561 and 488 nm lasers from each construct onto individual halves of the CCD detector. After alignment of the two halves of the detector, co-moving trajectories in each channel were selected for analysis when at least two consecutive single molecules appeared in each channel, within acceptable Gaussian width (WI) and localization precision bounds, and no further than 200 nm from each other during each step of the trajectory. This 200 nm requirement was implemented to control for two trajectories were further than a biologically reasonable distance. 200 nm is the approximate diameter of the axoneme plus single molecule localization precision and, therefore, the maximum size of a potential vesicle carrier entering primary cilia. When these requirements were implemented, the positive control – dual labeled 100 nm Tetraspeck beads in 55% glycerol to mimic approximate diffusion constant of *in vivo* molecules - co-movements were highly correlated and negative control – fluorescein and JF561 dye in 92.5% glycerol - co-movements where largely uncorrelated. Glycerol was used to adjust the viscosity of the media and, thus, the diffusion coefficient of the above particles using the following equation: $D = \frac{kT}{6\pi\eta R}$ where D is the diffusion coefficient, k is the Boltzmann constant, T is temperature, η is the viscosity, and R is the radius of the particle. The laser powers for the controls were adjusted to produce localization precisions comparable to experimental conditions. A standard curve was developed from these controls and used to characterize the results *in vivo* (Figure 8 M).

2.9 Calculation of diffusion coefficient and α value

Calculation of the diffusion coefficient was performed by first plotting each trajectory (>4 frames) on a Mean Square Displacement (MSD) vs. Time (t) plot. The data was fitted with the function $MSD = 4Dt^{\alpha}$ where α is a measurement of graph skewedness. α represents directional movement (or super-diffusion), passive diffusion or sub-diffusion if its value is bigger than 1.1, between 0.9 and 1.1, or smaller than 0.9 respectively.

2.10 Determination of cilia movement, cilia orientation and axial position

In our experiments, we typically needed two minutes to complete imaging of the entire cilium and collection of ten single-molecule videos (5,000 frames per video and 2 ms per frame) from a primary cilium. Combination of 24-hour serum-starving cell growth and incubation of cells in transport buffer prior to microscopy imaging made the shift of primary cilia in live cells less than 5 nm during the 10 s detection time for each video. As for the cilia orientation, an epi-fluorescence image of Arl13b-mCherry, SSTR3-GFP, HTR6-GFP, or GFP-RAB8A labeled cilium provided a complete image of entire cilium, which clearly indicated the ciliary base, the ciliary tip and the cilia orientation. Then after pre-photobleaching of the tracked protein to locally reduce its concentration, the point-illumination of SPEED microscopy generated 2D super-resolution spatial distribution of protein molecules moving within a range of approximately 1 μm along the ciliary axis in the shaft or TZ of primary cilia. Consequently, the precise location of the middle axis of a primary cilium was obtained by determining the peak position of these 2D super-resolution spatial locations in the ciliary radial dimension with fitting of Gaussian function as well as the Gaussian fitting of the ciliary fluorescence in the image of the ciliary marker.

2.11 Determination of the location of transport route for SSTR3 labeled externally

The length of the linker was determined by summing the estimated length of each component of the external label: SSTR3 N termius:acceptor peptide:biotin:streptavidin:3xAlexaFluor647 (Howarth and Ting, 2008). The SSTR3 N terminus is 43 amino acids long with a marginal level of disorder according to IUPred. To estimate the length, the average (9.6 nm) between the fully disordered length (43 a.a., 0.4 nm/a.a.) (Erickson, 2009), and fully globular diameter (2 nm) was taken (Ainavarapu et al., 2007, Erickson, 2009, Dosztányi et al., 2005). According to IUPred, the AP domain had a consistent structured prediction. Thus, the globular diameter was used (1.5 nm) (Ainavarapu et al., 2007, Dosztányi et al., 2005). Using the average length of a C-C bond, biotin's length was estimated to be ~1 nm. Lastly, the globular diameter of tetravalent streptavidin was estimated to be 5 nm (Erickson, 2009). However, three of the four binding domains, on average, are occupied by biotin conjugated AlexaFluor647. Since the average center of fluorescence will be roughly in the middle of the tetravalent streptavidin molecule, 2.5 nm was used for its length. Thus, the total length of the external tag is ~14.6 nm.

2.12 Determination of spatial localization precision of transport route

Monte Carlo simulations were performed where varying numbers of single molecule locations were randomly simulated on an ideal radius (R_I). Then, localization error (σ_{LE}) was added in the y and z dimensions by sampling an error value from a normal distribution with a standard deviation of σ_{LE}. Subsequently, the 3D transformation algorithm was performed on only the y dimensional data to model the loss of z dimensional information during the 2D microscopy projection process. The peak position of the transformed 3D density histogram was then determined by Gaussian fitting to produce a measured radius (R_M) which may deviate from the R_I due to number of simulated locations and simulated localization error. After 10,000 iterations of this process, 10,000 R_M values are collected and the resulting histogram of R_M values can be obtained. After 10,000 iterations, the mean of the R_M values converges on R_I, the ideal radius from which all the simulations were originally sampled, while the standard deviation of the R_M values varies based largely on the number of simulated locations and σ_{LE}. As shown in Table 1, the localization precision and precision/radius ratio of each transport route in primary cilia is high enough to distinguish it and make biologically relevant conclusions.

In addition, all studies indicate that primary cilia have two states of protein transport: 1) ciliary growth state (ciliogenesis) and 2) maintenance state. The process of ciliary growth takes place over the 24 hours of serum starvation. Thus, the 2 minute imaging time of our approach should be sufficient to distinguish even the smallest transition states of protein transport in primary cilia.

2.13 Statistics

Experimental measurements were reported as mean ± standard error of the mean unless otherwise noted.

2.14 Code availability

The code for the simulations, 2D-to-3D transformation, and sample and experimental data are available at https://github.com/andrewruba/YangLab.

CHAPTER 3: RESULTS

3.1 Mapping the transport routes for ciliary proteins in the transition zone

3.1.1 SPEED microscopy maps the known transit routes of well-studied ciliary proteins at the TZ

We began by validating SPEED microscopy to track protein localization in the TZ of live primary cilia by determining the transport routes for proteins whose localizations are already known. Namely, we looked at IFT components (IFT20 and IFT43), freely diffusing soluble molecules (free GFP), IFT motors and cargos (KIF3A and -tubulin), and an externally-labeled TM protein (SSTR3). We chose the NIH-3T3 cell line because serum starvation causes cell cycle arrest and ciliogenesis in the majority of the cells (Ott and Lippincott-Schwartz, 2012).

Experimentally, we localized the position of the TZ in live primary cilia by detecting the fluorescence of NPHP4-mCherry, a TZ marker that localizes to the Y-shaped linkers (Awata et al., 2014), and then pre-photobleaching the fluorophore-labeled protein-of-interest by SPEED microscopy to locally reduce the labeled concentration of that protein in the TZ (Figure 1 B, Figure 3 A). Then, a high-speed CCD camera set at 500 frames per second (2 ms/frame) was used to record the 2D localizations of single, labeled proteins-of-interest in the TZ after they entered the photobleached area from neighboring regions (Video: https://youtu.be/1JkFeCeeknA). We typically recorded 30,000-60,000 frames within 1-2 minutes and found the spatial shift of primary cilium is < 10 nm during the detection time. Furthermore, thousands of single molecule events with localization precisions of 10-20 nm were selected from these frames. These single-molecule localizations were then superimposed on the NPHP4-mCherry fluorescence spot used to mark the TZ to generate a 2D high-resolution spatial distribution of the protein-of-interest (Figure 3 B,C). Finally, we utilized the 2D-to-3D transformation algorithm to obtain the 3D spatial density histogram of the protein-of-interest along the radius of the primary cilium (Figure 3 D-F) (Ma and Yang, 2010, Ruba et al., 2018a, Ruba et al., 2018b, Ma et al., 2016, Ruba et al., 2019).

For the 3D reconstructed back-projection to validly represent the actual 3D distribution of single molecules, the single molecule locations must be distributed with rotational symmetry. Electron microscopy (EM) studies of the primary cilium validate the rotational symmetry of the ciliary ultrastructure, which provides a scaffolding for protein transport routes to occur (Czarnecki and Shah, 2012). Through Monte Carlo simulation, given typical experimental parameters (single molecule localization precision, number of single molecule locations, imperfect degree of labeling symmetry and rotational symmetry), we are able to obtain < 10 nm standard error on the mean peak position of routes for targeted proteins in primary cilia. This method will be explained in great detail in section 3.3. Overall though, the standard error on the mean peak positions for the protein transport routes in sections 3.1 and 3.2 are summarized in Table 1. The code for the simulations, 2D-to-3D transformation, and sample and experimental data is available at https://github.com/andrewruba/YangLab.

Using SPEED microscopy, we first determined the 3D transport routes of components of the IFT A and B subcomplexes: IFT43 and IFT20, respectively (Follitt et al., 2006, Arts et al., 2011, Hirano et al., 2017). EM work has shown that IFT occurs between the axonemal microtubules and ciliary membrane in *Chlamydomonas reinhardtii* flagella (Rosenbaum and Witman, 2002, Kozminski et al., 1995). Therefore, we expected the 3D density histograms of IFT43 and IFT20 to lie in this space. Indeed, the transport routes for IFT20-GFP and GFP-IFT43 were found at 105±2 and 108±1 nm along the cilium radius with route widths (defined as ±1 standard deviation about the peak position in Gaussian function) of 46±1 nm and 52±2 nm (Figure 1 C-H). Compared to the ciliary dimensions as determined by EM, these IFT components localize between the microtubules and ciliary membrane.

Next, we verified the localizations of IFT motors, IFT cargos, and freely diffusing small molecules. For the IFT motor, we determined the transport route for KIF3A-GFP, a component of the heterotrimeric kinesin-2 (KIF3A/KIF3B/KAP) complex that drives anterograde IFT (Kozminski et al., 1995, Engelke et al., 2019, Pazour and Rosenbaum, 2002). In agreement with our previous work in the cilium shaft (Luo et al., 2017), KIF3A was found to have two transport routes at the TZ: an outer route at 92±7 nm along the cilium radius with a route width of 71±17 nm, and an inner route at 0±3 nm along the cilium radius with a 52±2 nm route width (Figure 1 I-K). The IFT cargo α-tubulin-GFP (Hao et al., 2011, Craft et al., 2015) also occupies two distinct transport routes in the TZ: one between the microtubules and ciliary membrane with a radial peak position at 111±1 nm and a route width of 43±3 nm, and the other inside the axonemal lumen with a radial peak position at 0±1 nm and a route width of 98±3 nm (Figure 1 L-N). That the outer transport routes of both KIF3A and -tubulin localize between the microtubules and ciliary membrane was expected as was the microtubule-proximal localization of the motor as compared to IFT proteins and cargo. We hypothesize that the inner route of both proteins corresponds to a passive diffusion route as we have previously observed in the cilium shaft (Luo et al., 2017).

To map the passive diffusion route in the TZ, we localized the freely diffusing molecule GFP using Arl13b-mCherry as a ciliary marker (Higginbotham et al., 2012). We found that GFP localizes at the center of the TZ with a peak position at 0±1 nm along the cilium radius and a route width of 70±1 nm (Figure 1 O-Q), which co-localizes well with the inner transport routes of the IFT motor and cargo. To demonstrate that transit of IFT motors and cargos in the lumen corresponds to passive diffusion, we examined the ATP dependence of this localization. We permeabilized NIH-3T3 cells with digitonin, which has been shown to selectively perforate the cellular membrane causing ATP to flow out of the cell (Breslow et al., 2013). This reduces ATP levels in the primary cilium by ~90% inhibiting IFT (Figure 4 I) (Ye et al., 2013). We found that in permeabilized cells, the IFT motor KIF3A and the IFT cargo α-tubulin no longer occupied the outer transport route but continued to undergo transport through the luminal route (Figure 1 R-W). Thus, the luminal transport routes for α-tubulin and KIF3A are able to persist in an energy-independent environment via passive diffusion.

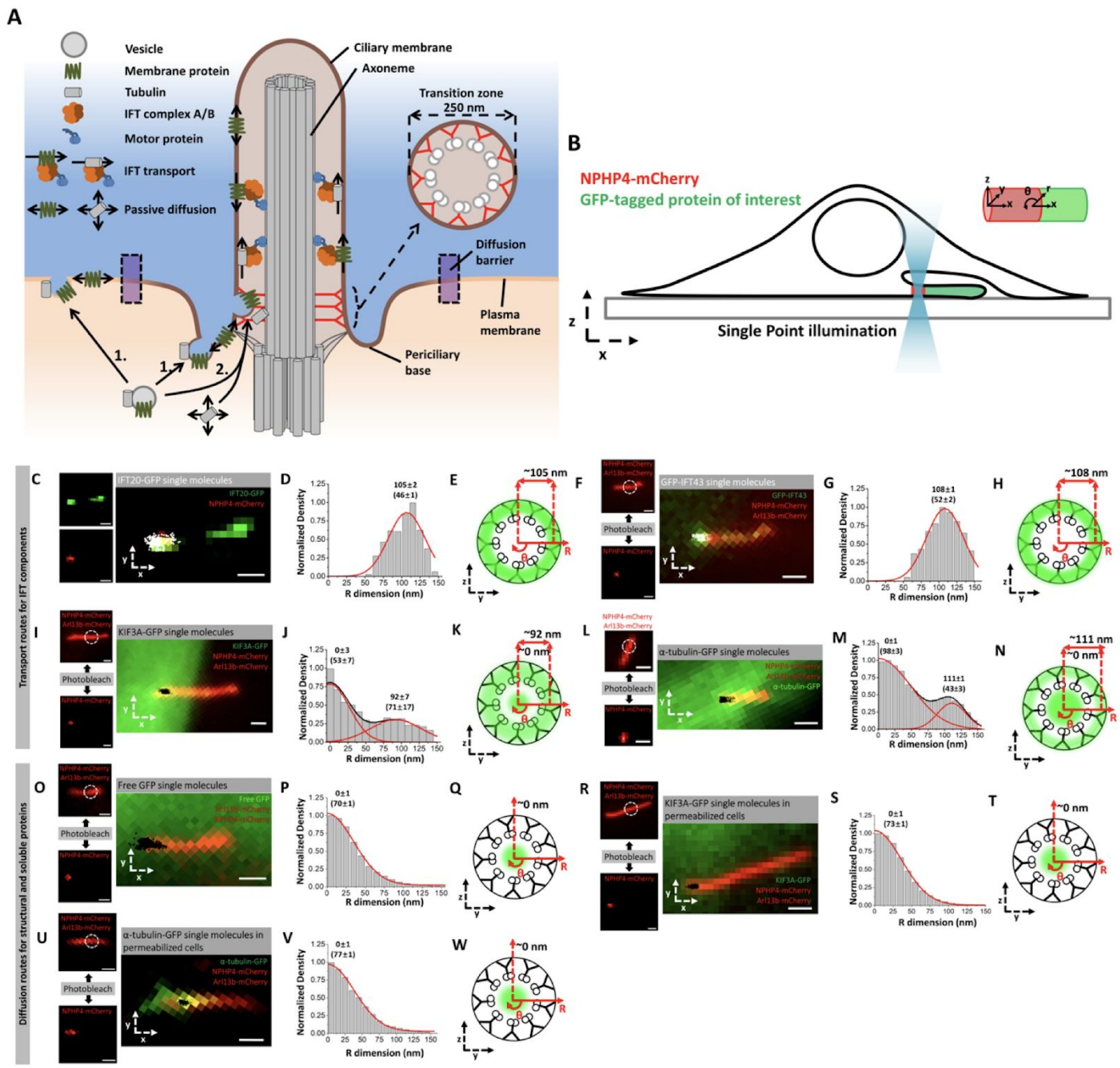


Figure 1. ***Transport routes of IFT, structural, and soluble proteins in live and permeabilized cells.*** **A)** Schematic highlighting the structure of TZ and two current models for the transport of structural and membrane proteins into primary cilia. **B)** Schematic of SPEED microscopy-based imaging of GFP-labeled proteins moving through the TZ labeled by NPHP4-mCherry. The spatial distributions of transiting molecules are described by both Cartesian (x,y,z) and cylindrical (x, q, r) coordinate systems. (C-E) Imaging of IFT-B component. **C)** Epifluorescence microscopy image of IFT20-GFP (green) and NPHP4-mCherry (red) overlaid with single-molecule IFT20-GFP locations (white spots). Scale bar: 1 μm. **D)** Single-molecule IFT20-GFP locations were converted to 3D locations and plotted as a histogram along the R dimension in the cylindrical system. Numbers indicate radial distance as mean±s.e. **E)** Spatial representation of the histogram in (D) as cross-sectional view of IFT20-GFP's transport

route (green cloud) in the TZ overlaid on the ultrastructure of the TZ. **(F-H)** Imaging of IFT-A component. **F)** Epifluorescence microscopy image of GFP-IFT43 (green), NPHP4-mCherry (red), and Arl13b-mCherry (red) overlaid with single-molecule GFP-IFT43 locations (white). Dashed white circle indicates region of photobleaching to remove Arl13b-mCherry fluorescence. Scale bar: 1 μm. Small images on the left throughout this figure show the result of photobleaching Arl13b-mCherry to reveal the precise location of NPHP4-mCherry. Scale bar: 1 μm. **G)** Single-molecule GFP-IFT43 locations along the R dimension. **H)** Spatial representation of the histogram in (G). **(I-K)** Imaging of IFT motor. **I)** Epifluorescence microscopy image of KIF3A-GFP (green), NPHP4-mCherry (red), and Arl13b-mCherry (red) overlaid with single-molecule KIF3A-GFP locations (white). Scale bar: 1 μm. **J)** Single-molecule KIF3A-GFP locations along the R dimension. **K)** Spatial representation of the histogram in (J). **(L-N)** Imaging of IFT cargo. **L)** Epifluorescence microscopy image of α-tubulin-GFP (green), NPHP4-mCherry (red), and Arl13b-mCherry (red) overlaid with single-molecule α-tubulin-GFP locations (black). Scale bar: 1 μm. **M)** Single-molecule α-tubulin-GFP locations along the R dimension. **N)** Spatial representation of the histogram in (M). **(O-Q)** Imaging of GFP. **O)** Epifluorescence microscopy image of Arl13b-mCherry (red) and NPHP4-mCherry (red) overlaid with single-molecule GFP locations (black). Scale bar: 1 μm. **P)** Single-molecule GFP locations along the R dimension. **Q)** Spatial representation of the histogram in (P). **(R-T)** Imaging of IFT motor in permeabilized cells. **R)** Epifluorescence microscopy image of Arl13b-mCherry (red) and NPHP4-mCherry (red) overlaid with single-molecule KIF3A-GFP locations (black) in permeabilized cells. Scale bar: 1 μm. **S)** Single-molecule KIF3A-GFP locations along the R dimension. **T)** Spatial representation of the histogram in (S). **(U-W)** Imaging of IFT cargo in permeabilized cells. **U)** Epifluorescence microscopy image of Arl13b-mCherry (red) and NPHP4-mCherry (red) overlaid with single-molecule α-tubulin-GFP locations (black) in permeabilized cells. Scale bar: 1 μm. **V)** Single-molecule α-tubulin-GFP locations along the R dimension. **W)** Spatial representation of the histogram in (V).

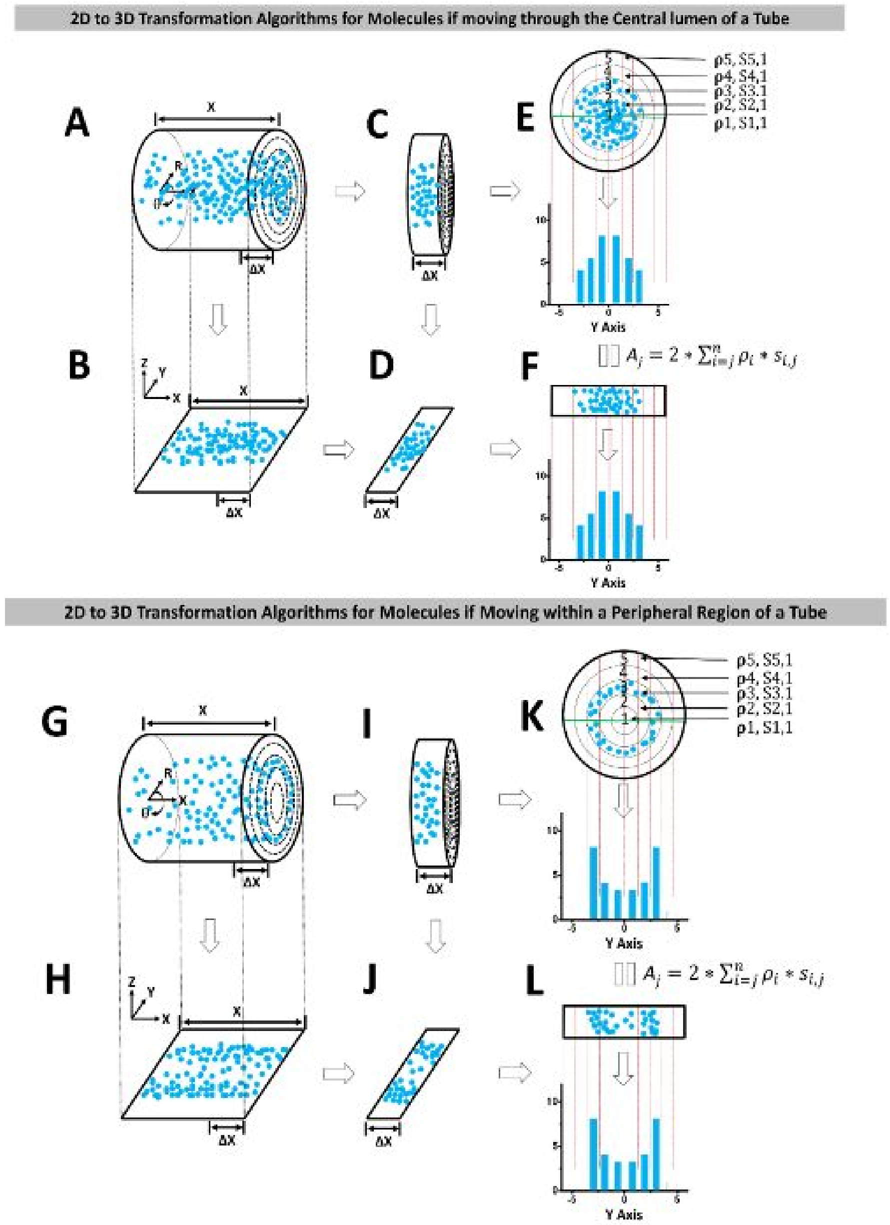


Figure 2. ***Graphical demonstration of 2D to 3D conversion process.*** Schematics demonstrate the 2D to 3D transformation algorithm for molecules, for example, if they diffuse within the central lumen (A-F) or a peripheral region (G-L) of a tube. **(A)** Ideal 3D spatial locations of randomly diffusing molecules inside a tube can be coordinated in a cylindrical coordination system (R, X, θ). In SPEED, only the 3D density map is ultimately calculated from the 2D spatial locations. However, this is an idealized case to illustrate that the 3D density map is equivalent whether the 2D or 3D spatial locations are known. **(B)** The 3D molecular locations in A are projected onto a 2D plane in a Cartesian

coordination system (X, Y, Z) by microscopy imaging. **(C)** A very thin slice (Δx) cut from the cylinder in A along x dimension. **(D)** The 3D spatial locations in the slice shown in C can be projected within a narrow 2D region. **(E)** Cross-section view of all the locations in the thin slice shown in C. These locations can be grouped into the sub-regions between concentric rings. Given the high-number randomly distributed molecules in the cylinder and the cut very thin slice, the spatial density of locations (ρ_i) in each sub-region (S_i) between two neighboring rings will be rotationally symmetrical and uniform. These locations can be further projected into 1D along the Y dimension. If the locations along Y dimension are clustered in a histogram with j columns. The total number of locations in each column (Aj) is equal to Equation 2, which can be experimentally measured as shown in **(F)**. **(G-L)** Similar as the above, the transformation process is presented for molecules diffusing within a peripheral region of a tube.

3.1.2 Tracking of TM proteins through the TZ shows their localizations at both the ciliary membrane and the ciliary lumen

To extend our technique to TM proteins, we investigated the transport route of an externally-labeled version of Somatostatin Receptor 3 (SSTR3), a G-protein coupled receptor that localizes to primary cilia and has critical functions in the hippocampus (Händel et al., 1999, Berbari et al., 2007, Einstein et al., 2010). We used an SSTR3 construct, AP-SSTR3-GFP, which can be externally labeled in live NIH-3T3 cells and marks the ciliary membrane (Ye et al., 2013, Howarth and Ting, 2008). The AP domain is an acceptor peptide attached to the N terminus of SSTR3 that, when expressed with the biotin ligase BirA, becomes biotinylated at the level of the endoplasmic reticulum. The biotinylated AP-SSTR3-GFP is trafficked to the ciliary membrane where it can be externally labeled using Alexa Fluor 647(AF647)-labeled streptavidin (Figure 4 A) (Ruba et al., 2018b, Ye et al., 2013). As expected, the AF647-labeled AP-SSTR3-GFP localized near the TZ ciliary membrane with a peak position at 131±3 nm along the cilium radius and a route width of 24±1 nm (Figure 3 B-D), after taking into consideration the estimated length of the external label (Methods).

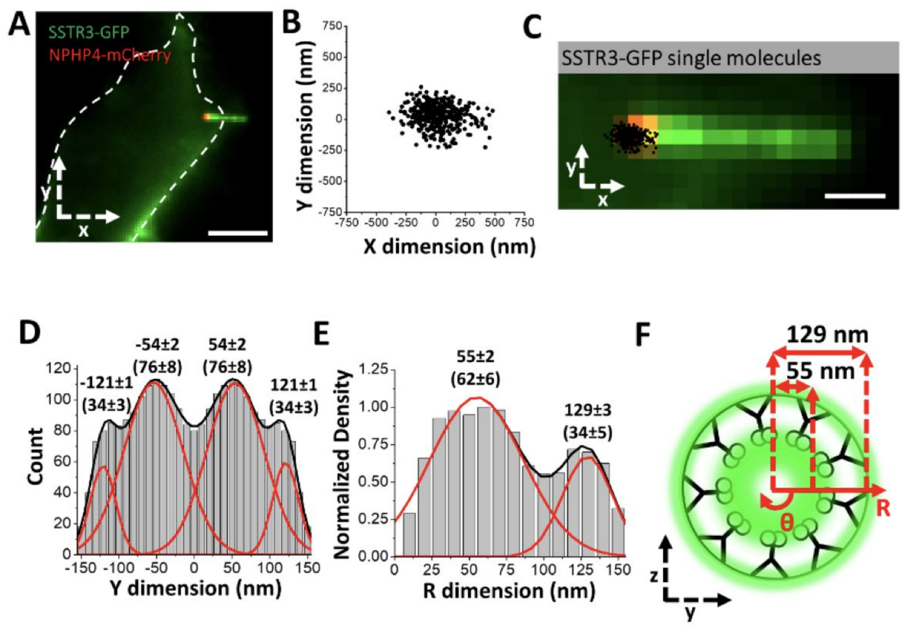


Figure 3. ***Data transformation process for obtaining 3D density histograms.*** **A)** Epifluorescence microscopy image of live cells co-expressing SSTR3-GFP (green) and NPHP4-mCherry (red) (scale bar: 5 µm). The dashed white line represents the cell border. **B)** Single-molecule SSTR3-GFP locations from the primary cilium in (A). **C)** Enlarged image of the primary cilium overlaid with single molecule SSTR3-GFP locations (black dots). Scale bar: 1 µm. Single-molecule data and epi-fluorescence images were rotated together clockwise, so that TZ is parallel to the x dimension to maintain consistency with subsequent data analyses. **D)** Y-dimensional histogram of the 2D single-molecule SSTR3-GFP locations shown in (B) and fitted by Gaussian functions, generating peak positions and Gaussian widths (in parentheses). Histogram was left-right average due to the bi-lateral symmetry of primary cilia. **E)** Conversion of the 2D data in (D) to a 3D density distribution along the R dimension in the cylindrical system for SSTR3-GFP in the TZ. **F)** Graphical representation of the histogram in (E) and cross-sectional view of SSTR3's two distinct transport routes (green clouds) in the TZ overlaid with the ultrastructure of TZ (grey).

We also examine the localization of SSTR3-GFP at the TZ in live cells (Figure 4 E,F). Surprisingly, we found SSTR3-GFP occupies two transport paths through the TZ: one near the ciliary membrane with a peak position at 129±3 nm and a route width of 34±5 nm, and the other inside the TZ lumen with a peak position at 55±2 nm and a route width of 62±6 nm (Figure 4 G,H). Based on Monte Carlo simulations which estimate the localization error of the mean for the 3D density histograms (Table 1), the inner and outer routes are spatially resolvable with localization precisions of 2.0 nm and 3.2 nm, respectively, due to the collection of a sufficient number of high-resolution single-molecule localizations. A comparison of the peak fitting areas determined that transiting molecules had a 73% frequency in the inner route and 27% in the outer route. The outer transport route for SSTR3-GFP likely corresponds to the ciliary membrane as it co-localizes with the externally-labeled AP-SSTR3-GFP (Figure 4 C,D). However, the inner transport route has not been visualized in previous measurements. It seems unlikely that the inner route reflects the GFP-labeled C-terminus of SSTR3 extending into the axonemal lumen as the 102 aa C-terminal domain of SSTR3 could extend a maximum of ~44 nm (Ainavarapu et al., 2007), given an average length of ~ 0.4 nm/aa and the 2-4 nm size of GFP, a distance that is not sufficient to bridge the ~70-100 nm between the outer and inner transport routes. It also seems unlikely that free GFP is cleaved from SSTR3-GFP and contributes to the inner route localization as the inner route of SSTR3-GFP and the passive diffusion route of free GFP are well-separated by a distance of ~ 50 nm. We conclude that our high-resolution 3D data in live cells may support a model for TM protein transport that utilizes the ciliary lumen to a greater extent than previously recognized.

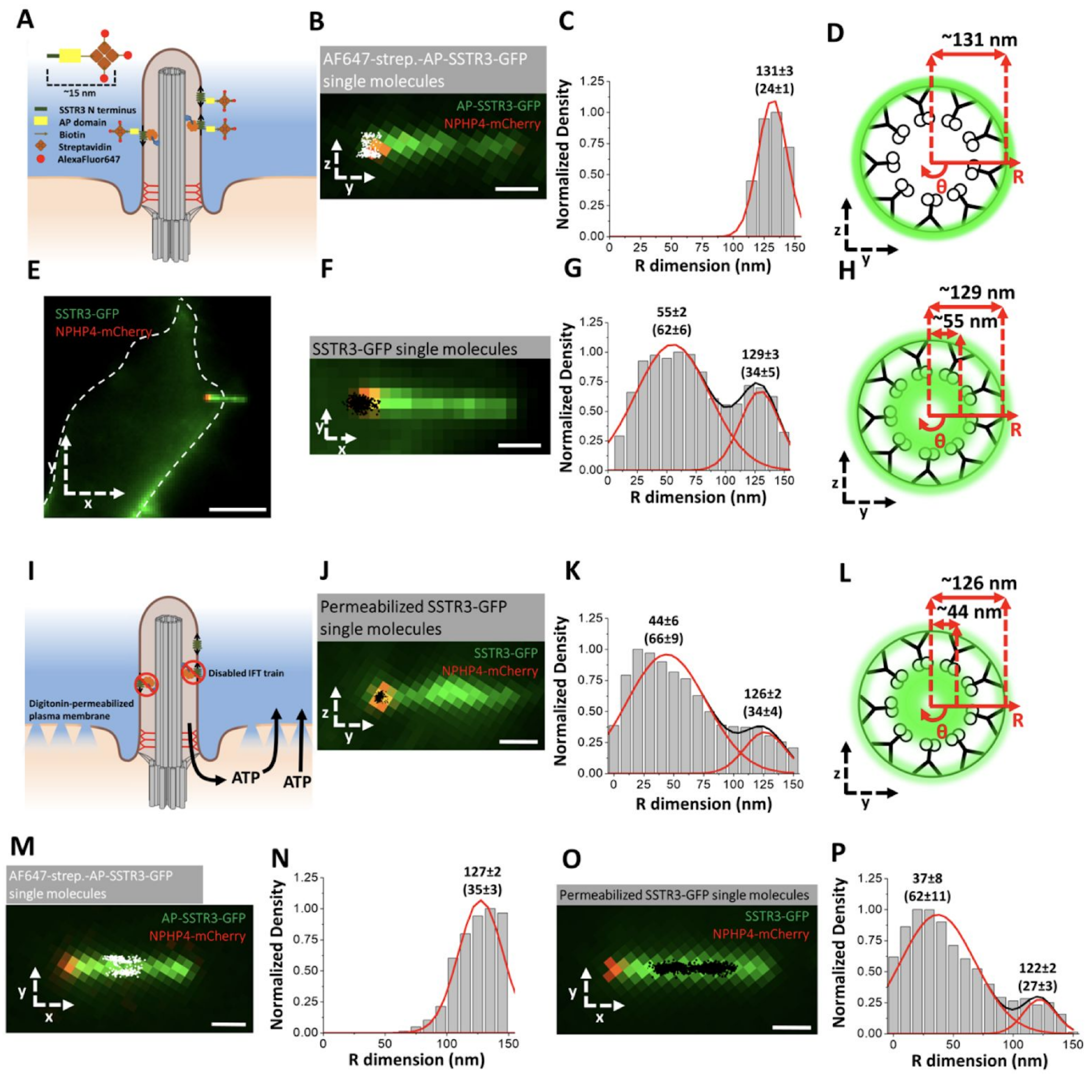


Figure 4. ***A previously uncharacterized transport route exists for membrane proteins inside the lumen of the primary cilium.*** **(A-D)** Imaging of externally-labeled AP-SSTR3-GFP. **A)** Schematic of the external labeling procedure. The extracellular SSTR3 N-terminus (green) is tagged with the AP (yellow) and biotinylated (arrow) for binding to AlexaFluor647 (red)-labeled streptavidin (orange). **B)** Image of primary cilium in live cells co-expressing AP-SSTR3-GFP (green) and NPHP4-mCherry (red) overlaid

with 2D single-molecule Alexa-Fluor-647 externally labeled SSTR3 locations (white). Scale bar: 1 μm. **C)** Single-molecule AP-SSTR3-GFP locations in the TZ plotted along the R dimension in the cylindrical system. **D)** Spatial representation of the histogram in (C). **(E-H)** Imaging of SSTR3-GFP. **E)** Epifluorescence microscopy image of live cell co-expressing SSTR3-GFP (green) and NPHP4-mCherry (red) (scale bar: 5 μm). The dashed white line represents the cell border. **F)** Enlarged image of the primary cilium overlaid with single molecule SSTR3-GFP locations (black dots). Scale bar: 1 μm. Single-molecule data and epi-fluorescence images were rotated together clockwise, so that TZ is parallel to the x dimension to maintain consistency with subsequent data analyses. **G)** Single molecule SSTR3-GFP locations in the TZ along the R dimension in the cylindrical system. **H)** Spatial representation of the histogram in (G). **(I-L)** Imaging of SSTR3-GFP in permeabilized cells **I)** Schematic of the permeabilization procedure, resulting in ATP diffusion from the cell. **J)** Image of primary cilium in live cells co-expressing SSTR3-GFP (green) and NPHP4-mCherry (red) overlaid with 2D single-molecule SSTR3-GFP locations (white) in permeabilized cells. Scale bar: 1 μm. **K)** Single-molecule SSTR3-GFP locations in the TZ plotted along the R dimension in the cylindrical system. **L)** Spatial representation of the histogram in (K). **M) and N)** Same as (F) and (G) for AP-SSTR3-GFP in the ciliary shaft. **O) and P)** Same as (J) and (K) for SSTR3-GFP in the ciliary shaft in permeabilized cells.

Given that transit in the ciliary lumen occurs by passive diffusion for free GFP, IFT motors, and IFT cargos (Figure 1), we hypothesized that energy-independent transport may account for SSTR3 mobility in the lumenal route. To test this, we permeabilized the cells with digitonin and then performed SPEED microscopy. Although permeabilization did reduce the total frequency of single-molecule events (from 32.4±14.9 events/s to 10.1±1.6 events/s, Figure 5 C), the location of transiting SSTR3-GFP molecules was largely unaffected (Figure 4 I-L). The majority of SSTR3-GFP molecules continued to utilize the inner route (85% compared to 73% in unpermeabilized cells). The location of the SSTR3 inner transport route showed a slight shift towards the cilium center in permeabilized cells, with a peak position at 44±6 nm along the cilium radius and a route width of 66±9 nm (Figure 4 K,L). For the outer transport route, permeabilization and loss of ATP increased the immobile portion and decreased the directionally-moving SSTR3 molecules (determined via Mean Squared Displacement (MSD) analysis of single molecule trajectories), in agreement with others (Ye et al., 2013) (Figure 5 A,B).

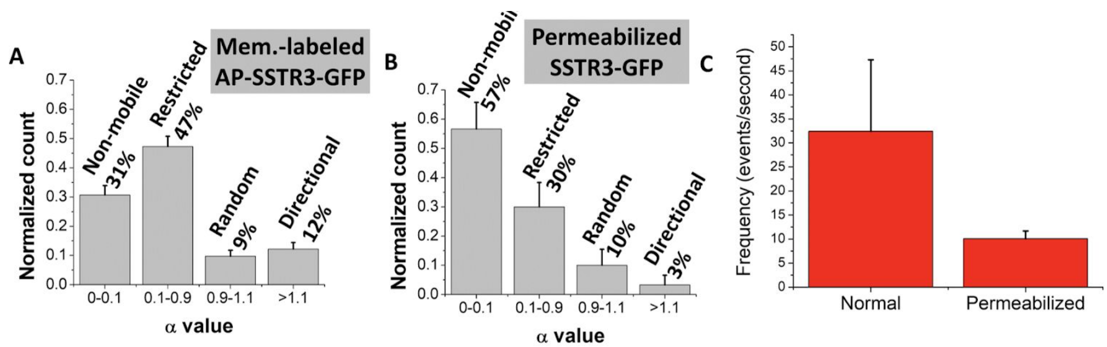


Figure 5. ***Mean squared displacement analysis for membrane labeled AP-SSTR3-GFP in normal cells and SSTR3-GFP in permeabilized cells as well as single molecule frequency for SSTR3-GFP in normal and permeabilized cells.*** **A)** Histogram of α values for membrane-labeled AP-SSTR3-GFP trajectories (12% directional movement, n = 205). **B)** Histogram of α values for SSTR3-GFP trajectories in permeabilized cells (3% directional movement, n = 30). **C)** Frequency of SSTR3 molecules in the transition zone of live (cilia n = 6) and permeabilized cells (cilia n = 5).

To determine whether these SSTR3 transport routes occur in different regions of the primary cilium, we determined the 3D density map for SSTR3 at the basal body and the cilium shaft (Figure 6 A-G). At the basal body, using γ-tubulin-mCherry as a marker (Muresan et al., 1993), SSTR3 again had two transport routes: one near the outer basal body with a peak position at 120±4 nm and a route width of 40±6 nm and the other inside the lumen of the basal body with a peak position at 55±2 nm and a route width of 54±4 nm (Figure 6 H,K). These localizations suggest that the majority of SSTR3 likely enters and/or exits the transition zone through the lumen of the basal body rather than through the gaps in the 9+0 microtubules or through diffusion along the membrane. Along the cilium shaft, two transport routes were also detected for SSTR3-GFP (Figure 6 J,M) while only the outer route was detected for externally-labeled AP-STR3-GFP (Figure 4 M,N).

To determine whether this paradigm is similar for other TM proteins, we determined the 3D transport routes at the basal body, the TZ, and the cilium shaft for HTR6, a GPCR that binds serotonin and localizes to primary cilia, tagged with GFP on its C-terminus (Berbari et al., 2008). We found that HTR6 also occupies two transport routes, one in the lumen and one at the ciliary membrane, with similar locations as those of SSTR3. In comparison to SSTR3, an even larger proportion of HTR6 localized to the lumenal route (Figure 6 N-P). One possibility for this phenomenon may be that HTR6 undergoes higher turnover in its targeting to the primary cilium and thus spends more time in intraciliary and intracellular transport rather than embedded in the ciliary membrane.

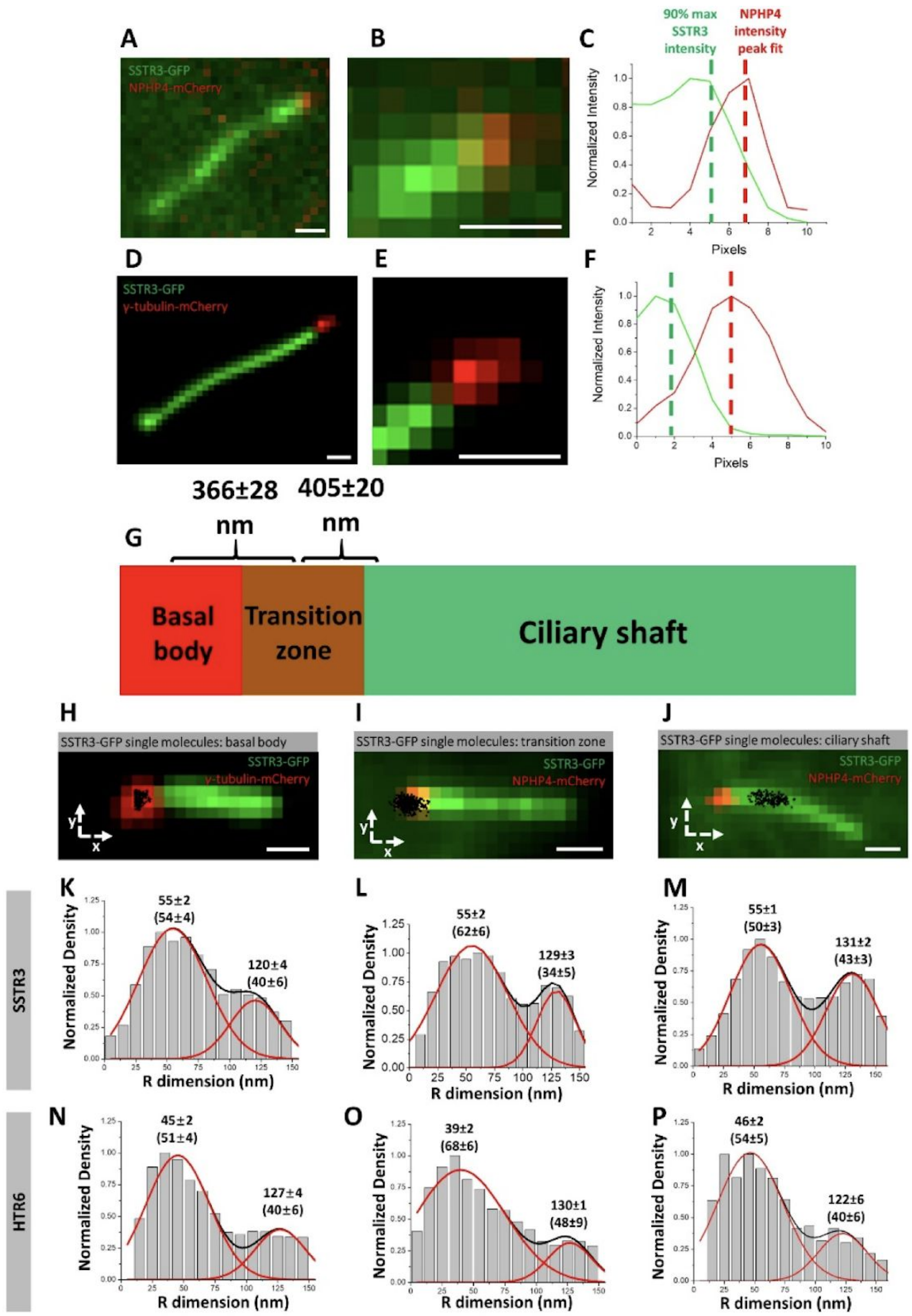


Figure 6. ***The outer and inner transport routes for HTR6 parallels SSTR3 and is continuous in the basal body, transition zone, and ciliary shaft.*** **A)** Epifluorescence image of SSTR3-GFP (green) and NPHP4-mCherry (red). Scale bar: 1 µm. **B)** Enlarged version of (A). Scale bar: 1 µm. **C)** The intensity plot showing the distance between the peak of NPHP4-mCherry and 90% of the maximum SSTR3-GFP intensity. **D)**

Epifluorescence image of SSTR3-GFP (green) and γ-tubulin-mCherry (red). Scale bar: 1 μm. **E)** Enlarged version of (D). Scale bar: 1 μm. **F)** The intensity plot showing the distance between the peak of γ-tubulin-mCherry and 90% of the maximum SSTR3-GFP intensity. **G)** Schematic of the organization of the basal body, transition zone, and ciliary shaft. **H)** Enlarged image of the primary cilium marked with SSTR3-GFP (green) and γ-tubulin-mCherry (red) and overlaid with single-molecule SSTR3-GFP locations (black dots). Scale bar: 1 μm. **I)** Same as (H) except NPHP4-mCherry (red). **J)** Same as (I). **K), L), and M)** 3D transformed density histogram for SSTR3 at the basal body, transition zone, and ciliary shaft, respectively. **N), O), and P)** Same as (K), (L), and (M) except with HTR6.

3.2 Determining the underlying mechanisms that drive membrane protein transport in primary cilia

3.2.1 RAB8A, a regulator of ciliary vesicle targeting co-localizes and co-transports with SSTR3

The luminal localization of SSTR3 and HTR6 was surprising and suggests that vesicular trafficking may contribute to the luminal transport of TM proteins. Vesicular trafficking components such as Rabin8 and RAB8A promote docking of ciliary vesicles to the microtubule triplets that comprise the basal body (Nachury et al., 2007, Yoshimura et al., 2007). We thus hypothesized that RAB8A may mediate the transport of SSTR3. We measured the transport routes of GFP-RAB8A through the TZ and cilium shaft in live primary cilia. Interestingly, we found that RAB8A utilized two transport routes in both of these locations (Figure 7 A-F). The majority of transit events (87%) at the TZ localized to the luminal transport route with a mean position of 50±6 nm along the cilium radius and a route width of 75±11 nm while the remainder of events localized at an outer transport route with a mean position of 133±3 nm along the cilium radius and a route width of 40±7 nm (Figure 7 A-C).

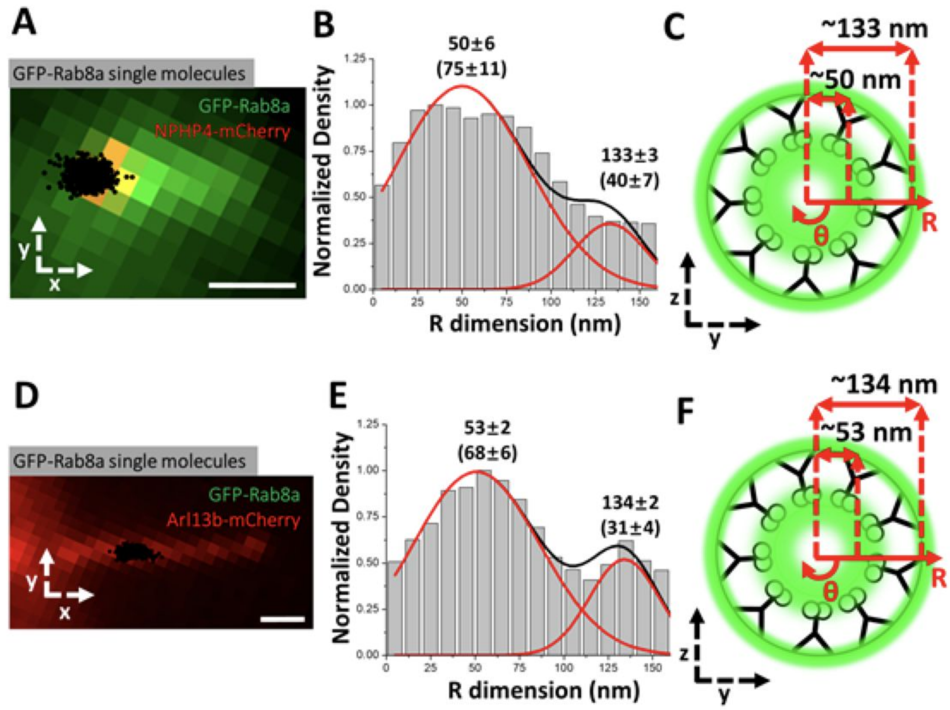


Figure 7. ***The transport routes for RAB8A are similar to those of SSTR3.*** **(A-C) Imaging of GFP-RAB8A in the TZ. A)** Epifluorescence microscopy image of GFP-RAB8A (green) and NPHP4-mCherry (red) overlaid with single-molecule GFP-RAB8A locations (black). Scale bar: 1 µm. **B)** Single-molecule GFP-RAB8A locations plotted along the R dimension. **C)** Spatial representation of (B). (D-F) Imaging of GFP-RAB8A in the cilium shaft. **D)** Epifluorescence microscopy image of GFP-RAB8A (green) and Arl13b-mCherry (red) overlaid with single-molecule GFP-RAB8A locations (black). Scale bar: 1 µm. **E)** Single-molecule GFP-RAB8A locations plotted along the R dimension. **F)** Spatial representation of (E).

RAB8A largely associates with membranes and has little turnover during the vesicle transport process in live cells (Grigoriev et al., 2011). The 3D transport routes for RAB8A, SSTR3 and HTR6 mapped by SPEED microscopy localize at the same position along the cilium radius that is outward from the passively-diffusing soluble molecules. We thus hypothesized that RAB8A may be tightly associated with vesicles bearing ciliary TM proteins, and causes them to hug the microtubule walls of the ciliary lumen. In this case, the single molecule trajectories of SSTR3 and RAB8A within primary cilia should be correlated. We utilized simultaneous dual-channel tracking of both SSTR3-mCherry and GFP-RAB8A in transfected NIH-3T3 cells and determined the angle difference between every step of each trajectory. In principle, an average angle difference close to 0° indicates a high degree of correlation between the trajectories of the proteins, whereas a wide distribution of angle differences indicates no correlation between the trajectory movements. We used measurements of the level of co-movement between positive and negative controls to generate a standard curve (Figure 8 M). The positive control experiments indicated that green and red dyes attached to 100-nm beads (Figure 8 C-F and Video: https://youtu.be/0_J0P40U270) have a probability of 95% for co-movement, when "co-movement" was defined as the total angle differences of their trajectories that fell within ~37° centered on the 0° bin (Figure 8 D-F). The negative control indicated that free fluorescein and JF561 dyes (Figure 8 G-J) have a probability of co-movement of ~13% (Figure 8 I,J). By referencing the standard curve (Figure 8 M), we identified a probability of 24% co-movement translates to a 15% co-movement for SSTR3 and RAB8A trajectories in cilia above the negative control. Both the co-localization of the 3D transport and the degree of co-movement suggest that RAB8A may indeed by mediating the transport of SSTR3 (Figure 8 A,B,K,L).

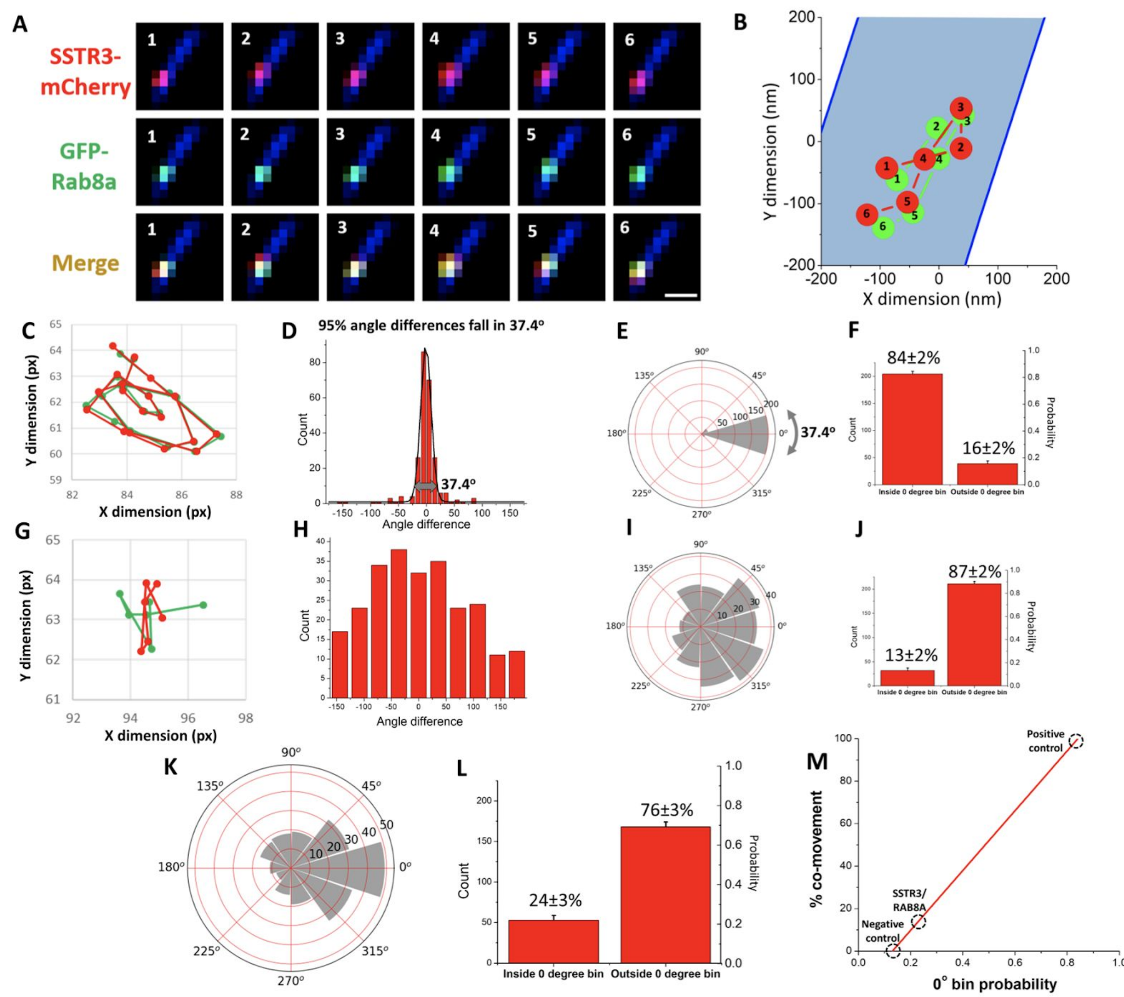


Figure 8. RAB8A shows co-movement with SSTR3 at the ciliary base. A) Montage of a typical SSTR3-mCherry and GFP-RAB8A single-molecule co-moving event appearing simultaneously and respectively in red and green detection channels near the base of the primary cilium. Scale bar: 1 μm. **B)** Plot of the single-molecule trajectories of SSTR3-mCherry (red) and GFP-RAB8A (green) from (A). Numbers denote frame number. **C)** Single molecule fitting and channel alignment of trajectory in Video 2. **D)** Histogram and peak fitting of all the angle differences in the positive control video. **E)** Radial histogram with the 0o bin sized to encompass 95% of the positive control angle

differences. **F)** Bar graph showing ratio of angle differences that fall in and out of the 0o bin. **G)-J)** Same as (C)-(F) except with co-tracking of negative control single molecule trajectories. **K)** Radial histogram of the angle difference between each step of all co-appearing SSTR3-mCherry and GFP-RAB8A trajectories that passed selection criteria (Methods). **L)** Bar graph showing the proportion of trajectories that fell in and out of the co-moving bin as defined by the positive control. **M)** Standard curve was calculated by sampling different ratios of trajectories from positive and negative controls which was repeated 1,000 times.

3.2.2 Golgicide A suppresses TM protein frequency equally in both routes and prevents ciliogenesis

As a second approach to test the role of vesicular trafficking in SSTR3 transport in the ciliary lumen, we treated cells with varying concentrations of Golgicide A (GCA), an inhibitor of Golgi export of TM proteins that works by inhibiting GBF1 from interacting with both Arf4 and the ciliary targeting sequence, a necessary step in sorting and export from the Golgi (Figure 9 A) (Wang et al., 2012, Wang et al., 2014). NIH-3T3 cells were serum starved in the absence or presence of GCA for 24 hr. GCA inhibited the length and ultimately the presence of primary cilia in a dose-dependent manner (Figure 9 B-D), indicating that membrane flux to primary cilia is an important component of ciliogenesis, particularly in relation to vesicle transport (Nachury et al., 2007). GCA treatment also reduced the frequency of single-molecule SSTR3-GFP events at the TZ in a dose-dependent manner (Figure 9 E). For the remaining events, the relative density and locations of the 3D transport routes were statistically unaffected (Figure 9 F-I), indicating that a similar mechanism of entry is utilized by both the inner and outer routes. It may be that the choice of which route to take is stochastic and based on the cross-sectional area of ciliary entry given that the space between the microtubules and the ciliary membrane accounts for ~30% of the cross-sectional area while the ciliary lumen accounts for ~70%. It thus appears that both the inner and outer SSTR3 transport routes are linked to vesicular transport mechanisms, with the inner route accommodating the bulk of SSTR3 transport in these experimental conditions.

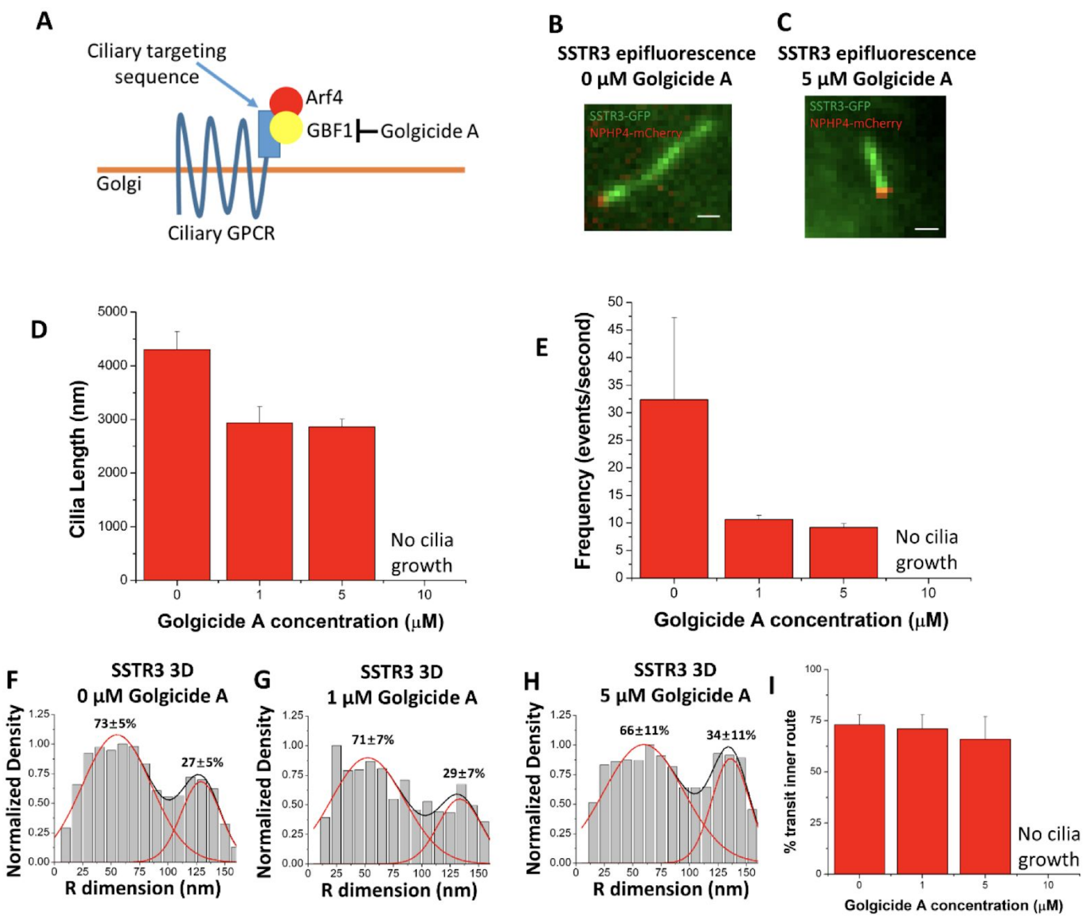


Figure 9. ***Golgicide A reduces SSTR3 frequency but does not alter transport routes.*** **A)** Schematic outlining Golgicide A's mechanism of action. **B,C)** Epifluorescence images of SSTR3-GFP in NIH-3T3 cells treated with B) no Golgicide A or C) 5 μM Golgicide A for 24 hr. Scale bars: 1 μm. **D)** Bar graph showing cilia length vs. Golgicide A concentration. (0 μM n = 29, 1 μM n = 8, 5 μM n = 12, bar graph represents mean ± s.e.)

E) Bar graph showing SSTR3 single molecule frequency vs. Golgicide A concentration (0 μM n = 5, 1 μM n = 3, 5 μM n = 4, bar graph represents mean ± s.e.) **F)-H)** 3D transport routes for SSTR3 in 0, 1, and 5 μM Golgicide A, respectively, plotted along the R dimension. **I)** Summary of percentage of SSTR3 transport in the inner transport route for 0, 1, 5, and 10 μM Golgicide A (statistics summarized in Table 1).

3.2.3 The ciliary lumen is the preferred transport route for TM proteins during clathrin-mediated, signal-dependent internalization

Given the presence of both inner and outer routes for SSTR3 transit at the TZ, we wondered whether the relative utilization of these routes could be affected under conditions of active GPCR signaling. Previous work has shown that SSTR3 is actively removed from primary cilia after binding somatostatin (Ye et al., 2018, Green et al., 2016). We found that after stimulation with 10 μM somatostatin for 1 hr, the majority of SSTR3 single-molecule trajectories at the base of primary cilia displayed characteristics of both random diffusion and unclear movement patterns (Figure 10 D-I) while a minority displayed long, directional movement (Figure 10 A-C and Video: https://youtu.be/brAs_R34i5g). We then focused on every step of each trajectory and measured the average directionality of all the trajectories for each concentration of somatostatin in a range from 0-100 μM. As the concentration of somatostatin increased, the average directionality of trajectories shifted from 6% into the primary cilium to 16% out of the primary cilium (Figure 10 J), reflecting an overall removal of SSTR3 from the primary cilium, consistent with previous work (Ye et al., 2018). In concert with the shift in the average directionality of the trajectories, we found that the percent of transiting molecules in the lumenal transport route also increased until 100% of molecules utilized the inner route at 100 μM somatostatin (Figure 10 K-M,O).

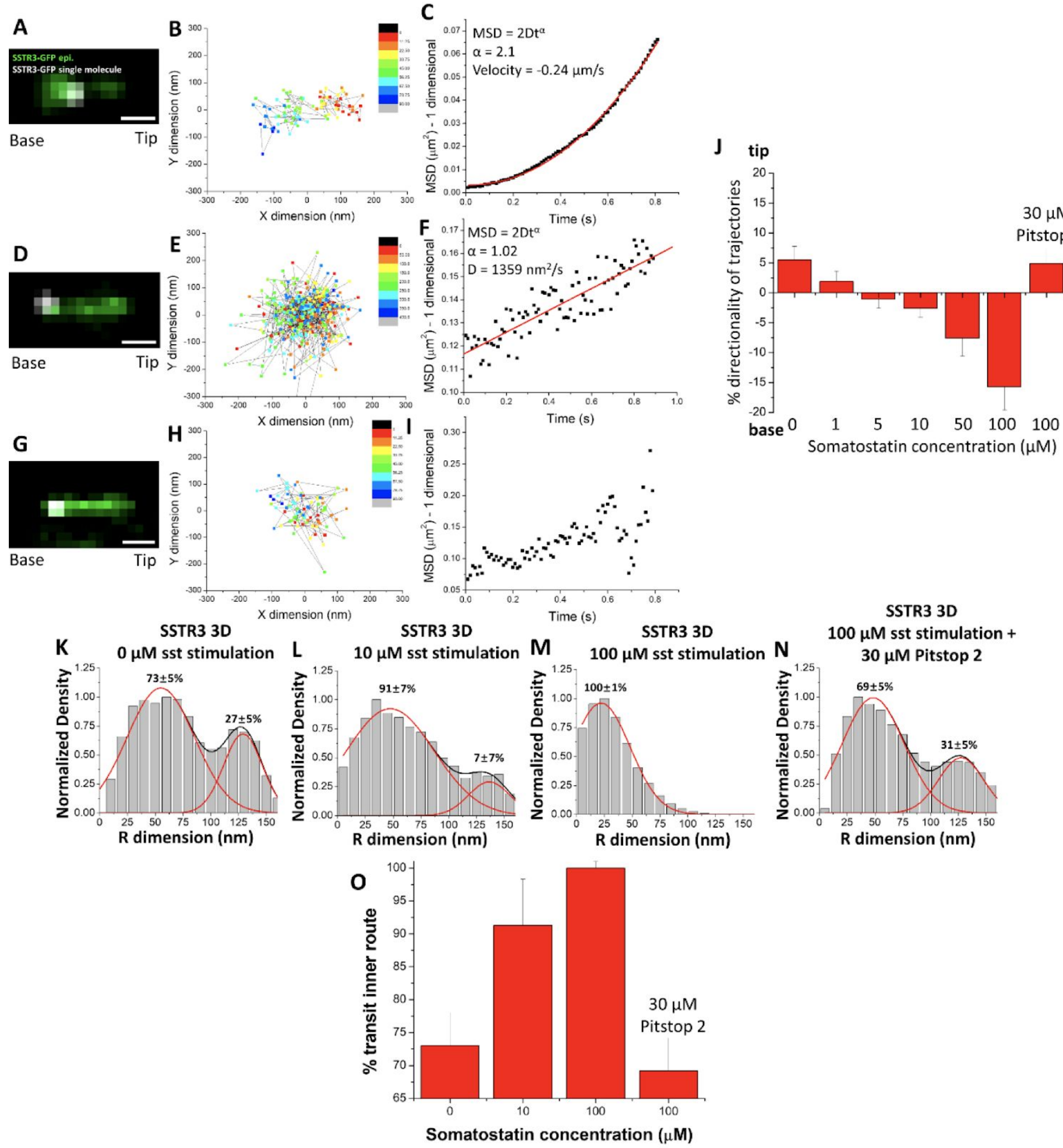


Figure 10. ***Somatostatin stimulation causes an increase in the transport of SSTR3 in the inner transport route.*** **A)** Epifluorescence image of SSTR3-GFP overlaid with single-molecule locations (white). Scale bar: 1 μm. **B)** 2D scatterplot of the Gaussian

fitting results of the single molecule trajectory in (A). **C)** MSD vs. time plot of trajectory in (A) and (B). **D)-F) and G)-I)** Same as (A)-(C) showing trajectories with different types of movement patterns. **J)** Bar chart showing the average percent directionality of every trajectory step in different concentrations of somatostatin and Pitstop 2. (0 μM sst n = 489, 1 μM sst n = 908, 5 μM sst n = 889, 10 μM sst n = 1080, 50 μM sst n = 275, 100 μM sst n = 159, 100 μM sst/30 μM Pitstop 2 n = 427, bar graphs represent mean ± s.e.) **K)-N)** 3D transport routes in the TZ for SSTR3 in 0, 10, 100 μM somatostatin and 100 μM somatostatin with 30 μM Pitstop 2. **O)** Percentage of SSTR3-GFP molecules utilizing the inner transport route from (K)-(N). (statistics summarized in Table 1).

The mechanism for SSTR3 removal from primary cilia upon somatostatin stimulation has been shown to involve recruitment of β-arrestin to primary cilia, binding to SSTR3, and promotion of clathrin-coated endocytosis (Green et al., 2016, Oakley et al., 1999). To investigate the role of clathrin-coated endocytosis in SSTR3 removal from primary cilia, we measured the directionality and 3D transport route of SSTR3 at 100 μM somatostatin following a 1 hour pre-incubation with 30 μM Pitstop 2, a potent clathrin-coated endocytosis inhibitor (von Kleist et al., 2011). Pitstop 2 treatment completely blocked the dose-dependent reversal of SSTR3 directionality during somatostatin stimulation (Figure 10 J) and prevented the shift of transiting SSTR3 to the luminal transport route (Figure 10 N,O) suggesting that clathrin-coated endocytosis plays a role in facilitating movement through the inner transport route.

3.2.4 The ciliary lumen is the preferred transport route for TM proteins during ciliogenesis

As a second test of whether the relative utilization of the inner and outer transport routes could be affected by cellular conditions, we examined the transit of SSTR3 and

HTR6 molecules during different stages of ciliary growth. Due to the non-linear growth kinetics of primary cilia (Figure 11 A), we selected time points of 1, 3, 6, 12, and 24 hours following serum starvation. For both SSTR3 and HTR6, the majority of transit events at the TZ occurred via the luminal transport route early in ciliary growth and shifted to include more outer transit events as the primary cilia matured (Figure 11 B-L). We used the Pearson correlation coefficient to correlate these changes with values close to +1 or -1 indicating high degrees of positive and negative correlation, respectively, and values close to 0 indicating no correlation. Changes in percent transit in the inner route and ciliary length had strong negative correlation at -0.90 and -0.83 for SSTR3 and HTR6, respectively (Figure 11 M), suggesting that these phenomena may arise from the same biological process that regulates ciliogenesis. The percent transit in the inner route showed no correlation with the frequency of single-molecule events (Pearson coefficients of 0.13 and 0.15 for SSTR3 and HTR6, respectively, Figure 12). These results suggest that, unlike structural and IFT proteins in flagellum elongation (Dirksen et al., 1975, Wood et al., 2014), TM proteins may not be transported to the growing cilium at a higher rate earlier in ciliogenesis. Thus, the changes in percent transit in the central transport route may be due to other factors, such as structural and selective competency of the transition zone which forms early in ciliogenesis (Williams et al., 2011).

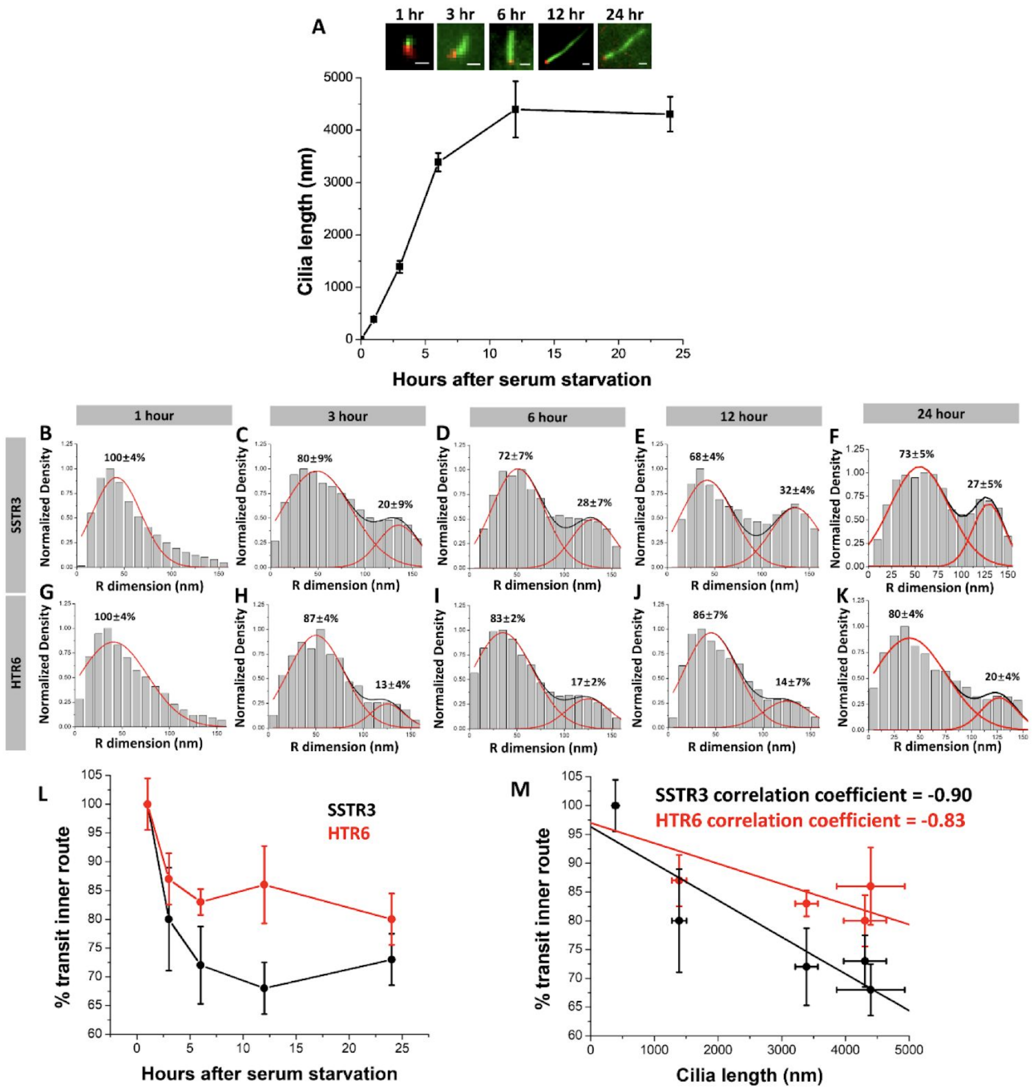


Figure 11. ***SSTR3 and HTR6 use the inner transport route at a higher frequency at earlier times of ciliogenesis.*** **A)** Growth curve of primary cilia following serum starvation. Insets: green = SSTR3-GFP, red = NPHP4-mCherry. **B)-F)** 3D transport routes for SSTR3 following 1, 3, 6, 12, and 24 hours of serum starvation. **G)-K)** Same as (B)-(F) except with HTR6. **L)** Summary of (B)-(K). **M)** Graph of percent transit in inner route vs. average cilia length with Pearson's correlation coefficient for both SSTR3 and HTR6.

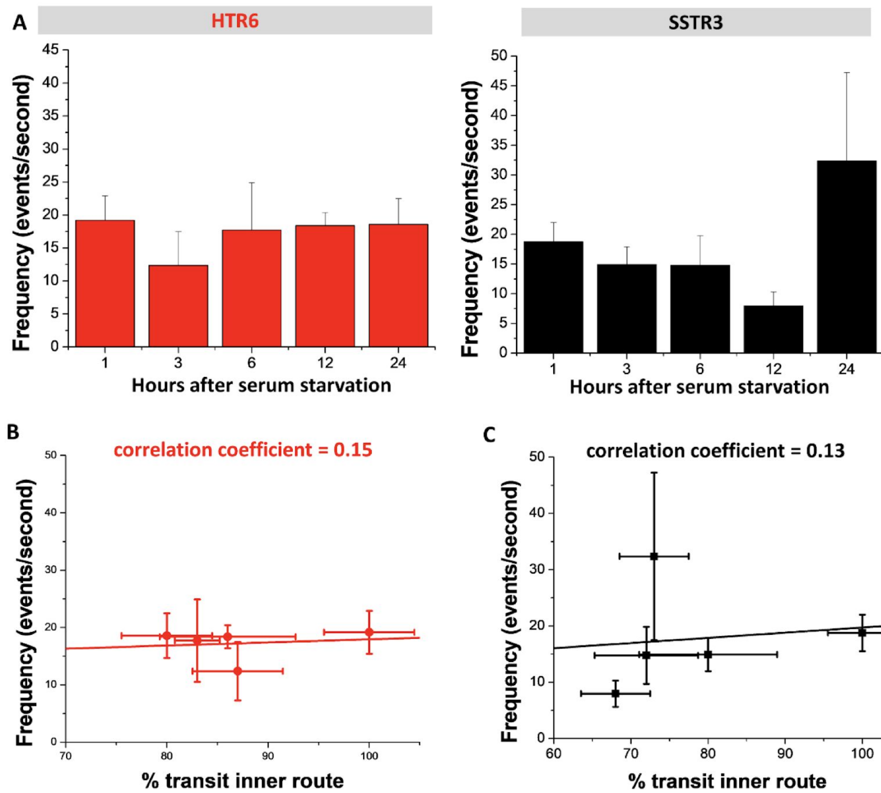
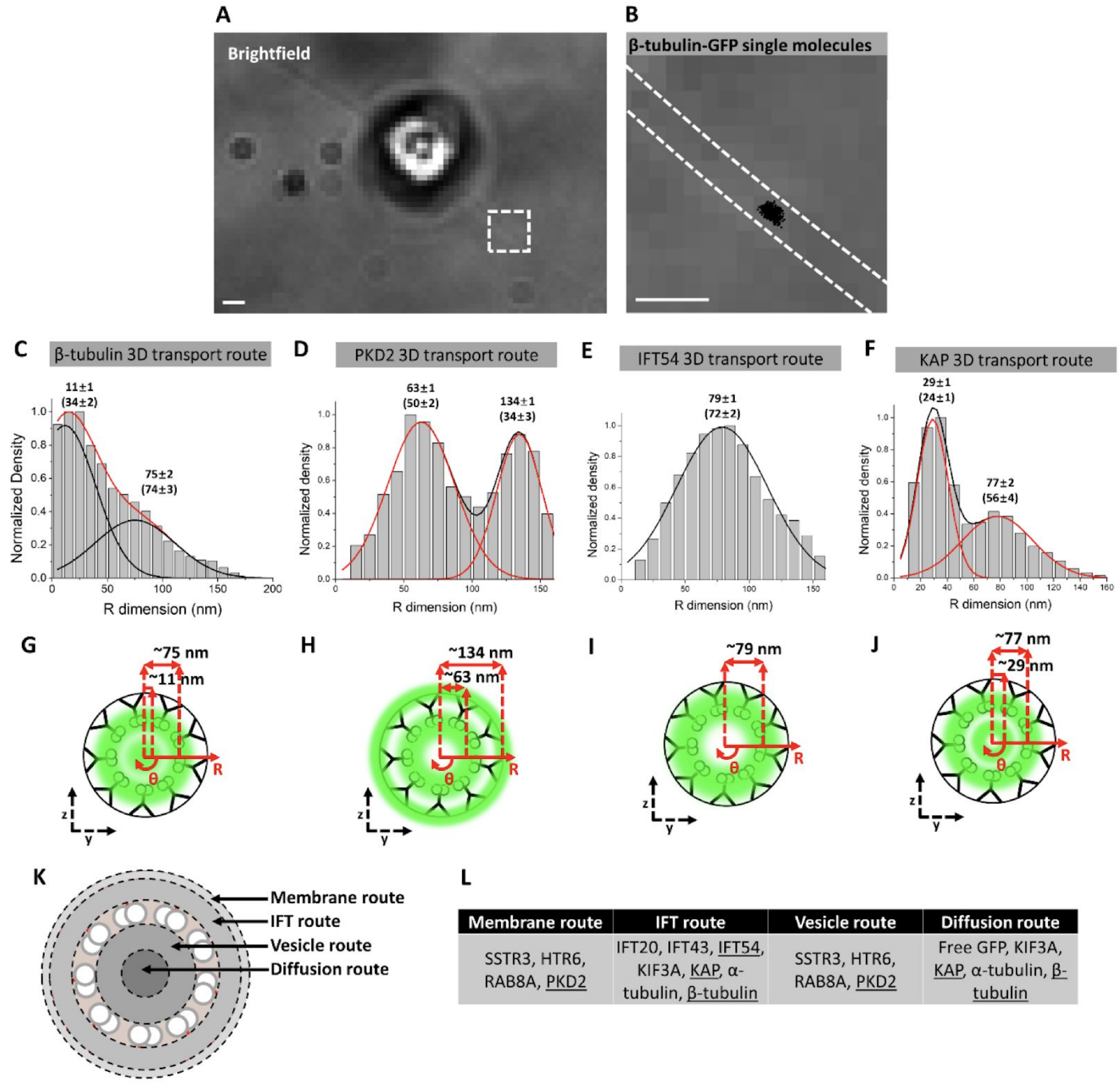


Figure 12. ***Single molecule frequency does not correlate with transport in the inner route.*** **A)** Single molecule frequency for both SSTR3 and HTR6 at different times after serum starvation. **B)** Single molecule frequency vs. percent transport in inner route for HTR6 with Pearson's correlation coefficient. **C)** Same as (B) with SSTR3.

3.2.5 The ciliary lumen accommodates similar transport routes in Chlamydomonas reinhardtii flagella

The utilization of an inner transport route by a TM protein in the TZ of mammalian primary cilia was surprising and we sought to determine whether such events occur in other cilia. Therefore, we performed single molecule tracking (Video: https://youtu.be/8uKfZzsPtbU) and determined the 3D transport routes for several classes of proteins in Chlamydomonas reinhardtii flagella: the IFT cargo β-tubulin (Craft et al., 2015), the TM protein PKD2 (Huang et al., 2007), the IFT protein IFT54 (Wingfield et al., 2017), and the kinesin-2 motor subunit KAP (Mueller et al., 2005). Overall, the results were quite similar to mammalian primary cilia in terms of transport routes observed for each protein. β-tubulin, PKD2, and KAP all had inner and outer transport routes in the shaft of the Chlamydomonas flagellum (Figure 13 A-J) that roughly correlated to the same transport routes for α-tubulin, SSTR3, and KIF3A in primary cilia (Figure 1). One difference between the systems was a radial shift in the inner routes for β-tubulin and KAP by 11 nm and 29 nm, respectively, perhaps due to the presence of the central pair of microtubules in the motile Chlamydomonas flagellum (Czarnecki and Shah, 2012). A second difference was a shift of the IFT54 transport route to 79 nm along the radius of flagella which is ~25 nm more central compared to IFT20 and IFT43 in primary cilia. This discrepancy may be due to slightly different locations of the axoneme in flagella and cilia and/or different placement of IFT54 within the IFT particle compared to IFT20 and IFT43. Taken together, the general location of the 3D transport routes appears to be similar in the motile flagellum of Chlamydomonas and the primary cilium of mammalian cells.



Membrane route	IFT route	Vesicle route	Diffusion route
SSTR3, HTR6, RAB8A, PKD2	IFT20, IFT43, IFT54, KIF3A, KAP, α-tubulin, β-tubulin	SSTR3, HTR6, RAB8A, PKD2	Free GFP, KIF3A, KAP, α-tubulin, β-tubulin

Figure 13. ***Transport routes in Chlamydomonas reinhardtii flagella closely reflect those in primary cilia.*** **A)** Brightfield image of *Chlamydomonas reinhardtii* stably expressing β-tubulin-GFP. Red dashed box shows the region of flagellum selected for single-molecule imaging. Scale bar: 1 µm. **B)** Brightfield image from boxed region in (A) with dashed white lines indicating location of flagellum and superimposed with single-molecule localizations (black dots). Scale bar: 1 µm. **C)-F)** 3D transport routes for β-tubulin, PKD2, IFT54, and KAP, respectively, in the shaft of the flagellum. **G)-J)** Spatial representations of (C)-(F), respectively. **K)** Summary of various transport routes observed in a cross-section view of a primary cilium. **L)** Table indicating which proteins utilize the transport routes depicted in (K). Underlined proteins indicate transport routes determined in *Chlamydomonas reinhardtii*.

Table 1. ***Transport route localization precision and precision/radius ratio for all 3D transport routes.***

Protein	Cilia area	Condition	# of cilia	Route mean (nm)	Route S.D. (nm)	# of single molecules	Average precision (nm)	Precision /radius ratio	σ_{TR} (nm)
α-tubulin	TZ	Normal	5	0	49	579	16	>>0.63	<0.1
				111	22	517	16	0.14	3.2
SSTR3	TZ	Normal	6	55	31	1471	18	0.32	2.0
				129	17	1152	18	0.14	3.2
IFT20	TZ	Normal	5	105	23	1162	16	0.15	1.8
IFT43	TZ	Normal	5	108	26	1078	16	0.15	1.8
KIF3A	TZ	Normal	5	0	27	613	16	>>0.63	<0.1
				92	36	633	16	0.17	2.7
GFP	TZ	Normal	5	0	33	1135	16	>>0.63	<0.1
α-tubulin	TZ	Perm.	6	0	39	808	16	>>0.63	<0.1
KIF3A	TZ	Perm.	5	0	37	918	16	>>0.63	<0.1
SSTR3	TZ	Perm.	5	44	33	1137	23	0.53	4.2
				126	17	509	23	0.18	4.9
SSTR3	TZ	Membrane	4	131	12	1142	13	0.10	1.7
SSTR3	BB	Normal	4	55	27	660	23	0.42	3.9
				120	20	159	23	0.19	7.6

SSTR3	Shaft	Normal	5	55	25	3117	15	0.27	1.0
				131	22	2078	15	0.11	1.4
HTR6	BB	Normal	4	45	26	765	27	0.59	10.3
				127	20	187	27	0.21	9.1
HTR6	TZ	Normal	5	39	34	1364	24	0.62	9.2
				130	24	337	24	0.19	6.1
HTR6	Shaft	Normal	5	46	27	1659	22	0.48	2.4
				122	20	457	22	0.18	4.6
RAB8A	TZ	Normal	4	50	38	2326	13	0.26	0.9
				133	20	402	13	0.10	2.9
RAB8A	Shaft	Normal	4	53	34	2648	14	0.26	1.3
				134	16	669	14	0.10	2.8
SSTR3	TZ	1 μM GCA	3	53	31	957	20	0.37	2.4
				130	22	372	20	0.15	4.4
SSTR3	TZ	5 μM GCA	4	60	35	824	22	0.37	4.1
				136	17	424	22	0.16	4.6
SSTR3	TZ	1 hr growth	3	42	26	1269	21	0.50	3.1
SSTR3	TZ	3 hr growth	4	49	37	1596	14	0.29	1.6
				135	43	239	14	0.10	3.9
SSTR3	TZ	6 hr growth	3	50	28	1122	21	0.42	3.2
				133	27	230	21	0.16	5.1

SSTR3	TZ	12 hr growth	3	40	28	742	35	0.88	14.3
				134	25	121	35	0.26	17.4
HTR6	TZ	1 hr growth	3	41	36	1132	19	0.45	2.5
HTR6	TZ	3 hr growth	3	50	30	729	14	0.28	2.0
				127	21	182	14	0.11	4.1
HTR6	TZ	6 hr growth	4	35	32	876	19	0.56	4.7
				125	24	341	19	0.15	4.7
HTR6	TZ	12 hr growth	3	45	28	1134	23	0.50	4.0
				124	20	534	23	0.19	4.2
SSTR3	TZ	10 µM sst	3	47	40	942	19	0.40	3.0
				137	19	143	19	0.14	7.9
SSTR3	TZ	100 µM sst	1	22	26	1476	22	1.0	8.8
SSTR3	TZ	100 µM sst + 30 µM PS2	2	48	28	393	23	0.48	5.4
SSTR3	TZ	100 µM sst + 30 µM PS2	2	131	24	138	23	0.18	7.8
β-tubulin	Flagella	Normal	6	11	27	1960	21	1.91	10.6
				75	37	1058	21	0.28	2.6
PKD2	Flagella	Normal	2	63	25	185	54	0.86	25.7
				134	17	125	54	0.4	27.8
IFT54	Flagella	Normal	2	79	36	1330	17	0.22	1.8

KAP	Flagella	Normal	2	29	12	164	25	0.86	10.9
				77	28	147	25	0.32	7.2

3.3 Estimating the error via simulation on the 3D transport routes in SPEED microscopy

While we had reproducibly determined a variety of previously-characterized and -uncharacterized 3D transport routes in primary cilia, it was important to have a more quantitative understanding of the error on those 3D transport routes. In the following sections, we demonstrated our 2D-to-3D transformation algorithm in more well-controlled systems as well as in simulation, which allowed us to more precisely control several of the main factors affecting 3D transport route error and quantify their effects.

3.3.1 Experimental validation of 2D-to-3D transformation process in several systems: glass nanocapillary tube, nuclear pore complex, primary cilia, and microtubules

Since the 2D-to-3D transformation algorithm requires radial symmetry, we first used an ideal artificial glass nanocapillary to test the algorithm's accuracy when coupled with SPEED microscopy for data acquisition. The glass nanocapillaries (GNCs) were fabricated using laser-assisted capillary-pulling of quartz micropipettes which can generate pore diameters ranging from 20 nm to 300 nm. The dimensions of GNCs used in this study were determined by Helium scanning transmission ion microscopy to have an inner radius ~35 nm (Zweifel et al., 2016, Luo et al., 2017). With that parameter in mind, the dimensions of the GNC were re-measured by determining the 3D density map of 1-nM Alexa Fluor 647 that was pumped into the inner lumen of the GNC. After thousands of 2D spatial locations for Alexa Fluor 647 were collected with a single-molecule localization precision of ≤ 5 nm, the final 3D density map revealed a radius of 37±2 nm (a width at the two standard deviations of the Gaussian function), agreeing well with the 35-nm inner radius of the GNC imaged by helium ion microscopy with a reproducibility rate of 100% and a route localization error of 0.25 nm determined by inputting the experimental parameters (localization precision, radius, and number of localization) into the following simulations (Figure 14 A-C). Reproducibility rate was defined as the number of experimental peak fittings from all simulation runs that fell within an acceptable range of the ground truth peak location while route localization error was defined as the standard deviation of the experimental peak fittings from all the simulation runs (Figure 15E). This value is a measure of the error one may expect given a certain level of localization precision and single molecule localizations in one trial during an experiment.

After the inner diameter of the GNC was confirmed by the application of the 2D-to-3D transformation algorithm, we moved to two different macromolecular trafficking in sub-cellular organelles: Importin β1 (a major transport receptor in nucleocytoplasmic transport (Weis, 2003, Fried and Kutay, 2003, Fahrenkrog and Aebi, 2003) moving through the NPC and externally-labeled SSTR3 (a major transmembrane protein in primary cilia (Händel et al., 1999, Sharma et al., 1996, Florio et al., 2003) on the surface of primary cilia on the ciliary shaft. Previously, our lab has revealed that

Importin β1 assists the movement of protein cargo via interactions at the periphery of the NPC, a selective gate between the cytoplasm and nucleus (Ma et al., 2012, Ma and Yang, 2010). In this analysis, we present a total of 450 2D spatial locations with a single-molecule localization precision of < 10 nm at the NPC's scaffold region for Importin β1 within the NPC. The corresponding 3D density map clearly shows a high density region for Importin β1 at 23±1 nm along the radius of the NPC with a 100% reproducibility rate and a route localization error of ~1 nm (Figure 14 D-F). Similarly in primary cilia, SSTR3 was externally-labeled with Alexa Fluor 64740 and tracked using SPEED microscopy along the length of primary cilia, a cellular projection ~125 nm in radius determined by electron microscopy (EM) (Marshall and Nonaka, 2006, Yang et al., 2015, Scholey and Anderson, 2006). Agreeing well with the EM-determined diameter, 260 externally-labeled SSTR3 molecules with a single-molecule localization precision of ≤ 10 nm were determined to have a high density region at 127±2 nm along the radius of primary cilia, with a 97% reproducibility rate and a route localization error of ~2 nm (Figure 14 GI-L).

Lastly, to test whether the 2D-to-3D density transformation algorithm could be applied beyond SPEED microscopy, we measured the diameter of microtubules (Borisy et al., 2016, Nogales, 2001, Desai and Mitchison, 1997), in which tubulins were labeled by a primary and secondary antibody conjugated to Alexa Fluor 647 and then imaged by 2D-STORM (Sage et al., 2015). By converting the published 2D super-resolution image for a microtubule (112 single-molecule locations) into its 3D structure by our 2D-to-3D transformation algorithm, we determined the diameter of microtubules in this specific sample to be 64±1 nm with a reproducibility rate of 90% and a route localization error of ~1 nm (Figure 14 J-L). This result agrees well with previous determinations by using EM (Nogales, 2001) and 3D super-resolution microscopy (Huang et al., 2009).

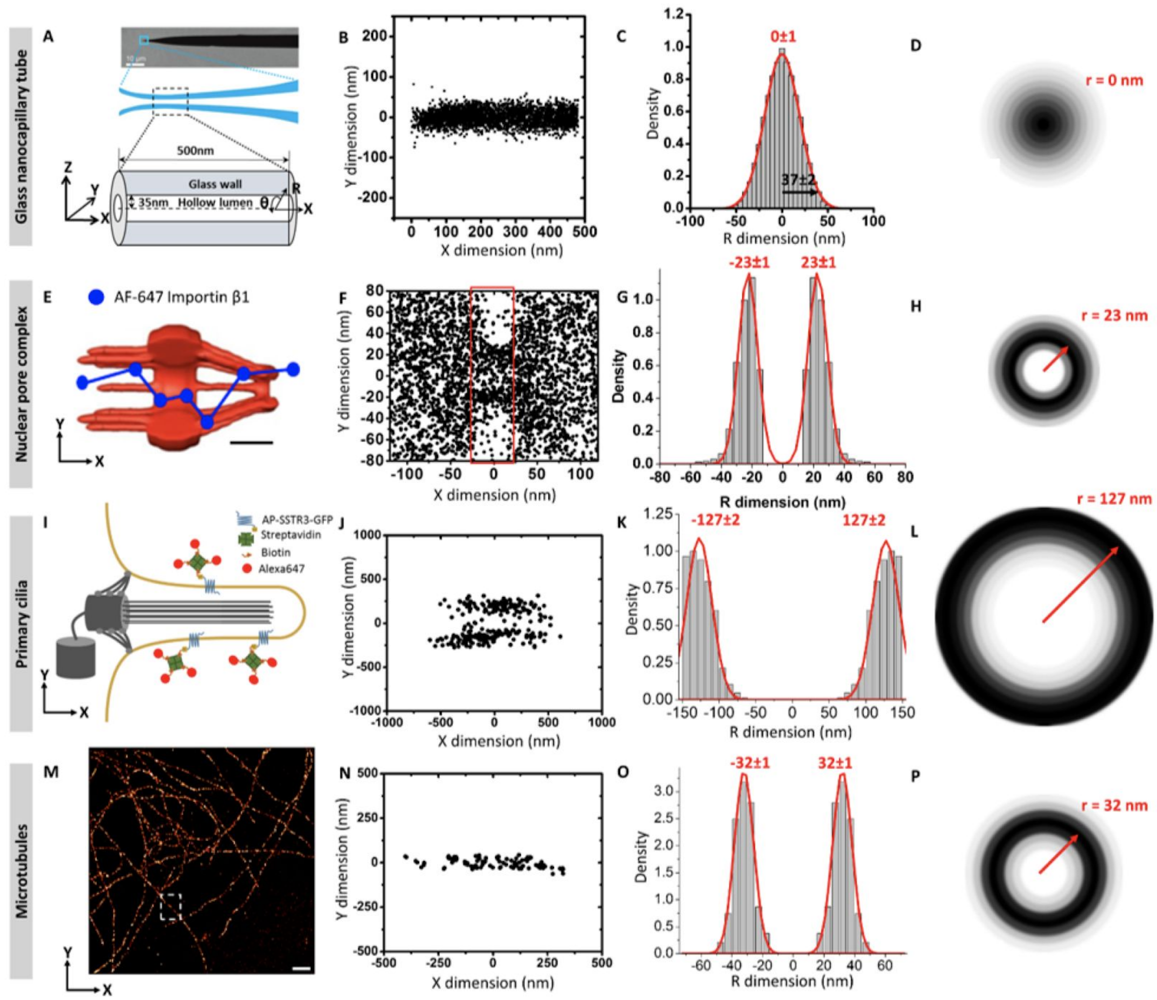


Figure 14. ***3D density maps accurately reconstruct the locations of labeled molecules in various structures.*** **(A)** Schematic of the glass nanocapillary tube with dimensions of 35 nm inner radius. **(B)** x and y dimensional single molecule data from the tracking of Alexa Fluor 647 inside the glass nanocapillary tube. **(C)** 3D density map of (B) showing the width at 2 standard deviations from the mean is ~ 37 nm. **(D)** Radial density map from (C). **(E)** Schematic of the nuclear pore complex with a model trajectory for Alexa Fluor 647-labeled Importin β1. Scale bar = 50 nm. **(F)** x and y dimensional single

molecule data from the tracking of Alexa Fluor 647-labeled Importin β1 in the nuclear pore complex. **(G)** 3D density map of (F) showing the peak fitting of the Importin β1 transport route is ~ 23 nm. **(H)** Radial density map from (G). **(I)** Schematic showing the Alexa Fluor 647 externally-labeled SSTR3 in the shaft of a primary cilium. **(J)** x and y dimensional single molecule data from the tracking of externally-labeled SSTR3 in the primary cilium. **(K)** 3D density map of (J) showing the peak fitting of the SSTR3 transport route is ~ 127 nm. **(L)** Radial density map from (K). **(M)** 2D STORM data showing the reconstructed super-resolution image of tubulin labeled with a primary and secondary antibody attached to Alexa Fluor 647. Scale bar = 1 μm. **(N)** x and y dimensional single molecule data from section of 2D STORM data in the dashed white box from (M). **(O)** 3D density map of (K) showing the peak fitting of the labeled tubulin from (N) to be ~ 32 nm. **(P)** Radial density map from (O).

3.3.2 Monte Carlo simulation demonstrates the parameters required for accurate 3D density map reconstruction

Next, we used Monte Carlo simulation (Mooney, 1997, Mahadevan, 1997) to demonstrate that two critical parameters, the single-molecule localization error of targeted molecules (Thompson et al., 2002) and the number of single-molecule locations (Betzig et al., 2006, Thompson et al., 2002), determine the reproducibility of obtaining accurate 3D super-resolution information in biological structures with rotational symmetry. Typically, the spatial localization of individual targeted molecules labeled with fluorophores or fluorescent proteins are determined with non-zero localization error because of, primarily, background noise and limited photon collection in real experiments (Thompson et al., 2002).

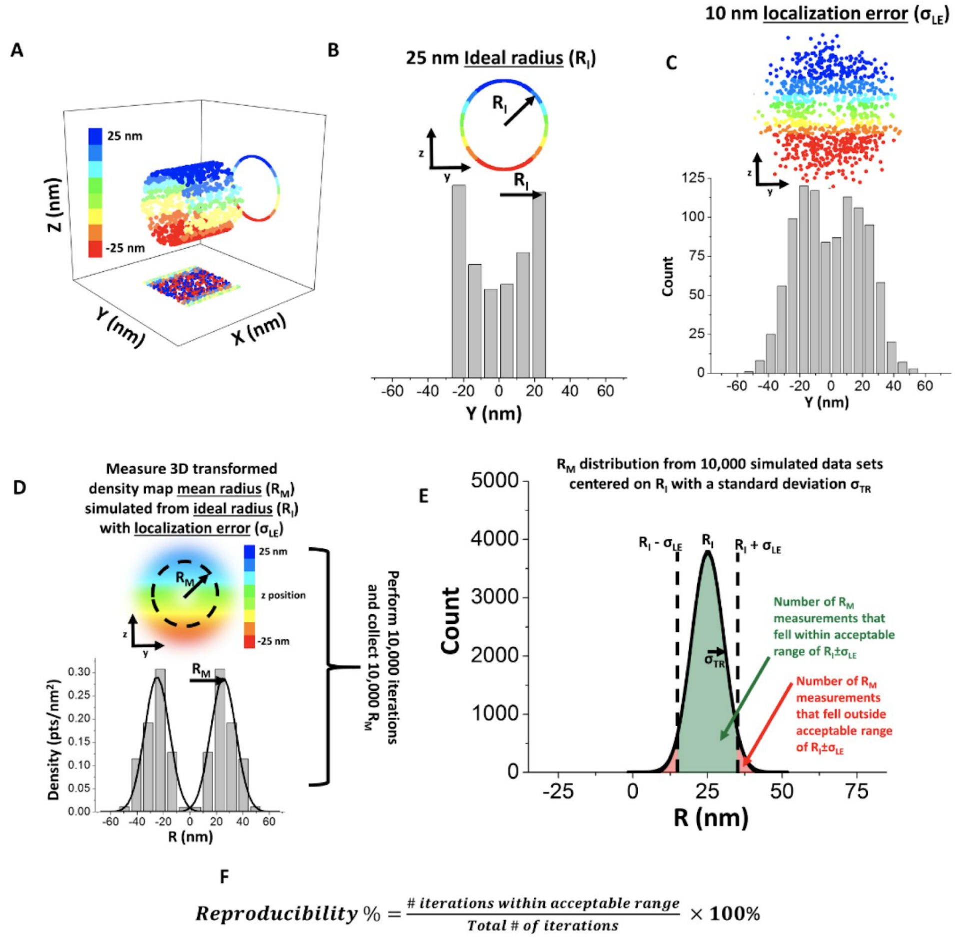


Figure 15. ***SPEED microscopy and 3D transformation reproducibility percentage using a simulation-based approach.*** For any given set of simulated data, the bin size is varied from 1 nm to the precision that is 10 nm in this example. **(A)** Data sets were simulated in three dimensions. Color bar indicates z position of the simulated points. **(B)** Each data set was simulated first with an ideal 25-nm radius (RI). **(C)** Subsequently, a localization error (σLE) of 10 nm was added to each point. (C) Using a 5-nm bin size for demonstration, the 2D histogram of the simulated data set with a 25-nm radius and 10-nm localization precision was determined. **(D)** 10,000 data sets were simulated with an ideal 25-nm radius (RI) and a localization error (σLE) of 10 nm. The resultant 3D histograms were then each fitted with a Gaussian function to localize the mean position of each peak, which is designated as the mean radius RM. **(E)** The histogram for all the RM values was determined and the number of simulated data sets that fell within the acceptable range of RI±σLE were counted. The acceptable range of RI±σLE was chosen because, in principle, the Rayleigh criterion limited the resolution of any single 3D histogram to the spread of that distribution, which was due to the simulated localization error (σLE). After

10,000 simulations, the histogram for RM values converges on the mean (RI) from which they were originally sampled, while the spread of the RM histogram (σTR) converges on a value that is due to the number of simulated points in each distribution and simulated localization error. **(F)** Reproducibility percentage was defined as the number of RM values that fell within the acceptable range of RI±σLE divided by the total number of simulated data sets and multiplied by 100%.

To mimic typical transport routes of proteins in the NPC or primary cilia, our Monte Carlo simulations were performed where varying numbers of single-molecule locations were randomly simulated on an ideal radius (R_I) (Figure 15 A-B). Then, single-molecule localization error (σ_{LE}) was added to R_I by sampling an error value from a normal distribution with a standard deviation of σ_{LE} (Figure 15 C). Subsequently, the 2D-to-3D transformation algorithm was performed on only the y dimensional data of the simulated single-molecule localization distribution around R_I to model the loss of z dimensional information during the 2D microscopy projection process. The peak position of the transformed 3D density histogram was then determined by Gaussian fitting to produce a measured mean radius ($R_{M)}$ which may deviate from the R_I due to limited number of simulated locations and non-zero single-molecule localization error (Figure 15 D). We conducted 10,000 iterations of this process and obtained 10,000 RM values, in which the mean of the R_M values converges on R_I as expected (Figure 15 E). With this many iterations, the error of our measurements and the resulting interpretations never exceeded 0.1 nm. To quantify how reproducible a single experimental data set is, we set out to determine how many individual R_M values from the whole distribution of R_M fell within an acceptable range of R_I. The acceptable range was defined as the $R_I \pm \sigma_{LE}$

because, in principle, any single R_M value can only be accurately localized within the range of approximately two standard deviations of its Gaussian fitting, similar to the concept of resolution stated by the Rayleigh Criterion (Ram et al., 2006) (Figure 15 F). We expect that a high number of simulated single-molecule locations or low single-molecule localization errors would increase the number of iterations that fall within the acceptable range, thus resulting in a high reproducibility rate. To determine the reproducibility rate, we found that two critical steps should be correctly followed first in the process: first, optimize the bin size for each set of simulation parameters. This is accomplished by determining the smallest bin size that produces no statistical difference by Chi square analysis between the original 2D histogram and the back-calculated 2D histogram obtained by multiplying the 3D density histogram by the corresponding area matrix (Figure 16). The second step is to account for the slightly increased sensitivity of the inner bins of the area matrix when determining the accurate R_M peak fitting. This phenomenon is due to the fact that the area of the inner bins is slightly smaller compared to the area of the outer bins making them more sensitive to noise (Figure 17).

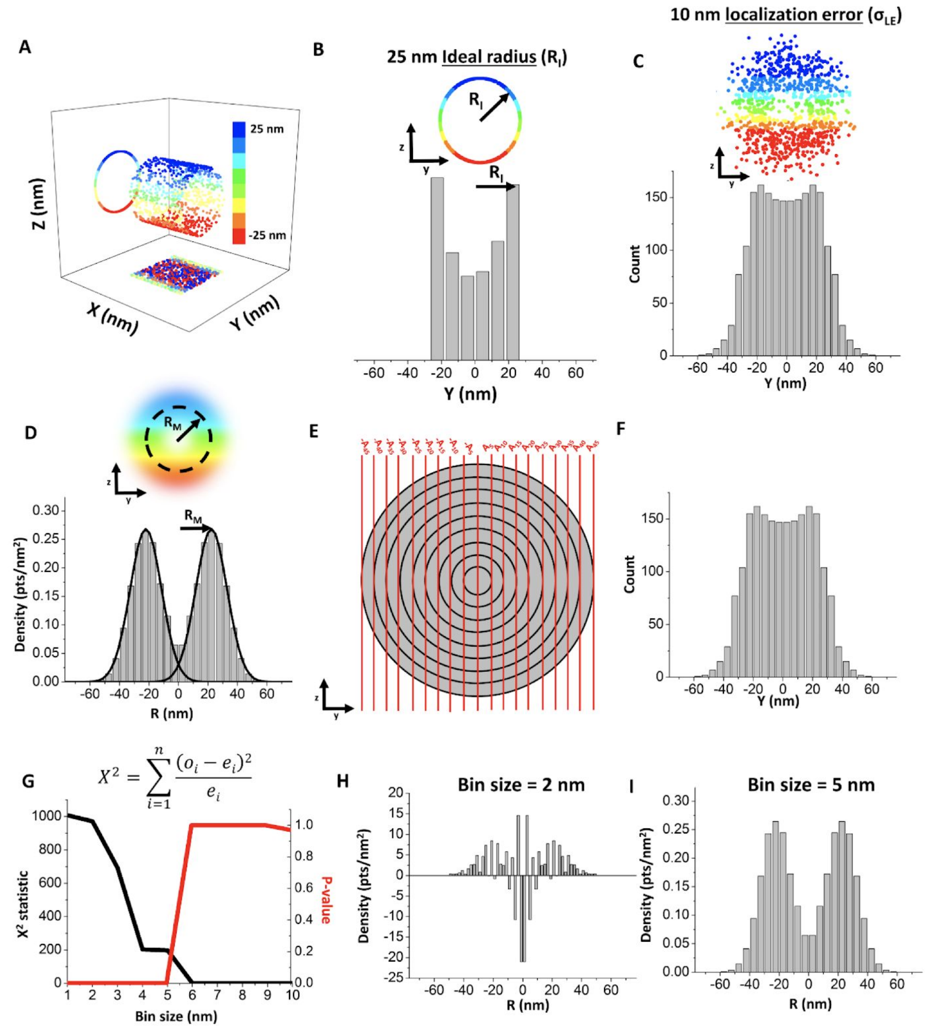
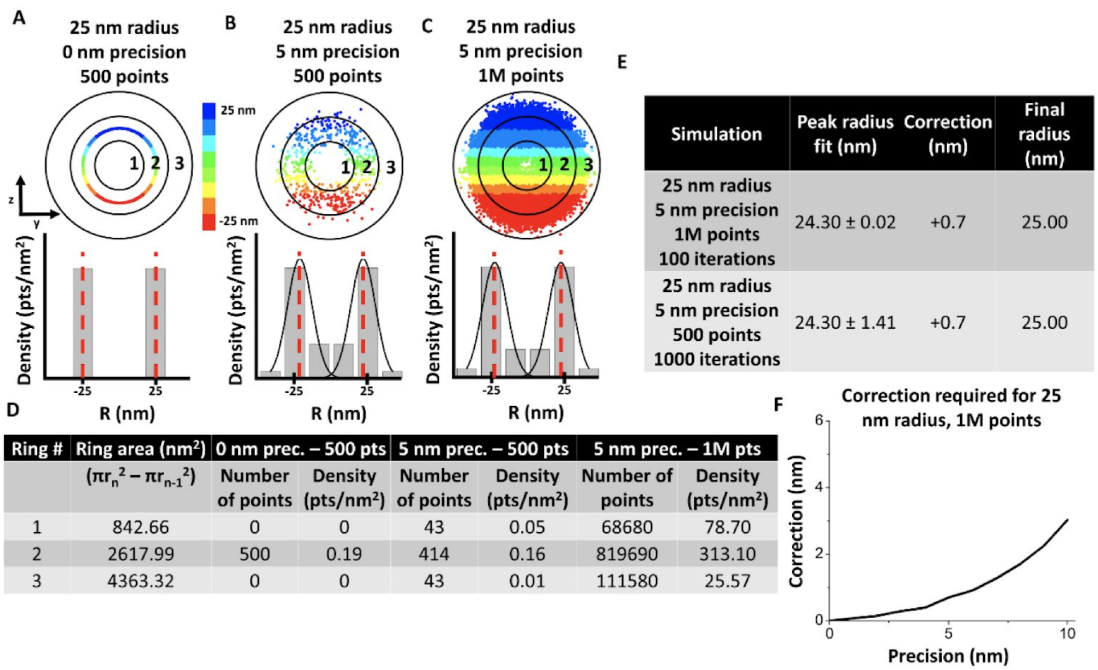


Figure 16. Optimal bin size determination using Chi-square error analysis. For any given set of simulated data, the bin size is varied from 1 nm to the precision that is 10 nm in this example. **(A)** Data sets were simulated in three dimensions. Color bar indicates z position of the simulated points. This representative data set contains 1,000 single molecule locations. **(B)** Each data sets was simulated first with an ideal 25-nm radius (R_I). **(C)** Subsequently, a localization error (σ_{LE}) of 10 nm was added to each point. Using a 5-nm bin size for demonstration, the 2D histogram of the simulated data set with a 25-nm radius and 10-nm localization precision was determined. **(D)** The 3D density histogram was then obtained via the 2D to 3D transformation algorithm and the peaks were fit with Gaussian distributions. **(E)** The 5-nm bin-size area matrix was calculated and multiplied by the 3D density distribution in (D) to reconstruct the 2D distribution (any negative values in the density distribution were set to zero) as shown in F. **(F)** The

values of the reconstructed 2D distribution were then compared bin-by-bin to the original 2D distribution (as shown in C) using the Chi-square analysis equation where 'o' refers to the observed histogram values in (F), 'e' refers to the expected histogram values in (C), 'i' refers to the bin, and 'n' refers to the total number of bins with histogram values in them. **(G)** The Chi-square test statistic and p-value were then plotted across the potential bin size values. A p-value ≤ 0.01 indicates that the 2D histograms in (C) and (F) are different from each other, suggesting the lack of enough data to allow sufficient sampling from each bin. On the other hand, a p-value > 0.99 suggests that the 2D histograms are statistically similar and likely have enough data points to accurately measure the value in each bin. Chi-square analysis was performed 1,000 times for each set of simulation parameters. **(H)** A 3D density histogram displaying large errors (negative density values) due to a bin size selection that was too small. Such errors occur when the single molecule density is not uniform throughout a given radial bin. This non-uniformity is due to under-sampling. **(I)** A 3D density histogram displaying no errors and a clear Gaussian distribution due to a properly selected bin size. A larger bin size can smooth the effects of under-sampling to a certain degree.



Simulation	Peak radius fit (nm)	Correction (nm)	Final radius (nm)
25 nm radius 5 nm precision 1M points 100 iterations	24.30 ± 0.02	+0.7	25.00
25 nm radius 5 nm precision 500 points 1000 iterations	24.30 ± 1.41	+0.7	25.00

Ring #	Ring area (nm^2)	0 nm prec. – 500 pts		5 nm prec. – 500 pts		5 nm prec. – 1M pts	
	($\pi r_n^2 - \pi r_{n-1}^2$)	Number of points	Density (pts/nm^2)	Number of points	Density (pts/nm^2)	Number of points	Density (pts/nm^2)
1	842.66	0	0	43	0.05	68680	78.70
2	2617.99	500	0.19	414	0.16	819690	313.10
3	4363.32	0	0	43	0.01	111580	25.57

Figure 17. ***Sensitivity of inner bins necessitates slight correction of peak position during simulation.*** **(A), (B), and (C)** Simulated single molecule data and corresponding

3D density histogram for simulation with 25 nm radius, 0 nm localization error, and 500 points. Red dashed lines indicate mean peak fitting. **(B)** Simulated single molecule data and corresponding 3D density histogram for simulation with 25 nm radius, 5 nm localization error, and 500 points. Red dashed lines indicate mean peak fitting. **(C)** Simulated single molecule data and corresponding 3D density histogram for simulation with 25 nm radius, 5 nm localization error, and 1 million points. Red dashed lines indicate mean peak fitting. **(D)** Table showing the calculation to obtain each bin of the 3D density histogram. **(E)** Table showing that even one million points does reconstruct a precise 25 nm peak fitting due to the fact that the inner radial bins have smaller area and are slightly more sensitive to changes in density. **(F)** Correction required for each precision up to 10 nm for a 25 nm radius. This correction process was performed before each simulation to accurately localize the RM density peak and correlate it to the ideal RI from which the data was simulated.

3.3.3 Reproducibility of obtaining accurate 3D density map reconstruction

First, to test the effects of single-molecule localization error on the final R dimensional peak fitting obtained for the 3D transformed density histogram, Monte Carlo simulations were performed with an R_I of 25 nm, data point number of 1,000,000 to control point number and represent the ideal case, and σ_{LE} ranging from 0 to 30 nm (Figure 18 A-D). In principle, as the single-molecule localization error becomes excessively large, the peaks in the 3D density map will become heavily overlapped on the y dimensional axis and, subsequently, the radial axis after the transformation algorithm. This will obscure the peak at R_I and make it indistinguishable. A ratio between the error of a bimodal Gaussian fitting and the error of a single Gaussian fitting is used to determine the indistinguishable overlap (Figure 18 E-H). As shown in Figure 18 I, a series of tests indicate that the bimodal fitting error becomes much larger than the single peak fitting error beyond a 21-nm localization error. This suggests that the experimental localization error cannot exceed 21 nm for any structure containing a transport route with a radius of 25 nm. While, this is smaller than the theoretical single-molecule localization error of 25 nm predicted by the Rayleigh Criterion, mainly due to the aforementioned sensitivity of the inner bins of the area matrix in this 2D-to-3D transformation process (Figure 17).

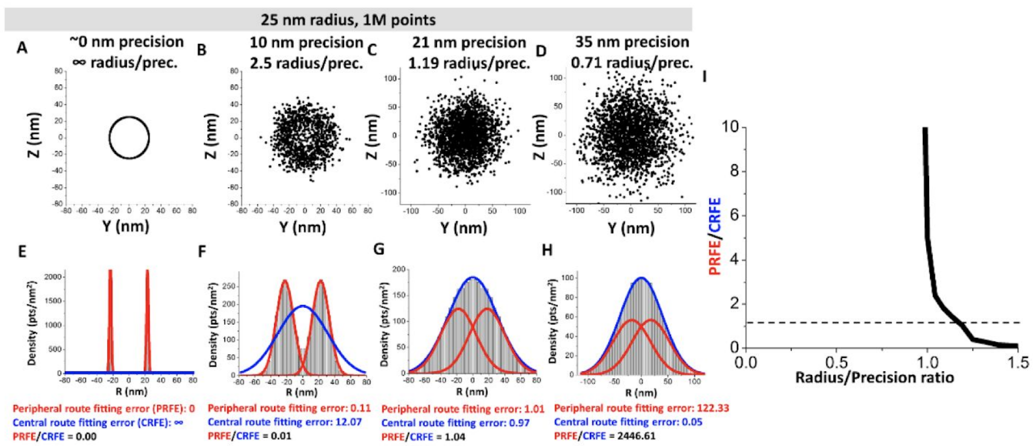


Figure 18. ***Varying the simulated precision shows the resolution limit of the 3D transformation algorithm.*** **(A), (B), (C), and (D)** 1,000,000 locations were simulated on an ideal radius (RI) of 25 nm with localization errors (σLE) of 0, 10, 21, and 35 nm respectively. Scatter plots were down-sampled to 2,000 locations for visualization. **(E), (F), (G), and (H)** The corresponding 3D transformed density histogram of each simulation from (A), (C), (E), and (G) respectively. To determine whether the density peaks could be distinguished after a given localization error had been added to the ideal data, the ratio between the fitting error of a bimodal Gaussian distribution (PRFE) and a single Gaussian distribution was used (CRFE). If the bimodal fitting error was less, then the peaks in the 3D transformed distribution can likely be distinguished. If the single peak fitting error was less, the localization error was too great and the two peaks were indistinguishable. **(I)** Up to 21 nm, the PRFE was much less than the CRFE. Above 21 nm, the CRFE was much less than the PRFE. Therefore, the maximum localization error allowed to distinguish a 25 nm ideal radius is ~21 nm. The results are generalized as precision/radius.

Moreover, the above results can be generalized by using a radius/precision (R/P) ratio to estimate whether the transport route can be distinguished before computational simulations or real experiments. As shown in Figure 18 C,G, the threshold case of a R/P ratio of 1.19 (25 nm/21 nm) presents the minimally distinguishable transport route in the R dimension, corresponding to a separation of ~68% of the single-molecule density around the radius. As mentioned above, 1.19 represents the case where the radius is approximately equal to the precision or standard deviation of the transport route; thus, separation of the radial distribution by ~1 standard deviation results in ~68% of single-molecules density separation. Meanwhile, when the R/P ratio is ≥ 2.0, a much higher degree (correspondingly ≥ 95% of single-molecule density) of peak separation and a well distinguished transport route can be achieved (Figure 18 B, F).

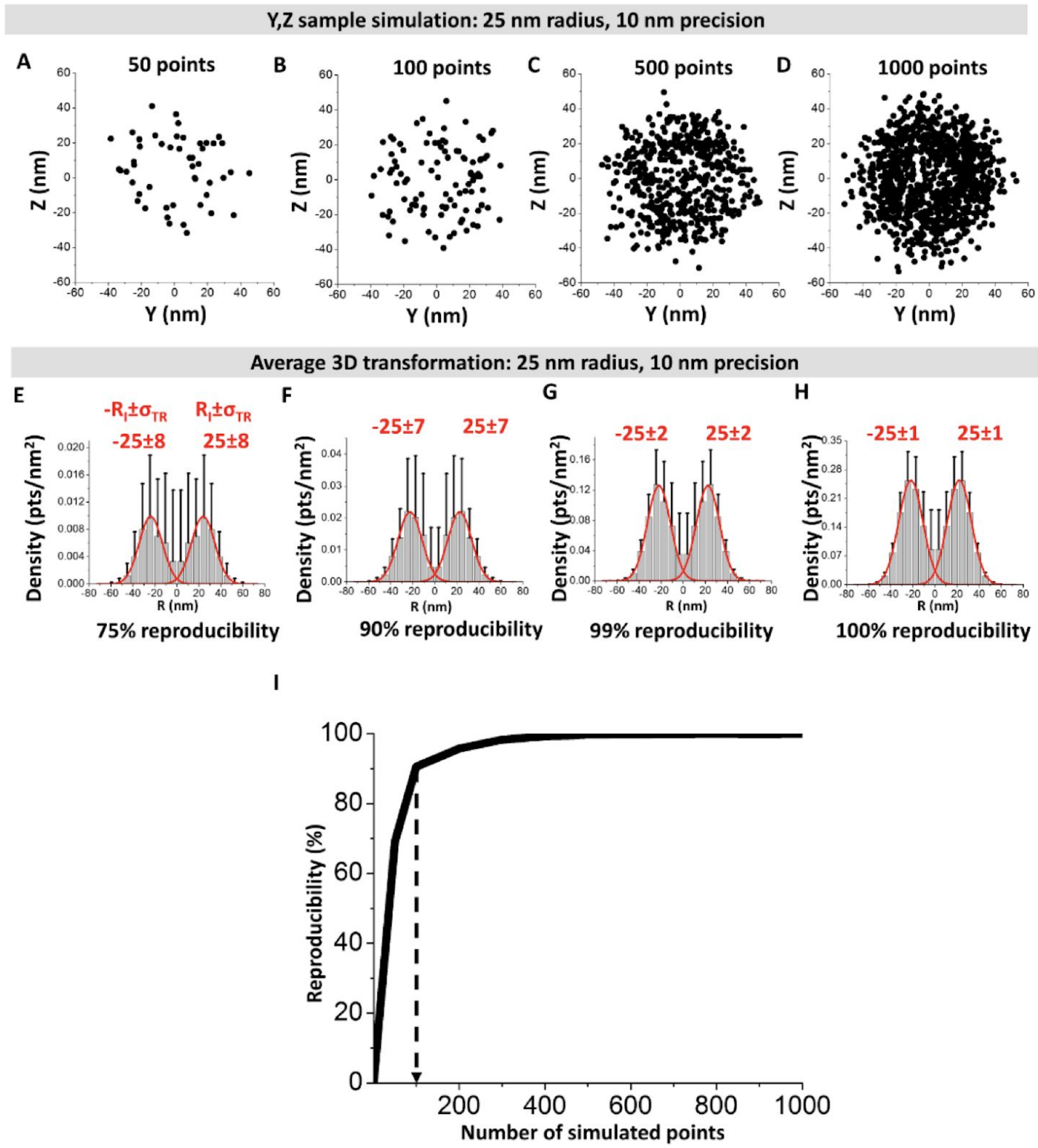


Figure 19. ***Varying the number of simulated points shows the effect of sampling error on the peak fitting of the 3D transformed data.*** **(A), (B), (C), and (D)** 50, 100, 500, and 1,000 locations were simulated on an ideal radius (R_I) of 25 nm with localization error (σ_{LE}) of 10 nm. **(E), (F), (G), and (H)** The average 3D transformation of 10,000 simulated data sets for each number of simulated locations. Error bars indicate standard deviation of each histogram bin value while $R_I \pm \sigma_{TR}$ indicates the average peak fitting ± the standard deviation of the peak fittings. That is, the standard deviation (σ_{TR}) of the R_M distribution outlined in Figure 15. The reproducibility percentage, the number of peak fittings that fell within the localization error, is shown beneath each 3D transformed

density histogram. **(I)** The plot of reproducibility percentage for 10,000 iterations of each number of points up to 1,000. The simulation parameters were the same as (A), (B), (C), and (D) except for the varying point number.

Next, using the R/P ratio of ≥ 2.0, we sought to determine the effects of the quantity of single molecule locations on re-constituting an accurate 3D transformed structure. To accomplish this, Monte Carlo simulations were performed with an RI of 25 nm, σ_{LE} of 10 nm (the R/P ratio of 2.5), and data point numbers ranging from 50 to 1,000. Representative simulations are shown for 50, 100, 500 and 1,000 points, and the corresponding reproducibility percentages were calculated for each point number after 10,000 iterations (Figure 19 A-H). Remarkably, only 100 and 350 points are sufficient to achieve 90% and 99% reproducibility respectively (Figure 19 I). It is highly feasible to obtain 100-1,000 points experimentally, although the number of points is higher than the minimum Nyquist Sampling theorem estimation of 38 single-molecule locations (Thompson et al., 2010). This is resultant from a non-uniform distribution of locations through the area matrix, which causes deviations from the requirement of radial symmetry. Stated in another way, it is necessary to achieve a reasonable level of radial bin uniformity – through sufficient collection of single molecule localizations – to accurately reconstruct the 3D density distribution. Of note, we found that 1,000 single-molecule locations can already result in an accuracy of 1 nm in obtaining the mean location of the 3D transformed density map (Figure 19 D,H,I). In a similar way, we can also determine the accuracy of other parameters of the 3D transformed density map, such as the width of the distribution (Figure 20). In theory, given enough single-molecule data, the accuracy could be unboundedly small as long as the R/P ratio is above 2. Finally, in some cases, we noticed that, if the width of transport route ($2\sigma_w$, the width at one standard deviation of its Gaussian distribution) is significantly bigger than the single-molecule localization precision, the above R/P ratio should be modified as R/P_W ($P_W = \sigma_w$) before determining the final radius of the transport route and the number of single-molecule locations needed for a high reproducibility rate. This discrepancy between the single molecule localization precision and σ_w is due to a non-negligible width of the biological transport route as is the case in Figure 15 I-L where the localization precision of SSTR3 molecules was 10 nm but the width of the transport route was 35 nm.

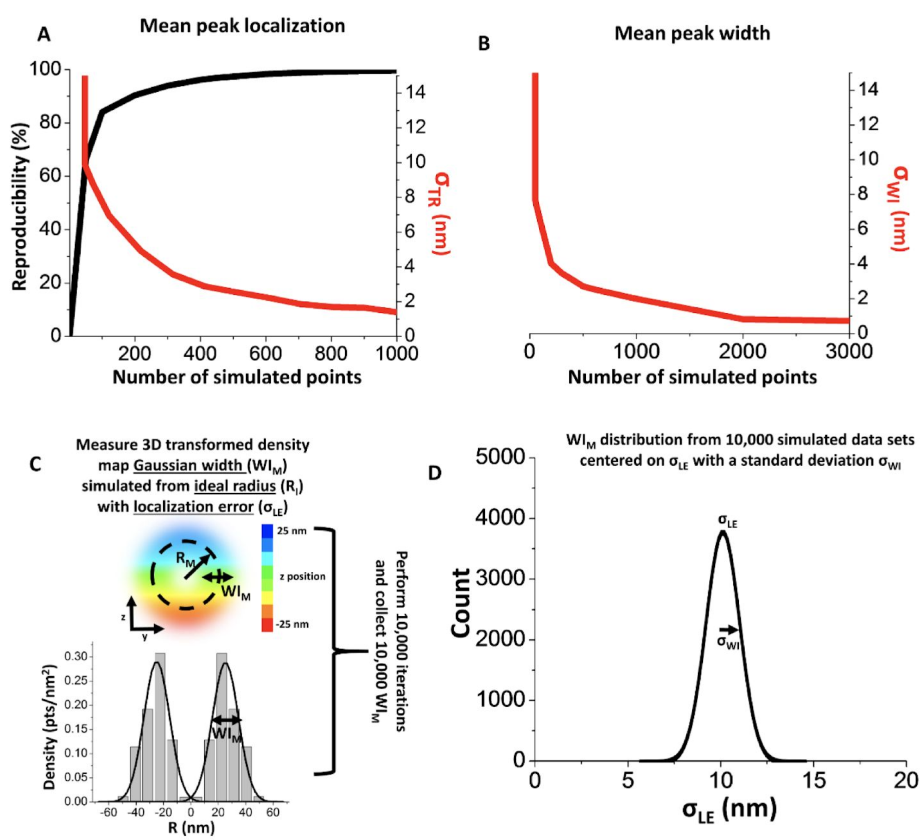


Figure 20. ***The accuracy of other parameters of the 3D density probability distribution may be estimated as well.*** **A)** Plot of reproducibility percentage across various simulated point numbers for the mean peak localization from Figure 19 superimposed on the plot of σ_{TR}, the corresponding nanometer value of mean 3D density peak localization error from which the reproducibility percentage was derived. **B)** Plot of σ_{WI}, the nanometer value of the mean 3D density peak Gaussian width. **C) and D)** The simulation was performed by the same method as Figure 15 with the exception that the standard deviation of the Gaussian widths (σ_{WI}) was retained for analysis.

3.3.4 Reliability of the 2D-to-3D transformation algorithm to distinguish multiple transport routes in the NPC

Third, there are experimental instances where there may be a mixture of transport modes that a protein may use as it traverses a rotationally symmetric organelle. For example, in the NPC, our previous work has shown that passively diffusing molecules <40 kDa pass through the central axis of the NPC while transport receptors such as Importin β1 pass through the NPC closer to the scaffold region, ~23 nm along the NPC's radial cross-section (Ma et al., 2012, Ma and Yang, 2010). As the transit of molecules is studied, it is conceivable that they could traverse the NPC at one or both of these transport routes. To test the ability of the 2D-to-3D transformation algorithm to distinguish between these potential cases, we simulated datasets from the three distinct distributions [peripheral only, central only, and bimodal (peripheral + central)], then compared the reconstructed distributions to the ground truth input distributions via a categorization routine (Figure 21) that calculates the sum of the absolute-valued residuals (SAR) between each trial and the ground truth input distributions (Figure 21 E). With these measures used in our simulations, ~100 locations allowed us to reach >90%

successful categorization for peripheral and central distributions when the single-molecule localization precision was set to 5 nm. A bimodal distribution requires more localizations for accurate categorization, but still only ~200 locations were sufficient to achieve >90% successful categorization (Figure 21 G-P). When the simulation was applied to the 10-nm localization precision case which is typical of experimental cases, ~100-300 points were sufficient to distinguish the three distributions with >90% success (Figure 21 Q). Of course, this approach has limitations. We can only assess the distribution against hypotheses and determine which hypothesis is the most likely result.

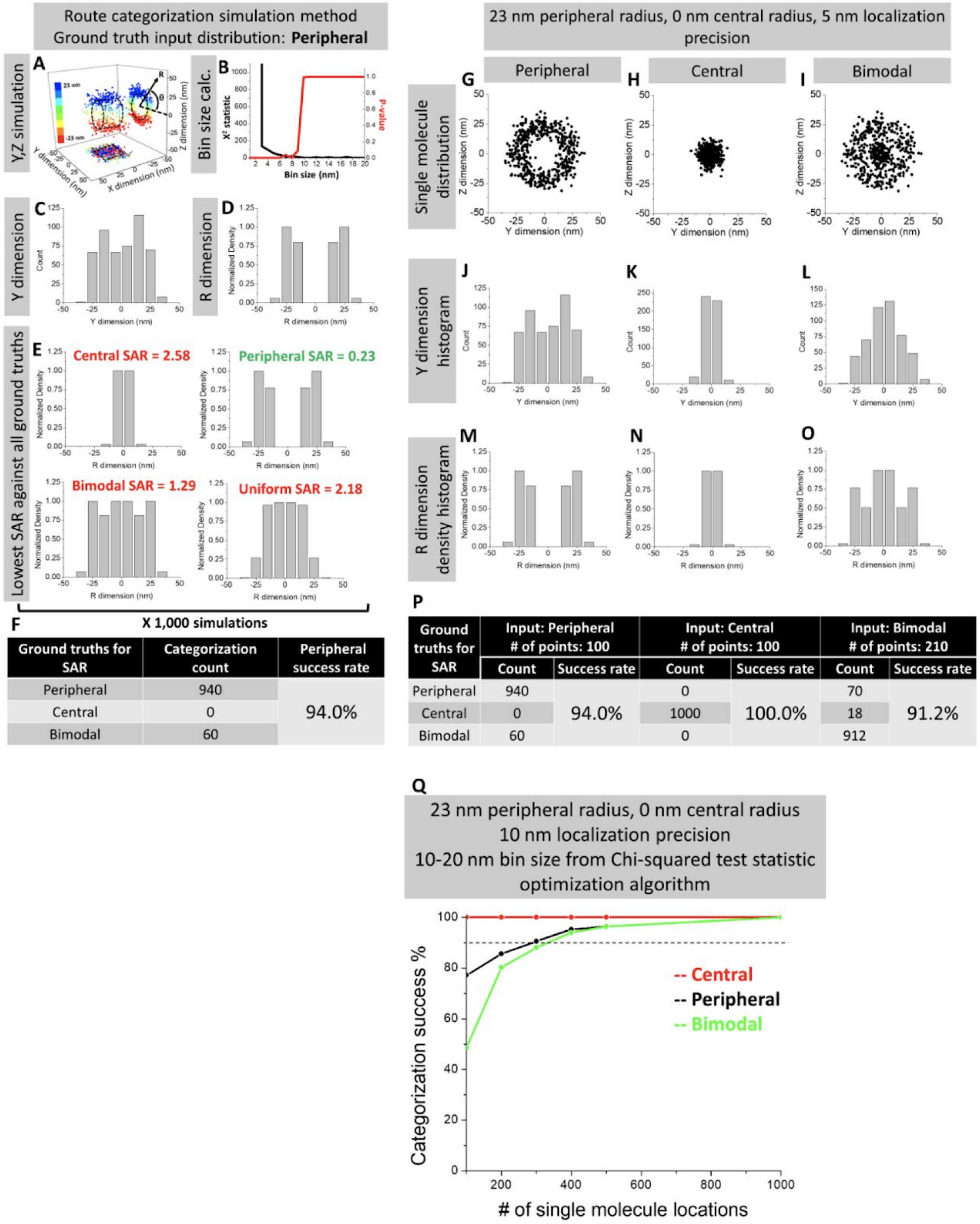


Figure 21. ***Route categorization simulation.*** **(A)** For each set of simulation parameters (distribution type, localization precision, peripheral radius, central radius, and number of single molecule locations), 1,000 datasets were generated in the Y, Z dimensions. Displayed here is a representative dataset from a peripheral distribution, localization precision of 5 nm, peripheral radius of 23 nm, central radius of 0 nm, and 500 single

molecule locations to clearly visualize the distribution from the Y,Z scatter data. Y,Z coordinates can also be transformed to analogous R,θ coordinates**.** **(B)** The bin size optimization algorithm was performed with the exception that the first bin with a p-value > 0.99 was selected. **(C)** The Y-dimensional histogram was obtained with the optimized bin size for every dataset. **(D)** The 2D-to-3D transformation algorithm was performed on every dataset. **(E)** The resultant 3D density histograms were compared via SAR to the central, peripheral, and bimodal ground truth distributions, which were obtained by simulating a dataset with all the same parameters except for 1,000,000 single molecule locations. The lowest SAR indicates which distribution a given dataset is most similar to and, thus, which ground truth distribution it is classified as. **(F)** The count of all classifications for the ground truth peripheral parameters set of 5 nm localization precision, 23 nm peripheral radius, 0 nm central radius, and 100 single molecule locations. **(G – I)** Sample datasets from simulation runs for the peripheral, central, and bimodal distributions respectively using the same parameters as (A). **(J – L)** Y dimensional histograms from (G – I). **(M – O)** 3D density histograms calculated from (J – L). **(P)** Resultant success rates for the different distributions. **(Q)** Simulation results for the peripheral, central, bimodal, and uniform distributions with parameters of 10 nm localization precision, 23 nm peripheral radius, 0 nm central radius, and 10-20 nm bin sizes obtained from optimization algorithm.

3.3.5 Effects of deviations from the radial symmetry on 3D density map reconstruction

In addition, we explored how deviations from the radial symmetry assumption at various rotation angles affected the final 3D probability density distribution. First, we looked at compression of the radial symmetry which may occur from a lateral force on the sub-diffraction biological structure of interest, using the NPC as an example. Currently, it is challenging to assess how radial symmetry of the NPC may be affected due to the possible distortion effects in the sample preparation technique for in vitro studies (Stoffler et al., 2003) as well as the natural distortion of the NPC in live cell imaging (Melčák et al., 2007), Beck et al have minimized these effects using cryo-fixation and visualized single NPCs using electron tomography which show a high degree of regularity (Beck et al., 2007). They found that natural distortion of the NPC's symmetry may have an approximate ratio of the long (a) and short (b) axis of 1.350 (Figure 22). Therefore, while holding the circumference of the biological structure constant, we varied the a/b ratio and the rotation of the structure in relation to the orientation of these axes. For the case of a 25 nm radius, 10 nm simulated localization precision, 1,000,000 simulated localizations, and an a/b ratio of 1.3 (Figure 22 A-H), we found that that the mean peak in the 3D density distribution can range up to 4.4 nm from the actual mean (Figure 22 I-L). Overall, the deviation from the a/b ratio of 1 grows linearly as the a/b ratio grows (Figure 22 M); however, variation in the biological range easily permits the distinction between peripheral and central routes, which our lab has previously shown (Ma et al., 2012, Ma and Yang, 2010). Second, we explored how labeling efficiency affected the ability to reconstruct an accurate 3D density map distribution.

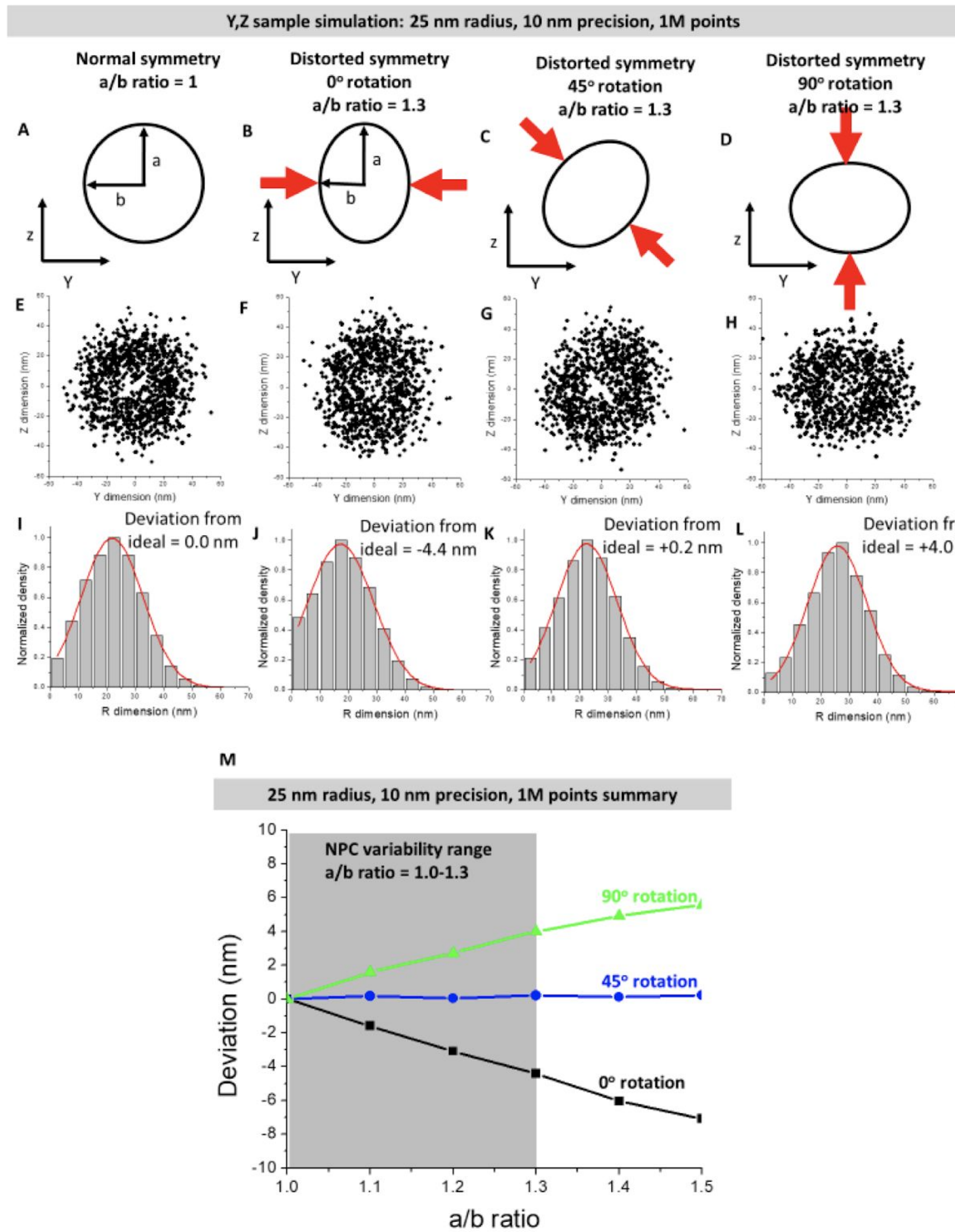
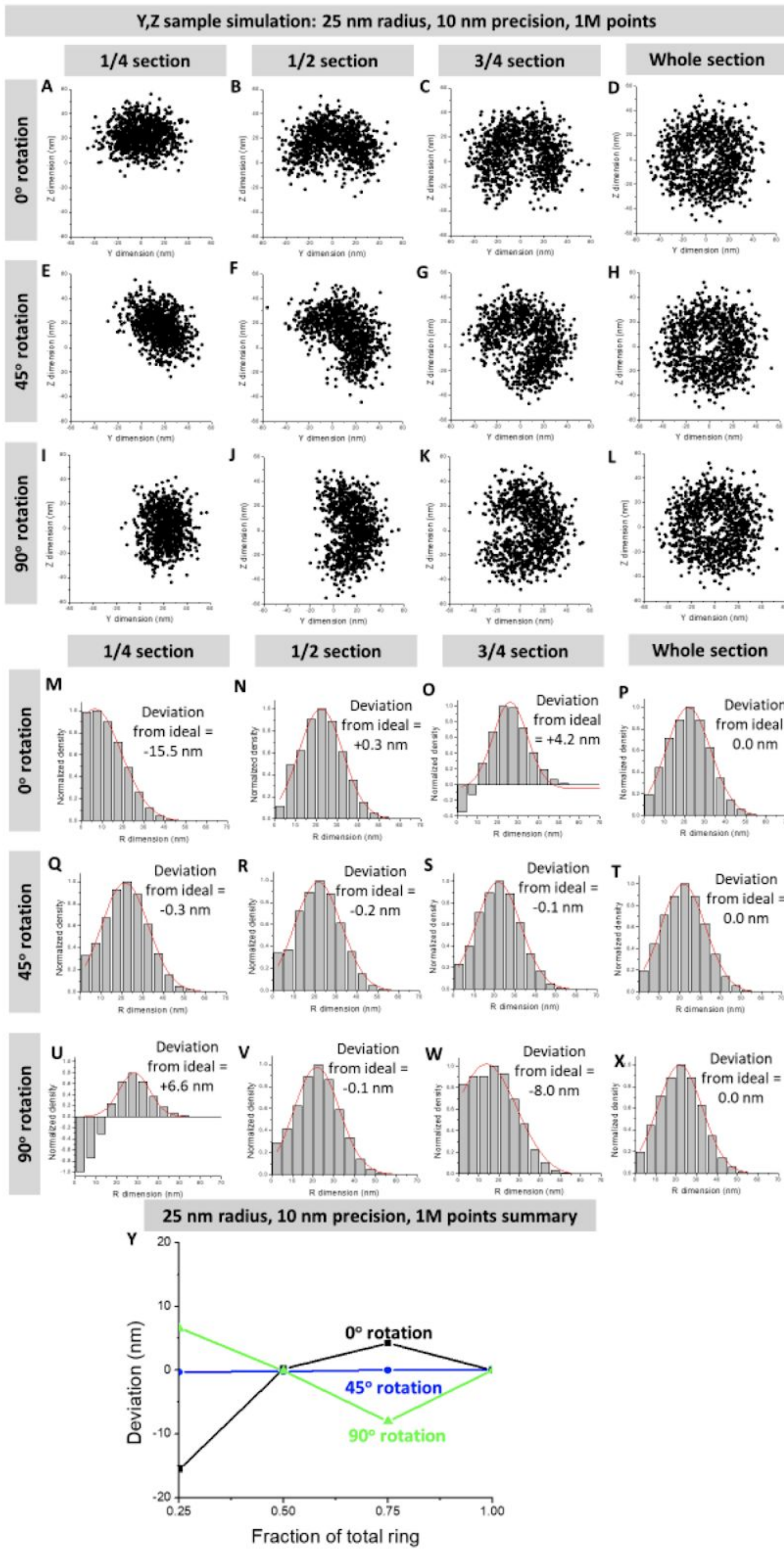


Figure 22. ***Varying the compression and rotation of the radial symmetry shows minimal effects at biological conditions.*** **(A)** Ideally, the radial symmetry is perfect with both the a and b axes being equal in the y and z dimensions. (**B)-(D)** In some cases, compression may occur where the a and b axes are unequal and rotated at any angle relative to the compression axis. (**E)-(H)** Single molecule data simulated from an ideal distribution of 25 nm radius, 10 nm localization precision, and an a/b ratio of either 1.0 or

1.3 at 0, 45, and 90 degree rotation. While 1,000,000 were used for the simulation, only 1,000 are shown here for ease of visualization. **(I)-(L)** 3D density distributions calculated from the data in E-H respectively with deviations from the ideal radius noted above the 3D density histogram. **(M)** Summary of a/b ratios ranging from 1.0-1.5 using the same simulation parameters as E-L.

For super-resolution techniques that used fixed cells and antibody labeling, this is an important factor to consider. Therefore, we looked at how unequal distributions around the radius at various rotation angles affected the final 3D probability density distribution. Indeed, there were negative values that occurred in some situations, typically an artifact of under-sampling or unequal labeling. While, in several of the distributions, the majority of the cases were unaffected for a 25 nm radius, 10 nm localization precision, 1,000,000 localizations, and various sections missing from the distribution at different rotation angles (Figure 23 A-X). This robustness is likely caused by the z-averaging that occurs during the microscopy projection process and the y-averaging that occurs during our data analysis process. The results are summarized in Figure 23 Y. While these simulations represent the deviations that may occur from any single experimental dataset, typically we average at least 5 such datasets to minimize the effects of these deviations to our results and have not seen significant deviations in overall distribution type. If such a case were to occur, it is advisable to report the different classes of distributions that may arise separately so as not to improperly average the results. However, the NPC structure has been shown to have great regularity (Beck et al., 2007). Furthermore, it is advisable, as we do in our experiments, to use a fluorophore-labeled nucleoporin as a marker for the NPC (Ma et al., 2012, Ma and Yang, 2010). In this way, the labeling efficiency and, thus, the proper formation of the NPC may be assessed via total fluorescence of the NPC compared to the total fluorescence of a single fluorophore (Ma et al., 2012, Ma and Yang, 2010).



Y,Z sample simulation: 25 nm radius, 10 nm precision, 1M points
1/4 section
1/2 section
3/4 section
Whole section
0° rotation
45° rotation
90° rotation
Deviation from ideal = -15.5 nm
Deviation from ideal = +0.3 nm
Deviation from ideal = +4.2 nm
Deviation from ideal = 0.0 nm
Deviation from ideal = -0.3 nm
Deviation from ideal = -0.2 nm
Deviation from ideal = -0.1 nm
Deviation from ideal = 0.0 nm
Deviation from ideal = +6.6 nm
Deviation from ideal = -0.1 nm
Deviation from ideal = -8.0 nm
Deviation from ideal = 0.0 nm
25 nm radius, 10 nm precision, 1M points summary
Deviation (nm)
Fraction of total ring
0° rotation
45° rotation
90° rotation

Figure 23. ***Varying the labeling ratio and rotation shows minimal effects beyond half labeling efficiency.*** **(A)-(D)** Single molecule distribution simulated from an ideal radius of 25 nm radius, 10 nm localization precision, and various labeling ratios are simulated. While the simulations used 1,000,000 points, only 1,000 are shown here for ease of visualization. **(E)-(L)** Rotation angles of 45 and 90 degrees were also simulated for the simulation conditions in A-D. **(M)-(X)** 3D density histograms from A-L with deviations from the ideal radius noted above the 3D histogram. **(Y)** Summarized results from M-X.

CHAPTER 4: CONCLUSIONS AND DISCUSSIONS

4.1 Determining the protein transport routes in the transition zone of primary cilia

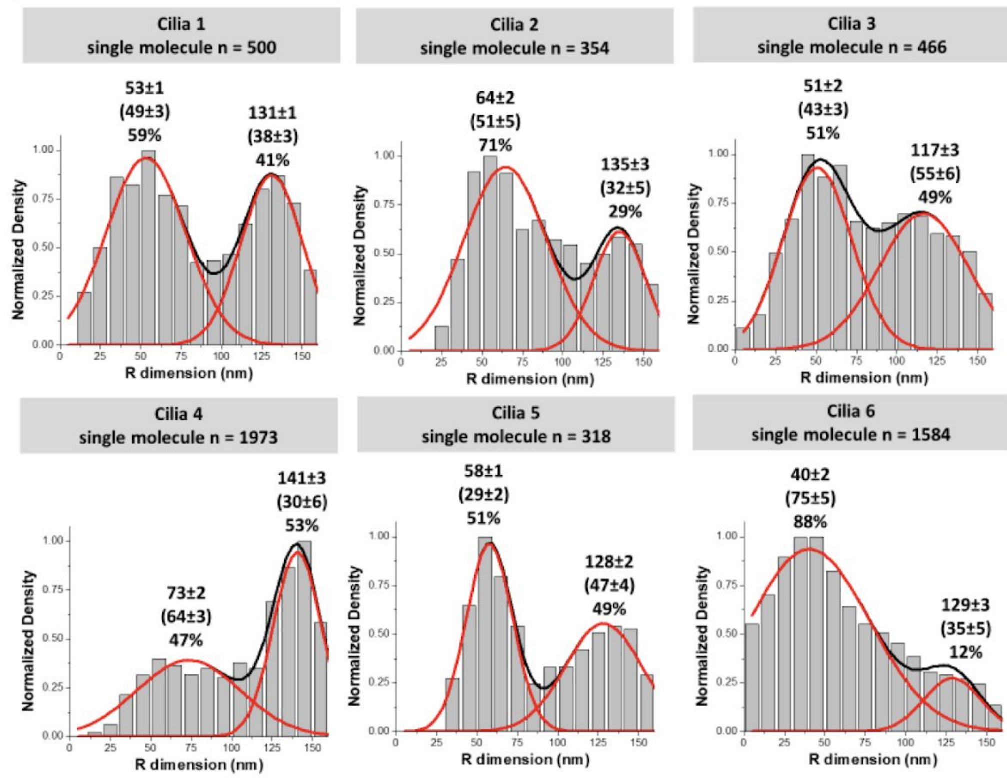
4.1.1 Vesicles as transmembrane protein carriers in the ciliary lumen

Our results show that the lumen of primary cilia is a prominent pathway for the transport of TM proteins in two model systems and that its relative usage can be regulated by a variety of ciliary states. The ciliary lumen also accommodates the transport of passively diffusing soluble molecules including α-tubulin, kinesin-2, and free GFP. We note that the axonemal lumen transport route is offset by ~50 nm from the passive diffusion route in the central axis (Figure 13 K). That vesicle transport is involved in the inner route is supported by 1) co-localization with Rab8A, 2) the reduction in transport in response to GCA treatment, and 3) sensitivity to Pitstop 2 treatment. A summary of the transport routes and the proteins that traverse these routes can be found in Figure 13 K,L.

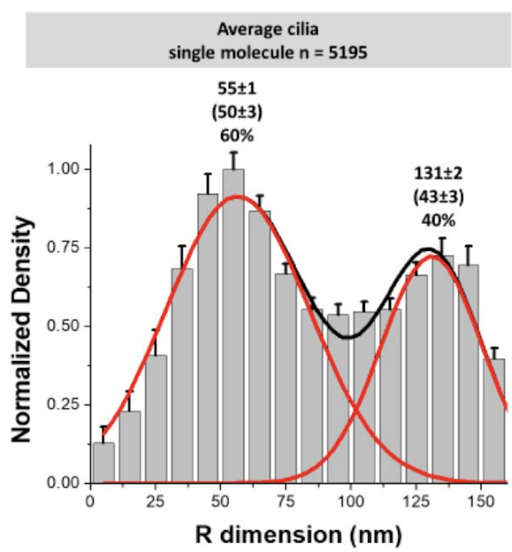
Our results characterize the ciliary lumen as a viable transport route for vesicles, a finding that has been entertained in the field based on EM studies of chondrocytes and rod photoreceptors (Chuang et al., 2015, Jensen et al., 2004). Recent work has also visualized vesicles in the amphid cilia of *C. elegans*, and vesicles accumulated in worms null for various IFT transport proteins (Doroquez et al., 2014, Li et al., 2019). A criticism of the EM work has been the propensity of the EM chemical fixation process to form vesicle-like bubbles. EM is also limited in its ability to resolve low-contrast structures like vesicles unless vesicular proteins are specifically and densely labeled by immunogold labeling (Nachury et al., 2010). In fact, early characterization studies of SSTR3 localization in the hippocampus showed dense immunogold labeling along the entire width of primary cilia (Händel et al., 1999) and electron-dense particulate collects in the empty space of primary cilia (Rogowski et al., 2013). Thus, EM would be ideal for observing vesicle transport of TM proteins in primary cilia if not for its main limitations. Therefore, we opted for a live-cell, super-resolution, and fluorescence-based approach to study this question.

We validated the ability of our SPEED microscopy and 2D-to-3D transformation algorithm to accurately reconstruct known details of protein transport in primary cilia using IFT components and the AP-SSTR3-GFP construct. These results elucidate the transport routes and mechanisms for TM and other proteins in primary cilia. Using the methods described in section 3, we also examined the effects of single molecule localization precision, number of single molecule localizations, labeling efficiency, and distortion on the precision of the final 3D transport routes via Monte Carlo simulation for every transport route presented in this thesis. Each transport route conforms to the precision standards of the biological claims we make according to simulation and the results are summarized in Table 1. Furthermore, we emphasize the reasonable consistency of the ciliary ultrastructure and the experimental reproducibility of each transport route by showing the 3D transport route histograms for SSTR3 in 6 different cilia and quantify the error (Figure 24).

A



B



C

cilia n = 6

Parameter	Standard deviation	Standard error
Inner route mean (nm)	11.4	4.7
Inner route width (nm)	16.1	6.6
Inner route area (%)	15.6	6.4
Outer route mean (nm)	8.0	3.2
Outer route width (nm)	9.6	3.9
Outer route area (%)	15.6	6.4

Figure 24. ***Variation between different 3D density histograms for SSTR3 in ciliary shaft.*** **A)** 3D transformed histogram from 6 different primary cilia. **B)** Averaged 3D histogram from different primary cilia in (A). **C)** Summary of statistics from (B).

Even though the lumen of the TZ is a relatively open and clear channel compared to the surrounding structural regions and likely free from the cartwheel structure that characterizes non-ciliary procentrioles (Alvey, 1986, Hoyer-Fender, 2013), an important consideration is how vesicles may pass through the basal body, TZ, and cilium shaft. First, there may be some vesicle distortion that occurs to allow vesicle passage into and along primary cilia. Indeed, the recycling endosome, a waypoint for TM proteins sent to the primary cilia, has a convoluted membrane structure and its vesicles are highly non-uniform (Goldenring, 2015). Second, there appears to be a requirement for proteins to mediate the transport of vesicles past the diffusion barrier into primary cilia. For example, RAB8A is responsible for mediating the entry into the connecting cilium of frog photoreceptors (Moritz et al., 2001) and we show that it occupies the same transport routes as SSTR3 in vivo. Third, small vesicles carrying TM proteins may be able to pass through the diffusion barrier, albeit at a low frequency due to being above the widely accepted barrier limit (Kee et al., 2012, Breslow et al., 2013, Calvert et al., 2010, Lin et al., 2013).

Previous models have suggested that the fusion of TM protein-containing vesicles likely occurs outside the primary cilia, at the periciliary base, or at the TZ ciliary membrane (Nachury et al., 2010). Overall, our results do not refute these findings. Indeed, RAB8A possessed transport routes in the lumen and near the ciliary membrane. In addition, inhibition of TM protein vesicle export from Golgi with GCA showed reduction in both the inner and outer transport routes. This suggests that vesicle transport is a component of transiting both TZ locations. Our work aims to parse out the usage of both of these routes during the different ciliary states.

4.1.2 The ciliary lumen as a specialized signaling transport route

Several groups have shown via fluorescence microscopy that markers for clathrin-coated pits localize to the ciliary pocket (Molla-Herman et al., 2010) and others find that β-arrestin is recruited to primary cilia upon SSTR3 stimulation and has a causal role in its removal (Green et al., 2016). Based on this evidence, it appears that β-arrestin is recruited to primary cilia following SSTR3 stimulation, attenuates SSTR3's signaling, and facilitates transport by IFT out of primary cilia where it promotes interaction with endocytic machinery. Our data extend this model by suggesting that the ciliary membrane possesses endocytic capabilities and that the internalized receptors can be transported through the axonemal lumen.

The primary cilium is a specialized signaling organelle whose evolutionary progenitor is the motile flagellum (Mitchell et al., 2007). Over time and with selection pressure against motility in eukaryotic cells, it appears that proto-primary cilia lost the central pair of microtubules and associated motility proteins. This loss of motility function may have paved the way for an increase in the signaling capabilities by reducing the geometric constraints of the flagellum. Here, we show that TM protein transport at various stages of the primary cilium lifecycle – growth, steady-state, and signaling states – can occur, at least in part, through the axonemal lumen. In future work, probing the differences between flagella and primary cilia and incorporating other well-known ciliary protein transport components, such as the BBSome, will be essential in developing a molecular view of the various ciliopathies.

4.2 Determining the error on 3D transport routes produced by SPEED microscopy and the 2D-to-3D transformation algorithm

We also presented a detailed analysis of the 2D-to-3D transformation algorithm that enabled us to obtain 3D super-resolution information from 2D super-resolution images or 2D single-molecule localization data without using 3D light microscopy setups. The roles that two critical factors played in reproducing the accurate 3D super-resolution information, the single molecule localization error and the number of single molecule locations, have been fully discussed. We also discussed a general rule that a transport route can be well distinguished if it has a radius/precision ratio greater than 2 and the ratio-based minimum number of single-molecule locations (Figure 25). The successful applications in various systems, including the GNC, the NPC and primary cilia in live cells, and microtubules in fixed samples, prove the robustness of achieving accurate 3D super-resolution information by combining 2D experimental data and the 2D-to-3D transformation algorithm.

A

Radius/Precision	Reproducibility %	# of points
10	90%	100
5	90%	100
3.33	90%	100
2.5	90%	100
2	90%	200
1.67	90%	500
1.43	90%	1500
1.25	90%	3100
1.19	90%	8500

B

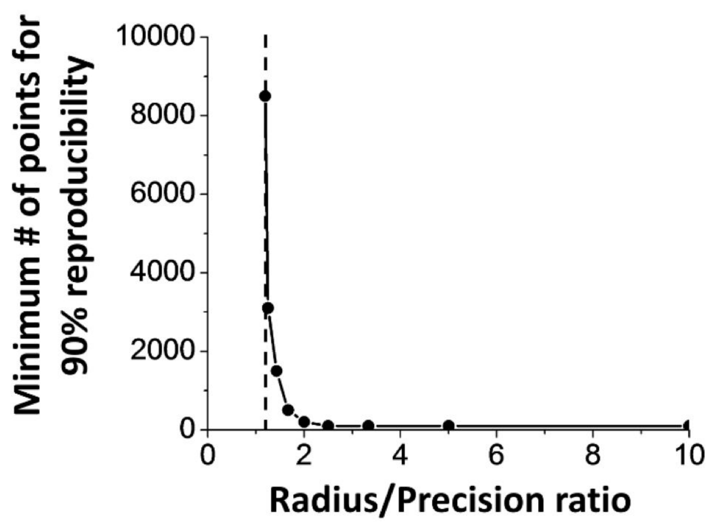


Figure 25. ***The minimum number of points required to resolve a transport route above the radius/precision threshold.*** **(A)** Table illustrating the minimum point number requirements for 90% reproducibility across a range of radius/s.d. ratios. Point numbers were rounded to the nearest 100 points. % route precision was calculated by dividing the transport route localization error (σ_{TR}) by the ideal radius (R_I). **(B)** Graph showing the table from (A) plotted in black dots and solid lines down to the minimum distinguishable R/P ratio threshold of 1.19 (dashed line).

It is noteworthy that one prerequisite of the algorithm is that the density of the molecules of interest is constant along a given radial bin for the biological structure. As a result, the rotational symmetry cannot be determined by our algorithm but, rather, must be determined by another technique. Normally, transmission electron microscopy, especially since the development of freezing techniques for specimen fixation which reduces distortion (Dubochet et al., 1988), is appropriate for determining the ground truth structure and its rotational symmetry with high spatial resolution. Another alternative approach is expansion microscopy, by which the size of a specimen can be enlarged 4.5 to 20 times without significant distortion and the enlarged structure can be labeled and imaged by epi-fluorescence or confocal light microscopy (Chen et al., 2015, Chang et al., 2017). As demonstrated, with the known structures of the NPC, primary cilia, and microtubules, area matrices have been developed for the cylindrical structures based on their rotational symmetry for 2D-to-3D transformation algorithm. Furthermore, area matrices could also be developed for other regular or even irregularly-shaped, radially symmetric structures, with the only prerequisite being constant density of the molecules of interest along a given radial bin.

4.3 Future directions of the use of SPEED microscopy and the 2D-to-3D algorithm to investigate protein transport routes in primary cilia

In this work, one of the major aims of this work was to investigate the transport routes primarily of transmembrane proteins in primary cilia. We looked at these routes at different ciliary locations, under normal cellular conditions, during ciliogenesis, during active receptor signaling and clathrin-mediated internalization, and in the evolutionarily-related model system, the flagellum.

While we used small molecule inhibitors to roughly probe the mechanisms responsible for these transport routes, a natural next step would be to individually probe the contribution of the various components of IFT and the BBSome on transmembrane protein transport and function. This is particularly important because of 1) IFT's role in intraciliary transmembrane protein dynamics (Scholey and Anderson, 2006) and 2) the BBSome's role in directly interacting with the ciliary targeting sequence of many ciliary transmembrane proteins and facilitating their proper localization to primary cilia (Berbari et al., 2008). Traditionally, IFT and BBSome function has been studied *in vivo* through knockout and knockdown experiments. However, this approach is often limited because IFT proteins have roles outside the primary cilium and knockout/knockdown causes cell death or inhibits ciliogenesis. Similarly, BBSome knockout/knockdown causes improper localization of ciliary transmembrane proteins, thus, increasing the difficulty of studying a ciliary transmembrane protein's response to ligand binding in the absence of the BBSome or BBSome components.

Recently, the auxin inducible degron has seen increased application to many different cellular model systems as a way to induce the intracellular depletion of any particular protein in minutes to hours (Natsume et al., 2016). This presents an exciting, new way to deplete particular IFT or BBSome components after ciliogenesis has been completed and transmembrane proteins have been properly trafficked to primary cilia in a timeframe that is unlikely to lead to cell death. Such a focused and reliable approach will be necessary to parse out the contribution of these proteins to the lifecycle of transmembrane proteins in primary cilia.

Ultimately, the primary cilium is a signaling organelle. It should be no surprise that the loss of primary cilia is one of the steps leading to a cancerous cell (Michaud and Yoder, 2006). In addition some of the most debilitating ciliopathies are those that affect transmembrane proteins or transport processes affecting transmembrane proteins (Hildebrandt et al., 2011). With the contribution of this work and further work in parsing out the contribution of IFT and BBSome to ciliary transmembrane protein transport, we can investigate transport-related ciliopathies with a clearer model of how a normally functioning primary cilium should operate.

REFERENCES CITED

Ainavarapu, Sri Rama Koti, et al. "Contour length and refolding rate of a small protein controlled by engineered disulfide bonds." Biophysical journal 92.1 (2007): 225-233.

Akey, Christopher W. "Interactions and structure of the nuclear pore complex revealed by cryo-electron microscopy." The Journal of cell biology 109.3 (1989): 955-970.

Akey, Christopher W., and Michael Radermacher. "Architecture of the Xenopus nuclear pore complex revealed by three-dimensional cryo-electron microscopy." The Journal of Cell Biology 122.1 (1993): 1-19.

Alvey, P. L. "Do adult centrioles contain cartwheels and lie at right angles to each other?." Cell biology international reports 10.8 (1986): 589-598.

Anholt, Robert RH, et al. "Transduction proteins of olfactory receptor cells: identification of guanine nucleotide binding proteins and protein kinase C." Biochemistry 26.3 (1987): 788-795.

Anvarian, Zeinab, et al. "Cellular signalling by primary cilia in development, organ function and disease." Nature Reviews Nephrology (2019): 1.

Anzalone, Andrea, Paolo Annibale, and Enrico Gratton. "3D orbital tracking in a modified two-photon microscope: an application to the tracking of intracellular vesicles." JoVE (Journal of Visualized Experiments) 92 (2014): e51794.

Arts, Heleen H., et al. "C14ORF179 encoding IFT43 is mutated in Sensenbrenner syndrome." Journal of medical genetics 48.6 (2011): 390-395.

Awata, Junya, et al. "Nephrocystin-4 controls ciliary trafficking of membrane and large soluble proteins at the transition zone." J Cell Sci (2014): jcs-155275.

Bangs, Fiona, and Kathryn V. Anderson. "Primary cilia and mammalian hedgehog signaling." Cold Spring Harbor perspectives in biology 9.5 (2017): a028175.

Barzi, Mercedes, et al. "Sonic Hedgehog-induced proliferation requires specific Gα inhibitory proteins." Journal of Biological Chemistry 286.10 (2011): 8067-8074.

Beck, Martin, et al. "Snapshots of nuclear pore complexes in action captured by cryo-electron tomography." Nature 449.7162 (2007): 611.

Berbari, N. F., A. D. Johnson, et al. Identification of ciliary localization sequences within the third intracellular loop of G protein-coupled receptors. Mol Biol Cell. 194:1540-1547, 2008.

Berbari, N. F., J. S. Lewis, et al. Bardet-Biedl syndrome proteins are required for the localization of G protein-coupled receptors to primary cilia. Proc Natl Acad Sci U S A. 10511:4242-4246, 2008.

Berbari, Nicolas F., et al. "Hippocampal neurons possess primary cilia in culture." Journal of neuroscience research 85.5 (2007): 1095-1100.

Betzig, E. et al. Imaging intracellular fluorescent proteins at nanometer resolution. Science. 313, 1642-1645 (2006).

Böhm, Ulrike, Stefan W. Hell, and Roman Schmidt. "4Pi-RESOLFT nanoscopy." Nature communications 7 (2016): 10504.

Borisy, Gary, et al. "Microtubules: 50 years on from the discovery of tubulin." Nature Reviews Molecular Cell Biology 17.5 (2016): 322.

Breslow, David K., et al. "A CRISPR-based screen for Hedgehog signaling provides insights into ciliary function and ciliopathies." Nature genetics 50.3 (2018): 460.

Breslow, David K., et al. "An in vitro assay for entry into cilia reveals unique properties of the soluble diffusion barrier." J Cell Biol 203.1 (2013): 129-147.

Breunig, Joshua J., et al. "Primary cilia regulate hippocampal neurogenesis by mediating sonic hedgehog signaling." Proceedings of the National Academy of Sciences 105.35 (2008): 13127-13132.

Brooks, Celine, et al. "Farnesylation of the Transducin G protein gamma subunit is a prerequisite for its ciliary targeting in rod photoreceptors." Frontiers in molecular neuroscience 11 (2018): 16.

Calvert, Peter D., William E. Schiesser, and Edward N. Pugh. "Diffusion of a soluble protein, photoactivatable GFP, through a sensory cilium." The Journal of general physiology 135.3 (2010): 173-196.

Chang, Jae-Byum, et al. "Iterative expansion microscopy." Nature methods 14.6 (2017): 593.

Chen, Fei, Paul W. Tillberg, and Edward S. Boyden. "Expansion microscopy." Science 347.6221 (2015): 543-548.

Chuang, Jen-Zen, Ya-Chu Hsu, and Ching-Hwa Sung. "Ultrastructural visualization of trans-ciliary rhodopsin cargoes in mammalian rods." Cilia 4.1 (2015): 4.

Corbit, Kevin C., et al. "Vertebrate Smoothened functions at the primary cilium." Nature 437.7061 (2005): 1018.

Craft, Julie M., et al. "Tubulin transport by IFT is upregulated during ciliary growth by a cilium-autonomous mechanism." J Cell Biol 208.2 (2015): 223-237.

Craige, Branch, et al. "CEP290 tethers flagellar transition zone microtubules to the membrane and regulates flagellar protein content." The Journal of cell biology 190.5 (2010): 927-940.

Czarnecki, Peter G., and Jagesh V. Shah. "The ciliary transition zone: from morphology and molecules to medicine." Trends in cell biology 22.4 (2012): 201-210.

Desai, Arshad, and Timothy J. Mitchison. "Microtubule polymerization dynamics." Annual review of cell and developmental biology 13.1 (1997): 83-117.

Deschout, Hendrik, Kristiaan Neyts, and Kevin Braeckmans. "The influence of movement on the localization precision of sub‐resolution particles in fluorescence microscopy." Journal of biophotonics 5.1 (2012): 97-109.

Dirksen, Ellen Roter, and Ilona Staprans. "Tubulin synthesis during ciliogenesis in the mouse oviduct." Developmental biology 46.1 (1975): 1-13.

Doroquez, David, et al. "A high-resolution morphological and ultrastructural map of anterior sensory cilia and glia in Caenorhabditis elegans." eLife 3 (2014): e01948.

Dosztányi, Zsuzsanna, et al. "IUPred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content." Bioinformatics 21.16 (2005): 3433-3434.

Dubochet, Jacques, et al. "Cryo-electron microscopy of vitrified specimens." Quarterly reviews of biophysics 21.2 (1988): 129-228.

Eguether, T., J. T. San Agustin, et al. IFT27 links the BBSome to IFT for maintenance of the ciliary signaling compartment. Dev Cell. 313:279-290, 2014.

Einstein, Emily B., et al. "Somatostatin signaling in neuronal cilia is criticalfor object recognition memory." Journal of Neuroscience 30.12 (2010): 4306-4314.

Elf, Johan, Gene-Wei Li, and X. Sunney Xie. "Probing transcription factor dynamics at the single-molecule level in a living cell." Science 316.5828 (2007): 1191-1194.

Engelke, Martin F., et al. "Acute inhibition of heterotrimeric kinesin-2 function reveals mechanisms of intraflagellar transport in mammalian cilia." Current Biology (2019).

Erickson, Harold P. "Size and shape of protein molecules at the nanometer level determined by sedimentation, gel filtration, and electron microscopy." Biological procedures online 11.1 (2009): 32.

Fahrenkrog, Birthe, and Ueli Aebi. "The nuclear pore complex: nucleocytoplasmic transport and beyond." Nature Reviews Molecular Cell Biology 4.10 (2003): 757.

Florio, T., et al. "Somatostatin inhibits tumor angiogenesis and growth via somatostatin receptor-3-mediated regulation of endothelial nitric oxide synthase and mitogen-activated protein kinase activities." Endocrinology 144.4 (2003): 1574-1584.

Follit, John A., et al. "The intraflagellar transport protein IFT20 is associated with the Golgi complex and is required for cilia assembly." Molecular biology of the cell 17.9 (2006): 3781-3792.

Franke, Christian, Markus Sauer, and Sebastian van de Linde. "Photometry unlocks 3D information from 2D localization microscopy data." Nature methods 14.1 (2017): 41.

Fried, Howard, and Ulrike Kutay. "Nucleocytoplasmic transport: taking an inventory." Cellular and Molecular Life Sciences CMLS 60.8 (2003): 1659-1688.

Gerdes, J. M., and N. Katsanis. "Microtubule transport defects in neurological and ciliary disease." Cellular and Molecular Life Sciences CMLS 62.14 (2005): 1556-1570.

Goldenring, James R. "Recycling endosomes." Current opinion in cell biology 35 (2015): 117-122.

Green, Jill A., et al. "Recruitment of β-arrestin into neuronal cilia modulates somatostatin receptor subtype 3 ciliary localization." Molecular and cellular biology 36.1 (2016): 223-235.

Grigoriev, I., K. L. Yu, et al. Rab6, Rab8, and MICAL3 cooperate in controlling docking and fusion of exocytotic carriers. Curr Biol. 2111:967-974, 2011.

Händel, M., et al. "Selective targeting of somatostatin receptor 3 to neuronal cilia." Neuroscience 89.3 (1999): 909-926.

Hao, Limin, et al. "Intraflagellar transport delivers tubulin isotypes to sensory cilium middle and distal segments." Nature cell biology 13.7 (2011): 790.

Harris, J. Aaron, et al. "Diffusion rather than IFT provides most of the tubulin required for axonemal assembly." BioRxiv (2018): 268573.

He, Xuelian, et al. "The G protein α subunit Gα s is a tumor suppressor in Sonic hedgehog− driven medulloblastoma." Nature medicine 20.9 (2014): 1035.

Hell, S.W., Wichmann, J. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. Opt Lett. 19, 780-782 (1994).

Higginbotham, Holden, et al. "Arl13b in primary cilia regulates the migration and placement of interneurons in the developing cerebral cortex." Developmental cell 23.5 (2012): 925-938.

Hildebrandt, Friedhelm, Thomas Benzing, and Nicholas Katsanis. "Ciliopathies." New England Journal of Medicine 364.16 (2011): 1533-1543.

Hilgendorf, Keren I., Carl T. Johnson, and Peter K. Jackson. "The primary cilium as a cellular receiver: organizing ciliary GPCR signaling." Current opinion in cell biology 39 (2016): 84-92.

Hirano, Tomoaki, Yohei Katoh, and Kazuhisa Nakayama. "Intraflagellar transport-A complex mediates ciliary entry and retrograde trafficking of ciliary G protein–coupled receptors." Molecular biology of the cell 28.3 (2017): 429-439.

Howarth, Mark, and Alice Y. Ting. "Imaging proteins in live mammalian cells with biotin ligase and monovalent streptavidin." Nature protocols 3.3 (2008): 534.

Hoyer-Fender, Sigrid. "Primary and motile cilia: their ultrastructure and ciliogenesis." Cilia and Nervous System Development and Function. Springer, Dordrecht, 2013. 1-53.

Huang, B., Bates, M. & Zhuang, X. Super-resolution fluorescence microscopy. Annu. Rev. Biochem. 78, 993-1016 doi:10.1146/annurev.biochem.77.061906.092014 (2009).

Huang, Bo, et al. "Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy." Science 319.5864 (2008): 810-813.

Huang, Kaiyao, et al. "Function and dynamics of PKD2 in Chlamydomonas reinhardtii flagella." The Journal of cell biology 179.3 (2007): 501-514.

Hunnicutt, Gary R., Maria G. Kosfiszer, and William J. Snell. "Cell body and flagellar agglutinins in Chlamydomonas reinhardtii: the cell body plasma membrane is a reservoir for agglutinins whose migration to the flagella is regulated by a functional barrier." The Journal of Cell Biology 111.4 (1990): 1605-1616.

Ishikawa, Hiroaki, and Wallace F. Marshall. "Ciliogenesis: building the cell's antenna." Nature reviews Molecular cell biology 12.4 (2011): 222.

Ishikawa, Hiroaki, et al. "Proteomic analysis of mammalian primary cilia." Current Biology 22.5 (2012): 414-419.

Jana, Swadhin Chandra, et al. "Differential regulation of transition zone and centriole proteins contributes to ciliary base diversity." Nature cell biology 20.8 (2018): 928.

Jensen, C. G., et al. "Ultrastructural, tomographic and confocal imaging of the chondrocyte primary cilium in situ." Cell biology international 28.2 (2004): 101-110.

Jensen, Victor L., and Michel R. Leroux. "Gates for soluble and membrane proteins, and two trafficking systems (IFT and LIFT), establish a dynamic ciliary signaling compartment." Current opinion in cell biology 47 (2017): 83-91.

Jin, H., S. R. White, et al. The conserved Bardet-Biedl syndrome proteins assemble a coat that traffics membrane proteins to cilia. Cell. 1417:1208-1219, 2010.

Jones, Chonnettia, et al. "Ciliary proteins link basal body polarization to planar cell polarity regulation." Nature genetics 40.1 (2008): 69.

Kaneshiro, E. S. "Lipids of Paramecium." Journal of lipid research 28.11 (1987): 1241-1258.

Kee, Hooi Lynn, et al. "A size-exclusion permeability barrier and nucleoporins characterize a ciliary pore complex that regulates transport into cilia." Nature cell biology 14.4 (2012): 431.

Kozminski, Keith G., Peter L. Beech, and Joel L. Rosenbaum. "The Chlamydomonas kinesin-like protein FLA10 is involved in motility associated with the flagellar membrane." The Journal of cell biology 131.6 (1995): 1517-1527.

Lee, Marissa K., et al. "Small-molecule labeling of live cell surfaces for three-dimensional super-resolution microscopy." Journal of the American Chemical Society 136.40 (2014): 14003-14006.

Lerea, Connie L., et al. "Identification of specific transducin alpha subunits in retinal rod and cone photoreceptors." Science 234.4772 (1986): 77-80.

Leung, B. O. & Chou, K. C. Review of super-resolution fluorescence microscopy for biology. Appl. Spectrosc. 65, 967-980 doi:10.1366/11-06398 (2011).

Lew, Matthew D., et al. "Three-dimensional superresolution colocalization of intracellular protein superstructures and the cell surface in live Caulobacter crescentus." Proceedings of the National Academy of Sciences 108.46 (2011): E1102-E1110.

Li, Ming, et al. "Ciliopathy-associated proteins are involved in vesicle distribution in sensory cilia." Journal of genetics and genomics 46 (2019): 269-271.

Lin, Yu-Chun, et al. "Chemically inducible diffusion trap at cilia reveals molecular sieve–like barrier." Nature chemical biology 9.7 (2013): 437.

Luo, Wangxi, et al. "Axonemal lumen dominates cytosolic protein diffusion inside the primary cilium." Scientific reports 7.1 (2017): 15793.

Ma, Jiong, and Weidong Yang. "Three-dimensional distribution of transient interactions in the nuclear pore complex obtained from single-molecule snapshots." Proceedings of the National Academy of Sciences 107.16 (2010): 7305-7310.

Ma, Jiong, et al. "High-resolution three-dimensional mapping of mRNA export through the nuclear pore." Nature communications 4 (2013): 2414.

Ma, Jiong, et al. "Self-regulated viscous channel in the nuclear pore complex." Proceedings of the National Academy of Sciences 109.19 (2012): 7326-7331.

Ma, Jiong, Joseph M. Kelich, and Weidong Yang. "SPEED microscopy and its application in nucleocytoplasmic transport." The Nuclear Envelope: Methods and Protocols (2016): 503-518.

Mahadevan, Sankaran. "Monte carlo simulation." Mechanical Engineering-New York and Basel-Marcel Dekker- (1997): 123-146.

Marshall, Wallace F., and Shigenori Nonaka. "Cilia: tuning in to the cell's antenna." Current Biology 16.15 (2006): R604-R614.

Melčák, Ivo, André Hoelz, and Günter Blobel. "Structure of Nup58/45 suggests flexible nuclear pore diameter by intermolecular sliding." Science 315.5819 (2007): 1729-1732.

Michaud, Edward J., and Bradley K. Yoder. "The primary cilium in cell signaling and cancer." Cancer research 66.13 (2006): 6463-6467.

Mitchell, David R. "The evolution of eukaryotic cilia and flagella as motile and sensory organelles." Eukaryotic Membranes and Cytoskeleton. Springer, New York, NY, 2007. 130-140.

Molla-Herman, Anahi, et al. "The ciliary pocket: an endocytic membrane domain at the base of primary and motile cilia." Journal of cell science (2010): jcs-059519.

Mooney, Christopher Z. Monte carlo simulation. Vol. 116. Sage Publications, 1997.

Moritz, Orson L., et al. "Mutant rab8 Impairs docking and fusion of rhodopsin-bearing post-Golgi membranes and causes cell death of transgenic Xenopus rods." Molecular biology of the cell 12.8 (2001): 2341-2351.

Mortensen, Kim I., et al. "Optimized localization analysis for single-molecule tracking and super-resolution microscopy." Nature methods 7.5 (2010): 377.

Mueller, Joshua, et al. "The FLA3 KAP subunit is required for localization of kinesin-2 to the site of flagellar assembly and processive anterograde intraflagellar transport." Molecular biology of the cell 16.3 (2005): 1341-1354.

Muresan, Virgil, Harish C. Joshi, and Joseph C. Besharse. "Gamma-tubulin in differentiated cell types: localization in the vicinity of basal bodies in retinal photoreceptors and ciliated epithelia." Journal of Cell Science 104.4 (1993): 1229-1237.

Nachury, Maxence V., and David U. Mick. "Establishing and regulating the composition of cilia for signal transduction." Nature Reviews Molecular Cell Biology (2019): 1.

Nachury, Maxence V., E. Scott Seeley, and Hua Jin. "Trafficking to the ciliary membrane: how to get across the periciliary diffusion barrier?." Annual review of cell and developmental biology 26 (2010): 59-87.

Nachury, Maxence V., et al. "A core complex of BBS proteins cooperates with the GTPase Rab8 to promote ciliary membrane biogenesis." Cell 129.6 (2007): 1201-1213.

Nager, Andrew R., et al. "An actin network dispatches ciliary GPCRs into extracellular vesicles to modulate signaling." Cell 168.1 (2017): 252-263.

Nair, K. Saidas, et al. "Light-dependent redistribution of arrestin in vertebrate rods is an energy-independent process governed by protein-protein interactions." Neuron 46.4 (2005): 555-567.

Najafi, Mehdi, Nycole A. Maza, and Peter D. Calvert. "Steric volume exclusion sets soluble protein concentrations in photoreceptor sensory cilia." Proceedings of the National Academy of Sciences 109.1 (2012): 203-208.

Natsume, Toyoaki, et al. "Rapid protein depletion in human cells by auxin-inducible degron tagging with short homology donors." Cell reports 15.1 (2016): 210-218.

Nauli, Surya M., et al. "Polycystins 1 and 2 mediate mechanosensation in the primary cilium of kidney cells." Nature genetics 33.2 (2003): 129.

Nogales, Eva. "Structural insights into microtubule function." Annual review of biophysics and biomolecular structure 30.1 (2001): 397-420.

O'Hagan, Robert, et al. "Glutamylation regulates transport, specializes function, and sculpts the structure of cilia." Current Biology 27.22 (2017): 3430-3441.

Oakley, Robert H., et al. "Association of β-arrestin with G protein-coupled receptors during clathrin-mediated endocytosis dictates the profile of receptor resensitization." Journal of Biological Chemistry 274.45 (1999): 32248-32257.

Ostrowski, Lawrence E., et al. "A proteomic analysis of human cilia: identification of novel components." Molecular & Cellular Proteomics 1.6 (2002): 451-465.

Oswald, Felix, et al. "MreB-dependent organization of the E. coli cytoplasmic membrane controls membrane protein diffusion." Biophysical journal 110.5 (2016): 1139-1149.

Ott, Carolyn, and Jennifer Lippincott‐Schwartz. "Visualization of live primary cilia dynamics using fluorescence microscopy." Current protocols in cell biology 57.1 (2012): 4-26.

Ou, G., O. E. Blacque, et al. Functional coordination of intraflagellar transport motors. Nature. 4367050:583-587, 2005.

Papermaster, David S., B. G. Schneider, and J. C. Besharse. "Vesicular transport of newly synthesized opsin from the Golgi apparatus toward the rod outer segment. Ultrastructural immunocytochemical and autoradiographic evidence in Xenopus retinas." Investigative ophthalmology & visual science 26.10 (1985): 1386-1404.

Pavani, Sri Rama Prasanna, et al. "Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function." Proceedings of the National Academy of Sciences 106.9 (2009): 2995-2999.

Pazour, Gregory J., and Joel L. Rosenbaum. "Intraflagellar transport and cilia-dependent diseases." Trends in cell biology 12.12 (2002): 551-555.

Pazour, Gregory J., and Robert A. Bloodgood. "Targeting proteins to the ciliary membrane." Current topics in developmental biology 85 (2008): 115-149.

Pazour, Gregory J., et al. "Polycystin-2 localizes to kidney cilia and the ciliary level is elevated in orpk mice with polycystic kidney disease." Current Biology 12.11 (2002): R378-R380.

Pazour, Gregory J., et al. "Proteomic analysis of a eukaryotic cilium." J Cell Biol 170.1 (2005): 103-113.

Phua, Siew Cheng, et al. "Dynamic remodeling of membrane composition drives cell cycle through primary cilia excision." Cell 168.1-2 (2017): 264-279.

Poole, C. Anthony, Michael H. Flint, and Brent W. Beaumont. "Analysis of the morphology and function of primary cilia in connective tissues: a cellular cybernetic probe?." Cell motility 5.3 (1985): 175-193.

Prabhat, Prashant, et al. "Simultaneous imaging of different focal planes in fluorescence microscopy for the study of cellular dynamics in three dimensions." IEEE transactions on nanobioscience 3.4 (2004): 237-242.

Praetorius, Helle A. "The primary cilium as sensor of fluid flow: new building blocks to the model. A review in the theme: cell signaling: proteins, pathways and mechanisms." American Journal of Physiology-Cell Physiology 308.3 (2014): C198-C208.

Praetorius, Helle A., and Kenneth R. Spring. "The renal cell primary cilium functions as a flow sensor." Current opinion in nephrology and hypertension 12.5 (2003): 517-520.

Prevo, Bram, Jonathan M. Scholey, and Erwin JG Peterman. "Intraflagellar transport: mechanisms of motor action, cooperation, and cargo delivery." The FEBS journal 284.18 (2017): 2905-2931.

Qin, H., D. T. Burnette, et al. Intraflagellar transport is required for the vectorial movement of TRPV channels in the ciliary membrane. Curr Biol. 1518:1695-1699, 2005.

Quan, Tingwei, Shaoqun Zeng, and Zhenli Huang. "Localization capability and limitation of electron-multiplying charge-coupled, scientific complementary metal-oxide semiconductor, and charge-coupled devices for superresolution imaging." Journal of biomedical optics 15.6 (2010): 066005.

Ram, Sripad, E. Sally Ward, and Raimund J. Ober. "Beyond Rayleigh's criterion: a resolution measure with application to single-molecule microscopy." Proceedings of the National Academy of Sciences 103.12 (2006): 4457-4462.

Reese, T. S. "Olfactory cilia in the frog." The Journal of cell biology 25.2 (1965): 209-230.

Ritter, Jörg Gerhard, et al. "Light sheet microscopy for single molecule tracking in living tissue." PloS one 5.7 (2010): e11639.

Robbins, Mark Stanford, and Benjamin James Hadwen. "The noise performance of electron multiplying charge-coupled devices." IEEE transactions on Electron Devices 50.5 (2003): 1227-1232.

Rogowski, Michaela, Dirk Scholz, and Stefan Geimer. "Electron microscopy of flagella, primary cilia, and intraflagellar transport in flat-embedded cells." Methods in enzymology. Vol. 524. Academic Press, 2013. 243-263.

Rohatgi, Rajat, Ljiljana Milenkovic, and Matthew P. Scott. "Patched1 regulates hedgehog signaling at the primary cilium." Science 317.5836 (2007): 372-376.

Rosenbaum, Joel L., and George B. Witman. "Intraflagellar transport." Nature reviews Molecular cell biology 3.11 (2002): 813.

Rosenzweig, Derek H., et al. "Subunit dissociation and diffusion determine the subcellular localization of rod and cone transducins." Journal of Neuroscience 27.20 (2007): 5484-5494.

Ross, Alison J., et al. "Disruption of Bardet-Biedl syndrome ciliary proteins perturbs planar cell polarity in vertebrates." Nature genetics 37.10 (2005): 1135.

Rout, Michael P., et al. "Virtual gating and nuclear transport: the hole picture." Trends in cell biology 13.12 (2003): 622-628.

Ruba, Andrew, et al. "3D Tracking-Free Approach for Obtaining 3D Super-Resolution Information in Rotationally Symmetric Bio-Structures." The Journal of Physical Chemistry B (2019).

Ruba, Andrew, et al. "Reply to 'Deconstructing transport-distribution reconstruction in the nuclear-pore complex'." Nature structural & molecular biology 25.12 (2018): 1062.

Ruba, Andrew, Wangxi Luo, and Weidong Yang. "Application of High-speed Super-resolution SPEED Microscopy in Live Primary Cilium." Journal of visualized experiments: JoVE 131 (2018).

Sage, Daniel, et al. "Quantitative evaluation of software packages for single-molecule localization microscopy." Nature methods 12.8 (2015): 717.

Scholey, Jonathan M., and Kathryn V. Anderson. "Intraflagellar transport and cilium-based signaling." Cell 125.3 (2006): 439-442.

Sharma, Kamal, Yogesh C. Patel, and Coimbatore B. Srikant. "Subtype-selective induction of wild-type p53 and apoptosis, but not cell cycle arrest, by human somatostatin receptor 3." Molecular Endocrinology 10.12 (1996): 1688-1696.

Shechtman, Yoav, et al. "Optimal point spread function design for 3D imaging." Physical review letters 113.13 (2014): 133902.

Shida, Toshinobu, et al. "The major α-tubulin K40 acetyltransferase αTAT1 promotes rapid ciliogenesis and efficient mechanosensation." Proceedings of the National Academy of Sciences 107.50 (2010): 21517-21522.

Singh, Jaskirat, Xiaohui Wen, and Suzie J. Scales. "The orphan G protein-coupled receptor Gpr175 (Tpra40) enhances Hedgehog signaling by modulating cAMP levels." Journal of Biological Chemistry 290.49 (2015): 29663-29675.

Singla, Veena, and Jeremy F. Reiter. "The primary cilium as the cell's antenna: signaling at a sensory organelle." science 313.5787 (2006): 629-633.

Stoffler, Daniel, et al. "Cryo-electron tomography provides novel insights into nuclear pore architecture: implications for nucleocytoplasmic transport." Journal of molecular biology 328.1 (2003): 119-130.

Sun, Shufeng, et al. "Three-dimensional architecture of epithelial primary cilia." Proceedings of the National Academy of Sciences 116.19 (2019): 9370-9379.

Taschner, M., S. Bhogaraju, et al. Architecture and function of IFT complex proteins in ciliogenesis. Differentiation. 832:S12-22, 2012.

Thompson, Michael A., et al. "Molecules and methods for super-resolution imaging." Methods in enzymology. Vol. 475. Academic Press, 2010. 27-59.

Thompson, Russell E., Daniel R. Larson, and Watt W. Webb. "Precise nanometer localization analysis for individual fluorescent probes." Biophysical journal 82.5 (2002): 2775-2783.

Vieira, Otilia V., et al. "FAPP2, cilium formation, and compartmentalization of the apical membrane in polarized Madin–Darby canine kidney (MDCK) cells." Proceedings of the National Academy of Sciences 103.49 (2006): 18556-18561.

von Kleist, Lisa, et al. "Role of the clathrin terminal domain in regulating coated pit dynamics revealed by small molecule inhibition." Cell 146.3 (2011): 471-484.

Wang, Jing, et al. "The Arf GAP ASAP1 provides a platform to regulate Arf4‐and Rab11–Rab8‐mediated ciliary receptor targeting." The EMBO Journal 31.20 (2012): 4057-4071.

Wang, Juan, et al. "C. elegans ciliated sensory neurons release extracellular vesicles that function in animal communication." Current Biology 24.5 (2014): 519-525.

Wang, Lei, and Brian D. Dynlacht. "The regulation of cilium assembly and disassembly in development and disease." Development 145.18 (2018): dev151407.

Wei, Qing, et al. "The BBSome controls IFT assembly and turnaround in cilia." Nature cell biology 14.9 (2012): 950.

Weis, Karsten. "Regulating access to the genome: nucleocytoplasmic transport throughout the cell cycle." Cell 112.4 (2003): 441-451.

Wheway, Gabrielle, Liliya Nazlamova, and John T. Hancock. "Signaling through the primary cilium." Frontiers in Cell and Developmental Biology 6 (2018): 8.

Williams, Corey L., et al. "MKS and NPHP modules cooperate to establish basal body/transition zone membrane associations and ciliary gate function during ciliogenesis." The Journal of cell biology 192.6 (2011): 1023-1041.

Wingfield, Jenna L., et al. "IFT trains in different stages of assembly queue at the ciliary base for consecutive release into the cilium." Elife 6 (2017): e26609.

Wloga, Dorota, et al. "Posttranslational modifications of tubulin and cilia." Cold Spring Harbor perspectives in biology 9.6 (2017): a028159.

Wood, Christopher R., and Joel L. Rosenbaum. "Proteins of the ciliary axoneme are found on cytoplasmic membrane vesicles during growth of cilia." Current Biology 24.10 (2014): 1114-1120.

Yang, T. Tony, et al. "Superresolution pattern recognition reveals the architectural map of the ciliary transition zone." Scientific reports 5 (2015): 14096.

Yang, Weidong, and Siegfried M. Musser. "Nuclear import time and transport efficiency depend on importin β concentration." The Journal of cell biology 174.7 (2006): 951-961.

Ye, Fan, Andrew R. Nager, and Maxence V. Nachury. "BBSome trains remove activated GPCRs from cilia by enabling passage through the transition zone." J Cell Biol217.5 (2018): 1847-1868.

Ye, Fan, et al. "Single molecule imaging reveals a major role for diffusion in the exploration of ciliary space by signaling receptors." Elife 2 (2013): e00654.

Yoder, Bradley K., Xiaoying Hou, and Lisa M. Guay-Woodford. "The polycystic kidney disease proteins, polycystin-1, polycystin-2, polaris, and cystin, are co-localized in renal cilia." Journal of the American Society of Nephrology 13.10 (2002): 2508-2516.

Yoshimura, S., J. Egerer, et al. Functional dissection of Rab GTPases involved in primary cilium formation. J Cell Biol. 1783:363-369, 2007.

Zweifel, Ludovit P., Ivan Shorubalko, and Roderick YH Lim. "Helium scanning transmission ion microscopy and electrical characterization of glass nanocapillaries with reproducible tip geometries." ACS nano 10.2 (2016): 1918-1925.

APPENDICES

APPENDIX A

PYTHON CODE TO GENERATE DATA IN FIGURE 18 - VARYING THE SIMULATED PRECISION SHOWS THE RESOLUTION LIMIT OF THE 3D TRANSFORMATION ALGORITHM

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 20 10:52:38 2017

@author: Andrew Ruba

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import math
import random
import csv
import os
from numpy.random import choice
import numpy as np
from scipy.optimize import curve_fit
import time
import matplotlib.pyplot as plt
from scipy import stats

## below arrays are for saving simulation data for statistical analysis
global gausslist
gausslist = []
global bimodallist
bimodallist = []
```

```
global bimodalmean
bimodalmean = []
global bimodalsd
bimodalsd = []
global bimodalheight
bimodalheight = []
global bimodalauc
bimodalauc = []

def sim(gui, PTNUM, RADIUS, PREC, ITER, BINSIZE, PERCERROR):
	def simulation(num_points, radius, dr, ss, mm):

		def area_fn(X):
			X = float(X)
			A = -(dr**2)*np.pi
			B = dr*2*np.pi
			return X*B+A

		def gauss_fn(x, s, m):
			a = area_fn(m)
			x = float(x)
			s = float(s)
			m = float(m)
			return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

		def combine(x):
			s = ss
			m = mm
			return (area_fn(x) * gauss_fn(x, s, m))

		##starting with perfect x,y and adding error
		xydata = []
		mm = mm + 0.00001
		while len(xydata) < num_points:
			theta = np.random.random()*360.0
			## precision distribution sampling
#		ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#		ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
			y_prec = np.random.normal(0.0, ss)
			z_prec = np.random.normal(0.0, ss)
			xydata.append((mm*np.cos(theta)+y_prec,
mm*np.sin(theta)+z_prec))

		with open('3d.csv', 'w') as csv_file:
```

```
		writer = csv.writer(csv_file, delimiter=',',  lineterminator='\n')
		writer.writerow(('y','z'))
		for i in xydata:
			writer.writerow(i)

	def gen_matrix(r, d_r):

		##'be' is short for bin edges
		if r%d_r > 0:
			be = range(0, r+r%d_r, d_r)
		else:
			be = range(0, r+d_r, d_r)

		matrix = []
		for i in range(len(be)-1):
			matrix.append([])

		x = 0
		for i in range(len(matrix)):
			for j in range(x):
				matrix[i].append(0)
			x += 1

		##generate areas of sections closest to x axis
		for i in range(len(matrix)):
			theta =
np.arccos(float(be[len(be)-2-i])/float(be[len(be)-1-i]))
			arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
			tri_area = 0.5 * float(be[len(be)-2-i]) * (np.sin(theta) *
float(be[len(be)-1-i]))
			matrix[i].append(4 * (arc_area - tri_area))

		##skipping factor
		x = 2
		##generate areas of layers going further out from x axis
		while len(matrix[0]) < len(matrix):
			for i in range(len(matrix) - len(matrix[0])):
				num = 0
				for j in range(len(matrix)):
					for k in range(len(matrix[i]) + 1):
						if j == i and k < len(matrix[i]):
							num += matrix[j][k]
						elif j > i:
```

```
                                                  num += matrix[j][k]
                              theta =
np.arccos(float(be[len(be)-1-x-i])/float(be[len(be)-1-i]))
                              arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
                              tri_area = 0.5 * float(be[len(be)-1-x-i]) *
(np.sin(theta) * float(be[len(be)-1-i]))
                              matrix[i].append(4 * (arc_area - tri_area) - num)
                         x += 1

                    return matrix

               def smoothdata(data, r, d_r):
                    """smoothds data with 3 moving window and takes abs value
average"""
                    smooth_data = []
                    r += 1

                    ##comment out for smoothing
                    smooth_data = []
                    for i in range(len(data)):
                         smooth_data.append(data[i])

                    ##adds + and - bins
                    final_smooth_data = []
                    for i in range(int(r/d_r)):
                         final_smooth_data.append(smooth_data[i] +
smooth_data[len(smooth_data)-1-i])

                    return list(reversed(final_smooth_data))

               def deconvolution(hv, be, r, d_r):
                    """hv = hist_values, be = bin_edges"""

                    density = []
                    matrix = gen_matrix(r, d_r)

                    while len(hv) > len(matrix):
                         hv.pop()
                    while len(matrix) > len(hv):
                         matrix.pop()

                    rev_hv = list(reversed(hv))
                    x = 0
```

```
		for i in range(len(rev_hv)):
			##calculate how much to subtract from bin
			density_sub = 0
			y = 0
			for j in range(x):
				density_sub += density[y] * matrix[j][i]
				y += 1

			##calculate final bin value
			density.append((rev_hv[i] - density_sub) / matrix[i][i])
			x += 1

		unrev_hv = list(reversed(density))

		smooth_data = []
		for i in range(len(unrev_hv)):
			if i == 0 or i == (len(unrev_hv) - 1):
				smooth_data.append(unrev_hv[i])
			else:
				smooth_data.append(np.average([unrev_hv[i-1],
unrev_hv[i], unrev_hv[i+1]]))

		return unrev_hv, smooth_data, hv

	def make_hist(data, r, d_r):

		hist_values, bin_edges = np.histogram(data, bins = 2 * int(r/d_r),
range = (-r, r))

		new_bin_edges = []
		for i in bin_edges:
			if i >= 0:
				new_bin_edges.append(i)

		new_hist_values = smoothdata(hist_values, r, d_r)

		return new_hist_values, new_bin_edges

	def csv_read(path):
		with open(path, 'rb') as csvfile:
			reader = csv.reader(csvfile, delimiter = ',')
			holdlist = []
			for row in reader:
				holdlist.append(float(row[1]))
```

```
			return holdlist

		jkl = []
		for y,z in xydata:
			jkl.append(y)

		radius = int(np.floor(radius/dr))*dr

		if num_points == PTNUM + 1:
			## decide the proper bin size
			minbinsize = 2
			binsizes = []
			binsizesdata = [[] for variable in range(1, int(PREC)+1)]
			gui.message.set('0% done calculating ideal bin size...')
			gui.update()
			for binoptimization in range(10):
				for binsize in range(1, int(PREC)+1):
					if binsize >= minbinsize:
						error = 0
						# print ('binsize ' + str(binsize))
						jkl = []
						mm = mm + 0.00001
						while len(jkl) < num_points-1:
							theta = np.random.random()*360.0
							## precision distribution sampling
#							ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#							ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
							y_prec = np.random.normal(0.0, ss)

jkl.append(mm*np.cos(theta)+y_prec)
						a,b = make_hist(jkl, radius, binsize)
						final_unsmooth, final_smooth, final_2d =
deconvolution(a, b, radius, binsize)
						holdlist = []
						addZero = False
						for val in list(reversed(final_unsmooth)):
							if not addZero:
								if val >= 0.0:
									holdlist.append(val)
								else:
									addZero = True
									holdlist.append(0.0)
							else:
								holdlist.append(0.0)
```

```
                                                  final_unsmooth = list(reversed(holdlist))
                                                  ##rescale ideal data
                                                  matrix = gen_matrix(radius, binsize)
                                                  newmatrix = []
                                                  for i in matrix:
                                                       newmatrix.append(list(reversed(i)))
                                                  matrix = list(reversed(newmatrix))
                                                  # print (a)
                                                  # print (final_unsmooth)
                                                  while len(a) > len(matrix):
                                                       a.pop()
                                                  while len(matrix) > len(a):
                                                       matrix.pop()

                                                  for ncol in range(len(matrix[0])):
                                                       binsub = 0.0
                                                       for mcol in range(len(matrix)):
                                                            binsub +=
float(matrix[mcol][ncol]*final_unsmooth[mcol])
                                                       try:
                                                            if a[ncol] != 0.0:
                                                                 # print (binsub)
                                                                 error +=
np.square(a[ncol] - binsub) / a[ncol]
                                                       except:
                                                            pass

                                                  popped = a.pop()
                                                  while popped == 0:
                                                       popped = a.pop()
                                                  binsizesdata[binsize-1].append((error,
len(a)+1,1-stats.chi2.cdf(error, len(a)+1),binsize))
                                             else:

binsizesdata[binsize-1].append((1000000.0,1,0.0,binsize))
                                             gui.message.set(str((binoptimization*10) + 10) + '
% done calculating ideal bin size...')
                                             gui.update()

                         finalbinsizes = []
                         for bintrial in range(len(binsizesdata)):
                              errhold = []
                              dfhold = []
                              pvalhold = []
```

```
                    binhold = []
                    for trial in range(len(binsizesdata[bintrial])):
                        chisq, df, pval, binsize = binsizesdata[bintrial][trial]
                        errhold.append(chisq)
                        dfhold.append(df)
                        pvalhold.append(pval)
                        binhold.append(binsize)
                    chisq = np.average(errhold)
                    df = np.round(np.average(dfhold))
                    pval = 1-stats.chi2.cdf(chisq,df)
                    binsize = binhold[0]
                    finalbinsizes.append((chisq,df,pval,binsize))
                # print (finalbinsizes)
                for binsizedata in finalbinsizes:
                    chisq, df, pval, binsize = binsizedata
                    if pval >= 0.95:
                        dr = binsize
                        break
                    else:
                        dr = int(PREC)

            a,b = make_hist(jkl, radius, dr)
            final = deconvolution(a,b,radius,dr)

            if num_points != PTNUM + 1:
                def gauss_fn(x, a, s, m):
                    return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

                def bimodal(x,mu1,sigma1,A1,mu2,sigma2,A2):
                    return gauss_fn(x, A1, sigma1, mu1)+gauss_fn(x, A2,
sigma2, mu2)

                try:
                    guess = [np.max(final[0]), ss, mm]
                    tempbins = list(range(int(dr/2), radius+int(dr/2), dr))
                    tempdensity = final[0]
                    holdlist = []
                    addZero = False
                    for val in list(reversed(tempdensity)):
                        if not addZero:
                            if val >= 0.0:
                                holdlist.append(val)
                            else:
                                addZero = True
```

```
                        holdlist.append(0.0)
            else:
                holdlist.append(0.0)
        tempdensity = list(reversed(holdlist))
        while len(tempdensity) > len(tempbins):
            tempdensity.pop()
        while len(tempbins) > len(tempdensity):
            tempbins.pop()
        revtempbins = list(np.negative(list(reversed(tempbins))))
        revtempdensity = list(reversed(tempdensity))
        bins = revtempbins + tempbins
        density = revtempdensity + tempdensity
        params, var = curve_fit(gauss_fn, bins, density, p0 = guess)
        params_gauss = np.abs(params)
        ## computes 1 SD errors
        var_gauss = np.sqrt(np.diag(var))

        def frange(beg, end, step):
            f_range = []
            while beg < end - (step/2.0):
                f_range.append(beg)
                beg += step
            return f_range

        guess = [-mm, ss, np.max(final[0]), mm, ss,
np.max(final[0])]

        tempbins = frange(dr/2.0, radius, dr)

        tempdensity = final[0]
        holdlist = []
        addZero = False
        for val in list(reversed(tempdensity)):
            if not addZero:
                if val >= 0.0:
                    holdlist.append(val)
                else:
                    addZero = True
                    holdlist.append(0.0)
            else:
                holdlist.append(0.0)
        tempdensity = list(reversed(holdlist))
        while len(tempdensity) > len(tempbins):
            tempdensity.pop()
```

```
					while len(tempbins) > len(tempdensity):
						tempbins.pop()
					revtempbins = list(np.negative(list(reversed(tempbins))))
					revtempdensity = list(reversed(tempdensity))
					bins = revtempbins + tempbins
					density = revtempdensity + tempdensity
					params, var = curve_fit(bimodal, bins, density, p0 = guess)
					params = np.abs(params)
					## computes 1 SD errors
					var = np.sqrt(np.diag(var))
					## average paramters
					stdev = np.average((params[1], params[4]))
					mean = np.average((params[0], params[3]))
					height = np.average((params[2], params[5]))
					stdev_e = np.average((var[1], var[4]))
					mean_e = np.average((var[0], var[3]))
					height_e = np.average((var[2], var[5]))
					params_bimodal = [height, stdev, mean]
					var_bimodal = [height_e, stdev_e, mean_e]

					## uncomment following for comparing central vs.
peripheral peak fitting errors
#				bimodalmean.append(params_gauss[0])
					bimodalmean.append(mean)
#				bimodalmean.append(tempdensity)
					bimodalsd.append(stdev)
					bimodalheight.append(height)
					auc = 0.0
					step = mean - 5.0*stdev
					while step < mean + 5.0*stdev:
						auc+=0.01*gauss_fn(step,height,stdev,mean)
						step += 0.01
					bimodalauc.append(auc)
#				bimodallist.append(var_bimodal[1])
					gausslist.append(var_gauss[1])
			#		if np.sum(var_bimodal) < np.sum(var_gauss):
					params = params_bimodal
					var = var_bimodal
			#		else:
#				params = params_gauss
#				var = var_gauss
				except RuntimeError:
					params = []
					var = []
```

```
                hist_mids = []
                for i in range(len(b)-1):
                        hist_mids.append(np.average((b[i],b[i+1])))
                norm_values = []
                for i in final[0]:
                        norm_values.append(i/np.max(final[0]))
                return params, var, norm_values, hist_mids, dr

        else:
                return dr

pt_min = PTNUM
pt_max = PTNUM
rt_min = RADIUS
rt_max = RADIUS
prec_min = PREC
prec_max = PREC
iterations = ITER
PREC = float(PREC)
one_diff = []
perc_err = PERCERROR*0.01


def roundup(x):
        val = int(math.ceil(x / 10.0)) * 10
        if val >= 30:
                return val
        else:
                return 30

ptlist = range(pt_min, pt_max+100, 100)
for pt in ptlist:
        for rt in range(rt_min, rt_max+1, 1):
                for prec in range(prec_min, prec_max+1, 1):
                        prec = prec+0.000001
                        xrng = roundup(2.0*rt + prec*5.0)
                        # DR = simulation(pt+1, xrng, BINSIZE, float(prec),
float(rt))
                        ## uncomment below to manually set bin size
                        DR = PTNUM
                        # print ('ideal bin size: '+ str(DR))
                        p, v, d, h_m, DR = simulation(1000000, xrng, DR,
float(prec), float(rt))
                        # print (p)
```

```
                        a, s, m = p
                        corr = m - float(rt)
##outlier detection
_mean = np.mean(bimodalmean)
_stdev = np.std(bimodalmean)
togui = []
for i in bimodalmean:
        if i >= _mean-3*_stdev and i <= _mean+3*_stdev:
                togui.append(i)
final_data = []
for i,j in zip(h_m, d):
        final_data.append((i,j))
        final_data.append((-i,j))
final_data.sort()
with open('precision_results.csv', 'w') as csv_file:
        writer = csv.writer(csv_file, delimiter=',',  lineterminator='\n')
        writer.writerow(('bin middle','normalized density'))
        for i in final_data:
                writer.writerow(i)
return "Output written to precision_results.csv"
```

APPENDIX B

PYTHON CODE TO GENERATE DATA IN FIGURE 19 - VARYING THE NUMBER OF SIMULATED POINTS SHOWS THE EFFECT OF SAMPLING ERROR ON THE PEAK FITTING OF THE 3D TRANSFORMED DATA

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 20 10:52:38 2017

@author: Andrew Ruba

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import math
import random
import csv
import os
from numpy.random import choice
import numpy as np
from scipy.optimize import curve_fit
import time
import matplotlib.pyplot as plt
from scipy import stats

## below arrays are for saving simulation data for statistical analysis
global gausslist
gausslist = []
global bimodallist
bimodallist = []
global bimodalmean
bimodalmean = []
```

```
global bimodalsd
bimodalsd = []
global bimodalheight
bimodalheight = []
global bimodalauc
bimodalauc = []

def sim(gui, PTNUM, RADIUS, PREC, ITER, BINSIZE, PERCERROR):
        def simulation(num_points, radius, dr, ss, mm):

                def area_fn(X):
                        X = float(X)
                        A = -(dr**2)*np.pi
                        B = dr*2*np.pi
                        return X*B+A

                def gauss_fn(x, s, m):
                        a = area_fn(m)
                        x = float(x)
                        s = float(s)
                        m = float(m)
                        return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

                def combine(x):
                        s = ss
                        m = mm
                        return (area_fn(x) * gauss_fn(x, s, m))

                ##starting with perfect x,y and adding error
                xydata = []
                mm = mm + 0.00001
                while len(xydata) < num_points:
                        theta = np.random.random()*360.0
                        ## precision distribution sampling
#          ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#          ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
                        y_prec = np.random.normal(0.0, ss)
                        z_prec = np.random.normal(0.0, ss)
                        xydata.append((mm*np.cos(theta)+y_prec,
mm*np.sin(theta)+z_prec))

                def gen_matrix(r, d_r):

                        ##'be' is short for bin edges
```

```
if r%d_r > 0:
    be = range(0, r+r%d_r, d_r)
else:
    be = range(0, r+d_r, d_r)

matrix = []
for i in range(len(be)-1):
    matrix.append([])

x = 0
for i in range(len(matrix)):
    for j in range(x):
        matrix[i].append(0)
    x += 1

##generate areas of sections closest to x axis
for i in range(len(matrix)):
    theta =
np.arccos(float(be[len(be)-2-i])/float(be[len(be)-1-i]))
    arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
    tri_area = 0.5 * float(be[len(be)-2-i]) * (np.sin(theta) *
float(be[len(be)-1-i]))
    matrix[i].append(4 * (arc_area - tri_area))

##skipping factor
x = 2
##generate areas of layers going further out from x axis
while len(matrix[0]) < len(matrix):
    for i in range(len(matrix) - len(matrix[0])):
        num = 0
        for j in range(len(matrix)):
            for k in range(len(matrix[i]) + 1):
                if j == i and k < len(matrix[i]):
                    num += matrix[j][k]
                elif j > i:
                    num += matrix[j][k]
        theta =
np.arccos(float(be[len(be)-1-x-i])/float(be[len(be)-1-i]))
        arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
        tri_area = 0.5 * float(be[len(be)-1-x-i]) *
(np.sin(theta) * float(be[len(be)-1-i]))
        matrix[i].append(4 * (arc_area - tri_area) - num)
```

```
            x += 1

        return matrix

    def smoothdata(data, r, d_r):
        """smoothds data with 3 moving window and takes abs value
average"""
        smooth_data = []
        r += 1

        ##comment out for smoothing
        smooth_data = []
        for i in range(len(data)):
            smooth_data.append(data[i])

        ##adds + and - bins
        final_smooth_data = []
        for i in range(int(r/d_r)):
            final_smooth_data.append(smooth_data[i] +
smooth_data[len(smooth_data)-1-i])

        return list(reversed(final_smooth_data))

    def deconvolution(hv, be, r, d_r):
        """hv = hist_values, be = bin_edges"""

        density = []
        matrix = gen_matrix(r, d_r)

        while len(hv) > len(matrix):
            hv.pop()
        while len(matrix) > len(hv):
            matrix.pop()

        rev_hv = list(reversed(hv))
        x = 0
        for i in range(len(rev_hv)):
            ##calculate how much to subtract from bin
            density_sub = 0
            y = 0
            for j in range(x):
                density_sub += density[y] * matrix[j][i]
                y += 1
```

```
                        ##calculate final bin value
                        density.append((rev_hv[i] - density_sub) / matrix[i][i])
                        x += 1

                unrev_hv = list(reversed(density))

                smooth_data = []
                for i in range(len(unrev_hv)):
                        if i == 0 or i == (len(unrev_hv) - 1):
                                smooth_data.append(unrev_hv[i])
                        else:
                                smooth_data.append(np.average([unrev_hv[i-1],
unrev_hv[i], unrev_hv[i+1]]))

                return unrev_hv, smooth_data, hv

        def make_hist(data, r, d_r):

                hist_values, bin_edges = np.histogram(data, bins = 2 * int(r/d_r),
range = (-r, r))

                new_bin_edges = []
                for i in bin_edges:
                        if i >= 0:
                                new_bin_edges.append(i)

                new_hist_values = smoothdata(hist_values, r, d_r)

                return new_hist_values, new_bin_edges

                return new_hist_values, new_bin_edges

        def csv_read(path):
                with open(path, 'rb') as csvfile:
                        reader = csv.reader(csvfile, delimiter = ',')
                        holdlist = []
                        for row in reader:
                                holdlist.append(float(row[1]))
                return holdlist

        jkl = []
        for y,z in xydata:
                jkl.append(y)
```

```
		radius = int(np.floor(radius/dr))*dr

		if num_points == PTNUM + 1:
			## decide the proper bin size
			minbinsize = 5
			binsizes = []
			binsizesdata = [[] for variable in range(1, int(PREC)+1)]
			gui.message.set('0% done calculating ideal bin size...')
			gui.update()
			for binoptimization in range(10):
				for binsize in range(1, int(PREC)+1):
					if binsize >= minbinsize:
						error = 0
						# print ('binsize ' + str(binsize))
						jkl = []
						mm = mm + 0.00001
						while len(jkl) < num_points-1:
							theta = np.random.random()*360.0
							## precision distribution sampling
#				ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#				ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
							y_prec = np.random.normal(0.0, ss)

jkl.append(mm*np.cos(theta)+y_prec)
						a,b = make_hist(jkl, radius, binsize)
						final_unsmooth, final_smooth, final_2d =
deconvolution(a, b, radius, binsize)
						holdlist = []
						addZero = False
						for val in list(reversed(final_unsmooth)):
							if not addZero:
								if val >= 0.0:
									holdlist.append(val)
								else:
									addZero = True
									holdlist.append(0.0)
							else:
								holdlist.append(0.0)
						final_unsmooth = list(reversed(holdlist))
						##rescale ideal data
						matrix = gen_matrix(radius, binsize)
						newmatrix = []
						for i in matrix:
							newmatrix.append(list(reversed(i)))
```

```
								matrix = list(reversed(newmatrix))
								# print (a)
								# print (final_unsmooth)
								while len(a) > len(matrix):
									a.pop()
								while len(matrix) > len(a):
									matrix.pop()

								for ncol in range(len(matrix[0])):
									binsub = 0.0
									for mcol in range(len(matrix)):
										binsub += 
float(matrix[mcol][ncol]*final_unsmooth[mcol])
									try:
										if a[ncol] != 0.0:
											# print (binsub)
											error += 
np.square(a[ncol] - binsub) / a[ncol]
									except:
										pass

								popped = a.pop()
								while popped == 0:
									popped = a.pop()
								binsizesdata[binsize-1].append((error, 
len(a)+1,1-stats.chi2.cdf(error, len(a)+1),binsize))
							else:

binsizesdata[binsize-1].append((1000000.0,1,0.0,binsize))
							gui.message.set(str((binoptimization*10) + 10) + '
% done calculating ideal bin size...')
							gui.update()

					finalbinsizes = []
					for bintrial in range(len(binsizesdata)):
						errhold = []
						dfhold = []
						pvalhold = []
						binhold = []
						for trial in range(len(binsizesdata[bintrial])):
							chisq, df, pval, binsize = binsizesdata[bintrial][trial]
							errhold.append(chisq)
							dfhold.append(df)
							pvalhold.append(pval)
```

```
                    binhold.append(binsize)
                chisq = np.average(errhold)
                df = np.round(np.average(dfhold))
                pval = 1-stats.chi2.cdf(chisq,df)
                binsize = binhold[0]
                finalbinsizes.append((chisq,df,pval,binsize))
            # print (finalbinsizes)
            for binsizedata in finalbinsizes:
                chisq, df, pval, binsize = binsizedata
                if pval >= 0.95:
                    dr = binsize
                    break
                else:
                    dr = int(PREC)

        a,b = make_hist(jkl, radius, dr)
        final = deconvolution(a,b,radius,dr)

        if num_points != PTNUM + 1:
            def gauss_fn(x, a, s, m):
                return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

            def bimodal(x,mu1,sigma1,A1,mu2,sigma2,A2):
                return gauss_fn(x, A1, sigma1, mu1)+gauss_fn(x, A2,
sigma2, mu2)

            try:
                guess = [np.max(final[0]), ss, mm]
                tempbins = list(range(int(dr/2), radius+int(dr/2), dr))
                tempdensity = final[0]
                holdlist = []
                addZero = False
                for val in list(reversed(tempdensity)):
                    if not addZero:
                        if val >= 0.0:
                            holdlist.append(val)
                        else:
                            addZero = True
                            holdlist.append(0.0)
                    else:
                        holdlist.append(0.0)
                tempdensity = list(reversed(holdlist))
                while len(tempdensity) > len(tempbins):
                    tempdensity.pop()
```

```
while len(tempbins) > len(tempdensity):
	tempbins.pop()
revtempbins = list(np.negative(list(reversed(tempbins))))
revtempdensity = list(reversed(tempdensity))
bins = revtempbins + tempbins
density = revtempdensity + tempdensity
params, var = curve_fit(gauss_fn, bins, density, p0 = guess)
params_gauss = np.abs(params)
## computes 1 SD errors
var_gauss = np.sqrt(np.diag(var))

def frange(beg, end, step):
	f_range = []
	while beg < end - (step/2.0):
		f_range.append(beg)
		beg += step
	return f_range

guess = [-mm, ss, np.max(final[0]), mm, ss,
np.max(final[0])]

tempbins = frange(dr/2.0, radius, dr)

tempdensity = final[0]
bimodallist.append(final[0])
holdlist = []
addZero = False
for val in list(reversed(tempdensity)):
	if not addZero:
		if val >= 0.0:
			holdlist.append(val)
		else:
			addZero = True
			holdlist.append(0.0)
	else:
		holdlist.append(0.0)
tempdensity = list(reversed(holdlist))
while len(tempdensity) > len(tempbins):
	tempdensity.pop()
while len(tempbins) > len(tempdensity):
	tempbins.pop()
revtempbins = list(np.negative(list(reversed(tempbins))))
revtempdensity = list(reversed(tempdensity))
bins = revtempbins + tempbins
```

```
                    density = revtempdensity + tempdensity
                    params, var = curve_fit(bimodal, bins, density, p0 = guess)
                    params = np.abs(params)
                    ## computes 1 SD errors
                    var = np.sqrt(np.diag(var))
                    ## average paramters
                    stdev = np.average((params[1], params[4]))
                    mean = np.average((params[0], params[3]))
                    height = np.average((params[2], params[5]))
                    stdev_e = np.average((var[1], var[4]))
                    mean_e = np.average((var[0], var[3]))
                    height_e = np.average((var[2], var[5]))
                    params_bimodal = [height, stdev, mean]
                    var_bimodal = [height_e, stdev_e, mean_e]

                    ## uncomment following for comparing central vs.
peripheral peak fitting errors
#           bimodalmean.append(params_gauss[0])
                    bimodalmean.append(mean)
#           bimodalmean.append(tempdensity)
                    bimodalsd.append(stdev)
                    bimodalheight.append(height)
                    auc = 0.0
                    step = mean - 5.0*stdev
                    while step < mean + 5.0*stdev:
                        auc+=0.01*gauss_fn(step,height,stdev,mean)
                        step += 0.01
                    bimodalauc.append(auc)
#           bimodallist.append(var_bimodal[1])
                    gausslist.append(var_gauss[1])
        #      if np.sum(var_bimodal) < np.sum(var_gauss):
                    params = params_bimodal
                    var = var_bimodal
        #      else:
                    # params = params_gauss
                    # var = var_gauss
                except RuntimeError:
                    params = []
                    var = []

                return params, var, final[0], dr

        else:
                return dr
```

```
pt_min = PTNUM
pt_max = PTNUM
rt_min = RADIUS
rt_max = RADIUS
prec_min = PREC
prec_max = PREC
iterations = ITER
PREC = float(PREC)
one_diff = []
perc_err = PERCERROR*0.01


def roundup(x):
    val = int(math.ceil(x / 10.0)) * 10
    if val >= 30:
        return val
    else:
        return 30

ptlist = range(pt_min, pt_max+100, 100)
for pt in ptlist:
    for rt in range(rt_min, rt_max+1, 1):
        for prec in range(prec_min, prec_max+1, 1):
            one_hold = 0
            two_hold = 0
            three_hold = 0
            four_hold = 0
            five_hold = 0
            six_hold = 0
            seven_hold = 0
            eight_hold = 0
            nine_hold = 0
            ten_hold = 0
            prec = prec+0.000001
            xrng = roundup(rt + prec*5.0)
            DR = simulation(pt+1, xrng, BINSIZE, float(prec),
float(rt))
            ## uncomment below to manually set bin size
            # DR = 10
            # print ('ideal bin size: '+ str(DR))
            p, v, d, DR = simulation(1000000, xrng, DR, float(prec),
float(rt))
            # print (p)
```

```
				a, s, m = p
				corr = m - float(rt)
				# print ('ideal peak location: ', corr, rt + corr + PREC, rt +
corr - PREC, a, s)

				for i in range(iterations):
					gui.message.set(str(100.0*i/iterations) + '% done
simulation.')

					gui.update()
					p, v, d, DR = simulation(pt, xrng, DR, float(prec),
float(rt))

					if p != []:
						a, s, m = p
						print (m)
						a_e, s_e, m_e = v
						if m + (0.1*PREC) >= rt + corr and m -
(0.1*PREC) <= rt + corr:

							one_hold += 1
						if m + (0.2*PREC) >= rt + corr and m -
(0.2*PREC) <= rt + corr:

							two_hold += 1
						if m + (0.3*PREC) >= rt + corr and m -
(0.3*PREC) <= rt + corr:

							three_hold += 1
						if m + (0.4*PREC) >= rt + corr and m -
(0.4*PREC) <= rt + corr:

							four_hold += 1
						if m + (0.5*PREC) >= rt + corr and m -
(0.5*PREC) <= rt + corr:

							five_hold += 1
						if m + (0.6*PREC) >= rt + corr and m -
(0.6*PREC) <= rt + corr:

							six_hold += 1
						if m + (0.7*PREC) >= rt + corr and m -
(0.7*PREC) <= rt + corr:

							seven_hold += 1
						if m + (0.8*PREC) >= rt + corr and m -
(0.8*PREC) <= rt + corr:

							eight_hold += 1
						if m + (0.9*PREC) >= rt + corr and m -
(0.9*PREC) <= rt + corr:

							nine_hold += 1
						if m + (1.0*PREC) >= rt + corr and m -
(1.0*PREC) <= rt + corr:

							ten_hold += 1
```

```
    ##printing out average 3d and sd of each bin
    templist = []
    bimodallist_temp = [np.array(i)/np.max(i) for i in bimodallist]
    for i in range(len(bimodallist_temp[0])):
        templist.append([])
        for j in bimodallist_temp:
            templist[i].append(j[i])
    print('average 3d')
    for i in templist:
        print(np.average(i))
    print('sd 3d')
    for i in templist:
        print(np.std(i))
    ##outlier detection
    _mean = np.mean(bimodalmean)
    _stdev = np.std(bimodalmean)
    togui = []
    for i in bimodalmean:
        if i >= _mean-3*_stdev and i <= _mean+3*_stdev:
            togui.append(i)

    return "%.3f" % np.std(togui), 100.0*one_hold/float(iterations),
100.0*two_hold/float(iterations), 100.0*three_hold/float(iterations),
100.0*four_hold/float(iterations), 100.0*five_hold/float(iterations),
100.0*six_hold/float(iterations), 100.0*seven_hold/float(iterations),
100.0*eight_hold/float(iterations), 100.0*nine_hold/float(iterations),
100.0*ten_hold/float(iterations)
```

APPENDIX C

PYTHON CODE TO GENERATE DATA IN FIGURE 21 - ROUTE CATEGORIZATION SIMULATION

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 17 11:06:15 2017

@author: tuf61935
"""


import tkinter as tk
import csv
import random
import os
import tkinter.filedialog as tkfd
import sys
import numpy as np
import math

class categorization(tk.Frame):

	def __init__(self, master = None):
		tk.Frame.__init__(self, master)
		self.message = tk.StringVar()
		self.pack()
		self.createwidgets()
		self.master = master

	def createwidgets(self):
		self.TOPCONTAINER = tk.Frame(self)
		self.TOPCONTAINER.pack({'side': 'top'})

		self.LEFTTOPCONTAINER = tk.Frame(self.TOPCONTAINER)
		self.LEFTTOPCONTAINER.pack({'side': 'left'})

		self.BOTTOMCONTAINER = tk.Frame(self)
		self.BOTTOMCONTAINER.pack({'side': 'bottom'})

		self.PERCERR = tk.Button(self.LEFTTOPCONTAINER)
		self.PERCERR['text'] = 'Categorize route'
```

```
        self.PERCERR['command'] = lambda: self.select_folder()
        self.PERCERR.pack({'side': 'top'})

        self.QUIT = tk.Button(self.LEFTTOPCONTAINER)
        self.QUIT['text'] = 'Quit'
        self.QUIT['fg'] = red
        self.QUIT['command'] = lambda: self.quit()
        self.QUIT.pack({'side': 'top'})

        self.BINSIZECONTAINER = tk.Frame(self.TOPCONTAINER)
        self.BINSIZECONTAINER.pack({'side': 'top'})

        self.BINSIZELABEL = tk.Label(self.BINSIZECONTAINER)
        self.BINSIZELABEL['text'] = 'ideal bin size (nm): '
        self.BINSIZELABEL.pack({'side': 'left'})

        self.BIN_SIZE = tk.Entry(self.BINSIZECONTAINER)
        self.BIN_SIZE['width'] = 3
        self.BIN_SIZE.insert(0, '10')
        self.BIN_SIZE.pack({'side': 'left'})

        self.PTNUMCONTAINER = tk.Frame(self.TOPCONTAINER)
        self.PTNUMCONTAINER.pack({'side': 'top'})

        self.PTNUMLABEL = tk.Label(self.PTNUMCONTAINER)
        self.PTNUMLABEL['text'] = 'ground truth point #: '
        self.PTNUMLABEL.pack({'side': 'left'})

        self.PTNUM = tk.Entry(self.PTNUMCONTAINER)
        self.PTNUM['width'] = 8
        self.PTNUM.insert(0, '10000')
        self.PTNUM.pack({'side': 'left'})

        self.MESSAGE = tk.Label(self.BOTTOMCONTAINER)
        self.MESSAGE['textvariable'] = self.message
        self.message.set('Click "categorize route" to begin')
        self.MESSAGE.pack({'side': 'right'})

    def categorization_algorithm(self, DATA, BINSIZE, PTNUM):

        def roundup(x):
            val = int(math.ceil(x / BINSIZE)) * BINSIZE
            if val >= 50:
                return val
```

```
			else:
				return 50
		global radius
		global peripheral_radius
		global precision
		global num_points
		## dist types: 'peripheral', 'uniform', 'central', 'bimodal'
		global dist_type
		global yrng
		yrng = roundup(np.max([np.abs(float(y)) for x,y in DATA]))
		global smooth
		## set to True for +/- 1 bin smoothing window
		smooth = False
		## set to True to remove high error values from 3D
		global allZero
		allZero = True
		## each bimodal distribution has equal density in central and outer peak
		global bimodal_factor
		npc_range = yrng
		sigma_range = yrng
		dr = BINSIZE

		def smoothdata(data, r, d_r):
			"""smoothds data with 3 moving window and takes abs value
average, related to smooth variable"""
			smooth_data = []
			r += 1

			smooth_data = []
			for i in range(len(data)):
				smooth_data.append(data[i])

			## adds + and - bins
			final_smooth_data = []
			for i in range(int(r/d_r)):
				final_smooth_data.append(smooth_data[i] +
smooth_data[len(smooth_data)-1-i])
			return list(reversed(final_smooth_data))

		def make_hist(data, r, d_r):
			r = int(r)
			hist_values, bin_edges = np.histogram(data, bins = int(2 * r/d_r),
range = (-r, r))
			new_bin_edges = []
```

```
        for i in bin_edges:
            if i >= 0:
                new_bin_edges.append(i)
        new_hist_values = smoothdata(hist_values, r, d_r)
        return new_hist_values, new_bin_edges

    def gen_matrix(r, d_r):

        r = int(r)
        if r%d_r > 0:
            be = range(0, r+r%d_r, d_r)
        else:
            be = range(0, r+d_r, d_r)

        matrix = []
        for i in range(len(be)-1):
            matrix.append([])

        x = 0
        for i in range(len(matrix)):
            for j in range(x):
                matrix[i].append(0)
            x += 1

        ## generate areas of sections closest to x axis
        for i in range(len(matrix)):
            theta =
np.arccos(float(be[len(be)-2-i])/float(be[len(be)-1-i]))
            arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
            tri_area = 0.5 * float(be[len(be)-2-i]) * (np.sin(theta) *
float(be[len(be)-1-i]))
            matrix[i].append(4 * (arc_area - tri_area))

        ## skipping factor
        x = 2
        ## generate areas of layers going further out from x axis
        while len(matrix[0]) < len(matrix):
            for i in range(len(matrix) - len(matrix[0])):
                num = 0
                for j in range(len(matrix)):
                    for k in range(len(matrix[i]) + 1):
                        if j == i and k < len(matrix[i]):
                            num += matrix[j][k]
```

```
                                        elif j > i:
                                            num += matrix[j][k]
                                theta =
np.arccos(float(be[len(be)-1-x-i])/float(be[len(be)-1-i]))
                                arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
                                tri_area = 0.5 * float(be[len(be)-1-x-i]) *
(np.sin(theta) * float(be[len(be)-1-i]))
                                matrix[i].append(4 * (arc_area - tri_area) - num)
                            x += 1

                        return matrix

                def deconvolution(hv, be, r, d_r):
                        """hv = hist_values, be = bin_edges"""

                        density = []
                        matrix = gen_matrix(r, d_r)
                        while len(hv) > len(matrix):
                            hv.pop()
                        while len(matrix) > len(hv):
                            matrix.pop()

                        rev_hv = list(reversed(hv))
                        x = 0
                        for i in range(len(rev_hv)):
                            ## calculate how much to subtract from bin
                            density_sub = 0
                            y = 0
                            for j in range(x):
                                density_sub += density[y] * matrix[j][i]
                                y += 1

                            ## calculate final bin value
                            density.append((rev_hv[i] - density_sub) / matrix[i][i])
                            x += 1

                        unrev_hv = list(reversed(density))
                        smooth_data = []
                        for i in range(len(unrev_hv)):
                            if i == 0 or i == (len(unrev_hv) - 1):
                                smooth_data.append(unrev_hv[i])
                            else:
```

```
                                        smooth_data.append(np.average([unrev_hv[i-1],
unrev_hv[i], unrev_hv[i+1]]))

                        return unrev_hv, smooth_data, hv

                ## simulate points from all four ground truth models
                def gauss_dist():
                        if dist_type != 'bimodal':
                                yzdata = []
                                while len(yzdata) < num_points:
                                        theta = np.random.random()*360.0
                                        y_prec = np.random.normal(0.0, precision)
                                        z_prec = np.random.normal(0.0, precision)
                                        yzdata.append((radius*np.cos(theta)+y_prec,
radius*np.sin(theta)+z_prec))
                        else:
                                yzdata = []
                                while len(yzdata) < num_points:
                                        theta = np.random.random()*360.0
                                        y_prec = np.random.normal(0.0, precision)
                                        z_prec = np.random.normal(0.0, precision)
                                        if random.random() < bimodal_factor:
                                                yzdata.append((0.0*np.cos(theta)+y_prec,
0.0*np.sin(theta)+z_prec))
                                        else:

yzdata.append((peripheral_radius*np.cos(theta)+y_prec,
peripheral_radius*np.sin(theta)+z_prec))
                        return yzdata

                def uniform_dist():
                        yzdata = []
                        while len(yzdata) < num_points:
                                rand_radius = random.random()
                                if random.random() < rand_radius:
                                        rand_radius = rand_radius * peripheral_radius
                                        theta = np.random.random()*360.0
                                        y_prec = np.random.normal(0.0, precision)
                                        z_prec = np.random.normal(0.0, precision)
                                        yzdata.append((rand_radius*np.cos(theta)+y_prec,
rand_radius*np.sin(theta)+z_prec))
                        return yzdata

                ## experimental simulation
```

```
exp_trials = []
# for i in range(num_trials):
# if dist_type == 'uniform':
        # exp = uniform_dist()
# else:
        # exp = gauss_dist()
exp = [float(y) for x,y in DATA]
a,b = make_hist(exp, yrng, dr)
final_unsmooth_exp, final_smooth_exp, final_2d_exp = deconvolution(a,
b, yrng, dr)
temp = []
addZero = False
if smooth:
        final_unsmooth_exp = [i for i in final_smooth_exp]
for i in list(reversed(final_unsmooth_exp)):
        if not addZero:
                if i >= 0.0:
                        temp.append(i)
                else:
                        # if allZero:
                                # addZero = True
                        temp.append(0.0)
        else:
                temp.append(0.0)
final_unsmooth_exp = [i/np.max(temp) for i in list(reversed(temp))]
for asdf in final_unsmooth_exp:
        print (asdf)
## outlier detection
bin_edges = []
for i in range(len(b)-1):
        bin_edges.append((b[i]+b[i+1])/2)
_mean = np.average(bin_edges, weights=final_unsmooth_exp)
_stdev =  math.sqrt(np.average((bin_edges-_mean)**2,
weights=final_unsmooth_exp))
for i in range(len(bin_edges)):
        if bin_edges[i] < _mean-2*_stdev or bin_edges[i] >
_mean+2*_stdev:
                final_unsmooth_exp[i] = 0.0
print (bin_edges)
print (final_unsmooth_exp)
exp_trials.append(final_unsmooth_exp)

all_ideals = {'peripheral':{}, 'central':{}, 'bimodal':{}, 'uniform':{}}
for all in ['peripheral','bimodal','uniform']:
```

```
        for r in range(1,npc_range+1):
            all_ideals[all][str(r)] = {}
    all_ideals['central']['0'] = {}

    ## calculate ground truth models for SAR
    ## peripheral
    num_points = PTNUM
    # hold_radius = radius
    # radius = peripheral_radius
    # hold_type = dist_type
    dist_type = 'peripheral'
    for r in range(1,npc_range+1):
        for p in range(1, sigma_range+1):
            print (r,p, dist_type)
            radius = r
            precision = p
            ideal = gauss_dist()
            ideal = [float(y) for y,z in ideal]
            a,b = make_hist(ideal, yrng, dr)
            final_unsmooth_ideal, final_smooth_ideal, final_2d_ideal
= deconvolution(a, b, yrng, dr)
            temp = []
            addZero = False
            if smooth:
                final_unsmooth_ideal = [i for i in
final_smooth_ideal]
            for i in list(reversed(final_unsmooth_ideal)):
                if not addZero:
                    if i >= 0.0:
                        temp.append(i)
                    else:
                        temp.append(0.0)
                else:
                    temp.append(0.0)
            final_unsmooth_ideal_peripheral = [i/np.max(temp) for i in
list(reversed(temp))]
            all_ideals['peripheral'][str(r)][str(p)] = [i/np.max(temp) for i
in list(reversed(temp))]

    # ## uniform
    for r in range(1,npc_range+1):
        for p in range(1, sigma_range+1):
            print (r, p, 'uniform')
            peripheral_radius = r
```

```
                precision = p
                ideal = uniform_dist()
                ideal = [float(y) for y,z in ideal]
                a,b = make_hist(ideal, yrng, dr)
                final_unsmooth_ideal, final_smooth_ideal, final_2d_ideal
= deconvolution(a, b, yrng, dr)
                temp = []
                addZero = False
                if smooth:
                    final_unsmooth_ideal = [i for i in
final_smooth_ideal]
                for i in list(reversed(final_unsmooth_ideal)):
                    if not addZero:
                        if i >= 0.0:
                            temp.append(i)
                        else:
                            temp.append(0.0)
                    else:
                        temp.append(0.0)
                final_unsmooth_ideal_uniform = [i/np.max(temp) for i in
list(reversed(temp))]
                all_ideals['uniform'][str(r)][str(p)] = [i/np.max(temp) for i
in list(reversed(temp))]

        ## bimodal
        dist_type = 'bimodal'
        for r in range(1,npc_range+1):
            for p in range(1, sigma_range+1):
                print (r,p, dist_type)
                peripheral_radius = r
                precision = p
                ##calculate bimodal_factor
                inner_area = np.pi*dr**2.0
                outer_area =
np.pi*(((peripheral_radius+dr/2.0)**2.0)-((peripheral_radius-dr/2.0)**2.0))
                bimodal_factor = inner_area / (inner_area + outer_area)
                ideal = gauss_dist()
                ideal = [float(y) for y,z in ideal]
                a,b = make_hist(ideal, yrng, dr)
                final_unsmooth_ideal, final_smooth_ideal, final_2d_ideal
= deconvolution(a, b, yrng, dr)
                temp = []
                addZero = False
                if smooth:
```

```
                    final_unsmooth_ideal = [i for i in
final_smooth_ideal]
                for i in list(reversed(final_unsmooth_ideal)):
                    if not addZero:
                        if i >= 0.0:
                            temp.append(i)
                        else:
                            temp.append(0.0)
                    else:
                        temp.append(0.0)
                final_unsmooth_ideal_bimodal = [i/np.max(temp) for i in
list(reversed(temp))]
                all_ideals['bimodal'][str(r)][str(p)] = [i/np.max(temp) for i
in list(reversed(temp))]

        ## central
        radius = 0.0000001
        dist_type = 'central'
        print (radius, dist_type)
        for p in range(1, sigma_range+1):
            print (p, dist_type)
            precision = p
            ideal = gauss_dist()
            ideal = [float(y) for y,z in ideal]
            a,b = make_hist(ideal, yrng, dr)
            final_unsmooth_ideal, final_smooth_ideal, final_2d_ideal =
deconvolution(a, b, yrng, dr)
            temp = []
            addZero = False
            if smooth:
                final_unsmooth_ideal = [i for i in final_smooth_ideal]
            for i in list(reversed(final_unsmooth_ideal)):
                if not addZero:
                    if i >= 0.0:
                        temp.append(i)
                    else:
                        temp.append(0.0)
                else:
                    temp.append(0.0)
            final_unsmooth_ideal_central = [i/np.max(temp) for i in
list(reversed(temp))]
            all_ideals['central']['0'][str(p)] = [i/np.max(temp) for i in
list(reversed(temp))]
```

```
		num_correct = {'peripheral':{}, 'central':{}, 'bimodal':{}, 'uniform':{}}
		min_sar = []
		lowest_sar = [999999.0,'','','']
		for trial in exp_trials:
			for asdf in trial:
				print (asdf)
			for i in all_ideals:
				for j in all_ideals[i]:
					for k in all_ideals[i][j]:
						sar = 0.0
						for l in range(len(trial)):
							exp_val = trial[l]
							sar += np.abs(exp_val -
all_ideals[i][j][k][l])
						if not math.isnan(sar):
							min_sar.append([sar,i,j,k])

		min_sar.sort()
		with open('categorization_results.csv', 'w', newline='') as csvfile:
			spamwriter = csv.writer(csvfile, delimiter=',')
			spamwriter.writerow(['sar','type','mean','sigma'])
			for row in min_sar:
				spamwriter.writerow(row)
		self.message.set('Top 3 results:\n'+'type: '+str(min_sar[0][1])+', mean:
'+str(min_sar[0][2])+', sigma: '+str(min_sar[0][3])+', sar: '+str(min_sar[0][0])+'\n'+'type:
'+str(min_sar[1][1])+', mean: '+str(min_sar[1][2])+', sigma: '+str(min_sar[1][3])+', sar:
'+str(min_sar[1][0])+'\n'+'type: '+str(min_sar[2][1])+', mean: '+str(min_sar[2][2])+',
sigma: '+str(min_sar[2][3])+', sar: '+str(min_sar[2][0]))

	def selectfile(self):

		file_opt = options = {}
		options['defaultextension'] = '.csv'
		options['filetypes'] = [('CSV (Comma Separated Value)', '.csv'), ('all files',
'.*')]
		options['initialdir'] = 'C:\\'
		options['parent'] = self
		options['title'] = 'Open file...'
		file = tkfd.askopenfilename(**file_opt)

		return file

	def selectfolder(self):
		"""
```

```
        Returns the filepath of selected folder. Requires self Tk frame as
argument.
        """

        folder = tkfd.askdirectory(initialdir = 'C:\\', parent = self, title = 'Select a
folder', mustexist = True)

        return folder

    def extractdata(self, file_path):
        """
        Returns data from a file as a list of lines. Works with csv, xls, and xlxs.
        """
        data = []
        file_ext = file_path.split(os.extsep).pop()

        if file_ext == 'csv':
            with open(file_path, 'r') as f:
                reader = csv.reader(f)
                for row in reader:
                    x,y = row
                    data.append(row)
        elif file_ext == 'xls':
            print ('must be csv')
        elif file_ext == 'xlsx':
            print ('must be csv')
        else:
            print ('must be csv')

        return data

    def select_folder(self):
        file = self.selectfile()
        data = self.extractdata(file)
        binsize = int(self.BIN_SIZE.get())
        ptnum = int(self.PTNUM.get())
        results = self.categorization_algorithm(data, binsize, ptnum)

root = tk.Tk()
root.title(string = 'Categorization algorithm')
root.geometry('400x150')
red = '#FF0000'
app = categorization(master = root)
app.mainloop()
```

```
root.destroy()
```

APPENDIX D

PYTHON CODE TO GENERATE DATA IN FIGURE 22 - VARYING THE COMPRESSION AND ROTATION OF THE RADIAL SYMMETRY SHOWS MINIMAL EFFECTS AT BIOLOGICAL CONDITIONS

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 20 10:52:38 2017

@author: Andrew Ruba

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import math
import random
import csv
import os
from numpy.random import choice
import numpy as np
from scipy.optimize import curve_fit
import time
import matplotlib.pyplot as plt
from scipy import stats

## below arrays are for saving simulation data for statistical analysis
global gausslist
gausslist = []
global bimodallist
bimodallist = []
global bimodalmean
bimodalmean = []
```

```
global bimodalsd
bimodalsd = []
global bimodalheight
bimodalheight = []
global bimodalauc
bimodalauc = []

def sim(gui, PTNUM, RADIUS, PREC, ITER, BINSIZE, PERCERROR, A, B,
ROTATION):
	def simulation(num_points, radius, dr, ss, mm, aa, bb, rotation):

		def area_fn(X):
			X = float(X)
			A = -(dr**2)*np.pi
			B = dr*2*np.pi
			return X*B+A

		def gauss_fn(x, s, m):
			a = area_fn(m)
			x = float(x)
			s = float(s)
			m = float(m)
			return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

		def combine(x):
			s = ss
			m = mm
			return (area_fn(x) * gauss_fn(x, s, m))

		##starting with perfect x,y and adding error
		xydata = []
		mm = mm + 0.00001
		while len(xydata) < num_points:
			theta = np.random.random()*360.0
			## precision distribution sampling
#		ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#		ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
			y_prec = np.random.normal(0.0, ss)
			z_prec = np.random.normal(0.0, ss)
			yy = aa*np.cos(theta)
			zz = bb*np.sin(theta)
			yyy =
yy*np.cos(np.radians(rotation))+zz*np.sin(np.radians(rotation))
```

```
                zzz =
-yy*np.sin(np.radians(rotation))+zz*np.cos(np.radians(rotation))
                xydata.append((yyy+y_prec, zzz+z_prec))

            def gen_matrix(r, d_r):

                ##'be' is short for bin edges
                if r%d_r > 0:
                    be = range(0, r+r%d_r, d_r)
                else:
                    be = range(0, r+d_r, d_r)

                matrix = []
                for i in range(len(be)-1):
                    matrix.append([])

                x = 0
                for i in range(len(matrix)):
                    for j in range(x):
                        matrix[i].append(0)
                    x += 1

                ##generate areas of sections closest to x axis
                for i in range(len(matrix)):
                    theta =
np.arccos(float(be[len(be)-2-i])/float(be[len(be)-1-i]))
                    arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
                    tri_area = 0.5 * float(be[len(be)-2-i]) * (np.sin(theta) *
float(be[len(be)-1-i]))
                    matrix[i].append(4 * (arc_area - tri_area))

                ##skipping factor
                x = 2
                ##generate areas of layers going further out from x axis
                while len(matrix[0]) < len(matrix):
                    for i in range(len(matrix) - len(matrix[0])):
                        num = 0
                        for j in range(len(matrix)):
                            for k in range(len(matrix[i]) + 1):
                                if j == i and k < len(matrix[i]):
                                    num += matrix[j][k]
                                elif j > i:
                                    num += matrix[j][k]
```

```
                                        theta =
np.arccos(float(be[len(be)-1-x-i])/float(be[len(be)-1-i]))
                                        arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
                                        tri_area = 0.5 * float(be[len(be)-1-x-i]) *
(np.sin(theta) * float(be[len(be)-1-i]))
                                        matrix[i].append(4 * (arc_area - tri_area) - num)
                                x += 1

                        return matrix

                def smoothdata(data, r, d_r):
                        """smoothds data with 3 moving window and takes abs value
average"""
                        smooth_data = []
                        r += 1

                        ##comment out for smoothing
                        smooth_data = []
                        for i in range(len(data)):
                                smooth_data.append(data[i])

                        ##adds + and - bins
                        final_smooth_data = []
                        for i in range(int(r/d_r)):
                                final_smooth_data.append(smooth_data[i] +
smooth_data[len(smooth_data)-1-i])

                        return list(reversed(final_smooth_data))

                def deconvolution(hv, be, r, d_r):
                        """hv = hist_values, be = bin_edges"""

                        density = []
                        matrix = gen_matrix(r, d_r)

                        while len(hv) > len(matrix):
                                hv.pop()
                        while len(matrix) > len(hv):
                                matrix.pop()

                        rev_hv = list(reversed(hv))
                        x = 0
                        for i in range(len(rev_hv)):
```

```
                ##calculate how much to subtract from bin
                density_sub = 0
                y = 0
                for j in range(x):
                    density_sub += density[y] * matrix[j][i]
                    y += 1

                ##calculate final bin value
                density.append((rev_hv[i] - density_sub) / matrix[i][i])
                x += 1

            unrev_hv = list(reversed(density))

            smooth_data = []
            for i in range(len(unrev_hv)):
                if i == 0 or i == (len(unrev_hv) - 1):
                    smooth_data.append(unrev_hv[i])
                else:
                    smooth_data.append(np.average([unrev_hv[i-1],
unrev_hv[i], unrev_hv[i+1]]))

            return unrev_hv, smooth_data, hv

        def make_hist(data, r, d_r):

            hist_values, bin_edges = np.histogram(data, bins = 2 * int(r/d_r),
range = (-r, r))

            new_bin_edges = []
            for i in bin_edges:
                if i >= 0:
                    new_bin_edges.append(i)

            new_hist_values = smoothdata(hist_values, r, d_r)

            return new_hist_values, new_bin_edges

        def csv_read(path):
            with open(path, 'rb') as csvfile:
                reader = csv.reader(csvfile, delimiter = ',')
                holdlist = []
                for row in reader:
                    holdlist.append(float(row[1]))
            return holdlist
```

```
		jkl = []
		for y,z in xydata:
			jkl.append(y)

		radius = int(np.floor(radius/dr))*dr

		if num_points == PTNUM + 1:
			## decide the proper bin size
			minbinsize = 2
			binsizes = []
			binsizesdata = [[] for variable in range(1, int(PREC)+1)]
			gui.message.set('0% done calculating ideal bin size...')
			gui.update()
			for binoptimization in range(10):
				for binsize in range(1, int(PREC)+1):
					if binsize >= minbinsize:
						error = 0
						# print ('binsize ' + str(binsize))
						jkl = []
						mm = mm + 0.00001
						while len(jkl) < num_points-1:
							theta = np.random.random()*360.0
							## precision distribution sampling
#							ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#							ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
							y_prec = np.random.normal(0.0, ss)

jkl.append(mm*np.cos(theta)+y_prec)
						a,b = make_hist(jkl, radius, binsize)
						final_unsmooth, final_smooth, final_2d =
deconvolution(a, b, radius, binsize)
						holdlist = []
						addZero = False
						for val in list(reversed(final_unsmooth)):
							if not addZero:
								if val >= 0.0:
									holdlist.append(val)
								else:
									addZero = True
									holdlist.append(0.0)
							else:
								holdlist.append(0.0)
						final_unsmooth = list(reversed(holdlist))
```

```
						##rescale ideal data
						matrix = gen_matrix(radius, binsize)
						newmatrix = []
						for i in matrix:
							newmatrix.append(list(reversed(i)))
						matrix = list(reversed(newmatrix))
						# print (a)
						# print (final_unsmooth)
						while len(a) > len(matrix):
							a.pop()
						while len(matrix) > len(a):
							matrix.pop()

						for ncol in range(len(matrix[0])):
							binsub = 0.0
							for mcol in range(len(matrix)):
								binsub +=
float(matrix[mcol][ncol]*final_unsmooth[mcol])
							try:
								if a[ncol] != 0.0:
									# print (binsub)
									error +=
np.square(a[ncol] - binsub) / a[ncol]
							except:
								pass

						popped = a.pop()
						while popped == 0:
							popped = a.pop()
						binsizesdata[binsize-1].append((error,
len(a)+1,1-stats.chi2.cdf(error, len(a)+1),binsize))
					else:

binsizesdata[binsize-1].append((1000000.0,1,0.0,binsize))
					gui.message.set(str((binoptimization*10) + 10) + '
% done calculating ideal bin size...')
					gui.update()

			finalbinsizes = []
			for bintrial in range(len(binsizesdata)):
				errhold = []
				dfhold = []
				pvalhold = []
				binhold = []
```

```
                for trial in range(len(binsizesdata[bintrial])):
                    chisq, df, pval, binsize = binsizesdata[bintrial][trial]
                    errhold.append(chisq)
                    dfhold.append(df)
                    pvalhold.append(pval)
                    binhold.append(binsize)
                chisq = np.average(errhold)
                df = np.round(np.average(dfhold))
                pval = 1-stats.chi2.cdf(chisq,df)
                binsize = binhold[0]
                finalbinsizes.append((chisq,df,pval,binsize))
            # print (finalbinsizes)
            for binsizedata in finalbinsizes:
                chisq, df, pval, binsize = binsizedata
                if pval >= 0.95:
                    dr = binsize
                    break
                else:
                    dr = int(PREC)

        a,b = make_hist(jkl, radius, dr)
        final = deconvolution(a,b,radius,dr)

        if num_points != PTNUM + 1:
            def gauss_fn(x, a, s, m):
                return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

            def bimodal(x,mu1,sigma1,A1,mu2,sigma2,A2):
                return gauss_fn(x, A1, sigma1, mu1)+gauss_fn(x, A2,
sigma2, mu2)

            try:
                guess = [np.max(final[0]), ss, mm]
                tempbins = list(range(int(dr/2), radius+int(dr/2), dr))
                tempdensity = final[0]
                holdlist = []
                addZero = False
                for val in list(reversed(tempdensity)):
                    if not addZero:
                        if val >= 0.0:
                            holdlist.append(val)
                        else:
                            addZero = True
                            holdlist.append(0.0)
```

```
		else:
			holdlist.append(0.0)
	tempdensity = list(reversed(holdlist))
	while len(tempdensity) > len(tempbins):
		tempdensity.pop()
	while len(tempbins) > len(tempdensity):
		tempbins.pop()
	revtempbins = list(np.negative(list(reversed(tempbins))))
	revtempdensity = list(reversed(tempdensity))
	bins = revtempbins + tempbins
	density = revtempdensity + tempdensity
	params, var = curve_fit(gauss_fn, bins, density, p0 = guess)
	params_gauss = np.abs(params)
	## computes 1 SD errors
	var_gauss = np.sqrt(np.diag(var))

	def frange(beg, end, step):
		f_range = []
		while beg < end - (step/2.0):
			f_range.append(beg)
			beg += step
		return f_range

	guess = [-mm, ss, np.max(final[0]), mm, ss,
np.max(final[0])]

	tempbins = frange(dr/2.0, radius, dr)

	tempdensity = final[0]
	holdlist = []
	addZero = False
	for val in list(reversed(tempdensity)):
		if not addZero:
			if val >= 0.0:
				holdlist.append(val)
			else:
				addZero = True
				holdlist.append(0.0)
		else:
			holdlist.append(0.0)
	tempdensity = list(reversed(holdlist))
	while len(tempdensity) > len(tempbins):
		tempdensity.pop()
	while len(tempbins) > len(tempdensity):
```

```
                tempbins.pop()
            revtempbins = list(np.negative(list(reversed(tempbins))))
            revtempdensity = list(reversed(tempdensity))
            bins = revtempbins + tempbins
            density = revtempdensity + tempdensity
            params, var = curve_fit(bimodal, bins, density, p0 = guess)
            params = np.abs(params)
            ## computes 1 SD errors
            var = np.sqrt(np.diag(var))
            ## average paramters
            stdev = np.average((params[1], params[4]))
            mean = np.average((params[0], params[3]))
            height = np.average((params[2], params[5]))
            stdev_e = np.average((var[1], var[4]))
            mean_e = np.average((var[0], var[3]))
            height_e = np.average((var[2], var[5]))
            params_bimodal = [height, stdev, mean]
            var_bimodal = [height_e, stdev_e, mean_e]

            ## uncomment following for comparing central vs.
peripheral peak fitting errors
#           bimodalmean.append(params_gauss[0])
            bimodalmean.append(mean)
#           bimodalmean.append(tempdensity)
            bimodalsd.append(stdev)
            bimodalheight.append(height)
            auc = 0.0
            step = mean - 5.0*stdev
            while step < mean + 5.0*stdev:
                auc+=0.01*gauss_fn(step,height,stdev,mean)
                step += 0.01
            bimodalauc.append(auc)
#           bimodallist.append(var_bimodal[1])
            gausslist.append(var_gauss[1])
        #   if np.sum(var_bimodal) < np.sum(var_gauss):
            params = params_bimodal
            var = var_bimodal
        #   else:
#           params = params_gauss
#           var = var_gauss
        except RuntimeError:
            params = []
            var = []
        hist_mids = []
```

```
                    for i in range(len(b)-1):
                        hist_mids.append(np.average((b[i],b[i+1])))
                    norm_values = []
                    for i in final[1]:
                        norm_values.append(i/np.max(final[1]))
                    return params, var, norm_values, hist_mids, dr

                else:
                    return dr

        pt_min = PTNUM
        pt_max = PTNUM
        rt_min = RADIUS
        rt_max = RADIUS
        prec_min = PREC
        prec_max = PREC
        iterations = ITER
        PREC = float(PREC)
        one_diff = []
        perc_err = PERCERROR*0.01

        def roundup(x):
            val = int(math.ceil(x / 10.0)) * 10
            if val >= 30:
                return val
            else:
                return 30
        ptlist = range(pt_min, pt_max+100, 100)
        for pt in ptlist:
            for rt in range(rt_min, rt_max+1, 1):
                for prec in range(prec_min, prec_max+1, 1):
                    prec = prec+0.000001
                    xrng = roundup(rt + prec*5.0)
                    # DR = simulation(pt+1, xrng, BINSIZE, float(prec),
float(rt))
                    ## uncomment below to manually set bin size
                    DR = PTNUM
                    # print ('ideal bin size: '+ str(DR))
                    p_ab, v, d, h_m, DR = simulation(1000000, xrng, DR,
float(prec), float(rt), A, B, ROTATION)
                    p_normal, v, d, h_m, DR = simulation(1000000, xrng, DR,
float(prec), float(rt), RADIUS, RADIUS, 0)
                    # print (p)
                    a, s, m_ab = p_ab
```

```
            a, s, m_normal = p_normal
    return '%.3f'%(m_ab-m_normal)
```

APPENDIX E

PYTHON CODE TO GENERATE DATA IN FIGURE 23 - VARYING THE LABELING RATIO AND ROTATION SHOWS MINIMAL EFFECTS BEYOND HALF LABELING EFFICIENCY

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 20 10:52:38 2017

@author: Andrew Ruba

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import math
import random
import csv
import os
from numpy.random import choice
import numpy as np
from scipy.optimize import curve_fit
import time
import matplotlib.pyplot as plt
from scipy import stats

## below arrays are for saving simulation data for statistical analysis
global gausslist
gausslist = []
global bimodallist
bimodallist = []
global bimodalmean
bimodalmean = []
```

```
global bimodalsd
bimodalsd = []
global bimodalheight
bimodalheight = []
global bimodalauc
bimodalauc = []

def sim(gui, PTNUM, RADIUS, PREC, ITER, BINSIZE, PERCERROR, A,
ROTATION):
	def simulation(num_points, radius, dr, ss, mm, aa, rotation):

		def area_fn(X):
			X = float(X)
			A = -(dr**2)*np.pi
			B = dr*2*np.pi
			return X*B+A

		def gauss_fn(x, s, m):
			a = area_fn(m)
			x = float(x)
			s = float(s)
			m = float(m)
			return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

		def combine(x):
			s = ss
			m = mm
			return (area_fn(x) * gauss_fn(x, s, m))

		##starting with perfect x,y and adding error
		xydata = []
		mm = mm + 0.00001
		while len(xydata) < num_points:
			theta =
np.radians((np.random.random()*360.0*aa)-0.5*(1.0-aa)*360.0)
			## precision distribution sampling
#		ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#		ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
			y_prec = np.random.normal(0.0, ss)
			z_prec = np.random.normal(0.0, ss)
			yy = mm*np.cos(theta)
			zz = mm*np.sin(theta)
			yyy =
yy*np.cos(np.radians(rotation))+zz*np.sin(np.radians(rotation))
```

```
                        zzz =
-yy*np.sin(np.radians(rotation))+zz*np.cos(np.radians(rotation))
                        xydata.append((yyy+y_prec, zzz+z_prec))

                def gen_matrix(r, d_r):

                        ##'be' is short for bin edges
                        if r%d_r > 0:
                                be = range(0, r+r%d_r, d_r)
                        else:
                                be = range(0, r+d_r, d_r)

                        matrix = []
                        for i in range(len(be)-1):
                                matrix.append([])

                        x = 0
                        for i in range(len(matrix)):
                                for j in range(x):
                                        matrix[i].append(0)
                                x += 1

                        ##generate areas of sections closest to x axis
                        for i in range(len(matrix)):
                                theta =
np.arccos(float(be[len(be)-2-i])/float(be[len(be)-1-i]))
                                arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
                                tri_area = 0.5 * float(be[len(be)-2-i]) * (np.sin(theta) *
float(be[len(be)-1-i]))
                                matrix[i].append(4 * (arc_area - tri_area))

                        ##skipping factor
                        x = 2
                        ##generate areas of layers going further out from x axis
                        while len(matrix[0]) < len(matrix):
                                for i in range(len(matrix) - len(matrix[0])):
                                        num = 0
                                        for j in range(len(matrix)):
                                                for k in range(len(matrix[i]) + 1):
                                                        if j == i and k < len(matrix[i]):
                                                                num += matrix[j][k]
                                                        elif j > i:
                                                                num += matrix[j][k]
```

```
					theta =
np.arccos(float(be[len(be)-1-x-i])/float(be[len(be)-1-i]))
					arc_area = (theta/(2*np.pi)) * np.pi *
float(be[len(be)-1-i])**2
					tri_area = 0.5 * float(be[len(be)-1-x-i]) *
(np.sin(theta) * float(be[len(be)-1-i]))
					matrix[i].append(4 * (arc_area - tri_area) - num)
				x += 1

			return matrix

		def smoothdata(data, r, d_r):
			"""smoothds data with 3 moving window and takes abs value
average"""
			smooth_data = []
			r += 1

			##comment out for smoothing
			smooth_data = []
			for i in range(len(data)):
				smooth_data.append(data[i])

			##adds + and - bins
			final_smooth_data = []
			for i in range(int(r/d_r)):
				final_smooth_data.append(smooth_data[i] +
smooth_data[len(smooth_data)-1-i])

			return list(reversed(final_smooth_data))

		def deconvolution(hv, be, r, d_r):
			"""hv = hist_values, be = bin_edges"""

			density = []
			matrix = gen_matrix(r, d_r)

			while len(hv) > len(matrix):
				hv.pop()
			while len(matrix) > len(hv):
				matrix.pop()

			rev_hv = list(reversed(hv))
			x = 0
			for i in range(len(rev_hv)):
```

```
            ##calculate how much to subtract from bin
            density_sub = 0
            y = 0
            for j in range(x):
                density_sub += density[y] * matrix[j][i]
                y += 1

            ##calculate final bin value
            density.append((rev_hv[i] - density_sub) / matrix[i][i])
            x += 1

        unrev_hv = list(reversed(density))

        smooth_data = []
        for i in range(len(unrev_hv)):
            if i == 0 or i == (len(unrev_hv) - 1):
                smooth_data.append(unrev_hv[i])
            else:
                smooth_data.append(np.average([unrev_hv[i-1],
unrev_hv[i], unrev_hv[i+1]]))

        return unrev_hv, smooth_data, hv

    def make_hist(data, r, d_r):

        hist_values, bin_edges = np.histogram(data, bins = 2 * int(r/d_r),
range = (-r, r))

        new_bin_edges = []
        for i in bin_edges:
            if i >= 0:
                new_bin_edges.append(i)

        new_hist_values = smoothdata(hist_values, r, d_r)

        return new_hist_values, new_bin_edges

    def csv_read(path):
        with open(path, 'rb') as csvfile:
            reader = csv.reader(csvfile, delimiter = ',')
            holdlist = []
            for row in reader:
                holdlist.append(float(row[1]))
        return holdlist
```

```
		jkl = []
		for y,z in xydata:
			jkl.append(y)

		radius = int(np.floor(radius/dr))*dr

		if num_points == PTNUM + 1:
			## decide the proper bin size
			minbinsize = 2
			binsizes = []
			binsizesdata = [[] for variable in range(1, int(PREC)+1)]
			gui.message.set('0% done calculating ideal bin size...')
			gui.update()
			for binoptimization in range(10):
				for binsize in range(1, int(PREC)+1):
					if binsize >= minbinsize:
						error = 0
						# print ('binsize ' + str(binsize))
						jkl = []
						mm = mm + 0.00001
						while len(jkl) < num_points-1:
							theta = np.random.random()*360.0
							## precision distribution sampling
#							ss = choice([3,5,7,9], p=[0.1475,0.2775,0.3075,0.2675])
#							ss = choice([4.5,5.5,6.5,7.5,8.5,9.5], p=[0.02,0.05,0.07,0.11,0.2,0.55])
							y_prec = np.random.normal(0.0, ss)

jkl.append(mm*np.cos(theta)+y_prec)
						a,b = make_hist(jkl, radius, binsize)
						final_unsmooth, final_smooth, final_2d =
deconvolution(a, b, radius, binsize)
						holdlist = []
						addZero = False
						for val in list(reversed(final_unsmooth)):
							if not addZero:
								if val >= 0.0:
									holdlist.append(val)
								else:
									addZero = True
									holdlist.append(0.0)
							else:
								holdlist.append(0.0)
						final_unsmooth = list(reversed(holdlist))
```

```
                                        ##rescale ideal data
                                        matrix = gen_matrix(radius, binsize)
                                        newmatrix = []
                                        for i in matrix:
                                            newmatrix.append(list(reversed(i)))
                                        matrix = list(reversed(newmatrix))
                                        # print (a)
                                        # print (final_unsmooth)
                                        while len(a) > len(matrix):
                                            a.pop()
                                        while len(matrix) > len(a):
                                            matrix.pop()

                                        for ncol in range(len(matrix[0])):
                                            binsub = 0.0
                                            for mcol in range(len(matrix)):
                                                binsub +=
float(matrix[mcol][ncol]*final_unsmooth[mcol])
                                            try:
                                                if a[ncol] != 0.0:
                                                    # print (binsub)
                                                    error +=
np.square(a[ncol] - binsub) / a[ncol]
                                            except:
                                                pass

                                        popped = a.pop()
                                        while popped == 0:
                                            popped = a.pop()
                                        binsizesdata[binsize-1].append((error,
len(a)+1,1-stats.chi2.cdf(error, len(a)+1),binsize))
                                    else:

binsizesdata[binsize-1].append((1000000.0,1,0.0,binsize))
                                    gui.message.set(str((binoptimization*10) + 10) + '
% done calculating ideal bin size...')
                                    gui.update()

                        finalbinsizes = []
                        for bintrial in range(len(binsizesdata)):
                            errhold = []
                            dfhold = []
                            pvalhold = []
                            binhold = []
```

```
                    for trial in range(len(binsizesdata[bintrial])):
                        chisq, df, pval, binsize = binsizesdata[bintrial][trial]
                        errhold.append(chisq)
                        dfhold.append(df)
                        pvalhold.append(pval)
                        binhold.append(binsize)
                    chisq = np.average(errhold)
                    df = np.round(np.average(dfhold))
                    pval = 1-stats.chi2.cdf(chisq,df)
                    binsize = binhold[0]
                    finalbinsizes.append((chisq,df,pval,binsize))
                # print (finalbinsizes)
                for binsizedata in finalbinsizes:
                    chisq, df, pval, binsize = binsizedata
                    if pval >= 0.95:
                        dr = binsize
                        break
                    else:
                        dr = int(PREC)

            a,b = make_hist(jkl, radius, dr)
            final = deconvolution(a,b,radius,dr)

            if num_points != PTNUM + 1:
                def gauss_fn(x, a, s, m):
                    return a*np.e**(-(x-m)**2.0/(2.0*s**2.0))

                def bimodal(x,mu1,sigma1,A1,mu2,sigma2,A2):
                    return gauss_fn(x, A1, sigma1, mu1)+gauss_fn(x, A2,
sigma2, mu2)

                try:
                    guess = [np.max(final[0]), ss, mm]
                    tempbins = list(range(int(dr/2), radius+int(dr/2), dr))
                    tempdensity = final[0]
                    holdlist = []
                    addZero = False
                    for val in list(reversed(tempdensity)):
                        if not addZero:
                            if val >= 0.0:
                                holdlist.append(val)
                            else:
                                addZero = True
                                holdlist.append(0.0)
```

```
        else:
            holdlist.append(0.0)
    tempdensity = list(reversed(holdlist))
    while len(tempdensity) > len(tempbins):
        tempdensity.pop()
    while len(tempbins) > len(tempdensity):
        tempbins.pop()
    revtempbins = list(np.negative(list(reversed(tempbins))))
    revtempdensity = list(reversed(tempdensity))
    bins = revtempbins + tempbins
    density = revtempdensity + tempdensity
    params, var = curve_fit(gauss_fn, bins, density, p0 = guess)
    params_gauss = np.abs(params)
    ## computes 1 SD errors
    var_gauss = np.sqrt(np.diag(var))

    def frange(beg, end, step):
        f_range = []
        while beg < end - (step/2.0):
            f_range.append(beg)
            beg += step
        return f_range

    guess = [-mm, ss, np.max(final[0]), mm, ss,
np.max(final[0])]

    tempbins = frange(dr/2.0, radius, dr)

    tempdensity = final[0]
    holdlist = []
    addZero = False
    for val in list(reversed(tempdensity)):
        if not addZero:
            if val >= 0.0:
                holdlist.append(val)
            else:
                addZero = True
                holdlist.append(0.0)
        else:
            holdlist.append(0.0)
    tempdensity = list(reversed(holdlist))
    while len(tempdensity) > len(tempbins):
        tempdensity.pop()
    while len(tempbins) > len(tempdensity):
```

```
                    tempbins.pop()
                revtempbins = list(np.negative(list(reversed(tempbins))))
                revtempdensity = list(reversed(tempdensity))
                bins = revtempbins + tempbins
                density = revtempdensity + tempdensity
                params, var = curve_fit(bimodal, bins, density, p0 = guess)
                params = np.abs(params)
                ## computes 1 SD errors
                var = np.sqrt(np.diag(var))
                ## average paramters
                stdev = np.average((params[1], params[4]))
                mean = np.average((params[0], params[3]))
                height = np.average((params[2], params[5]))
                stdev_e = np.average((var[1], var[4]))
                mean_e = np.average((var[0], var[3]))
                height_e = np.average((var[2], var[5]))
                params_bimodal = [height, stdev, mean]
                var_bimodal = [height_e, stdev_e, mean_e]

                ## uncomment following for comparing central vs.
peripheral peak fitting errors
#           bimodalmean.append(params_gauss[0])
                bimodalmean.append(mean)
#           bimodalmean.append(tempdensity)
                bimodalsd.append(stdev)
                bimodalheight.append(height)
                auc = 0.0
                step = mean - 5.0*stdev
                while step < mean + 5.0*stdev:
                    auc+=0.01*gauss_fn(step,height,stdev,mean)
                    step += 0.01
                bimodalauc.append(auc)
#           bimodallist.append(var_bimodal[1])
                gausslist.append(var_gauss[1])
        #       if np.sum(var_bimodal) < np.sum(var_gauss):
                params = params_bimodal
                var = var_bimodal
        #       else:
#           params = params_gauss
#           var = var_gauss
            except RuntimeError:
                params = []
                var = []
            hist_mids = []
```

```
				for i in range(len(b)-1):
					hist_mids.append(np.average((b[i],b[i+1])))
				norm_values = []
				for i in final[1]:
					norm_values.append(i/np.max(final[1]))
				return params, var, norm_values, hist_mids, dr

			else:
				return dr

		pt_min = PTNUM
		pt_max = PTNUM
		rt_min = RADIUS
		rt_max = RADIUS
		prec_min = PREC
		prec_max = PREC
		iterations = ITER
		PREC = float(PREC)
		one_diff = []
		perc_err = PERCERROR*0.01


		def roundup(x):
			val = int(math.ceil(x / 10.0)) * 10
			if val >= 30:
				return val
			else:
				return 30

		ptlist = range(pt_min, pt_max+100, 100)
		for pt in ptlist:
			for rt in range(rt_min, rt_max+1, 1):
				for prec in range(prec_min, prec_max+1, 1):
					prec = prec+0.000001
					xrng = roundup(rt + prec*5.0)
					# DR = simulation(pt+1, xrng, BINSIZE, float(prec),
float(rt))
					## uncomment below to manually set bin size
					DR = PTNUM
					# print ('ideal bin size: '+ str(DR))
					p_ab, v, d, h_m, DR = simulation(1000000, xrng, DR,
float(prec), float(rt), A, ROTATION)
					print ('density')
					for i in d:
```

```
                    print (i)
                p_normal, v, d, h_m, DR = simulation(1000000, xrng, DR,
float(prec), float(rt), 1.0, 0)
                # print (p)
                a, s, m_ab = p_ab
                a, s, m_normal = p_normal

    return '%.3f'%(m_ab-m_normal)
```